Performance Evaluation of Apache Spark on Cray XC

Nicholas Chaimov (U Oregon)
Costin Iancu, Khaled Ibrahim, Shane Canon, Jay Srinivasan (LBNL)
Allen Malony (U Oregon)

Data Analytics

- **Spark- "fast and general engine for large-scale data processing"**

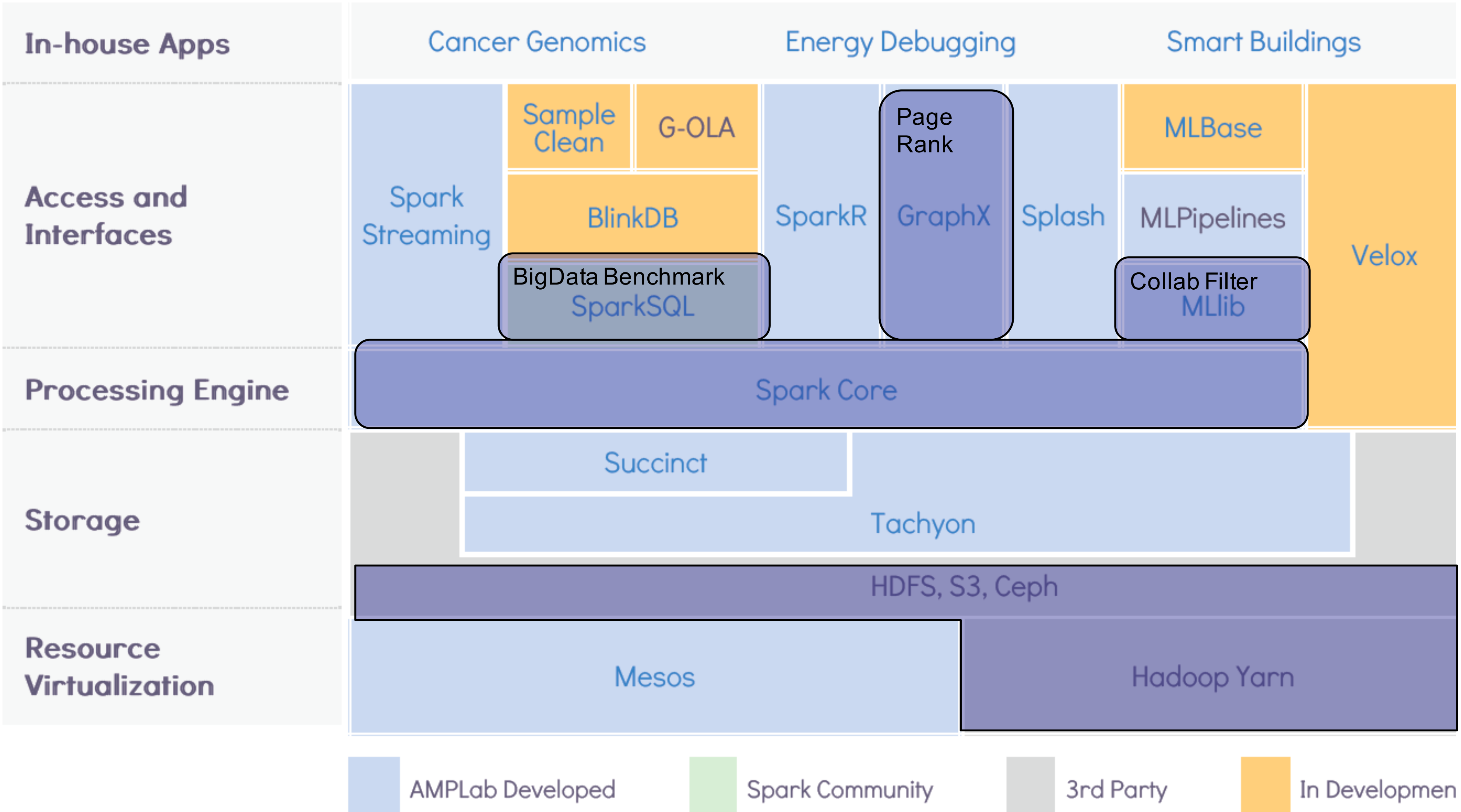
- **Specialized runtime provides for**
  - Performance (☺)
  - Elastic parallelism
  - Resilience

- **Improves programmer productivity through**
  - HLL front-ends (Scala, R, SQL)
  - Multiple domain-specific libraries: Streaming, SparkSQL, SparkR, GraphX, Splash, MLLib, Velox

- **Developed for cloud environments, performance "satisfactory"**

Berkeley Data Analytics Stack

COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP



From https://amplab.cs.berkeley.edu/software/

Design Assumptions

- **Spark expects**
  - Local disk with HDFS overlay for distributed file system
  - Fast local disk (SSD) for shuffle files
  - Assumes ALL disk operations are fast

Clouds

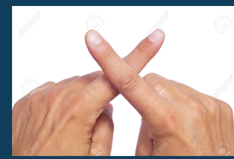
Disk I/O optimized for latency

Network optimized for bandwidth

HPC

Disk I/O optimized for bandwidth

Network optimized for latency

HPC  Cloud

- **Differences in architecture guide the software design**
  - Evaluation of Spark on HPC architecture (Cray XC30)
  - Techniques to improve performance on HPC architectures by eliminating disk I/O overhead

- **Can HPC architectures provide performance advantages?**
  - **Do we need local disks?**
    - Cloud = node local SSD
    - BurstBuffer = mid layer of SSD storage
    - Lustre = backend storage system
  - **Can we exploit the advantages of HPC networks?**
    - Do we need RDMA optimizations?

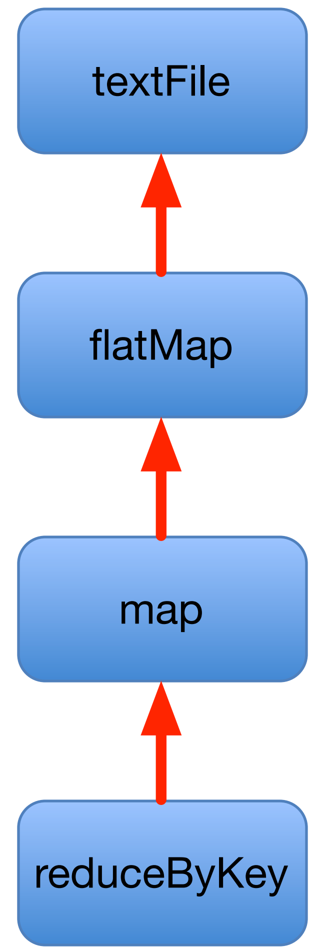
What's in Spark?

Spark

- **Central abstraction is the Resilient Distributed Dataset, or RDD.**
  - Composed of **partitions** of data
    - which are composed of **blocks**.
  - RDDs are created from other RDDs by applying **transformations** or **actions**.
  - Has a **lineage** specifying how its blocks are computed.
  - Requesting a block either retrieves from cache or triggers computation.

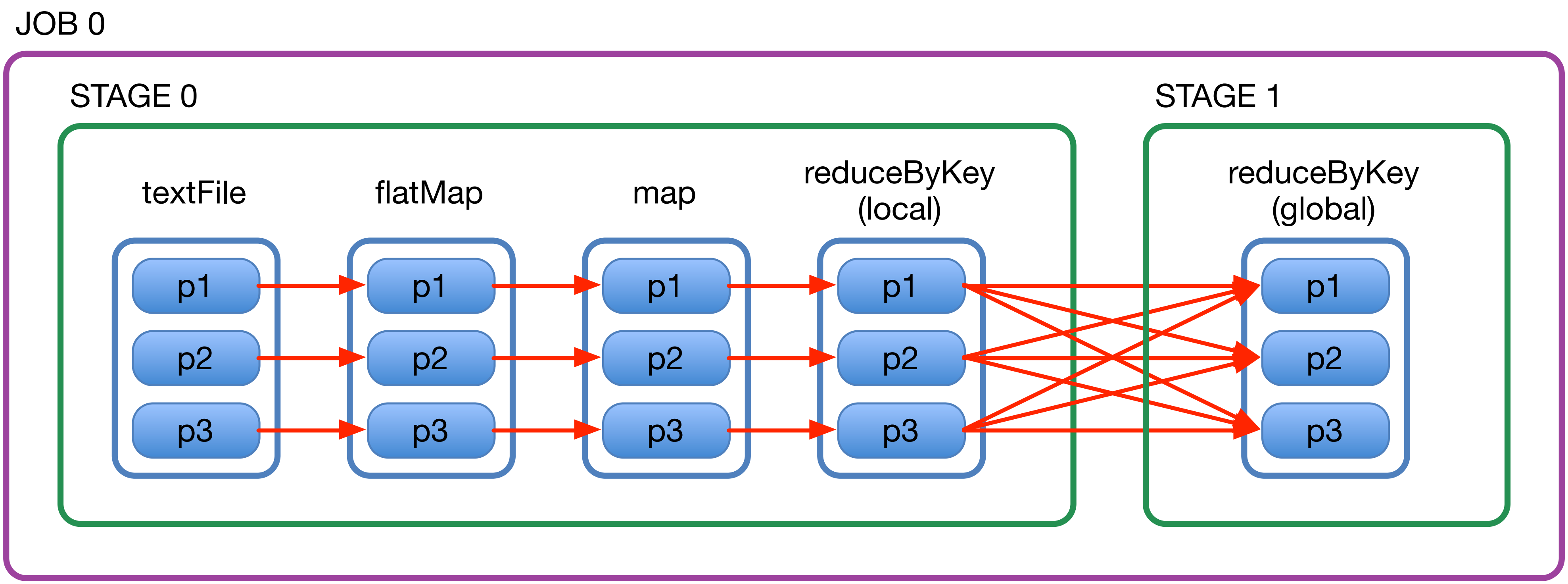
Word Count Example


```
val textFile = sc.textFile(”input.txt")
val counts = textFile.flatMap(line => line.split(" "))
                     .map(word => (word, 1))
                     .reduceByKey(_ + _)
counts.collect()
```



- *textFile*, *flatMap*, *map* and *reduceByKey* are **transformations**.
  - They do not trigger computation but simply build the lineage.
- *collect* is an **action.**
  - Triggers computation on parent RDD.

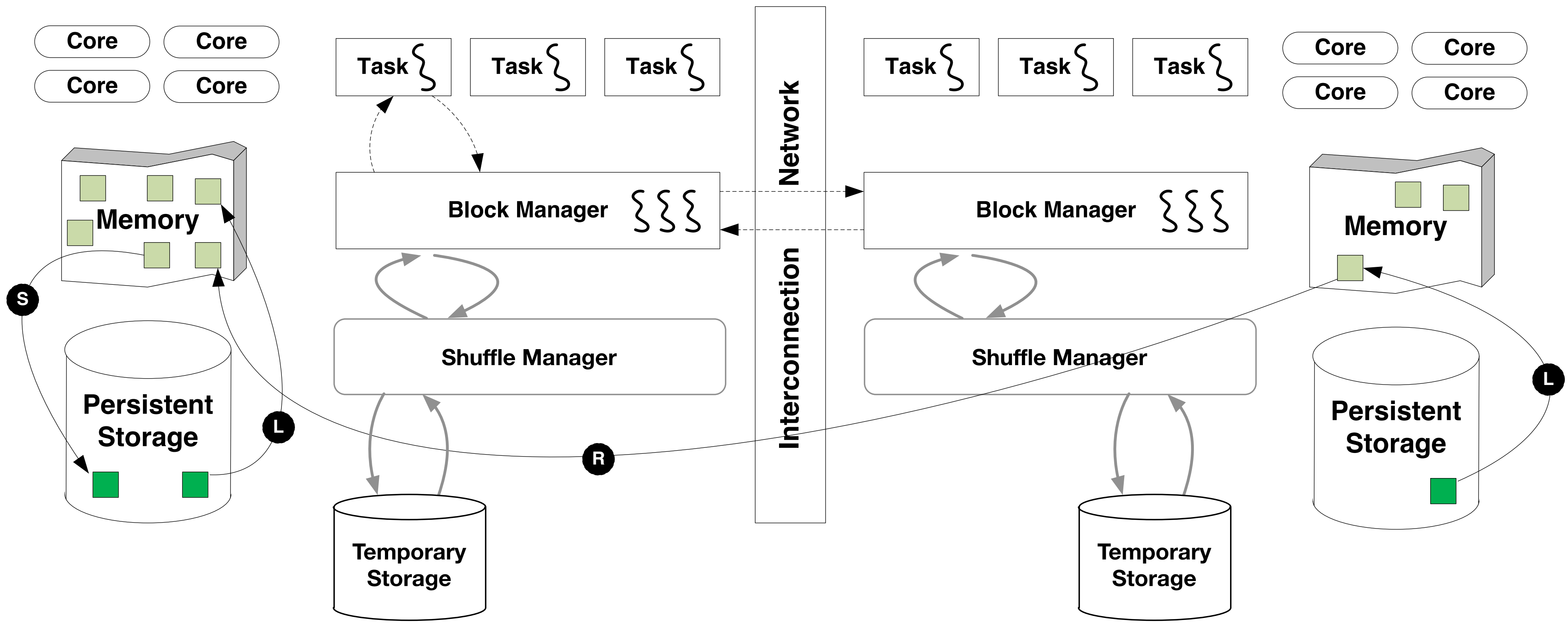
Stages



- Within a stage, each partition is independently computed.
- Inter-partition communication occurs at stage boundaries through **shuffling**.

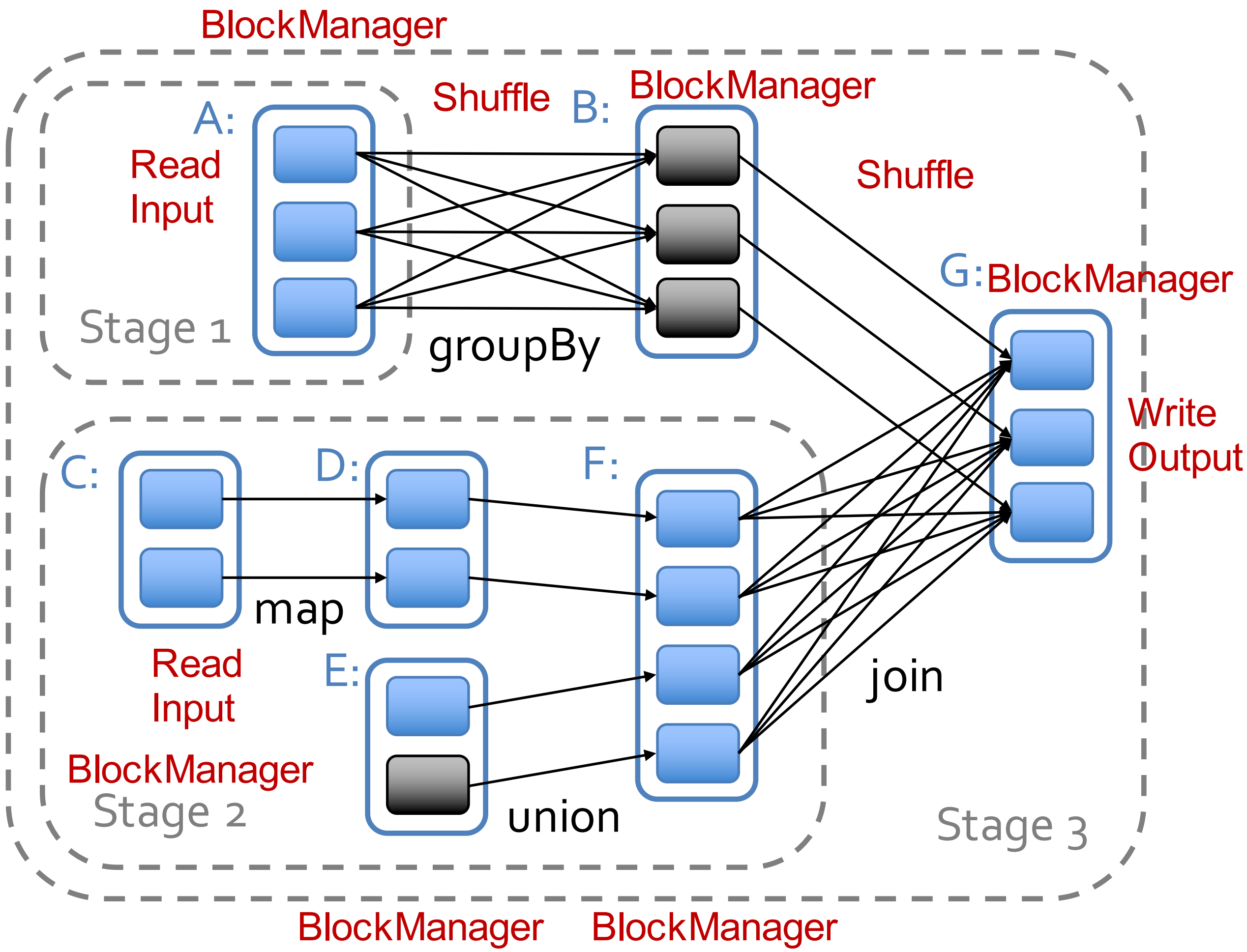
Data Movement in Spark

- **Block is the unit of movement and execution**
  - Ideally, blocks are resident in memory
  - Blocks are cached, evicts or spills to disk as necessary
  - If swapped, bring the block from a persistent storage
  - If remote, request the block from a remote manager
- **Shuffle Manager interacts with Block Manager and local storage**



I/O Happens Everywhere

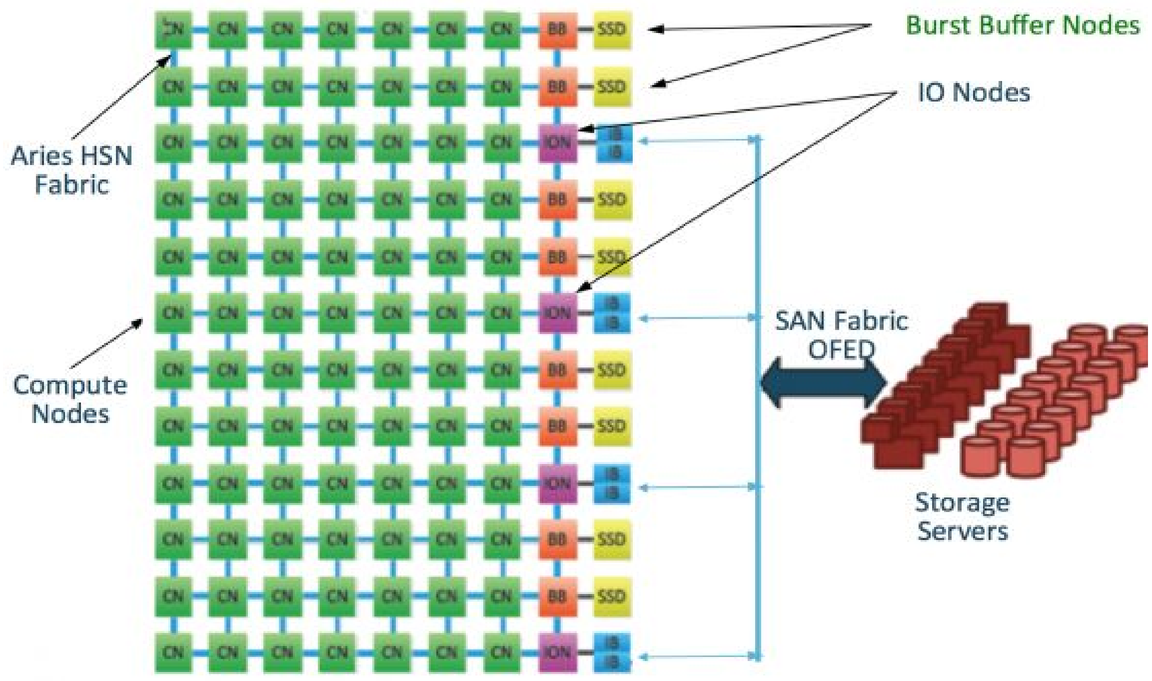
- Program input/output
  - ***Expected to be distributed*** with global namespace (HDFS)
- Shuffle and Block Manager
  - ***Expected to be local*** (Java FileOutputStream)



Evaluation

Setup

- Cray XC 30 at NERSC (Edison): 2.4 GHz IvyBridge
- Cray XC 40 at NERSC (Cori): 2.3 GHz Haswell + Cray DataWarp



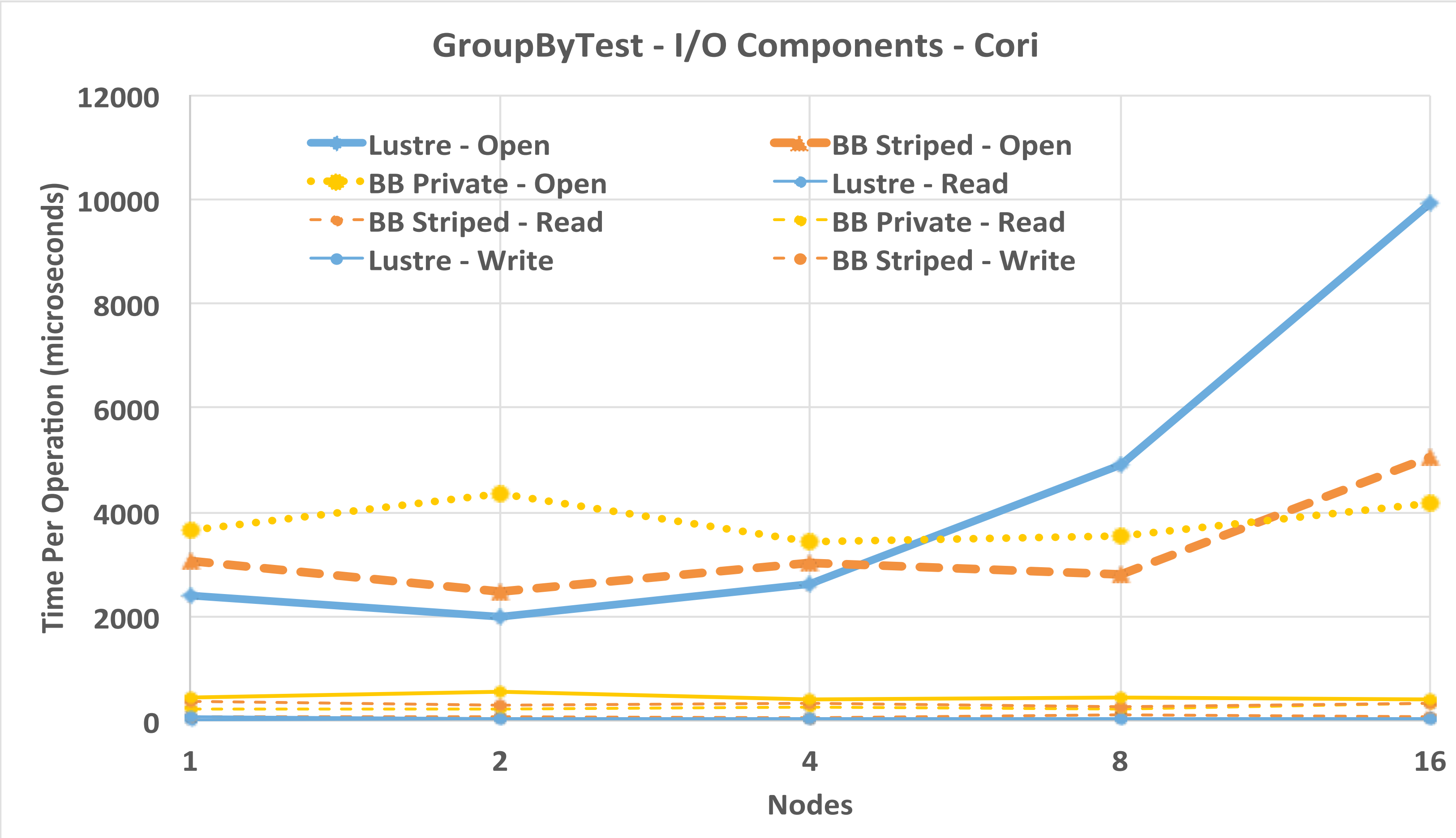
- Comet at SDSC: 2.5GHz Haswell, InfiniBand FDR, 320 GB SSD, 1.5TB memory
- Spark 1.5.0, spark-perf benchmarks

Scaling Spark on Cray XC40

(LUSTRE woes)

I/O Scalability

COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP



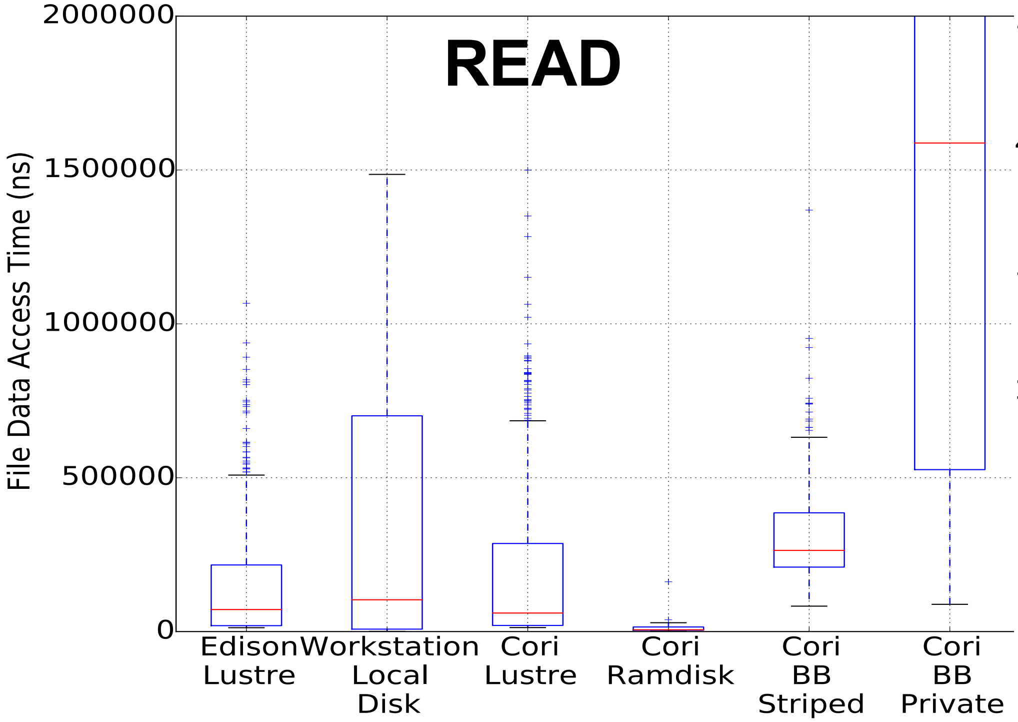
Read/Write scales better than Fopen

Execution is rife with fopen/fclose operations O(cores^2)

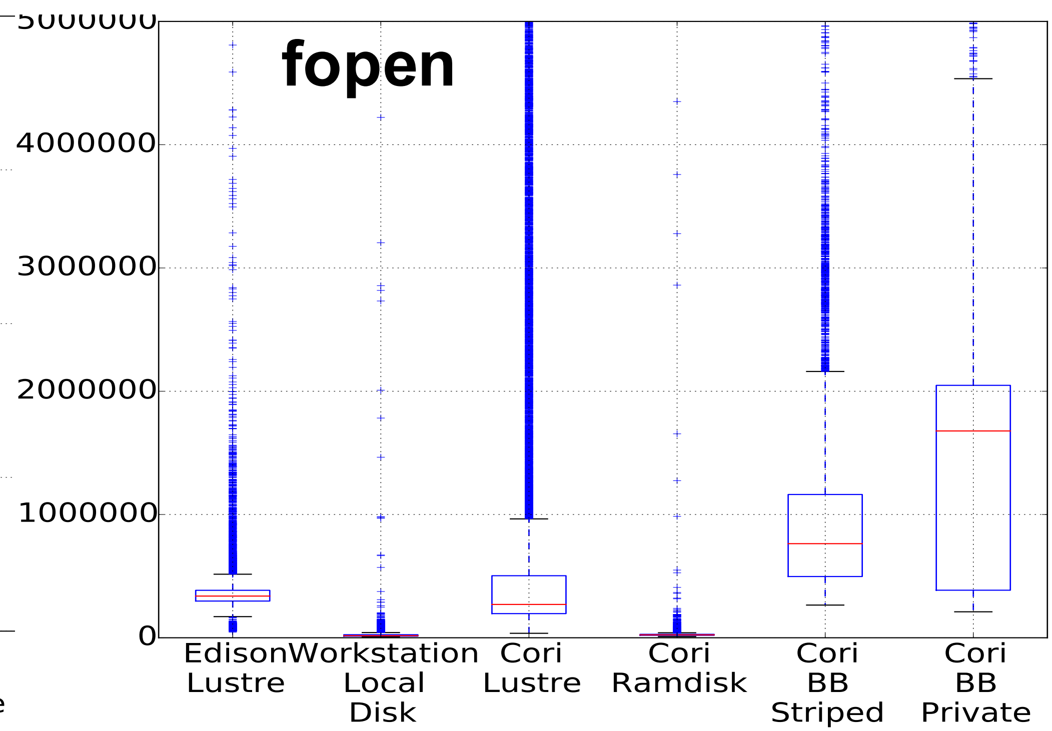
I/O Variability is HIGH

COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP

	Edison Lustre	Workstation Local Disk	Cori Lustre	Cori Ramdisk	Cori BB Striped	Cori BB Private
Mean	0.33	1	0.43	0.01	2.5	4.5
Var	0.07	1	1.4	0.0001	0.3	9.1



	Edison Lustre	Workstation Local Disk	Cori Lustre	Cori Ramdisk	Cori BB Striped	Cori BB Private
Mean	23	1	32	1.2	43	59
Var	14,000	1	34,200	13.5	7,000	8,100



Fopen is a problem:

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

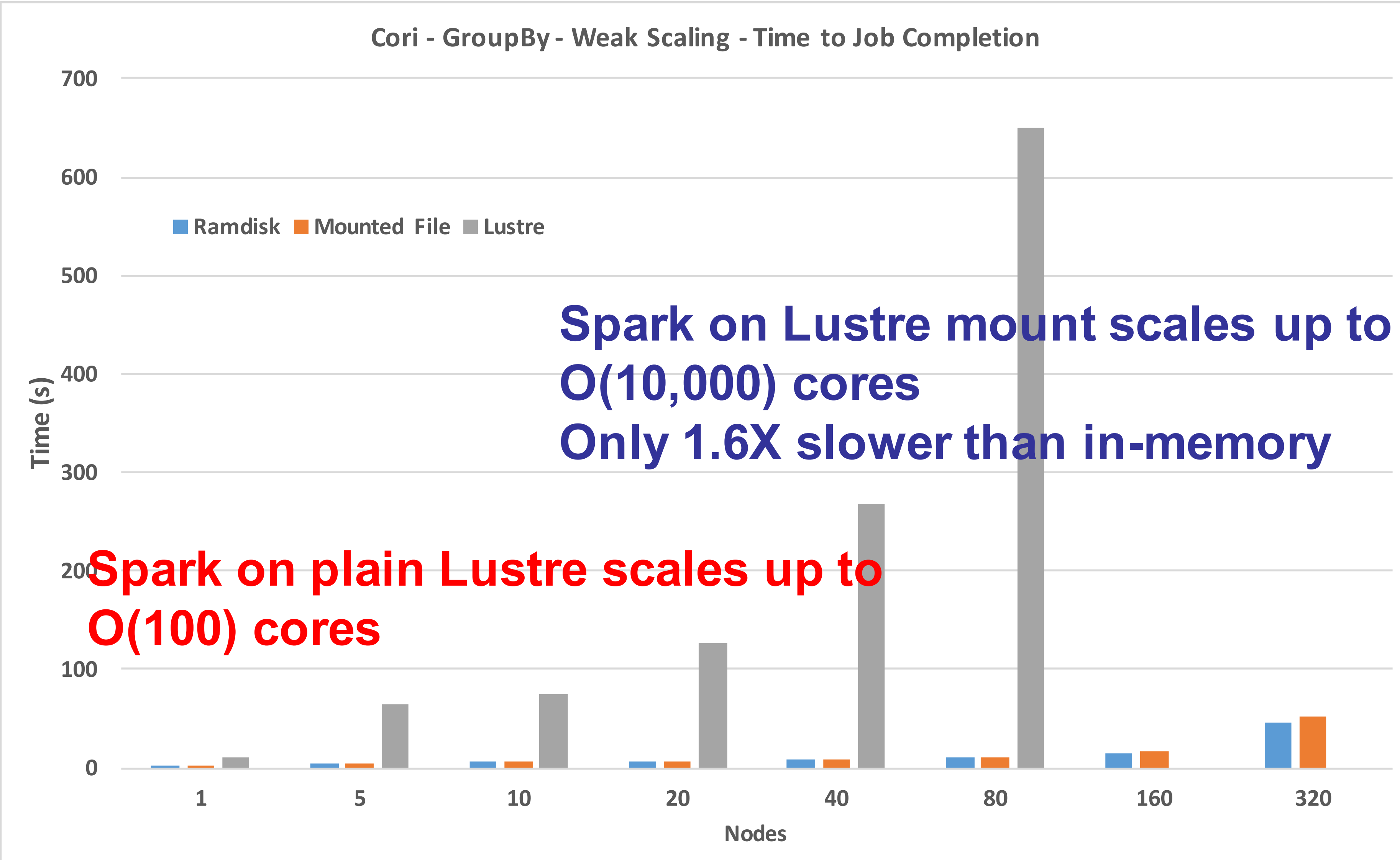
Improving I/O Performance

- **Eliminate file operations that affect the metadata server**
  - Combine files within a node (currently per core combine)
  - Keep files open (cache `fopen`)
  - Use memory mapped local file system `/dev/shm` (no spill)
  - Use file system backed by single Lustre file

- **Partial solutions that need to be used in conjunction**
  - Memory pressure is high is Spark due to resilience and poor garbage collection
  - Fopen() not necessarily from Spark (e.g. Parquet reader)
  - Third party layers not optimized for HPC/Lustre

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

Scalability



File-Backed Filesystems in Shifter

NERSC Shifter

- Lightweight container infrastructure for HPC
- Compatible with Docker images
- Integrated with Slurm scheduler
- Can control mounting of filesystems within container

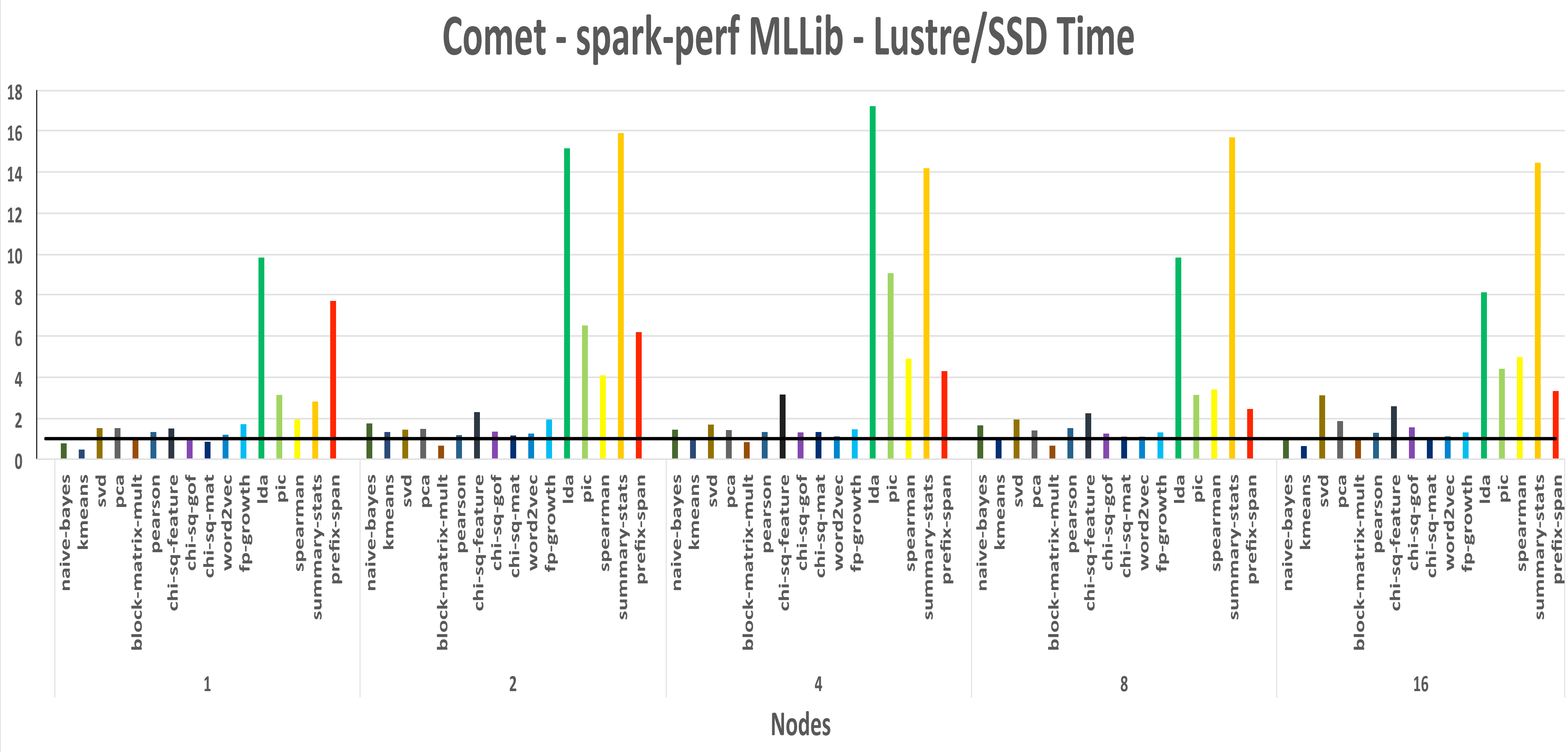
Per-Node Cache

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

Architectural and Software Design Considerations

Global vs. Local Storage

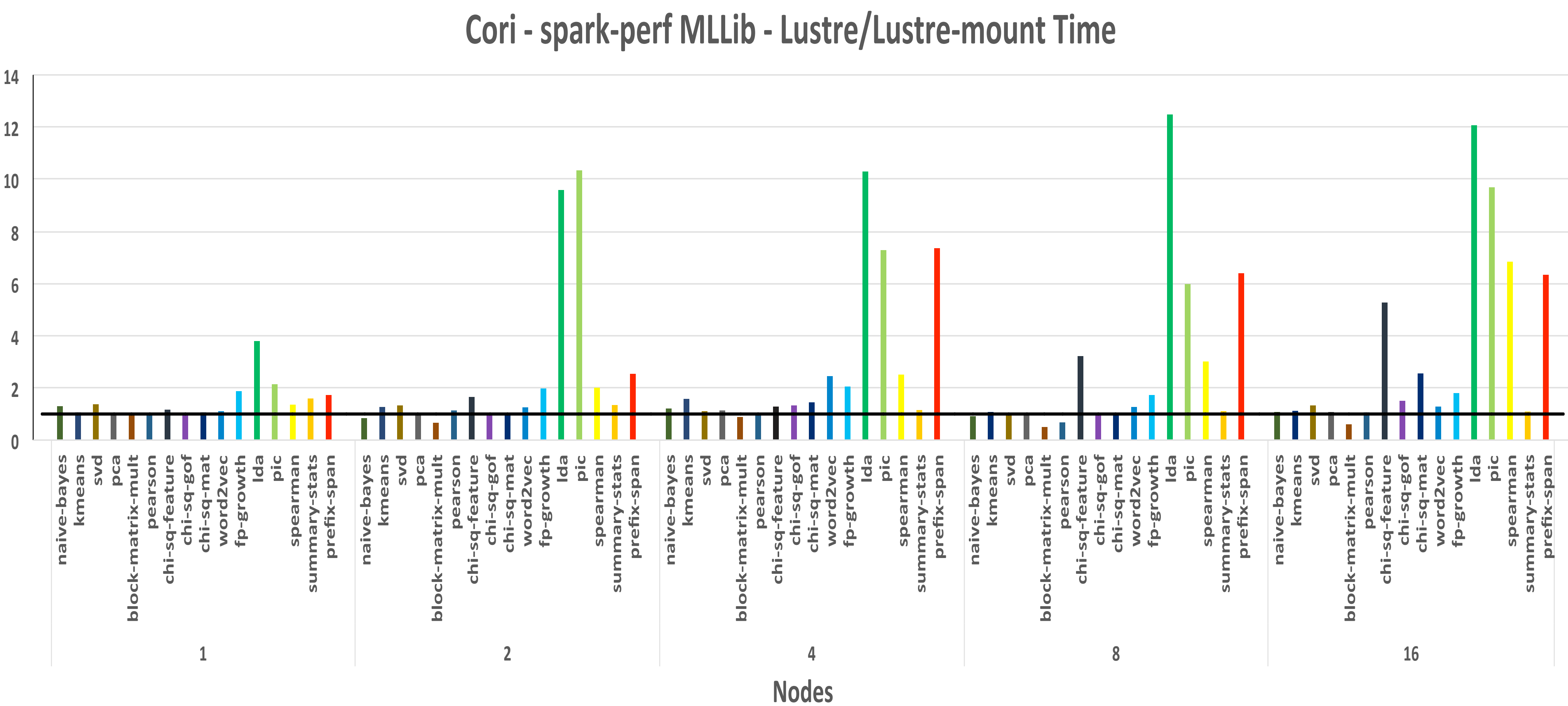
COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP



When the shuffle is not quadratic, Lustre and NAS storage is competitive

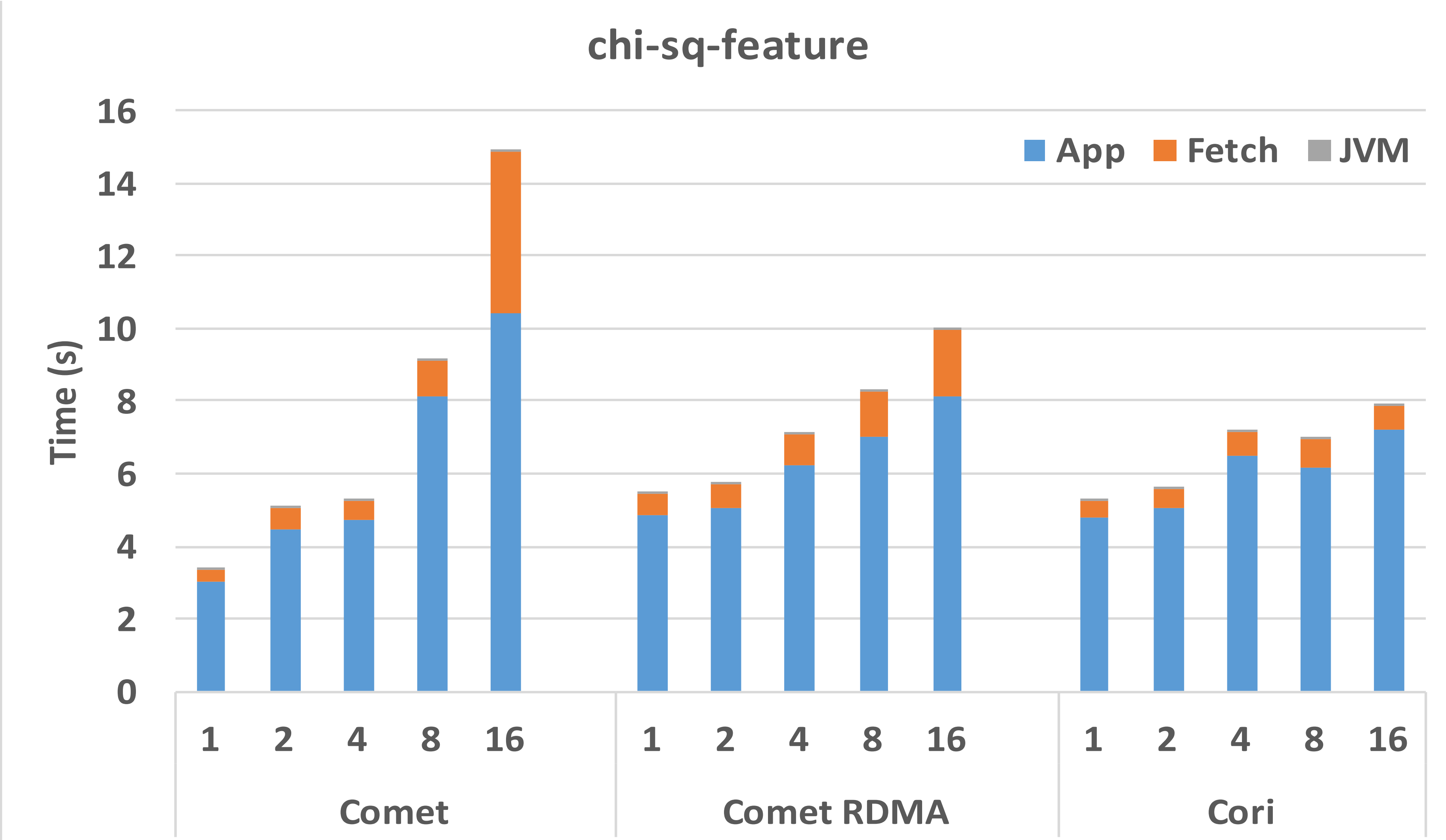
Optimizations can make global storage bearable

COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP



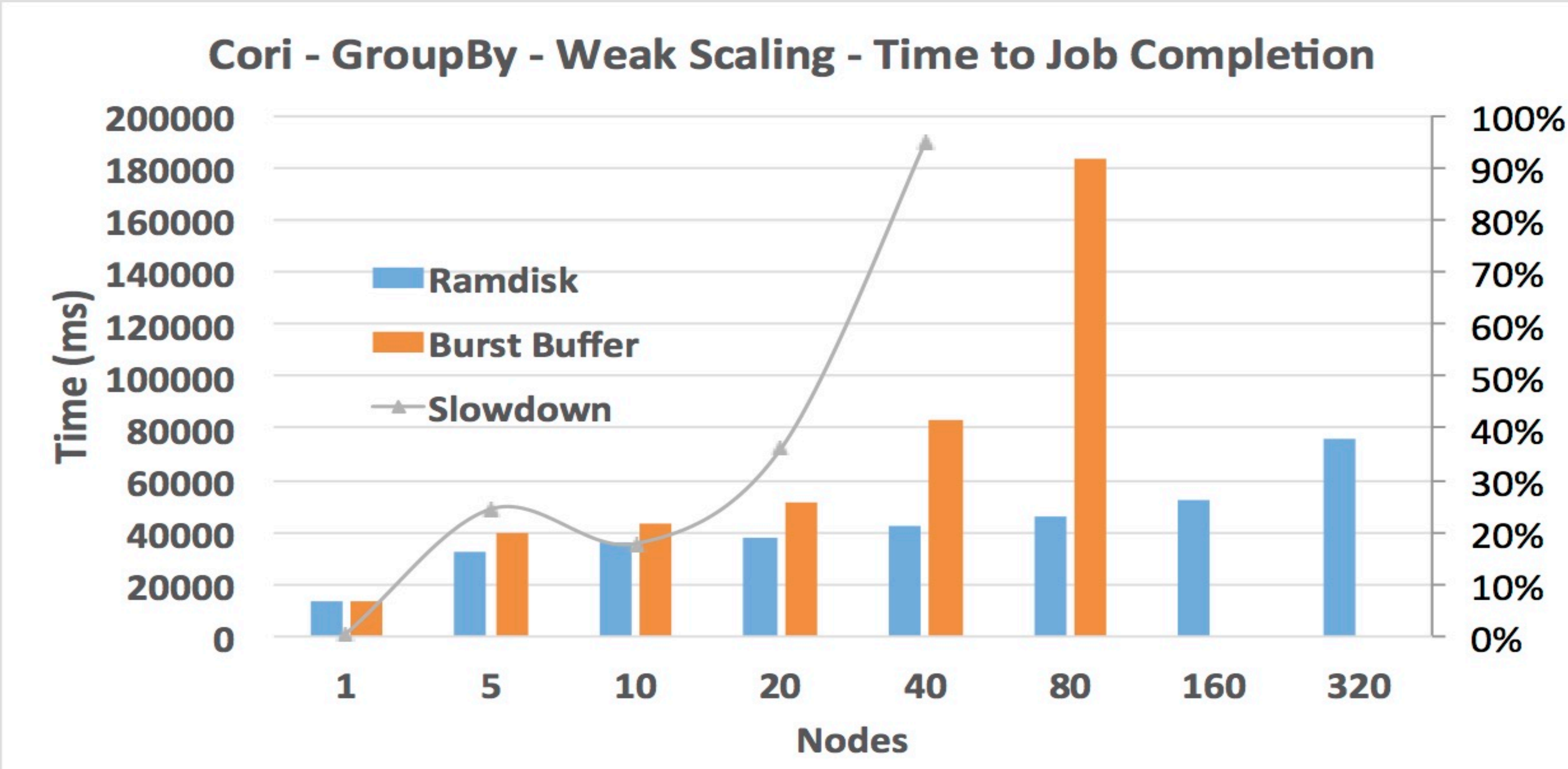
The two figures indicate that most overhead (>10X) is in fopen, not in latency/BW to storage (<2X)

Global Storage matches Local Storage



Cori/TCP is 20% faster than Comet/RDMA on 512 cores

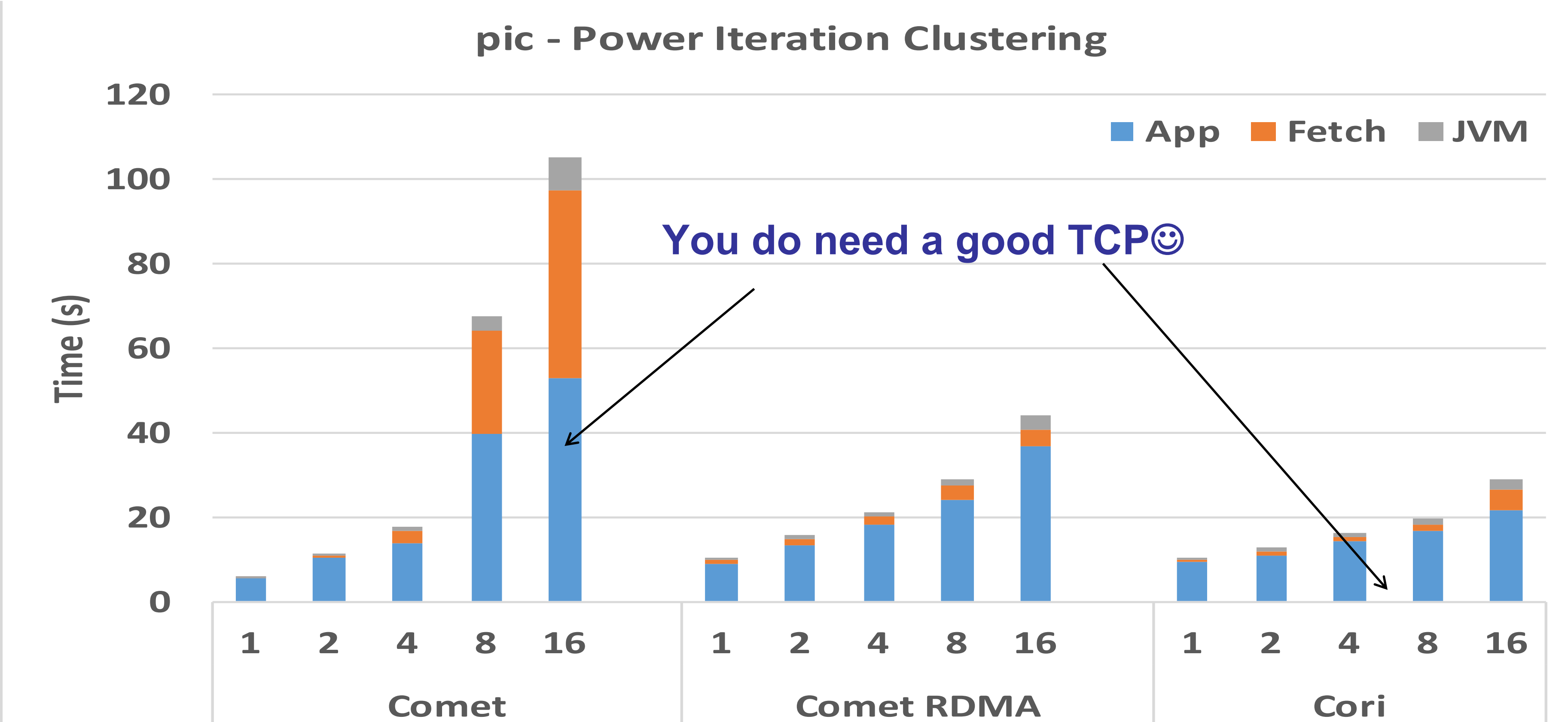
Midlayer storage: Optimizing for the Tail Helps



BB median open is twice slower than Lustre
BB open variance is 5x smaller

BB scales better than standalone Lustre

Competitive Advantage from Network Performance



- ❖ **Benefit of RDMA optimizations is target dependent**
  - **Single node Comet is 50% faster than Cori**
  - **Cori/TCP is 27% faster than Comet/RDMA on 16 nodes**

Better communication → higher availability of compute cores.

Conclusions & Impact

- **Good recipe for tuning Spark on Lustre based systems**
  - We started at O(100) cores scalability
  - Showed O(10,000) cores scalability
- **NERSC and Cray are already using our solutions**
  - Our NERSC/Cray colleagues ran at 50,000 cores
- **Lustre mount released in Shifter**
  - Enjoy☺
- **NAS storage surprisingly close to local SSDs**
- **Competitive advantage from better networks still TBD**

Acknowledgement

COMPUTER LANGUAGES & SYSTEMS SOFTWARE GROUP

Work partially funded by Intel Parallel Computing Center

Thank You!