SPARK SUMMIT 2017

Spark on Supercomputers: A Tale of the Storage Hierarchy

Costin Iancu, Khaled Ibrahim – LBNL

Nicholas Chaimov – U. Oregon

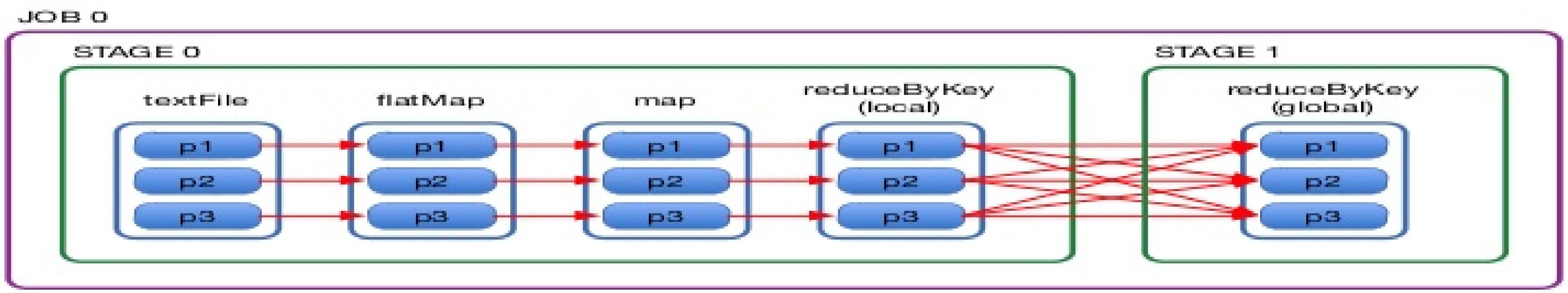
Apache Spark

- Developed for cloud environments
- Specialized runtime provides for
  - Performance ☺, Elastic parallelism, Resilience
- Programming productivity through
  - HLL front-ends (Scala, R, SQL), multiple domain-specific libraries: Streaming, SparkSQL, SparkR, GraphX, Splash, MLLib, Velox
- We have huge datasets but little penetration in HPC

Apache Spark

- In-memory Map-Reduce framework
- Central abstraction is the Resilient Distributed Dataset.

- **Data movement is important**
  - **Lazy, on-demand**
  - **Horizontal (node-to-node) – shuffle/Reduce**
  - **Vertical (node-to-storage) - Map/Reduce**



Data Centers/Clouds

Node local storage, assumes all disk operations are equal

Disk I/O optimized for latency

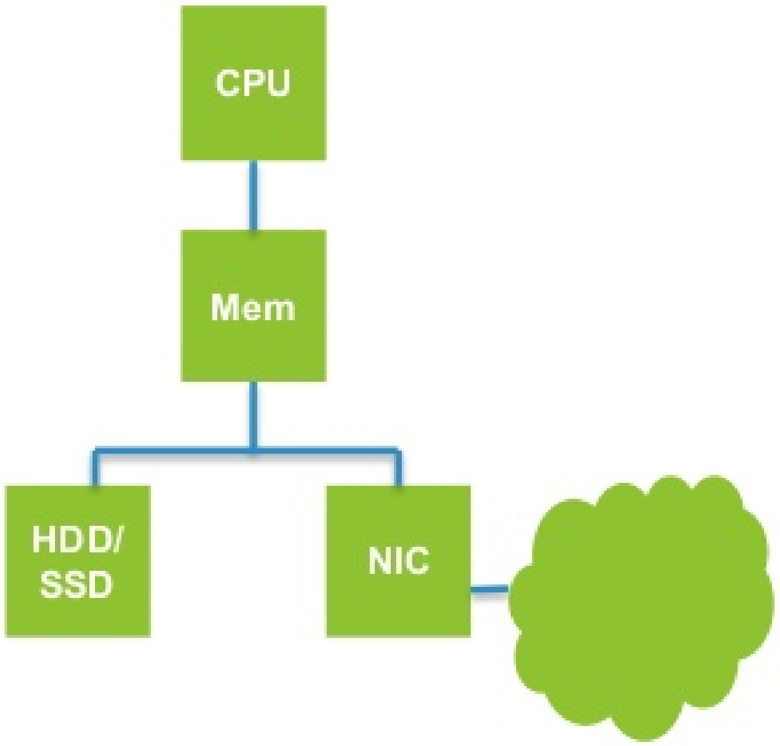
Network optimized for bandwidth

HPC

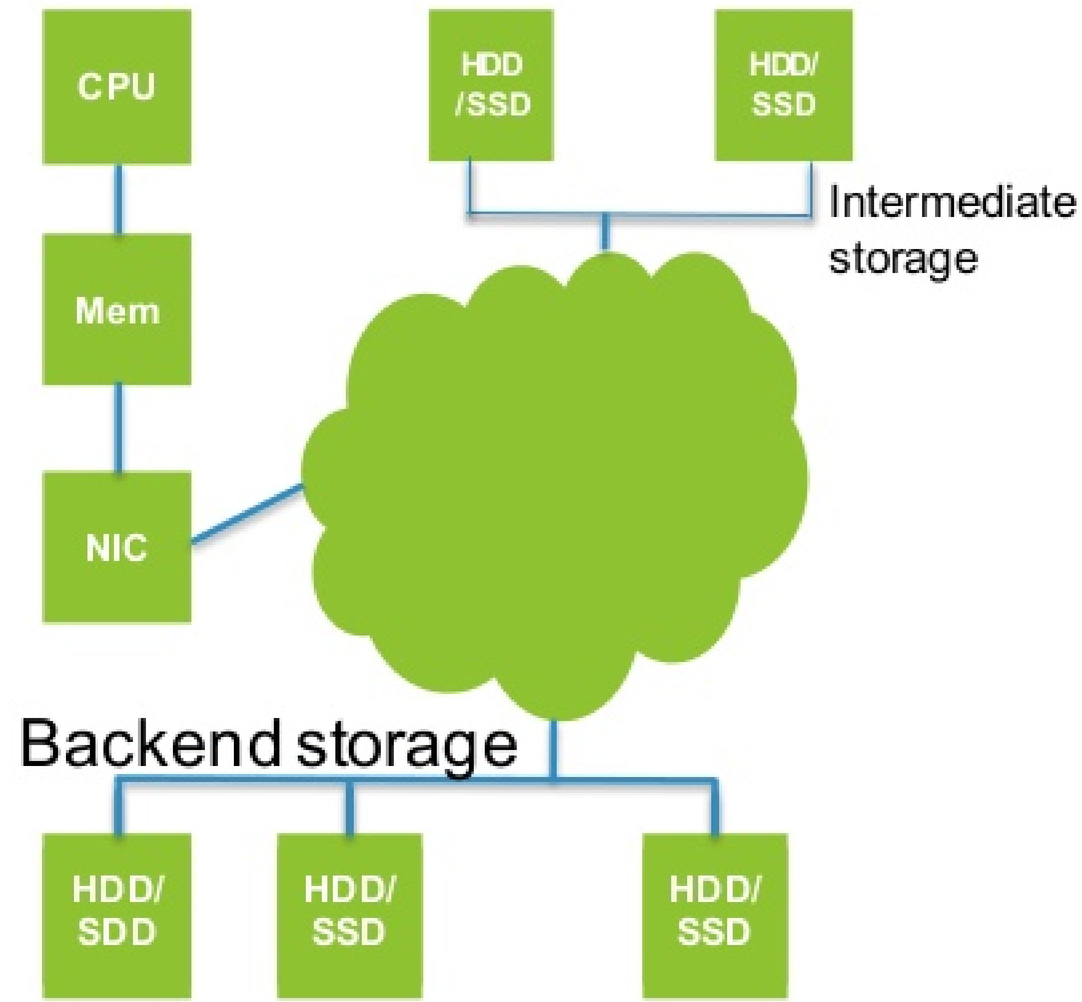
Global file system, asymmetry expected

Disk I/O optimized for bandwidth

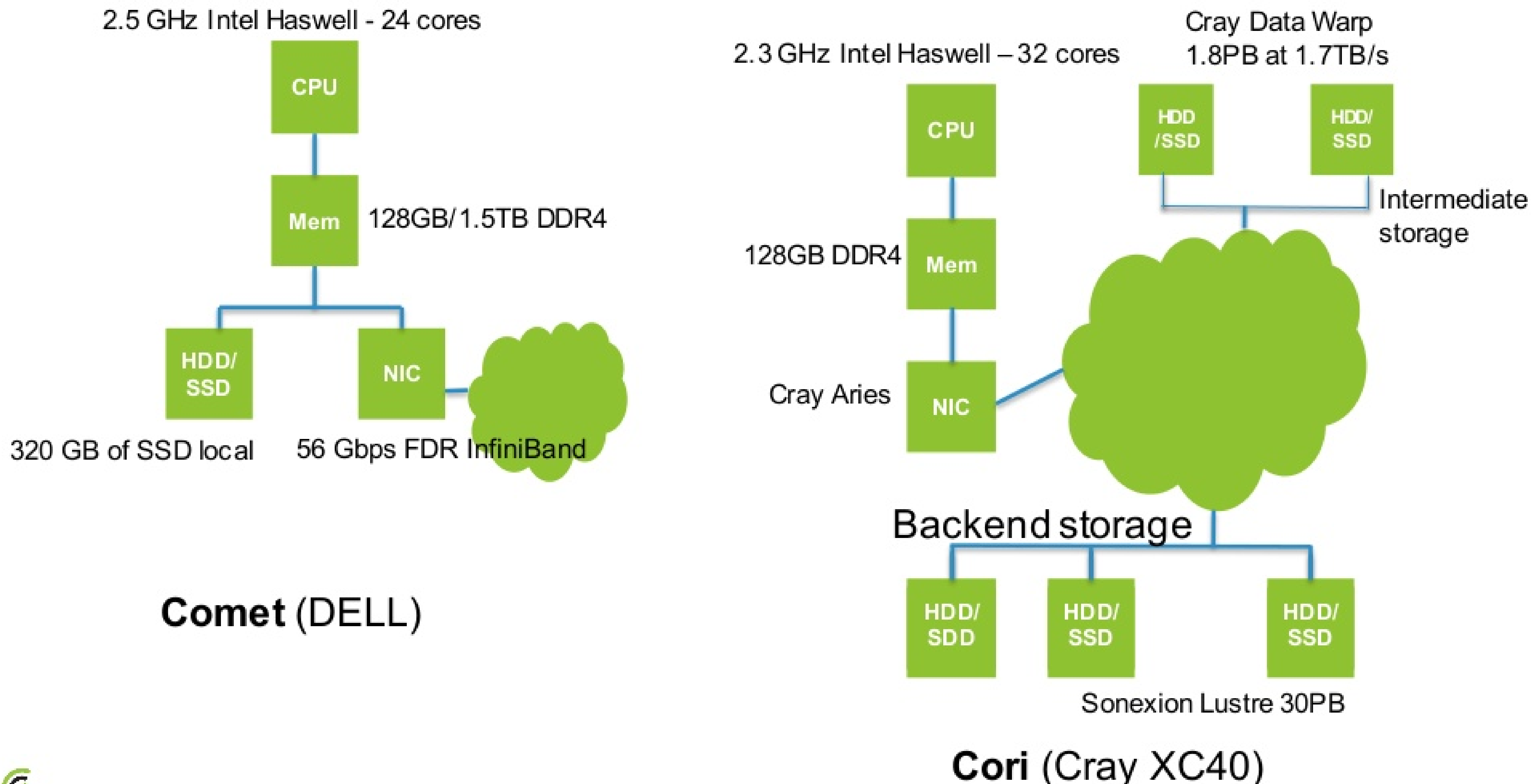
Network optimized for latency



Cloud: commodity CPU, memory, HDD/SSD NIC
Data appliance: server CPU, large fast memory, fast SSD



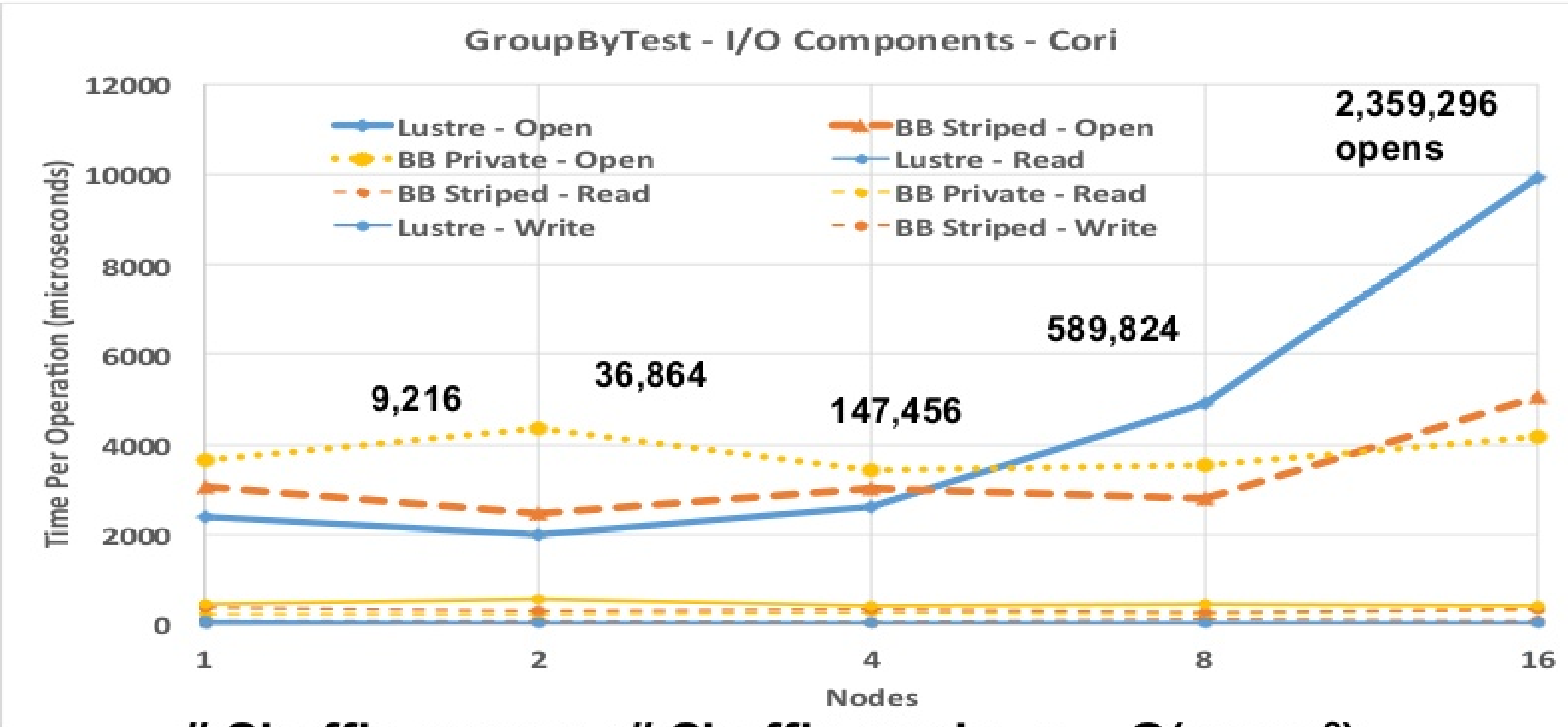
HPC: server CPU, fast memory, combo of fast and slower storage


2.5 GHz Intel Haswell - 24 cores
CPU
Mem
128GB/1.5TB DDR4
HDD/
SSD
NIC
320 GB of SSD local
56 Gbps FDR InfiniBand
Comet (DELL)
2.3 GHz Intel Haswell – 32 cores
Cray Data Warp
1.8PB at 1.7TB/s
CPU
HDD
/SSD
HDD/
SSD
Intermediate
storage
128GB DDR4
Mem
Cray Aries
NIC
Backend storage
HDD/
SDD
HDD/
SSD
HDD/
SSD
Sonexion Lustre 30PB
Cori (Cray XC40)

Scaling Spark on Cray XC40

(It’s all about file system metadata)

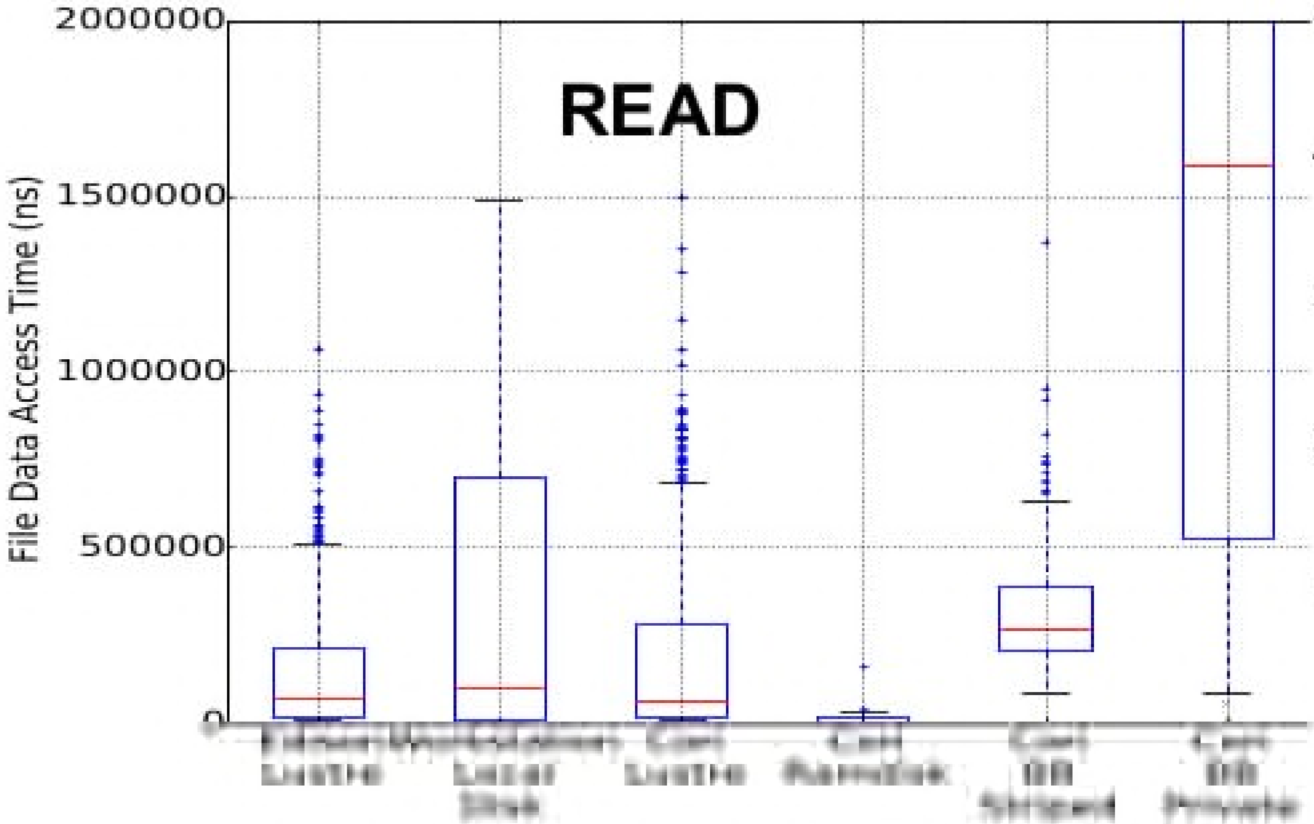
Not ALL I/O is Created Equal

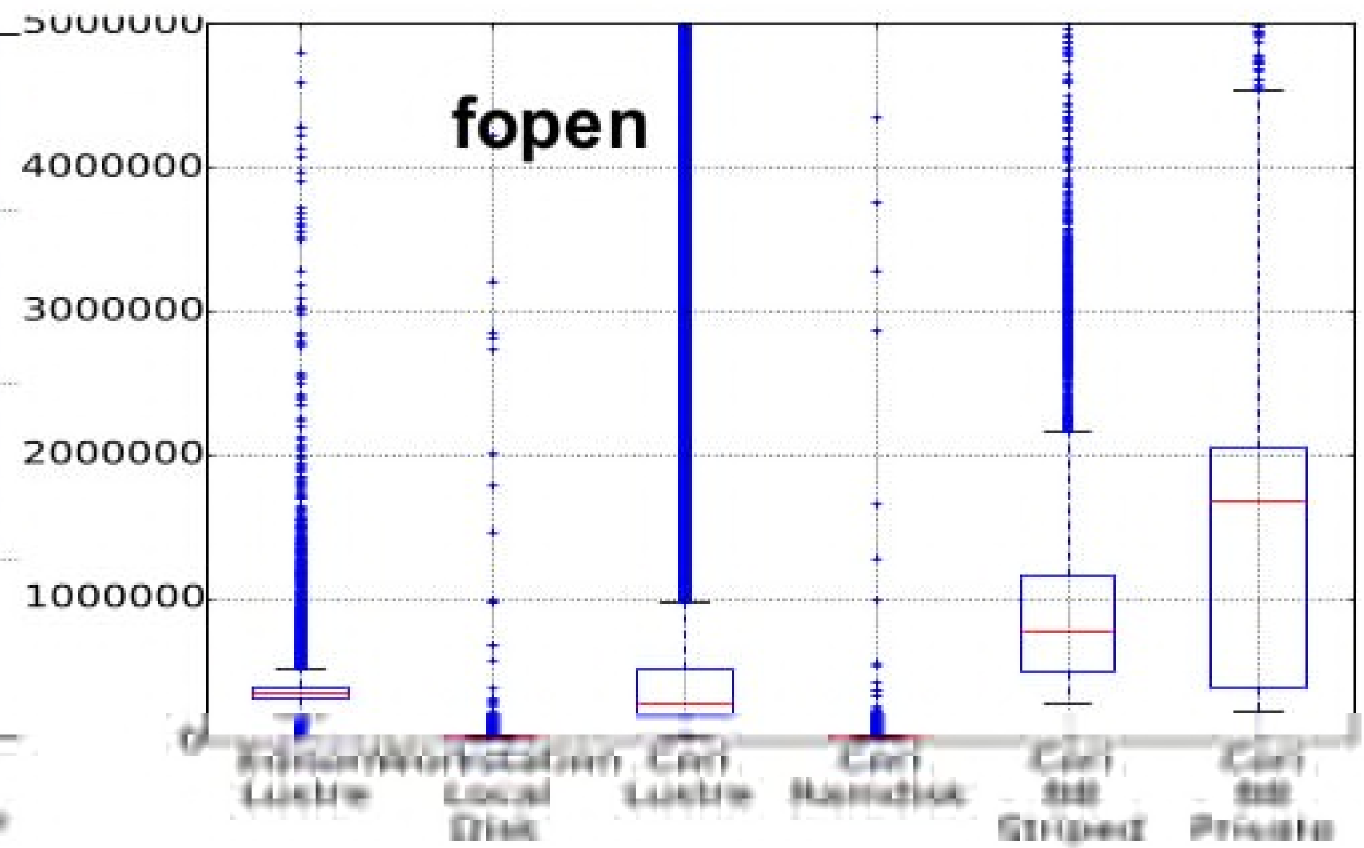


 # Shuffle opens = # Shuffle reads → O(cores^2)

Time per open increases with scale, unlike read/write

I/O Variability is HIGH





fopen is a problem:

- Mean time is 23X larger than SSD
- Variability is 14,000X

Improving I/O Performance

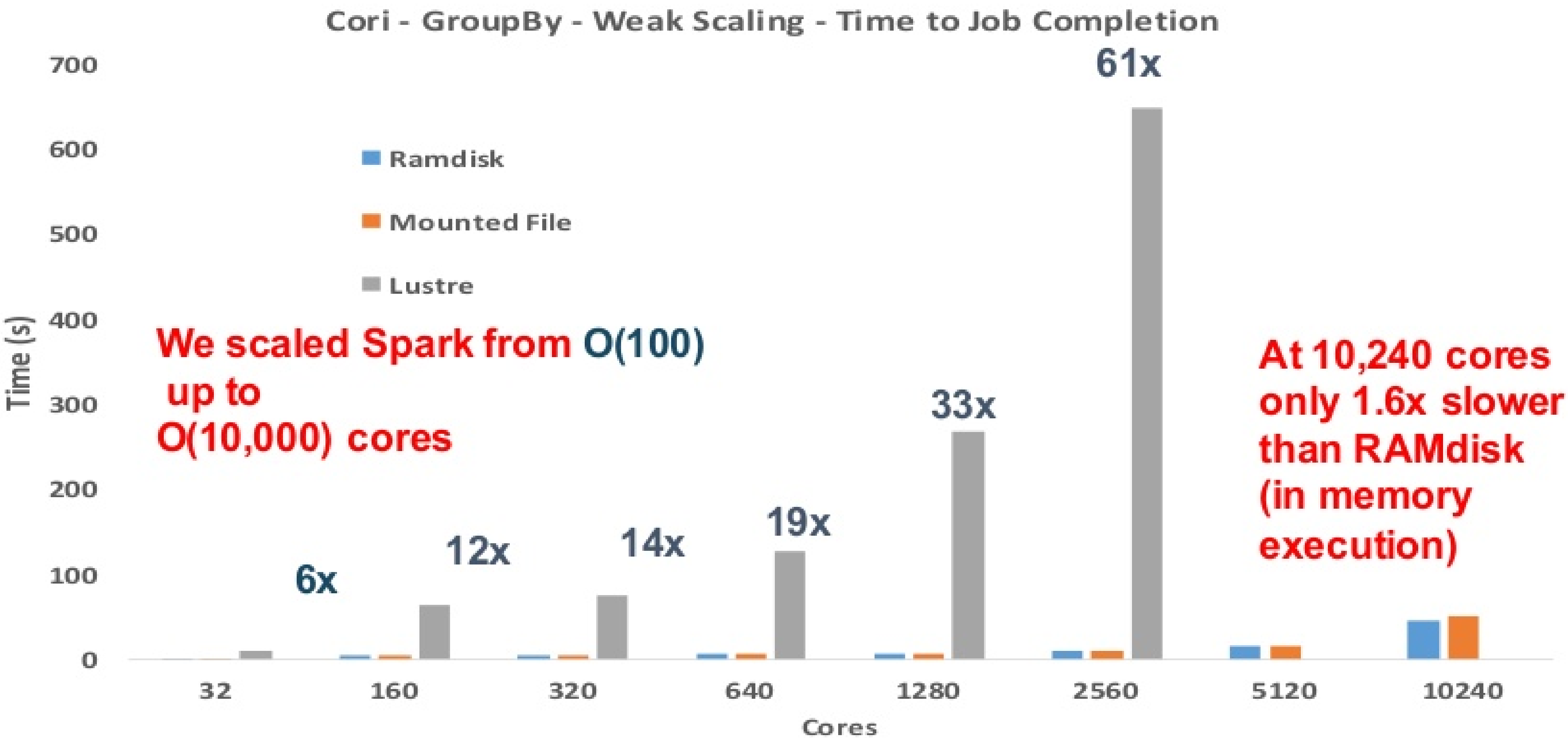
Eliminate file metadata operations

1. **Keep files open (cache `fopen`)**
    - Surprising 10%-20% improvement on data appliance
    - Argues for user level file systems, gets rid of serialized system calls
2. **Use file system backed by single Lustre file for shuffle**
    - This should also help on systems with local SSDs
3. **Use containers**
    - Speeds up startup, up to 20% end-to-end performance improvement

- Solutions need to be used in conjunction
    - E.g. fopen from Parquet reader

Plenty of details in "Scaling Spark on HPC Systems". HPDC 2016

Scalability



SPARK SUMMIT 2017

File-Backed Filesystems

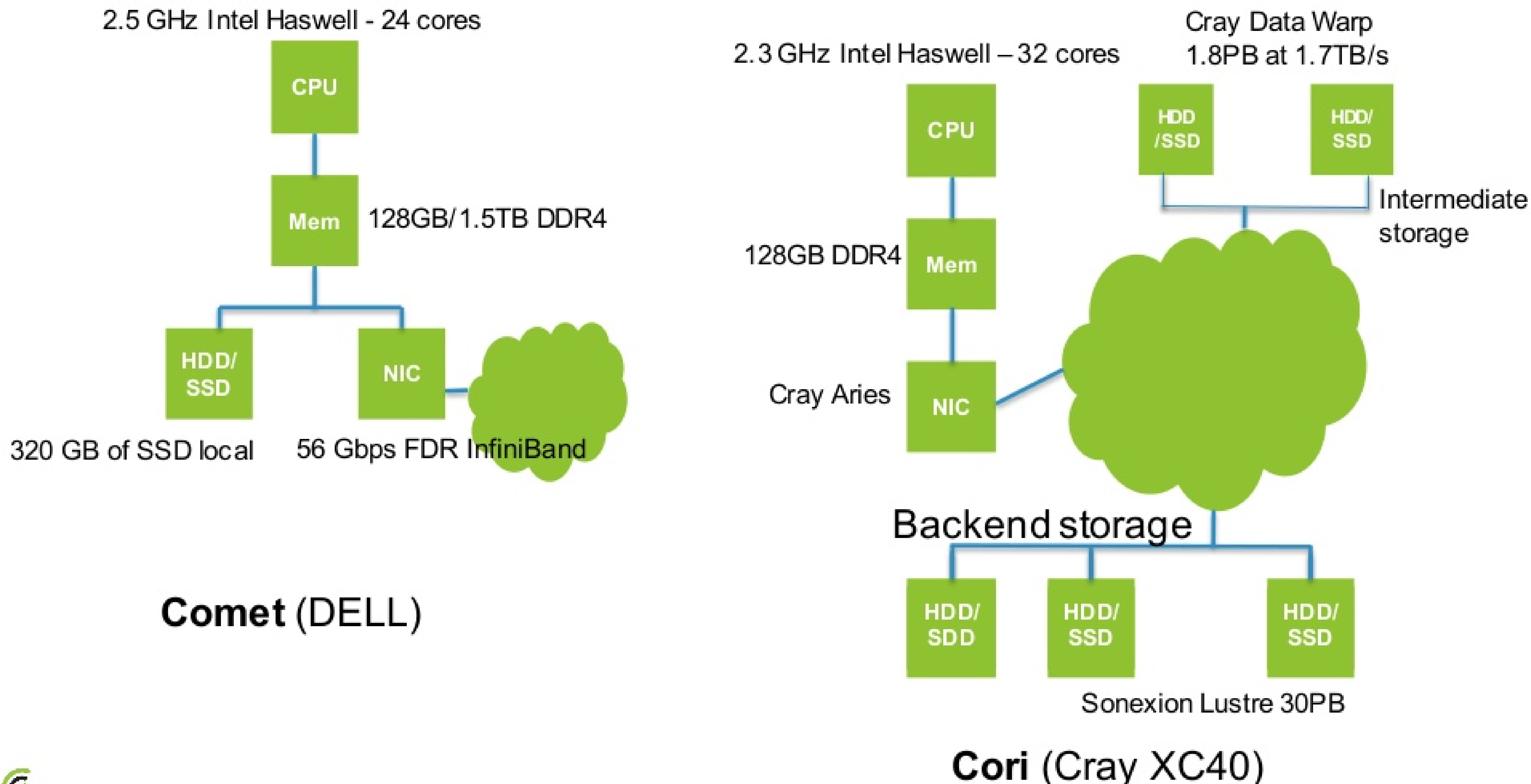
- **NERSC Shifter** (container infrastructure for HPC)
  - Compatible with Docker images
  - Integrated with Slurm scheduler
  - Can control mounting of filesystems within container

- **Per-Node Cache**
  - File-backed filesystem mounted within each node's container instance at common path (`/mnt`)
  - `--volume=$SCRATCH/backingFile:/mnt:perNodeCache=size=100G`
  - File for each node is created stored on backend Lustre filesystem
  - Single file open — intermediate data file opens are kept local

Now the fun part ☺

Architectural Performance Considerations

Cori		**Comet**
The Supercomputer	**vs**	**The Data Appliance**

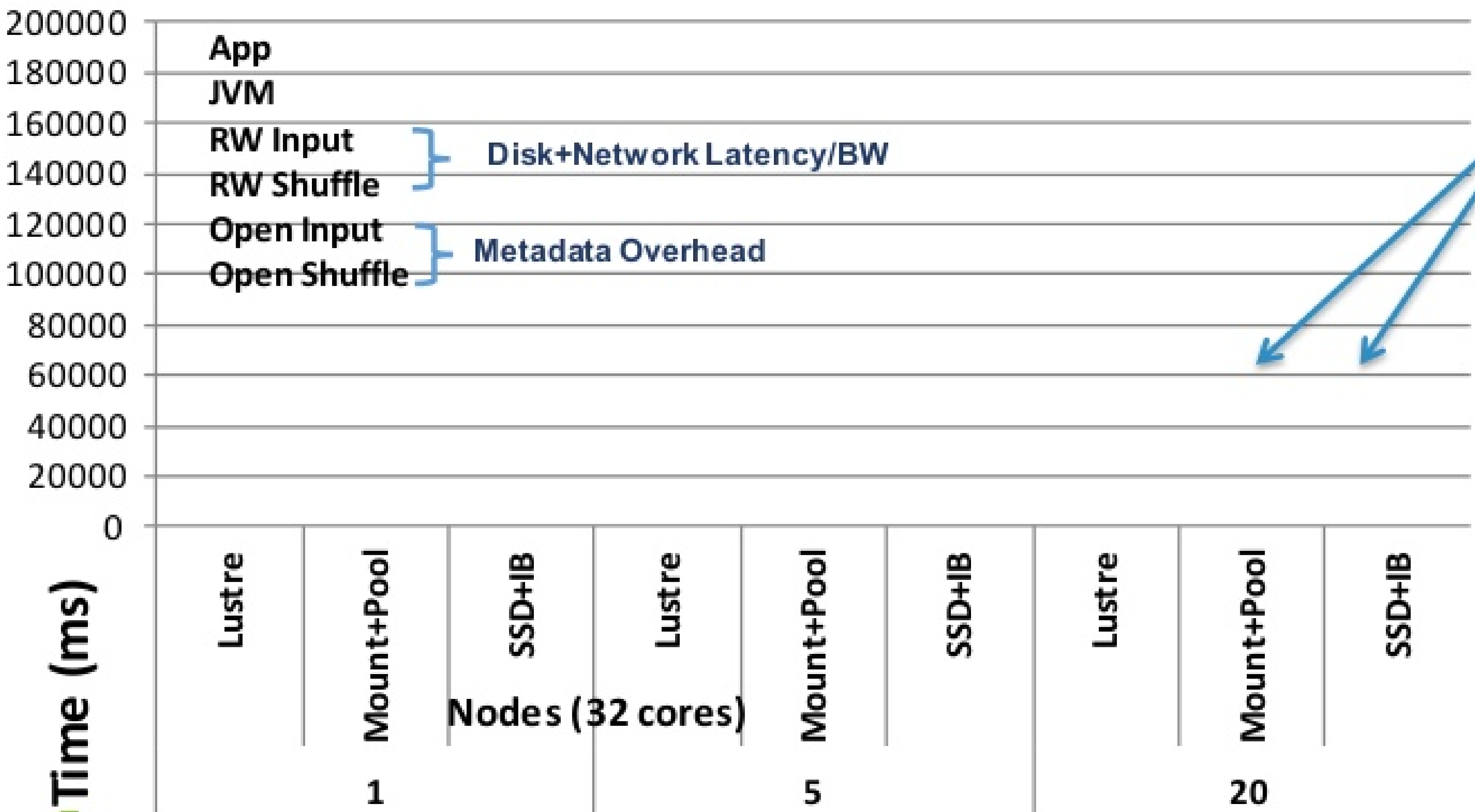

2.5 GHz Intel Haswell - 24 cores
CPU
Mem
128GB/1.5TB DDR4
HDD/ SSD
NIC
320 GB of SSD local
56 Gbps FDR InfiniBand
Comet (DELL)
2.3 GHz Intel Haswell – 32 cores
Cray Data Warp
1.8PB at 1.7TB/s
CPU
HDD /SSD
HDD/ SSD
Intermediate storage
128GB DDR4
Mem
Cray Aries
NIC
Backend storage
HDD/ SDD
HDD/ SSD
HDD/ SSD
Sonexion Lustre 30PB
Cori (Cray XC40)

CPU, Memory, Network, Disk?

- Multiple extensions to Blocked Time Analysis (Ousterhout, 2015)
- BTA indicated that CPU dominates
  - Network 2%, disk 19%
- Concentrate on scaling out, weak scaling studies
  - Spark-perf, BigDataBenchmark, TPC-DS, TeraSort
- Interested in determining right ratio, machine balance for
  - CPU, memory, network, disk …
- Spark 2.0.2 & Spark-RDMA 0.9.4 from Ohio State University, Hadoop 2.6

Storage hierarchy and performance

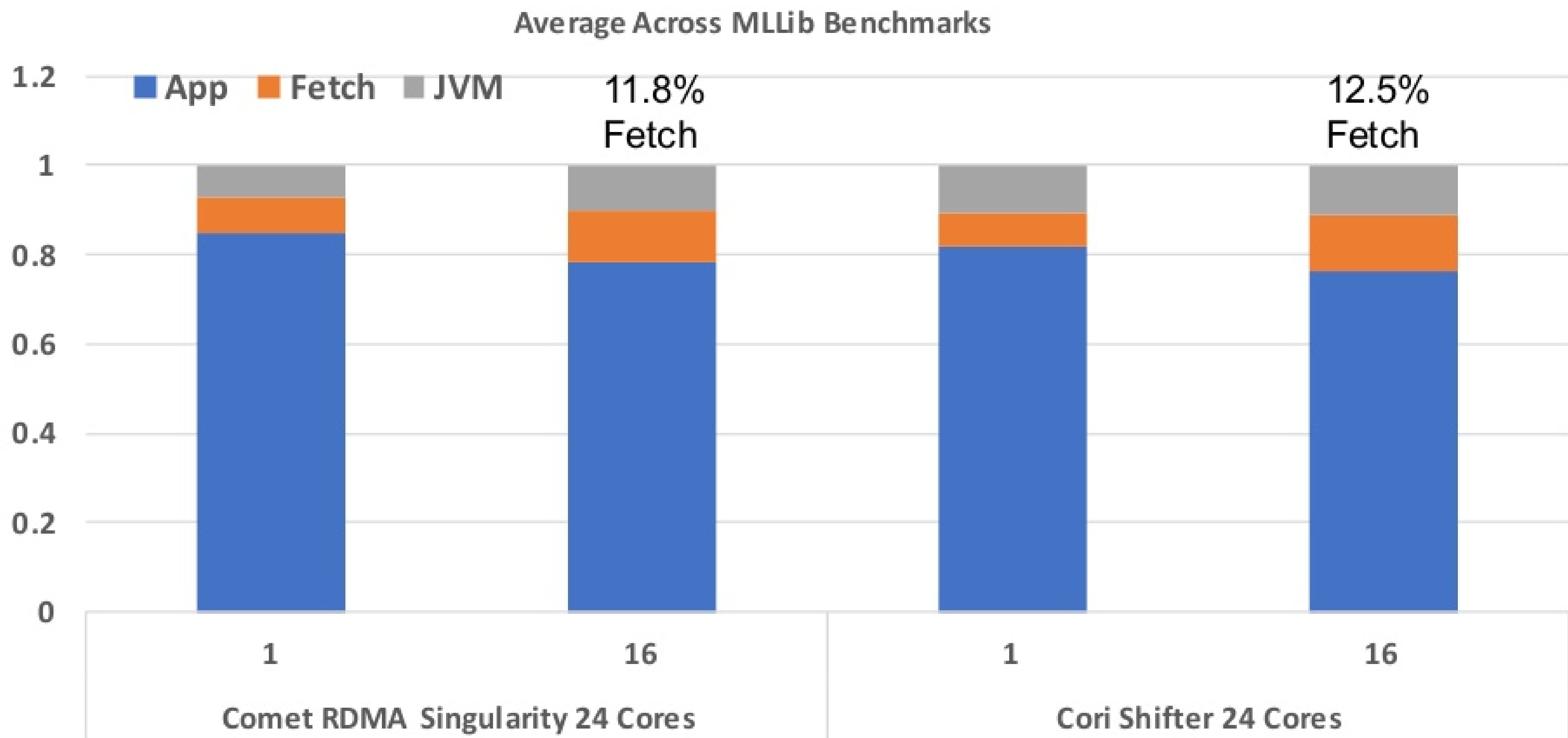
Global Storage Matches Local Storage



- Variability matters more than advertised latency and bandwidth number
- Storage performance obscured/mitigated by network due to client/server in BlockManager
    - Small scale local is slightly faster
    - Large scale global is faster

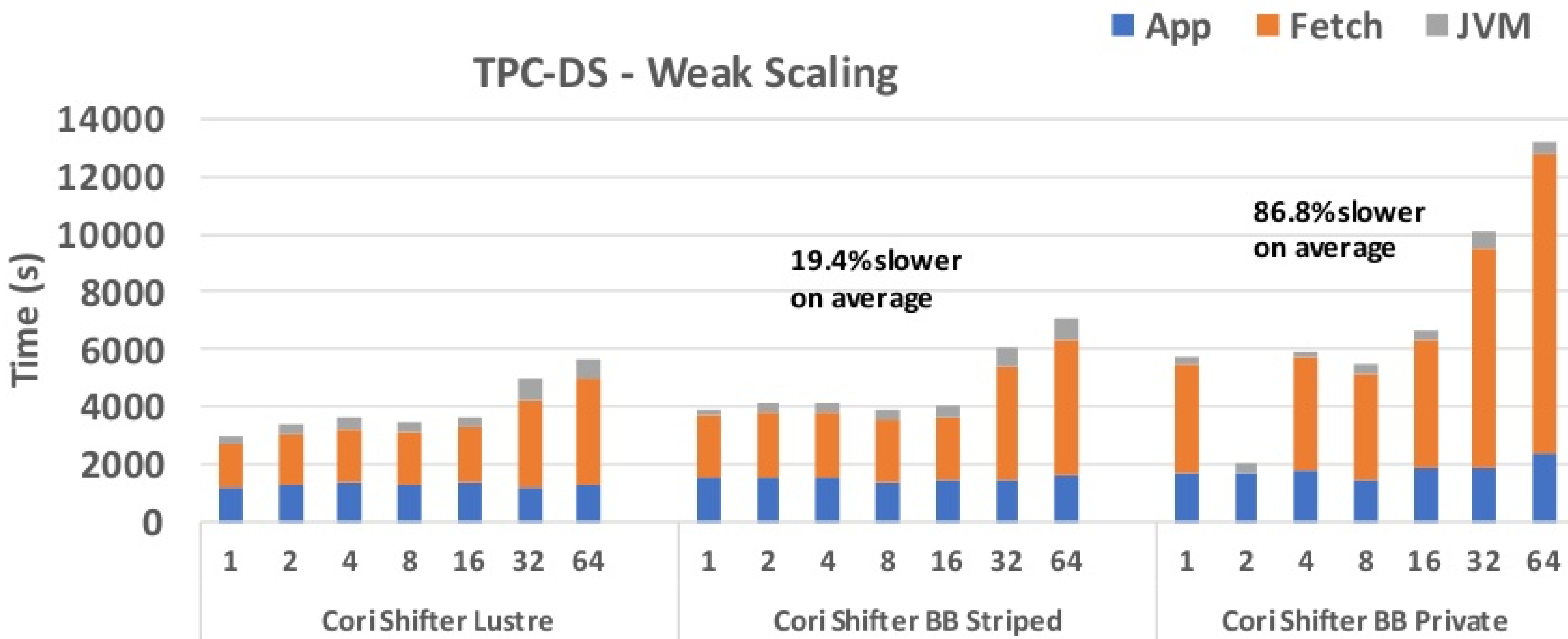
Cray XC40 – TeraSort (100GB/node)

Global Storage Matches Local Storage



SPARK SUMMIT 2017

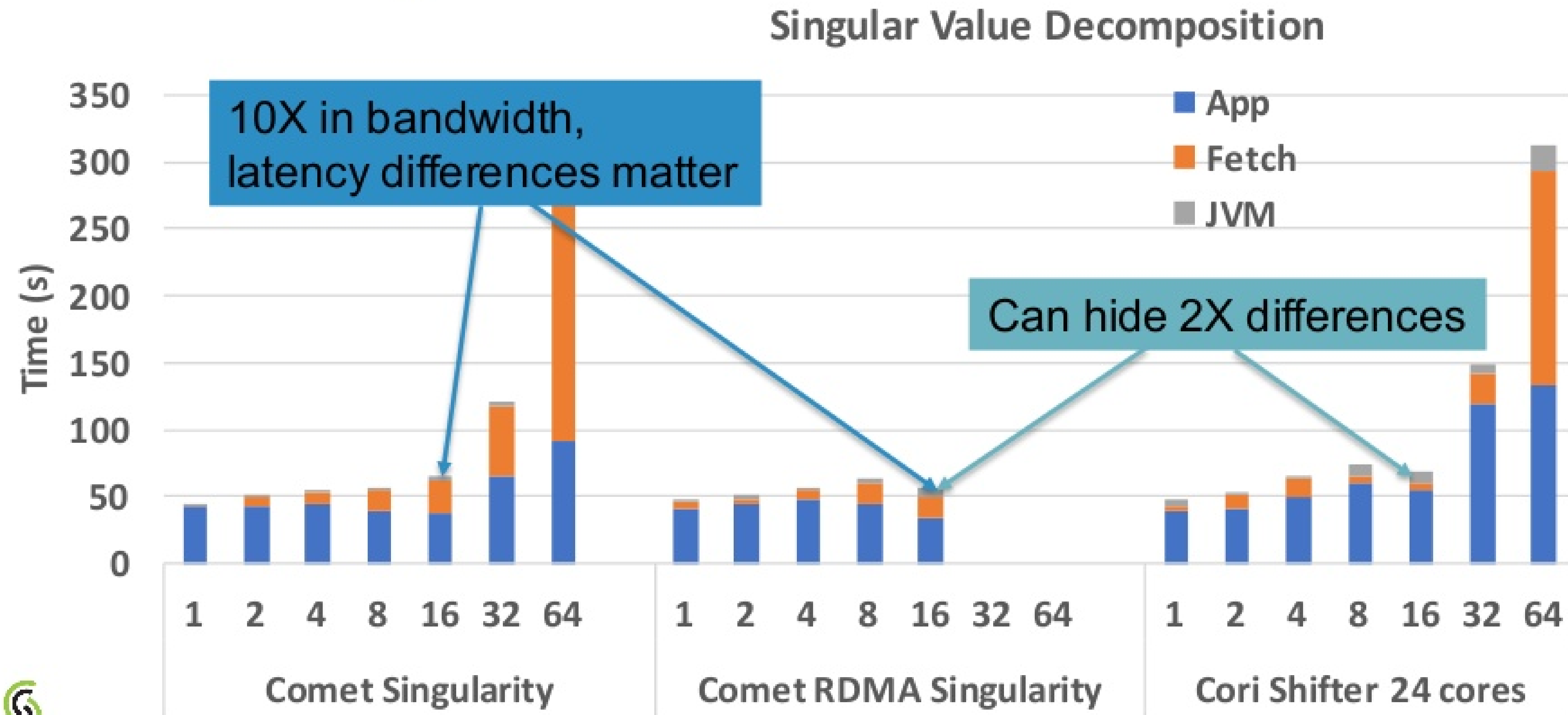
Intermediate Storage Hurts Performance



(Without our optimizations, intermediate storage scaled better)

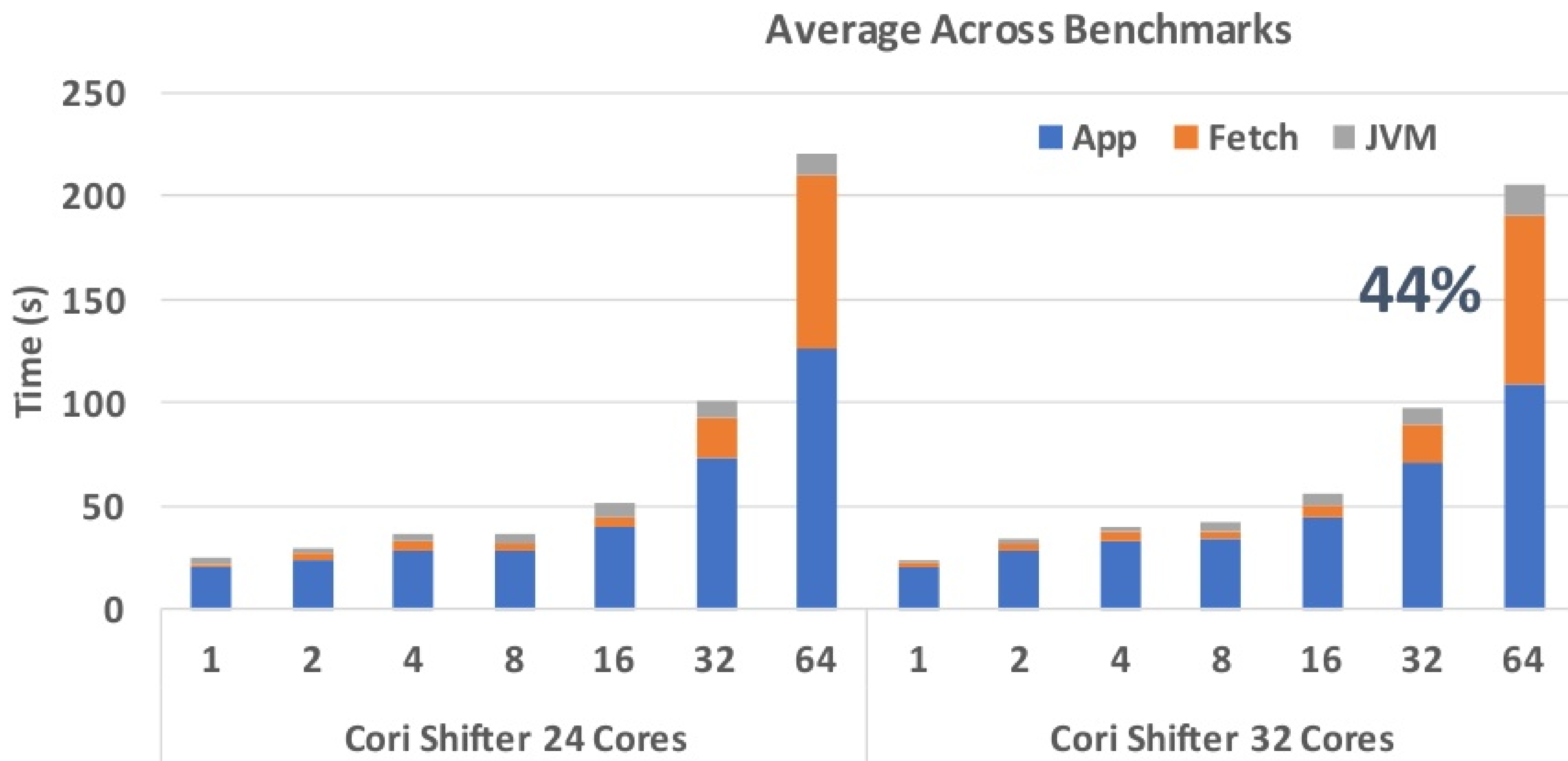
Networking performance

Latency or Bandwidth?



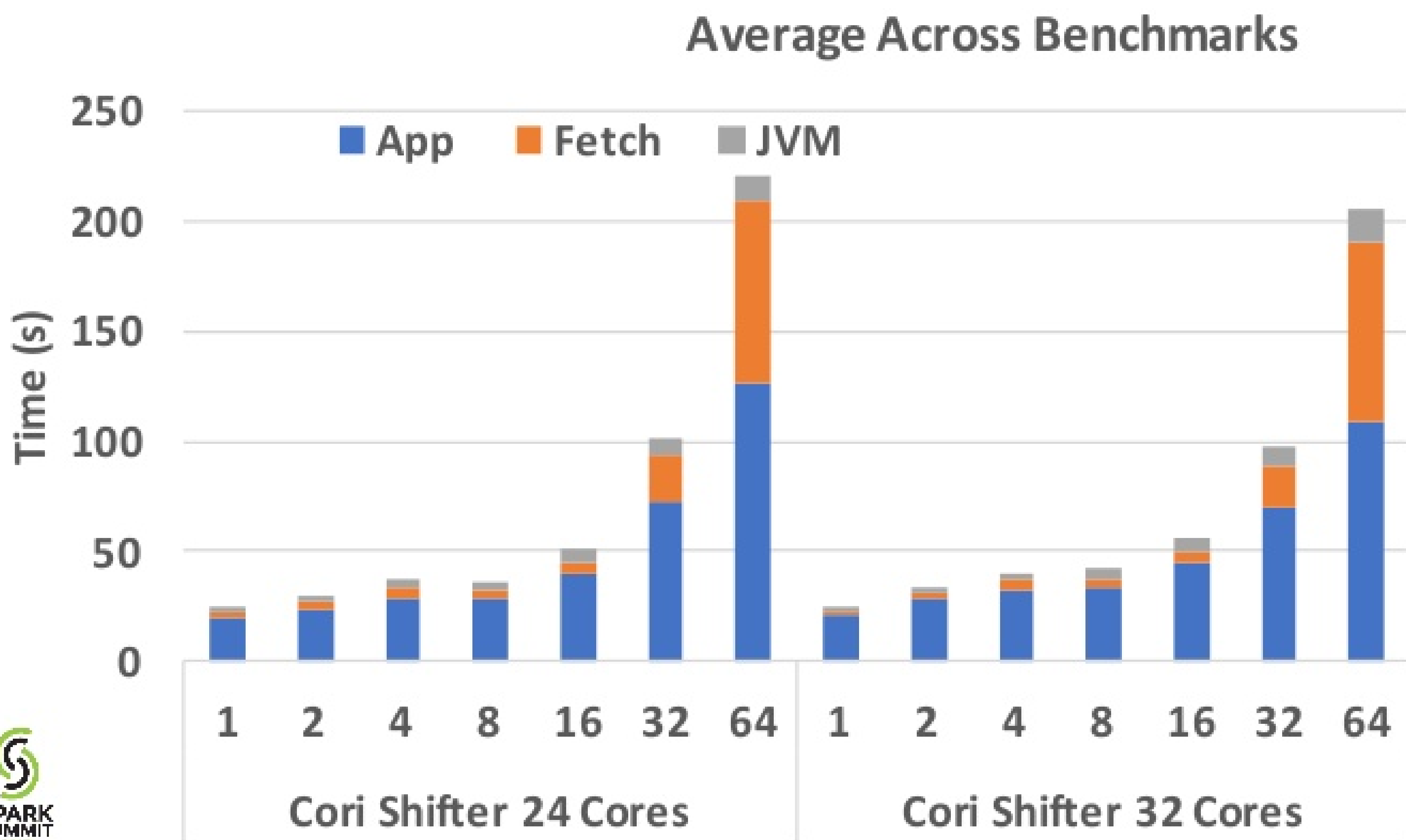
Average message size for spark-perf is 43B

Network Matters at Scale



CPU

More cores or better memory?



- Need more cores to hide disk and network latency at scale.
- Preliminary experiences with Intel KNL are bad
  - Too much concurrency
  - Not enough integer throughput
- Execution does not seem to be memory bandwidth limited

Summary/Conclusions

- Latency and bandwidth are important, but not dominant
  - Variability more important than marketing numbers
- Network time dominates at scale
  - Network, disk is mis-attributed as CPU

- Comet matches Cori up to 512 cores, Cori twice as fast at 2048 cores
  - Spark can run well on global storage

- Global storage opens the possibility of global name space, no more client-server

Ackowledgement

Work partially supported by

Intel Parallel Computing Center: Big Data Support for HPC

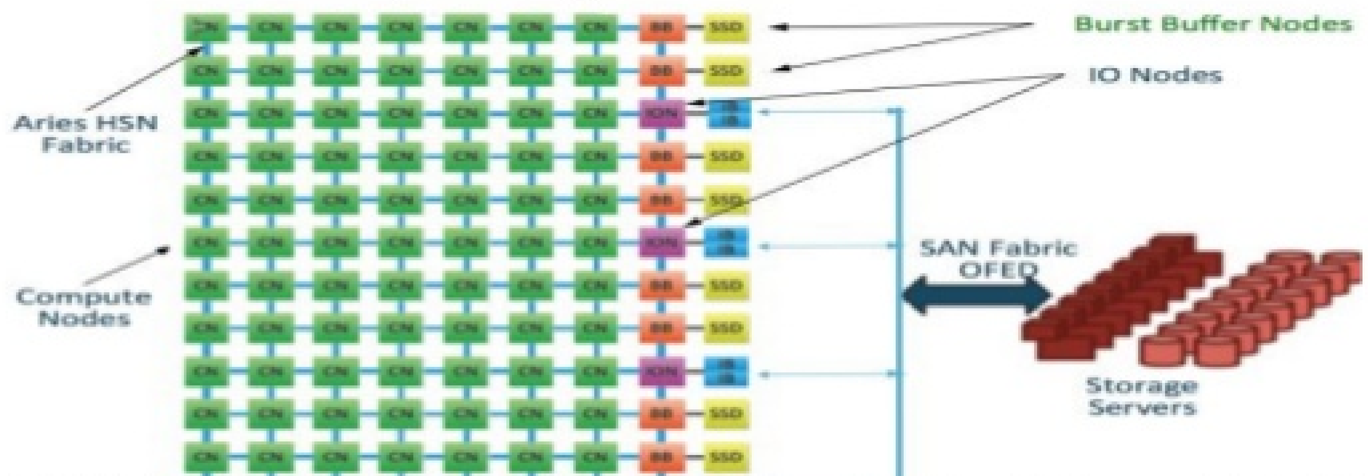
Thank You.

Questions, collaborations, free software

cciancu@lbl.gov
kzibrahim@lbl.gov
nchaimov@uoregon.edu

Setup

- **Cray XC30 at NERSC (Edison): 2.4 GHz IvyBridge - Global**
- **Cray XC40 at NERSC (Cori): 2.3 GHz Haswell + Cray DataWarp**



- **Comet at SDSC: 2.5GHz Haswell, InfiniBand FDR, 320 GB SSD, 1.5TB memory - LOCAL**