Distributed Coordinate Descent for Big Data Optimization

Martin Takáč   Jakub Mareček   Peter Richtárik

University of Edinburgh

NAIS Centre for Numerical Algorithms and Intelligent Software

1. Problem Formulation

$$\min_{x \in \mathbb{R}^n} [F(x) \equiv f(x) + \Psi(x)]$$

1. f convex, partially separable of degree ω and $\forall x \in \mathbb{R}^n, t \in \mathbb{R}$ and $i \in \{1,2,\dots,n\}$ satisfying $|\nabla_i f(x) - \nabla_i f(x + te_i)| \le L_i |t|$, where L_i are coordinate Lipschitz constants
2. Ψ convex and separable ($\Psi(x) = \sum_i \Psi_i(x^i)$)
3. Description of f so large that it does not fit onto a single computer! $\Rightarrow$ a cluster of C nodes

2. The Algorithm

Pre-processing: Partition coordinates $\{1,2,\dots,n\}$ to C sets $S_1, S_2, \dots, S_C$

In one iteration computers $c = 1,2,\dots,C$ **in parallel** do

1. Choose **random** $\hat{S}_c \subset S_c$
2. For each $i \in \hat{S}_c$ **in parallel** compute $t_i^* \leftarrow \arg\min_{t \in \mathbb{R}} \nabla_i f(x_k) t + \beta \frac{L_i}{2} t^2 + \Psi_i(x_k^i + t)$
3. $x_{k+1} \leftarrow x_k + \sum_{i \in \hat{S}_c} t_i^* e_i$

2. Distributed Sampling

We can analyze the above algorithm under the following assumptions:

- $|S_c| = \frac{n}{C}$ for all $c = 1,2,\dots,C$
- $\hat{S}_c$ is chosen uniformly as one of the subsets of S_c of cardinality τ

Distributed sampling: $\hat{S} = \cup_{c=1}^C \hat{S}_c$

$$\Downarrow$$

$$\beta := 1 + \frac{-C(\tau-1) + \omega[\tau C(1 + \frac{C-1}{n}) - 1]}{\max\{n - C, 1\}} \quad \text{(see [1])}$$

Special cases:

- $C = 1 \Rightarrow \beta = 1 + \frac{(\tau-1)(\omega-1)}{\max\{n-1,1\}}$ (see [2])
- $C = \tau = 1 \Rightarrow \beta = 1$ (see [3])

However, we need new analysis for the $C > 1$ case.

4. Complexity Theorem

$$k \ge \frac{\beta n}{\tau C} \frac{2R^2}{\epsilon} \log\left(\frac{F(x_0) - F^*}{\epsilon \rho}\right)$$

$$\Downarrow$$

$$\textbf{Prob}(F(x_k) - F(x_*) \le \epsilon) \ge 1 - \rho$$

$$(R^2 \approx \sum_i L_i (x_0^i - x_*^i)^2)$$

5. AC/DC Solver

We developed a solver (`http://code.google.com/p/ac-dc/`) for

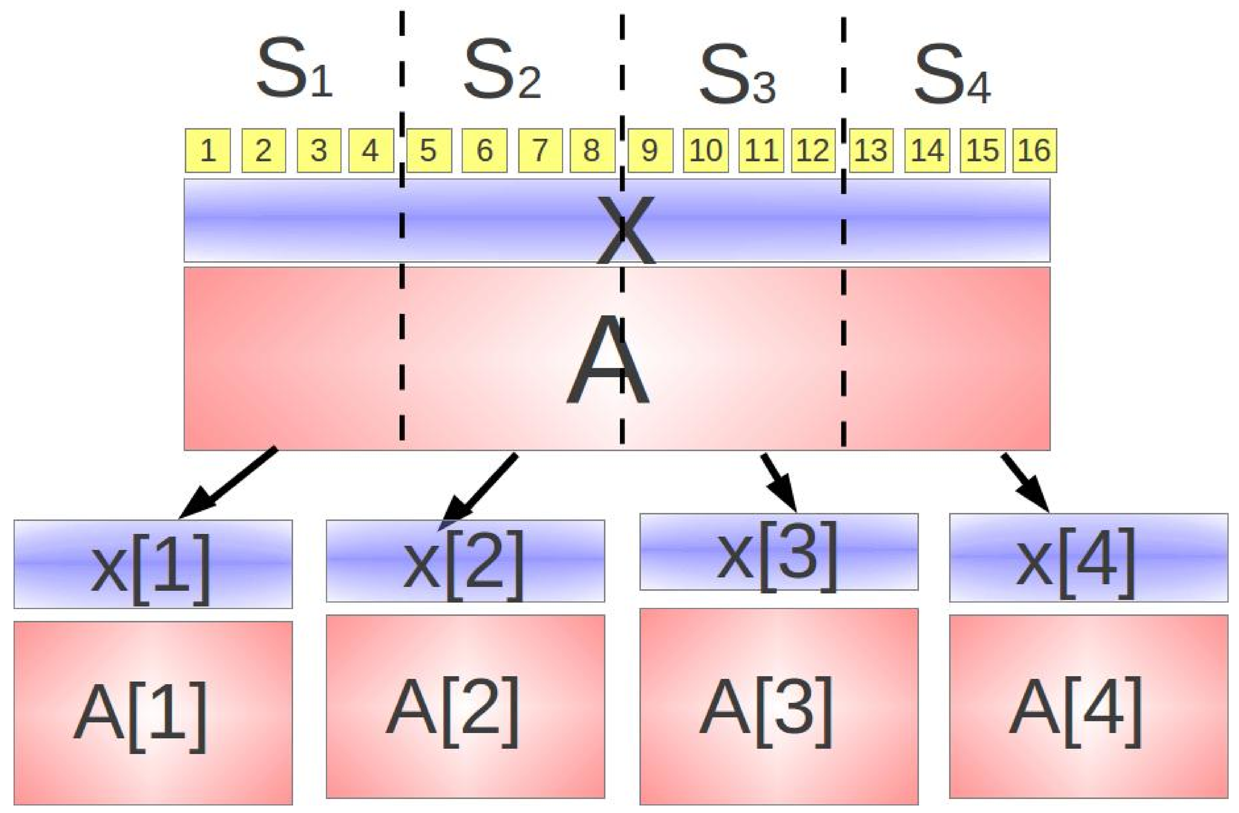
$$f(x) = \sum_{i=1}^m Loss(x; A_j, b_j), \qquad \Psi(x) = \lambda \|x\|_1$$

3 supported losses	$Loss(x, A_j, b_j)$
square loss	$\frac{1}{2}(b_j - A_j x)^2$
logistic loss	$\log(1 + e^{-b_j A_j x})$
hinge square loss	$\frac{1}{2}\max\{0, 1 - b_j A_j x\}^2$

Note that $A_j \in \mathbb{R}^n$ is a row vector and later will represent the j-th row of matrix A.

6. Data Distribution

Assume that we have $C = 4$ compute nodes and $n = 16$ coordinates. The coordinates can be partitioned into 4 balanced groups $\{S_1, S_2, S_3, S_4\}$.



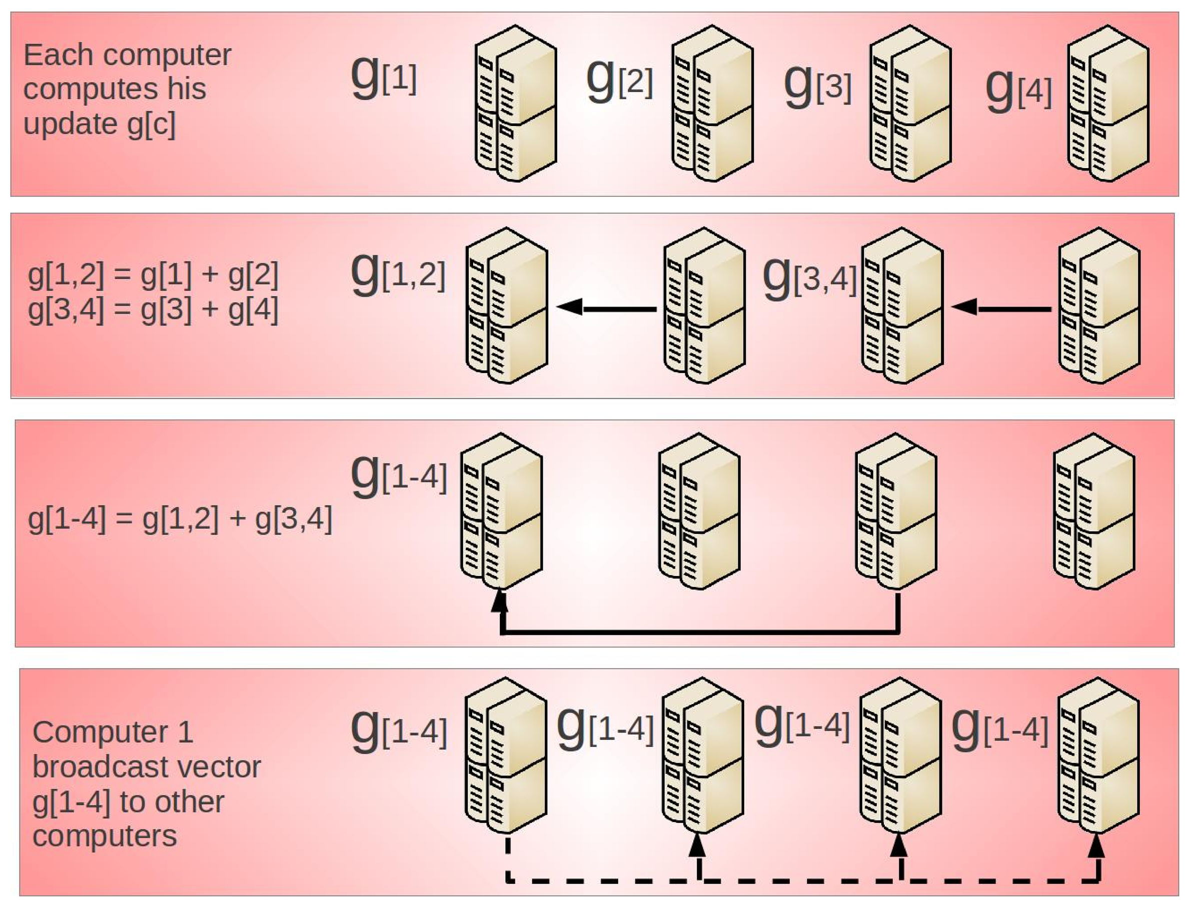
On computer 1, only the first 4 coordinates of vector x are stored and also the corresponding 4 columns of matrix A. **Data distribution is crucial** for **problems** whose **size exceeds available memory** of a single computer!

7. Implementation Details (square loss example)

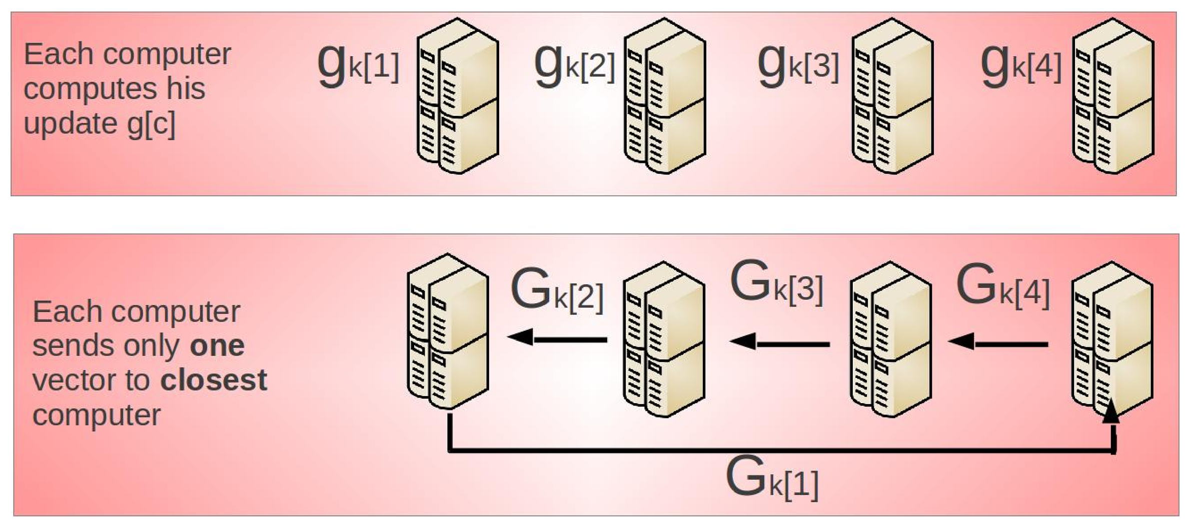
If we can maintain $g_k = Ax_k - b$ on all computers, then since $\nabla_i f(x_k) = \langle a_i, g_k \rangle$ (a_i is the i-th column of matrix A), computer c can compute $\nabla_i f(x_k)$ for $i \in S_c$, and hence the algorithm can be run.

- Note that $g_{k+1} = Ax_{k+1} - b = A(x_k + \sum_{c=1}^C \sum_{i \in \hat{S}_c} t_i^* e_i) - b = g_k + \sum_{c=1}^C \sum_{i \in \hat{S}_c} a_i t_i^*$
- That is, computer c additively contributes $g_k[c] := \sum_{i \in \hat{S}_c} a_i t_i^*$ to the update of g_k
- So, we need to add up the distributed updates $g_k[c]$

Reduce All (RA)



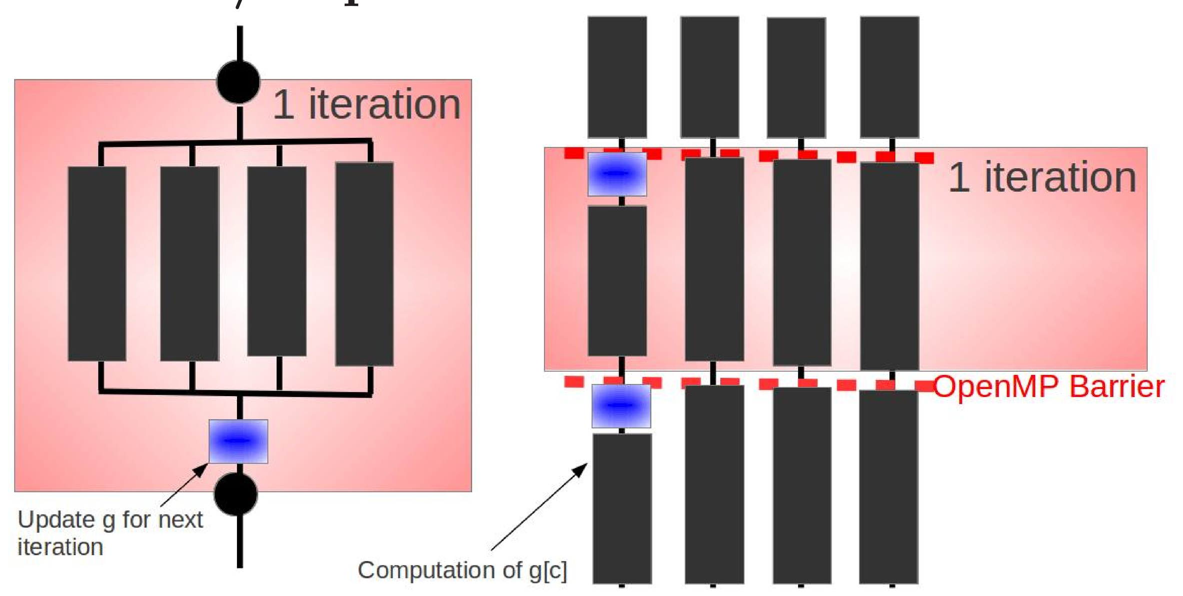
Asynchronous StreamLine (ASL)



- $G_k[c] = G_{k-1}[Prev(c)] + g_k[c] - g_{k-C}[c]$
- $g_{k+1}^c = g_k^c + g_k[c] + G_k[Prev(c)] - g_{k-C}[c]$
- ASL: **much LESS communication** that RA!
- ASL: **asynchronous** (non-blocking) communication
- ASL: **communication** only **between** two **closest computers**

8. Hybrid Implementations

Parallel/Sequential   **Just Parallel**

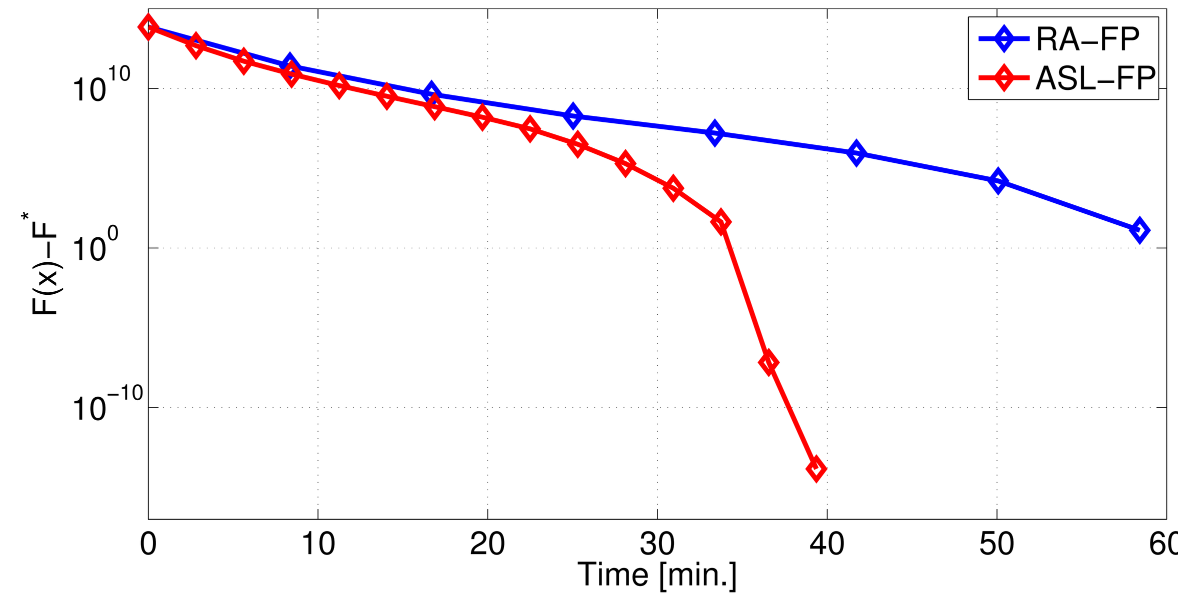


Left image shows **Parallel and Serial (PS)** approach, where each MPI process runs few OpenMP threads for **computing** t_i^* **and** $g_k[c]$ **(black boxes)** and afterwards, MPI communication takes places (blue boxes). Right image shows **Fully Parallel (FP)** approach in which one of the threads deals with communication and when waiting for a new communication, it helps the other threads to do some computation.

9. Numerical Experiments

All experiments were done on HECToR - Cray XE6 using **2,048 cores.** Problem size $A \in \mathbb{R}^{10^9 \times 5 \cdot 10^8}$ had **1.2 TBytes** and we used $\tau = 10^3$.

method	avg. time / iter.
RA-PS	2.252
RA-FP	2.052
ASL-FP	0.691



10. References

[1] Takáč, M., Mareček, J. and Richtárik, P.: Distributed coordinate descent methods for big data optimization, 2013

[2] Richtárik, P., Takáč, M.: Parallel coordinate descent methods for big data optimization, 2012

[3] Richtárik, P., Takáč, M.: Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function, Mathematical Programming, 2012