Volunteer Computing on Clusters

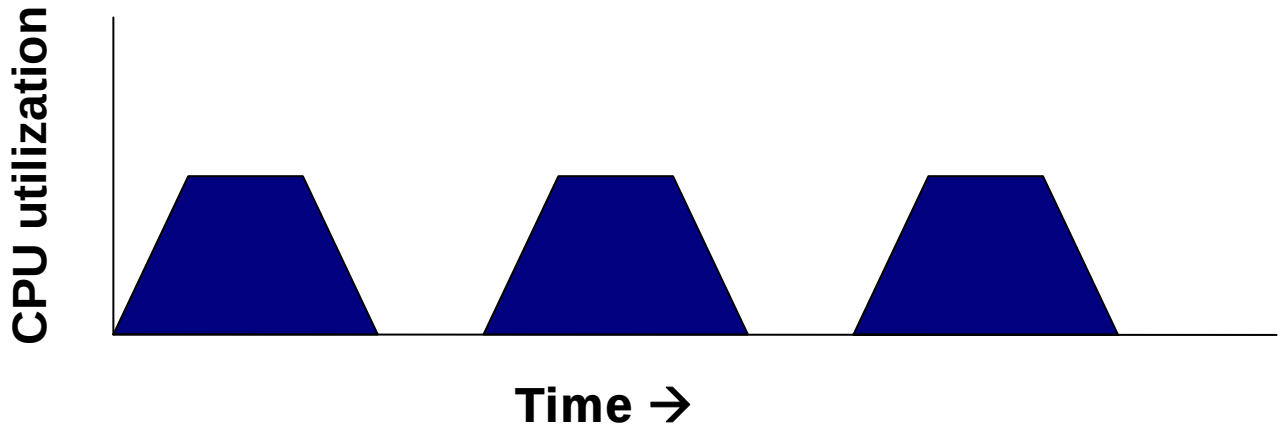
Deepti Vyas* & Jaspal Subhlok

University of Houston

***Currently with Halliburton**

Idea of Volunteer Computing

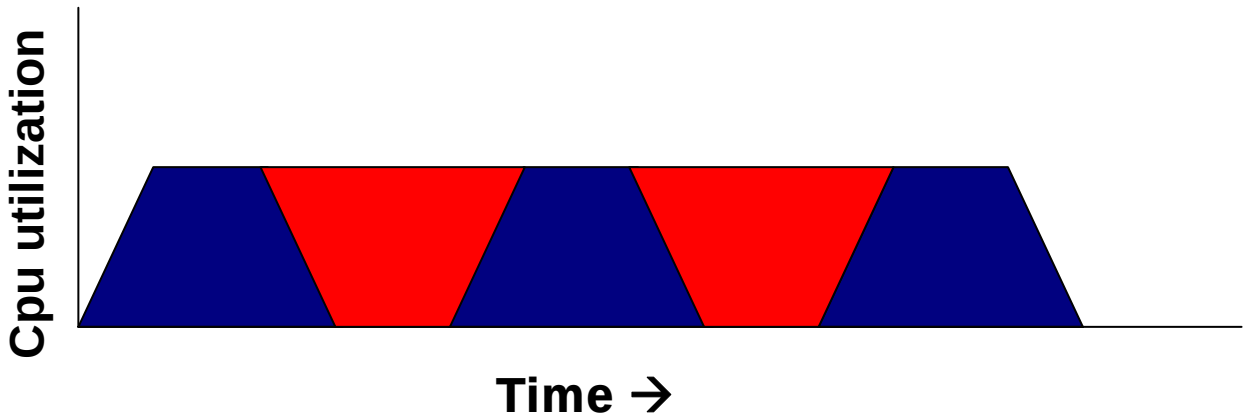
- aka *global computing* or *public resource* computing
- Perform computations by exploiting unused cycles:



Sample execution of a HOST application

Volunteer Computing

- **Run another GUEST application simultaneously with the HOST application**



- **Guest exploits idle cycles**
- **No impact on host execution**

Volunteer Computing Today

- Exploit idle compute cycles to solve large scale (scientific) applications.
  - Primarily “embarrassingly parallel” or “bag of tasks” applications
- Volunteer Computing Systems
  - BOINC: Compute time donated by public on PCs
    - SETI@Home (1 million PCs) , Protein folding, Climate Prediction, …
  - CONDOR: Idle desktops in an organization
  - ENTROPIA: Commercial product

Volunteer Computing on Clusters

Compute Clusters are a large source of CPU cycles

For volunteer computing:

- **Advantages**
  - **Homogeneous groups of high performance nodes**
  - **Maintained by IT professionals**
  - **Always running**
  - **High interconnectivity between nodes**
- **Disadvantages**
  - **They are always busy!**

Contributions of this Work

Address the following questions:

- **Pattern and extent of unused cpu cycles and memory on compute clusters ?**
- **Can they be exploited for guest applications without impacting the main host applications ?**

Availability of CPU Cycles on Clusters

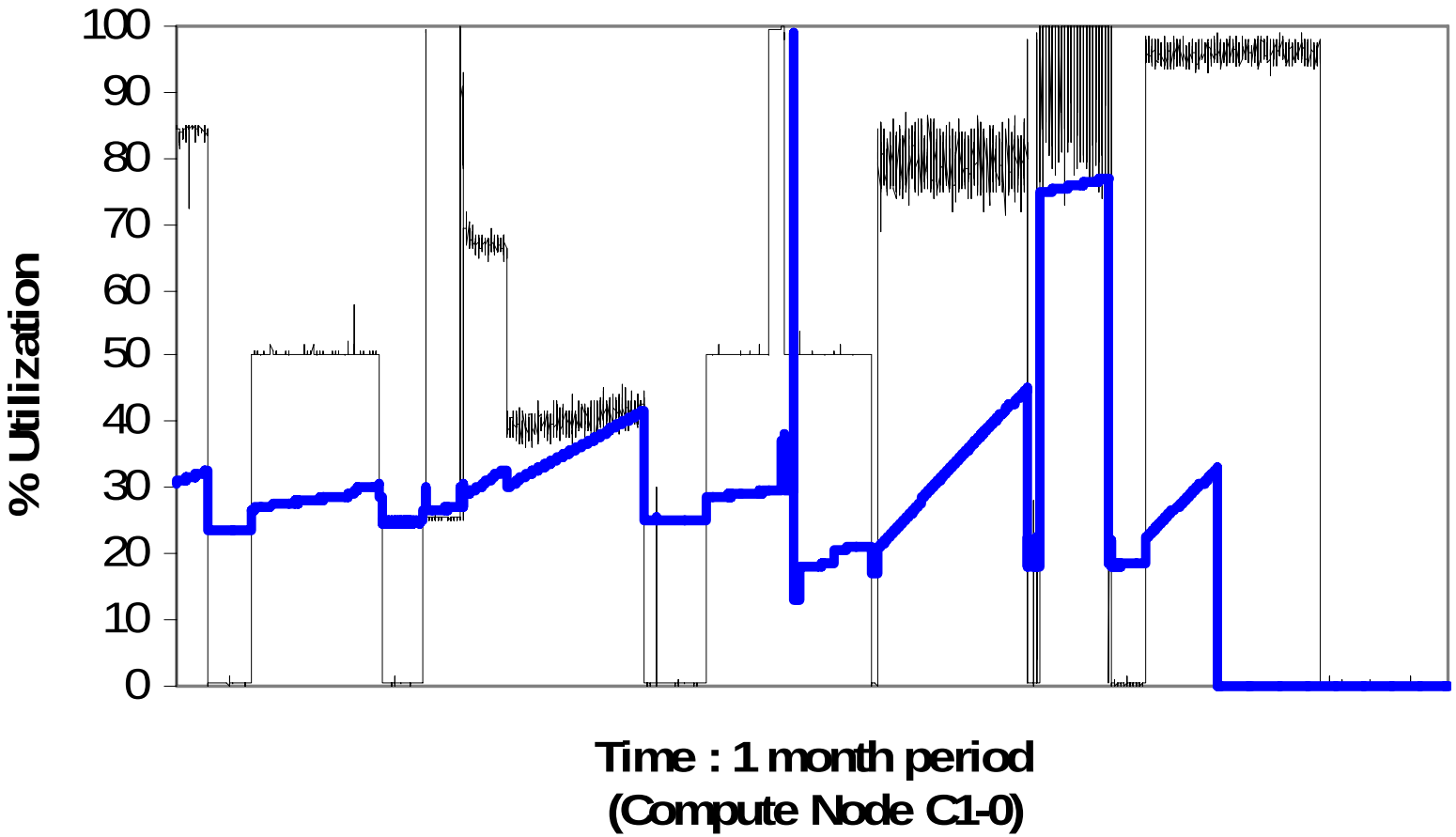
- **Clusters vary widely in usage**
  - **many are used for computing 100% of the time**
  - **Others may not be: a group of research clusters in a recent study varied 7-22% in usage**
- **… And when they are busy executing applications:**
  - **What fraction of cpu cycles and memory are unused ?**
  - **What are the usage patterns ?**

POINT: Idle cluster can be trivially used for volunteer computing. Can "busy" clusters also be used ?

Empirical Study of CPU / Memory Usage on a Cluster

- Data Collected from a busy cluster at University of Houston
  - 30 Node Beowulf cluster - Intel Xeon Dual processor nodes with 2 Gb RAM, 1 Gbps ethernet network
- CPU and memory usage and availability monitored
  - Information source was */proc* filesystem
  - Data collected every 5 minutes over 1 month
- Usage graphs for a 1 month period (July 2005) coming up!

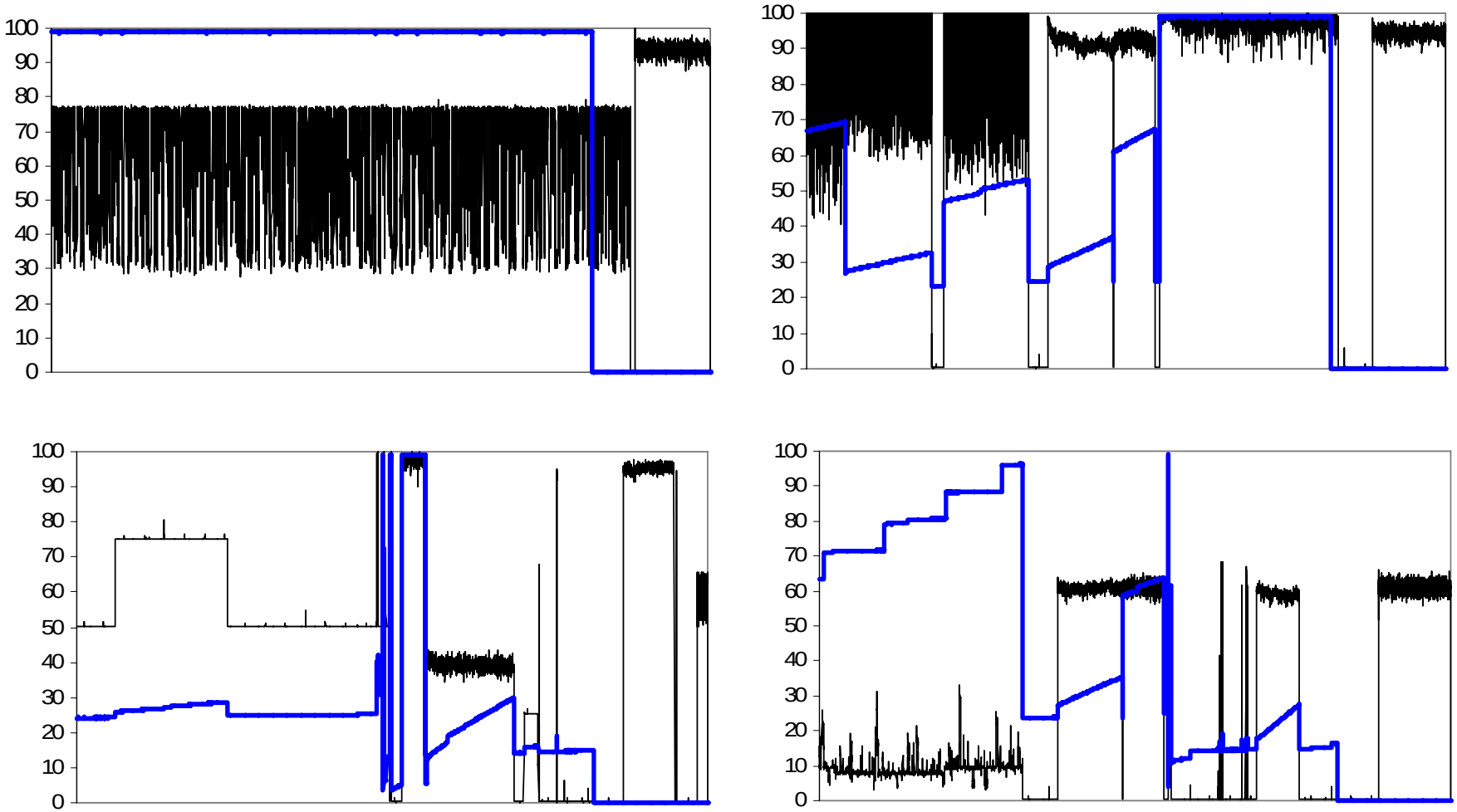
Sample Cluster Node Usage



CPU utilization / Memory Utilization

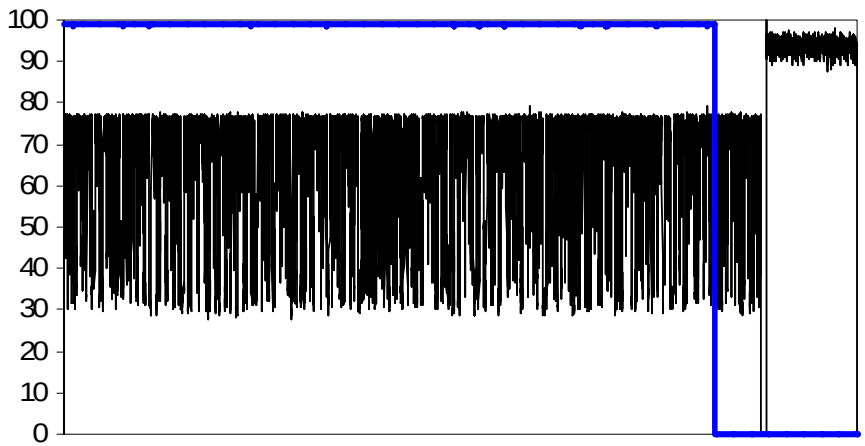
CS@UH

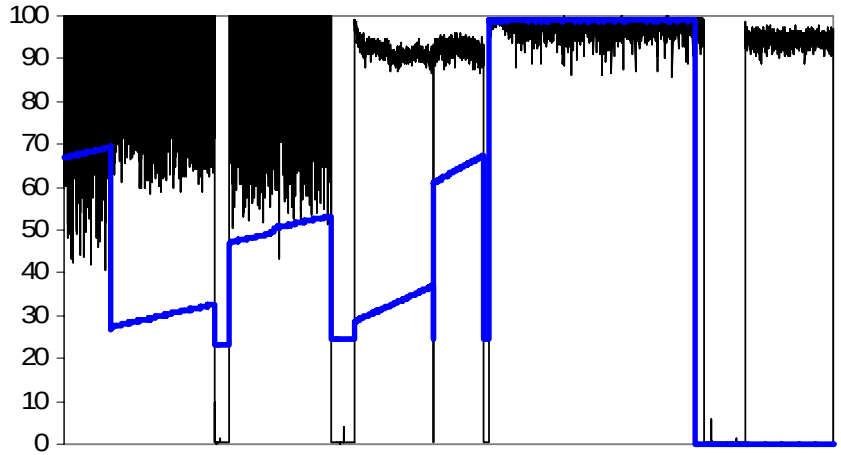
Usage of Representative Nodes



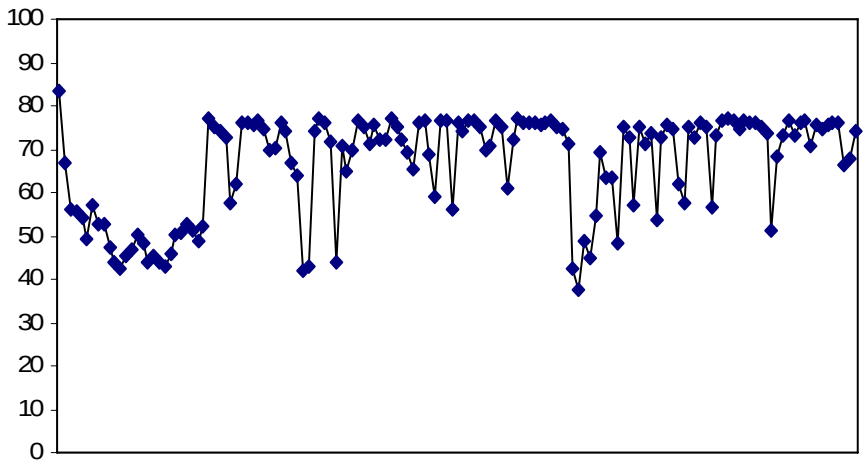
CPU utilization / Memory Utilization

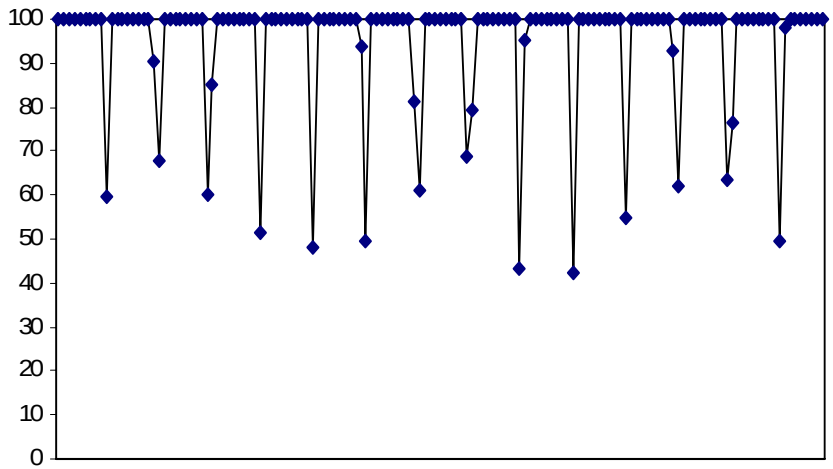
CPU Usage on different time scales





TOTAL 1 MONTH PERIOD





TOTAL 12 HOUR PERIOD FROM THE BEGINNING OF ABOVE GRAPHS

Usage Experiment Observations

- **CPU Utilization varies, with monthly average over a node varying from 25% to 85%**
- **Memory usage also varies – average between 30% and 90% for nodes**
- **Stability over windows of hours to days- steady or a slow climb (for memory)**

Mini conclusion: long and predictable periods of CPU and memory underutilization could be used for volunteer computing even when nodes are "busy"

Part 2 of Talk:
Is Fine Grain Cycle-Stealing practical ?

Processor may have unused cycles (typically host process blocked on I/O), at a fine grain (msecs)

- **Can they be used for guest applications ?**
- **Would this slow down the main host application ?**
  - **Is the slowdown acceptable ?**

APPROACH: Empirical measurements to gain insight. Focus on measuring/minimizing host application slowdown.

Experiments Overview

Step 1 : **Host application executed in dedicated mode**

Step 2 : **The Host application executed in shared mode with a Guest application *at lowest priority***

Then **Slowdown of Host application due to cycle stealing by Guest application is computed:**

$$\textit{Percentage Slowdown} = (Ts - Td) / Td * 100$$

Ts – Execution time in shared mode

Td – Execution time in dedicated mode

Experiments on small (10 dual nodes) Linux cluster.

NAS benchmarks used as host/guest applications

Experimental Setup

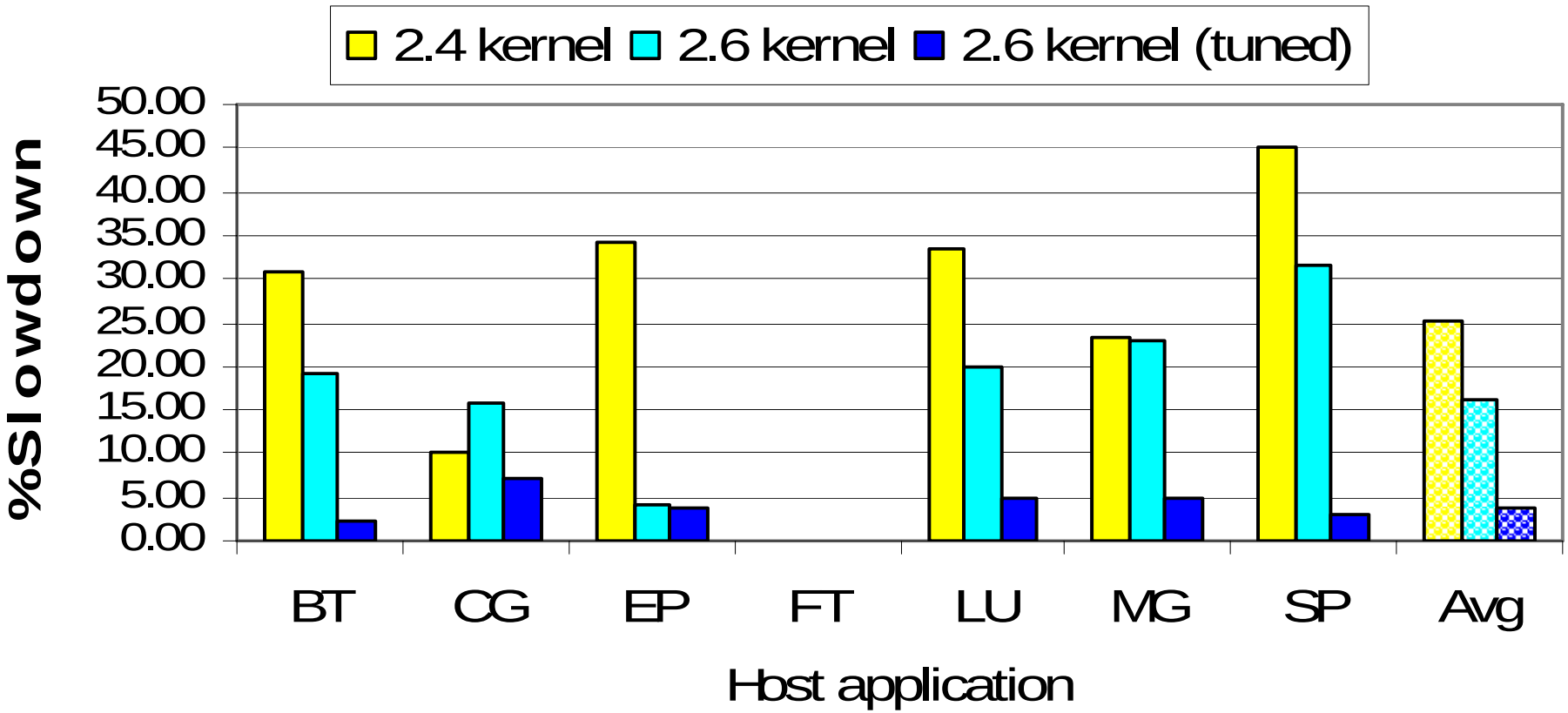
GOAL: Measure slowdown of parallel host applis due to a (sequential) guest application:

- **Number of nodes = 4 (8 processors)**
- **Host applications: NAS Class B benchmarks**
- **Guest application: NAS EP benchmark ("sequential")**
- **Host application threads = 8 (2 per node)**
- **Guest application threads = 4 (1 per node)**
- **Priority of Host application = Normal (nice = 0)**
- **Priority of Guest application = Lowest (nice = 19)**

- **Linux 2.4 and Linux 2.6 kernels**

CS@UH

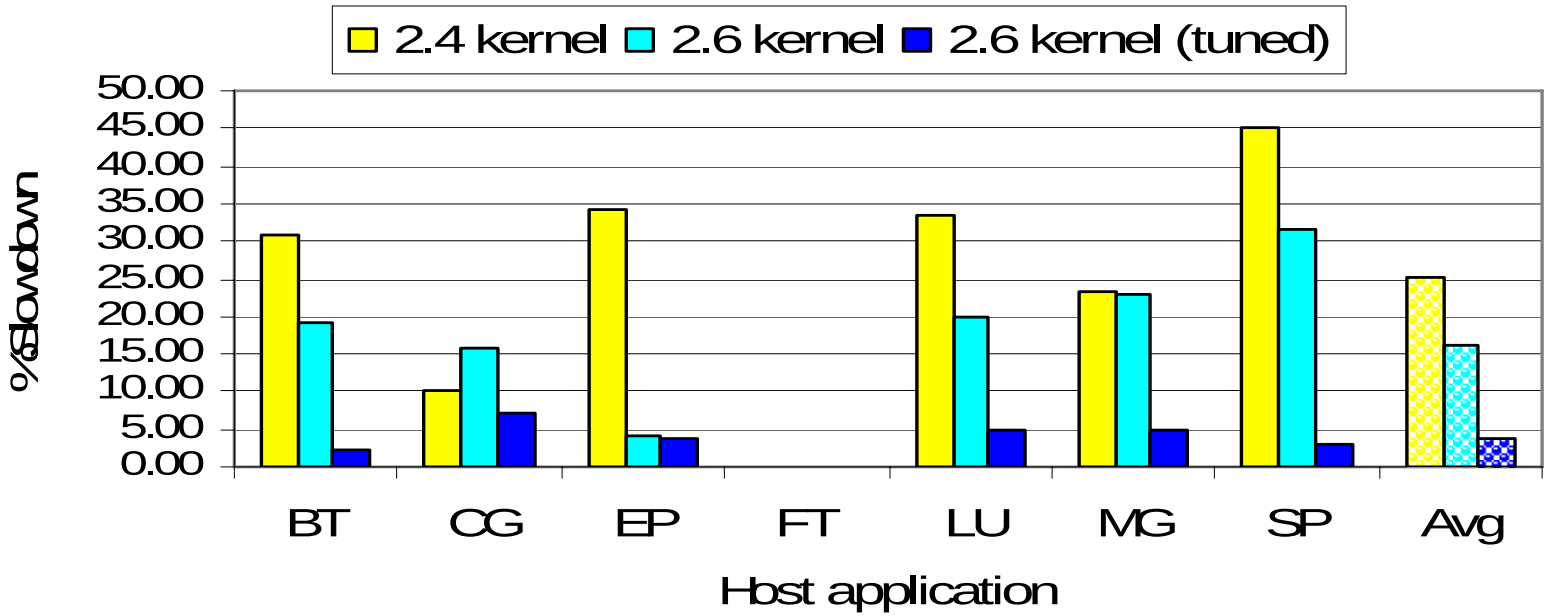
Slowdown on different OS Kernels



"Tuning": Changing the load balance frequency among CPU queues from 200 msecs to 10 msecs.

CS@UH

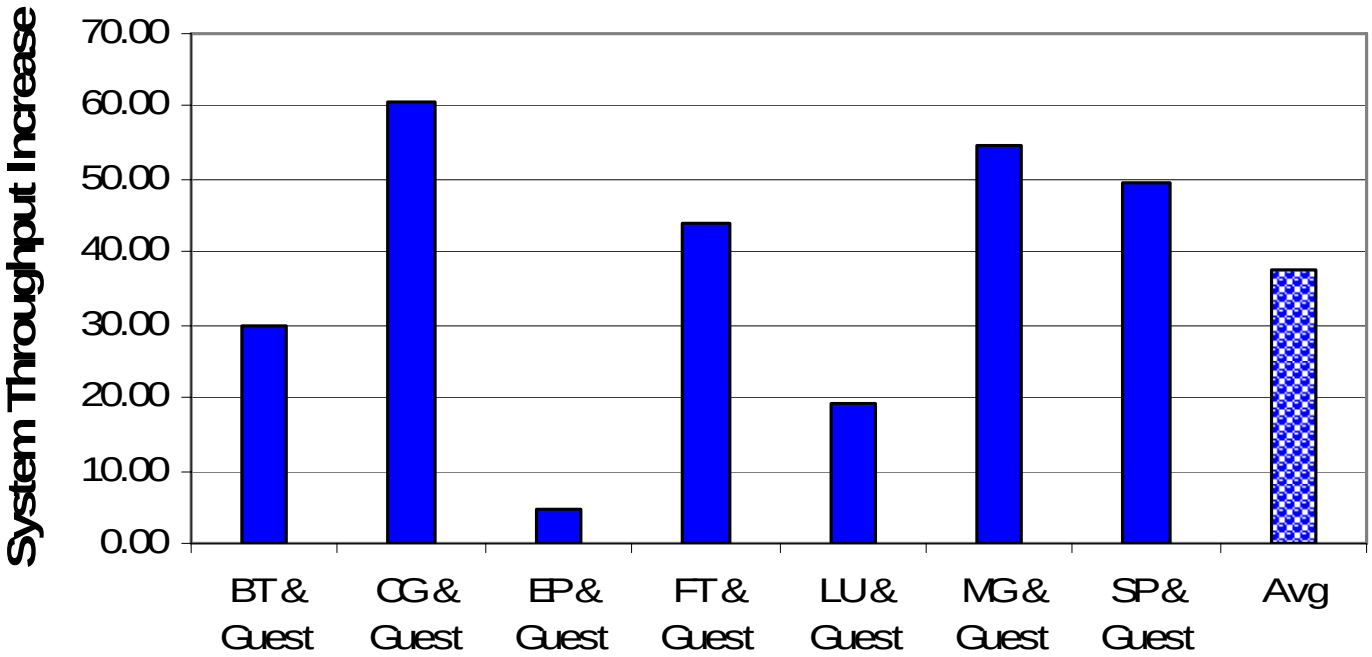
Observations:
Slowdown on different OS Kernels



- **Slowdown with regular Linux is unnacceptably high, although lower with 2.6 kernel**
- **Slowdown with "tuning" typically < 5 % (avg 3.8 %). Not zero but could be tolerable**

Benefit to Guest Application

Measure increase in *normalized* system throughput with a guest app vs dedicated host app execution

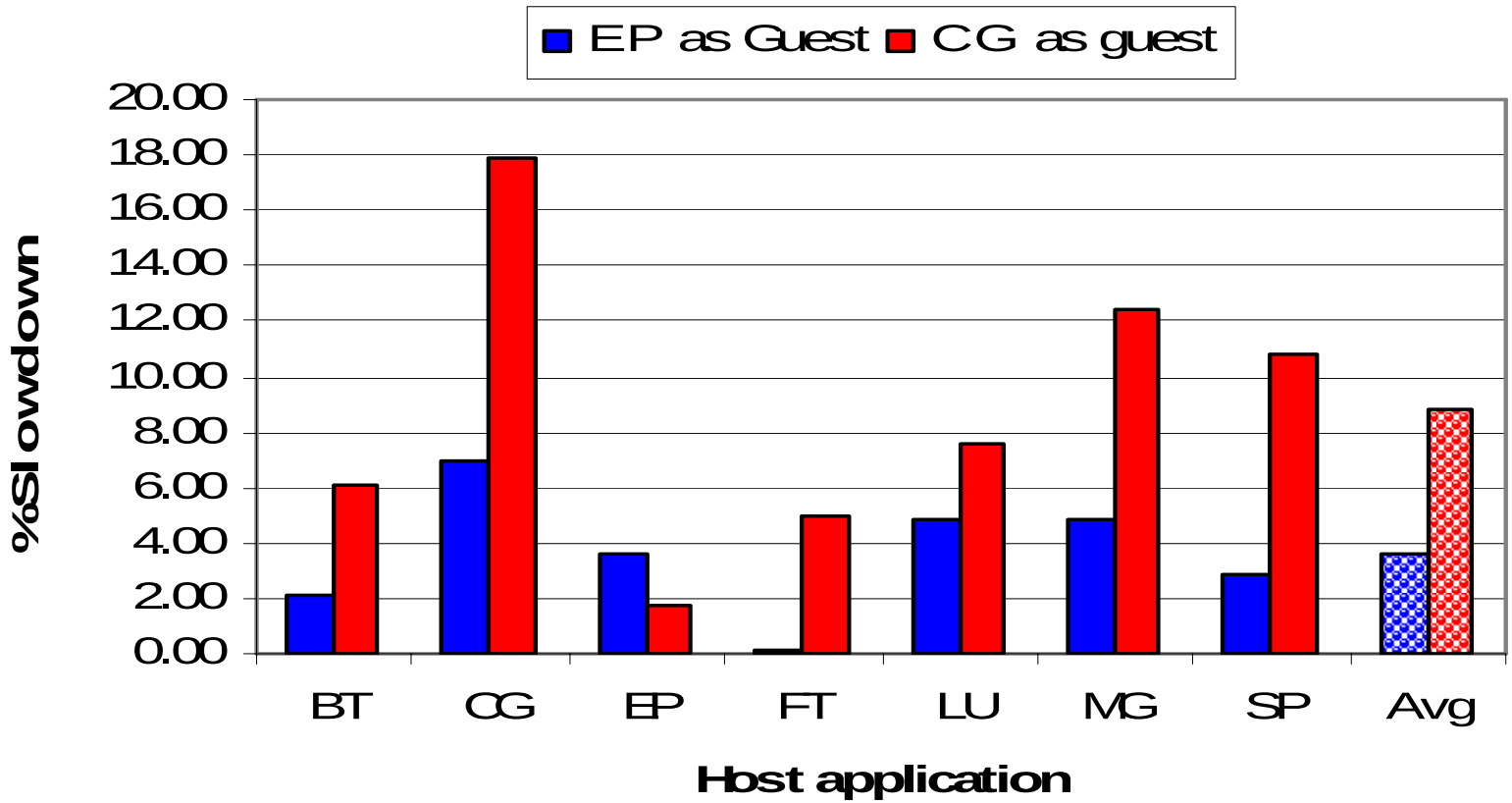


Average improvement around 38%

(progress of guest app – slowdown of host app)

Parallel Guest Application

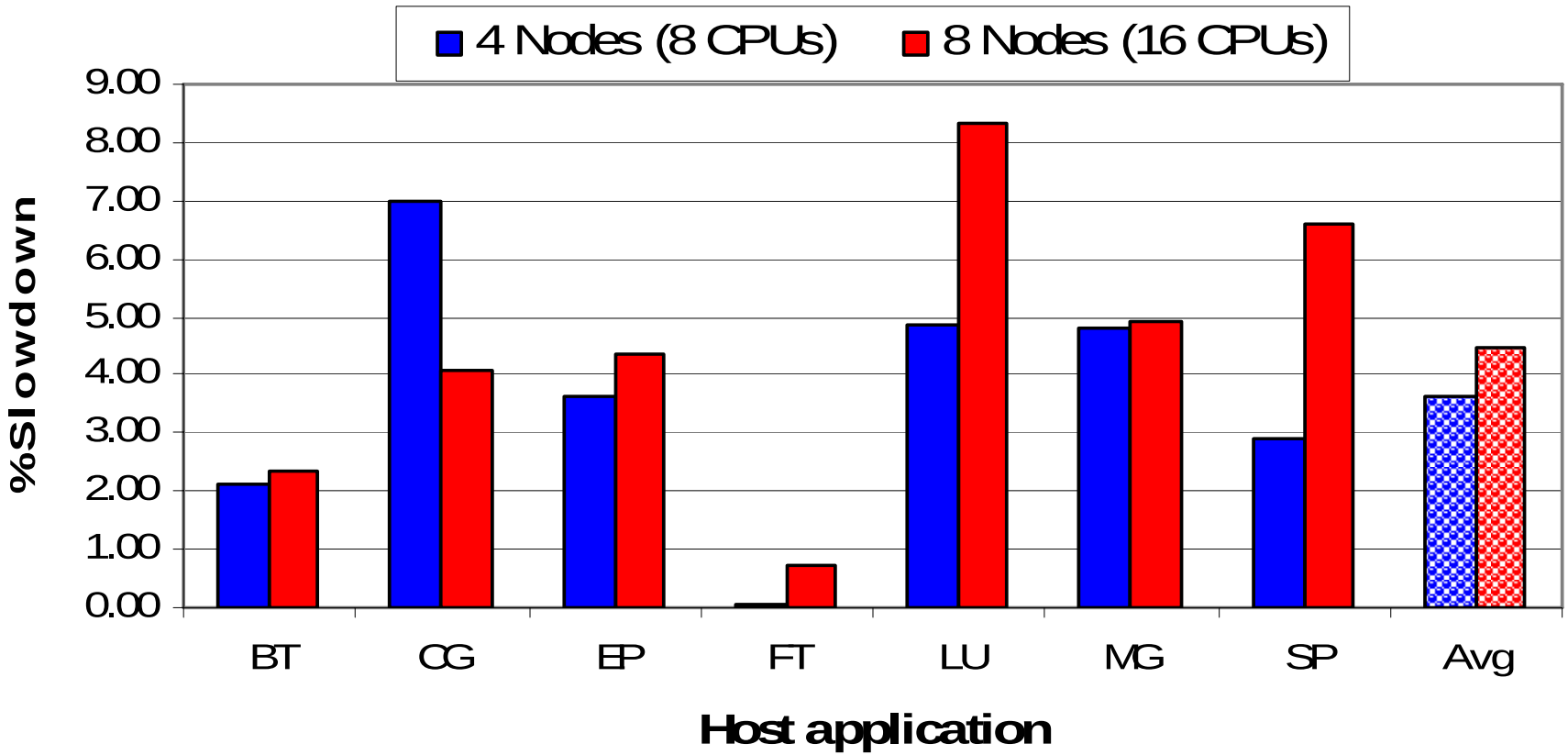
Parallel App CG as guest versus sequential EP



Average slowdown increases to ~9 %

Scaling Behavior

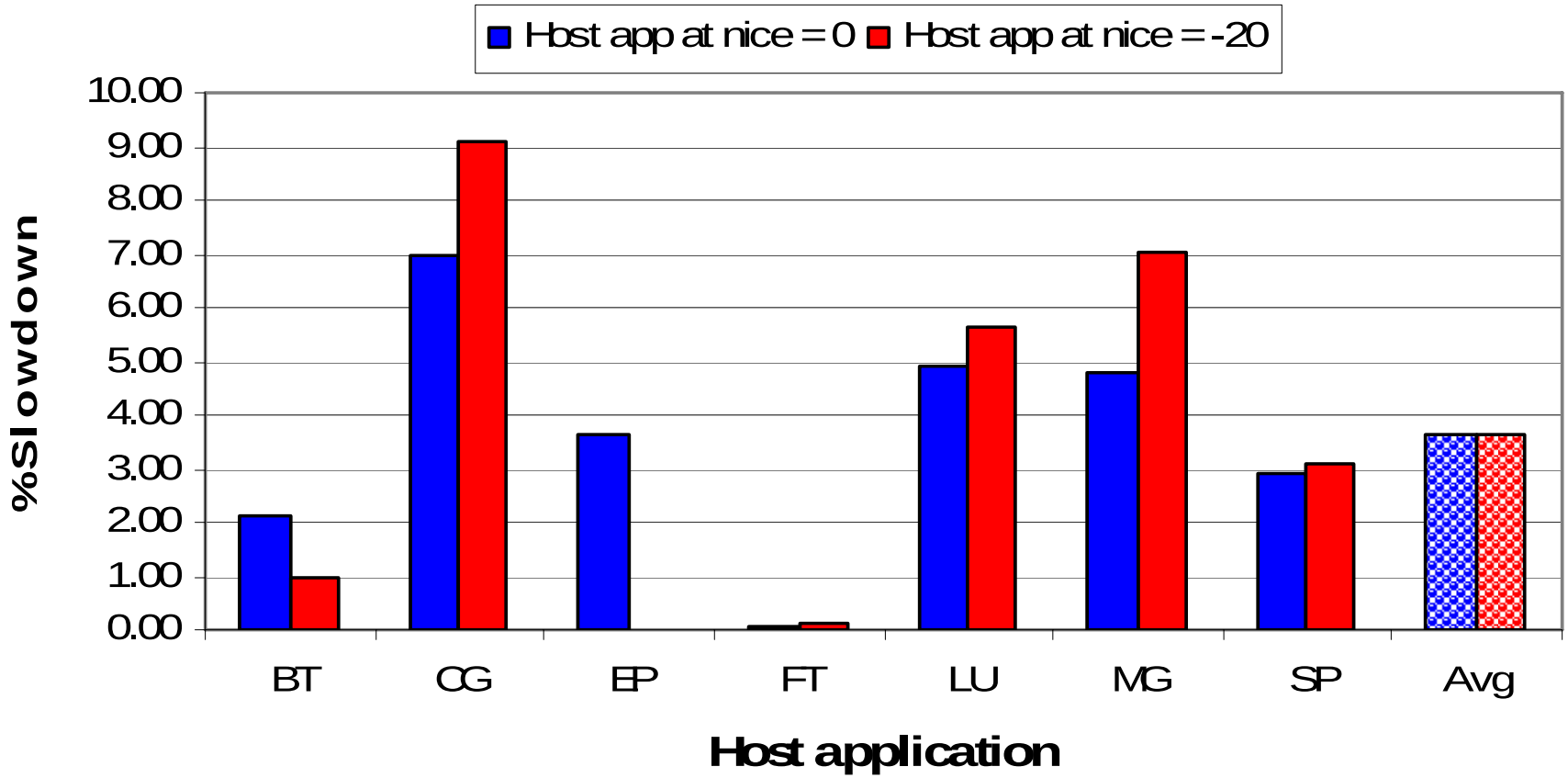
Employ 8 nodes (16 threads) versus 4 nodes (8 threads)



Avg slowdown increases modestly – 3.6% to 4.5%

Raising Priority of Host

Increase priority of host application: Normal to Highest



Overall tie – slowdown increases for some apps!

CS@UH

Discussion

- Clusters have unused CPU and memory resources
  - Beside idle time, resources are often underused

- Utilizing busy clusters for volunteer computing is a challenge with current Linux
  - Some tuning necessary for acceptable behavior
  - Even slowdowns < 5 % are an issue
  - Scalability needs to be investigated further
  - Performance with parallel guests discouraging
    - But most guests today are “sequential”

Conclusive Discussion

- Paper offers some basic guidelines to employ volunteer computing on clusters
  - Summary – do it when CPU is relatively idle and enough memory is available

- Support for Zero Priority Processes that always yield to other higher priority processes will go a long way in solving these problems
  - Current schedulers too worried about starvation

Conclusive Conclusions

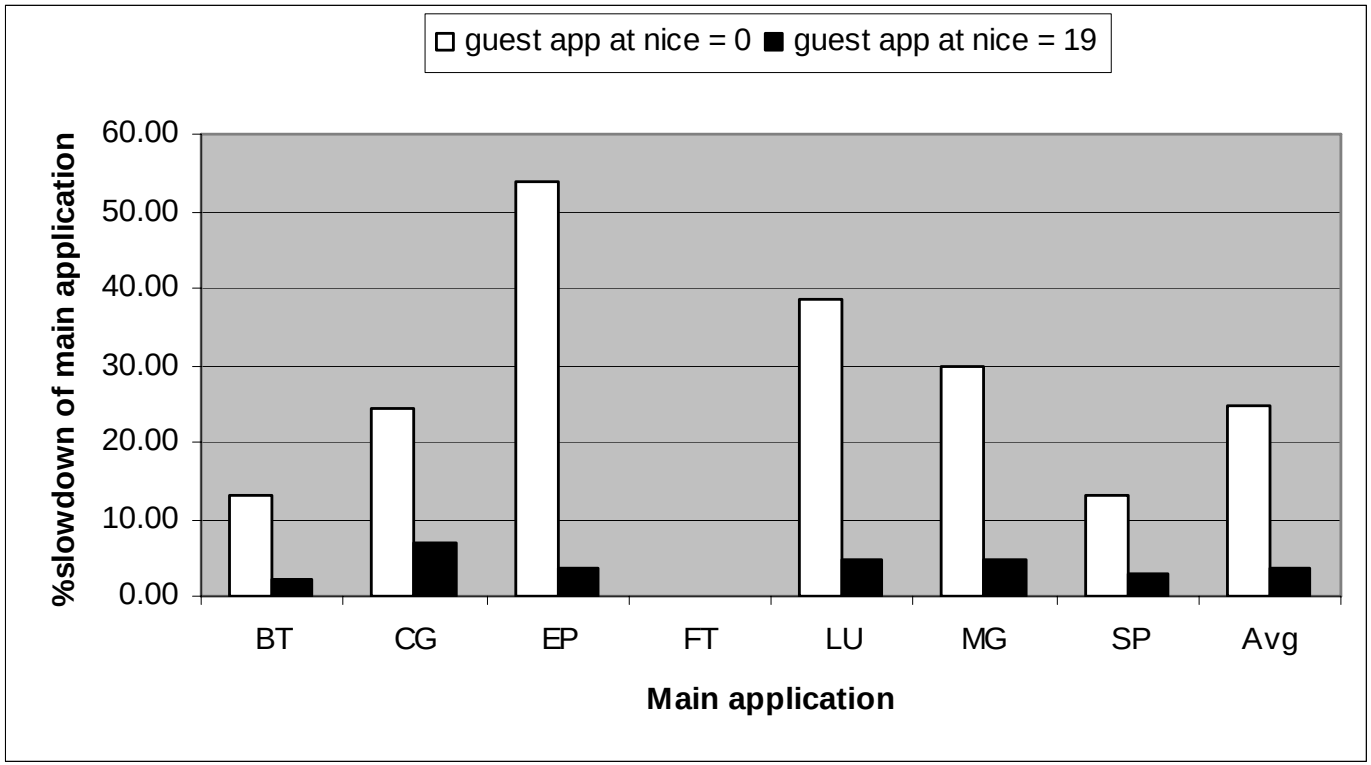
- **Volunteer computing on clusters is very attractive**
  - **Number of clusters is increasing and many are relatively idle**
- **This is one component of making true parallel volunteer computing possible**
  - **Most poor scientists will be able to use other people's clusters**
- **Significant hurdles remain, especially in making scheduling more friendly**

Contact: www.cs.uh.edu/~jaspal
jaspal@uh.edu

Thanks !!

Questions?

Impact of lowering the priority of Guest application

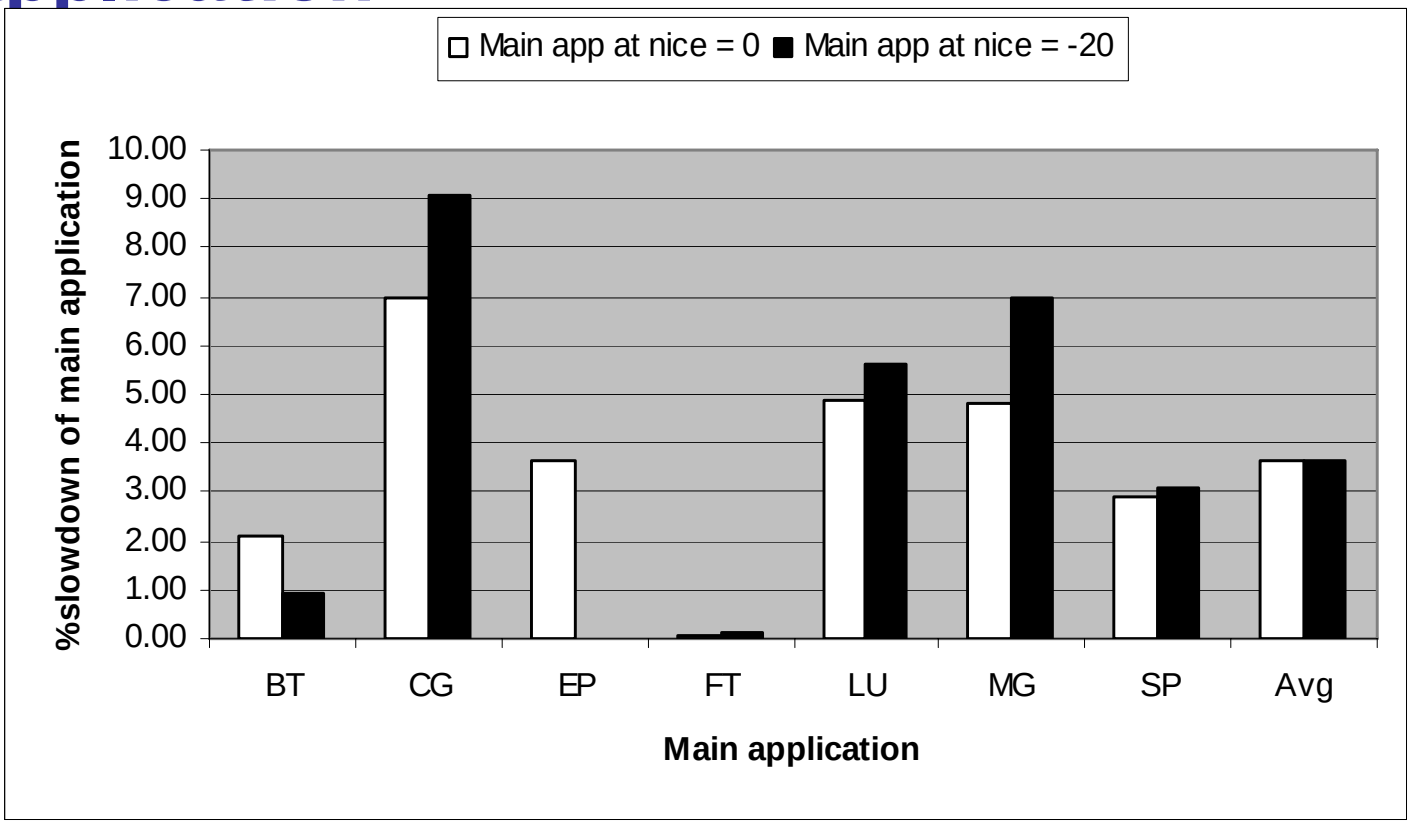


EP as guest app

Observations:

- **By running the guest application at lower priority, the slowdown of main app reduces considerably**
- **FT does not slowdown at all (guest app gets enough cpu from idle time)**

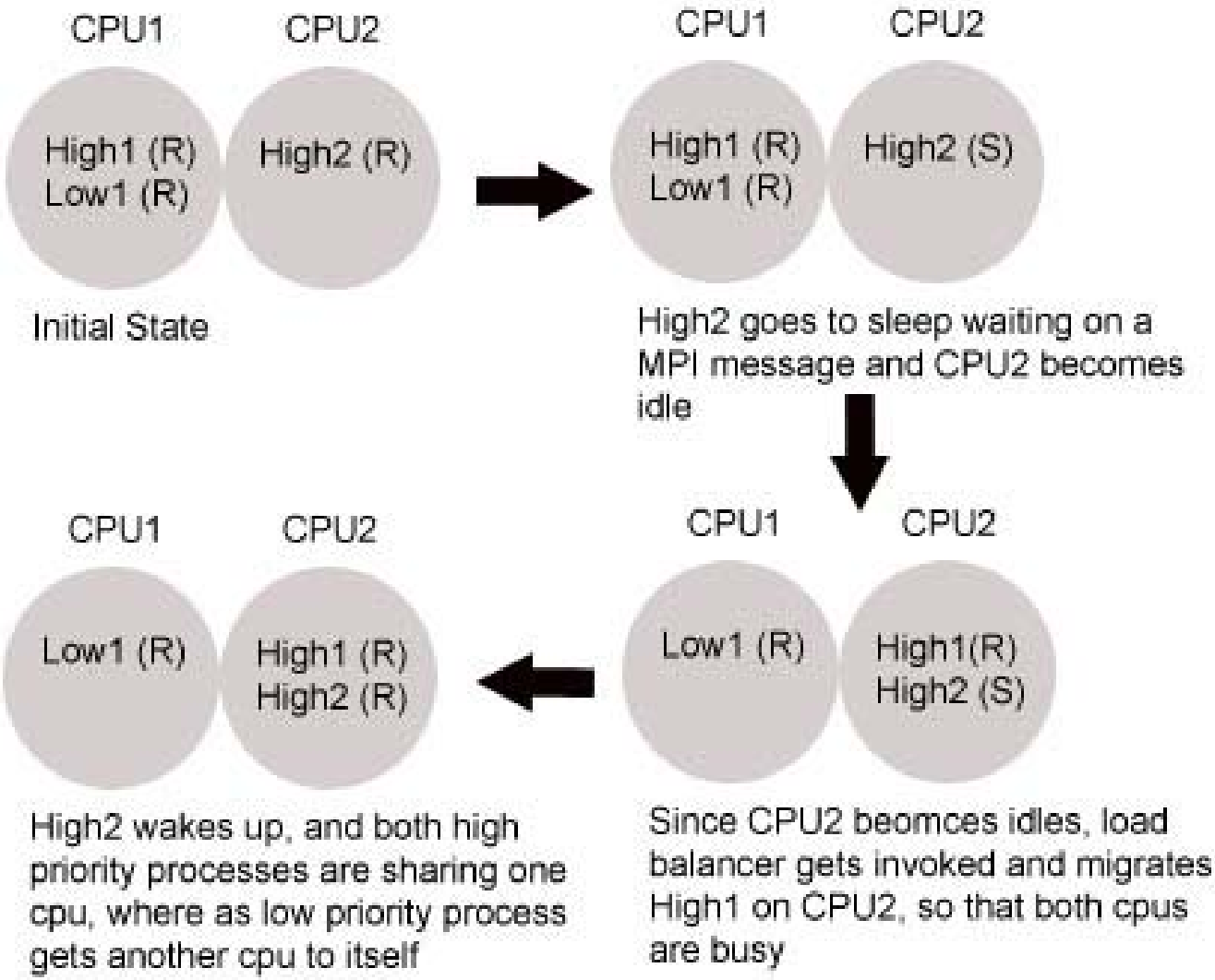
Impact of lowering the priority of Main application



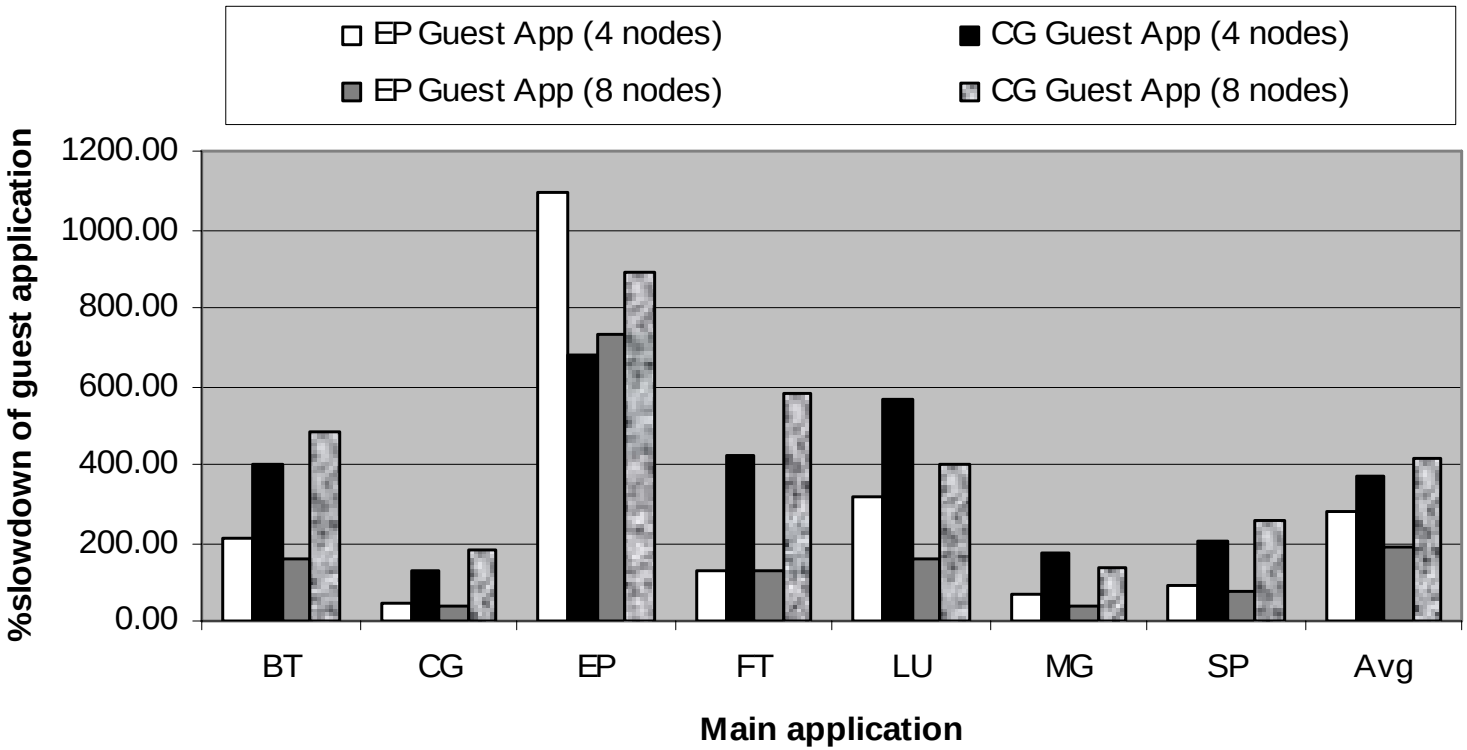
Running at nice = -20, needs root access

Observations:

- **Increasing priority of Main app to highest does not help**
- **Except EP (main app)**


CPU1
CPU2
High1 (R)
Low1 (R)
High2 (R)
Initial State
CPU1
CPU2
High1 (R)
Low1 (R)
High2 (S)
High2 goes to sleep waiting on a MPI message and CPU2 becomes idle
CPU1
CPU2
Low1 (R)
High1(R)
High2 (S)
Since CPU2 beomces idles, load balancer gets invoked and migrates High1 on CPU2, so that both cpus are busy
CPU1
CPU2
Low1 (R)
High1 (R)
High2 (R)
High2 wakes up, and both high priority processes are sharing one cpu, where as low priority process gets another cpu to itself

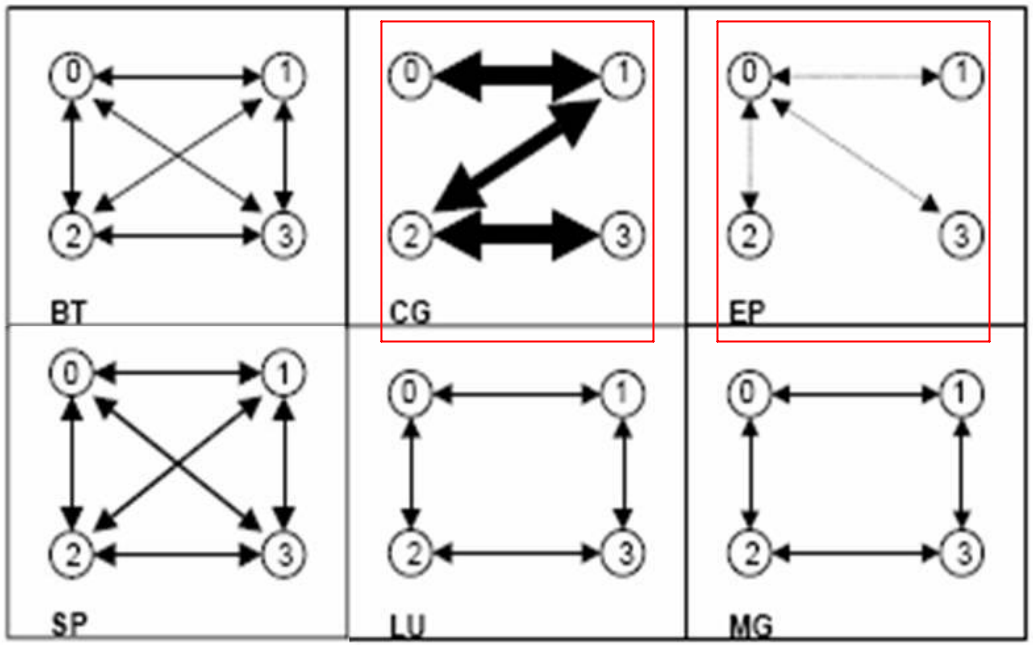
Slowdown of different types of guest applications



Observations:

- **CG guest application slows down more than EP guest application**
- **As number of nodes increase, slowdown of CG increases whereas slowdown of EP decreases**

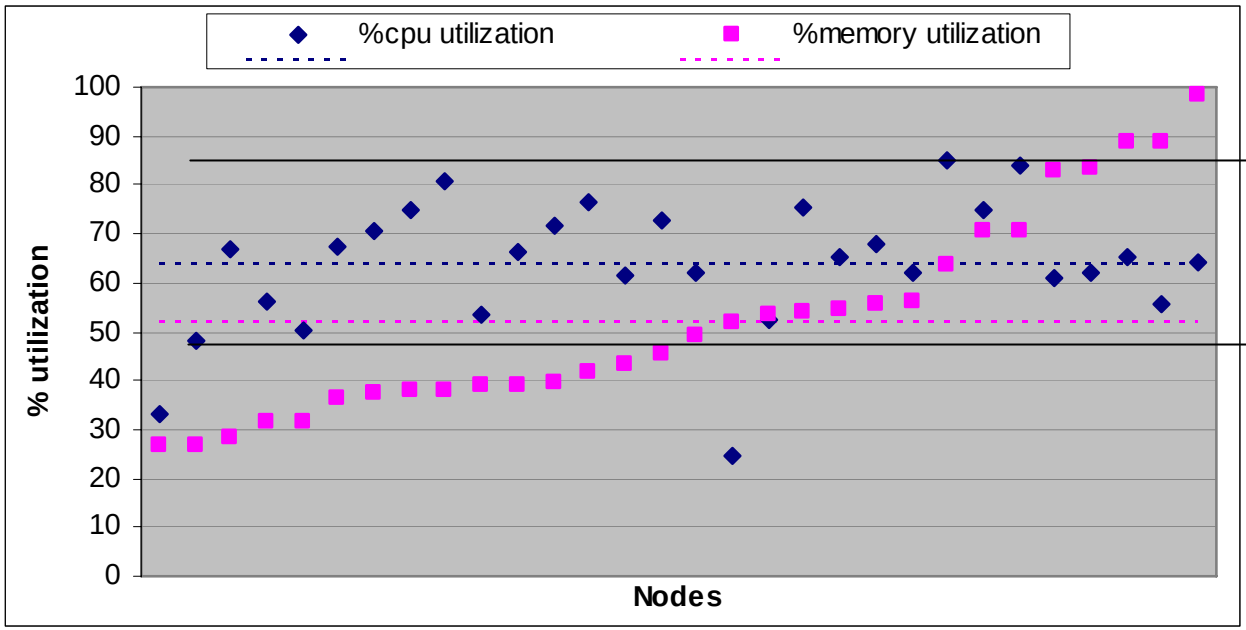
Types of Guest Applications



Communication Pattern of NAS Benchmark, where thickness of line shows bandwidth

Benchmark	Avg cpu utilization (%)
BT	90
CG	65
EP	100
FT	53
LU	94
MG	73
SP	81

Avg cpu and memory utilization of 30 nodes over 1 month period



Observations:

Avg cpu utilization = 63.80%

Avg memory utilization = 52.14%

There are idle cpu cycles available to steal (at a fine grain)

Results are presented in order of increasing memory utilization

Linux 2.6 kernel Scheduler and *nice* values

- **Scheduling = f (dynamic priority)**
- **Dynamic priority = static priority + interactivity bonus**
- **Static priority = nice value**
- **Timeslice = f (nice value)**

Nice value	Timeslice	Priority
-20	800ms	highest
0	100ms	Normal (default)
+19	5ms	lowest

- **Load balancer introduced as part of kernel (Run queue per cpu)**