Gang Scheduling with Lightweight User-Level Communication

Eitan Frachtenberg,[⋆] Fabrizio Petrini,[⋆] Salvador Coll,[⋆] and Wu-chun Feng[†]

[⋆] CCS-3 Modeling, Algorithms, and Informatics Group

[†] CCS-1 Advanced Computing

Computer and Computational Sciences (CCS) Division

Los Alamos National Laboratory

Motivation

Buffered Coscheduling - a new approach to resource management:
Bufferered Coscheduling addresses the following issues:

- Improved utilization of system resources
- Improved responsiveness
- Transparent fault-tolerance (self-healing)

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

Motivation (cont.)

- First implementation of BCS will use the Quadrics hardware to exploit the advantages of the hardware and software.
- We present a preliminary study of the advantages to scheduling by analysing Quadrics' scheduler, RMS:
  - How does the software and hardware of the Quadrics interconnect affect scheduling?
  - How does the Quadrics scheduler RMS perform?

Outline

- Background: The Quadrics HW and SW
- Experimental goals and methodology
- Experimental results
- Conclusions

Background

The Quadrics Hardware

- Processes can map portions of their address space into the Elan and read/write to other processes address space through the network.
- The Elan network interface card (NIC) has a dedicated processor and 64 MB of SDRAM.
- The NIC has its own TLBs.
- A context switch does not require buffer flushing, only TLB changes in the NIC.
- Capable of delivering more than 300 MB/s of data.

Background (cont.)

Gang Scheduling

- Schedule and deschedule all processes of a job together using global context switch.
- Jobs “believe” they have a dedicated machine.
- More responsive than batch systems.
- Better utilization of resources under varying workloads.
- Can incur overheads: TLBs, communication buffers, swapping.

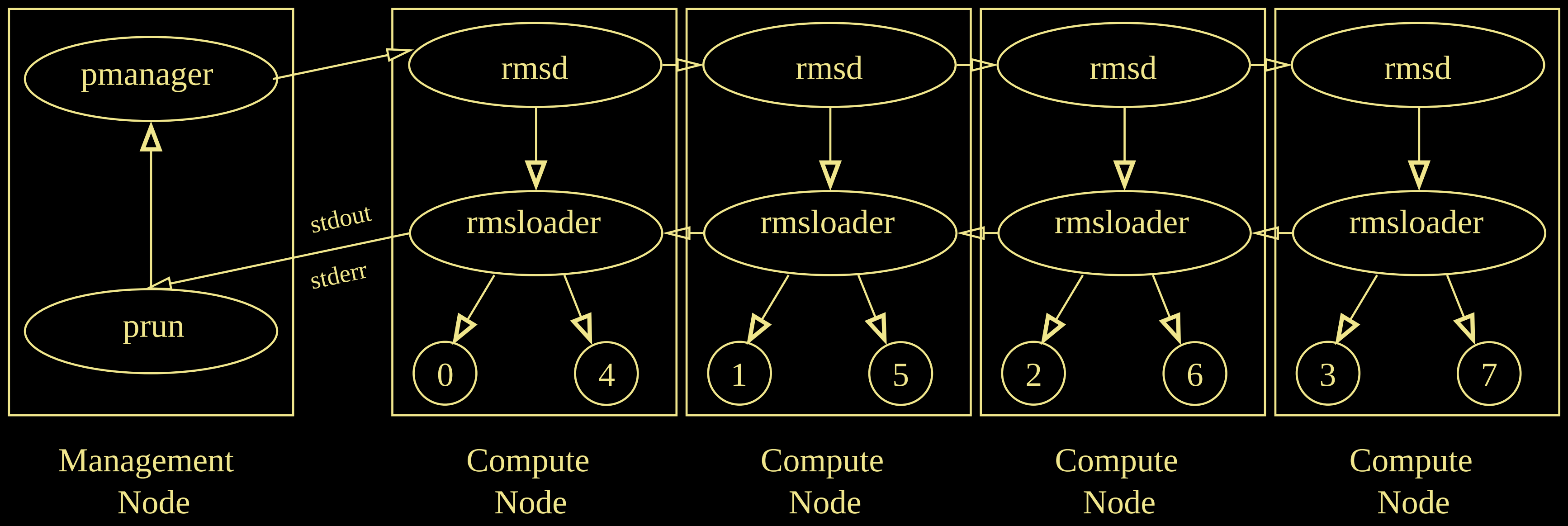
Background (cont.)

The Quadrics gang scheduler (RMS)

- connects a cluster of computers with a management and Quadrics network.
- Manages cluster resources including PEs and user-level communication.
- Composed of a set of programs, daemons and an SQL database.

Background (cont.)

Example: running a program in RMS



Goals

1. Measure overhead of gang scheduler under varying conditions:
   (a) Memory requirements.
   (b) Timeslice values.
   (c) Latency-bound communication.
   (d) Bandwidth-bound communication.
2. Scalability issues:
   (a) Number of nodes.
   (b) multiprogramming level.

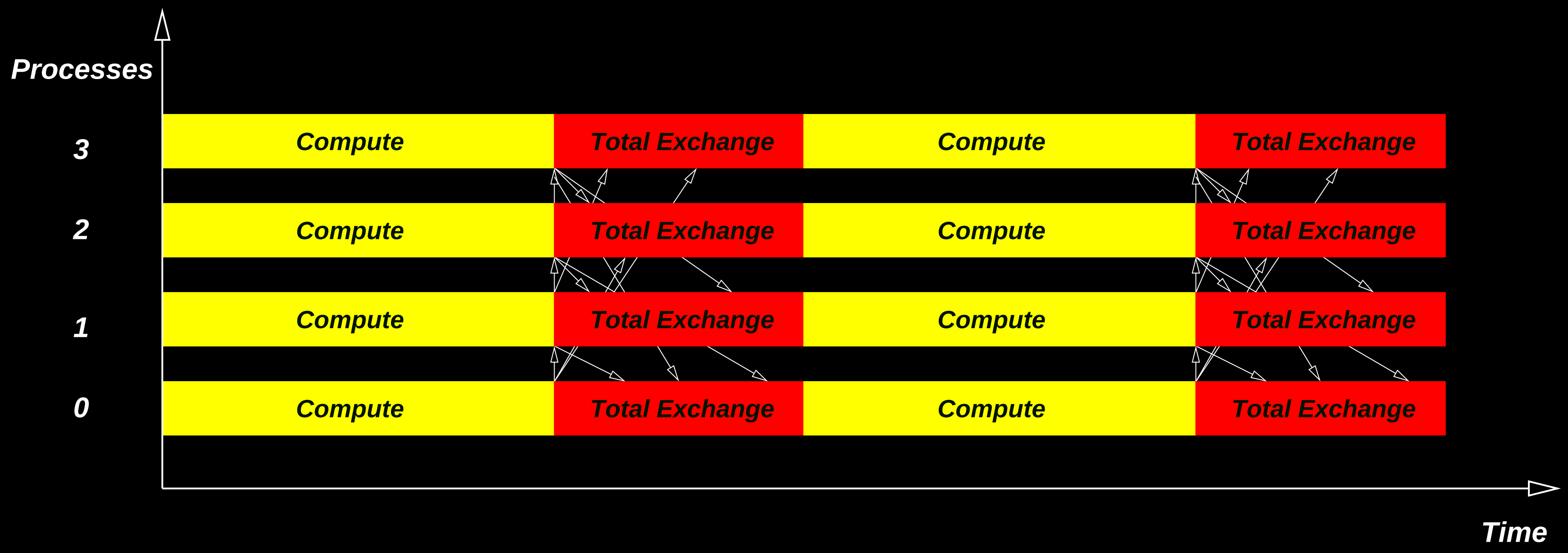
Experimental Methodology

- We developed a micro-benchmark that performs computation and communication.
- The following parameters are adjustable:
  - Number of computation cycles.
  - Amount of memory used. Large stride is used to avoid cache benefits.
  - Number of total exchanges (TEs) and TE buffer size.
- An external Perl script is used to run predefined sets of experiments.

Experimental platform

- 1-16 dual Pentium-III 733 Mhz nodes.
- 64 MB/s, 66 MHz PCI bus.
- 1 GB ECC SDRAM per node.
- First node serves as management node for pmanager.

Workload

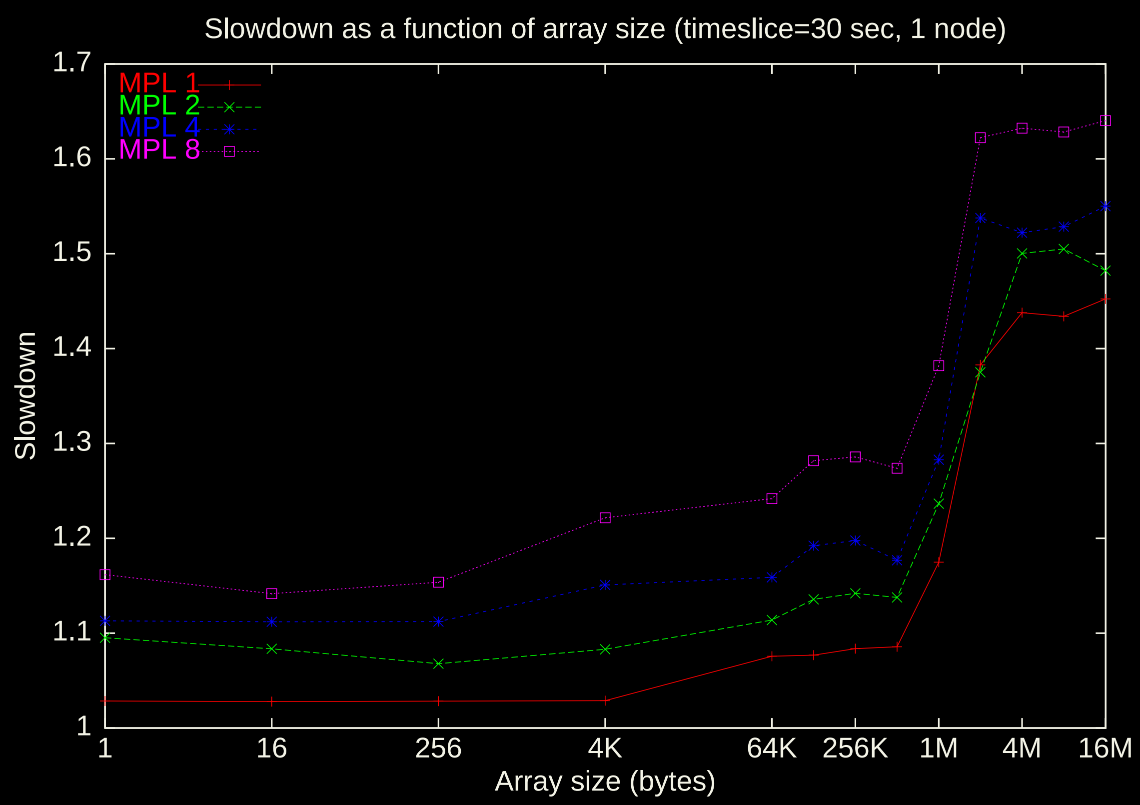

Processes
3
2
1
0
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Compute
Total Exchange
Time

Workload (cont.)

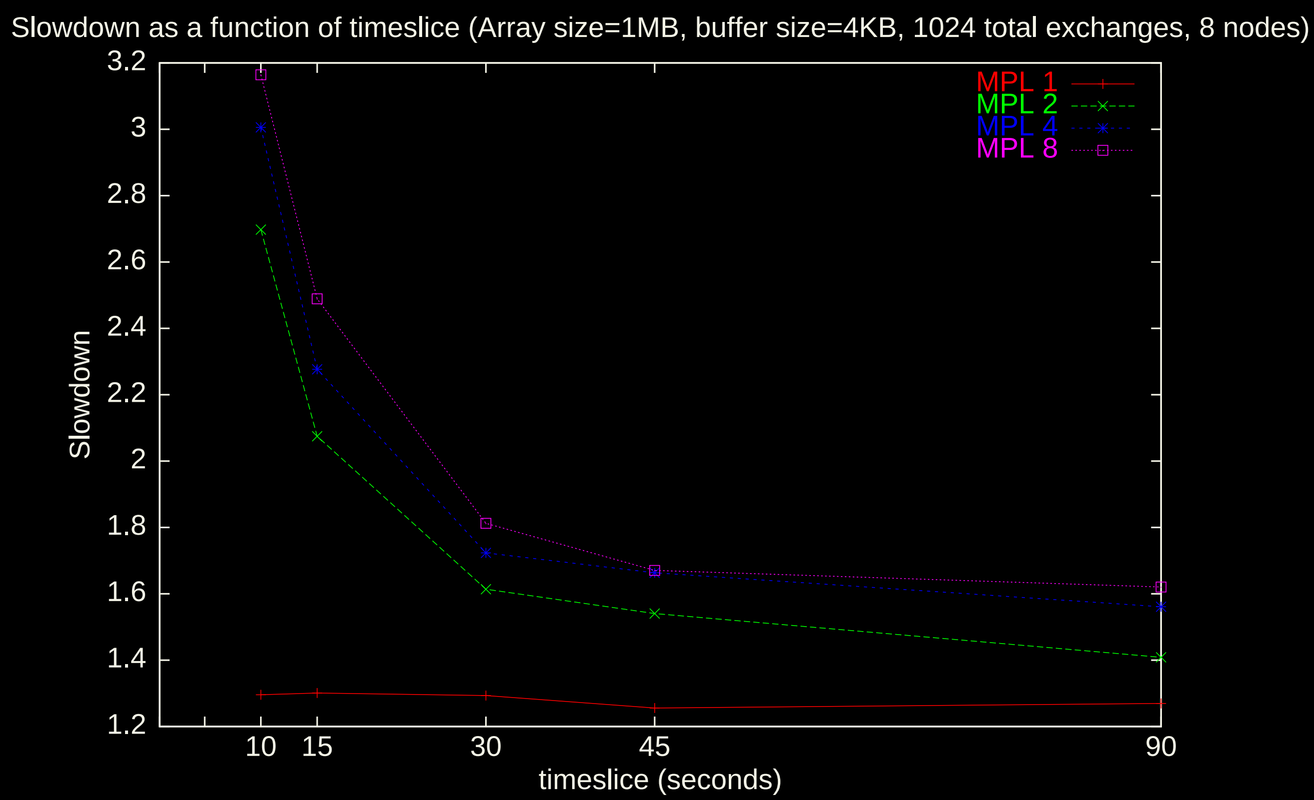
Default parameters for experiments

- Computation amount: 10^8 cycles (equivalent to ≈ 50 CPU seconds).
- Array size of 1 MB.
- Timeslice quantum: 30 sec.
- Number of nodes: 8 (16 PEs).
- 1,024 total exchanges with a buffer size of 4KB (≈ 1 total exchange per 50 ms of computation).

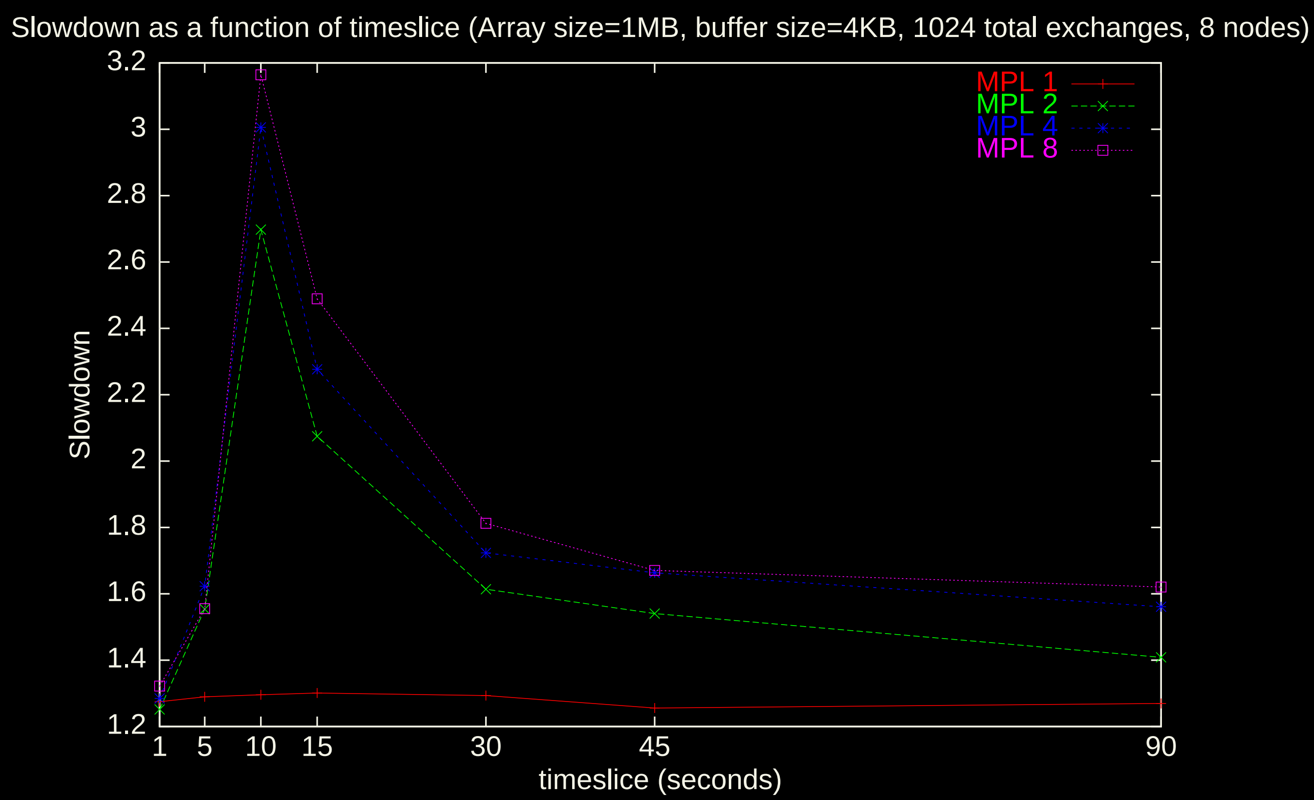
Results - Memory Requirements



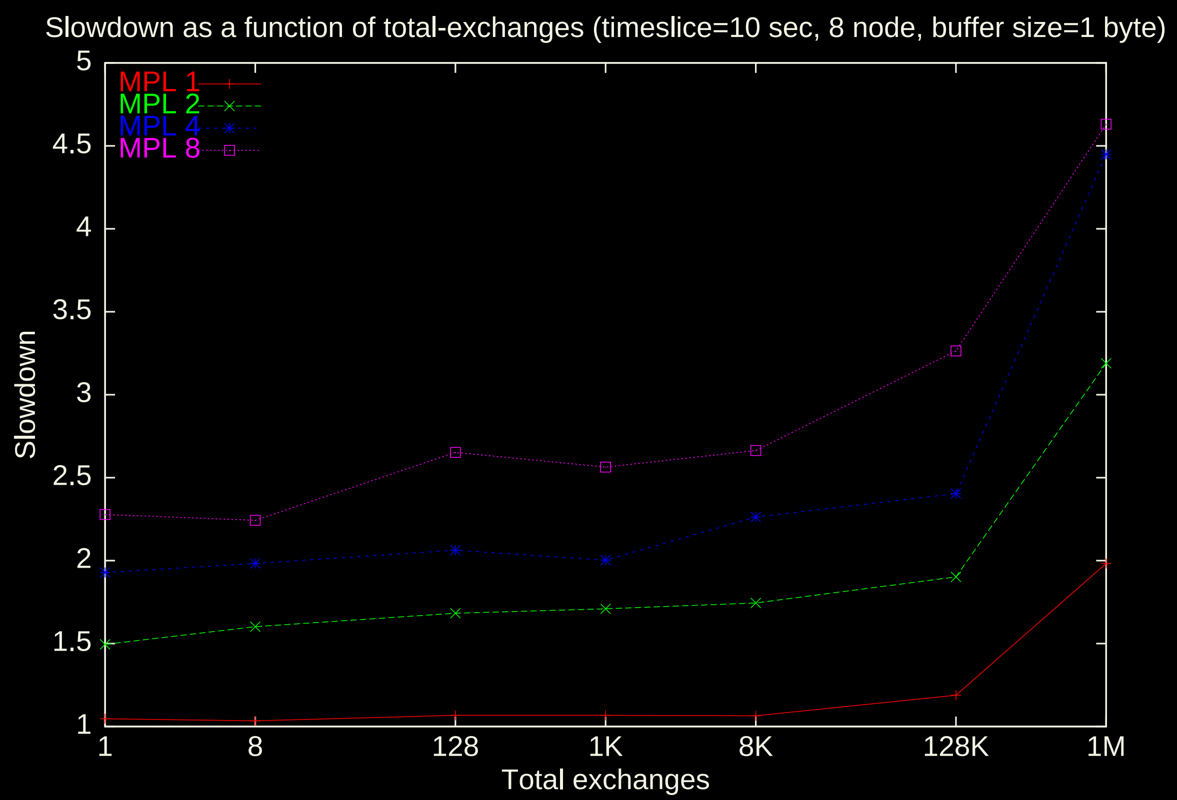
Results - Timeslice Quantum



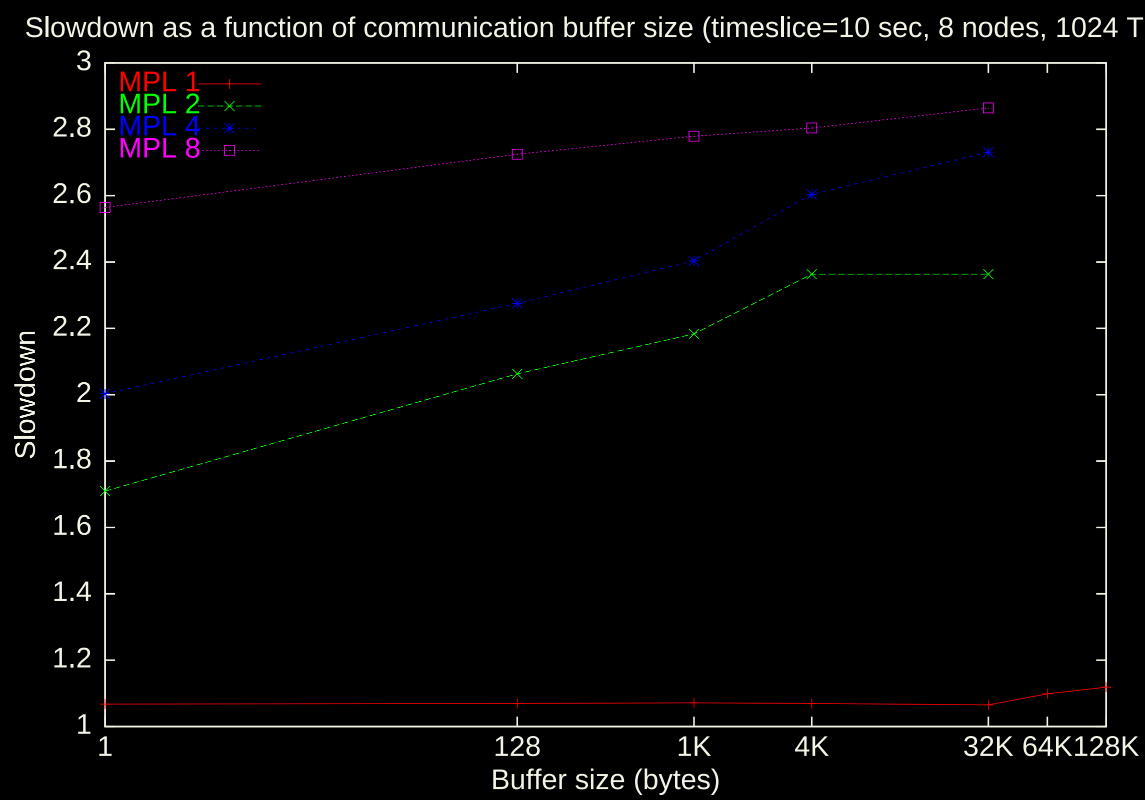
Results - Timeslice Quantum



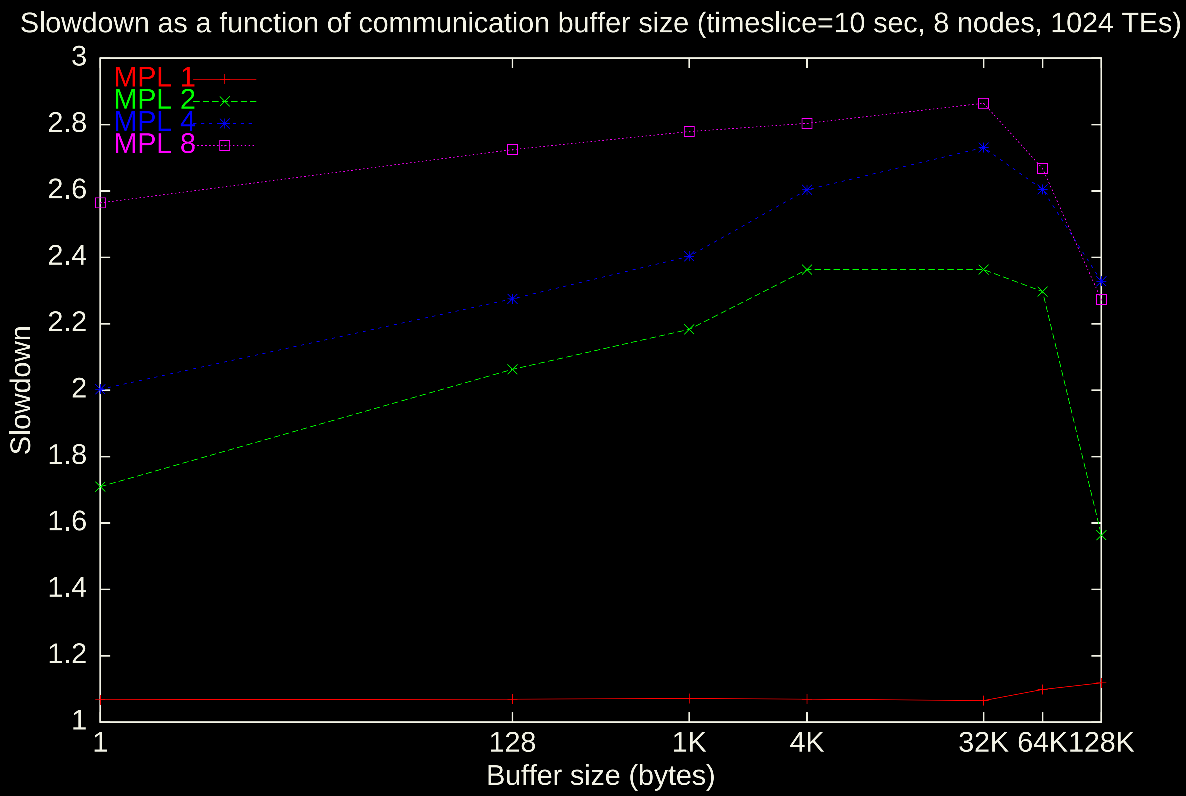
Results - Latency-Bound Communication



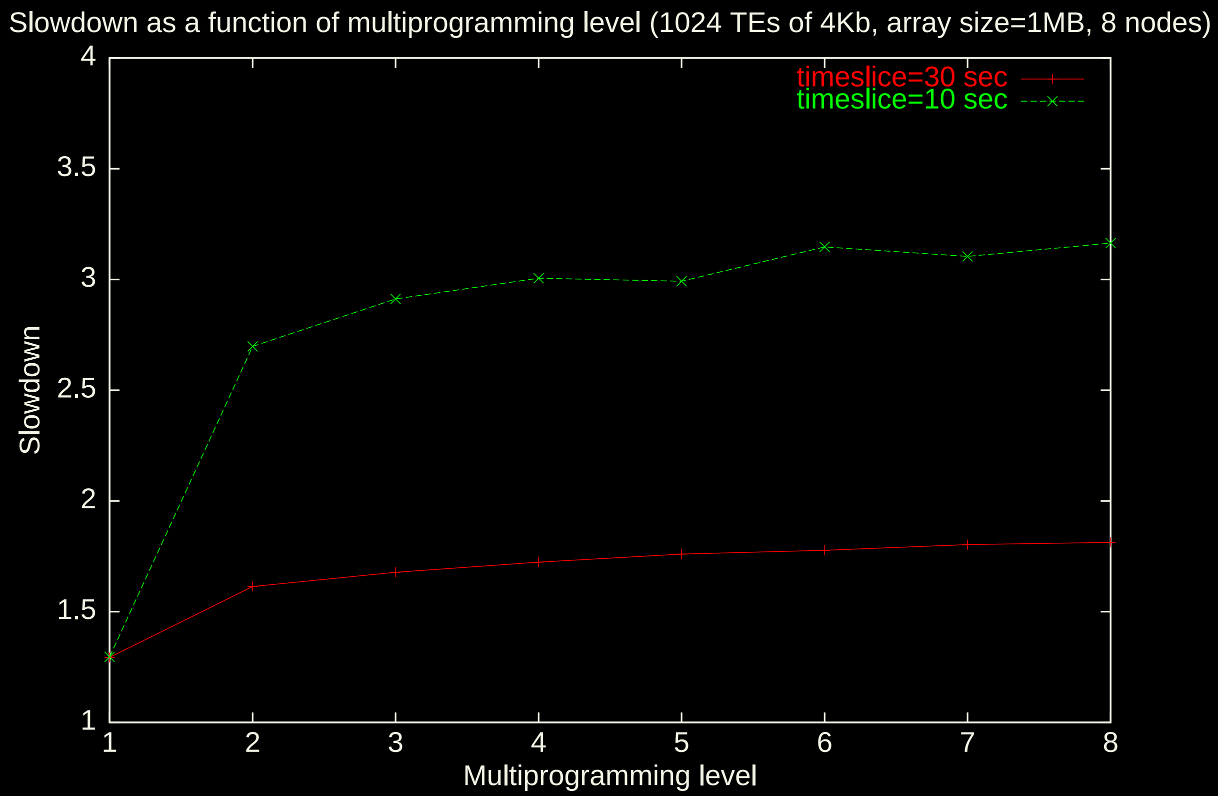
Results - Bandwidth-Bound communication



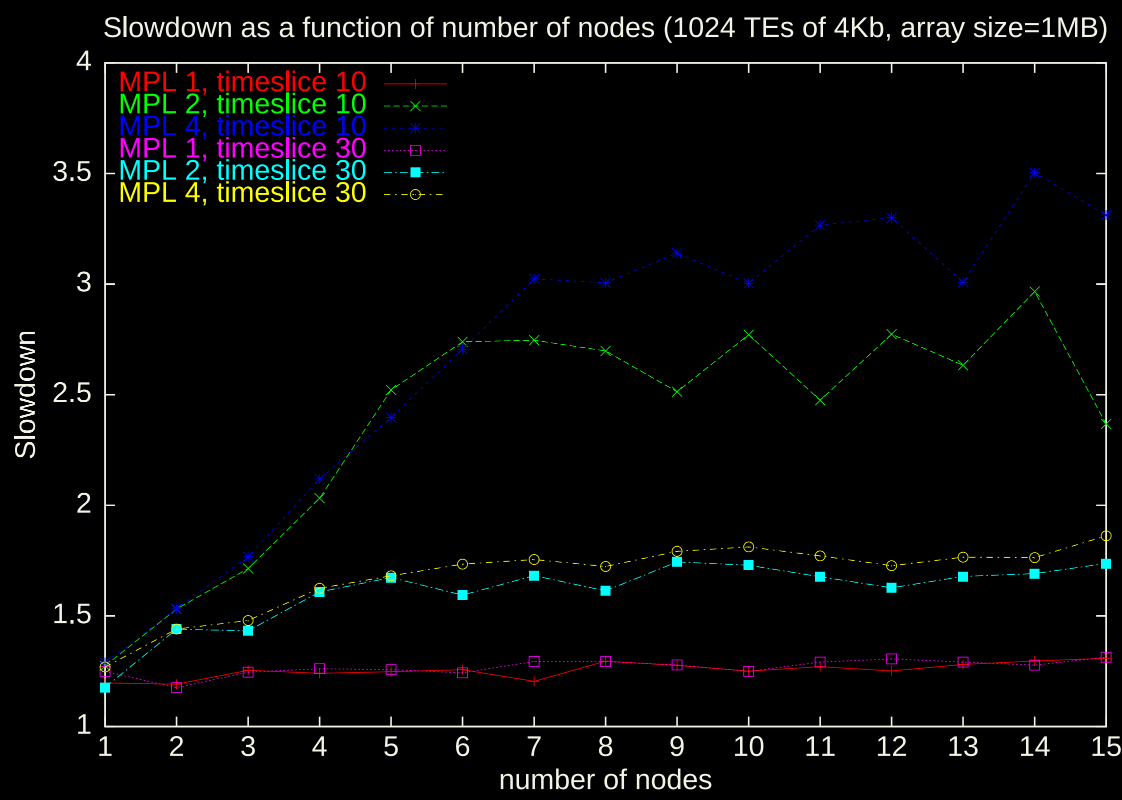
Results - Bandwidth-Bound communication



Results - Multiprogramming level



Results - Node scalability



Conclusions

✔ Scheduler is relatively insensitive to communication granularity.

Conclusions

- ✔ Scheduler is relatively insensitive to communication granularity.
- ✔ Can improve performance of coscheduled bandwidth-hungry programs.

Conclusions

- ✔ Scheduler is relatively insensitive to communication granularity.
- ✔ Can improve performance of coscheduled bandwidth-hungry programs.
- ✔ Scheduler is relatively insensitive to memory requirements (when not swapping).

Conclusions

- ✔ Scheduler is relatively insensitive to communication granularity.
- ✔ Can improve performance of coscheduled bandwidth-hungry programs.
- ✔ Scheduler is relatively insensitive to memory requirements (when not swapping).
- ✔ Scalable both in terms of PEs and multiprogramming level.

Conclusions

- ✔ Scheduler is relatively insensitive to communication granularity.
- ✔ Can improve performance of coscheduled bandwidth-hungry programs.
- ✔ Scheduler is relatively insensitive to memory requirements (when not swapping).
- ✔ Scalable both in terms of PEs and multiprogramming level.
- ✘ Very sensitive to timeslice quantum.

Resources

- More information can be found at

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

- Quadrics web site

http://www.quadrics.com

- Or by sending an email to

eitanf@lanl.gov