Parallelism and Amdahl’s Law for Higher

Overview

This exercise aims to teach children a quick introduction to Amdahl’s law, a law to calculute speedup from parallelizing a program.The formula can be given as 1/1-f, where f is the fraction of the program that has been sped up. For this exercise to simplify comprehension, the formula has been given in terms of the time reduction rather than actual speedup. In practice, Amdahl’s law provides an estimate of the overall speed at which the algorithm can be executed.

Suitable For

This exercise is the version that is suitable for students studying Higher Computing, or Advanced Higher students. Mathematics levels should be at at least National 4/5.

Key Concepts

Amdahl’s law, parallelisation, speedup, multi-core architecture

Learning Intentions

* Remember Amdahl’s Law.
* Calculate speedup in a series of problems given using Amdahl’s law.
* Parallelize parts of a program to help speed up the execution.

Success Criteria

* I can understand how parallelisation is an important factor in speeding up a program.
* I can work with Amdahl’s law to predict the time speedup after a program has been parallelised.
* I can explain the need for multiple cores when computing.

Time Required

1 period - 1 hour

Preparation

Print out the items given in the pack, one of each.

Prior Learning Assumed

Perhaps the idea of multiple cores and parallelism might already be known to them - this could solidify the idea.

Outline of Activity

1. Explain to the class that we are going to be looking at a new topic for computing, called parallelism.
2. Outline that for example imagine you want to want to dig a very big hole. One man might take 60 minutes to dig this big hole, and that this is like giving one core all the work to do. It’s going to be really slow.
3. Ask the class for any feedback on what they might do instead? Some options might be:
   1. Get more people involved to share out the task
   2. Dig a smaller hole - maybe, but we need this hole to be this size!
4. Say that the concept of getting more people involved to share out the task is correct - this idea is called parallelism!
5. Show the printed out board and explain that this is what a model computer might look like inside, with its multiple cores - which are essentially like little computer brains. Explain that this is an example of a four core computer.
6. Explain that essentially they can turn a program from a serial one (all in one core) to a program that has a parallelisable section and a non-parallelisable section.
7. Introduce the program (long strip of pink paper) and lay it out along the top core, in one line. This is a serial program and is running in one core. Show that how along the bottom of the board, where the time is, if you are to put the program all in one core, it takes a very long time to do it.
8. Cut up the program into two sections and show that one section is the parallelisable section of the program. This section can be shared out among the other processors.
9. Cut the parallaelisable section into 4 pieces and place one piece on every core. Note that the time at the bottom has been reduced.
10. Finish by saying that this is the main idea of parallelism.

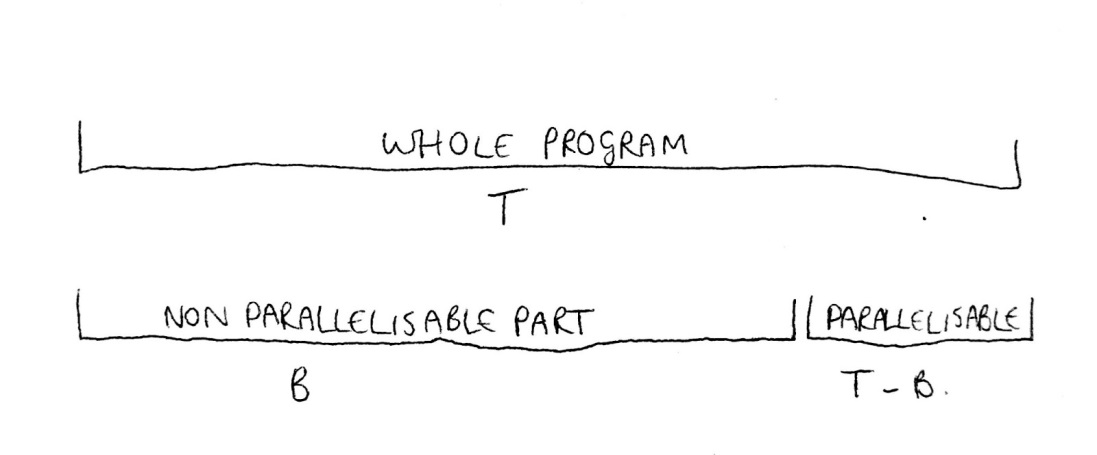
Extra

Outline that there is a function that can help figure out how quick a program could be if it was parallelised. This is called Amdahl’s law.

You can move to the board to demonstrate the following points:

* The total time of the whole program to run is T.
* The total time of the non-parallelisable part is B.
* Thus, the total time of the parallelisable part is T-B.
* If there are N cores running the parallelisable part, this means that the fastest the parallelisable part can be run at is (T-B)/N.
* This means, the fastest total time of the whole program running on N cores is:
  + T(N) = B + (T-B)/N
  + (Adding in the non-parallelisable section)
* This is Amdahl’s law.

This diagram should be drawn out to further explain.



Give students the exercises from Handout.docx to further explain Amdahl’s law.

Answers:

1. 7 seconds

2. 7 seconds

3. 0.76 seconds

4. 1.46 seconds

5. 0.3 seconds