Parallel Computing/Programming Assignment #1: Amdahl's Law for Multicore

Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180-3590
Email: chrisc@cs.rpi.edu

January 18, 2024

DUE DATE: 11:59 p.m., Friday, January 26th, 2024

1 Assignment Description

For this assignment, you are to write a serial C program which uses the revised formulations of Amdahl's Law from Hill et al [1] to determine the speedup for a supercomputer consisting of multicore nodes connected by a zero overhead network. Recall from the paper that:

$$Speedup_{base} = 1/((1-f) + f/n) \tag{1}$$

where $Speedup_{base}$ is the number of times faster the parallel execution is compared to the execution time on a single processor (i.e., sequential) execution, f is the fraction of the total execution time in the serial case that can be made parallel and n is the number of processors.

In [1], we found that a heterogeneous multicore chip where one or more cores are more powerful than others yields the following speedup formula:

$$Speedup_{asymmetric}(f, c, r) = 1/((1-f)/perf(r) + f/(perf(r) + c - r)) \tag{2}$$

Here, r cores are combined into a super-fast core yielding a serial performance of $perf(r) = \sqrt{r}$ but we see that we have to penalize the parallel fraction of the program by not allowing the full c cores to be used within that node but instead only $c + perf(r) - r$ processors can be applied.

Now, the ultimate "cake and eat it too" design allows the dynamic combining of upto r cores when necessary for fast serial execution of the non-parallel part of the program, but can reconfigure those r cores at zero overhead to be fully applied to the parallel part of the program. This yields the following dynamic formulation of speedup:

$$Speedup_{dynamic}(f, c, r) = 1/((1-f)/perf(r) + f/c) \tag{3}$$

We are adding a new wrinkle into this, by assuming that the above describes the speedup for a single node within a larger supercomputer system. Thus, there could be N nodes and so the new speedup equations become:

$$Speedup_{asymmetric}(N, f, c, r) = N * 1/((1 - f)/perf(r) + f/(perf(r) + c - r)) \quad (4)$$

and,

$$Speedup_{dynamic}(N, f, c, r) = N * 1/((1 - f)/perf(r) + f/c) \quad (5)$$

Now, suppose you are given a supercomputer with upto 1024 nodes and each nodes has upto 1024 cores. Write a serial C program that generates speedup data that can be plotted in 3-D according to the following:

1. Each 3-D plot will be based on a fixed f that will have the follow 5 possible values $f = \{0.99999999, 0.9999, 0.99, 0.9, 0.5\}$ for both $Speedup_{asymmetric}(N, f, c, r)$ and $Speedup_{dynamic}(N, f, c, r)$. Thus, there will be 10 plots in total generated.

2. Within each plot, N will vary from 1 to 1024 in jumps of 4x (i.e., 1, 4, 16, 64, 256 and 1024).

3. Within each plot, r will vary from 1 to 1024 in jumps of 4x.

4. Within each plot, c will vary from r to 1024 in jumps of 4x.

5. The axises for each plot are $N * c$ total cores in the supercomputer (x), r cores used to make the super-fast serial processors (y) and the overall computed speedup (z). *Note, $N * c$ does not produce a unique value, e.g. 512 node for 32 cores and 32 nodes with 512 cores have an equal number of cores - e.g., 16,384. To make your plots appear better, plot only the max speedup obtained each $N * c$ and r configuration.*

6. There is a template on `submitty.cs.rpi.edu`, `/tmp/assignment1-template.tar.gz` to help you get started. Please use this template.

Your C-code output `printf` statement will appear as follows:

```
printf("%10.8lf %8u %6u %6u %6u %12.4lf %12.4lf \n",
                    f[f_index], (node_index * core_index),
                    node_index, r_index, core_index,
                    speedup_asymmetric, speedup_dynamic);
```

The first variable is the fraction f followed by total number of cores, number of nodes, R cores, number of cores per node, the asymetric speedup and dynamic speedup.

In total you will have 1260 data points – 630 data points for asymmetric and 630 data points for dynamic. Each 3-D plot for a given f and speedup formula combination will have 126 data points (some of which are duplicate total core counts). So, again, you will have 10 data plots overall.

You can use GNU Plot, numpy or Excel to perform the plots. For each graph, indicate the overall trend you observe and compare both speedup formulas for each f value. Does dynamic always win ?? Are there places where dynamic and asymmetric yield nearly the same speedup? Document what those cases are and provide some reasoning.

2 HAND-IN INSTRUCTIONS

Put your assignment C code and write-up with graphs in PDF form under the `ASSIGNMENT-01` gradeable on `submitty.cs.rpi.edu`. Autograding will grade for output correctness. Example output will be available.

References

[1] M.D. Hill and M.R. Marty. Amdahl's law in the multicore era. *Computer*, 41(7):33 –38, july 2008.