COL380

Introduction to
Parallel & Distributed Programming

Why PRAM?

- Easy to design, specify, analyze algorithms
  - ➡ Independent of machine details
- Fidelity of predicted performance
  - Fine-grained synchronization
  - ➡ Not many surprises for shared-memory architecture
  - ➡ Only partially successful for distributed memory
  - ➡ Note that memory-access and message latency is often bounded
- Strong model
  - ➡ Possible to simulate on a wide variety of hardware
  - ➡ Poor PRAM solution often implies a hard problem

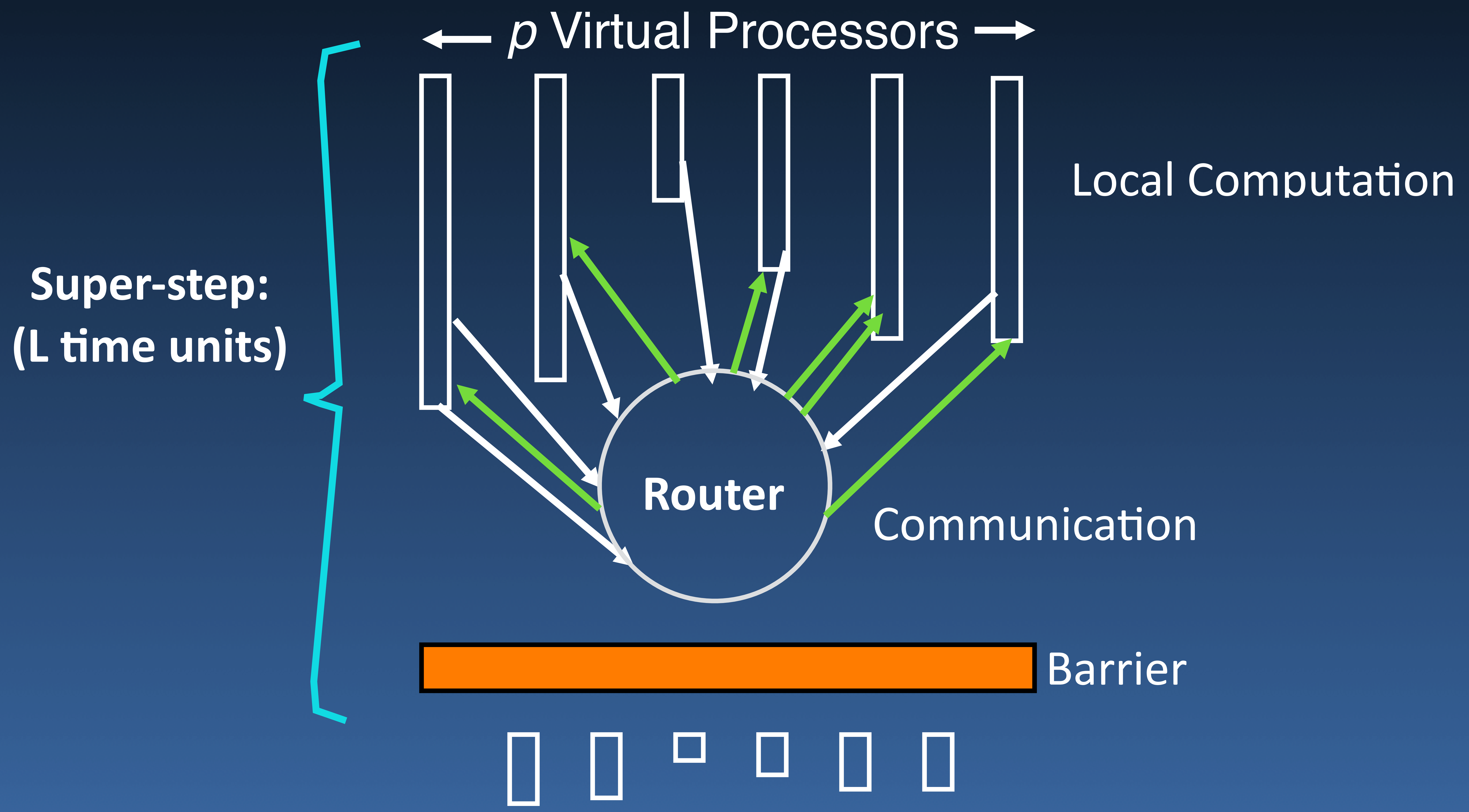
Bulk Synchronous Parallel Model

- A set of virtual (processor, memory) pairs
    - ➡ No notion of "locality" in mapping to physical processor
- A point-to-point interconnect
- Barrier synchronization (all or subset)
- Repeat "super-step":
    - ➡ Local computation
    - ➡ Communication
    - ➡ Barrier synchronization

Designed as a Bridging model for parallel programming

Subodh Kumar

BSP



Subodh Kumar

BSP Characteristics

- Steps are machine wide, similar to PRAM
    - ➡ Communication and computation
- Simple to program and analyze algorithms
    - ➡ Do not have to separately model individual communication
- “Bridge” between PRAM and real architecture
- Locality of processor or computation ignored
    - ➡ Leaves such optimization out of the equation

BSP

- In each super-step, VP does a task with
  - ➡ implicit message arrivals, local computation, message sends
  - ➡ Computation may only access data available at the beginning of super-step
- VPs synchronize at regular intervals of L time units
  - ➡ L: the periodicity parameter
    - L may be a constant
    - If a VP not done with in L, the super-step continues (a multiple of L)

Interconnect

- Realize arbitrary h-relation in a super-step:
  - ➡ VPs send (receive) at most h messages
    - each of length “1”
  - ➡ Communication cost: $gh = th + s$
    - t: network latency
    - s: constant overhead
  - ➡ Ignores actual transfer size, distribution

BSP cost Accounting

- Barrier cost of *l*
  - May depend on the count of virtual processors, p
  - A lower bound on l is the diameter of the network
- Super-step cost
  - cost of the longest running local computation
  - cost of communication
  - cost of the barrier synchronization
  - $w + hg + l$ (Non Overlap model)
- Total cost = summation over super-steps s:
  - $\sum w_s + g \sum h_s + Nl$ (N is the number of super-steps)

Subodh Kumar

Complexity Class NC

- $t\,(n,p(n)) = O(\log_{k1} n)$ with $p(n)=O(n^{k2})$ processors
  - ➡ $k1, k2 = O(1)$
  - ➡ Assume PRAM model (Doesn't matter which)
  - ➡ Robust with respect to other models as well

Is parallel "p-ary" search in NC?

- Many NC algorithms are work optimal
  - ➡ e.g., for a linear-time sequential algorithm:
    - $t\,(n,p(n)) = O(\log n)$ with $p(n)=n/\log n$
    - $t\,(n,p(n)) = O(\log \log n)$ with $p(n)=n/\log \log n$

$NC \subseteq P$

Open Question: is P == NC?

- "Inherently sequential," if no NC algorithms exist for a problem