Twister2: A High-Performance Big Data Programming Environment

HPBDC 2018: The 4th IEEE International Workshop on High-Performance Big Data, Deep Learning, and Cloud Computing

Geoffrey Fox, May 21, 2018
Judy Qiu, Supun Kamburugamuve
Department of Intelligent Systems Engineering
gcf@indiana.edu, http://www.dsc.soic.indiana.edu/, http://spidal.org/

Work with Shantenu Jha, Kannan Govindarajan, Pulasthi Wickramasinghe, Gurhan Gunduz, Ahmet Uyar

Abstract

- We analyse the components that are needed in programming environments for Big Data Analysis Systems with scalable HPC performance and the functionality of ABDS – the Apache Big Data Software Stack.
- One highlight is Harp-DAAL which is a machine library exploiting the Intel node library DAAL and HPC communication collectives within the Hadoop ecosystem.
- Another highlight is Twister2 which consists of a set of middleware components to support batch or streaming data capabilities familiar from Apache Hadoop, Spark, Heron and Flink but with high performance
- Twister2 covers bulk synchronous and data flow communication; task management as in Mesos, Yarn and Kubernetes; dataflow graph execution models; launching of the Harp-DAAL library; streaming and repository data access interfaces, in-memory databases and fault tolerance at dataflow nodes.
- Similar capabilities are available in current Apache systems but as integrated packages which do not allow needed customization for different application scenarios.

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 agreed 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 Apache Big Data Software (with no HPC)
  - This explains why Spark with poor GML performance can be so successful

Need a toolkit covering all applications with same API but different implementations

Difficulty in Parallelism

Size of Synchronization constraints

Loosely Coupled

Tightly Coupled

Commodity Clouds

HPC Clouds
High Performance Interconnect

HPC Clouds/Supercomputers
Memory access also critical

Size of Disk I/O

MapReduce as in scalable databases

Global Machine Learning e.g. parallel clustering

Deep Learning

Unstructured Adaptive Sparsity
Medium size Jobs

Pleasingly Parallel
Often independent events

LDA

Graph Analytics e.g. subgraph mining

Current major Big Data category

Linear Algebra at core (typically not sparse)

Large scale simulations

Parameter sweep simulations

Structured Adaptive Sparsity
Huge Jobs

Spectrum of Applications and Algorithms

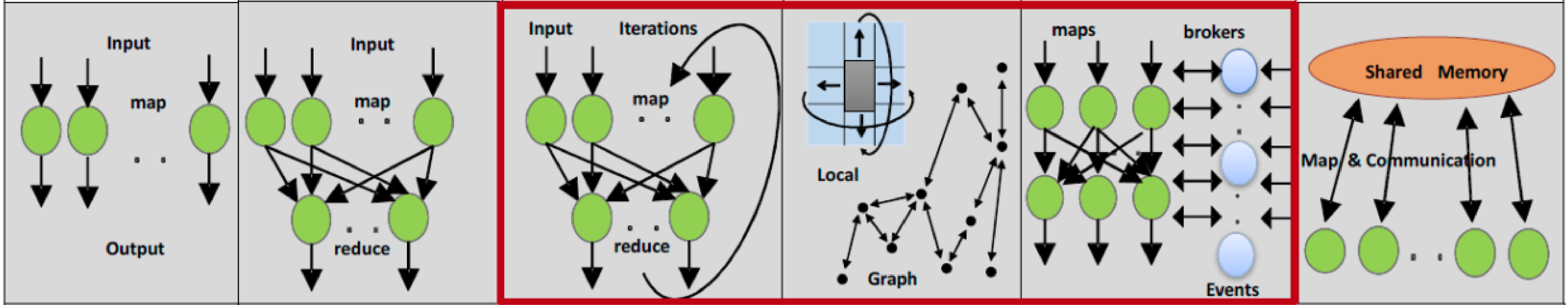
There is also distribution seen in grid/edge computing

Exascale Supercomputers

Need a toolkit covering 5 main paradigms with same API but different implementations

Six Computation Paradigms for Data Analytics

Note Problem and System Architecture as efficient execution says they must match



(1) Map Only Pleasingly Parallel	(2) Classic Map-Reduce	(3) Iterative Map Reduce or Map-Collective	(4) Point to Point or Map-Communication	(5) Map-Streaming	(6) Shared memory Map-Communication
- BLAST Analysis - Local Machine Learning - Pleasingly Parallel	- High Energy Physics (HEP) Histograms, - Web search - Recommender Engines	- Expectation Maximization - Clustering - Linear Algebra - PageRank	- Classic MPI - PDE Solvers and Particle Dynamics - Graph	- Streaming images from Synchrotron sources, Telescopes, Internet of Things	- Difficult to parallelize - asynchronous parallel Graph
		Global Machine	Learning		

Classic Cloud Workload

These 3 are focus of Twister2 but we need to preserve capability on first 2 paradigms

Comparing Spark, Flink and MPI

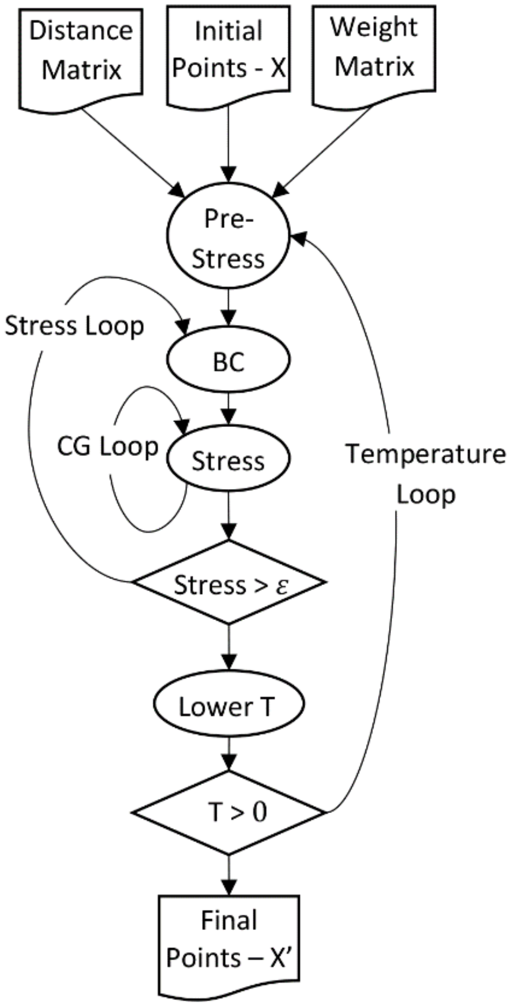
- On Global Machine Learning GML.

Machine Learning with MPI, Spark and Flink

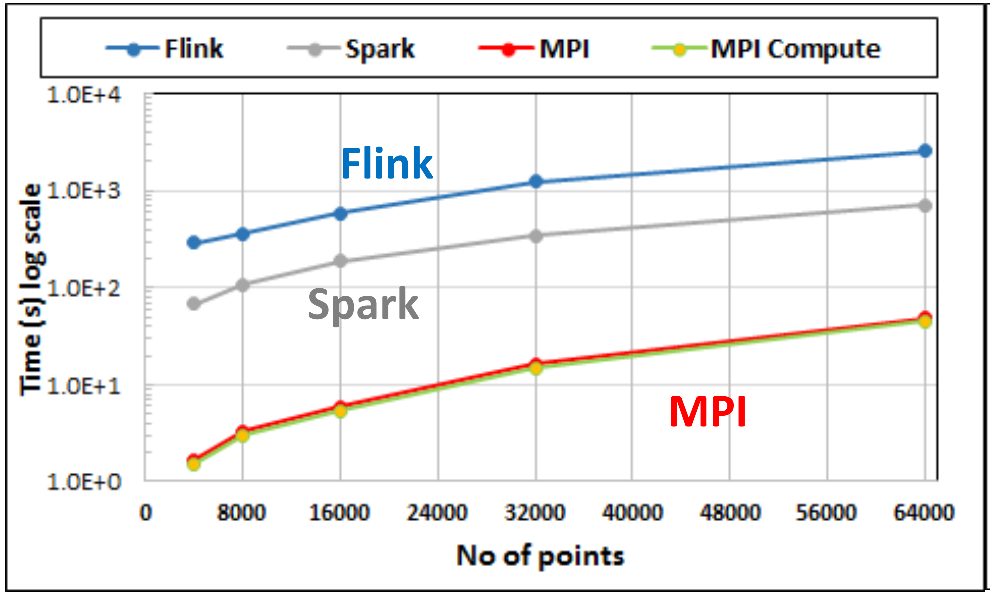
- Three algorithms implemented in three runtimes
  - Multidimensional Scaling (MDS)
  - Terasort
  - K-Means (drop as no time and looked at later)
- Implementation in Java
  - MDS is the most complex algorithm - three nested parallel loops
  - K-Means - one parallel loop
  - Terasort - no iterations

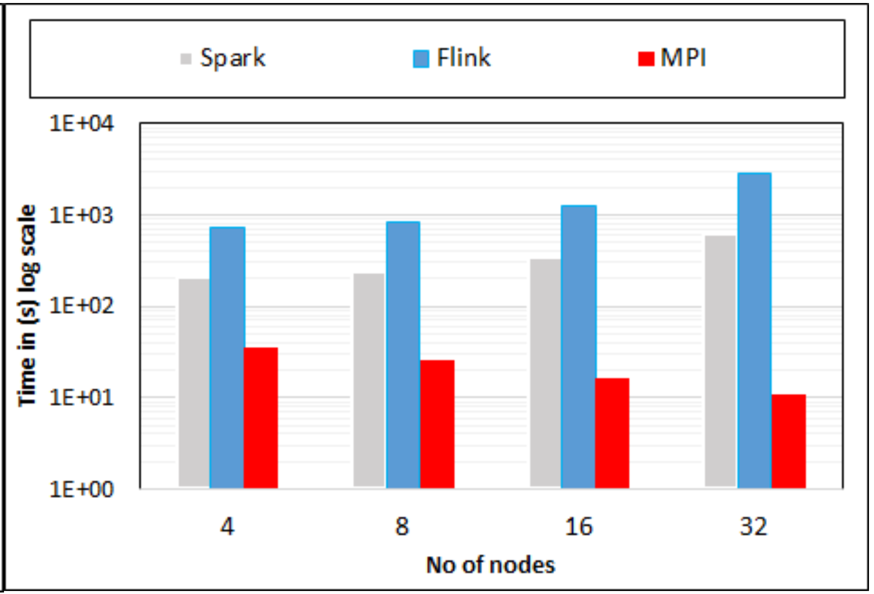
- With care, Java performance ~ C performance
- Without care, Java performance << C performance (details omitted)

Multidimensional Scaling: 3 Nested Parallel Sections



Kmeans also bad – see later





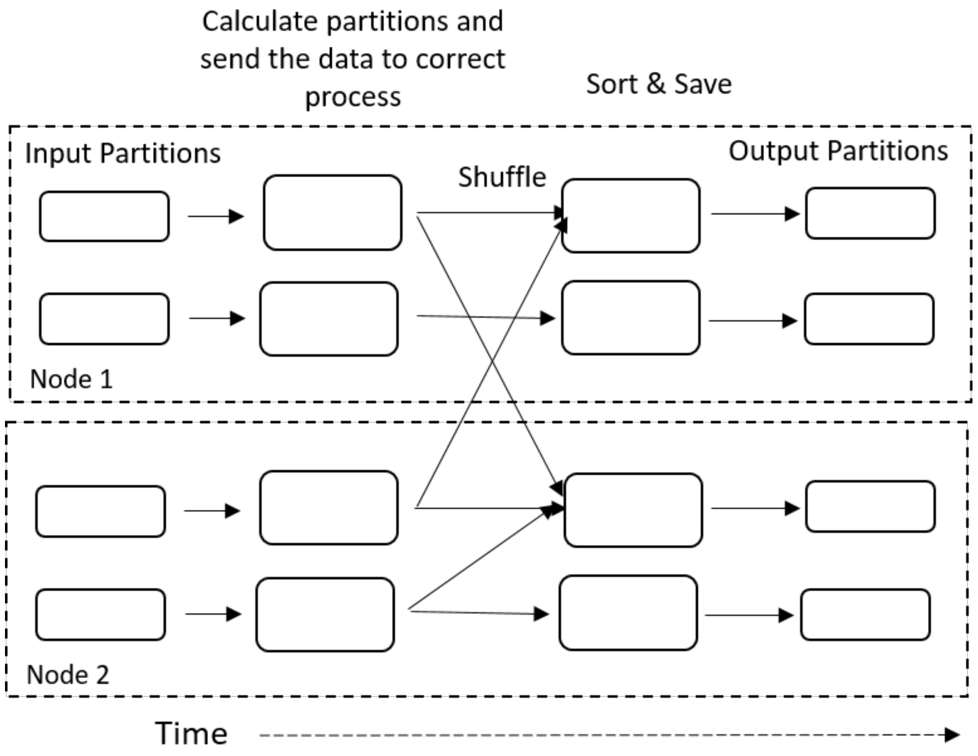
MPI Factor of 20-200 Faster than Spark/Flink

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

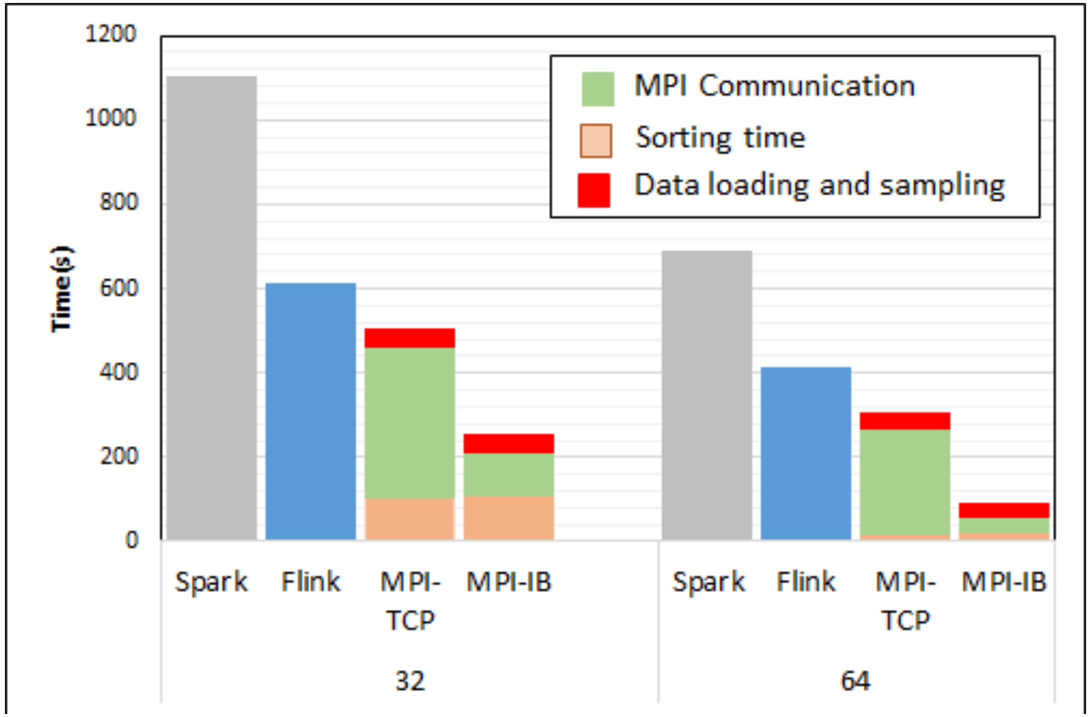
MDS execution time with 32000 points on **varying number of nodes**. Each node runs 20 parallel tasks
Spark, Flink No Speedup

Terasort

Sorting 1TB of data records



Partition the data using a sample and regroup



Terasort execution time in 64 and 32 nodes. Only MPI shows the sorting time and communication time as other two frameworks doesn't provide a clear method to accurately measure them. Sorting time includes data save time.
MPI-IB - MPI with Infiniband

Software
HPC-ABDS
HPC-FaaS

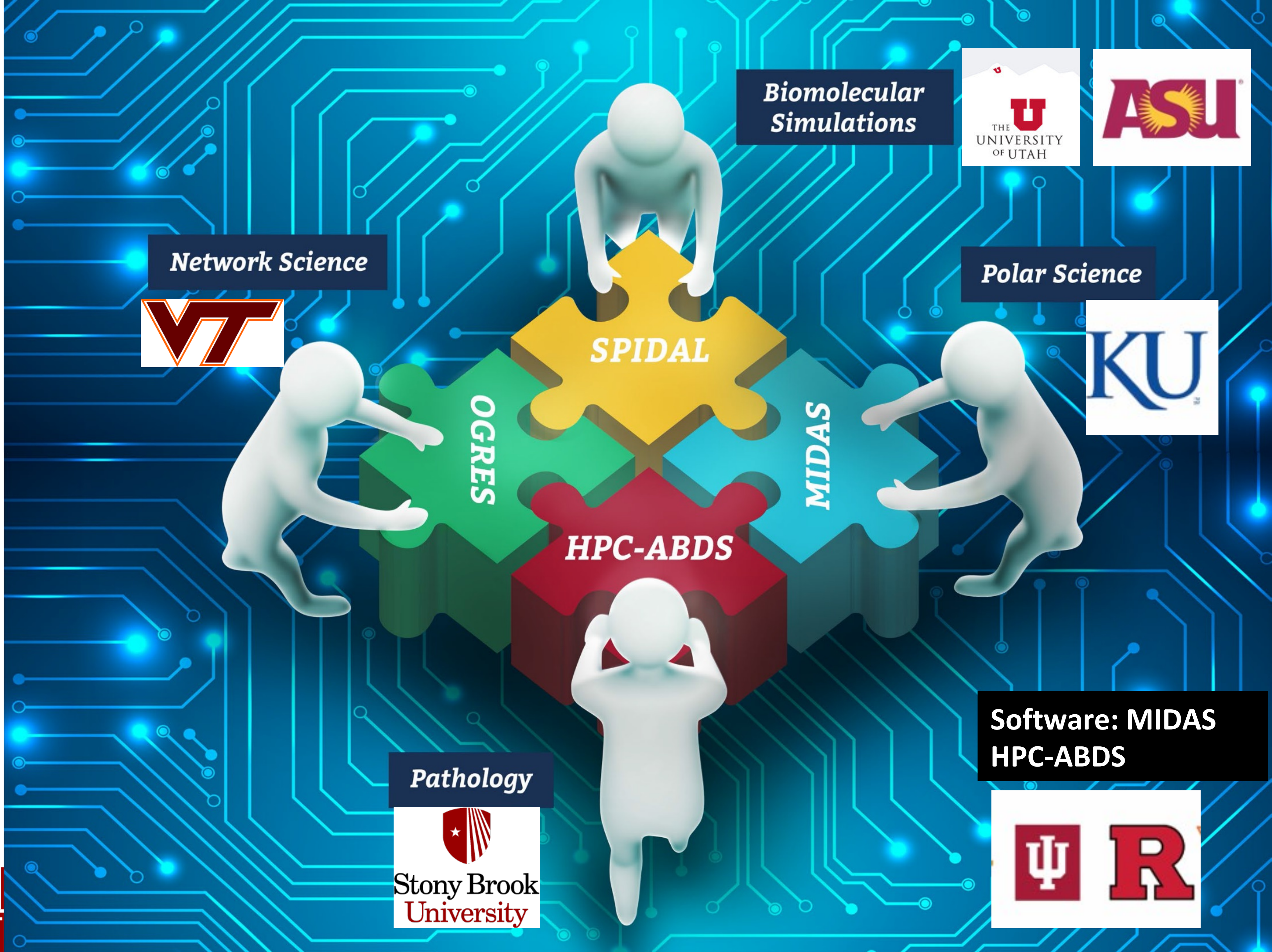
NSF 1443054: CIF21 DIBBs: Middleware and High Performance Analytics Libraries for Scalable Data Science

Ogres Application Analysis

HPC-ABDS and HPC-FaaS Software

Harp and Twister2 Building Blocks

SPIDAL Data Analytics Library



HPC-ABDS

Integrated wide range of HPC and Big Data technologies.

I gave up updating list in January 2016!

INDIANA UNIVERSITY
SCHOOL OF I

Kaleidoscope of (Apache) Big Data Stack (ABDS) and HPC Technologies

Cross-Cutting Functions

1) **Message and Data Protocols:** Avro, Thrift, Protobuf

2) **Distributed Coordination:** Google Chubby, Zookeeper, Giraffe, JGroups

3) **Security & Privacy:** InCommon, Eduroam OpenStack Keystone, LDAP, Sentry, Sqrrl, OpenID, SAML OAuth

4) **Monitoring:** Ambari, Ganglia, Nagios, Inca

21 layers
Over 350 Software Packages

January 29 2016

17) **Workflow-Orchestration:** ODE, ActiveBPEL, Airavata, Pegasus, Kepler, Swift, Taverna, Triana, Trident, BioKepler, Galaxy, IPython, Dryad, Naiad, Oozie, Tez, Google FlumeJava, Crunch, Cascading, Scalding, e-Science Central, Azure Data Factory, Google Cloud Dataflow, NiFi (NSA), Jitterbit, Talend, Pentaho, Apatar, Docker Compose, KeystoneML

16) **Application and Analytics:** Mahout , MLlib , MLbase, DataFu, R, pbdR, Bioconductor, ImageJ, OpenCV, Scalapack, PetSc, PLASMA MAGMA, Azure Machine Learning, Google Prediction API & Translation API, mlpy, scikit-learn, PyBrain, CompLearn, DAAL(Intel), Caffe, Torch, Theano, DL4j, H2O, IBM Watson, Oracle PGX, GraphLab, GraphX, IBM System G, GraphBuilder(Intel), TinkerPop, Parasol, Dream:Lab, Google Fusion Tables, CINET, NWB, Elasticsearch, Kibana, Logstash, Graylog, Splunk, Tableau, D3.js, three.js, Potree, DC.js, TensorFlow, CNTK

15B) **Application Hosting Frameworks:** Google App Engine, AppScale, Red Hat OpenShift, Heroku, Aerobatic, AWS Elastic Beanstalk, Azure, Cloud Foundry, Pivotal, IBM BlueMix, Ninefold, Jelastic, Stackato, appfog, CloudBees, Engine Yard, CloudControl, dotCloud, Dokku, OSGi, HUBzero, OODT, Agave, Atmosphere

15A) **High level Programming:** Kite, Hive, HCatalog, Tajo, Shark, Phoenix, Impala, MRQL, SAP HANA, HadoopDB, PolyBase, Pivotal HD/Hawq, Presto, Google Dremel, Google BigQuery, Amazon Redshift, Drill, Kyoto Cabinet, Pig, Sawzall, Google Cloud DataFlow, Summingbird

14B) **Streams:** Storm, S4, Samza, Granules, Neptune, Google MillWheel, Amazon Kinesis, LinkedIn, Twitter Heron, Databus, Facebook Puma/Ptail/Scribe/ODS, Azure Stream Analytics, Floe, Spark Streaming, Flink Streaming, DataTurbine

14A) **Basic Programming model and runtime, SPMD, MapReduce:** Hadoop, Spark, Twister, MR-MPI, Stratosphere (Apache Flink), Reef, Disco, Hama, Giraph, Pregel, Pegasus, Ligra, GraphChi, Galois, Medusa-GPU, MapGraph, Totem

13) **Inter process communication Collectives, point-to-point, publish-subscribe:** MPI, HPX-5, Argo BEAST HPX-5 BEAST PULSAR, Harp, Netty, ZeroMQ, ActiveMQ, RabbitMQ, NaradaBrokering, QPid, Kafka, Kestrel, JMS, AMQP, Stomp, MQTT, Marionette Collective, **Public Cloud:** Amazon SNS, Lambda, Google Pub Sub, Azure Queues, Event Hubs

12) **In-memory databases/caches:** Gora (general object from NoSQL), Memcached, Redis, LMDB (key value), Hazelcast, Ehcache, Infinispan, VoltDB, H-Store

12) **Object-relational mapping:** Hibernate, OpenJPA, EclipseLink, DataNucleus, ODBC/JDBC

12) **Extraction Tools:** UIMA, Tika

11C) **SQL(NewSQL):** Oracle, DB2, SQL Server, SQLite, MySQL, PostgreSQL, CUBRID, Galera Cluster, SciDB, Rasdaman, Apache Derby, Pivotal Greenplum, Google Cloud SQL, Azure SQL, Amazon RDS, Google F1, IBM dashDB, N1QL, BlinkDB, Spark SQL

11B) **NoSQL:** Lucene, Solr, Solandra, Voldemort, Riak, ZHT, Berkeley DB, Kyoto/Tokyo Cabinet, Tycoon, Tyrant, MongoDB, Espresso, CouchDB, Couchbase, IBM Cloudant, Pivotal Gemfire, HBase, Google Bigtable, LevelDB, Megastore and Spanner, Accumulo, Cassandra, RYA, Sqrrl, Neo4J, graphdb, Yarcdata, AllegroGraph, Blazegraph, Facebook Tao, Titan:db, Jena, Sesame
Public Cloud: Azure Table, Amazon Dynamo, Google DataStore

11A) **File management:** iRODS, NetCDF, CDF, HDF, OPeNDAP, FITS, RCFile, ORC, Parquet

10) **Data Transport:** BitTorrent, HTTP, FTP, SSH, Globus Online (GridFTP), Flume, Sqoop, Pivotal GPLOAD/GPFDIST

9) **Cluster Resource Management**: Mesos, Yarn, Helix, Llama, Google Omega, Facebook Corona, Celery, HTCondor, SGE, OpenPBS, Moab, Slurm, Torque, Globus Tools, Pilot Jobs

8) **File systems:** HDFS, Swift, Haystack, f4, Cinder, Ceph, FUSE, Gluster, Lustre, GPFS, GFFS
Public Cloud: Amazon S3, Azure Blob, Google Cloud Storage

7) **Interoperability:** Libvirt, Libcloud, JClouds, TOSCA, OCCI, CDMI, Whirr, Saga, Genesis

6) **DevOps:** Docker (Machine, Swarm), Puppet, Chef, Ansible, SaltStack, Boto, Cobbler, Xcat, Razor, CloudMesh, Juju, Foreman, OpenStack Heat, Sahara, Rocks, Cisco Intelligent Automation for Cloud, Ubuntu MaaS, Facebook Tupperware, AWS OpsWorks, OpenStack Ironic, Google Kubernetes, Buildstep, Gitreceive, OpenTOSCA, Winery, CloudML, Blueprints, Terraform, DevOpSlang, Any2Api

5) **IaaS Management from HPC to hypervisors:** Xen, KVM, QEMU, Hyper-V, VirtualBox, OpenVZ, LXC, Linux-Vserver, OpenStack, OpenNebula, Eucalyptus, Nimbus, CloudStack, CoreOS, rkt, VMware ESXi, vSphere and vCloud, Amazon, Azure, Google and other public Clouds
Networking: Google Cloud DNS, Amazon Route 53

Different choices in software systems in Clouds and HPC. HPC-ABDS takes cloud software augmented by HPC when needed to improve performance

16 of 21 layers plus languages

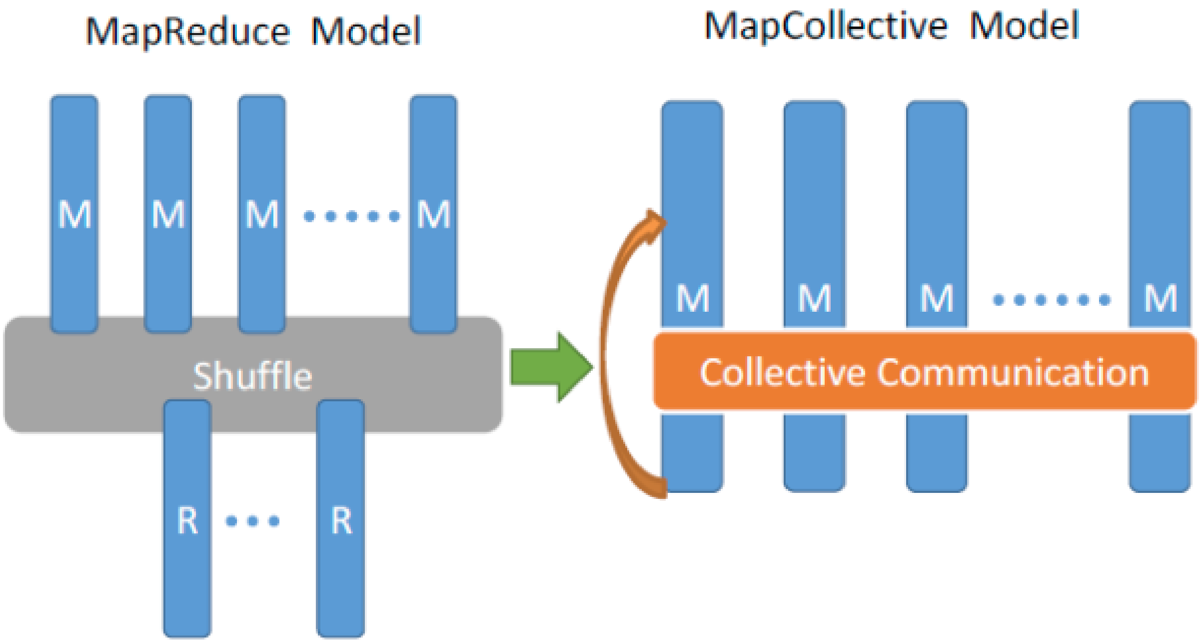
	HPC-ABDS Stack	Equivalent HPC, Cluster
17. Orchestration	Beam	Kepler, Pegasus, Taverna
16. Libraries	SPIDAL, TensorFlow, Python	ScaLAPACK, PETSc, Matlab
15A. High Level Programming	Pig, Hive, Drill	Domain-specific Languages
15B. Platform as a Service	Twister2	XSEDE Software Stack
Languages	Java, Erlang, Scala, SQL, SPARQL, Python	Fortran, C/C++, Python
14B. Streaming	Heron	
13,14A. Parallel Runtime	Hadoop, Harp	MPI OpenMP CUDA, OpenCL Exascale Runtime
2. Coordination	Zookeeper	
12. Caching	Memcached	
11. Data Management	Hbase, MongoDB	iRODS
10. Data Transfer	Globus, HTTP, Pub-Sub	Globus
9. Scheduling	Yarn, Mesos	Slurm
8. File Systems	HDFS, Object Stores	Lustre
1, 11A Formats	Thrift, Protobuf	FITS, HDF
5. IaaS	Docker, Serverless	Linux, Bare-metal, SR-IOV
Infrastructure	HPC CLOUDS	Clusters, SUPERCOMPUTERS

Big Data Simulation Clouds HPC Integrated Software Stack

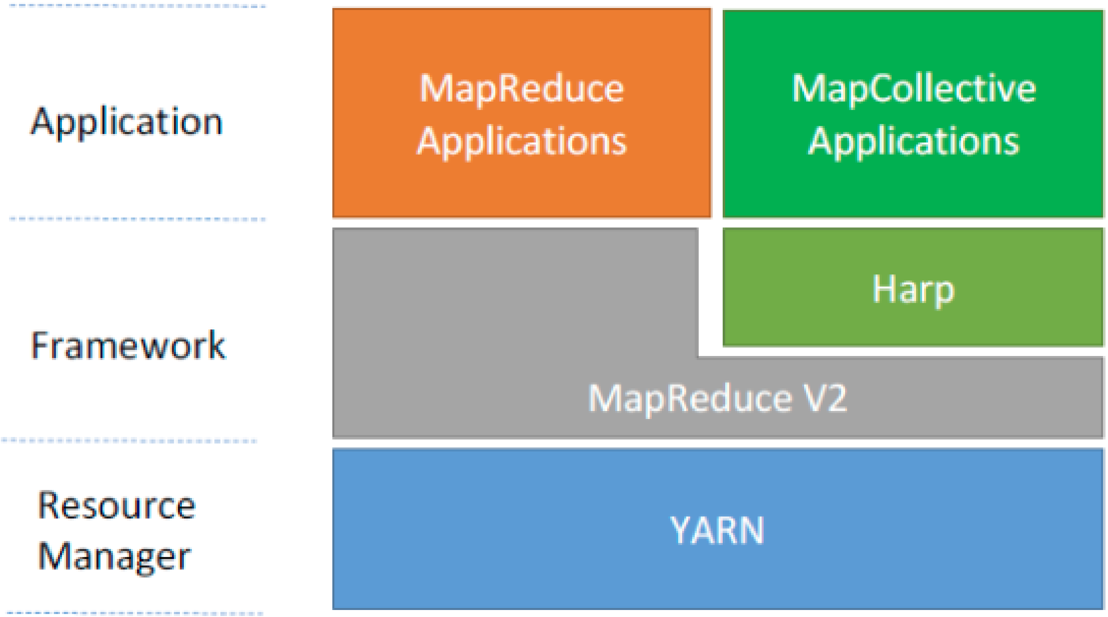
Harp Plugin for Hadoop: Important part of Twister2

Work of Judy Qiu

Parallelism Model



Architecture

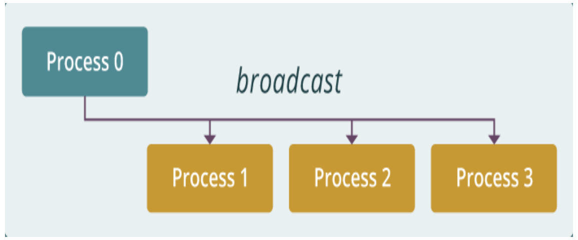


Harp is an open-source project developed at Indiana University [6], it has:

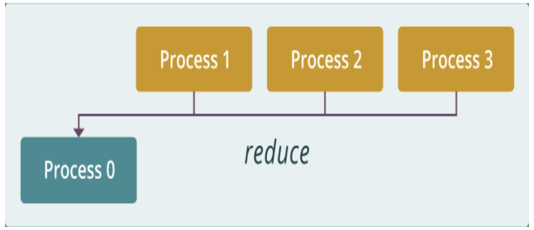
- MPI-like **collective communication** operations that are highly optimized for big data problems.
- Harp has efficient and innovative **computation models** for different machine learning problems.

[6] Harp project. Available at https://dsc-spidal.github.io/harp

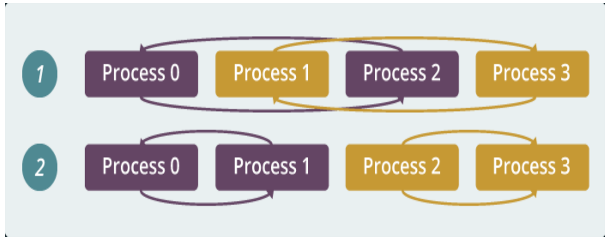
Run time software for Harp



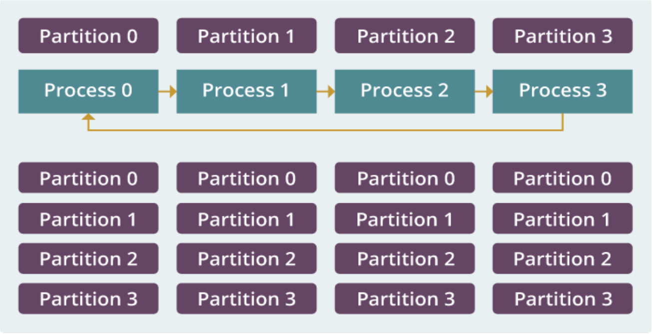
broadcast



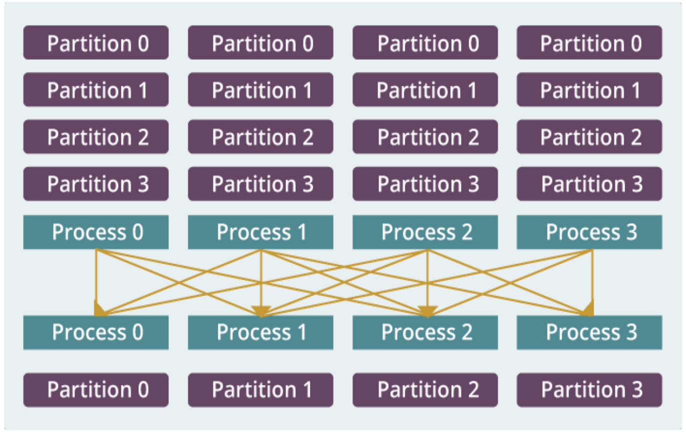
reduce

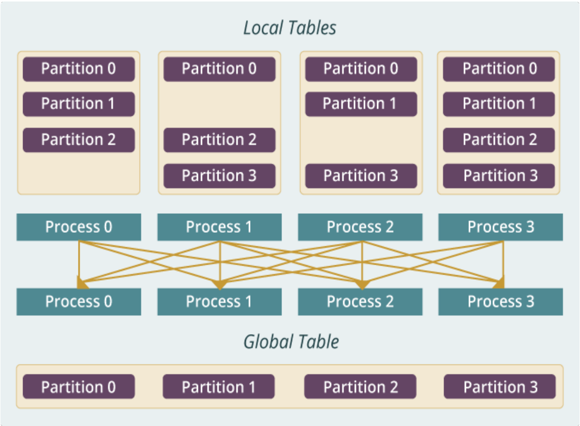


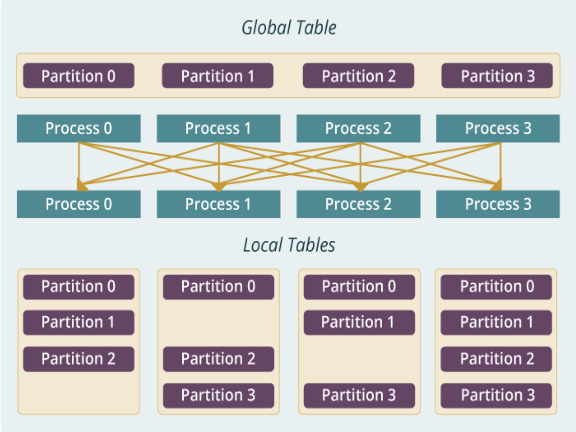
allreduce

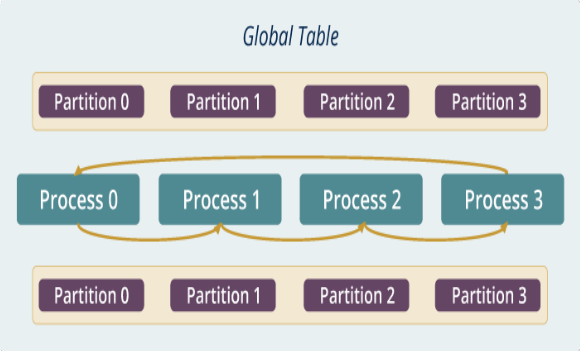


allgather





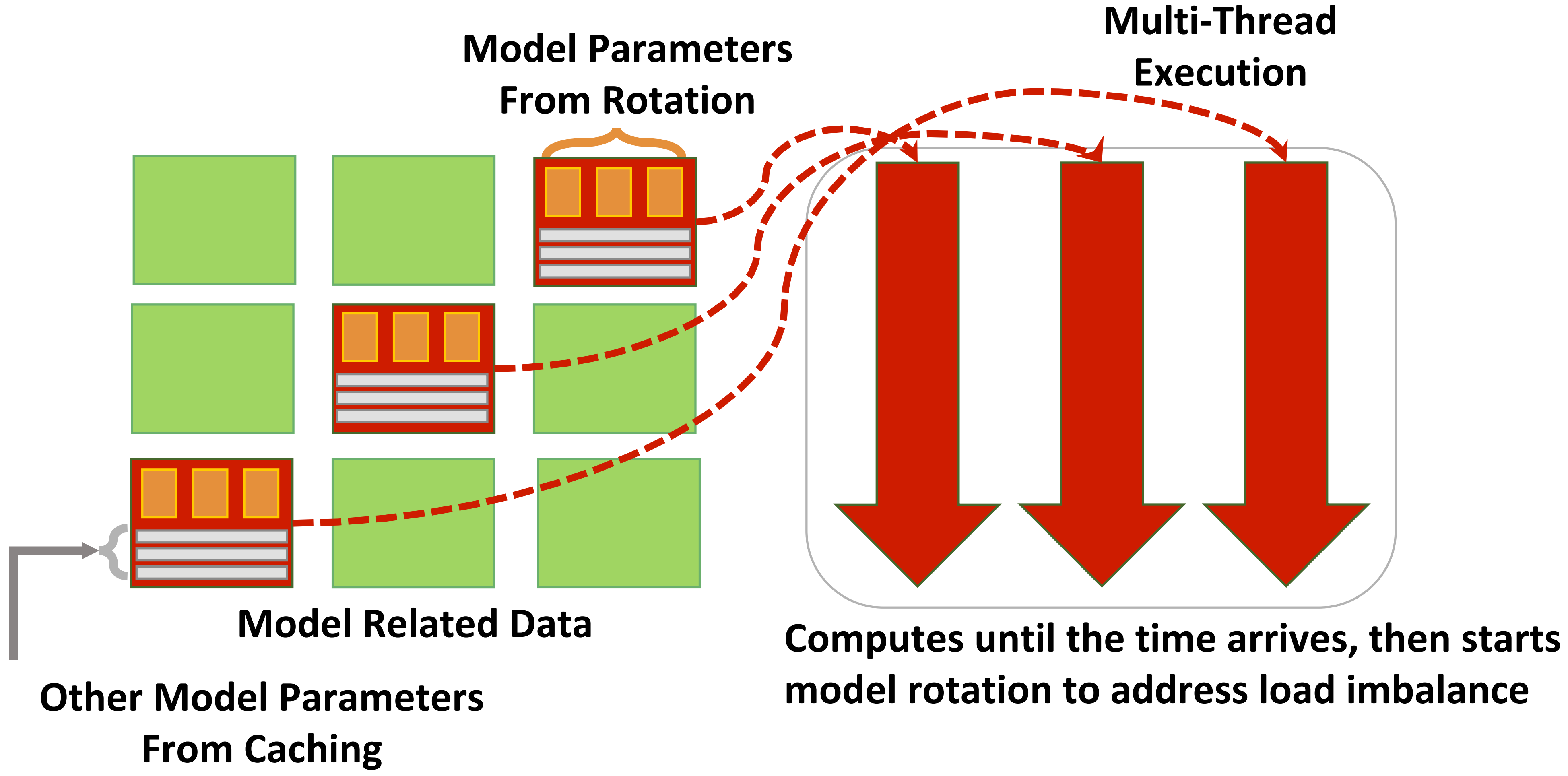




regroup **push & pull** **rotate**

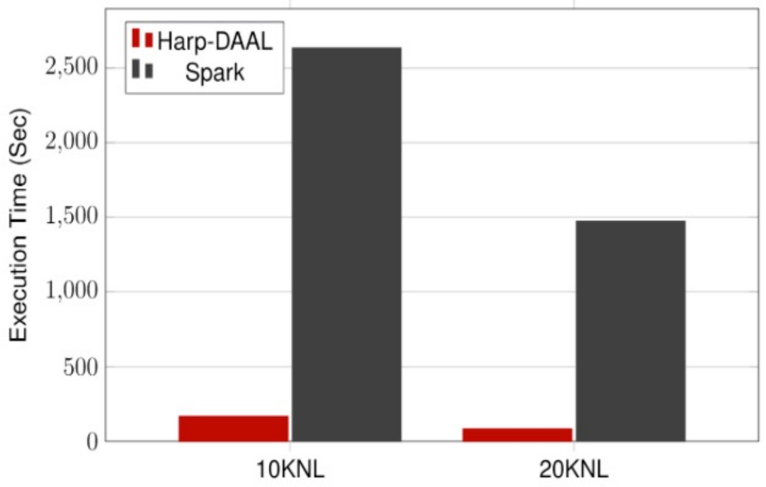
Map Collective Run time merges MapReduce and HPC

Dynamic Rotation Control for Latent Dirichlet Allocation and Matrix Factorization SGD (stochastic gradient descent)



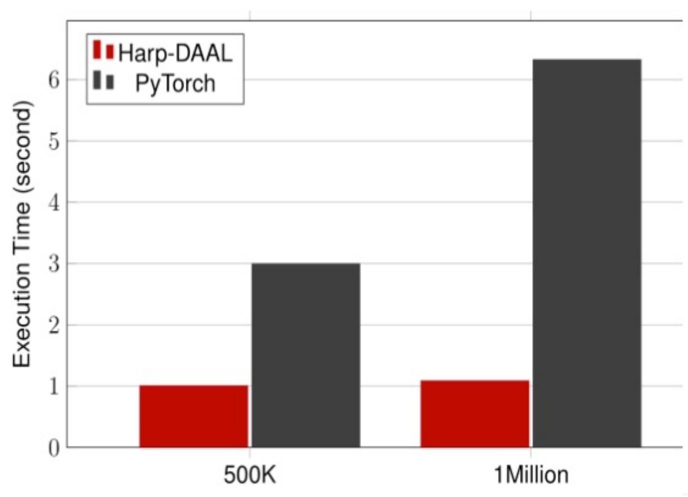
Harp v. Spark Harp v. Torch Harp v. MPI

Performance Comparison



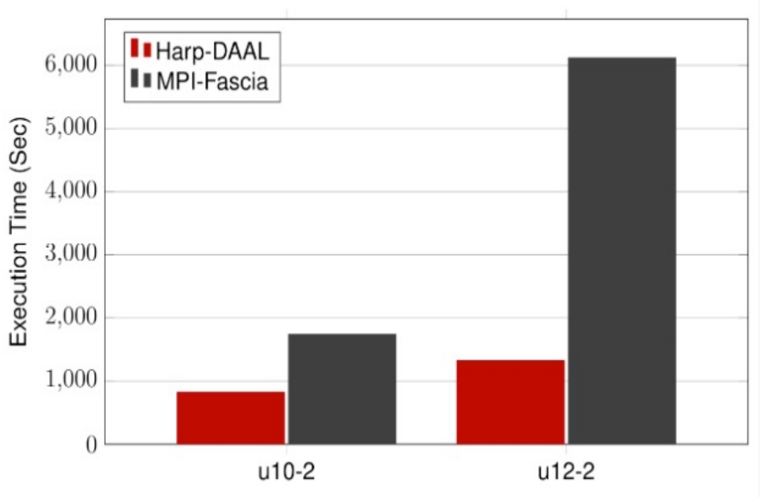
K means

- Datasets: 5 million points, 10 thousand centroids, 10 feature dimensions
- 10 to 20 nodes of Intel KNL7250 processors
- Harp-DAAL has 15x speedups over Spark MLlib



PCA

- Datasets: 500K or 1 million data points of feature dimension 300
- Running on single KNL 7250 (Harp-DAAL) vs. single K80 GPU (PyTorch)
- Harp-DAAL achieves 3x to 6x speedups

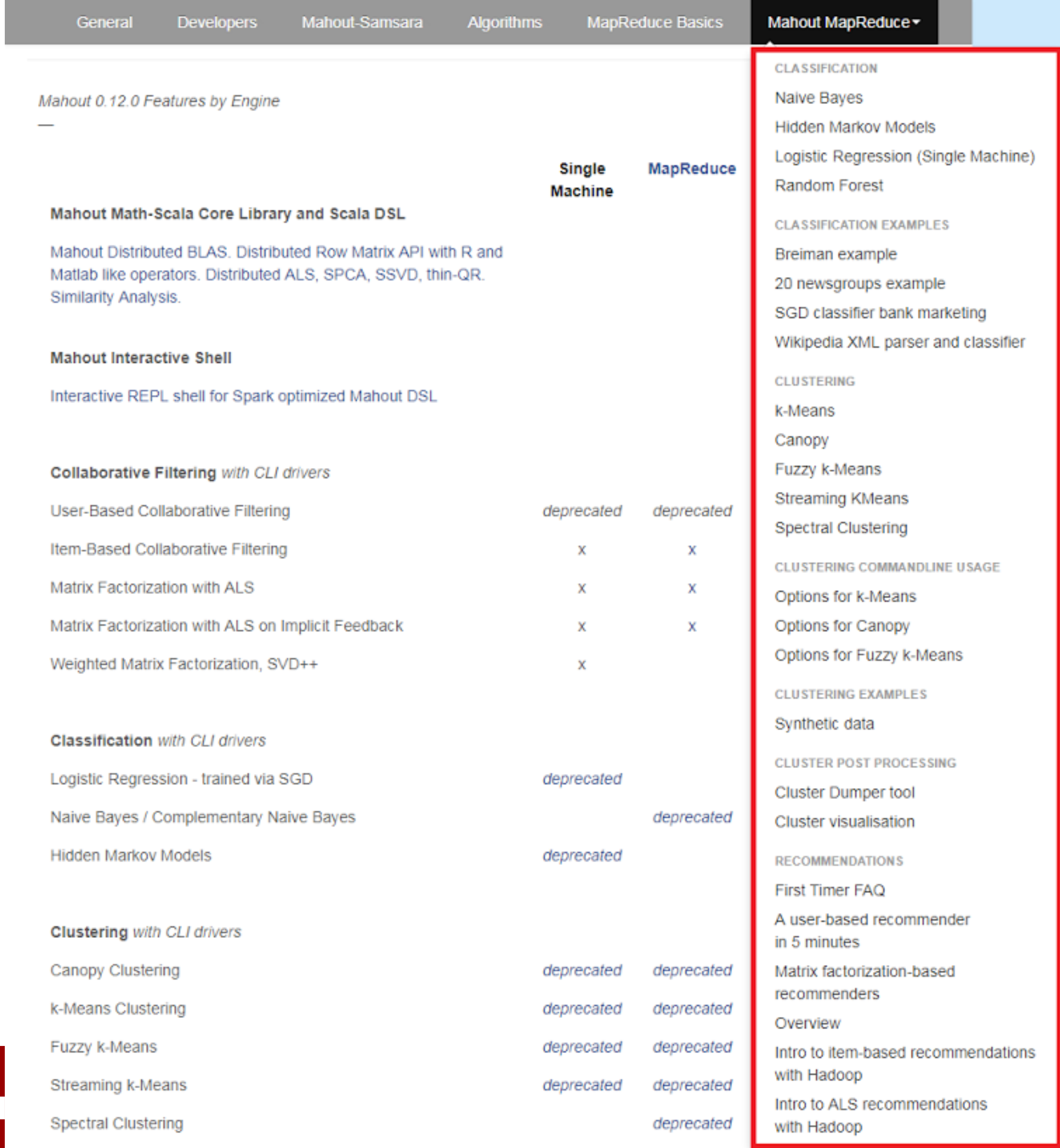


Subgraph

- Datasets: Twitter with 44 million vertices, 2 billion edges, subgraph templates of 10 to 12 vertices
- 25 nodes of Intel Xeon E5 2670
- Harp-DAAL has 2x to 5x speedups over state-of-the-art MPI-Fascia solution

Mahout and SPIDAL

- Mahout was Hadoop machine learning library but largely abandoned as Spark outperformed Hadoop
- SPIDAL outperforms Spark MLlib and Flink due to better communication and better dataflow or BSP communication.
- Has Harp-(DAAL) optimized machine learning interface
- SPIDAL also has community algorithms
  - Biomolecular Simulation
  - Graphs for Network Science
  - Image processing for pathology and polar science


General | Developers | Mahout-Samsara | Algorithms | MapReduce Basics | Mahout MapReduce

Mahout 0.12.0 Features by Engine

	Single Machine	MapReduce
Mahout Math-Scala Core Library and Scala DSL		
Mahout Distributed BLAS. Distributed Row Matrix API with R and Matlab like operators. Distributed ALS, SPCA, SSVD, thin-QR. Similarity Analysis.		
Mahout Interactive Shell		
Interactive REPL shell for Spark optimized Mahout DSL		
Collaborative Filtering *with CLI drivers*		
User-Based Collaborative Filtering	*deprecated*	*deprecated*
Item-Based Collaborative Filtering	x	x
Matrix Factorization with ALS	x	x
Matrix Factorization with ALS on Implicit Feedback	x	x
Weighted Matrix Factorization, SVD++	x	
Classification *with CLI drivers*		
Logistic Regression - trained via SGD	*deprecated*	
Naive Bayes / Complementary Naive Bayes		*deprecated*
Hidden Markov Models	*deprecated*	
Clustering *with CLI drivers*		
Canopy Clustering	*deprecated*	*deprecated*
k-Means Clustering	*deprecated*	*deprecated*
Fuzzy k-Means	*deprecated*	*deprecated*
Streaming k-Means	*deprecated*	*deprecated*
Spectral Clustering		*deprecated*

CLASSIFICATION
Naive Bayes
Hidden Markov Models
Logistic Regression (Single Machine)
Random Forest
CLASSIFICATION EXAMPLES
Breiman example
20 newsgroups example
SGD classifier bank marketing
Wikipedia XML parser and classifier
CLUSTERING
k-Means
Canopy
Fuzzy k-Means
Streaming KMeans
Spectral Clustering
CLUSTERING COMMANDLINE USAGE
Options for k-Means
Options for Canopy
Options for Fuzzy k-Means
CLUSTERING EXAMPLES
Synthetic data
CLUSTER POST PROCESSING
Cluster Dumper tool
Cluster visualisation
RECOMMENDATIONS
First Timer FAQ
A user-based recommender in 5 minutes
Matrix factorization-based recommenders
Overview
Intro to item-based recommendations with Hadoop
Intro to ALS recommendations with Hadoop

Qiu 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 (Deterministic Annealing)** 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 on node with Intel DAAL Optimized Data Analytics Library

Twister

Iterative MapReduce

http://www.iterativemapreduce.org/

Implementing Twister2 in detail I

This breaks rule from 2012-2017 of not “competing” with but rather “enhancing” Apache

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

- Analyze the **runtime of existing systems**
  - Hadoop, Spark, Flink, Pregel Big Data Processing
  - OpenWhisk and commercial FaaS
  - Storm, Heron, Apex Streaming Dataflow
  - Kepler, Pegasus, NiFi workflow systems
  - Harp Map-Collective, MPI and HPC AMT runtime like DARMA
  - And approaches such as GridFTP and CORBA/HLA (!) for wide area data links
- 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 (unclear) requirements

http://www.iterativemapreduce.org/

Integrating HPC and Apache Programming Environments

- **Harp-DAAL** with a kernel Machine Learning library exploiting the Intel node library DAAL and HPC communication collectives within the Hadoop ecosystem. The broad applicability of Harp-DAAL is supporting all 5 classes of data-intensive computation, from pleasingly parallel to machine learning and simulations.
- **Twister2** is a toolkit of components that can be packaged in different ways
  - Integrated batch or streaming data capabilities familiar from Apache Hadoop, Spark, Heron and Flink but with high performance.
  - Separate bulk synchronous and data flow communication;
  - Task management as in Mesos, Yarn and Kubernetes
  - Dataflow graph execution models
  - Launching of the Harp-DAAL library
  - Streaming and repository data access interfaces,
  - In-memory databases and fault tolerance at dataflow nodes. (use RDD to do classic checkpoint-restart)

Approach

- Clearly define and develop functional layers (using existing technology when possible)
- Develop layers as independent components
- Use **interoperable** common abstractions but multiple **polymorphic** implementations.
- Allow users to pick and choose according to requirements such as
  - Communication + Data Management
  - Communication + Static graph
- Use HPC features when possible

Twister2 Components I

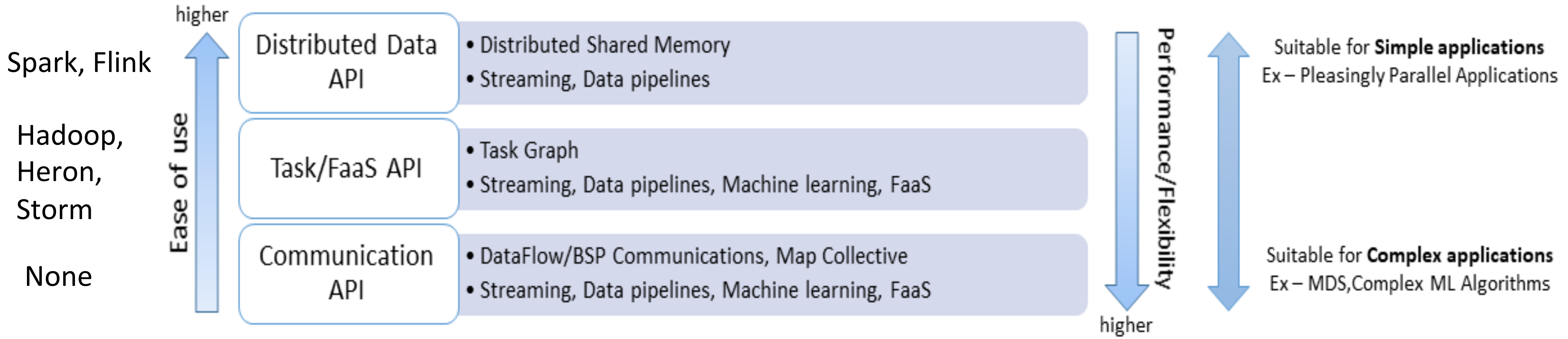
Area	Component	Implementation	Comments: User API
Architecture Specification	Coordination Points	State and Configuration Management; Program, Data and Message Level	Change execution mode; save and reset state
	Execution Semantics	Mapping of Resources to Bolts/Maps in Containers, Processes, Threads	Different systems make different choices - why?
	Parallel Computing	Spark Flink Hadoop Pregel MPI modes	Owner Computes Rule
Job Submission	(Dynamic/Static) Resource Allocation	Plugins for Slurm, Yarn, Mesos, Marathon, Aurora	Client API (e.g. Python) for Job Management
Task System	Task migration	Monitoring of tasks and migrating tasks for better resource utilization	Task-based programming with Dynamic or Static Graph API; FaaS API; Support accelerators (CUDA,KNL)
	Elasticity	OpenWhisk	
	Streaming and FaaS Events	Heron, OpenWhisk, Kafka/RabbitMQ	
	Task Execution	Process, Threads, Queues	
	Task Scheduling	Dynamic Scheduling, Static Scheduling, Pluggable Scheduling Algorithms	
	Task Graph	Static Graph, Dynamic Graph Generation	

Twister2 Components II

Area	Component	Implementation	Comments
Communication API	Messages	Heron	This is user level and could map to multiple communication systems
	Dataflow Communication	Fine-Grain Twister2 Dataflow communications: MPI,TCP and RMA Coarse grain Dataflow from NiFi, Kepler?	Streaming, ETL data pipelines; Define new Dataflow communication API and library
	BSP Communication Map-Collective	Conventional MPI, Harp	MPI Point to Point and Collective API
Data Access	Static (Batch) Data	File Systems, NoSQL, SQL	Data API
	Streaming Data	Message Brokers, Spouts	
Data Management	Distributed Data Set	Relaxed Distributed Shared Memory(immutable data), Mutable Distributed Data	Data Transformation API; Spark RDD, Heron Streamlet
Fault Tolerance	Check Pointing	Upstream (streaming) backup; Lightweight; Coordination Points; Spark/Flink, MPI and Heron models	Streaming and batch cases distinct; Crosses all components
Security	Storage, Messaging, execution	Research needed	Crosses all Components

Different applications at different layers

Type of applications	Capabilities		
	Data	Task System	Communications
Streaming	Distributed Data Set	Static Graph	Dataflow Communications
Data Pipelines	Distributed Data Set	Static Graph or Dynamic Graph	Dataflow Communications
Machine Learning	Distributed Shared Memory	Dynamic Graph	Dataflow Communications / BSP Communications
FaaS	Stateless	Dynamic Graph	Dataflow, P2P Communication



Twister

Iterative MapReduce

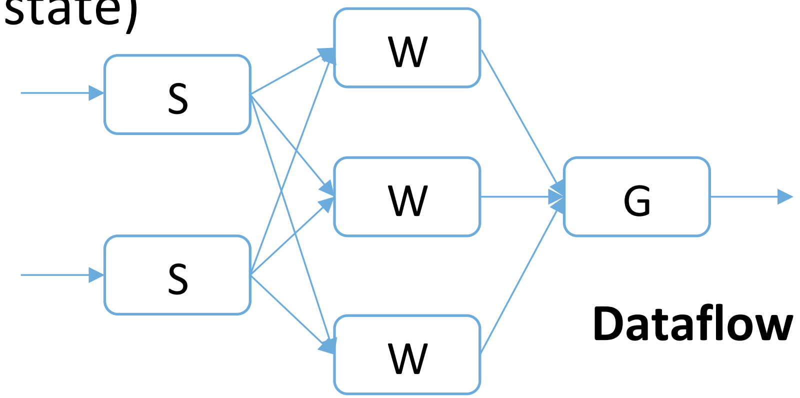
http://www.iterativemapreduce.org/

Implementing Twister2 in detail II

Look at Communication in detail

Communication Models

- **MPI Characteristics:** Tightly synchronized applications
  - Efficient communications (µs latency) with use of advanced hardware
  - In place communications and computations (Process scope for state)
- **Basic 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

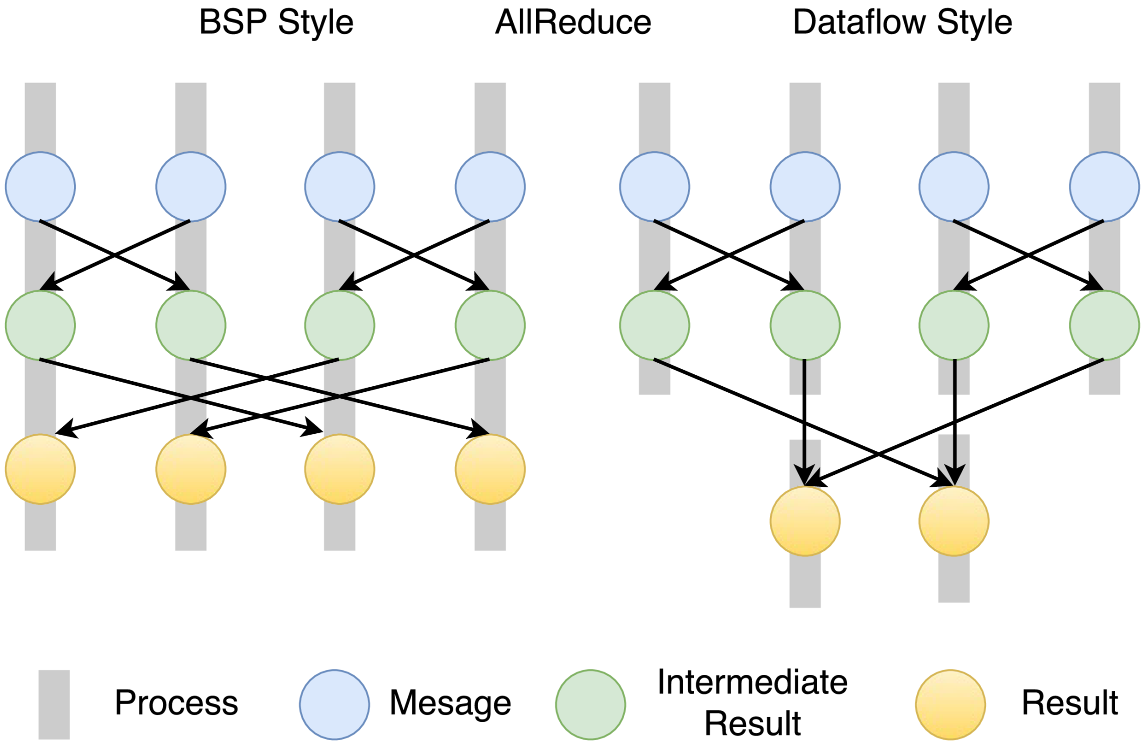


Dataflow

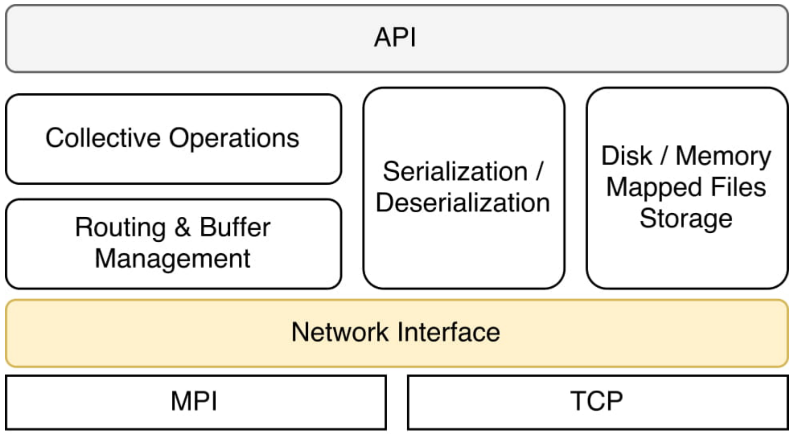
- **Streaming dataflow: Pub-Sub** with data partitioned into streams
  - Streams are unbounded, ordered data tuples
  - Order of events important and group data into time windows
- **Machine Learning dataflow:** Iterative computations and keep track of state
  - There is both Model and Data, but typically 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

Twister2 Dataflow Communications

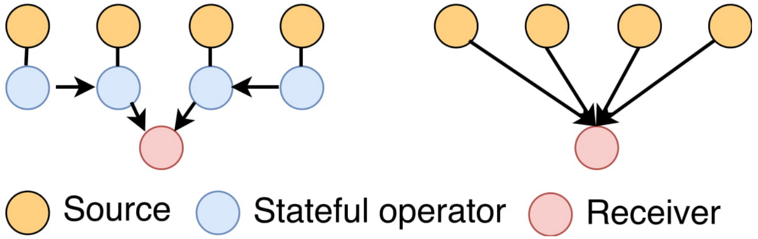
- **Twister:Net** offers two communication models
  - **BSP** (Bulk Synchronous Processing) communication using TC or MPI separated from its task management plus extra Harp collectives
- plus a new **Dataflow library DFW** built using MPI software but at data movement not message level
  - Non-blocking
  - Dynamic data sizes
  - Streaming model
    - Batch case is modeled as a finite stream
  - The communications are between a set of tasks in an arbitrary task graph
  - Key based communications
  - Communications spilling to disks
  - Target tasks can be different from source tasks



Twister:Net



Architecture



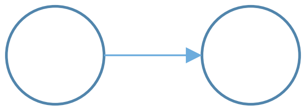
Optimized operation vs Basic (Flink, Heron)

- Communication operators are stateful
  - Buffer data
  - handle imbalanced dynamically sized communications,
  - act as a combiner
- Thread safe
- Initialization
  - MPI
  - TCP / ZooKeeper
- Buffer management
  - The messages are serialized by the library
- Back-pressure
  - Uses flow control by the underlying channel

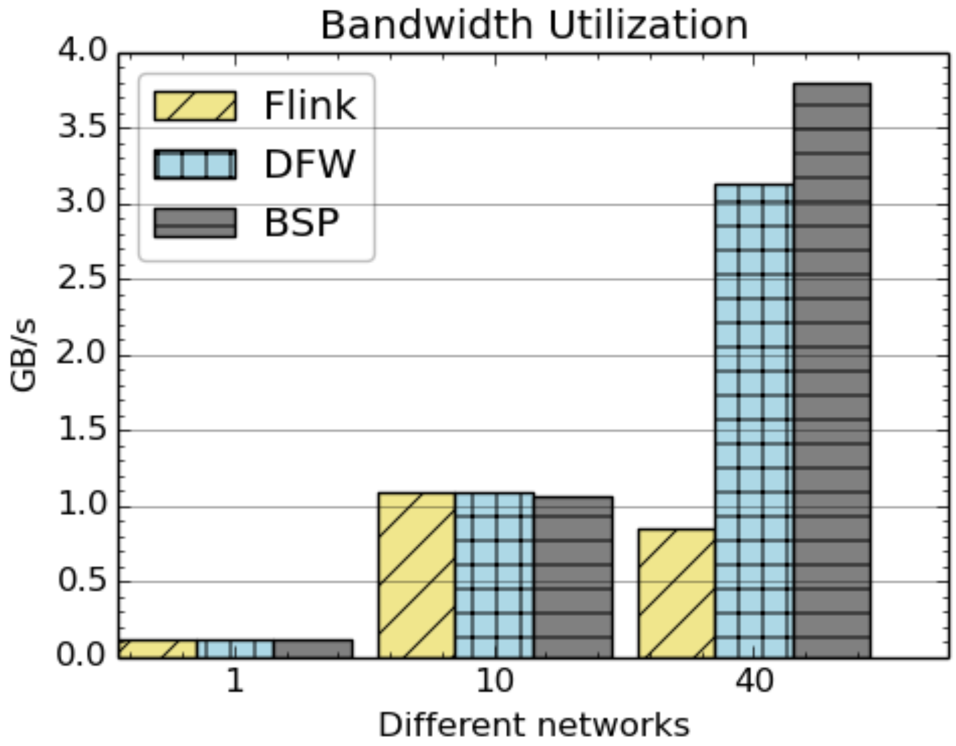
Reduce	Gather	Partition	Broadcast
AllReduce	AllGather	Keyed-Partition	
Keyed-Reduce	KeyedGather		

Batch and Streaming versions of above currently available

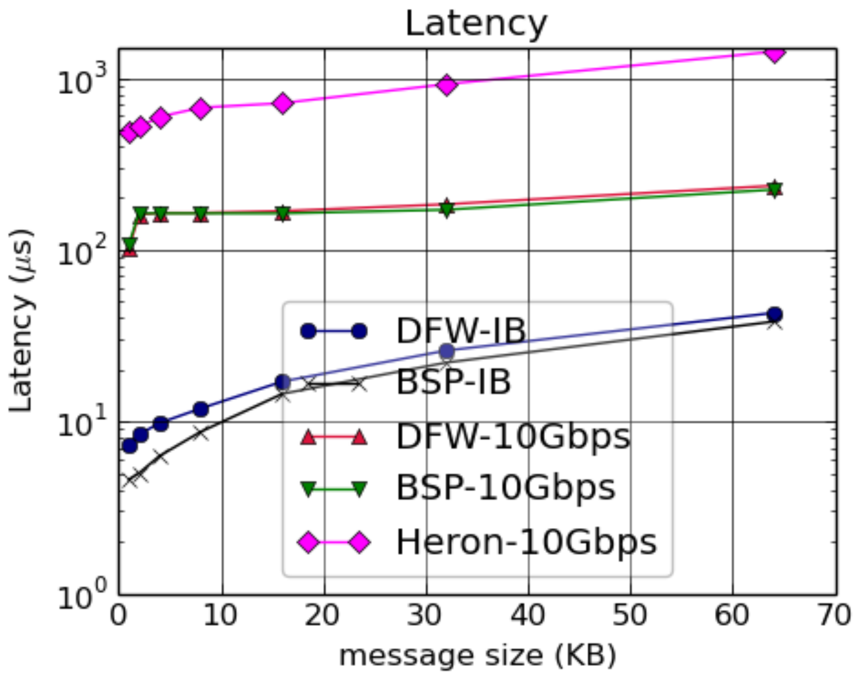
Bandwidth & Latency Kernel



Latency and bandwidth between two tasks running in two nodes

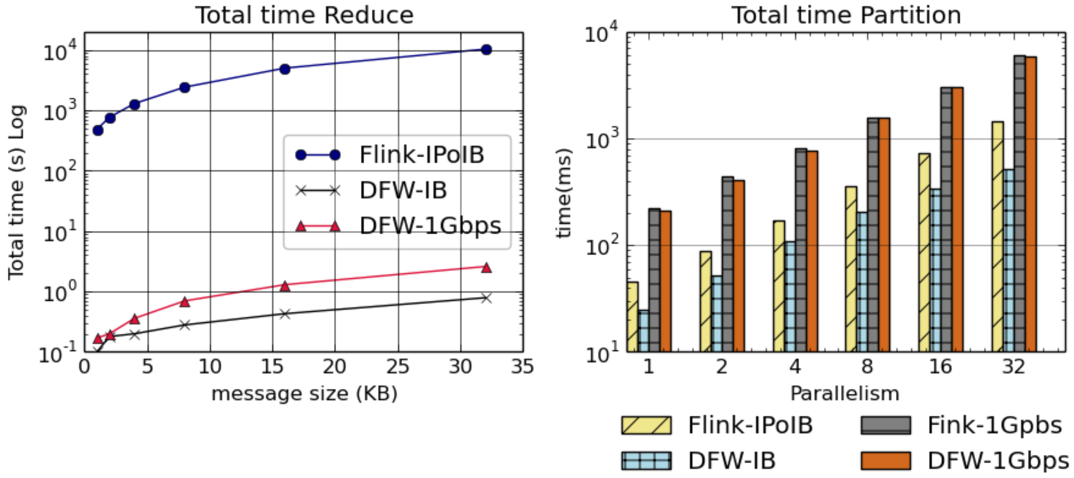


Bandwidth utilization of Flink, Twister2 and OpenMPI over 1Gbps, 10Gbps and IB with Flink on IPoIB

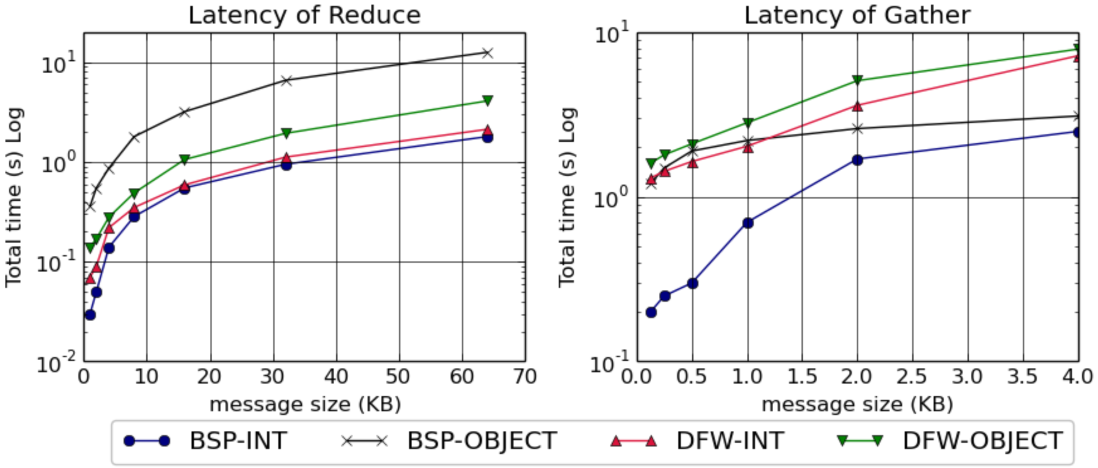


Latency of MPI and Twister:Net with different message sizes on a two-node setup

Flink, BSP and DFW Performance

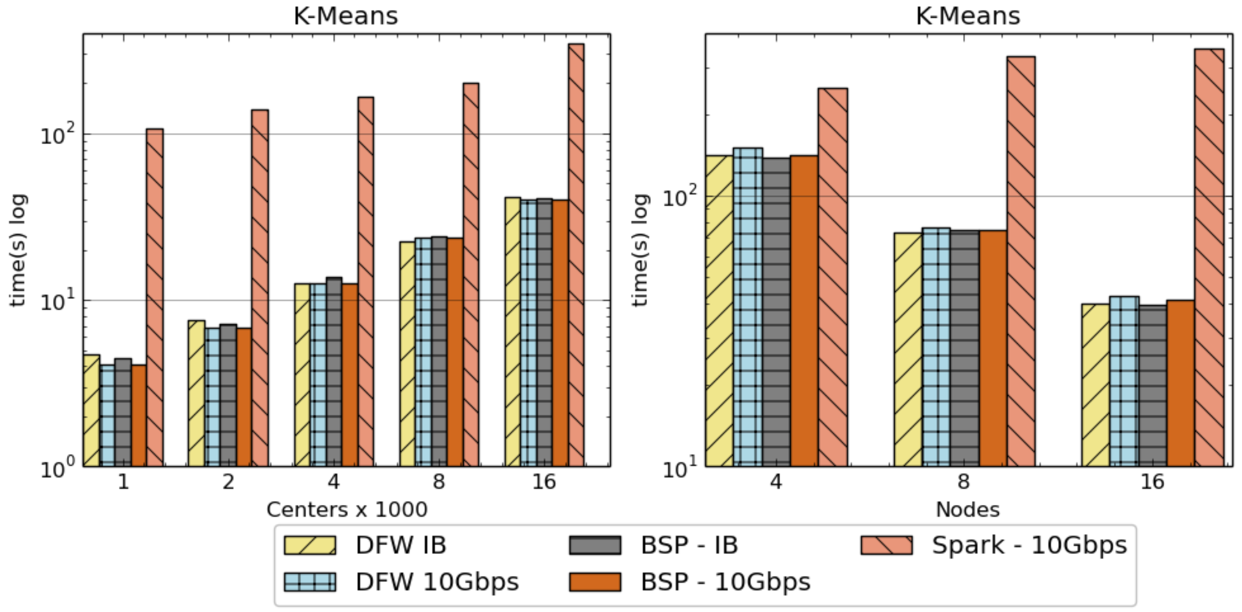


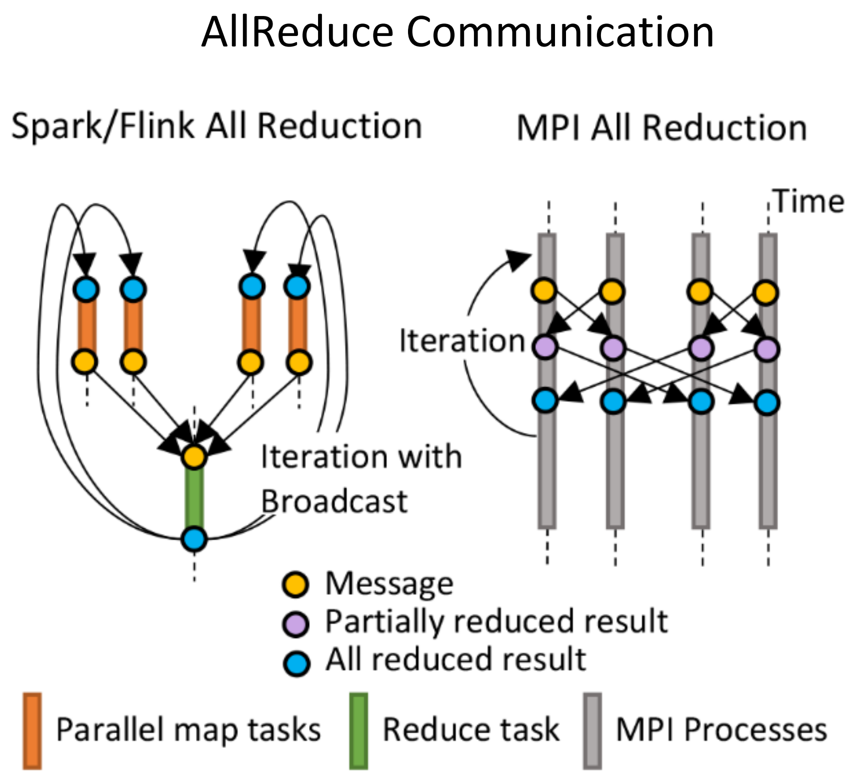
Total time for Flink and Twister:Net for Reduce and Partition operations in 32 nodes with 640-way parallelism. The time is for 1 million messages in each parallel unit, with the given message size



Latency for Reduce and Gather operations in 32 nodes with 256-way parallelism. The time is for 1 million messages in each parallel unit, with the given message size. For BSP-Object case we do two MPI calls with MPIAllReduce / MPIAllGather first to get the lengths of the messages and the actual call. InfiniBand network is used.

K-Means algorithm performance

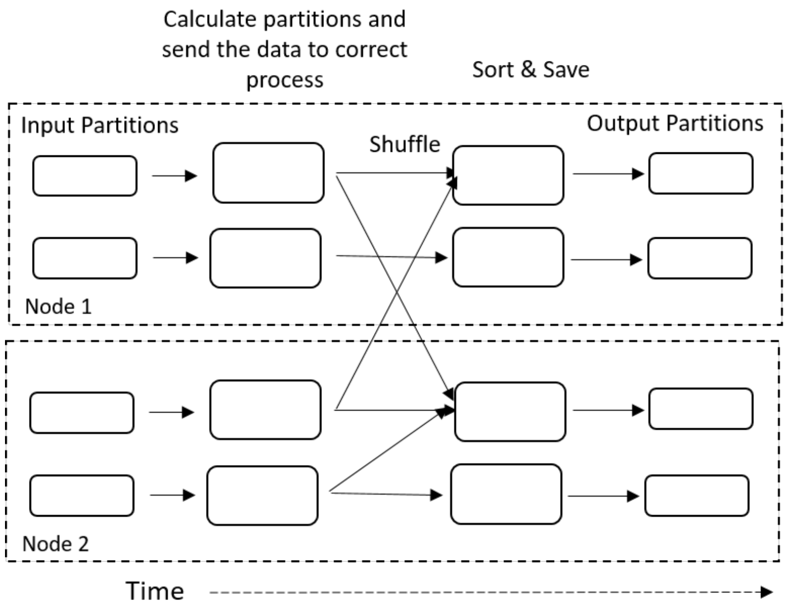




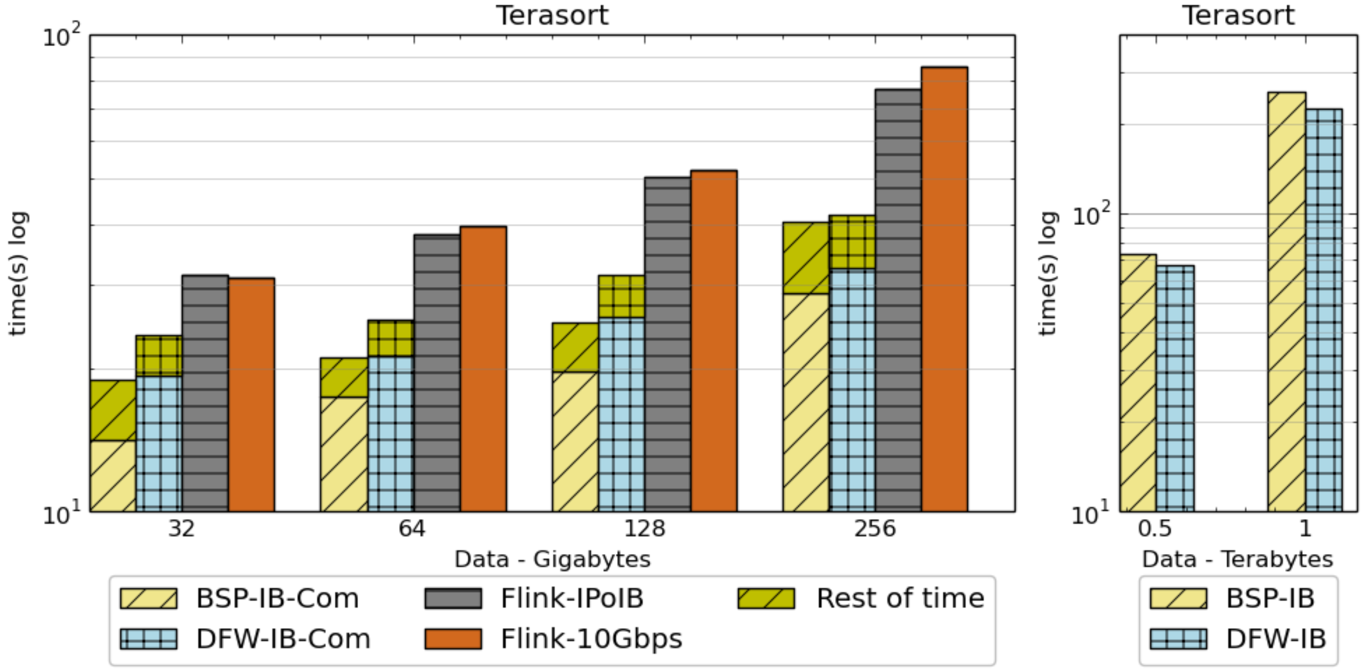
Left: K-means job execution time on 16 nodes with varying centers, 2 million points with 320-way parallelism. Right: K-Means wth 4,8 and 16 nodes where each node having 20 tasks. 2 million points with 16000 centers used.

For DFW case, a single node can get congested if many processes send message simultaneously.

Sorting Records



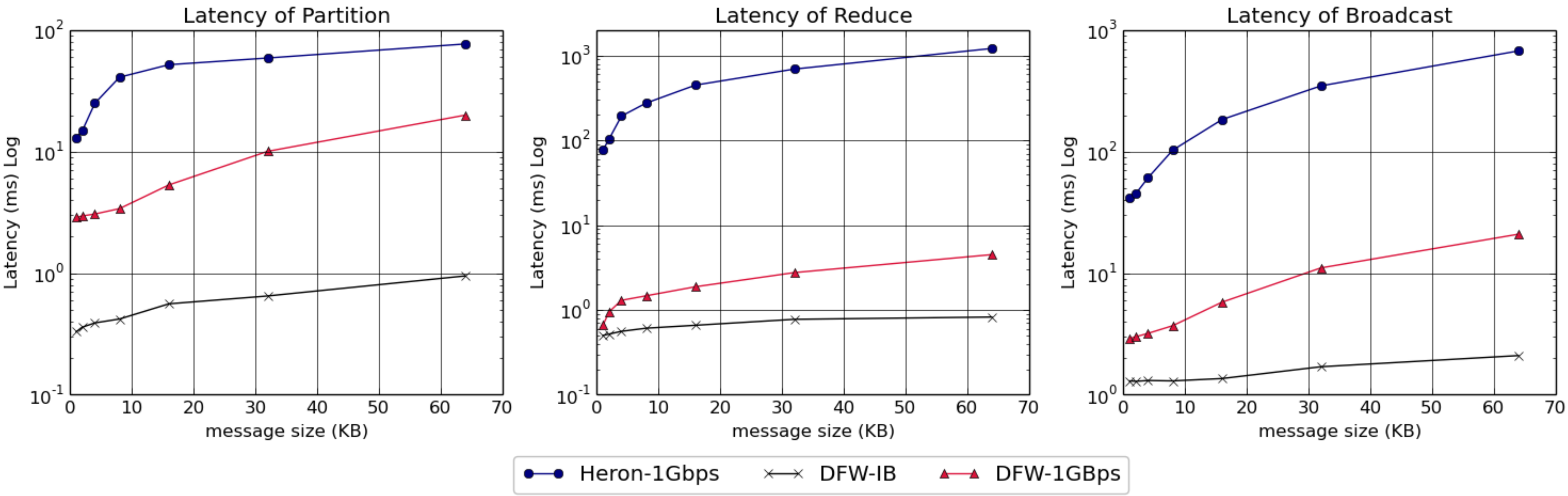
Partition the data using a sample and regroup



Left: Terasort time on a 16 node cluster with 384 parallelism. BSP and DFW shows the communication time. Right: Terasort on 32 nodes with .5 TB and 1TB datasets. Parallelism of 320. Right 16 node cluster (Victor), Left 32 node cluster (Juliet) with InfiniBand.

BSP algorithm waits for others to send messages in a ring topology and can be in-efficient compared to DFW case where processes do not wait.

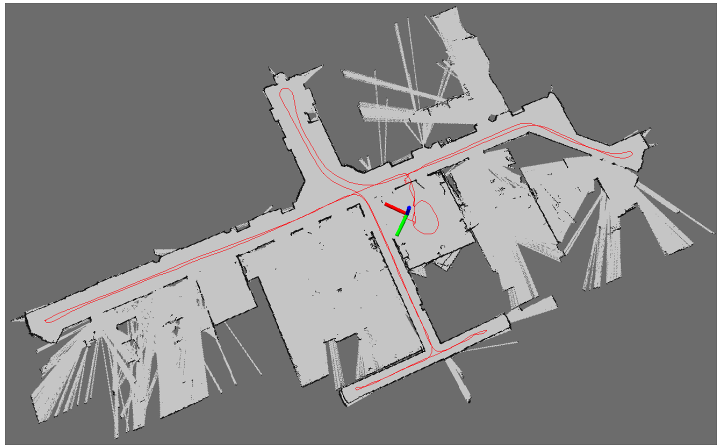
Twister:Net and Apache Heron for Streaming



Latency of Apache Heron and Twister:Net DFW (Dataflow) for Reduce, Broadcast and Partition operations in 16 nodes with 256-way parallelism

Robot Algorithms

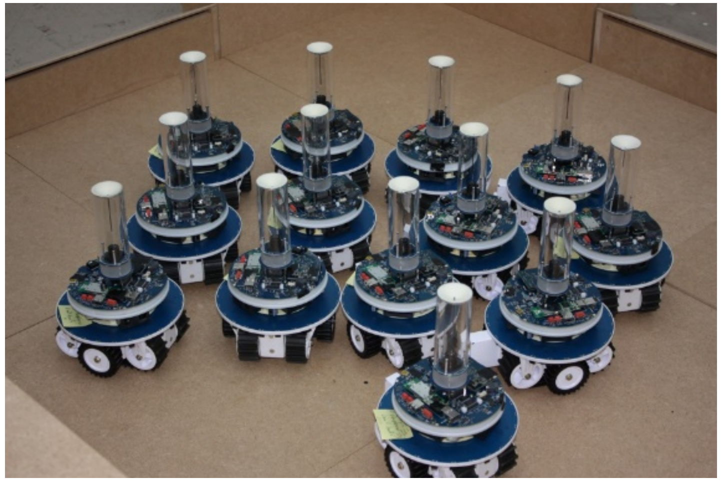
Simultaneous Localization and Mapping



Map Built from Robot data

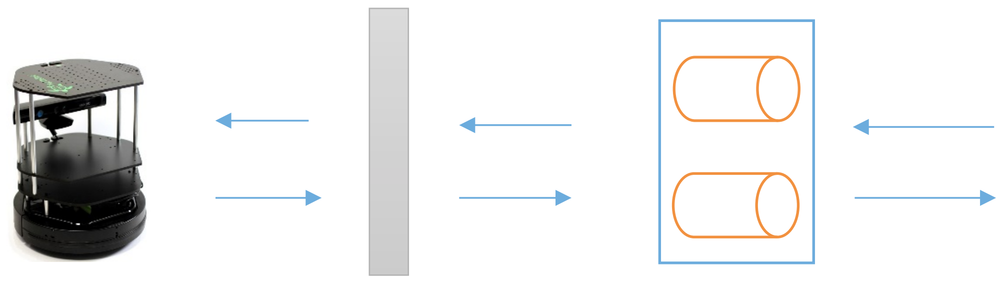
Robot with a Laser Range Finder

N-Body Collision Avoidance



Robots need to avoid collisions when they move

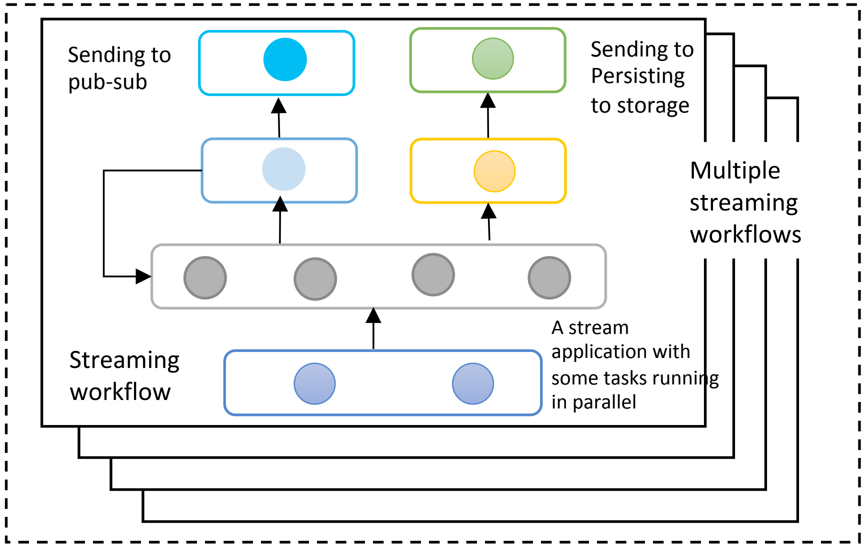
SLAM Simultaneous Localization and Mapping



Gateway

Message Brokers
RabbitMQ, Kafka

Rao blackwellized particle filter based SLAM



Streaming SLAM Algorithm

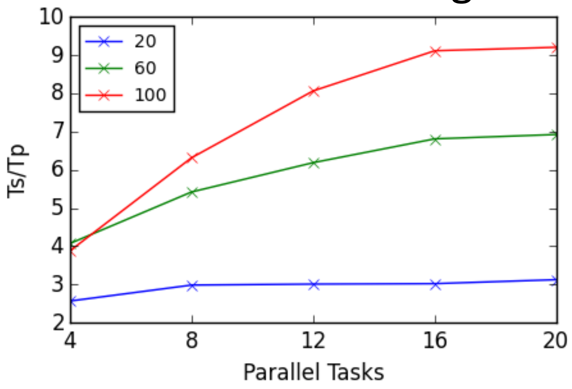
Apache Storm

End to end delays without any processing is less than 10ms

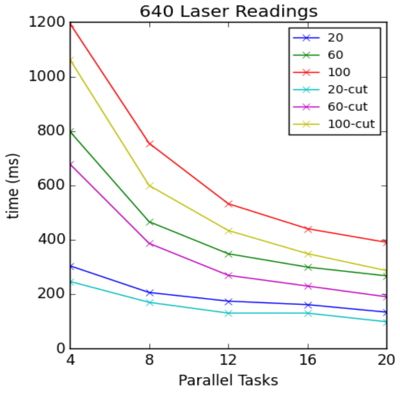
Hosted in FutureSystems OpenStack cloud which is accessible through IU network

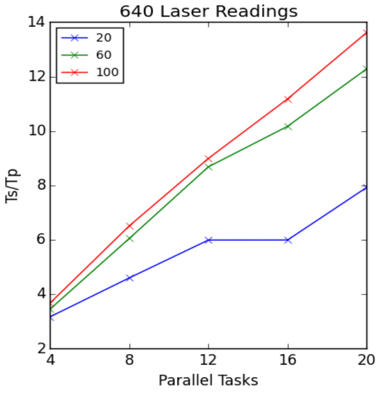
Performance of SLAM Storm v. Twister2

180 Laser readings



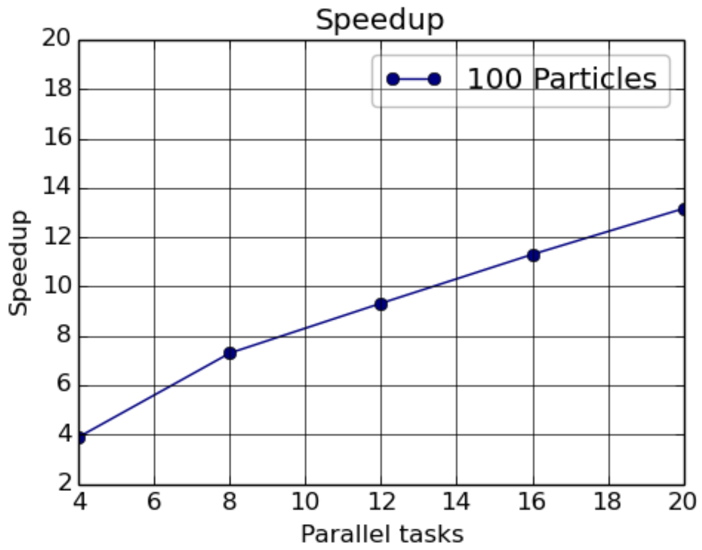
640 Laser readings



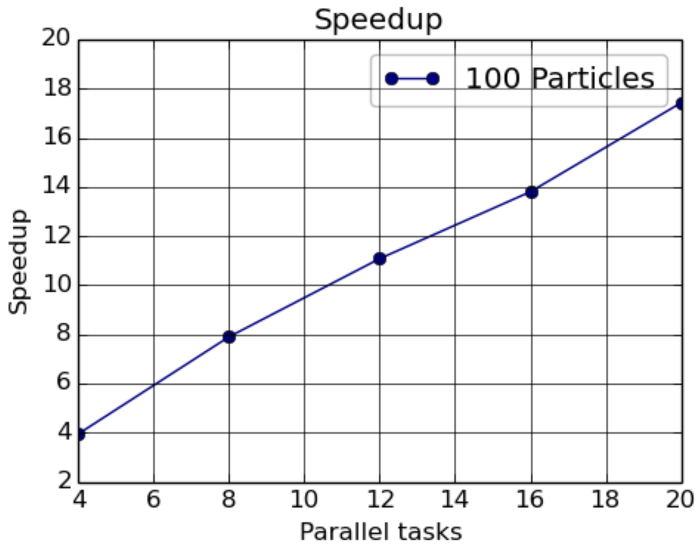


Storm Implementation Speedup

Twister2 Implementation speedup.



180 Laser readings



640 Laser readings

Twister

Iterative MapReduce

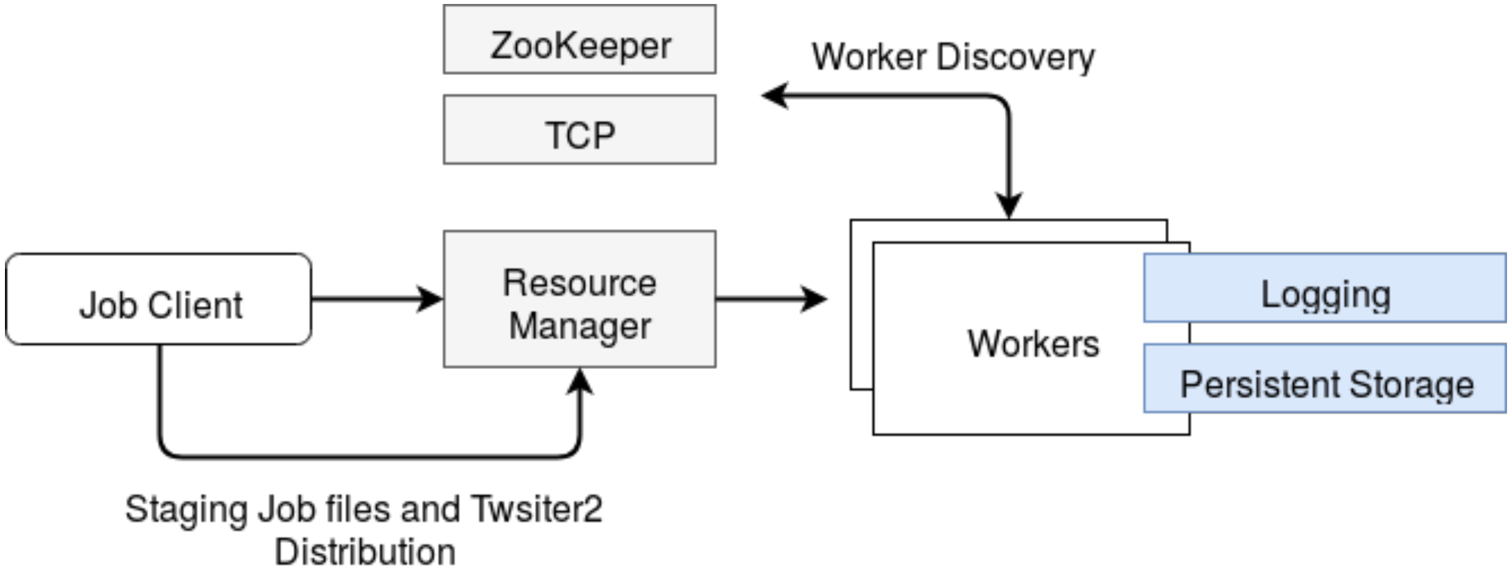
http://www.iterativemapreduce.org/

Implementing Twister2 in detail III

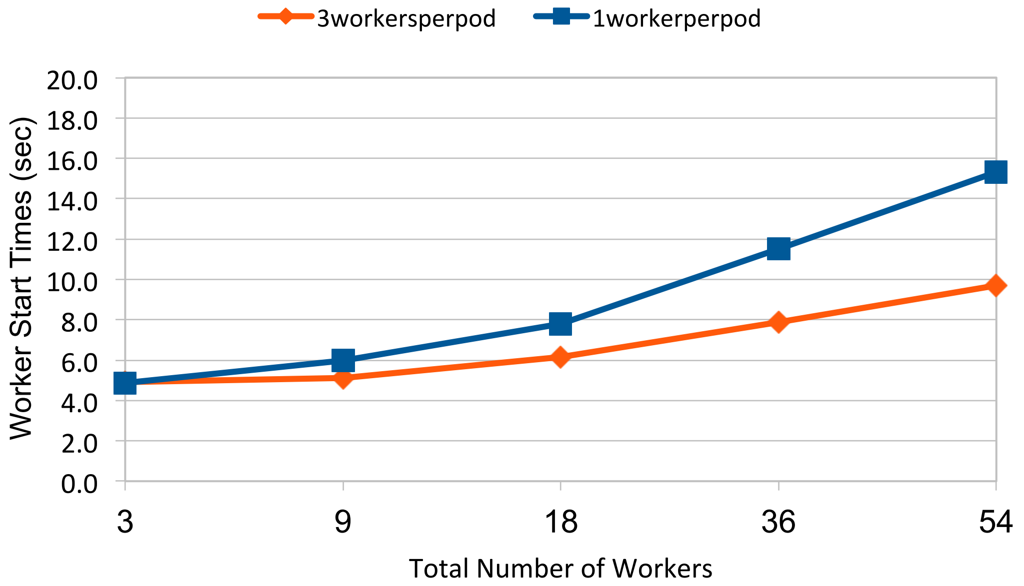
State

Resource Allocation

- Job Submission & Management
  - twister2 submit
- Resource Managers
  - Slurm
  - Nomad
  - Kubernetes
  - Mesos



Kubernetes and Mesos Worker Initialization Times

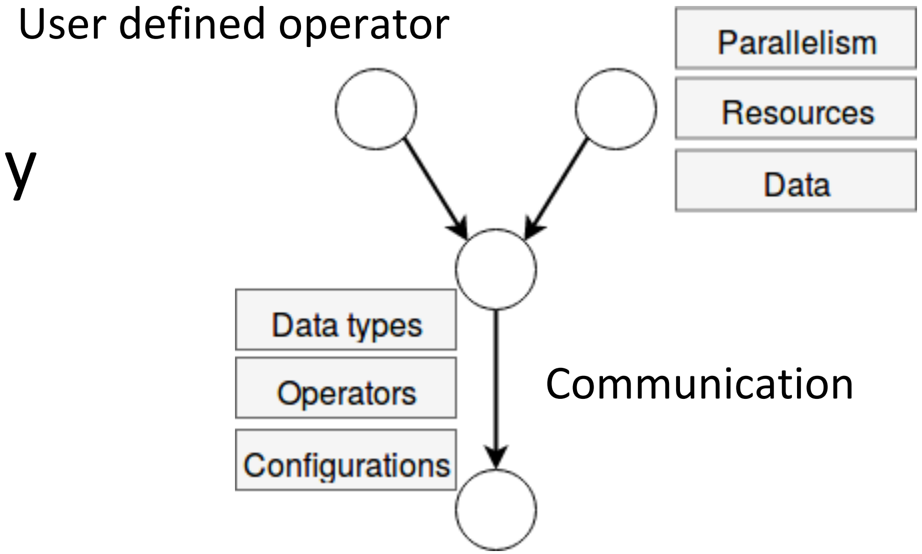


Kubernetes

Mesos

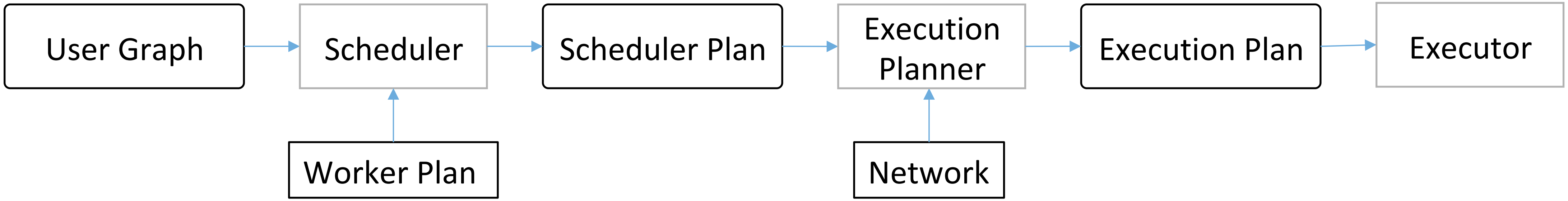
- It takes around 5 seconds to initialize a worker in Kubernetes.
- It takes around 3 seconds to initialize a worker in Mesos.
- When 3 workers are deployed in one executor or pod, initialization times are faster in both systems.

Task System



- Generate computation graph dynamically
  - Dynamic scheduling of tasks
  - Allow fine grained control of the graph
- Generate computation graph statically
  - Dynamic or static scheduling
  - Suitable for streaming and data query applications
  - Hard to express complex computations, especially with loops
- Hybrid approach
  - Combine both static and dynamic graphs

Task Graph Execution

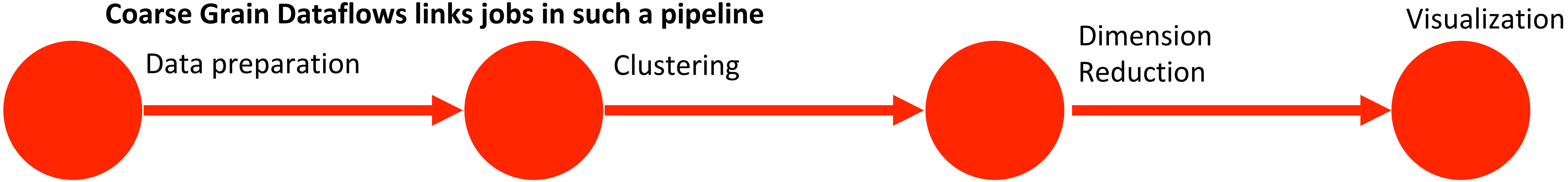


- Task Scheduler is pluggable
- Executor is pluggable
- Scheduler running on all the workers

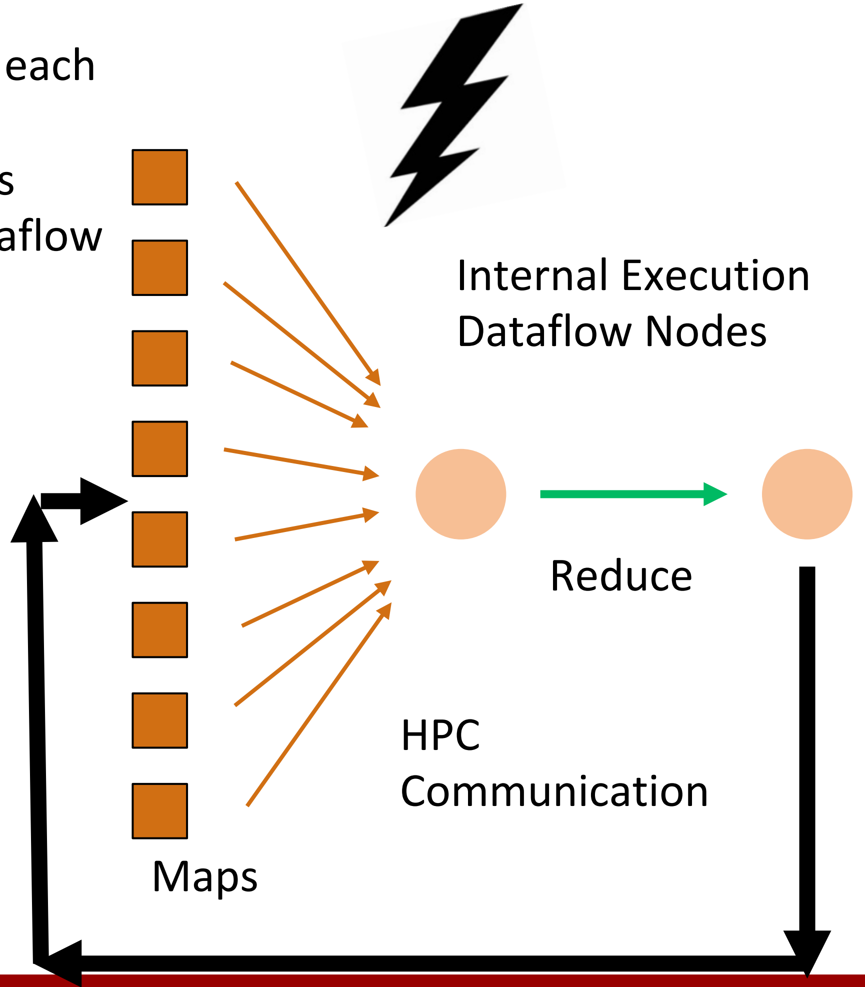
Scheduling Algorithms

- Streaming
  - Round robin
  - First fit
- Batch
  - Data locality aware

Coarse Grain Dataflows links jobs in such a pipeline



But internally to each job you can also elegantly express algorithm as dataflow but with more stringent performance constraints



Corresponding to classic Spark K-means Dataflow

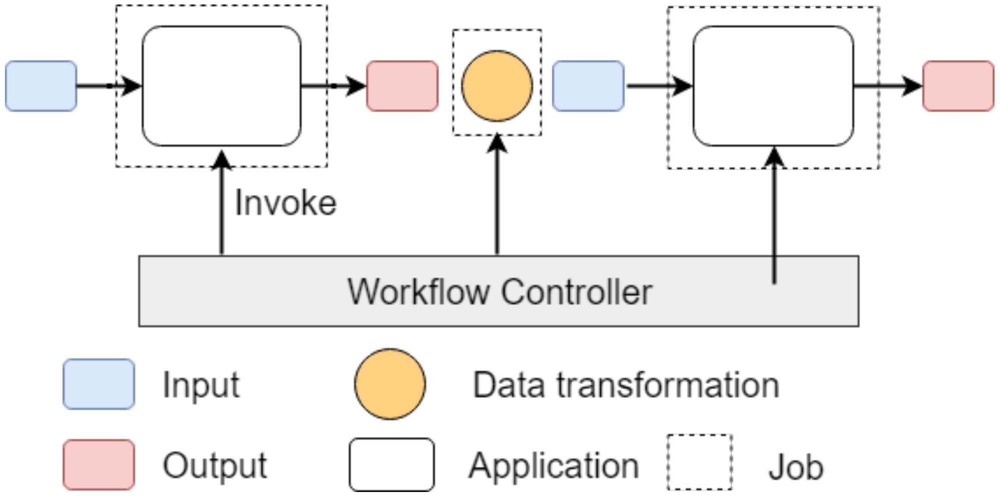
- P = loadPoints()
- C = loadInitCenters()
- for (int i = 0; i < 10; i++) {
- T = P.map().withBroadcast(C)
- C = T.reduce() }

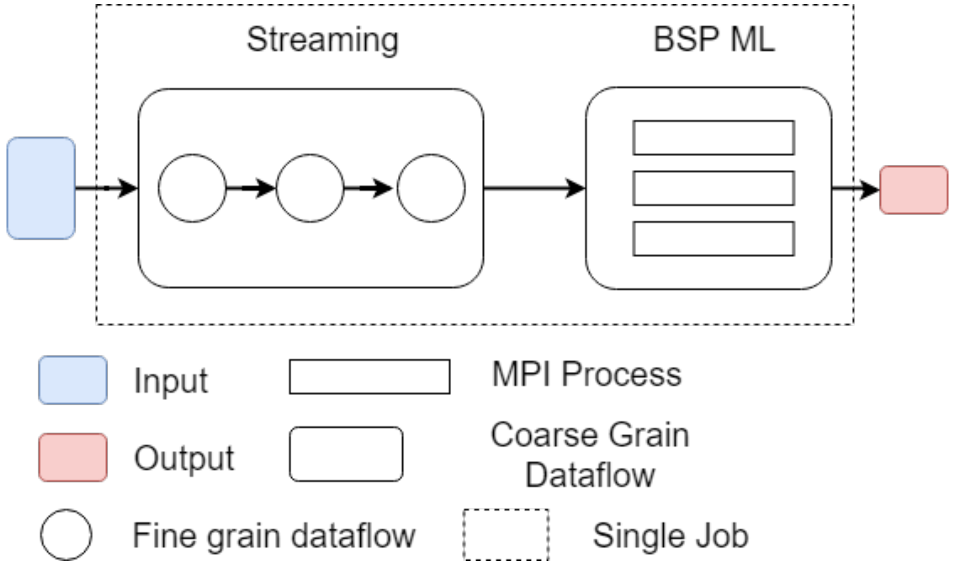
Iterate

Dataflow at Different Grain sizes

Workflow vs Dataflow: Different grain sizes and different performance trade-offs

The dataflow can expand from Edge to Cloud

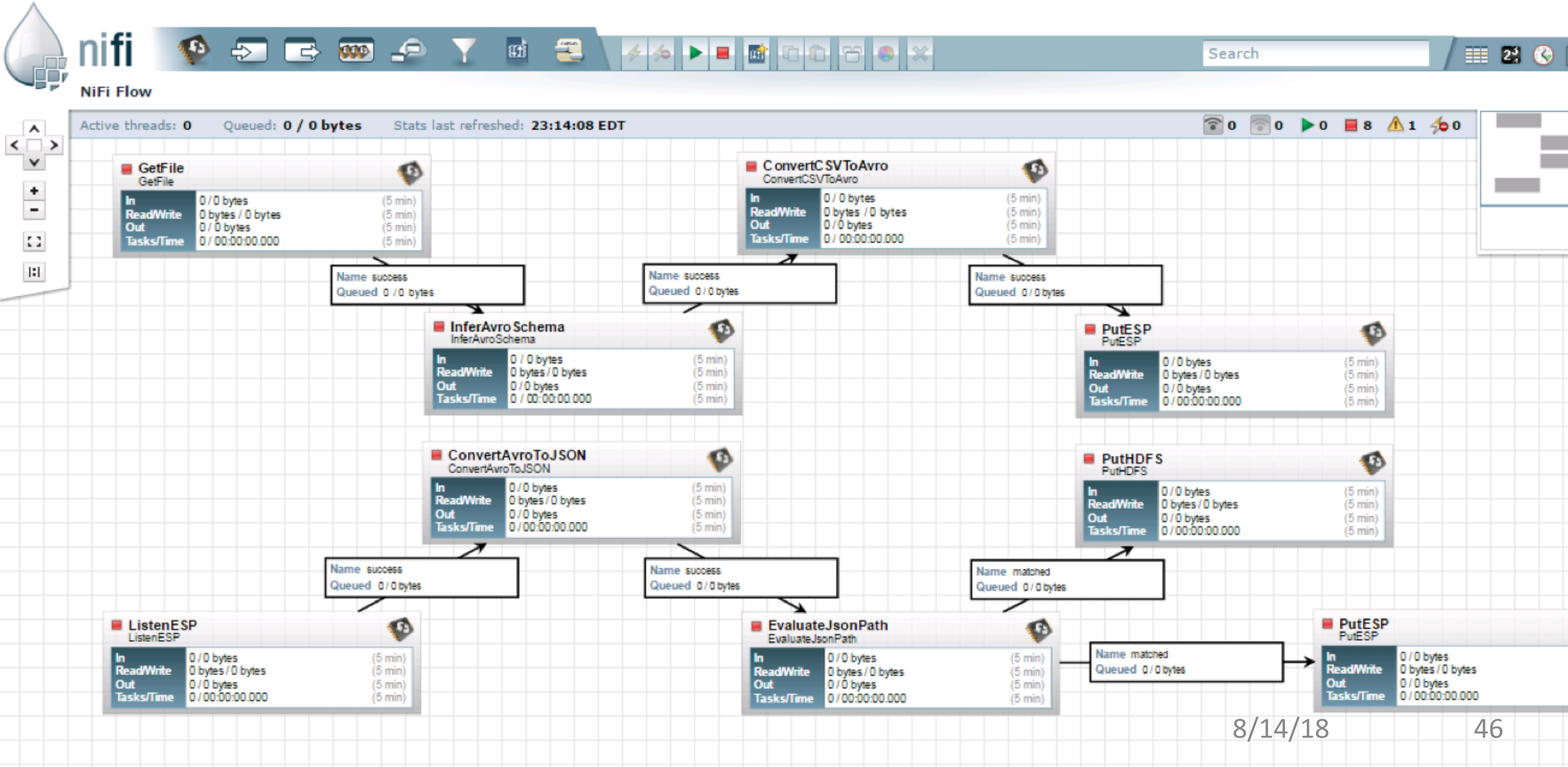




Workflow Controlled by Workflow Engine or a Script

Dataflow application running as a single job

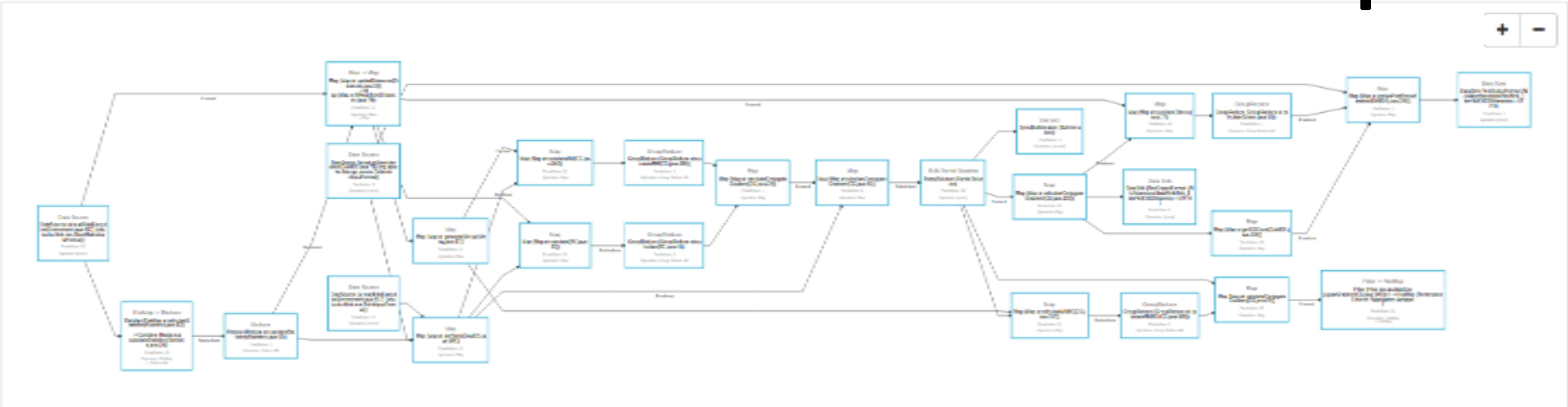
NiFi Workflow



Flink MDS Dataflow Graph

Flink Java Job at Mon Oct 03 12:27:10 EDT 2016 b3057e083f05a3b5c4ee2f515ab7d7b8 0 0 0 0 27 0 0 0 2016-10-03, 12:27:10 - 2016-10-03, 12:39:34 12m 23s

Plan Timeline Exceptions Properties Configuration



Subtasks TaskManagers Accumulators Checkpoints

Start Time	End Time	Duration	Name	Bytes received	Records received	Bytes sent	Records sent	Tasks	Status
2016-10-03, 12:27:10	2016-10-03, 12:33:18	6m 7s	DataSource (at readFile(ExecutionEnvironment.java:517) (edu.iu.dsc.flink.mm.ShortMatrixInputFormat))	0 B	0	3.81 GB	64	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:27:10	2016-10-03, 12:27:10	45ms	DataSource (at setupStressIteration(DAMDS.java:71) (org.apache.flink.api.java.io.CollectionInputFormat))	0 B	0	1.68 KB	33	0 0 0 1 0 0 0	FINISHED
2016-10-03, 12:27:10	2016-10-03, 12:27:11	739ms	DataSource (at readFile(ExecutionEnvironment.java:517) (edu.iu.dsc.flink.mm.PointInputFormat))	0 B	0	750 KB	1	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:33:18	2016-10-03, 12:33:23	5s	CHAIN FlatMap (FlatMap at calculateStatistics(Statistics.java:12)) -> Combine (Reduce at calculateStatistics(Statistics.java:20))	1.91 GB	32	2.56 KB	32	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:33:23	2016-10-03, 12:33:23	426ms	Reduce (Reduce at calculateStatistics(Statistics.java:20))	2.56 KB	32	5.13 KB	64	0 0 0 1 0 0 0	FINISHED
2016-10-03, 12:33:18	2016-10-03, 12:33:57	38s	CHAIN Map (Map at updateDistances(Distances.java:33)) -> Map (Map at filReadJoin(Distances.java:74))	1.91 GB	32	11.4 GB	96	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:33:57	2016-10-03, 12:34:29	32s	Map (Map at generateVArray(VArray.java:17))	3.81 GB	32	3.82 GB	64	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:27:10	2016-10-03, 12:33:24	6m 13s	Map (Map at joinStats(DAMDS.java:345))	750 KB	1	47.6 MB	65	0 0 0 32 0 0 0	FINISHED
2016-10-03, 12:33:24	2016-10-03, 12:34:40	1m 16s	Map (Map at calculateMM(CG.java:260))	1.91 GB	32	752 KB	32	0 0 0 32 0 0 0	FINISHED

Systems State

- **State** is handled differently in systems
  - CORBA, AMT, MPI and Storm/ Heron have long running tasks that preserve state
  - Spark and Flink preserve datasets across dataflow node using in-memory databases
  - All systems agree on coarse grain dataflow; only keep state by exchanging data

Spark Kmeans Dataflow

- P = loadPoints()
- C = loadInitCenters()

Iterate

- for (int i = 0; i < 10; i++) {
-   T = P.map().withBroadcast(C)
-   C = T.reduce()   }

Save State at Coordination Point
Store C in RDD

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** **(at location of intelligent dataflow nodes)**
  - 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

Twister

Iterative MapReduce

http://www.iterativemapreduce.org/

Implementing Twister2 Futures

Twister2 Timeline: End of August 2018

- Twister:Net Dataflow Communication API
  - Dataflow communications with MPI or TCP
- Harp for Machine Learning (Custom BSP Communications)
  - Rich collectives
  - Around 30 ML algorithms
- HDFS Integration
- Task Graph
  - Streaming - Storm model
  - Batch analytics - Hadoop
- Deployments on Docker, Kubernetes, Mesos (Aurora), Nomad, Slurm

Twister2 Timeline: End of December 2018

- Native MPI integration to Mesos, Yarn
- Naiad model based Task system for Machine Learning
- Link to Pilot Jobs
- Fault tolerance
  - Streaming
  - Batch
- Hierarchical dataflows with Streaming, Machine Learning and Batch integrated seamlessly
- Data abstractions for streaming and batch (Streamlets, RDD)
- Workflow graphs (Kepler, Spark) with linkage defined by Data Abstractions (RDD)
- End to end applications

Twister2 Timeline: After December 2018

- Dynamic task migrations
- RDMA and other communication enhancements
- Integrate parts of Twister2 components as big data systems enhancements (i.e. run current Big Data software invoking Twister2 components)
  - Heron (easiest), Spark, Flink, Hadoop (like Harp today)
- Support different APIs (i.e. run Twister2 looking like current Big Data Software)
  - Hadoop
  - Spark (Flink)
  - Storm
- Refinements like Marathon with Mesos etc.
- Function as a Service and Serverless
- Support higher level abstractions
  - Twister:SQL

Summary of Twister2: Next Generation HPC Cloud + Edge + Grid

- We have built a high performance data analysis library SPIDAL
- We have integrated HPC into many Apache systems with HPC-ABDS with rich set of collectives
- We have done a preliminary analysis of the different runtimes of Hadoop, Spark, Flink, Storm, Heron, Naiad, DARMA (HPC Asynchronous Many Task) and identified key components
- There are different technologies for different circumstances but can be unified by high level abstractions such as communication/data/task API's
- Apache systems use dataflow communication which is natural for distributed systems but slower for classic parallel computing
  - No standard dataflow library (why?). **Add Dataflow primitives in MPI-4?**
- HPC could adopt some of tools of Big Data as in Coordination Points (dataflow nodes), State management (fault tolerance) with RDD (datasets)
- Could integrate dataflow and workflow in a cleaner fashion
- Not clear so many big data and resource management approaches needed