

Fault Tolerance for Remote Memory Access Programming Models

MACIEJ BESTA, TORSTEN HOEFLER



REMOTE MEMORY ACCESS (RMA) PROGRAMMING

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

Process q

Memory

B

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

Process q

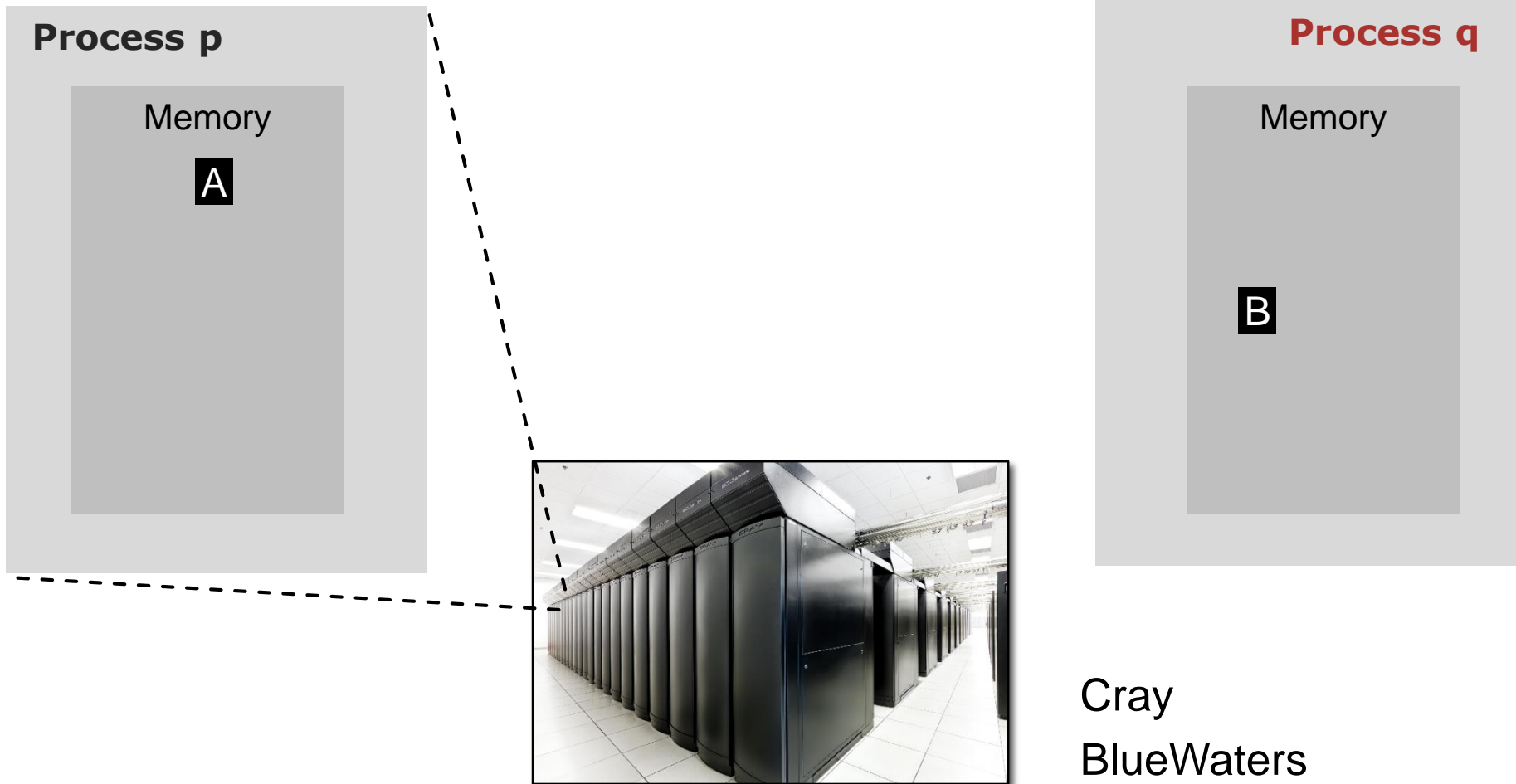
Memory

B

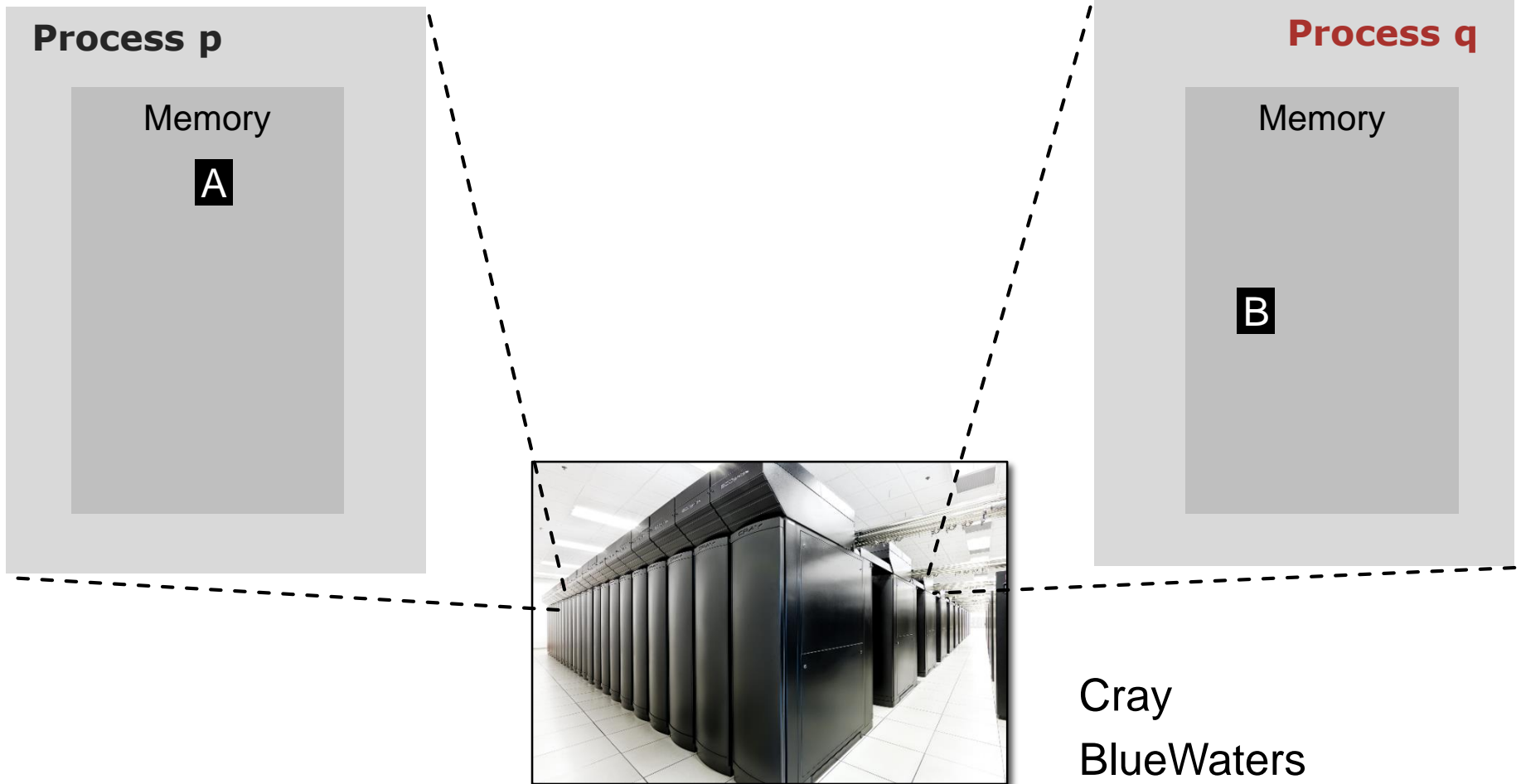


Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



REMOTE MEMORY ACCESS (RMA) PROGRAMMING



REMOTE MEMORY ACCESS (RMA) PROGRAMMING

Process p

Memory

A

Process q

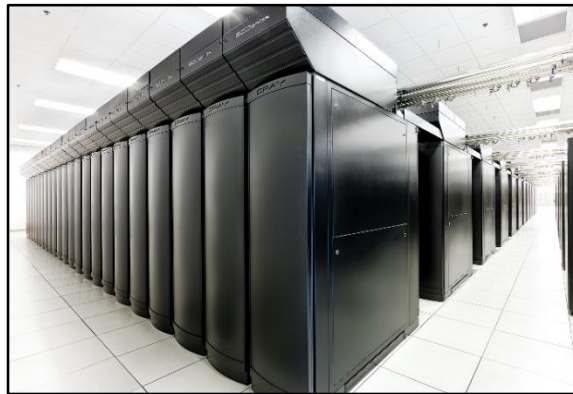
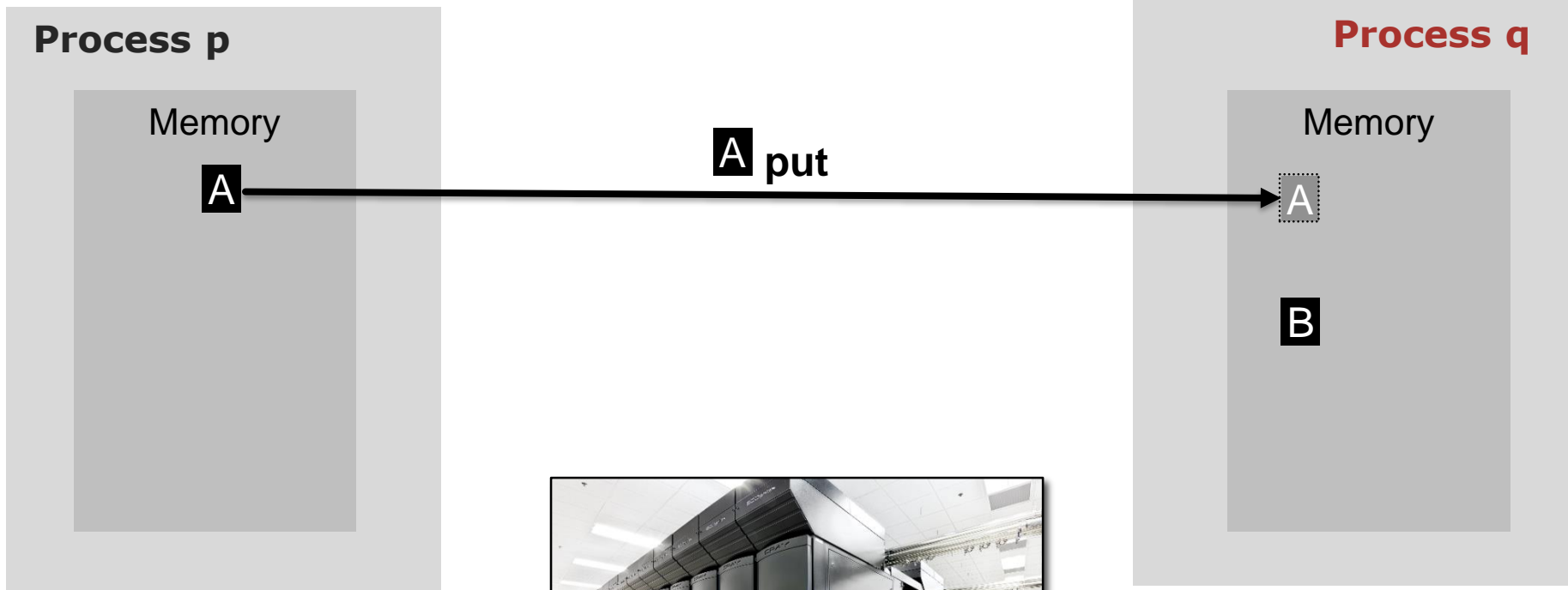
Memory

B



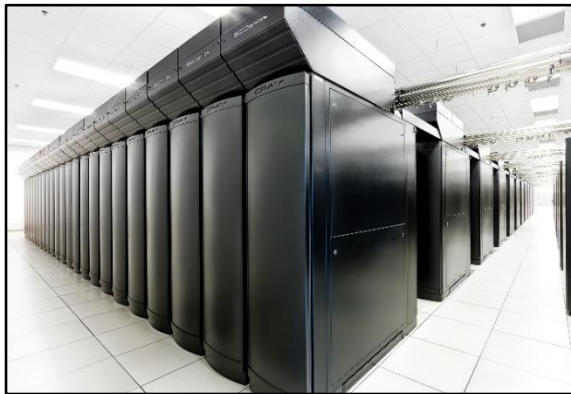
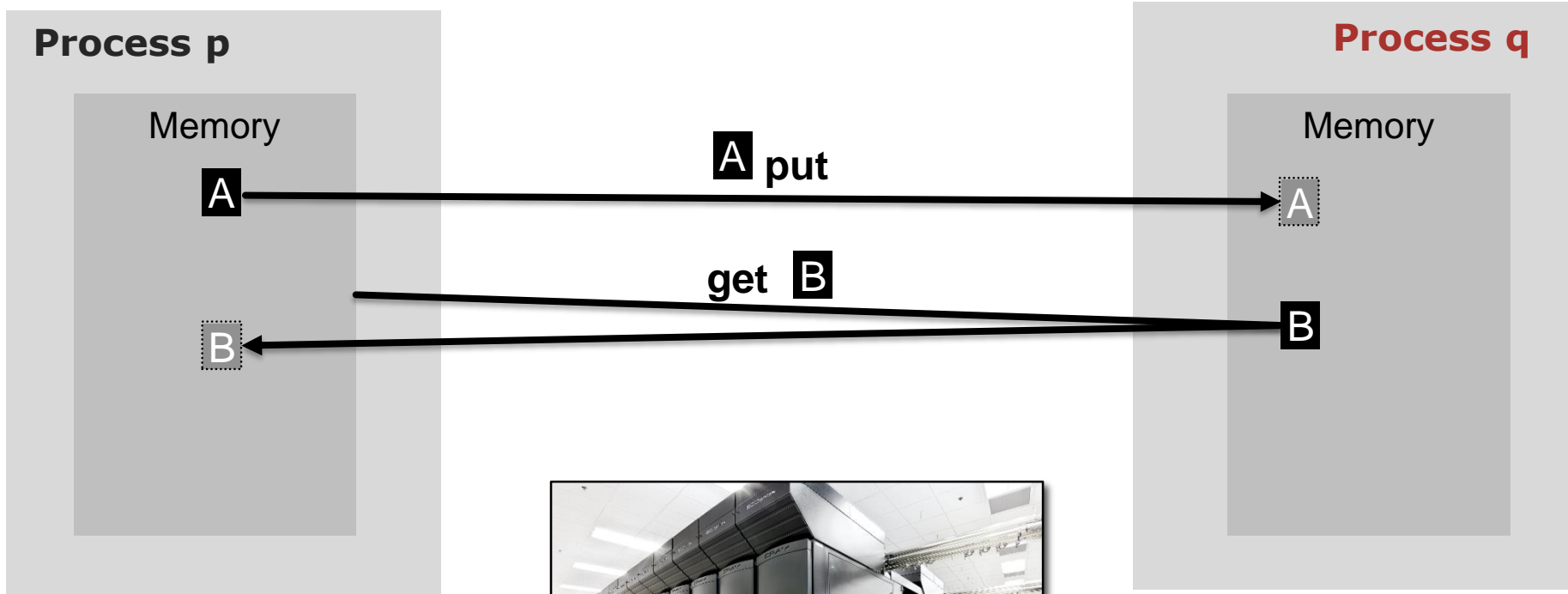
Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



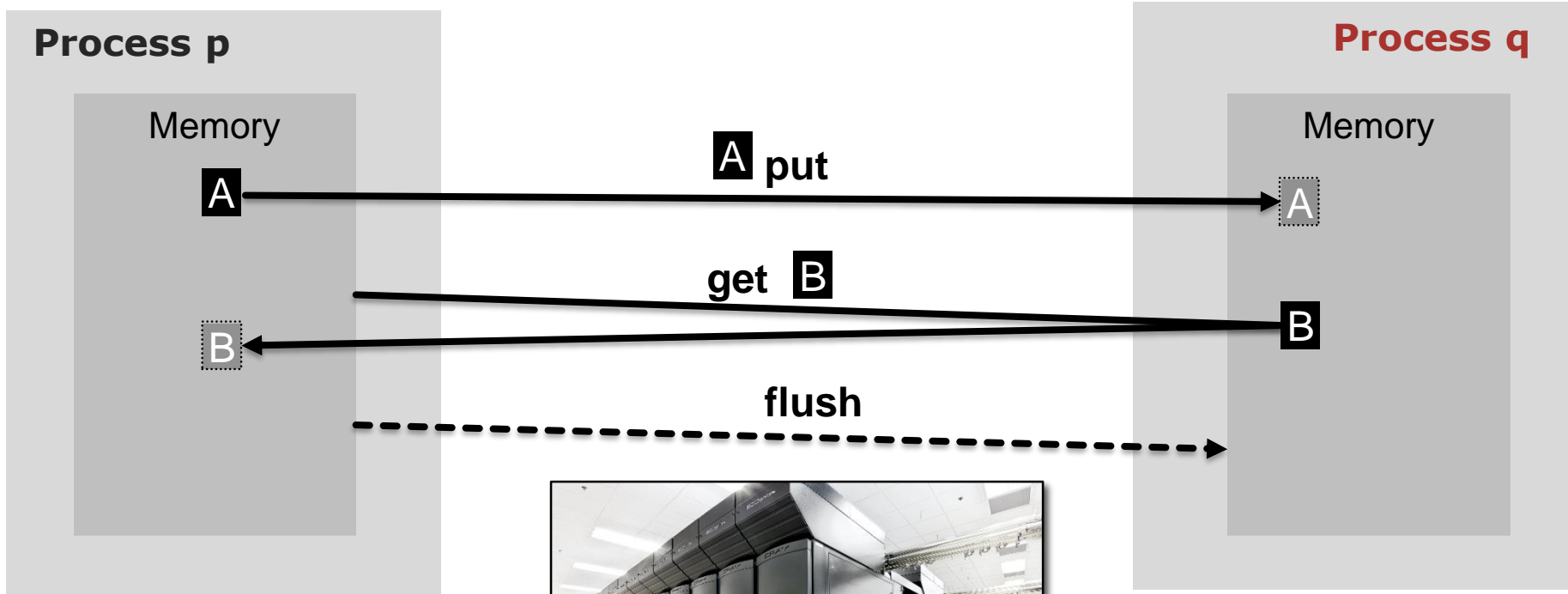
Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



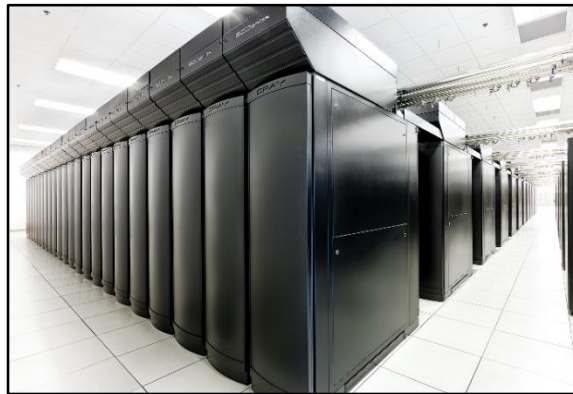
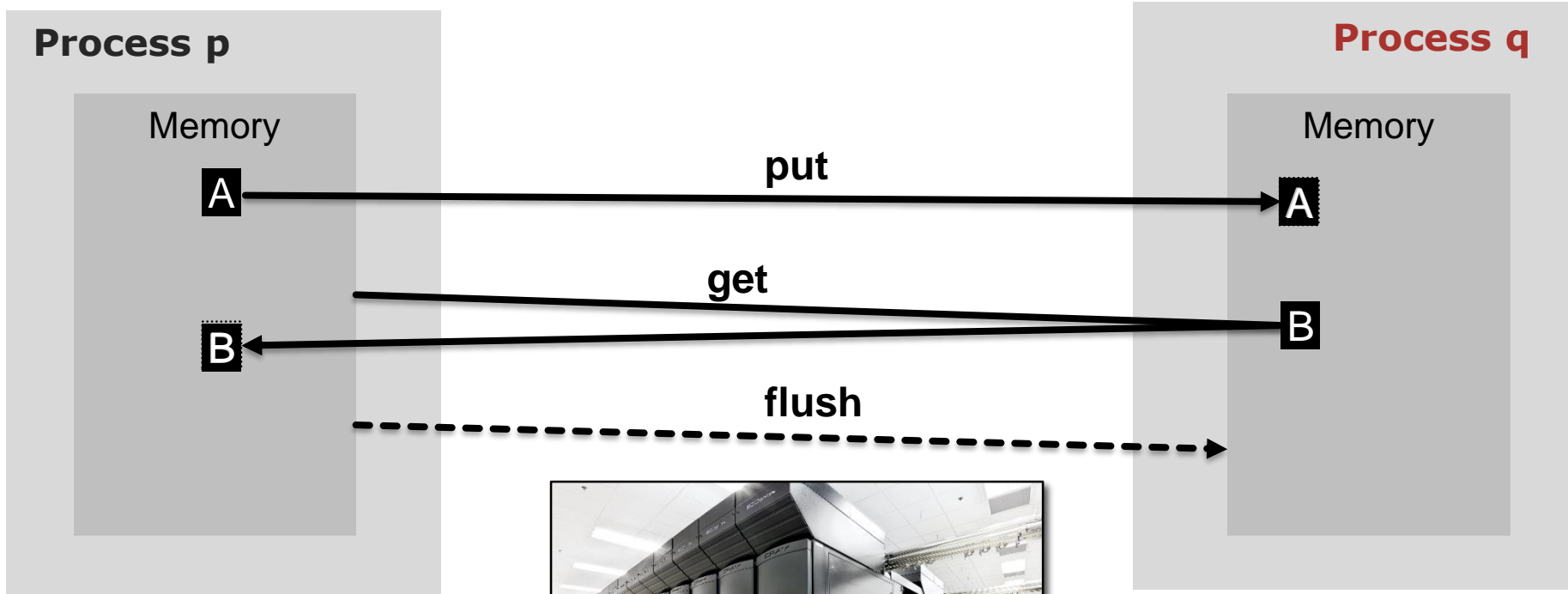
Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING



Cray
BlueWaters

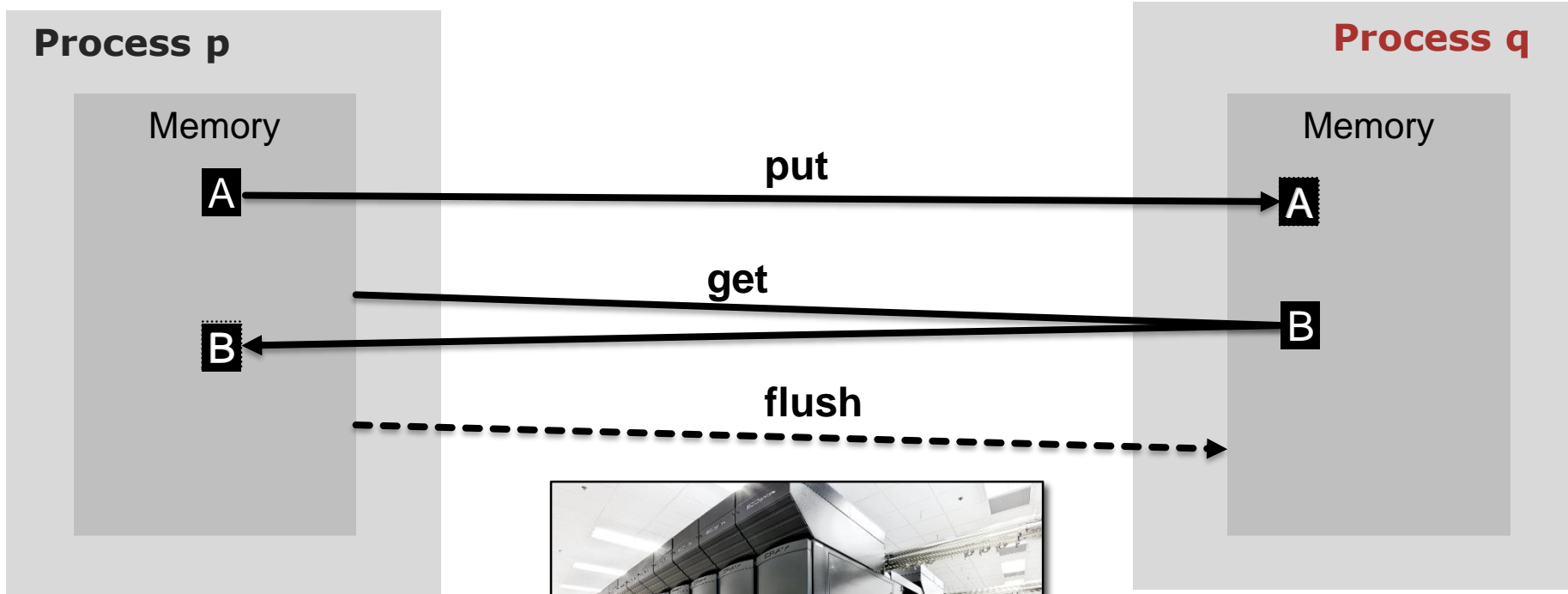
REMOTE MEMORY ACCESS (RMA) PROGRAMMING



Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

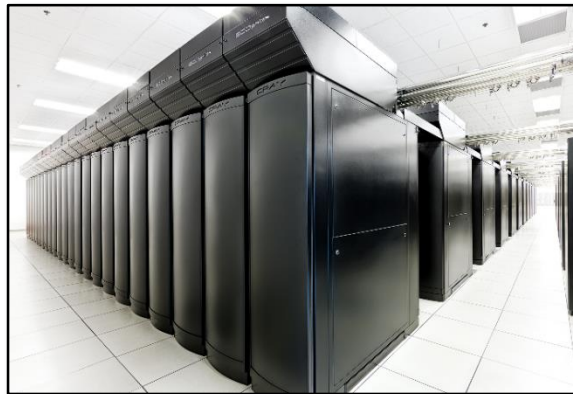
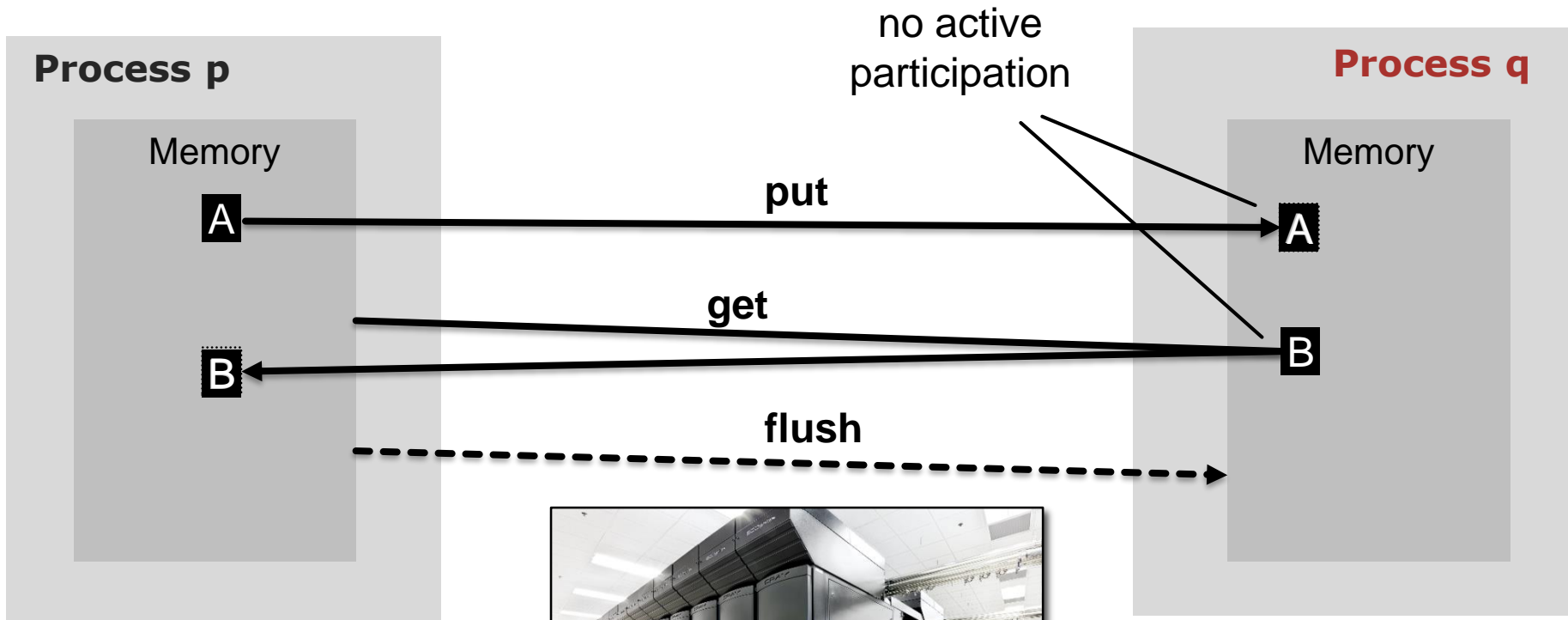
- One-sided communication



Cray
BlueWaters

REMOTE MEMORY ACCESS (RMA) PROGRAMMING

- One-sided communication



Cray
BlueWaters

REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



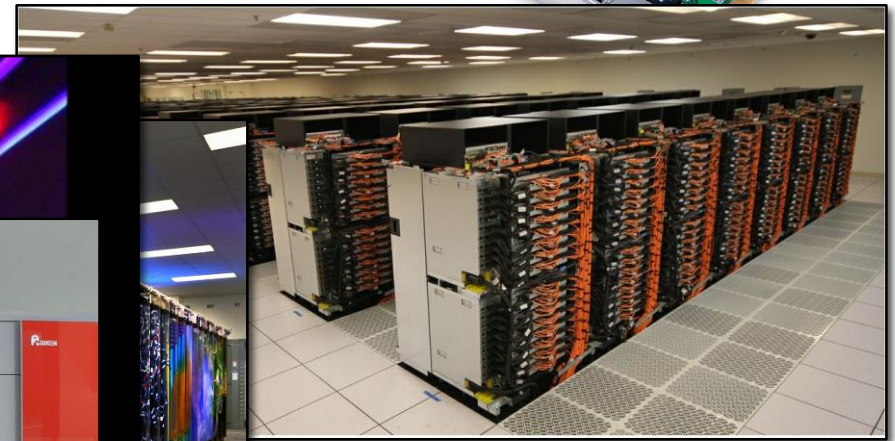
REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



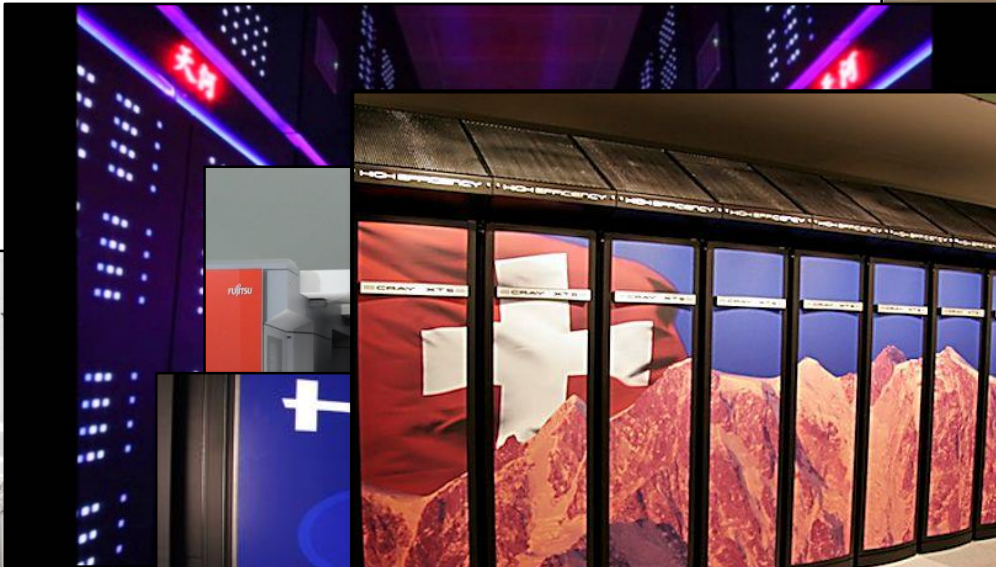
REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA



REMOTE MEMORY ACCESS PROGRAMMING

- Implemented in hardware in NICs in the majority of HPC networks support RDMA

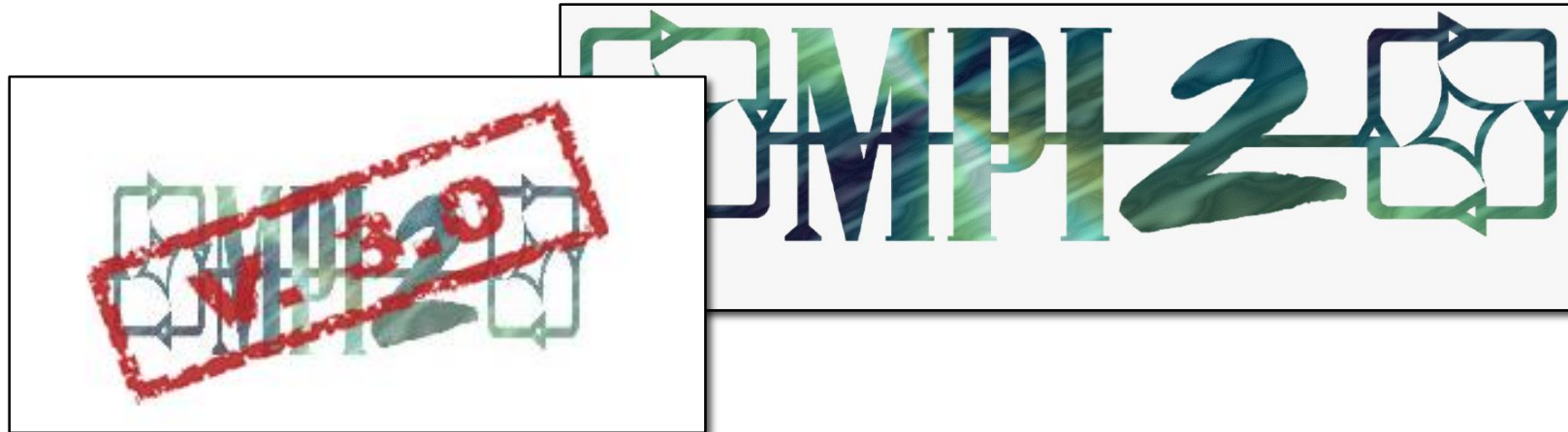


REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages

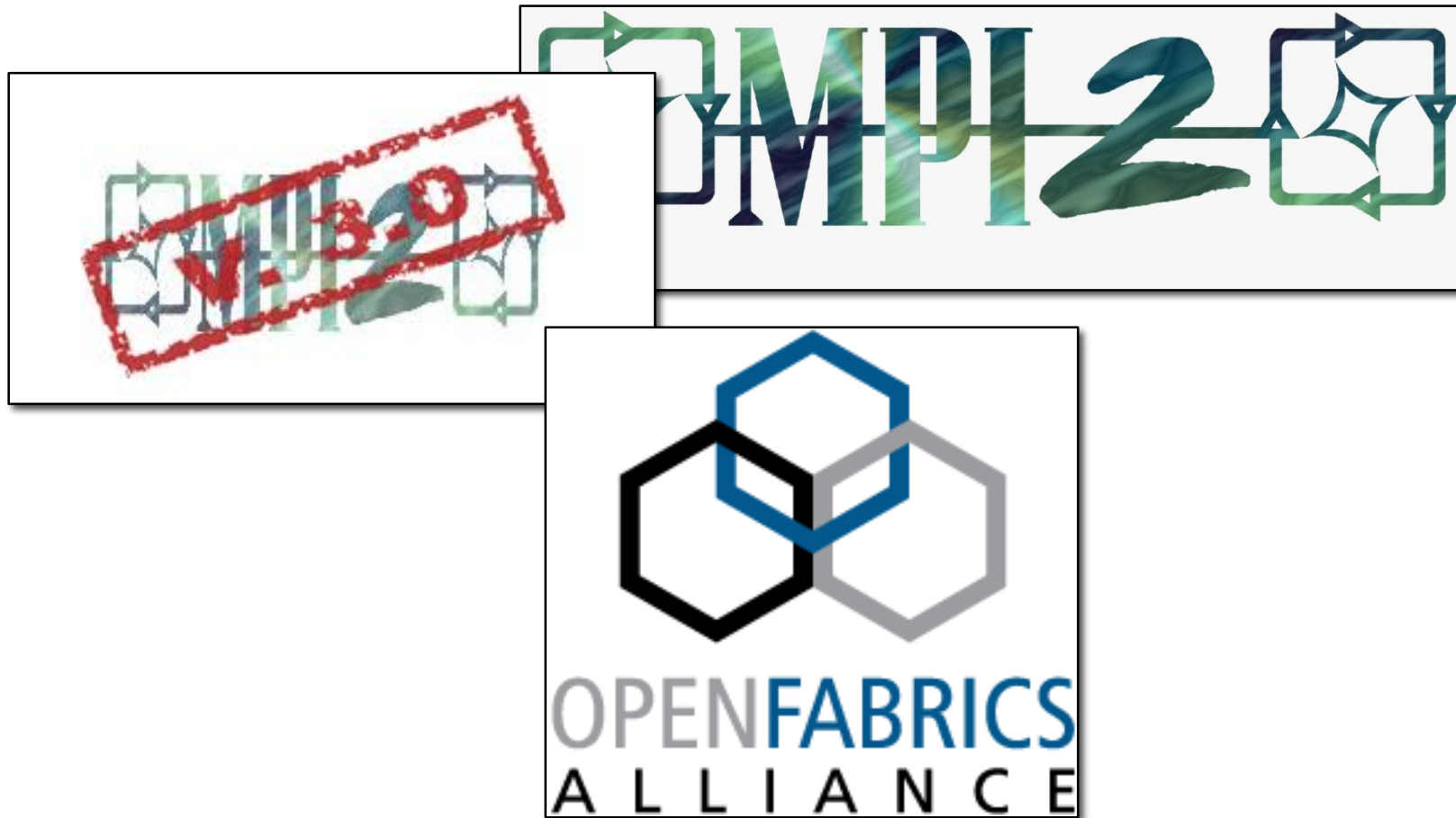
REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages



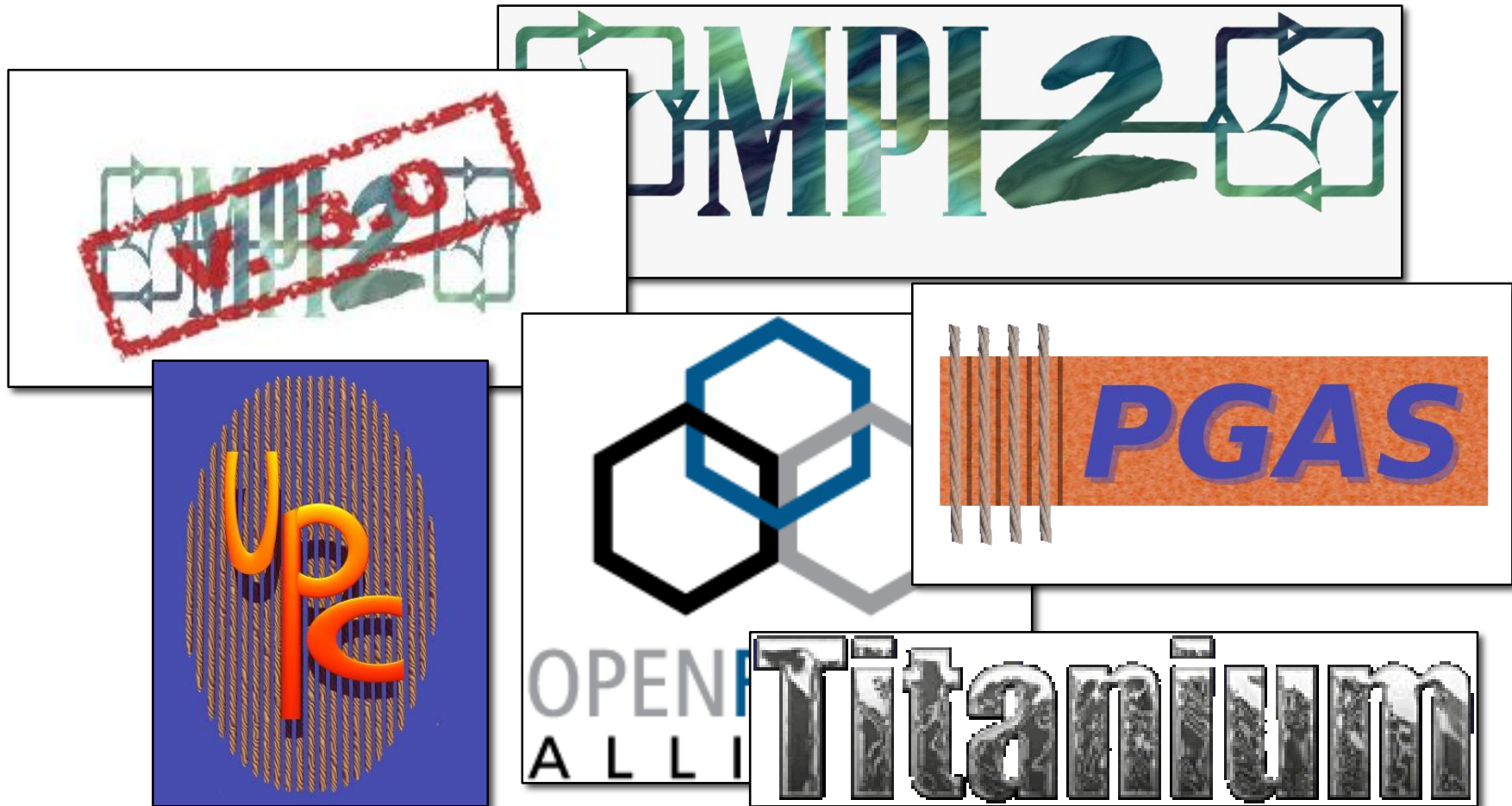
REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages



REMOTE MEMORY ACCESS PROGRAMMING

- Supported by many HPC libraries and languages



REMOTE MEMORY ACCESS PROGRAMMING

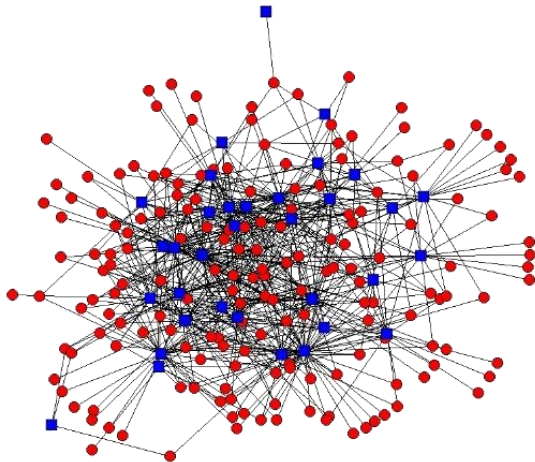
- Enables significant speedups over message passing in many types of applications, e.g.:

[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

REMOTE MEMORY ACCESS PROGRAMMING

- Enables significant speedups over message passing in many types of applications, e.g.:
 - Speedup of ~ 1.5 for communication patterns in graph analytics

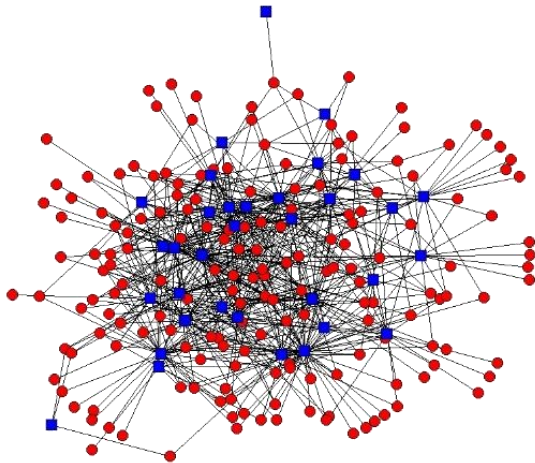
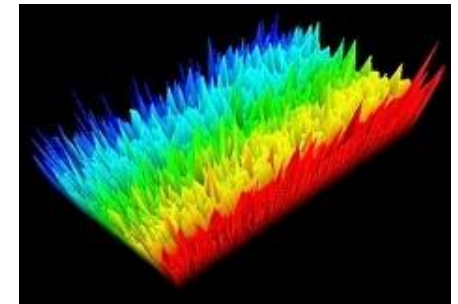


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

REMOTE MEMORY ACCESS PROGRAMMING

- Enables significant speedups over message passing in many types of applications, e.g.:
 - Speedup of ~ 1.5 for communication patterns in graph analytics
 - Speedup of $\sim 1.4-2$ in physics computations



$$\frac{1}{\sqrt{2}} |\text{cat}\rangle + \frac{1}{\sqrt{2}} |\text{dog}\rangle$$

[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

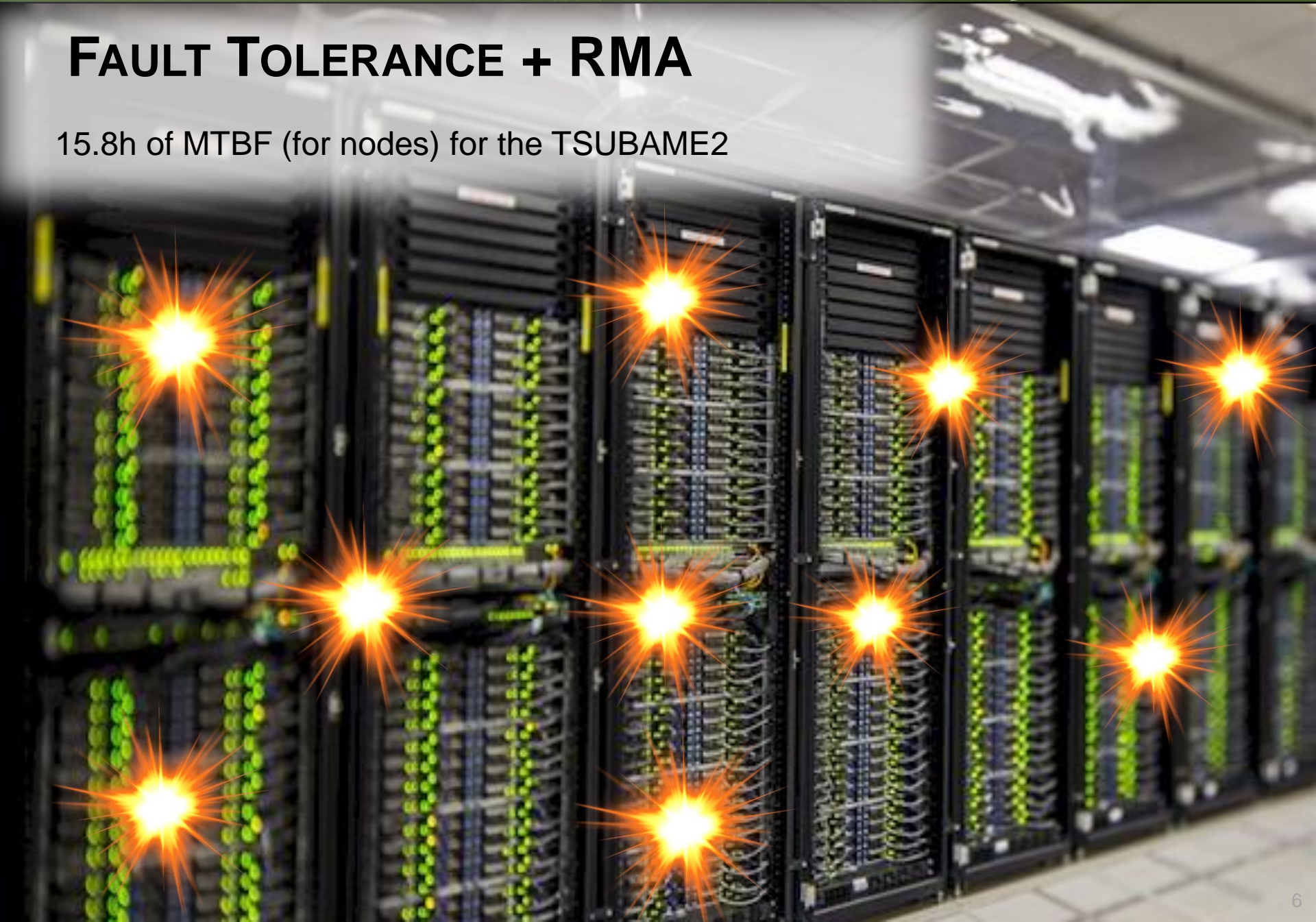
[2] D. Petrovic et al., High-performance RMA-based broadcast on the Intel SCC. SPAA'12

FAULT TOLERANCE + RMA



FAULT TOLERANCE + RMA

15.8h of MTBF (for nodes) for the TSUBAME2



FAULT TOLERANCE + RMA

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated
Checkpointing (CC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated
Checkpointing (CC)

uncoordinated
checkpointing
and message
logging (UC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing

Message Passing

Coordinated Checkpointing (CC)

uncoordinated checkpointing and message logging (UC)

FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)

This block contains a large grid of small thumbnail images, each representing a research paper or technical report related to Coordinated Checkpointing (CC) in message passing systems. The thumbnails are arranged in a grid pattern, with some larger text boxes overlaid on top, providing a visual overview of the research landscape in this area.

RMA

This block represents the area of uncoordinated checkpointing and message logging (UC) in Remote Memory Access (RMA). It is characterized by a large, mostly blank space, indicating a significant gap in research compared to the message passing domain. A few small, faint text boxes are visible, suggesting the presence of limited research in this area.

uncoordinated checkpointing and message logging (UC)



FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)

This block contains a grid of approximately 100 small thumbnail images, each representing a research paper or technical document related to Coordinated Checkpointing (CC) in message passing systems. The thumbnails are arranged in a grid and are mostly white with some text visible, though too small to read. The grid is contained within a green-bordered box.

RMA

[5] N. AE, S. Krishnamoorthy, N. Govind, and B. Palmer. A Redundant Communication Approach to Scalable Fault Tolerance in PGAS Programming Models. In *Proc. Dist. and Net. Proc. (DNP)*, the Eur. Inst. Conf., pages 24-31, 2011.

logging memory accesses vs. messages

uncoordinated checkpointing and message logging (UC)



FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)

1. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1990 ACM Conference on Supercomputing*, pp. 318-328, 1990.

2. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 39, no. 12, pp. 1478-1488, 1990.

3. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1991 ACM Conference on Supercomputing*, pp. 28-38, 1991.

4. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1558-1568, 1991.

5. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1992 ACM Conference on Supercomputing*, pp. 12-22, 1992.

6. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 41, no. 12, pp. 1648-1658, 1992.

7. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1993 ACM Conference on Supercomputing*, pp. 18-28, 1993.

8. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 42, no. 12, pp. 1748-1758, 1993.

9. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1994 ACM Conference on Supercomputing*, pp. 24-34, 1994.

10. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1838-1848, 1994.

11. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1995 ACM Conference on Supercomputing*, pp. 30-40, 1995.

12. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1928-1938, 1995.

13. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1996 ACM Conference on Supercomputing*, pp. 36-46, 1996.

14. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 45, no. 12, pp. 2018-2028, 1996.

15. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1997 ACM Conference on Supercomputing*, pp. 42-52, 1997.

16. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 46, no. 12, pp. 2108-2118, 1997.

17. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1998 ACM Conference on Supercomputing*, pp. 48-58, 1998.

18. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 47, no. 12, pp. 2198-2208, 1998.

19. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 1999 ACM Conference on Supercomputing*, pp. 54-64, 1999.

20. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 48, no. 12, pp. 2288-2298, 1999.

21. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2000 ACM Conference on Supercomputing*, pp. 60-70, 2000.

22. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 49, no. 12, pp. 2378-2388, 2000.

23. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2001 ACM Conference on Supercomputing*, pp. 66-76, 2001.

24. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 2468-2478, 2001.

25. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2002 ACM Conference on Supercomputing*, pp. 72-82, 2002.

26. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 2558-2568, 2002.

27. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2003 ACM Conference on Supercomputing*, pp. 78-88, 2003.

28. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 52, no. 12, pp. 2648-2658, 2003.

29. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2004 ACM Conference on Supercomputing*, pp. 84-94, 2004.

30. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 53, no. 12, pp. 2738-2748, 2004.

31. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2005 ACM Conference on Supercomputing*, pp. 90-100, 2005.

32. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 54, no. 12, pp. 2828-2838, 2005.

33. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2006 ACM Conference on Supercomputing*, pp. 96-106, 2006.

34. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 55, no. 12, pp. 2918-2928, 2006.

35. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2007 ACM Conference on Supercomputing*, pp. 102-112, 2007.

36. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 56, no. 12, pp. 3008-3018, 2007.

37. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2008 ACM Conference on Supercomputing*, pp. 108-118, 2008.

38. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 3098-3108, 2008.

39. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2009 ACM Conference on Supercomputing*, pp. 114-124, 2009.

40. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 58, no. 12, pp. 3188-3198, 2009.

41. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2010 ACM Conference on Supercomputing*, pp. 120-130, 2010.

42. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 59, no. 12, pp. 3278-3288, 2010.

43. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2011 ACM Conference on Supercomputing*, pp. 126-136, 2011.

44. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 60, no. 12, pp. 3368-3378, 2011.

45. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2012 ACM Conference on Supercomputing*, pp. 132-142, 2012.

46. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 3458-3468, 2012.

47. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2013 ACM Conference on Supercomputing*, pp. 138-148, 2013.

48. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 3548-3558, 2013.

49. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2014 ACM Conference on Supercomputing*, pp. 144-154, 2014.

50. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 63, no. 12, pp. 3638-3648, 2014.

51. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2015 ACM Conference on Supercomputing*, pp. 150-160, 2015.

52. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3728-3738, 2015.

53. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2016 ACM Conference on Supercomputing*, pp. 156-166, 2016.

54. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3818-3828, 2016.

55. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2017 ACM Conference on Supercomputing*, pp. 162-172, 2017.

56. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 66, no. 12, pp. 3908-3918, 2017.

57. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2018 ACM Conference on Supercomputing*, pp. 168-178, 2018.

58. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 67, no. 12, pp. 3998-4008, 2018.

59. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2019 ACM Conference on Supercomputing*, pp. 174-184, 2019.

60. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 68, no. 12, pp. 4088-4098, 2019.

61. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2020 ACM Conference on Supercomputing*, pp. 180-190, 2020.

62. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 69, no. 12, pp. 4178-4188, 2020.

63. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2021 ACM Conference on Supercomputing*, pp. 186-196, 2021.

64. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 70, no. 12, pp. 4268-4278, 2021.

65. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2022 ACM Conference on Supercomputing*, pp. 192-202, 2022.

66. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 71, no. 12, pp. 4358-4368, 2022.

67. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2023 ACM Conference on Supercomputing*, pp. 198-208, 2023.

68. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 72, no. 12, pp. 4448-4458, 2023.

69. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2024 ACM Conference on Supercomputing*, pp. 204-214, 2024.

70. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 73, no. 12, pp. 4538-4548, 2024.

71. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *Proceedings of the 2025 ACM Conference on Supercomputing*, pp. 210-220, 2025.

72. J. D. Dupont and S. J. Lee, "Fault-tolerant distributed memory computing," *IEEE Transactions on Computers*, vol. 74, no. 12, pp. 4628-4638, 2025.

RMA

[5] N. AE, S. Krishnamoorthy, N. Govind, and B. Palmer. A Redundant Communication Approach to Scalable Fault Tolerance in PGAS Programming Models. In *Proc. Dist. and Net. Proc. (DNP)*, the Eur. Inst. Conf., pages 24-31, 2011.

logging memory accesses vs. messages

checkpointing in RMA-based applications

uncoordinated checkpointing and message logging (UC)



FAULT TOLERANCE + RMA

- Fault tolerance is well studied for message passing
- Scarce research exists for fault tolerance for RMA

Message Passing

Coordinated Checkpointing (CC)

This block contains a grid of approximately 40 small thumbnail images, each representing a research paper or technical report related to Coordinated Checkpointing (CC) in message passing systems. The thumbnails are arranged in a roughly rectangular grid, with some overlapping. The text within the thumbnails is too small to read but appears to be a list of references or a collection of abstracts.

RMA

[5] N. AE, S. Krishnamoorthy, N. Govind, and B. Palmer. A Redundant Communication Approach to Scalable Fault Tolerance in PGAS Programming Models. In *Proc. Dist. and Net. Proc. (DNP)*, the Eur. Inst. Conf., pages 24-31, 2011.

logging memory accesses vs. messages

checkpointing in RMA-based applications

fault tolerance mechanisms and schemes
performance

uncoordinated checkpointing and message logging (UC)

OVERVIEW OF OUR RESEARCH

OVERVIEW OF OUR RESEARCH

Generic model

A large, solid grey rectangular area that serves as a placeholder for content related to the 'Generic model'.

OVERVIEW OF OUR RESEARCH

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

OVERVIEW OF OUR RESEARCH

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$$\text{PUT}(p \xrightarrow{\Rightarrow} q)$$
$$\text{GET}(p \xleftarrow{\Leftarrow} q)$$

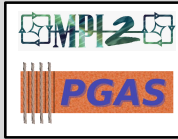
OVERVIEW OF OUR RESEARCH

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

PUT($p \Rightarrow q$)

GET($p \Leftarrow q$)



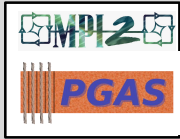
OVERVIEW OF OUR RESEARCH

Generic model

$$\mathcal{D} = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

PUT($p \Rightarrow q$)

GET($p \Leftarrow q$)



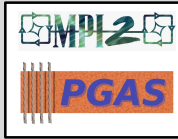
CC in RMA

OVERVIEW OF OUR RESEARCH

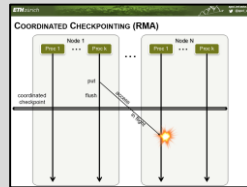
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

$$\text{PUT}(p \rightrightarrows q)$$

$$\text{GET}(p \leftrightsquigarrow q)$$


CC in RMA



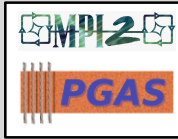
MP vs. RMA

OVERVIEW OF OUR RESEARCH

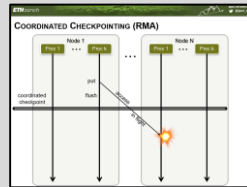
Generic model

$$D = \langle \mathcal{P}, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

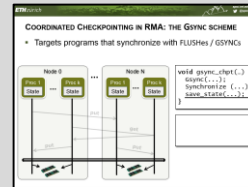
$$\text{PUT}(p \xrightarrow{\Rightarrow} q)$$

$$\text{GET}(p \xleftarrow{\Leftarrow} q)$$


CC in RMA



MP vs. RMA



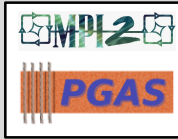
Schemes

OVERVIEW OF OUR RESEARCH

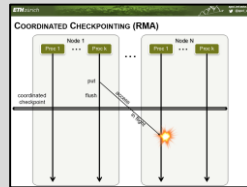
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

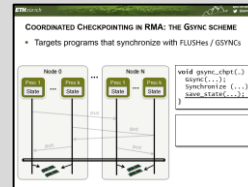
$$\text{PUT}(p \xrightarrow{so} q)$$

$$\text{GET}(p \xrightarrow{so} q)$$


CC in RMA



MP vs. RMA



Schemes

UC in RMA

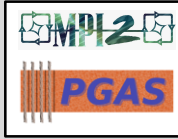
OVERVIEW OF OUR RESEARCH

Generic model

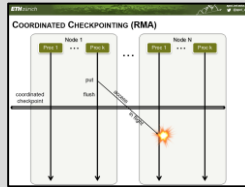
$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

PUT($p \Rightarrow q$)

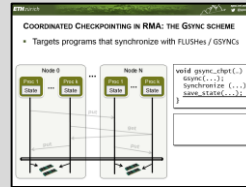
GET($p \Leftarrow q$)



CC in RMA

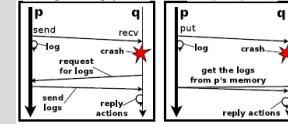


MP vs. RMA



Schemes

UC in RMA



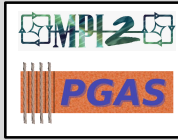
MP vs. RMA

OVERVIEW OF OUR RESEARCH

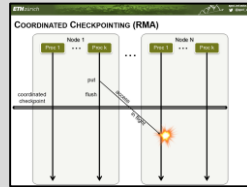
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

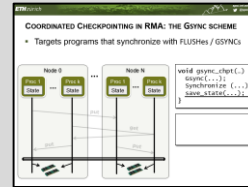
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

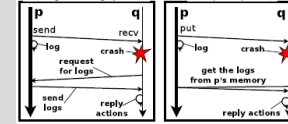


MP vs. RMA

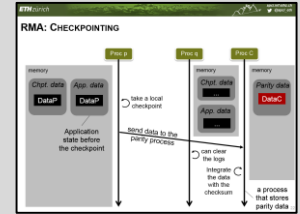


Schemes

UC in RMA



MP vs. RMA



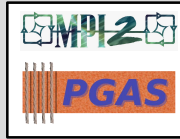
Checkpointing schemes

OVERVIEW OF OUR RESEARCH

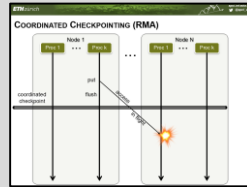
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

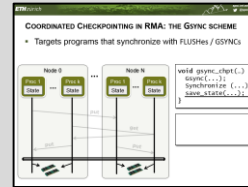
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

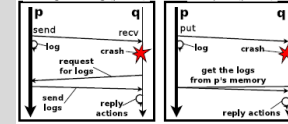


MP vs. RMA

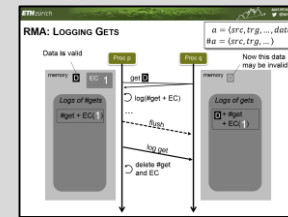


Schemes

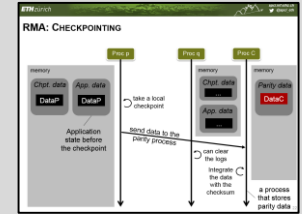
UC in RMA



MP vs. RMA



Logging accesses



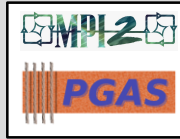
Checkpointing schemes

OVERVIEW OF OUR RESEARCH

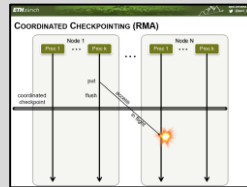
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

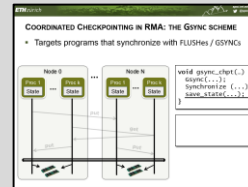
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA



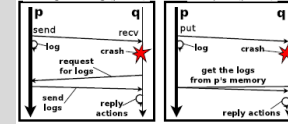
MP vs. RMA



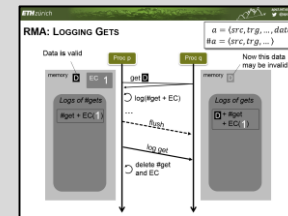
Schemes

Recovery in RMA

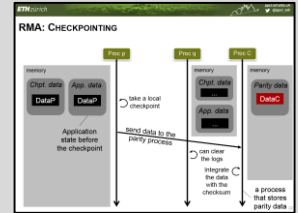
UC in RMA



MP vs. RMA



Logging accesses



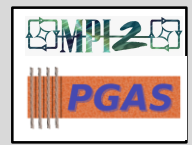
Checkpointing schemes

OVERVIEW OF OUR RESEARCH

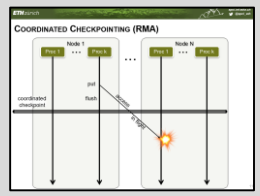
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

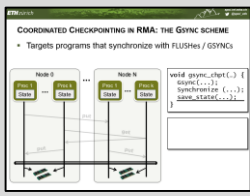
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

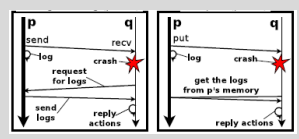


MP vs. RMA

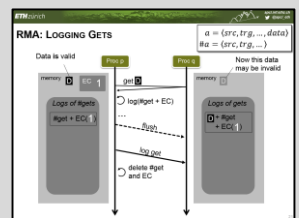


Schemes

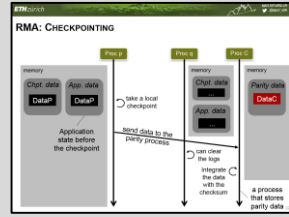
UC in RMA



MP vs. RMA

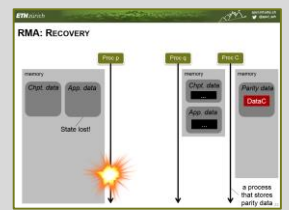


Logging accesses



Checkpointing schemes

Recovery in RMA



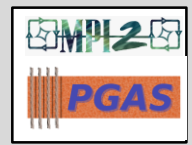
Basic scheme

OVERVIEW OF OUR RESEARCH

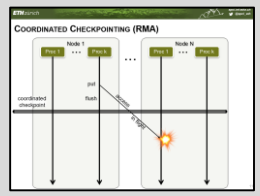
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

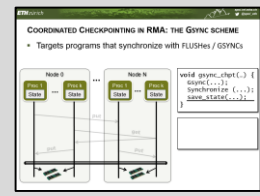
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

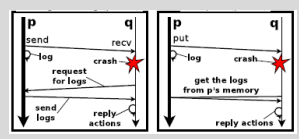


MP vs. RMA

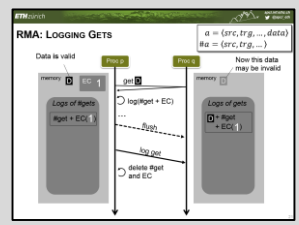


Schemes

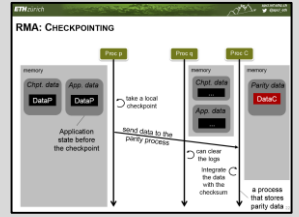
UC in RMA



MP vs. RMA

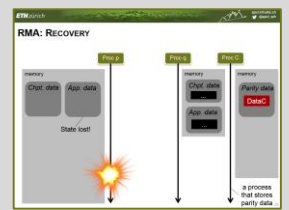


Logging accesses



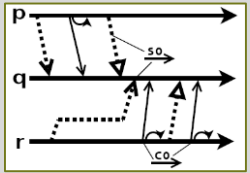
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

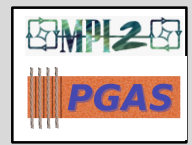


OVERVIEW OF OUR RESEARCH

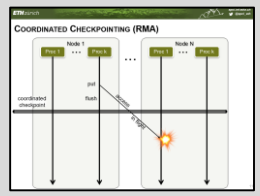
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

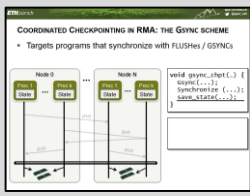
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

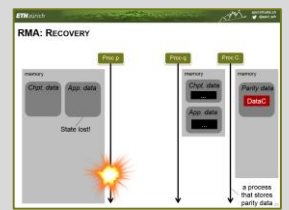


MP vs. RMA



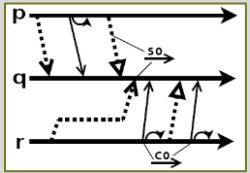
Schemes

Recovery in RMA

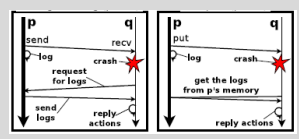


Basic scheme

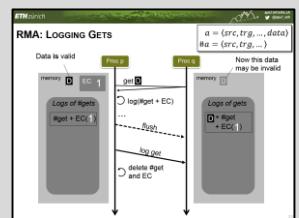
Extended RMA semantics



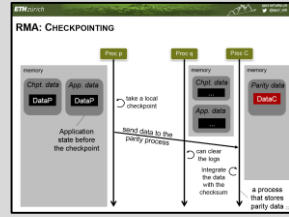
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

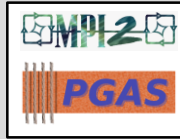
Topology-awareness

OVERVIEW OF OUR RESEARCH

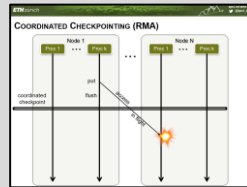
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

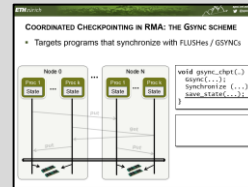
PUT($p \xrightarrow{so} q$)
GET($p \xleftarrow{so} q$)



CC in RMA

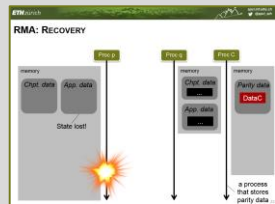


MP vs. RMA



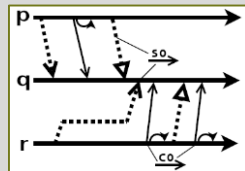
Schemes

Recovery in RMA

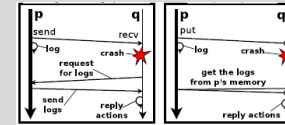


Basic scheme

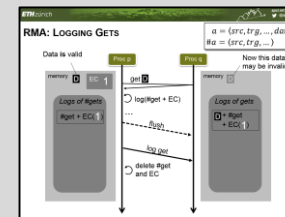
Extended RMA semantics



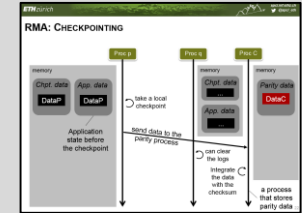
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

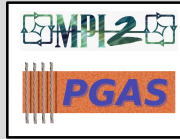
Model extensions

OVERVIEW OF OUR RESEARCH

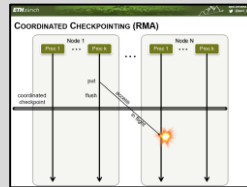
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

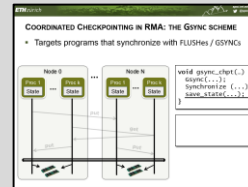
PUT($p \xrightarrow{so} q$)
GET($p \xleftarrow{so} q$)



CC in RMA

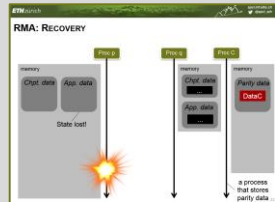


MP vs. RMA



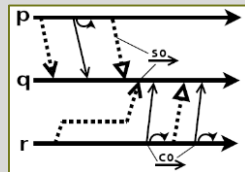
Schemes

Recovery in RMA

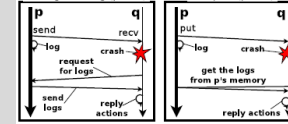


Basic scheme

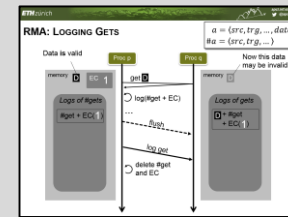
Extended RMA semantics



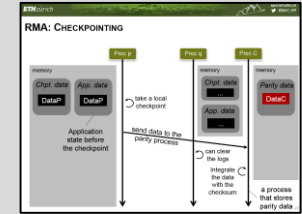
UC in RMA



MP vs. RMA



Logging accesses

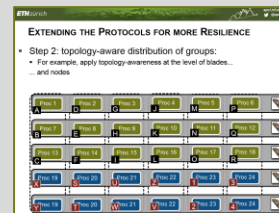


Checkpointing schemes

Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



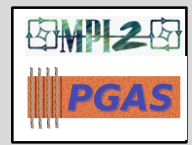
Distribution of processes

OVERVIEW OF OUR RESEARCH

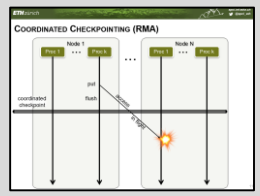
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

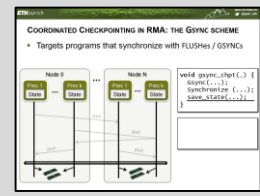
PUT($p \xrightarrow{so} q$)
GET($p \xleftarrow{so} q$)



CC in RMA

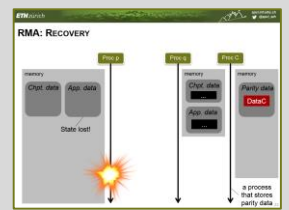


MP vs. RMA



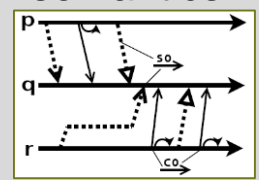
Schemes

Recovery in RMA

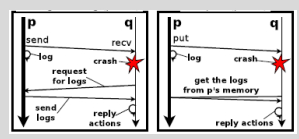


Basic scheme

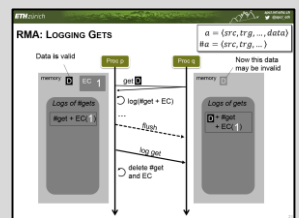
Extended RMA semantics



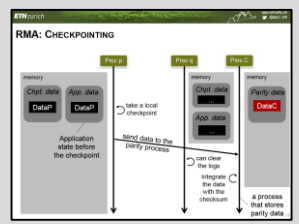
UC in RMA



MP vs. RMA



Logging accesses

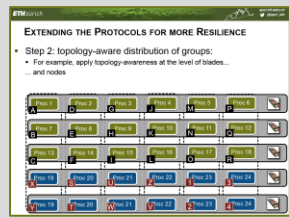


Checkpointing schemes

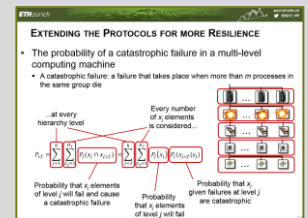
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



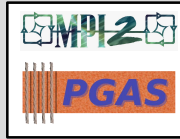
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

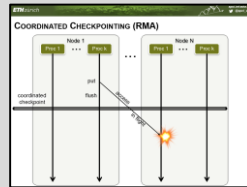
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

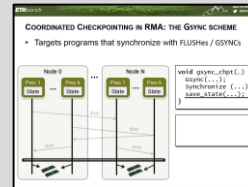
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

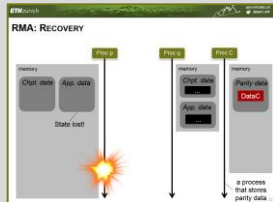


MP vs. RMA



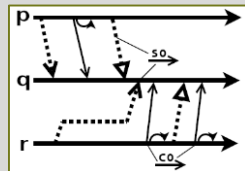
Schemes

Recovery in RMA



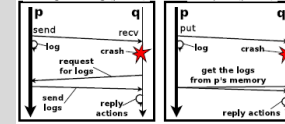
Basic scheme

Extended RMA semantics

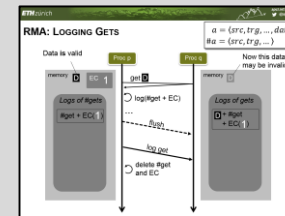


Holistic fault-tolerance library

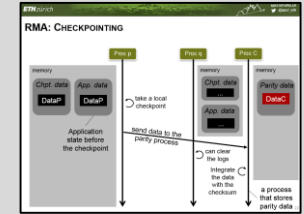
UC in RMA



MP vs. RMA



Logging accesses

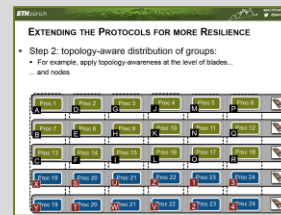


Checkpointing schemes

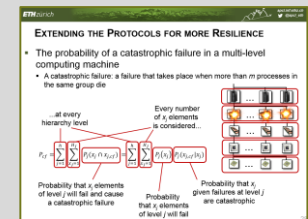
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



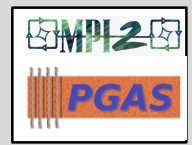
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

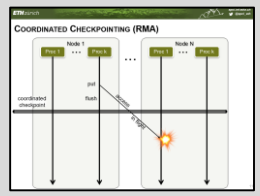
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

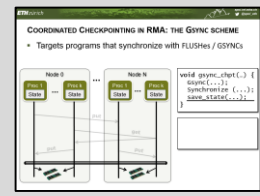
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

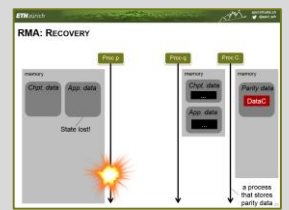


MP vs. RMA



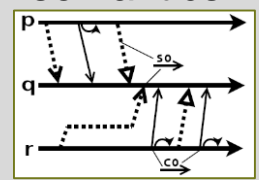
Schemes

Recovery in RMA

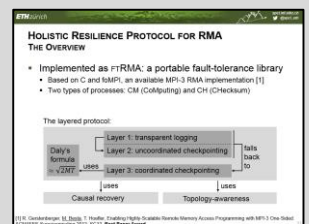


Basic scheme

Extended RMA semantics

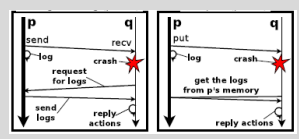


Holistic fault-tolerance library

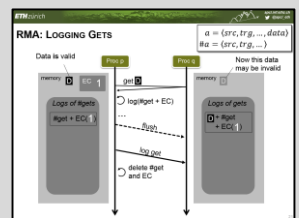


Design

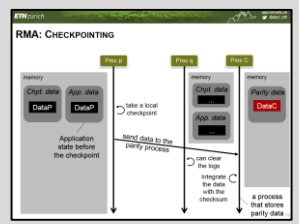
UC in RMA



MP vs. RMA



Logging accesses

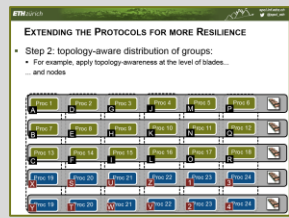


Checkpointing schemes

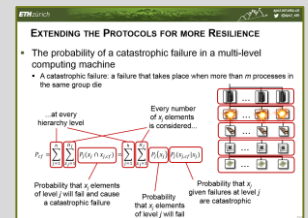
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

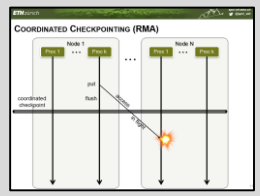
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

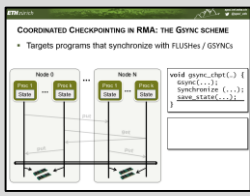
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

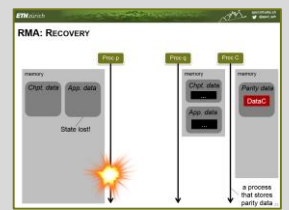


MP vs. RMA



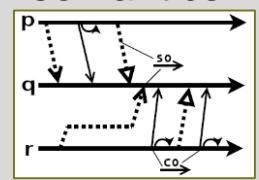
Schemes

Recovery in RMA

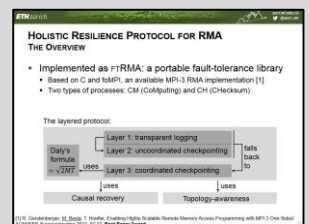


Basic scheme

Extended RMA semantics



Holistic fault-tolerance library

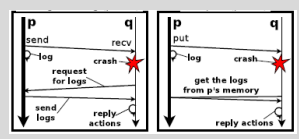


Design

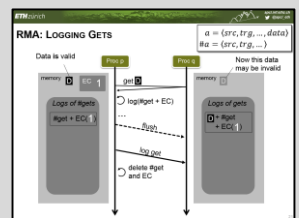
Checkpoints on demand

Optimizations

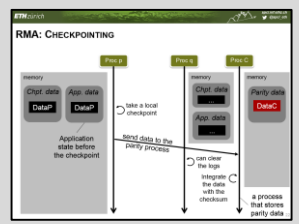
UC in RMA



MP vs. RMA



Logging accesses

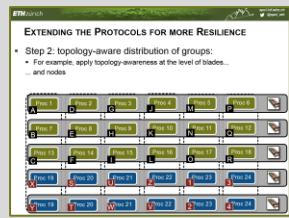


Checkpointing schemes

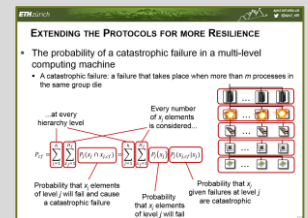
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

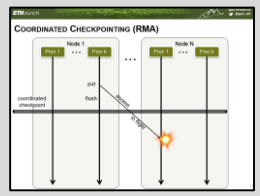
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

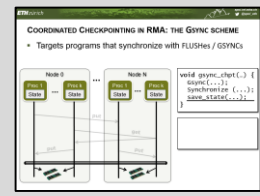
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

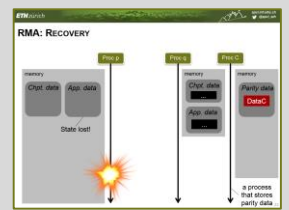


MP vs. RMA



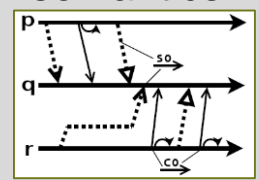
Schemes

Recovery in RMA

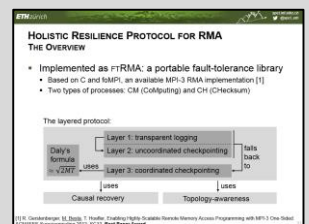


Basic scheme

Extended RMA semantics



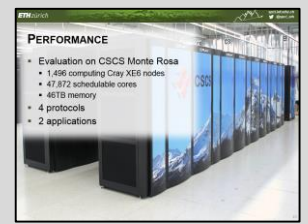
Holistic fault-tolerance library



Design

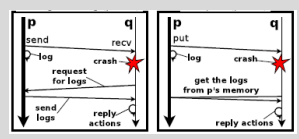
Checkpoints on demand

Optimizations

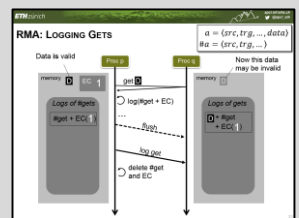


Performance

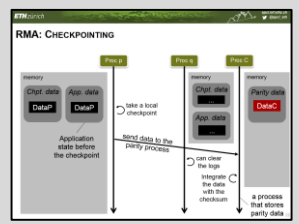
UC in RMA



MP vs. RMA



Logging accesses

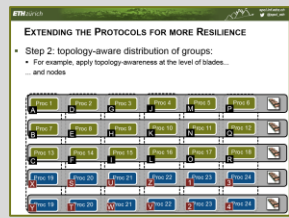


Checkpointing schemes

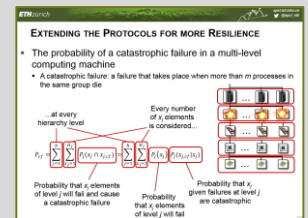
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



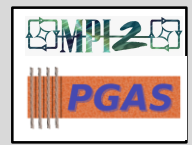
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

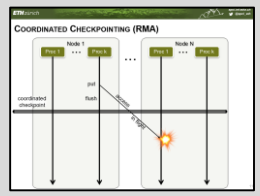
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

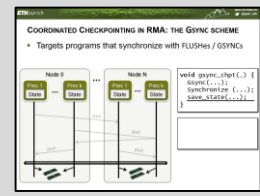
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

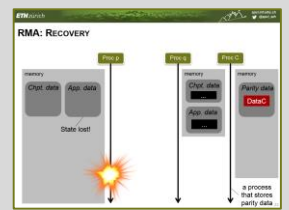


MP vs. RMA



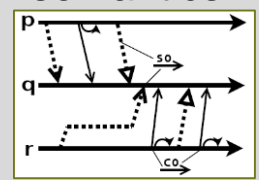
Schemes

Recovery in RMA



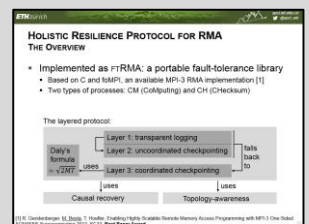
Basic scheme

Extended RMA semantics



Proofs

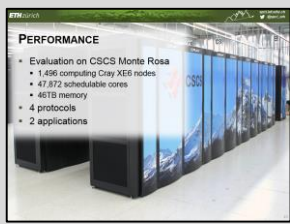
Holistic fault-tolerance library



Design

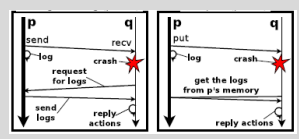
Checkpoints on demand

Optimizations

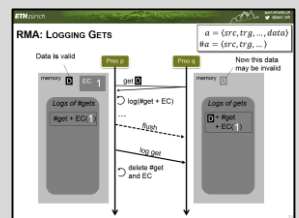


Performance

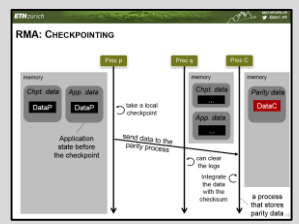
UC in RMA



MP vs. RMA



Logging accesses

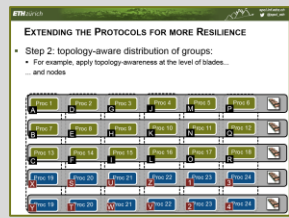


Checkpointing schemes

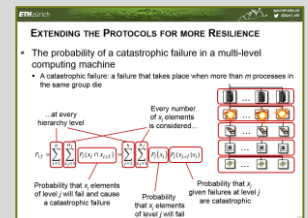
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



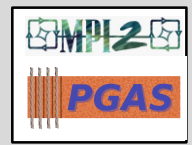
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

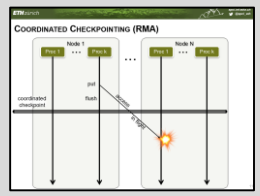
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

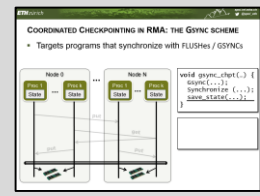
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

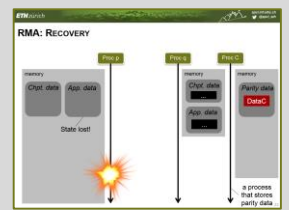


MP vs. RMA



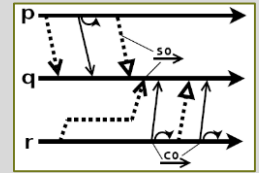
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

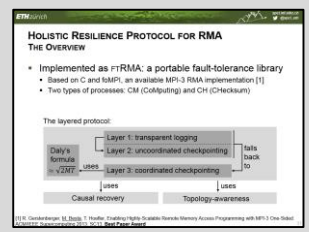


Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the cohb order to as the gsync order).*

Deadlock freedom
Correct recovery

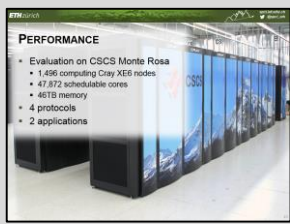
Holistic fault-tolerance library



Design

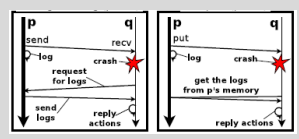
Checkpoints on demand

Optimizations

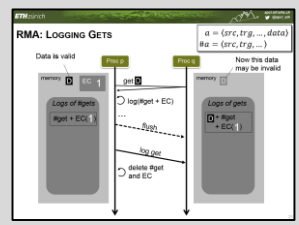


Performance

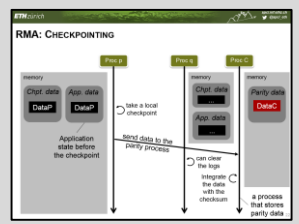
UC in RMA



MP vs. RMA



Logging accesses

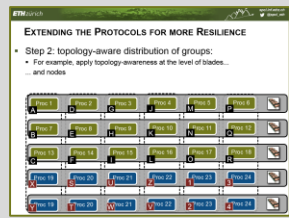


Checkpointing schemes

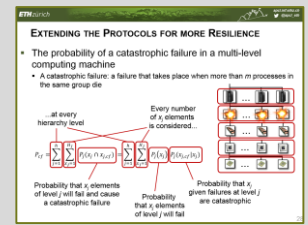
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



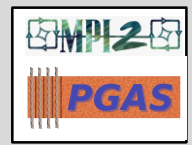
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

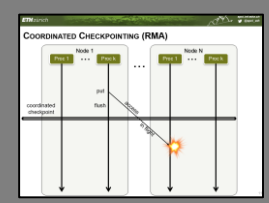
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

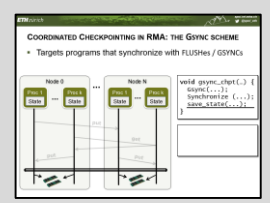
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

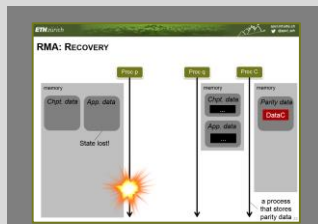


MP vs. RMA



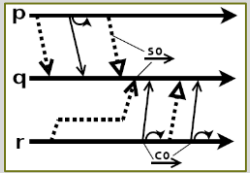
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

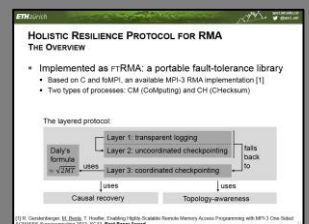


Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the cohb order to as the gsync order).*

Deadlock freedom
Correct recovery

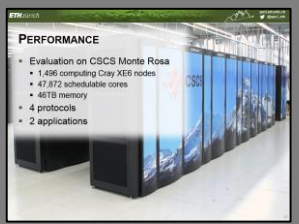
Holistic fault-tolerance library



Design

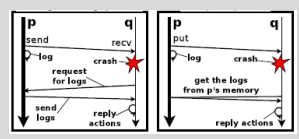
Checkpoints on demand

Optimizations

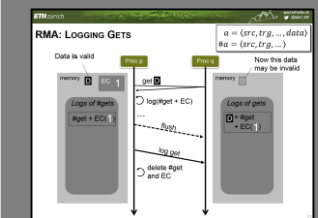


Performance

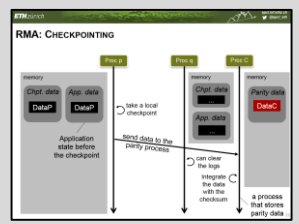
UC in RMA



MP vs. RMA



Logging accesses

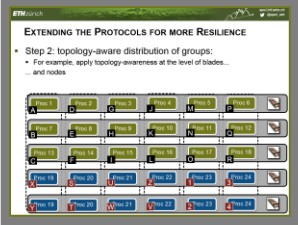


Checkpointing schemes

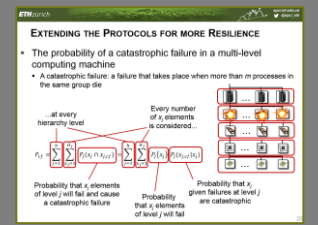
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

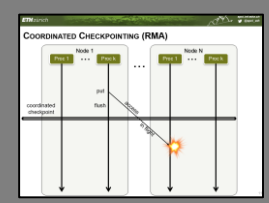
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

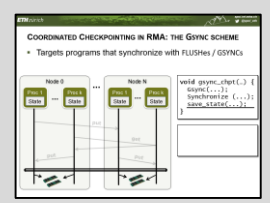
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

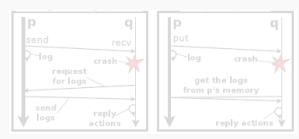


MP vs. RMA

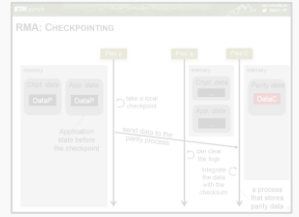


Schemes

UC in RMA

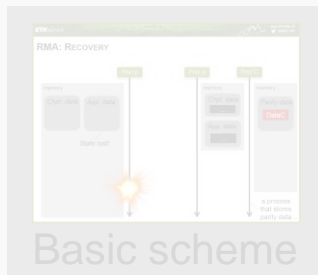


MP vs. RMA



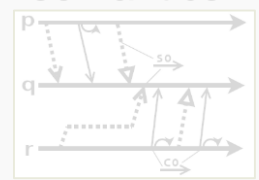
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

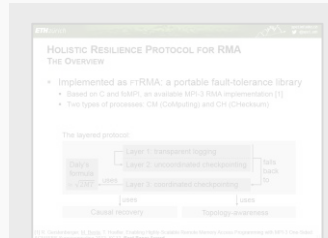
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

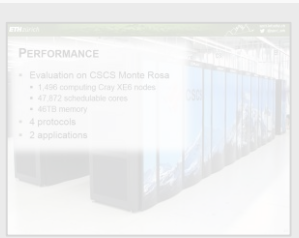
Holistic fault-tolerance library



Design

Checkpoints on demand

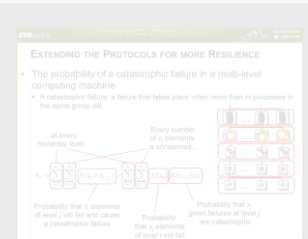
Optimizations



Performance



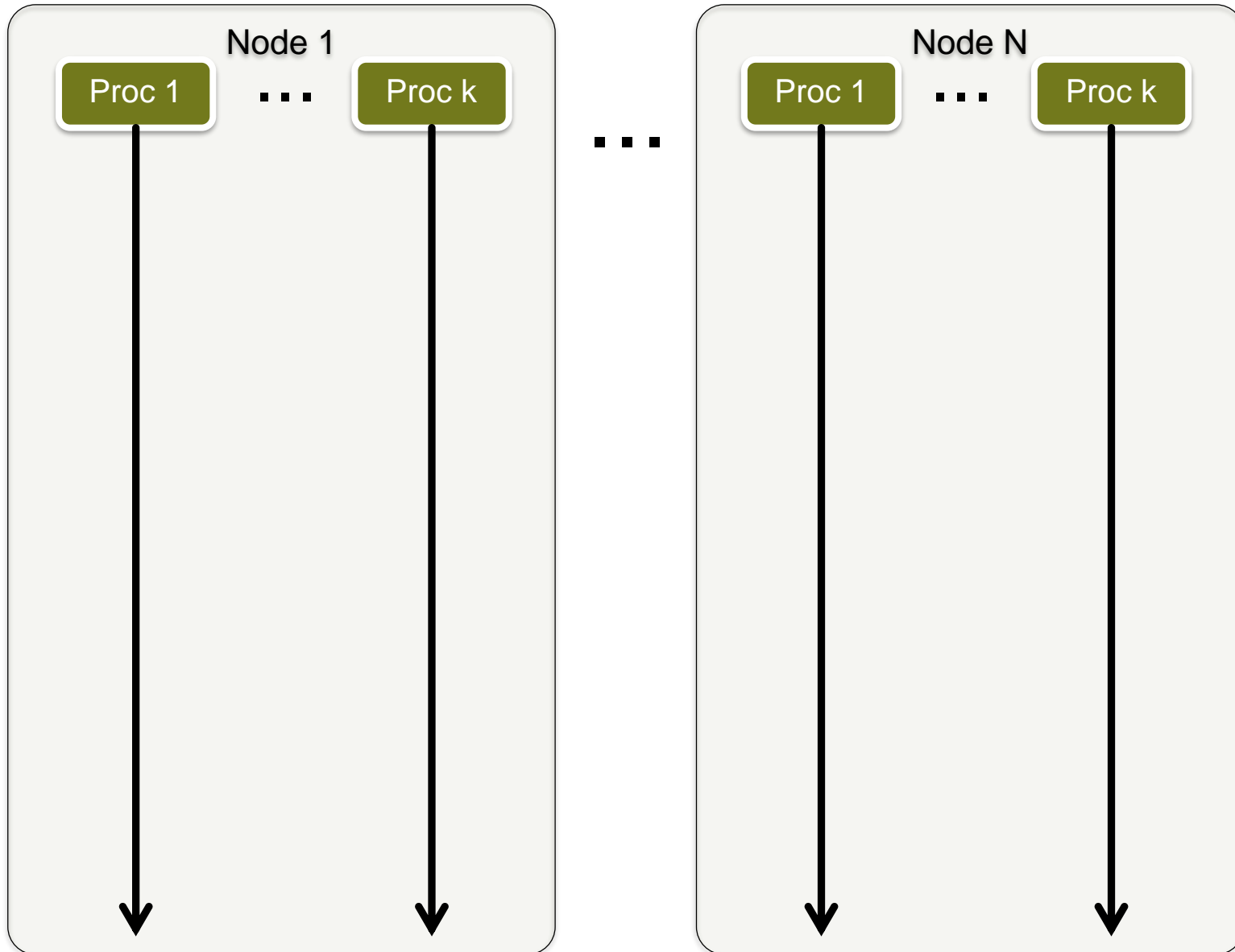
Distribution of processes



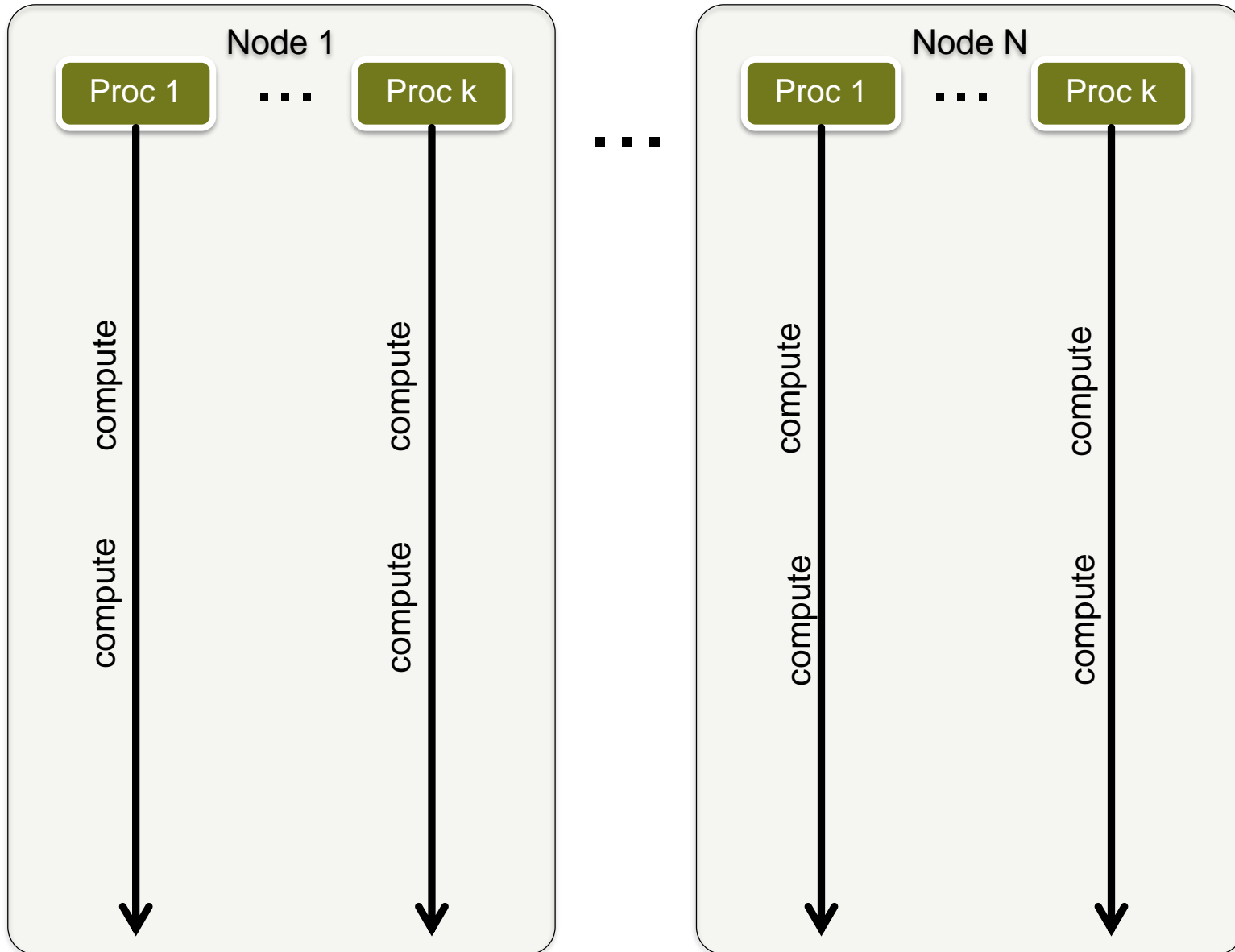
Decreasing failure prob.

COORDINATED CHECKPOINTING (MP)

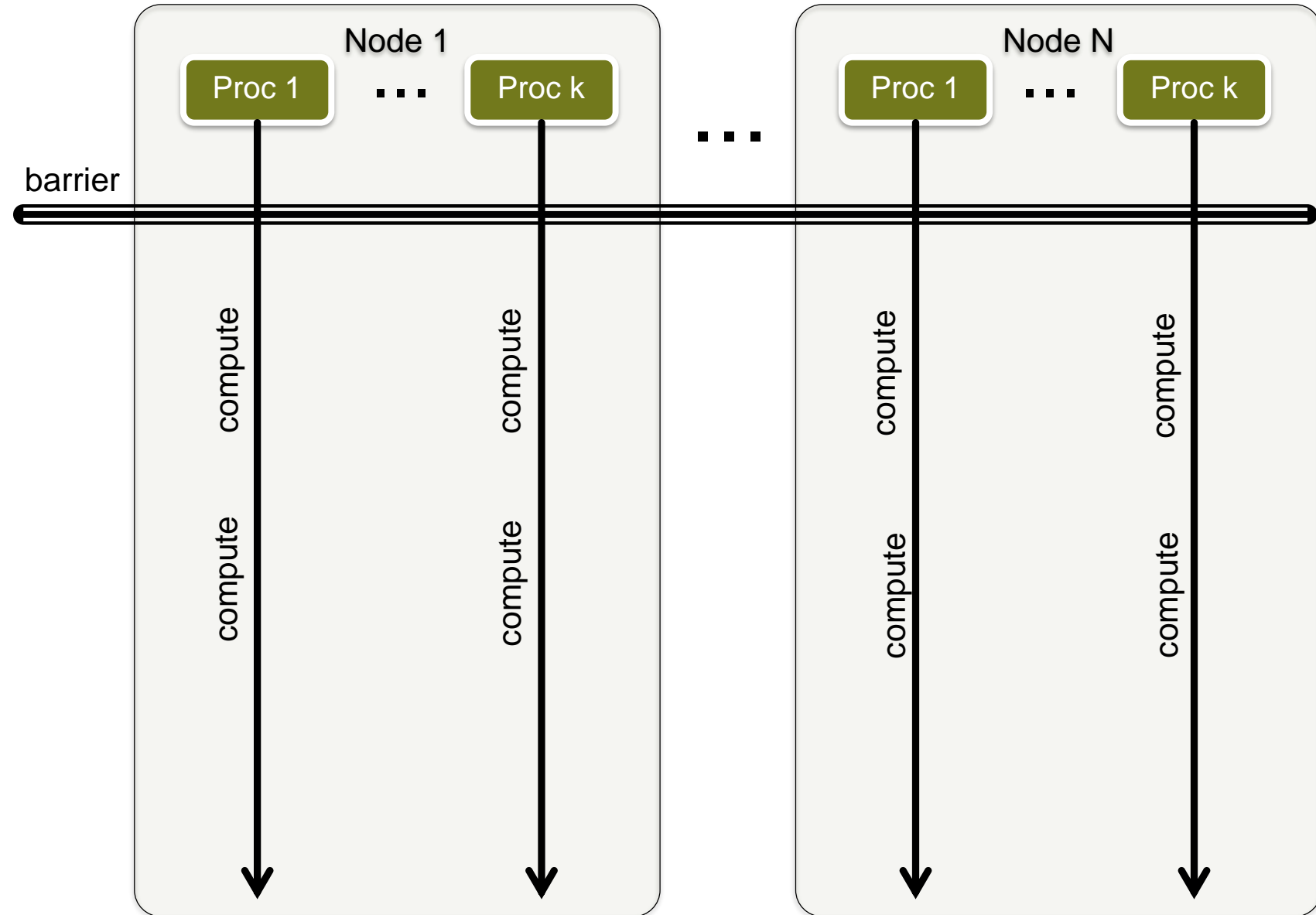
COORDINATED CHECKPOINTING (MP)



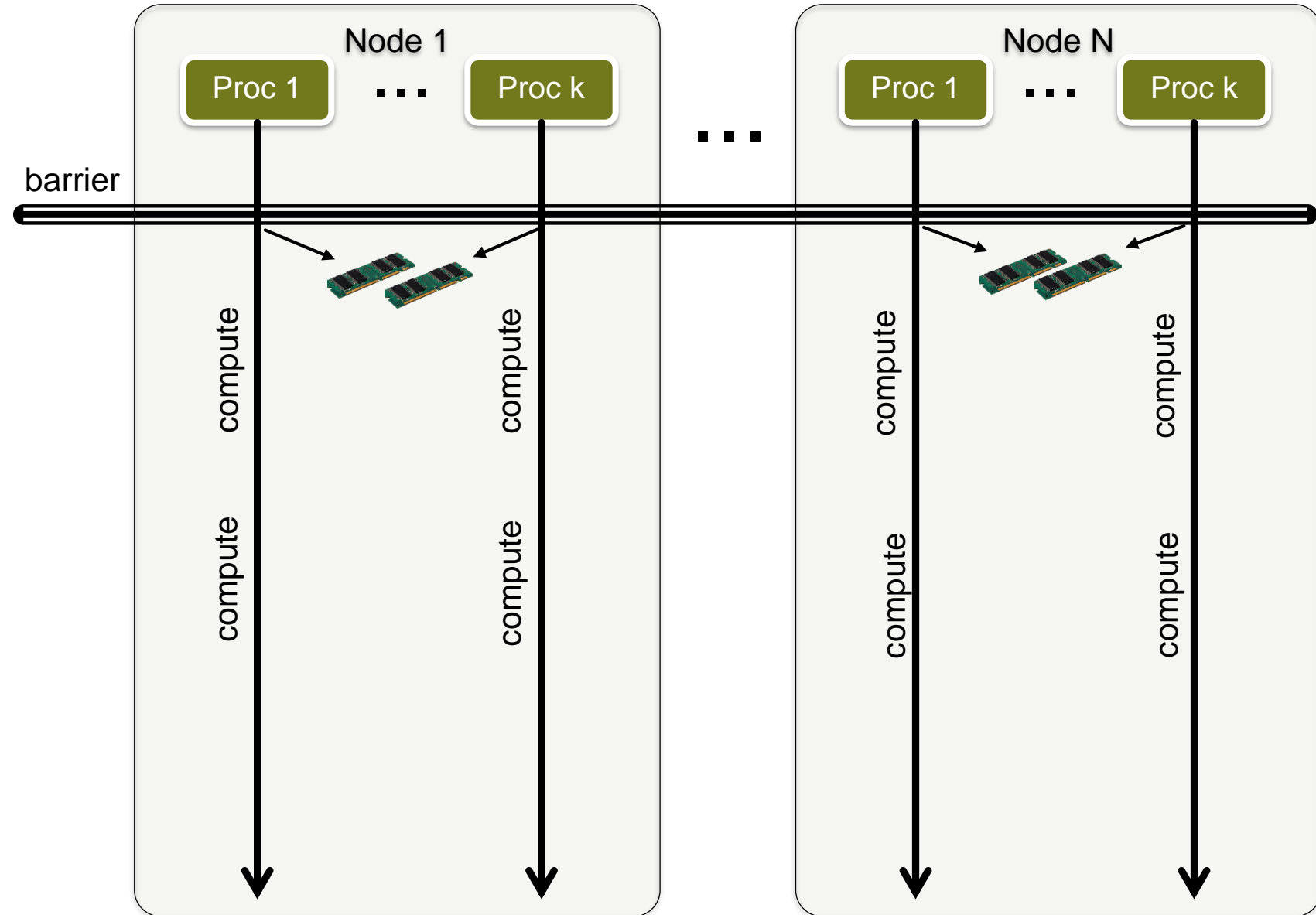
COORDINATED CHECKPOINTING (MP)



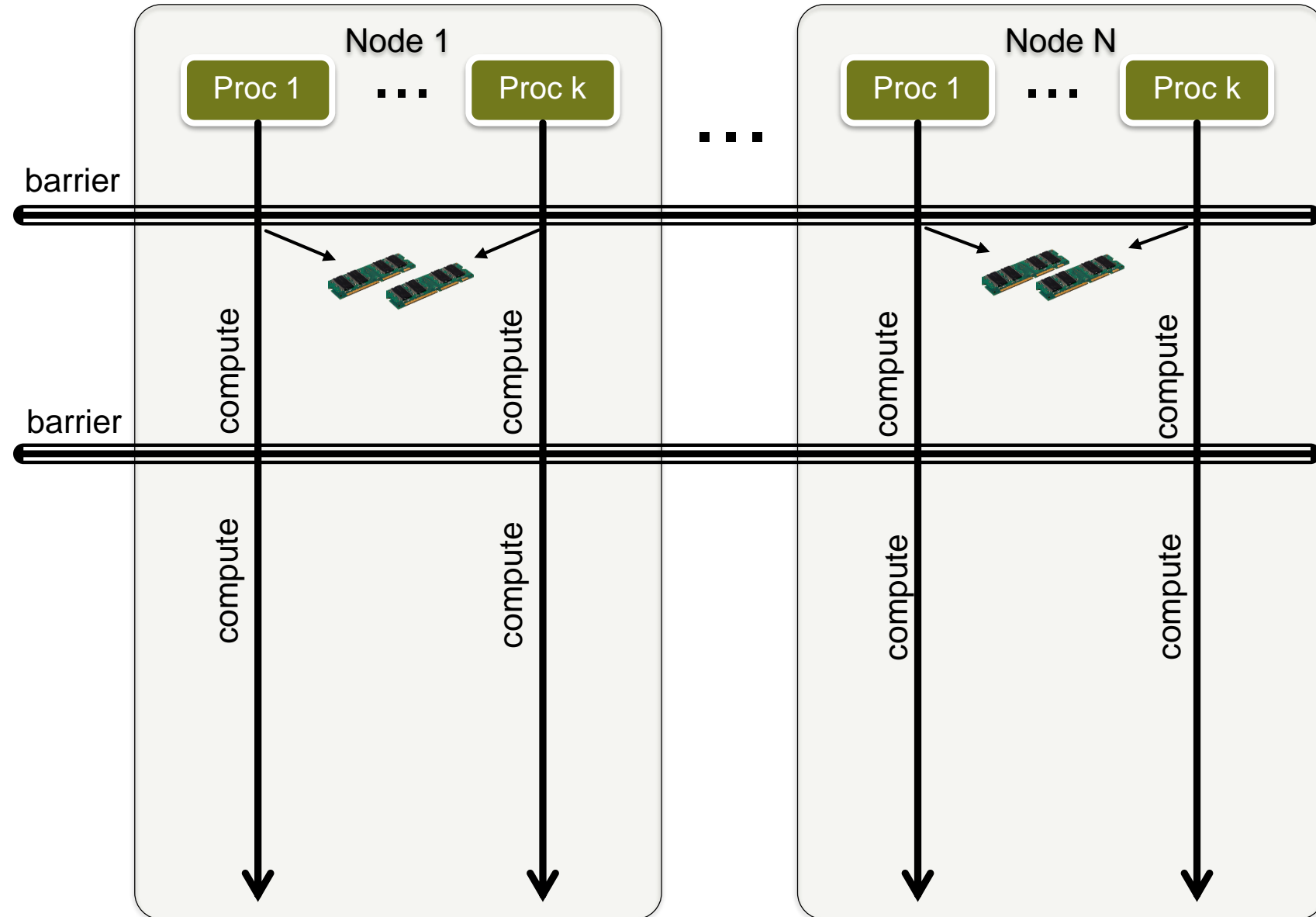
COORDINATED CHECKPOINTING (MP)



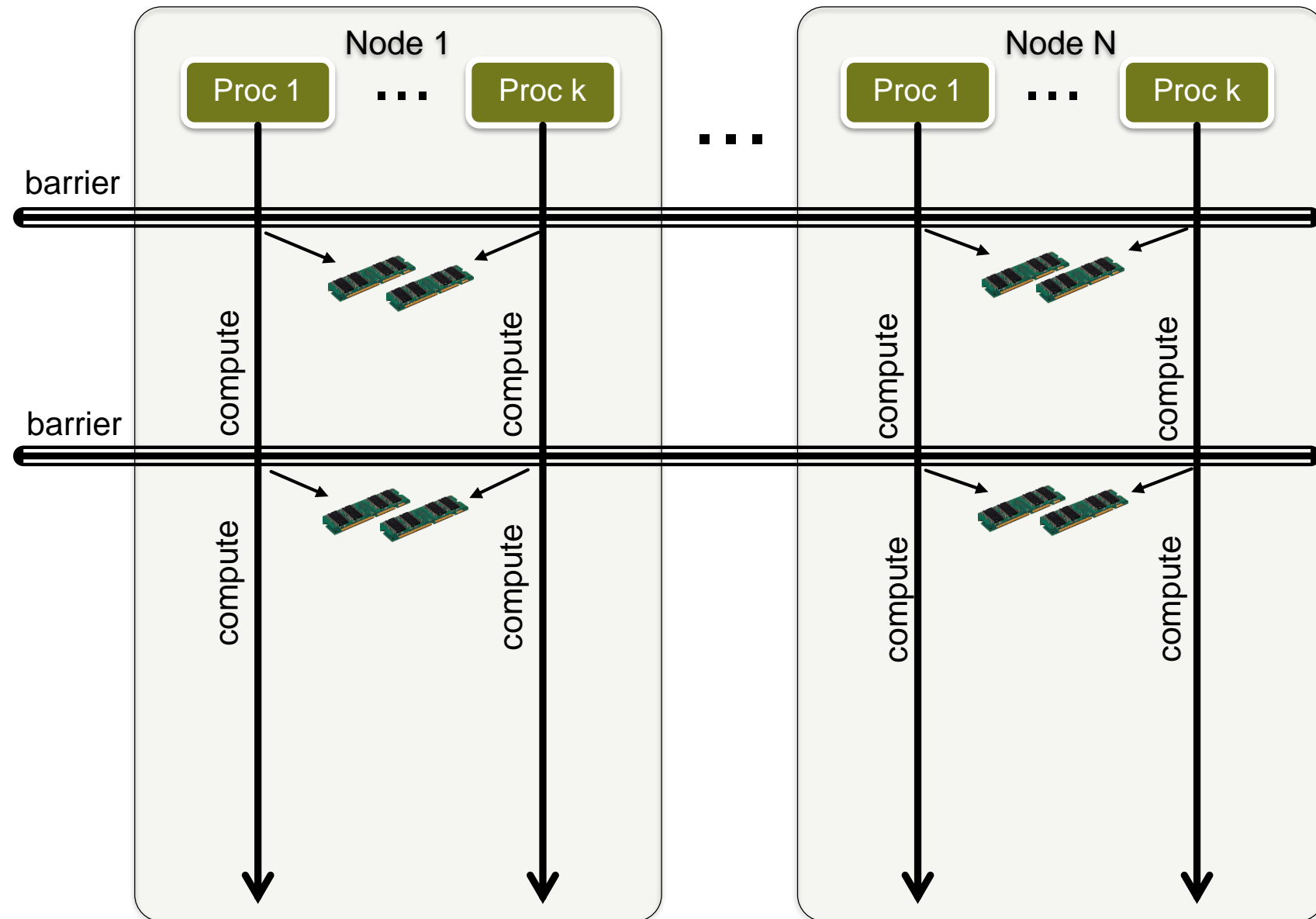
COORDINATED CHECKPOINTING (MP)



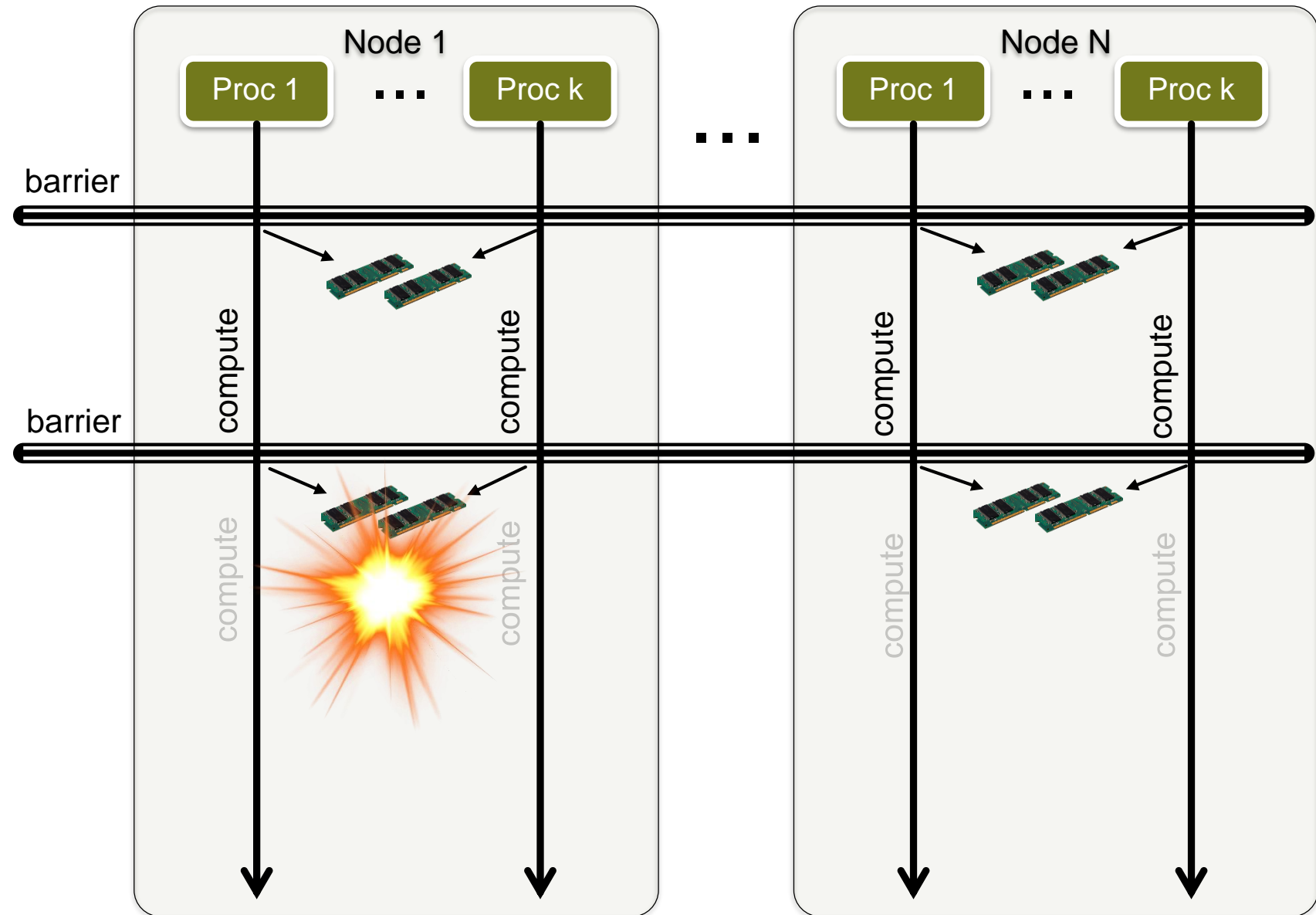
COORDINATED CHECKPOINTING (MP)



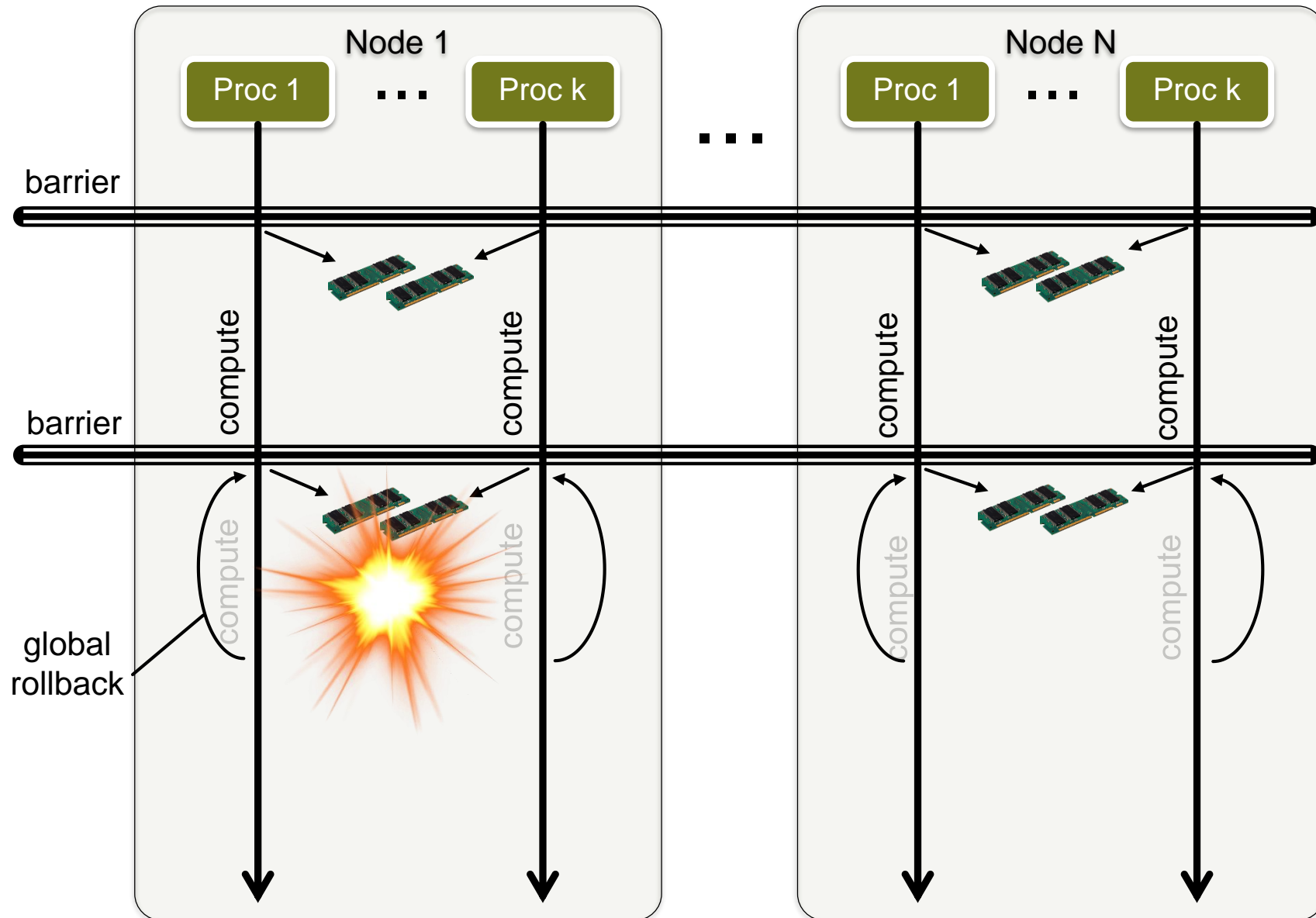
COORDINATED CHECKPOINTING (MP)



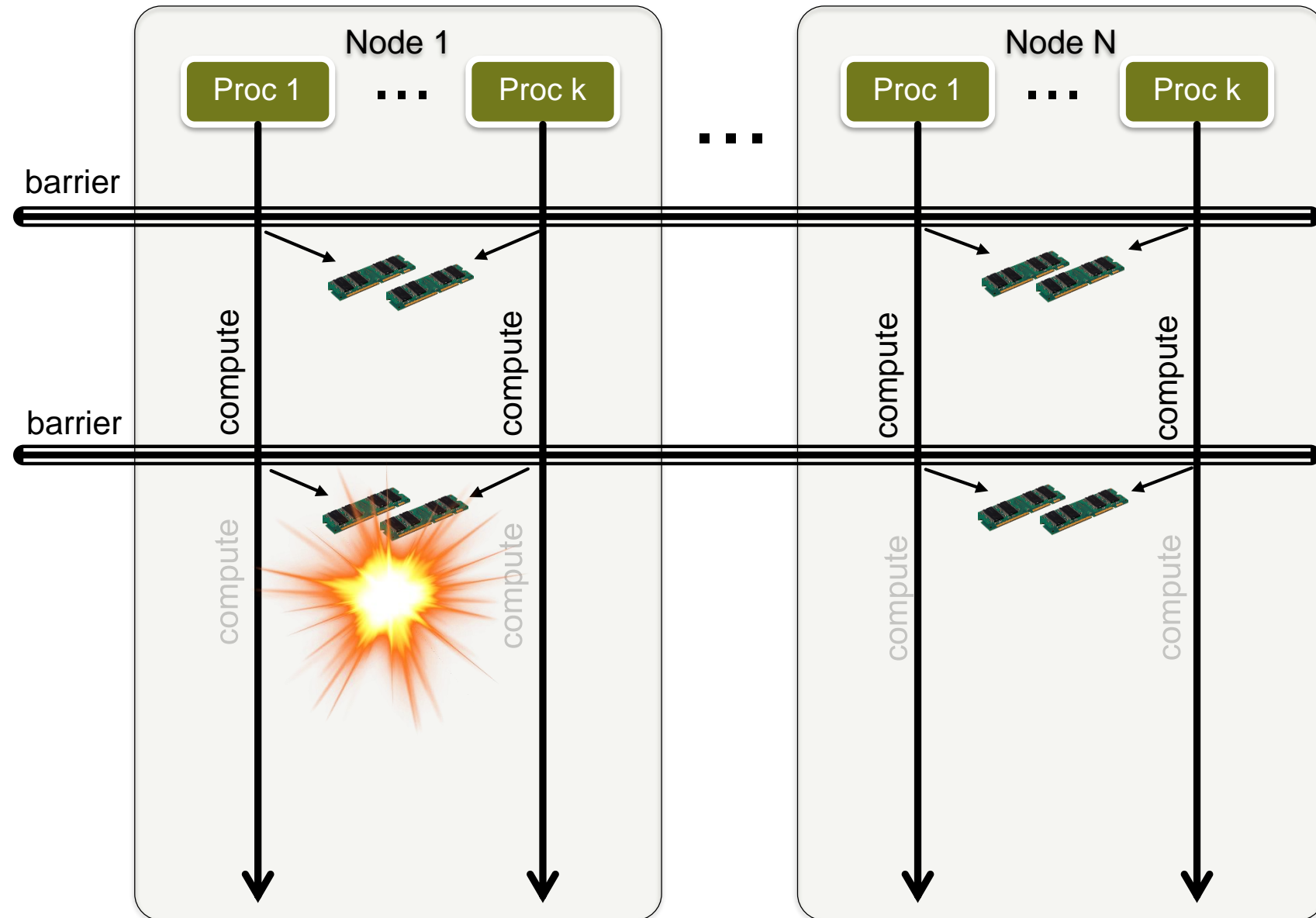
COORDINATED CHECKPOINTING (MP)



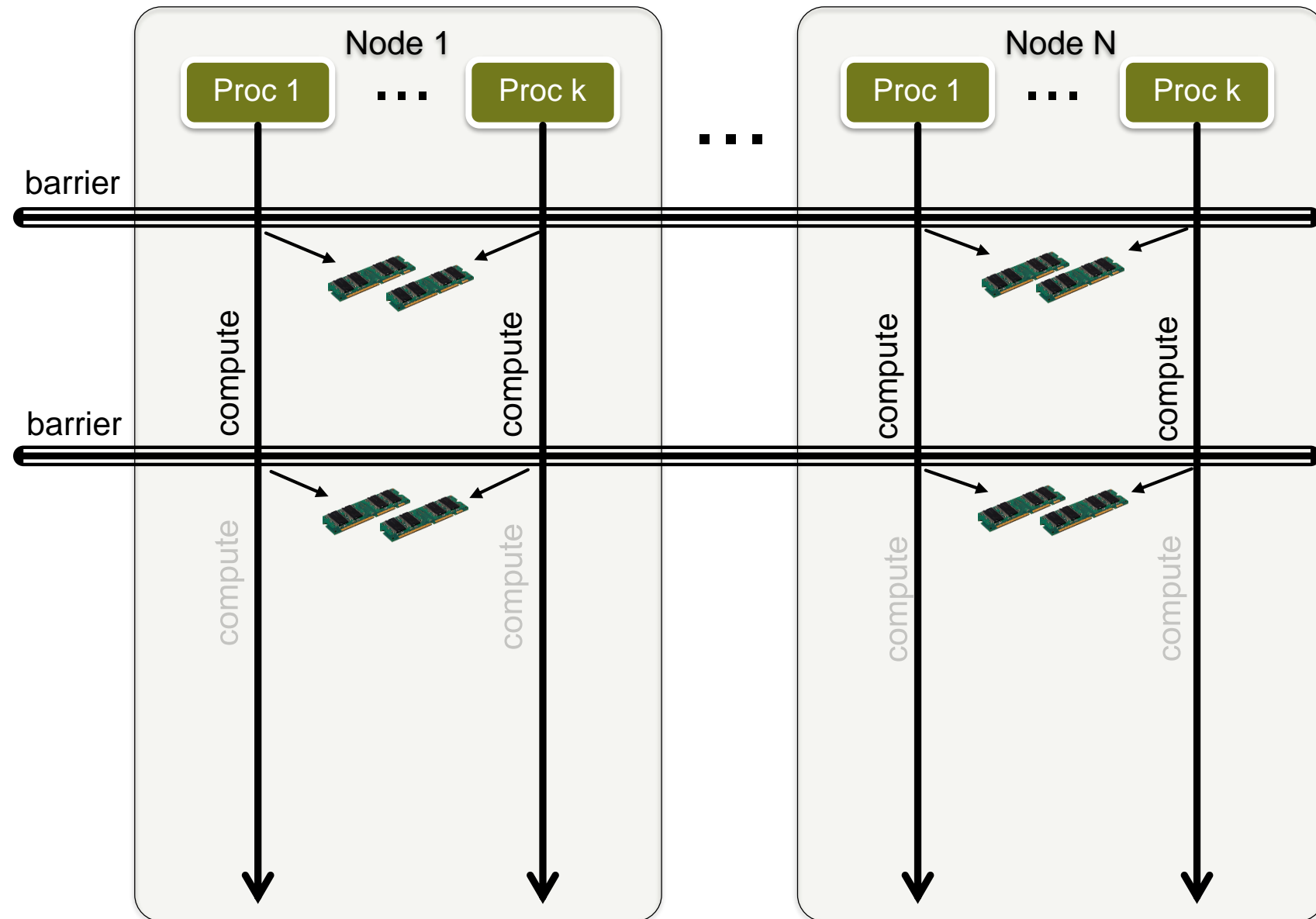
COORDINATED CHECKPOINTING (MP)



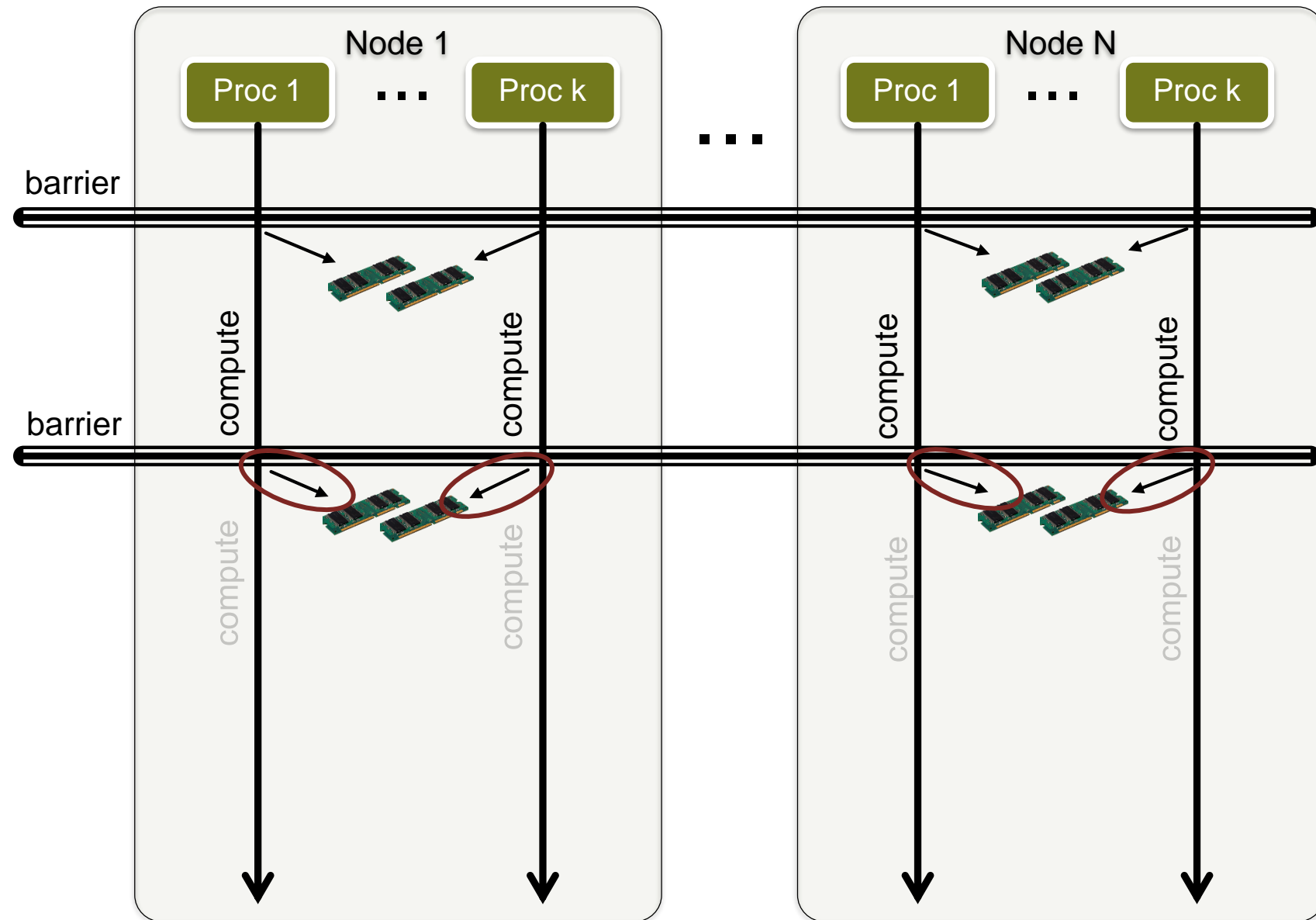
COORDINATED CHECKPOINTING (MP)



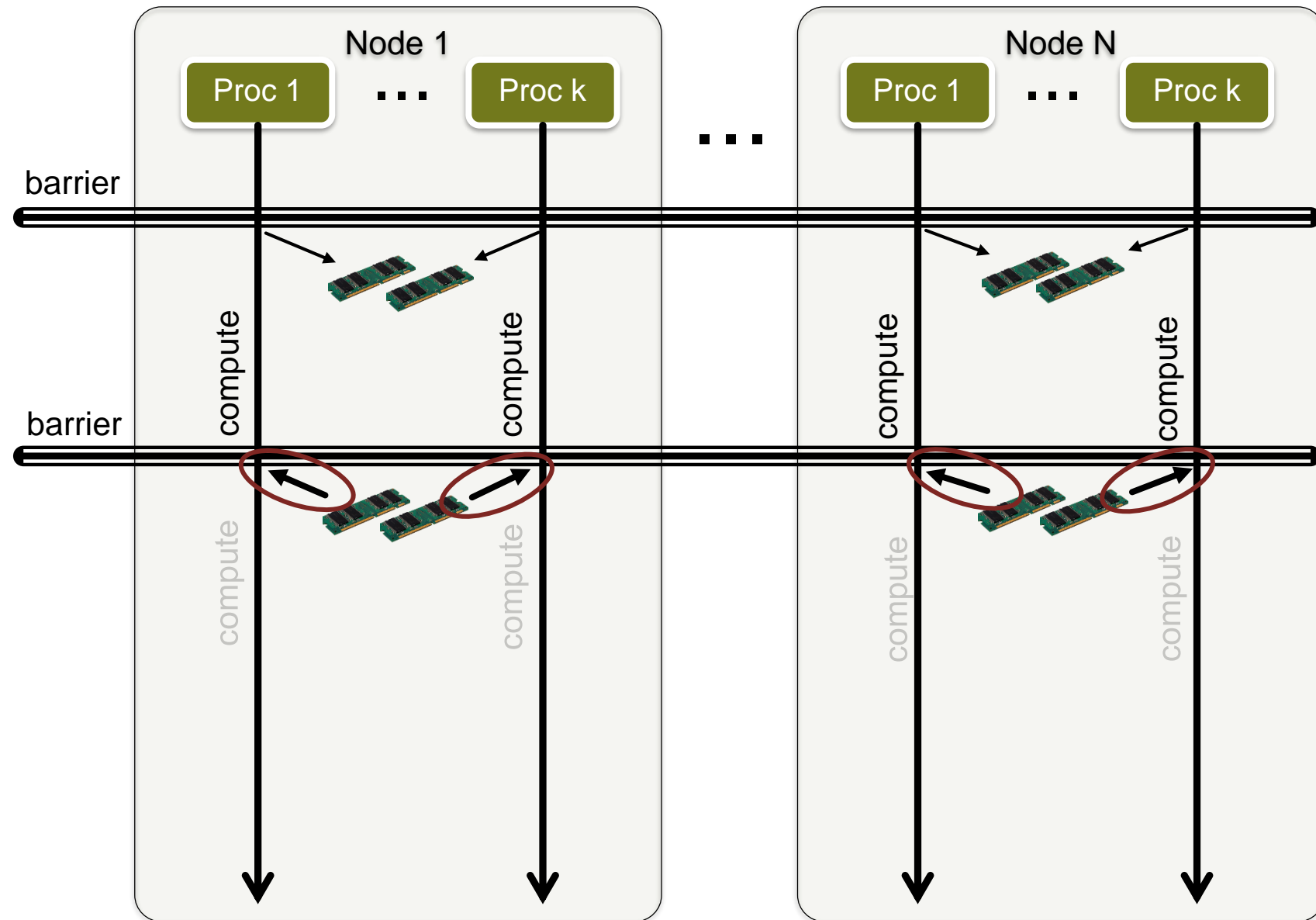
COORDINATED CHECKPOINTING (MP)



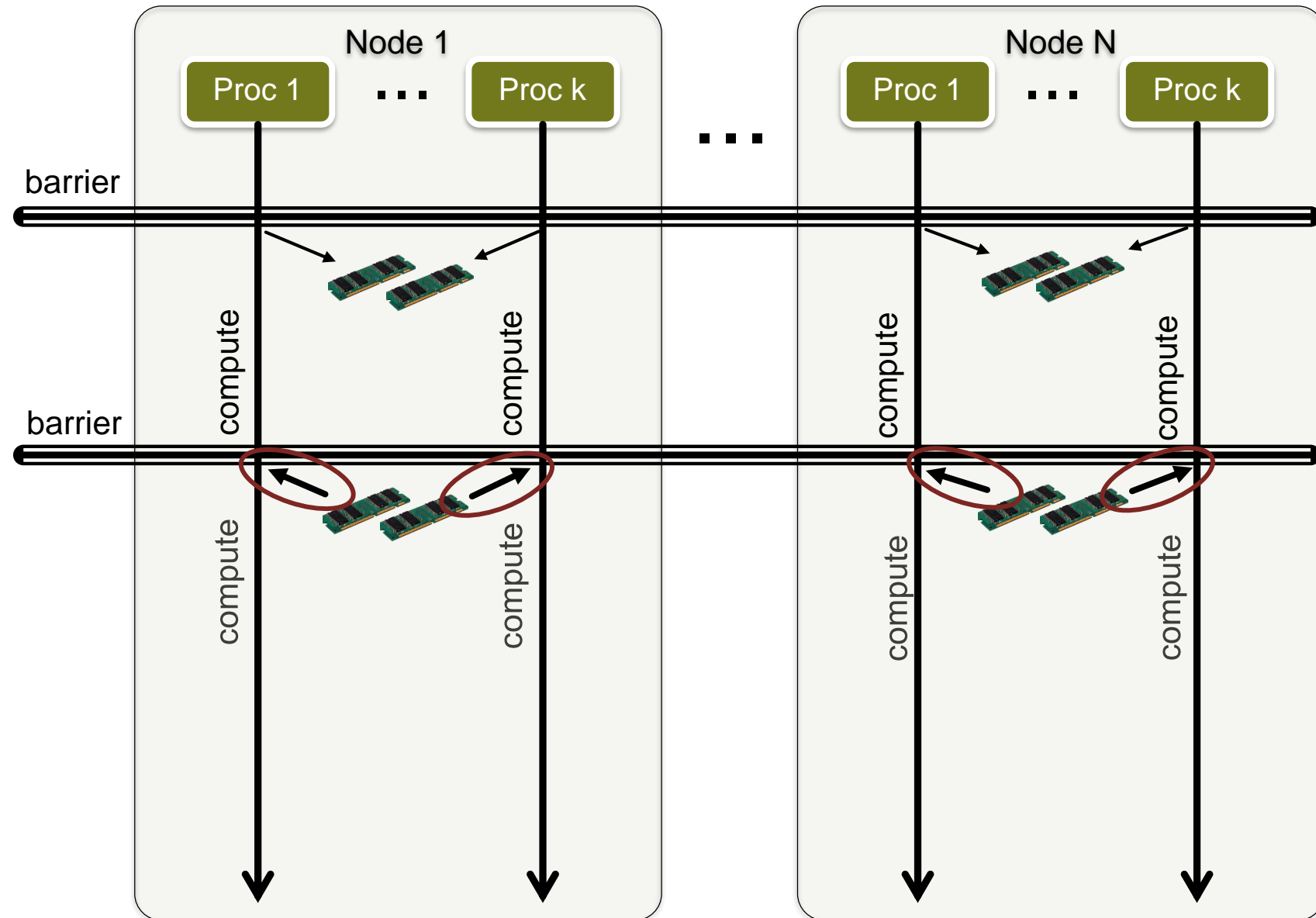
COORDINATED CHECKPOINTING (MP)



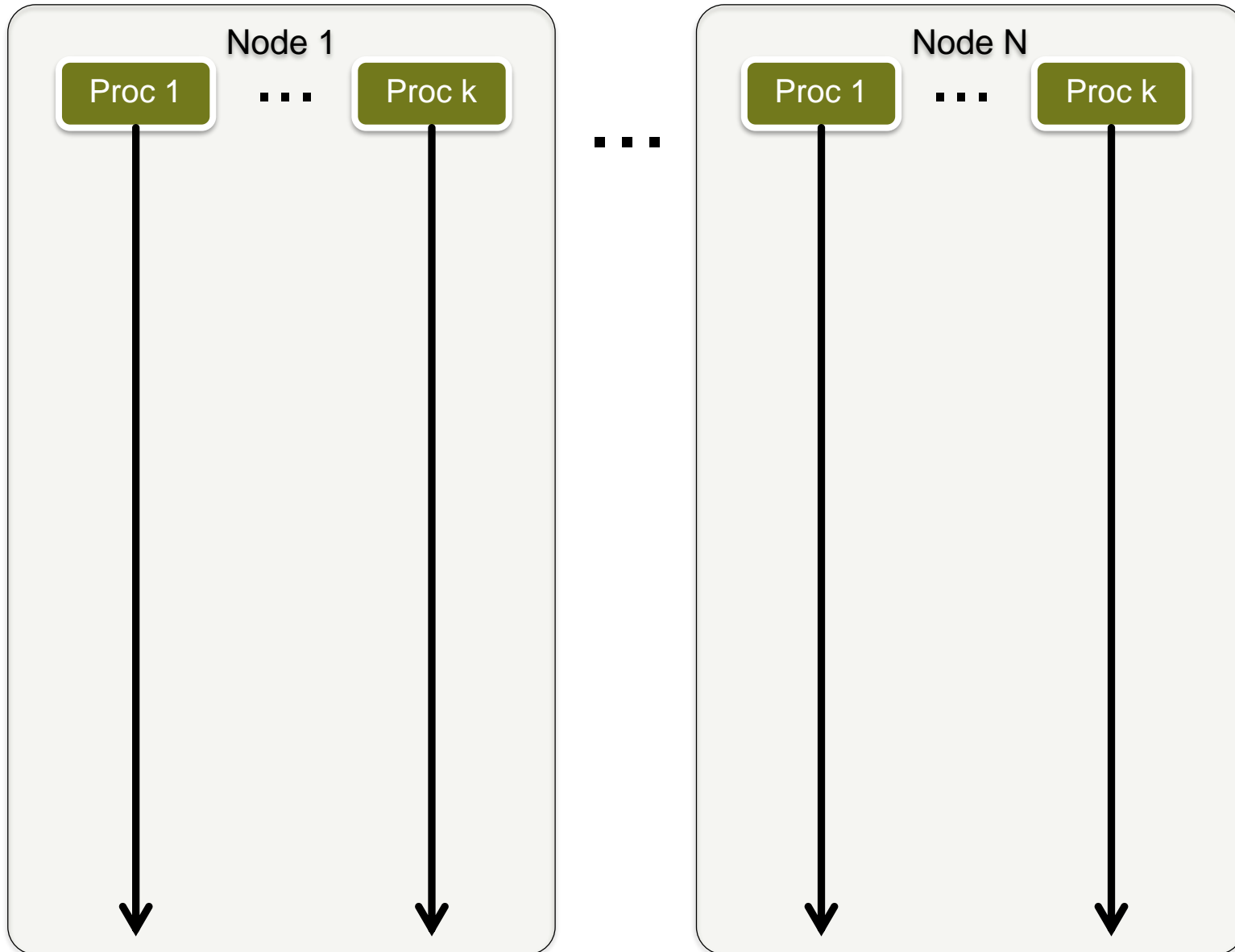
COORDINATED CHECKPOINTING (MP)



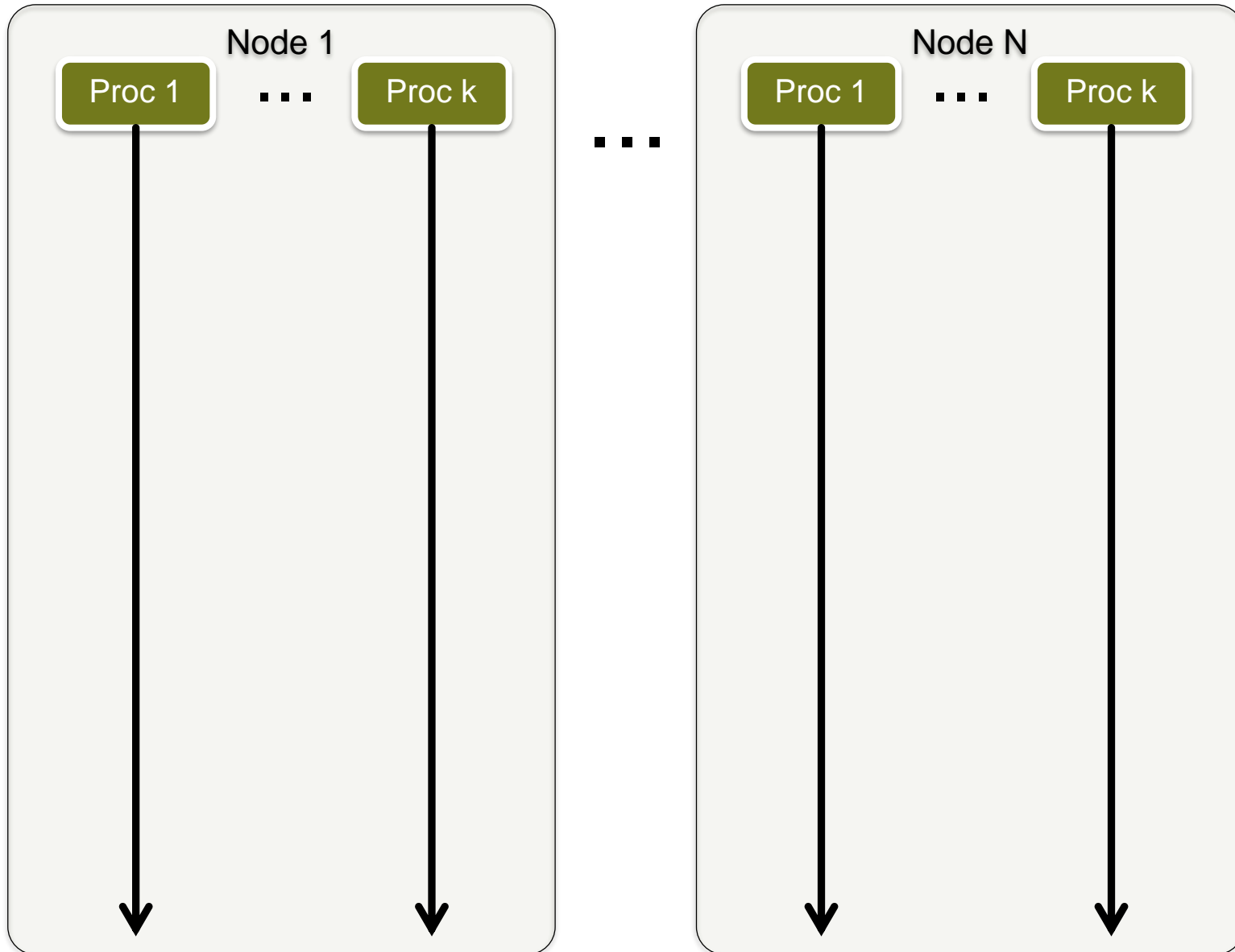
COORDINATED CHECKPOINTING (MP)



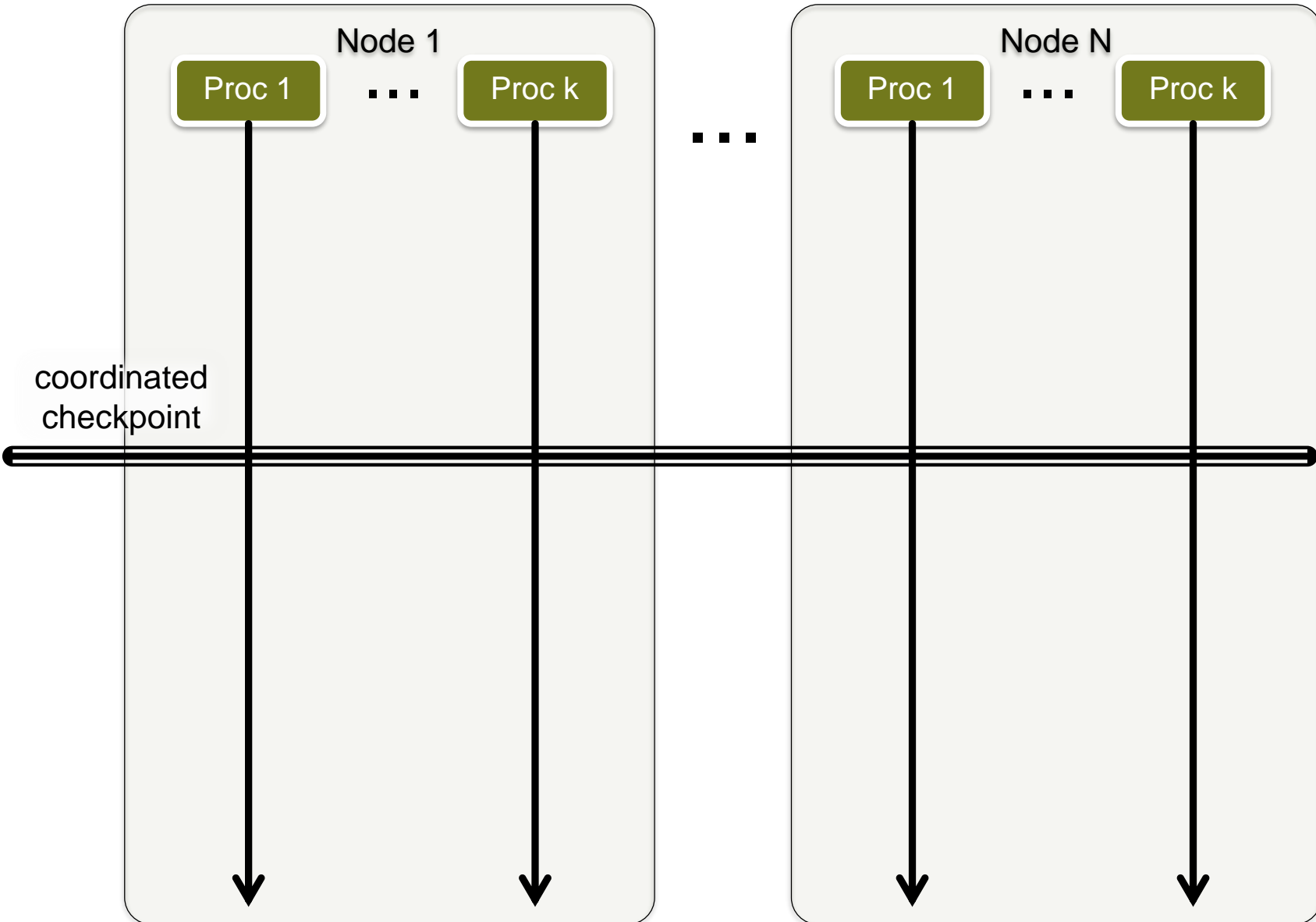
COORDINATED CHECKPOINTING (MP)



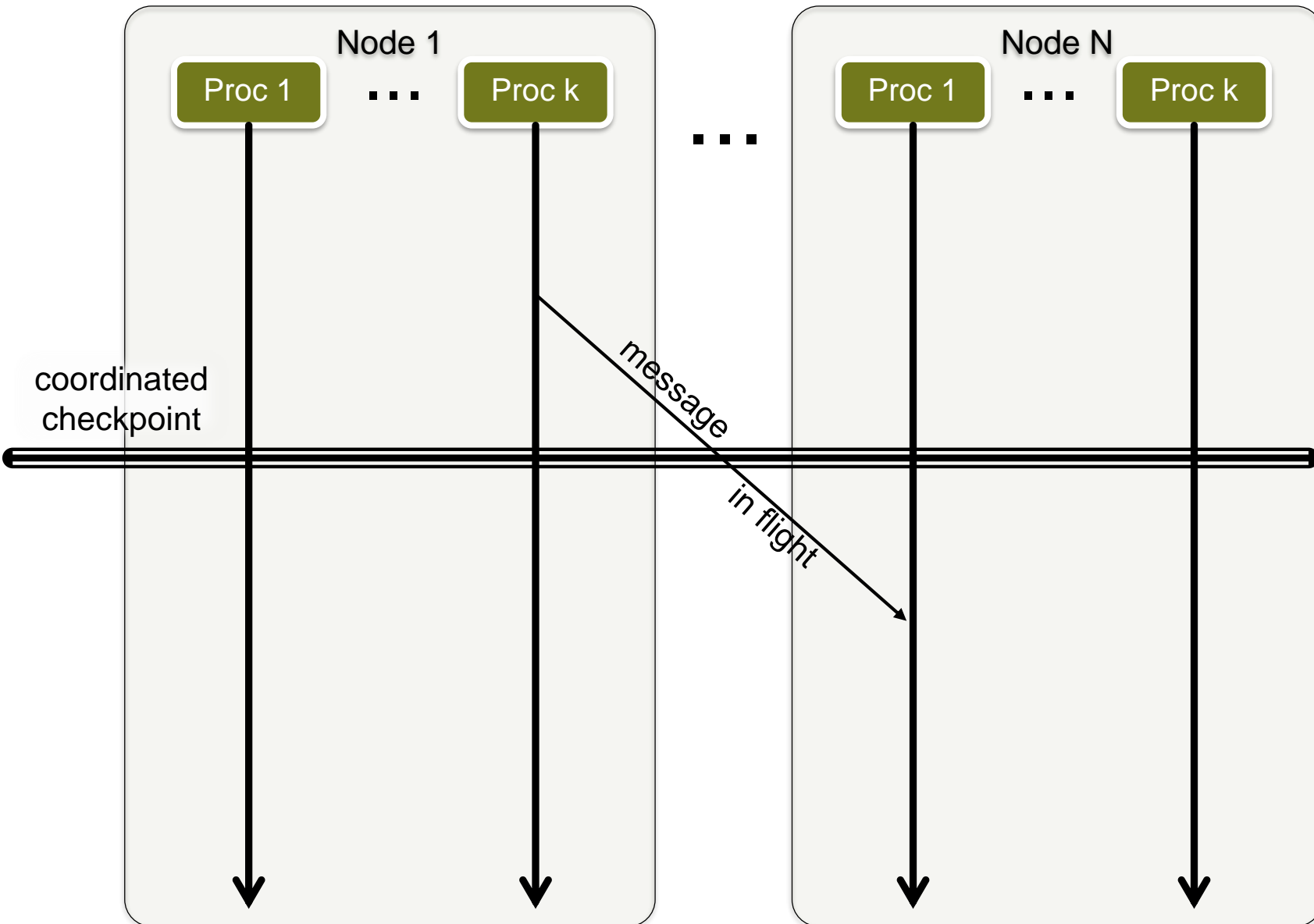
COORDINATED CHECKPOINTING (MP)



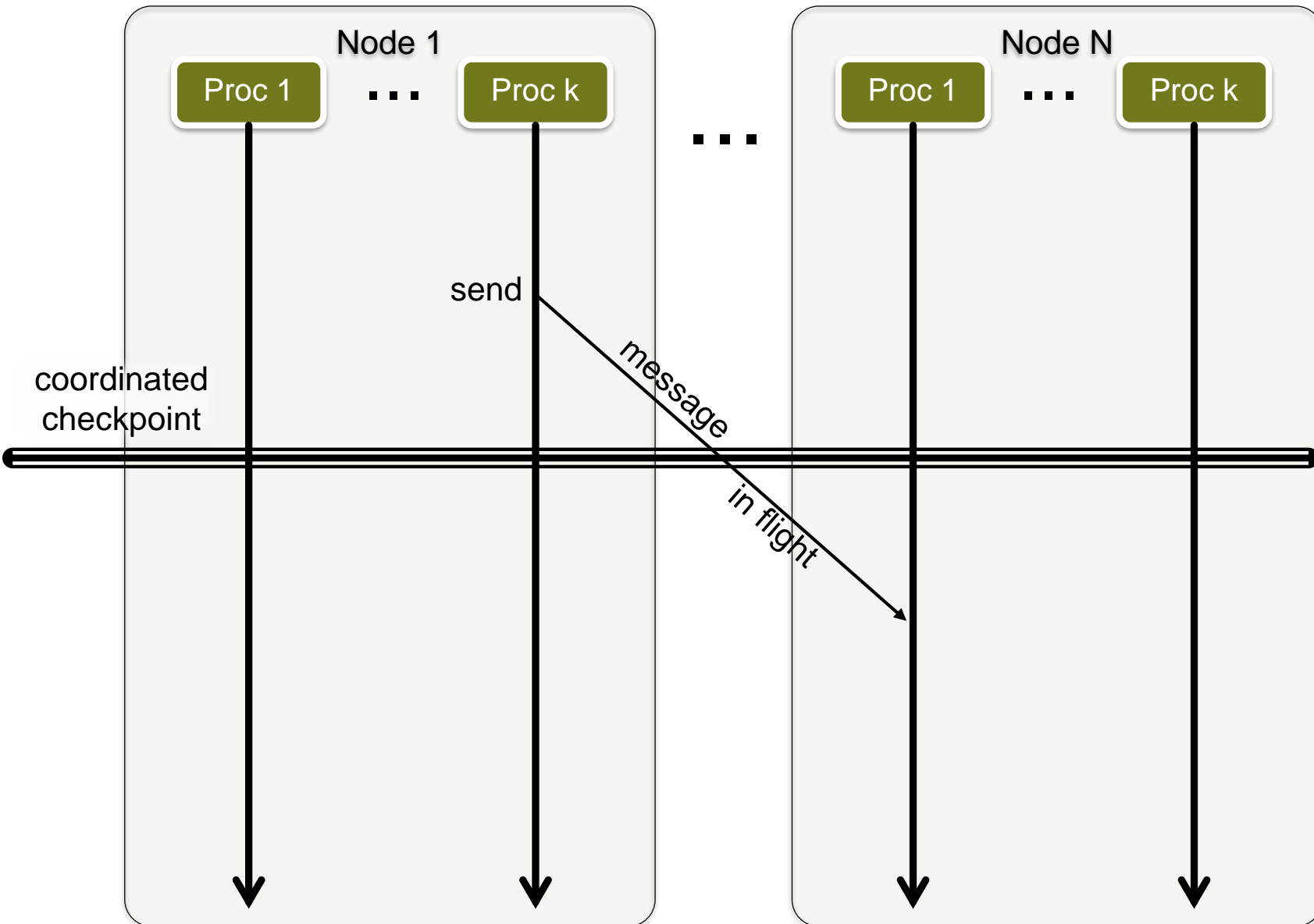
COORDINATED CHECKPOINTING (MP)



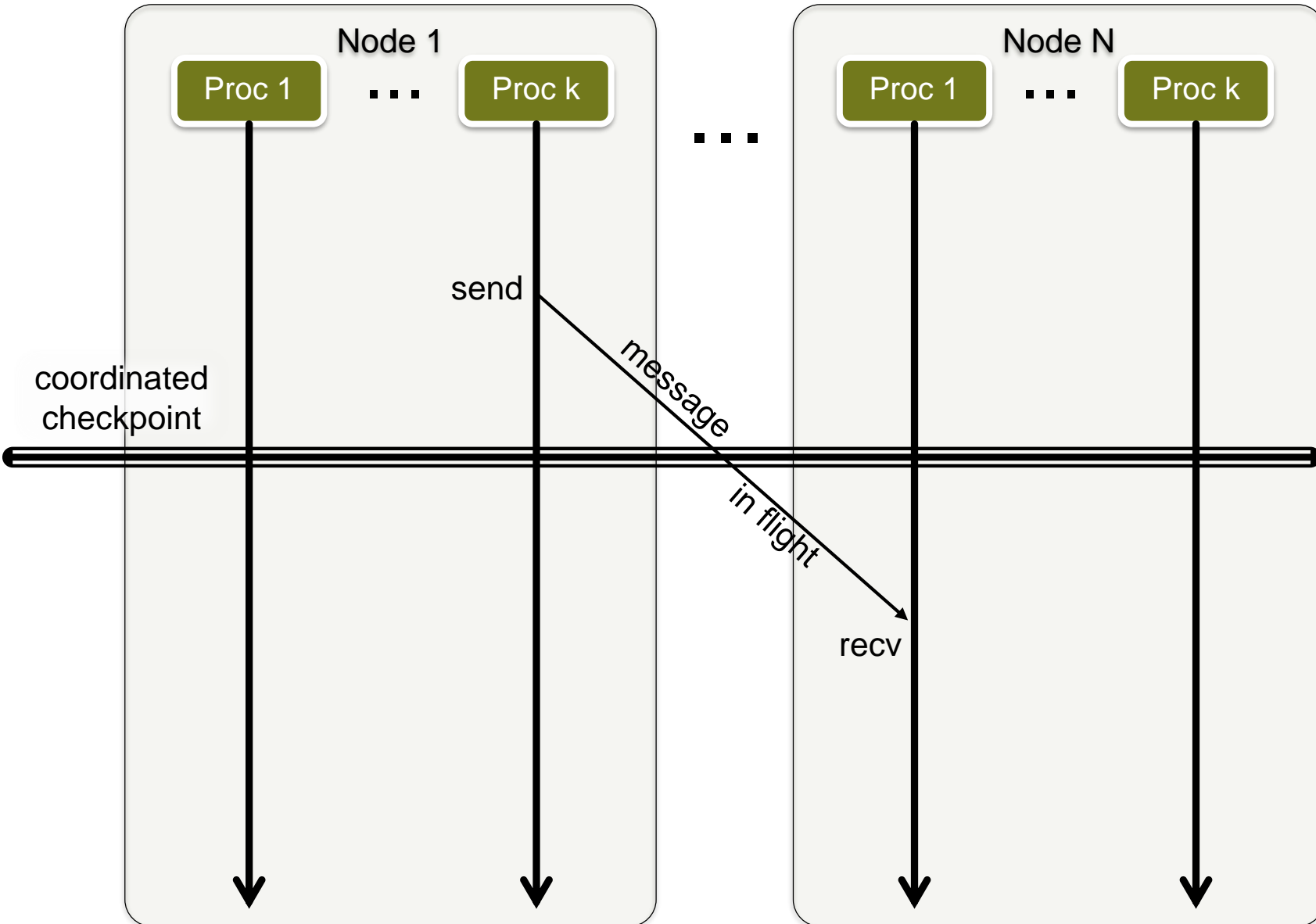
COORDINATED CHECKPOINTING (MP)



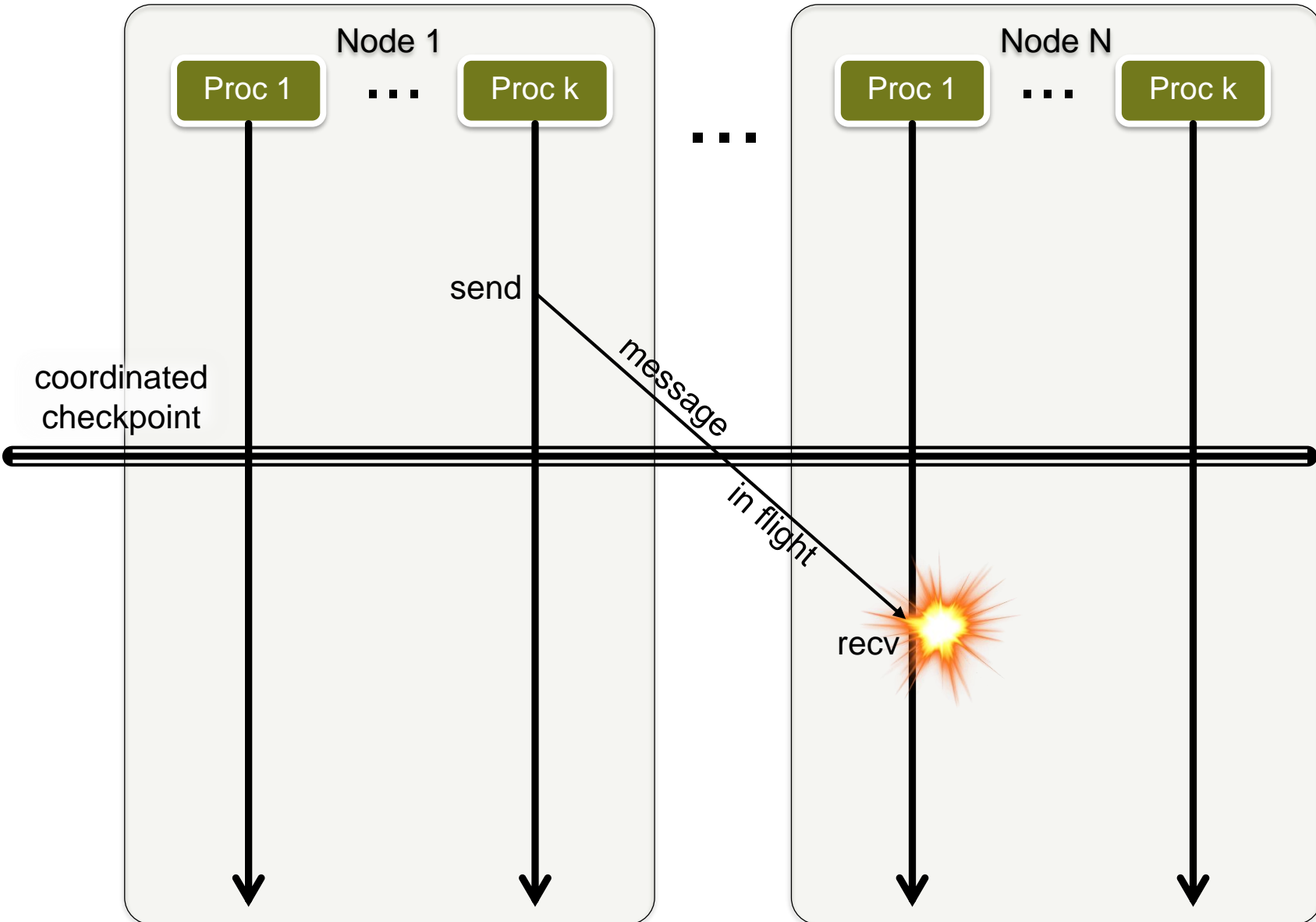
COORDINATED CHECKPOINTING (MP)



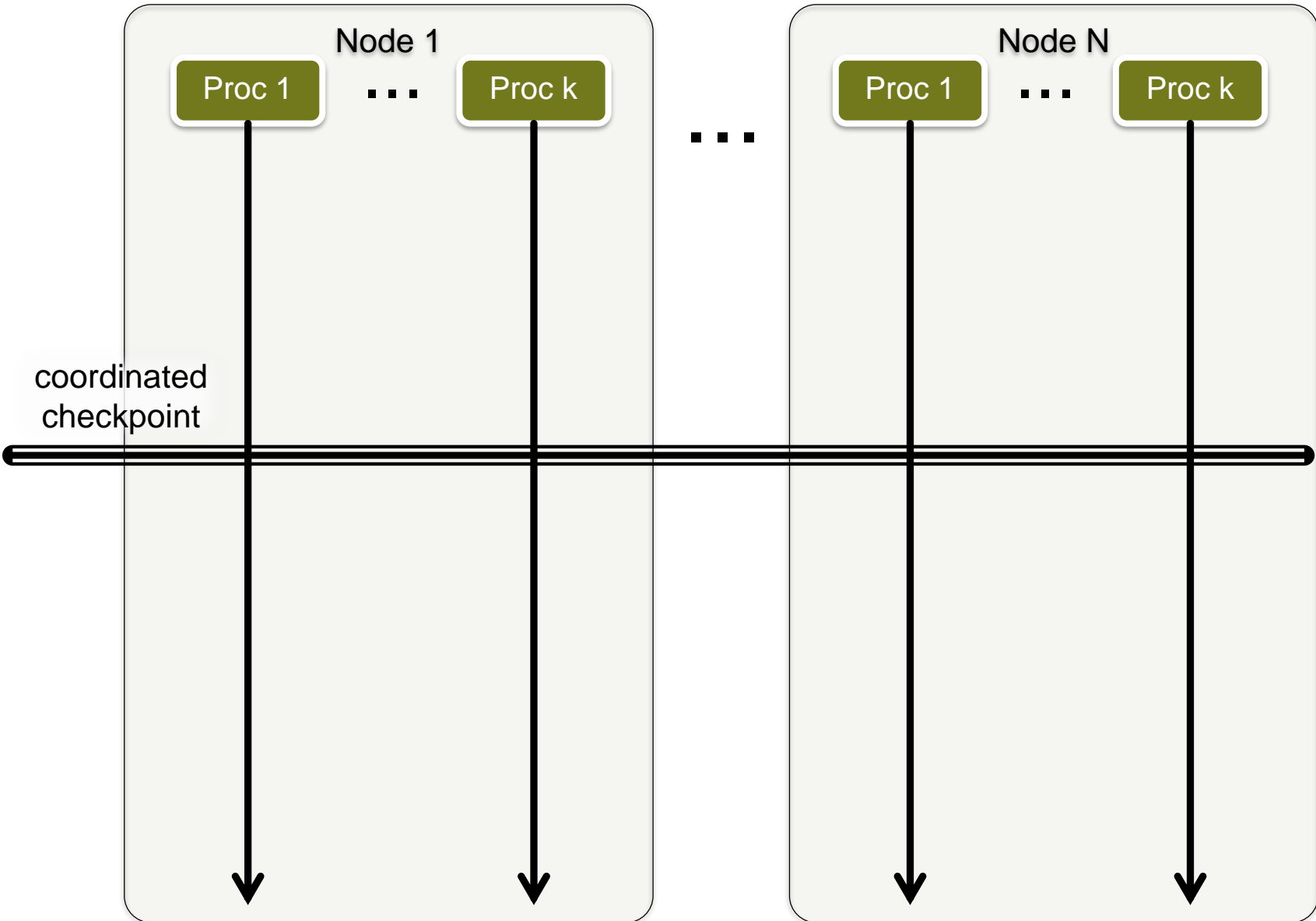
COORDINATED CHECKPOINTING (MP)



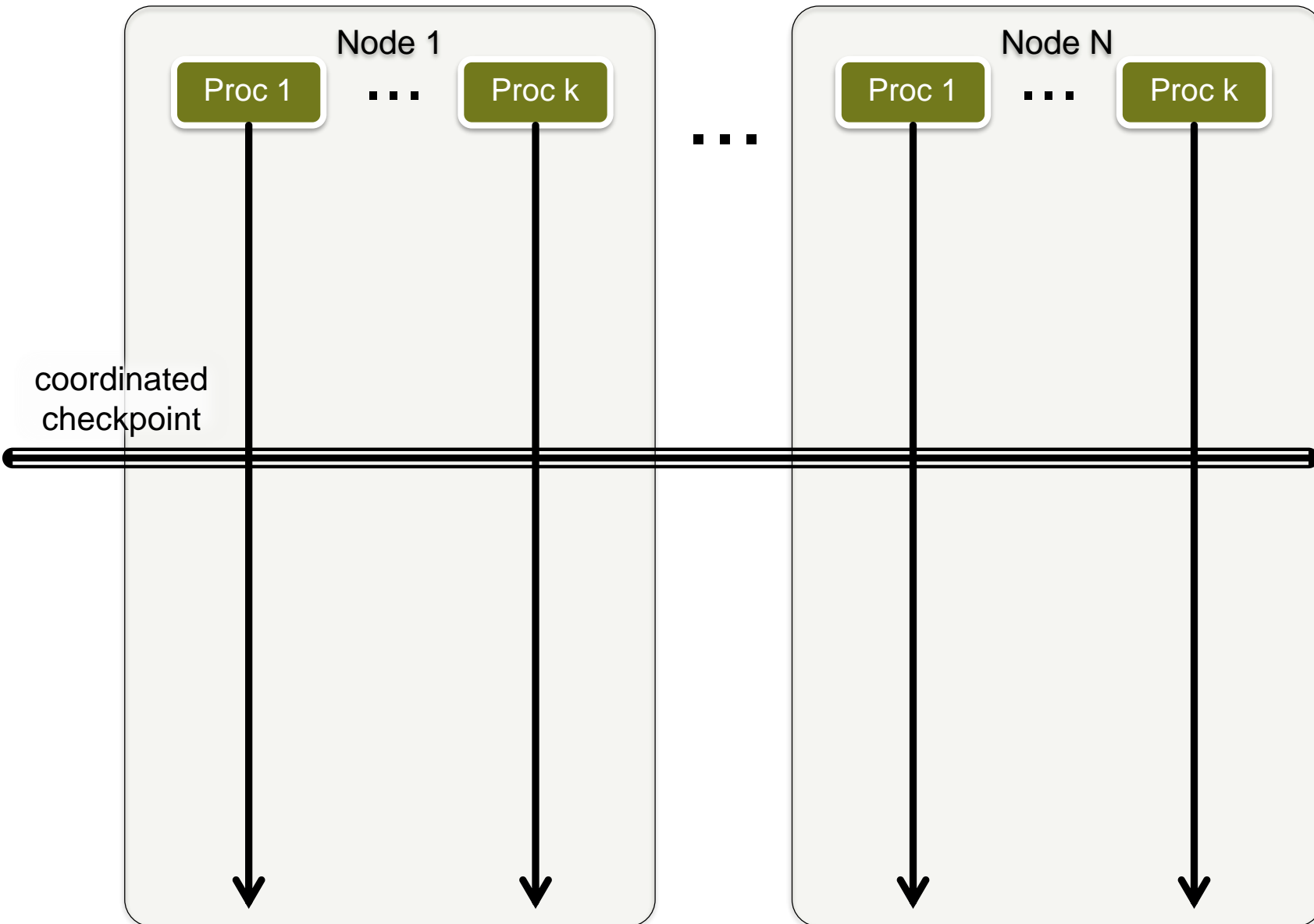
COORDINATED CHECKPOINTING (MP)



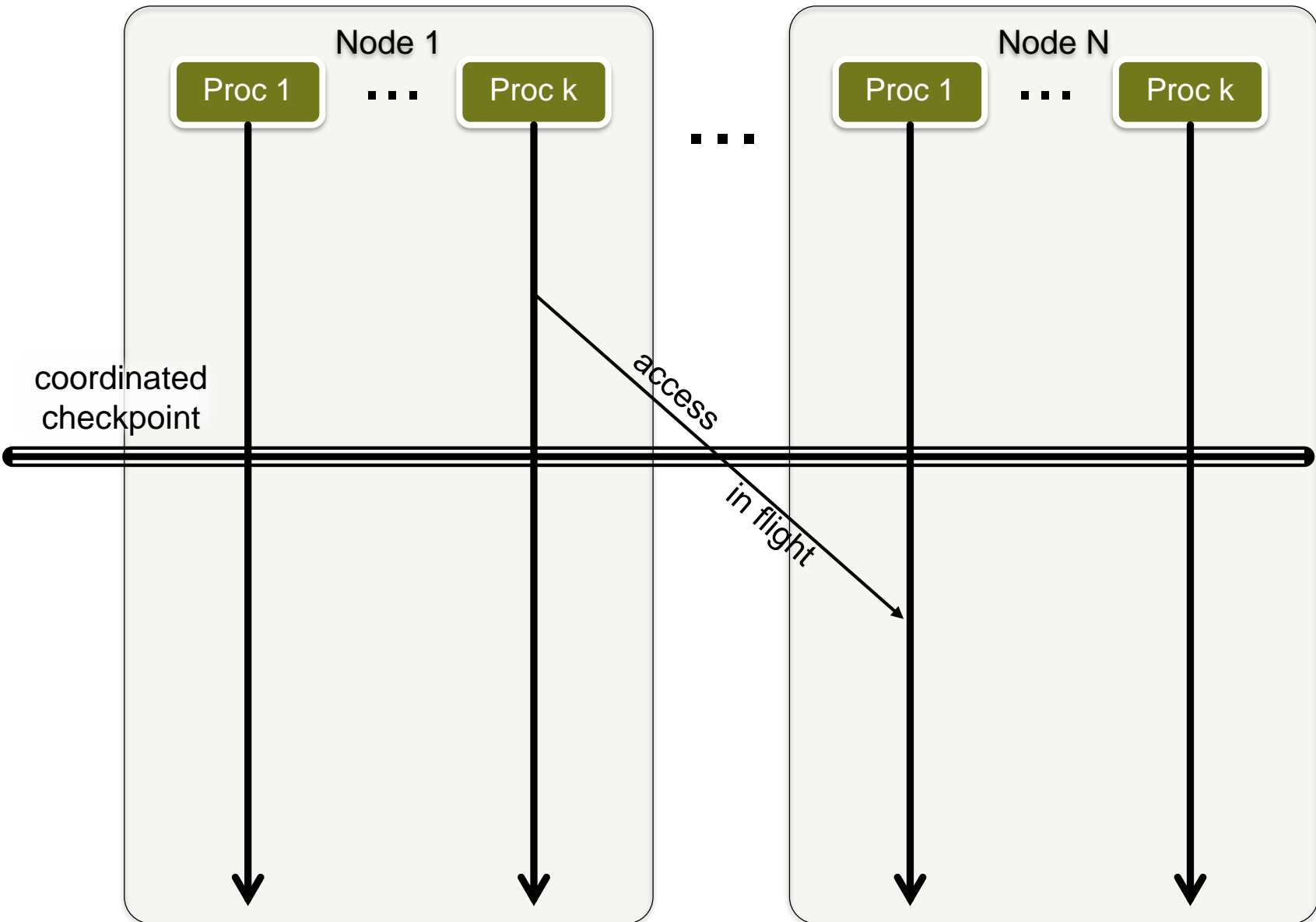
COORDINATED CHECKPOINTING (MP)



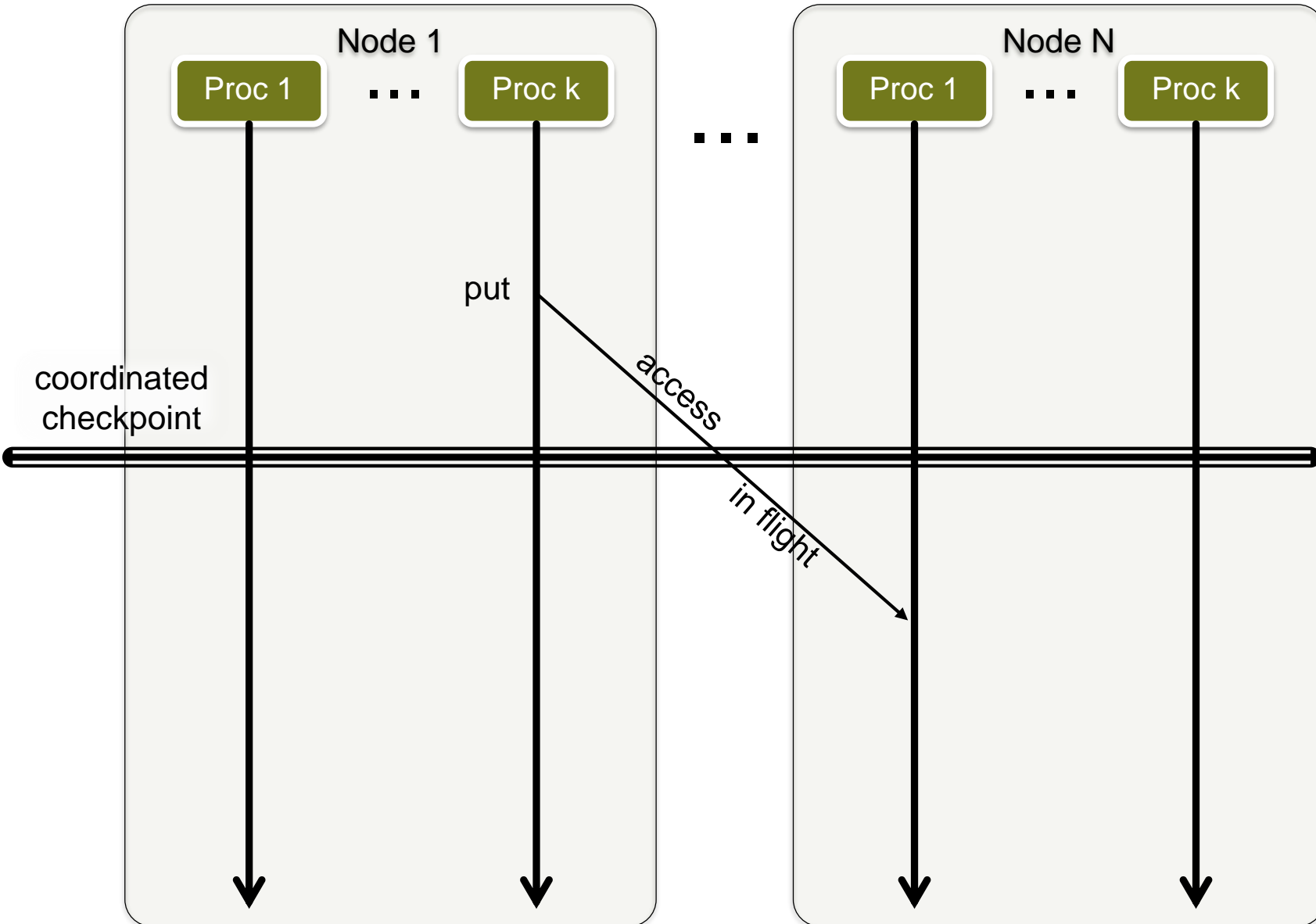
COORDINATED CHECKPOINTING (RMA)



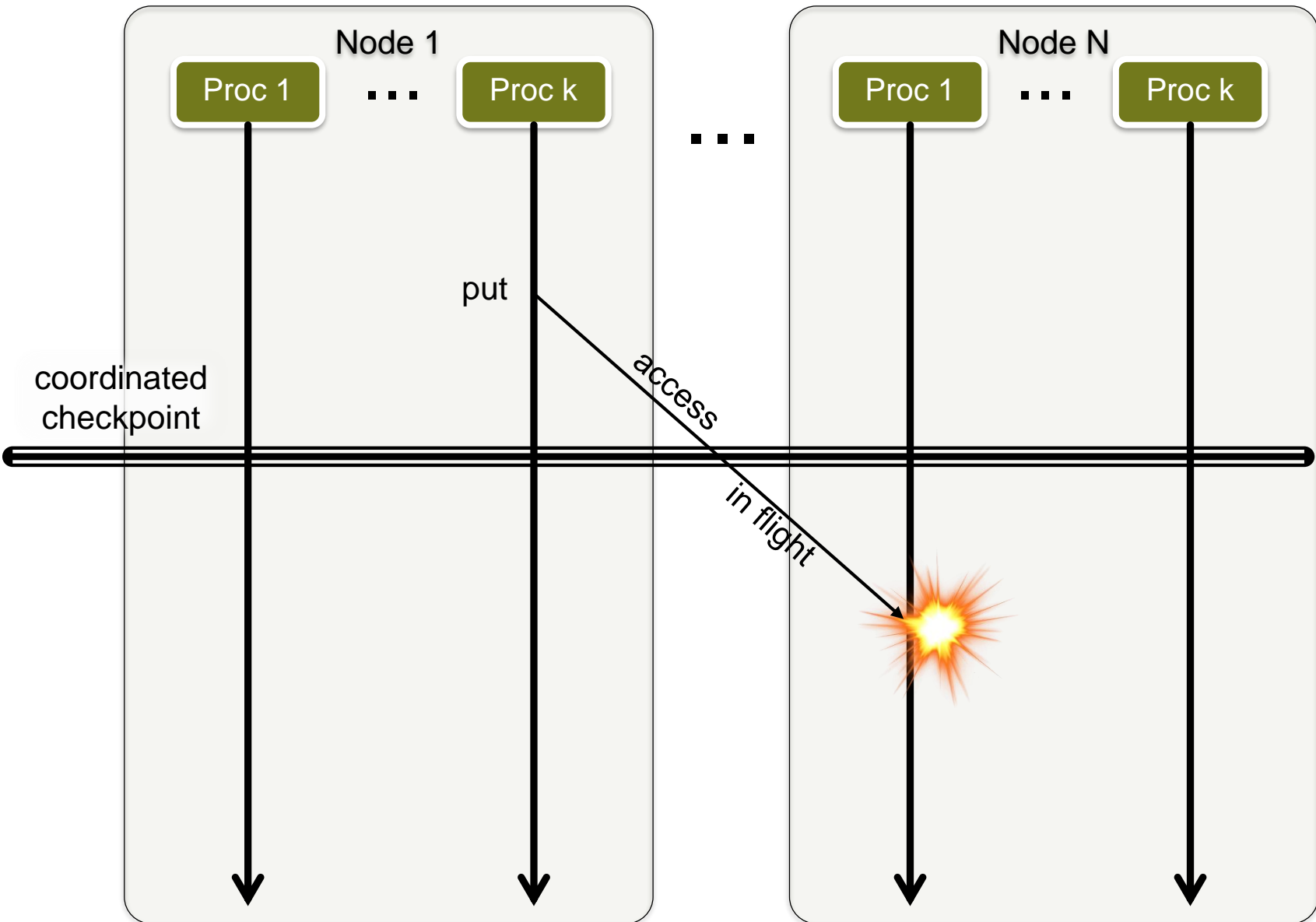
COORDINATED CHECKPOINTING (RMA)



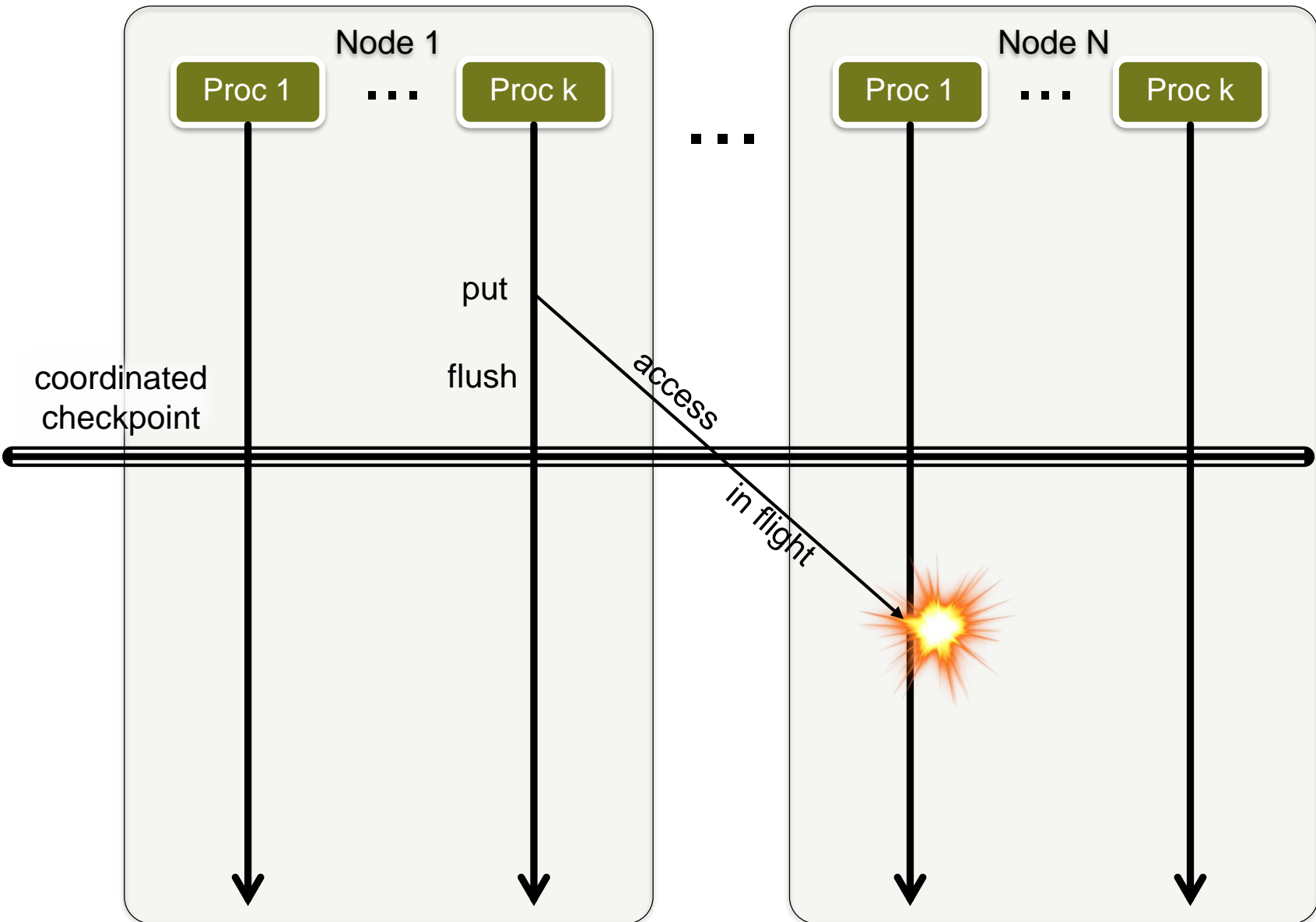
COORDINATED CHECKPOINTING (RMA)



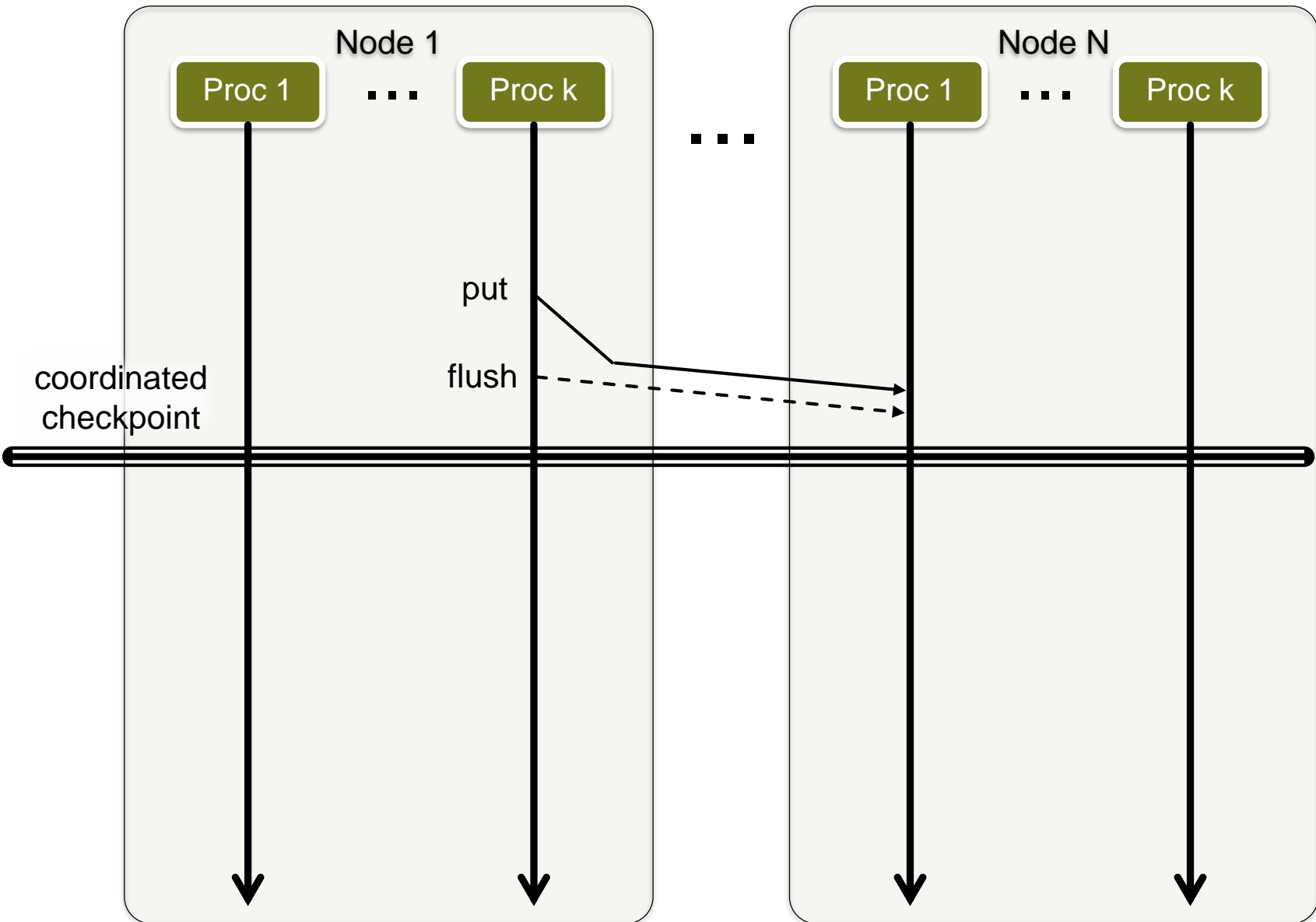
COORDINATED CHECKPOINTING (RMA)



COORDINATED CHECKPOINTING (RMA)



COORDINATED CHECKPOINTING (RMA)



OVERVIEW OF OUR RESEARCH

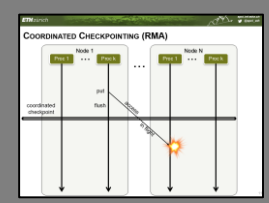
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

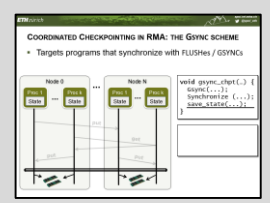
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

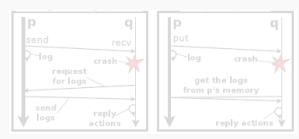


MP vs. RMA

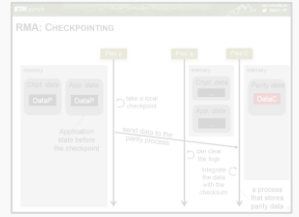


Schemes

UC in RMA

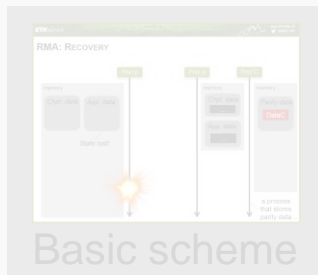


MP vs. RMA



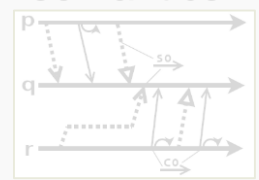
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the cohb order (as the gsync order).*

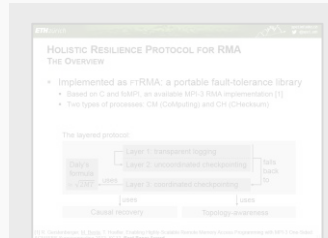
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

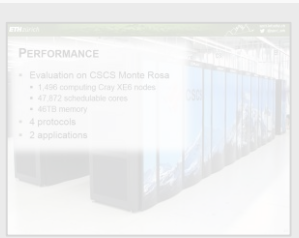
Holistic fault-tolerance library



Design

Checkpoints on demand

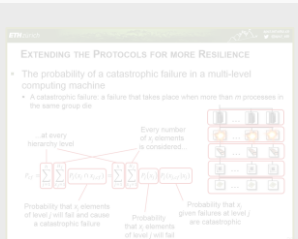
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

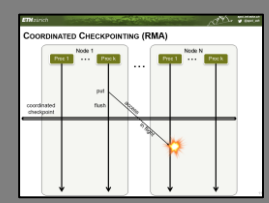
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

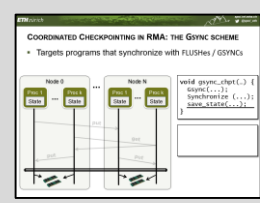
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

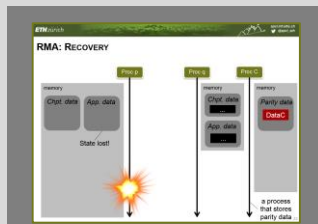


MP vs. RMA



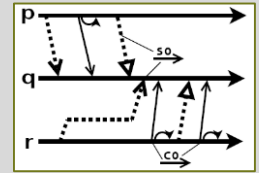
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

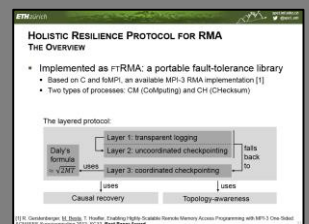


Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the cohb order to as the gsync order).*

Deadlock freedom
Correct recovery

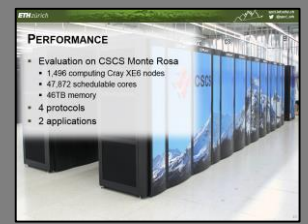
Holistic fault-tolerance library



Design

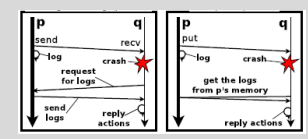
Checkpoints on demand

Optimizations

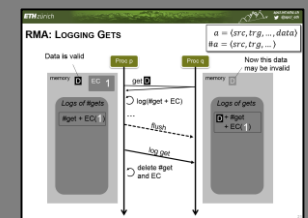


Performance

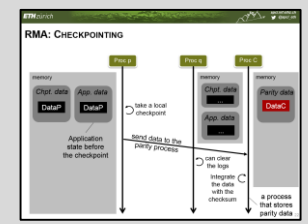
UC in RMA



MP vs. RMA



Logging accesses

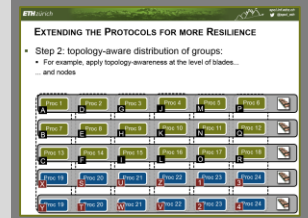


Checkpointing schemes

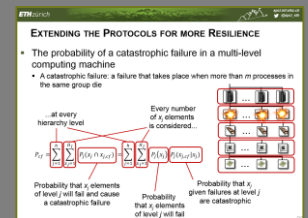
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



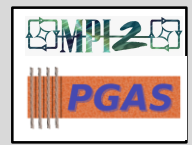
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

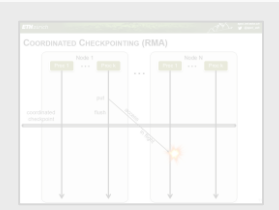
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

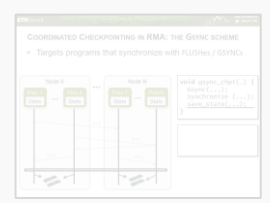
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

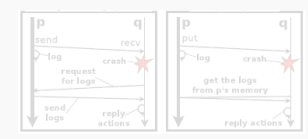


MP vs. RMA

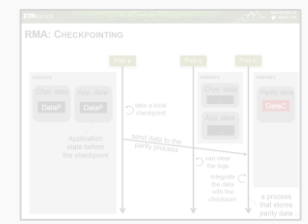


Schemes

UC in RMA

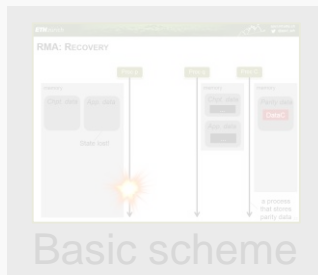


MP vs. RMA



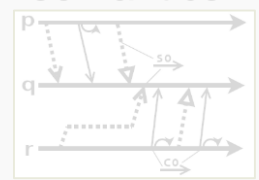
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

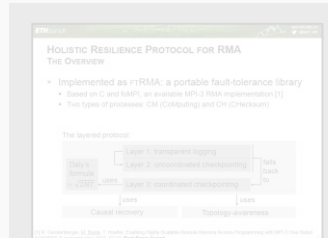
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

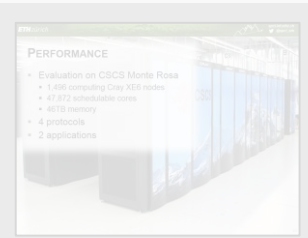
Holistic fault-tolerance library



Design

Checkpoints on demand

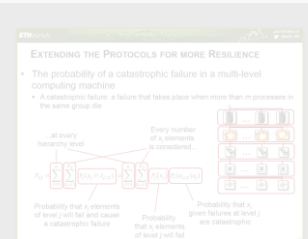
Optimizations



Performance

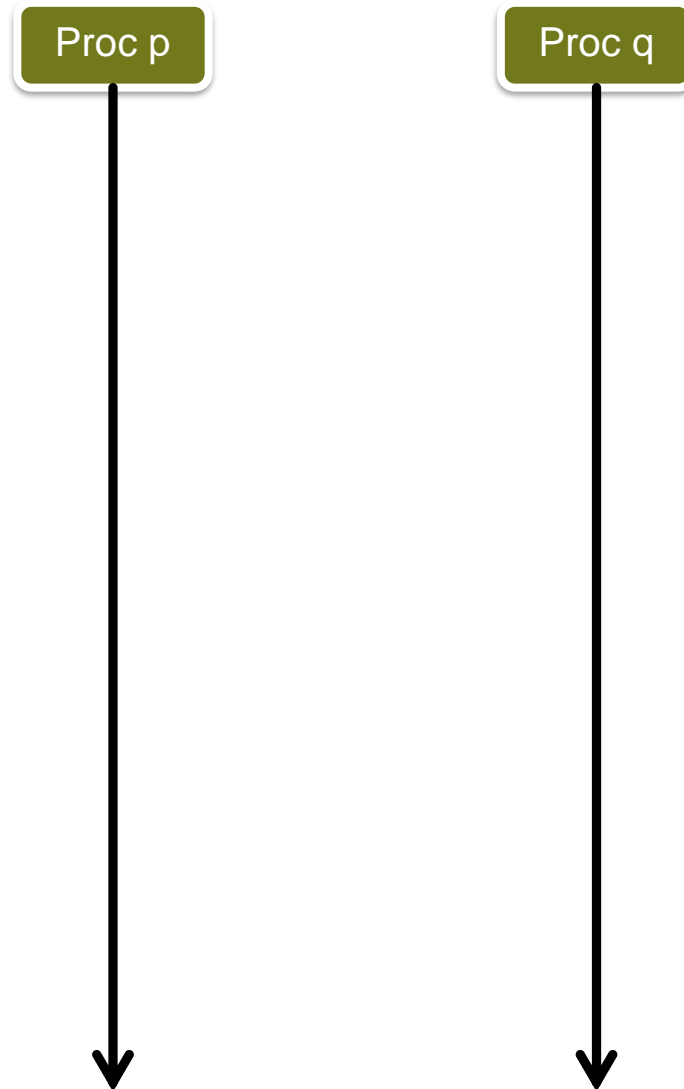


Distribution of processes

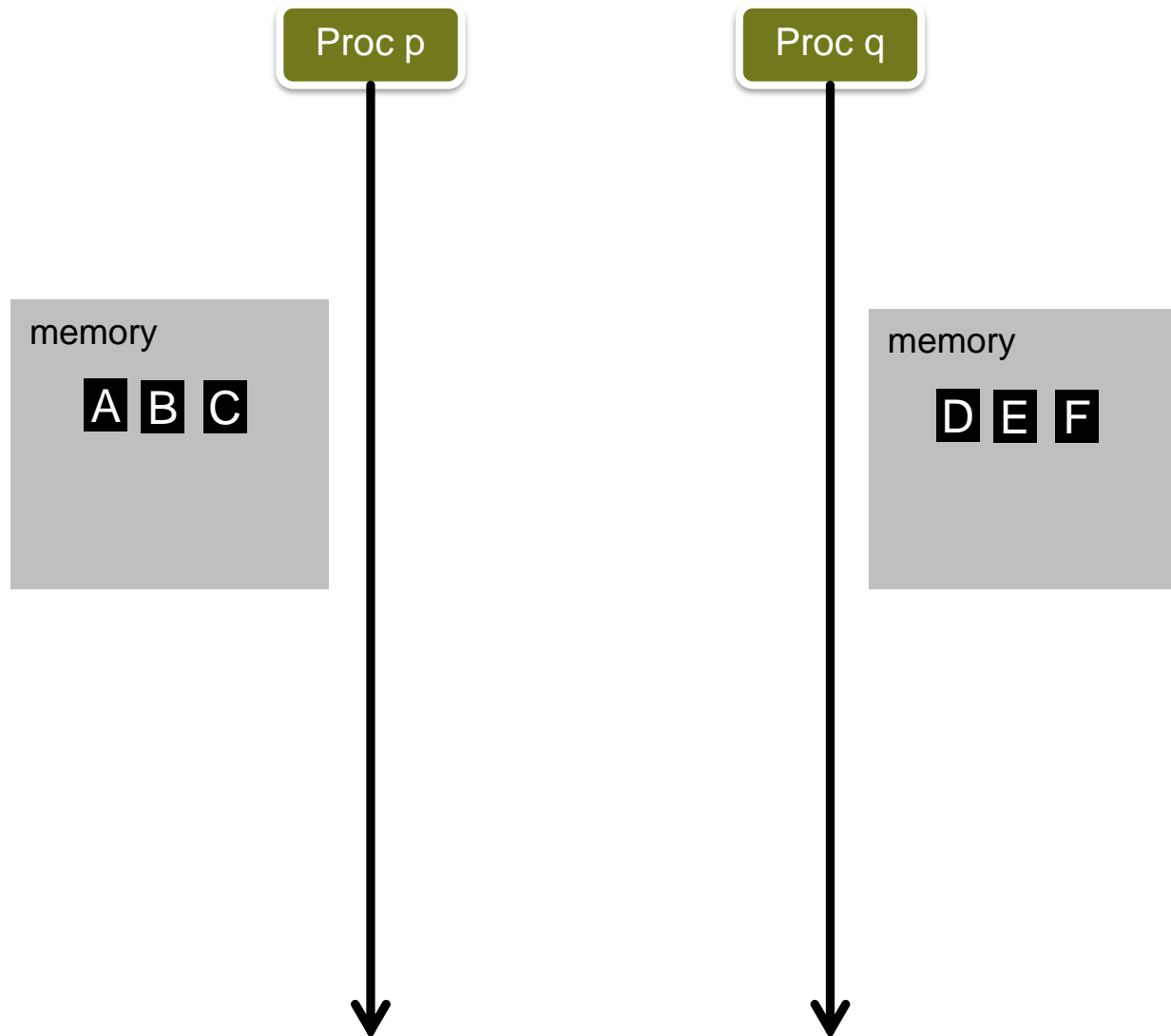


Decreasing failure prob.

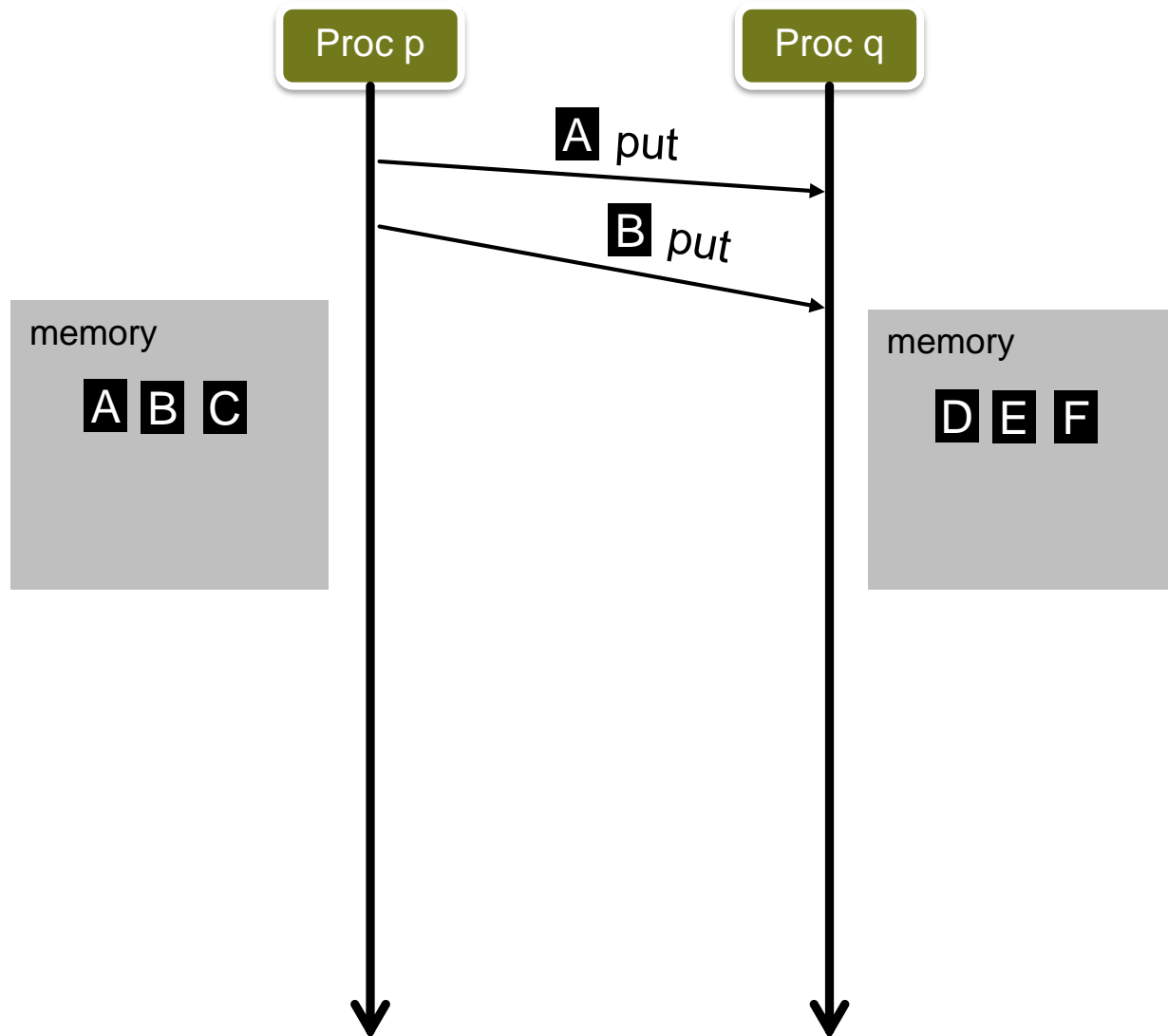
RMA: EPOCHS



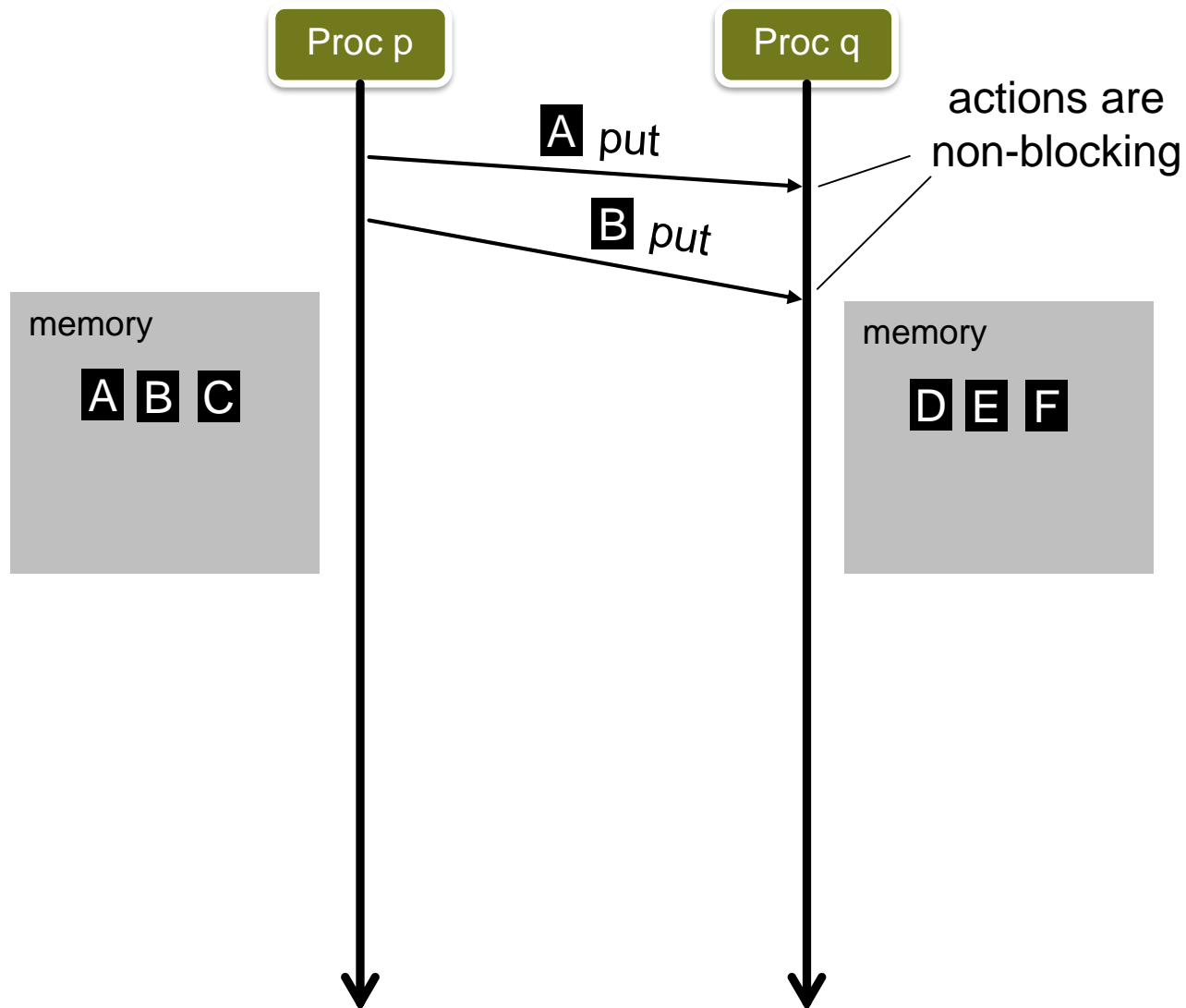
RMA: EPOCHS



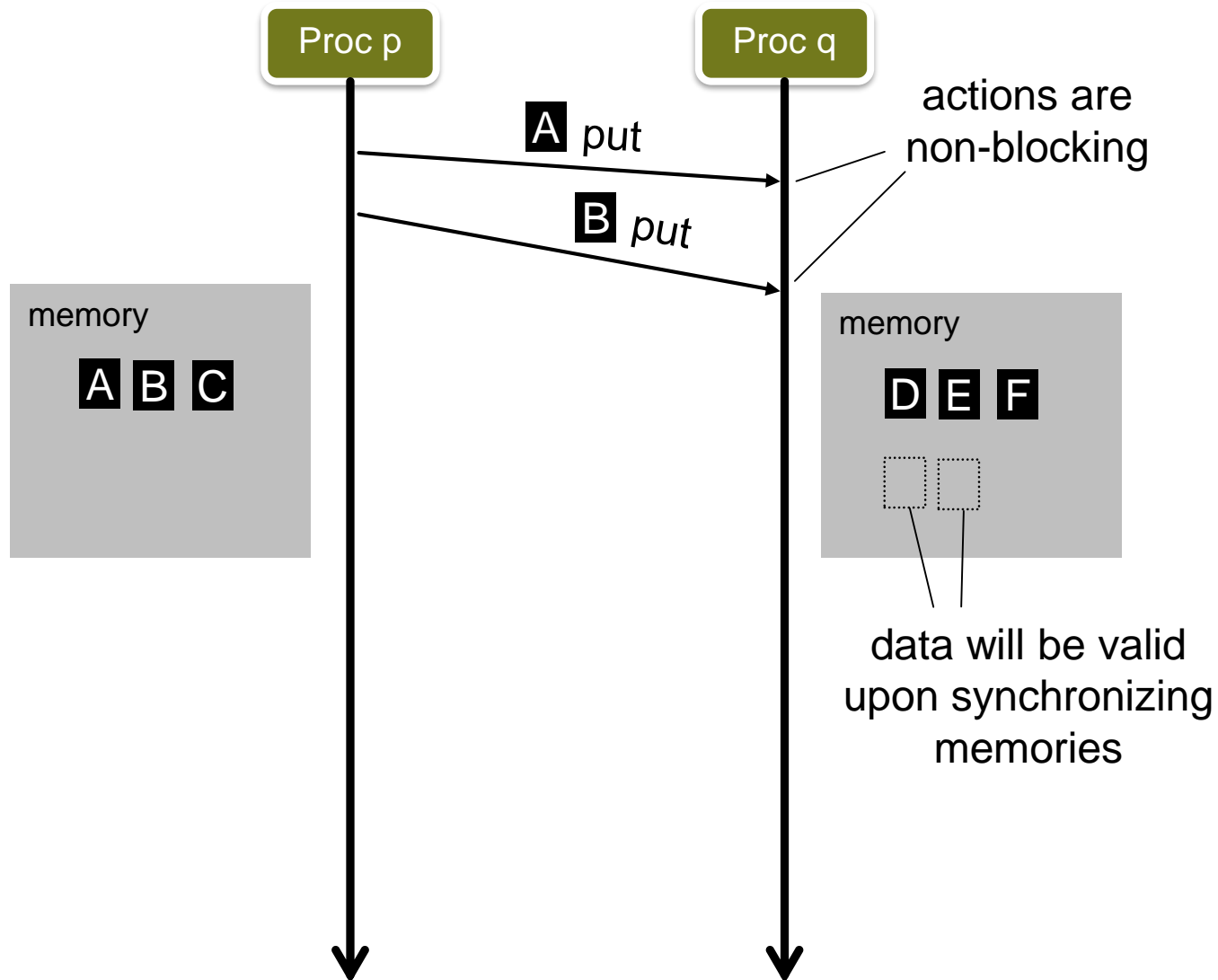
RMA: EPOCHS



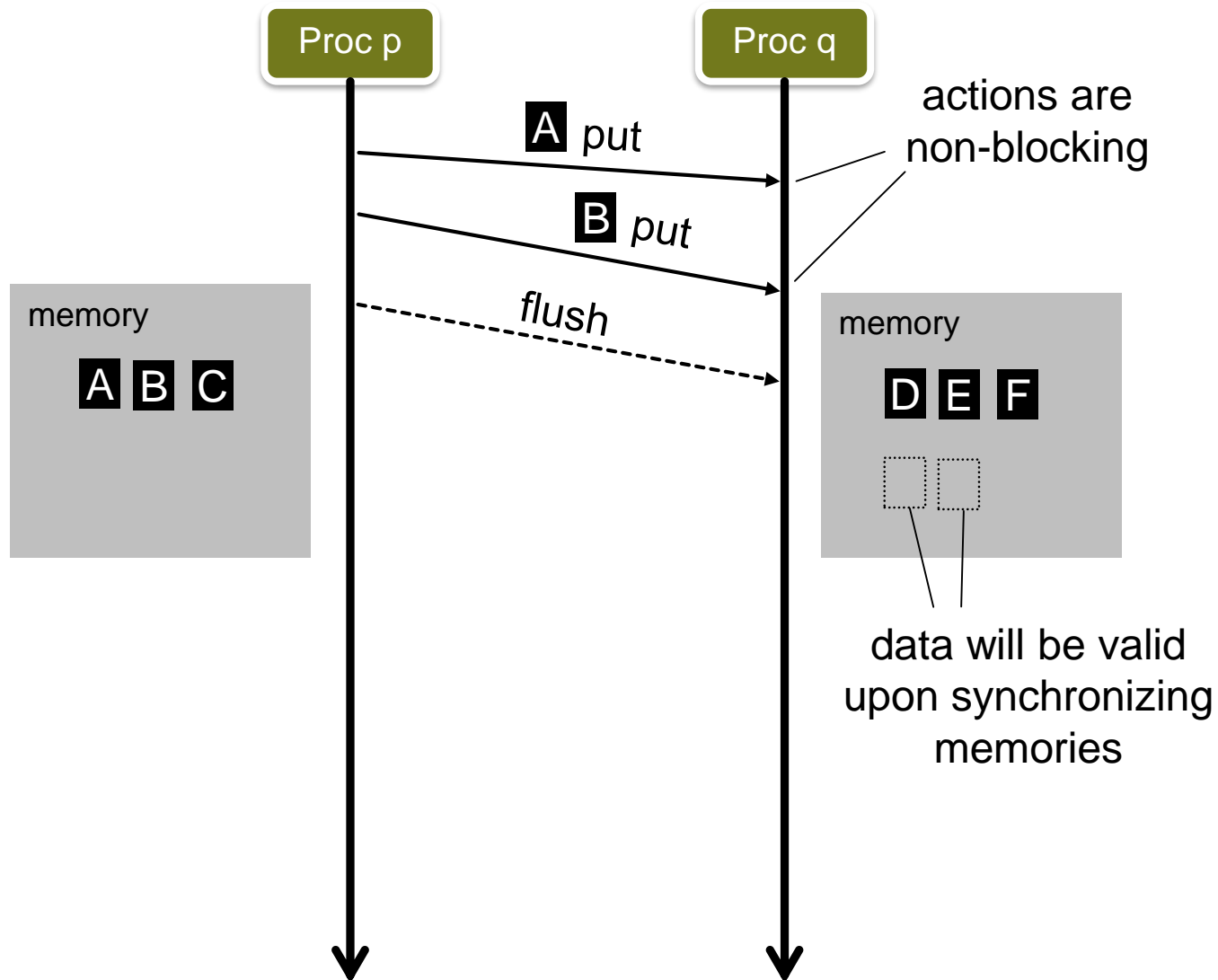
RMA: EPOCHS



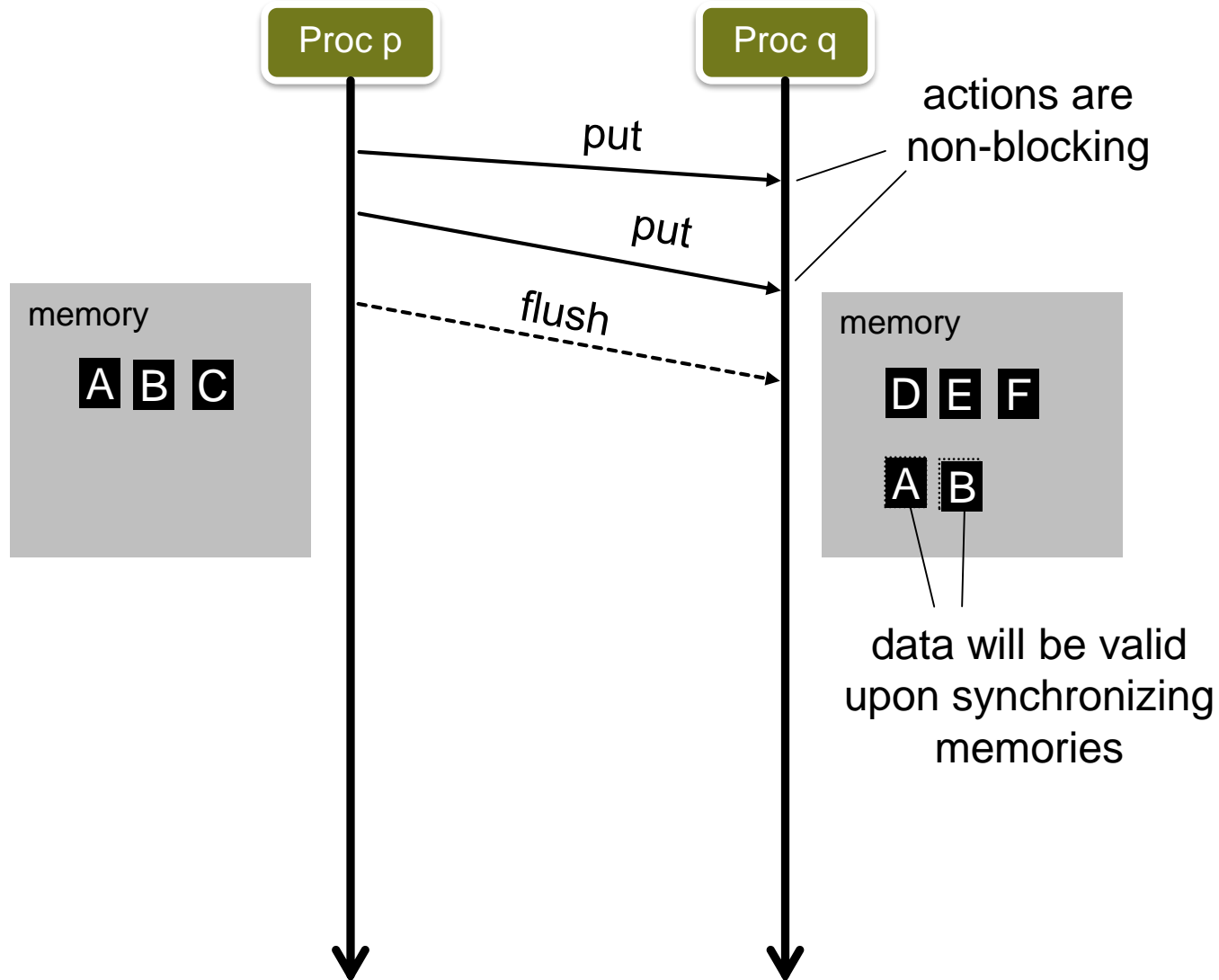
RMA: EPOCHS



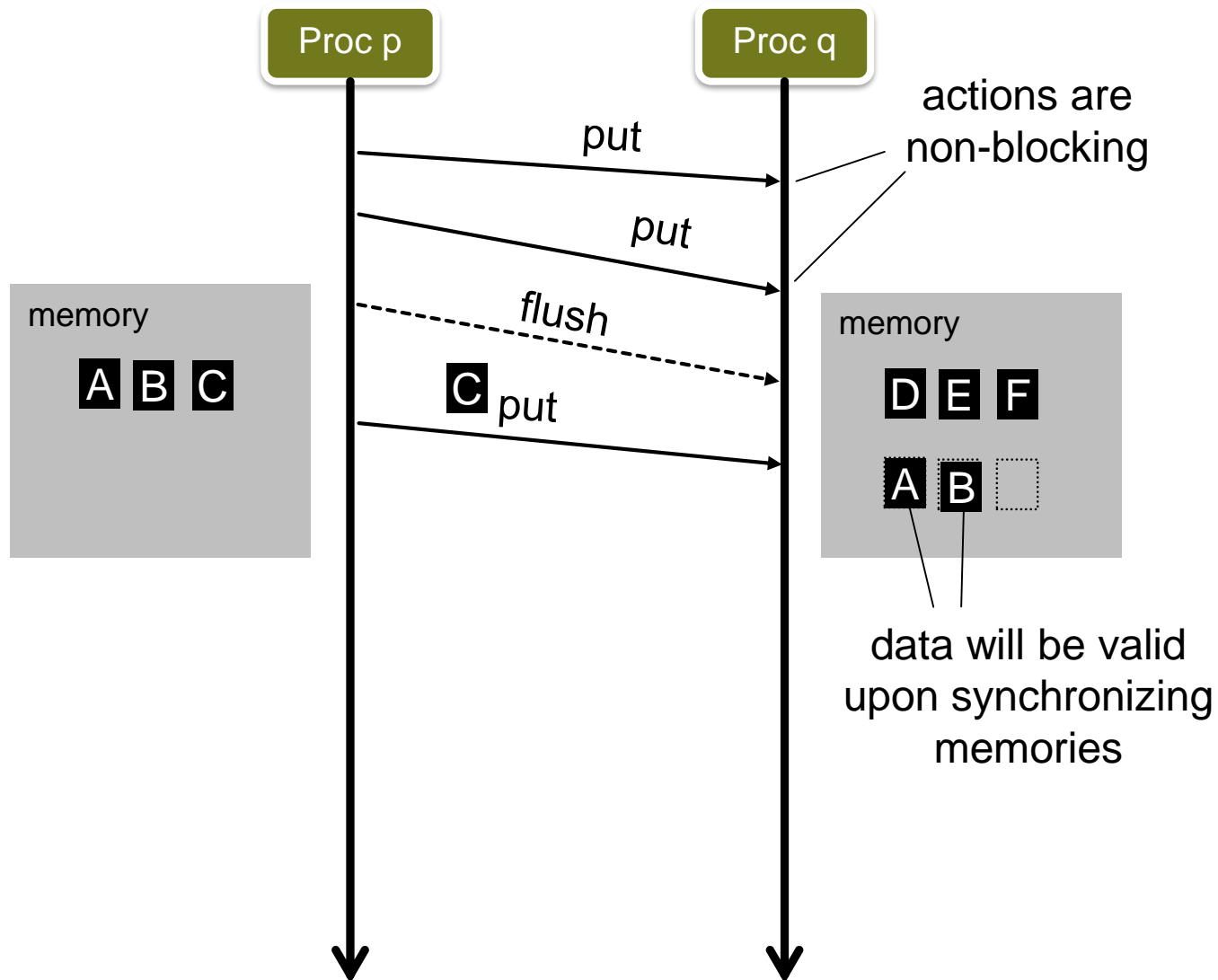
RMA: EPOCHS



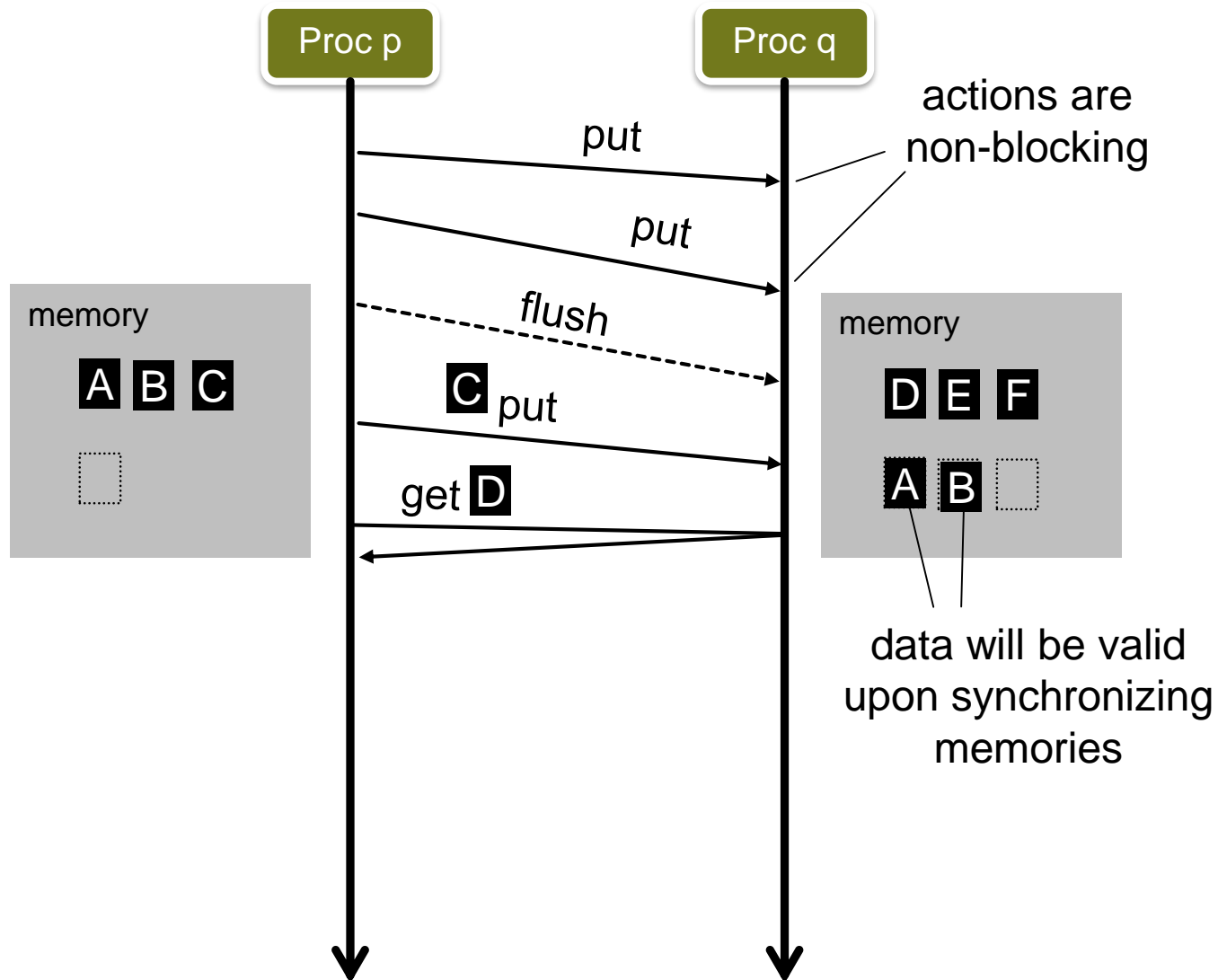
RMA: EPOCHS



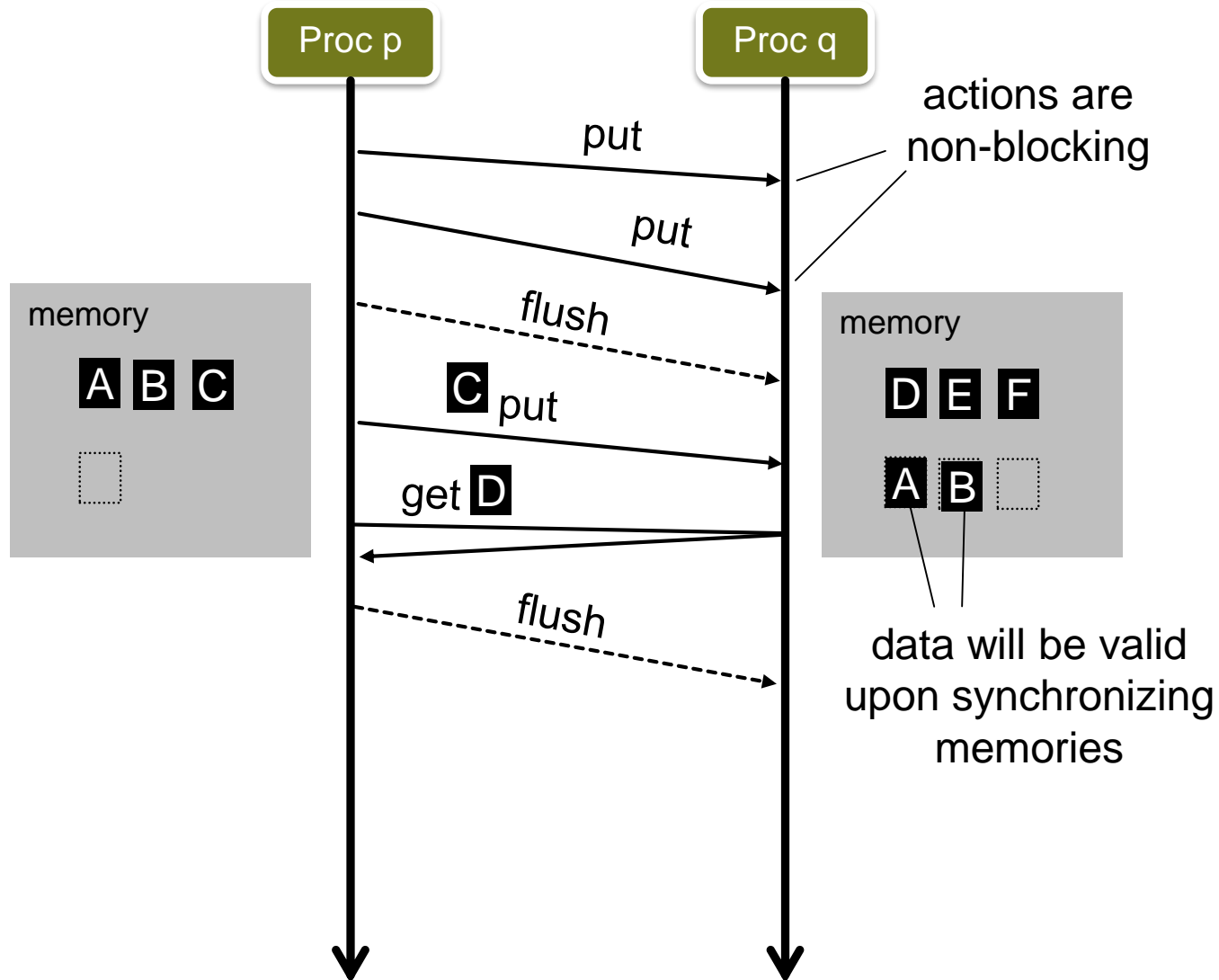
RMA: EPOCHS



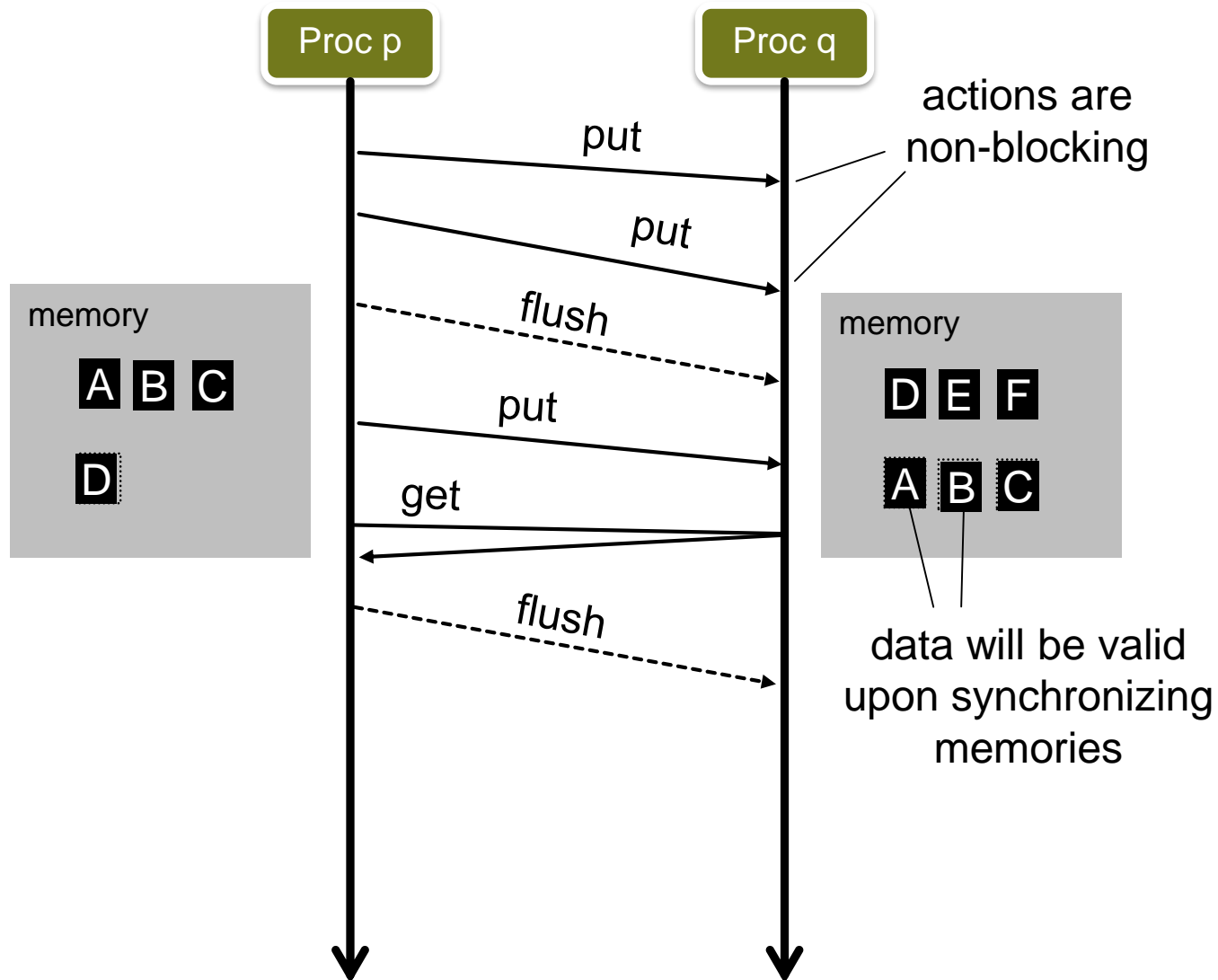
RMA: EPOCHS



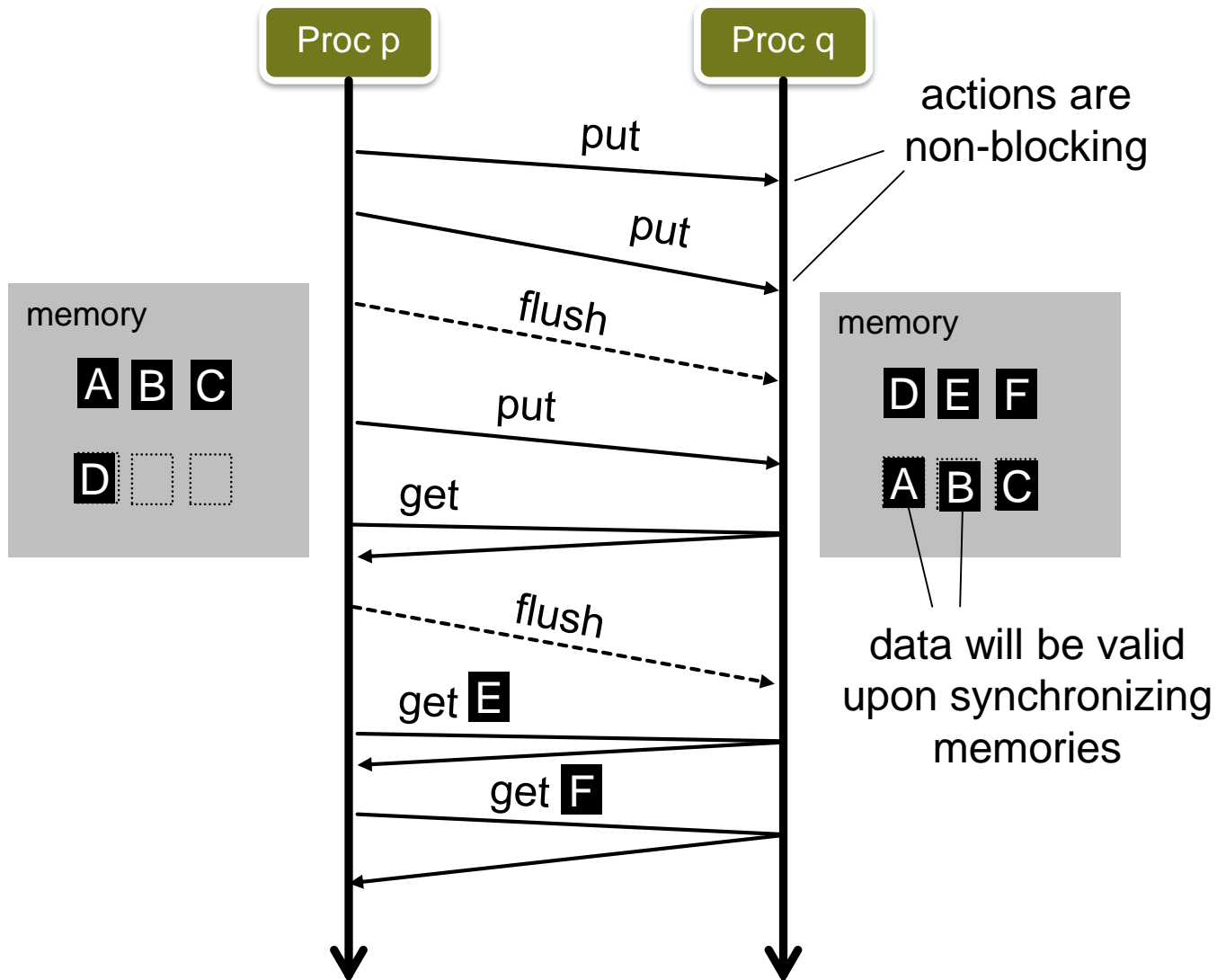
RMA: EPOCHS



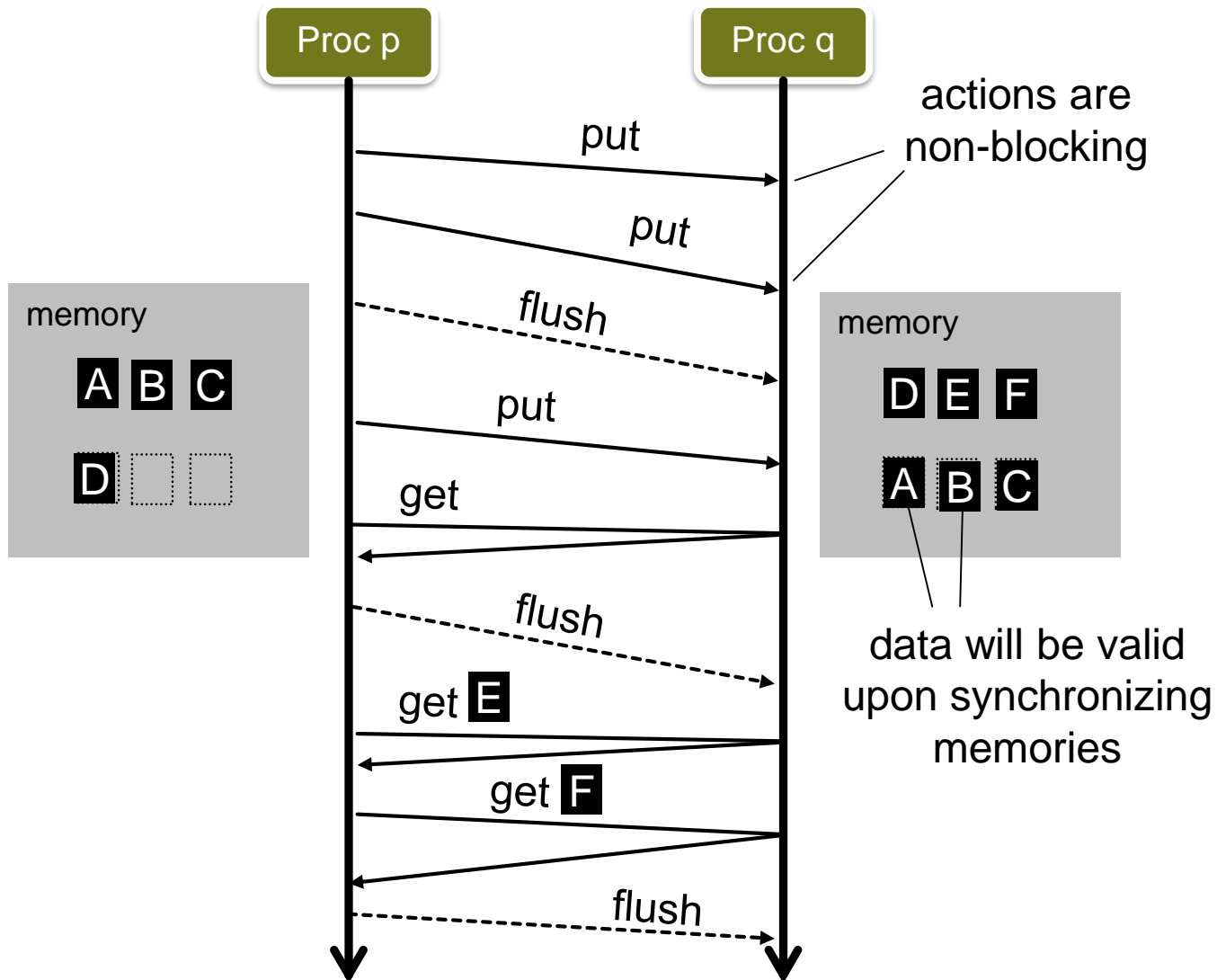
RMA: EPOCHS



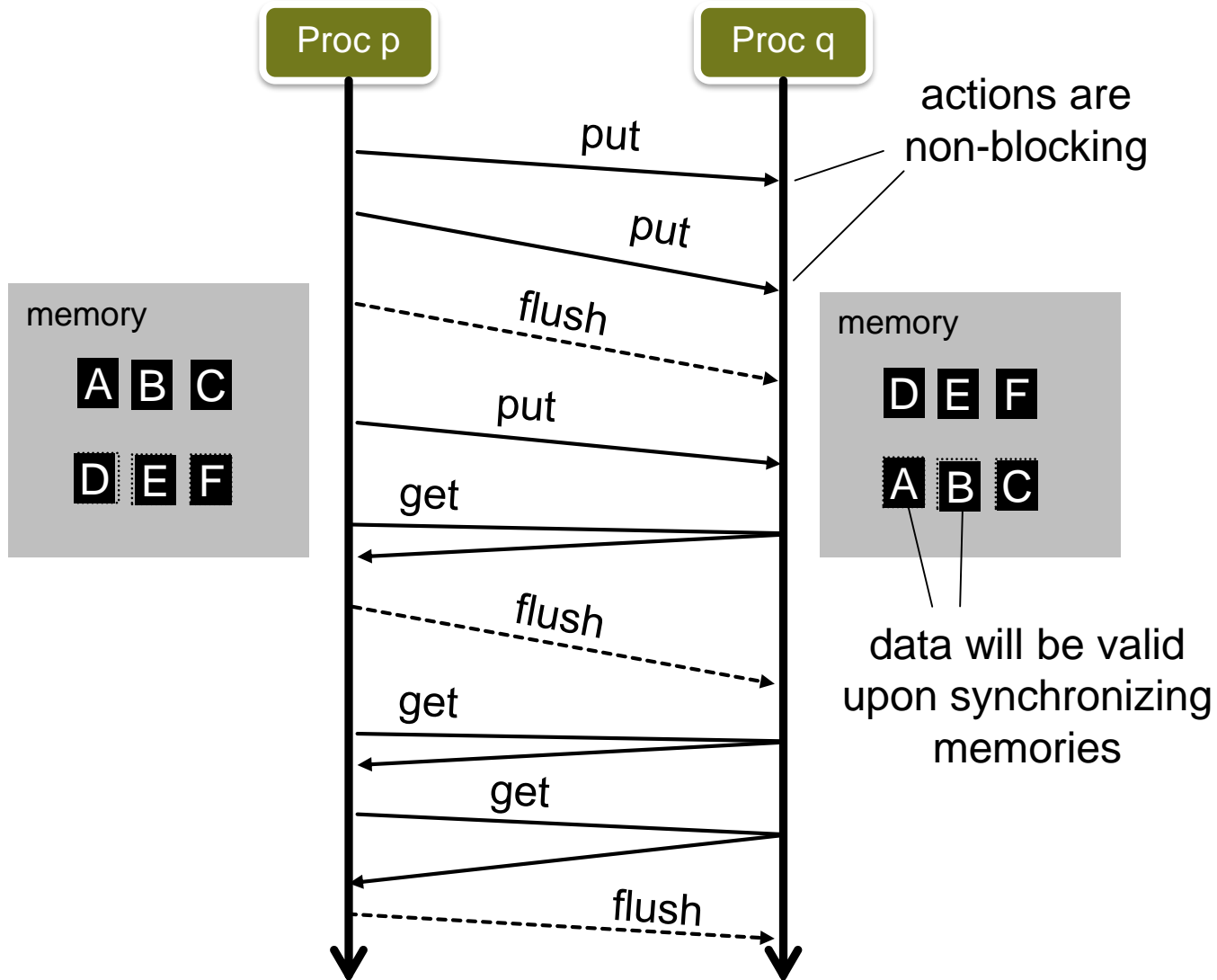
RMA: EPOCHS



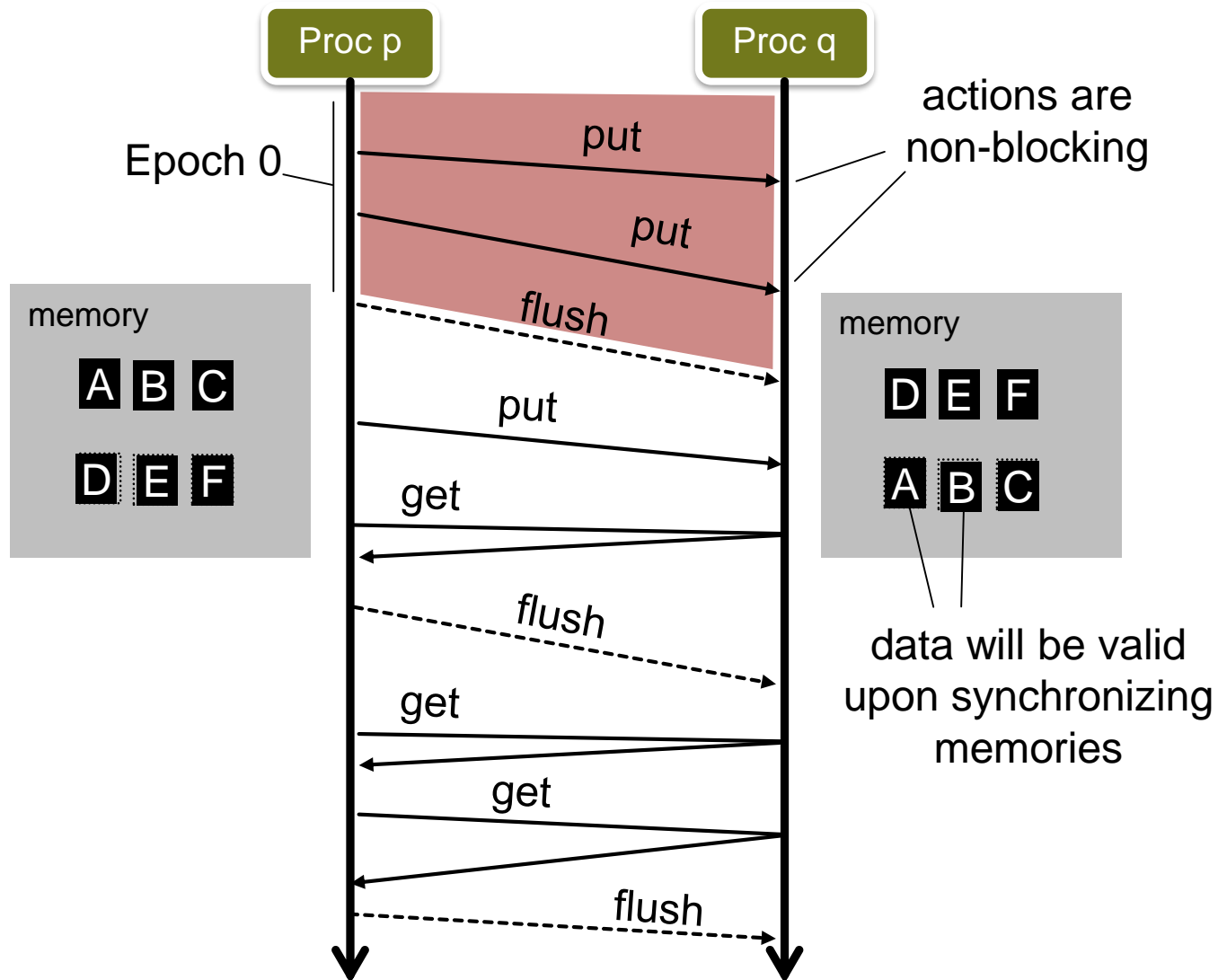
RMA: EPOCHS



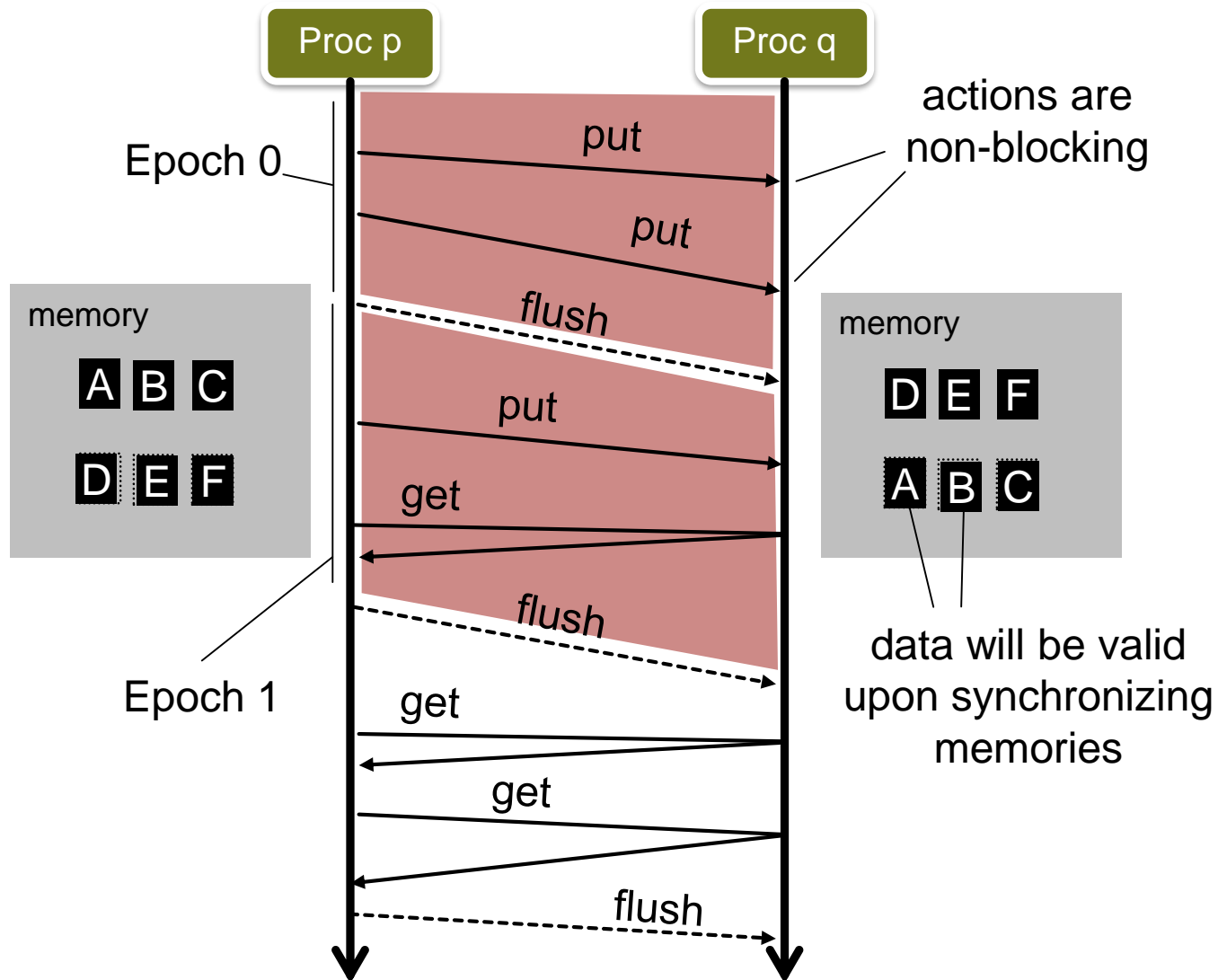
RMA: EPOCHS



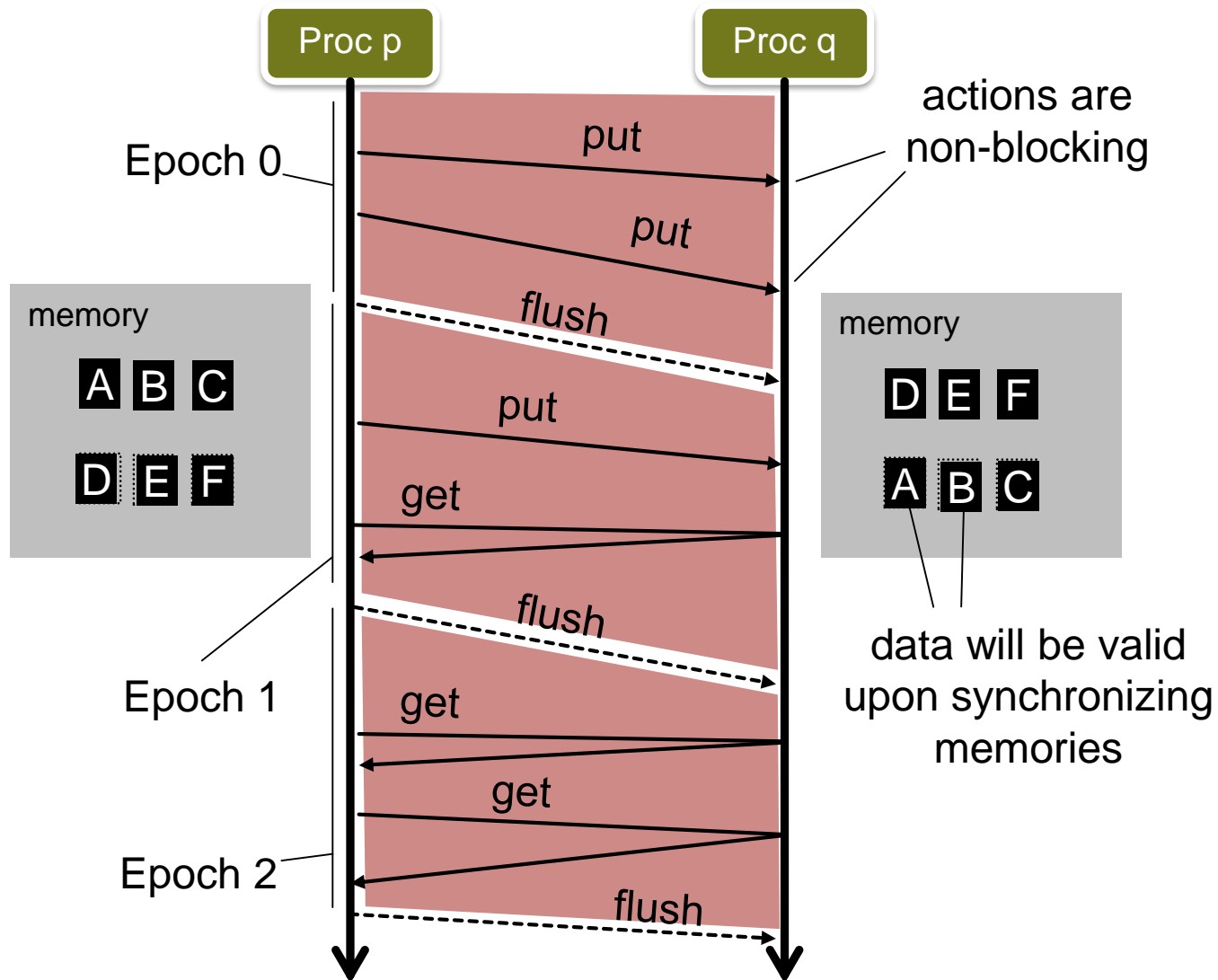
RMA: EPOCHS



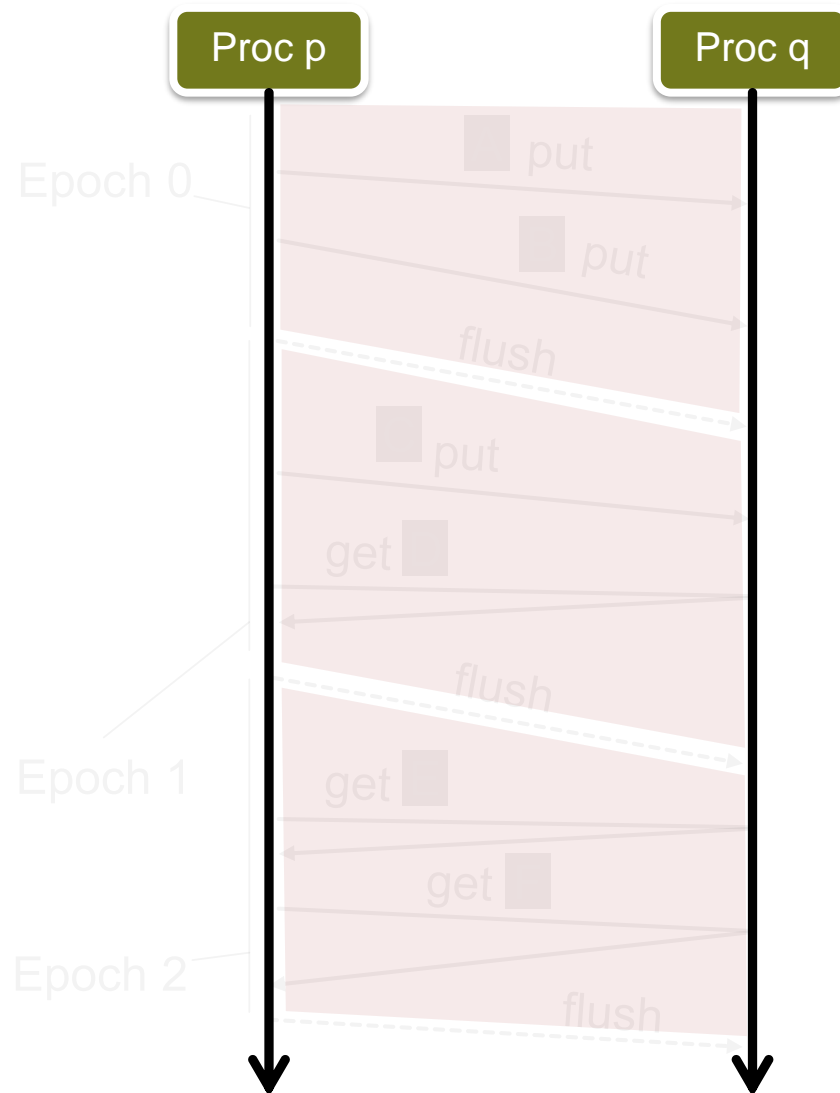
RMA: EPOCHS



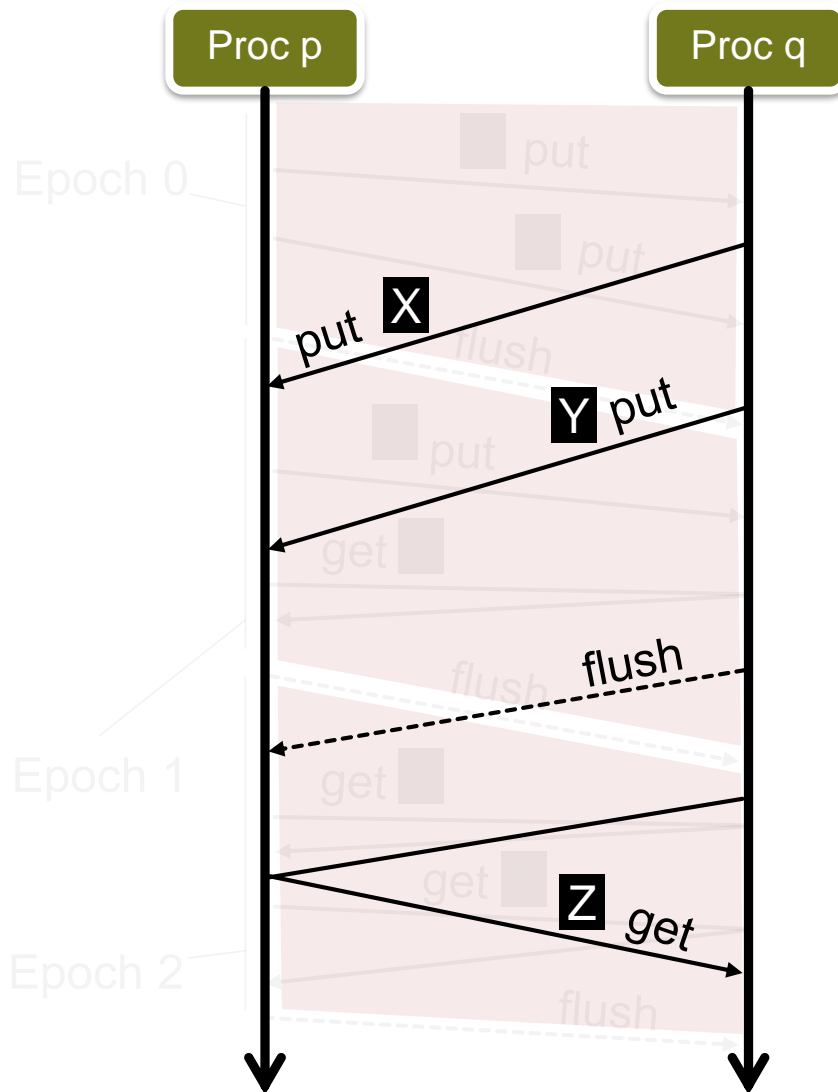
RMA: EPOCHS



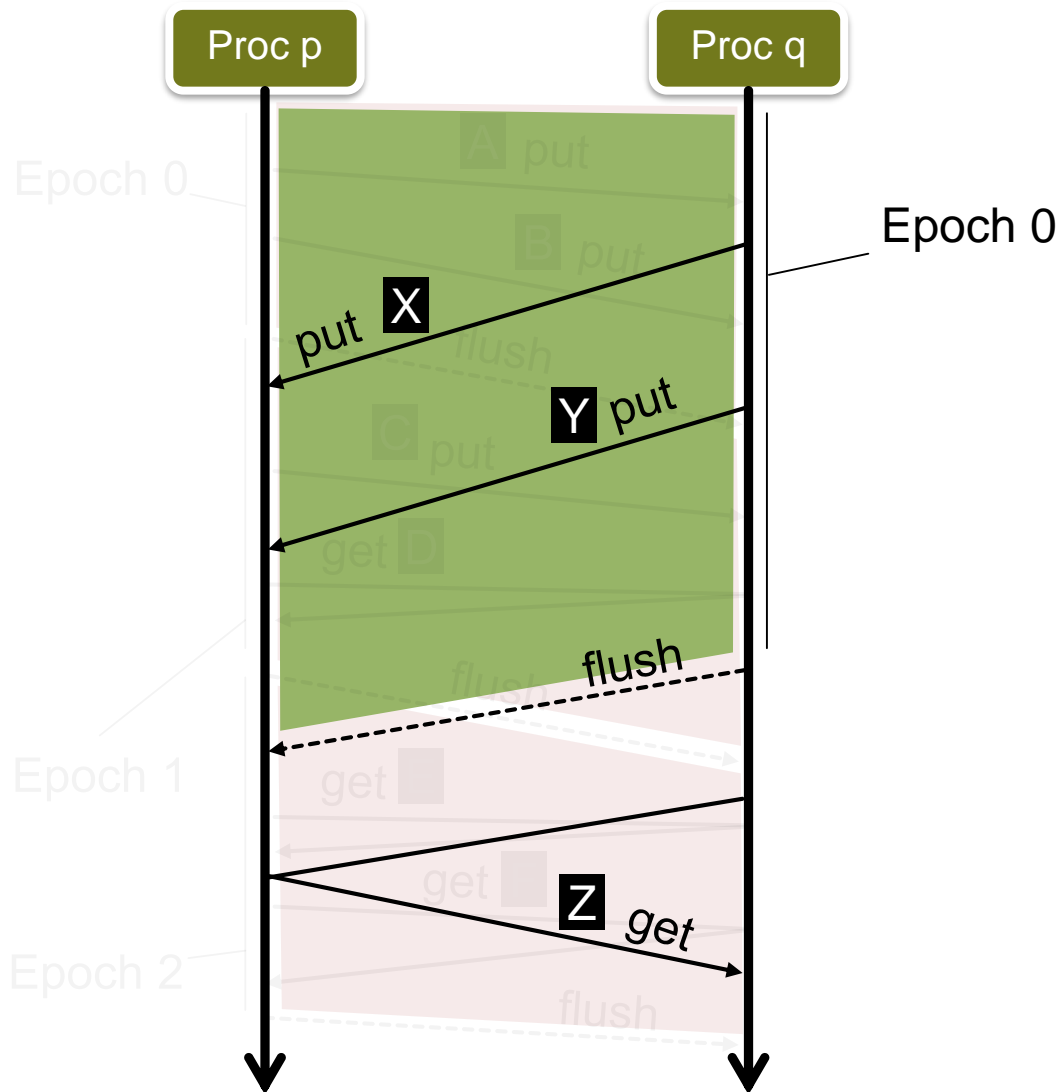
RMA: EPOCHS



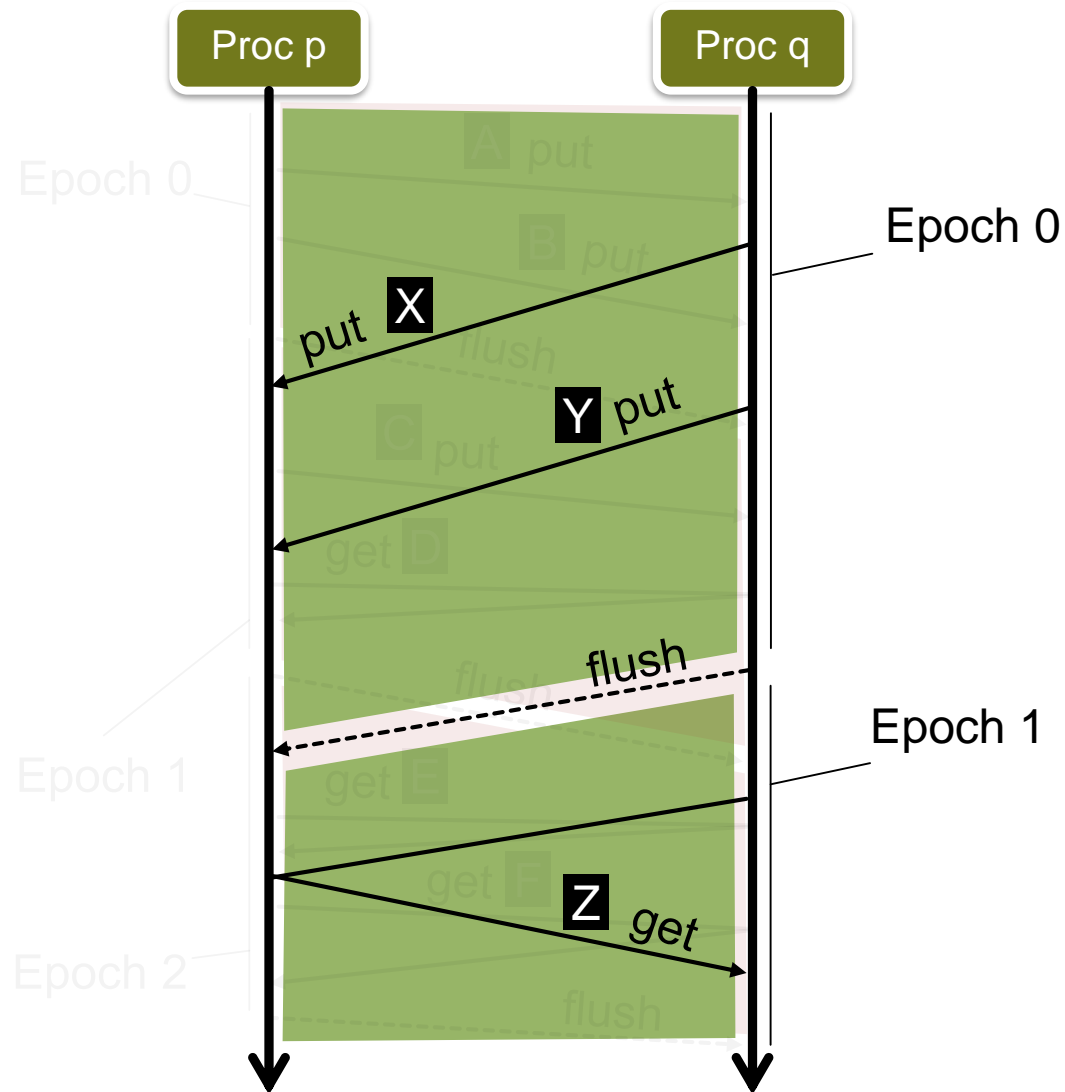
RMA: EPOCHS



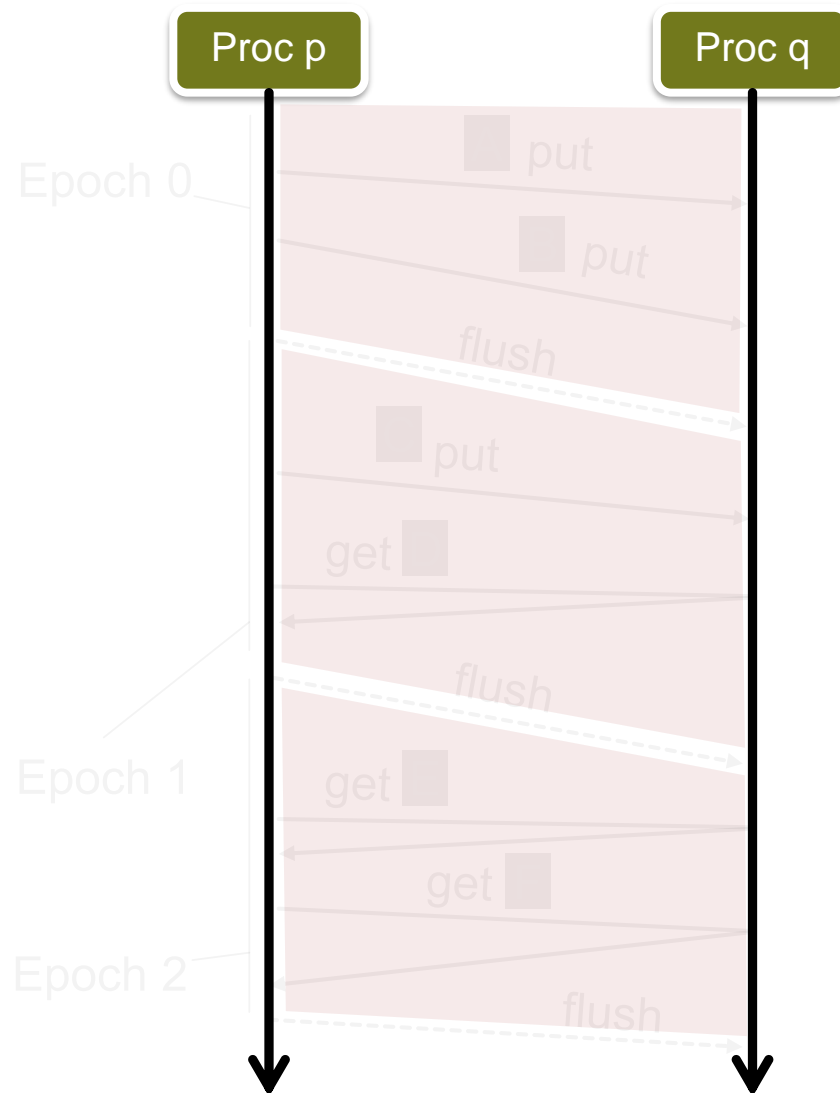
RMA: EPOCHS



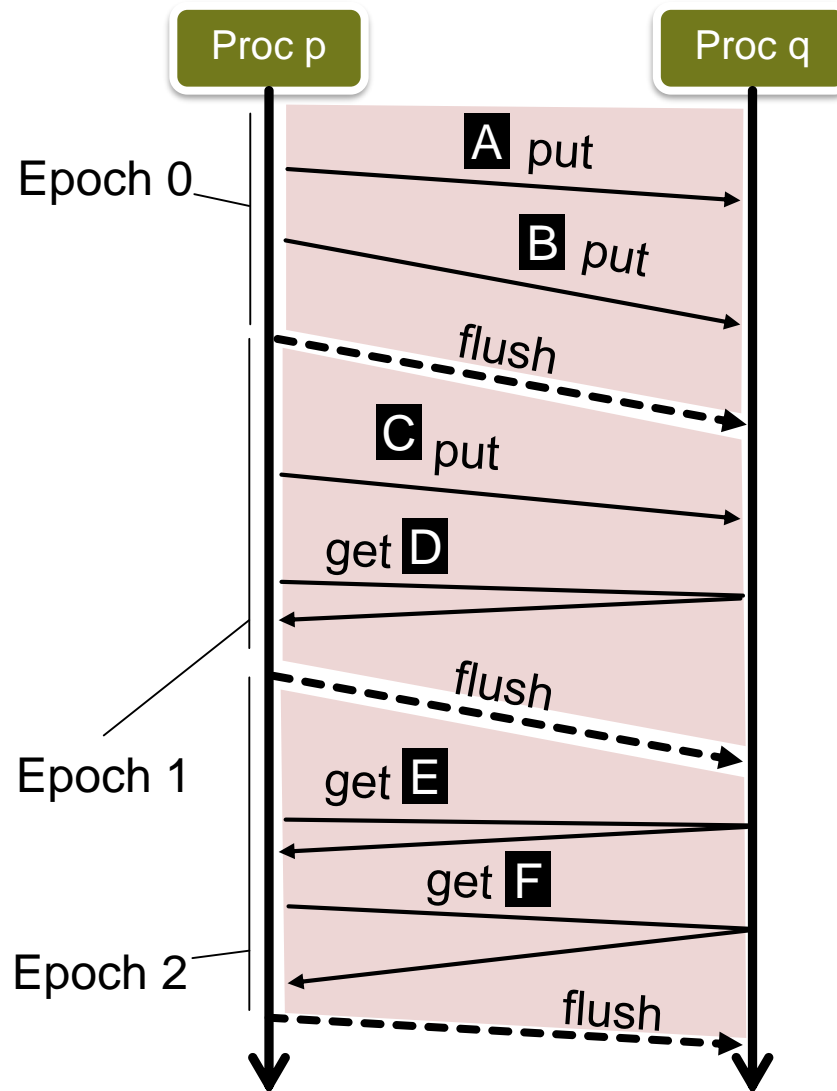
RMA: EPOCHS



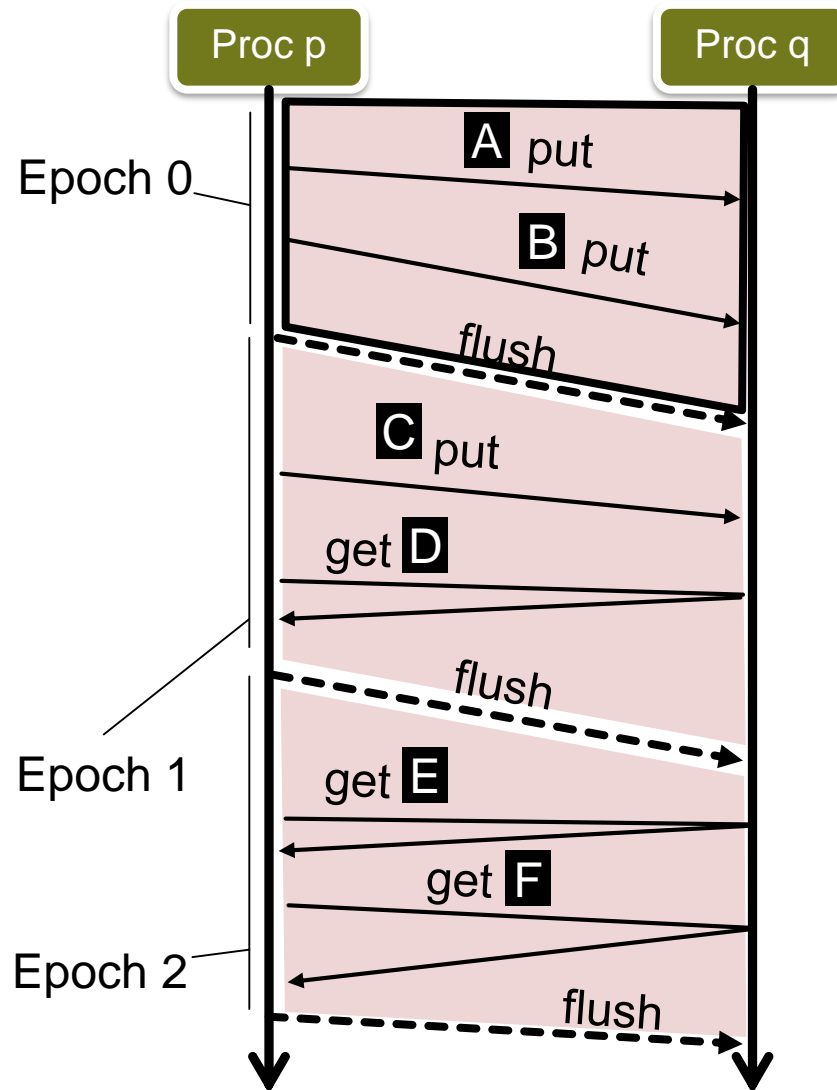
RMA: EPOCHS



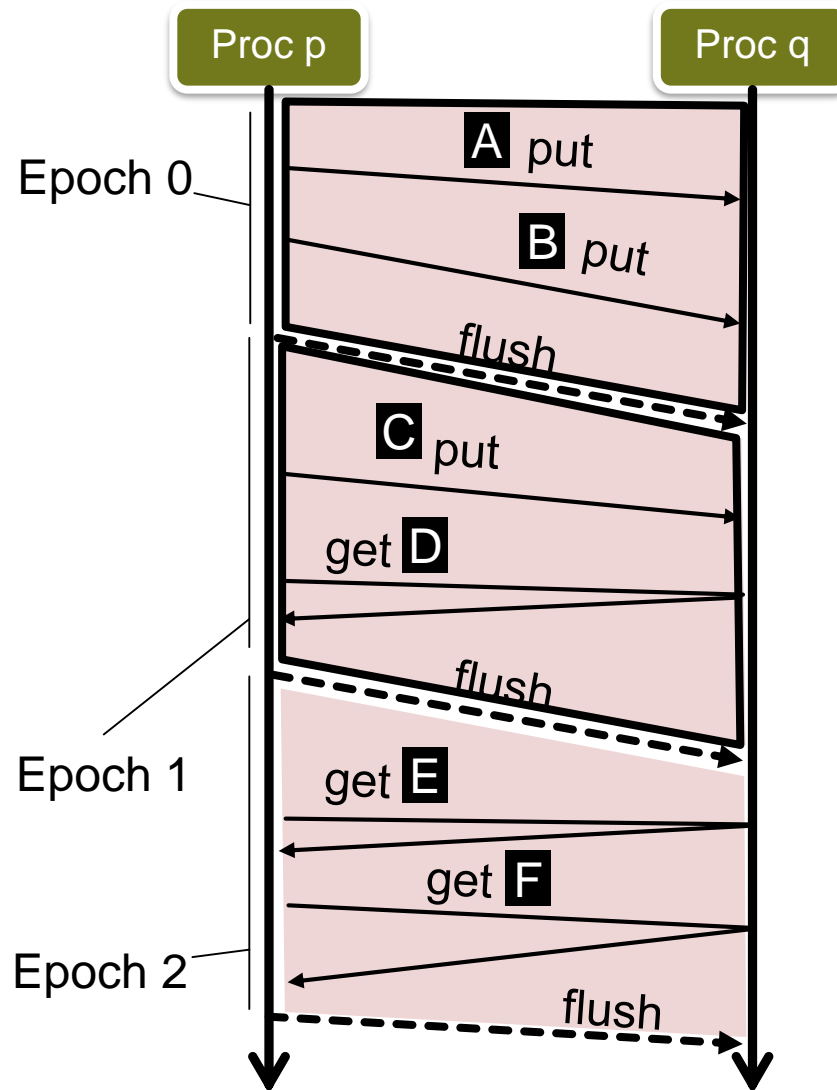
RMA: THE CONSISTENCY ORDER \xrightarrow{CO}



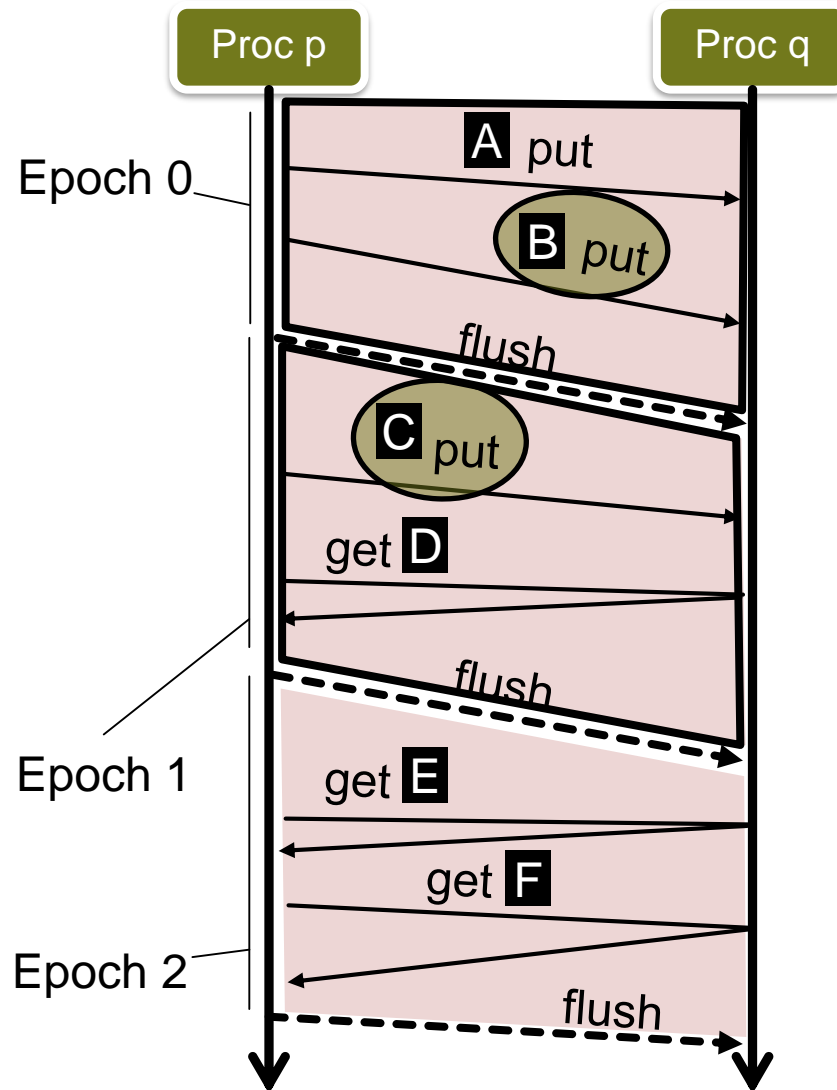
RMA: THE CONSISTENCY ORDER \xrightarrow{CO}



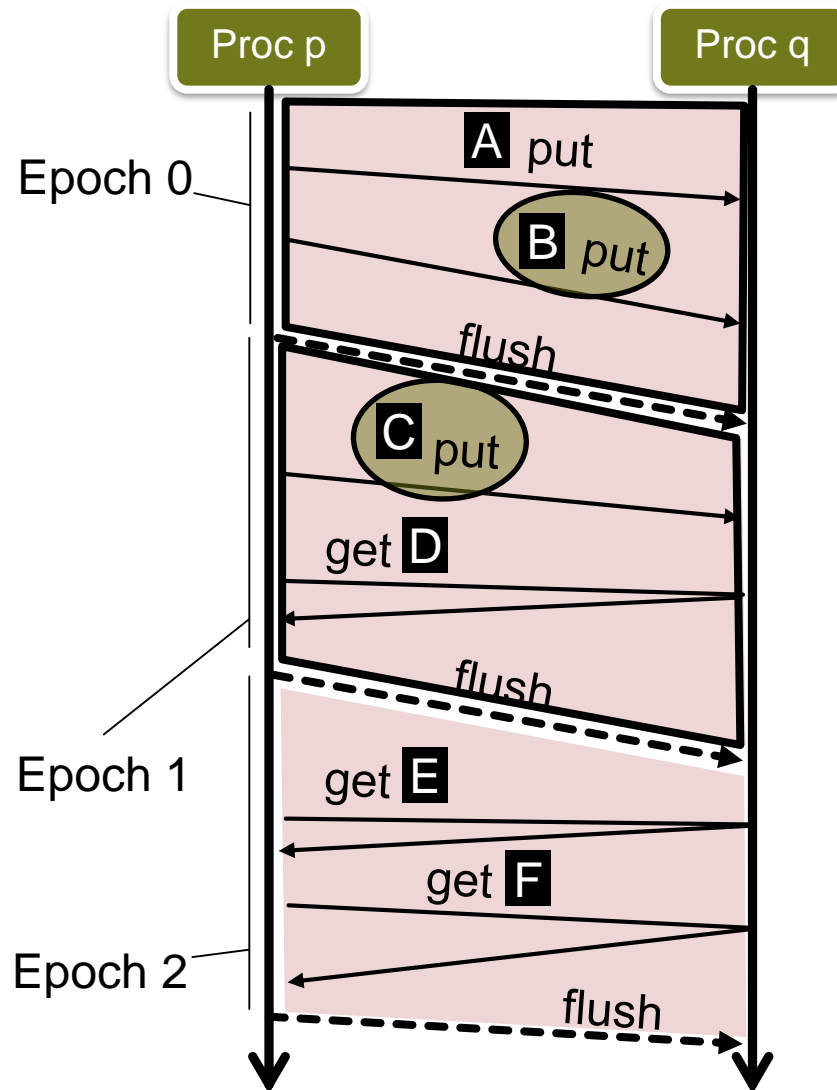
RMA: THE CONSISTENCY ORDER \xrightarrow{CO}



RMA: THE CONSISTENCY ORDER \xrightarrow{CO}

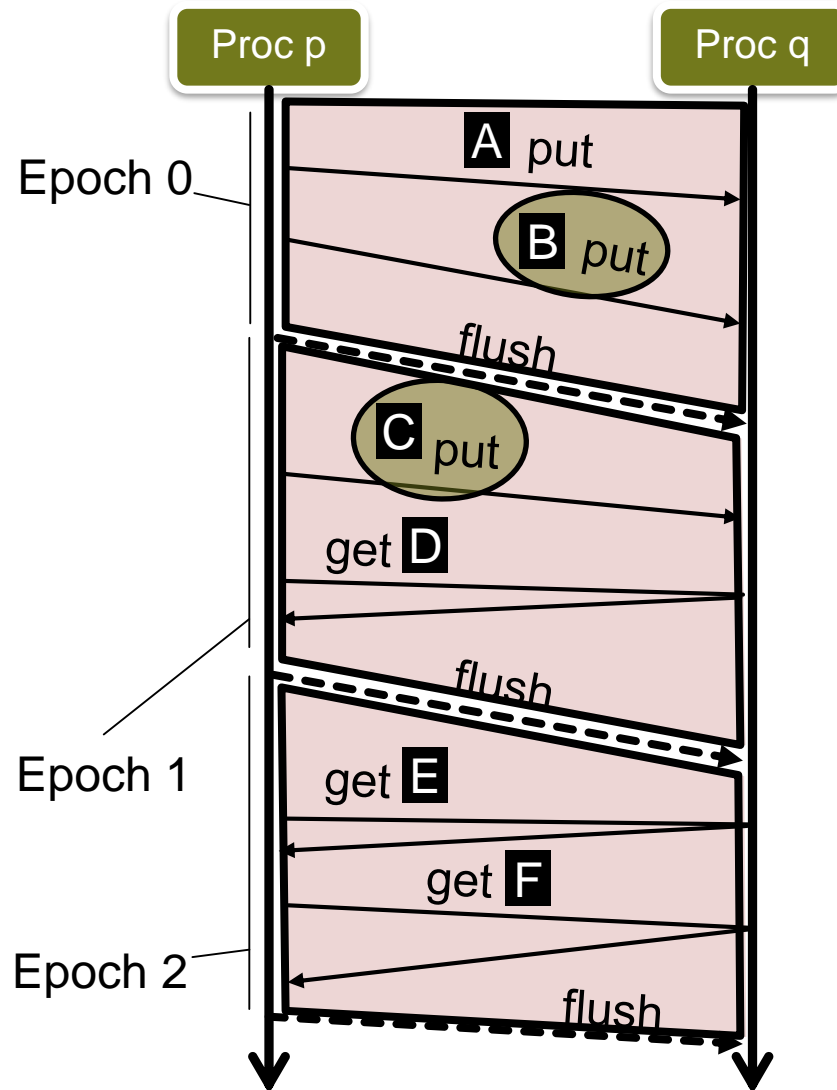


RMA: THE CONSISTENCY ORDER \xrightarrow{co}



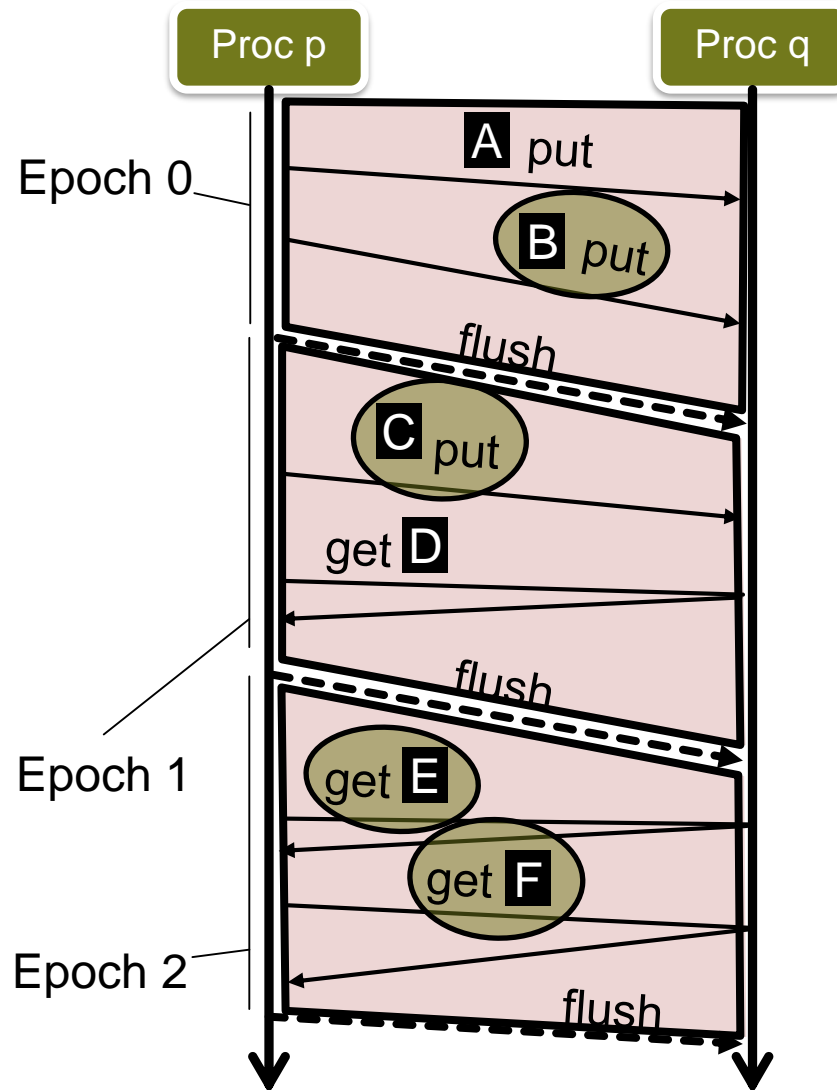
B put \xrightarrow{co} **C put**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}



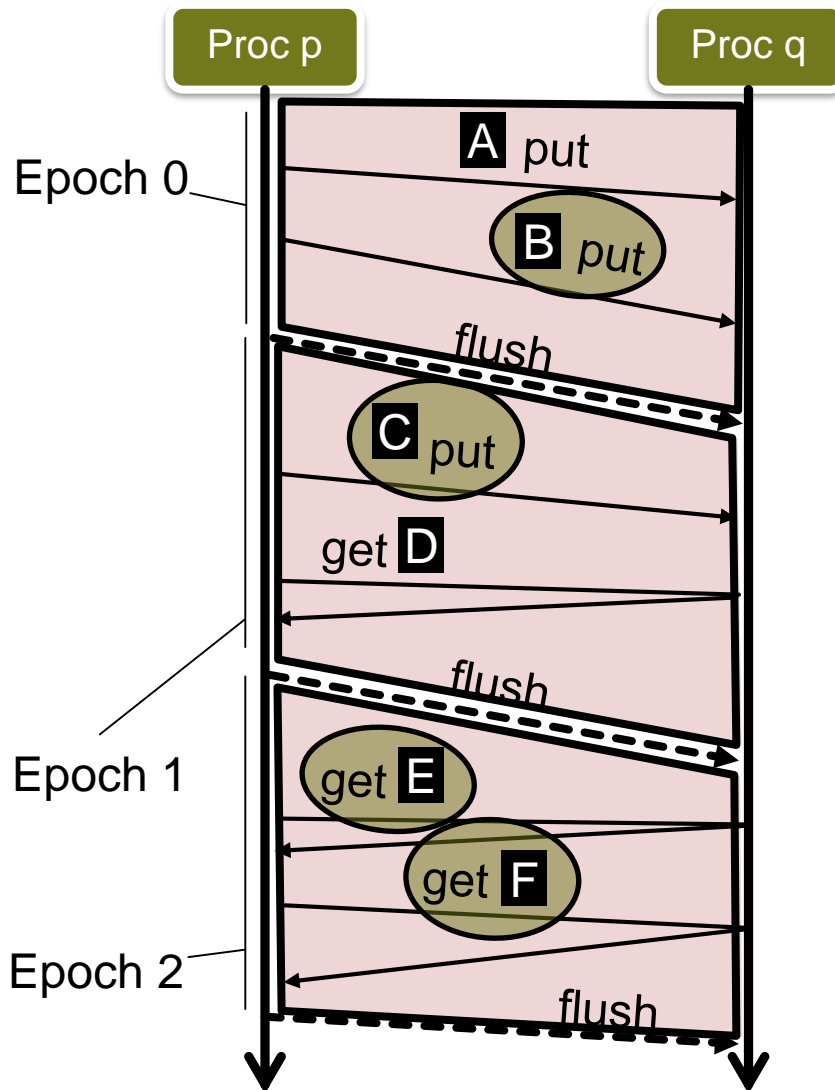
B put \xrightarrow{co} **C put**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}



B put \xrightarrow{co} **C put**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}

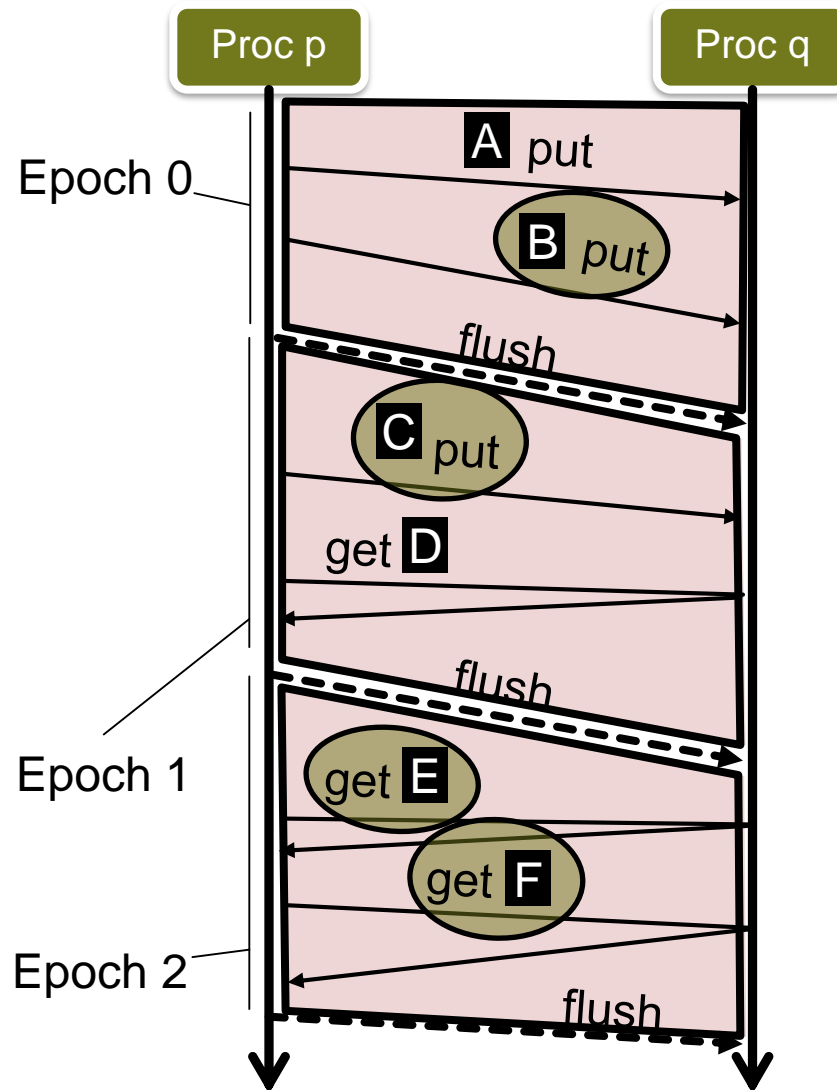


B put \xrightarrow{co} **C put**

get **E** \parallel_{co} get **F**

RMA: THE CONSISTENCY ORDER $\xrightarrow{\text{co}}$

For recovery, a process has to replay actions in the correct $\xrightarrow{\text{co}}$ order!



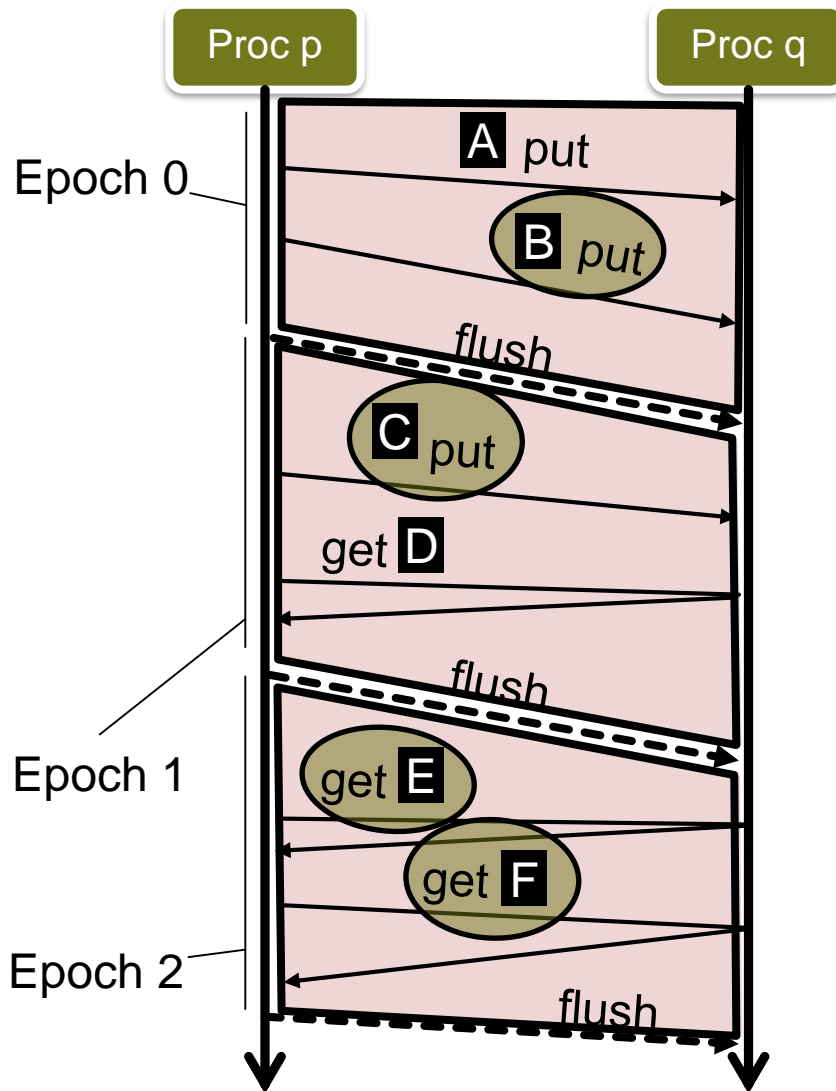
B put $\xrightarrow{\text{co}}$ **C put**

get E \parallel_{co} **get F**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}

For recovery, a process has to replay actions in the correct \xrightarrow{co} order!

For this, we use **epoch counters (EC)**



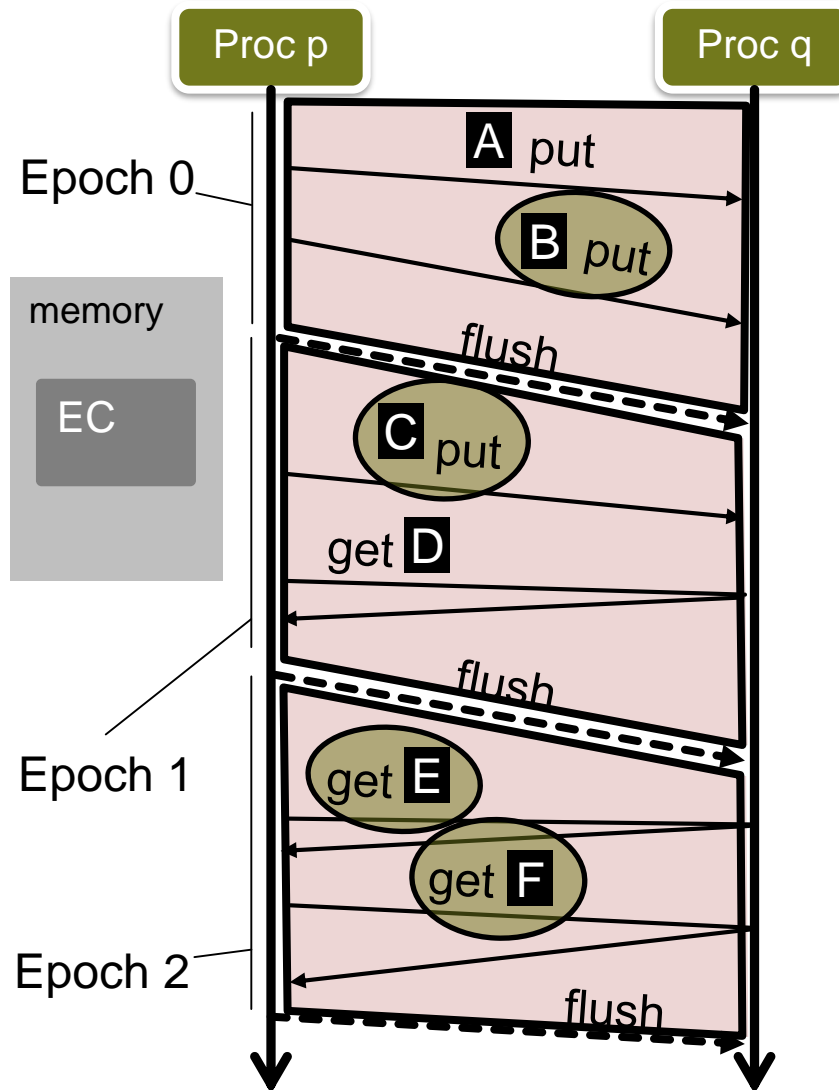
B put \xrightarrow{co} **C** put

get **E** \parallel_{co} get **F**

RMA: THE CONSISTENCY ORDER $\xrightarrow{\text{co}}$

For recovery, a process has to replay actions in the correct $\xrightarrow{\text{co}}$ order!

For this, we use **epoch counters (EC)**



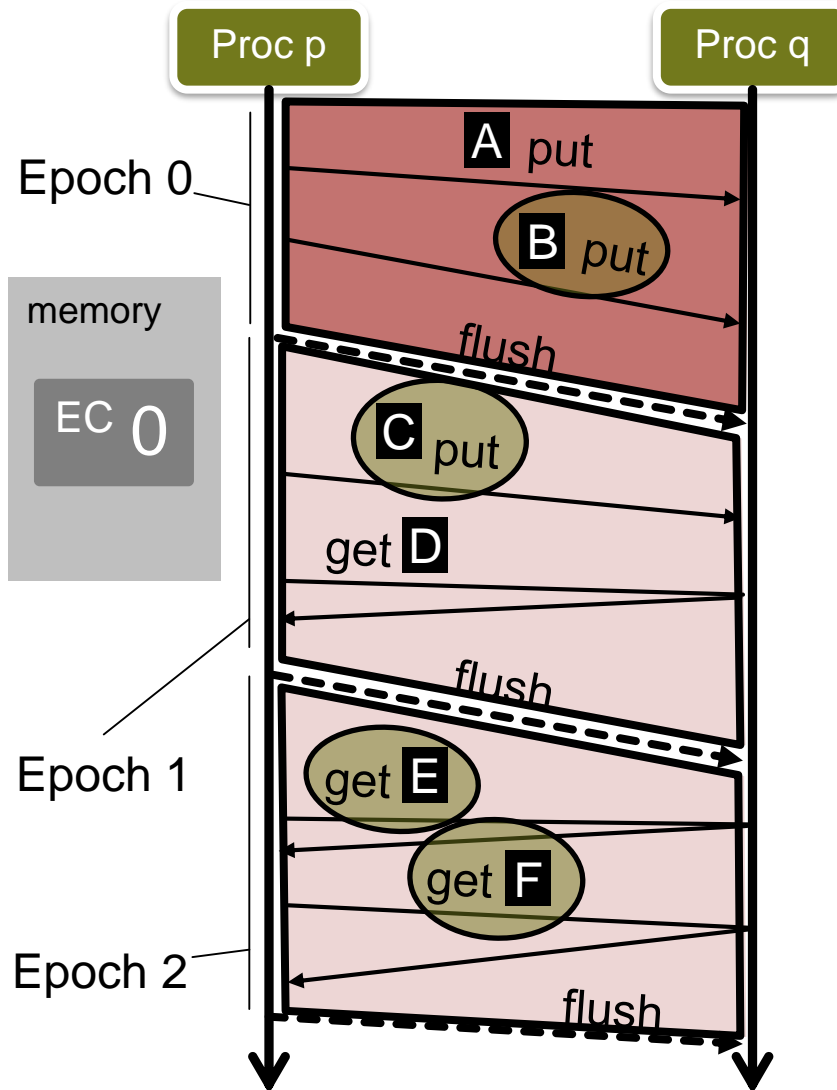
B put $\xrightarrow{\text{co}}$ **C put**

get E \parallel_{co} **get F**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}

For recovery, a process has to replay actions in the correct \xrightarrow{co} order!

For this, we use **epoch counters (EC)**



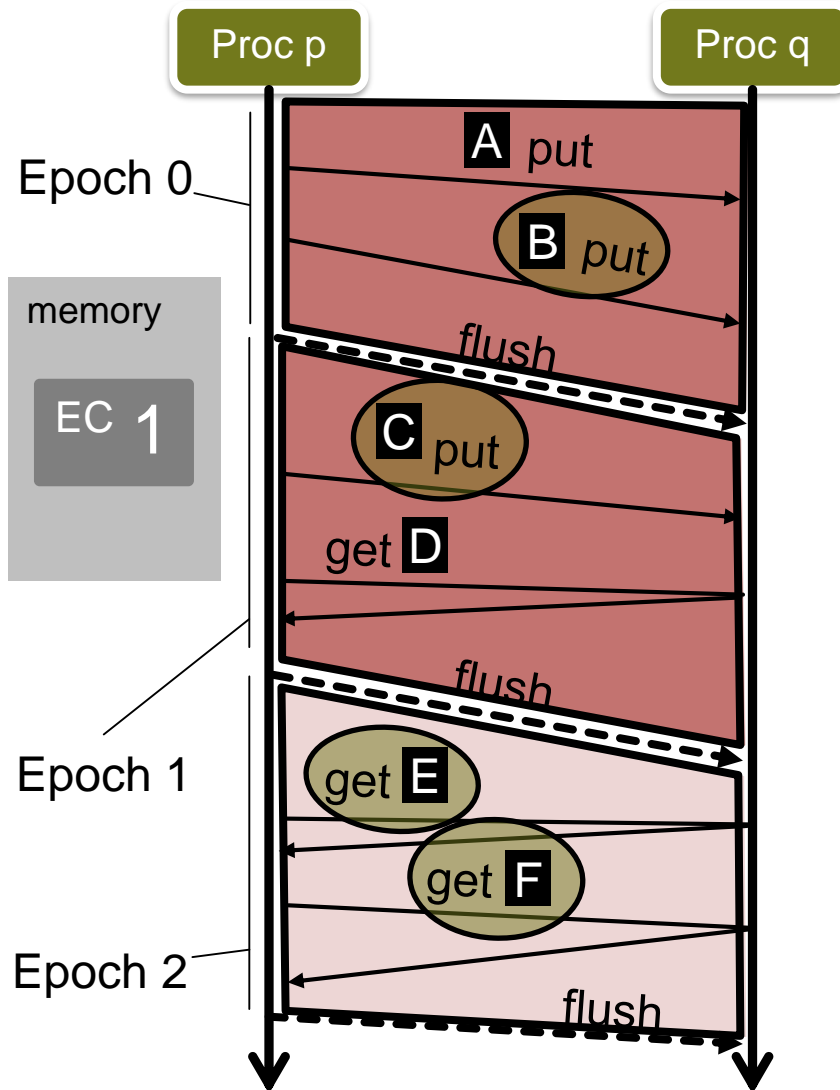
B put \xrightarrow{co} **C put**

get **E** \parallel_{co} get **F**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}

For recovery, a process has to replay actions in the correct \xrightarrow{co} order!

For this, we use **epoch counters (EC)**



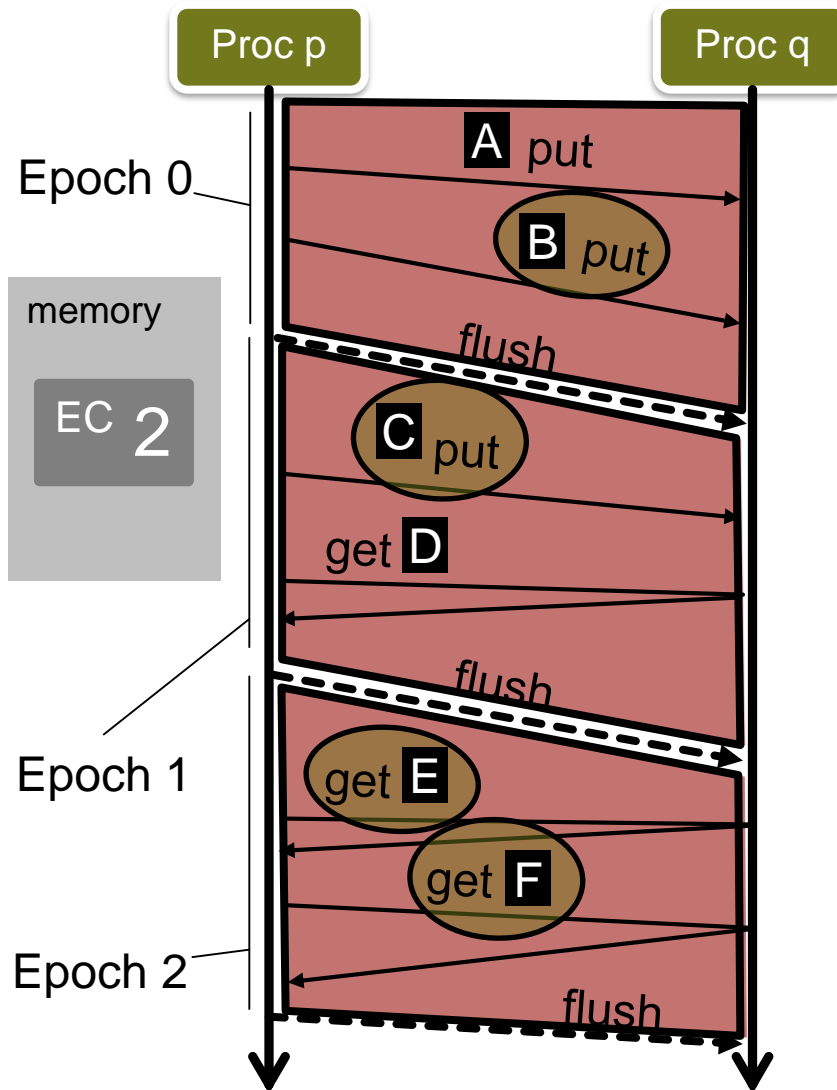
B put \xrightarrow{co} **C put**

get **E** \parallel_{co} get **F**

RMA: THE CONSISTENCY ORDER \xrightarrow{co}

For recovery, a process has to replay actions in the correct \xrightarrow{co} order!

For this, we use **epoch counters (EC)**



B put \xrightarrow{co} **C put**

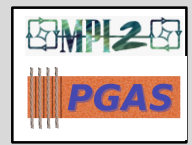
get E \parallel_{co} **get F**

OVERVIEW OF OUR RESEARCH

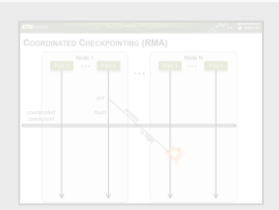
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

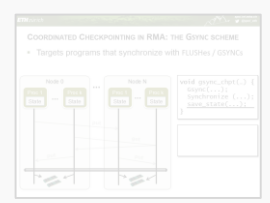
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

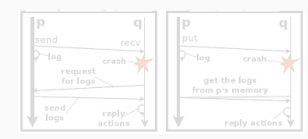


MP vs. RMA

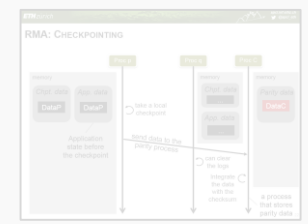


Schemes

UC in RMA

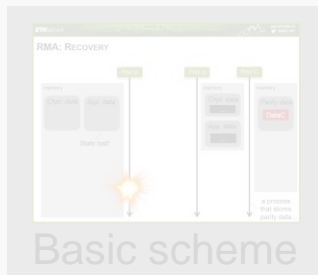


MP vs. RMA



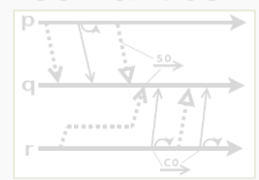
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{coh} order (referred to as the gsync order).*

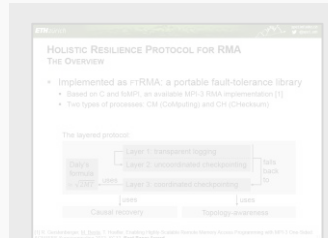
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

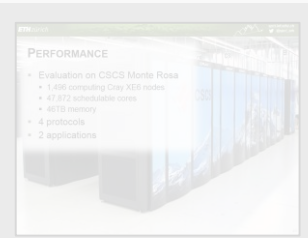
Holistic fault-tolerance library



Design

Checkpoints on demand

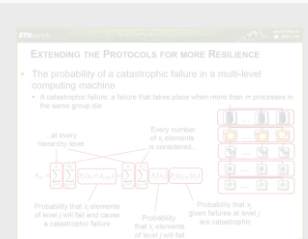
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

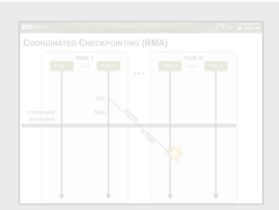
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

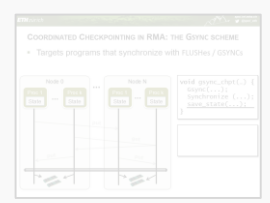
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

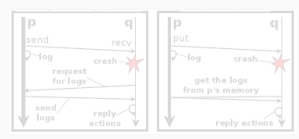


MP vs. RMA

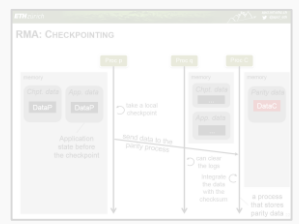


Schemes

UC in RMA

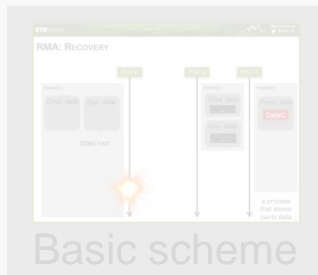


MP vs. RMA



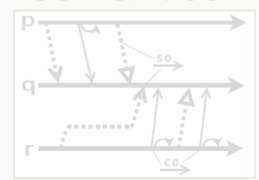
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

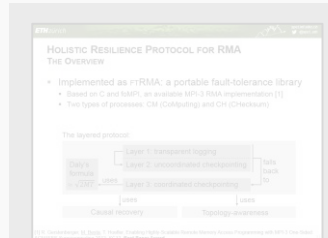
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

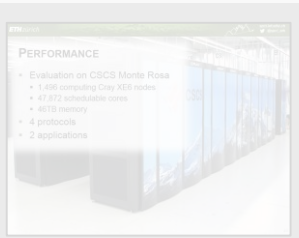
Holistic fault-tolerance library



Design

Checkpoints on demand

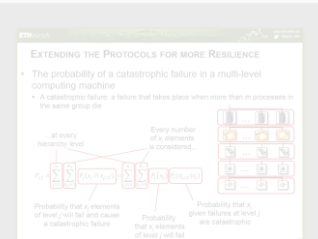
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

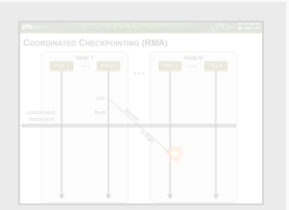
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

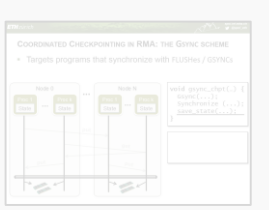
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

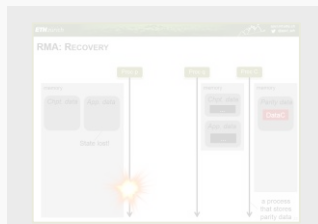


MP vs. RMA



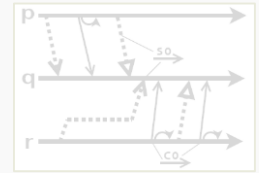
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recoverithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery

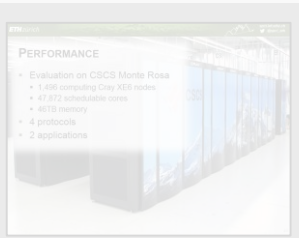
Holistic fault-tolerance library



Design

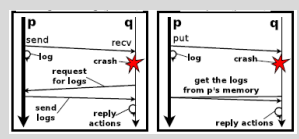
Checkpoints on demand

Optimizations

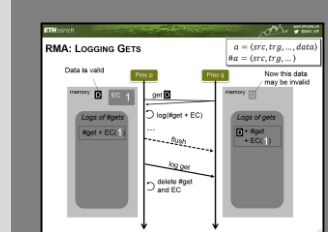


Performance

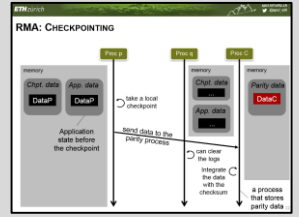
UC in RMA



MP vs. RMA



Logging accesses

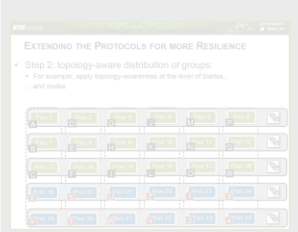


Checkpointing schemes

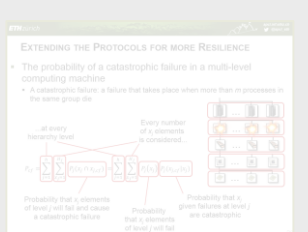
Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

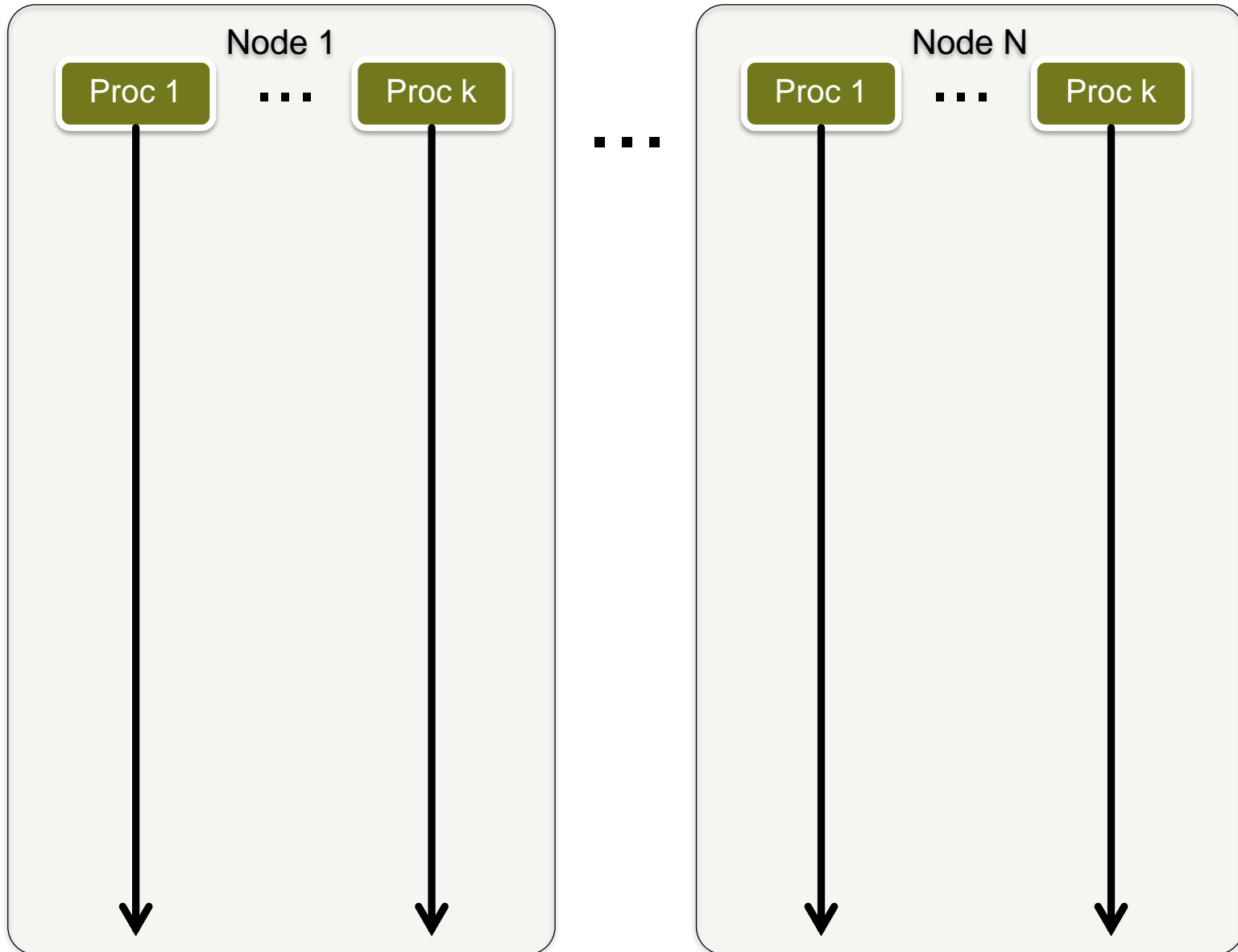


Distribution of processes

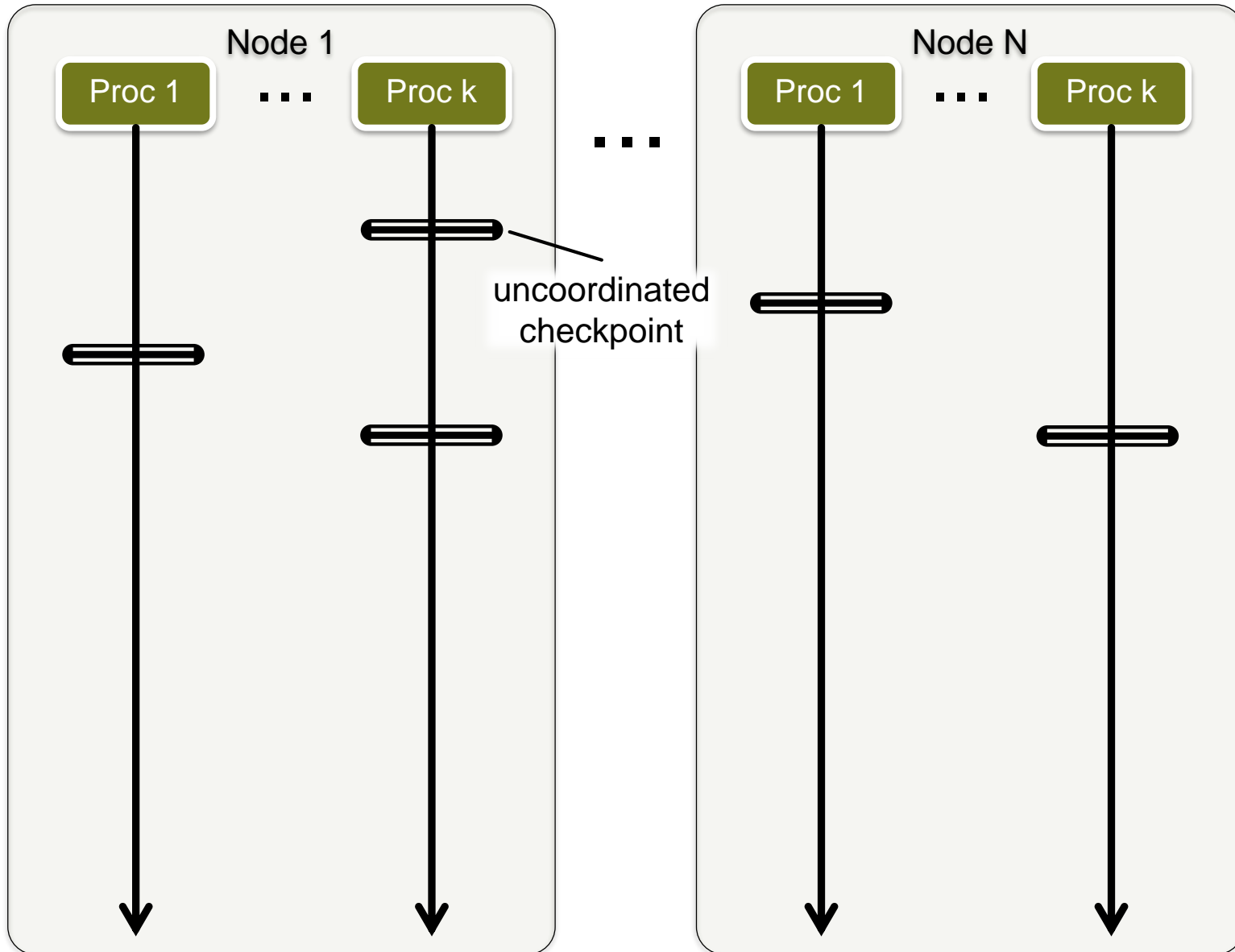


Decreasing failure prob.

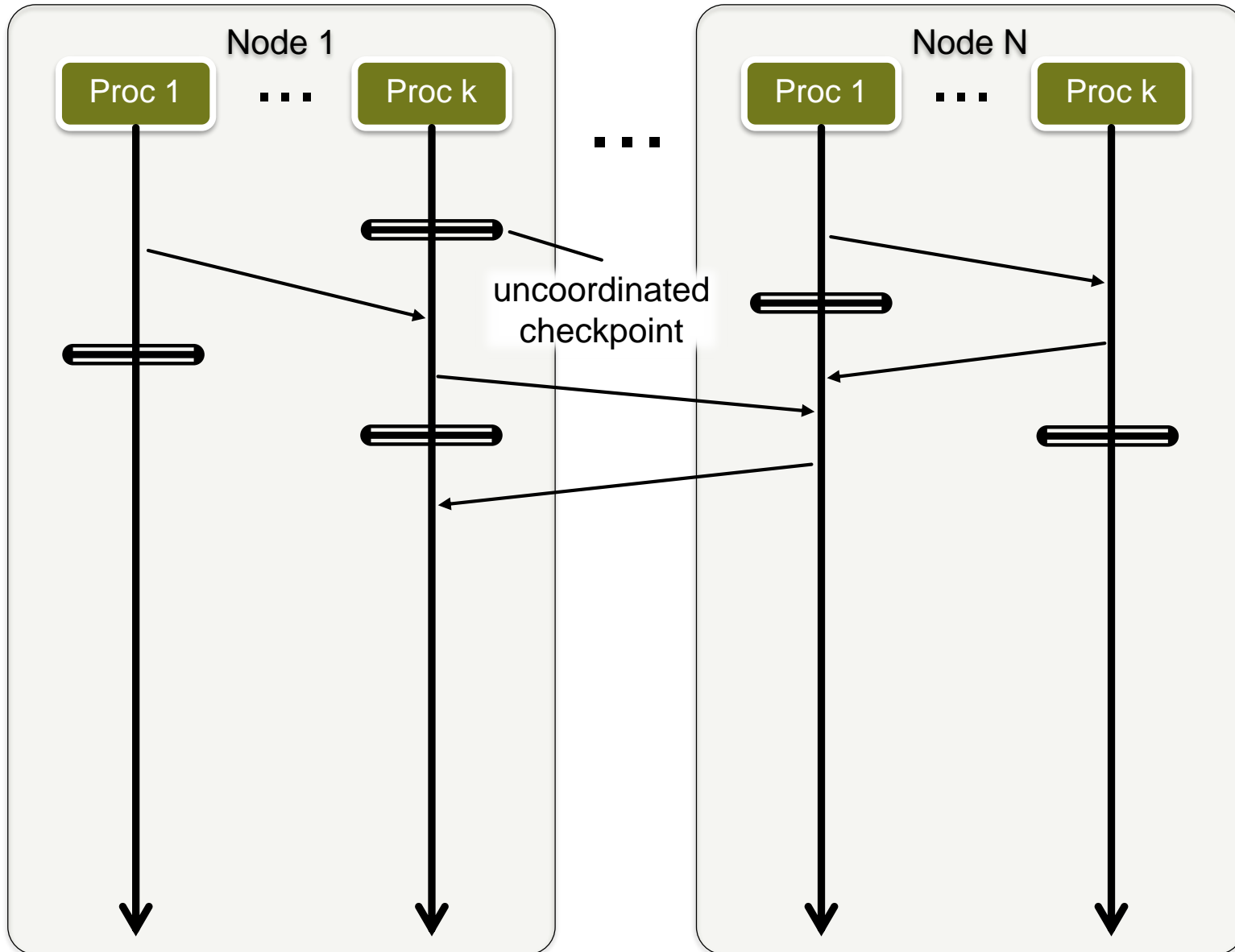
UNCOORDINATED CHECKPOINTING (MP)



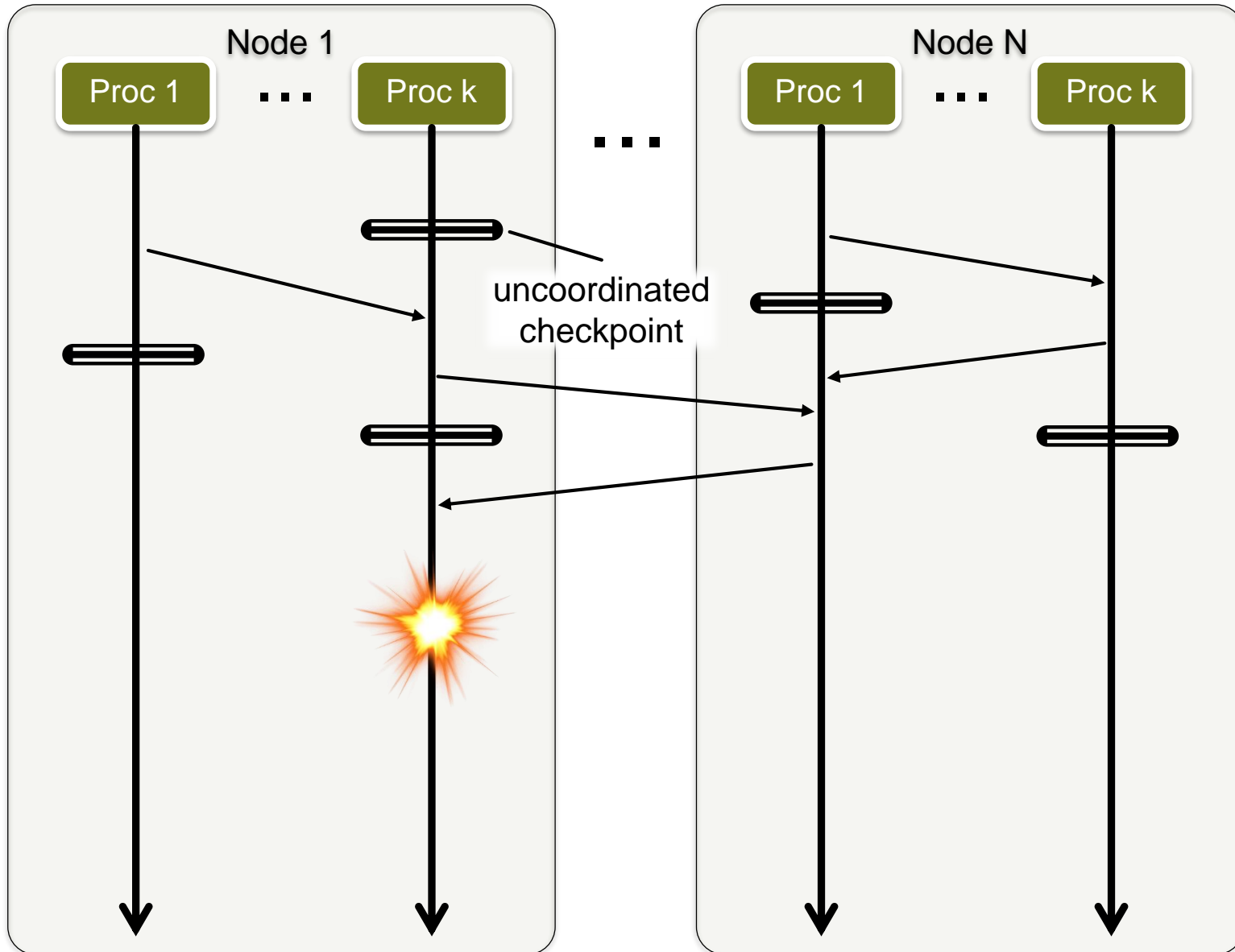
UNCOORDINATED CHECKPOINTING (MP)



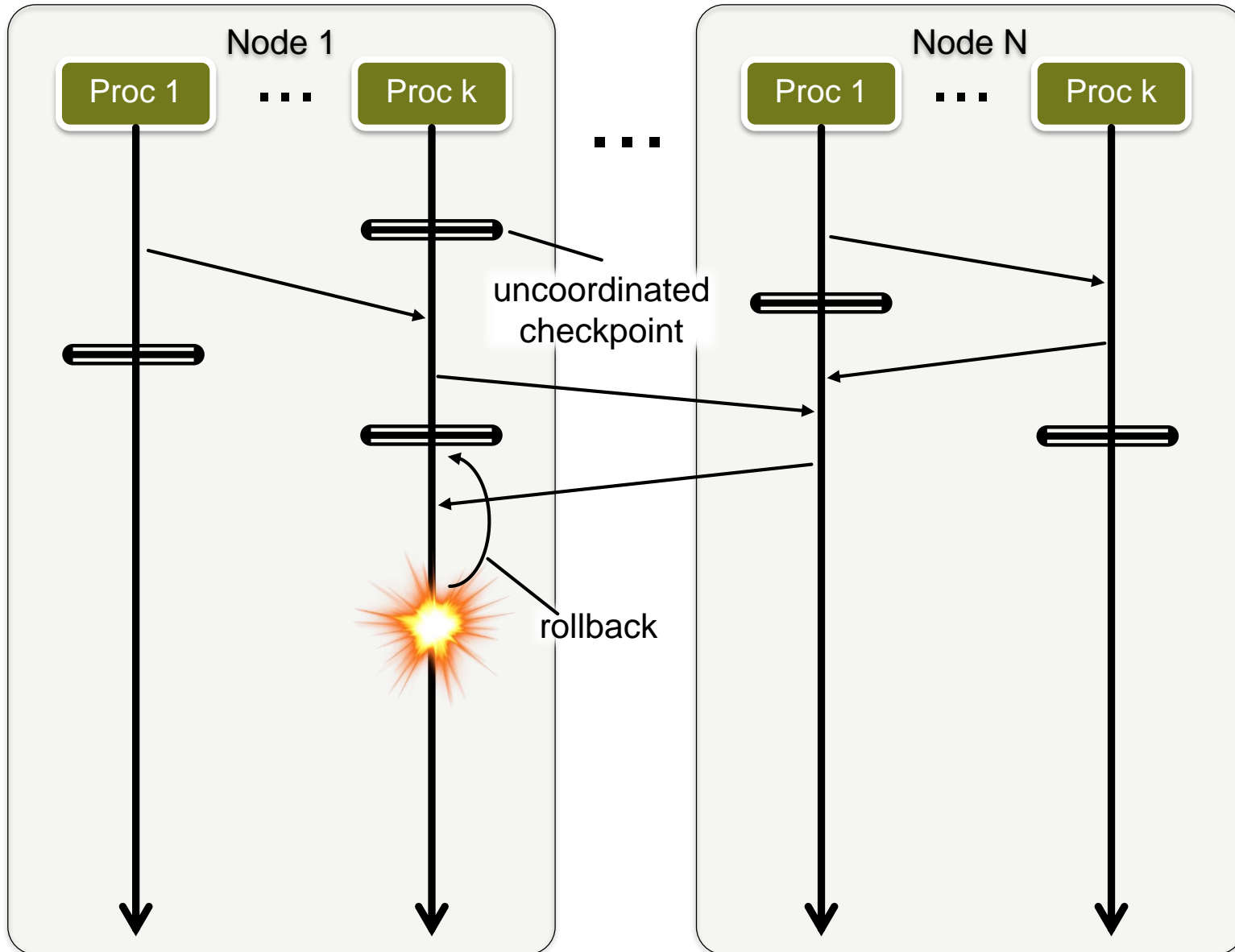
UNCOORDINATED CHECKPOINTING (MP)



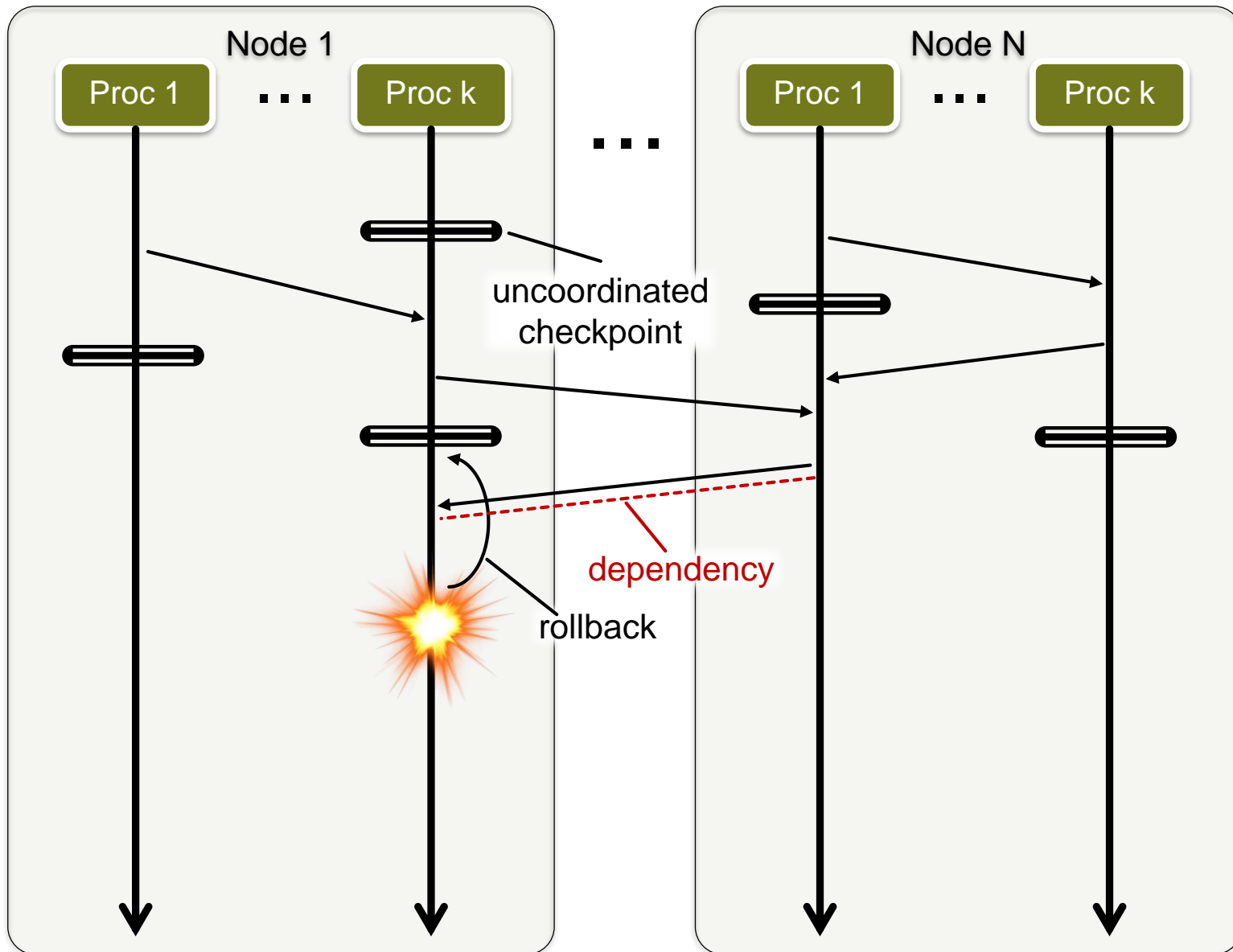
UNCOORDINATED CHECKPOINTING (MP)



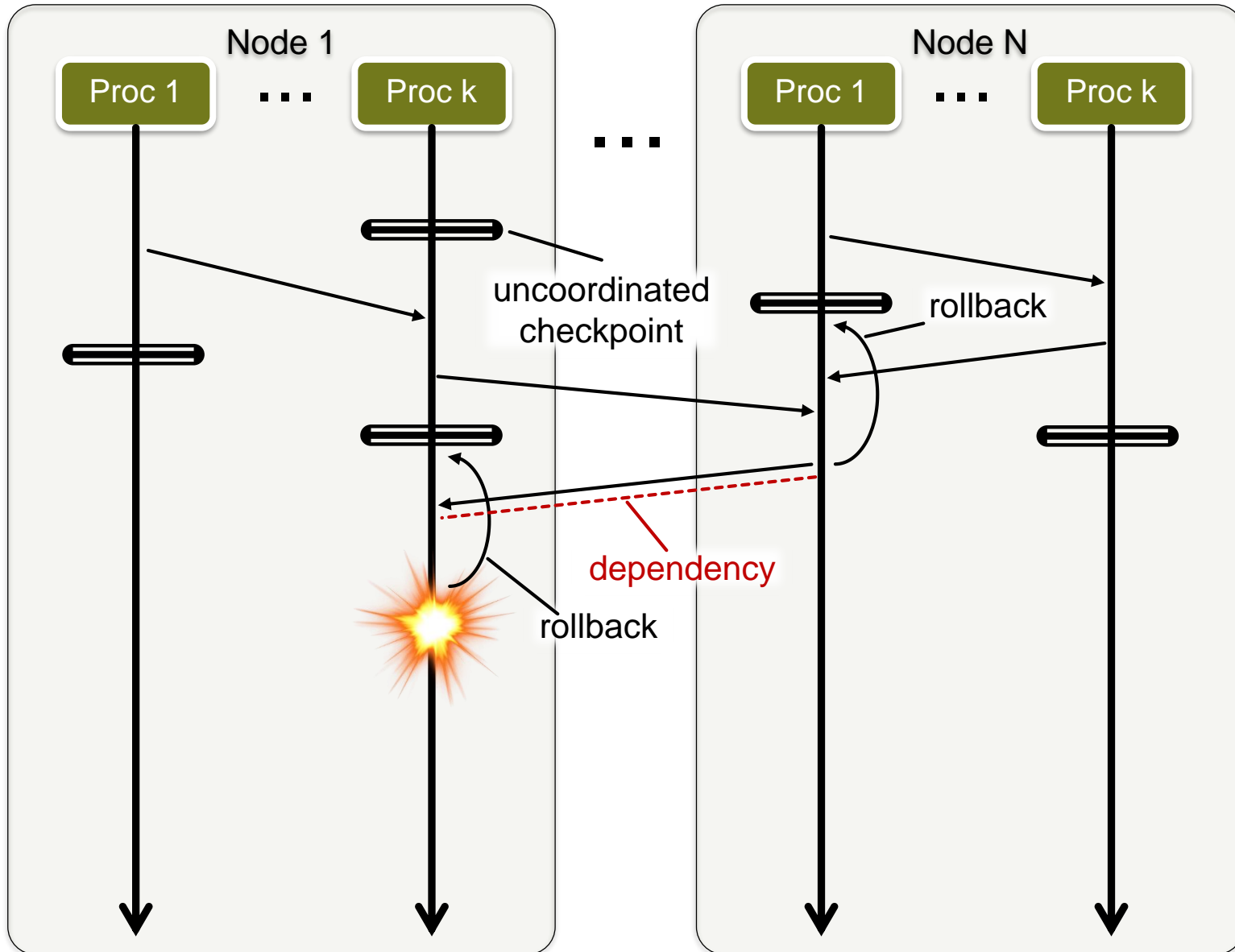
UNCOORDINATED CHECKPOINTING (MP)



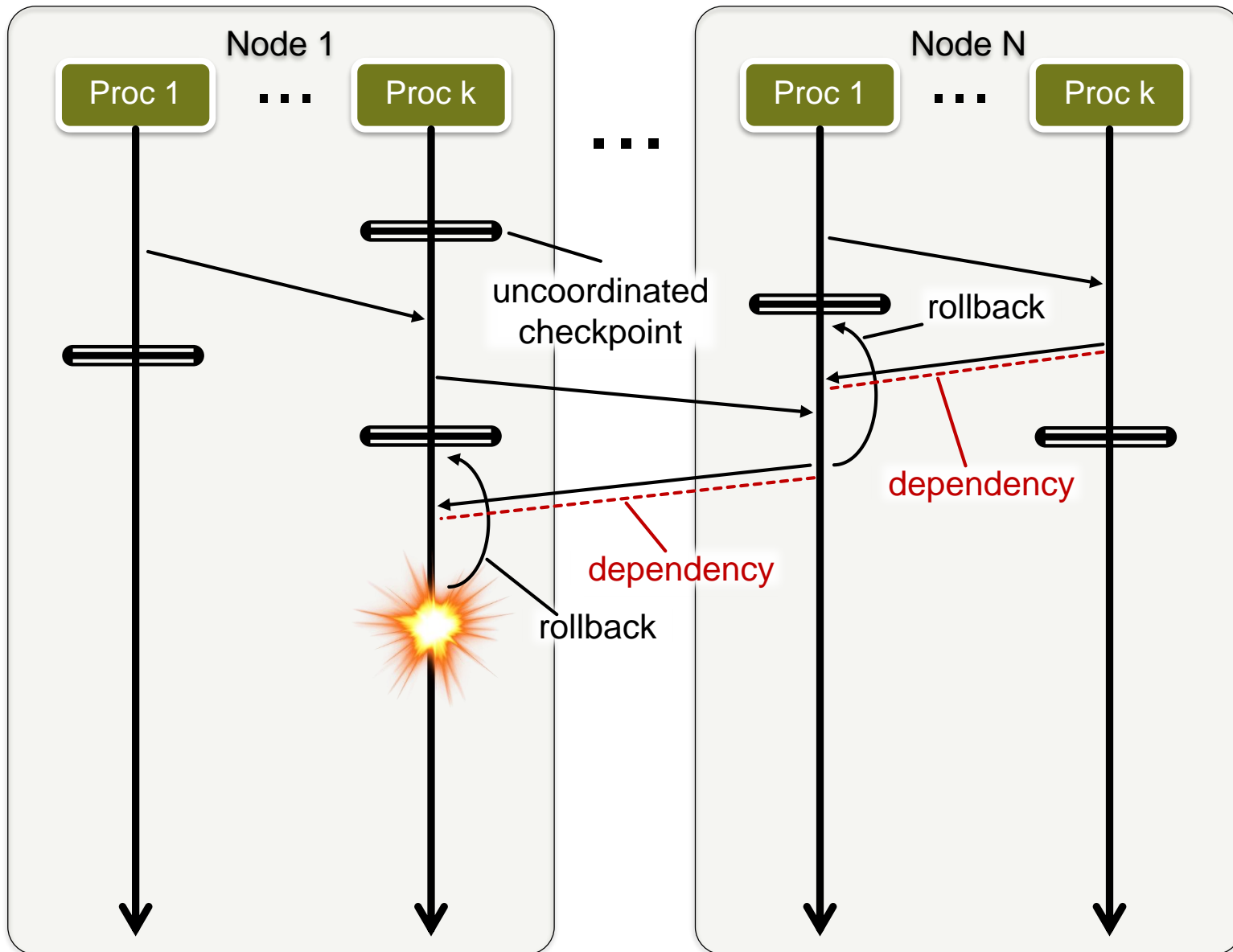
UNCOORDINATED CHECKPOINTING (MP)



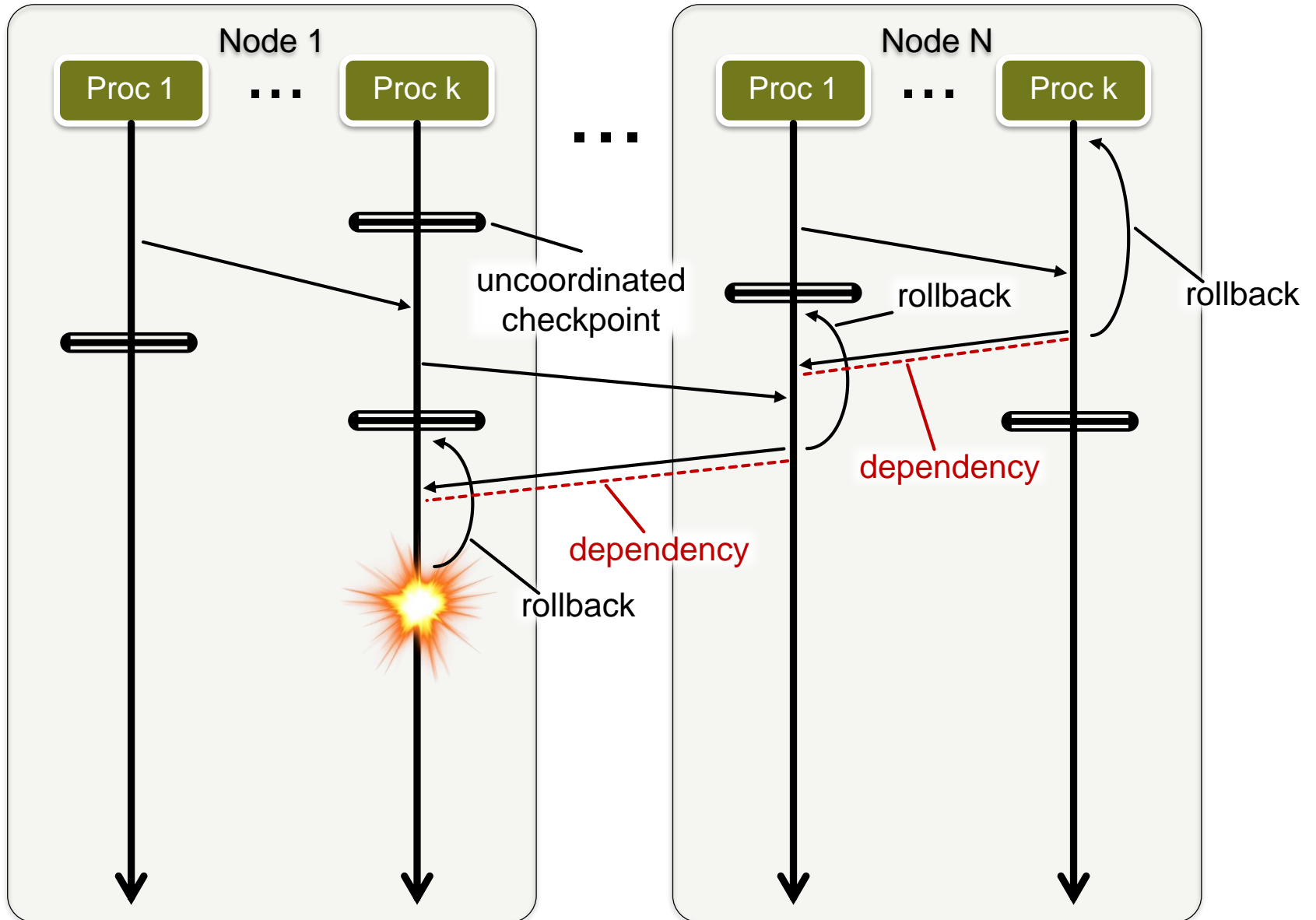
UNCOORDINATED CHECKPOINTING (MP)



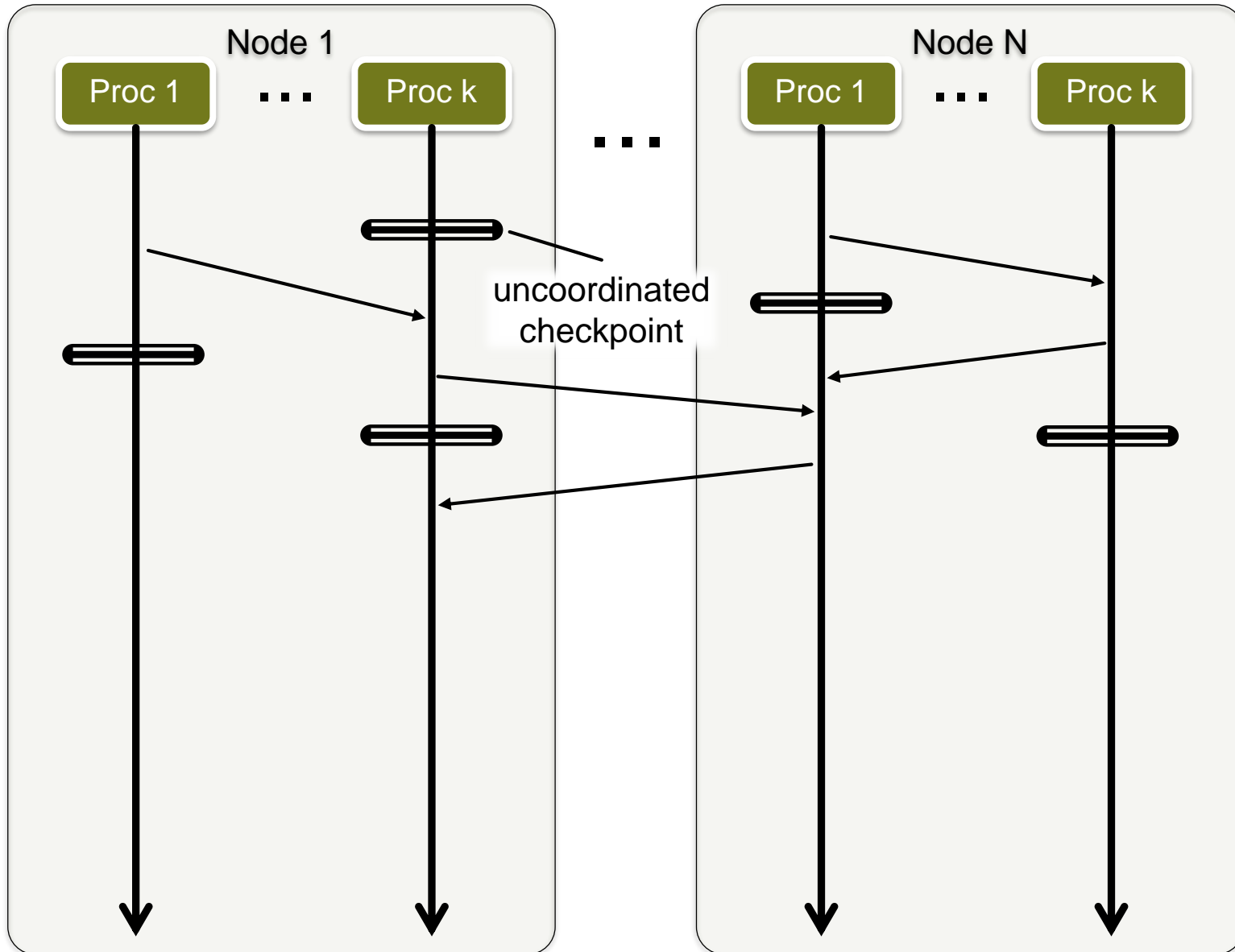
UNCOORDINATED CHECKPOINTING (MP)



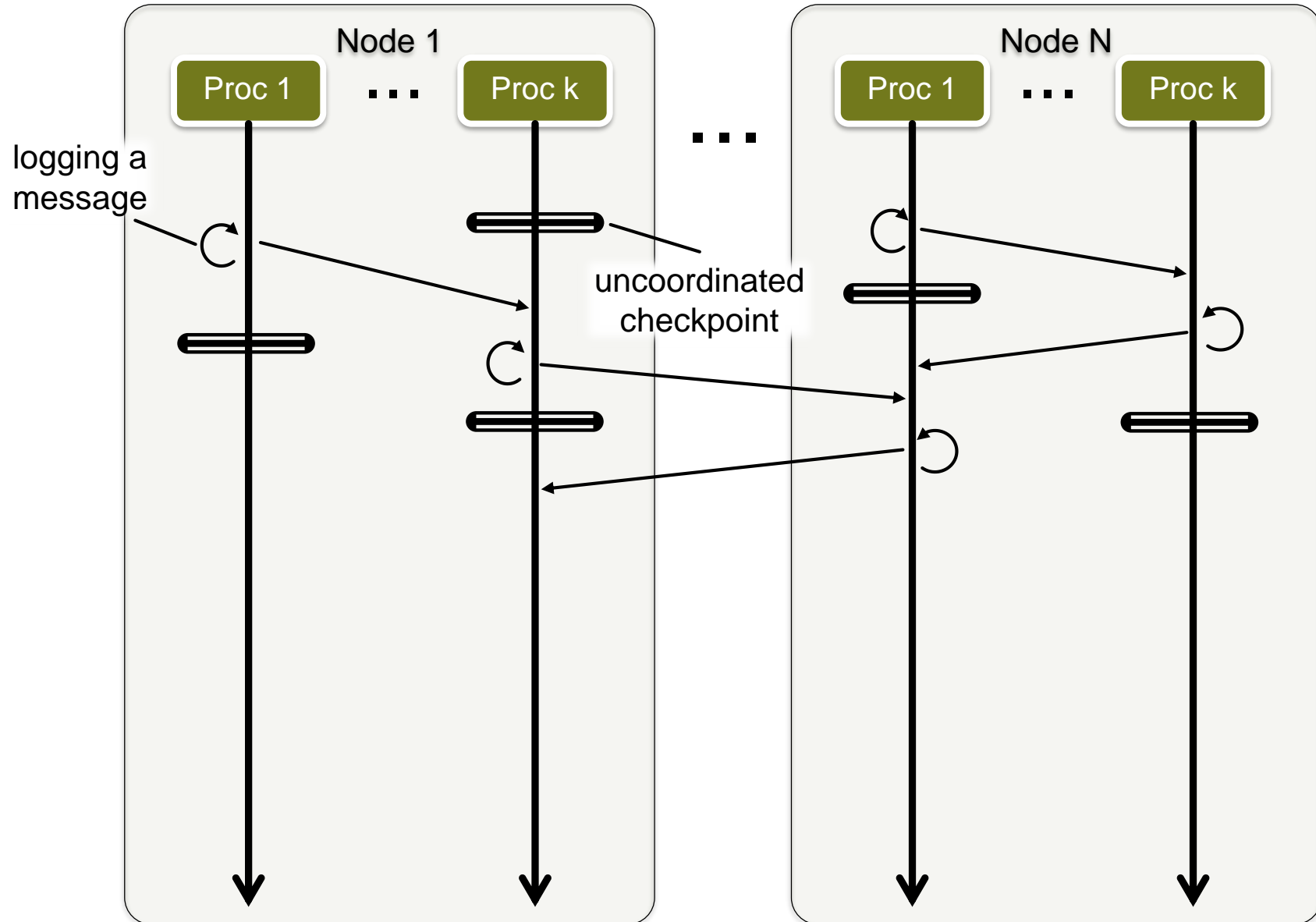
UNCOORDINATED CHECKPOINTING (MP)



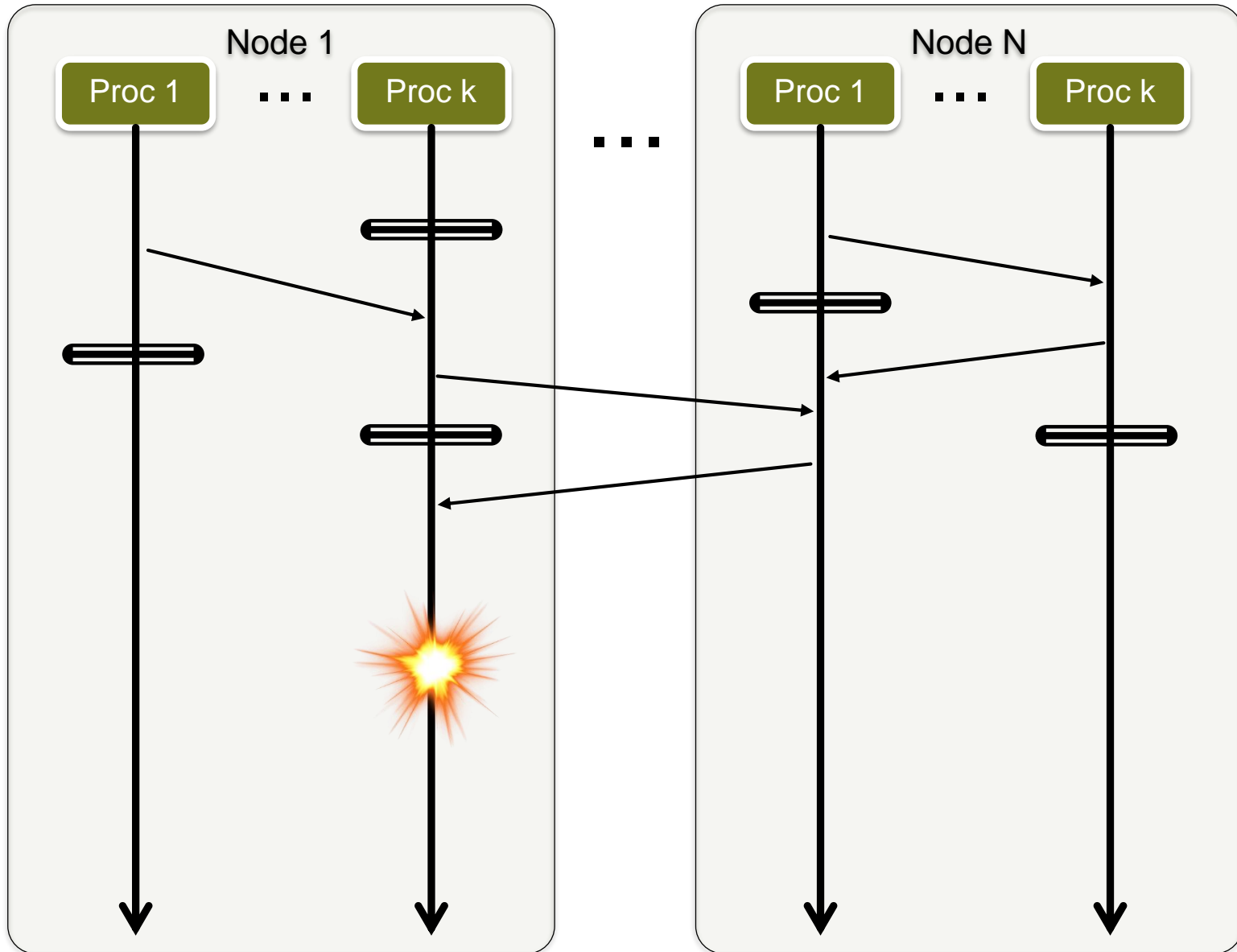
UNCOORDINATED CHECKPOINTING (MP)



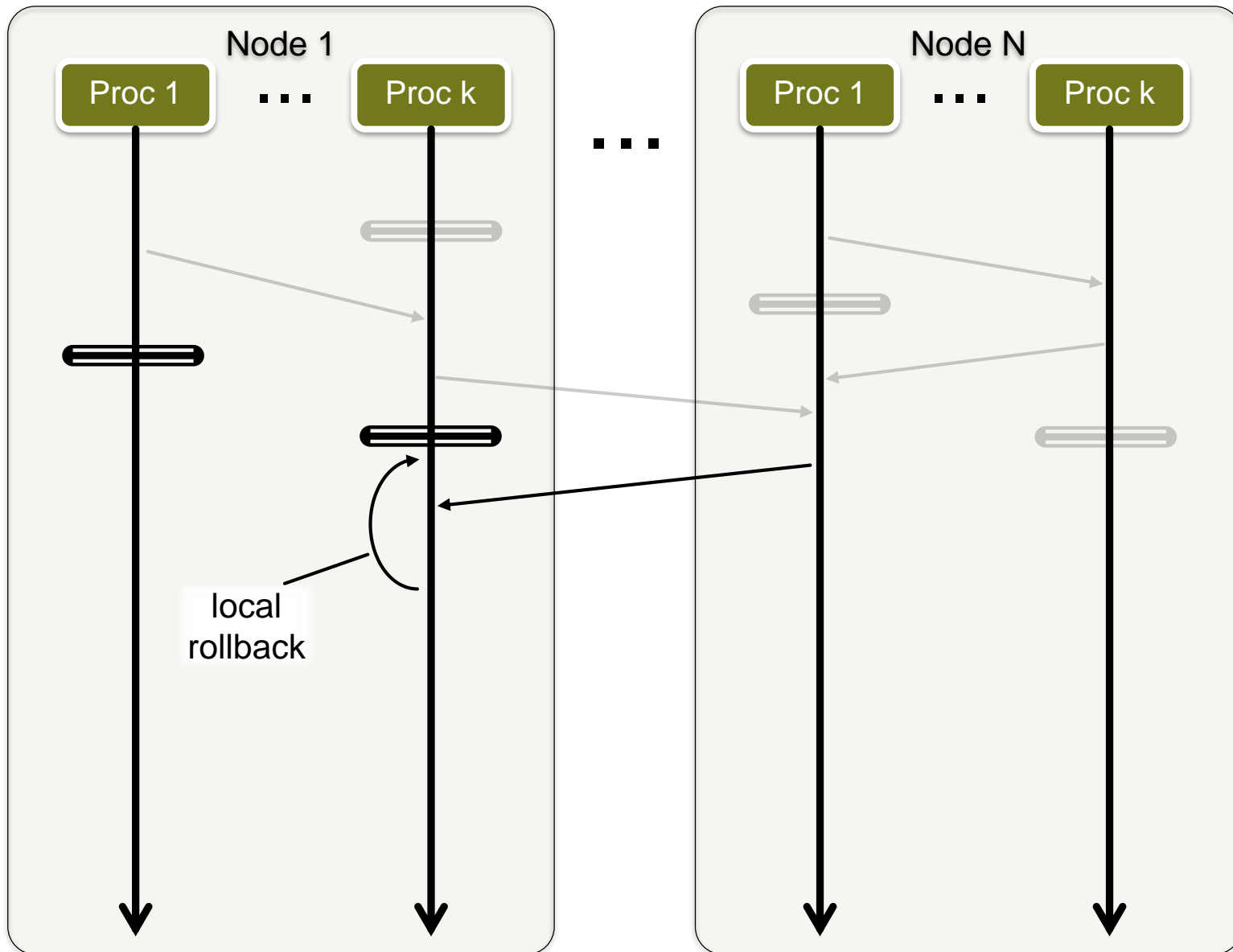
UNCOORDINATED CHECKPOINTING (MP)



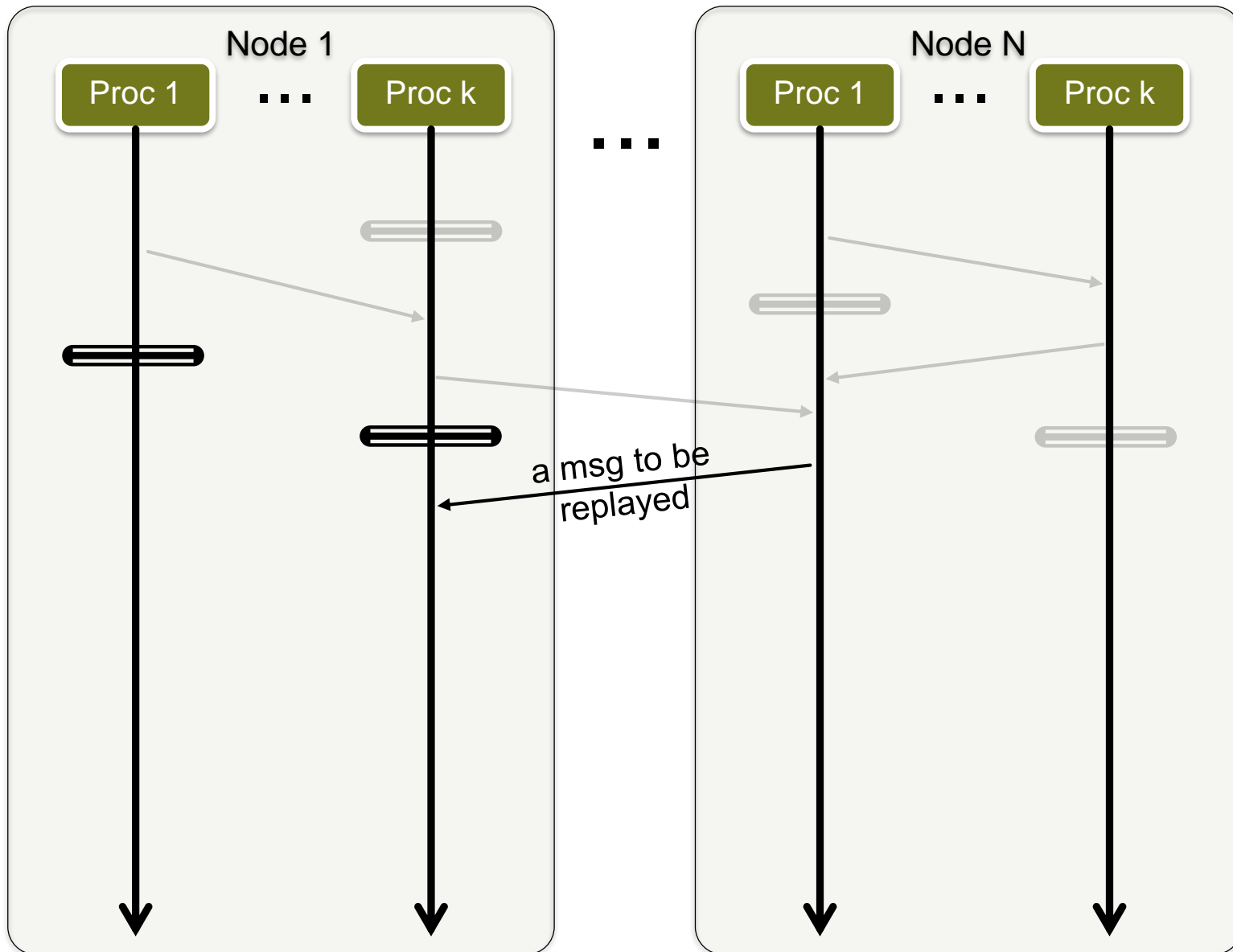
UNCOORDINATED CHECKPOINTING (MP)



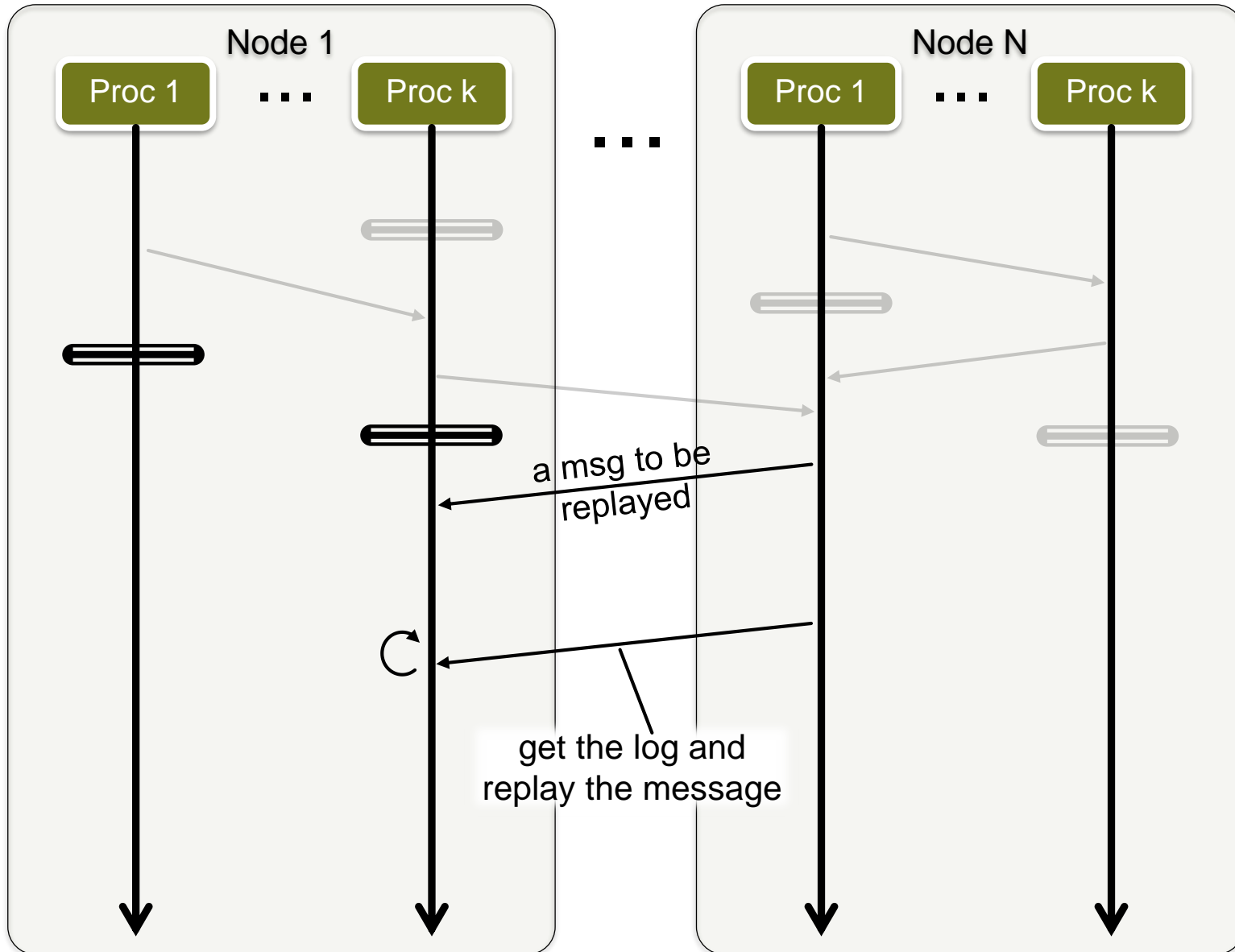
UNCOORDINATED CHECKPOINTING (MP)



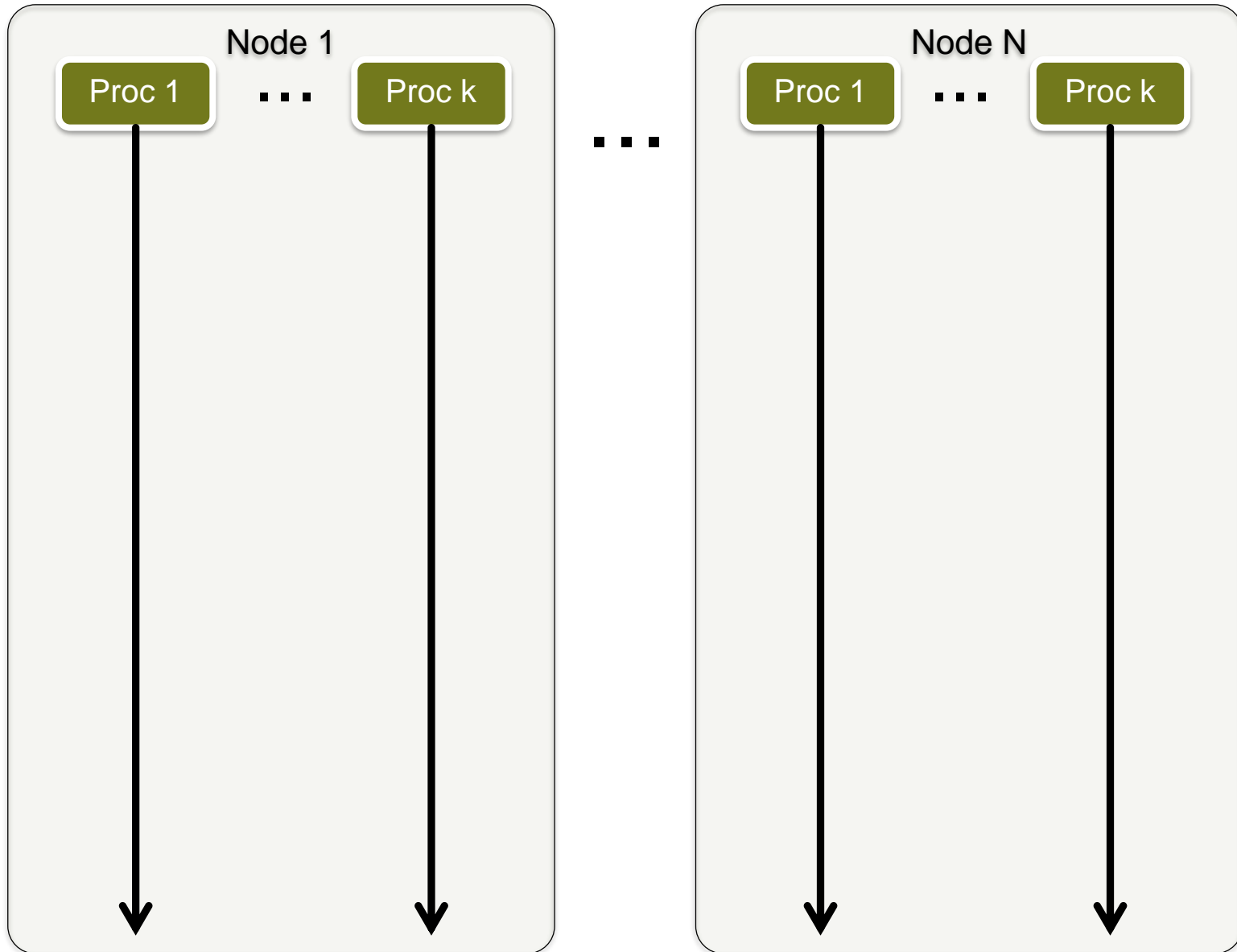
UNCOORDINATED CHECKPOINTING (MP)



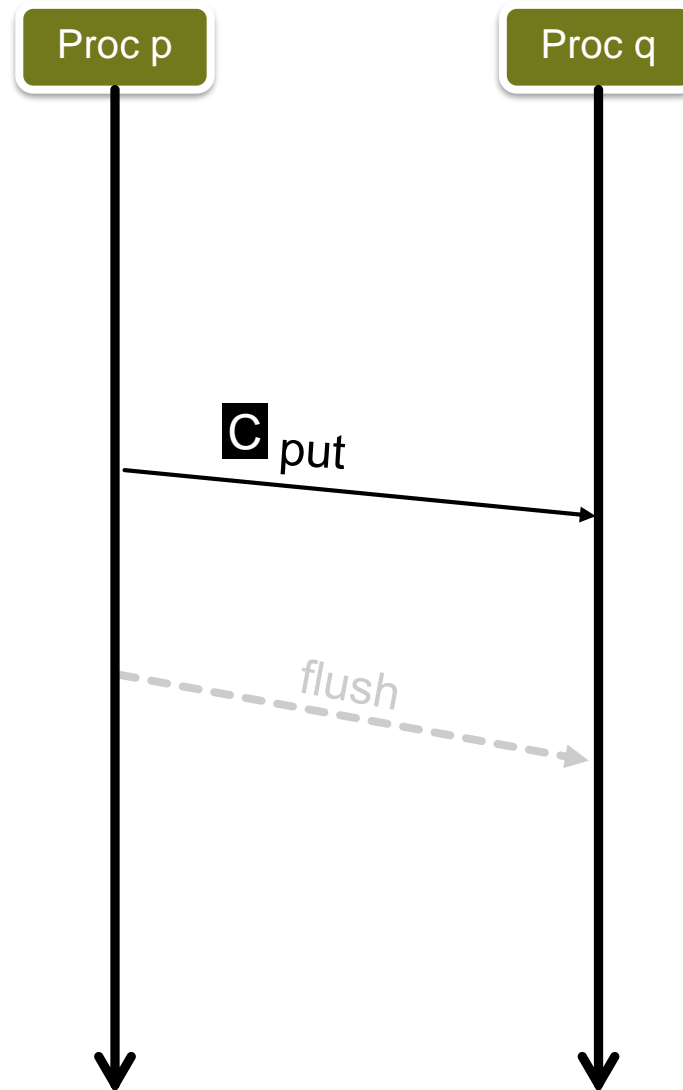
UNCOORDINATED CHECKPOINTING (MP)



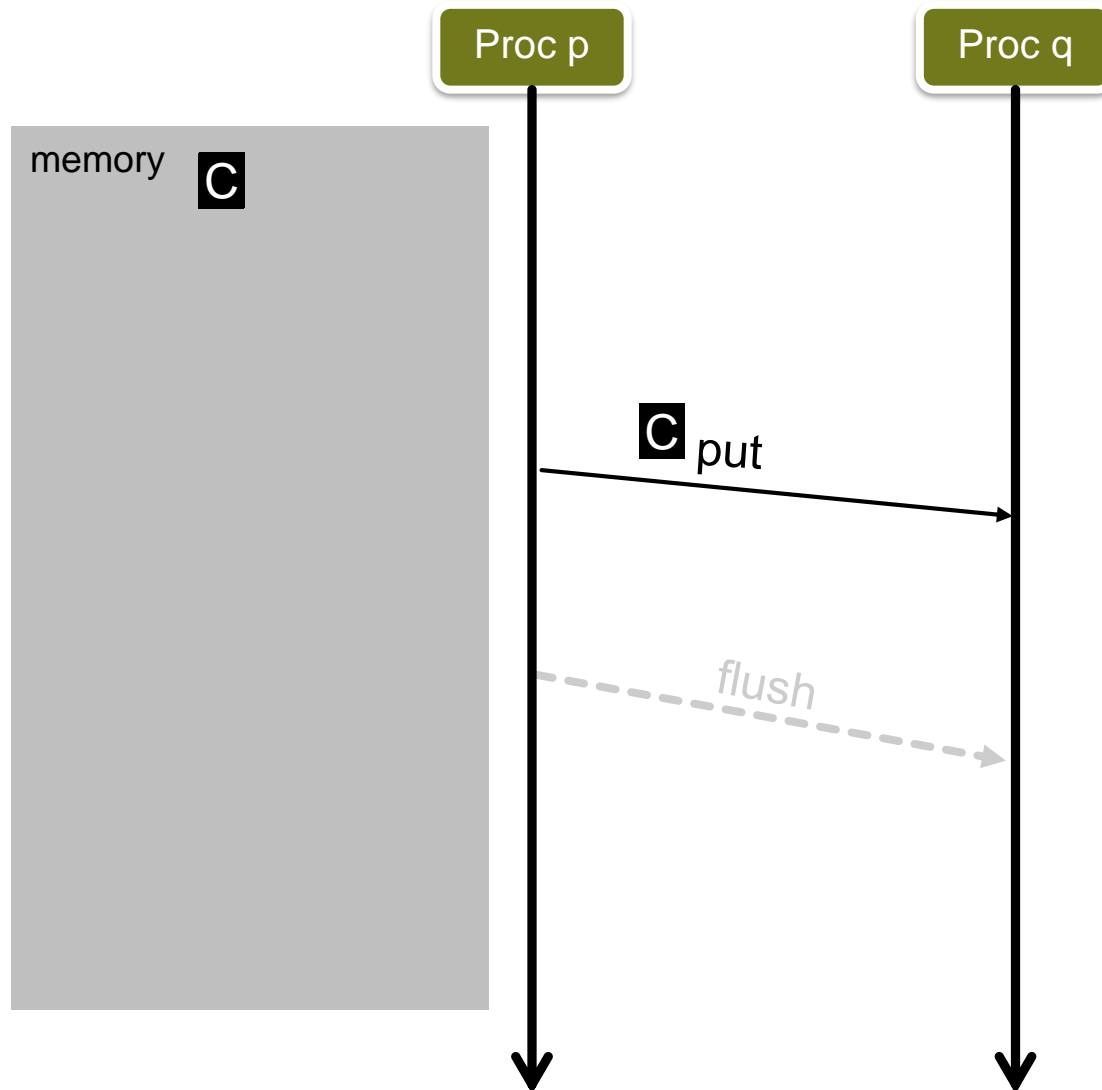
UNCOORDINATED CHECKPOINTING (MP)



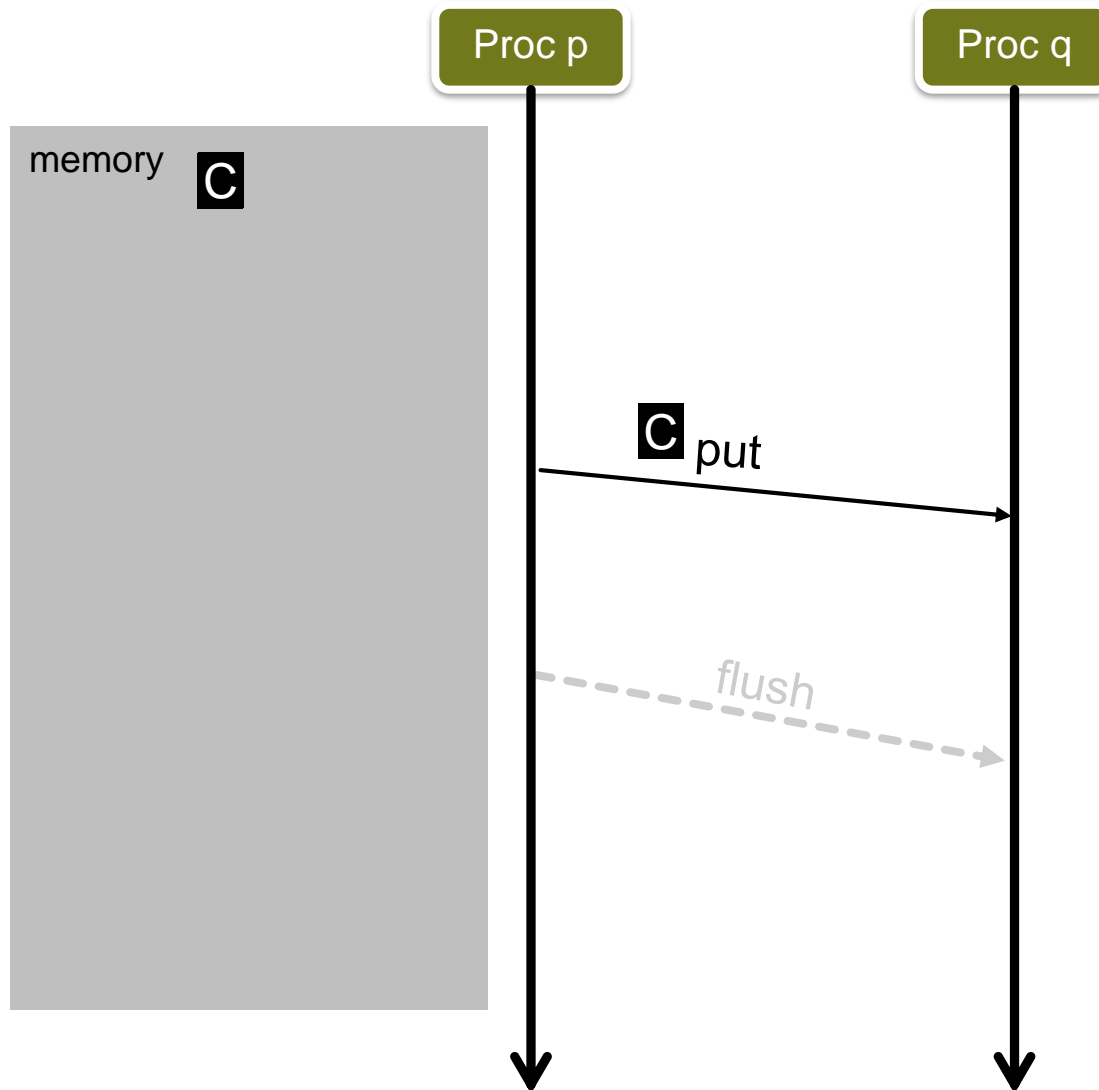
RMA: LOGGING PUTS



RMA: LOGGING PUTS

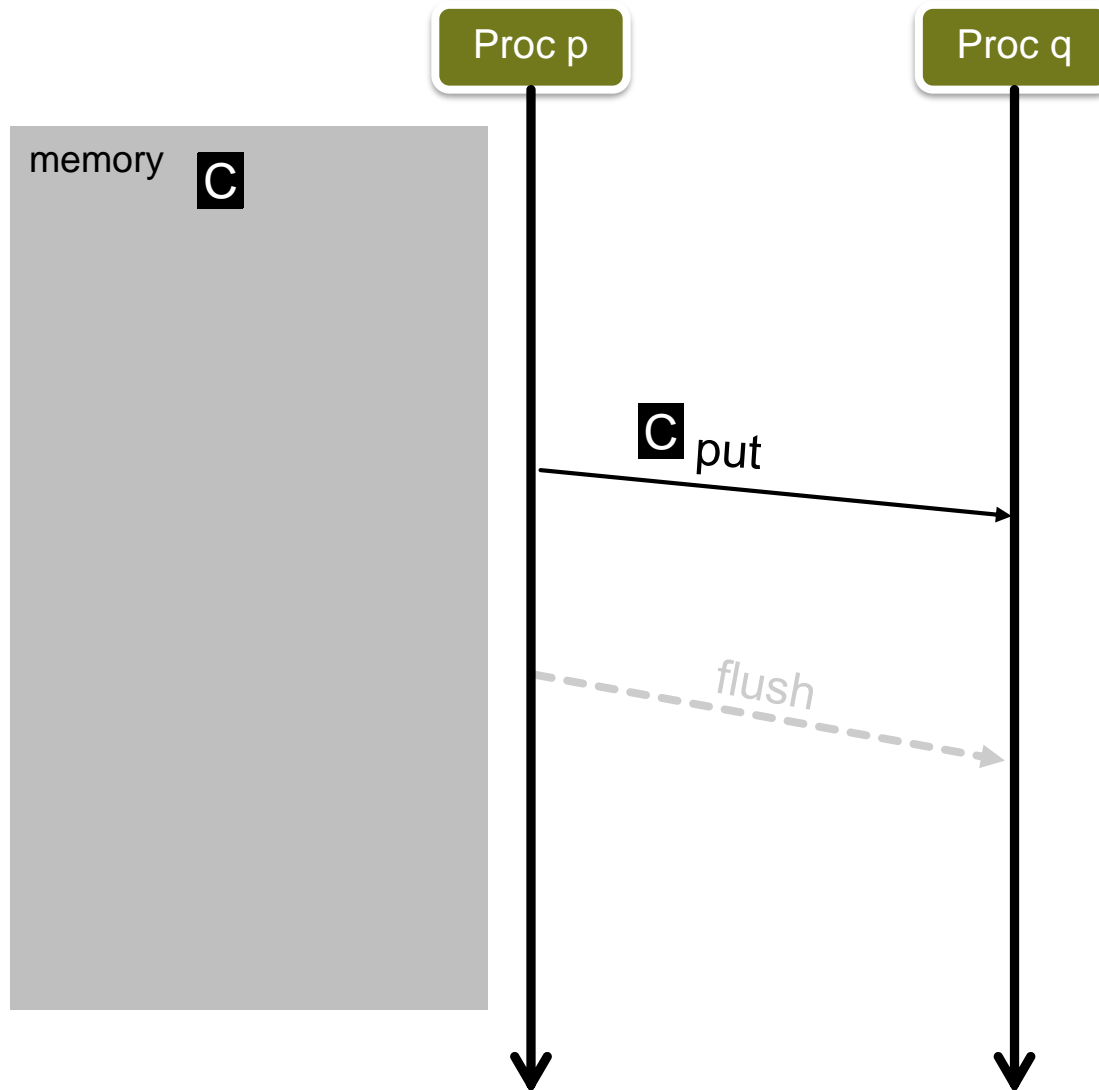


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$


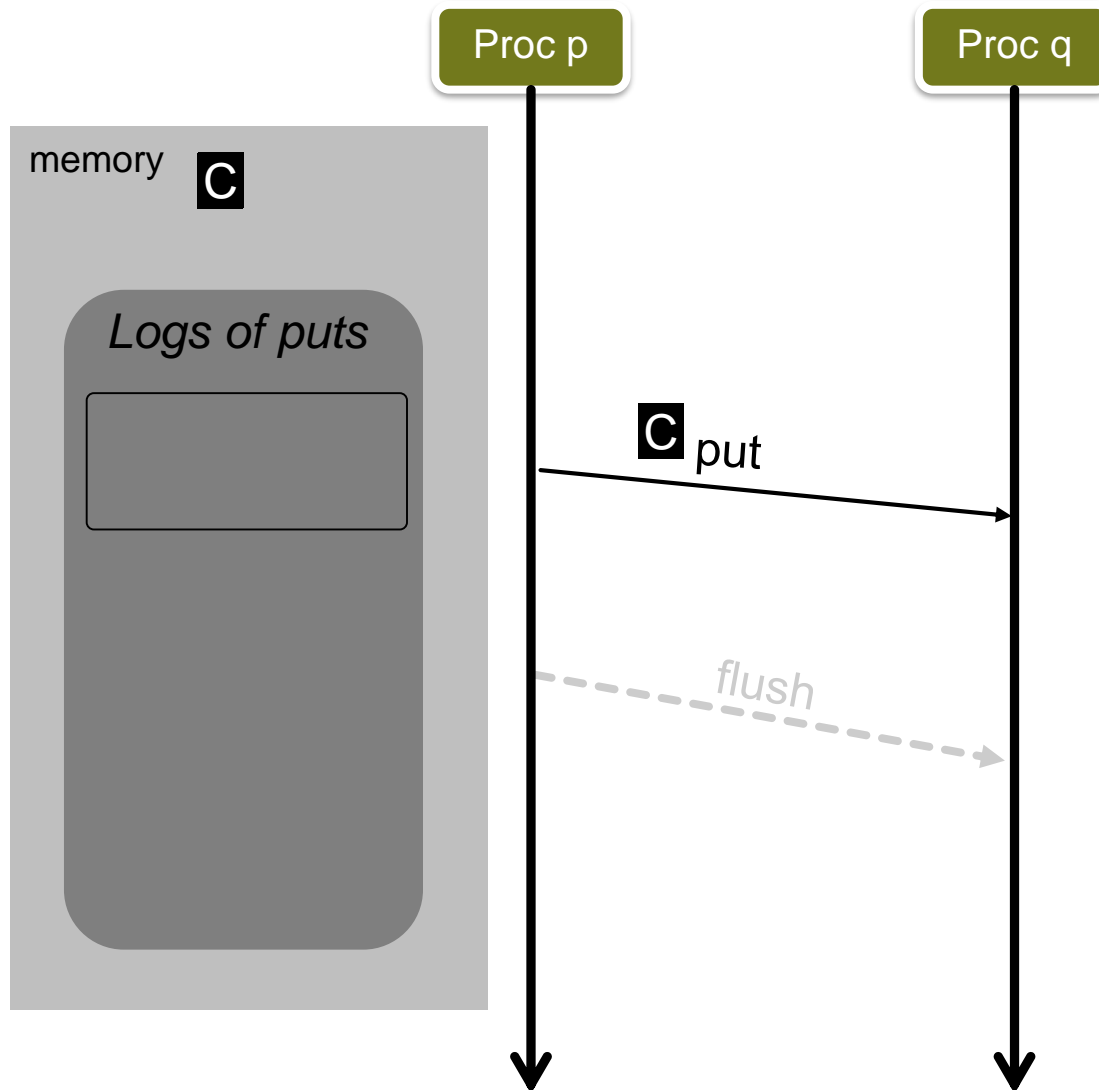
RMA: LOGGING PUTS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



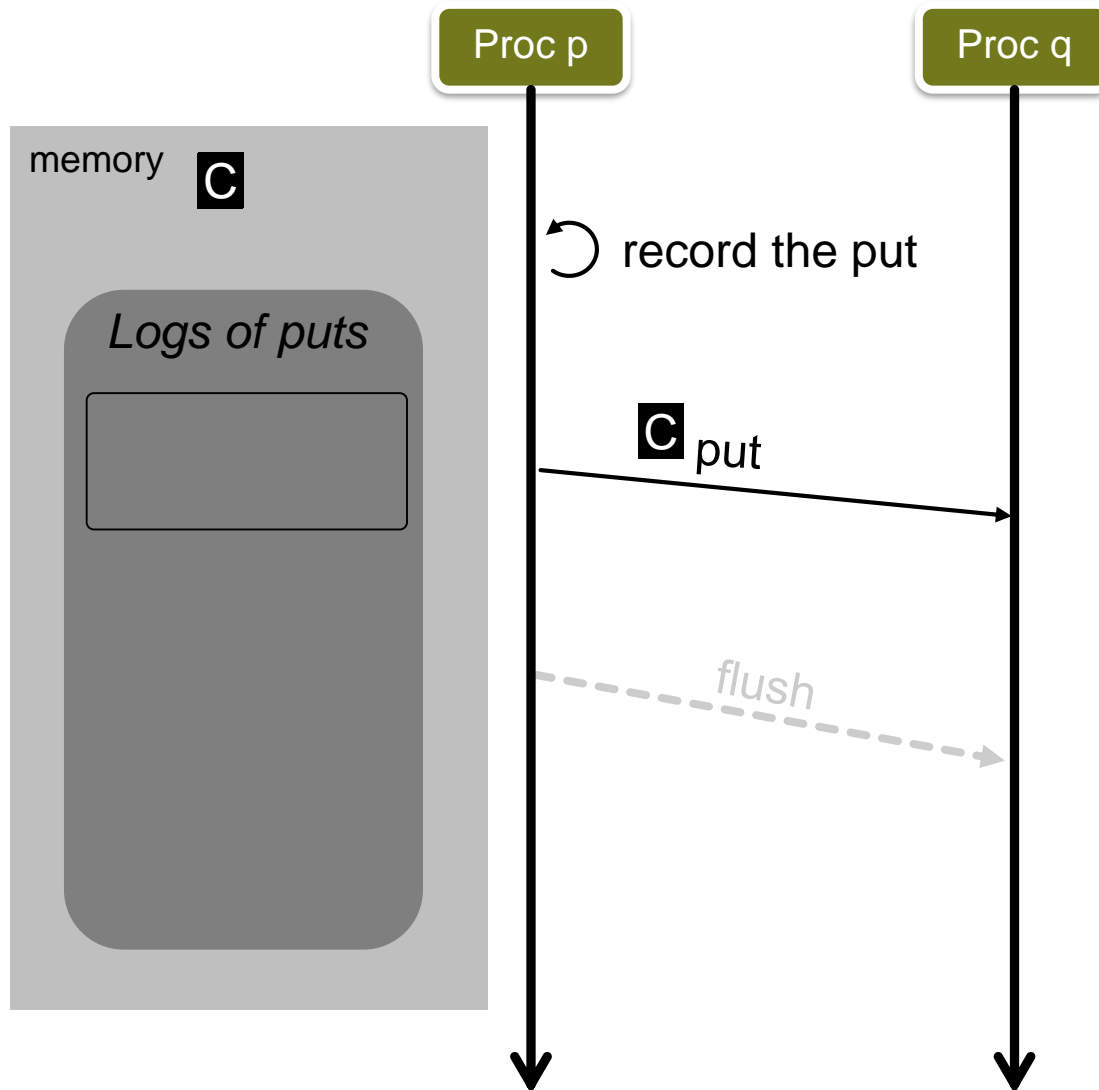
RMA: LOGGING PUTS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

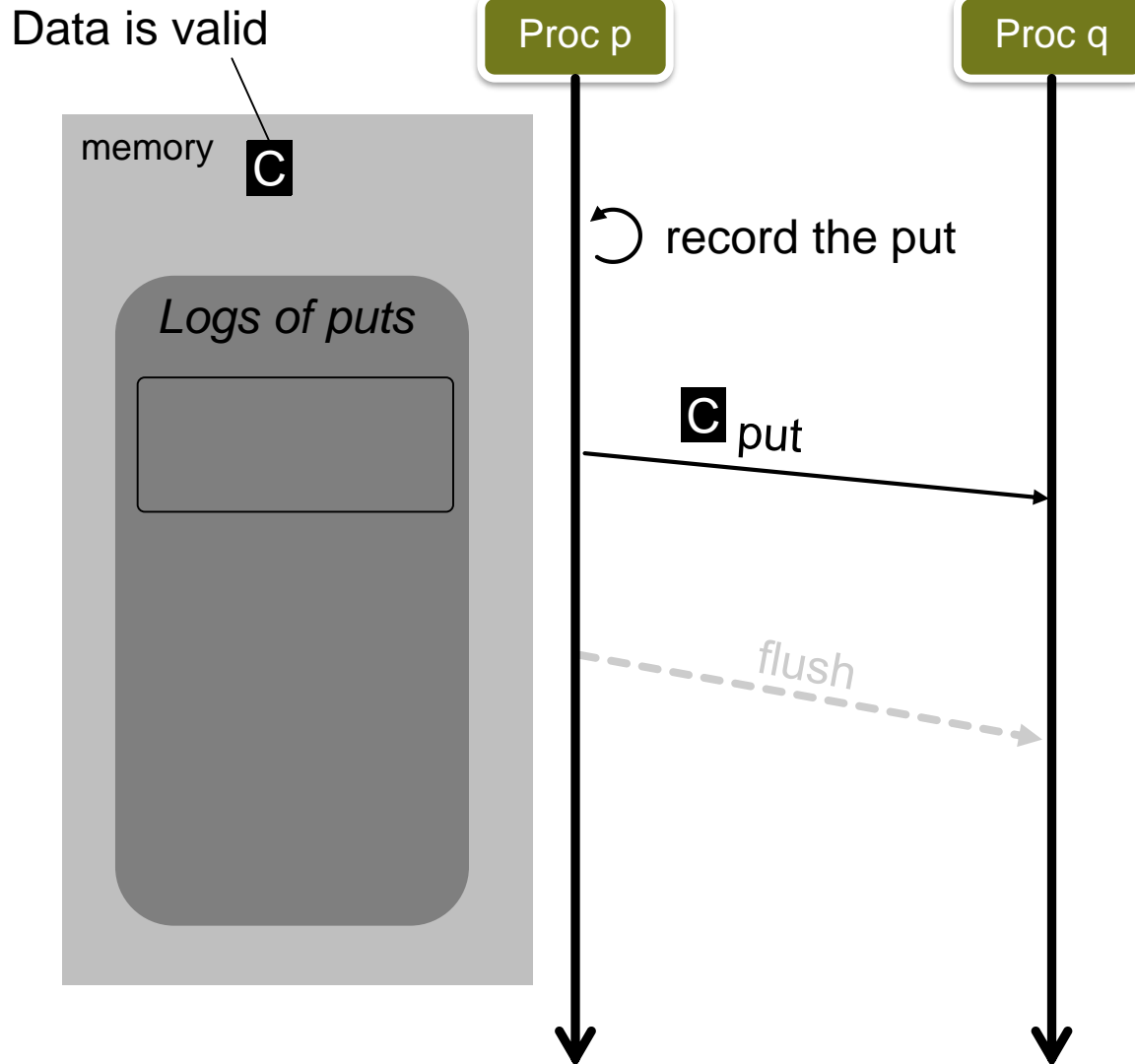


RMA: LOGGING PUTS

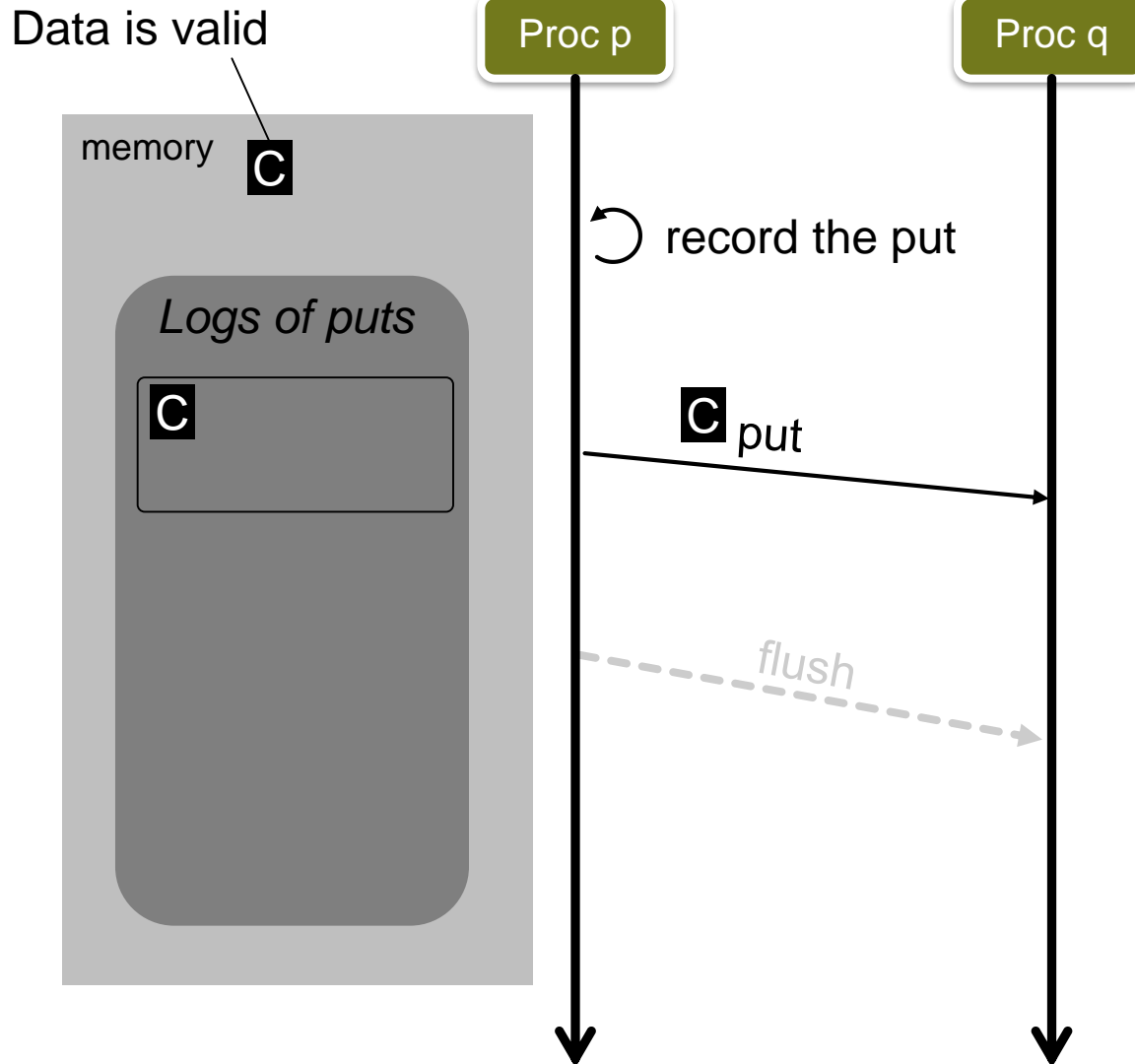
$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



RMA: LOGGING PUTS

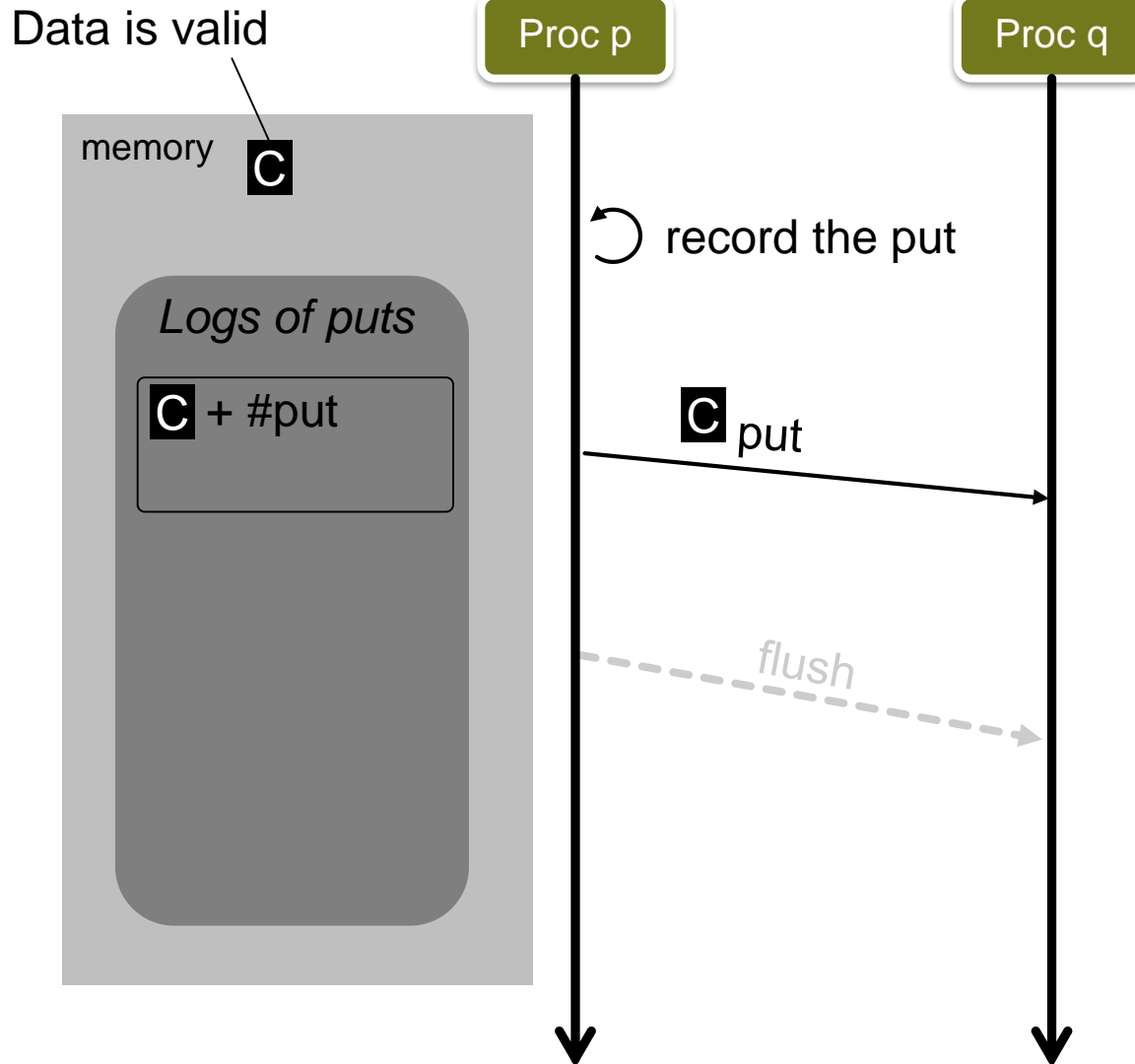
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

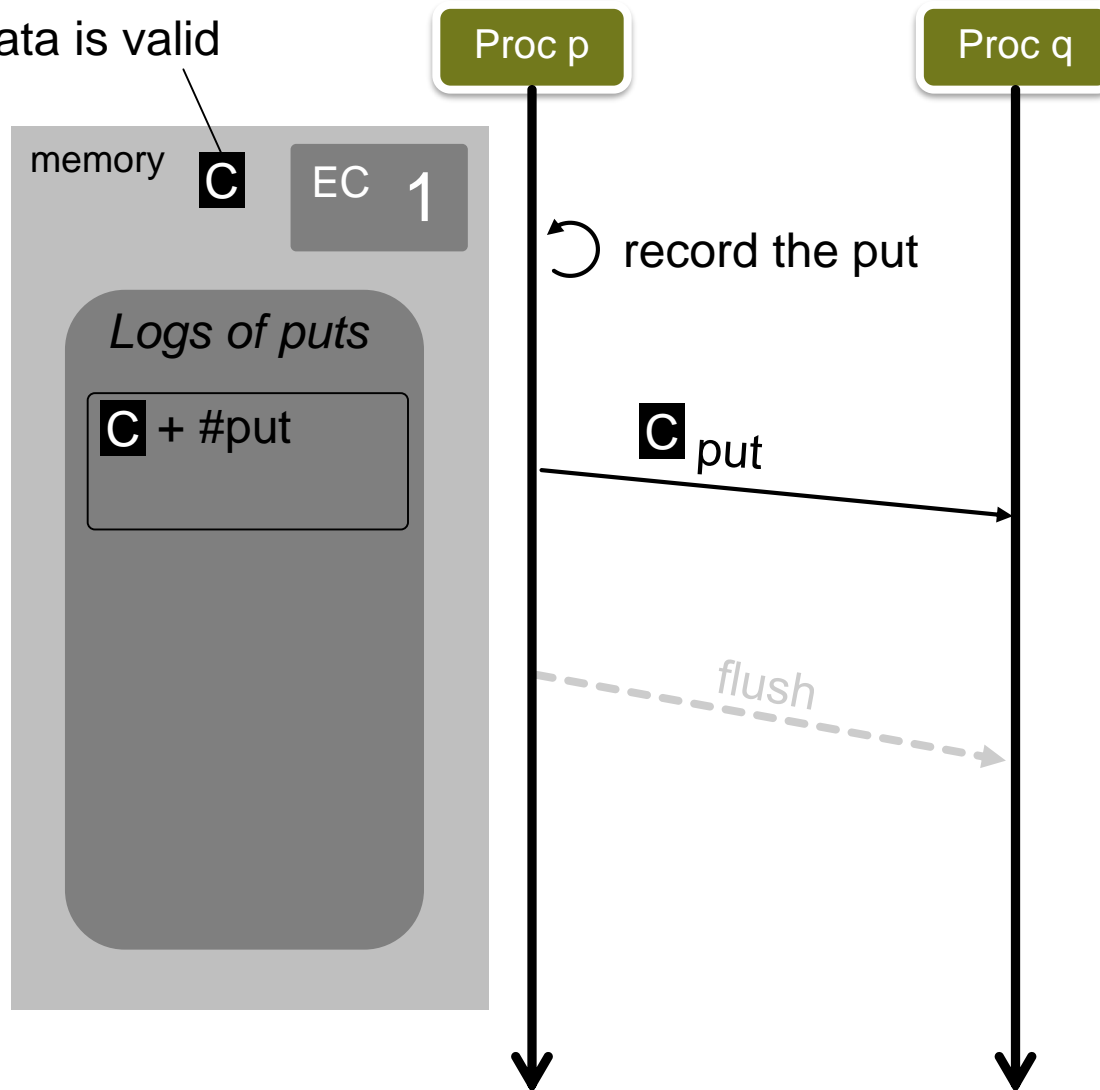
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid

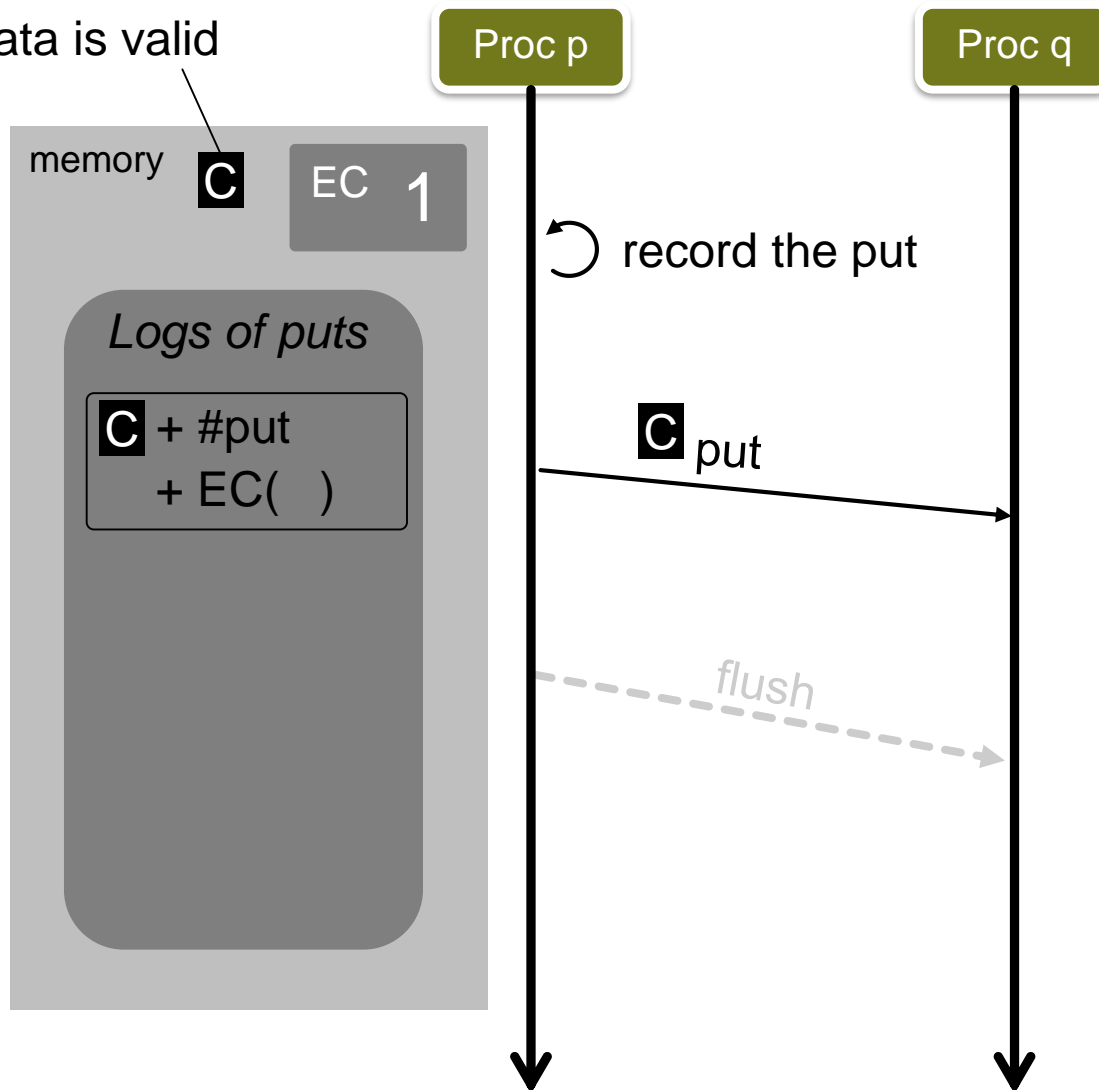


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid

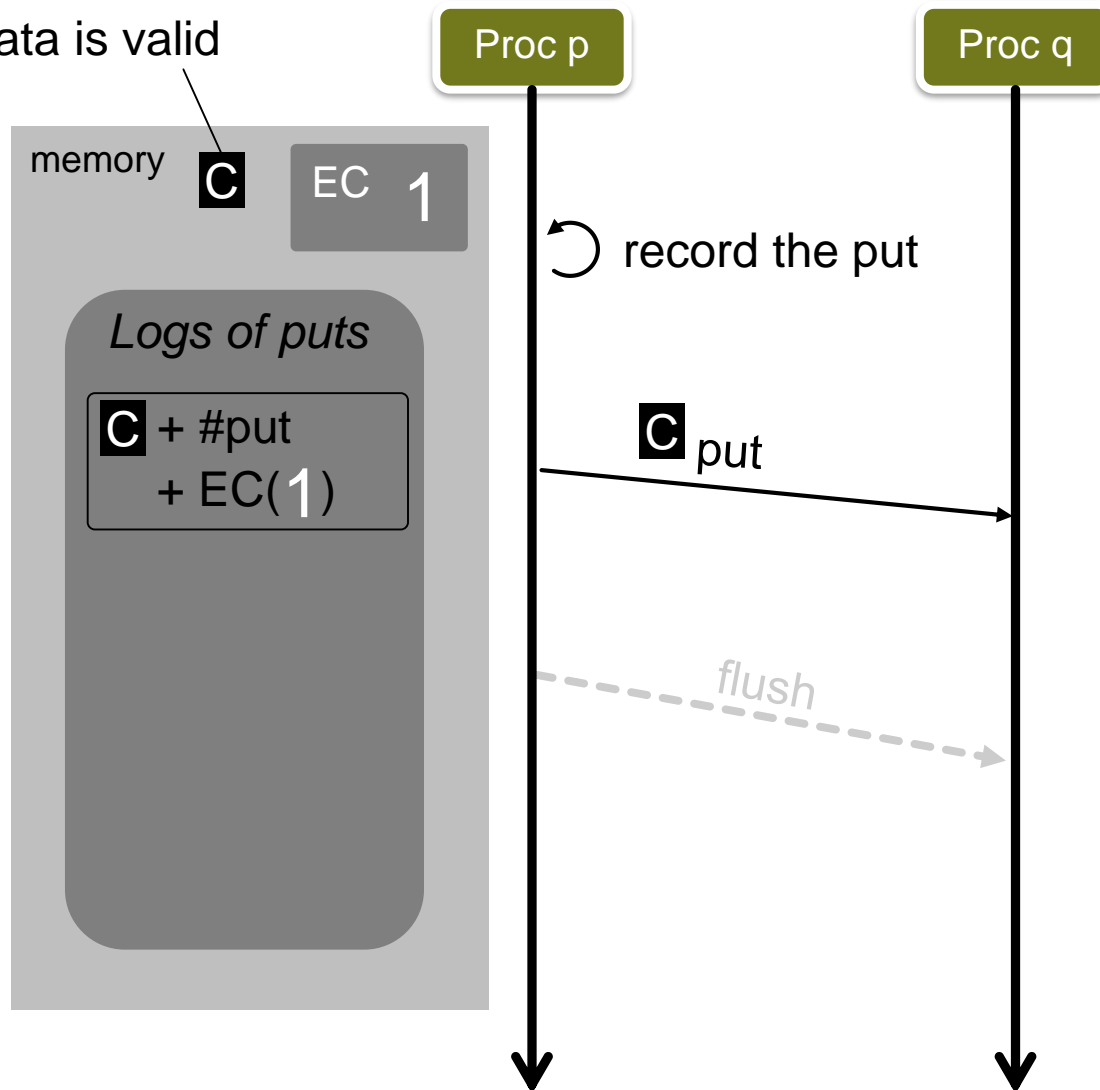


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

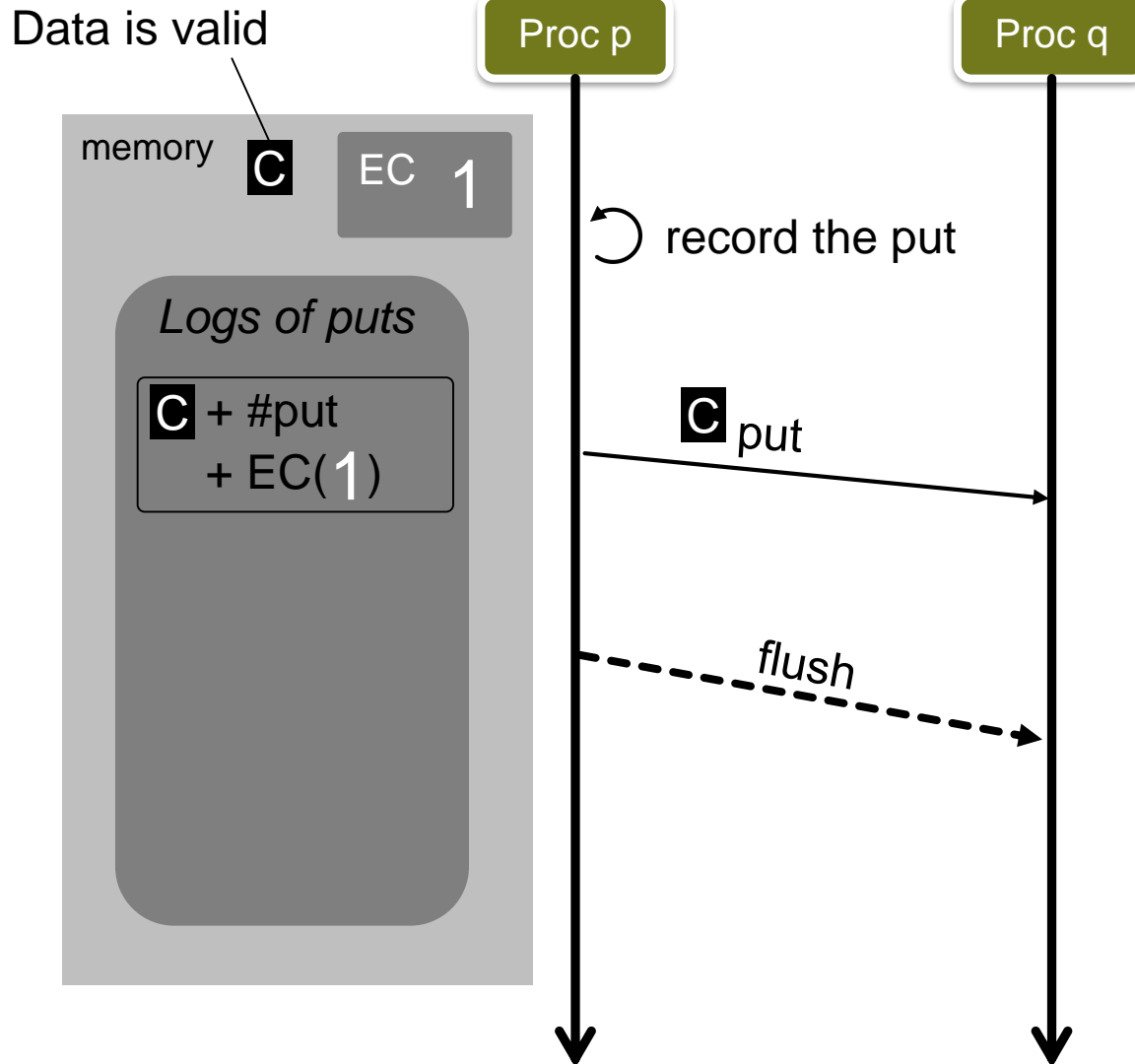
$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



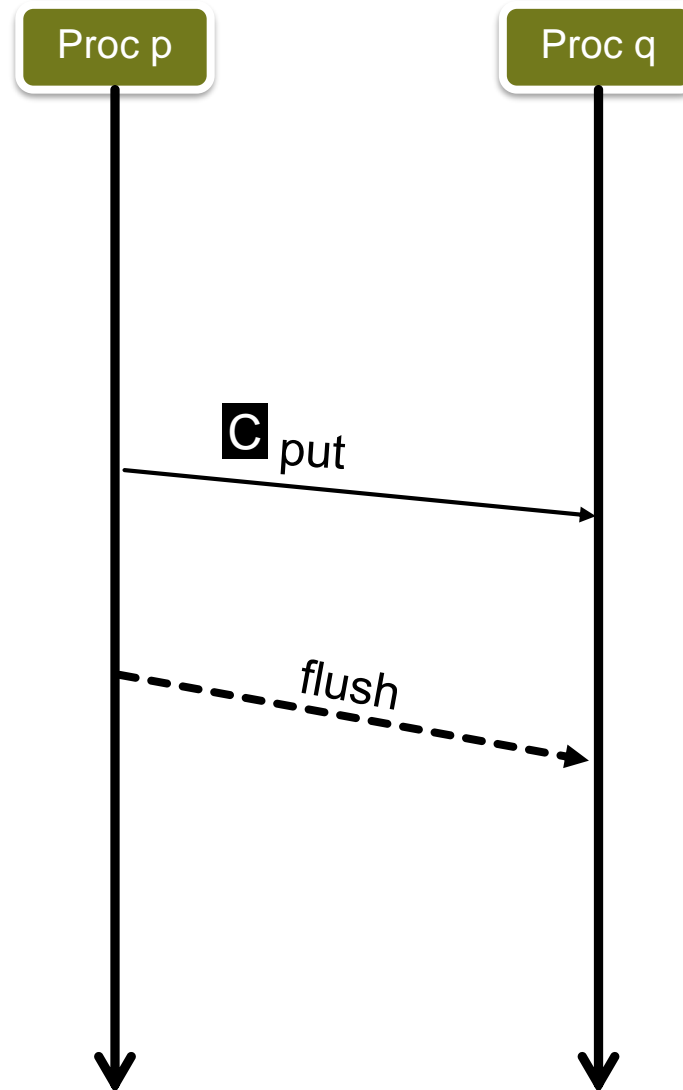
RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$


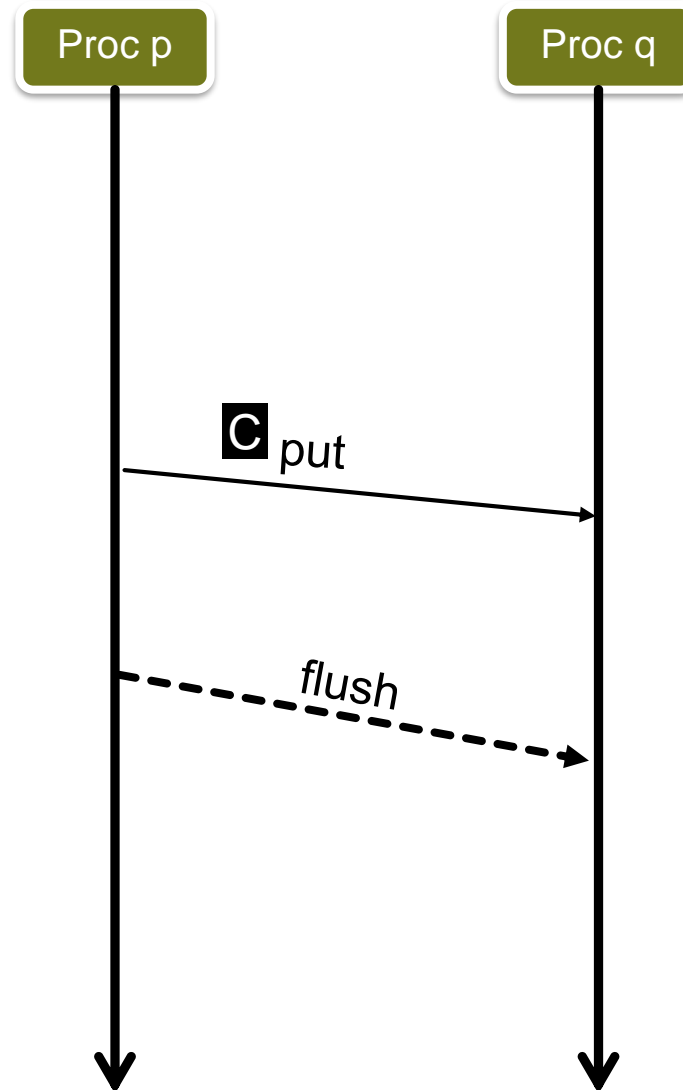
RMA: LOGGING PUTS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



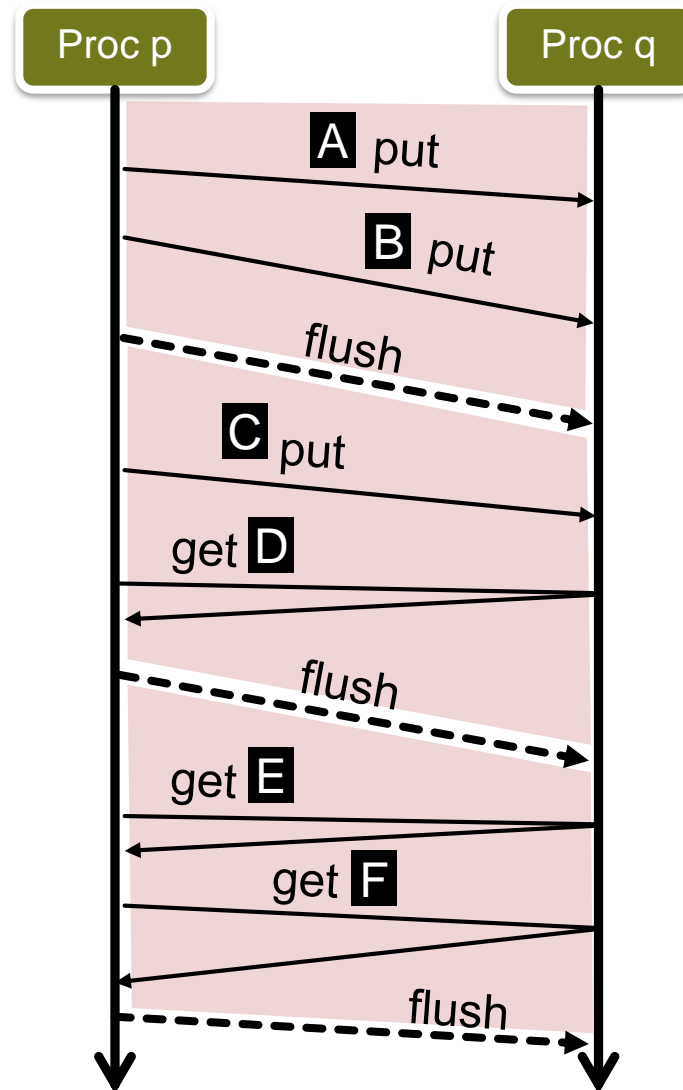
RMA: LOGGING GETS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



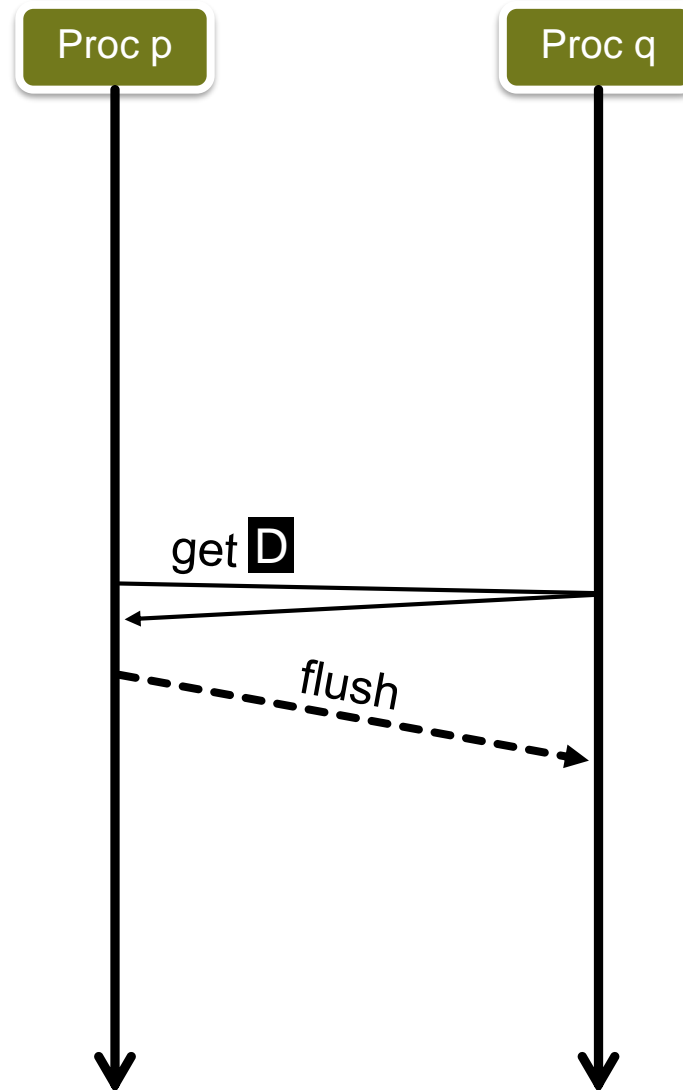
RMA: LOGGING GETS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

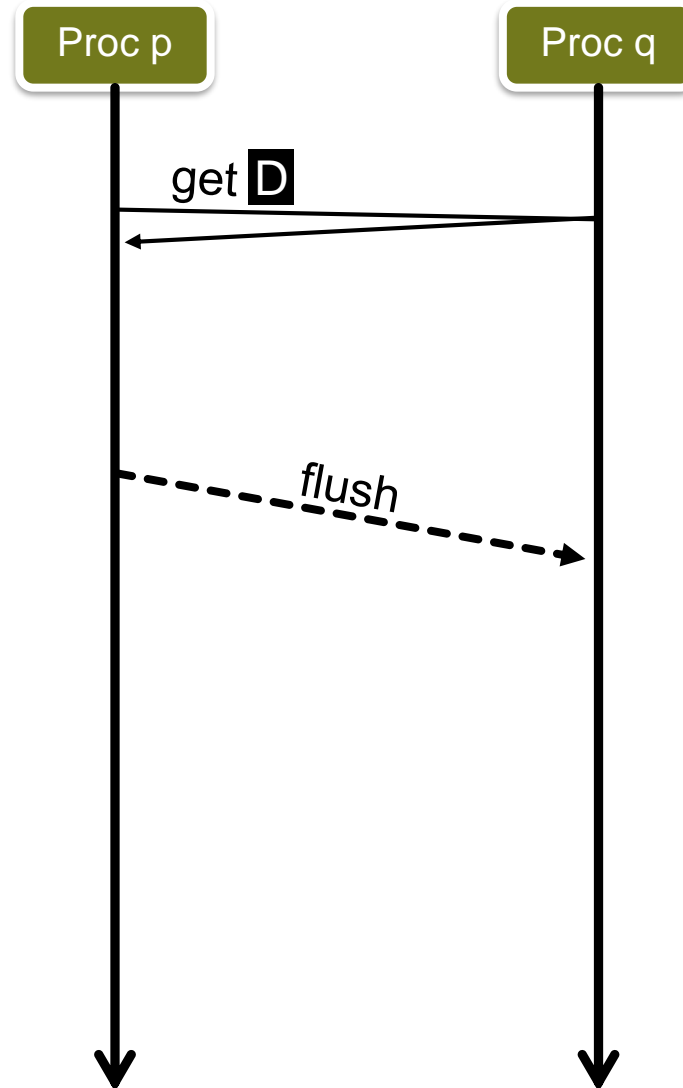


RMA: LOGGING GETS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

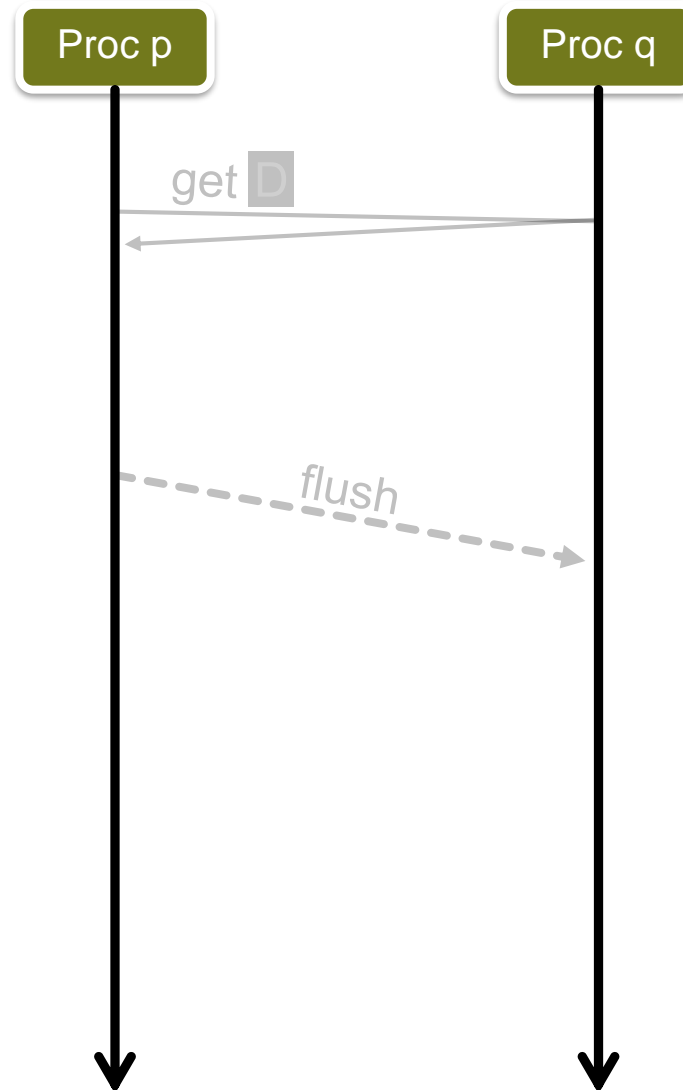


RMA: LOGGING GETS

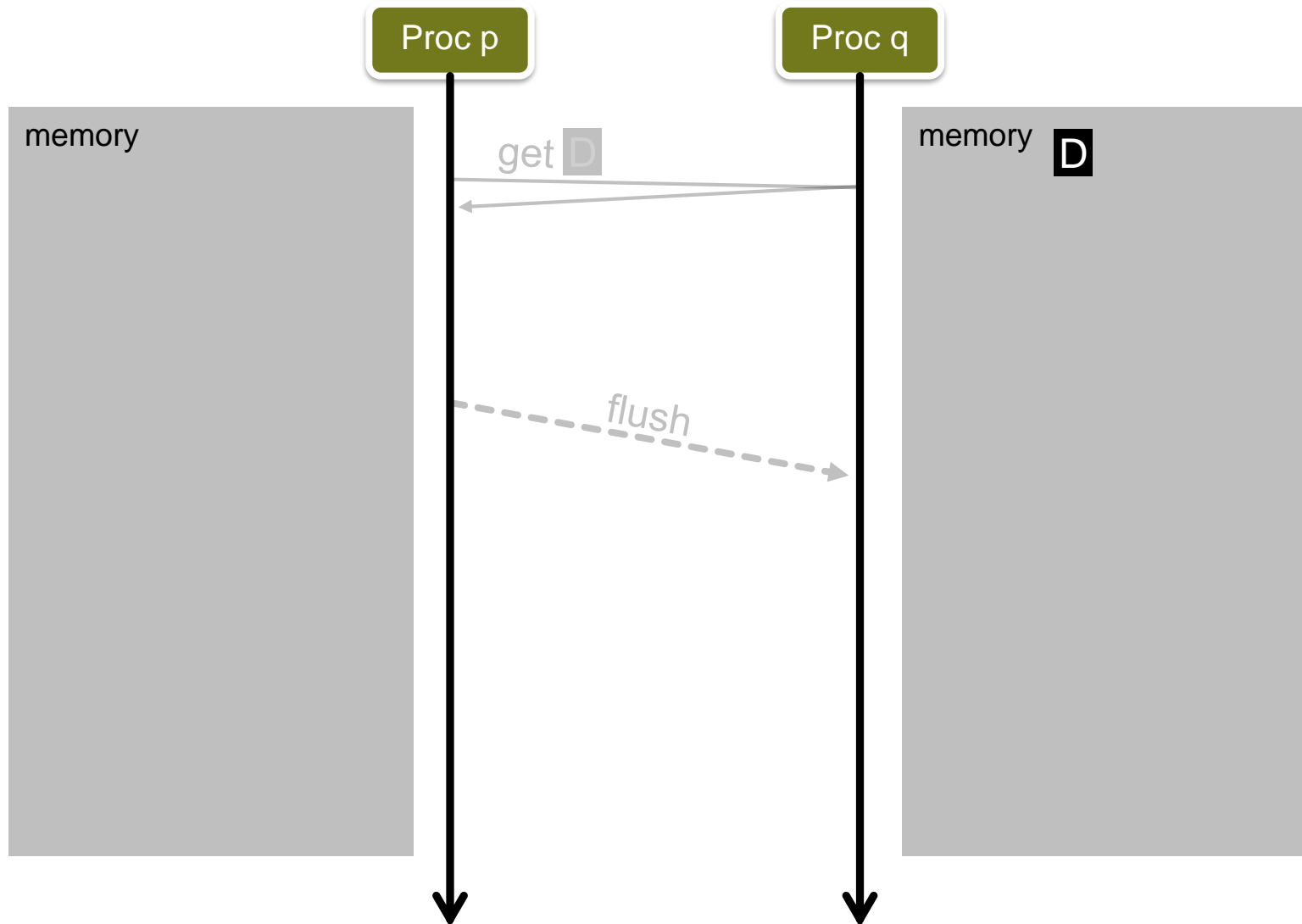
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS

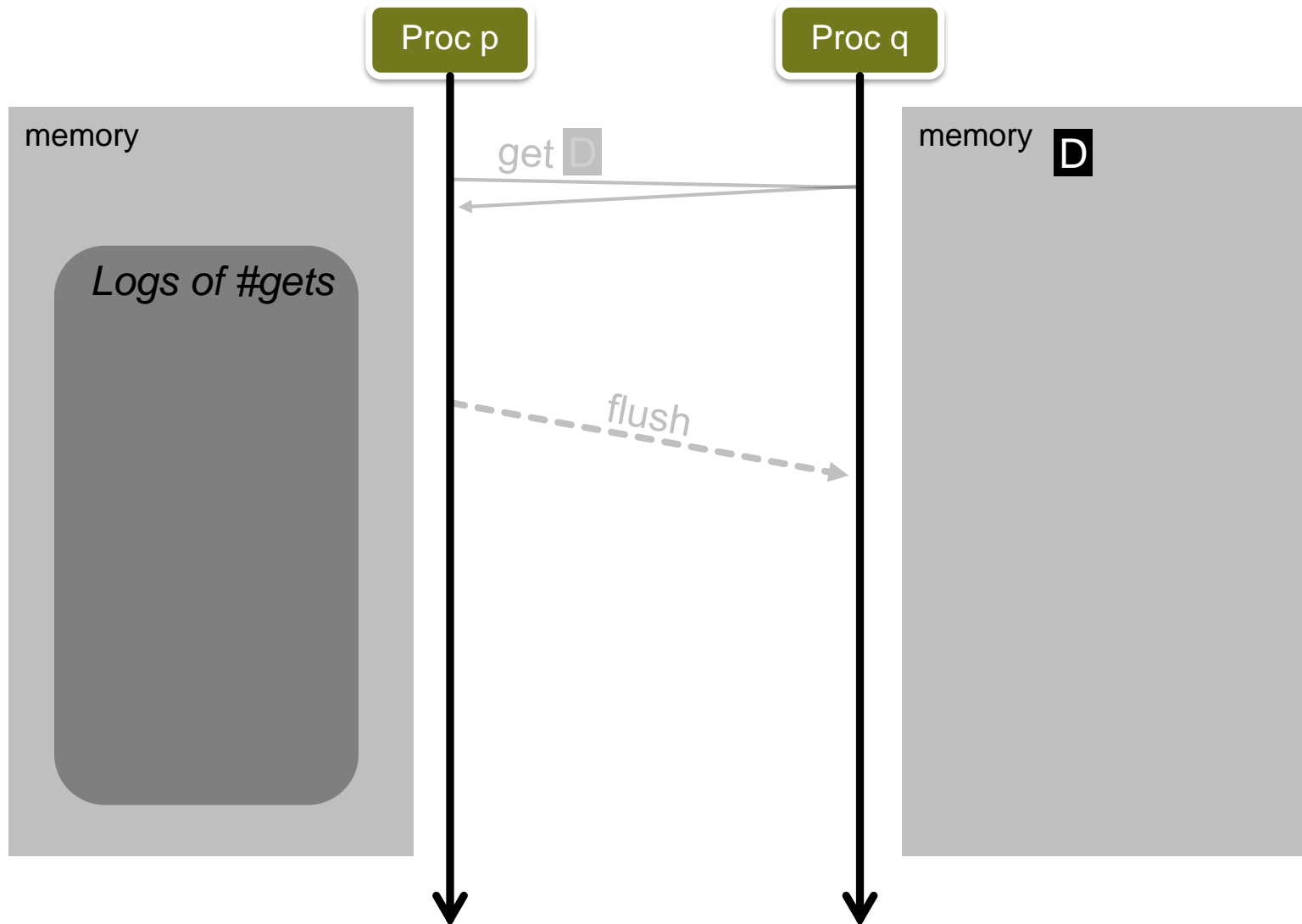
$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



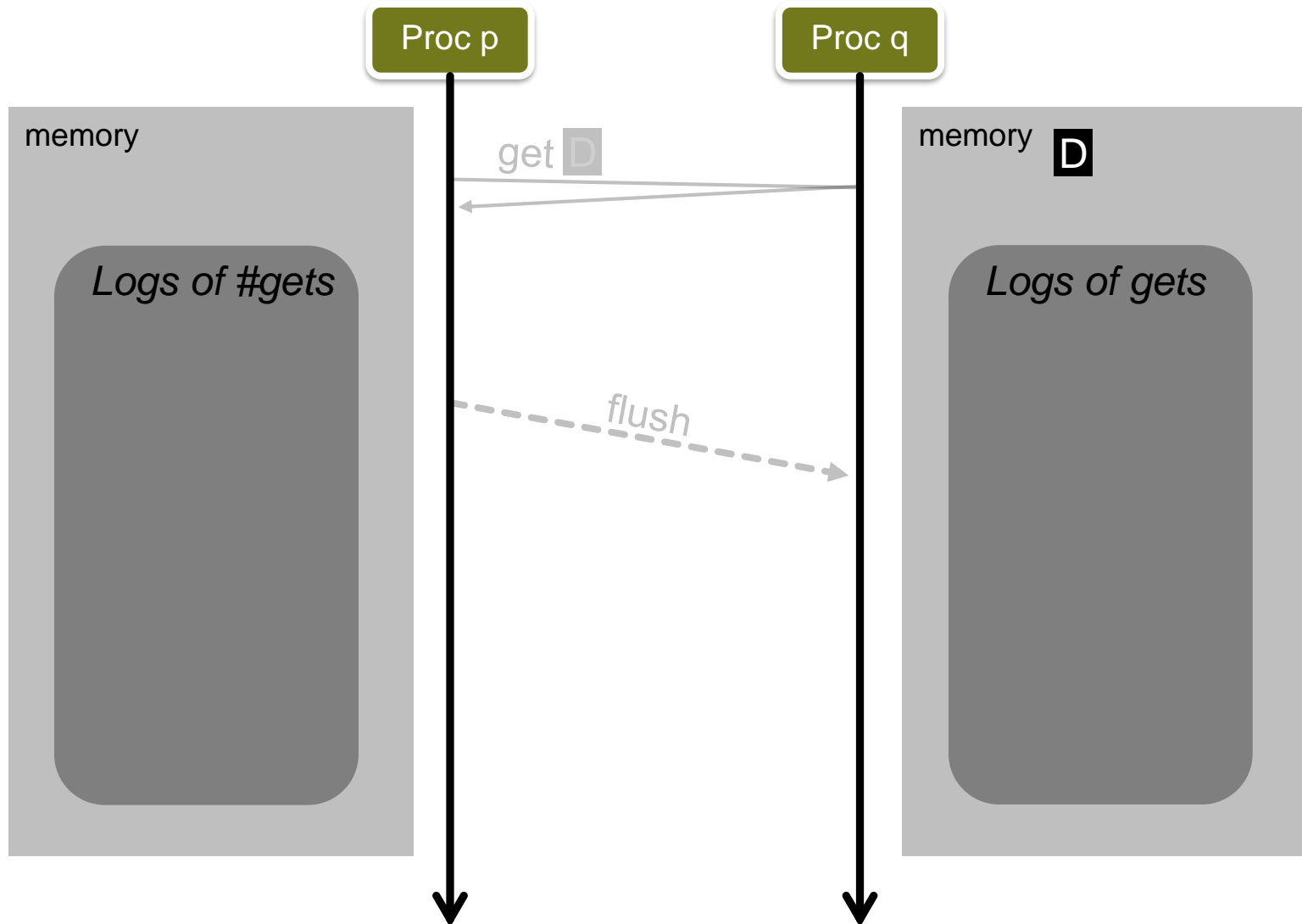
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


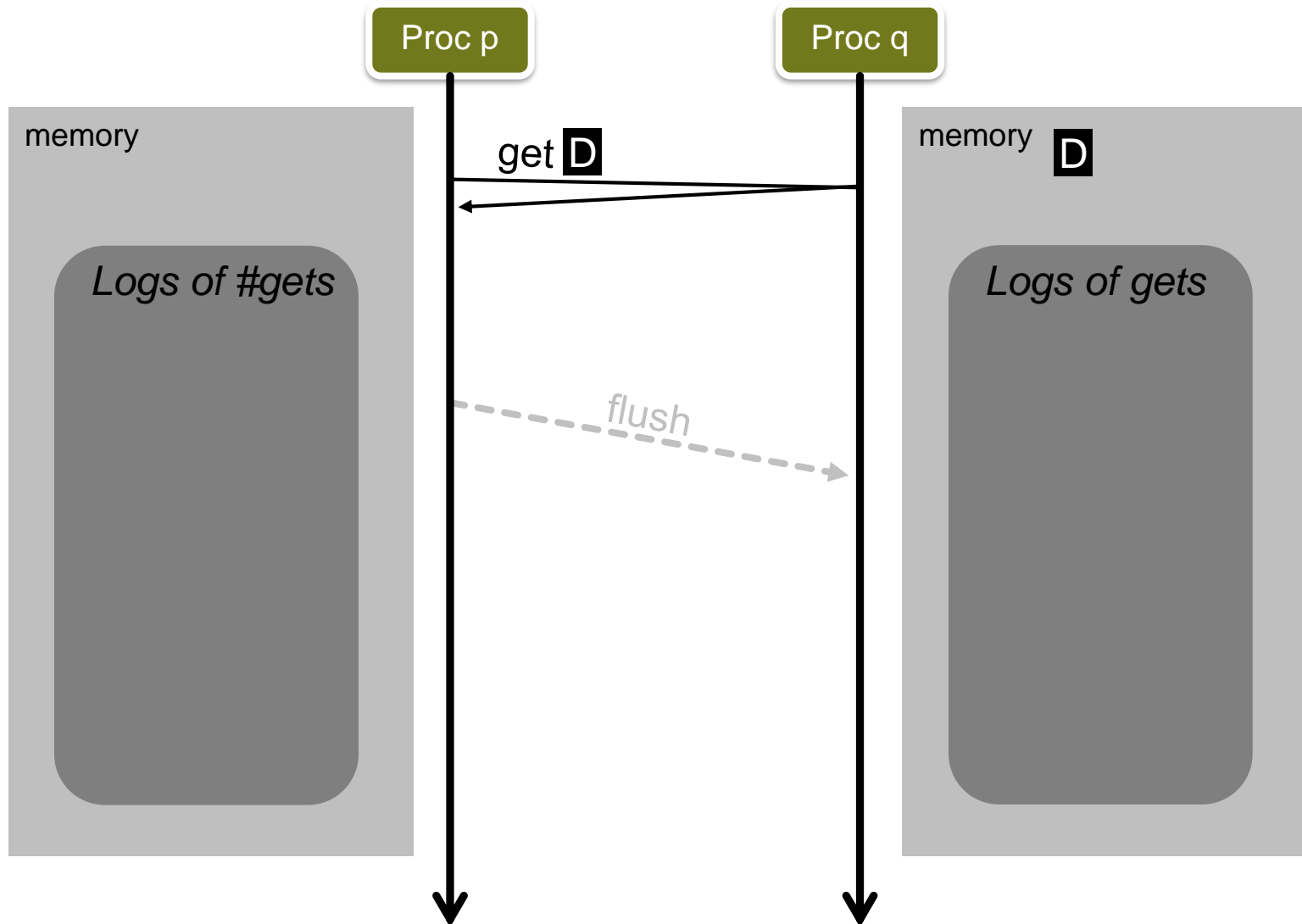
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


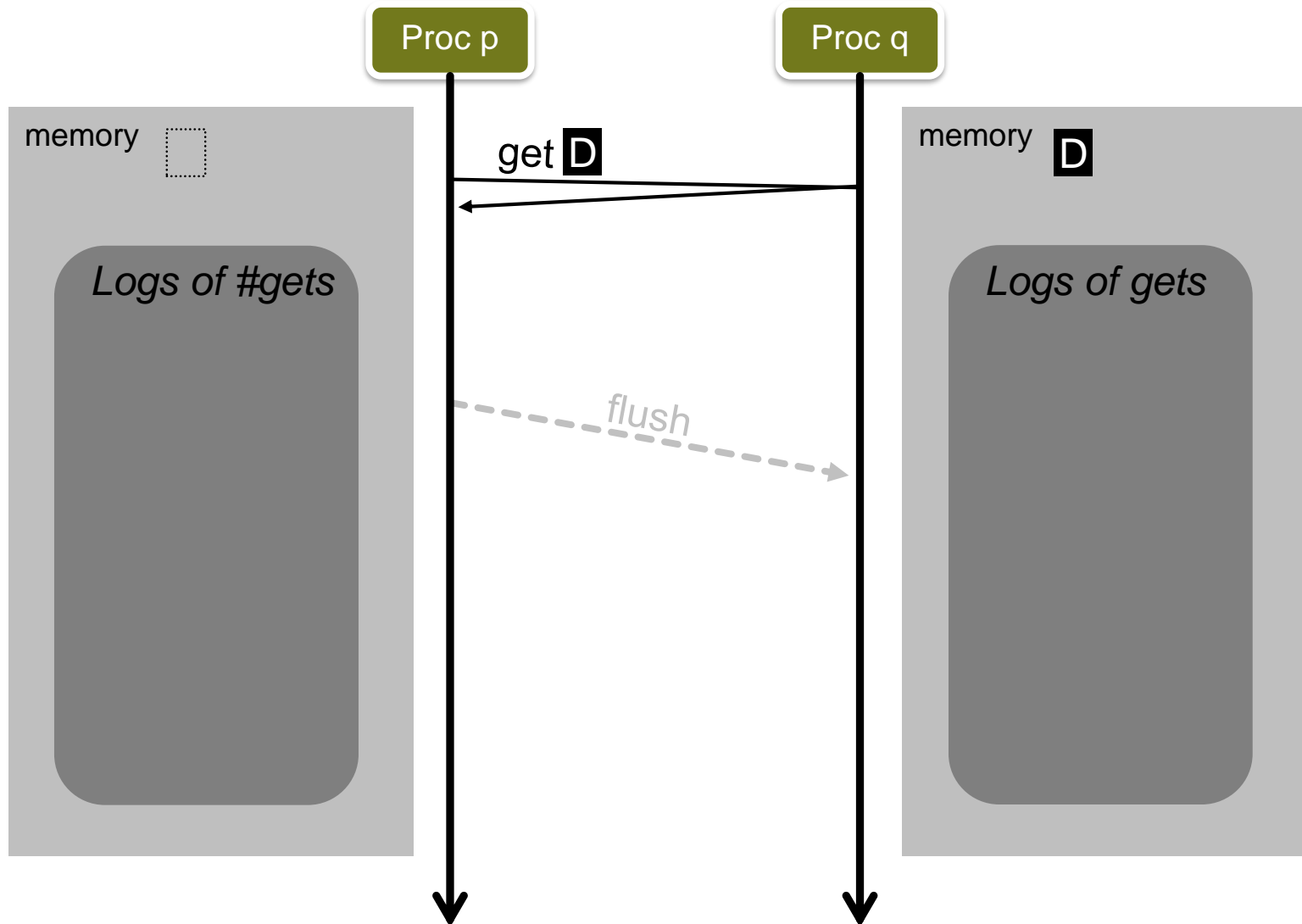
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS

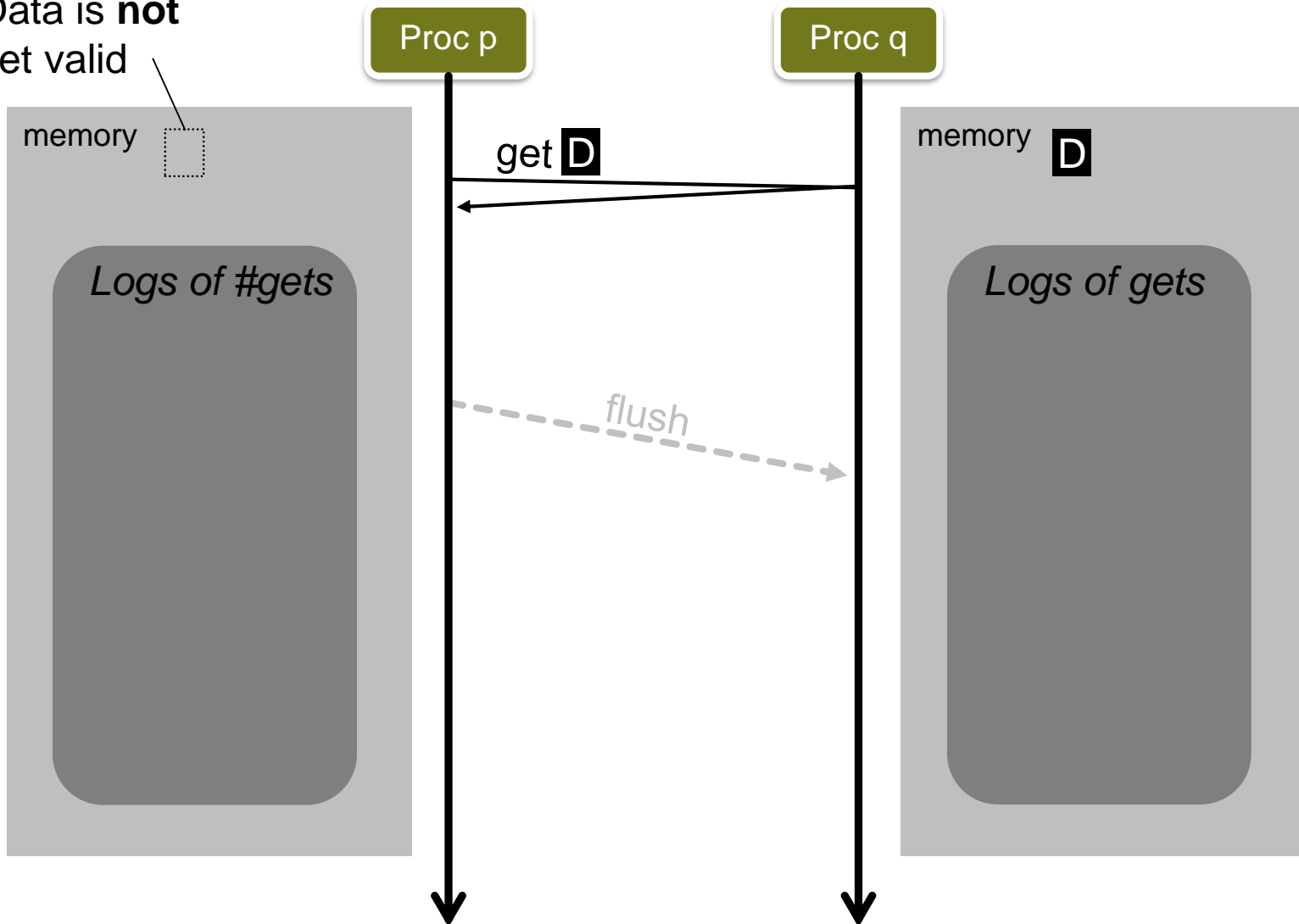
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

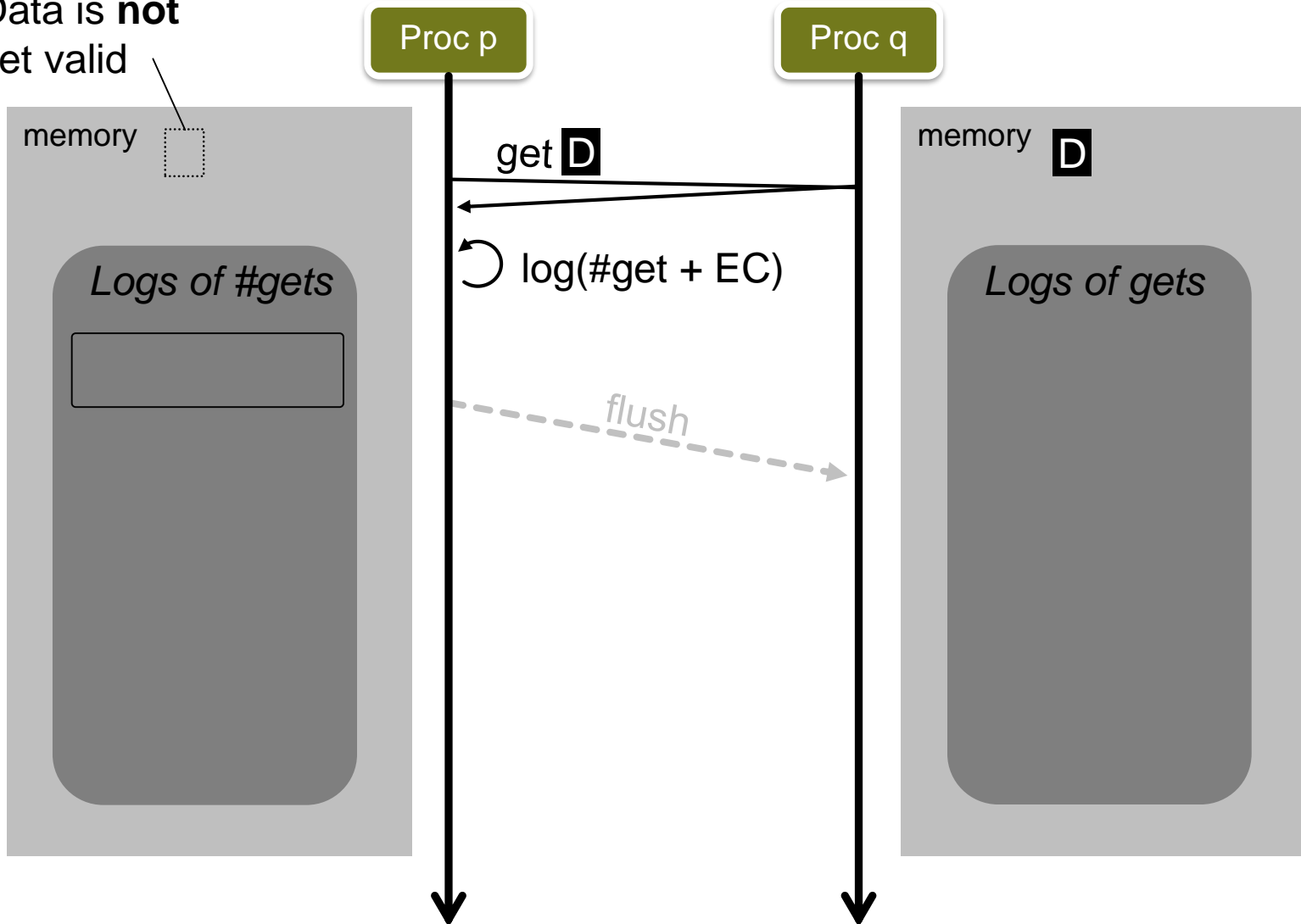


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

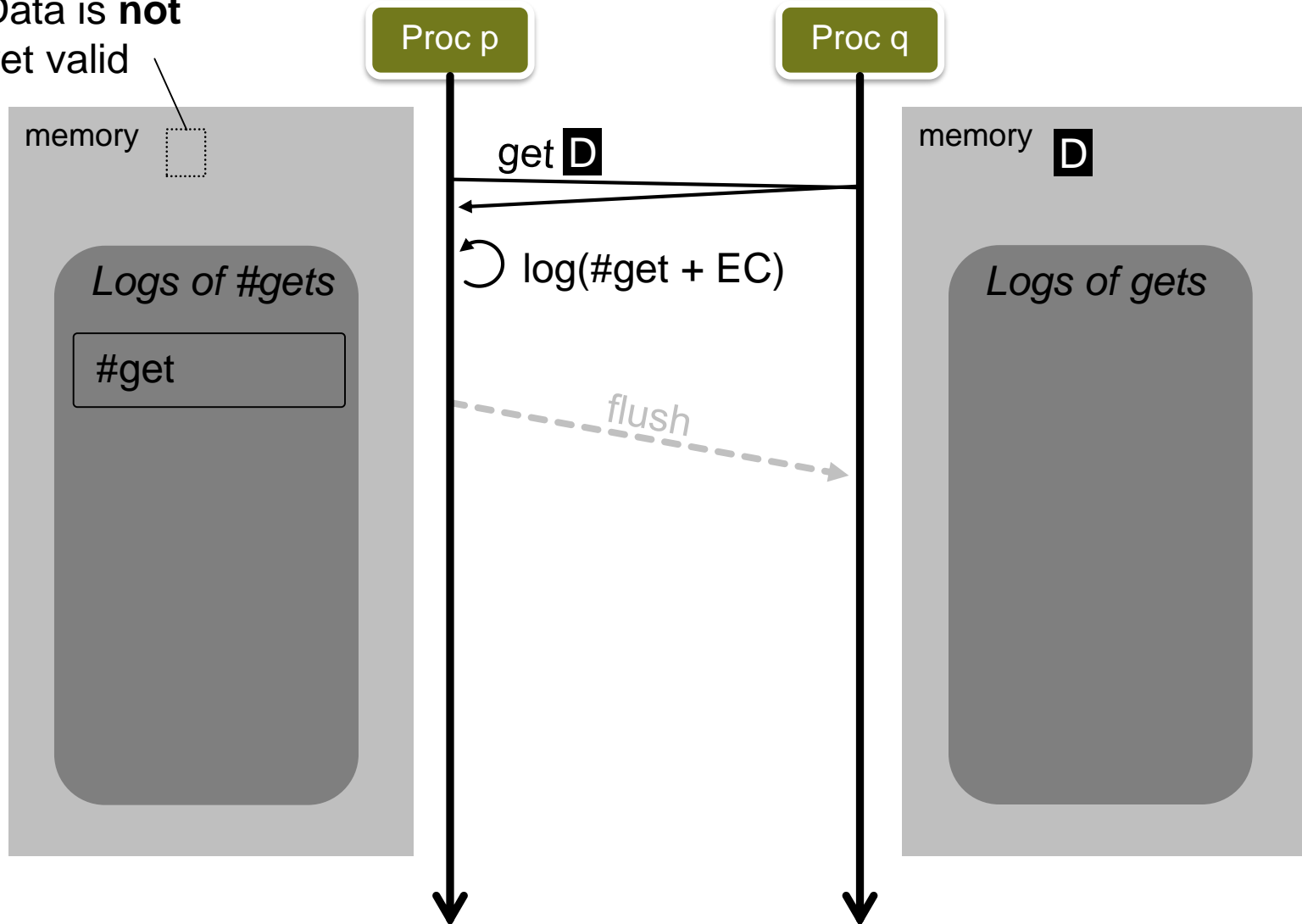


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

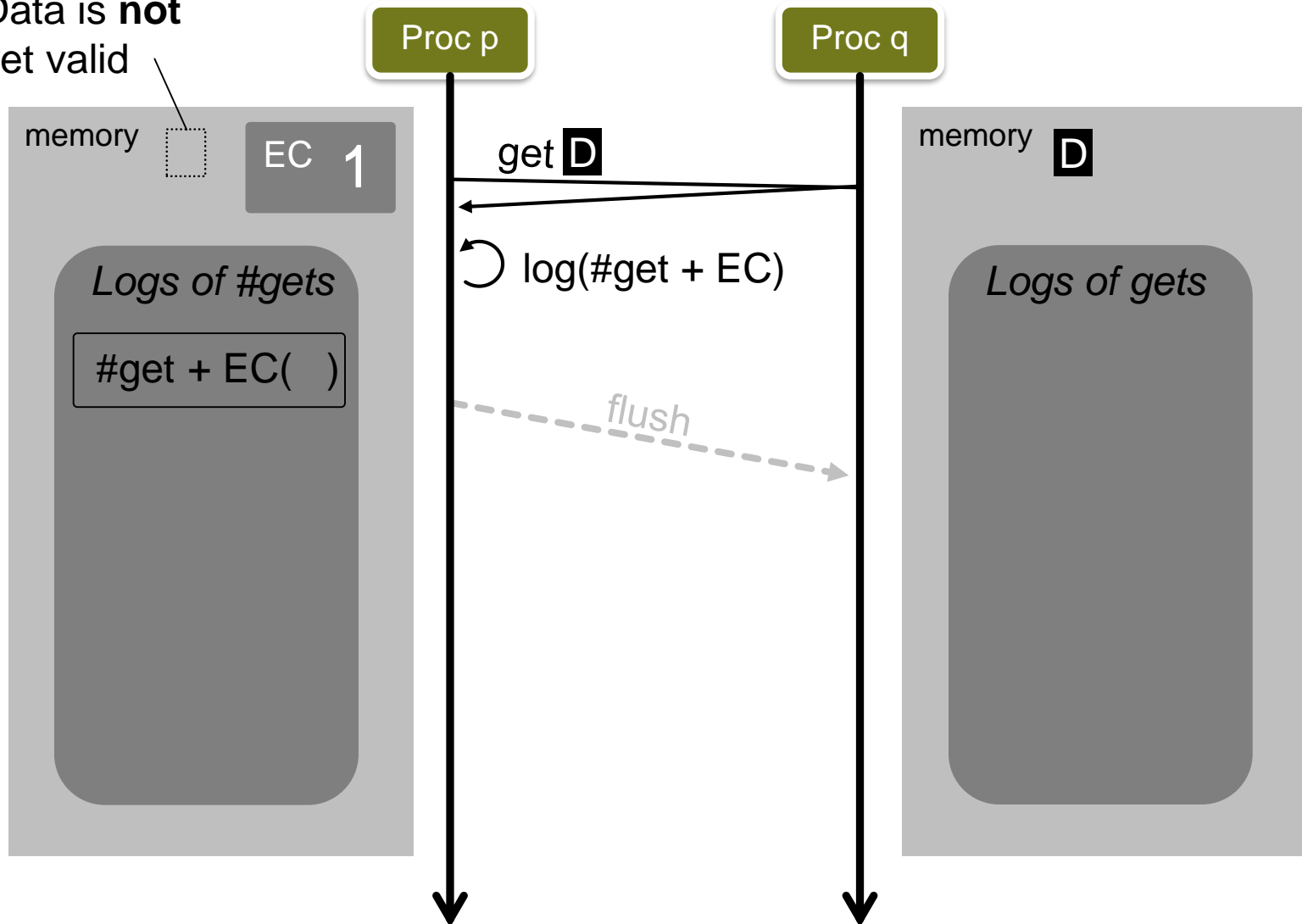


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

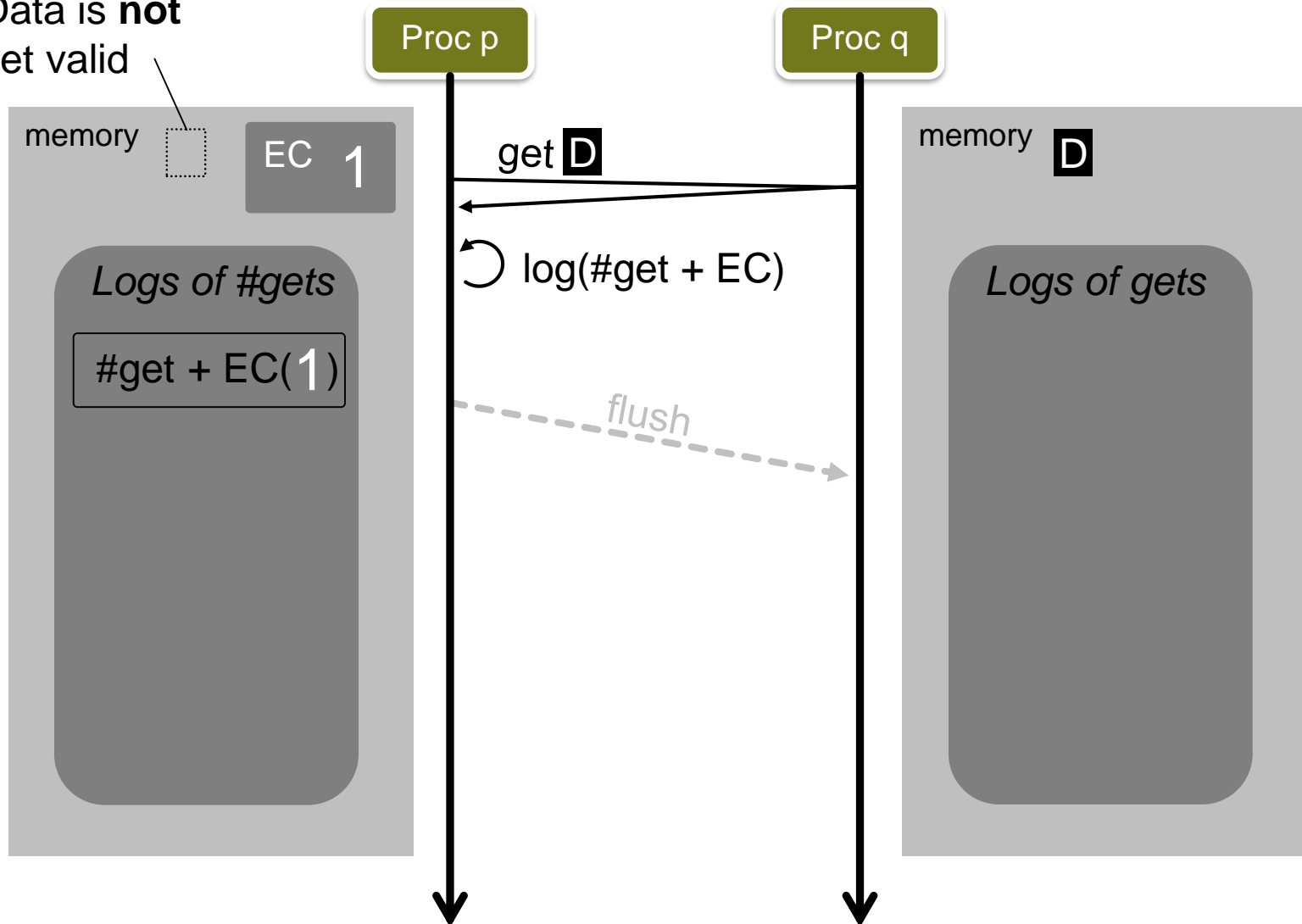


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

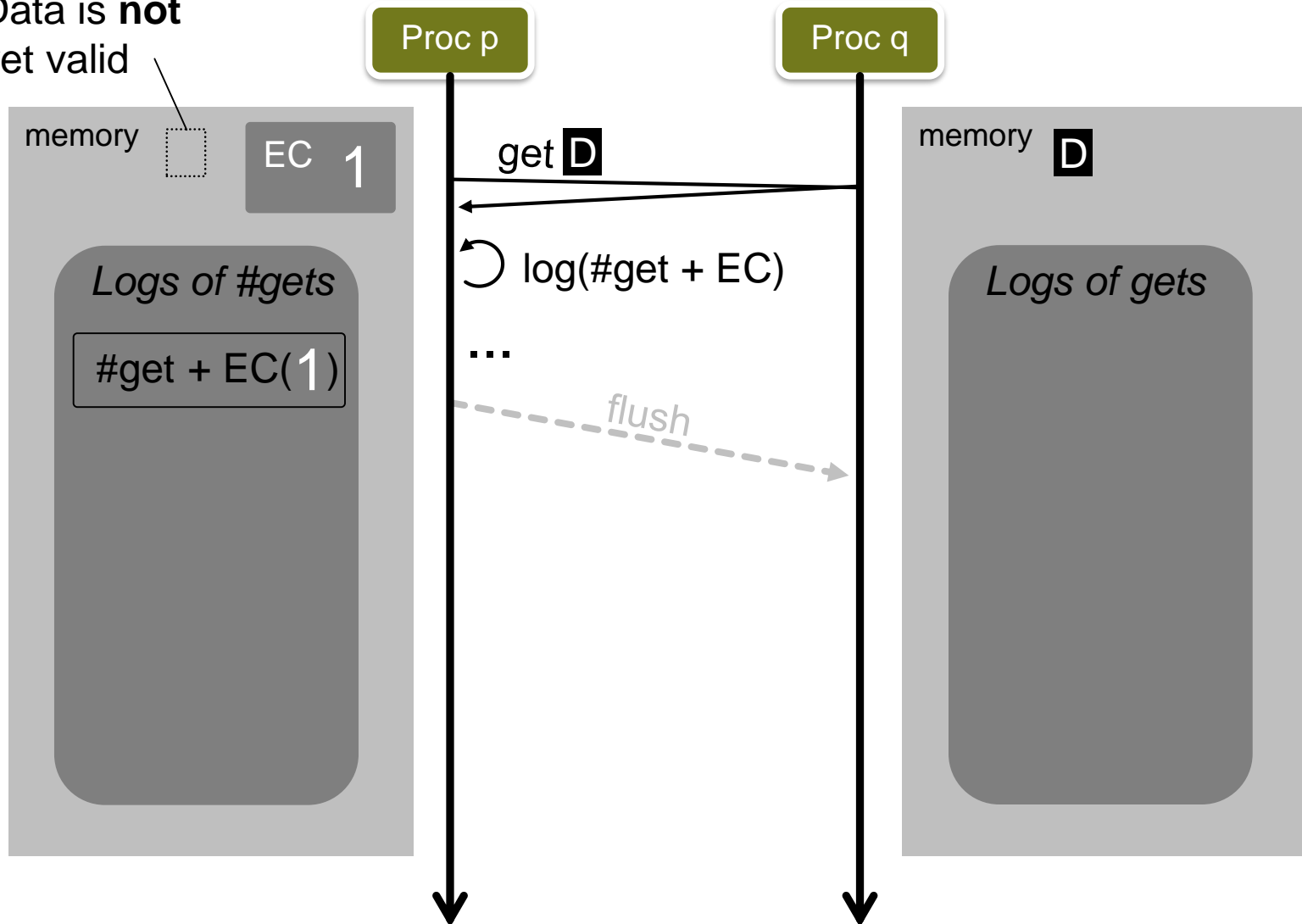


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

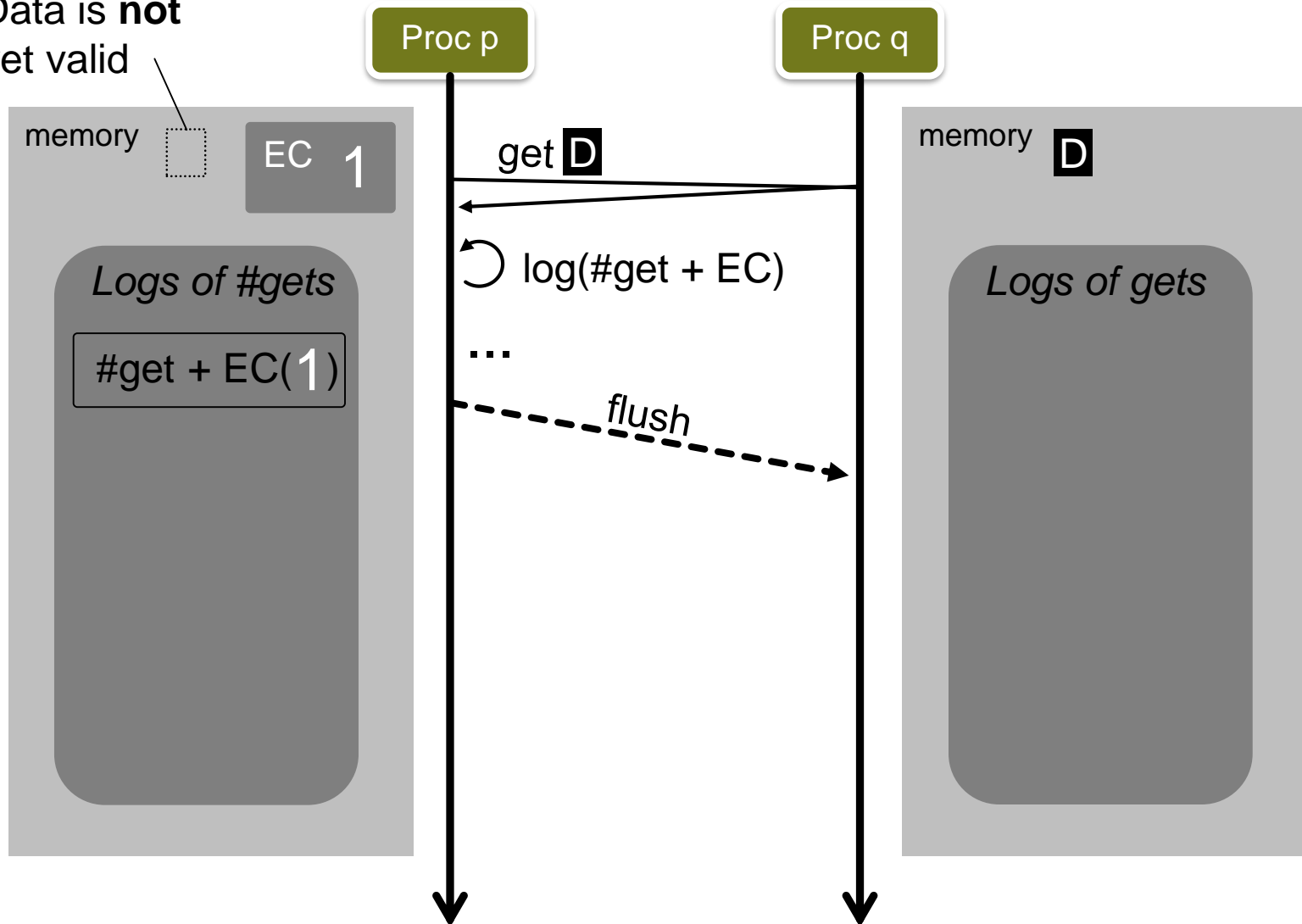


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

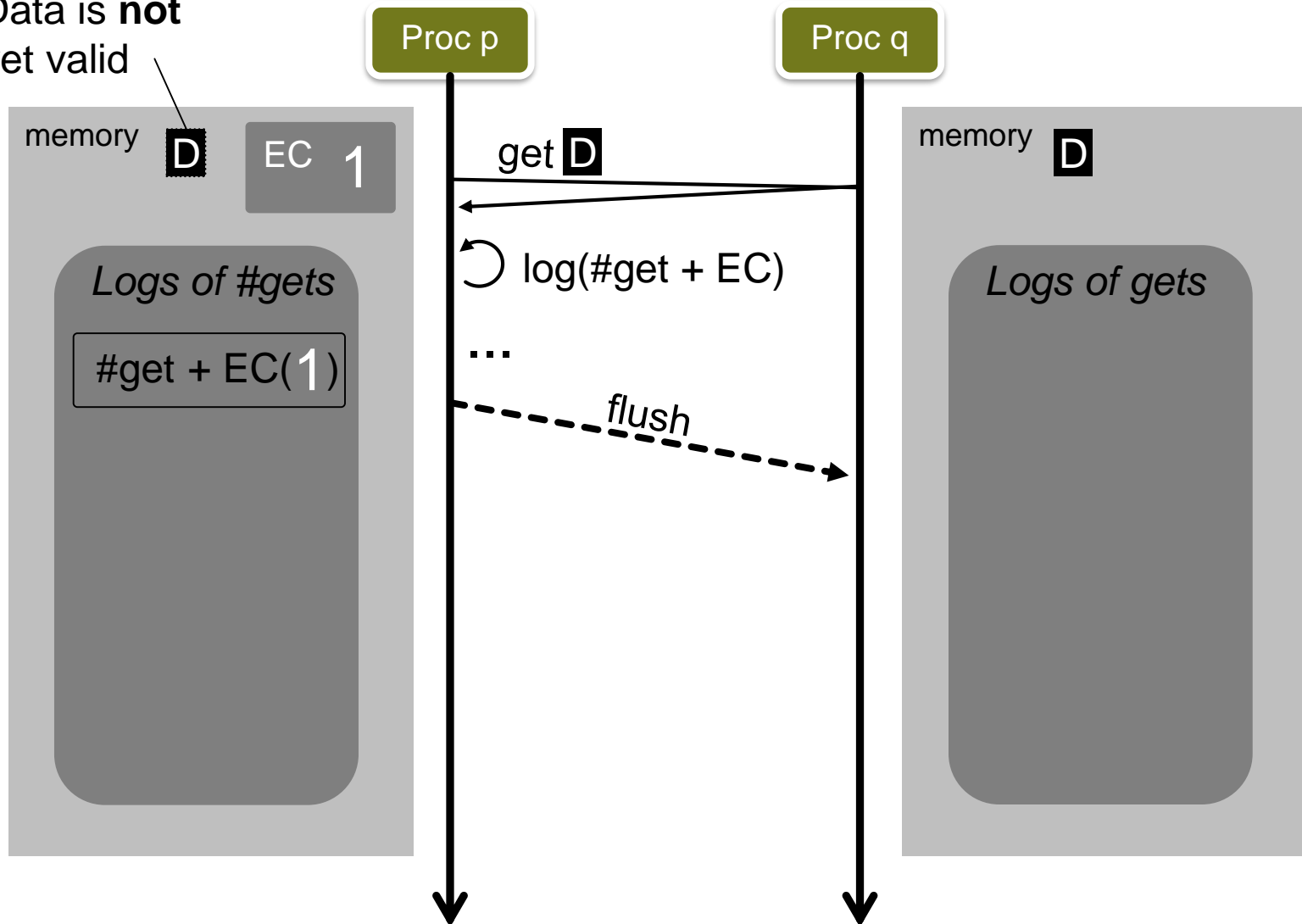


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is **not**
yet valid

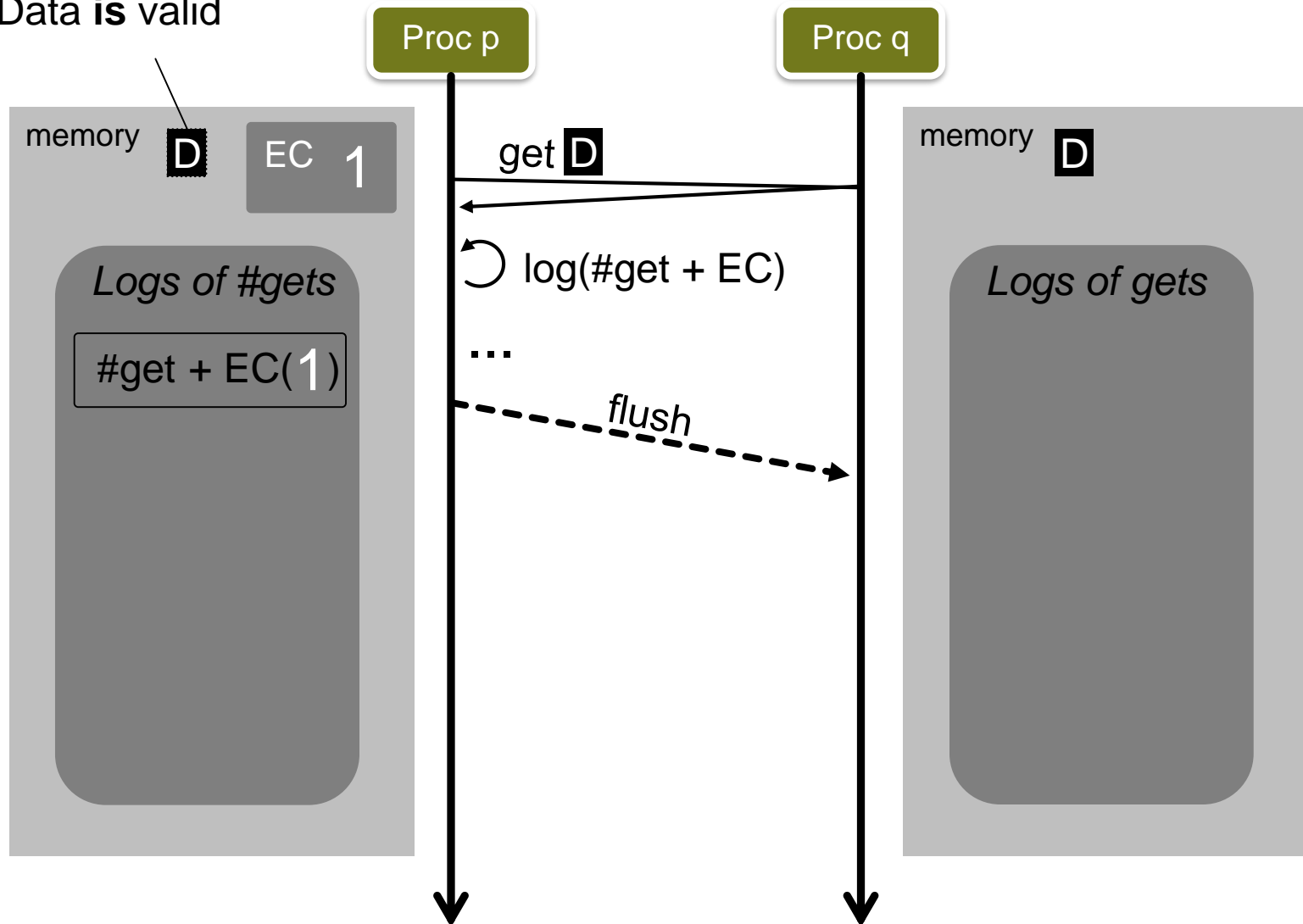


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

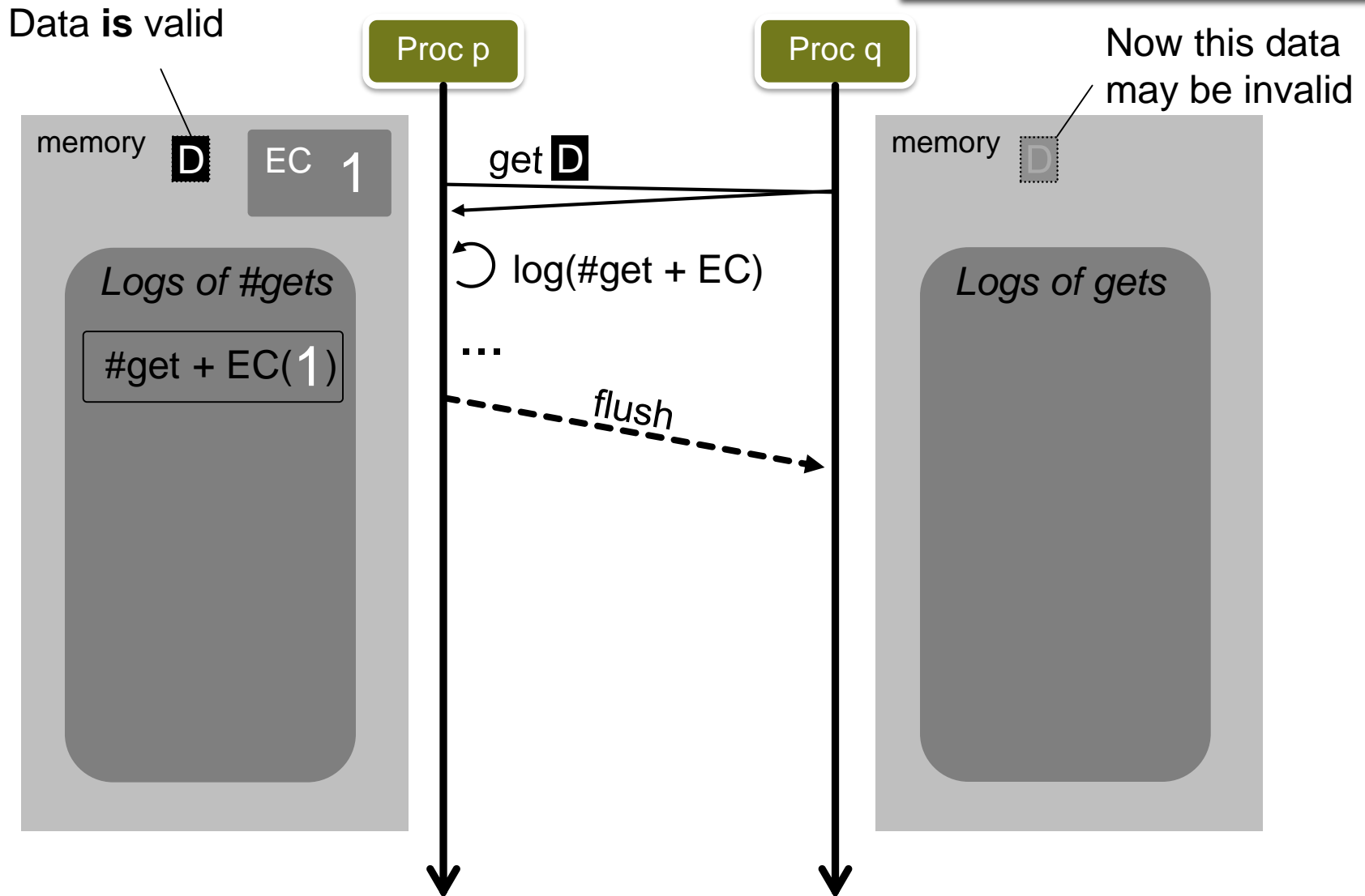
$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



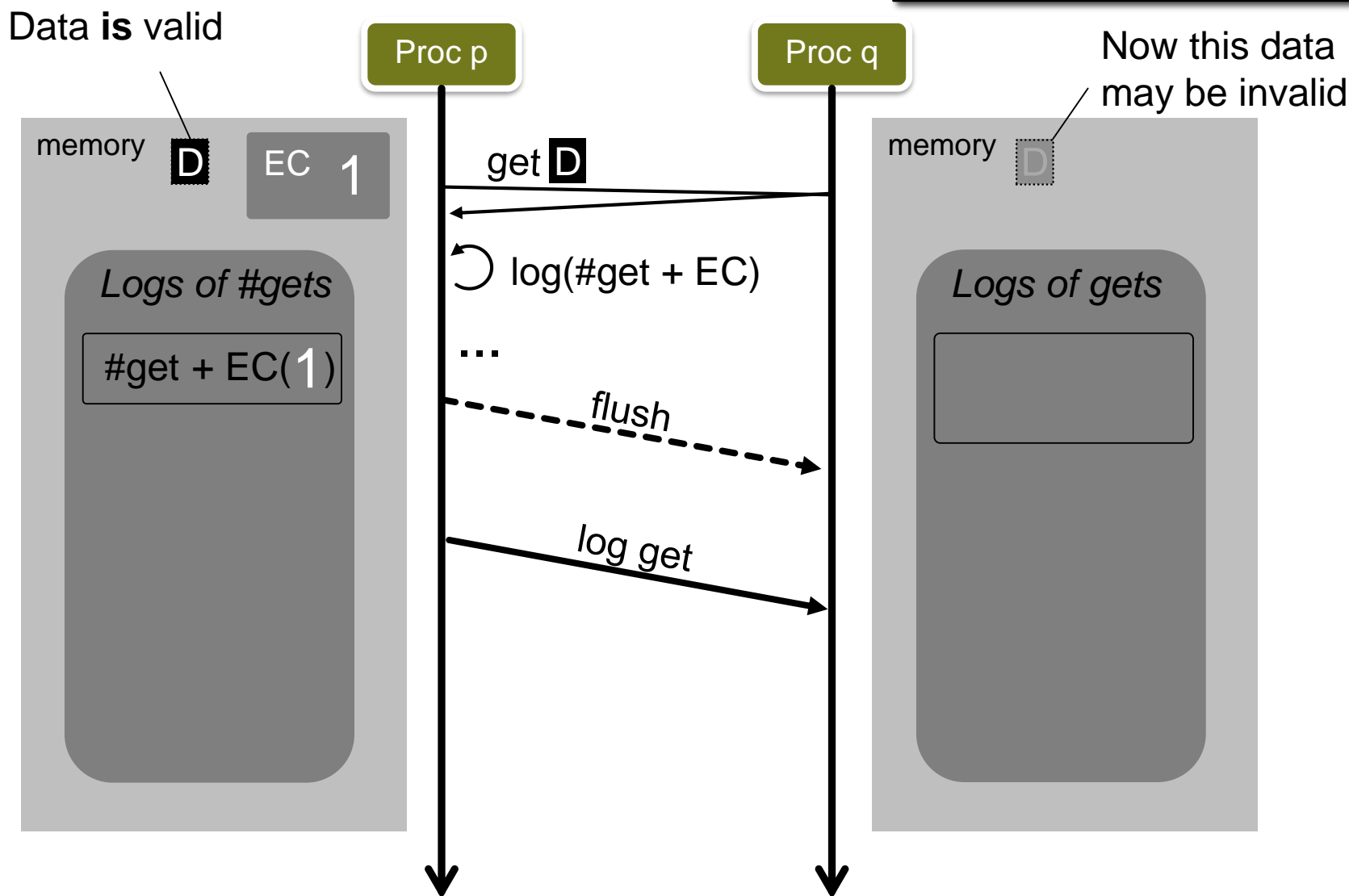
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$


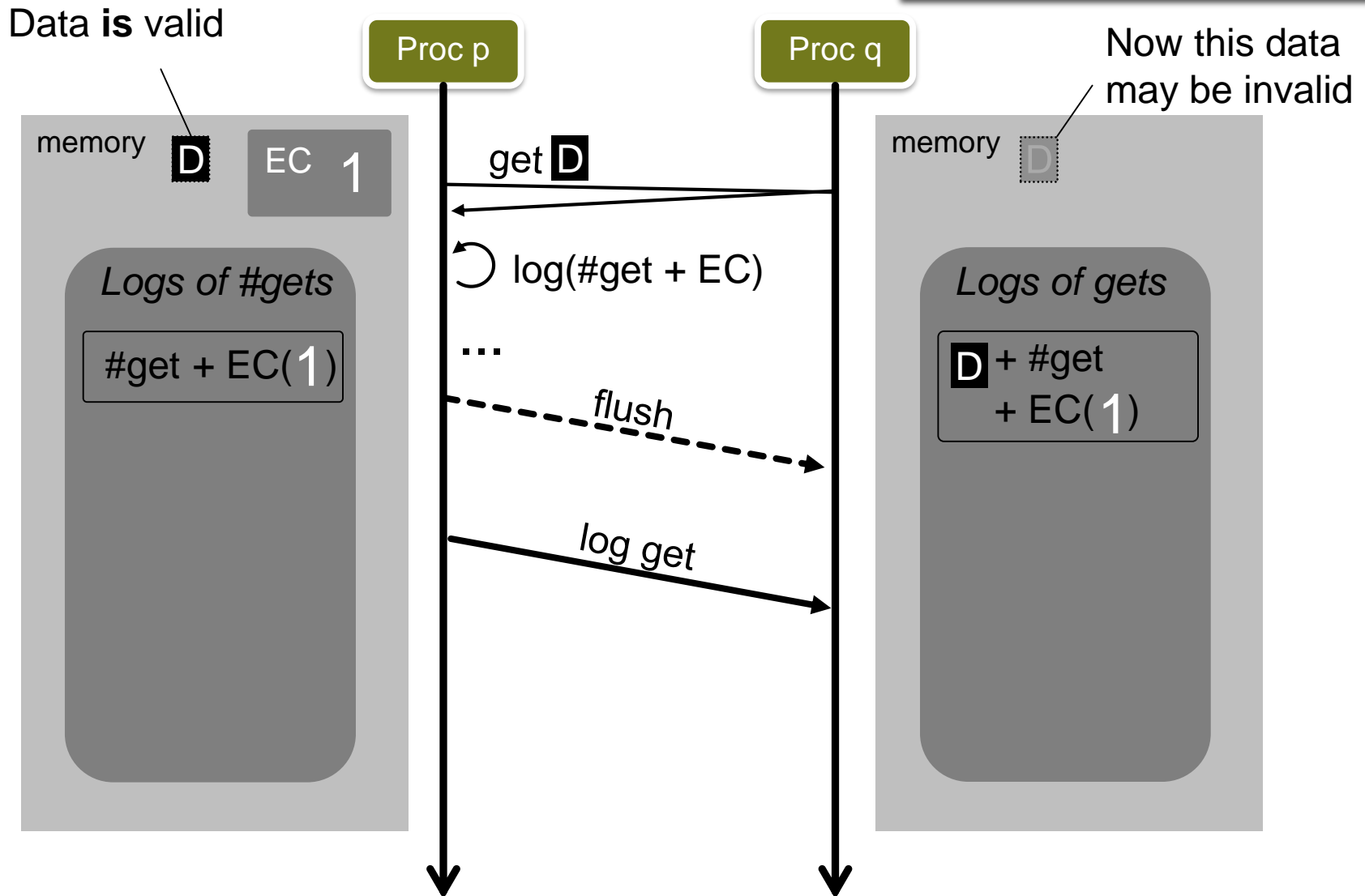
RMA: LOGGING GETS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



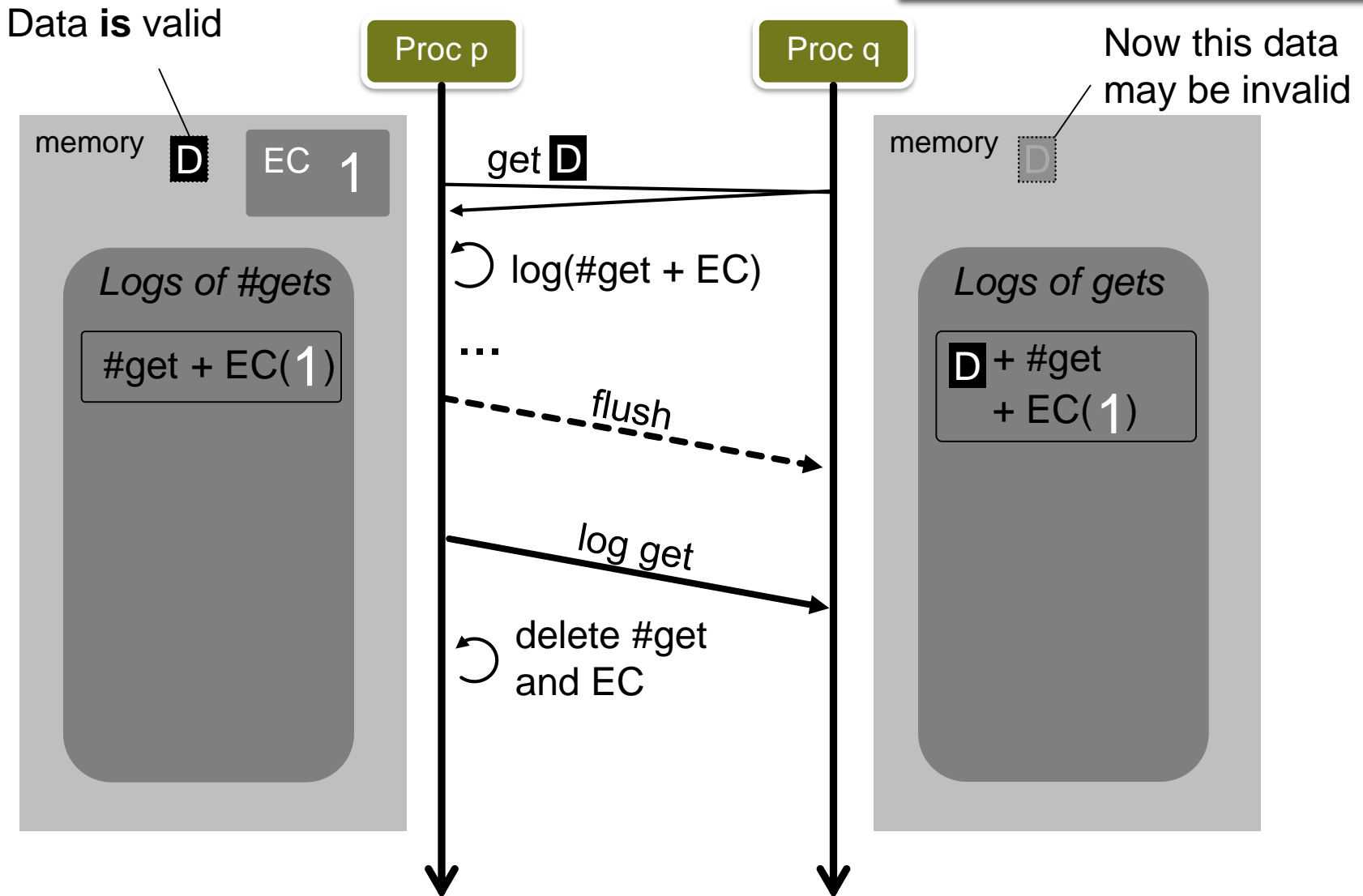
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$


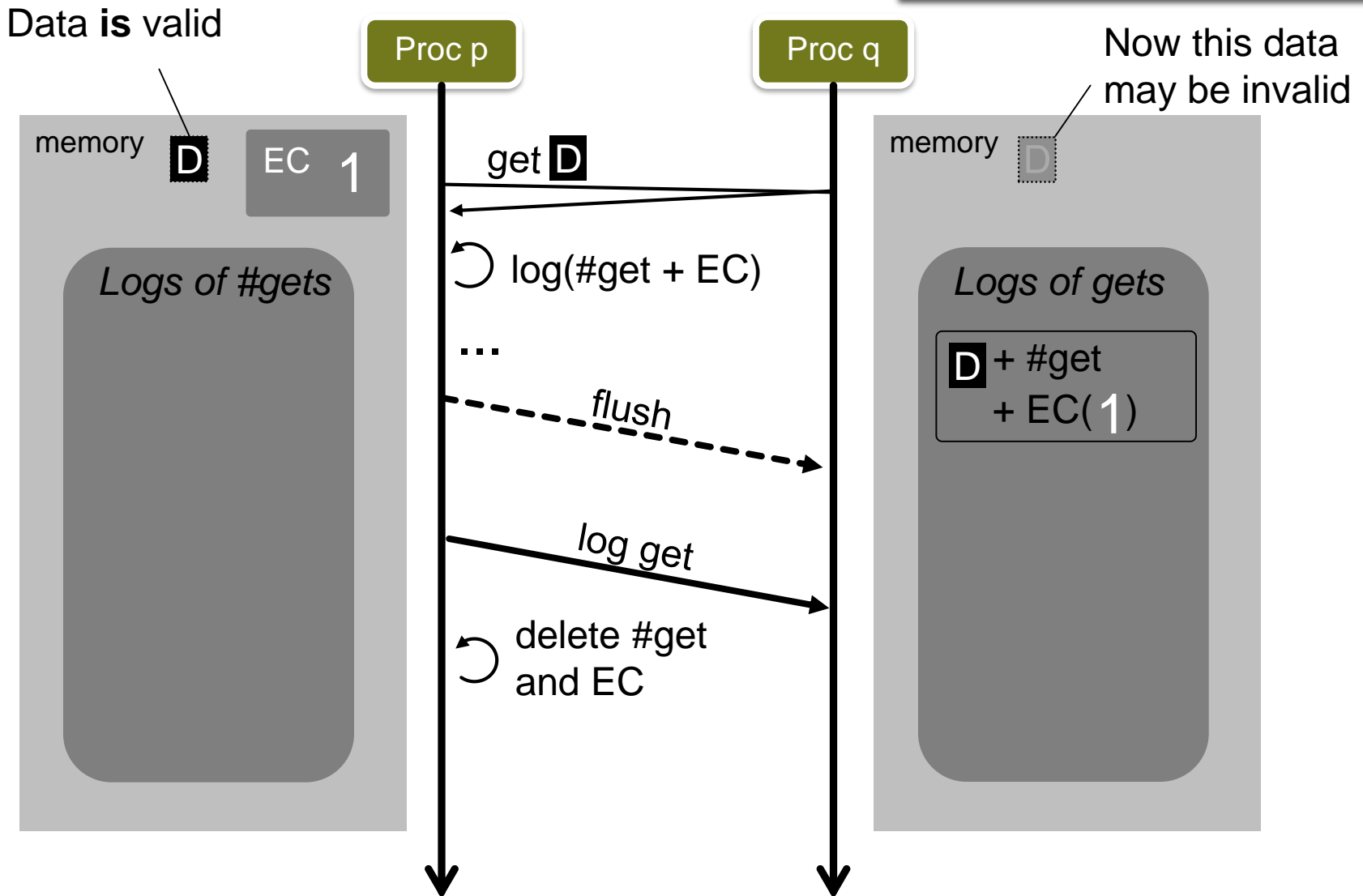
RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

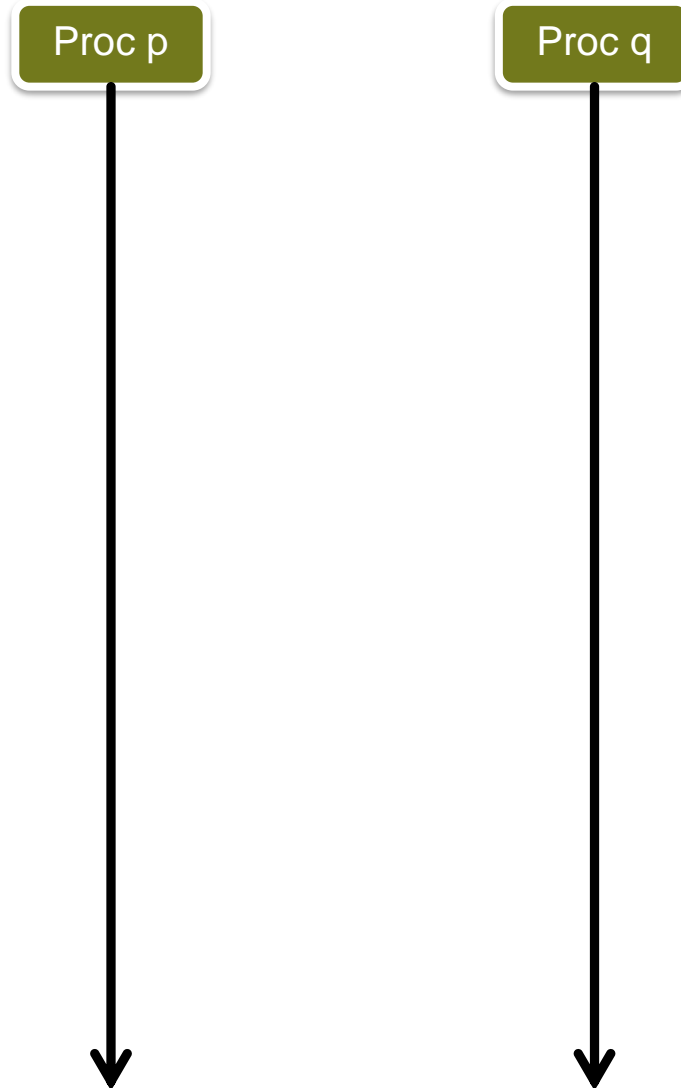
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING GETS



OVERVIEW OF OUR RESEARCH

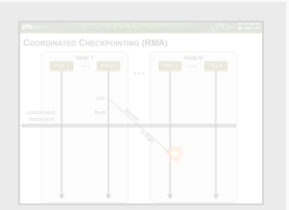
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

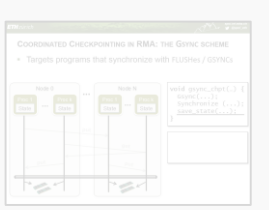
PUT($p \rightrightarrows q$)
GET($p \leftrightharpoons q$)



CC in RMA

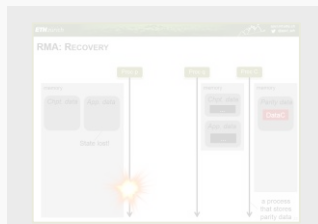


MP vs. RMA



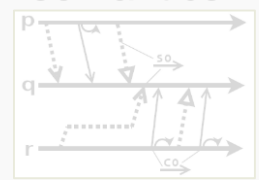
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recoverithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery

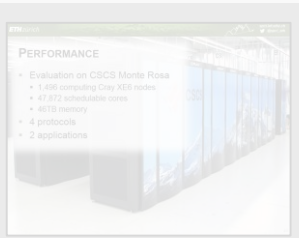
Holistic fault-tolerance library



Design

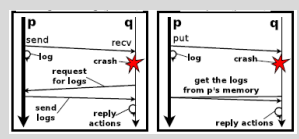
Checkpoints on demand

Optimizations

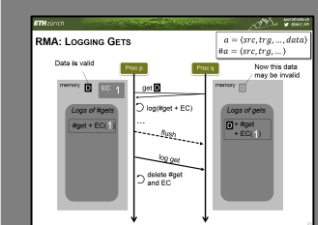


Performance

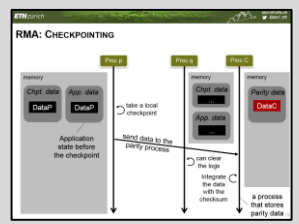
UC in RMA



MP vs. RMA



Logging accesses

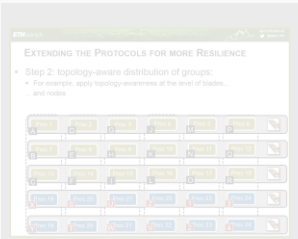


Checkpointing schemes

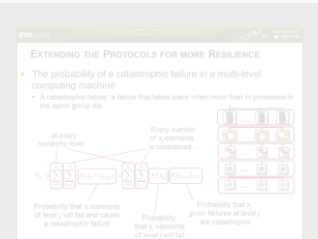
Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

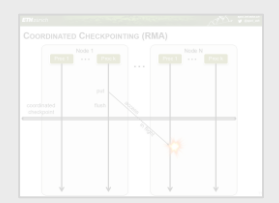
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

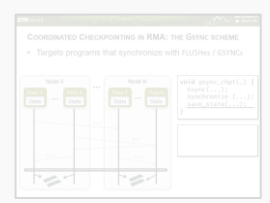
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

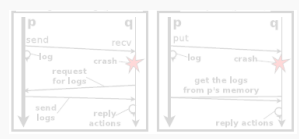


MP vs. RMA

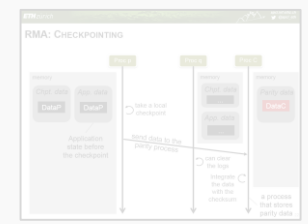


Schemes

UC in RMA

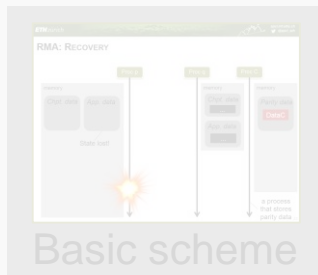


MP vs. RMA



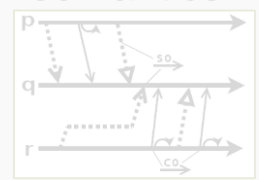
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

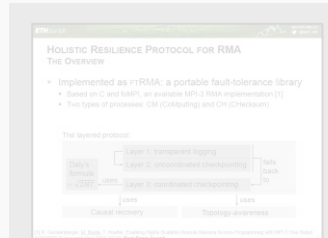
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

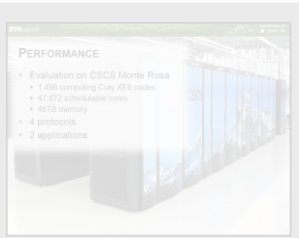
Holistic fault-tolerance library



Design

Checkpoints on demand

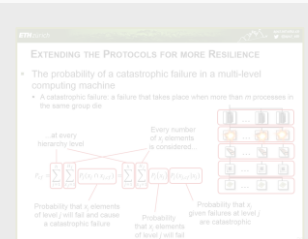
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

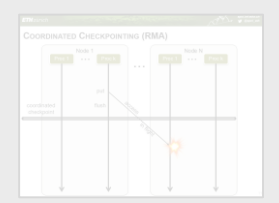
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

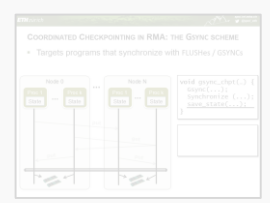
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

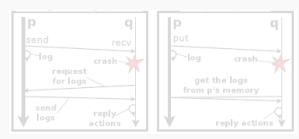


MP vs. RMA

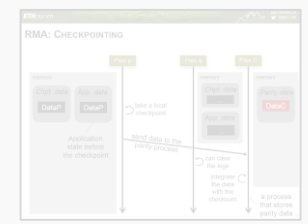


Schemes

UC in RMA

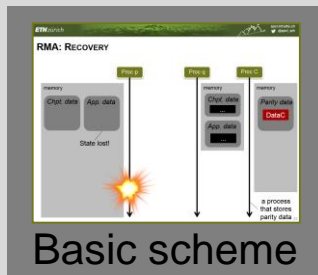


MP vs. RMA

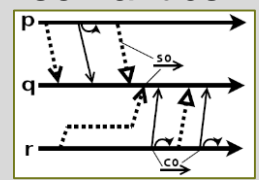


Checkpointing schemes

Recovery in RMA



Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{coh} order (referred to as the gsync order).*

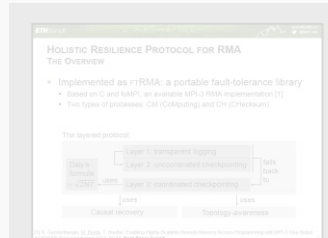
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

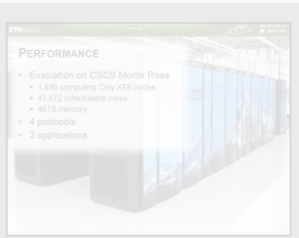
Holistic fault-tolerance library



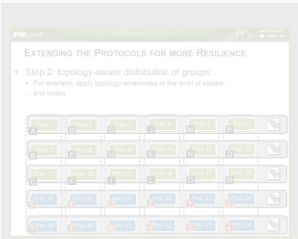
Design

Checkpoints on demand

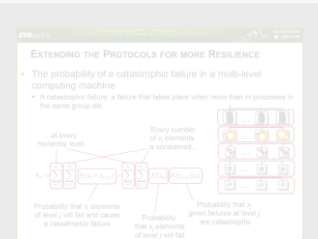
Optimizations



Performance

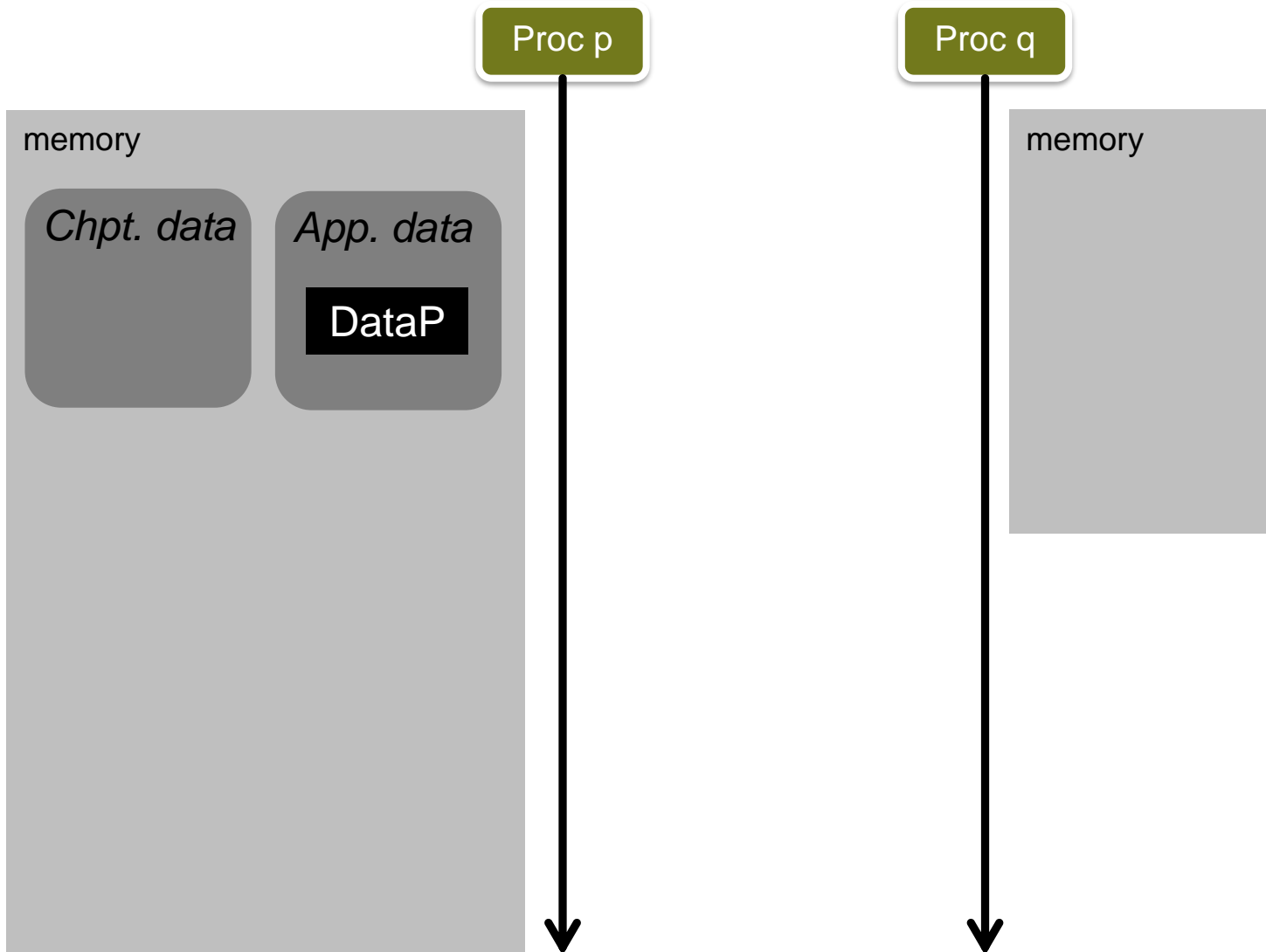


Distribution of processes

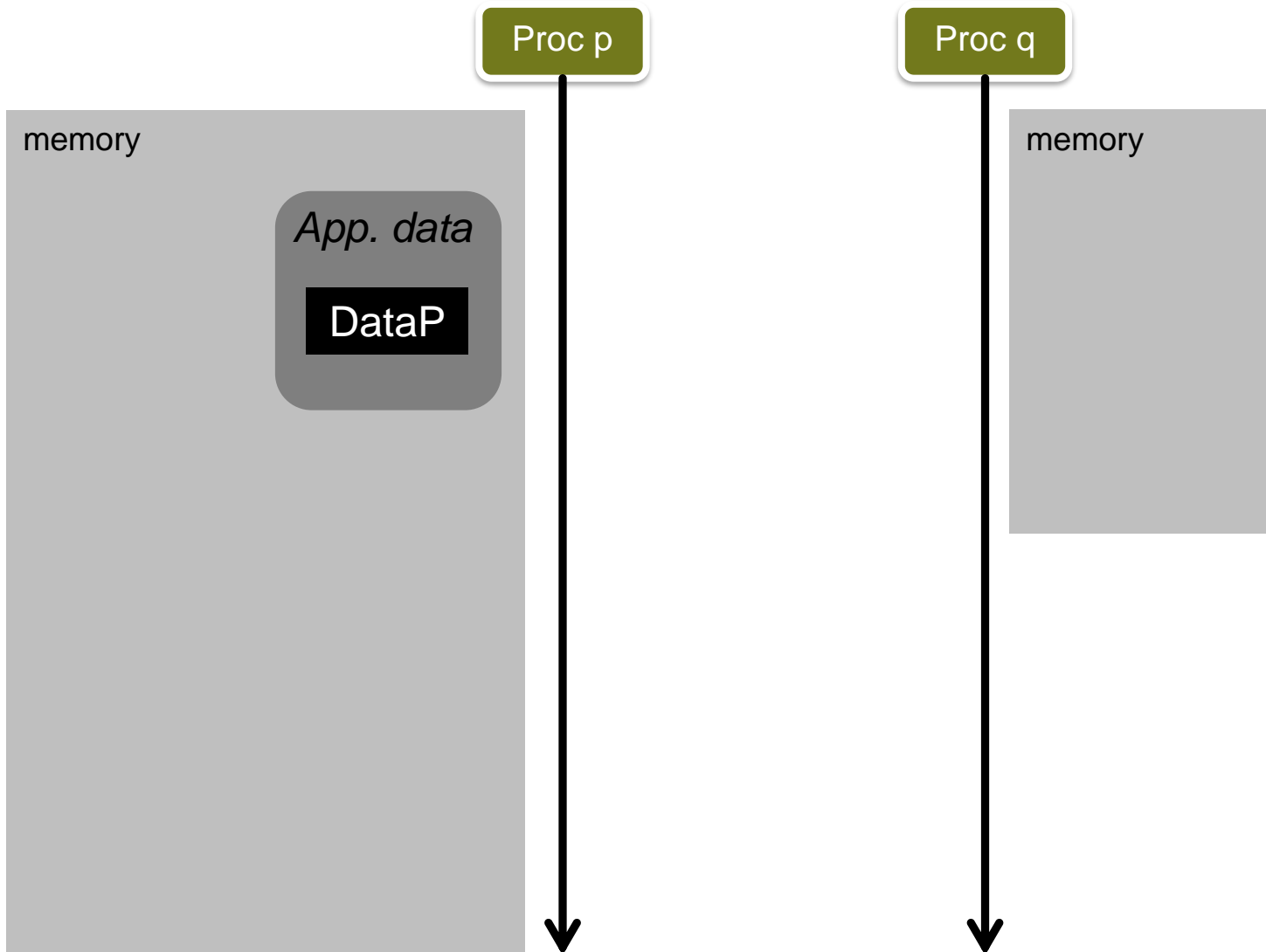


Decreasing failure prob.

RMA: RECOVERY

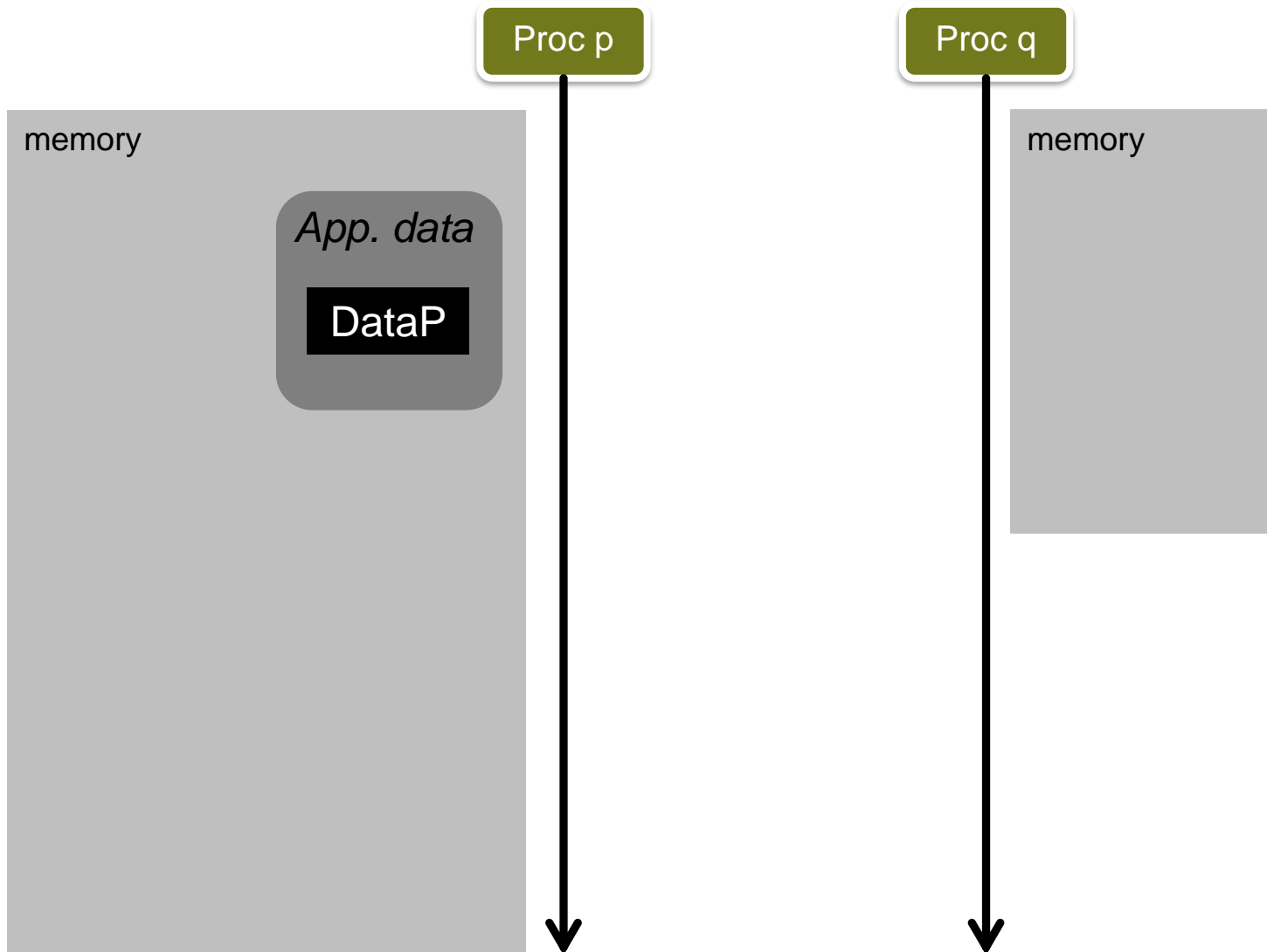


RMA: RECOVERY



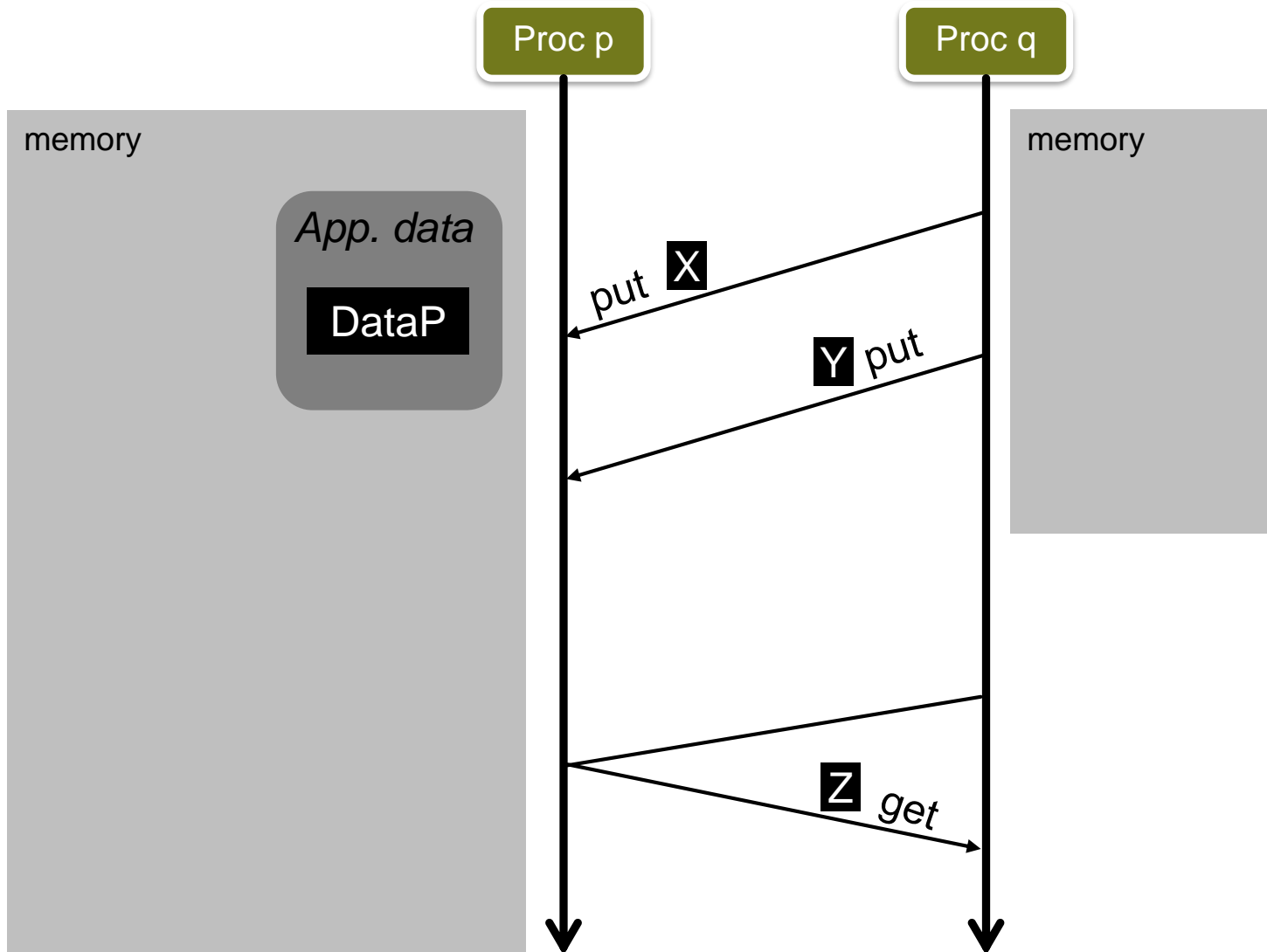
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



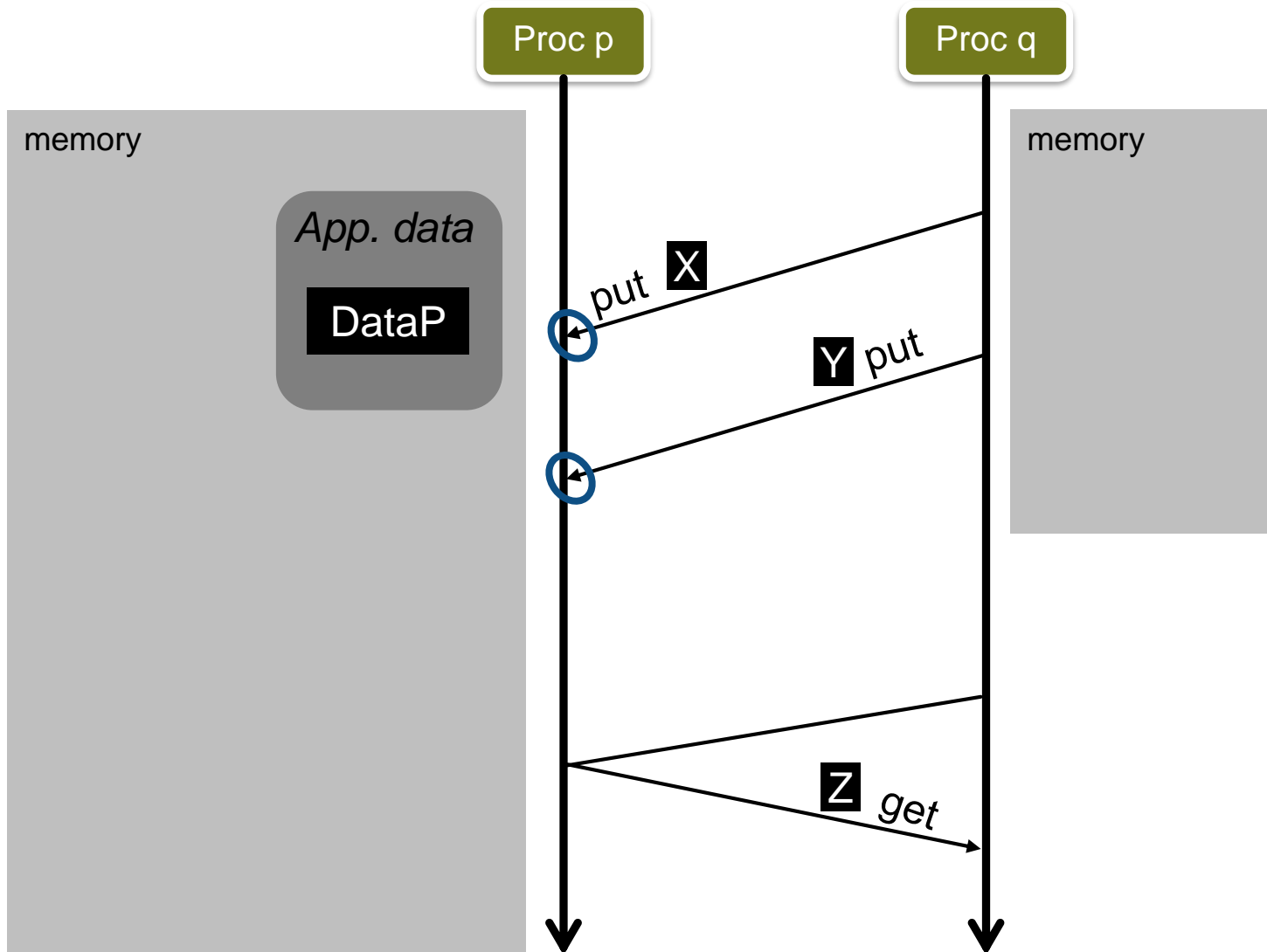
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



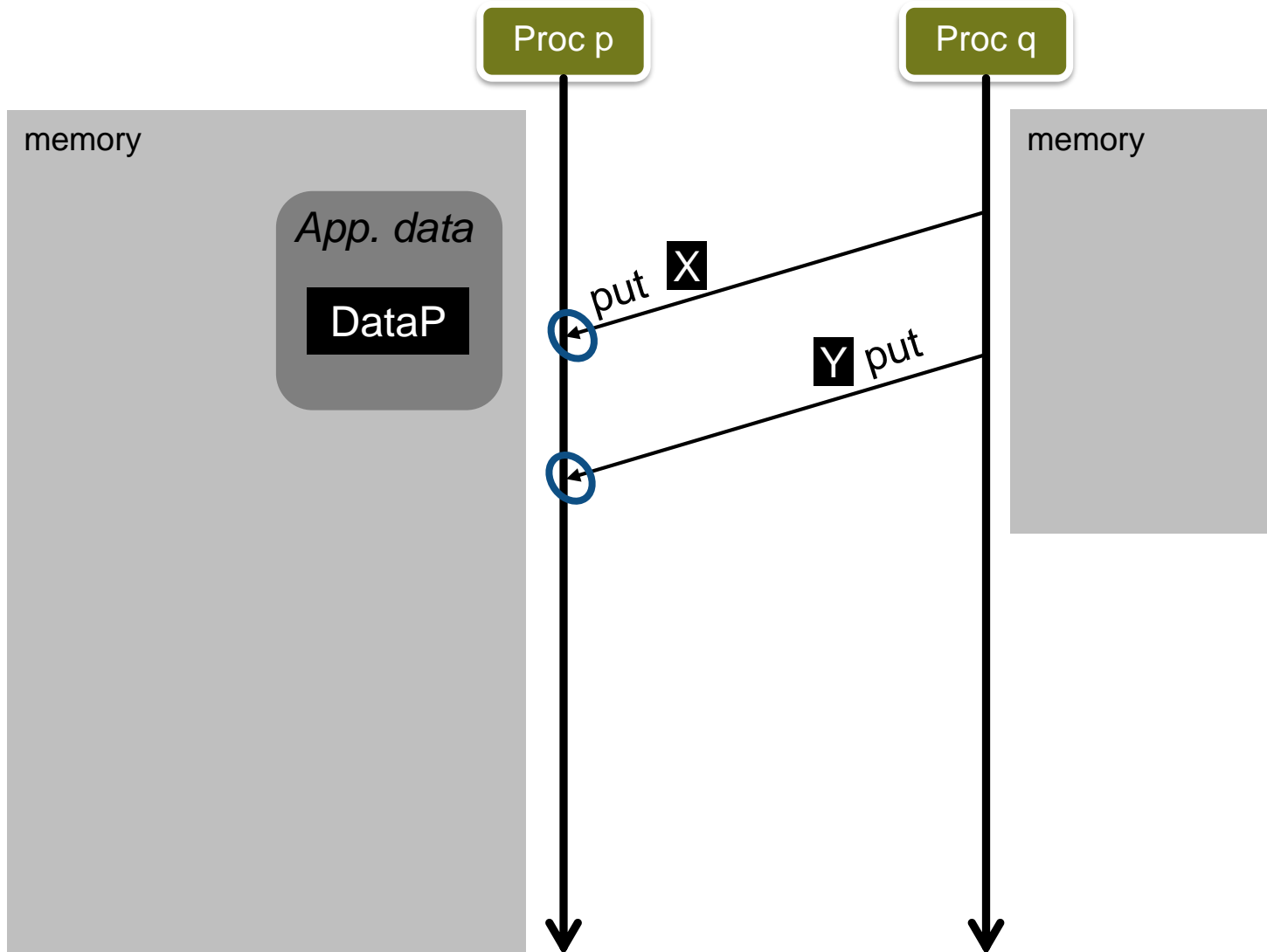
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



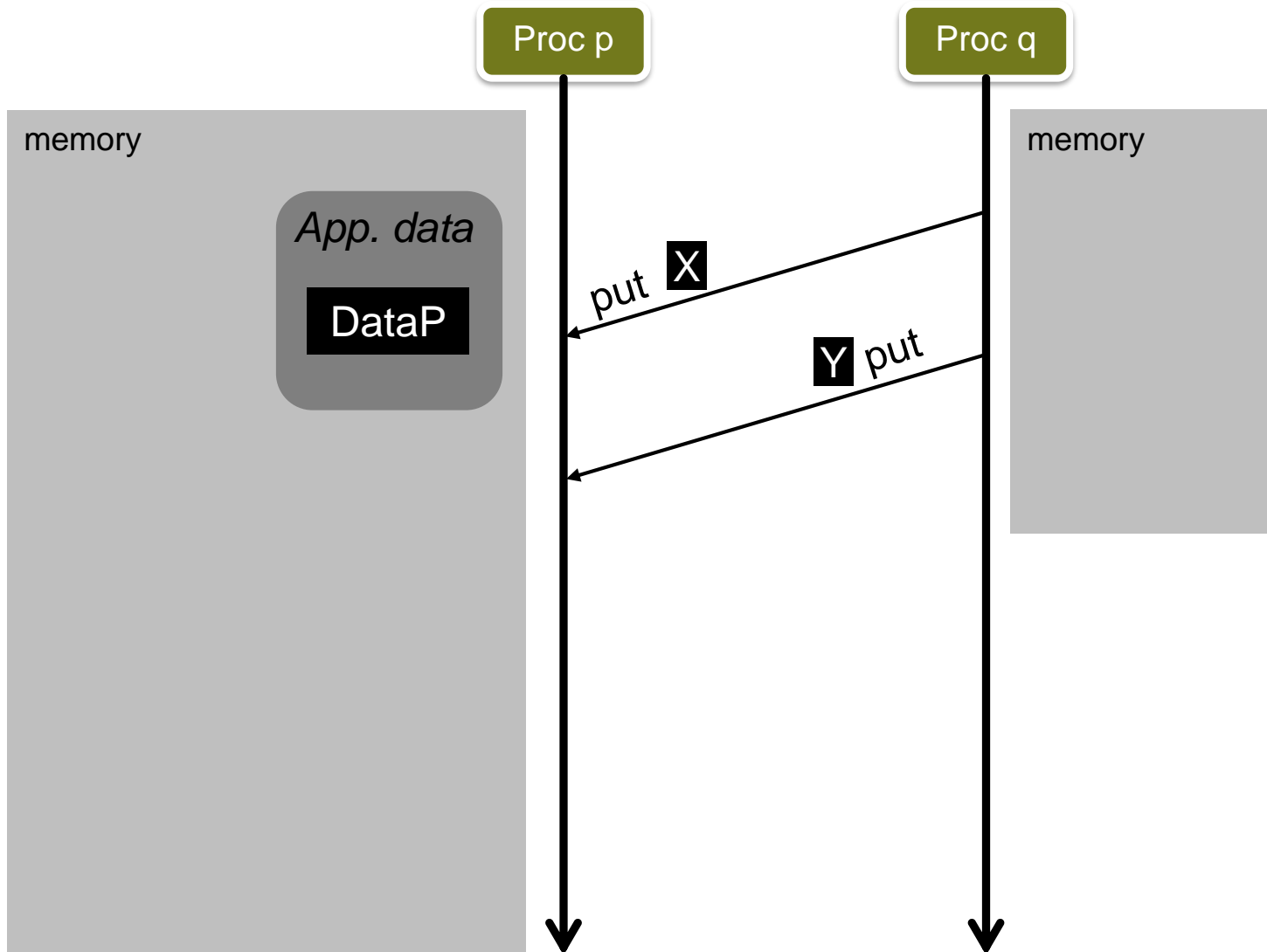
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



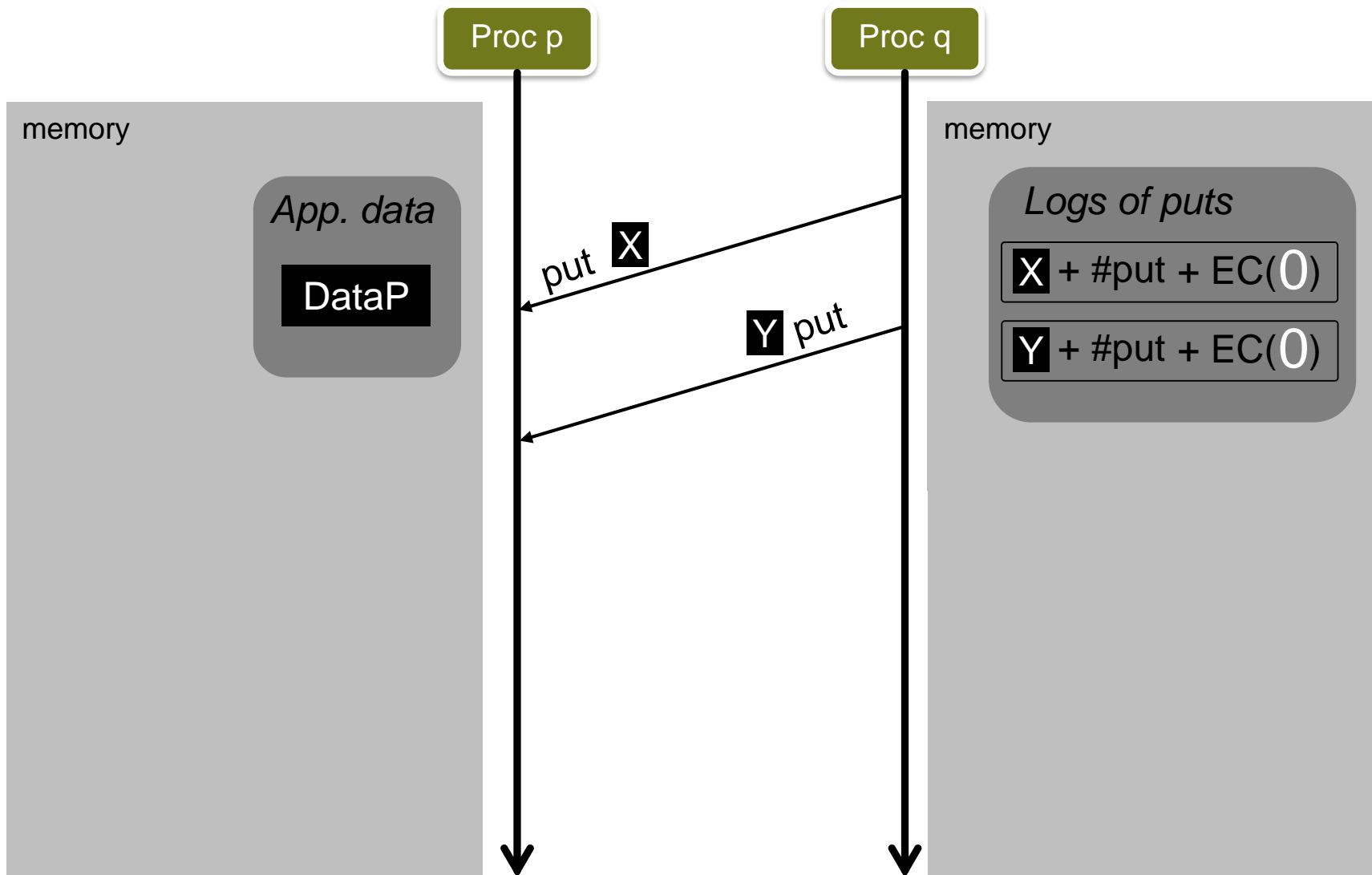
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



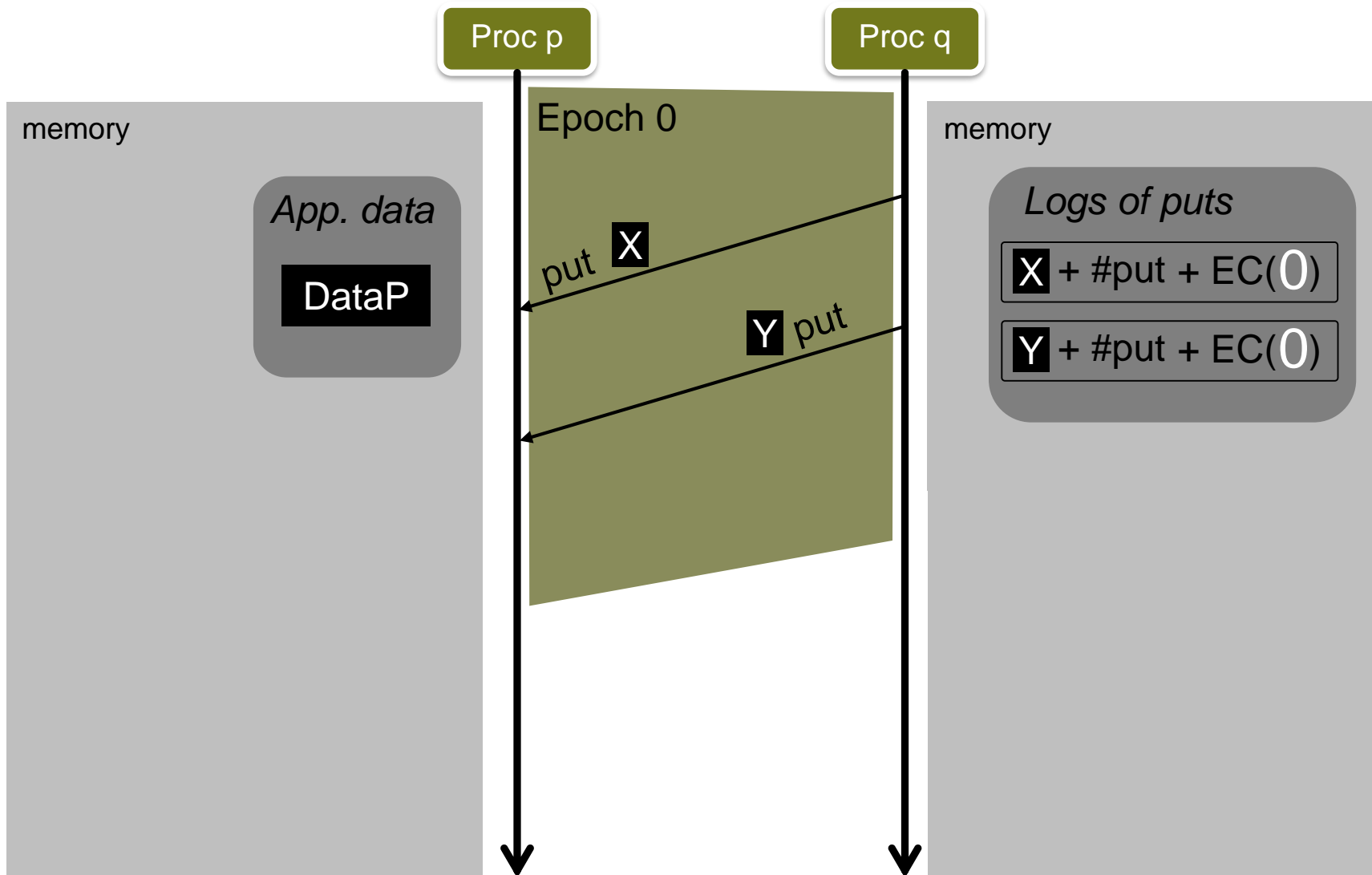
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



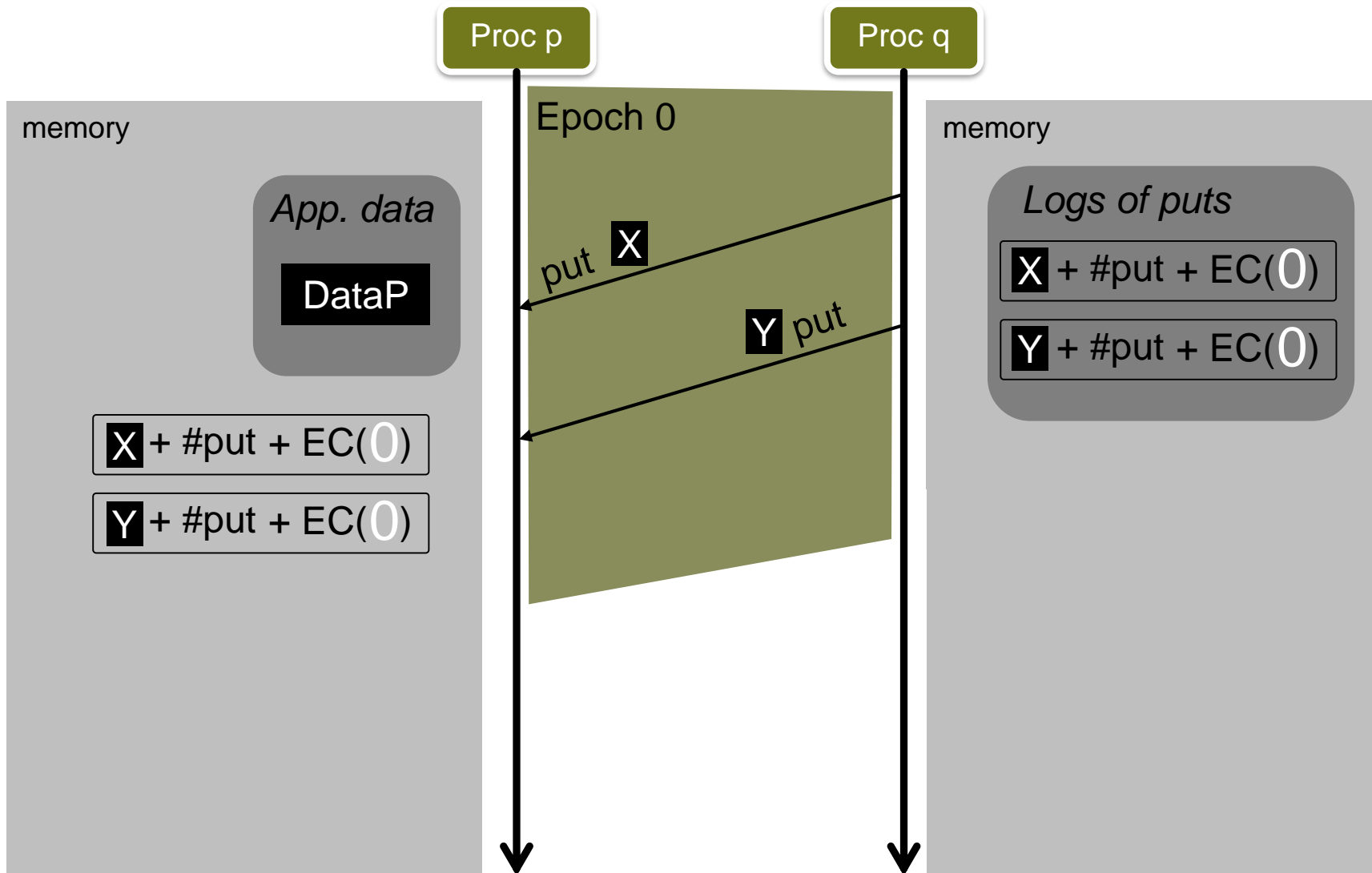
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



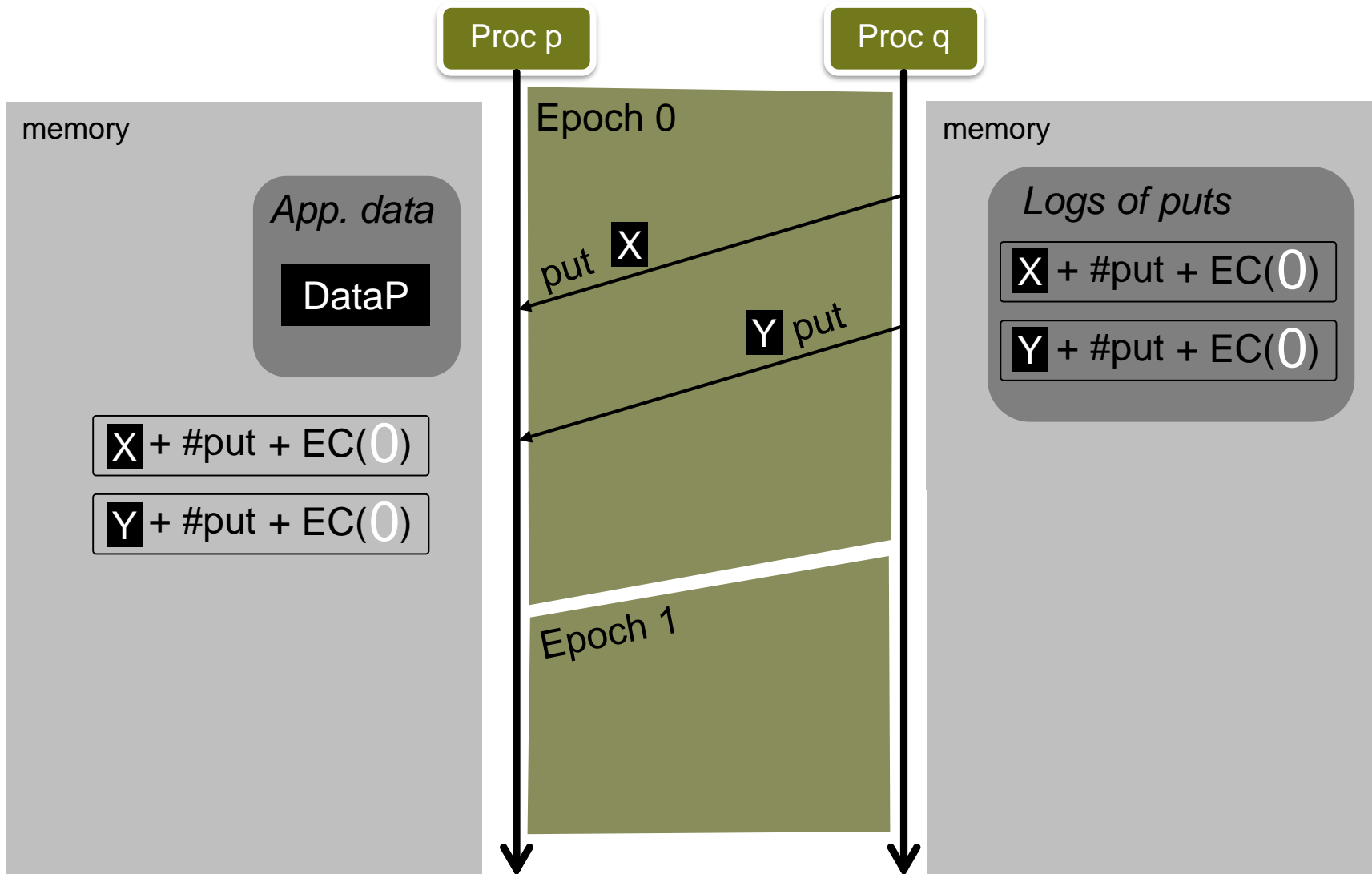
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



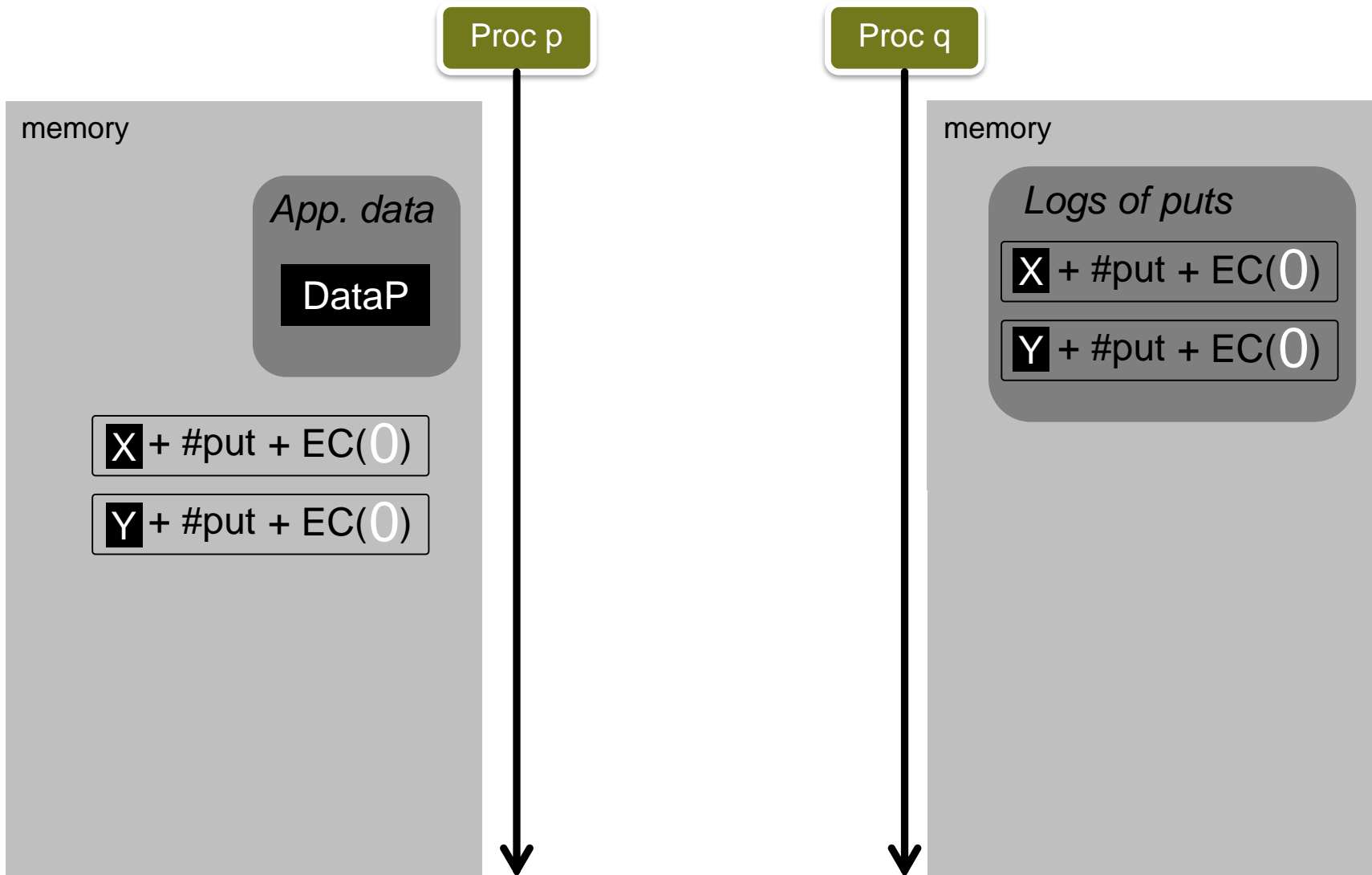
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



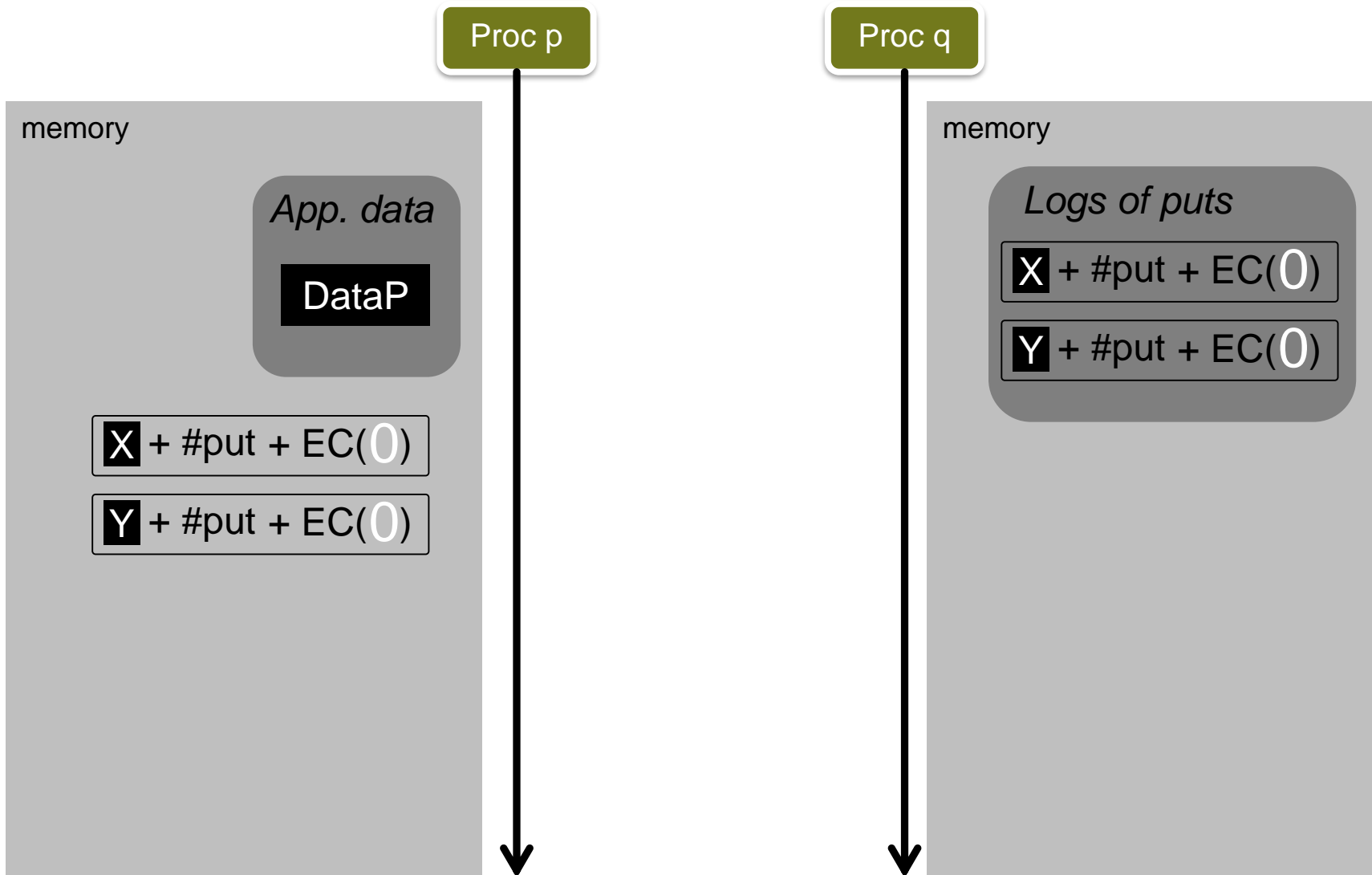
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



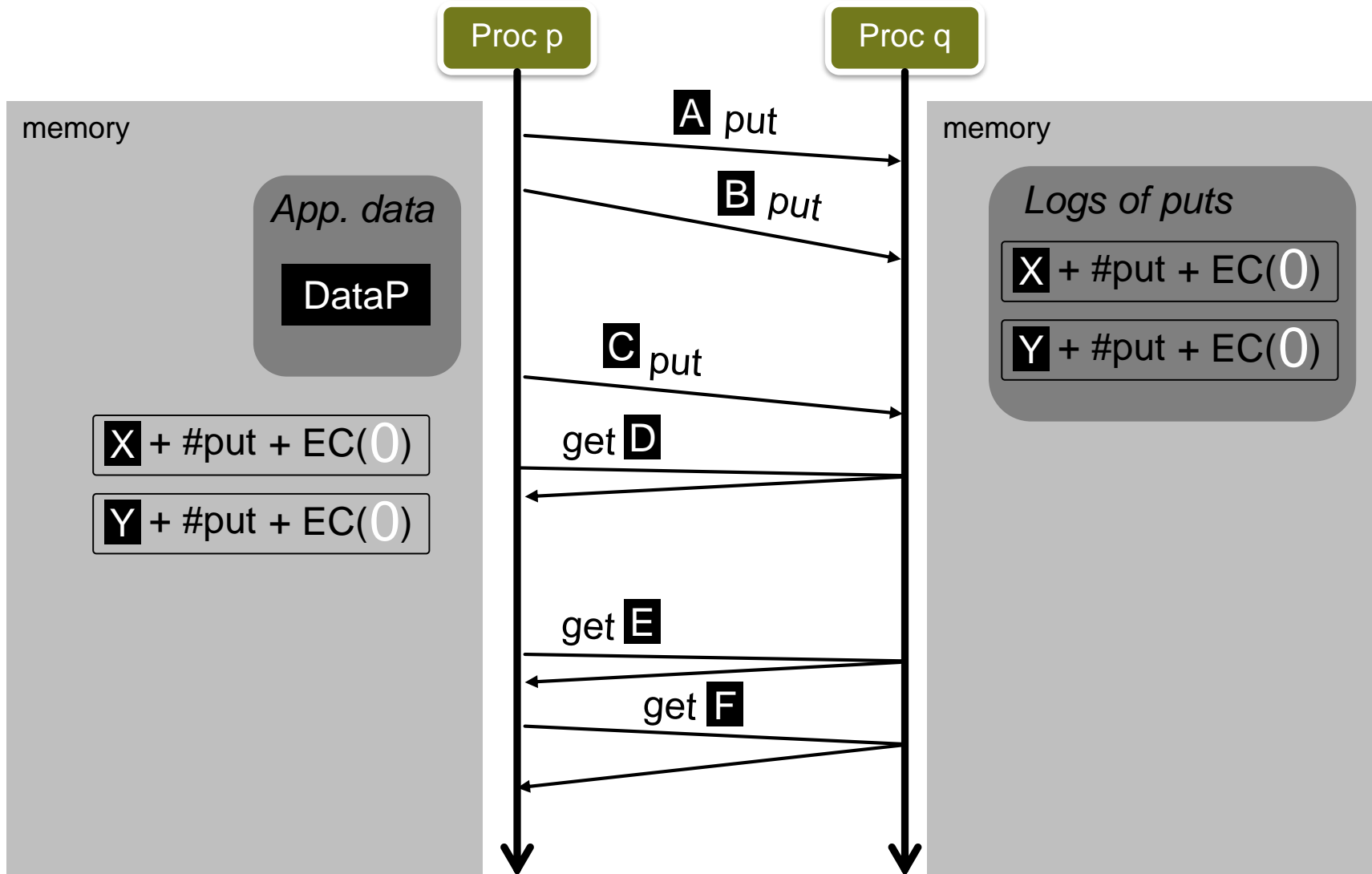
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



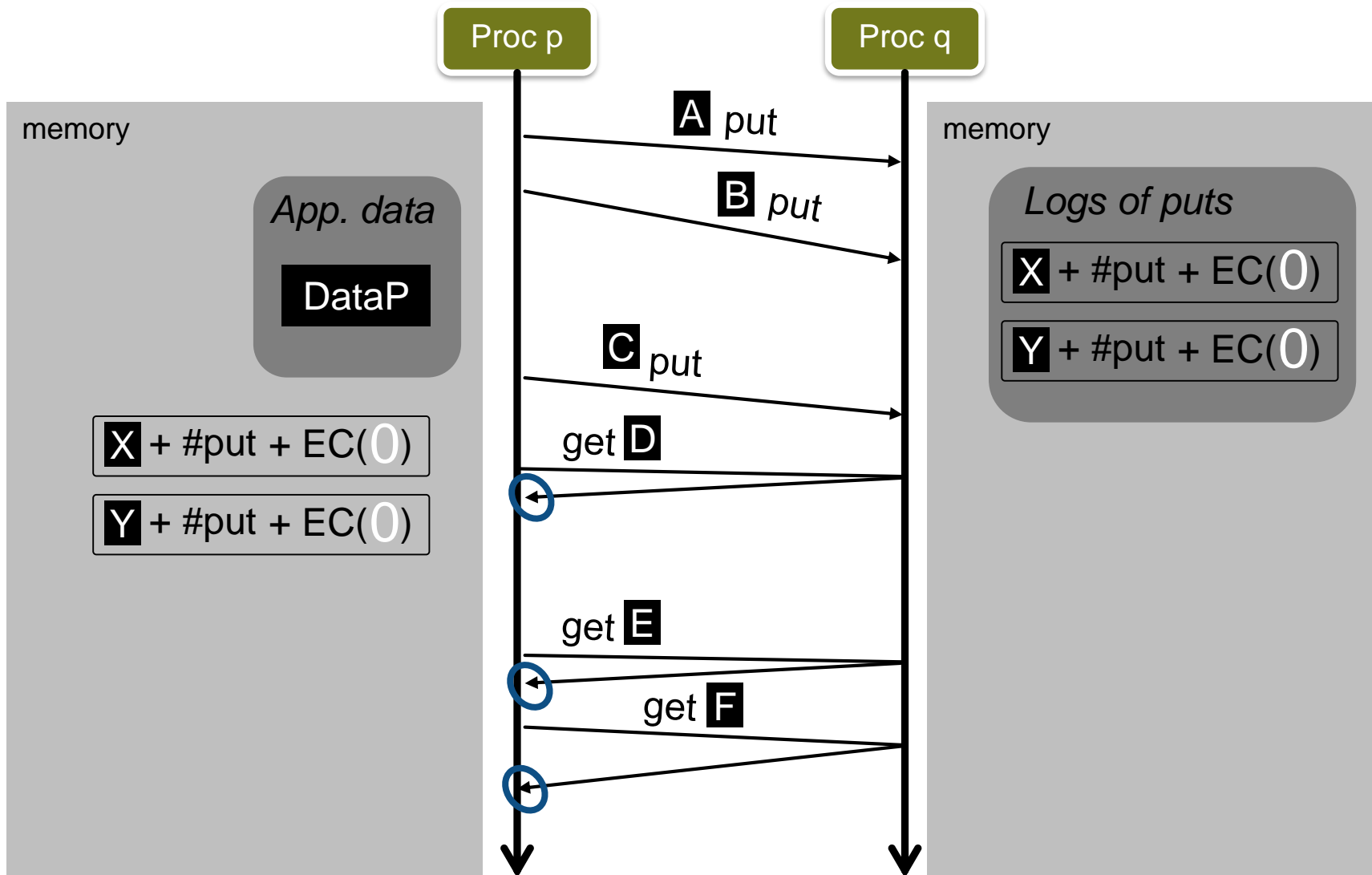
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



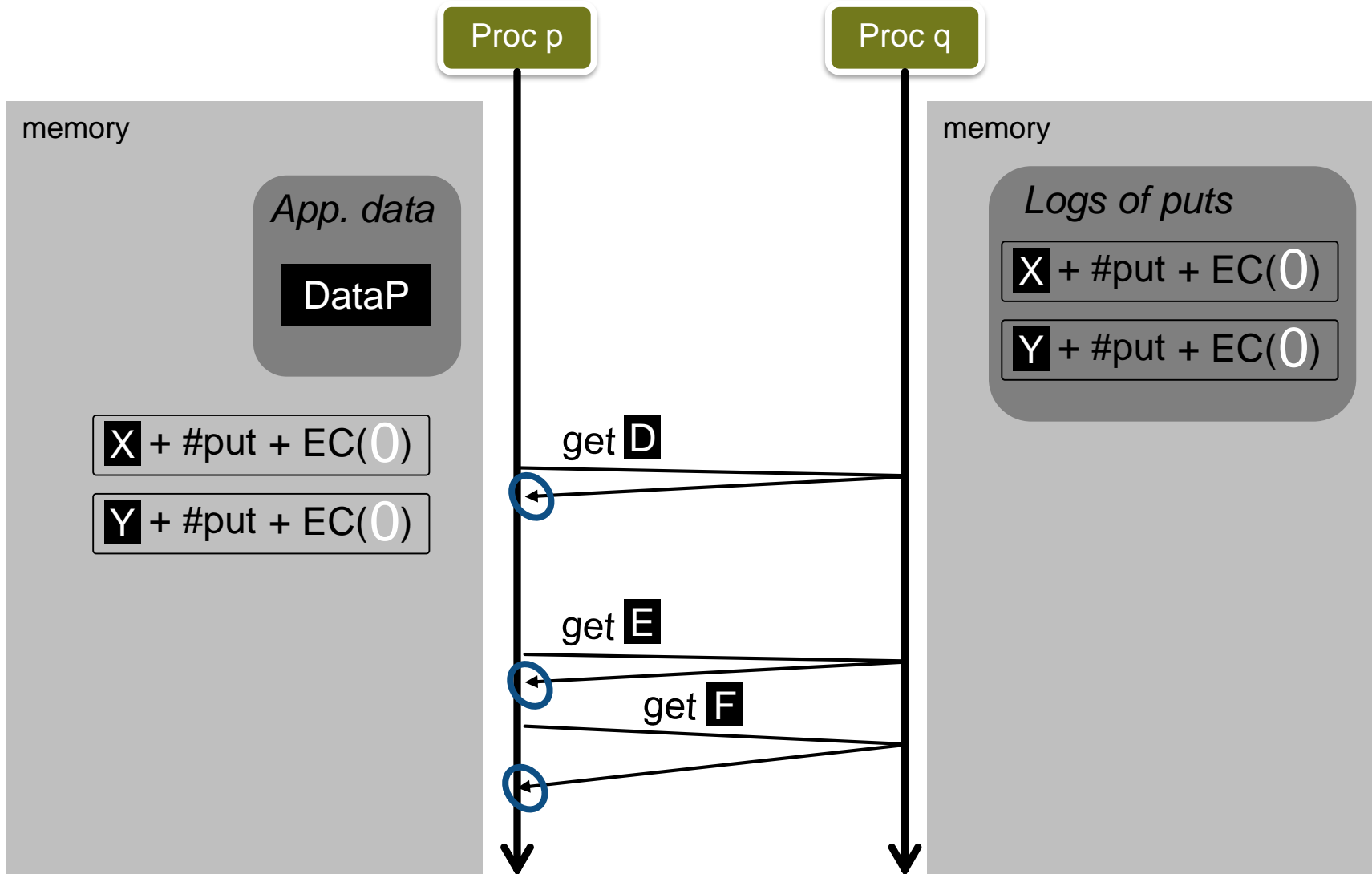
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



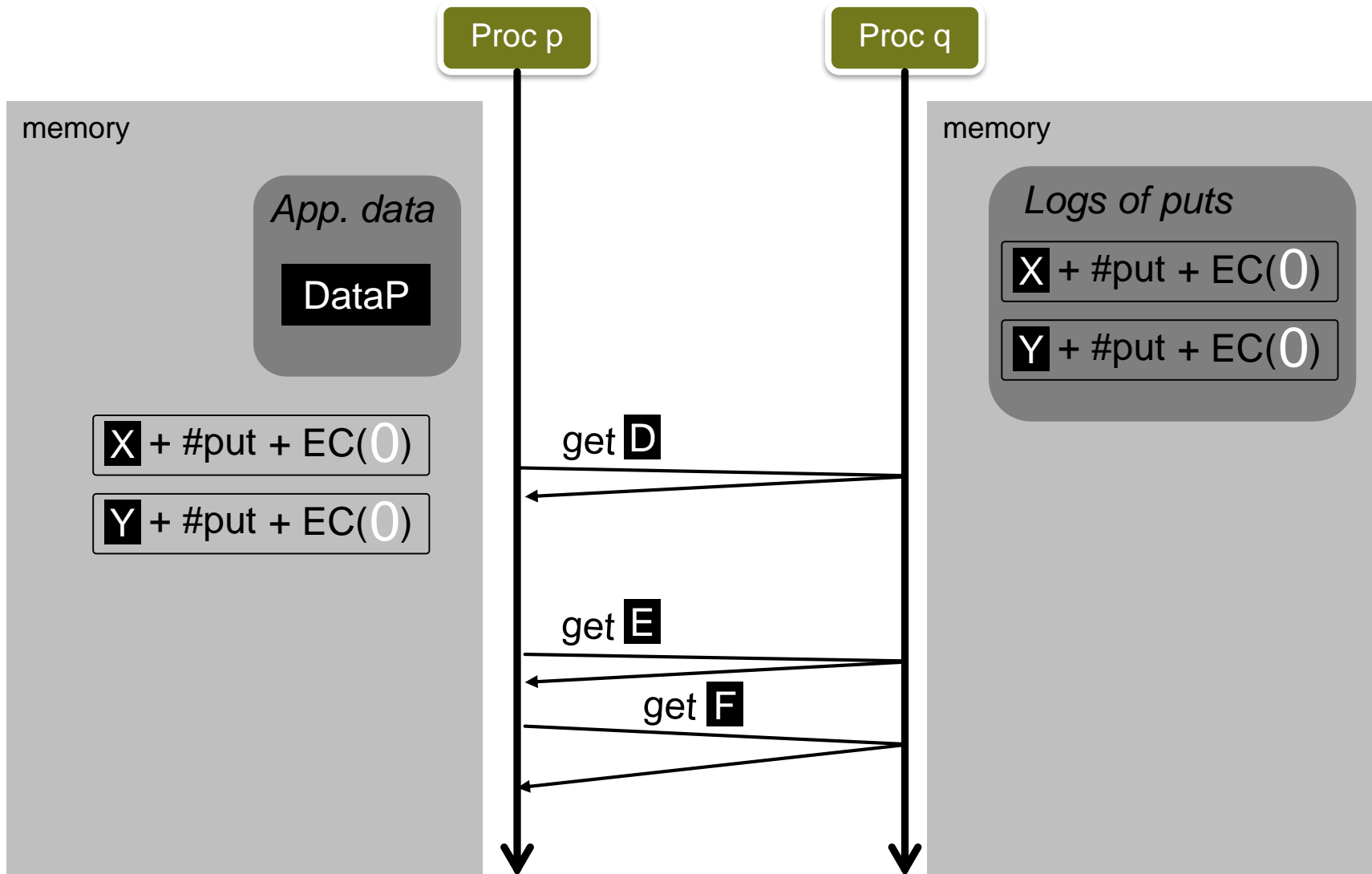
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



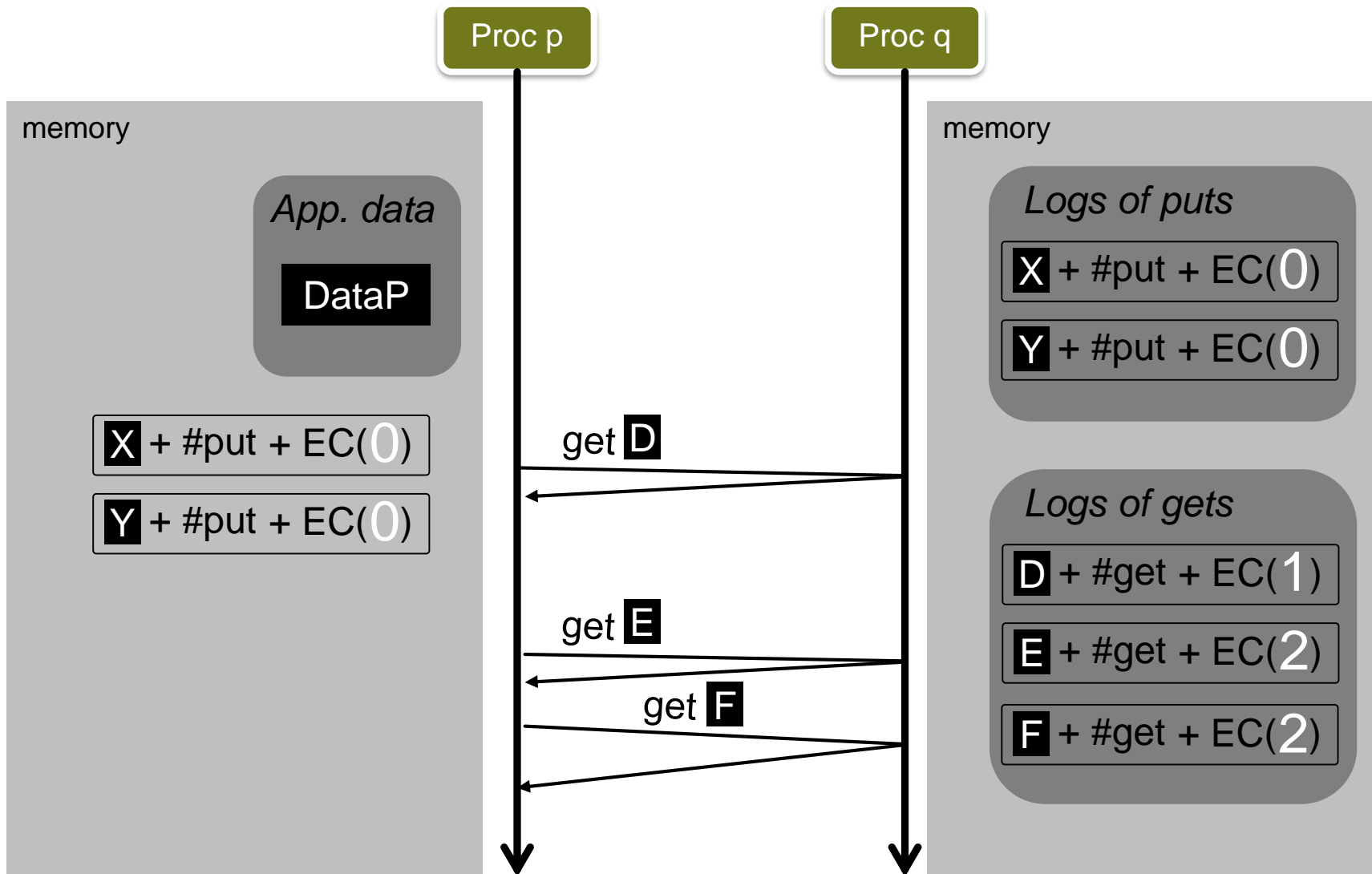
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



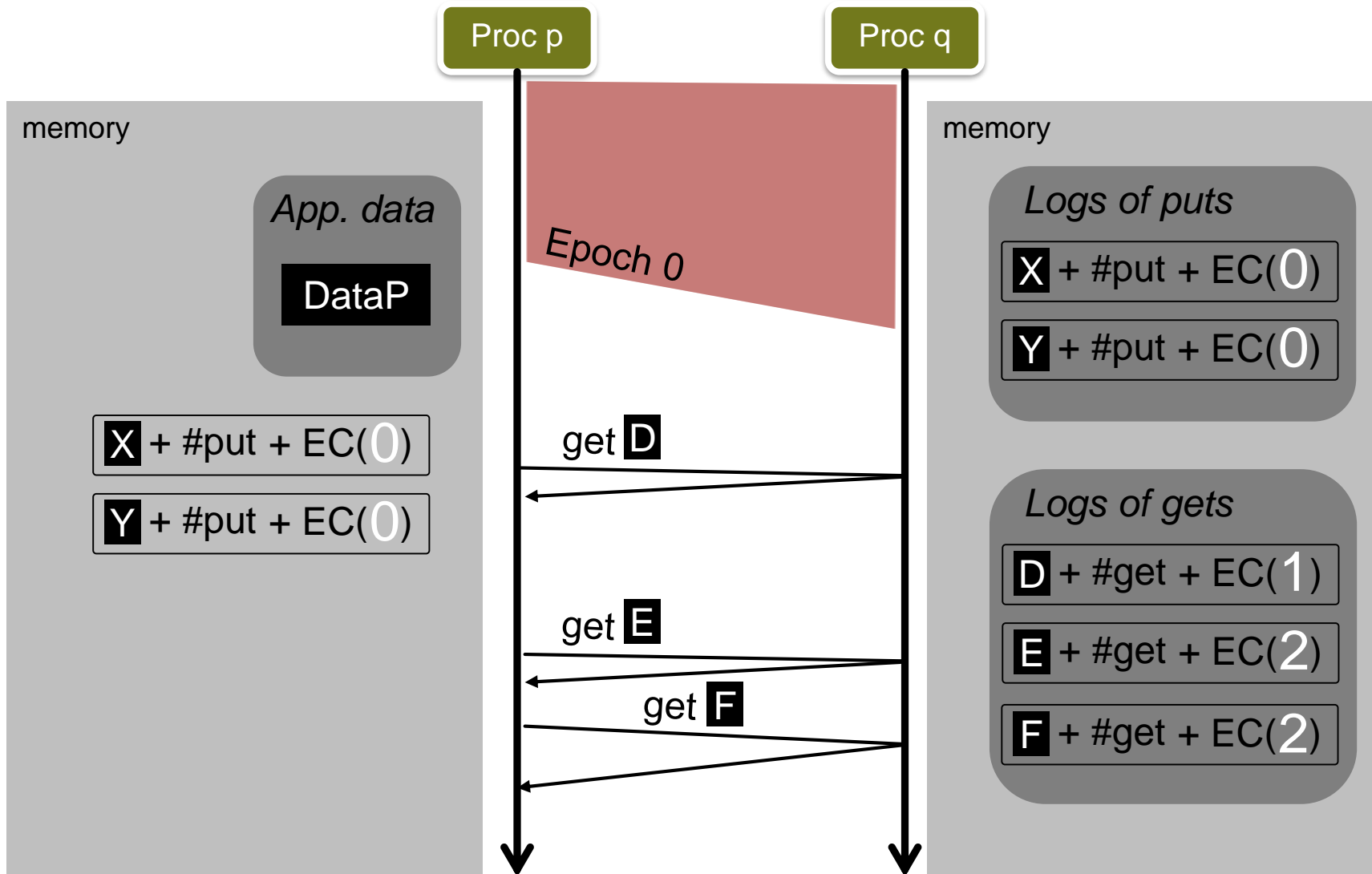
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



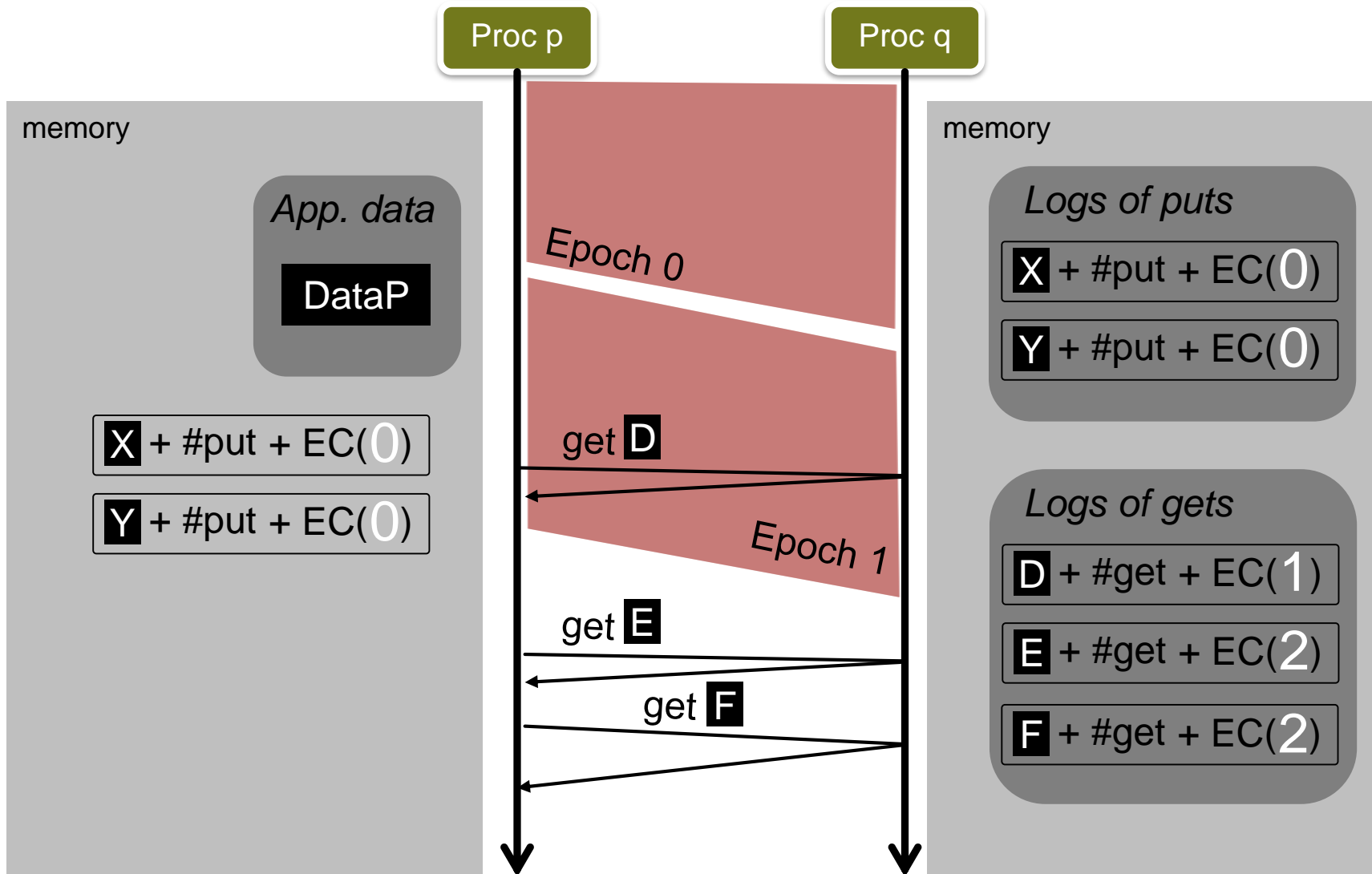
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



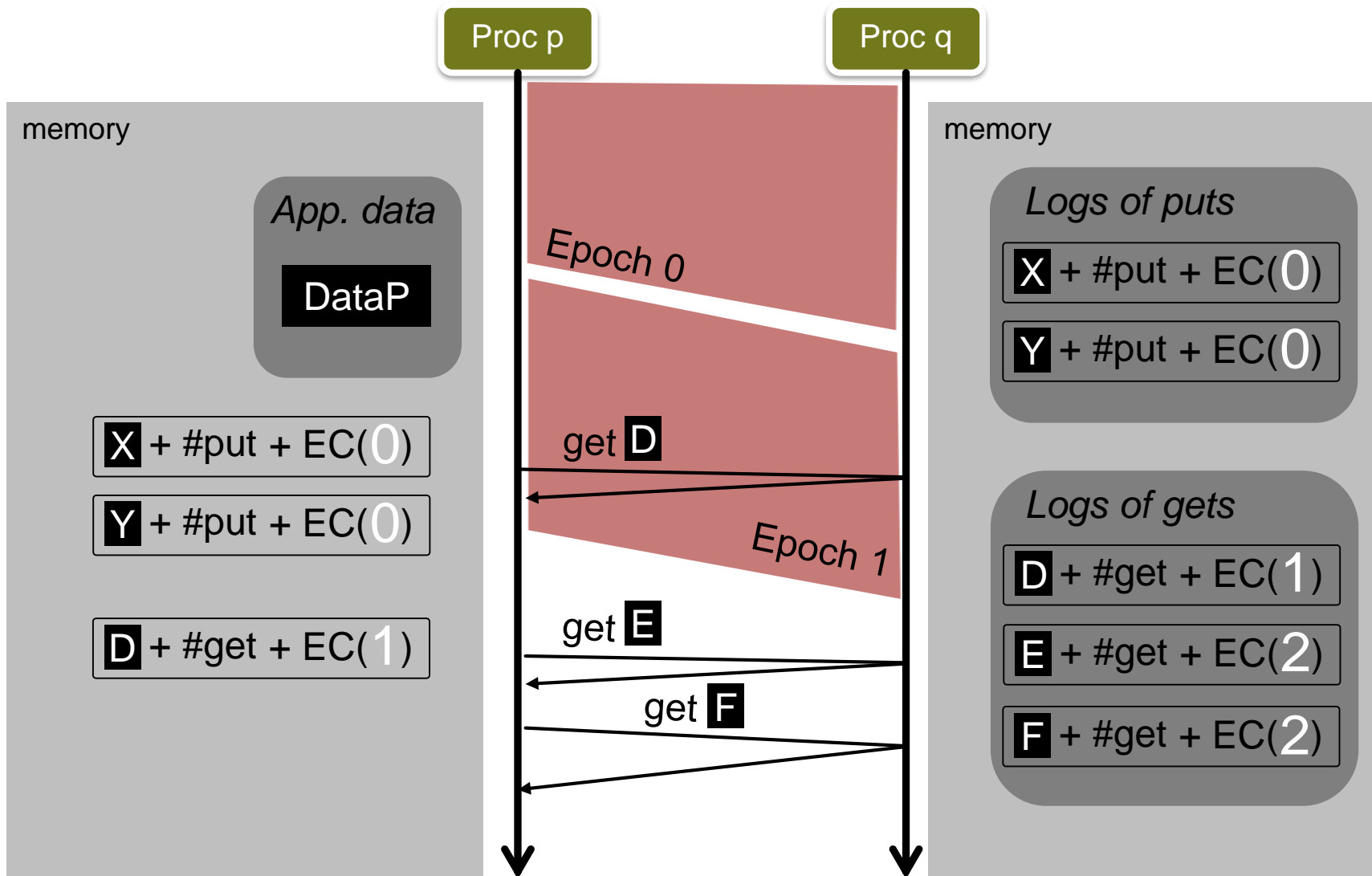
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



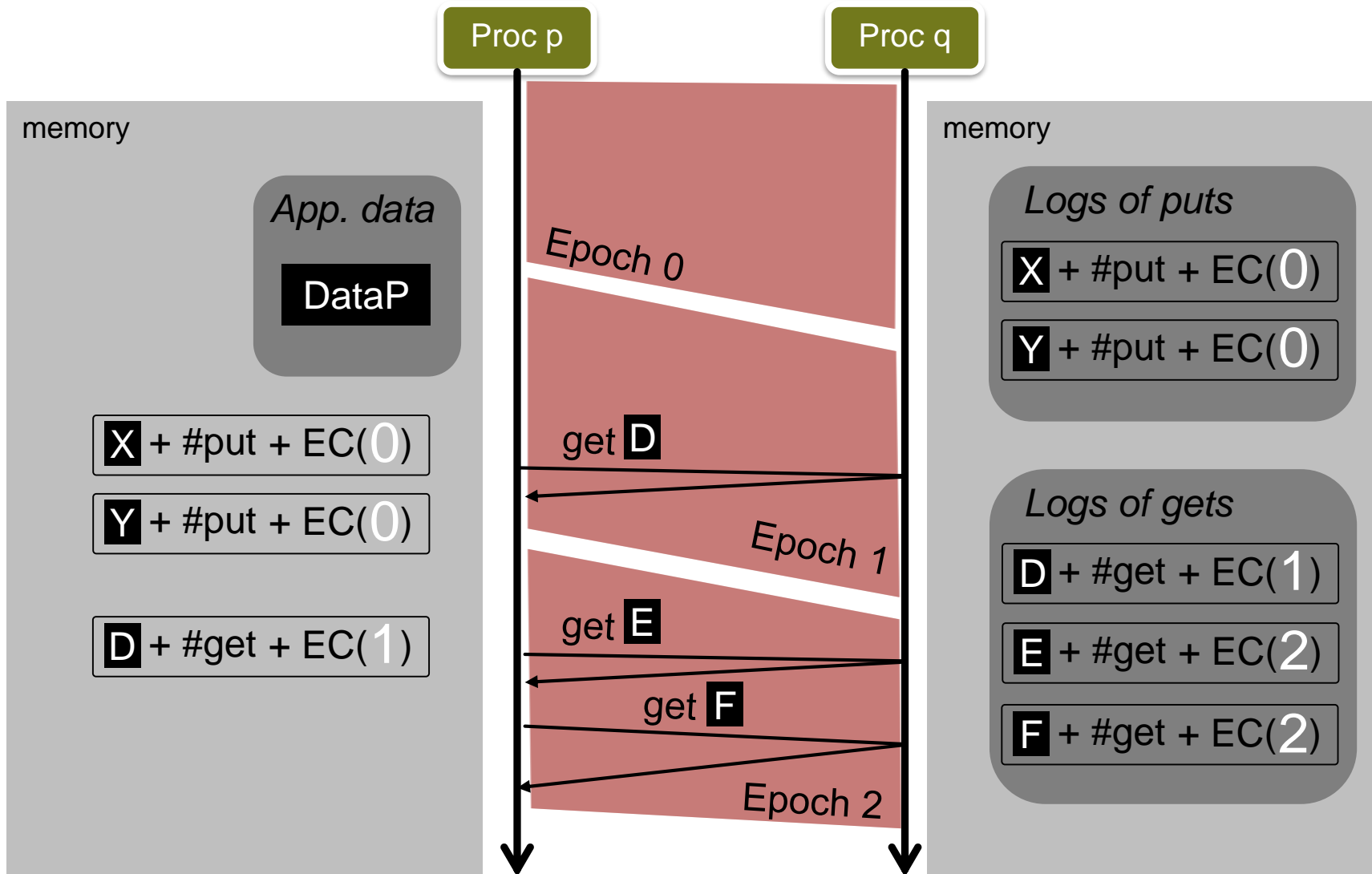
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



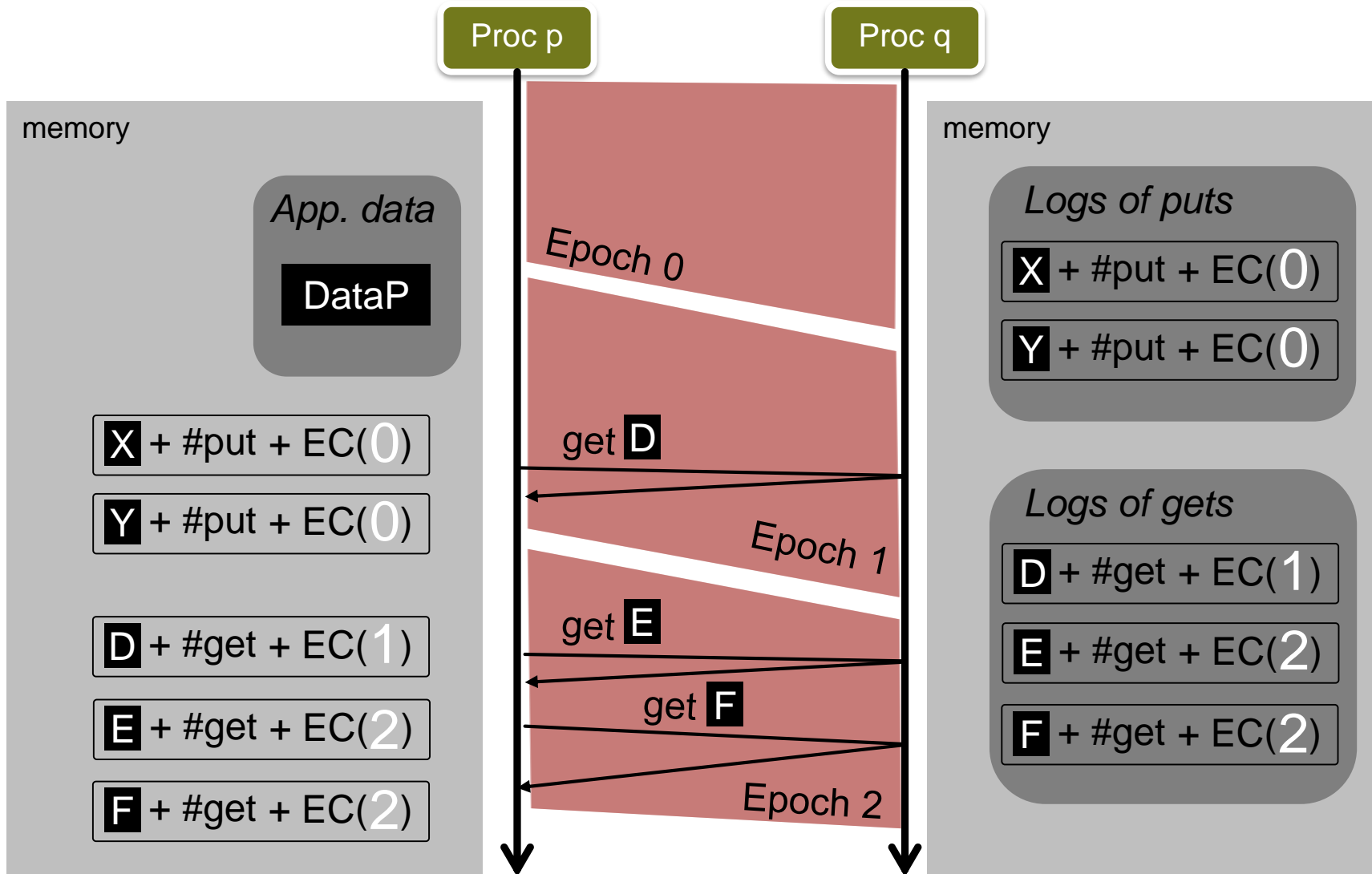
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



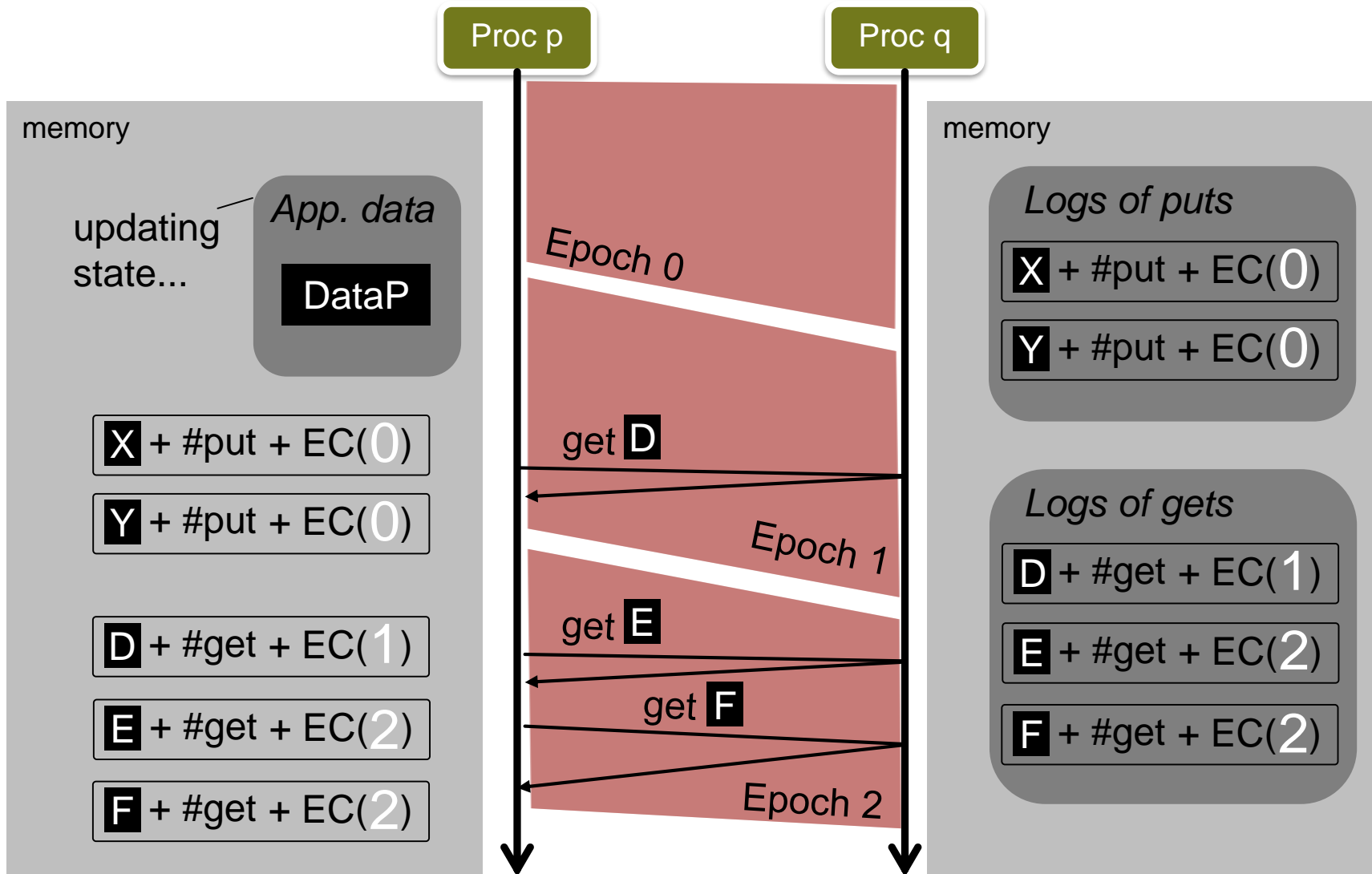
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



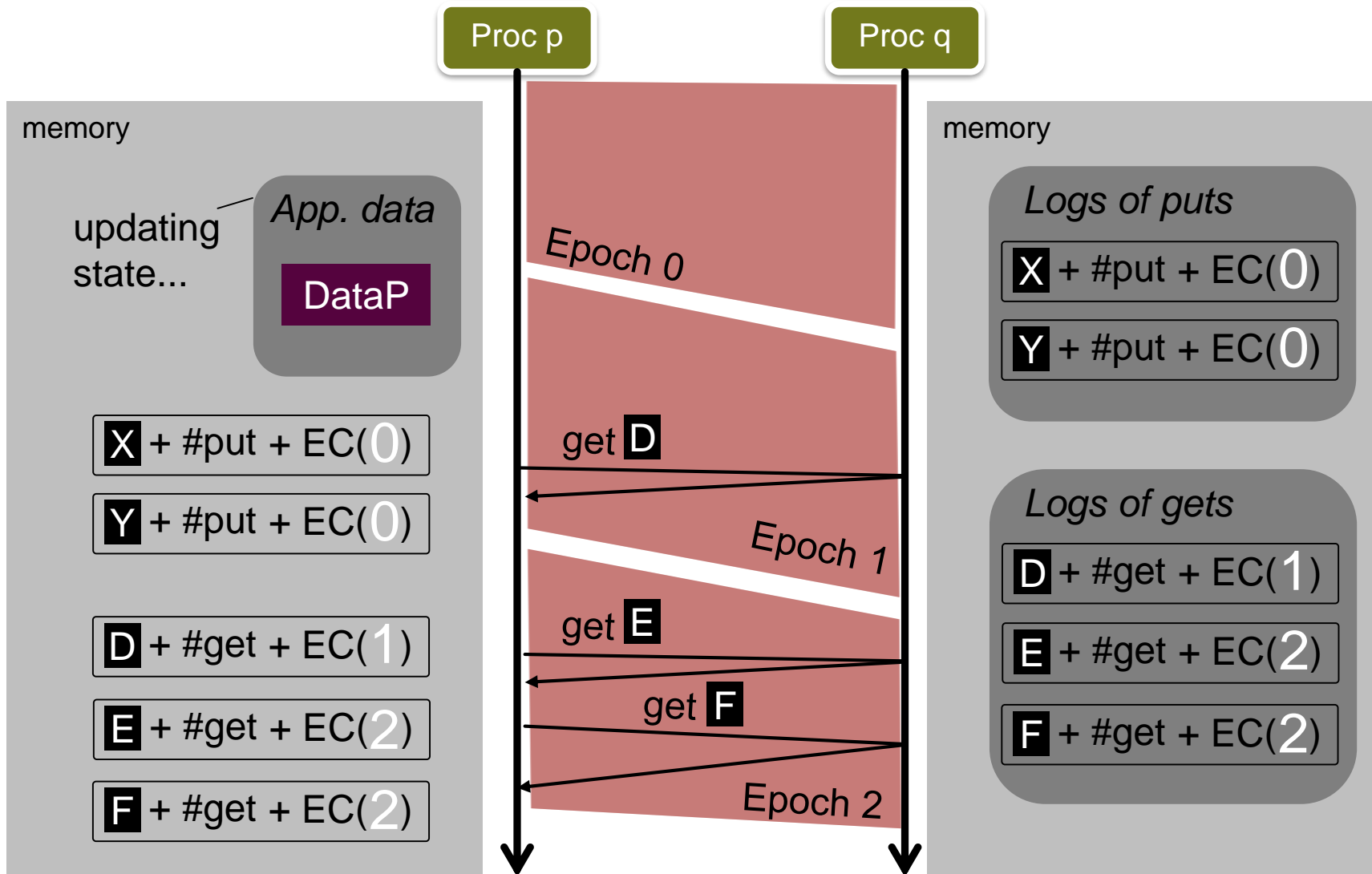
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



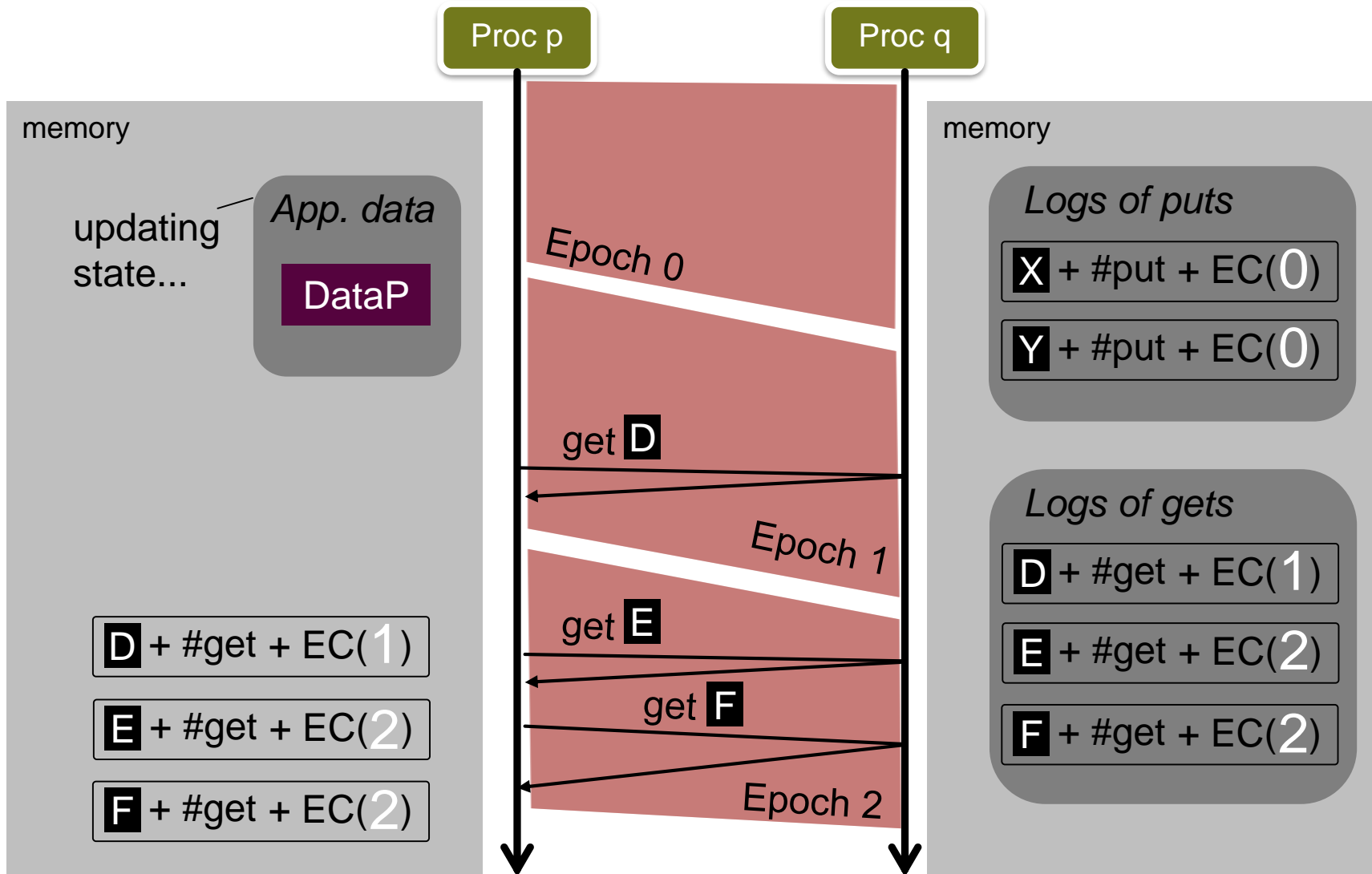
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



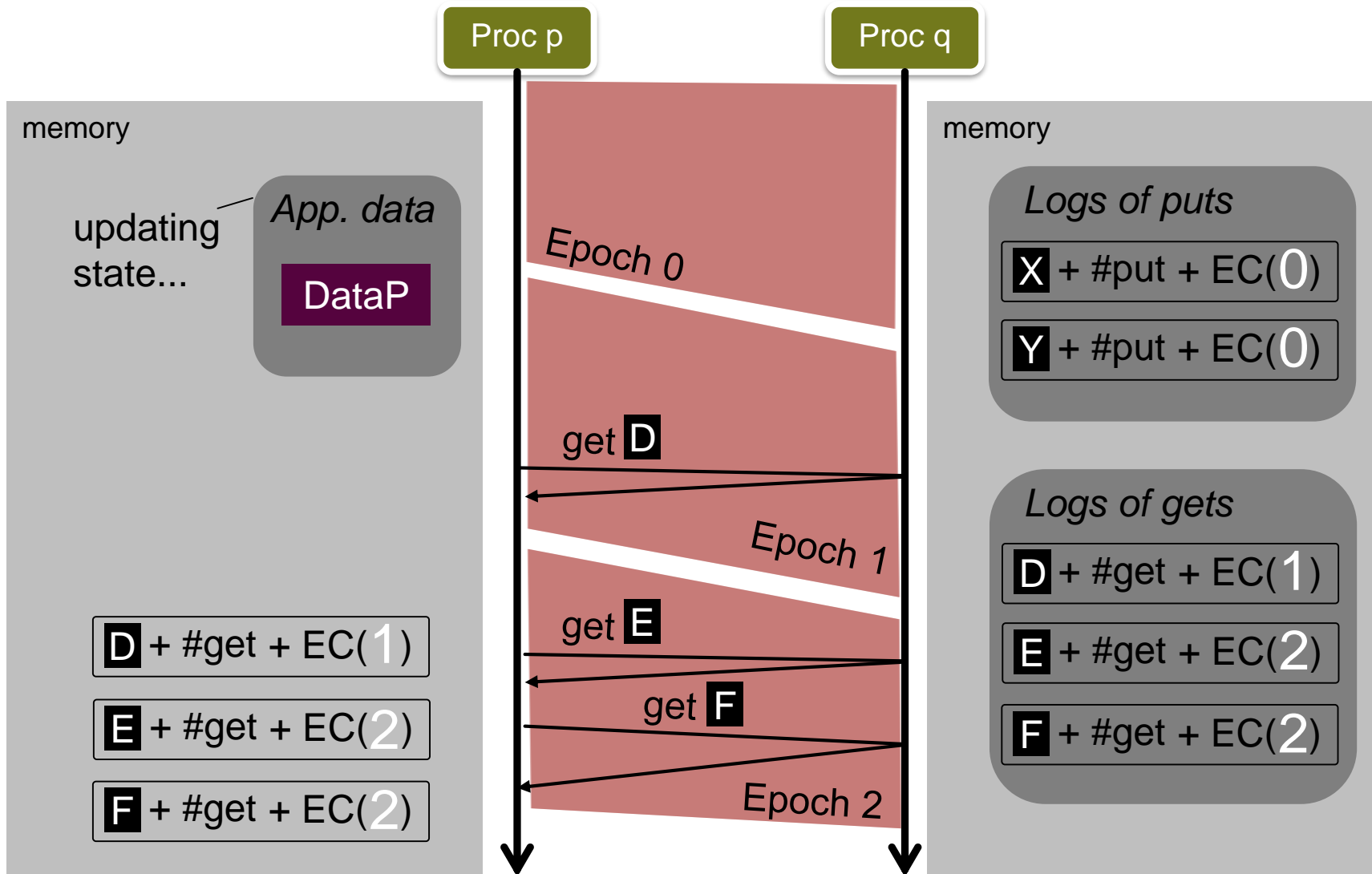
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



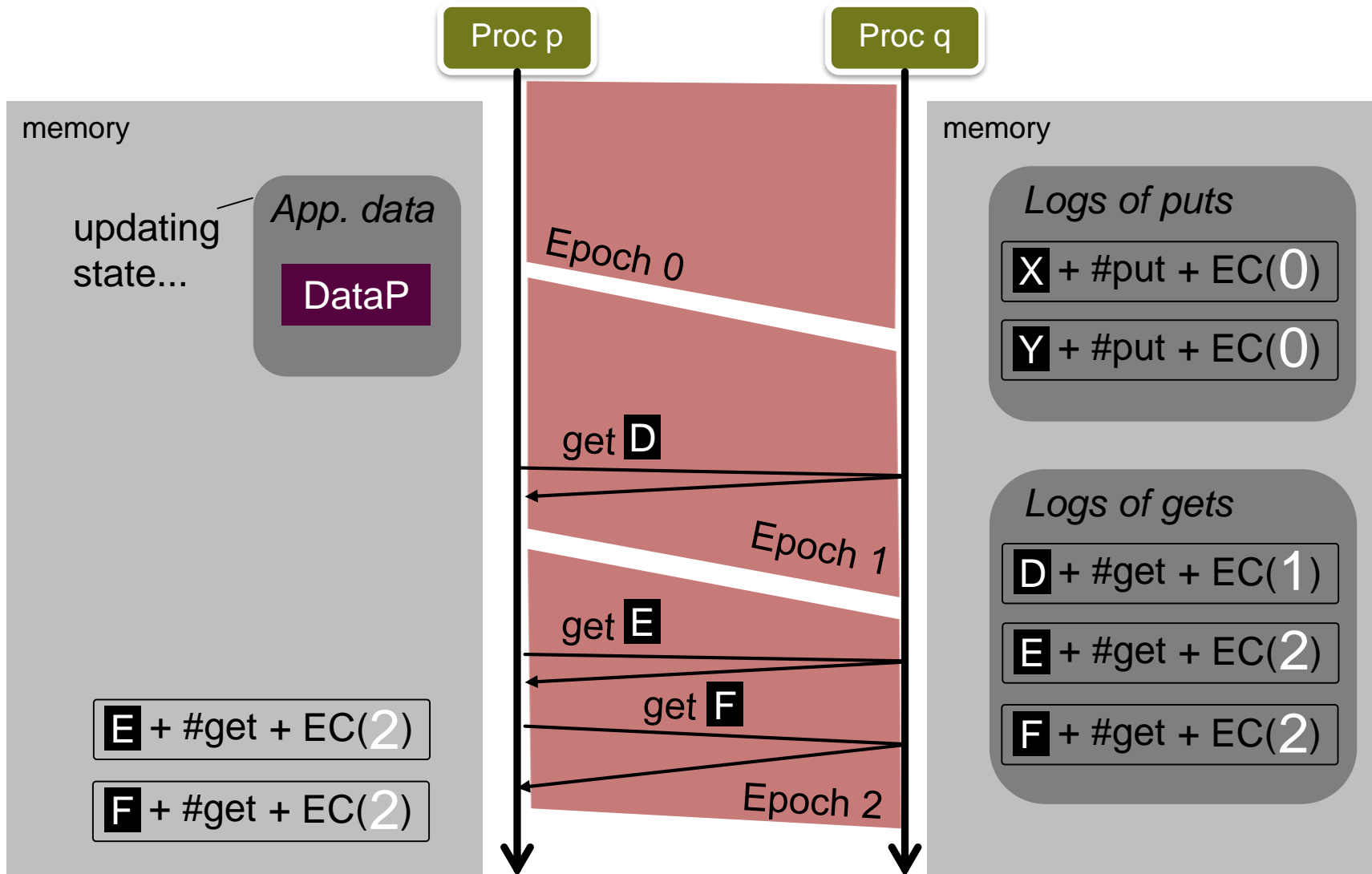
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



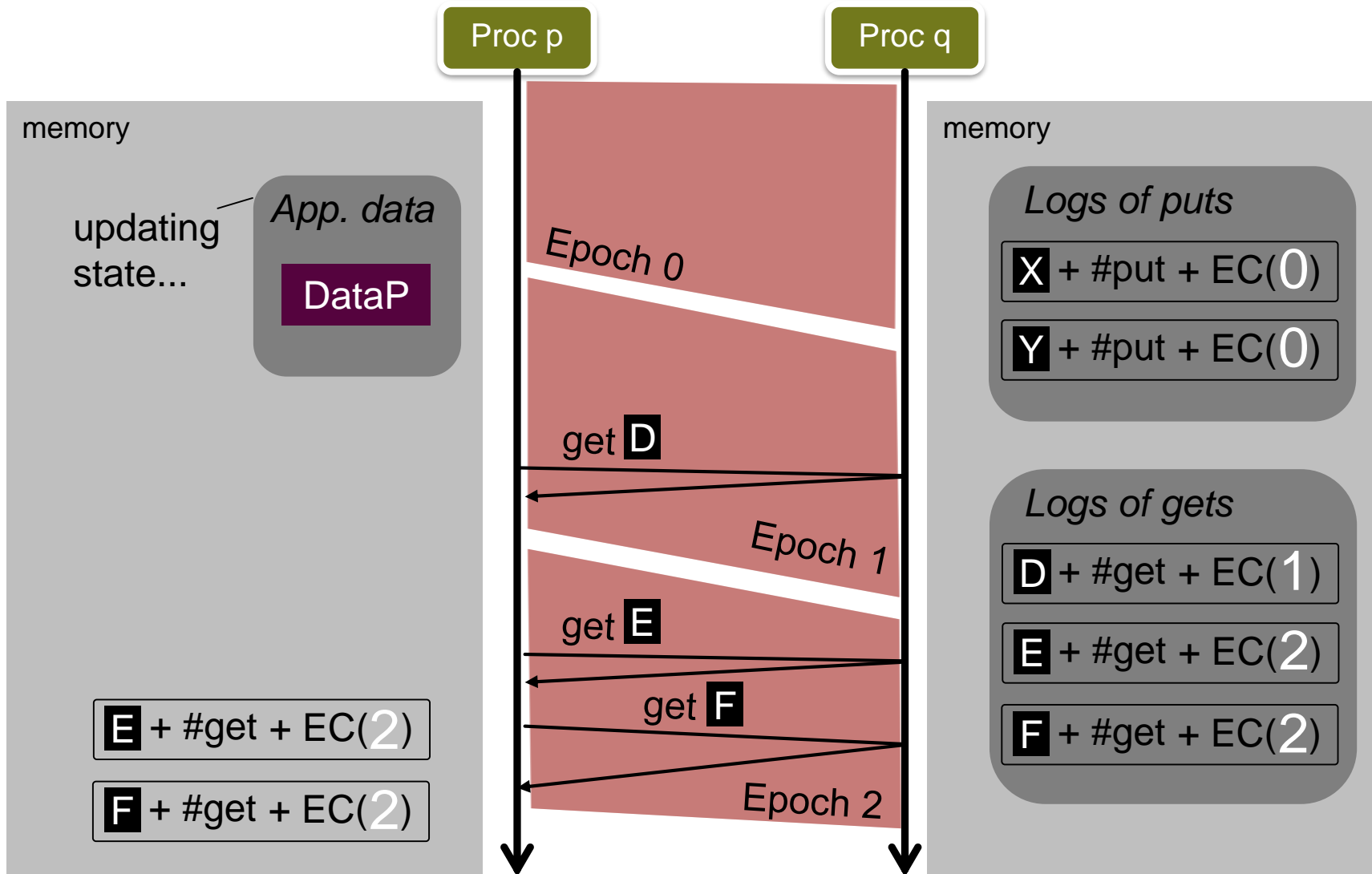
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



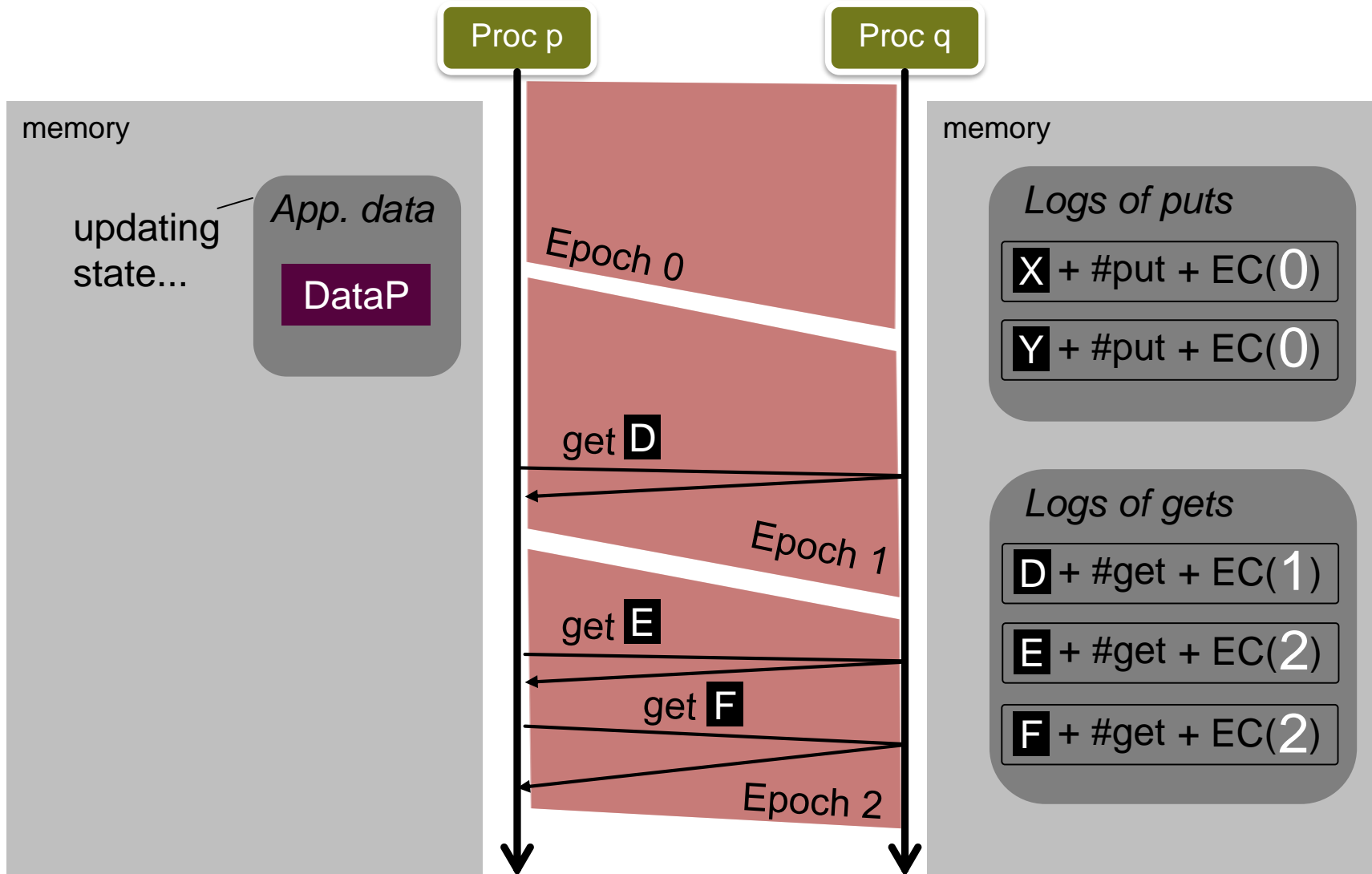
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



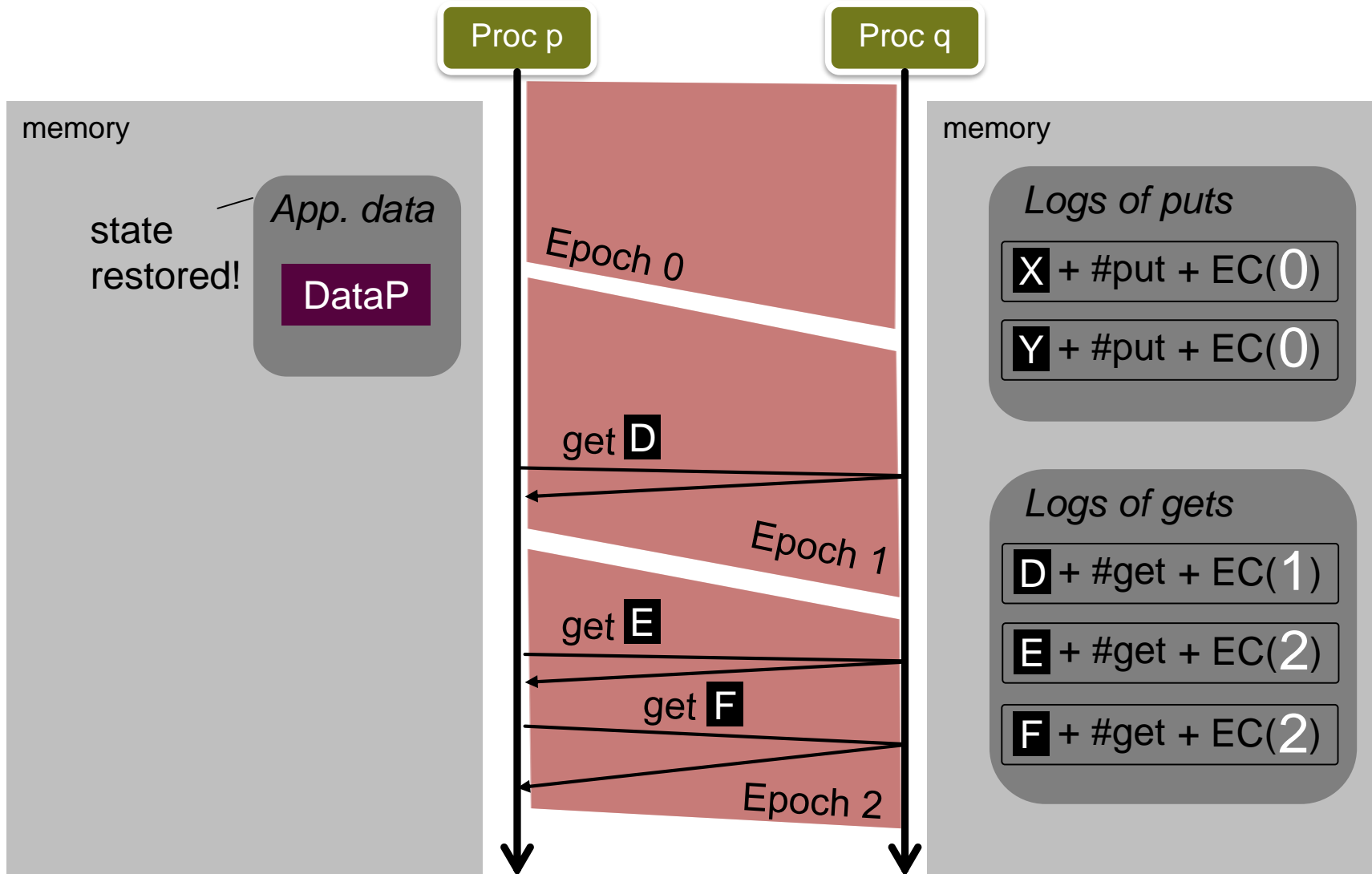
RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint



RMA: RECOVERY

Stage 2: replay actions beyond the checkpoint





OVERVIEW OF OUR RESEARCH

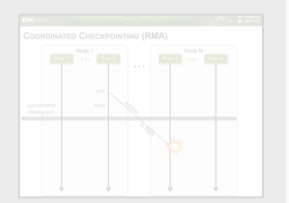
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

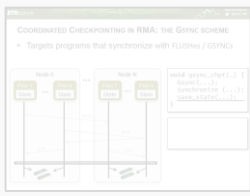
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

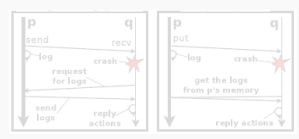


MP vs. RMA

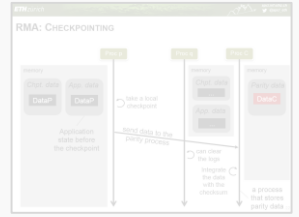


Schemes

UC in RMA

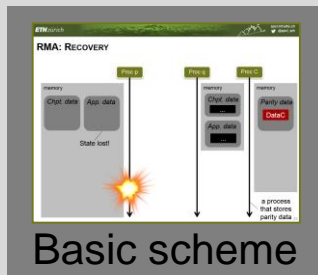


MP vs. RMA



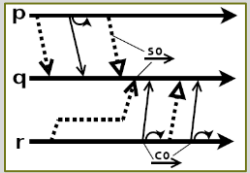
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order as the gsync order).*

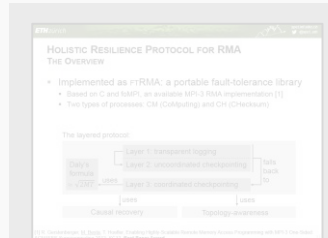
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

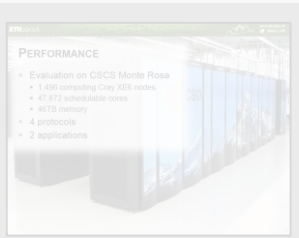
Holistic fault-tolerance library



Design

Checkpoints on demand

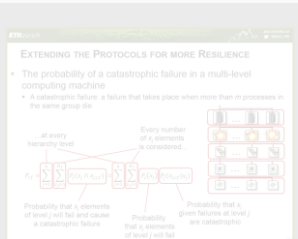
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

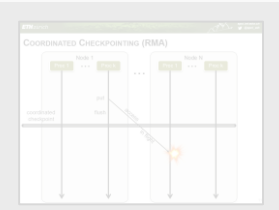
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

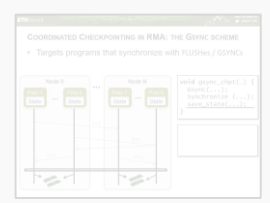
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

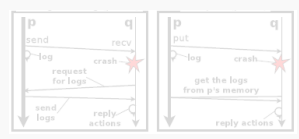


MP vs. RMA

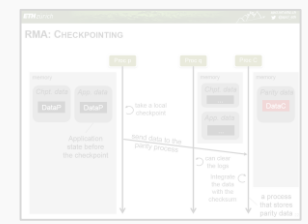


Schemes

UC in RMA

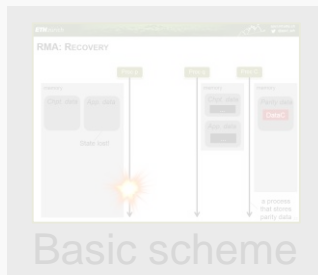


MP vs. RMA



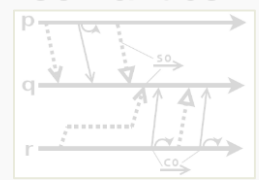
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

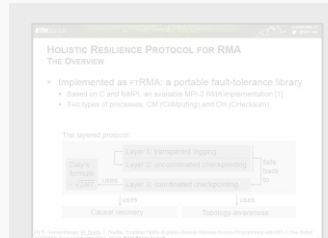
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

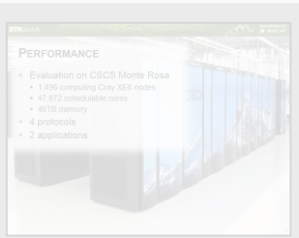
Holistic fault-tolerance library



Design

Checkpoints on demand

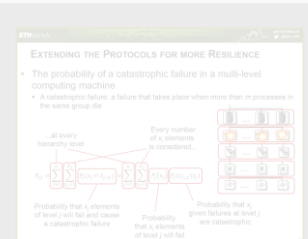
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

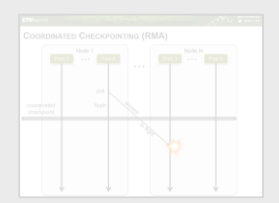
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

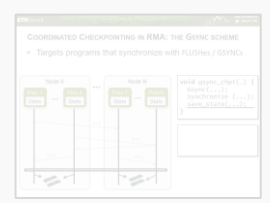
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

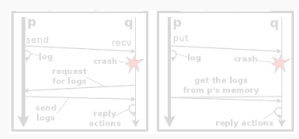


MP vs. RMA

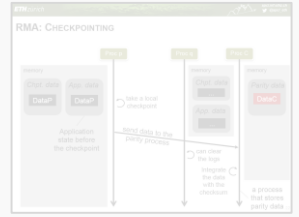


Schemes

UC in RMA

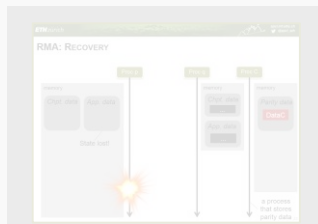


MP vs. RMA



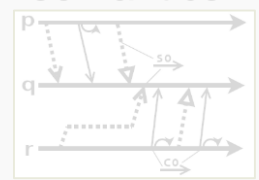
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{coh} order (referred to as the gsync order).*

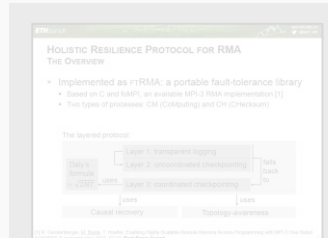
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

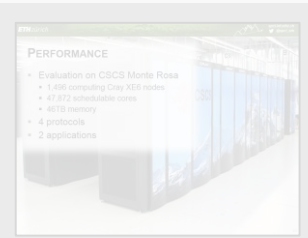
Holistic fault-tolerance library



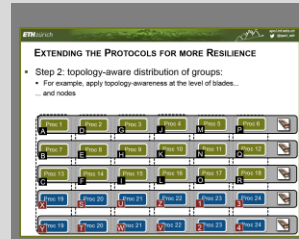
Design

Checkpoints on demand

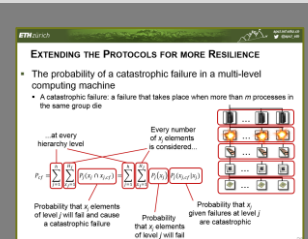
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

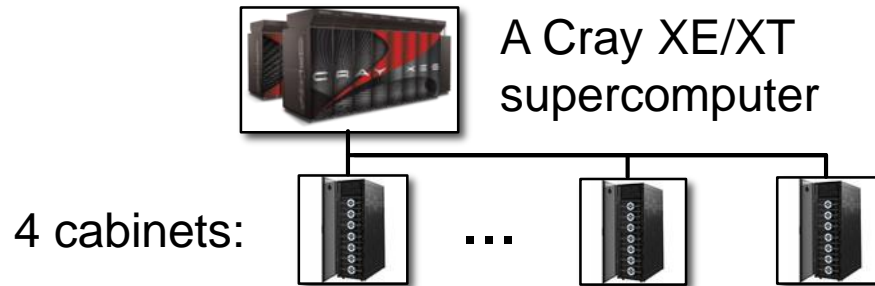
- Today's supercomputers have a hierarchical layout



A Cray XE/XT
supercomputer

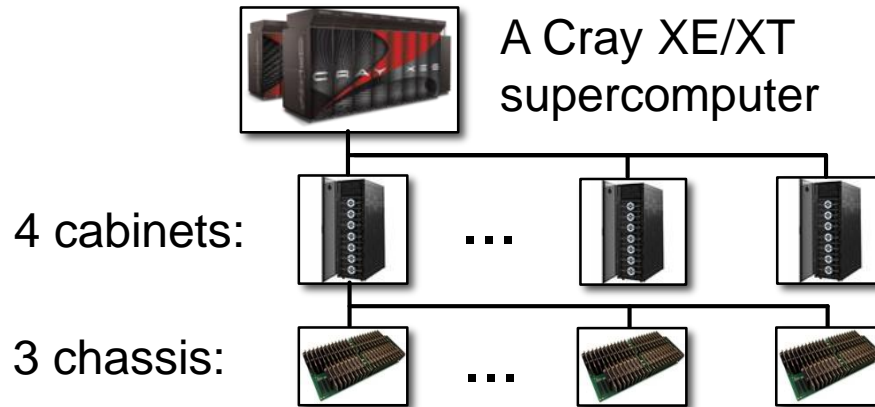
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



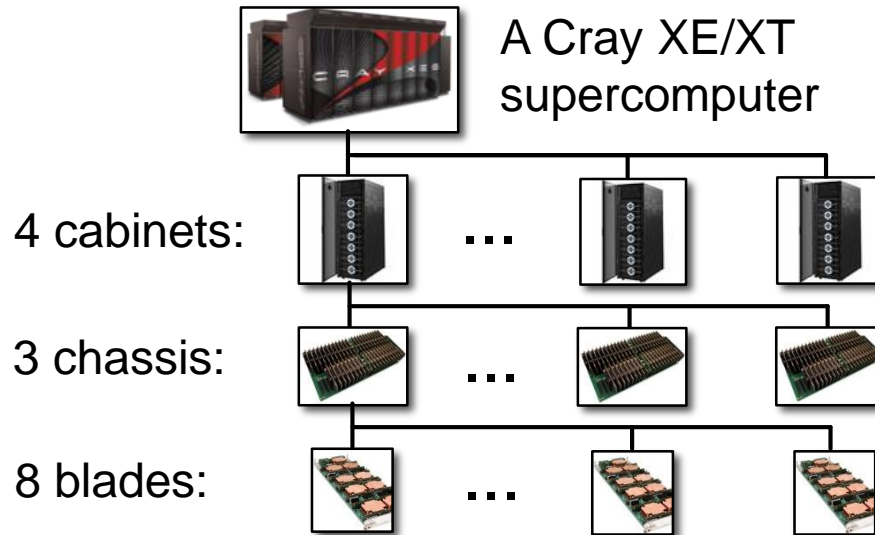
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



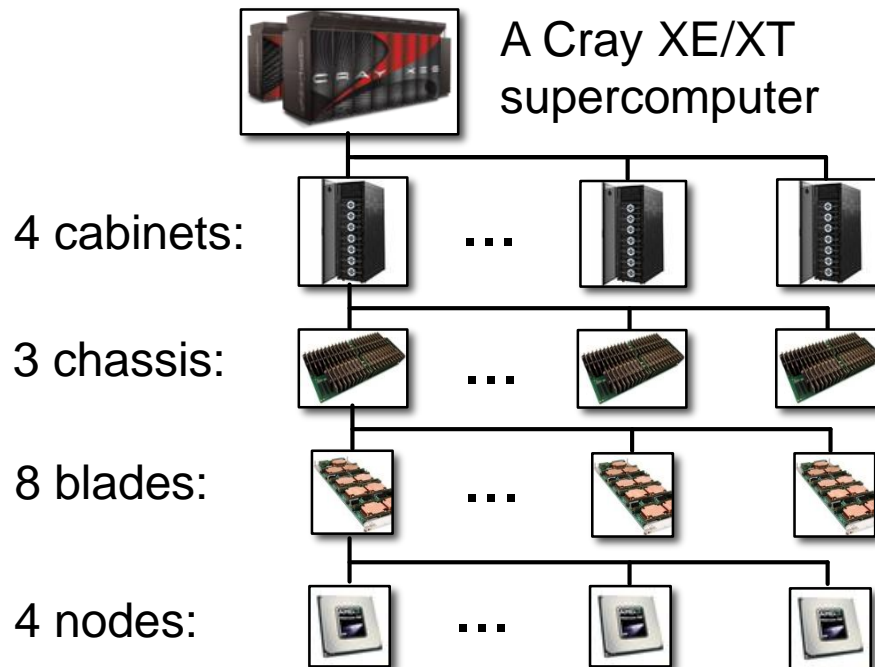
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



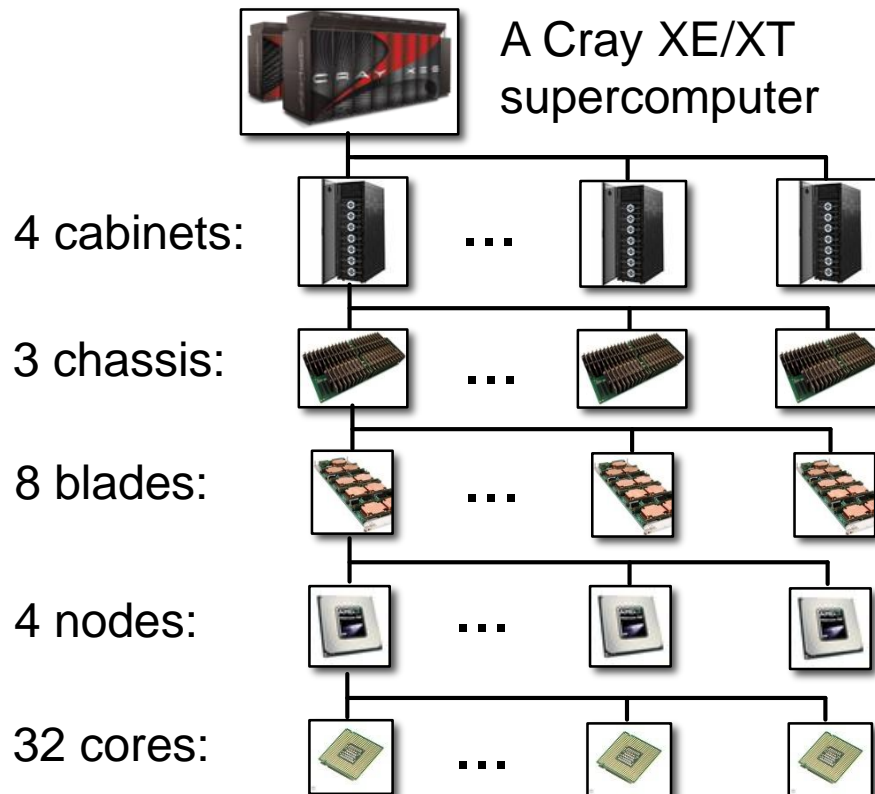
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



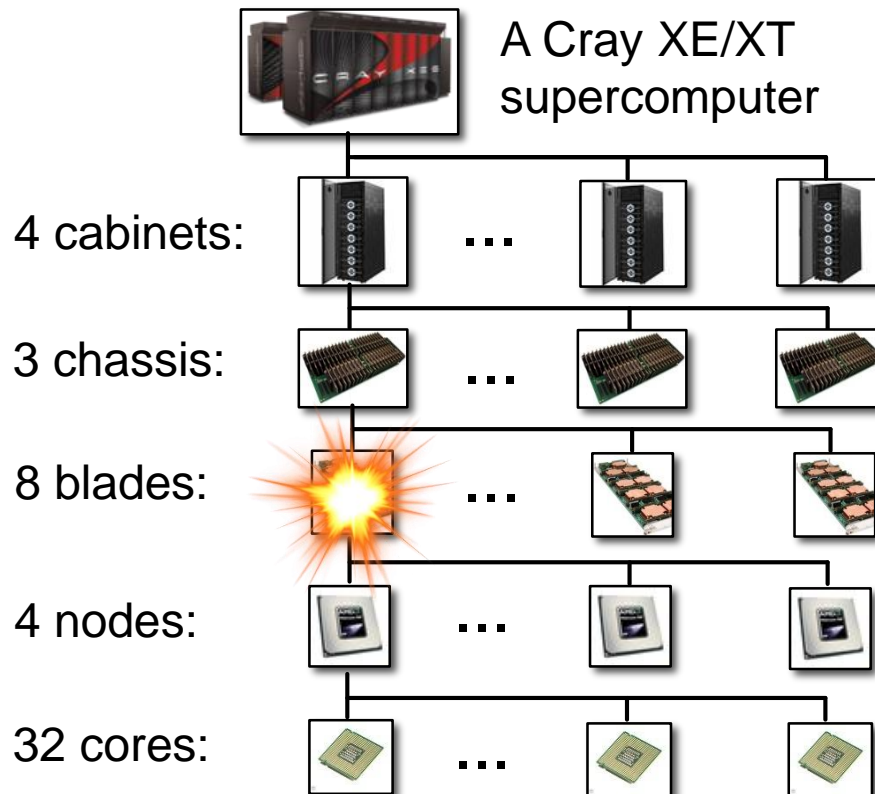
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout



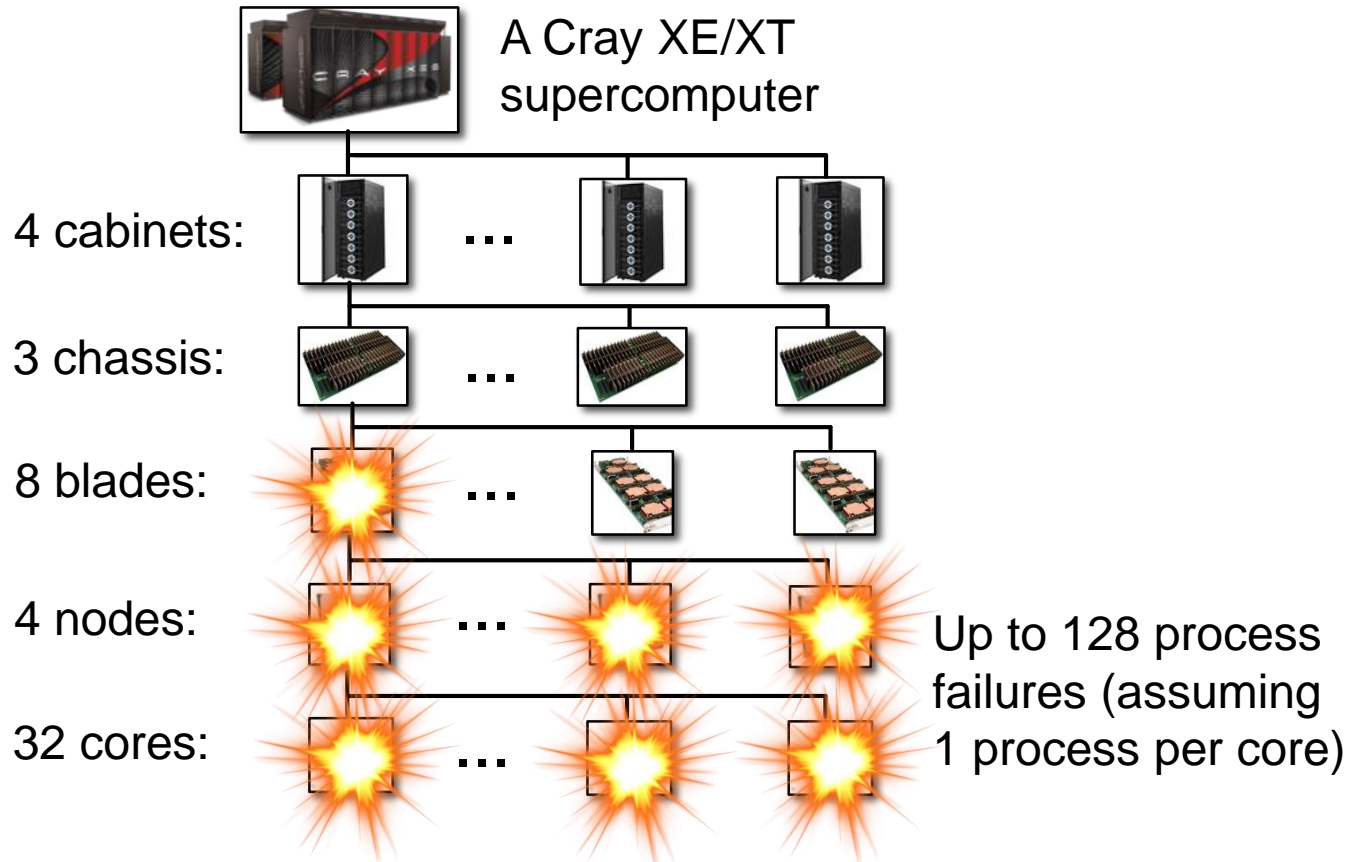
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes



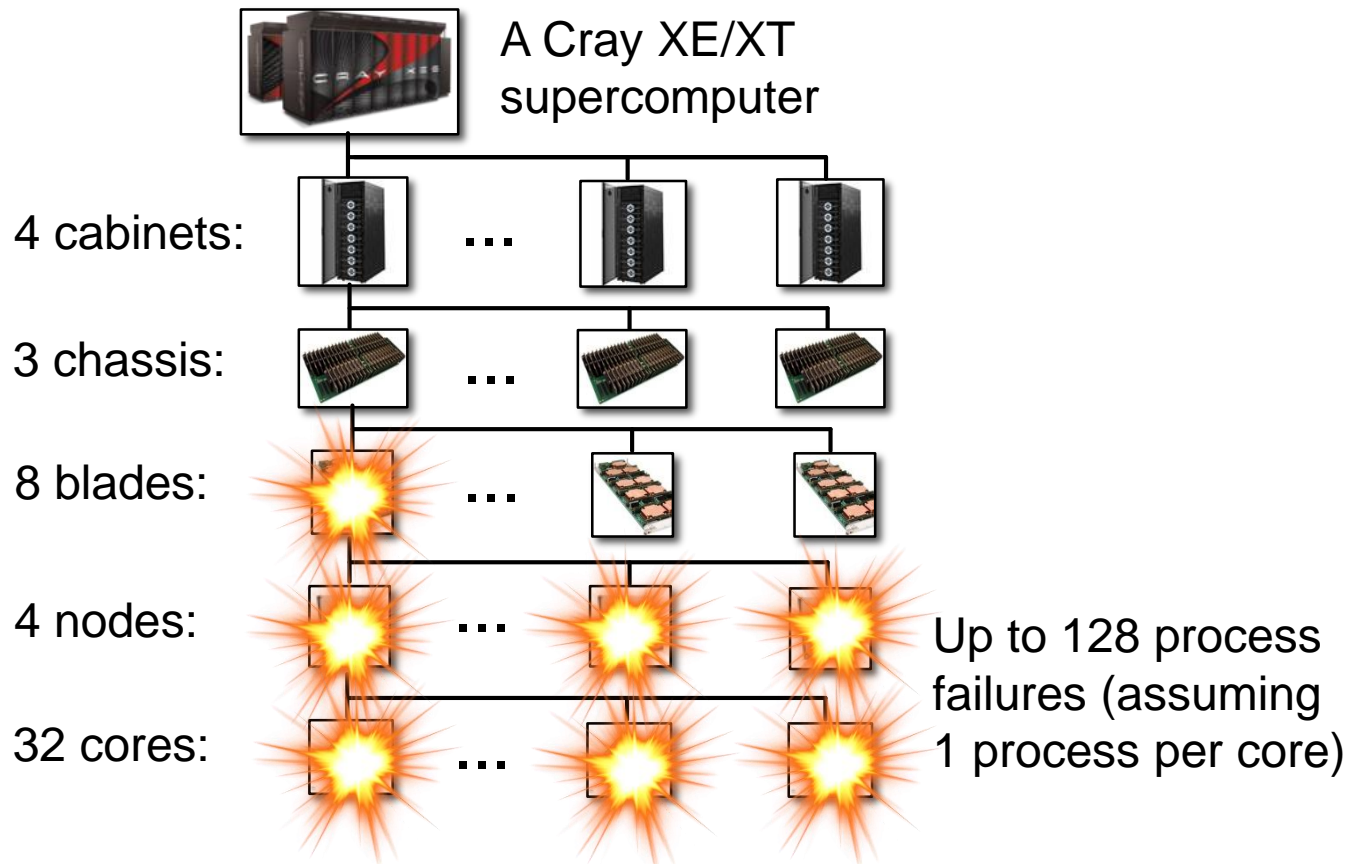
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Today's supercomputers have a hierarchical layout
- A single hardware crash may kill multiple processes
- Introduced protocols usually cannot handle > 1 process crash



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE



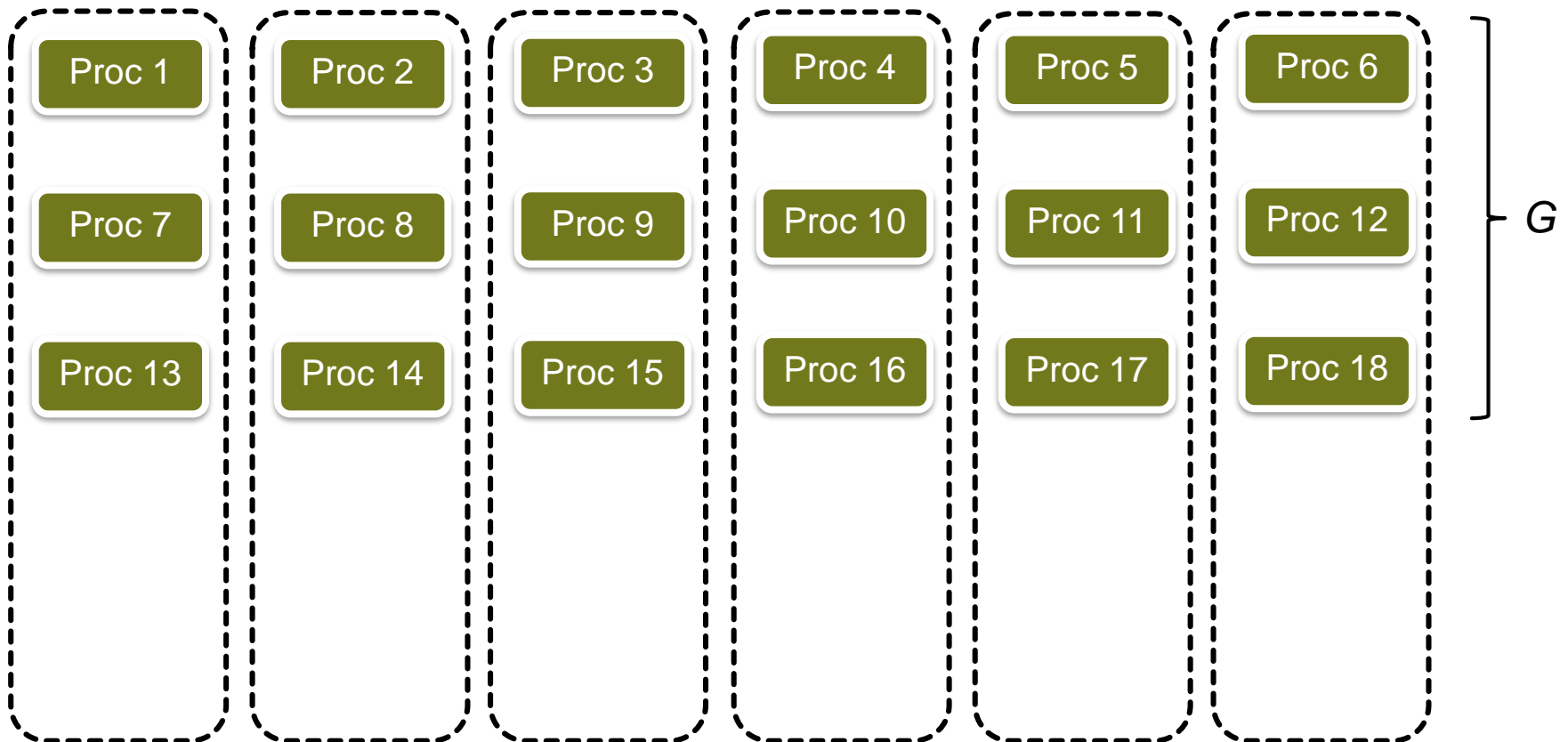
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

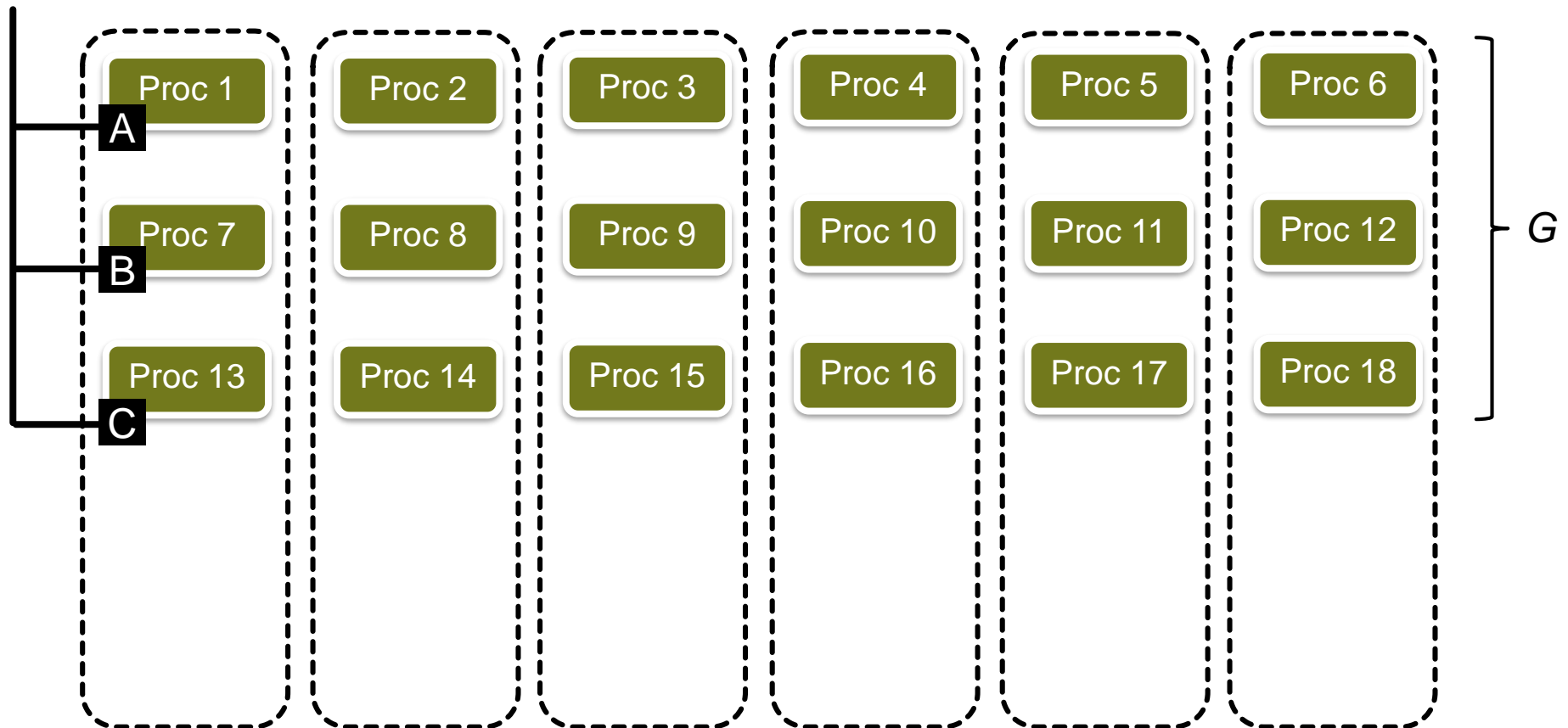
- Step 1: groups of processes:
 - Divide processes into groups of size G each



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each

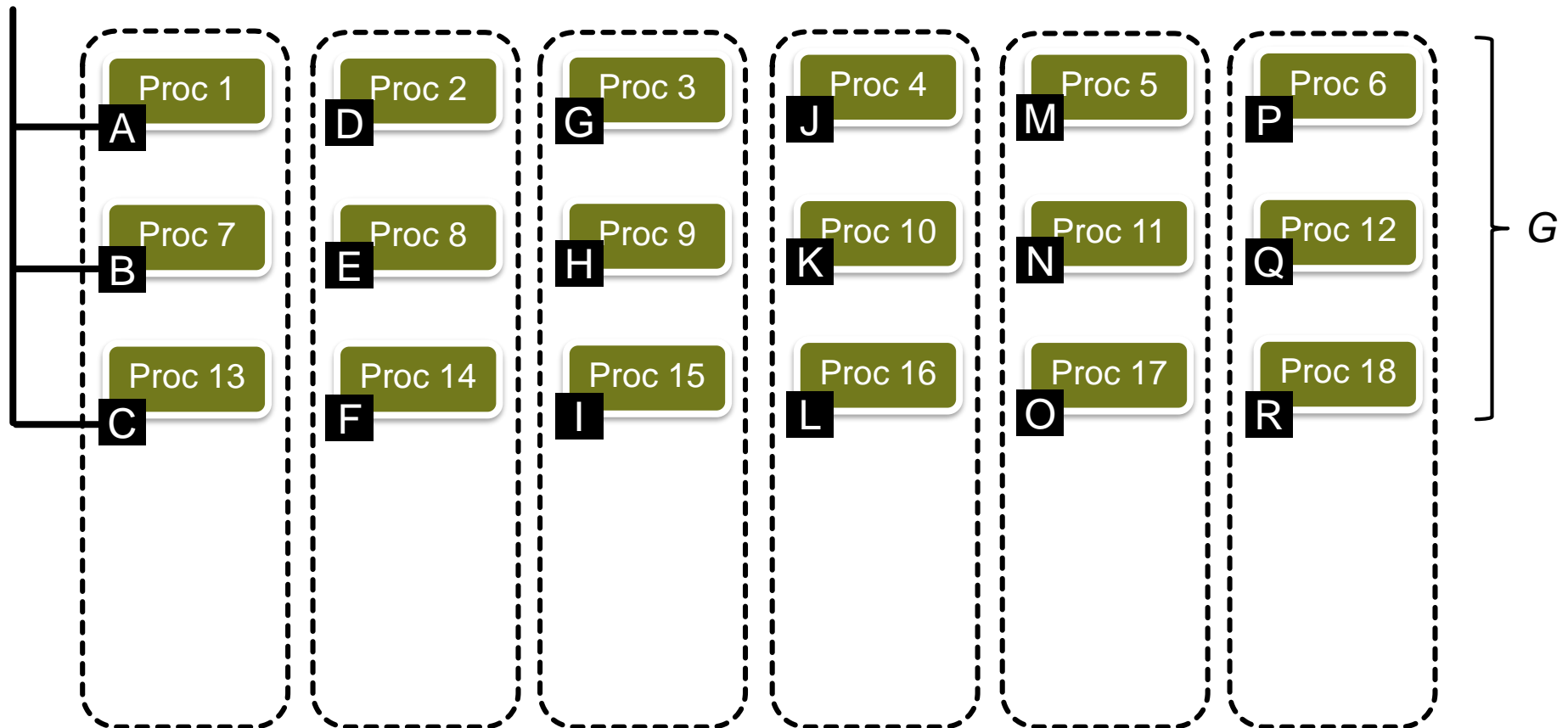
Application data



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

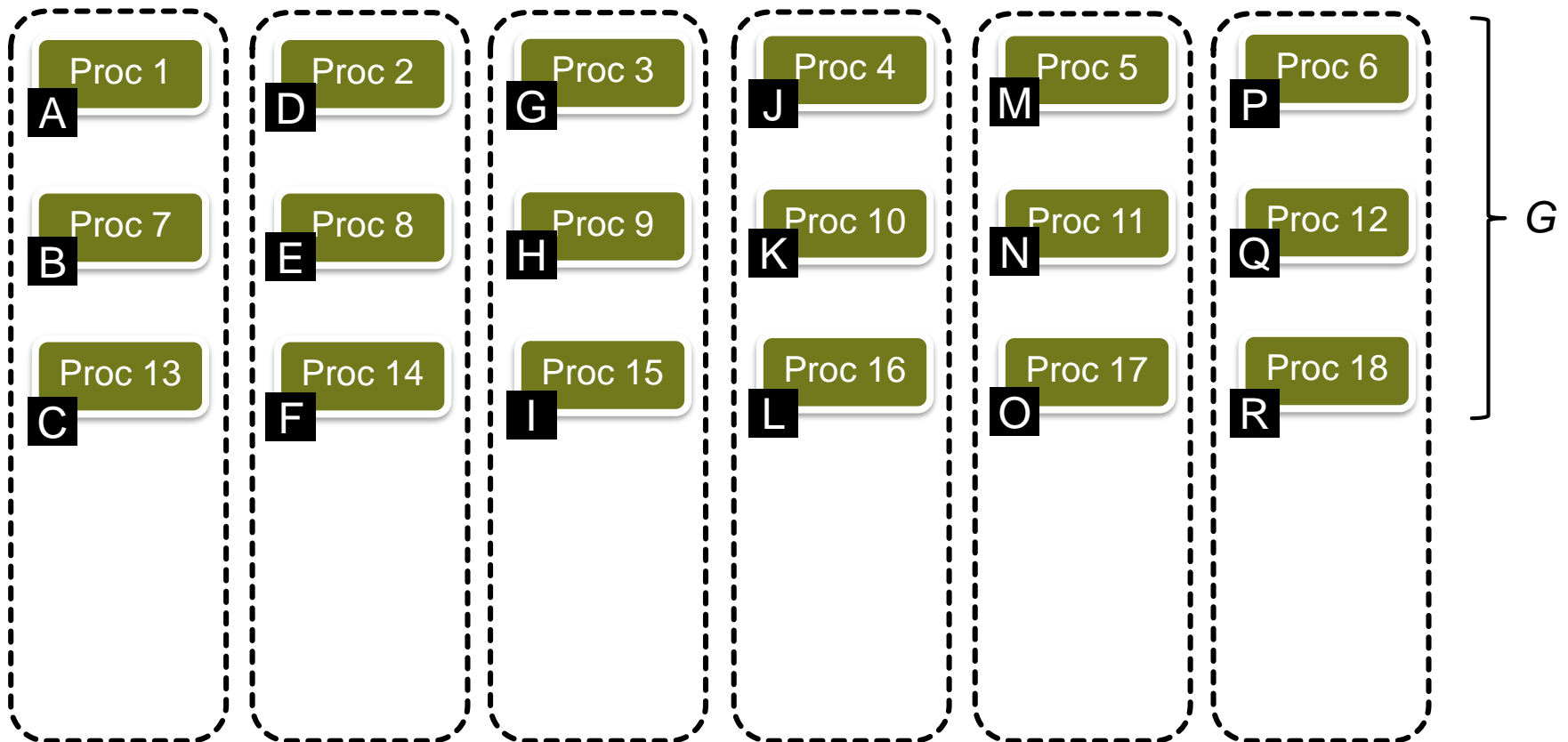
- Step 1: groups of processes:
 - Divide processes into groups of size G each

Application data



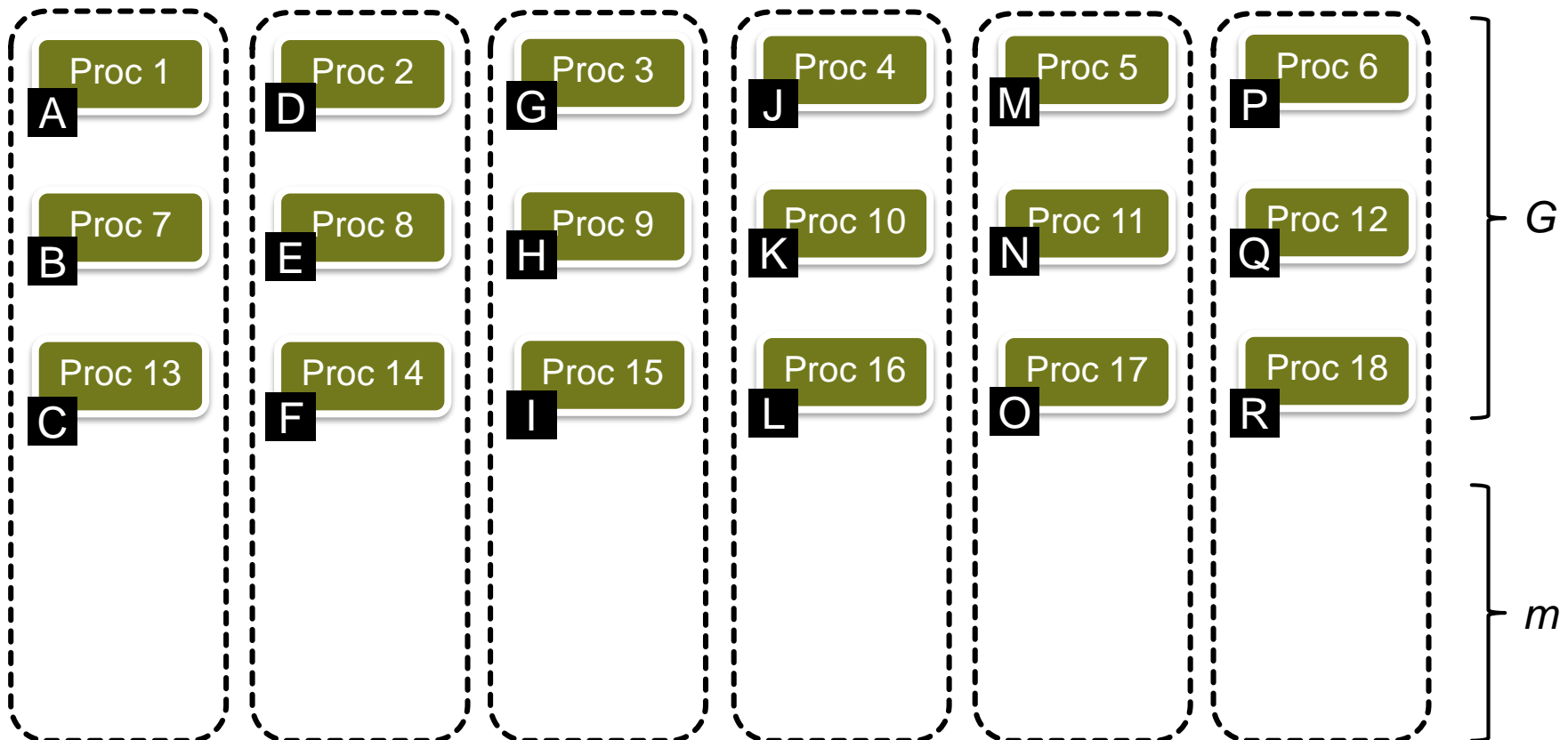
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each



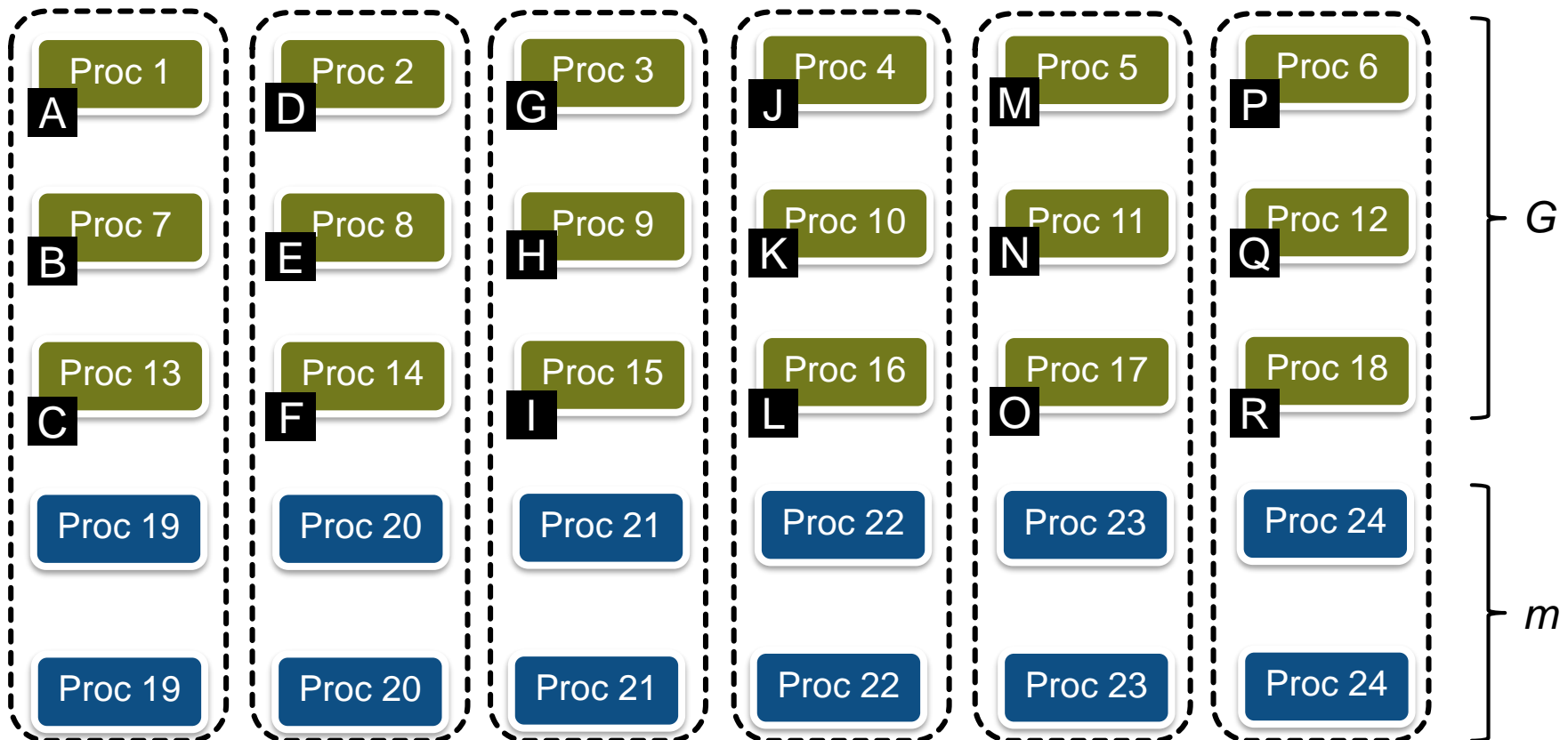
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



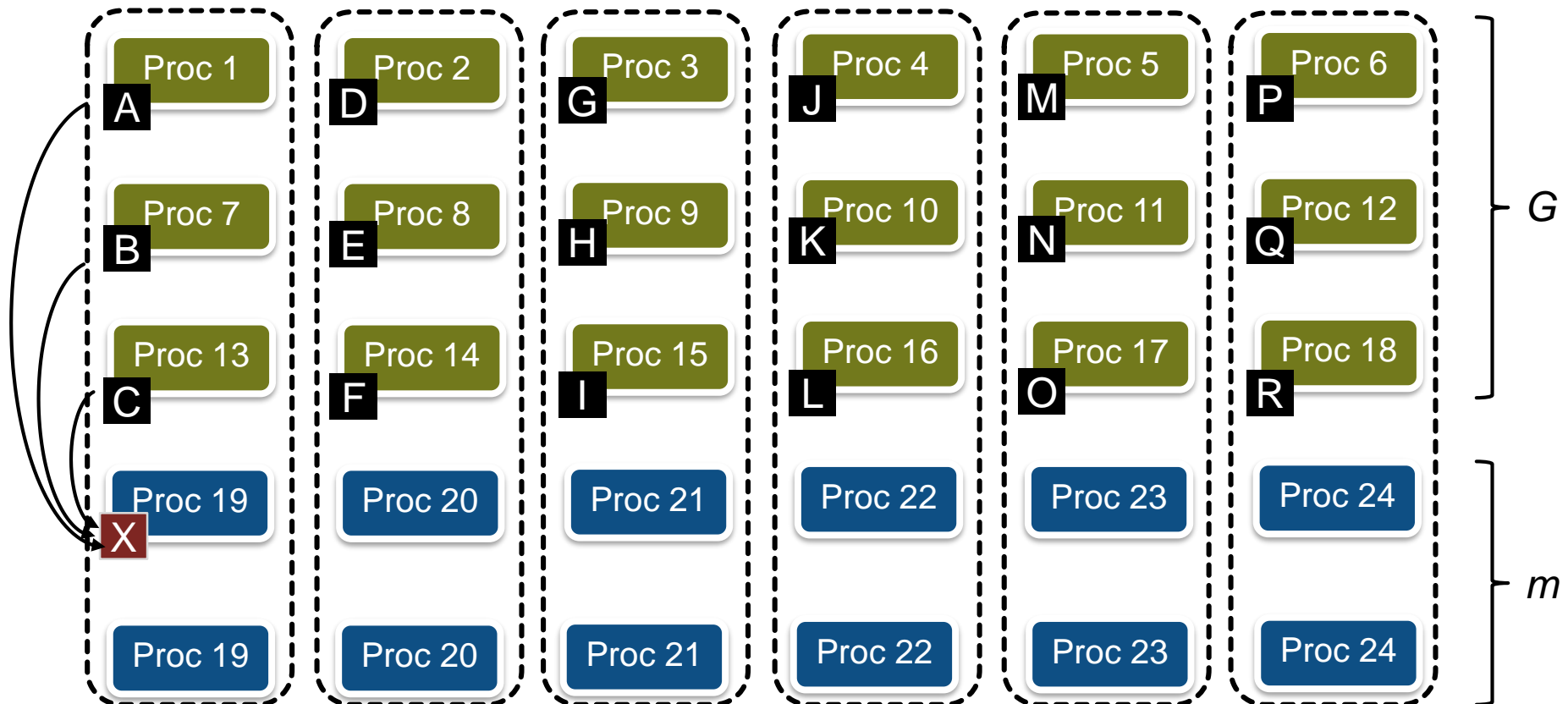
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



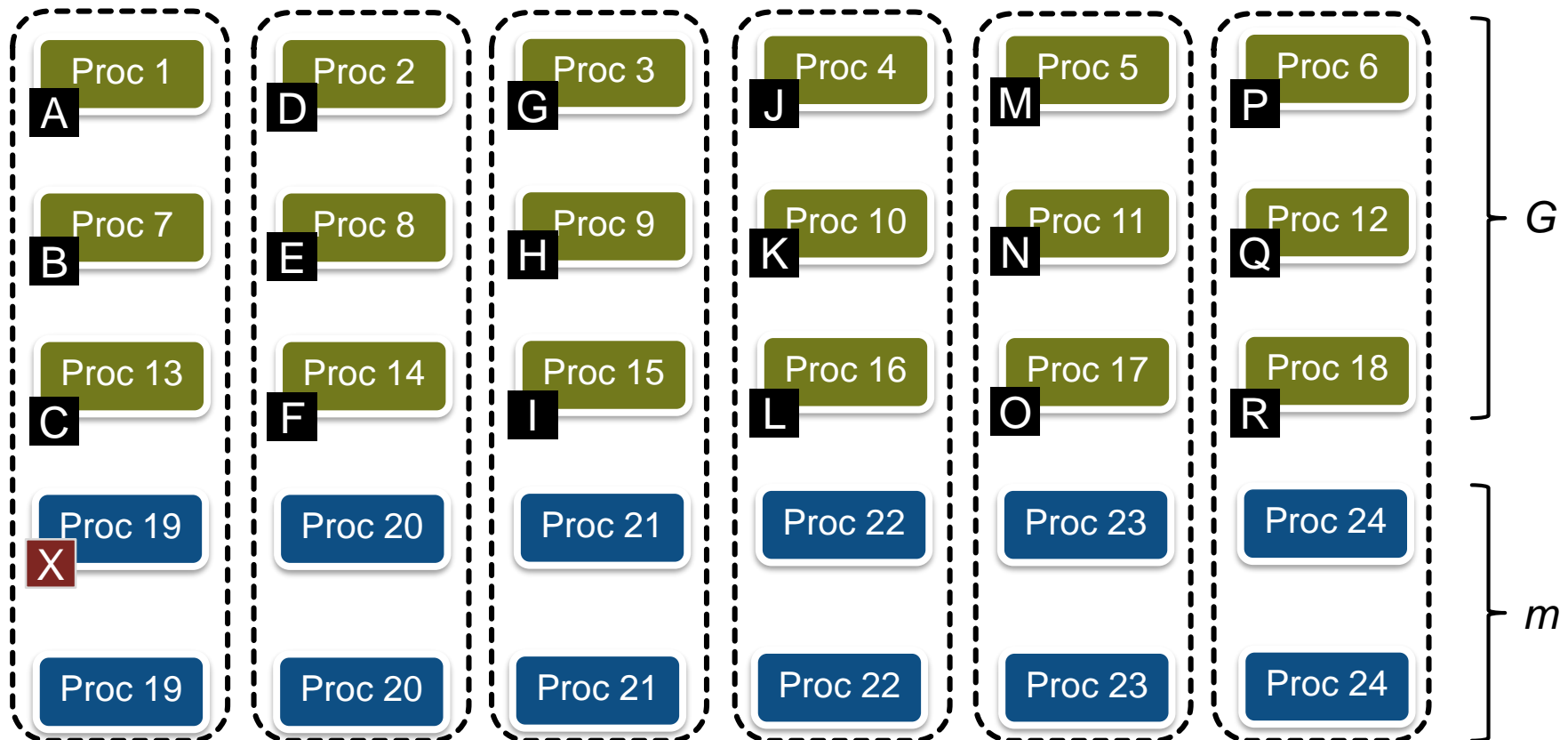
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



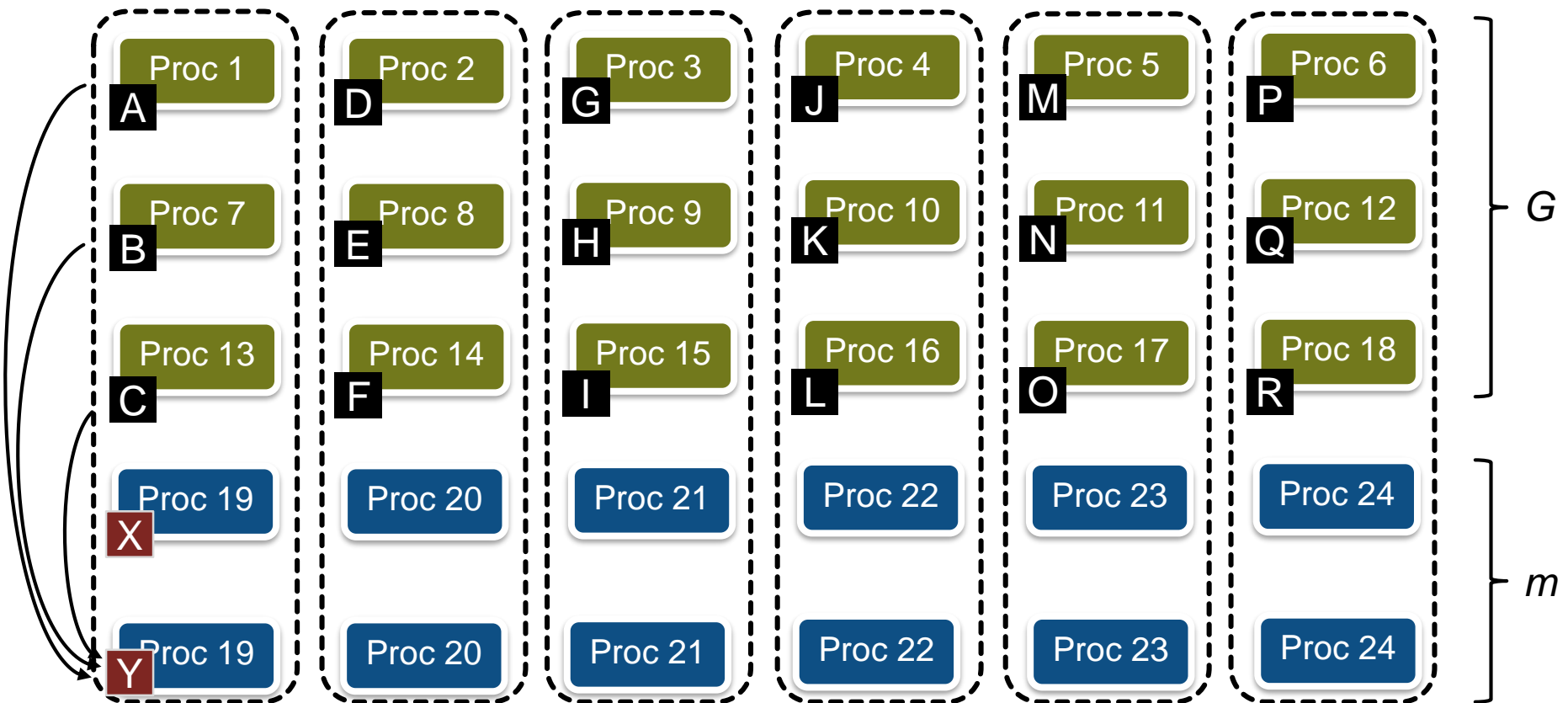
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



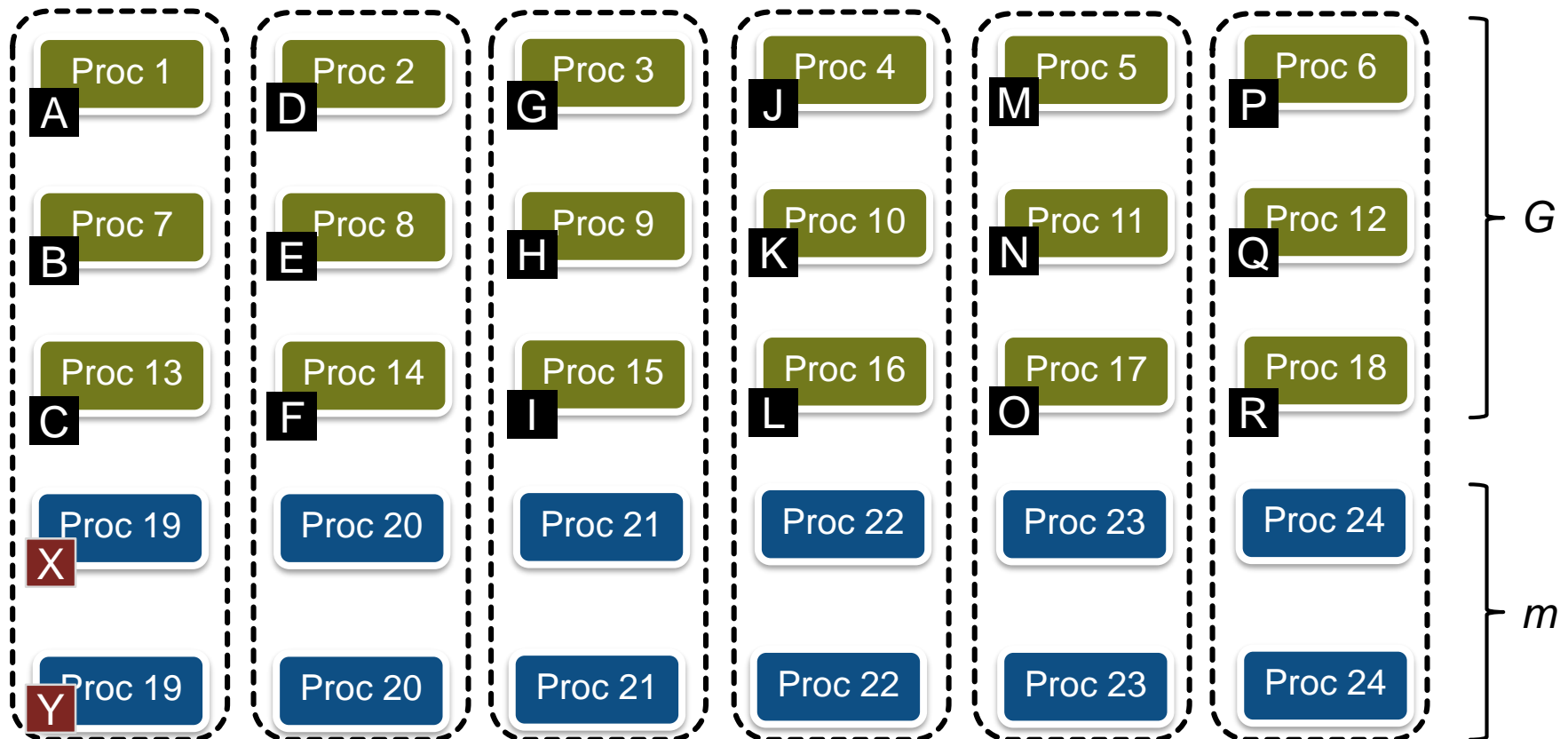
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

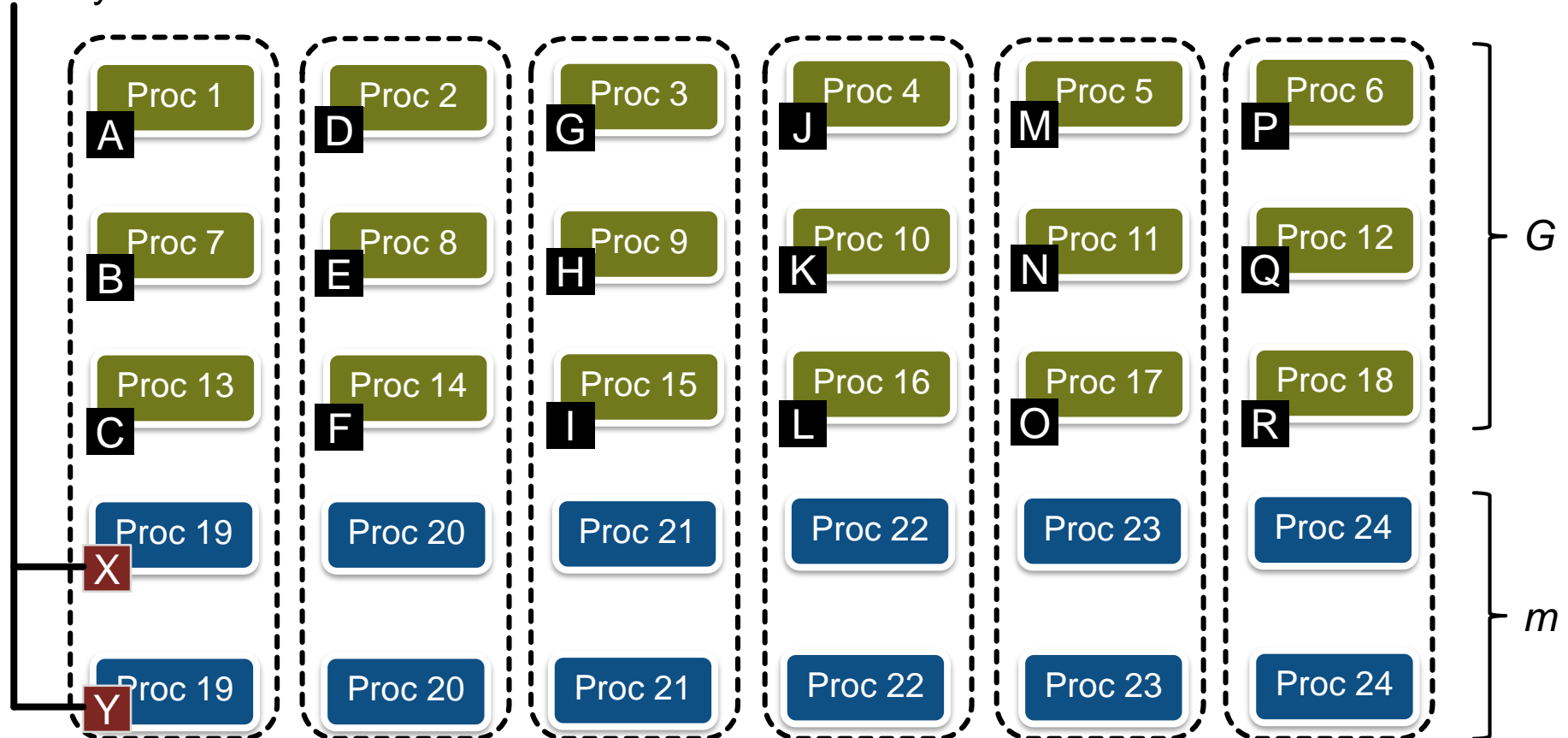
- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

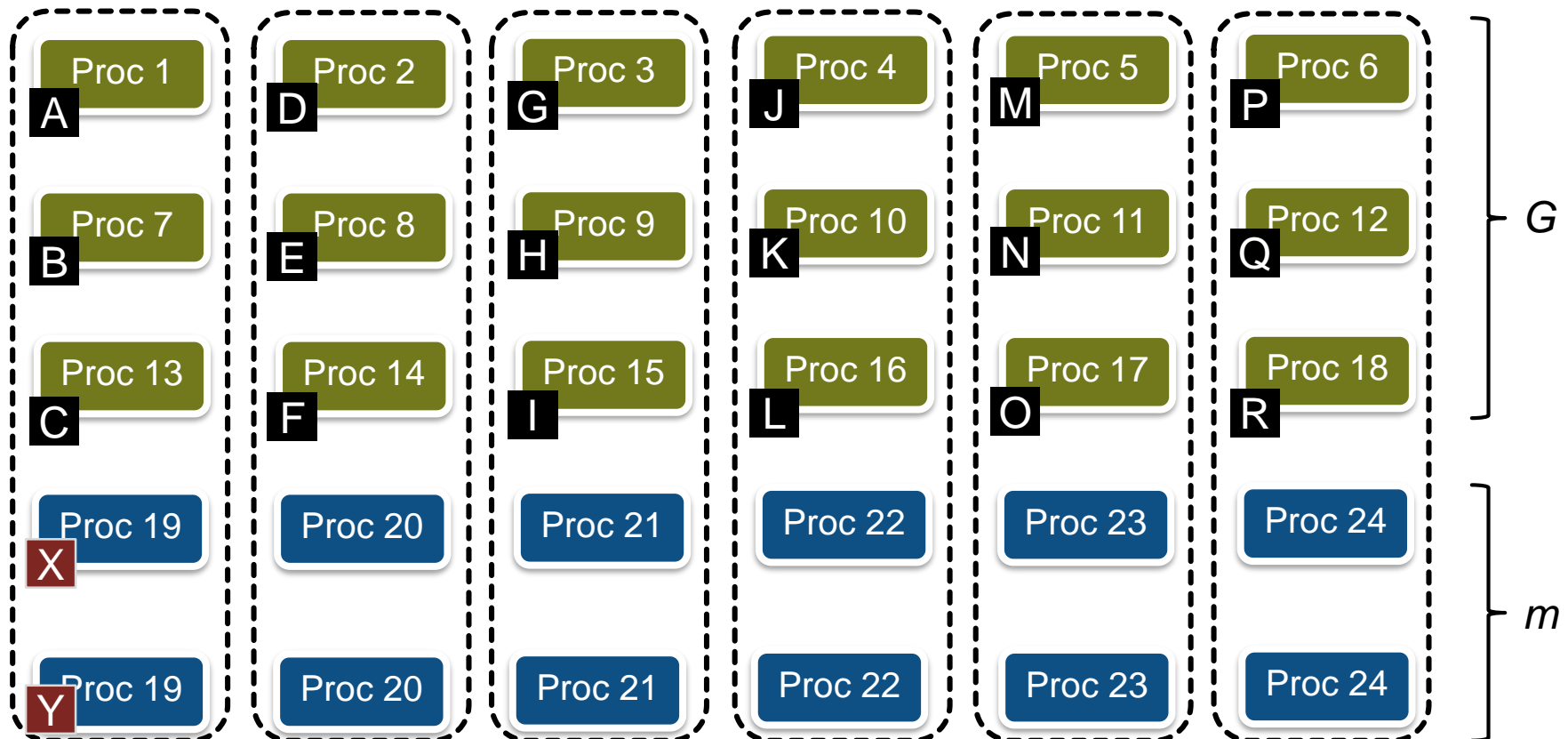
- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data

Parity data



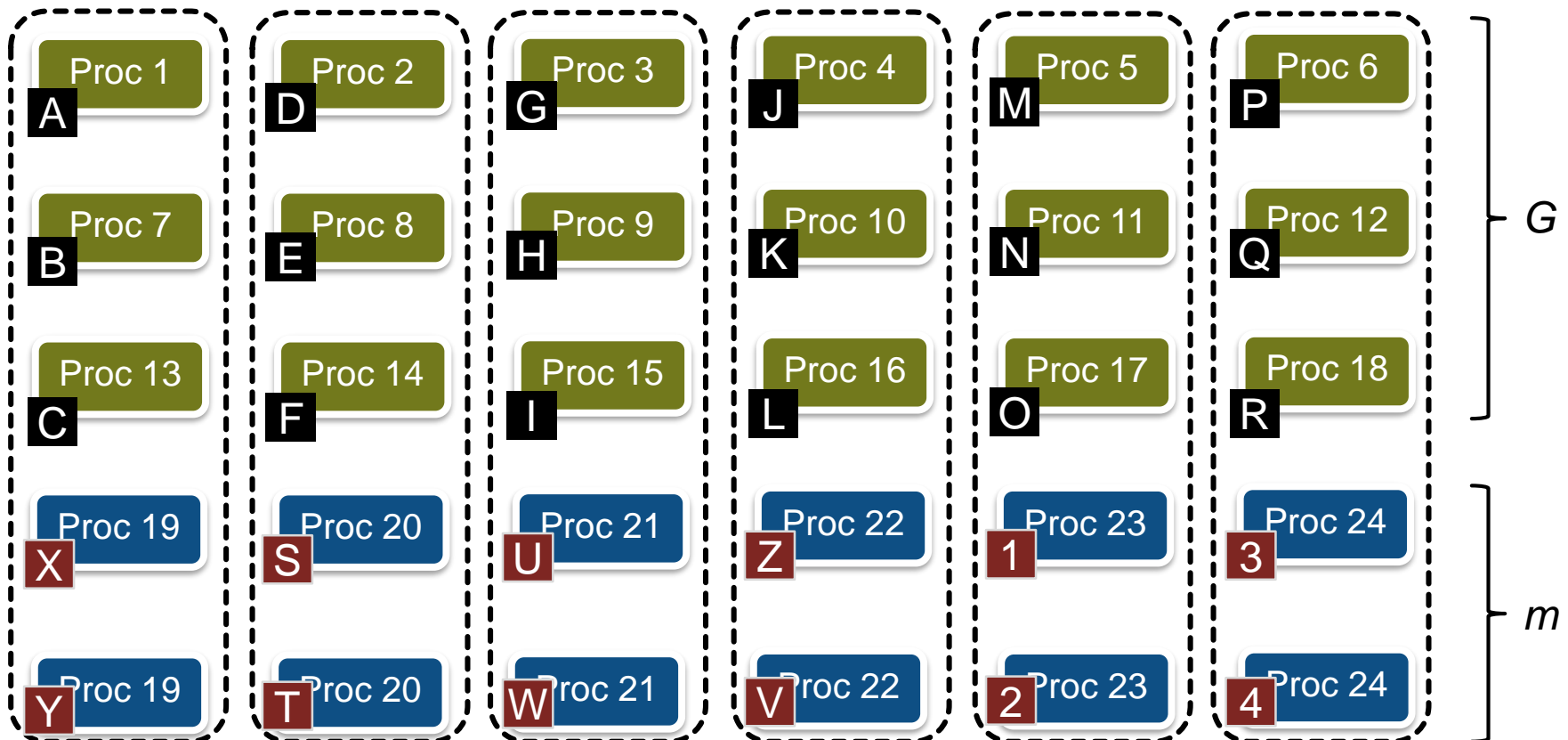
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



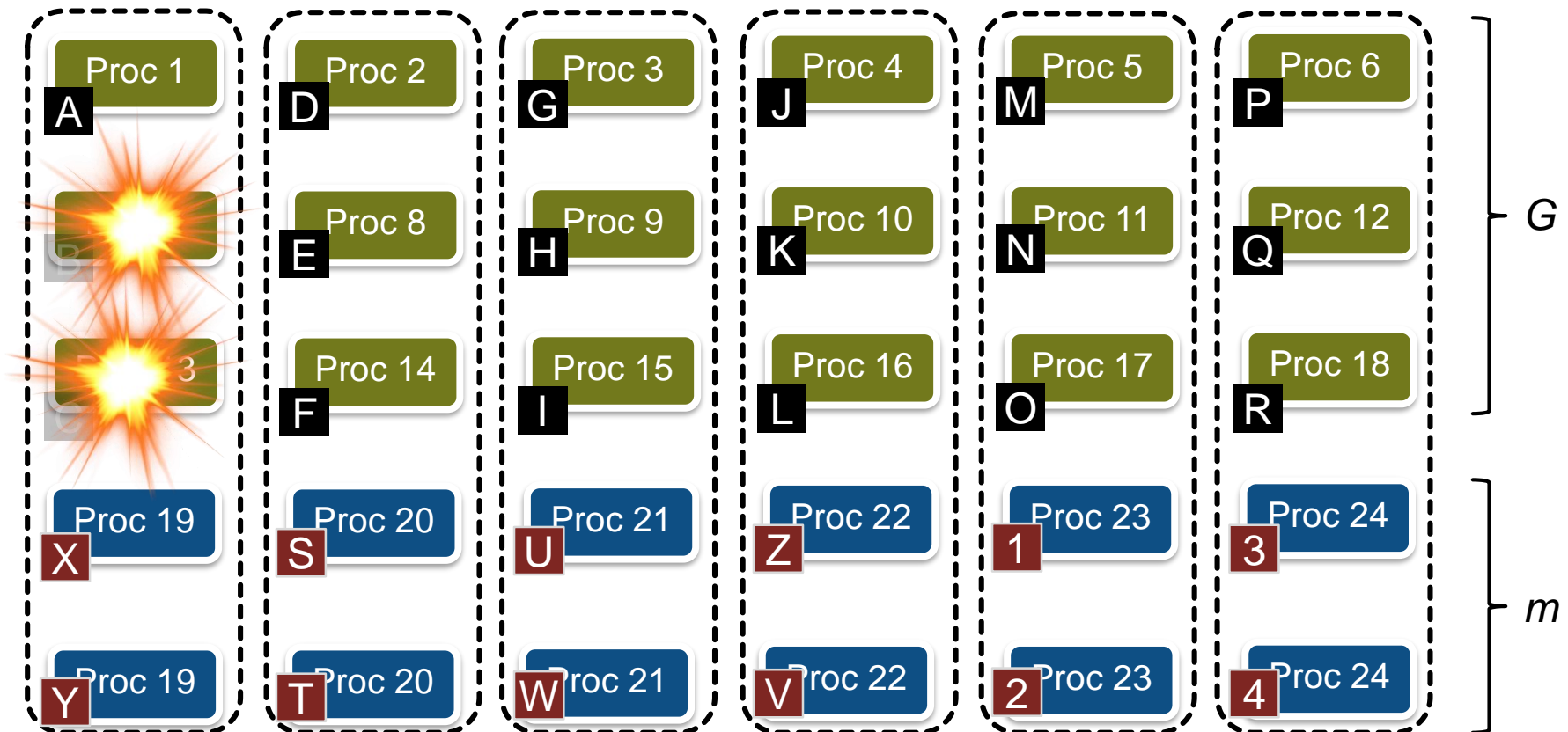
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



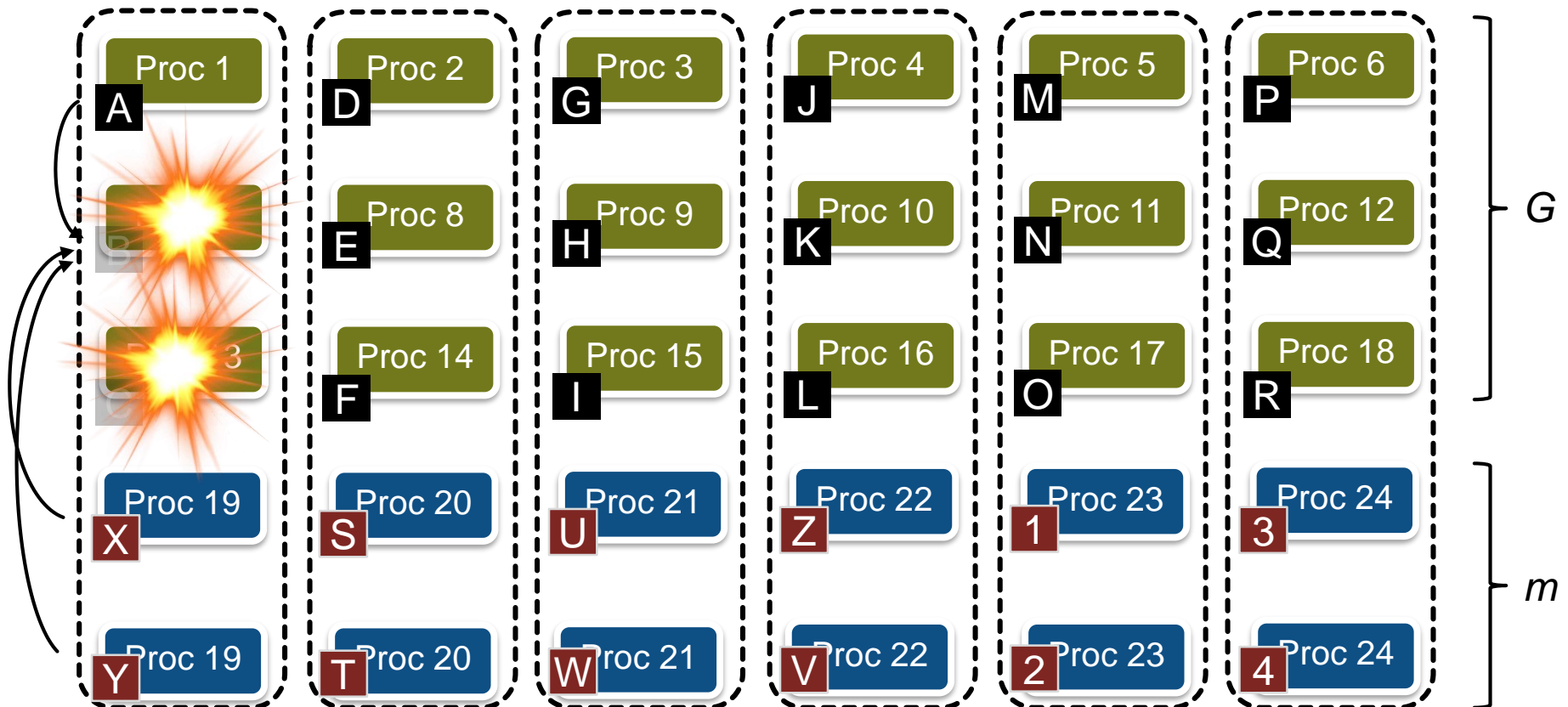
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



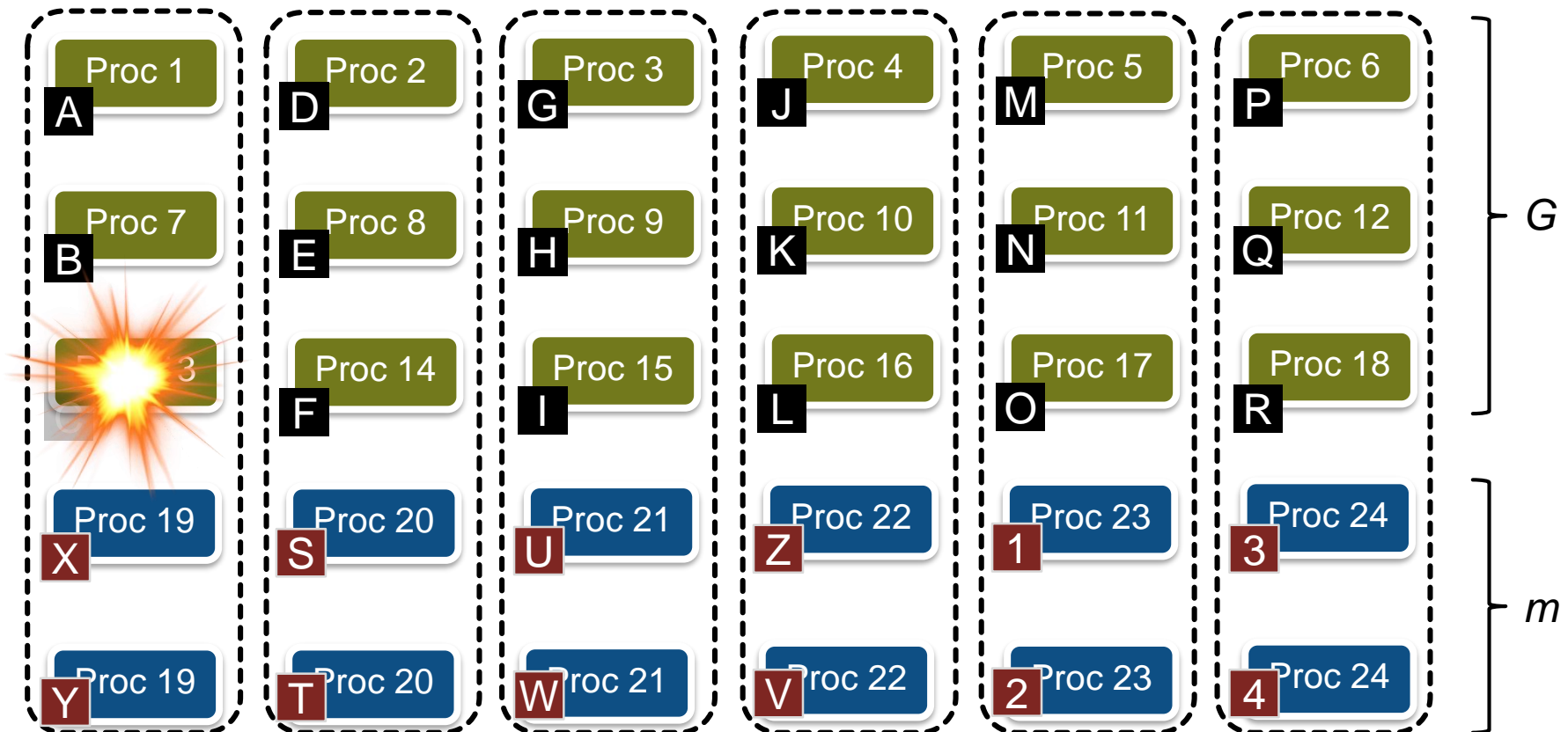
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



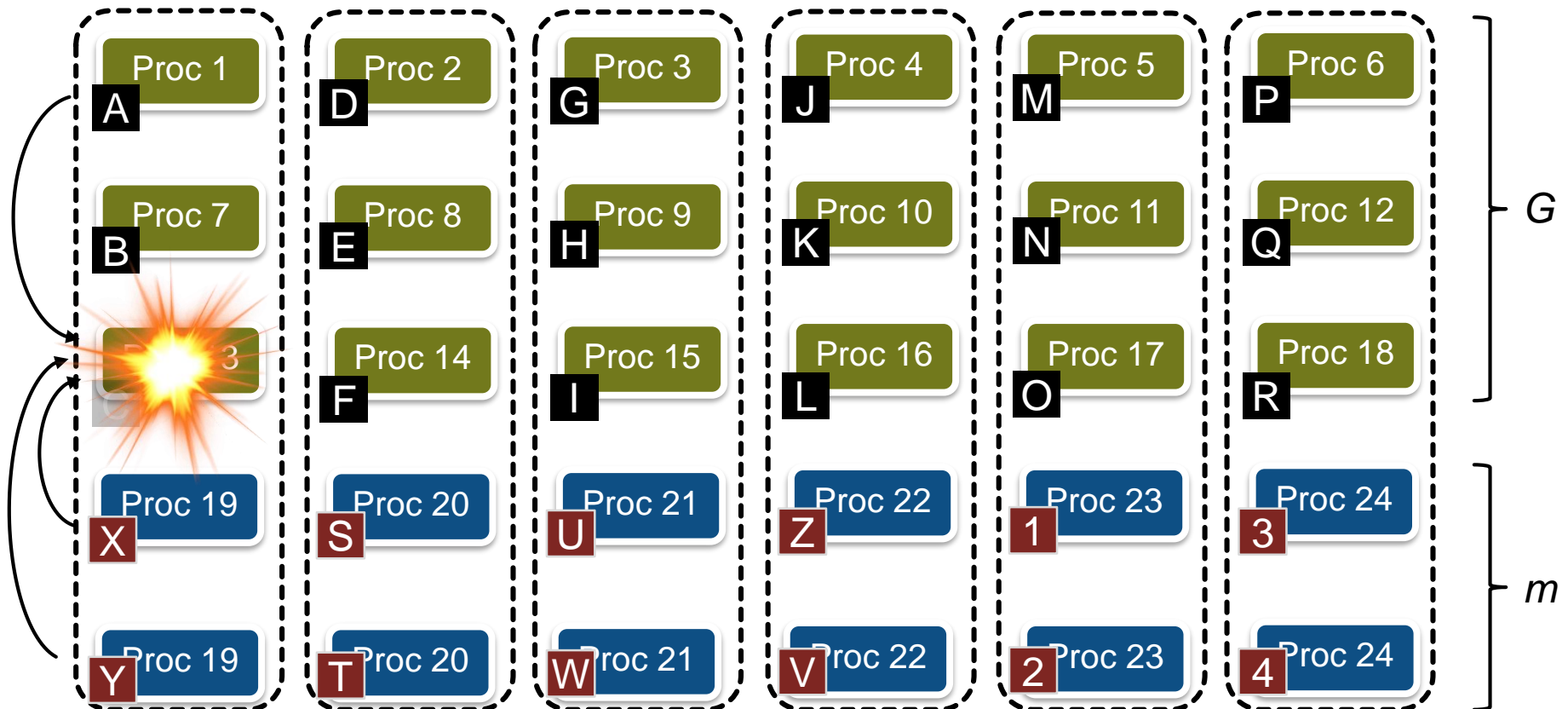
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



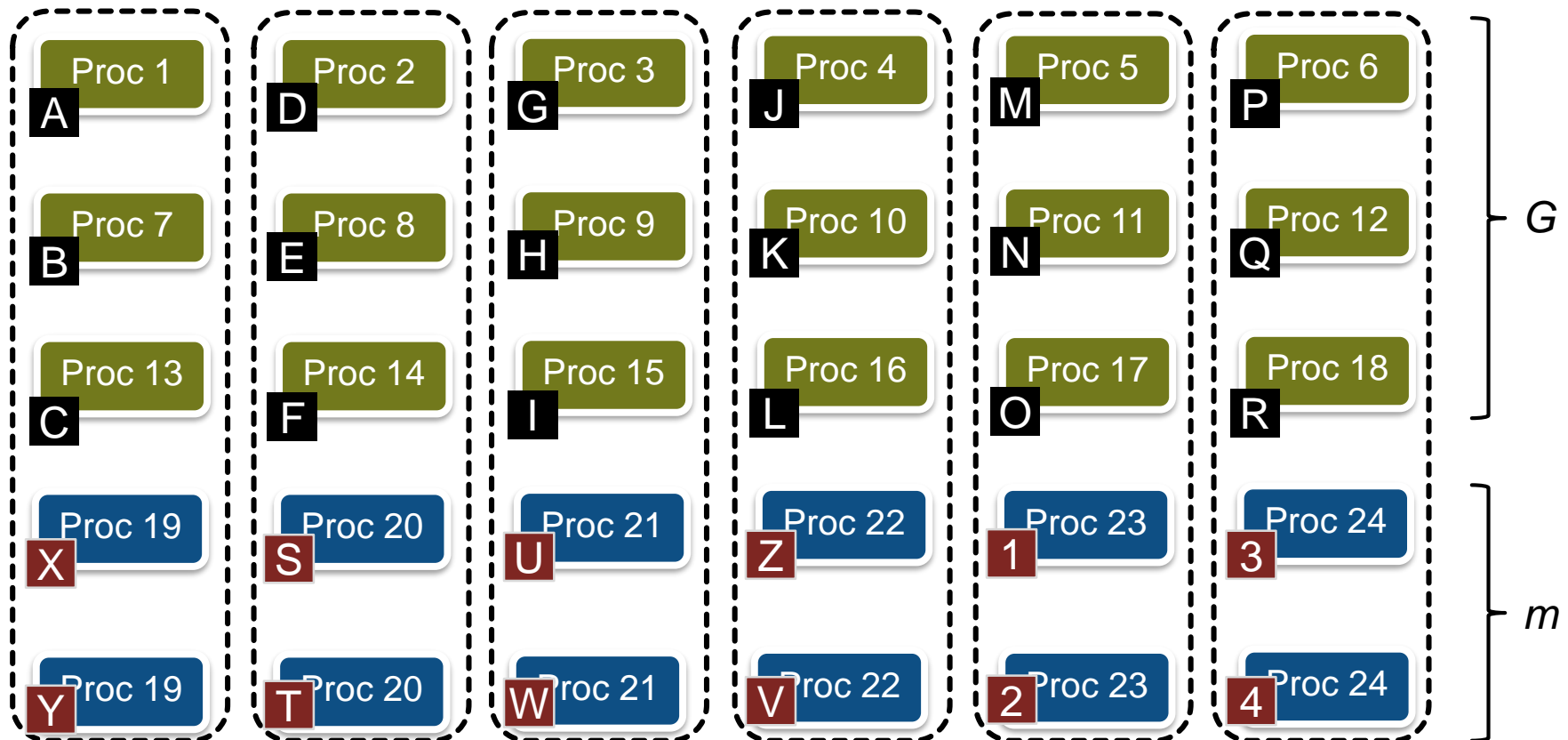
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



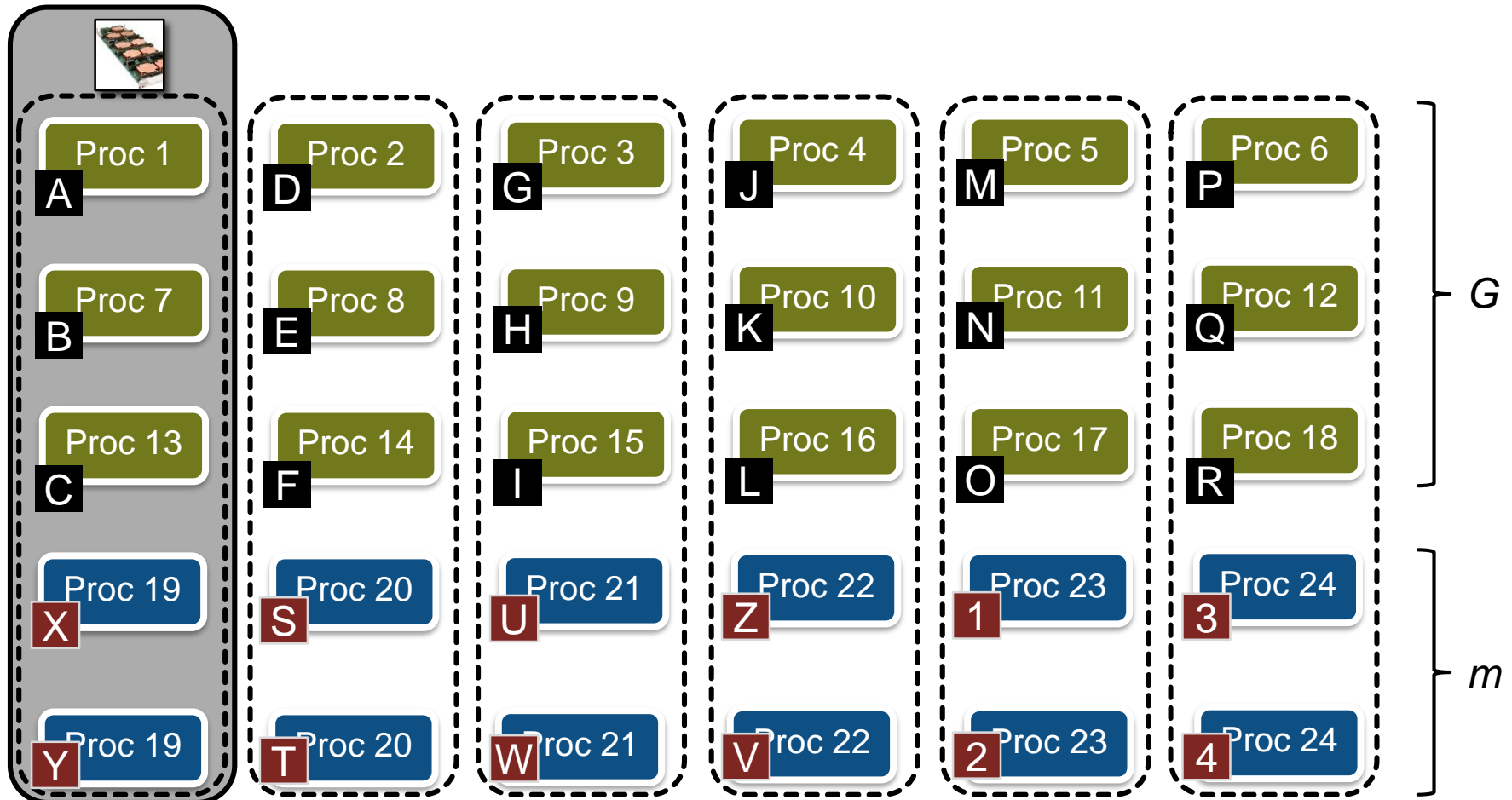
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



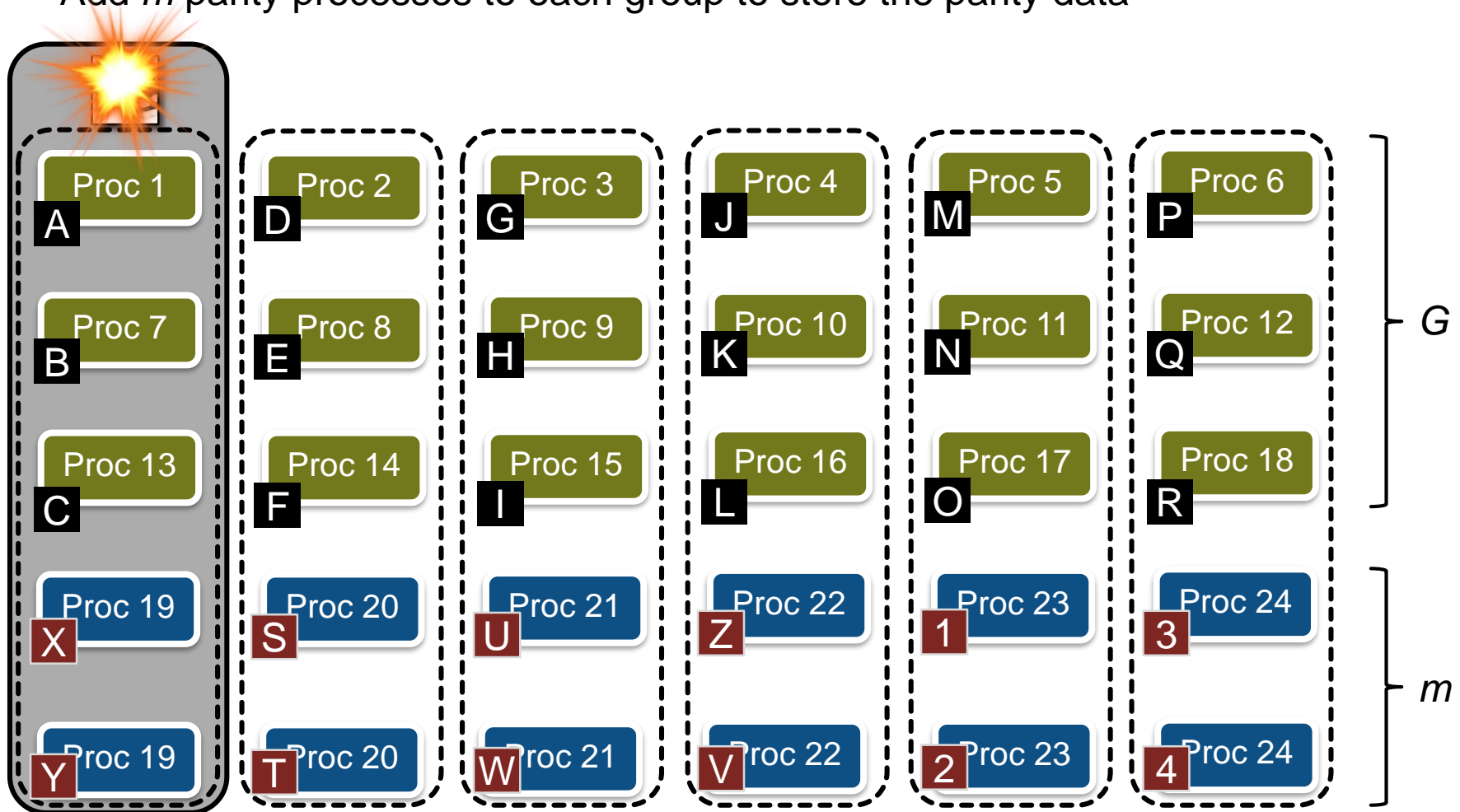
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data



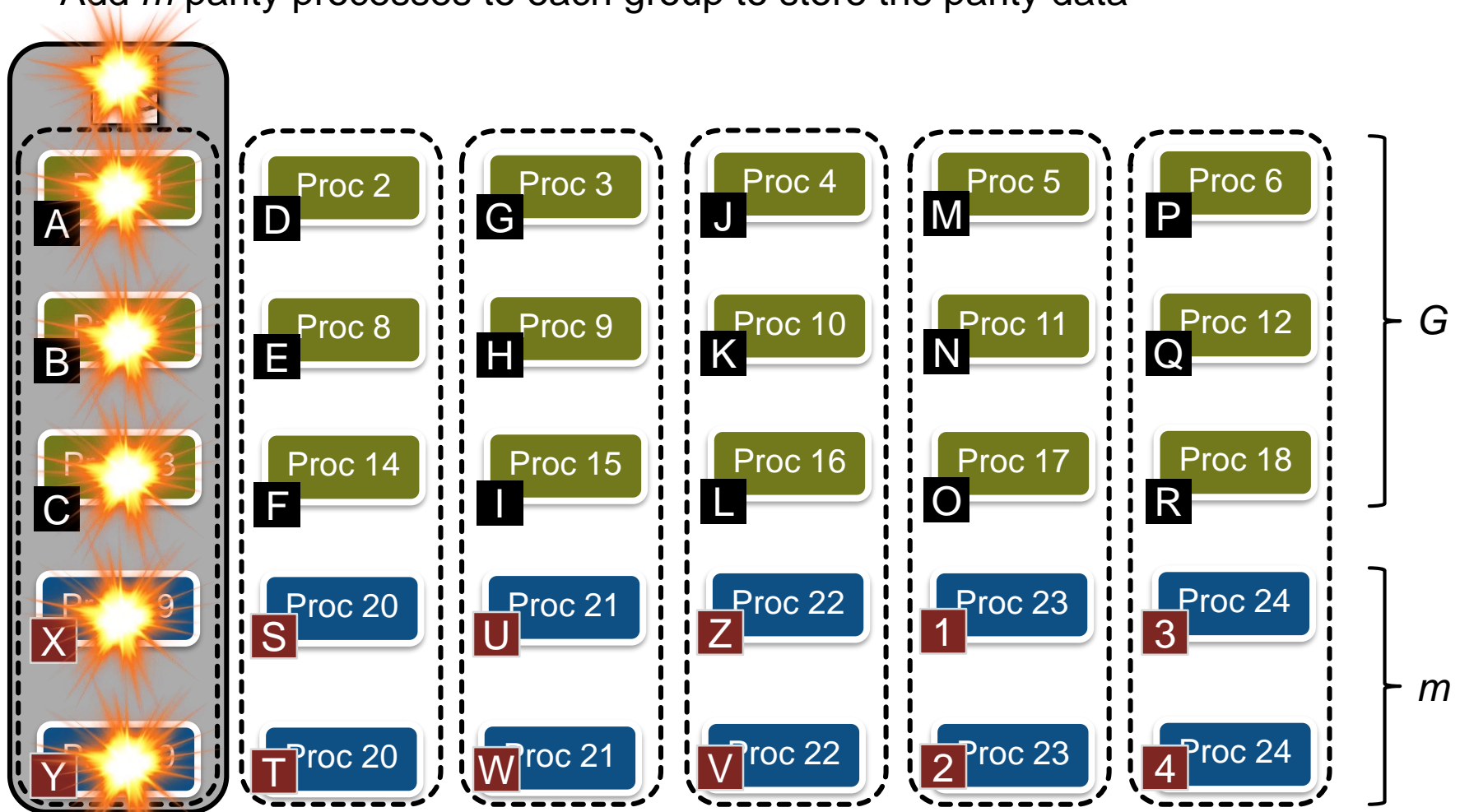
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data

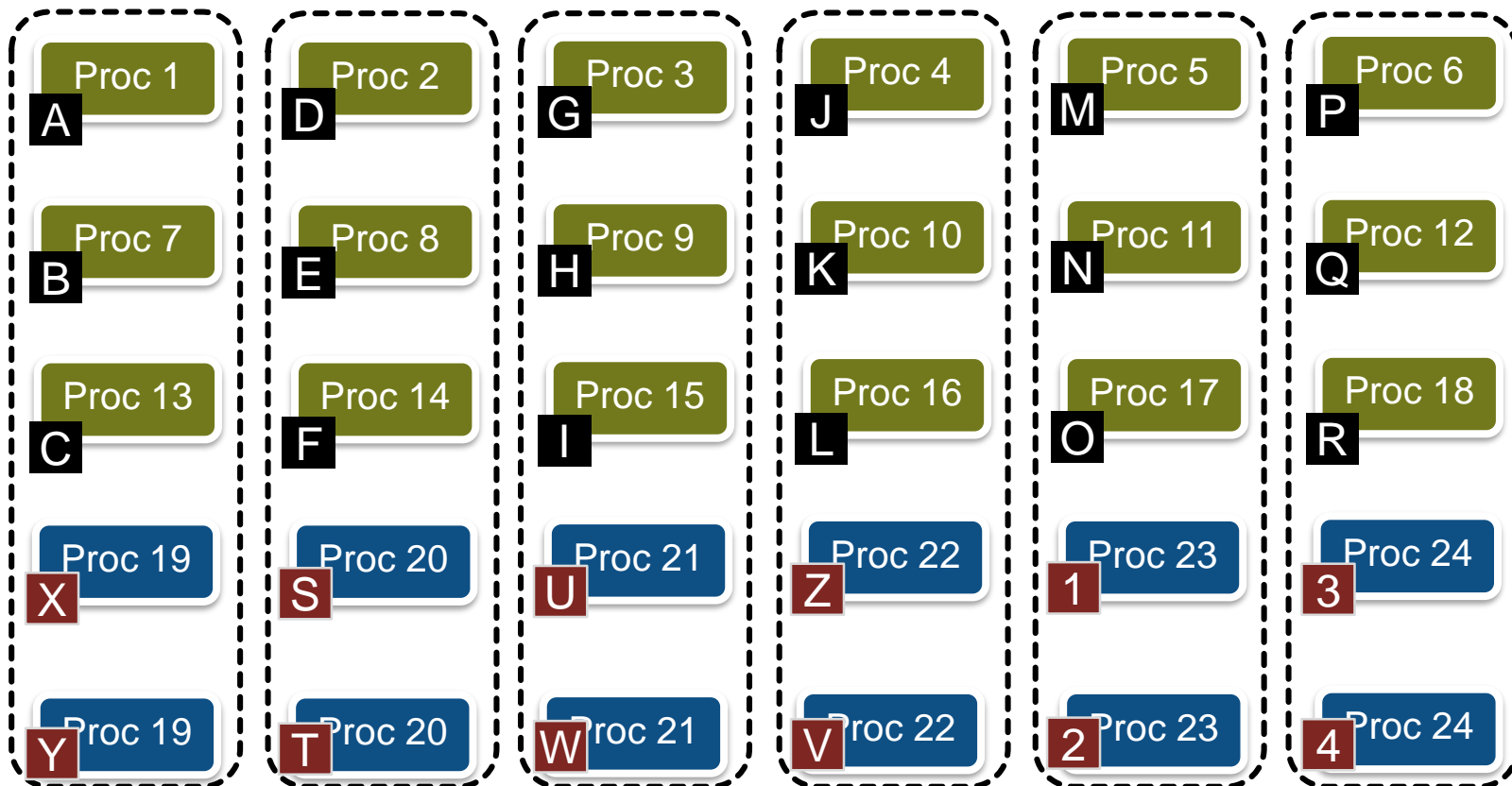


EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 1: groups of processes:
 - Divide processes into groups of size G each
 - Add m parity processes to each group to store the parity data

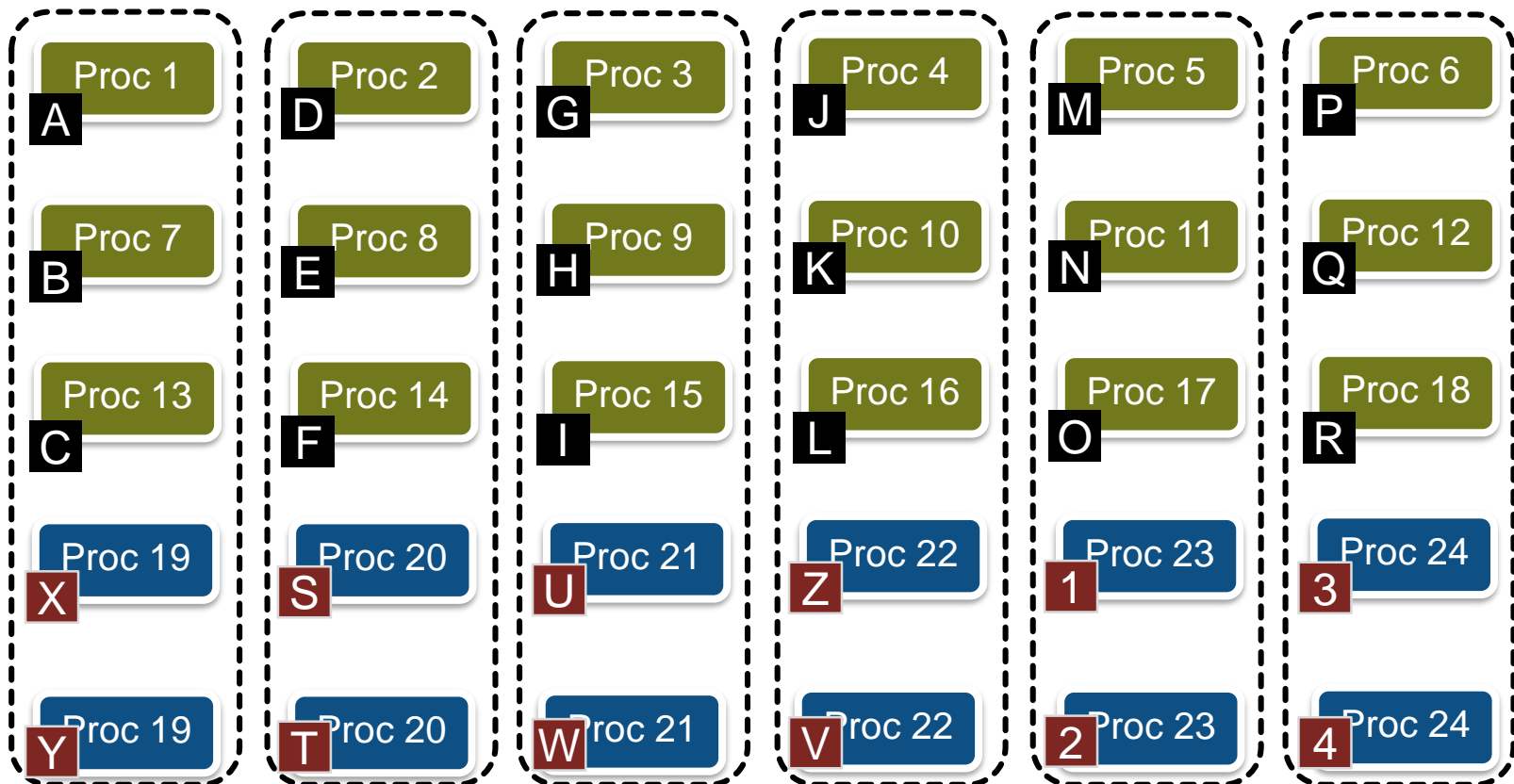


EXTENDING THE PROTOCOLS FOR MORE RESILIENCE



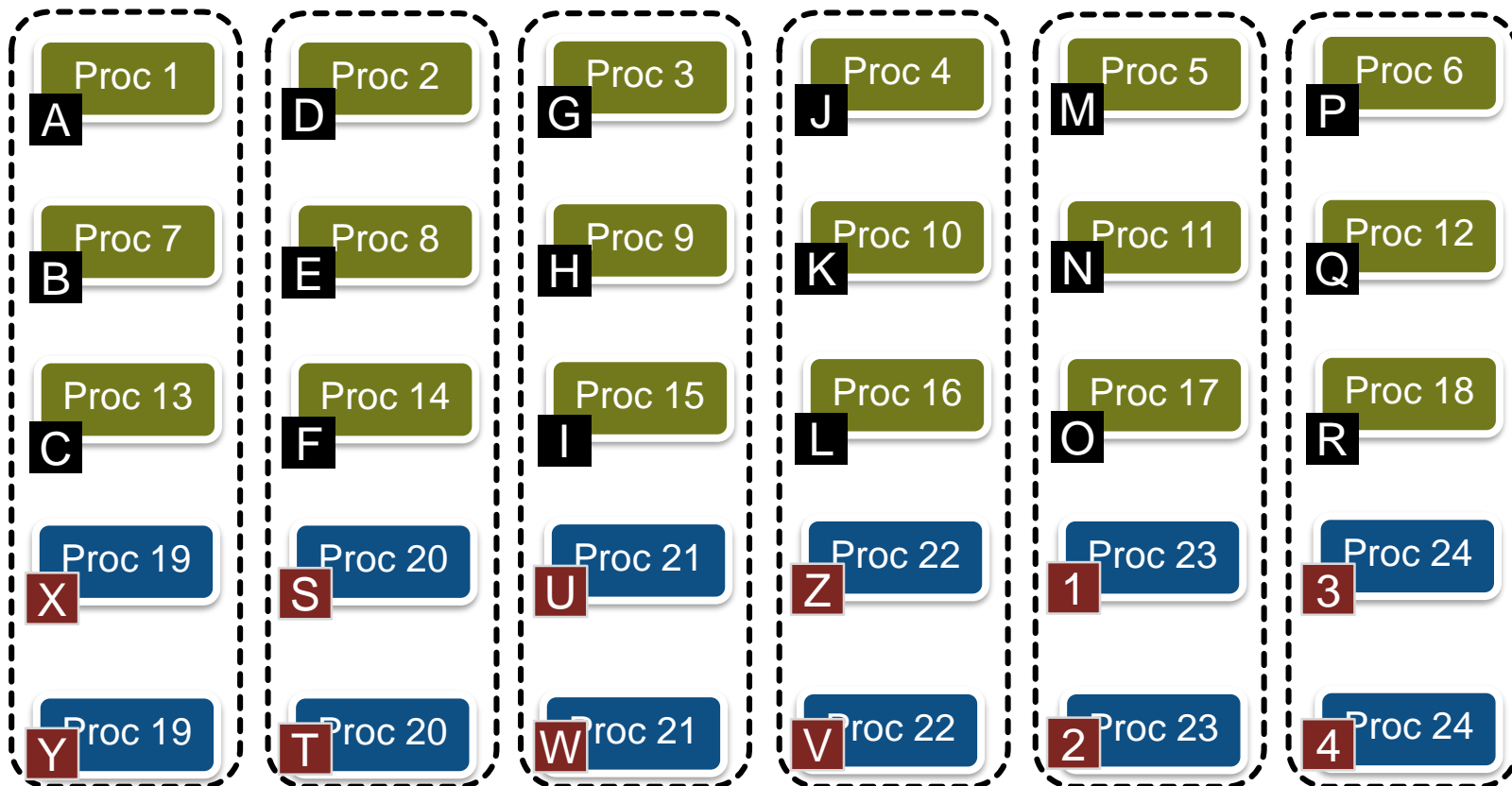
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:



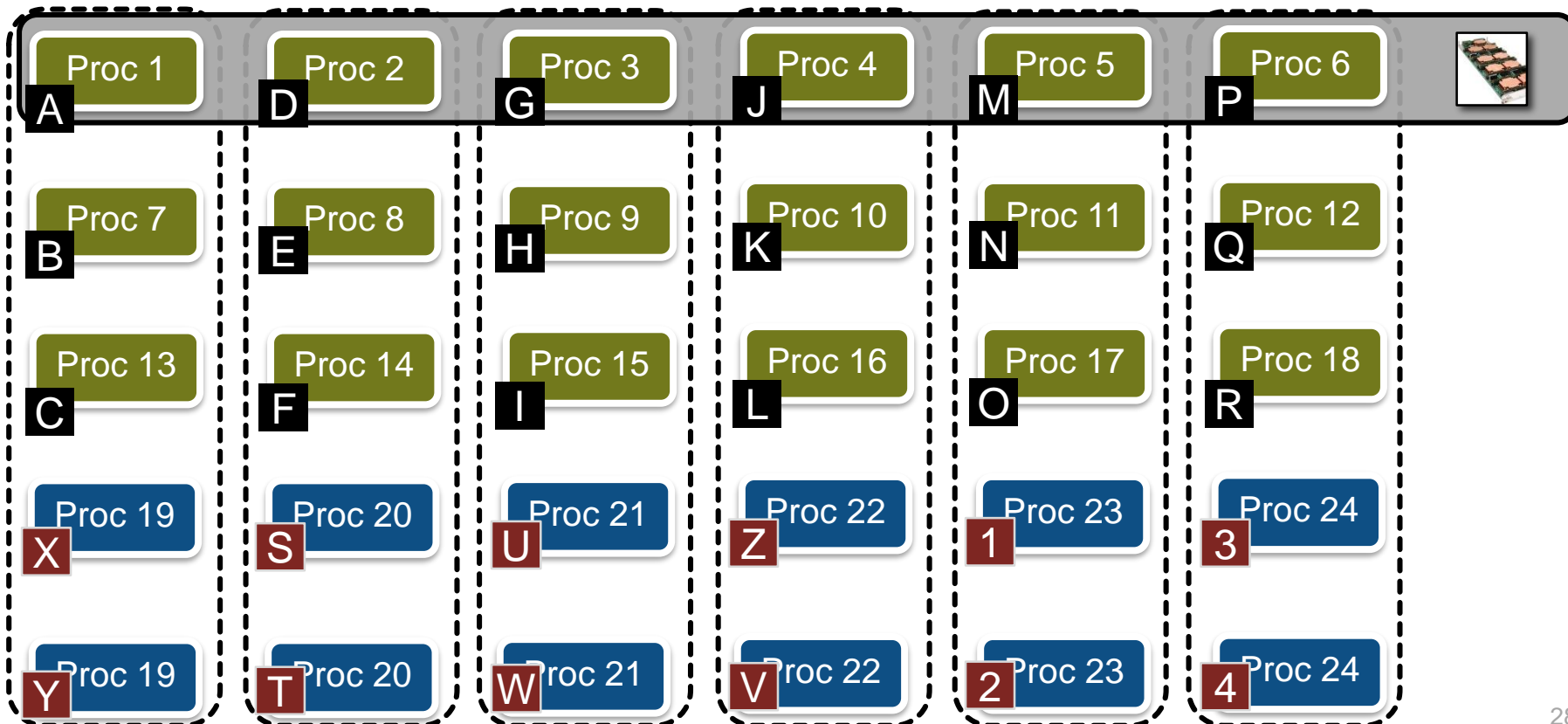
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...



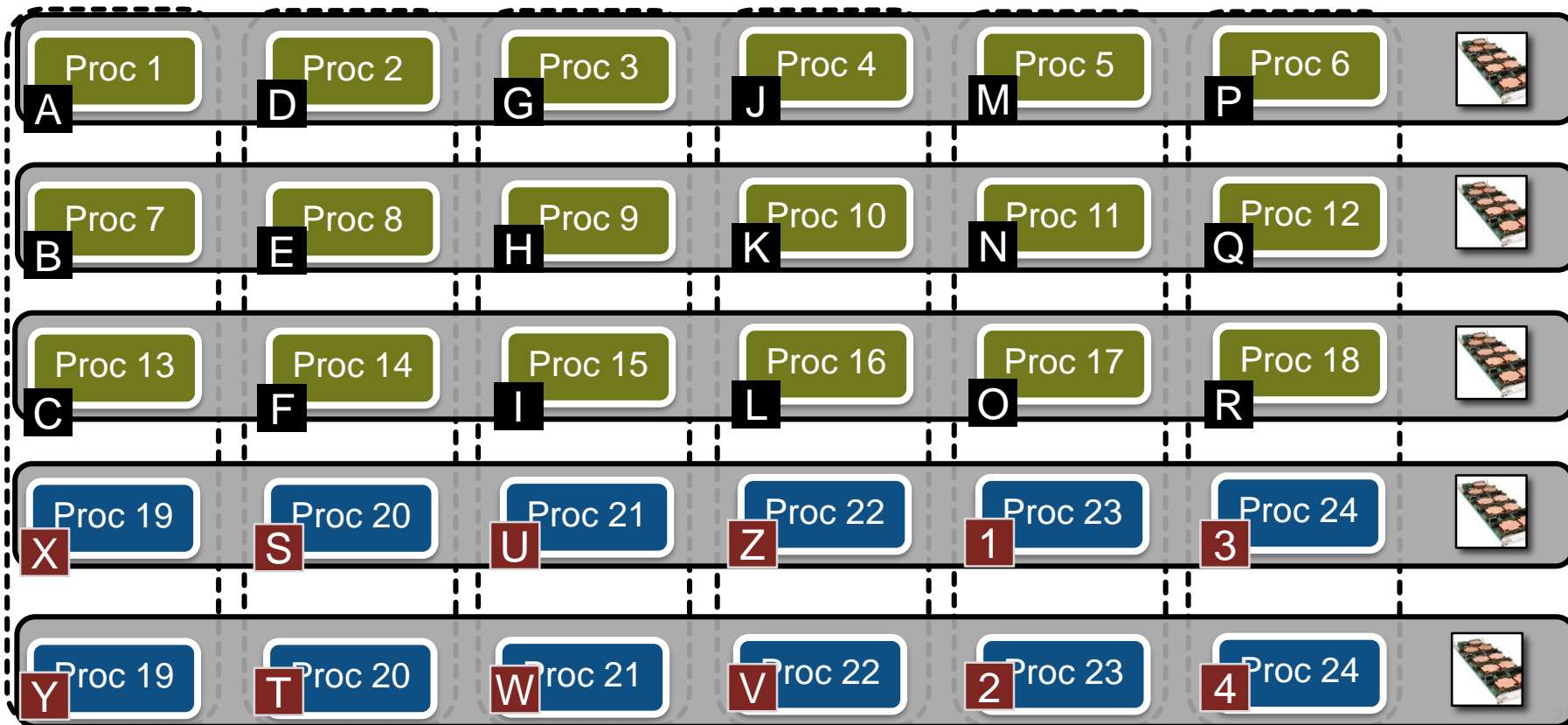
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...



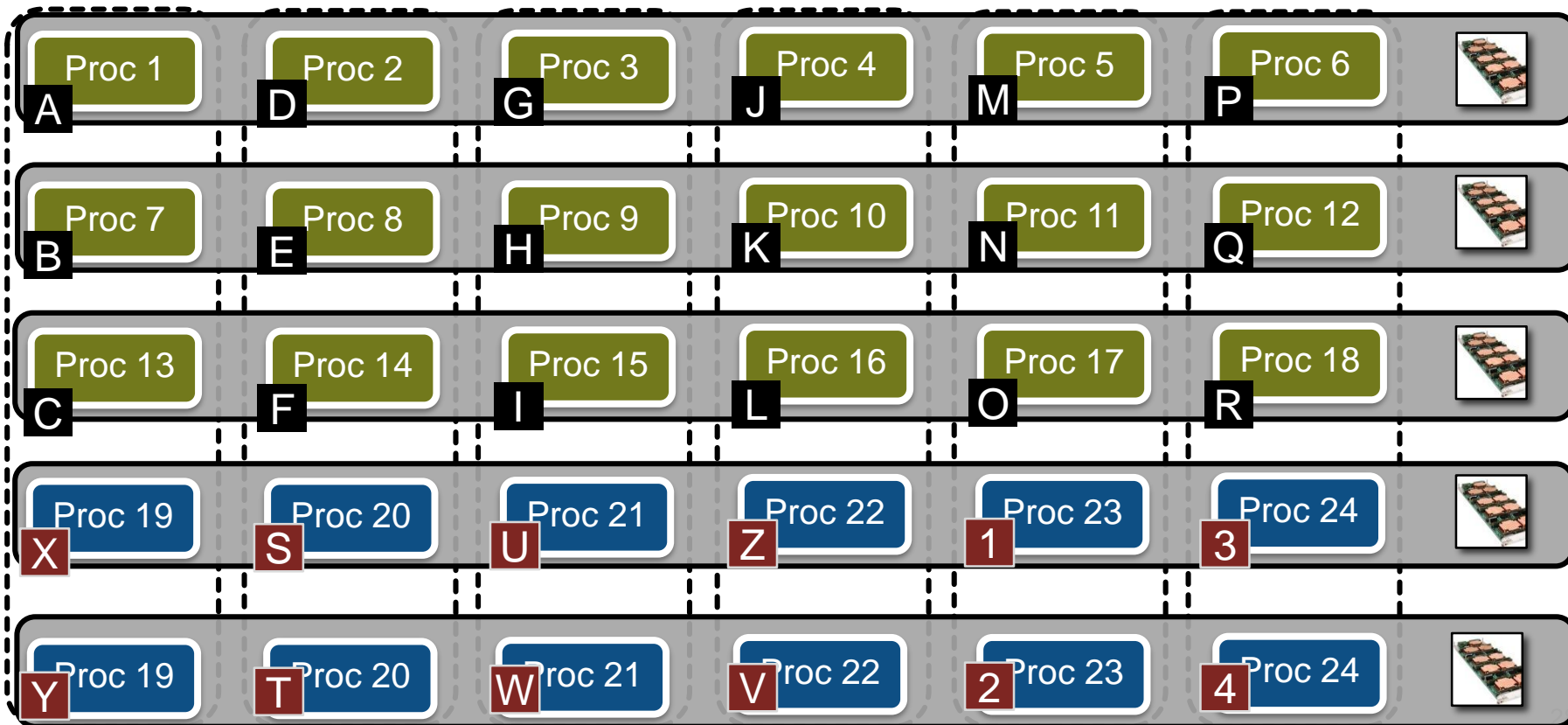
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...



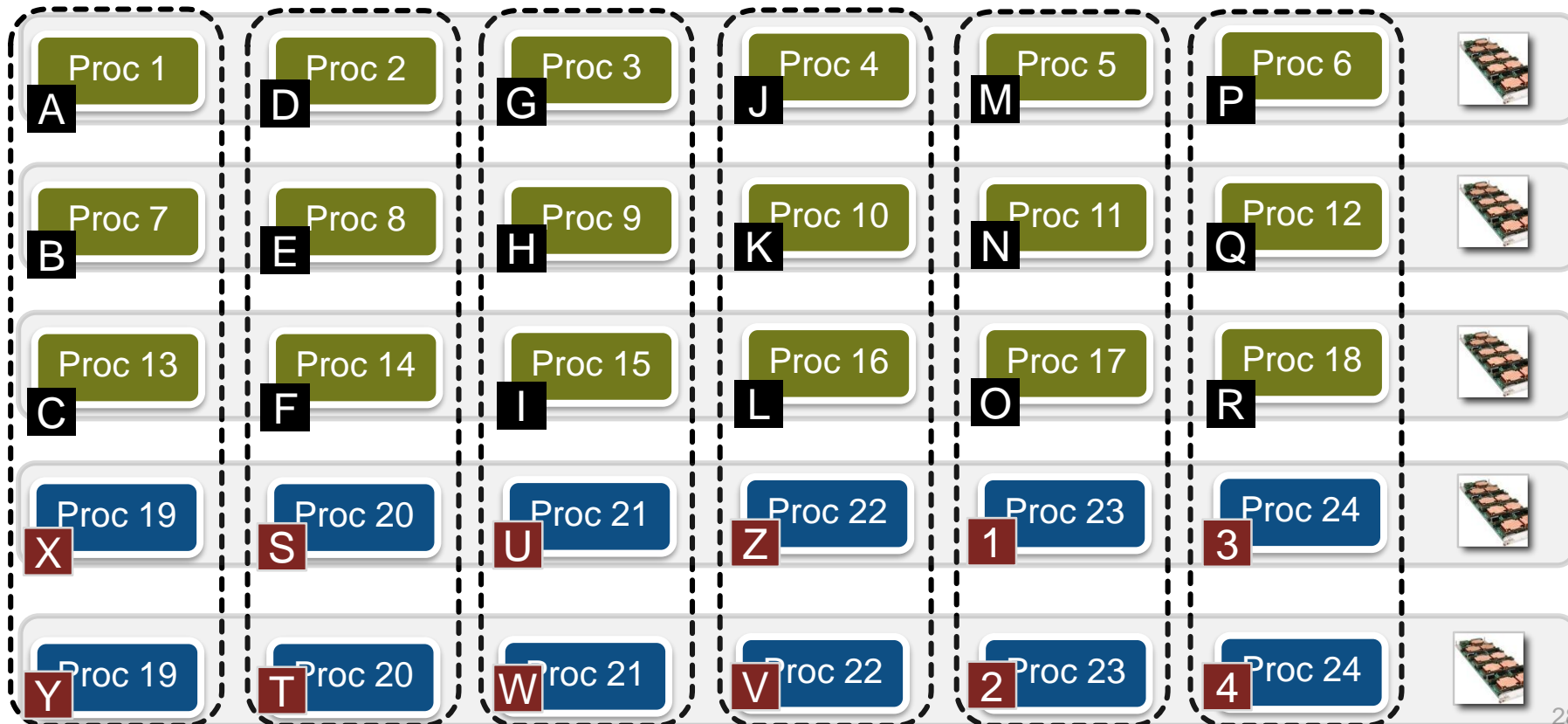
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



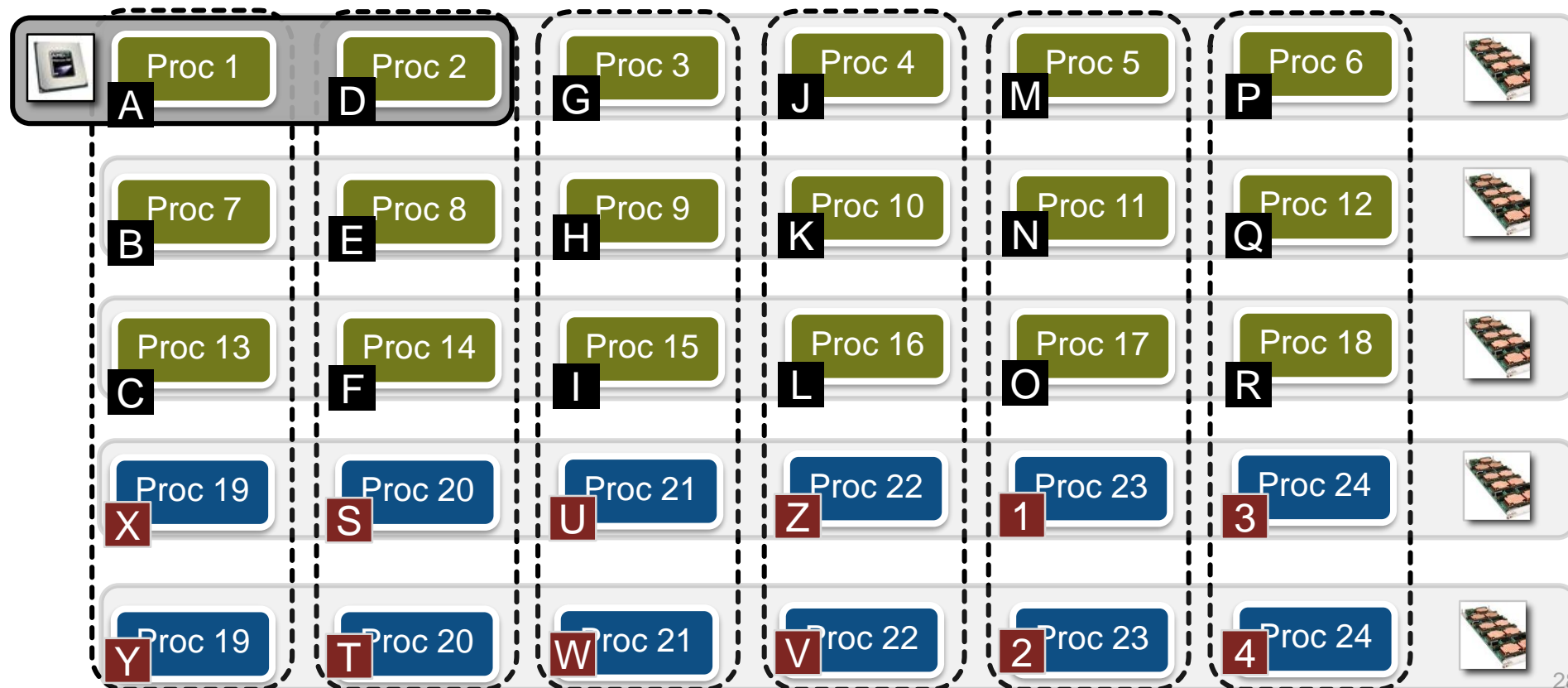
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



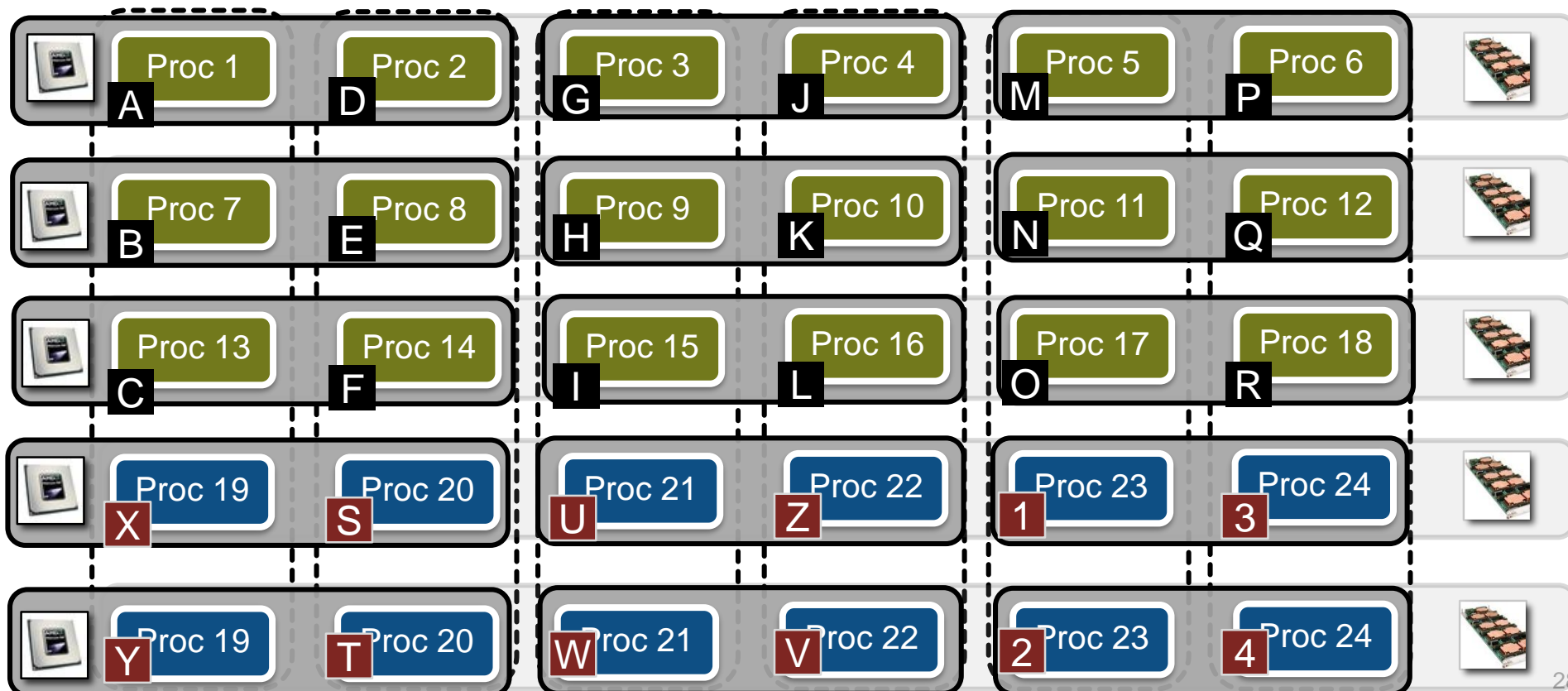
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



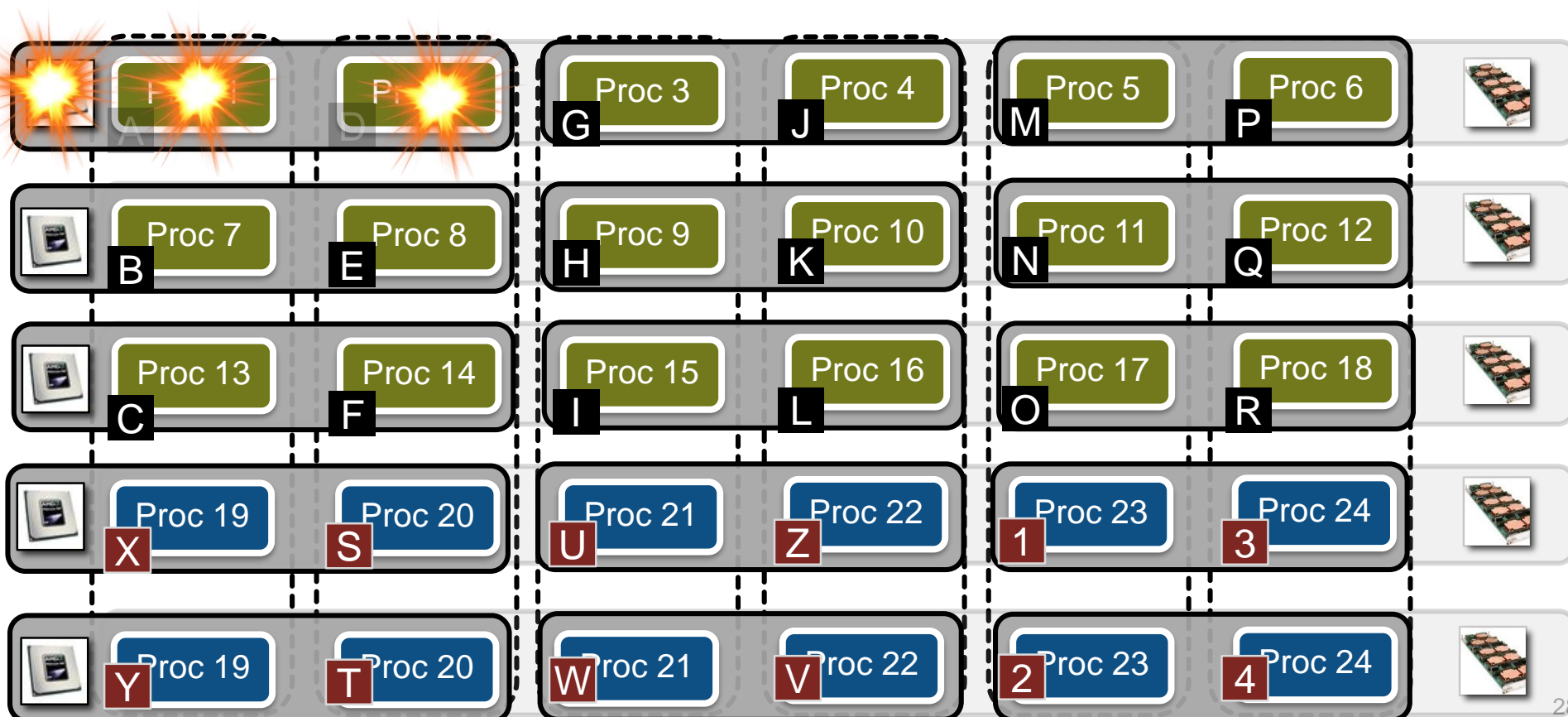
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



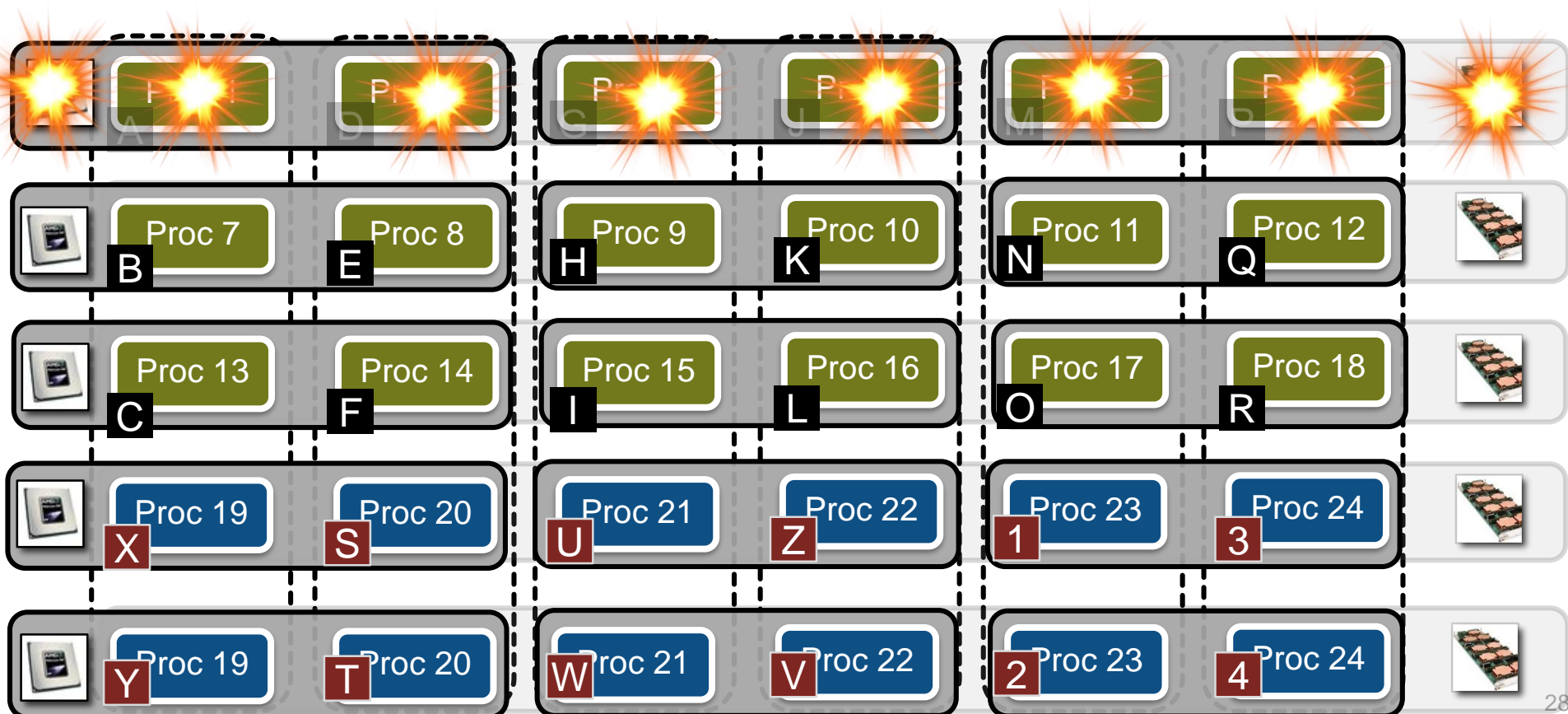
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



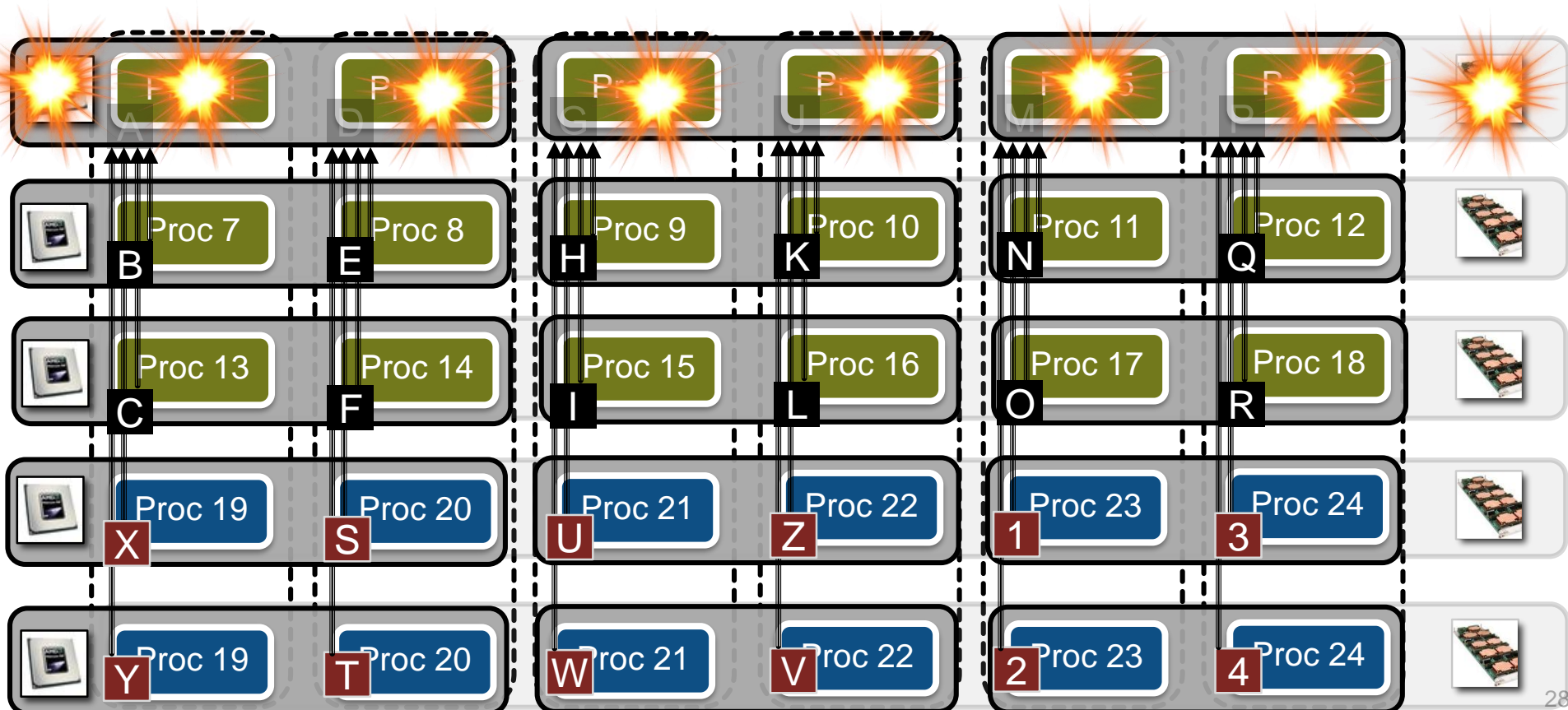
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



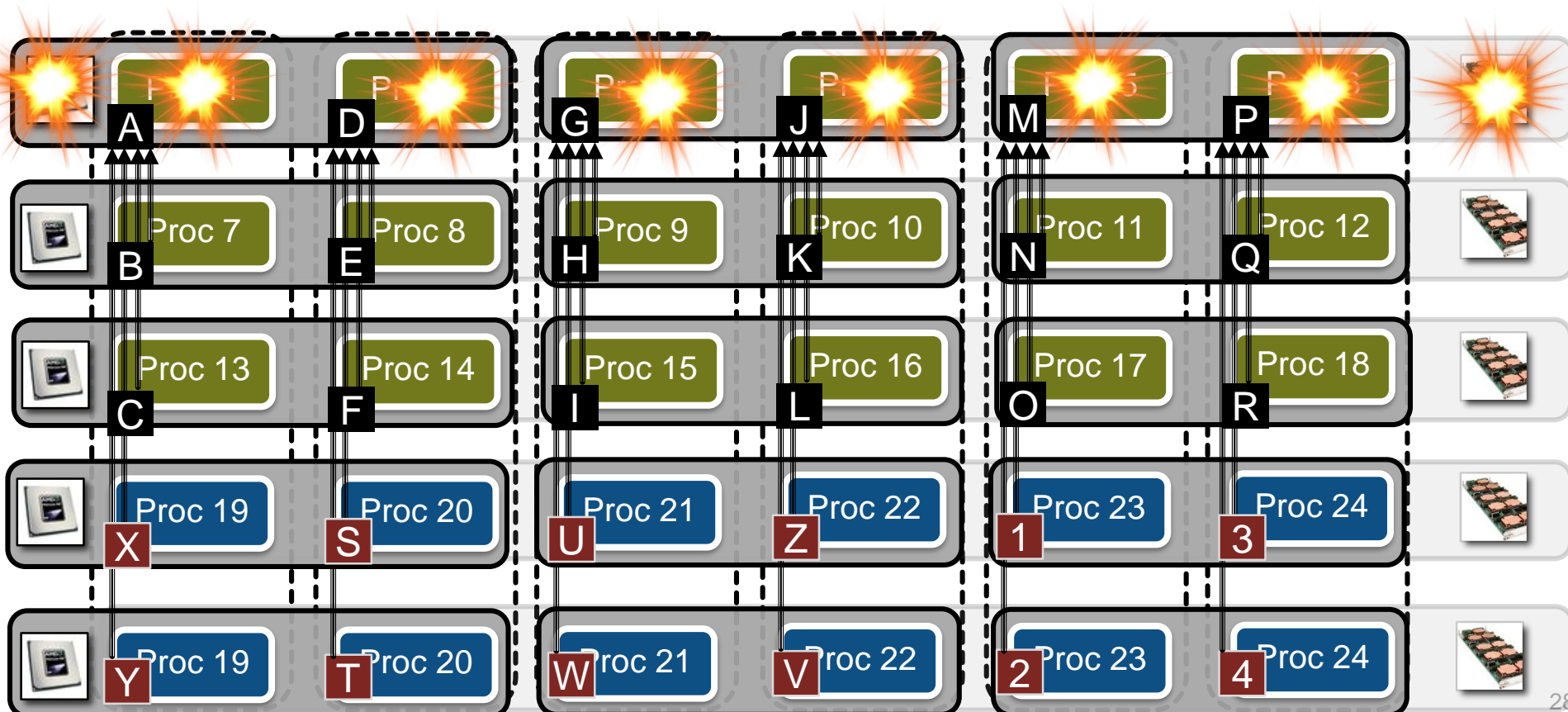
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



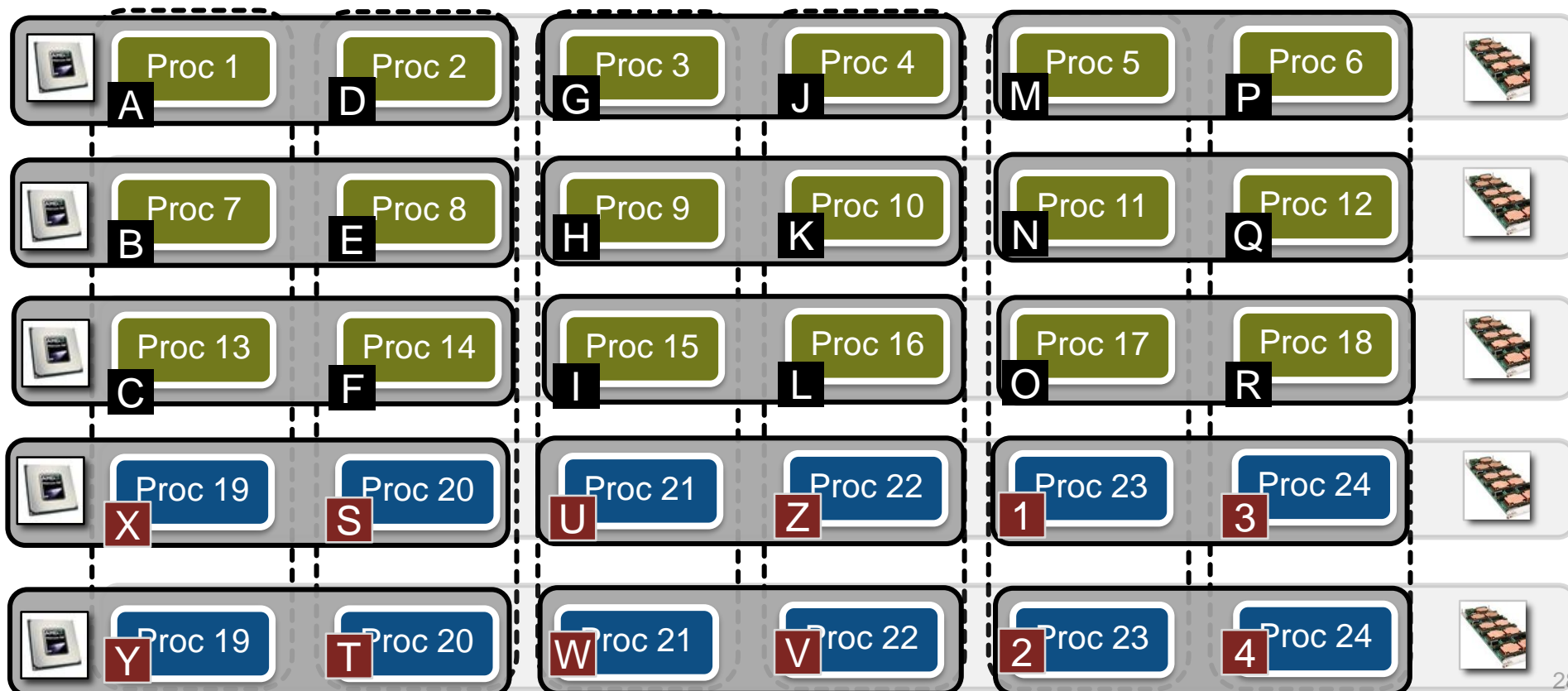
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- Step 2: topology-aware distribution of groups:
 - For example, apply topology-awareness at the level of blades...
... and nodes



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

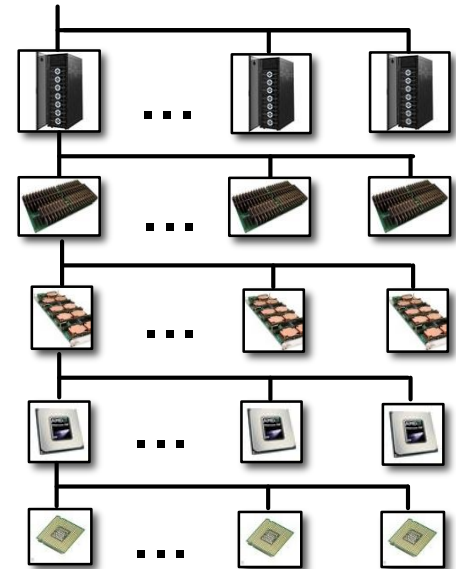
- The probability of a catastrophic failure in a multi-level computing machine

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

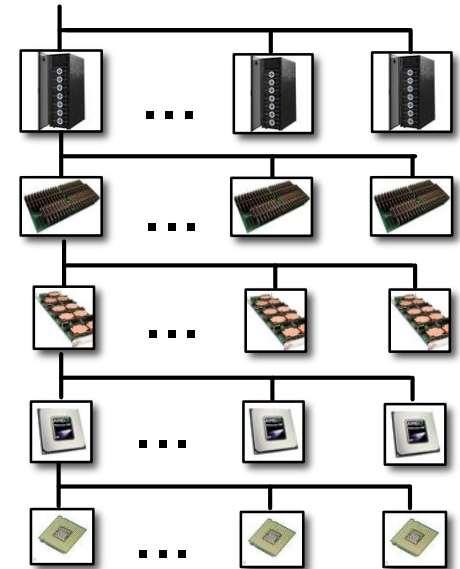
- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = P_j(x_j \cap x_{j,cf}) =$$



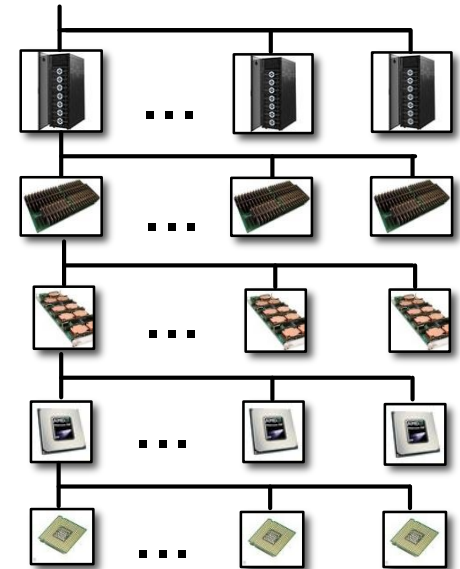
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

 $P_{cf} =$

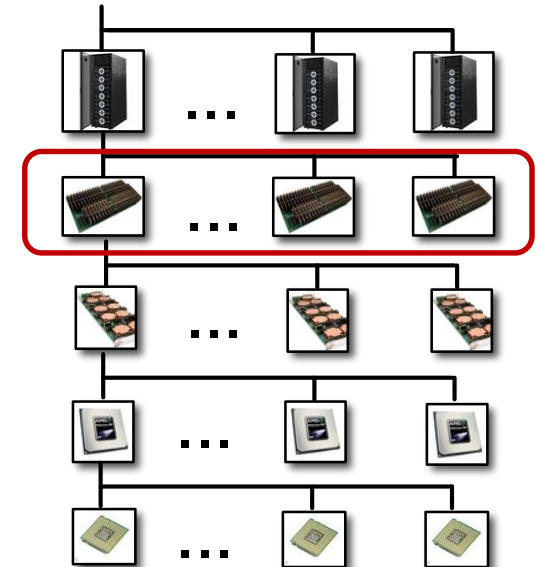
$$P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level j will fail and cause a catastrophic failure



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

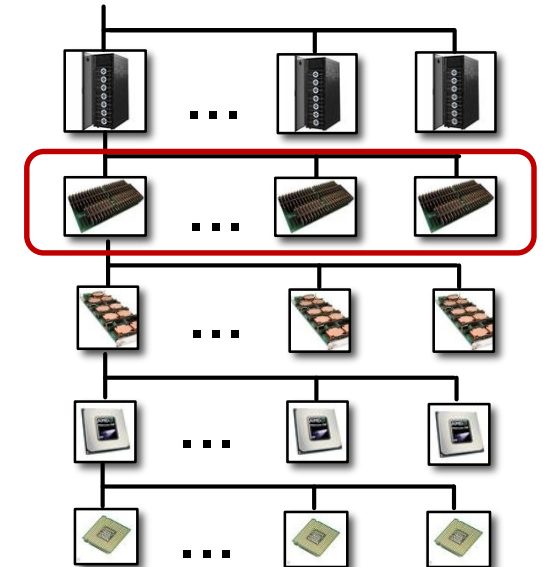

 $P_{cf} =$

$$P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level j will fail and cause a catastrophic failure

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die


 $P_{cf} =$

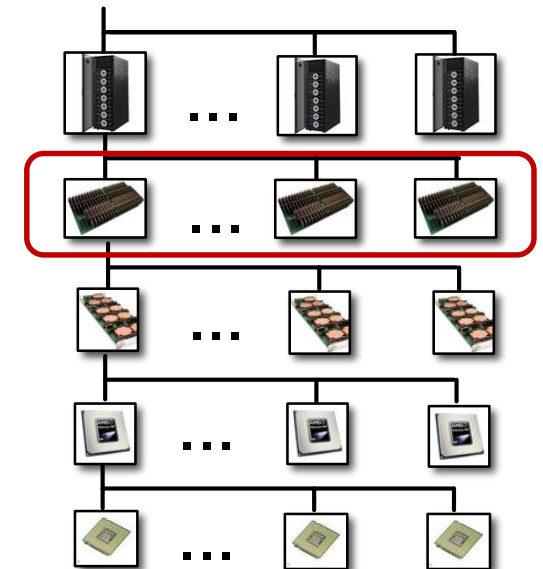
$$P_j(x_j \cap x_{j,cf}) =$$

$$P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail and cause a catastrophic failure

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die


 $P_{cf} =$

$$P_j(x_j \cap x_{j,cf}) =$$

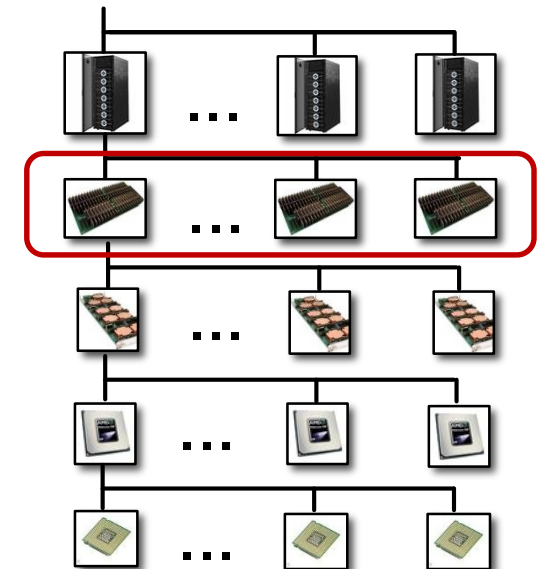
Probability that x_j elements of level j will fail and cause a catastrophic failure

$$P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die


 $P_{cf} =$

$$P_j(x_j \cap x_{j,cf}) =$$

Probability that x_j elements of level j will fail and cause a catastrophic failure

$$P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

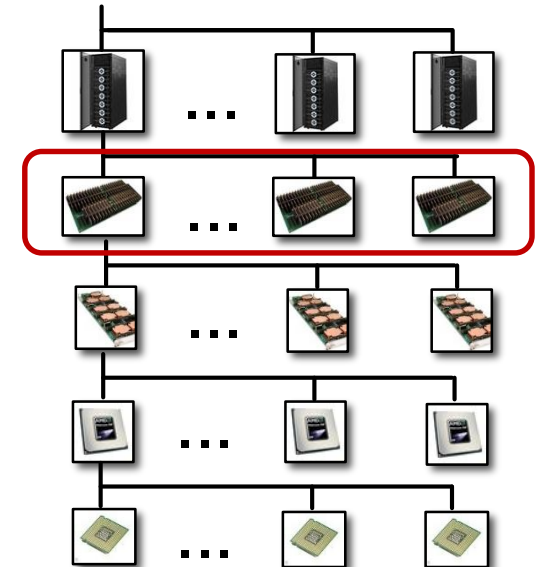
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

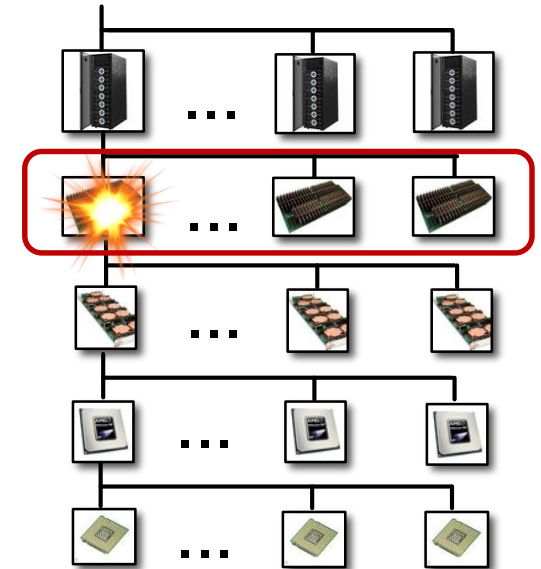
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

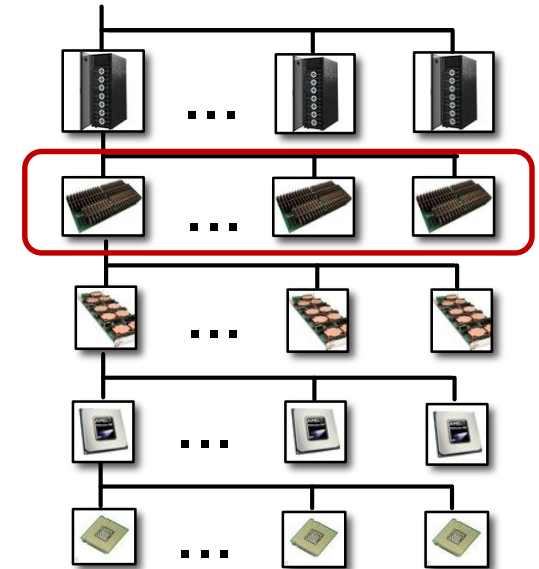
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

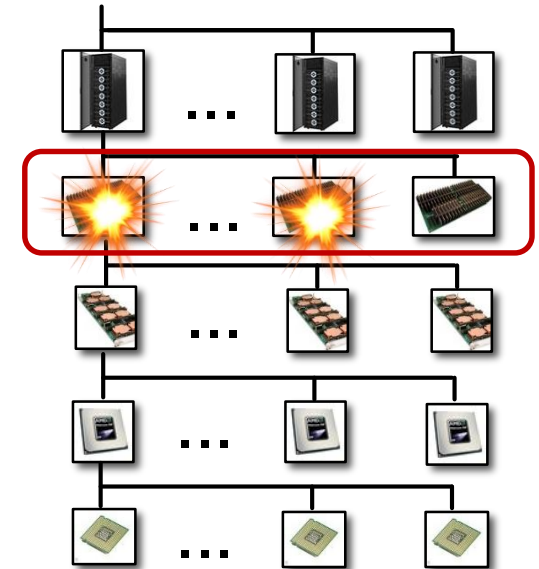
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

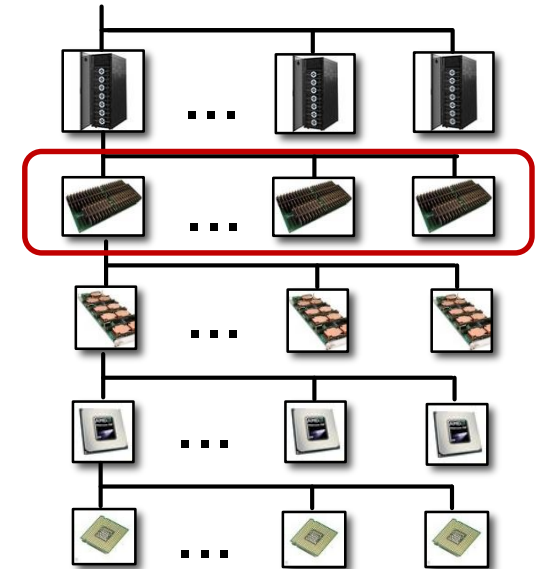
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

$$P_{cf} = \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) =$$

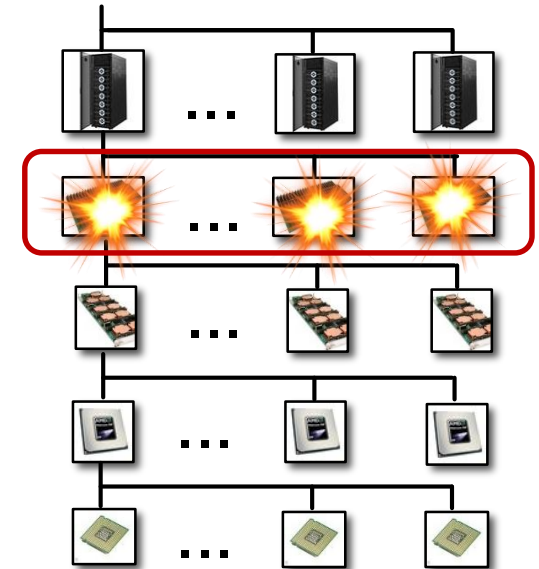
Probability that x_j elements of level j will fail and cause a catastrophic failure

Every number of x_j elements is considered...

$$\sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

...at every hierarchy level

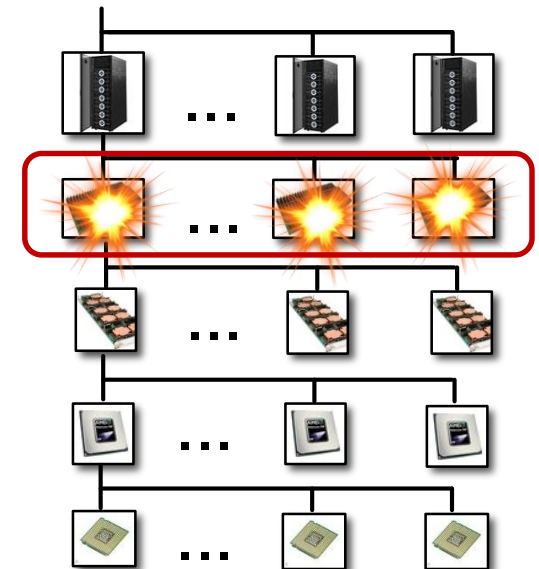
$$P_{cf} = \sum_{j=1}^h \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{j=1}^h \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

Probability that x_j elements of level j will fail and cause a catastrophic failure

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

...at every hierarchy level

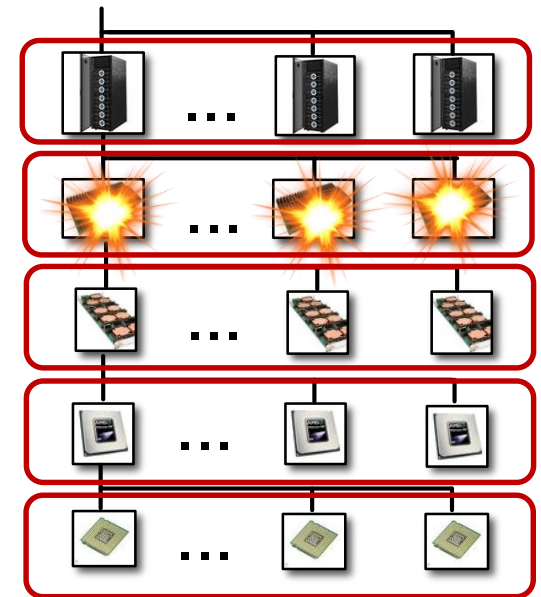
$$P_{cf} = \sum_{j=1}^h \sum_{x_j=1}^{H_j} P_j(x_j \cap x_{j,cf}) = \sum_{j=1}^h \sum_{x_j=1}^{H_j} P_j(x_j) P_j(x_{j,cf} | x_j)$$

Every number of x_j elements is considered...

Probability that x_j elements of level j will fail and cause a catastrophic failure

Probability that x_j elements of level j will fail

Probability that x_j given failures at level j are catastrophic



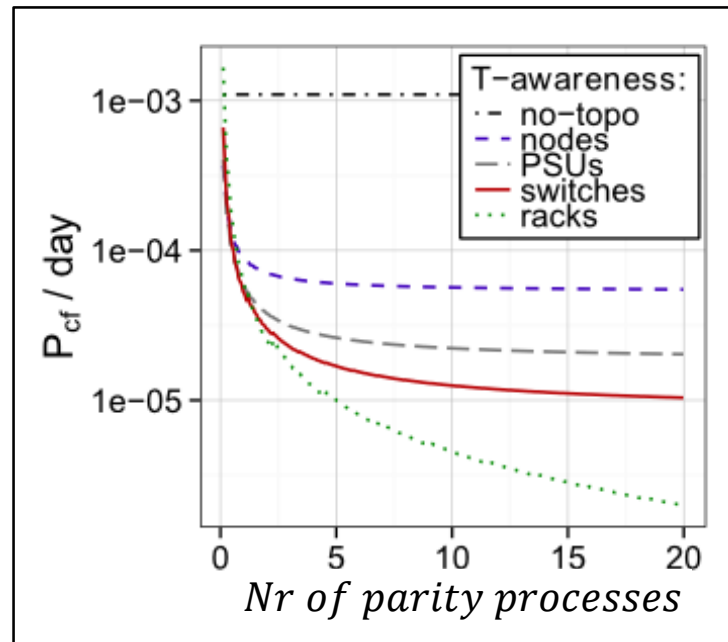
EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
 - A catastrophic failure: a failure that takes place when more than m processes in the same group die

P_{cf}/day :



OVERVIEW OF OUR RESEARCH

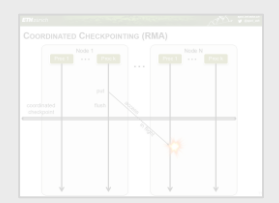
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

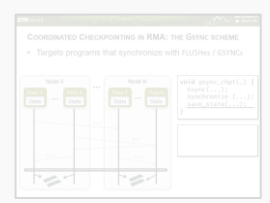
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

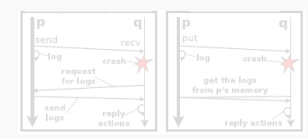


MP vs. RMA

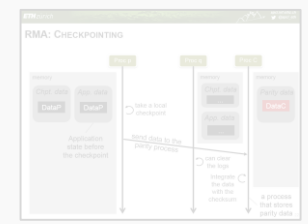


Schemes

UC in RMA

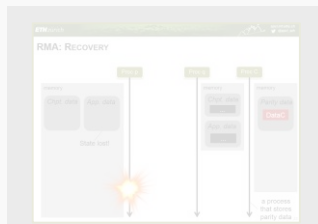


MP vs. RMA



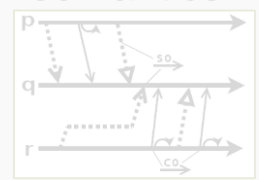
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{coh} order (referred to as the gsync order).*

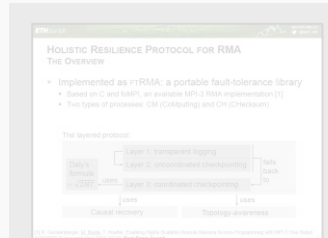
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

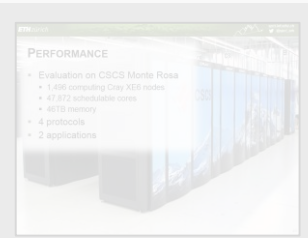
Holistic fault-tolerance library



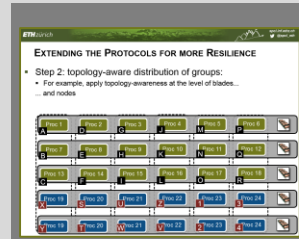
Design

Checkpoints on demand

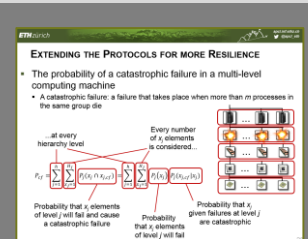
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

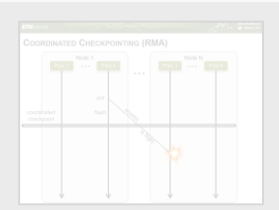
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

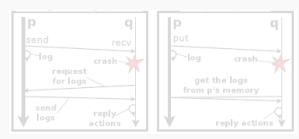


MP vs. RMA

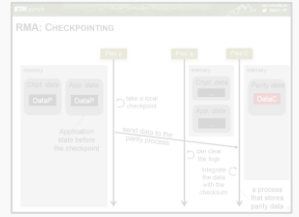


Schemes

UC in RMA

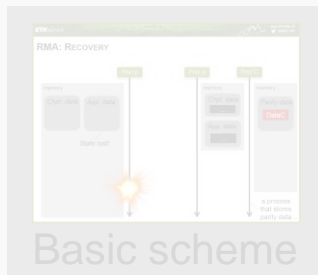


MP vs. RMA



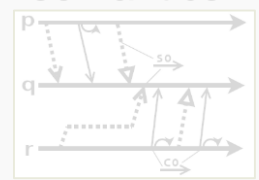
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

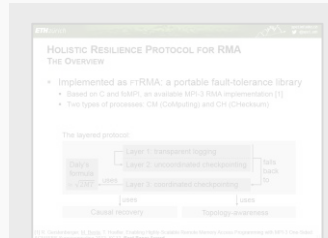
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

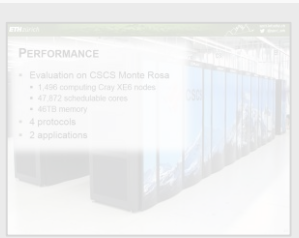
Holistic fault-tolerance library



Design

Checkpoints on demand

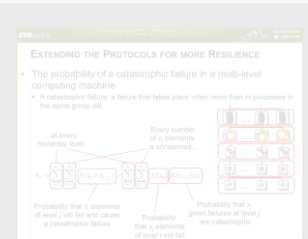
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

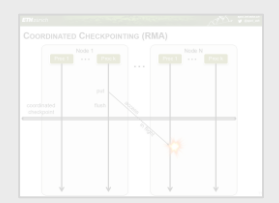
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

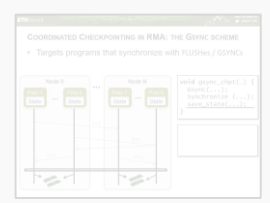
PUT($p \xrightarrow{so} q$)
GET($p \xleftarrow{so} q$)



CC in RMA

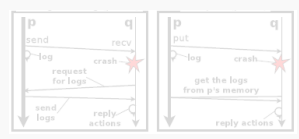


MP vs. RMA

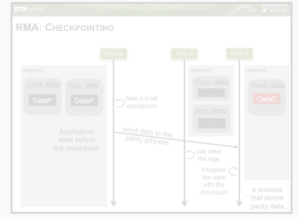


Schemes

UC in RMA

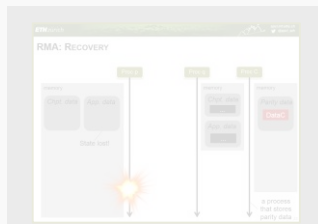


MP vs. RMA



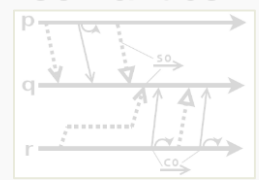
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{coh} order (referred to as the gsync order).*

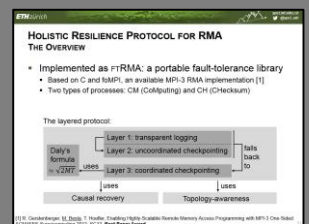
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

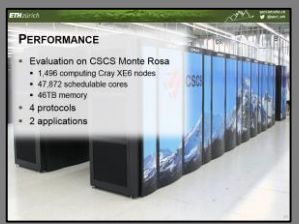
Holistic fault-tolerance library



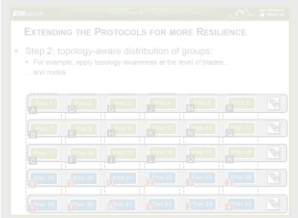
Design

Checkpoints on demand

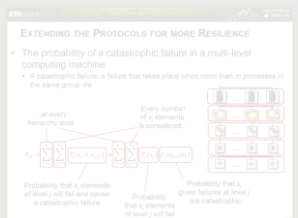
Optimizations



Performance



Distribution of processes



Decreasing failure prob.

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as `FTRMA`: a portable fault-tolerance library
 - Based on C and `foMPI`, an available MPI-3 RMA implementation [1]

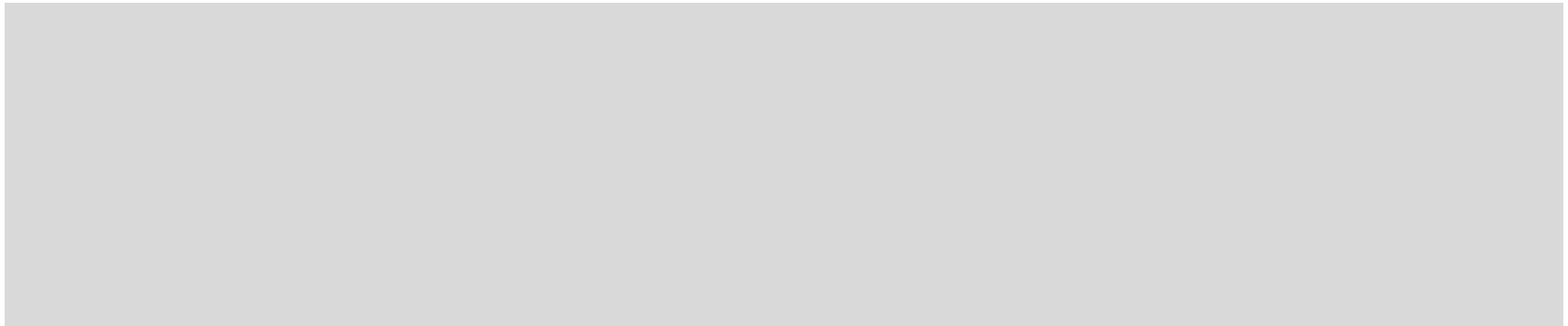
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as FTRMA : a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:



[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as FTRMA : a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

A diagram illustrating a layered protocol. It consists of a large light gray rectangular area containing a smaller, darker gray rectangular area in the center. The text 'transparent logging' is centered within the darker gray area.

transparent logging

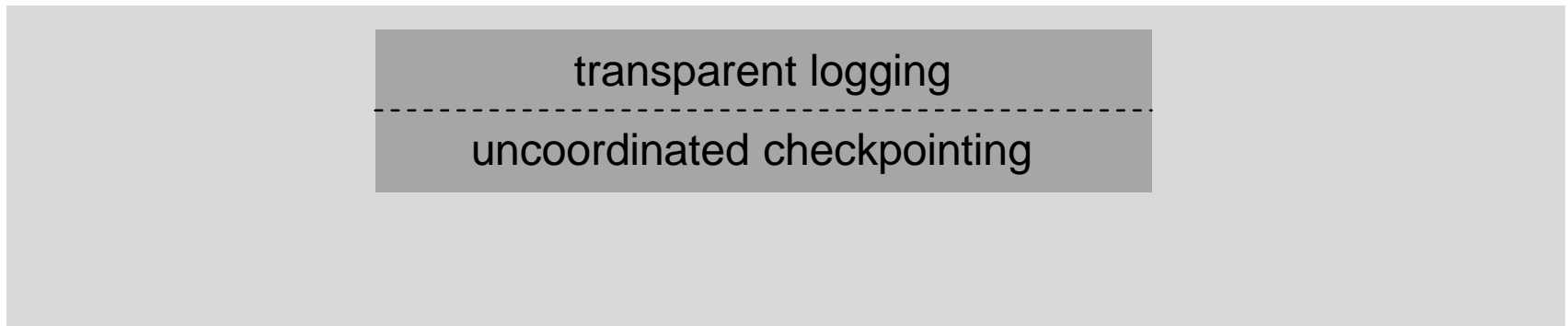
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as FTRMA : a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:



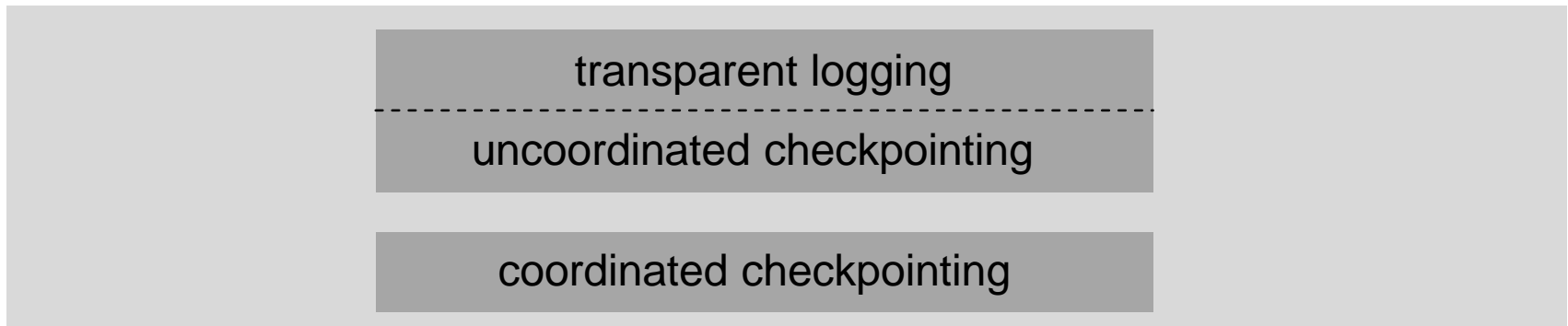
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as FTRMA : a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:



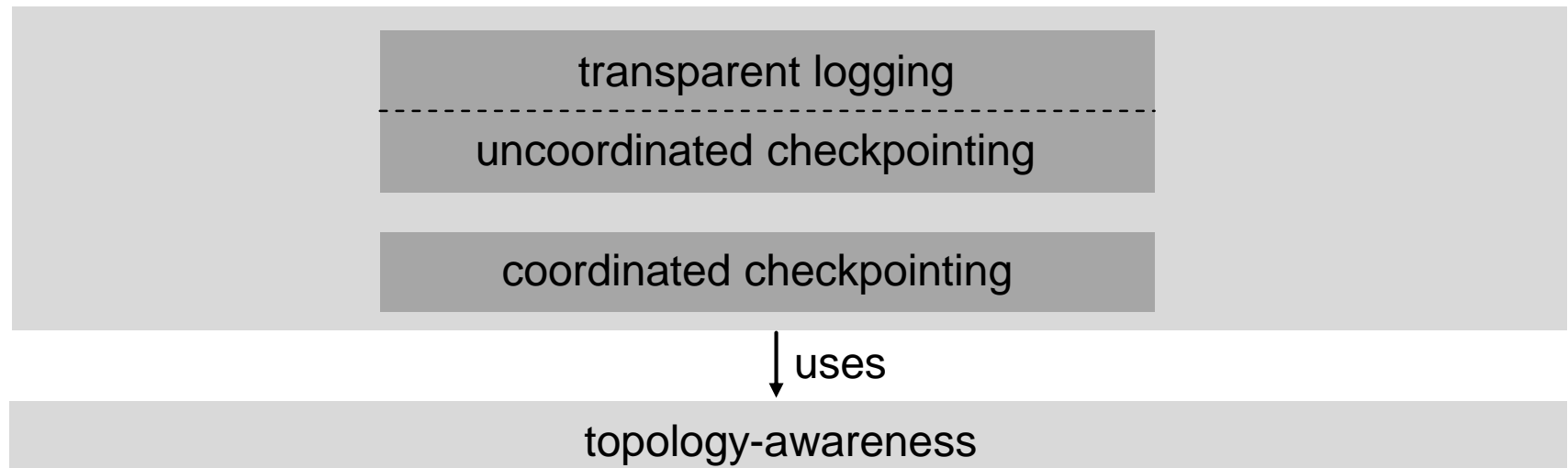
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as `FTRMA`: a portable fault-tolerance library
 - Based on C and `foMPI`, an available MPI-3 RMA implementation [1]

The layered protocol:



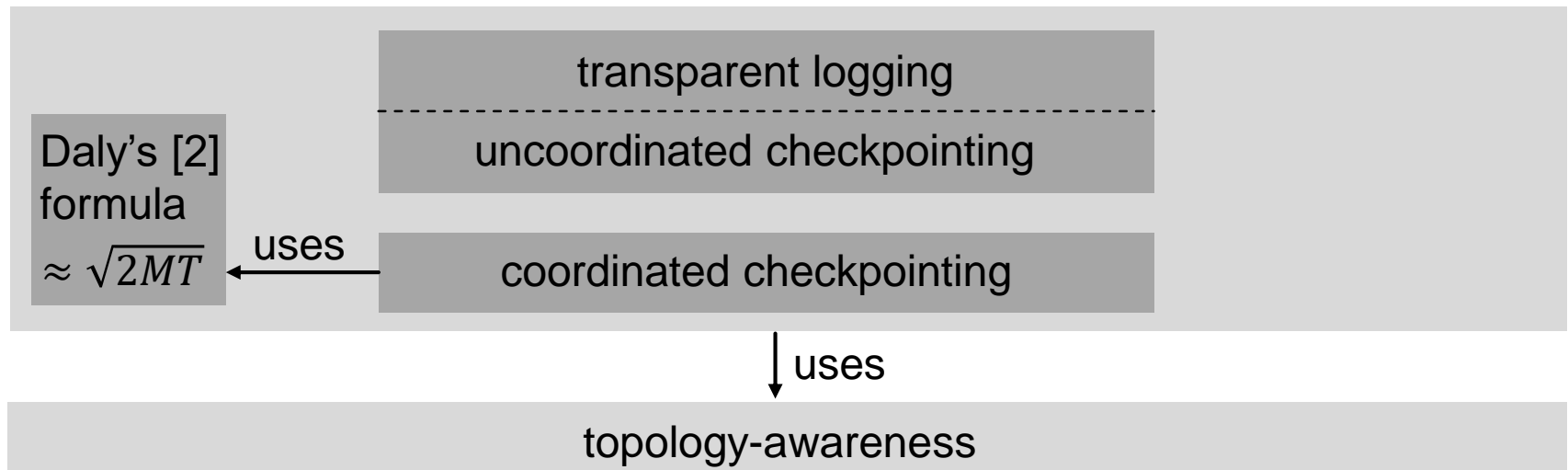
[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as FTRMA : a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:



[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

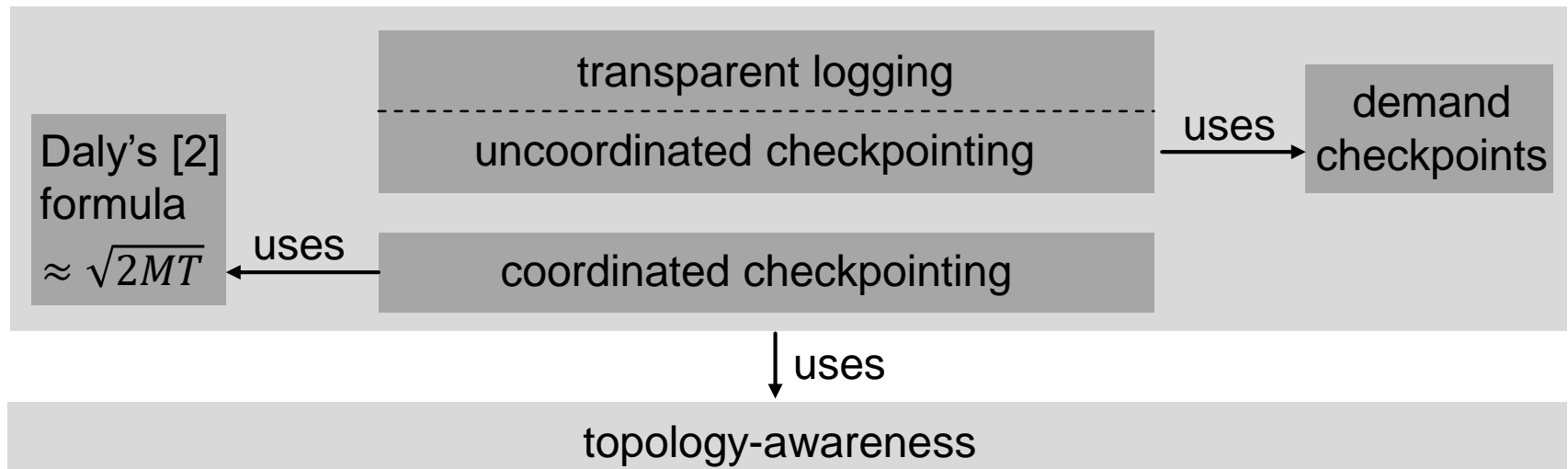
[2] J.T.Daly, A higher order estimate of the optimum checkpoint interval for restart dumps. Future Generation Computer Systems, Volume 22 Issue 3, February 2006, Pages 303-312

HOLISTIC RESILIENCE PROTOCOL FOR RMA

THE OVERVIEW

- Implemented as **FTRMA**: a portable fault-tolerance library
 - Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

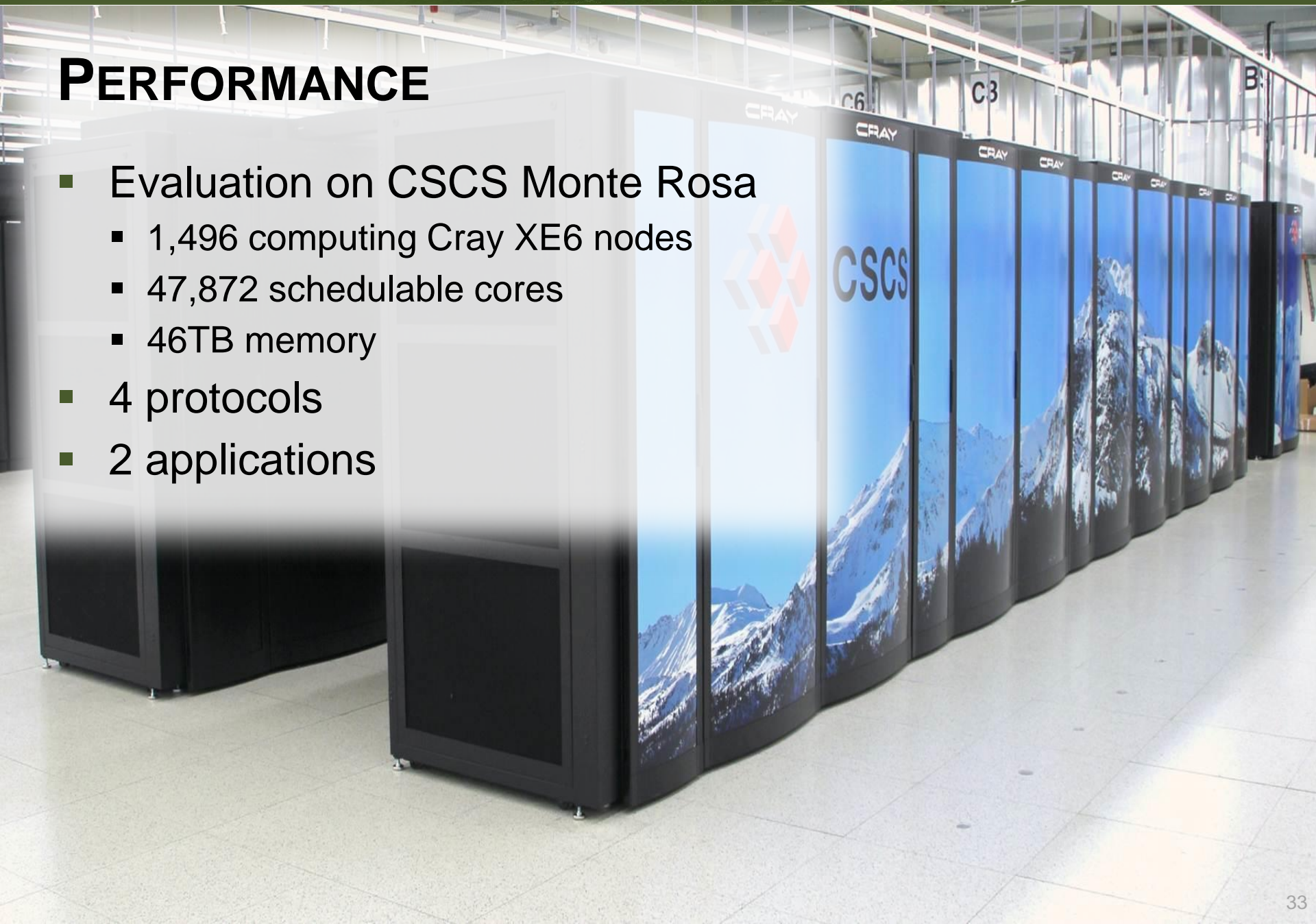


[1] R. Gerstenberger, M. Besta, T. Hoefler, Enabling Highly-Scalable Remote Memory Access Programming with MPI-3 One-Sided. ACM/IEEE Supercomputing 2013, SC13, **Best Paper Award**

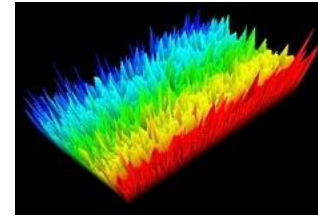
[2] J.T.Daly, A higher order estimate of the optimum checkpoint interval for restart dumps. Future Generation Computer Systems, Volume 22 Issue 3, February 2006, Pages 303-312

PERFORMANCE

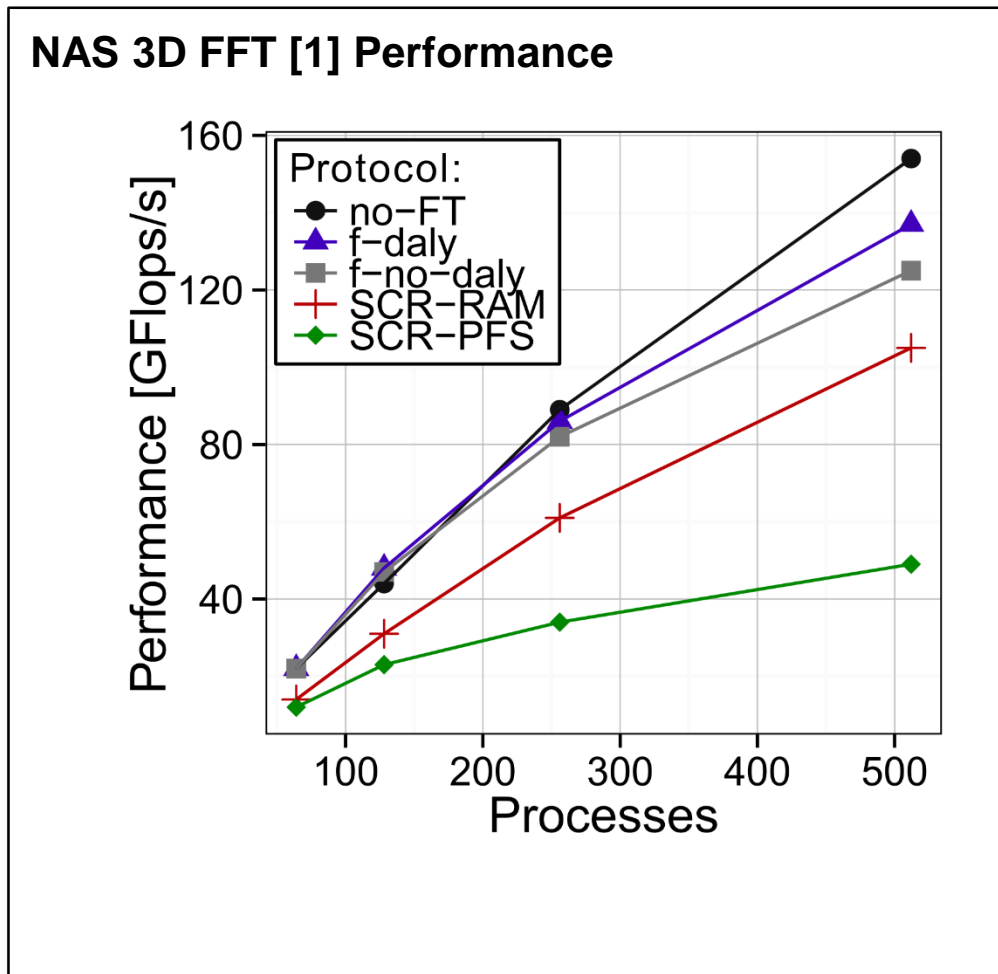
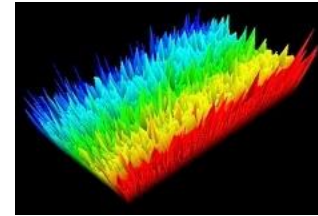
- Evaluation on CSCS Monte Rosa
 - 1,496 computing Cray XE6 nodes
 - 47,872 schedulable cores
 - 46TB memory
- 4 protocols
- 2 applications



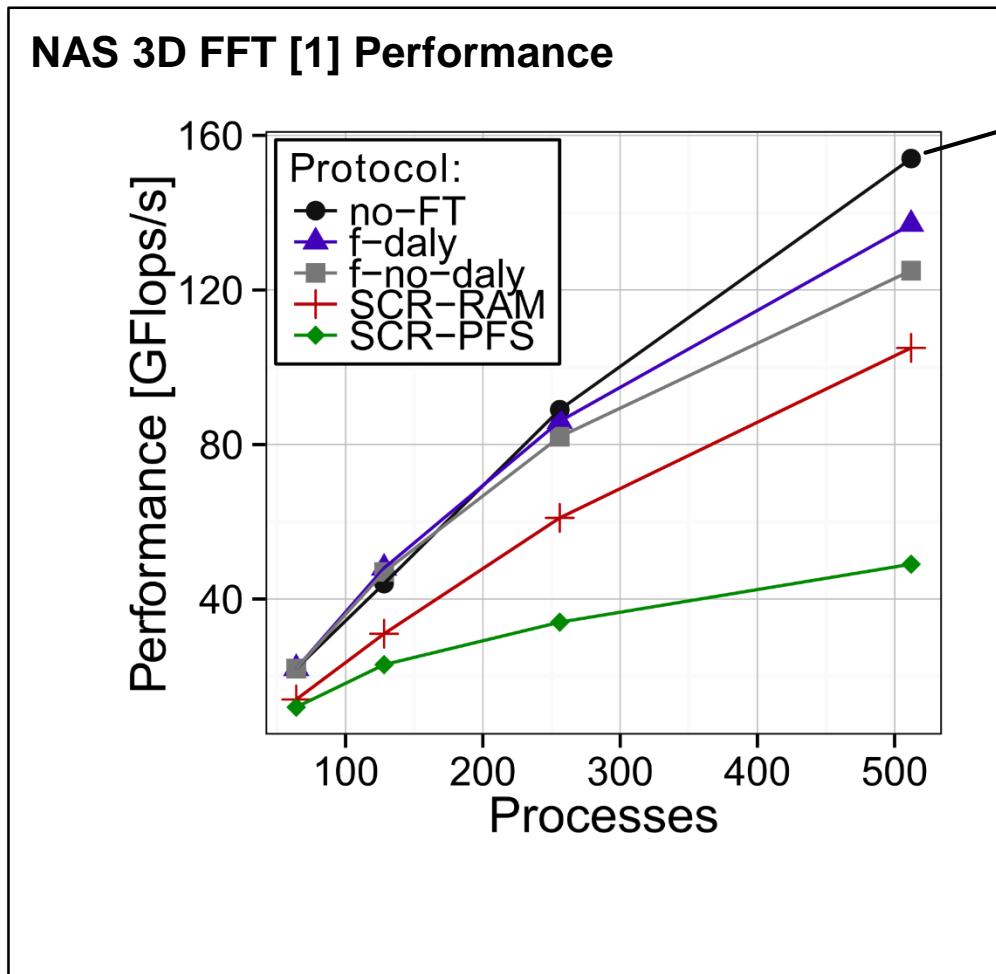
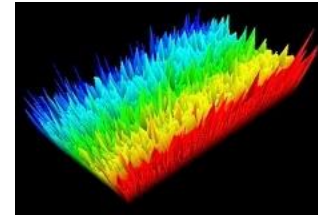
PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT



PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT

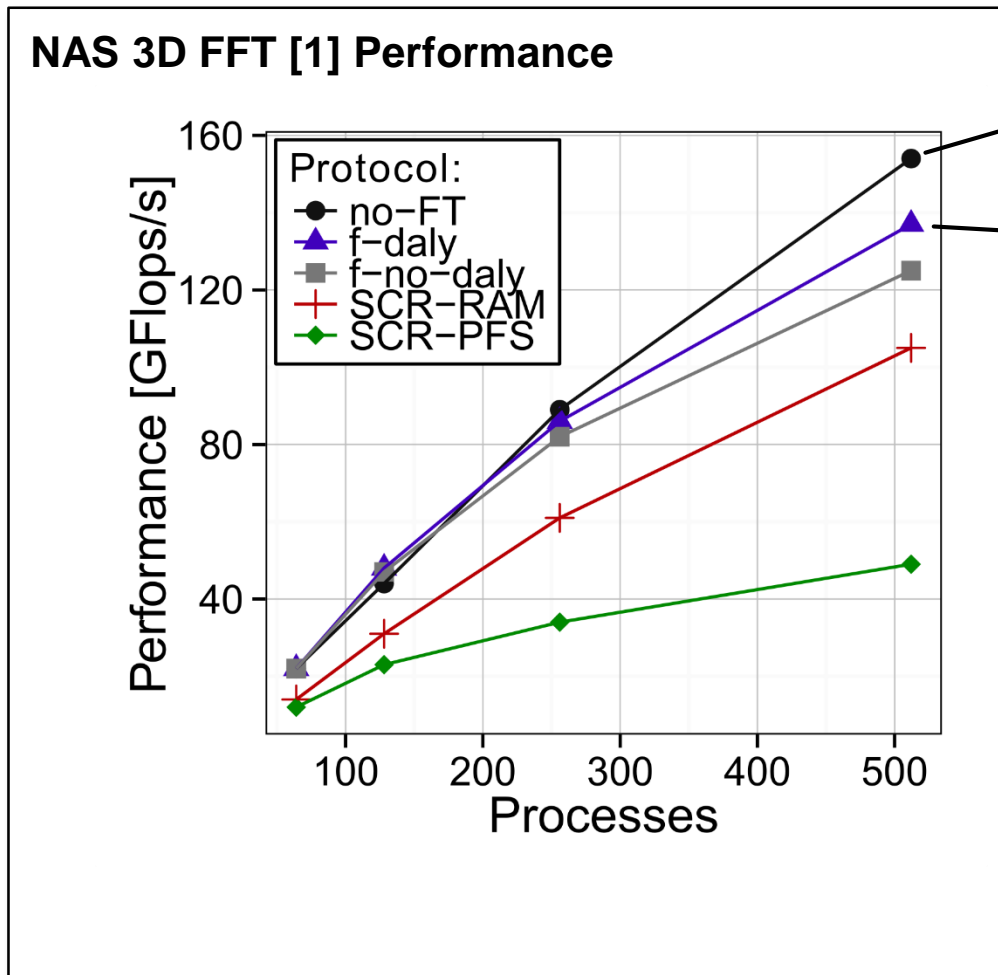
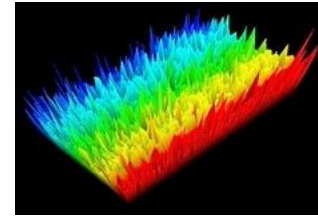


PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT



no-FT: no fault tolerance

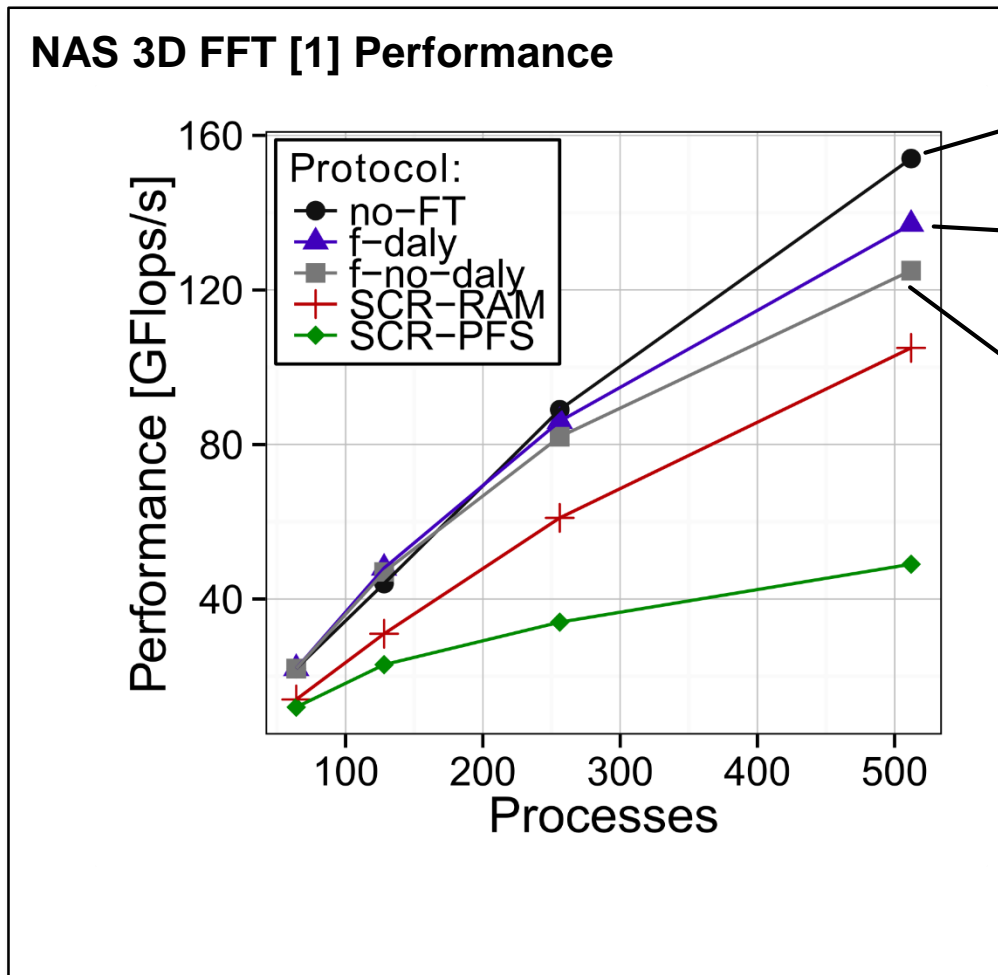
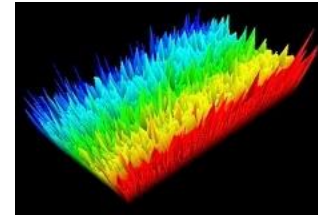
PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT



no-FT: no fault tolerance

f-daly: using Daly's interval
1-5% slower than no-FT

PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT

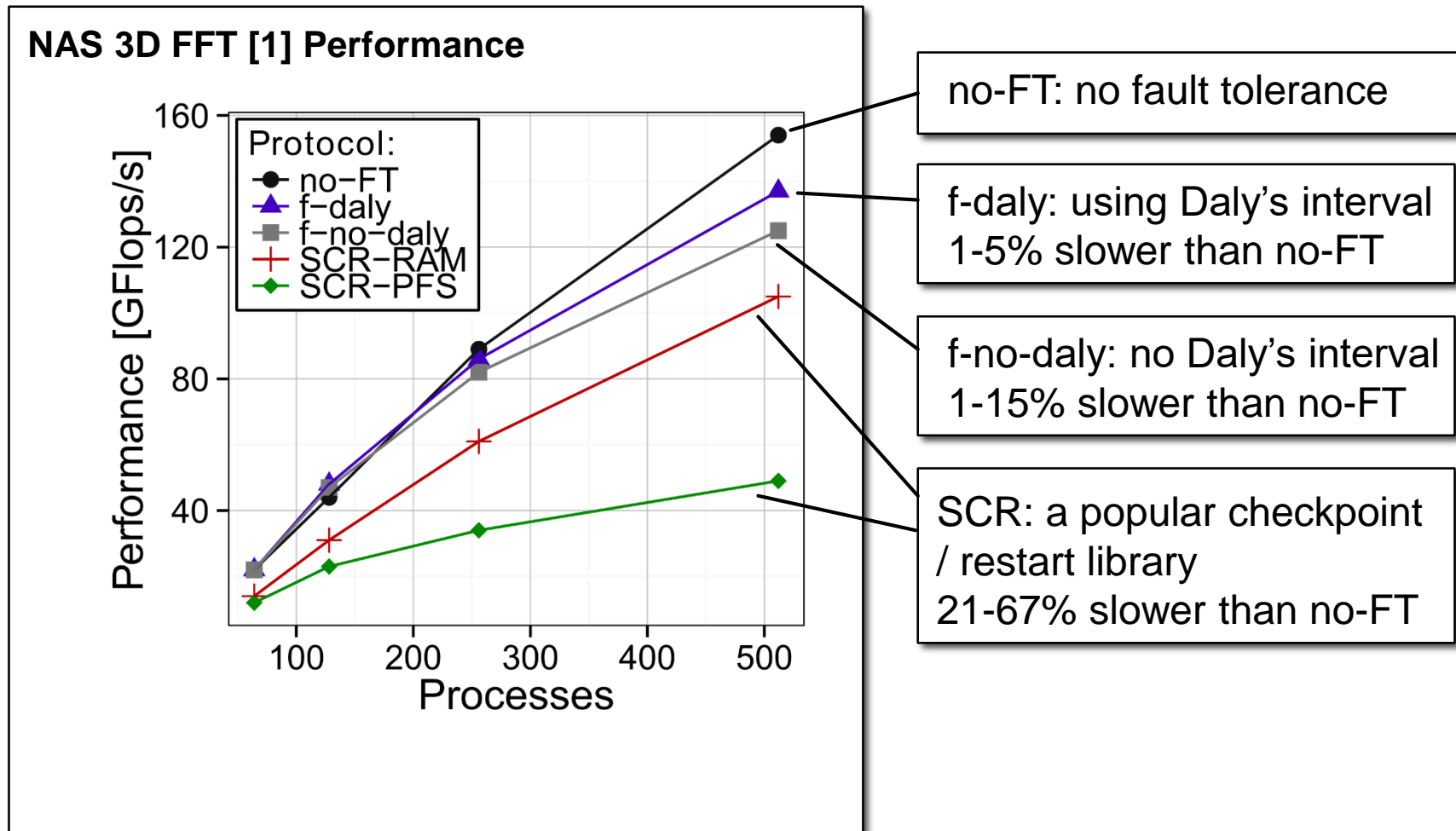
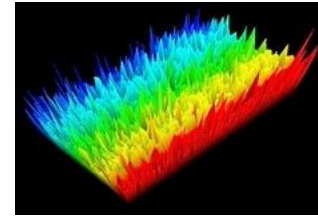


no-FT: no fault tolerance

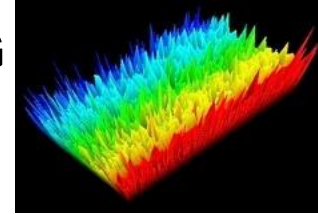
f-daly: using Daly's interval
1-5% slower than no-FT

f-no-daly: no Daly's interval
1-15% slower than no-FT

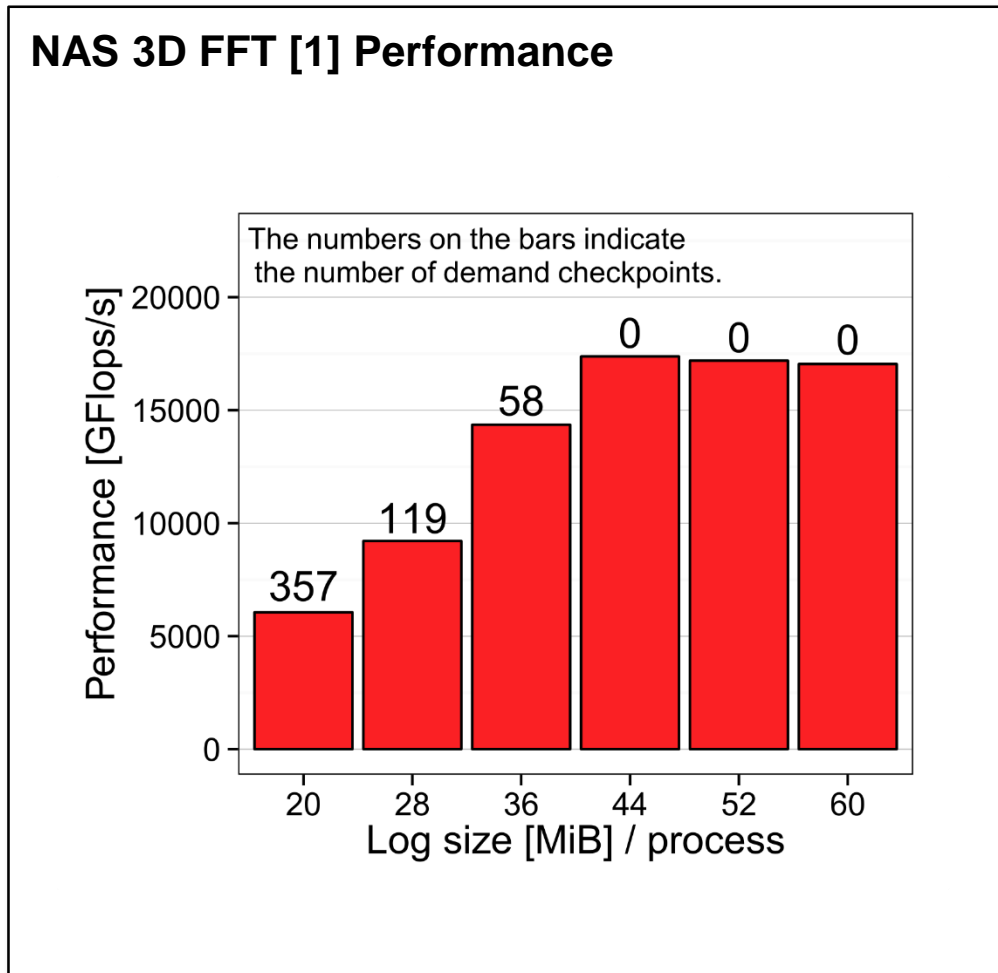
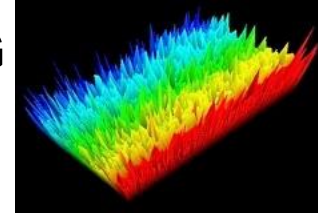
PERFORMANCE: COORDINATED CHECKPOINTING NAS 3D FFT



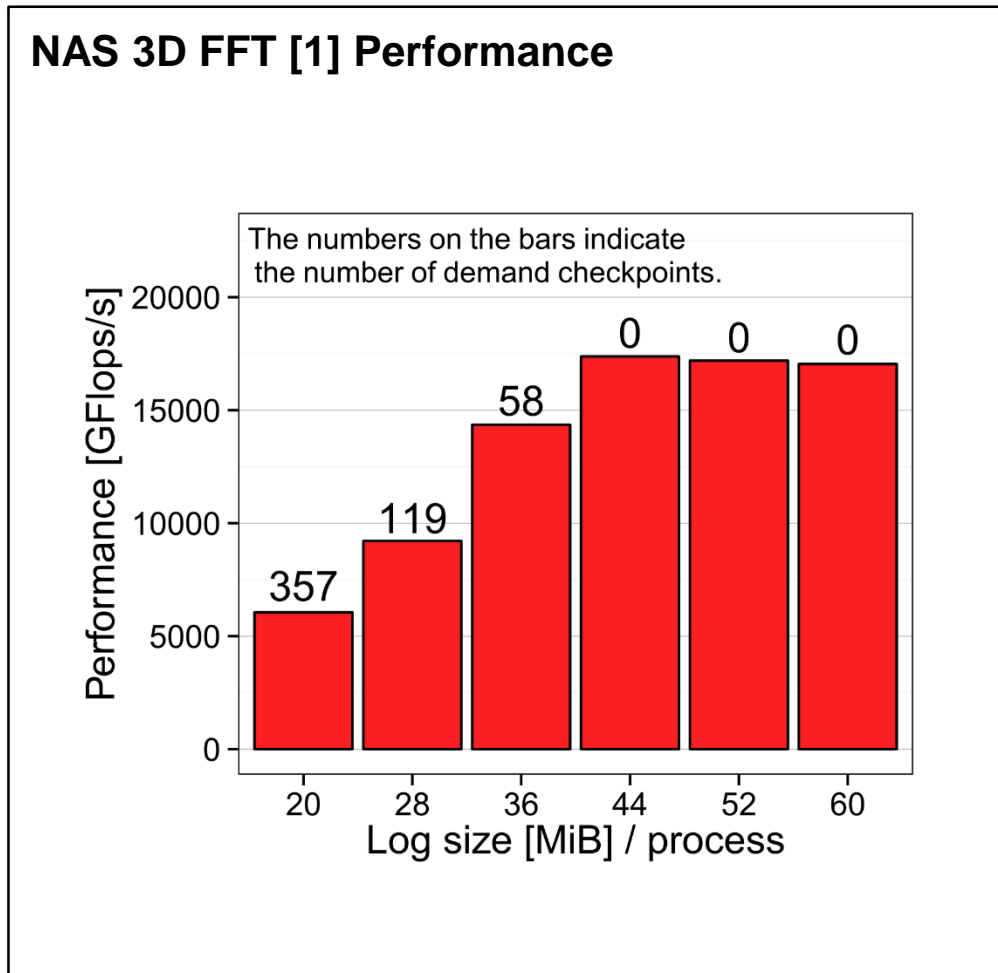
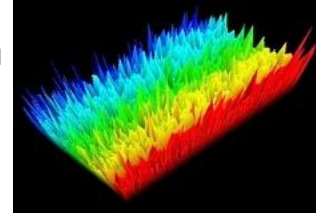
PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT



PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT

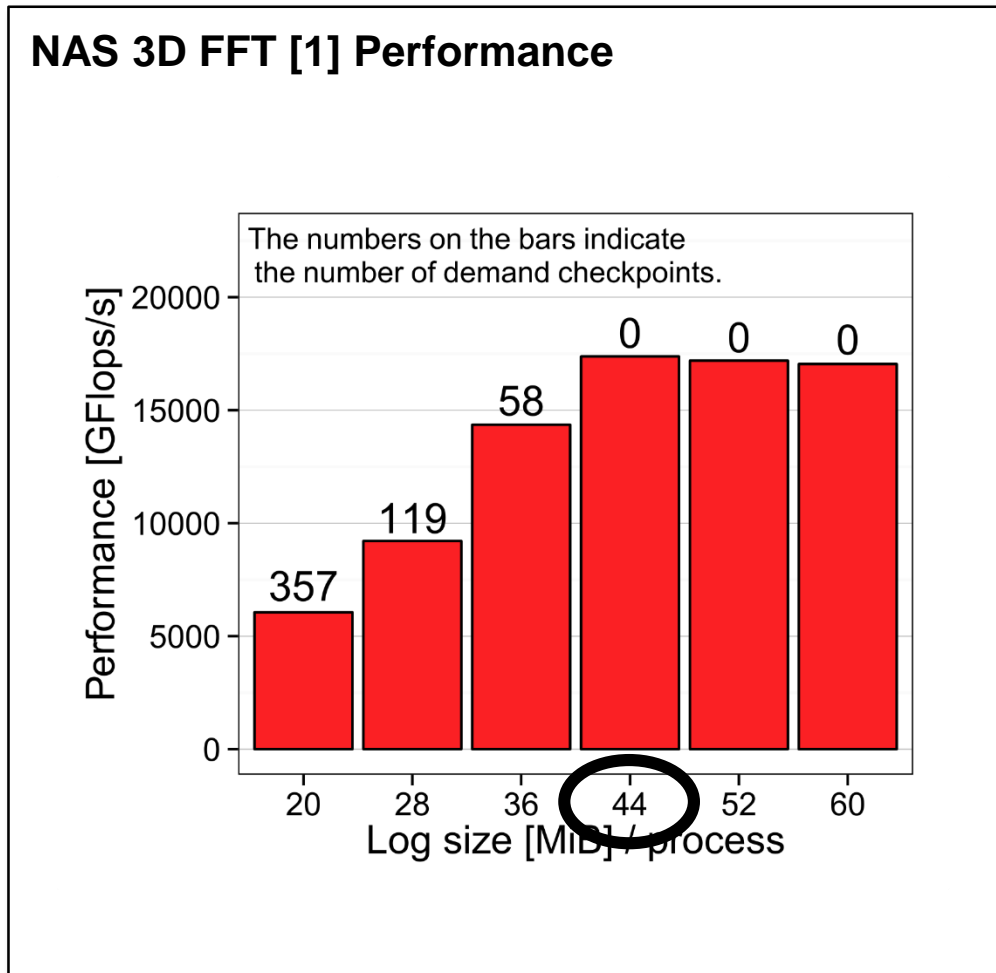
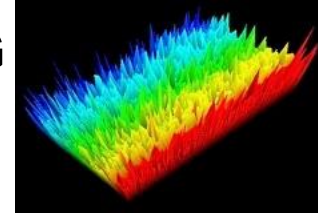


PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT



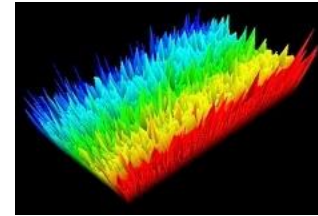
How the log size impacts the number of uncoordinated checkpoints and the performance of the code?

PERFORMANCE: UNCOORDINATED CHECKPOINTING NAS 3D FFT

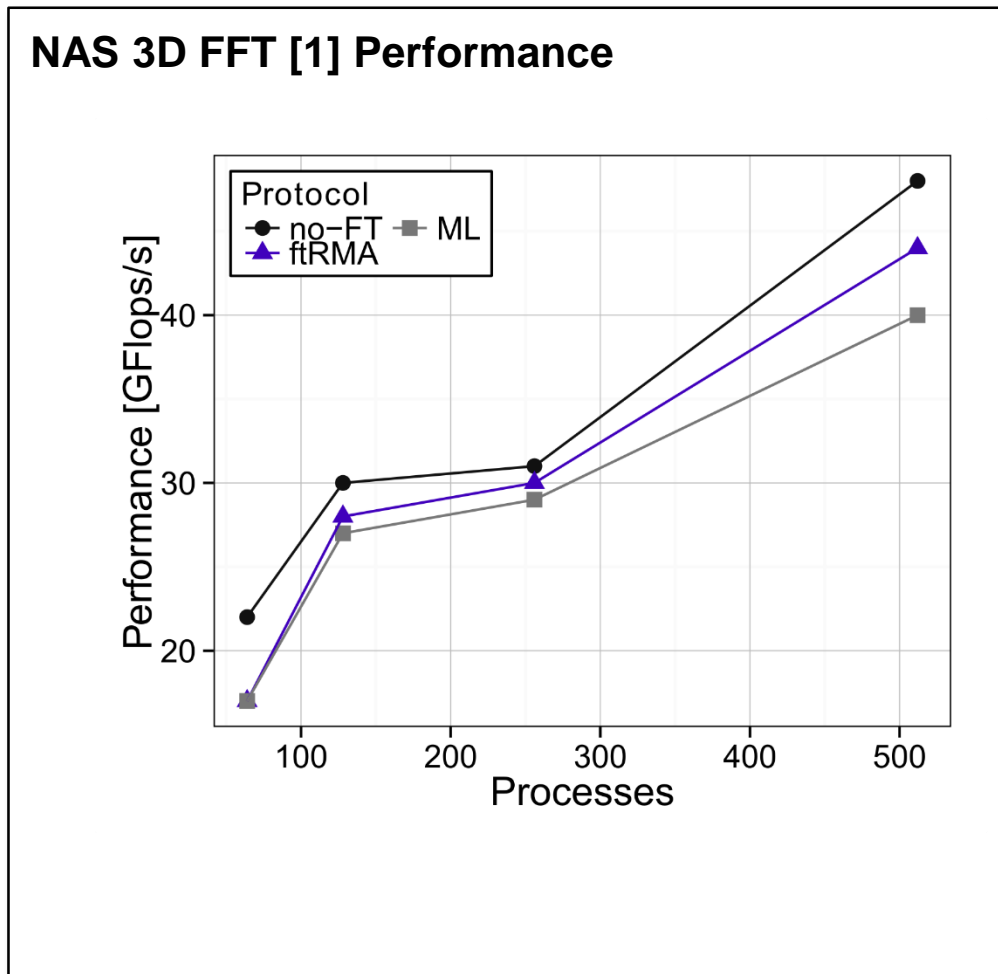
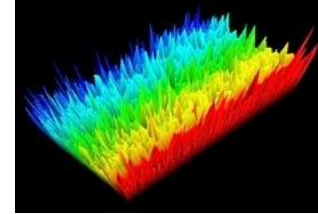


How the log size impacts the number of uncoordinated checkpoints and the performance of the code?

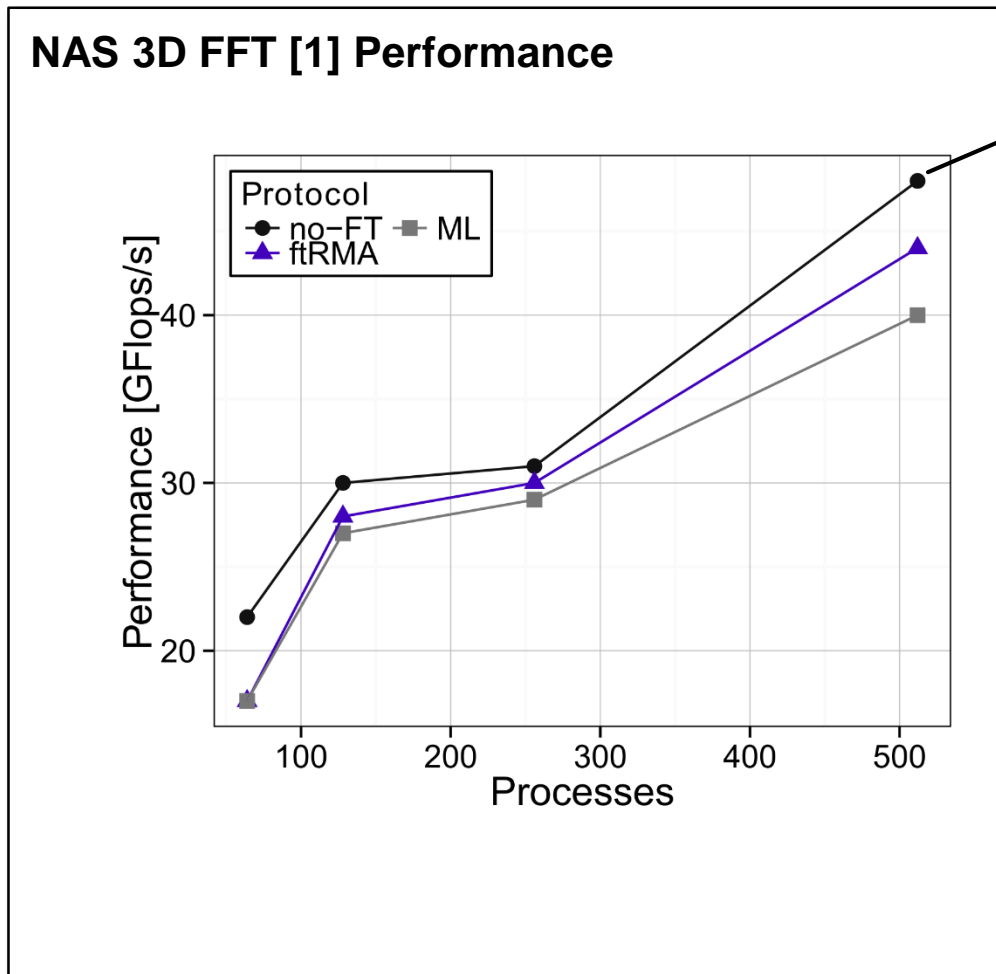
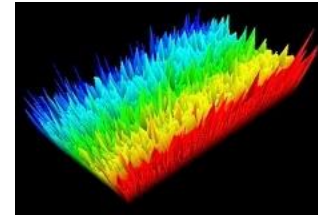
PERFORMANCE: ACCESS LOGGING NAS 3D FFT



PERFORMANCE: ACCESS LOGGING NAS 3D FFT

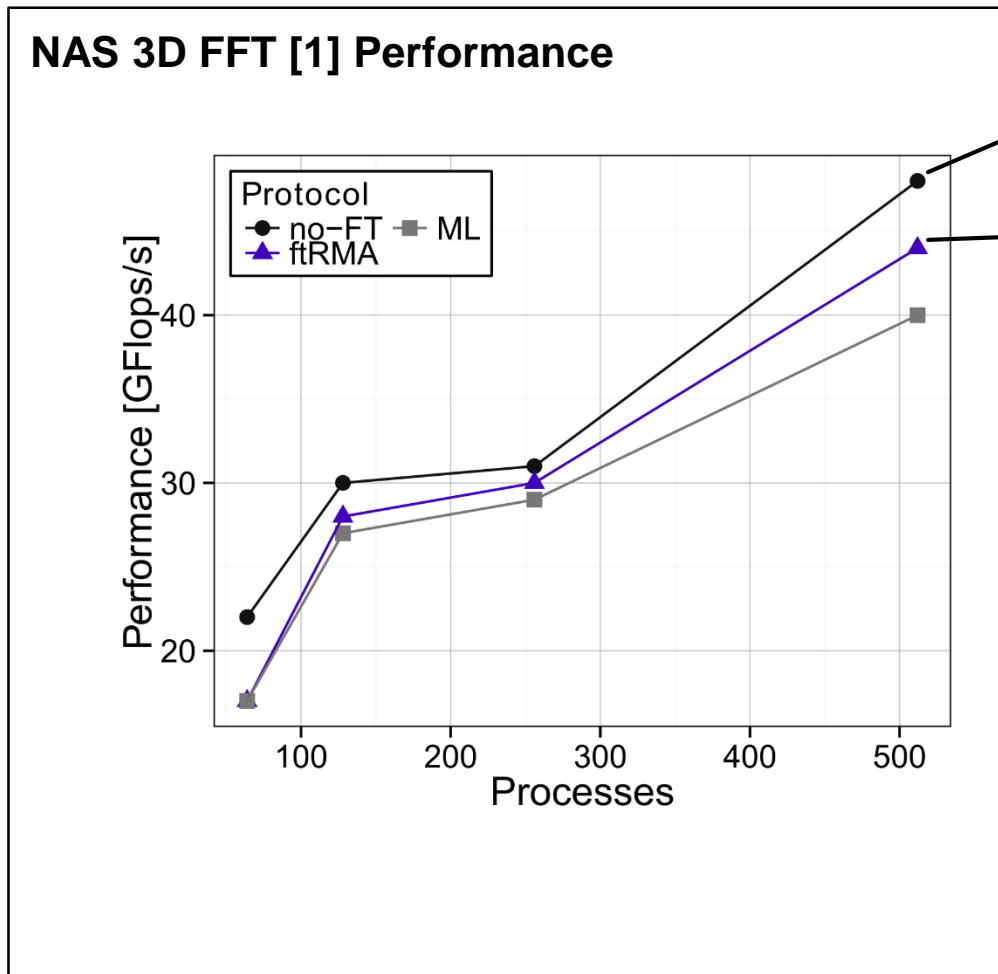
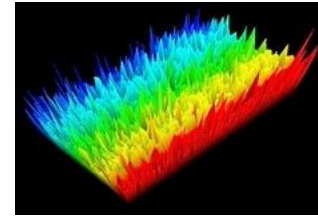


PERFORMANCE: ACCESS LOGGING NAS 3D FFT



no-FT: no fault tolerance

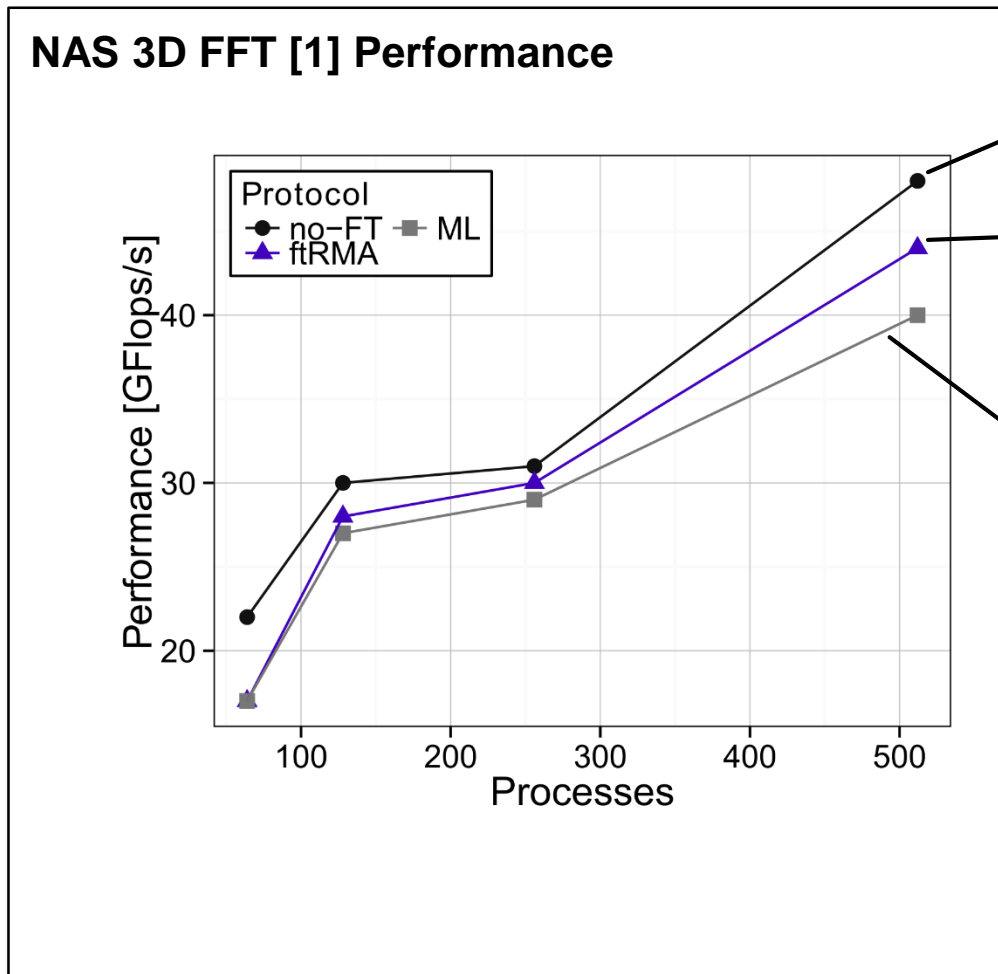
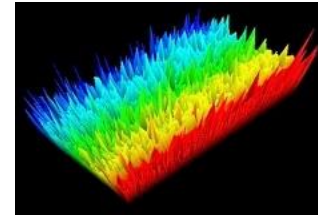
PERFORMANCE: ACCESS LOGGING NAS 3D FFT



no-FT: no fault tolerance

FTRMA: logging puts (FFT code does not use gets)
8-9% slower than no-FT

PERFORMANCE: ACCESS LOGGING NAS 3D FFT



no-FT: no fault tolerance

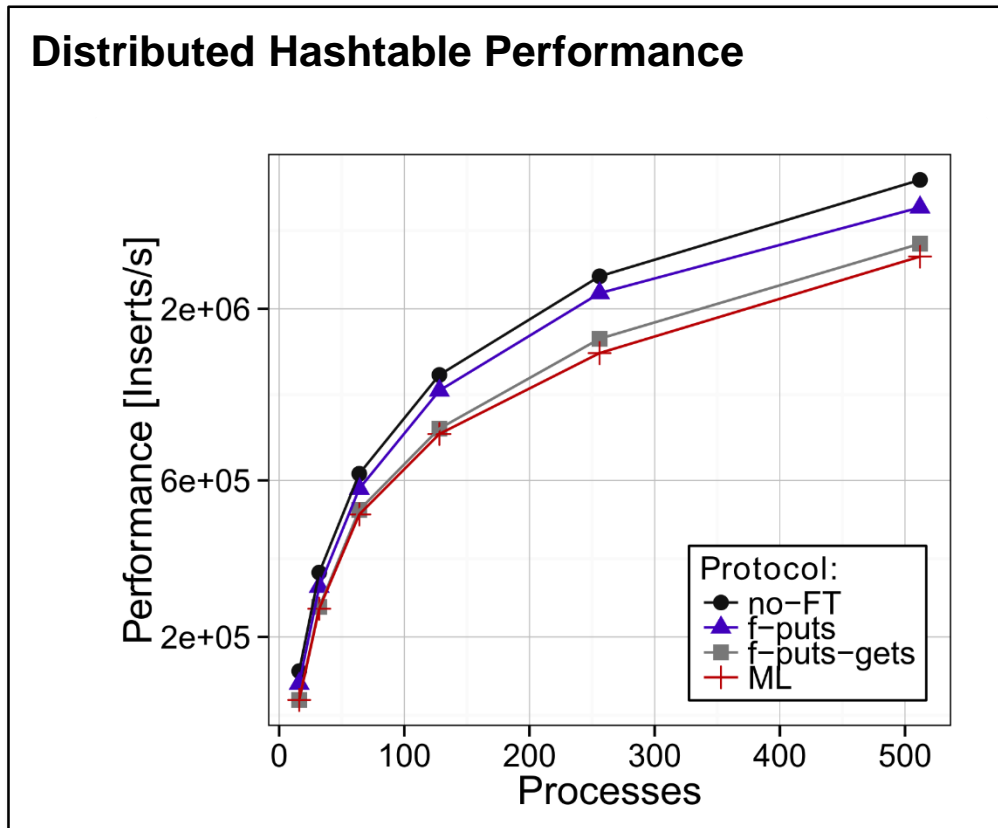
ftRMA: logging puts (FFT code does not use gets)
8-9% slower than no-FT

ML: a simple protocol based on message logging
18% slower than no-FT

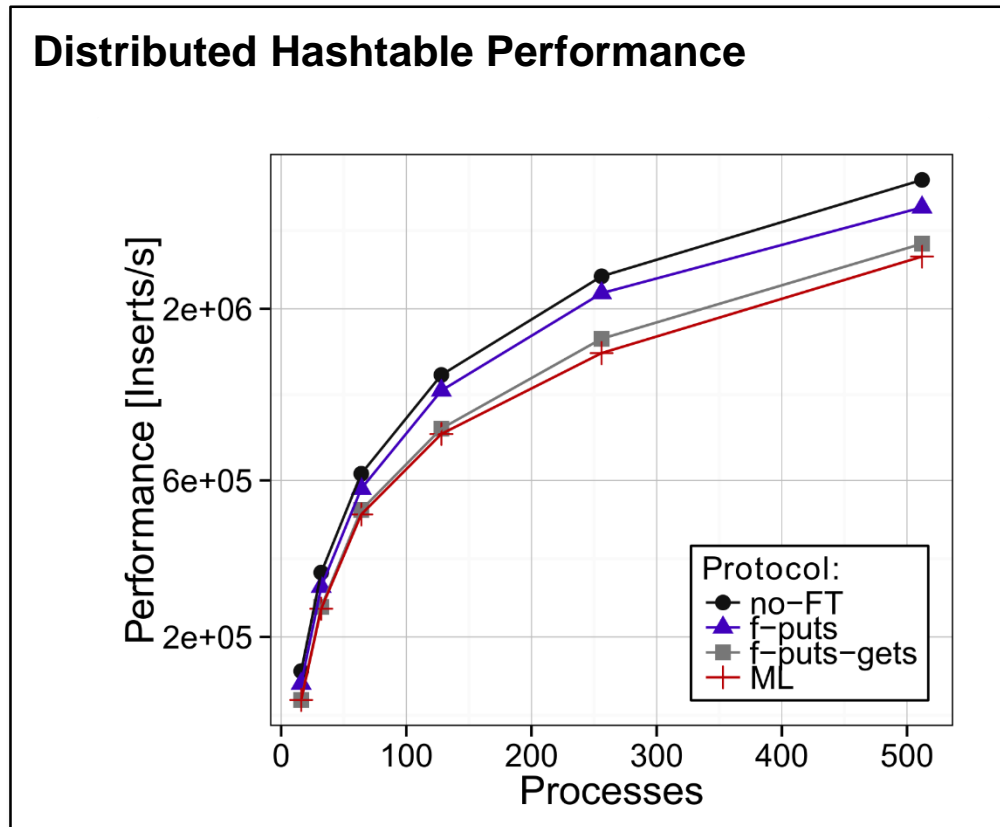
PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE



PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE

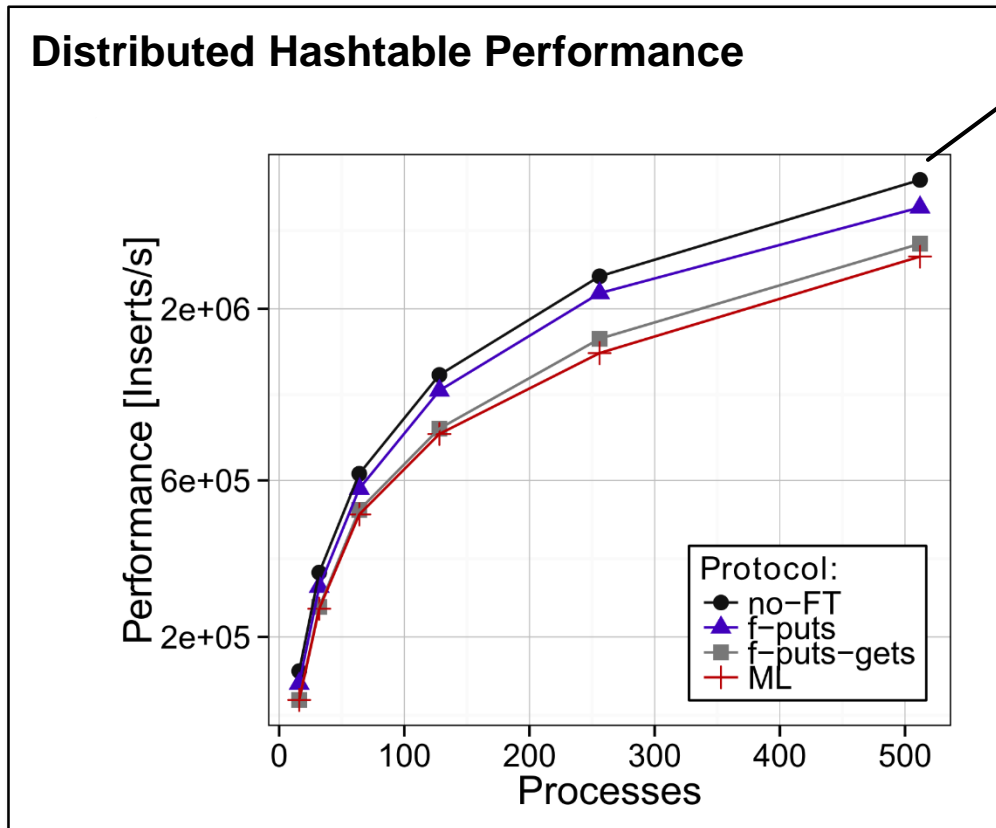


PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE



An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

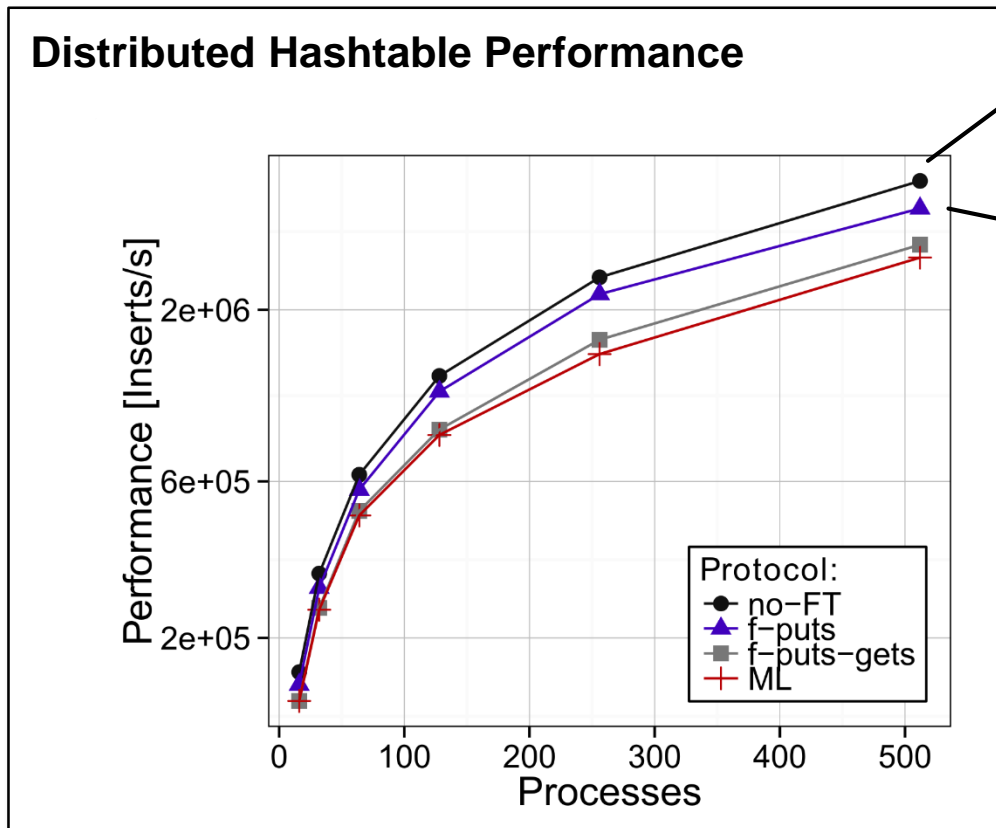
PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE



no-FT: no fault tolerance

An insert: 1 get and 1 put are logged
 A collision: 4 gets and 6 puts are logged

PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE

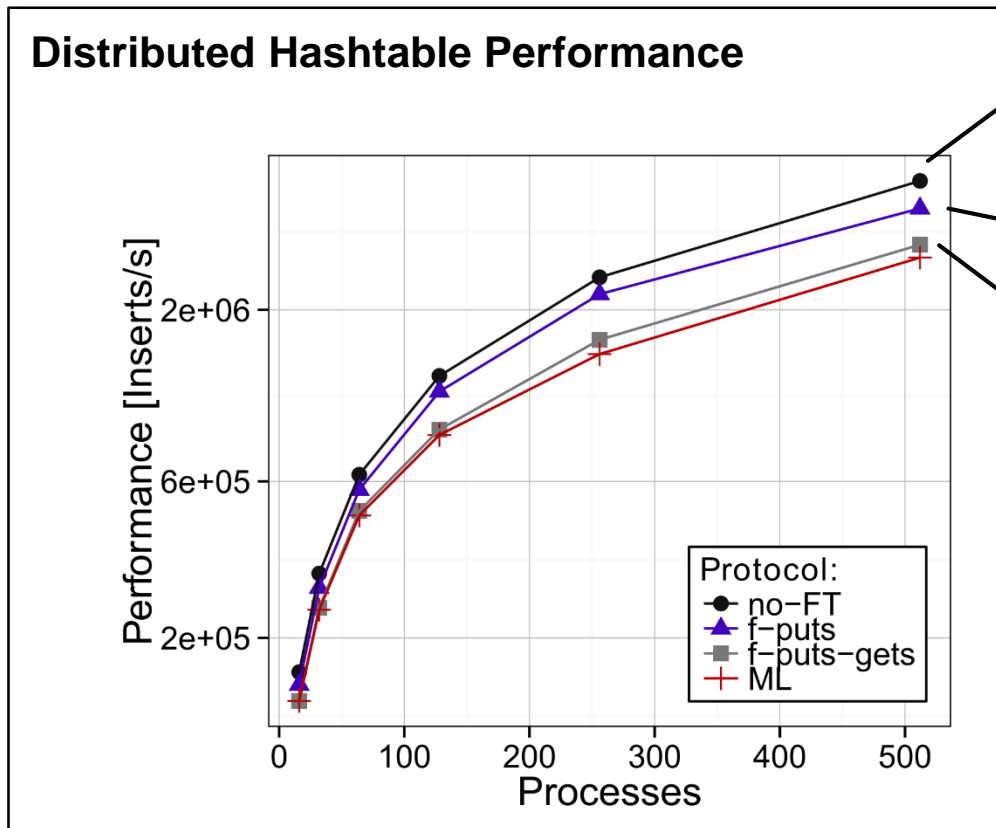


no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE



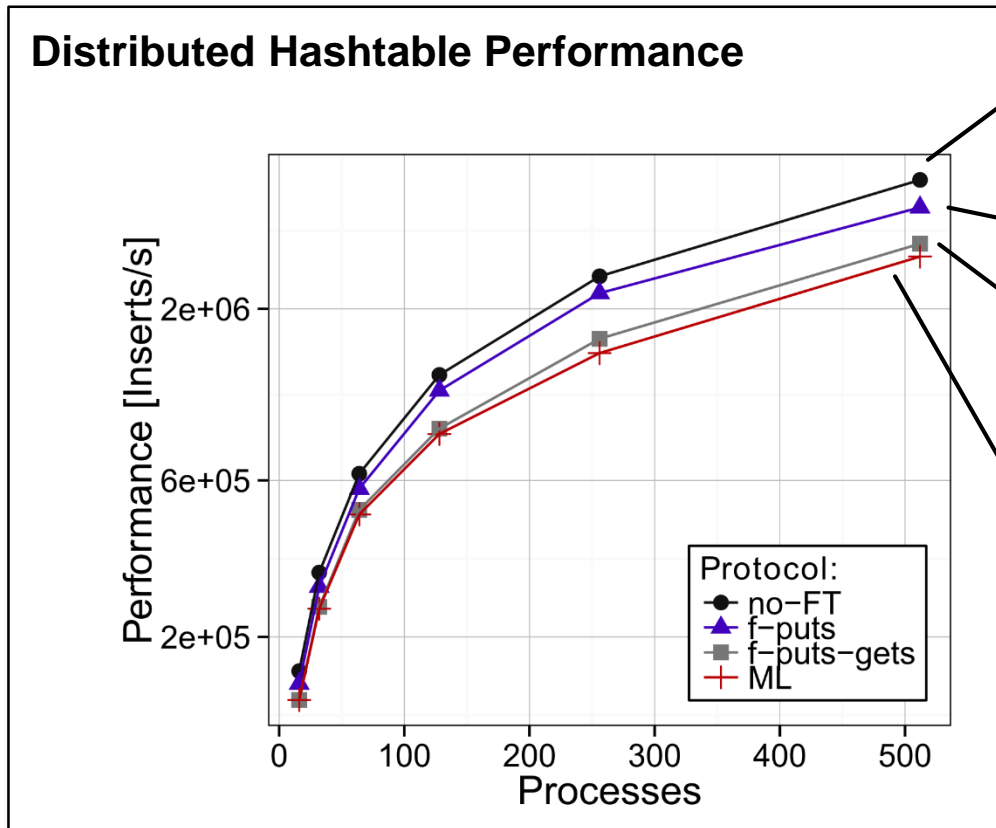
no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

f-puts-gets: logging puts
and gets
33% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE



no-FT: no fault tolerance

f-puts: logging puts
12% slower than no-FT

f-puts-gets: logging puts
and gets
33% slower than no-FT

ML: logging puts and gets
40% slower than no-FT

An insert: 1 get and 1 put are logged
A collision: 4 gets and 6 puts are logged

OVERVIEW OF OUR RESEARCH

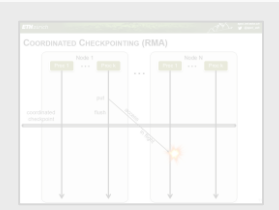
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

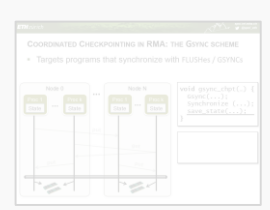
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

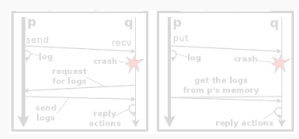


MP vs. RMA

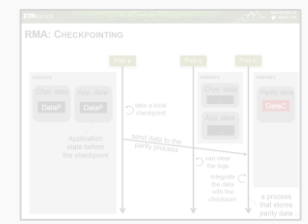


Schemes

UC in RMA

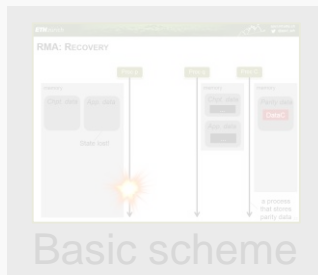


MP vs. RMA



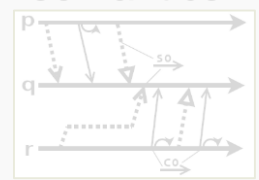
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

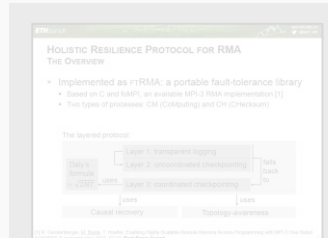
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

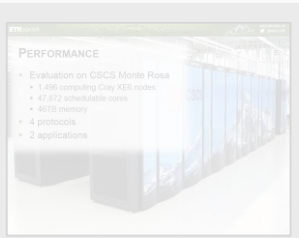
Holistic fault-tolerance library



Design

Checkpoints on demand

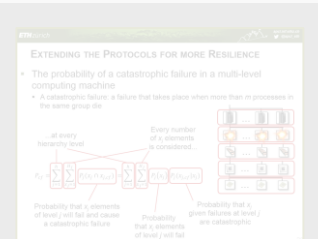
Optimizations



Performance



Distribution of processes



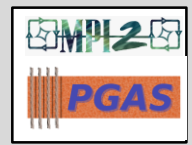
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

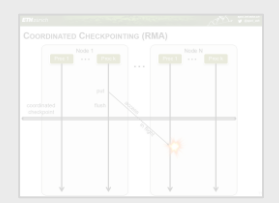
Generic model

$$D = \langle P, \mathcal{E}, \mathcal{S}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

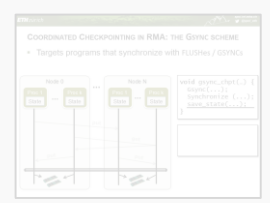
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

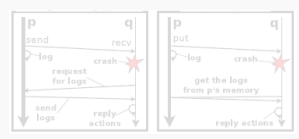


MP vs. RMA

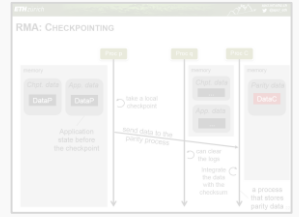


Schemes

UC in RMA

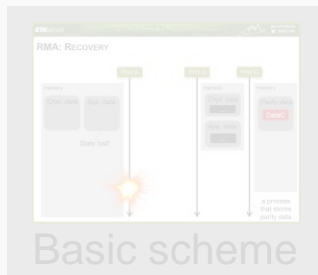


MP vs. RMA



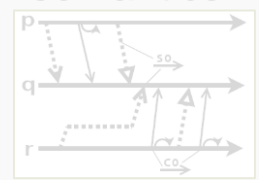
Checkpointing schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order to as the gsync order).*

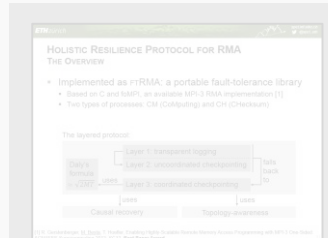
Deadlock freedom
Correct recovery

Topology-awareness

$$\langle P, \mathcal{E}, \mathcal{S}, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

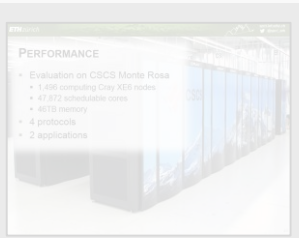
Holistic fault-tolerance library



Design

Checkpoints on demand

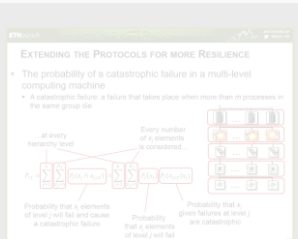
Optimizations



Performance



Distribution of processes



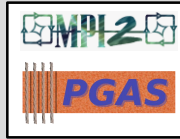
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

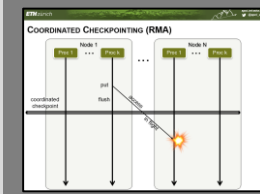
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

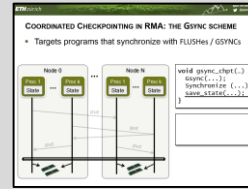
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

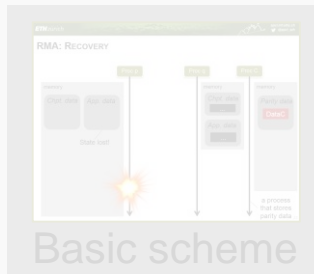


MP vs. RMA



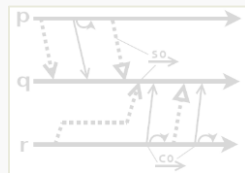
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery

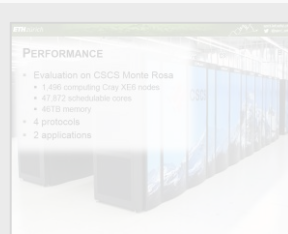
Holistic fault-tolerance library



Design

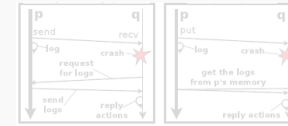
Checkpoints on demand

Optimizations

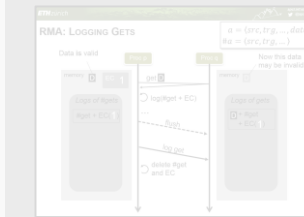


Performance

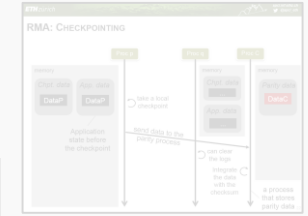
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

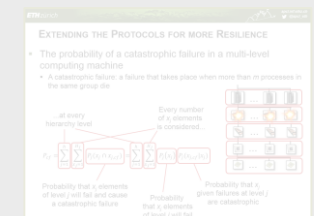
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



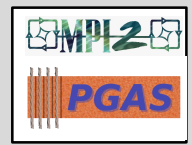
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

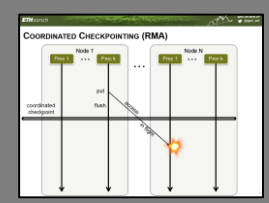
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

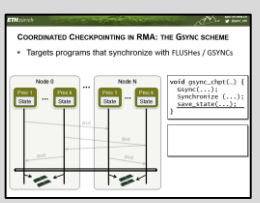
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

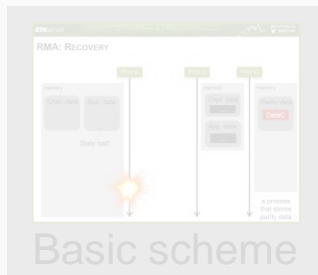


MP vs. RMA



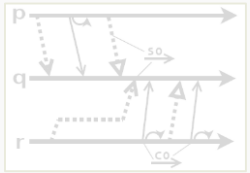
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery

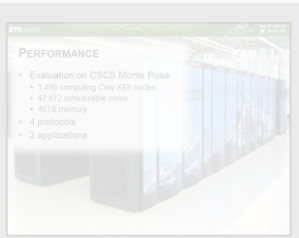
Holistic fault-tolerance library



Design

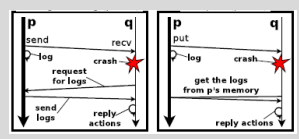
Checkpoints on demand

Optimizations

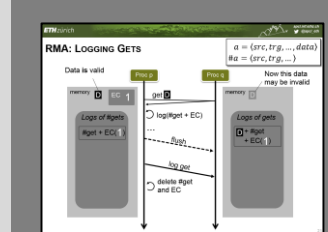


Performance

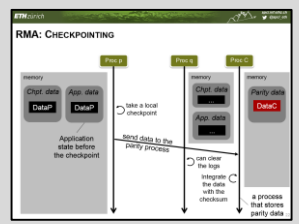
UC in RMA



MP vs. RMA



Logging accesses

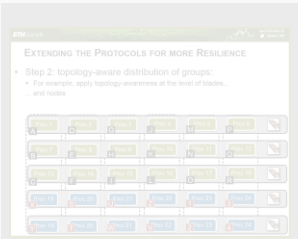


Checkpointing schemes

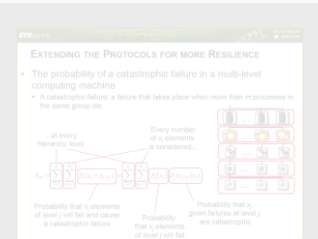
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



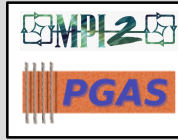
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

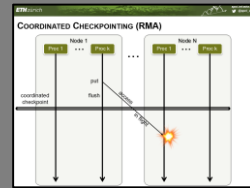
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

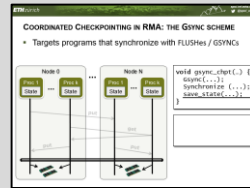
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

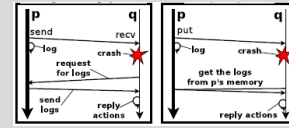


MP vs. RMA

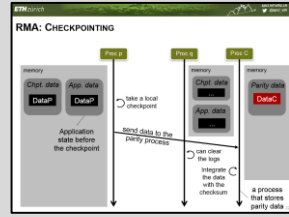


Schemes

UC in RMA

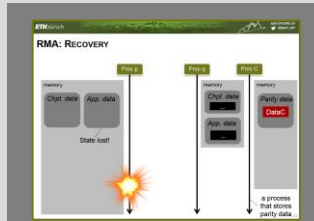


MP vs. RMA



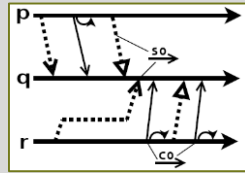
Checkpointing schemes

Recovery in RMA



Basic scheme

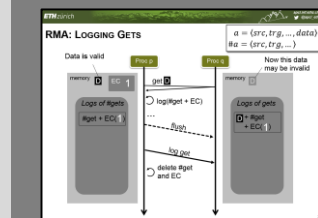
Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery



Logging accesses

Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions

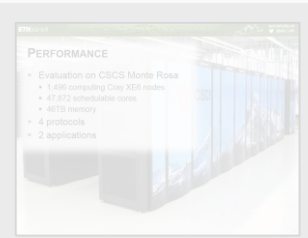
Holistic fault-tolerance library



Design

Checkpoints on demand

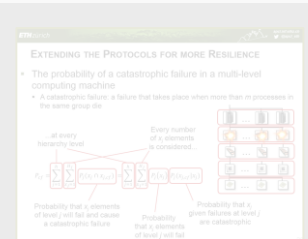
Optimizations



Performance



Distribution of processes



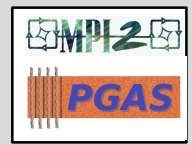
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

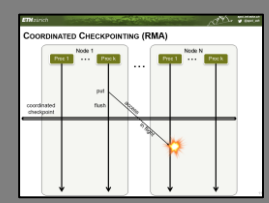
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

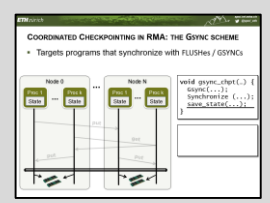
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

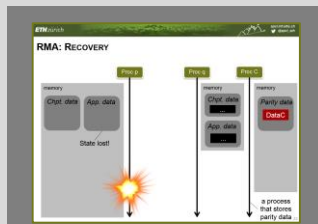


MP vs. RMA



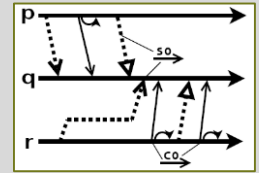
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics



Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{cohb} order as the gsync order).*

Deadlock freedom
Correct recovery

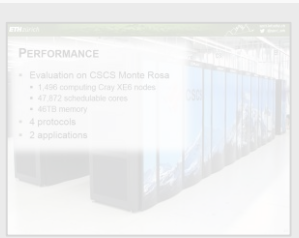
Holistic fault-tolerance library



Design

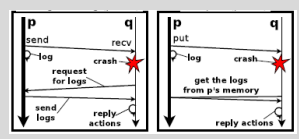
Checkpoints on demand

Optimizations

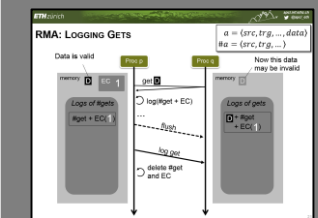


Performance

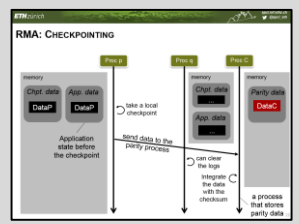
UC in RMA



MP vs. RMA



Logging accesses

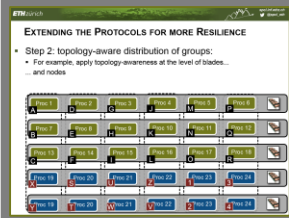


Checkpointing schemes

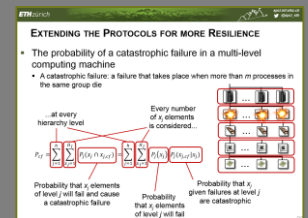
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

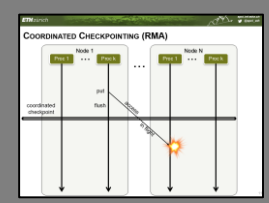
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

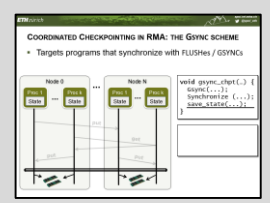
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

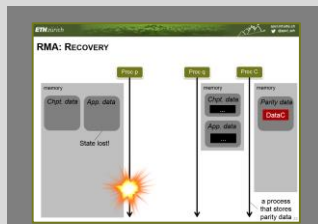


MP vs. RMA



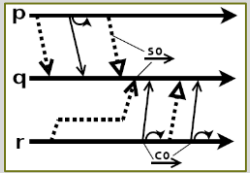
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

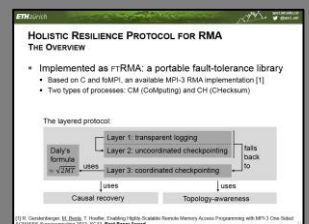


Proofs

THEOREM 4.2. *The recovery algorithm 2 preserves the \xrightarrow{cohb} order referred to as the gsync order).*

Deadlock freedom
Correct recovery

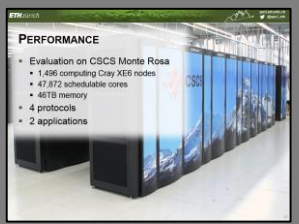
Holistic fault-tolerance library



Design

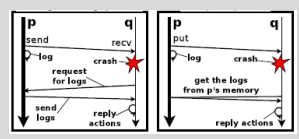
Checkpoints on demand

Optimizations

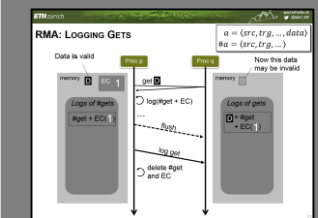


Performance

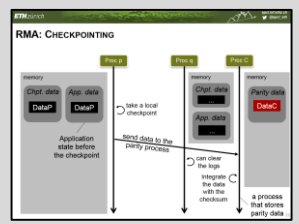
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

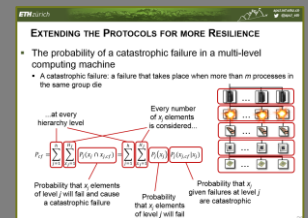
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



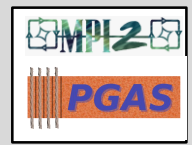
Decreasing failure prob.

OVERVIEW OF OUR RESEARCH

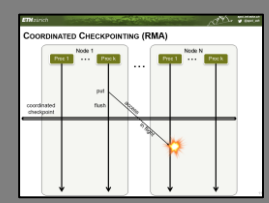
Generic model

$$D = \langle P, \mathcal{E}, S, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co} \rangle,$$

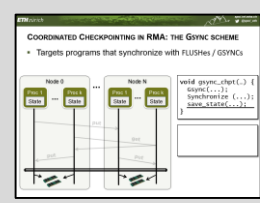
PUT($p \Rightarrow q$)
GET($p \Leftarrow q$)



CC in RMA

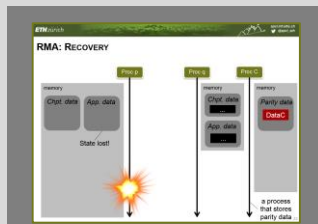


MP vs. RMA



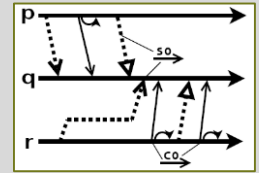
Schemes

Recovery in RMA



Basic scheme

Extended RMA semantics

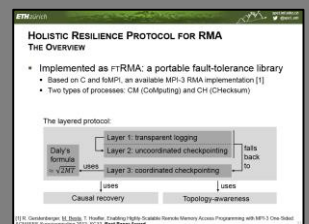


Proofs

THEOREM 4.2. *The recovery rithm 2 preserves the coh order to as the gsync order).*

Deadlock freedom
Correct recovery

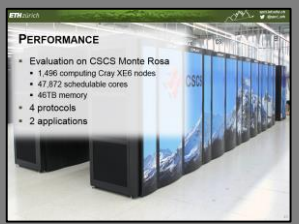
Holistic fault-tolerance library



Design

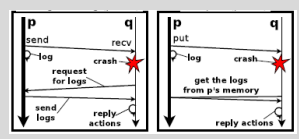
Checkpoints on demand

Optimizations

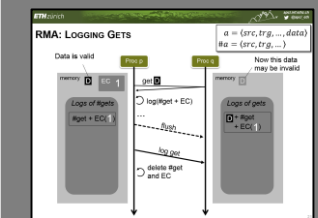


Performance

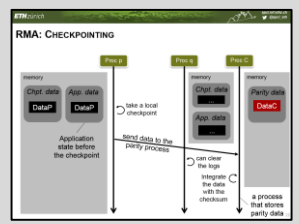
UC in RMA



MP vs. RMA



Logging accesses



Checkpointing schemes

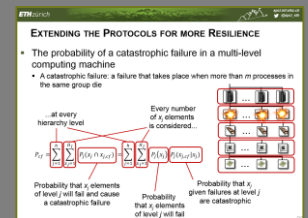
Topology-awareness

$$\langle P, \mathcal{E}, S, \mathcal{H}, \mathcal{G}, \xrightarrow{po}, \xrightarrow{so}, \xrightarrow{hb}, \xrightarrow{co}, \mathcal{M} \rangle$$

Model extensions



Distribution of processes



Decreasing failure prob.

ACKNOWLEDGMENTS

Thanks to:

Paul Hargrove (and the whole UPC team)
and the MPI Forum RMA WG ...

... and the institutions:



CSCS



Google™

PERFORMANCE: ACCESS LOGGING DISTRIBUTED HASHTABLE

no-FT: no fault tolerance
 f-puts: logging puts 12% slower than no-FT
 f-puts-gets: logging puts and gets

PERFORMANCE: DEMAND CHECKPOINTING NAS 3D FFT

How the log size impacts the number of demand checkpoints and the performance of the code?

[CH] = 12.5%[CM]

[1] Nishitani et al. Scaling communication-intensive applications on BlueGene/P using one-sided communication and overlap (PDPSS'09)

PERFORMANCE: ACCESS LOGGING

no-FT: no fault tolerance
 fRMA: logging puts (FFT code does not use gets) 8-9% slower than no-FT
 ML: a simple protocol based on message logging 18% slower than no-FT

[1] Performance

[2] BlueGene/P using one-sided communication and

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
- A catastrophic failure: a failure that takes place when more than m processes in

Every number of x elements is considered...

$P_i(x)$ $P_i(x_{i,j} | x)$

HOLISTIC RESILIENCE PROTOCOL FOR RMA THE OVERVIEW

Implemented as fRMA: a portable fault-tolerance library

- Based on C and foMPI, an available MPI-3 RMA implementation [1]

The layered protocol:

- transparent logging
- uncoordinated checkpoint
- coordinated checkpoint
- Causal recovery

Today's supercomputers

- A single hard

[1] R. Gerstenberger, M. Botta, T. Hofer, Enabling Highly-Scalable Remote Accesses: Supercomputing 2015, Sci 13, Best Paper Award
 [2] J. Daily, A higher order estimate of the optimum checkpoint interval for resiliency, Volume 22 Issue 3, February 2008, Pages 303-312

EXTENDING THE PROTOCOLS FOR MORE RESILIENCE

- The probability of a catastrophic failure in a multi-level computing machine
- A catastrophic failure: a failure that takes place when more than m processes in the same group die

P_{cat}/day

Level 5: 4 cabinets: ...
 Level 4: 3 chassis: ...
 Level 3: 8 blades: ...
 Level 2: 4 nodes: ...
 Level 1: 32 cores: ...

Up to 128 process failures (assuming 1 process per core)

Thank you for your attention

RMA: LOGGING GETS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

Data is valid

RMA: LOGGING PUTS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

Data is valid

RMA: CHECKPOINTING The Epoch Condition

memory

Chpt. data App. data

DataP DataP

flush

take a local checkpoint

send data to the parity process

put/get

can clear the logs

integrate the data with the checksum

a process that stores parity data

memory

Parity data

DataC

memory

Logs of puts

X + #put + EC(0)

Y + #put + EC(0)

Logs of gets

D + #get + EC(1)

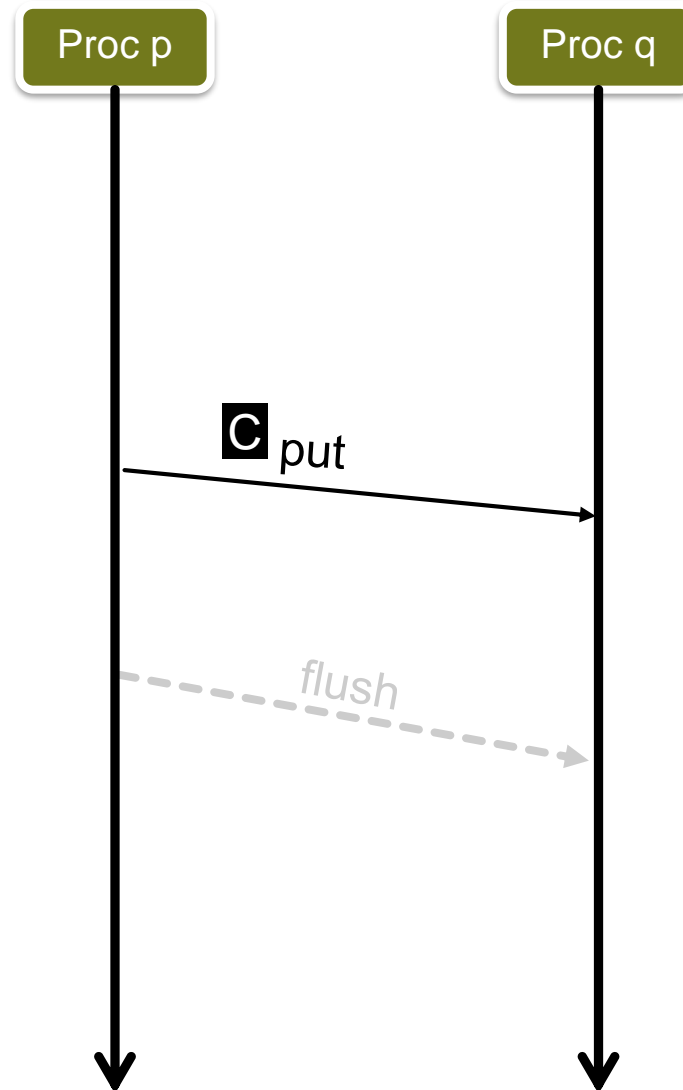
E + #get + EC(2)

F + #get + EC(2)

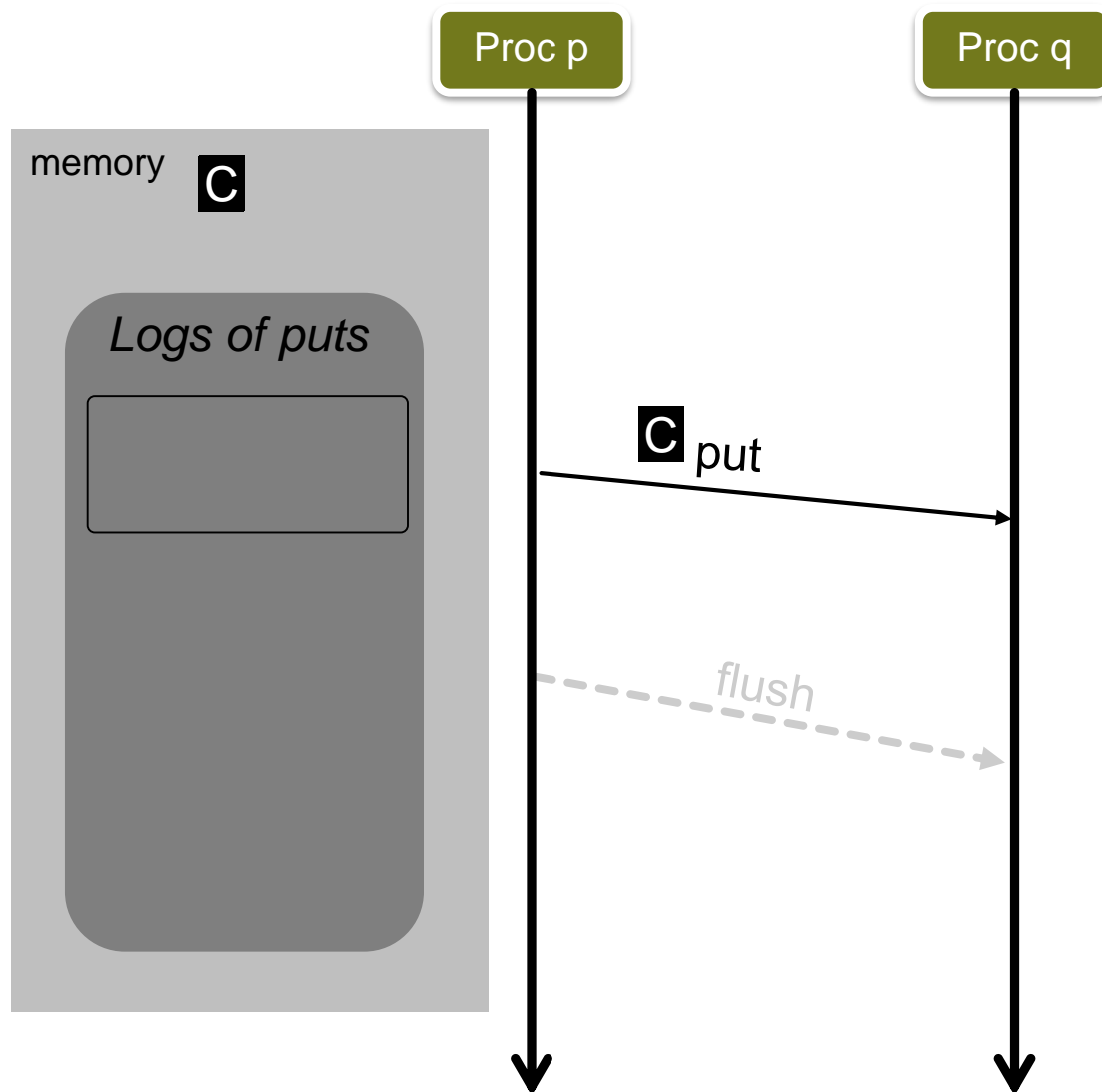
Epoch 1

Epoch 2

RMA: LOGGING PUTS

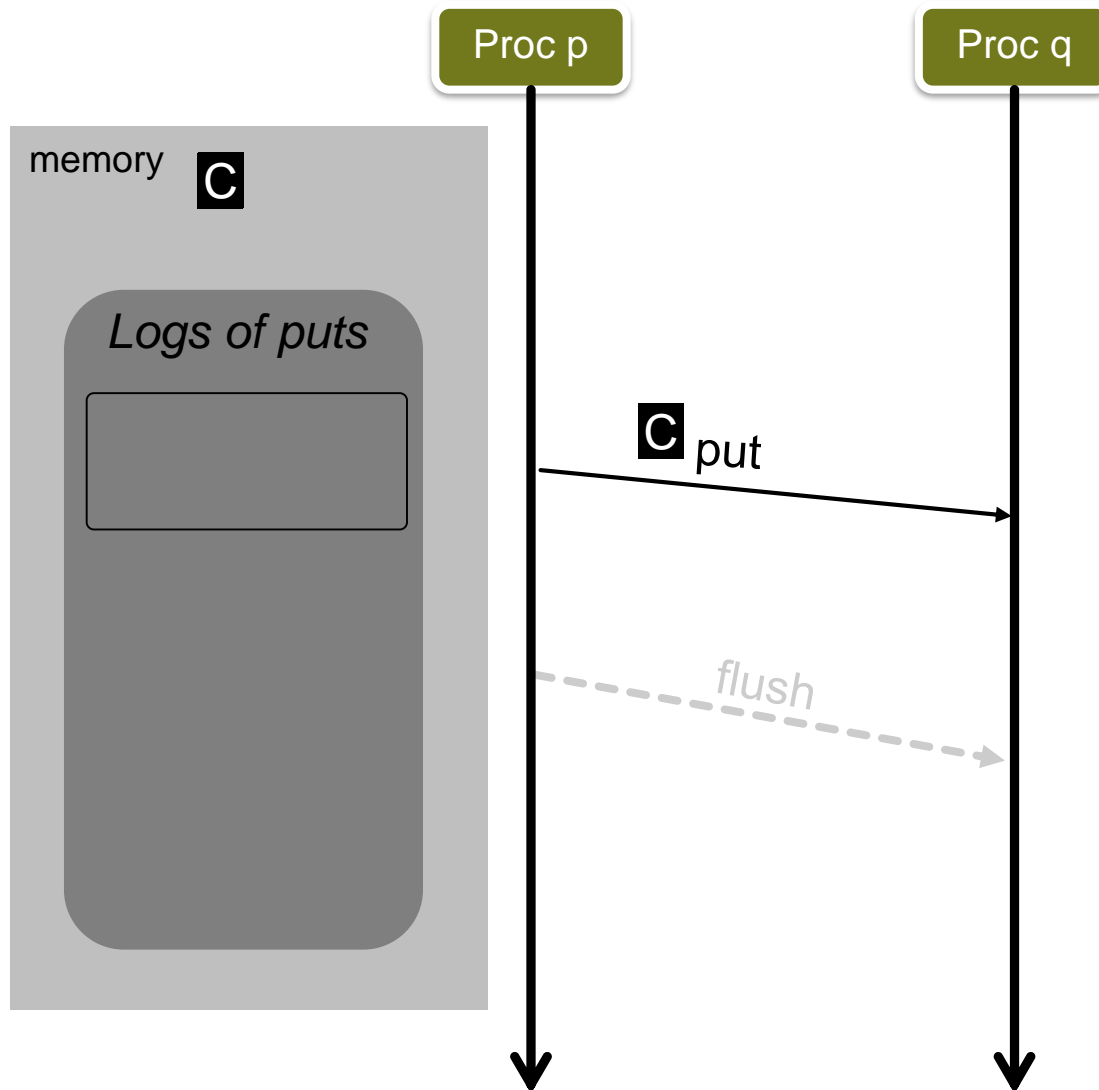


RMA: LOGGING PUTS



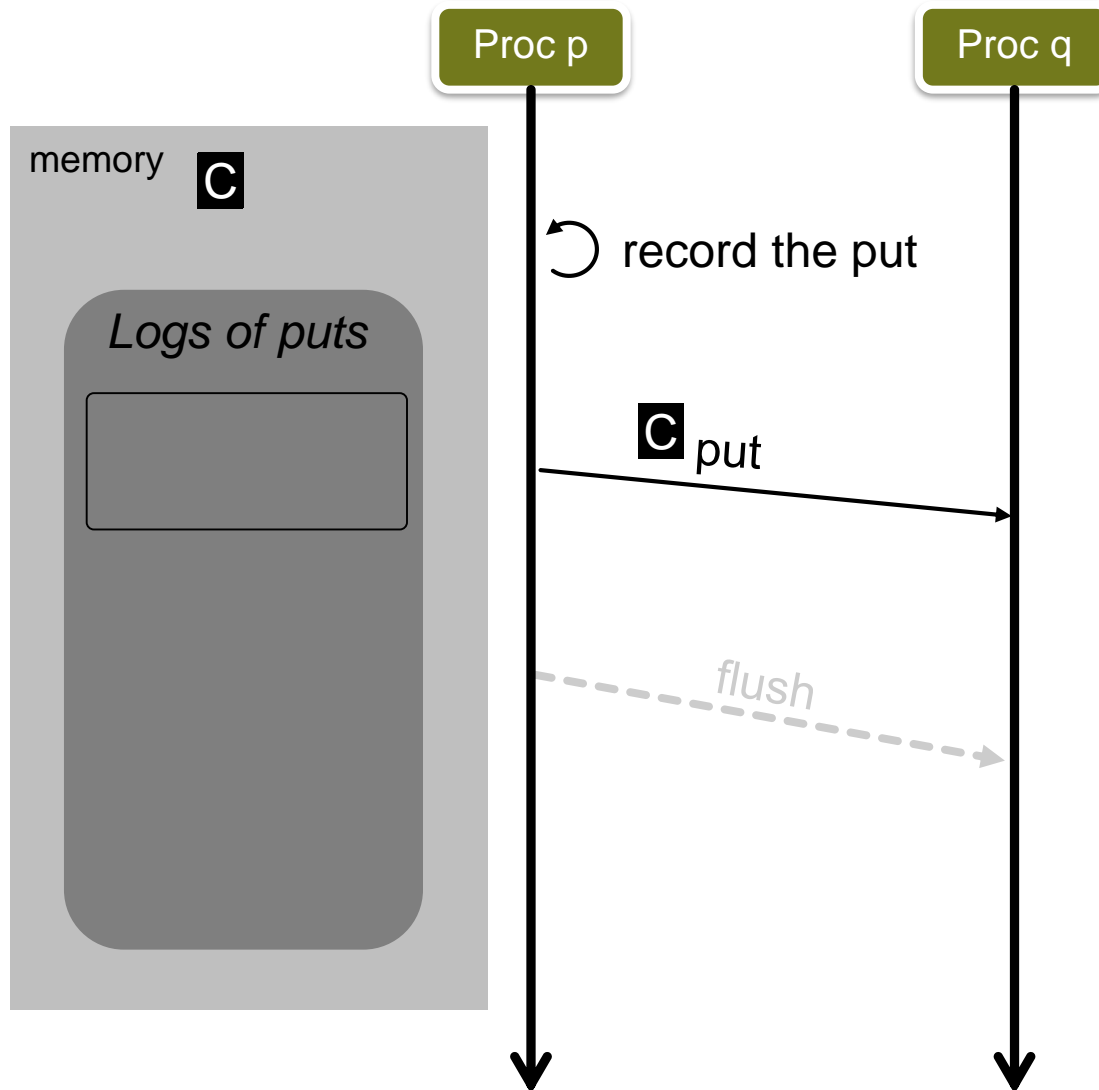
RMA: LOGGING PUTS

$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$

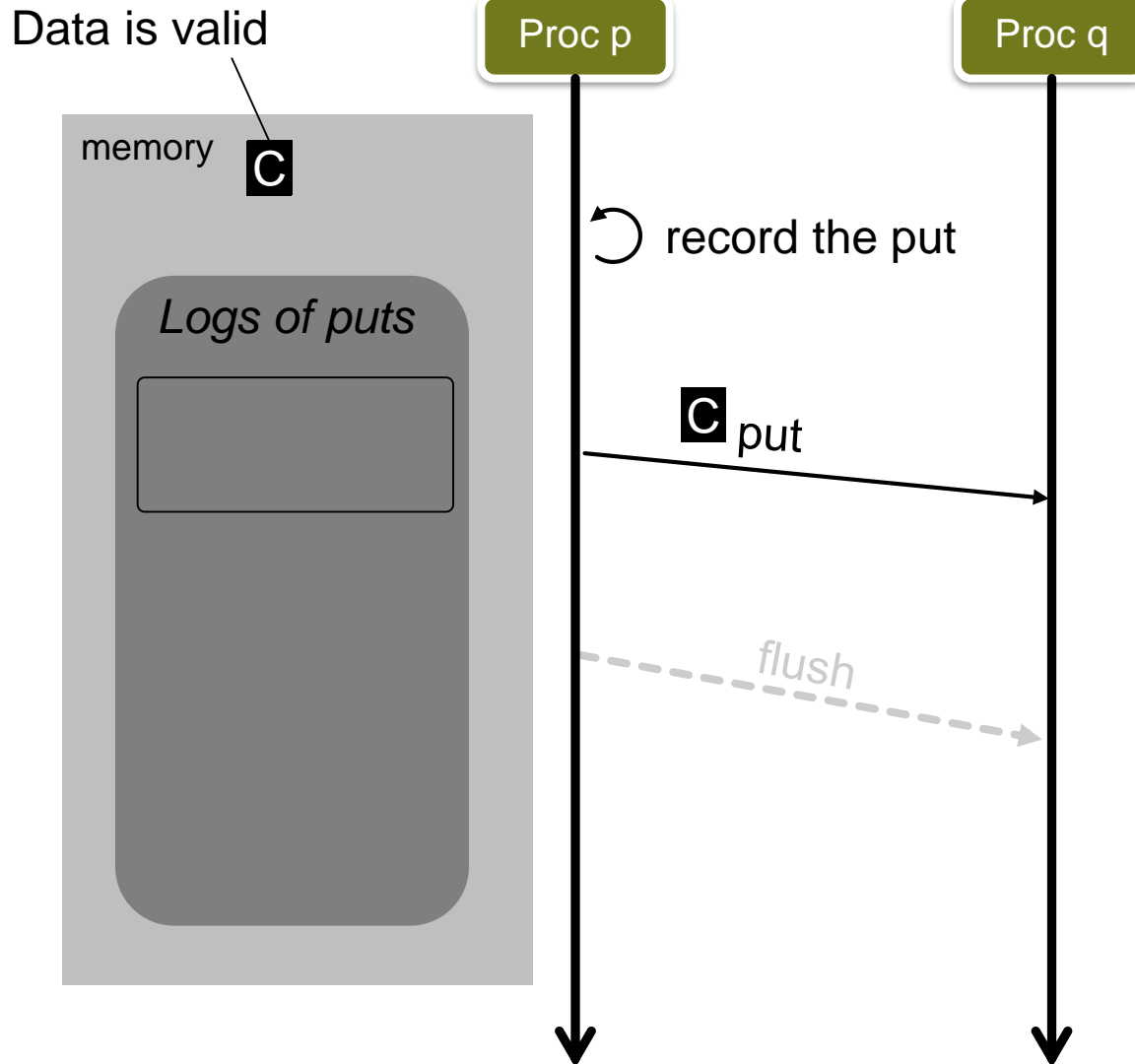


RMA: LOGGING PUTS

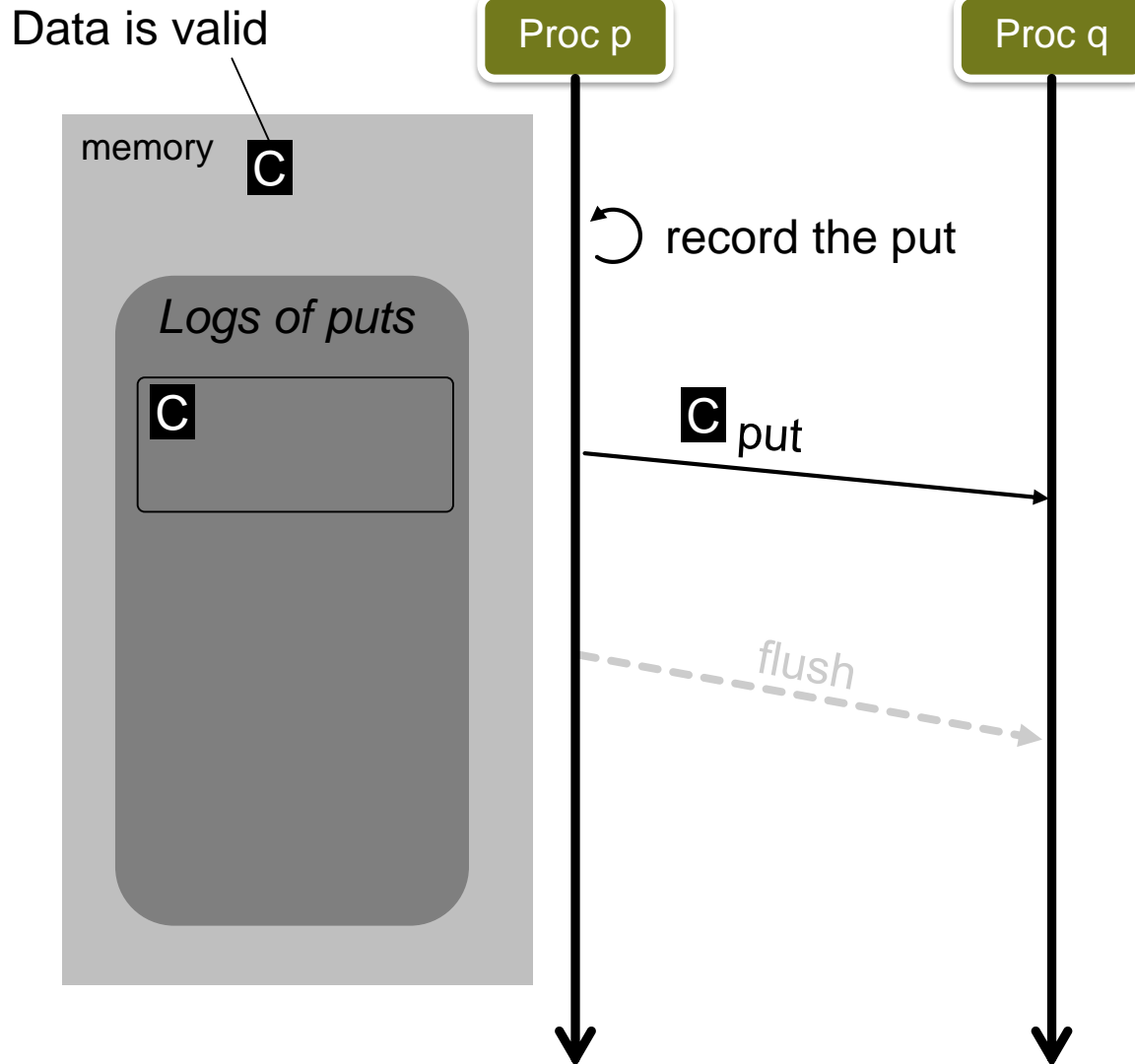
$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING PUTS

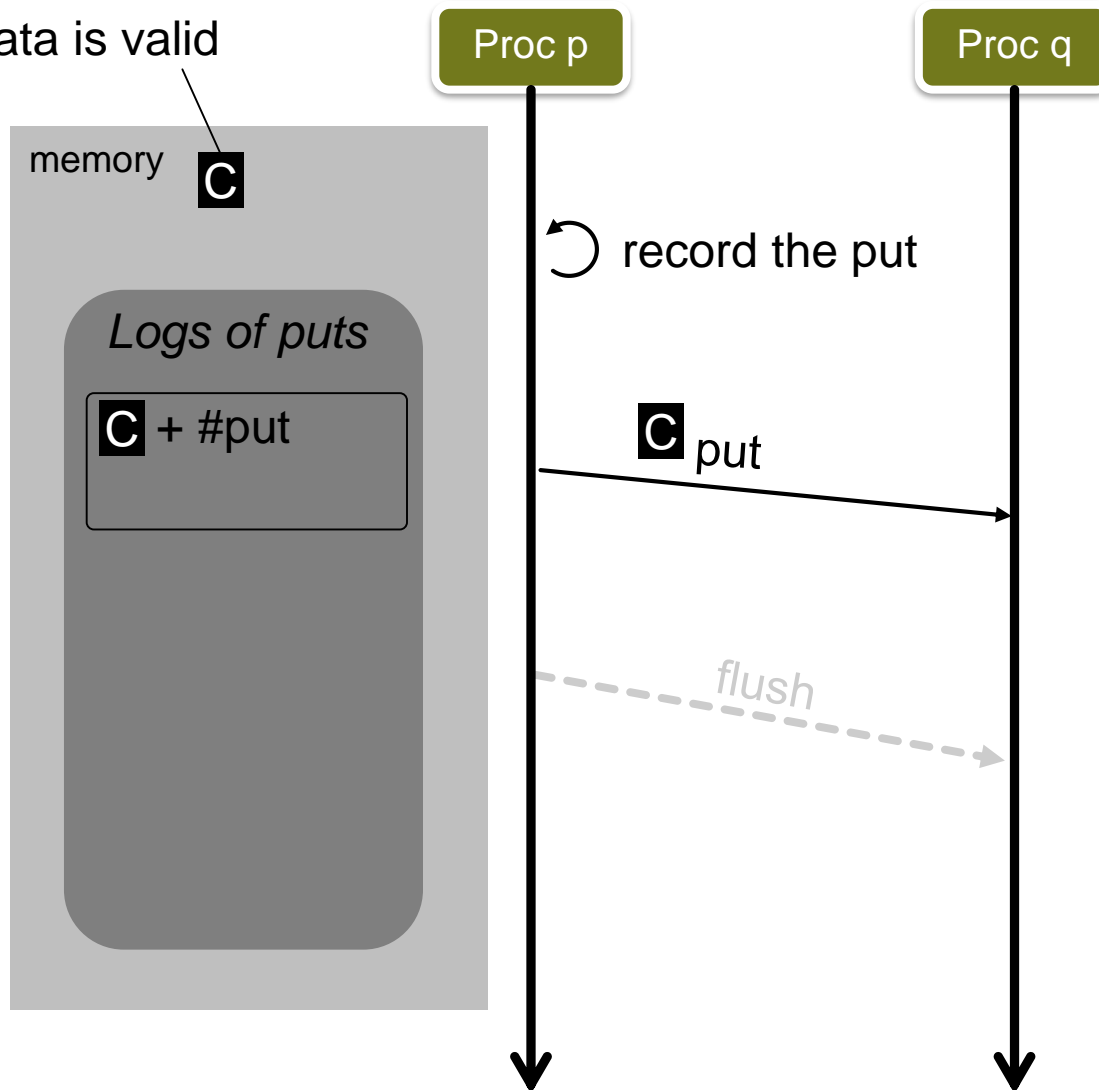
$$a = \langle src, trg, \dots, data \rangle$$
$$\#a = \langle src, trg, \dots \rangle$$


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid

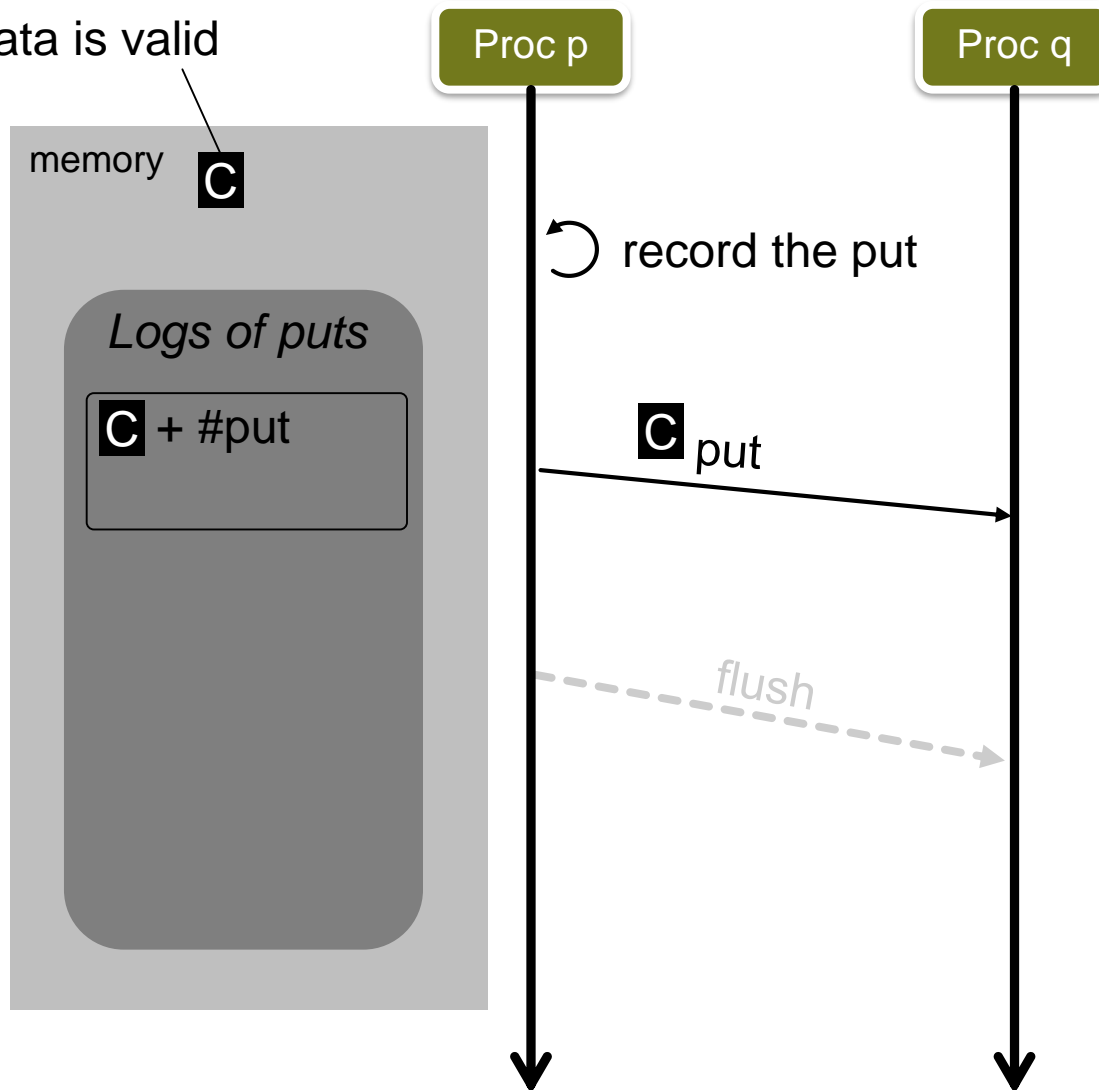


RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



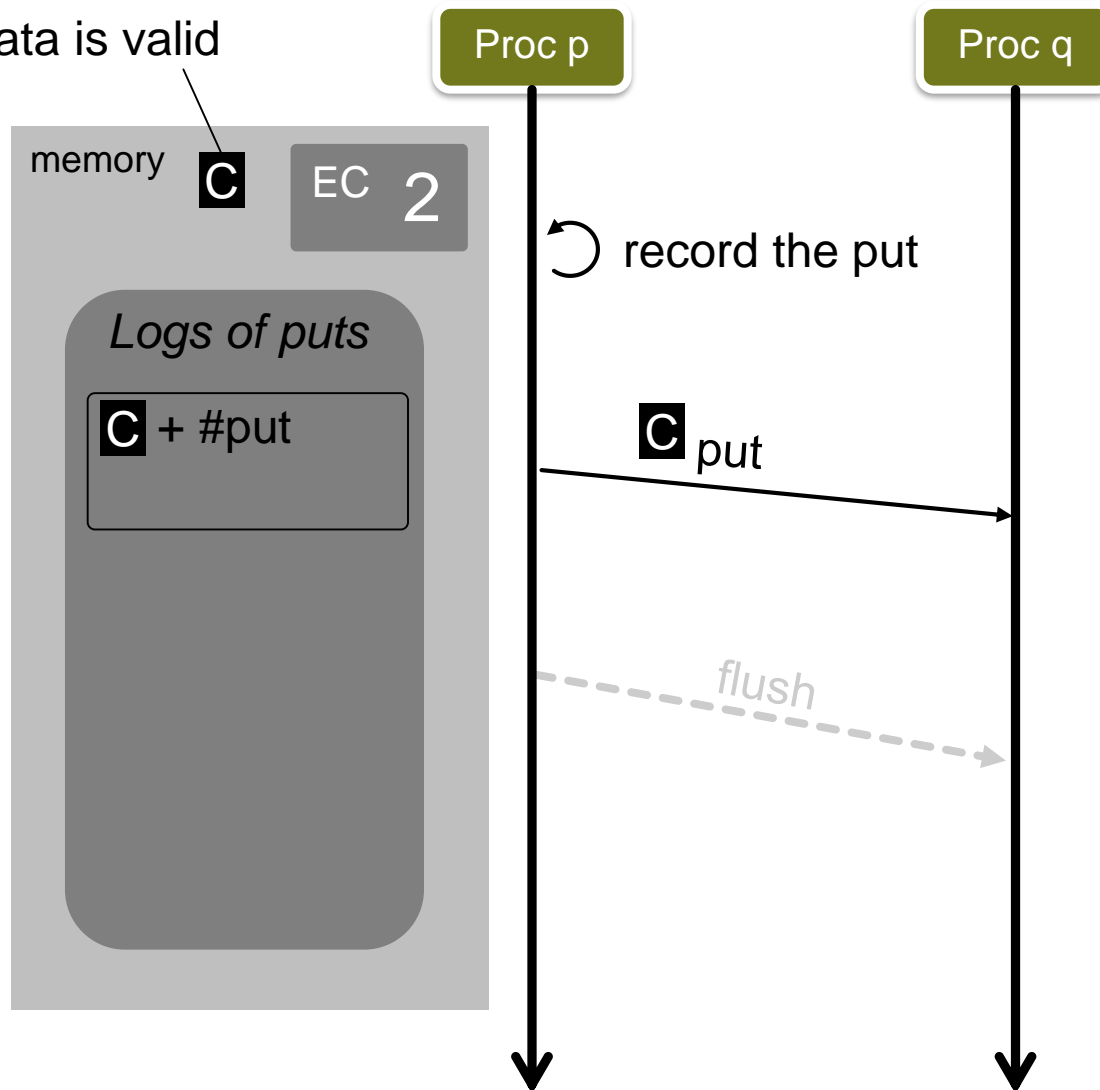
To replay the actions preserving \xrightarrow{co} , we also record **epoch counters**

RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



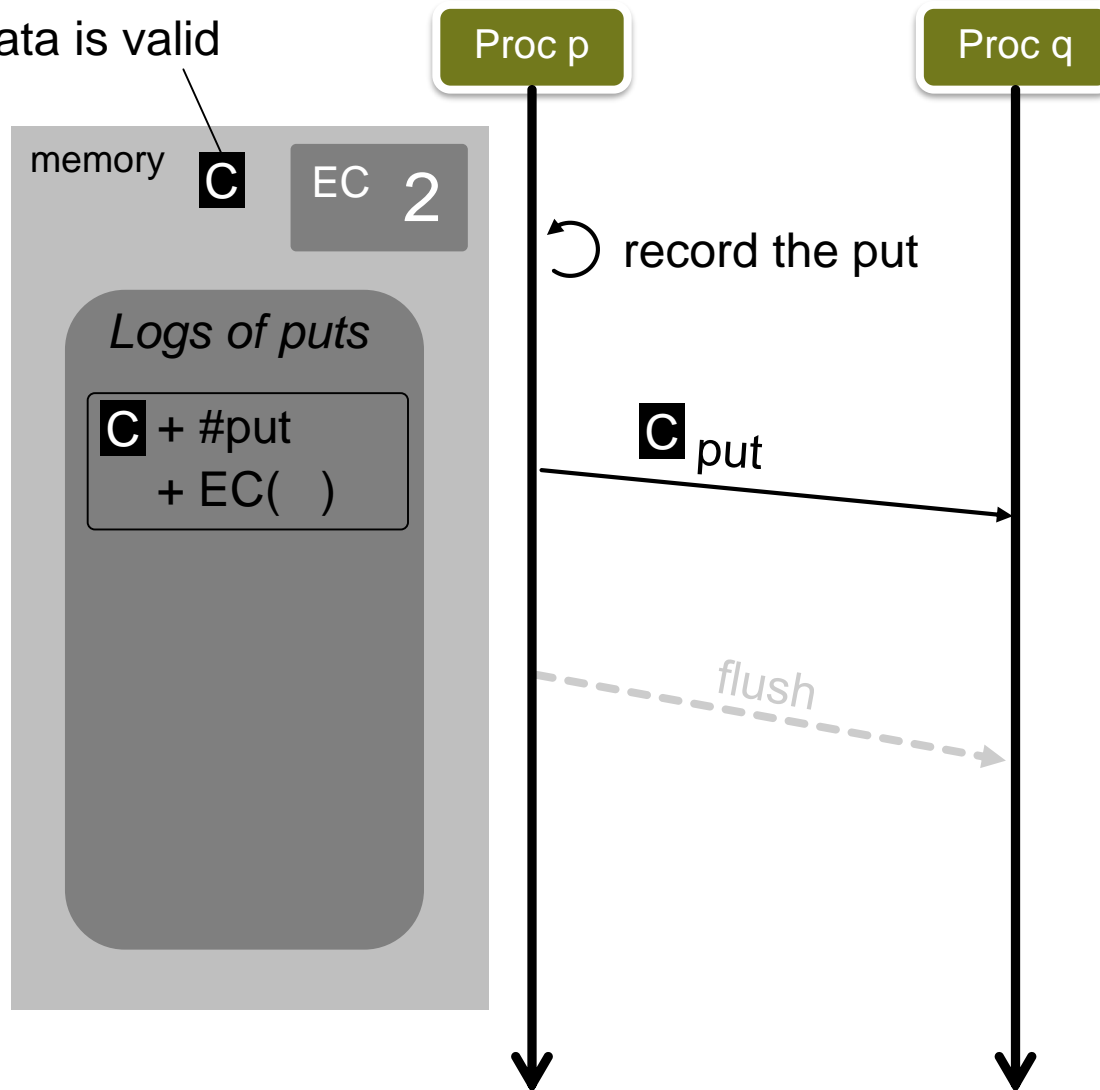
To replay the actions preserving \xrightarrow{co} , we also record **epoch counters**

RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



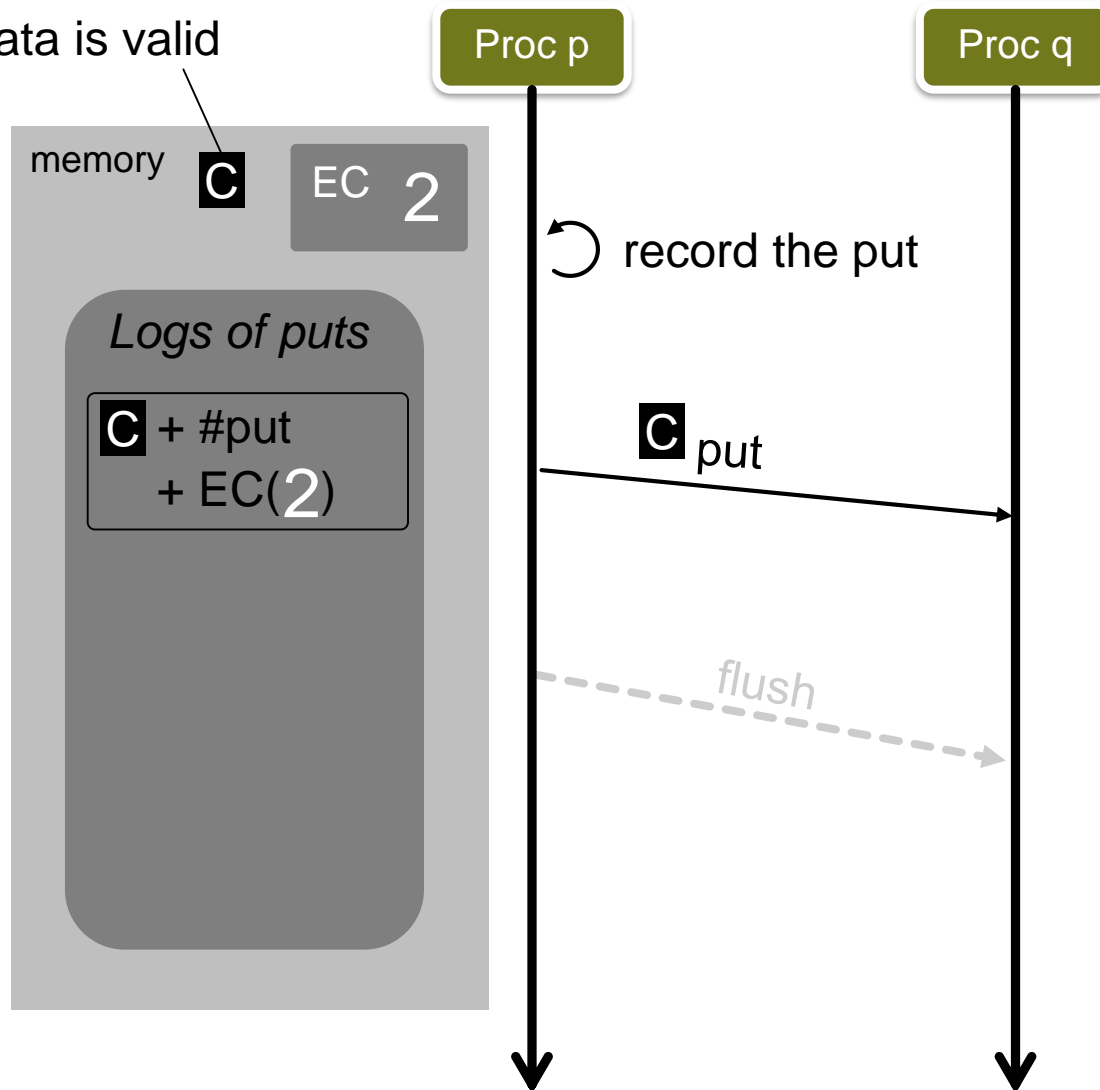
To replay the actions preserving \xrightarrow{co} , we also record **epoch counters**

RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

Data is valid



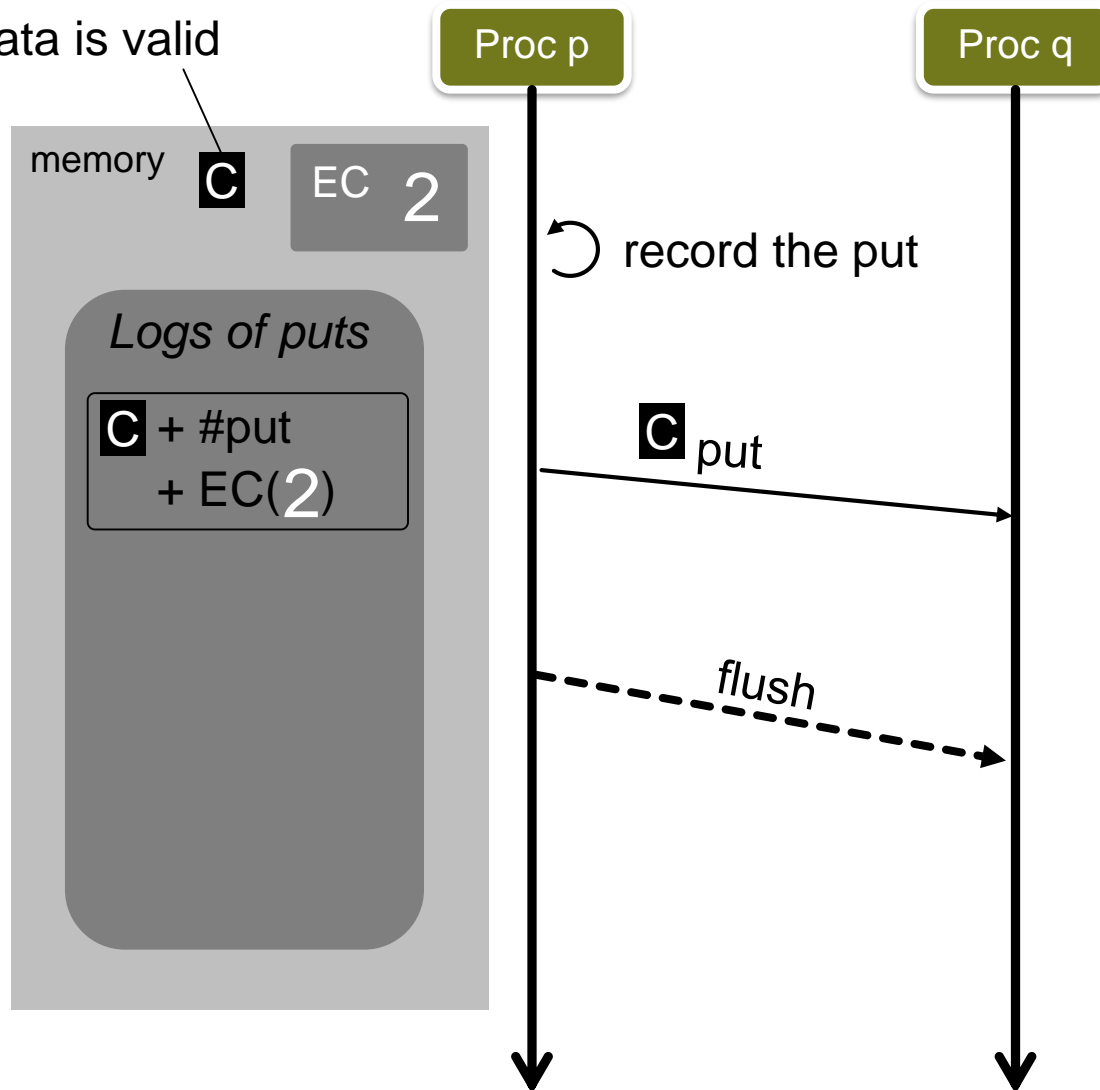
To replay the actions preserving \xrightarrow{co} , we also record **epoch counters**

RMA: LOGGING PUTS

$$a = \langle src, trg, \dots, data \rangle$$

$$\#a = \langle src, trg, \dots \rangle$$

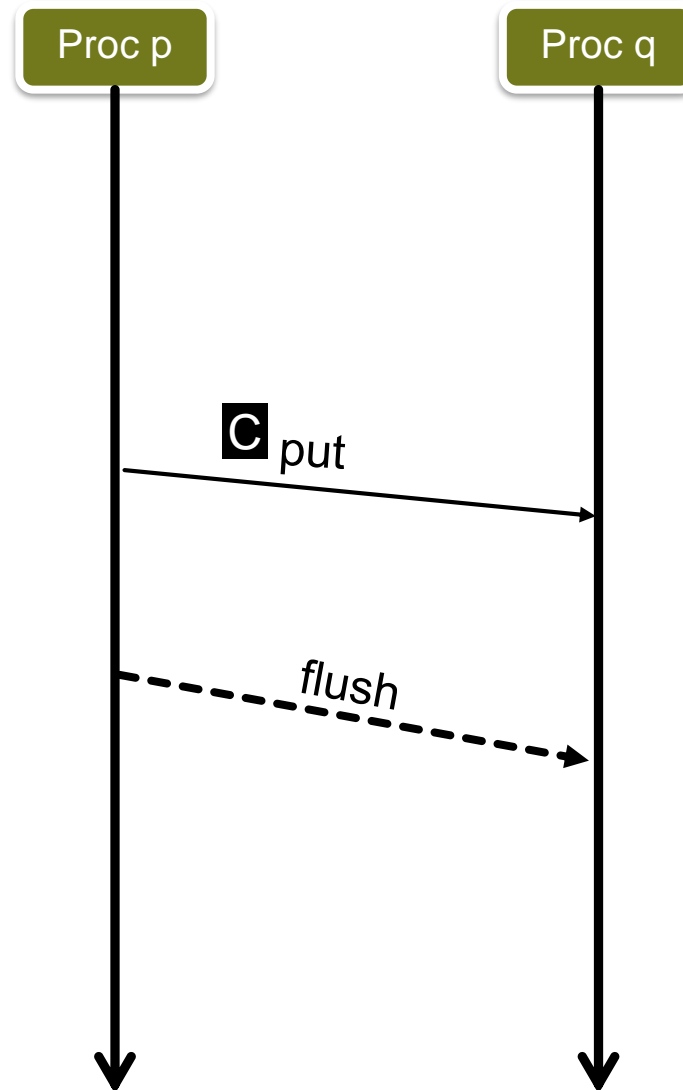
Data is valid



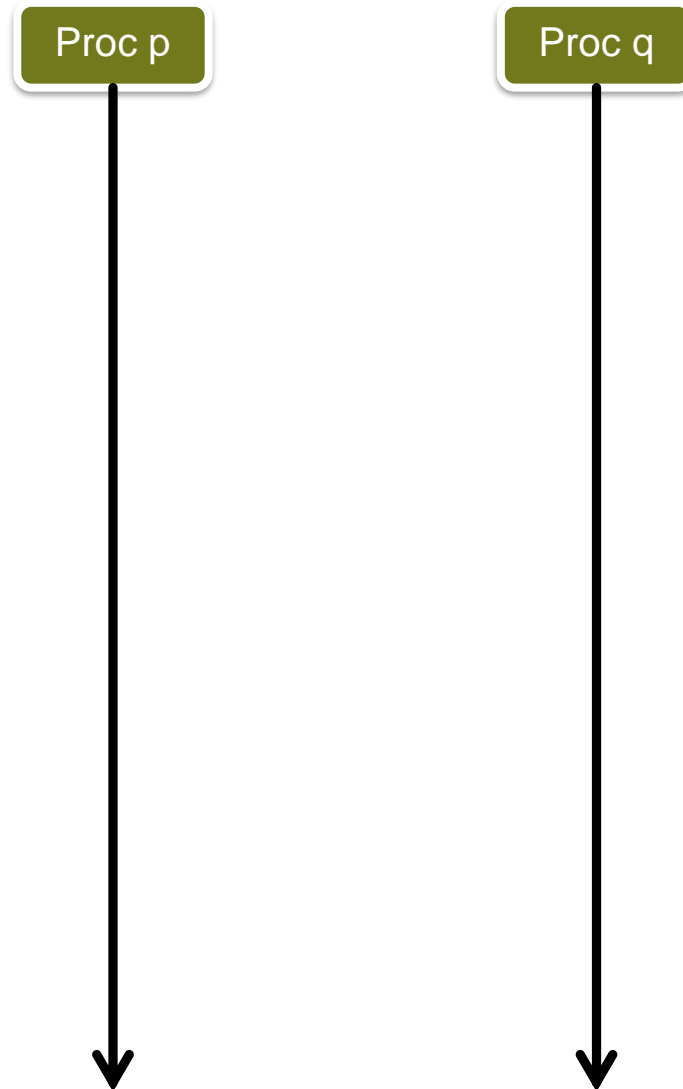
To replay the actions preserving \xrightarrow{co} , we also record **epoch counters**

RMA: LOGGING PUTS

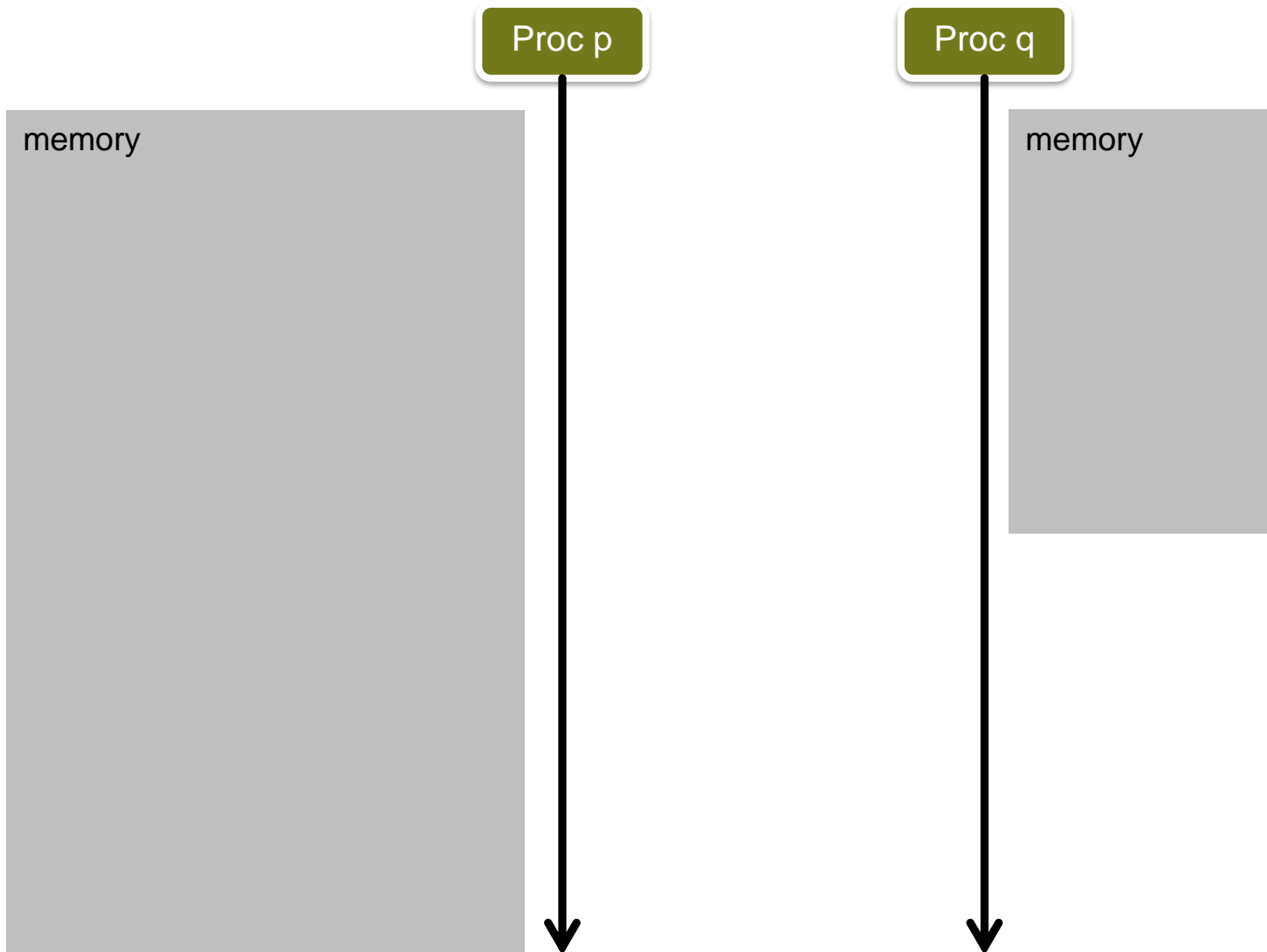
$a = \langle src, trg, \dots, data \rangle$
 $\#a = \langle src, trg, \dots \rangle$



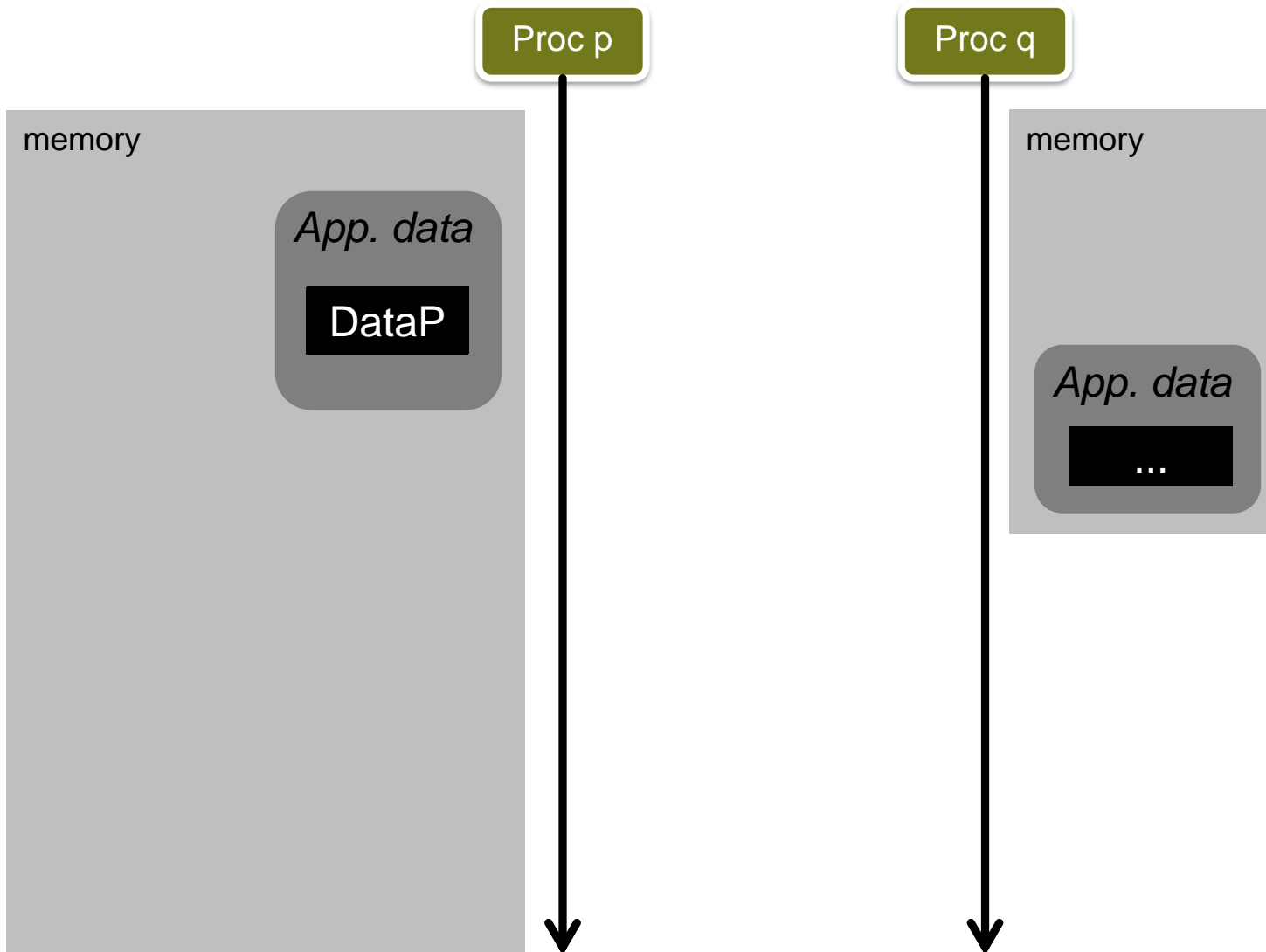
RMA: CHECKPOINTING



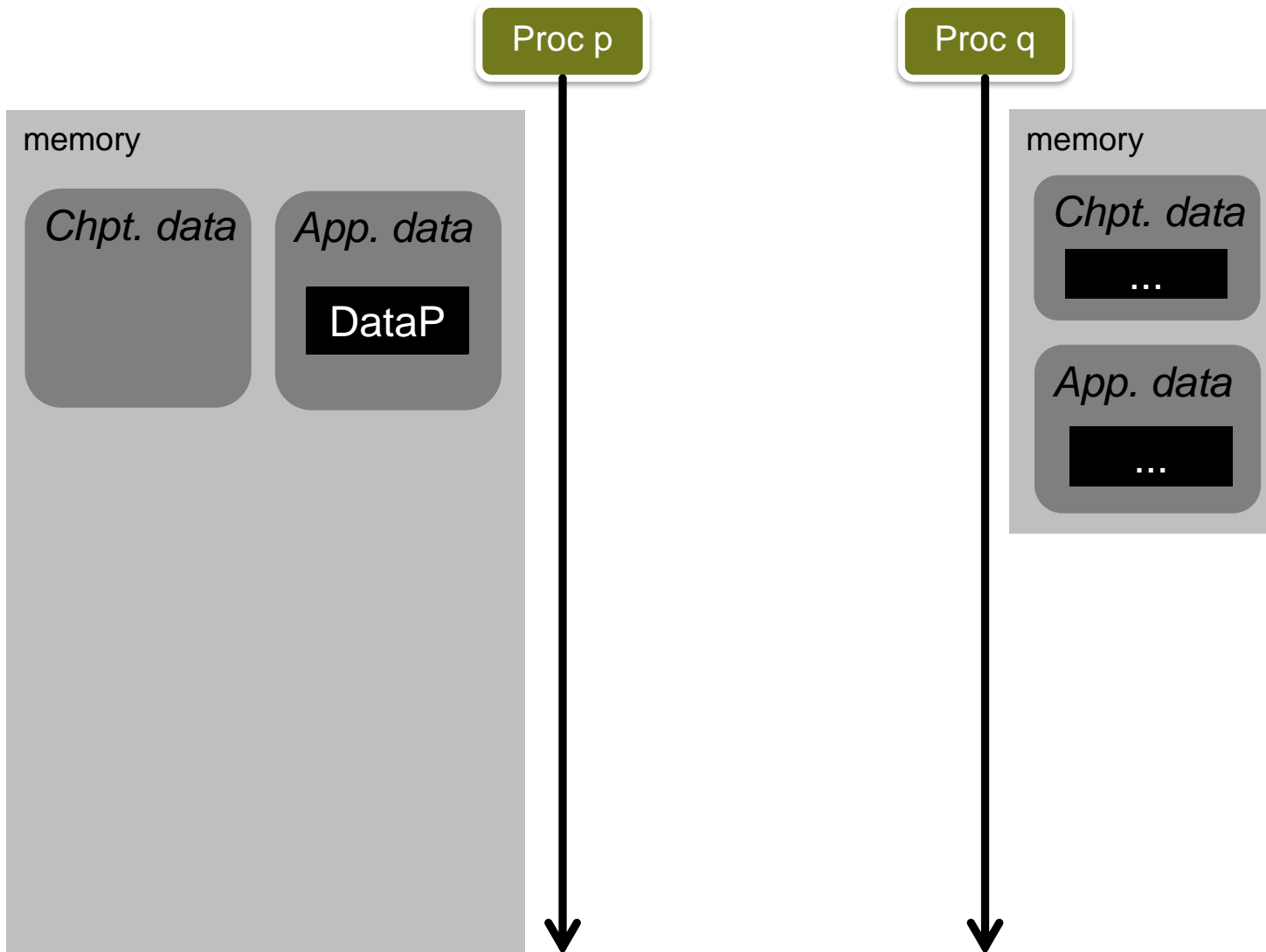
RMA: CHECKPOINTING



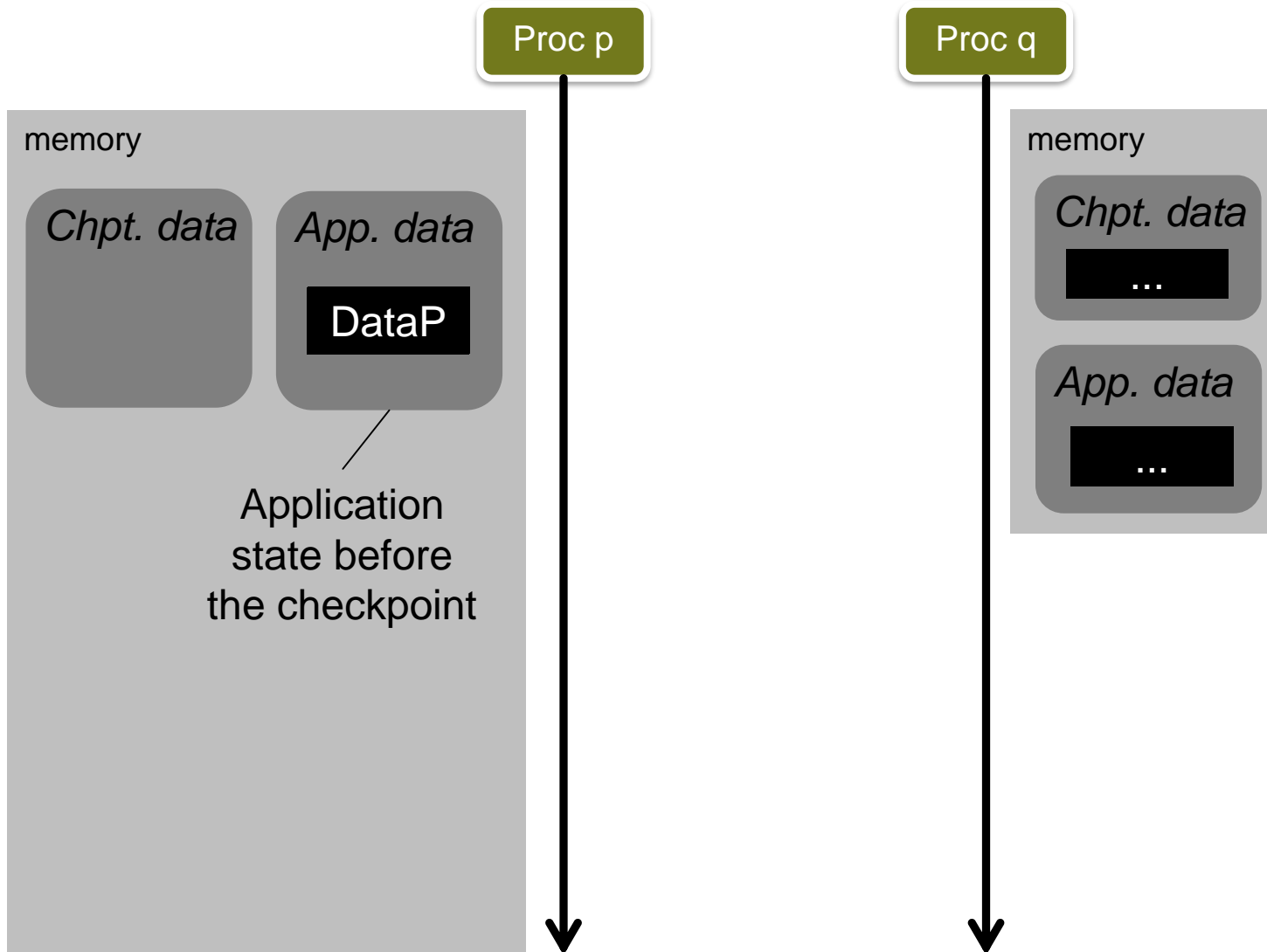
RMA: CHECKPOINTING



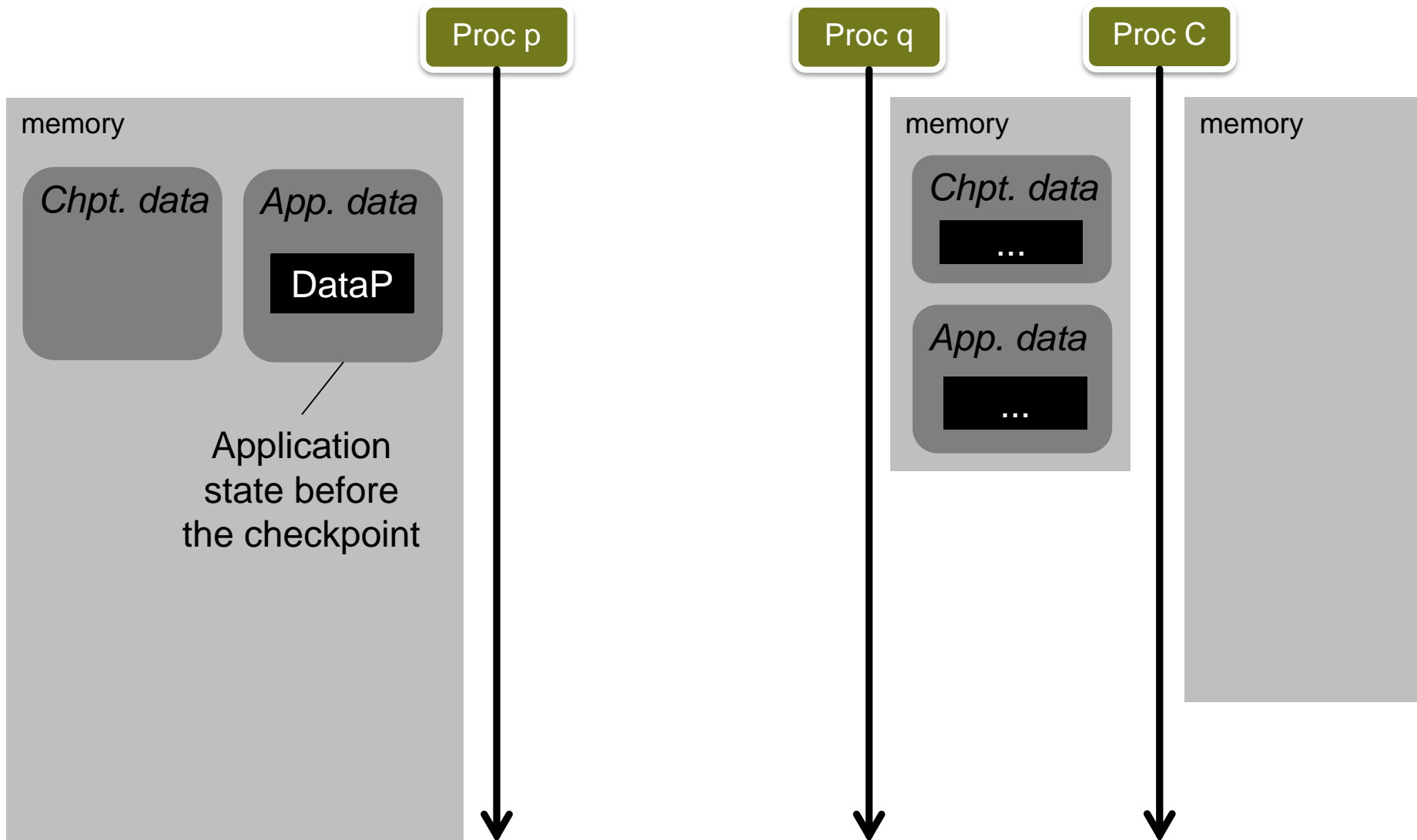
RMA: CHECKPOINTING



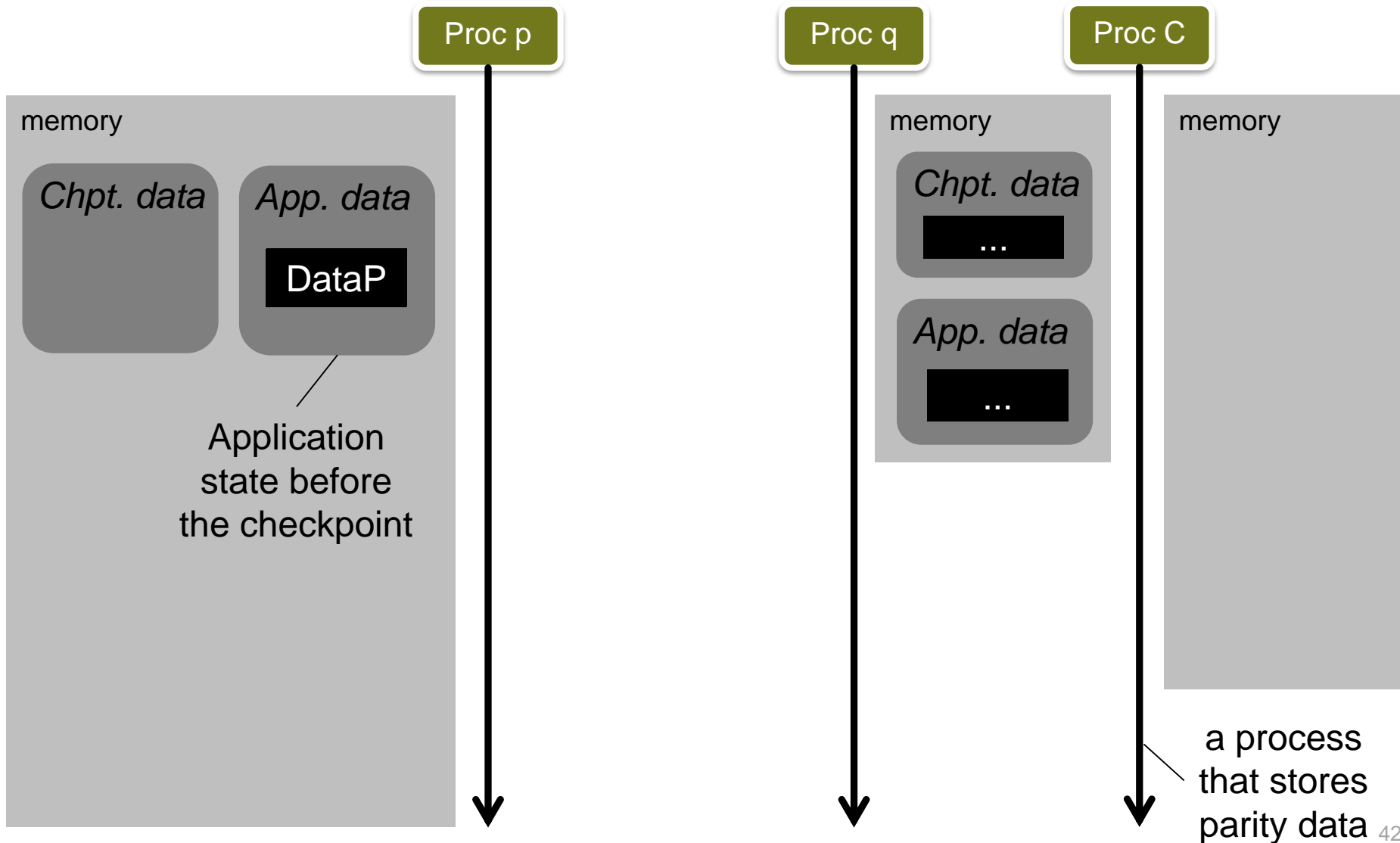
RMA: CHECKPOINTING



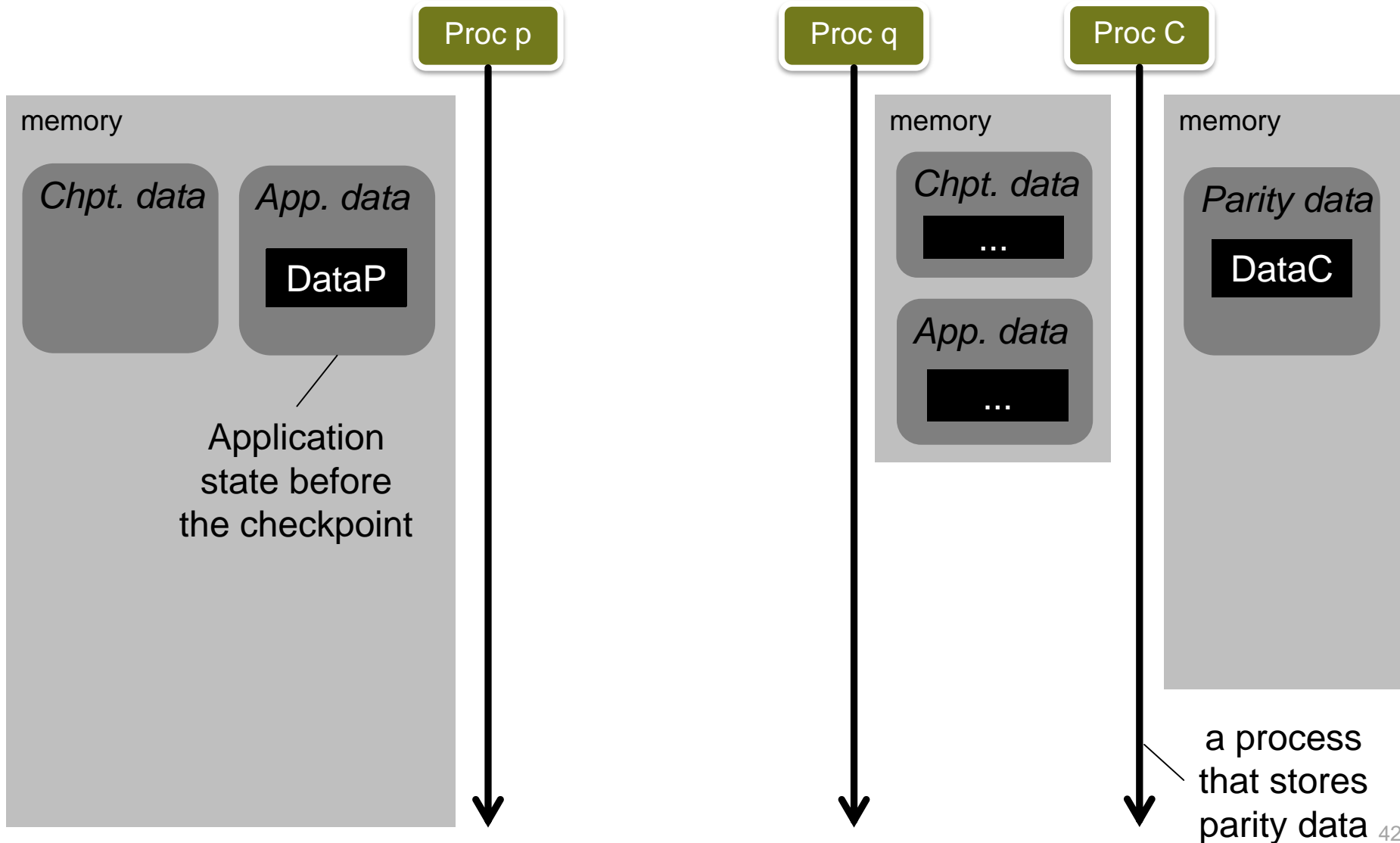
RMA: CHECKPOINTING



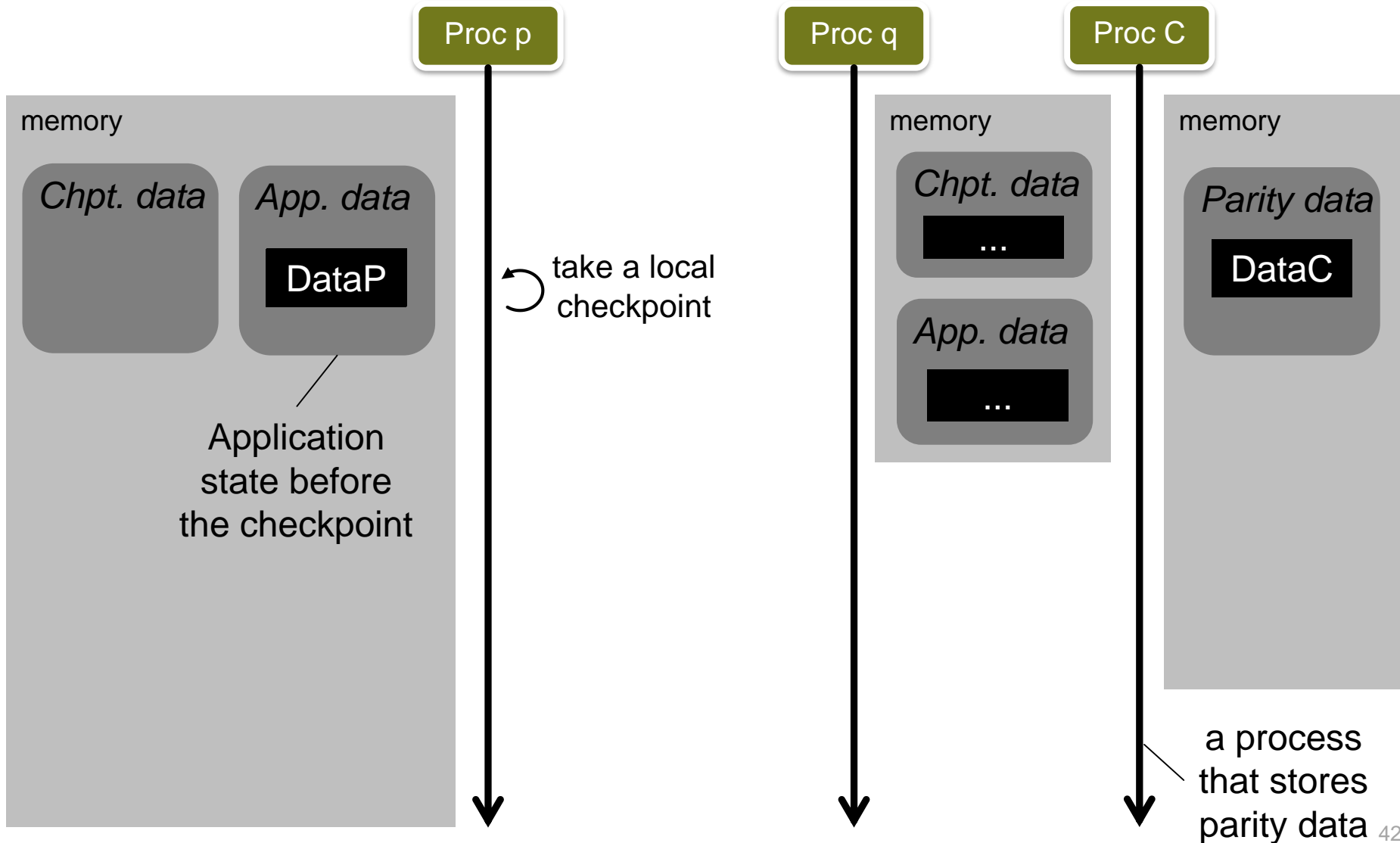
RMA: CHECKPOINTING



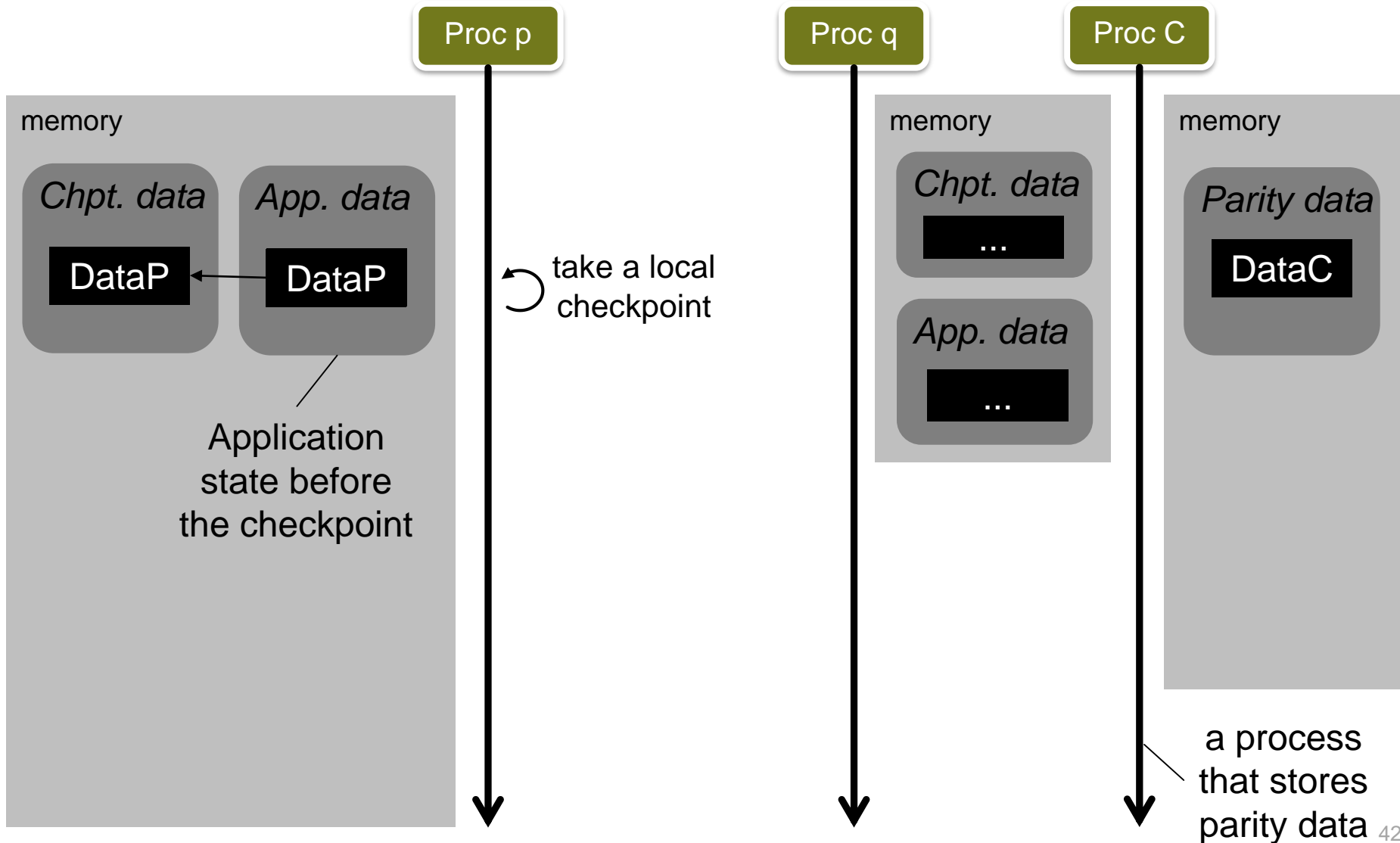
RMA: CHECKPOINTING



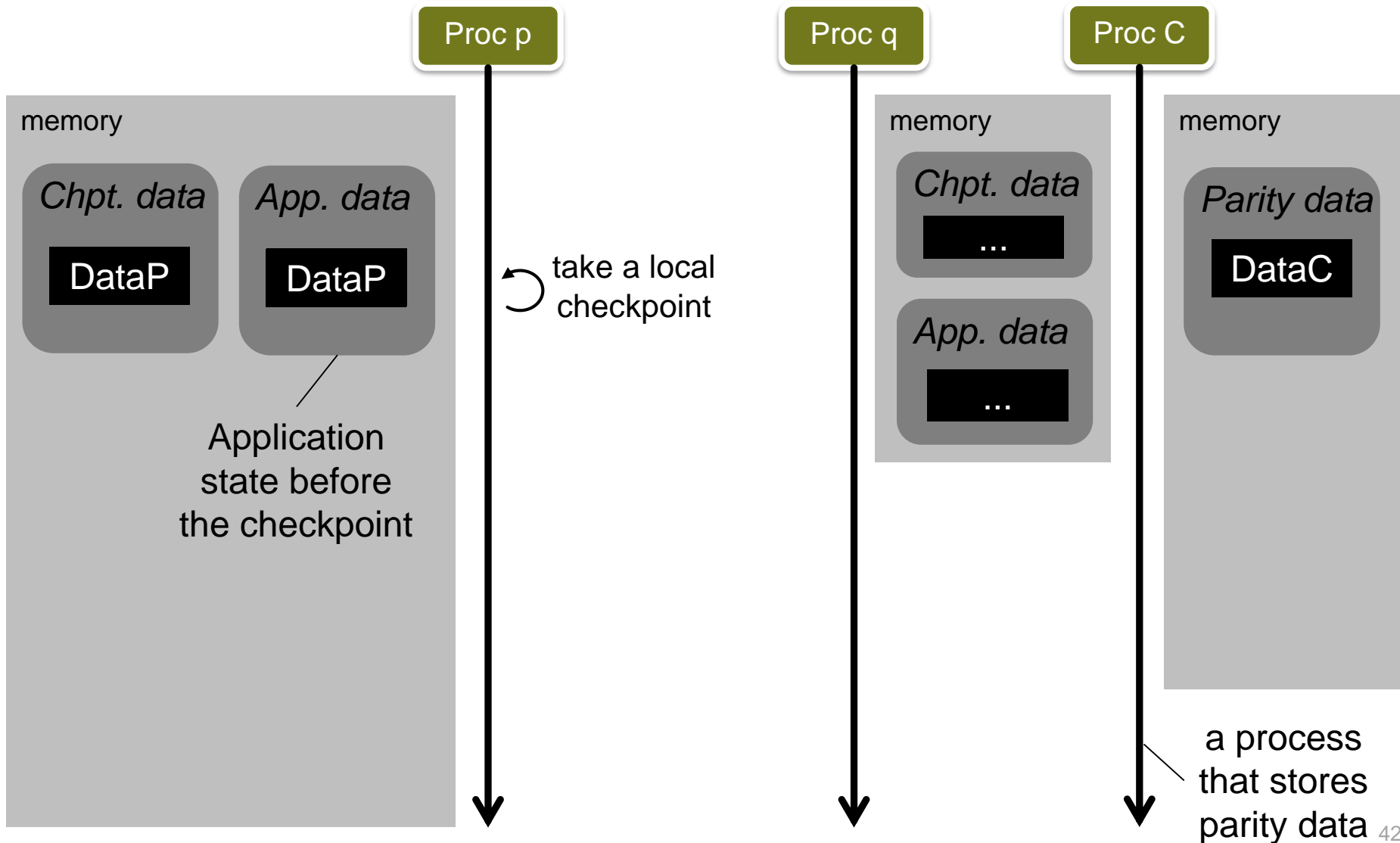
RMA: CHECKPOINTING



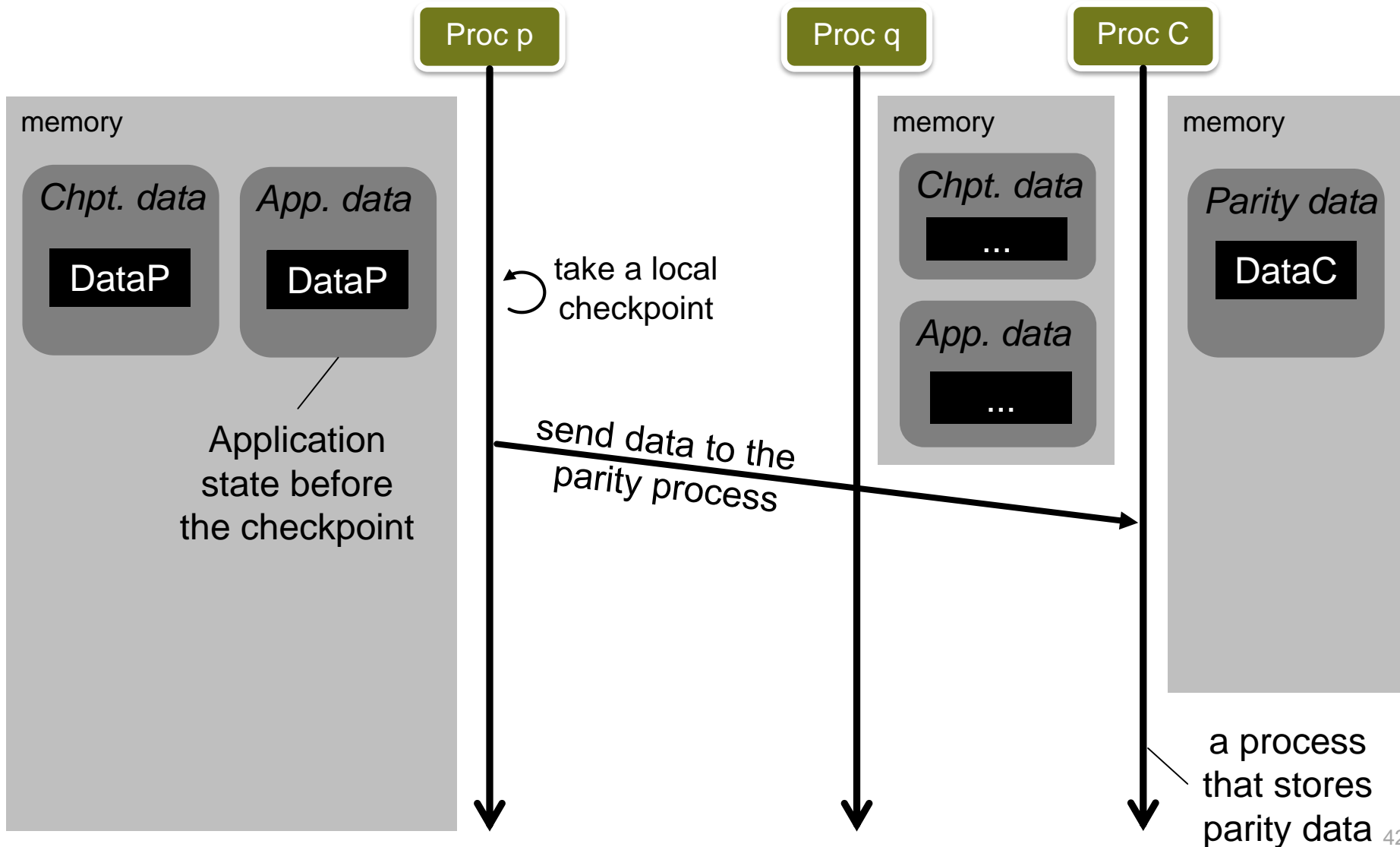
RMA: CHECKPOINTING



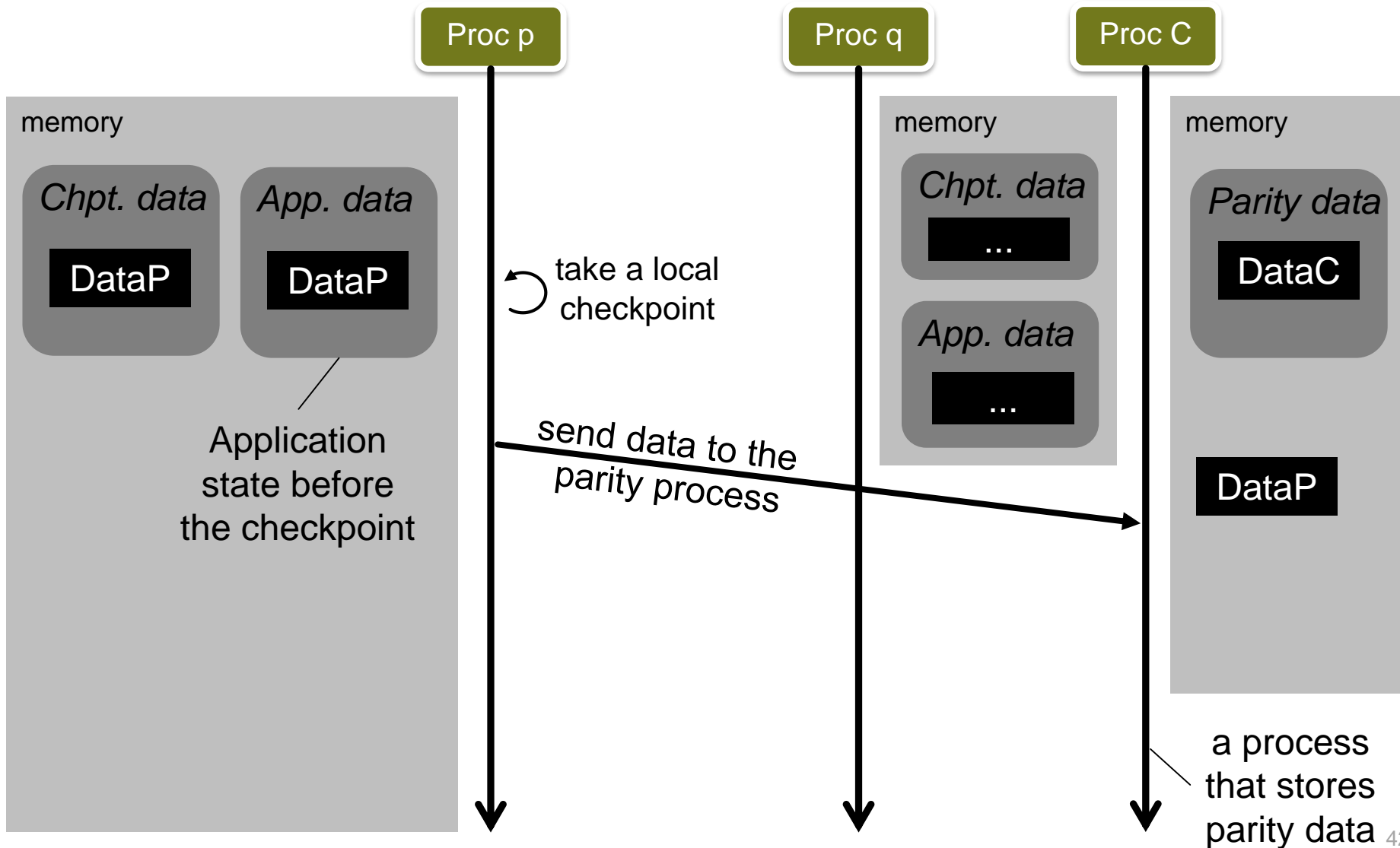
RMA: CHECKPOINTING



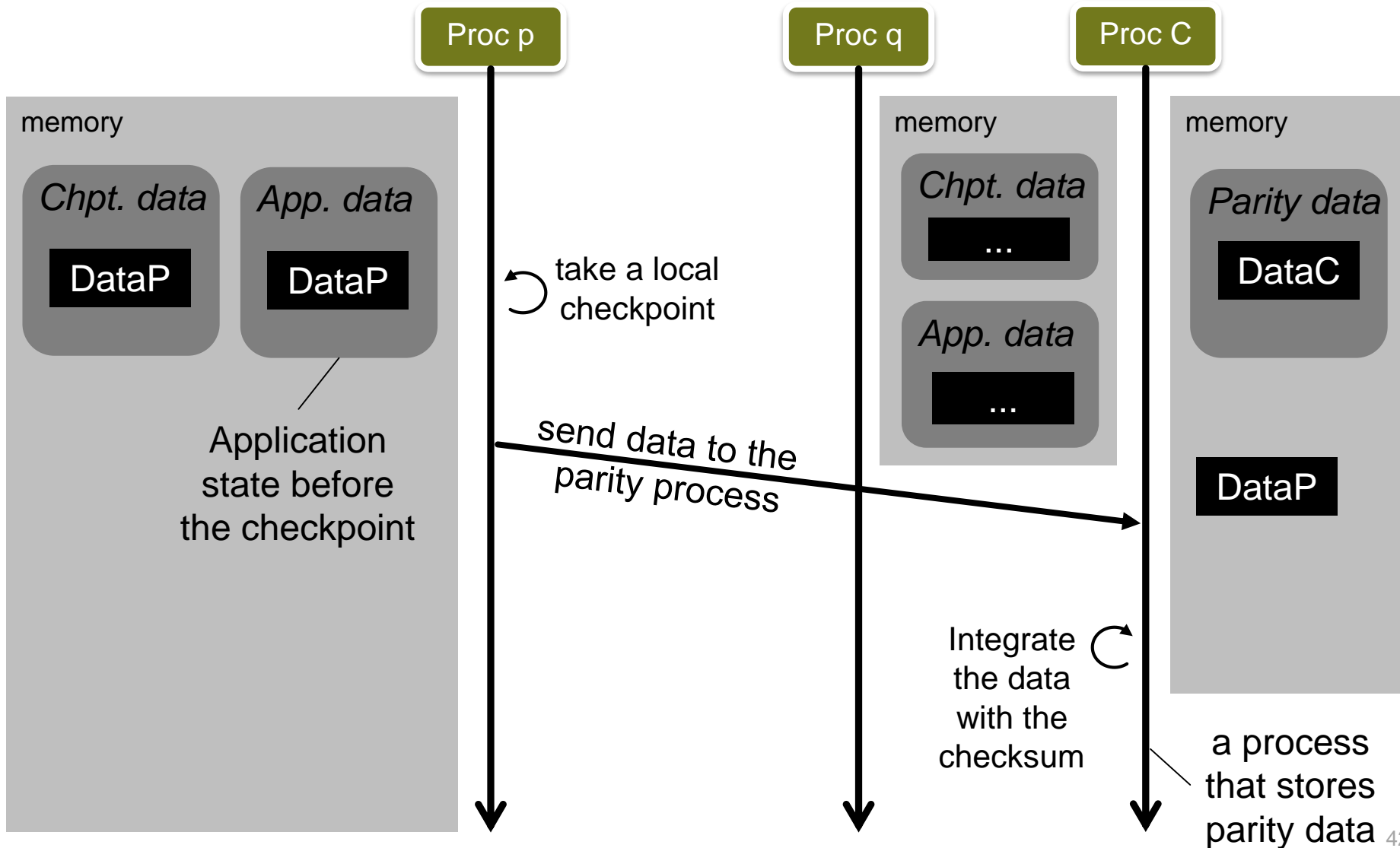
RMA: CHECKPOINTING



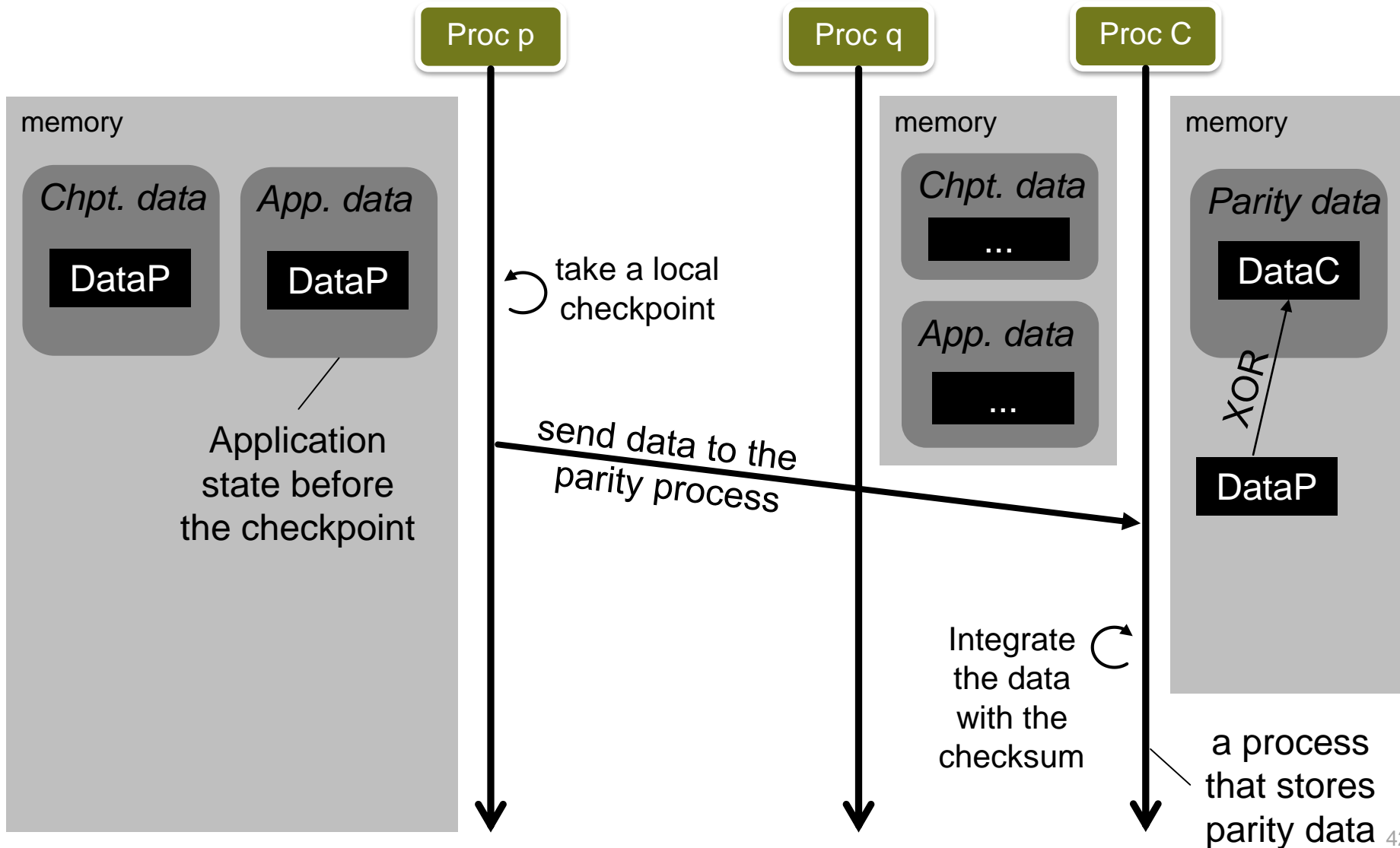
RMA: CHECKPOINTING



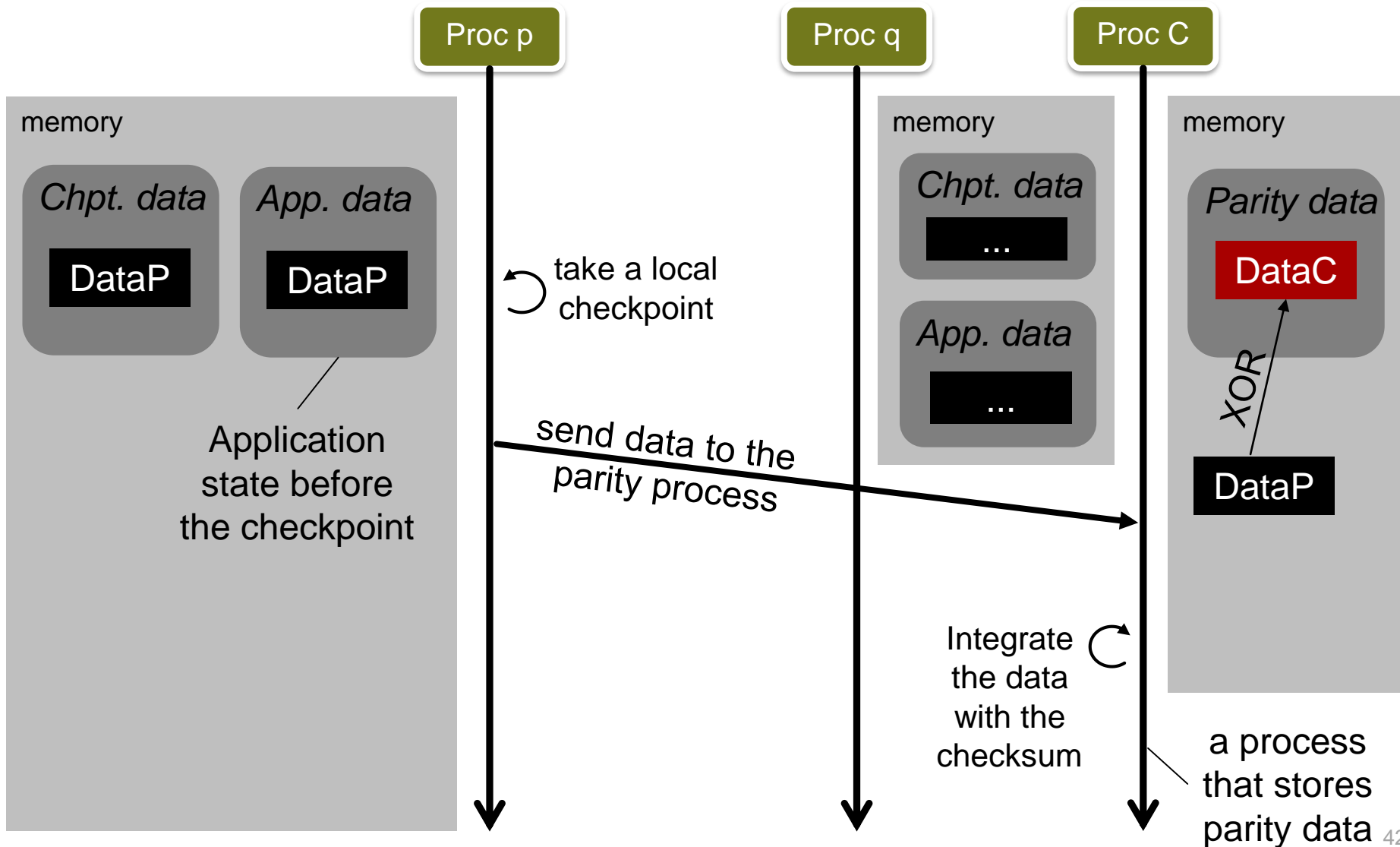
RMA: CHECKPOINTING



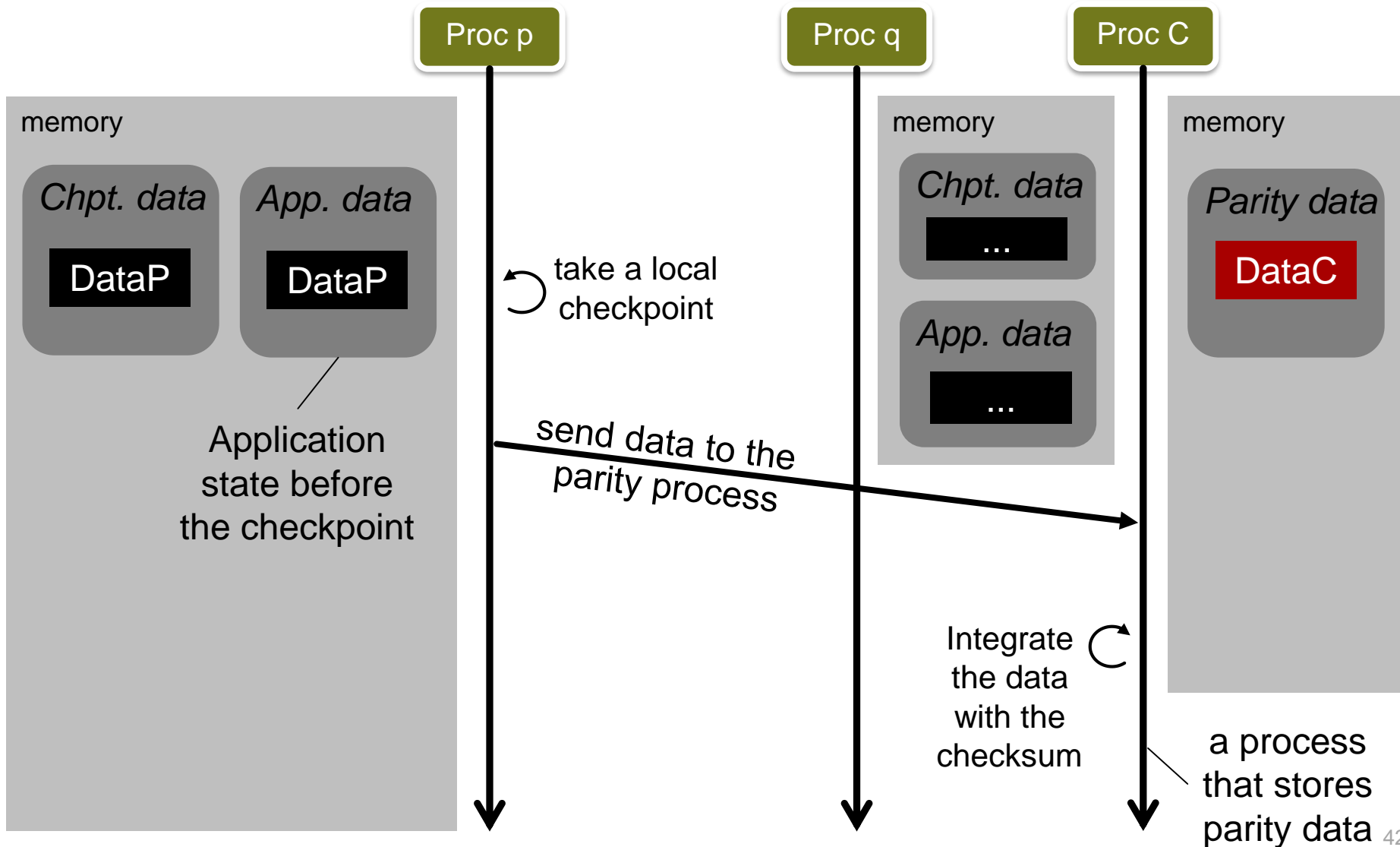
RMA: CHECKPOINTING



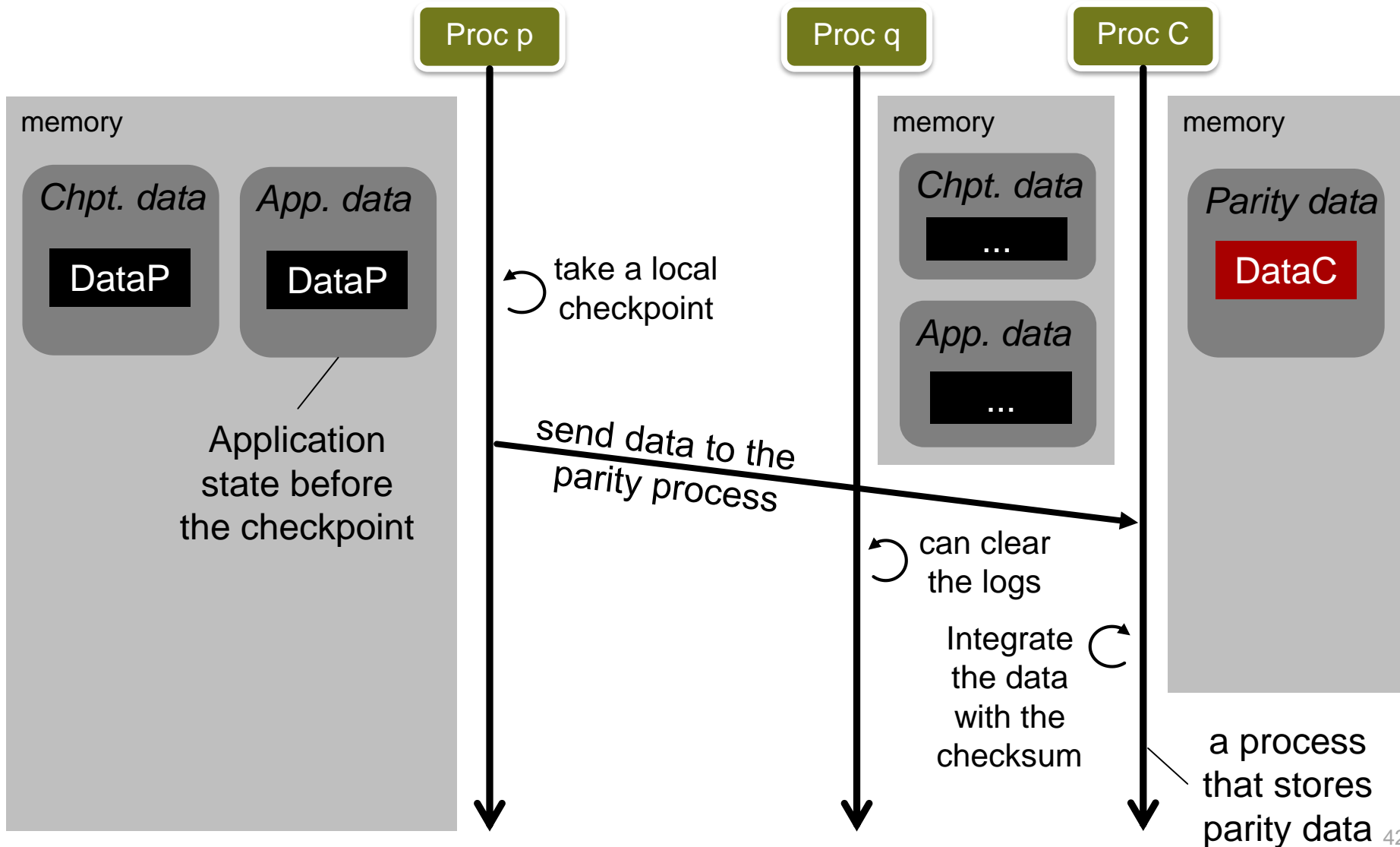
RMA: CHECKPOINTING



RMA: CHECKPOINTING

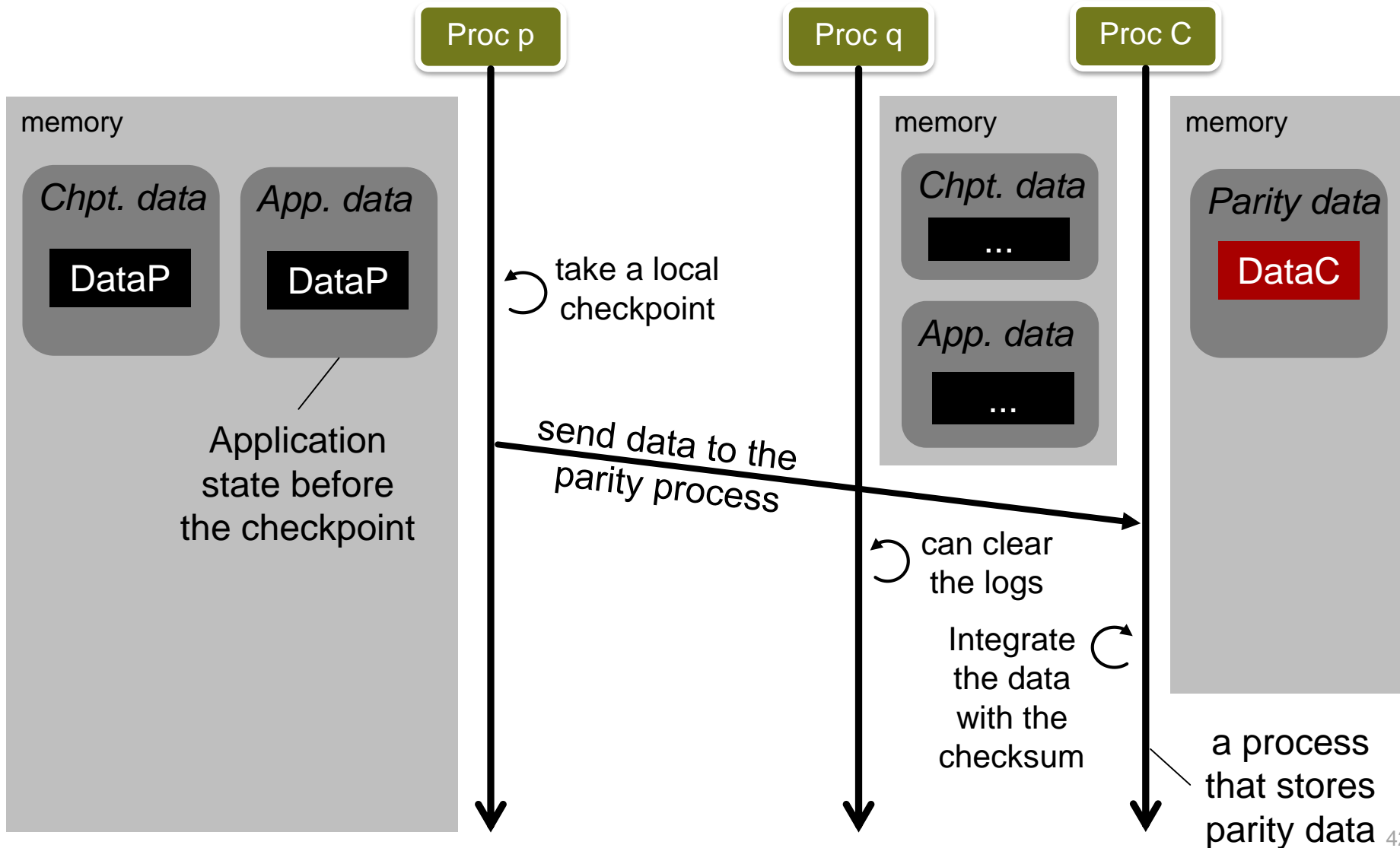


RMA: CHECKPOINTING



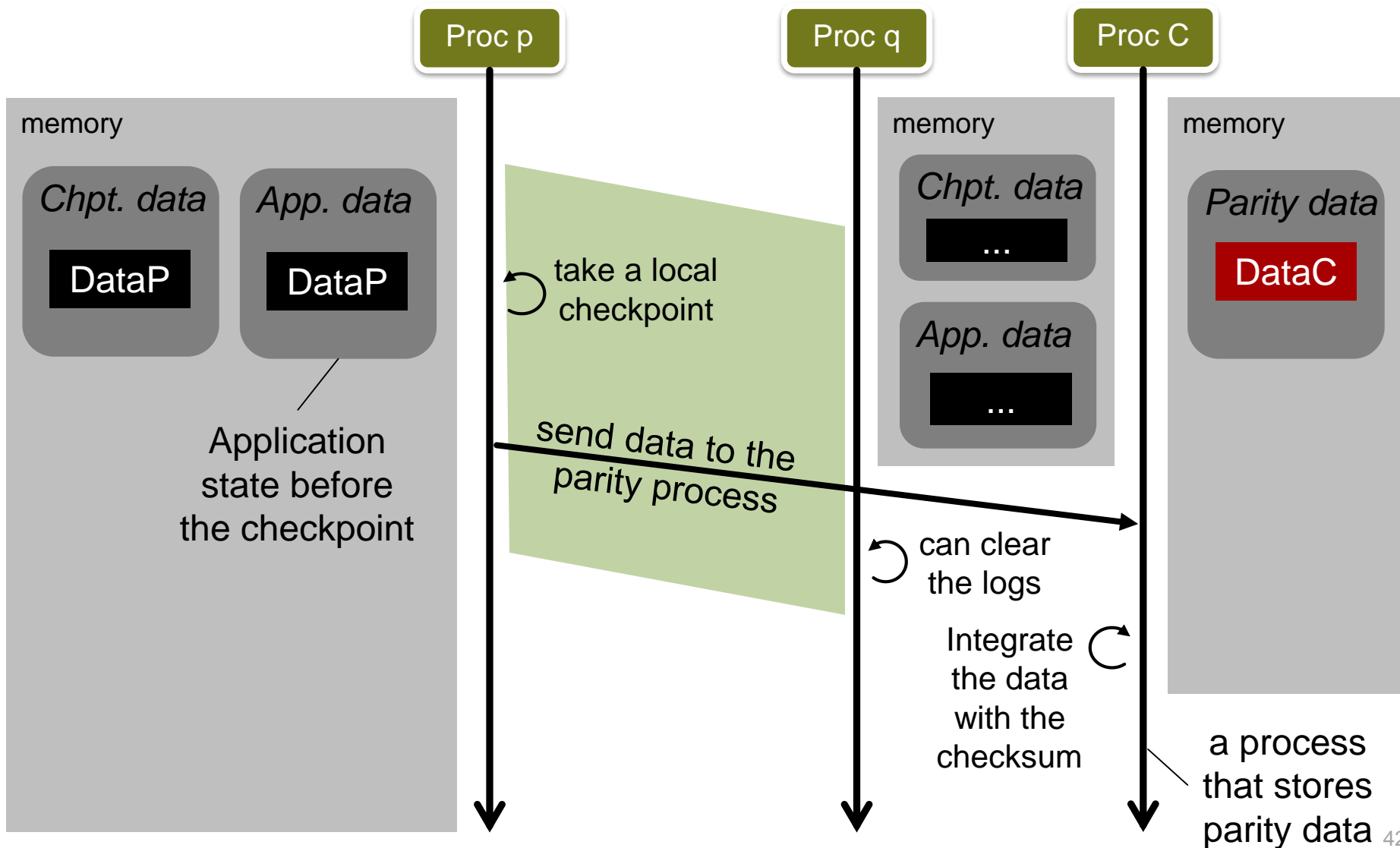
RMA: CHECKPOINTING

The Epoch Condition



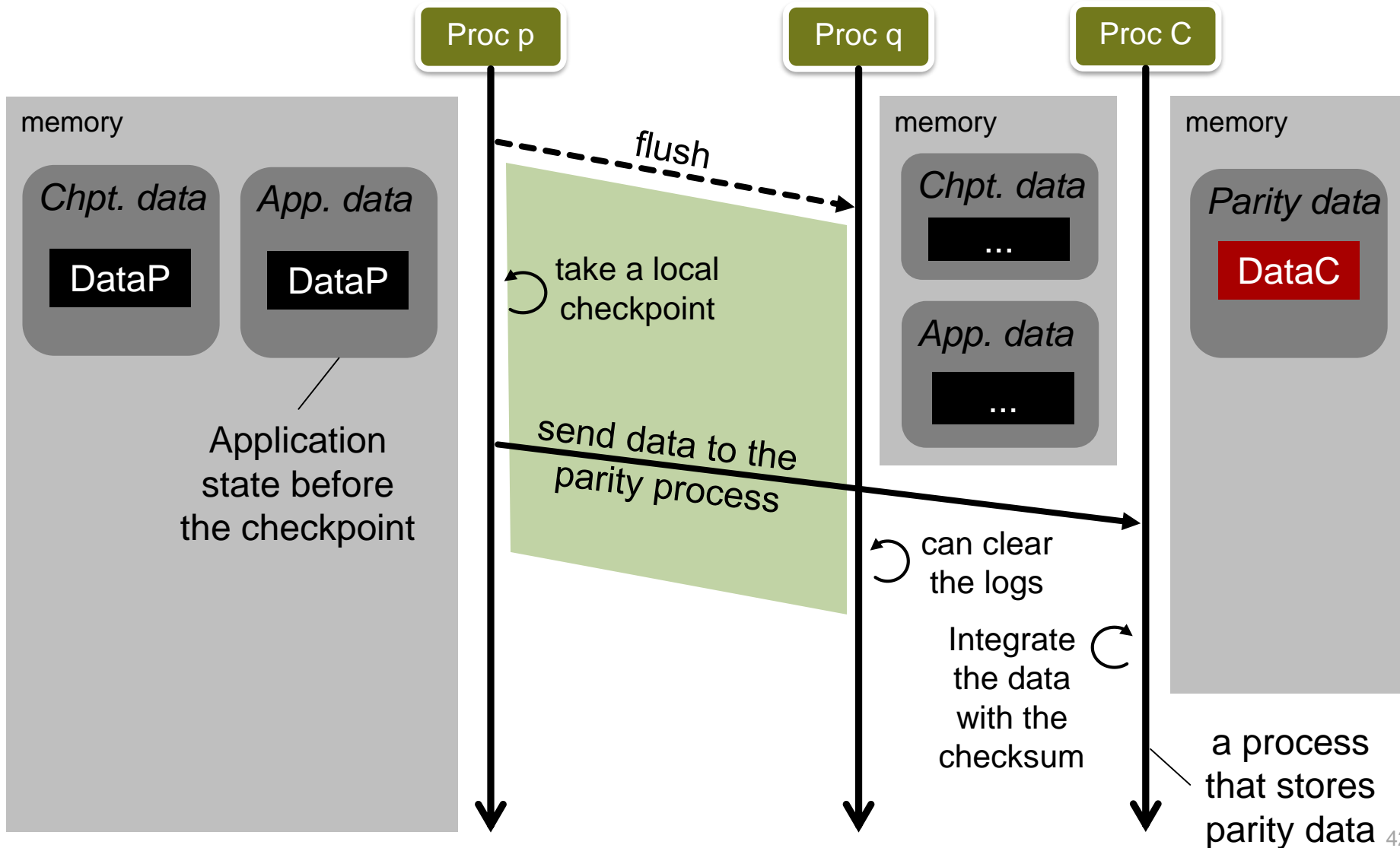
RMA: CHECKPOINTING

The Epoch Condition



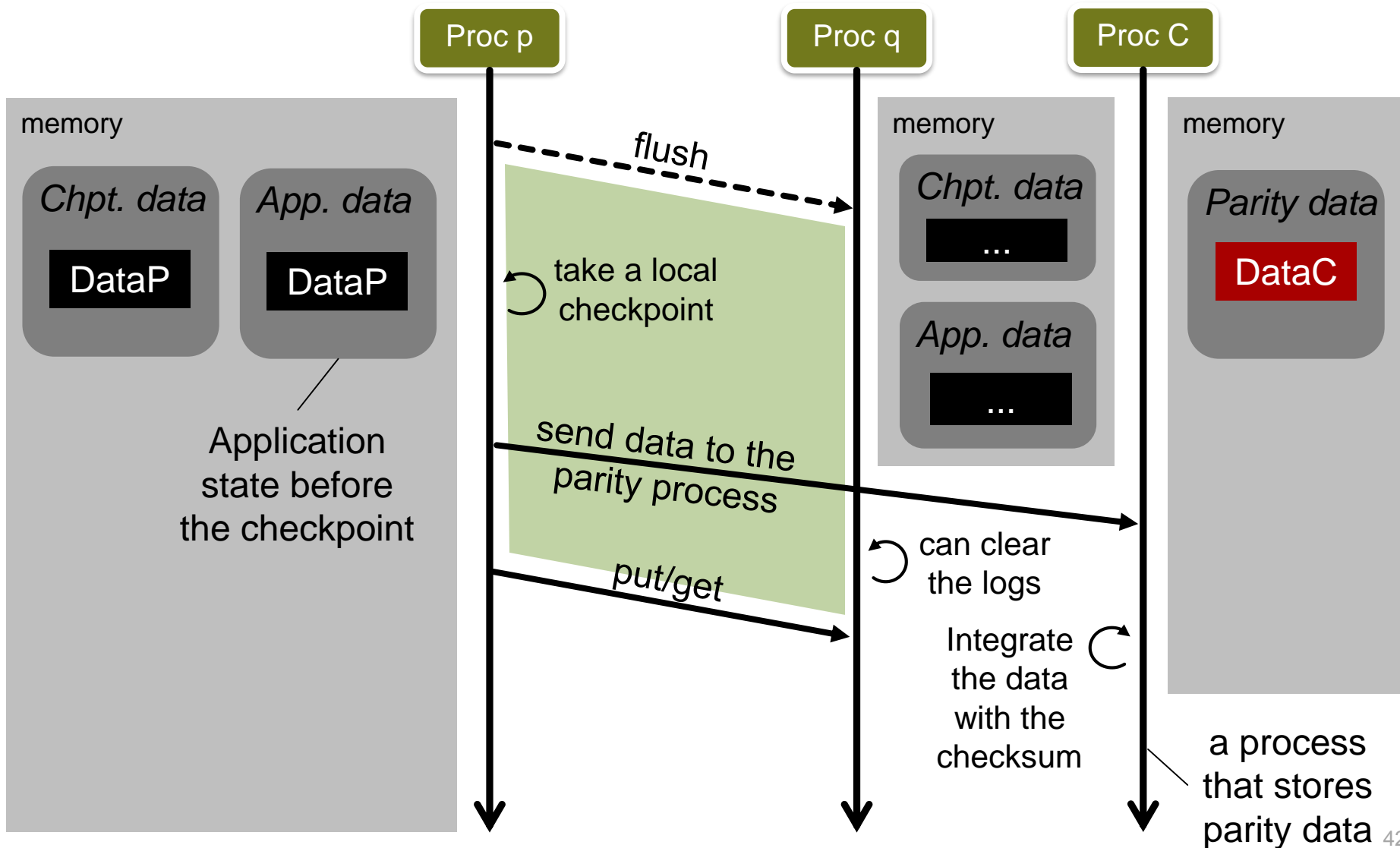
RMA: CHECKPOINTING

The Epoch Condition



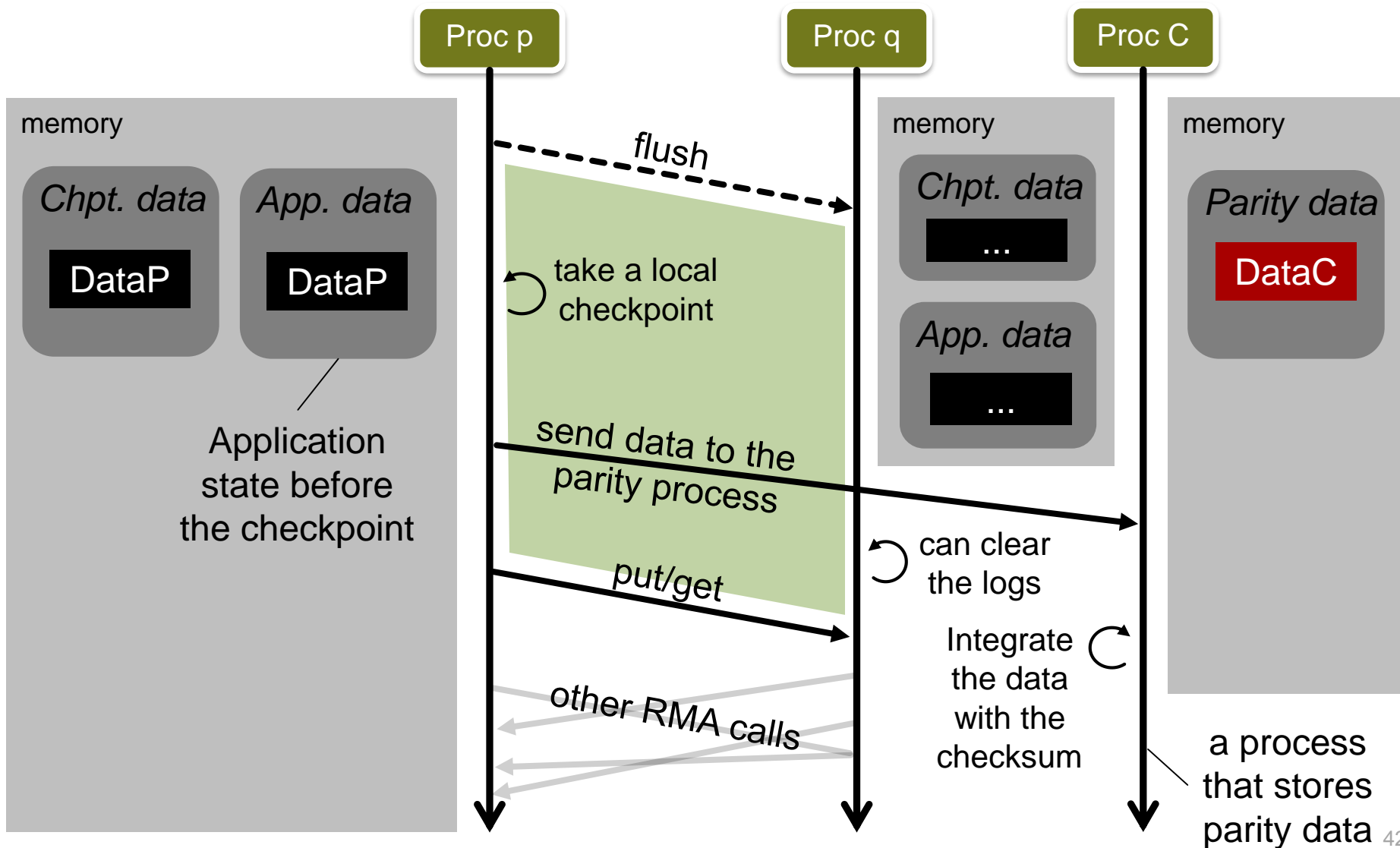
RMA: CHECKPOINTING

The Epoch Condition



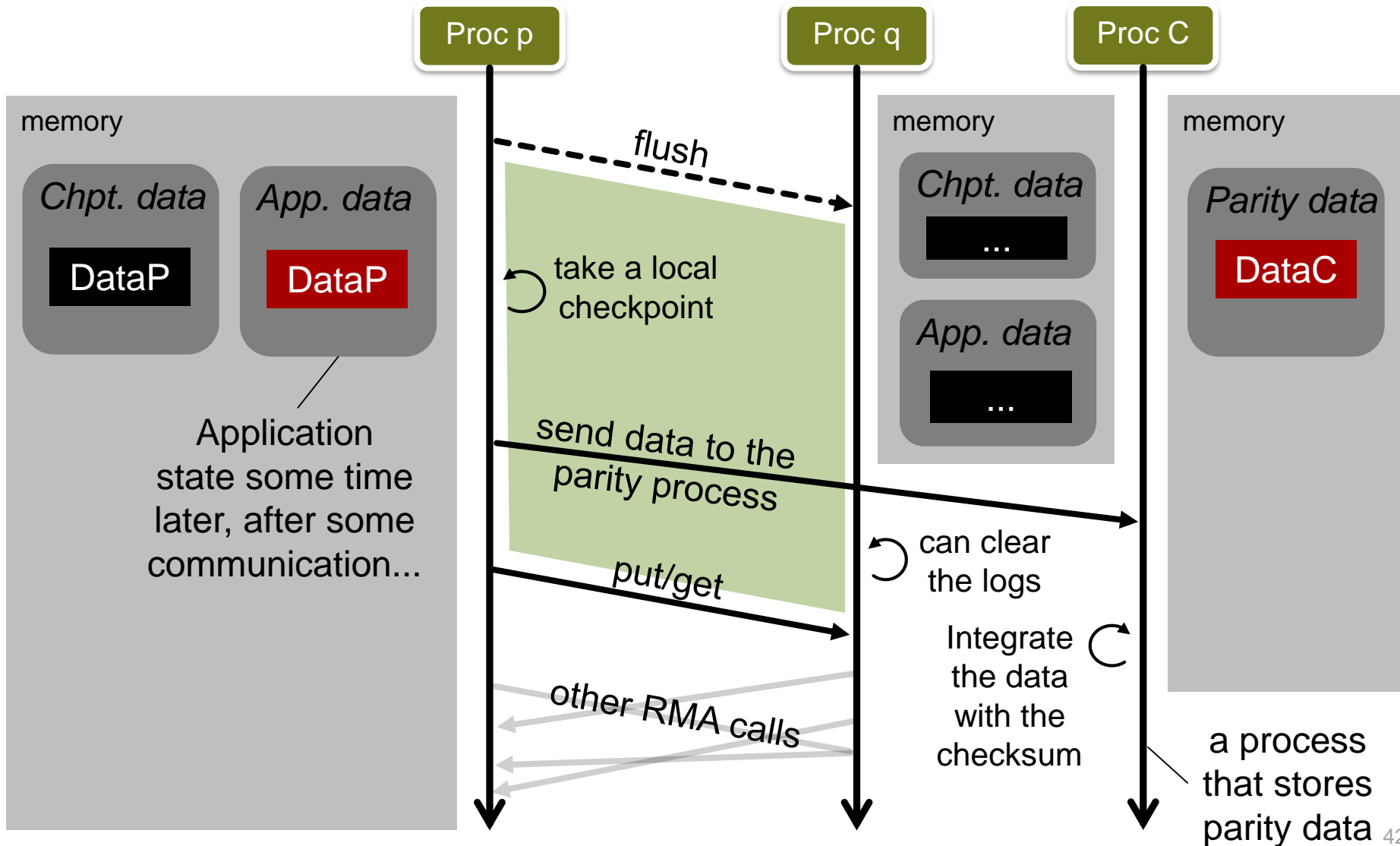
RMA: CHECKPOINTING

The Epoch Condition



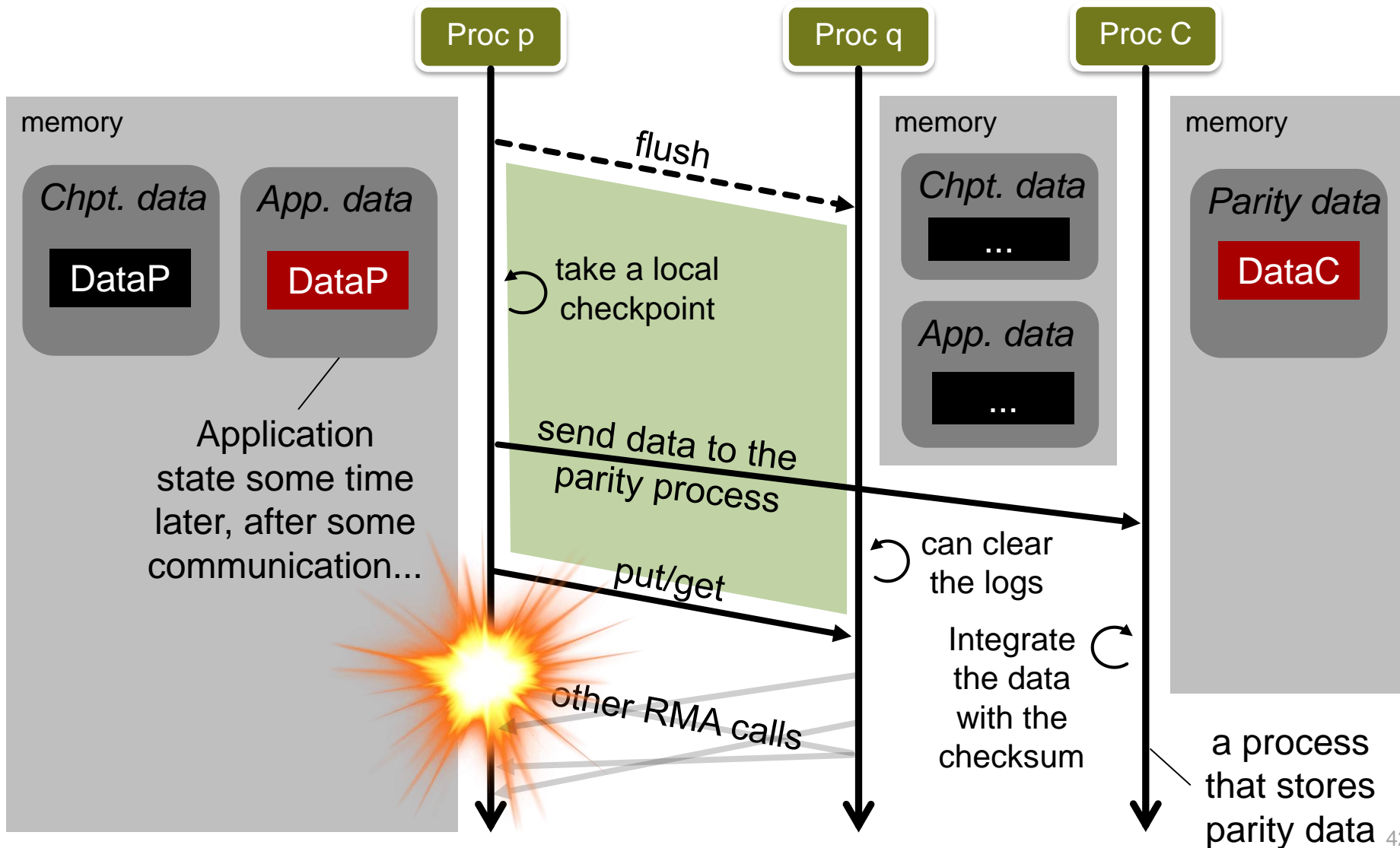
RMA: CHECKPOINTING

The Epoch Condition



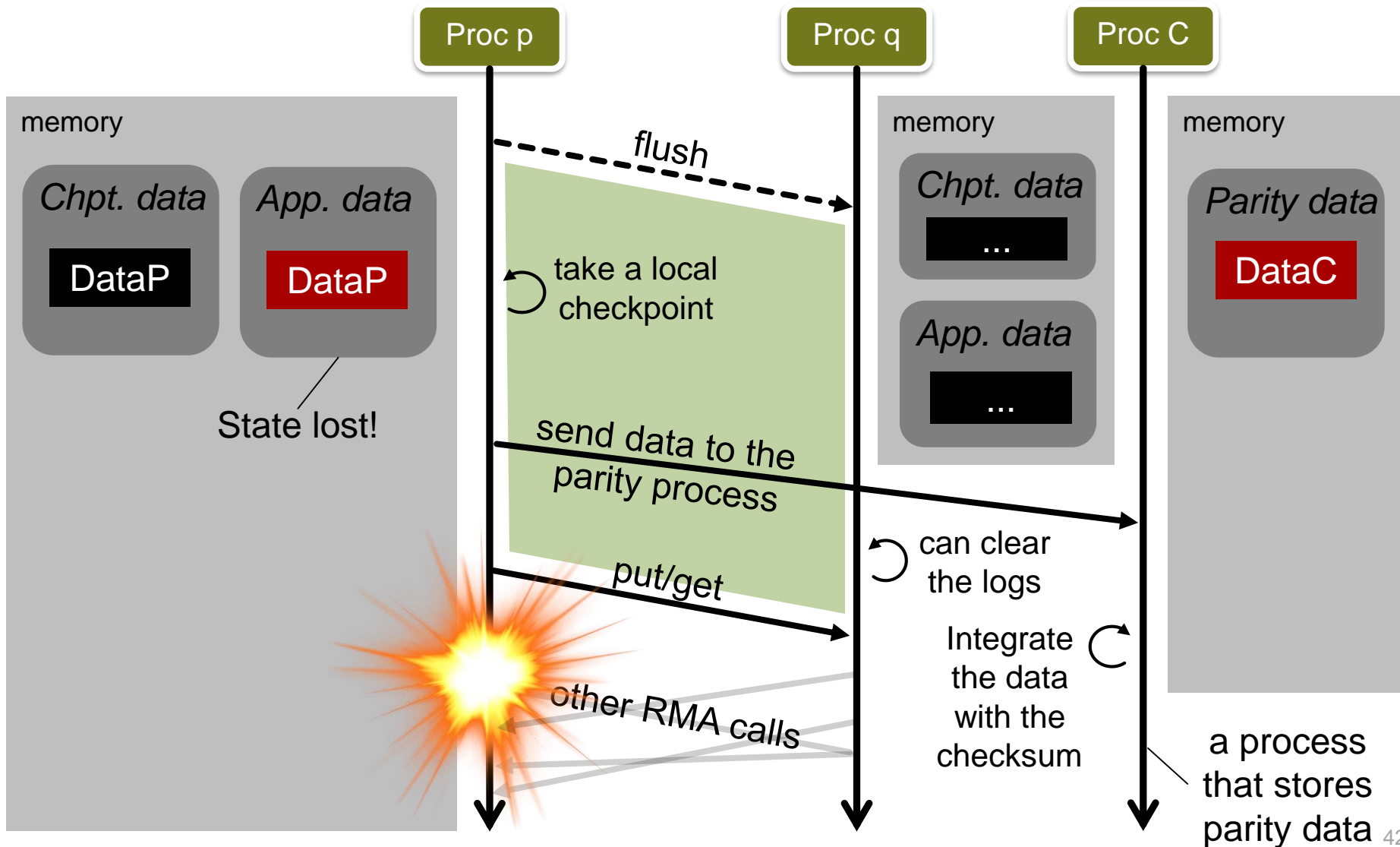
RMA: CHECKPOINTING

The Epoch Condition



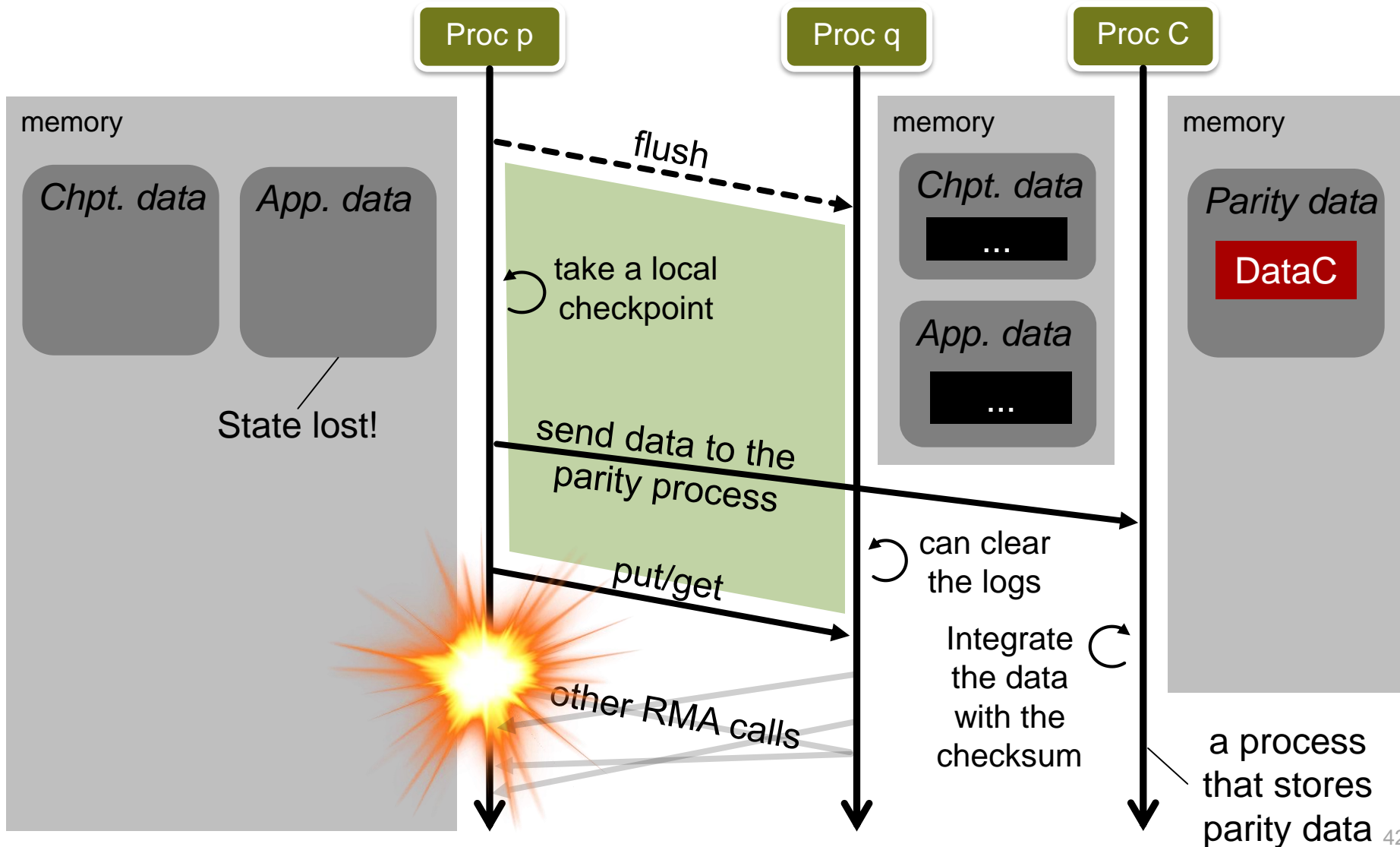
RMA: CHECKPOINTING

The Epoch Condition



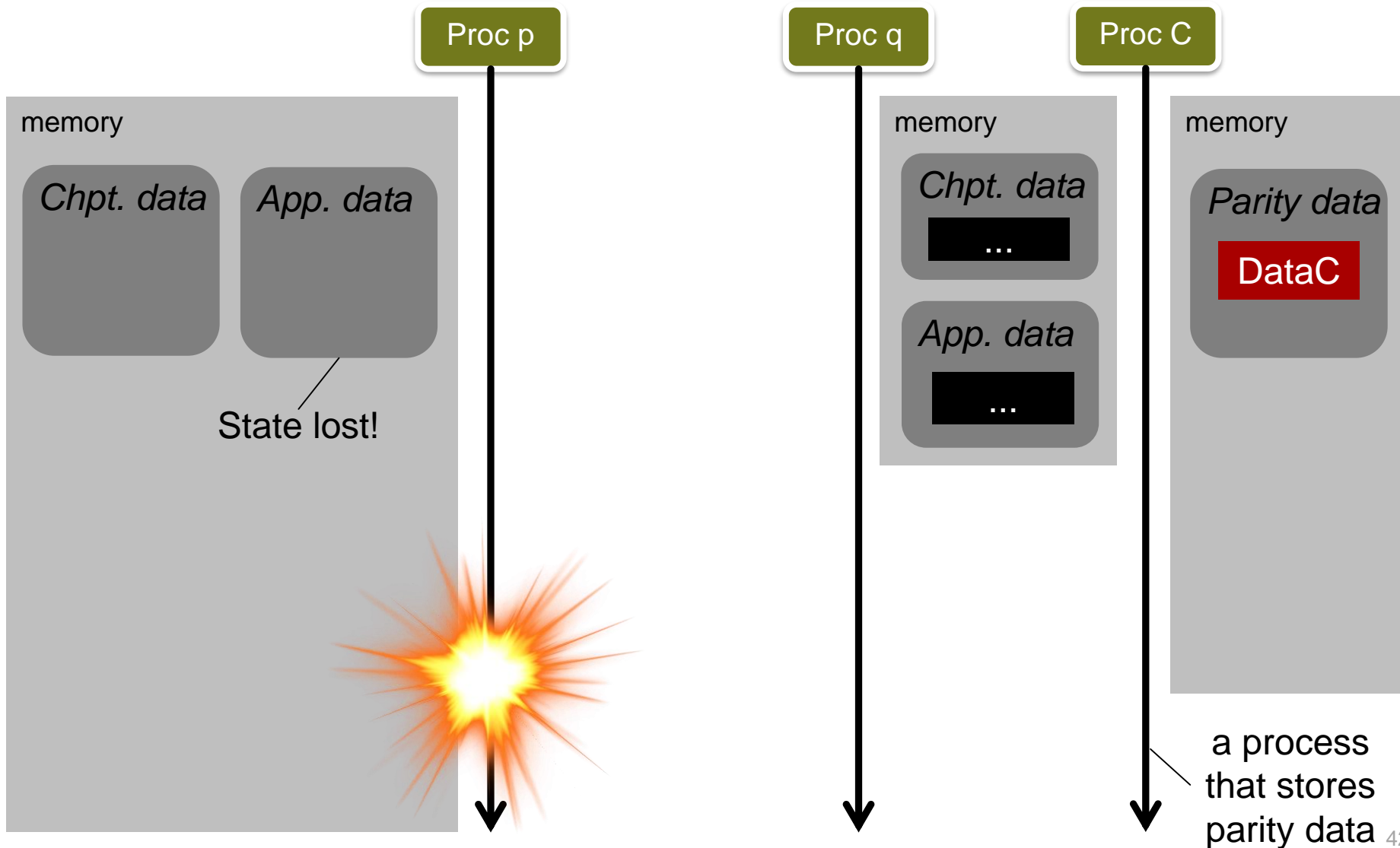
RMA: CHECKPOINTING

The Epoch Condition

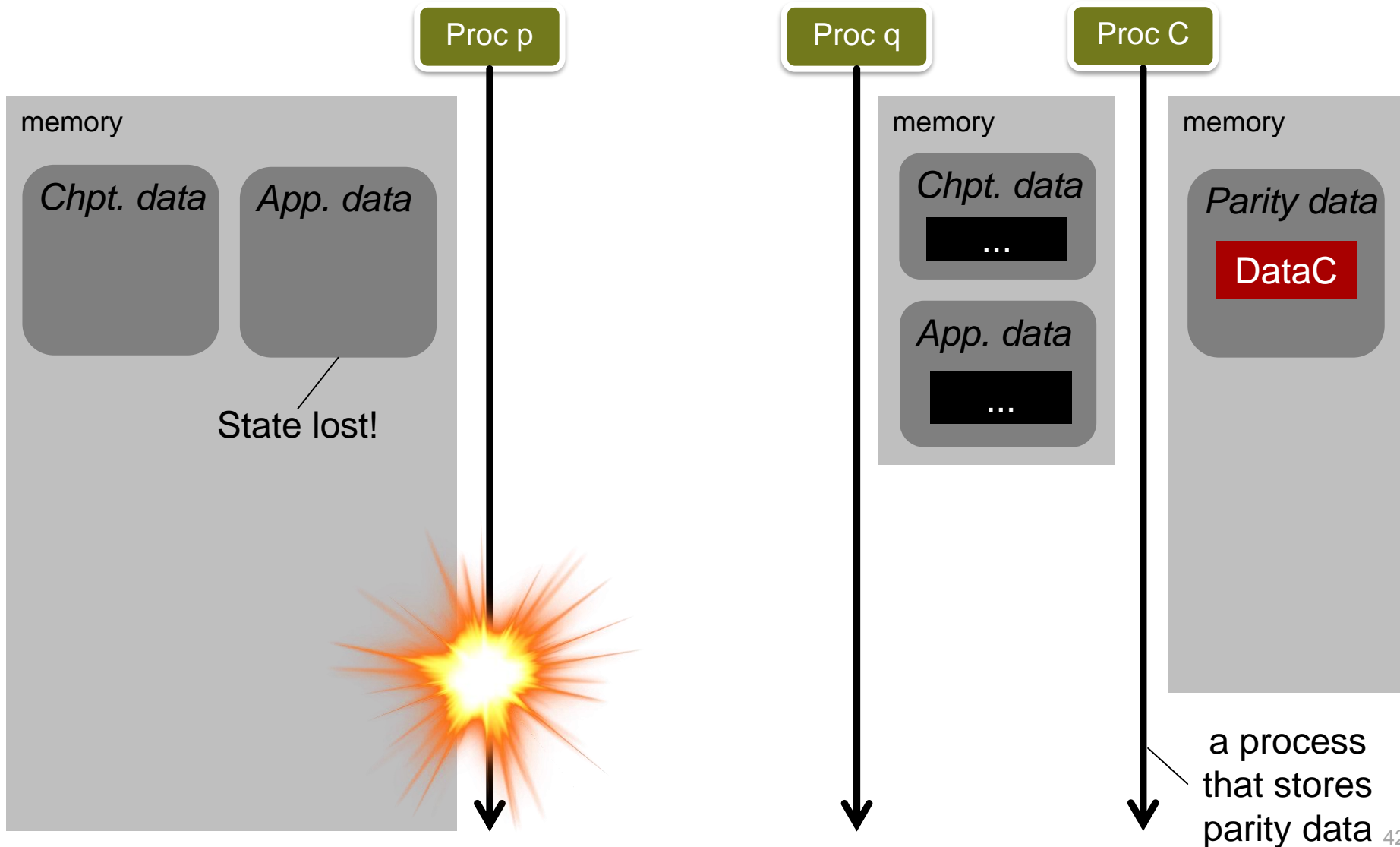


RMA: CHECKPOINTING

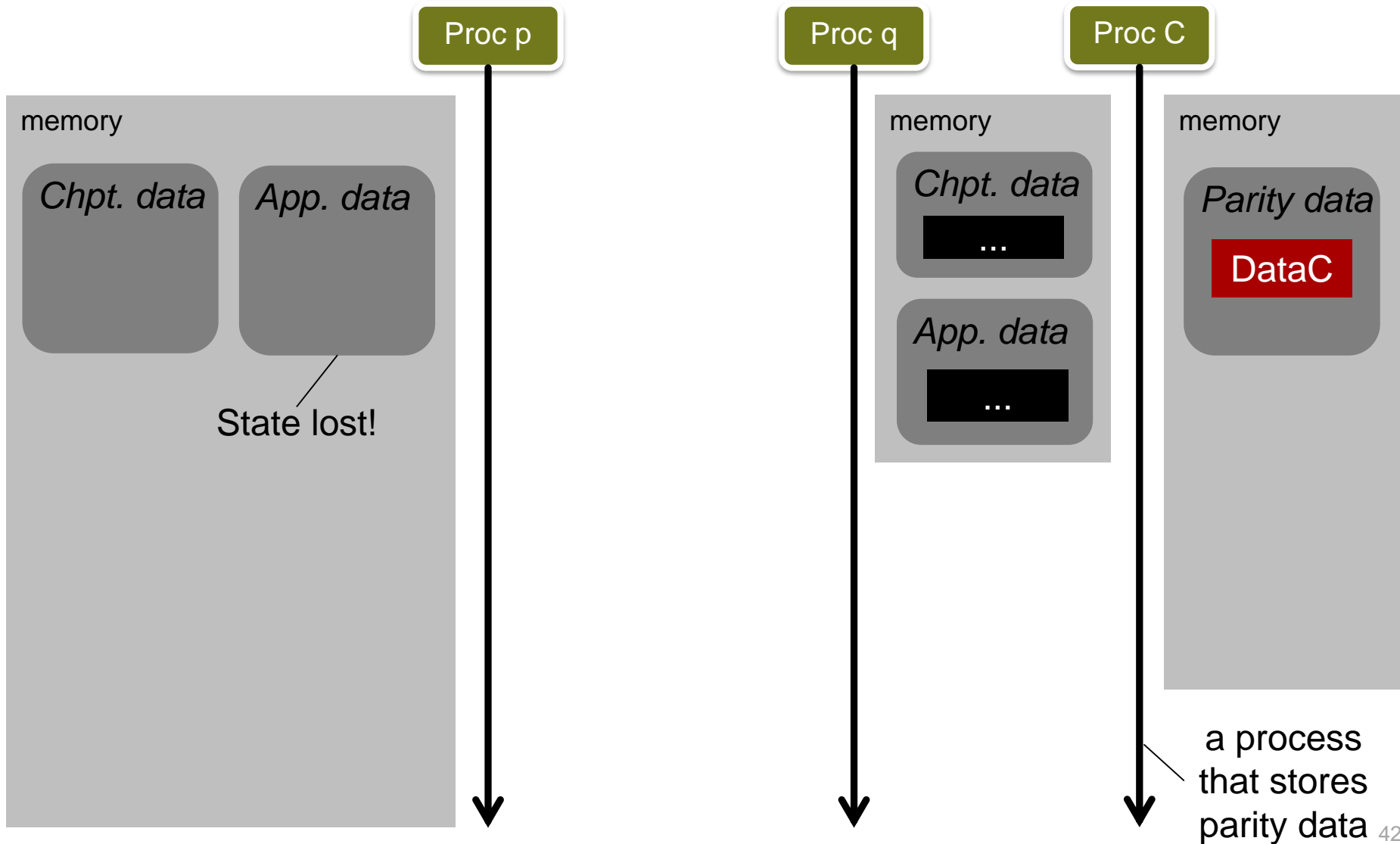
The Epoch Condition



RMA: RECOVERY

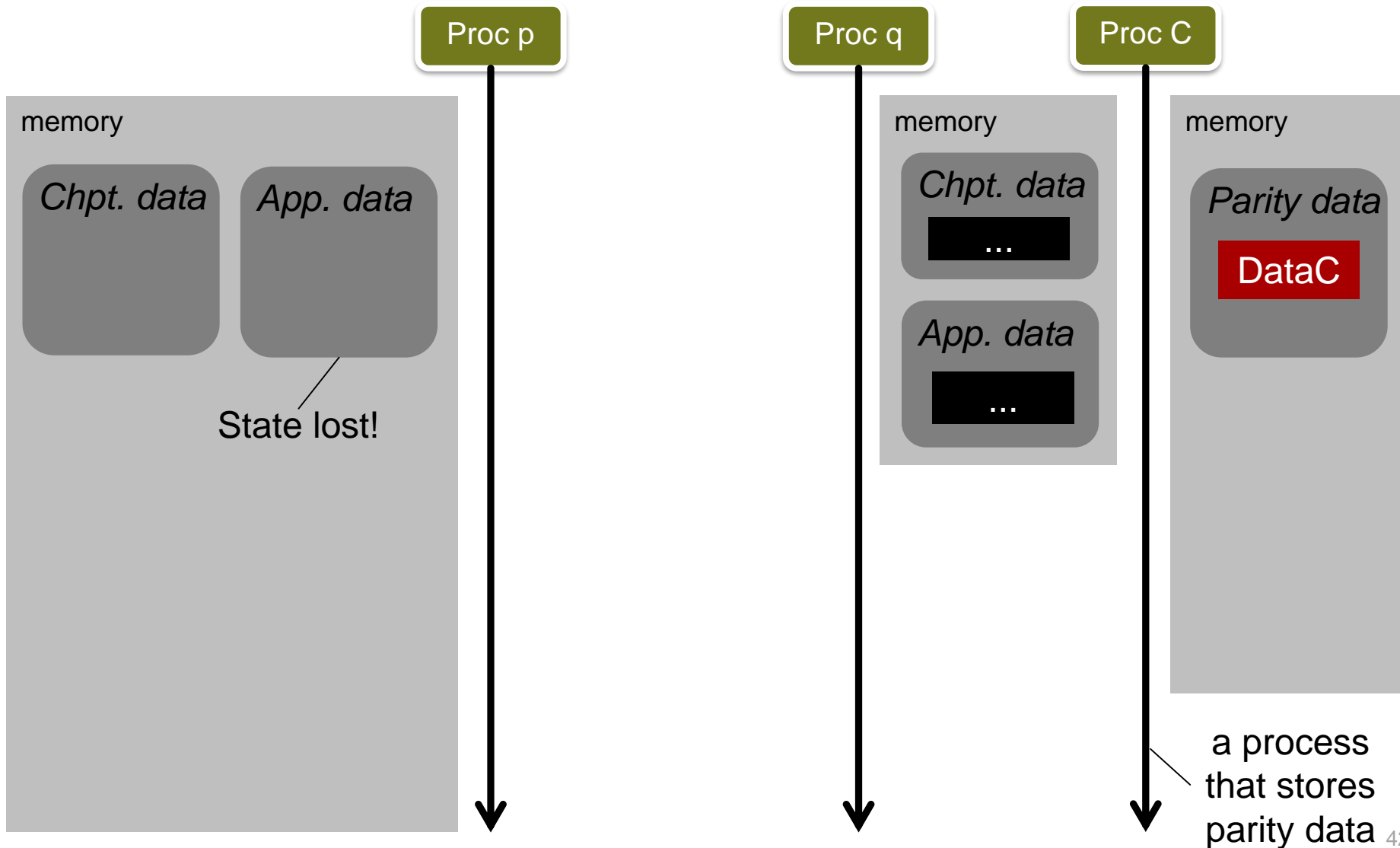


RMA: RECOVERY



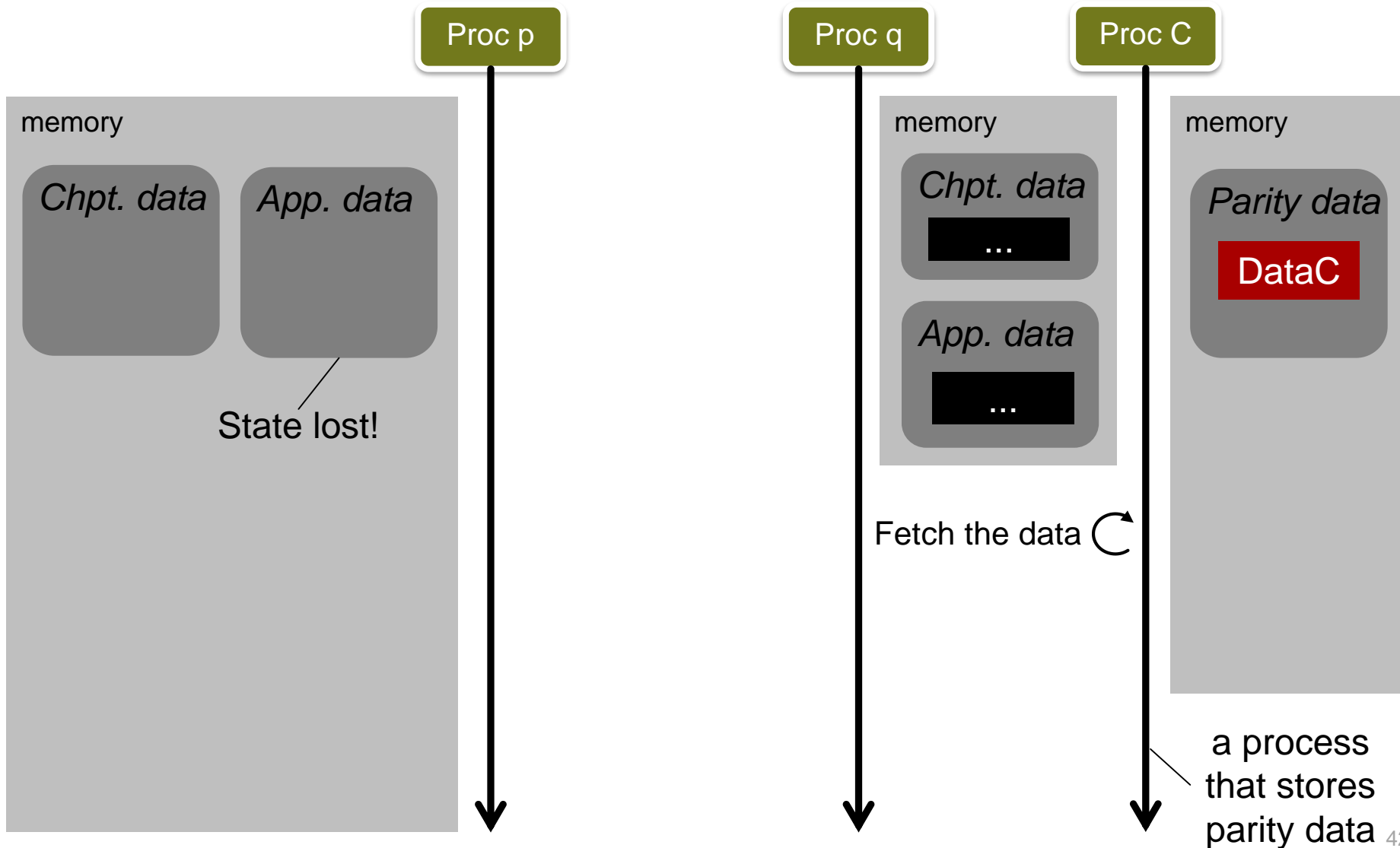
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



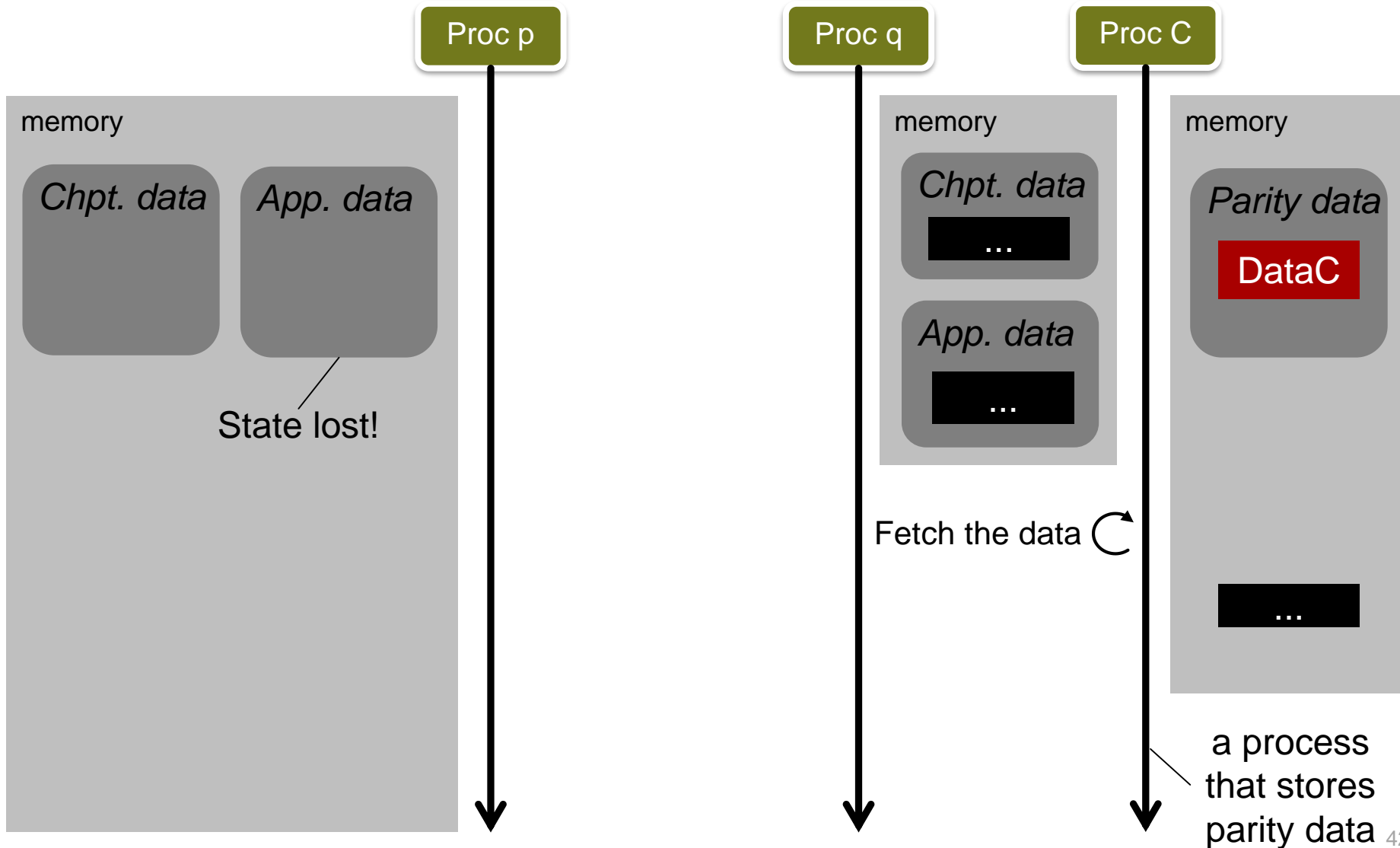
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



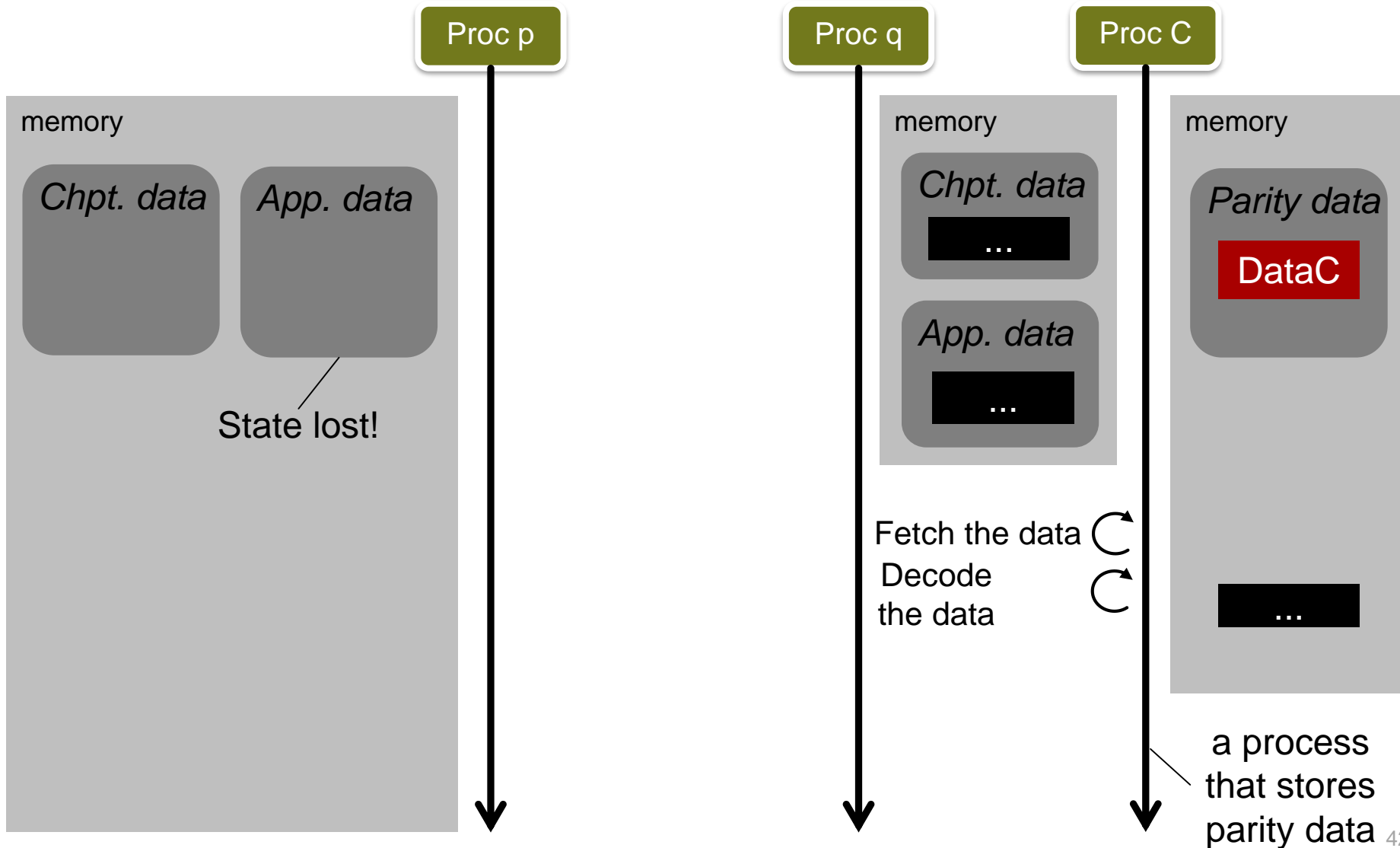
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



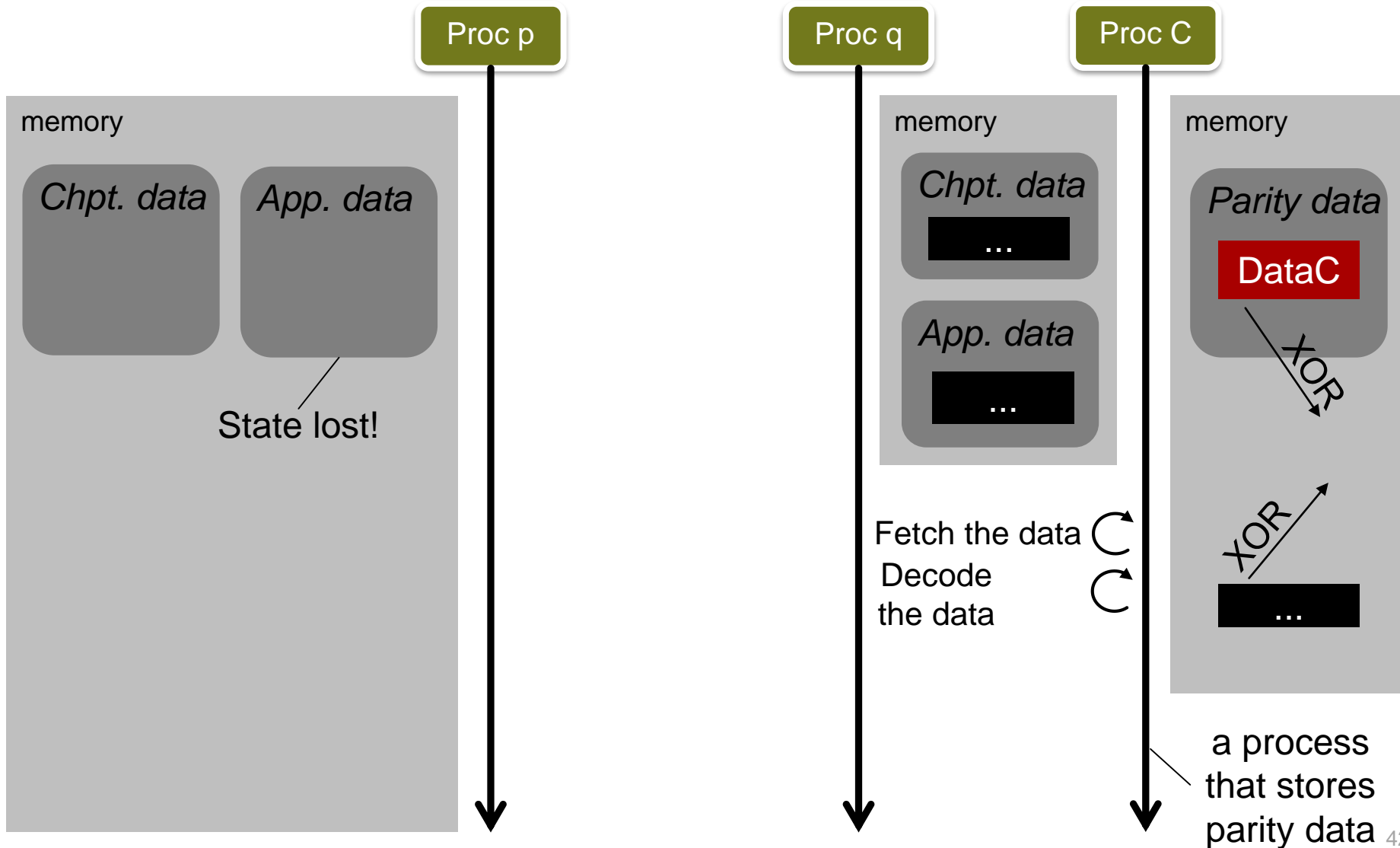
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



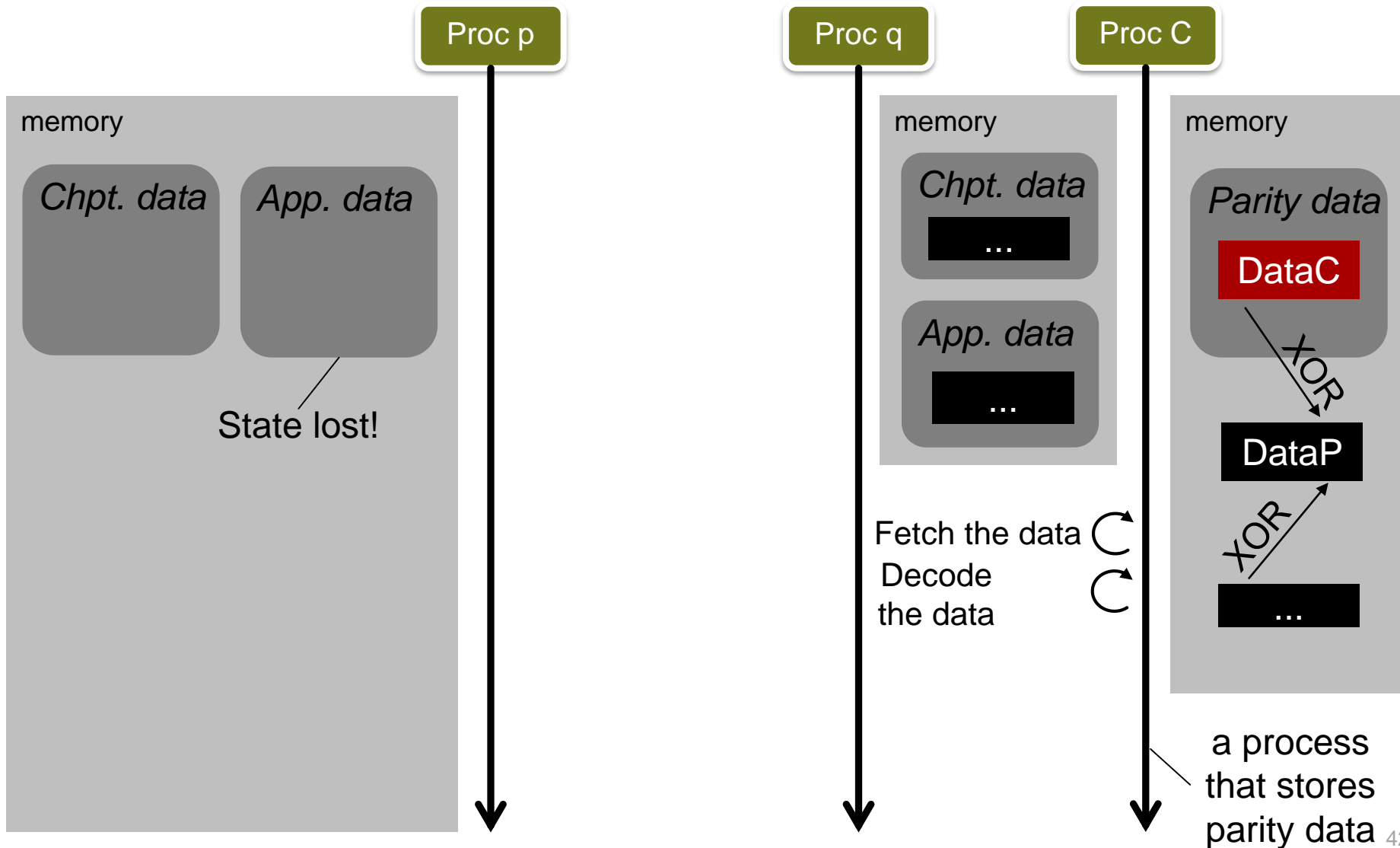
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



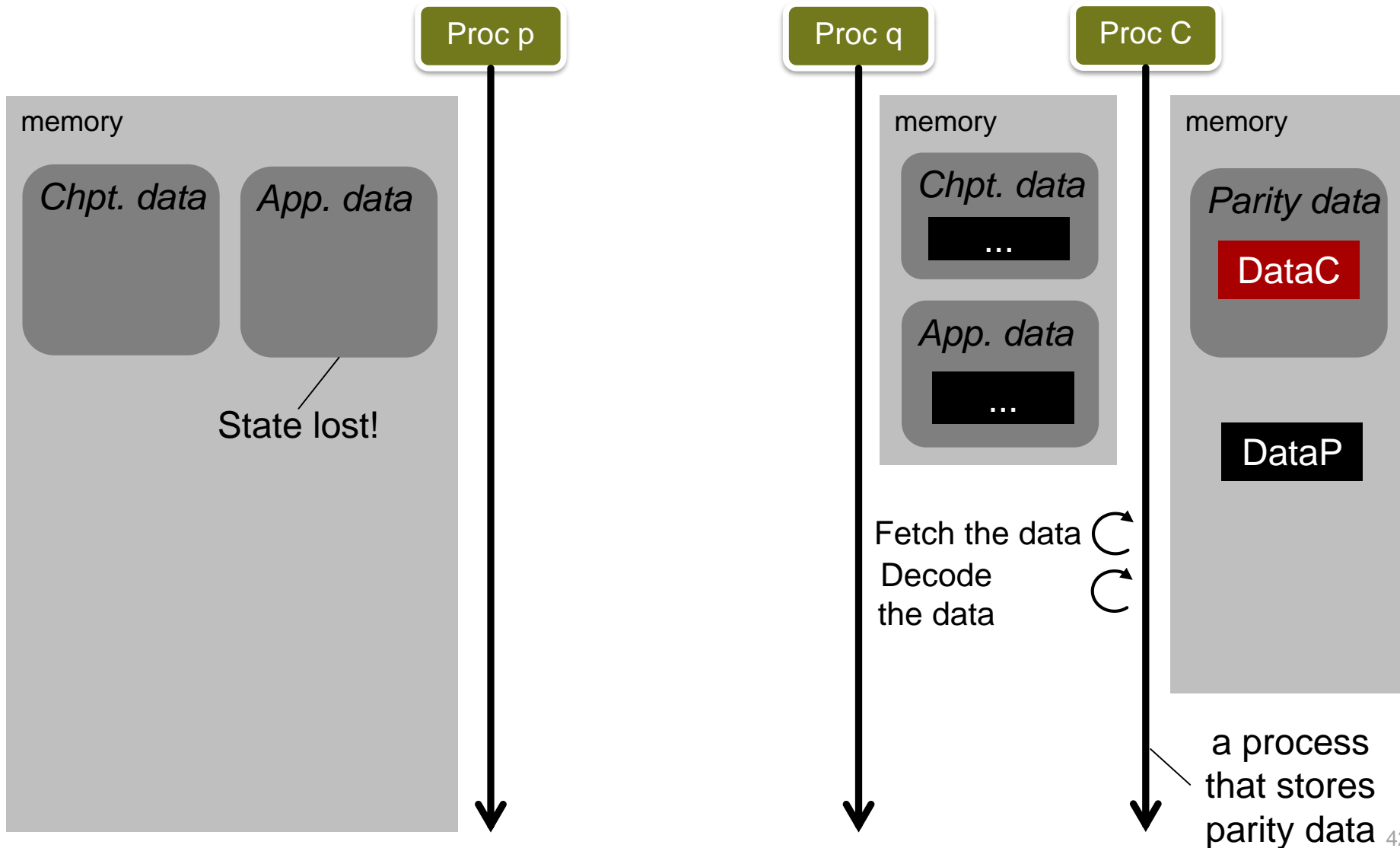
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



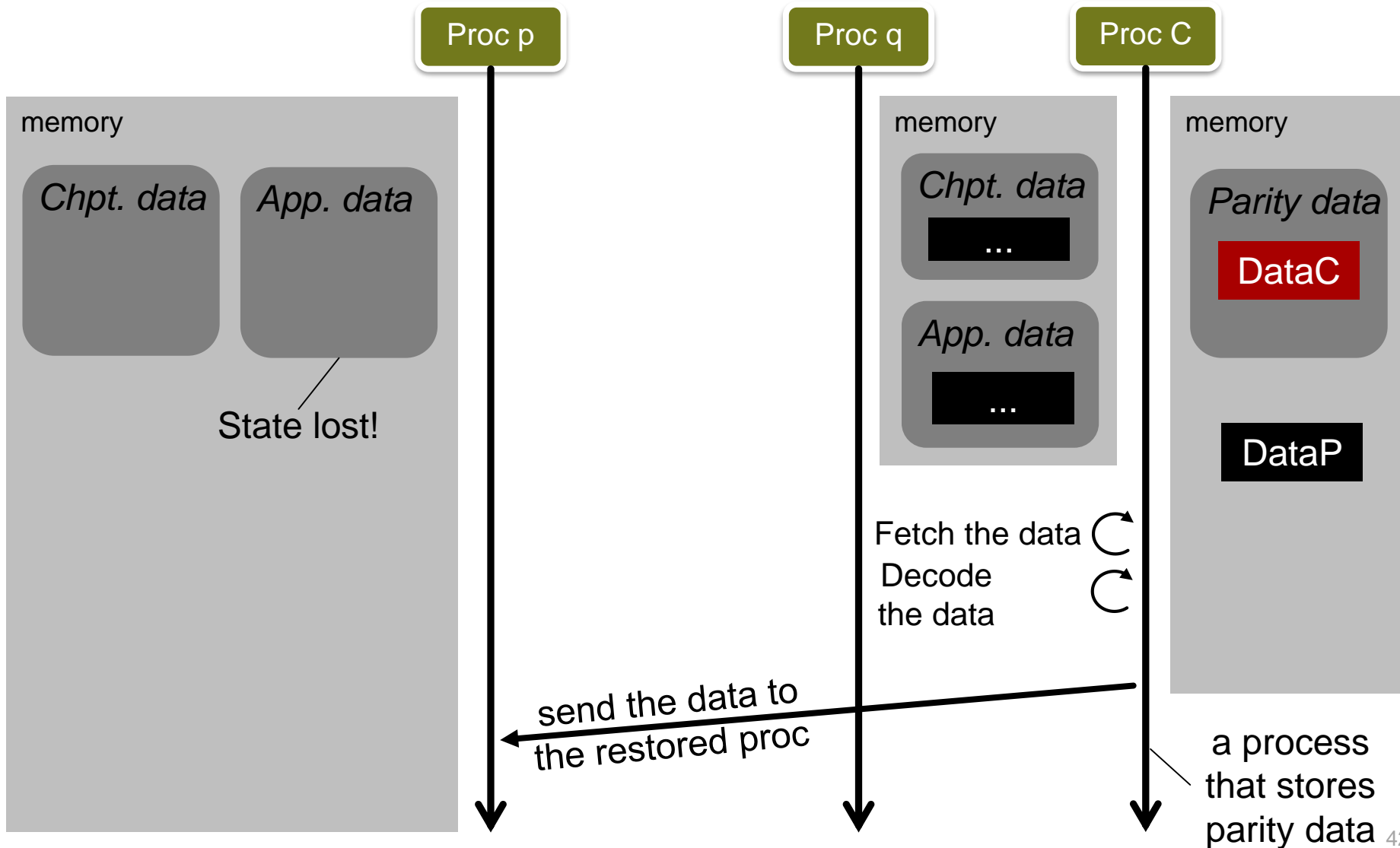
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



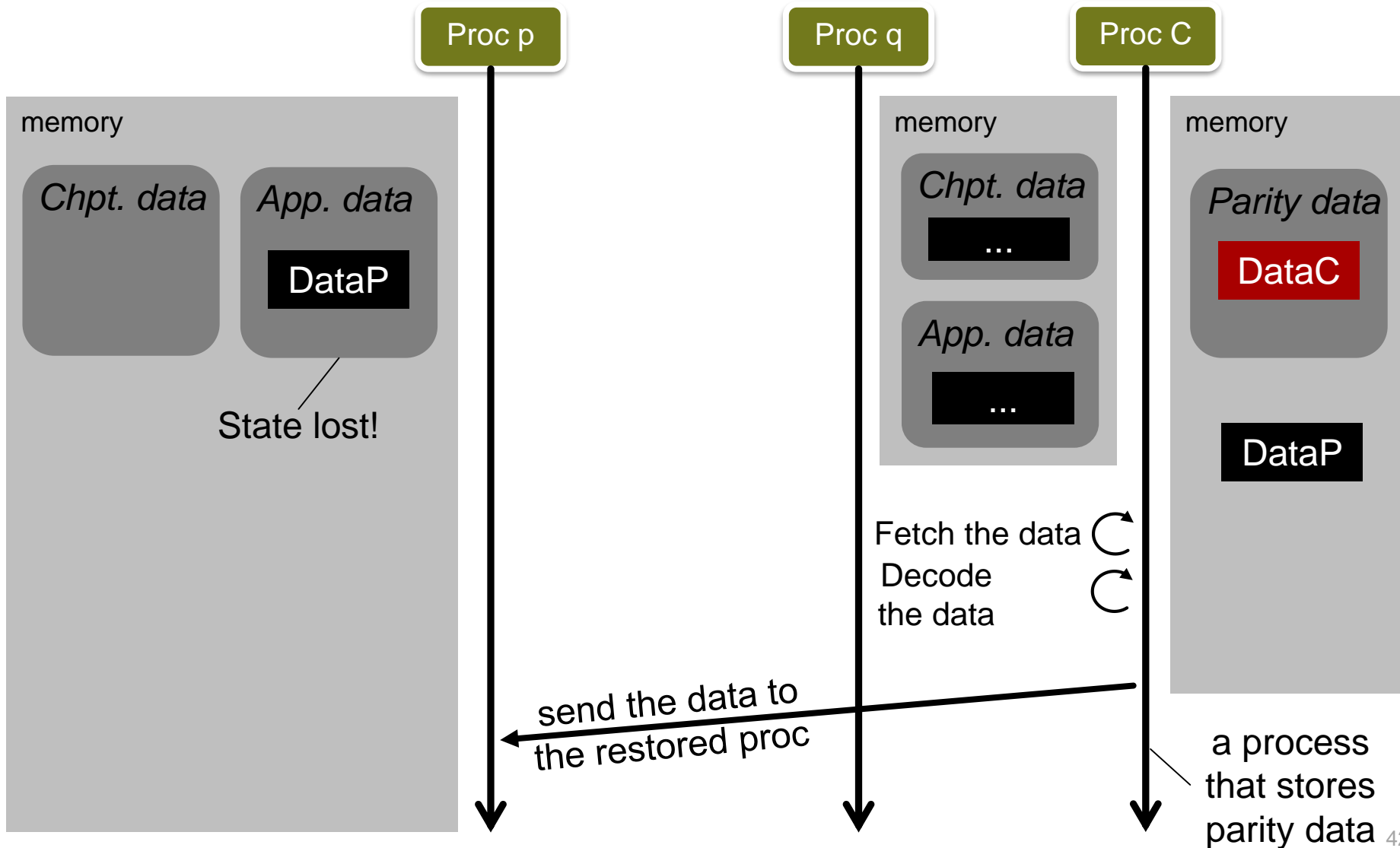
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



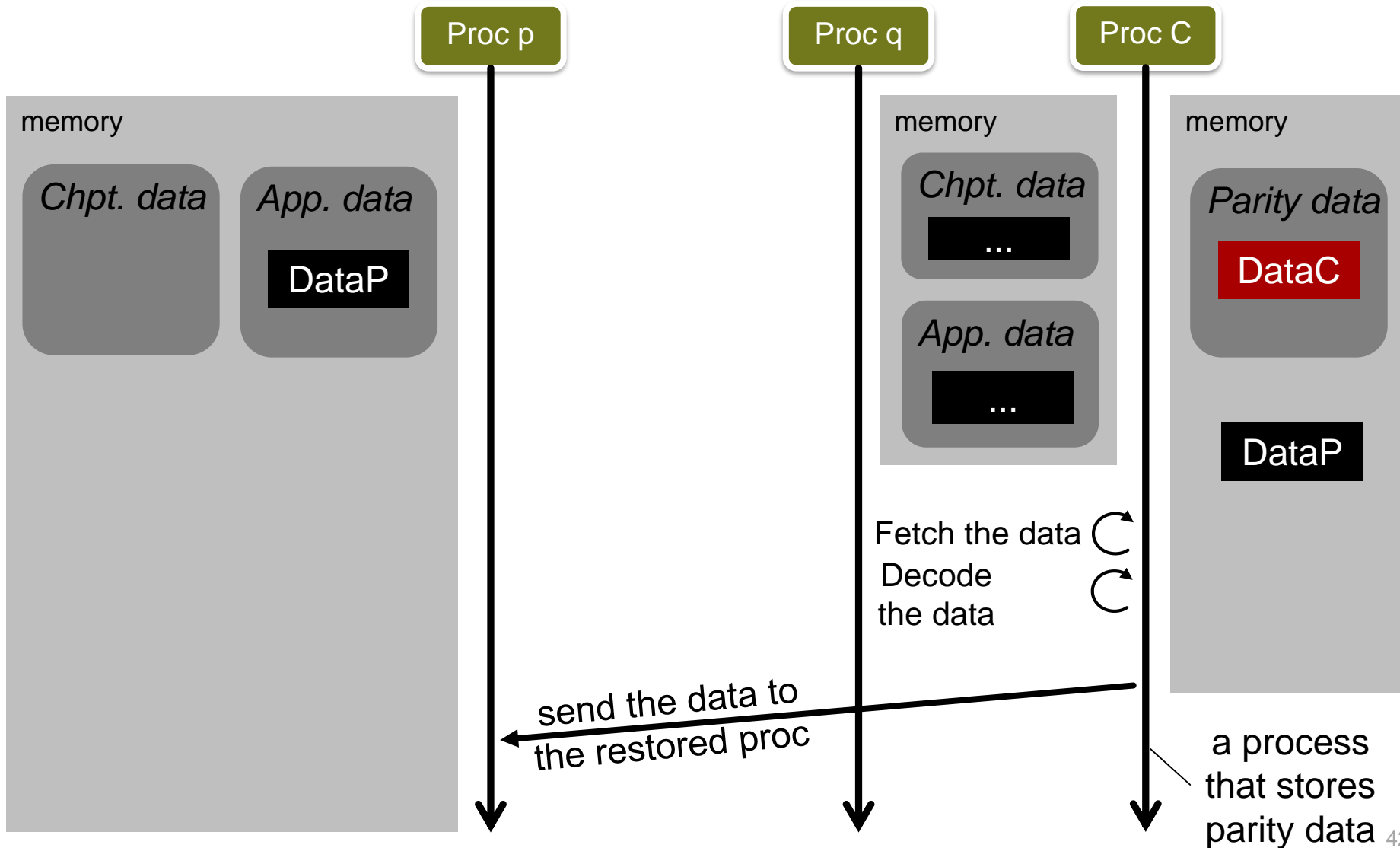
RMA: RECOVERY

Stage 1: restore the state upon checkpointing



RMA: RECOVERY

Stage 1: restore the state upon checkpointing



RMA: RECOVERY

