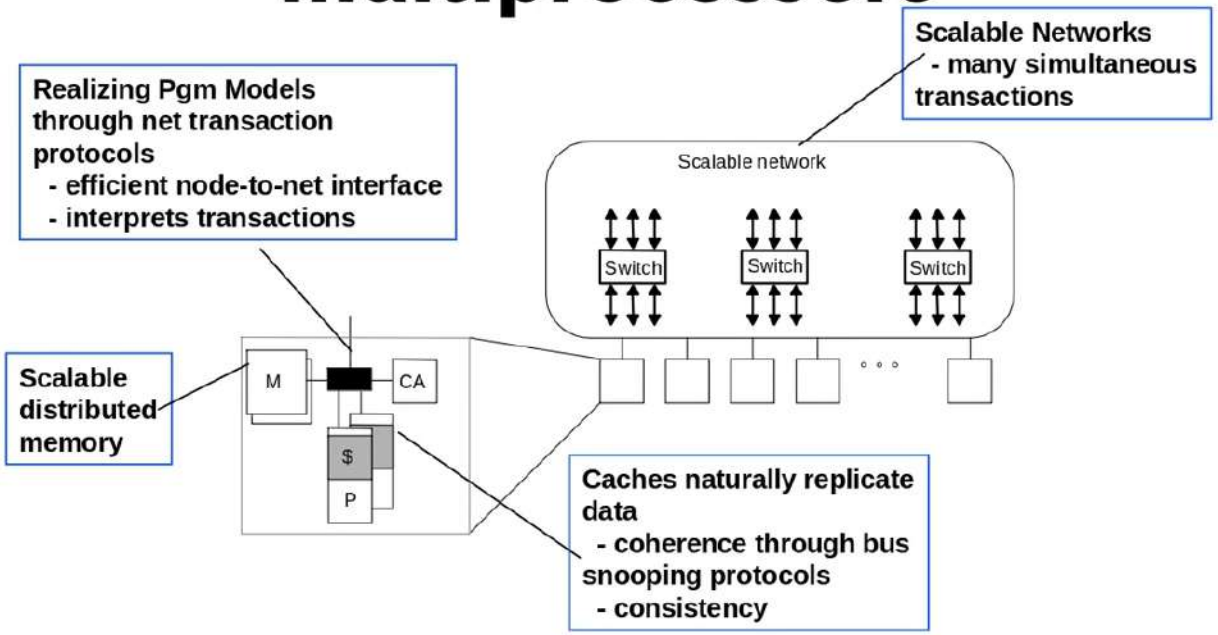
Directory based Cache Coherence

Topic-4
Chapter-8

Why snooping does not scale?

- Limitations of snooping
  - Broadcasting uses lot of bus bandwidth
  - Snooping every transaction uses lots of controller bandwidth
- Snooping is great for small to medium size machines
  - Largest current snooping system has 128 processors
- Snooping hits bottleneck for large machine
- How to overcome bottleneck?
  - Get rid of protocol that requires: Broadcasting / Logical bus

Large scale shared memory multiprocessors



- 100s to 1000s of nodes with single shared physical address space
- General purpose interconnection network
  - Still have cache coherence protocol
  - Use message instead of bus transaction
  - no broadcast or single point of order

Bus based coherence

- All (a), (b), (c) done through broadcast on bus
  - Faulting processor sends out a search
  - Others respond to search and take necessary action
- Could do it in scalable network too
  - Broadcast to all processors and let them respond
- Conceptually simple, but broadcast does not scale with p
  - On bus, bandwidth does not scale
  - On scalable network, every fault leads to atleast p network transactions
- Scalable coherence:
  - Can have same cache state and state transition diagram
  - Different mechanism to manage protocol

One Approach: Hierarchical Snooping

- Extend snooping approach. Have a hierarchy of broadcast media
  - Tree of bus or rings
  - Processors are in bus or ring based multiprocessors at the leaves
  - Snoop both busses and propagate relevant transactions
  - Main memory can be centralised at root or distributed among leaves
- Issues (a)-(c) handled similar to bus, but not full broadcast
  - Faulting processor sends out search on the bus
  - Propagates up and down the hierarchy based on snoop results
- Hope is that most of the time request is not propagated very far
- Problems:
  - High latency: multiple levels and snoop/lookup at every level
  - Bandwidth bottleneck at root
- Not popular today

27_Sep_2021 - Goog

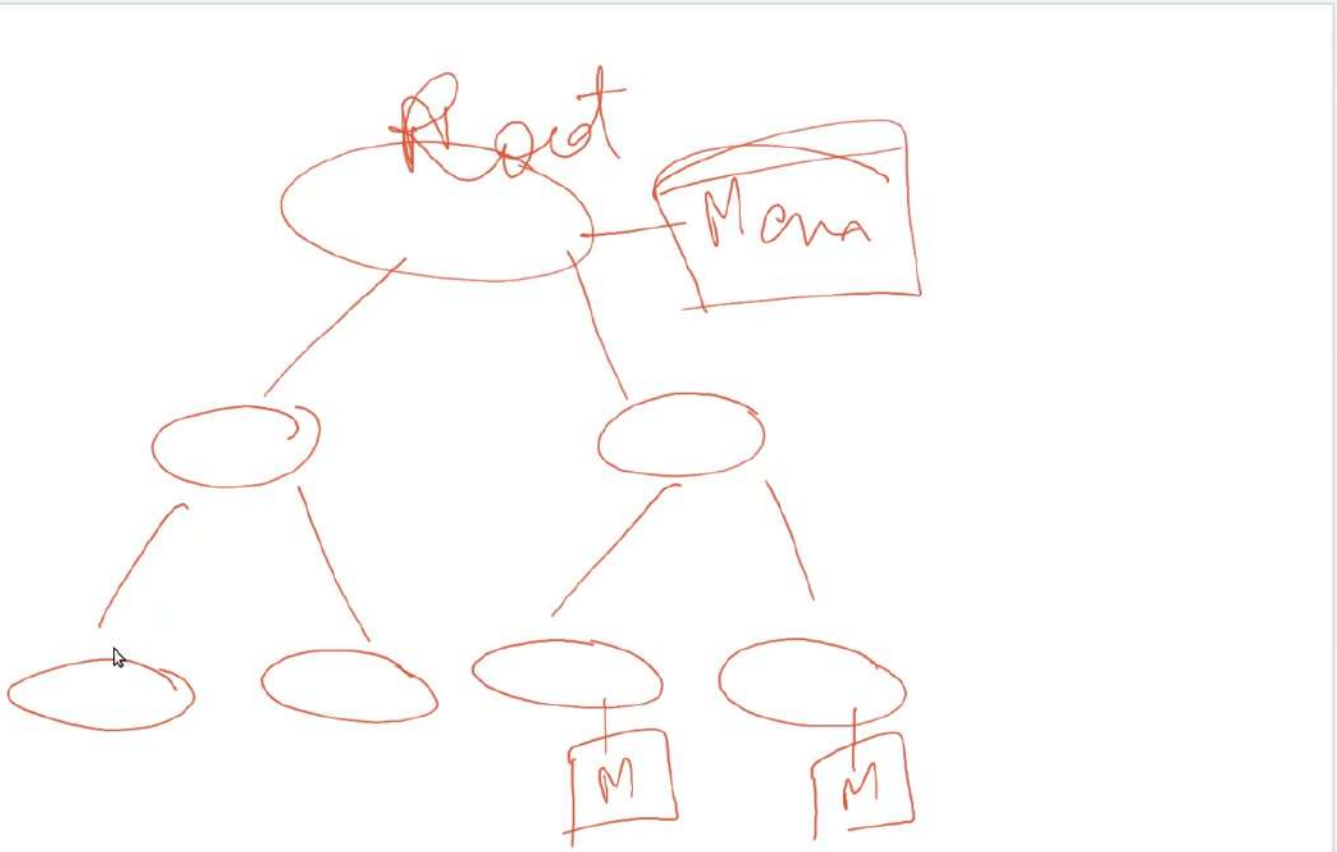
jamboard.google.com/d/1AIogDeB2COtifaU930L4zY8balv-UOA_HbR3-LKr-L4/viewer?f=2

27_Sep_2021

3 / 3

Share

Set background | Clear frame



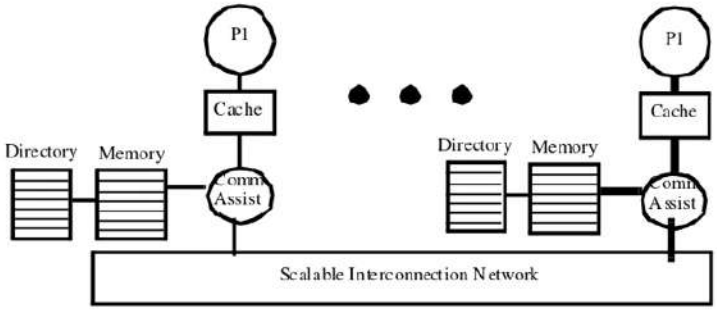
88% (3:39) Thu, September 30, 14:07

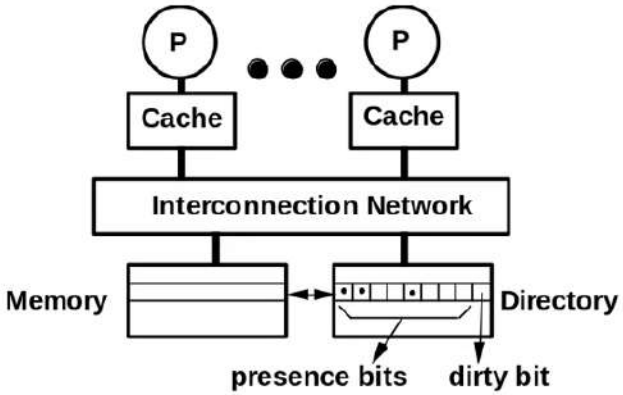
+4 | A | VK | SM | TANVISH | VP | VATSHAL NILESHKUMAR PATEL | AS | ASWATHY N S | Syam Sankar | YK | YOGESH KUMAR | SUBRATA ROY | Hemangee Kalpesh Kapoor

Scalable solution: Directories

- Avoid broadcast requests to all nodes on a miss
- Every memory block has associated directory information
  - Keeps track of copies of cached block and their states
  - On a miss, find directory entry, send message to cached copies if necessary
  - In scalable network, communication with directory is through network transactions
- Maintain directory of which nodes have cached copies of the block (directory controller + directory state)
  - On a cache miss, cache controller sends message to directory
  - Directory determines what (if any) protocol action is required – e.g. inv of shared nodes
  - Directory waits for protocol actions to finish and then responds to the original request

Generic solution: Directories





Basic operation of directory

- K-processors
- With each cache-block in memory: k-presence bits, 1-dirty bit
- With each cache block: 1 valid-bit, 1 dirty (owner) bit, state-bit
- Read from main memory by processor-i
  - If dirty-bit = OFF then { read from memory ; turn p[i] = 1 }
  - If dirty-bit = ON then { recall block from dirty processors' cache, make state shared 'S' ; update memory, dirty-bit=OFF ; supply recalled data to proc-i; turn p[i]=1 }
- Write to memory by processor-i
  - If dirty-bit OFF then { supply data to i ; send inv to all caches (j) with p[j]=1 then p[j]=0 ; dirty-bit=ON ; p[i]=ON ;}
  - If dirty-bit = ON then { recall block from dirty processors' cache, make state shared 'I' ; dirty-bit=ON ; p[i]=ON, p[earlier node]=OFF ; supply recalled data to proc-i }
- Block replacement by cache
  - Dirty – write-back. Turn OFF dirty bit, p[i]=OFF
  - Clean: may/may-not tell directory

27_Sep_2021 - Goog

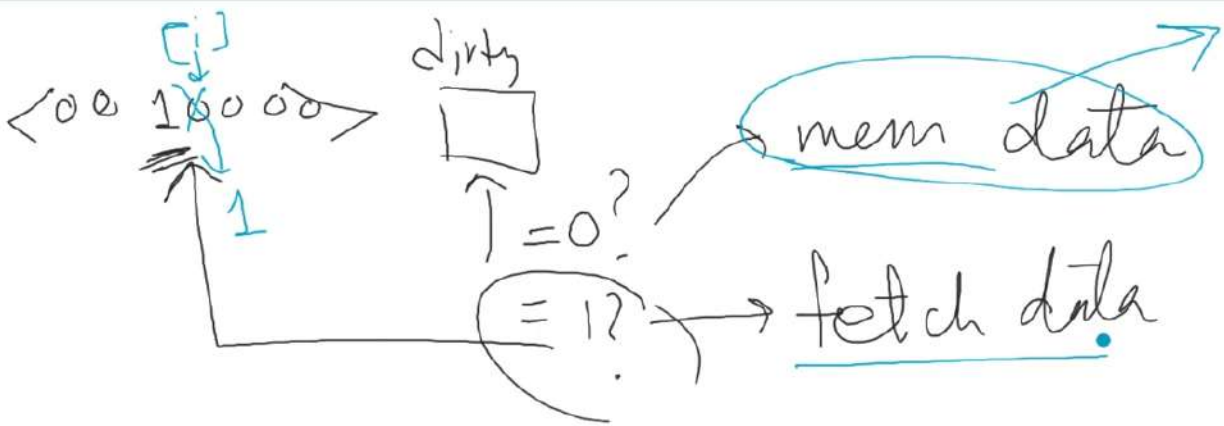
jamboard.google.com/d/1AIogDeB2COtifaU930L4zY8balv-UOA_HbR3-LKr-L4/viewer?f=3

27_Sep_2021

4 / 4

Share

Set background | Clear frame



84% (3:24) Thu, September 30, 14:16

+4 | A | VK | SM | TANVISH | VP VATSHAL NILESHKUMAR PATEL | AS ASWATHY N S | Syam Sankar | YK YOGESH KUMAR | SUBRATA ROY | Hemangee Kalpesh Kapoor

Value of Directories

- Value of directory is that they keep track of which nodes have copies of the block, eliminating the need for a broadcast
- Valuable on Read Miss
  - Request satisfied by the memory
  - Or the directory will tell it exactly where to got to retrieve the exclusive copy of data
- Valuable on Write Miss
  - Value over broadcast method is the highest if the number of sharers to which invalidations have to be sent is small
  - This number of sharers does not scale up quickly with the number of processor nodes
- Empirical measurements show that the number of sharers is small, the number of sharers does not go up quickly as the processor nodes increase and the frequency of writes that generate invalidations is also small
- These fact make directories as a scalable approach and also help in organising the directory in a cost-effective manner

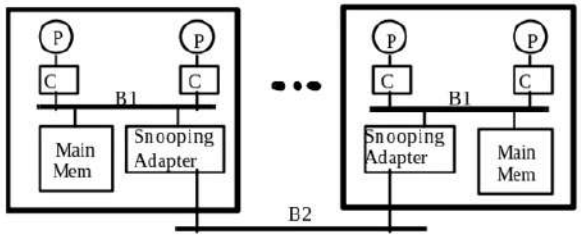
Value of Directories

- Value of directory is that they keep track of which nodes have copies of the block, eliminating the need for a broadcast
- Valuable on Read Miss
  - Request satisfied by the memory
  - Or the directory will tell it exactly where to got to retrieve the exclusive copy of data
- Valuable on Write Miss
  - Value over broadcast method is the highest if the number of sharers to which invalidations have to be sent is small
  - This number of sharers does not scale up quickly with the number of processor nodes
- Empirical measurements show that the number of sharers is small, the number of sharers does not go up quickly as the processor nodes increase and the frequency of writes that generate invalidations is also small
- These fact make directories as a scalable approach and also help in organising the directory in a cost-effective manner

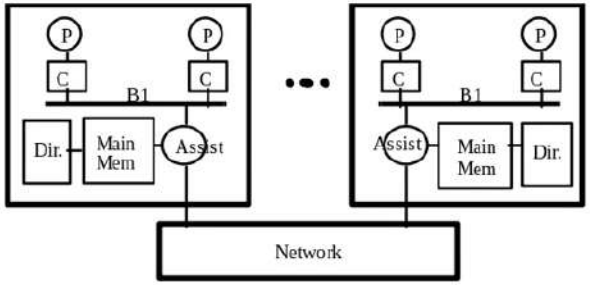
A popular middle ground

- Two-level hierarchy
- Individual nodes are multiprocessors
  - Caches in this node kept coherent by the inner protocol
- Coherence across nodes is directory based maintained by using an outer protocol
  - For outer protocol, each multiprocessor nodes is treated as a single cache
- Directory keeps track of nodes and not individual processors
  - Each node is represented by a communication assist or a tertiary cache
- Coherence within node is snooping or directory
  - Needs good interface of functionality
- Ex:
  - Snooping – snooping : Bus – Bus
  - Snooping – directory : Bus - n/w
  - Directory – snooping : n/w - Bus
  - Directory – directory : n/w - n/w

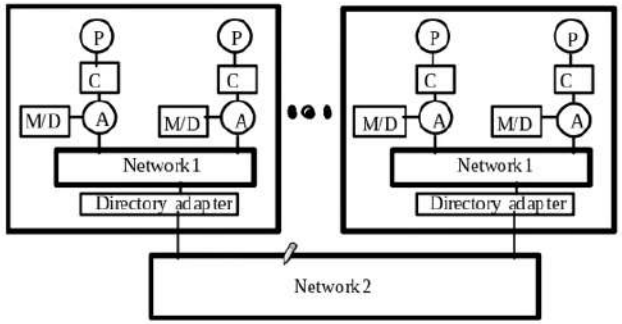
Examples: 2-level hierarchies



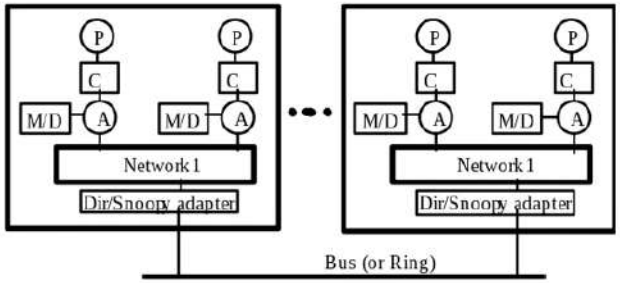
(a) Snooping-snooping



(b) Snooping-directory



(c) Directory-directory



(d) Directory-snooping

27_Sep_2021 - Goog

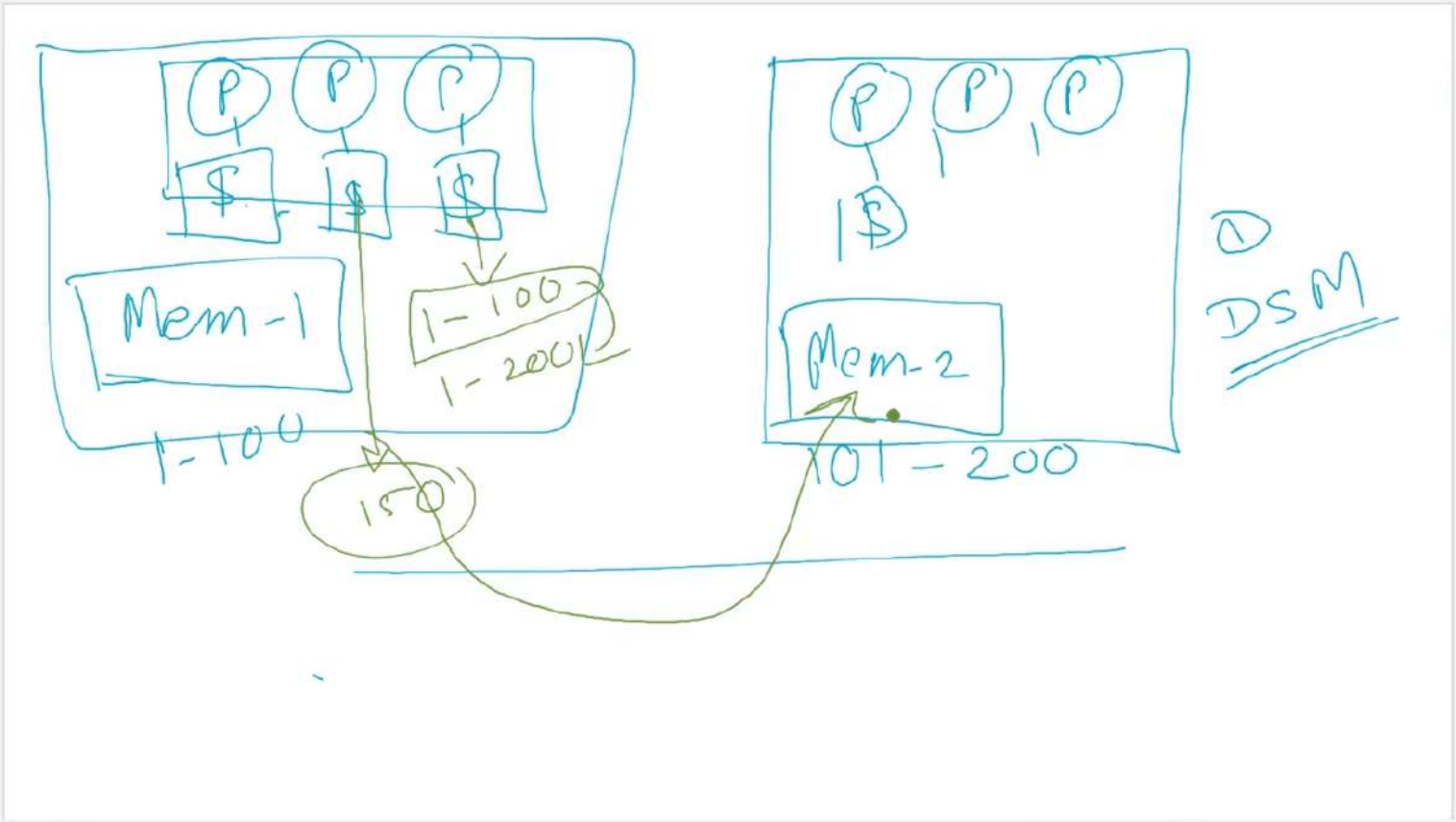
jamboard.google.com/d/1AlogDeB2COtifaU930L4zY8balv-UOA_HbR3-LKr-L4/viewer?f=4

27_Sep_2021

5/5

Share

Set background | Clear frame



74% (2:14) Thu, September 30, 14:34

+8 | A | VP | SM | NS NALABOLU SANDEEP | TANVISH | AS ASWATHY N S | Syam Sankar | YK YOGESH KUMAR | SUBRATA ROY | Hemangee Kalpesh Kapoor

27_Sep_2021 - Goog

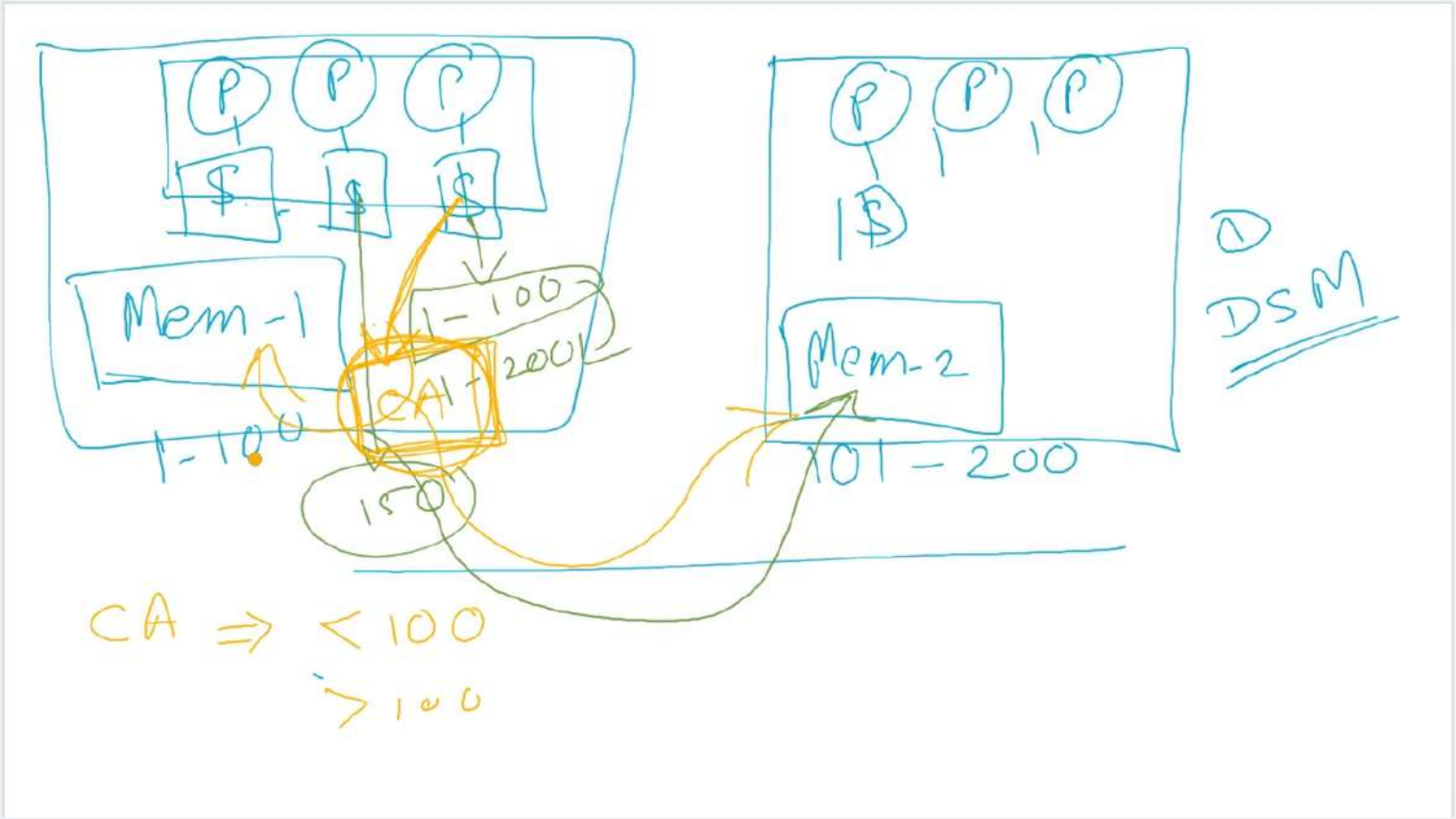
jamboard.google.com/d/1AlogDeB2COtifaU930L4zY8balv-UOA_HbR3-LKr-L4/viewer?f=4

27_Sep_2021

5/5

Share

Set background | Clear frame



74% (2:14) Thu, September 30, 14:35

+8 | A | VP | SM | NS NALABOLU SANDEEP | TANVISH | AS ASWATHY N S | Syam Sankar | YK YOGESH KUMAR | SUBRATA ROY | Hemangee Kalpesh Kapoor

Advantages of multiprocessor nodes

- Potential for cost and performance advantages
- Amortization of fixed per-node costs over multiple processors
  - Applies even if processors simply packaged together but not coherent
- Can use commodity SMPs
- Less nodes for directory to keep track of
- Much communication may be contained within node (cheaper)
- Nodes prefetch data for each other (fewer "remote" misses)
- Combining of requests (like hierarchical, only two-level)
- Can even share caches (overlapping of working sets)
- Benefits depend on sharing pattern (and mapping)
  - good for widely read-shared: e.g. tree data in Barnes-Hut
  - good for nearest-neighbor, if properly mapped
  - not so good for all-to-all communication

Advantages of multiprocessor nodes

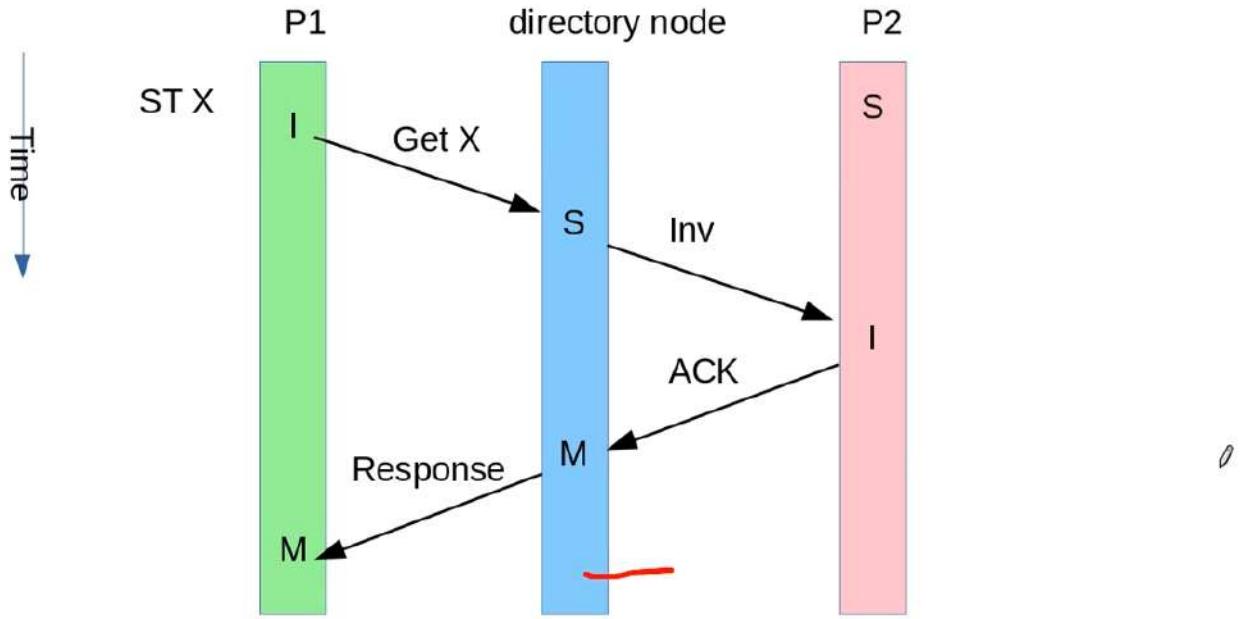
- Potential for cost and performance advantages
- Amortization of fixed per-node costs over multiple processors
  - Applies even if processors simply packaged together but not coherent
- Can use commodity SMPs
- Less nodes for directory to keep track of
- Much communication may be contained within node (cheaper)
- Nodes prefetch data for each other (fewer "remote" misses)
- Combining of requests (like hierarchical, only two-level)
- Can even share caches (overlapping of working sets)
- Benefits depend on sharing pattern (and mapping)
  - good for widely read-shared: e.g. tree data in Barnes-Hut
  - good for nearest-neighbor, if properly mapped
  - not so good for all-to-all communication

Dis-Advantages of coherent multiprocessor nodes

- Bandwidth shared among nodes
- Bus increases latency to local memory
- With coherence, typically wait for local snoop results before sending remote requests
- Snoopy bus at remote node increases delays there too, increasing latency and reducing bandwidth
- May hurt performance if sharing patterns don't comply

New serialisation point

- Directory is the new serialisation point instead of the bus



Directory types

- Centralised
  - Single directory that contains a copy of all nodes' cache tag
  - Disadvantage
    - Bottleneck (1000s proc)
    - Directory structure changes with number of nodes
  - Advantage
    - Only send inv/update to those nodes that have copy of the block
- Flat, Memory-based Distributed
  - Distribute directory among memory modules
  - Maintain directory for each memory block
    - Block's home node = node with directory information for that block

27_Sep_2021 - Goog

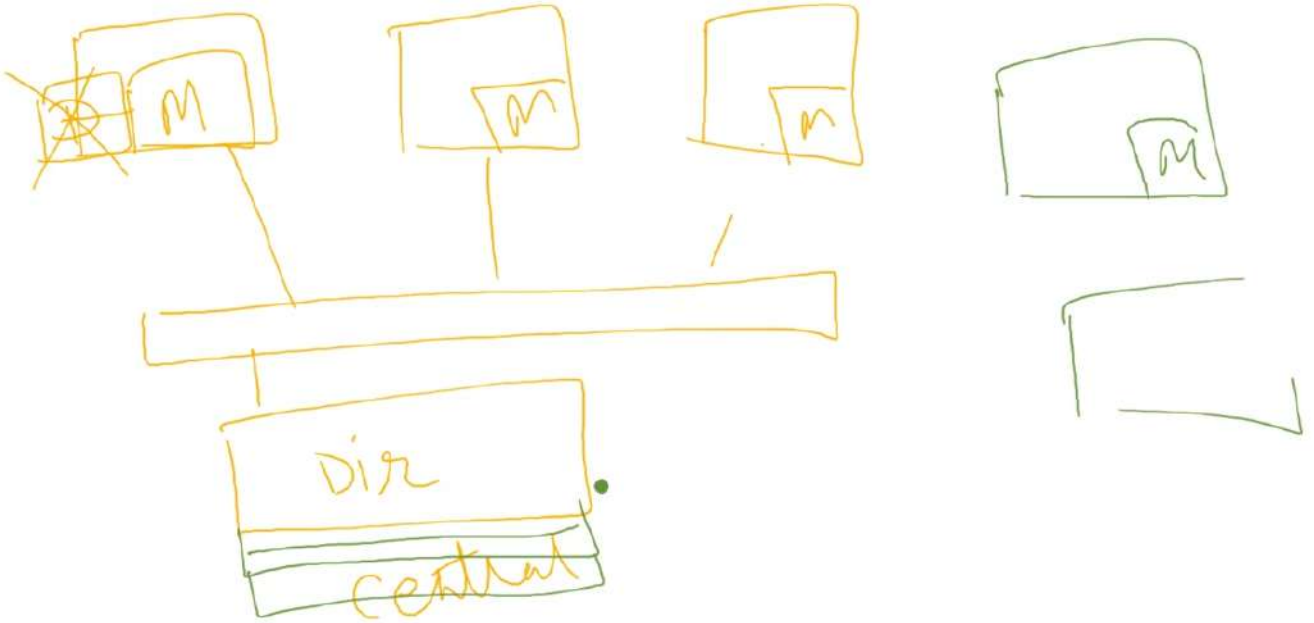
jamboard.google.com/d/1AlogDeB2COtifaU930L4zY8balv-UOA_HbR3-LKr-L4/viewer?f=5

27_Sep_2021

6/6

Share

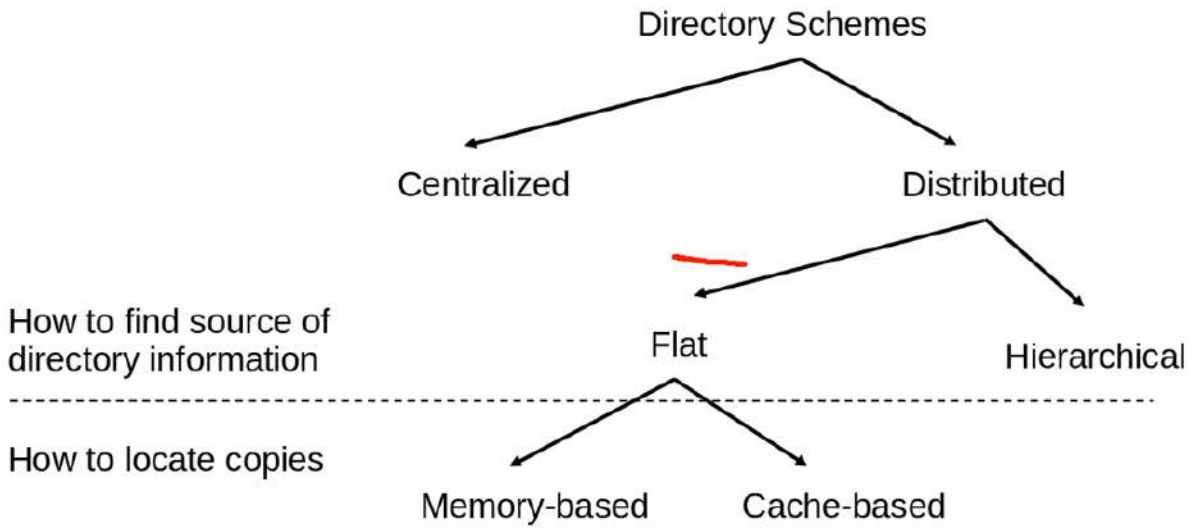
Set background | Clear frame



67% (1:55) Thu, September 30, 14:50

+8 | A | VP | SM | NS NALABOLU SANDEEP | TANVISH | AS ASWATHY N S | Syam Sankar | YK YOGESH KUMAR | SUBRATA ROY | Hemangee Kalpesh Kapoor

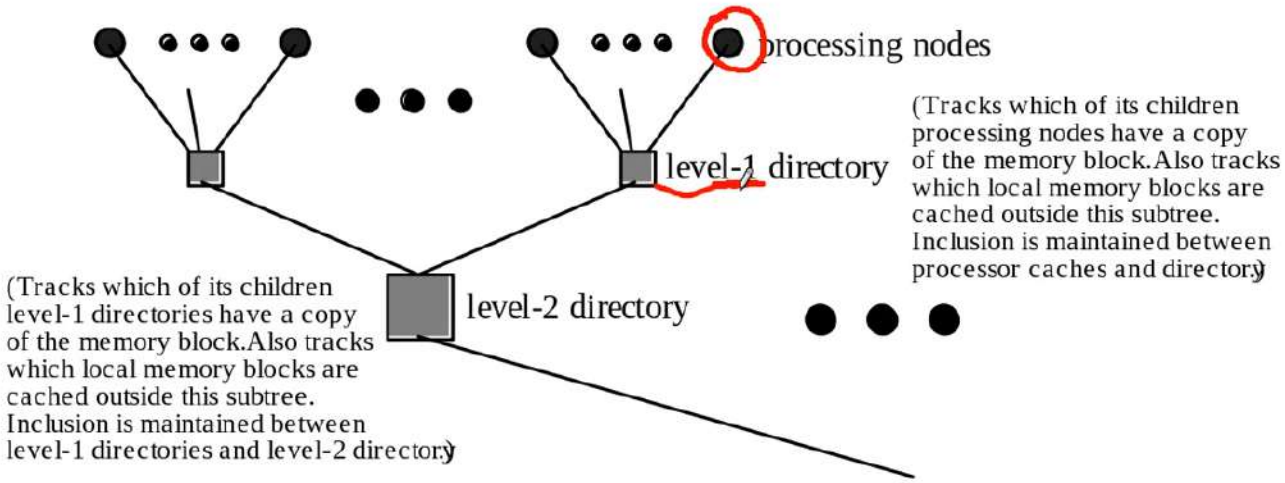
Organising Directories



How to find directory information?

- **Centralised**
  - Easy
  - Go to directory and get information
  - Not scalable
- **Distributed**
  - Flat
    - Directory information at home nodes
    - Location based addressing (hash)
    - Network transaction sent to home
  - Hierarchical
    - Leaves are processor nodes
    - Internal node is directory
    - Logical hierarchy (not necessarily physical), embedded in the interconnection network
      - In fact, in practice, every processing node in the system not only serves as a leaf node for the blocks it contains but also stores directory information as an internal tree node for other blocks

How hierarchical directories work?



- Nodes directory entry for a block says whether each subtree caches the block
- To find directory information, send “search” message up to the parent
- Point-to-point messages between children and parents

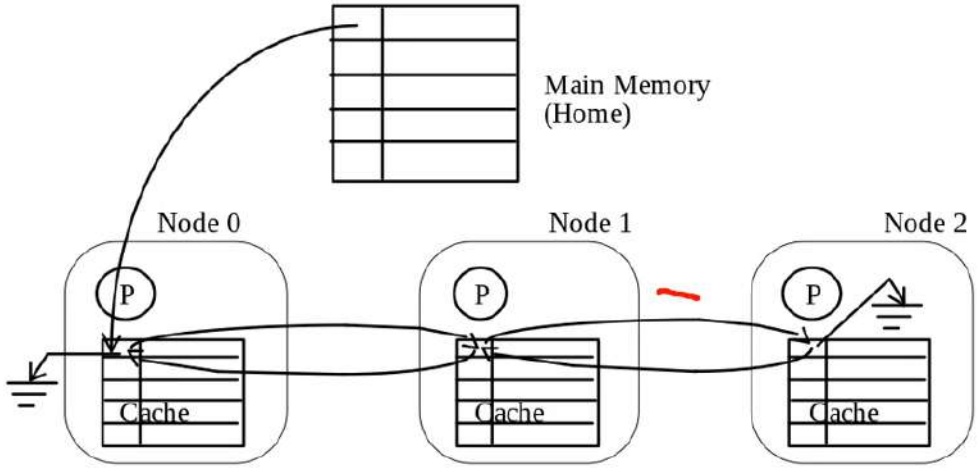
How is location of copies store?

- Hierarchical
  - Each directory has presence bits for child-subtrees and dirty-bit
- Flat
  - Vary a lot
  - Different storage overheads and performance characteristics
  - Memory based
    - Information of all copies at home node
    - Ex: Dash, Alewife, SGI Origin, Flash
  - Cache based
    - Information of copies distributed among copies themselves
      - Each copy points to next
    - Ex: Scalable Coherence Interface (SCI) IEEE Standard

Flat, memory based schemes

- Information about copies co-located with block at the home
- Performance scaling
  - Traffic on write: sent to nodes only in sharer list
    - Proportional to #sharers
  - Latency on write: need to inv others
    - Message can be sent to all in parallel
    - The number of fully serialised network tranactions in the critical path is thus not proportional to the number of sharers => reduced latency
  - Storage overhead
    - Simple representation: 1-bit per node: full bit-vector
    - Storage overhead does not scale with 'P' (no. of processors)

Flat, Cache-based schemes



- Memory has pointer to first node = Head
- List maintained as distributed doubly linked list
- Each cache has tag, coherence state and pointer to forward and backward node in the list of sharers