



Multicore Workshop

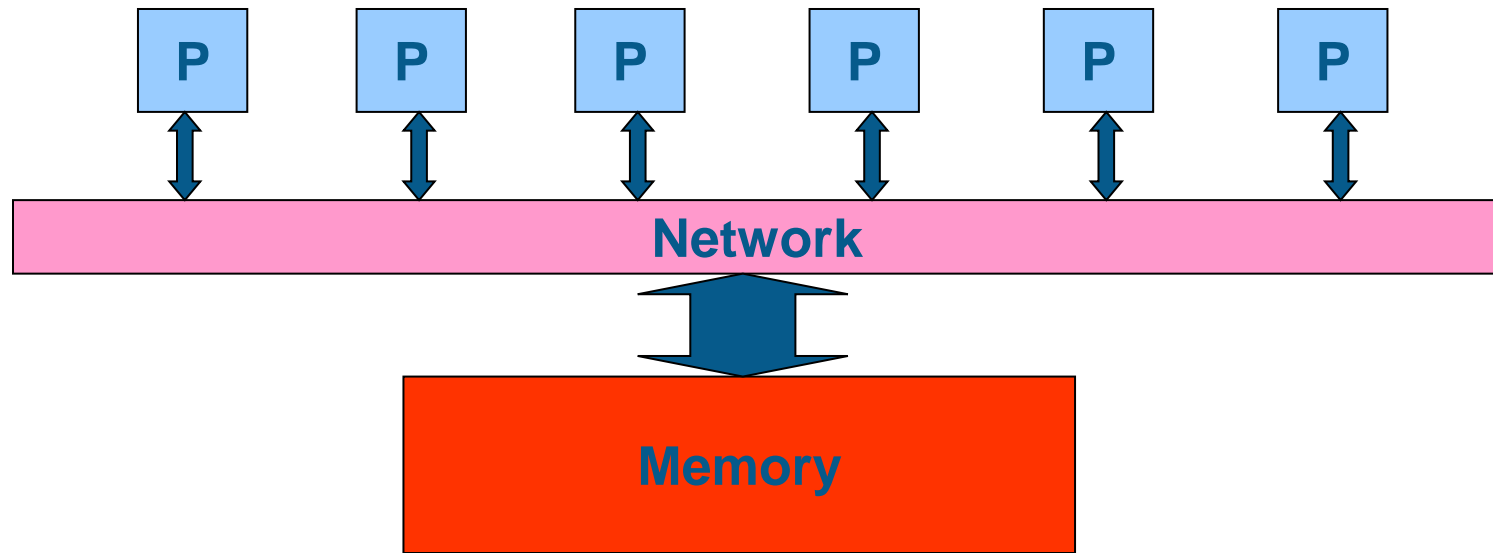
NUMA

Mark Bull

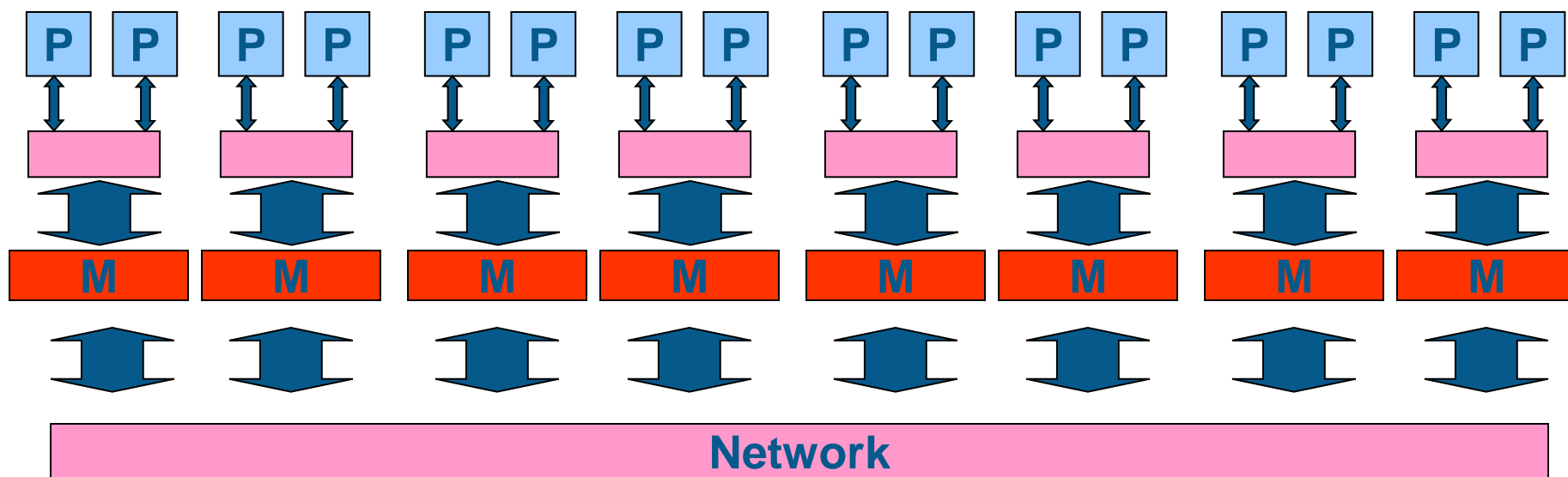
David Henty

EPCC, University of Edinburgh

- Shared memory machines using buses and a single main memory do not scale to large numbers of processors
 - bus and memory become a bottleneck
- Distributed shared memory machines designed to:
 - scale to larger numbers of processors
 - retain a single address space
- Modest sized multi-socket systems connected with HyperTransport or QPI are, in fact, distributed shared memory
- Also true of recent multicore chips
 - multiple “dies” on a single chip (i.e. single socket)



Examples: Sun X4600, all multicore PCs, IBM p575, NEC SX8, Fujitsu PRIMEQUEST



- For scalability, there is no bus, so snooping is not possible
- Instead use a directory structure
 - bit vector for every block
 - one bit per processor
 - stored in (distributed) memory
 - bit is set to 1 whenever the corresponding processor caches the block.
- Still some scalability issues:
 - directory takes up a lot of space for large machines
 - e.g. 128 byte cache block, 256 processors: directory is 20% of memory
 - some techniques to get round this

- Node where memory (and directory entry) is located is called the **home** node.
- Basic principal is same as snoopy protocol
 - cache block has same 3 states (modified, shared, invalid)
 - directory entry has modified, shared and uncached states.
- Cache misses go to the home node for data, and directory bits are set accordingly for read/write misses.
- Directory can:
 - invalidate a copy in a remote cache
 - fetch the data back from a remote cache
- Cache can write back to home node.

- We have described a distributed shared memory system where every memory address has a home node.
- This type of system is known as a cache-coherent non-uniform memory architecture (cc-NUMA).
- Main problem is that access to remote memories take longer than to local memory
 - difficult to determine which is the best node to allocate given page on
- OS is responsible for allocating pages
- Common policies are:
 - first touch: allocate on node which makes first access to the page
 - round robin: allocate cyclically

- Possible for the OS to move pages between nodes as an application is running
- Pages can either be **migrated** or **replicated**.
- Migration involves the relocation of a page to a new home node.
- Replication involves the creation of a “shadow” of the page on another node.
 - read miss can go to the shadow page
- Cache coherency is still maintained by hardware on a cache block basis.