STORM: Lightning-Fast Resource Management

Eitan Frachtenberg, **Fabrizio Petrini**, Juan Fernandez, Scott Pakin, and Salvador Coll

`fabrizio@lanl.gov`

`http://www.c3.lanl.gov/~fabrizio`

Performance and Architecture Laboratory

CCS-3 Modeling, Algorithms, and Informatics Group

Los Alamos National Laboratory

Vision

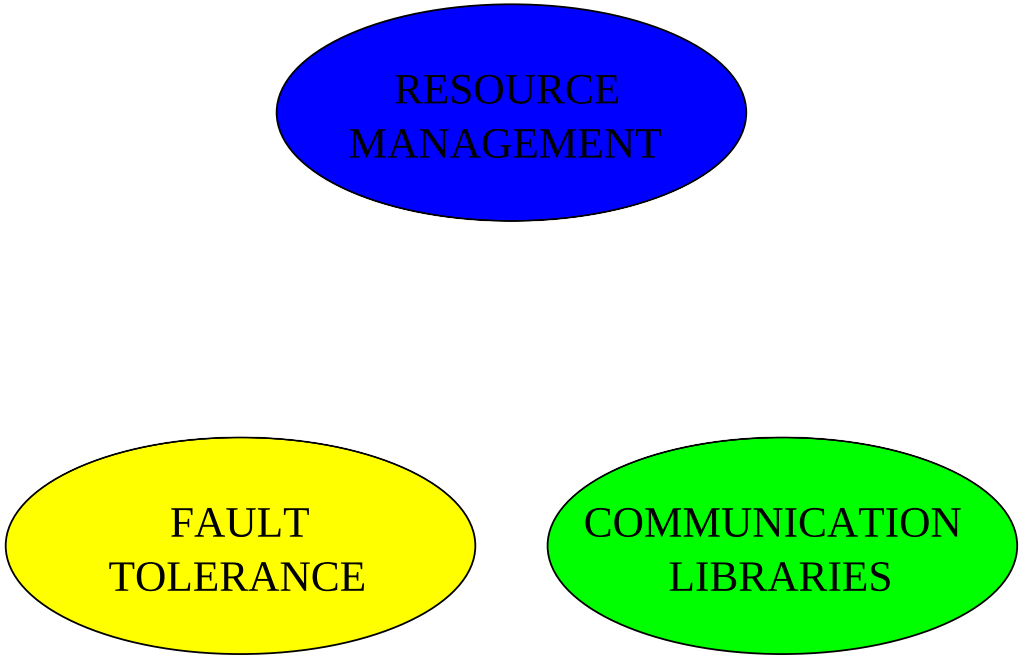
- More effective use of cluster resources
  - Lower response time
  - Higher throughput
- *Transparent* fault tolerance
  - No application modifications

Los Alamos
NATIONAL LABORATORY

Vision

- Buffered Coscheduling (BCS) is a new methodology to:
  - Improve system responsiveness and utilization,
  - Tolerate inefficient programs (communication and load imbalance),
  - Implement fault-tolerance

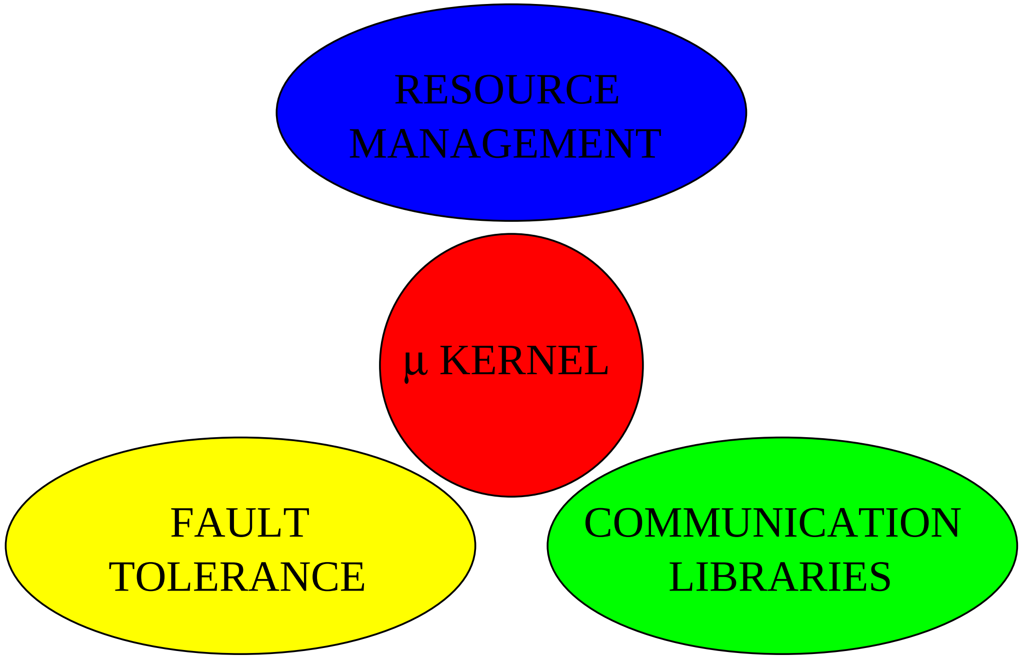
Vision



- Buffered Coscheduling tries to achieve these goals by greatly simplifying the system software (resource management, communication libraries and fault-tolerance)

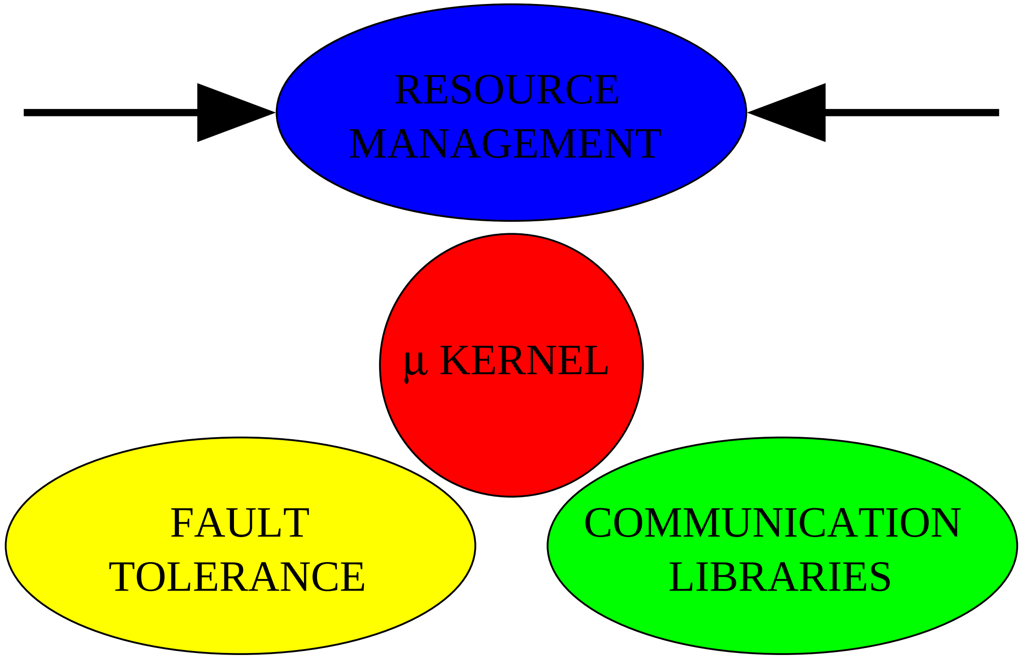
Los Alamos NATIONAL LABORATORY

Vision



- Buffered Coscheduling implements resource management, communication libraries and fault-tolerance on top of a common microkernel

Los Alamos NATIONAL LABORATORY



- In this talk we will focus on STORM, a resource manager implemented on top of the Buffered Coscheduling microkernel

Los Alamos
NATIONAL LABORATORY

STORM (Scalable TOol for Resource Management)

- Goals
  - Portability
  - High performance resource management
  - Research tool to investigate new job scheduling algorithms
- Key innovation: software architecture that enables resource management to exploit low-level network features

Los Alamos
NATIONAL LABORATORY

Outline

- Overview of resource management
- STORM architecture
- Implementation
- Performance evaluation
- Scalability analysis

Los Alamos
NATIONAL LABORATORY

Resource Management

- Resource allocation for parallel jobs
- Job launch and termination
- Cluster management
- Monitoring and debugging

Characteristics of Desktops versus Clusters

Characteristic	Desktop	Cluster
Mean time between user-visible failures	Years	Days down to hours
Scheduling	Timeshared	Batch queued or gang scheduled with large quanta
Job-launching speed	< 1 second	Arbitrarily long (batch) or many seconds (gang scheduled)

Los Alamos
NATIONAL LABORATORY

State of the art in Resource Management

Resource Managers (e.g., PBS, LSF, RMS, LoadLeveler, Maui) are typically implemented using

- TCP/IP
  - Favors portability over performance
- Non-scalable algorithms for the distribution/collection of data and control messages
  - Favors development time over performance
- Performance not important for small clusters, but crucial for large clusters $\rightarrow$ need fast and scalable resource management

Los Alamos NATIONAL LABORATORY

STORM implementation structure

Layer	Examples
STORM functions	heartbeat, file transfer, termination detection
(STORM helper functions)	flow control, queue management
STORM mechanisms	XFER-AND-SIGNAL, TEST-EVENT, COMPARE-AND-WRITE
Network primitives	remote DMA, network conditionals, event signaling, . . .

STORM mechanisms

STORM is based on only three mechanisms

XFER-AND-SIGNAL Transfer (PUT) a block of data from local memory to the global memory of a set of nodes (possibly a single node).

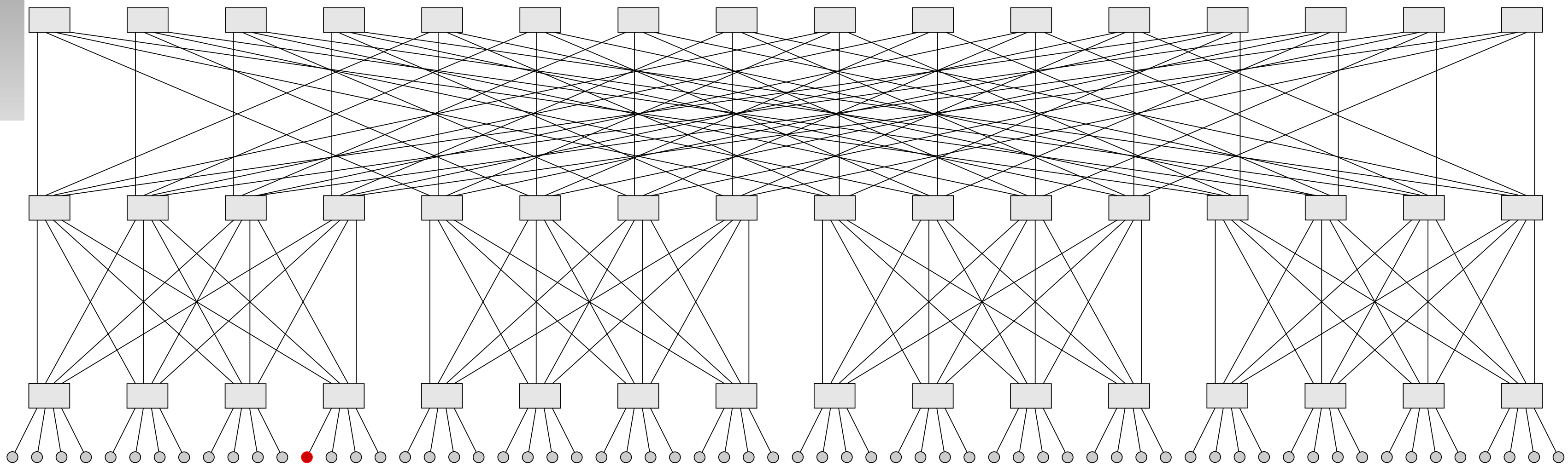
TEST-EVENT Local synchronization

COMPARE-AND-WRITE Global query with boolean reduction

Efficient and scalable implementation of these mechanisms $\rightarrow$ STORM is scalable

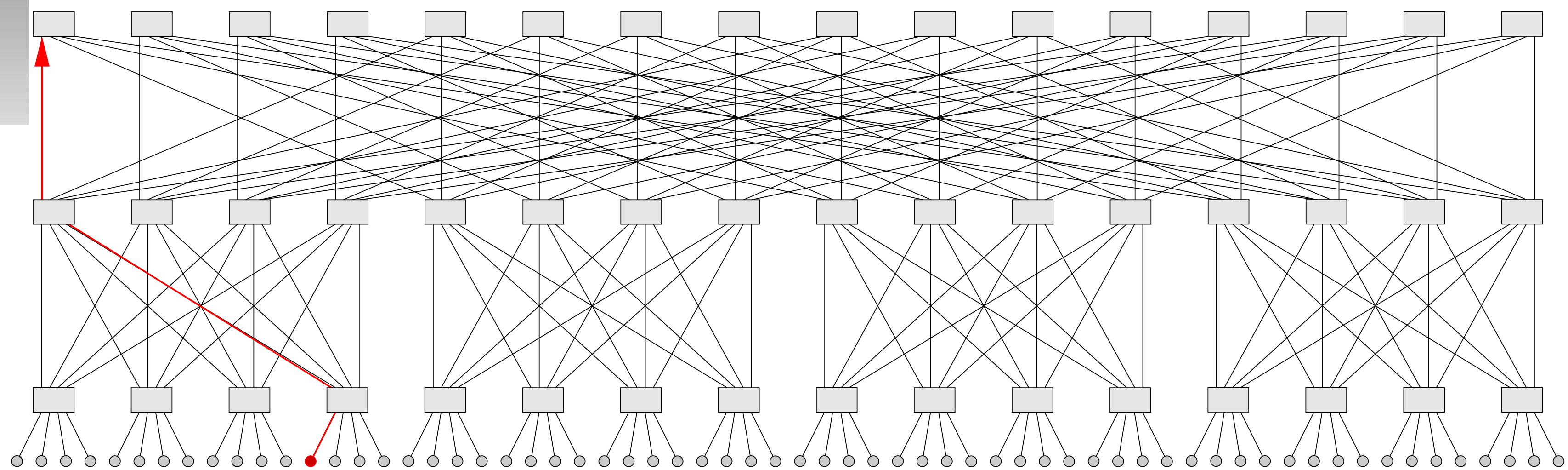
Hardware support for

XFER-AND-SIGNAL



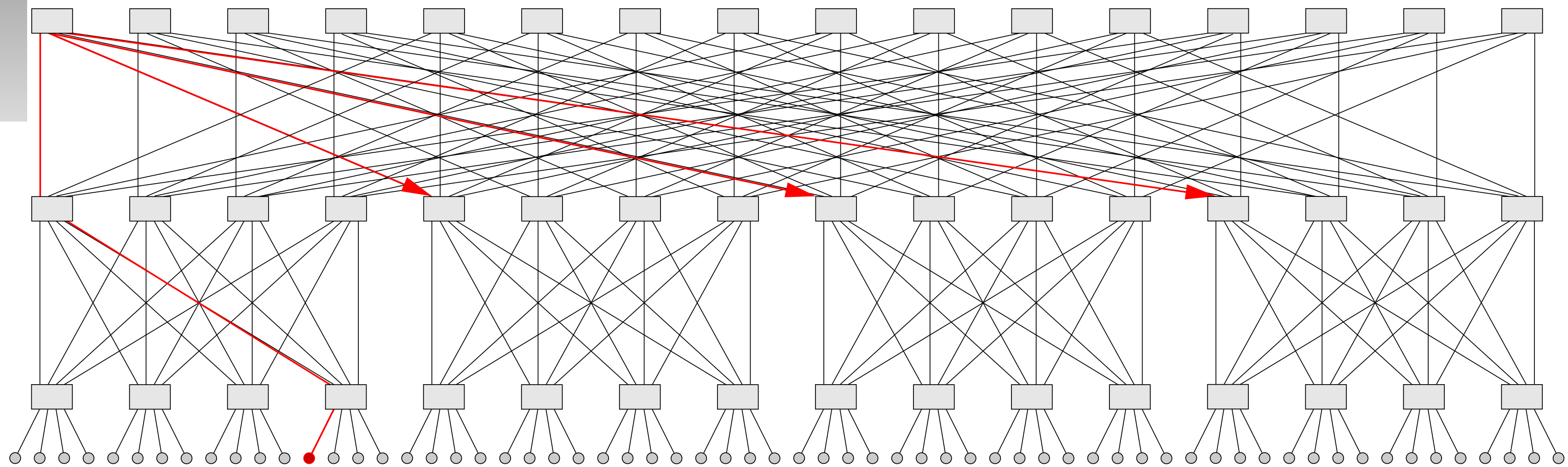
- XFER-AND-SIGNAL transfers multicast a block of data to a group of nodes
- The multicast can be executed in HW

Hardware support for
XFER-AND-SIGNAL



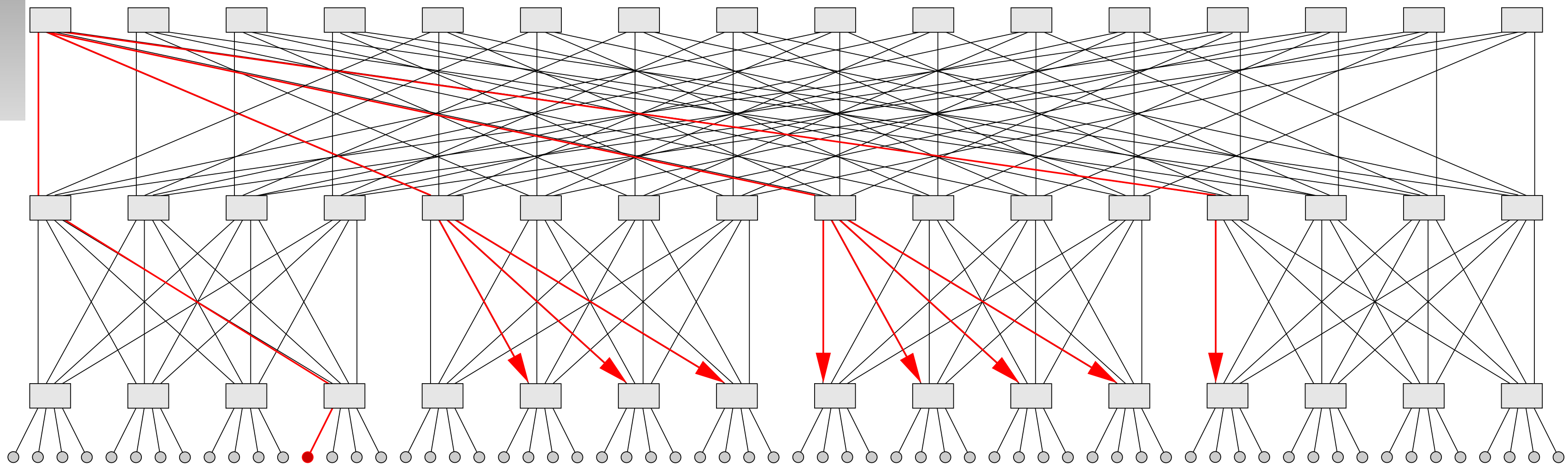
- The packet is routed through a root node during the ascending phase
- The flow-through latency of each switch is only a few tens of nanoseconds

Hardware support for XFER-AND-SIGNAL



- The packet reaches the set of destinations during the descending phase

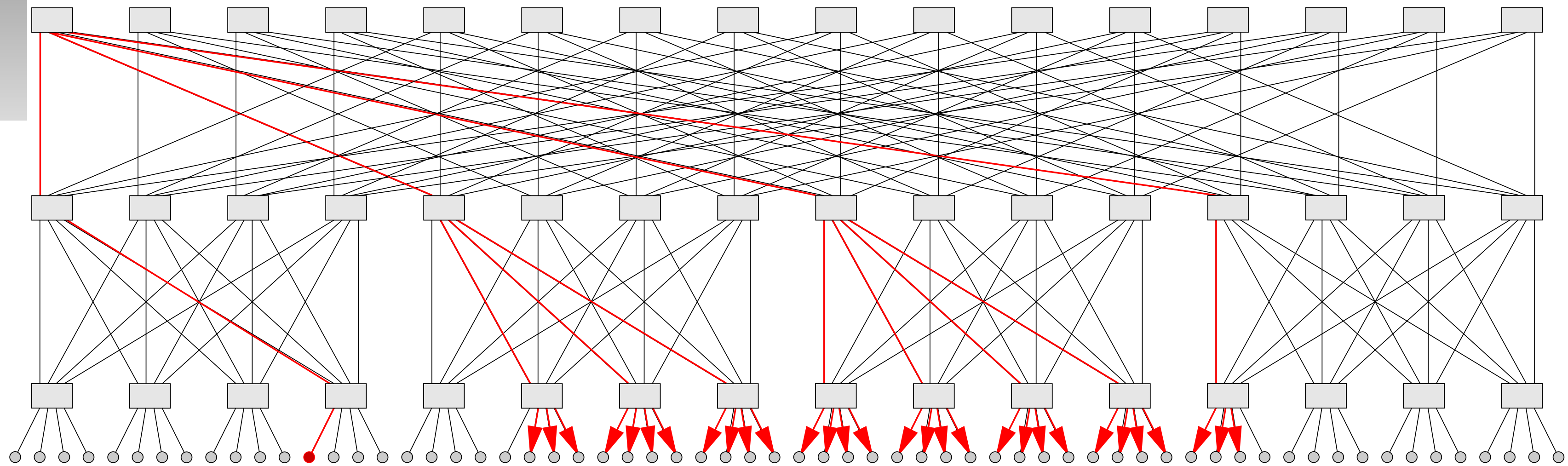
Hardware support for XFER-AND-SIGNAL



- The packet reaches the set of destinations during the descending phase

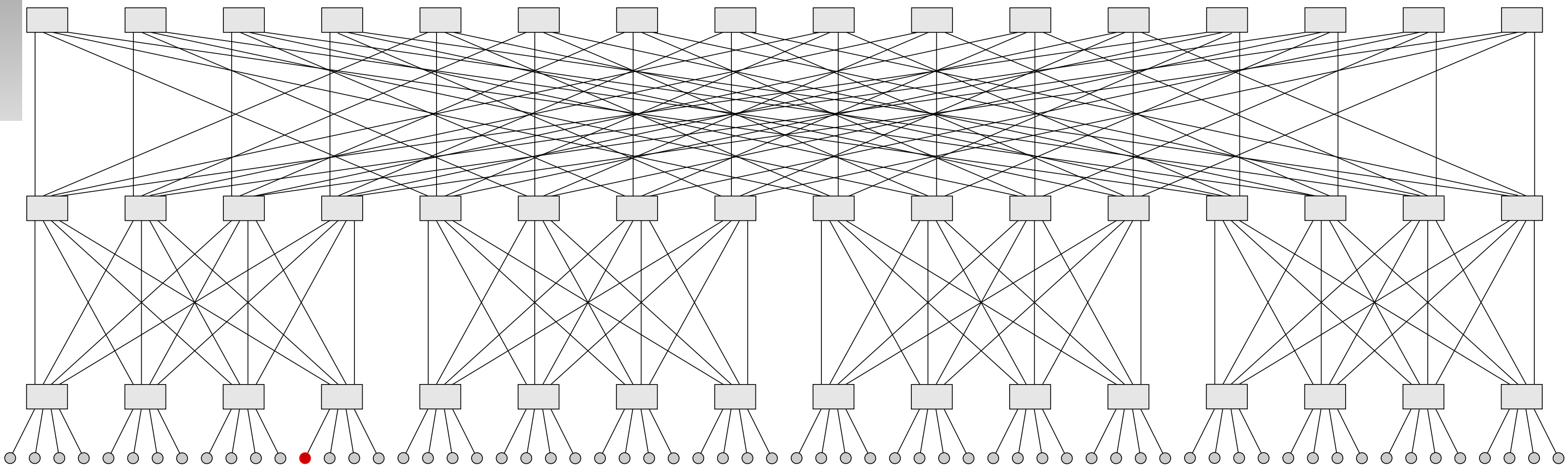
Hardware support for

XFER-AND-SIGNAL



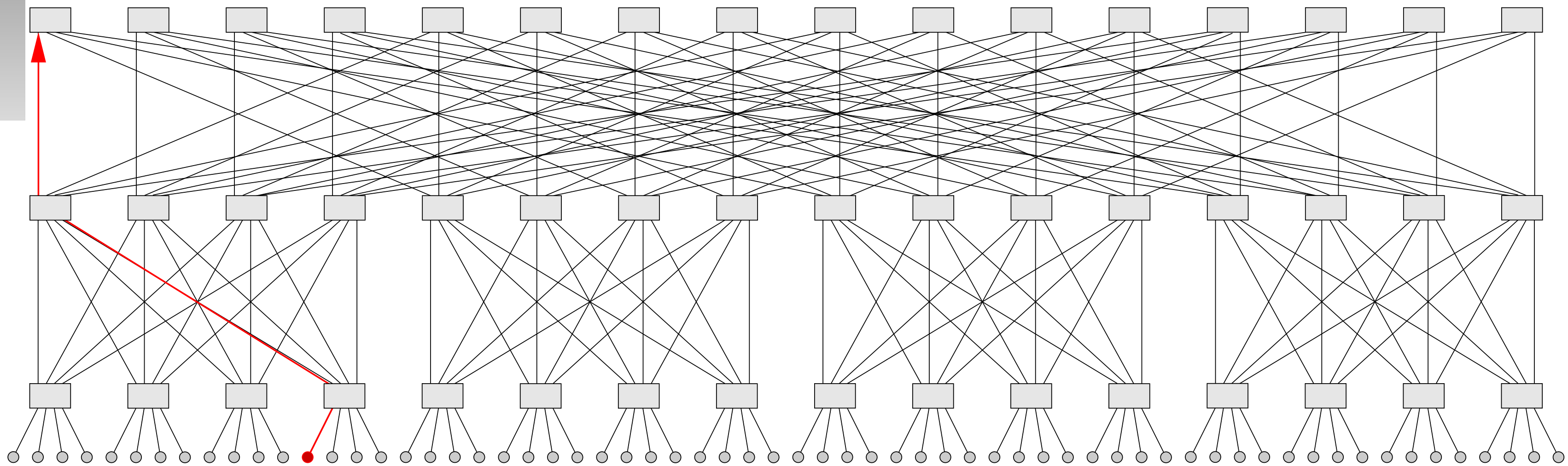
- The packet reaches the set of destinations during the descending phase

Hardware support for COMPARE-AND-WRITE



- COMPARE-AND-WRITE executes a binary query on a set of nodes

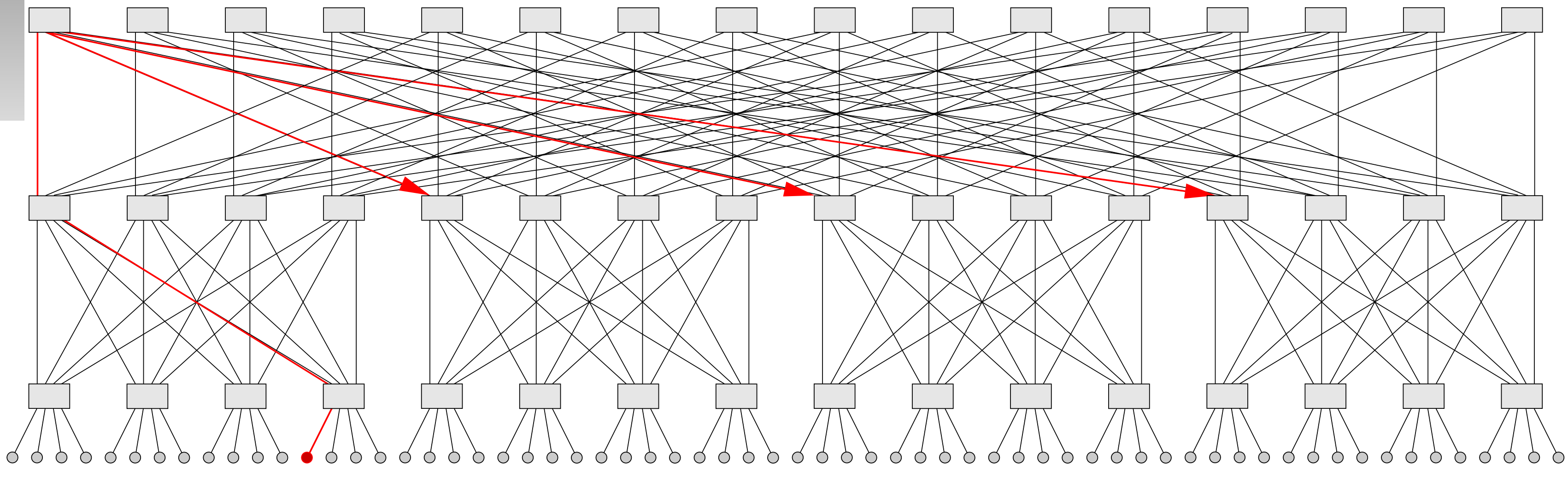
Hardware support for COMPARE-AND-WRITE



- COMPARE-AND-WRITE executes a binary query on a set of nodes

Los Alamos
NATIONAL LABORATORY

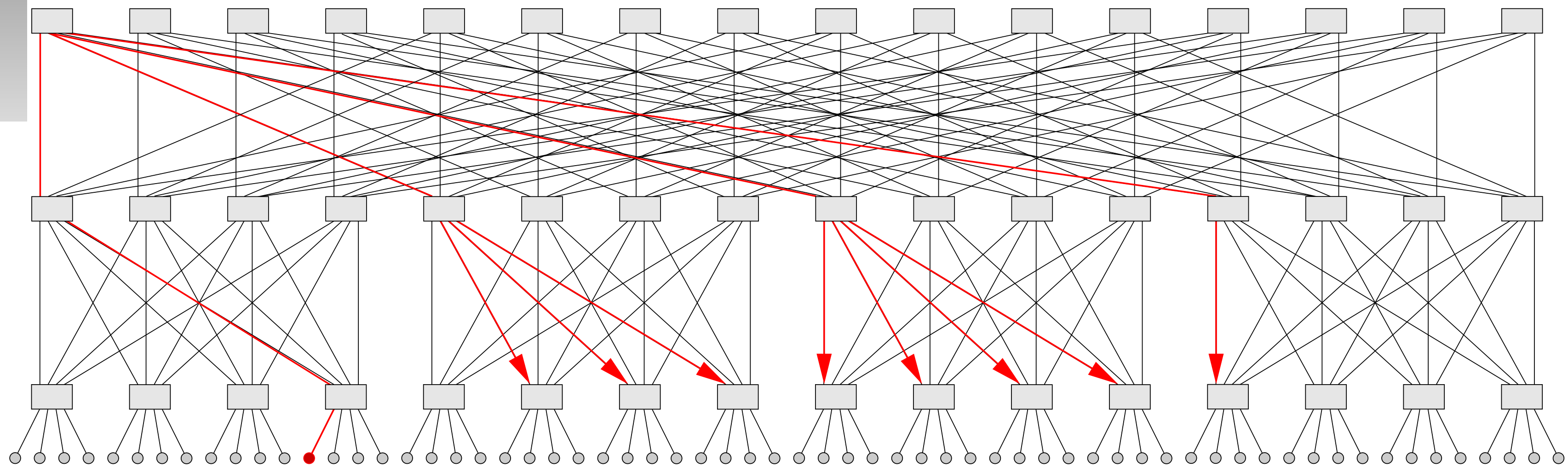
Hardware support for COMPARE-AND-WRITE



- COMPARE-AND-WRITE executes a binary query on a set of nodes

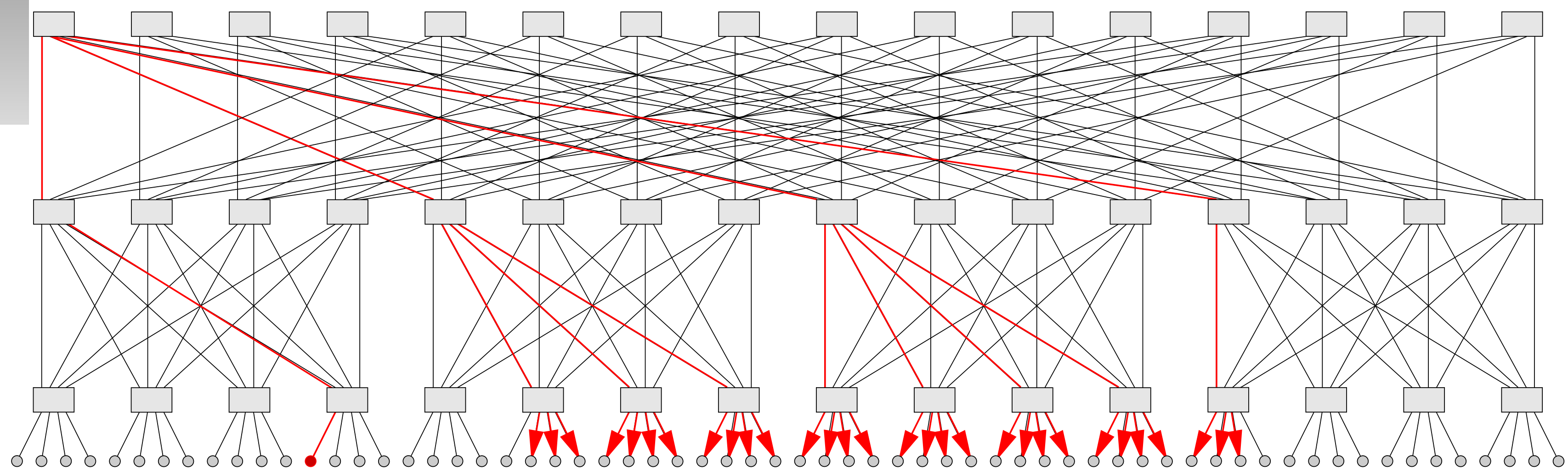
Los Alamos NATIONAL LABORATORY

Hardware support for
COMPARE-AND-WRITE



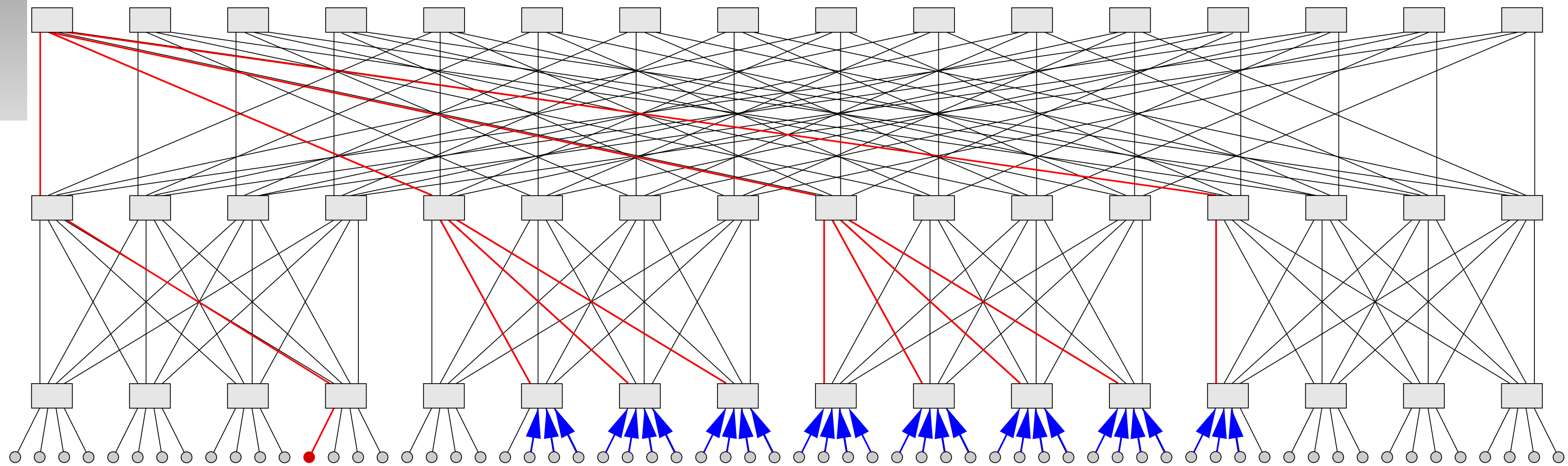
- COMPARE-AND-WRITE executes a binary query on a set of nodes

Hardware support for COMPARE-AND-WRITE



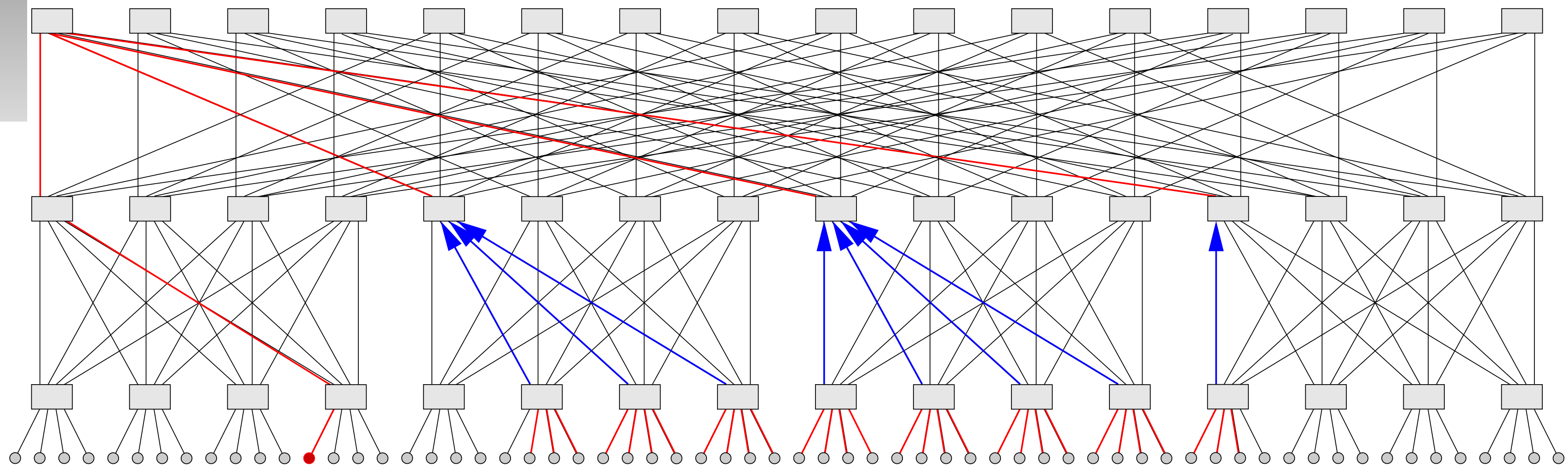
- COMPARE-AND-WRITE executes a binary query on a set of nodes

Hardware support for
COMPARE-AND-WRITE



- The results of the global query are combined on the way up
- The "worst" result wins: *Yes* if all the nodes send a positive ack, *No* otherwise

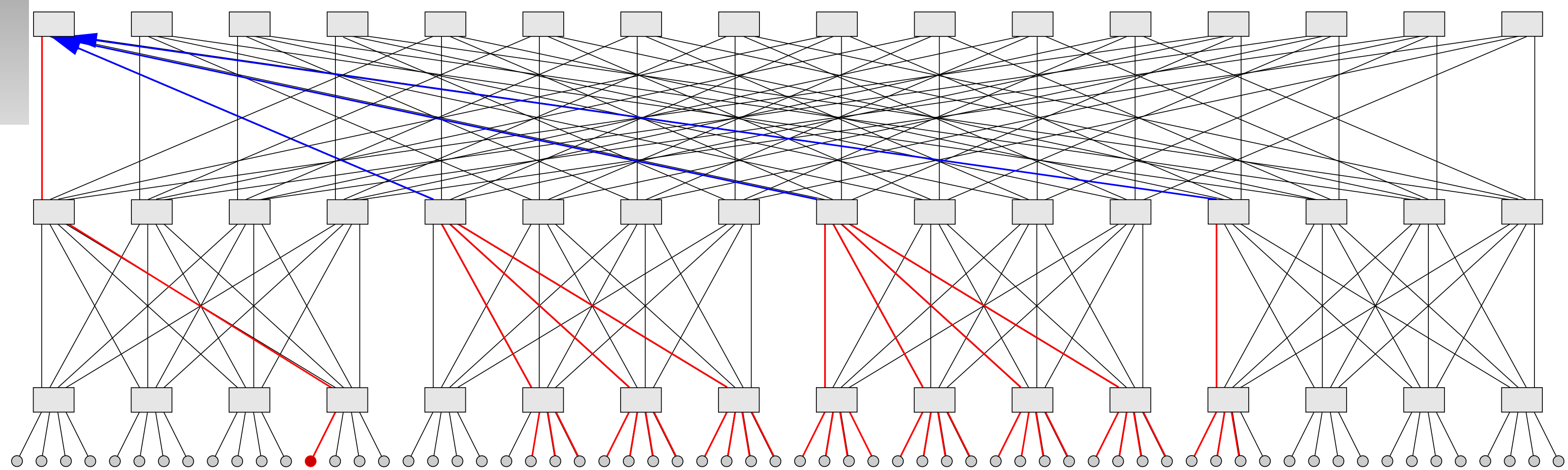
Hardware support for
COMPARE-AND-WRITE



- The results of the global query are combined on the way up
- The “worst” result wins: *Yes* if all the nodes send a positive ack, *No* otherwise

Los Alamos
NATIONAL LABORATORY

Hardware support for COMPARE-AND-WRITE

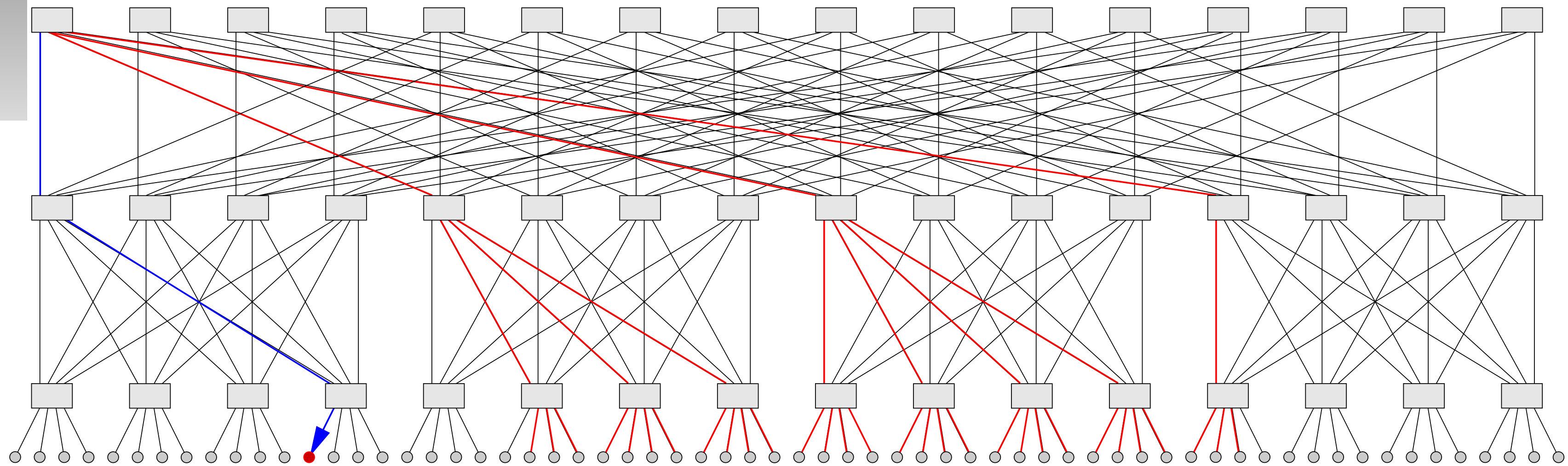


- The results of the global query are combined on the way up
- The “worst” result wins: *Yes* if all the nodes send a positive ack, *No* otherwise

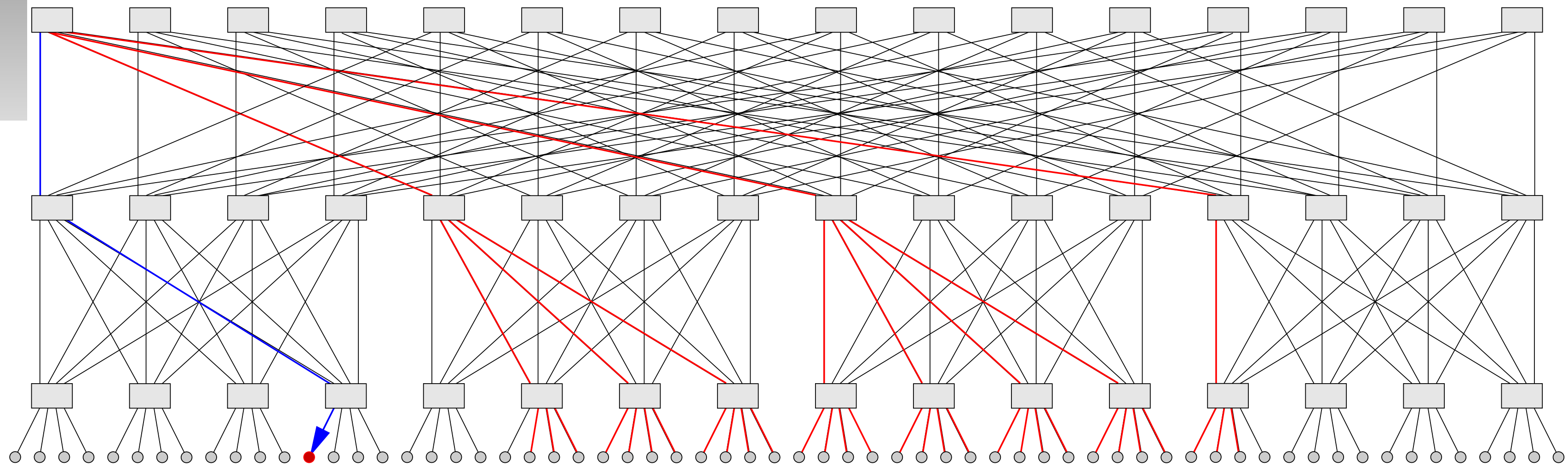
Los Alamos
NATIONAL LABORATORY

Hardware support for

COMPARE-AND-WRITE

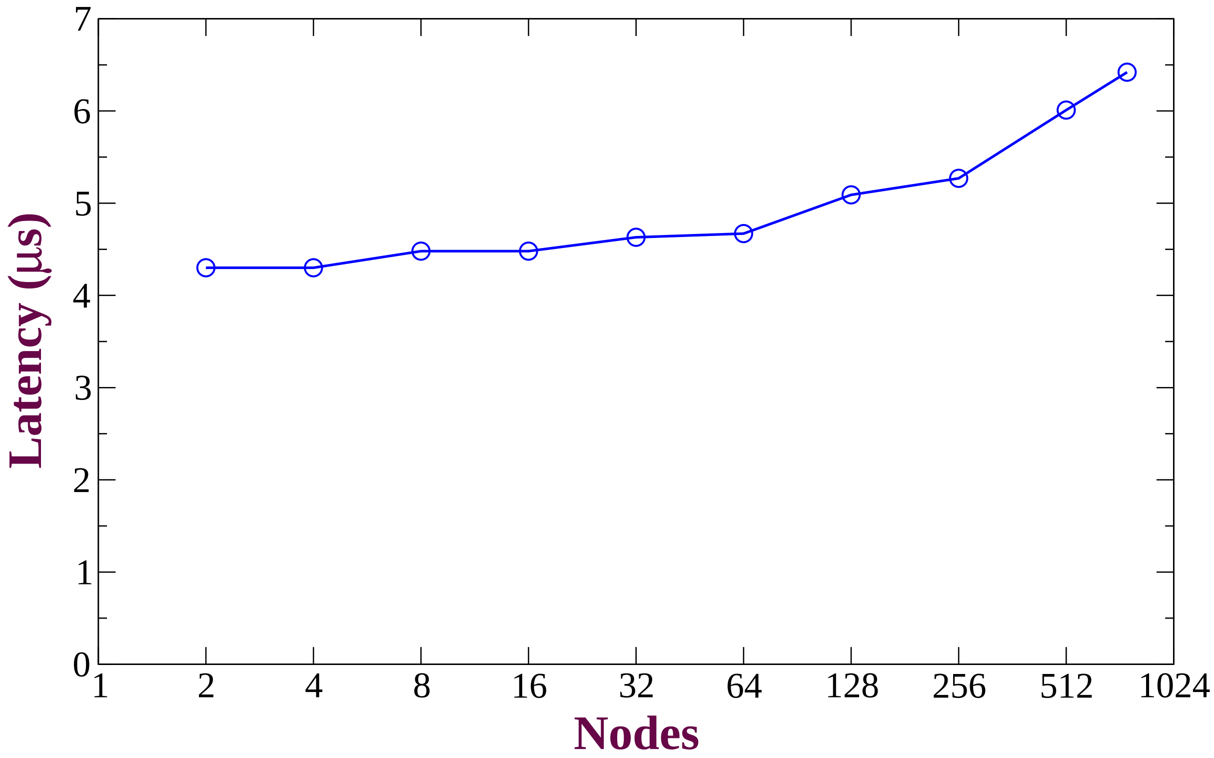


Hardware support for COMPARE-AND-WRITE



The STORM mechanisms XFER-AND-SIGNAL and COMPARE-AND-WRITE can be easily and efficiently implemented on top of the hardware broadcast.

Scalability of the STORM Mechanisms



- COMPARE-AND-WRITE scales efficiently on Lemieux, Pittsburgh Supercomputing Center. Less than 10 μs on 768 nodes/3072 processors

Los Alamos NATIONAL LABORATORY

Portability of the STORM mechanisms

Network	COMPARE-AND-WRITE (μs)	XFER-AND-SIGNAL (MB/s)
Gigabit Ethernet	$46 \log n$	Unknown
Myrinet	$20 \log n$	$\sim 15n$
Infiniband	$20 \log n$	Unknown
QsNET	< 10	$> 150n$
BlueGene/L	< 2	$700n$

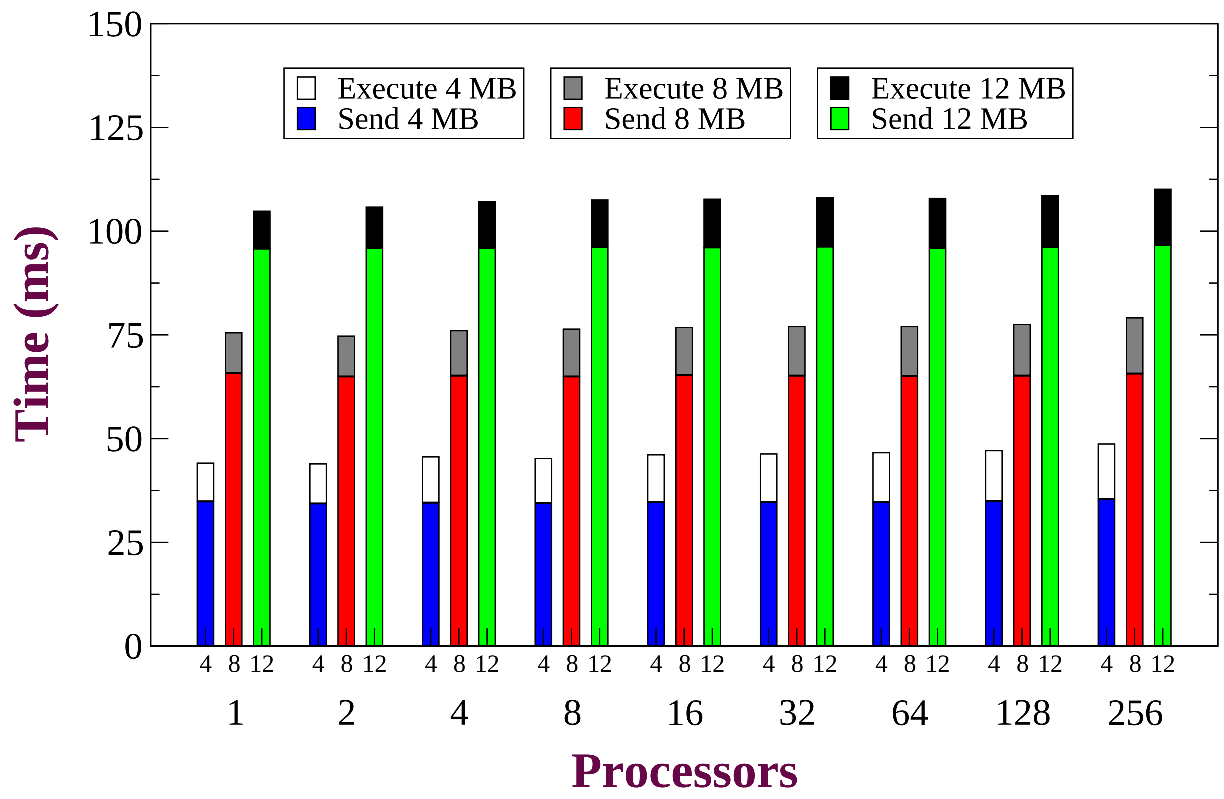
Los Alamos NATIONAL LABORATORY

Experimental Results

- Setup
  - 64 nodes/256 processors ES40 Alphaserver cluster
  - 2 independent rails of Quadrics
  - Linux 2.4.3
  - Files are placed in a RAM disk, in order to avoid I/O bottlenecks
- Experiments
  - Job Launching
  - Job Scheduling

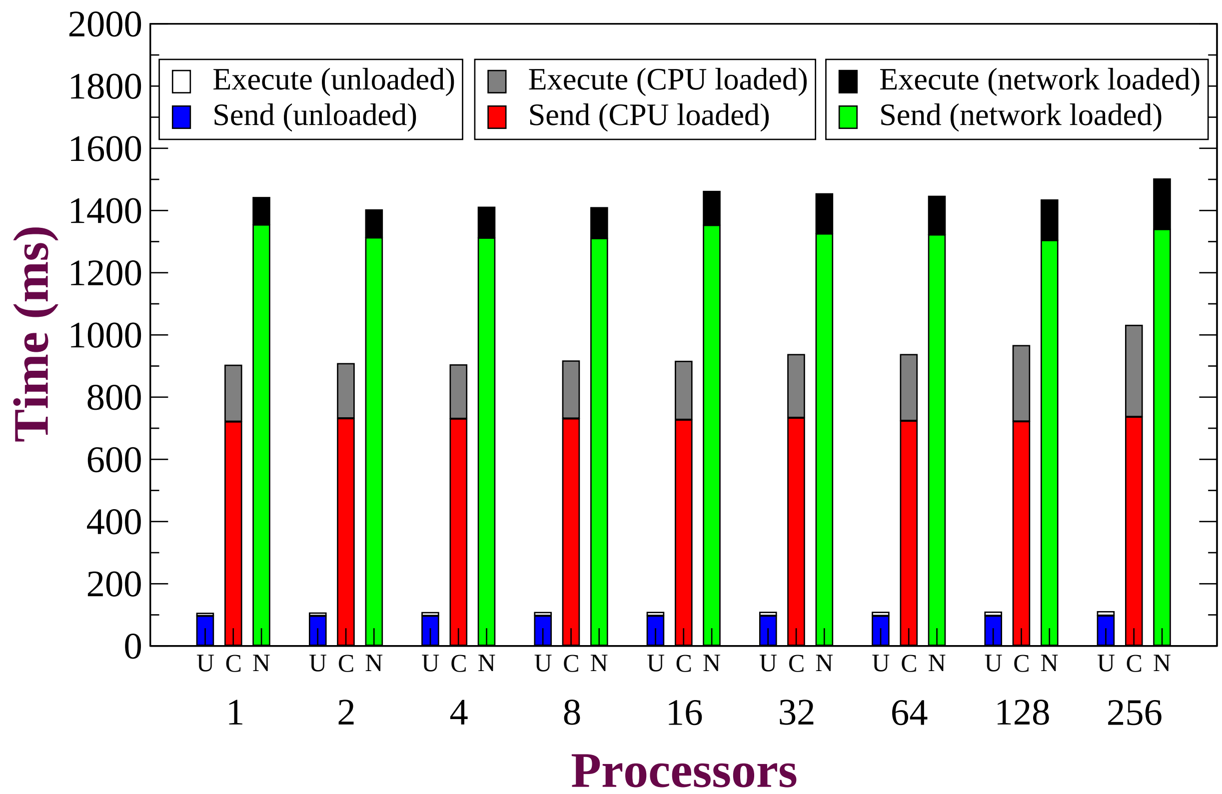
Los Alamos NATIONAL LABORATORY

Launch times (unloaded system)



- The launch time is essentially constant when we increase the number of processors $\rightarrow$ STORM is highly scalable

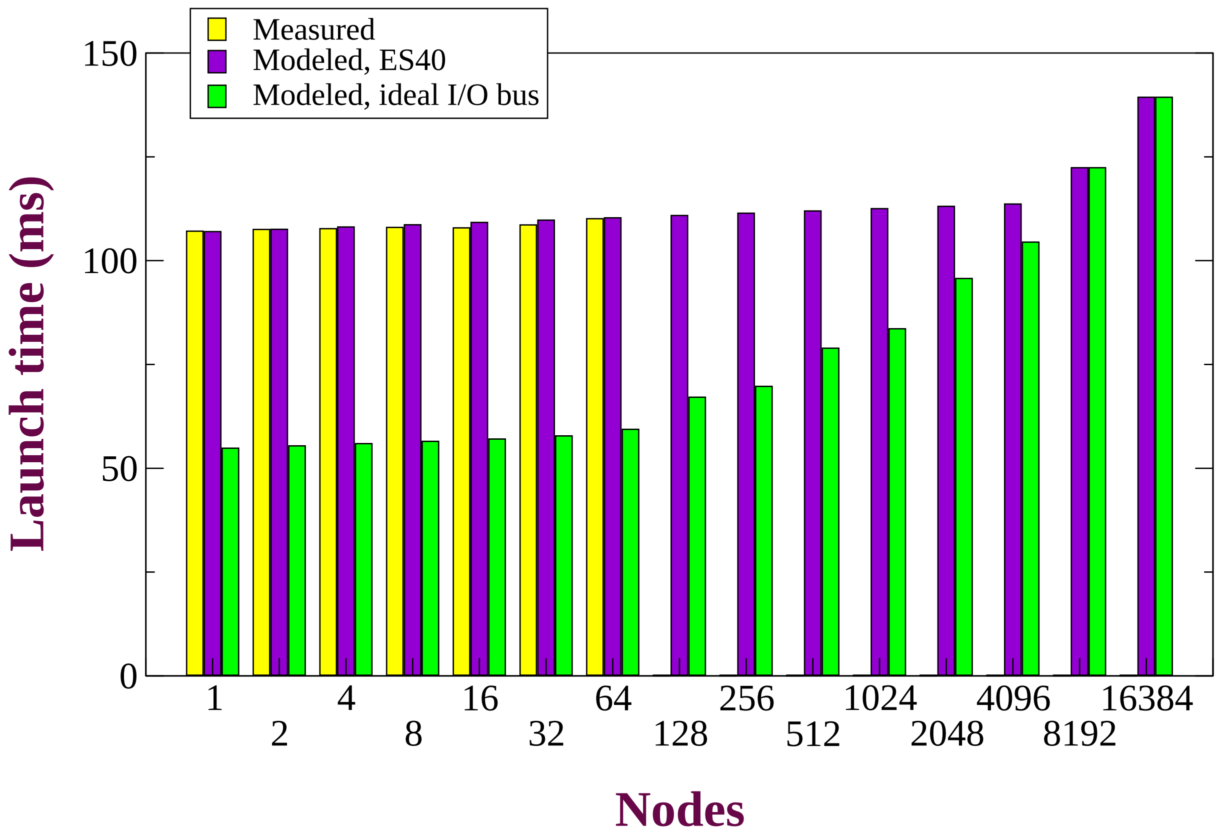
Launch times (loaded system, 12MB executable)



- Launch time is more sensitive to network load rather than CPU load
- In the worst-case scenario it still takes only 1.5 seconds to launch a 12 MB file on 256 processors

Los Alamos NATIONAL LABORATORY

Measured and estimated launch times



- The model shows that in an ES40-based Alphaserver a 12 MB binary can be launched in only 135 ms on 16,384 nodes

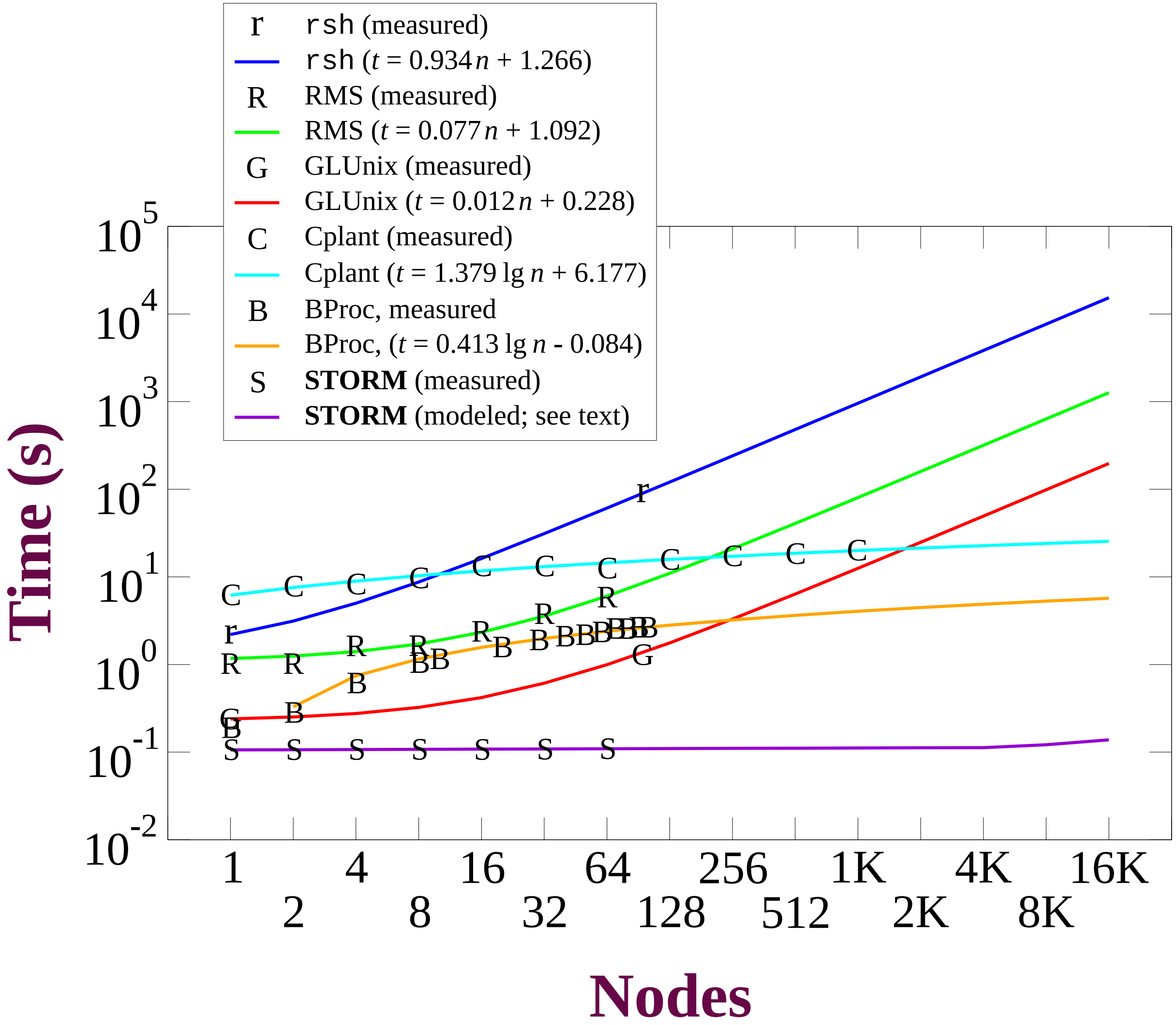
Measured and predicted performance of existing job launchers

We compare the job launching performance of STORM with

- rsh
- RMS
- GLUnix
- Cplant
- Bproc

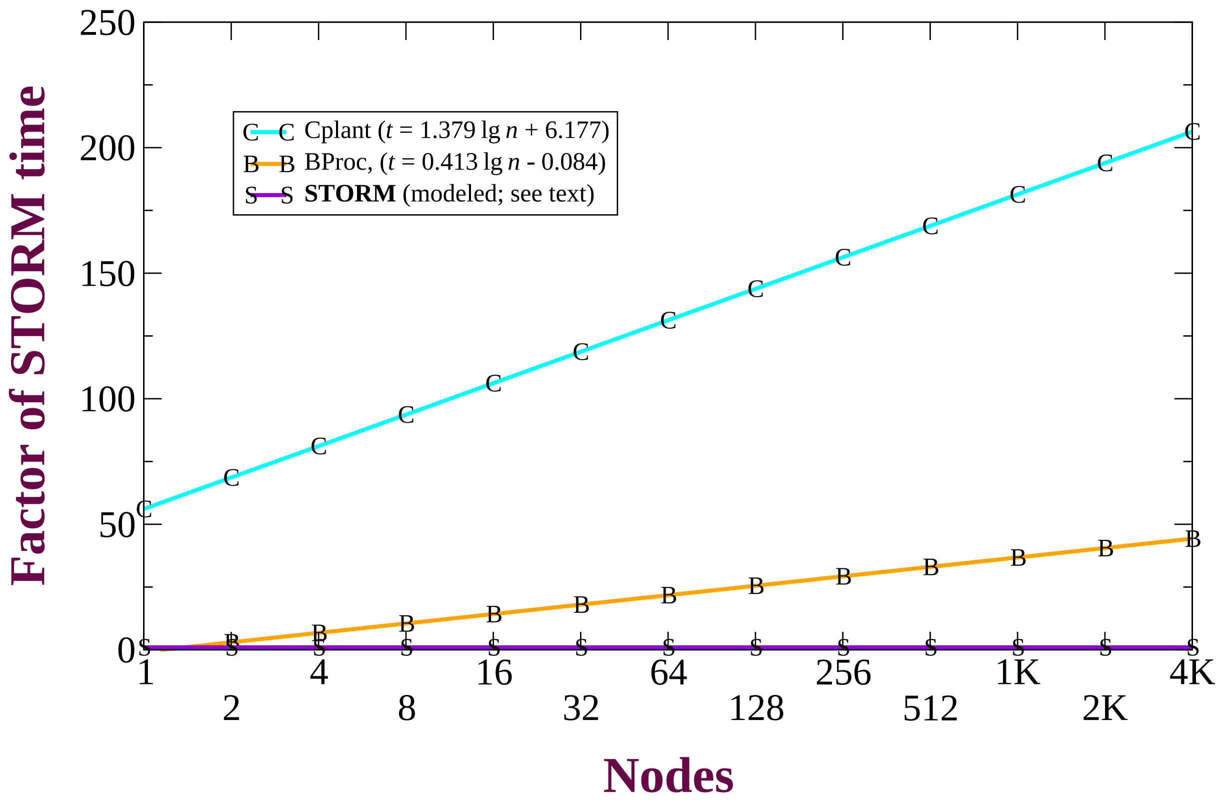
Los Alamos
NATIONAL LABORATORY

Measured and predicted performance of existing job launchers



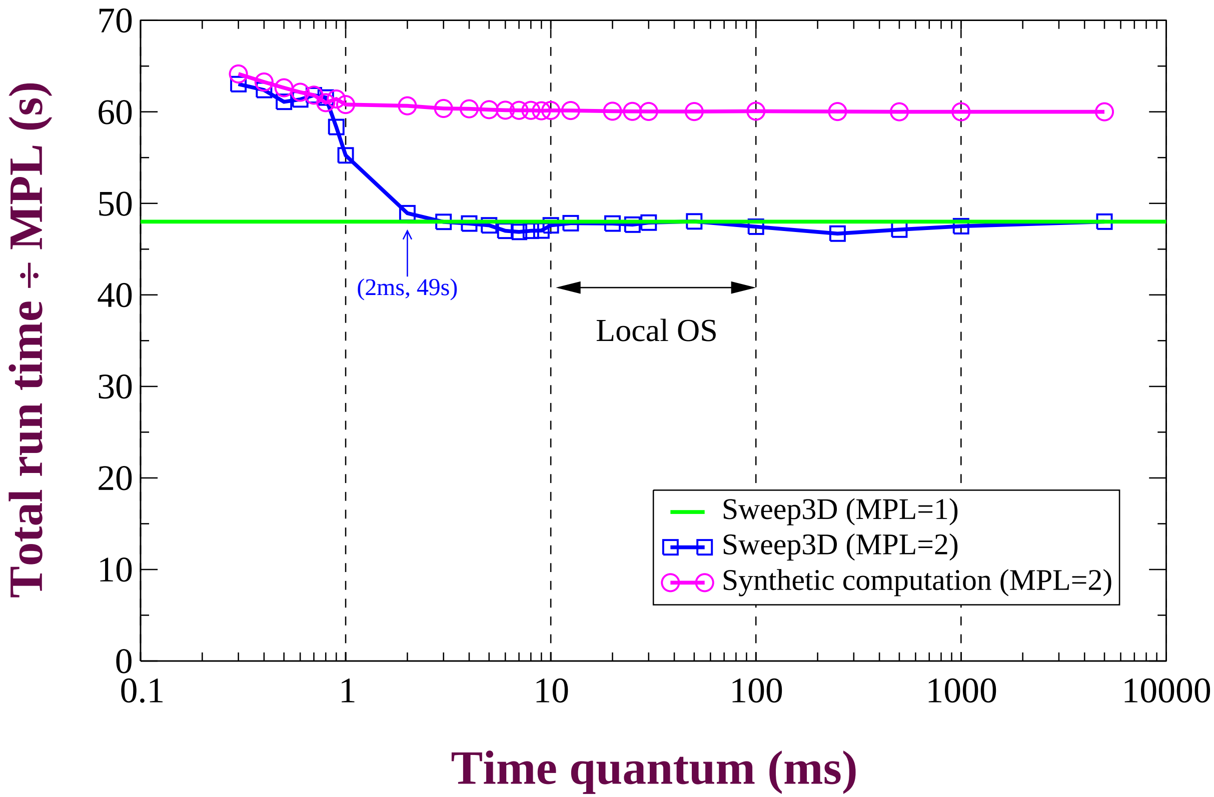
Los Alamos NATIONAL LABORATORY

Relative performance of Cplant, BProc, and STORM



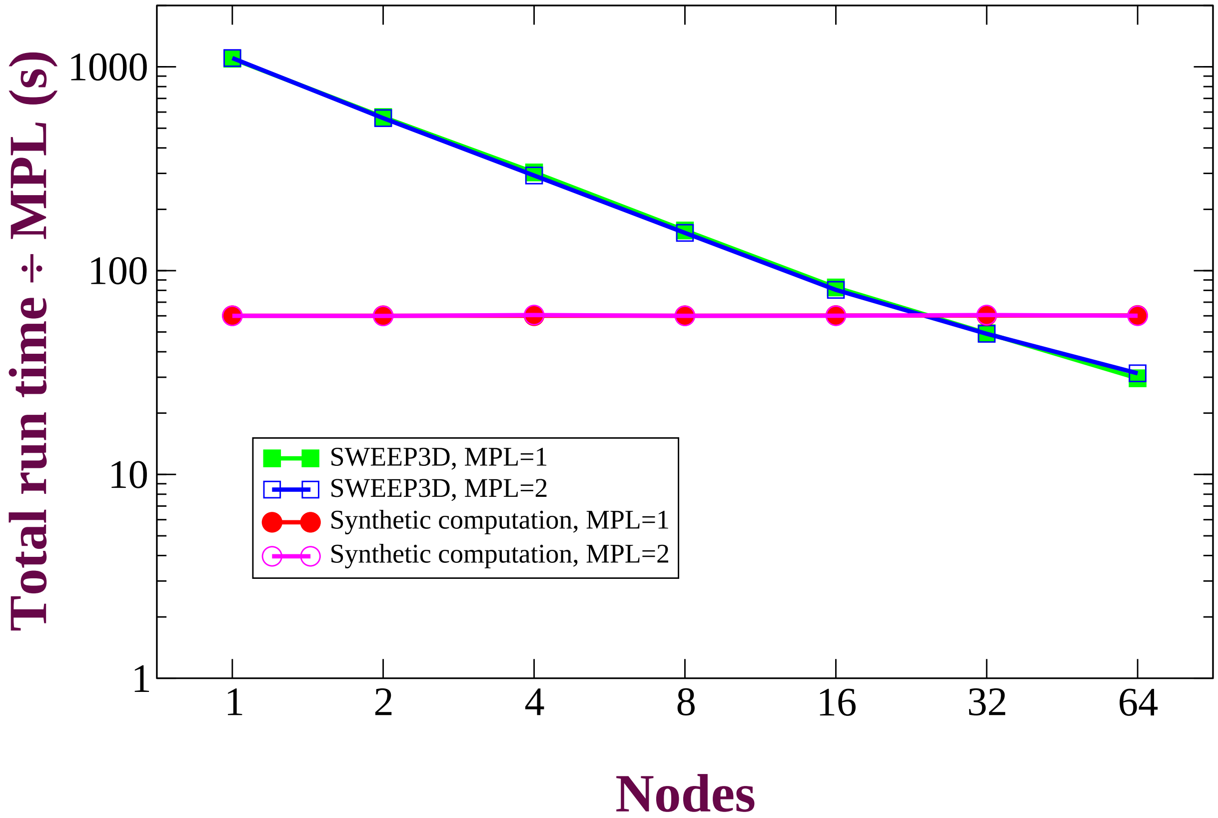
Los Alamos NATIONAL LABORATORY

Effect of time quantum with an MPL of 2



- Cluster-wide jobs can be scheduled as fast a local process on a desktop OS.

Effect of node scalability



- The scheduling algorithm is scalable with the number of nodes

A selection of scheduling quanta found in the literature

Resource Manager	Minimal feasible scheduling quantum	
RMS	$30,000$	milliseconds on 15 nodes (1.8% slowdown)
SCore-D	100	milliseconds on 64 nodes (2% slowdown)
STORM	2	milliseconds on 64 nodes (no observable slowdown)

Los Alamos
NATIONAL LABORATORY

Conclusions

- STORM uses an innovative design based on a small set of data-transfer and synchronization mechanisms:
  - XFER-AND-SIGNAL
  - TEST-EVENT
  - COMPARE-AND-WRITE
- STORM's design makes it orders of magnitude faster than the best reported results in the literature for both job launching and process scheduling.

Los Alamos
NATIONAL LABORATORY

Conclusions (continued)

- STORM is a lightweight, flexible and scalable environment for performing resource management in large-scale clusters
- It is indeed possible to scale up a cluster without sacrificing job-launching times, machine efficiency or interactive response time.
- HW support for collective communication can simplify system software and can help to achieve efficiency and scalability

Los Alamos
NATIONAL LABORATORY

Resources

More information can be found at the following URLs:

Los Alamos Performance and Architecture Laboratory

http://www.c3.lanl.gov/par_arch

Resource management

http://www.c3.lanl.gov/~fabrizio

Quadrics network

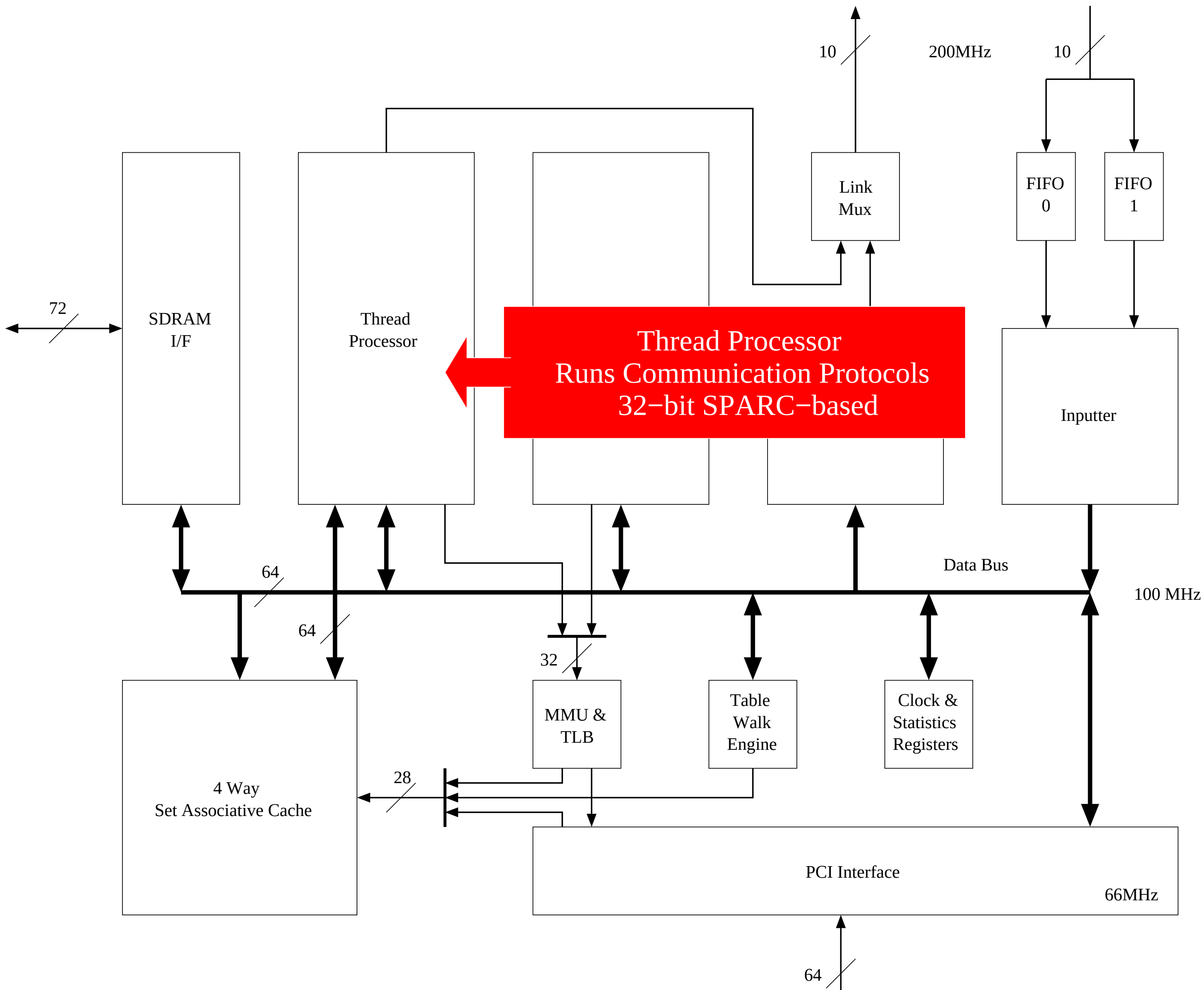
http://www.quadrics.com and

http://www.c3.lanl.gov/~fabrizio/quadrics.html

Los Alamos NATIONAL LABORATORY

DEMO in LANL booth (R3211)

Quadrics Network: Elan



Quadrics Network: Elan