Evan Lobeto
elobeto
CS 4234 HW1

1. When n%p = 0:

   For process 0, my_first_i = 0, my_last_i = (n/p) - 1

   For process 1, my_first_i = n/p, my_last_i = n/p + n/p - 1

   For process k, my_first_i = (k-1)(n/p), my_last_i = k(n/p) - 1

   When n%p != 0:

   For process 0, my_first_i = 0, my_last_i = (n/p) + (n%p) – 1

   For process 1, my_first_i = (n/p) + (n%p) – 1, my_last_i = 2(n/p) + (n%p) – 1

   For process k, my_first_i = (k-1)(n/p) + (n%p) – 1, my_last_i = k(n/p) + (n%p) - 1

2. A) p - 1
   B) $\lceil log_2(p+1) \rceil$

3. An example of a MISD computing system would be the Space Shuttle flight control.

4. A) $(10^{12}/1000)$ instructions/processor * $((10^6)$instructions/second) / $(10^{12}$instructions) + $(10^9(1000-1))(10^{-9}) = 1999$ seconds
   B) $(10^{12}/1000)(10^6/10^{12}) + (10^9(1000-1))(10^{-3}) = 9.99 \times 10^8$ seconds

5. A) Seymour Cray was an electrical engineer who pioneered supercomputing. He was the first to realize that there was more to performance than just the speed that a processor could run and set out to maximize the performance of systems of computers. Cray created the CDC 6600 which was the first commercial supercomputer and he maximized the I/O bandwidth so that the processor always had data to work on. He is considered the father of supercomputing and helped develop many of the methodologies that go into constructing large supercomputers and how they can be optimized. Without his work, we probably would not have as advanced supercomputing systems as we do today.
   B) A Beowulf cluster is a computer cluster consisting of several commercial grade systems networked into a local area network and are able to share computing power between them. This is an example of a distributed-memory system and allows many people to have the power of multiple computers under one system. It is important to parallel computing because it is a simple task to network commercial computers together and use a library such as OpenMPI to take advantage of the additional power so that many companies or individuals can have access to parallel computing.
   C) A fat tree network is a network organized in a tree-like fashion where the network links grow "thicker" as the root is neared. This allows the network to be specifically tailored to the application for which it will be used. This is important because it allows certain distributed-memory systems to be further optimized for their tasks. If a system is going to be used for the same task over a long period of time, it is very helpful to have the communication between nodes optimized so that total run time can be decreased and the task can be completed faster.
   D) The OpenACC API is a library for C, C++, and Fortran that allows programmers to specify

loops and sets of code to be offloaded to an accelerator. The accelerator can range from another CPU, APU, or GPU. The main goal of this API is to simplify parallel programming of CPU and GPU systems. This allows programmers to take advantage of additional computational power in their systems to leverage the performance gains of parallel computing. It is important because it is another tool that programmers can use to exploit parallelism without having to have access to large supercomputers and can use the power in their own machine.
E) Amazon AWS for HPC allows users to have remote access to a large supercomputing network in the cloud. This is powerful because it allows tons of companies and researchers to leverage the power of large supercomputing systems without having to spend the money to develop one and create the infrastructure. It saves them money and allows them to focus on developing their product or service much faster than they could otherwise.
F) A warp is a group of 32 threads. The warp allows programmers to divide tasks so that a block does not contain inactive threads. It also allows programmers to organize tasks so all of the code inside a warp follows the same code path so the code will be further optimized.

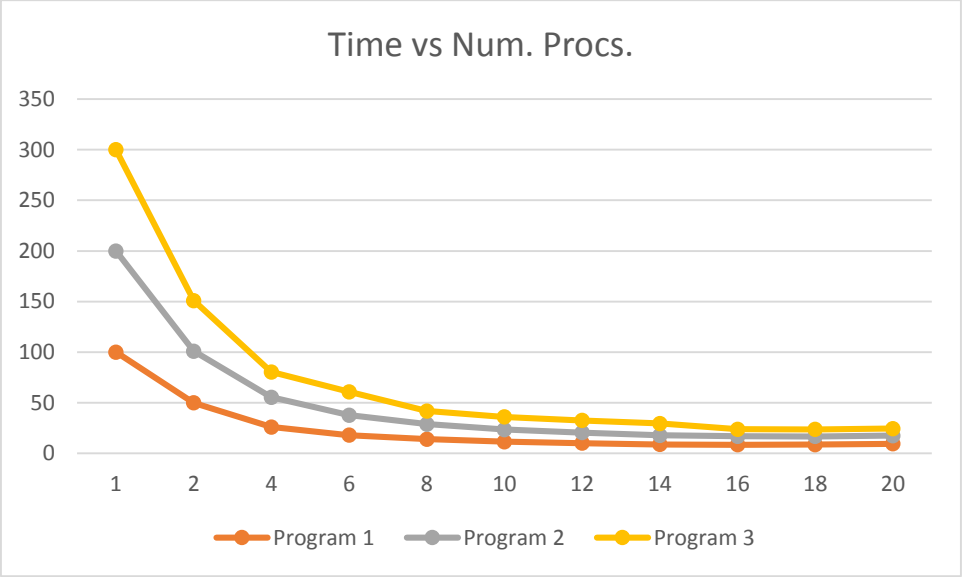
6. Top500: The top 500 list ranks supercomputers based on their performance on the LINPACK Benchmark. In this benchmark, supercomputers must solve a dense system of linear equations. It is measured in terms of calculations per second and uses the Rmax value from the benchmark. Currently Tianhe-2 is ranked 1, a supercomputer developed in China with a benchmark score of 33.86 Pflops/s.
   Green500: the Green 500 list ranks supercomputers based on their energy-efficiency. It is measured using performance-per-watt. The rankings are derived by taking the LINPACK score from the top 500 list and factoring in the total energy that it uses in computing and cooling. Currently TSUBAME-KFC-GSIC Center is ranked number 1, a supercomputer from Tokyo with a score of 4,389.82 Mflops/W.
   Graph500: The Graph 500 ranks supercomputers based on data intensive loads and the benchmark used stresses the network subsystem that the system uses to communicate. The results are ranked in terms of traversed edges per second resulting from the breadth first search that the benchmark performs. Currently K Computer from RIKEN Advanced Institute for Computational Science is ranked 1 with a score of 17,977.1 GTEP/s.

7. Moore's Law: Moore's Law states that the number of transistors in a dense integrated circuit will double approximately every two years. This relates to parallel computing because in the past, more transistors were added in order to increase clock speed, but now the focus is on adding more transistors to create more parallel machines with more cores.
   Amdahl's Law: Amdahl's law relates to parallel computation because it states that there is a general limit to speedup that can be obtained from parallelizing a serial program.
   Gustafson's Law: Gustafson's Law relates to parallel computing because it provides a counterpoint to Amdahl's Law saying that computations involving arbitrarily large data sets can be efficiently parallelized. It addresses the shortcomings of Amdahl's Law which doesn't take into account the computing power that becomes available in parallel machines.

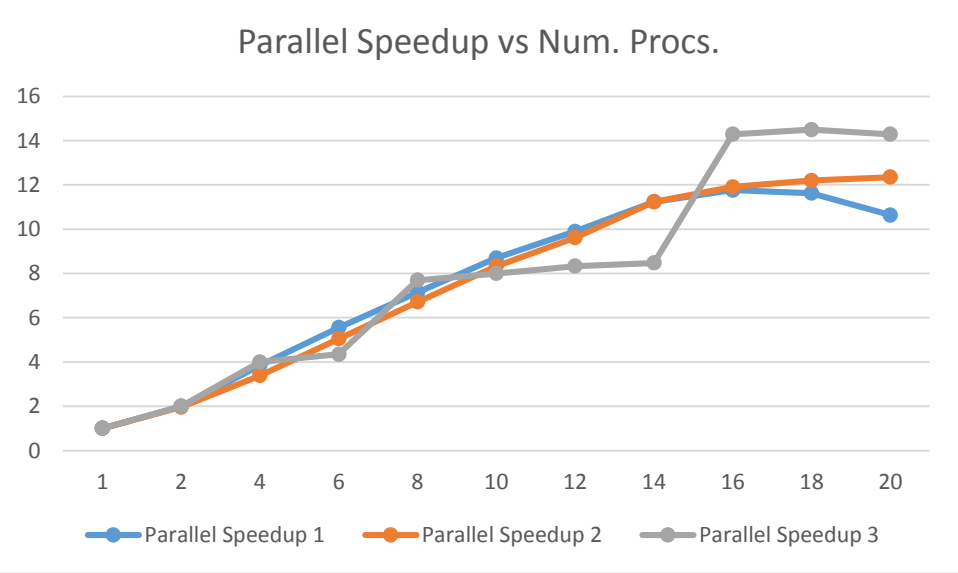
Evan Lobeto
elobeto
CS 4234 HW1

8. O(n): 100x
   $O(n^2)$: 10x
   $O(n^3)$: 5x

9.

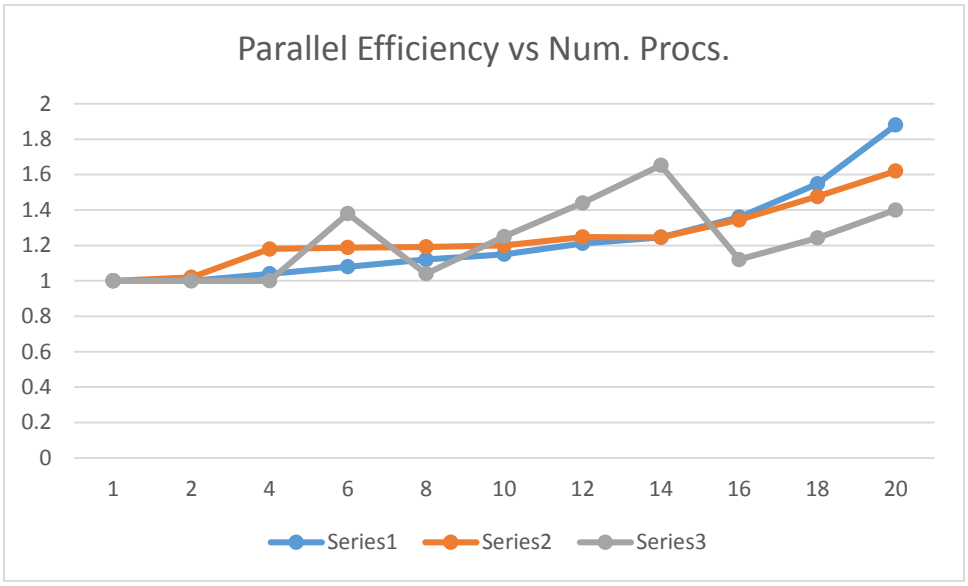


This chart is useful in showing the decrease in time that can be obtained from parallelizing the serial program. It clearly shows that as the number of processors increases, the time needed to run decreases.



This chart shows the exact speedup that is obtained when the serial program is parallelized. It can be seen that as the number of processors increases, the speedup increases as well. Program 3, however, experiences a huge jump in speedup when the

number of processors goes from 14 to 16, which may imply that the program was optimized for that many processors.



This chart isn't terribly useful, it just shows how the parallel efficiency fluctuates as the number of processors increases. Generally, most of the programs trend towards becoming more efficient as the number of processors increases. Program 3 experiences several drops in efficiencies as the number of processors increase from 6 to 8 and then again from 14 to 16. This mirrors the jump in speedup that it experienced in the graph above over the same number of processors.