High performance computing

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 25

Notes

Limits to computing

To store a bit you need to change the energy state. Physics dictates that such energy change should be larger than typical energy in the ambient. Since cosmic background radiation has 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

$$P_{consumed} = \frac{N_b N_f \Delta E}{\Delta T}$$

N_b number of bits
N_f number of bit flips
ΔT time interval

Based on this modern computer (clock frequency 1 GHz) with 64 bit word at room temperature should consume 240 pW.
Based on this we still have something like 12 orders of magnitude to increase the speed of computing for a 100 W computer.
But there is a limit at the end. After all Sun has a power limit.

Notes

Practical limits

Previous slide shows fundamental limits. In practice the main problem is a cooling of the computers. They generate too much heat.
According to Physics, radiation from the accelerating charge and thus a computer power dissipation

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

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

- Increase of frequency by factor of 2 leads to power increase by factor of 16

Notes

From single CPU to many

It seems to be impractical to boost the speed of single CPU from energy and price consideration.
Alternative approach needed. Multiple interconnected computing nodes: **Clusters**

Node 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.

Pipe line approach is very suitable for GPU for high end video cards or in video decoders/encoders. Since they do relatively simple operations over different data set over and over.

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

Notes

Data streams/Instruction set chart

	Single instruction	Multiple instruction
Single Data stream	SISD	MISD
Multiple data stream	SIMD	MIMD

Notes

Conditions for parallel problem

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

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

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

Notes

How much gain vs latency

Speed up very quickly drops if the latency (time of communication) increases. After a certain limit there is no point to increase number of cores.

$$S_p = \frac{T_1}{T_1/p + \tau}$$

- S_p speed up
- T_1 execution time for a problem on a single core
- p number of cores
- τ latency, usually grows with number of cores

Notes

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 inner part of the body should not depend on previous results.

See also `batch`

Notes