Predicting the Impact of Configuration Changes

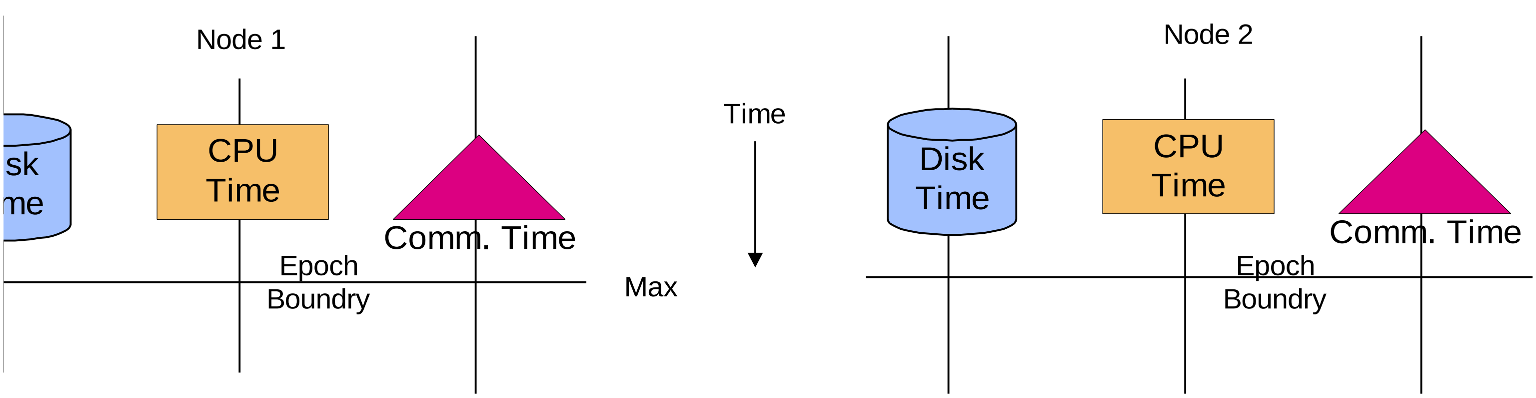
Jeff Hollingsworth
Hyeonsang Eom

A Family of Simulators

- Explore accuracy vs. time trade-off
  - Use simple static estimation of I/O and communicatio
  - Exploring adding stochastic variation
- Simplifying assumptions
  - no network link contention
  - predictable computation/communication interference
  - infinite memory

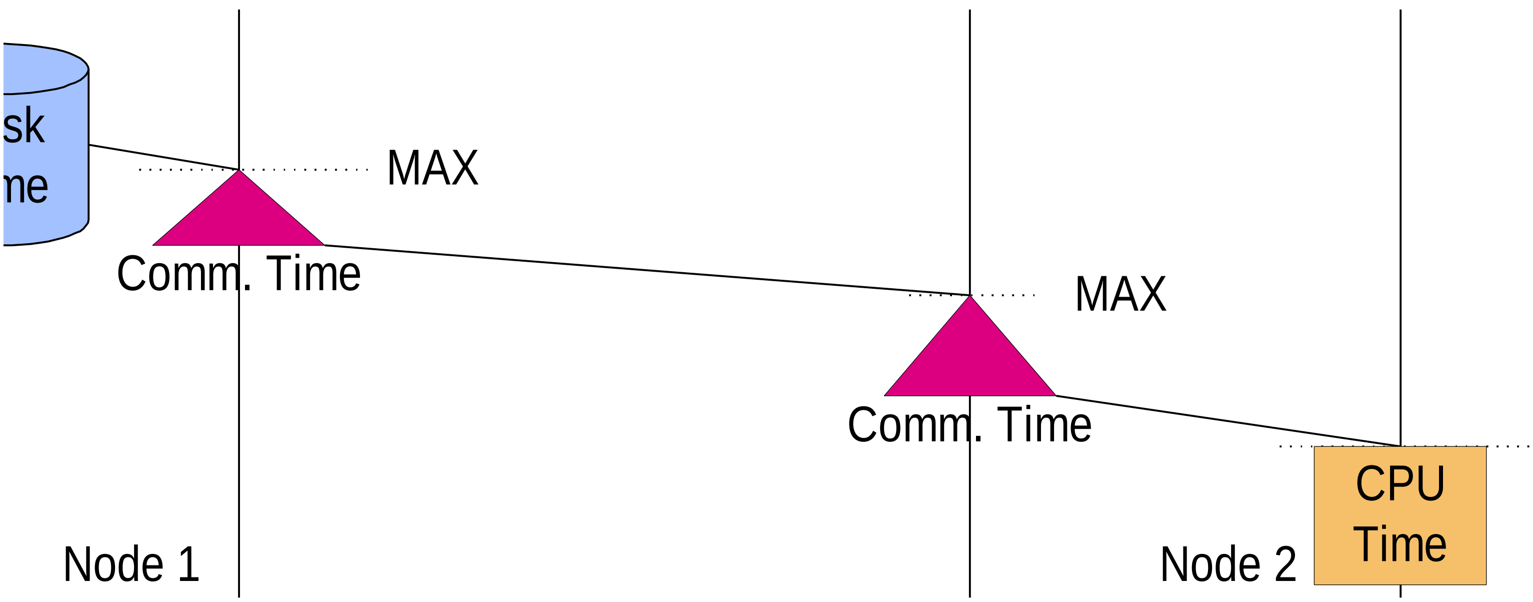
DumbSim

- Very Fast, Optimistic Simulator
  - assumes perfect overlap of I/O and computation
  - ignores block producer-consumer relationship
- Epochs used for intra-node synchronizatio
- Is embarrassingly parallel



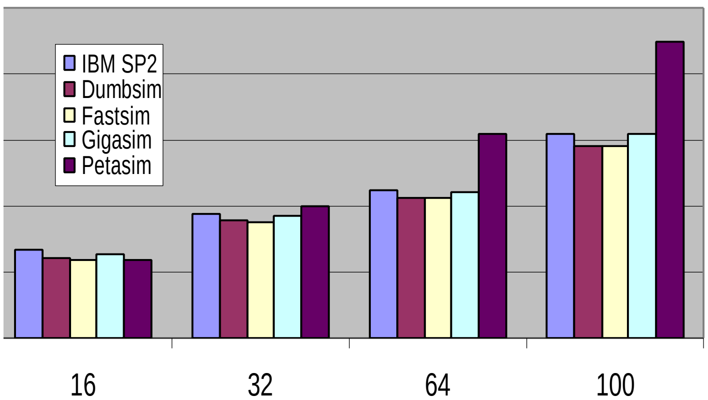
FastSim: Fast Simulator

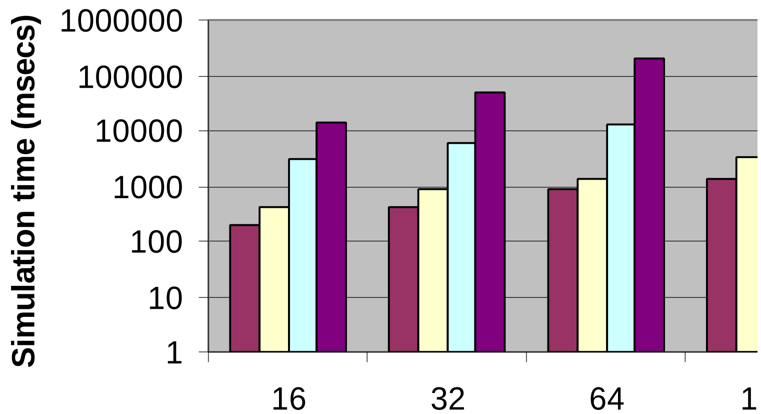
- Flexible event processing loop
  - round-robin: process next event for each node
    - most accurate when load is balanced
  - discrete event: find earliest time of next event
    - more overhead than round-robin
- Uses Graph to update timing for each resou



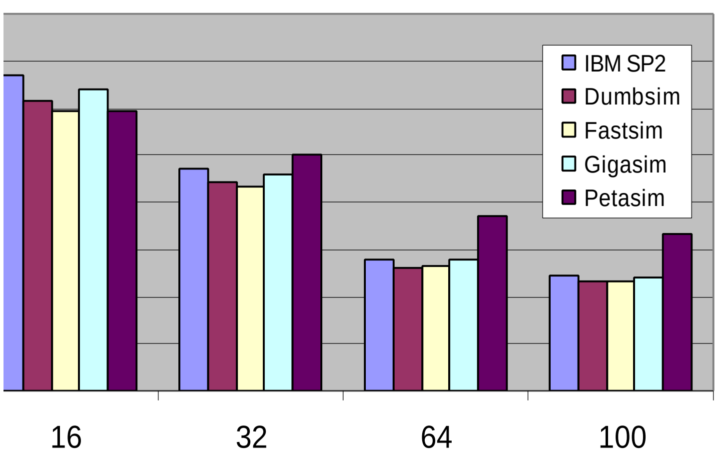
Titan Emulator (SDSC Machine)

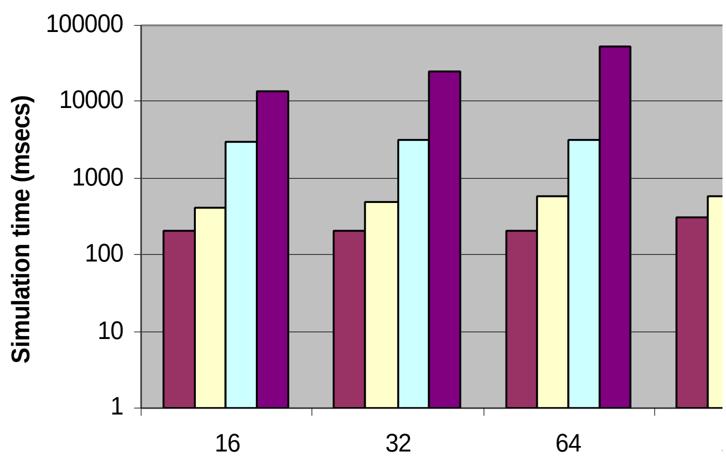
Scaled Input





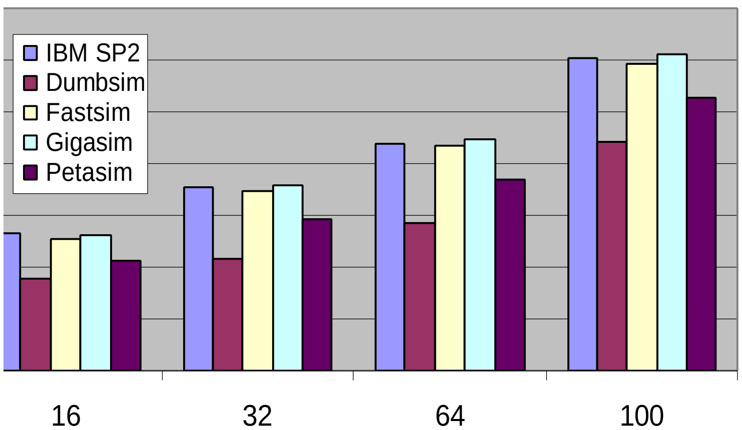
Non-scaled Input

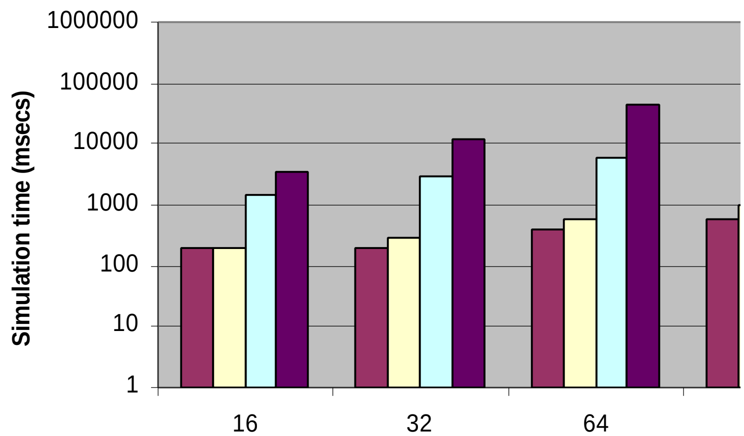




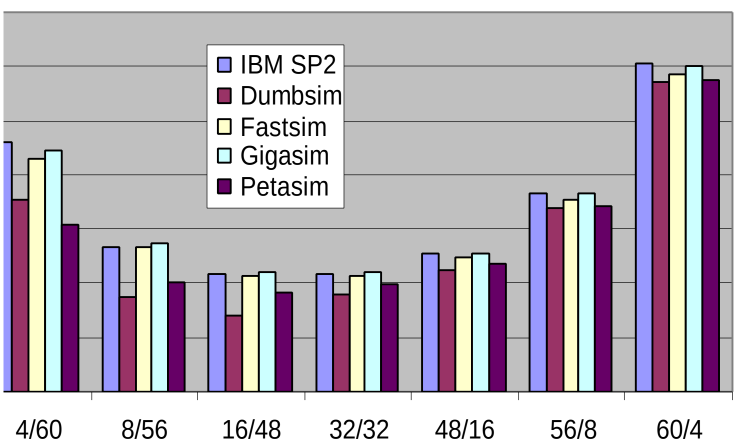
Pathfinder Emulator (SDSC Machine)

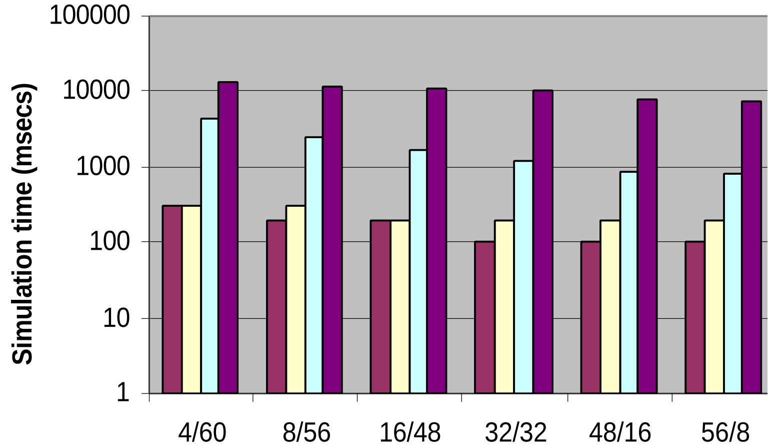
Scaled Input



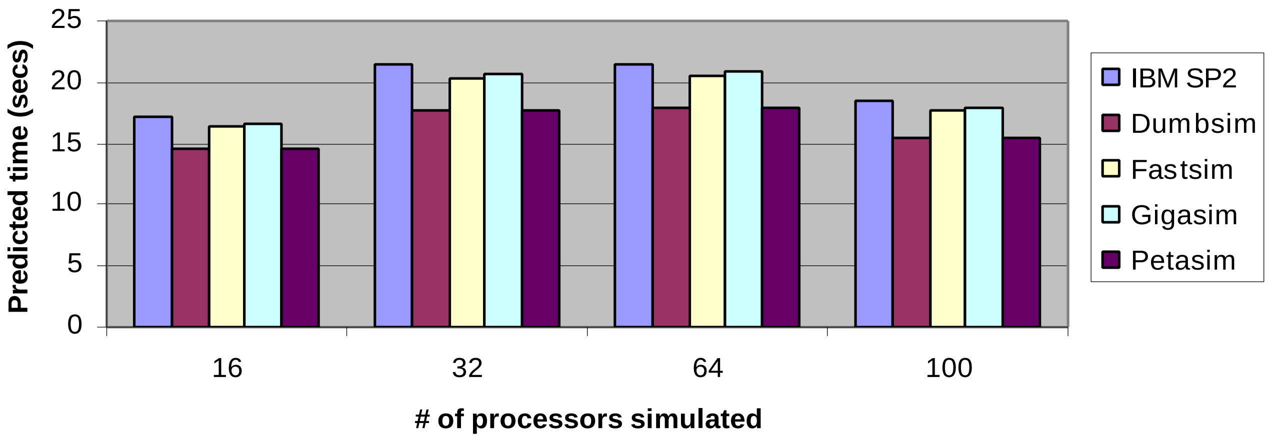


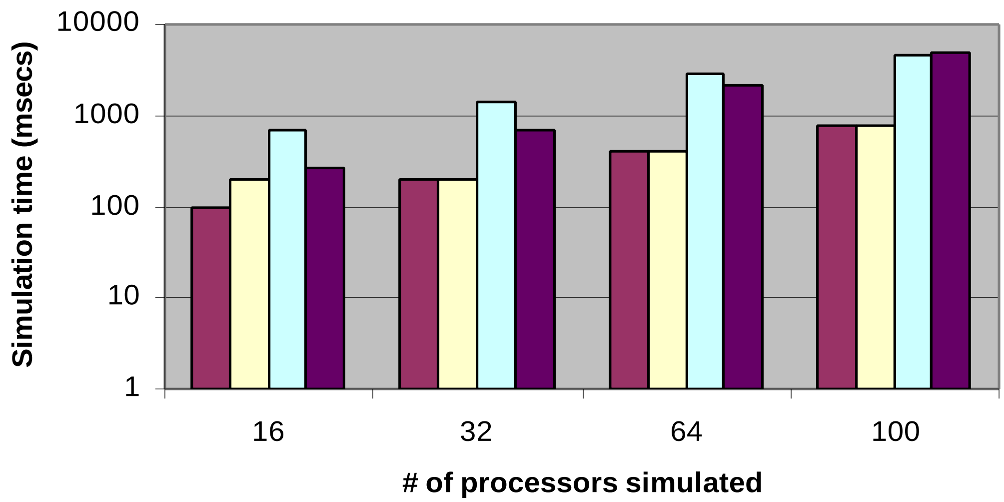
Varying IO/Compute Node Ratio





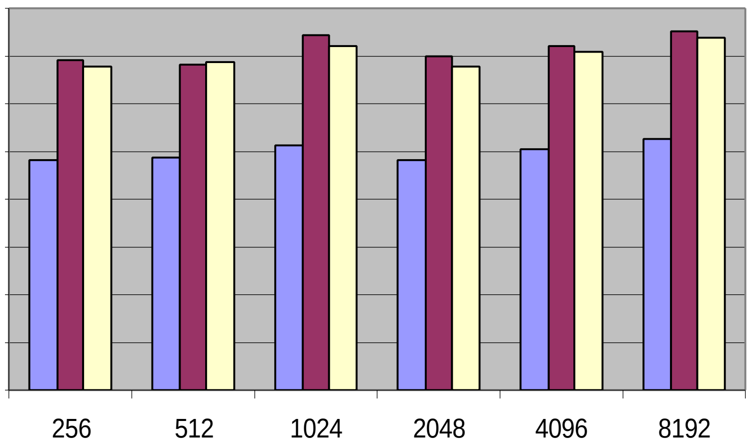
Virtual Microscope (SDSC Machine)

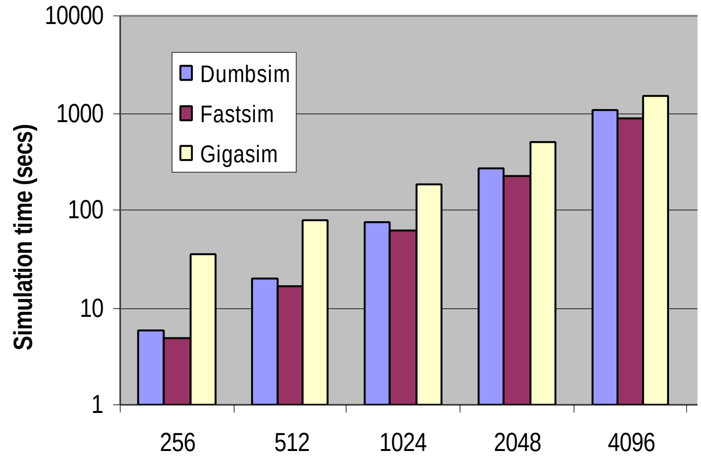




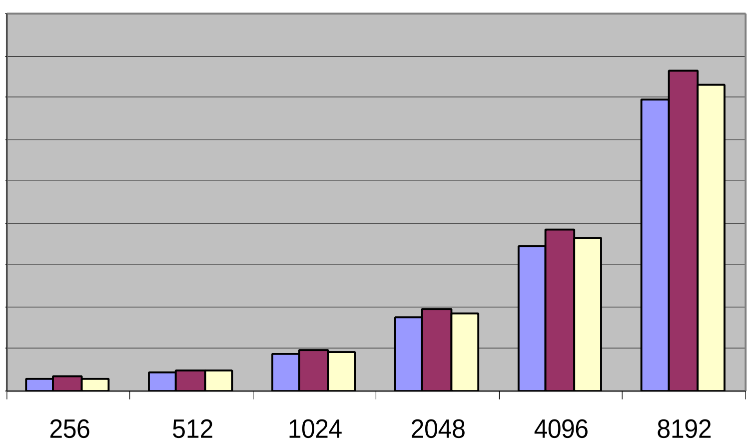
Scaling up the number of Nodes

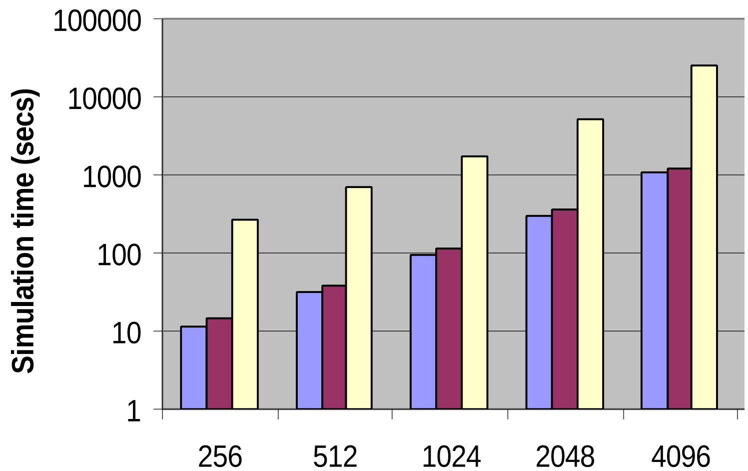
Virtual Microscope Application Emulator





Pathfinder Application Emulator





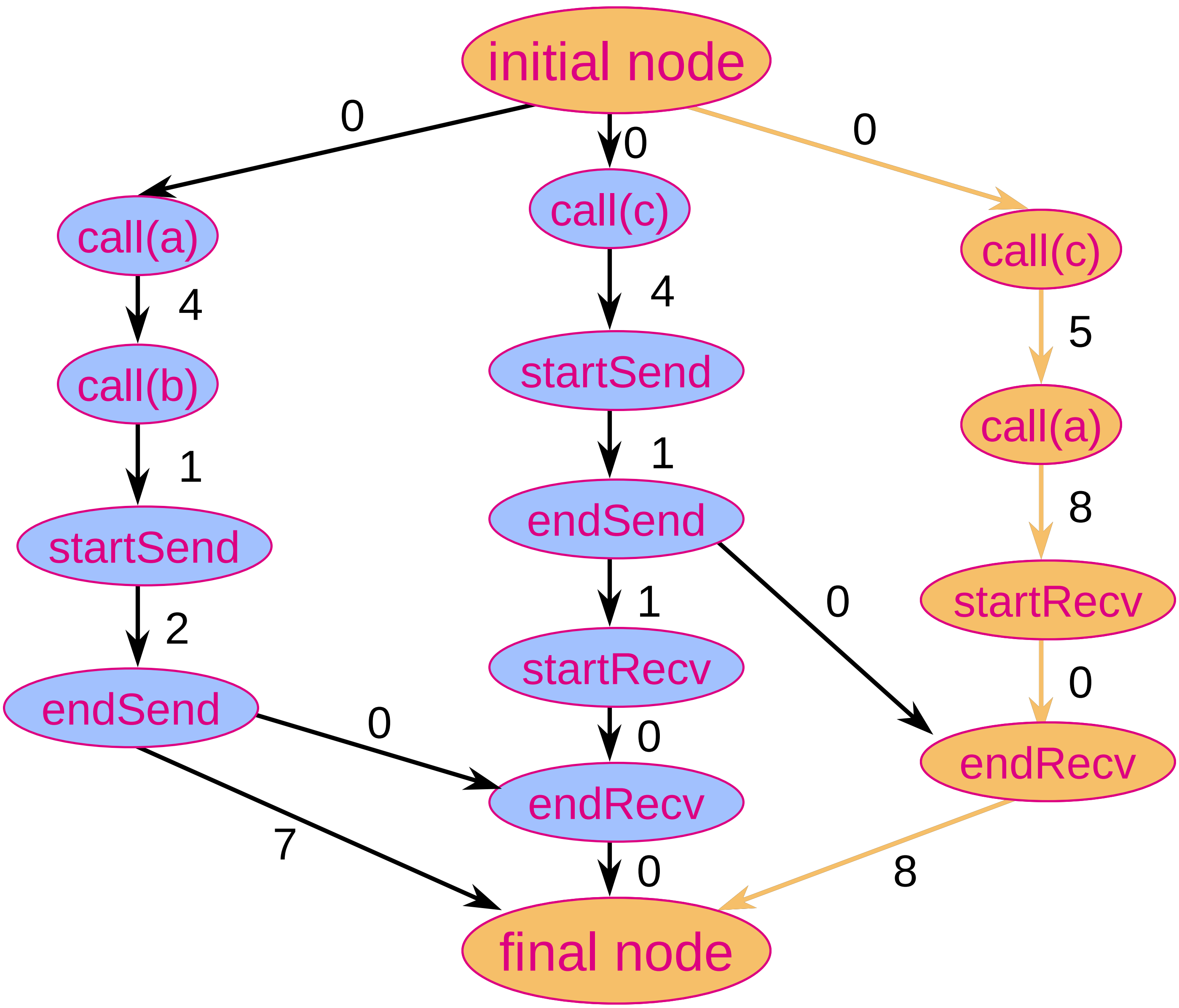
Summary of I/O Results

- Application Emulators
  - can generate complex I/O patterns quickly.
  - enable efficient simulation of large systems.
- Family of Simulators
  - permits cross checking results.
  - allows trading simulation speed and accuracy.

Critical Path Profiling

- Critical Path
  - Longest path through a parallel program
  - To speedup program, must reduce path
- Critical Path Profile
  - Time each procedure is on the critical path
- CP Zeroing
  - compute the CP as if the a procedure's time is 0.
  - use a variation of online CP algorithm
    - CP_{net} = CP - Share
    - at receive, keep tuple with largest CP_{net}

Program Activity Graph



University of Maryland

NAS IS Application

Procedure	CP	% CP	CPU	% CPU
nas_is_ben	**12.4**	**56.4**	**54.8**	**74.1**
create_seq	**9.2**	**42.0**	**9.2**	**12.4**
do_rank	**0.4**	**1.6**	**9.2**	**12.5**

create_seq is more important than CPU time indicates.

do_rank is ranked higher than create_seq by CPU time

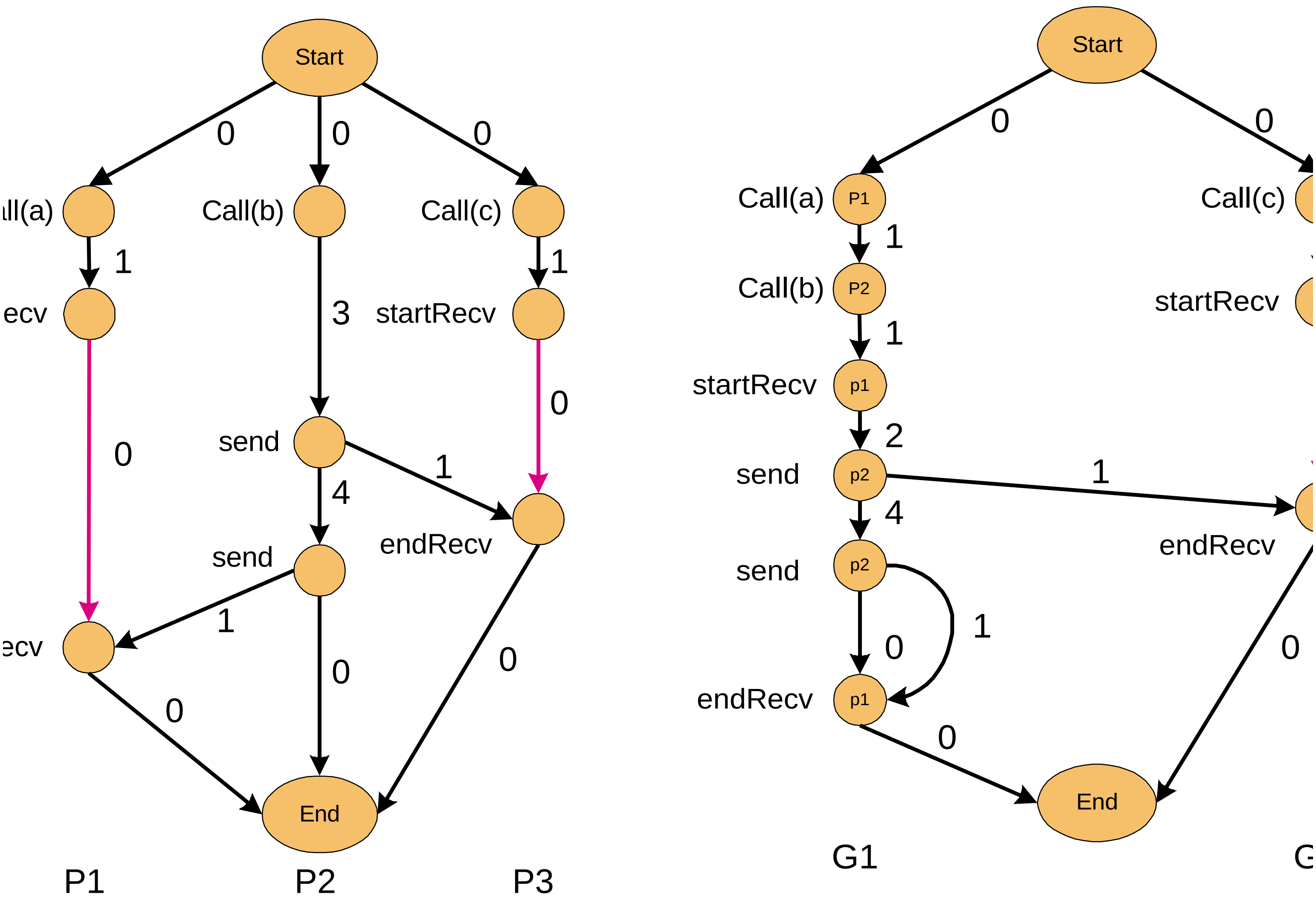
Load Balancing Factor

- Key Idea: what-if we move work
  - length of activity remains the same
  - where computation is performed changes
- Two Granularities Possible
  - process level
    - process placement or migration
  - procedure level
    - function shipping
    - fine grained thread migration

Process LBF

- What-if we change processor assignment
  - predict execution time on larger configurations
  - try out different allocations
- Issues:
  - changes in communication cost
    - local vs. non-local communications
  - interaction with scheduling policy
    - how are nodes shared?
    - assume round robin

Computing Load Balancing Factor



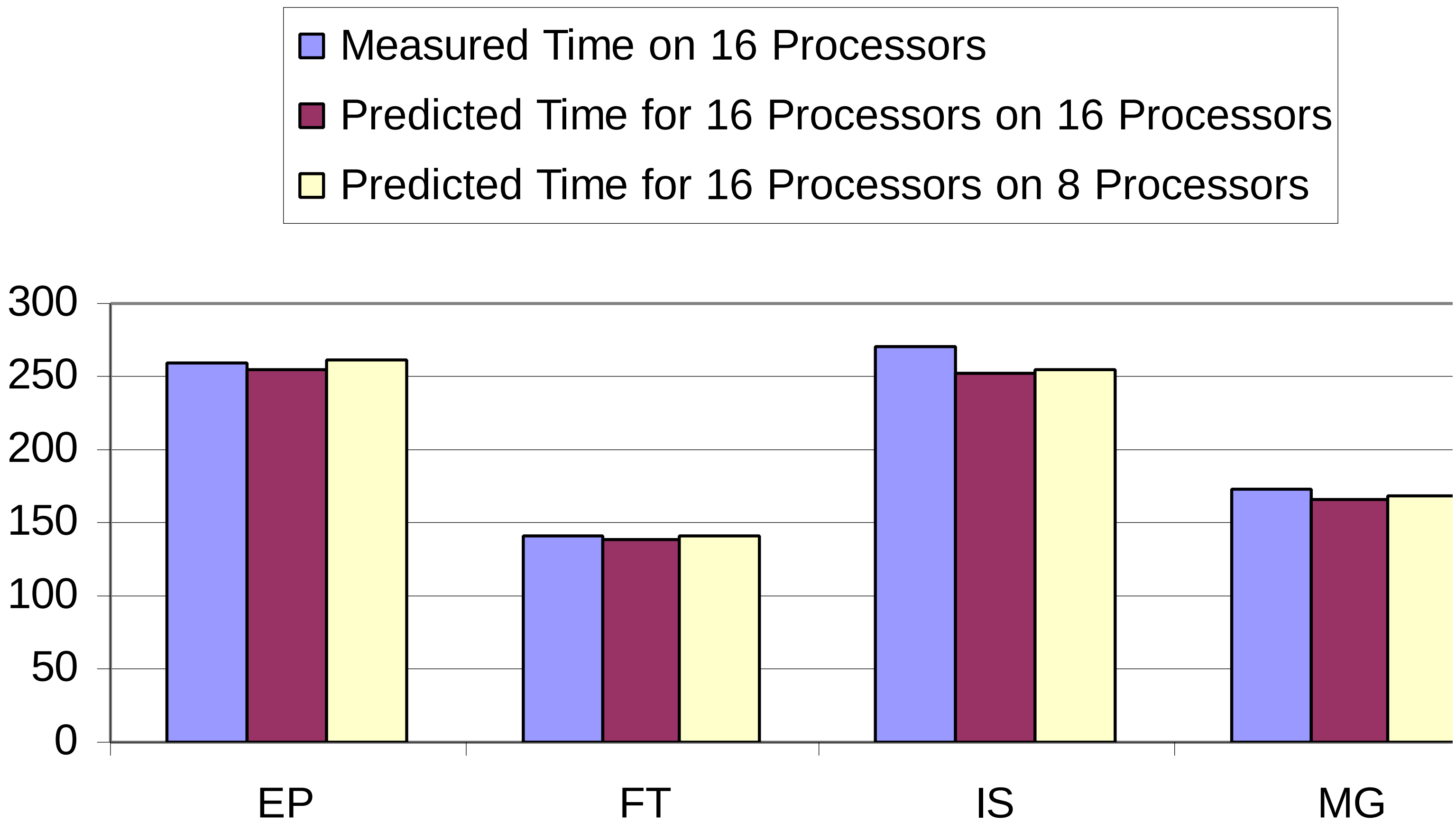
Program Activity Graph

Group Activity Graph

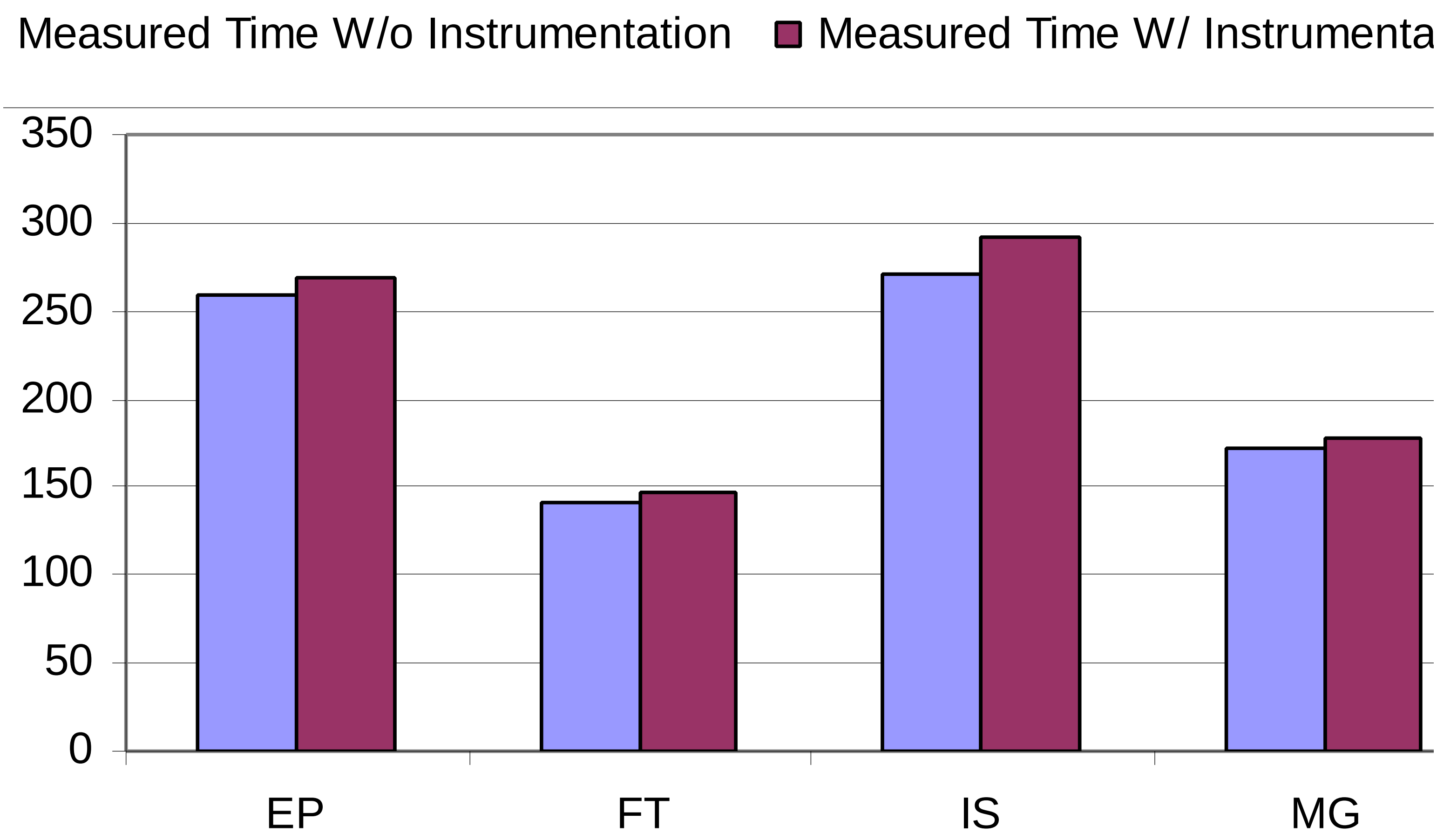
Jsing Paradyn to Implement Process LBF

- forward data from application to monitor
- Need to forward events to central point
  – supports samples
  – requires extensions to data collection system
- provides dynamic control of data collectior
  – only piggy pack instrumentation on demand
- need to correlate data from different nodes
  – use $globalId MDL variable

Results : Accuracy



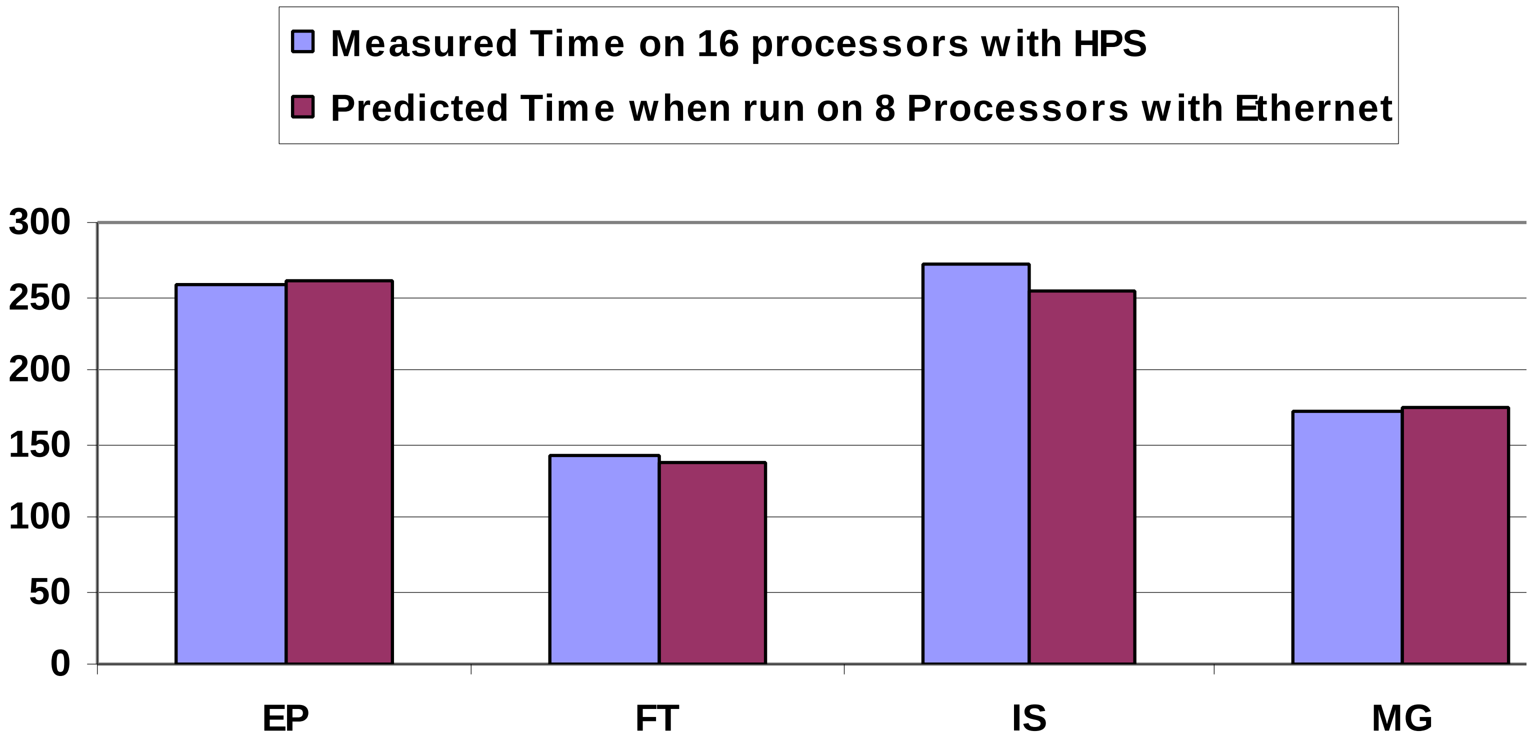
LBF Overhead (16 nodes)



Changing Network and Processes

Change: # of nodes (8->16)

network (10Mbps Ethernet -> 320Mbps HPS)



Linger Longer

- Many Idle Cycles on Workstations
  - Even when users are active, most processing pow
    not used
- Idea: Fine-grained cycle stealing
  - Run processes a very low priority
  - Migration becomes an optimization not a necessity
- Issues:
  - How long to Linger?
  - How much disruption of foreground users
    - delay of local jobs: process switching
    - virtual memory interactions

Simulation of Policies

Model workstation as

- foreground process (high priority)
  - requests CPU, then blocks
  - hybrid of trace-based data and model
- background process (low priority)
  - always ready to run, and have a fixed CPU time
- context switches (each takes 100 micro-seconds)
  - accounts for both direct state and cache re-load

Study:

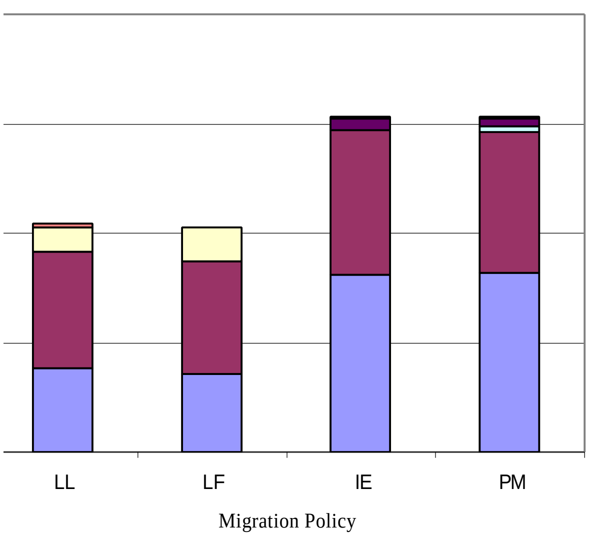
- What is the benefit of Lingering?
- How much will lingering slow foreground processes?

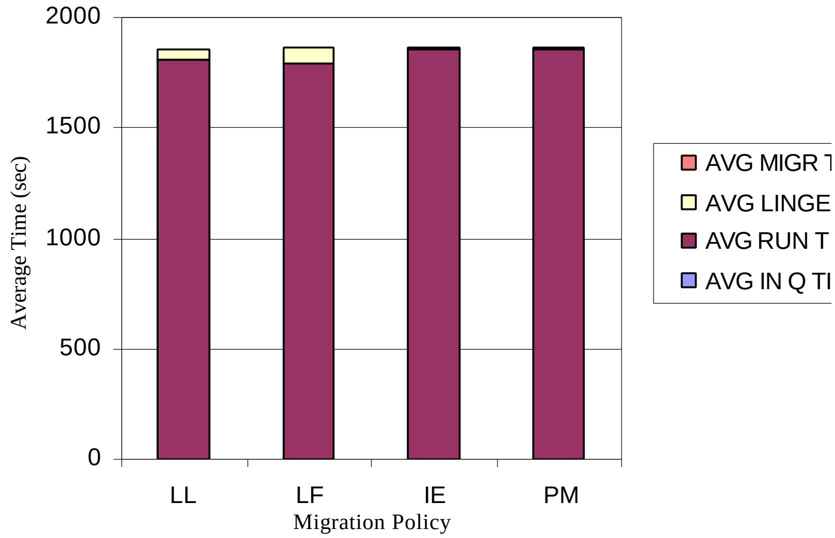
Migration Policies

- Immediate Eviction (IE)
  - when a user returns, migrate the job
  - policy used by Berkeley NOW
  - assumes free workstation or no penalty to stop job
- Pause and Migrate (PM)
  - when a user returns, migrate the job
  - used by Wisconsin condor
- Linger Longer (LL)
  - when user returns, decrease priority and remain
    - monitor situation to decide when to migrate
  - permits fine grained cycle stealing
- Linger Forever (LF)
  - like Linger Longer, but never migrate

Simulation Results - Sequential Workload

– LF is fastest, but variation is higher than LL
– LL and LF have lower variation than IE or PM.
– Slowdown for foreground jobs is under 1%.





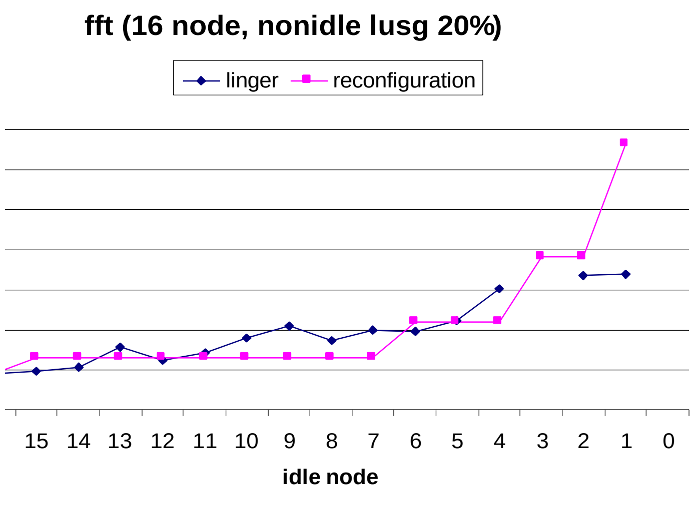
– LF is a 60% improvement over the PM policy.

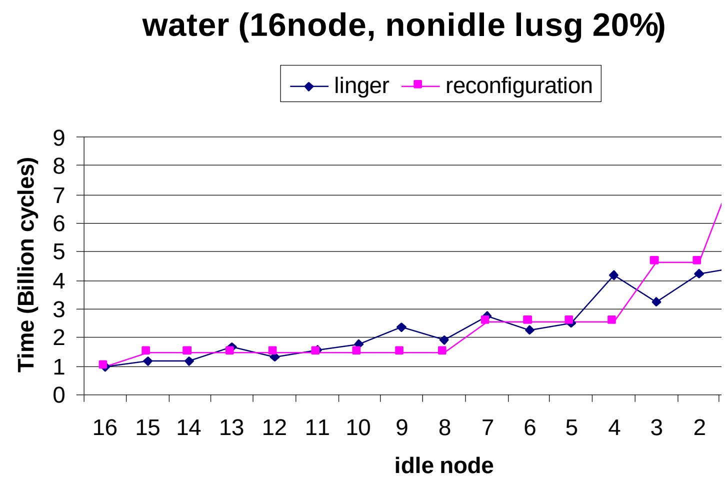
Simulation Results - Parallel Applications

– Use DSM Applications on non-idle workstations

– Assumes 1.0 Gbps LAN

– Compare Lingering vs. reconfiguration





– **Lingering is often faster than reconfiguration!**

Future Directions

/ide Area Test Configuration

- simulate high latency/high bandwidth network
- a controlled testbed for wide area computing

arallel Computing on non-dedicated cluster

- current simulations show promise, but ...
  - need to include data about memory hierarchy
  - real test is to build the system

evelopment of the Metric and Option Interfa

- prototype applications that can adapt to change
- evaluate different adaptation policies