Scalable Cache Coherent Shared Memory Case Studies

Underlying lessons

- Point of integration into the storage hierarchy
- Power of accessing memory by name (rather than address)
- Units of translation, access control, transfer
- Performing necessary comm. vs avoiding unnecessary

Large CC Design and Implementation Experiments

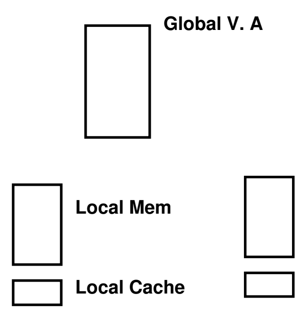
- Stanford DASH
  - **scalable directory structure, memory consistency models, ...**
- MIT Alewife
  - **SW assisted directory, integration with Msg Passing, fast interrupts**
- KSR
  - **Ring-based CC, Cache-Only Memory Architecture**
- Wisconsin Wind Tunnel / Tempest / Typhoon / T-Zero
  - **Minimal HW support**
- Stanford FLASH
  - **Flexible protocol processing, Msg Passing, General Purpose**
- plus compiler-based shared memory, O/S based shared memory

Requirements

- Observe all memory references that potentially involve communication
  - **depends on specific form of sharing permitted**
    - » **granularity**
    - » **read-only, RW with invalidate, ...**
  - **read or write miss on shared data (go get it)**
  - **write-hit on non-exclusive shared data**
  - **read or write miss that replaces shared data**
- Take action on behalf of processor to rearrange the memory hierarchy to that the reference can complete without communication
- Service the requests of other processors

O/S Level Shared Memory

- Provide each process with global virtual address space
  - **observe only through page faults**
- On access to an invalid shared page -> fault
  - **Communicate with home node to obtain page**
    - » **it will invalidate, update, recall copies as appropriate**
  - **Map it in as read-only if shared, RW if exclusive**
- Buffer as many writes as possible
  - **Lazy release consistency Models**
- Coarse-grain sharing, no HW support
  - **many paper proposals for NI assists**
- See Ivy, Munin, Treadmarks



The Stanford DASH Experiment

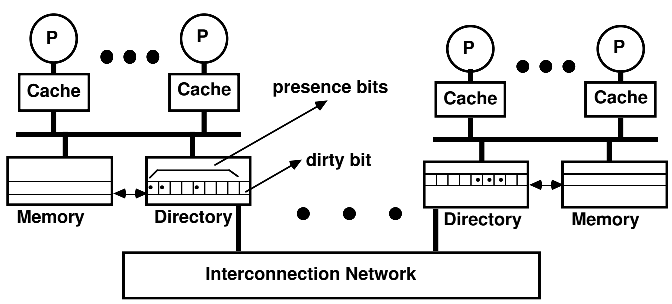
Build a large-scale multiprocessor that provides:

1. **an effcient implementation of the single-address-space model with hardware support for caching**

2. **methods for tolerating latency**

3. **significant instrumentation to allow applications and architecture to be monitored and studied**

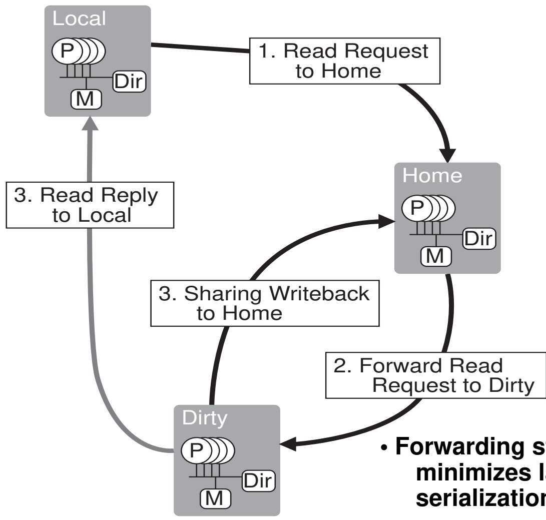
DASH ==> Directory Architecture for SHared memory



- **Nodes connected by scalable interconnect**
- **Partitioned shared memory**
- **Processing nodes are themselves multiprocessors**
- **Distributed directory-based cache coherence**

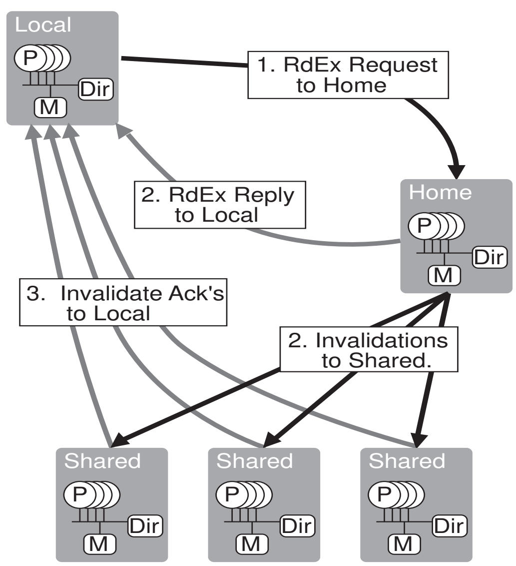
Directory Protocol Examples

- **Read of remote-dirty data**

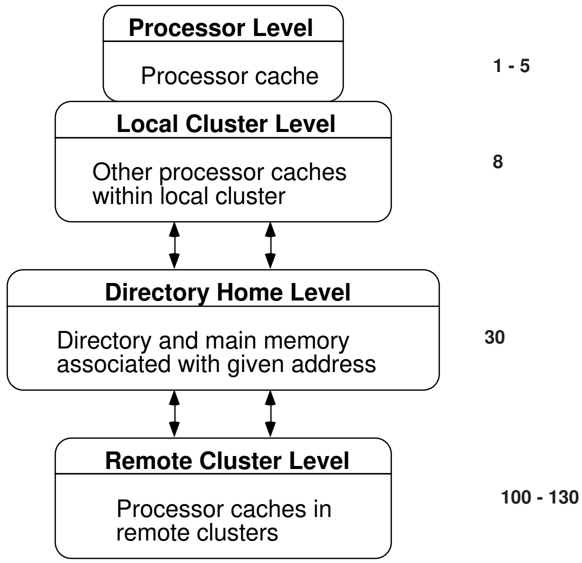


- **Forwarding strategy minimizes latency and serialization**

Read-exclusive (write) to shared data



DASH Memory Hierarchy



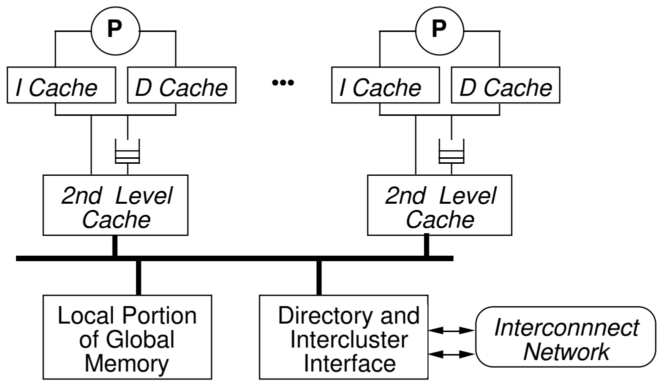
DASH Prototype

64 processors (16 clusters with 4 PEs each)

- **peak 1.6 GIPS and 600 MFLOPS**

- **Cluster hardware:**
  - **modified Silicon Graphics 4D/340 nodes**
  - **new directory and reply controller boards**
- **Interconnection network:**
  - **pair of wormhole-routed mesh networks**
  - **16-bit paths, 50ns fall-thru time, 35ns cycle time**
- **Single multithreaded Unix System V operating system**

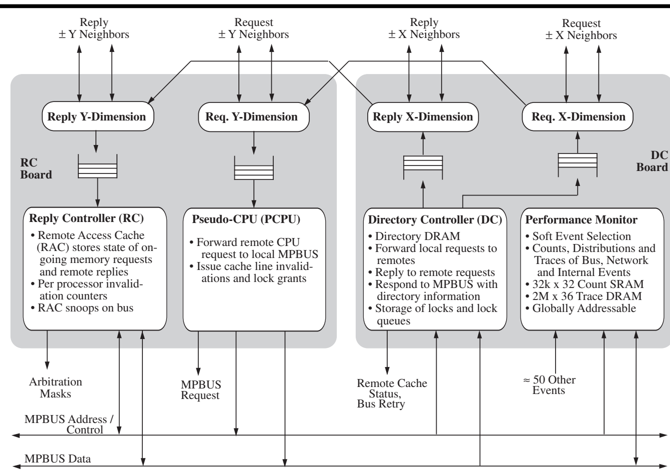
Cluster Structure



Illinois protocol maintains local cache coherency

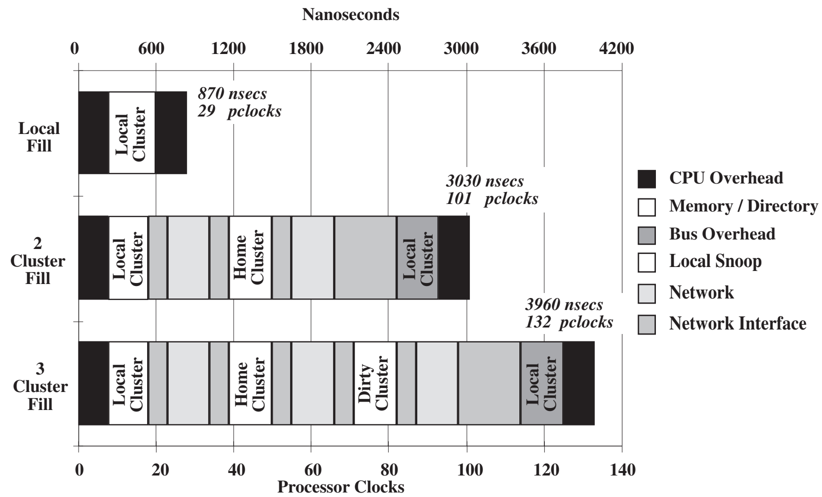
- **Directory logic adds:**
  1. **Status information about remote caches**
  2. **Interface for local processors to remote clusters**
  3. **Interface for remote processors to local cluster**

Directory / Interface Logic

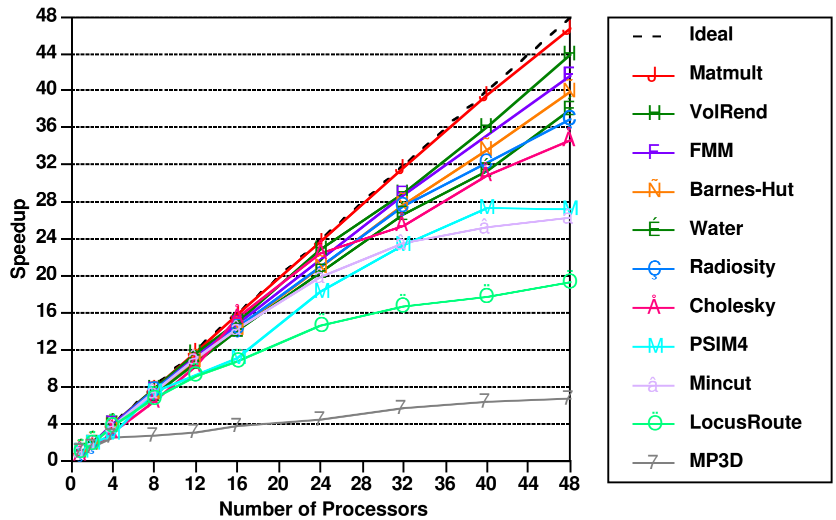


1. **Directory Controller**
2. **Pseudo CPU**
3. **Reply Controller**
4. **Network Interface**
5. **Performance Monitor**

DASH Memory Access Latencies



Speedups on DASH



Software Strategy

- **Single multithreaded operating system**
  - **scalability of kernel data structures and locking**
  - **scheduling and virtual memory system design**
  - **concurrent I/O and file system design; fault containment**
- **Extended programming languages**
  - **coarse-grain parallelism indicated by user**
  - **fine-grain and regular parallelism via compiler**
  - **sophisticated runtime with focus on locality**
- **New compiler technology**
  - **emphasis on both locality and parallelism**
  - **study of both compiler-HW and compiler-language interactions**
- **Performance debugging tools**
- Application studies
  - **parallelization of important classes of applications**
  - **implications for system design**
  - **scaling considerations**

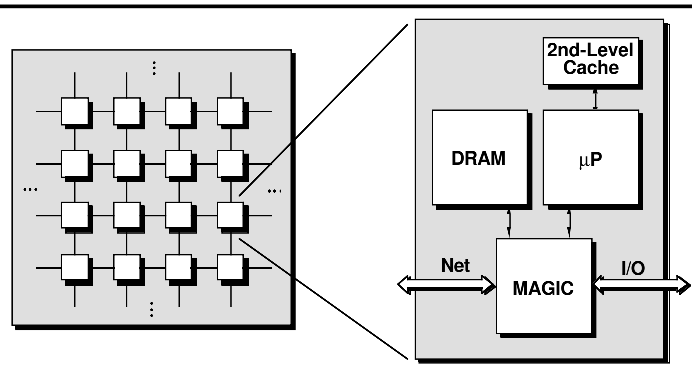
Observations

- Cache-coherent shared memory seeks to avoid unnessary communication
  - **bring data to where it is accessed**
  - **attempt to closely track access patterns**
    - » **tailor protocols to sharing patterns**
- Communication occurs as a last resort
  - **each level of hierarchy is optimized to reduce hit time**
  - **communication occurs only on a miss**
- Communication performance is obtained through HW integration
  - **machinery in place for moving cache lines around, buffering, etc.**
- What if the application **knows** that it wants to move data, i.e., communicate?
  - **exert direct control over the CC machinery**
  - **memory-to-memory DMA transfers**

Design Goals of FLASH

- Integration of compute-server and network-of-workstations models
  - **single programming model and similar hardware performance**
- Integration of cache-coherent shared memory and message passing
  - **effective support for both short and long user-level messages**
- Scalability to thousands of processors
  - **good cost-performance from small-scale to large-scale machines**
- Debuggability and extensibility
  - **ability to debug complex coherence protocols**
  - **ability to support new synchronization primtives, performance statistics, ...**

FLASH: FLexible Architecture for SHared memory

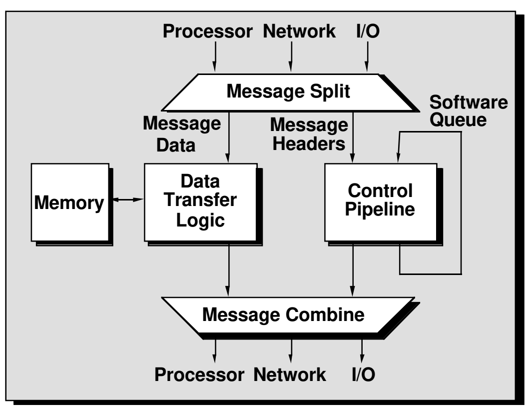


- Replace FSMs in DASH's directory controller with programmable engine
- Replace separate directory memory with portion of main memory
- All transactions must go through the MAGIC controller chip
- Support both multiprocessor and local-area network technology

Requirements of the MAGIC Chip

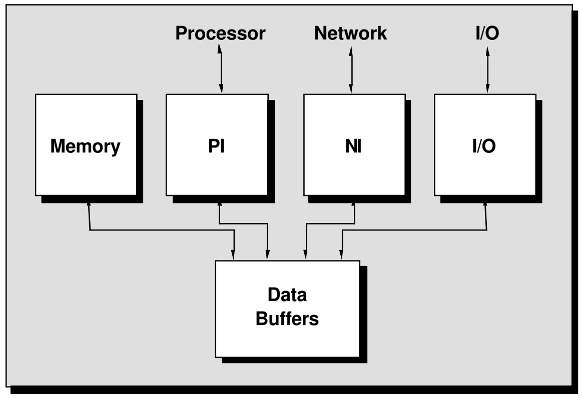
- Ability to shuffle lots of data around quickly
  - **e.g., data portion of cache lines**
- Ability to process lots of short code sequences quickly
  - **e.g., rd/wr requests from network and local PE**
- Interface to multiple entities
  - **e.g., cache controller, memory controller, I/O, network, processor bus**

Magic Architecture



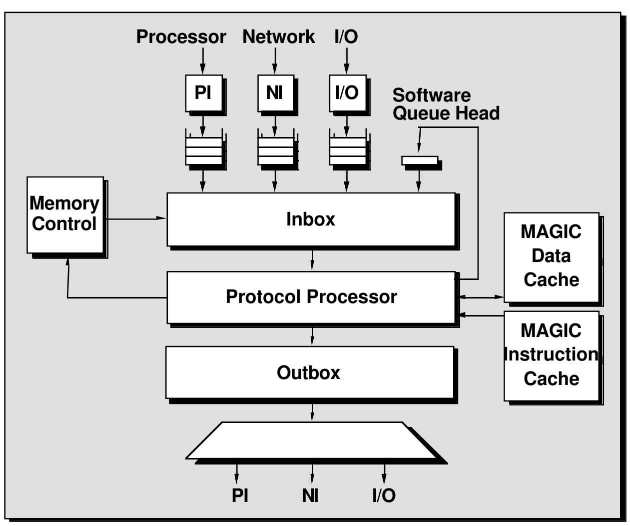
- Separate data-transfer logic and control logic
  - **Data shuffling handled by hardware**
  - **Control handled by software**

Data Transfer Logic



- Data directly stored in data buffers; no redundant copying
- Extensive use of pipelining to reduce latency

Control Pipeline

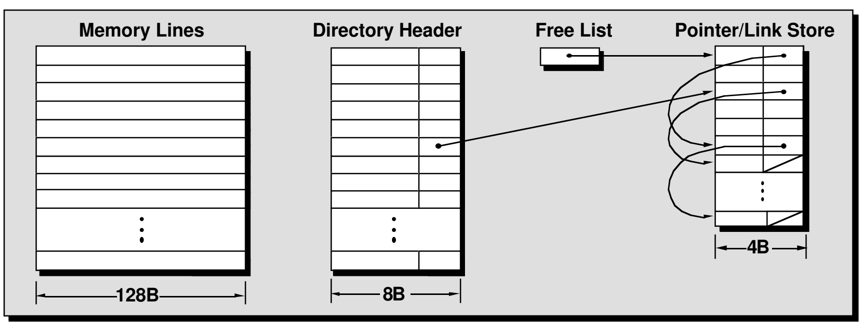


Key Challenges

- Low protocol processor occupancy
  - **coarse-level pipeline consisting of inbox, protocol processor, and outbox**
  - **hardware jump table implies no software decoding**
  - **use of instruction and data caches**
  - **dual issue CPU with slightly specialized instruction set**

- Low latency in servicing requests
  - **features that ensure low processor occupancy help here too**
  - **speculative start of memory operations**
  - **extensive pipelining of data**

Support for Cache Coherence

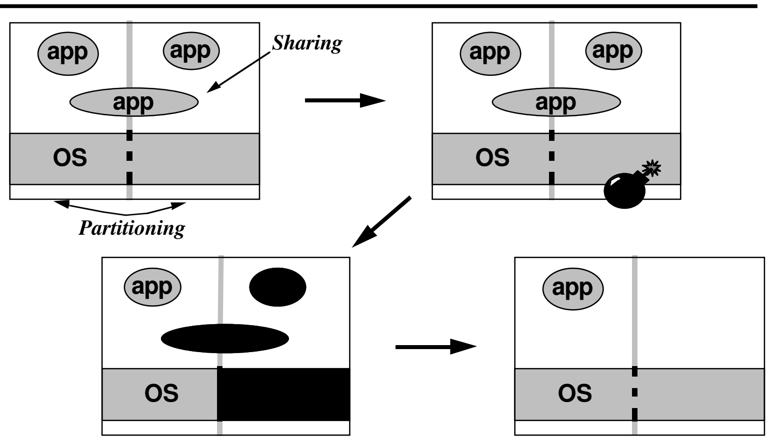


- Protocol processor allows sophisticated directory data structures
- Current plan is to use a dynamic-pointer directory scheme:
  - **for 128-byte cache lines main-memory overhead is 8-9%**
  - **PP occupancy balanced with occupancy of main memory**
- Easy to support different directory schemes, even COMA.

Support for Message Passing

- Full protection and multiprogramming are supported
- Message transferred in cache-line sized chunks
  - **exploits existing specialized datapath for cache-line transfers**
  - **both protocols can exist without causing deadlock**
- Message usually stored directly in user buffer
- Performance expected to be several hundred Mbytes/sec

Flash System Architecture

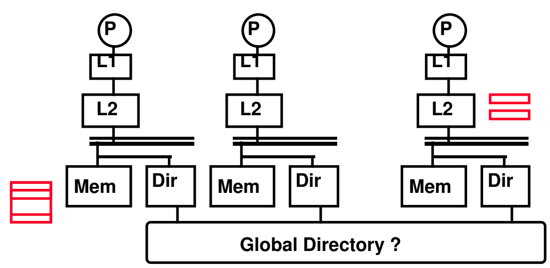


- Hybrid organization:
  - **partitioning: fault containment & scalability**
  - **shared memory/fast messaging: efficient resource usage/ allocation**

WWT / Typhoon and GVA management

- Apply (Page level) Shared Virtual Ideas
  - **no real page faults on a CM-5**
- Allow empty lines in a shared page by setting ECC bits
  - **cause trap on access to invalid page**
  - **(no real memory errors)**
  - **trap handler performs coherence operations and does communication**
- Typhoon uses an auxilliary table instead
- What happens when a local miss causes a remote line to be thrown out of the cache?
  - **it is retired to local memory!**
  - **this is the key idea in the Stache protocol**
- COMA migrates the home to the point of update
  - **replace exclusive copy of "superblock" to local memory**
  - **simulate on DASH by changing page mapping**

COMA



- On replacement, want to fall back one level in hierarchy of primary user
  - **not necessarily the original home**
- Primary benefit is for programs
  - **fine-grain interleaving**
  - **capacity misses dominate coherence misses**
- On Miss need mechanism for locating Home
  - **rely on interconnection topology to do a global match**
  - **hierarchy or ring**

Summary: Points of Integration