STREAM2015 Final Report

Geoffrey Fox, Indiana University
Shantenu Jha, Rutgers University
Lavanya Ramakrishnan, Lawrence Berkeley National Laboratory

http://streamingsystems.org/stream2015.html. Indianapolis October 27-28 2015

See this web site for more information.

Contents

1 Executive Summary

The analysis of data streaming from on-line instruments, large scale simulations, and distributed sensors now enables near real-time steering and control of complex systems such as scientific experiments,

transportation systems, and urban environments. There has been an explosion of new research and technologies for stream analytics arising from the academic and private sectors, but there has been no corresponding effort in either documenting the critical research opportunities or building a community that can create and foster productive collaborations. To ameliorate this situation, we are organizing two workshops - STREAM2015 and STREAM2016. This report describes the discussions, outcomes and conclusions from STREAM2015, the first of these workshops.

We define a *stream* to be a possibly unbounded sequence of events. Successive events may or may not be correlated and each event may optionally include a timestamp. Exemplars of streams include time-series data generated by instruments, experiments, simulations, or commercial applications including social media posts. Steering is defined as the ability to dynamically change the progression of a computational process such as a large-scale simulation via an external computational process. Steering, which is inevitably real-time, might include changing progress of simulations, or realigning experimental sensors, or control of autonomous vehicles. Streaming and steering often occur together. An example could be for an exascale simulations where it is impractical to store every timestep and the data must be reduced, resulting in streams which may constitute the final results from the simulation in a manner similar to the way we use data from an instrument in a massive physics experiment.

The goal of the STREAM 2015 interdisciplinary workshop was to identify application and technology communities working in the broad areas of streaming, steering and their intersection, and to clarify challenges that they face. We solicited white papers on related topics from the community. The white papers submitted covered a mixture of applications and technologies. The final workshop program included 43 attendees and 29 talks that included 17 accepted papers and other invited talks, four breakout sections and three plenary discussions. We also identified research challenges and future steps for community engagement. The discussions from the workshop are captured as topic areas covered in this report's sections. The workshop identified four important research directions driven by current and future application requirements.

Programming Models: There is tremendous diversity and churn in stream analytics programming models and systems and no common understanding of application requirements, making them impractical to compare directly. There is a need to research and develop programming models and computational paradigms that can raise the level of abstraction to simplify the programming tasks for end-users.

Algorithms: Challenging stream analytics problems arise from applications where the complexity and scale of these unbounded events are such that they cannot be analyzed at the rate they are produced with traditional batch algorithms. The workshop attendees suggested a variety of compelling research topics including adaptive sampling, online clustering, sketching and approximation that trade space and time complexity for accuracy.

Steering and Human in the Loop: In many cases it is essential that a human guides the automatic responses to event streams. Research ranging over systems, visualization and human computer interaction is needed to integrate human and computing knowledge in decision processes.

Benchmarks and Application Collections: We require an understanding of the true complexity and breadth of application needs to make true progress on algorithm, programming models and software systems for stream data analytics and steering. Similar to "grand challenges" in domain sciences, the development of a series of "representative" streaming and steering examples will be of enormous value. These should include representative data sets at scale and individually clean simple mini-applications.

The workshop participants felt that the diverse nature of the nascent community was well represented by the diverse group of attendees and the topics presented, but that the topic is in need of further attention because of its growing importance. There is a strong support for establishing and enhancing this community and supporting and expanding the role of http://streamingsystems.org.

2 Introduction

STREAM2015 grew out of NSF ACI's Dear Colleague Letter (DCL) [DCL15053] to the community to identify the gaps, requirements and challenges of future production cyberinfrastructure beyond traditional HPC. The NSF has a long track record in production cyberinfrastructure to support HPC applications, but arguably a less consistent record in deploying and supporting infrastructure for the rich space of distributed applications. Given the trends in data, instrumentation and scientific computing, an area of particular interest was the investigation of the landscape -- systems, middleware and applications --- at the integration point of HPC, data-intensive computing and real-time computing as required by large scientific instruments and projects: streaming and steering lie at the heart of the overlap. Given DOE's mission, it was not a coincidence that these topics were of interest and relevance to ASCR at DOE. Thus, was born the coordinated NSF and DOE efforts to organize STREAM 2015 and STREAM 2016.

The analysis of data streaming from on-line instruments, large scale simulations and distributed sensors now enables near real-time steering and control of complex systems such as scientific experiments, transportation systems and urban environments. There has been an explosion of new research and technology for stream analytics coming from the research and private sectors, but many of the activities are disconnected. This report describes the outcomes and conclusions from the first of two workshops, STREAM2015 and STREAM2016, that brings together a wide range of practitioners and stakeholders to cover the full breadth and potential of this topic. Significant effort was made to cover government, industry and academic activities.

We solicited white papers on related topics from the community for STREAM2015. The white papers submitted covered a mixture of applications and technologies. 17 white papers were accepted and are listed in sec. 8.3 with links to posted versions. The white paper authors and several additional participants from academia and industry were invited to the STREAM2015 workshop that covered a rich selection of applications, algorithms and software systems. The final workshop program included 29 talks, four breakout sections and three plenary discussions with 43 attendees.

This report is arranged into three major sections: State of the art; Next steps and research directions; Building and sustaining a streaming system community. These capture the workshop discussions and presentations/white papers and are followed by a summary with appendices listing participants, presentations (labelled [Pnn]), white papers (labelled [WPnn]) and citations (labelled [nn]).

We provided working definitions for streaming and steering; it is thus important to clarify stream processing. Stream processing is characterized by (i) operations on sections of data much shorter than the overall length of the stream, (ii) sections are processed in a (partially) ordered way, and (ii) the streaming process keeps a

(potentially distributed) state between processing of data sections and there is no possibility of analyzing a previously processed data section.

3 State of Art

Streams can be characterized by concurrent sequences of events, which might be associated with timestamps as in social media posts or time series generated from one or more instruments, experiments, simulations, or commercial applications. Steering might include steering simulations or realigning experimental sensors, or control of autonomous vehicles. Both of these cases were covered in workshop and presentations on current state of the art set the scene for futures discussion; in particular identification of research challenges and future steps for community engagement covered in secs. 4 and 5.

Processing of streaming data is being addressed by many communities including industry, many branches of academic research and national laboratories [Survey16]. The goal of the STREAM 2015 interdisciplinary workshop was to identify application and technology communities working in this area and to clarify challenges that they face. The workshop focused on application features and requirements covered in sec 3.1 as well as hardware and software needed to support them covered in sec. 3.2. Sec 3.3 covers the special topic of HPC-Big Data convergence.

3.1 Exemplar applications: Characteristics of Applications, Industry-Science differences.

3.1.1 Application Categories and Exemplars

Streaming/Steering Application Class		Details and Examples	Features	Talks, WPs
1	**Industrial Internet of Things, Cyberphysical Systems, DDDAS, Control**	Software Defined Machines, Smart buildings, transportation, Electrical Grid, Environmental and seismic sensors, Robotics, Autonomous vehicles, Drones	Real-time response often needed; data varies from large to small events	WP2,8,10,14 ,17 P2,10,11,14, 16,19,20,27
2	**Internet of People: wearables**	Smart watches, bands, health, glasses, telemedicine	Small independent events	P1
3	**Social media, Twitter, cell phones, blogs, e-commerce and financial transactions**	Study of information flow, online algorithms, outliers, graph analytics	Sophisticated analytics across many events; text and numerical data	P1,3,12,13, 18
4	**Satellite and airborne monitors, National Security: Justice, Military**	Surveillance, remote sensing, Missile defense, Anti-submarine, Naval tactical cloud [NavyCloud14]	Often large volumes of data and sophisticated image analysis	WP13 P28,29
5	**Astronomy, Light and Neutron Sources, TEM, Instruments like LHC, Sequencers**	Scientific Data Analysis in real time or batch from "large"	Real-time or sometimes batch, or	WP1,5,6,9, 12 P4,5,6,15,26

		sources. LSST, DES, SKA in astronomy	even both. large complex events	
6	**Data Assimilation**	Integrate typically distributed data into simulations to enhance quality.	Link large scale parallel simulations with time dependent data. Sensitivity to latency.	WP3 P21
7	**Analysis of Simulation Results**	Climate, Fusion, Molecular Dynamics, Materials. Typically local or in-situ data. HPC Big Data Convergence	Increasing bottleneck as simulations scale in size.	WP1,4,7,11 P5,7,9,23,24
8	**Steering and Control**	Avionics. Control of simulations or Experiments. Network monitoring. Data could be local or distributed	Variety of scenarios with similarities to robotics	WP7,15 P5,9,17,25

Table 1: Eight Streaming and/or Steering Application Classes

Applications cover a wide range. As one example, for exascale simulations where it is impractical to store every timestep, it may be necessary to do in-situ analysis of critical indicators from each step which can then be used to modify simulation parameters or context on the fly. The resulting streams from the in-situ analysis may constitute the final results from the simulation in a manner similar to the way we use data from an instrument in a massive physics experiment. As another example, smart city sensors generate streams of data that capture the health and state of an urban environment. Analyses of these data streams can guide city planners and emergency responders.

We surveyed the field and identified application classes including Internet of People, Social media, financial transactions, Industrial Internet of Things, Dynamic Data Driven Applications Systems (DDDAS) [DDDAS15], Cyber-physical Systems, Satellite and airborne monitors, National Security, Astronomy, Data Assimilation, Analysis of Simulation Results, Steering and Control and Instruments like the Large Hadron Collider(LHC), high-energy x-ray sources, and sequencers. Although, streaming applications differ widely e.g., in the type of science they support (simulations versus experiments), a large fraction of them can be understood using a few common characteristics, e.g, in whether they are distributed or not, size of streamed data unit, whether they have real-time constraints etc. This allows introduction of application classes summarized in table 1 which also lists presentations and white papers discussing each class. Note that these contributions are sometimes technology and sometimes application focussed; the technology contributions often give use cases that are used for listing them in table 1.

There were several applications brought up during the workshop. Some examples include microbiome and metagenomics which were not in any of the formal contributions but likely to grow in need for streaming processing.

- UAV monitoring for oil and gas exploration use case noted the complexity of the analysis of radar images; harder than related optical images seen in surveillance applications. Size is another issue; The size of radar images are several times bigger than optical images because they contain phase and polarization information in addition to intensity value.
- The LHC physics use case is also one when current batch processing is moving to a more dynamic model which is essentially streaming. This allows easier use of opportunistic resources such as

Amazon and NERSC available cycles and to avoid the complex replica schemes in data grids. Similarly, photon science experiment workflows are becoming increasingly automated, and are coupling detector data sources to high-performance computers for near-real-time analysis [P4].

- The DDDAS Avionics application [P25] in the steering and control category brought up some important issues including the role of bad data and its recognition detection especially by automated systems used today in aircraft. We often need to support sophisticated application sitting on top of streaming such as simulations to give feedback on how to respond to data. Cyber-physical system applications like the Smart Grid, IIOT (Industrial Internet of Things) or IOB (Internet of Buildings) are growing rapidly -- interoperability, anonymization, and need to integrate diverse data are important. The trade-off between anonymization and utility in the Array of Things was highlighted.
- TEM produces streams of connected images with human in the loop with the need to preserve message (image) order. Network monitoring can be viewed as streaming data with network routing becoming a dataflow graph.
- The Square Kilometer Array telescope SKA -- shifts mechanical engineering (100 square meter mirror) to computing (lots of one square meter mirrors). Here “steering” will be used to update in real time the corrections for atmospheric and mechanical effects on the observed data; and more generalised stream processing will be used because it is too expensive to contemplate keeping all of the observed data and therefore it must be processed into reduced data production.

The discussion noted some differences between industry and research applications. One was event size -- these are often large in research use cases. Another was approach to getting needed performance; industry might “just” add more nodes to improve performance; this is typically considered too costly for research applications. Finally SQL important in many industry deployments but its broad value in science is not so clear.

3.1.2 Application Characteristics

Several characteristics were highlighted that could be used to develop a more detailed classification than table 1 and in deriving requirement for processing system. These include the size of events; the use of control or steering; the connection between events such as their ordering and stateful event processing; use of human in the loop for feedback; universality of interfaces; adaptive pipelines needed in research today; need to identify the important information “rapidly”; access control; adaptive flow control; the challenge of real-time data assimilation; complexity of data in individual events; need for fault tolerance; provenance especially in adaptive applications where data result of previous workflow; accuracy and use of sampled data; error recovery - problematic data will choke when passed through the algorithm again; what are data structures and appropriate storage matching hardware and application needs.

Synchronization might be needed to produce for example a coherent state (see [P13] for a clustering example) but this is hard with inherently distribution asynchronous data sources and processing. It is useful to understand the latency that can be tolerated between data collection and processing data. Coupled streams, multiple streams and interaction of streams with distributed data are important in some cases. The response time is important for UAV applications and robots; it is not so critical for e-commerce transaction streams. The needed data longevity is unclear-- some fields assume data is kept all the time but where do you store it and what is the cost. We need to characterize the value of data - how much data are we willing to lose during the processing? Application processing can be characterized by the complexity of processing, the possible need for a quick turnaround, offline or online mode etc. We can distinguish between two types of streaming applications - firstly a closed process that runs for long-time, and secondly adaptive computation with lot of human involvement. The latter is seen in processing of experimental data with

adjustments to data-gathering equipment based on analysis of results. Some science applications require high-speed, fully automatic and complex adaptive processing. Here it is often not cost-effective to make an automated pipeline for each case.

Within large scale simulations, there are event streaming techniques which will be essential, as every snapshot can't be saved. This is category 7 in table 1. We also noted that steering (category 8 in table 1) is pervasive; in particular sensors sometimes need to be steered

There is a need to respond to variable rates or load changes such as between peak and non-peak hours. This requires an elastic system capable of dynamic scalability based on the changing rates. This creates interesting programming paradigms involving unbounded data with sliding windows with the need to detect changes between windows.

In most cases, the data is distributed, which affects synchronization, parallelism and algorithms. One needs to characterize workflows better; for example where are computations done - are they close to the source of the data or does data stream to the cloud? This is exemplified by comparison of Akamai edge vs Google server farms. There is a well known discussion of three processing locations: sensor (source), fog (computing near source) or cloud (back-end). Does it matter which cloud one chooses as location will affect turnaround? Does the sensor's connectivity facilitate certain processing modes or does it constrain processing modes - the answer could be time dependent. Does the processing (cloud) need to be self distributed? One needs to address the case with no cloud connection (at a particular time). What aspects of processing are sensitive to hierarchical (sensor-fog-cloud) programming model; does query processing get decomposed at all levels? What parts of system need to talk to all services/nodes? How much can be done at each level of hierarchy?.

Projects such as large telescopes or accelerators with long timelines need to consider timeline of evolving technologies for streaming data as industry is driving rapid change. One can identify needed functions and components and if one programs to this high level model, it should be easier to incorporate underlying technology change.

The availability of test data - where security and privacy concerns prevents wider release - needs to be addressed as in sec. 4.3. We need better anonymization techniques or privacy policy changes

3.2 Current solutions -- Industry, Apache, Domain-Specific

3.2.1 Particular Solutions

Three impressive solutions from industry -- Amazon [P1], Google [P3] and Microsoft [P18] -- were described. Although operating at large scale their uses like e-commerce and logging transactions have important differences from the research applications discussed in sec. 3.1. Amazon has Kinesis streaming software, Google Cloud Dataflow supporting general Big Data analytics while Microsoft has introduced Trill-based Azure Analytics as a high-performance in-memory incremental analytics engine. This first class (commercial) of streaming software infrastructure can be compared with three others presented at the workshop.

The second set consists of solutions [Survey16] built around open source Apache solutions like Samza ([Samza15], [WP16] and [P22]) or Storm ([Storm15], [WP17] and [P19]). One could consider latter as the previous generation of Industry solutions as Samza came from LinkedIn and Storm from Twitter.

The third set consisted of current domain-specific solutions used today in different observational science areas. As described in sec. 3.1, we heard about technology for Astronomy, Light Sources, LHC, Smart Grid, Array of Things, IIOT, UAV, Avionics and simulations. Early academic work includes Aurora [Cherniack03], Borealis [Abadi05], StreamIt [Thies02] and SPADE [Gedik08].
As the fourth category we can put research software designed for general streaming applications. These included SEEP [Seep15], StreamMapreduce ([Streammine15], [WP2] and [P20]), Workflow ([WP3] and [P21]), cloud video streaming ([WP13] and [P28]), HPC (exascale) technology ([WP11] and [P23]) and an academic approach Neptune [P16] [Neptune16] with a similar architecture to the Apache solutions. In contrast StreamMapReduce [P20] offers MapReduce with a built-in streaming engine with fault tolerance and elasticity and supporting SQL, CQL and K3. Time Windows permit the definition of a dataset to operate on in StreamMapReduce. SamzaSQL [P22] builds on Apache Kafka [Kafka15], Samza and Calcite with monitoring use cases from Linkedin. IoTCloud [P19] adds parallel processing and low latency to Storm so as to support the use of back-end clouds controlling robots. MPI is not designed for streaming and a library to support streaming is added [P23] with special benchmarks and a comparison between MPI I/O and Streaming I/O.

3.2.2 Technology Challenges and Features

Integration of the many technologies to build “end-to-end” streaming infrastructure is a basic challenge as is the lack of consensus as to appropriate hardware and software infrastructure. Other issues identified included securing data; dynamic provisioning of cloud resources to reduce cost; connecting different programming models in streaming environment; supporting online algorithms; understanding concept of time and space with respect to streaming data; use of both legacy applications and new online algorithms together in a scaling fashion; use of and identification of reusable components; need for data management to be scalable and dynamic; impact of non-volatile memory, and GPU’s, which currently are outside streaming frameworks.

We discussed state of the open source software and industry participants noted Storm (and Spark [Spark15] and Flink [Flink15]) was really obsolete and a new hardened version is needed. Amazon Kinesis can drive Apache Storm but latter does not scale well enough to use in large production systems like Amazon Kinesis. Neptune has higher performance than Storm in analysis reported in [P16] [Neptune16]. Apache Flink, building on EU Stratosphere, is a current favorite cloud programming environment and supported in Google Cloud Dataflow. For industry, interoperability is a difficulty for linking many manufacturers and data sources.

The hierarchical sensor-fog-cloud model is not supported in today’s systems which focus on the backend (cloud) solution.

3.3 Connections - Streaming + HPC convergence. Role of workflow.

The issues at the intersection of streaming and HPC were discussed in several white papers [WP1, WP3, WP4, WP5, WP6, WP11, WP12] and associated presentations [P4, P5, P7, P9, P17, P21, P23, P24]. One reason to explore this intersection is that commercial solutions are not converging with the needs of science big data. Another is that resources available for large scale research applications are typified by NSF resources configured as HPC clusters.

One HPC-Streaming intersection is the analysis of visualization data streamed from HPC simulations. As I/O is limited, this analysis is usually done in situ on the machine hosting the simulation. The role of new

exascale runtime for streaming should be investigated in both these cases. The new exascale-motivated approaches are more dynamic and could be more effective than traditional MPI for streaming. We need to understand the interaction between big and streaming data and the connection between in-situ data processing and big machines.

There is a need for a deeper coordination between the HPC and streaming communities. There are similarities between workflow and stream communities with for example Swift from Chicago supporting MapReduce implemented with MPI. We could learn from workflow and in particular from the difficulties in promoting interoperable workflow solutions.

Particular issues discussed included the role of storage and networks in streaming; the impact of RDMA networks on streaming; the impact of SDN on streaming; the I/O performance and architecture needed on HPC machines to support streaming; the need to benchmark HPC vs Cloud solutions with architectural differences and optimization differences; connection with hardware vendors.

There is perhaps a disconnect between HPC and BigData software stacks in the areas of performance and architectural components e.g. HPC might use Slurm and Big Data (streaming) Yarn or Mesos for cluster management/scheduling. We need to engage with XSEDE and XSEDE ECSS to understand better how to do streaming on big NSF resources and look at usability and performance issues.

4 Next Steps and Research Directions

The workshop Identified several important research directions driven by current and future application requirements. These are covered in following sub-sections.

4.1 New Algorithms

Some of the most challenging stream analytics problems arise from applications where the complexity and scale of unbounded streams are such that they cannot be analyzed at the rate they are produced with traditional batch algorithms. Consequently we need different algorithms that can reduce the complexity of the analysis and there is an active computer science research here but limited practical experience. The workshop attendees suggested a variety of compelling research topics including adaptive sampling, online clustering [Guha13] and sketching.

Sketch algorithms provide approximate results for queries performed over voluminous datasets. This is achieved by using data structures that serve as a surrogate (or a sketch) for the actual dataset, and then using this data structure to evaluate queries. More importantly, the space-complexity or memory footprint of this data structure is independent of the underlying voluminous dataset. Examples of such algorithms include the well-known Bloom Filter that can be used to evaluate set memberships. The Bloom Filter may produce false positives on set memberships, but will never produce a false negative [Bloom70]. The HyperLogLog algorithm is a probabilistic estimator for approximating the cardinality of a dataset [Flajolet07]. The Count-Min algorithm effectively approximates the frequencies of elements within a dataset that has repeated elements [Cormode12]. In essence the space complexity (size of sketch) trade-off becomes a time-complexity versus accuracy trade-off. Frequent itemset mining over data streams [Freq04][Freq05][Freq06], stream classification [Classify00] and streaming clustering algorithms [Cluster02][Cluster03] are well known but needs to be applied in massively parallel settings.

In workshop presentations, [WP1] [P5] [P7] described streaming Principal Component algorithms PCA and the identification of “heavy hitters” (mass concentrations) corresponding to galaxy halos in astrophysical simulations. [P13] described parallel online clustering algorithms to find Twitter memes (information concentrations) using Apache Storm environment. [Storm15] [WP10] [P11] described baseline methods for electrical grid data understanding.

The NRC Frontiers in Massive Data Analysis report identifies 7 “computational giants” which are essentially algorithmic challenges. These seven are Basic Statistics; Generalized N-Body Problems; Graph-Theoretic Computations; Linear Algebraic Computations; Optimizations; Integration and Alignment Problems. They remark that two of the most pervasive strategies for achieving computational efficiency are sampling and parallel/distributed computing. They back this up with a full chapter on “Sampling and Massive Data”. Further they identify 4 common themes.

1) State-of-the-art algorithms exist that can provide accelerations of major practical importance, by significantly changing the runtime order, for example, from $O(N^2)$ to O(N log N).
2) High dimensionality in the number of variables is a persistent challenge to obtaining computational efficiency, and this demands ongoing research effort.
3) The “non-default” settings (default is sequential in memory algorithms)—streaming, disk-based, distributed, multi-threaded—are quite important, yet mostly under-explored in terms of research effort. More broadly they note below that many key ideas have been proven in principle but not yet applied broadly in practice. This includes for example the O(N logN) algorithms mentioned in first theme.
4) Most of the best fast algorithms described in the computational giant section have only been demonstrated in research implementations. More work is required to create robust and reliable software before these algorithms can be used widely in practice.

Sampling was discussed in the meeting including: sampling improving performance; the sketch concept which is a smaller but representative dataset; the general issues of approximation and accuracy and the trade-off between accuracy and time complexity; socially coupled systems have intrinsic inaccuracy; where, when and what to sample is a challenge; error bounds with sampling; need for users to understand algorithms with approximations and be able to make good choices. Incremental accuracy was discussed with initial sample results being improved as the data increases in size.

Streaming and online algorithms clearly are very promising and in some cases essential to satisfy compute/time complexity constraints. These algorithms are characterized by only using the data once whereas typical iterative batch algorithms go over the data many times. Sampling can go further and only use a subset of the data once (or in a batch sampling many times). The concept of training is not clear in the sampling class of algorithm. Note that streaming algorithms are a distinct idea from real-time algorithms.

In accord with NRC report, the workshop identified need for scalable parallel algorithms covering multicore and parallel computing. Information visualization and dimension reduction (illustrated by PCA presentation [P5]) was also emphasized and covered by NRC. Dimension reduction is linear (e.g. PCA) or nonlinear (e.g. MDS) and comparison of these approaches is needed for streaming data.

Classical batch algorithms such as probabilistic Models, classifiers, Bayesian, Markov, Classical EnKF (Ensemble Kalman Filtering in Data Assimilation), Umbrella Sampling, Importance Sampling, Monte Carlo, and other machine learning algorithms should be applied to streaming scenarios. Is there a streaming version of every batch algorithm? When is it hard or impossible to generate the streaming version? What are

the typical performance and accuracy comparisons of streaming versus batch algorithms. Correspondingly is there a batch version of every streaming algorithm? How much processing should be done online and how much offline (batch)?

We need to research steering scenarios involving active learning: for example a set of molecular dynamics simulations that drive towards rare events. Another example is machine learning classification algorithms that can take streams to improve (adaptively) their accuracy.

4.2 Programming and Runtime Model; Languages

White papers discussing this area include [WP2, WP4, WP7, WP11].

Current practice in programming models and runtime (both must be discussed) spans commercial solutions, open source (Apache) and domain-specific solutions. Additionally, there is tremendous diversity and rapid evolution in stream analytics programming models and systems. Furthermore there is no common understanding of application requirements and programming models, making them impractical to compare authoritatively. The workshop exposed the fact that the major science and commercial applications have significant differences. There is a need to identify best practices and computational paradigms that can raise the level of abstraction to simplify the programming tasks for end-users, removing the requirement of expertise in distributed computing, ad-hoc analytics and integrating diverse software packages. We expect that a few major software and hardware architecture will emerge covering the range of applications and that one will then need to design and build sustainable ecosystems including core middleware and libraries. Note sec. 4.4 discusses producing community systems software and sec. 4.5 the hardware needed to support streaming systems.

We heard about some particular research systems. StreamMapReduce [WP2] [P20] allows event stream processing to work with MapReduce.and extends fine-grained event driven execution models for large-data volumes. High-Performance Implementation of streaming models in MPI was presented in [WP11] [P23]. Neptune [P16] [Neptune16] and Twitter Heron [Heron15] addresses performance problems [WP17] in Apache Storm -- in particular those associated with Apache Kafka [Kafka15].

We heard from three impressive solutions from Industry -- Amazon, Google, Microsoft. It was interesting to see the transition in Google from MapReduce to Cloud DataFlow via FlumeJava and MillWheel. It would interesting to evaluate these systems and compare their approaches to workflow and streaming which are integrated.

Some research issues uncovered included:need for programming abstractions and runtime support for adaptive and steered applications in large scale-simulation science; the concept of time is needed in the abstraction/programming model which is addressing time series processing; the performance of two big data languages -- Python and Java -- could be significantly improved if their compilers and runtime were approached in a way that is well understood for “simulation” languages C++ and Java; more understanding is needed of the Google concepts of bounded or unbounded and the relationship to batch and streaming programming;

What are the (acceptable) costs of layers of abstraction? Practical programmers tend think in terms of conventional abstractions which are complex to map to streaming and dataflow concepts, but nevertheless there may well be benefit in doing so if it opens the field to a much broader range of users. A relevant

example from the HPC field is the PGAS programming model. A majority of practical programmers tend to think in terms of the von Neumann computing model where a unified memory is updated as the computation proceeds. This model is however very difficult to scale up, leading to the introduction of message passing models in HPC which can be efficient but are in turn difficult to program. The PGAS programming model for HPC makes a minimal change to the von Neumann model by partitioning the memory into an area which is cheap to access (mapped to local memory) and the remainder that is globally shared. This then presents a simple high-level abstraction to users which is mapped to message passing by the run-time system, introducing some efficiency loss but being much more friendly to the practical or even casual programmer.

Similarly, there are other areas that provide potentially interesting and relevant approaches, e.g., SQL query optimization [WP16] hasn't been fully exploited for distributed systems although this has been studied in the context of data streams [SQL02] [SQL04]; actor models -- natural for events -- should be studied; maybe there are optimized Java virtual machines for streaming?; maybe there are optimized openStack style virtual machines for streaming?

Another reason why processing streams is difficult compared to traditional batch jobs is the scheduling issue. Optimal stream scheduling where you also try to maximize a performance measure such as latency or throughput is NP-Hard. There are however possible stochastic solutions to the problem and additionally heuristics based on previous runs can be used to efficiently find good approximate solutions [AuroraSched03]. Further research on this could be of benefit to many streaming applications. There is also the trade-off between latency and throughput in streaming processing settings. Further research is needed to understand this and other stream optimizations [Hirzel14] better.

We discussed the data model required for streaming; issues include efficient parallel data management ; are Spark RDD, AsterixDB and Tachyon important? Can one adapt ideas from adaptive meshing to handle key partitioning and sharding; is MPI-IO. and/or lessons from it useful?; do we only ingest into edge elements?

Domain specific Languages are a popular feature in many parallel computing areas. They do not seem to have been explored deeply for streaming systems. In fact industry seems to use API's (between services) and one should evaluate this approach. One needs more abstractions for streaming systems and good implementations that lower the programming burden.

An interesting abstraction is map-streaming [Ogre14] which is a high level description of systems like Apache Storm that describes parts of the system but not all; for example it describes the pub-sub aspects of Storm but misses its dataflow aspects, and additional need to support full parallel computation in some applications such as SLAM in [WP17]. We also need to add the needed operators or primitives and algorithms. The relation of these abstractions to workflow and its abstractions needs study. Perhaps there is an algebra for distributed streaming systems [Soule10, Soule12]. We do not expect a single solution but rather a family of solutions addressing different application classes -- such as those given in table 1. Further we emphasis that this is all work in progress -- much research, benchmarking and experimental systems building will be needed.

The performance cost of using virtual machines -- especially those managed with systems like OpenStack -- needs more study. The overheads are going down but are still unnecessarily high -- partly due to software drivers and partly to the overhead of managing sharing at core level.

Streaming will always give rise to data movement, which needs to be integrated with processing efficiently. We certainly need data movement in parallel and that has often been hard to express and implement when linking parallelism with dataflow. Google's Dataflow name captures the data movement idea. GridGain (a commercial system built on Apache Ignite) captures the further idea that data doesn't hit disk unless necessary.

There is no consensus as to data placement and programming models. Traditional science computing models are based on batch executables but that's not appropriate for always-on event streams. Event and stream queries need novel data management and computing infrastructure, which needs to match the application requirements.

We need ability to dynamically convert streams into scientific and analytical data types within workflows and make automated processing of such data possible at the rates that the data is being produced.

The workshop asked about the possible use of software designed for exascale runtimes in streaming systems. For example, exascale adds support of dynamic inhomogeneous threads which could be very useful in streaming.

4.3 Benchmarks and Application Collections and Scenarios

We require an understanding of the true complexity and breadth of application requirements to make substantive progress on algorithm, programming models and software systems for stream data analytics and steering. Similar to "grand challenges" in domain sciences, the development of a series of representative streaming and steering examples will be of enormous value. These should include representative data sets at scale. Linear road [Bench04] is an early benchmark developed around Aurora streaming system focusing on one application. There may be a need for benchmark suites like TPC or HPCC that foster collaboration between industry and the research community but we need benchmarks that cover the full complexity of the distributed streaming area. There is a clear need to investigate the performance of streaming applications and software on clouds and HPC systems. This need is covered in this subsection for the applications and in the next subsection for software.

Initial candidate benchmarks were given at the workshop in [WP2] [P20] (taxicab data) , [WP11] [P23] (based on STREAM memory benchmark), [WP16] (LinkedIn monitoring data) and [WP17] [P19] (SLAM robotic planning). Many of the other talks implicitly defined possible benchmarks: [WP1] [P5] [P7] with streaming Principal Component Analysis, [P13] with online clustering algorithms and [WP10] [P11] with smart electrical grids. Other examples that convey the breadth of challenges associated with streaming include ATLAS-LHC analysis, Avionics, Drug Discovery and Galaxy Zoo [Galaxy16]. The inspiration for many of the benchmarks must come from Industry and must involve interesting unrestricted datasets. Further we need to include benchmarks that present real time data from real time sources. A simple initial task is to agree on a "Hello Streaming World" example.

We need a more comprehensive understanding of and classification of streaming applications (perhaps expanding table 1) to scope a benchmark set. This could build on the Dwarfs [Dwarf06] for parallel computing and the Ogres [Ogre15] for big data applications. Some of the streaming characteristics include the type of events, the size of events, and the lifetime of events.

We need some number N (6-20?) of exemplar problems that identify use cases clearly and are well described. From this collection, we can work on challenges, abstractions, scaling, theory, stream operators, algorithms, languages and tools. We can look at them implemented with current software on current hardware and see what works and what doesn't. This process will generate a robust benchmark set. Note some examples can involve actual streaming data and others a repository of data that can be used to generate streams.

4.4 Streaming Software System and Algorithm Library

We need to assemble a general purpose high-utility open source scientific streaming software library that can be used by the community. It should be carefully integrated with the Apache Foundation software. It would hold enhancements to Apache software (for example Apache Storm and Mahout) as well as stand alone software. It would contain systems (middleware) and algorithms and would be streaming analog to SPIDAL and MIDAS [Spidal15]. It would have broadly applicable functions, support data reuse, and implement a suite of algorithms exploiting key streaming ideas such data sketches and certainly the challenges highlighted in NRC's computational giants. It would also have core capabilities such as state of art tools to allow Java and Python to run at high performance.

We need to aim at a toolkit that will support the stringent requirements of the major streaming science applications but is architected so same toolkit can be used across domains including LHC analysis, light sources, astronomy and simulation data. The development of a community toolkit should not be occur before the primary set of requirements and existing system support are understood and have stabilized.

4.5 Streaming System Infrastructure and its Characteristics

Although there is significant progress in developing streaming system software as discussed in sec. 4.2, there is no clear consensus on the type of hardware best suited to host this software. Rather communities use existing systems designed for other applications-- clouds or HPC -- to support streaming solutions. Of course if we include the sources of the data, any system must be distributed and must be scalable. Appropriate infrastructure may differ for research and production systems; it may also differ across application classes.

Then we need to understand differences in hardware, systems software and algorithms for the current batch bounded model and the newer unbounded streaming and interactive processing models. Are we at a point where unbounded tasks become a significant part of the workload and justify dedicated hardware?

Note that some communities such as ATLAS at LHC [WP12] [P26] are moving to a streaming model as reduce costs by reducing replica count and data transfer. This only works if streaming systems achieve high performance and are cost-effective. Simply good performance -- achieved by adding more nodes to our cloud -- is not enough unless cost of streaming model is less than traditional approaches as for example used today by the LHC.

Important system issues include latency in cloud-fog response and at the device level, power consumption. We have already stressed the need to understand what processing should be done at the three levels -- sensor (data source), fog (shared resources near source) and "cloud" (backend system). As well as performance there are cost and security issues in this choice. As well as hardware, these issues are all software challenges.

It is not clear if choice of HPC and Cloud infrastructures are intrinsic features of applications (commercial applications are different from scientific ones), software (are systems like Apache Storm biased to clouds) or perhaps both clouds and HPC are efficient and cost-effective. We need more comparisons between HPC and cloud implementations for the same application.

One important consideration is the cluster interconnection network performance requirements; one might expect that streaming data would need good network bandwidth but not the same latency seen in MPI (publish-subscribe systems tend to have millisecond and not microsecond latency). We expect it will be important to support interoperability between clouds and HPC clusters and to allow users to easily scale the size of their solution system.

As in sec. 4.5, we expect major challenges like streaming for ATLAS will develop much higher performance science streaming software tools and this would then allow more meaningful evaluations of needed hardware. Current solutions like Apache Storm do not match performance of commercial systems like Amazon Kinesis and will not meet ATLAS requirements. If we build a community as discussed in sec. 5, we can use answers to questions posed above to present community requirements to the hardware vendors.

4.6 Steering and Human in the Loop

White papers discussing this area include [WP3, WP5. WP6, WP7, WP8, WP9, WP15] and see [AIM15].

In many cases it is essential that a human guides the automatic responses to event streams and there is limited software support to enable this in current analytics systems. These cases range from avionics, where a pilot's decisions may be a matter of life or death, to cases when human expertise might identify regions or candidates of interest that need to be studied in greater detail in computational simulations or the drug discovery process. Another relevant use cases arises from light-sources such as APS/ALS as well as other experimental and observational facilities where a basic requirement is the integration of streaming data into analysis-simulation pipelines with humans in the loop. Further large scale simulation science motivates data and human-driven control for steering application/simulations. A final example is the steering of a UAV based upon real-time analysis of data received from the drone. Research ranging over systems, visualization and human computer interaction is needed to integrate human and computing knowledge into decision processes.

The NRC Frontiers in Massive Data Analysis report has a full chapter on "Human Interaction with Data". This covers Data Visualization and Exploration, as well as crowdsourcing and Hybrid Human/Computer Data Analysis. In this workshop, crowdsourcing was not discussed and visualization was not stressed. In latter, the importance of Python and Pandas was noted and the challenge of visualizing high dimensional time series was posed.

General challenges in this area include: getting a deeper understanding how scientists interact with streams; understanding how Human Computer Interaction HCI principles should be used; achieving scalability for both the algorithms and as well for tightly coupled workflows across different programming languages and models; how the system can maintain message ordering in spite of the high data volumes.; establishing and verifying ground truth; providing reproducibility in a highly adaptive environment with extensive user interaction, input and steering.

Looking at Transmission Electron Microscopy TEM and many related use cases, one can ask: how steering towards very rare events is managed; and secondly how one manages scenarios that depend upon both human and computing awareness.

Steering needs more attention -- partly because it is not a significant consideration in some major industrial applications where (near) real-time user interaction is not essential. Steering needs to be built in at all levels of system -- not just added at user level. Steering application types include: Steering the simulations, steering the sensors, steering the coupling/stream. We can pose this an optimization problem. Looking at industry, applications like connected vehicles could involve steering but software like Apache Storm does not directly support steering.

Steering opens up many issues related to resource management. There is a contrast between Dynamic Resources Management for steering and approaches based on Adaptive Workflow.

Steering can be considered as hypothesis based reasoning. A general research question is to understand how, when and where to intercede in automated reasoning. We need to give users just the right amount of information and to have a clear logical model for steering. Note that scientists are often more comfortable steering the instrument, as opposed to relying on a machine learning algorithm. We must understand how to select a division of decision making between system and human. We need to understand the dividing line between human knowledge and computing knowledge in a system that has a broad range of tuning parameters? As in [WP10], we must often have baseline observations to identify normal behavior, and then design a response as a function of violation of accepted tolerance values.

5 Build and sustain community

The workshop was well-received and deemed successful by the participants. It was viewed as an interesting start to the development of a community with a good mix of participants and much potential for a community to be expanded. There was a critical recognition that the topic is in need of further attention because of its growing importance. The workshop participants felt that the diverse nature of the nascent community was well represented by the workshop attendees and the topics presented. There was a strong support for establishing and enhancing this community and supporting and encouraging variety of community activities.

5.1 Community Activities

There were several suggestions for community building activities starting with expanding the STREAM2015 workshop website http://streamingsystems.org to a clearinghouse of community information and activities. Additional efforts will be required to organize community activities including application surveys, special issues, workshops, and industry engagement through advisory committees. The community would have as usual multiple subgroups covering topics including those we identified in this meeting and report.

We will organize the STREAM 2016 workshop which will be informed by the results of this workshop and hopefully broaden the community as well as covered applications and technologies. It has a DoE focus and we discussed reaching out to the DOE Light and Neutron sources and trying to cover all relevant DOE applications. It was suggested that we look at other ASCR workshops as some white papers there discuss streaming. The “Data Management, Analysis and Visualization for Experimental and Observational Data (EOD) Workshop” [EOD15] held recently was highlighted.

Several existing conference and meeting venues were discussed to grow the community. A BoF or workshop as SC'16 was seen as promising. Efforts to engage additional key people and activities that could not make it to STREAM 2015 should continue. This includes industry representatives from GE, IBM, FB, Twitter and the financial sector.

The application benchmarks and streaming software libraries discussed in secs. 4.3 and 4.4 are clearly valuable community resources. As a precursor to these collections, we could consider a survey of existing software systems (highlighting those in open source) and a survey of applications - existing and potential. These surveys can be analyzed to refine research challenges. We could also consider streaming data challenges following either the successful provenance challenge [Provenance08] or those popularized by Kaggle [Kaggle15]. Work with Apache on software systems such as an improved Storm [Storm15] was supported.

The need for a crisp definition of Streaming systems as an elevator speech was stressed. Our community organization could act as a broker for summer student opportunities. We also discussed journal special Issues and an edited book.

5.2 Education and Training

Streaming data applications are seen as critical to economic growth and development, and they present a tremendous opportunity for workforce development. In addition, a growing number of scientific and engineering disciplines are being inundated with streaming data, either from observational facilities or through simulations. As a result, an understanding of the emerging tools and environments for streaming data will become increasingly important for future skilled workers. Understanding the best practices and developing curriculum support was universally regarded as essential at the workshop. Industrial participants agreed there is tremendous, and growing, demand for the skills that the current community is only now discovering.

Brunner [P8] stressed the relationship between a streaming systems curriculum and the broader area of data science and the discussed an interesting link to a data science incubator at UIUC. Braverman (http://www.cs.jhu.edu/~vova/) has already integrated a discussion on new algorithms needed for streaming data into courses. The workshop participants widely agreed that data science education, and in particular education in streaming data applications, requires realistic datasets---in particular from industry. This point reflects the previous discussion in sec. 4.3.

6 Summary

This report presents a summary of discussions of 43 attendees at the STREAM2015 workshop together with information presented in 29 presentations and 17 white papers. Streaming data is of growing importance in the number of applications, the volume of data in each application and the sophistication of the analytics applied. The study of streaming data is not a new field but there are few activities directly tackling it. Rather streaming data is often viewed as a byproduct of other topics such as particular experiments, real-time systems, publish-subscribe middleware or associated with domains like cyber-physical systems or online collaboration. The result is a paucity of research and a lack of understanding as to how to set up hardware and software infrastructure to best deal with streaming data.

The design of distributed computing infrastructure that supports streaming data as a first class capability and not an adjunct consideration, is a major interest of both NSF and DoE as they design and deploy National distributed computing infrastructure, to support the analysis of streamed observational data.

The meeting covered existing technology, and divided current approaches to into four areas: commercial, Apache Foundation software-based, domain specific, and finally research systems. The current solutions and future research were discussed for two major themes -- new algorithms and programming model/run-time. There has been significant research into interesting aspects of online (streaming) algorithms; these include efficiency (only touch each data-point once), sampling (of huge datasets) but there is modest practical use and attention to issues like parallelism. For programming models, highlights included need to move away from traditional batch scheduling, the challenges of performance and fault-tolerance plus the identification of application requirements and in particular the differences between commercial and research streaming data.

Although we identified 8 categories of applications, we believe there is a need for more detailed categorization, an analysis of software and hardware infrastructure requirements and development of associated “mini-applications” for use in research and testing.

The importance of the area, the significant number of interesting challenges and enthusiasm of participants motivated discussion of forming a formal “streaming data” community. This could guide and certify application surveys, journal special issues, workshops, industry engagement through advisory committees, and Education and Training. Further it could act as steward of collections of important software as well as benchmarks.

The many open issues will be addressed in the next STREAM2016 workshop.

7 Acknowledgements

The workshops STREAM2015 and STREAM2016 are made possible by support from the National Science Foundation (Division of Advanced Cyberinfrastructure) and Department of Energy (Advanced Scientific Computing Research) and the Air Force Office of Scientific Research (AFOSR). We thank Julie Overfield and Gary Miksik for their efficient support of meeting logistics.

8 Appendices

8.1 Participants

- Yanif Ahmad, Johns Hopkins University
- Ilkay Altintas, UCSD and SDSC
- Amy Apon, National Science Foundation and Clemson University
- Roger Barga, Amazon
- Pete Beckman, Argonne National Laboratory
- Vladimir Braverman, Johns Hopkins
- Robert Brunner, UIUC
- Richard Carlson, Department of Energy
- David Crandall, Indiana University

- Kaushik De, University of Texas at Arlington
- Dan Fay, Microsoft Research
- Geoffrey Fox, Indiana University
- Dennis Gannon, Indiana University
- Mark Greaves, Pacific Northwest National Laboratory
- Shantenu Jha, Rutgers University
- Supun Kamburugamuve, Indiana University
- Kerstin Kleese Van Dam, Brookhaven National Laboratory
- Xiangbo Li, University of Louisiana Lafayette
- Madhav Marathe, Virginia Tech
- Stefano Markidis, KTH University, Stockholm, Sweden
- Suresh Marru, Indiana University
- André Martin, TU Dresden, Germany
- Bojan Nikolic, University of Cambridge, UK
- Shrideep Pallickara, Colorado State University
- Milinda Pathirage, Indiana University
- Erik Paulson, Johnson Controls Corp
- Jelena Pjesivac-Grbovic, Google
- Judy Qiu, Indiana University
- Maryam Rahnemoonfar, Texas A&M Corpus Christi
- Lavanya Ramakrishnan, Lawrence Berkeley National Laboratory
- Ben Ring, Johns Hopkins University
- Alex Szalay, Johns Hopkins University
- Joshua Suetterlein, Pacific Northwest National Laboratory
- Martin Swany, Indiana University
- Alex Szalay, Johns Hopkins
- Mathew Thomas, Pacific Northwest National Laboratory
- Carlos Varela, Rensselaer Polytechnic Institute
- Von Welch, Indiana University
- Torre Wenaus, Brookhaven National Laboratory
- Tom Woolf, Johns Hopkins University
- Justin Wozniak, Argonne National Laboratory
- John Wu, Lawrence Berkeley National Laboratory
- Tak-Lon Wu, Indiana University

8.2 Workshop Presentations

P1. Roger Barga, *Kinesis: Data Stream Processing Services* [pdf]
P2. Pete Beckman, *Array of Things* [video] [pdf]
P3. Jelena Pjesivac-Grbovic, *Google Cloud Dataflow* [video] [pdf]
P4. Justin Wozniak, *Streaming, Storing, and Sharing Big Data for Light Source Science* [video] [pdf]
P5. Alex Szalay, *Streaming Problems in Astrophysics* [video] [pdf]
P6. Bojan Nikolic, *Data Processing for the Square Kilometre Array Telescope* [video] [pdf]
P7. Vladimir Braverman, *Streaming Algorithms for Cosmological Simulations and Beyond* [video] [pdf]
P8. Robert Brunner, *Breaking Data Science* [video] [pdf]

P9. Yanif Ahmad, *Accelerating Scientific Discovery through Data-Driven Control and Real-Time Analytics* [video] [pdf]
P10. Erik Paulson and Samuel Hamilton, *IIOT at Johnson Controls* [video] [pdf]
P11. John Wu, *Baseline in Streaming Data Analysis: What, Why, and How?* [video] [pdf]
P12. Madhav Marathe, *Real-Time Network Science* [video] [pdf]
P13. Judy Qiu, *Parallel Clustering of High-Dimensional Social Media Data Streams* [video] [pdf]
P14. Von Welch, *Urban Sensor Data Privacy Issues: Findings of the Array of Things (AoT)* [video] [pdf]
P15. Kerstin Kleese Van Dam, Analysis and Decision Making in Big Data Environments Analysis in Motion (AIM) [video] [pdf]
P16. Shrideep Pallickara, *Real-Time Stream Processing For Sensing Environments* [video] [pdf]
P17. Martin Swany, *In-Network Processing for Streaming Data* [video] [pdf]
P18. Dan Fay, *Streaming on Microsoft Azure* [video] [pdf]
P19. Supun Kamburugamuve, *Streaming Applications for Robots with Real Time QoS* [video] [pdf]
P20. Andre Martin, *StreamMapReduce: When Stream Processing crosses MapReduce* [video] [pdf]
P21. Ilkay Altinas, *Dynamic Data-Driven Applications using Workflows as a Programming Model for Scalable and Reproducible Streaming and Steering* [video] [pdf]
P22. Milinda Pathirage, *Distributed Streaming SQL* [video] [pdf]
P23. Stefano Markidis, *Streaming Programming Systems for Exascale* [video] [pdf]
P24. Joshua Suetterlein, *Toward a Unified HPC and Big Data Runtime* [video] [pdf]
P25. Carlos Varela, *Dynamic Data-Driven Avionics Systems: Inferring Failure Modes from Data Streams* [video] [pdf]
P26. Torre Wenaus, *Science-Aware Dynamic Data Delivery at the Exascale* [video] [pdf]
P27. Maryam Rahnemoonfar, *Real time Oil and Gas source Identification using Unmanned Aerial Systems* [pdf]
P28. Xiangbo Li, *CVS: A Cost-Efficient and QoS-Aware Cloud Video Streaming* [video] [pdf]
P29. David Crandall, *Streaming Deep Learning for Robotics and Mobile Cameras* [video] [pdf]

8.3 Workshop White Papers

WP1. Title: Streaming Algorithms for Astronomic Simulations; Authors: Vladimir Braverman, Alex Szalay; Institution: Johns Hopkins University
WP2. Title: StreamMapReduce: When Stream Processing crosses MapReduce; Authors: André Martin, Andrey Brito, Christof Fetzer; Institution: Technische Universität Dresden, Dresden, Germany; Universidade Federal de Campina Grande, Campina Grande, Brazil
WP3. Title: Dynamic Data-Driven Applications using Workflows as a Programming Model for Scalable and Reproducible Streaming and Steering; Authors: Ilkay Altintas; Institution: UC San Diego
WP4. Title: Toward a Unified HPC and Big Data Runtime; Authors: Joshua Suetterlein, Joshua Landwehr, Joseph Manzano, Andres Marquez; Institution: University of Delaware, Pacific Northwest National Lab
WP5. Title: Streaming, Storing, and Sharing Big Data for Light Source Science; Authors: Justin M. Wozniak, Kyle Chard, Ben Blaiszik, Michael Wilde, Ian Foster; Institution: Argonne National Laboratory, University of Chicago

WP6. Title: Accelerating the Experimental Feedback Loop: Data Streams at the APS; Authors: Ian Foster, Tekin Bicer, Raj Kettimuthu, Michael Wilde, Justin Wozniak, Francesco de Carlo, Ben Blaiszik, Kyle Chard, Francesco de Carlo, and Ray Osborn; Institution: Argonne National Laboratory

WP7. Title: Accelerating Scientific Discovery through Data-Driven Control and Scalable Real-Time Analytics; Authors: Ben Ring, Yanif Ahmad, Tom Woolf; Institution: Johns Hopkins University

WP8. Title: Real time Oil and Gas source Identification using Unmanned Aerial Systems; Authors: Maryam Rahnemoonfar; Institution: Texas A & M University - Corpus Christi

WP9. Title: Analysis in Motion (AIM); Authors: Kerstin Kleese van Dam, Mark Greaves, Rob Jasper, Nigel Browning; Institution: Pacific Northwest National Laboratory, Brookhaven National Laboratory

WP10. Title: Baseline in Streaming Data Analysis: What, Why, How?; Authors: John Wu; Institution: Lawrence Berkeley National Laboratory

WP11. Title: Streaming Programming Systems at Exascale; Authors: Stefano Markidis, Ivy Bo Peng, and Erwin Laure; Institution: KTH Royal Institute of Technology

WP12. Title: Science-Aware Dynamic Data Delivery at the Exascale; Authors: M. Ernst, A. Klimentov, N. Malitsky, T. Wenaus, P. Calafiura, D. Malon, R. Mount, S. Jha, K. De; Institution: Brookhaven National Laboratory, Lawrence Berkeley National Laboratory, Rutgers University, Argonne National Laboratory, University of Texas at Arlington, Stanford Linear Accelerator Center

WP13. Title: Cloud-Based Video Streaming for Energy and Compute-Limited Thin Clients; Authors: Xiangbo Li, Mohsen Amini Salehi, Magdy Bayoumi; Institution: The Center of Advanced Computer Studies (CACS), University of Louisiana Lafayette

WP14. Title: Urban Sensor Data Privacy Issues: Findings of the Array of Things (AoT) Privacy Breakout Group; Authors: Von Welch, Charlie Catlett; Institution: Indiana University, Argonne National Laboratory

WP15. Title: Dynamic Data Driven Avionics Systems; Authors: Carlos Varela; Institution: Rensselaer Polytechnic Institute

WP16. Title: Fast Data Management with Distributed Streaming SQL; Authors: Milinda Parthirage and Beth Plale; Institution: School of Informatics and Computing, Indiana University

WP17. Title: Streaming Application for IOT with Real Time QoS; Authors: Supun Kamburugamuve & Geoffrey Fox; Institution: School of Informatics and Computing, Indiana University

8.4 Citations

- **[Abadi05]** Abadi, D.J., Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. Maskey, A. Rasin, and E. Ryvkina. “The Design of the Borealis Stream Processing Engine”. 2nd Biennial Conference on Innovative Data Systems Research (CIDR'05), Asilomar, CA, January 2005.
- **[AIM15]** Pacific Northwest National Laboratory “Analysis in Motion” Initiative http://aim.pnnl.gov
- **[AuroraSched03]** Carney, Don, Uğur Çetintemel, Alex Rasin, Stan Zdonik, Mitch Cherniack, and Mike Stonebraker. "Operator scheduling in a data stream manager." In*Proceedings of the 29th international conference on Very large data bases-Volume 29*, pp. 838-849. VLDB Endowment, 2003.
- **[Bloom70]** B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” Communications of the ACM, vol. 13, no. 7, pp. 422– 426, 1970.
- **[Bench04]** Arasu, Arvind, Mitch Cherniack, Eduardo Galvez, David Maier, Anurag S. Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. "Linear road: a stream data management benchmark." In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pp. 480-491. VLDB Endowment, 2004.

- **[Cherniack03]** Cherniack, M., H. Balakrishnan, M. Balazinska, D. Carney, U. Cetintemel, Y. Xing, and S.B. Zdonik. “Scalable Distributed Stream Processing”. First Biennial Conference on Innovative Database Systems (CIDR'03), Asilomar, CA, January 2003.
- **[Classify00]** Domingos, Pedro, and Geoff Hulten. "Mining high-speed data streams." In*Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71-80. ACM, 2000.
- **[Cluster02]** O'callaghan, Liadan, Adam Meyerson, Rajeev Motwani, Nina Mishra, and Sudipto Guha. "Streaming-data algorithms for high-quality clustering." In*icde*, p. 0685. IEEE, 2002.
- **[Cluster03]** Charikar, Moses, Liadan O'Callaghan, and Rina Panigrahy. "Better streaming algorithms for clustering problems." In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pp. 30-39. ACM, 2003.
- **[Cormode12]** G. Cormode and S. Muthukrishnan. “Approximating data with the count-min data structure”. IEEE Software, (2012). http://dimacs.rutgers.edu/~graham/pubs/papers/cmsoft.pdf A general introduction to the sketch method and its uses
- **[DCL15053]** Dear Colleague Letter: Future Directions for NSF Advanced Computing Infrastructure (ACI) http://www.nsf.gov/pubs/2015/nsf15053/nsf15053.jsp [accessed 2016 Jan 09]
- **[DDDAS15]** DDDAS: Dynamic Data Driven Applications Systems NSF Site. [accessed 2015 July 22]; http://www.nsf.gov/cise/cns/dddas/; Dynamic Data Driven Applications Systems (DDDAS) AFOSR Site. [accessed 2015 July 22];: https://community.apan.org/wg/afosr/w/researchareas/7661.dynamic-data-driven-applications-systems-dddas/;
  DDDAS Dynamic Data-Driven Applications System Showcase. [accessed 2015 July 22]; http://www.1dddas.org/
- **[Dwarf06]** Asanovic, K., R. Bodik, B.C. Catanzaro, J.J. Gebis, P. Husbands, K. Keutzer, D.A. Patterson, W.L. Plishker, J. Shalf, S.W. Williams, and K.A. Yelick. “The Landscape of Parallel Computing Research: A View from Berkeley”. 2006 December 18 http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html.
- **[EOD15]** “Data Management, Analysis and Visualization for Experimental and Observational Data (EOD) Workshop” September 29 – October 1, 2015 Bethesda, MD. https://www.orau.gov/eodworkshop2015/default.htm
- **[Flajolet07]** Flajolet, P.; Fusy, E.; Gandouet, O.; Meunier, F. “HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm”. AOFA ’07: Proceedings of the 2007 International Conference on the Analysis of Algorithms.
- **[Flink15]** https://flink.apache.org/ Apache Flink Batch and Stream Data Processing System
- **[Freq04]** Chi, Yun, Haixun Wang, Philip S. Yu, and Richard R. Muntz. "Moment: Maintaining closed frequent itemsets over a stream sliding window." In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pp. 59-66. IEEE, 2004.
- **[Freq05]** Jin, Ruoming, and Gagan Agrawal. "An algorithm for in-core frequent itemset mining on streaming data." In *Data Mining, Fifth IEEE International Conference on*, pp. 8-pp. IEEE, 2005.
- **[Freq06]** Chang, Joong Hyuk, and Won Suk Lee. "Finding recent frequent itemsets adaptively over online data streams." In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 487-492. ACM, 2003.
- **[Galaxy16]** http://www.galaxyzoo.org/ Citizen Science galaxy classification project
- **[Gedik08]** Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S. Yu, and Myungcheol Doo. 2008. “SPADE: the system S declarative stream processing engine”. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (SIGMOD '08). ACM, New York, NY, USA, 1123-1134. DOI=http://dx.doi.org/10.1145/1376616.1376729

- **[Guha13]** Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. 2003. “Clustering Data Streams: Theory and Practice”. IEEE Trans. on Knowl. and Data Eng. 15, 3 (March 2003), 515-528. DOI=http://dx.doi.org/10.1109/TKDE.2003.1198387
- **[Hirzel14]** Martin Hirzel, Robert Soulé, Scott Schneider, Bugra Gedik, and Robert Grimm. 2014. A catalog of stream processing optimizations. ACM Comput. Surv. 46, 4, Article 46 (March 2014), 34 pages. DOI=http://dx.doi.org/10.1145/2528412
- **[Heron15]** Kulkarni, Sanjeev, Nikunj Bhagat, Masong 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*, pp. 239-250. ACM, 2015.
- **[Kafka15]** http://kafka.apache.org/ Apache Kafka Publish-Subscribe Environment
- **[Kaggle15]** https://www.kaggle.com/ Home of Data Science and Competitions
- **[NavyCloud14]** ONR. Data Focused Naval Tactical Cloud (DF-NTC): ONR Information Package. 2014 June 24; http://www.onr.navy.mil/~/media/Files/Funding-Announcements/BAA/2014/14-011-Attachment-0001.ashx
- **[Neptune16]** Thilina Buddhika and Shrideep Pallickara. Neptune: Real Time Stream Processing for Internet of Things and Sensing Environments. (To appear) Proceedings of the 30th IEEE International Parallel & Distributed Processing Symposium. Chicago, USA. 2016.
- **[Ogre14]** Geoffrey C. Fox, Shantenu Jha, Judy Qiu, and Andre Luckow, “Towards an Understanding of Facets and Exemplars of Big Data Applications”, in 20 Years of Beowulf: Workshop to Honor Thomas Sterling's 65th Birthday. October 13-14, 2014 Annapolis. http://dx.doi.org/10.1145/2737909.2737912
- **[Ogre15]** Geoffrey C. Fox , Shantenu Jha, Judy Qiu, Saliya Ekanayake, and Andre Luckow, “Towards a Comprehensive Set of Big Data Benchmarks”, Chapter in Big Data and High Performance Computing, Lucio Grandinetti and Gerhard Joubert, Editors. 2015, IOS. http://grids.ucs.indiana.edu/ptliupages/publications/OgreFacetsv9.pdf.
- **[Provenance08]** Moreau, L. et al. (2008), Special Issue: The First Provenance Challenge. Concurrency Computat.: Pract. Exper., 20: 409–418. doi:10.1002/cpe.1233
- **[Seep15]** SEEP parallel data processing system from Large-Scale Distributed Systems (LSDS) research group at Imperial College London https://github.com/lsds/Seep/
- **[Soule10]** Robert Soulé, Martin Hirzel, Robert Grimm, Bugra Gedik, Henrique Andrade, Vibhore Kumar, and Kun-Lung Wu. 2010. “A universal calculus for stream processing languages”. In Proceedings of the 19th European conference on Programming Languages and Systems (ESOP'10), Andrew D. Gordon (Ed.). Springer-Verlag, Berlin, Heidelberg, 507-528. DOI=http://dx.doi.org/10.1007/978-3-642-11957-6_27 https://www.cs.nyu.edu/rgrimm/papers/esop10.pdf
- **[Soule12]** Robert Soulé, Martin Hirzel, Bugra Gedik, and Robert Grimm. 2012. From a calculus to an execution environment for stream processing. In Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems (DEBS '12). ACM, New York, NY, USA, 20-31. DOI=http://dx.doi.org/10.1145/2335484.2335487 http://www.inf.usi.ch/faculty/soule/debs12a.pdf
- **[Samza15]** http://samza.apache.org/ Apache Samza Streaming Software
- **[Spark15]** http://spark.apache.org/ fast and general engine for large-scale data processing
- **[Spidal15]** http://spidal.org/ Middleware for Data-Intensive Analytics and Science (MIDAS) and Scalable Parallel Interoperable Data Analytics Library (SPIDAL)
- **[SQL02]** Babcock, Brian, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. "Models and issues in data stream systems." In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 1-16. ACM, 2002.

- **[SQL04]** Arasu, Arvind, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. "Stream: The stanford data stream management system." *Book chapter*(2004).
- **[Storm15]** http://storm.apache.org/ Apache Storm Streaming Software
- **[Streammine15]** StreamMine3G scalable, elastic and fault tolerant event processing and data streaming engine. https://streammine3g.inf.tu-dresden.de/
- **[Survey16]** Supun Kamburugamuve and Geoffrey Fox, “Survey of Distributed Stream Processing”, Technical Report January 8 2016 http://dsc.soic.indiana.edu/publications/survey_distributed_stream_frameworks.pdf
- **[Thies02]** William Thies, Michal Karczmarek, and Saman P. Amarasinghe. 2002. StreamIt: A Language for Streaming Applications. In Proceedings of the 11th International Conference on Compiler Construction (CC '02), R. Nigel Horspool (Ed.). Springer-Verlag, London, UK, UK, 179-196.