# Explaining the Fundamental Secrets of the Universe: How Spark and HPC might help Experimental High Energy Physics Analyses

Saba Sehrish

2016

ANITA BORG INSTITUTE
GRACE HOPPER CELEBRATION
OF WOMEN IN COMPUTING

#GHC16

U.S. DEPARTMENT OF ENERGY
Office of Science

Fermilab

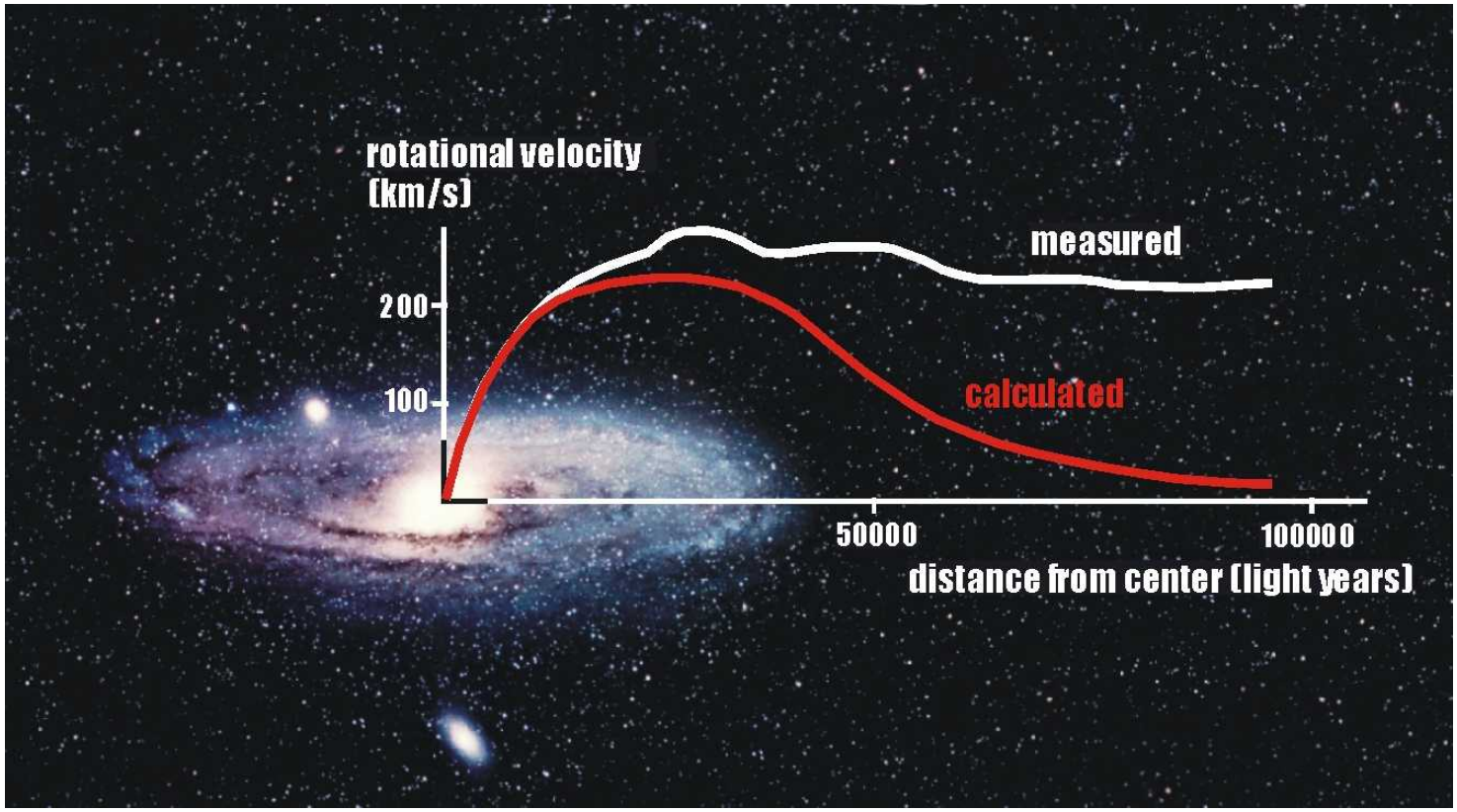
ANITA BORG INSTITUTE

acm Association for Computing Machinery

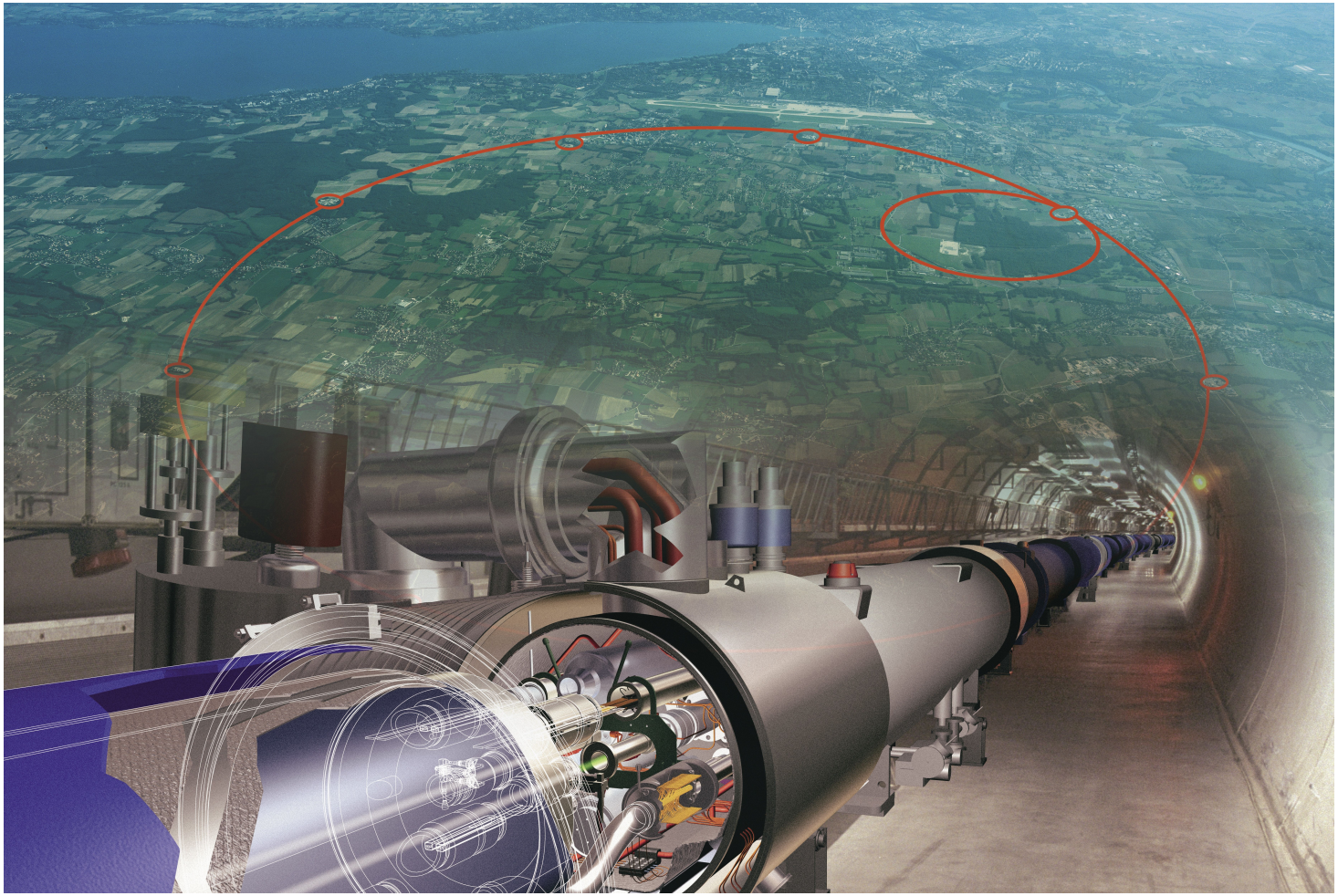
# Goal

- Can big data tools and High Performance Computing (HPC) resources benefit data- and compute-intensive statistical analyses in high energy physics (HEP) to improve time-to-physics?
- We use Spark to implement an active Large Hadron Collider (LHC) analysis, searching for Dark Matter with the Compact Muon Solenoid (CMS) detector as our use case, and evaluate its performance on the supercomputing resources provided by National Energy Research Scientific Computing Center (NERSC).

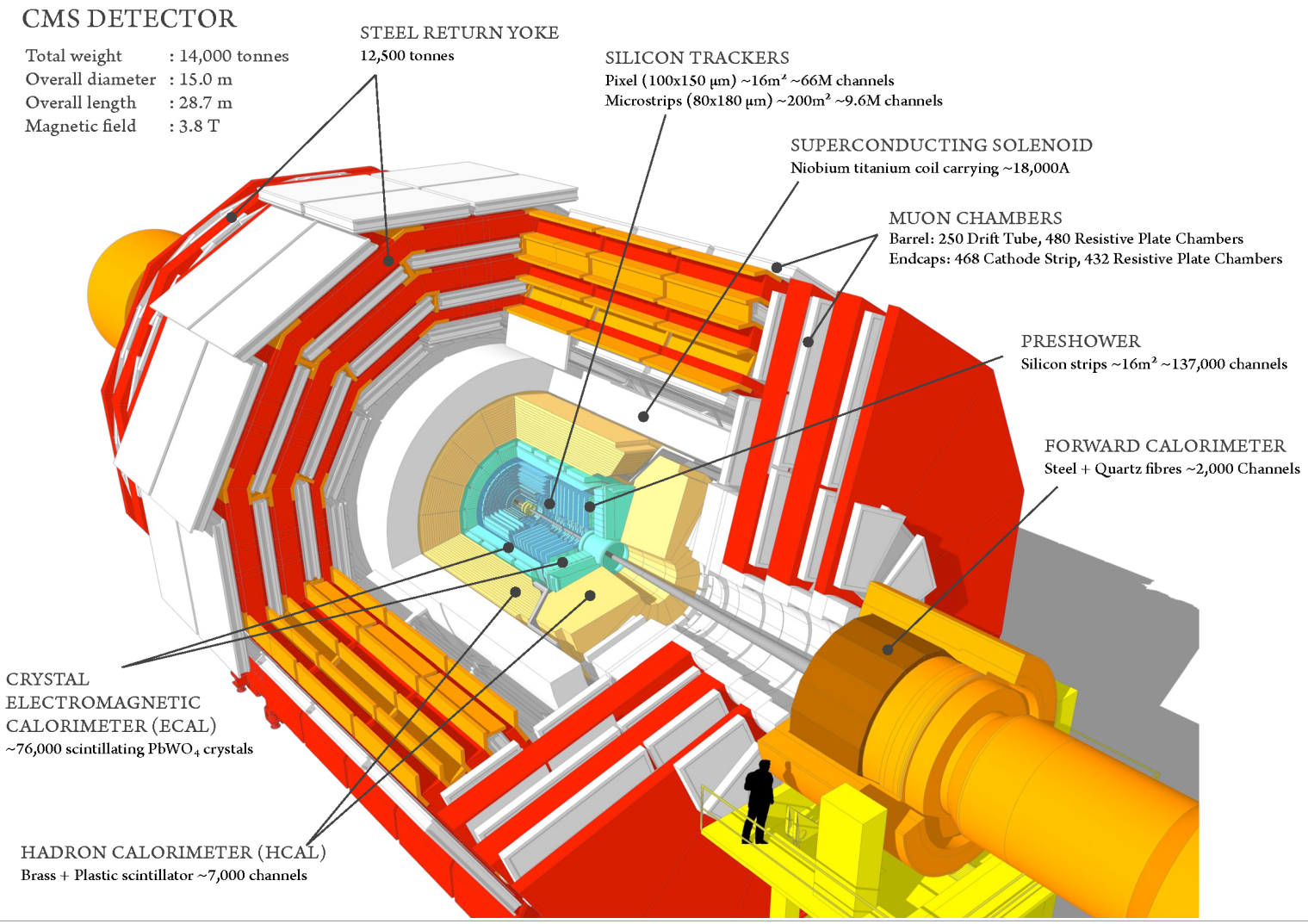
# Physics use case: Search for Dark Matter



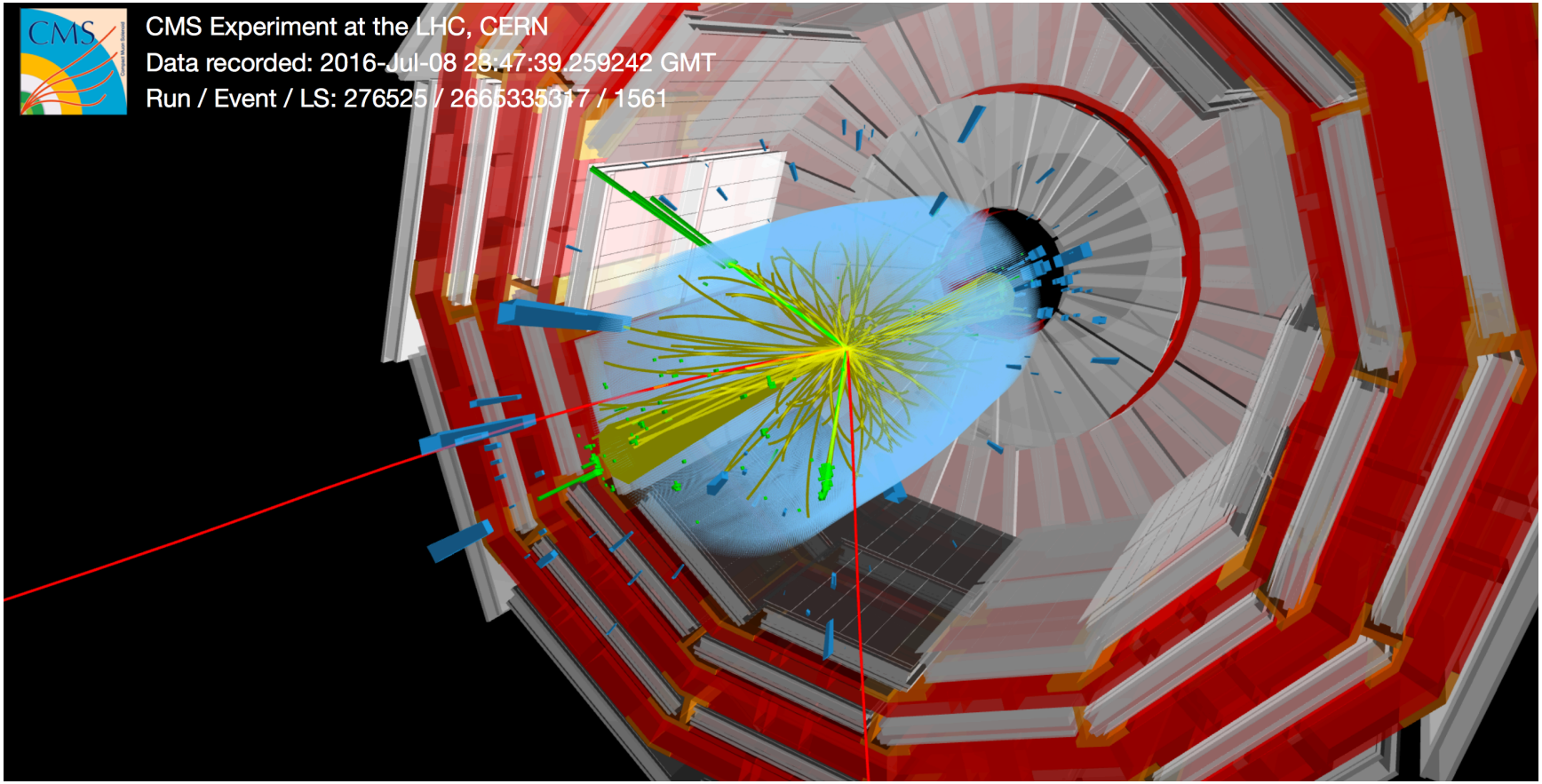
# LHC (Large Hadron Collider) at CERN



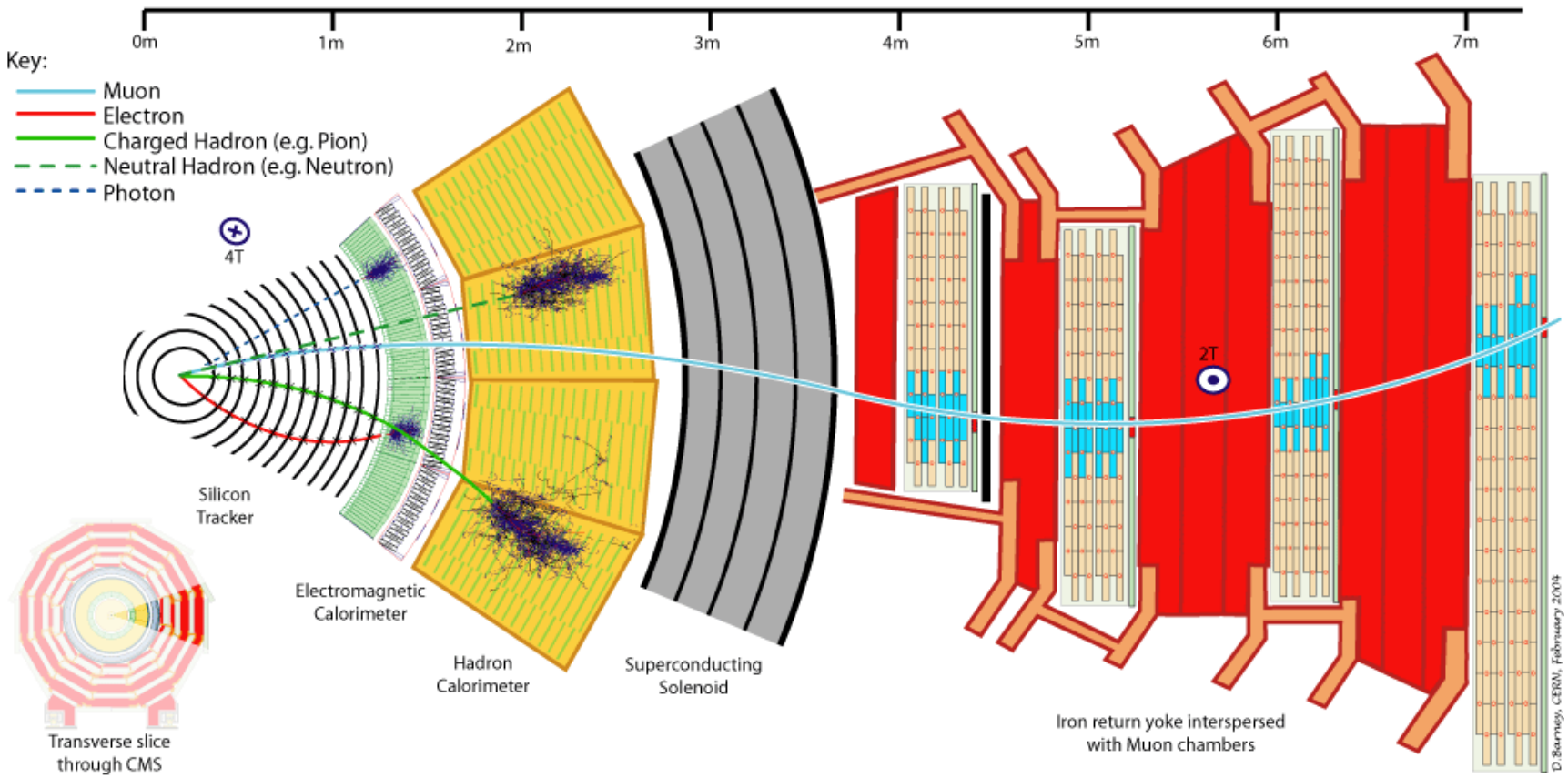
# CMS (Compact Muon Solenoid) detector at LHC



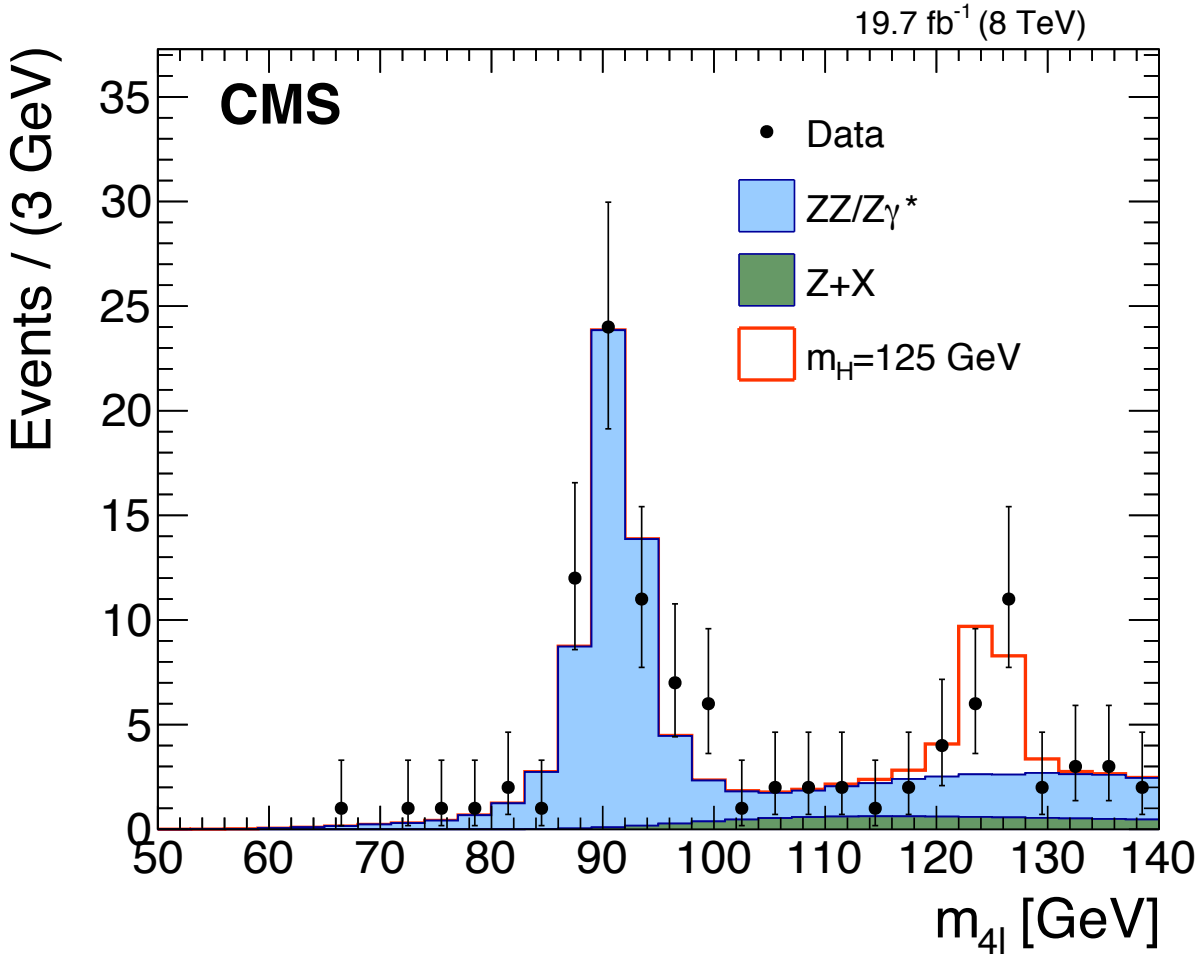
# Particle collision example



# Different Particles as they get detected …



# Searching for new particles



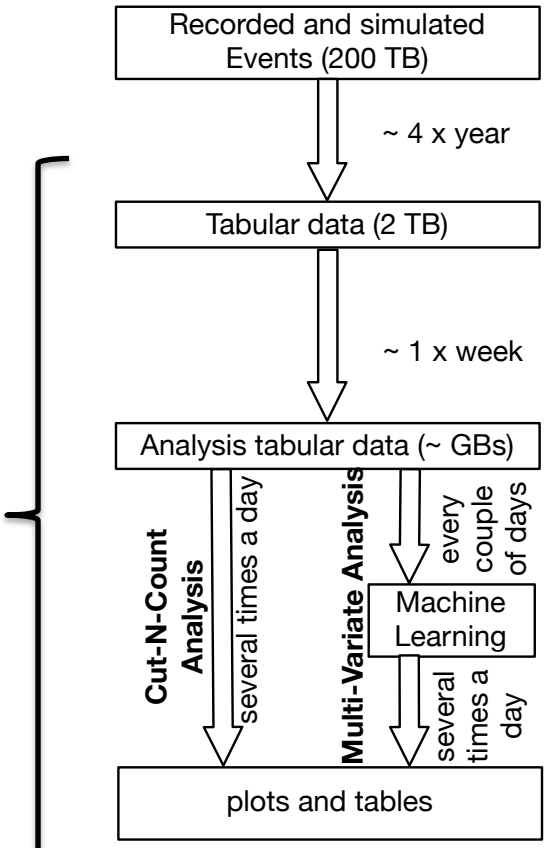
# Current solution

Data Organization:

- Data represented as rows that describe physics objects (particles) and the event in which they were seen
- 400kB/event and 5 x $10^8$ simulated events for backgrounds and signals

Data Processing:

- Event-based processing, sequential file-based solution
- Batch processing on distributed computing farms
- 28,000 CPU hours to generate 2 TB tabular data, ~1 day of processing to generate GBs of analysis ready tabular data, 5 – 30 minutes to run end-user analysis



# Why Spark may be an attractive option

- In-memory large-scale distributed processing
  - Resilient distributed datasets (RDDs)
- Able to use parallel and distributed file system
- Implement the algorithm using high level description in Scala, without worrying about task and data distribution
  - Data partitioning and task parallelism
- Good for repeated analysis performed on the same large data sets

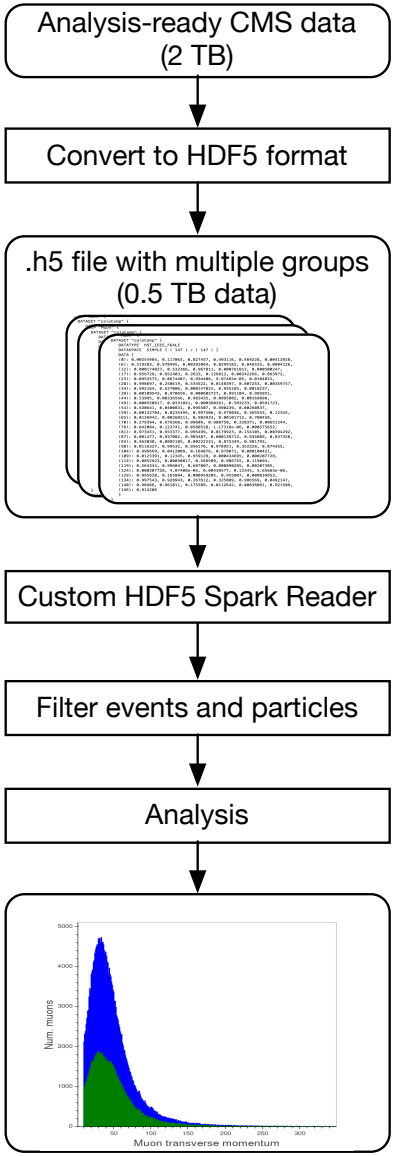
# Our approach

Data organization:

- We convert the data to column oriented format in HDF5.
- HDF5 is a well-known format for the HPC systems; it also allows us to use non-big data technologies to process these files.
- Read HDF5 files into multiple Spark DataFrames, one Spark DataFrame per particle type

Data processing

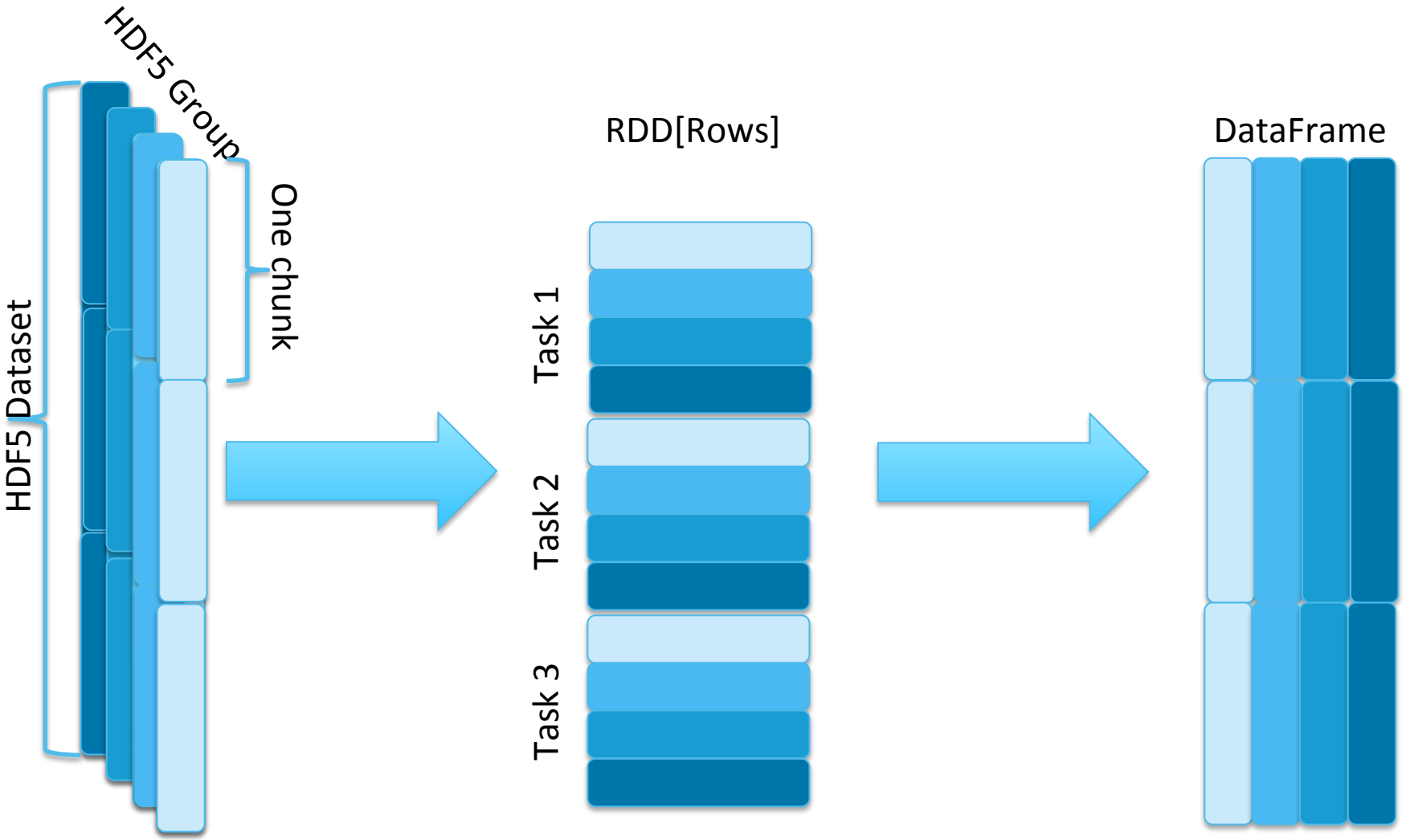
- Define filtering operations on a DataFrame as a whole instead of an event.
- Data is loaded once in memory and processed several times.
- Make plots, repeat as needed.



# Strategies to convert data

- The conversion from CMS tabular format follows a straightforward pattern to represent each particle in an event as an HDF5 group, and each property of the particle as an HDF5 dataset within the group.
- Number of HDF5 groups and datasets
  - One HDF5 group with compound datasets, one compound type per particle type
  - Multiple HDF5 groups per particle type, and multiple 1D datasets per particle property with in a group
  - Multiple HDF5 groups per particle type, and multiple nD datasets per particle properties grouped based on same data type
- Column-oriented data for faster access
- Number of HDF5 files
  - One HDF5 file for each original file
  - One HDF5 file for a group of similar simulation types
  - One HDF5 file per particle type
  - One HDF5 file for all the event data

# Custom HDF5 reader in action



# Encoding problem in Spark

- Use Apache Spark 2.0 DataFrame API in Scala
- Operating on multiple DataFrames, define operations for the DataFrame as a whole and often join a couple of DataFrames
- Use `filter` transformation defined by computation on multiple columns
  - Implement the functionality using User-defined Functions (`UDF`)
  - `UDF` is a feature of Spark SQL to define new column based functions
  - Spark doesn't attempt to optimize `UDF`
- Use SQL `select` queries to join columns from different DataFrames
- Use `groupBy` and aggregations (`count, sum`) to apply on the grouped data

# DataFrames Organization

Event Info

| Event | Lumi | Weight |
|---|---|---|
| 1 | 3245 | 0.5 |
| 2 | 3444 | 0.65 |

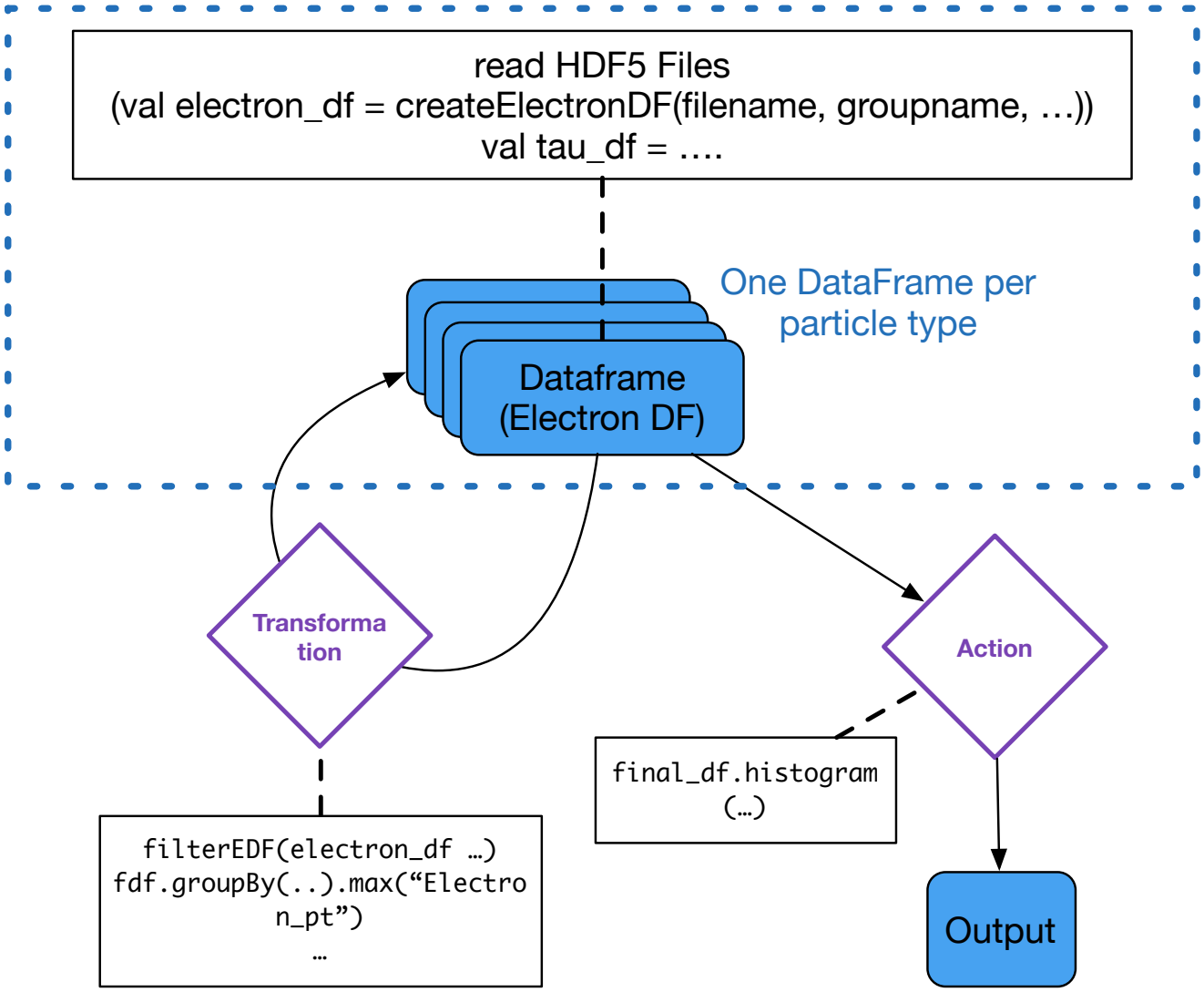
Particle A

| Event | pt | eta |
|---|---|---|
| 1 | 0.034 | 10.5 |
| 1 | 0.045 | 7.8 |
| 1 | -0.92 | 7.6 |
| 2 | 1.0 | 10.9 |
| 2 | 0.54 | 11.0 |

Particle B

| Event | phi | mass |
|---|---|---|
| 1 | 4.5 | 89 |
| 1 | 6.5 | 89 |
| 2 | 4.3 | 89 |
| 2 | 5.7 | 89 |

# Flow of operations and data in Spark



# Examples we implemented

1. Calculate sum of “weights” of all the simulated events
2. Histogram of transverse potentials of jets
    1. Read in the jets information into a DataFrame
    2. Look at each jet independently to see if it passes the given filter
    3. Group by event and keep the jet with the highest transverse potential
    4. Histogram of transverse potentials

# Test setup

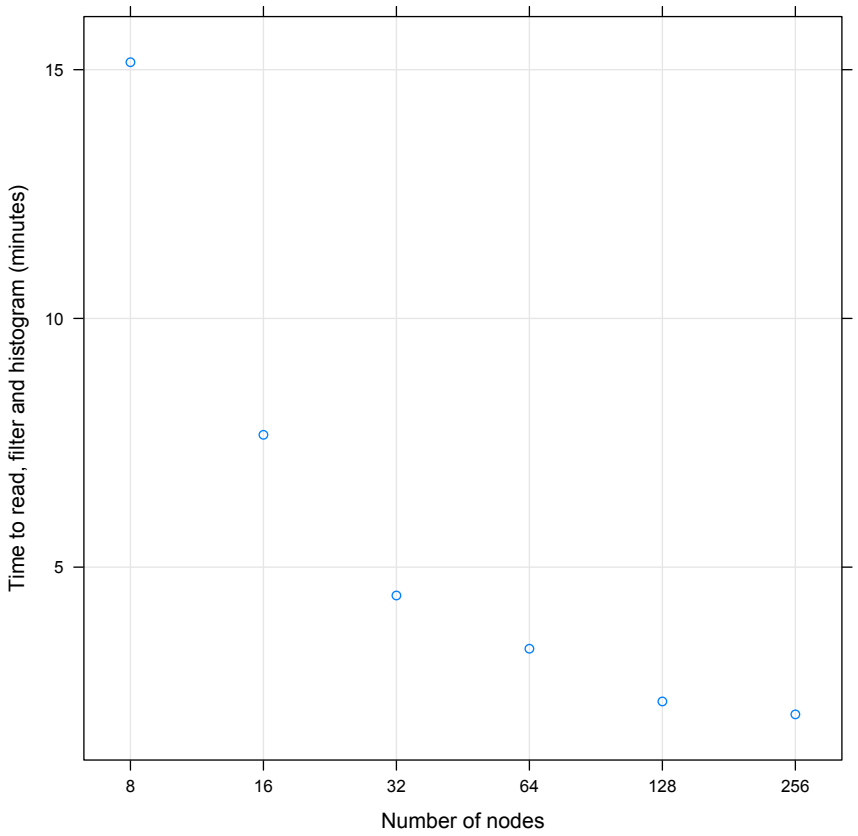
Edison at NERSC is used in the initial development and testing. It is a Cray XC30, with a peak performance of 2.57 petaflops/sec, 133,824 compute cores, 357 terabytes of memory, and 7.56 petabytes of disk.

- We used Apache Spark 2.0, it is available on both Edison and Cori
- We ran the tests using 8, 16, 32, 64, 128 and 256 nodes
- Input data consists of 360 million events (0.5 TB in memory)

# Test Results observations

1. Good scaling for the sum of weights test, i.e. by increasing the number of events on a fixed number of nodes, time increases linearly.
2. For test 2, with complex operation on data for the 360 million events, we observe good scaling behavior for the initial read and DataFrame creation, however, the operations afterward took < 1 sec to complete.

# Strong Scaling



# Lessons Learned /Experience

The goal of our exploratory study is to shorten *time to physics* using Spark on HPC

- We have observed good scalability and task distribution
- It is hard to tune a Spark system
  - Optimal data partitioning, number of executor cores, executor memory, etc is challenging
  - Difficult to isolate slow performing stages due to lazy evaluation
- Availability of pySpark and SparkR high level APIs are appealing to the HEP user community and good for rapid prototyping
- Encoding this multi-stage analysis workflow using Scala best practices is challenging along with the optimal use of Spark DataFrame features
- Documentation, and error reporting needs to be better

# Contributions

- A cross-cutting project bringing HEP, HPC and big data technology together
- A prototype of an HDF5 Spark reader for reading several 1D HDF5 datasets within a HDF5 group into a single Spark DataFrame with user specified partitioning size across multiple files
- Provided a tabular data representation and ability to perform analyses in distributed environment beyond batch processing.
  - eliminate the need of intermediate file storage

# Future work

- Improve the Spark HDF5 reader interface
- Performance of different DataFrame organizations
- Data partitioning
- Optimize the workflow to filter the data
  - Design of SQL queries and UDF
- Scale up to greater than TB data set
- Cori will be used for next set of results

# Collaborators

Fermilab

- Jim Kowalkowski
- Marc Paterno
- Oliver Gutsche
- Matteo Cremonesi
- Bo Jayatilaka
- Cristina Mantilla

Princeton University

- Jim Pivarski
- Alexey Svyatkovskiy

# References


1. LHC http://home.cern/topics/large-hadron-collider
2. CMS http://cms.web.cern.ch
3. HDF5 https://www.hdfgroup.org
4. Spark at NERSC http://www.nersc.gov/users/data-analytics/data-analytics/spark-distributed-analytic-framework/

# Thank you

ANITA BORG INSTITUTE
GRACE HOPPER CELEBRATION
OF WOMEN IN COMPUTING

## Feedback?

**Rate and review the session on our mobile app**

Download at http://bit.ly/ghc16app
or search GHC 16 in the app store

---

ANITA BORG INSTITUTE
WOMEN TRANSFORMING TECHNOLOGY

acm
Association for Computing Machinery