SAND2012-4071P

Scientific Data Analysis using MapReduce

MapReduce Workshop

May 16, 2012

Yung Ryn (Elisha) Choe, Ph.D.

Sandia National Laboratories, Livermore, CA

Joint work with

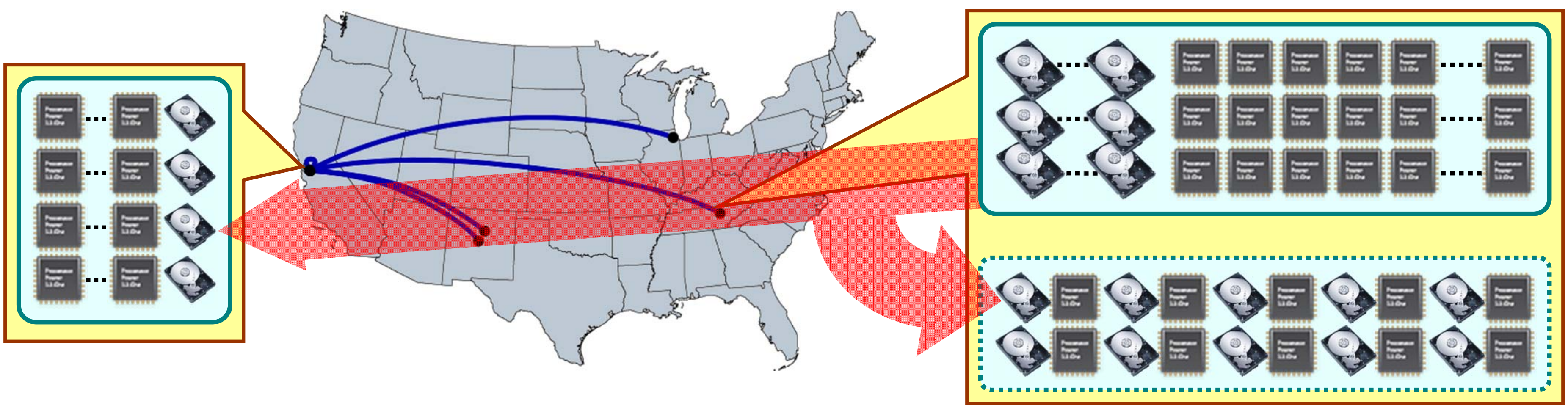
Craig Ulmer, Greg Bayer[1], Shyamali Mukherjee, Diana Roe, Janine Bennett

[1]Now at Pulse - Alphonso Labs

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Background

- Post-processing in exascale computing is a significant challenge
  - Massively-parallel scientific simulations: 10s to 100s of TBs of data
  - Post processing requires out-of-core data processing algorithms
- Data-Intensive Computing: Systems for simplifying large data work
  - Industry: Significant progress in parallel DBs and cloud computing
- Can we leverage these technologies to improve our workflows?

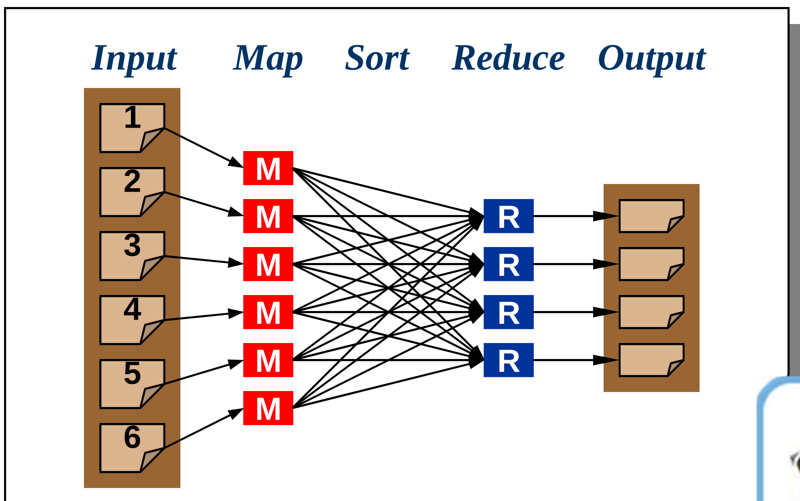


Sandia National Laboratories

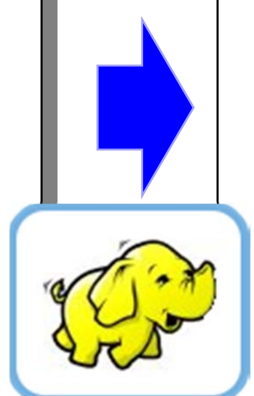
Data-Intensive Computing

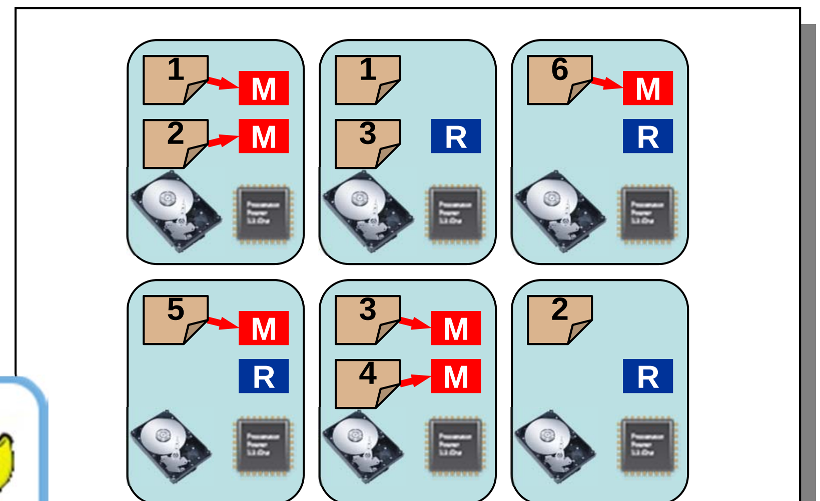
[1]

- Databases, Data Warehouses, Dataflow Engines, NoSQL
- Key idea: Make life easier by separating API from implementation
  - Developers: write algorithms, Architects: handle reliability/performance
- Yahoo's Hadoop: Combination of MapReduce and Framework



MapReduce Abstraction



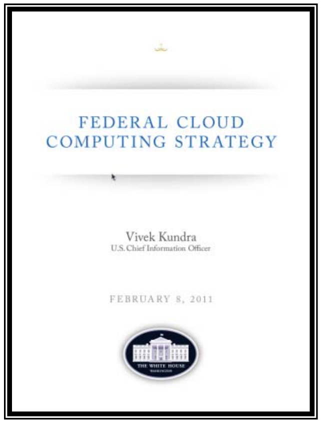


MapReduce Implementation

[1] Hardware Technologies for High-Performance Data-Intensive Computing, IEEE Computer, 2008.

Sandia's Research Direction

- Understand and leverage existing frameworks
  - Adapt SNL-relevant analysis codes to different platforms
  - Explore hardware/software/economic/security/political tradeoffs
  - Enhance with HPC technologies
  - *Do these platforms **make analysis easier**?*
- Funding: ASC, LDRD, CSRF
- Sandia's connectivity
  - LLNL, LBNL, MITRE, Wisconsin-Madison
  - Government agencies
  - Bay Area Hadoop Users + Companies
- Work with peers towards formation of *useful* government cloud
  - CIO + DOE Cloud Audit Reports

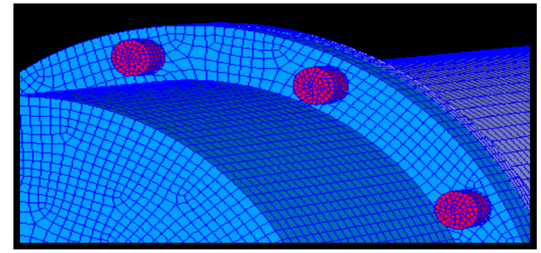


Sandia National Laboratories

Outline

- Scientific Computing users
  - Need to embed data analysis in storage systems
  - Two scientific data analysis algorithms using MapReduce
- Mesh Analysis on DWAs
  - Data Warehouse Appliances (DWAs) have been proven effective for data mining and informatics
  - Very few examples in scientific computing
  - MapReduce: Hadoop, Others: Netezza, XtremeData, LexisNexis
- Combustion Data Analysis
  - Turbulent Kinetic Energy (TKE)
  - Autocorrelation calculation

Scientific Simulation



- Advanced Simulation and Computing (ASC) - Sandia
- Simulating properties of complex systems
  - Mechanical, Thermal, Electrical
- Mesh-based analysis
  - Unstructured meshes
  - Non-uniform elements
- Massive datasets
  - Significant variable data

A 100M element simulation can generate many terabytes of data.

Tables		Rows/Bytes
Structural Data	Element	100 Million 3.2GB
	Vertex	100 – 800 Million 2.4GB – 19.2GB
Variable Data	Element	20 Billion 2.5TB
	Vertex	20 – 160 Billion 2.5TB – 20TB

- In-memory, single machine analysis infeasible
  - Need for out-of-core post-processing analysis tools
  - Example: ParaView

Sandia National Laboratories

Data Warehouse Appliances (DWAs)

- DW appliances
  - Mid-to-large volume data warehouse market
    - Terabyte to Petabyte range
  - Large number of parallel storage devices
  - Near-storage processing, via programming interface
- SQL-based

NETEZZA XtremeData, Inc. ORACLE Greenplum TERADATA

- Dataflow-based (other data-parallel languages)

hadoop LexisNexis

Sandia National Laboratories

An Early Evaluation

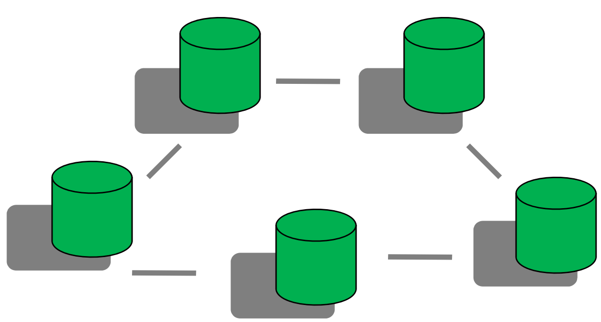
- Traditional approach of moving data to the user's analysis code is infeasible
  - Increased Dataset Sizes
  - Constant disk and network speeds
  - Capability computing consolidation
- Must provided processing within storage system
- DWA's have been successfully used to solve informatics problems
- Intent of this work:
  - Gain insight into tradeoffs involved in using DWAs to analyze scientific datasets

Evaluation Platforms

Hadoop MapReduce

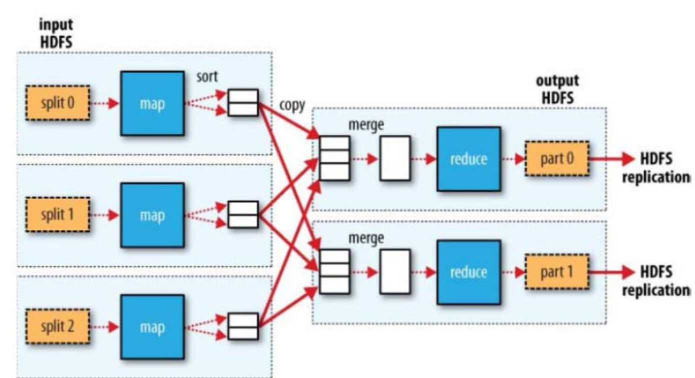
HDFS (Hadoop Distributed File System)

- High throughput
- Fault-tolerant
  - Replicated data
  - Automatic failover
- Complexity abstracted



Map/Reduce

- Automatic parallelization
  - Partitions Input Data
  - Schedules Execution
  - Handles Machine Failures
  - Manages Communication
  - No deadlocks, race conditions



Sandia National Laboratories

SNL Hadoop Cluster History

Decline Cluster
Purpose: Research
40 Nodes

Dual Core, Dual 40GB Disks
GigE + InfiniBand
January 2009

80 1TB SATA2 disks
Revamped installation
June 2009

Nebula Cluster
Purpose: Production
70 Nodes, 0.5PB

Intel I7 Quad-Core,
12GB Memory
4x 2TB Disks
GigE

July 2010

Mini Clusters
Recline: **Temporary replacement for Decline**
~20 small nodes from Reapp, GigE

Ion: **Low-power, cluster**
8 Atom/Ion (35W) nodes
GigE

Buzz Cluster
Purpose: Research
10 I/O Nodes
30 Diskless Nodes

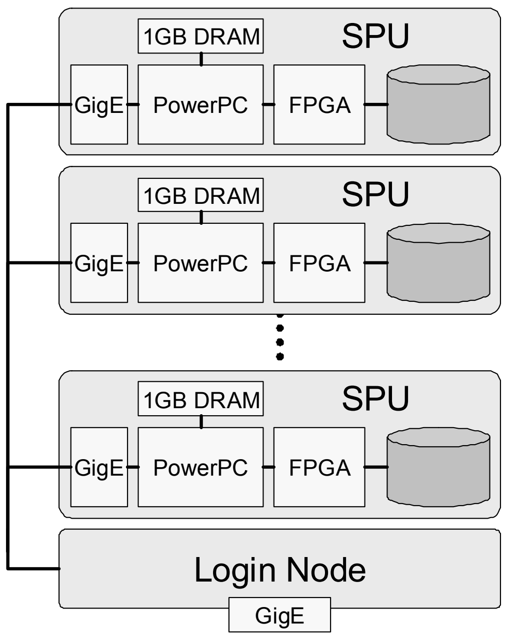
6x HDD, 2x SSD
GigE/10GigE/DDR IB
ATI 4890 GPU

December 2010

Netezza: Parallel Database

NETEZZA

The Netezza system utilizes multiple snippet-processing units (SPUs) to process data in parallel



- **Netezza Performance Server 10050**
- **Half-rack system**
- **54 active SPUs**
- **5 terabytes of database space**
- **Built-in PC with dual AMD Opteron processors functions as head node and access point for the database**
- **Users connect to the database remotely through ODBC connections**

Sandia National Laboratories

Data Warehouse Appliances (DWAs)

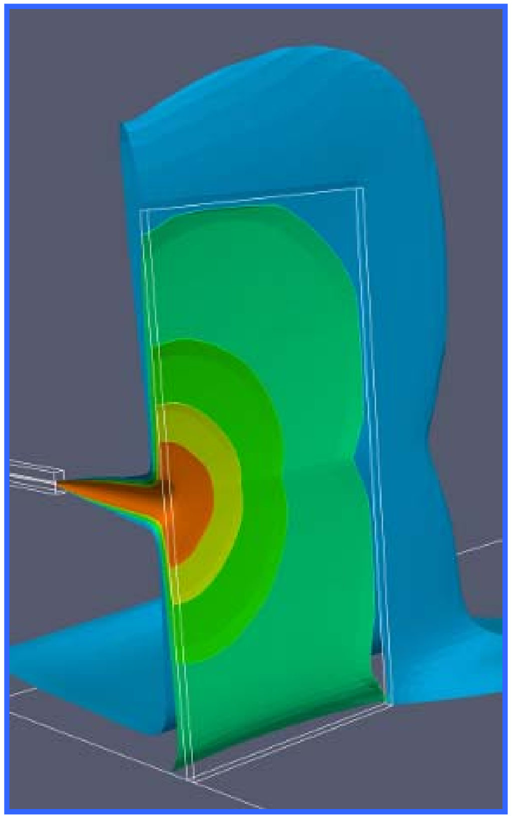
- SQL-based DWAs: Netezza and XtremeData
- Dataflow-base: LexisNexis Data Analytics Supercomputer (DAS)
- Hadoop MapReduce: Local cluster and Amazon EC2

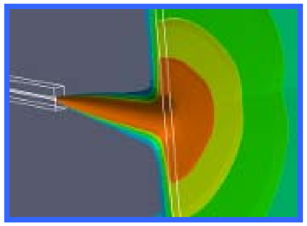
Platform	Compute Nodes	Cores/ Node	Memory/ Node	Disks/ Node	FPGAs/ Node
Netezza Mustang	54	1 PowerPC	1 GB	1	1
Netezza TwinFin6	6	8 x86	16 GB	8	2
XtremeData dbX 1008	8	6 x86	32 GB	12	1
XtremeData dbX 1016	16	6 x86	32 GB	12	1
LexisNexis DAS-20	10	4 x86	4 GB	2	0
LexisNexis DAS-60	32	4 x86	8 GB	1	0
Hadoop-Decline	32	2 x86	4 GB	2	0
Hadoop-Amazon-32	32	2 x86	1.7 GB	1	0
Hadoop-Amazon-128	128	2 x86	1.7 GB	1	0

Application Experiments

Threshold Volume

- Goal: Measure volume of gas exceeding threshold
- Mesh schema
  - Static element data stored separately from variable data
- Marching cubes style approach
  - Sums contribution of each element to total volume
- Very Parallel





Threshold Volume - Data

- Structural and Variable Data representation

Node ID	X	Y	Z

Node lookup table

Element ID	Node 1 ID, X,Y,Z	Node 2 ID, X,Y,Z	Node 3 ID, X,Y,Z	Node 4 ID, X,Y,Z	Node 5 ID, X,Y,Z	Node 6 ID, X,Y,Z	Node 7 ID, X,Y,Z	Node 8 ID, X,Y,Z

Extended element lookup table

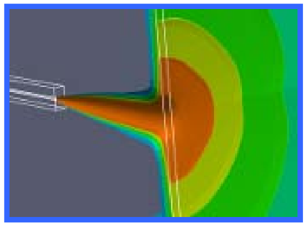
Timestep ID	Node ID	DISP X	DISP Y	DISP Z	NVAR1	NVAR2

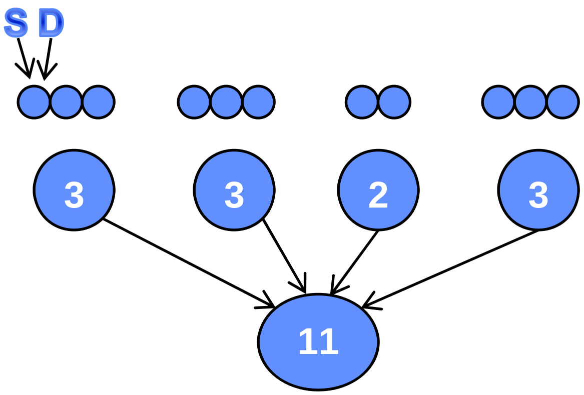
Node variable data table

Timestep ID	Element ID	EVAR1	EVAR2	EVAR3

Element variable data table

Threshold Volume - Implementation

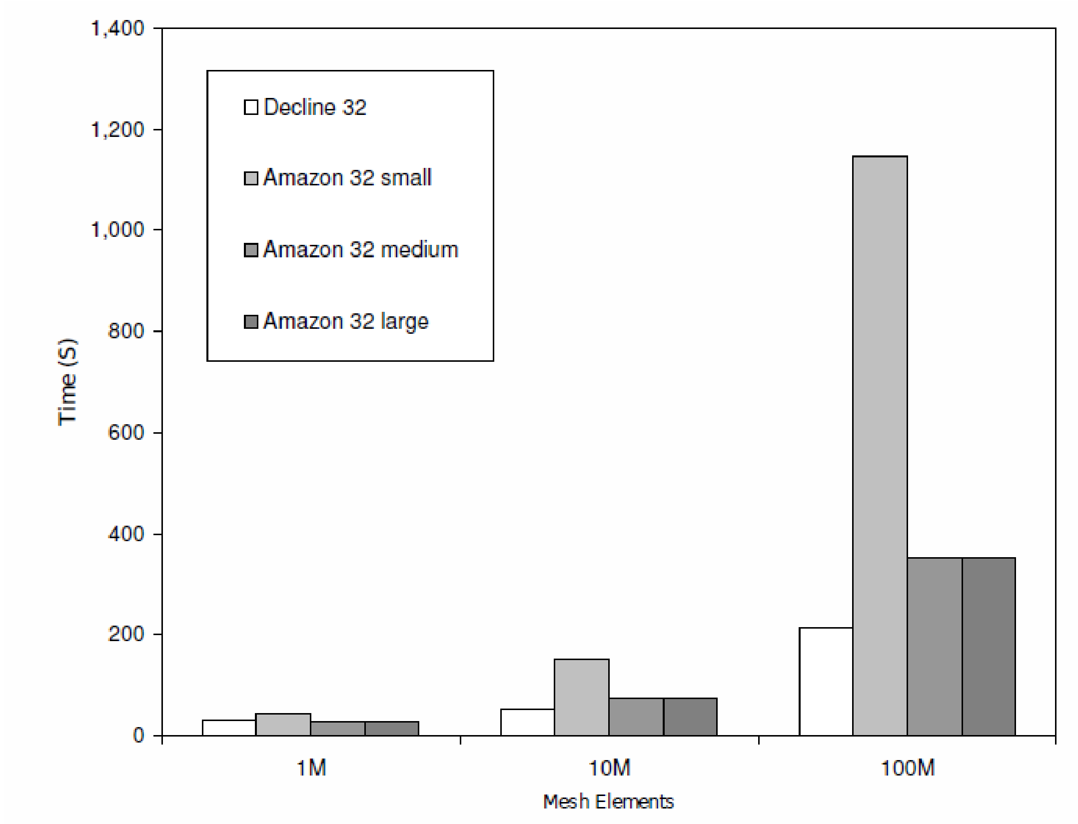




- Load and Stage static data with Join
- Volume Calculation
  - Join with dynamic data (8 joins per element)
  - Filter on node variable-based threshold
  - Calculate volume of a single element
  - Sum volume locally, then globally
- Netezza and XtremeData
  - 2^{nd} version of join: 4 joins for each tetrahedron x 6 (tetrahedra)
- LexisNexis
  - Enterprise Control Language (ECL)
  - Use built-in JOIN, ROLLUP, PROJECT and SUM operators

Threshold Volume

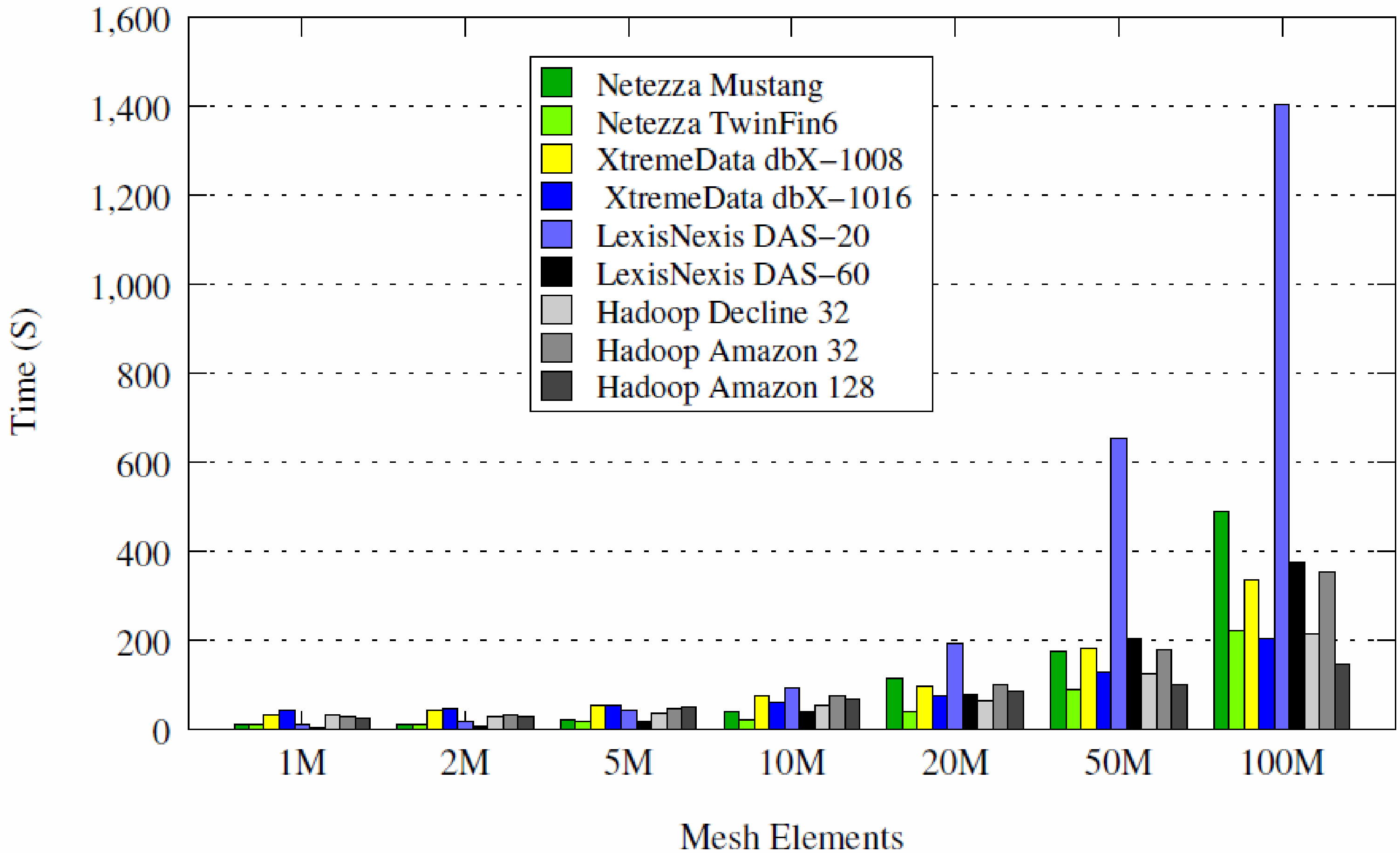
Performance: Hadoop Cluster



- **32 node hadoop clusters with different node types**

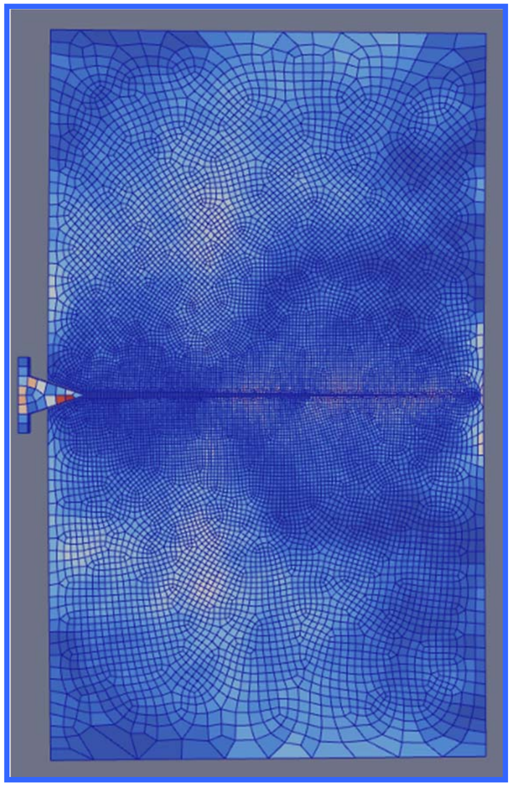
Threshold Volume

Performance: Data Warehouse Appliances



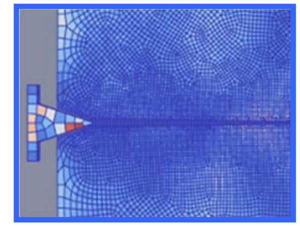
Element Pairing

- **Goal: Simulate realistic fractures**
- **Two separate meshes**
- **Central challenge: auto-generate list of element pairs pressed against each other closest**
- **Phase 1: Generate all faces of mesh then eliminate interior faces**
- **Phase 2: Find the closest faces**



- Phase 1: Extremely parallel
- Phase 2: $O(n^2)$ – All to all distance calculation
  - Chose not to use bounding-box filter (test extreme case)

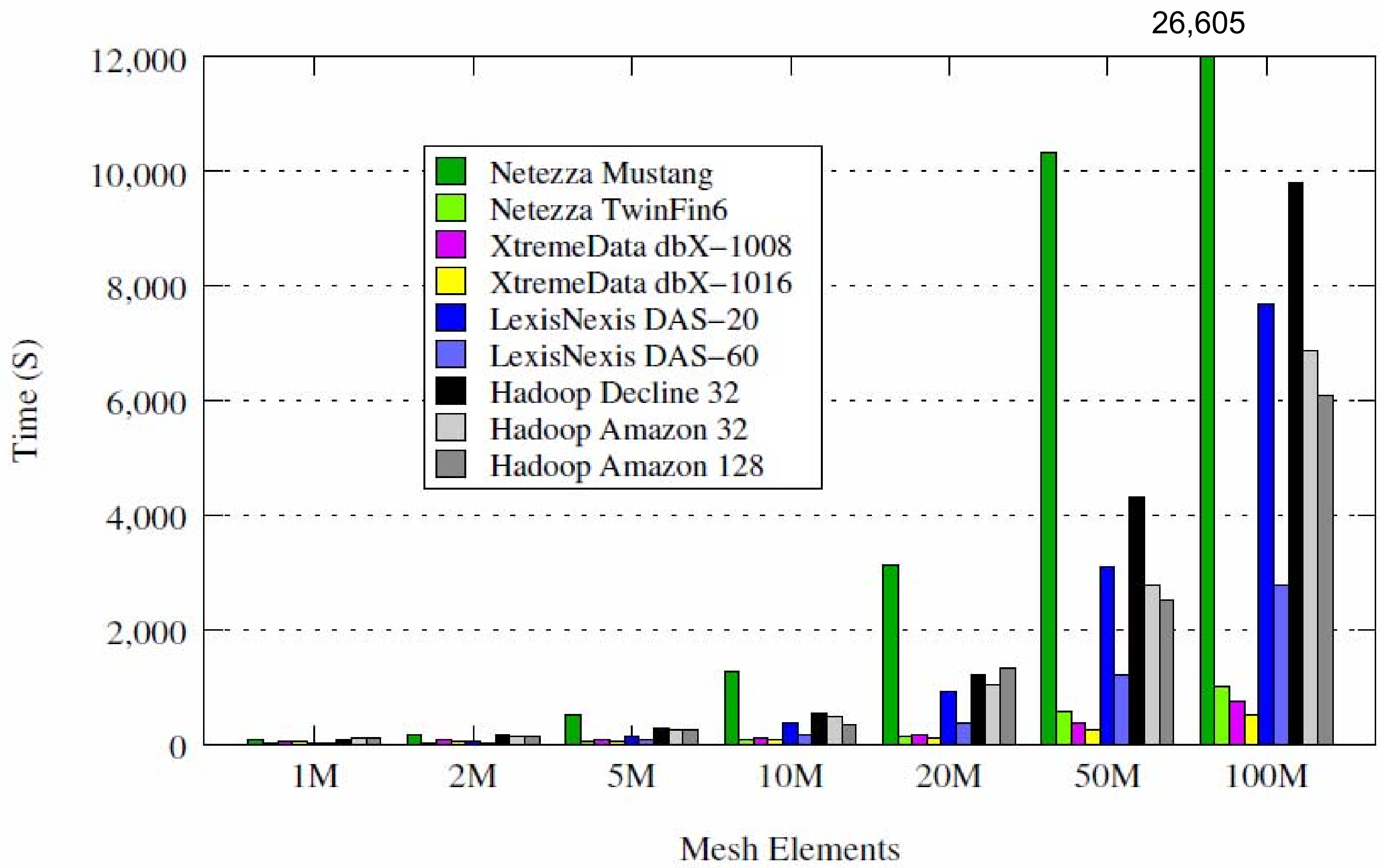
Element Pairing - Implementation



- Load and Stage static data
  - Join static data
- Phase 1: Filter for surface faces of each mesh
  - Join with dynamic data
  - Compare vertices of each hexahedron in sorted order
- Hadoop: Distribute surface faces of mesh 2 using Distributed Cache
- Phase 2: $O(n^2)$ – All to all distance calculation
  - SQL-based: “Group By”
  - Hadoop: Map-join, in memory streaming comparison, iterative approach
  - LexisNexis: PROJECT and DENORMALIZE operators
  - Chose not to use bounding-box filter (test extreme case)
- Select min distance pair for each element of mesh 1

Element Pairing

Performance: Data Warehouse Appliances

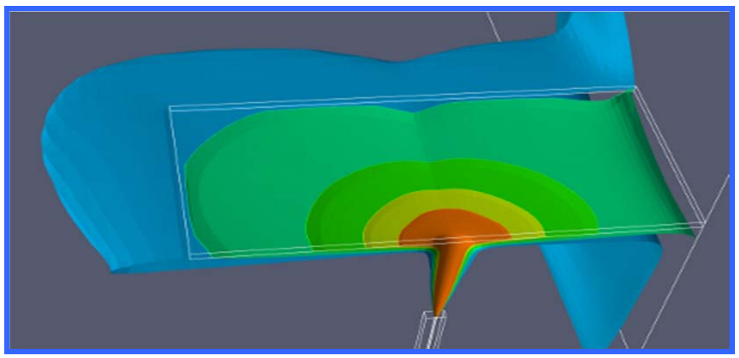


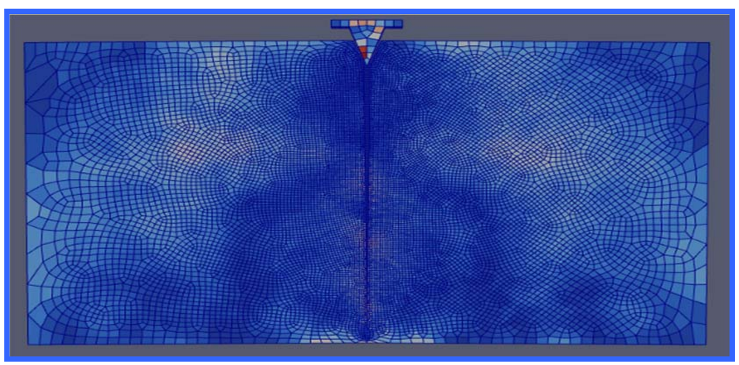
Lessons Learned

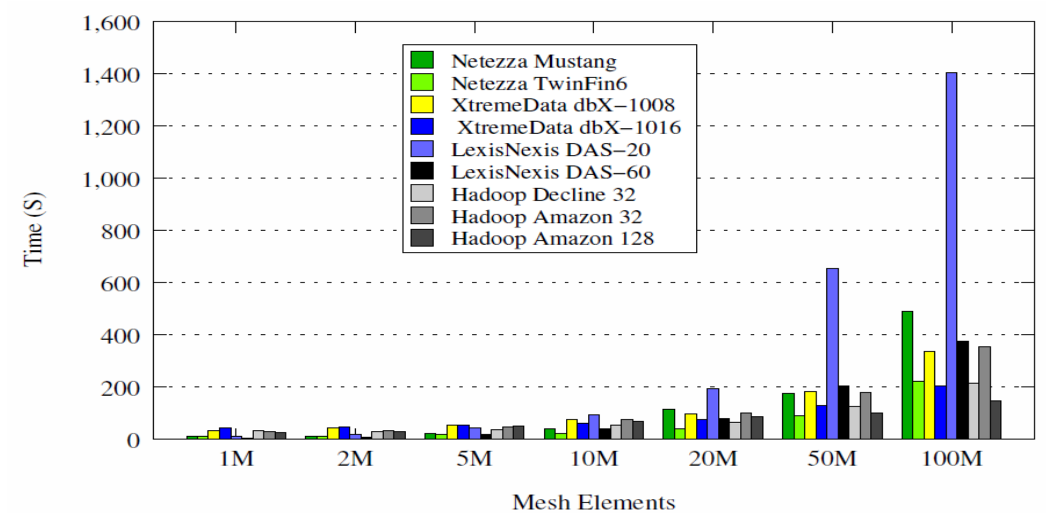
- Hadoop
  - Distributed Cache, Combiners, Binary Seek, explicitly moving calculation into memory helped
  - Highly tunable and Job startup cost is high
- Netezza
  - Proprietary commands: “Generate statistics” helped but less portable, UDF debugging difficult, Floating-point limitations (Mustang)
  - SQL portable but less control
- All Platforms
  - Performance optimization was non-trivial
  - Final phase (NxM calculation) was dominant

Mesh Analysis

- Ported **mesh analysis** algorithms to multiple platforms
  - Traditional SQL Parallel Database: Netezza, XtremeData
  - “NoSQL” Platforms: LexisNexis DAS, Hadoop (Local + Amazon)
  - *Unique Sandia Research*: Breadth study, 4 languages, 9 platforms







Point 1: Hadoop provides competitive choice
Point 2: Algorithms may be difficult to express
Point 3: Refactoring required for performance

[1] Scientific Data Analysis on Data-Parallel Platforms, SAND 2010
[2] Exploring Data Warehouse Appliances for Mesh Analysis Applications, Digital Media at Scale 2010

Sandia National Laboratories

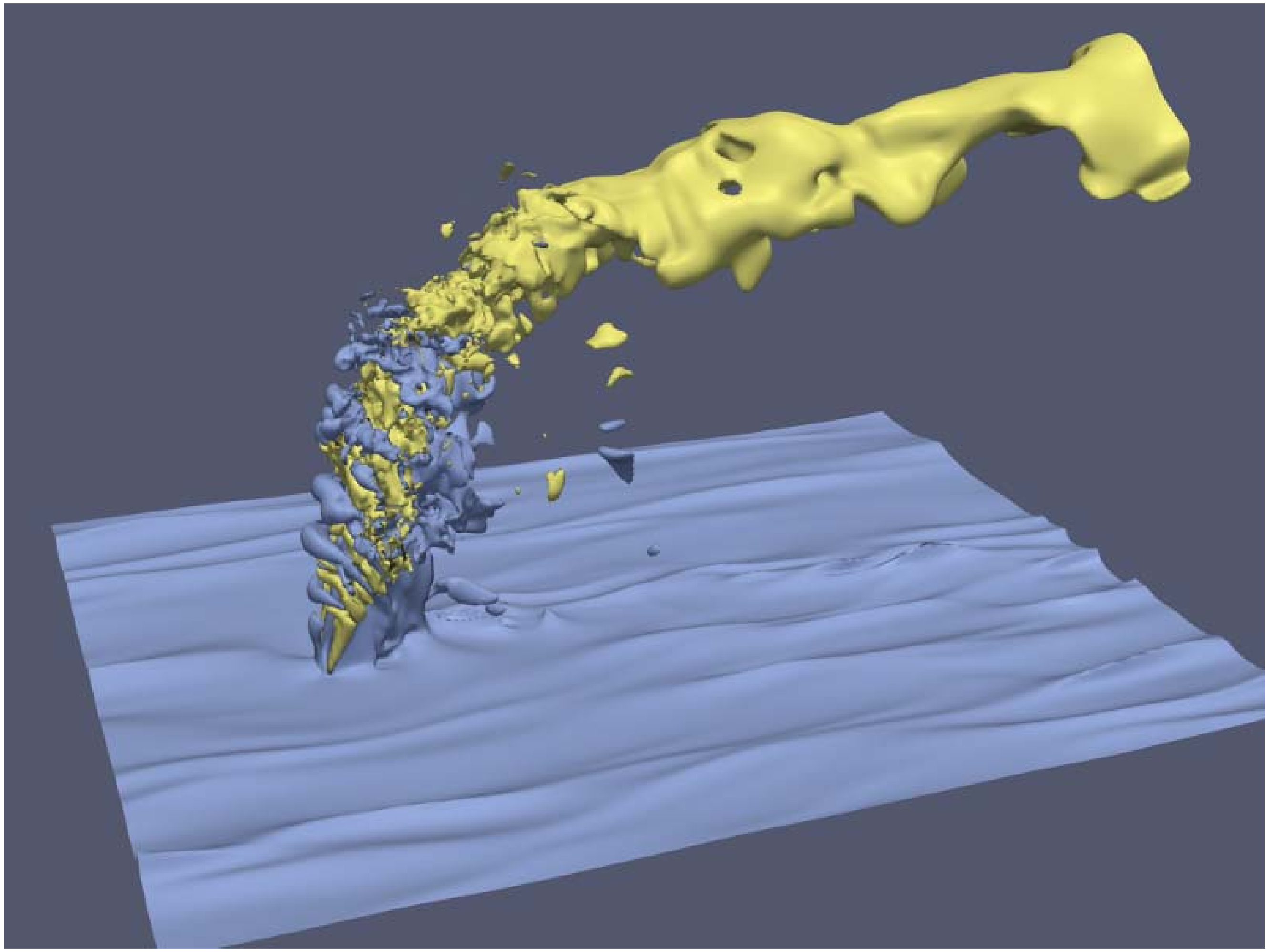
Nebula Cluster

- New Hadoop Cluster - $150K Multi-Customer
  - 2 Racks / 70 1U nodes: 280 cores + 280 disk drives
  - 560 TB Total Raw Disk Capacity (~0.5 PB)
  - 876 GB Total RAM
- Motivation for Hadoop / Nebula
  - Fault Tolerant, TB-Scale Analysis on Commodity Hardware
  - Best Suited to Algorithms that are IO-Limited (instead of CPU-Limited or Communication- Limited)
- Performance Comparison – Hadoop Terasort Benchmark

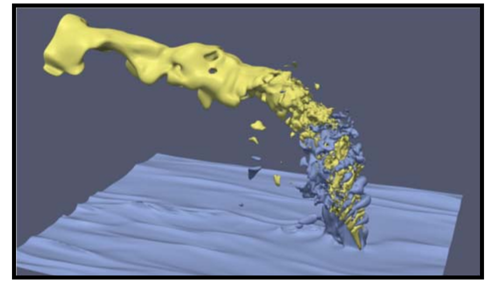
Cluster	Test Date	Nodes	Cores	Data Size	Time	Normalized to Yahoo Cluster*
Recline	July 2010	18	18	100GB	3807 sec	0.28x
Nebula	**July 2010**	**65**	**260**	**100GB**	**346 sec**	**0.84x – 1.68x**
Nebula	**July 2010**	**65**	**260**	**1TB**	**3390 sec**	**0.87x – 1.75x**
Yahoo	May 2008	910	7280	1TB	208 sec	1x

* Performance normalized by # nodes – # cores. 100GB runs compared via runtime x 10.

Combustion Data Analysis



Combustion Simulation Data



- Jet in Cross Flow Vector Data
  - JICF square-jet simulation by R. Grout, A. Gruber, and J.H. Chen
  - S3D code run on Jaguar and produced 100+TB datasets at ORNL
  - CRF researchers use in situ analysis to obtain early results
  - Collaborators provided access to data in cloud resource
- Data File Format
  - Time-varying 3-dimensional grid of vectors
  - Three files for each time step, $<u, v, w>$ coordinates of velocity vector at each point in domain
  - Binary file with x varying the fastest then y then z
  - The dimension of the data: 1408 by 1080 by 1100 (21 time steps)
  - Total 400 GB of data
  - Dataset generators were also used for scaling experiments

Turbulent Kinetic Energy (TKE)

Given a set of $i = 1, \ldots, n$ timesteps, for any given point p in the domain the *turbulent kinetic energy* at that point is defined as:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\left(u_i\left(p\right)-\overline{u\left(p\right)}\right)^2+\left(v_i\left(p\right)-\overline{v\left(p\right)}\right)^2+\left(w_i\left(p\right)-\overline{w\left(p\right)}\right)^2\right) \tag{1}$$

Through rearrangement of terms, equation 1 can be re-written as follows:

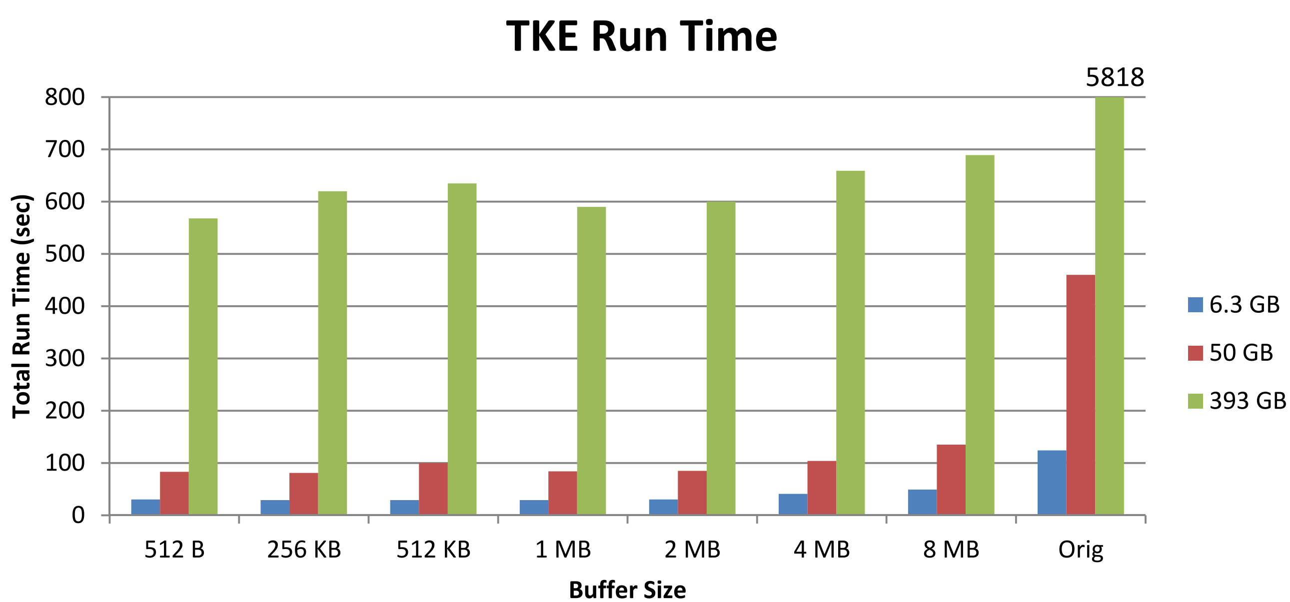
$$\frac{1}{n}\sum_{i=1}^{n}\left(\left(u_i\left(p\right)-\overline{u\left(p\right)}\right)^2+\left(v_i\left(p\right)-\overline{v\left(p\right)}\right)^2+\left(w_i\left(p\right)-\overline{w\left(p\right)}\right)^2\right)$$

$$\frac{1}{n}\sum_{i=1}^{n}\left(u_i\left(p\right)-\overline{u\left(p\right)}\right)^2+\frac{1}{n}\sum_{i=1}^{n}\left(v_i\left(p\right)-\overline{v\left(p\right)}\right)^2+\frac{1}{n}\sum_{i=1}^{n}\left(w_i\left(p\right)-\overline{w\left(p\right)}\right)^2$$

$$\text{Variance}\left(u\left(p\right)\right)+\text{Variance}\left(v\left(p\right)\right)+\text{Variance}\left(w\left(p\right)\right)$$

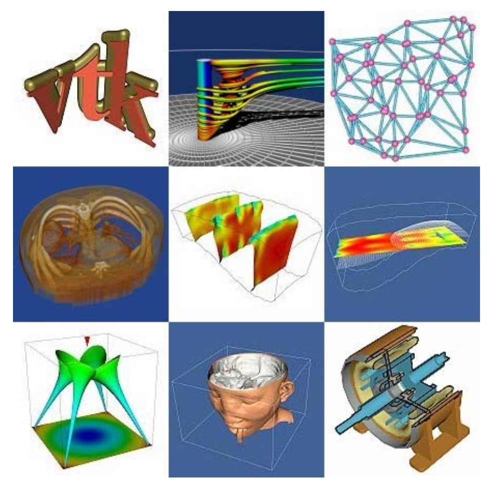
Turbulent Kinetic Energy (TKE)

- Developed MapReduce implementation
  - First Implementation (Orig): Groups data by point (sort)
  - Calculate total TKE for each point in parallel
  - Second (optimized) Implementation: Groups by buffer



Basic Descriptive Statistics

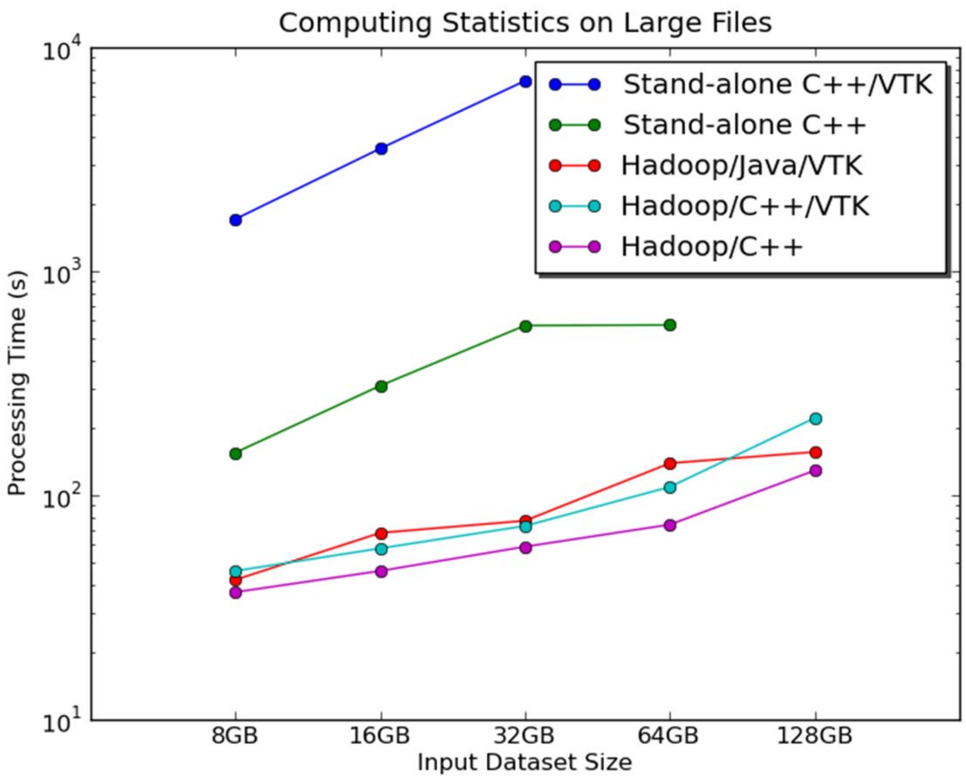
- Integrated Hadoop with VTK statistics package
- VTK: Visualization Toolkit
- VTK Descriptive Stats Library calculates:
  - Learn: cardinality, min, max, mean, centered M_2, M_3, M_4
  - Derive: variance, standard deviation, skewness, kurtosis, sum
- Performance comparison study
  - C++ implementation of basic statistics
  - VTK C++ Library
  - Hadoop Java integration with VTK
  - Hadoop Pipes C++ Implementation
  - Hadoop Pipes C++ using VTK
- Compatible with any Hadoop-based cloud



Sandia National Laboratories

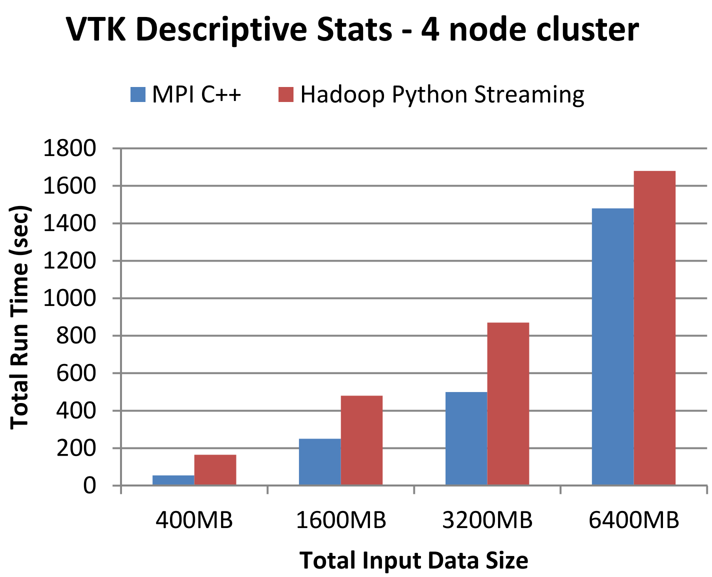
Basic Descriptive Statistics

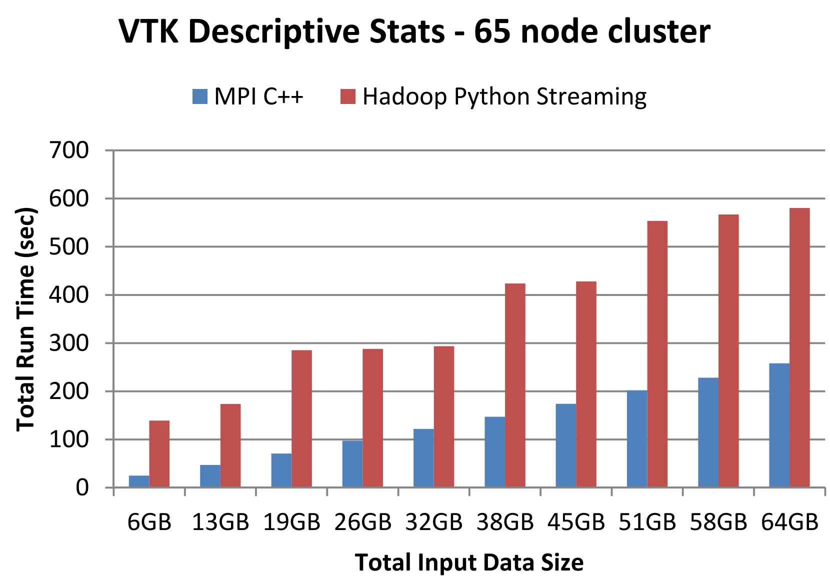
- Using Hadoop Pipes C++ support shown efficient



Basic Descriptive Statistics

- Using Hadoop Streaming in Python and VTK
  - Using binary (typedbytes) input file format
  - Small cluster with MPI/NFS and Baseline (local disk) comparisons
  - Distributed Files System (e.g. GlusterFS or Ceph) comparison on going





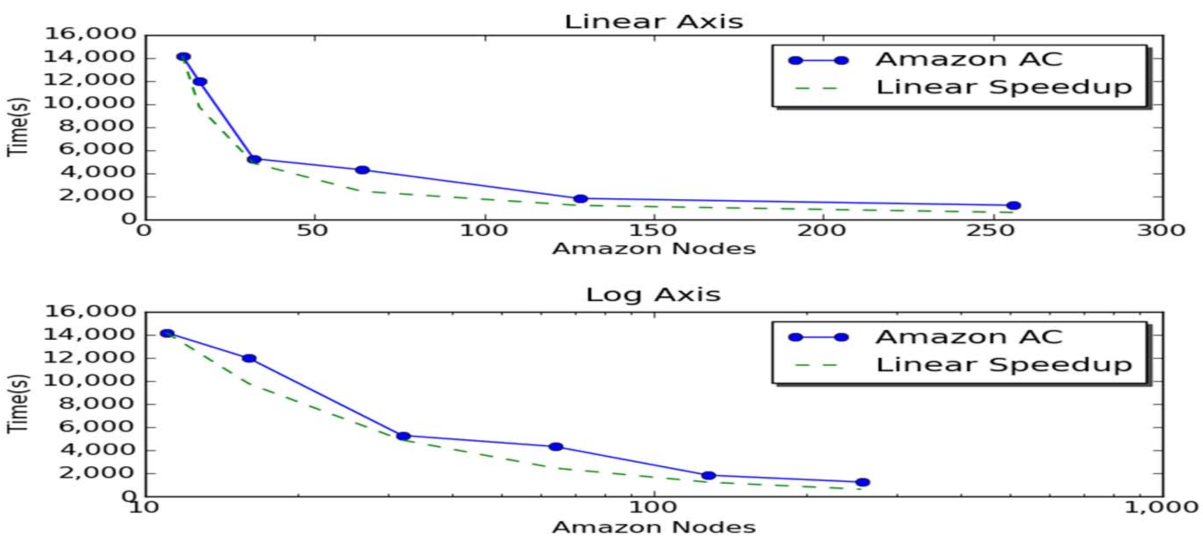
Autocorrelation

Autocorrelation functions show how correlated the velocity is in different regions in the domain. The following is a series of fields that are of interest to combustion scientists:

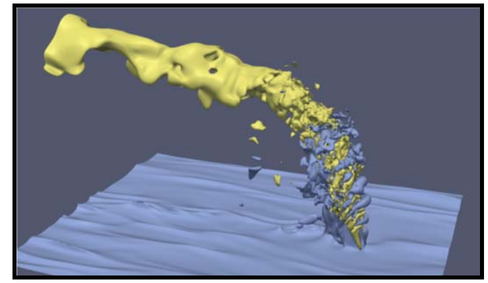
for all timesteps i **do**
  for all points j in the domain **do**
    for all points k in x (with same y & z coordinates) **do**
      compute $f_{ux}(j,k) += \frac{u_i(x_j)u_i(x_k)}{\overline{u(x_j)^2}}$
      compute $f_{vx}(j,k) += \frac{v_i(x_j)v_i(x_k)}{\overline{v(x_j)^2}}$
      compute $f_{wx}(j,k) += \frac{w_i(x_j)w_i(x_k)}{\overline{w(x_j)^2}}$
      compute $r_x(j,k) = |x_j - x_k|$
    end for
    for all points k in y (with same x & z coordinates) **do**
      compute $f_{uy}(j,k) += \frac{u_i(y_j)u_i(y_k)}{\overline{u(y_j)^2}}$
      compute $f_{vy}(j,k) += \frac{v_i(y_j)v_i(y_k)}{\overline{v(y_j)^2}}$
      compute $f_{wy}(j,k) += \frac{w_i(y_j)w_i(y_k)}{\overline{w(y_j)^2}}$
      compute $r_y(j,k) = |y_j - y_k|$
    end for
    for all points k in z (with same x & y coordinates) **do**
      compute $f_{uz}(j,k) += \frac{u_i(z_j)u_i(z_k)}{\overline{u(z_j)^2}}$
      compute $f_{vz}(j,k) += \frac{v_i(z_j)v_i(z_k)}{\overline{v(z_j)^2}}$
      compute $f_{wz}(j,k) += \frac{w_i(z_j)w_i(z_k)}{\overline{w(z_j)^2}}$
      compute $r_z(j,k) = |z_j - z_k|$
    end for
  end for
end for

Autocorrelation

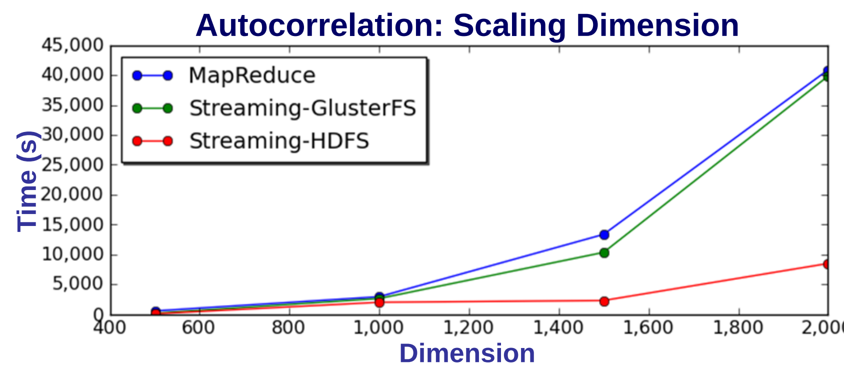
- Developed MapReduce implementation
  - Group data by each row in the domain
    - Read data once, expand 3x
  - For each row, calculate average at each “correlation distance” in parallel
  - Average autocorrelation over all rows, by distance
- Scalability:

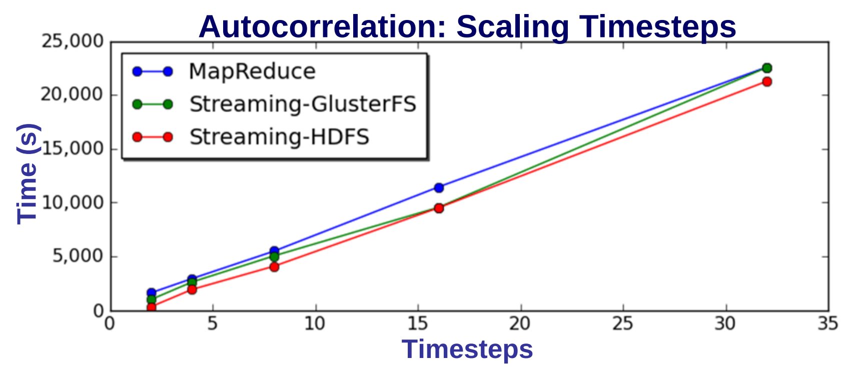


Adapting Hadoop



- How can we make Hadoop more accessible to scientific community?
  - Hadoop Streaming: utilize other codes/libraries
- MapReduce Implementations
  - Java
  - Scalable
- Custom Hadoop Streaming
  - C/OpenMP
  - GlusterFS and HDFS Storage



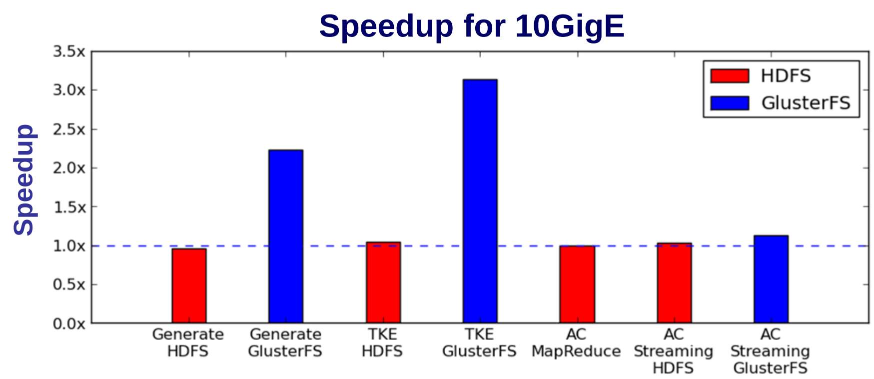


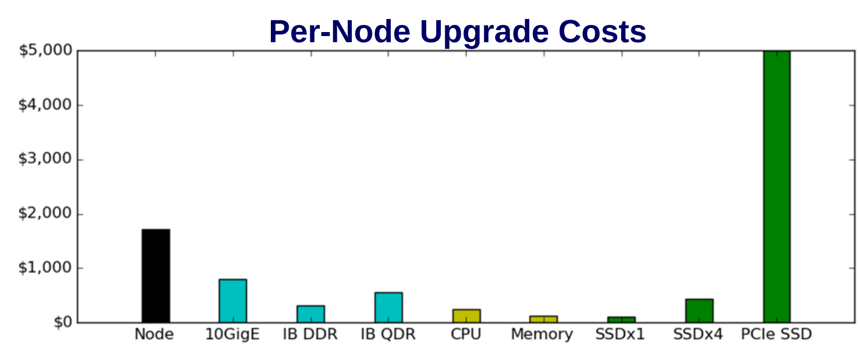
Point 1: MapReduce easier/quicker to code
Point 2: Streaming simplified legacy scale up
Point 3: Attractive for parameter studies

Sandia National Laboratories

Hadoop: Cluster Tradeoffs

- How should we build clouds for data-intensive work?
  - Typically constrained by Cost, Power, and Size
- Hardware experiments
- Which components to upgrade?
  - Interconnect: no
  - SSD: maybe
  - CPU/Memory: yes
  - Node count: yes





Point 1: Hadoop does not leverage interconnect
Point 2: Other DFS do get boost for some tasks
Point 3: Scale out, then up

Platform Comparison

Platform	API	Storage	Target
Netezza	SQL + UDF	Proprietary	Real Time Analytics
XtremeDB			
IBM InfoSphere Streams	Spade	Sources	Streaming Analytics
LexisNexis DAS	ECL	Proprietary	Complex Dataflows
Hadoop	MapReduce	HDFS	Batch Processing
MapR Tech	MapReduce	Proprietary	Enterprise Hadoop
SectorSphere	Sphere/UDF	Sector	Batch Processing
Cassandra	Column DB (Thrift)	Key/Value	Write Throughput, Eventual consistency
MongoDB	M-QL	Sharding	Doc-oriented DB
Membase	MemcacheD	Key/Value	Persistent KV

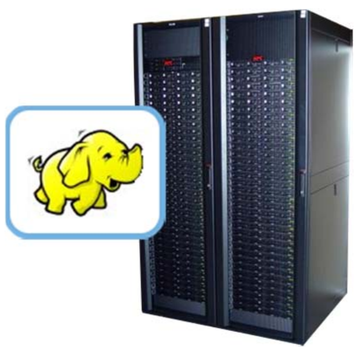
Sandia National Laboratories

Distributed File Systems

- Renewed interest in parallel or distributed file system research
  - Kernel Space: Lustre, PVFS, GPFS, Ceph, MapR Tech
  - User Space/Overlay: HDFS, GlusterFS, Ceph, Sector
  - Key-Value/Objects: Cassandra, Membase
- Data-intensive applications benefit from DFS capabilities
  - Locality: Enables scheduler to collocate computation with data
  - Caching: Helps scalability and load balancing (Ceph,HDFS)
  - Block Distribution: Helps load balance system
  - Replication: Improves reliability and decreases hotspots
- How do we leverage in Hadoop?
  - Native: Shim layers provide IO and locality
  - Streaming: Do-it-yourself locality

Concluding Remarks

- Main points
  - Data-intensive computing platforms can do meaningful scientific work
  - Best targets today: post processing or low-communication capacity jobs
  - Benefit: Simplifies out-of-core development, improves reliability
  - Need for improvement in maximizing hardware components
  - Hadoop MapReduce framework is a good option
- Impact: Stimulated interest by a number of large-data users
  - Internal: Combustion, Satellite, Network Security, Radiation Portals
  - External: Requests for unbiased FFRDC views
- Ongoing work
  - Integrating data-intensive frameworks into Cray
  - Security in MapReduce frameworks and cloud facilities
  - Collaboration with peers towards government cloud



Sandia National Laboratories

Relevant Publications

T. Plantenga, Y. Choe, A. Yoshimura, “Using Performance Measurements to Improve MapReduce Algorithms”, ICCS 2012, Tools for Program Development and Analysis in Computational Science Workshop, Omaha, NE, June 2012 (accepted).

C. Chen, Y. Choe, C. Chuah, and P. Mohapatra, “Experimental Evaluation of the Impact of Packet Capturing Tools for Web Services”, IEEE Global Communications Conference, Exhibition & Industry Forum (GLOBECOM), Houston, TX, December, 2011.

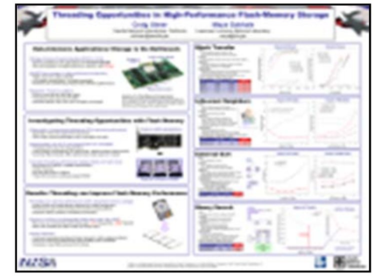
C. Ulmer, G. Bayer, Y. Choe, and D. Roe, “Scientific Data Analysis on Data-Parallel Platforms,” Sandia Technical Report SAND2010-7471, September 2010.

C. Ulmer, G. Bayer, Y. Choe, and D. Roe, “Exploring Data Warehouse Appliances for Mesh Analysis Applications,” in HICSS-43 Digital Media at Scale, January 2010.

M. Gokhale, J. Cohen, A. Yoo, M. Stokes, A. Jacob, C. Ulmer, and R. Pearce, “Hardware Technologies for High-Performance Data-Intensive Computing”, IEEE Computer, Vol. 41 No. 4, April 2008.

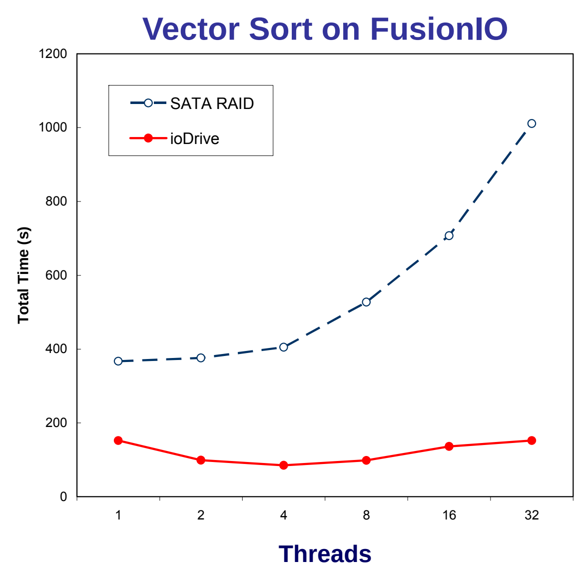
Additional Slides

Can SSDs Help?



- Solid-state Storage Devices (SSDs) vs Hard disk drives (HDDs)
  - 10-100x Latency Improvement (250 μs – 25 μs)
  - 2x-10x Bandwidth Improvement (200 MB/s – 1.2 GB/s)
  - 32x more expensive per GB ($1.9/GB)
  - 12x less capacity per SATA port

- SSDs have performance oddities
  - Switch to highly-threaded IO for latency benefits
  - Slowdowns may occur on dirty devices
- Should we expect a boost for Hadoop?
  - Hadoop geared towards streaming operations
    - Easy/Cheap to scale HDD bandwidth in RAIDs
  - Intermediate values, sorting may benefit though
    - See LBNL work in Magellan w/ Virident



Sandia National Laboratories