22S:166

Parallel computing

Lecture 31 Nov. 20, 2009

Kate Cowles
374 SH
kcowles@stat.uiowa.edu

What is Parallel Computing?

Traditionally, software has been written for serial computation:

- To be run on a single computer having a single Central Processing Unit (CPU);
- A problem is broken into a discrete series of instructions.
- Instructions are executed one after another.
- Only one instruction may execute at any moment in time.

Parallel computing

- In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem.
  - To be run using multiple CPUs
  - A problem is broken into discrete parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
  - Instructions from each part execute simultaneously on different CPUs
- The compute resources can include:
  - A single computer with multiple processors;
  - An arbitrary number of computers connected by a network;
  - A combination of both.

- The computational problem usually demonstrates characteristics such as the ability to be:
  - Broken apart into discrete pieces of work that can be solved simultaneously;
  - Execute multiple program instructions at any moment in time;
  - Solved in less time with multiple compute resources than with a single compute resource.
- Traditionally, parallel computing has been considered to be "the high end of computing" and has been motivated by numerical simulations of complex systems and "Grand Challenge Problems" such as:
  - weather and climate
  - chemical and nuclear reactions
  - biological, human genome
  - geological, seismic activity
  - mechanical devices - from prosthetics to spacecraft
  - electronic circuits
  - manufacturing processes

Commercial applications are providing an equal or greater driving force in the development of faster computers

- parallel databases, data mining
- oil exploration
- web search engines, web based business services
- computer-aided diagnosis in medicine
- management of national and multi-national corporations
- advanced graphics and virtual reality, particularly in the entertainment industry
- networked video and multi-media technologies
- collaborative work environments

Why Use Parallel Computing?

- The primary reasons for using parallel computing:
  - Save time - wall clock time
  - Solve larger problems
  - Provide concurrency (do multiple things at the same time)
- Other reasons might include:
  - Taking advantage of non-local resources - using available compute resources on a wide area network, or even the Internet when local compute resources are scarce.
  - Cost savings - using multiple "cheap" computing resources instead of paying for time on a supercomputer.
  - Overcoming memory constraints - single computers have very finite memory resources. For large problems, using the memories of multiple computers may overcome this obstacle.

General terminology in Parallel Computing

- Task: A logically discrete section of computational work. A task is typically a program or program-like set of instructions that is executed by a processor.
- Parallel Task: A task that can be executed by multiple processors safely (yields correct results)
- Serial Execution: Execution of a program sequentially, one statement at a time. In the simplest sense, this is what happens on a one processor machine. However, virtually all parallel tasks will have sections of a parallel program that must be executed serially.
- Parallel Execution: Execution of a program by more than one task, with each task being able to execute the same or different statement at the same moment in time.
- Shared Memory: From a strictly hardware point of view, describes a computer architecture where all processors have direct (usually bus based) access to common physical memory. In a programming sense, it describes a model where parallel tasks all have the same "picture" of memory and can directly address and access the same logical memory locations regardless of where the physical memory actually exists.

- Distributed Memory: In hardware, refers to network based memory access for physical memory that is not common. As a programming model, tasks can only logically "see" local machine memory and must use communications to access memory on other machines where other tasks are executing.
- Communications Parallel tasks typically need to exchange data. There are several ways this can be accomplished, such as through a shared memory bus or over a network, however the actual event of data exchange is commonly referred to as communications regardless of the method employed.
- Synchronization The coordination of parallel tasks in real time, very often associated with communications. Often implemented by establishing a synchronization point within an application where a task may not proceed further until another task(s) reaches the same or logically equivalent point.

  Synchronization usually involves waiting by at least one task, and can therefore cause a parallel application's wall clock execution time to increase.

- Granularity: In parallel computing, granularity is a qualitative measure of the ratio of computation to communication.
  - Coarse: relatively large amounts of computational work are done between communication events
  - Fine: relatively small amounts of computational work are done between communication events
- Observed Speedup: Observed speedup of a code which has been parallelized, defined as:

  wall-clock time of serial execution/ wall-clock time of parallel execution

  One of the simplest and most widely used indicators for a parallel program's performance.

- Parallel Overhead: The amount of time required to coordinate parallel tasks, as opposed to doing useful work. Parallel overhead can include factors such as:
  - Task start-up time
  - Synchronizations
  - Data communications
  - Software overhead imposed by parallel compilers, libraries, tools, operating system, etc.
  - Task termination time
- Scalability: Refers to a parallel system's (hardware and/or software) ability to demonstrate a proportionate increase in parallel speedup with the addition of more processors. Factors that contribute to scalability include:
  - Hardware - particularly memory-cpu bandwidths and network communications
  - Application algorithm
  - Parallel overhead related
  - Characteristics of your specific application and coding