A Distributed Learning Architecture for Big Imaging Problems in Astrophysics

Athanasia Panousopoulou*, Sammuel Farrens†, Yannis Mastorakis*,
Jean-Luck Starck†, and Panagiotis Tsakailides*‡
*Institute of Computer Science, Foundation for Research and Technology Hellas (FORTH), Greece
†Laboratoire AIM, UMR CEA-CNRS-Paris 7, Irfu, Service d'Astrophysique, CEA Saclay, France
‡Department of Computer Science, University of Crete, Greece

***Abstract*—Future challenges in Big Imaging problems will require that traditional, "black-box" machine learning methods, be revisited from the perspective of ongoing efforts in distributed computing. This paper proposes a distributed architecture for astrophysical imagery, which exploits the Apache Spark framework for the efficient parallelization of the learning problem at hand. The use case is related to the challenging problem of deconvolving a space variant point spread function from noisy galaxy images. We conduct benchmark studies considering relevant datasets and analyze the efficacy of the herein developed parallelization approaches. The experimental results report 58% improvement in time response terms against the conventional computing solutions, while useful insights into the computational trade-offs and the limitations of Spark are extracted.**

I. Introduction

Recent advances in distributed computing are causing a paradigm shift in the analysis of petascale data [1], with the objective to enable the accurate trends prediction, typically in the domain of retail and social networking. Empowering distributed learning over large-scale scientific datasets, is considered the next emerging challenge, e.g., [2]; Considering the astrophysics domain, the respective instrumentation (e.g., Large Synoptic Survey Telescope [3]) can generate on a daily basis comparable magnitudes of data to the ones generated by popular social media platforms. At the same time, analytics over astrophysical datasets are performed at a significantly larger amount of information than the one considered in the social or Interner-based media [4].

Current bibliography trends thus highlight the benefits of using Big Data technology in large-scale astronomic imaging. Kira [5], is a representative framework, which leverages on Apache Spark [6] for speeding-up the source extraction process in astronomical imaging. An alternative approach [7] exploits the concept of Message Passing Interface [8] for parallelizing the cross-match between reference and sample catalogs. Finally, the Corral framework [9] proposes an Application Programming Interface (API) that automates the transformation of raw astronomical data into valuable information over distributed computing platforms.

These representative approaches focus on essential low-level tools for extracting datasets before applying any learning or optimization algorithm. Nevertheless, very little is reported on how learning algorithms can be profiled for providing an alternative computing solution that outperforms conventional approaches in the astrophysics domain. In this work we address this practical gap by proposing a Spark-compliant distributed architecture for exploring how the problem characteristics can be utilized for the efficient parallelization of the learning algorithm. We consider the emerging problem of removing distortion from noisy galaxy images. Our benchmark studies highlight the efficacy of the herein proposed concept versus the conventional, non-parallel approaches with respect to time response, while useful insights are extracted on the trade-off between computational capacity and memory.

II. Distributed Learning over Imaging

A common concept in learning over imaging data is that variant information from different origins is combined for removing noisy artifacts and

ISBN 978-0-9928626-7-1

extracting essential information. Considering such problems, one can think of heterogeneous imagery that correspond to the same spatial information (*e.g.*, patches of noisy and reference images) to be jointly processed for solving single/multi-objective optimization problems. As such, a substantial volume of bundled imaging data should become readily available in iterative processes for enabling large-scale imaging analytics.

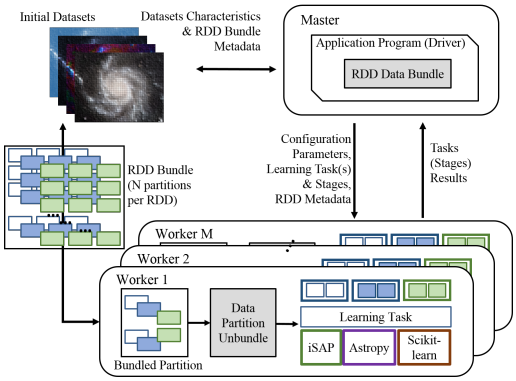

Fig. 1. The proposed distributed learning architecture.

The herein proposed architecture, presented in Fig. 1 exploits this rationale for performing distributed learning over bundled data. Motivated by the necessity of performing fast iterative computations, our architecture considers the Apache Spark framework [6], which enables in-memory storage of intermediate results. Specifically, it considers a set of networked computing resources, that are organized into a hierarchical cluster of computing resources, comprised of a master and M workers.

The learning problem and the respective datasets, are submitted to the master of the cluster through the driver program. The initial imagery datasets can be either locally available on the master or stored in a Spark-compliant distributed file system. The partitioning of the initial datasets relies on the concept of the Resilient Distributed Dataset (RDD), which is defined as a read-only collection of N data records [10]. Through the driver the application program can control several aspects of the distributed learning process, namely: (a) control how each individual dataset will be modeled as a RDD; (b) create RDD bundles through the RDD Data Bundle Component; (c) define the transformations and the respective actions on the bundled RDDs.

The resulting bundled partitions and the learning tasks are parceled into the M workers. Each worker is responsible for separating the combined inputs (RDD Data Unbundle Component), which are provided as inputs to the learning tasks. In addition to the application-defined tasks, image processing libraries (*e.g.*, Astropy [11], iSAP [12], SciKit-Learn [13]) can be incorporated into each worker for solving the learning problem at hand.

Each learning task is split into sequential computing stages, performed on different data blocks by different workers. The result of each stage returns back to the driver program, which assigns another stage of the same learning task, until all stages have been completed. This process is repeated for all learning tasks. The result can either be returned to the driver or exported to the storage system.

III. Use Case: Point Spread Function

The removal of distortions introduced by the Point Spread Function (PSF) is considered a fundamental challenge in astronomical image processing, especially when the PSF varies across the sky as is the case for the Euclid satellite [14].

An elegant method for dealing with a space variant PSF is *Object-Oriented Deconvolution* [15]. This method assumes that the individual galaxy images can be extracted and the PSF at each position is known. This leads to an inverse problem of the form $\mathbf{Y} = \mathcal{H}(\mathbf{X}) + \mathbf{n}$, where $\mathbf{Y} = [\mathbf{y}^0, \mathbf{y}^1, \cdots, \mathbf{y}^n]$ is a stack of observed noisy galaxy images, $\mathbf{X} = [\mathbf{x}^0, \mathbf{x}^1, \cdots, \mathbf{x}^n]$ is a stack of the true galaxy images, $\mathbf{n} = [\mathbf{n}^0, \mathbf{n}^1, \cdots, \mathbf{n}^n]$ is the noise corresponding to each image and $\mathcal{H}(\mathbf{X}) = [\mathbf{H}^0\mathbf{x}^0, \mathbf{H}^1\mathbf{x}^1, \cdots, \mathbf{H}^n\mathbf{x}^n]$ is an operator that represents the convolution of each galaxy image with the corresponding PSF for its position.

A typical method for addressing this problem considers a convex optimization approach for finding the solution $\hat{\mathbf{X}}$ that gives the lowest possible residual $(\mathbf{Y} - \mathcal{H}(\hat{\mathbf{X}}))$. Nevertheless, this problem is ill-posed; the tiniest amount of noise will have a large impact on the result of the operation. In order to obtain a stable and unique solution, the problem

ISBN 978-0-9928626-7-1 © EURASIP 2017

is regularized by adding prior knowledge that true images are sparse in a given domain. This leads to the following optimization problem:

$$C(\mathbf{X}) = \underset{\mathbf{X}}{\operatorname{argmin}} \left(\frac{1}{2} \|\mathbf{Y} - \mathcal{H}(\mathbf{X})\|_2^2 + \|\mathbf{W}^{(k)} \odot \Phi(\mathbf{X})\|_1 \right), \quad (1)$$

s.t. $\mathbf{X} \geq 0$, where $\|\bullet\|_2^2$ is the Frobenius norm. The Φ operator realizes the isotropic undecimated wavelet transform without the coarse scale [16], $\odot$ is the Hadamard (entry-wise) product, $\mathbf{W}^{(k)}$ is a weighting matrix, and k is a reweighting index[1] [15].

Equation (1) is implemented via a primal-dual splitting technique [17]. We herein propose Algorithm 1 for distributing the optimization phase across the proposed learning architecture. This approach entails the parallelization of $\mathbf{Y}$, PSF data, and $\mathbf{X}$, into $RDD_{\mathbf{Y}}$, RDD_{PSF}, $RDD_{\mathbf{X}}$ respectively. We exploit primitive operations of the Spark API (*i.e.*, `lambda` anonymous functions, `zip`-`map`-`reduce` operators) [18] for performing operations on the resulting RDDs. Specifically, the herein proposed solution considers the `map` transformation of RDD_{PSF} to the corresponding weighting data blocks $RDD_{\mathbf{W}}$ on each worker.

All requested input is in turn compressed into the $RDD_{<\mathbf{Y},PSF,\mathbf{W},\mathbf{X}>}$ bundle, using the `zip` transformation. The later is used to calculate the updated value of the optimization variable $\hat{\mathbf{X}}$. The cluster performs a combined `map-reduce` action, in order calculate the value of the cost function $C(\hat{\mathbf{X}})$. This process relies on the interaction between the driver and the workers, and is repeated until either the value of $C(\hat{\mathbf{X}}) \leq \epsilon$, or until the maximum number of iterations $i_{\max}$ is reached. The resulting stack of galaxy images $\hat{\mathbf{X}}$ is directly saved on the disk of the driver program.

IV. Benchmark Studies

The efficacy of Algorithm 1 is evaluated over a dataset of 10,000 galaxy images convolved with a space variant PSF with additive Gaussian noise.

[1]Matrix $\mathbf{W}^{(k)}$ is related to the standard deviation of the noise in the input images. The re-weighting index, k, is necessary to compensate for the bias introduced by using the $l1$-norm.

Algorithm 1: The PSF algorithm parallelization (driver , cluster).

Data: The data $\mathbf{Y}$, the respective PSF, the maximum number of iterations $i_{\max}$, and the cost tolerance ϵ. Typically, $i_{\max}$ =300, and $\epsilon = 10^{-4}$.
Result: The estimated images $\hat{\mathbf{X}}$ that minimize Eq. (1).

1 Initialize i =0, the matrix $\hat{\mathbf{X}}$ & calculate $\mathcal{H}(\hat{\mathbf{X}})$.
2 Parallelize $\mathbf{Y}$, PSF, $\hat{\mathbf{X}}$ into $RDD_{\mathbf{Y}}$, RDD_{PSF}, $RDD_{\hat{\mathbf{X}}}$, respectively, with N partitions per RDD.
3 Apply the weighting matrix calculator to RDD_{PSF}: $RDD_{\mathbf{W}} = RDD_{PSF}$.`map`(`lambda` $x : \mathbf{W}^{(k)}(x)$).
4 Create the RDD bundle: $RDD_{<\mathbf{Y},PSF,\mathbf{W},\hat{\mathbf{X}}>} = RDD_{\mathbf{Y}}$.`zip`($RDD_{PSF}$). `zip`($RDD_{\mathbf{W}}$).`zip`($RDD_{\hat{\mathbf{X}}}$).
5 **while** $i \leq i_{\max}$ **do**
6 Update $\hat{\mathbf{X}}$ in the bundle using [17]: $RDD_{<\mathbf{Y},PSF,\mathbf{W},\hat{\mathbf{X}}>} = RDD_{<\mathbf{Y},PSF,\mathbf{W},\hat{\mathbf{X}}>}$.`map`(`lambda` x : Update x).
7 Update $C(\hat{\mathbf{X}})$ (Eq.1): $C(\hat{\mathbf{X}}) = RDD_{<\mathbf{Y},PSF,\mathbf{W},\hat{\mathbf{X}}>}$.`map`(`lambda` x : Calculate $C(x)$).`reduce`(`lambda` $x, y : x + y$).
8 **if** $C(\hat{\mathbf{X}}) \leq \epsilon$ **then**
9 break
  end
10 $i \leftarrow i$+1.
 end
11 Save the data tuple to disk and return $\hat{\mathbf{X}}$ to driver.

The images were obtained from the GREAT3 challenge [19]. In total there are 600 unique PSFs [20], which are down-sampled by a factor of 6 to avoid aliasing issues when convolving with the galaxy images [21].

The distributed learning architecture features Spark 2.0.0, deployed over a cluster of M =6 workers. The driver allocates 6GB RAM, while each worker allocates 2.8GB RAM and 4 CPU cores. The resulting cluster has in total 24 CPU cores and 16.8GB RAM.

Our studies emphasize on execution time and disk usage with respect to $N = \{1, 32, 48, 64, 80, 128\}$ partitions per RDD, where N =1 corresponds to the standalone solution [2]. Notably, considering other parallel computing frameworks for comparison (e.g., Hadoop [22], Kira[5]) is beyond the scope of this work, as they are either unsuitable for the essential iterative computations typically met in learning imaging problems, or focus on the extraction of astronomical imaging data.

[2]https://github.com/sfarrens/sf_deconvolve

ISBN 978-0-9928626-7-1 © EURASIP 2017

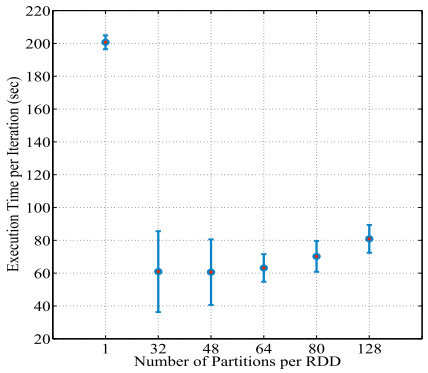

Fig. 2. The mean execution time per iteration of the optimization loop for solving Eq. (1).

Execution Time. Figure 2 presents the mean execution time per iteration of the optimization process for the calculation of $C(\hat{\mathbf{X}})$. The conventional approach (N =1) requires $\sim$ 200 secs per iteration, opposed to the parallelized approach, for which the execution time remains $<$ 85 sec. Notably, as the number of partitions increases the mean time of execution additionally increases; as N 32$\rightarrow$128, the mean execution time 60.9 sec $\rightarrow$ 80.9 sec. This is due to the fact that as N becomes larger the number of learning tasks, and subsequently the number of map-reduce networked interactions between the workers and the driver also increases. Even so, as the value of N increases the size of the individual data blocks decreases, leading to reduced load for each computing stage per task and worker. Thus, for N 32 $\rightarrow$ 128 the cluster exhibits a more stable behavior in terms of time response, and the standard deviation of the execution time decreases from 24.66 sec to 8.48 sec.

Disk Usage. For measuring the disk usage we employed the Spark-automated log files that are generated on each worker during the execution of the application program. The disk usage for the calculation of the intermediate results with respect to the time elapsed per worker is presented in Fig. 3. We observe that as the number of partitions increases, the requirements in disk usage decrease; when N 48$\rightarrow$ 128 the disk usage after an hour of execution drops from 9.3 GB to 6.6 GB. The increased disk demands illustrate that the memory allocated per worker is not sufficient for storage and computations. In addition, these results highlight that the scale of an input imaging dataset is relevant to the RAM capacity of the cluster, and the demands of the solving approach; while the initial dataset herein considered belongs to the order of MB, it unfolds to intermediate results that can rapidly reach the order of GBs.

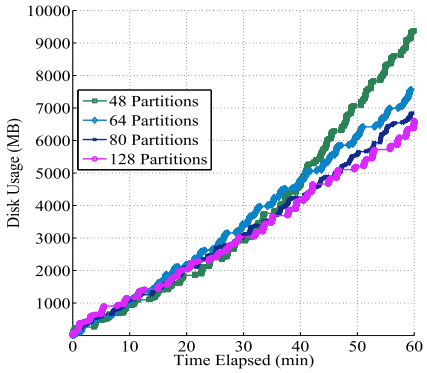

Fig. 3. The disk usage per worker during the $C(\hat{\mathbf{X}})$ calculation.

The results thus far highlight the trade off between CPU work load and memory usage. A small number of partitions results in fewer data blocks with relatively large size and increased demands in memory, which in turn increases the disk demands. On the other hand, as the number of partitions increases, the size of the data blocks decreases, and subsequently, the disk usage requirements become more relaxed. However, more stages are needed to complete an action, which implies more CPU computations per worker and learning task.

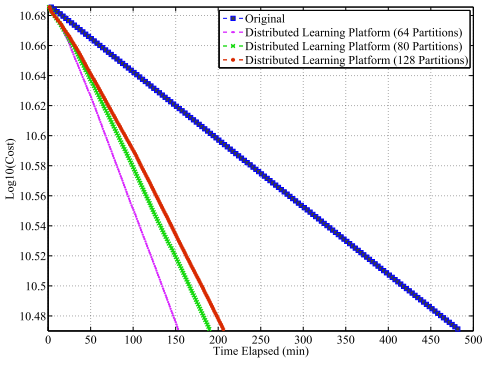

Fig. 4. The convergence of $C(\hat{\mathbf{X}})$ w.r.t. the time elapsed.

ISBN 978-0-9928626-7-1 © EURASIP 2017

Convergence Behaviour. Figure 4 illustrates the convergence behavior of the value of $C(\hat{\mathbf{X}})$ versus the time elapsed when $i_{\max} = 150$, and either the standalone or PSF parallelization (Algorithm 1) approach is adopted. The distributed learning approach is 58.33% faster than the conventional one; the completion time of the standalone approach equals to $\sim$480 min, opposed to the parallelized version, which does not exceed 190 min and 210 min for N =80 and N =128 respectively. These results highlight the fact that despite the memory and disk overhead for storing intermediate results on each worker, the herein proposed solution is extremely beneficial in terms of time response for enabling large-scale imaging analytics.

V. Conclusions

In this paper we propose an Spark-based distributed learning architecture for bundled imaging data and present the respective algorithm parallelization for a high-impact use case from the computational astrophysics domain. The benchmark studies highlight the practical benefits of changing the implementation exemplar and moving towards distributed computational approaches. Considering the lessons learned from this study, our next steps are related to the extension of the proposed architecture towards the design of an API for emerging astrophysical problems.

Acknowledgment

This work was funded by the DEDALE project (no. 665044) within the H2020 Framework Program of the European Commission.

References

[1] H. Hu, Y. Wen, T. S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE Access*, vol. 2, pp. 652–687, 2014.

[2] V. Marx, "Biology: The big challenges of big data," *Nature*, vol. 498, no. 7453, pp. 255–260, 2013.

[3] B. Jain, D. Spergel, R. Bean, A. Connolly,*et al.*, "The whole is greater than the sum of the parts: Optimizing the joint science return from lsst, euclid and wfirst," *arXiv preprint arXiv:1501.07897*, 2015.

[4] P. Huijse, P. A. Estevez, P. Protopapas, J. C. Principe, and P. Zegers, "Computational intelligence challenges and applications on large-scale astronomical time series databases," *IEEE Computational Intelligence Magazine*, vol. 9, no. 3, pp. 27–39, Aug 2014.

[5] Z. Zhang, K. Barbary, F. A. Nothaft, E. R. Sparks, *et al.*, "Kira: Processing astronomy imagery using big data technology," *IEEE Transactions on Big Data*, no. 99, 2016.

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, ser. HotCloud'10. 2010, pp. 10–10.

[7] X. Jia and Q. Luo, "Multi-assignment single joins for parallel cross-match of astronomic catalogs on heterogeneous clusters," in *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, ser. SSDBM '16. 2016, pp. 12:1–12:12.

[8] D. B. Kirk and W. H. Wen-Mei, *Programming massively parallel processors: a hands-on approach*. Morgan Kaufmann, 2016.

[9] J. B. Cabral, B. Snchez, M. Beroiz, M. Domnguez, M. Lares, *et al.*, "Corral framework: Trustworthy and fully functional data intensive parallel astronomical pipelines,*arXiv preprint arXiv:1701.05566*, 2017.

[10] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'12. 2012, pp. 2–2.

[11] T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, *et al.*, "Astropy: A community python package for astronomy," *Astronomy & Astrophysics*, vol. 558, p. A33, 2013.

[12] O. Fourt, J.-L. Starck, F. Sureau, J. Bobin, Y. Moudden, *et al.*, "isap: Interactive sparse astronomical data analysis packages," *Astrophysics Source Code Library*, 2013.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] R. Laureijs, J. Amiaux, S. Arduini, J. . Augueres, J. Brinchmann, *et al.*, "Euclid Definition Study Report," *ArXiv e-prints*, Oct. 2011.

[15] S. Farrens, F.M. Ngole Mboula, and J.-L. Starck, "Space variant deconvolution of galaxy survey images," *Astronomy & Astrophysics*, vol. 601, A66 , 2017.

[16] J.-L. Starck, F. Murtagh, and M. Bertero, "Starlet transform in astronomical data processing." in *Handbook of Mathematical Methods in Imaging*, O. Scherzer, Ed. Springer, 2015, pp. 2053–2098.

[17] L. Condat, "A primal-dual splitting method for convex optimization involving Lipschitzian, proximable and linear composite terms," *Journal of Optimization Theory and Applications*, vol. 158, no. 2, pp. 460–479, 2013.

[18] https://spark.apache.org/releases/spark-release-2-0-0.html

[19] R. Mandelbaum, B. Rowe, J. Bosch *et al.*, "The Third Gravitational Lensing Accuracy Testing (GREAT3) Challenge Handbook," *ApJS*, vol. 212, p. 5, May 2014.

[20] T. Kuntzer, M. Tewes, and F. Courbin, "Stellar classification from single-band imaging using machine learning," *Astronomy & Astrophysics*, vol. 591, p. A54, Jun. 2016.

[21] M. Cropper, H. Hoekstra, T. Kitching, R. Massey, J. Amiaux, *et al.*, "Defining a weak lensing experiment in space," *MNRAS*, vol. 431, pp. 3103–3126, Jun. 2013.

[22] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

ISBN 978-0-9928626-7-1