Converged Ecosystem for Data Analytics and Extreme-Scale Computing

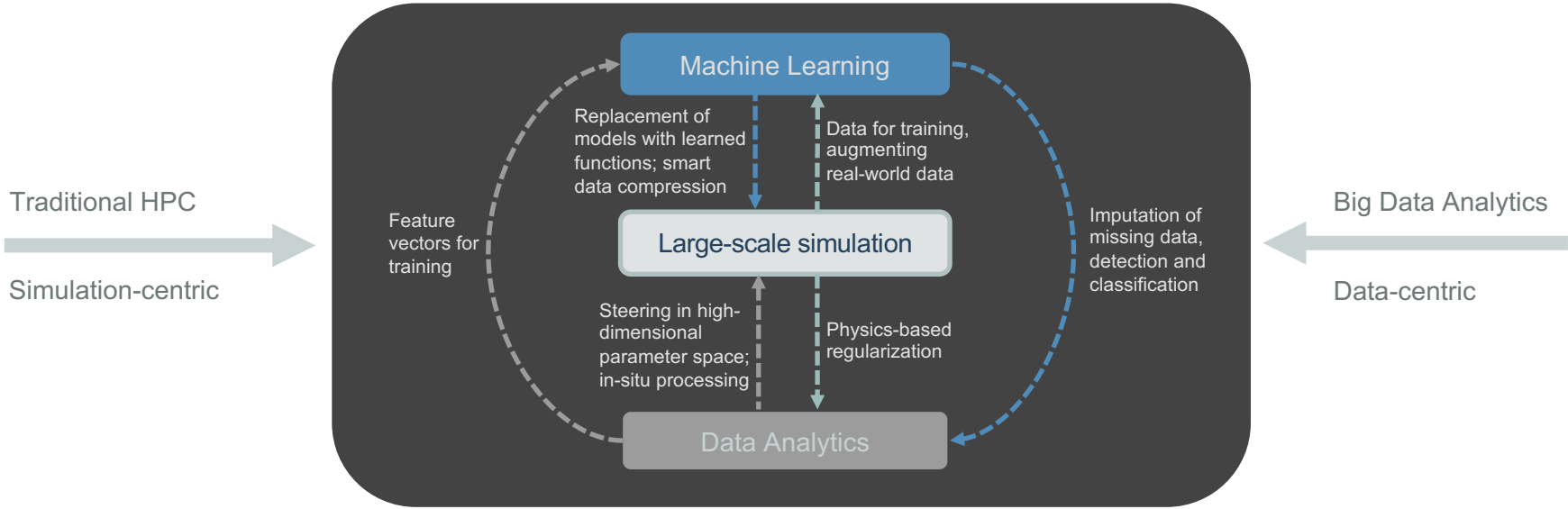
Carlos Costa
Data-Centric Solutions (DCS)
IBM T.J. Watson Research Center

chcost@us.ibm.com

IBM Research

Era of Data-Centric and Intelligent Discovery

IBM **Research**



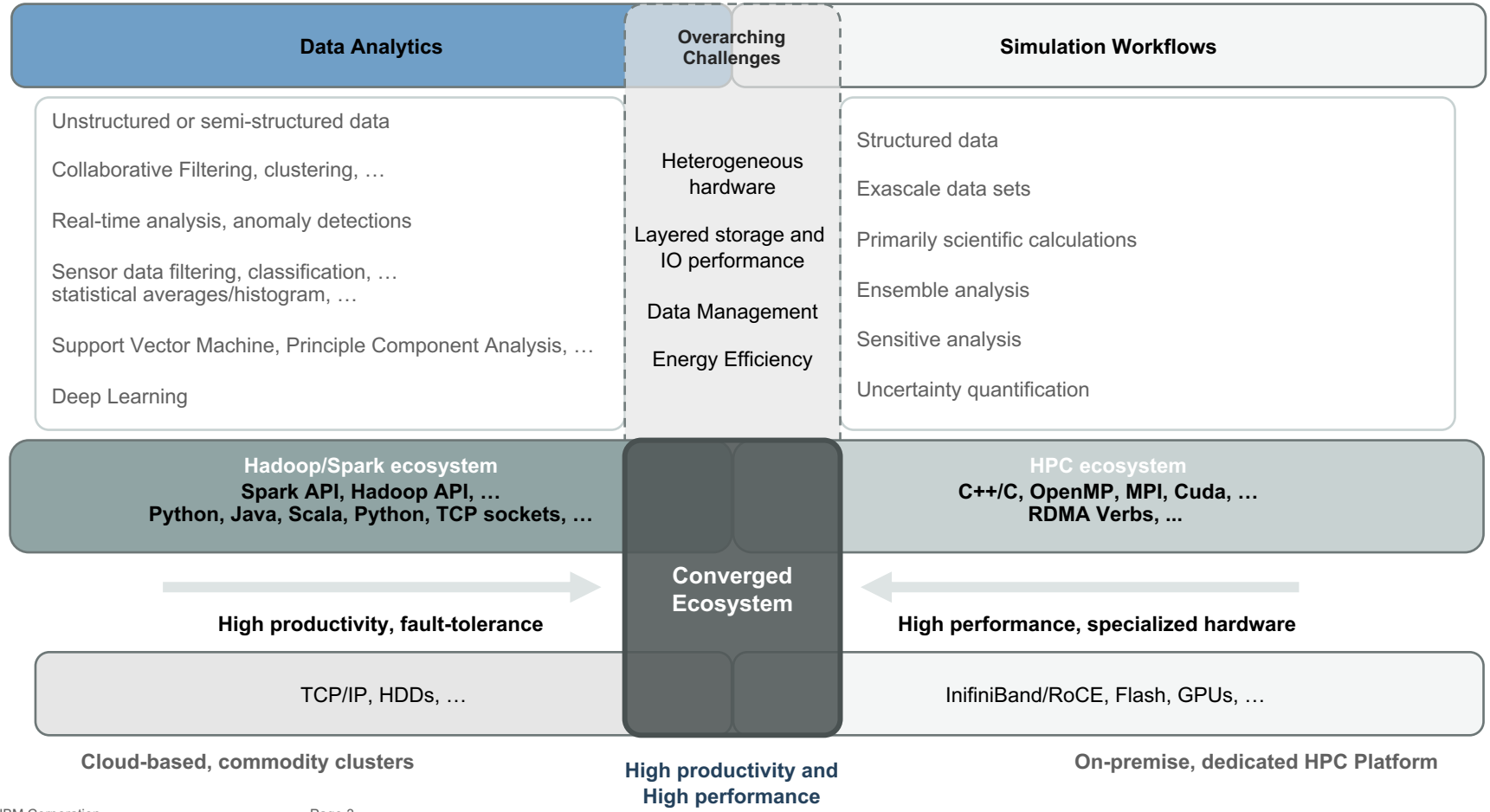
- Explosion of data generated by large-scale simulation leading to a paradigm shift: from simulation-centric to **data-centric discovery**
- Data analytics and machine learning used to turn reams of simulation data into **actionable** information that can be used for better interpretation and steering
- Applying machine learning for making existing simulation codes more **intelligent**, more productive, and more robust

- Increasing interest in **large-scale** analytics and machine learning on high-end platforms
- Emerging hybrid workflows that embody the **entire** inference cycle of discovery
- Co-deployment of **heterogenous** software stacks

Diagram adapted from BDEC report
https://www.exascale.org/bdec/report

Towards a Converged Ecosystem

IBM Research



Next-Gen Large-scale Computing Platforms

Designing Next-Generation Systems

IBM **Research**

Data-centric approach

Exascale

1954
The Naval Ordnance Research Calculator helped forecast weather and performed other complex calculations.

1961
The **IBM 7030** was capable of 2 million operations per second.

1966
The **IBM 360** and its successors helped power NASA's Apollo program.

1997
Deep Blue wins its match with chess grandmaster Garry Kasparov.

2004
Blue Gene ushers in a new era of high-performance computing as it helps biologists explore gene development.

2008
Built for Los Alamos National Laboratory, **Roadrunner** is the first supercomputer in the world to reach petaflop speed.

2011
Watson beats human competitors on *Jeopardy!*, earning a million-dollar jackpot for charity.

2012
Sequoia, the third-generation **Blue Gene** system, reaches speeds of 16.32 petaflops.

2018
Summit begins work at Oak Ridge National Laboratory; a sister machine, **Sierra,** launches at Lawrence Livermore National Laboratory.

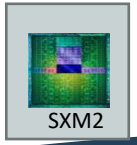
- DoE's CORAL systems, ORNL's Summit and LLNL's Sierra, first instantiation of IBM's data-centric approach for system design
- Platforms for innovative emerging workflows, laying the groundwork for efficient scientific discovery at Exascale
- Designed for data and AI from the ground up
- Summit, currently world's most powerful system
- 5 2018 Gordon Bell finalists used Summit or Sierra (including record 2.31 exaops)

OpenPOWER CORAL Systems Design

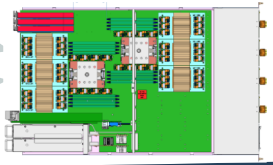
IBM **Research**

POWER9:
22 Cores
4 Threads/core
0.65 DP TF/s
3.7 GHz

Volta:
7.0 DP TF/s
16GB @ 1.2TB/s



POWER9 2-Socket Server
2 P9 + 4/6 Volta GPU (@7 TF/s)
512 GiB SMP Memory (32 GB DDR4 RDIMMs)
64/96GiB GPU Memory (HBM stacks)



Converged 2U server drawer for HPC and Cloud

Compute Rack:
18 Servers/rack
779 TFlop/rack
10.8 TB/rack
55 kWatts max

Scalable Active Network:
Mellanox IB4X EDR Switch

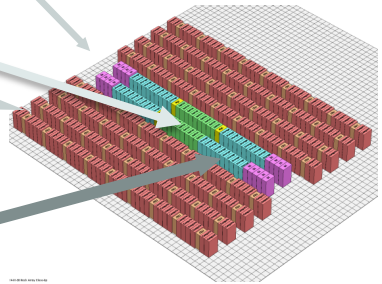
256 Compute Racks

- Scalable system software and data architecture
- LLVM Open Source compiler
- Water cooling
- Integrated Local Active Storage

16 Optional Flash Racks

TMS drawers or Flash cards.
CAPI attached.
Globally accessible with local processing

System:
200 Pflops compute
+ 5 PB Active Flash
+120 PB Disk



GSS Rack:

40 Disk Racks

CORAL Systems

IBM **Research**

- Scalable system solution – scale up, scale down – to address a wide range of application domains
  - Modular, flexible, cost-effective, 2U building blocks
  - Directly leverages OpenPOWER partnerships and IBM's Power system roadmap
  - Can scale to over 500 PF
  - Air and water cooling
- Heterogeneous compute elements
  - Power9 processor,
  - Nvidia Volta GPU coupled with NVLINK 2.0 (Coherent, high-bandwidth links to GPUs)
- Heterogeneous memory elements
  - DRAM for low latency
  - HMC Stacked memory for high bandwidth
  - Flash for local store
- System Resource Manager coupled with Platform Computing LSF

ORNL's Summit

- 4,608 POWER9 nodes with
- 6x Nvidia Volta GPU
- 2,282,544 cores
- ~10 PB (DDR4+HBM2+Non-volatile) system memory
- Dual-rail InfiniBand Fat tree network
- ~200PF peak
- ~13 MW

LLNL's Sierra

- ~4,474 POWER9 nodes with 4x Nvidia Volta GPU
- 1,572,480 cores
- ~1.9 PB system memory
- Dual-rail InfiniBand Fat tree network
- ~125 PF peak
- ~ 11 MW

Challenges for a Converged Software Ecosystem

Converged Software Ecosystem Challenges

IBM **Research**

Bifurcated software **development paradigms** and cultures

Disjoint data management approaches and lack of a unified **data flow** model

Distinct **scheduling** (stateless vs state full) and **deployment** requirements and strategies

Distinct **storage** and **computing** models

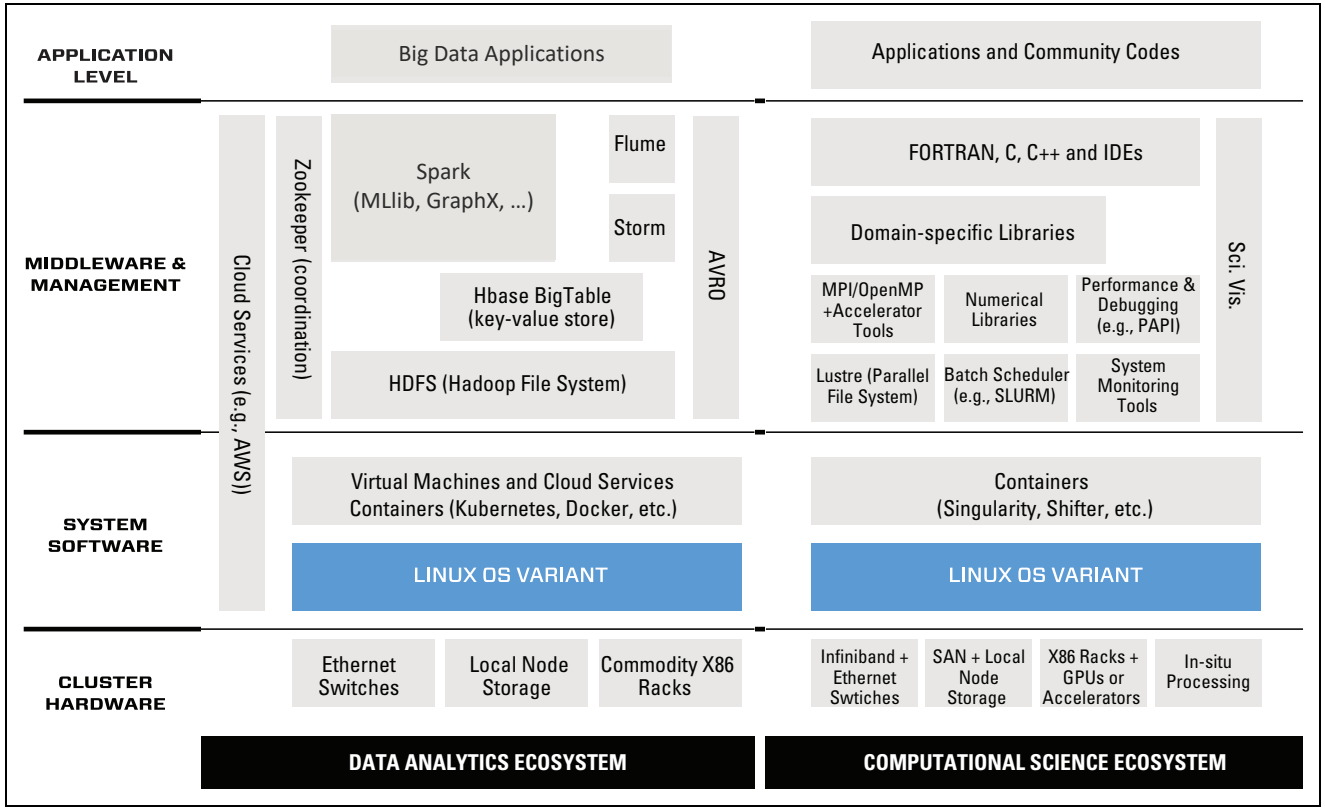
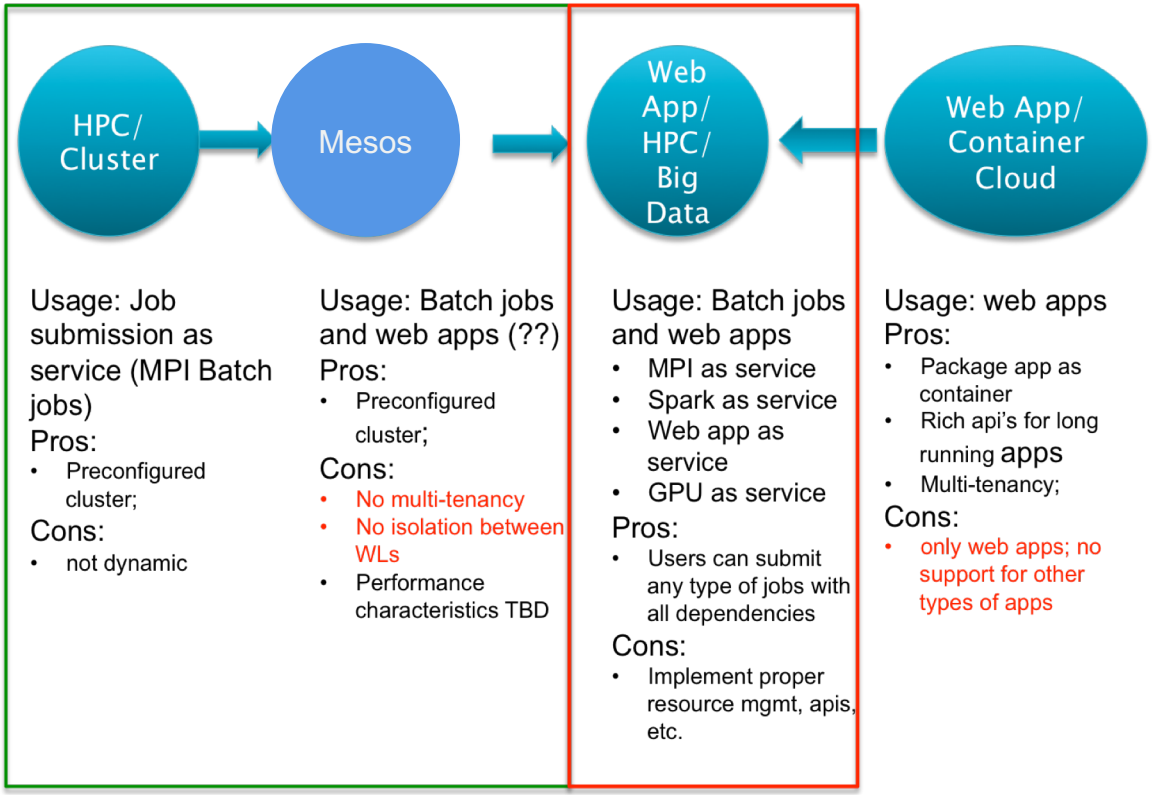


Diagram adapted from BDEC report
https://www.exascale.org/bdec/report

Converged Software Ecosystem Challenges

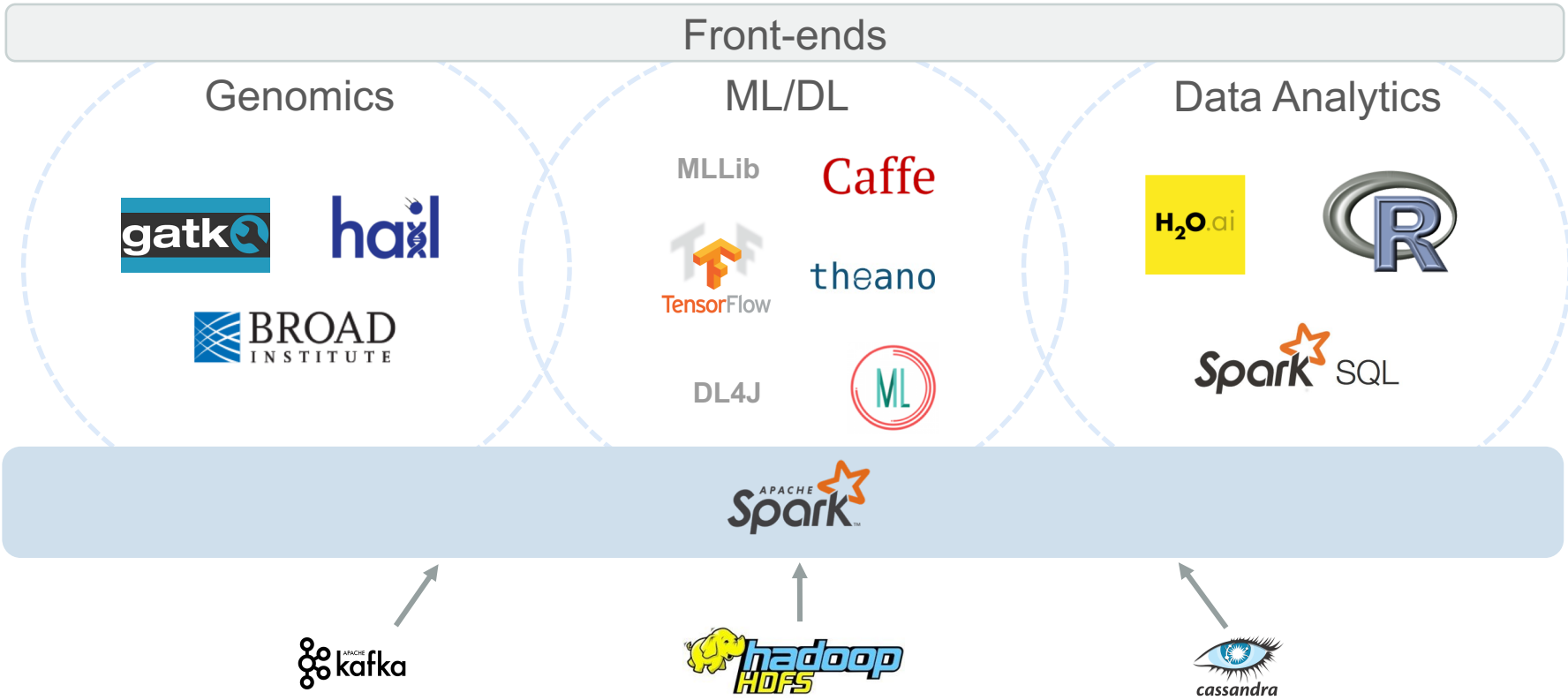
IBM **Research**

- Need for better resource managers and **schedulers**
- Enabling co-existence of **batch** processing and **interactive** analysis
- Need of interoperability of **data formats**
- Enabling **integration** of on-premises and cloud HPC environments
- Enabling cloud **bursting**



Towards a Common Platform for Analytics

IBM **Research**



Benefits and Challenges with Spark-based Stack

IBM Research

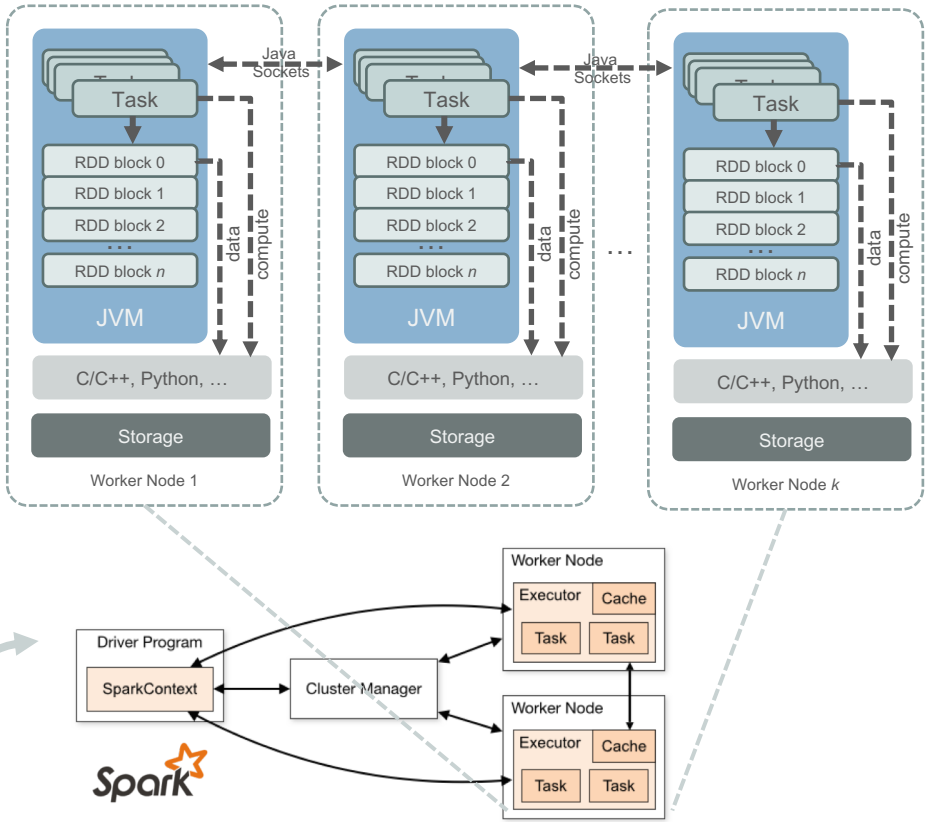
Benefits

- Functional programming targeting data analytics that naturally apply the same **operation** to multiple data items
- Operators expressive enough to capture wide class of computation and **cluster programming models** (MapReduce, SQL, Pregel, …) for analytics

Challenges

- JVM-based approach creates challenges with **specialized hardware** (GPU, FPGA, transport off-load)
- Challenges managing **data** across frameworks
- Efforts in the community with off-JVM heap optimizations and native **code generation** (Tungsten and Catalyst projects)

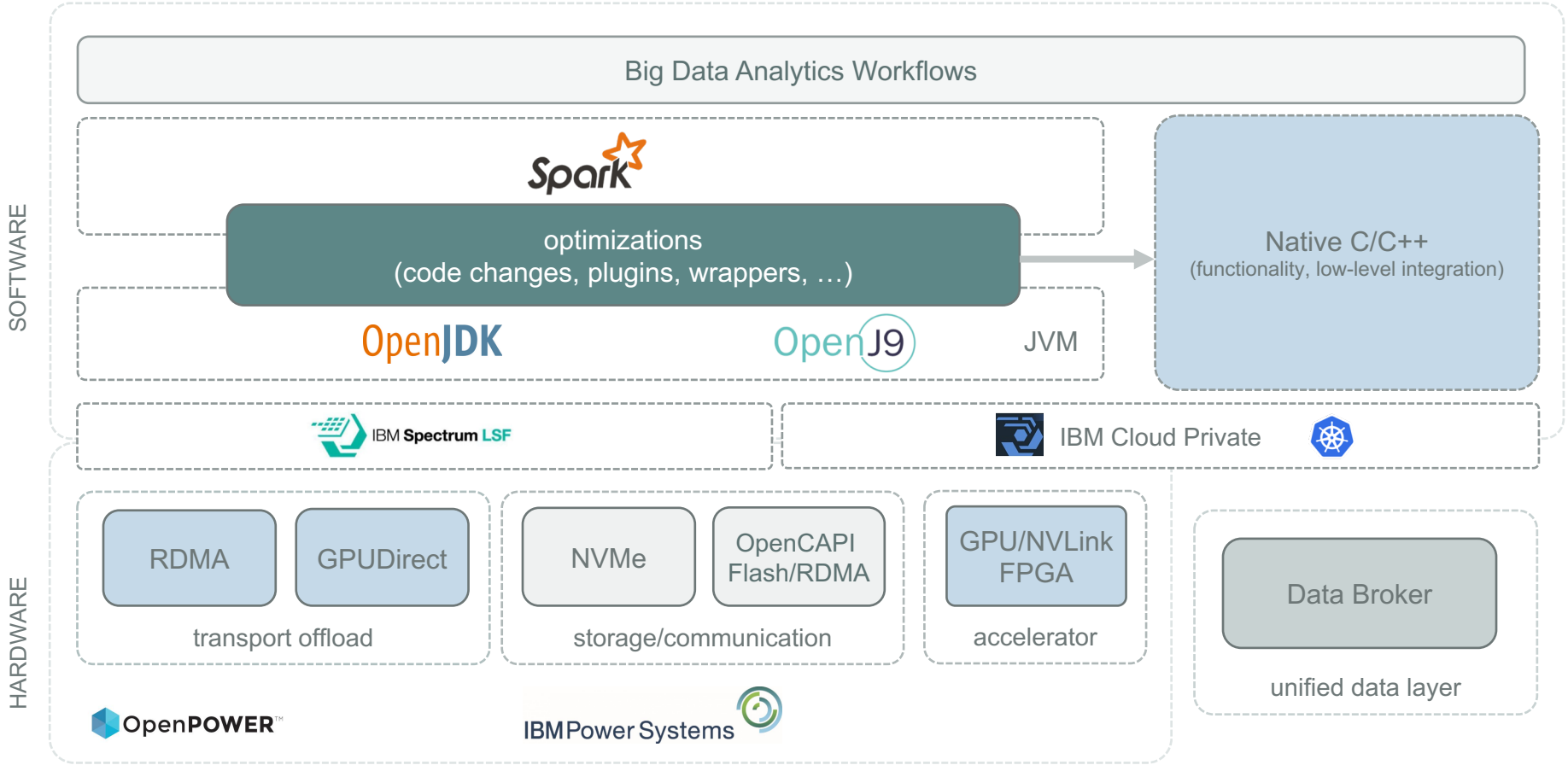
```
points = spark.textFile(...).map(parsePoint).cache()
w = numpy.random.ranf(size = D) # current separating plane
for i in range(ITERATIONS):
    gradient = points.map(
        lambda p: (1 / (1 + exp(-p.y*(w.dot(p.x)))) - 1) * p.y * p.x
    ).reduce(lambda a, b: a + b)
    w -= gradient
print "Final separating plane: %s" % w
```



Accelerated Middleware for High-End Analytics

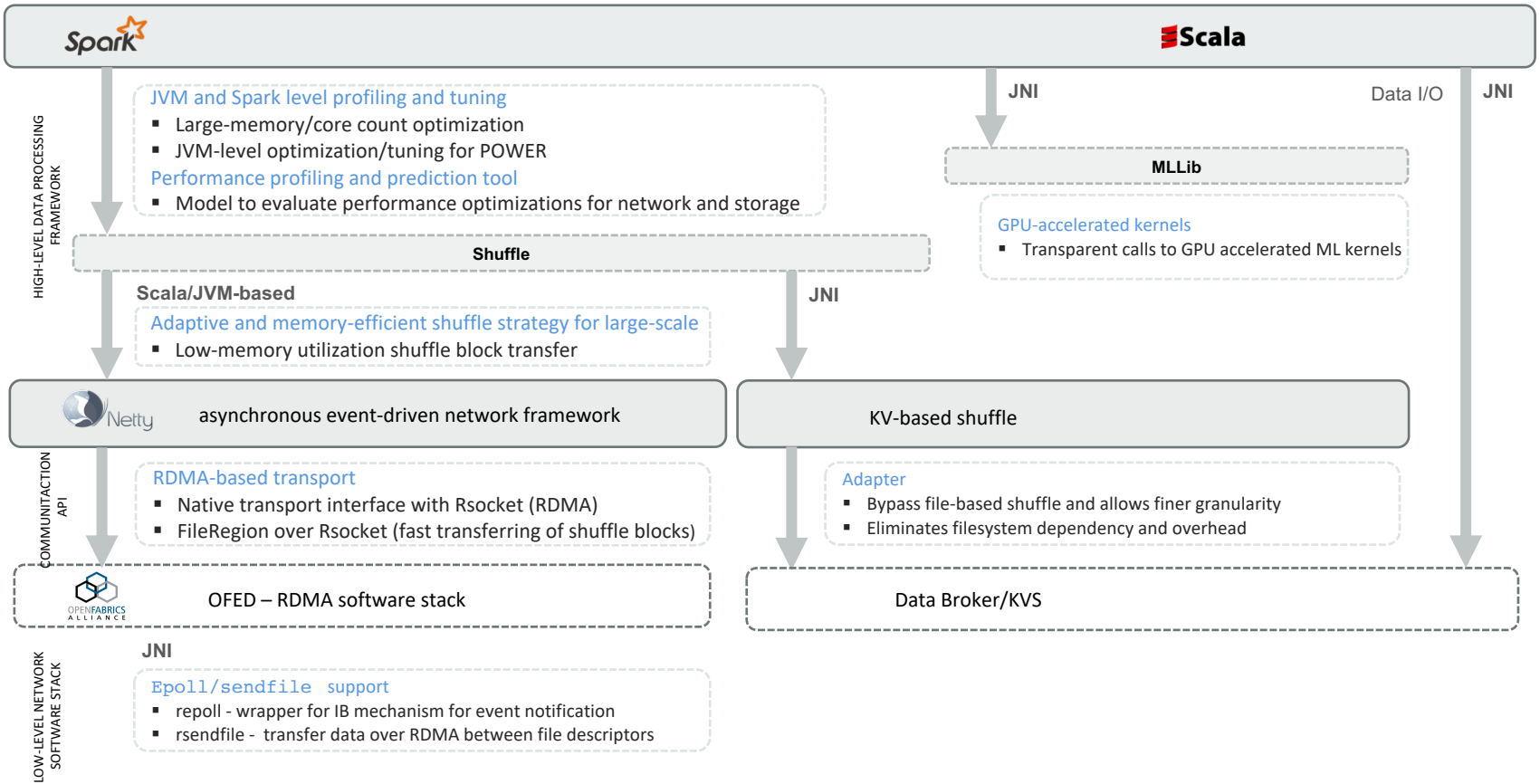
Accelerated and Scalable Middleware for HDA

IBM Research



Transparent Acceleration for HDA on Spark

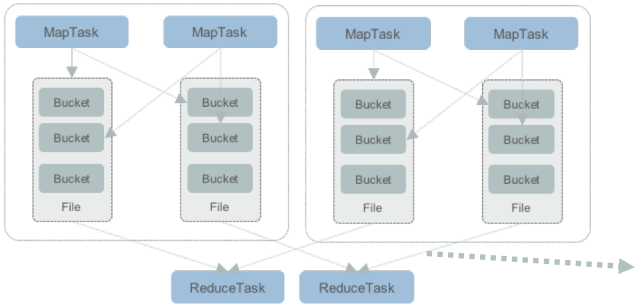
IBM Research

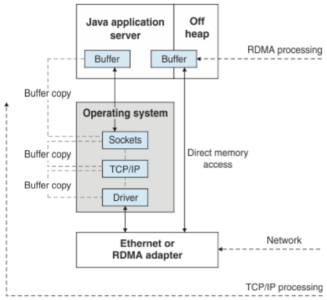


Transparent Transport Acceleration

IBM **Research**

- **RDMA-based shuffle**
  - Replaces Java socket-based transfer of temporary shuffle files



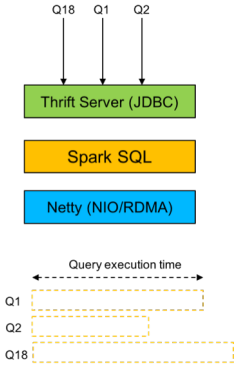


Implementation (two options)

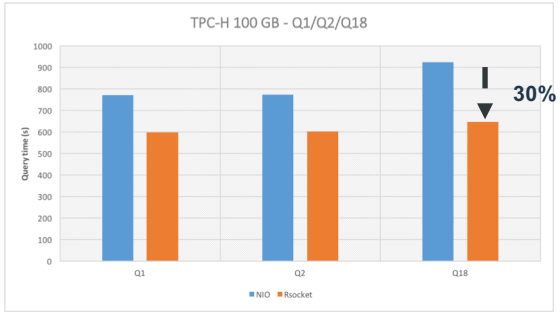
- **Netty RDMA-enabled**
  - Extends native socket support with Rsockets
  - Packaged as a Spark option (`shuffle.io.mode`)
  - Requires custom IBM-provided Netty and OFED

- **IBM OpenJ9 with Java Sockets over RDMA (JSoR)**
  - JVM-level support to transparent switch Java sockets over RDMA (supports NIO)
  - Enabled with Java option (`-Dcom.ibm.nio.rdma`)
  - Fully packaged as a featured in **IBM Java SDK**

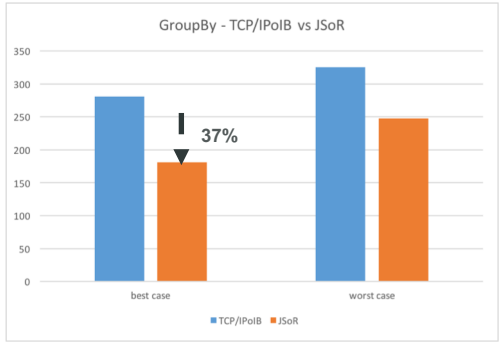
TPC-H/Spark SQL



NIO is unstable, Varying performance (task fail more frequently)



RDMA is more stable (no tasks failed)



- Lower CPU utilization leading to higher throughput
- Same or better performance
- Improved stability (less memory pressure)

32768 (no of key-value pairs) 32768 (key size)

Adaptive I/O to Optimize Collective Data Shuffling

IBM Research

Current approach

- Current implementation is unaware of load distribution and randomizes block requests
- Shuffle data is acquired much faster than it can be consumed with faster transport
- Large amounts of memory used for prefetching shuffle data

Adaptive approach

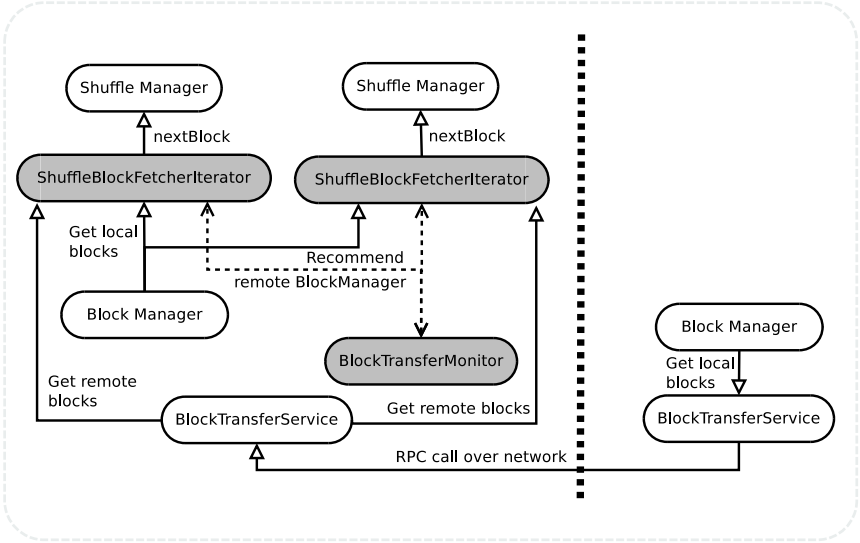
- Mechanism to monitor load and latency based on block request and wait time
- Algorithm to assign a score to each remote host with decentralized coordination
- Dynamically optimizes block fetch requests to minimize latency

Benefits

- Improved network utilization with low memory utilization
- Improved load balancing
- Benefits as a function of number of cores/nodes

Status

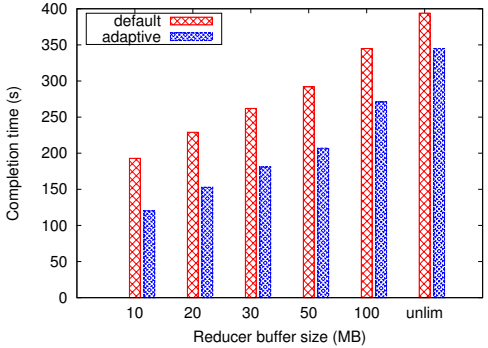
- Patch for Spark 1.5.1, 1.6; 2.0+



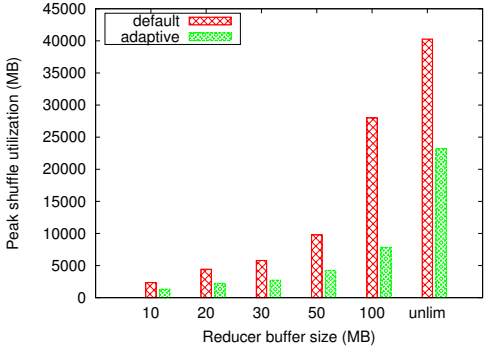
IEEE CCGrid'16 paper [1]

Adaptive I/O to Optimize Collective Data Shuffling

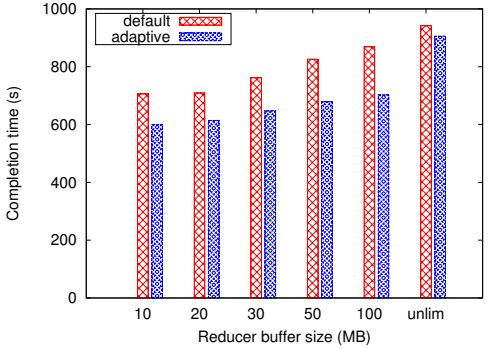
IBM **Research**



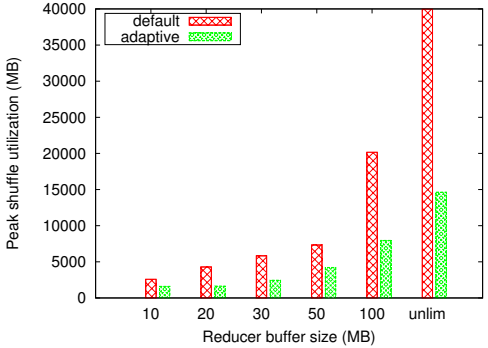
(a) *groupByKey*: Completion time for a variable reducer in-flight limit



(b) *groupByKey*: Peak shuffle memory utilization for a variable reducer in-flight limit



(c) *sortByKey*: Completion time for a variable reducer in-flight limit



(d) *sortByKey*: Peak shuffle memory utilization for a variable reducer in-flight limit

Experimental evaluation

- ~2,240 cores - POWER8 (PowerNV) S822LC (2x SCM 10-core, SMT8)
- Mellanox Infiniband EDR ConnectX-4 adapters (100Gb/s)
- 512 GB of RAM per machine
- 1 TB HDD
- Shuffle I/O: **1TB, 1.4TB** (GroupBy, SortBy)

Raw shuffle performance

- **GroupByKey**: raw performance of groupBy
- **SortByKey**: raw performance of sortBy

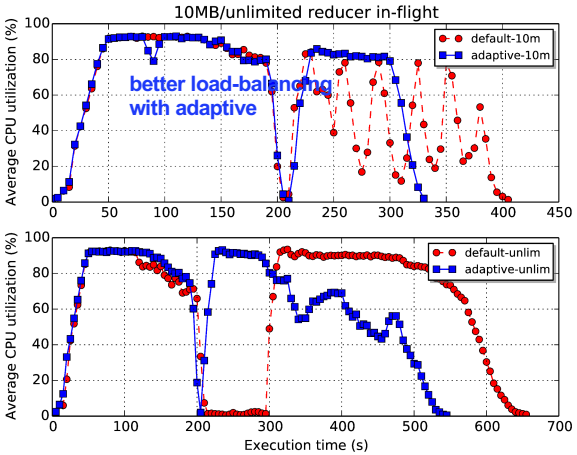
Results

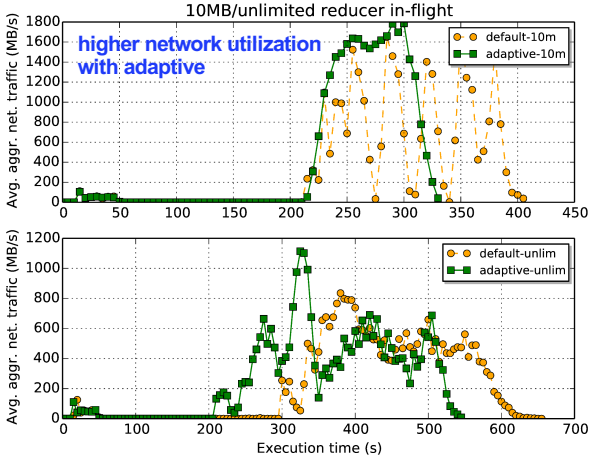
- **up to 40% faster and 50% less memory utilization**

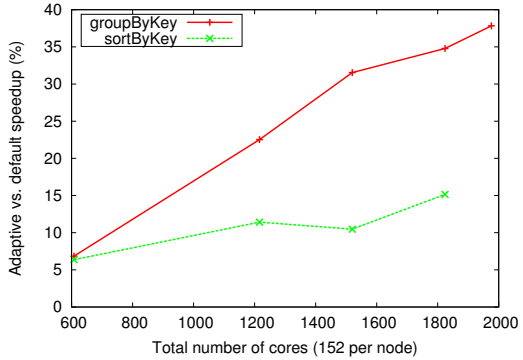
IEEE TPDS'17 article [2]

Adaptive I/O to Optimize Collective Data Shuffling

IBM **Research**







(a) Speedup of completion time for *adaptive* vs. *default*

Experimental evaluation

- ~ 2,240 (PowerNV) S822LC (2x SCM 10-core, SMT8; total 160 hw threads)
- Mellanox Infiniband EDR ConnectX-4 adapters (100Gb/s)
- 512 GB of RAM per machine
- 1 TB HDD

Raw shuffle performance

- **GroupByKey**: raw performance of groupBy
- **SortByKey**: raw performance of sortBy

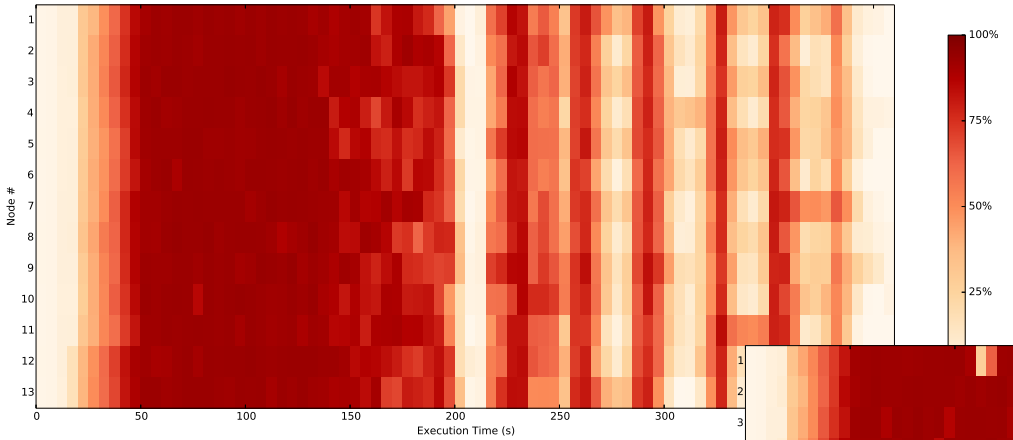
Results

- Increase in speed-up with increasing size of Spark deployment
  - greater **speed-up** with increasing number of cores
  - Alleviates memory pressure, leading to better load balance
- Even more benefits at larger scale both for performance and memory utilization

IEEE TPDS'17 article [2]

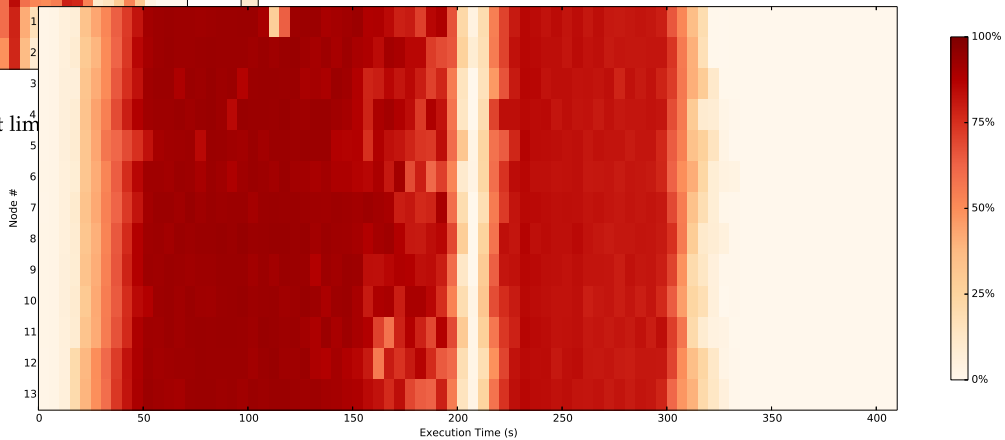
Adaptive I/O to Optimize Collective Data Shuffling

IBM Research



(a) Default strategy using 10 MB reducer in-flight lim

- Load balancing
- Better resource allocation



(b) Adaptive strategy using 10 MB reducer in-flight limit

GroupByKey

2,240 POWER8 (PowerNV) S822LC (2x SCM 10-core, SMT8; total 160 hw threads)

- Mellanox Infiniband EDR ConnectX-4 adapters (100Gb/s)
- 512 GB of RAM per machine
- 1 TB HDD

IEEE TPDS'17 article [2]

Optimizing the JVM for Analytics

IBM Research

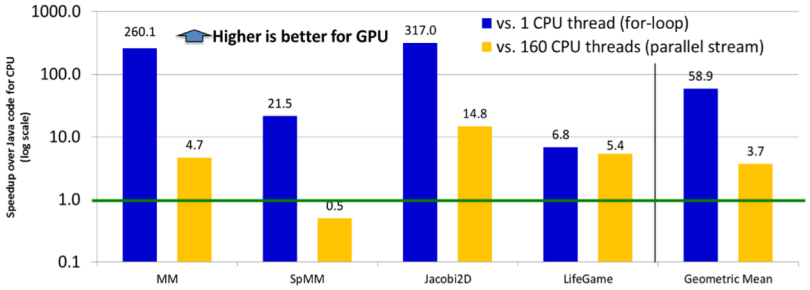
- OpenJ9: high-performance JVM tuned for analytics
- J9 improvements with significant impact on Spark performance
  - Java object models with smaller memory footprint
  - More efficient garbage collection technologies (*gencon* with Concurrent Scavenge)
  - More efficient JVM lock contention schemes
  - Just-in-time compilation (Testarossa JIT compiler)
  - GPU-enabled JIT
  - Shared classes technology (stores ahead of time (AOT) compiled code)

OpenJ9

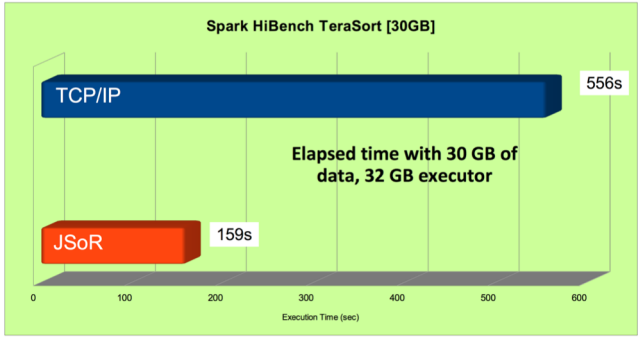
https://www.eclipse.org/openj9/

- IBM Java SDK enhancements
  - Transparent RDMA acceleration (Java Sockets over RDMA - JSoR)

JIT with GPU-offload



Java Sockets over RDMA (JSoR)



32 cores (1 master, 4 nodes x 8 cores/node, 32GB Mem/node), IBM Java 8

Enabling Transparent GPU-acceleration

IBM **Research**

GPU/FPGA-enable Spark Apps

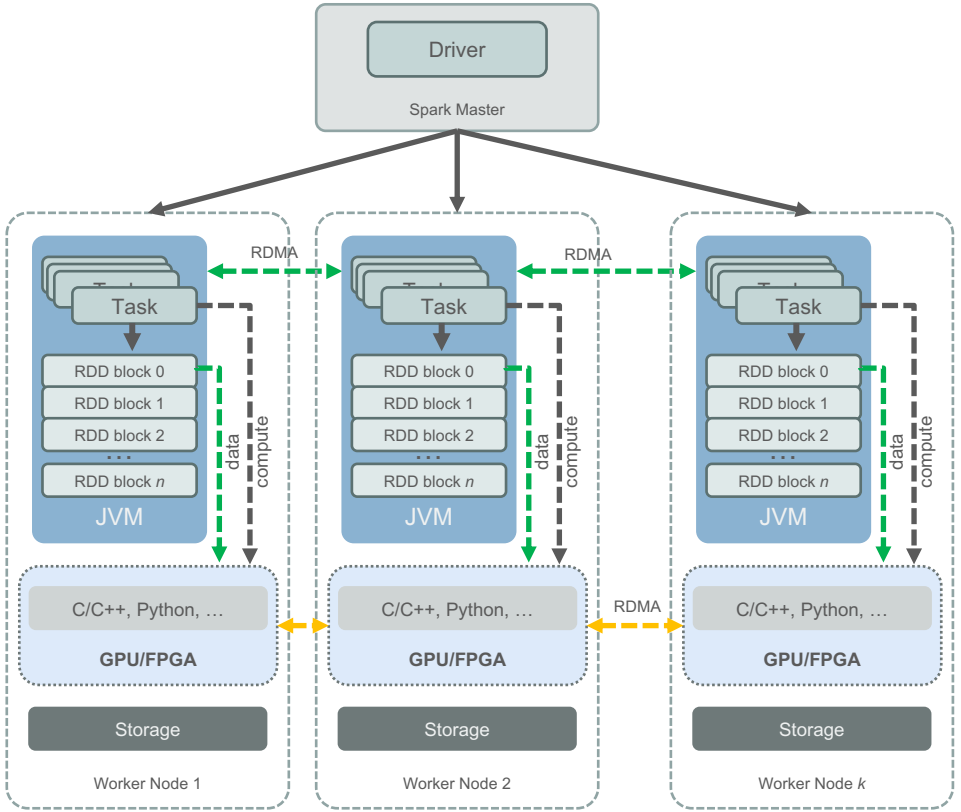
- Off-load compute-intensive kernels in Spark's native apps/libraries
  - GPU-enable MLLib/SparkSQL/GrapX
  - e.g., IBM SparkGPU: GPU code generation for Tungsten; CUDA code GPU accelerated MLLib algorithms)
  - GPU/FPGA acceleration of genomics kernels (FPGA-based PairHMM in GATK4-Spark)

Spark-enable GPU accelerated Apps Spark Apps

- Distributed deep learning (e.g. Spark port of existing GPU-enabled frameworks like TensorFlow, Caffe, Theano, …)
- Third-party frameworks (e.g H_2O) and home-grown apps in financial services

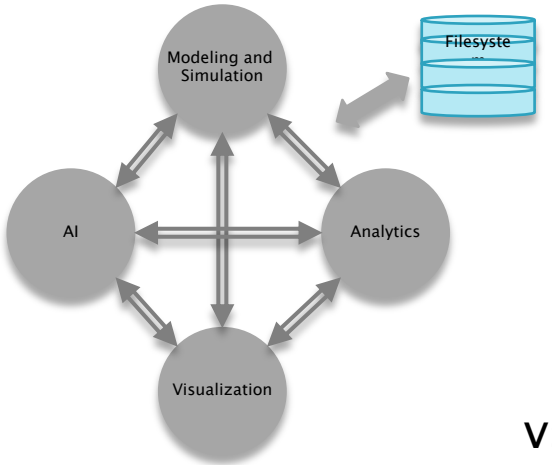
Optimizations

- Reduced communication overhead for Spark-native apps (data layer in the JVM path – e.g., MLLib)
- Efficient direct communication among accelerators (e.g., Spark used for job distribution with compute and communication off the JVM path)

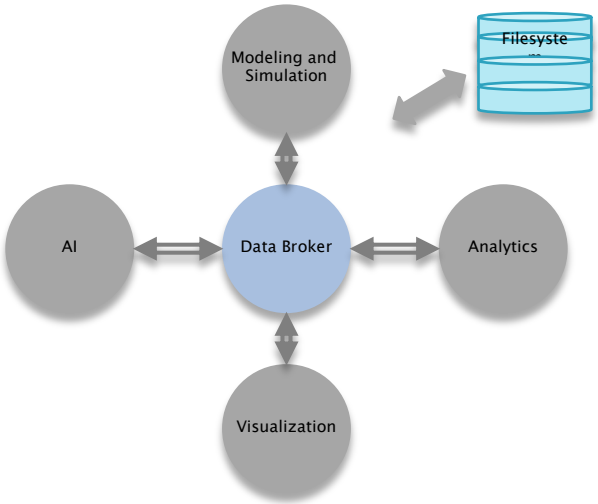


https://github.com/IBMSparkGPU

Data Broker



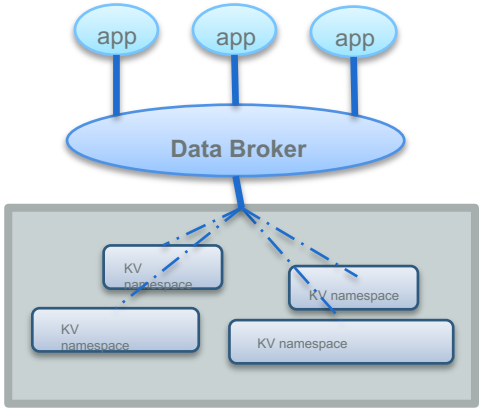
vs.

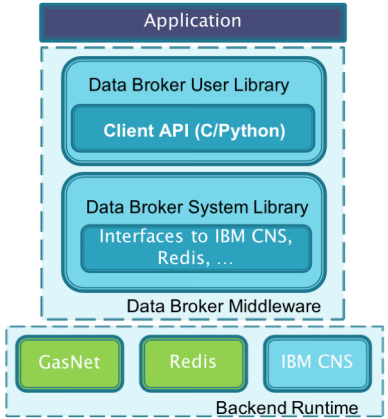


Files IO
- Longer latency
- Less granularity

Sockets
- Longer latency
- Multiple sockets per application
- Discovery for new apps is complicated

Data Broker
- Each app connects to data broker
- Distributes data in DRAM over multiple nodes
- Latency generally lower
- Data Broker can be accelerated via H/W
- Discovery of apps via data broker





https://github.com/IBM/data-broker

Data Broker

Data Broker Management Functions

Name	Description
dbrCreate()	Create a new Data Broker
dbrDelete()	Delete an existing Data Broker
dbrAttach()	Attach to an existing Data Broker
dbrDetach()	Detach from an existing Data Broker
dbrQuery()	Query information about an existing Data Broker
dbrTest()	Test the status of an asynchronous call
dbrCancel()	Cancel an asynchronous call

Data Broker Access Functions

Name	Description
dbrPut()	Insert a tuple in the Data Broker
dbrRead()	Read a tuple in the Data Broker
dbrGet()	Pop a tuple in the Data Broker
dbrReadA()	Read a tuple in the Data Broker, non blocking
dbrPutA()	Put a tuple in the Data Broker, non blocking
dbrGetA()	Pop a tuple in the Data Broker, non blocking
dbrMove()	Move a tuple from a source namespace to a destination namespace

Benchmarking: 1M requests, key size 1K, data size 4K, 5 nodes

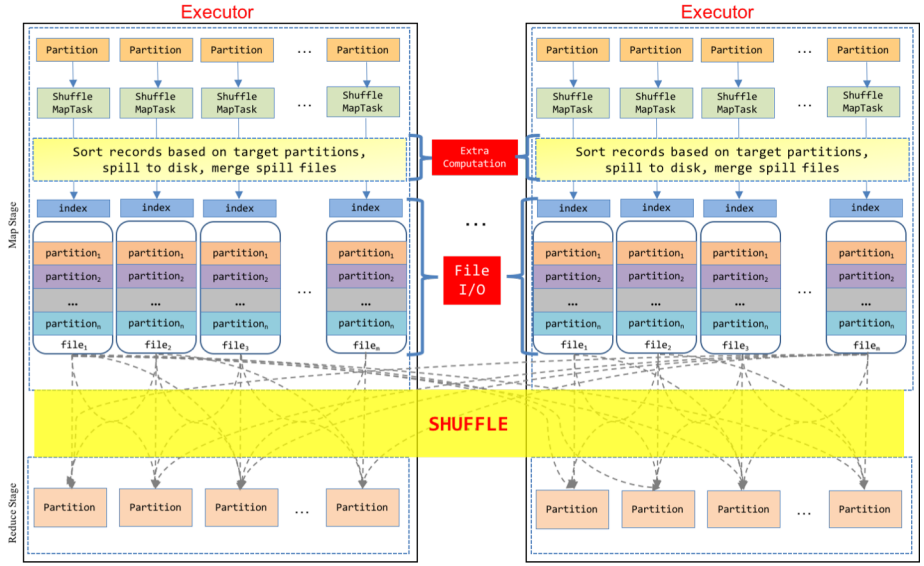
Put	Get	Read
Time avg/p: 81774.3ms	Time avg/p: 61279.5ms	Time avg/p: 48877.2ms
requests: 1.6e+07	requests: 1.6e+07	requests: 1.6e07
IOPS: 195660	IOPS: 261099	IOPS: 327351
BW: 801.425MB/s	BW: 1069.46MB/s	BW: 1340.83MB/s
min: 427.197ms	min: 451.922ms	min: 345.789ms
max: 2206.72ms	max: 1618.8ms	max: 902.828ms

jsrun --nrs 160 --rs_per_host 32 parallel -d 4096 -k 1024 -p 1000 -t READ/PUT/GET -n 100000

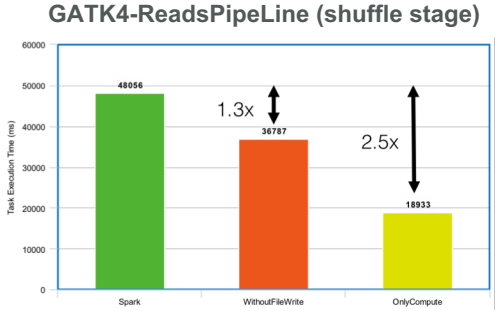
Enhancing Spark with Data Broker

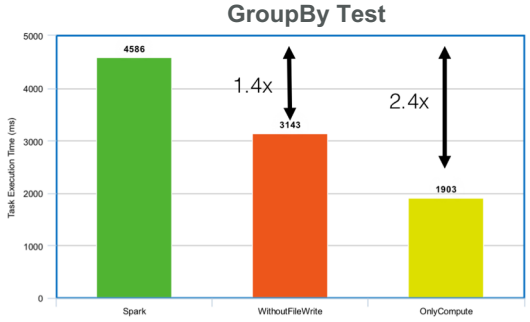
IBM Research

- Filesystem dependency and IO limitations
- Current shuffle implementation stores data in blocks on local disk I/O for data shuffling
- Major overhead on the OS
  - both the source and the destination side requires many file and network I/O operations
- Data aggregation techniques are used for filesystem and communication optimization but this adds extra computation overheads



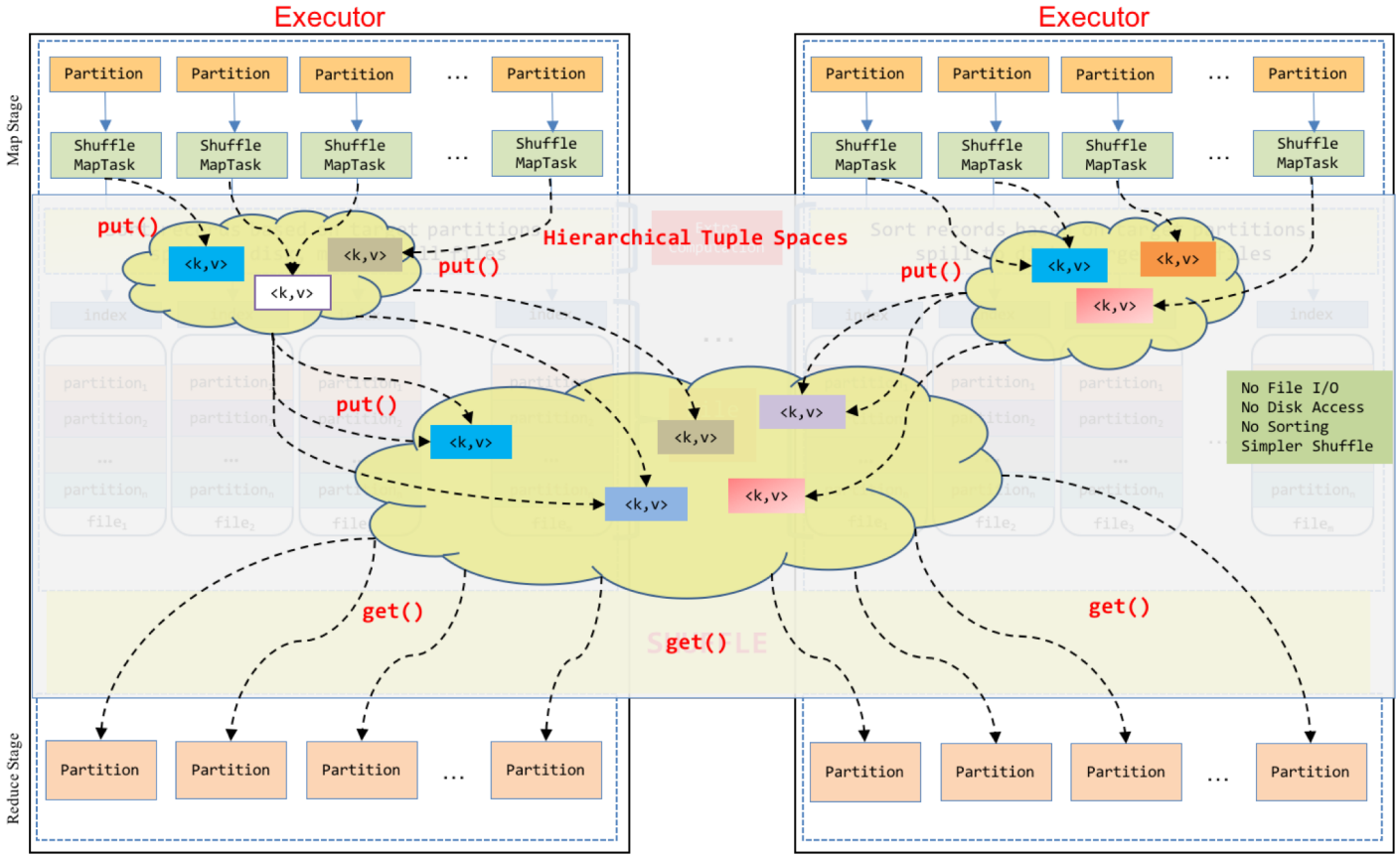
Acceleration opportunity





Enhancing Spark with Data Broker

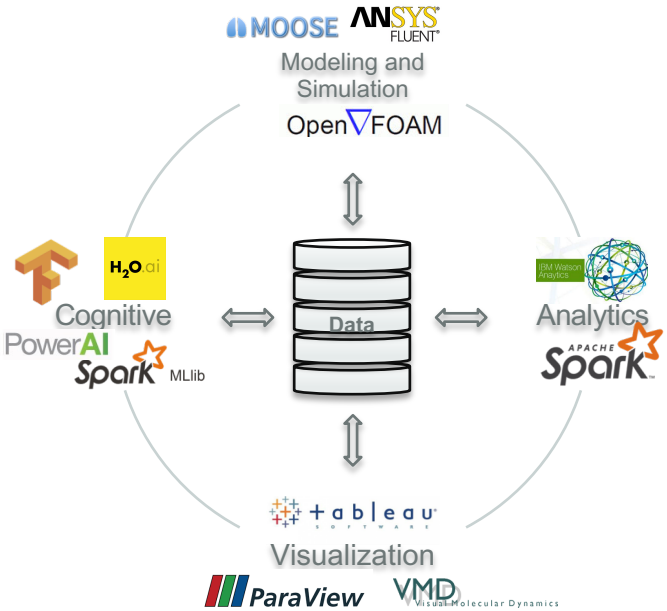
IBM **Research**



Enhancing Spark with Data Broker

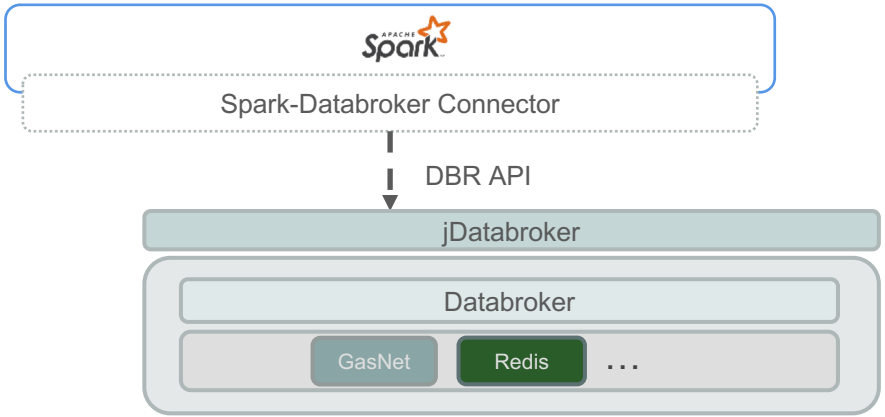
IBM **Research**

- Implementation of Data Broker concept used to accelerate shuffle and enable efficient data management across frameworks
- Bypass of filesystem with data sharing based on tuples



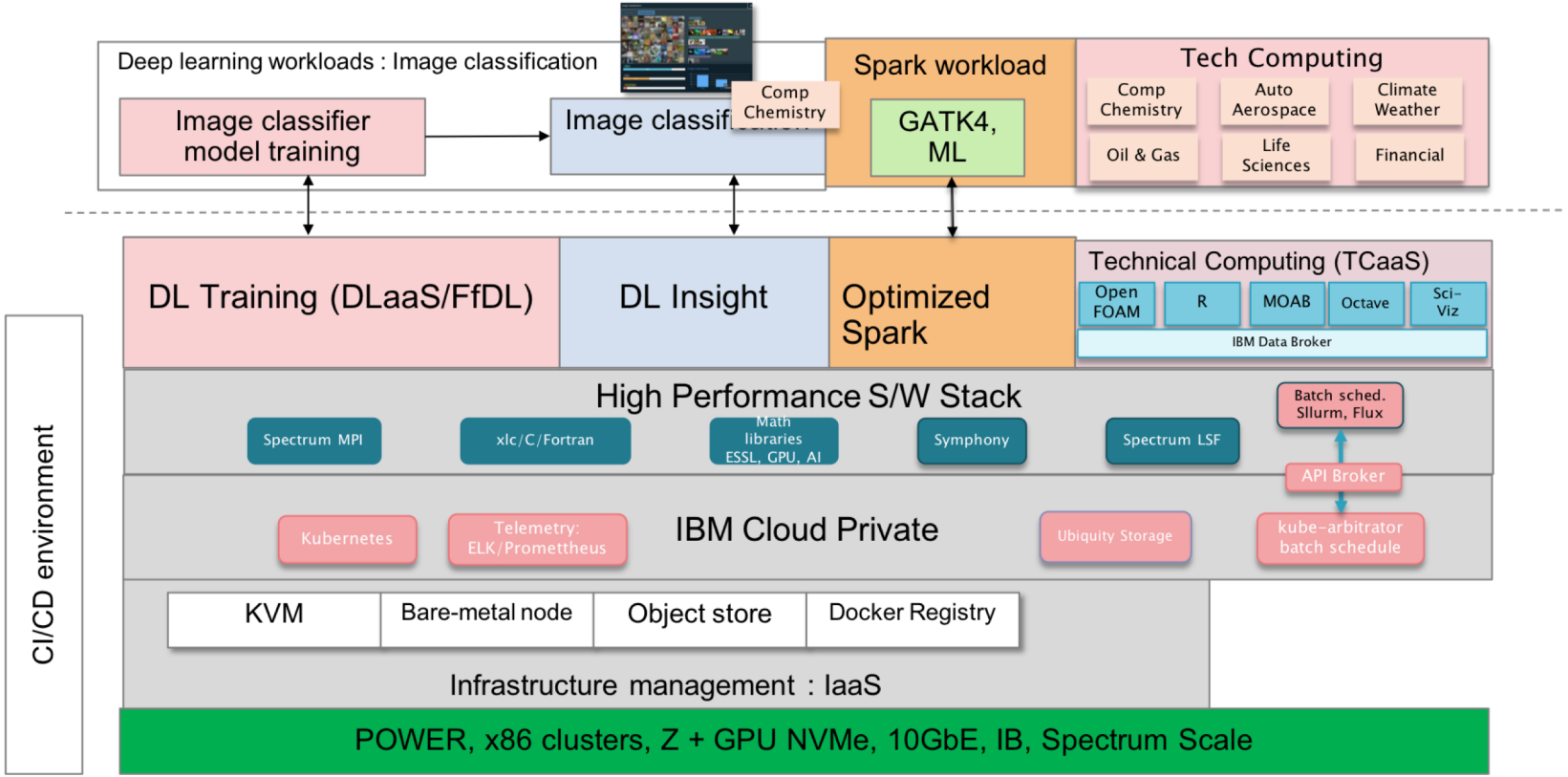
Spark-Data Broker adapter

- Overwrites shuffle related classes
- Software-based Tuple Space implementation
- Customized Spark Shuffle Write/Read operations through jDatabroker
- API for reading/writing to/from the Data Broker



Ecosystem for High Performance Workloads

IBM Research



HPC Benchmarks on a Cloud Stack

IBM **Research**

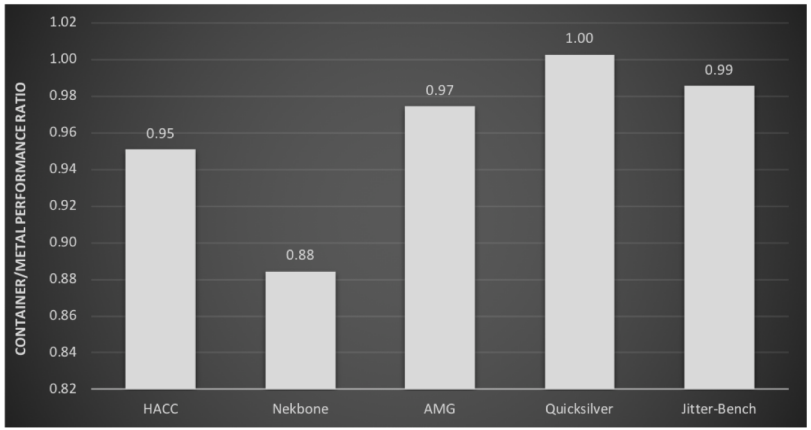
CORAL2 Benchmarks on ICp *(selected)*

- **HACC** ***(MPI/OpenMP/C++)***
  - Compute intensity, random memory access, all-to-all communication
- **Nekbone** ***(MPI/Fortran/C)***
  - Compute intensity, small messages, all-reduce
- **AMG** ***(MPI/OpenMP/C)***
  - Algebraic Multi-Grid linear system solver for unstructured mesh physics packages.
- **Quicksilver** ***(MPI/OpenMP/C++)***
  - Monte Carlo transport benchmark

- Demonstrated relevant HPC benchmarks in a container environment
- Measured small performance degradation in comparison with bare metal

Container vs bare metal performance comparison

SUMMARY (16 nodes)			
Bench	**ICp (metal)**	**ICp (containers)**	**Factor**
HACC	7.07	7.44	0.95
Nekbone	8.75E+02	7.74E+02	0.88
AMG	1.16E+10	1.13E+10	0.97
Quicksilver	1.58E+06	1.59E+06	1.00
Jitter-Bench	4.21%	4.27%	0.99



Enabling Large-Scale Analytics for Hybrid Workflows

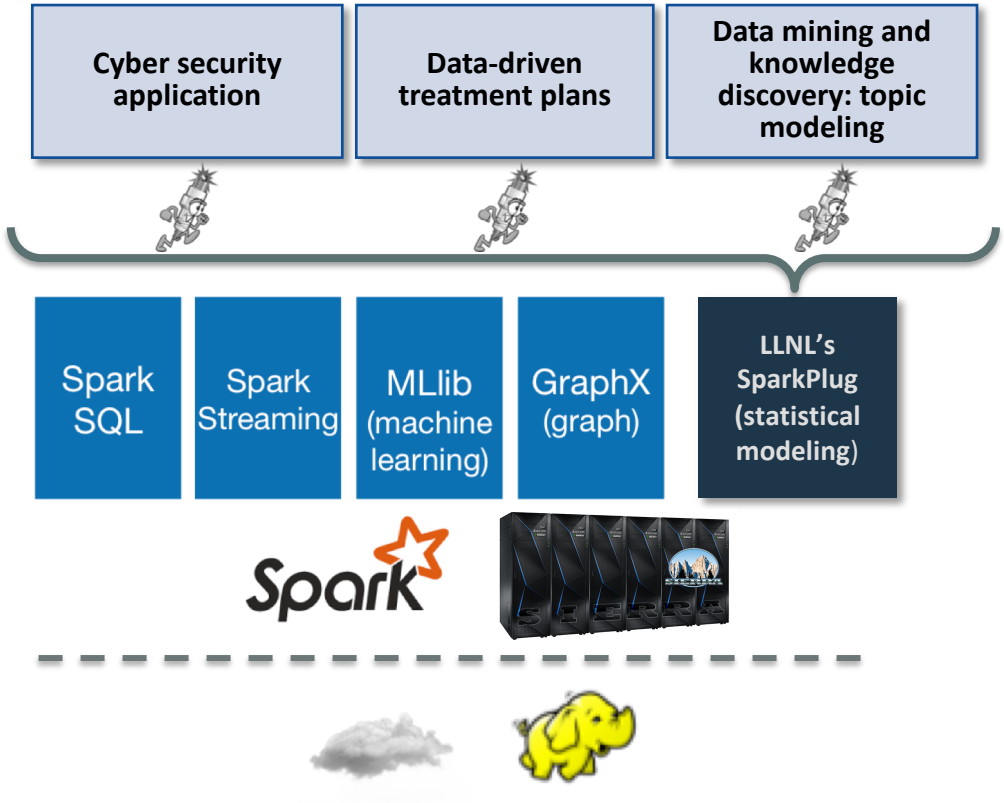
Knowledge Discovery at Scale

LLNL/IBM collaboration
LLNL: Barry Y. Chen, Grant M. Bouquet
IBM: Carlos Costa, Claudia Misale, Guojing Cong

IBM Research

LLNL's SparkPlug

- Addressing recurrent data challenges in mission application
  - Huge data sets, sparse labels, heterogeneous, complex structure
- Density estimation toolbox for big data machine learning at scale
  - Distributions, estimators, combinators, graphical model templates, samplers
- Implemented as a Spark library allowing modeling without requirement advanced software development background
- Allows complex models able to utilize application specific understanding
- Scalable design supports large data sizes
- Current collaboration focusing on the scaling of **SparkPlug's LDA implementation** on LLNL's Sierra



Knowledge Discovery at Scale

LLNL/IBM collaboration
LLNL: Barry Y. Chen, Grant M. Bouquet
IBM: Carlos Costa, Claudia Misale, Guojing Cong

IBM Research

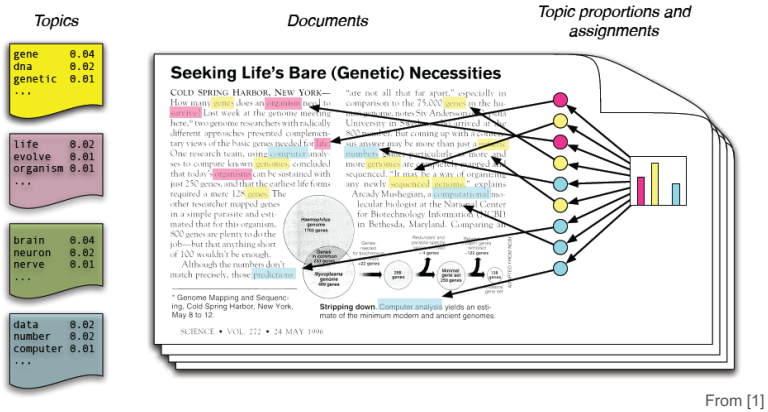
Distributions / samplers
- Beta
- Binomial
- Categorical
- CensoredExponential
- CensoredGeometric
- Composite
- Conditional
- Either
- Exponential
- Gamma
- Geometric
- HMM
- Hierarchical mixture
- Inverse gamma
- Inverse Wishart
- Markov chain
- Multinomial
- Multinomial logistic regression
- Multivariate Gaussian
- Negative binomial
- Normal inverse gamma
- Pareto
- Poisson
- Product
- Two-level mixture
- Uniform
- Univariate Gaussian
- von Mises
- Zero-altered negative binomial
- Zero-altered Poisson

- EM estimators
  - Binomial
  - Categorical
  - CensoredExponential
  - CensoredGeometric
  - Composite
  - Conditional
  - Either
  - Exponential
  - Gamma
  - Geometric
  - HMM
  - Hierarchical mixture
  - Linear regression
  - Markov chain
  - Multinomial
  - Multinomial logistic regression
  - Multivariate Gaussian
  - Negative binomial
  - Pareto
  - Poisson
  - Product
  - Two-level mixture
  - Uniform
  - Univariate Gaussian
  - von Mises
  - Zero-altered negative binomial
  - Zero-altered Poisson
- MixtureModel
- HierarchicalMixtureModel
- Clustering
  - K-means
- MCMC estimators
  - Binomial
  - Categorical
  - Composite
  - Dirichlet
  - Exponential
  - Geometric
  - Multinomial
  - Multinomial logistic regression
  - Multivariate Gaussian
  - Pareto
  - Poisson
  - Uniform
  - Univariate Gaussian
- Distributed MCMC
  - Neiswanger nonparametric
  - Neiswanger semiparametric
  - Neiswanger parametric
  - Dunson median posterior

Knowledge Discovery at Scale

LLNL/IBM collaboration
LLNL: Barry Y. Chen, Grant M. Bouquet
IBM: Carlos Costa, Claudia Misale, Guojing Cong

IBM **Research**



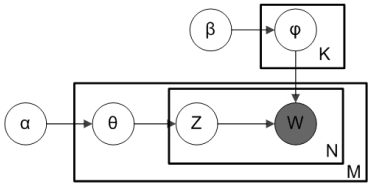
From [1]

Topic Modeling

- Used in text-mining and detection of hidden instructive structures in data such as genetic information, images and networks
  - e.g. What is document A discussing? How similar are documents A and B? If I am interested in topic X, which documents should I read first?

Latent Dirichlet Allocation (LDA)

- Generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar
- Widely used clustering/latent factors model and commonly part of analytics pipelines

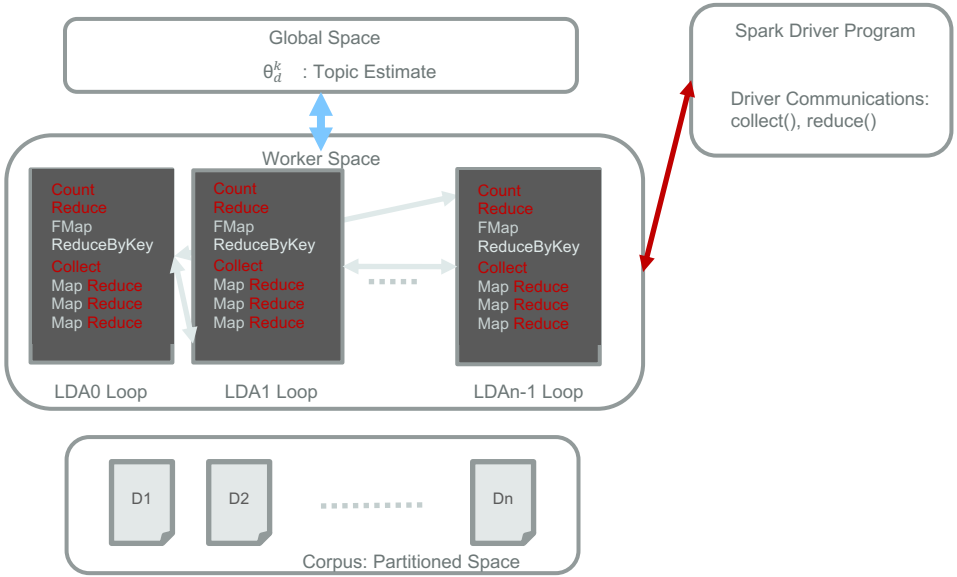


Define $\Theta_i \sim Dir(\alpha) : i \in \{1, \ldots, M\}$, with $Dir(\alpha)$ a Dirichlet distribution;
Define $\varphi_k \sim Dir(\beta) : k \in \{1, \ldots, M\}, j \in \{1, \ldots, N_i\}$;
For each word $w_{i,j} : i \in \{1, \ldots, M\}, j \in \{1, \ldots, N_i\}$:

- Choose topic $z_{i,j} \sim Multinomial(\Theta_i)$
- Choose word $w_{i,j} \sim Multinomial(\varphi_{z_{i,j}})$

α: parameter of the Dirichlet prior on the per-document topic distribution;
β: parameter of the Dirichlet prior on the per-topic distribution;

Θ_m: topic distribution over the document m;
φ_k: word distribution over the topic k;
Z_{nm}: topic for the n-th word in document m;
w_{nm}: specific word n in document m.



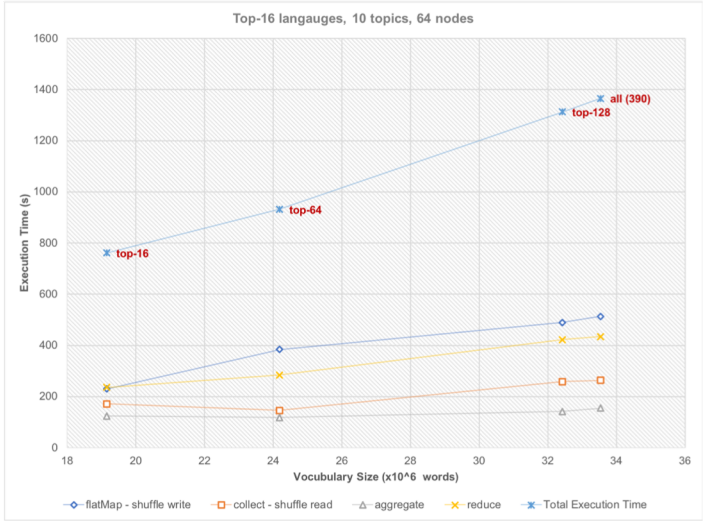
Knowledge Discovery at Scale

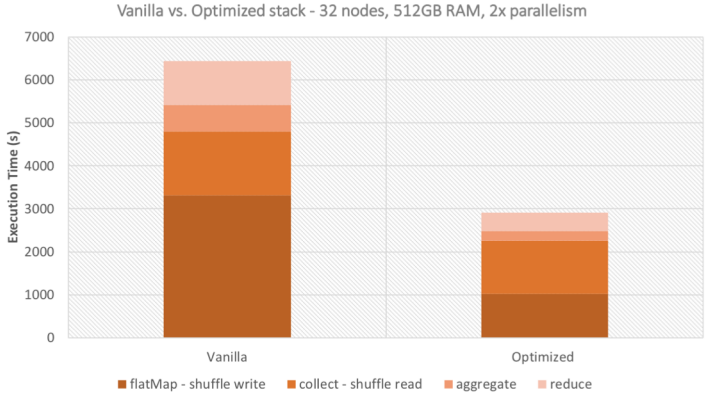
LLNL/IBM collaboration
LLNL: Barry Y. Chen, Grant M. Bouquet
IBM: Carlos Costa, Claudia Misale, Guojing Cong

IBM **Research**

Large-scale Latent Dirichlet Allocation (LDA)

- Demonstrated large-scale topic modeling with all-languages Wikipedia database on LLNL's Sierra
  - 300+ raw dump files from Wikimedia.org (all languages)
  - ~34 million unique words (including non-Latin alphabets)
  - 100 topics
  - Sierra runs: up to ~20,000 cores
- Significant acceleration with optimized stack (2x faster)
- Improved scaling
- Optimized all-to-one operations (treeAggregate)
- On-going work on further scaling improvements for collect operation





Genomics Analysis Toolkit (GATK4)

IBM Research

MIT/Harvard - world's leading biomedical and genomic research center

GATK: High-throughput Sequencing (HTS) data analysis workflow

- Pipeline of a variety of tools with a primary focus on variant discovery and genotyping
- 31,000 registered users of GATK (providers, clinicians, research centers, focused on **personalized medicine**)
- Requires extensive local compute and storage infrastructure to process vast amount of data required to conduct personalized analysis

Next-Gen: GATK4 - Spark-based GATK for Cloud-Based Access

- Spark being used to facilitate parallelism and in-memory computation to speedup methods
- Broad Institute working with collaborators to develop **scalable** options to expand access and facilitate usage
- Critical to accelerate precision medicine by lowering the **cost** of genome sequencing
- GATK4 will extend the range of use cases supported to include cancer, structural variation, copy number variation, and more
- Genome analysis can be seen as one step in potentially larger workflows applying machine learning to personalized **treatment plans**

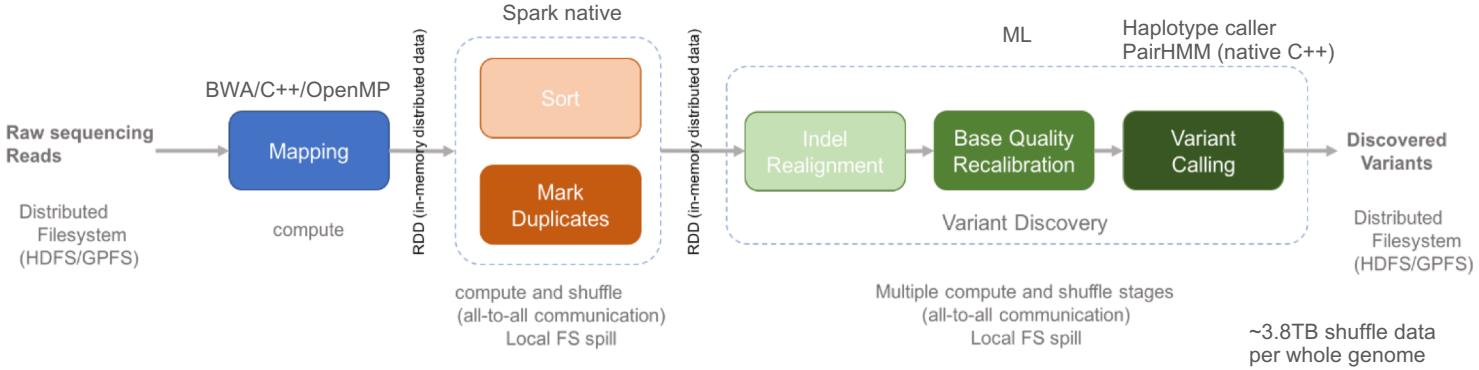
GATK4 Collaboration Members

https://www.broadinstitute.org/news/8065

GATK4-Spark Pipelines on POWER

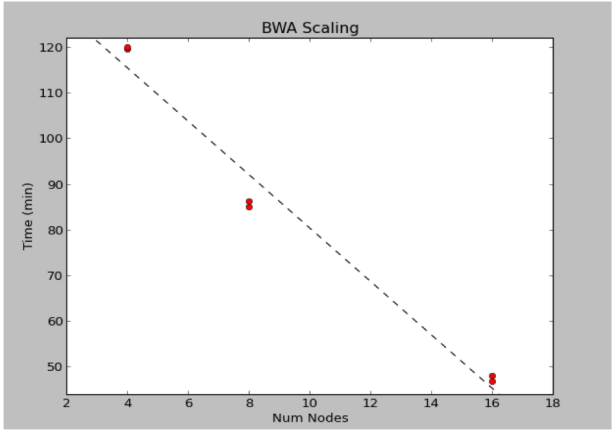
IBM Research

SNP pipeline



- Heterogenous pipeline (Spark with calls to native code OpenMP and accelerators)
- Significant speedup with
- **Strong scaling** for BWA (dominant stage)
- Better performance with higher memory bandwidth and SMTs on POWER

Burrows-Wheeler Aligner (BWA)



Input: G15512.HCC1954.bam (WGS)

POWER8 Minsky 8335-GTB
2-socket, 10-core SCM
512GB
2.9 TB Local NVMe
InfiniBand EDR (100Gb/s)

Spectrum LSF (Spark standalone scheduler)
Spectrum ESS

GATK4 Genomics Workflow on the Cloud

IBM Research

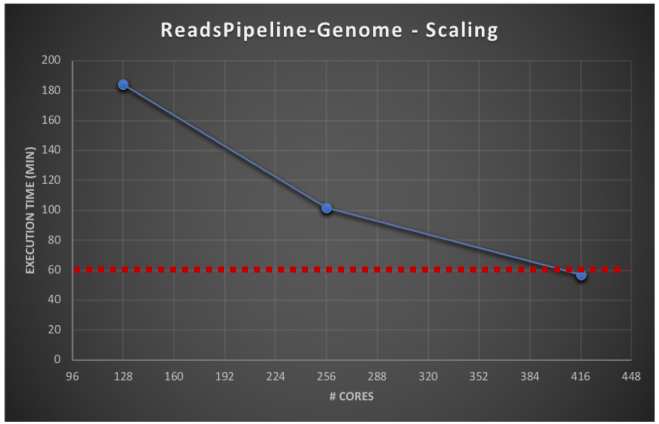
GATK4-Spark on IBM cloud platform

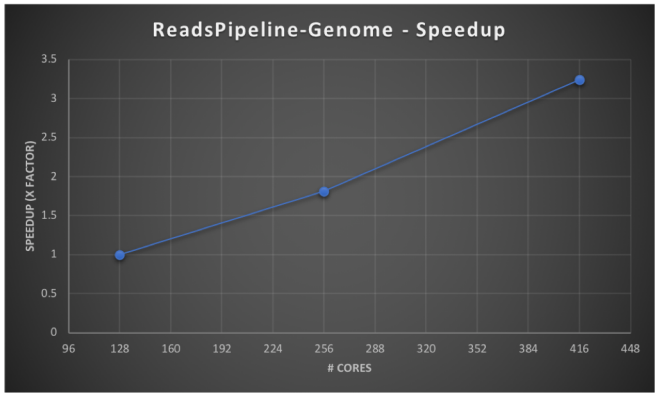
- Scalable SNP pipeline with optimized Spark and JVM for efficient resource utilization
- Benefits from high-bandwidth network and Spark configuration and JVM tuning and enhancements with Eclipse OpenJ9 (efficient garbage collection, optimized Just-in-time compilation, JVM lock contention schemes, …)

Whole genome SNP processing in less than <1hr

- **~32hrs:** GATK3-Walker mode (single node)
  - Single node: 36 cores (Intel Xeon processor E5-2699 v3 @ 2.30 GHz, 256 GB RAM) [1]
- **~57min: GATK4-Spark on cloud platform**
  - 22 VMs: 18x m1.16x128 + 4x m1.32x128 (416 cores; 3.25TB RAM)

< 1hr whole genome pipeline



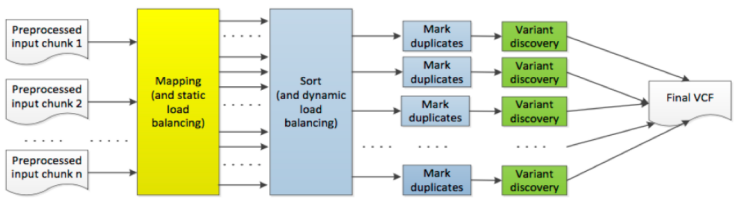


TUDelft Genomics Workflow

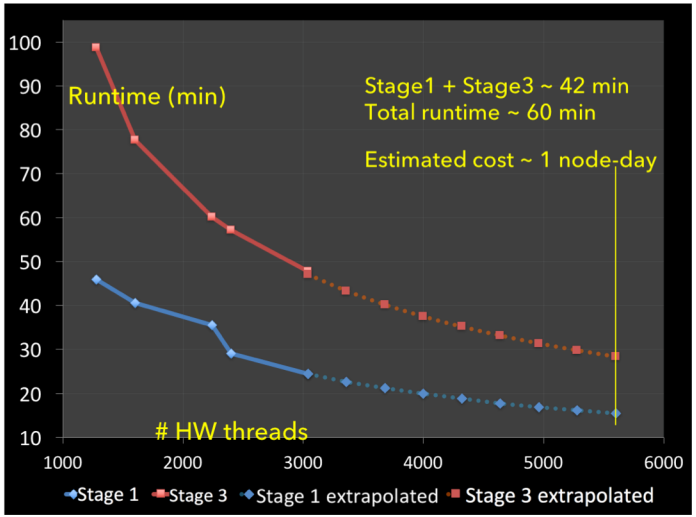
DNA variant discovery uses a pipeline of different tools

- **Phase 1**
- Custom approach from Delft University to parallelize BWA, GATK, and Piccard
- Leveraged native code tuned for POWER (vectorized code for PairHMM)
- Demonstrated whole SNP pipeline at low-cost with POWER at the Spark Summit (first appearance of POWER)
- **Phase 2 (on-going)**
- Extension for population genomics analysis (thousands of full genomes)
- NVLink, heterogeneous CPU-GPU computing kernels
- RDMA-based communication

TUDelft



- 90 min runtime on 20-node cluster
- Stage 1 & 3 are scalable
- Runtime scales down to 1 hour for 35 node cluster



ACM BCB'17 paper [3]

SPARK SUMMIT 2016

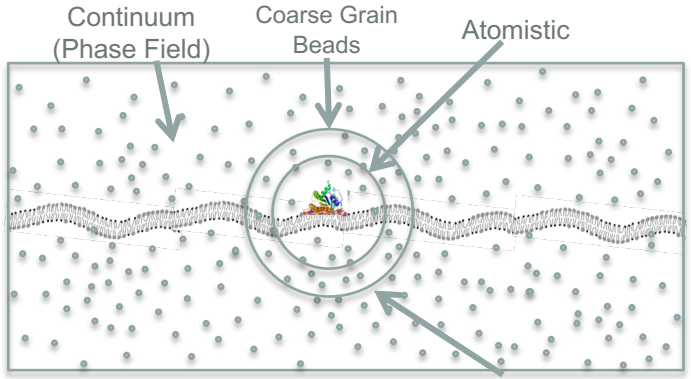
https://spark-summit.org/2016/events/a-spark-framework-for-100-1-hour-accurate-personalized-dna-analysis-at-scale/

LLNL's Precision Medicine: SPLASH Workflow

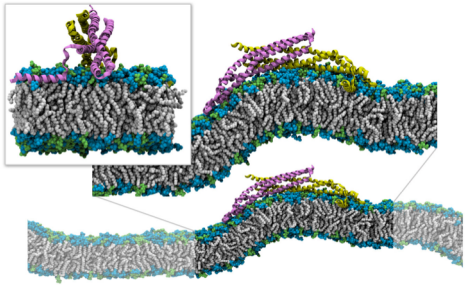
IBM **Research**

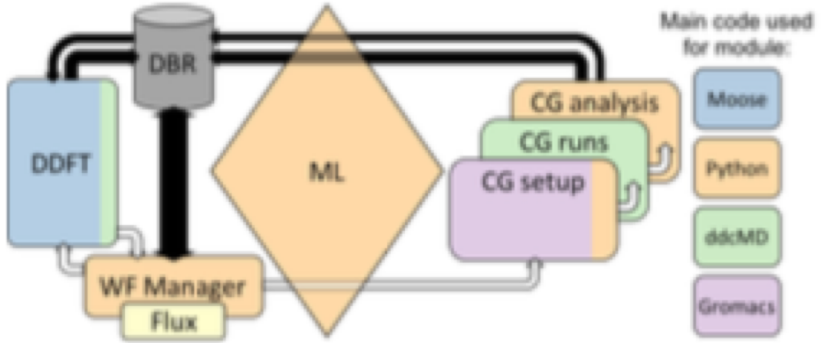
Simulate the Bending of Lipid Cell membrane

- Impacts how molecules (drugs) enter the cell
- Surrounding environment impacts how membrane bends



Accelerate particles with parallel replica dynamics





Multiscale code framework:

- The WorkFlow (WF) Manager connects two scales: Dynamic Density Functional Theory (DDFT) and coarse grain (CG)
- Frames resulting from the DDFT simulation are decomposed into patches, and the WF Manager feeds them to the machine learning (ML) infrastructure, which maintains a priority queue of candidate patches.
- When new resources become available, the WF Manager picks top candidates and uses the Flux resource manager to start new CG simulations.
- Data transfer and messaging is handled through the DataBroker (DBR), which implements a fast, system-wide key-value store. Thickness of black arrows represents the bandwidth of data flow to and from the DBR.

Other Use Cases: Engineering

IBM Research

Simulation Data:
- Inputs / Execution Snapshots
- Various Grid Mappings for different components of workflow

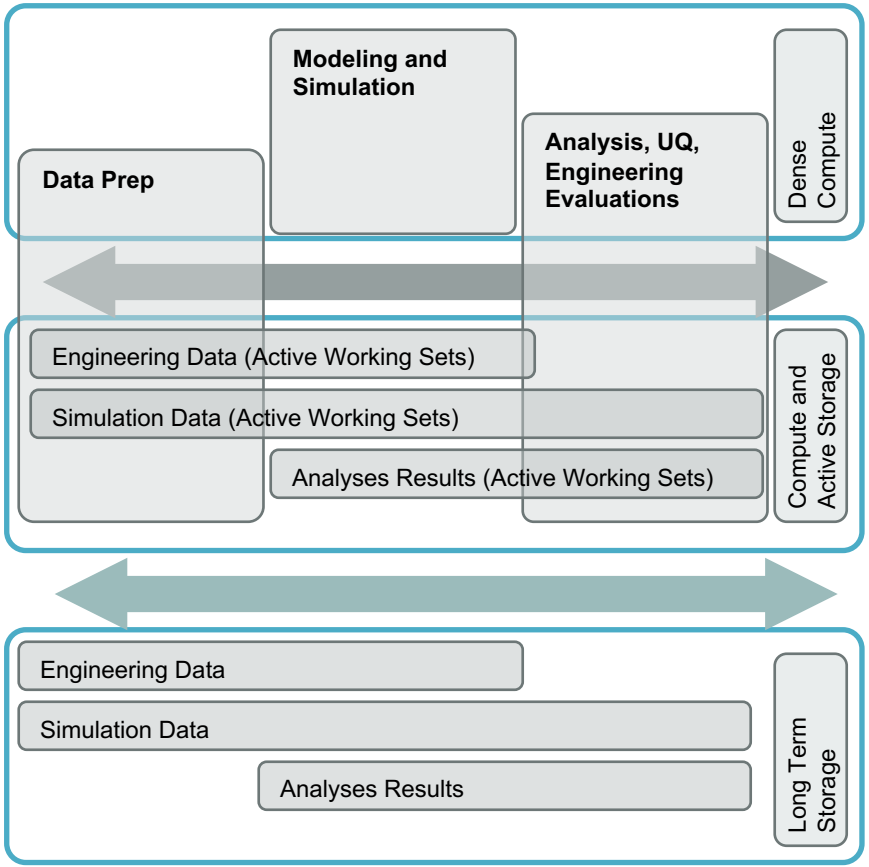
Engineering Data:
- Designs
- Experimental Results

Analysis Results
- Visualizations
- UQ and Sensitivity Data
- Engineering Analyses

Data Prep
- Mesh Generation
- Model setup, initial conditions

Modeling and Simulation
- Computational Modeling

Analysis, UQ, Engineering Evaluation
- Visualization
- Post Processing
- UQ, Sensitivity Analysis
- Engineering Analyses



Climate Use Case

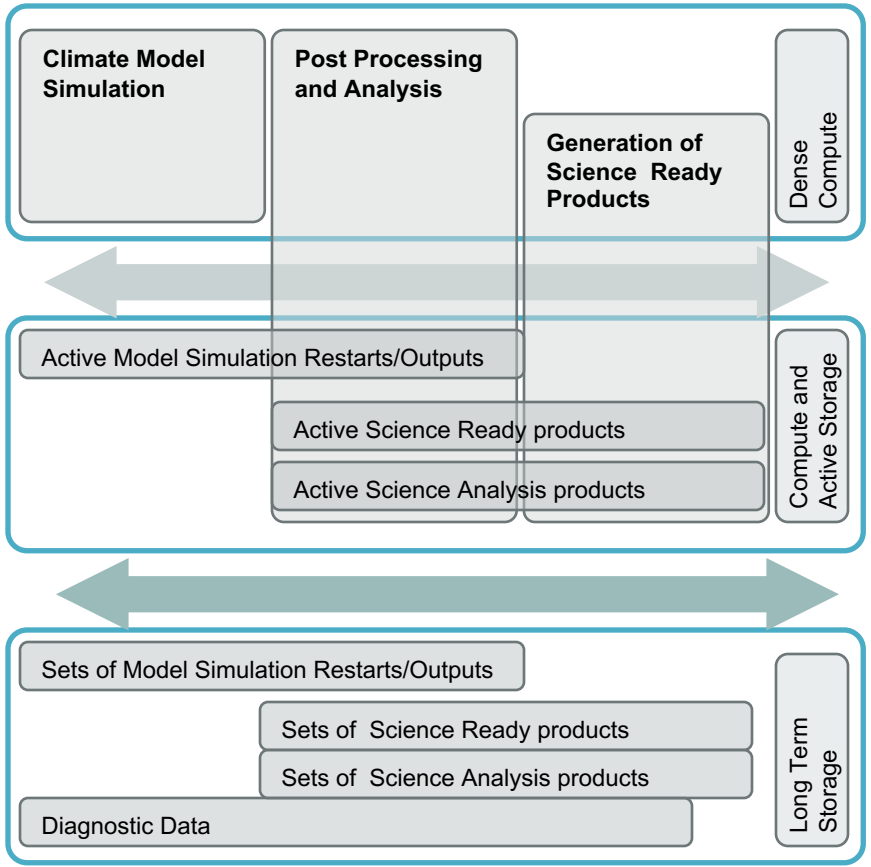
IBM Research

Climate Data:
•Inputs / Execution Snapshots
•Physics Parameterizations and variable forcing data
•Various Grid Mappings for different components of workflow
•Various Scientific Outputs
•Experimental Data Sets for Validation and Verification

Climate Model Simulation
•Climate Simulation over multiple years, compute intense

Post Processing and Analysis
•Analysis of climate runs
•Visualization
•UQ, Sensitivity Analysis

Science Ready Products
•Generation of various "downstream" products that predict different impacts of climate projections in different sectors

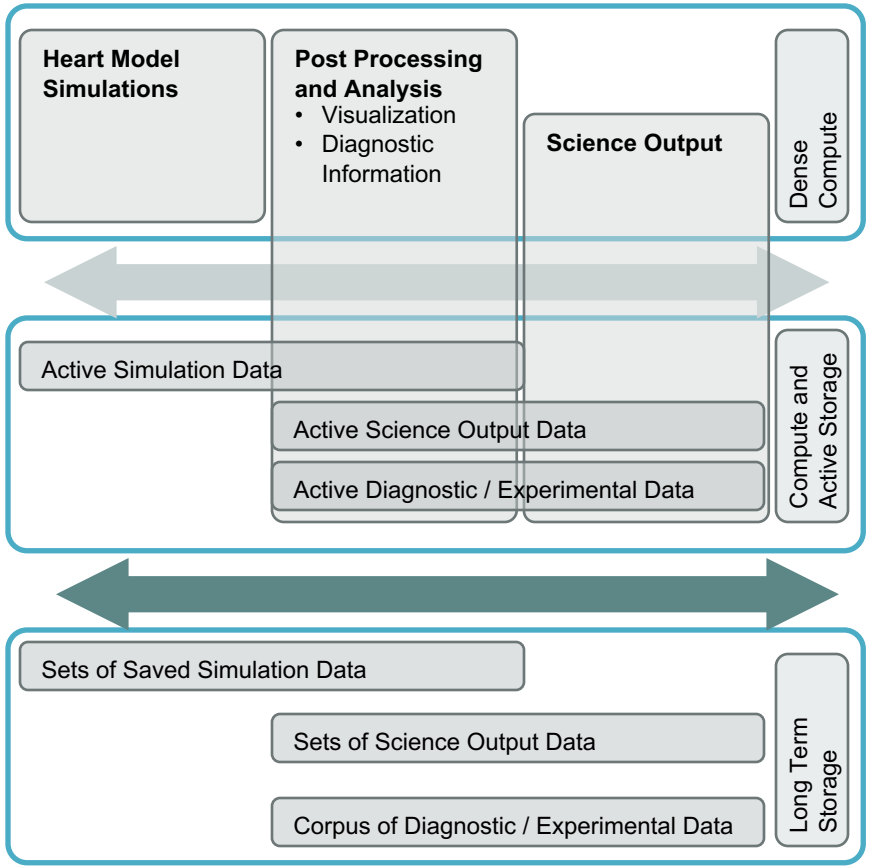


Full organ simulation Use Case

IBM Research

Experimental Data
•Scans of actual human hearts
•Diagnostic Information
Simulation Data
•Domain decomposed grids with structure, circulatory system, cell types for individual simulations
•Generated Simulation Histories UQ, Sensitivity Analysis
Scientific Output Data
•Predicted Diagnostic Information (ECG's …)

Heart Model Simulations:
•Generates computational simulations of heart
Post Processing:
•Visualization,
•Generation of ECG, other diagnostic data
Science Output:
•Generated "downstream" data used to apply simulation in given scenario (e.g. Drug Evaluation, Device Evaluation, etc)



Next Steps

- Workflows
  - Continue to explore unified data layer to accelerate data analytics across frameworks in a heterogeneous workflow
  - Demonstrate extreme-scale heterogeneous workflows on leading HPC systems

Thank you!

Publications/Presentations

- **[1] Towards memory-optimized data shuffling patterns for big data analytics.** *B Nicolae, C Costa, C Misale, K Katrinis, Y Park* - **Cluster, Cloud and Grid Computing (CCGrid)**, 2016
- **[2] Leveraging adaptive I/O to optimize collective data shuffling patterns for big data analytics.** *B Nicolae, CHA Costa, C Misale, K Katrinis, Y Park* - **IEEE Transactions on Parallel and Distributed Systems (TPDS)**, 2017
- [3] **SparkGA: A Spark Framework for Cost Effective, Fast and Accurate DNA Analysis at ScaleH Mushtaq.** *F Liu, C Costa, G Liu, P Hofstee, Z Al-Ars* - **8th ACM International Conference on Bioinformatics, Computational Biology,and Health Informatics (ACM-BCB),** 2017
- **[4] A Spark Frameowkr for < $100,< 1Hour, Accurate Personalized DNA Analysis at Scale – Spark Summit,** 2016