

Implications for Programming Models

Todd C. Mowry
CS 418
January 27, 2011

Implications for Programming Models

Shared address space and explicit message passing

- SAS may provide coherent replication or may not
- Focus primarily on former case

Assume distributed memory in all cases

Recall any model can be supported on any architecture

- Assume both are supported efficiently
- Assume communication in SAS is only through loads and stores
- Assume communication in SAS is at cache block granularity

- 2 -

CS 418

Issues to Consider

Functional issues

- Naming
- Replication and coherence
- Synchronization

Organizational issues

- Granularity at which communication is performed

Performance issues

- Endpoint overhead of communication
 - (latency and bandwidth depend on network so considered similar)
- Ease of performance modeling

Cost Issues

- Hardware cost and design complexity

- 3 -

CS 418

Naming

SAS: similar to uniprocessor; system does it all

MP: each process can only directly name the data in its address space

- Need to specify from where to obtain or where to transfer non-local data
- Easy for regular applications (e.g. Ocean)
- Difficult for applications with irregular, time-varying data needs
 - Barnes-Hut: where the parts of the tree that I need? (change with time)
 - Raytrace: where are the parts of the scene that I need (unpredictable)
- Solution methods exist
 - Barnes-Hut: Extra phase determines needs and transfers data before computation phase
 - Raytrace: scene-oriented rather than ray-oriented approach
 - both: emulate application-specific shared address space using hashing

- 4 -

CS 418

Replication

Who manages it (i.e. who makes local copies of data)?

- SAS: **system**, MP: **program**

Where in local memory hierarchy is replication first done?

- SAS: **cache** (or memory too), MP: **main memory**

At what granularity is data allocated in replication store?

- SAS: **cache block**, MP: **program-determined**

How are replicated data kept coherent?

- SAS: **system**, MP: **program**

How is replacement of replicated data managed?

- SAS: **dynamically at fine spatial and temporal grain** (every access)
- MP: **at phase boundaries**, or emulate cache in main memory in software

Of course, SAS affords many more options too (discussed later)

- 5 -

CS 418

Amount of Replication Needed

Mostly local data accessed => little replication

Cache-coherent SAS:

- **Cache holds active working set**
 - replaces at fine temporal and spatial grain (so little fragmentation too)
- **Small enough working sets => need little or no replication in memory**

Message Passing or SAS without hardware caching:

- **Replicate all data needed in a phase in main memory**
 - replication overhead can be very large (Barnes-Hut, Raytrace)
 - limits scalability of problem size with no. of processors
- **Emulate cache in software to achieve fine-temporal-grain replacement**
 - expensive to manage in software (hardware is better at this)
 - may have to be conservative in size of cache used
 - fine-grained message generated by misses expensive (in message passing)
 - programming cost for cache and coalescing messages

- 6 -

CS 418

Communication Overhead and Granularity

Overhead directly related to hardware support provided

- Lower in SAS (order of magnitude or more)

Major tasks:

- **Address translation and protection**
 - SAS uses **MMU**
 - MP requires **software protection**, usually involving **OS** in some way
- **Buffer management**
 - **fixed-size small messages** in SAS easy to do in hardware
 - **flexible-sized message** in MP usually need software involvement
- **Type checking and matching**
 - MP does it in software: lots of possible message types due to flexibility
- **A lot of research in reducing these costs in MP, but still much larger**

Naming, replication and overhead favor SAS

- Many irregular MP applications now emulate SAS/cache in software

- 7 -

CS 418

Block Data Transfer

Fine-grained communication not most efficient for long messages

- Latency and overhead as well as traffic (headers for each cache line)

SAS: can use block data transfer

- **Explicit** in system we assume, but can be automated at page or object level in general (more later)
- Especially important to amortize overhead when it is high
 - latency can be hidden by other techniques too

Message passing:

- Overheads are larger, so block transfer more important
- But very natural to use since message are explicit and flexible
 - **Inherent in model**

- 8 -

CS 418

Synchronization

SAS: Separate from communication (data transfer)

- Programmer must orchestrate separately

Message passing

- Mutual exclusion by fiat
- Event synchronization already in send-receive match in synchronous
 - need separate orchestration (using probes or flags) in asynchronous

- 9 -

CS 418

Hardware Cost and Design Complexity

Higher in SAS, and especially cache-coherent SAS

But both are more complex issues

- **Cost**
 - must be compared with cost of replication in memory
 - depends on market factors, sales volume and other non-technical issues
- **Complexity**
 - must be compared with complexity of writing high-performance programs
 - reduced by increasing experience

- 10 -

CS 418

Performance Model

Three components:

- Modeling cost of primitive system events of different types
- Modeling occurrence of these events in workload
- Integrating the two in a model to predict performance

Second and third are most challenging

Second is the case where cache-coherent SAS is more difficult

- replication and communication implicit, so events of interest implicit
 - similar to problems introduced by caching in uniprocessors
- MP has good guideline: messages are expensive, send infrequently
- Difficult for irregular applications in either case (but more so in SAS)

Block transfer, synchronization, cost/complexity, and performance modeling advantageous for MP

- 11 -

CS 418

Summary for Programming Models

Given tradeoffs, architect must address:

- Hardware support for SAS (transparent naming) worthwhile?
- Hardware support for replication and coherence worthwhile?
- Should explicit communication support also be provided in SAS?

Current trend:

- Tightly-coupled multiprocessors support for cache-coherent SAS in hw
- Other major platform is clusters of commodity PCs/blades
 - currently don't support SAS in hardware, mostly use message passing

- 12 -

CS 418

Summary

Crucial to understand characteristics of parallel programs

- Implications for a host of architectural issues at all levels

Architectural convergence has led to:

- **Greater portability of programming models and software**
 - Many performance issues similar across programming models too
- **Clearer articulation of performance issues**
 - Used to use PRAM model for algorithm design
 - Now models that incorporate communication cost (BSP, logP,....)
 - Emphasis in modeling shifted to end-points, where cost is greatest
 - But need techniques to model application behavior, not just machines

Performance issues trade off with one another; iterative refinement

Ready to understand using workloads to evaluate systems issues