Distributed NoSQL Storage for Extreme-scale System Services

Tonglin Li

ILLINOIS INSTITUTE OF TECHNOLOGY

Short Bio

- 6th year PhD student from DataSys Lab, CS Department, Illinois Institute of Technology, Chicago
- Academic advisor: Dr. Ioan Raicu
- Research internships
  - Summer 2014, Argonne National Lab, Lemont, IL
  - Summer 2013, Lawrence Berkeley National Lab, Berkeley, CA
- Research interests:
  - Distributed systems: NoSQL storage systems, file systems, system services, HPC, Clouds and Big data

Collaborators

- **Dr. Dongfang Zhao** (PNNL)
- **Dr. Ke Wang** (Intel)
- **Dr. Kate Keahey** (ANL/MCS)
- **Dr. Lavanya Ramakrishnan** (LBL/ACS)
- **Dr. Zhao Zhang** (UC Berkeley/AMP Lab)

Outline

- Background and motivation
- ZHT: distributed key-value store for high-end computing systems
  - Proposed work: overview
  - System Design and Implementation
  - Features
  - Evaluation
- Application use cases
- Conclusion and Future work

World's most powerful machines

RANK	SITE	SYSTEM	CORES	RMAX (TFLOP/S)	RPEAK (TFLOP/S)	POWER (KW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945

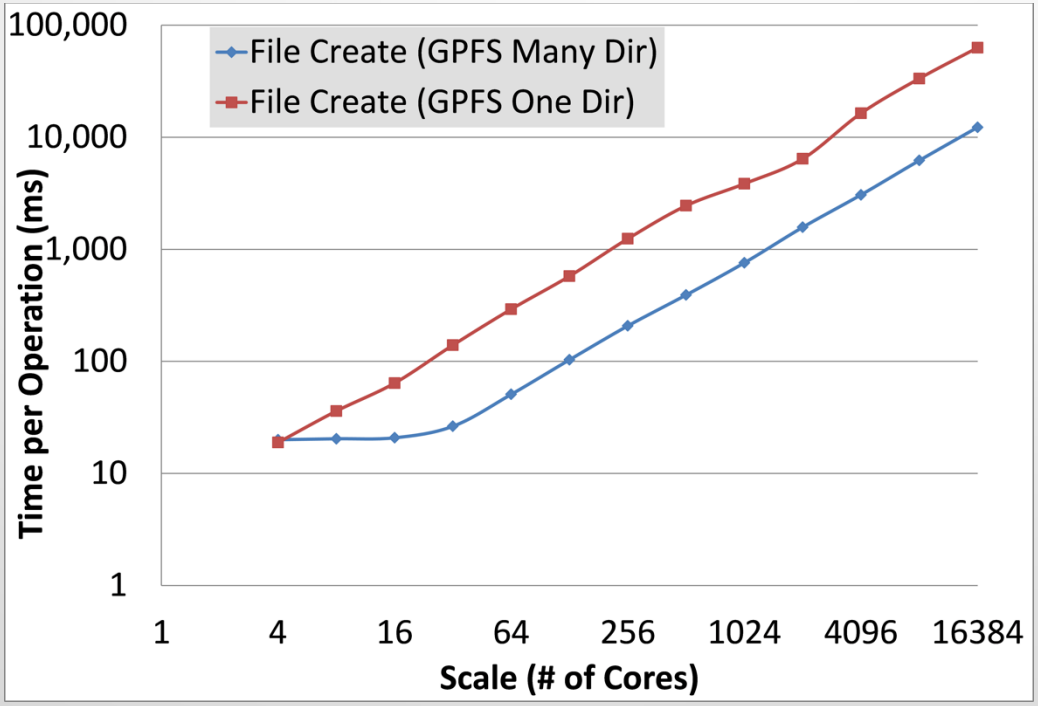
Ways to better performance: Scale up

- Scale up: build ONE bigger machine
  - Faster CPUs, larger memory, make everything faster
  - Exponentially expensive
  - Can’t go too big

Ways to better performance: Scale out

- Scale out: build a bigger GROUP of machines
  - Up-to-date and commodity hard ware, fast network
  - Biggest issue: scalability: 1+1 < 2
    - Workload: Amdahl's law, Gustafson's law
    - System services: metadata, scheduling, controlling, monitoring

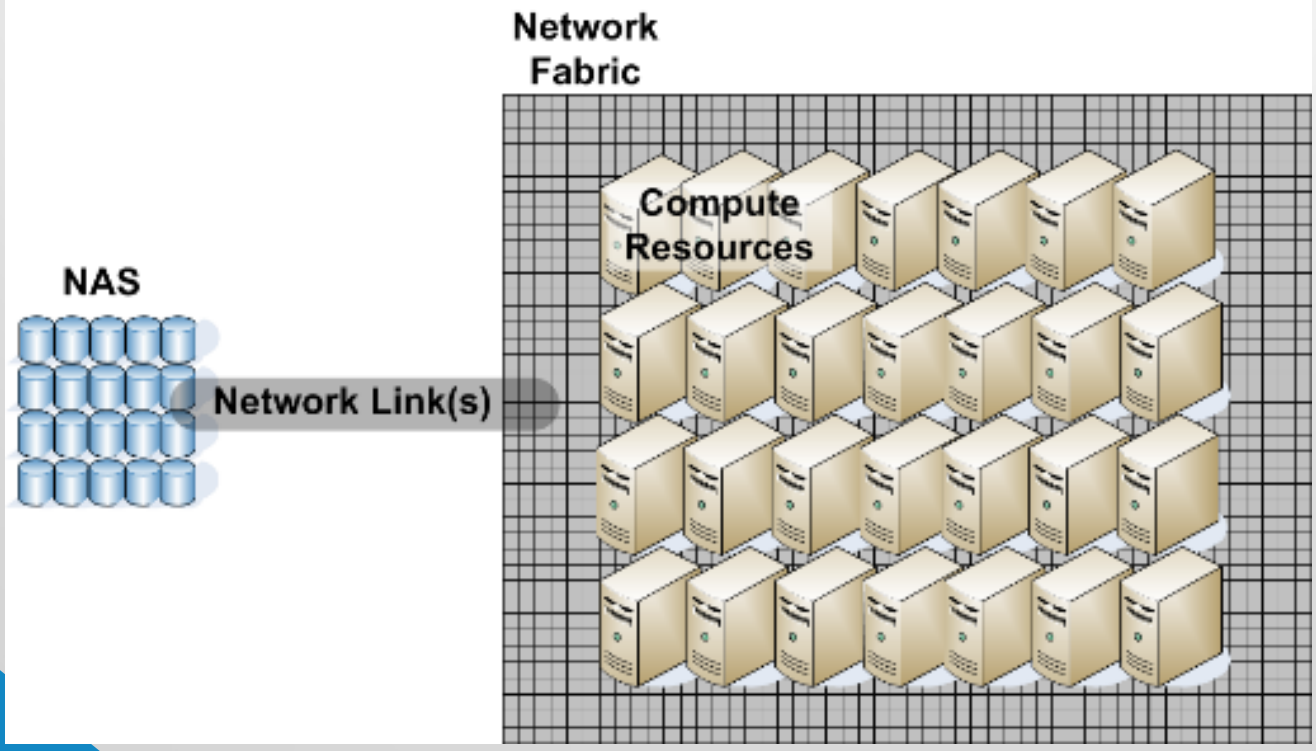
System services: bad example



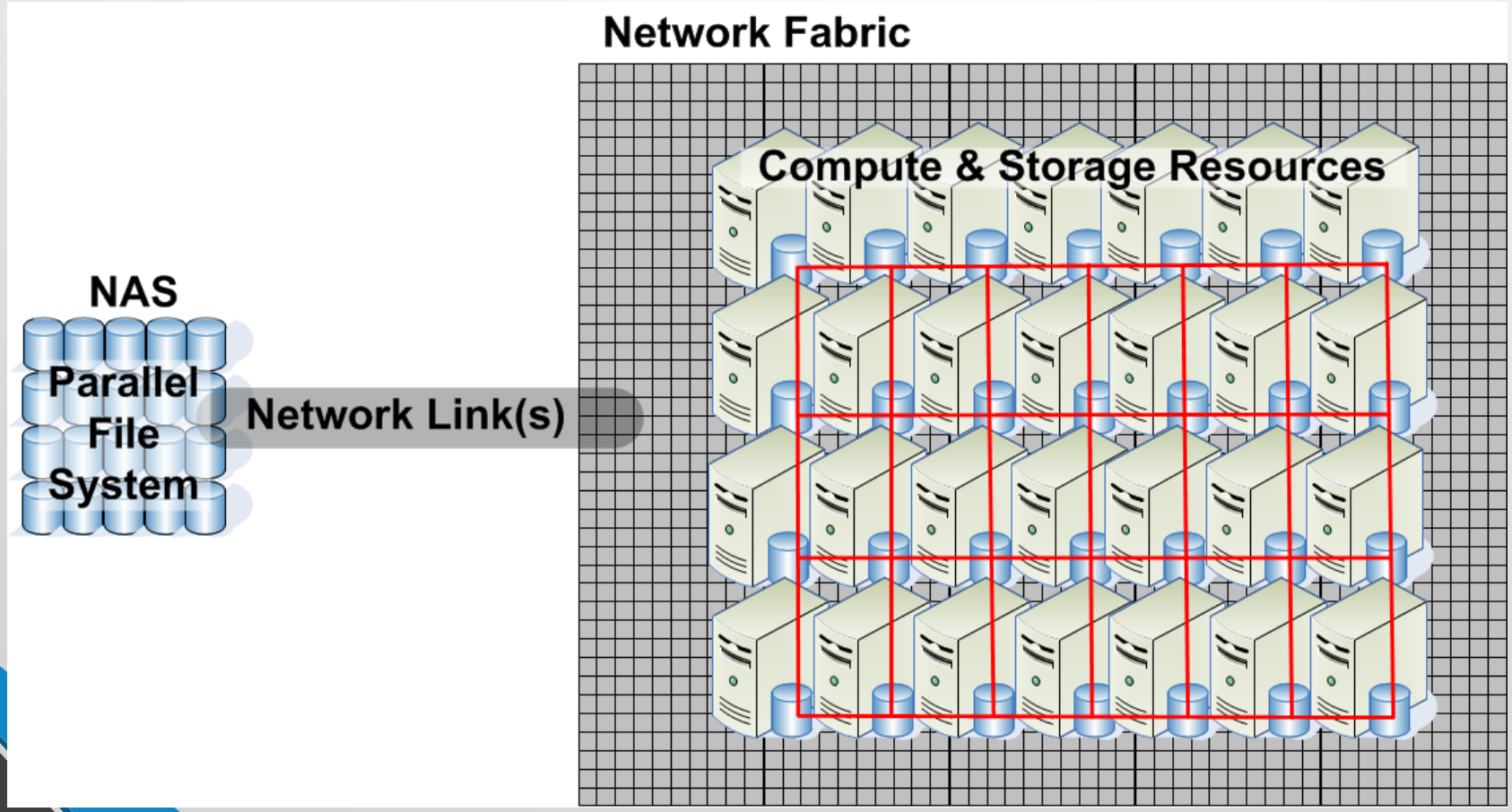
IBM GPFS parallel file system metadata performance
(BlueGene/P, Intrepid, Argonne)

Background: storage problem

- Limitations of current architecture:
  - Separated Compute and storage: shared network infrastructure
  - All I/Os are (eventually) remote
  - Frequent checkpointing: extremely write-intensive



New architecture: distributed storage



System services are critical!

- Metadata management (file systems, databases…)
- System scheduling systems (job, I/O…)
- System state monitoring system
- …

Problem statement and motivation

- Scalability is limited by system management and services.
- Decades old relatively centralized architecture no longer catches up the growth of system scale.
- There is no proper storage system to support large scale distributed system services.

Proposed work

- Build a scalable storage system that meets the needs of scalable system services for exa-scale machines
  - High Performance
  - High Scalability
  - Fault tolerance

Various storage system solutions

Storage type	Capacity	Single unit size limit	Latency	Scalability	Resilience	Query & indexing
File systems	Very large	Large	O(10) ms	Low	Medium	No
SQL Databases	Large	Various - small	O(10) ms	Very low	Very high (ACID)	Best
NoSQL data stores	Large	Various - small	O(0.1~1) ms	High	High	Good

Limitations of Current NoSQL DBs

- Performance
  - High latency: 10ms ~ seconds
- Not good enough scalability
  - Logarithmic routing algorithms
  - No deployment of O(1000) nodes
- Portability
  - Many are implemented in Java – no support on supercomputers
  - Complex dependencies

Proposed NoSQL solution

- ZHT (zero-hop distributed hash table)
  - A fast and **lightweight** distributed key-value store
  - A building block for **HPC** systems and **clouds**
- High Performance
  - Low Latency
  - High Throughput
- Scalability towards O(10K) nodes
- Reliability across failures

Lead author peer-reviewed publications

- 1 Journal paper
  - A Convergence of Distributed Key-Value Storage in Cloud Computing and Supercomputing, **Journal of Concurrency and Computation Practice and Experience (CCPE) 2015**
- 5 Conference papers
  - A Flexible QoS Fortified Distributed Key-Value Storage System for the Cloud, **IEEE Big Data 2015**
  - Distributed NoSQL Storage for Extreme-Scale System Services, **SC 15, PhD showcase**
  - A Dynamically Scalable Cloud Data infrastructure for Sensor Networks, **ScienceCloud 2015**
  - Scalable State Management for Scientific Applications in the Cloud, **IEEE BigData 2014**
  - ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table, **IEEE IPDPS 2013**
- 3 research posters
  - GRAPH/Z: A Key-Value Store Based Scalable Graph Processing System, **IEEE Cluster 2015**
  - A Cloud-based Interactive Data Infrastructure for Sensor Networks, **IEEE/ACM SC 2014**
  - Exploring Distributed Hash Tables in High-End Computing, **ACM SIGMETRIC PER 2011**

Co-author peer-reviewed Publications

- 2 Journal papers
  - Load-balanced and locality-aware scheduling for data-intensive workloads at extreme scales, **Journal of Concurrency and Computation: Practice and Experience (CCPE), 2015**
  - Understanding the Performance and Potential of Cloud Computing for Scientific Applications, **IEEE Transaction on Cloud Computing (TCC), 2015**
- 4 Conference papers
  - Overcoming Hadoop scaling limitations through distributed task execution, **IEEE Cluster 2015**
  - FaBRiQ: Leveraging Distributed Hash Tables towards Distributed Publish-Subscribe Message Queues, **IEEE/ACM BDC 2015**
  - FusionFS: Towards Supporting Data-Intensive Scientific Applications on Extreme-Scale High-Performance Computing Systems, **IEEE Big Data 2014**
  - Optimizing Load Balancing and Data-Locality with Data-aware Scheduling, **IEEE Big Data 2014**

Technical reports and other posters

- Distributed Kev-Value Store on HPC and Cloud Systems, GCASR 2013
- NoVoHT: a Lightweight Dynamic Persistent NoSQL Key/Value Store, GCASR 2013
- FusionFS: a distributed file system for large scale data-intensive computing, GCASR 2013
- OHT: Hierarchical Distributed Hash Tables, IIT 2013
- Exploring Eventual Consistency Support in ZHT, IIT 2013
- Understanding the Cost of Cloud Computing and Storage, GCASR 2012
- ZHT: a Zero-hop DHT for High-End Computing Environment, GCASR 2012

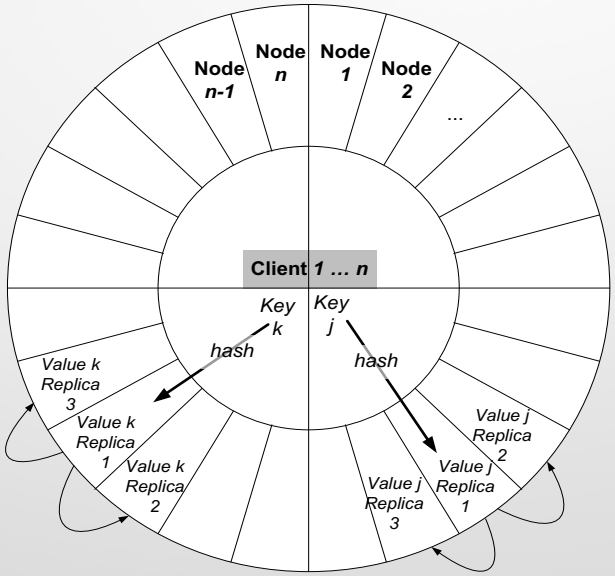
Overview: ZHT highlights

- ZHT: a zero-hop key-value store system
  - Written in C/C++, few dependencies
  - Tuned for supercomputers and clouds
- Performance Highlights (on BG/P)
  - Scale: 8K-nodes and 32K instances
  - Latency: 1.5 ms at 32K-core scales
  - Throughput: 18M ops/s

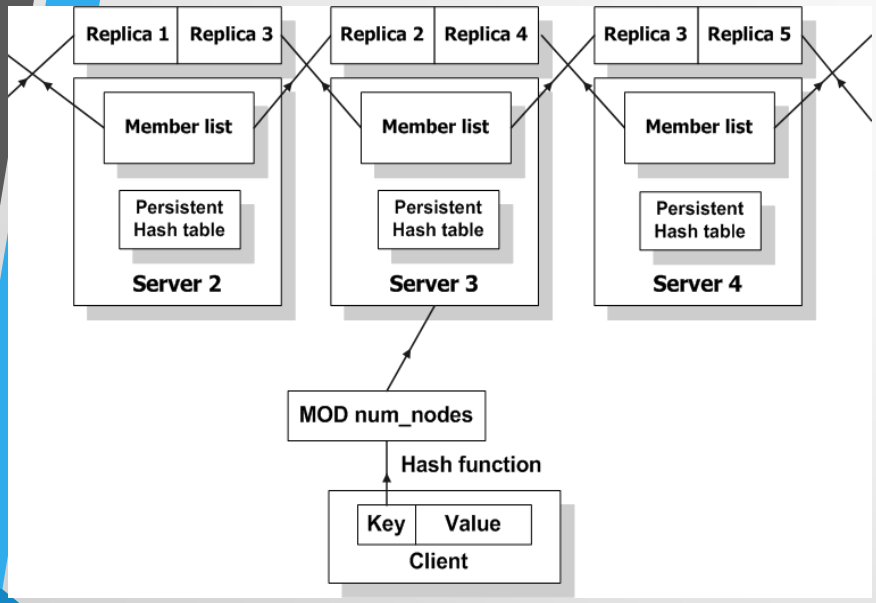
ZHT Features

- Unconventional operations
  - Insert, lookup, remove
  - Append, Compare_swap, Change_callback
- Dynamic membership: allowing node joins and leaves
  - modified consistent hashing
  - Constant routing: 2 hops at most
  - Bulk partition moving upon rehashing
- Fault tolerance
  - Replication
  - Strong or eventual consistency

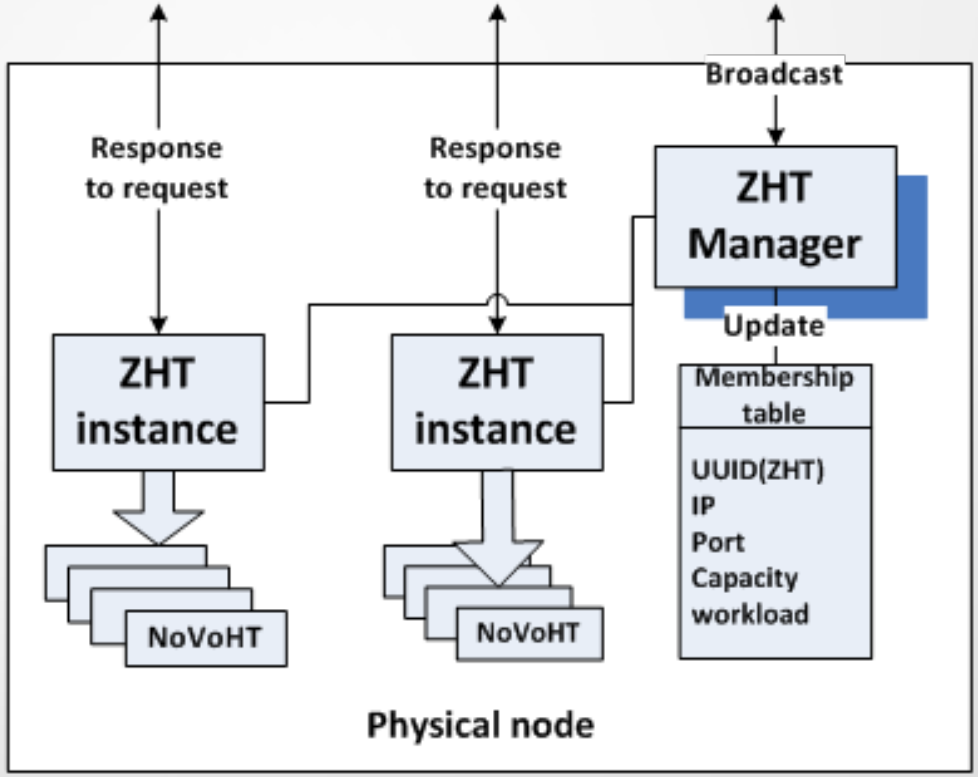
Zero-hop hash mapping



ZHT architecture



Topology



Server architecture

ZHT Related work

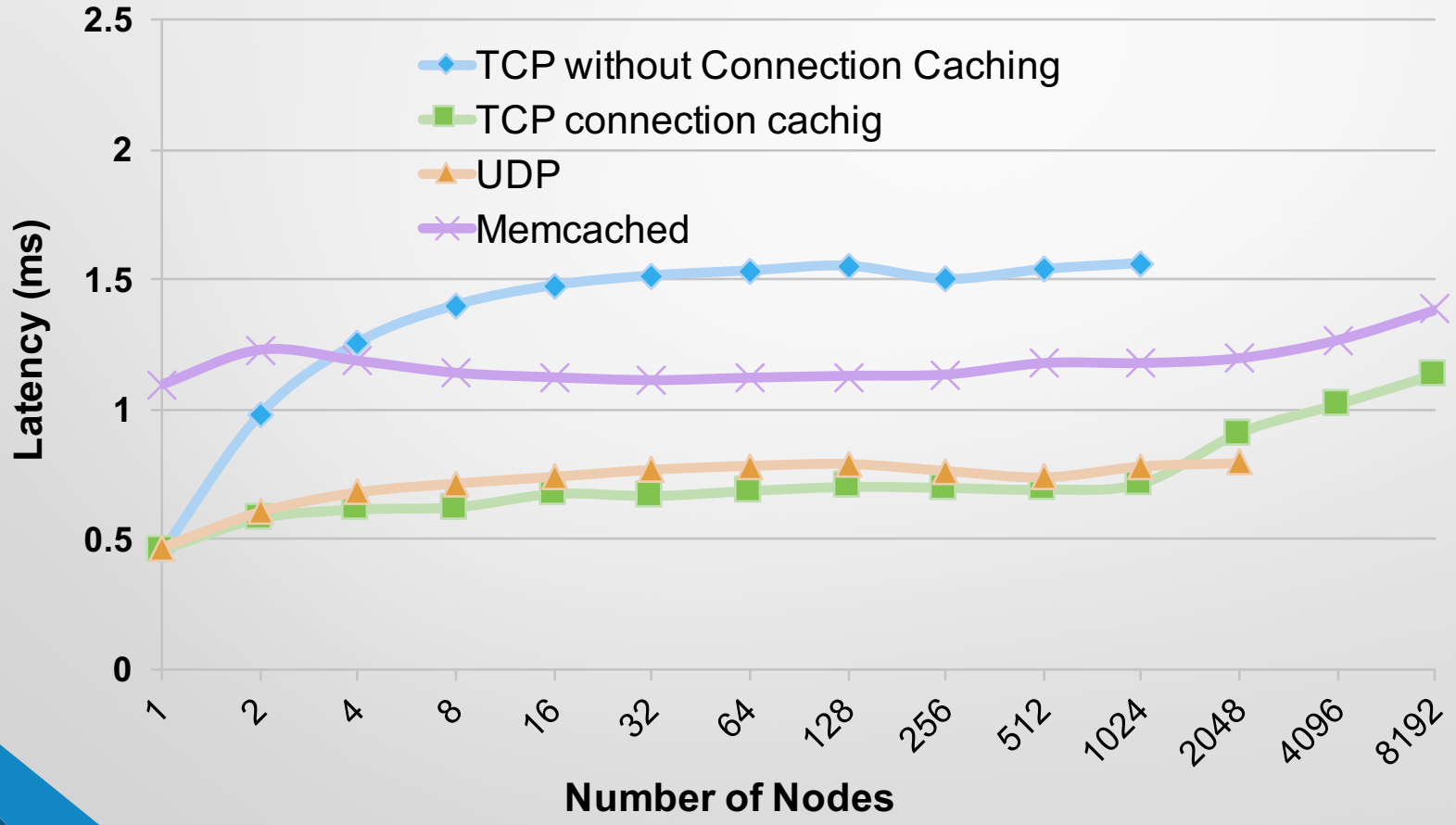
- Many DHTs: Chord, Kademlia, Pastry, Cassandra, C-MPI, Memcached, Dynamo
- Why another?

Name	Impl.	Routing Time	Persistence	Dynamic membership	Additional Operation
Cassandra	Java	Log(N)	Yes	Yes	No
C-MPI	C/MPI	Log(N)	No	No	No
Dynamo	Java	o to Log(N)	Yes	Yes	No
Memcached	C	o	No	No	No
ZHT	C++	o to 2	Yes	Yes	Yes

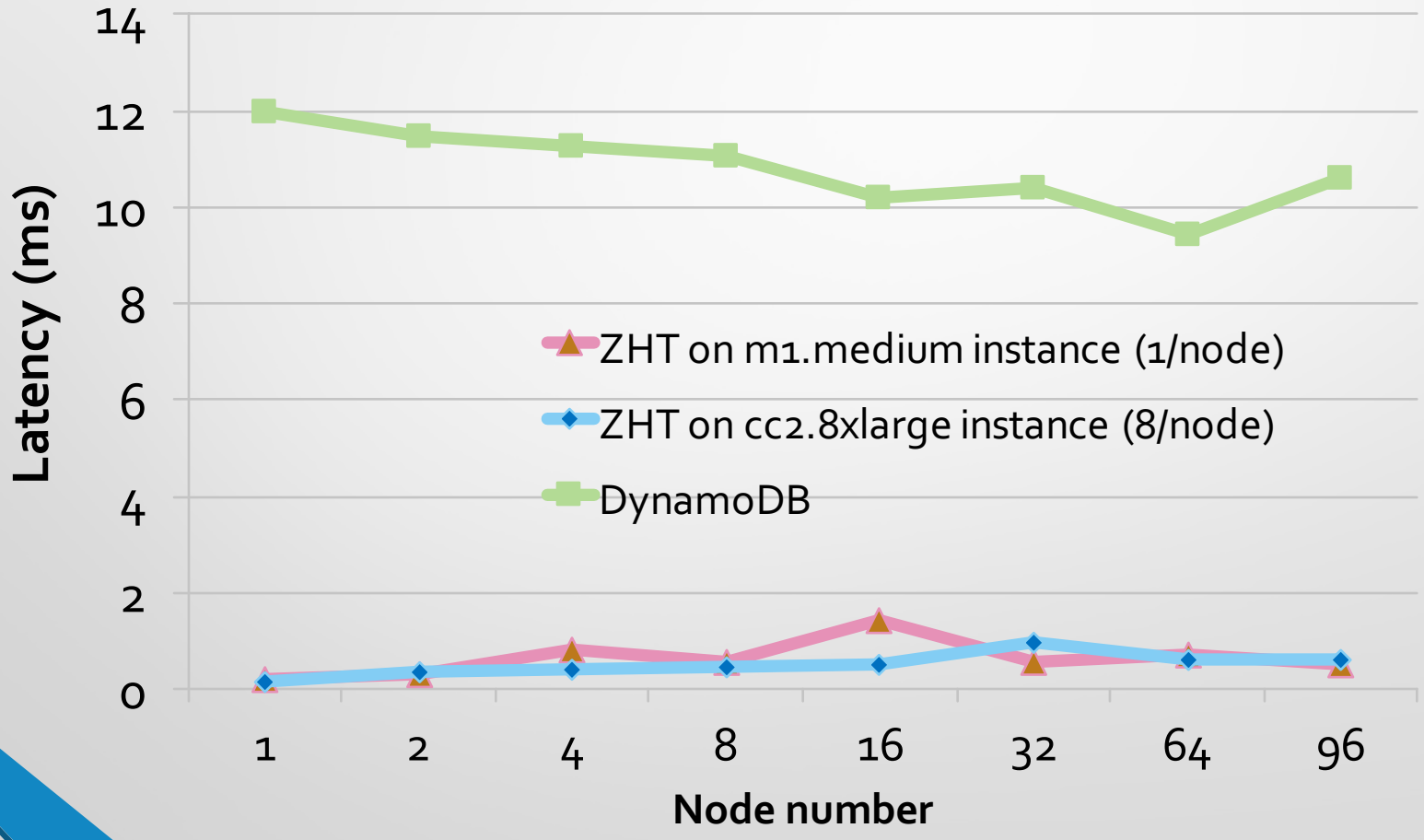
Evaluation

- Test beds
  - IBM Blue Gene/P supercomputer
    - Up to 8K-nodes and 32K-cores
    - 32K-instance deployed
  - Commodity Cluster
    - Up to 64-nodes
  - Amazon EC2
    - M1.medium and Cc2.8xlarge
    - 96 VMs, 768 ZHT instances deployed
- Systems comparison
  - Cassandra, Memcached, DynamoDB

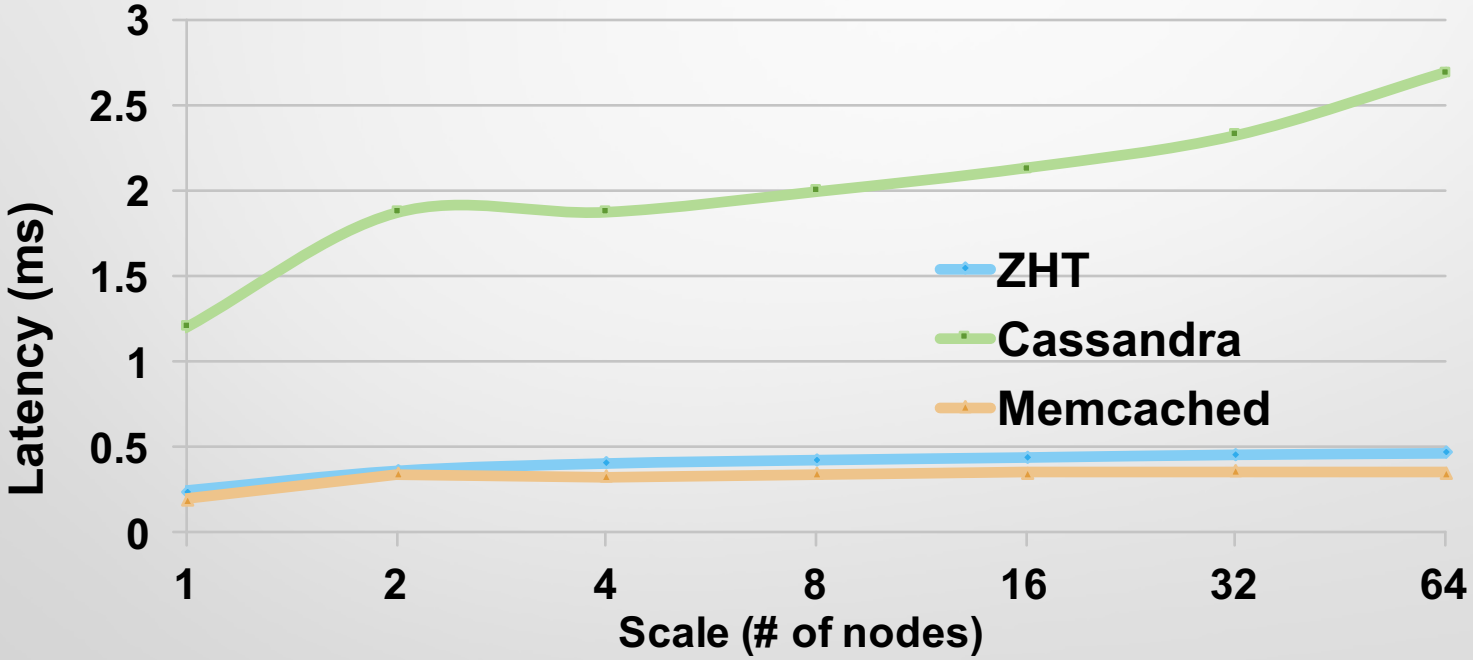
Performance on supercomputer: Intrepid, IBM BG/P



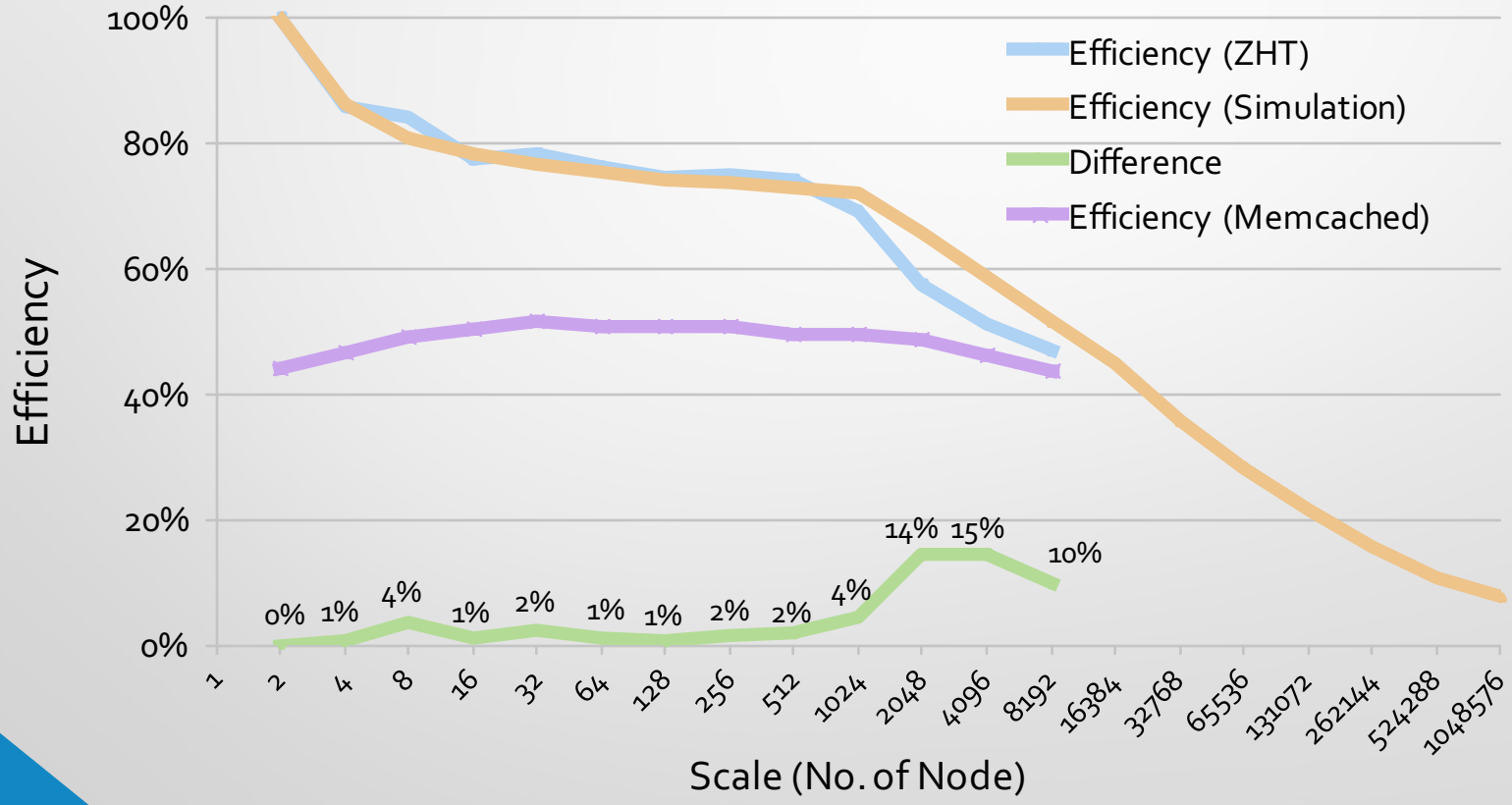
Performance on a cloud: Amazon EC2



Performance on a commodity cluster: HEC



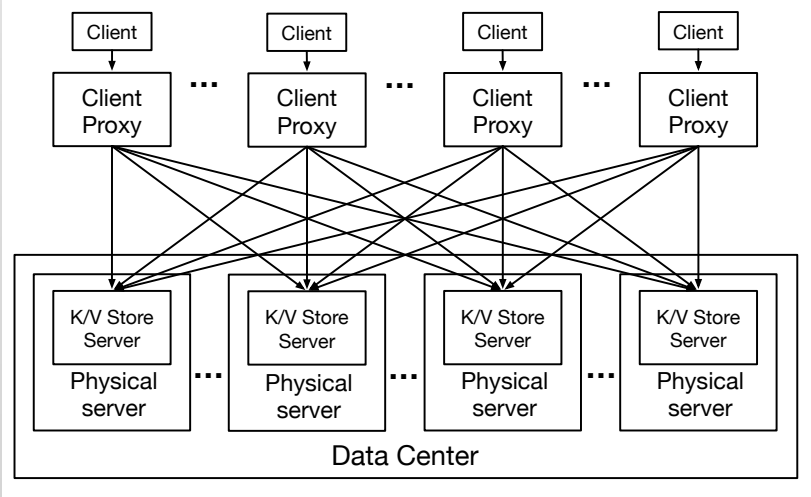
Efficiency: Simulation with 1M nodes



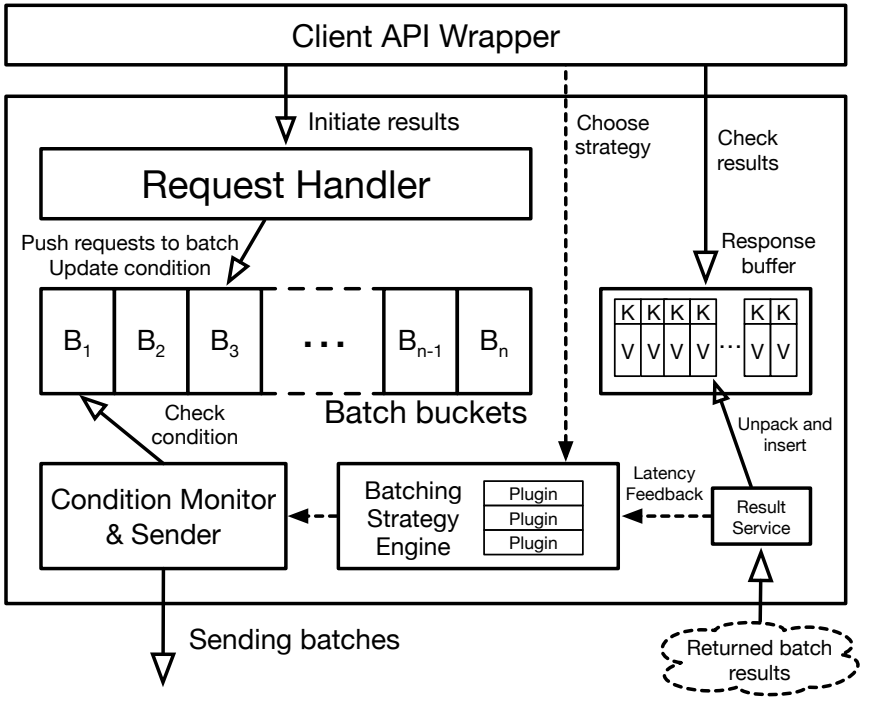
ZHT/Q: a Flexible QoS Fortified Distributed Key-Value Storage System for Data Centers and Clouds

- Built on top of ZHT
- Meet the needs of clouds and data centers
- Support simultaneous multiple applications
- QoS on request response time (latency)
- Dynamic request batching strategies
- Publication
  - A Flexible QoS Fortified Distributed Key-Value Storage System for the Cloud, **IEEE Big Data 2015**

Client side batcher



Data center network topology

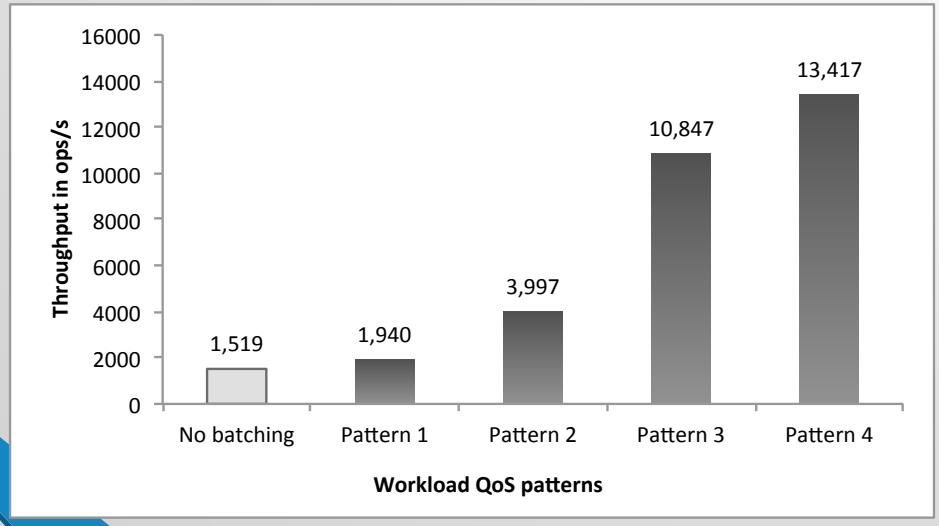


Client proxy (batcher)

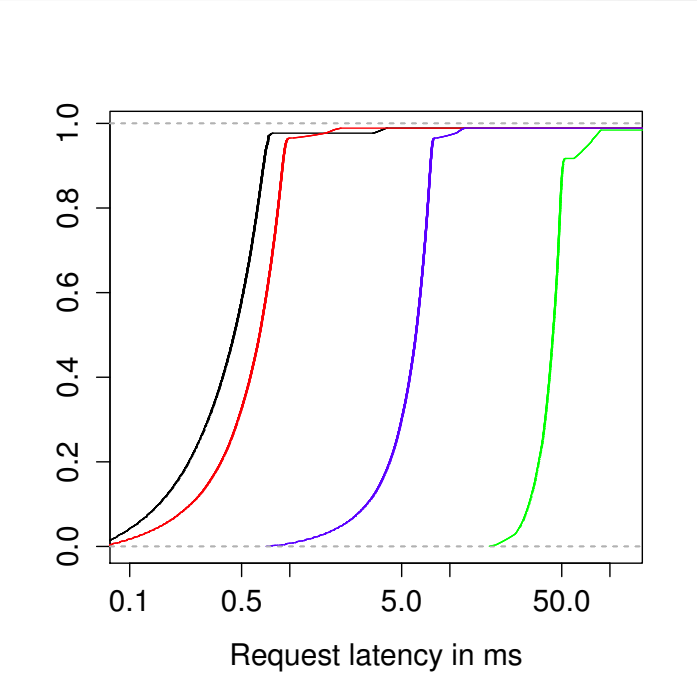
Mixed workloads with multi-QoS

Workload patterns

QoS latency time	1ms	10ms	100ms	1000ms
Pattern 1	25%	25%	25%	25%
Pattern 2	4%	32%	32%	32%
Pattern 3	0%	33%	33%	33%
Pattern 4	0%	0%	50%	50%



Throughput



Batch latency distribution

Application Use Cases

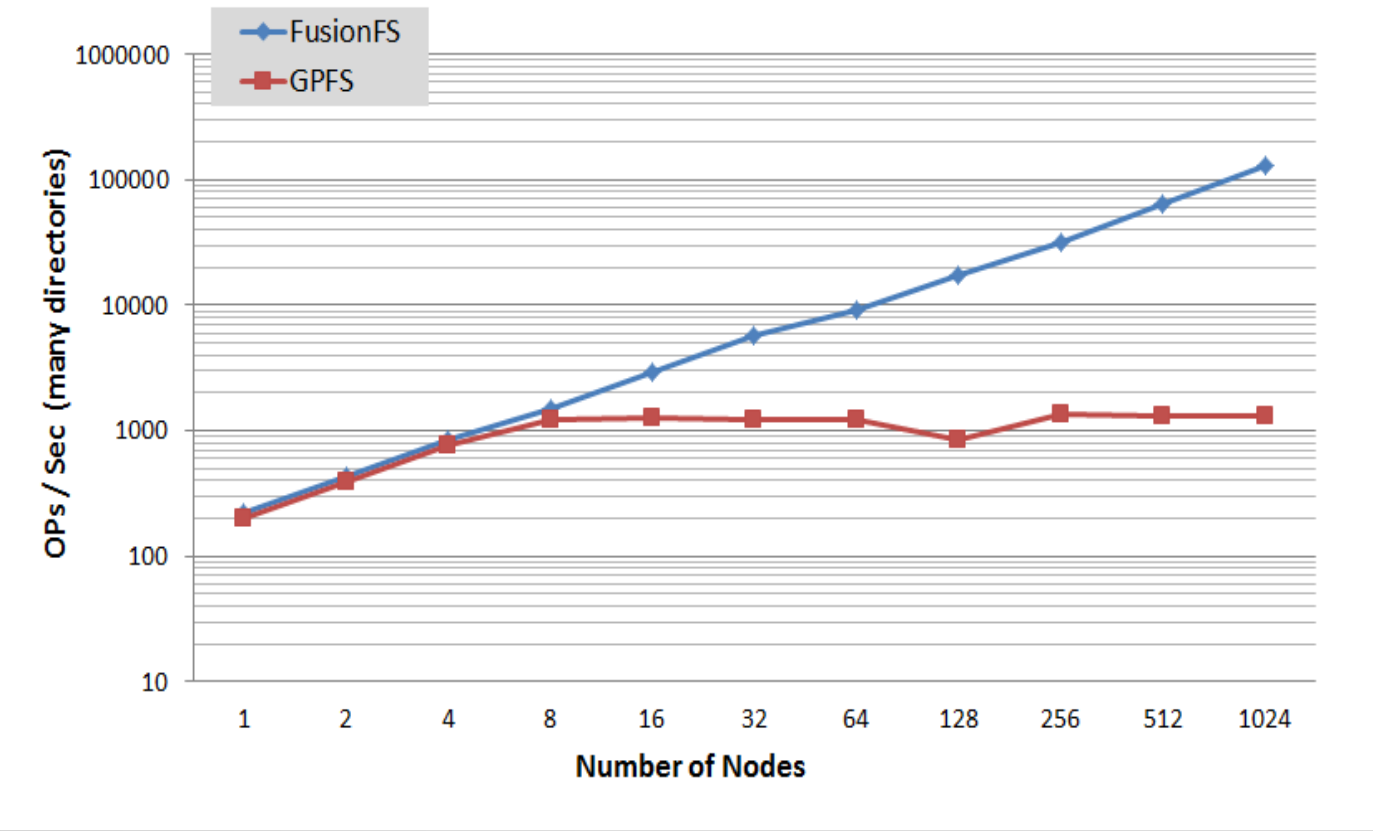
- File systems: FusionFS [TPDS15, BigData14, CCGrid14, SC13, Cluster13, TaPP13, SC12, LSAP11]
- Sensor network storage: WaggleDB [SC14, ScienceCloud15]
- Statement management: FREIDA-State [BigData14]
- Graph processing system: GRAPH/Z [Cluster15]
- MTC scheduling: MATRIX [BigData14, HPC13, IIT13, SC12]
- HPC scheduling: Slurm++ [HPDC15, HPDC14]
- Distributed message queues: FabriQ [BDC15]
- Simulations: DKVS [SC13]

Lead author papers
Co-author papers
Papers based on my work

Use case FusionFS: a distributed file system

- A fully distributed file system, based on FUSE
- All-in-one: compute node, storage server, metadata server on one machine
- Using ZHT for metadata management
- Evaluated on BlueGene/P and Kodiak up to 1K node
- Collaboration with Dongfang Zhao, PhD 2015

Use case FusionFS: a distributed file system

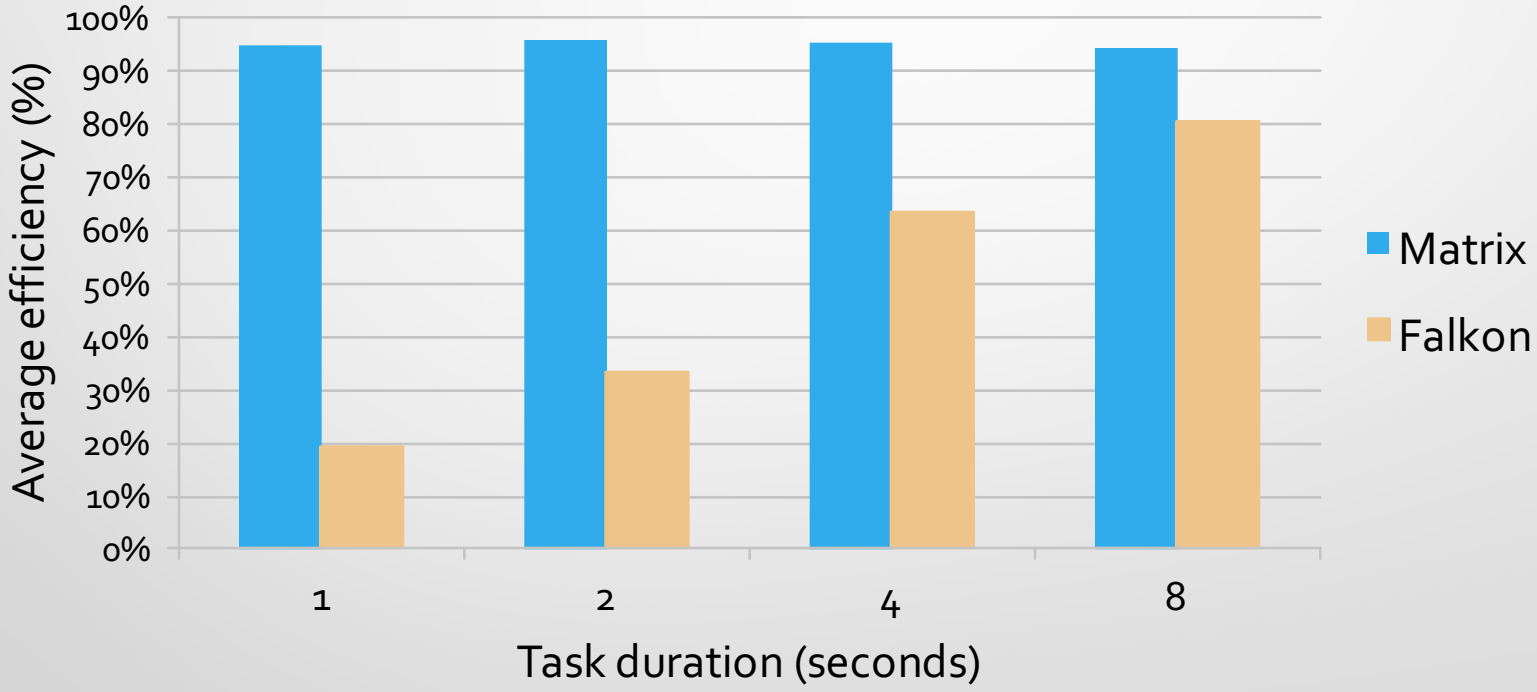


Metadata performance:
Weak scaling: creating 10k empty files in 1 directory on each node

Use case MATRIX: a distributed scheduling framework

- Optimized for many-task-computing
- Fully distributed design
- Adaptive work stealing
- Using ZHT to submit jobs and monitor status
- Collaboration with Ke Wang, PhD 2015

Use case MATRIX: a distributed scheduling framework

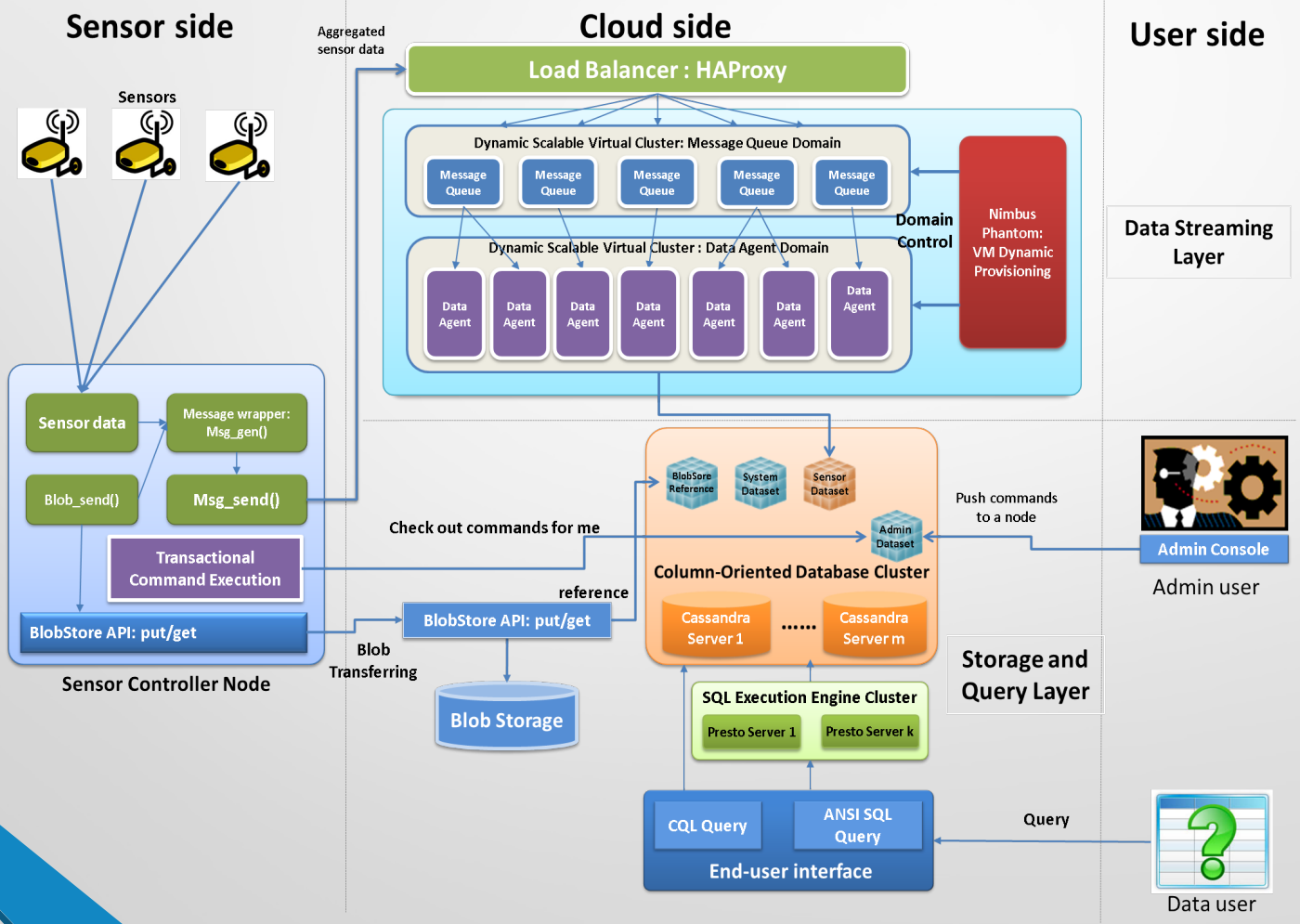


Scheduling efficiency on different job granularity

Use case WaggleDB:
NoSQL DB for sensor network data service

- Project: WaggleDB at ANL, 2014
- NoSQL database + distributed message queue
- Collaborator: Kate Keahey, ANL
- Publication:
  - A Dynamically Scalable Cloud Data Infrastructure for Sensor Networks, **ScienceCloud15**
  - A Cloud-based Interactive Data Infrastructure for Sensor Networks, **SC14 research poster**

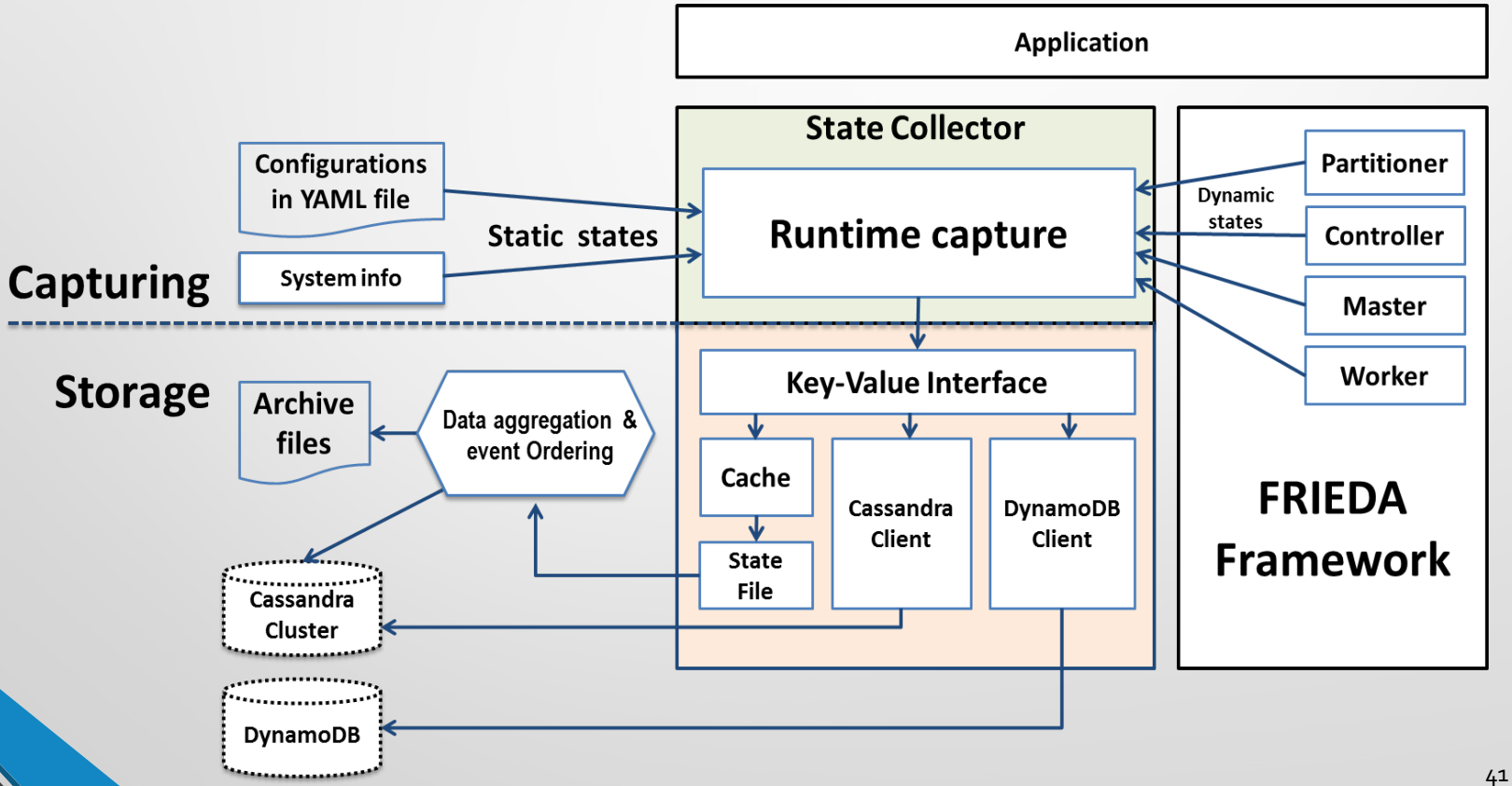
WaggleDB Architecture



Use case FRIEDA-State: NoSQL DB for State Management on Cloud

- Project: FRIEDA-State at LBL, 2013
- Collaborator: Lavanya Ramakrishnan, LBL
- Publication: Scalable State Management for Scientific Applications in the Cloud**, IEEE BigData 2014**
- Funding info
  - DOE DE-AC02-05CH11231
  - NSF No. 0910812

FRIEDA-State architecture



Conclusion

- Prove that NoSQL systems are a fundamental building block for more complex distributed systems
  - Storage systems: ZHT/Q, FusionFS, Istore
  - Provenance: FusionProv
  - Job scheduling systems: MATRIX, Slurm++
  - Event streaming systems: WaggleDB, FRIEDA-State
  - Message queue systems: FqbriQ

Lessons Learned

- Decentralized architecture
  - High Performance
  - Excellent scalability
  - Improved fault tolerance
- Simplicity
  - Light-weight design
  - Little reliance on complex software stack
  - Easy adoption as building block for more complex systems

Future Research Directions

- Building extreme-scale system services with NoSQL storage systems
- Using NoSQL storage to help on traditional HPC applications problems
  - Scalability
  - Fault tolerance

Thank you!

Q&A