A Simple and Asymptotically Accurate Model for Parallel Computation

Ananth Grama, Department of Computer Science, Purdue University, W. Lafayette, IN 47906, ayg@cs.purdue.edu

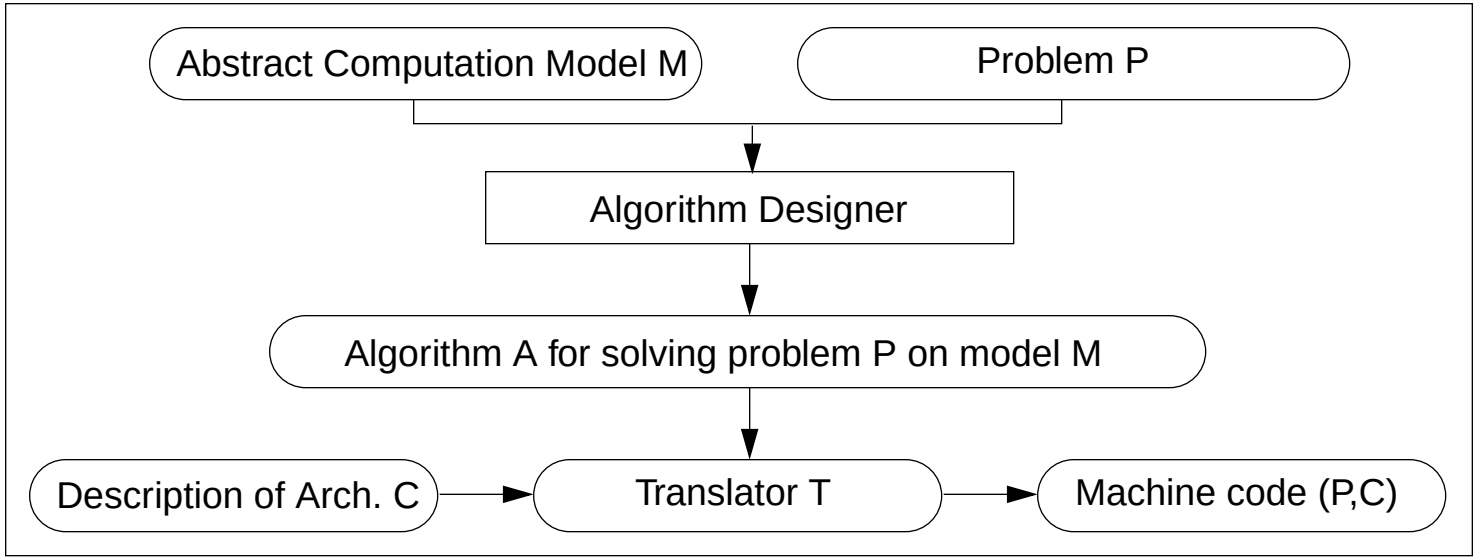
Vipin Kumar, University of Minnesota, Minneapolis, MN 55455, kumar@cs.umn.edu

Sanjay Ranka, Paramark Corp. and University of Florida, Gainesville, FL 32611, ranka@cise.ufl.edu

Vineet Singh 10535 Cordova Road, Cupertino, CA, 95014

(papers relating to this talk are available at http://www.cs.umn.edu/~kumar)

What is an Algorithm Design Model?



❑ Abstract computation models should not be confused with programming models.

❑ von Neumann model provides the abstract computational model for serial computers.

Need for Parallel Algorithm Design Models.

- **Portability:** A program designed for a computer must be portable across a “reasonably” wide range of computers and subsequent generations of the same computer.
- **Shorter design cycles** and **longer software life**.
- **Economics**.

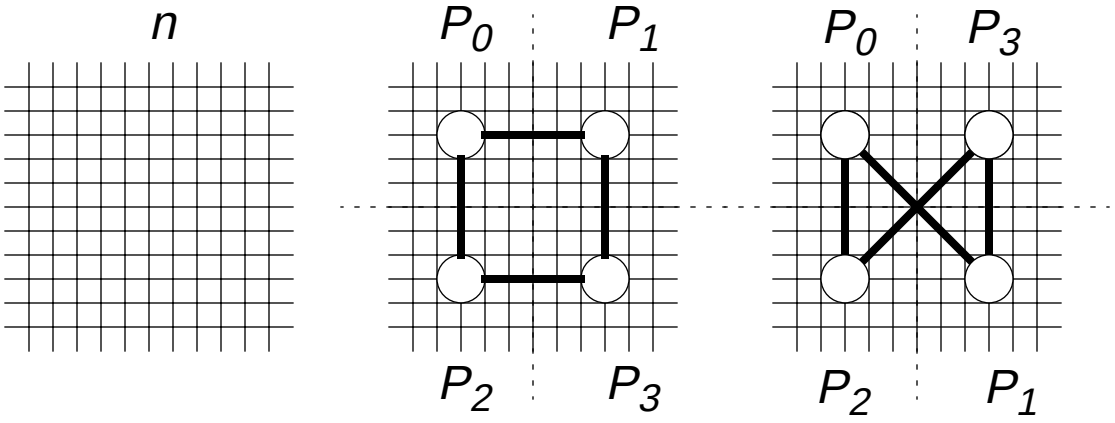
Why are abstract parallel computation models difficult to develop?

- ❏ Processors in a parallel computer must communicate to accomplish assigned task.
- ❏ The communication schedules are often sensitive functions of underlying connectivity between processors.
- ❏ An abstract model for a parallel computer must make some assumptions about the connectivity.
- ❏ A model assuming a low degree of connectivity will yield sub-optimal algorithms for higher degree networks such as hypercubes; and vice-versa.
- ❏ The connectivity (and therefore the communication overhead) is very difficult to capture in abstract models.

Does cut-through routing not solve the problem?

- The time for a point-to-point message in a cut-through routed network is given by: $t = t_s + lt_h + mt_w$

- By increasing the size of the message (m), the bandwidth term (mt_w) can be made to dominate the per-hop term (lt_h). So it does not matter how far a message must travel. But does this solve our problem?



Link Traffic n $2 \times n$

- By making n large, the per-hop term can be masked, but in the second case, the bandwidth term is twice; in general it is $p/2$ times optimal. ***It is all about bandwidth****!!!*

Role of Data Locality

✎ The cost of communicating a m words between processors l hops apart is $t_s + mt_w + lt_h$. For longer messages (or larger latency networks), this time can be adequately modeled as $t_s + mt_w$. This time implies two forms of data locality:

❑ **Bulk Access Locality** (spatial locality): While accessing remote data, it is desirable to access a sequence of words. In the context of uniprocessors, it is referred to as spatial locality.

❑ **Data Reuse Locality** (temporal locality): Minimize remote data accesses, reuse as much data as possible. In the context of uniprocessors, it is referred to as temporal locality.

❑ **Structured Data Access**: Certain data access patterns are more expensive than others. This is what results in architecture dependence of parallel algorithms.

Models for Parallel Computing: Parallel Random Access Machine (PRAM)

- Perhaps the first and best known abstract model for parallel computers.
- Synchronous shared-memory MIMD computer.
- The theoretical model assumes unlimited private and shared memory.
- The model assumes single cycle access to private and shared memory.
- Concurrent access to words in PRAMs are handled differently based on the model (EREW, CREW, CRCW).

Drawbacks of PRAMS:

- ❑ The model is not realizable due to **complexity of the interconnect**.
- ❑ The model **does not account for communication costs**. Therefore, algorithms designed for PRAMs might have very different performance characteristics on real machines.
- ❑ PRAMs **do not have a concept of memory banking**.

Subsequent enhancements of PRAM models (**seclusive PRAM**) assume banked memories.

The **module parallel computer (MPC)** is based on the seclusive PRAM. It assumes that the memory is organized into modules, and access to the modules is exclusive.

Locality Based Models

- **LPRAM (Latency-PRAM)**: Incorporates data volume locality into PRAM cost model. Fetching m words from remote memory takes $t_w m$ cycles (local accesses are assumed to take one cycle).

- **BPRAM (Block-PRAM)**: Incorporates bulk access locality. Fetching m words from remote memory takes $l+m$ cycles (l is the latency).

(neither model addresses blocking of memory)

Completely Connected Network (CCN): Set of processors connected via a completely connected network. Remote access of m words costs time $t_s + t_w m$.

(the model does not account for structured data accesses).

Other models such as Hierarchical PRAM (HPRAM) and YPRAM account for a class of recursively expressible data accesses.

Bulk Synchronous Parallel (BSP) Model

- Consider the following: *p* kids throwing stones randomly into *p* buckets. ***What is the expected value of the maximum number of stones in one bucket?*** Answer: *log p / (log log p).*
- Now instead of one stone, each kid throws *log p* stones into *p* buckets. ***What is the expected value now?*** Answer: *3 x log p.*
- The kids can now be looked at as processors generating memory requests. The buckets can be viewed as the *p* processors' local memories.
- Assume that in one instruction, one processor generates one memory request. If the memory allocation is randomized, the request goes to a random processor. In such a case, *log p* requests can be asymptotically optimally serviced in *3 x log p* time if the network can sustain the traffic.

The BSP model is based on this premise.

Drawbacks of the BSP Model

- Requires ***O(p)*** **cross-section** bandwidth for the network to sustain the traffic (dense networks: crossbars, hypercubes).
- **Large constants** associated with data accesses.
- Since each access is made to a different processor, there is **no bulk locality** in data accesses. This further increases the constant. (A randomized access scheme does not work even on a single processor because of caches).
- The model requires a **slack of *p log p*** (i.e. the total computation must grow at least as *p log p* since each processor must have atleast *log p* computation).
- Since there is no concept of local data, each memory access costs time $t_s+t_w \log p$. To mask network latency t_s, message size must be more than *log p*. The **effective slack required is therefore much higher than *p log p***.

The CGM model

CGM is an extension of the BSP model.

- It addresses the undesirable possibility in BSP of a large number of small messages.
- A CGM algorithm consists of alternating phases of computation and global communication. A single computation-communication pair corresponds to a BSP superstep and the communication phase corresponds to a single h-relation in BSP.
- In the case of CGM, the size of the h-relation is given by $h=n/p$, where n corresponds to the problem size.
- With this restriction on permutation size, the communication cost of a parallel algorithm is simply captured in the number of communication phases.
- This communication cost model, rewards a small number of larger messages, thus capturing spatial locality better than the BSP model.

The BSP* Model

- BSP* also attempts to increase the granularity of messages by adding a parameter B, which is the minimum message size for the message to be bandwidth bound as opposed to latency bound.
- Messages of size less than B are assigned the same cost as messages of size B, thus rewarding higher message granularity.
- The time taken by a BSP* algorithm is determined by the sum of computation and communication supersteps. A computation superstep with n local computations at each processor is assigned a cost max{L, n}.
- A communication step with a h-relation of size s is assigned a cost max{g x h x ceil(s/B), L}.
- Thus algorithms aggregate messages into larger h-relations to utilize all available bandwidth.

LogP Model of Computation

- *LogP* assumes an **ensemble of processors with local memory connected over a completely connected network** (whose aggregate b/w can be scaled).

- Communication in *LogP* occurs by **breaking messages into smaller packets**. The time for a packet to be communicated is modeled using

  - **Latency *L***: the time for a single small message to travel from source to destination.

  - **Overhead *o***: overhead paid for copying message into buffers, setting up network routers etc.

  - **Gap *g***: the time that must be allowed between two messages. This is dictated by the network bandwidth.

- LogP allows multiple packets to be pipelined.

Drawbacks of the LogP Model

The logP model allows us to capture bulk and reuse localities. Its communication cost is based on permutations. A permutation of *m* words takes *O(m)* time. This time is scaled by a bandwidth factor to account for other networks.

- The model is still based on ***O(p)* bisection width** networks.
- Does **not capture structured communication patterns**.
- Yields **neither lower, nor upper bounds** on performance (depending on the data access pattern and architecture). This is the biggest drawback of LogP.

The postal model

- This model, which predates the LogP model, is a simplification of the LogP model.
- The postal model assumes a message passing system of p processors with full connectivity (processors can send point-to-point messages to other processors), and simultaneous I/O (a processor can send one message and receive a message from another processor simultaneously).
- As is the case with LogP, messages have a predefined size. Larger messages are composed as aggregates of these smaller messages.
- In a postal model with latency L, a (small) message sent at time t renders the sender busy for the time interval [t, t+1] and the receiver busy for the interval [t+L-1, t+L].
- Notice that this corresponds to the overhead o of the LogP model being 1 and the latency L being identical to L in LogP.

The LogGP Model.

- The LogGP model adds a parameter G, which captures the bandwidth for large messages. G corresponds to the gap per byte (or data item) for large messages.
- In the LogP model, sending a k item message takes time (o + (k - 1) * max{g,o} + L + o) cycles.
- In contrast, sending the message in the LogGP model (assuming the message is large) is given by (o + (k - 1)G + L + o) cycles.
- As before, the sending and receiving processors are busy only during the o cycles of overhead and the rest of the time can be overlapped with useful computation.
- By reducing the overhead associated with large messages, LogGP rewards increased communication granularity.

Hierarchical Models - HPRAM, YPRAM.

- Models based on **recursive decomposition**.
- The computation is assigned to subgroups of processors.
- The cost of **communicating within a subgroup of *p* processors is given by some function *f(p)***. This cost function is more accurate than those of *LogP* and BSP since it induces some locality.
- The cost function **does not cover a large set of data access patterns** that are not based on recursive decomposition.
- The predicted communication cost can be **asymptotically incorrect by a factor equal to the diameter** of the interconnection network.

Contention Modeling Abstractions.

- The Queuing Shared Memory Model (QSM) also follows a bulk synchronous format with phases of shared memory reads, shared memory writes, and local computation.
- The performance model of QSM explicitly accounts for remote memory accesses, contention, and congestion. The primary parameters in the QSM model include the number of processors p and the remote data access rate in terms of local instruction cycles g.
- If in a phase, a parallel algorithm executes m_{op} local memory operations and m_{rw} remote operations, with a contention of k, QSM associates a cost $\max\{m_{op}, g.m_{rw}, k\}$ with it. Minimizing this maximizes local data access and minimizes contention to shared data.
- In addition to these, QSM also has a set of secondary parameters - latency l, barrier time L, message sending overhead o, memory bank contention h_r, and network congestion c.

The C^3 model.

- This model uses communication packet size l, the setup cost for a message s, communication latency h, and bisection width b.
- A parallel algorithm is assumed to proceed in supersteps of computation and communication. The cost of the computation phase of a superstep is determined by the maximum number of data items accessed by any processor (expressed in terms of number of packets).
- The cost of the communication phase is determined by whether the communication is blocking or non-blocking. A blocking communication of m packets is charged a time 2(s+h) + m + h at the sender and s + 2h + m at the receiver.
- A non-blocking communication of m packets is charged a time s + m + h at the sender and m at the receiver.
- Communication congestion is modeled in terms of network congestion and processor congestion. This is determined by the number of processor pairs communicating c, and the average number of packets r being communicated by any pair. The network congestion C_l is given by $C_l = (r\ xc) / b$ and the processor congestion C_p by $C_p = (r \times c \times h)/p$. The overall communication cost is the maximum of sender time, receiver time, processor congestion, and network congestion.

Summary of Existing Models:

- Most models incorporate communication costs for bulk access and data reuse locality.

- None of the models capture data access patterns and therefore are limited in their coverage of architectures or algorithms.

- Most models are either too loose in their bounds; or yield lower / upper bounds based on algorithms and architectures.

Realistic models must therefore incorporate data access patterns in addition to abstract machine features.

Characteristics of Algorithms

Parallel algorithms may be classified into one of two categories:

❑ **Asynchronous Algorithms**: Processors are allowed to communicate asynchronously; independent of each other.

❑ **Synchronous Algorithms**: In this class of algorithms, execution proceeds in interleaved steps of computation and communication. All processors (or groups thereof) participate in these steps.

The design philosophy of synchronous algorithms is consistent with the MPI/PVM philosophy of aggregate communication operations.

Role of Aggregate Communication Operations

Most synchronous algorithms are composed of a small set of aggregate communication operations:

- **Broadcast Operations**: data item needs to go to all processors.
  - **One-to-all**: one processor has data that must be communicated to all others.
  - **All-to-all**: all processors have a piece of data that must be made available to every other processor.
- **Personalized Operations**: data items have unique destinations.
  - **One-to-all**: One processor has *p* data items, each with a destination that must be communicated.
  - **All-to-all**: Each processor has *p* data items, each with a destination that must be communicated.
- **Circular *q*-shift**: Shifting each data item *q* hops defined by some linear ordering of the processors.

Timings for Various Aggregate Communication Operations

Operation	1-D Mesh	2-D Mesh	Log P-D Mesh
Bandwidth Insensitive Operations			
One-to-all Broadcast	$(t_s+t_w m)\log p + t_h(p-1)$	$(t_s+t_w m)\log p + 2t_h(\sqrt{p}-1)$	$(t_s+t_w m)\log p$
One-to-all Broadcast	$2(t_s p + t_w m)$	$2(2t_s\sqrt{p} + t_w m)$	$2(t_s \log p + t_w m)$
All-to-all Broadcast	$(t_s+t_w m)(p-1)$	$2t_s(\sqrt{p}-1)+t_w m(p-1)$	$t_s \log p + t_w m(p-1)$
One-to-all Person.	$(t_s+t_w m)(p-1)$	$2t_s(\sqrt{p}-1)+t_w m(p-1)$	$t_s \log p + t_w m(p-1)$
Bandwidth Sensitive Operations			
All-to-all Pers.	$(t_s+t_w mp/2)(p-1)$	$(2t_s+t_w mp)(\sqrt{p}-1)$	$(t_s+t_w m)(p-1) + (t_h/2)p \log p$
Circular q-shift	$(t_s+t_w m)\left\lfloor \frac{p}{2} \right\rfloor$	$(t_s+t_w m)(2\left\lfloor \frac{p}{2} \right\rfloor + 1)$	$t_s + t_w m + t_h \log p$

A Two-Level Model for Estimating Communication Overheads

The space of aggregate communication operations can be partitioned into two: those that are **sensitive to bisection** width, and **those that are not**.

Bisection Insensitive	Bisection Sensitive
Broadcasts (one-to-all, all-to-all)	Random permutations
	All-to-all Personalized
One-to-all Personalized	
	Circular q-shift
NEWS Shifts	

A NEWS shift for a *k-d* mesh is defined as a communication with *d* nearest neighbors. In 2-D, it refers to the north, east, west, and south neighbors.

Communication costs:

Given an architecture with bisection-width c, link bandwidth t_w, and p processors, the cost is as follows:

If the maximum amount of data entering or leaving any processor during a communication phase is m, then the cost of the operation is:

- For bisection-sensitive operations:

$$t = t_w m(p/c)$$

- For bisection-insensitive operations:

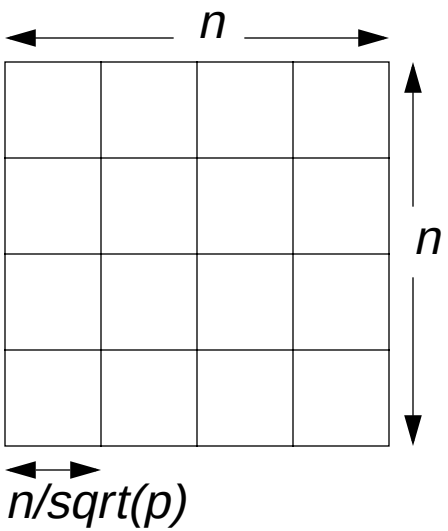
$$t = t_w m$$

Algorithm Design Methodology

- ❏ Determine candidate synchronous algorithm.
- ❏ Determine the aggregate communication operations required.
- ❏ Determine if they are bandwidth sensitive or insensitive.
- ❏ Evaluate communication cost of operation for given architecture.

Case Studies: Dense Matrix Multiplication

Matrix partitioning: 2-D checkerboard partitioning.



Each processor is assigned blocks of $n/\sqrt{p} \times n/\sqrt{p}$. The algorithm proceeds in following steps:

- ❏ All-to-all broadcast of rows of matrix A to all processors in the row.
- ❏ All-to-all broadcast of columns of matrix B to all processors in the column.
- ❏ Compute corresponding sub-blocks of matrix C.

Dense Matrix Multiplication - Analysis:

- Communication cost is due to two all-to-all broadcasts.
- The amount of data involved in each is $n^2/\sqrt{p}$.
- Both operations are bisection-insensitive.

The total communication cost is therefore given by $2*t_w*n^2/\sqrt{p}$ for the applicable class of architectures. The corresponding computation time is $t_c n^3/p$.

	Hypercube	Mesh
LogP	$2t_w n^2/\sqrt{p}$	$2t_w n^2$
A3	$2t_w n^2/\sqrt{p}$	$2t_w n^2/\sqrt{p}$
Accur.	$2(t_s \log p + 2t_w n^2/\sqrt{p})$	$2t_s\sqrt{p} + 2t_w n^2/\sqrt{p}$

Dense Matrix Multiplication: Comparison of Models - Hypercube.

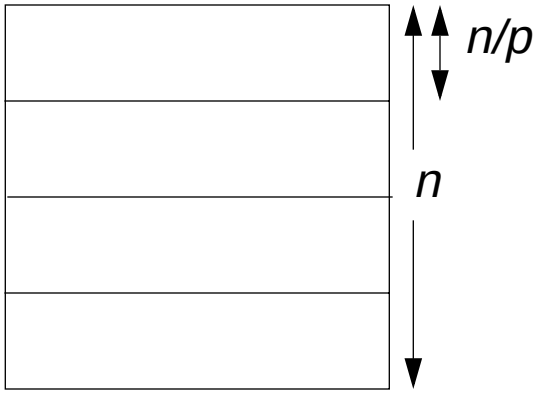
Dense Matrix Multiplication: Comparison of Models - Mesh.

Case Studies: Gaussian Elimination (Dense Matrix Factorization)

❏ Gaussian elimination is used for reducing a matrix into a triangular form.

❏ Pick a pivot; eliminate corresponding variable from all other equations.

❏ Parallel formulations are based on row-wise, blocked or cyclic partitioning.



Here we consider a row-wise partitioning:

❏ Pick pivot.

❏ Broadcast pivot row (*n-i* numbers in the i^{th} iteration) to all processors.

❏ Eliminate variable corresponding to pivot row.

Gaussian Elimination (Dense Matrix Factorization): Performance

- Overhead results from one-to-all broadcasts.
- Algorithm works in n steps; in step *i*, we broadcast *n-i* data.
- Since $n > p$, we can use a **two step broadcast** (one-to-all personalized, followed by an all-to-all broadcast)
- Total communication cost = $2 \sum_{i=1}^{n} (n-i) t_w = t_w n^2$

	Hypercube	Mesh
LogP	$t_w n^2$	$t_w n^2 \sqrt{p}$
A3	$t_w n^2$	$t_w n^2$
Accurate	$2t_s n \log P + t_w n^2$	$4t_s n \sqrt{p} + t_w n^2$

Dense Matrix Factorization: Comparison of Models - Hypercube.

Ananth Grama, Vipin Kumar, Sanjay Ranka, and Vineet Singh

Dense Matrix Factorization: Comparison of Models - Mesh.

Other Tested Algorithms:

- ❑ Fast Fourier Transforms
- ❑ Volume Rendering (shear-warp, volume space partitioning)
- ❑ Sample Sort

Conclusions

- Abstracting machines without application characteristics is very difficult.
- Typical application characteristics allow a very simple classification into bandwidth sensitive and insensitive operations.
- These operations can be assigned different (and simple) costs, and these costs can be used for a simple, asymptotically accurate prediction of runtime complexity.