Choosing the right HPC approach(es) means appropriately addressing complex and computationally intensive operations in a cost-effective manner. This paper focuses on addressing the impact of current hardware design, the approaches to grid and cloud computing, and considerations in choosing HPC solutions.

High Performance Computing

What is HPC?

High Performance Computing (HPC) refers to a computation platform used to address the performance challenges inherent to solving problems in scientific research and other computationally intensive fields. Currently, the HPC field can be further defined and characterized by the approach to solving problems. In some cases, the term HPC is likely to be used for problems that are highly parallel, exhibit a relatively high degree of coupling and require low latency connections. This type of computing is often implemented through technologies such as Message Passing Interface (MPI).

High Throughput Computing (HTC) defines scenarios where the elements can be widely distributed and accept high latency. Examples of implementations include World Community Grid and Seti@Home.

Many-Task Computing (MTC) is a relatively new definition that bridges the gap between HTC and HPC. MTC recognizes that there is a class of problems made up of many tasks that must be managed as a group to achieve a meaningful result. These tasks are loosely coupled and may or may not have inter-task dependencies, presenting potentially significant scheduling challenges.

Impact of Current Hardware Design

Historically, HPC was associated with names such as Cray Computing and Vector Processing Hardware. This field eventually began to encounter limitations. This includes the number of vector units that could be built into a single system, power consumption, clock speed limitations and the issue that not every large scale computational model is best expressed in vector operations.

"There are many tools and techniques available to assist in implementing HPC solutions, such as high performance math libraries that abstract CPU family difference, high performance communication libraries, multithreading libraries and debugging tools. The fundamental issues of algorithm design and recognizing which tools to apply still remain."

HPC hardware transitioned to parallel architectures. We now have the multi-core Central Processing Unit (CPU), which supports hardware as a commodity item. A laptop computer with four physical CPU cores can be readily purchased at the local electronics store.

At larger scales, advances in switching fabrics in communication have produced new ways to cross-connect large numbers of CPUs, reducing the bottleneck of communication scalability. Advances in network management tools and techniques, and the advent of managing large numbers of cloud-based compute resources, have reduced the complexity and costs of managing large computer clusters.

Along with the enabling trends that have allowed us to cost effectively manage large parallel compute clusters, the advances in circuit design and fabrications have allowed special purpose instruction to become available on mainstream CPUs. Most modern CPUs have seen the addition of special purpose instruction sets that enhance native vector processing capabilities, and were originally added to support multi-media processing. Simultaneous Instruction Multiple Data (SIMD) instructions are now available for scientific computing purposes.

A natural extension of special purpose instruction sets is those implemented in graphics chips. Modern graphics chipset have powerful vector and parallel capabilities – so much so that graphics processor vendors such as NVidia and ATI support a market for General Purpose Graphics Processing Unit (GPGPU) processing.

The current CPU design trends enable parallel computation on commodity hardware; at the level of single computers and with compute clusters. The key to cost effective HPC is implementation applications that leverage this parallel capability, and make use of specialty hardware such as GPGPU where it is effective.

Grid Computing

Grid computing refers to a self-managed set of computers composed of a combination of grid-dedicated computers and computers that can donate spare cycles. This implies that there will be an upper limit on the number of compute nodes available.

Grids are often positioned to capture wasted compute cycles by dispatching compute tasks to underutilized computers. In reality, the wasted cycles may be so limited that the job management costs will not be recovered. If useful cycles are recovered (e.g., overnight usage of desktop computers) consideration must be given to the issues around lost time due to IT management (e.g., software upgrades) and the impact of the unpredictable duration of time-sensitive computation.

"The goal with grid computing is to complete a number of work units rather than minimizing elapsed time for a single problem."

Grid computing lends itself more to batch processes rather than real-time deadlines. The goal with grid computing is to complete a number of work units rather than minimizing elapsed time for a single problem (e.g., similar to HTC).

Cloud Computing

Cloud computing is conceptually a superset of grid computing with many compute nodes. However, node management is the cloud vendor's issue. Cloud computing can potentially access a much greater number of compute nodes than can grid computing, depending on the circumstances.

The cloud compute nodes are virtual machine instances. Virtualization has the advantage of supplying additional standard machine instances with very low deployment costs. However, virtualization overhead may impact compute times and the virtual machine image may not support the specialty instruction sets discussed earlier.

Data locality may also be an issue for cloud computing depending on the scope of the problem. Consideration must be given to data access latency times. For instance, how close are your compute nodes and your data, and to what extent does this matter for your problem?

Problem Characterization

HPC has always presented challenges in matching algorithm design to the available compute infrastructure. These challenges are here to stay. In fact, the increase in infrastructure architectures has presented additional questions that must be examined.

The following historical algorithmic macro challenges must be addressed:

- To what degree can the overall problem be broken up into parallel tasks, and to what extent can task serialization be minimized?
- What size of input data must be supplied to the tasks, and how can this be done efficiently?
- What degree of coupling exists between tasks and how does this data flow?
- How much output data does each task produce, and how must this data be consolidated to produce the desired end result?

About Quadrus

Quadrus is a recognized leader in IT professional services and solutions. Headquartered in Calgary, Alberta, Quadrus has delivered hundreds of successful projects across Western Canada since 1993. We are committed to providing the highest quality service to our valued clients.

Contact us:
info@quadrus.com
www.quadrus.com
+1 (403) 257 0850

In addition, algorithm design must address the following:

- What is the best way to make effective use of CPU resources without producing a solution that is tightly coupled to a single hardware platform? Considerations include register set size, availability of specialty instruction sets (e.g., SIMD), and effective use of memory caching hardware.
- Effective use of off-chip hardware resources, principally memory and communications bandwidth.
- Effective multi-threading architecture that provides multiple CPU cores to a single compute task.
- Evaluation of the use of specialty hardware such as GPGPU or programmable hardware such as Field Programmable Grid Arrays.

There are many tools and techniques available to assist in implementing HPC solutions, such as high performance math libraries that abstract CPU family difference, high performance communication libraries, multithreading libraries and debugging tools. The fundamental issues of algorithm design and recognizing which tools to apply still remain. Choosing the right HPC approach(es) means appropriately addressing complex and computationally intensive operations in a cost-effective manner.