What is High Throughput Computing?

There is no strict definition of an HTC application. Computer Scientists tend to define HTC in terms of how it is different from High Performance or Parallel Computing. Wikipedia suggests that the main differences have to do with execution times and coupling. Most parallel applications are tightly coupled[1], while HTC tends to be very loosely coupled[2]. More generally, we tend to say that a true parallel application is a collection of computational components all running at the same time and cooperatively working to solve a single problem—in essence it is a single large application split among a number of computational resources. In contrast, an HTC application is really a number of identical programs each running simultaneously on a group of computers and each working on a different set of input data.

Sometimes called ‘‘bag-of-tasks’’ or ‘‘parameter sweep’’ applications, HTC jobs can more formally be described in terms of sets of inputs and associated results. Consider the set of inputs $X = \{x_1, x_2, \ldots x_n\}$. Any given input x_i represents some arbitrary collection of files and/or command line parameters used by a sequential program which implements the function $\boldsymbol{f}$ such that it produces the result $\boldsymbol{r_i}$. Therefore, the HTC application is defined to fill in the result set $R = \{r_1, r_2, \ldots r_n\}$ such that $r_i = f(x_i) \mid \forall x_i \in X$. Another way of saying this is that for all inputs of interest, a program is run for each input that produces a corresponding resultant output. Thus, the HTC application is the conglomeration of these mappings.

Many HTC applications begin as a single sequential program that someone writes to solve a problem or answer a question. For example, what is the lift produced by this wing configuration? How similar is this protein sequence to that of a frog’s? What does the computer generated scene look like for this frame of this movie? Each of these programs can run as single job on a single machine. They become HTC applications when someone turns around decides to run the program 1000 times with different inputs. The questions change accordingly. What does the space of solutions for wing configurations look like? Which protein sequence is closest to my sample? What does the entire scene from the movie look like?

In HTC applications, particularly when there are hundreds or thousands of jobs to complete, the performance metric of interest is jobs per hour or total time to complete all of the jobs rather than the time to completion of any single job. This makes HTC applications particularly well suited to run on large numbers of slower, inexpensive, machines rather than a small number of very expensive and fast machines.

The reason is simple. As a simplistic example, suppose I can use either a fast, 10 cents per core hour, machine that can complete on average two jobs per hour, or I can use a machine that costs one penny per core hour, but is half as fast and can only complete one job per hour. Assume I can get as many cores as I want of either machine and I have 1000 jobs to complete.

Which should I choose? The answer is obvious. It costs five cents a job on the fast machine and a penny per job on the slower machines. The rational cost-effective, choice is to use the slower machines.

Now suppose I need a minimum job completion rate. How many inexpensive cores do I need to get to realize the same job completion rate as say, 10, expensive cores? Again, the answer is simple. I need twice as many, 20. In general, if the slower cores are S% as fast as faster cores, we need 100/S slower cores. In the example above that is 100/50=2 times as many slower cores.

---

[1] The term ‘‘tightly coupled’’ refers to the tendency of these applications to require frequent communications between the constituent parallel components.

[2] http://en.wikipedia.org/wiki/High_Throughput_Computing