Shared Memory High Throughput Computing with Apache Arrow™

Geoffrey Lentner
glentner@purdue.edu
Purdue University
West Lafayette, Indiana

ABSTRACT

As the barriers to entry in scientific computing have lowered with languages like Python and their libraries, the demand for ever more sophisticated and capable frameworks that provide almost turn-key functionality has grown proportionally (see *keras*).[1]

For researchers who use the *pilot job*,[2] *many-task*[3] design pattern for *high-throughput computing*[4] on modern systems, the *Apache Arrow* project [1] with its now included *Plasma* in-memory object store provides a high-level interface for sharing data structures between processes in a way that requires no serialization or copying of that data.

This poster outlines a common scenario in which a constraint is induced by the way memory is managed. The direct sharing of memory by a common middle data layer allows for accelerated workflows that can operate at a more rapid cadence. A *real-world* example is provided; the goal is to raise awareness in facilitators at other major research computing centers who work with users to architect similar such high-throughput data pipelines.

CCS CONCEPTS

• **General and reference** → **Performance**; • **Information systems** → **Data structures**; • **Software and its engineering** → **Software libraries and repositories**; • **Computing methodologies**; • **Applied computing** → Astronomy;

KEYWORDS

shared memory, in-memory formats, high-throughput computing, many-task computing

ACM Reference Format:
Geoffrey Lentner. 2019. Shared Memory High Throughput Computing with Apache Arrow™ . In *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3332186.3335197

1 BACKGROUND

There are multiple paradigms with differing requirements and limitations in terms of compute speed, memory, IOPS, etc. Traditionally, HPC has been defined by workloads involving large coupled tasks that require high-speed low-latency interconnects that allow an operation to be distributed across a cluster of homogeneous nodes. More common in recent years, the objective is instead to processes large volumes of data (either in size or in number), where the tasks are weakly coupled (if not entirely independent). In the many-task scenario, a common and effective design pattern is to submit one or more jobs to a cluster that iterate through a subset of the tasks – a *pilot job*.

This is not a new problem and tools exist for managing these workflows; e.g., *Launcher* [5], *GNU Parallel* [4]. There exists a subclass here; however, in which a potentially very large reference dataset is required for many or all of the individual tasks. This is problematic in a number of ways depending on the solution pursued by the user. (1) Attempt some form of *out-of-core* computing[5] where each task pulls in partial data as it's needed; or (2) the task execution is handled manually and each *worker* has its own copy of the reference data – holding it in memory between tasks. In the first case, a non-trivial amount of time is spent on data loading; in the second, there may not be enough memory for that level of duplication. In either case, we are spending time and resources not on the task, but the prerequisites.

Further, an exotic varient of the previously described workflow arises when these data analysis tasks come in batches where completing the tasks soon after arrival matters. In these cases, a form of job *preemtion* or *ahead-of-time* scheduling would allow for a job to start *before* the data arrives and begin initial setup. In the case of a large reference dataset, being able to load that into memory ahead of time and access it with a comparatively no-overhead proxy reference offers a wholely new capability.

[1] *Keras* is a high-level library for developing deep neural networks that abstracts away the complexities of the algorithms and linear algebra involved. It has lowered the barrier to entry for deep learning to the extent that even novice programmers can be productive almost immediately. [3]

[2] A *pilot job* is a batch job submitted to a computing cluster in which the job script is not the task itself but merely acquires the resource. The job manages the task execution manually or with a tool such as *GNU Parallel*. The idea being that the individual tasks are short enough in duration that submitting them to the scheduler would be ineffectual both in the limit in allowed total jobs as well as the latency involved in the scheduling software.

[3] *Many-task computing* (MTC) is a relatively new paradigm to be defined that bridges the gap between *high-performance computing* (HPC) and *high-throughput computing* (HTC). MTC is characterized by many small, weakly coupled (or entirely independent) tasks. There is only a subtle distinction between MTC and HTC.

[4] *High-throughput computing* (HTC) is a paradigm in which the emphasis and constraint is not necessarily the compute speed but the volume of tasks completed over a long period of time. In some cases it is characterized by the rate at which data is being processed (as opposed to FLOPS). It can be distinguished from *high performance computing* (HPC) in the required low-latency interconnects. HTC can be distributed across disparate systems and even administrative boundaries.

[5] *Out-of-core* computing is a design pattern where data is *lazy-loaded* at the last possible moment before an operation and then immediately offloaded. This is necessary in situations where there is not enough memory capacity for the full dataset and it must be decomposed into blocks that can be operated on iteratively.

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07.
https://doi.org/10.1145/3332186.3335197

2 DISCUSSION

The author has developed such a pipeline for the *Time-Domain Astronomy*[6] group at Purdue University. We use a relatively large body of synthetic *light-curves*[7] of various types of *supernovae*[8] along side a *deep neural network* to categorize and forecast new supernova candidates from on-going astronomical surveys[9] in order to recommend targets to participating facilities – so they can engage in higher resolution follow-up observations. We expect some thousands of candidate targets in a steady stream over the course of a given night. We are able to preemptively schedule jobs ahead of time; the pilot job loads the reference data as well as a significant number of prior observations into the *Plasma* store, then awaits stellar target identification numbers on a queue. The forecasting pipeline for a single target is greatly accelerated by the fact that it only need acquire a proxy reference to existing data structures via a *client* connection to the Plasma store; otherwise, it would need to load the reference dataset itself.

This is a tangible example of a growing need in research computing. The purpose here is to demonstrate the usefulness of this framework and present a conversation piece for other facilitators.

3 POSTER

The poster will include descriptions of the problem space as well as the Arrow framework. Visually compelling diagrams of the structure of such a pipeline and the interconnecting parts of the job script will be presented.

Apache Arrow is free and open source software. Full details of its development and implementation are available online and are not the focus of this piece. Only minimal attention will be given to its composition on the poster. Primarily its columnar data format, as well as how *Plasma* is built on top of Arrow.

Instead, the focus will be on how using this library facilitates greater productivity. Positioned at center is a complete infographic diagram of a representative job script that includes all of the interconnected parts of this workflow. The aesthetic will be a minimalist *flat-design* which does include some code references but *is not* verbatim code. The idea being that it can simultaneously offer an anatomy as well as be a navigation tool on the poster and pull all of the elements together.

The supernova forecasting-pipeline described in section 2 will have an account of the necessary context and background information, implementation details of its architecture, and how Arrow was used.

Finally, figures showing the performance characteristics of Arrow/Plasma in use within a typical HPC environment and a comparative analysis showing how use of Plasma scales with multiple concurrent readers.

[6] *Time-Domain Astronomy* (TDA) is a branch of astrophysics that focuses on sources which change over time (be they *transient* or *periodic*). *Supernovae* are only one such example; other transient phenomena include *Gamma-Ray Bursts, microlensing*, and *asteroids*.

[7] A *light-curve* is a time-series representation of the brightness of an astrophysical source, often used to characterize supernovae. Different types of supernovae can be distinguished by the shape of their light-curve.

[8] A *Supernova* (plural: *supernovae*) is the explosive "death" of certain types of stars. Supernovae can out-shine their host galaxy for the life of the event and are an active area of research in modern astrophysics.

[9] *Astronomical Surveys* stand in contrast to "traditional" observing behavior in that it is not typically an individual person or group interested in a particular astrophysical source; but rather, a dedicated (entirely or in part) facility that makes observations of the sky in a regular pattern – usually making both the raw data and finished data products available in an online archive. This pipeline will process candidates coming from the *Large Synoptic Survey Telescope* [2] when it comes online.

REFERENCES

[1] Apache Software Foundation. 2019. *Arrow*. A cross-language development platform for in-memory data. https://arrow.apache.org

[2] M. Jurić, J. Kantor, K. Lim, R. H. Lupton, G. Dubois-Felsmann, T. Jenness, T. S. Axelrod, J. Aleksić, R. A. Allsman, Y. AlSayyad, J. Alt, R. Armstrong, J. Basney, A. C. Becker, J. Becla, S. J. Bickerton, R. Biswas, J. Bosch, D. Boutigny, M. Carrasco Kind, D. R. Ciardi, A. J. Connolly, S. F. Daniel, G. E. Daues, F. Economou, H.-F. Chiang, A. Fausti, M. Fisher-Levine, D. M. Freemon, P. Gee, P. Gris, F. Hernandez, J. Hoblitt, Ž. Ivezić, F. Jammes, D. Jevremović, R. L. Jones, J. Bryce Kalmbach, V. P. Kasliwal, K. S. Krughoff, D. Lang, J. Lurie, N. B. Lust, F. Mullally, L. A. MacArthur, P. Melchior, J. Moeyens, D. L. Nidever, R. Owen, J. K Parejko, J. M. Peterson, D. Petravick, S. R. Pietrowicz, P. A. Price, D. J. Reiss, R. A. Shaw, J. Sick, C. T. Slater, M. A. Strauss, I. S. Sullivan, J. D. Swinbank, S. Van Dyk, V. Vujčić, A. Withers, P. Yoachim, and f. t. LSST Project. 2015. The LSST Data Management System. *ArXiv e-prints* (Dec. 2015). arXiv:astro-ph.IM/1512.07914 https://docushare.lsst.org/docushare/dsweb/Get/LSE-163

[3] Keras Team. 2018. *Keras*. The Python Deep Learning Library. https://keras.io

[4] Ole Tange. 2018. *GNU Parallel 2018*. Ole Tange. https://doi.org/10.5281/zenodo.1146014

[5] Lucas A. Wilson, John M. Fonner, Oscar Esteban, Jason R. Allison, Marshall Lerner, and Harry Kenya. 2017. *Launcher*: A simple tool for executing high throughput computing workloads. *Journal of Open Source Software* (Aug. 2017). https://doi.org/10.21105/joss.00289