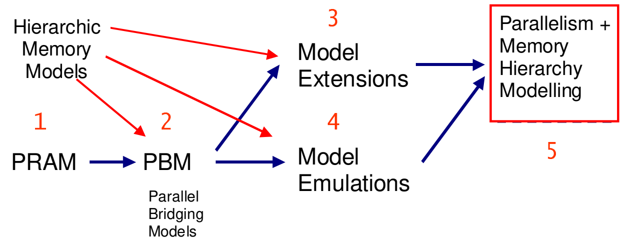
(Schloss Dagstuhl – GI Seminar on Memory Hierarchies – 14/03)

Parallel and Hierarchical Computational Models

Evolution from the PRAM to more realistic parallel models, and its relationship with hierarchical memory models.

Massimo Coppola, C.S. Dept., University of Pisa 17/05/2002

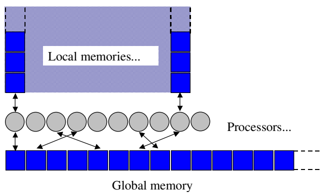
Overview



Universal Parallel Computational Model : is the PRAM not enough?

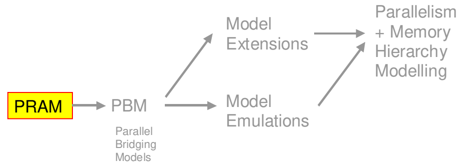
Relationships : Parallel Bridging Models and Hierarchical Memory Models

The PRAM



- Shared-memory + local memory
- Synchronous execution
- Unit-time, uniform memory access
- Unlimited resources (processors, memory)

The PRAM is no realistic model



Basic assumptions are too strong for real architectures

- Synchronous and UMA
- Unlimited communication bandwidth
- Unlimited processing power

the PRAM captures abstract *complexity*, not performance

Sequential Models

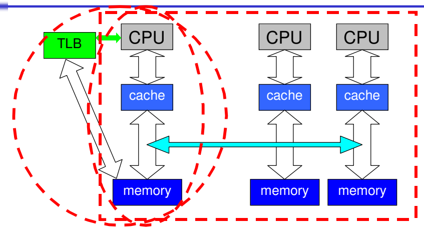
Physical memory hierarchies lead to (large) overheads when accessing blocks of data

High-detail models
- sequential processor architecture :
  - – Cache Memory Model, Internal M. Model
- deeply tied to CPU/memory hardware
- often network-specific

Hierarchical Memory models [VS94]

PDM - parallel disk model
- P parallel processors and D disks
- aims at I/O overhead modeling ($10^5 – 10^7$)
- focus is on (parallel) disc access
- loose hypotesis on processor interconnection

High-detail Memory Models



Cache Memory Model

Cache : size and number of cache lines, associativity

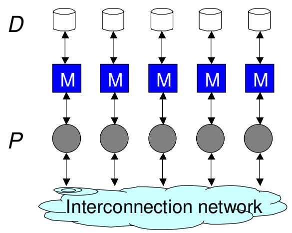
Internal Memory Model

TLB modeling (TLB *and* cache misses)

Parallel case

Coherency overhead (true and false sharing)

Parallel Disk Model



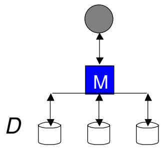
P processors
D disks
M local memory size
B size of disk blocks
N,Z input and output size
n=N/B m=M/B

I/O (block) operations
(incorporate locality into the algorithm)

Network is not specified

Many results for EM fundamental data str.
(graphs, lists, arrays)

most common case: P=1



PDM results [Vit01]

l.b. of fundamental operations
- scan $\Theta(N/DB) = \Theta(n/D)$
- search $\Theta(\log_{DB} N)$
- output $\Theta(\lceil Z/DB \rceil) = \Theta(\lceil z/D \rceil)$
- sort

$$\Theta\left(\frac{N}{DB}\log_{M/B}\frac{N}{B}\right) = \Theta\left(\frac{n}{D}\log_m n\right)$$

general lower bound

permutation == sorting

special cases:
- matrix transposition
- BPC permutations

Optimal algorithms are known
(m.way merge, radix, column sort)

Most results : p=D=1

Sort is everywhere...

Common solution if D>1
- *striping*
- (search suboptimal)

Few true parallel results
- (D>1, p>1)
- often randomized

Communication cost?

Parallel Models and Detail Tradeoff

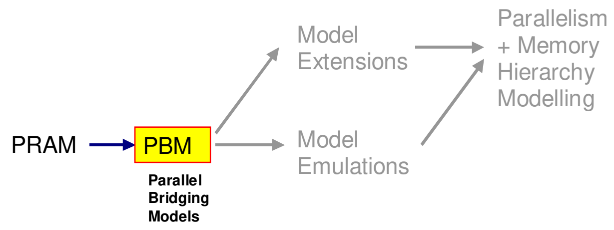
- Network geometrical structure
- Network physical properties & performance
- Asynchronous communications
- Problem decomposition
- Concurrent behaviour
- All of the details about cache hierarchies?

But we need a unifying model

- Universal (performance) results
- Portable algorithms

9

Parallel Bridging Models



Parallel Bridging Models should act as a standard separating algorithm design from architecture

10

Features of Bridging Models

- Cost measure for performance prediction
- Independent from architecture/technology
- Reflect fundamental constraints
- *Efficient universality*
  (avoid logaritmic simulation losses)
- Neutral w.r.t. number of processors
- Portability of programs
- Better *programming* models

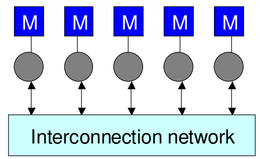
Parallel slackness

Virtual parallelism ***v*** is greater than actual parallelism ***p***

11

The BSP model [Val90]

p processors with local memory
complete connection
abstract network specification
computation proceeds by supersteps



12

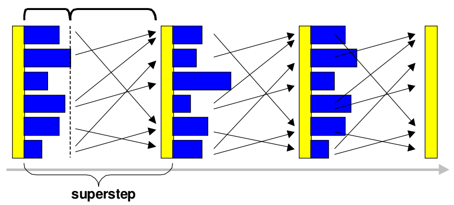
BSP superstep

L (latency) g (gap) p (parallelism)

superstep t cost : $w_t + g\,h_t + L$

local work + h-relation routing + synchronization

w_t $g\,h_t$ L



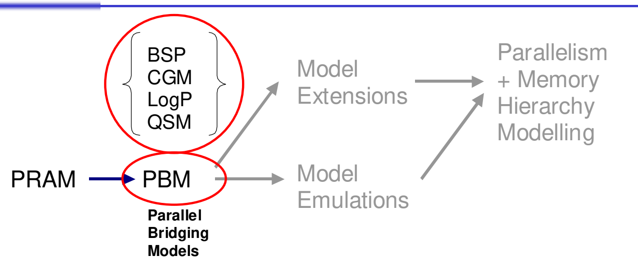
13

Other Bridging Models

- CGM coarse grain multicomputer
  alternate phases, local computation and global routing
  routing cost depends on network geometry
  decompose work to minimize the number of phases
- QSM queued shared memory
  uses supersteps, memory cost still unrealistic
  easy conversion of PRAM results
  flat model
- LogP named from its parameters
  detailed communication model
  asynchronous model, algorithm analysis is complex

14

PBM cross-emulation [Ram01]



Slowdown of work-preserving emulations

Emulated Model ↓	BSP	LogP	QSM
BSP		*polylog* (p)	g/L log p
LogP	L/l		g/l log p
QSM	L/g + g log(p)	*polylog* (p)	

15

Parallelism and Memory Hierarchies

There are two hierarchies, but

we cannot separate them

SM overheads : cache coherency, thread synchronization
DM overheads : physical delays, process scheduling, system calls
DM communications and local I/O are not strictly ordered

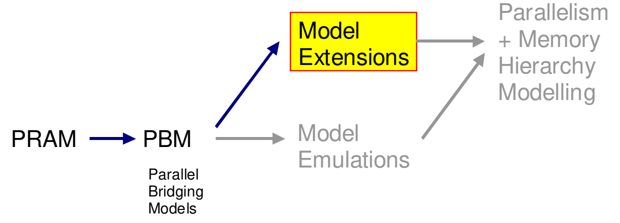
Layer	Bandwidth	Latency
CPU	10-100 Gb/s	1-10 ns
Main Memory	>1000 Mb/s	10-100 ns
SM comm.s	100-1000 Mb/s	10-1000 ns
DM comm.s	100 Mb/s	0.001-1 ms
Local I/O	10-100 Mb/s	5 ms

?

these are orders of magnitude

16

BSP shortcomings



BSP disregards

Message size (size of h-relations)

Network hierarchical structure (network locality effects)

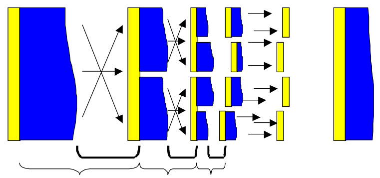
Decomposable BSP [TK96]

Parallel locality network physical structure

Recursive sub-machine splitting

l,g $\Rightarrow$ L(m) and G(m)

Smaller BSPs exploit better communication parameters



D-BSP generality [BFPP01]

functions $g_i^{(\alpha)} = (p/2^i)^\alpha \quad l_i^{(\beta)} = (p/2^i)^\beta$

clusters of moderate bandwidth/latency, like multidim. arrays

- slowdown results (D-BSP vs proc. networks)
- arrays (at most constant loss)
- D-BSP handles unbalanced comm. (E-BSP)
- shared memory emulation (w.h.p.)
  p memory accesses $O(p^\alpha + p^\beta)$

several problems are faster to solve (broadcast)

BSP simulates D-BSP with non constant time loss

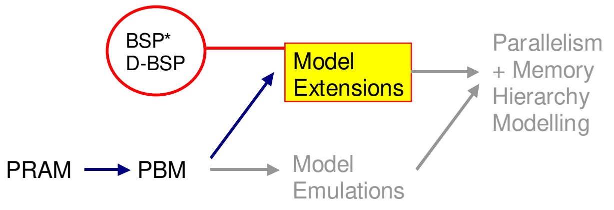
BSP * [BDMR96]

practical communication bandwidth
is a function of message size
g does not reflect communication startup

B = critical block size
s_t = communication volume
superstep cost : $w_t + g^* (s_t + B\, h_t) + L$

messages larger than *B* are rewarded

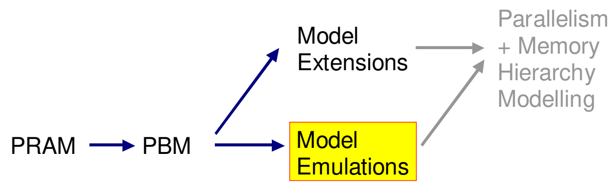
Model Extensions



Extensions exploit concepts common to parallel and HMM

- Blocking (communications)
- (Hierarchical) Hardware structure

Model Emulations



Emulations of parallel algorithms over EM models

Why emulate PBM algorithms?

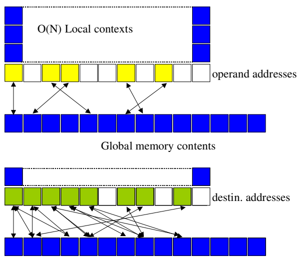
- there are a lot of BSP results
- model structures have a clear mapping

PRAM Emulation in Ext. Memory [CGG+95]

(simple case) O(N) space

O(N) task list and memory list

1. sort tasks by operand address
2. merge in operands (2 scans)
3. sort by destination address
4. update memory (2 scans)



Each parallel step is O(1) sort/scan operations

I/O complexity : *T sort(N)*

Asymptotically optimal

Large constant hidden

The bottleneck is I/O sorting bandwidth

some changes to deal with

- space > O(N)
- geometrically decreasing P

BSP-like to External Memory [SK97]

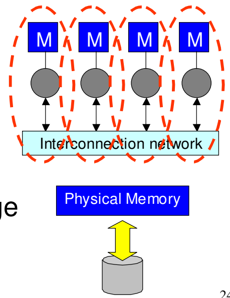
Sequential execution of BSP in EM

BSP-like : subclass of BSP

local memory < N/P mess. size p.p. <m'/P

m' =M/3 P=N/m' =3N/M

1. for each superstep
2. for each processor
3. load processor image
4. local computation
5. schedule messages in a matrix



(sequential) M memory, 3N storage

3M/B I/O per superstep

BSP-like to External Memory

BSP cost parameters of emulation

- P = 3 N/M
- G = CPU execution to I/O ratio
- L = M G / 3

EM optimality for emulated BSP

x-optimal if $T_{ext} < x\, T_{seq}$

Results

- some EM optimal algorithms are found by emulation of known BSP algorithms (matrix multiplication, sorting)

But

- subclass of BSP, ad-hoc parameters and bounds
- good EM algorithms are missed for geometrically decreasing size problems (like List Ranking !)

25

BSP Emulation in Parallel EM

[DDH97]
[DDHM99]
[DDHM-]

Parallel + Hierarchical Memory model

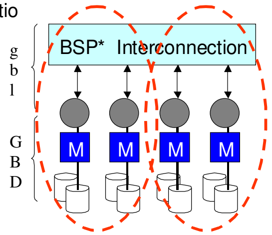
BSP*	g, b, l
EM-BSP*	g, b, l, G, B, D, M

G	CPU execution to I/O ratio
B	disk block size
D	number of disks
M	memory size

messages go to the disks

split into packets

standard consecutive format



26

Parallel BSP Emulation in EM

Superstep simulation

1. read in a certain amount of processor contexts
2. read the packets they have received
3. simulate local computation
4. write the packet sent to the D disks (randomized!)
5. write the changed contexts
6. rearrange the message blocks for each processor group

steps 1-4 exploit the data format

step 5 exploits BSP* blocking [DDH97]
distribute packets into buckets on disk

step 6 routing simulation in EM

CGM (det) and BSP (block) emulation into BSP*

27

EM-BSP*, EM-CGM Results

extended c-optimality criterion

asymptotical c-optimality of computation time
o(1) asymptotical ratio of comm. and I/O to computation

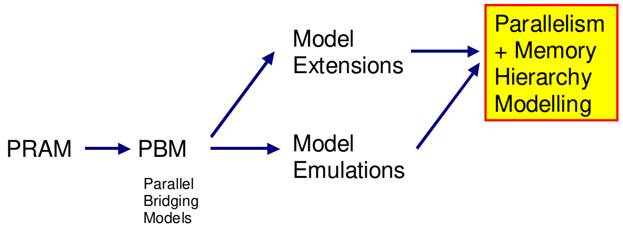
simulate v-processor BSP* algorithms
on p < v processors in EM-BSP*

(w.h.p.) condition for preserving c-optimality
involving slackness (v/p)

(appropriate parameters : message size, v/p)
optimal or improved EM algorithms for several problems

28

Combining the Models



Two results that combine the approaches

- Sorting on cube-like machines
- Distribution sweeping on clusters

Parallel Sorting [AP94]

Hypercube-like architectures

bounded degree : butterfly,CCC,shuffle exchange

Set of primitives working on *d*-subcubes

- assignment O(1)
- BPC route O(d)
- merge O(d)
- monotone r. O(d)
- prefix O(d)
- row route O(d lg d)

Recursive model of hierarchical storage

units within a group connected by bounded-degree sorting networks

Distribution Sweeping on Clusters [DMPP02]

D-BSP + local disks

D-BSP* p, g(i), l(i), G, B, M

G CPU execution to I/O ratio

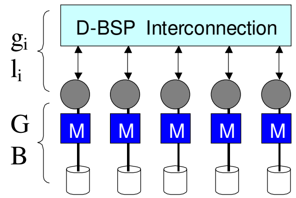
B disk block size

M memory size

$M / B = \Omega(p^{\varepsilon})$

superstep cost (size i)

$w_1 + w_2 G + h g_i + l_i$



Algorithmic primitives on D-BSP*

Assuming that $g_i^{(\alpha)} = (p/2^i)^{\alpha}$ $l_i^{(\beta)} = (p/2^i)^{\beta}$ $0 < \alpha, \beta < 1$

Sorting

$$T_{sort}(n,p) = O\left(\frac{n}{p}(\log(n/p) + p^{\alpha}) + p^{\beta} + \frac{n}{Bp}\left\lceil \frac{\log(n/B)}{\log(M/B)} \right\rceil G\right)$$

Composition of local (EM-aware) and parallel (D-BSP) sorting algorithms

1. constant number of rounds
2. each p. locally sorting O(n/p) keys, exchanges O(n/p)
3. locally employ (M/B)–way merge sort

Claim of optimality, by bandwidth cons. and EM sort lower bound

Distribution Sweeping

Minimize I/O and communication cost

partition input into K=min{n/p, M/B} subsets (vertical slabs)

if K<p assign each subproblem to a (log K)-cluster (p/K)
recursive apply

else assign p/K subproblems per processor

New: "orthogonal" partitioning
of input and results from recursive calls into
subsets of size Θ(K) (horizontal slabs)
all interactions are computable indepently (in parallel)

Choice of K implies in-core processing and load balancing

33

Distribution Sweeping

Results shown

Measure of the union of rectangles

$$T_{MUR}(n,p) = T_{sort}(n,p) = O\left(\frac{n}{p}(\log(n/p) + p^{\alpha}) + p^{\beta} + \frac{n}{Bp}\left\lceil\frac{\log(n/B)}{\log(M/B)}\right\rceil G\right)$$

A lot of details ...

34

Summary

- Common hardware issues
  bandwidth constraints and latencies
- "sequential" PDM approach
  focus on I/O and block locality, scan & sort oriented
- CGM / BSP approach
  parallel slackness, superstep
  bulk parallelism ≈ "parallel locality"
- (sequential hierarchic models)
- Parallel hierachic models (D-BSP, Y-PRAM ...)
- External-memory emulation results
  (sequential and parallel)

35

Conclusions

- Mixing bulk-parallel and EM techniques (emulations and new models)
  ⇒ new and improved complexity results
- New models employ standard EM and bulk parallel results as building blocks
- Large, complex problems need both techniques anyway
  (Data Mining, Computational Chemistry and Biology, Simulation...)
- What kind of hardware?

36

Hierarchic Parallel Applications

- Structured networks (e.g. Grid computing)
- SMP clusters
- PIM
  - internal memory small
  - internal bandwidth 100 Gb/s *per processor* !!
  - off-chip links and networks cannot match
- active disks
  - simpler, SIMD-style computation
  - on-disk special primitives (sort, search)

37

Main References

[ADJ+01] V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, editors. Computational Science - ICCS 2001, LNCS 2074.

[AP94] Alok Aggarwal and C. Greg Plaxton. Optimal parallel sorting in multi-level storage. In *Proc. Fifth Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 659--668, 1994.

[BDMR96] A. Bäumker, W. Dittrich, F. Meyer auf der Heide, and I. Rieping. Realistic Parallel Algorithms: Priority Queue Operations and Selection for the BSP* Model. In LNCS 1124, pages 369--376.

[BFMR96] L. Bougè, P. Fraigniaud, A. Mignotte, and Y. Roberts, editors. *Euro-Par '96 Parallel Processing, vol. II*, LNCS 1124

[BFPP01] G. Bilardi, C. Fantozzi, A. Pietracaprina, and G. Pucci. On the effectiveness of D-BSP as a bridging model of parallel computation. In LNCS 2074, pages 579--588.

[CGG+95] Yi-Jen Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and Jeffrey S. Vitter. External-memory graph algorithms. In *Proc. of the 6th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 139--149, 1995.

[CKP+96] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: A practical model of parallel computation. *Communications of ACM*, 39(11):78--85, 1996.

[DDH97] F. Dehne, W. Dittrich, and D. Hutchinson. Efficient external memory algorithms by simulating coarse-grained parallel algorithms. In *SPAA '97*, pages 106--115, 1997.

[DDHM] F. Dehne, W. Dittrich, D. Hutchinson, and A. Maheshwari. Bulk synchronous parallel algorithms for the external memory model. *Theory of Computing Systems*. To appear.

[DDHM99] F. Dehne, W. Dittrich, D. Hutchinson, and A. Maheshwari. Reducing I/O complexity by simulating coarse grained parallel algorithms. In *Proc. 13th International Parallel Processing Symposium (IPPS'99)*, pages 14--20, 1999.

[DFRC93] F. Dehne, A. Fabri, and A. Rau-Chaplin. Scalable parallel computational geometry for coarse-grained multicomputers. In *Proc. of the 9th Symp. on Computational Geometry*, pages 298--307. ACM, 1993.

38

Main References

[DMPP02] F. Dehne, S. Mardegan, A. Pietracaprina, and G. Prencipe. Distribution sweeping on clustered machines with hierarchical memories. In *Proc. of 2002 International Parallel and Distributed Processing Symposium (IPDPS 2002)*, April 2002. *To appear.*

[GMR97] P. B. Gibbons, Y. Mathias, and V. Ramachandran. Can shared-memory model serve as a bridging model for parallel computation? In *Proc. of the 9th ACM Symp. on Parallel Algorithms and Architectures*, pages 72--83, 1997.

[Ram01] Vijaya Ramachandran. Parallel algorithm design with coarse-grained synchronization In LNCS 2074, pages 619--627.

[SK97] Jop F. Sybein and M. Kaufmann. Bsp-like external memory computation. In *Proc. 3rd Italian Conference on Algorithms and Complexity*, LNCS 1203, pages 229--240, 1997.

[TK96] P. De La Torre and C. P. Kruskal. Submachine locality in the bulk synchronous setting. In LNCS 1124, pages 352--360.

[Val90] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103--111, August 1990.

[Vit01]J. S. Vitter. External Memory Algorithms and Data Structures: Dealing with MASSIVE DATA. *ACM Computing Surveys*, 33(2):209--271, June 2001.

[VS94] J. S. Vitter and E. A. M. Shriver. Algorithms for Parallel Memory I: Two-level Memories. *Algorithmica*, 12(2-3):110--147, 1994.

39