
Models for Understanding Parallel Performance

Dr. John Mellor-Crummey

**Department of Computer Science
Rice University**

`johnmc@cs.rice.edu`

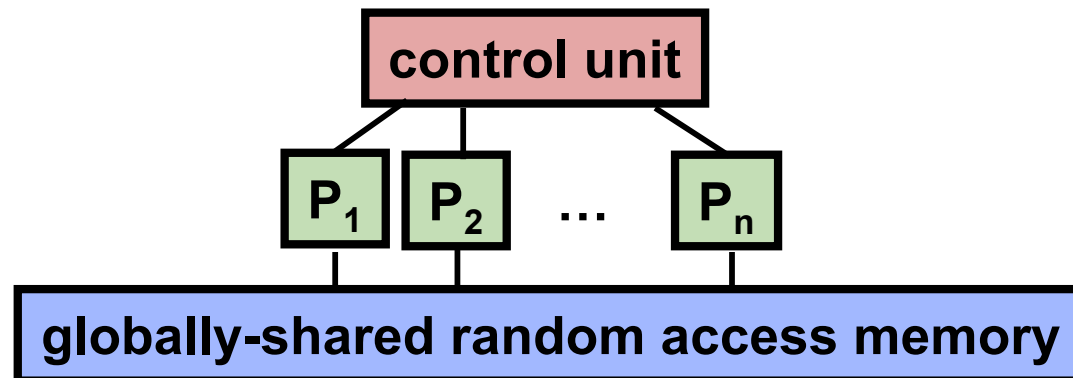


Topics for Today

- A brief tour through older parallel models
 - PRAM
 - BSP
- LogP model
- LogGP refinements
- Modeling performance of modern networks with LogGP

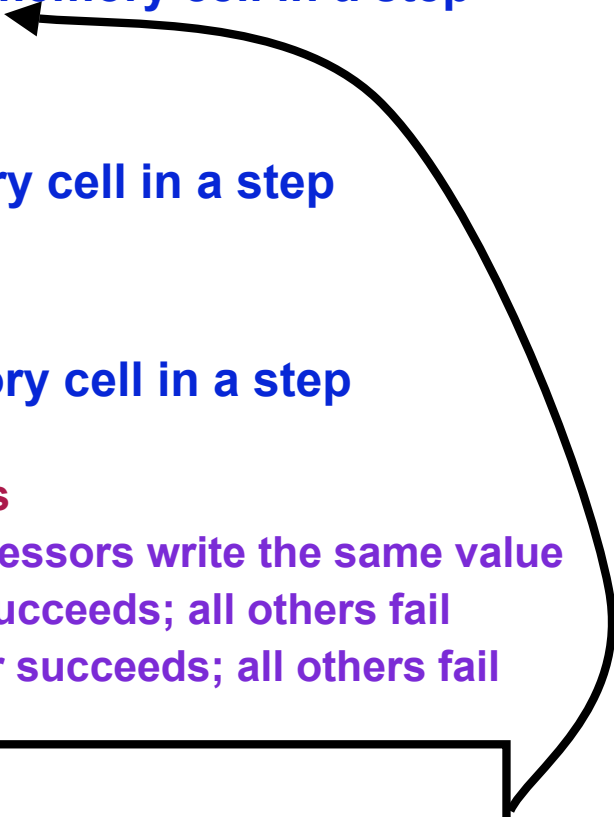
Parallel Random Access Memory (PRAM)

- Idealized abstraction of parallel systems
 - used for understanding design of parallel algorithms
 - not subject to physical limitations of realizable machines



- Properties
 - Single-Instruction-Multiple-Data (SIMD)
 - all processors perform the same operation in a cycle
 - polynomial number of processors
 - polynomial amount of shared memory
 - uniform latency operations: read, write, compute

PRAM Variants

- **Exclusive-Read-Exclusive-Write (EREW)**
 - at most one processor can read or write any memory cell in a step
 - **Concurrent-Read-Exclusive-Write (CREW)**
 - any processor can read any location
 - only one processor may write any one memory cell in a step
 - admits a larger class of algorithms
 - **Concurrent-Read-Concurrent-Write (CRCW)**
 - each processor can read or write any 1 memory cell in a step
 - no consideration of memory contention
 - variants depend on handling of write collisions
 - common: assumes that all competing processors write the same value
 - arbitrary: one arbitrary processor's write succeeds; all others fail
 - priority: write by highest priority processor succeeds; all others fail
- **Queue-Read, Queue-Write (QRQR)**
 - permits concurrent reads and writing to shared-memory locations
 - cost: α the # of readers/writers to any one memory cell in a given step
- 

Why Use a PRAM Model?

- Suited for the development/analysis of optimal algorithms
 - facilitates development of algorithms for ideal systems
 - future systems may well look more like ideal parallel machines
- A link to theory: complexity class NC
 - problems solvable on a PRAM with
 - polynomial number of processors, $O(n^k)$
 - poly-logarithmic time, $O((\log n)^c)$
 - problems solvable with Boolean circuits of
 - polynomial size, $O(n^k)$
 - poly-logarithmic depth, $O((\log n)^c)$
 - fan-in 2
 - NC = class of problems that can be solved efficiently on a parallel computer

<http://www.answers.com/topic/nc-complexity>

Drawbacks of the PRAM Model

- **Has several unrealistic features**
 - zero communication delay
 - infinite bandwidth
- **Consequences of PRAM features**
 - surprisingly fast algorithms PRAM algorithms exist

From: D.A. Bader and G. Cong, A fast parallel spanning tree algorithm for symmetric multiprocessors, IPDPS 2004, Santa Fe, NM.

- **parallel spanning tree**
 - CRCW PRAM: $O(\log |V|)$ time, $O((|E|+|V|)\alpha(|E|, |V|))$ space
- **such PRAM algorithms perform poorly on real machines**
 - no known “efficient” practical parallel implementation

Goal: model of parallel computation to serve as basis for design and analysis of portable parallel algorithms
—applicable to current and future parallel machines

Network Models

- **Communication only between directly-connected processors**
 - other communication explicitly forwarded by nodes along path
- **One step**
 - nodes can communicate with nearest neighbors
 - operate on local data
- **Strength: foster development of realistic algorithms**
 - algorithms matched to network topology, e.g.
 - parallel prefix on a tree; PDE on a mesh; sorting or FFT on butterfly
- **Weaknesses**
 - algorithms for network topologies lack robustness
 - perfect algorithm may require $P = O(\text{data elements})$
 - do not map efficiently to other topologies
 - network-centric model not always appropriate
 - stencil calculations: efficiency = surface to volume ratio
 - modern networks allow cut-through routing

Bulk-Synchronous Parallel (BSP) Model

Bridge theory and practice by restricting programming model

- **Execution as series of supersteps**
 - in one superstep, a processor
 - sends limited number of messages
 - performs local computation
 - receives all messages
 - must allow sufficient time for message routing
 - performs a global barrier
- **Efficient BSP algorithms**
 - overlap communication and computation
 - communication bandwidth and latency can be ignored
 - up to a constant factor
- **Strengths**
 - simple enough to be used for design of portable algorithms
 - enable designer to address key performance issues for algorithms
 - evaluate algorithms using machine performance characteristics

Toward Realistic Models

- **Early 1990s: plethora of architectures is a significant challenge**
 - SIMD
 - vector
 - MIMD
 - systolic arrays
 - dataflow
 - shared-memory
 - message-passing
- **Late 1990s: architectural convergence**
 - microprocessor-based systems
 - each node with cache, large DRAM, network interface
 - variations
 - SMP vs. single CPU nodes
 - processor-network interface

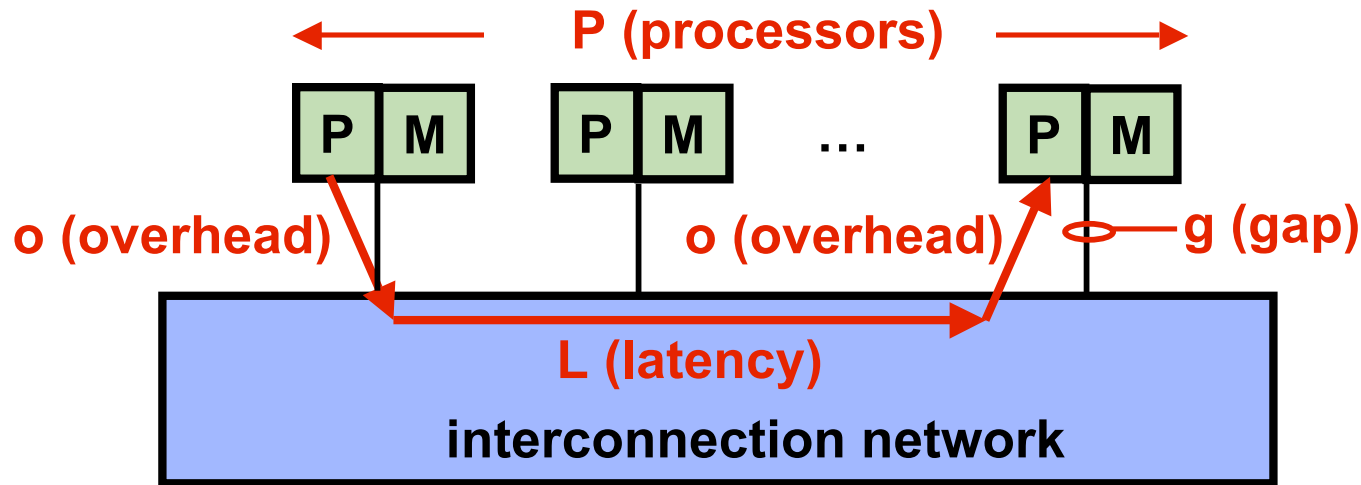
Assumptions for Parallel Algorithms

(for microprocessor-based systems)

- Coarse-grain processors
- Large number of data elements per processor
- Communication bandwidth lags memory bandwidth
—but perhaps not for long!
 - Opteron/Infiniband systems
 - 6.4 GB/s of peak memory bandwidth (DDR 400)
 - PathScale InfiniPath: 1.8 GB/s of bi-directional bandwidth
 - Cal-IT² Optiputer
 - Chiaro Network's Optical Phased Array (OPA) switch: 160Gb/port
- No consensus (yet) on network topology
—exploiting topology unlikely to yield portable algorithms

LogP Model

- Abstract machine model



- 4 performance parameters

- **L**: latency experienced in each communication event
 - time to communicate word or small # of words
- **o**: send/recv overhead experienced by processor
 - time processor fully engaged in transmission or reception
- **g**: gap between successive sends or recvs by a processor
 - $1/g$ = communication bandwidth
- **P**: number of processor/memory modules

Characteristics of LogP Model

- Asynchronous processors that work independently
- Messaging assumptions
 - all messages of small fixed size
 - network has finite capacity
 - $\leq \lceil L/g \rceil$ messages in transit from p to q at once
 - attempting to transmit more causes processor to stall
- Out-of-order network
- Unpredictable msg latency
 - bounded from above by L in absence of stalls
- Notable missing aspect: local computation
 - does not model local computation
 - ignores cache size, pipeline structure

LogP Model Rationale

- **Addresses common issues**
 - communication latency
 - finite bandwidth
- **Suppresses machine specific ones**
 - network topology
 - routing algorithm
- **Compromise between**
 - faithfully representing of execution characteristics
 - providing reasonable framework for algorithm design
 - dropping parameters would encourage impractical algorithms
 - adding parameters would only capture phenomena of modest import

Analyzing an Algorithm with LogP

Good algorithm

embodies a strategy for adapting to different machine parameters

- **Correctness**
 - algorithm must be correct for all msg interleavings, latency $\leq L$
- **Key performance metrics**
 - maximum time used by a processor
 - maximum storage used by a processor
- **What about the model parameters?**
 - often possible to ignore 1 or more without weakening analysis
 - examples
 - for algorithms with infrequent communication
 - reasonable to ignore bandwidth and capacity limits
 - for algorithms with streaming communication
 - transmission time dominated by gaps; L may be ignored
 - when overhead dominates g , g may be ignored

LogP Eliminates Many Loopholes

- **PRAM algorithm loopholes**
 - no penalty for communication → excessively fine-grained
 - each cell independent → neglects contention within M modules
 - processors operate synchronously → no synch overhead
- **LogP addresses each of the above**
- **Another PRAM assumption**
 - no fixed P → concurrency can increase up to F(input size)
 - requires multithreading (MT) to provide this fiction
- **LogP: MT can be used when convenient, but it is not required**
 - don't model context switch overhead
 - multithreading bounded by limit of $\lceil L/g \rceil$ virtual processors

LogP Encourages Practical Techniques

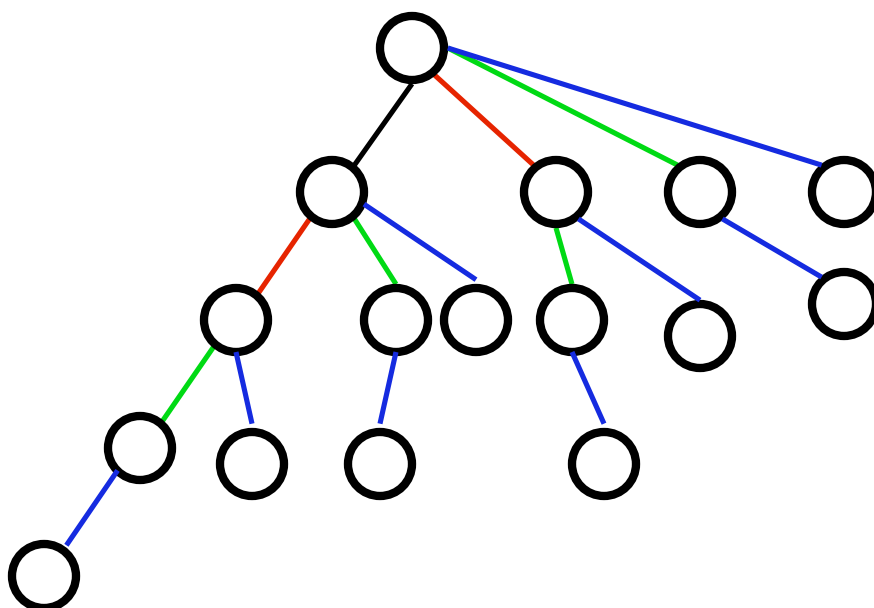
- **Careful data placement: co-locating data and computation**
—avoid communication
- **Careful scheduling**
—communication/computation overlap w/in network capacity limits
- **Balanced communication patterns**
—limits in network capacity discourage swamping a processor

Model Utility Questions

- Do solutions to basic theory problems differ with LogP?
- Does using LogP for design lead to qualitatively good solns?
- Can one predict performance of algorithms on machines?

Optimal Broadcast

- Traditional algorithm: simple, balanced tree
- LogP algorithm: unbalanced tree
 - fan-out determined by relative values of L , o , g
 - nodes that start later must have fewer children
 - accounts for non-zero time to communicate with each



Other Examples

- **Cooley-Tukey FFT**
 - element-wise butterfly communication pattern
 - LogP: consider data layout as part of algorithm design
 - rather than strict blocked or block-cyclic data layout
 - hybrid data layout reduces communication
 - balanced communication schedule for remapping data
 - order of magnitude faster than naïve schedules
- **Triangular solvers**
 - LogP useful for
 - predicting performance
 - deriving lower bounds on running time
 - result: showed neither blocked, block-cyclic layouts optimal

See references in

D.E. Culler, et al. *LogP: A practical model of parallel computation.*

CACM, 39(11):78 - 85, 1996.

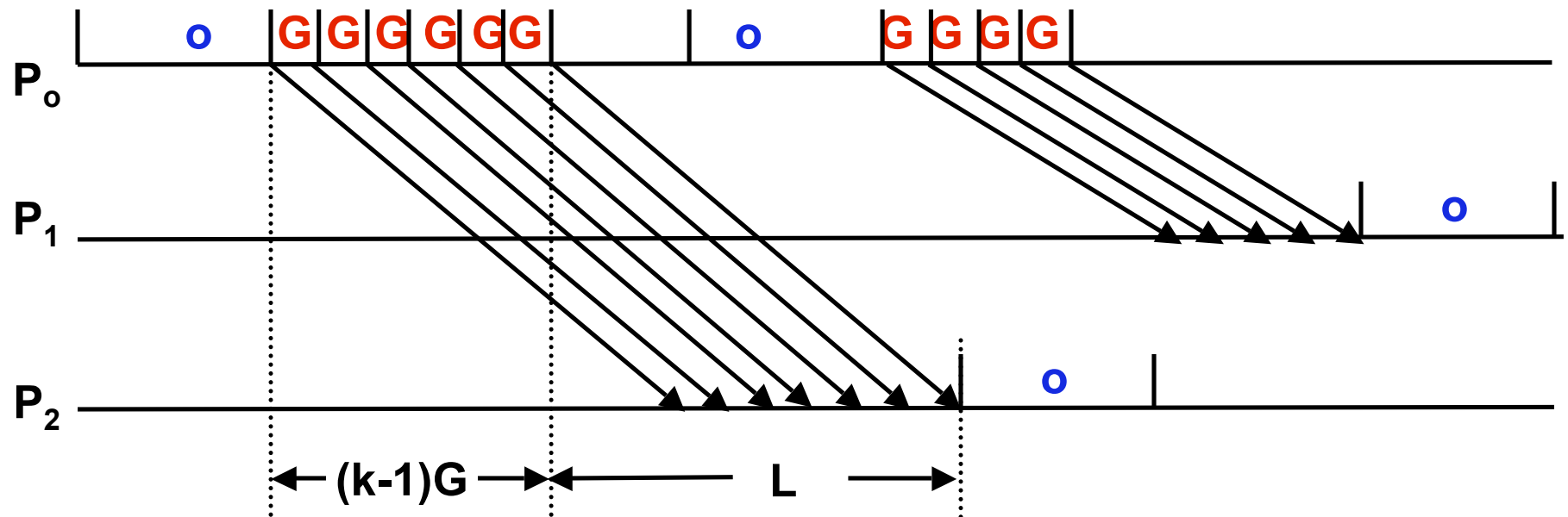
Matching LogP to Real Machines

- Normal msg transmission time: $2o + L$
- Available bandwidth
 - determined by g and network capacity $\lceil L/g \rceil$
 - network is pipeline
 - pipeline depth of L
 - initiation rate g
 - processor overhead o on each end
- Estimating LogP parameters
 - $o = (T_{\text{snd}} + T_{\text{rcv}})/2$
 - $L = Hr + \lceil M/w \rceil$
 - M number of message bits
 - w = channel width in bits
 - H = # hops in longest route
 - r = delay in each hop (in cut-through routing)
 - $g \geq M / (\text{per processor bisection bandwidth})$

LogGP: Account for Long Messages

- **Motivation**
 - LogP: predicts performance for fixed-size short messages only
 - model has implicit 5th parameter: msg size w
 - modern machines support long msgs with higher bandwidth
- **Goal: model performance with both short and long msgs**
- **Modeling long msgs**
 - transmission time: $t = t_o + t_B * n$
 - t_o : startup time - lumps together α and L in of LogP
 - t_B : time per byte
 - insufficiently detailed for short msgs
- **LogGP Approach**
 - extend LogP with additional parameter G
 - G = Gap per byte for long msg = time per byte for long msg
 - $1/G$ = bandwidth for long msg

Sending and Recving with LogGP



- Sending a small message: $2o + L$ cycles
 - o cycles on sender + L cycles in network + o cycles in receiver
- Under LogP, sending k byte msg requires
 - sending $\lceil k/w \rceil$ msgs of size w
 - time = $o + \lceil (k-1)/w \rceil \cdot \max(g, o) + L + o$ cycles
- Under LogGP
 - time = $o + (k-1) \cdot G + L + o$ cycles

Discussion of LogGP Model

- α captures time main processor is involved in sending/recving
- G reflects network bandwidth for long messages
- g captures startup bottleneck of network
 - what does this mean?
 - Meiko CS-2: time for comm co-proc to open comm channel
- Do long msgs matter?
 - yes: Meiko-CS2, Split-C
 - short msg: 2 MB/s
 - long msgs: 33MB/s
- Simplified models
 - for short msgs only: LogGP reduces to LogP
 - for very long msgs only: approximate xfer-time as kG
- Impact on algorithm design
 - aggregate short msgs into long msgs for higher bandwidth

Using LogP and LogGP

Bell et al. *An evaluation of current high-performance networks*, IPDPS, Nice, France, April 2003.

- **Set of network benchmarks for measuring bandwidth, latency, and software overhead**
 - implemented over a wide variety of network APIs,
 - MPI, VIPL, SHMEM, LAPI, E-registers, and GM
- **Gather data for small and large message performance**
 - study many of the supercomputer networks in use today
 - compare MPI performance to that of lower-level network APIs
- **Examine various application speedups that can be achieved via network-related optimizations**
 - overlapping communication and computation
 - msg pipelining
 - msg packing

Benchmarks

- **Ping-pong: measure end-to-end latency (EEL) of 8 byte msg**
 - blocking calls
- **Flood: inject msgs of various sizes as fast as possible**
 - 8B-128KB msgs, non-blocking msgs; queue depth q
 - measure g for small msg
 - measure G for large msg
- **CPU overlap test: determine s/w overhead of send/recv**
 - use flood test of 8B
 - insert computation between non-blocking send/wait
 - when computation exceeds $o+g$, time increases
 - $o = g - (\text{CPU time that can be hidden})$

Results: Send and Recv Overhead

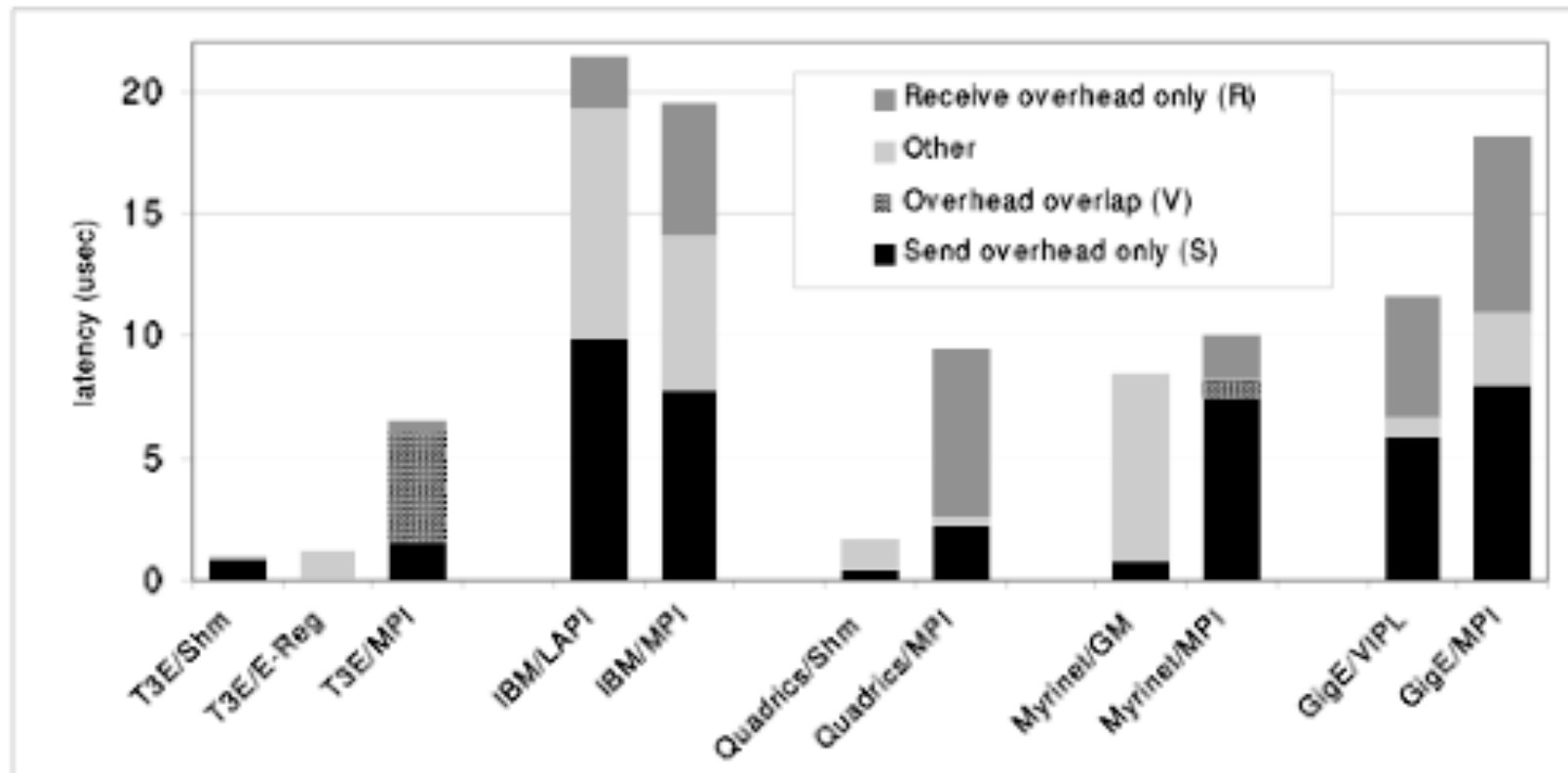
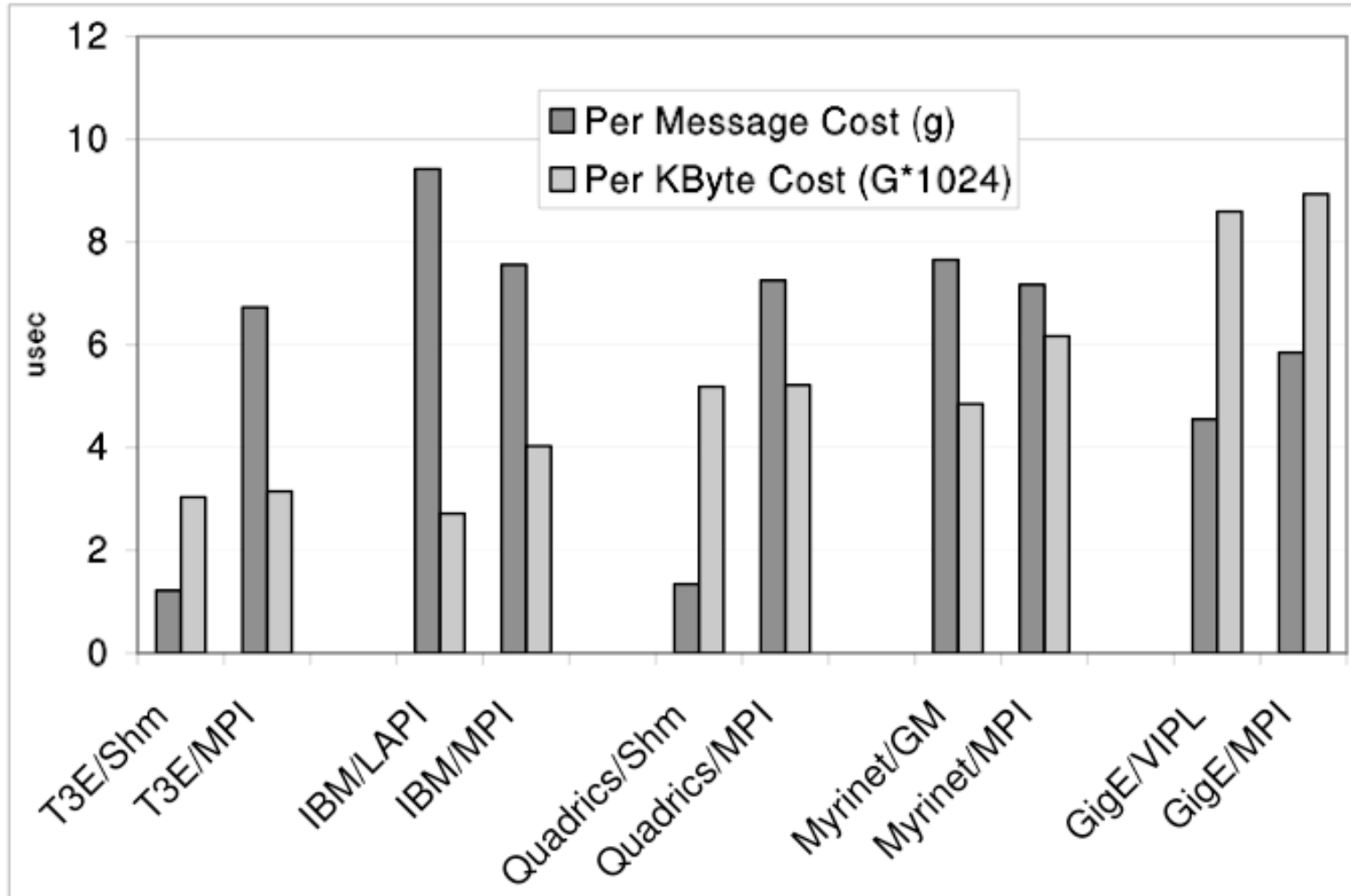


Figure 3. Send and receive software overheads (o_s and o_r) superimposed on end-to-end latency (EEL). For MPI on both the T3E and Myrinet, the sum of the overheads is greater than EEL , and so $o_s = S + V$ and $o_r = R + V$. For the other configurations $o_s = S$ and $o_r = R$.

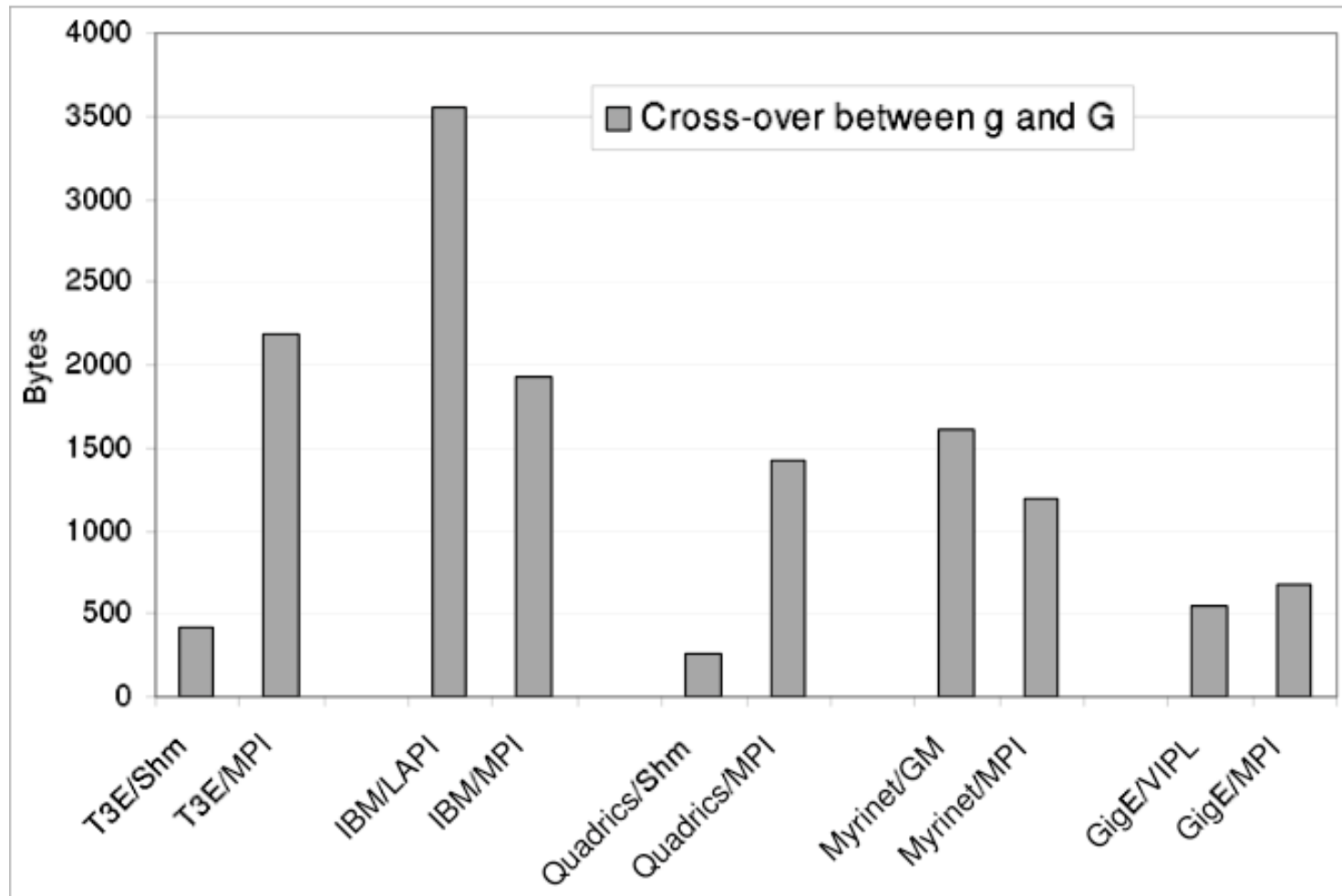
Bell et al. *An evaluation of current high-performance networks*,
IPDPS, Nice, France, April 2003.

Transmission Gaps: G and g



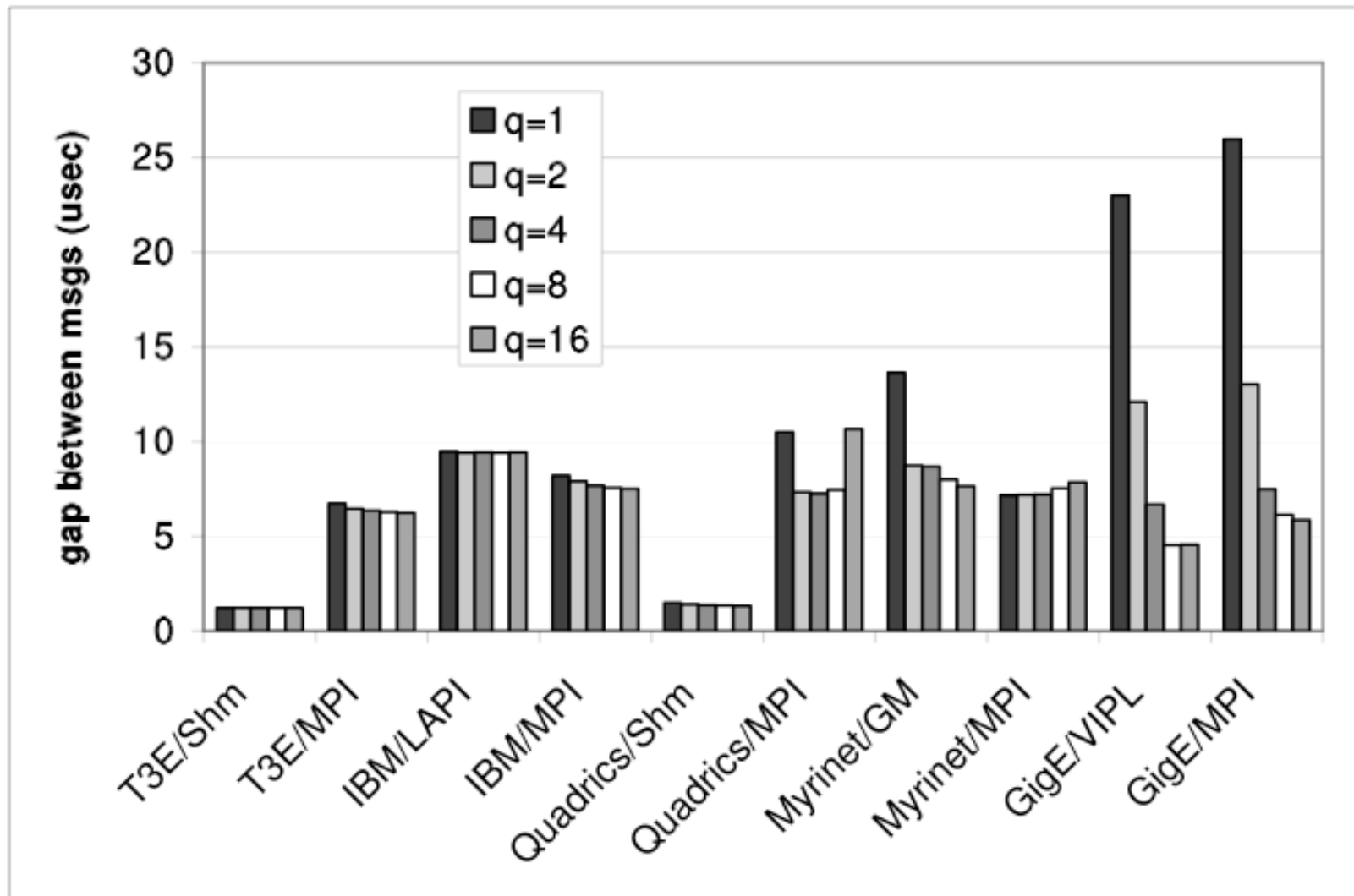
Bell et al. *An evaluation of current high-performance networks*,
IPDPS, Nice, France, April 2003.

Crossover from g to G



Bell et al. *An evaluation of current high-performance networks*,
IPDPS, Nice, France, April 2003.

Potential for Overlapping Communication



Bell et al. *An evaluation of current high-performance networks*,
IPDPS, Nice, France, April 2003.

Footnote to the Future: PRAM is Alive and Well!

The PRAM is noted as an enabling technology for software and algorithms for FY 2010-2014

—an unlikely turn for a model previously considered unrealistic!

**Summary Report on the Workshop on The Roadmap for the Revitalization of High-End Computing,
Washington, D.C., June 2003,
Dan Reed, editor. (Table 2.1, page 19.)**

http://www.nitrd.gov/hecrtf-outreach/20040112_cra_hecrtf_report.pdf

References

- D.E. Culler, R.M. Karp, D. Patterson, A. Sahay, E.E. Santos, K. E. Schauser, R. Subramonian, T. von Eicken. *LogP: A practical model of parallel computation*. Communications of the ACM, 39(11):78 - 85, November 1996.
- A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Scheiman. *LogGP: Incorporating long messages into the LogP model*. JPDC, 44(1):71–79, 1997.
- P.B. Gibbons, Y. Matias, V. Ramachandran. *The Queue-Read Queue-Write PRAM Model: Accounting for Contention in Parallel Algorithms*. SIAM J. Computing 28(2):733-769, 1998.
- Bell et al. *An evaluation of current high-performance networks*, IPDPS, Nice, France, April 2003.