



Parallel Processing

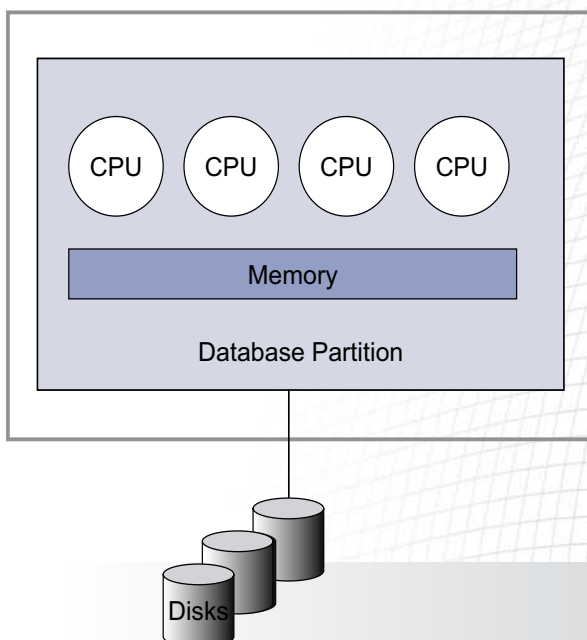
“To pull a bigger wagon, it is easier to add more horses than to grow a gigantic horse.” This paraphrased quotation nicely expresses the basic concept of parallel processing. The speed of sequential computers has been doubling every eighteen months, according to Moore’s law. However, at any given time, that speed is limited by the state of the art in integrated circuit design and manufacturing. To circumvent that limitation, it is possible to split a given computationally intensive task among multiple processors working simultaneously.

There are several different flavors of parallel architecture available. The first, originally developed by Cray, is data parallelism or vector processing. With this, an entire array of numbers can be operated at once rather than treating each element one by one. An intelligent compiler is all that is needed for this level of parallelism.

SMP

The next to be developed was Symmetric MultiProcessing (SMP) often called shared memory processing, where some number of central processing units (cpus) share the same memory. This architecture, exemplified by the SGI Power Challenge, is practically limited to a maximum of 32 cpus. Implicit parallel applications on SMP platforms typically do not scale well beyond 8 to 16 cpus.

SMP machine



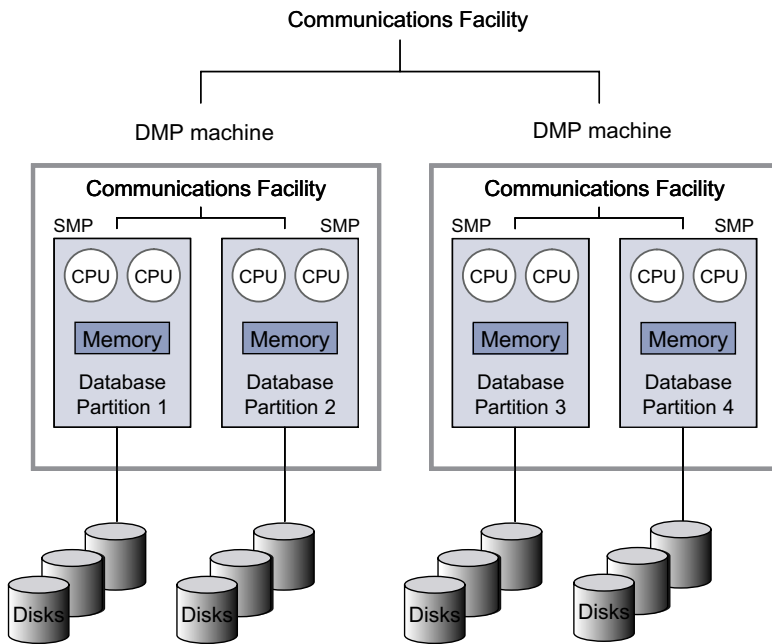
The Open Multi-Processing, or OpenMP, software standard was developed for managing shared memory parallelism. The first standard for FORTRAN was released in October 1997. Prior to this, proprietary systems were in use. A parallelized software process using this standard starts off as a single thread. When it comes to a parallelized section of the code, it forks to multiple threads. These threads are then joined back together when the next sequential section of code needs to be executed. The forking and joining operations are controlled by compiler directives in the software.

DMP

Distributed Memory Processing (DMP) is an alternative architecture where each cpu accesses its own memory. Data is shared between processors through message passing. The send and receive operations require action by the processors on both ends of the communication. The Message-Passing Interface, or MPI, is the software standard that has been developed by an industry/government consortium. MPI encompasses most of the capabilities of the proprietary systems that preceded it. The first standard was completed in May 1994. Using MPI, it is possible to design parallel applications that are scalable to a greater number of cpus. The programming effort, however, can be quite substantial.

Combined

Increasingly popular are combined SMP/DMP architectures where each compute node contains 2 to 4 cpus which share memory. These compute nodes are then linked together through high speed switches. MPI can take advantage of the shared memory to accelerate message passing between processes on the same node while simultaneously controlling the message passing between compute nodes. An example of this combined architecture would be a Beowulf cluster where each compute node had two processors. The principles of Beowulf clusters are that the compute nodes be commodity, off-the-shelf machines and that the operating system be open source, such as Linux. With a high speed network connecting the compute nodes, it is possible to achieve supercomputer performance on a shoe-string budget.



Domain decomposition

In a finite element application, such as ProCAST, it is necessary to divide up the mesh according to the number of processors. This is called domain decomposition or partitioning. At the border of each sub-group of nodes and elements, it is necessary to communicate with nodes that are assigned to a different processor. These are referred to as ghost nodes. The domain decomposition needs to be done in such a way that two objectives are met: 1) Load balancing, so that each cpu has about the same amount of work to do and 2) Communication minimization so that there are the least number of ghost nodes between partitions. When the solution domain is changing, such as the flow domain during filling, it is necessary to do dynamic domain decomposition to keep the load balanced. Fortunately, the domain decomposition routines themselves are parallelized, so the cost of the repartitioning is not too high.

Performance

The performance of a parallelized application is rated by its efficiency or speedup. If T_s is the time it takes to run a serial job and T_p is the time it takes to run the same job on P processors, then the efficiency is defined as,

$$E = T_s / (P * T_p)$$

An efficiency of one means that the application scales perfectly. Speedup is defined as,

$$S_p = T_s / T_p = E * P$$

Thus, if the efficiency is one, then the speedup is the same as P , the number of processors.

An efficiency of one is very difficult to achieve in practice because of the communications overhead and because of those sections of a program that cannot be parallelized. If s is the fraction of a program that operates in a serial mode (or the fraction of cpu time lost in communications), then according to Amdahl's Theorem,

$$S_{p,max} = (s + p) / (s + p/P) = 1 / (s + (1 - s)/P) \leq 1/s$$

If $s = .1$, the maximum speedup is 10. Running such an application on more than 8 processors would not be beneficial.

Obviously, it is important to make this fraction s as small as possible. In terms of hardware assistance, the communications overhead can be minimized through the use of high speed network switches. The important network metrics are latency and bandwidth. Latency refers to the communications initialization time and can be considered as the time it takes to send one byte of data. Bandwidth is the data rate of communications, or the bytes per second. The ideal is to have a high bandwidth and a low latency. An example would be the Myrinet network components produced by Myricom. These advertise a bandwidth of 489 MBps (3912 Mbps) for two-way transfer and a latency of 6.3 micro-seconds. Note the difference between MBps, megabytes per second, and Mbps, megabits per second. Both units are used for quoting bandwidth. It is also fairly common to find now Gigabit Ethernet switches and even some 10 Gbps products are starting to appear.

It is actually possible to achieve superlinear speedup, which means that the efficiency can be greater than one. This seems impossible, but it is due to the fact that, as the problem size is divided into smaller parcels, relatively more of the data fits into the cache memory of each processor. The cache memory is considerably faster than RAM, so the cpu can operate more efficiently. Of course, the application also needs to be well parallelized in order to obtain this exciting conclusion.