High performance computing

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 26

Energy considerations for limits to computing speed

To store a bit, you need to change the energy state. Physics dictates that such energy change should be larger than the typical energy within the ambient. Since cosmic background radiation has a temperature of 2.725 K, we conclude that

$$\Delta E = k_B T = 3.76 \times 10^{-23}\ \text{J} = 0.235\ \text{meV}$$

Computing is energy shifting

$$P_{consumed} = f N_f \Delta E$$

f computer clock frequency
N_f number of bit flips per clock tick

A computer with typical modern day specs, which runs at room temperature ($\approx$ 300 K) with the clock frequency 1 GHz and fully flips a 64 bit word per cycle, should consume 240 pW.
Based on this, we can expect about 12 orders of magnitude speed up for a computer consuming 200 W.
But, there is a limit at the end.

Practical limits due to heat dissipation requirement

Previous slide shows the fundamental limit but, in practice, the main problem is the cooling of computers. They generate too much heat and, thus, consume a lot of energy.
According to physics, radiation from the accelerating charge and, thus, the required computer power

$$P_{radiated} \sim f^4$$

So for the same architecture, the cooling system has to be adjusted accordingly.

- Increase of the clock frequency by the factor of 2 leads to the power consumption increase by the factor of 16.

For the last several years, the CPU clock frequency has not grown much beyond 4 GHz.

From single CPU to many

Because of energy and price considerations, it is impractical to boost the speed (i.e., the clock frequency) of a single CPU.
An alternative approach is needed. Multiple interconnected computing nodes: **Clusters**

Node is a computer which might have a single or multiple cores (CPUs)

Such nodes/cores can be connected either in pipe lines (serially) or in parallel.

The pipe line approach is very suitable for a GPU, since they do series of relatively simple operations over different data sets over and over. The high end video cards have thousands of pipelines.

The parallel approach is used for problems which can be run independently on different nodes (think about Monte Carlo simulations).

Conditions for parallel problem

Not all problems can be parallelized, for example a general ODE solver. The state at the time t depends on the state at the time $t - \Delta t$, so you cannot calculate it independently, i.e., without obtaining the previous time state. Recall a planetary motion problem, you cannot calculate a planet position independently without taking in account other bodies.
Such problem are called synchronous.
However, there are problems which can be done in parallel, i.e., asynchronous problems.
Think about matrix multiplication

$$\vec{C} = \mathbf{A}\vec{B} \rightarrow C_i = \sum_{j=1}^{N} A_{ij} B_j$$

There is the “vector” computer term, since the very first super computers were designed to do such matrix products in the parallel fashion.

Speed up vs. latency

Even in the best case, where we have a problem which can be equally split to N cores, the speed up does not scale with N.

$$S(N) = \frac{T_1}{T_1/N + \tau}$$

- S the speed up
- N the number of nodes
- T_1 the execution time for a problem on a single node
- τ latency or communication time, it usually grows with the number of nodes

There is no point to infinitely increase the number of nodes.

Matlab parallel tools

`parfor` vs. `for`

```
N=5e5;
for i=1:N
  c=inv(rand(10));
end
```

```
% matlabpool 2
N=5e5;
parfor i=1:N
  c=inv(rand(10));
end
% matlabpool close
```

Important: for every iteration, the evaluation of the `for` body should not depend on previous calculations in this loop.

See also `batch`