Unit 2

lec 10-20

PRAm is gud cuz its easy to design algorithms, analyse.

Work Time Scheduling Principle

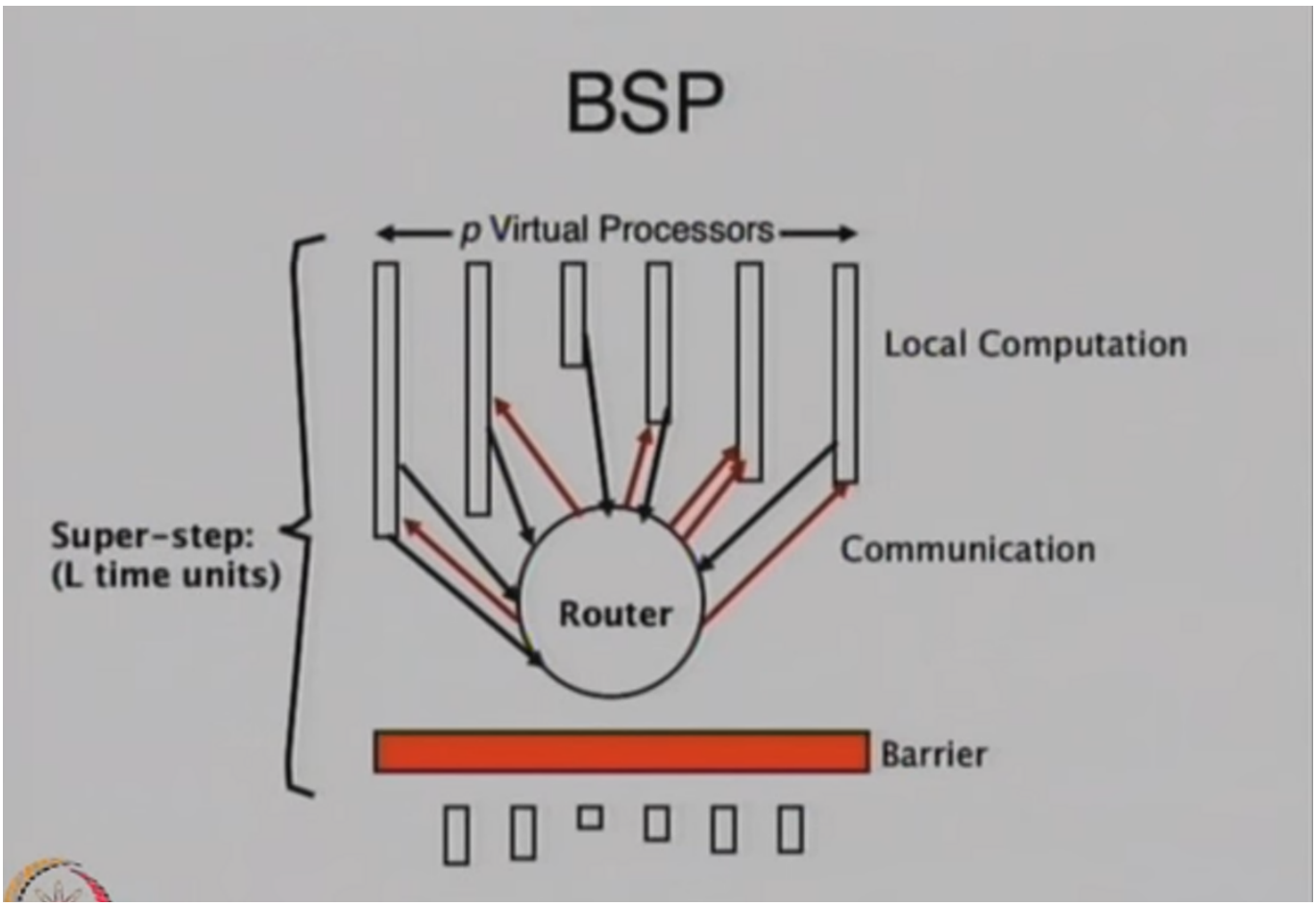
This essentially says do not care about the actual number of processors present. Make a algorithm that evaluates in t(n) steps using p processors and let a auto scheduler manage the algorithm in the bare metal.

Cost optimal algorithm is automatically work optimal (this is compared to sequential).

Bulk Synchronous Parallel Model

This says that there are virtual pairs of processor and memory. There is not local mapping but instead network memory. It operates by executing "super step":

- local computation
- communication
- barrier synchronization



BSP has system wide steps, like pram. It is simple to analyze. It connects well to physical architecture.

Calculating cost for this is a little more complex.

- barrier cost of l (barrier sync)
- superstep cost: local computation + communication + barrier ==> $w + hg + l$ g is thruput, h is message something
- $Total/cost = \sum w_s + g \sum h_s + Sl$

LogP Model

btw this isn't math log. This is l,g,p from above. wat is o? no-one knows.

Let's say we are doing a regular search. N elements and p processors. This will be technically faster than regular binary search **but** WORK DONE is way higher.

Memory Consistency

Cache Coherency

This is L1 cache of a thread making sure to write to L2, or whatever is the shared memory, to maintain that data's "freshness" or coherency. This makes sure that all threads work with proper updated data.

Snoopy cache coherence protocol is literally wiretapping.

Strict Consistency

- read(x) must return the latest write(x). It should also appear as if operations are instantaneous.
- linearizability is a weaker version. It has a point for sync called linearization point. It is instantaneous here.
- Impractical: wth is this? It said it is done for uniuser, so its just nothing happening? o_o

Sequentially Consistent Model

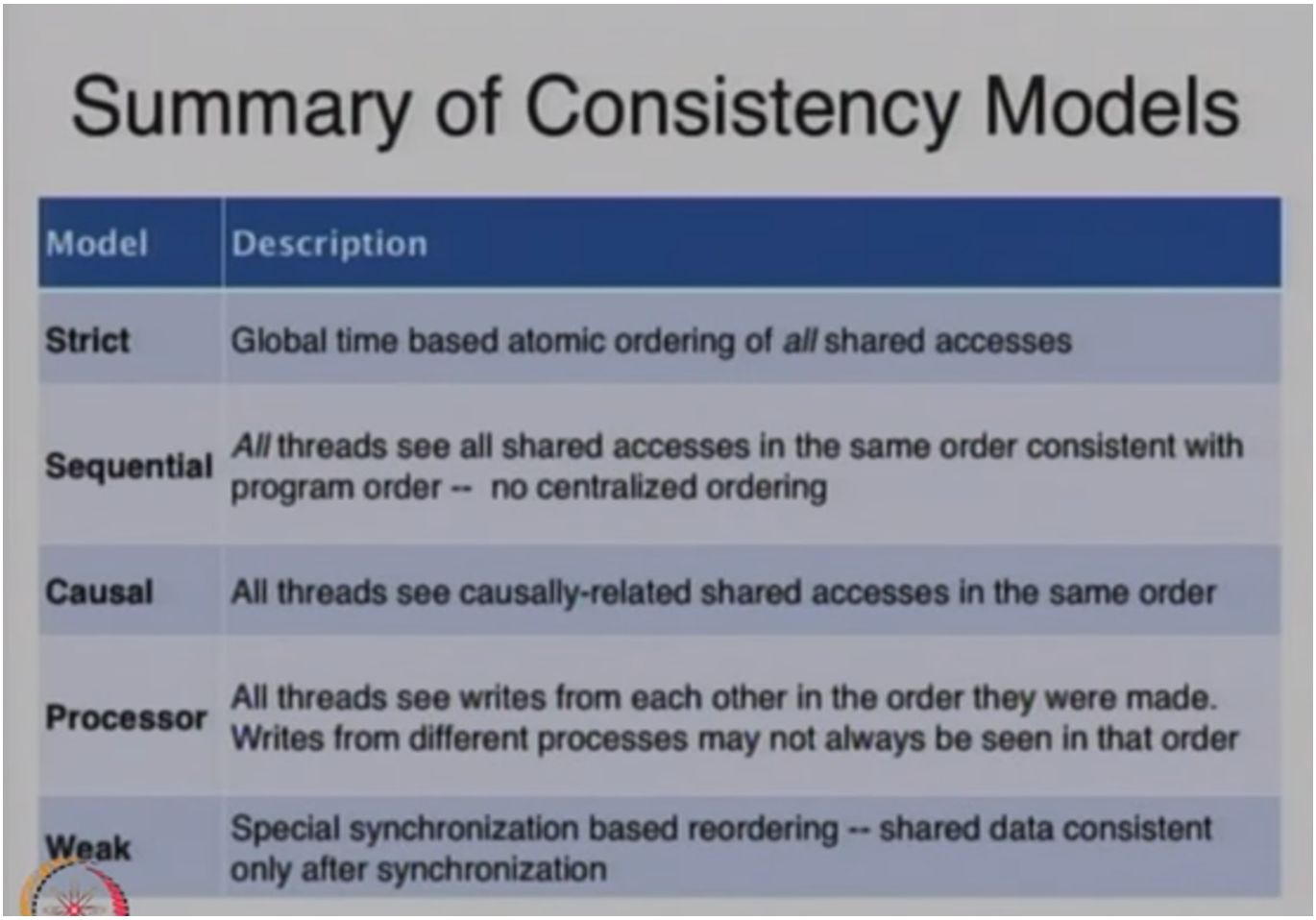
This is where the memory is consistent to how the memory would have been if the program was sequential. This is hard to implement.

Processor Consistency

All processes must see memory writes from one process in the order they were issues. This is FIFO consistency. Processor consistency is when we say all processes see memory writes to a variable (data) must be seen in the order of issue.

Weak Consistency

Consistency is only enforced on request. OpenMP uses this.



Summary of Consistency Models

Model	Description
Strict	Global time based atomic ordering of *all* shared accesses
Sequential	*All* threads see all shared accesses in the same order consistent with program order -- no centralized ordering
Causal	All threads see causally-related shared accesses in the same order
Processor	All threads see writes from each other in the order they were made. Writes from different processes may not always be seen in that order
Weak	Special synchronization based reordering -- shared data consistent only after synchronization

Performance issues

- True sharing:
    - make copies for each processor
- False sharing
    - give continuous blocks💀. These blocks have the variables. In this we send lines of memory. So we get some useless stuff.

There are other things like make code short, use as many threads as required etc.