Buffered Co-scheduling: A New Methodology for Multitasking Parallel Jobs on Distributed Systems

Fabrizio Petrini and **Wu-chun Feng**
{fabrizio,feng}@lanl.gov

Los Alamos National Laboratory

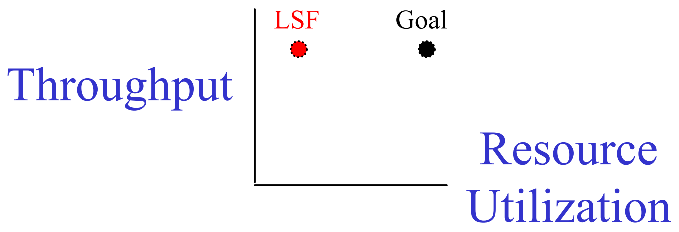
LA-UR-00-0894

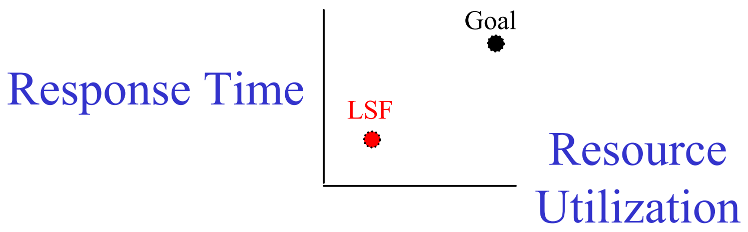
Outline

- Motivation & Background
- Related Work
- Buffered Coscheduling
  - Communication Buffering
  - Strobing
  - Non-Blocking Communication
- Workload and Simulation Model
- Experimental Results
- Conclusion

Motivation

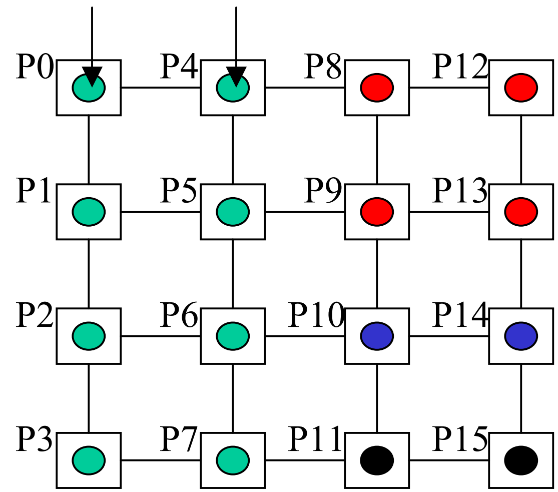
- Goals
  - High system throughput.
  - Fast response time for interactive & high-priority jobs.
  - Efficient resource utilization.
- Current Commercial Solution
  - Third-party software: LSF.
  - Problem: Efficient space sharing (> 90%) but no time sharing.
    - CPU utilization ≈ 55% (typical)
    - Network utilization ≈ 5 % (typical)
    - *Result: High system throughput, poor response time, and inefficient resource utilization.*



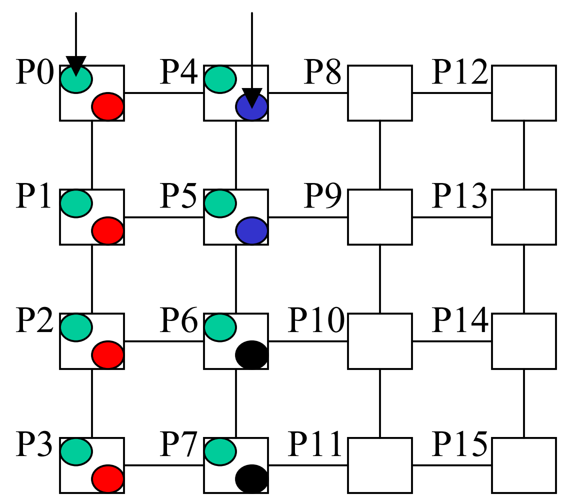


Space Sharing vs. Time Sharing

- Space Sharing (LSF)
  - Parallel Program 1 (8 parallel jobs)
  - Parallel Program 2 (4 parallel jobs)
  - Parallel Program 3 (2 parallel jobs)
  - Parallel Program 4 (2 parallel jobs)

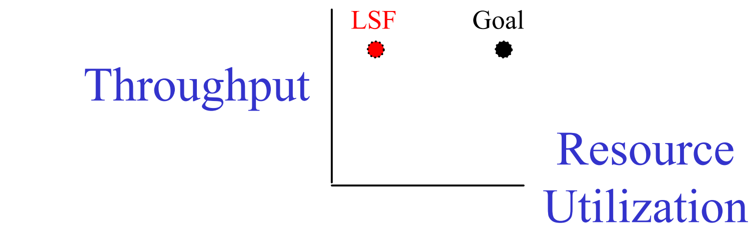


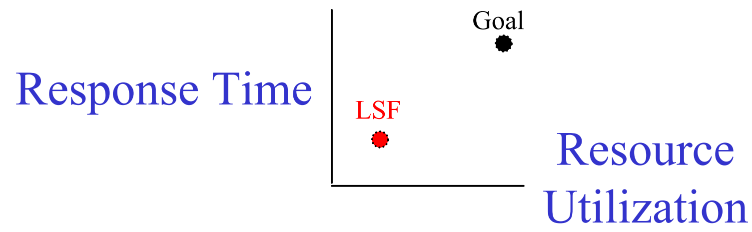
- Time Sharing
  - Parallel Program 1 (8 parallel jobs)
  - Parallel Program 2 (4 parallel jobs)
  - Parallel Program 3 (2 parallel jobs)
  - Parallel Program 4 (2 parallel jobs)

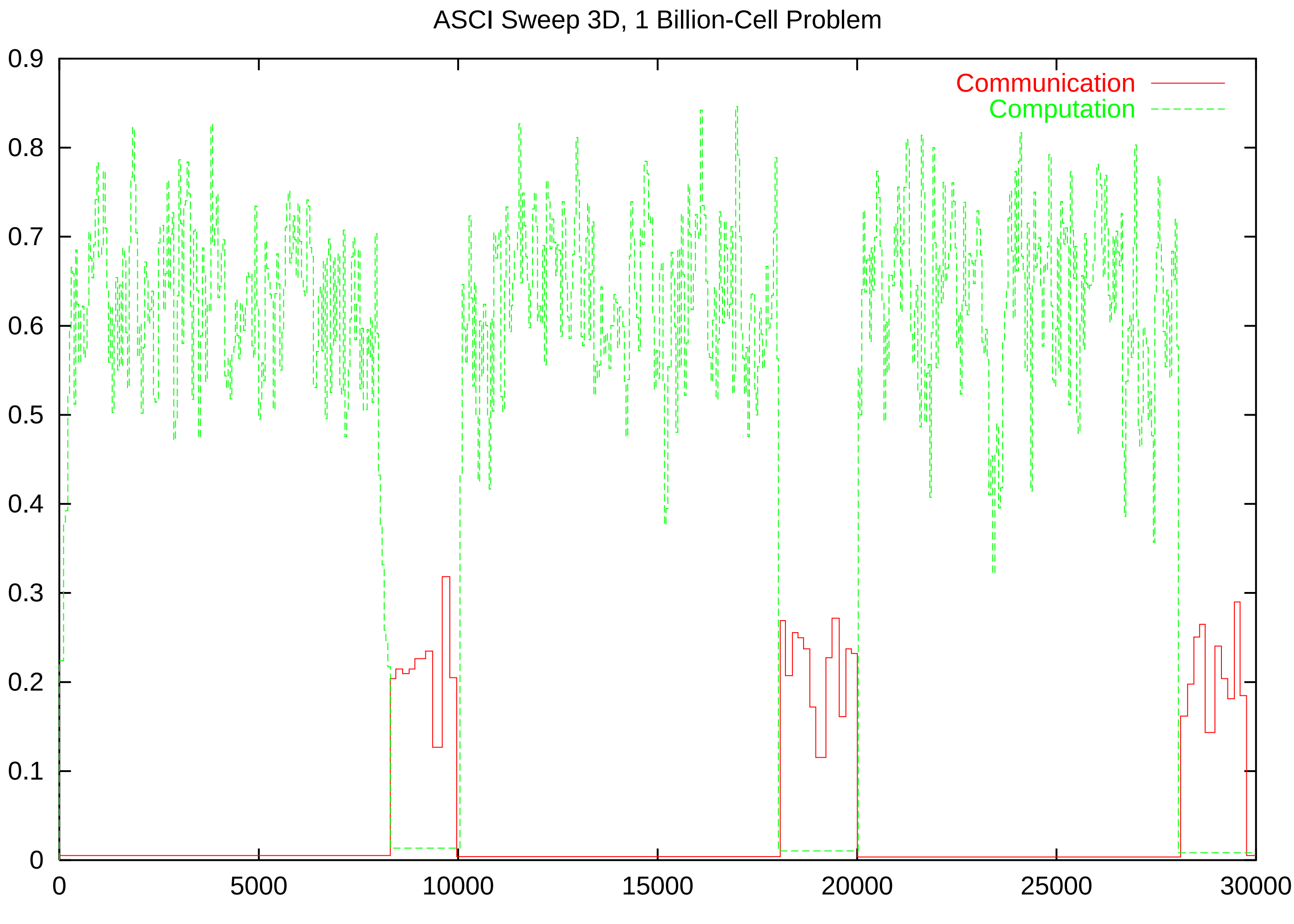


Motivation

- Goals
  - High system throughput.
  - Fast response time for interactive & high-priority jobs.
  - Efficient resource utilization.
- Current Commercial Solution
  - Third-party software: LSF.
  - Problem: Efficient space sharing (> 90%) but no time sharing.
    - CPU utilization ≈ 55% (typical)
    - Network utilization ≈ 5 % (typical)
    - *Result: High system throughput, poor response time, and inefficient resource utilization.*






ASCI Sweep 3D, 1 Billion-Cell Problem
Communication
Computation
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5000
10000
15000
20000
25000
30000

Related Work

- Time Sharing (a.k.a. Coscheduling or Gang Scheduling)
  - *Explicit Coscheduling*
    - Precomputed job schedule that requires simultaneous context-switching across all processors.
    - System throughput (+), response time (---), resource utilization (+)
  - *Local Scheduling*
    - Each processor independently schedules its processes.
    - System throughput (---), response time (-), resource utilization (---)
  - *Implicit or Dynamic Coscheduling* (UC-Berkeley & MIT)
    - Each processor makes decisions that dynamically coordinate scheduling actions of cooperating processes across processors.
    - System throughput (-), response time (+), resource utilization (+)

Buffered Coscheduling

A New Methodology to Multitask Parallel Programs

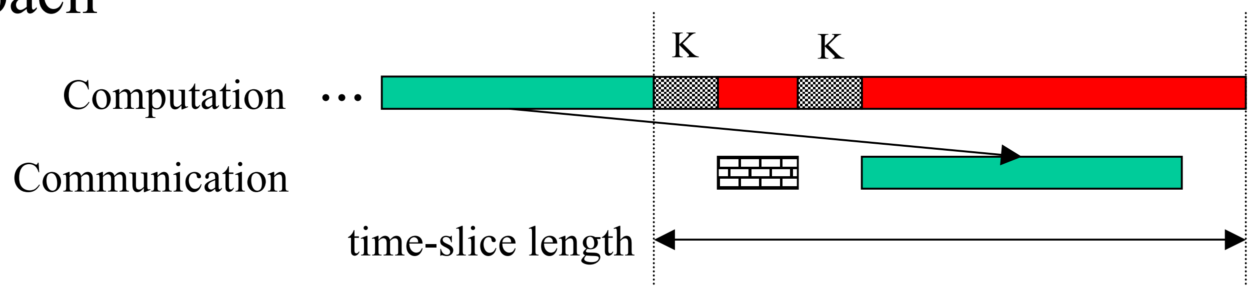
- Components
  - Communication Buffering
  - Strobing for Information Exchange & Synchronization
  - Non-Blocking Communication
- Features
  - Push resource management from MPI down into the operating system.
    - Allows computation, communication, and I/O which arise from a *set* of parallel programs to be overlapped with the computations in those programs.
  - Amortize communication overhead.
  - Provide a framework for fault tolerance.
  - Decrease software development time for parallel programs.
  - Enable accurate performance analysis.

Communication Buffering

- Goal
  - Amortize communication overhead over a *set* of messages.
- Approach
  - Buffer communication over a given time slice.
  - Perform communication at the end of a time slice.
  - (Future Note: Jobs can be multitasked at any time.)
- Result
  - Up to 87% decrease in execution time for parallel programs.
  - Speedup of up to 7.5 times.

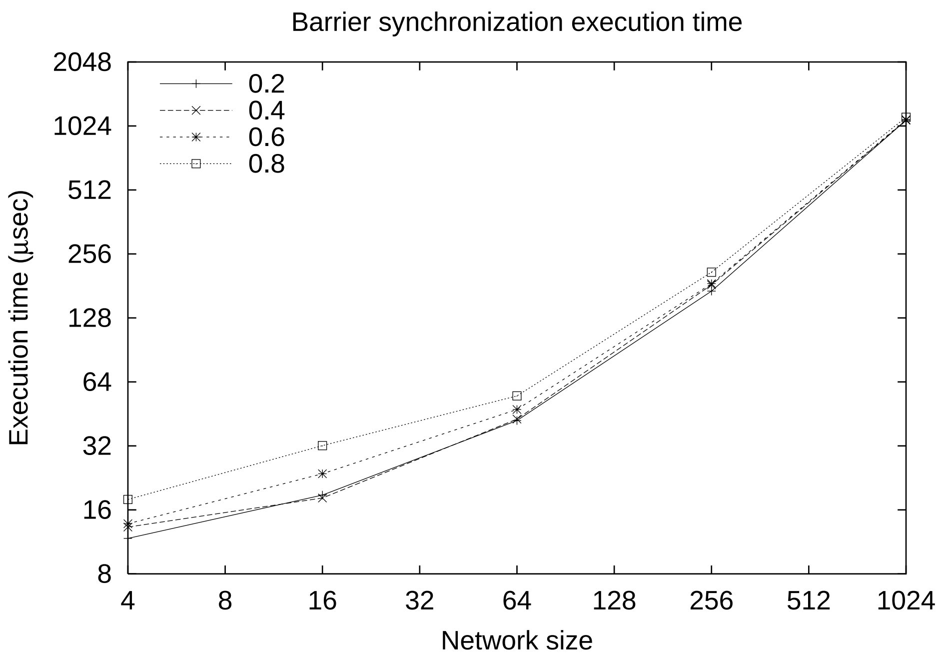
Strobing

- Goal
  - Provide a mechanism for a total exchange of control information at the end of each time slice (to fill in communication "holes").
- Approach

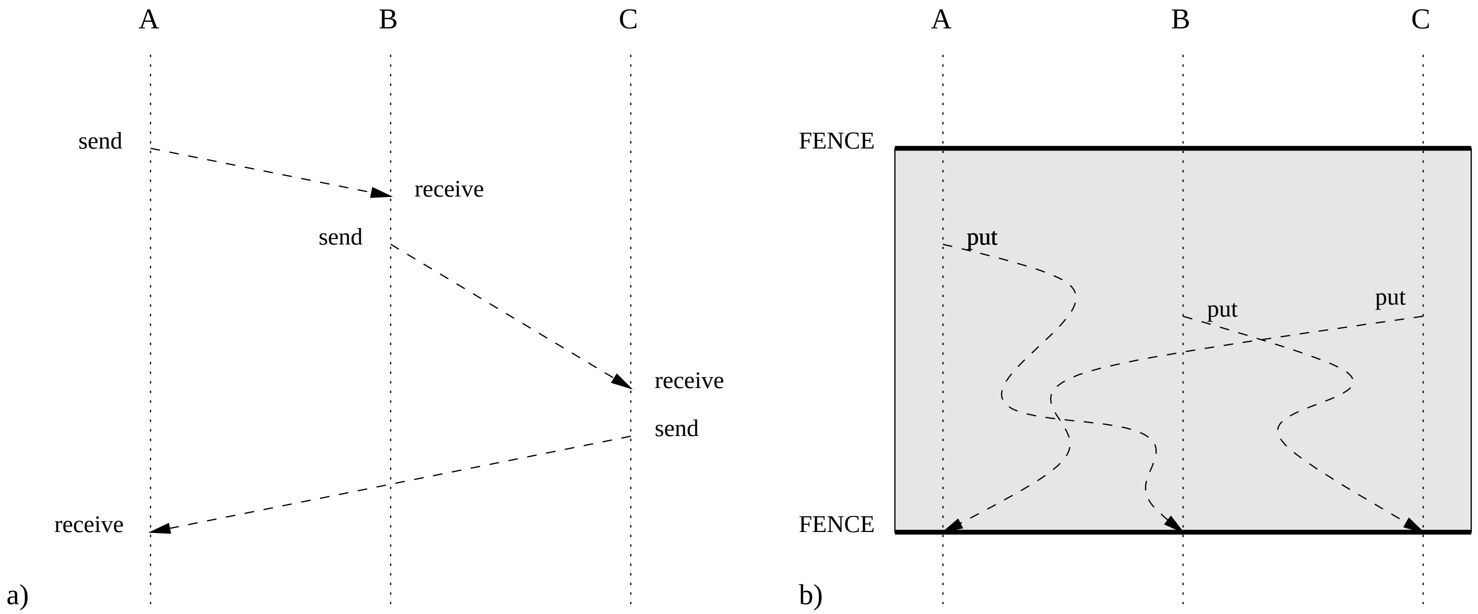


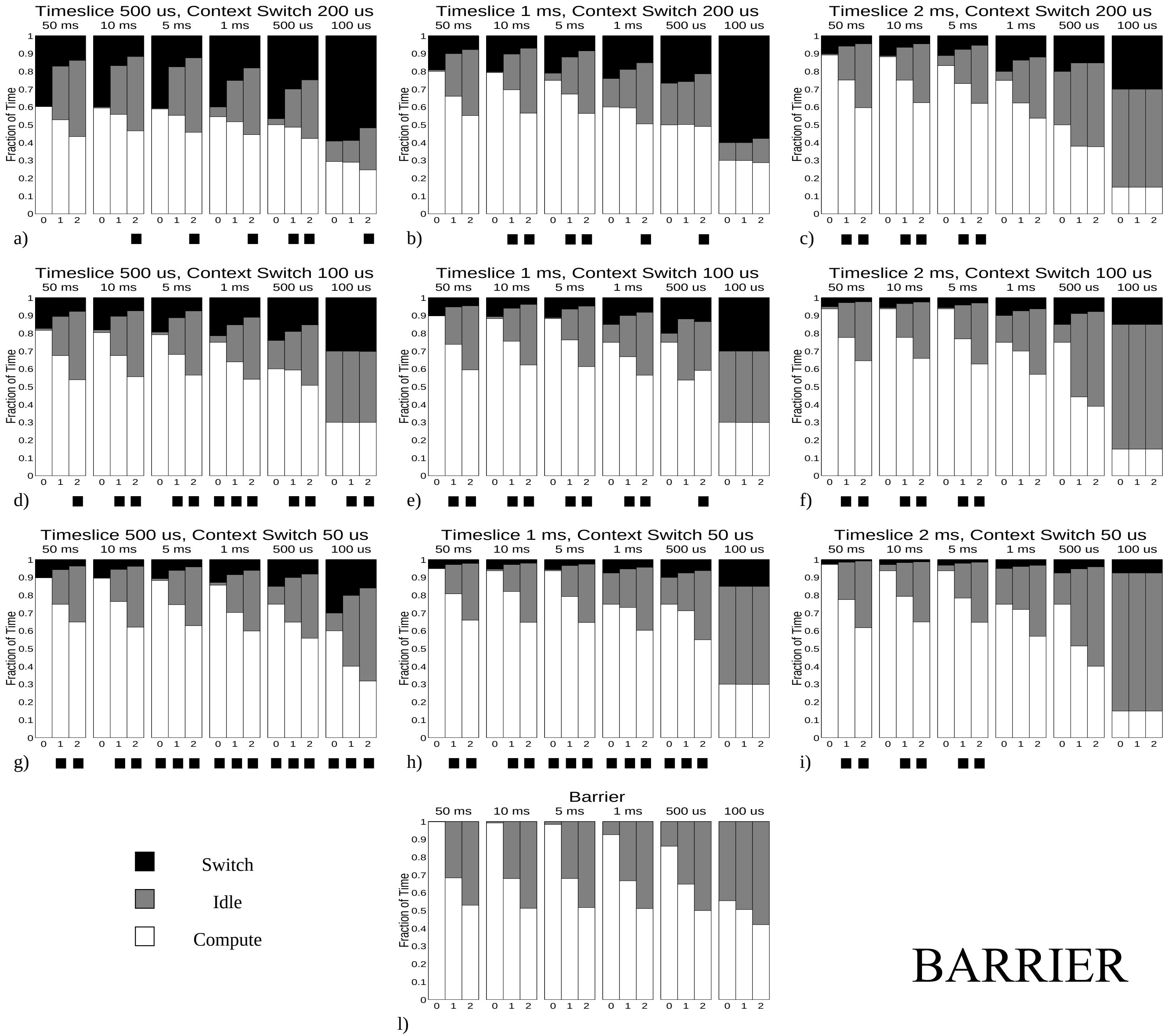
- Result: Predictable ("bounded") barrier synchronization.
- Implication
  - Enables global flow-control strategy as every processor knows how much information to expect & who to expect it from.

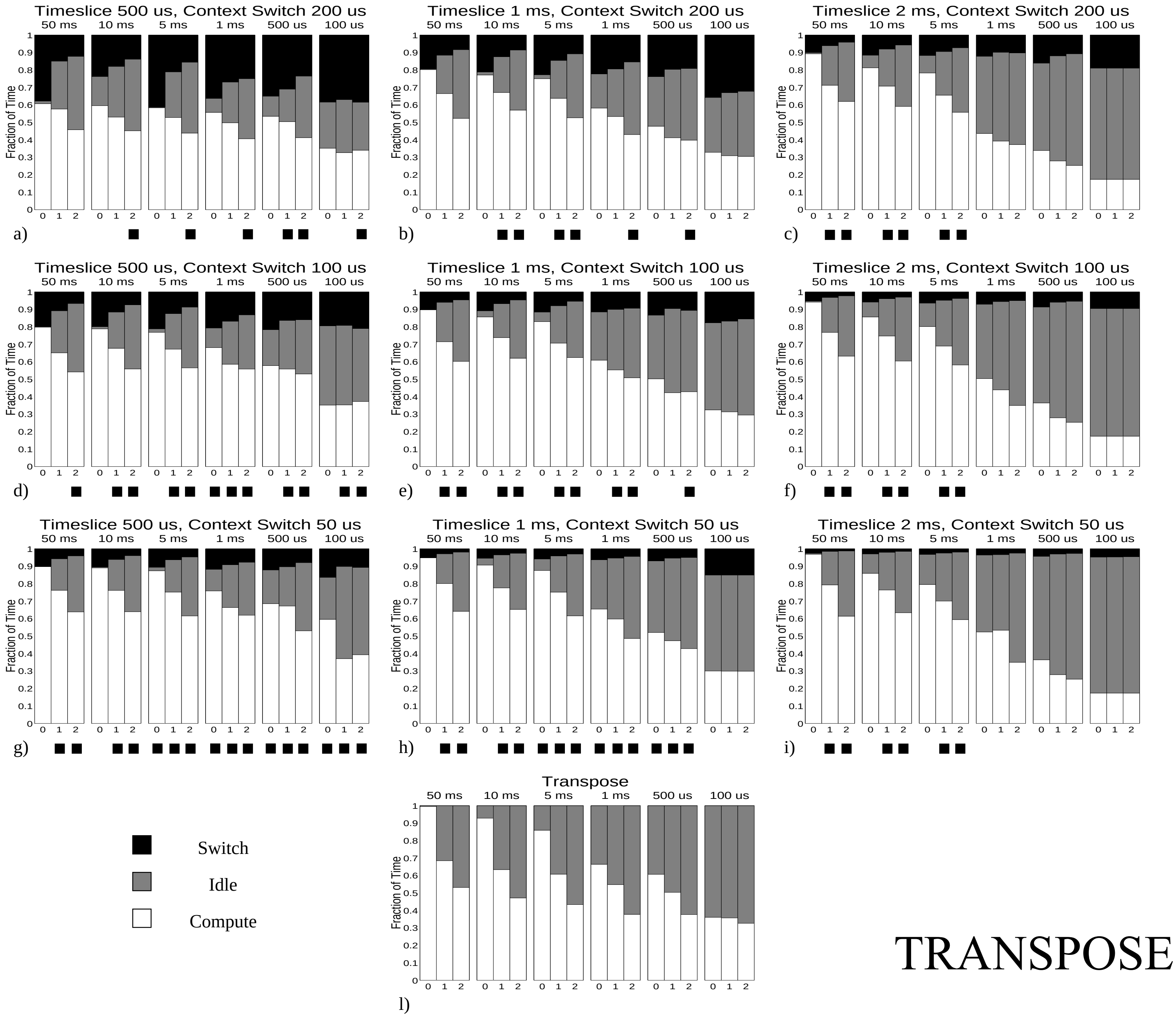
Barrier Synchronization Times



Non-Blocking Communication




Timeslice 500 us, Context Switch 200 us
Timeslice 1 ms, Context Switch 200 us
Timeslice 2 ms, Context Switch 200 us
Timeslice 500 us, Context Switch 100 us
Timeslice 1 ms, Context Switch 100 us
Timeslice 2 ms, Context Switch 100 us
Timeslice 500 us, Context Switch 50 us
Timeslice 1 ms, Context Switch 50 us
Timeslice 2 ms, Context Switch 50 us
Barrier
50 ms
10 ms
5 ms
1 ms
500 us
100 us
Fraction of Time
a)
b)
c)
d)
e)
f)
g)
h)
i)
l)
Switch
Idle
Compute
BARRIER


Timeslice 500 us, Context Switch 200 us
Timeslice 1 ms, Context Switch 200 us
Timeslice 2 ms, Context Switch 200 us
Timeslice 500 us, Context Switch 100 us
Timeslice 1 ms, Context Switch 100 us
Timeslice 2 ms, Context Switch 100 us
Timeslice 500 us, Context Switch 50 us
Timeslice 1 ms, Context Switch 50 us
Timeslice 2 ms, Context Switch 50 us
Transpose
50 ms
10 ms
5 ms
1 ms
500 us
100 us
Fraction of Time
a)
b)
c)
d)
e)
f)
g)
h)
i)
l)
Switch
Idle
Compute
TRANSPOSE

Conclusion

- Future Work
  - Benchmark and compare with commercial solutions (i.e., LSF & LoadLeveler) as well as implicit/dynamic coscheduling.
  - Leverage lessons learned to implement a distributed OS to support the buffered coscheduling methodology.
- Publications
  - Efficient Scheduling of Parallel Jobs on Massively Parallel Systems. In *7th IEEE Int'l Conference on Advanced Communications*. 12/99.
  - Scheduling with Global Information in Distributed Systems. To appear in *20th IEEE Int'l Conference on Distributed Computing Systems*. 4/00.
  - Buffered Coscheduling: A New Methodology for Multitasking Parallel Jobs on Distributed Systems. To appear in *2000 IEEE Int'l Parallel & Distributed Processing Symposium*. 5/00.