LogP: Towards a realistic Model of Parallel Computation

Presented by Chris Derrick

Motivation

Current models are over simplistic

This leads to "loop holes" that lets you create algorithms that on paper should perform well, but in practice don't

We need a model that takes into account

- Computing Bandwidth
- Communication Bandwidth
- Communication Delay
- Efficiency from coupling communication with computation

While still remaining simple and easy to work with

PRAM

The most widely used parallel model

Overly simplistic in that it assumes:

- Processors work synchronously
- Interprocessor communication is free

This leads to the creation of overly fine grained algorithms that in practice perform poorly since they spend so much time communicating

LogP Reasoning

LogP was designed with these goals in mind:

- No small set of parameters can describe all machines completely
- A model that contains too many details makes analysis of complicated algorithms overly difficult
- No assumptions about the underlying architecture should be made

LogP Model

The main parameters of the model are:

- L: Upper bound for the latency, or delay, incurred in communicating an empty message
- o: Overhead, time required for transmissions wherein the processor cannot perform other operations
- g: Gap, the minimum time interval between consecutive messages (so at most L/g messages can be in transit from any process or to any process at any time)
- P: Number of processors

The basic model assumes all messages are of a small size

Advantages of LogP

LogP addresses Loop holes found in other models:

- Interprocess communication, unlike PRAM, LogP does not hide the cost
- Multithreading, LogP is able to express the upper bound of this technique (L/g)

Encourages good practices like:

- Coordinating work assignment (so as to reduce the communication bandwith)
- Overlapping the computation and communication

Case Study: Broadcast

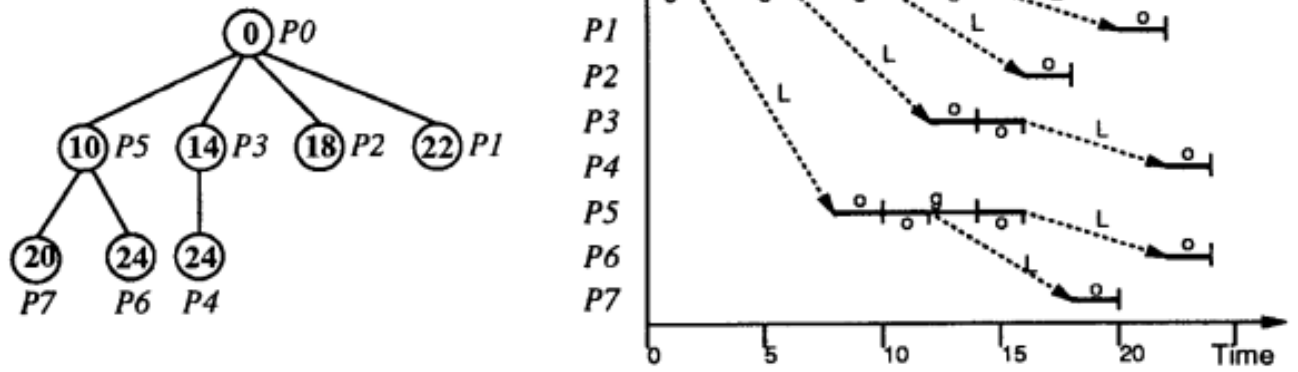

Figure 3: *Optimal broadcast tree for $P = 8, L = 6, g = 4, o = 2$ (left) and the activity of each processor over time (right). The number shown for each node is the time at which it has received the datum and can begin sending it on. The last value is received at time 24.*

With the details provided by a LogP model we see optimal way to structure the tree

Communication Closeup

Four main parts common to all communication:

1. Send Overhead, time spent before the first bit can leave
2. Transition, time needed to get data onto the network
3. Transmission, time required for the bits to travel the network
4. Receive Overhead, time after last bit arrives until the processor is free to work on something else

$T(M, H) = T_{snd} + M/w + Hr + T_{rcv}$

- M = size of message
- w = width of the network channel
- H = distance of the route taken
- r = delay through each node
- Equation assumes lightly loaded networks

Communication Closeup

Machine	Network	Cycle ns	w bits	$T_{snd} + T_{rcv}$ cycles	r cycles	avg. H (1024 Proc.)	T(M=160) (1024 Proc.)
nCUBE/2	Hypercube	25	1	6400	40	5	6760
CM-5	Fattree	25	4	3600	8	9.3	3714
Dash[12]	Torus	30	16	30	2	6.8	53
J-Machine[10]	3d Mesh	31	8	16	2	12.1	60
Monsoon[25]	Butterfly	20	16	10	2	5	30
nCUBE/2 (AM)	Hypercube	25	1	1000	40	5	1360
CM-5 (AM)	Fattree	25	4	132	8	9.3	246

Generating LogP

$T(M, H) = T_{snd} + M/w + Hr + T_{rcv}$

- M = size of message
- w = width of the network channel
- H = distance of the route taken
- r = delay through each node
- Equation assumes lightly loaded networks

$o = (T_{snd} + T_{rcv}) / 2$

$L = Hr + (M/w)$

g = M / (per processor bisection bandwith)

P = Number of physical processors

Alternatives

Extenstions of PRAM:

- Module Parallel Computer assumes that memory is divided into modules and that each can handle one request at once
  - Does not handle bandwidth or network capacity
- PhasePRAM divides computation into "phases" or sections where processors work asynchronously and then synchronize at the end of the phase
  - Still does not fully capture asynchronous possibilities
- And many more their own strengths like
  - The Delay Model
  - BPRAM
  - LPRAM
  - PMH

Inspirational Competition

Valiant's Bulk Synchronous Parallel (BSP) model

- Models distributed-memory multiprocessors in terms of
  - Processor/Memory Modules
  - Interconnection Network
  - Synchronizer which performs barrier synchronizations
- Computations are modeled as a sequence of supersteps
  - Consists of local computation and transfer of messages
  - A processor can send and receive at most h messages
- Weaknesses
  - The largest possible value of h determines the length of each superstep
  - Messages can only be used in the following superstep
  - Assumes special hardware support to synchronize

Network Models

Communication only between directly connected processors

Many algorithms are created for a specific architecture

Predicting their behavior on other architectures is very hard

Since LogP makes no assumptions about the interconnection network it is a great way to design portable algorithms and predict their behavior on different systems