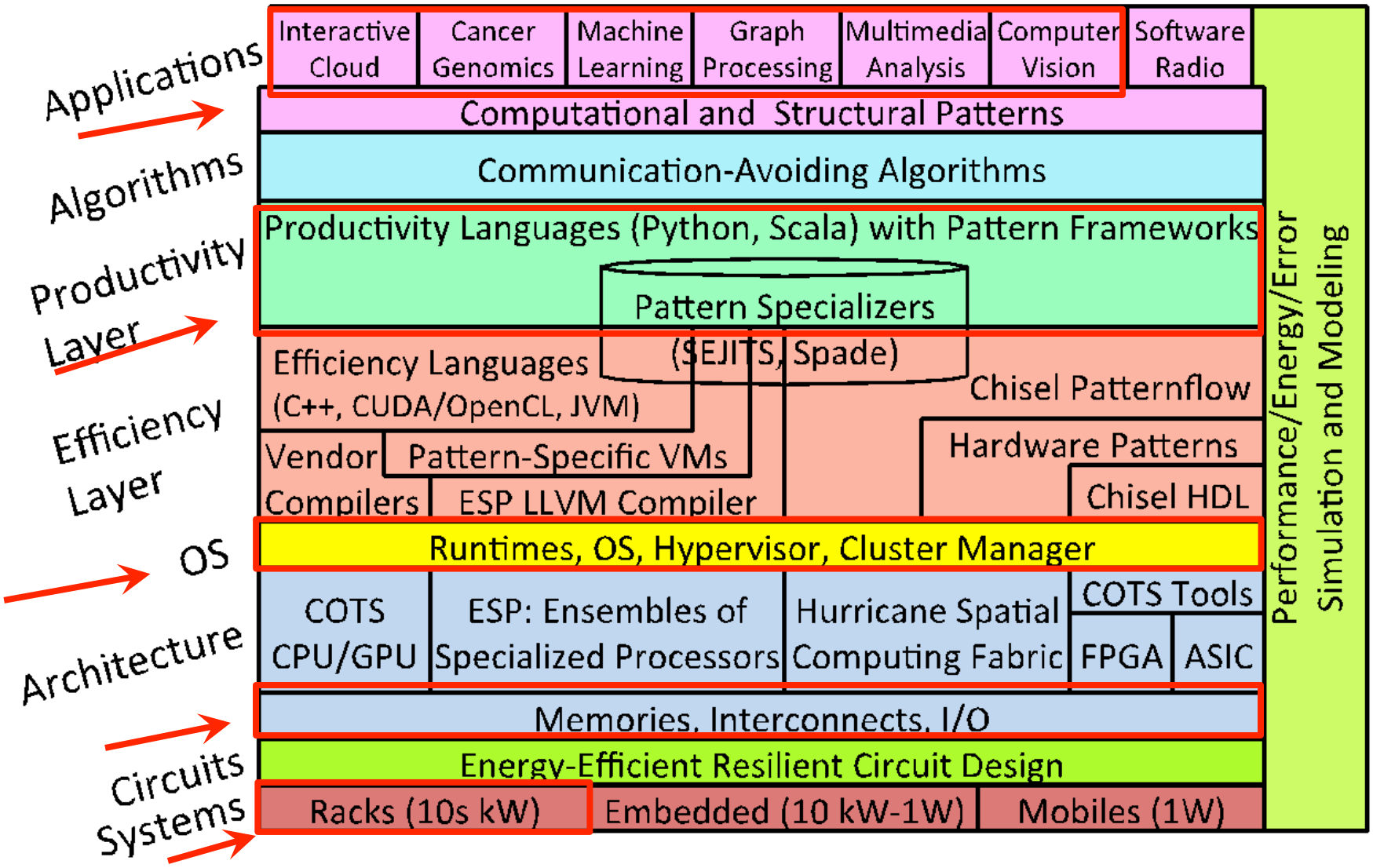
PROTOTYPING

FIREBOX-0

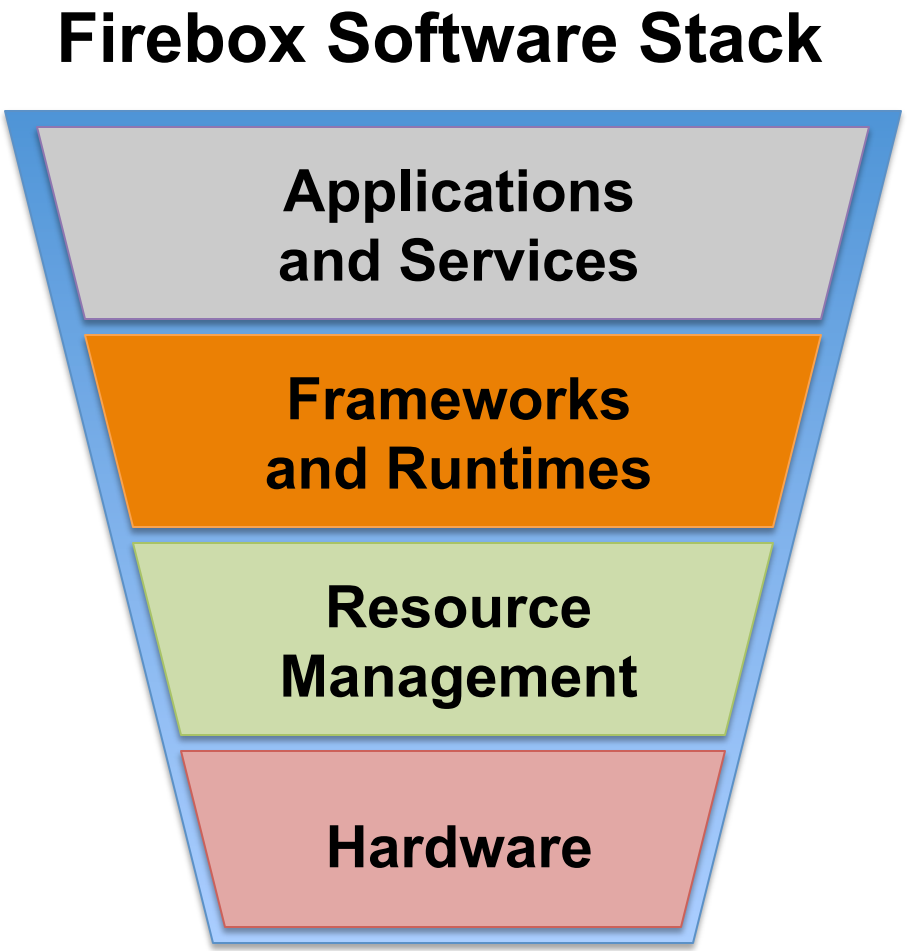
Nathan Pemberton Joao Carreira Zach Rowinski
Martin Maas Frank Nothaft
Krste Asanović Randy Katz

FireBox Stack



This Talk

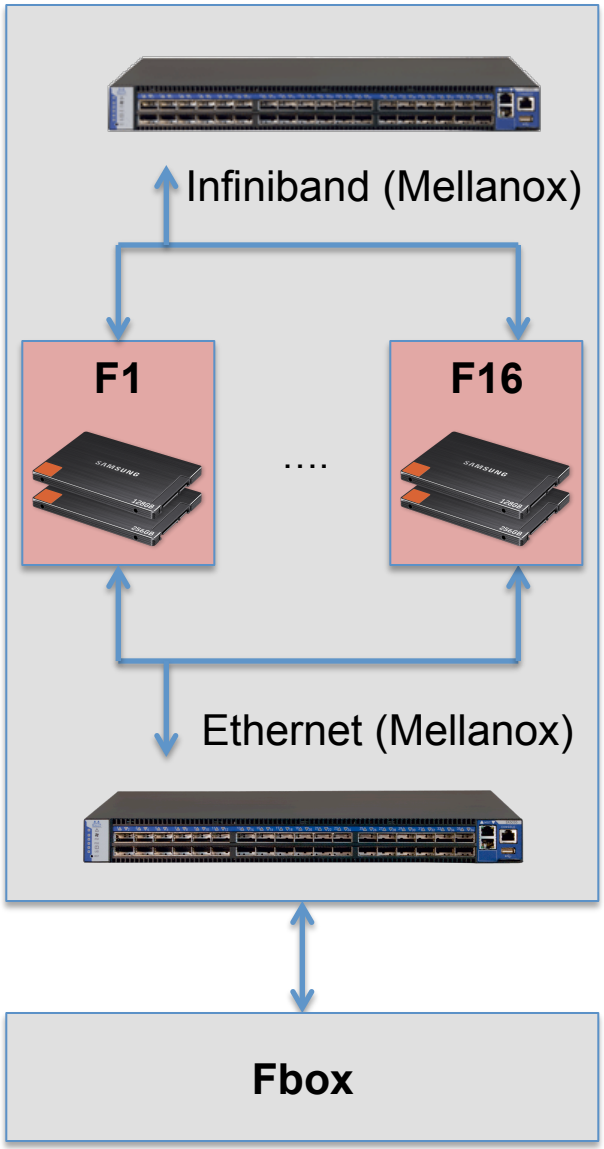
- Talk :
  - Benchmarks of different layers of the stack – BITS (Berkeley Interactive Throughput Suite)
  - Early project building a new Xen-based hypervisor (XARCC)



What is Firebox-0

- Hardware configuration:

- Intel Ivy Bridge E5-1680V2
- 8Cores / 16Threads 3.0GHz 25MB
- 64GB RAM per node (1TB aggregate)
- 2 Samsung SATA SSDs per node (120GB + 480GB) (9.4TB aggregate)
- 2 extra Samsung PCIe SSDs (734GB)
- Infiniband: Mellanox ConnectX-3 VPI FDR QSFP 56Gbps
- Ethernet: Mellanox ConnextX-3 EN QSFP 40Gb
- Running Ubuntu 14.04 (Linux 3.13.0)



Firebox – 0: Goals

- 2014 state-of-the-art baseline using off-the-shelf hardware and software
- platform to analyze current workloads
- platform to develop new software stack ideas
- host XAM-board prototype of Firebox bulk memory and networking

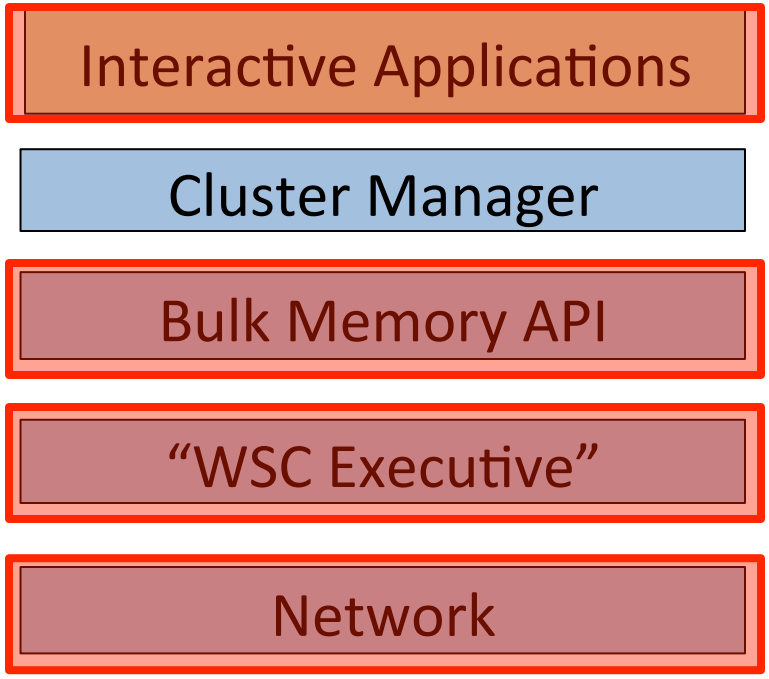
Outline

Apache Solr

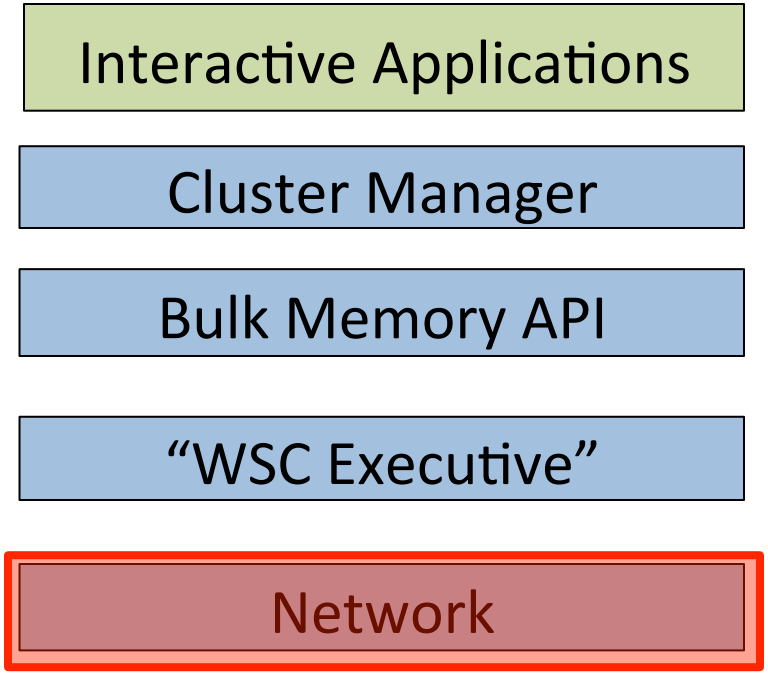
RAMCloud, Aerospike, Cassandra

XARCC

RDMA
Latency / Bandwidth

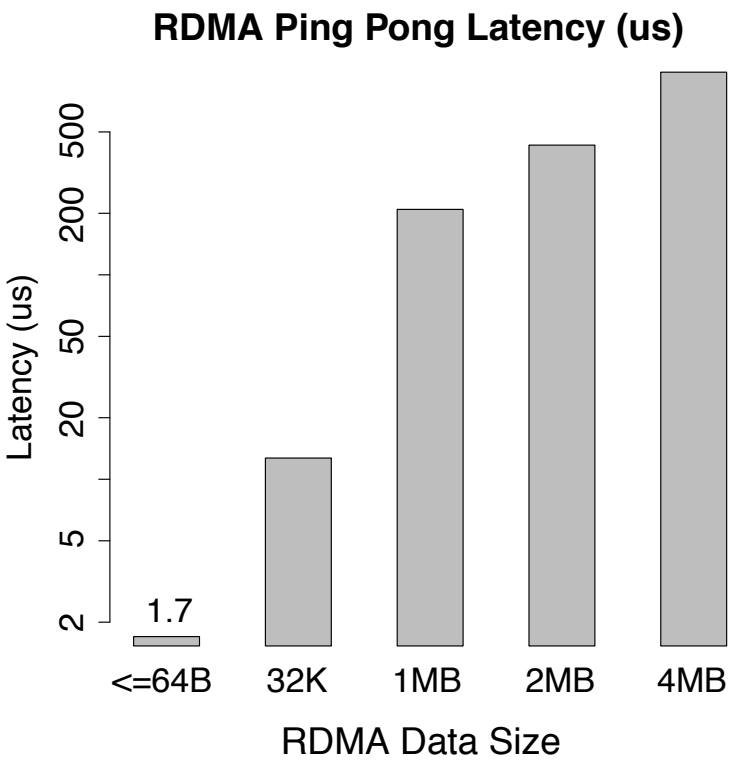


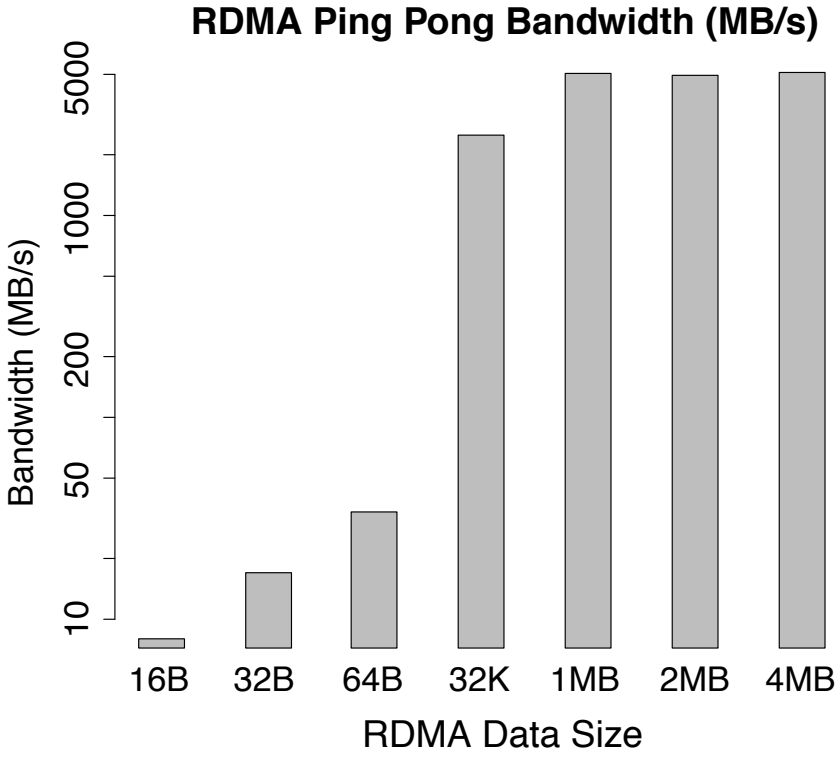
Network



RDMA
Latency / Bandwidth

Firebox-0 Network Latency / Bandwidth





RDMA Ping Pong Benchmark:

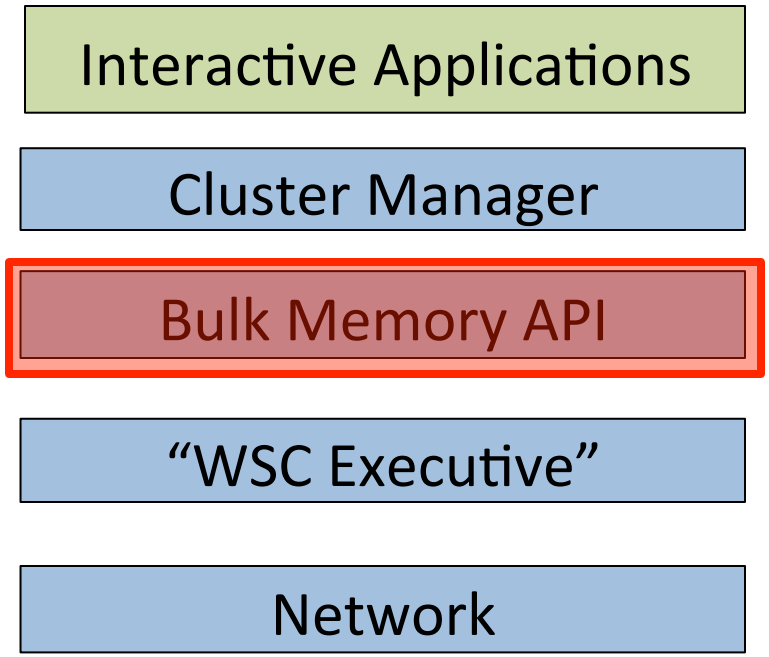
- RDMA connection is setup throughTCP/IP
- Sequential RDMA reads with different sizes

Results:

- Low latency / bandwidth for small packets
- Lowest latency: 1.7us
- Maximum bandwidth of ~5 GB/s

Outline

RAMCloud, Aerospike, Cassandra



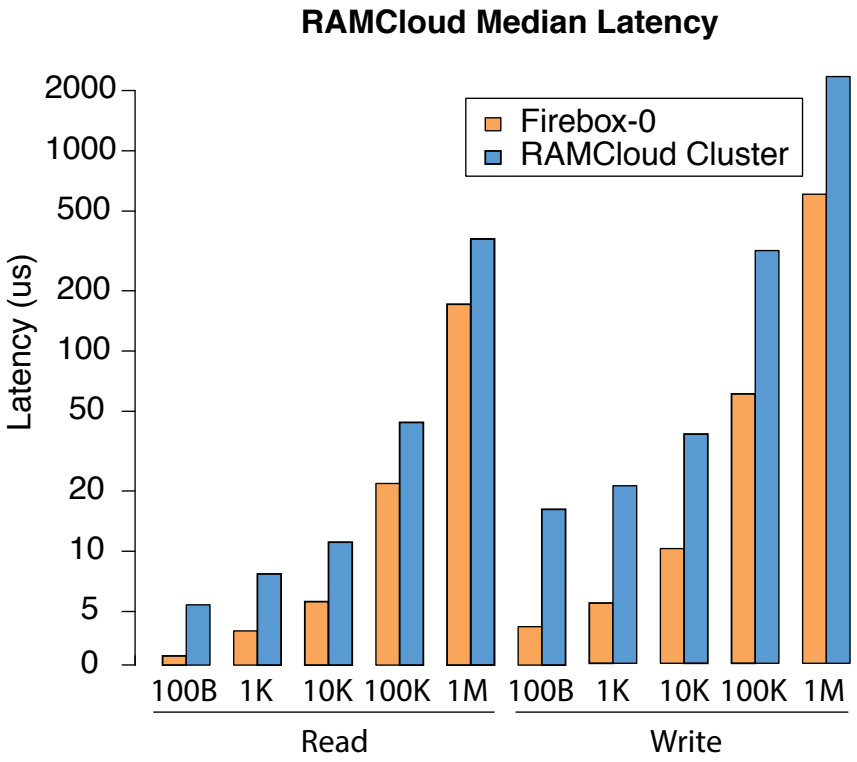
Benchmark of Key-value stores

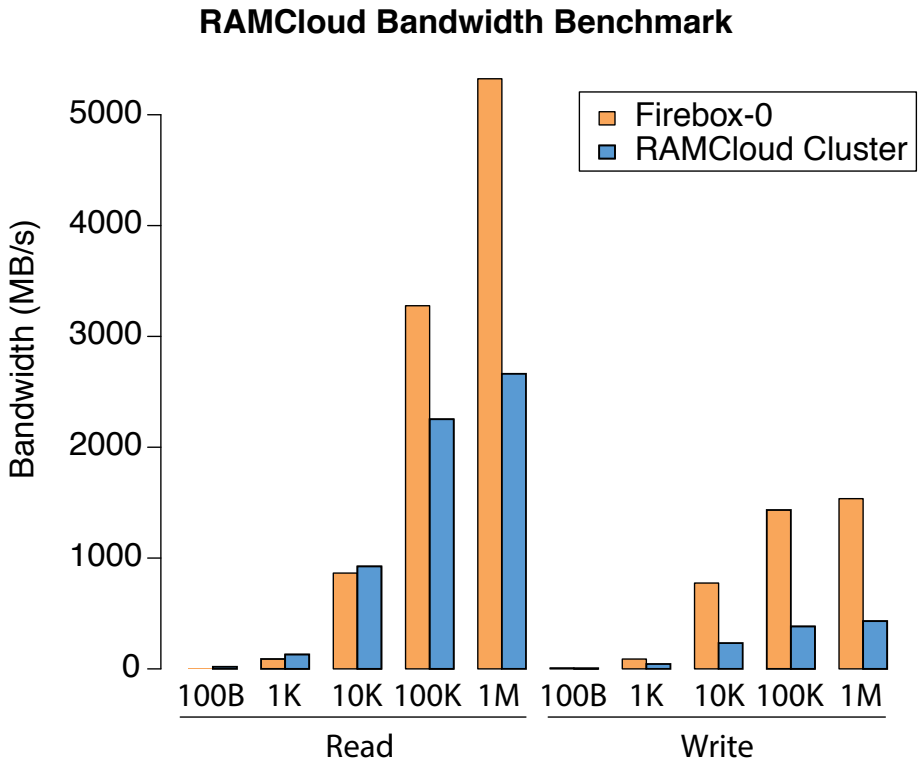
Key-value stores benchmarked:

1) RAMCloud (RAM storage system with low-latency RPCs)
2) Aerospike (flash-optimized key-value store)
3) Cassandra (semi-structured and scalable key-value store)

Name	Atomicity / Consistency	Durability / Replication	Latency
RAMCloud	Atomic single-key updates. Supports conditional updates	Data is replicated to other nodes (RAM) and written to disk asynchronously	Ultra-low
Aerospike	Atomic single-key update Supports atomic multi-key reads	Data is replicated and written to disk sync/asynchronously	Low
Cassandra	Tunable / linearizable consistency No atomic writes across replicas	Data is replicated and written to disk sync/asynchronously	Low - Medium

RAMCloud





ClusterPerf:

- RAMCloud test suite (eg., multiReads)
- We use one test (sequential read / write of random objects)

RAMCloud:

- 4 nodes, replication factor = 0

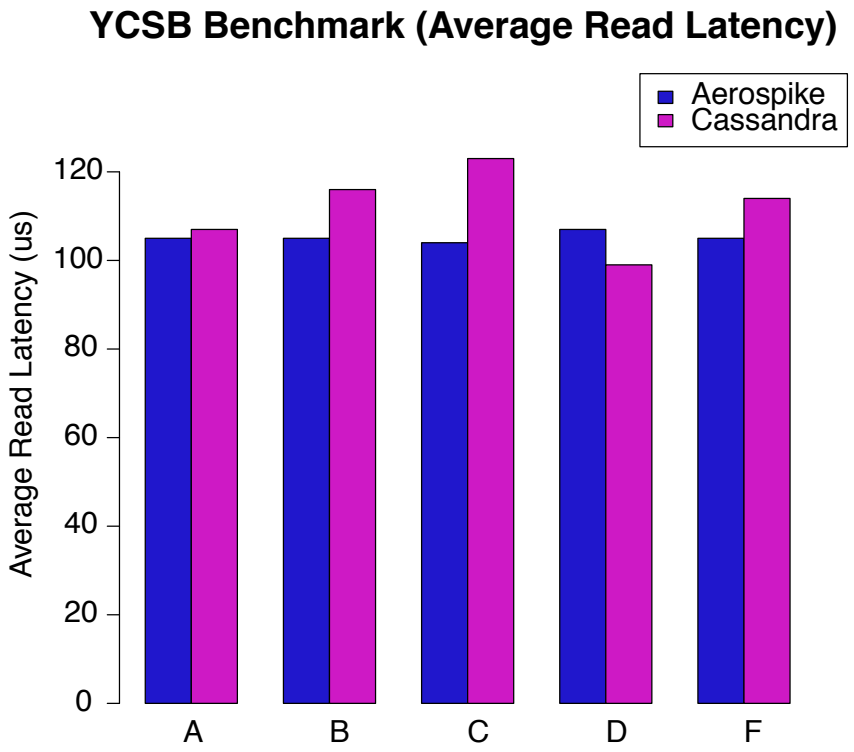
Latency / Bandwidth:

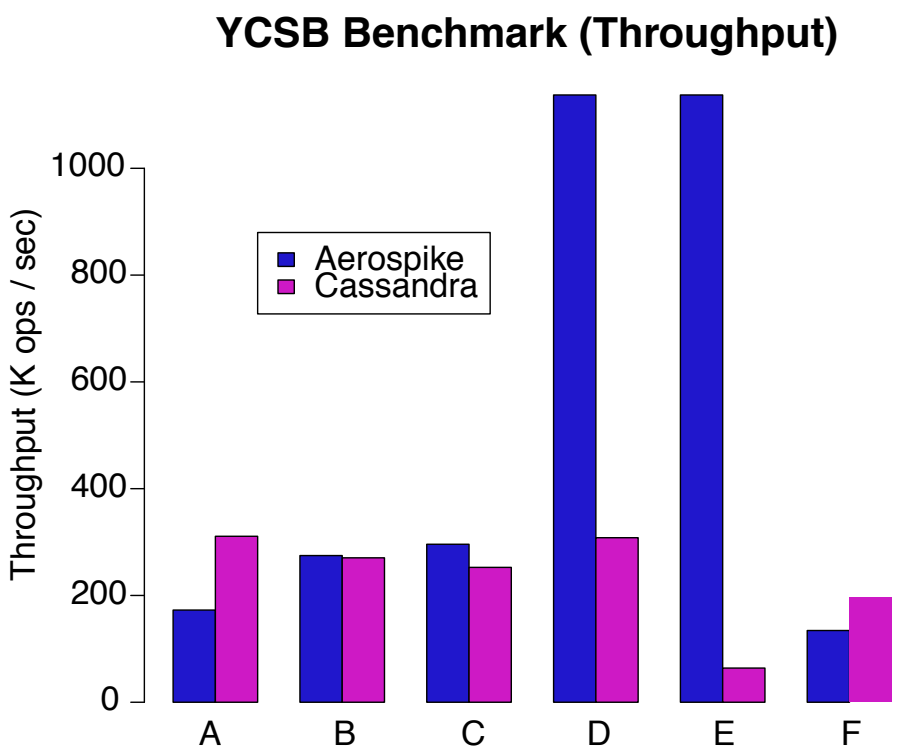
- 67-107% higher read bandwidth
- 45-53% lower read latency

Challenges:

- ACID multi-key transactions with low-latency and at scale

AeroSpike + Cassandra (YCSB)





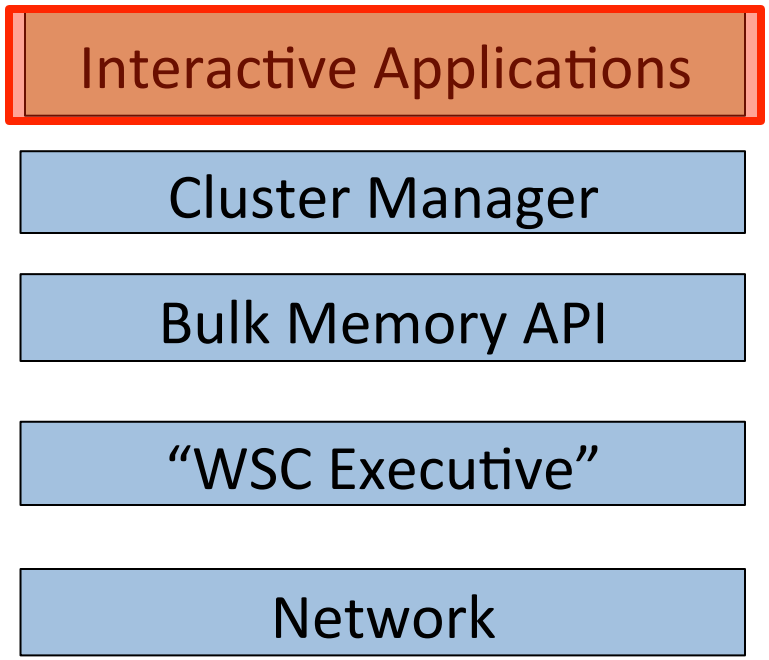
- YCSB:
- 32 concurrent clients from Firebox

- Aerospike and Cassandra setup:
- 4 nodes
- Replication factor = 3

YCSB results:
- Sub-millisecond average read latency for both Aerospike and Cassandra
- Aerospike with higher throughput for Workload D (“read latest”) and E (“short ranges”)

Interactive Applications on Firebox-0

Apache Solr

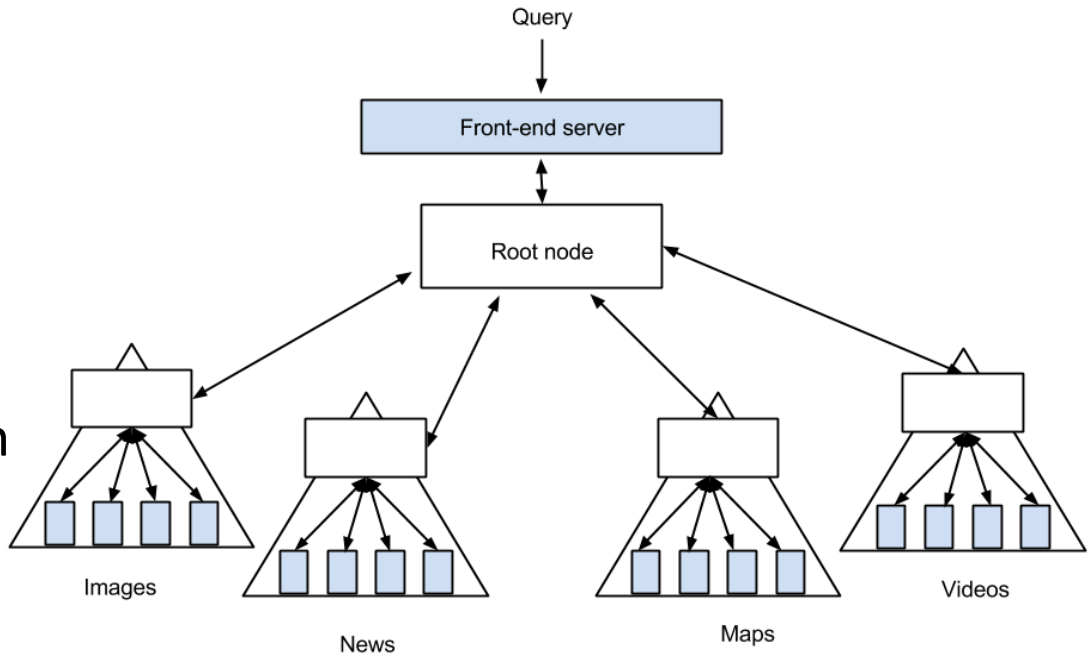


Benchmarking Distributed Search: Apache Solr on Firebox-0

Goal: Test latency sensitive (interactive) applications on Firebox-0

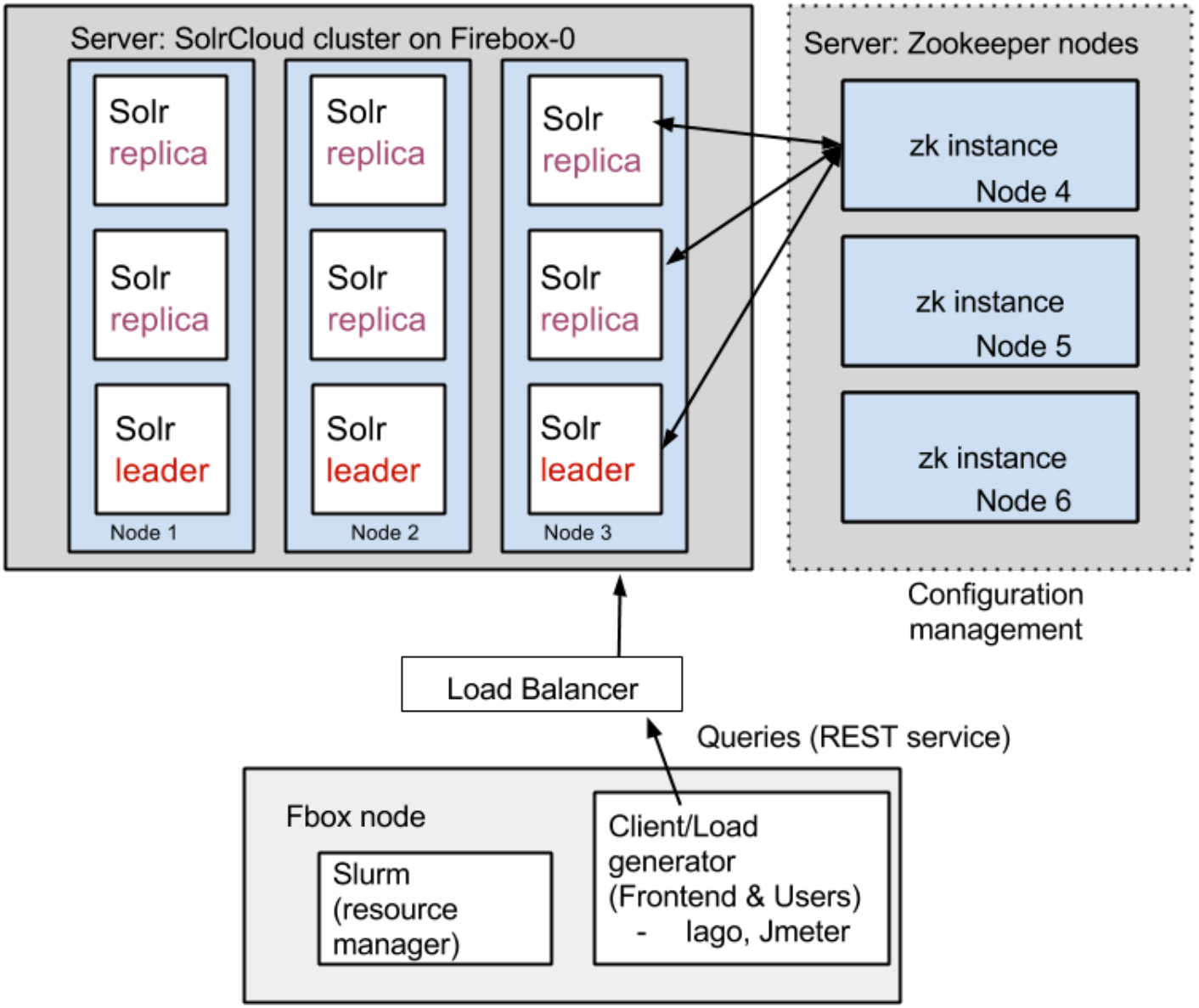
Apache Solr is a popular open source program for text search with widespread use

Distributed search architecture



Fan-out pattern -- many machines piece together the search response for a user

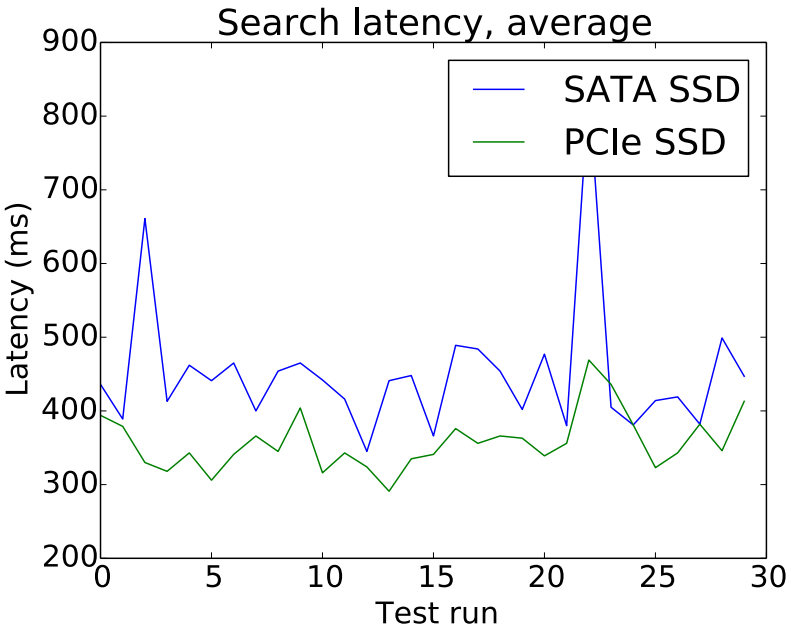
Solr on Firebox-0

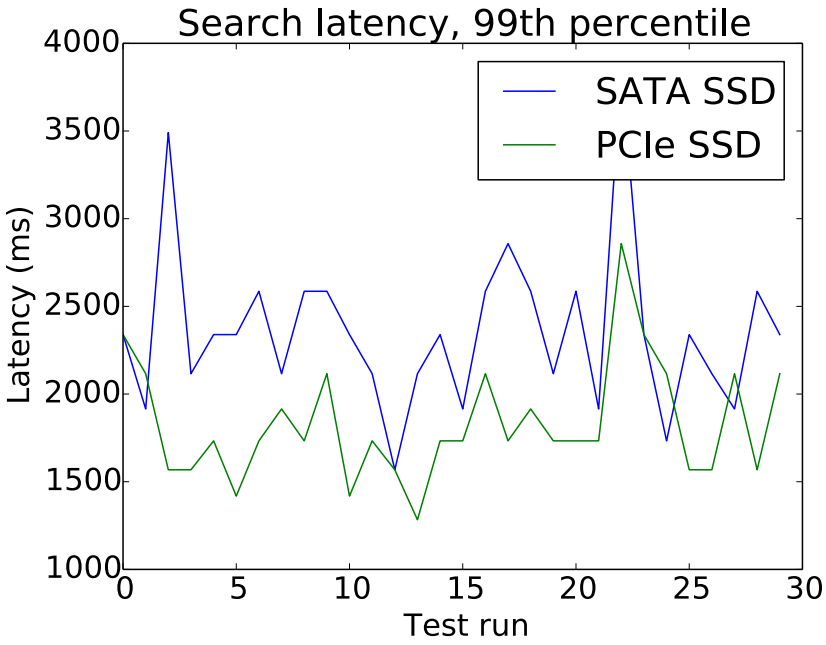


Experiment: Serving a large search index off of disk

Samsung 843T MZ7WD480 SATA vs. Samsung XS1715 PCIe 3.0 SSDs

2 shards, no replicas, 50 million indexed documents, 4 GB heap max, default GC settings





PCI-Express ~25% lower latency

Challenges benchmarking large distributed systems

- Caching effects at the disk, OS, and program level have a significant impact on latency
- Java GC is unpredictable - See Martin's talk
- many permutations of parameters to choose from for any given test
- exogeneous factors that disrupt performance hard to find and diagnose

Testing Framework (BITS) - Berkeley Interactive and Throughput Suite

Berkeley Interactive Throughput Suite (BITS):
a benchmarking suite for datacenter-scale computing, currently in development.

Goal: test batch and interactive (latency-sensitive) workloads.

4 stages: setup, data generation, execute, teardown.

Development: Written in Python, it also makes use of popular, open-source tools for tasks such as load generation and performance monitoring

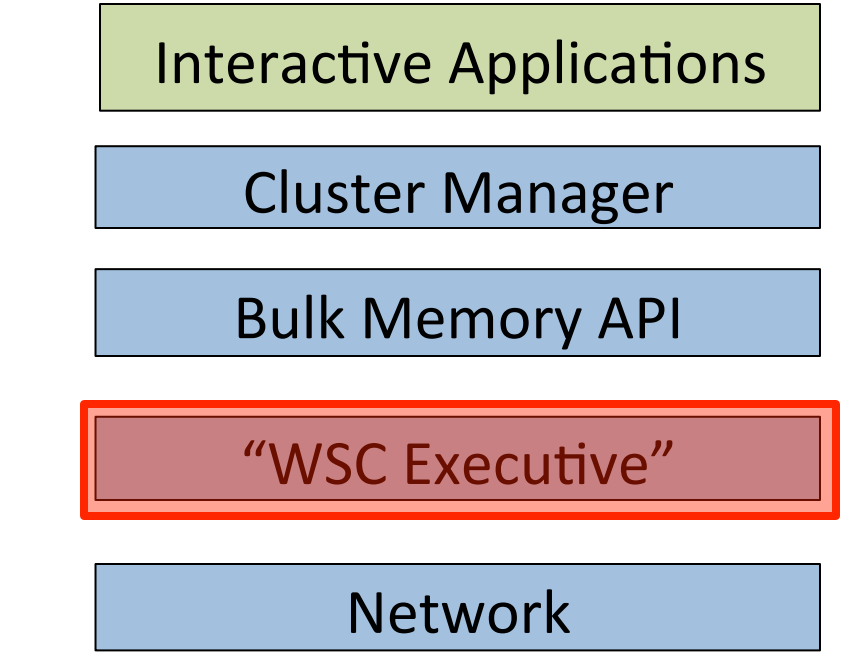
Future Work

- **Benchmarks / Workloads:**
  - A number of genomics and batch workloads are running on EC2, need to run on Firebox-0
  - Currently no benchmark at the "WSC Executive" layer
  - What other workloads to look at?

- **Aerospike vs Cassandra results:**
  - Understand Aerospike / Cassandra performance on the D and E workloads (YCSB)

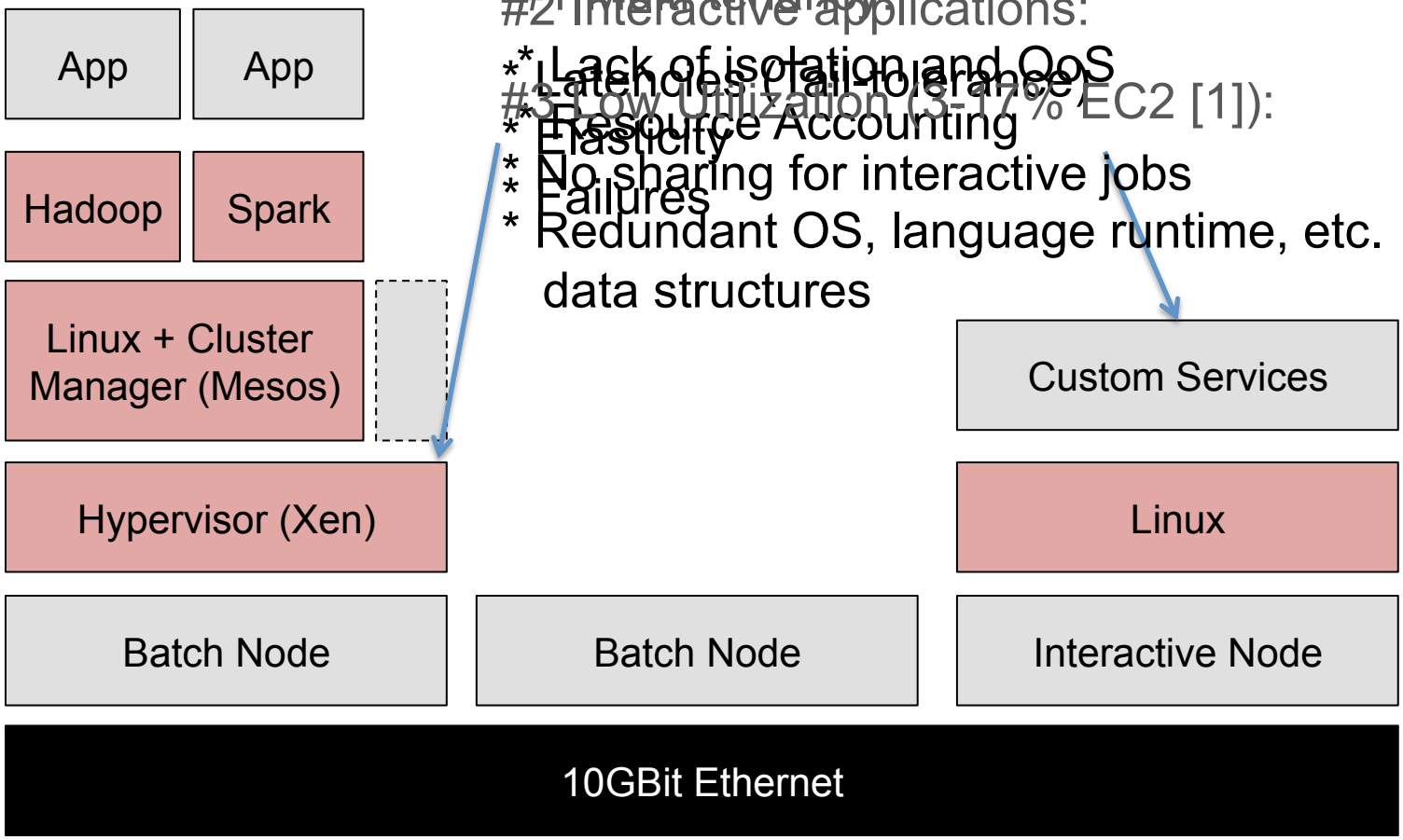
- **BITS:**
  - Effort just getting started
  - There are other benchmarks / frameworks we can leverage

Outline



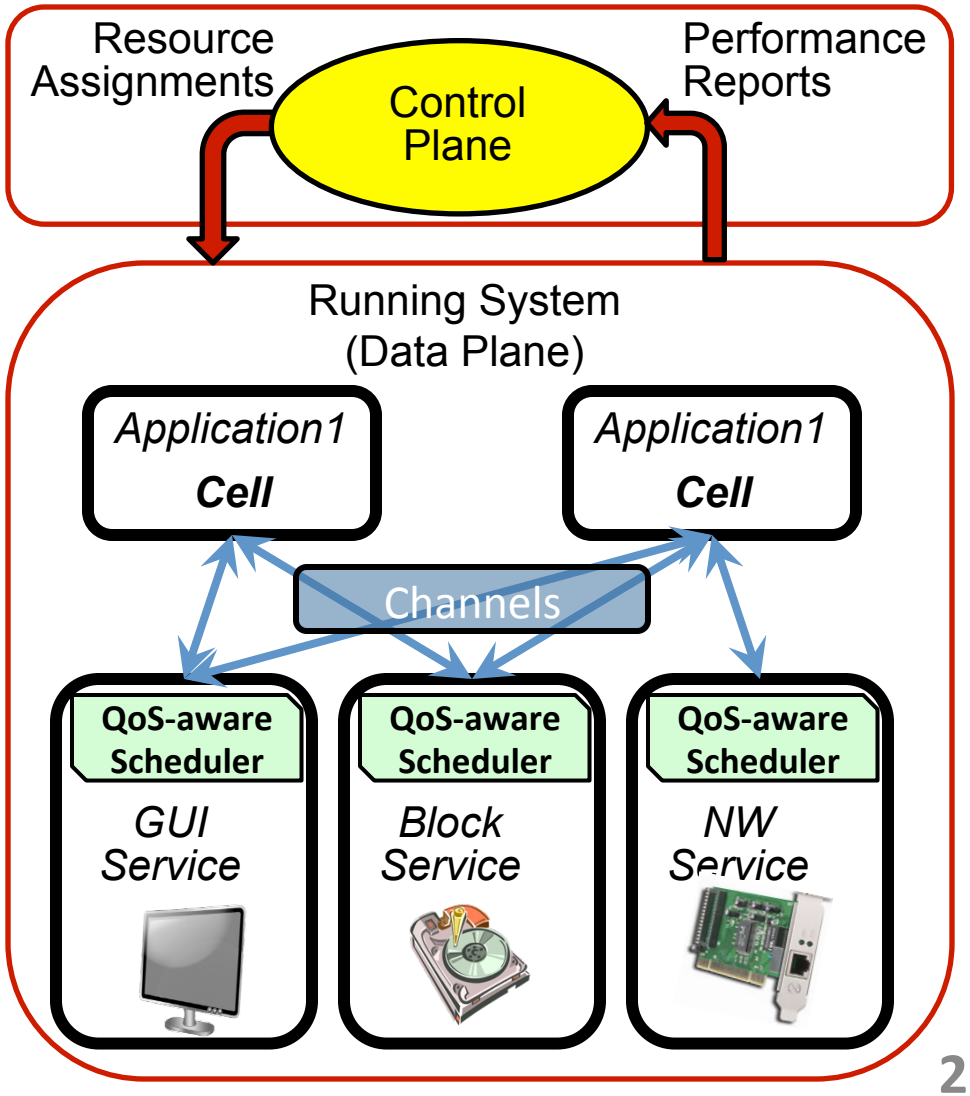
XARCC

Motivation


App
App
Hadoop
Spark
Linux + Cluster Manager (Mesos)
Hypervisor (Xen)
Batch Node
Batch Node
Custom Services
Linux
Interactive Node
10GBit Ethernet
#1 Multi-tenancy:
* Lack of isolation and QoS
* Resource Accounting
* No sharing for interactive jobs
* Redundant OS, language runtime, etc.
data structures
#2 Interactive applications:
* Latencies (tail-tolerance)
* Elasticity
* Failures
#3 Low Utilization (3-17% EC2 [1]):

Prior ParLab Work: Tessellation

- Adaptive Resource Centric Computing (ARCC)
  - Package applications into *cells* with explicit resource allocation and QoS guarantees
  - Manage cells and resource allocation through a thin control plane
- Service-oriented
  - Cells subscribe to services provided by other cells
  - Service subscriptions == resource allocations



XARCC
(aka Tessellation 2.0)

Observation: If the Tessellation kernel looks like a hypervisor, why not just use a hypervisor?

We chose Xen:

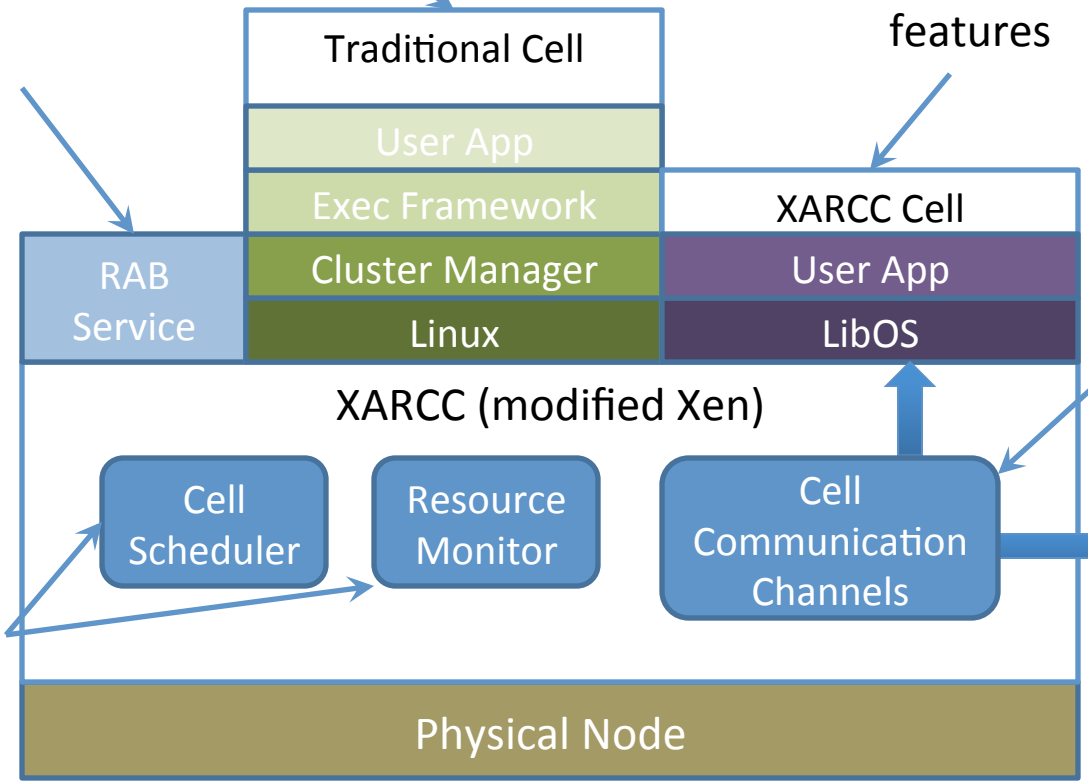
"**X**en with **A**daptive **R**esource **C**entric **C**omputing"

XARCC

Traditional virtual-machine approaches continue to work

Resource allocation broker provides QoS and ARCC policy

Most cells run modified OS and applications to take full advantage of XARCC features



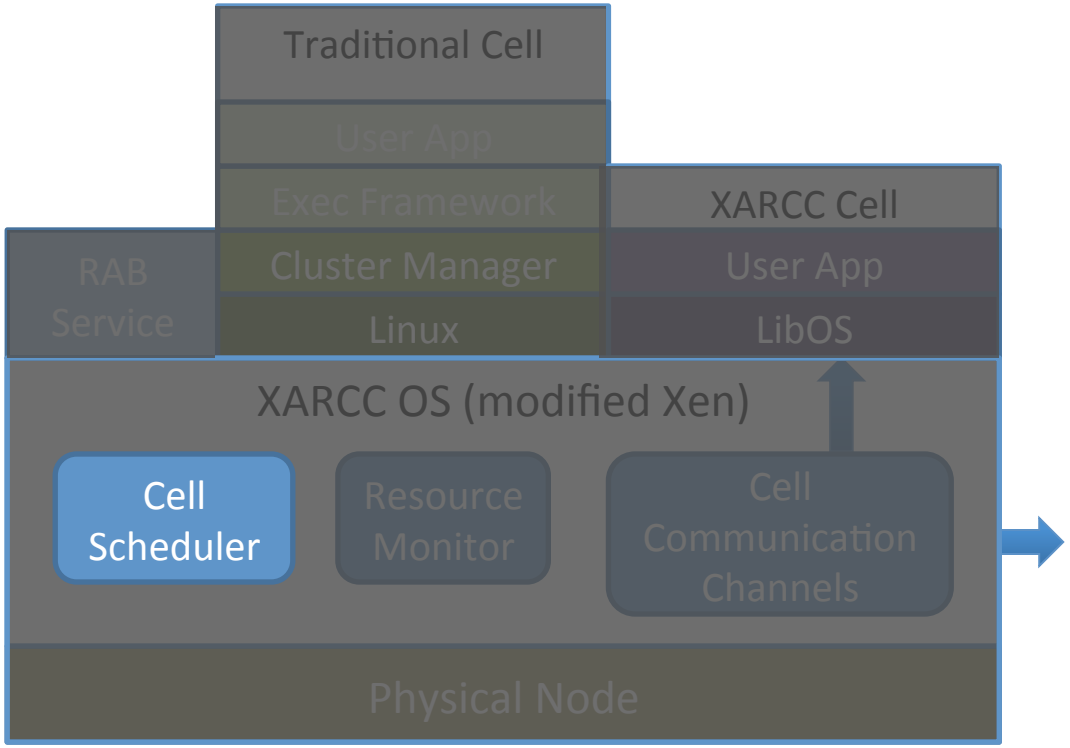
Unified Inter-node and Intra-node IPC interfaces

XARCC kernel Provides QoS and ARCC mechanisms

XARCC

Scheduling: Status and Experiences

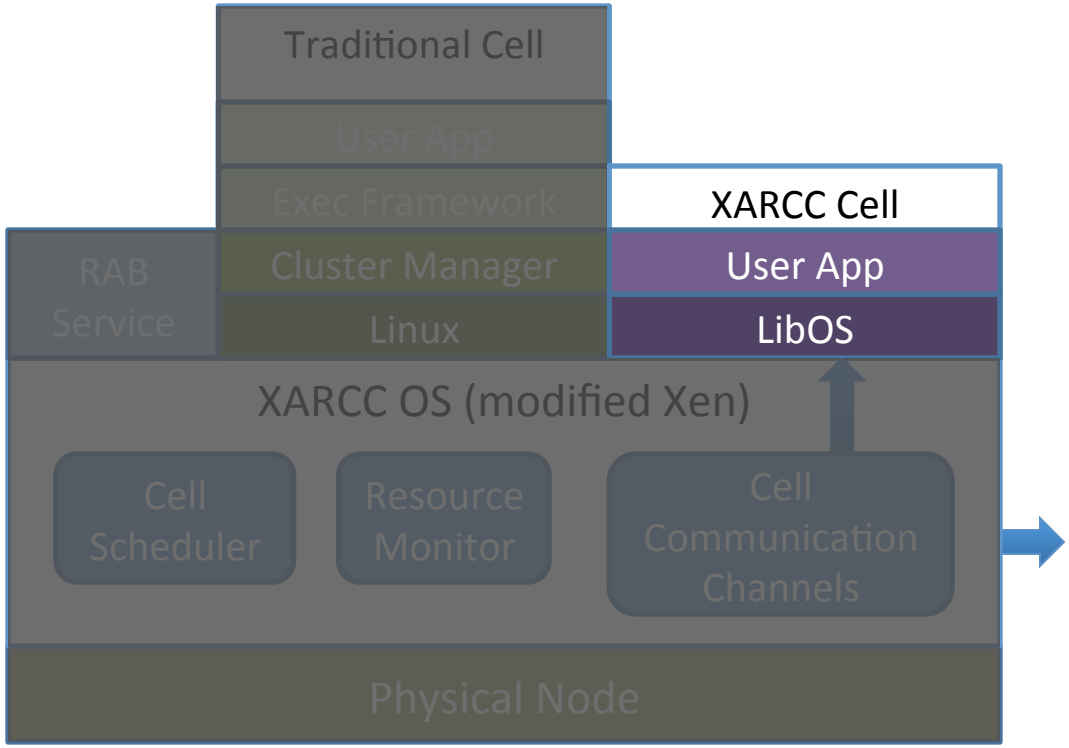
- Status: Implemented gang scheduler on Xen
  - Target performance predictability and synchronization overheads
  - Benchmarked with Micro-benchmark and HPC-oriented solid deformation simulation
- Experiences
  - Micro-benchmark shows reduced variability when gang scheduling
  - Macro-benchmarks suffer from reduced CPU utilization
  - Takeaway: Tradeoff between utilization and predictability



XARCC

Library OS: Status and Experiences

- Status: Implemented stripped down OS for cells (CellOS)
  - Based on Guk[1] and newLib[2]
  - Supports a subset of POSIX
  - Ported Pthread based benchmarks
- Experiences
  - Some single-application benchmarks are slower on CellOS than on Linux
  - Debugging is very difficult
  - Need to understand pros/ cons better



Future Work

- Still early days: Most features yet to be implemented
  - Resource Monitor
  - Dynamic Allocation Policies
  - Communication channels and service discovery

- Run on FireBox-0 with BITS
  - Can current workloads take advantage?
  - Would custom HW change things?

- Big area of interest: 2-level scheduling and resource allocation
  - Gang scheduling needs deeper analysis and more sophisticated implementation
  - Advantages of minimal OS vs Linux

Questions & Feedback

- This work is in the early stages
- Where would you like to see this go?
  - FireBox-0
  - BITS
  - XARCC