Scientific Data Management at Exascale

John Wu
Lawrence Berkeley National Laboratory

Scientific Data Management at Exascale

John Wu

Outline

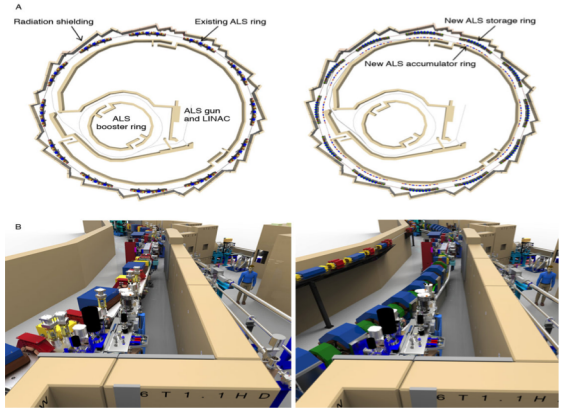
- ❑ Introduction
- ❑ Examples of Scientific Data Management
- ❑ Deep Dive: FastBit Indexing
- ❑ Deep Dive: FasTensor Automatic Parallelization Engine
- ❑ Summary

Mountains of Scientific Data Wait for Analysis

Light Source

180 PB/year

(ALS-U at Berkeley Lab)

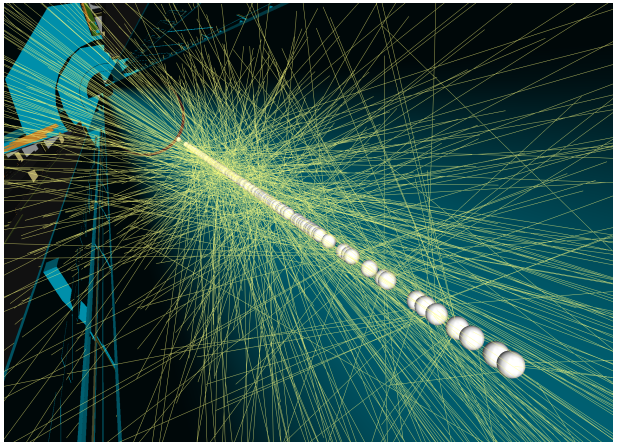


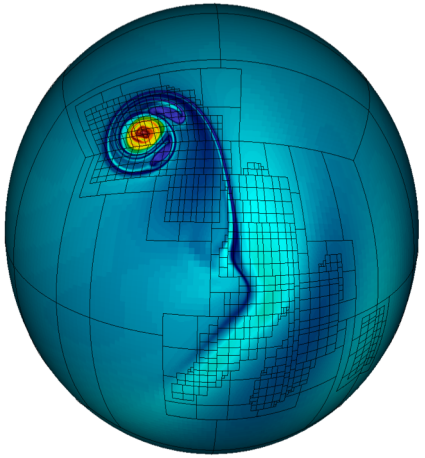
Genomics

10 PB/year

High Energy Physics

200 PB/year





Climate

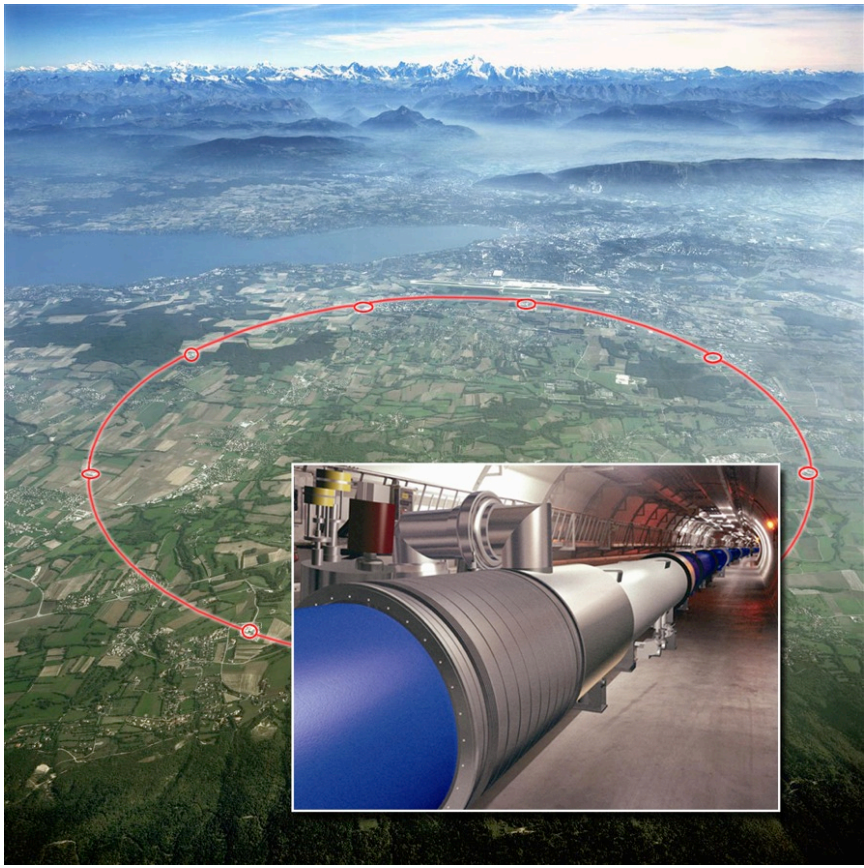
100 EB/year

Data and Picture source: L. Nowell, D., Ushizima, JGI and ALS at LBNL, etc.

Example of Big Data in Science

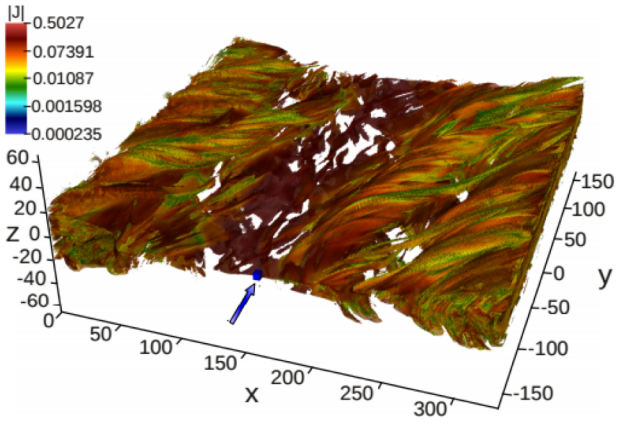
Large Hadron Collider: to find the god particle

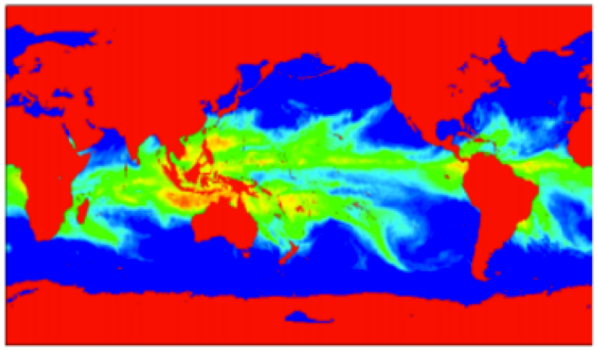
- **15 PB** per year kept – sensors capable of 140PB/s
- 27 km tunnel
- ~10,000 superconducting magnets
- Operating temperature 1.9 Kevin
- Construction cost:
  - US$9Billion
- Power consumption: ~120 MW

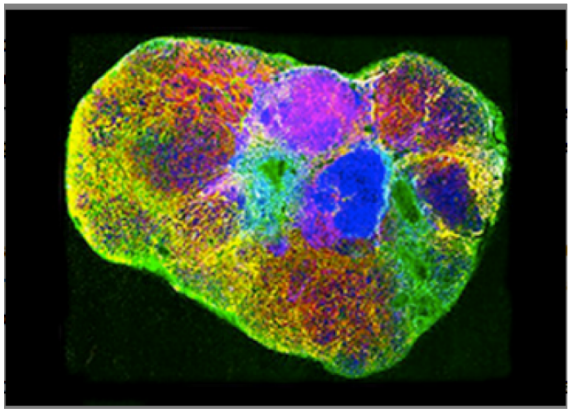


Example: Small Amount of Data Often Holds Key

- Physics: 3D magnetic field reconnection
  - Energy > 10 && 157.654 < x < 1652.441 && −165 < y < −160.025 && −2.5607 < z < 2.5607
- Climate: atmospheric river
  - spatial constraint (north America) and attribute constraints (Integrated Water Vapor > 2cm))
- Life sciences: Mass Spectrometry Imaging
  - spectra selection (x_{min}:x_{max}, y_{min}:y_{max}, m/z)







How to locate and retrieve these records?

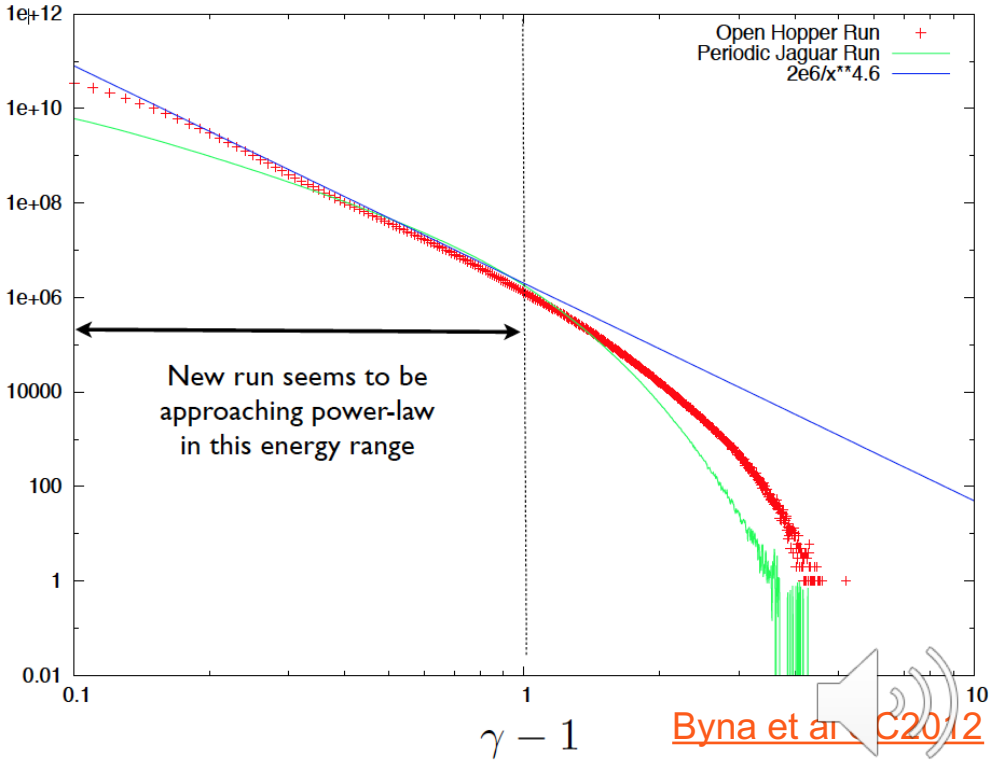
Example: Feature Extraction from Large Data

Magnetic reconnection

- ❑ Applications: magnetic confinement fusion, solar wind
- ❑ Data from simulation of trillions of ions and electrons
  - Example: space weather simulation on 120,000 hopper cores @NERSC
    - 20,000 MPI tasks * 6 OpenMP threads
  - ~35TB per timestep
  - Total ~350TB
  - Example science result
    - Histogram of particle energy distribution appear following the power law

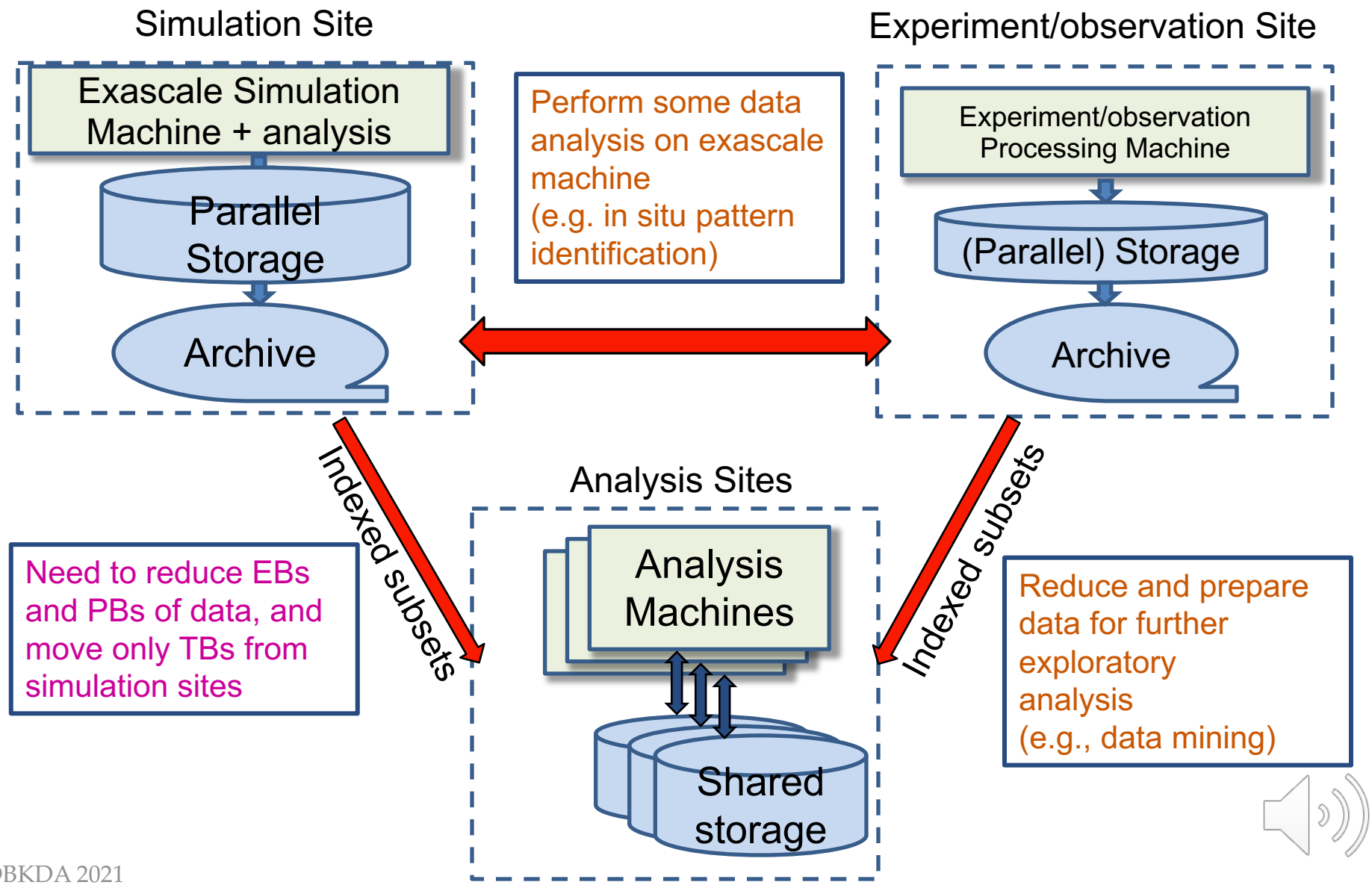
Challenge

- How to quickly and easily compute the power spectrum from 350TB of raw data?



Byna et al., SC2012

Example: Online Comparative Analytics



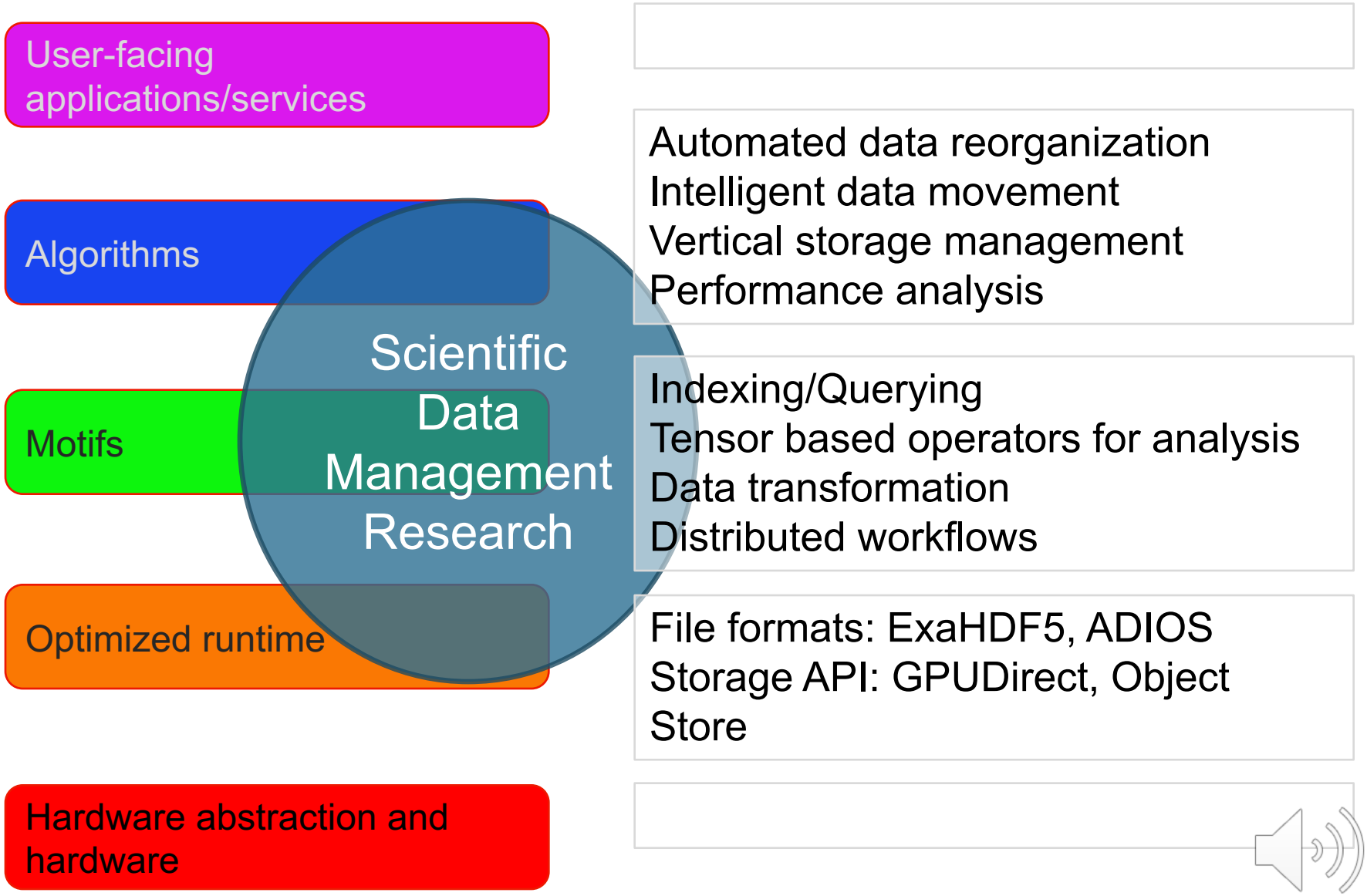
Scientific Data Management at Exascale

John Wu

Outline

- ❑ Introduction
- ❑ Examples of Scientific Data Management
- ❑ Deep Dive: FastBit Indexing
- ❑ Deep Dive: FasTensor Automatic Parallelization Engine
- ❑ Summary

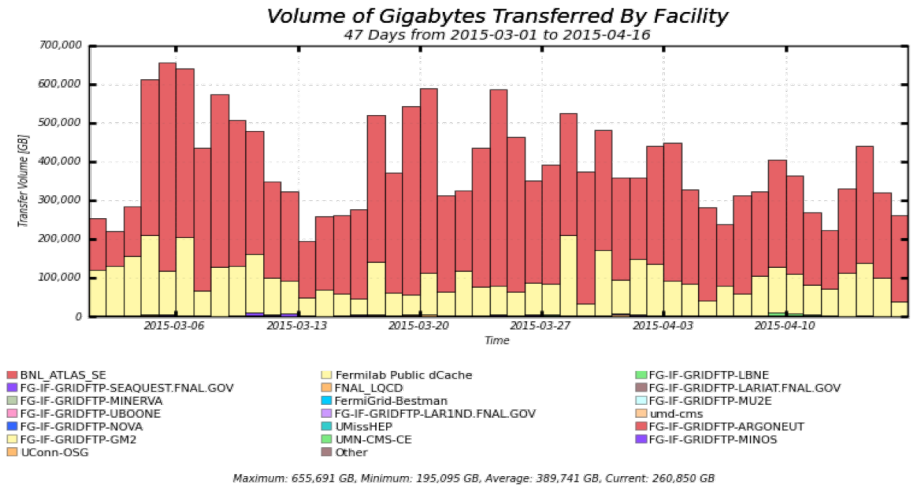
Scientific Data Management in Computer Science



Berkeley Storage Manager (BeStMan) -- Unified API for Many Storage Systems

What BeStMan does

- Unified API for accessing many storage systems
- Supports multiple transfer protocols and load balancing for multiple transfer servers
- Implements Storage Resource Management (SRM) interface v2.2, and compatible and interoperable with other 4 SRM implementations in WLCG



Daily data transfer volume in OSG from 3/1/2015 to 4/15/2015. BeStMan is used to transfer 100s TB/day in OSG.

Accomplishments

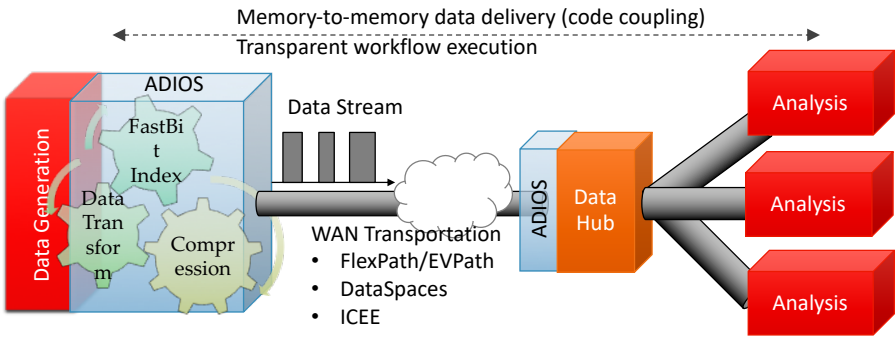
- Open source under BSD license, distributed with OSG software
- Scalable performance on many file systems and storages, such as Xrootd and Hadoop
- Organized an international standard through OFG - GFD.129, 2008
- US Patent 8,705,342 B2, 2014. Co-scheduling of network resource provisioning and host-to-host bandwidth reservation on high-performance network and storage systems

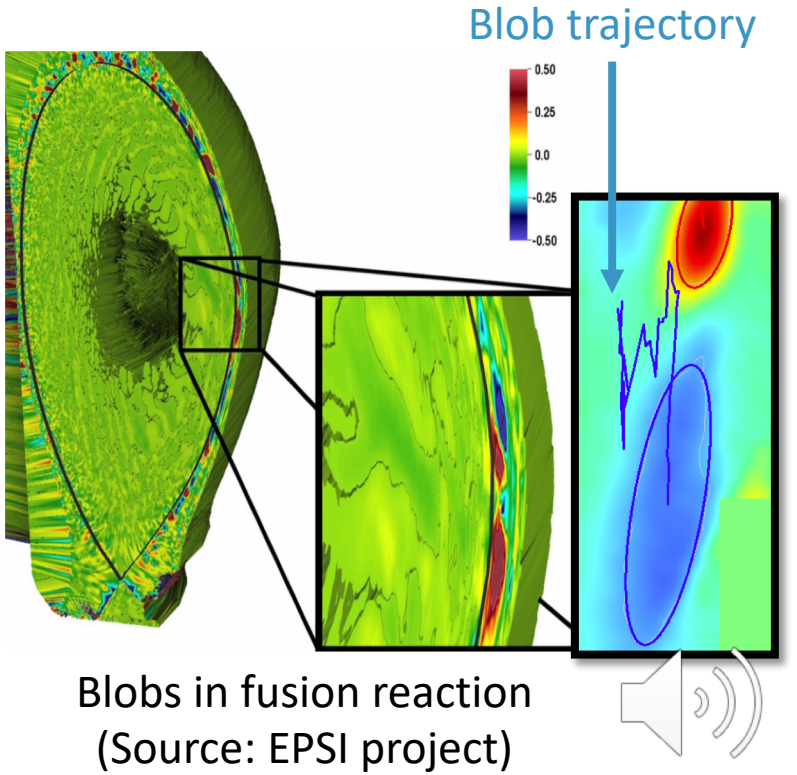
Impacts

- Improve user productivity with a unified API for many storage systems
- 43 BeStMan deployments worldwide and 5 backend deployments for CERN EOS system, as of 2015
- Being used in scientific collaborations such as ESGF, OSG, and WLCG

Feature Extraction: Near Real Time Detection of Blobs

- Fusion plasma blobs
  - Lead to loss of energy from tokamak plasmas
  - Could damage multi-billion tokamak
- Experimental facility may not have enough computing power for necessary data processing
- Distributed *in transient* processing
  - Make more processing power available
  - Allow more scientists to participate in data analysis and monitor the experiment remotely
  - Enable scientists to share knowledge and expertise
- Lingfei Wu, Alex Sim, Jong Choi, M. Churchill, K Wu, S Klasky, CS Chang 10.1109/TBDATA.2016.2599929

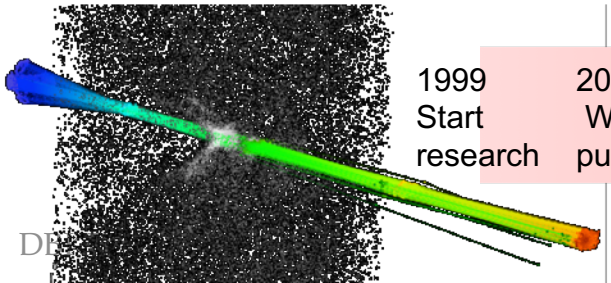




Blobs in fusion reaction
(Source: EPSI project)

FastBit Indexing

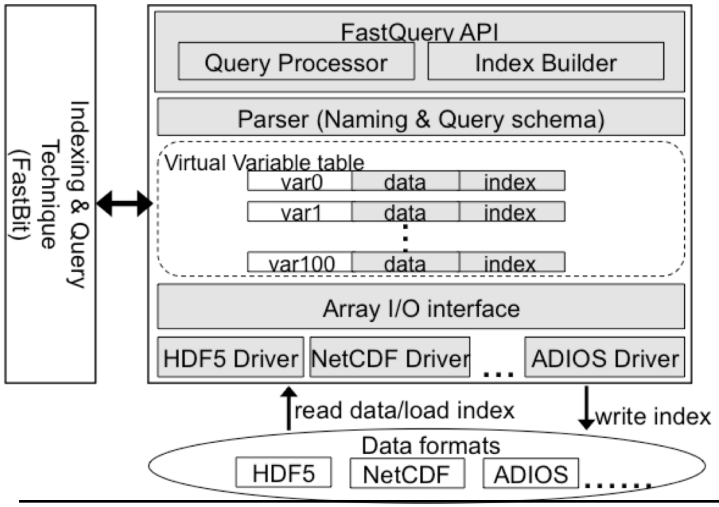
- **Problem**: given a large data collection, quickly find records satisfying user-specified conditions
  - Example: in billions of high-energy collision events, find a few thousand based on energy level, number of particles and so on
- **Solutions**
  - Algorithmic research: developed new indexing techniques, achieved 10-100 fold speedup compared with existing methods
  - Efficient software: open source (2008), received R&D 100 award
- **Science use cases**
  - Laser Wakefield Particle Accelerator: FastBit acts as an efficient back-end for identifying and tracking particles (lower left figure)
  - Combustion: FastBit identifies ignition kernels based on user specified conditions and tracks evolution of the regions
- **Testimonial** "FastBit is at least 10x, in many situations 100x, faster than current commercial database technologies" – Senior Software Engineer, Yahoo!

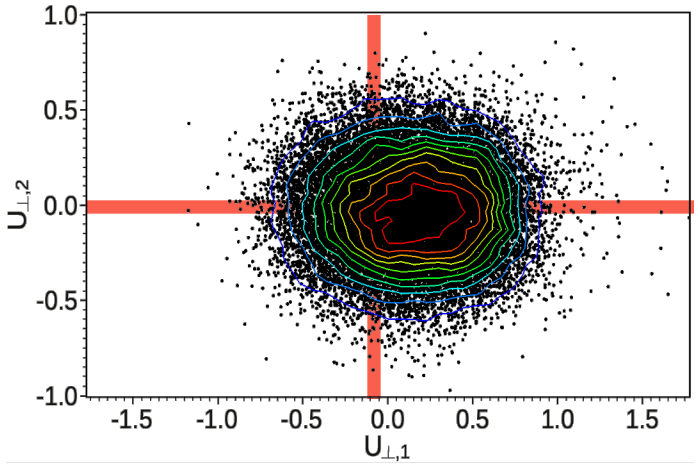


1999	2001	2004	2006	2007	2008
Start research	WAH published	WAH patented	- Query Driven Vis - Published theory	- FastBit released - BioSolveIT TrixX	- R&D100 Award - Yahoo!

[Wu, et al. 2009]

FastQuery: Parallel Queries on Science Files





Particle distribution of highly energetic particles around the region of magnetic reconnection. The off-centered and oblong distribution confirms the existence of a previously speculated property known as agyrotropy.

Key Ideas

- Provide uniform array interface for scientific data in commonly used file formats, e.g., HDF5, ADIOS, and NetCDF
- Provide efficient searching functionality on top of existing user analysis frameworks while expand data handling capability and improve user productivity

Results

- Indexed and queried a trillion particle dataset for studying magnetic reconnection (or "space weather")
  - ✓ Built index in 10 minutes
  - ✓ Located highly energetic particles in seconds
- "This is the first time anyone has ever queried and visualized 3D particle datasets of this size." -- Homa Karimabadi, Physicist from UCSD

VPIC IO Utilities(VIOU) for Fast Output

Scientific Achievement

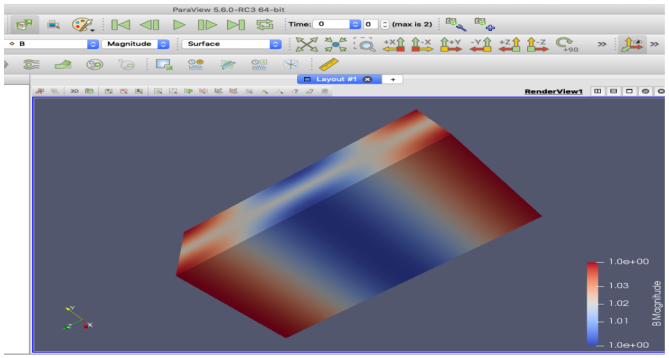
Reduce the cost of producing output files from a popular general purpose particle-in-cell simulation code VPIC for modeling kinetics of charged plasmas in magnetic fields. Due to the large number of particles need to represent plasma in magnetic field, the checkpointing files are very large. This utility leverages HDF5 to reduce I/O cost and generate portable self-describing output files.

Significance and Impact

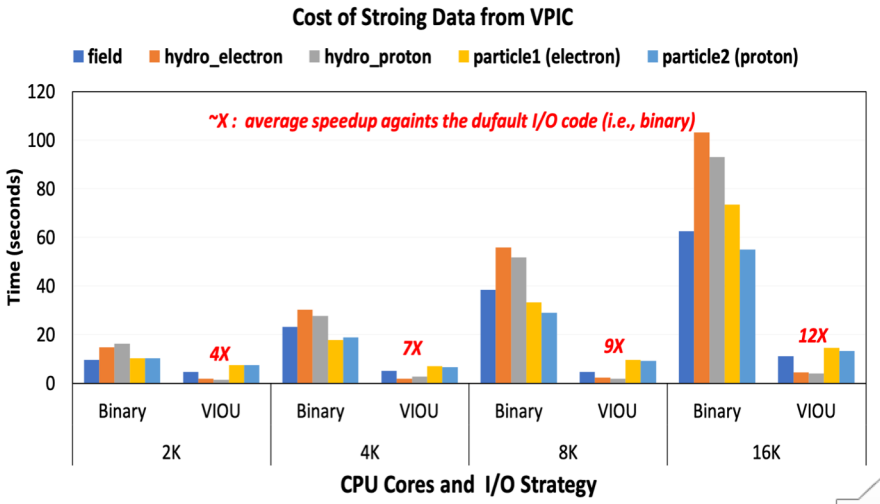
- The VIOU is going to be merged into the main VPIC repo at github by LANL
- The VPIC has wide applications in particle accelerators, space weather, and so on.

Research Details

- Explore parameter space for I/O operations: collective I/O, stripe size, number of OSTs, multiple (sub-)files, compound data structure, async I/O, and so on.
- Find the parameter combinations to optimize I/O performance



XDMF based Visual Inspection in ParaView and others



Developer : Bin Dong, Suren Byna, Kesheng Wu (LBNL)
Other Collaborator: Patrick Kilian, Borb Bird, Fan Guo, Qile Zhang (LANL),Scot Breitenfeld (HDF5)

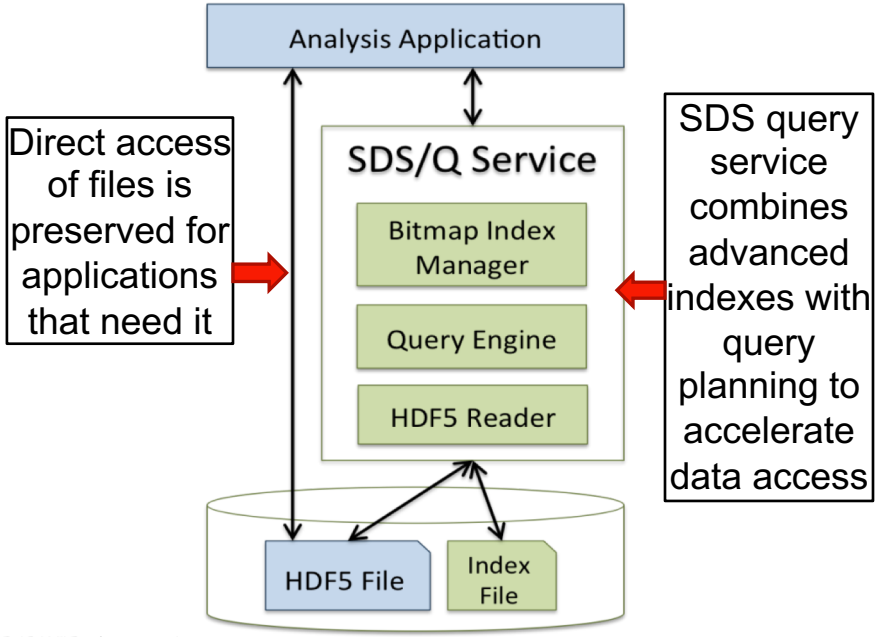
Working Directly with Scientific Data Files is Good for Performance

Problem

- Astronomy project Palomar Transient Factory (PTF) produces ~10GB image every 45 minutes
- Currently, data is loaded into a PostgreSQL database system
- Loading takes nearly 45 minutes, thus preventing real-time data analysis

Key Ideas

- Parallel query processing directly on data files
- Avoid costly loading of data to database management systems
- Leverage efficient FastBit indexes
- Implement database techniques to answer queries



Results

- Answering complex queries directly on scientific data files
- Complete the PTF query 40X faster than PostgreSQL, and 10X faster than Hive, the Hadoop database system
- Reduce the data loading time from 45 minutes to about 1 minute

[Blanas, et al. SIGMOD 2014]

Asynchronous HDF5 Reduce Output Latency

Scientific Achievement

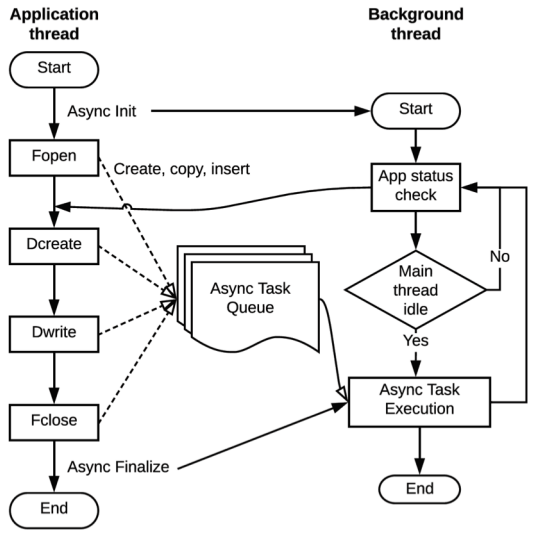
- I/O latency can slow down applications and waste precious computing resources while they wait to move data between storage and memory. Asynchronous I/O in HDF5 hides data movement costs by executing I/O operations in the background.

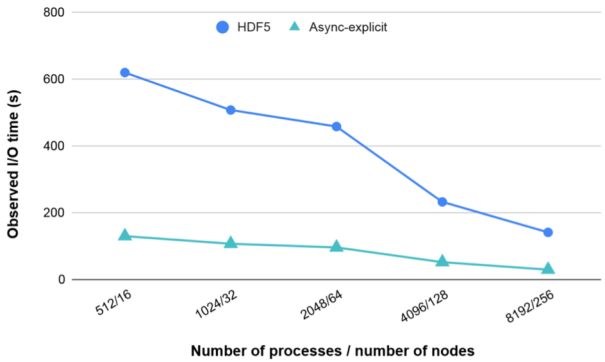
Significance and Impact

- Applications can use asynchronous I/O without major code changes
- Performance evaluation demonstrates multi-fold speedup for applications that read/write data periodically

Research Details

- Asynchronous I/O, developed as an HDF5 VOL connector, uses background threads to execute I/O operations asynchronously
- Two modes are supported:
  - Implicit mode for unmodified applications, but less performance benefit
  - Explicit mode for applications that want more control of I/O operations to fully benefit from asynchronous operations
- Asynchronous I/O can hide the majority of application I/O time by overlapping it with time spent computing or communicating, making I/O visible only at the last time step for write or the first step for read





Castro/AMReX performance on Cori between the default synchronous 5 asynchronous (explicit mode) writing 5 timesteps of data . Observed I/O time includes the last timestep's write time and the overhead of the asynchronous I/O framework for all timesteps. 2-4x improvement seen in observed I/O performance.

H. Tang, Q. Koziol, S. Byna, J. Ravi, "Transparent Asynchronous Parallel I/O using Background Threads", IEEE TPDS Special Section on Innovative R&D toward the Exascale Era, 2021.

Similarity-based Compression with Multidimensional Pattern Matching

Scientific Achievement

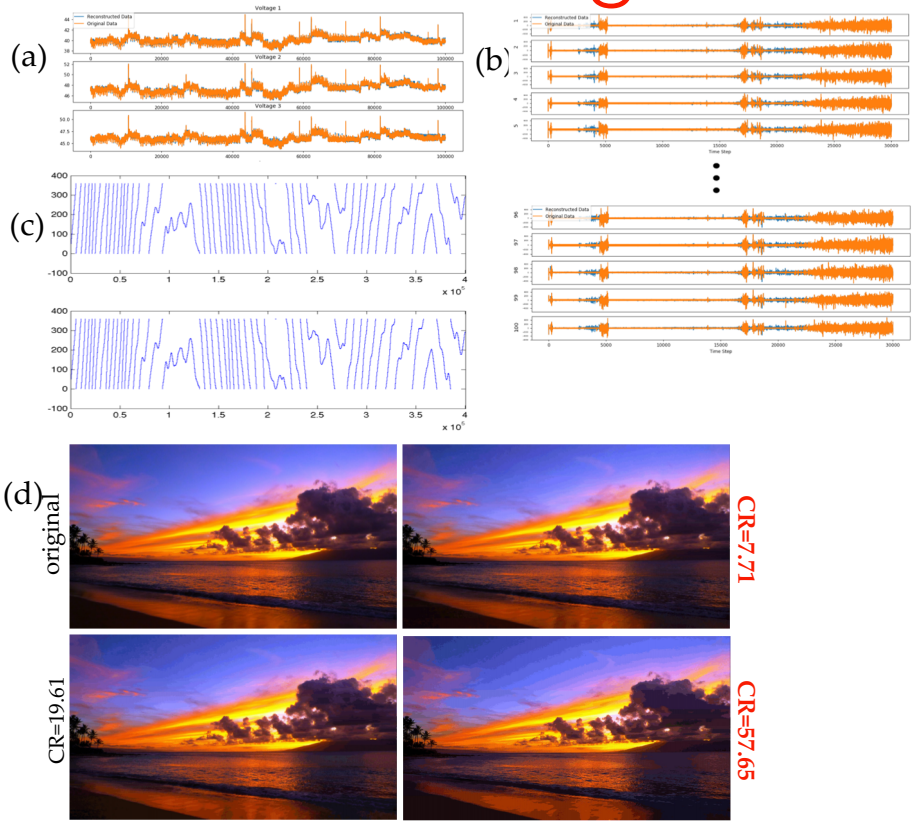
Newly extended the IDEALEM (Implementation of Dynamic Extensible Adaptive Locally Exchangeable Measures) algorithm to support 2D and N-dimensional data with multidimensional similarity features

Significance and Impact

We developed extended approaches in statistical similarity based data compression algorithm with multidimensional pattern matching, which is a promising alternative to leading lossy compression algorithms.

Research Details

- Our study extended the algorithm to support multi-dimensional measures with Dynamic Time Warp (DTW) and Minimum Jump Cost (MJC).
- IDEALEM also offers transforming non-stationary data such as phase angle of electricity data into locally stationary block to promote exchangeability/similarity
- IDEALEM method supports event/feature detection directly on the compressed data.
- IDEALEM can be applied to photos and videos, in addition to scientific multidimensional floating point data.
- U.S. Patent No. 10,366,078, 2019



(a) Compression on 3-dimensional power grid measurements
(b) Compression on the distributed acoustic sensing measurement dataset with 100 dimensions
(c) Compression on the phase angle of electricity measurement data, with CR= 56.56
(d) Photo compression sample: Original photo (2560x1440). Each of ~3.68 million pixels has three dimensions of color (RGB).

O. Del Guercio, R. Orozco, A. Sim, K. Wu, "Similarity-based Compression with Multidimensional Pattern Matching", the 2nd International Workshop on Systems and Network Telemetry and Analytics (SNTA 2019), 2019. doi: 10.1145/3322798.3329252

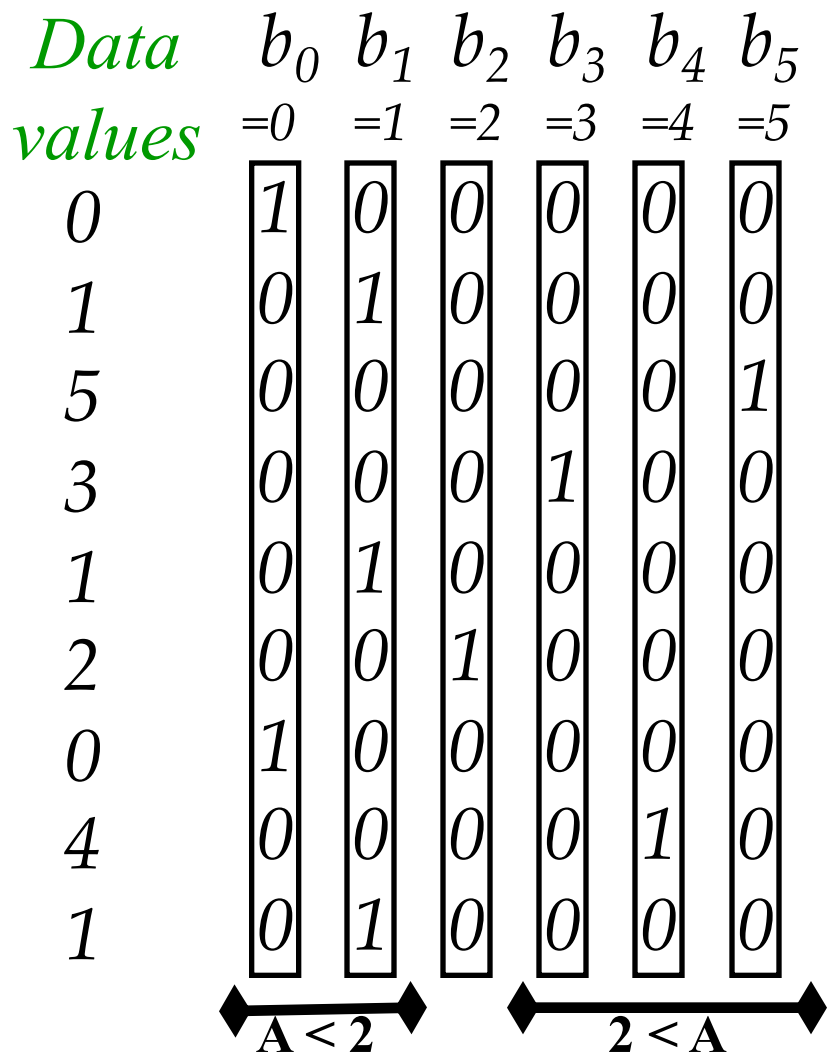
Scientific Data Management at Exascale

John Wu

Outline

- ❑ Introduction
- ❑ Examples of Scientific Data Management
- ❑ Deep Dive: FastBit Indexing
- ❑ Deep Dive: FasTensor Automatic Parallelization Engine
- ❑ Summary

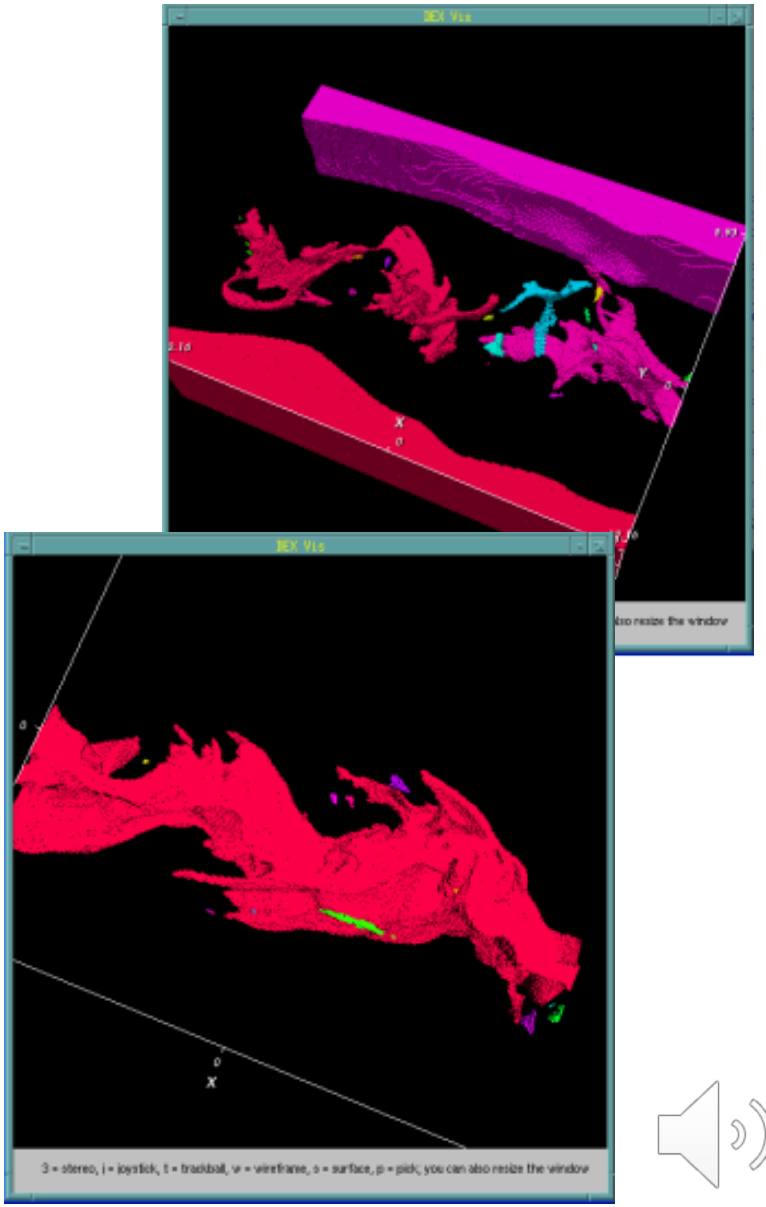
Basic Bitmap Index



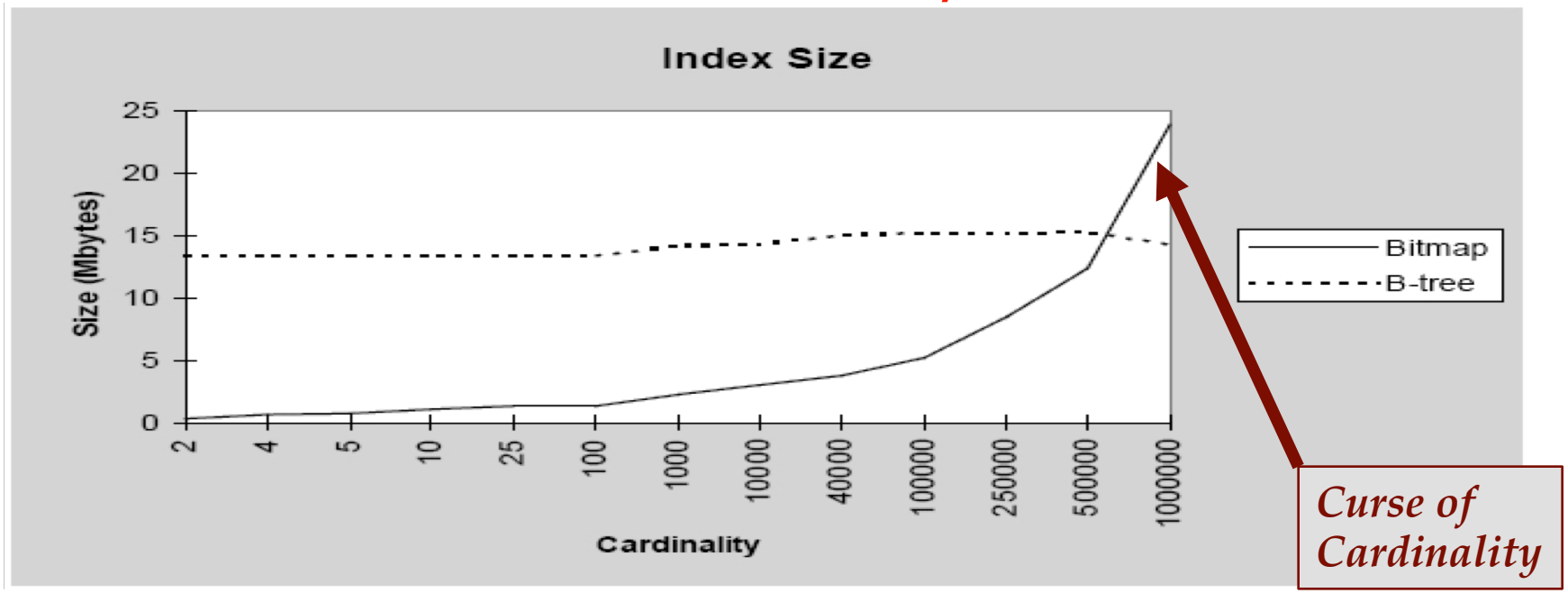
- First commercial version
  - Model 204, P. O'Neil, 1987
- Easy to build: faster than building B-trees
- Efficient for querying: only bitwise logical operations
  - $A < 2 \rightarrow b_0$ OR b_1
  - $A > 2 \rightarrow b_3$ OR b_4 OR b_5
- Efficient for multi-dimensional queries
  - Use bitwise operations to combine the partial results
- Size: one bit per distinct value per row
  - Definition: Cardinality == number of distinct values
  - Compact for low cardinality attributes, say, cardinality < 100
  - Worst case: cardinality = N, number of rows; index size: N*N bits

However, There is a Catch

- The efficiency of bitmap indexes decreases as the number of distinct values increases!
- Definition: column cardinality = number of distinct values of a column in a dataset
- As column cardinality increase,
  - The index size increases
  - The query responses time increases
- Some restrictions on using the bitmap index include: The indexed columns must be of low cardinality—usually with less than 300 distinct values.
  - How and when to use Oracle9i bitmap join indexes, Donald Burleson, November 12, 2002
- A value-based bitmap for processing queries on low-cardinality data. (Recommended for up to 1,000 distinct values …
  - Introduction to Adaptive Server IQ, Ch 5, Sybase



Curse of Cardinality: Evidences



- Index sides, adapted from a presentation by Hakan Jakobsson, ORACLE, 1997 (Stanford Database Seminar)
- 1 million rows (bitmap index compressed with BBC)
- Sizes of compressed bitmap indexes increase with column cardinality – this is generally the case, not just in ORACLE, see formula below (VLDB'06)

$$E(C_1) + \sum_{i=2}^{A} \left(E(C_i) \prod_{j=1}^{i-1} C_j - C_i \left[\left(\prod_{j=1}^{i-1} C_j \right) - 1 \right] \right)$$

Ways to Improve Bitmap Indexes

- **Compression**
  - Byte-aligned Bitmap Code (BBC), used in ORACLE
  - Word-Aligned Hybrid (WAH) code, used in FastBit, produce optimal bitmap indexes [Wu, et al. TODS 2006]
  - In the worst cases, the index sizes are still larger than B-trees
- **Encoding** [Wu, et al. TODS 2010]
  - Many bitmap encoding schemes exist, the most compact is the binary encoding
  - The binary encoded index (bit-slice index) is slower than the projection index in the worst case
- **Binning**
  - Designed to handle high-cardinality data, but needs to scan raw data, which makes it slower than the projection index
  - Solution: Order-preserving Bin-based Clustering (OrBiC) [Wu, et al, 2008]

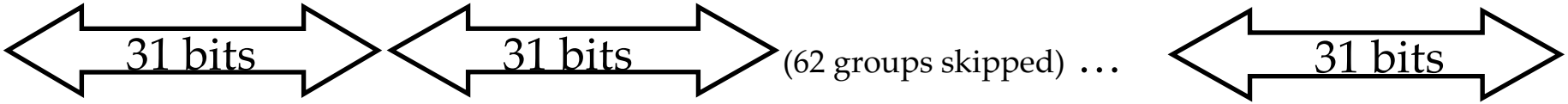
FastBit Technology 1: Compression

[Wu, Otoo, and Shoshani 2006]

Example: 2015 bits

100000000000000000000111000000000000000000000000000000.....000000000000000000000000000000011111111111111111111111

Main Idea: Use run-length-encoding, but...
partition bits into 31-bit groups [not 32 bit] on 32-bit machines



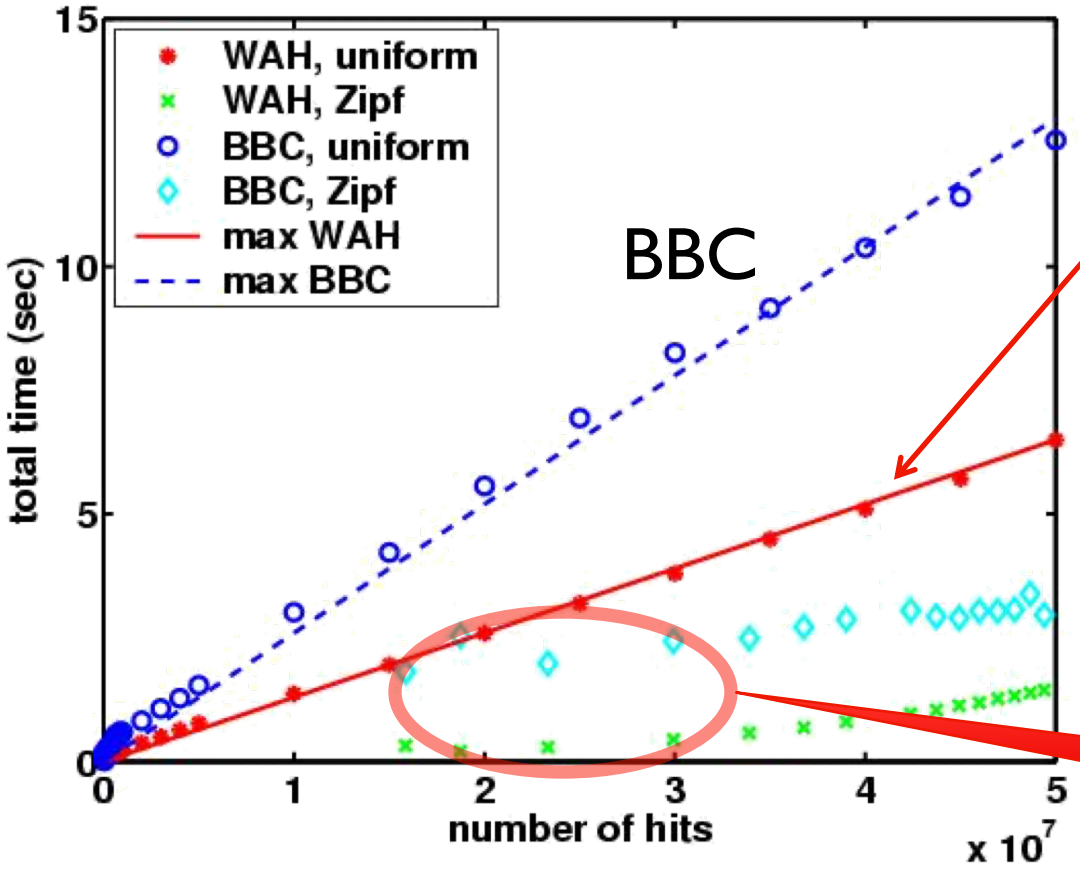
Merge neighboring groups with identical bits

Encode each group using one 32-bit word

- Name: Word-Aligned Hybrid (WAH) code (US patent)
- Key features: WAH is compute-efficient
  - Uses the run-length encoding (simple)
  - Allows operations directly on compressed bitmaps
  - Never breaks any words into smaller pieces during operations
  - Worst case index size 4N words, not N*N (without compression)

WAH Compressed Index Is Optimal

- In the worst case, query response time is a linear function of the number of hits, H
- WAH Compressed indexes are optimal for one-dimensional range queries, search time O(H)



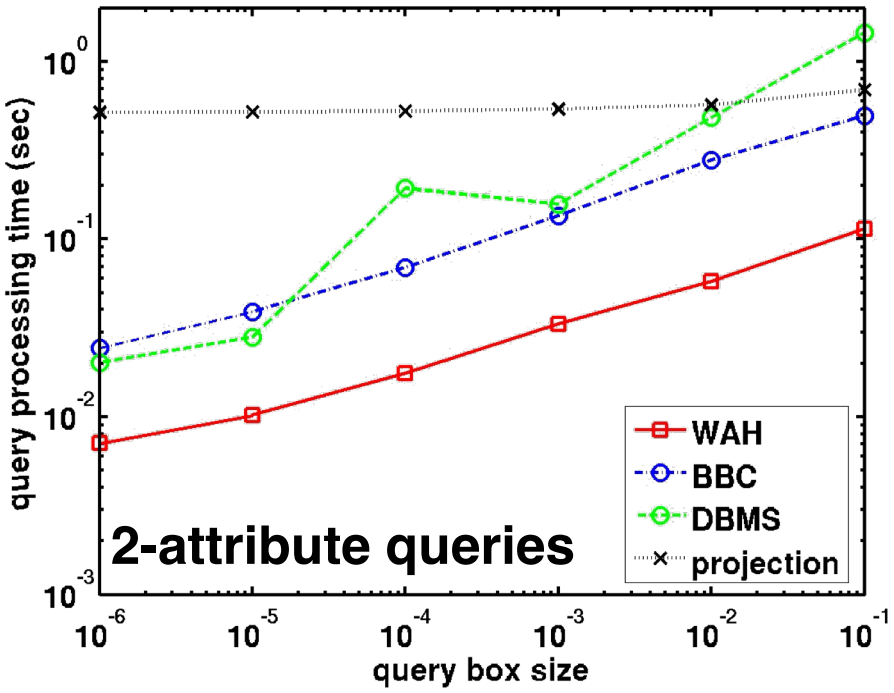
Better

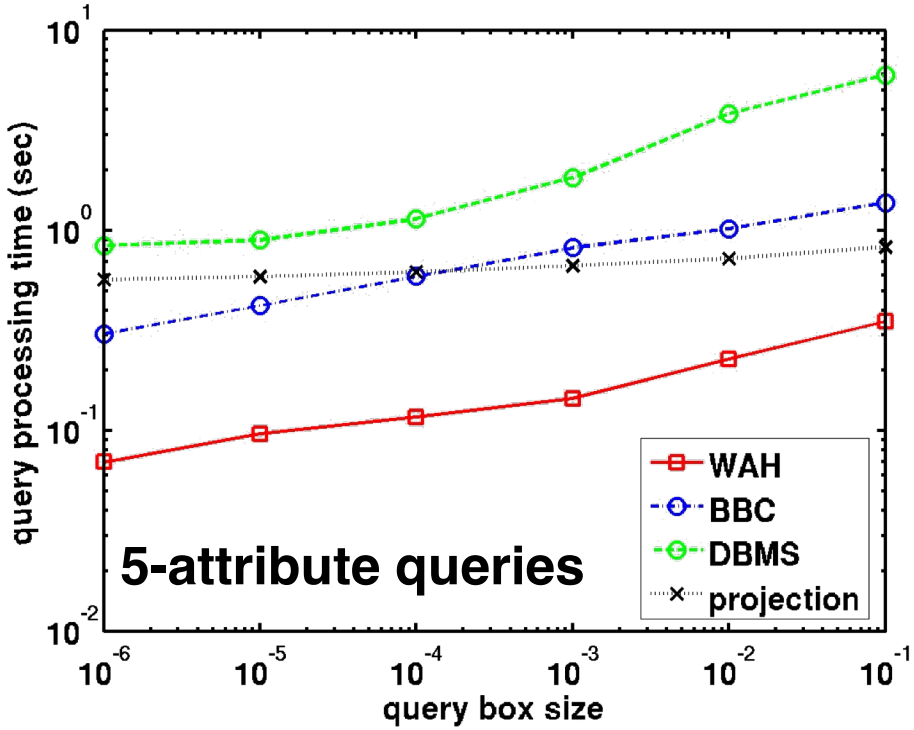
Optimality means straight line

Pentium 4, 2.8 GHz
40 MB/s disk array
100 M synthetic records

Expand this area

Compressed Index Performance

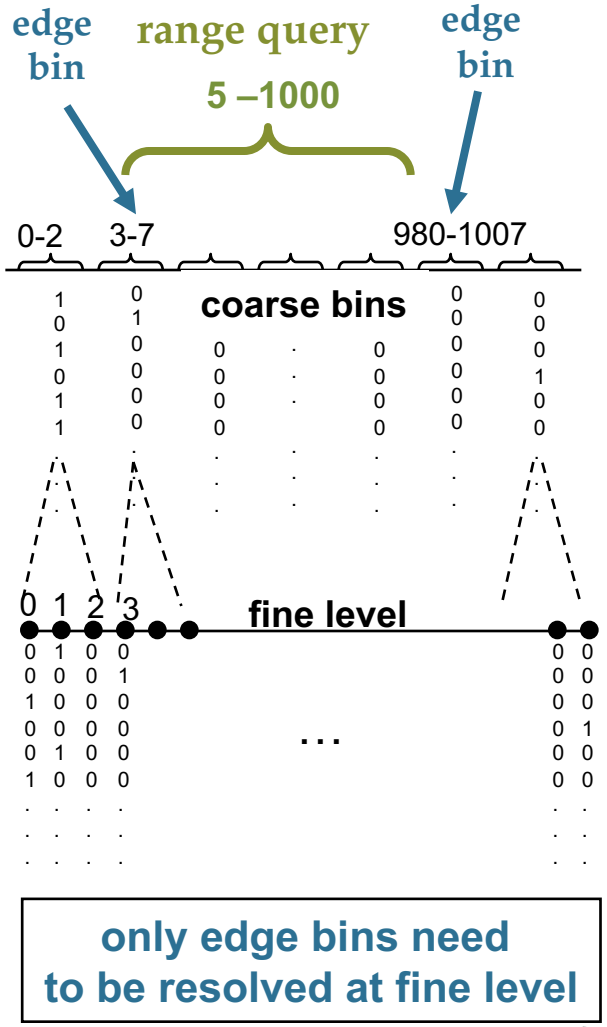




- **WAH compressed indexes are 10X faster than DBMS, 5X faster than our own version of BBC**
- Based on 12 most queried variables from a STAR dataset with 2.2 million rows, average column cardinality 222,000

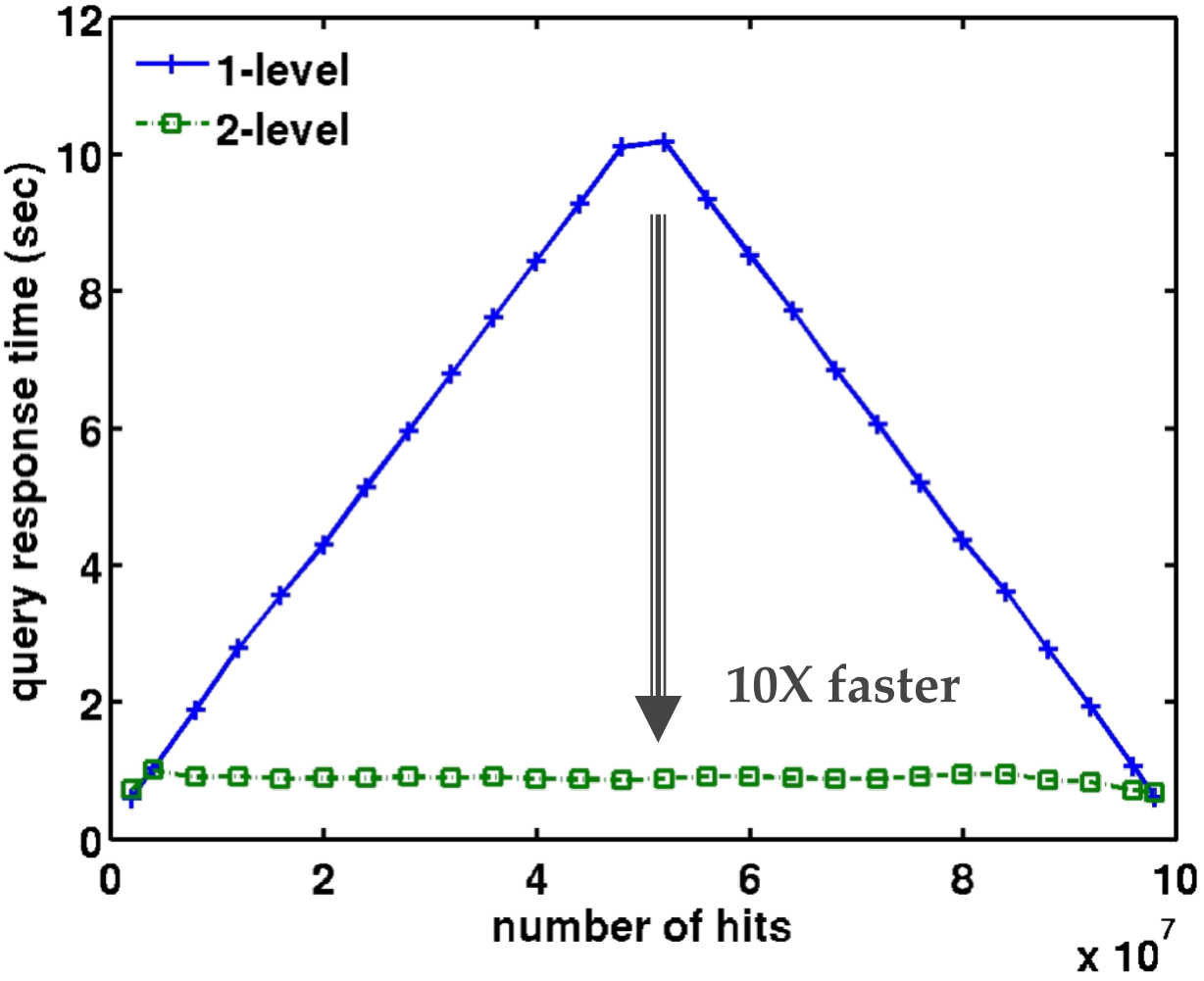
FastBit Technology 2: Binning

- Binning reduce the number of bitmaps used in an index, which reduces index size and query processing time
- Binning is also useful for building multi-level indexes (to be described next)
- Challenge: if query falls in the middle of edge bins
- Solution: Order-preserving Bin-based Clustering (OrBiC)
- 5x speedup for searching bins



FastBit Technology 3: Multi-Level Encoding

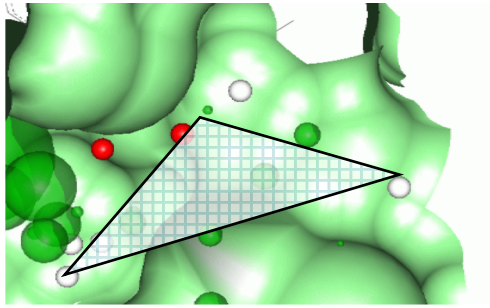
- Prove theoretically that the second level needs only a small number of bins (15 ~ 50 depending on the skewness of the data)
- Only two levels are needed
- Result: 5X speedup on average
- Combined with WAH, could achieve 50X speedup

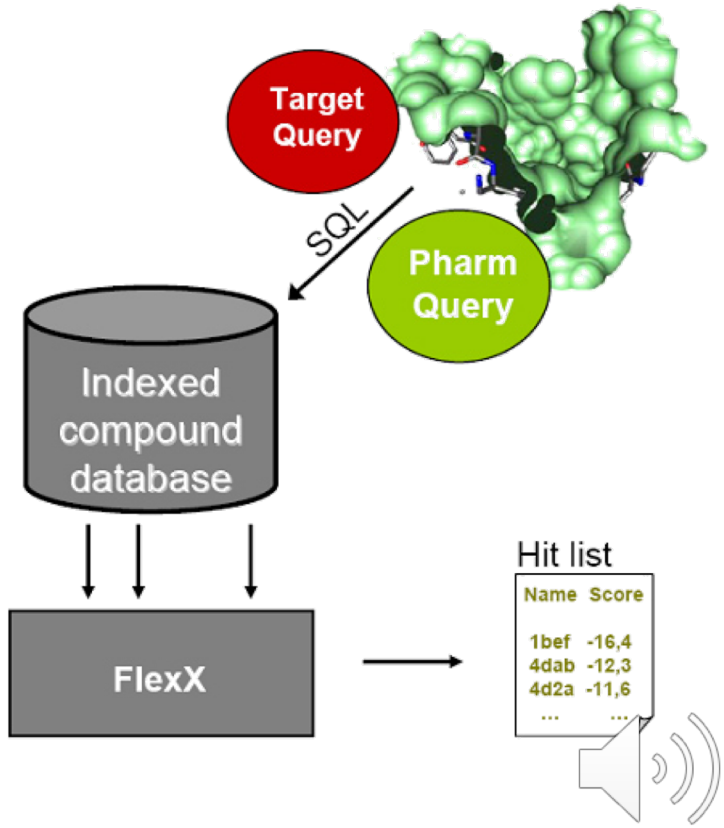


[Wu, Shoshani and Stockin 2010]

FastBit Application Example: Molecular Docking

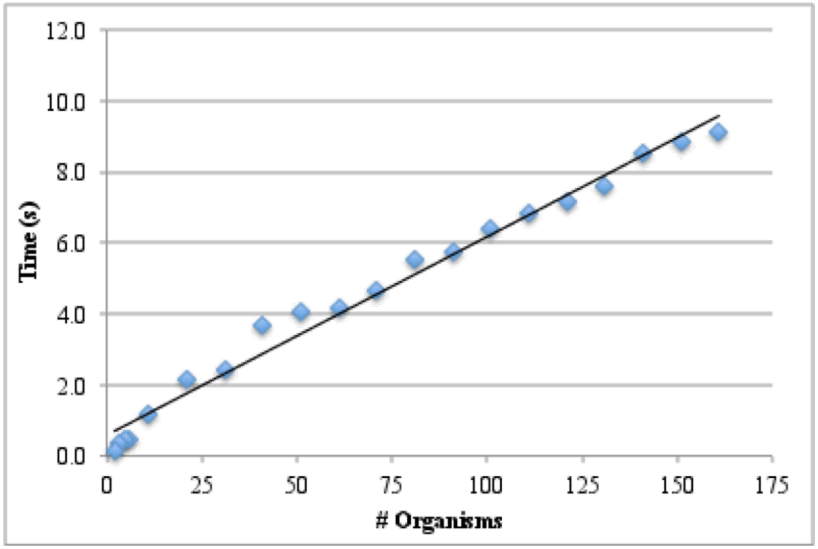
- **From: Jochen Schlosser** [schlosser@zbh.uni-hamburg.de]
  University of Hamburg
- **Application:** Structure-based virtual screening (J. Chem. Infor. Mod., 2009)
- **Specification of the descriptor as triangle geometry**
  - Types of interaction centers
  - Triangle side lengths
  - Interaction directions
  - 80 bulk dimensions
- **Receptors**
  - Receptor descriptors are generated similarly
  - Using complementary information where necessary
- **Use of pharmacophore constraints on receptor triangles**
  - Reduces number of queries
  - Improved query selectivity because the pharmacophore tends to be inside the protein cavity
  - 250X faster than previous docking software

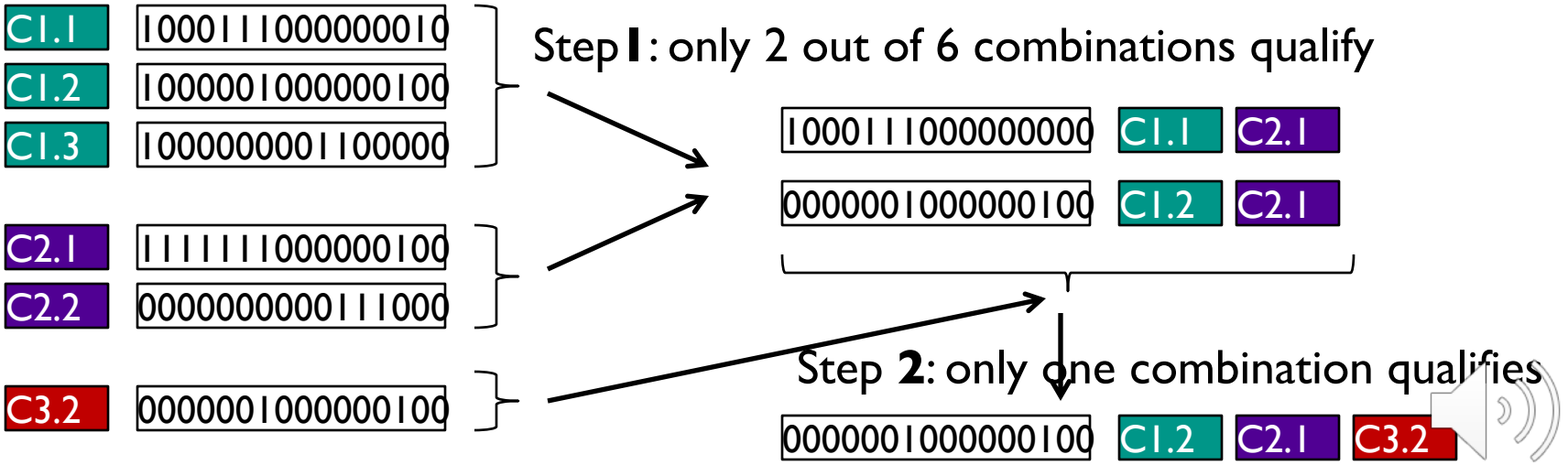




Gene Context Analysis Query Processing: Timed out with DBMS; but <10 s with FastBit

- Concentrate on maximal matches only
  - Don't list small matches
- Progressively match more organisms
  - Build up answers gradually, eliminate empty combinations
- Performance
  - Match 161 organisms in 10 seconds on 8000 genomes and 3.3 million cassettes
- Available in IMG <https://img.jgi.doe.gov/> (Romosan et al.)





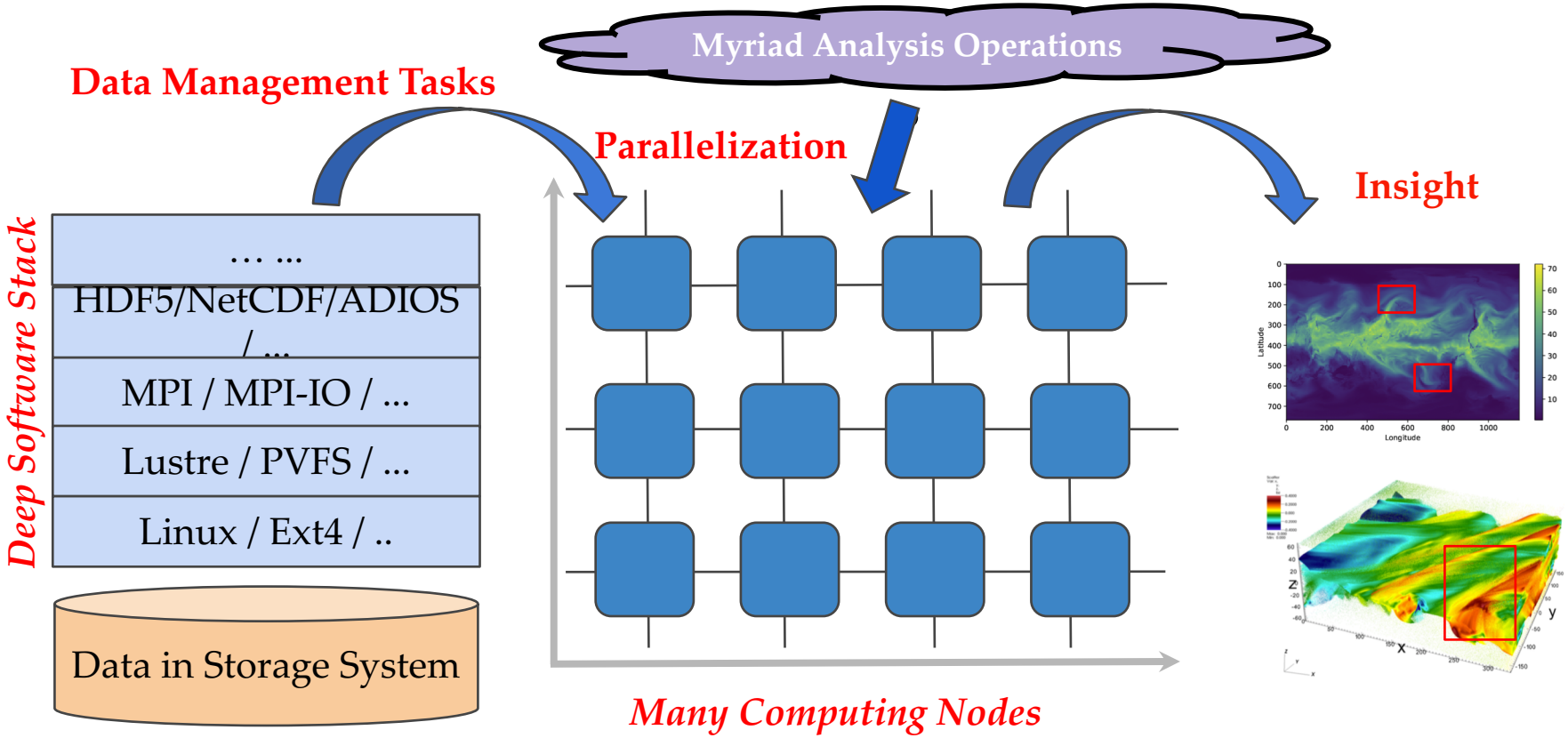
Scientific Data Management at Exascale

John Wu

Outline

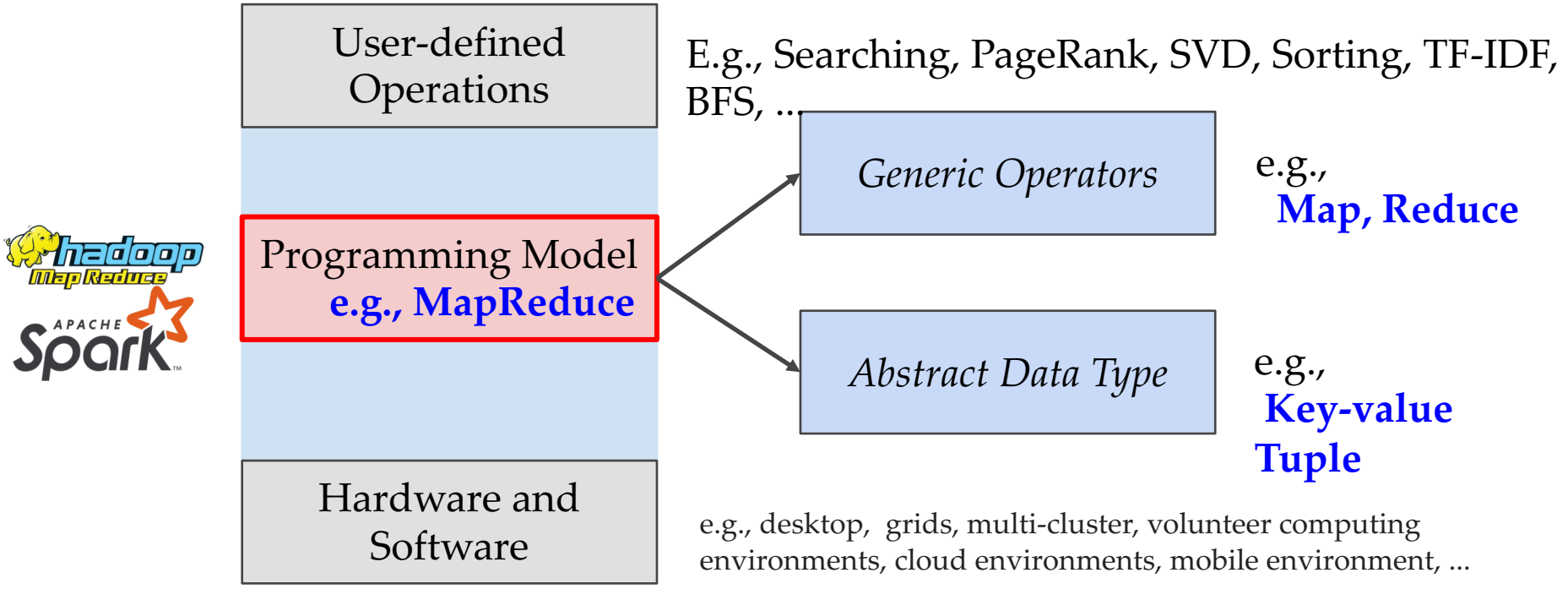
- ❑ Introduction
- ❑ Examples of Scientific Data Management
- ❑ Deep Dive: FastBit Indexing
- ❑ Deep Dive: FasTensor Automatic Parallelization Engine
- ❑ Summary

Parallelizing Scientific Data Analysis is Difficult



Commercial Big Data Systems Simplify Parallel Computing

Data Parallel Programming: generic programming abstraction for various data analysis operations + execution engine to hide complex data management tasks

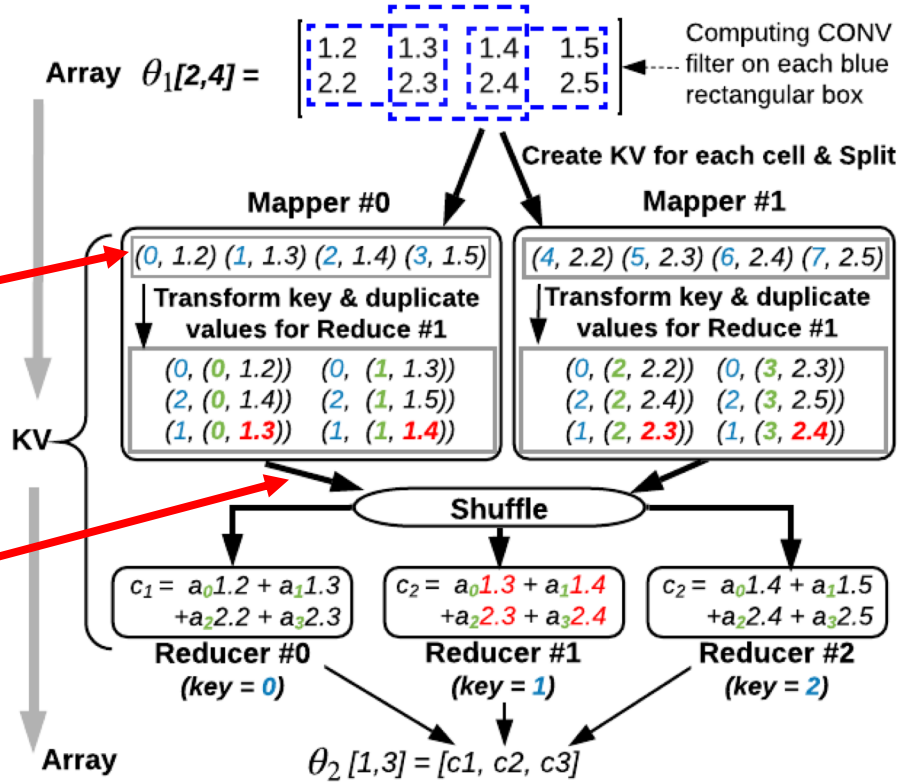


Challenge: Many Operations are Hard in MapReduce

Example:

Convolution on a 2 by 4 2D Tensor

Kernel is 2 by 2



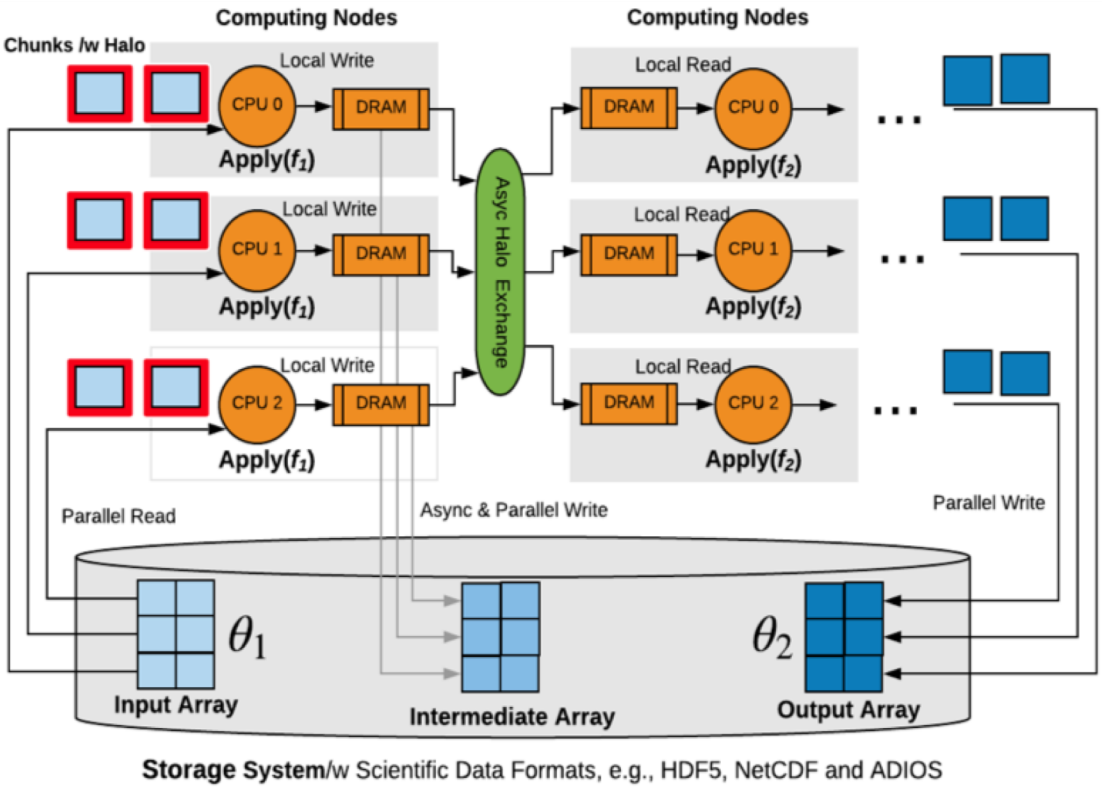
1. Mismatched Data Model
-- Convert Tensor to KV list at Map stage

2. Expensive reduce operations
-- Duplicate KV for Reduce stage

Additional examples for Connected Component Labeling (CCL), gradient computing, and interpolation are available in Dong et al 2019

SLOPE Execution Engine in FasTensor

1. MPI based Single Program Multiple Data (SPMD) Pattern
2. Directly on Scientific Data Formats, e.g., HDF5
3. Manual/auto-chunking & ghost zone building
4. Efficient Parallel I/O



Evaluations

1. Cori Supercomputer at NERSC* with over 2400 nodes
2. Peer systems for comparison

   Apache Spark, TensorFlow, C++ Imp (hand optimized)

3. Workload

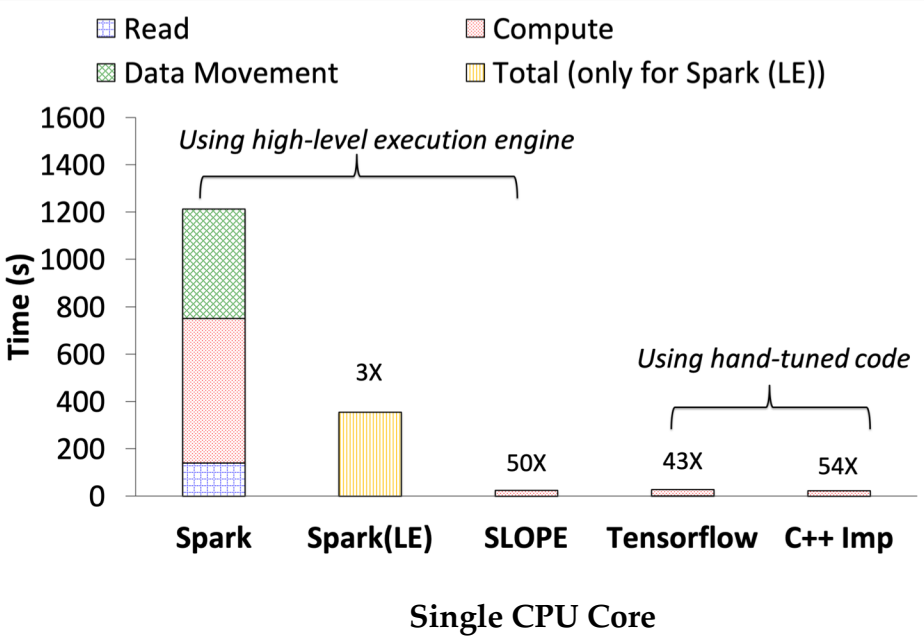
   Synthetic Workloads: 2 layers CNN with forward steps

   Real Applications Workloads: CAMR5, VPIC, BISICLES

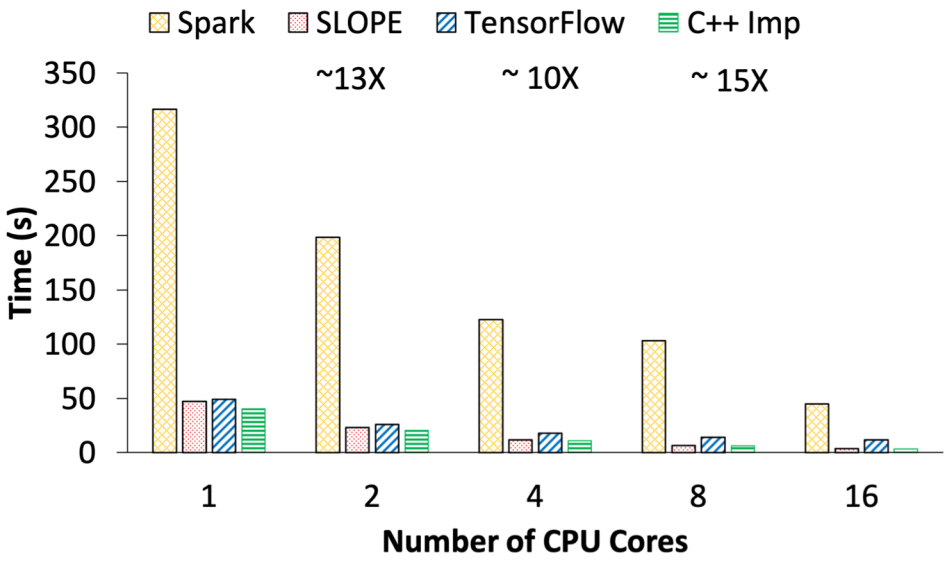
*National Energy Research Scientific Computing Center https://www.nersc.gov/systems/cori/

Comparing SLOPE, TensorFlow, Spark

Synthetic Workloads: 2 layers CNN with forward steps on a 64K by 64K 2D array



Single CPU Core

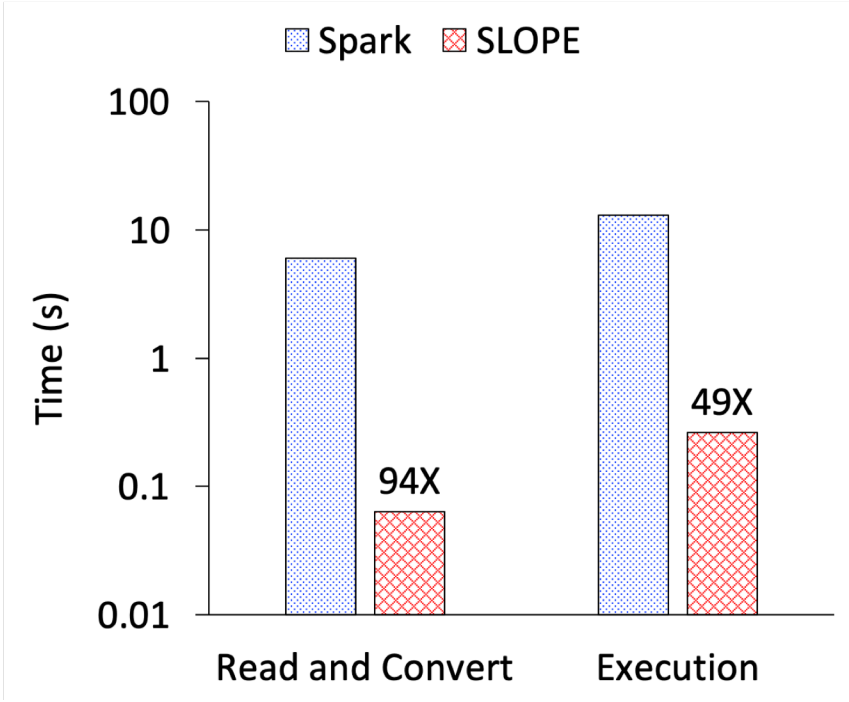


SLOPE is up to *106X* faster on real-applications

CAM5

768 × 1152 2D array

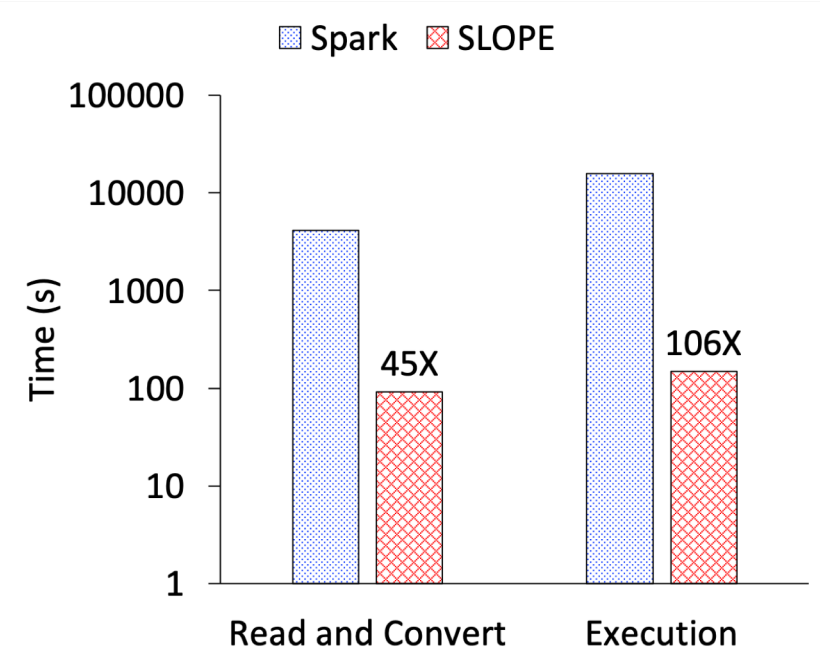
3 Layers CNN (CONV, ReLU, Pooling)



VPIC

512 x 256 x 256 3D filed array + 263 GB particle data

Gradient and interpolation



SLOPE could be 92,824X faster

BISICLES

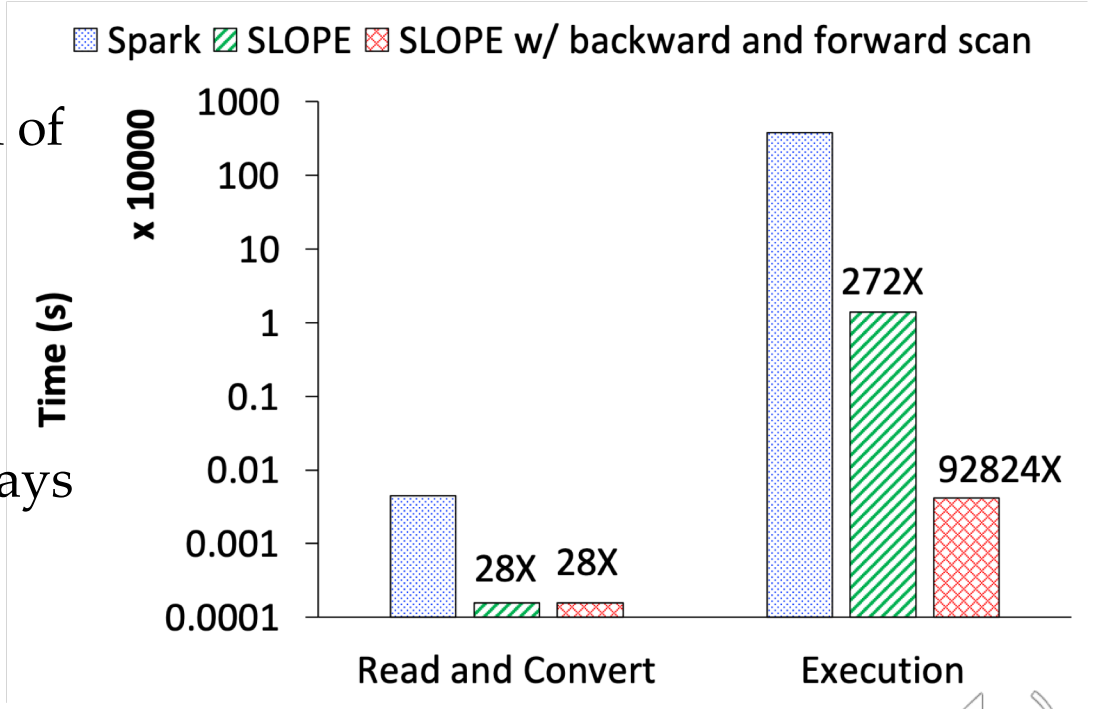
12288 x 12288 2D array

Connected Component Labeling (CCL)

	# of steps to convergence
w/o advanced feature	*10868*
w/ advanced feature	*8*

SLOPE advanced features

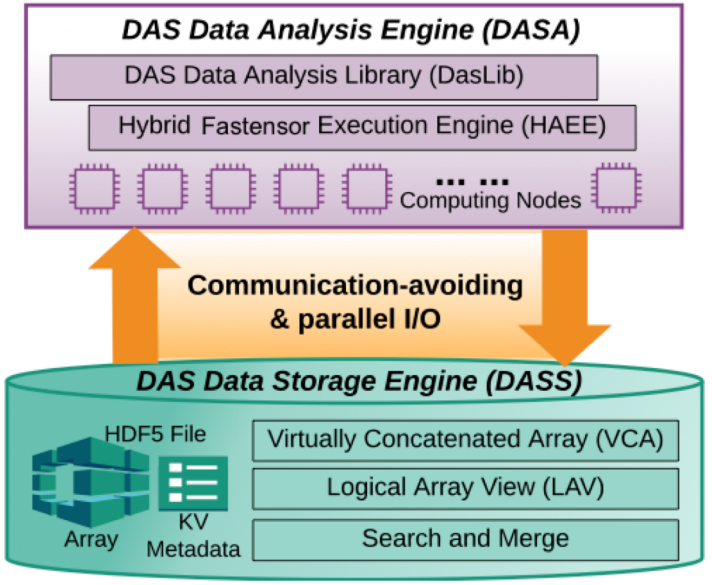
- iterate over many invocation of UDFs
- distributed array cache
- complex execution patterns: forward and backward scan
- in-place modifications of arrays



DASSA: FasTensor for Distributed Acoustic Sensing

Scientific Achievement

Distributed acoustic sensing (DAS) records strain or strain-rate along fiber-optic cables in subsurface at high frequency and large geophysical scale, producing mounts of data. This tool uses FasTensor to implement a variety of data analysis operations on high-performance computing systems for fast event detection.



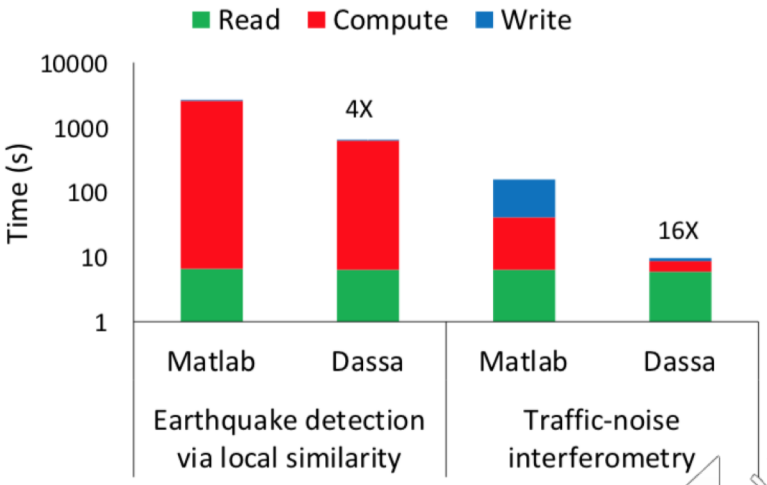
Significance and Impact

Reduced the execution time of data analysis pipeline to identify earthquake from a set of distributed acoustic sensing data from weeks to seconds

Research Details

Extend Fastensor to support DAS specific data analysis operations

- DasLib: sequential DAS data analysis operation
- Multithreaded Fastensor: auto-parallel DasLib with less data duplications
- DAS Storage Engine: efficient access to DAS data scattered among many small files
- Software available at http://sdm.lbl.gov/fastensor



B. Dong, V. R. Tribaldos, X. Xing, S. Byna, J. Ajo-Franklin and K. Wu, "DASSA: Parallel DAS Data Storage and Analysis for Subsurface Event Detection," IPDPS 2020,

Scientific Data Management at Exascale

John Wu

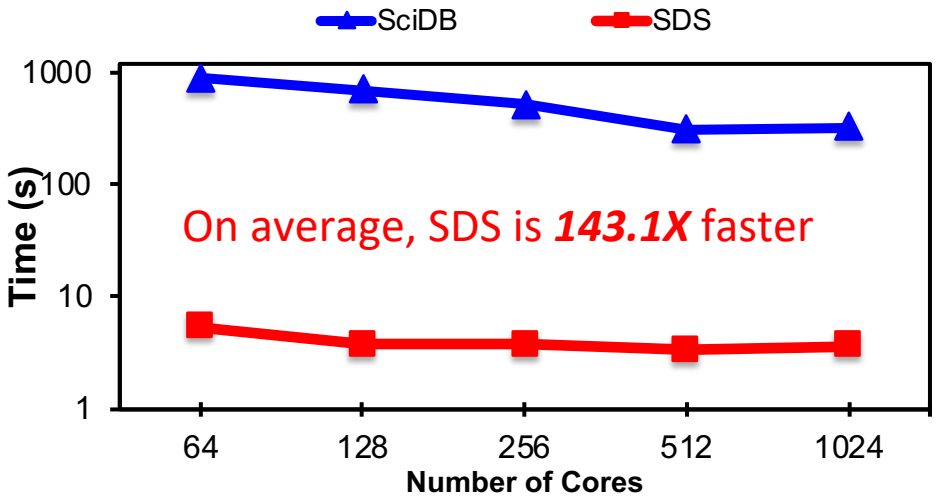
Outline

- ❑ Introduction
- ❑ Examples of Scientific Data Management
- ❑ Deep Dive: FastBit Indexing
- ❑ Deep Dive: FasTensor Automatic Parallelization Engine
- ❑ Summary

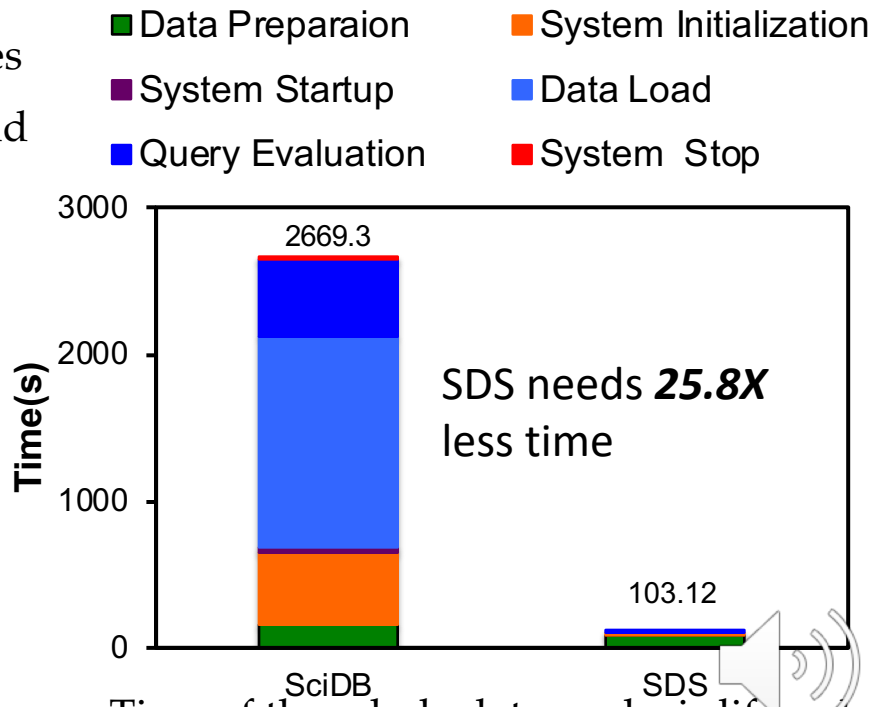
Scientific Data Management Means Many Things

Data Analysis Challenges	FasTensor Technologies
Heterogamous scientific data formats	Lightweight Two-level Metadata Model
Independent data organization cross data sets	MDBin: Multi-dimensional Array Binning
Common scientific operations (e.g., interpolation) turn into complex SQL expression (e.g., 9-way join)	SCJoin: Spatially Clustered Join algorithm

- Evaluations on Edison with tasks from analysis of magnetic reconnection (from Karimabadi, UCSD) [Dong, Byna, Wu, 2015]
- 2TB (HDF5 file) with seven 1D particle attributes
- Three 12 GB files (Binary) with 3D magnetic field



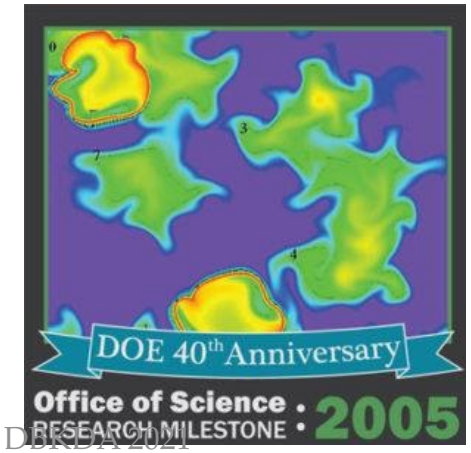
Query Running Time

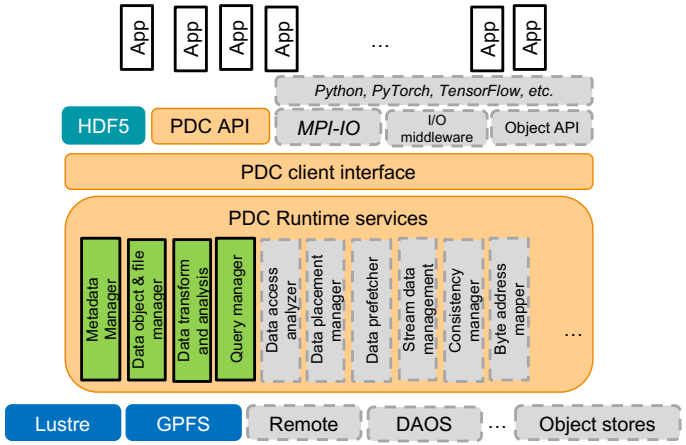


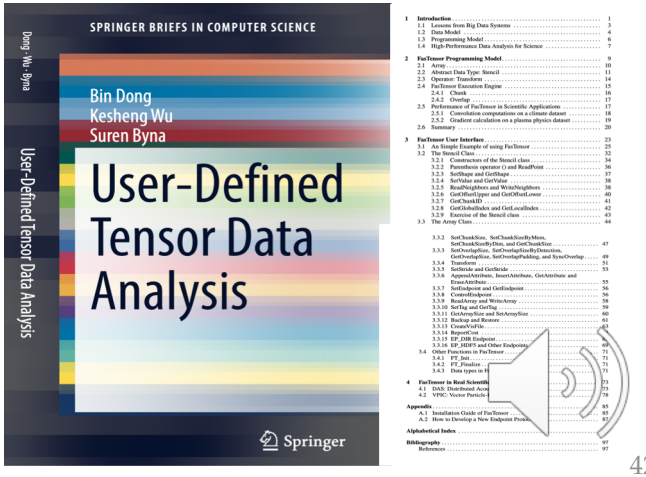
Time of the whole data analysis lifecycle

Summary

- Commercial Big Data tools may not match scientific application requirements, e.g., key-value pair data model can't easily represent multidimensional arrays, map-reduce not effective for some common operations
- Processing exascale data requires many different data management tools, e.g., FastBit, FasTensor, HDF5, ADIOS, …
- Exabytes of scientific data stored in files, directly working with these files is important to achieving good performance
- Right tools could make a big difference: up to 92,824X in one example
- A lot more to be done: (1) beyond simply arrays, such as AMR data, (2) GPU and accelerators, (3) deep storage integration, (4) distributed workflows, (5) supporting machine learning, …







Thanks!

FastBit software http://sdm.lbl.gov/fastbit

FasTensor software http://sdm.lbl.gov/fastensor

Team members:
J. L. Bez, S. Byna, B Dong, J. Gu, Q. Kang, M. Kiran, B. Mohammed, S. Shen, A. Sim, H. Tang
Computer science contributors:
H. Abbasi, E. W. Bethel, H Childs, J. Choi, J. Chou, M. Howison, K.-J. Hsu, C. Jones, S. Klasky, Q. Koziol, W-K Liao, K.-W. Lin, J. Liu, Q. Liu, J. Lofstead, J. Logan, K.-L. Ma, G. Ostrouchov, E. Otoo, M. Parashar, N. Podhorszki, E. Pourabbas. Prabhat, D. Pugmire, A. Romosan, O. Rübel, N. Samatova, K. Schwan, A. Shoshani, R. R. Sinha, K. Stockinger, Y. Tian, A. Uselton, G. Weber, M. Winslett, M. Wolf, W. Yoo, W. Yu
Application science contributors:
CS Chang, J. Chen, M. Churchill, W. Collins, E. Cormier-Michel, W. S. Daughton, S. Ethier, C. Geddes, H. Karimabadi, W. Koegler, J. Lauret, Z. Lukić , V. Markowitz, K. Mavrommatis, Md. Mostofa, P. Nugent, Ali Patwary, V. Perevoztchikov, A. M. Poskanzer, V. Roytershteynz, R. Strelitz, K. Suzuki, M. F. Wehner, W. Zhang
Funding:
Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract number DE-AC02-05CH11231