UNIVERSITY OF SOUTHERN CALIFORNIA

# ADMM & Shotgun: Two Parallel Solvers for LASSO


**Sanjay Purushotham, Tomer Levinboim**

*Electrical Engineering, Computer Science*

*spurusho@usc.edu, levinboi@usc.edu*


## Parallel L1-Regularization Methods

### Introduction

- L1-regularization has become popular approach for feature/variable selection problems
- L1-regularization biases learning towards sparse solutions, and is especially useful for high-dimensional problems
- LASSO is the least squares problem, subject to L1-regularization of the model:

$$\min\ (1/2)\|Ax-b\|_2^2 + \lambda\|x\|_1$$

### Algorithm 1: Alternating Direction Method of Multipliers

- **Initialize *N*** processes, along with $x_i, u_i, r_i, z$:
- **Repeat**
  1. Update $u_i := u_i + x_i - z_i$
  2. Update $x_i := argmin_x\ (f_i(x) + (\rho/2)\ \|x - z + u_i\|_2^2)$
  3. Let $w := x_i + u_i$ and $t := \ \|r_i\|_2^2$
  4. ***AllReduce*** w and t
  5. Let $z^{prev} := z$ and update $z := \boldsymbol{prox}_{g,N\rho}(w/N)$
  6. **Exit if** $\rho\sqrt{N}\|z - z^{prev}\|_2 \le \epsilon^{conv}$ and $\sqrt{t} \le \epsilon^{feas}$
  7. Update $r_i := x_i\ - z$

### Implementation

- The ADMM algorithm was implemented in C using MPI for inter-process communication and the GNU scientific library for linear algebra
- Shotgun was implemented in python, using multi-processing and shared memory. No locking mechanism was used between processes

### Algorithm 2 : Shotgun – Parallel SCD

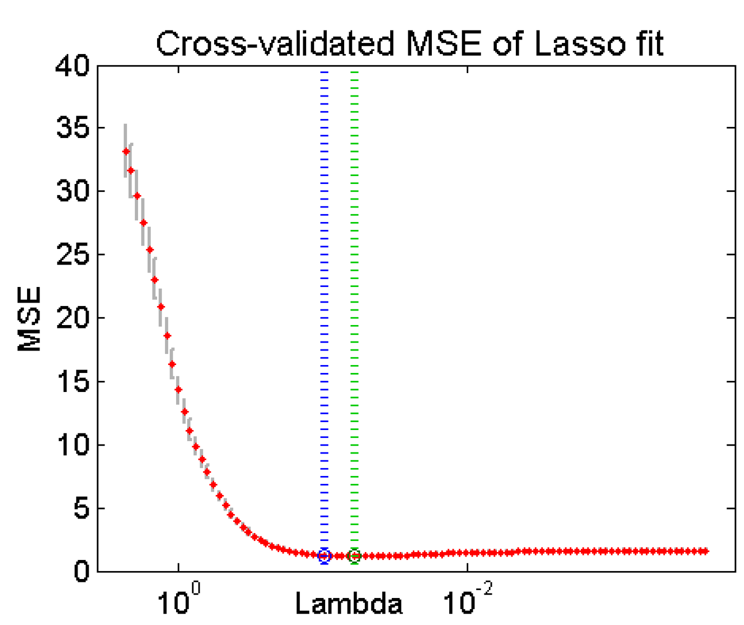
1. **While** not converged:
   a) For each $p \in P$ in parallel **do**
      i. **Sample** $j \in \{1, \dots, D\}$ uniformly at random
         A. Compute gradient $g_j := \nabla f(x^t)_j$ of j'th co-ordinate
         B. **Update** $x_j^{t+1} := S_{\lambda/\rho}(x_j^t - g_j/\rho)$

## Experimental Results

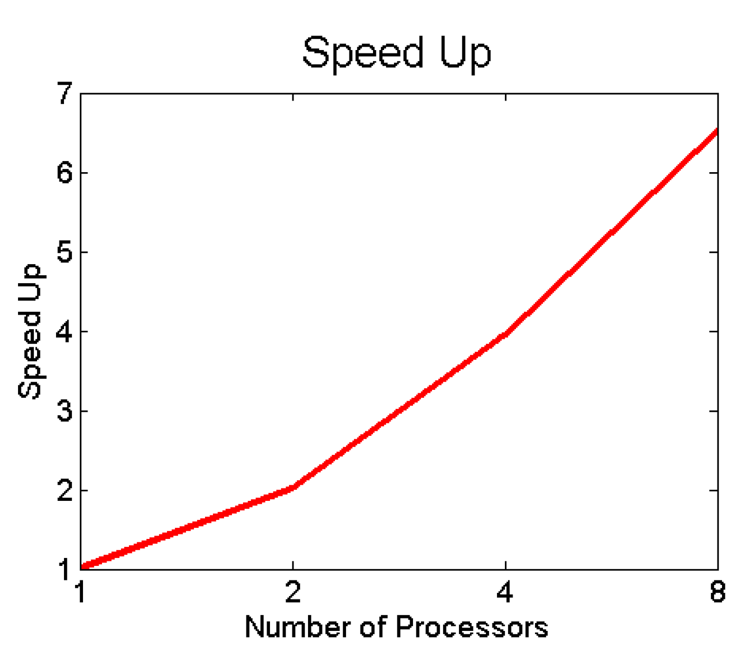
### Results

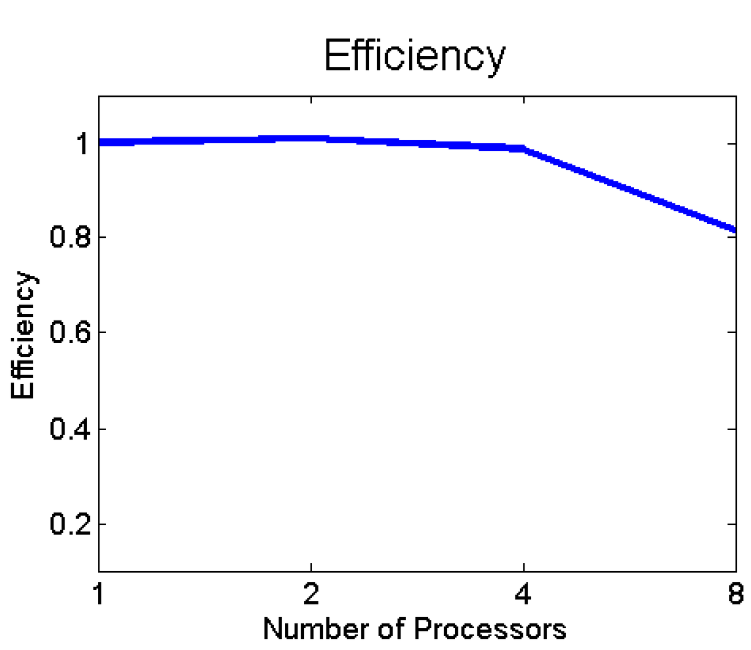
- Simulations were conducted for both dense and sparse matrices A for the high-dimensional case (p>>n) (n ~100 to 10k, p ~ 500 to 50k)

| Measures | LASSO |
|---|---|
| Avg. PE | 1.089 (0.43) |
| MSE | 0.75 (0.32) |
| Avg. FP | 11.45 (7.04) |
| Avg. FN | 0.88 (0.68) |

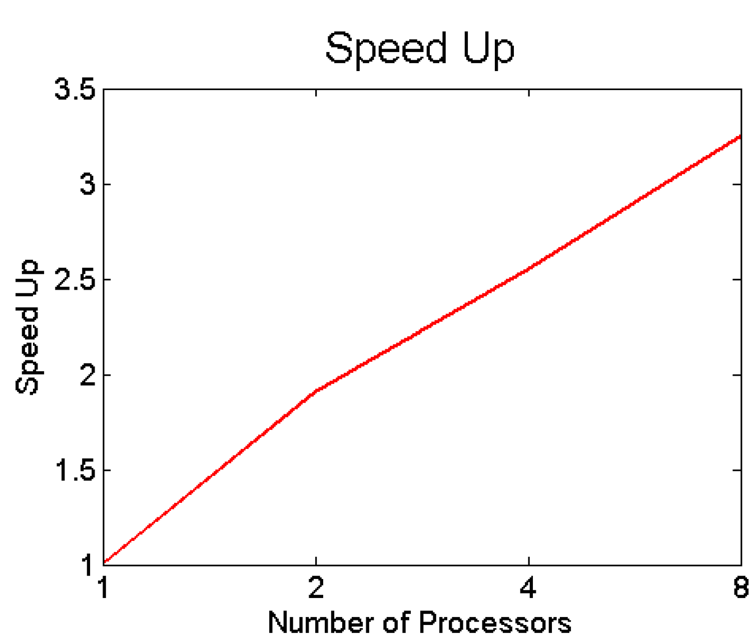


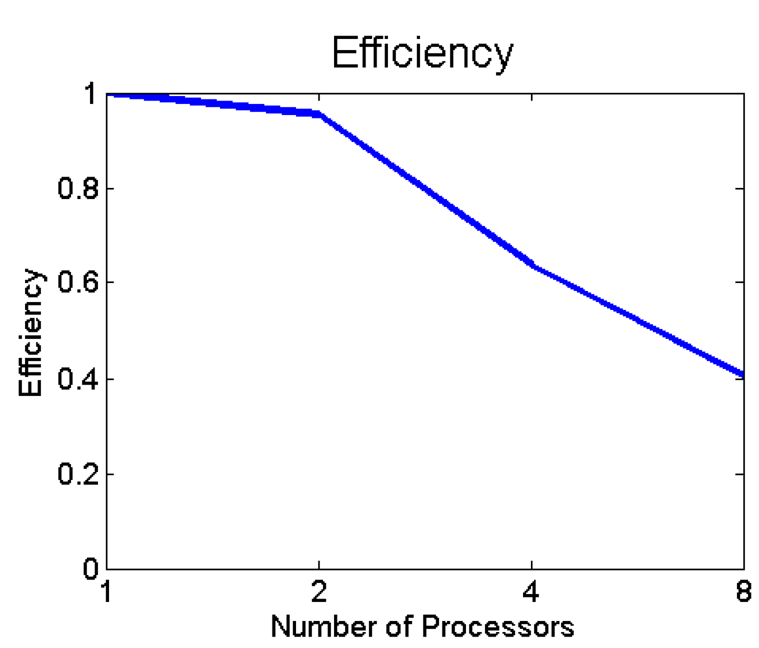

**Algorithm 1**



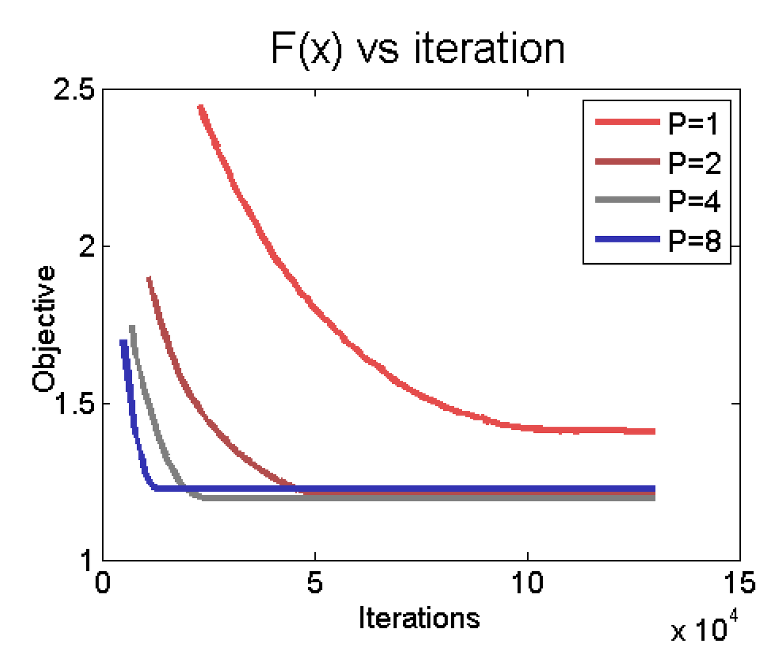



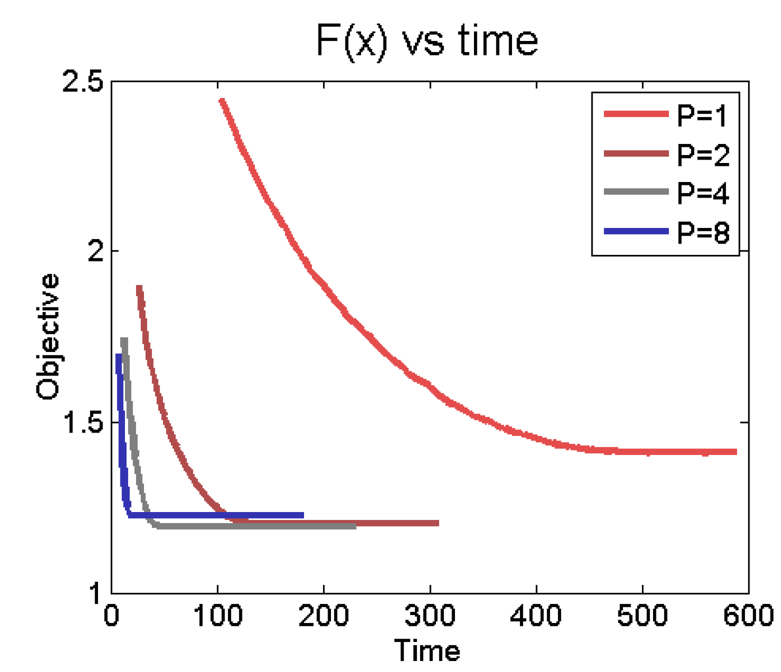

**Algorithm 2**




**Convergence**




### Discussion

- In ADMM algorithm, the dominant computation is forming and computing the Cholesky factorization - locally and in parallel. The performance of our algorithm can be further improved by using LAPACK package and hardware-optimized libraries such as Intel MKL or ATLAS
- The shotgun algorithm seems to be able to scale nicely, however, we observe that even when no locking is involved, Python's shared memory seem to suffer from scalability issues as the number of processes grows

### Conclusions

- In our project, we present two efficient parallel implementations for L1-regularization method (LASSO). Our Implementation can be further improved with specialized libraries and fine tuning
- Both our algorithms converge faster with the increase in number of processors at a slight cost on efficiency

### References


- *S Boyd, N Parikh, E Chu, B Peleato, J Eckstein*, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers", Foundations and Trends in Machine Learning, 2010
- *J Bradley, A Kyrola, D Bickson, C Guestrin*, "Parallel Coordinate Descent for $L_1$-Regularized Loss Minimization", ICML 2011