LASSO Parallel with MPI

Yihe Yu

May 7, 2021

Why LASSO?

- LASSO is short for least absolute shrinkage and selection operator.
- It is a **regression analysis method** that performs both variable selection and regularization.
- It enhances the prediction accuracy and interpretability of the resulting statistical model.
- It has a variety of interpretations in terms of geometry, Bayesian statistics and convex analysis.
- It helps in the models analysis and provide an **optimum linear combination**.
- Its applications include **cross-section of return forecasts** and asset portfolio management etc.

Lasso: Formulation

A subset-selection problem in **linear regression**:

$$y = X\beta$$

where y is $n \times 1$, X is $n \times K$, β is $K \times 1$. n is the sample size, K is the number of features (candidate variables).

We can solve β by

$$\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

LASSO: Optimization

We can solve the optimization problem by considering it as an OLS problem with a constraint, i.e.,

$$\beta^{OLS} = (X^T X)^{-1} X^T Y$$

s.t.

$$\sum_{j=1}^{K} |\beta_j| \leq c$$

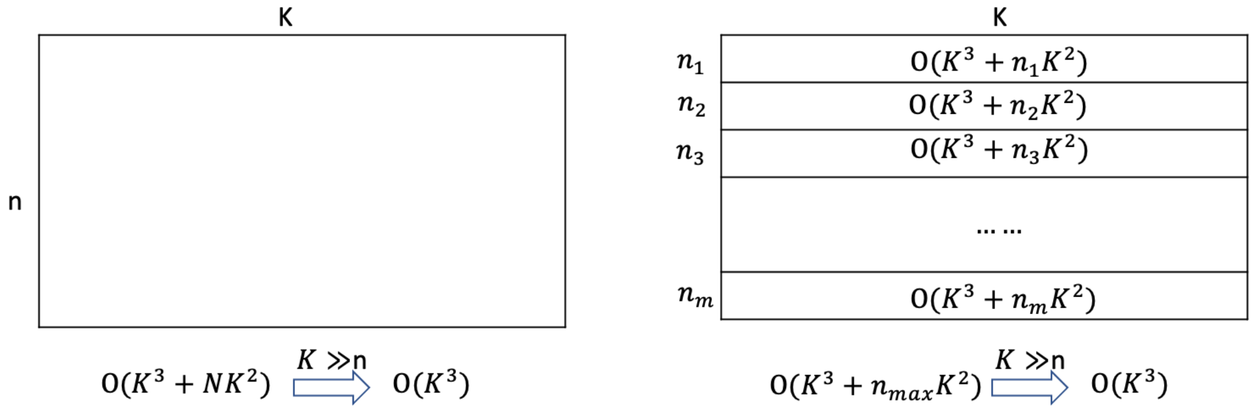
LASSO: Computational Complexity

For $(X^TX)^{-1}X^TY$,

- (X^TX) takes $\mathcal{O}(nK^2)$ time and produces a $(K \times K)$ matrix.
- The inverse of a $(K \times K)$ matrix takes $\mathcal{O}(K^3)$ time.
- (X^TY) takes $\mathcal{O}(nK^2)$ time and produces a $(K \times K)$ matrix.
- The final matrix multiplication of two $(K \times K)$ matrices takes $\mathcal{O}(K^3)$ time.

The computational complexity of LASSO implemented using LARS algorithm (Efron et al., 2004) is $\mathcal{O}(K^3 + nK^2)$.

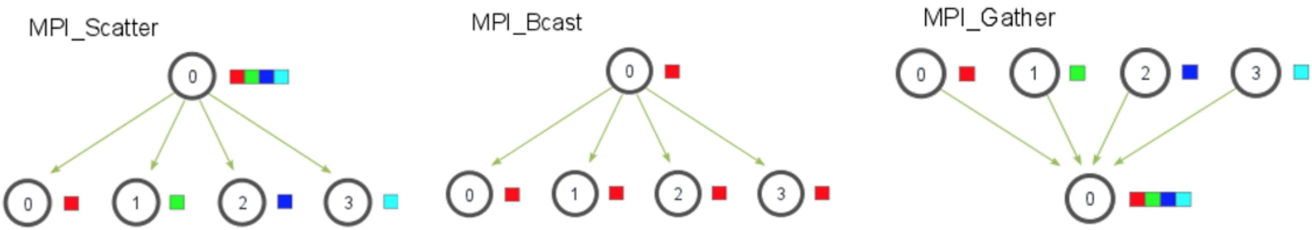
Matrix Multiplication: Parallel Implementation

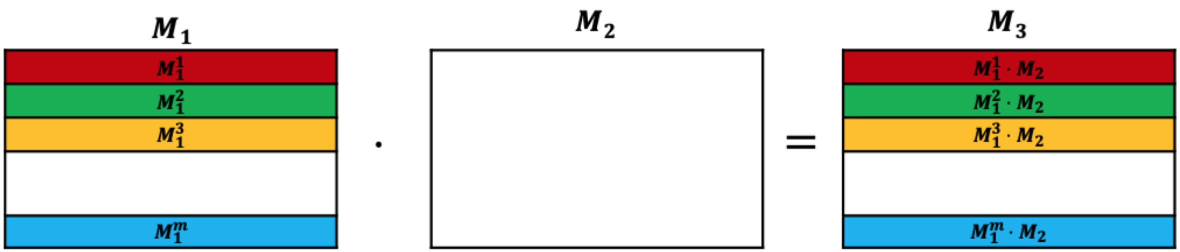
- For typical LASSO settings $K \gg n$, so the computational complexity $\mathcal{O}(K^3 + nK^2)$ then become $\mathcal{O}(K^3)$.
- 

- Therefore the data parallelism which divide the matrix X along example dimension n does not boost the regression process.
- A possible way to improve the performance is to apply MPI to the matrix multiplication.

Matrix Multiplication: MPI Implementation

Consider matrix multiplication $M_1 \cdot M_2 = M_3$.





LASSO with MPI Matrix Multiplication

Algorithm 1 LASSO coefficients

Input: $DataSet(X, Y)$, lr(learning rate), p(penalty)
Output: β (LASSO coefficients)

```
for X, Y in DataSet do
    Y_pred ← X · β
    dβ ← (−2 · X · (Y − Y_pred) + I(β) · p)/X.shape[0]
    β ← β–lr · dβ
end for
return β
```

$$Y_{pred} \leftarrow X \cdot \beta$$

$$d\beta \leftarrow (-2 \cdot X \cdot (Y - Y_{pred}) + I(\beta) \cdot p)/X.shape[0]$$

$$\beta \leftarrow \beta - lr \cdot d\beta$$

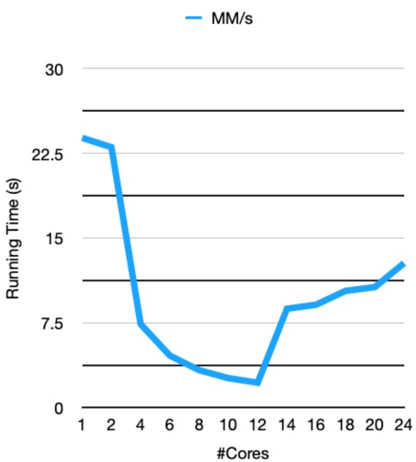
Results

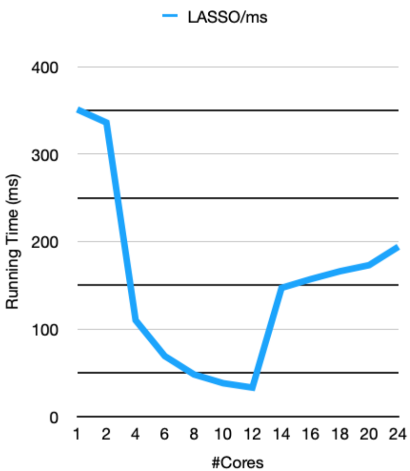
We conduct our experiment on UB-CCR debug partition nodes, which has 12 cores per node.

- If cores smaller than 12, deploy on 1 node, else on 2 nodes
- For MPI, 1 core as master and the rest as computational cores
- Matrix Multiplication: $M1 \cdot M2$, each matrix is of 500×500
- LASSO: 30 examples, each example has 250 features

Experiment (CCR debug-partition #core=12)

#Cores	MM/s	Speedup	LASSO/ms	Speedup
1	23.876	1.00	351	1.00
2	23.046	1.04	336	1.04
4	7.387	3.23	110	3.19
6	4.604	5.19	69	5.09
8	3.317	7.20	48	7.31
10	2.611	9.14	38	9.24
12	2.214	10.78	33	10.64
14	8.753	2.73	147	2.39
16	9.117	2.62	157	2.24
18	10.335	2.31	166	2.11
20	10.668	2.24	173	2.03
24	12.789	1.87	194	1.81

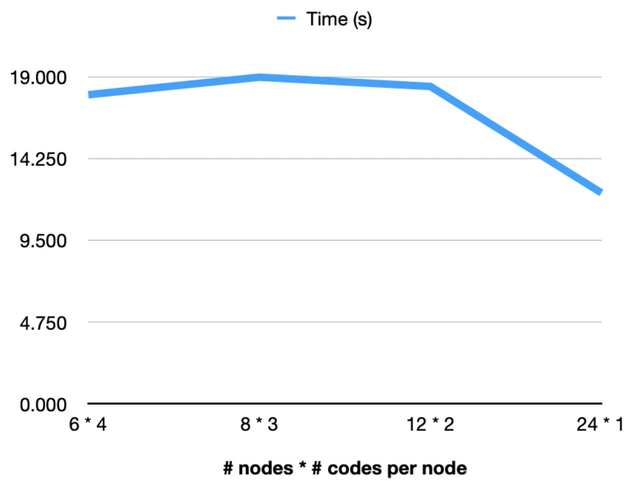




Results

We compare the time using in total 24 cores on 6, 8, 12 and 24 nodes for the Matrix Multiplication.

# nodes * # codes per node	Time (s)	Speed up
6 * 4	17.954	1x
8 * 3	18.973	0.946x
12 * 2	18.434	0.974x
24 * 1	12.253	1.465x



Conclusion

- MPI in matrix multiplication and LASSO achieves linear speedup wrt. number of cores on single node.
- LASSO has similar speed up with matrix multiplication, which shows the correctness of our computational complexity analysis and implementation.
- MPI on multiple nodes may suffer from the communication as shown in the previous section, the best performance is achieved when we utilize all the cores.

References

- Algorithms Sequential & Parallel: A Unified Approach. 3rd Ed. *Russ Miller and Laurence Boxer*. Cengage Learning. 2012.
- Least Angle Regression. *Efron, Bradley and Hastie, Trevor and Johnstone, Iain and Tibshirani, Robert*. The Annals of Statistics. 2004 Apr. http://statweb.stanford.edu/~imj/WEBLIST/2004/LarsAnnStat04.pdf
- https://ubccr.freshdesk.com/support/solutions/articles/13000010161-mpi-and-parallel-computing
- https://rabernat.github.io/research_computing/parallel-programming-with-mpi-for-python.html
- https://stats.stackexchange.com/questions/76518/what-is-the-time-complexity-of-lasso-regression
- https://www.geeksforgeeks.org/implementation-of-lasso-regression-from-scratch-using-pyt

Thank you