Parallel Computer Architecture
Hemangee K. Kapoor
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Week - 01
Lecture - 04

Lec 4: Performance and Benchmarking

Hello everyone, we are doing module 1, Introduction to Parallel Architectures. This is lecture number 4, whose title is Performance and Benchmarking. Okay, so what is performance? It is used to compare systems as the popular understanding is. So, how do I perform defines how good or bad a system is or a machine is or a person is. Right. So, performance depends on how fast you are able to do an operation or how efficiently you are able to perform a task in real life. In terms of a computer or a machine, to define performance we can use two matrix M1 and M2.

The first matrix M1 is execution time. Okay. So, the first matrix M1 is execution time and the second matrix M2 is the throughput. Execution time is the time taken to complete one complete execution from beginning to the end. How much time using the wall clock for example, how much time you take to complete the program is the execution time. The other matrix throughput essentially says the number of processes that get completed per unit time.

So, it does not care how long one process takes, but per unit time how much work is getting output that is more important. So, if you recollect in a pipeline processor we had a good throughput because potentially we were able to complete one instruction every clock tick. Okay. So, the throughput would be good, but however the execution time or the latency per instruction could be little higher. Okay. So, with these two matrix M1 and M2 it is very intuitive that if I have a shorter execution time I would also have better throughput. Right. So, M1 implies M2, but does this always hold, is it true in all situations? Okay

So, we will take an example. Suppose we want to compare against two enhancements. So, as an engineer you were advised to improve the system and one engineer suggested that I will do an upgrade by doubling the frequency of my processor and the second suggestion is that instead of doubling the frequency we will add one more processor to the system. Okay. So, U1 is saying the frequency 2 which is the new frequency is twice that of F1 that is the old frequency and the second one is essentially F1 plus F2 you have two processors both running on the same frequency. So, if I have a program say I have a

program P and I want to run it on a system which implements U1.

So, how much time will it take? Okay. And in the same program I run on U2 how much time will it take? Because on U1, so I am denoting the machine and the upgrade with the same syllable. So, this U1 represents also the machine. So, here the frequency is two times. Right. So, the frequency on machine U1 is two times hence the time here will be half of the original. So, if I say t was the original time it will be now half of the original time because of the double frequency.

What happens in case of U2? Our frequency is same we have a single threaded sequential serial program and hence it is not able to take advantage of those two cores. Hence here the time remains equal to t original. So, the time, execution time of that program on U2 does not change at all. So, with the upgrade 1 the doubling of the frequency improves the execution time with respect to suppose I have these two matrix execution time and throughput. So, here U1 and U2 both these upgrades are not giving the same outcome in case of M1 because the execution time on U1 is better than the execution time on U2.

Now what happens for the second matrix of throughput? Okay. So, second matrix is throughput which says how much work we complete per unit time. So, we have multiple tasks. So, we have these many tasks running on both U1 and U2. So, if I send these tasks U1 to this I have 6 tasks here and they will take originally they were taking 6 t time units on U1 they will take 3 t time units to finish. Whereas on U2 it will take each of them will take a longer time, but your throughput will be more.

So, if I compute the throughput, throughput is number of tasks completed per unit time. In this example, for U2 I will be able to schedule 3 tasks on the first processor and 3 tasks on the second processor. Okay. So, essentially we are being able to execute it with a faster speed. So, the execution time does not change, but the throughput is equal in both the systems. The first one is able to do it faster, 2 time speed and the second one is able to do 2 tasks in parallel. Alright.

So, U1 and U2 are similar with respect to the throughput matrix. Okay. In a single threaded program I would say U1 is the better upgrade because I want a better frequency and per application execution time is good. However, with multi-cores we also are interested in the throughput and hence the upgrade number 2 is also equally important. How do I measure this execution time? Right. So, I will use this T_exe to represent the execution time. So, computing the time of execution, what are the components? Your program has got instructions and the time taken to complete all these instructions is the execution time.

So, the formula will have the instruction count, how many instructions you have, how much time every instruction takes. Okay. So, let us put down these two numbers here. I will say it is IC instruction count in the program which is a number. Then I will multiply this with the time required to execute every instruction and the popular term for this is called clocks per instruction. So, CPI is or clocks or cycles required, cycles per instruction.

So, CPI cycles per instruction. Now this is cycles divided by instructions. The first term is cycles. So, we have instruction count, cycles required to do every instruction and then TC, TC is the time taken by one clock cycle. So, I hope it is clear.

The execution time is the total instruction count multiplied by the cycles per instruction and then the time taken to execute one cycle. Okay. From here, from this same expression, I can now derive the formula for CPI. So, if I use the same one, what is CPI? So, CPI can be easily derived arithmetically using the expression on the left hand side. Okay. So, execution time as we saw instruction count, cycles per instruction and clock cycle time. Now CPI in this how much cycle does one instruction take is not so easy to obtain because in our old scalar processors it was more derivable or we can predict what was the CPI.

But in today's multi-core or out of order execution based processors, it is not so straightforward to compute CPI. Why? Because when one instruction is running, along with it other instructions are also running and they tend to interfere with each other. If one has a cache miss then that is rescheduled, something else runs in its own place and ILP is happening. Many parallelisms and many out of order executions keep taking place and hence the instructions rather interfere with each other and there is no isolated path of saying that this is how one instruction is executed. Hence, I would not be able to know what is the exact CPI, you looking at the hardware I cannot find this out.

But I can use the previous formula to find this out. I run a program, I know the number of instructions, we know T_c which know that is we know the clock period. So, with these three informations we can easily derive the CPI. Okay. So what is CPI? CPI is essentially the execution time

T_{exe}/IC * Tc

the time taken to execute one clock. So that is the cycles per instruction.

Another interesting metric is IPC which is instructions per cycle and as you rightly understand this is the reciprocal of the CPI. So this is 1 over CPI. So cycles per instruction have to be as small as possible because I need lesser cycles to do more instruction for having a better performing system and IPC that is instructions executed per clock cycle essentially points to throughput and we want this value to be higher. So CPI should be low and IPC should be high. Fine. So we have defined performance in terms of execution time.

Now we will look at the concept of benchmarking. So If I run a program on a machine, I need to decide the goodness of my program that is the algorithmic goodness and the goodness of my machine that is how efficiently it is able to execute my program. So deciding the goodness of a machine is the main theme of benchmarking. For a single threaded program, I would say it is simply the execution time because we are not concerned about throughput, we do not have anything else to run. We have a single program and the faster it runs the better my machine is.

Okay. We are not looking at the algorithmic goodness in this subject, we are looking at the architectural goodness. So my machine is good if my single threaded program gives me the best execution time. But I have another program which does not run so nicely on my machine. So how do I decide whether my machine is good or not for a single threaded program? And then how do I compare two machines? To compare two machines, we would take one program, the same program and run it on the two machines separately. Now running a program on the machine is not the exe but we have to take the source code, compile it and then run it on the respective machines because the machines could have different hardware, they could have different compilers on it.

So taking the same program and running on two different machines may or may not give similar results because a particular compiler could be optimized for a particular type of a program and that same compiler might not optimize other types of programs. Okay. So a machine is optimized for a particular type giving good results whereas it can give terrible results for another type of program. And so with this, how do I decide that my machine, sorry machine M1 is better than M2, because it could be unfortunate that I have given the bad programs to M1 which it was not able to handle and M2 got better programs which it was able to handle efficiently. So it won't be a fair comparison. So for a fair comparison, we should take a broad set of programs which can exercise all compiler features or hardware features of the machine and overall give me the complete picture.

So we need a set of programs that represent different behaviors of an application and this selected programs is then used for a rigorous comparison and this set is called the

benchmark suite. There are very popular benchmark suites available within the research community. So this slide shows the examples of all of them. The first one on the left is the SPEC benchmark. So it is provided by the Standard Performance Evaluation Corporation, mainly targets personal computers, workstations and microprocessors.

It comes in the two categories for integer workloads, SPECINT and for floating point workloads, SPECFP. This benchmark has been around for quite some time and the initial version was SPEC 1989. So SPEC89, then came SPEC92 because with every generation of the processor, the capabilities changed, the compilers evolved and the type of applications also kept on getting added. So for all these enhancements, the benchmarks had to keep pace with the advancements in the software and the hardware and therefore the SPEC benchmarks also had variety of versions. From 89, we had SPEC92, SPEC2000, SPEC2006 and the most recent one is SPEC2017.

So these are the SPEC benchmarks. The next set of benchmarks on the left hand side are PARSEC and SPLASH. So PARSEC are multi-threaded workloads and SPLASH, they are rather old but still useful set of benchmarks. On the right hand side, we have TPC benchmarks mainly for transaction processing. We have three types of benchmarks provided by them. One is for online transaction processing servers, that is OLTP servers.

The second is for decision support systems, that is they manage or check the efficiency of databases in businesses and administration, TPC-D and TPC-H and for web servers, we have TPC-W. So these are the three different categories of the TPC benchmarks. So that was for workstations and bigger systems. So what about small systems like embedded devices, your mobile phones or your embedded machines, washing machines, micro-wares, any embedded system you design, even that needs to be benchmarked. So for that, we have EEMBC and MiBench.

So MiBench, these are the two benchmark suites available for embedded systems and for multimedia applications, we have the media bench. So MiBench and MediaBench are two other benchmark suites for applications related to embedded and multimedia. Thank you.