Advanced Computer Architecture II: Multiprocessor Design

Parallel Architecture Fundamentals

Professor Russ Joseph
Department of Electrical and Computer Engineering
Northwestern University

January 5, 2007

Parallel Architecture Fundamentals

Abstractions and Implementations

Communication Architecture

Parallel Programming Models

Hardware Organization

Design Issues

What Do We Mean By *Parallel Architecutre*?

Parallel architecture is just a an extension of conventional computer architecture (e.g. as covered in ECE361)

A classic definition for the term *parallel computer*:
a collection of processing elements that communicate and cooperate to solve large problems fast (Almasi and Gottlieb 1989)
In our case:

- **Processing elements** are conventional uniprocessors.
- **Communication** and **cooperation** are new concepts.

Although, there are some new wrinkles, there are some important connections to concepts that you know already.

Architectural Abstractions and Organization

Computer architecture is all about building and interfacing components to work together efficiently.

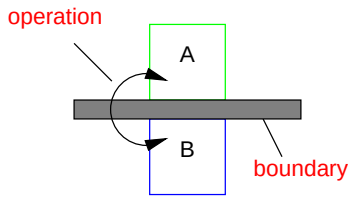
Two central concepts:

- abstractions - define the relationships between system components
- organization - define the inner-workings that enable performance

Architectural Abstractions

Critical abstractions describe the relationship and interactions between two components:

- **Boundaries** separate the components into distinct entities.
- **Operations** are tasks which can be performed at the boundary.



Architectural Organization

Implementations are the organizational structures achieves the required functionality.

Designers add complexity to implementations to maximize their performance.

However, complexity must be balanced with cost (which may not necessarily be price).

Hardware/Software Interface

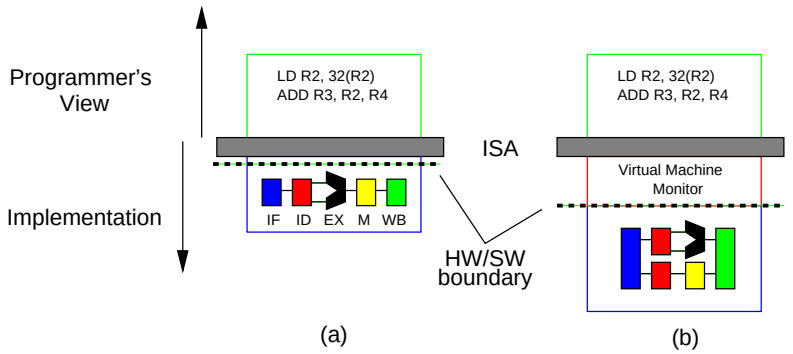
Instruction Set Architecture (ISA) serves as a contract between hardware and software.

Draws dividing line and establishes responsibilities.

Specifies operations performed at the boundary (instructions/interrupts) and associated data (registers/memory)

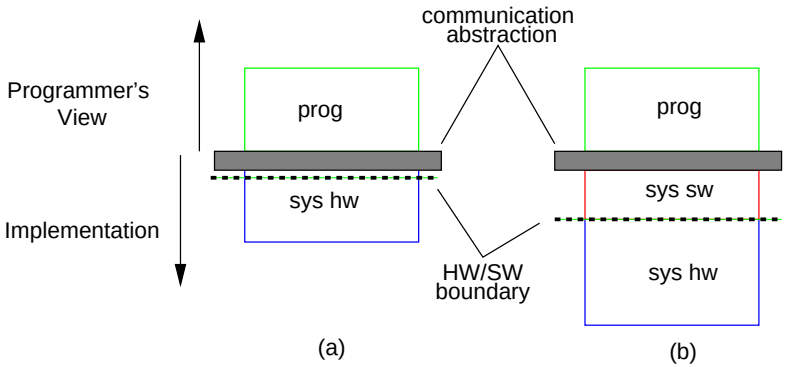
Physical implementation maybe realized any way as long as it respects the contract.

Example: Hardware/Software Interface



Parallel Architecture Interface

Communication Architecture describes both the set of abstractions as well as the organization.



Fundamental Design Issues

- **naming** - the way in which data/processes are referenced.
- **operations** - the actions that can be taken on the data
- **order** - the way in which accesses to the data are ordered or coordinated
- **communication cost** - performance issues relating to information exchange (e.g. latency, bandwidth, occupancy, overhead)
- **replication** - the methods for reducing communication by storing duplicated data

Naming Policy

Refers to the way in which processes may access data.

Specifically, which data items a given process may reach.

In a uniprocessor system, a thread may access any address in its virtual address space as well as machine registers.

Operations

These are actions that can be taken on the named data.

In uniprocessor load/store architecture, threads can perform many kinds of ALU operations on registers, but can only perform simple loads and stores from/to memory.

Of course, some CISC instruction sets support a wide number of operations on memory locations.

Ordering Policy

Refers to the logical sequencing between operations.

In uniprocessor architectures, we commonly expect sequential program order for most operations.

We expect the machine to behave as if the instructions were executed one at time in the order specified by the assembler.

Of course, modern processors have out-of-order implementations, but they still must enforce the ordering policy specified by the ISA.

Communication Cost

How does communication affect performance?

Communication usually costs something.

There is "no free lunch".

Highly dependent on algorithm, programmer, compiler, library, hardware implementation.

We will take a closer look at this soon...

Replication

Because communication can be expensive, we try to replicate data when it makes sense.

Although it can sometimes help performance by lowering communication costs, it may introduce communication of its own.

Data must be replicated judiciously.

We will spend a lot of time talking about this (via cache coherence).

Obviously closely coupled with communication factors.

Parallel Programming Models

Parallel architectures are implemented via layers of abstraction.

Programming model is the conceptualization of the machine that the programmer uses.

Examples:
- multiprogramming
- shared address space
- data parallel
- dataflow
- systolic arrays

Multiprogramming

Concept: All parallel threads have private address spaces and do not communicate at the programming level.

Analogy: Imagine a company where all the employees work in their own offices/cubicles and do not talk to each other.

Example: Standalone applications use this model.

Design Issues for Multiprogramming

Naming: No process/thread can access memory outside its own address space.

Operations: Garden variety loads/stores on memory operations.

Ordering: Most commonly, sequential program order for loads/stores.

Communication Cost: N/A No inter-thread communication.

Replication: N/A No inter-thread communication.

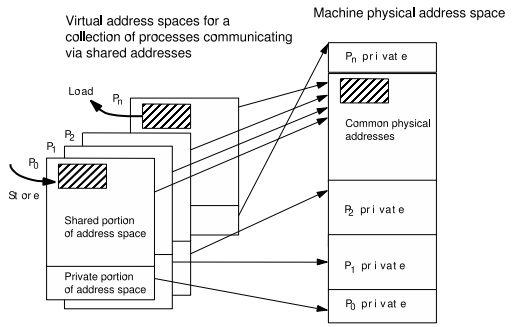
Shared Address Space

Concept: Parallel threads share some portion of their virtual address space.

Analogy: Another company were everyone sits at a large table and freely looks at everyone else's on-going work.

Example: Database applications are often implemented in this style.

Memory Model for Shared Memory Programs



Design Issues for Shared Memory Programs

Naming: Any thread can access any memory location regardless of which thread last wrote to it.

Operations: Loads/stores and often special instructions for synchronization.

Ordering:
Within Thread: Sequential program ordering
Across Threads: Interleaved data access. Need to use mutual exclusion primitives.

Communication Cost: Depends on implementation.

Replication: Depends on implementation.

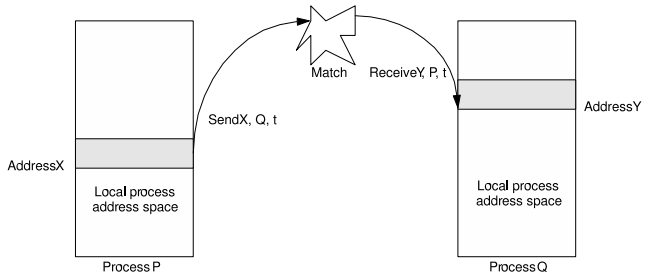
Message Passing

Concept: Parallel threads have private address space, but communicate via explicit exchange via send/receive operations.

Analogy: In this company people work in their own offices/cubicles, but they exchange phone calls, e-mails, and instant messages from time to time.

Example: Some large scale scientific modeling programs use this approach.

Send/Receive Primitives for Message Passing Programs



Design Issues for Message Passing Programs

Naming: Threads may access their own private data directly, but there is no shared memory.

Operations: Loads/stores on own private data, Send/Receive primitives for exchanging information across threads.

Ordering:
Within Thread: Sequential program ordering
Across Threads: Synchronization is achieved point-to-point via send/receive. Mutual exclusion is implicit.

Communication Cost: Depends on hardware implementation.

Replication: Determined by algorithm and programmer.

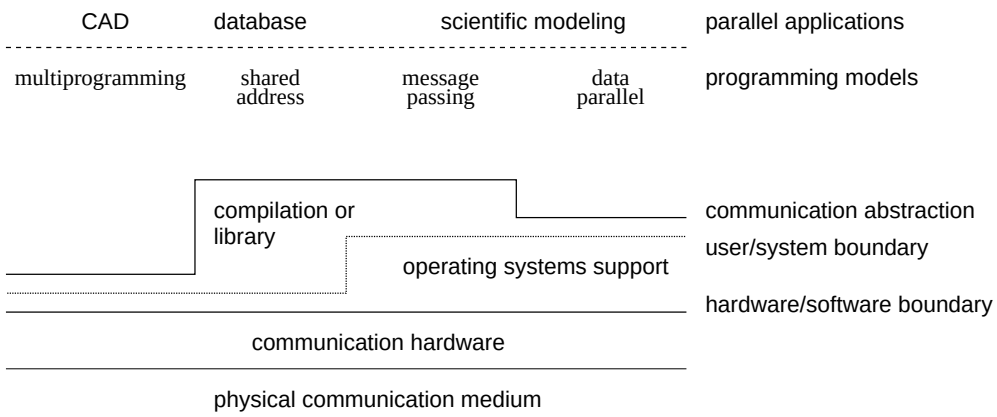
Data Parallel

Concept: Parallel threads perform same action on different elements of a data set and simulataneously engage in global communication at intervals.

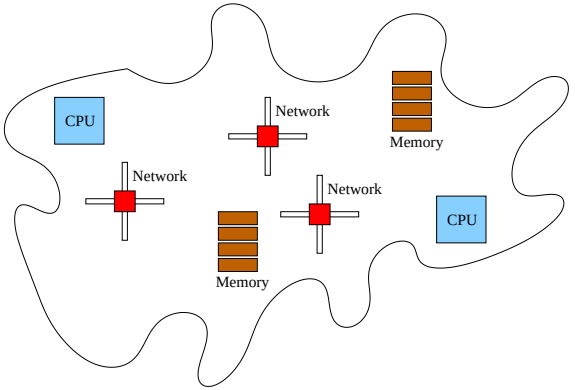
Analogy: Everyone in this company has an identical job description and title. They work independently at their own desks/cubicles, and have regularly scheduled meetings where all employees attend and give updates on their progress.

Example: This is popular for large scale scientific modeling.

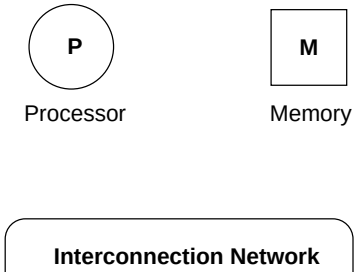
Layers of Abstraction in Parallel Computer Architecture



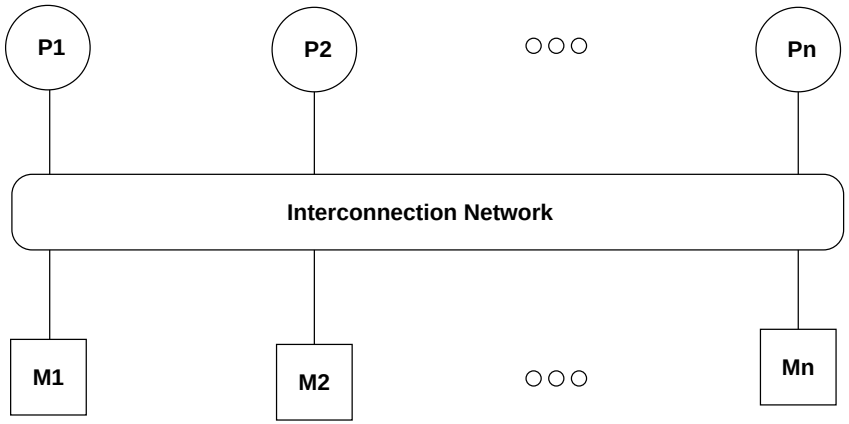
Machine Organization



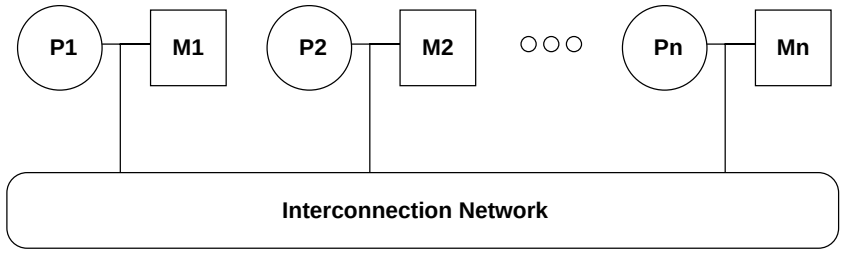
Hardware Elements



Uniform Memory Architecture



Non-Uniform Memory Architecture



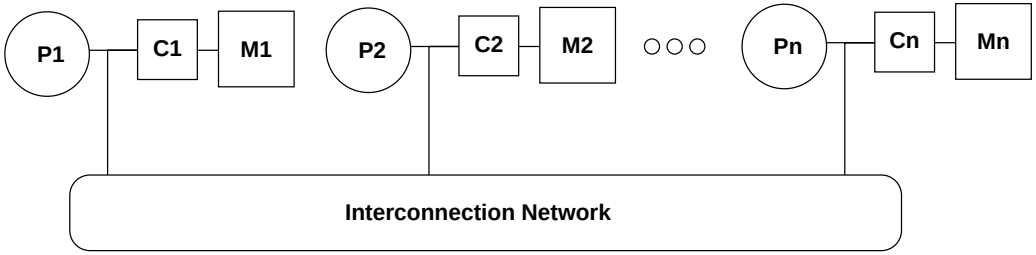
Benefits of Caches

Obviously, we are sorely missing caches in our designs so far.
Benefits of caches:

- Reduced contention for main memory
- Reduced contention for bus
- Lower average memory access time

Shared Cache

One possible configuration is to let processors share a cache.
Reduces accesses to main memory.
May reduce average memory access time.
But there will be heavy contention for the shared cache (we have just moved the contention issue from the main memory bus closer to the processor).

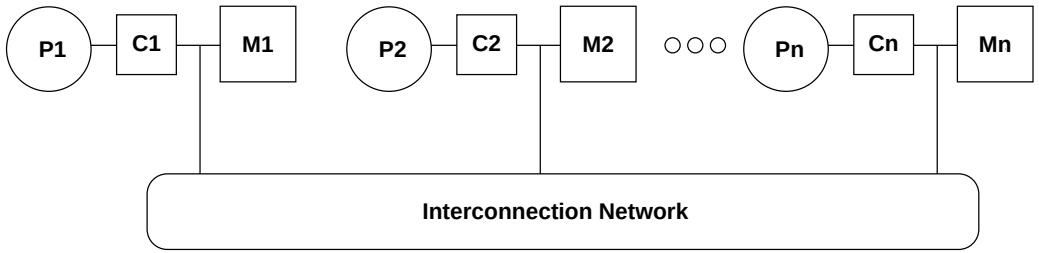


Private Cache

This is by far the more common solution.

Reduces the bandwidth/contention issues while improving average access time.

However, we have introduced an interesting kind of replication (two copies of the same data item in multiple caches)...

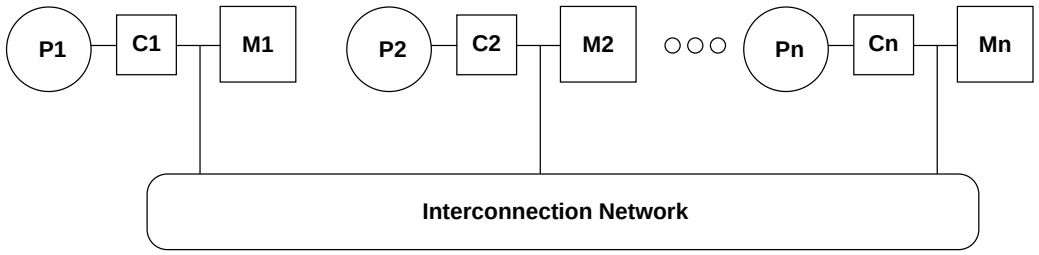


Problem with Replication

What happens when one of the processors writes to a shared data item?

We want other processors to see the new (updated value), not the old (stale) one.

This is known as *cache coherence* and we will focus much of our energy on addressing this issue.



Summary

Today: Communication Architecture and Programming Models

Communication Architecture describes key abstractions and organization of a machine.

Consequently, the communication architecture affects design issues (e.g. naming, operations, order, communication cost, replication)

A programming model is the conceptualization of the machine that the programmer uses.

In general, most communication architectures can support most programming models, but with varying costs and complexities.

Next Time: Parallelization and Programming Models

Parallelization: Parallel algorithm $\rightarrow$ Parallel executable

Programming Models