



Modeling Internet-wide Systems

Students: Ravi Prithivathi Bhayankaram, Tsung-i "Mark" Huang, and Deepti Vyas

Advisor: Jaspal Subhlok (jaspal@uh.edu)



Fine Grain Global Computing

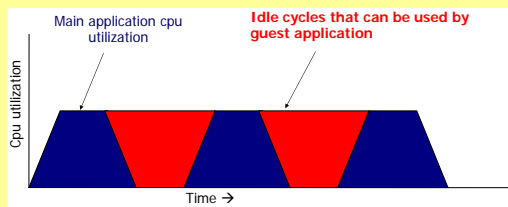
Contact Person: Ravi Prithivathi Bhayankaram (ravipb02@yahoo.com)

• Problem Statement

To find and utilize idle cycles for global computing applications without affecting the native application.

• Solution

Fine Grain Cycle Stealing is the process of exploiting the fine grained availability (of the order of milliseconds) of workstations to run sequential and parallel jobs.



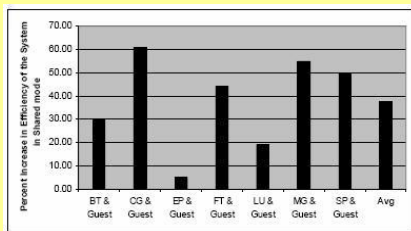
• Experimental Setup

- A low priority guest application is executed concurrently with high priority host application
- NAS Class B benchmarks were used as the host and guest applications
- Independent scheduling using Linux 2.4 and 2.6 OS scheduler

• Increase in Efficiency of the Cluster Due to Sharing

Observations:

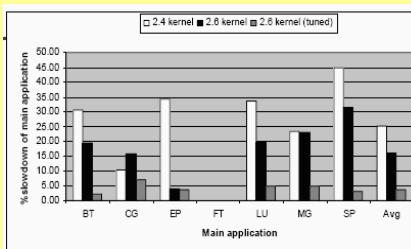
- Guest achieves reasonable performance with minimum impact on the host
- 38 percent increase in the system throughput



• Impact of Kernel Version and Tuning

Observations:

- 2.6 kernel gives the best results.
- Average slowdown drops from 25 percent in 2.4 kernel to 3.6 percent in 2.6 kernel



2.6 kernel (tuned) → kernel with load balancer frequency reduced from 200ms to 10ms

• Conclusion:

Compute Clusters provides a vast source of idle cycles and using fine grained cycle stealing we have achieved 38 % increase in system utilization with a slowdown less than 5 % for host applications.

Fast Pattern-Based Throughput Prediction for TCP Bulk Transfers

Contact Person: Tsung-i "Mark" Huang (tihuang@cs.uh.edu)

• Goal:

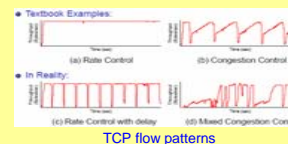
Quickly predict the throughput of a TCP stream by inspecting the flow pattern of a few segments arrived at the receiver.

• Applications:

- Replica / Mirror site selection
- Grid resource selection

• Approach:

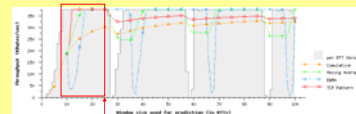
1. A stream of incoming TCP segments is partitioned and converted to a time series of throughput values.
2. This time series is then analyzed to identify known TCP flow patterns.



3. Based on the TCP flow patterns found, a prediction for future throughput is generated
4. A quality ranking is generated to indicate the likelihood that this predicted throughput is accurate

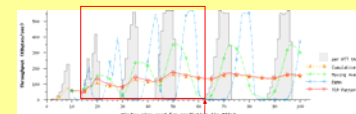
• Example:

- Rate Control based prediction:



Prediction made at the 25th RTT using average throughput over the flat region

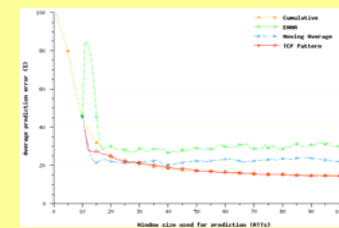
- Congestion Control based prediction



Prediction made at the 63rd RTT using average throughput of the last 3 complete Congestion Control cycles

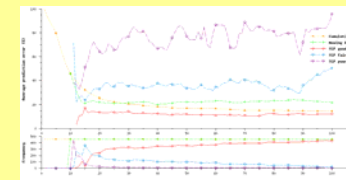
• Experiments and results:

- A total of 460 download traces were used for experiments
- We compare following prediction methods:
 - TCP Pattern-based throughput prediction
 - Moving Average
 - Exponential Weighted Moving Average
 - Past Cumulative throughput
- TCP Pattern-based prediction performs as well or better than other prediction methods.



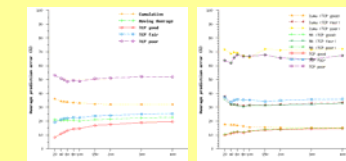
Performance of predict methods for next 200 RTTs

- Good quality predictions have lowest prediction error.



Performance of different quality rankings for predictions made for next 200 RTTs

- 77% of traces have their first prediction with good quality ranking before 25th RTT.
- A "good" prediction can be made in ~ 5 secs.



(a) Prediction made at the 15th RTT (b) Prediction made at the 25th RTT Performance of different quality rankings