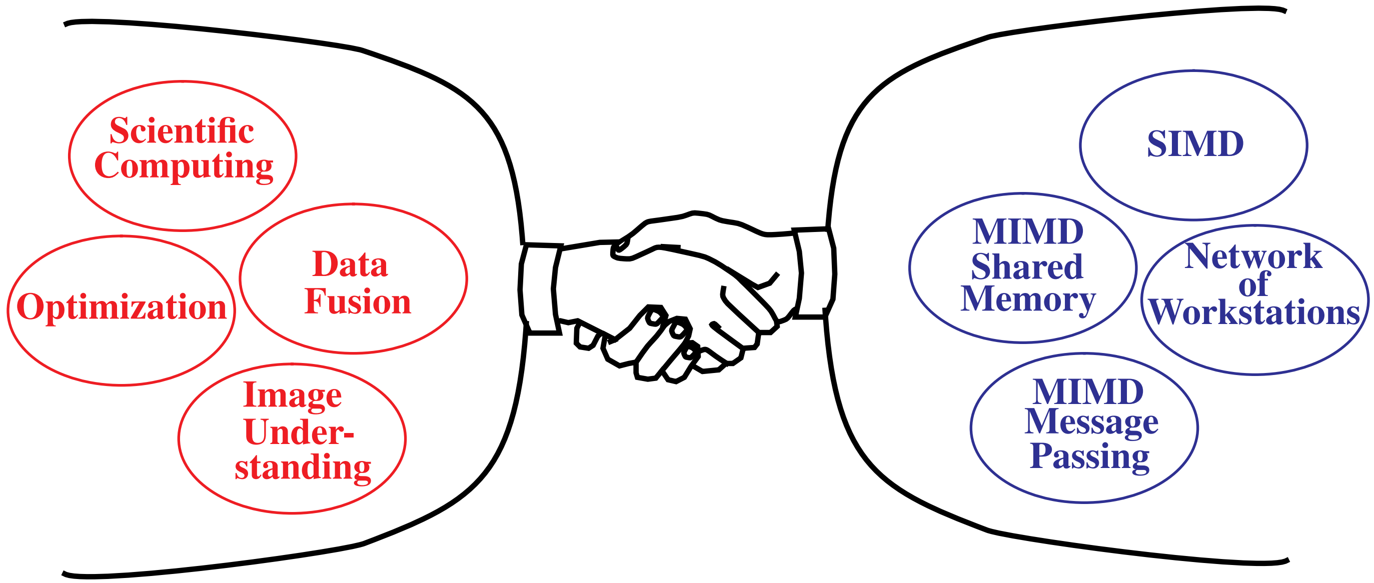
A Framework for Evaluating Architectural Issues of Parallel Systems

Anand Sivasubramaniam

College of Computing
Georgia Institute of Technology
Atlanta, GA



Satisfying application demands => Parallel Computing

Application-driven study

- **How do we predict performance of parallel systems?**
- **How do we identify application/architectural bottlenecks?**
- **How do we use this information to enhance performance?**
- **What should be the values for various architectural parameters?**
- **What mechanisms should we provide in the underlying architecture?**

Contributions

• Novel framework for answering the questions.

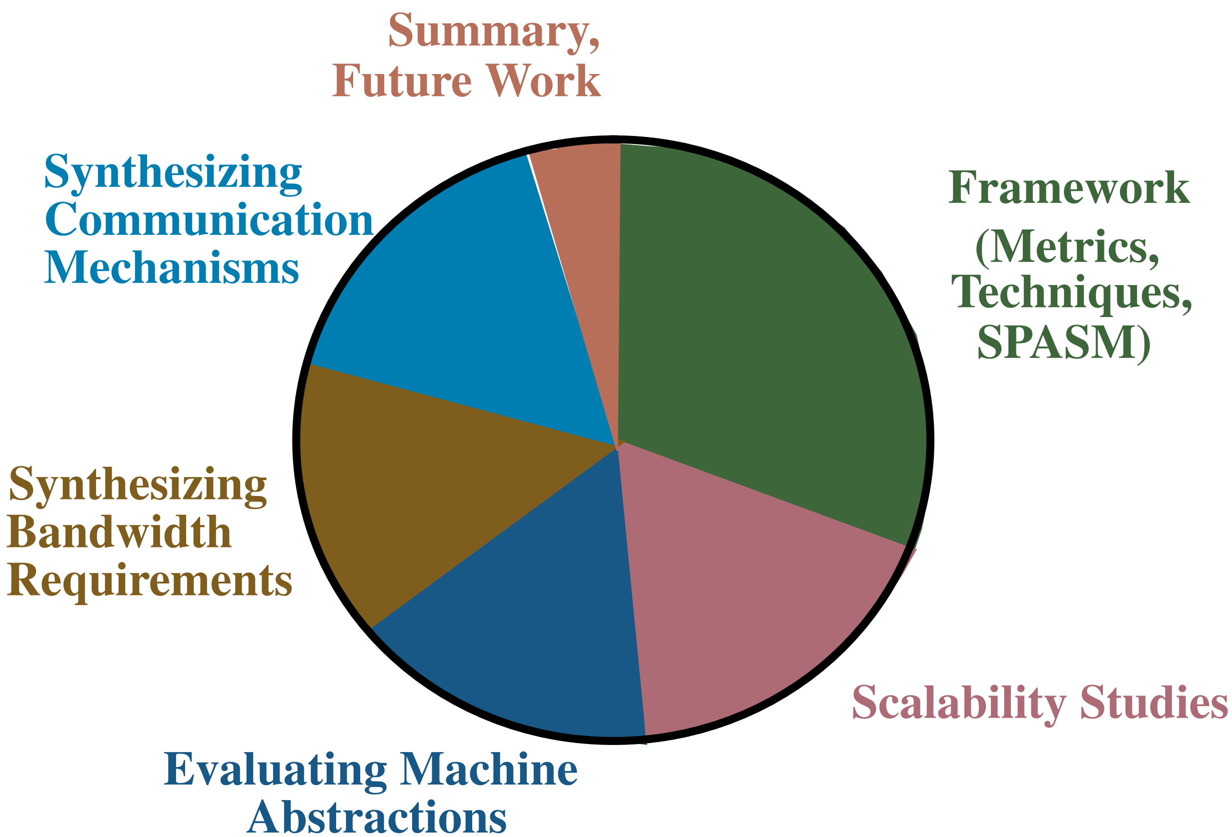
• Set of tools for implementing the framework.

• An understanding of the scalability of some parallel systems.

• Network bandwidth requirements of parallel applications.

• Explicit communication primitives.

Outline



Summary,
Future Work
Synthesizing
Communication
Mechanisms
Framework
(Metrics,
Techniques,
SPASM)
Synthesizing
Bandwidth
Requirements
Scalability Studies
Evaluating Machine
Abstractions

Performance Metrics

- **Speedup, Scaled speedup [Gustafson 88], Sizeup [Sun & Gustafson 91].**

- **Several other metrics [Kumar 87, Karp & Flatt 90, Nussbaum & Agarwal 91]**

- **Comparative evaluation in [GIT-CC-94/38].**

☞ ***Need to identify, isolate and quantify the inadequacies (overheads) in a parallel system.***

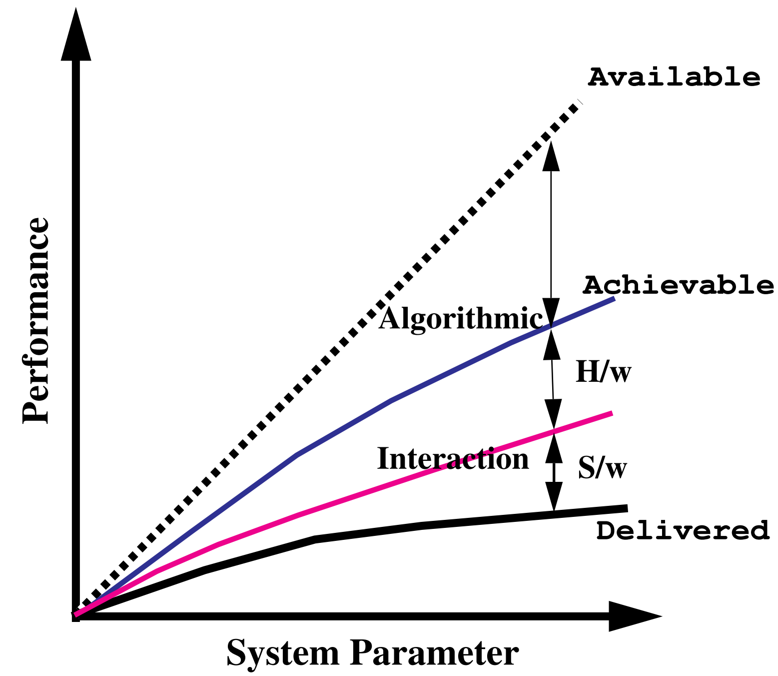
Overhead Functions [Sigmetrics 94, JPDC 94]
Lost Cycles [Crovella & LeBlanc 94]

Parallel System Overheads

Algorithmic Overhead
- Serial Part
- Work Imbalance
- Additional Work

Hardware Interaction Overhead
- Network Latency
- Contention
- Synchronization

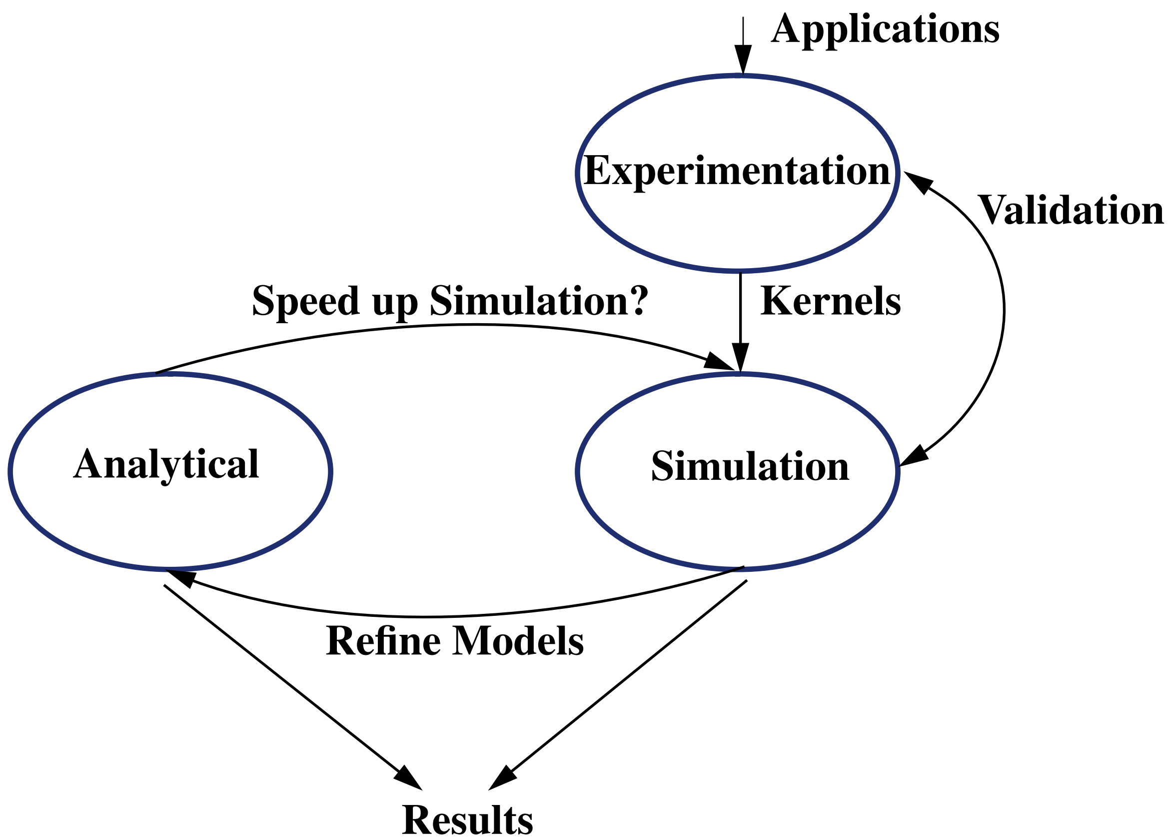
Software Interaction Overhead
- Scheduling
- Software Messaging/Sync
- Memory Management



Performance Evaluation Techniques

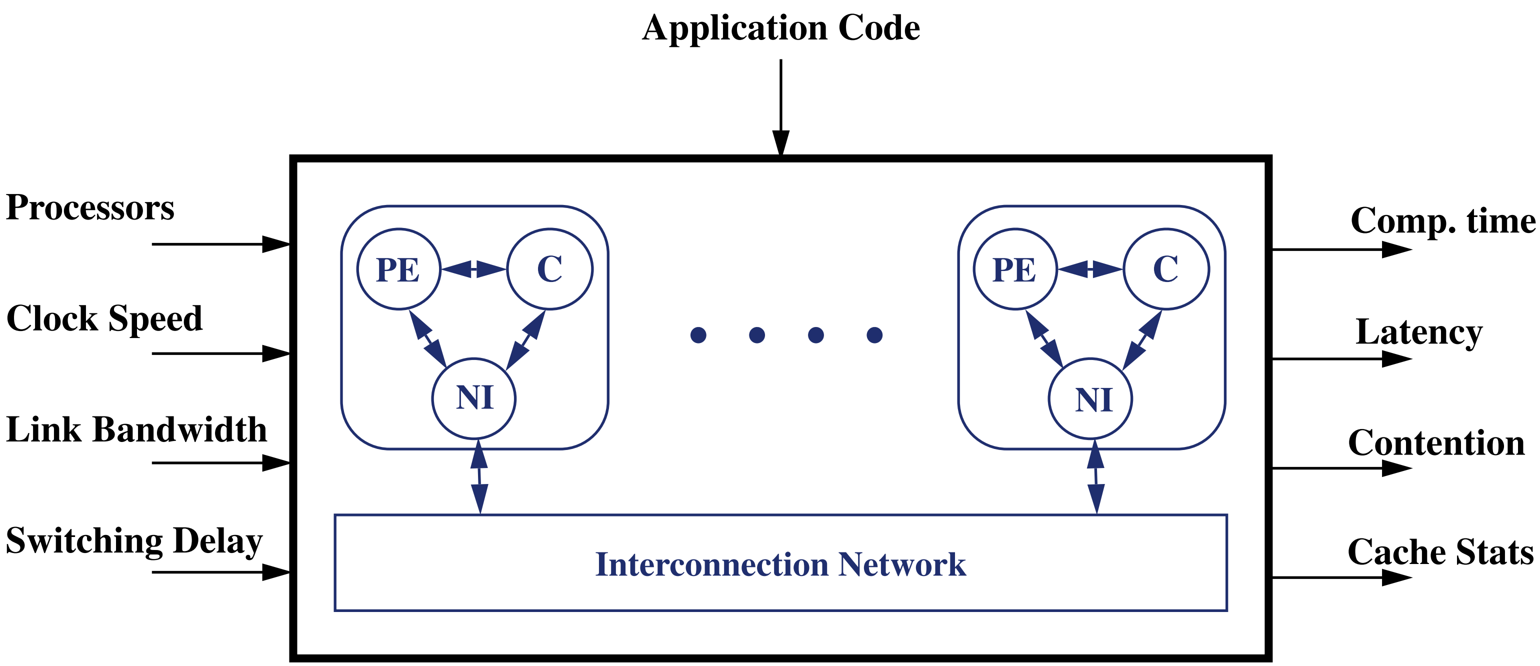
	Statistics	Accuracy	Space/Time
Experimentation	-	+	+
Analytical Methods	+	-	+
Simulation	+	+	-

The Framework



SPASM

(Simulator for Parallel Architectural Scalability Measurements)



Anand Sivasubramaniam

Simulators for Parallel Architectures

- Rice Parallel Processing Testbed [Covington et al. 88]
- Proteus [Brewer et al. 91]
- Tango [Davis et al. 91]
- MINT [Veenstra et al. 94]
- PAPS [Wabnig and Haring 94]
- Wisconsin Wind Tunnel [Reinhardt et al. 93]

Statistics

Modes of shared memory program execution

- ***BARRIER*** **- Barrier synchronization operation**
- ***MUTEX*** **- Mutual exclusion lock**
- ***PGM_SYNC*** **- Signal-Wait for pair-wise synchronization**
- ***NORMAL*** **- Useful work in the application**

Statistics (contd.)

Computation Time **- Execution time on an ideal machine**

Latency **- Time spent in actual message transmission in absence of contention (network latency vs. observed latency)**

Contention **- Time spent waiting for availability of network resources (network contention vs. observed contention)**

$$\text{Mode-wise Execution Time} = \text{Comp. Time} + \text{Latency} + \text{Contention}$$

$$\text{Total Time} = \sum_{\text{modes}} \text{Mode-wise Execution Time}$$

Statistics (contd.)

	Comp. Time	Latency	Contention
NORMAL	Useful Work	Data Access	Data Access
BARRIER	Work Imbalance	Sync. Access	Sync. Access
PGM_SYNC	Work Imbalance	Sync. Access	Sync. Access
MUTEX	Serial Part	Sync. Access	Sync. Access

Validation

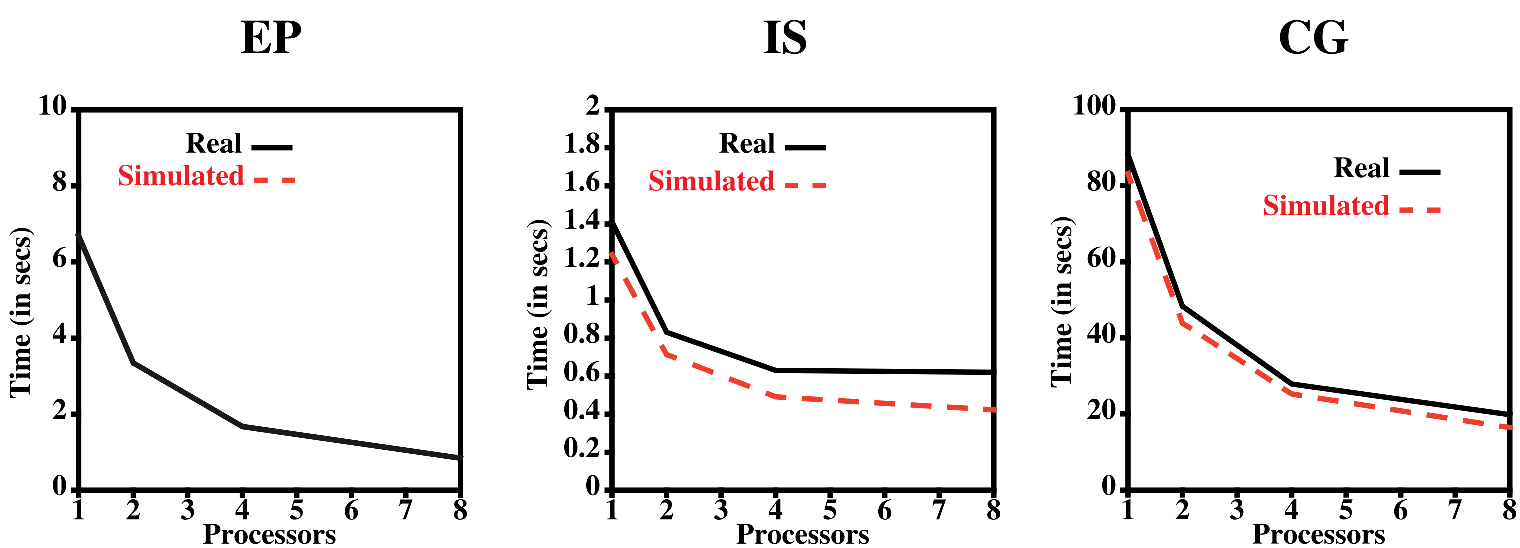
Validate by comparing with execution on actual hardware.

Difficulty in validating individual overheads.

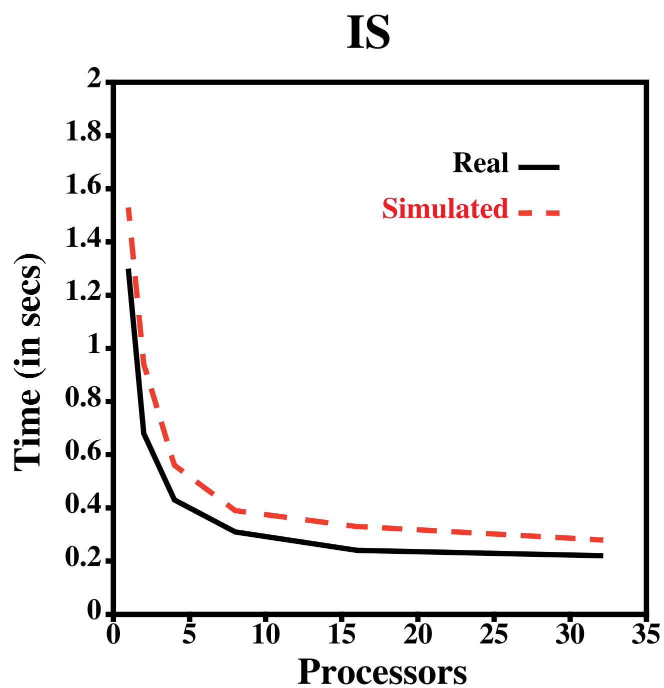
Within 10% accuracy for the iPSC/860.

Around 10-15% accurate for the KSR-1.

Validation on iPSC/860



Validation on KSR-1

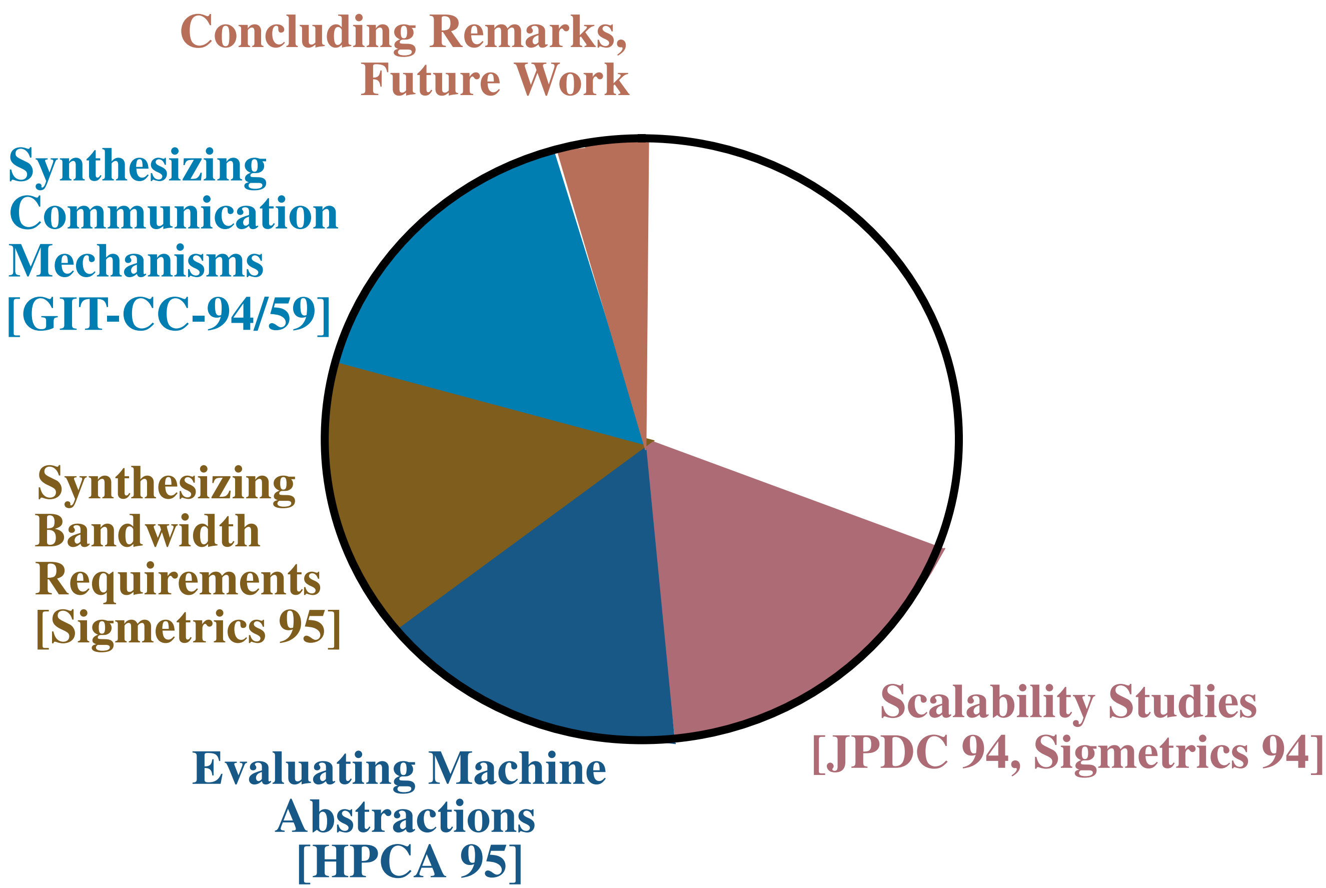


Key Results

Overhead functions.

An approach for quantifying overhead functions.

A set of tools for implementing the framework.


Concluding Remarks, Future Work
Synthesizing Communication Mechanisms [GIT-CC-94/59]
Synthesizing Bandwidth Requirements [Sigmetrics 95]
Evaluating Machine Abstractions [HPCA 95]
Scalability Studies [JPDC 94, Sigmetrics 94]

Scalability Studies
[Sigmetrics 94, JPDC 94]

How do we predict the performance of parallel systems?

How do we identify application/architectural bottlenecks?

How do we use this information for enhancing performance?

Applications: EP, IS, CG (NAS); CHOLESKY, FFT (SPLASH)

Architectures:

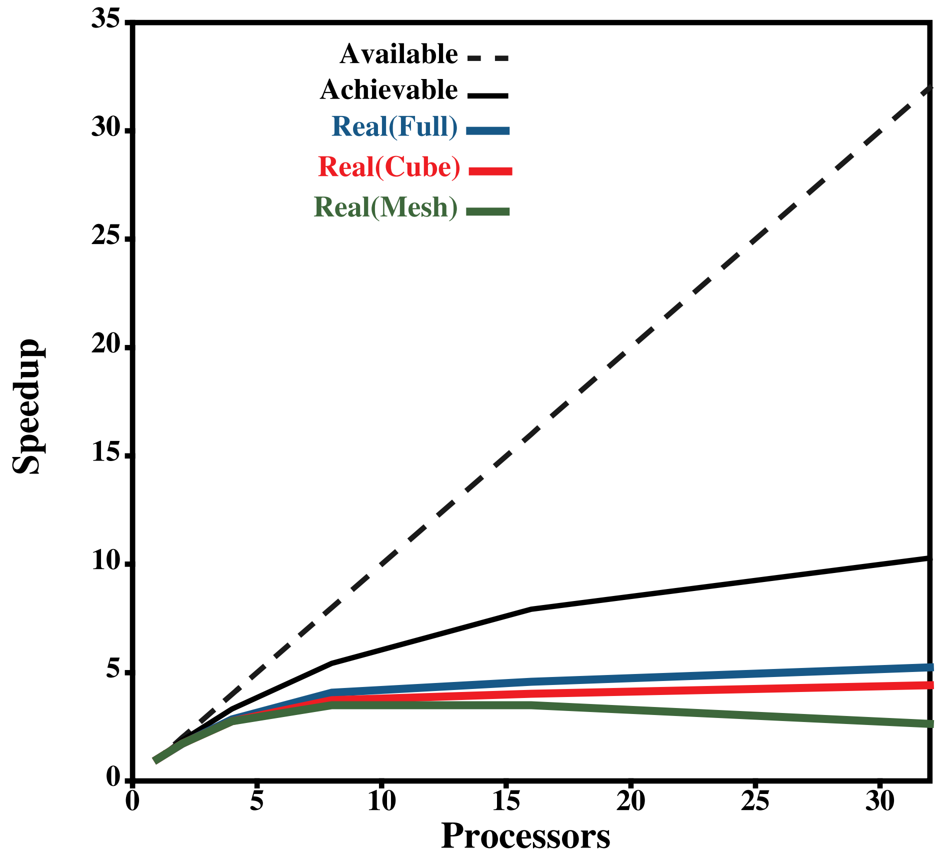
Cache-coherent shared memory machines with three different network topologies - fully connected, hypercube, 2-D mesh.

Scalability Study of IS (Integer Sort)

Data Size = 64K, Buckets = 2K

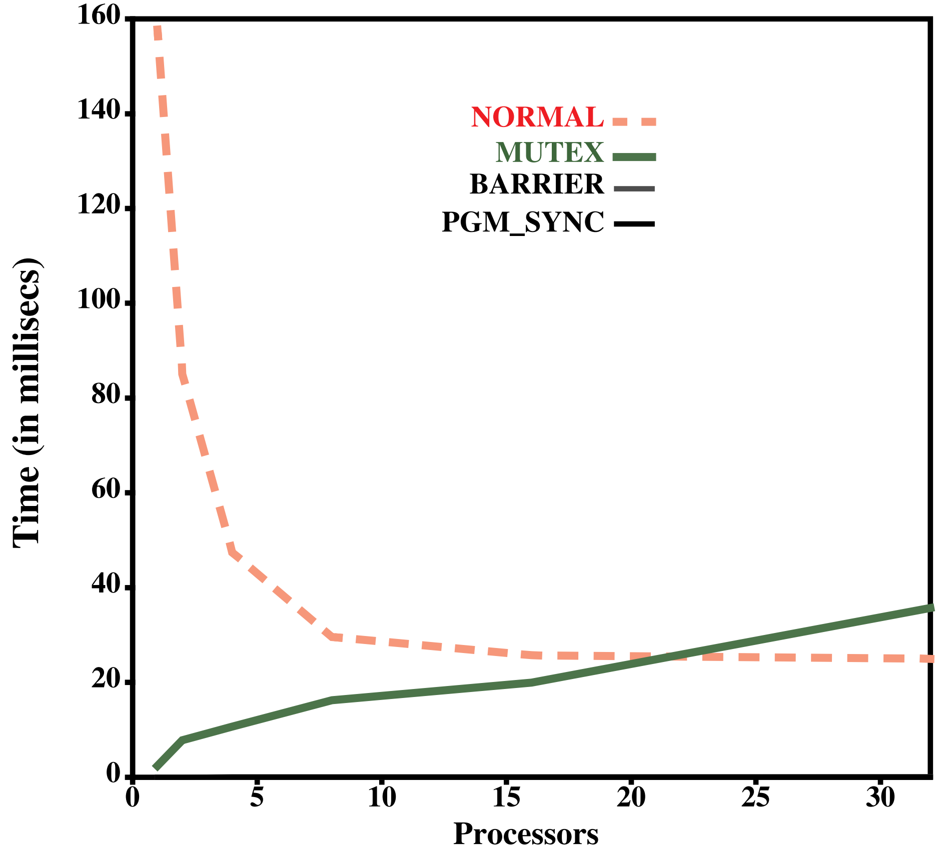
Phase	Description	Comp. Gran.	Data Gran.	Sync.
1	Local bucket updates	Small	--	--
2	Barrier Sync.	--	--	Barrier
3	Global bucket merge	Small	chunk *X* (p-1) integers	--
4	Global sum	Integer Add	Integer	Signal-Wait
5	Global bucket updates	Small	--	--
6	Barrier Sync.	--	--	Barrier
7	Global bucket updates	Small	Bucket Size	Mutex
8	Local list ranking	Small	--	--

Speedup

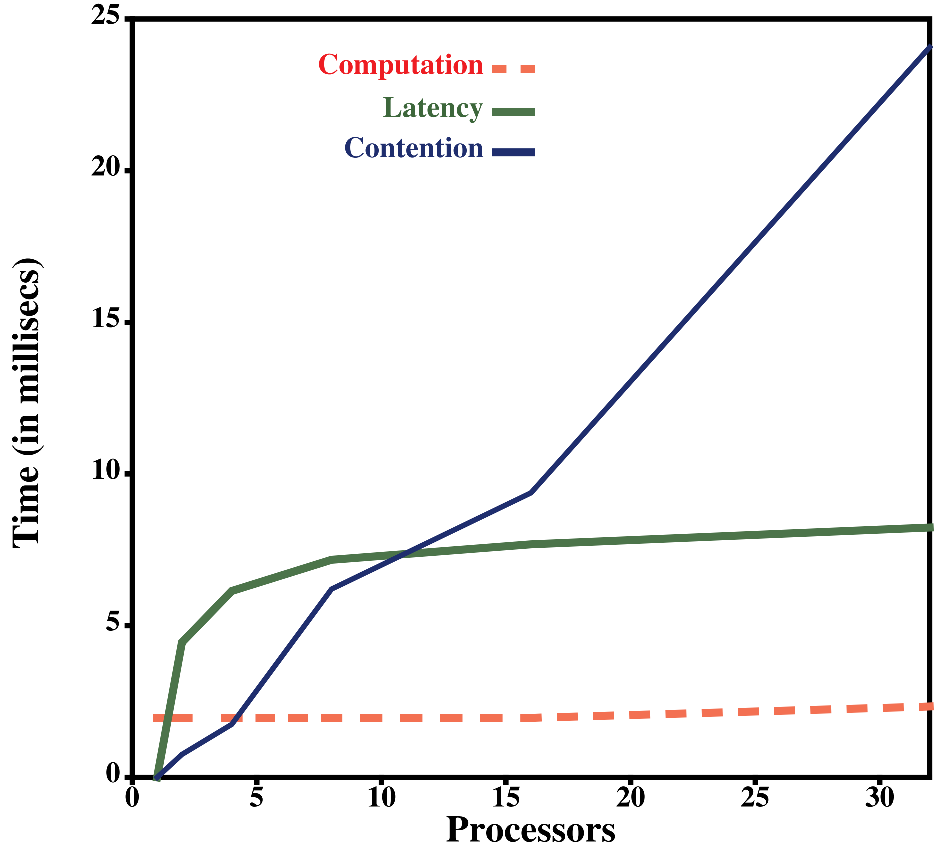


Anand Sivasubramaniam

Execution Time (Mesh)



MUTEX Mode (Mesh)



Anand Sivasubramaniam

Summary of Evaluation for IS

IS	FULL	CUBE	MESH
Comp. Time (ms)	$\frac{129.3}{p^{0.7}}$	$\frac{129.3}{p^{0.7}}$	$\frac{129.3}{p^{0.7}}$
Latency (ms)	13.2(1 - 1/p)	13.2(1 - 1/p)	13.2(1 - 1/p)
Contention (ms)	*Negligible*	4.0 *log* p	0.9p

- Algorithmic Overhead (additional work) limits scalability.
- Interaction overheads in MUTEX mode dominate for larger *p*.
- Latency converges to a constant.
- Contention becomes dominant for lower connectivity networks.

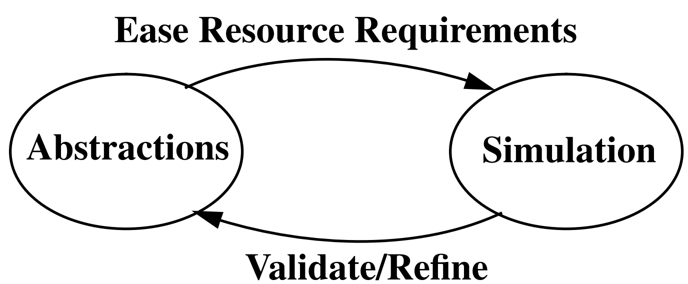
Key Results

- Performance prediction.
- Feedback to programmer for application restructuring.
- Feedback to architect for architectural enhancements.

Evaluating Machine Abstractions [HPCA 95]

Execution-driven simulation is costly in terms of space and time.

Abstractions such as theoretical/analytical models have been used to simplify parallel system details.



- ***Can we use these abstractions to speed up simulation?***
- ***How accurate are these abstractions?***

Case Study: Interconnection Network, Locality

Abstracting the Network

Can we abstract away the network completely? - NO!

LogP [Culler et al. 93]

Latency L: Each message takes time L to go from any source to any destination regardless of message size.

Gap g: Minimum time between successive message transactions at a node (calculated using bisection bandwidth available per processor).

Abstracting Locality

Application characteristic modeled by hardware cache.

Can we abstract away locality completely?

Ideal cache at each node.

Maintained consistent using an invalidation-based protocol.

Zero overheads for coherence maintenance.

Optimistically model true inherent communication in applications without details of locality maintenance.

Only if results show significant deviation do we need to model more complicated details.

Evaluation Approach

Three simulated machine models

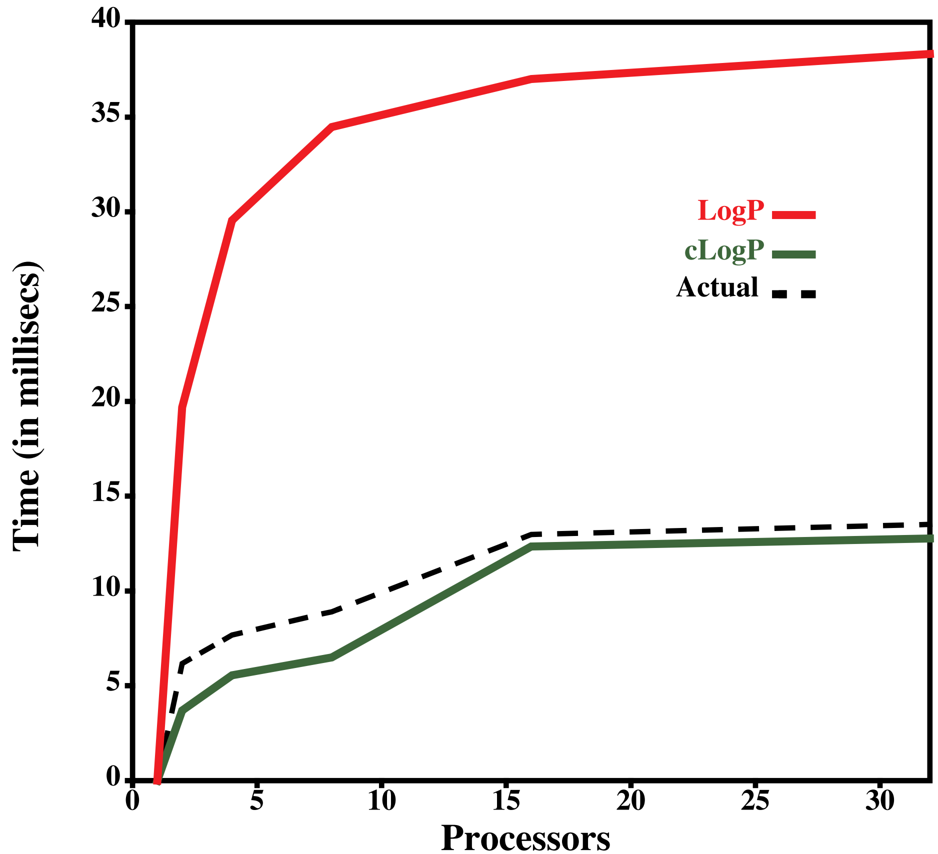
- **LogP machine (no cache + LogP for network)**
- **cLogP machine (LogP machine + ideal cache)**
- **Actual Machine (all hardware details)**

How accurate is LogP for the network?
- Compare latency/contention overheads for cLogP & Actual.

How pessimistic is it when we ignore locality?
- Compare LogP and Actual.

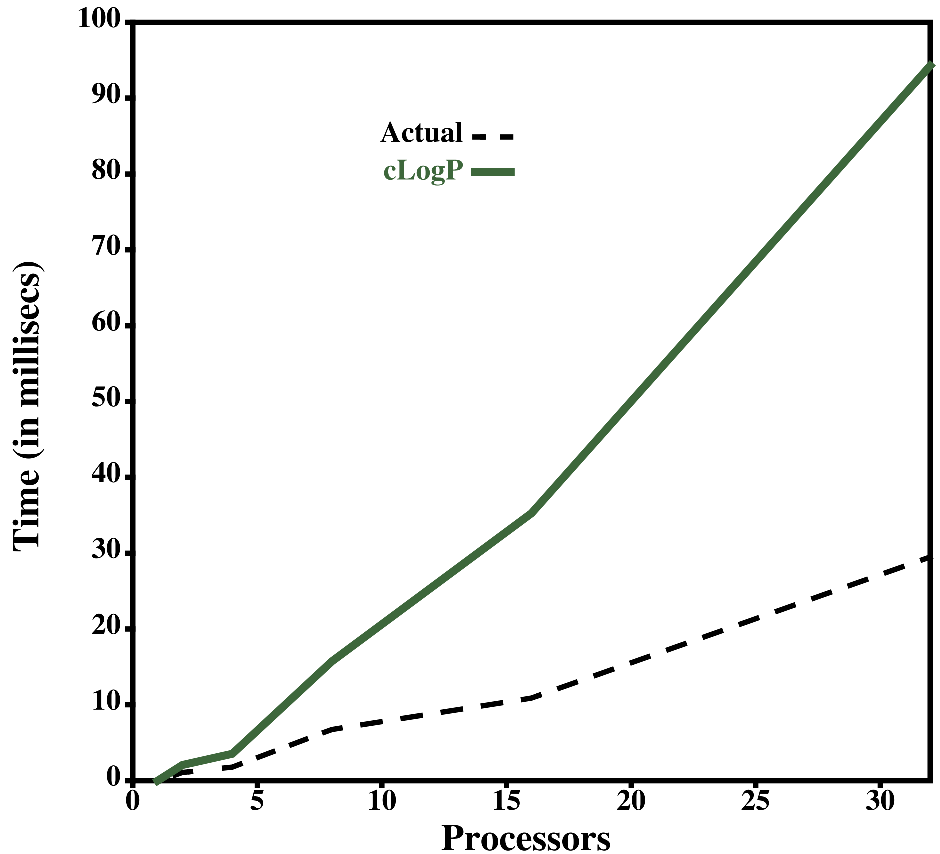
How good is our locality abstraction?
- Compare # of messages between cLogP and Actual.

IS on Mesh: Latency

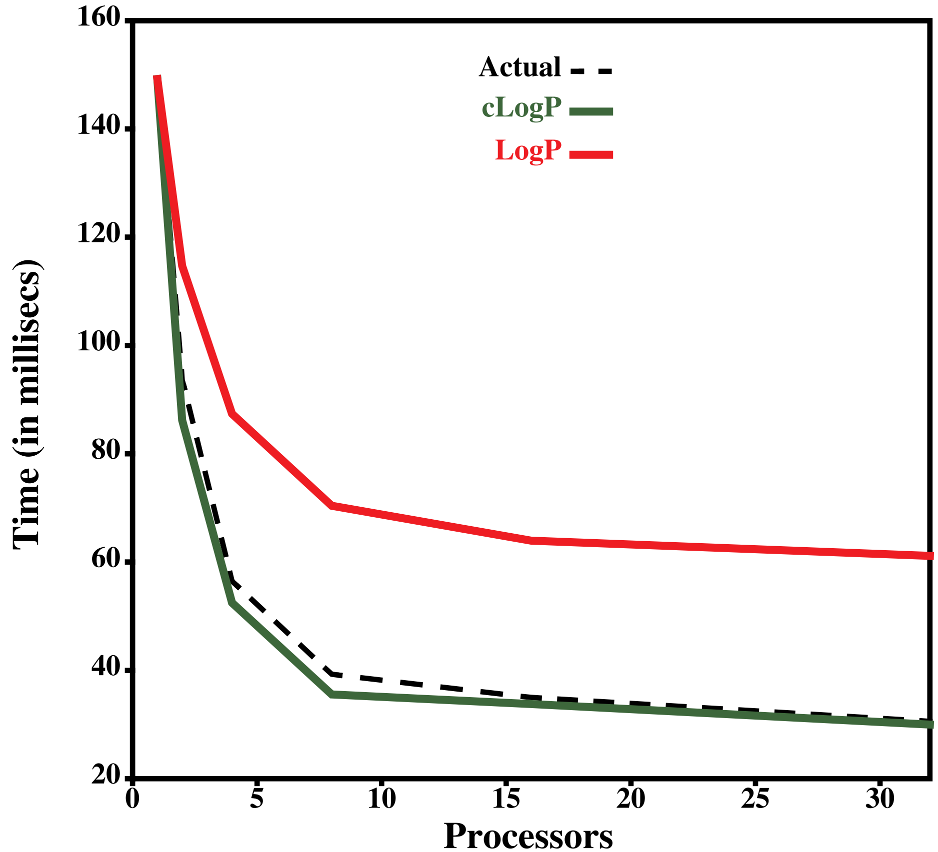


Anand Sivasubramaniam

IS on Mesh: Contention



IS on Full


160
140
120
100
80
60
40
20
0
5
10
15
20
25
30
Time (in millisecs)
Processors
Actual
cLogP
LogP

Key Results

Abstracting the network

- ***L*** **is fairly accurate.**
- ***g*** **is pessimistic.**

Abstracting locality

- **an ideal coherent cache is a good performance estimator for the chosen applications.**

Simulation of cLogP is around 300% faster.

Synthesizing Bandwidth Requirements [Sigmetrics 95]

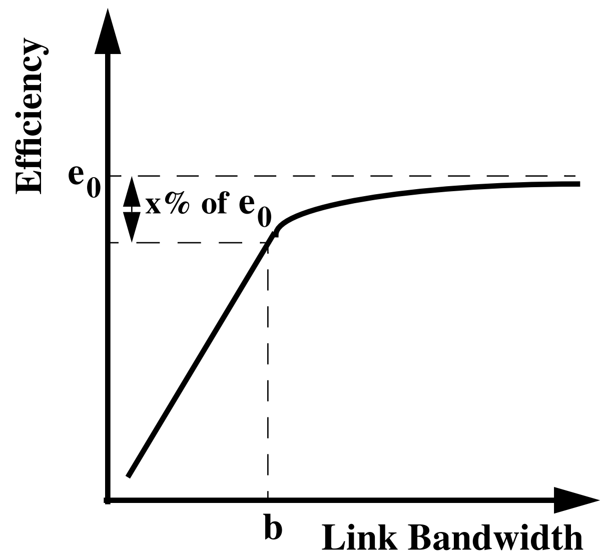
What link bandwidth should I provide in the network to assure good performance over a range of applications?

To answer this question:

- **Topology of the network**
- **Communication attributes of application: volume, pattern, frequency, tolerance**
- **Impact of system parameters: processors, CPU clock speed, problem size**

Evaluation Approach

Vary link bandwidth and study impact on application.



Quantify link bandwidth needed to limit network overheads to x% of overall execution time.

Study impact of system parameters (processors, CPU clock speed, problem size)

Key Results

Requirements for some applications (EP,FFT) can be satisfied with existing link bandwidths of 200-300 MBytes/s for 300 MHz CPU clock and 1024 node systems.

Network overheads for some other applications (IS, CG, CHOLESKY) can be limited to 30% of execution time provided problem sizes are increased commensurate with the processor clock speed.

Analysis quantifies the rate at which the problem size should be scaled up as a function of the number of processors and CPU clock speed.

Synthesizing Communication Mechanisms [GIT-CC-94/59]

Locality is an application characteristic.

Cache is the hardware artifact that captures the locality behavior in shared memory systems.

Applications have different communication characteristics.

No one universal solution.

☞ ***Studying communication behavior of applications can suggest useful architectural primitives.***

Communication in Applications

Sender-initiated

- Static
  - Data access pattern determinable at compile time.
  - eg. FFT, SOR
- Dynamic
  - Data access pattern determinable at runtime.
  - • Phases of sharing patterns
    - Set of sharers changes across synchronization phases
    - eg. BARNES-HUT
  - • Data accesses governed by locks
    - eg. PTHOR

Receiver-initiated

- Producer does not know the next consumer.
- eg. CHOLESKY

Key Results

• Set of explicit communication primitives
  - PSET_WRITE(address, value, pmask)
  - SEL_WRITE(address, value)
  - SYNC_WRITE(address, value, pmask)
  - RSTREAM(address, # bytes)

• Primitives co-exist within the underlying cache coherence scheme.

• Performance results show significant improvements with these primitives.

Summary

The framework

- **Overhead functions.**
- **SPASM.**

Scalability studies of parallel systems.

Evaluating machine abstractions.

Synthesizing bandwidth requirements of parallel applications.

Synthesizing communication mechanisms from an application viewpoint.

Future Directions

Studying other architectural issues and synthesizing requirements from an application view-point.

Broader spectrum of applications spanning diverse domains.

Issues in compile-time and OS related runtime support.

Identifying and quantifying software related overheads.

Parallelizing SPASM.

Performance monitoring/visualization support.

Further Information

http://www.cc.gatech.edu/computing/Architecture/projects/tass.html

- On Characterizing Bandwidth Requirements of Parallel Applications. To appear in Proc. of ACM SIGMETRICS, May 1995.
- Abstracting Network Characteristics and Locality Properties of Parallel Systems. In Proc. of High Perf. Comp. Arch., pages 54-63, January 1995.
- A Simulation-based Scalability Study of Parallel Systems. Journ. of Par. & Dist. Comp., Vol. 22(3):411-426, 1994.
- An Approach to Scalability Study of Parallel Systems. In Proc. of ACM SIGMETRICS, pages 171-180, May 1994.
- Architectural Mechanisms for Explicit Communication in Shared Memory Multiprocessors. Tech. Rep. GIT-CC-94/59.
- A Comparative Evaluation of Techniques for Studying Parallel System Performance. Tech. Rep. GIT-CC-94/38.
- A Computational Model for Message-Passing. In Proc. of Intl. Par. Proc. Symp., pages 358-361, March 1992.
- Experimental Evaluation of Algorithmic Performance on Two Shared Memory Multiprocessors. In Proc. of Intl. Symp. on Shared Memory Multiprocessing, pages 13-24, April 1991.