steady-state temperature distribution

james toy

2010-06-02

The purpose of this project was to analyze the performance of various 2D steady-state temperature distribution implementations. We accomplished this via the jacobi method and finite differences. The program was written sequentially, in parallel with row-wise domain decomposition, and finally with a checkerboard domain decomposition. Upon analysis we understand that with simpler and smaller simulations the sequential program has an advantage because it is capable of "knowing" everything about domain of the program. This also gives us a better ability for utilizing caching and locality of reference. As a simple example we note an array of size $100x100$ at $\varepsilon = 0.01$ the sequential returns a time of: 0.66 seconds. This is an order of magnitude faster than the same computation on four processors. Interestingly enough though when done on two processors we see a time of 0.295 which means that MPI is running the code on one machine that has two cores (which will take advantage of no inter-node communication). This data is present in the graphs and shows that there is a performance increase when running the program with MPI on one machine with multiple cores.

Also when comparing the parallel versions, it is very apparent that they checkerboard mesh is highly scalable while the row-wise decomposition is not. This is because when computing the isoefficiency function for the algorithm based on a row-wise domain decomposition it turns out to be:

$$n \geq Cp$$

with a scalability function of:

$$\frac{M(Cp)}{p} = C^2 p$$

this shows that the row-wise block decomposition is not very scalable. However, with the checkerboard domain decomposition each of the p processes are responsible for a smaller mesh that is $\frac{n}{\sqrt{p}} * \frac{n}{\sqrt{p}}$. Computing the isoefficiency function shows:

$$N^2 \geq C\sqrt{p}$$

with a scalability function of:

$$\frac{M(C\sqrt{p}}{p} = C^2$$

this is much more scalable and what we are looking for when we plan on computing values $n \geq 10,000$ with $\varepsilon = 0.01$.

Looking at the data in the charts, it doesn't appear that we are getting that much of a speedup until we get to the problem size being $n \geq 1000$. The most drastic difference is between the sequential (30.93) and the 16 processor checkerboard domain decomposition (17.1838). This is healthy speed-up! Also note that the row-wise was very close behind on 16 processors with 18.1624. It would appear that once we reach values of $n \geq 10000$ we will see significant speed-up; even in comparison to the row-wise parallel algorithm since the scalability function is so much better on the checkerboard decomposition.

NOTE: only problems 1,2,3 were completed for this project, I ran out of time to do 4. And yes, as per your last comments the column change i made to implement column-wise domain decomposition is basically row-wise decomposition.