EEE4120F 2021
Quiz1 – View of Parallel Computing Landscape

You need to complete this quiz by 23h55 Monday 29 Mar. This is a first quiz for marks. This quiz is to be done as an individual assignment, not as teamwork. You are required to abide by the honour pledge of treating this as an individual assignment that is entirely your own work.

(Note: in Vula some questions had their answer letters randomized, so keep this in mind if comparing your answers).

Answers indicated in red with ← to emphasise answer if viewed in monochrome.

Question 1

The paper discussed predictions regarding clock rates, indicating that if trends towards 2003 were maintained, clock rates should have exceeded 10GHz in 2008. We're now in 2021 and clock rates are not really going beyond 5GHz, as per the latest Intel i9 architecture. Yet the prediction had expected 15GHz in 2010 already. What is the main problem highlighted in the paper causing this problem?

a) It is largely the problem of parallelization, going from single or few cores to many cores which has an impact of exponential growth in power consumption.
b) It is partly the problem of high clock rates leading to high power use and challenges of heat dissipation, collectively referred to as the power wall. ←
c) It is the consequence of the high complication of new chip architectures, which cause lengthy propagation delays. Some call it, ironically, the little *c* problem, i.e. speed-of-light problem (I say ironically here because it is actually a big problem).
d) It is all due to the reducing fabrication size, sub 10nm connections and smaller transistors. As these get smaller, they block more power and thus you need to push in more power to make the whole caboodle work.

Question 2

The authors talk about inspiring 'parallel revolutionaries' … something your lecture, and many other academics have been infatuated by. They talk about a major goal of the 'parallel revolution' is to make parallel coding as easy as sequential coding. But that is a very broad statement. They go on to identifying a set of characteristics of programming that should be of equivalent ease between parallel programming and sequential programming; select the one option below that correctly specifies which of these aspects they emphasise:

a) Easy of writing parallel programs that are as adaptable, portable, fast, and concise as sequential programming.
b) Easy of writing parallel programs that are as dependable, utilize the same debugging methods, and as consistent as writing sequential programs.
c) Easy of writing parallel programs that are as reusable, cost-effective, and dependable as writing sequential programs.
d) Easy of writing parallel programs that are as efficient, portable, and correct as it has been to write sequential programs. ←

Question 3

The authors indicate a “technology sweet spot” around a pipelined processor of five-to-eight stages that is most efficient. But a “sweet spot” (or, more formally, a design optimization) that give what sort of optimization goals? Select the option below that best explains this:

a) Optimal in terms of ease of programmability as well as compiling.
b) Optimal in terms of performance per joule and silicon area. ←
c) Optimal in terms of minimal propagation latencies (beyond five, and certainly beyond eight pipeline stages, will need a slower clock than fewer stages).
d) Optimal in terms of placement (i.e., placing cores on silicon), core integration (i.e., linking cores), and speed (clock rates beyond that of larger pipelines).

Question 4

The paper mentions processor chip potentially comprising both thin and fat cores. For example, having a fat core that might take 10x the resources of a thin core, but have more sophisticated instructions. One of your colleagues who has also glanced at this paper is convinced that more cores are going to be better for any application, regardless of whether they are fat or thin. Select the option below that provides a correct response to your colleague’s viewpoint relation what this paper indicates on this issue:

a) It depends much on the processing involved. For some tasks, it may be better to just have thin cores all doing just the same basic instructions. But other tasks may be done more optimally by having a few fat cores doing complex instructions quickly, and lots of thin cores running just basic instructions. Since a computer often does more than one task, a combination of fat and thin cores is likely to work out better in the long run. ←
b) It depends on the number of instructions you want to run, i.e. the ultimate deciding factor for the performance of any program. If running many instructions then, yes, your colleague is right: more cores are better than fewer, even if their instructions are very basic. Similarly, if there are only a few instructions to run but they repeated often; same thing: more cores will be better regardless of how fast each individual core may be.
c) Both a and b are right.
d) Neither a nor b are right.

Turn over for Question 5 …

Question 5

The paper introduces the concept of a "Dwarf", as some sort of computational pattern.

Consider that you have suggested to a colleague to think in terms of Dwarfs for a new project you are both involved in, which makes heavy use of graph algorithms (e.g. a machine learning application you're building).

But your colleague hasn't heard or dwarfs or parallel patterns.

For your answer to this question write up a briefly explain of the concept of a Dwarf, using your own words, which you might use to explain this concept to a colleague.

Answer: [marking advice in square brackets]

Dear colleague,

To explain a Dwarf I need to quickly recap the concept of a structural pattern and a computational pattern. [+1 identifying where things are leading]

A structural pattern, or more formally a ‘structural design pattern’, describes how computation components are put together to realize a software solution or system. A computational pattern describes the operation, or processing, done by one of these computation components within the structural pattern.
[+2 structural aspect, +2 processing component; at least saying that to be useful dwarfs need to be working with other parts, a Dwarf on its own isn’t useful.]

A factory analogy helps reinforce this distinction, in which structural patterns describes the physical parts of the factory’s structure and workflow between these, to deliverer widgets (or end-products). The computational patterns describe the workings of these intermediate physical parts or machines that work on resource flowing through the system, culminating in the end-products. [+1 reinforcing the distinction between structure and processing, or ensuring it is clear that Dwarfs are just part of the solution.]

The notion of Dwarfs came out of a research study to identified widely used computational patterns that were a crucial part of particularly successful high-performance computing software applications. A Dwarf, and in this case one of the initial 7 Dwarfs, refers specifically to one of these computational patterns, as apposed to computational patterns generally.
[+2 somehow emphasising the fact that a dwarf is a particularly useful computational pattern that appears in multiple (successful) applications, and was found through a process of investigating / mining real applications]
[+2 overall writing quality, good structure and language to your prose.]

(Oh, dear colleague, I’m glad to see you are still alive. I thought I should just poke you with my pen to make sure, as your eyes had glazed over I thought you had gone offline after my speech; but hopefully you now understand these important terms.)

[-1, and no mark, maybe even a negative mark, if you try to be silly or too funny]

End of Solutions