Multiprocessing and Scalability

A.R. Hurson
Computer Science and Engineering
The Pennsylvania State University
Multiprocessing and Scalability

- Large-scale multiprocessor systems have long held the promise of substantially higher performance than traditional uniprocessor systems.

- However, due to a number of difficult problems, the potential of these machines has been difficult to realize. This is because of the:
Multiprocessing and Scalability

- Advances in technology — Rate of increase in performance of uni-processor,
- Complexity of multiprocessor system design — This drastically effected the cost and implementation cycle.
- Programmability of multiprocessor system — Design complexity of parallel algorithms and parallel programs.
Multiprocessing and Scalability

- Programming a parallel machine is more difficult than a sequential one. In addition, it takes much effort to port an existing sequential program to a parallel machine than to a newly developed sequential machine.

- Lack of good parallel programming environments and standard parallel languages also has further aggravated this issue.
Multiprocessing and Scalability

As a result, absolute performance of many early concurrent machines was not significantly better than available or soon-to-be available uni-processors.
Recently, there has been an increased interest in large-scale or massively parallel processing systems. This interest stems from many factors, including:

- Advances in integrated technology.
- Very high aggregate performance of these machines.
- Cost performance ratio.
- Widespread use of small-scale multiprocessors.
- Advent of high performance microprocessors.
Multiprocessing and Scalability

- Integrated technology
  - Advances in integrated technology is slowing down.
  - Chip density — integrated technology now allows all performance features found in a complex processor be implemented on a single chip and adding more functionality has diminishing returns
 Integrated technology

* Studies of more advanced superscalar processors indicates that they may not offer a performance improvement more than 2 to 4 for general applications.
**Multiprocessing and Scalability**

- **Aggregate performance of machines**
  - Cray X-MP (1983) had a peak performance of 235 MFLOPS.
  - A node in an Intel iPSC/2 (1987) with attached vector units had a peak performance of 10 MFLOPS. As a result, a 256-processor configuration of Intel iPSC/2 would offer less than 11 times the peak performance of a single Cray X-MP.
Multiprocessing and Scalability

- Aggregate performance of machines
  - Today a 256-processor system might offer 100 times the peak performance of the fastest systems.
    - Cray C90 has a peak performance of 952 MFLOPS, while a 256-processor configuration using MIPS R8000 would have a peak performance of 76,800 MFLOPS.
Multiprocessing and Scalability

Economy

- Cray C90 or Convex C4 which exploit IC and packaging technology cost about $1 million.
- High end microprocessor cost somewhere between $2000-5000.
- By using a small number of microprocessors, equal performance can be achieved at a fraction of the cost of more advanced supercomputers.
Due to the advances in technology, multiprocessing has come to the desktops and high-end personal computers. This has led to improvements in parallel processing hardware and software.
In spite of these developments, many challenges need to be overcome in order to exploit the potential of large-scale multiprocessors:

- Scalability
- Ease of programming
**Multiprocessing and Scalability**

- **Scalability**: Maintaining the cost-performance of a uni-processor while linearly increasing overall performance as processors are added.

- **Ease of programming**: Programming a parallel system is inherently more difficult than programming a uni-processor where there is a single thread of control. Providing a single shared address space to the programmer is one way to alleviate this problem — Shared-memory architecture.
**Multiprocessing and Scalability**

- **Shared-memory** eases the burden of programming a parallel machine since there is no need to distribute data or explicitly communicate data between the processors in software.

- **Shared-memory** is also the model used on small-scale multiprocessors, so it is easier to transport programs that have been developed for a small system to a larger shared-memory system.
Multiprocessing and Scalability

◆ A taxonomy of MIMD organizations
   ✫ Two parameters effect the organization of an MIMD system:
   ▪ The way processors communicate with each other,
   ▪ Organization of the global memory
Multiprocessing and Scalability

A taxonomy of MIMD organization

Traditionally, based on the way processors communicate with each other, multiprocessor systems can be classified into:

- Message passing systems (also called distributed memory systems) — Intel iPSC, Paragon nCUBE2, IBM SP1 and SP2
- Shared-memory systems — IBM RP3, Cray X-MP, Cray Y-MP, Cray C90, Sequent Symmetry
Multiprocessing and Scalability

- In message passing multiprocessor systems (distributed memory), processors communicate with each other by sending explicit messages.
- In shared-memory multiprocessor systems the processors are more tightly coupled. Memory is accessible to all processors, and communication among processors is through shared variables or messages deposited in shared memory buffers.
Multiprocessing and Scalability

Message passing System (programmer’s view)

Processors communicate via explicit messages
Multiprocessing and Scalability

**Message passing System** (programmer’s view)

- In a message passing machine, the user must explicitly communicate all information passed between processors. Unless the communication patterns are very regular, the management of this data movement is very difficult.

- Note, multi-computers are equivalent to this organization. It is also referred to as No Remote Memory Access Machine (NORMA).
Multiprocessing and Scalability

- **Shared-memory System** (programmer’s view)

Processors communicate indirectly via shared variables stored in main memory.
Multiprocessing and Scalability

- **Shared-memory System** (programmer’s view)
  - In a shared-memory machine, processors can distinguish communication destination, type, and value through shared-memory addresses. There is no requirement for the programmer to manage the movement of data.
Multiprocessing and Scalability

- Message passing multiprocessor
Multiprocessing and Scalability

- Shared-Memory Multiprocessor

- Interconnection Network

- Global Shared Memory
A shared-memory architecture allows all memory locations to be accessed by every processor. This helps to ease the burden of programming a parallel system, since:

- There is no need to distribute data or explicitly communicate data between processors in the system.
- Shared-memory is the model adopted on small-scale multiprocessors, so it is easier to map programs that have been parallelized for small-scale systems to a larger shared-memory system.
Multiprocessing and Scalability

Inter-processor communication is also critical in comparing the performance of message passing and shared-memory machines:

- Communication in message passing environment is direct and is initiated by the producer of data.

- In shared-memory system, communication is indirect, and producer usually moves data no further than memory. The consumer, then has to fetch the data from the memory — decrease in performance.
In a shared-memory system, communication requires no intervention on the part of a runtime library or operating system. In a message passing environment, access to the network port is typically managed by the system software. This overhead at the sending and receiving processors makes the start up costs of communication much higher (usually 10s to 100s of microsecond).
Multiprocessing and Scalability

As a result of high communication cost in a message passing organization either performance is compromised or a coarser grain parallelism, and hence more limitation on exploitation of available parallelism, must be adapted.

Note that the start up overhead of communication in shared-memory organization is typically on the order of microseconds.
Multiprocessing and Scalability

- Communication in shared-memory systems is usually demand-driven by the consuming processor.
- The problem here is overlapping communication with computation. This is not a problem for small data items. However, it can degrade the performance if there is frequent communication or a large amount of data is exchanged.
Consider the following case, in a message passing environment:

A producer process wants to send 10 words to a consumer process. In a typical message passing environment with blocking send and receive protocol this problem would be coded as:

**Producer process**
Send (`proc_i, process_j, @sbuffer, num-bytes`);

**Consumer process**
Receive (`@rbuffer, max-bytes`);
This code usually is broken into the following steps:

- The operating system checks protections and then programs the network DMA controller to move the message from the sender’s buffer to the network interface.

- A DMA channel on the consumer processor has been programmed to move all messages to a common system buffer. When the message arrives, it is moved from the network interface to the system buffer and an interrupt is posted to the processor.
**Multiprocessing and Scalability**

- The receiving processor services the interrupt and determines which process the message is intended for. It then copies the message to the specified receiver buffer and reschedules the program on the processor’s ready queue.
- The process is dispatched on the processor and reads the message from the user’s receive buffer.
Multiprocessing and Scalability

On a shared-memory system there is no operating system involved and the process can transfer data using a shared data area. Assuming the data is protected by a flag indicating its availability and the size, we have:

Producer process
For i := 0 to num-bytes
    buffer [i] := source [i];
Flag := num-bytes;

Consumer process
While (Flag == 0);
For i := 0 to Flag
    dest [i] := buffer [i];
**Multiprocessing and Scalability**

- For the *message passing* environment, the dominant factors are the operating system overhead; programming the DMA; and the internal processing.

- For the *shared-memory* environment, the overhead is primarily on the consumer reading the data, since it is then that data moves from the global memory to the consumer processor.
Multiprocessing and Scalability

Thus, for a short message the shared-memory system is much more efficient. For long messages, the message-passing environment has similar or possibly higher performance.
Multiprocessing and Scalability

- **Message passing vs. Shared-memory systems**
  - The message passing systems **minimize** the hardware overhead.
  - A **single-thread performance** of a message passing system is as high as a uni-processor system, since memory can be tightly coupled to a single processor.
  - Shared-memory systems are **easier** to program.
Multiprocessing and Scalability

Message passing vs. Shared-memory systems

- Overall, the shared-memory paradigm is preferred since it is easier to use and it is more flexible.
- A shared-memory organization can emulate a message passing environment while the reverse is not possible without a significant performance degradation.
Multiprocessing and Scalability

Message passing vs. Shared-memory systems

- Shared-memory systems offer lower performance than message passing systems if communication is frequent and not overlapped with computation.
- In shared-memory environment, interconnection network between processors and memory usually requires higher bandwidth and more sophistication than the network in message passing environment. This can increase overhead costs to the level that the system does not scale well.
Multiprocessing and Scalability

- Message passing vs. Shared-memory systems
  - Solving the latency problem through memory caching and cache coherence is the key to a shared-memory multiprocessor.
A taxonomy of MIMD organization

In an MIMD organization, the memory can be:
- Centralized
- Distributed

Note it is natural to assume that the memory in message passing system in distributed.
Multiprocessing and Scalability

A taxonomy of MIMD organization

As a result, based on the memory organization, shared memory multiprocessor systems are classified as:

- Uniform Memory Access (UMA) systems, and
- Non-uniform Memory Access (NUMA) systems

Note, most large-scale shared memory machines utilize a NUMA structure.
Multiprocessing and Scalability

- Distributed Shared-Memory Multiprocessor Architecture (NUMA)
Multiprocessing and Scalability

- Distributed Shared-Memory Multiprocessor Architecture (NUMA)
  - NUMA is a shared memory system in which access time varies with the location of the memory word.
Multiprocessing and Scalability

Cache Only Memory Architecture (COMA)

This is a special class of NUMA in which distributed global main memory are caches.
Multiprocessing and Scalability

Cache Only Memory Architecture (COMA)

\[ P_0 \quad \text{Cache} \quad \text{Attraction Memory} \]
\[ P_1 \quad \text{Cache} \quad \text{Attraction Memory} \]
\[ \ldots \]
\[ P_{n-1} \quad \text{Cache} \quad \text{Attraction Memory} \]
Multiprocessing and Scalability

Uniform Memory Access Architecture

Processor

Processor

... 

Processor

Interconnection Network

Memory

Memory

... 

Memory

Fall 2004
Multiprocessing and Scalability

- **Uniform Memory Access Architecture**
  - The physical memory is uniformly shared by all the processors — all processors have equal access time to all memory words.
  - Each processor may use a private cache.
  - UMA is equivalent to tightly coupled multiprocessor class.
  - When all processors have equal access to peripheral devices, the system is called a symmetric multiprocessor.
Multiprocessing and Scalability

• Symmetric Multiprocessor

Processor • Processor • Processor

Interconnection Network

I/O Memory Memory Memory
Multiprocessing and Scalability

◆ Shared Memory Multiprocessor

🌟 To conclude, a practical way to achieve concurrency is through a collection of small to moderate scale processors that provides a global physical address space and symmetric access to all of the main memory of processors and peripheral devices — Symmetric Multiprocessor (SMP).
Symmetric Multiprocessor

Each processor has its own cache, all the processors and memory modules are attached to the same interconnect — Usually a shared bus.

- The ability to access all shared data efficiently and uniformly using ordinary load and store operations, and
- Automatic movement and replication of data in the local caches

makes this organization attractive for concurrent processing.
Multiprocessing and Scalability

Shared Memory Multiprocessor

Since all communication and local computations generate memory accesses in a shared address space, from a system architecture’s perspective, the key high level design issue lies in the organization of memory hierarchy.

In general there are the following choices:
**Multiprocessing and Scalability**

- **Shared Memory Multiprocessor — Shared cache**

- Processor nodes: $P_0$, $P_1$, ..., $P_{n-1}$
- Interconnection Network
- Interleaved Main Memory
- Interleaved (1st level) cache
- Switch
Multiprocessing and Scalability

- **Shared Memory Multiprocessor** — Shared cache
  - This platform is more suitable for small scale configuration — 2 to 8 processors.
  - It is a more common approach for multiprocessor on chip.
Multiprocessing and Scalability

- **Shared Memory Multiprocessor** — Bus based

![Diagram of a shared memory multiprocessor with processors P₀, P₁, ..., Pₙ₋₁, caches, memory, and I/O connected through a bus.]
Multiprocessing and Scalability

**Shared Memory Multiprocessor** — Bus based Shared Memory

- This platform is suitable for small to medium configuration — 20 to 30 processors.
- Due to its simplicity it is very popular.
- Bandwidth of the shared bus is the major bottleneck of the system.
- This configuration also is known as Cache Coherent Shared Memory Configuration.
Multiprocessing and Scalability

- Shared Memory Multiprocessor — Dance Hall

- Interconnection Network

- Memory

- Cache
Multiprocessing and Scalability

- **Shared Memory Multiprocessor — Dance Hall**
  - This platform is symmetric and scalable.
  - All memory modules are far away from processors.
  - In large configurations, several hops are needed to access memory.
Multiprocessing and Scalability

- Shared Memory Multiprocessor — Distributed Memory
Multiprocessing and Scalability

Shared Memory Multiprocessor — Distributed Memory

* This configuration is not symmetric, as a result, locality should be exploited to improve performance.
As noted, it is possible to build a shared-memory machine with a distributed-memory organization. Such a machine has the same structure as the message passing system, but instead of sending messages to other processors, every processor can directly address both its local memory and remote memories of other processors. This model is referred to as distributed-shared-memory (DSM) organization.
Multiprocessing and Scalability

- **Shared Memory Multiprocessor**
  - distributed-shared-memory (DSM) organization is also called non-uniform memory access architecture (NUMA).
  - This is in contrast to the uniform memory access structure (UMA) as shown before.
  - UMA is easier to program than NUMA.
  - Most of the small-scale shared-memory systems are based on UMA philosophy and large-scale shared-memory systems are based on NUMA.
Multiprocessing and Scalability

**Shared Memory Multiprocessor**

In most shared-memory organizations, processors and memory are separated by an interconnection network.

- For small-scale shared-memory systems, the interconnection network is a simple bus.
- For large-scale shared-memory systems, similar to the message passing systems, we use multistage networks.
Multiprocessing and Scalability

- **Shared Memory Multiprocessor**
  - In all shared memory multiprocessor organization, the concept of cache is becoming very attractive in reducing the memory latency and the required bandwidth.
  - In all design cases, except shared cache, each processor has at least one level of private cache. This raises the issue of cache coherence as an attractive research directions.