Parallel Programming in Algorithmic Trading - A Proposal

Atheendra Tarun
ap778@cornell.edu

Harsh Pandey
hp349@cornell.edu

Santoshkalyan R
scr96@cornell.edu

Shashank Adimulam
sa793@cornell.edu

Prasannjit Kumar
pk435@cornell.edu

March 28, 2014

1 Problem Statement

Our project aims to study and understand the effects of parallel computing in real world applications. An ideal example of a computationally-intensive problem that would benefit substantially by utilizing multiple processors would be algorithmic trading. The problem, here, is to be able to process a huge number of stock prices within milliseconds, so that decisions related to buying or selling the stock can be taken in the least amount of time possible, to maximize profits. There are a multitude of algorithms for trading, but for demonstration, we will pick the Moving Average Convergence/Divergence (abbreviated as $MACD$) algorithm, which belongs to the overarching class of Moving Average trading techniques.

The MACD signal is generated by calculating the average of the previous n data points, which represents the "momentum" of the stock price. When the stock price crosses the MACD signal in either upward or downward direction, we can surmise that there is a disparity between the current price of the stock and that of the potential future price. When the stock signal crosses MACD upwards, in other words when the stock signal cuts the MACD and obtains a value higher than that of the MACD, it generates a sell signal. If the reverse happens, we generate a buy signal. While there are other variations of this algorithm, this method is sufficient to demonstrate the advantages of parallel execution.

1.1 Tasks involved

- Create a serial implementation of MACD for comparison.
- Create several variations of parallel implementations for identifying the best technique. This includes:
  - Allocate a complete stock to each processor. This eliminates advantages gained through caching. However, there is almost no necessity for communication between the processors.

  - Distribute work load based on time period. Each processor works on calculating the average of only a certain time frame of a single stock, rather than on a stock-by-stock basis. We predict that this performs slightly better in terms of memory coherency.
  - Explore multiple time periods for the MACD signal by calculating moving averages for different number of data points.
- Compare ideal speedup with actual speedup and perform tuning and optimizations.