

Obtaining Performance Measures through Microbenchmarking in a P2P Overlay Computer

P. Bertasi, M. Bianco, A. Pietracaprina, G. Pucci

University of Padova

April 11, 2007

DEPARTMENT OF
INFORMATION
ENGINEERING

UNIVERSITY OF PADOVA



DEI - UNIVERSITÀ DI PADOVA



ADVANCED COMPUTING GROUP

- 1 Introduction
- 2 Techniques for Performance Measurement and Improvement
 - Overview
 - Topology awareness techniques
- 3 Design and implementation of initial tests
- 4 Conclusions and Future Work

The goal:

Exploit unused bandwidth and computational power of computers connected to the Internet.

The goal:

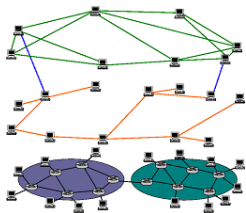
Exploit unused bandwidth and computational power of computers connected to the Internet.

The solution:

Develop an Overlay Computer (OC).

Definition of OC

A parallel computing platform whose network topology is embedded within the Internet.



To be successful an OC must provide higher performance and/or capabilities than the individual computing equipment available to the user.

It is necessary to provide the user with tools for estimating the effectiveness of design choices on such an OC.

Microbenchmarks

Suite of simple codes carefully designed to evaluate individual features of the system.



Useful to provide a quantitative assessment of system capabilities:

- improve the OC topology
- determine application side optimizations

- Overview of performance measurement techniques for related parallel distributed platforms
- Set of experimental results to serve as *proof-of-concept* for the development of a microbenchmarking tool



Measures of interest:

- latency (often referred to as *distance*)
- bandwidth

Latency depends on:

mainly topology

methodology:

- overlay *ping time*
- fictitious coordinate systems
 - using landmarks
 - totally distributed (VIVALDI)

Bandwidth depends on:

- quality of the path
- amount of competing traffic
- load at end-points

methodology:

- $O(n^2)$ *direct measures*
- EXPLOITING INTERNET ROUTE SHARING...
 - linear overhead
 - limited cooperation
 - **complex structure difficult to satisfy in an OC**



Topology awareness techniques for performance improvement on P2P Systems

Problem:

PEERMETRIC project suggests that

- last hop bandwidth is a major bottleneck
- latency-based optimization is somewhat a less impacting issue

Possible solution:

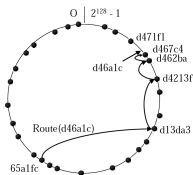
Use of topology awareness:

- to improve performance
- to save bandwidth
- to manage churn

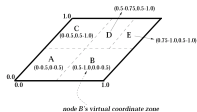


To improve performance:

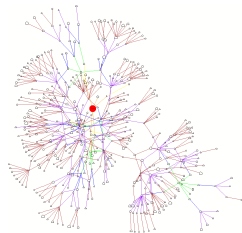
- Pastry** uses the number of Internet hops to optimize the peer “position”.
- C.A.N.** uses ping against unconscious landmarks (web server, DNS server...) to optimize the peer “position”.
- G.I.A.** uses statistics on peer capacity to determine network topology.



Pastry lookup graph.



Construction of CAN.



Gnutella network graph.

To save bandwidth:

Kazaa caching may provide 60% saving on external bandwidth

Gnutella a hierarchical topology yields a substantial bandwidth savings



To deal with churn:

Bamboo uses topology awareness also decreasing latency

Special case: P2P computing

ZORILLA uses the same algorithms as G.I.A. to allocate jobs on peers.

Result:

reached nodes are close to the submitter node

- speeds-up the initial phase
- closeness may be a limitation

JXTA: P2P API built over JAVA, becoming *de facto* standard for P2P platform development.

JXTASocket:

- reliable
- bi-directional



- Network:
100 Mbit/s
switched Ethernet
- Machines:
 - slow (< 1 GHz)
 - fast (> 1.5 GHz)

Cat.	CPU	Frequency	O.S.
slow	Pentium III	800 MHz	GNU/Linux 2.6.16
slow	Pentium III	600 MHz	GNU/Linux 2.6.12
slow	Pentium III	866 MHz	GNU/Linux 2.6.16
slow	Pentium III	866 MHz	GNU/Linux 2.4.18
fast	Dual Opteron	2.2 GHz	GNU/Linux 2.6.5 smp
fast	Core Duo	1.66 GHz	Windows Xp sp2
fast	Pentium 4 HT	3 GHz	GNU/Linux 2.6.13 smp

Latency

Ping test of 8 bytes between two nodes.

Bandwidth

- is measured with respect to specific communication patterns

CPU power

- fixing iterations
- fixing time

The tests are repeated several time to filter out noise.

Sender	Receiver	JXTA Time (ms)	ICMP (ms)
Fast	Fast	17.2	0.24
Fast	Slow	70.4	0.16
Slow	Fast	57.6	0.14
Slow	Slow	85.5	0.22

Considerations

- ICMP is 3 orders of magnitude faster than JXTA ping (due to software overhead)
- slow to fast is better than fast to slow

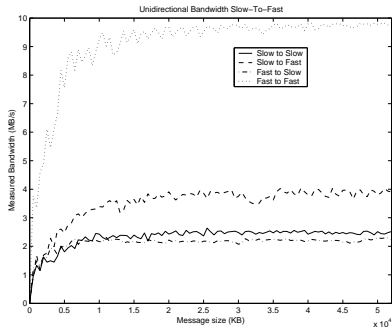
Communication patterns

- point-to-point
- scatter
- gather
- all-to-all



- Due to small buffering capacity of the IP stack the sending of large amount of data could be considered *blocking*
- For gather and all-to-all measurement a simple clock alignment is employed to stress the network

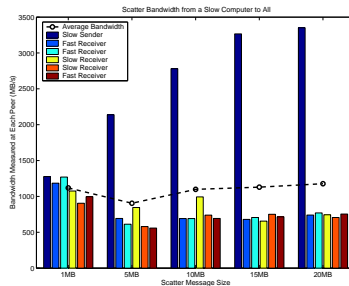
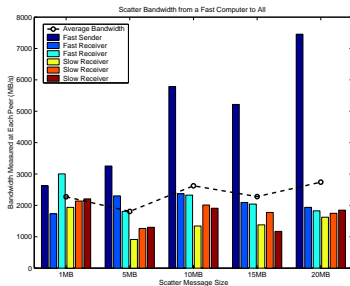
Bandwidth measurement: point-to-point



Results

- slow to fast is better than fast to slow
- high dependency on computational power

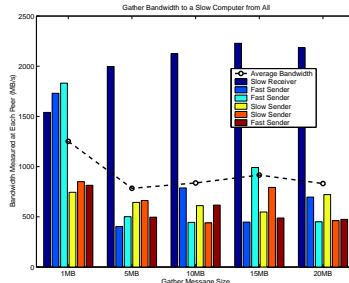
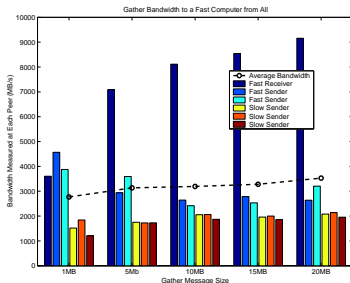
Bandwidth measurement: scatter



Results

- 1MB is too short to create congestion
- in our testbed 10 MB are needed to obtain significant measures

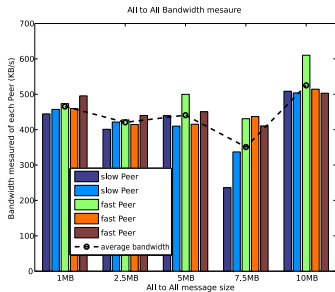
Bandwidth measurement: gather



Results

- incoming bandwidth of the *slow* computer gathering data is smaller than the outgoing bandwidth of the *slow* computer scattering data
- the incoming bandwidth of the *fast* computer gathering data is larger than the outgoing bandwidth of the *fast* computer scattering data

Bandwidth measurement: all-to-all



Results

- individual bandwidths are flattened near the one of the slowest node



Description

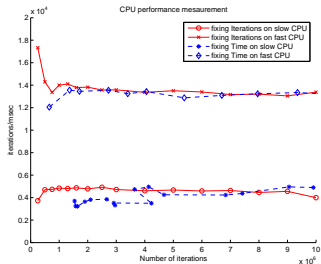
- quiz like approach
- requirements
 - small input/output
 - given amount of computation that cannot be avoided

Simple integer random generator:

$$seed = MOD(8121 * seed + 28411, 134456)$$

Implementation

- fixed # iteration
- fixed time



Conclusions

- Different patterns exhibit highly different behaviors (must be hence careful in choosing executing peers)
- Countermeasures against selfishness and free-riders (especially for computer power measures)

Future work

- Add more patterns
- Performance conscious spawning primitive
- Experiment on larger heterogeneous testbeds (e.g. PlanetLab)

