Distributed Frank-Wolfe under Pipelined Stale Synchronous Parallelism

Thomas Peel & Nam-Luc Tran

EURA NOVA

research.euranova.eu

{thomas.peel, nam-luc.tran}@euranova.eu

Introduction

Big Data analytics have reached a certain level of maturity in both industry and science. Within the context of Big Data distributed processing frameworks, iterative machine learning algorithms are usually implemented through the Bulk Synchronous Parallel (BSP) paradigm. This is namely the case for *Twister*, *Haloop* and *ScalOps*.

From the recent evolutions in data centers towards the unification of computing resources, we are witnessing the rise of resource managers such as *Mesos*, *Yarn*, *Omega* and *Corona*. These define the concept of *data center operating system* as illustrated on Figure 1.

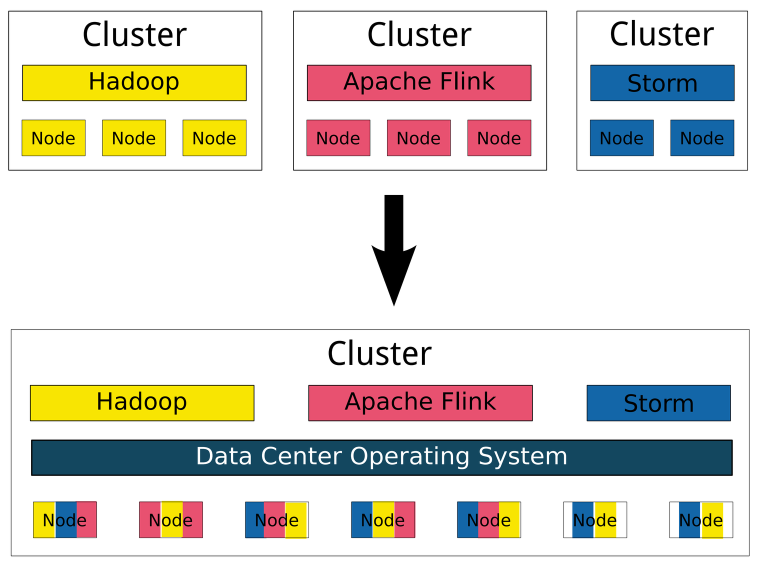

Figure 1: Within a data center OS, all resources are unified and different frameworks compete with each other for the allocation of the resources with regards to the tasks they execute.

Stale Synchronous Parallelism (SSP) for convergent and iterative algorithms

Coming back to iterative-convergent tasks, that means that a BSP worker running on a machine is not isolated anymore from other frameworks in the cluster. This leads to situations where a worker might temporarily be overloaded with other tasks, identified as a *straggler* by [2, 3]. They show that the BSP performance dramatically suffers in the presence of stragglers and propose the *Stale Synchronous Parallel programming (SSP)* as illustrated on Figure 2. In this paradigm the synchronization barrier is relaxed in each super-step and faster workers continue to iterate on stale versions of the model stored in a parameter server. Convergence and correctness of the algorithm are however still guaranteed for iterative convergent algorithms [2].

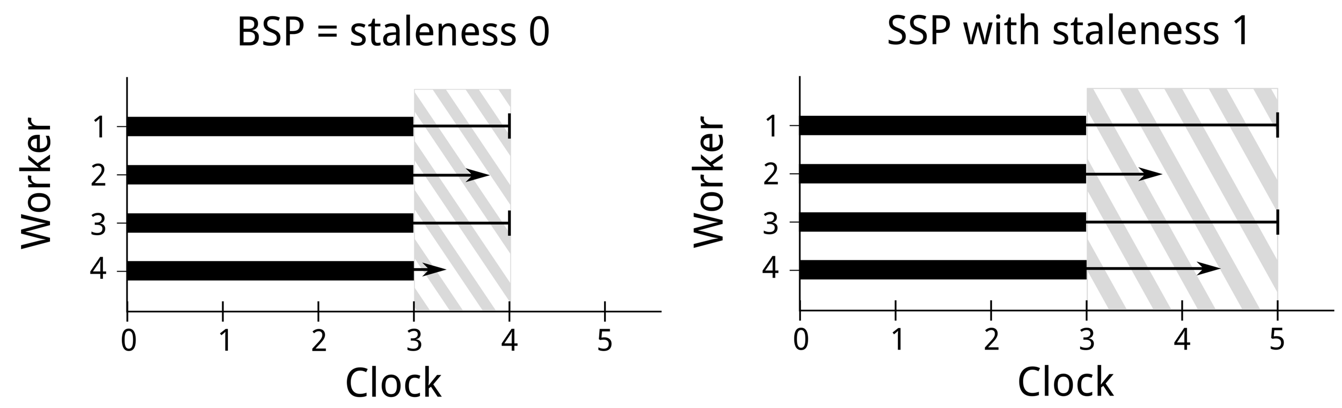

Figure 2: In Bulk Synchronous Parallelism (left), workers synchronize their update to the model after each clock. Under Stale Synchronous Parallelism (right), workers access the updates of their co-workers in a best-effort mode within the bounds of staleness.

Distributed Frank-Wolfe under SSP

The Frank-Wolfe algorithm [4] is a simple yet powerful algorithm targeting the following optimization problem :

$$\min_{x \in \mathcal{D}} f(x), \tag{1}$$

where the function f is convex and continuously differentiable, and the domain $\mathcal{D}$ is a compact convex subset of $\mathbb{R}^n$. Algorithm 1 shows the basic Frank-Wolfe algorithm.

Algorithm 1 Frank-Wolfe algorithm

Let $\boldsymbol{\alpha}^{(0)} \in \mathcal{D}$
for $k = 0, 1, 2, \dots$ **do**
  $\boldsymbol{s}^{(k)} = \arg\min_{\boldsymbol{s} \in \mathcal{D}} \langle \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}^{(k)}) \rangle$
  $\boldsymbol{\alpha}^{(k+1)} = (1-\gamma)\boldsymbol{\alpha}^{(k)} + \gamma \boldsymbol{s}^{(k)}$,
end for
stopping criterion : $\langle \boldsymbol{\alpha}^{(k)} - \boldsymbol{s}^{(k)}, \nabla f(\boldsymbol{\alpha}^{(k)}) \rangle \leq \epsilon$

In [1], the authors propose a distributed version of this algorithm to solve a separable variant of the problem stated in Eq. (1) :

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} f(\boldsymbol{\alpha}) \; s.t. \; \|\boldsymbol{\alpha}\|_1 \leq \beta, \tag{2}$$

where $f(\boldsymbol{\alpha}) = g(\boldsymbol{A}\boldsymbol{\alpha})$ for an atom matrix $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_n] \in \mathbb{R}^{d \times n}$. We consider the column-wise partitioning of $\boldsymbol{A}$ across a set of N worker nodes $V = \{v_i\}_{i=1}^N$. A node v_i is given a set of columns denoted by $\mathcal{A}_i$ such that $\bigcup_i \mathcal{A}_i = \boldsymbol{A}$ and $\mathcal{A}_i \bigcap \mathcal{A}_j = \emptyset \; \forall i \neq j$. This formulation is tightly related to optimizing over an atomic set as mentioned in [5].

The algorithm proposed in [1] relies on three steps : a first one that computes locally the best atom $\boldsymbol{s}_i$ and broadcast it to its co-workers, a second one that elects the best atom among all those candidates and a third one where each worker updates the parameter $\boldsymbol{\alpha}$ given the best atom at current iteration.

Algorithm 2 Stale Synchronous Distributed Frank-Wolfe Algorithm

Let $\boldsymbol{\alpha}_i^{(0)} = \boldsymbol{0}$, $c_i = 0$ for all worker node $v_i \in V$, $clock = 0$
for all worker node $v_i \in V$ in parallel **do**
  repeat
    if $c_i \leq clock + s$ **then**
      $\boldsymbol{\alpha}^{(c_i)} = getParameter()$
      $\boldsymbol{s}^{(c_i)} = \arg\min_{\boldsymbol{s} \in \mathcal{D}_i} \langle \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}^{(c_i)})_i \rangle$
      $\boldsymbol{\alpha}^{(c_i+1)} = (1-\gamma)\boldsymbol{\alpha}^{(c_i)} + \gamma \boldsymbol{s}^{(c_i)}$, where $\gamma = \frac{2}{k+2}$ or obtained via line-search
      $updateParameter\left(i, c_i, \boldsymbol{\alpha}^{(c_i+1)}\right)$
      $c_i = c_i + 1$
      $clock = \min\{c_i\}$
    else
      wait until $c_i \leq clock + s$
    end if
  until $\langle \boldsymbol{\alpha}^{(c_i)} - \boldsymbol{s}^{(c_i)}, \nabla f(\boldsymbol{\alpha}^{(c_i)}) \rangle \leq \epsilon$ for all worker nodes v_i
end for

We consider a setting where stragglers randomly appear among the workers. Thus, an unbalanced partitioning like the one proposed in [1] is not well suited and dynamic load-balancing scheduling policies are costly to obtain. We study the asynchronous setting where each worker can use a locally optimal atom in order to update its current possibly out-of-date, but with bounded staleness, version of the global solution. Our claim is that avoiding the synchronized update step can help the algorithm to be tolerant to the straggler problem without sacrificing convergence rate. Algorithm 2 formalizes the use of the SSP paradigm to reach our goal.

Application : LASSO regression

In this work we focus on the LASSO algorithm [7]. LASSO is a linear regression method for solving the following sparse approximation problem :

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \tfrac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{\alpha}\|_2^2 \; s.t. \; \|\boldsymbol{\alpha}\|_1 \leq \beta, \tag{3}$$

where we seek to approximate the target value y_i for the training point i by a sparse linear combination of its features a_{ij}, using the same small number of features for all data points.

Duality Gap and Line Search

For the LASSO problem, the duality-gap is fast to compute. Given that $\boldsymbol{s}$ is a minimizer of the linearised problem at point $\boldsymbol{\alpha}$, the duality gap is given by $\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle$. This quantity only depends on information that is available at each worker. Moreover, the solution to the line search problem can be obtained *analytically* with nearly no additional computational cost. Indeed, the optimal step-size is obtained by solving :

$$\gamma^* = \arg\min_{\gamma \in [0,1]} f\left(\boldsymbol{\alpha}^{(k)} + \gamma(\boldsymbol{s}^{(k)} - \boldsymbol{\alpha}^{(k)})\right) = \max\left(0, \min\left(1, \frac{\langle \boldsymbol{\alpha} - \boldsymbol{s}, \nabla f(\boldsymbol{\alpha}) \rangle}{\|\boldsymbol{A}(\boldsymbol{s} - \boldsymbol{\alpha})\|_2^2}\right)\right). \tag{4}$$

The optimal step-size involves the duality-gap value and $\|\boldsymbol{A}(\boldsymbol{s} - \boldsymbol{\alpha})\|_2^2$ which can be evaluated efficiently (because $\boldsymbol{A}(\boldsymbol{s} - \boldsymbol{\alpha})$ is available as it is involved in the duality-gap computation).

Preliminary results

The following preliminary results were obtained on a 5-nodes cluster running Apache Flink with our SSP implementation.

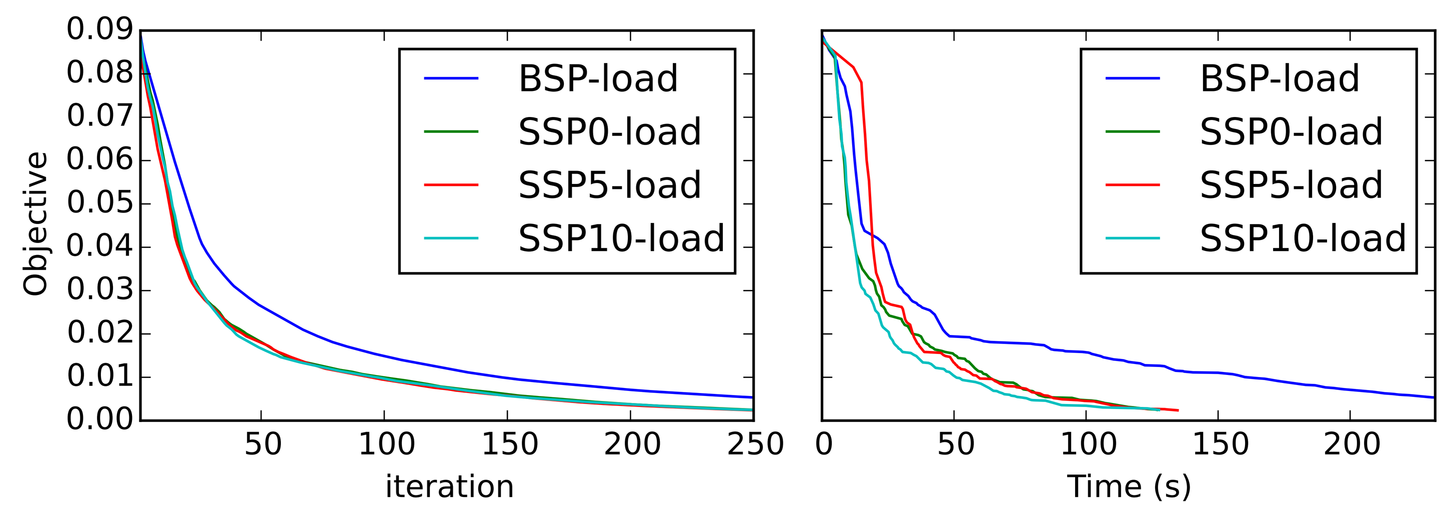

Figure 3: Convergence of the objective function in the primal with workload on the cluster.

We generate sparse uniform random matrices of dimensions 1.000×10.000 with sparsity ratio of 0.001 and a random vector $\boldsymbol{\alpha}^*$ such that $\|\boldsymbol{\alpha}^*\|_0 = 100$. Figures are averages over multiple runs of the same experiment.

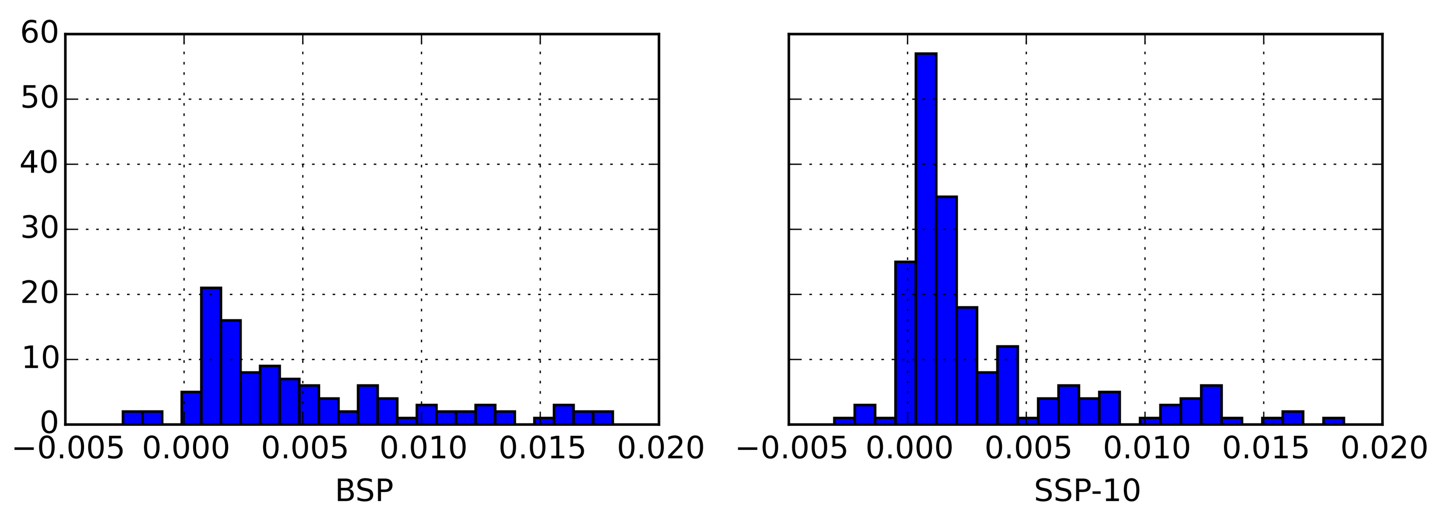

Figure 4: Distribution of the coefficients α_i after 250 iterations.

Future directions

The results shown here are encouraging. They show that the Distributed Frank-Wolfe algorithm under SSP empirically converges and is faster than its BSP counterpart. This is especially the case when random nodes in the cluster are under charge, which mimics the setup of a data center OS. Our future direction of research includes the theoretical proof for the convergence, the study of the more general context of optimizing over atomic set, the study of the sparsity of the iterates (away steps might be useful) and the comparison of our solution with other asynchronous approaches like [6, 8].

References

[1] Aurélien Bellet, Yingyu Liang, Alireza Bagheri Garakani, Fei Sha, and Maria-Florina Balcan. A Distributed Frank-Wolfe Algorithm for Communication-Efficient Sparse Learning. 2015.

[2] Henggang Cui, James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Abhimanu Kumar, Jinliang Wei, Wei Dai, Gregory R. Ganger, Phillip B. Gibbons, Garth A. Gibson, and Eric P. Xing. Exploiting bounded staleness to speed up big data analytics. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 37–48, Berkeley, CA, USA, 2014. USENIX Association.

[3] Wei Dai, Abhimanu Kumar, Jinliang Wei, Qirong Ho, Garth A. Gibson, and Eric P. Xing. High-performance distributed ML at scale through parameter server consistency models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 79–87, 2015.

[4] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

[5] Martin Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. *Proceedings of the 30th International Conference on Machine Learning*, 28:427–435, 2013.

[6] Ji Liu, Sj Wright, C Ré, and Victor Bittorf. An asynchronous parallel stochastic coordinate descent algorithm. 2014.

[7] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

[8] Yu-xiang Wang, Veeranjaneyulu Sadhanala, Wei Dai, Willie Neiswanger, Suvrit Sra, and Eric P Xing. Asynchronous Parallel Block-Coordinate Frank-Wolfe. 2014.