An optimization-inspired approach to parallel sorting

Team Metropolis:
James Farzi, JJ Lay, Graham West

February 13, 2019

Outline

1. The Algorithm
   - Distributing/Importing Files
   - Sorting
   - Binning
   - Exchanging data
2. Testing
   - Methodology
   - Results
3. Conclusions

The Algorithm

The Algorithm

XXX

- XXX
- XXX

XXX

- XXX
- XXX

Distributing/Importing Files

XXX

- XXX
- XXX

XXX

- XXX
- XXX

Sorting

We implemented Merge and Bubble Sort

XXX

- XXX
- XXX

XXX

- XXX
- XXX

Binning

Binary search

- Since the data is sorted, we can use a binary search to find where the bin edges lie in index space
- We can then subtract successive edges' indices to find the number of elements in that bin

xxx

- xxx
- xxx

Binning

Adapting the bins

for interior bin edges (endpoint bins stay constant):

$$\begin{aligned}\Delta C &= 2.0(c_i^n - c_{i-1}^n)/(c_i^n + c_{i-1}^n) \\ \Delta B &= b_{i+1}^n - b_i^n \\ b_i^{n+1} &= b_i^n + \alpha \Delta C \Delta B\end{aligned} \tag{1}$$

where $0 < \alpha < 0.5$ and $b_i^n < b_{i+1}^n$ for all n

Uniformity metric

$$U^n = \max(\frac{c_{\text{max}} - c_{\text{avg}}}{c_{\text{avg}}}, \frac{c_{\text{avg}} - c_{\text{min}}}{c_{\text{avg}}}) \tag{2}$$

Binning

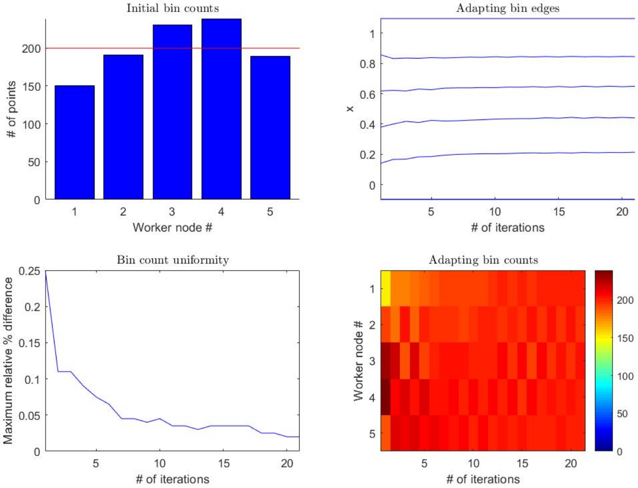

Figure 1: Using 5 nodes to uniformly bin 1000 data points

Exchanging data

XXX

- XXX
- XXX

XXX

- XXX
- XXX

Testing

Conclusions