Project 1A: SUMMA Algorithm

David Noble, Jack Chua

September 24, 2011

1 Overview

The objective of this checkpoint is to implement the SUMMA distributed matrix multiplication algorithm for non-square matrices and analyze the results.

2 Plots

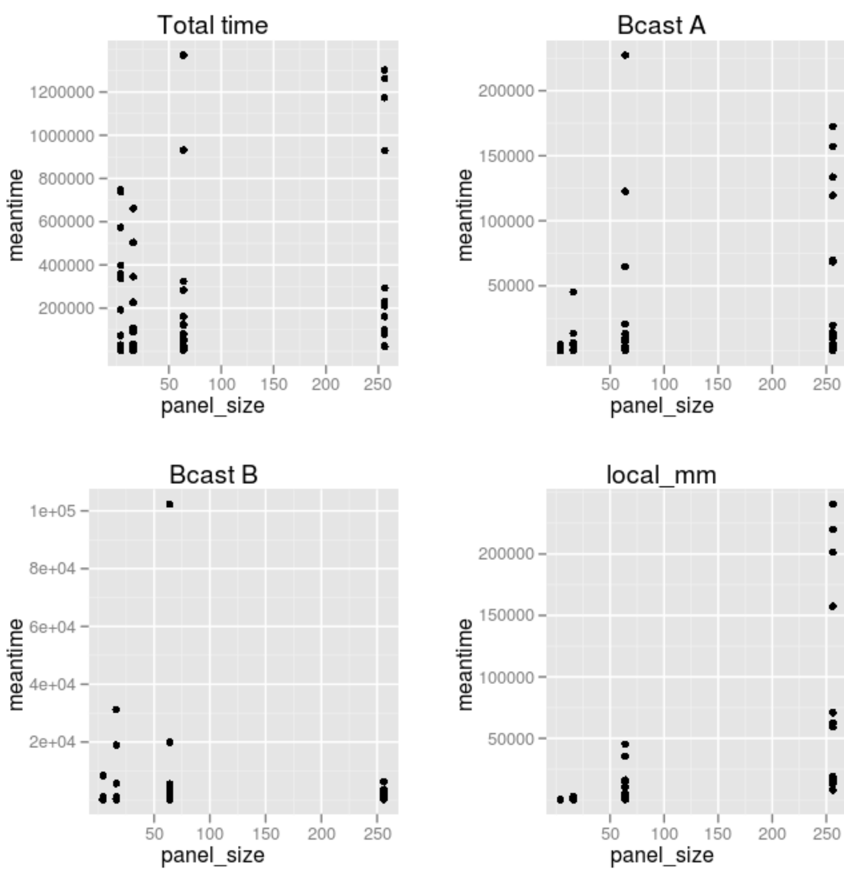

Figure 1: How total time, broadcast time, and local multiplication time varies as panel size varies

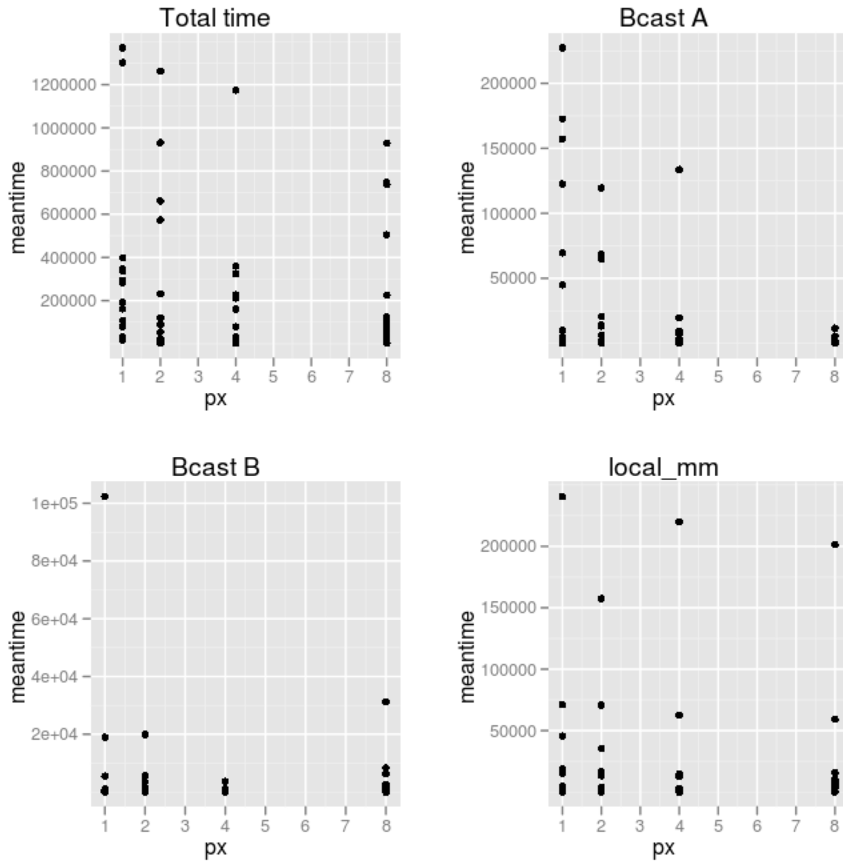

Figure 2: How total time, broadcast time, and local multiplication time varies according to grid layout

3 Analysis

Methodology. These plots were generated by taking the average times collected from each slice for every unique test case. Uniqueness is defined by the combination of variables m, n, k, px, py, and panel_size. We collected the total time elapsed, the time taken to broadcast Ablock and Block, and the time taken to compute the local matrix multiply. We then used R to plot the times against panel size and grid layout.

Analysis of panel size. We analyzed how changing the panel size might affect various potential bottlenecks in the algorithm.

- Looking at the **total time elapsed**, we noticed that increasing the panel size tended to **worsen** performance. This makes sense because in a broad sense, we shouldn't achieve the benefits of parallelism if the panel size is too big and we get closer and closer to a (by-nature) sequential outer product.
- The **time taken to broadcast Ablock** seemed to **benefit from smaller panel sizes**. The largest panel size provided the largest concentration of high time values, while the smallest panel size provided the largest concentration of small time values.
- The **time taken to broadcast Bblock** seemed to **benefit from the largest panel size, and the smallest panel size**. The middling panel sizes seemed to provide the worst performance.
- Finally, **time taken for local matrix multiplication** showed a clear increase as the panel size grew. This makes obvious sense as larger blocks call for larger overhead during sequential multiplication.

A feasible conclusion is that the smaller the panel size is, the better - this allows use to benefit from parallelism as much as possible. However, it does not seem like the **absolute smallest** panel size is the optimal configuration. There may be some small panel_size between 1 and 5 that is optimal from an $\alpha - \beta$ perspective.

Analysis of process grid layout. We then analyzed how changing the grid layout might affect the times.

- Over all of our test cases (which include all panel sizes and matrix dimensions), using a square layout improved the total running time of the algorithm. We have enough observations to say fairly confidently that this improvement is not luck - a square layout probably lends to the best tradeoff between latency and bandwidth.
- Along the same lines, we can see that our broadcast times are improved with a square layout. This probably contributed the most to total running time.
- The square layout again seemed to help the local matrix multiply, but not as evidently.

Best configurations. We found the best configurations via the following calls:

```
> ttimedata[ttimedata$meantime == min(ttimedata$meantime),]
m    n   k   px  py panel_size meantime
256 256 256   4  16    16      2354
> Atimedata[Atimedata$meantime == min(Atimedata$meantime),]

m    n   k   px  py panel_size meantime
256 256 1024 8     8       6.453125
> Btimedata[Btimedata$meantime == min(Btimedata$meantime),]
m    n   k   px  py panel_size meantime
256 256 1024 1     64      0.1171875
> mtimedata[mtimedata$meantime == min(mtimedata$meantime),]

m   n    k   px  py panel_size meantime
256 256 256  4     16      7.40625
```

A few observations:

- As expected, passing in the smallest matrices possible resulted in the smallest (mean) total time elapsed.
- For some odd reason, the 1×64 grid layout provided the best (mean) broadcast time for Bblock.
- The 8×8 grid layout provided the best (mean) broadcast time for Ablock.

4 Performance Model

We generalize some results from the square SUMMA model, which states

$$T_{net} = \alpha \frac{n}{b} log(p) + \frac{n^2}{\beta} \frac{log(p)}{\sqrt{p}}$$

We didn't have time to really derive a specific formula for rectangular matrices, but perhaps a feasible way to look at it is to view $n = average(m, n, k)$. Since n is a marker for the size of a matrix, doing this might not to be too far off from the truth.

5 Conclusions

Our analysis shows that a small panel size relative to block size and a square grid layout can definitively improve matrix multiplication over a naive implementation.