STORM

Scalable TOol for Resource Management

Eitan Frachtenberg, Juan Fernandez, Fabrizio Petrini and Scott Pakin
{eitanf,juanf,fabrizio,pakin}@lanl.gov
Modeling, Algorithms and Informatics Group (CCS-3)
Los Alamos National Laboratory

Goals

Scalable, lightweight and fast resource-management:
- Resource Allocation
- Job-launching
- Cluster-wide synchronization and context-switching

Increase the usability of a cluster:
- Checkpointing and fault-tolerance
- Improved system utilization
- Improved system responsiveness

Testbed for current and new scheduling algorithms.

Orders of magnitude faster than existing production systems (ASCI Q at LANL, C-Plant at Sandia) and the best published results.

For more information:
"STORM:
Lightning-Fast Resource Management"
In Proceedings of the IEEE/ACM SC2002
http://www.c3.lanl.gov/~fabrizio

Los Alamos NATIONAL LABORATORY

Job Launching

STORM employs several innovative mechanisms to enable extremely fast job launching, even on thousands of nodes:
- use of hardware collective communications to multicast binary and data files.
- I/O-bypass mechanism to transmit the files directly by the NICs, without CPU intervention.

Figure 1 shows measured results for job launching on a *256-processor AlphaServer ES40 cluster* at LANL (134th in the top500 list). Results are shown for three binary image sizes and are split into the time to send the binary file from the file server to all the compute nodes, and the time to actually execute it.

Figure1

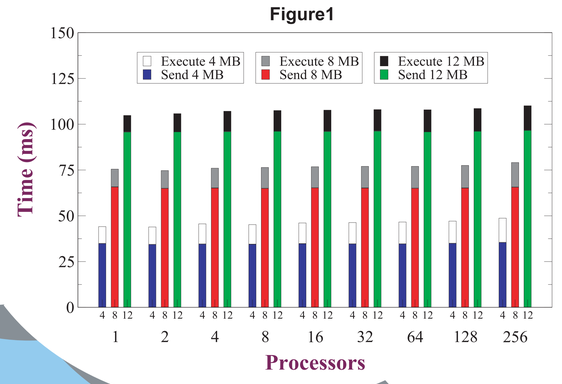
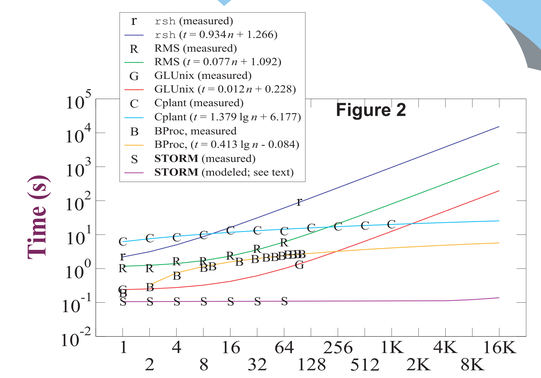


Figure 2

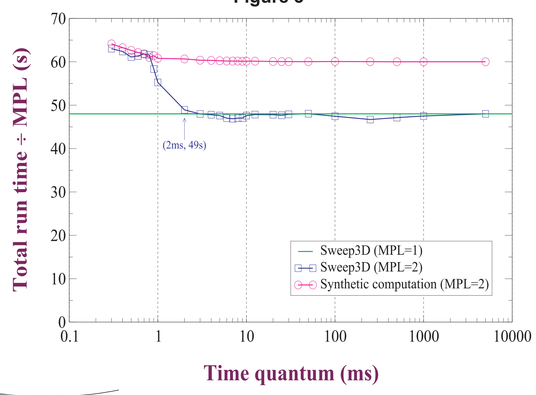


STORM's launch times are orders of magnitude better than other production and research systems.

To compare STORM with other systems, we gathered results from the literature, and projected how well each system would scale with the number of nodes.
Figure 2 shows the measured and predicted launch times of a 12MB executable for up to 16,384 nodes.
We used a very detailed model to predict STORM's launch time, which remains well under a second even for 16K nodes.

Resource Management

Figure 3



STORM implements several job scheduling algorithms, including batch scheduling with back-filling and gang scheduling. *Gang scheduling* enables the system to switch a program in favor of a higher priority program, or just run several parallel jobs concurrently for improved responsiveness. However, gang-scheduling is not widely used, partly because the overhead of context-switching an entire parallel job can be prohibitive. We implement a global context-switch in STORM that uses efficient communication primitives to make the entire operation extremely lightweight.

Table 1

Resource Manager	Minimal feasible quantum
RMS	30,000 ms on 15 nodes
SCore-D	100 ms on 64 nodes
STORM	2 ms on 64 nodes

How well this compare to other systems?
Table 1 shows the minimal value of usable context-switch quanta of STORM, RMS and Score-D.

Figure 3 shows the effect of using different time-quanta values when running two copies of a job concurrently.
We consider a synthetic compute-bound job and SWEEP3D, a real LANL kernel.
The overhead of context-switching only appears when using time quanta of less than 2ms.
In fact, Figure 3 shows that STORM can perform a global context-switch with similar frequency and efficiency of that of a local-node, enabling interactive responsiveness even in a large cluster.