Homework 3

Peter McGill | github :petermcgill94 | UW peterm28

Question 1

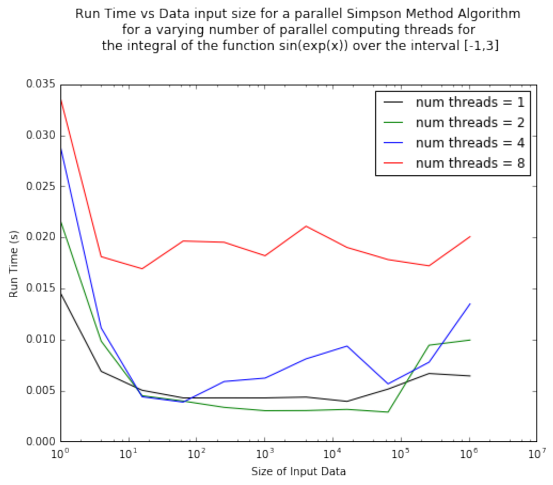

Figure 1: Run Time vs Data input size for a parallel Simpson Method Algorithm for a varying number of parallel computing threads for the integral of the function sin(exp(x)) over the interval [-1,3]

- For n = 8 and small chunk size we can see the run time is much higher than the other tests. This is down to two factors. Firstly sage math cloud is running with access to 4 cores, so having 8 threads doesn't increase the paralleling, it just adds more overhead time when creating and managing more threads. Also when chunk size is small there is a lot of overhead to manage and assign threads to work many times during the process, which again pushes up run time.
- When the chunk size is equal to the problem size one thread does all the work, and nothing is computed in parallel, hence the run-time is longer. We can see this reflected in the plot as run-times for 8. 4, and 2 threads increase as chuck size approaches 2^{20} (the problem size). However, with one thread it roughly stays the same as we would expect.
- Chuck size is relevant to the Simpson method because each computation in the the for loop requires 3 elements from the array, which means each thread needs all the elements in its cash line for fast computation - so the thread doesn't have to pull required element from ram, which is slow. This means the chunk size should ideally be some multiple of 3.
- Looking at figure 1, somewhere around 10^3 chunk size appears to be optimal for 2 threads. However, this could be unreliable as sage math cloud shares its cores amongst users and number of threads specified may not be exactly the number of threads actually used in the computation. In practice I think it is tricky to conjecture an optimal chunk size as function of problem size. This is because you have to balance the chunk size against over head for scheduling threads many times. Also because the Simpson method accesses 3 elements per computation you also have to make sure each thread has required elements in its cache line for high performance. The number of threads available will also be a contributing factor ideally you would want to avoid having threads being idle.