Pilot-Streaming: Design Considerations for a Stream Processing Framework for High-Performance Computing

Andre Luckow[1,2], Peter M. Kasson[3], Shantenu Jha[1]

[1]*RADICAL, ECE, Rutgers University, Piscataway,NJ 08854, USA*
[2]*Clemson University, Clemson, SC 29634, USA*
[3]*University of Virginia, Charlottesville, VA 22908, USA*

I. Introduction

Streaming capabilities are becoming increasingly important for scientific applications [1], [2] supporting important needs, such as the ability to act on incoming data and steering. The interoperable use of streaming data sources within HPC environments is a critical capability for an emerging set of applications. Scientific instruments, such as x-ray light sources (e. g., the Advanced Photon Source (APS) and the Advanced Light Source (ALS) [3]), can generate large amounts of high-velocity data in a diverse set of experiments. Coupling this data stream to computational HPC capabilities is an important challenge.

Supporting the processing of high data rates streams executing, e. g., predictions and outlier detection algorithms on it, while running larger models in batch mode on the entire dataset, is a challenging task. The increasing demands lead to a heterogeneous landscape of infrastructures and tools supporting streaming needs on different levels. Batch frameworks, such as Spark [4] have been extended to provide streaming capabilities [5], while different native streaming frameworks, such as Storm [6] and Flink [7] have emerged.

We define a streaming application as an application that processes and acts on real-time data, also referred to as event stream. Different usage modes for stream processing can be observed:

- **Coordination:** Usage of stream processing to connect a data source and data analysis phase. Sometime this includes the pre-processing and transformation of the data before it becomes persistent (e. g. the Hadoop Filesystem (HDFS)) and analyzed (e. g. using a Hadoop processing framework).
- **Realtime Analytics:** This type of application utilizes machine learning on incoming data, e. g. for scoring, classification or outlier detection.
- **Analytics and Model Update:** The applications combine stream processing with other forms of processing, e. g. the continuous update of a machine learning model on historical data and real-time scoring/classification or a simulation.

The complexity of the application increases from the top to bottom. In the last application type streaming, batch and interactive processes utilizing different abstractions and runtime systems need to be combined, algorithms need to be adapted to process windows of data instead of the complete bounded data-set.

There are several challenges that need to be address when developing streaming applications:

- **Infrastructure:** How to efficiently handle data streams (delivery guarantees, low latencies, varying data rate)? How to decouple data producer and consumer? How to store data to allow flexible stream and batch processing (event stream as a log, random access and mutable storage)?
- **Abstractions:** How to decouple application concerns from streaming infrastructures? How can high-level abstraction, such as SQL oder data pipelines be efficiently supported in streaming mode?
- **Applications:** The application itself needs to be capable of utilizing streaming data. Often, the algorithm needs to be adapted to meaningful incorporate incoming data. While batch algorithms assume that they operate on a complete dataset, streaming applications need to operate on a window of data (e. g. a fixed, sliding or session window) [8]. Thus, it is often necessary to balance historical and recent data in machine learning algorithms using e.g. decay factors. Simulations e. g. need an algorithm for including streaming data into their current state.

As alluded in Ref. [1], there is no consensus on software and hardware infrastructure for streaming applications, which increases the barrier for adoption of streaming technology in a broader set of application. Notwithstanding the lack of general consensus, in this White Paper we will explore the usage of the existing Pilot-Abstractions as a unified layer for the development of streaming applications. We further explore a specific application example from the domain of genome sequencing where computational and streaming capabilities are critical for real-time sequencing control.

II. Background and Related Work

The landscape of tools and frameworks for stream processing is manifold (see [9] for survey). The majority of these tools are open source and emerged in the Hadoop ecosystem. In the following, we briefly highlight three main components

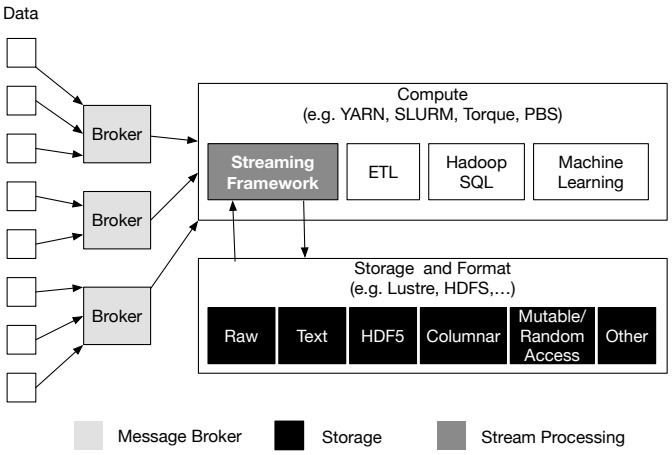

Fig. 1. **Streaming Applications Architecture:** The *message broker* decouples streaming applications from incoming data feeds and enables multiple applications to process the data. The *streaming framework* typically provides a window function abstraction on which a user-defined function is performed. Another important consideration are *data formats*: mutable/random access formats are particular desirable as they simplify the reconciliation of streaming and historical data.

for stream processing (see Figure 1): the message brokering system, the storage and the stream processing engine.

Broker: Message broker are a key component for streaming application. The brokering system decouples data sources and applications; it enables application to observe a consistent event stream of data at its own pace executing complex analytics on that data stream. Kafka [10] is one such distributed message broker optimized for large volume log files containing event streams of data. RabbitMQ and ZeroMQ provide more complex capabilities (e.g., with respect to message routing and delivery guarantees), but are generally less scalable than Kafka. Amazon Kinesis and Google Cloud Pub-Sub are two distinct message brokers offered as "platform as a service" in the cloud.

Storage: Several storage formats optimized for high data velocities and random access read/writes have been proposed. HBase [11] introduced a random access storage engine on top of the Hadoop filesystem. Kudu [12] attempts to improve the HBase design providing a storage system that enables both high-throughput reads as well as mutable datasets.

Processing Engines for Streaming: The landscape of streaming engines is still evolving. The most widely used engines are currently: Storm [6], Flink [7], Spark Streaming [5], Samza [13] and Heron [14]. The different streaming engines differs significantly in the ways they handle data and provide processing guarantees: Native stream engines, such as Storm and Flink, continuously process data as it arrives. Spark Streaming uses micro-batches, i. e., incoming data is partitioned into batches according to a user-defined criteria (e. g. time window). The advantage of micro-batching is that it provides better fault tolerance and exactly-once processing guarantees. Google's Dataflow [8] is another native stream processing engine available in the Google cloud. The engine is based on a rigorous model for stream processing and provides well-defined and rich semantics for windowing and operations. Apache Beam attempts to implement the model proposed by Google Dataflow on top of different open source streaming engines [15].

Higher-Level Abstractions for Streaming: For batch processing various high-level abstractions for data analytics have emerged, e. g., in SQL, dataframes and machine learning libraries. There is ongoing work in providing similar abstractions for streaming data, e. g., extend Spark Dataframes [16] for stream processing. Further, MLlib offers some algorithms for streaming machine learning (e. g. K-Means). In general, streaming applications are required to combine and mix a complex landscape of infrastructures, frameworks and tools. It can be expected, that the design space for abstractions will be further explored and more hybrid approaches will emerge.

Pilot-Abstraction: The Pilot-Abstraction offers a unified approach for data management in conjunction with Pilot-Jobs across complex storage hierarchies comprising of local disks, cloud storage, parallel filesystems, SSD and memory, and allows the efficient management of Pilot-/task-level input data as well as intermediate and output data taking into account data locality. We have explored the applicability of the Pilot-Abstraction [17], [18] to data-intensive applications on HPC and Hadoop environment [19], [20], [21]. The so-called Pilot-Data abstraction supports the management of data in conjunction with Pilot-Jobs and compute tasks. In this white paper, we propose the extension of the Pilot-Abstraction to streaming to enable applications to combine streaming and batch analytics.

III. Pilot-Streaming: Coupling Realtime and Batch Processing

Pilot-Jobs and Pilot-Data [19] provide efficient mechanisms for managing data and compute across different, possibly distributed backends. As shown in Figure 2 the Pilot-Abstraction facilitates the orchestration of Compute-Units and associated datasets, the so called Data-Units across heterogeneous infrastructures. While the original framework was designed for batch-oriented applications, we believe the addition of streaming capabilities will enhance its applicability and suitability of distributed data-intensive applications. We are planning to add streaming capabilities to this framework: *Pilot-Streaming* - a framework for HPC stream processing.

Pilot-Data is an extensible framework allowing the simple addition of new data sources and processing frameworks. In particular we plan the following two extensions:

- **Stream Data Source Access:** Make streaming source (e. g. Kafka topics) accessible for Compute-Units and enable the micro-batch processing of discretized chunks of incoming data. The Kafka adaptor will be implemented using the Kafka Python API.
- **Extensibility and Interoperability:** An interoperable access to streaming platforms (e. g. Spark Streaming and Flink) enables applications to utilize the different capabilities of these frameworks in a unified way. By generalizing the window function abstraction into a higher-level abstrac-

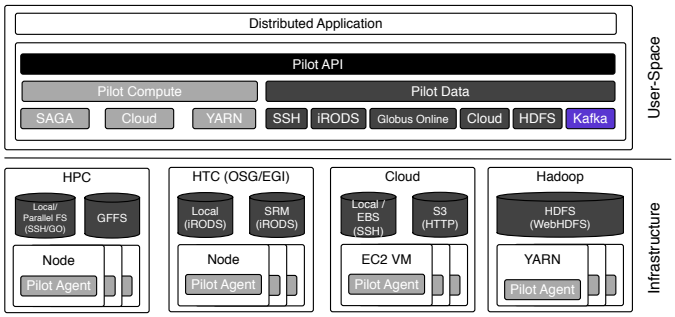

Fig. 2. **Pilot-Job Extensions for Streaming:** The Kafka adaptor for Pilot-Data enables application to subscribe to data streams and execute Compute-Units on discretized micro-batches of incoming data.

tion, applications can be expressed independently of the underlying execution engine.

IV. Application Exemplar

Real-time sequencing control: Some current-generation nanopore sequencing devices include the capability for selective sequencing: either continuing to read a DNA strand in one of the nanopore channels or ejecting it to enable another DNA molecule to bind [22]. This ejection decision can be made based on the individual-channel conductance trace. Such a capability enables optimization of sequencing "coverage": rejecting unwanted sequences to enrich a rare sample from a more frequent background, equalizing coverage across a sample, or other more complex optimization strategies.

Taking advantage of this however, requires two sets of streaming computations: modeling the channel conductance trace as a nucleic acid sequence and, if the statistical model for sampling is not fixed, updating this model in real time based on many parallel channel inputs (512 per sensor chip in current implementations, but multiple sensor chips may be used). Both of these computations carry substantial latency requirements: DNA transits the nanopore channel at 500 bases per second, while the sampling model should be updated in approximately real time but permits a somewhat greater latency. This sampling model might be unique to the physical sample being processed and thus specific to the sensor(s) at one location or might be global in scope.

Furthermore, both scenarios require significant computational processing in order to investigate, either via models or simulations with adequate atomistic precision and accuracy. Thus, the desired computational platform is a high-performance resource with both streaming and batch capabilities.

V. Conclusion

The number of applications requiring stream processing capabilities is increasing. The landscape of tool and frameworks for message brokering, data storage, processing and analytics is manifold. The Pilot-Abstraction can serve an important gap in unifying streaming and batch processing for HPC applications. It enables applications e. g. to couple streaming data with task parallel applications (e. g. for data processing and machine learning).

References

[1] Stream 2015 final report. http://streamingsystems.org/finalreport.pdf, 2015.

[2] Geoffrey Fox, Shantenu Jha, and Lavanya Ramakrishnan. Scalable hpc workflow infrastructure for steering scientific instruments and streaming applications. Technical report, 2015.

[3] US Department of Energy. X-Ray Light Sources. http://science.energy.gov/bes/suf/user-facilities/x-ray-light-sources/, 2016.

[4] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[5] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, and Ion Stoica. Discretized streams: Fault-tolerant streaming computation at scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA, 2013. ACM.

[6] Nathan Marz. Storm: Distributed and fault-tolerant realtime computation. http://cloud.berkeley.edu/data/storm-berkeley.pdf, 2011.

[7] Apache Flink. https://flink.apache.org/, 2015.

[8] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 8:1792–1803, 2015.

[9] Supun Kamburugamuve and Geoffrey Fox. Survey of distributed stream processing. Technical report, Indiana University, Bloomington, IN, USA, 2016.

[10] Guozhang Wang, Joel Koshy, Sriram Subramanian, Kartik Paramasivam, Mammad Zadeh, Neha Narkhede, Jun Rao, Jay Kreps, and Joe Stein. Building a replicated logging system with apache kafka. *PVLDB*, 8(12):1654–1665, 2015.

[11] HBase. http://hadoop.apache.org/hbase/.

[12] Todd Lipcon, David Alves, Dan Burkert, Jean-Daniel Cryans, Adar Dembo, Mike Percy, Silvius Rus, Dave Wang, Matteo Bertozzi, Colin Patrick McCabe, and Andrew Wang. Kudu: Storage for fast analytics on fast data. 2015.

[13] Martin Kleppmann and Jay Kreps. Kafka, Samza and the Unix philosophy of distributed data. *IEEE Data Engineering Bulletin*, December 2015. Journal Article.

[14] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter heron: Stream processing at scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 239–250, New York, NY, USA, 2015. ACM.

[15] Apache beam proposal. https://wiki.apache.org/incubator/BeamProposal, 2016.

[16] Michael Armbrust, Reynold S. Xin, Cheng Lian, Yin Huai, Davies Liu, Joseph K. Bradley, Xiangrui Meng, Tomer Kaftan, Michael J. Franklin, Ali Ghodsi, and Matei Zaharia. Spark sql: Relational data processing in spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1383–1394, New York, NY, USA, 2015. ACM.

[17] Andre Luckow, Mark Santcroos, Andre Merzky, Ole Weidner, Pradeep Mantha, and Shantenu Jha. P*: A model of pilot-abstractions. *2012 IEEE 8th International Conference on E-Science*, pages 1–10, 2012. http://doi.org/10.1109/eScience.2012.6404423.

[18] Andre Luckow, Lukas Lacinski, and Shantenu Jha. SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems. In *The 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 135–144, 2010.

[19] Andre Luckow, Mark Santcroos, Ashley Zebrowski, and Shantenu Jha. Pilot-data: An abstraction for distributed data. *Journal of Parallel and Distributed Computing*, 2014.

[20] André Luckow, Pradeep Kumar Mantha, and Shantenu Jha. Pilot-abstraction: A valid abstraction for data-intensive applications on hpc, hadoop and cloud infrastructures? *CoRR*, abs/1501.05041, 2015.

[21] A. Luckow, I. Paraskevakos, G. Chantzialexiou, and S. Jha. Hadoop on HPC: Integrating Hadoop and Pilot-based Dynamic Resource Management. *IEEE International Workshop on High-Performance Big Data Computing in conjunction with The 30th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2016)*, 2016.

[22] Matthew Loose, Sunir Malla, and Michael Stout. Real time selective sequencing using nanopore technology. *bioRxiv*, 2016. http://biorxiv.org/content/early/2016/02/03/038760.