Evaluate the Dask distributed computing framework in respect to various scientific computing tasks

Jeyhun Abbasov
Institute of Computer Science
University of Tartu
Ulikooli 18, 50090 Tartu, Estonia
jeyhun.abbasov@ut.ee

Abstract— **Data Science is becoming more prevalent every day. That is why, there are a lot software libraries for data manipulation and analysis. One of the software libraries that using for scientific computing tasks is Dask. Dask scales Python code from multi-core local machines to large distributed clusters in the cloud. In this paper, we will evaluate the Dask distributed computing framework in respect to various scientific computing tasks. We will look at how the reading file speed comparison, computing metrics (mean, std), finding the unique value, cumulative sum and groupby aggregation differ on different platforms.**

I. INTRODUCTION

Dask is a flexible library for parallel computing in Python. Dask is composed of two parts:

- Dynamic task scheduling optimized for computation. This is similar to Airflow, Luigi, Celery, or Make, but optimized for interactive computational workloads
- "Big Data" collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers

Dask emphasizes the following virtues:

- Familiar: Provides parallelized NumPy array and Pandas DataFrame objects
- Flexible: Provides a task scheduling interface for more custom workloads and integration with other projects
- Native: Enables distributed computing in pure Python with access to the PyData stack
- Fast: Operates with low overhead, low latency, and minimal serialization necessary for fast numerical algorithms
- Scales up: Runs resiliently on clusters with 1000s of cores
- Scales down: Trivial to set up and run on a laptop in a single process
- Responsive: Designed with interactive computing in mind, it provides rapid feedback and diagnostics to aid humans

Dask is convenient on a laptop. It installs trivially with conda or pip and extends the size of convenient datasets from "fits in memory" to "fits on disk". Dask can scale to a cluster of 100s of machines. It is resilient, elastic, data local, and low latency. For more information, see the documentation about the distributed scheduler. This ease of transition between single-machine to moderate cluster enables users to both start simple and grow when necessary.

Dask represents parallel computations with task graphs. These directed acyclic graphs may have arbitrary structure, which enables both developers and users the freedom to build sophisticated algorithms and to handle messy situations not easily managed by the map/filter/groupby paradigm common in most data engineering frameworks. We originally needed this complexity to build complex algorithms for n-dimensional arrays but have found it to be equally valuable when dealing with messy situations in everyday problems.

We fill find the answers for the following research questions:

- RQ 1. How Dask's parallel and out-of-core computation extends the effective scale of modern hardware to larger datasets?
- RQ 2. How these ideas can be more broadly applied to other parallel collections?
- RQ 3. How the Dask performs on various scientific computing tasks?

II. BACKGROUND

"Dask Parallel Computation with Blocked algorithms and Task Scheduling" that was published in 2015 by

Matthew Rocklin is one of the main articles used in this paper. Their main focus was to investigate how the performance differs on dask parallel computation with blocked algorithms and task Scheduling. They explores dask graphs, dask arrays, dynamic task scheduling, dask for general computing.

Furthermore, the works [5, 12] analyze the many elements of Dask distributed computing framework, and how it could be applied to various scientific computing tasks. In [5], the authors explores under the hood, CARS makes use of Dask graph-based and distributed computing of the different steps with lazy evaluation and dynamic task allocation, allowing to run the computation on several tenth of nodes seamlessly.

Finally, work [16, 17] assesses the performance of the Apache Hadoop, Apache Spark, MPI software libraries for large scale data sets.

III. Characteristics

IV. Evaluation

V. Conclusions

References

[1] M. Dugré, V. Hayot-Sasson and T. Glatard, ”A Performance Comparison of Dask and Apache Spark for Data-Intensive Neuroimaging Pipelines,” 2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), 2019, pp. 40-49, doi: 10.1109/WORKS49585.2019.00010.

[2] S. Böhm and J. Beránek, ”Runtime vs Scheduler: Analyzing Dask's Overheads,” 2020 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS), 2020, pp. 1-8, doi: 10.1109/WORKS51914.2020.00006.

[3] Mehta, Parmita and Dorkenwald, Sven and Zhao, Dongfang and Kaftan, Tomer and Cheung, Alvin and Balazinska, Magdalena and Rokem, Ariel and Connolly, Andrew and Vanderplas, Jacob and AlSayyad, Yusra. (2016). Comparative Evaluation of Big-Data Systems on Scientific Image Analytics Workloads. Proceedings of the VLDB Endowment. 10. 10.14778/3137628.3137634.

[4] Hernández, B. et al. (2020). Performance Evaluation of Python Based Data Analytics Frameworks in Summit: Early Experiences. In: Nichols, J., Verastegui, B., Maccabe, A.‘., Hernandez, O., Parete-Koon, S., Ahearn, T. (eds) Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI. SMC 2020. Communications in Computer and Information Science, vol 1315. Springer, Cham. https://doi.org/10.1007/978-3-030-63393-6-24

[5] D. Youssefi et al., ”CARS: A Photogrammetry Pipeline Using Dask Graphs to Construct A Global 3D Model,” IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium, 2020, pp. 453-456, doi: 10.1109/IGARSS39084.2020.9324020.

[6] Ioannis Paraskevakos, Andre Luckow, Mahzad Khoshlessan, George Chantzialexiou, Thomas E. Cheatham, Oliver Beckstein, Geoffrey C. Fox, and Shantenu Jha. 2018. Task-parallel Analysis of Molecular Dynamics Trajectories. In Proceedings of the 47th International Conference on Parallel Processing (ICPP 2018). Association for Computing Machinery, New York, NY, USA, Article 49, 1–10. https://doi.org/10.1145/3225058.3225128

[7] A. Shafi, J. M. Hashmi, H. Subramoni and D. K. D. Panda, ”Efficient MPI-based Communication for GPU-Accelerated Dask Applications,” 2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid), 2021, pp. 277-286, doi: 10.1109/CCGrid51090.2021.00037.

[8] Elibol, Melih and Benara, Vinamra and Yagati, Samyu and Zheng, Lianmin and Cheung, Alvin and Jordan, Michael and Stoica, Ion. (2022). NumS: Scalable Array Programming for the Cloud. 10.48550/arXiv.2206.14276.

[9] J. Crist, ”Dask and Numba: Simple libraries for optimizing scientific python code,” 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 2342-2343, doi: 10.1109/BigData.2016.7840867.

[10] Rocklin, Matthew. (2015). Dask: Parallel Computation with Blocked algorithms and Task Scheduling. 126-132. 10.25080/Majora-7b98e3ed-013.

[11] Sievert, Scott and Augspurger, Tom and Rocklin, Matthew. (2019). Better and faster hyperparameter optimization with Dask. 118-125. 10.25080/Majora-7ddc1dd1-011.

[12] T. E. Oliphant, ”Python for Scientific Computing,” in Computing in Science and Engineering, vol. 9, no. 3, pp. 10-20, May-June 2007, doi: 10.1109/MCSE.2007.58.

[13] Shafi, Aamir and Hashmi, Jahanzeb and Subramoni, Hari and K., Dhabaleswar and Panda,. (2021). Efficient MPI-based Communication for GPU-Accelerated Dask Applications.

[14] Kumar, Deepa and Rahman, M. (2017). Performance Evaluation of Apache Spark Vs MPI: A Practical Case Study on Twitter Sentiment Analysis. Journal of Computer Science. 13. 781-794. 10.3844/jcssp.2017.781.794.

[15] Ahmed, Nasim and Barczak, Andre and Susnjak, Teo and Rashid, Mohammad. (2020). A Comprehensive Performance Analysis of Apache Hadoop and Apache Spark for Large Scale Data Sets Using HiBench. 10.21203/rs.3.rs-43526/v1.