Massively Parallel Processors: Observations and a Realistic Theoretical Model (LogP)

David A. Patterson
Computer Science Division/EECS Dept.
University of California, Berkeley

Outline

Part I: Predictions for Massively Parallel Processors (MPP)

- MPP three parts: Processor, Memory, Network
- Review 3 technologies past & future
- Common MPP architecture
- Prediction on possible surprise for MPP

Part II: More Realistic model of MPP

- PRAM model too inaccurate
- Need for a new model
- Describe LogP
- Example of LogP

Massively Par. Processor Background

- Performance, Performance, Performance

  Otherwise, use uniprocessor

- MPP argument is better cost-performance than vector supercomputer

- MPP Node => Good cost-performance
  Otherwise, 1000 × bad cost-performance => very bad cost-performance

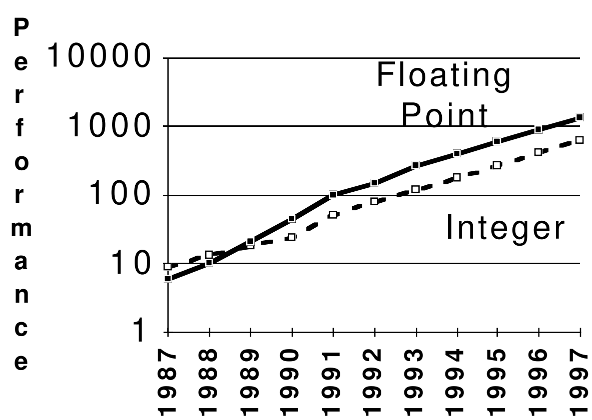
- Good Performance via Bandwidth

- Good Cost via high volume components

Hence must predict underlying technologies to predict future MPPs

Part 1: Processor Performance

- Performance Trend: Grows integer 50%/year, 75% Floating Point per year through 1997; MP after?



- Predict Performance: 1995: ≥ 250 SPECint92, 500 fp92
  2000: ≥1000 SPECint92, 2000 fp92

Part 1:Processor Perspective

- Putting performance growth in perspective:

	HP 9000/735 Workstation	Cray 1 Vector Supercomputer
Year	1992	1976
MIPS	>130 MIPS	< 10 MIPS
Linpack	41 MFLOPS	27 MFLOPS
Cost	$40,000	$4,000,000 ($8M in 1992$)
Clock	99 MHz	80 MHz
Cache	128 KB	0.25 KB
Memory	400 MB	32 MB

1976 supercomputer in 1993 workstation!

Predict: 1993 vector supercomputer in 2000 workstation!

Part 1:Processor Perspective

- Putting performance growth in perspective:

	IBM POWER2 Workstation	Cray YMP Vector Supercomputer
Year	1993	1988
MIPS	> 200 MIPS	< 50 MIPS
Linpack	140 MFLOPS	160 MFLOPS
Cost	$120,000	$1,000,000 ($1.5M in 1992$)
Clock	71.5 MHz	167 MHz
Cache	256 KB	0.25 KB
Memory	512 MB	256 MB

1982 supercomputer in 1993 workstation!

Predict: 1993 vector supercomputer in 2000 workstation!

Part 1: Processors

- Development Cost High for Microprocessor

  MIPS R4000: $30 M develop, $10 M fabricate

  (30 engineers × 3 years)

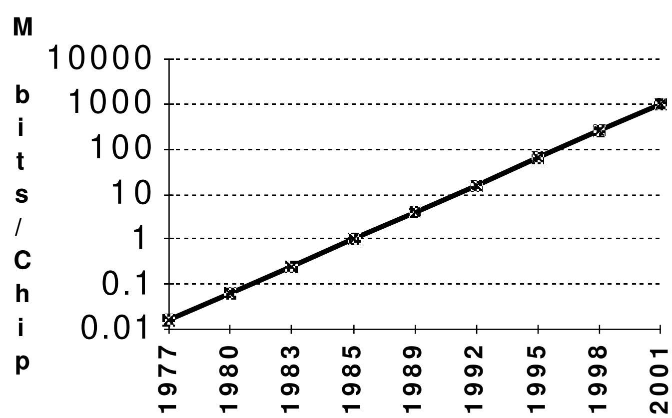
- Add to difficulty/cost of building MPP if design own processor vs. better match to MPP

- Prediction:

  Lose if try to match performance of new chips: must compete with engineering at HP, IBM, DEC, Sun, MIPS, Intel in processor design

Part 2: DRAM capacity growth

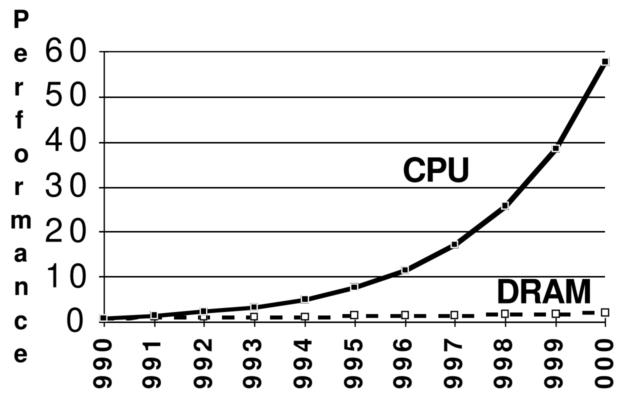
- Trend: 4X every 3 years:



- Feb '93 @ ISSCC: 3 companies unveil 256 Mbits

Part 2: Memory

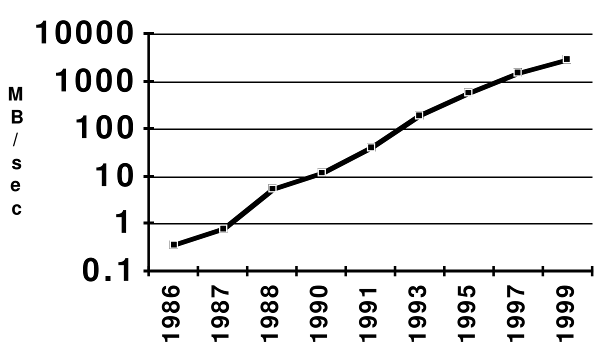
- Latency & Bandwidth
- Trend: Incredible processor progress creates processor-memory performance gap:



- Prediction: By 2000 DRAM latency 2X improved; CPU 50X; => memory hierarchy increases in importance

Part 3: Network Bandwidth

- Improvements in link bandwidth over time:higher clock rate, multiple bits per link



- Predict: continued improvement in bandwidth
  1995: 0.5-1.5 GB/sec; 2000: 2-4 GB/sec

Part 3: Network Overhead

- Overhead launch a message on network in clock cycles

	Machine	Overhead
Commercial Machines	iPSC 860	1500
	Ncube	1000
	CM-5	132
Research Machines	Dash	40
	J machine	40
	Monsoon	10

Part 3: Future Network overhead

- **Trend:**
  Network overhead decreasing,
  but processors getting faster too

- **Prediction:**
  Network overhead 1995 to 2000:
  no less than 20 to 40 instructions; maybe 100s?
  Non-local much slower than local access

MPP Software: Topology Insensitive

- Many good topologies to chose from

  (today grids and fat-trees popular)

- Trend: High overhead to send message makes network "flat": same latency near or far

- Standard topology will never happen

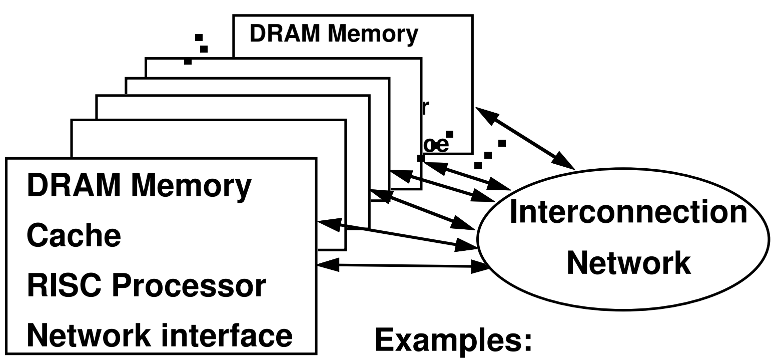
⇒ if rewrite MPP software for different topologies

⇒ MPP Software never portable

⇒ MPP industry fails

- Prediction: MPP software must be independent of network topology

MPP Hardware Organizations have converged



Examples:

Cray T3D

IBM SP-1

Intel Paragon

TMC CM-5…

MPP Systems: Number of Processors

- Trend: Node looks like a workstation
- Workstations cost \$5,000 to \$10,000

  Supercomputers cost \$25,000,000

  ⇒ ≤ 2500 to 5000 nodes

  Parallel processors priced higher, doesn't include network

Predict: Big Iron in 1995: ≤ 2,000 processors

(No commercial firm shipping general purpose SIMD in 1995-96)

- Workstations in 2000 have 4 processors?

Predict: Big Iron in 2000: ≤ 10,000 processors

Implication: Problem size >> number processors

MPP Systems: Off-the-Shelf vs. Custom

- Memory:

⇒ Off-the-Shelf trend: Everyone uses DRAM
Custom trend: none

Predict 1995 to 2000: Off-the-Shelf DRAM

- Processors:

⇒ Off-the-Shelf trend: Cray T3D, Intel, CM-5
Custom trend: nCube, KSR

Predict 1995 to 2000: Off-the-Shelf

- Network (MPP edge): Off-the-Shelf trend: none

⇒ Custom trend: Everyone

Predict 1995 to 2000: Custom (at least interface)

MPP Systems: Addressing / Memory

• "Shared Memory vs. Distributed Memory #1":

Single Address Space vs. Multiple Address Spaces

⇒ Single Address Space trend: KSR, Sequent,Cray

Multiple Address Spaces trend: Intel, CM-5, nCube, ...

Predict 1995 to 2000:??? (Single Address Space?)

• "Shared Memory vs. Distributed Memory #2":

Central Memory vs. Distributed Physical Memory

Central Memory trend: Sequent

⇒ Distributed Physical Memory trend: Cray, Intel, CM-5, nCube, KSR ...

Predict 1995 to 2000: Distributed Physical Memory

MPP Systems: Who moves Data?

- *Software* Responsible (Cray, Intel, TMC, ...)

  Software makes copies in memory

- *Cache* Controller Responsible (Sequent)

  Caches kept consistent via invalidation; reads cause copies in caches

- *Main Memory* Controller Responsible (KSR)

  Memories kept consistent via invalidation; reads cause copies in *memories* (can be cached)

- Drawbacks to Memory Controller:

  move data to query vs. query to data

  may want to redistribute data & nail it down

Predict 1995 to 2000: HW trap invokes Software to move (incoherent global address)

MPP Summary: 1995 to 2000

- MPP Node ≈ Off-the-shelf Processor/Memory (≈ workstation: in 2000 ≥ 1993 supercomputer)
- SIMD moves back to the history pages
- Locality of access critical for speed in MPP
- Distributed memory (shared address helps evolve sequential codes to parallel)
- Custom network (better have) much lower latency and much higher bandwidth than LAN

Assessment of MPPs (negatives)

- Lags workstation technology
  - 2-3 years behind on processor technology
    (e.g, 32MHz SPARC in CM-5 in 1992)
  - 5 years behind on systems and software
    (e.g., paging in CM-5 in 199?)
- The Cost Factor
  - CM-5 node+disk vs. SS10+disk: \$40k vs. \$12k
  - T3D node vs. Alpha (150 MHz): \$60k vs. \$10k
  - high cost of development, high cost of sales

=> Not fully exploiting commercial technology

What if you provided?

Worst possible environment:

- **Traditional message passing on top of your, already slow, TCP/IP** (5 millisecond overhead)
- **High Latency, Low Bandwidth, Unreliable network** (Ethernet)
- **No debugging software, programming environment, or support.**

And distribute for free on workstations/LAN?

PVM

PVM: SW runs parallel processors over network

Result:
9000 FTPs of PVM from ORNL since 1990
+ sales and support by Cray, Intel, IBM, SGI
+ 1992 Gordon Bell Prize

=> More people using PVM than all MPPs combined!

How can this weak system be so successful?

=> Must leverage workstation cost / reliability and offer portability

MPP Surprise: 1995 to 2000

- NOW (Network of Workstations & central switch): workstations by day, supercomputer by night

The building is the computer!

- MPP vs. NOW may be battleground for massive parallelism:

Node cost:	advantage NOW
Network performance:	advantage MPP
3rd Party software:	advantage NOW

- Massive parallelism in 2000 *may* be software on clusters of workstations

Massively Parallel Processor Conclusion

- MPPs leverage advances in DRAM and Processors; MPP node organization stable for ≥ 5 years
- MPPs better add custom networks with much lower latency and much higher bandwidth than LAN & compete in cost-performance
- By 2000: ≤ 10,000 1993 supercomputers working together; peak performance of 20 TFLOPS
- NOWs will challenge traditional MPPs

Theoretical models & MPP: Are they accurate

- Common Theoretical Fallacies with MPP
- Grading PRAM on MPP issues: 2/8
- Grading Valiant's BSP on MPP issues: 3.5/8
- Towards A New Model for MPP (LogP)

Theory Fallacies about MPP

- **Number of processors ≈ ∞**

  Cost of node (≈ workstation) suggests 100 ≤ P ≤ 1000; N >> P
- **No locality in data accesses**

  Every computer system today relies on locality: processor cache, page size, virtual memory, disk cache, file read ahead, ...
- **Simply scale problem to overcome difficulty**

  Scaling may make problem take weeks or years to run
- **Paper or research machine = machines shipped**

  e.g., Combining networks, optical crossbar switches may never be used by someone trying to solve a problem. Don't justify simplifying assumptions with machines no one uses.

Grading PRAM on MPP issues

PRAM

0 ***Speed compared to best alg. on uniproc.*** **papers use scalability as measure**

1 ***Number of PEs:*** **PRAM => P**

0 ***Latency of communication:*** **PRAM => 1**

0 ***Gap between messages:*** **PRAM => 1**

0 ***Fraction accesses not local:*** **PRAM => 100%**

0 ***Size of local memory:*** **PRAM => ∞ (1 clock global)**

0 ***Algorithm/Layout of Data to PE:*** **PRAM irrelevant**

1 ***Number of steps in Algorithm:*** **PRAM counts**

Total: 2/8

Quotes on PRAM

"The literature on PRAM algorithms has been almost completely irrelevant to our research - even the simplest PRAM algorithms described in the theory literature have constant factors that make them prohibitive when a real implementation is considered."

Richard Anderson, University of Washington

Quotes on PRAM

Valiant's Algorithm for finding max on n elements with n=p processors; running time is O(log log n)

Divided into stages, each stage has input values partitioned into sets of equal size

For each set processors compare pairs and record in Boolean array

But unit time access model is wrong; it will take at least Ω(log n) time on real machine.

Hence O(log log log n) on real machine; but could take only O(log n) if support binary tree in network.

"This is a serious matter. ... [PRAM] has been harm rather than helpful ...[since it] led the programmer to the wrong choice."

Larry Snyder, University of Washington

"Type Architectures, Shared Memory, and the Corollary of Modest Potential", *Ann. Rev. Comp. Sci.*, 1986, p.298-300.

Grading BSP on MPP issues

0 *Speed compared to best alg. on uniproc.* BSP use scalability as measure: 2 to 1,000,000

1 *Number of PEs:* BSP => P

1 *Latency of communication:* PRAM => L

0.5 *Gap between messages:* PRAM => g (but assumes g is a small constant that can be ignored)

0 *Fraction accesses not local:* BSP => 100%

0 *Size of local memory:* BSP => ∞ (1 clock global)

– 1 *Algorithm/Layout of Data to PE:* BSP is random! (Hash on every memory access!!)

1 *Number of steps in Algorithm:* BSP counts

Total: 3.5/8

Also expects high network bandwidth, free context switch

John von Neumann, 'The Mathematician', 1947

"As a mathematical discipline travels far from its empirical source, or still more, if it is a second and third generation only indirectly inspired from ideas coming from 'reality', it is beset with very grave dangers … that the subject will develop along the line of least resistance, that the stream, so far from its source, will separate into a multitude of insignificant branches… ."

"In other words, at a great distance from its empirical source, or after much 'abstract' inbreeding, a mathematical subject is in danger of degeneration. …whenever this stage is reached, the only remedy seems to me to be the rejuvenating return to the source: the reinjection of more or less directly empirical ideas. I am convinced that this is a necessary condition to conserve the freshness and the vitality of the subject, and that this will remain so in the future."

A New Model for the 1990s

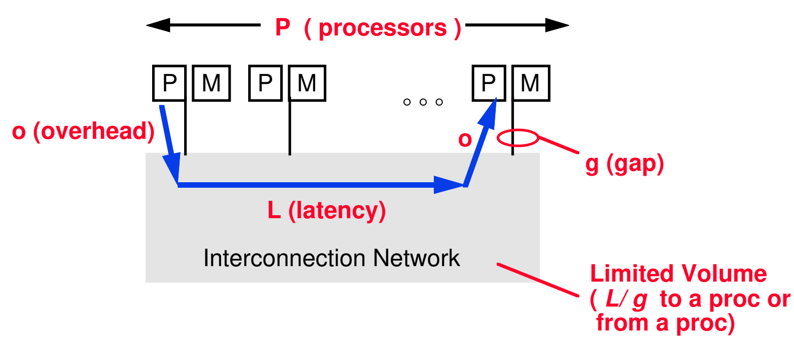
- Goal: Useful for theoretical investigations and predicts performance on real MPPs
- investigating with Prof. David Culler, Prof. Richard Karp, Abhijit Sahay, Prof. Klaus Schauser (UCSB), Eunice Santos, Ramesh Subramonian, and Prof. Thorsten von Eicken (Cornell)

(see "LogP: Towards a Realistic Model of Parallel Computation," *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, San Diego, CA May 1993)

Deriving the Model

- Processing
  - – powerful microprocessor, large DRAM, cache => **P**
- Communication
  - \+ significant latency (10's of cycles) => **L**
  - \+ limited bandwidth (1 – 5% of memory bw) => **g**
  - \+ significant overhead (10's – 100's of cycles) => **o**
    - \- on both ends
  - – no consensus on topology
    - => should not exploit structure
  - \+ limited capacity
  - – no consensus on programming model
    - => should not enforce one

The LogP Model



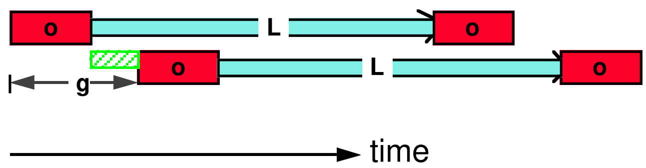
L: Latency incurred in sending message between modules

o: processor overhead to send *or receive* a message

g: Gap between messages sent or received by processor

P: number of processors/memory modules

Illustration of Parameters

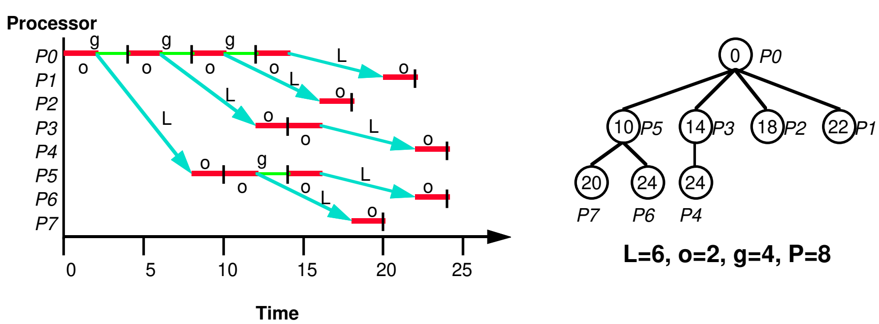


- Send N messages from proc to proc in time
  2o + L + g(N-1)
  - each processor does *oN* cycles of overhead
  - has *(o-g)(N-1) + L* available compute cycles
- Send N messages from *one to many* in same time
- Send N messages from *many to one* in same time
  - all but *L/g* processors block

Theory Sampler

Optimal single item broadcast is an unbalanced tree

- **shape determined by relative values of *L, o,* and *g.***

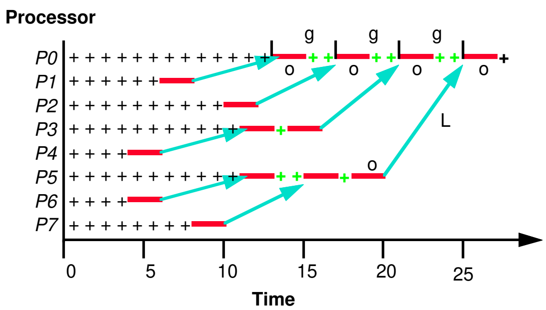


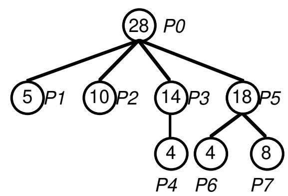
L=6, o=2, g=4, P=8

Optimal Summation

Yields a related tree where each processor finishes local work just as partial sum arrives

- **initial data placement determined by *L, o,* and *g.***
- **communication schedule is key**





L=5, o=2, g=4, P=8

For more see Karp, Sahay, Santos and Schauser, SPAA 93

Simplifying LogP

- **PRAM has only one parameter (P) of LogP: when can you reduce the number of parameters?**
- **Many machines today o = g; future expect o to be small due to DMA, so it may be neglected for some machines => LgP**
- **Many algorithms have long messages that can be pipelined through the network hiding latency L => gP**
- **Ideally, first show can ignore some parameters and then use the remaining parameters**

LogP for real machines

Machine/Net	SW	L	o	g	P
			(µseconds)		
HP 720/Ethernet	PVM	50	2400	80*P	4 to 16
HP 720/Ethernet	UDP	50	360	20*P	4 to 16
HP 720/FDDI	---	4	400	2*P	8 to 32
HP 720/FDDI	A.M.*	4	8	2*P	8 to 32
DEC 5000/ATM	---	20	20	20	8 to 32
Intel Paragon	NX	1	250	2	up to 500
TMC CM-5	CMMD	4	45	4	up to 1024
TMC CM-5	A.M.*	4	2	4	up to 1024

* Active Messages: see T. von Eicken, D. E. Culler, S. Goldstein and K. Schauser,"Active Messages: a Mechanism for Integrated Communication and Computation,"Proc. of the 19th Annual International Symposium on Computer Architecture, Gold Coast,

Advantages of LogP

Discourages common loopholes:

- excessive communication
- excessive memory module contention
- excessive multi-threading to hide latency

Rewards proven techniques:

- coordinating work assignment with data placement to reduce communication
- overlapping computation with communication (within limits)
- balancing communication

Programming model independent

Why LogP?

- **But LogP is harder than PRAM; why worry?**
- **Unfortunately PRAM doesn't work**

"When theory becomes inbred – when it has grown several generations away from its roots, until it has lost touch with the real world– it degenerates and becomes sterile. ...The best theory is inspired by practice and the best practice is inspired by theory." **Donald Knuth,** **'Theory and Practice', 1991**

Who cares about Theory?

- **Given Zadeh's Law:**

"Friends come and go; enemies accumulate."

why antagonize/care about theory?

- **Strength of CS&E in past has been faculty in theory & systems who talk and influence each other**
- **We are at a crossroads; theory and practice hand-in-hand, or separate paths like Math and Engineering?**
- **PRAM or models like LogP?**

Conclusion

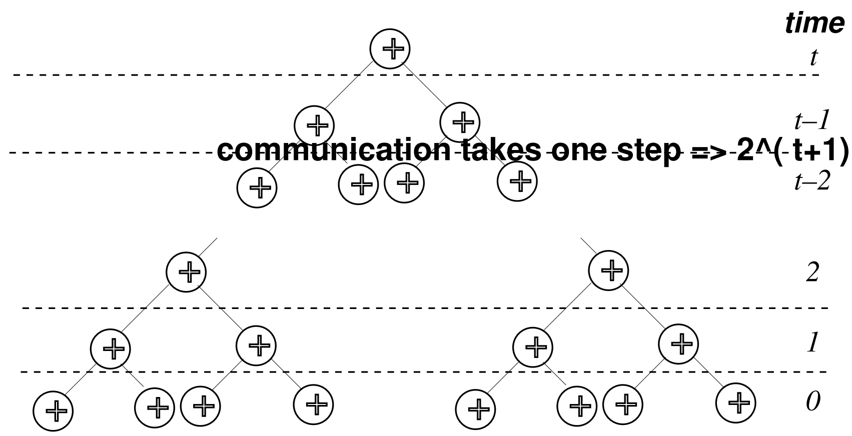
- Hardware and Software MPP models are converging; theory is diverging
- Opportunity for you to contribute provided you are using the right assumptions
- Finding algorithms for important applications that are "embarrassingly parallel" (low local memory miss rate) results in CSE Olympic Medal
- Candidate CSE Olympians need to know architecture, algorithms, and applications
- For theory to have impact, need a model that leads to faster programs
- LogP provides a theoretical base ignoring topology: Latency (L), overhead(o), bandwidth gap (g), and processor-memory nodes (P)

Extras

(The slides following this one are considered backup slides, and may be used to answer questions.)

An example: Sum of Numbers

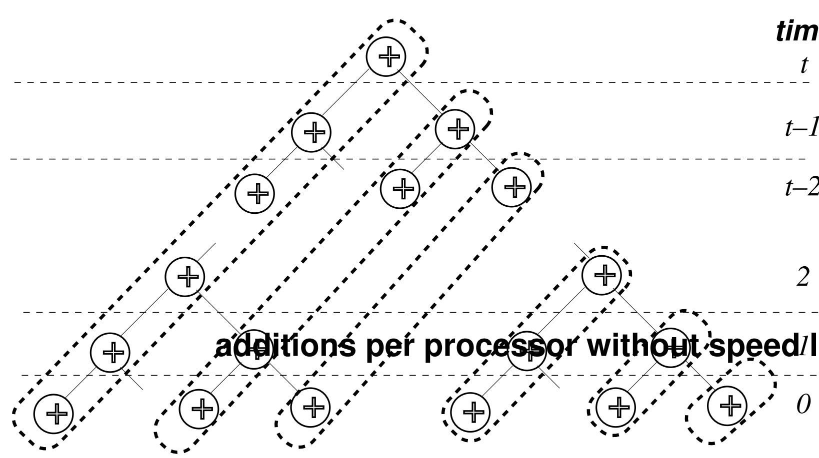
- How many numbers can be added in time t?
- Summing numbers normally done as binary tree with one number per processor in PRAM since assume



communication takes one step => 2^(t+1) num

Sum of Numbers for PRAM

- To avoid wasting processors, you can assign multiple



additions per processor without speed/loss in

Sum of Numbers: considering latency

- If Latency L is considered, solution is no longer optimal; its better to add numbers locally while waiting for sums created in parallel to be sent
- Solution is specified top down via recursive function

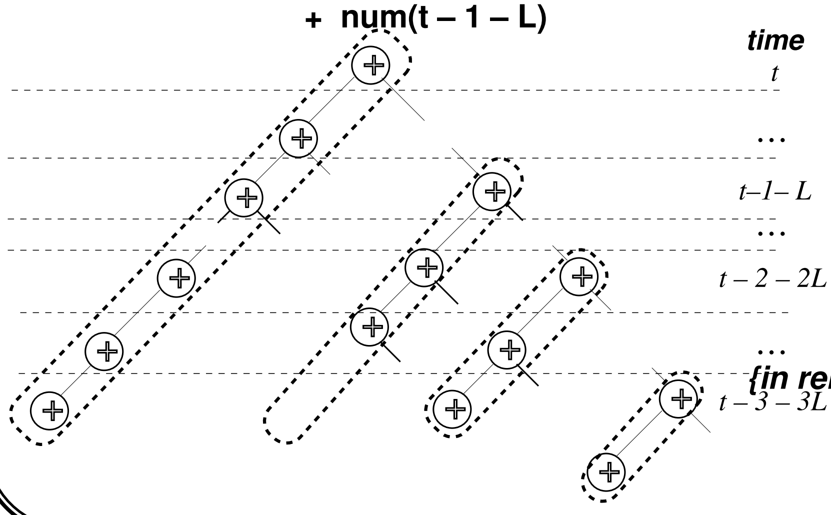
num(T): $t \leq L$ add t numbers *{locally in single proc.}*

$t > L$ num(t – 1) *{in local processor}*
+ num(t – 1 – L) *{in remote processor}*

L for network to deliver sum from other processor

Sum of Numbers: considering L

num(T): $t \leq L$ add t numbers *{locally in single proc.}*
$t > L$ num(t – 1) *{in local processor}*
+ num(t – 1 – L)



Sum of Numbers: considering gap & L

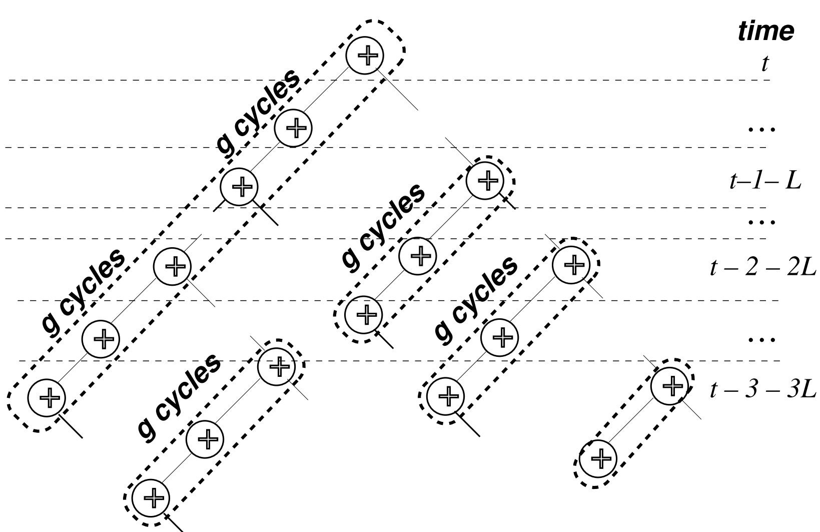
- If gap g is considered, solution is no longer optimal since can only receive every g cycles.
- Solution is to sum g numbers locally while waiting for another processor

num(T): $t \leq L$ add t numbers *{locally in single proc.}*

$t > L$ num(t – 1) *{in local processor}*
+ num(t – 1 – L – g)*{in remote processor}*

L for network to deliver sum from other processor,
g for gap between messages on same processor

Sum of Numbers: considering g & L



Sum of Numbers: considering overhead, g & L

- If overhead o is considered, must account for time when sending since you can't compute while sending or receiving; useful only if sum of more than o numbers
- Solution is to sum g numbers locally while waiting for another processor

num(T): $t \leq L + 2o$
add t numbers *{locally in single proc.}*

$t > L + 2o$
num(t – 1 – o) *{in local processor}*
\+ num(t – 1 – L – g – 2o)*{in remote processor}*

o for other processor to send sum, L for network to deliver it, g for gap between messages, o local processor to receive sum

Theory Reaction to Criticisms

- PRAM is is good for negative proofs

  Reply: Agree (if that's a useful contribution)
- PRAM is good first step

  Reply: Demonstrate by using doing second step; eventually a program runs faster or it doesn't
- Feeling is that 8MB per node $\approx \infty$, so just drop parameter (or don't worry, people will do the right thing)

  A model is like a contract, and that you must avoid loopholes, otherwise you get perverted results (e.g., every node a copy of all data to minimize communication)

LogP Variations

- **Variation 1: Broadcast Network**

  LogP model + Network can broadcast simultaneously to all processors in time it takes to send a single message to one processor (2*o*+*L*).

- **Variation 2: Barrier Synchronization**

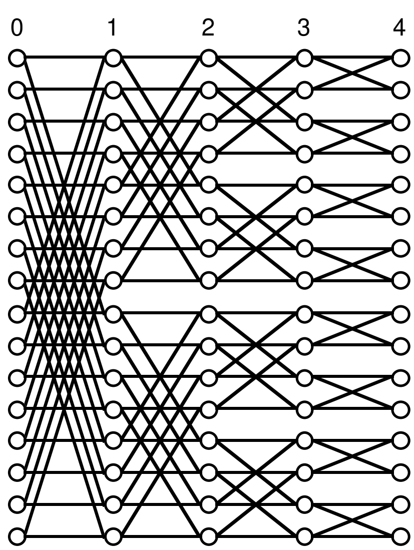
  LogP model + Network can all synchronize in time it takes to send a single message to one processor (2*o*+ *L*).

- **Variation 3: Local Cluster Organization**

  LogP model + P processors consists of P/C clusters; each processor has unlimited bandwidth to cluster memory (e.g., multiple Processors per local memory)

Idea: first proofs using LogP; proofs using variations help architects evaluate new features

FFT



Butterfly:

– n rows, log n columns (here n = 16)

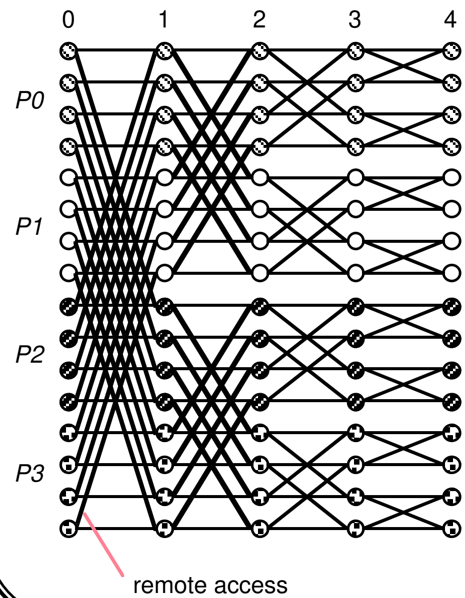
– elegant element-wise structure

– remember n >> P

Computation time:

n/P (log n)

Blocked FFT



Data placement:

blocked layout: - row i gets mapped to processor (i div P)

Computation time:

n/P (log n)

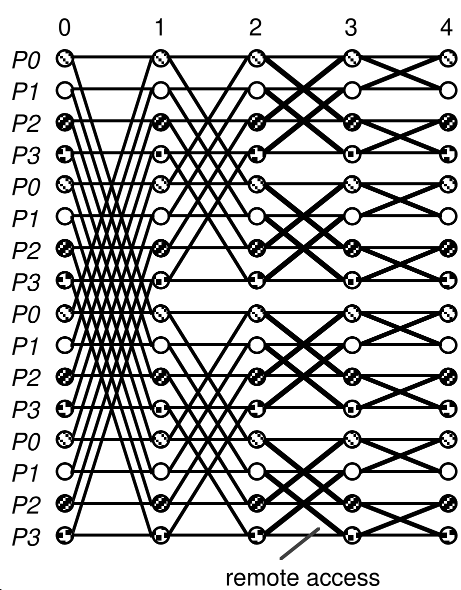
Communication time:

(g n/P + L) * log P

– first log P stages remote

– last log n/P stages local

Cyclic FFT



Data placement:

cyclic layout: - row i gets mapped to processor (i mod P)

Computation time:

n/P (log n)

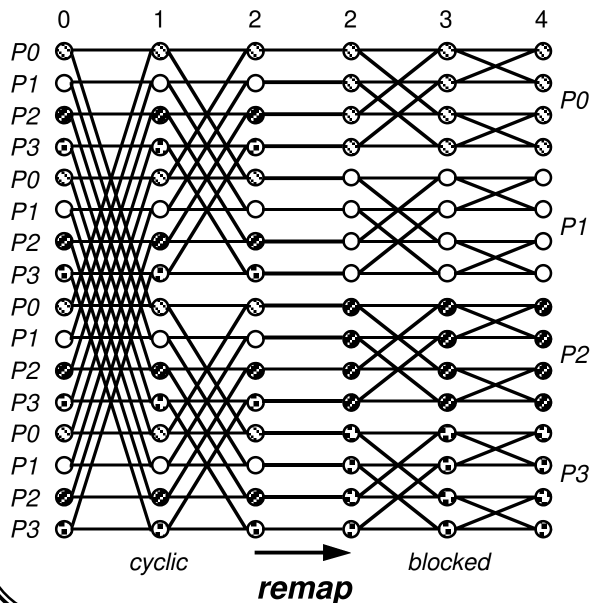
Communication time:

(g n/P + L) * log P

– first log n/P stages local

– last log P stages remote

Hybrid Layout



hybrid layout:

- first log n/P columns are cyclic,
- the last log P are blocked,
- only one communication step for $n \geq P^2$

Computation time:

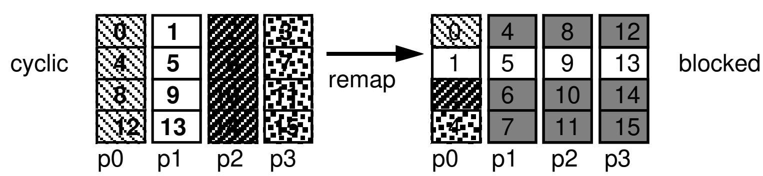
n/P (log n)

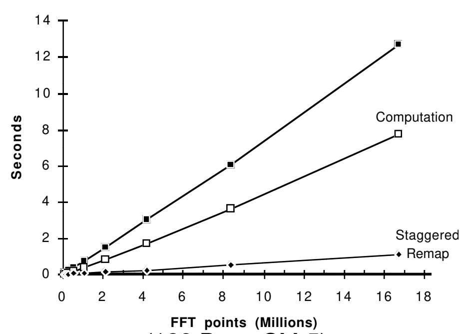
Communication time:

$(g\ (n/P - n/P^2) + L)$

Balanced Communication

With many elements per processor there is a single, general communication step — cyclic to blocked remap





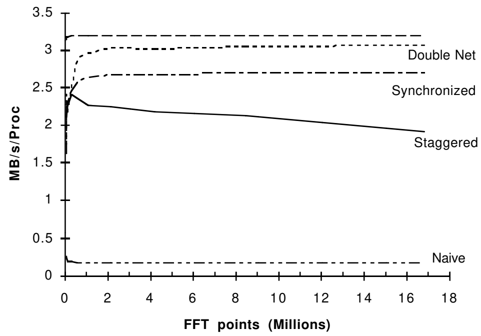
(128 Proc. CM-5)

— all start with first row
=> high contention
communication bound

— processor ***i*** starts with row ***i n/P***
=> no contention
compute bound

Communication Performance

Communication rate per processor for remap



Model parameters:

L = 6 us
o = 2.3 us
measured unloaded
g = 4 us
bisection

Predicted communication time:

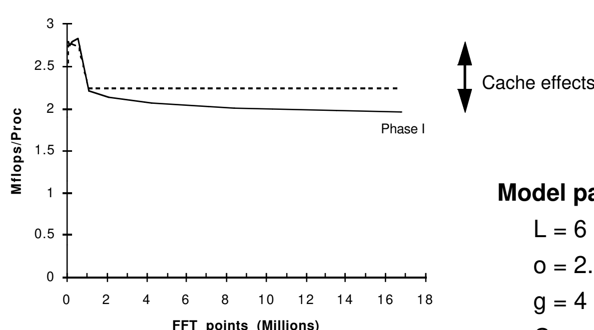
n/p max(2o,g) + L

Asynchronous execution of staggered phase leads to some contention.

Using both data networks increases available bandwidth (reduces g).

Computation Performance

FFT implementation on a 128 Processor CM-5



Computation rate per processor for the two computation phases
(CM-5 Sparc node 3.2 MFLOPS on Linpack)

Model parameters:

L = 6 us

o = 2.3 us

g = 4 us (bisection)

Computation time per Butterfly node = 4.5 us

Predicted computation time:

4.5 n/p log n

Other Studies

- Linear Algebra
- Sorting
- Stencil Operations
- Connected Components

Potential concerns

Is LogP a good abstraction of networks?

\+ All current parallel machines experience a processor overhead in message handling of 10's to 100's of cycles.
- nCUBE, CM-5, Delta, BBN, Dash, J-machine, Monsoon, . . .
- software overhead can make it look like 1000's
- programming model may prevent overlap

\+ With wormhole routing the unloaded latency is:

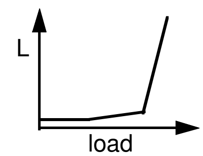
$$\frac{\text{Message Size}}{\text{Link Width}} + \text{Hops} * \text{Router Delay}$$

\+ Latency increases rapidly in saturation
- limited capacity

– Hops influences bandwidth (g)

P=1024

Network	Ave. Hops
hypercube	5
butterfly	10
4° Fat Tree	9.3
3D Torus	7.5
2D Mesh	21



Do we really need 4 parameters?

° In many cases it is safe to ignore some of them

Ignore *L:*
- many independent operations: $T = n^* \max(2o,g) + L$
- long messages

Combine o and g:
- $o > g$
- forego overlap

Ignore *g*:
- chain of dependent messages

° If we drop any of them we open up loopholes
- ideally *o* would disappear as architectures advance

Other models

Model	primitives	sync	network	lat.	ovhd	bw	contention
PRAM [FW78]	CRCW,CREW,EREW	sync	memory	1	-	∞	-
MPC [MV84]	EREW	sync	memory	1	-	∞	module
Phase PRAM [G89]	EREW	async	memory	1	-	∞	-
Delay model [PY88]	CREW	sync	memory	L	-	∞	-
BPRAM [ACS89]	CREW (block tranfers)	sync	memory	(block transfer time)			-
LPRAM [ACS90]	CREW	sync	memory	g	-	g	-
Scan model [B87]	EREW, scans	sync	memory	1	-	∞	-
Network models	msgs	sy/as	explicit	hops	-	links	module
CTA [S86]	msgs	async	explicit	hops	-	g	module
BSP [V90]	msgs, sync	async	h-relation	L	-	g	module
LogP [93]	**msgs**	**async**	**fully conn.**	**L**	**o**	**g**	**module**
PMH [ACFS93]	block transfers	async	tree	(block transfer time)			module

all models except for PMH are based on RAM model, and thus do not model memory hierarchy

Summary

° A realistic model for parallel computation is needed to develop efficient portable algorithms.

° Convergence of MPP hardware organization makes this possible.

° It must strike a balance between detail and simplicity to reveal important bottlenecks, yet keep analysis tractable

° LogP models communication latency, overhead, and bandwidth as simple parameters (L,o,g).

° Allows algorithms that *adapt* to these parameters.

° Leads to new algorithms

° Examples achieve high performance on CM-5

Work Ahead

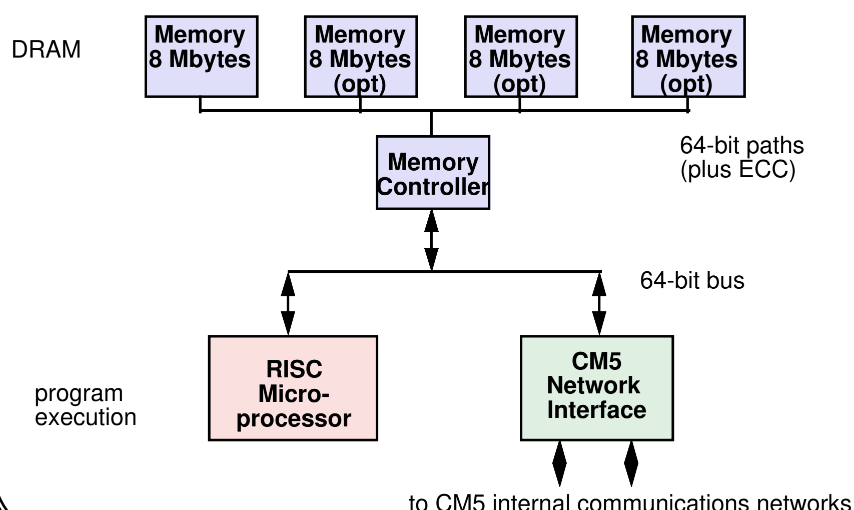
Calibration of machines

machine	L(P)	o	g(P)
CM-5	$100+50 \log_4 P$	70	130
DASH	20	15	60
.			
.			
.			

Analysis of algorithms

Validation of the model

Base CM5 Processing Node



Data Net Implementation: 16 Leaves (3D View)