Control Systems Optimization

Igor Wojnicki

AGH – Univeristy of Science and Technology

2010

Outline

Parallel Systems

- Concurrency.
- Parallelism.

Parallelism/Concurrency: Why?

- Simpler (more natural, straightforward) modeling of real world processes.
- Robust, Fault Tolerant Systems.
  - If a process crashes the crash is contained.
- Computation time optimization/speedup.
  - Real-time compliance – results (partial results) delivered on time.

Program Performance Metrics: Speedup

- The parallel run time (Tpar) is the time from the moment when computation starts to the moment when the last processor finished its execution
- The speedup (S) is defined as the ratio of the time needed to solve the problem on a single processor (Tseq) to the time required to solve the same problem on parallel system with "p" processors (Tpar):

$$S = \frac{T_{seq}}{T_{par}}$$

  - relative – Tseq is the execution time of parallel algorithm executing on one of the processors of parallel computer
  - real – Tseq is the execution time for the best-know algorithm using one of the processors of parallel computer
  - absolute – Tseq is the execution time for the best-know algorithm using the best-know computer

Program Performance Metrics: Efficiency

- The efficiency (E) of parallel program is defined as a ratio of speedup to the number of processors

$$E = \frac{S}{p}$$

- The cost is usually defined as a product of a parallel run time and the number of processors

$$C$$

- The scalability of parallel system is a measure of its capacity to increase speedup in proportion to the number of processors

Stone's Table

Speedup (S)	Type of Algorithms
$\alpha * p$	matrix computations, discretization
$\frac{\alpha * p}{\log_2 p}$	sorts, linear recursions, evaluation of polynomials
$\alpha * \log_2 p$	search for an element in a set
α	certain non-linear recursions

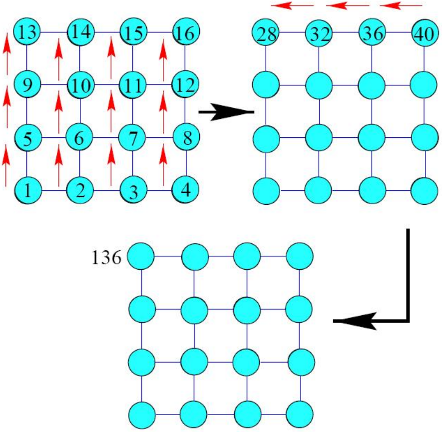
Where p is a number of used processors and α is a positive number smaller then 1 which depends on the machine/architecture.

Scalability of Parallel Systems

- The parallel system is scalable if it maintains speedup at a fixed value while increasing the number of processors and the size of the problem.
- The scalability of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processors.
- The scalability reflects a parallel system ability to utilize increasing processing resources effectively.

Mesh Architecture

Adding n numbers on the mesh with p processors: $\sqrt{p}$ steps.



Mesh: details

- In the first step each processor locally adds its n/p numbers, in the next steps half partial sums are transmitted to adjacent processors and added, the procedure finished when one chosen processor gets the final sum
- Assume that it takes one unit of time both to add two numbers and to send a number between two directly connected processors
- Then adding the n/p numbers local to each processor takes $n/p - 1$ time
- The p partial sums are added in $\sqrt{p}$ steps (one addition and one communication: $2 * \sqrt{p}$)

Mesh: Speedup, Efficiency

- Thus the total parallel run time can be approximated by:

$$T_{par} = n/p + 2 * \sqrt{p}$$

- Since serial run time can be approximated by n the expression for speedup and efficiency are as follows:

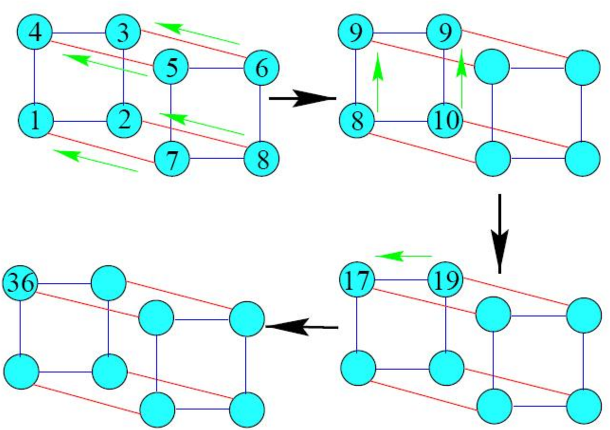
$$S = \frac{n * p}{n + 2p\sqrt{p}}$$

$$E = \frac{n}{n + 2p\sqrt{p}}$$

Hypercube Architecture

http://4d.shadowpuppet.net/4d.html

- Adding numbers on a Hypercube with p processors: $\log_2 p$ steps.



Hypercube: details

- In the first step each processor locally adds its n/p numbers, in the next steps half partial sums are transmitted to adjacent processors and added, the procedure finished when one chosen processor gets the final sum
- Assume that it takes one unit of time both to add two numbers and to send a number between two directly connected processors
- Then adding the n/p numbers local to each processor takes $n/p - 1$ time
- The p partial sums are added in $\log_2 p$ steps (one addition and one communication: $2 * \log_2 p$)

Hypercube: Speedup, Efficiency

- Thus the total parallel run time can be approximated by:

$$T_{par} = n/p + 2 * \log_2 p$$

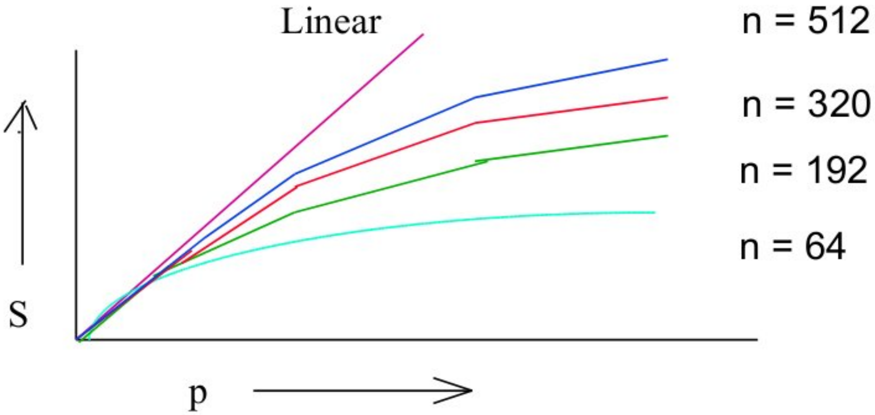
- Since serial run time can be approximated by n the expression for speedup and efficiency are as follows:

$$S = \frac{n * p}{n + 2p \log_2 p}$$

$$E = \frac{n}{n + 2p \log_2 p}$$

- These expressions can be used to calculate the speedup and efficiency for any pair of n and p

Speedup versus the number of processors



Speedup: Conclusions

- For the given problem instance, the speedup does not increase linearly as the number of processors increases
- A larger instance of the same problems yields higher speedup and efficiency for the same number of processors

Scalability, recap.

- The parallel system is scalable if it maintains speedup at a fixed value while increasing the number of processors and the size of the problem.
- The scalability of a parallel system is a measure of its capacity to increase speedup in proportion to the number of processors.
- The scalability reflects a parallel system ability to utilize increasing processing resources effectively.

Amdahl's Law

- When executing a parallel program we can distinguish two program parts: sequential part (Pseq) which needs to be executed sequentially using one processor parallel part (1-Pseq) which can be executed independently using number of processors
- Let's assume that if we execute this program at single processor the serial execution time will be t1. Then if p indicates the number of used processors during parallel execution the parallel run time can be expressed by:

$$T_{par} = t_1 * P_{seq} + (1 - P_{seq}) * t_1 / p$$

- Speedup:

$$S = \frac{t_1}{t_1 * P_{seq} + (1 - P_{seq}) * t_1 / p} \leq \frac{1}{P_{seq}}$$

Amdahl's Law: Example

- Question : Suppose you want to achieve a speedup of 80 with 100 processors. What fraction of the original computation can be sequential?
- Answer: From Amdhal's law: speedup is limited by the sequential fraction of the program:

$$S = \frac{t_1}{t_1 * P_{seq} + (1 - P_{seq}) * t_1 / p}$$

- Thus:

$$P_{seq} = \frac{\frac{p}{S} - 1}{p - 1}$$

- Then:

$$P_{seq} = 0.0025$$

Bottom Line

- Use parallelism if:
  - time limits,
  - a need for fastest computations.
- Additional requirements:
  - sequential part of the program,
  - parallel part of the program.
- Gain can be achieved on the parallel part only.
- Keep in mind communication lags.