High Performance Computing Systems (CMSC714)

Lecture 13: Isoefficiency and Perf. Modeling

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Summary of last lecture

- MPI trace visualization
- Projections performance analysis tool
- Hatchet: programmable by the user

Isoefficiency

- Relationship between problem size and number of processors to maintain a certain level of efficiency
- At what rate should we increase problem size with respect to number of processors to keep efficiency constant

Speedup and efficiency

- Speedup: Ratio of execution time on one process to that on *n* processes

$$\text{Speedup} = \frac{t_1}{t_n}$$

- Efficiency: Speedup per process

$$\text{Efficiency} = \frac{t_1}{t_n \times n}$$

Efficiency in terms of overhead

- Total time = (useful) computation + overhead (communication + idle time)

$$n \times t_n = t_1 + t_o$$

$$\text{Efficiency} = \frac{t_1}{t_n \times n} = \frac{t_1}{t_1 + t_o} = \frac{1}{1 + \frac{t_o}{t_1}}$$

Isoefficiency function

- Sequential time = Problem size (number of operations) X time to do each operation

$$t_1 = W \times t_c$$

$$\text{Efficiency} = \frac{1}{1 + \frac{t_o}{W \times t_c}}$$

- Efficiency is constant if t_o / W is constant

$$W = K \times t_o$$

Performance Modeling

- Model the performance of a parallel application
- Different methods
  - Analytical
  - Empirical
  - Simulation

LogP model

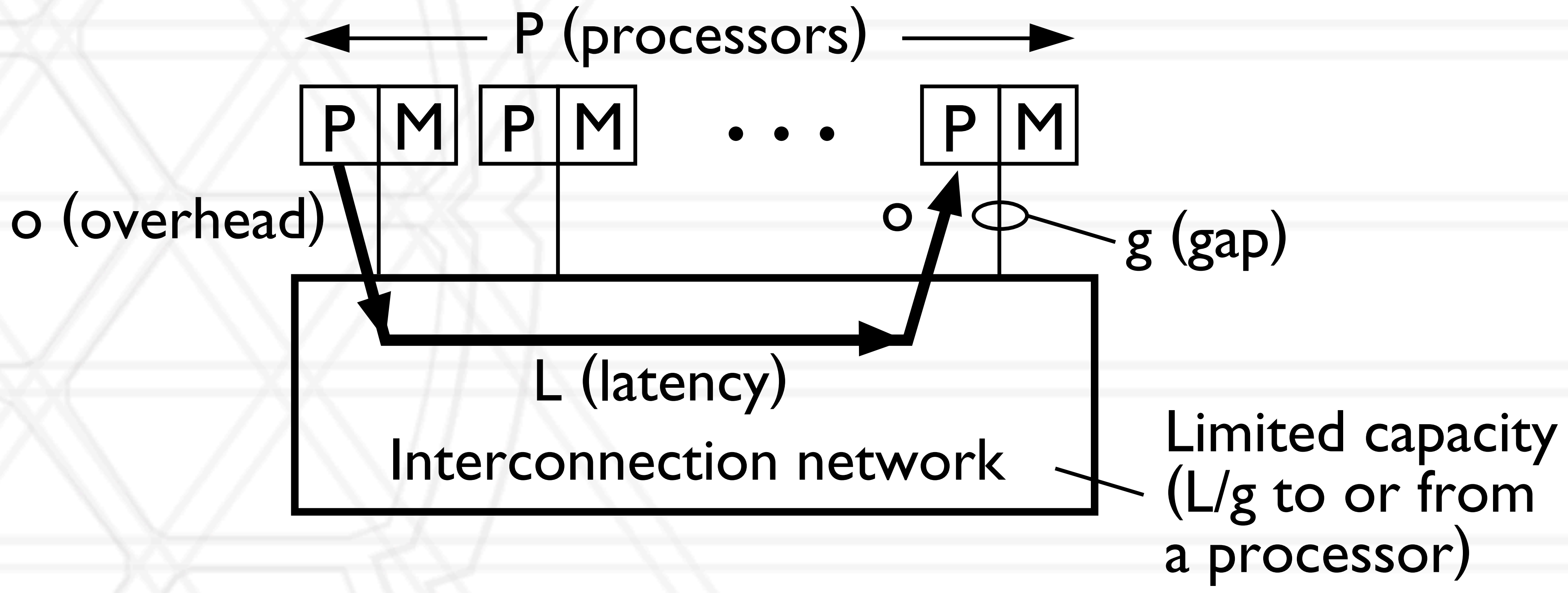
- Model for communication on an interconnection network

L: latency or delay

O: overhead (processor busy in communication)

g: gap

P: number of processors / processes



1/g = bandwidth

alpha + n * beta model

- Another model for communication

$$T_{\text{comm}} = \alpha + n \times \beta$$

α: latency

n: size of message

β: bandwidth

Questions

Isoefficiency: Measuring the Scalability of Parallel Algorithms and Architectures

- Main assumption made in the paper is that overhead increases linearly with the processor count but it is independent of work 'w'. Is this a feasible assumption? Don't we usually increase our messaging, waits etc. when there's more input? What is the situation in real systems?
- Is there a consensus on parallel algorithm modeling, like do academics use something like isoefficiency to explain how approximately efficient an algorithm is in their papers? I'm curious how exactly a parallel algorithm is designed and tested to prove it's scalable.
- Can you show some examples of the isoefficiency metric? And how does the speedup curve look like?
- For problem that have low overhead and limited concurrency (like Dijkstra's problem), we can create a fake high concurrency environment, can we use the dynamic load balancing strategy?
- How to analyze the system's isoefficiency due to contention? What is the most important factor that affect the overhead time?

Questions

LogP: A Practical Parallel Model of Computation

- How is gap related to the bandwidth? Also why network capacity is defined as L/g?
- What is AM (Active Message Libraries)? how do they work and how we can utilize them?
- Is LogP only suitable for fast algorithms? Any other tools designed for analyzing the complexity?
- Why block and block cyclic layout do not yield optimal parallel algorithm? Briefly illustrate
- What does saturation, long messages specialized hardware support and communication patterns be useful as parameters?
- In general do you think LogP is a good candidate for analysis? Which analysis tool you recommend most and most widely used for a wide range of problems?

Questions?

UNIVERSITY OF MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu