Concurrency, Parallelism and Distribution (CPD)

Modeling and analyzing performance

Eduard Ayguadé, Josep R. Herrero, Daniel Jiménez
({eduard,josepr,djimenez}@ac.upc.edu)

Computer Architecture Department
Universitat Politècnica de Catalunya

2012/13-Autumn

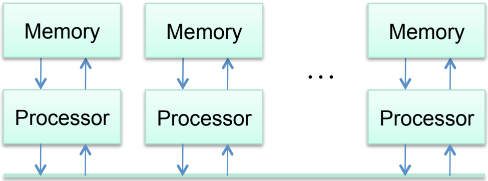
Outline

Data sharing modeling

Tools for performance understanding

How to model data sharing?

We start with a simple architectural model in which each processor P_i has its own memory, interconnected with the other processors through an interconnection network.



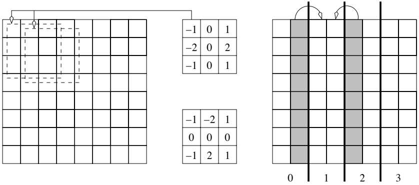
- Processors access to local data (in its own memory) using regular load/store instructions
- We will assume that local accesses take zero time units.

How to model data sharing?

- Processors can access remote data (in other processors) using a message-passing model, in which a pair of processors get involved in the data sharing (sender and receiver)
- To model the time needed to access remote data we will use two components:
  - Start up: time spent (t_s) in preparing the access to remote data
  - Remote access: time spent in accessing and transferring the data (number of bytes m, time per byte t_w) from the remote location

$$T_{access} = t_s + m \times t_w$$

Parallel time, speedup and efficiency example[1]



(a) (b) (c)

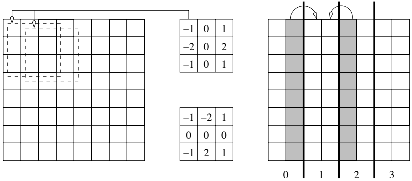
- Edge-Detection in a image of $n \times n$ pixels (output in a different matrix)
- Apply 3×3 template to each pixel

$$apply\_ij = \sum_{ii=i-1}^{i+1} \sum_{jj=j-1}^{jj=j+1} image[ii][jj] * template[ii-i+1][jj-j+1]$$

- If each multiply-add takes t_c, which is the sequential time?

[1] Introduction to Parallel Computing, A. Grama et al. Addison Wesley.

Parallel time, speedup and efficiency example (cont.)



(a) (b) (c)

- Data decomposition:
  - Column decomposition, each processor with n^2/P pixels (segment)
  - Left and right boundaries, each with n pixels
- Parallelization strategy:
  1. Exchange boundaries with the two adjacent processors
  2. Each processor applies the template to its segment
- Questions:
  1. What is the data sharing time per segment assuming each boundary is accessed using a single message?
  2. What is the total time (computation and data sharing)?

Parallel time, speedup and efficiency example (cont.)

- The total time for the algorithm is:

$$T_P = 9t_c\frac{n^2}{P} + 2(t_s + t_w n)$$

- The corresponding values of speedup and efficiency are:

$$S = \frac{9t_c n^2}{9t_c\frac{n^2}{P} + 2(t_s + t_w n)}$$

$$E = \frac{1}{1 + \frac{2P(t_s + t_w n)}{9t_c n^2}}$$

- Which is the $N_{1/2}$ for half performance?

Blocking parallelization example

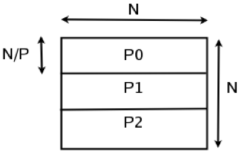
- Stencil algorithm: updates each matrix element using its 4 neighbors

```
#include <math.h>
void compute( int N, double *u) {
    int i, k;
    double tmp;

    for ( i = 1; i < N-1; i++ ) {
       for ( k = 1; k < N-1; k++ ) {
          tmp = u[N*(i+1) + k] + u[N*(i-1) + k] + u[N*i + (k+1)] + u[N*i + (k-1)]
                       - 4 * u[N*i + k];
          u[N*i + k] = tmp/4;
       }
    }
}
```

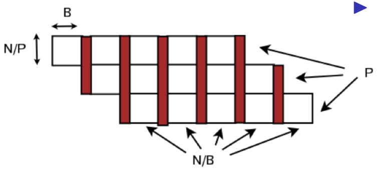
- If t_c is the computation time for one element, which is the sequential time for a $N \times N$ matrix?

Blocking parallelization example (cont.)



- Data decomposition:
  - Row distribution, each processor with N^2/P elements (segment)
  - Upper and lower boundaries, each with N elements
- Parallelization strategy:
  1. Communication of lower boundary (no dependence)
  2. Wait for upper boundary (dependence with previous processor)
  3. Apply stencil algorithm to segment
  4. Send last row of segment to next processor (upper boundary)
- Questions:
  1. What is the data sharing time per segment?
  2. What is the total time (computation and data sharing)?

Blocking parallelization example (cont.)

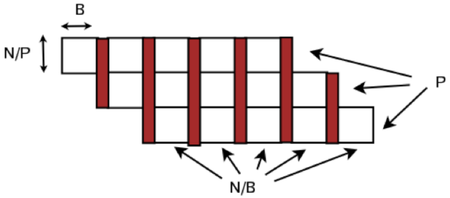


- Data decomposition:
  - Row distribution, each processor with N^2/P elements (segment, logically divided in blocks of B columns)
  - Upper and lower boundaries, each with N elements
- Parallelization strategy:

  Once: Communication of lower boundary (no dependence)

  1. Wait for B elements in upper boundary (dependence with previous processor)
  2. Apply stencil algorithm to block (B columns) in segment
  3. Send B elements in last row to next processor
- Questions:
  1. What is the data sharing time per segment?
  2. What is the total time (computation and data sharing)?

Blocking parallelization example: T_p



Simplification: assume $N - 2 \Rightarrow N$

$$T_p = (t_s + Nt_w) + (\frac{N}{P}B)(\frac{N}{B} + P - 1)t_c + (\frac{N}{B} + P - 2)(t_s + t_wB)$$

Blocking parallelization example: T_p simplified

$$T_p = (t_s + Nt_w) + (\frac{N}{P}B)(\frac{N}{B} + P - 1)t_c + (\frac{N}{B} + P - 2)(t_s + t_w B)$$

Assuming $P >> 2$:

$$\begin{aligned} T_p &\simeq (t_s + Nt_w) + (\frac{N}{P}B)(\frac{N}{B} + P)t_c + (\frac{N}{B} + P)(t_s + t_w B) \\ &= (t_s + Nt_w) + \frac{N^2}{P}t_c + NBt_c + t_s\frac{N}{B} + t_w N + t_s P + t_w PB \end{aligned}$$

In order to obtain the optimum block size B we have to apply the derivative and equal it to zero.

Blocking parallelization example: optimum B computation

Using previous equation and assuming $N >>> P$, optimum B block size is:

$$\begin{aligned}
\frac{\partial T}{\partial B} &= Nt_c - t_s \frac{N}{B^2} + t_w P = 0 \\
B_{opt} &= \sqrt{\frac{t_s N}{N t_c + t_w P}} = \sqrt{\frac{t_s}{t_c + t_w \frac{P}{N}}} \\
&\simeq \sqrt{\frac{t_s}{t_c}}
\end{aligned}$$

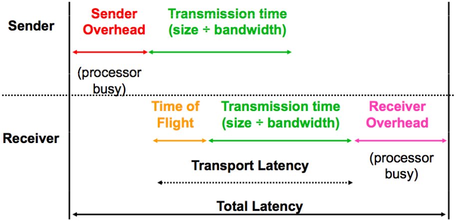
Then, T_{opt} (using aprox. B_{opt}) is :

$$T_{opt} = t_s + N t_w + (\frac{N^2}{P}) t_c + 2N\sqrt{t_s t_c} + t_w N + t_s P + t_w P \sqrt{\frac{t_s}{t_c}}$$

Better communication model

Two metrics to better model communication: network diameter and latency

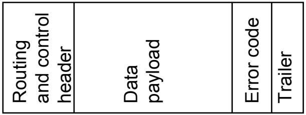
- Diameter: the maximum (over all pairs of nodes) of the shortest path between a given pair of nodes.
- Maximum hardware latency (wire delay) varies with diameter
- Software latency



- Latency is important for programs with many small messages

Network banwidth

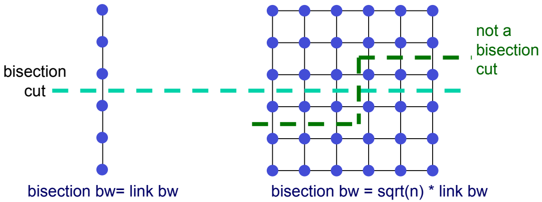
- Banwidth of a link $w * 1/t$, being w the number of wires and t the time per bit
- Effective bandwidth usually lower than physical link bandwidth due to packet overhead



- Bandwidth is important for applications with mostly large messages

Network bisection banwidth

- Bisection bandwidth: bandwidth across smallest cut that divides network into two equal halves
- Bandwidth across 'narrowest' part of the network



- Bisection bandwidth is important for algorithms in which all processors need to communicate with all others

Diameter and bisection banwidth: examples

Topology	Diameter	Bisection banwidth (#links)
1D line	$n-1$	1
1D ring	$\frac{n}{2}$	2
2D mesh	$2 \times (\sqrt{n}-1)$	$\sqrt{n}$
2D torus	$2 \times \lfloor \sqrt{n}/2 \rfloor$	$2 \times \sqrt{n}$
d-hypercube	$\log n$	$\frac{n}{2}$
Tree	$2 \times \log((n+1)/2)$	1
Fat tree	$2 \times \log((n+1)/2)$	$\frac{n}{2}$
Omega	$\log n$	$\frac{n}{2}$

One-to-All (Broadcast) Cost Example

Topology	One-to-All cost (m words)	# Processors
1D line	$(t_s + t_w m) \log n$	n
1D ring	$(t_s + t_w m) \log n$	n
2D mesh	$2 \times ((t_s + t_w m) \log \sqrt{n})$	n
2D torus	$2 \times ((t_s + t_w m) \log \sqrt{n})$	n
d-hypercube	$d \times ((t_s + t_w m) \log 2)$	$n = 2^d$
Binary Tree	$(t_s + t_w m) \log n$	n
Fat Tree	$(t_s + t_w m) \log n$	n
Omega	$(t_s + t_w m) \log n$	n

What is the cost of a Reduce (All-to-one) in...?

Topology	All-to-one cost (m words)	# Processors
1D line		n
1D ring		n
2D mesh		n
2D torus		n
d-hypercube		$n = 2^d$
Binary Tree		n
Fat Tree		n
Omega		n

Outline

Data sharing modeling

Tools for performance understanding

Concurrency, Parallelism and Distribution (CPD)

Modeling and analyzing performance

Eduard Ayguadé, Josep R. Herrero, Daniel Jiménez
({eduard,josepr,djimenez}@ac.upc.edu)

Computer Architecture Department
Universitat Politècnica de Catalunya

2012/13-Autumn