Design of Parallel and High Performance Computing
HS 2013
Markus Püschel, Torsten Hoefler
Department of Computer Science
ETH Zurich

Homework 8
Out: 2013-29-11
Revision: 1

Performance Modeling

Little's Law

Imagine you want to board a train which leaves in 20 minutes. But before you have to buy the train ticket at a counter. You see that there are about 50 people in line before you. Serving a customer takes 40 seconds on average.

What property has to hold for this system to be stable? Will you miss your train?

Solution

A system is stable if the arrival rate is equal to the departure rate. Then the equality $\alpha\beta = N$ holds, where N is the number of things in the system, α is the amount of time between entering and leaving the system (latency), and β is the arrival rate.

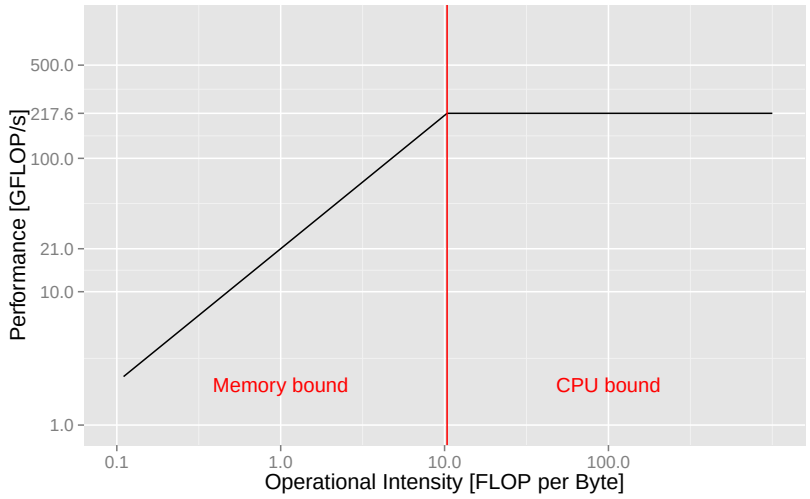
In our example $N = 50$, $\beta = \frac{1}{40s}$. Therefore $\alpha = 2000s \approx 33min$.

Roofline Model

The Intel Core i7 2600 CPU has a peak memory bandwidth of 21 GB/s. It is running at 3.4 GHz. It has 4 cores and can carry out up to 16 SP FLOPs/cycle. Draw a roofline plot for this processor. If a program and input combination land on the lower left of the plot, what does this tell you about the program?

Will all program executions yield points which lie either on the diagonal or on the "roof" of the roofline plot?

Solution



If a measurement result lies on the left side of the diagram and on the roofline (drawn in black) it is bound by memory bandwidth. If it lies on the right side of the roofline it is bound by computational bandwidth. If it lies below the roofline, it is bound by something else i.e., integer performance, which is not represented in the (simple) roofline model.

Balance Principles

Matrix Multiplication

Show that the operational intensity of a tiled matrix multiplication is in the order of $\sqrt{m}$, where m is the cache-size of the processor.

If we assume the matrix multiplication requires exactly n^3 flops and $12 * n^2$ memory operations of byte granularity, is matrix multiplication memory bound or CPU bound on a the machine described above? Does this change (if yes, when) if we assume:

- computational bandwidth of processors doubles every 18 months
- memory bandwidth doubles every three years
- cache size doubles every three years

Solution

Let's say we want to compute $AB = C$ where A, B and C are matrices of size $N \times N$. We decompose the matrices into blocks of size m. Now we can compute a block of size $\sqrt{m} \times \sqrt{m}$ by multiplying a sub-matrix of A of size $N \times \sqrt{m}$ with a sub-matrix of B of size $\sqrt{m} \times N$. This requires $W = \Theta\left(\sqrt{m} \times \sqrt{m} \times N\right)$ arithmetic operations. But only $Q = \Theta\left(\sqrt{m} \times N\right)$ memory operations. Therefore

$$\frac{W}{Q} = \frac{\Theta\left(\sqrt{m} \times \sqrt{m} \times N\right)}{\Theta\left(\sqrt{m} \times N\right)} = \Theta\left(\sqrt{m}\right)$$

A computation is balanced (computational bandwidth π and memory bandwidth β are utilized optimally) if $\frac{W}{Q} = \frac{\pi}{\beta}$.

If we assume a matrix size of $N = 10000$, we need 0.0571s for the memory operations and 4.5956s for computations. However, after 228 months the computation would take the same amount of time as the computation (with a matrix size of $N = 10000$). If we increase the matrix size with the same speed as the memory size increases, this happens after 456 months.

Now we can only re-balance by increasing the cache size. However, for matrix multiplication, if the computational bandwidth is increased by a factor of α the cache size has the be increased by α^2.

Stencil Computation

For the following code executed on a single core

```
for (i=0..n)
    for (j=0..n)
        a[i,j] = (a[i+1,j]+a[i-1,j]+a[i,j+1]+a[i,j-1]+a[i,j]) / 5
```

if we increase the floating-point performance by a factor of 2, how much does the cache size m have to be increased to redolence?

How does this change if we assume many iterations of the above code are carried out, parallelized across multiple cores?

For the sequential version of this code, we get

$$\frac{W}{Q} = \frac{\Theta\left(n^2\right)}{\Theta\left(n^2\right)}$$

The balancedness of stencil computation does not depend on the cache size.

If we perform the stencil computation in parallel on a machine where each processing element has a cache of size m and we perform many iterations of the above code, we have to load a $\Theta\left(\sqrt{m} \times \sqrt{m}\right)$ sized block into the cache once. This will be amortized by the following computations. For each iteration we perform $\Theta\left(\sqrt{m} \times \sqrt{m}\right)$ computations and exchange boundary regions of size $\Theta\left(\sqrt{m}\right)$ which translates to memory operations. Thus, we get the same result as for grid computations: $\frac{W}{Q} = \Theta\left(\sqrt{m}\right)$.

Balance Principles and Multicore

Imagine a processor X_p as a collection of processing elements, connected by a shared bus with bandwidth β. The main memory is also connected to the bus. Each processing element has a local memory (cache) of size m.

We used the processor X_1 to perform matrix multiplication, and m was tuned in such a way that the computation is balanced. Now we increase the number of processing elements, so instead of X_1 we use a parallel version, X_{16}. How should we increase m, so that the computation remains balanced (if β remains unchanged).

Solution

The memory bandwidth β is the same for all X_p. The computational bandwidth of a processor X_p is $p\pi$ where π is the computational bandwidth of X_1. As we saw earlier, for a matrix multiplication to remain balanced after π has been increased by a factor of p, the following must hold: $m_{new} \geq p^2 m_{old}$. Since the total amount of cache in X_p is only p times larger than in in X_1 we need to increase m by a factor of p to get to p^2.