

## **4. SOFTWARE**

A number of concurrent programming languages, libraries, APIs, and parallel programming models have been created for programming parallel computers.

These can generally be divided into classes based on the assumptions they make about the underlying memory architecture—shared memory, distributed memory, or shared distributed memory. Shared memory programming languages communicate by means of manipulating shared memory variables. Distributed memory uses message passing. POSIX Threads and OpenMP are two of most widely used shared memory APIs, whereas Message Passing Interface (MPI) is the most widely used message passing system API. One concept used in programming parallel programs is the future concept, where one part of a program promises to deliver a required datum to another part of a program at some future time.

### **4.1 Automatic parallelization**

Automatic parallelization of a sequential program by a compiler is the "holy grail" of parallel computing. Despite decades of work by compiler researchers, automatic parallelization has had only limited success.

Mainstream parallel programming languages remain either explicitly parallel or (at best) partially implicit with the programmer giving the compiler directives for parallelization. A few fully implicit parallel programming languages exist - SISAL, Parallel Haskell, and (for FPGAs) Mittrion-C - but these are niche languages that are not widely used.

### **4.2 Application checkpointing**

The larger and more complex a computer gets, the more can go wrong, and the smaller the mean time between failures becomes. Application checkpointing is a technique whereby the computer system takes a "snapshot" of the application—a record of all current resource allocations and variable states, akin to a core dump. This information can then be used to restore the program if the computer should fail. Application checkpointing means that the program will only have to restart from its last checkpoint, rather than the beginning. For an application that may take months, this is critically important. Application checkpointing may also be used to facilitate process migration.

### 4.3 History

The origins of true (MIMD) parallelism go back to Federico Luigi, Conte Menabrea and his "Sketch of the Analytic Engine Invented by Charles Babbage." IBM introduced the 704 in 1954, through the project in which Gene Amdahl was one of the principal architects. It became the first commercially available computer to use fully automatic floating point arithmetic commands. In 1958, IBM researchers John Cocke and Daniel Slotnick discussed the use of parallelism in numerical calculations for the first time. Burroughs Corporation introduced the D825 in 1962, a four-processor computer that accessed up to 16 memory modules through a crossbar switch. In 1967, Amdahl and Slotnick published a debate about the feasibility of parallel processing at American Federation of Information Processing Societies Conference. Amdahl's argument about limits to parallelism became called as "Amdahl's Law".

In 1969, US company Honeywell introduced its first Multics system, a symmetric multiprocessor system capable of running up to eight processors in parallel. C.mmp, a 1970s multi-processor project at Carnegie Mellon University, was "among the first multiprocessors with more than a few processors." "The first bus-connected multi-processor with snooping caches was the Synapse N+1 in 1984".

SIMD parallel computers can be traced back to the 1970s. The motivation behind early SIMD computers was to amortize the gate delay of the processor's control unit over multiple instructions. Earlier, in 1964, Slotnick had proposed building a massively-parallel computer for the Lawrence Livermore National Laboratory. His design was funded by the US Air Force, which materialized as the earliest SIMD parallel computing effort, ILLIAC IV. Key to its design was a fairly high parallelism with up to 256 processors, used to allow the machine to work on large data sets in what would later be known as vector processing. However, ILLIAC IV was called "the most infamous of Supercomputers" because the project was built only one-fourth to completion, but took 11 years to build and cost almost four times its original estimated cost. When it was finally ready to run its first real application in 1976, it was outperformed by existing commercial supercomputers like the Cray-1.

## References

1. G. S. Almasi and A. Gottlieb. Highly Parallel Computing. Benjamin-Cummings publishers, Redwood city, CA, 1989
2. Krste Asanovic et al. The Landscape of Parallel Computing Research: A View from Berkeley. University of California, Berkeley. Technical Report No. UCB/EECS-2006-183. December 18, 2006: "Old [conventional wisdom]: Increasing clock frequency is the primary method of improving processor performance. New [conventional wisdom]: Increasing parallelism is the primary method of improving processor performance... Even representatives from Intel, a company generally associated with the "higher clock-speed is better" position, warned that traditional approaches to maximizing performance through maximizing clock speed have been pushed to their limit."
3. David A. Patterson and John L. Hennessy. Computer Organization and Design (Second Edition) Morgan Kaufmann Publishers, 1998. ISBN 1558604286, pg 715
4. Asanovic et al: Old [conventional wisdom]: Power is free, but transistors are expensive. New [conventional wisdom] is [that] power is expensive, but transistors are "free".
5. a b Blaise Barney. Introduction to Parallel Computing. Lawrence Livermore National Laboratory. Retrieved on 2007-11-09.
6. John L. Hennessy and David A. Patterson. Computer Architecture: A Quantitative Approach. 3rd edition, 2002. Morgan Kaufmann, ISBN 1558607242. Page 43.
7. J. M. Rabaey. Digital Integrated Circuits. Prentice Hall, 1996.
8. Laurie J. Flynn. Intel Halts Development of 2 New Microprocessors. New York Times, May 8, 2004.
9. G. Amdahl. The validity of the single processor approach to achieving large-scale computing capabilities. In Proceedings of AFIPS Spring Joint Computer Conference, pages 483–485, Atlantic City, N.J., April 1967. AFIPS Press.
10. Reevaluating Amdahl's Law Communications of the ACM 31(5), 1988. pp. 532-533
11. A. J. Bernstein, "Program Analysis for Parallel Processing,' IEEE Trans. on Electronic Computers, EC-15, Oct 66, 757-762.
12. K. Hwang and F. A. Briggs. Computer architecture and parallel processing. McGraw-Hill, 1984.
13. Leslie Lamport. "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs", IEEE Transactions on Computers, C-28,9 (September 1979), 690–691.
14. Patterson and Hennessy, pg 748
15. David E. Culler, Jaswinder Pal Singh, Anoop Gupta. Parallel Computer Architecture - A Hardware/Software Approach. Morgan Kaufmann Publishers, 1999. ISBN 1558603433, pg 15
16. Culler et al, pg 15
17. Yale Patt. "The Microprocessor Ten Years From Now: What Are The Challenges, How Do We Meet Them? (wmv). Distinguished Lecturer talk at Carnegie Mellon University, April 2004. Retrieved on November 7, 2007.
18. Culler et al, pg 124
19. Culler et al, pg 125
20. Patterson and Hennessy, pg 713
21. Hennessy and Patterson, pg 549
22. Patterson and Hennessy, pg 714

23. What is clustering? Webopedia computer dictionary. Retrieved on November 7, 2007.
24. Beowulf definition. PC Magazine. Retrieved on November 7, 2007.
25. Architecture share for 06/2007. TOP500 Supercomputing Sites. Clusters make up 74.60% of the machines on the list. Retrieved on November 7, 2007.
26. Hennessy and Patterson, pg 537
27. MPP Definition. PC Magazine. Retrieved on November 7, 2007.
28. Michael R. D'Amour, CEO DRC Computer Corporation. "Standard Reconfigurable Computing" Invited speaker at the University of Delaware, February 28, 2007
29. Sha'Kia Boggan and Daniel M. Pressel. GPUs: An Emerging Platform for General-Purpose Computation (PDF). ARL-SR-154, U.S. Army Research Lab. August 2007. Retrieved on November 7, 2007.
30. Oleg Maslennikov (2002). Systematic Generation of Executing Programs for Processor Elements in Parallel ASIC or FPGA-Based Systems and Their Transformation into VHDL-Descriptions of Processor Element Control Units. *Lecture Notes in Computer Science*, 2328/2002:272.
31. Y. Shimokawa, Y. Fuwa, N. Aramaki. A parallel ASIC VLSI neurocomputer for a large number of neurons and billion connections per second speed. *IEEE International Joint Conference on Neural Networks*, 18-21 November 1991. 3: 2162–2167.
32. K.P. Acken, M.J. Irwin, R.M. Owens. A Parallel ASIC Architecture for Efficient Fractal Image Coding. *The Journal of VLSI Signal Processing*, July 1998, 19(2):97–113(17)
33. Andrew B. Kahng. "Scoping the Problem of DFM in the Semiconductor Industry." University of California, San Diego. June 21, 2004: "Future design for manufacturing (DFM) technology must reduce design [non-recoverable expenditure] cost and directly address manufacturing [non-recoverable expenditures] – the cost of a mask set and probe card – which is well over \$1 million at the 90 nm technology node and creates a significant damper on semiconductor-based innovation."
34. Patterson and Hennessy, pg 751
35. L.F. Menabrea, Sketch of the Analytic Engine Invented by Charles Babbage. *Bibliothèque Universelle de Genève*, 1842. Retrieved on November 7, 2007.
36. Patterson and Hennessy, pg 753
37. da Cruz, Frank (2003). *Columbia University Computing History: The IBM 704*. Columbia University. Retrieved on 2008-01-08.
38. Wilson, Gregory V. (1994). *The History of the Development of Parallel Computing*. Retrieved on 2008-01-08.
39. Anthes, Gary (2001-11-19). *The Power of Parallelism*. *Computerworld*. Retrieved on 2008-01-08.
40. Patterson and Hennessy, pg 749
41. Patterson and Hennessy, pgs 749–750: Although successful in pushing several technologies useful in later projects, the ILLIAC IV failed as a computer. Costs escalated from the \$8 million estimated in 1966 to \$31 million by 1972, despite the construction of only a quarter of the planned machine... It was perhaps the most infamous of supercomputers. The project started in 1965 and ran its first real application in 1976

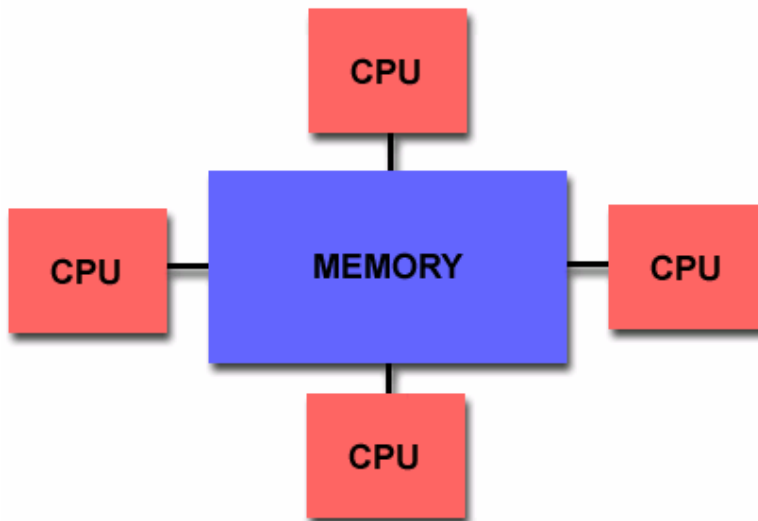
Note: Chapter 1- 4 is modified from Wikipedia.

## 5. Parallel Computer Memory Architectures

### 5.1 Shared Memory

#### General Characteristics:

Shared memory parallel computers vary widely, but generally have in common the ability for all processors to access all memory as global address space.



#### Shared memory architecture

Multiple processors can operate independently but share the same memory resources.

Changes in a memory location effected by one processor are visible to all other processors.

Shared memory machines can be divided into two main classes based upon memory access times: UMA and NUMA.

#### Uniform Memory Access (UMA):

Most commonly represented today by Symmetric Multiprocessor (SMP) machines  
Identical processors

Equal access and access times to memory

Sometimes called CC-UMA - Cache Coherent UMA. Cache coherent means if one processor updates a location in shared memory, all the other processors know about the update. Cache coherency is accomplished at the hardware level.

#### Non-Uniform Memory Access (NUMA):

Often made by physically linking two or more SMPs

One SMP can directly access memory of another SMP

Not all processors have equal access time to all memories

Memory access across link is slower

If cache coherency is maintained, then may also be called CC-NUMA - Cache Coherent NUMA

**Advantages:**

- Global address space provides a user-friendly programming perspective to memory
- Data sharing between tasks is both fast and uniform due to the proximity of memory to CPUs

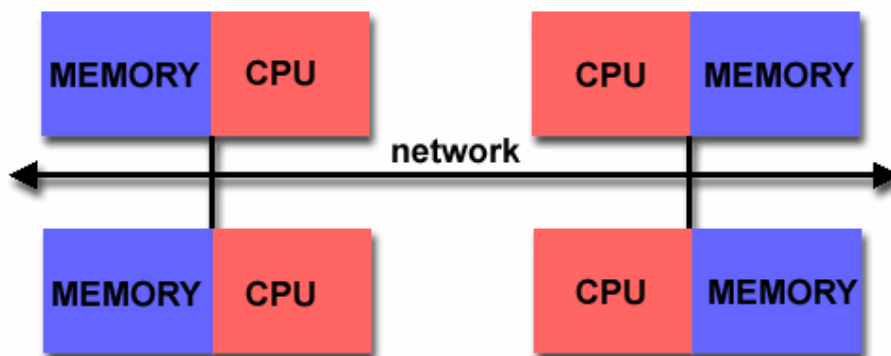
**Disadvantages:**

- Primary disadvantage is the lack of scalability between memory and CPUs. Adding more CPUs can geometrically increase traffic on the shared memory-CPU path, and for cache coherent systems, geometrically increase traffic associated with cache/memory management.
- Programmer responsibility for synchronization constructs that insure "correct" access of global memory.
- Expense: it becomes increasingly difficult and expensive to design and produce shared memory machines with ever increasing numbers of processors.

**5.2 Distributed Memory**

**General Characteristics:**

Like shared memory systems, distributed memory systems vary widely but share a common characteristic. Distributed memory systems require a communication network to connect inter-processor memory.



**Distributed memory architecture**

Processors have their own local memory. Memory addresses in one processor do not map to another processor, so there is no concept of global address space across all processors.

Because each processor has its own local memory, it operates independently. Changes it makes to its local memory have no effect on the memory of other processors. Hence, the concept of cache coherency does not apply.

When a processor needs access to data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. Synchronization between tasks is likewise the programmer's responsibility.

The network "fabric" used for data transfer varies widely, though it can be as simple as Ethernet.

**Advantages:**

- Memory is scalable with number of processors. Increase the number of processors and the size of memory increases proportionately.
- Each processor can rapidly access its own memory without interference and without the overhead incurred with trying to maintain cache coherency.
- Cost effectiveness: can use commodity, off-the-shelf processors and networking.

**Disadvantages:**

- The programmer is responsible for many of the details associated with data communication between processors.
- It may be difficult to map existing data structures, based on global memory, to this memory organization.
- Non-uniform memory access (NUMA) times

**5.3 Hybrid Distributed-Shared Memory**

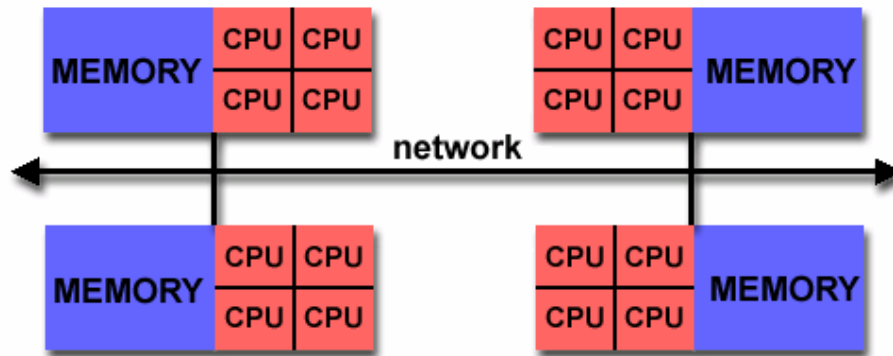
Summarizing a few of the key characteristics of shared and distributed memory machines:

<b>Comparison of Shared and Distributed Memory Architectures</b>			
<b>Architecture</b>	<b>CC-UMA</b>	<b>CC-NUMA</b>	<b>Distributed</b>
<b>Examples</b>	SMPs Sun Vexx DEC/Compaq SGI Challenge IBM POWER3	SGI Origin Sequent HP Exemplar DEC/Compaq IBM POWER4 (MCM)	Cray T3E Maspar IBM SP2
<b>Communications</b>	MPI Threads OpenMP shmem	MPI Threads OpenMP shmem	MPI
<b>Scalability</b>	to 10s of processors	to 100s of processors	to 1000s of processors
<b>Draw Backs</b>	Memory-CPU bandwidth	Memory-CPU bandwidth Non-uniform access times	System administration Programming is hard to develop and maintain
<b>Software Availability</b>	many 1000s ISVs	many 1000s ISVs	100s ISVs

The largest and fastest computers in the world today employ both shared and distributed memory architectures.

### Hybrid memory architecture

The shared memory component is usually a cache coherent SMP machine. Processors on a given SMP can address that machine's memory as global.



The distributed memory component is the networking of multiple SMPs. SMPs know only about their own memory - not the memory on another SMP. Therefore, network communications are required to move data from one SMP to another.

Current trends seem to indicate that this type of memory architecture will continue to prevail and increase at the high end of computing for the foreseeable future.

Advantages and Disadvantages: whatever is common to both shared and distributed memory architectures.