MPI, Dataflow, Streaming: Messaging for Diverse Requirements

Argonne National Lab, Chicago, Illinois,

Geoffrey Fox, September 25, 2017

Indiana University, Department of Intelligent Systems Engineering

gcf@indiana.edu, http://www.dsc.soic.indiana.edu/, http://spidal.org/

Work with Judy Qiu, Shantenu Jha, Supun Kamburugamuve, Kannan Govindarajan, Pulasthi Wickramasinghe

Abstract: MPI, Dataflow, Streaming: Messaging for Diverse Requirements

- We look at messaging needed in a variety of parallel, distributed, cloud and edge computing applications.
- We compare technology approaches in MPI, Asynchronous Many-Task systems, Apache NiFi, Heron, Kafka, OpenWhisk, Pregel, Spark and Flink, event-driven simulations (HLA) and Microsoft Naiad.
- We suggest an event-triggered dataflow polymorphic runtime with implementations that trade-off performance, fault tolerance, and usability.
- Integrate Parallel Computing, Big Data, Grids

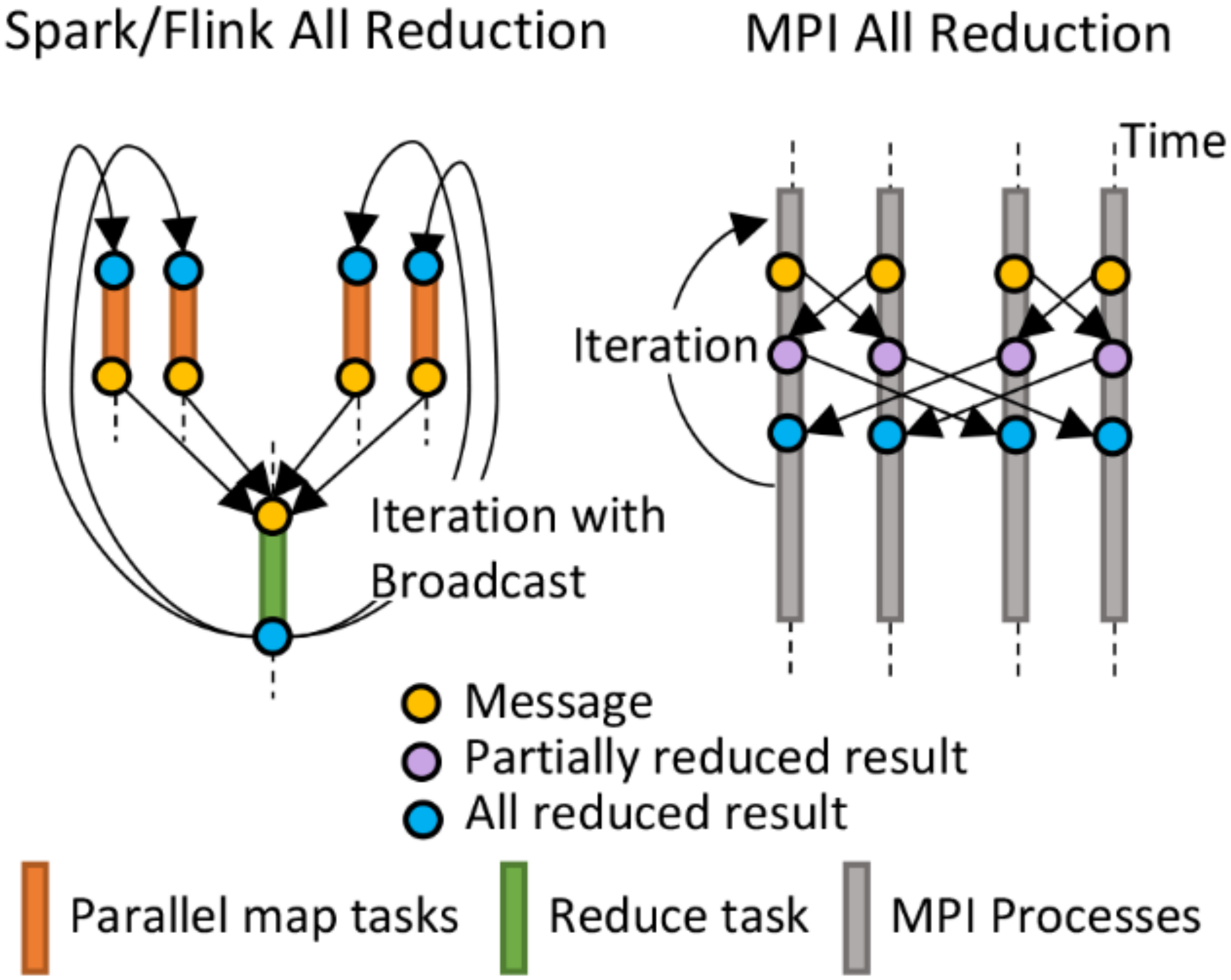
Motivating Remarks

- **MPI is wonderful** (and impossible to beat?) for closely coupled parallel computing but
  - There are many other regimes where either parallel computing and/or message passing essential
  - Application domains where other/higher-level concepts successful/necessary
- **Internet of Things** and **Edge Computing** growing in importance
- **Use of public clouds increasing rapidly**
  - Clouds becoming diverse with subsystems containing GPU's, FPGA's, high performance networks, storage, memory …
- **Rich software stacks**:
  - HPC (High Performance Computing) for Parallel Computing **less used than(?)**
  - Apache for Big Data Software Stack ABDS including some edge computing (streaming data)
- A lot of confusion coming from different communities (database, distributed, parallel computing, machine learning, computational/data science) investigating similar ideas with little knowledge exchange and mixed up requirements

Requirements

- On general principles **parallel and distributed computing** have different requirements even if sometimes similar functionalities
  - Apache stack ABDS typically uses distributed computing concepts
  - For example, Reduce operation is different in MPI (Harp) and Spark
- Large scale simulation requirements are well understood
- Big Data requirements are not clear but there are a few key use types
  1) **Pleasingly parallel** processing (including **local machine learning LML**) as of different tweets from different users with perhaps MapReduce style of statistics and visualizations; possibly Streaming
  2) **Database model** with queries again supported by MapReduce for horizontal scaling
  3) **Global Machine Learning GML** with single job using multiple nodes as classic parallel computing
  4) **Deep Learning** certainly needs HPC – possibly only multiple small systems
- Current workloads stress 1) and 2) and are suited to current clouds and to ABDS (with no HPC)
  - This explains why Spark with poor GML performance is so successful and why it can ignore MPI

HPC Runtime versus ABDS distributed Computing Model on Data Analytics



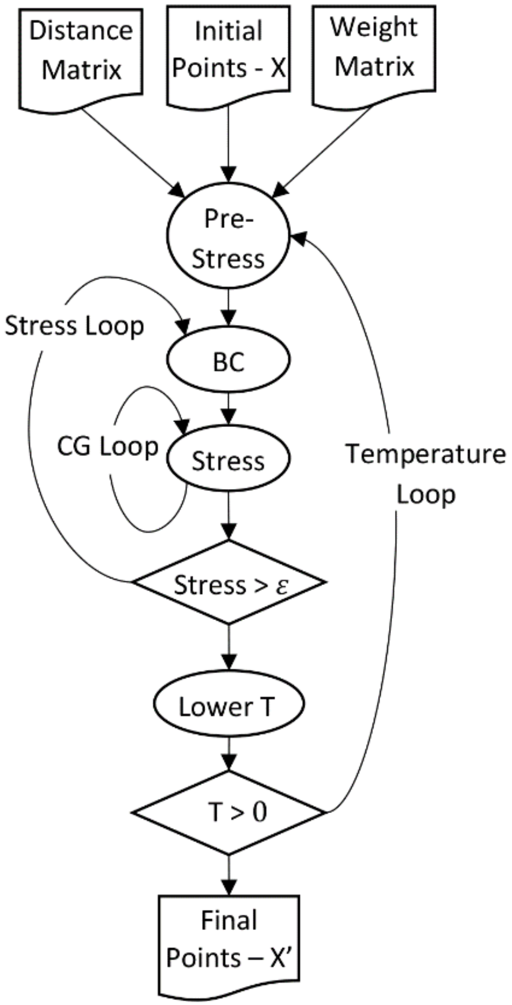
Hadoop writes to disk and is **slowest**; **Spark** and **Flink** spawn many processes and do not support AllReduce directly; **MPI** does in-place combined reduce/ broadcast and is **fastest**

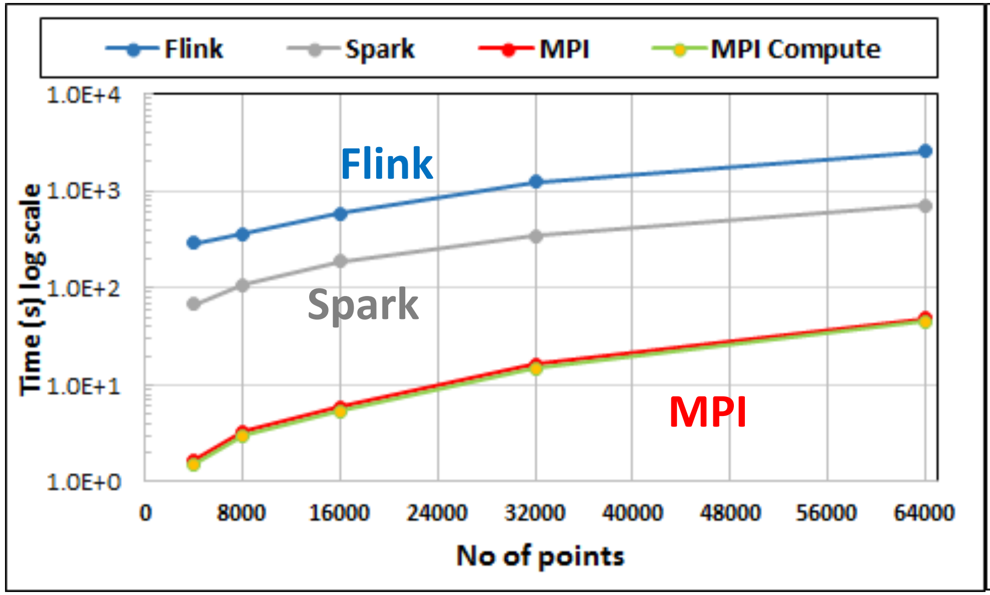
Need Polymorphic Reduction capability choosing best implementation

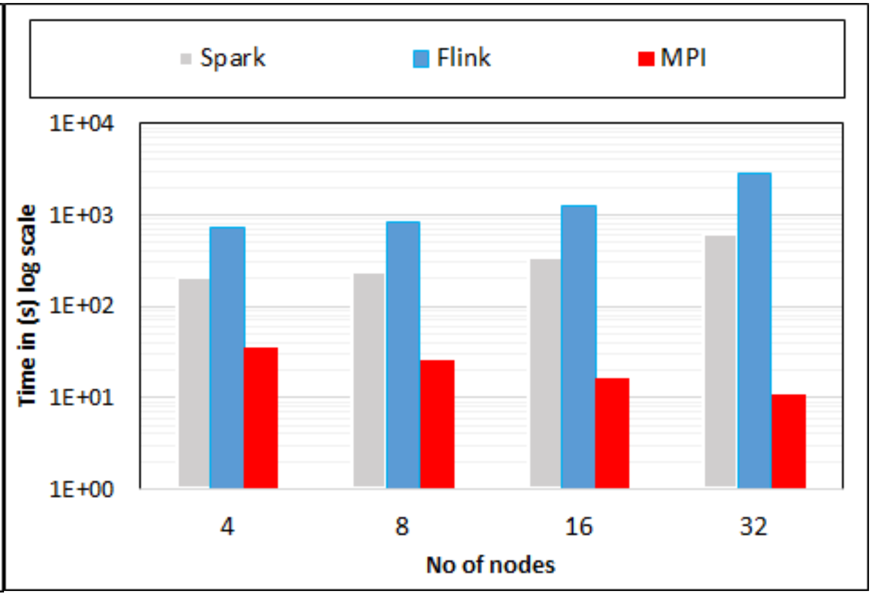
Use HPC architecture with

- Mutable model
- Immutable data

Multidimensional Scaling: 3 Nested Parallel Sections



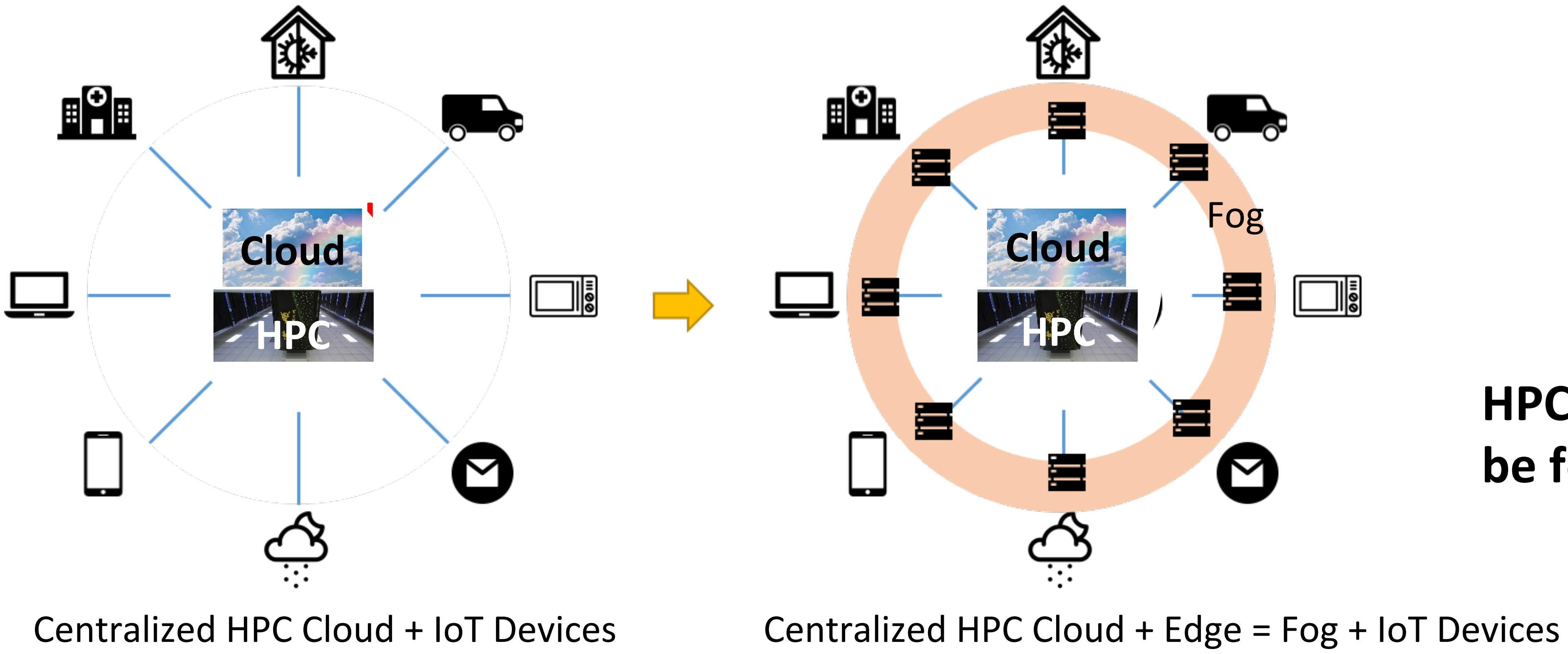




MPI Factor of 20-200 Faster than Spark/Flink

MDS execution time on **16 nodes** with 20 processes in each node with varying number of points

MDS execution time with 32000 points on **varying number of nodes**. Each node runs 20 parallel tasks



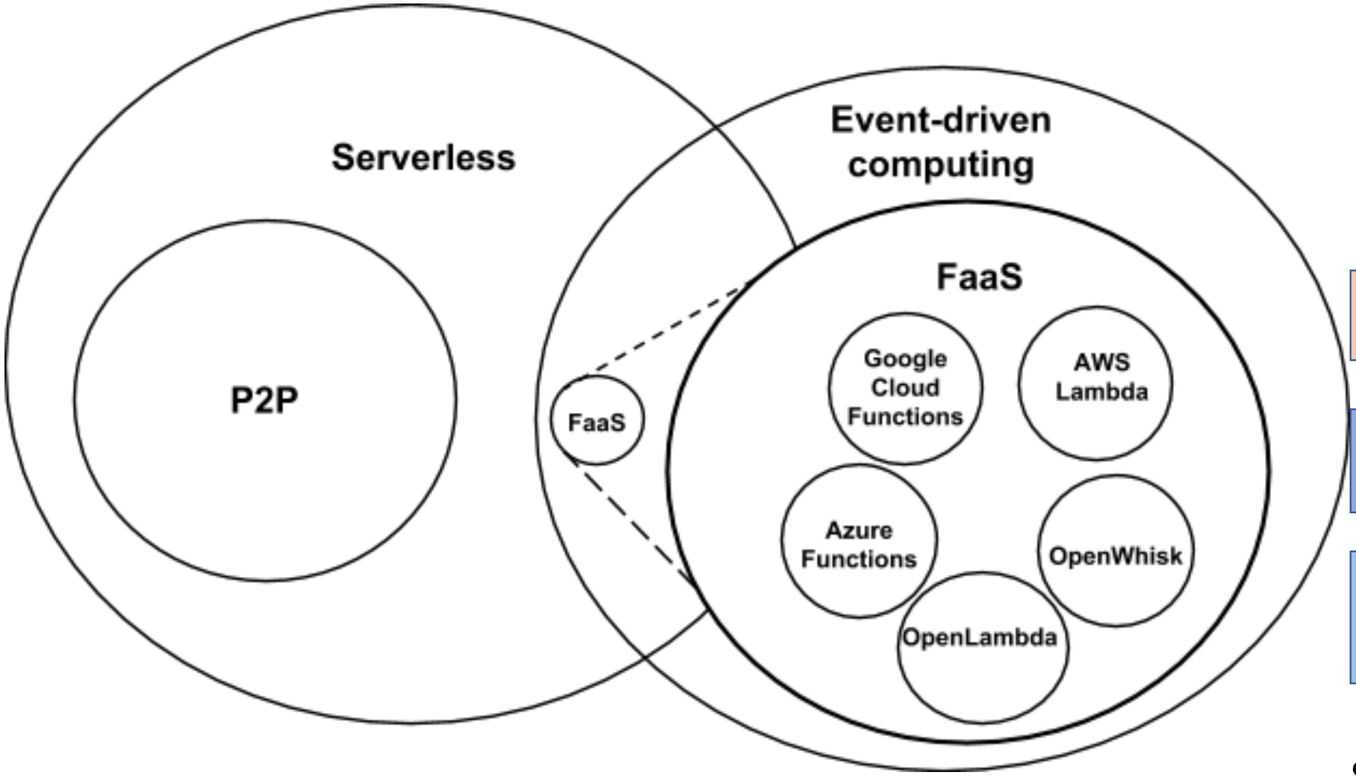
HPC Cloud can be federated

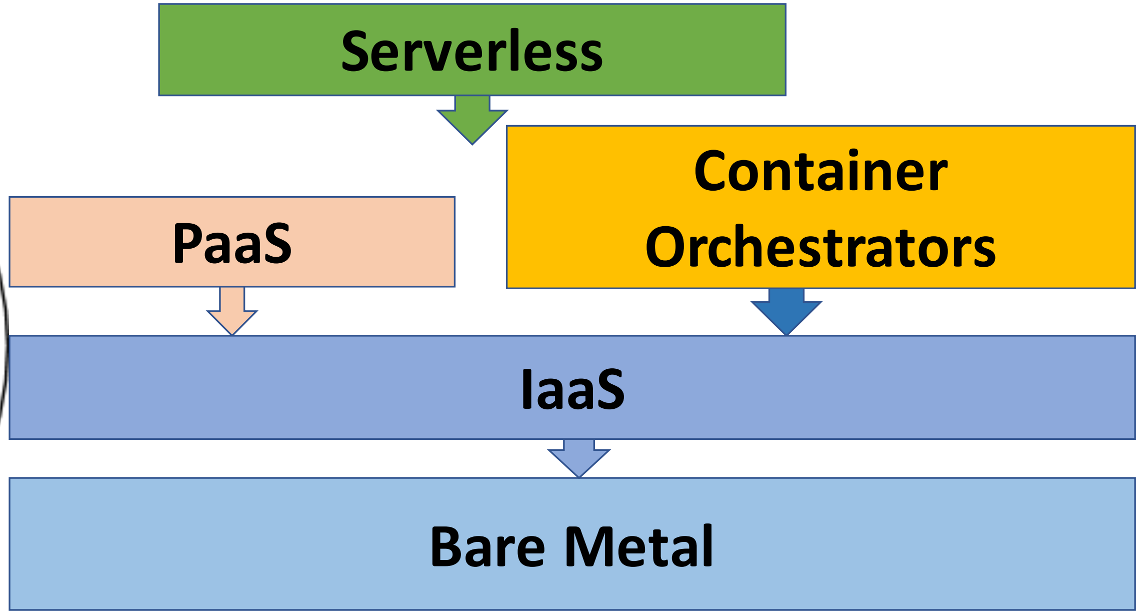
Implementing Twister2 to support a Grid linked to an HPC Cloud

Serverless (server hidden) computing attractive to user:
"No server is easier to manage than no server"

- Cloud-owner Provided Cloud-native platform for
- Event-driven applications which
- Scale up and down instantly and automatically
  Charges for actual usage at a millisecond granularity

GridSolve, Neos were FaaS





See review http://dx.doi.org/10.13140/RG.2.2.15007.87206

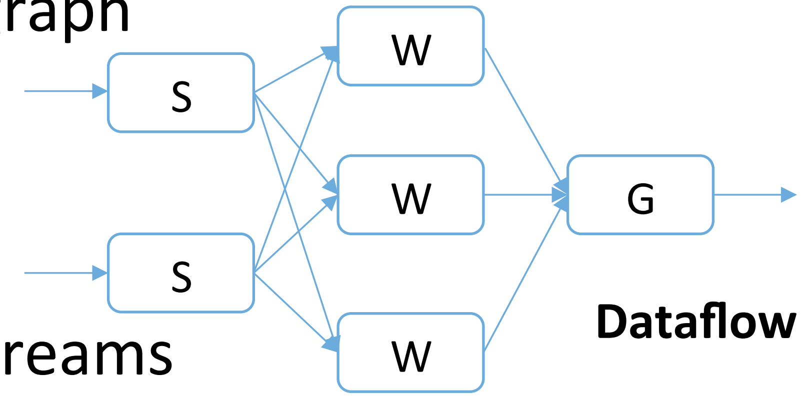
Twister2: “Next Generation Grid - Edge – HPC Cloud”

- Original 2010 **Twister** paper was a particular approach to Map-Collective iterative processing for machine learning
- **Re-engineer** current Apache Big Data software systems as **a toolkit** with MPI as an option
  - Base on **Apache Heron** as most modern and “neutral” on controversial issues
- Support a **serverless (cloud-native) dataflow event-driven HPC-FaaS (microservice)** framework running across application and geographic domains.
  - Support all types of Data analysis from GML to Edge computing
- Build on Cloud best practice but use HPC wherever possible to get high performance
- Smoothly support current paradigms Naiad, Hadoop, Spark, Flink, Storm, Heron, MPI …
- Use **interoperable** common abstractions but multiple **polymorphic** implementations.
  - i.e. do not require a single runtime
- Focus on Runtime but this implies HPC-FaaS programming and execution model
- This describes **a next generation Grid** based on data and edge devices – not computing as in original Grid

See long paper http://dsc.soic.indiana.edu/publications/Twister2.pdf

Communication (Messaging) Models

- **MPI Gold Standard:** Tightly synchronized applications
  - Efficient communications (µs latency) with use of advanced hardware
  - In place communications and computations (Process scope for state)
- **Basic (coarse-grain) dataflow:** Model a computation as a graph
  - Nodes do computations with Task as computations and edges are asynchronous communications
  - A computation is activated when its input data dependencies are satisfied



- **Streaming dataflow: Pub-Sub** with data partitioned into streams
  - Streams are unbounded, ordered data tuples
  - Order of events important and group data into time windows
- **Machine Learning dataflow:** Iterative computations
  - There is both Model and Data, but only communicate the model
  - **Collective communication** operations such as AllReduce AllGather (no differential operators in Big Data problems
  - Can use in-place MPI style communication

Core SPIDAL Parallel HPC Library with Collective Used

- DA-MDS Rotate, AllReduce, Broadcast
- Directed Force Dimension Reduction AllGather, Allreduce
- Irregular DAVS Clustering Partial Rotate, AllReduce, Broadcast
- DA Semimetric Clustering Rotate, AllReduce, Broadcast
- K-means AllReduce, Broadcast, AllGather DAAL
- SVM AllReduce, AllGather
- SubGraph Mining AllGather, AllReduce
- Latent Dirichlet Allocation Rotate, AllReduce
- Matrix Factorization (SGD) Rotate DAAL
- Recommender System (ALS) Rotate DAAL
- Singular Value Decomposition (SVD) AllGather DAAL

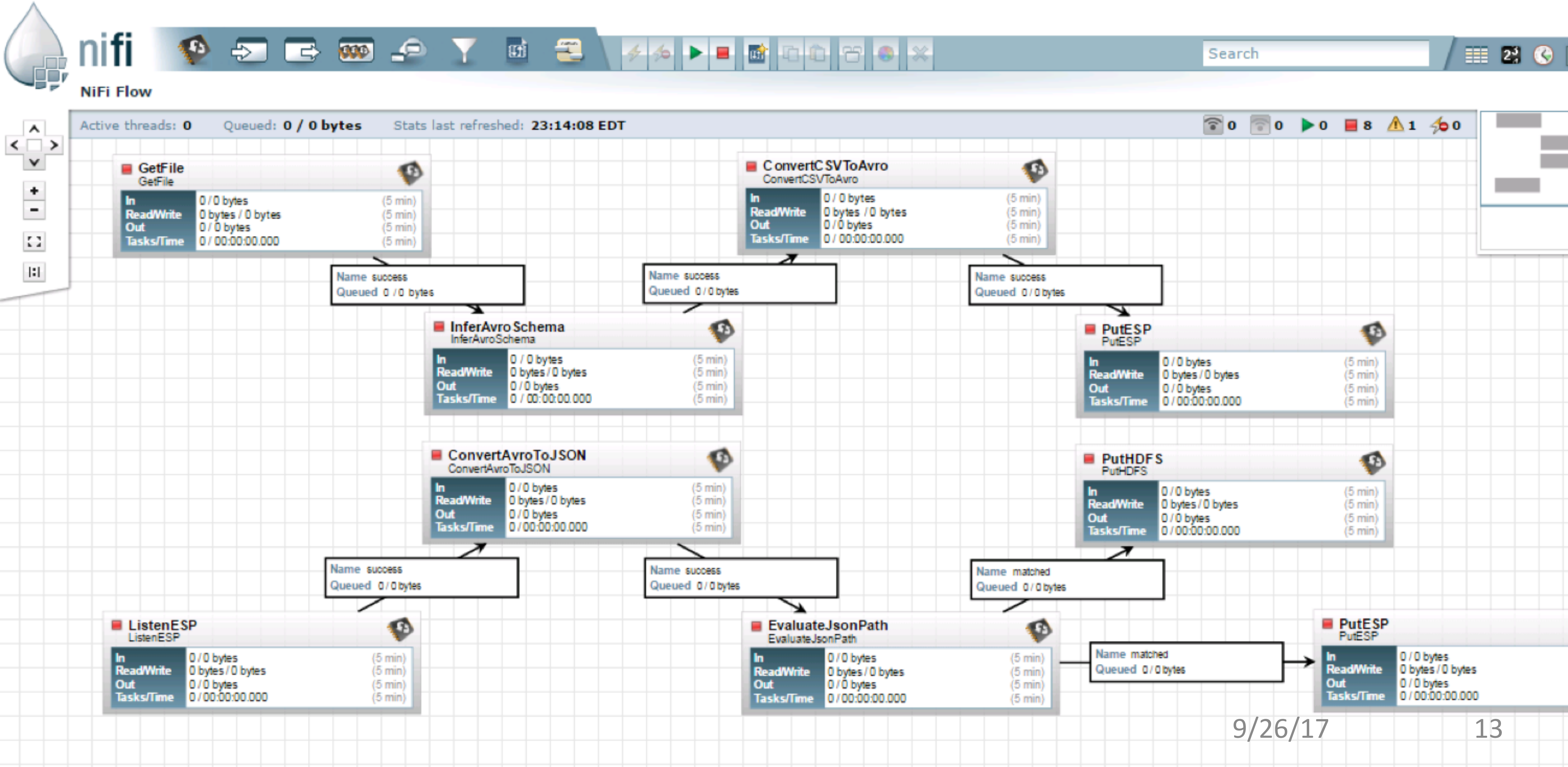
- QR Decomposition (QR) Reduce, Broadcast DAAL
- Neural Network AllReduce DAAL
- Covariance AllReduce DAAL
- Low Order Moments Reduce DAAL
- Naive Bayes Reduce DAAL
- Linear Regression Reduce DAAL
- Ridge Regression Reduce DAAL
- Multi-class Logistic Regression Regroup, Rotate, AllGather
- Random Forest AllReduce
- Principal Component Analysis (PCA) AllReduce DAAL

DAAL implies integrated with Intel DAAL Optimized Data Analytics Library (Runs on KNL!)

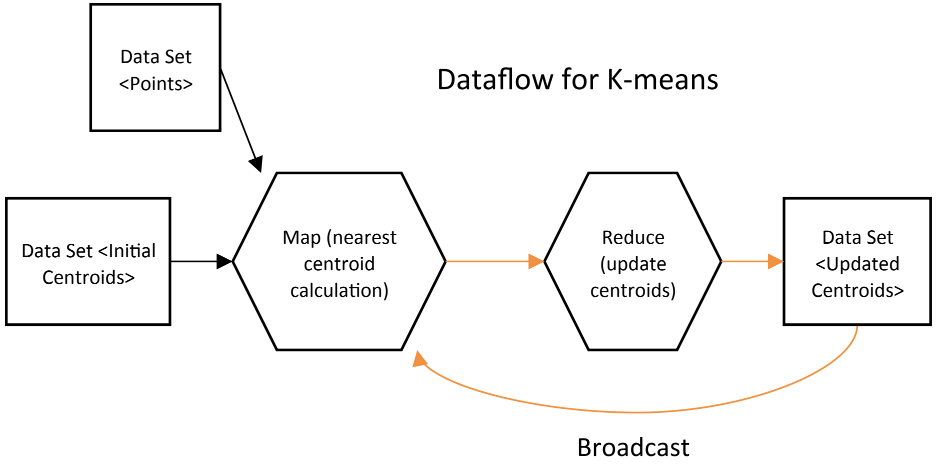
Coordination Points

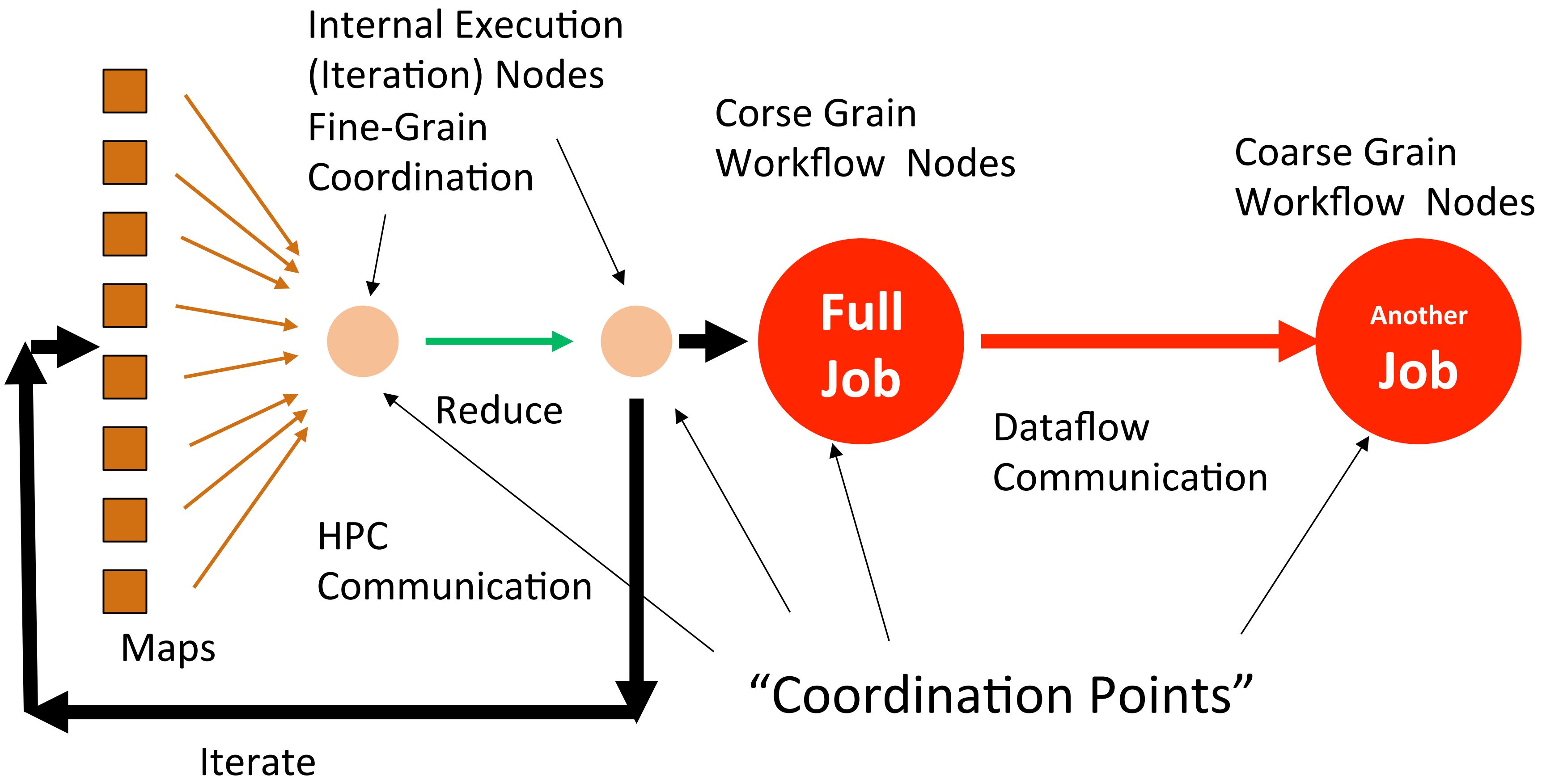
- There are in many approaches, “coordination points” that can be implicit or explicit
- Twister2 makes coordination points an important (first class) concept
  - Dataflow nodes in Heron, Flink, Spark, Naiad; we call these fine-grain data flow
  - Issuance of a Collective communication command in MPI
  - Start and End of a Parallel section in OpenMP
  - End of a job; we call these coarse-grain data flow nodes and these are seen in workflow systems such as Pegasus, Taverna, Kepler and NiFi (from Apache)
- Twister2 will allow users to specify the existence of a named coordination point and allow actions to be initiated
  - Produce an RDD style dataset from user specified
  - Launch new tasks as in Heron, Flink, Spark, Naiad
  - Change execution model as in OpenMP Parallel section

NiFi Workflow with Coarse Grain Coordination



K-means and Dataflow





Handling of State

- **State** is a key issue and handled differently in systems
- MPI Naiad, Storm, Heron have long running tasks that preserve state
  - MPI tasks stop at end of job
  - Naiad Storm Heron tasks change at (fine-grain) dataflow nodes but all tasks run forever
  - Spark and Flink tasks stop and refresh at dataflow nodes but preserve some state as RDD/datasets using in-memory databases
- All systems agree on actions at a coarse grain dataflow (at job level); only keep state by exchanging data.

Fault Tolerance and State

- Similar form of **check-pointing** mechanism is used already in HPC and Big Data
  - although HPC informal as doesn't typically specify as a dataflow graph
  - Flink and Spark do better than MPI due to use of **database** technologies; MPI is a bit harder due to richer state but there is an obvious integrated model using RDD type snapshots of MPI style jobs
- Checkpoint **after each stage of the dataflow graph**
  - Natural synchronization point
  - Let's allows user to choose when to checkpoint (not every stage)
  - Save state as user specifies; Spark just saves Model state which is insufficient for complex algorithms

Twister2 Components I

Area	Component	Implementation	Comments
Distributed Data API	Relaxed Distributed data set	Similar to Spark RDD	ETL type data applications; Streaming Backup for Fault Tolerance
	Streaming	Pub-Sub and Spouts as in Heron	API to pub-sub messages
	Data access	Access common data sources including file, connecting to message brokers etc.	All the above applications can use this base functionality
	Distributed Shared Memory	Similar to PGAS	Machine learning such as graph algorithms
Task API **FaaS API** **(Function as a Service)**	Dynamic Task Scheduling	Dynamic scheduling as in AMT	Some machine learning FaaS
	Static Task Scheduling	Static scheduling as in Flink & Heron	Streaming ETL data pipelines
	Task Execution	Thread based execution as seen in Spark, Flink, Naiad, OpenMP	Look at hybrid MPI/thread support available
	Task Graph	Twister2 Tasks similar to Naiad and Heron	
	Streaming and FaaS Events	Heron, OpenWhisk, Kafka/RabbitMQ	Classic Streaming Scaling of FaaS needs Research
	Elasticity	OpenWhisk	Needs experimentation
	Task migration	Monitoring of tasks and migrating tasks for better resource utilization	

Twister2 Components II

Area	Component	Implementation	Comments
Communication API	Messages	Heron	This is user level and could map to multiple communication
	Dataflow Communication	Fine-Grain Twister2 Dataflow communications: MPI,TCP and RMA Coarse grain Dataflow from NiFi, Kepler?	Streaming Machine learning ETL data pipelines
	BSP Communication Map-Collective	MPI Style communication Harp	Machine learning
Execution Model	Architecture	Spark, Flink	Container/Processes/Tasks=Threads
	Job Submit API Resource Scheduler	Pluggable architecture for any resource scheduler (Yarn, Mesos, Slurm)	All the above applications need this base functionality
	Dataflow graph analyzer & optimizer	Flink	Spark is dynamic and implicit
	Coordination Points Specification and Actions	Research based on MPI, Spark, Flink, NiFi (Kepler)	Synchronization Point. Backup to datasets Refresh Tasks
Security	Storage, Messaging, execution	Research	Crosses all Components

Summary of MPI in a HPC Cloud + Edge + Grid Environment

- We suggest value of an event driven computing model built around Cloud and HPC and spanning batch, streaming, and edge applications
  - Highly parallel on cloud; possibly sequential at the edge
- We have done a preliminary analysis of the different runtimes of MPI, Hadoop, Spark, Flink, Storm, Heron, Naiad, HPC Asynchronous Many Task(AMT)
- There are different technologies for different circumstances but can be unified by high level abstractions such as communication collectives
  - Obviously MPI best for parallel computing (by definition)
- Apache systems use dataflow communication which is natural for distributed systems but inevitably slow for classic parallel computing
  - No standard dataflow library (why?). **Add Dataflow primitives in MPI-4?**
- MPI could adopt some of tools of Big Data as in Coordination Points (dataflow nodes), State management with RDD (datasets)