MANCHESTER 1824
The University of Manchester

COMP35112 Chip Multiprocessors

Directory-Based Cache Coherence

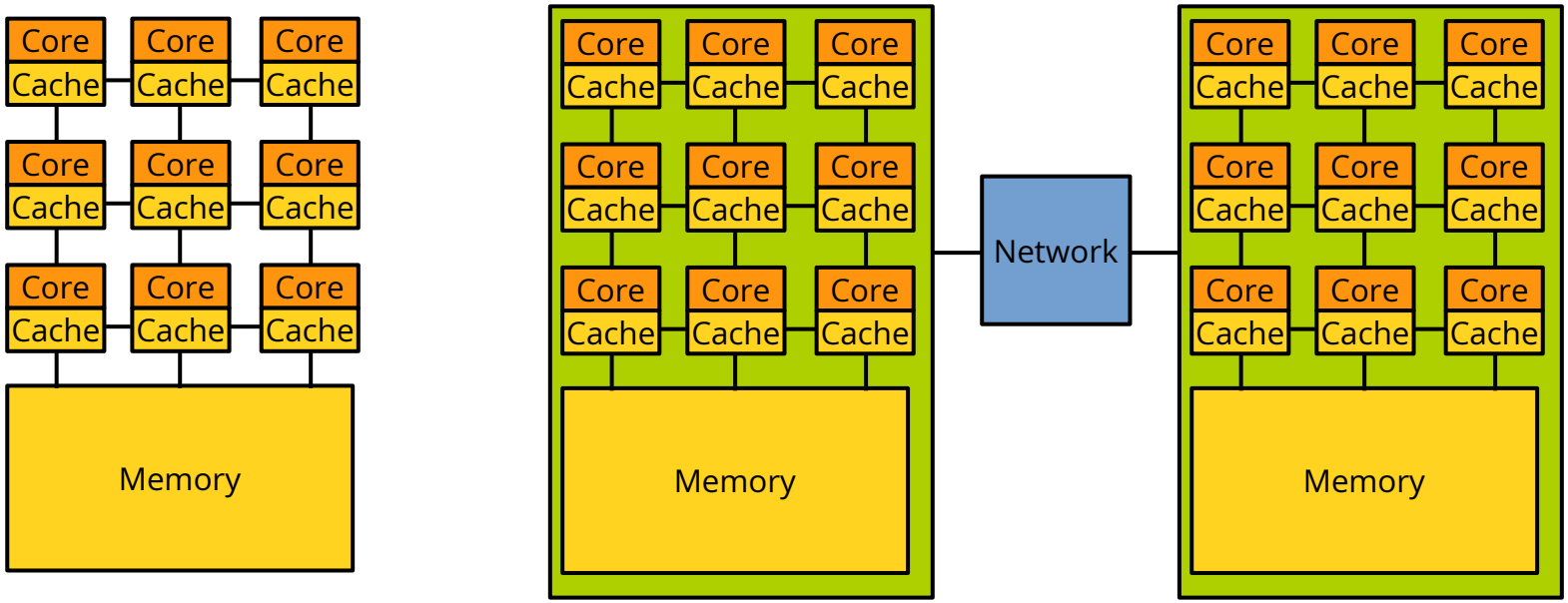
Pierre Olivier

Directory Based Coherence

- Shared bus coherence does not scale to a large number of cores
- Coherence scheme with a less directly connected network?

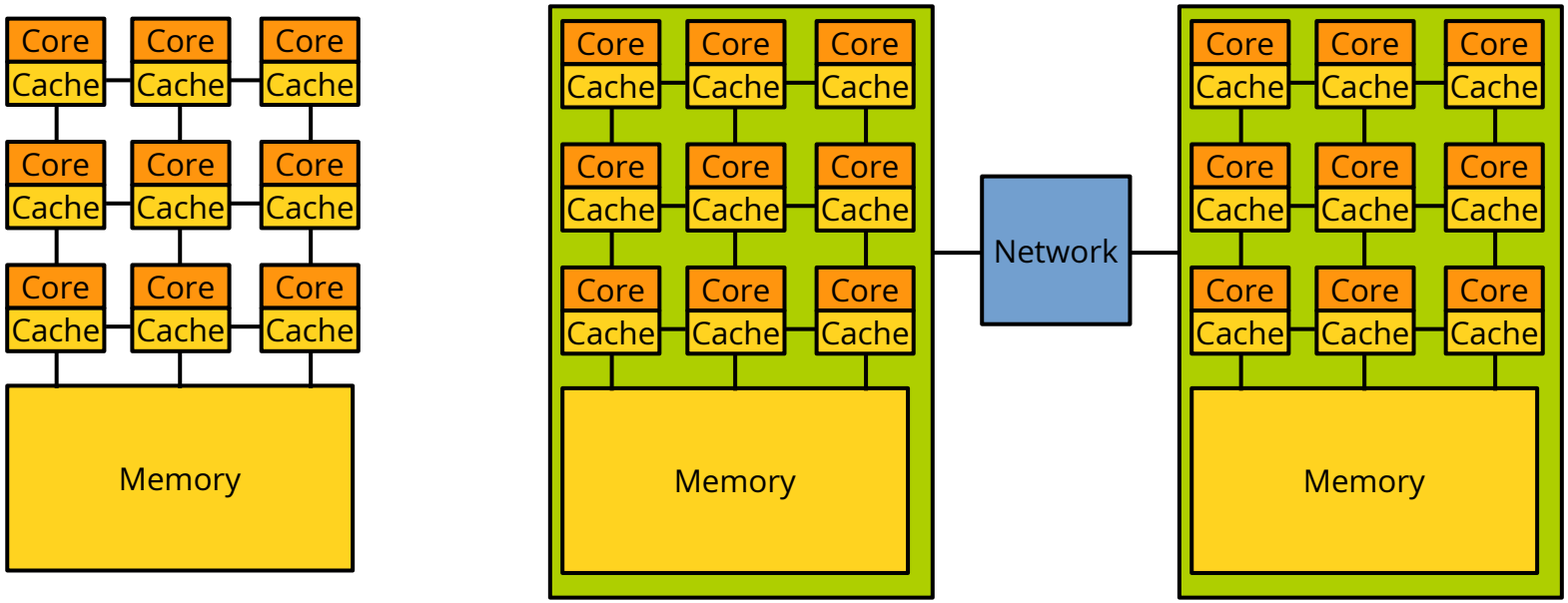
Directory Based Coherence

- Shared bus coherence does not scale to a large number of cores
- Coherence scheme with a less directly connected network?



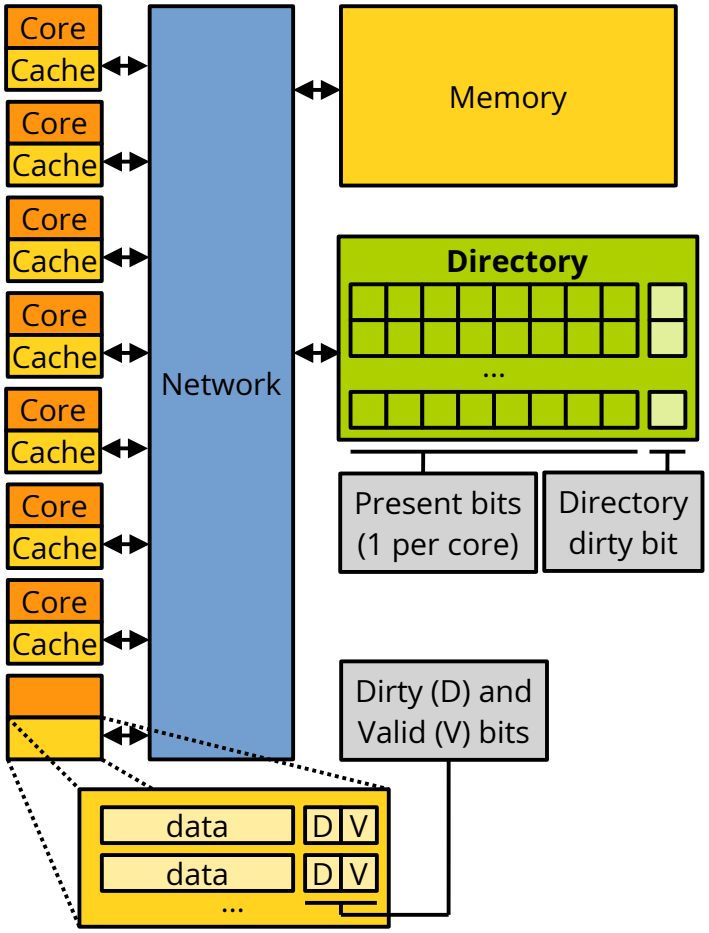
Directory Based Coherence

- Shared bus coherence does not scale to a large number of cores
- Coherence scheme with a less directly connected network?



- **1 possible solution: directory centralising cache line information**

Directory Structure

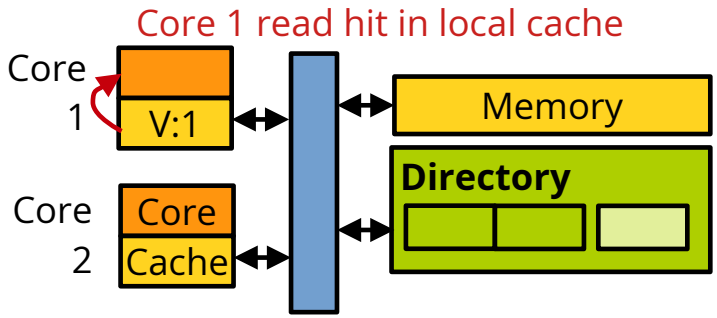


- Each **directory entry** (1 per cache line) has:
  - *Present* bitmap: which core has a copy
  - *Directory dirty bit*: only one owner and that copy is out of sync with memory
- Each line in each cache also has:
  - *Local valid bit*: is the line valid?
  - *Local dirty bit*: core is sole owner and data is out of sync with memory

Directory Protocol

Read hit in local cache

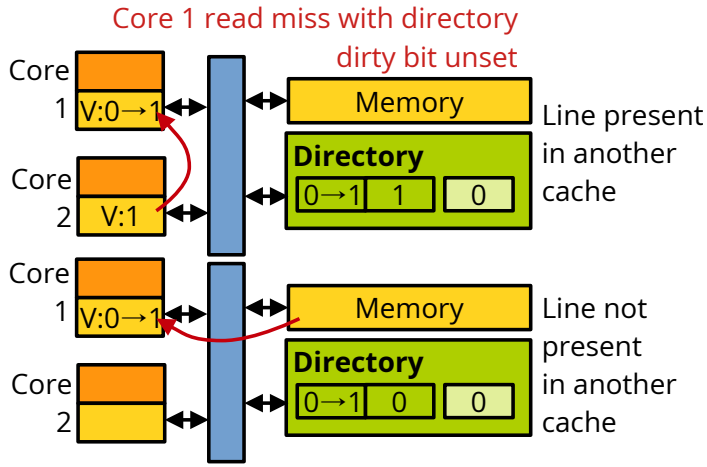
- No need for directory access: simply read local value



Directory Protocol

Read miss in local cache:

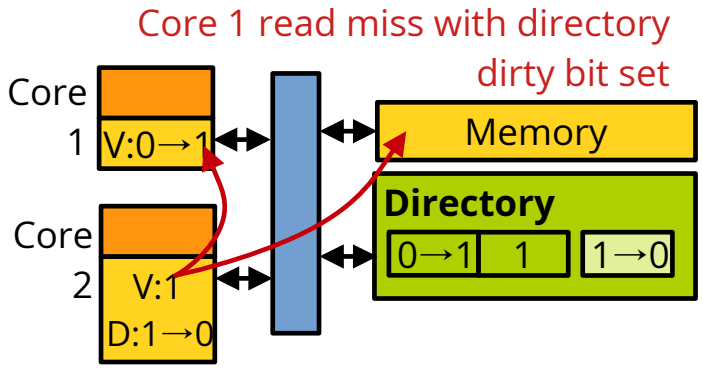
- ***Directory dirty bit unset?***
  - Query directory and get line from other cache if present, fetch from main memory otherwise
  - Set directory present bit for the reading core
  - Set local valid bit in the cache of the reading core



Directory Protocol

Read miss in local cache:

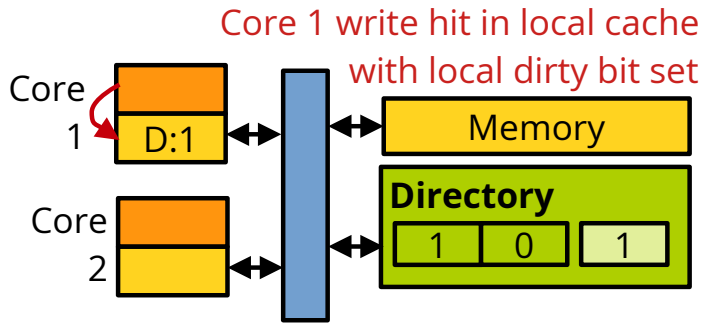
- ***Directory dirty bit set?***
  - One-and-only owner core updates memory, clears its dirty bit, and sends the line to the reading core
  - Clear directory dirty bit
  - Sets for the reaidng core present bit in teh directory, and local valid bit



Directory Protocol

Write hit in local cache with local dirty bit set

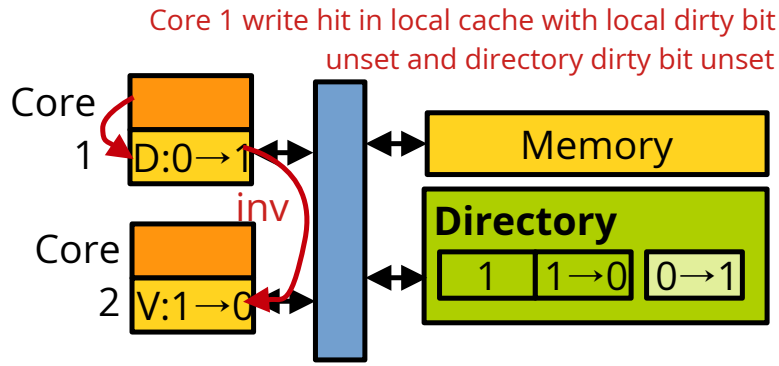
- Just update local cache



Directory Protocol

Write hit in local cache with local dirty bit unset

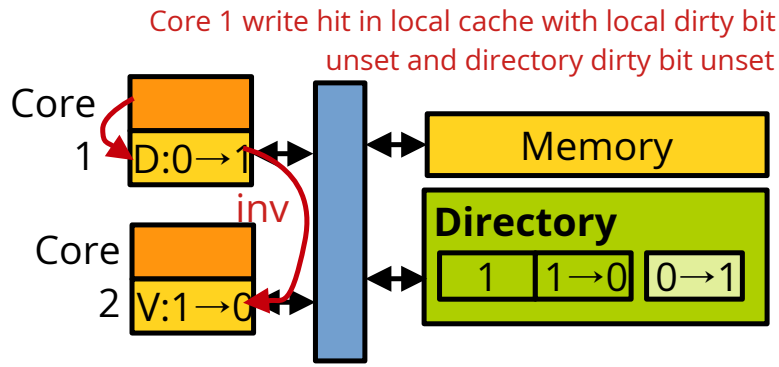
- ***Directory dirty bit unset?***
  - Consult present bits in directory and send invalidate messages to any cores having the data
  - Set local dirty bit for writing code
  - Set directory dirty bit



Directory Protocol

Write hit in local cache with local dirty bit unset

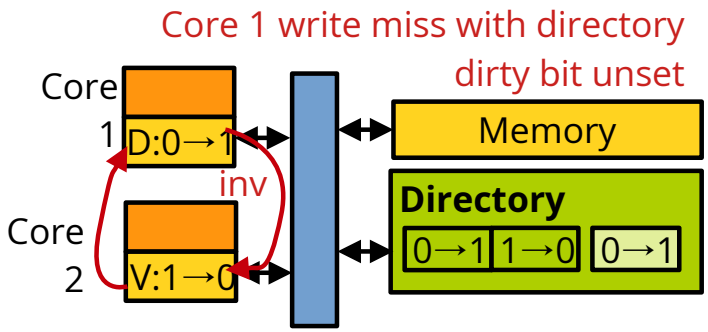
- ***Directory dirty bit unset?***
  - Send invalidate to any cores x with p[x] set and then clear these bits
  - Set p[i] bit for writing core and set directory dirty bit
  - Set local dirty bit
- ***Directory dirty bit set?***
  - **Cannot happen:** local dirty bit unset means the line is shared and there cannot be a unique owner



Directory Protocol

Write miss in local cache

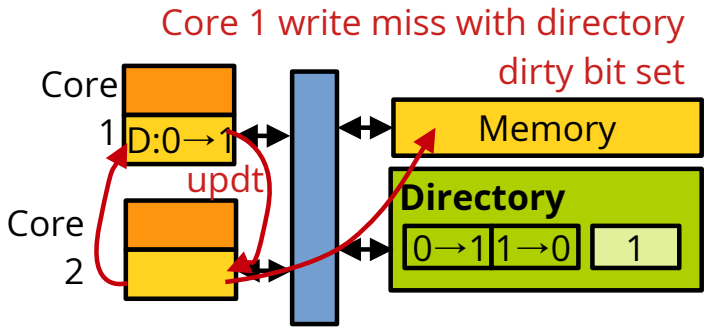
- ***Directory dirty bit unset?***
  - Consult direcotry present bits, get cache line from any core that has it or if not fetch line from memory
  - Send invalidate to any core x with present bit set in the directory, and clear those bits
  - Set directory present bit bit for the writing core and set directory dirty bit
  - Set local dirty bit in the writing core



Directory Protocol

Write miss in local cache

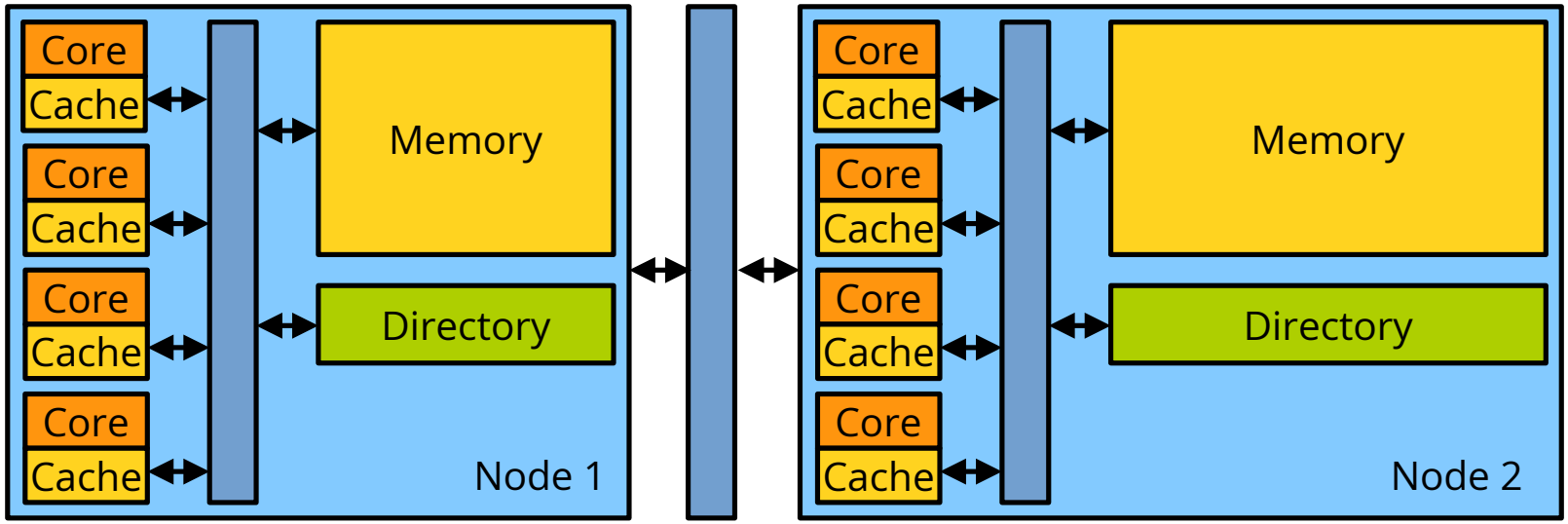
- ***Directory dirty bit set?***
  - Writing core sends message to owner core to update memory and send the cache line
  - Writing core updates data and sets its local dirty bit
  - Leave directory dirty bit set
  - Clear previous owner's directory present bit, set the writing core's directory present bit



Analysis

- ~MSI
- Optimisations are possible
- **Central directory is a serious bottleneck**
  - Distribute directory and cache it
  - Used in multi-sockets (CPU chips) systems

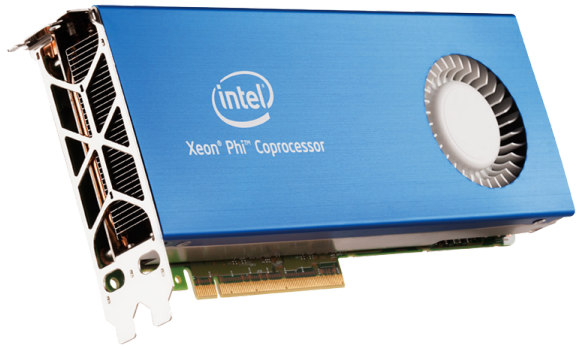
NUMA Systems



NUMA stands for Non-Uniform Memory Access

Drawbacks

- Slower communications
- Long/variable delays requires handshakes
- Machines used directory-based coherency:
  - SGI Origin (up to 2048 cores), Xeon Phi (60 cores).



Summary

- Bus-based snooping does not scale to large numbers of core
- Directory-based coherency to the rescue, but not a panacea
- ***Do we really need cache coherency?***