Models of Parallel Computation

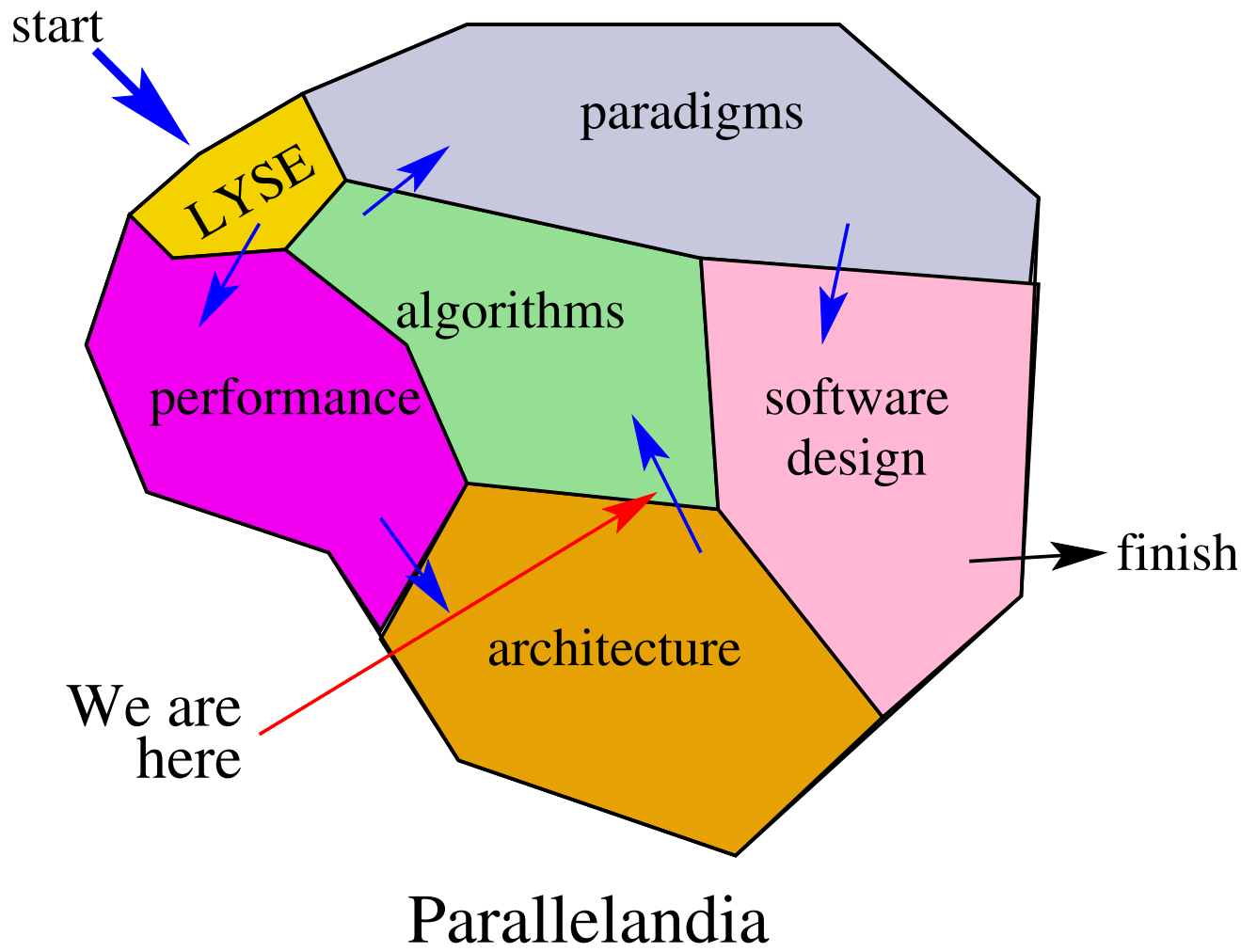
Mark Greenstreet

CpSc 448B – Oct. 18, 2011

Lecture Outline

- The RAM Model of Sequential Computation
- Models of Parallel Computation
  - PRAM
  - CTA
  - logP

The Big Picture



For more details, see the midterm review:

http://www.ugrad.cs.ubc.ca/~cs448b/exams/midterm-review.pdf

The RAM Model

RAM = Random Access Machine

- Axioms of the model
  - Machines work on words of a “reasonable” size.
  - A machine can perform a “reasonable” operation on a word as a single step.
    - such operations include addition, subtraction, multiplication, division, comparisons, bitwise logical operations, bitwise shifts and rotates.
  - The machine has an unbounded amount of memory.
    - A memory address is a “word” as described above.
    - Reading or writing a word of memory can be done in a single step.

The Relevance of the RAM Model

- If a single step of a RAM corresponds (to within a factor close to 1) to a single step of a real machine.
- Then algorithms that are efficient on a RAM will also be efficient on a real machine.
- Historically, this assumption has held up pretty well.
  - For example, `mergesort` and `quicksort` are better than `bubblesort` on a RAM and on real machines, and the RAM model predicts the advantage quite accurately.
  - Likewise, for many other algorithms
    - ⋆ graph algorithms, matrix computations, dynamic programming, ...
    - ⋆ hard on a RAM generally means hard on a real machine as well: NP complete problems, undecidable problems, ...

The Irrelevance of the RAM Model

The RAM model is based on assumptions that don't correspond to physical reality:

- Memory access time is highly non-uniform.
  - Architects make heroic efforts to preserve the illusion of uniform access time fast memory –
    - caches, out-of-order execution, prefetching, . . .
  - – but the illusion is getting harder and harder to maintain.
    - Algorithms that randomly access large data sets run much slower than more localized algorithms.
    - Growing memory size and processor speeds means that more and more algorithms have performance that is sensitive to the memory hierarchy.
- The RAM model does not account for energy:
  - Energy is the critical factor in determining the performance of a computation.
  - The energy to perform an operation drops rapidly with the amount of time allowed to perform the operation.

The PRAM Model

PRAM = Parallel Random Access Machine

- Axioms of the model
  - A computer is composed of multiple processors and a shared memory.
  - The processors are like those from the RAM model.
    - ★ The processors operate in lockstep.
    - ★ I.e. for each $k > 0$, all processors perform their k^{th} step at the same time.
  - The memory allows each processor to perform a read or write in a single step.
    - ★ Multiple reads and writes can be performed in the same cycle.
    - ★ If each processor accesses a different word, the model is simple.
    - ★ If two or more processors try to access the same word on the same step, then we get a bunch of possible models:

      EREW: Exclusive-Read, Exclusive-Write
      CREW: Concurrent-Read, Exclusive-Write
      CRCW: Concurrent-Read, Concurrent-Write

EREW, CREW, and CRCW

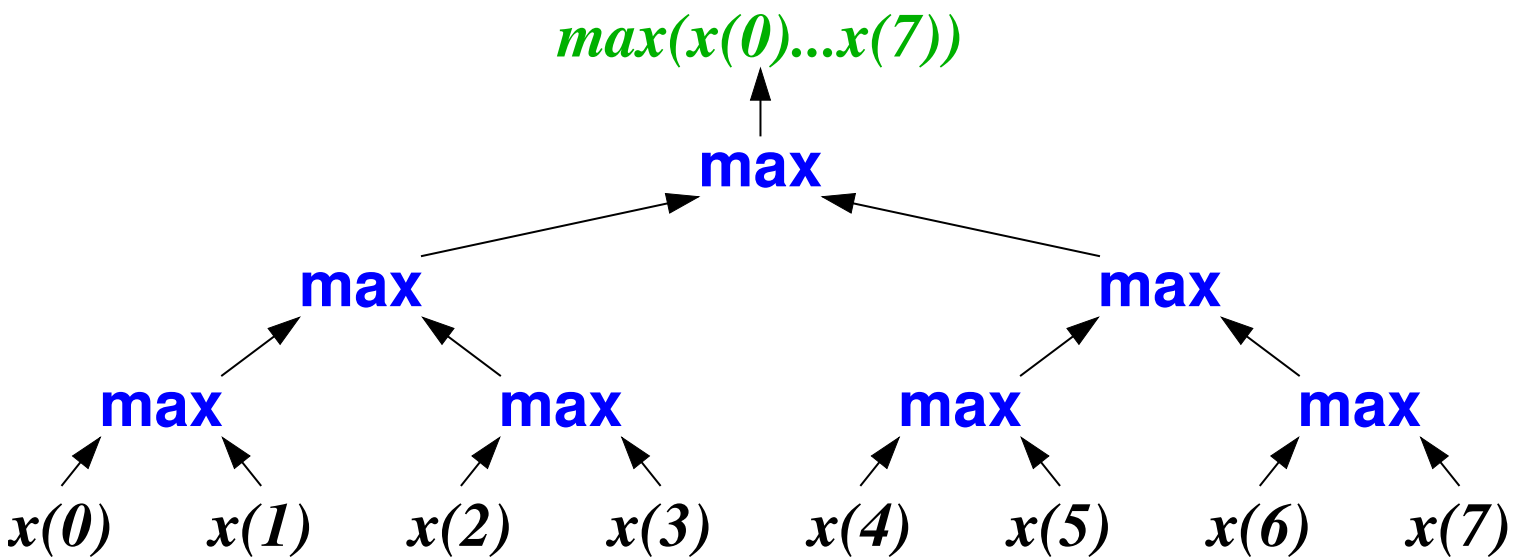
- EREW: Exclusive-Read, Exclusive-Write
  - If two processors access the same location on the same step,
    - ⋆ then the machine fails.
- CREW: Concurrent-Read, Exclusive-Write
  - Multiple machines can read the same location at the same time, and they all get the same value.
  - At most one machine can try to write a particular location on any given step.
  - If one processor writes to a memory location and another tries to read or write that location on the same step,
    - ⋆ then the machine fails.
- CRCW: Concurrent-Read, Concurrent-Write

  If two or more machines try to write the same memory word at the same time, then if they are all writing the same value, that value will be written. Otherwise,
  - the machine fails, or
  - one of the writes “wins”, or
  - an arbitrary value is written to that address.

Fun with the PRAM Model

Finding the maximum element of an array of N elements.

- The obvious approach
  - Do a reduce.
  - Use $N/2$ processors to compute the result in $\Theta(\log_2 N)$ time.

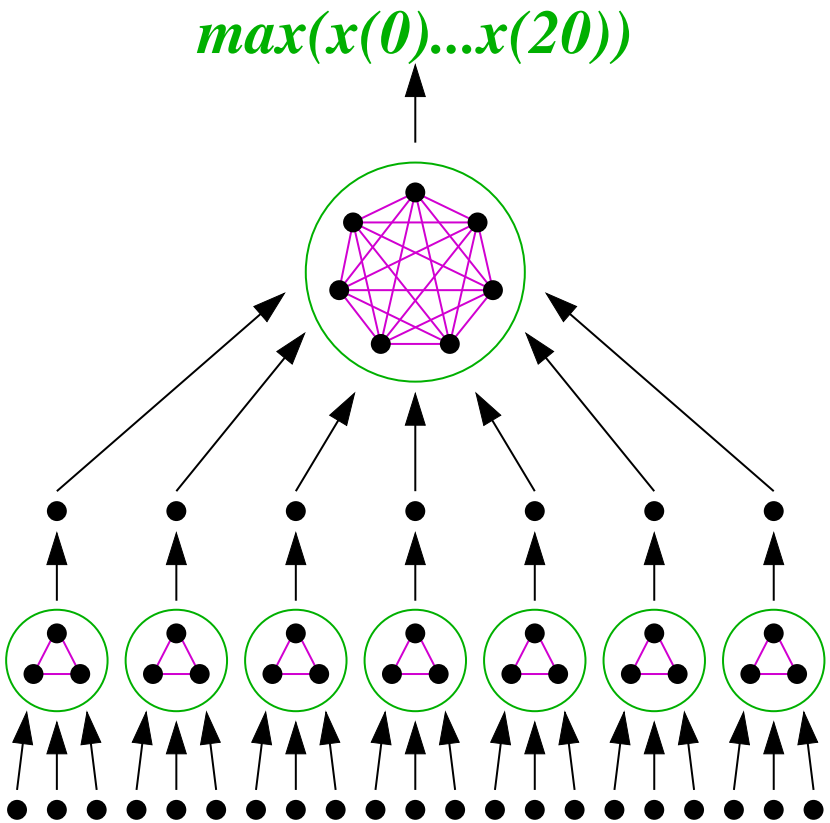


A Valiant Solution

L. Valiant, 1975

- Use P processors.
- Step 1:
  - Divide the N elements into $N/3$ sets of size 3.
  - Assign 3 processors to each set, and perform all three pairwise comparisons in parallel.
  - Mark all the "losers" (requires a CRCW PRAM) and move the max of each set of three to a fixed location.
- Step 2:
  - We now have $N/3$ elements left and still have N processors.
  - We can make groups of 7 elements, and have 21 processors per group, which is enough to perform all $\binom{7}{2} = 21$ pairwise comparisons in a single step.
  - Thus, in $O(1)$ time we move the max of each set to a fixed location. We now have $N/21$ elements left to consider.

Visualizing Valiant



max from group of 7
(21 parallel comparisons)

group of 7 values

max from each group
(3 parallel comparisons/group)

groups of 3 values

N values, N processors

A Valiant Solution

- Subsequent steps:
  - On step k, we have N/m_k elements left.
  - We can make groups of $2m_k + 1$ elements, and have $m_k(2m_k + 1) = \binom{2m_k + 1}{2}$ processors per group, which is enough to perform all pairwise comparisons in a single step.
  - We now have $N/(m_k(2m_k + 1))$ elements to consider.
- Run-time:
  - The sparsity is squared at each step.
  - It follows that the algorithm requires $O(\log \log N)$.
  - Valiant showed a matching lower bound and extended the results to show merging is $\theta(\log \log N)$ and sorting is $\theta(\log N)$ on a CRCW PRAM.

The Irrelevance of the PRAM Model

The PRAM model is based on assumptions that don't correspond to physical reality:

- Connecting N processors with memory requires a switching network.
  - Logic gates have bounded fan-in and fan-out.
  - $\Rightarrow$ and switch fabric with N inputs (and/or N outputs) must have depth of at least $\log N$.
  - This gives a lower bound on memory access time of $\Omega(\log N)$.
- Processors exist in physical space
  - N processors take up $\Omega(N)$ volume.
  - The processor has a diameter of $\Omega(N^{1/3})$.
  - Signals travel at a speed of at most c (the speed of light).
  - This gives a lower bound on memory access time of $\Omega(N^{1/3})$.
- Valiant acknowledged that he was neglecting these issues in his original paper.
  - but that didn't deter lots of results being published for the PRAM model.

The CTA Model

CTA = Candidate Type Architecture

- Axioms of the model
  - A computer is composed of multiple processors.
  - Each processor has
    - ★ Local memory that can be accessed in a single processor step (like the RAM model).
    - ★ A small number of connections to an communications network.
  - A communication mechanism:
    - ★ Conveying a value between processors takes λ time steps.
    - ★ λ can range from 10^2 to 10^5 or more depending on the architecture.
    - ★ The exact communication mechanism is not specified.

Communication Mechanisms

- Shared Memory: $\lambda \approx 100 - 1000$.
- One-sided communication:
  - Used on some supercomputers (e.g. Cray).
  - `put(addr, data)`: copies data into the memory of a remote node.
  - `read(addr)`: reads data from the memory of a remote node.
  - Called "one-sided" because the remote-node doesn't do anything to receive or transmit the data involved.
- Message passing: $\lambda \approx 5000 - 10000^{+}$.

Latency vs. Throughput

- Definitions:
  - Latency is the amount of time it takes to perform an operation from start to finish.
  - Throughput is the number of operations that can be performed per unit time.
- Relations:
  - If we did everything sequentially, we would have

$$\text{Throughput} = \frac{1}{\text{Latency}}$$

  - But, with pipelined and/or parallel execution, we can have

$$\text{Throughput} \gg \frac{1}{\text{Latency}}$$

Latency vs. Throughput

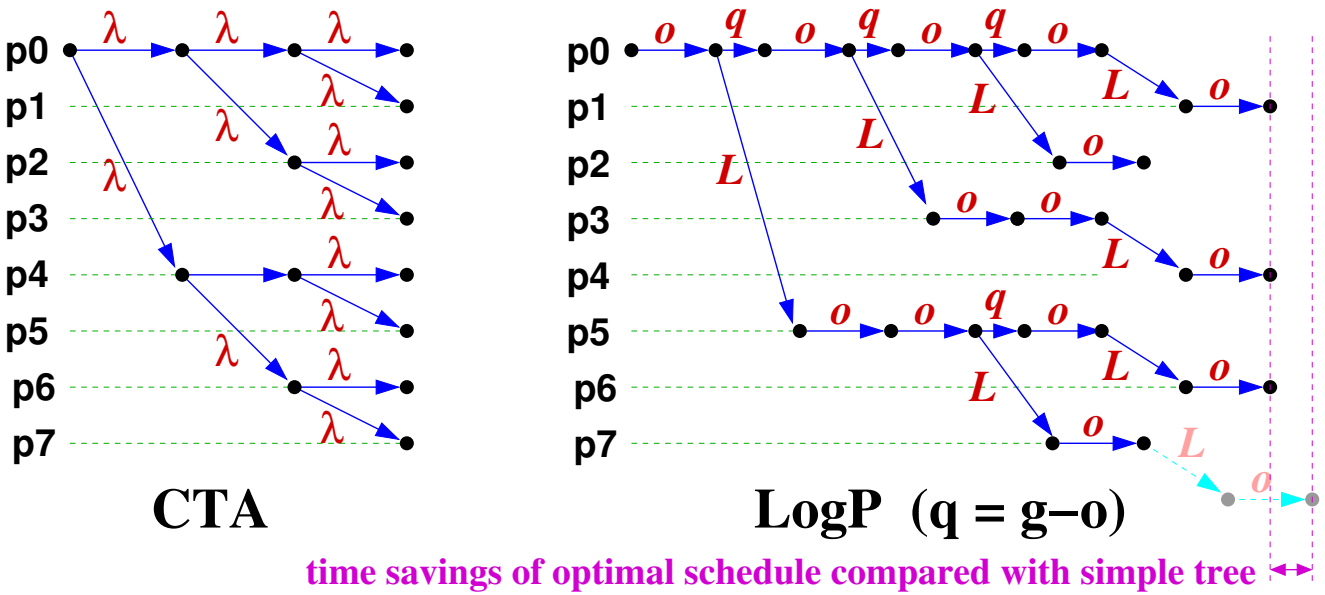
- Why does it matter:
  - Throughput (a.k.a. peak performance) is usually a lousy measurement of real performance: real programs have some latency critical operations.
  - Latency does not completely capture the performance of a parallel architecture
    - ⋆ If it take λ time units to send convey one word between two processors,
    - ⋆ We can probably send two words in $< 2\lambda$ time units.
    - ⋆ On the other hand, can we send a million words in $\approx \lambda$ time units?
    - ⋆ Bandwidth matters.

The LogP Model

- **Motivation (1993): convergence of parallel architectures**
  - Individual nodes have microprocessors and memory of a workstation or PC.
  - A large parallel machine had at most 2000 such nodes.
  - Point-to-point interconnect –
    - ★ Network bandwidth much lower than memory bandwidth.
    - ★ Network latency much higher than memory latency.
    - ★ Relatively small network diameter: 5 to 20 “hops” for a 1000 node machine.
- **The model parameters:**

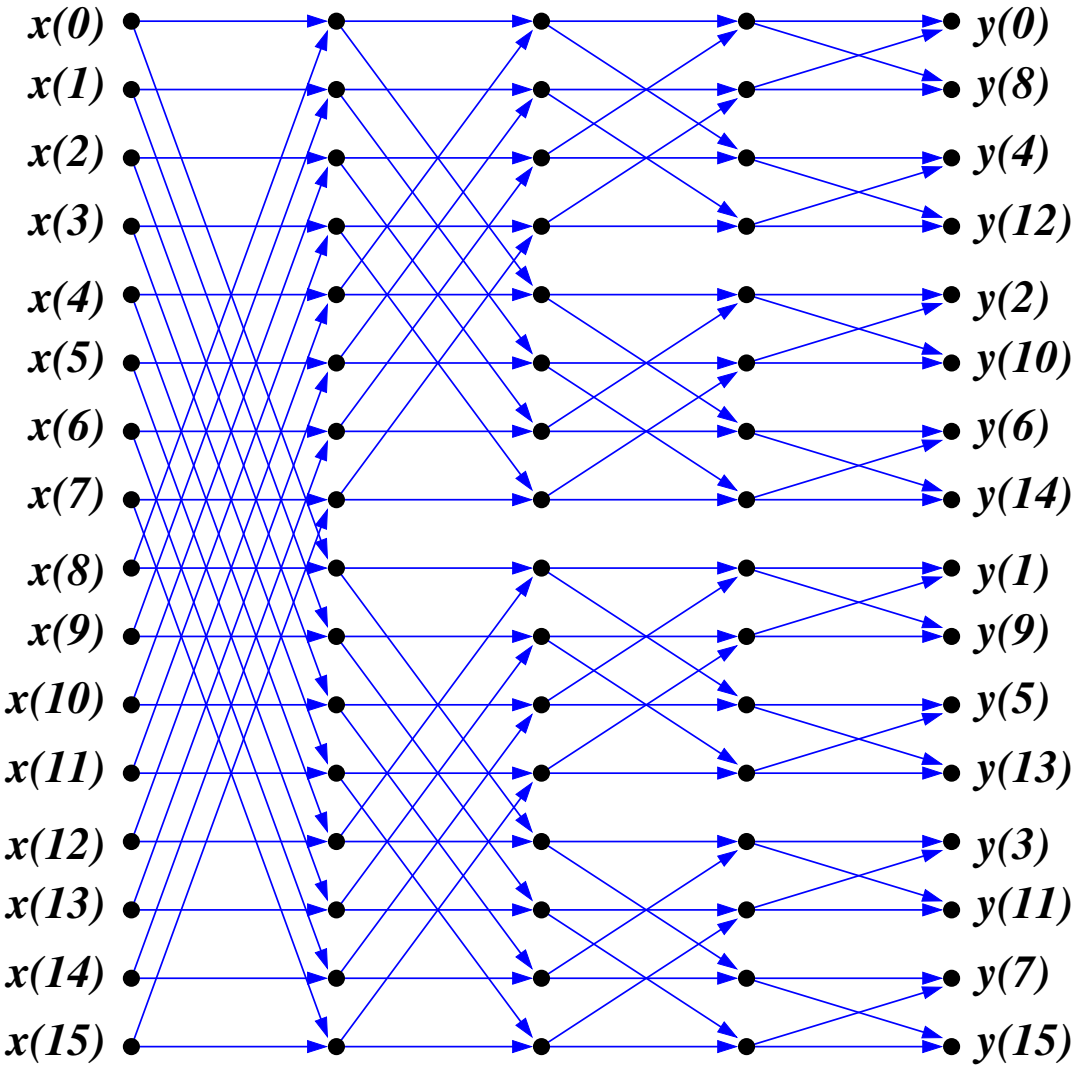
  L the latency of the communication network fabric
  o the overhead of a communication action
  g the bandwidth of the communication network
  P the number of processors

LogP Example: Broadcast



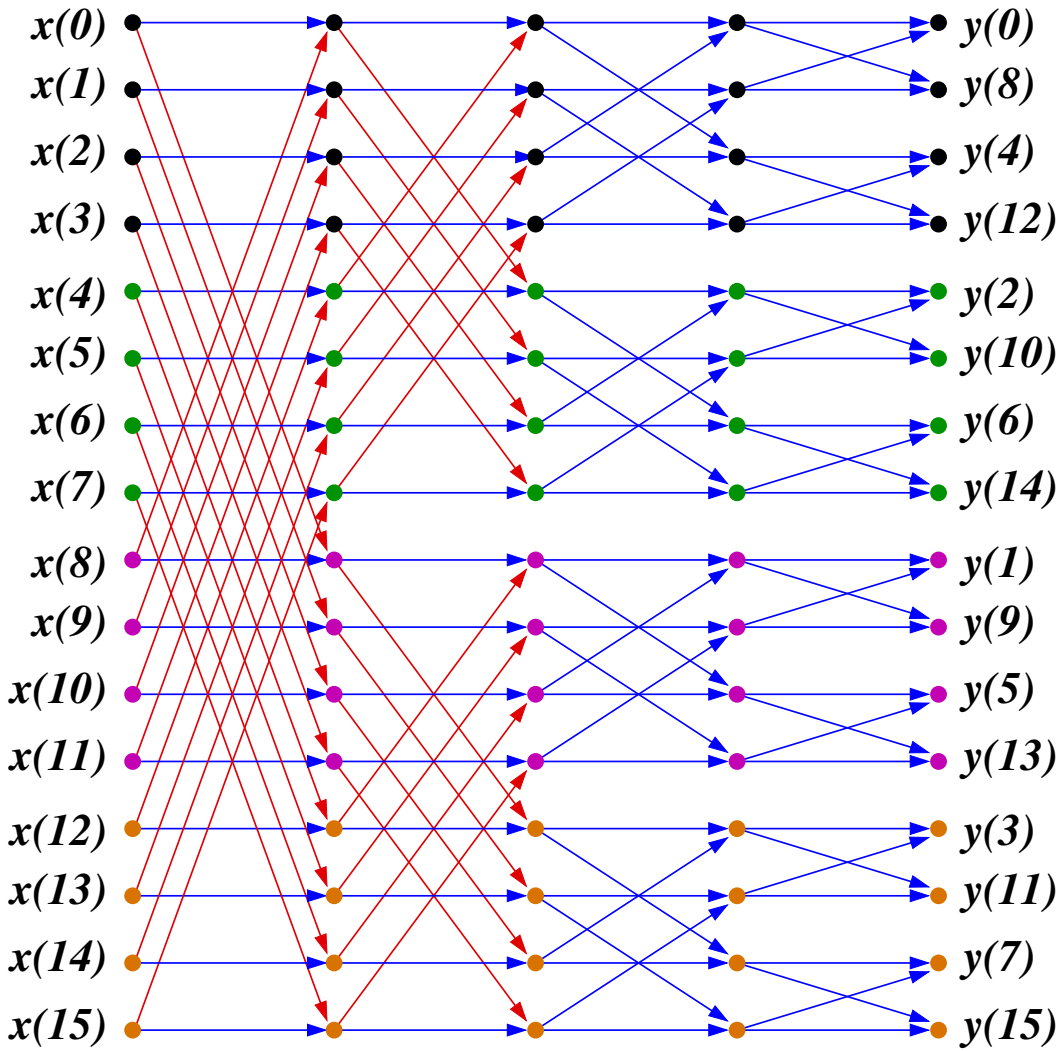
- LogP breaks communication into more detailed phases than CTA.
- If $g - o$ is enough smaller than L, then LogP shows that a the simple binary tree isn't exactly optimal for broadcast.
- Example: $L = 10$, $o = 2$, $g = 3$, $P = 8$:
  - Simple binary tree completes broadcast in $3L + 6o = 42$ time units.
  - Optimized tree completes in 28 time units
    - ★ p0 sends to p1, p2, p3, p4, p5, and p7
    - ★ p1 sends to p6
  - Is it worth it?

LogP Example: FFT (1/6)



- The Fast-Fourier transform is used in many signal processing applications:
  - audio signals
  - wi-fi modulation and demodulation
  - image filtering
  - voice recognition
  - ...
- The data flow of the FFT has the "butterfly" structure shown on the left.

LogP Example: FFT (2/6)



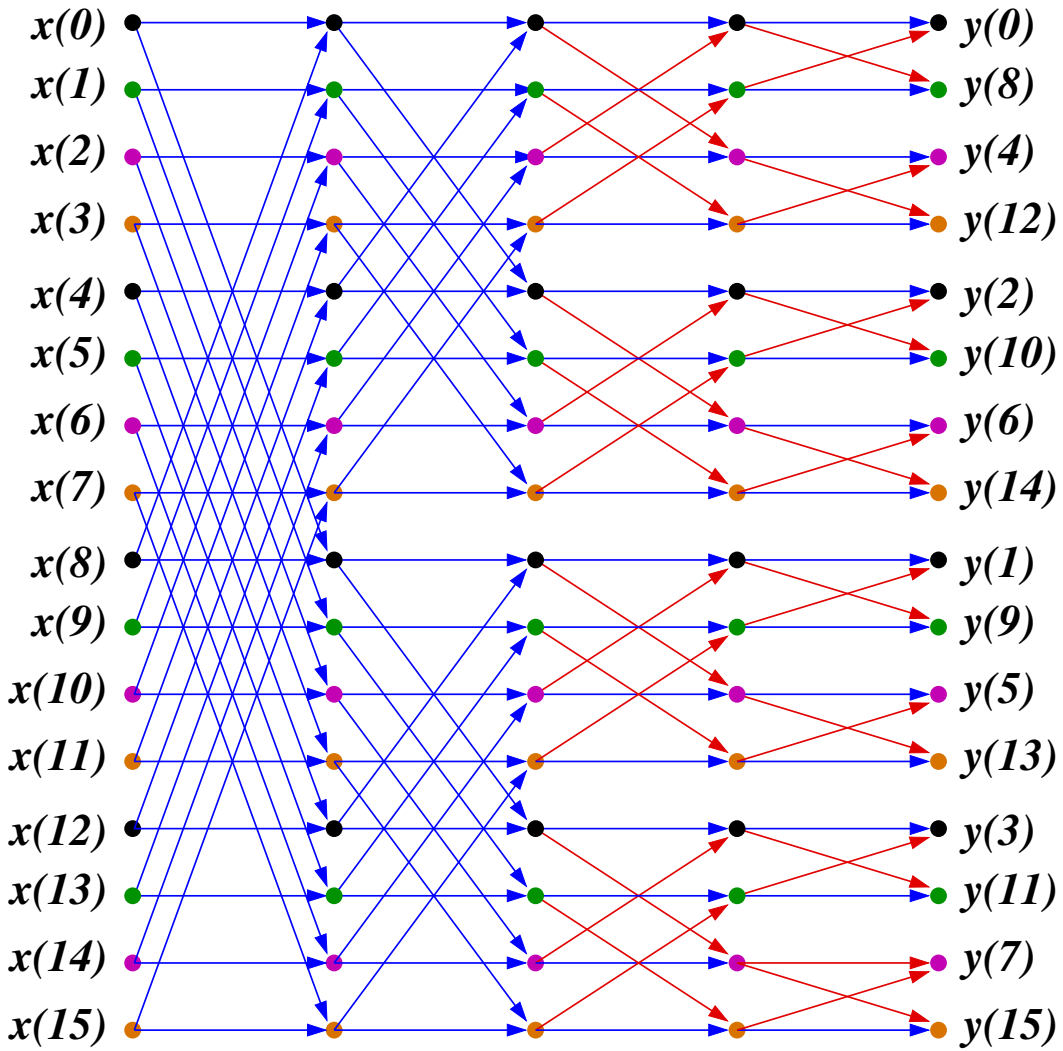
- processor 0
- processor 1
- processor 2
- processor 3

local communication

inter–processor communication

- First attempt to parallelize:
  - assign blocks of rows to processors.
  - lots of communication at the left
  - everything local at the right.

LogP Example: FFT (3/6)



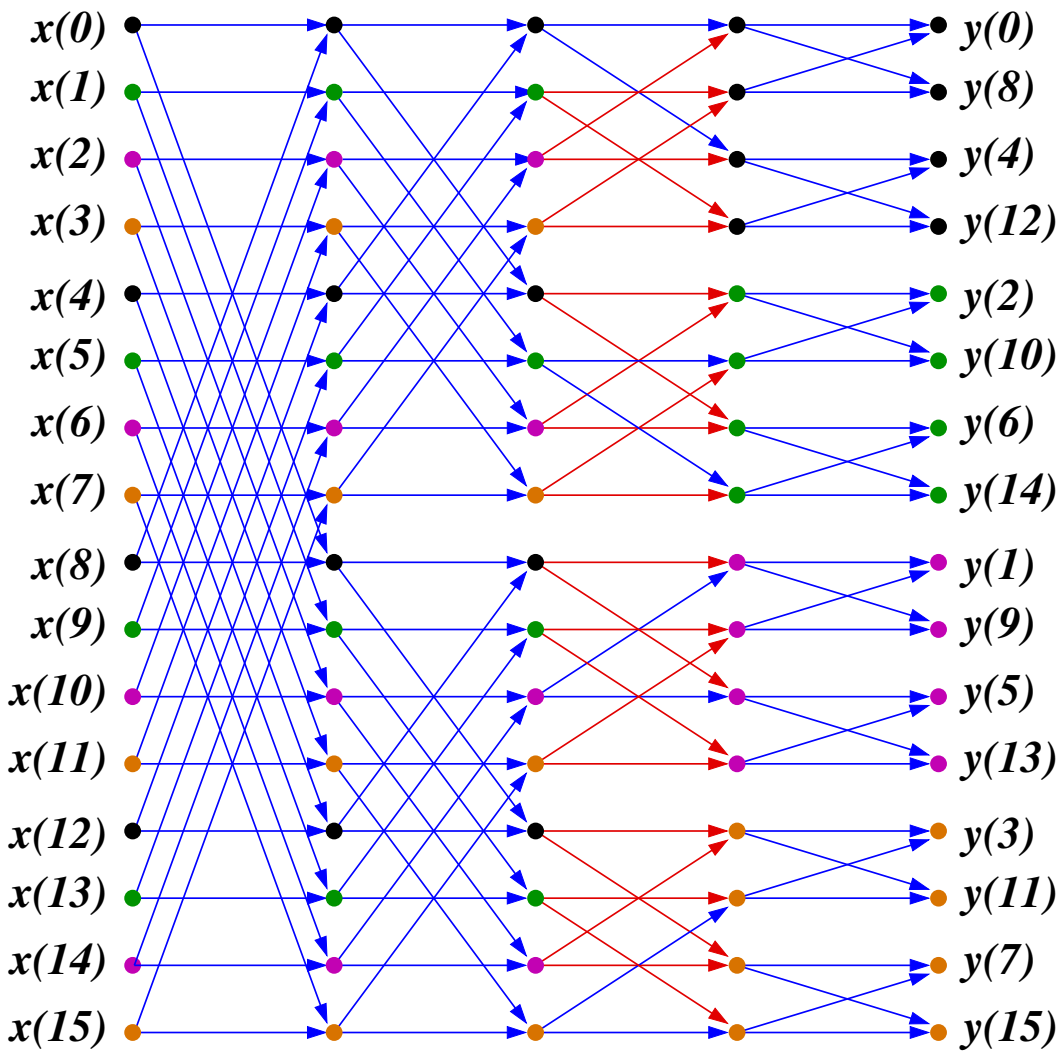
- processor 0
- processor 1
- processor 2
- processor 3

local communication

inter−processor communication

- Second attempt to parallelize:
  - interleave rows among processors
  - everything local on the left
  - lots of communication on the right.

LogP Example: FFT (4/6)



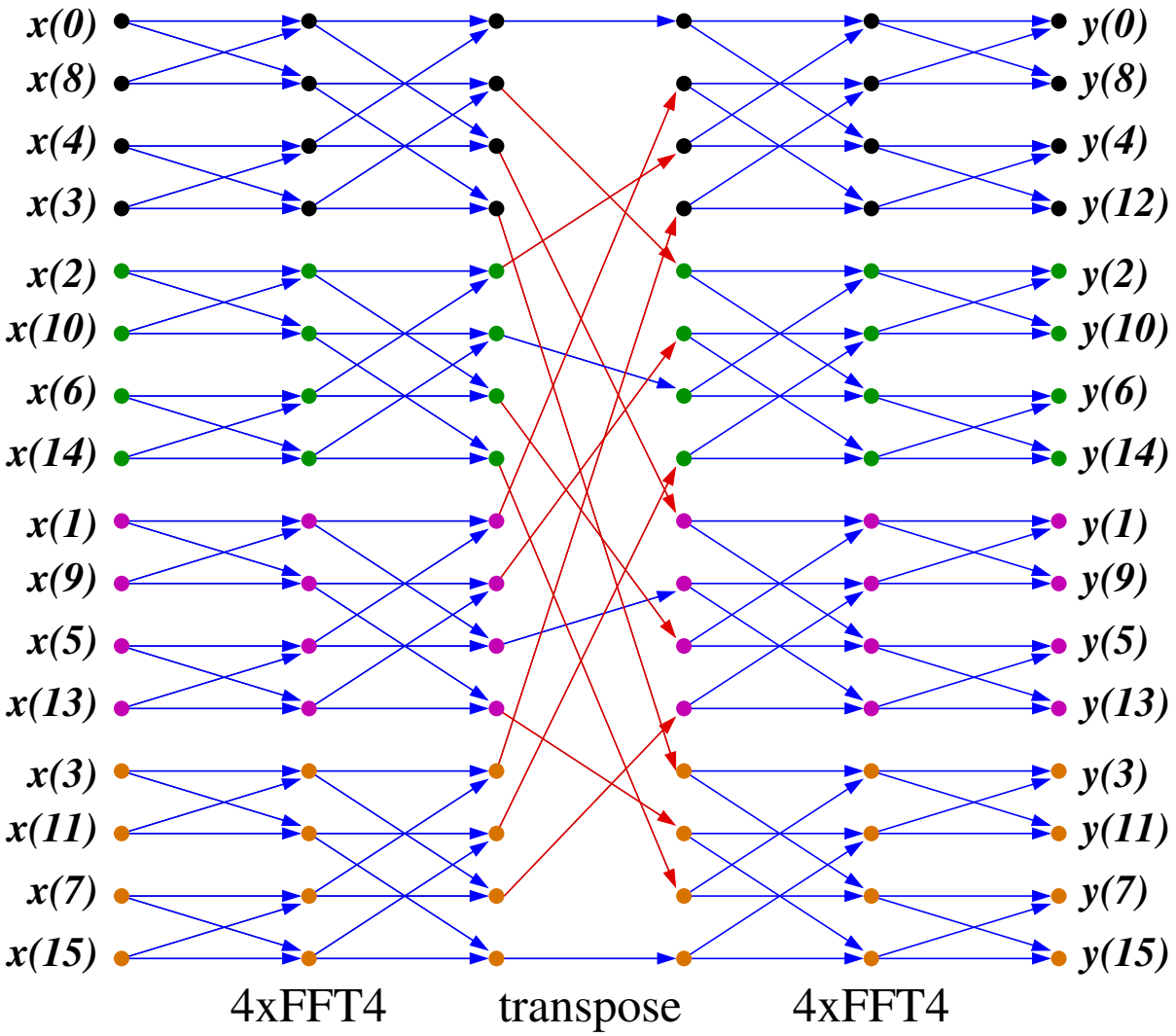
- processor 0
- processor 1
- processor 2
- processor 3

local communication

inter–processor communication

- Combined approach
  - interleave rows on the left
  - one big round of communication in the middle
  - block of rows on the right

LogP Example: FFT (5/6)



- Another view of the combined approach
  - the FFT and transpose phases drawn separately.

LogP Example: FFT (6/6)

- LogP shows that the combined approach is better.
  - So does CTA – one round of messages is clearly better than log P rounds.
  - The technique is well-known – the same approach is important to get good cache utilization.
- LogP shows that staggering messages is better than naively flooding one destination at a time.
  - So does CTA with it's assumption of bounded fan-in and fan-out of the network.
- Note: The "transpose in the middle" pattern of the FFT occurs in many other algorithms as well.
  - It's important to be able to handle this pattern efficiently.

Comparing the models

- CTA is simpler than LogP
- LogP accounts for more machine details
  - but these details don't seem essential for the examples that they give in the paper.
  - It's not clear that the extra details would account for more than a factor of 2 in time estimates,
  - and there are lots of other system details that logP ignores that can cause errors of that magnitude or larger.
  - but the marketing is better: "logP" just sounds better than CTA. ☺
- Both are based on a 10-20 year old machine model
  - That's ok, the papers are 18-25 years old.
  - Doesn't account for the heterogeneity of today's parallel computers:
    - ★ multi-core on chip, faster communication between processors on the same board than across boards, etc.
- CTA seems like a simple, and reasonable place to start.
  - But recognize the limitations of any of these models.
- Getting a model of parallel computation that's as all-purpose as the RAM is still a work-in-progress.

For further reading

- [Valiant1975] Leslie G. Valiant, "Parallelism in Comparison Problems," *SIAM Journal of Computing*, vol. 4, no. 3, pp. 348–355, (Sept. 1975).
- [Fortune1979] Steven Fortune and James Wyllie, "Parallelism in Random Access Machines," *Proceeding of the* 11^{th} *ACM Symposium on Theory of Computing* (STOC'79), pp. 114–118, May 1978.
- [Culler1993] David Culler, Richard Karp, *et al.*, "logP: towards a realistic model of parallel computation," *ACM SIGPLAN Notices*, vol. 28, no. 7, pp. 1–12, (July 1993).

Announcements and reminders

- Oct. 20: midterm
- Oct. 25: reduce and scan
  Read Lin & Snyder Chapter 5, beginning of chapter through the end of *The Reduce and Scan Abstractions* $\rightarrow$ *Generalized Vector Operations* (pp. 112-134).
- Oct. 27: work allocation
  Read the rest of Lin & Snyder Chapter 5, *Assigning Work to Processes Statically* through the end of *Chapter Summary*.

Review

- Compare and Contrast the main features of the PRAM, CTA, and LogP models?
- How does each model represent computation?
- How does each model represent communication?
- How does one determine parameter values for the CTA and LogP models? Describe at a high-level the kinds of experiments you could run to estimate the parameters.