*Exceptional service in the national interest*

Sandia National Laboratories

# XASM: A Cross-Enclave Composition Mechanism for Exascale System Software

Noah Evans, Kevin Pedretti, Brian Kocoloski, John Lange, Michael Lang, Patrick G. Bridges
nevans@sandia.gov
6/1/16

U.S. DEPARTMENT OF ENERGY NNSA

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

# Outline

- **Application composition and why it matters**
- Hobbes: System software support for application composition
- XASM: Cross Enclave Shared Memory
  - Conceptual modifications needed for Hobbes
  - Implementation on the Kitten lightweight kernel
  - Performance evaluation
- Future work
- Conclusions

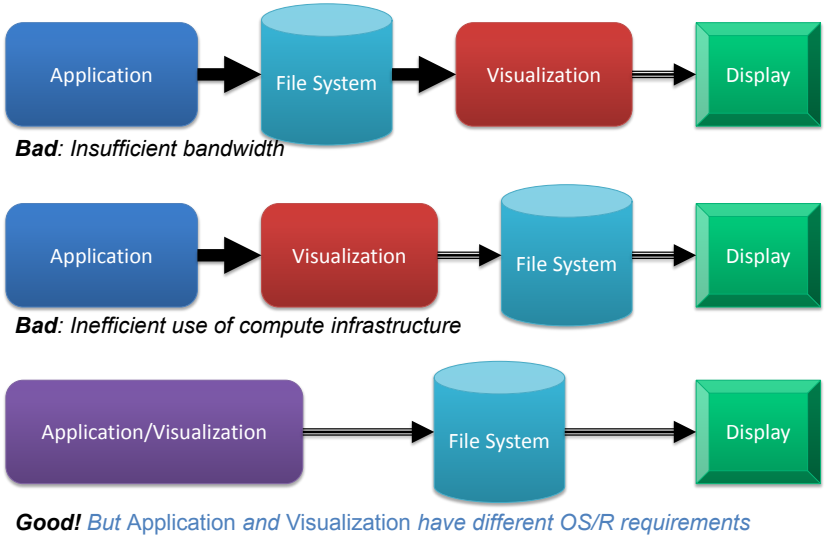
# Composition Use Cases in Next-Generation HPC

Sandia National Laboratories

- End-to-end science workflows
  - Coupled simulation, analysis, and tools
  - In-situ and in-transit analytics
- Multi-physics applications
- Application Introspection
  - Performance analysis, concurrency throttling
  - Debugging
- This presentation concentrates on co-located simulation and analytics workloads

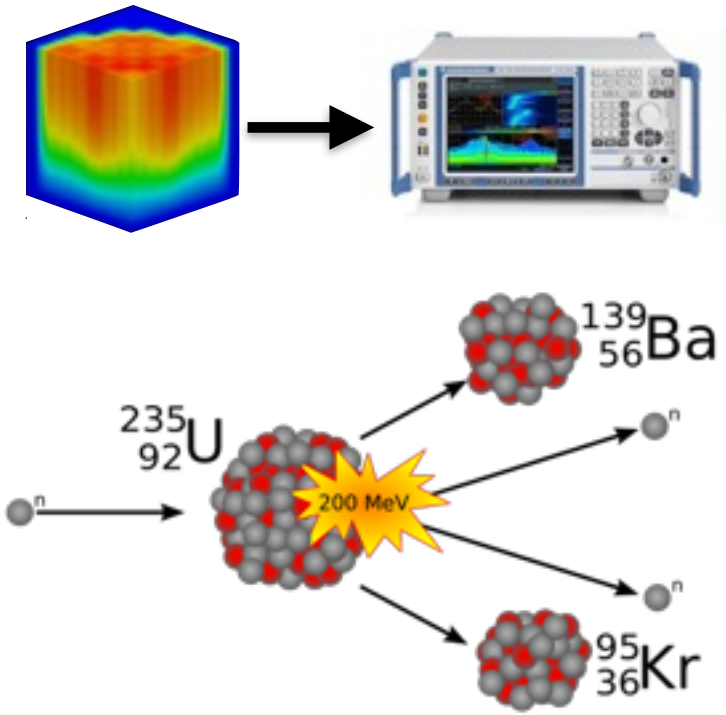
# Why Composition is Important

- Data movement is expensive
  - Writes to filesystem especially
- Need to compartmentalize complexity
  - Jamming everything into one executable is a pain, fragile



# Example: SNAP and Spectrum Analysis

- SNAP
  - Neutronics proxy, based on PARTISN
  - Simulates reactor using sweep3d
- Spectrum analysis
  - After each timestep
- Two separate processes communicating



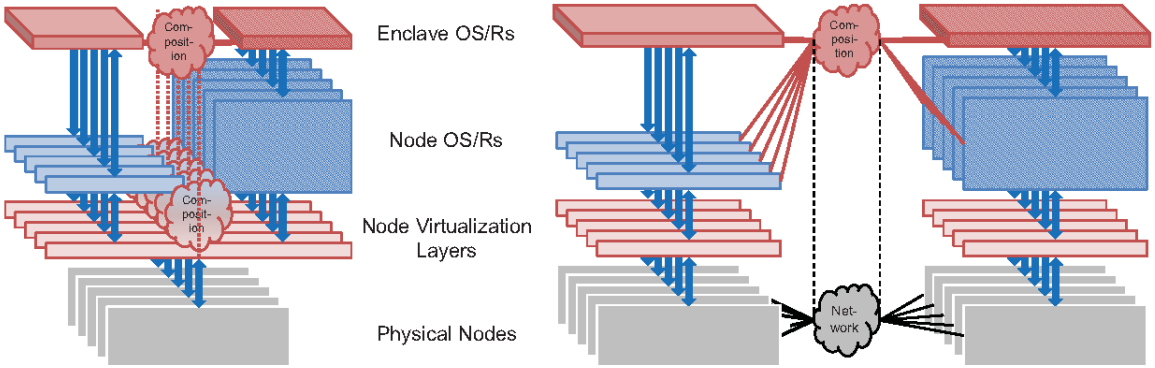
# Outline

- Application composition and why it matters
- **Hobbes: System software support for application composition**
- XASM: Cross Enclave Shared Memory
  - Conceptual modifications needed for Hobbes
  - Implementation for Linux and Kitten lightweight kernel
  - Performance evaluation
- Future work
- Conclusions

# Hobbes Project: Systems Software Support for Composition

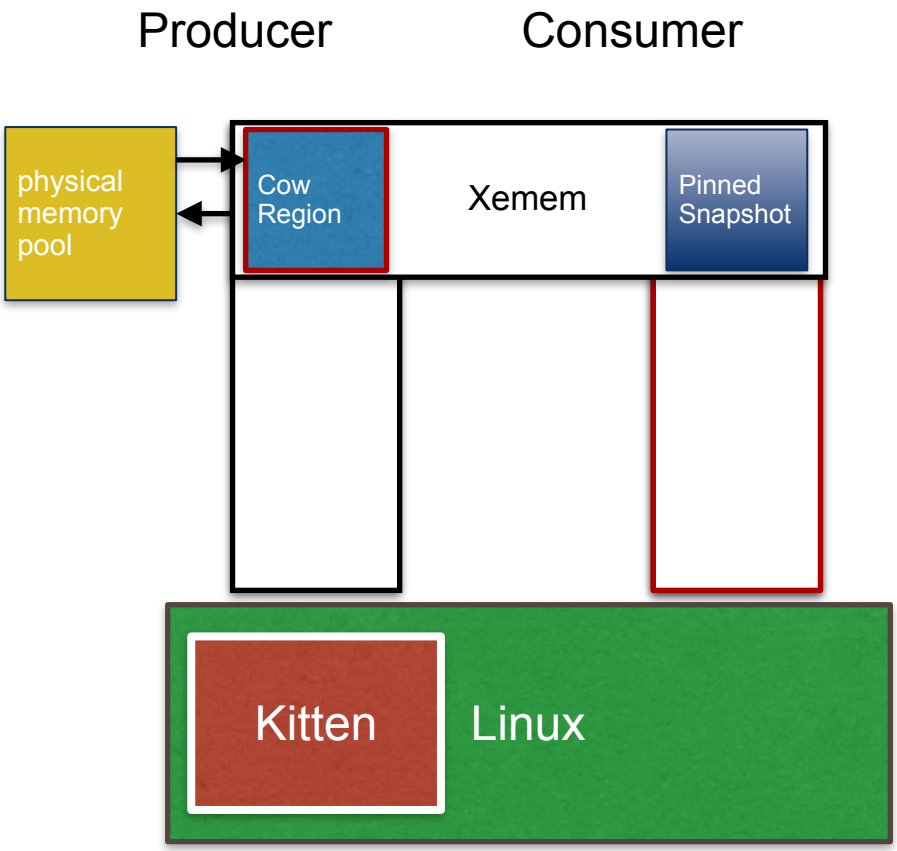
Sandia National Laboratories

- Application level composition difficult for application writer
- Lots of research on how to support (Adios ’10, Gold-rush ’13)

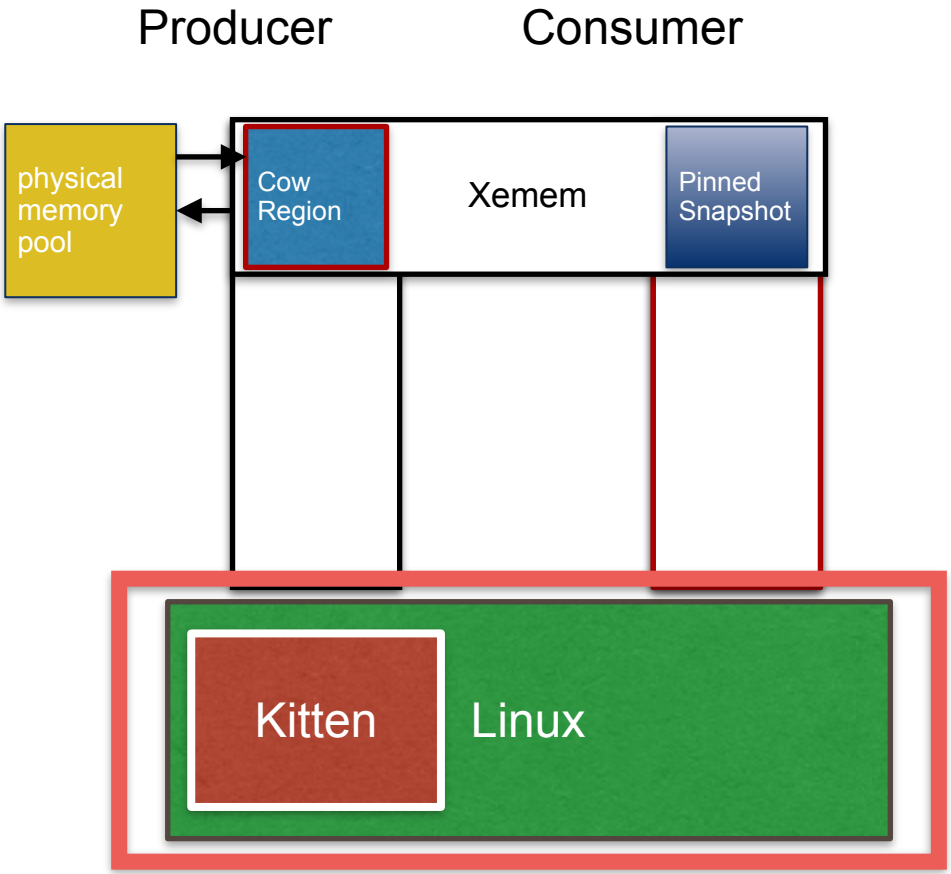



- **Goals**
  - Minimize Data movement in composition
  - Optimizing the scheduling of composed workloads

# Hobbes Project: Why Systems Software Should Support Composition

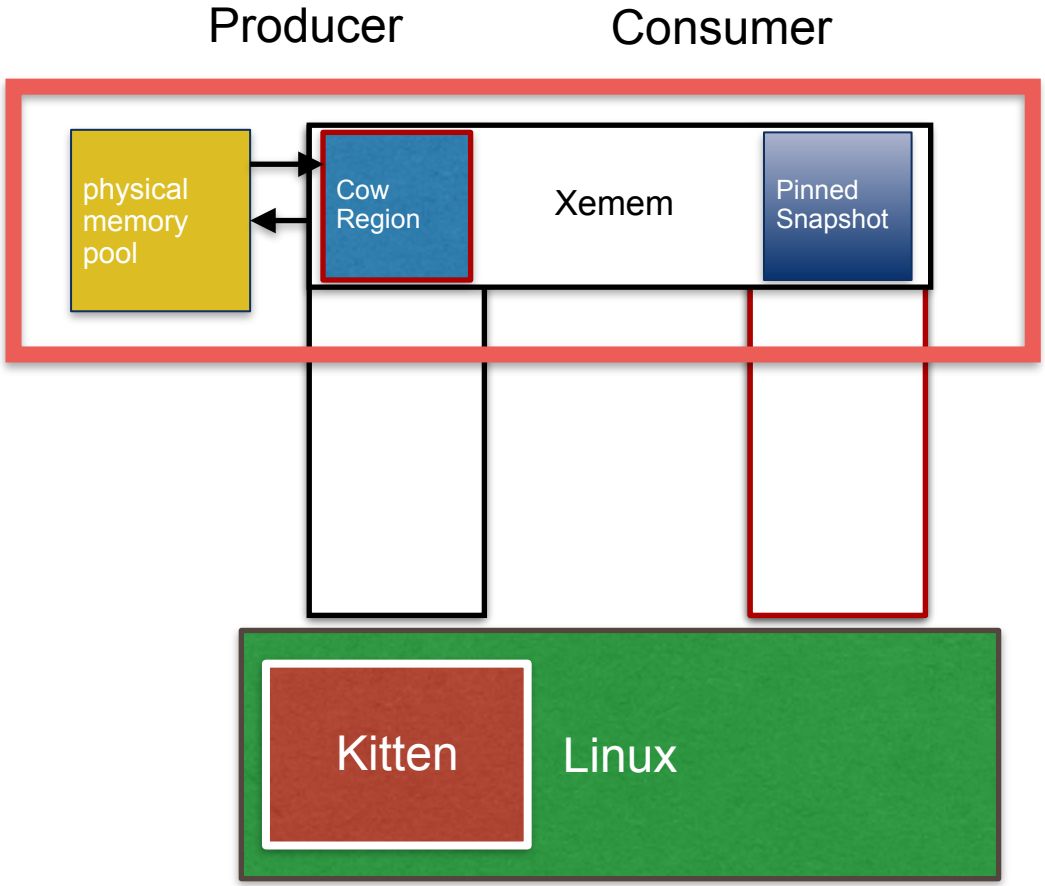


# Hobbes Project: Why Systems Software Should Support Composition




- Space sharing and time sharing virtualization using “Enclaves”

# Hobbes Project: Why Systems Software Should Support Composition



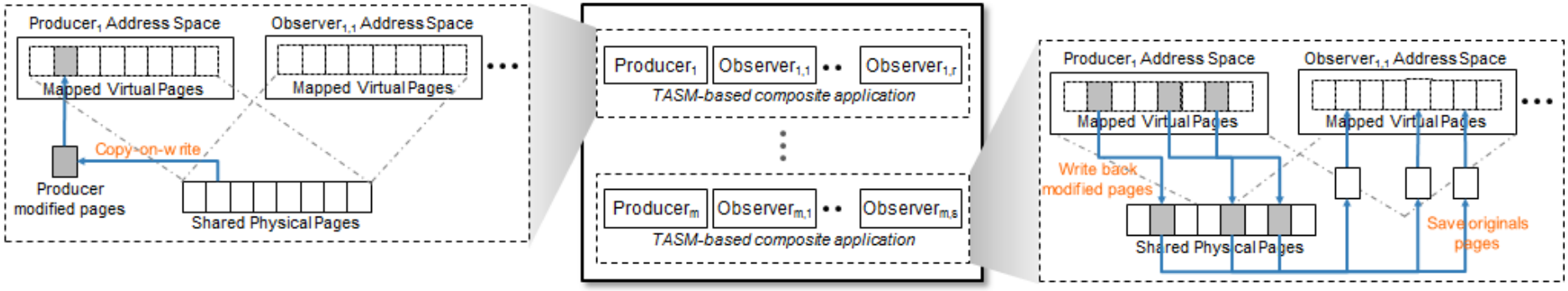

- Space sharing and time sharing virtualization using “Enclaves”
- Communicate using optimized transports

# Outline

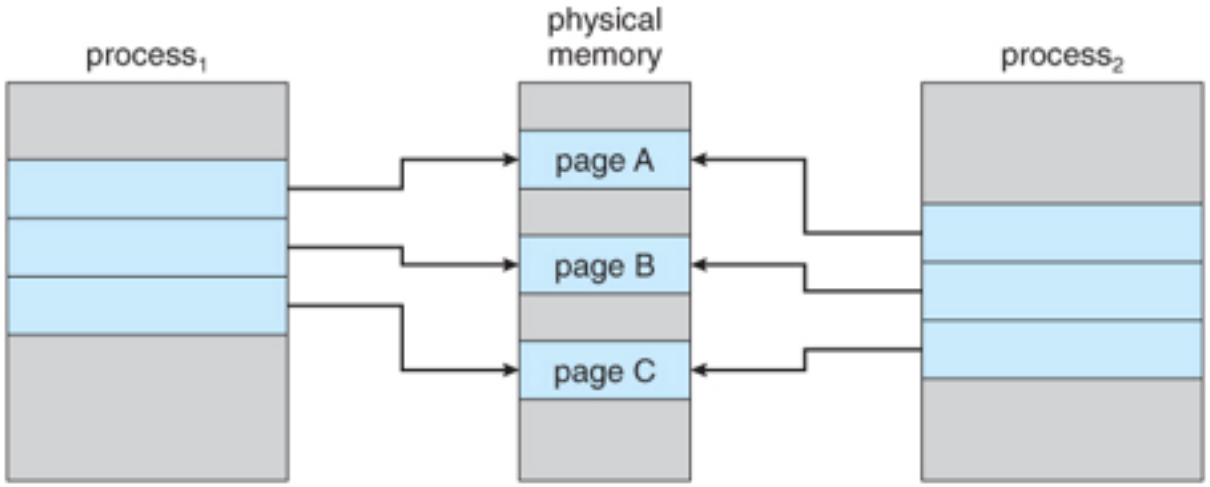
- Application composition and why it matters
- Hobbes: System software support for application composition
- **XASM: Cross Enclave Shared Memory**
  - **Conceptual modifications needed for Hobbes**
  - Implementation on the Kitten lightweight kernel
  - Performance evaluation
- Future work
- Conclusions

# XASM: Optimizing Data Movement for Composition

- **Transparent**: No changes to APIs
- **Consistent**: Neither side, producer or consumer, sees changes made by the other
- **Asynchronous**: No locking needed



# Trick: Copy On Write




- Allows “lazy” copying of data - can avoid the copy in some situations
- OS notified when process trying to modify shared page

- No modification = no copy
- Modification incurs the extra cost of a page fault

# Outline

- Application composition and why it matters
- Hobbes: System software support for application composition
- XASM: Cross Enclave Shared Memory
  - Conceptual modifications needed for Hobbes
  - **Implementation on the Kitten lightweight kernel**
  - Performance evaluation
- Future work
- Conclusions

# Kitten Implementation

Sandia National Laboratories

- Implementations heavily dependent on virtual memory systems
- How are the virtual to physical mappings are maintained will affect contention and allocation policy
- Need to optimize contention and allocation tradeoffs for performance

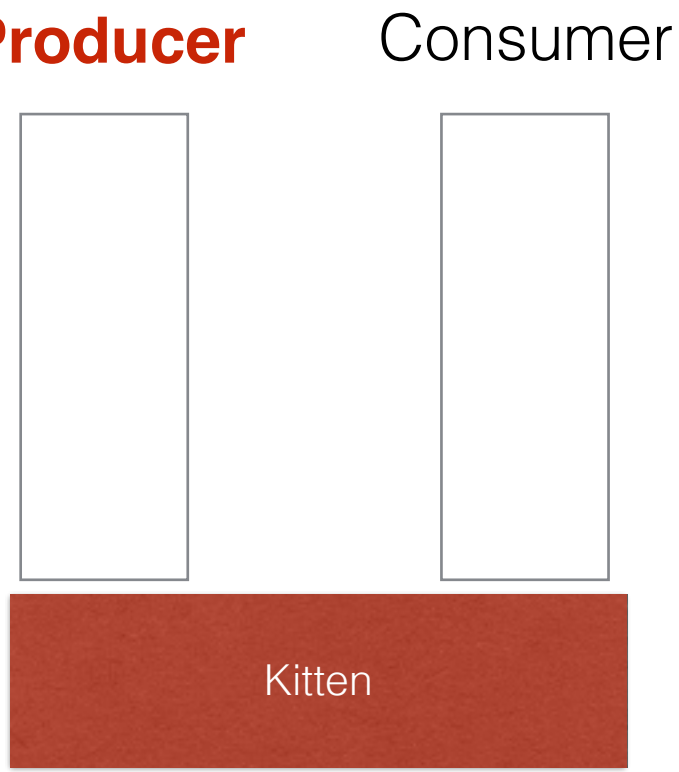
# Kitten Virtual Memory

Sandia National Laboratories

- In Kitten, user allocates physical memory explicitly
  - User chooses own virtual to physical mappings
- Kitten flat-mapped, **no page faults!**
- **Additions to Kitten needed for XASM:**
  - Add a page fault handler to Kitten
  - Add a mechanism to make physical memory pools available to individual processes

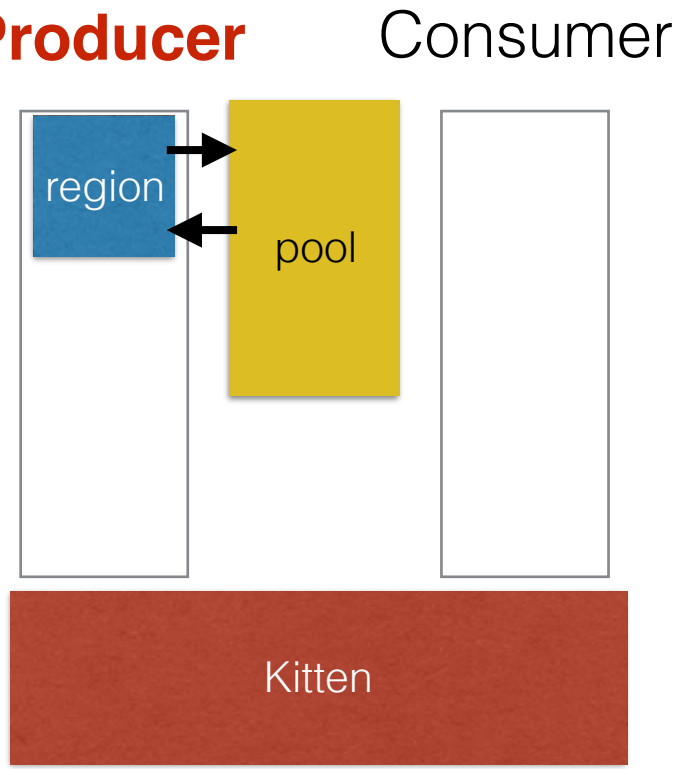
# Kitten XASM

```
// PRODUCER
arena_map_backed_region_anywhere(my_aspace, &region,
…);
for(i=0; i < datalen; i++)
     simulate(data[i]);
aspace_copy(id, &dst, 0);
```



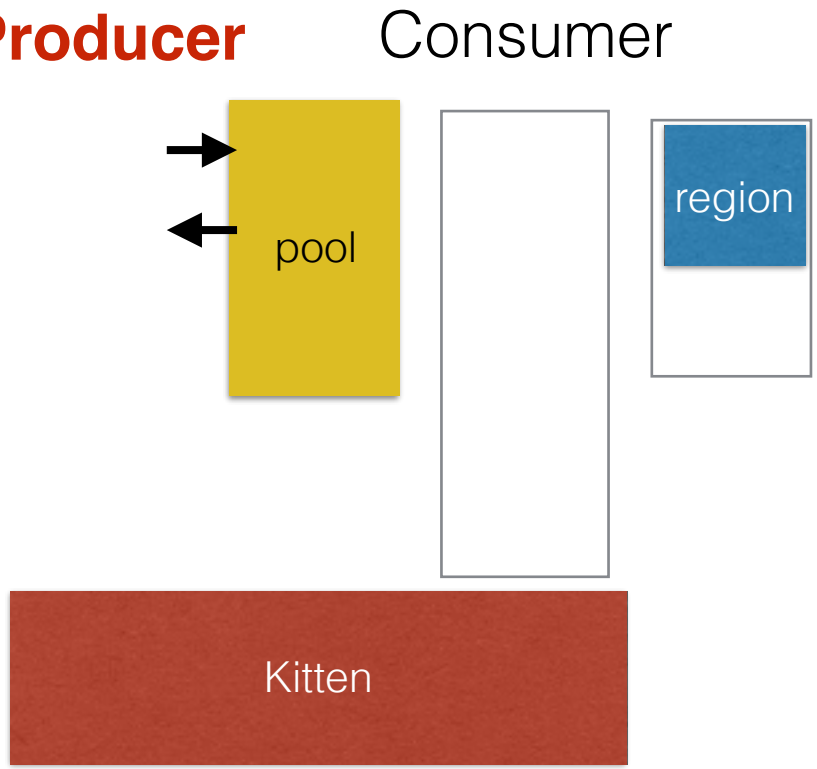
# Kitten XASM

```
// PRODUCER
arena_map_backed_region_anywhere(my_aspace, &region,
…);
for(i=0; i < datalen; i++)
     simulate(data[i]);
aspace_copy(id, &dst, 0);
```



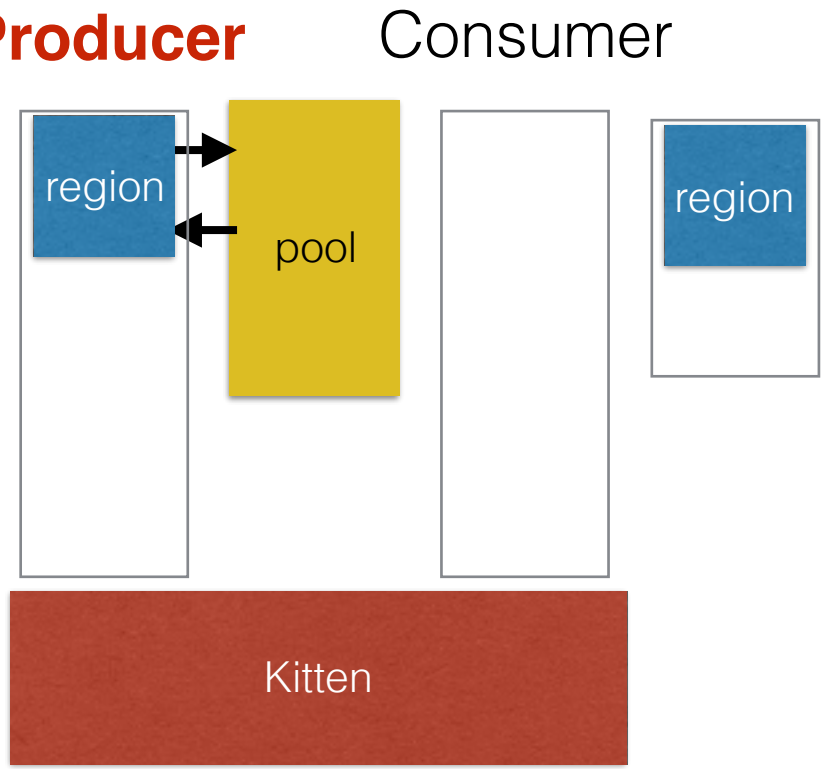
# Kitten XASM

```
// PRODUCER
arena_map_backed_region_anywhere(my_aspace, &region,
…);
for(i=0; i < datalen; i++)
     simulate(data[i]);
aspace_copy(id, &dst, 0);
```



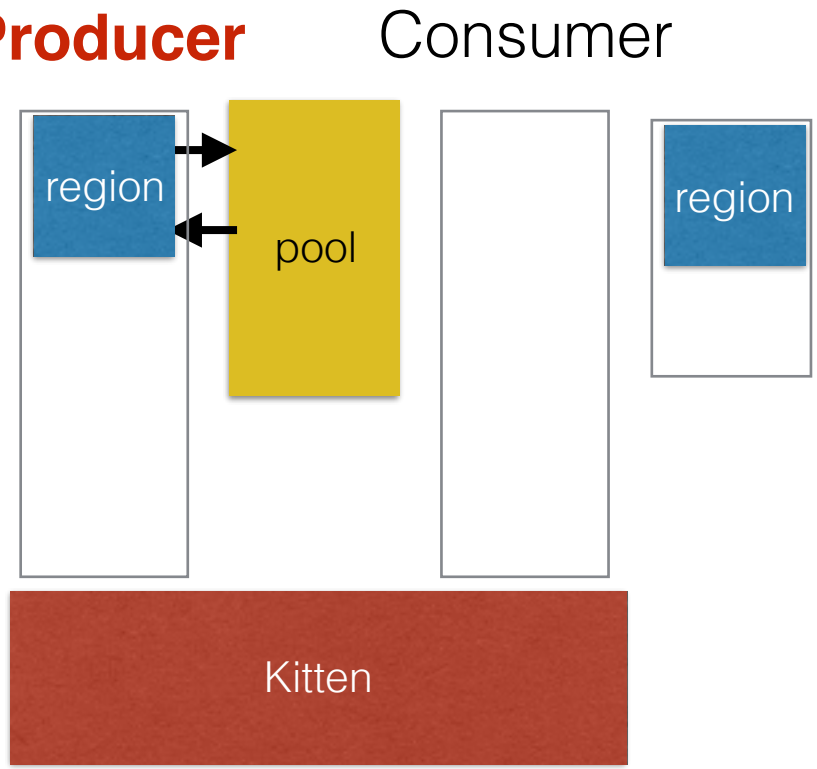
# Kitten XASM

```
// PRODUCER
arena_map_backed_region_anywhere(my_aspace, &region,
…);
for(i=0; i < datalen; i++)
     simulate(data[i]);
aspace_copy(id, &dst, 0);
```



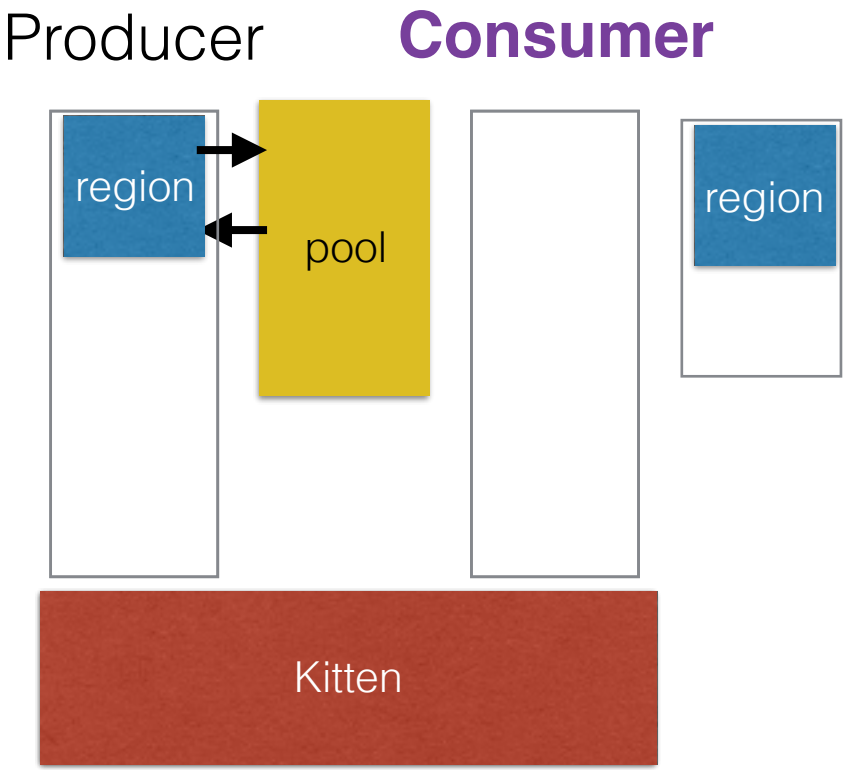
# Kitten XASM

```
// PRODUCER
arena_map_backed_region_anywhere(my_aspace, &region,
…);
for(i=0; i < datalen; i++)
     simulate(data[i]);
aspace_copy(id, &dst, 0);
```



# Kitten XASM

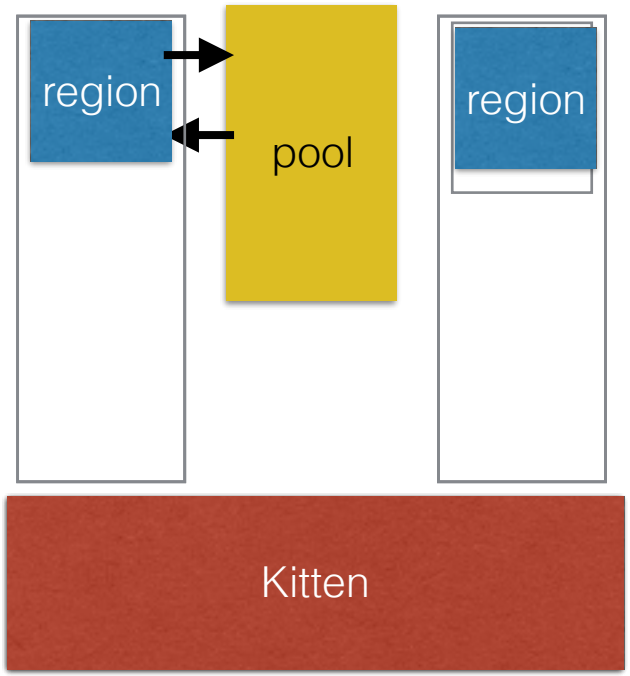
```
// CONSUMER
aspace_smartmap(xasm_id, my_id, SMARTMAP_ALIGN,
SMARTMAP_ALIGN);
for(i=0; i < datalen; i++)
     analyze(data[i]);
aspace_unsmartmap(xasm_id, my_id, …);
aspace_destroy(xasm_id);
```



# Kitten XASM

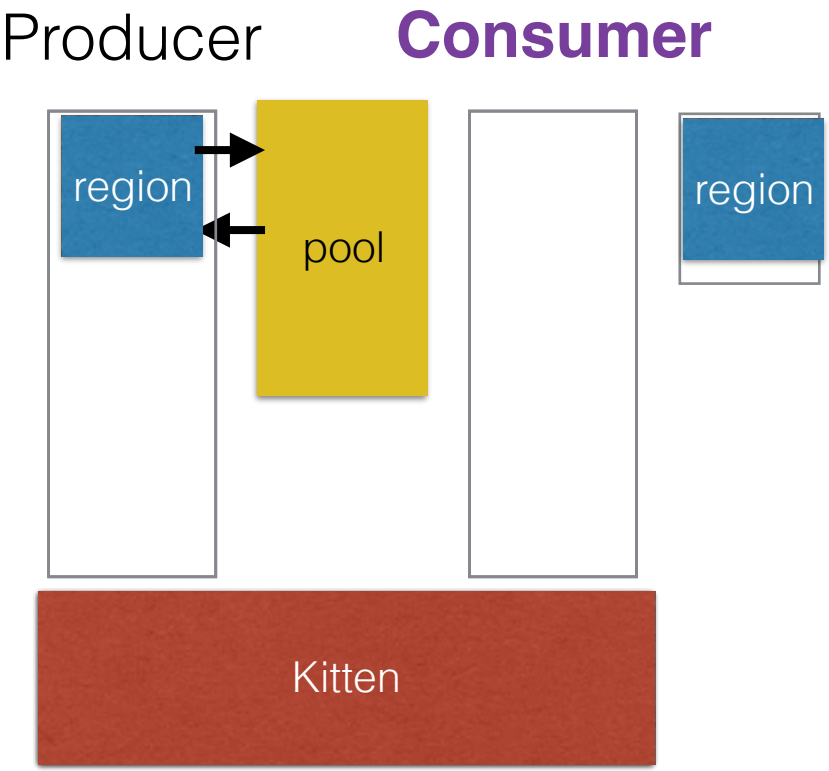
```
// CONSUMER
aspace_smartmap(xasm_id, my_id, SMARTMAP_ALIGN,
SMARTMAP_ALIGN);
for(i=0; i < datalen; i++)
     analyze(data[i]);
aspace_unsmartmap(xasm_id, my_id, …);
aspace_destroy(xasm_id);
```

Producer **Consumer**



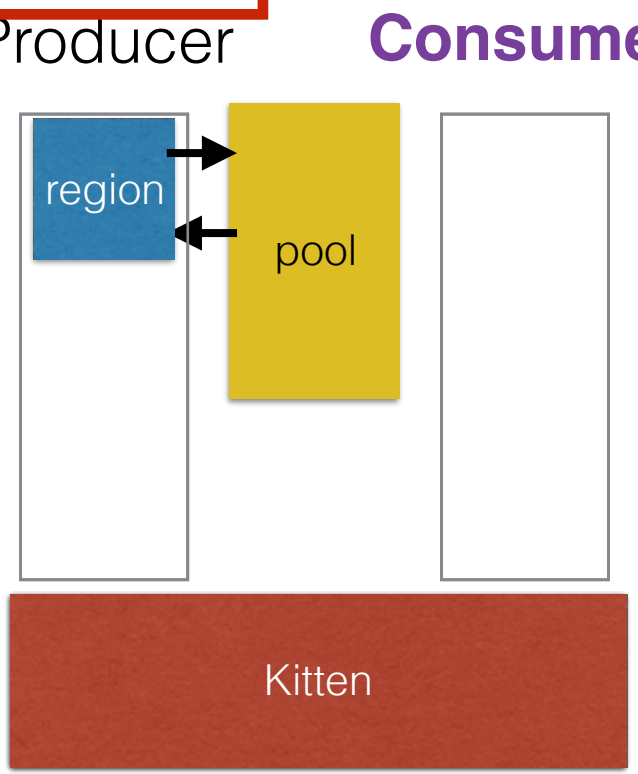
# Kitten XASM

```
// CONSUMER
aspace_smartmap(xasm_id, my_id, SMARTMAP_ALIGN,
SMARTMAP_ALIGN);
for(i=0; i < datalen; i++)
     analyze(data[i]);
aspace_unsmartmap(xasm_id, my_id, …);
aspace_destroy(xasm_id);
```



# Kitten XASM

```
// CONSUMER
aspace_smartmap(xasm_id, my_id, SMARTMAP_ALIGN,
SMARTMAP_ALIGN);
for(i=0; i < datalen; i++)
     analyze(data[i]);
aspace_unsmartmap(xasm_id, my_id, …);
aspace_destroy(xasm_id);
```



# Outline

- Application composition and why it matters
- Hobbes: System software support for application composition
- Xasm: Transparently Consistent Asynchronous Shared Memory
  - Conceptual modifications need for Hobbes
  - Implementation for Linux and Kitten lightweight kernel
  - Performance evaluation
- Future work
- Conclusions

# Performance Evaluation

Sandia National Laboratories

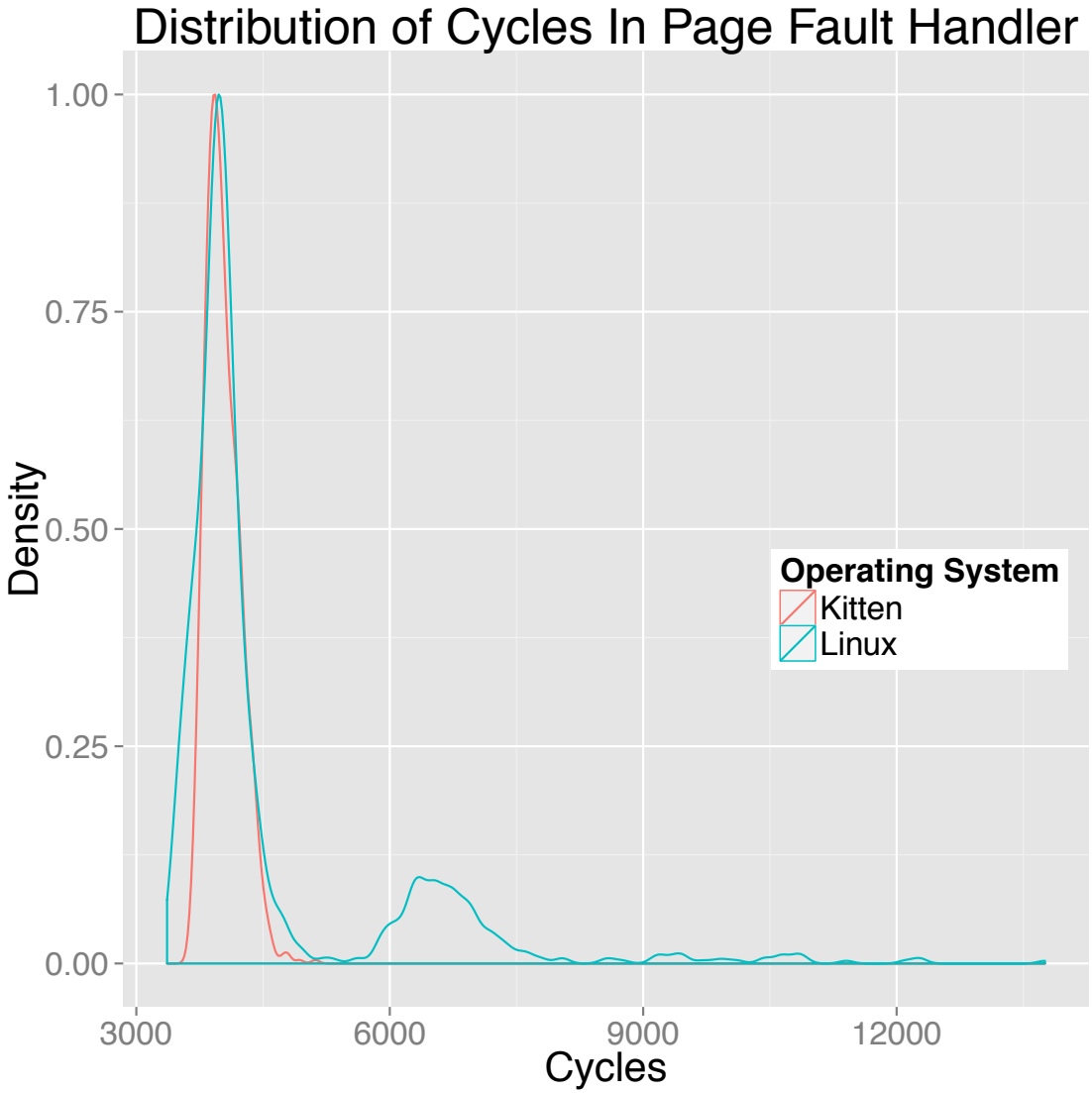
- Need to show that it works with minimal performance overhead
- Questions to answer:
  - What is the overhead of page fault handling?
  - How does the overhead of Xasm compare to base case and synchronized shared memory?

# Experimental Design

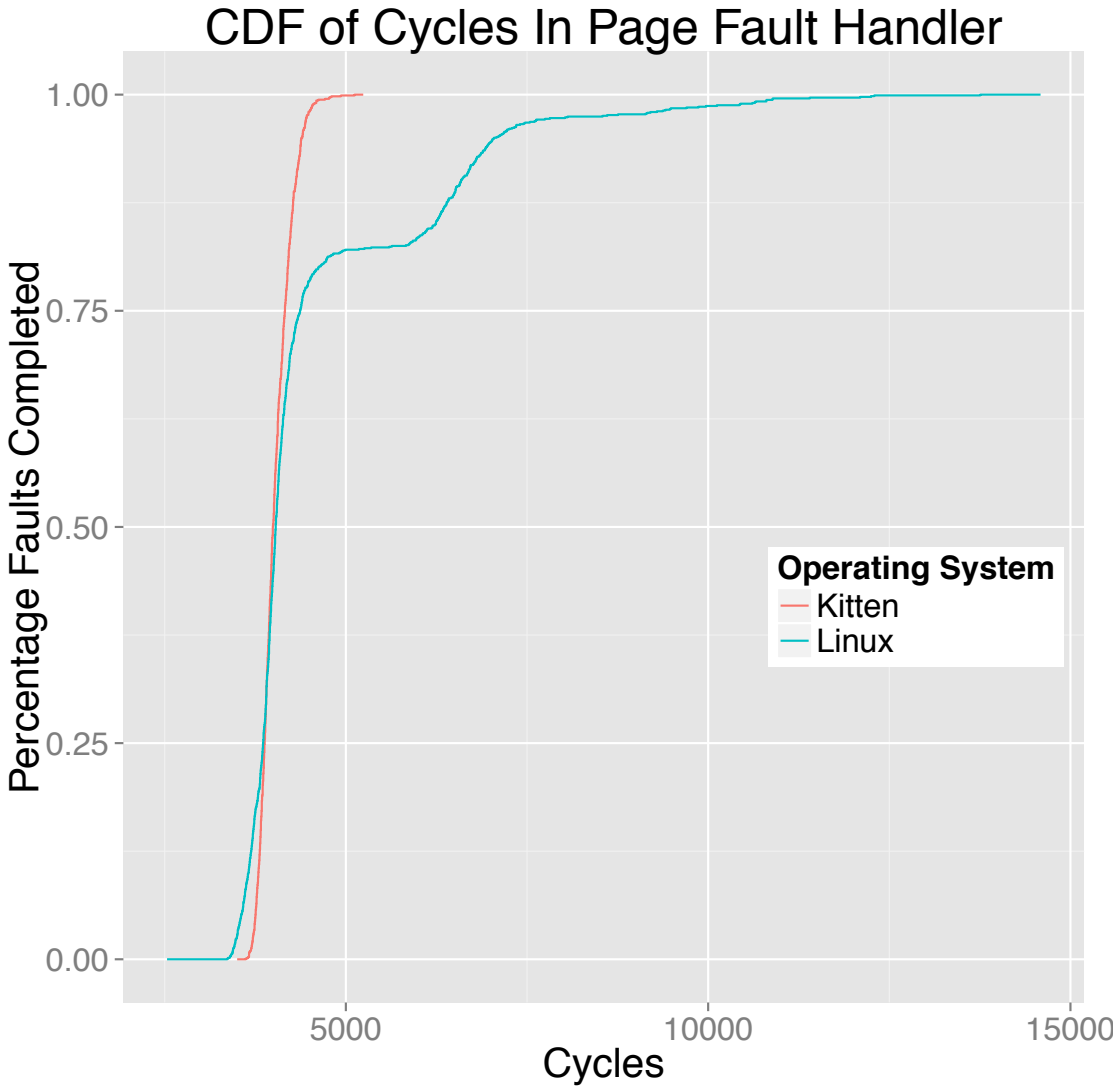
Sandia National Laboratories

- Sandy bridge 2.2 GHz,12 core, 2 socket system, 24 GB (Hyper-threading off)
- Hobbes environment on Linux
- Use cycle counter for kernel measurements of page faults
- SNAP + Spectrum Analysis as macro benchmark
  - Compare worst case (xpmem+spin locks), Xasm, best case (no analytics)
  - Inter-enclave on Kitten (6 trials per size)
  - x*y*z = 96, 200, 324, 490, 768, 6144

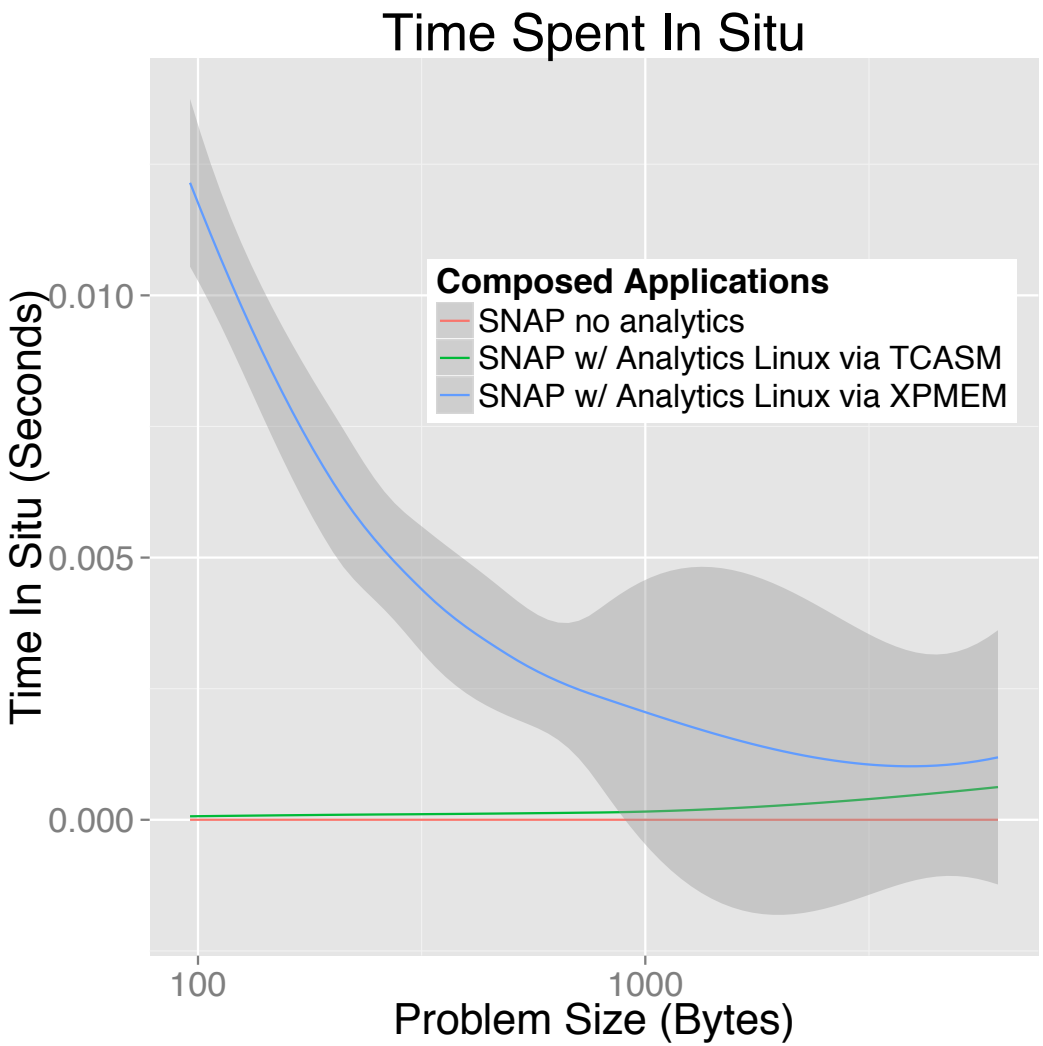
# Kitten faults less noisy



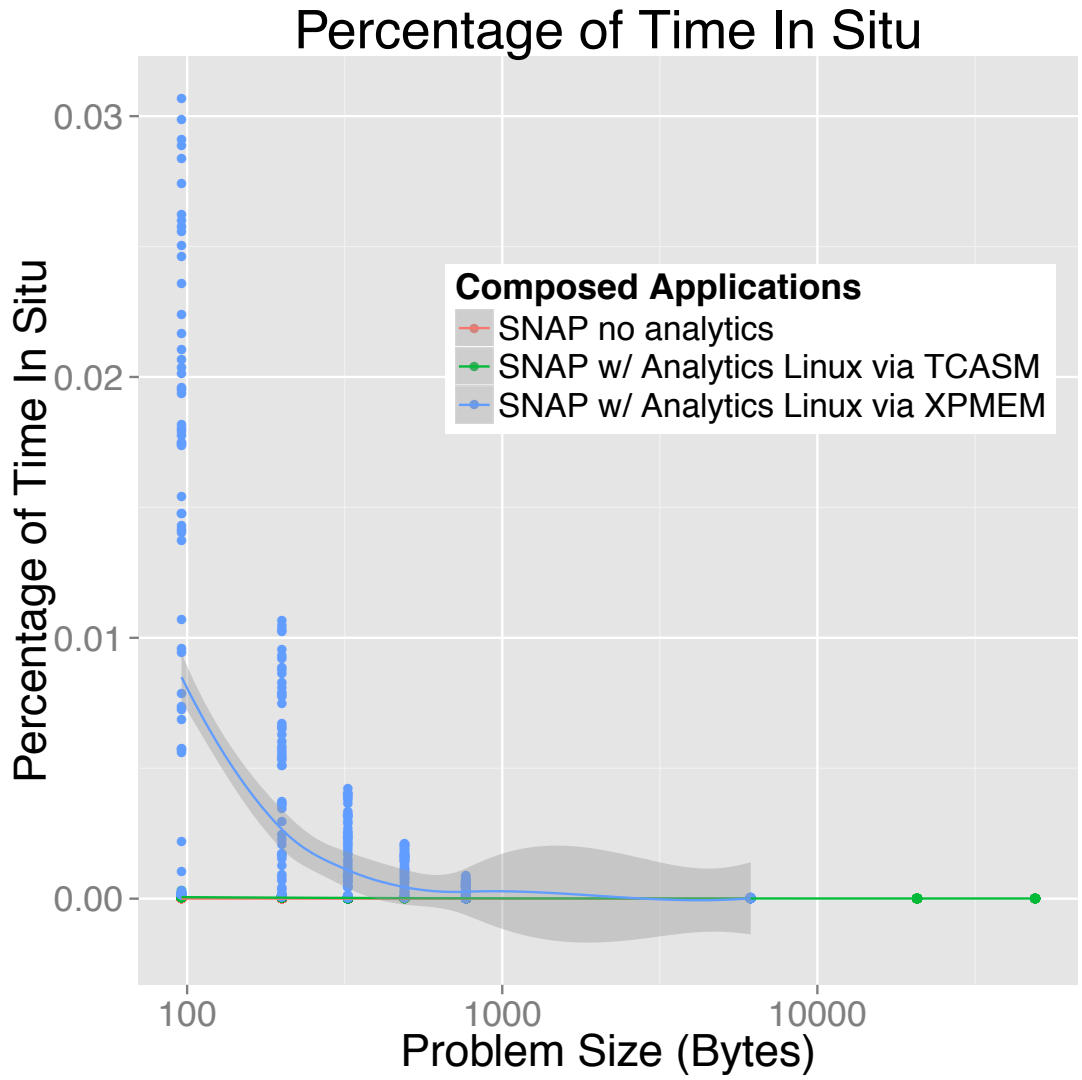
# Linux slower 25% of time



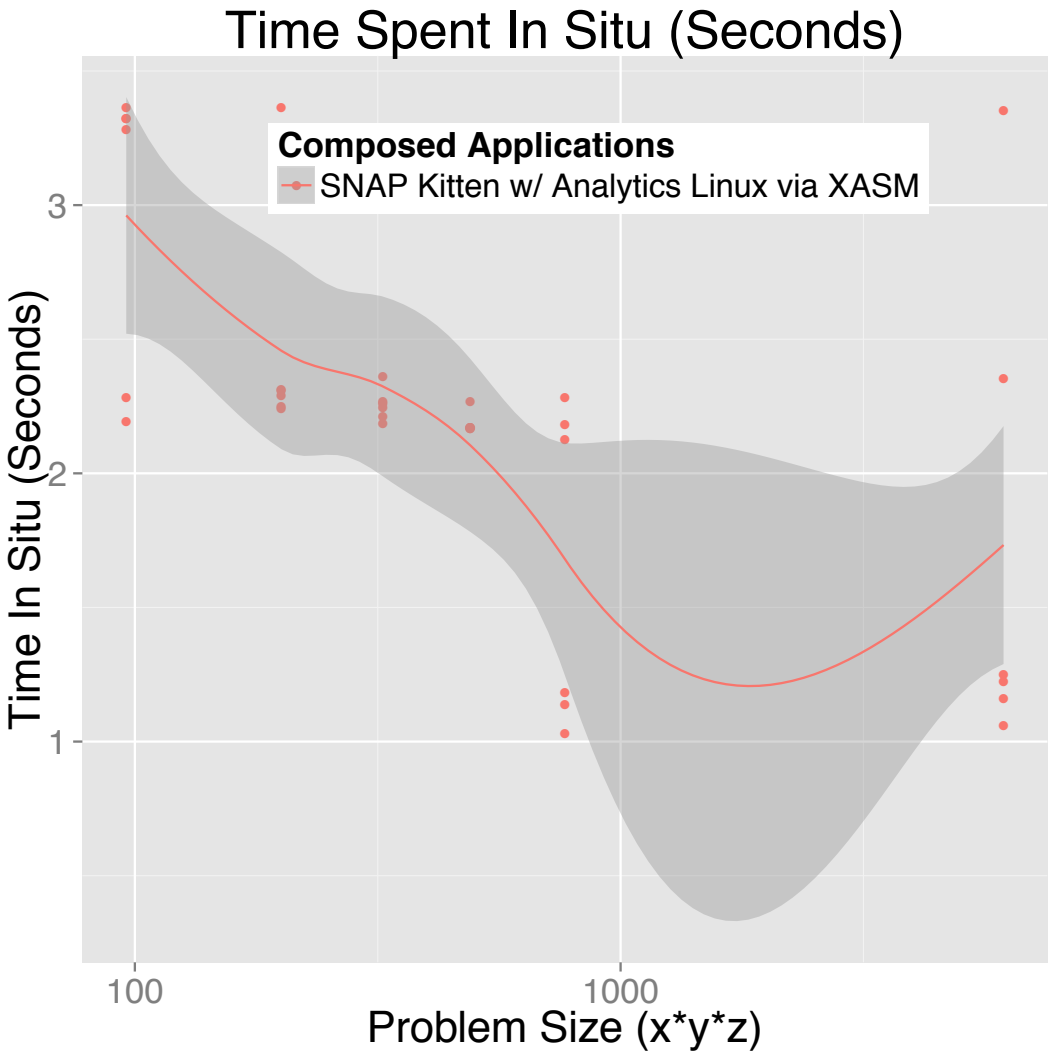
# XASM Overhead Negligible Between Processes



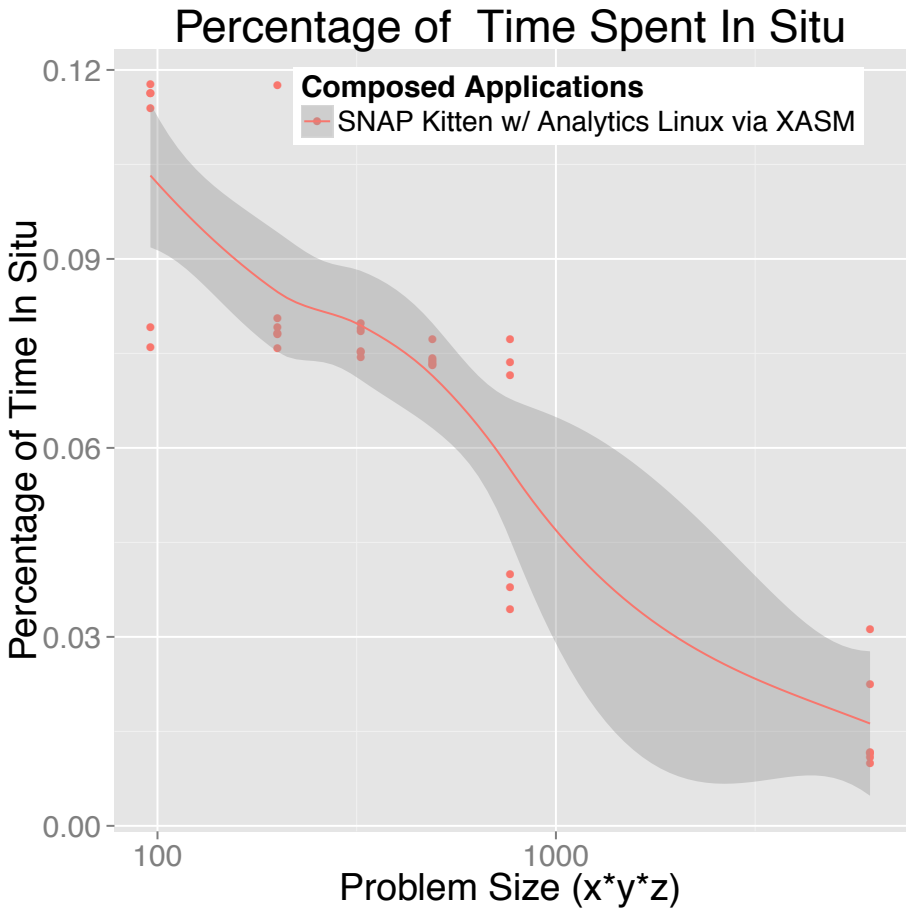
# <4% In Worst Case



# Address Space Copies Expensive but Still Manageable



# Still Less Than 3% of Execution Time with Scale



# Outline

- Application composition and why it matters
- Hobbes: System software support for application composition
- XASM: Transparently Consistent Asynchronous Shared Memory
  - Conceptual modifications need for Hobbes
  - Implementation for Linux and Kitten lightweight kernel
  - Performance evaluation
- Future work
- Conclusions

# Future work

- We know what systems software *can* do in this case but we don't know what it should do
  - For a Copy on Write mechanism what does fault tolerance and flow control look like?
- Copy on Write not necessarily good for some HPC applications
  - Anything that touches every page
  - Wait free queue instead?
- Integrate XASM with in-situ runtimes like ADIOS

# Conclusions

- Systems software can simplify composition
- XASM is one potential mechanism
- XASM is working in the Hobbes OS/R stack
- Performance is reasonable and using XASM is easy
- Potential piece of exascale infrastructure

Sandia National Laboratories

# Thank You