Parallel computing, models and their performances

A high level exploration of the HPC world

George Bosilca
bosilca@eecs.utk.edu
University of Tennessee, Knoxville
Innovative Computer Laboratory

Overview

- Definition of parallel application
- Architectures taxonomy
- Laws managing the parallel domain
- Models in parallel computation
- Examples

Formal definition

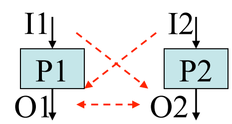
Bernstein

{ I1 ∩ O2 = ∅ and I2 ∩ O1 = ∅ and O1 ∩ O2 = ∅ }

General case: P1… Pn are parallel if and only if
each for each pair Pi, Pj we have Pi || Pj.

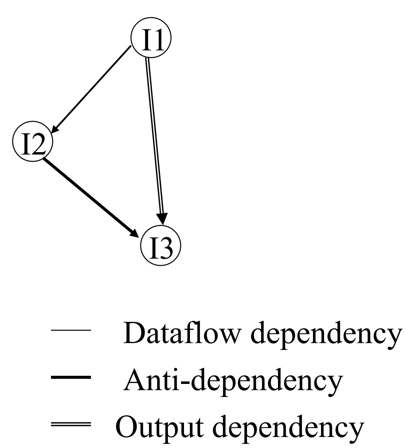
3 limit to the parallel applications:

1. Data dependencies
2. Flow dependencies
3. Resources dependencies



Data dependencies

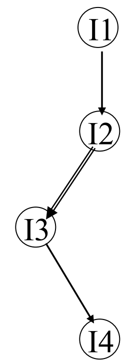
I1: A = B + C
I2: E = D + A
I3: A = F + G



How to avoid them?
Which can be avoided ?

Flow dependencies

I1: A = B + C
I2: if(A) {
I3: D = E + F }
I4: G = D + H



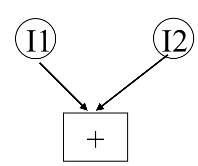
— Dataflow dependency

═ Flow dependency

How to avoid ?

Resources dependencies

I1: A = B + C
I2: G = D + H



How to avoid ?

Flynn Taxonomy

•Computers classified by instruction delivery mechanism and data stream
•4 characters code: 2 for instruction stream and 2 for data stream

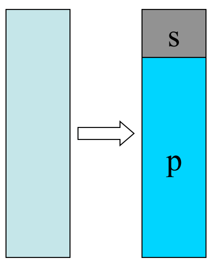
	1 Instruction flow	> 1 Instruction flow
1 data stream	SISD Von Neumann	MISD pipeline
> 1 data stream	SIMD	MIMD

Flynn Taxonomy: Analogy

- SISD: lost people in the desert
- SIMD: rowing
- MISD: pipeline in the car construction chain
- MIMD: airport facility, several desks working at their own pace, synchronizing via a central database.

Amdahl Law

- First law of parallel applications (1967)
- Limit the speedup for all parallel applications



N = number of processors

$$speedup = \frac{s + p}{s + p/N}$$

$$speedup = \frac{1}{a + (1 - a)/N}$$

Amdahl Law

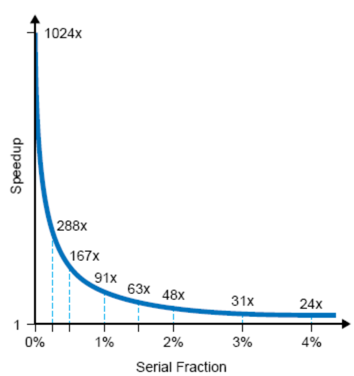


FIGURE 1. Speedup under Amdahl's Law

Speedup is bound by 1/a.

Amdahl Law

- Bad news for parallel applications
- 2 interesting facts:
  - We should limit the sequential part
  - A parallel computer should be a fast sequential computer to be able to resolve the sequential part quickly
- What about increasing the size of the initial problem ?

Gustafson Law

- Less constraints than the Amdahl law.
- In a parallel program the quantity of data to be processed increase, so the sequential part decrease.

$$\left.\begin{array}{l} t = s + P/n \\ P = a * n \end{array}\right\} speedup = \frac{s + a * n}{s + a}$$

$$a \to \infty \Rightarrow speedup \to n$$

Gustafson Law

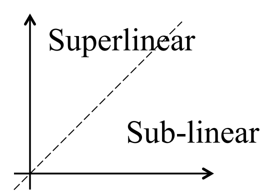
- The limit of Amdahl Law can be transgressed if the quantity of data to be processed increase.

$$speedup \leq n + (1 - n)s$$

Rule stating that if the size of most problems is scaled up sufficiently, then any required efficiency can be achieved on any number of processors.

Speedup

- Superlinear speedup ?



Sometimes superlinear speedups can be observed!
- •Memory/cache effects
- •More processors typically also provide more memory/cache.
- •Total computation time decreases due to more page/cache hits.

•Search anomalies
- •Parallel search algorithms.
- •Decomposition of search range and/or multiple search strategies.
- •One task may be "lucky" to find result early.

Parallel execution models

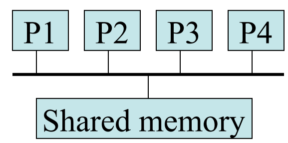
- Amdahl and Gustafson laws define the limits without taking in account the properties of the computer architecture.
- They cannot be used to predict the real performance of any parallel application.
- We should integrate in the same model the architecture of the computer and the architecture of the application.

What are models good for ?

- Abstracting the computer properties
  - Making programming simple
  - Making programs portable ?
- Reflecting essential properties
  - Functionality
  - Costs
- What is the von-Neumann model for parallel architectures ?

Parallel Random Access Machine

- One of the most studied
- World described as a collection of synchronous processors which communicate with a global shared memory unit.



How to represent the architecture

- 2 resources have a major impact on the performances:
  - The couple (processor, memory)
  - The communication network.
- The application should be described using those 2 resources.

$$T_{app} = T_{comp} + T_{comm}$$

Models

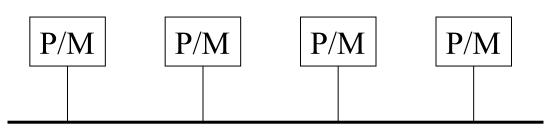
- 2 models are often used.
- They represent the whole system as composed by n identical processors, each of them having his own memory.
- They are interconnected with a predictable network.
- They can realize synchronizations.

Bulk Synchronous Parallel – BSP

- Distributed-memory parallel computer

Valiant 1990

- Global vision as a number of processor/memory pairs interconnected by a communication network

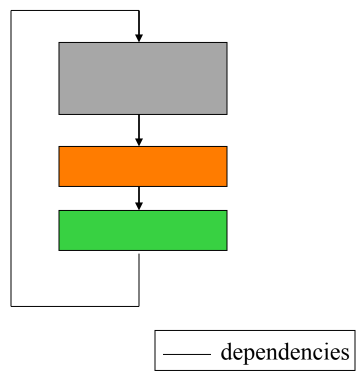


- Each processor can access his own memory without overhead and have a uniform slow access to remote memory

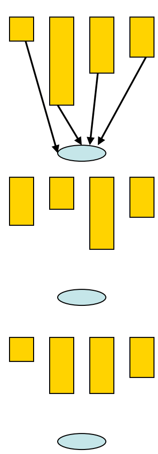
BSP

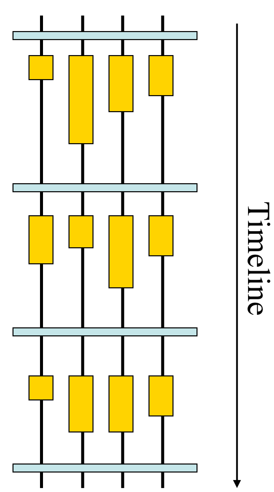
- Applications composed by Supersteps separated by global synchronizations.
- One superstep include:
  - – A computation step
  - – A communication step
  - – A synchronization step

Synchronization used to insure that all processors complete the computation + communication steps in the same amount of time.



BSP





BSP

$$T_{\text{superstep}} = w + g * h + l$$

Where:

w = max of computation time

g = 1/(network bandwidth)

h = max of number of messages

l = time for the synchronization

Sketch the communications

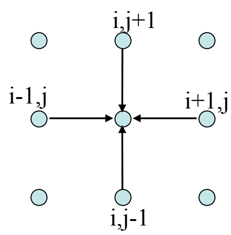
BSP

- An algorithm can be described using only w, h and the *problem size*.
- Collections of algorithms are available depending on the computer characteristics.
  - Small L
  - Small g
- The best algorithm can be selected depending on the computer properties.

BSP - example

- Numerical solution to Laplace's equation

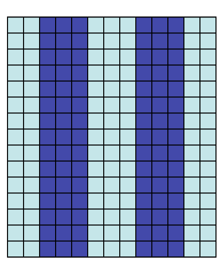
$$U_{i,j}^{n+1} = \frac{1}{4}\left(U_{i-1,j}^{n} + U_{i+1}^{n} + U_{i,j-1}^{n} + U_{i,j+1}^{n}\right)$$




```
for j = 1 to jmax
  for i = 1 to imax
    Unew(i,j) = 0.25 * ( U(i-1,j) + U(i+1,j)
                       +   U(i,j-1) + U(i,j+1))
  end for
end for
```

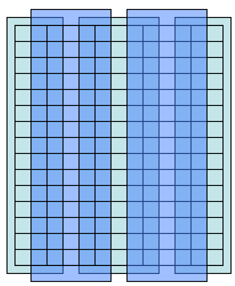
BSP - example

- The approach to make it parallel is by partitioning the data



BSP - example

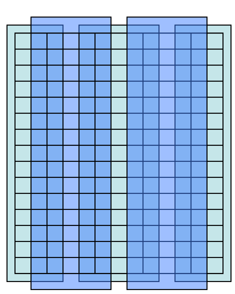
- The approach to make it parallel is by partitioning the data



Overlapping the data boundaries allow computation without communication for each superstep

On the communication step each processor update the corresponding columns on the remote processors.

BSP - example



```
for j = 1 to jmax
  for i = 1 to imax
    unew(i,j) = 0.25 * ( U(i-1,j) + U(i+1,j)
                     +  U(i,j-1) + U(i,j+1))
  end for
end for
if me not 0 then
  bsp_put( to the left )
endif
if me not NPROCS – 1 then
  bsp_put( to the right )
Endif
bsp_sync()
```

BSP - example

$$T_{\text{superstep}} = w + g * h + l$$

h = max number of messages

= I values to the left +

I values to the right

= 2 * I (ignoring the inverse communication!)

w = 4 * I * I / p

$$T_{\text{superstep}} = 4\frac{I^2}{p} + 2 * g * I + l$$

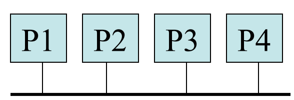
BSP - example

- BSP parameters for a wide variety of architectures has been published.

Machine	s	p	l	g
Origin 2000	101	4 32	1789 39057	10.24 66.7
Cray T3E	46.7	4 16	357 751	1.77 1.66
Pentium 10Mbit	61	4 8	139981 826054	1128.5 2436.3
Pentium II 100Mbit	88	4 8	27583 38788	39.6 38.7

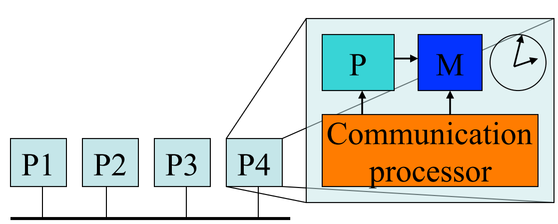
A more sophisticated model LogP

- Tend to be more empirical and network-related.



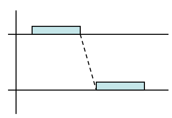
A more sophisticated model LogP

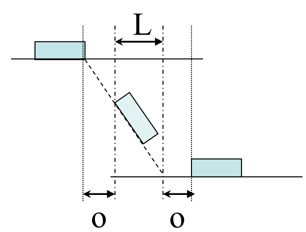
- Tend to be more empirical and network-related.



LogP

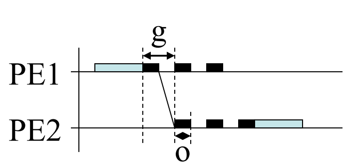
- Decompose the communications in 3 elements:
  - Latency : small message cross the network
  - overhead : lost time in communication

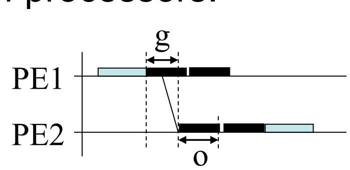




LogP

- Decompose the communications in 3 elements:
  - Latency : small message cross the network
  - overhead : lost time in communication
  - gap : between 2 consecutive messages
- And P the number of processors.





Both g and o matter !

g > o

g < o

LogP

- The total time for a message to go from the processor A to the processor B is:

  L + 2 * o

- There is no model for the application
- We can describe the application using the same approach as for BSP: supersteps

$$T_{\text{superstep}} = w + h * (L + 2o) + l$$

LogP

- The P parameter does not interfere in the superstep computation ?
- When the number of processors is not fixed:
  - The time of the computation change w(p)
  - The number of messages change h(p)
  - The synchronization time change l(p)

LogP

- Allow/encourage the usage of general techniques of designing algorithms for distributed memory machines: exploiting locality, reducing communication complexity and overlapping communication and computation.
- Balanced communication to avoid overloading the processors.

LogP

- Interesting concept : idea of finite capacity of the network. Any attempt to transit more than a certain amount of data will stall the processor.

$\left\lceil \frac{L}{g} \right\rceil$

- This model does not address on the issue of message size, even the worst is the assumption of all messages are of ``small'' size.
- Does not address the global capacity of the network.

Design a LogP program

- Execution time is the time of the slowest process
- Implications for algorithms:
  - – Balance computation
  - – Balance communications

  Are only sub-goals !
- Remember the capacity constraint $\left\lceil \frac{L}{g} \right\rceil$

LogP Machines

Maschine	L	o	g	P
CM-5	6	2.2	4	512
Meiko CS-2	8.6	1.7	$14.2 + 0.03x$	64
Power Xplorer	$21 - 0.82x$	$70 + x$	$115 + 1.43x$	8
Para-Station	$50 - 0.10x$	$3 + 0.112x$	$3 + 0.119x$	4
IBM SP-2	$13 - 0.005x$	$8 + 0.008x$	$10 + 0.01x$	128
IBM SP-2	$17 - 0.005x$	$8 + 0.008x$	$10 + 0.01x$	256

Improving LogP

- First model to break the synchrony of parallel execution
- LogGP : augments the LogP model with a linear model for long messages
- LogGPC model extends the LogGP model to include contention analysis using queuing model on the k-ary n-cubes network
- LogPQ model augments the LogP model on the stalling issue of the network constraint by adding buffer queues in the communication lines.

The CCM model

- Collective Computing Model transform the BSP superstep framework to support high-level programming models as MPI and PVM.
- Remove the requirement of global synchronization between supersteps, but combines the message exchanges and synchronization properties into the execution of a collective communication.
- Prediction quality usually high.