LogP Network Characterization by Microbenchmarks

David E. Culler

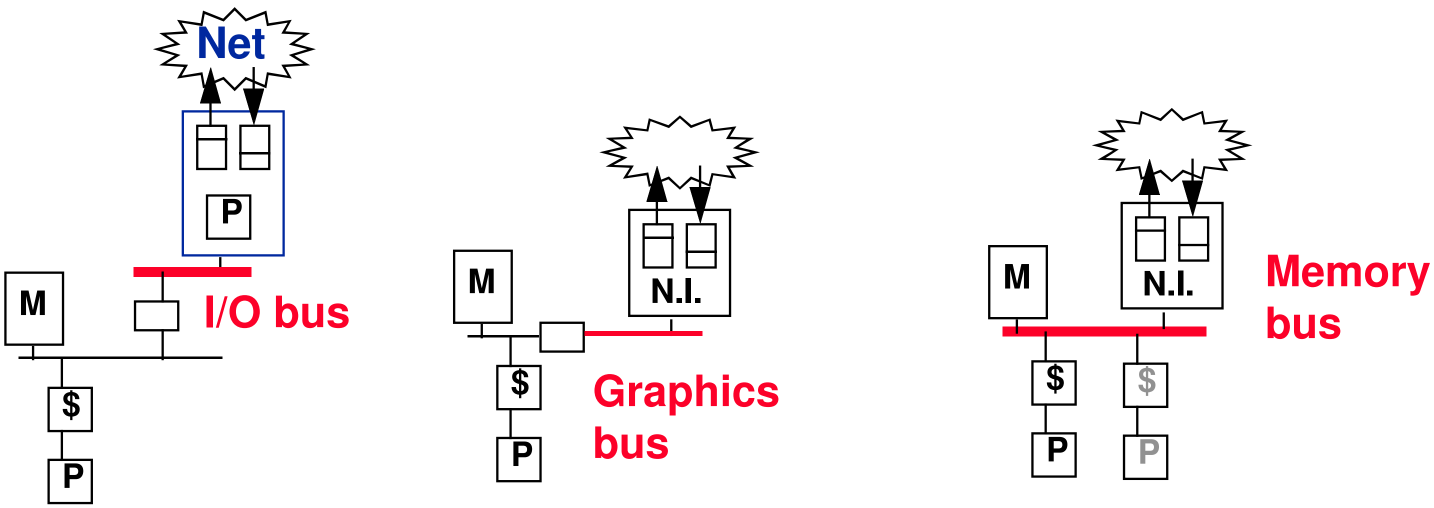
U.C. Berkeley

WOPA, May 1996

(How Practical Models can bring a little Sanity to Practice)

Motivation

- Vast, idiosyncratic design space of network interfaces:



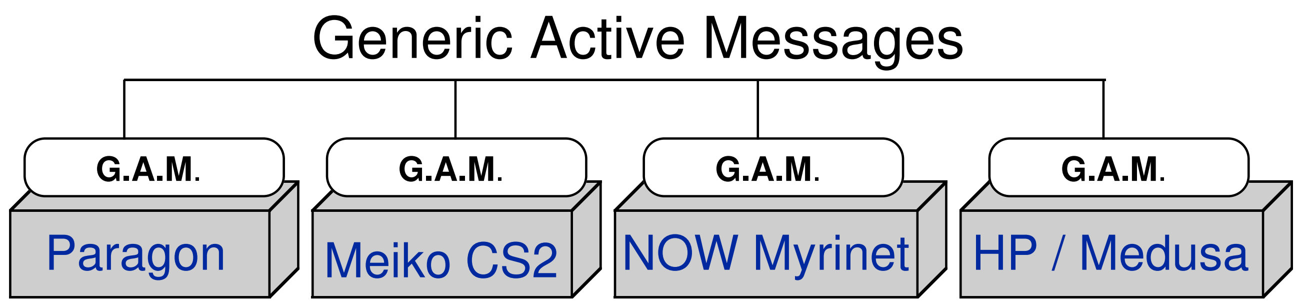
- Where does the network interface attach?
- How much power in the network processor?
- What is the division of labor?

Example Hardware Design Points

	Controller	Embedded Processor	General Purpose Processor
I/O Bus	Sun ATM	Myrinet LANai FORE SBA-200	IBM SP-2
Graphics Bus	HP Medusa		
Memory Bus	Cray T3D	Meiko CS-2	Paragon

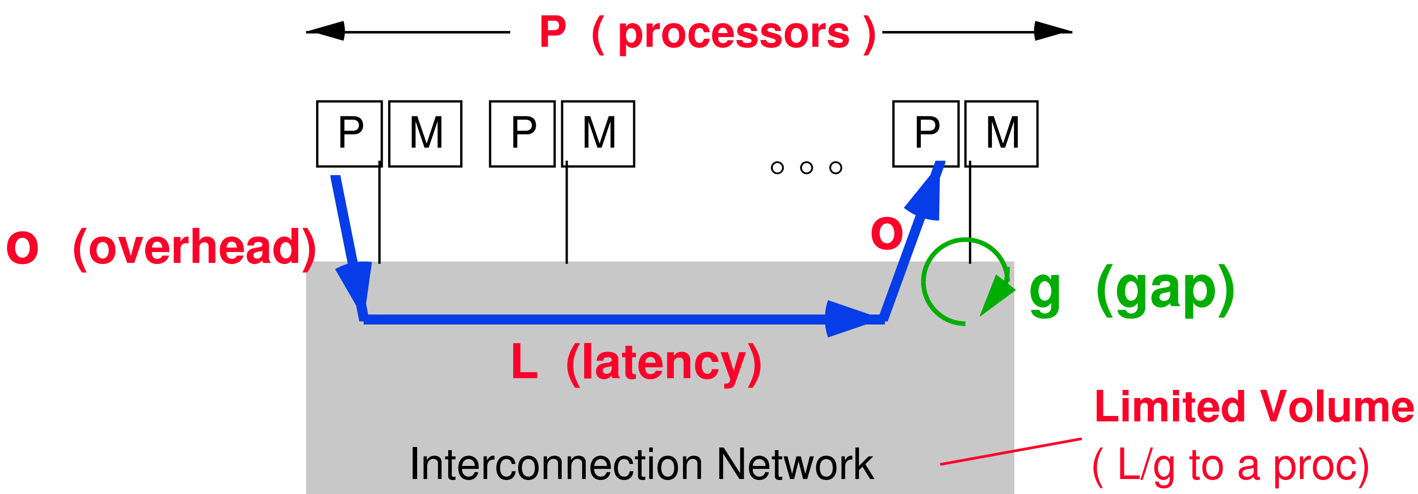
=> want a concise, meaningful framework for evaluating design alternatives

Starting Point: Fast, Generic Msg Layer



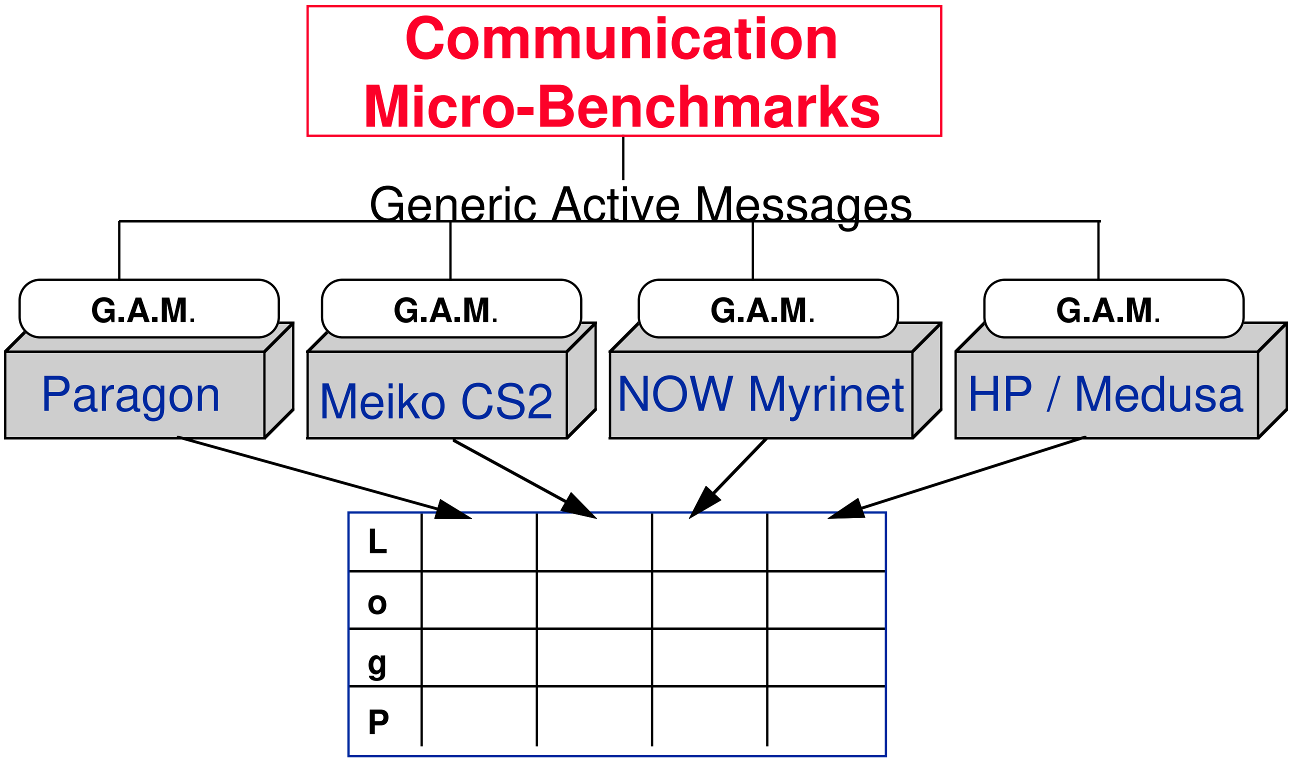
- **Most efficient implementation possible on each platform**
- **How do we make sense of the measurements?**

Proposal: LogP



- **Latency in sending a (small) message between modules**
- **overhead felt by the processor on sending or receiving msg**
- **gap between successive sends or receives** (1/BW)
- **Processors**

Basic Approach



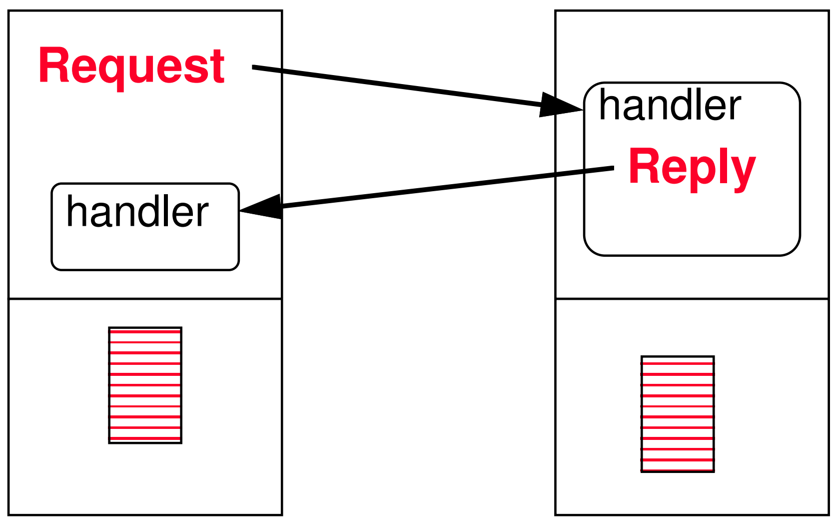
- **Assume the system behaves as LogP**
- **Conduct a set of GAM “gray box” experiments**
- **Analyze “signatures” to LogP deduce parameters**

Outline

- **Background**
  - **LogP**
  - **Generic Active Messages**
- **Microbenchmark Method**
- **Short Message Performance**
- **Bulk Transfer Performance**

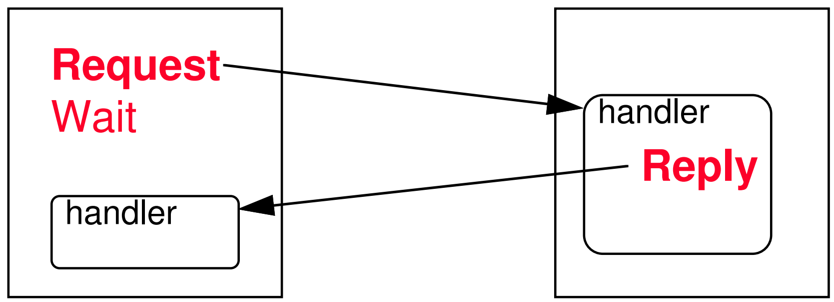
Generic Active Messages (GAM)

- Highly optimized communication layer on a range of HW
- Request / Reply small active messages (RPC)
- Bulk-Transfer (store & get)



Short Message LogP Performance

Time for a Complete Operation



	Intel Paragon	Meiko CS-2	SS20 Myrinet	HP 735 Medusa
Round Trip (μs)	**19.9 - 20.1**	**20.3 - 21.6**	**30.6 - 31.5**	**33.8 -**
$2(O_s + L + O_r)$				

Time to Issue a Sequence of Requests

Start Timer

Repeat *M* times

Issue request

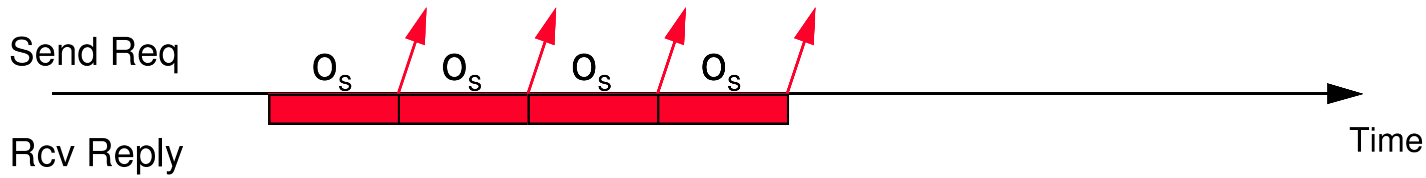
Stop Timer

. . . things complete

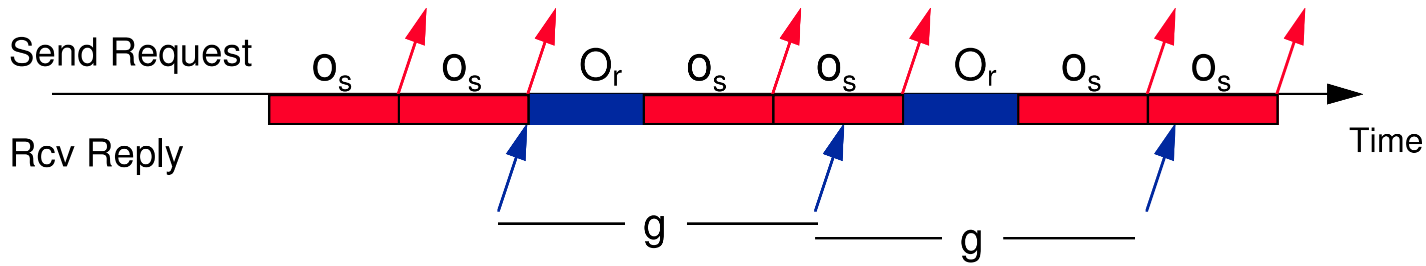
Time(M)

Time to Issue Sequence

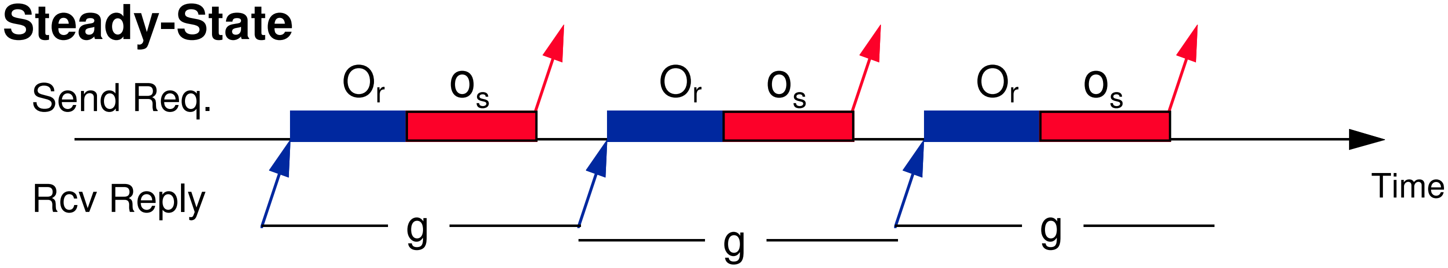
Send Only



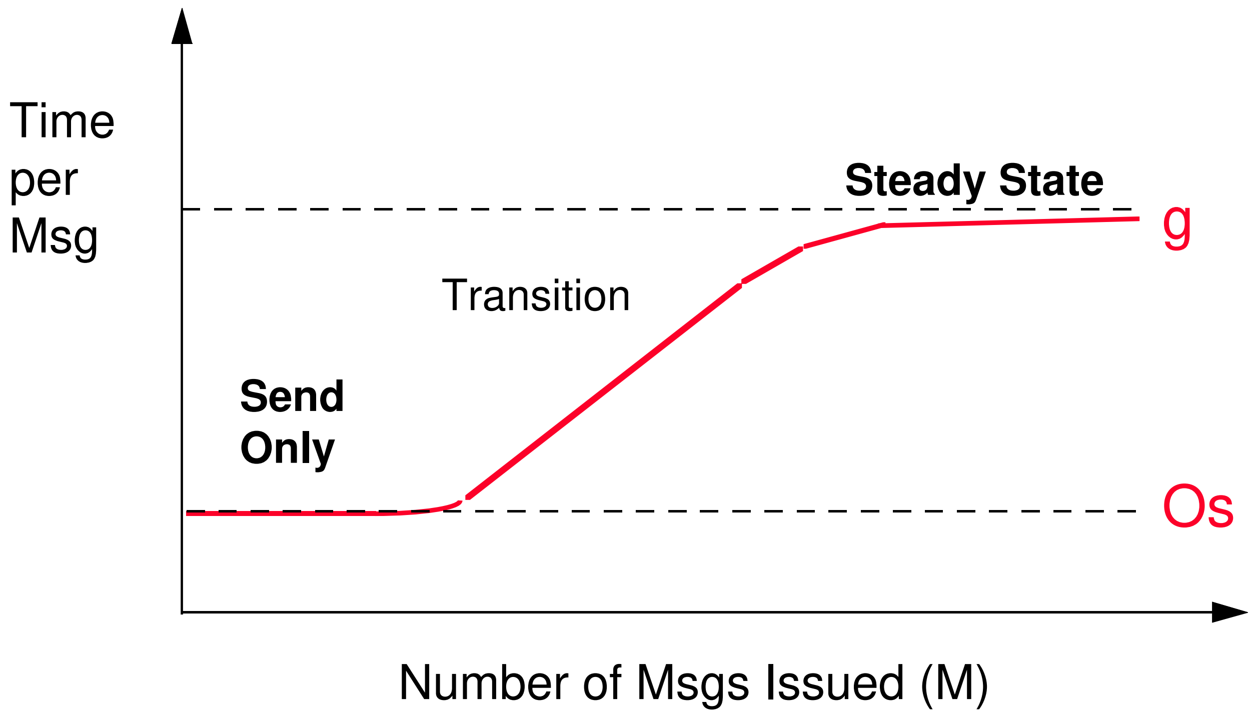
Transition



Steady-State



Analysis of Issue Profile



Add compute interval Δ between requests

For Range of Δ and *M*

Repeat *M* times

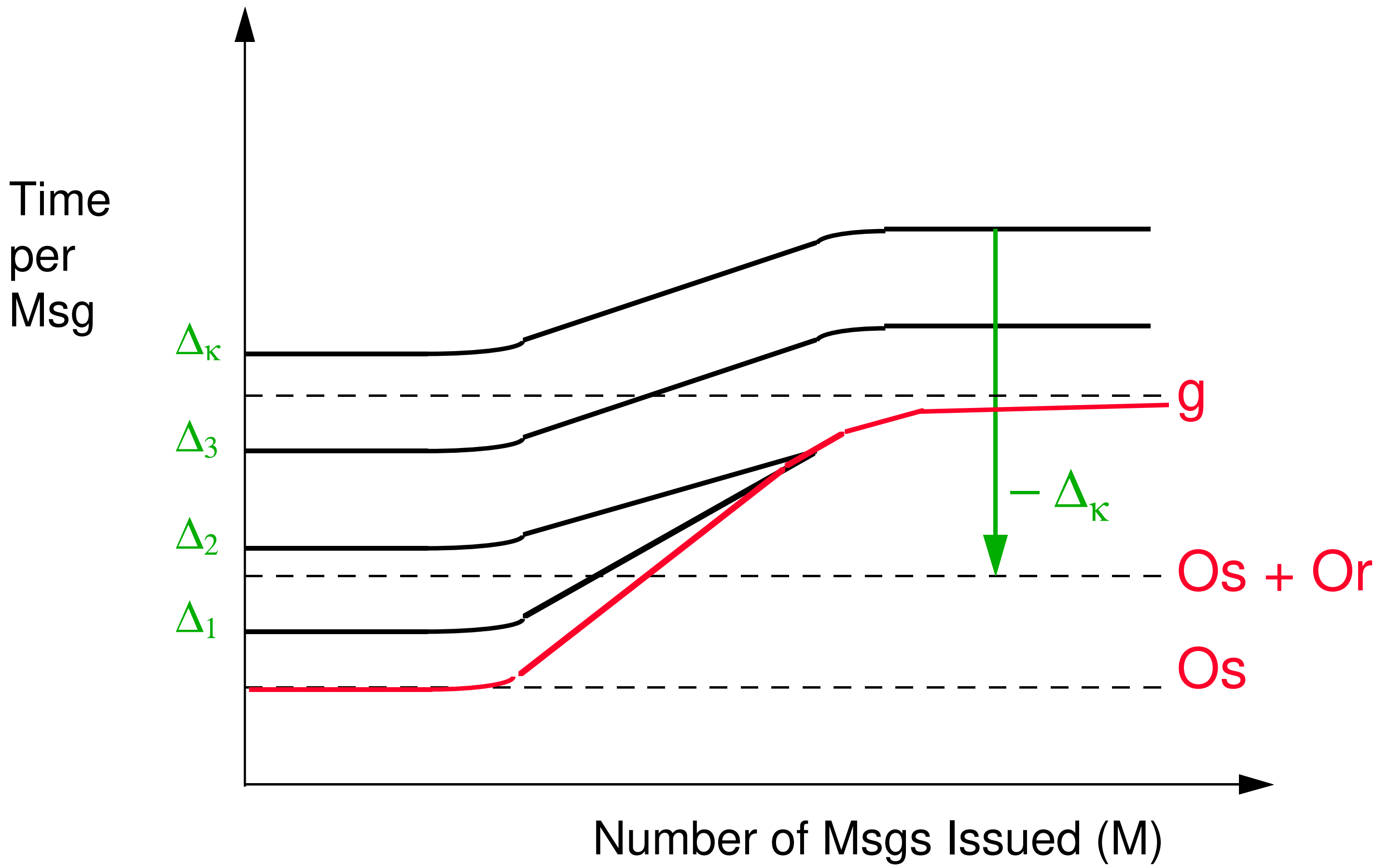
Issue request

Compute for Δ

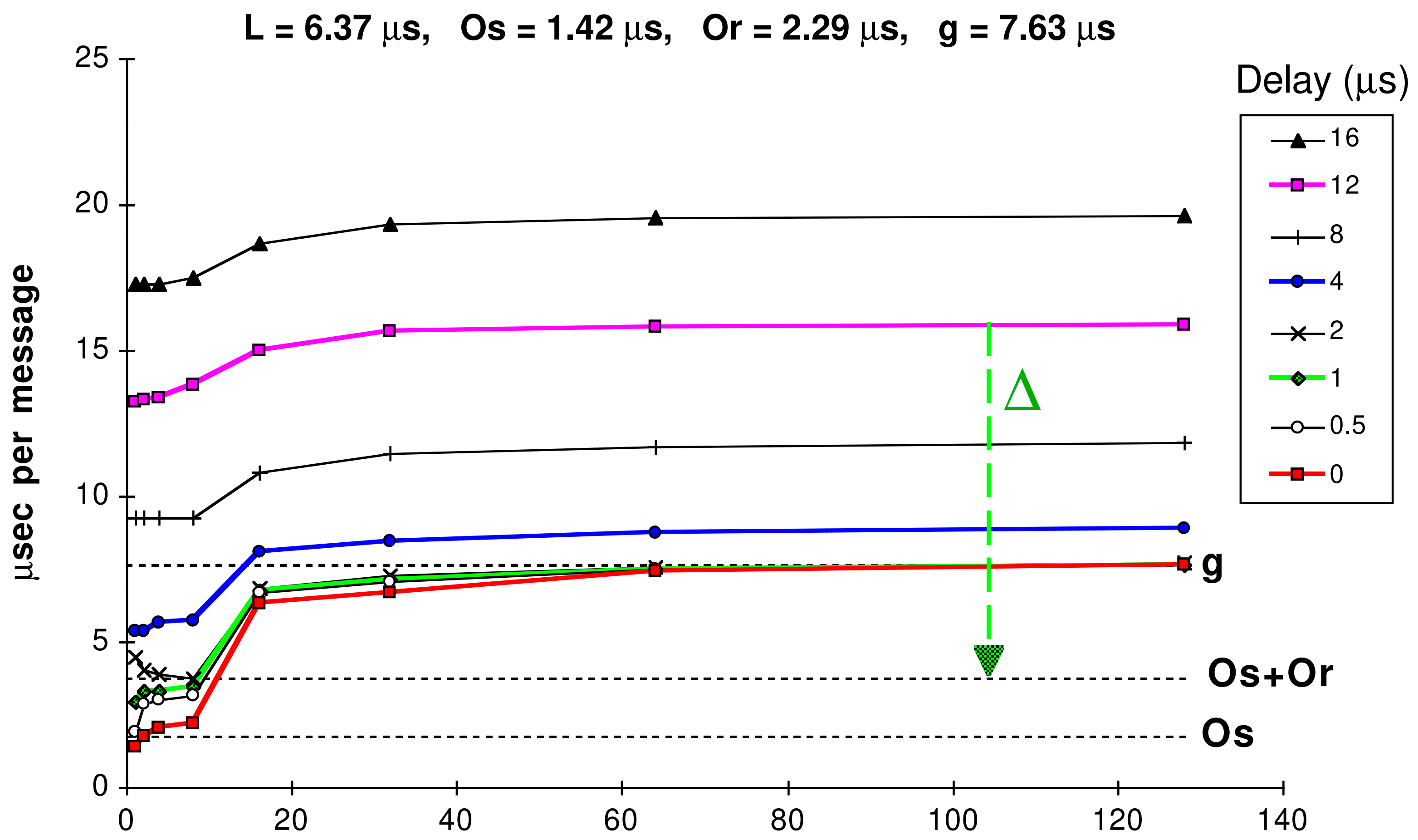
Time(***M*** , Δ)

... things complete

Extracting O and Gap from Issue Profile



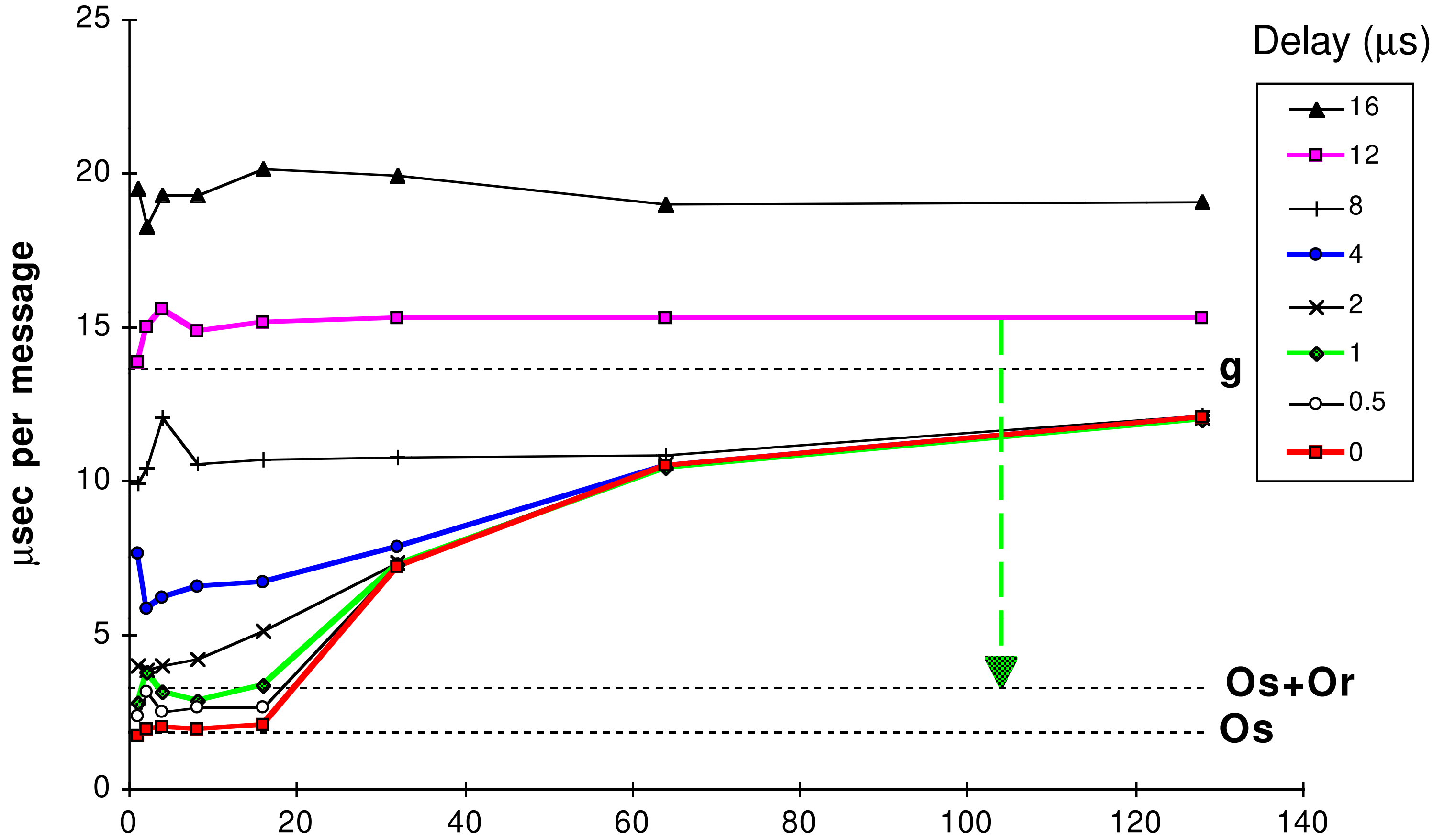
Paragon Microbenchmark Signature



Credit: Lok Tin Liu

Meiko CS-2 Microbenchmark Signature

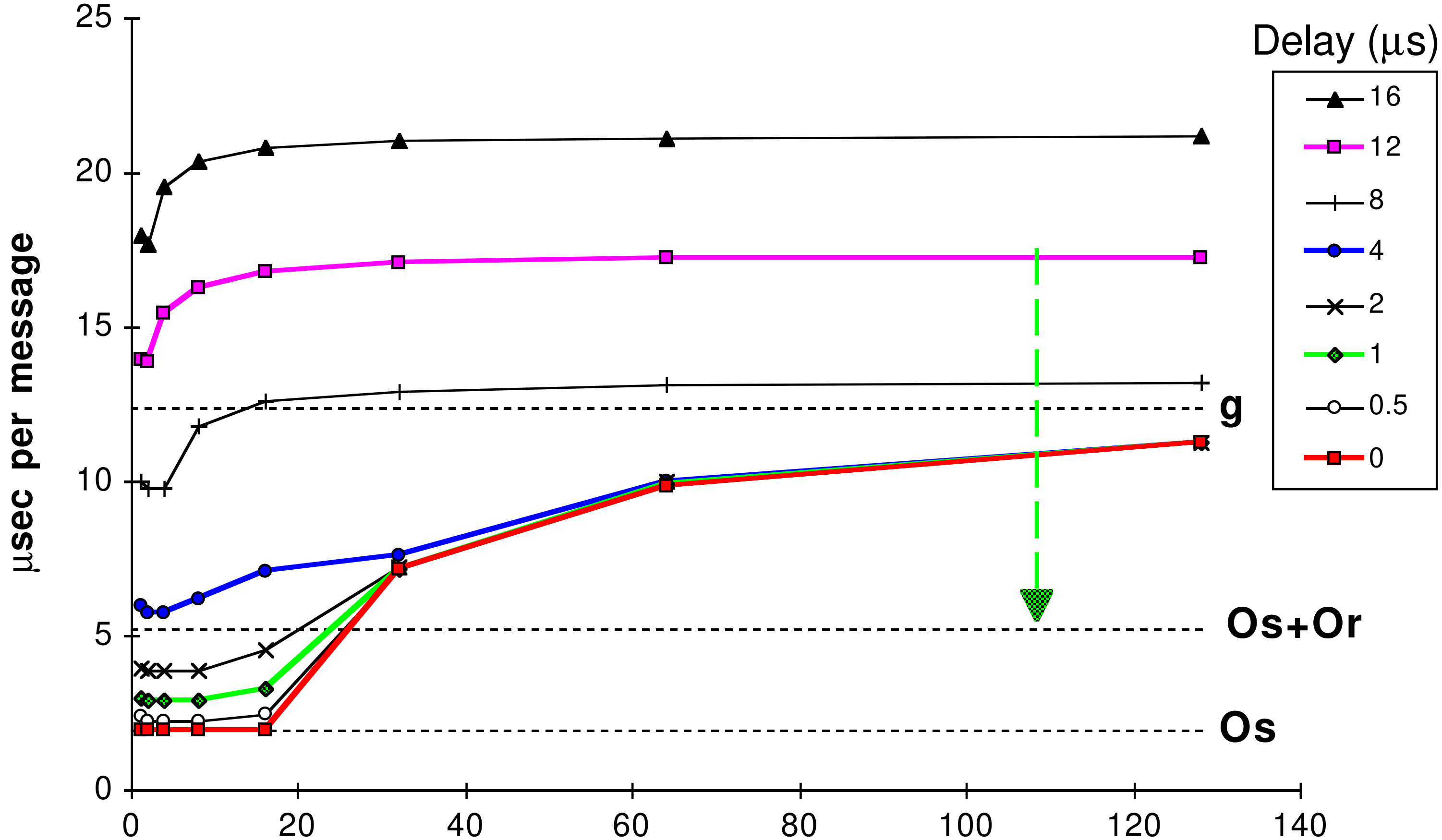
L = 7.5 μs,   Os = 1.7 μs,   Or = 1.6 μs,   g = 13.6 μs



Credit: Chad Yoshikawa

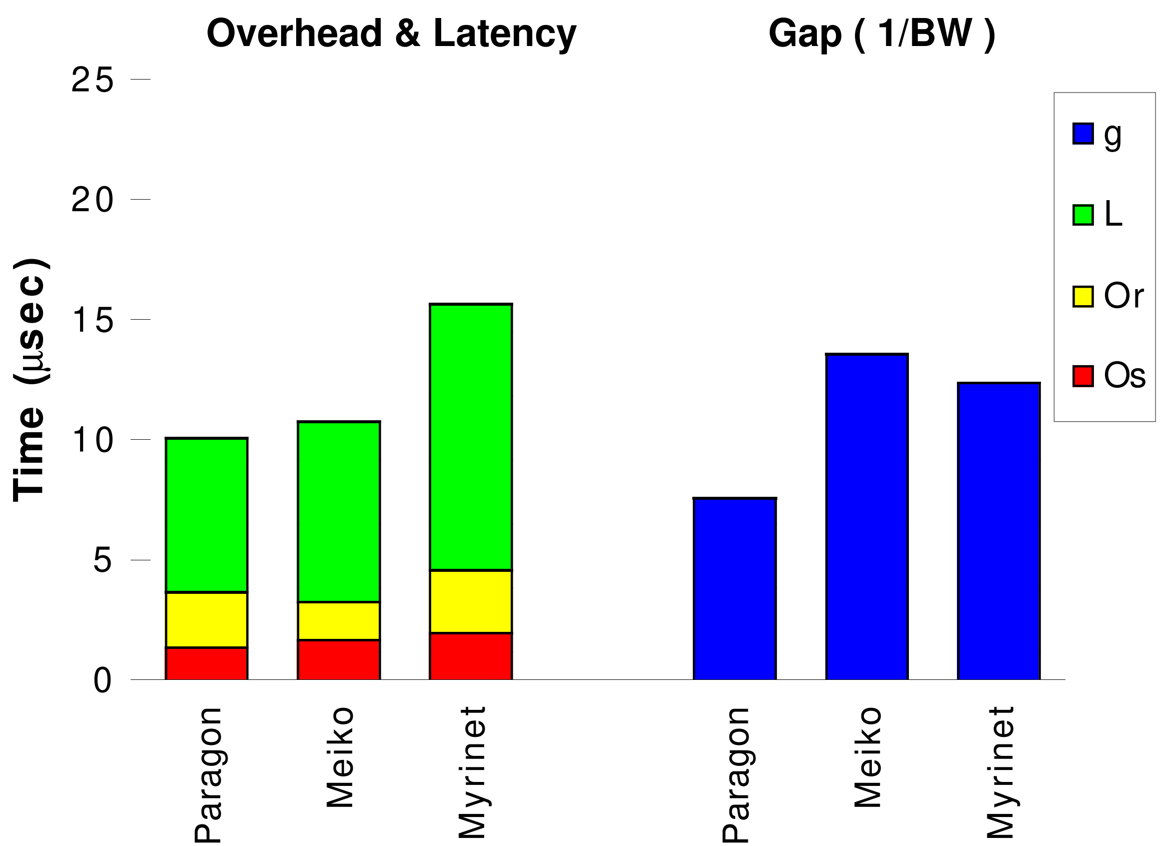
SS20 Myrinet Microbenchmark Signature

L = 11.1 µs, Os = 2.0 µs, Or = 2.6 µs, g = 12.4 µs



Credit: Richard Martin

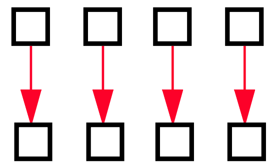
LogP Summary



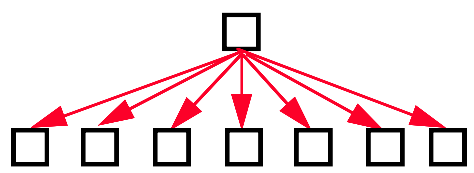
Uncontended Validation

- **LogP says 1-1, 1-k, k-k should behave the same as long as no destination contention**

- **half-to-half measurement => g * 1.0**

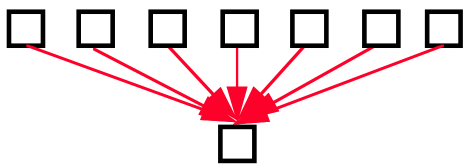


- **one-to-many measurement => g / 1.3**



Contended Validation

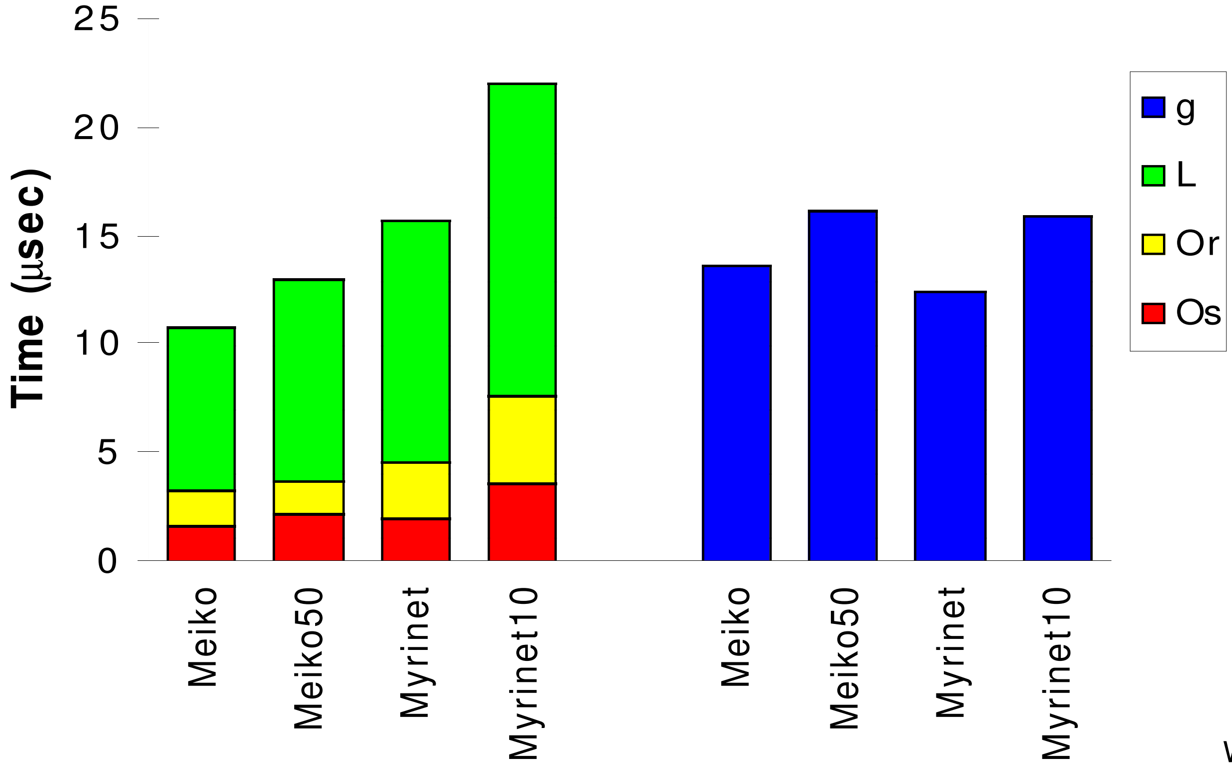
- **K to 1 should increase g by K**



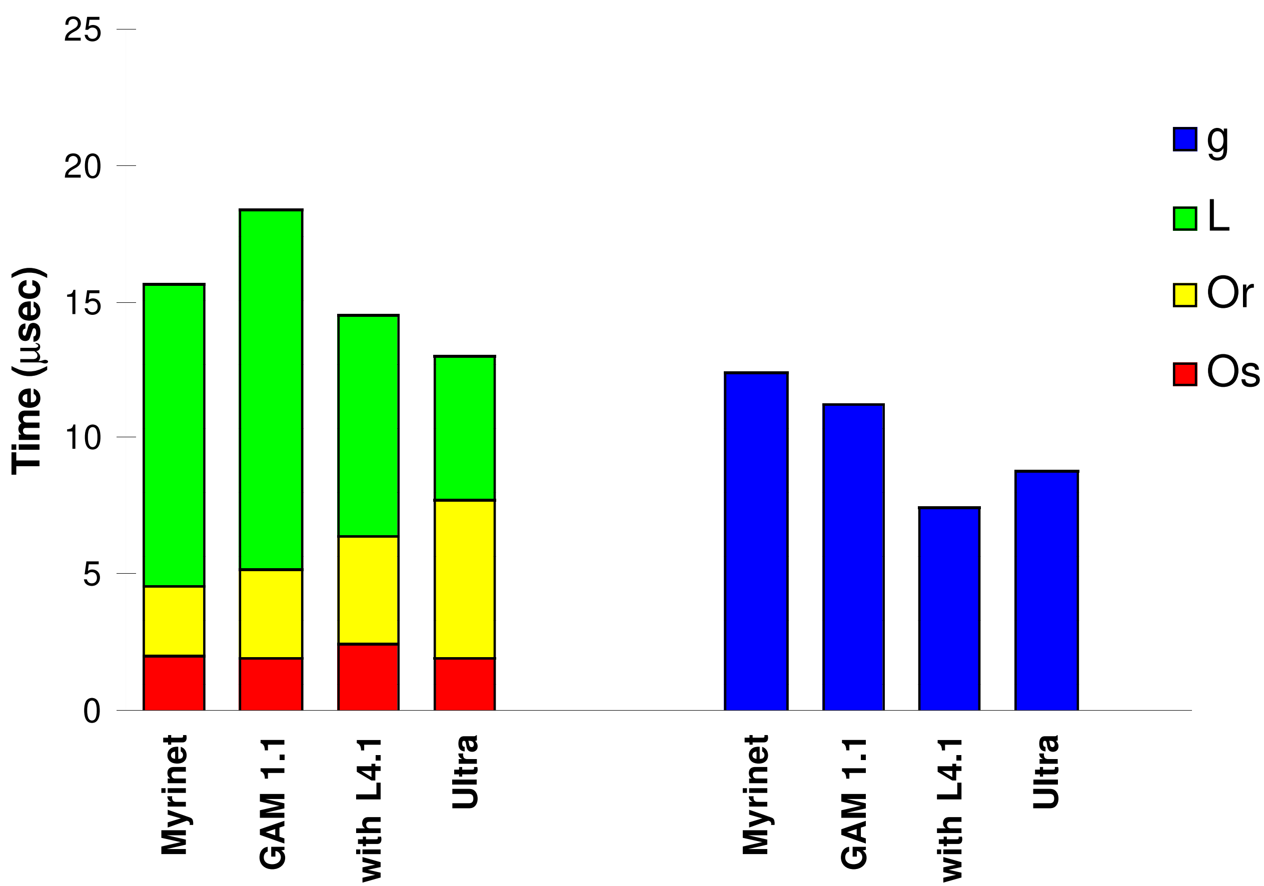
	Paragon	Meiko	Myrinet
K	7	15	3
gk / g1	6	16.8	3

Evaluating HW Alternatives

- **A faster Meiko**
  - **40 -> 45 MHz MBus, 50-> 66 MHz Processor**
- **Switching to an SparcStation with L2 cache**

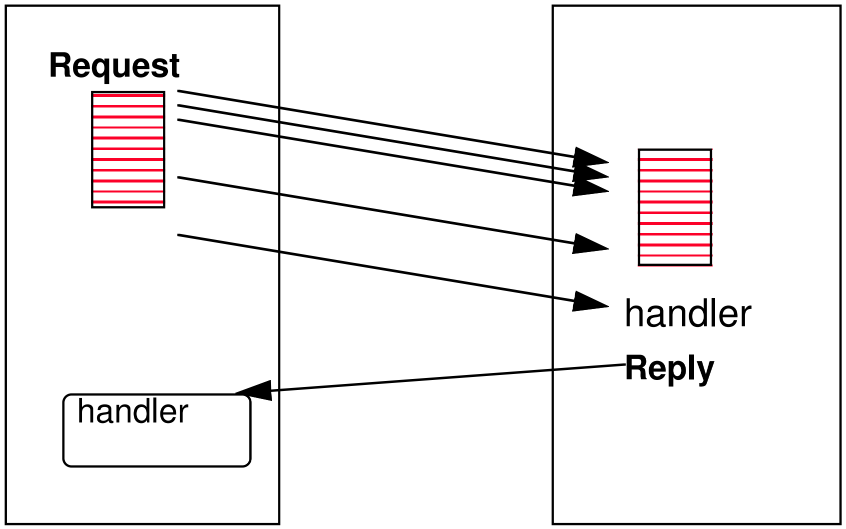


Alternative NI Software & Hardware

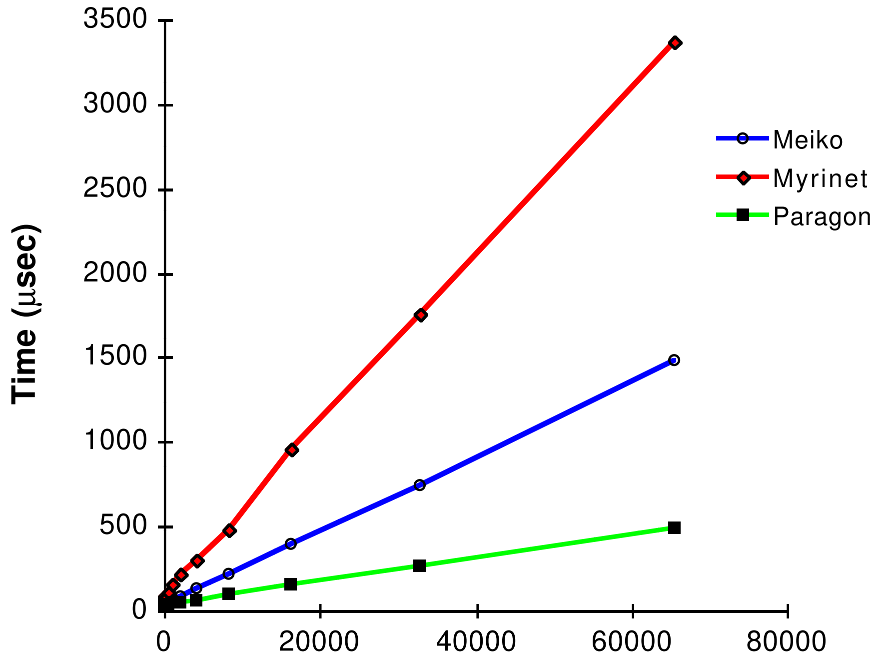


Bulk Transfer Performance

Time for Complete Transfer Operation

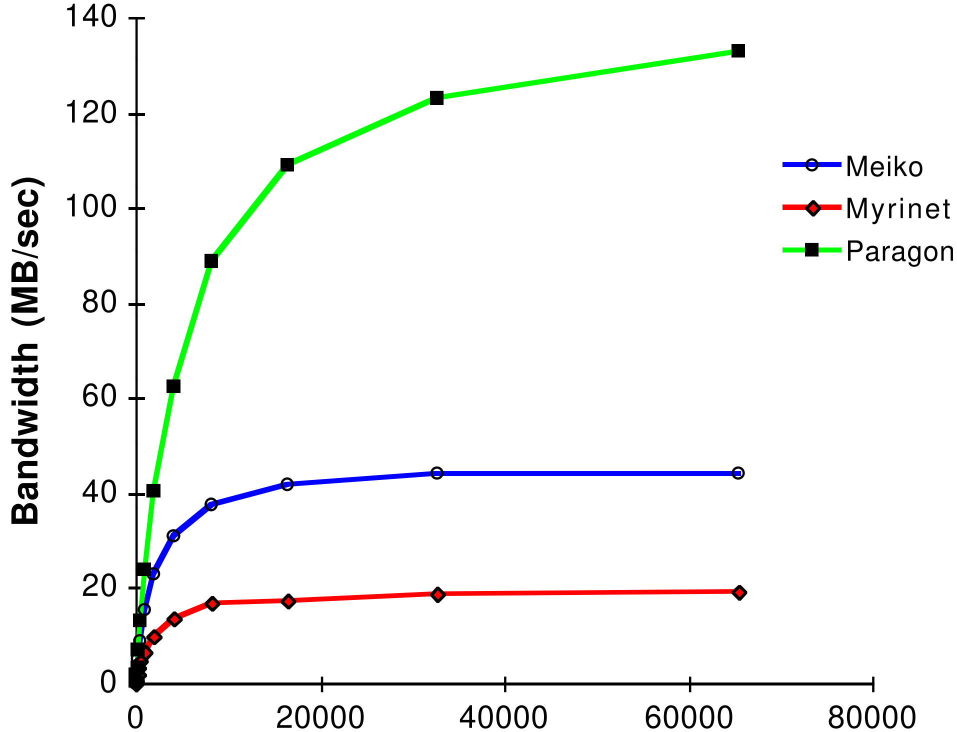


Single Bulk-Transfer Time: T(n)



- **Typically Fit to linear model: $T_0 + n/B$**
  - **T_0 often meaningless**
  - **completion ambiguous**
  - **Does not reveal processing "overhead"**

Single Bulk-Transfer Bandwidth



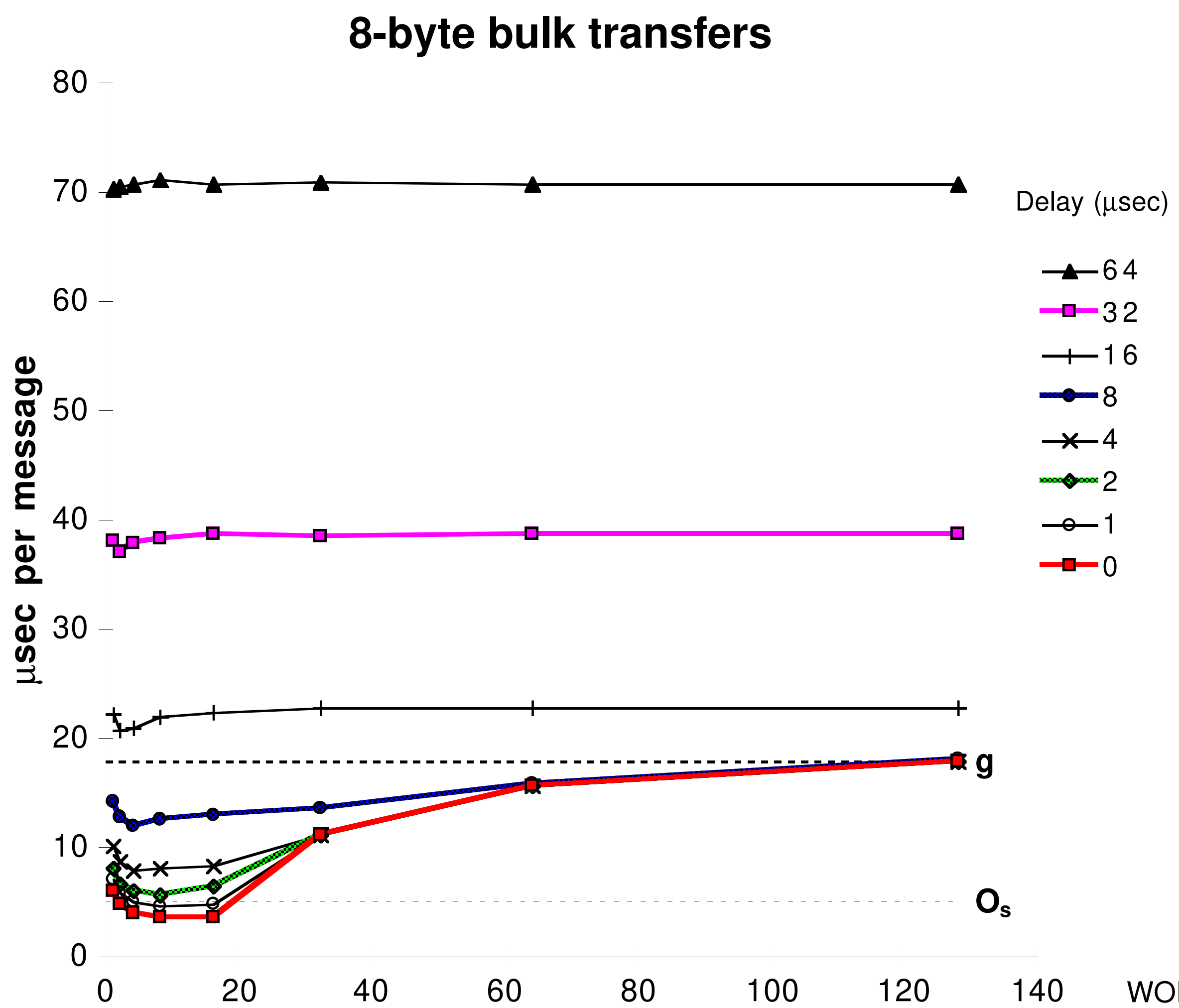
- **BW(n) = n/ T(n) carries no more information**

Solution: elevate LogP

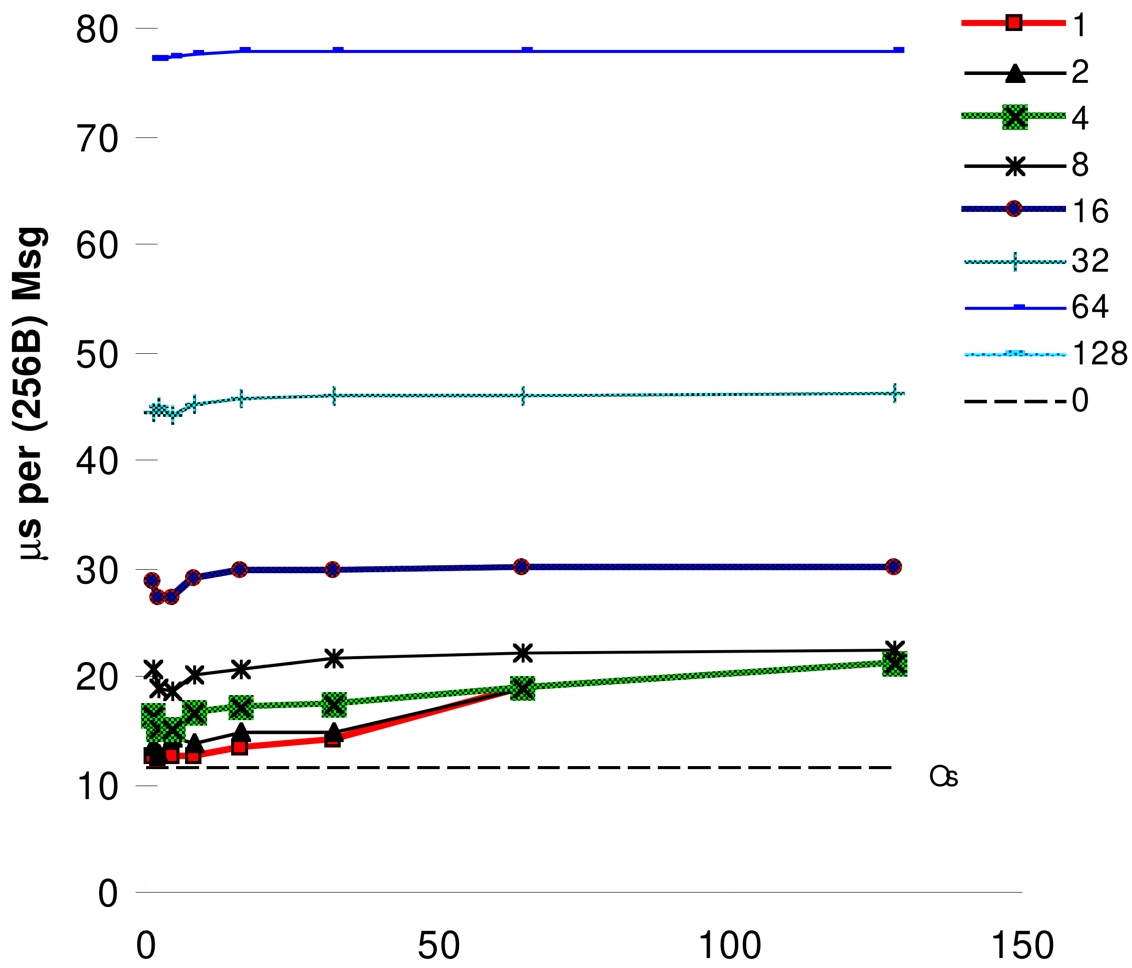
- $T(n) = O_s(n) + L(n) + O_r\ (n)$

- New operation every $g(n)$

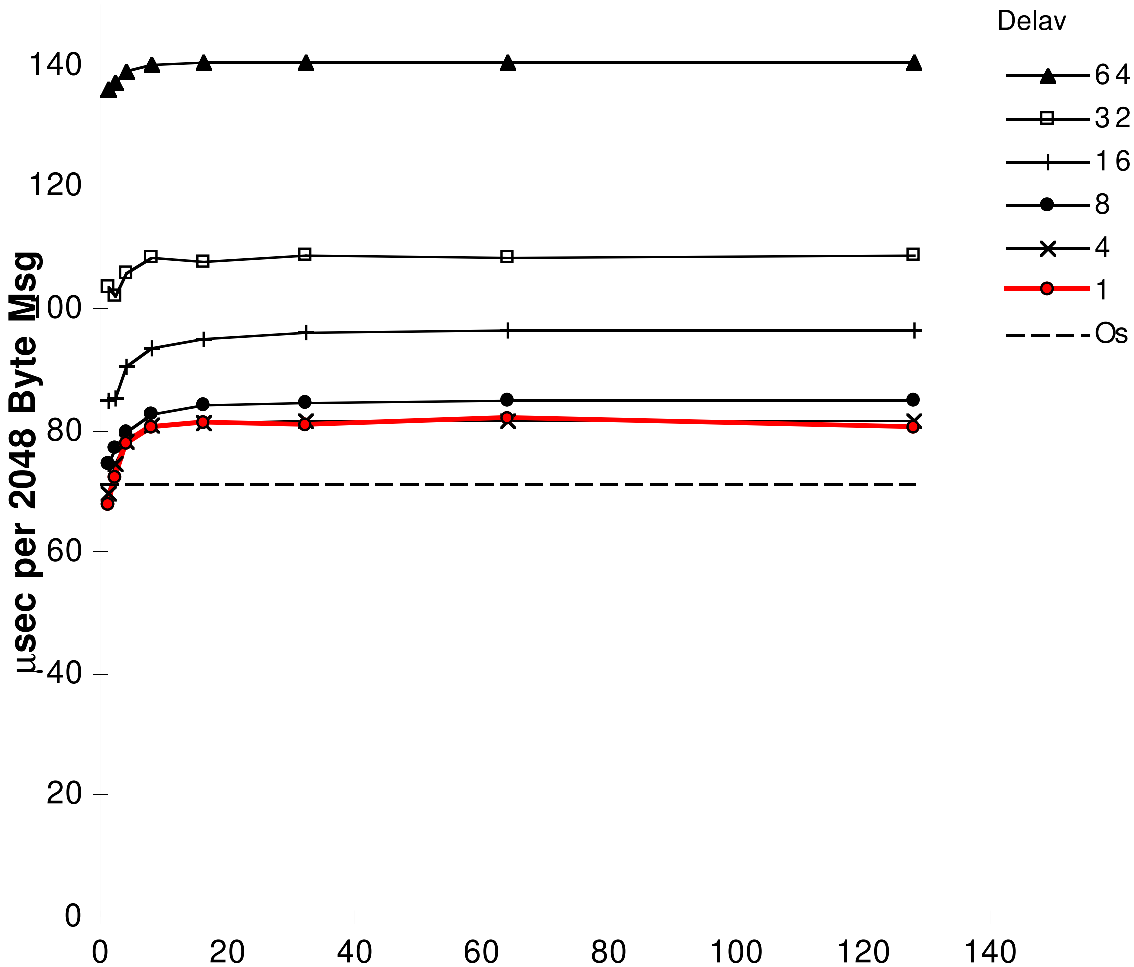
Overhead & Gap from Bulk Transfer Issue



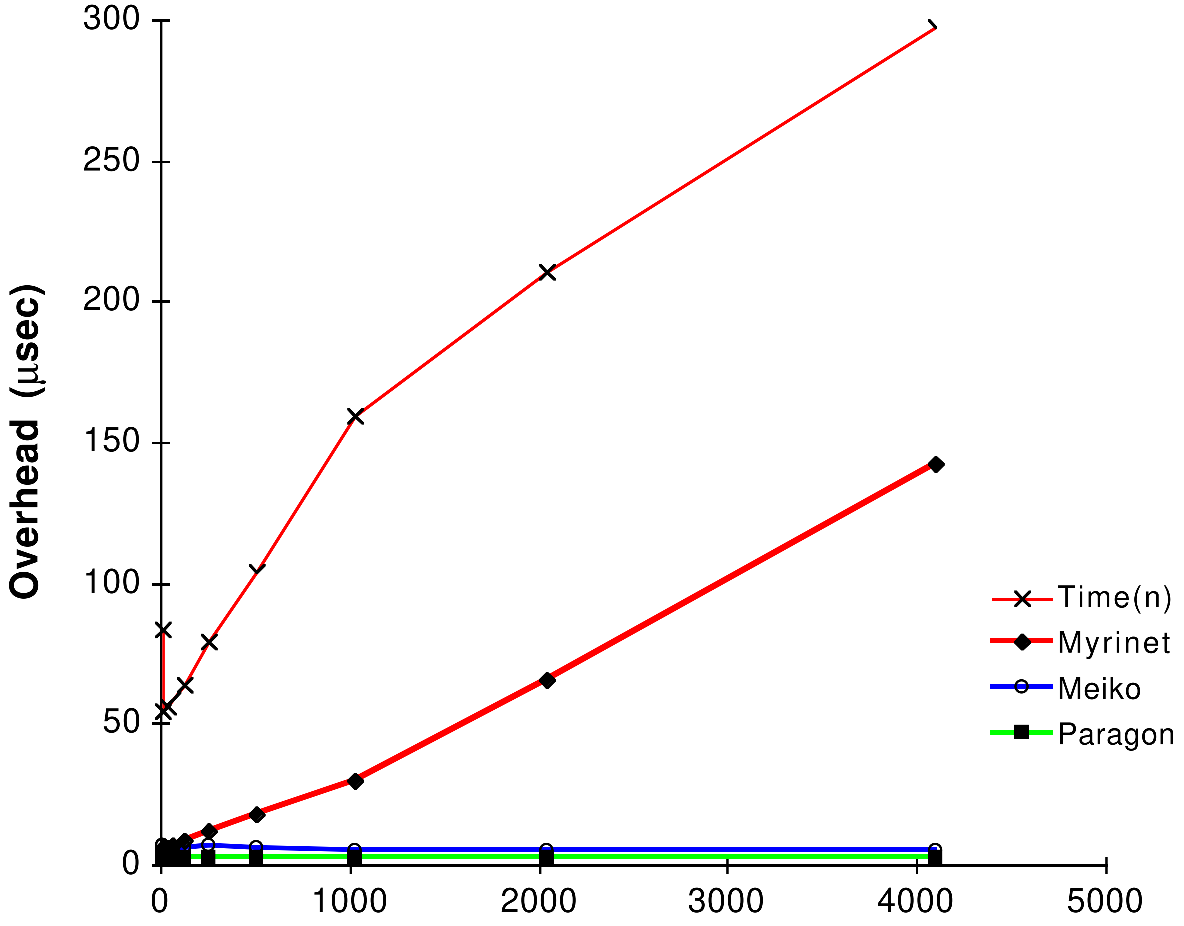
256 Byte Bulk Transfers



2048 Byte Transfer

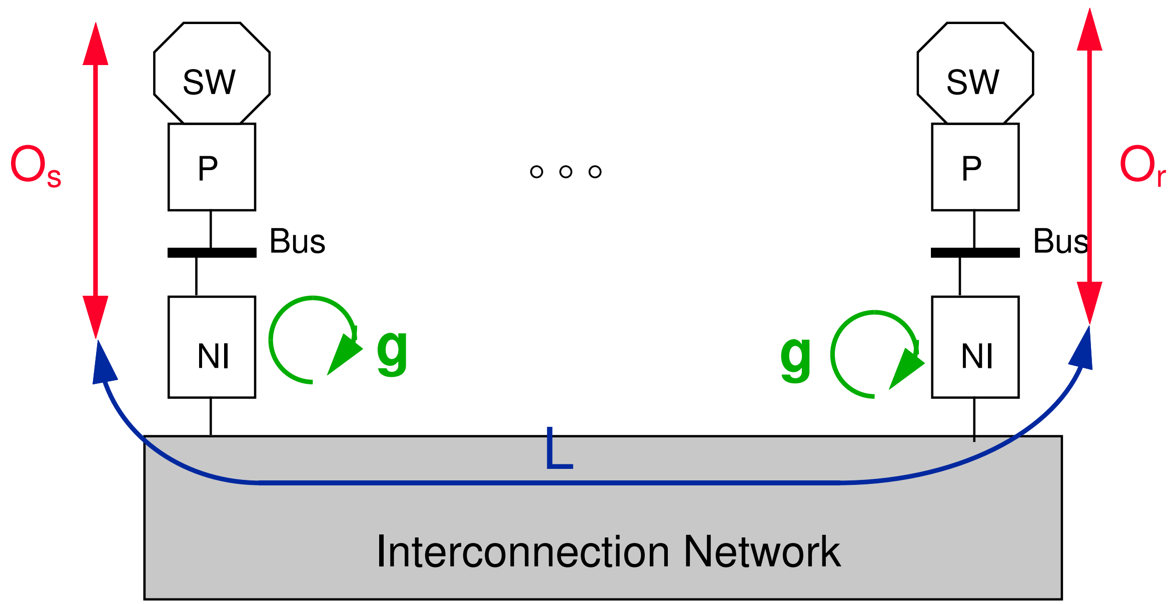


Bulk Issue Overhead Comparison



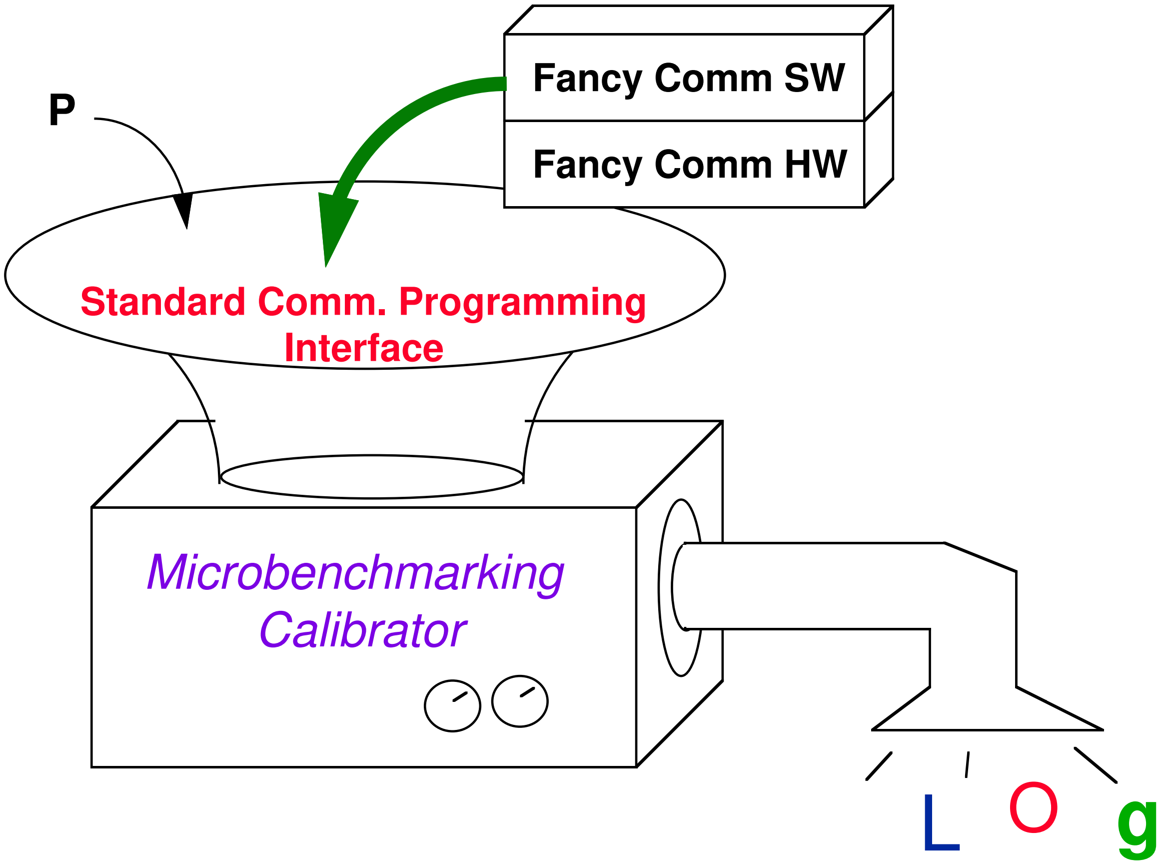
Conclusions

Log Parameterize Salient HW/SW behavior



Consensus on bridging model => Better Machine Summary

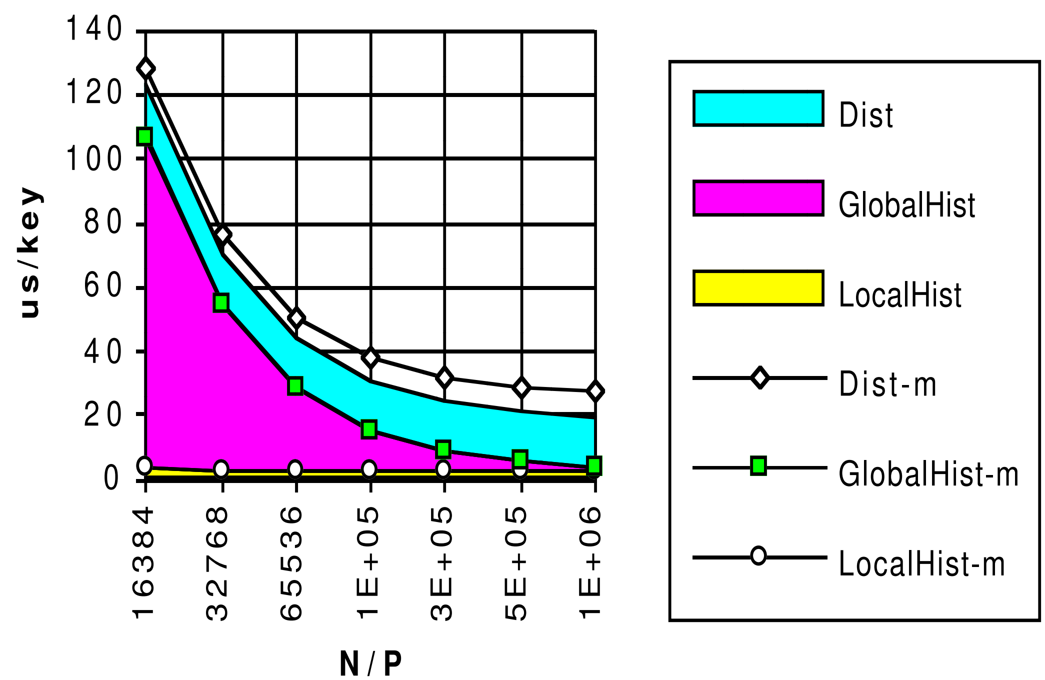
Parametric Summary should be empirical



PVM, MPI, UDP, Shared Mem ... Anyone ?

Predictable Parallel Prog. Performance

Example: Histo-radix Sort on 512 Processor CM-5



- **enables adaptive algorithms**
  - **select strategy based on model coefficients for platform**

To find out more

- "Assessing Fast Network Interfaces", IEEE Micro, Feb. 1996
- "Toward a Practical Model of Parallel Computing", PPOPP 93
  – references cover many alternatives
- http://now.cs.Berkeley.edu/

Asynchronous RPC