Multithreading - Architecture Environment and Computations

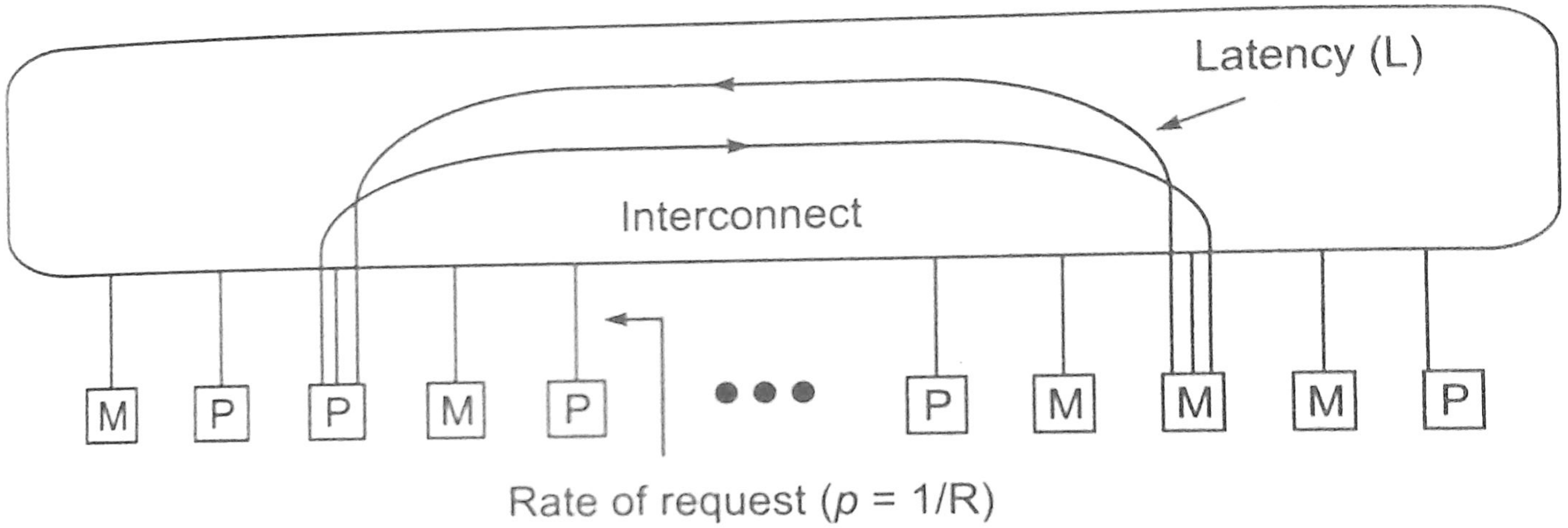
Submitted By

Manu John
CSE S7, 41

Architecture Environment

One of the possible multithreaded MPP system is modeled by network of processor(p) and memory(p). The distributed memories form a global address space. Four machine parameter are defined below

- The Latency (L): Communication latency on a remote memory access. The value of L includes the network delay, cache-miss penalty, and delays caused by contentions in split transaction
- The number of threads (N): Number of process that can be interleaved in each processor
- The context-switching overhead (C): The cycles lost performing context switching in a processor
- The interval between switches (R): The cycle between switches triggered by remote reference. The inverse p=1/R the rate of requests



(a) The architecture environment. (Courtesy of Rafael Saavedra, 1992)

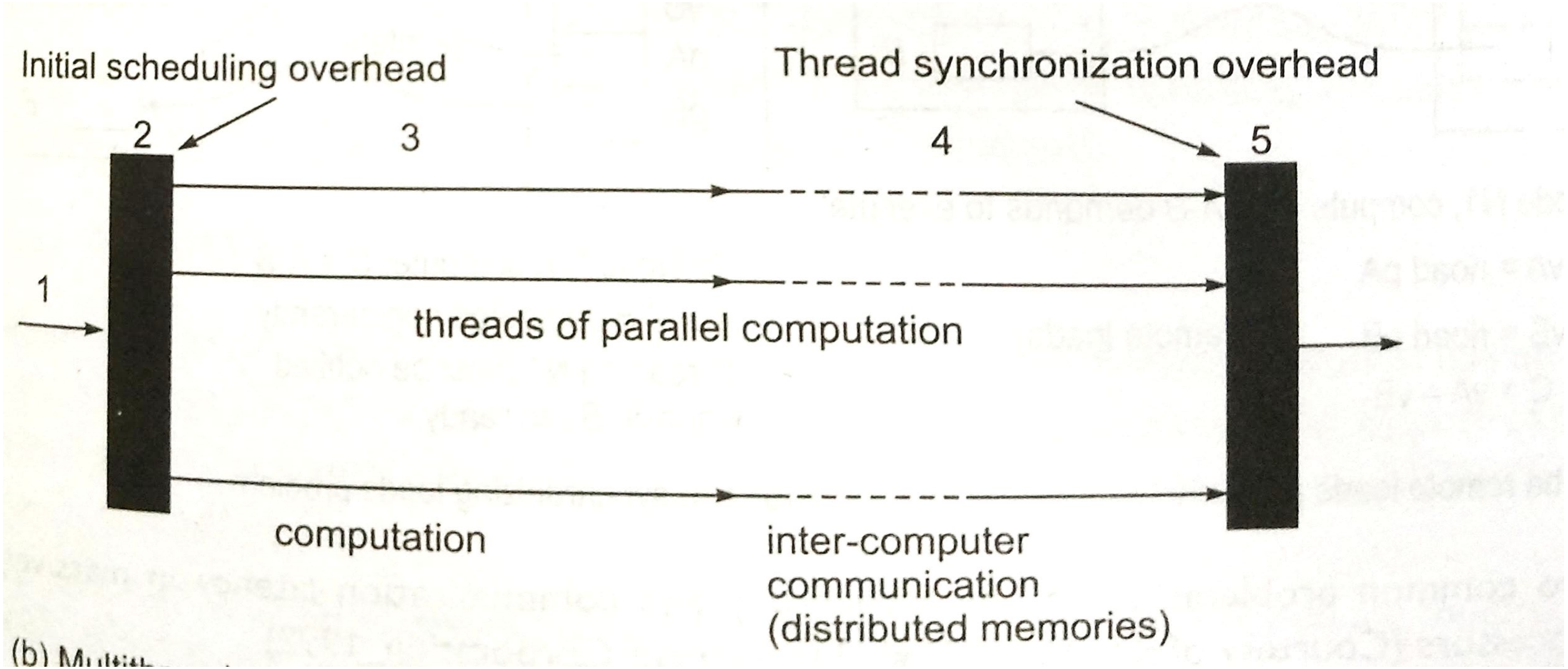
In order to increase efficiency,

- One approach is to reduce the rate of requests by using distributed coherent caches
- Eliminate processor waiting through multithreading

Multithreaded Computations

- The computation model starts with a sequential thread
- Followed by supervisory scheduling
- Where the processors begin threads of computation
- Intercomputer messages that update variable among the nodes when the computer has a distributed memory
- Finally by synchronization prior to beginning the next unit of parallel work

Multithreaded Computations



Initial scheduling overhead
Thread synchronization overhead
2
3
4
5
1
threads of parallel computation
computation
inter-computer communication (distributed memories)
(b) Multith

Relaxed Memory Consistency - Processor and Release Consistency

Liya Yohannan,S7 CS

Relaxed Memory Models

- Relaxed memory models do not make programming any more complex(many would disagree however).However, you need to follow additional rules about how to synchronize threads.C11, Pthreads and Java are specified for relaxed memory models.
- In relaxed memory models, both the program order of memory references and the write atomicity requirements are removed and are replaced with different rules.
- For compiler writers and computer architects this means that more optimizations are permitted.
- For programmers it means two things:
    - you must protect data with special system recognized synchronization primitives, e.g. locks, instead of normal variables used as flags.
    - your code will almost certainly become faster, perhaps by between 10 and 20 percent, due to eliminating the write access penalty.

Relaxed Memory Models

- Relaxed memory models relax the two constraints of memory accesses: program order and write atomicity.
- We have the following possibilities of relaxing SC.
- Relaxing A to B program order: we permit execution of B before A.
    - write to read program order to different addresses
    - write to write program order to different addresses
    - read to write program order to different addresses
    - read to read program order to different addresses
    - read other CPU’s write early
    - read own write early
- Different relaxed memory models permit different subsets of these.

Assumptions

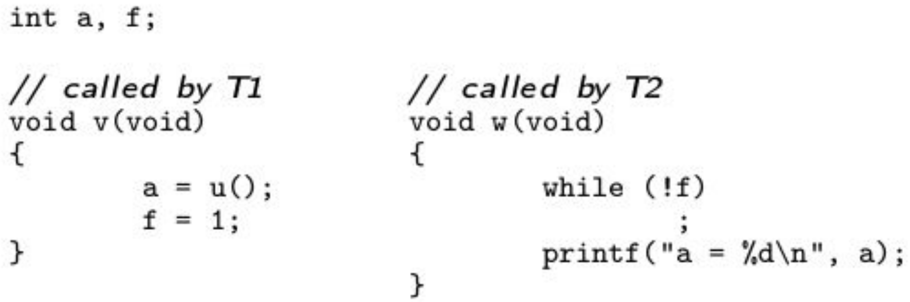
- All writes will eventually be visible to all CPUs.
- All writes are serialized (recall: this can be done at the memory by letting one write be handled at a time — other pending writes must be retried).
- Uniprocessor data and control dependences are enforced.

Relaxing the Write to Read Program Order Constraint

-A read may be executed before a preceding write has completed.

Additionally Relaxing the Write to Write Program Order Constraint

-Consider the program below



```
int a, f;

// called by T1              // called by T2
void v(void)                 void w(void)
{                            {
        a = u();                     while (!f)
        f = 1;                               ;
}                                    printf("a = %d\n", a);
                             }
```

-By relaxing the write to write program order constraint, the write to f may be executed by T 1 even before the function call to u, resulting in somewhat unexpected output.

- Machines with relaxed memory models have special machine instructions for synchronization.
- Consider a machine with a sync instruction with the following semantics:
- When executed, all memory access instructions issued before the sync must complete before the sync may complete.
- All memory access instructions issued after the sync must wait (i.e. not execute) until the sync has completed.
- The memory consistency model introduced with the sync instruction is called **Weak Ordering**

Release Consistency

- Release Consistency, RC, is an extension to WO, and was invented for the Stanford DASH research project.
- Two different synchronization operations are identified.
    - An acquire at a lock.
    - A release at an unlock.
- An acquire orders all subsequent memory accesses, i.e. no read orwrite is allowed to execute before the acquire. Neither the compilernor the hardware may move the access to before the acquire, and all acknowledged invalidations that have arrived before the acquire mustbe applied to the cache before the acquire can complete (i.e. leave the pipeline).
- A release orders all previous memory accesses.The processor must wait for all reads to have been performed (i.e. the write which produced the value read must have been acknowledged by all CPUs) and all writes made by itself must be acknowledged before the release can complete.

WO vs RC

- Recall: by A → B we mean B may execute before A
- WO relaxation: data → data
- RC relaxation:
  - data → data
  - data → acquire
  - release → data
  - release → acquire
  - acquire → data: forbidden
  - data → release: forbidden

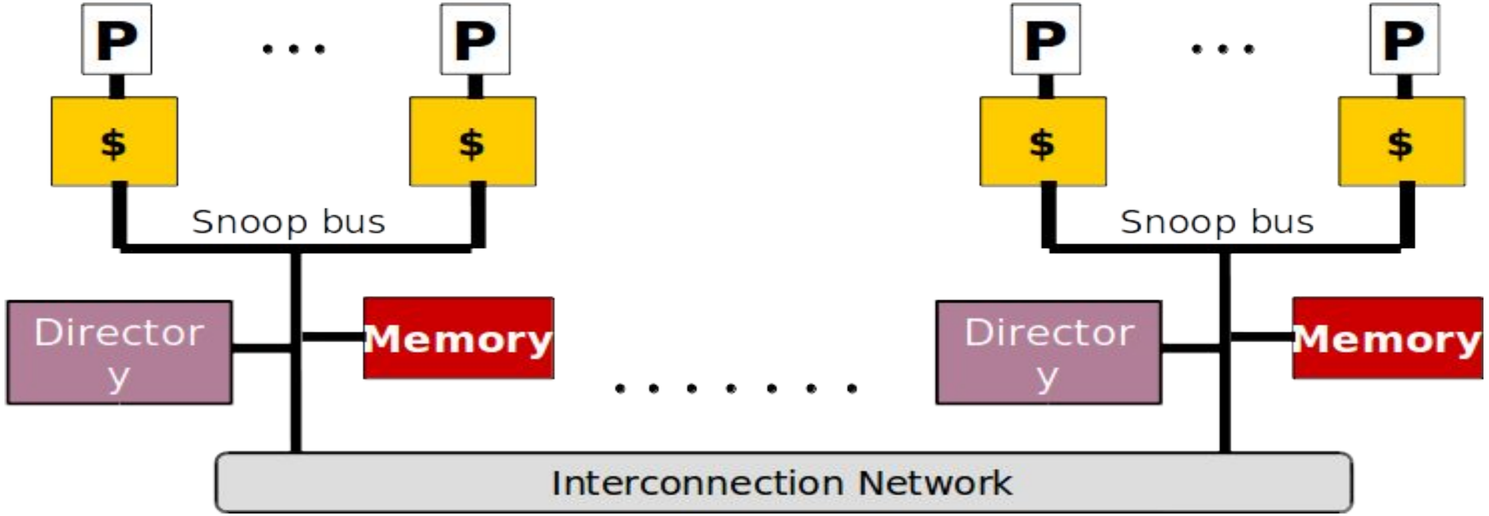
Stanford Dash Multiprocessor - Data Memory Hierarchy and Directory Protocol

Submitted by ,

Vimal P Viswan
S7 CSE

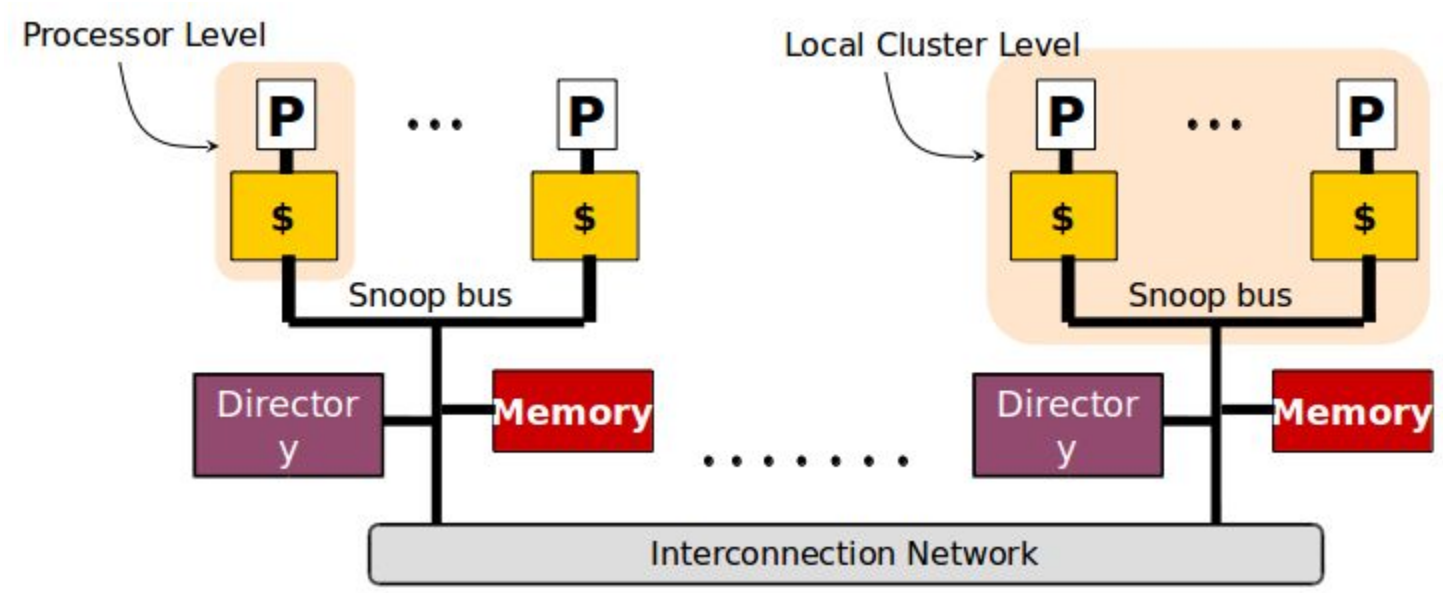
Stanford DASH multiprocessor

- The Computer Systems Laboratory at Stanford University is developing a shared-memory multiprocessor called Dash(an abbreviation for Directory Architecture for Shared Memory)
- .The fundamental premise behind the architecture is that it is possible to build a scalable high-performance machine with a single address space and coherent caches.


P
...
P
P
...
P
$
$
$
$
Snoop bus
Snoop bus
Director
y
Memory
Director
y
Memory
....
Interconnection Network

- Stanford DASH: 4 CPUs in each cluster, total 16 clusters (1992)
  - Invalidation-based cache coherence
  - Directory keeps one of the 3 status of a cache block at its home node
    - Uncached
    - Shared (unmodified state)
    - Dirty

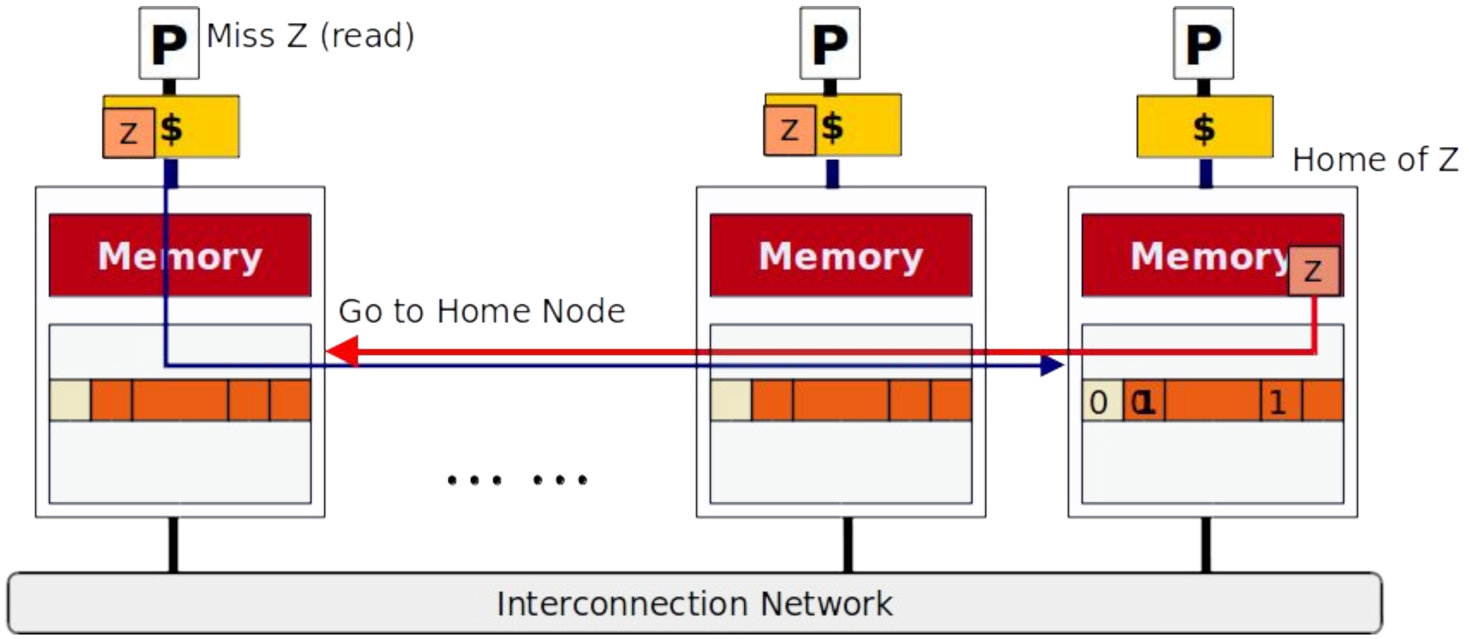
Data Memory Hierarchy



Hierarchy

- Processor Level
- Local Cluster Level
- Home Cluster Level (address is at home)
    - If dirty, needs to get it from remote node which owns it
- Remote Cluster Level

Directory Coherence Protocol


P
Miss Z (read)
Z
$
P
Z
$
P
$
Home of Z
Memory
Memory
Memory
Z
Go to Home Node
0
1
1
....
Interconnection Network