Atom Smashing using Machine Learning at CERN

Siddha Ganju

Master in Computational Data Science

Carnegie Mellon University

Strata+Hadoop WORLD

March 28-31, San Jose, CA

About Me

@SiddhaGanju

http://sidgan.me/

sganju1[at]cs[dot]cmu[dot]edu

Carnegie Mellon University
Master of Computational Data Science

Evaluation of Apache Spark as Analytics Framework for CERN's Big Data Analytics

- Thanks to my mentors
  - Valentin Kuznetsov, Cornell University
  - Tony Wildish, Princeton University
  - Manuel Martin Marquez, CERN
  - Antonio Romero Marin, CERN

Project Scope

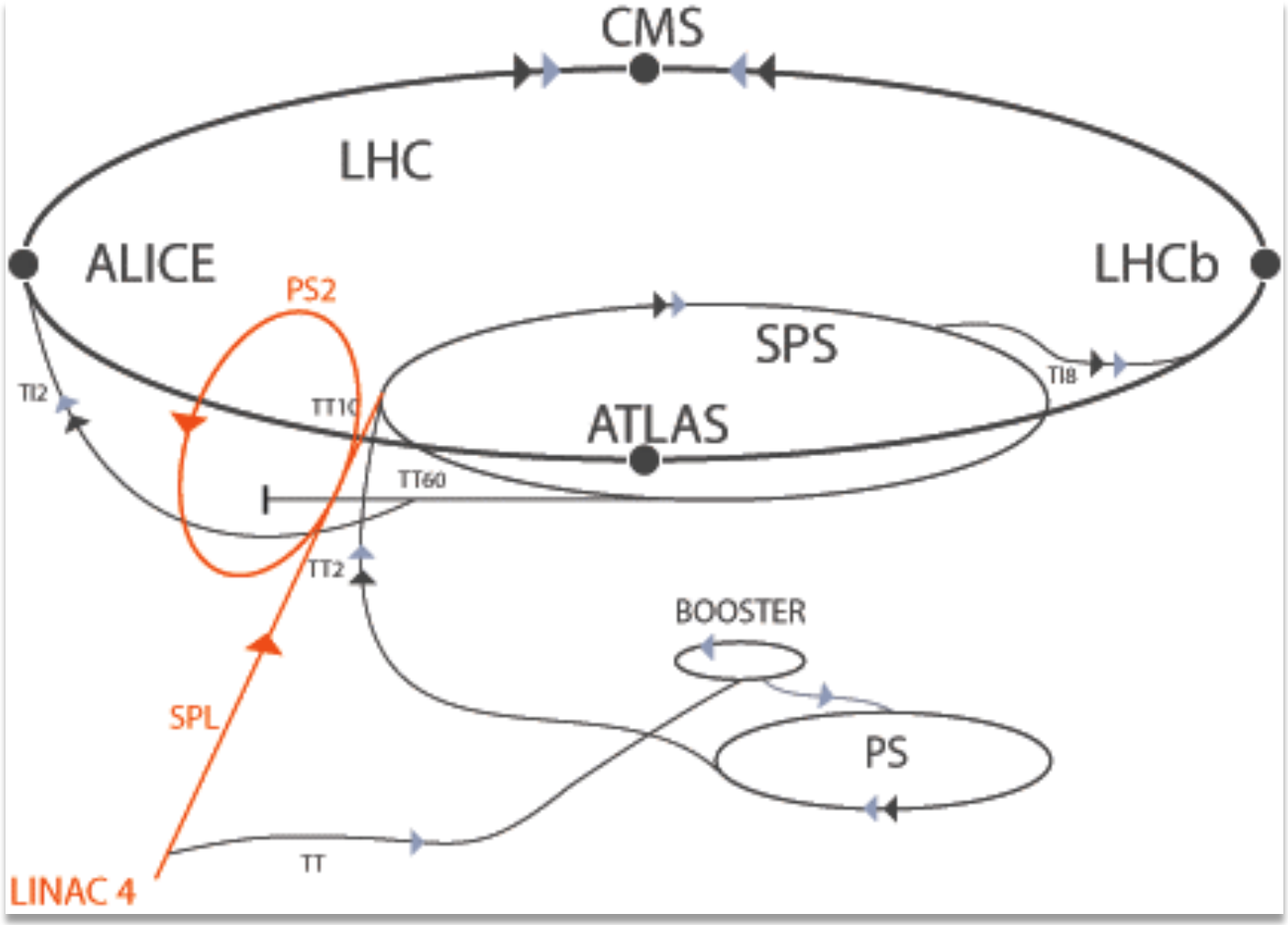
- Understanding CMS BIG data
  - Static + Streaming
- Exploring Apache Spark
  - Potential framework for big data analysis
  - Predict popular datasets in near real time
  - Facilitate Dynamic Data Placement
  - Efficient resource utilization

Outline

- CERN
- Understanding CMS Data
- Evaluation of Apache Spark
- Results and Conclusion

European Organization for Nuclear Research

LHC@CERN



Source: Manuel Martin Marquez, CERN

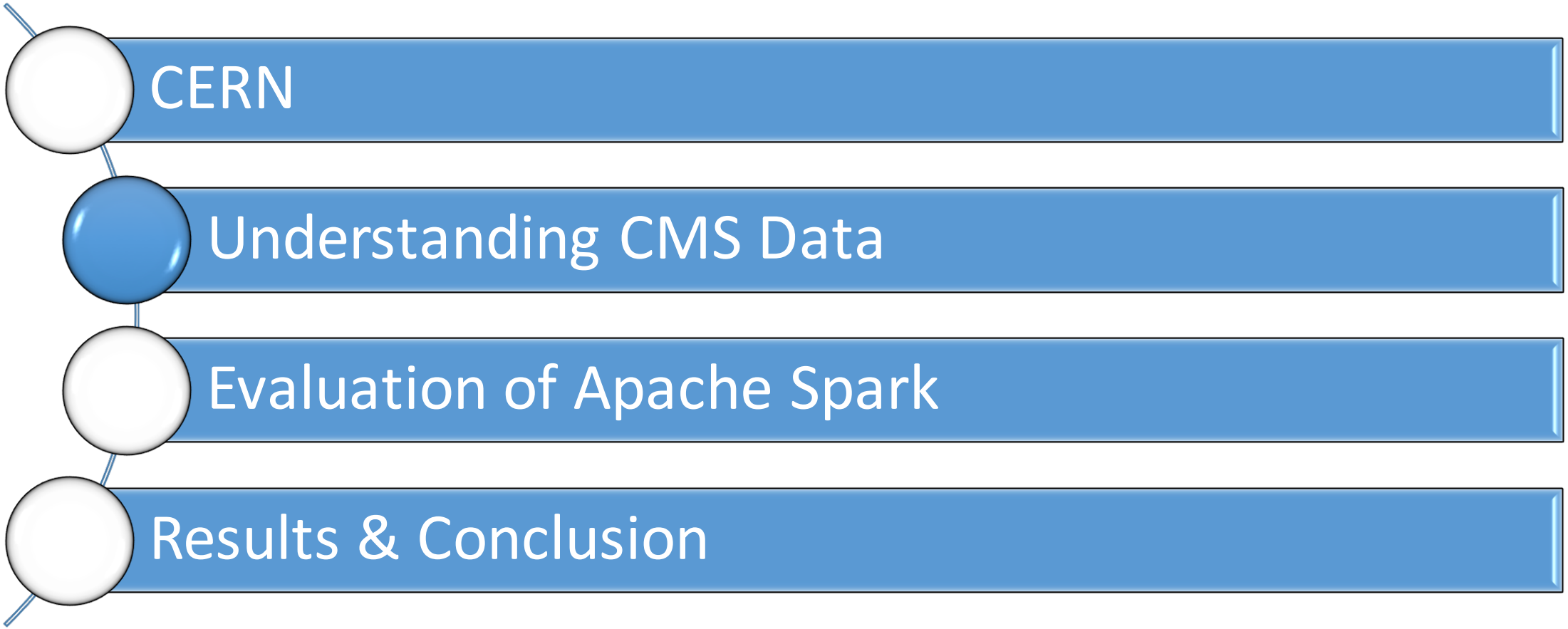
European Organization for Nuclear Research

LHC@CERN

Atom Smashing at CERN

- Compact Muon Solenoid(CMS) hunts for Higgs boson particles and clues to the nature of dark matter
- Beams of protons collide with energy of 13 TeV
  - Visualized in 3 dimensions
  - Digital summary of collisions
- Peta Byte (PB) -order data obtained, post filtering

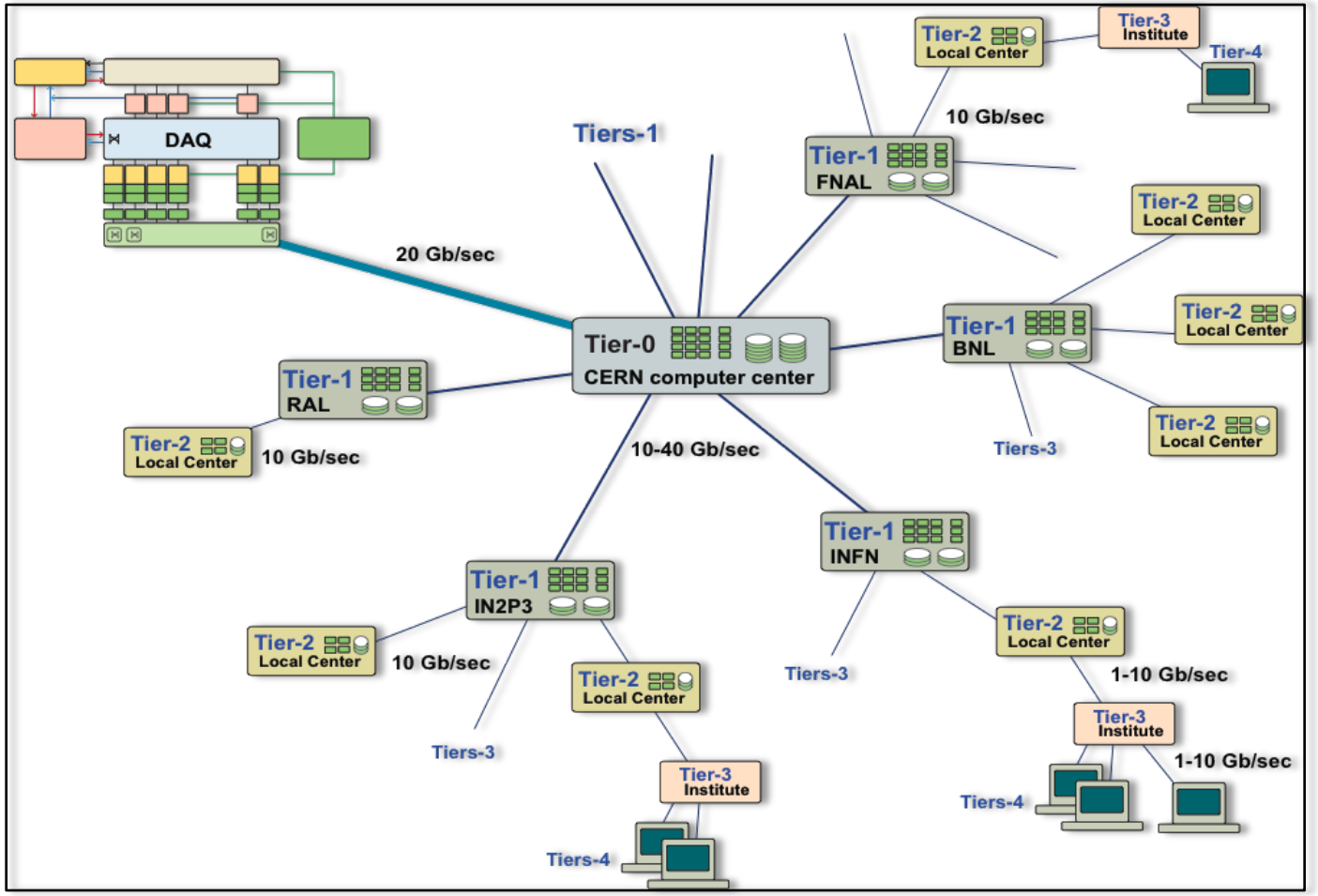
Outline


CERN
Understanding CMS Data
Evaluation of Apache Spark
Results & Conclusion

CERN Computing Grid

CERN

- Recording
- Reconstruction
- Distribution



Source: Manuel Martin Marquez, CERN

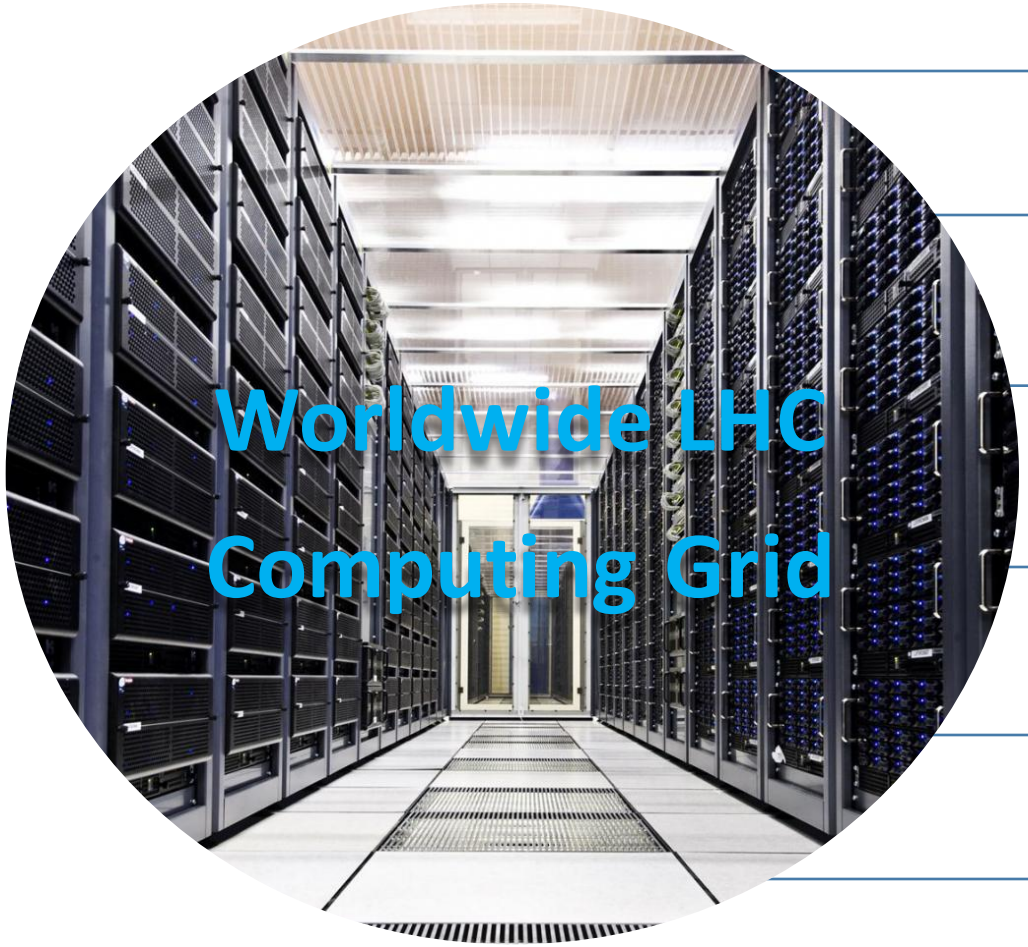
Physics Workflow and Data Analytics

- Physics data
  - Different streaming data scenario
  - The new data is of course interesting!
  - Old data is all the more interesting!!
    - Re-processed old data with signals and background check influences the importance of new data

Big Data Approach

Volume	Velocity	Variety	Veracity
• Number of collisions in the LHC • 15 PB order post filtering	• 300 Mb/s production rate • Quick availability of results • Real time feedback is NOT required	• Irregular data • Different kinds data sets	• Uncertain • Limited data integrity

Data Acquisition – Structured sources



PhEDEx: Physics Experiment Data Export

CMS Data Management System

CMS Workload Management System

PopDB

SiteDB

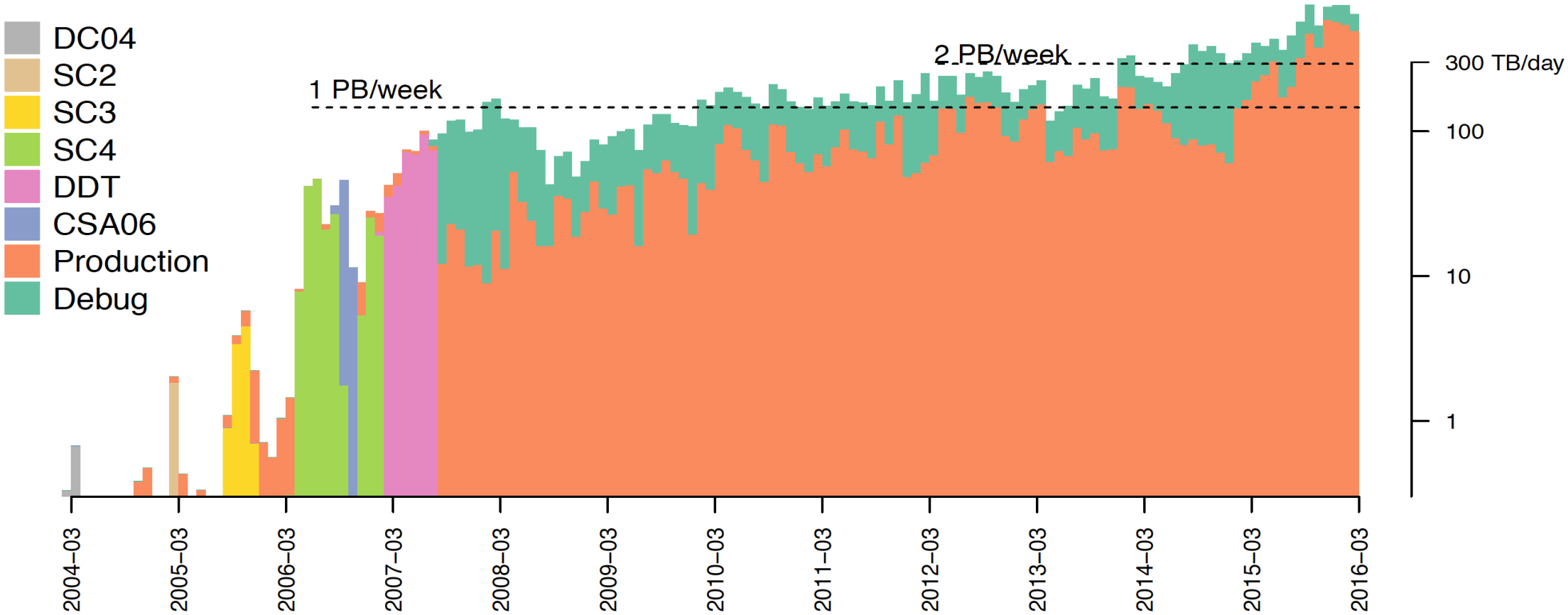
Dashboard

Data Acquisition – Unstructured sources

- CERN Indico
- CERN Zenodo
- CERN Vidyo

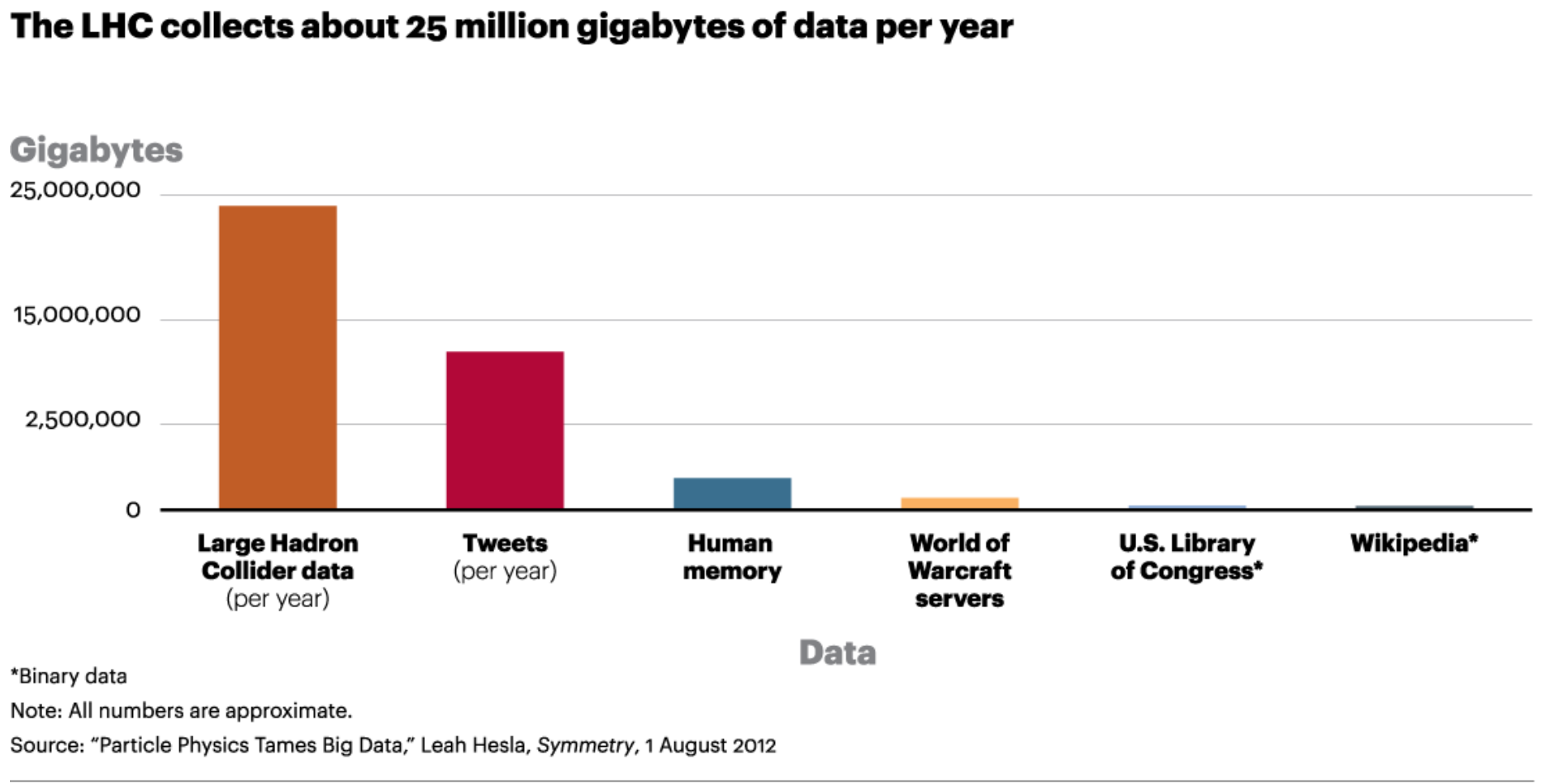
Data is important

Data is really important

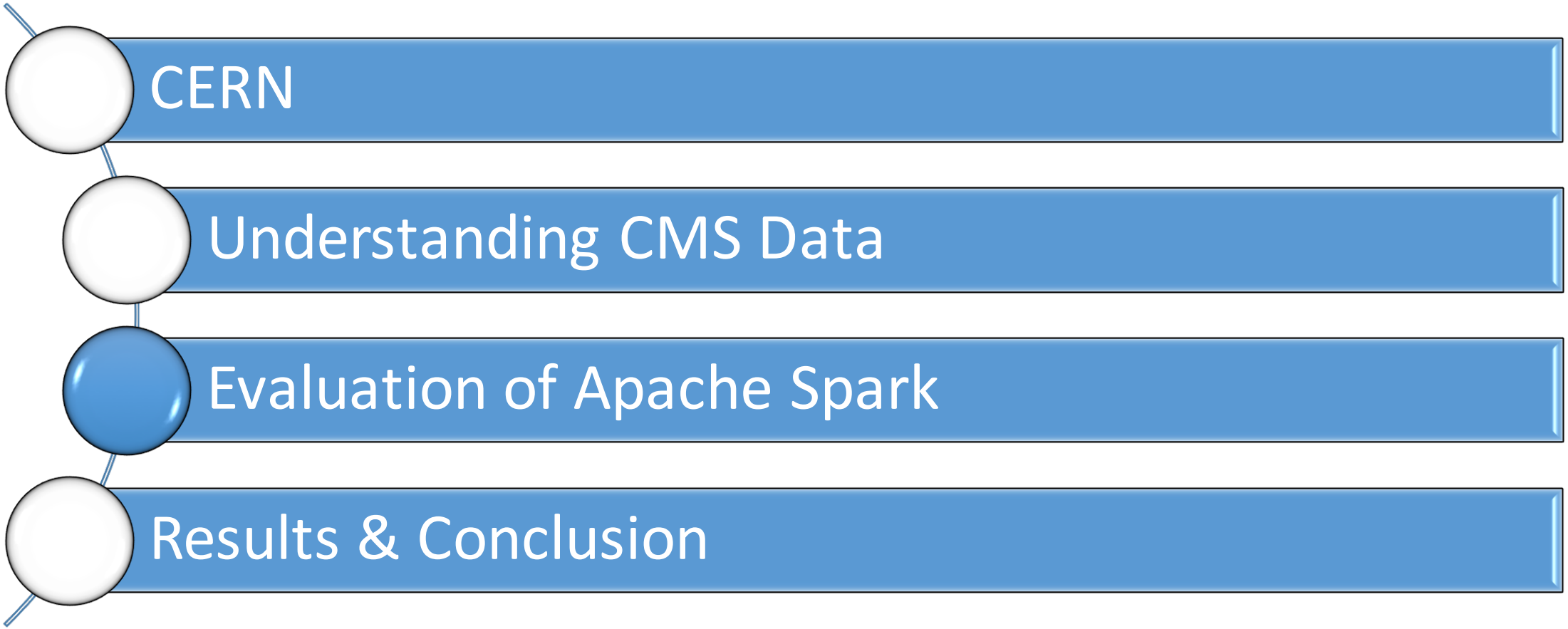


Source: Tony Wildish, Princeton University

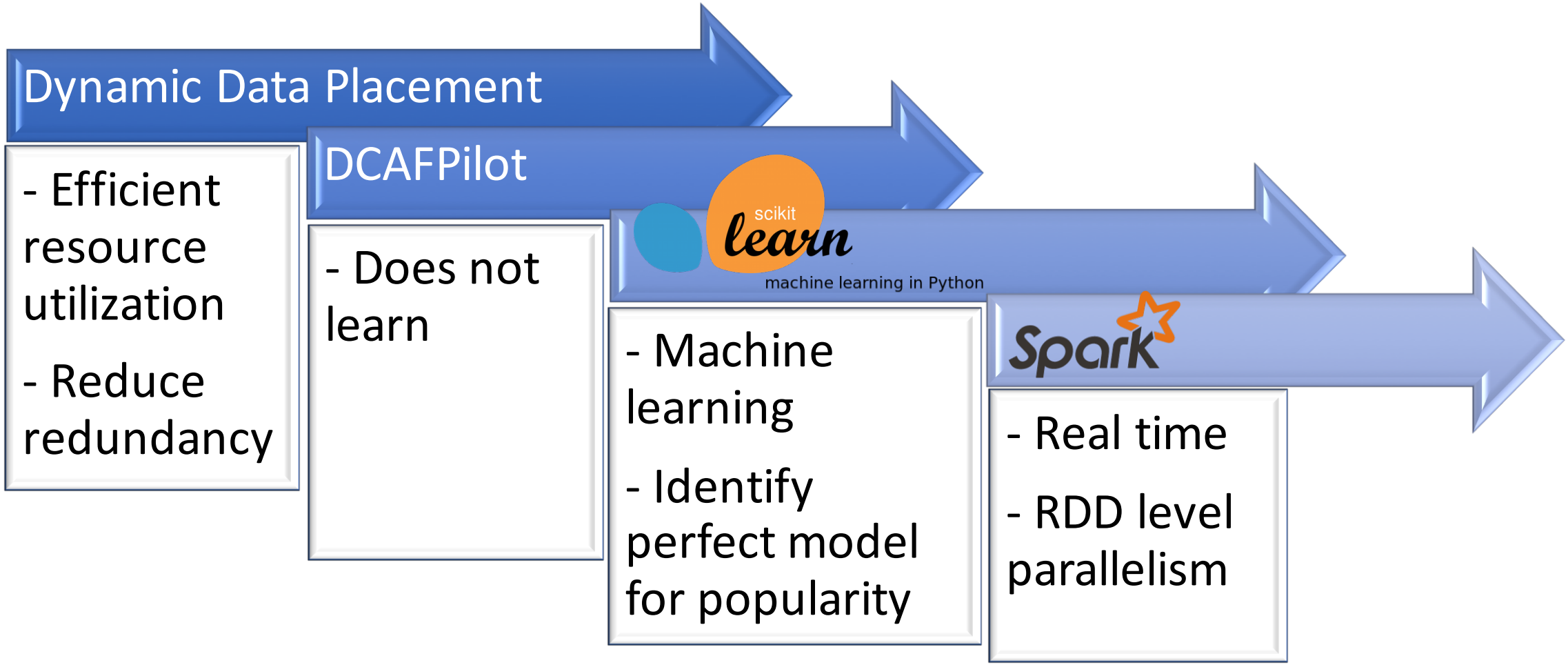
Data is indeed important



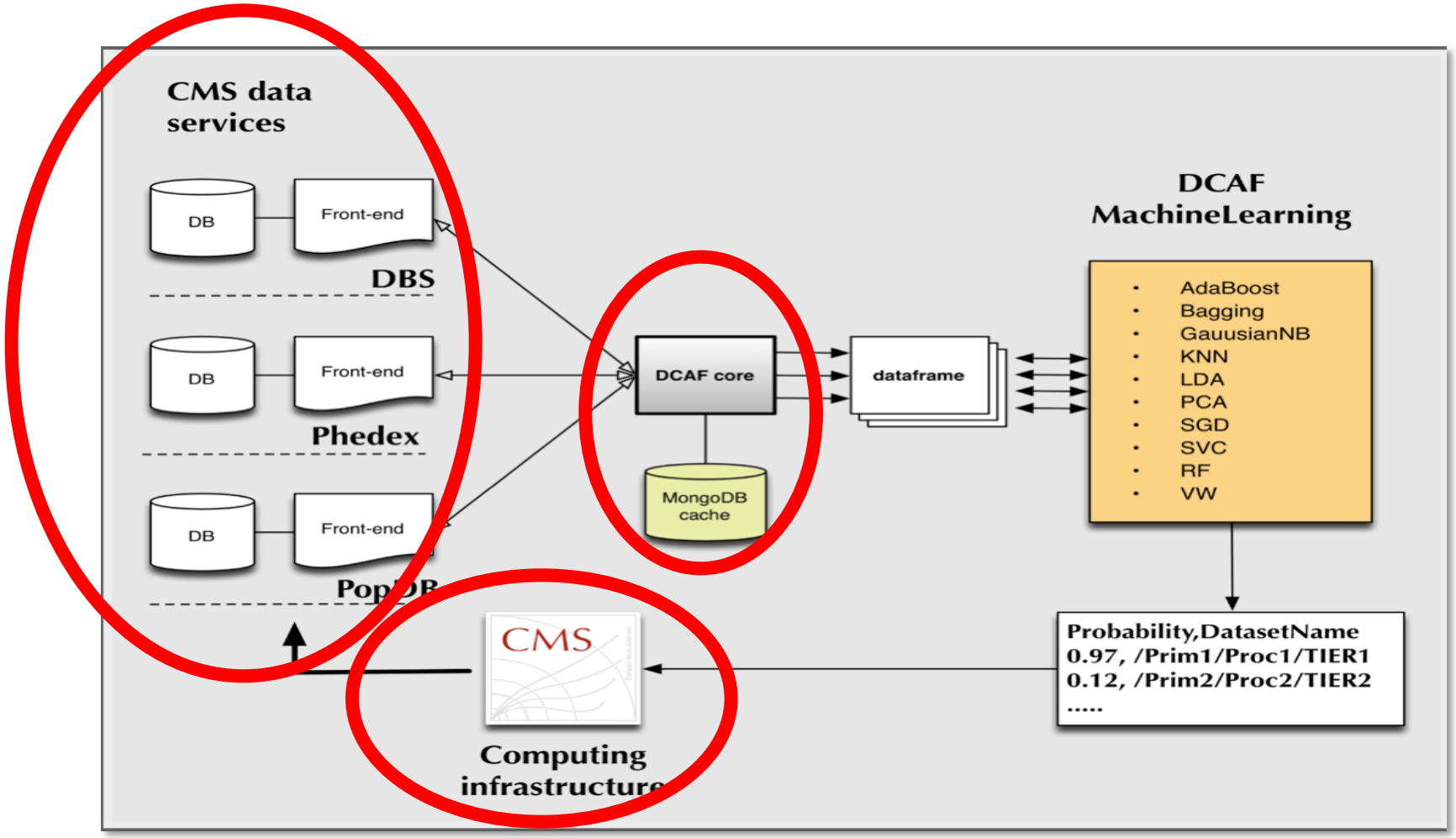
Outline


CERN
Understanding CMS Data
Evaluation of Apache Spark
Results & Conclusion

Timeline


Dynamic Data Placement
- Efficient resource utilization
- Reduce redundancy
DCAFPilot
- Does not learn
scikit
learn
machine learning in Python
- Machine learning
- Identify perfect model for popularity
Spark
- Real time
- RDD level parallelism

Data and Computing Analysis Framework



Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/DCAFPilot_flow.png

1K popular

&

10K unpopular

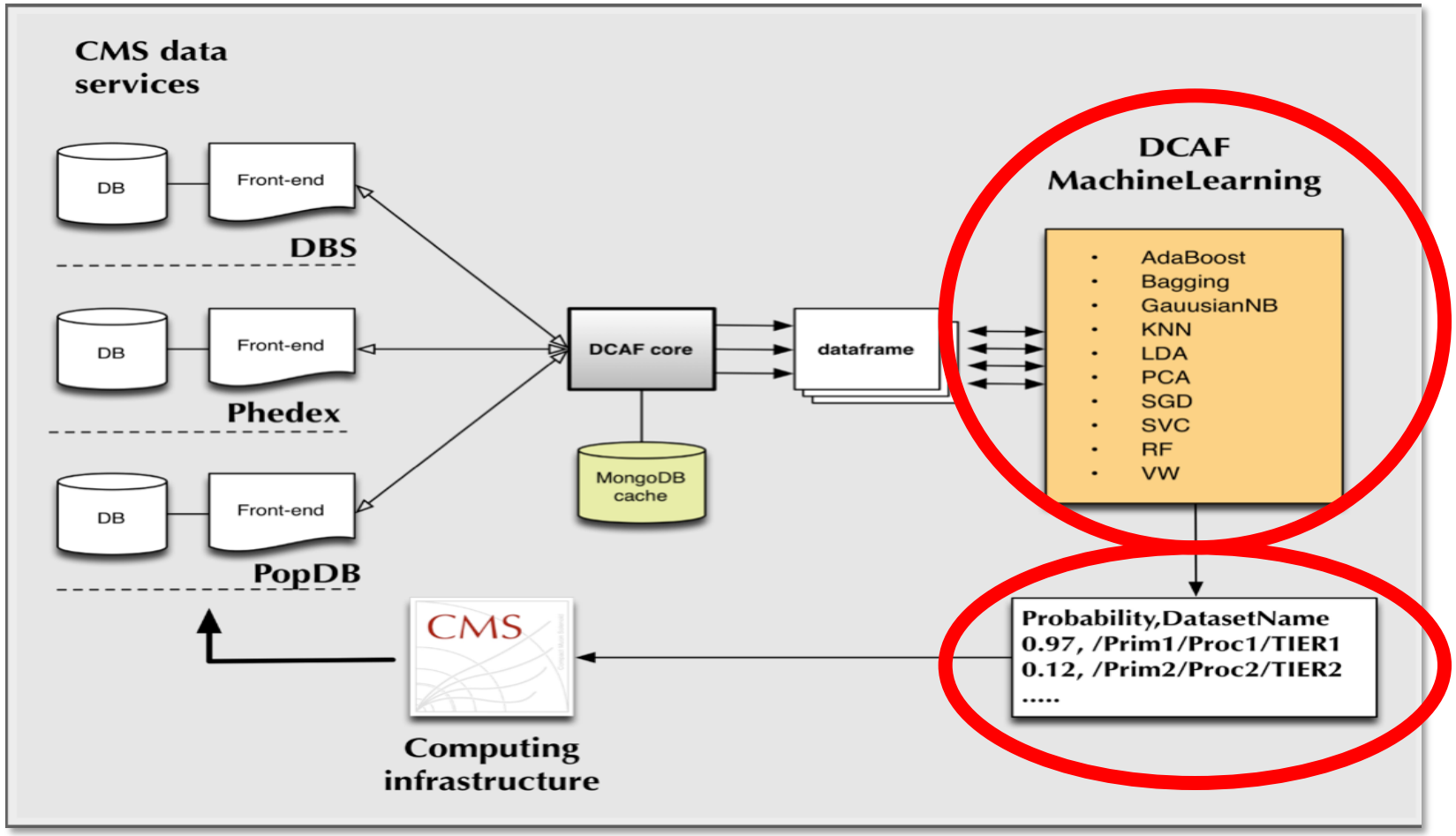
78 features

25 relevant features

Feature Extraction

id	cpu	nlumis	nfiles	nblk
creator	tier	nevt	proc_evts	nusers
wct	size	rnaccess	primds	nsites
totcpu	rtotcpu	rnusers	parent	nrel
naccess	era	dtype	dbs	dataset

Data and Computing Analysis Framework



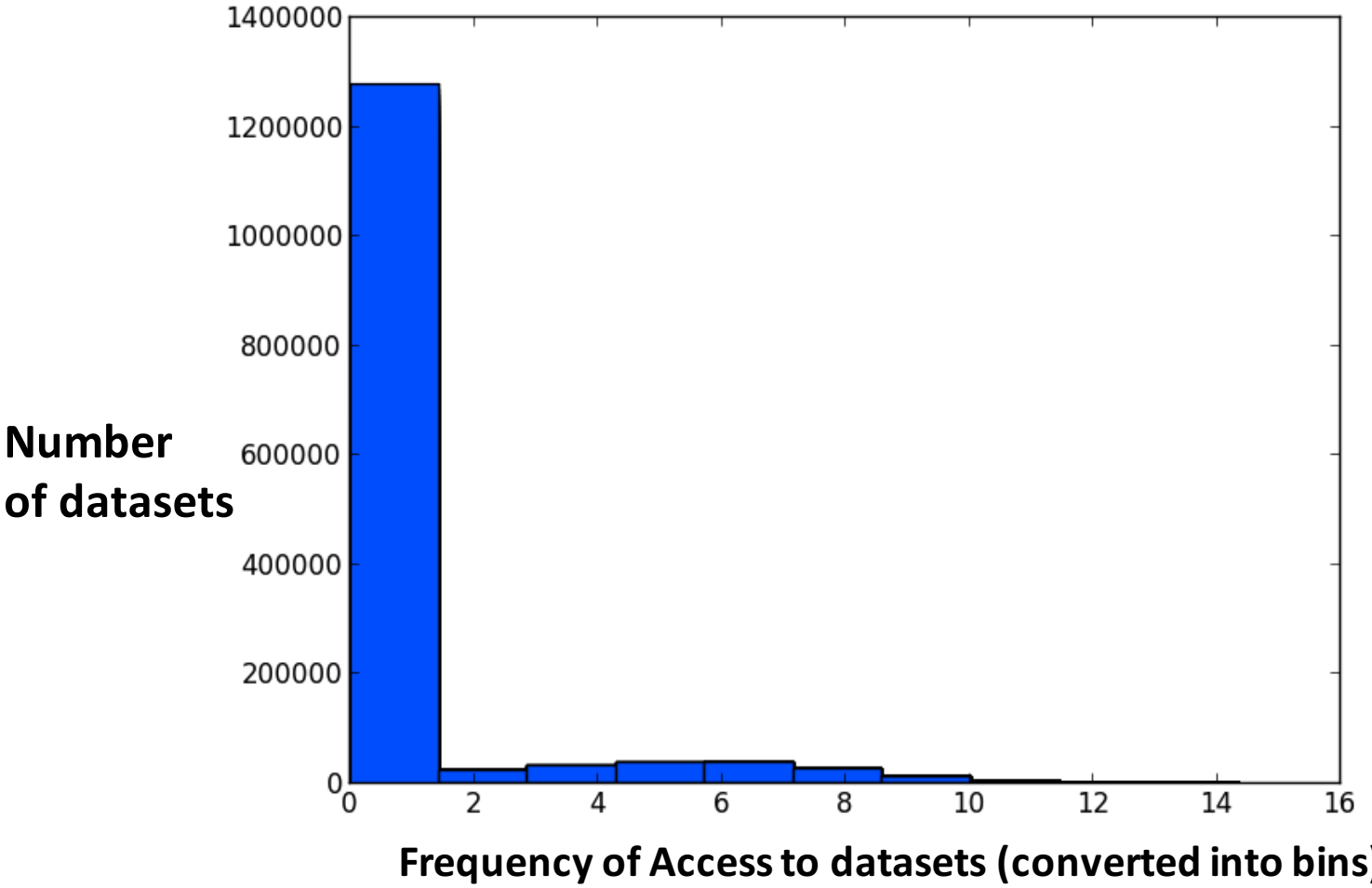
Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/DCAFPilot_flow.png

Transforming to classification problem

- Classification problem
  - Target column is 1(popular) when naccess is greater than 10 and nusers is greater than 5

```
def convert(df):
    threshold_naccess = 10
    threshold_nusers = 5
    return df['naccess'] > threshold_naccess and
df['nusers'] > threshold_nusers
```

Frequency of Accesses



Naccess cut found by plotting the entire 2014 dataset against frequency of access

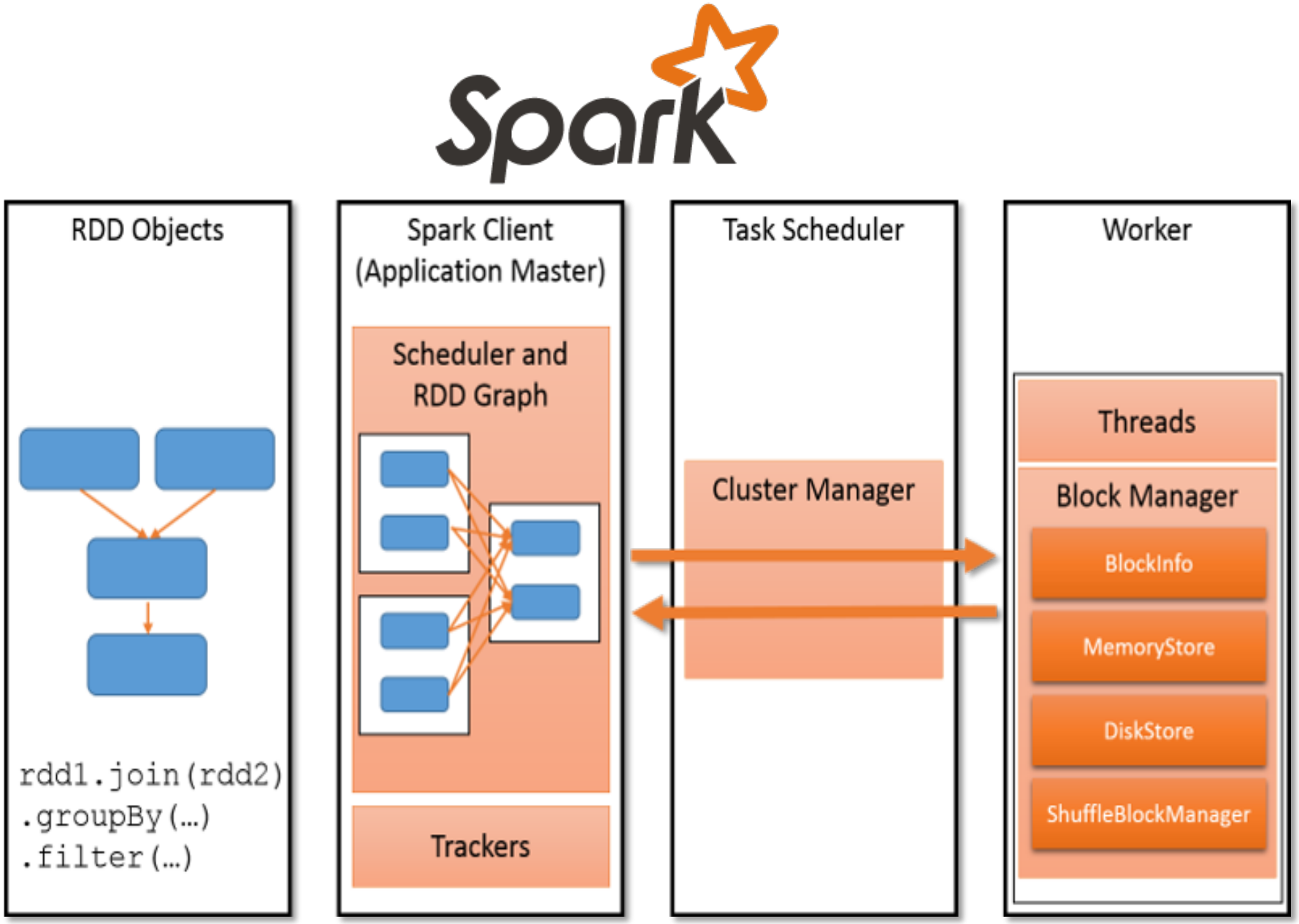
600KB compressed data frames

600KB compressed/file

* 52 files/year

* 3 years/run

~15 PB/year

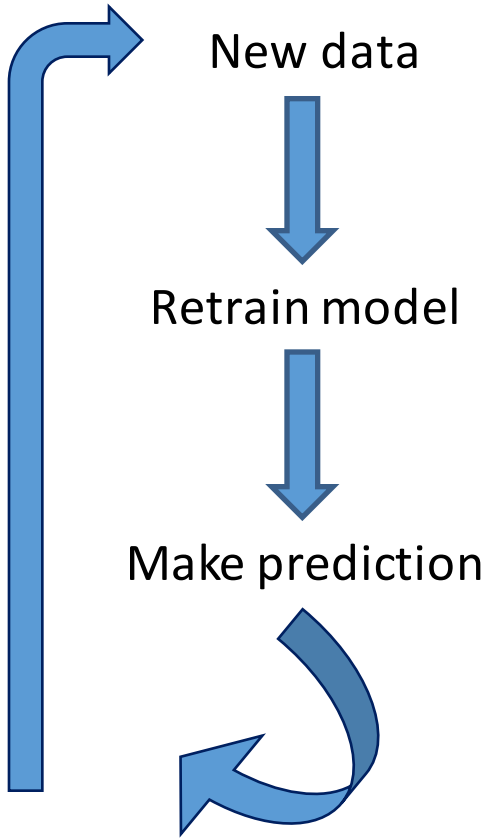


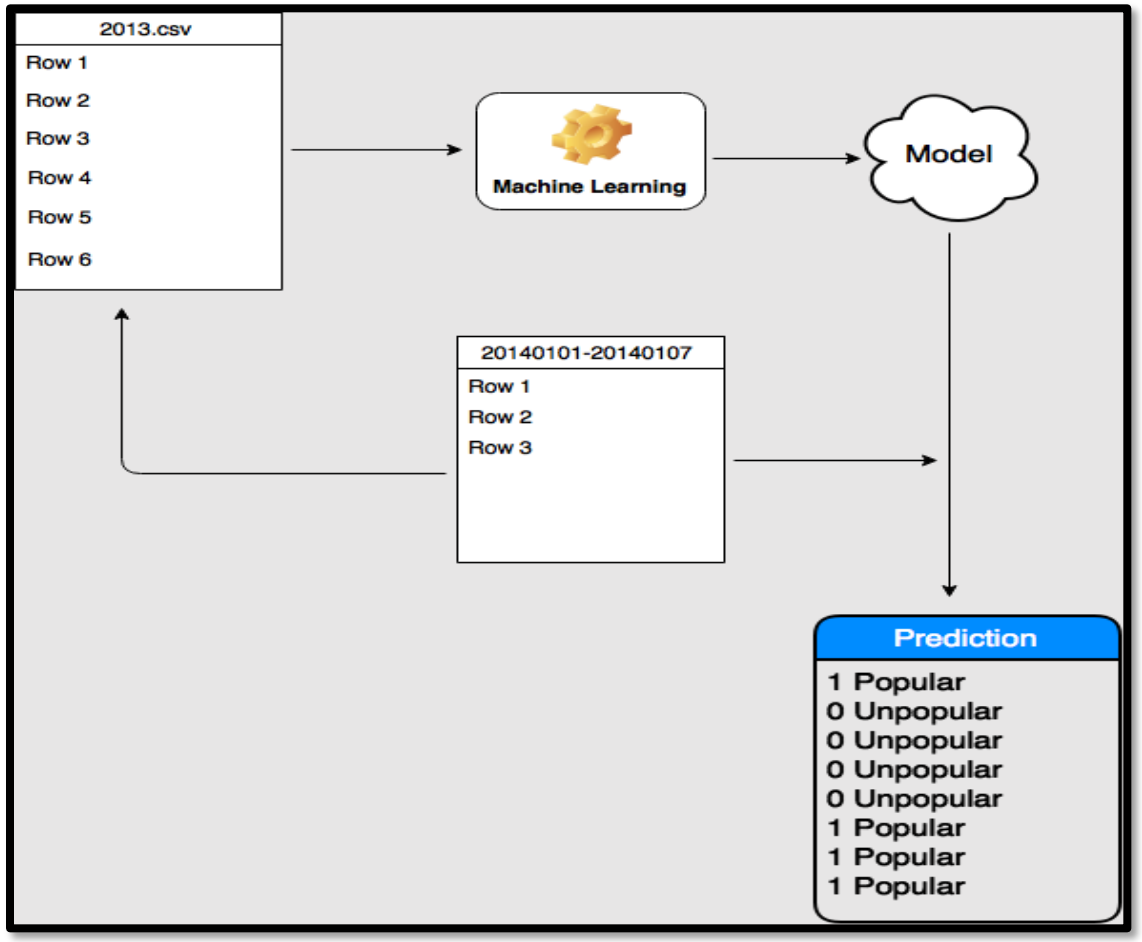
Source: 15-319 / 15-619 Cloud Computing, CMU

Apache Spark analysis

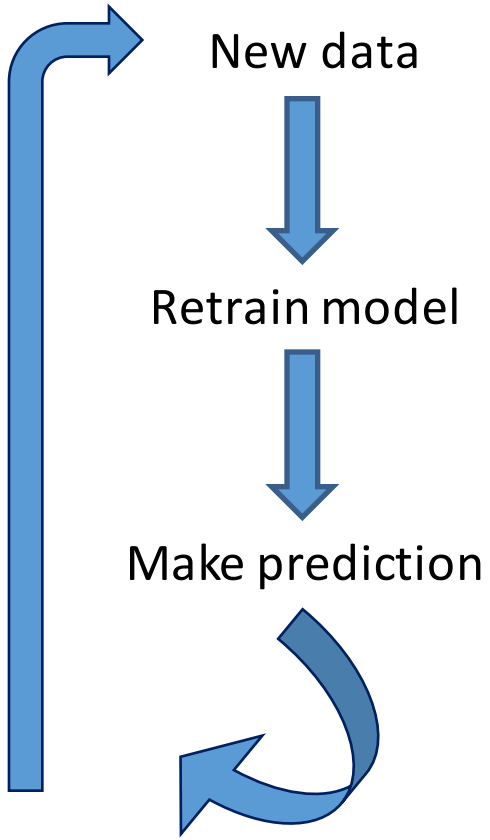
- Fast and compatible with already existing HDFS at CERN
- Run in Hadoop clusters through MESOS or Spark's standalone mode
  - Process data in HDFS
  - Batching of a week's data
  - Processing for new workloads like streaming (live prediction of each week), interactive queries, and machine learning.

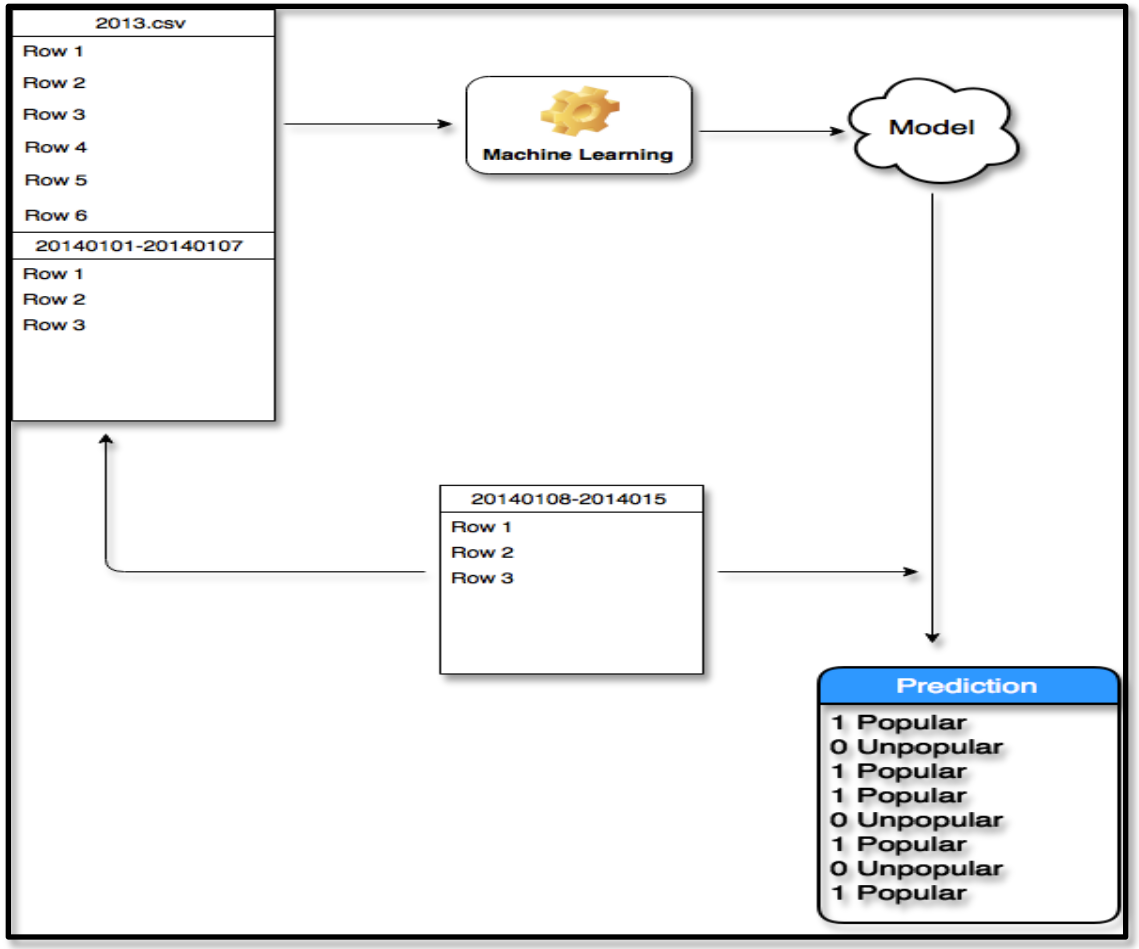
Rolling Forecast



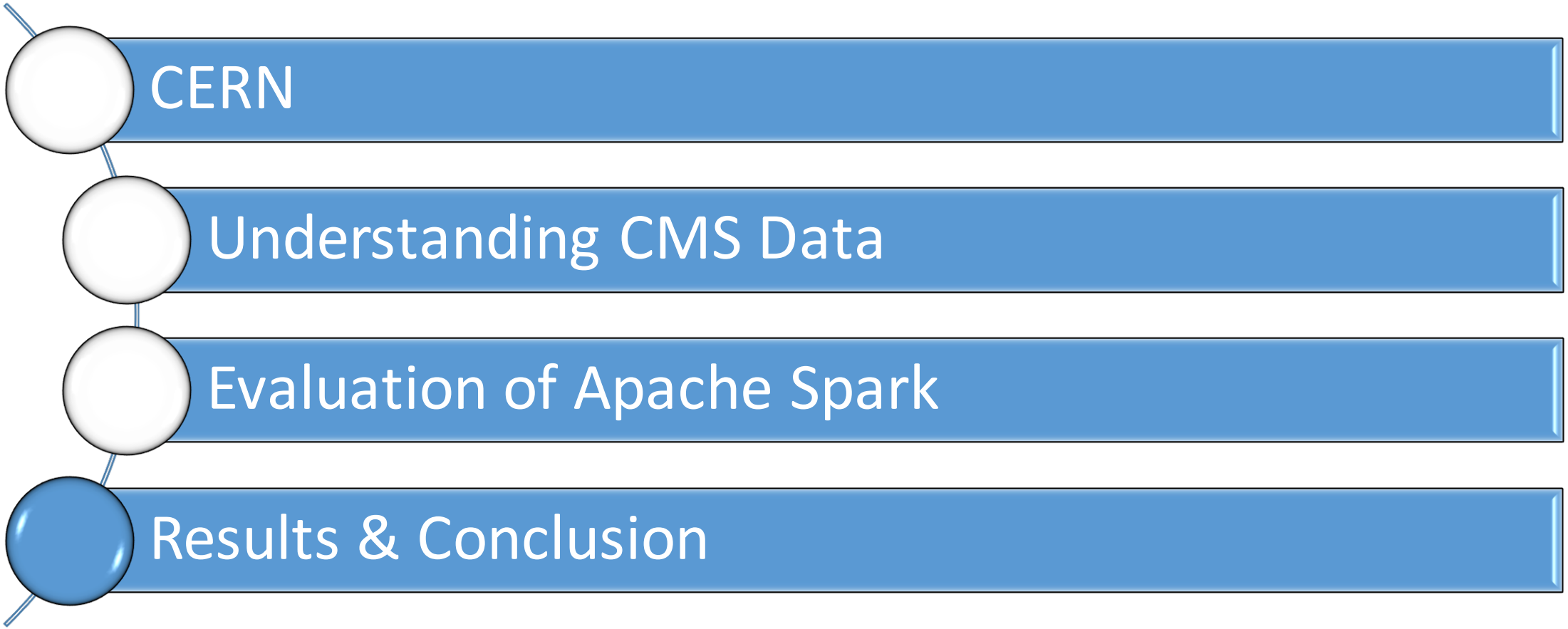


Rolling Forecast





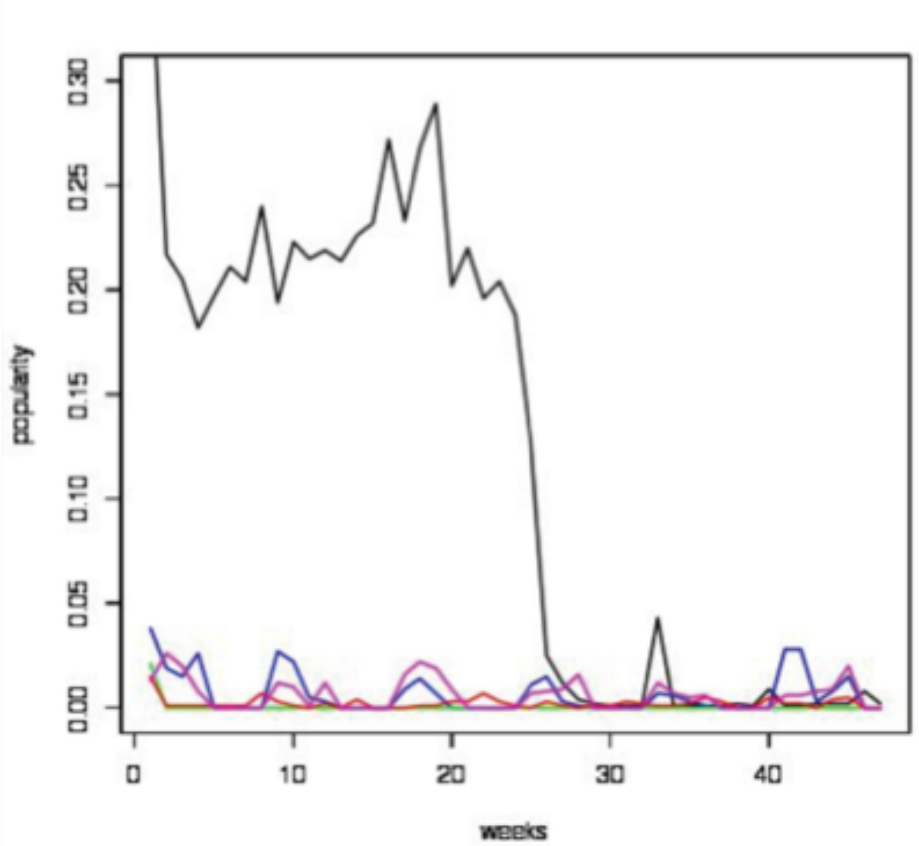
Outline


CERN
Understanding CMS Data
Evaluation of Apache Spark
Results & Conclusion

- Parallel analysis on distributed data
- Decreased total execution time for the individual algorithm or combined ensemble to execute

Input data **Apache Spark**

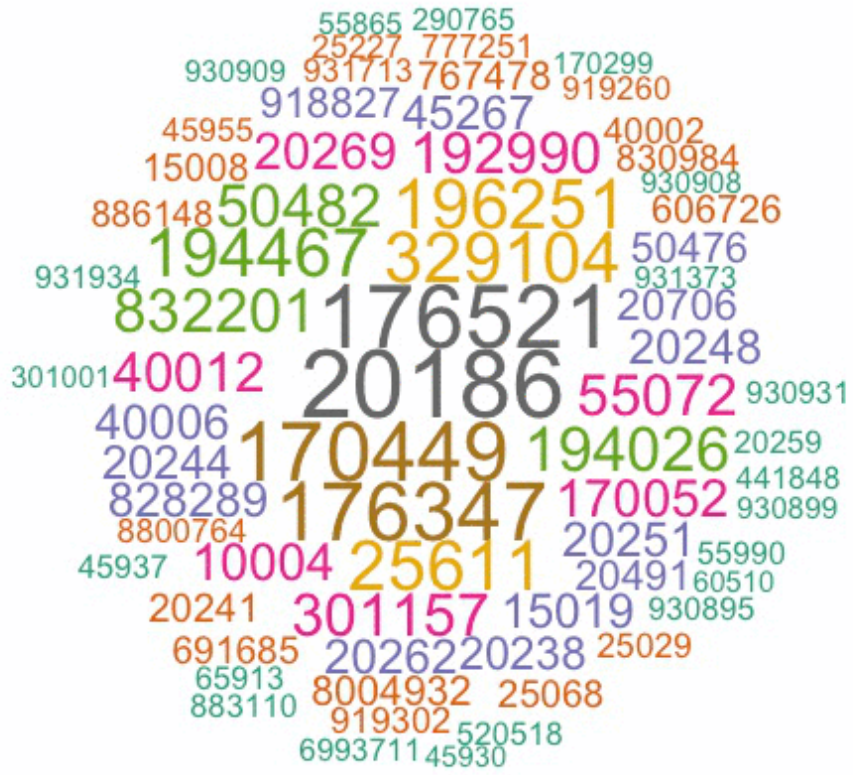
Popularity Metrics



Popularity of Random datasets

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/popularity.jpg

Popularity Metrics - nusers



Total number of users who accessed the dataset

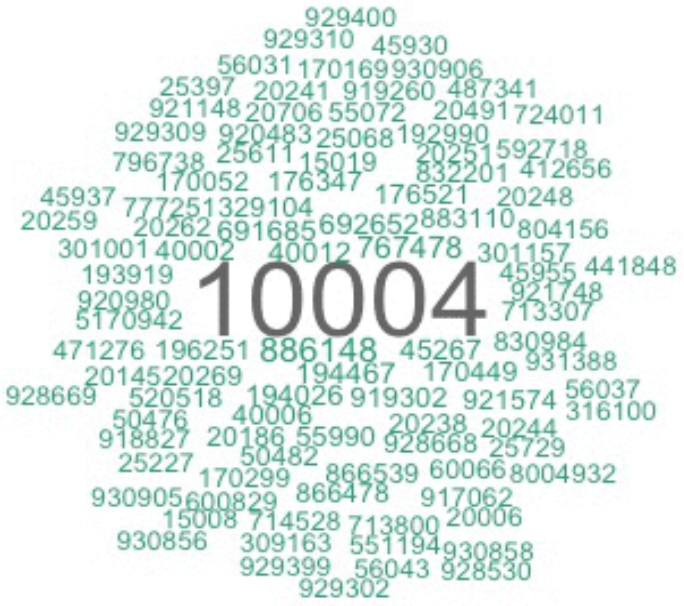
Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/nusers_cloud.gif

Popularity Metrics - totcpu

Total CPU hours spent analyzing a dataset

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/cpu_cloud.gif

Popularity Metrics - naccess



Count of individual accesses to a dataset

Source: https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/DCAFPilot/doc/talks/Pilot1/images/naccess_cloud.gif

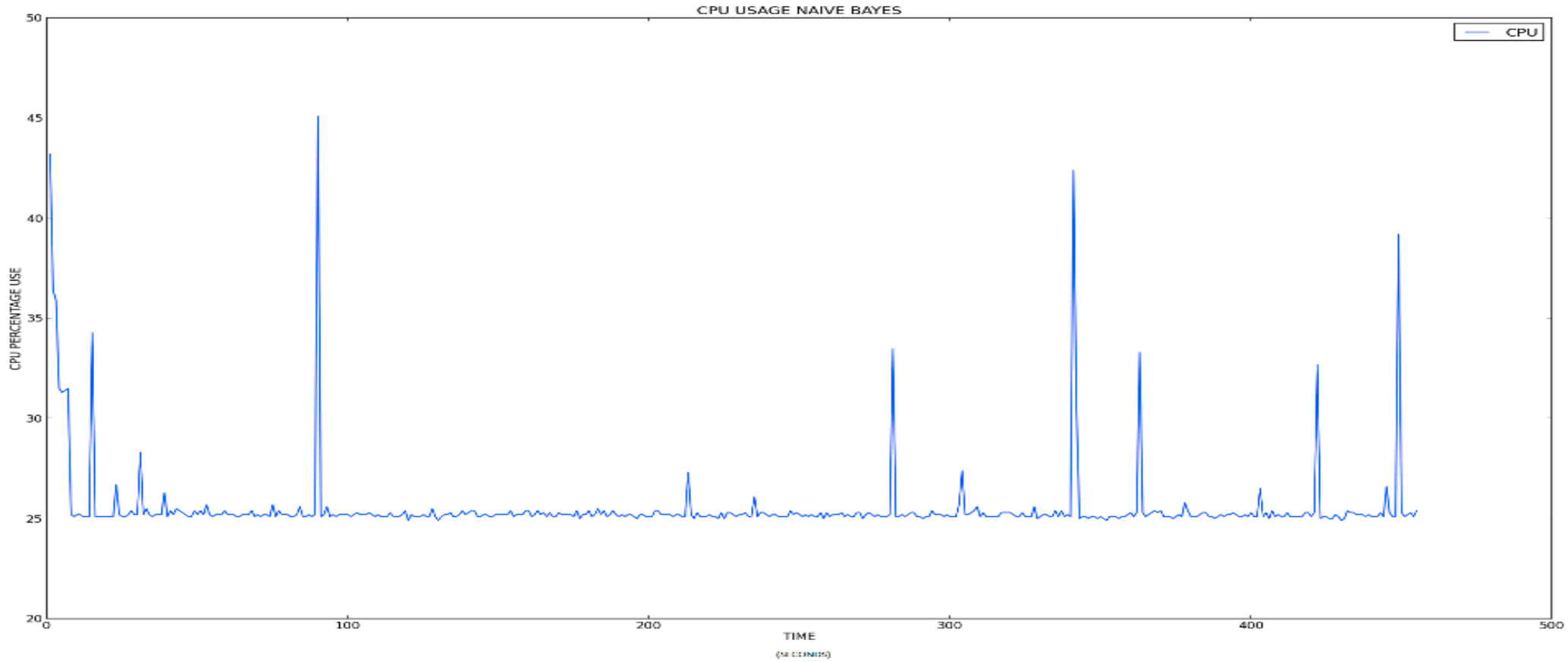
CPU Usage

- Using psutil (python system and process utilities)
  - /proc/pid/stat of pid
- Approximately 25% of the CPU is always being used by the algorithms

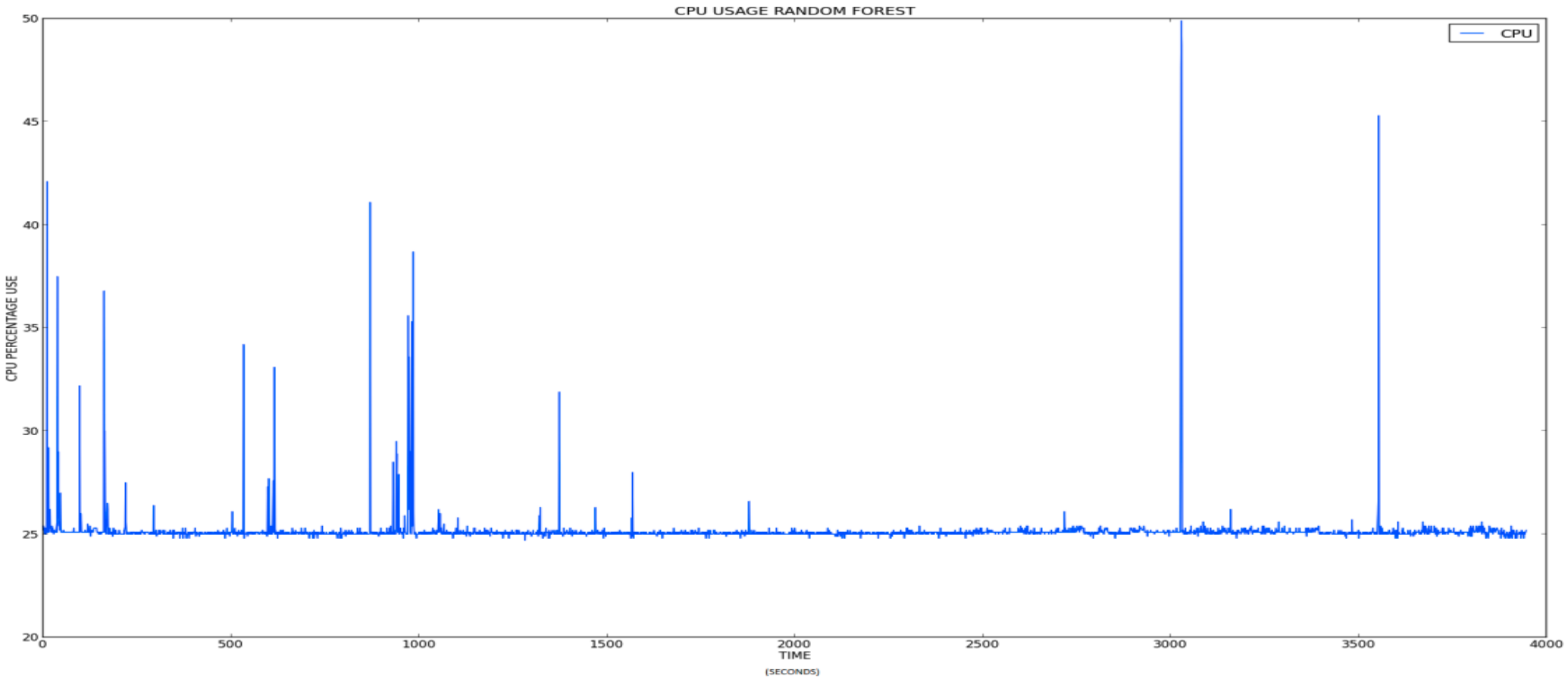
CPU Usage

- Importance
  - Lots of processes submitted to the CERN Grid
  - All processes should work in a cooperative environment
  - All processes should be able to access the required resources
- Statistical difference between CPU usage
  - Scikit-learn uses more CPU for longer time duration

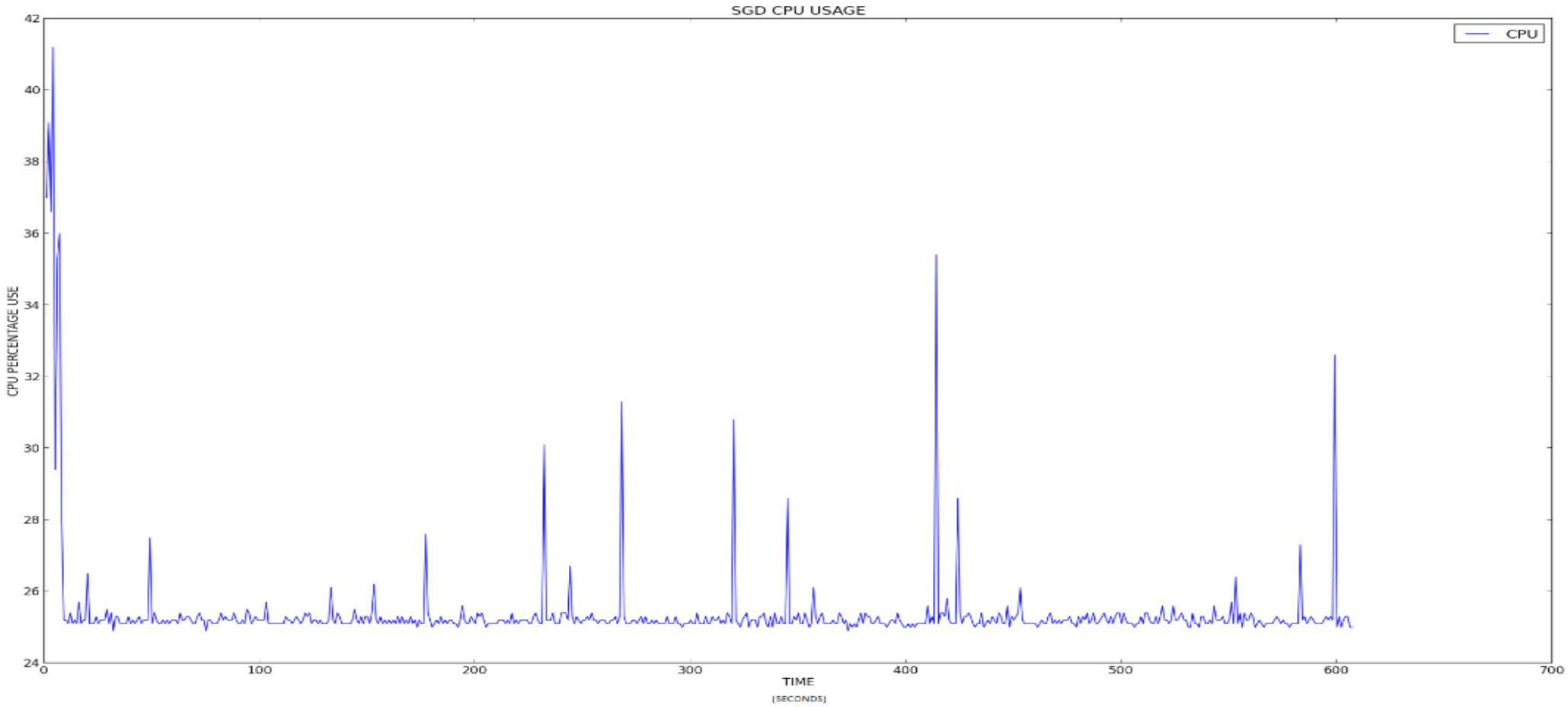
Naïve Bayes – Apache Spark



Random Forest – Apache Spark



Stochastic Gradient Descent – Apache Spark



Performance matrix – Apache Spark

Algorithm	Memory Used (GB)	Execution Time (Seconds)
Random Forest	5.072920	8184.136
Naïve Bayes	5.376336	1204.97
Stochastic Gradient Descent	5.181784	1261.94

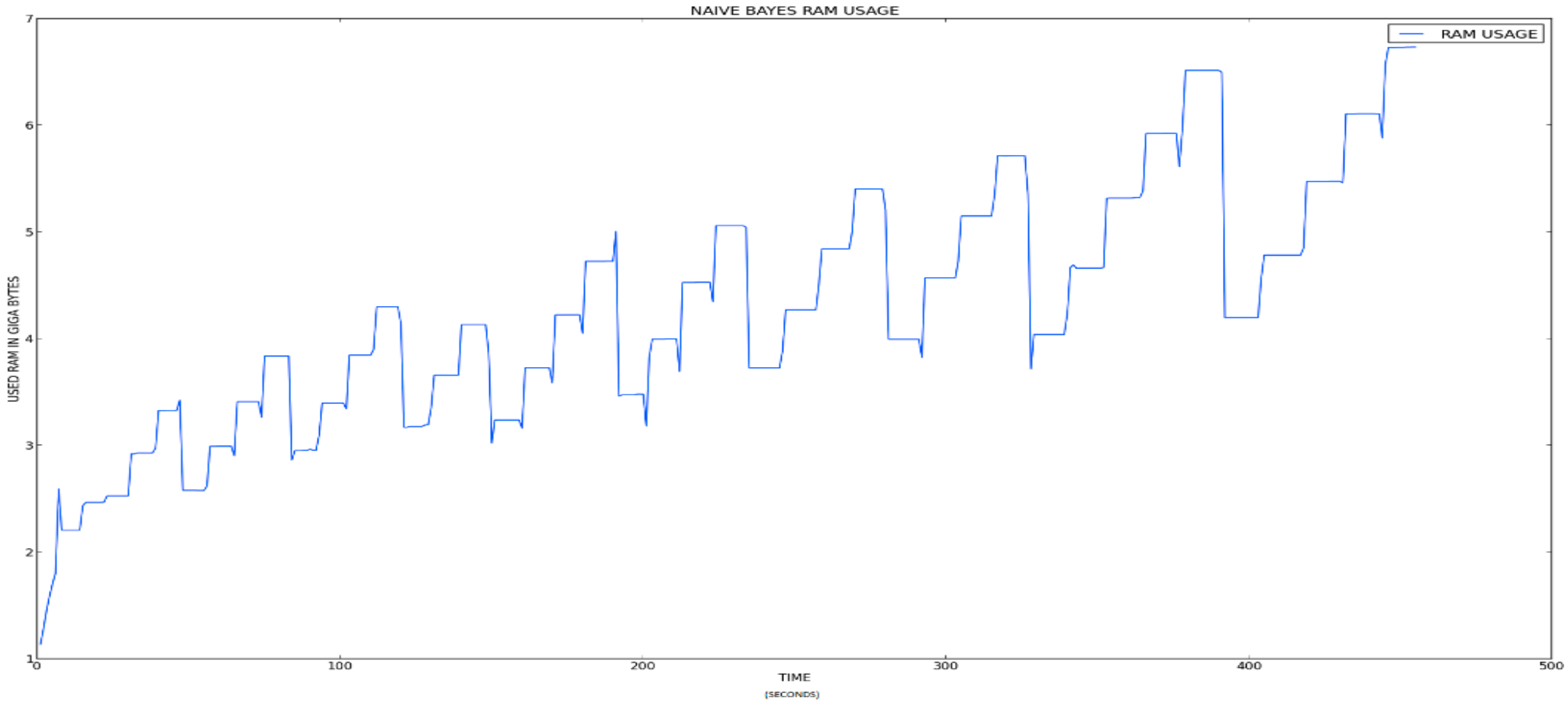
RAM Usage

- Incremental learning
  - With each iteration more RAM is utilized
  - Amount of data being processed is continually increased
- Graph does not indicate memory leak
  - Rolling approach
  - Incremental learning part

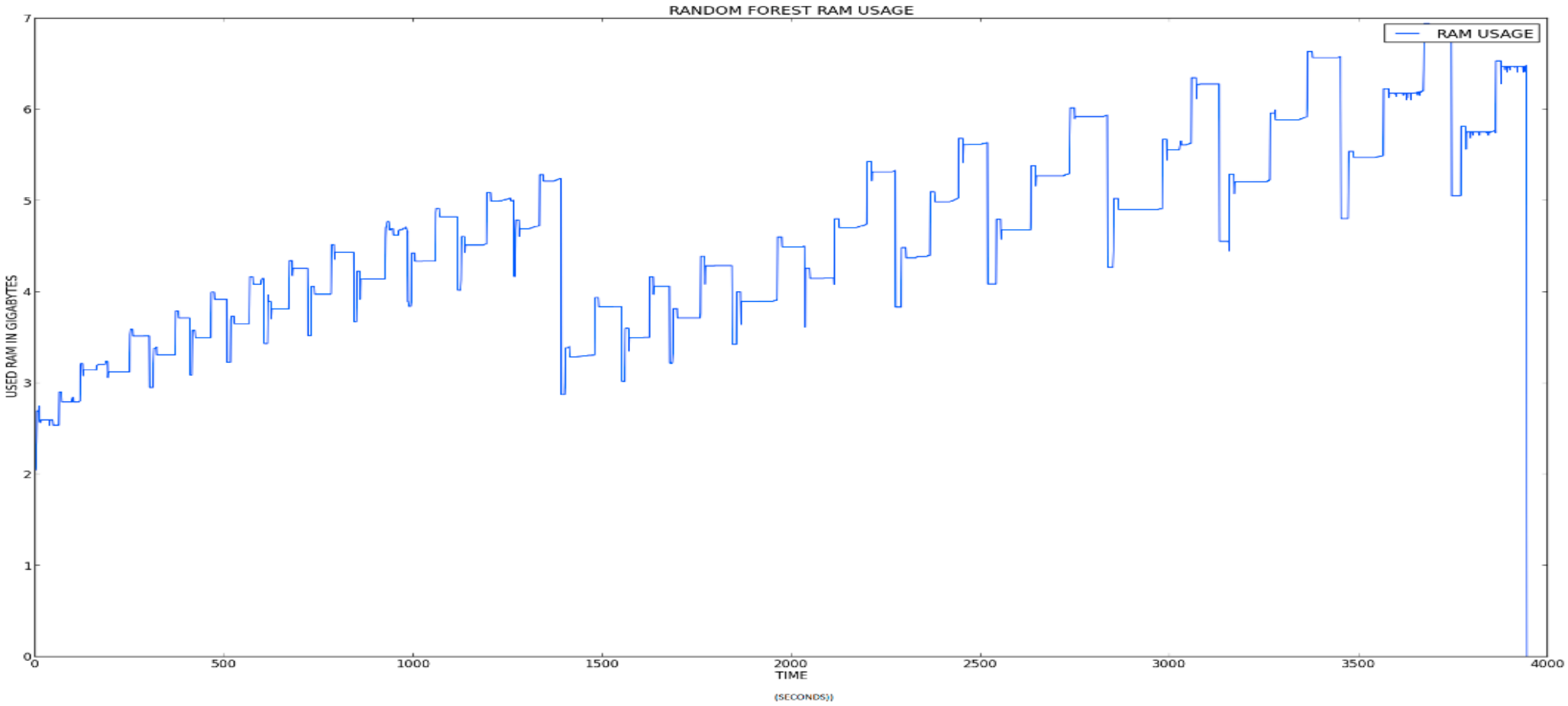
RAM Usage

- Statistical difference between RAM usage
  - Scikit-learn uses more RAM for longer time duration
  - RDD parallelism in Apache Spark

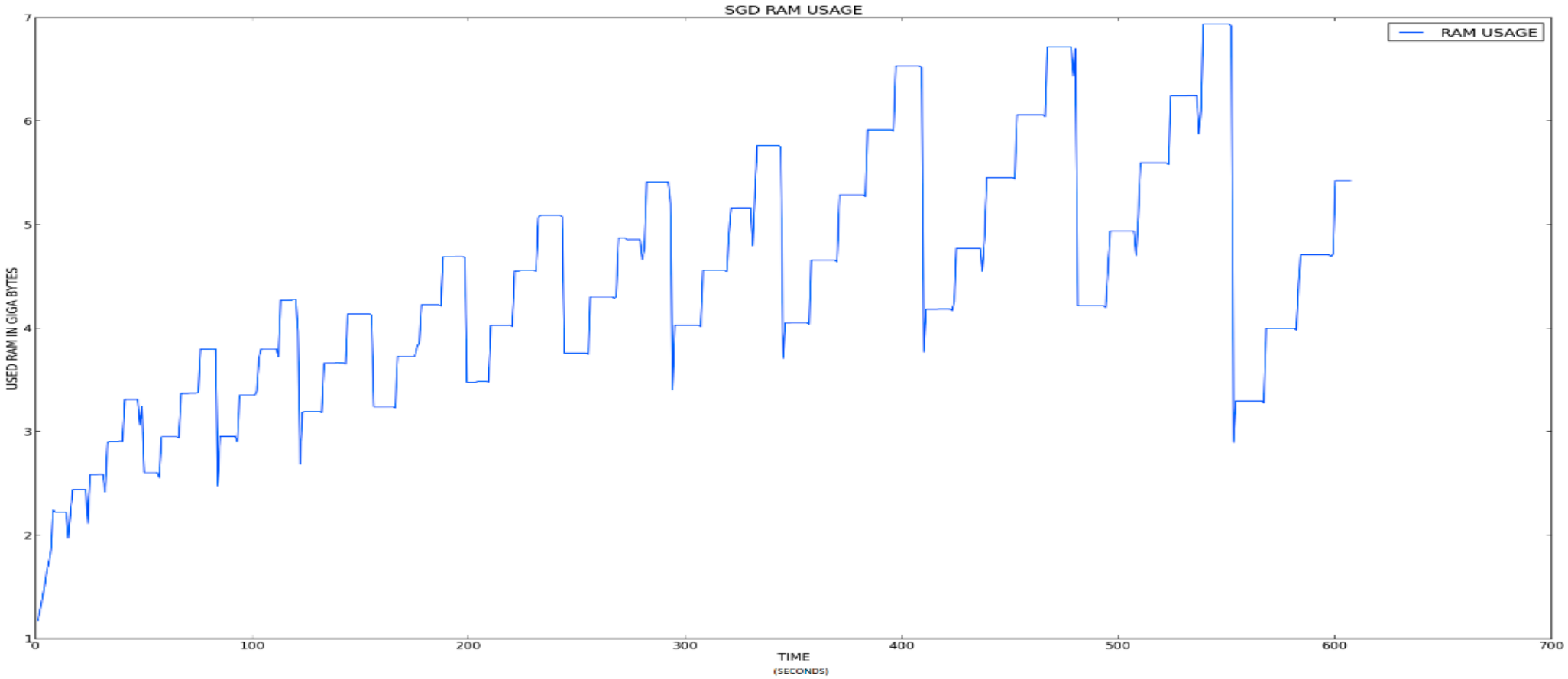
Naïve Bayes – Apache Spark



Random Forest – Apache Spark



Stochastic Gradient Descent – Apache Spark

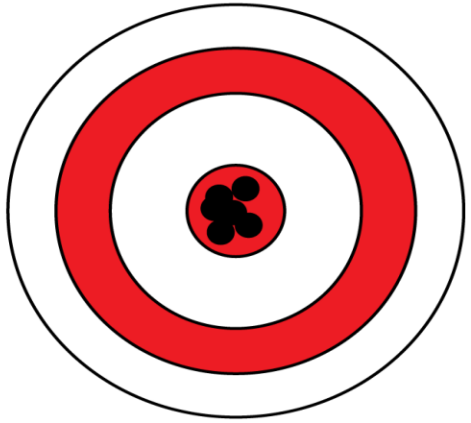


Results

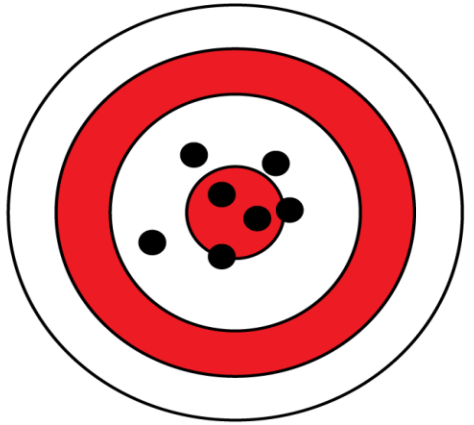
- False Positive Rate
  - FPR = FP / (FP + TN)
- False positive
  - Unnecessary replication
  - Dataset unpopular but predicted popular
- True Negative
  - Correct prediction
  - Dataset unpopular and predicted unpopular

Results

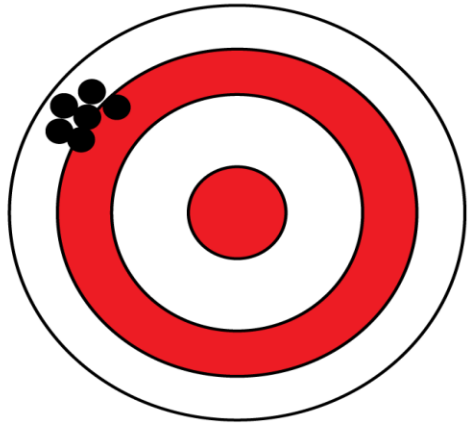
- Difference is not statistically significant



High precision and high recall
Returns many correctly labelled results



Returns many results
Most predictions incorrect



Returns few results
Predicted labels are correct

Results

- Accuracy problems because of data
  - Not reliable
  - Data can be unbalanced
  - Data Integrity
- Using the cross validation scoring method

```
from sklearn.metrics import accuracy_score
score(clf, test, target_test)
```

Confusion matrices

RF	P	N
T	570374	0
F	2	6438

SGD	P	N
T	568040	3351
F	2336	3042

NB	P	N
T	568415	6
F	12783	6432

Confusion matrix

Classifier	Accuracy	Precision	Recall	F1
RF	.98	.86	.98	.92
SGD	.96	.98	.62	.76

Classifier	Accuracy	Precision	Recall	F1
RF	.99	.99	1	.99
SGD	.99	.99	.99	.99

VS

From history to real

- Historical study of popularity & prediction of future popularity are two different concepts
- What was popular in the past
  - naccess, totcpu, nusers, likely to be related to popularity
- Predictions
  - Don't know the number of accesses
    - Use different features
    - From solid understanding (history) to future prediction is difficult
    - May not be able to use the best metrics

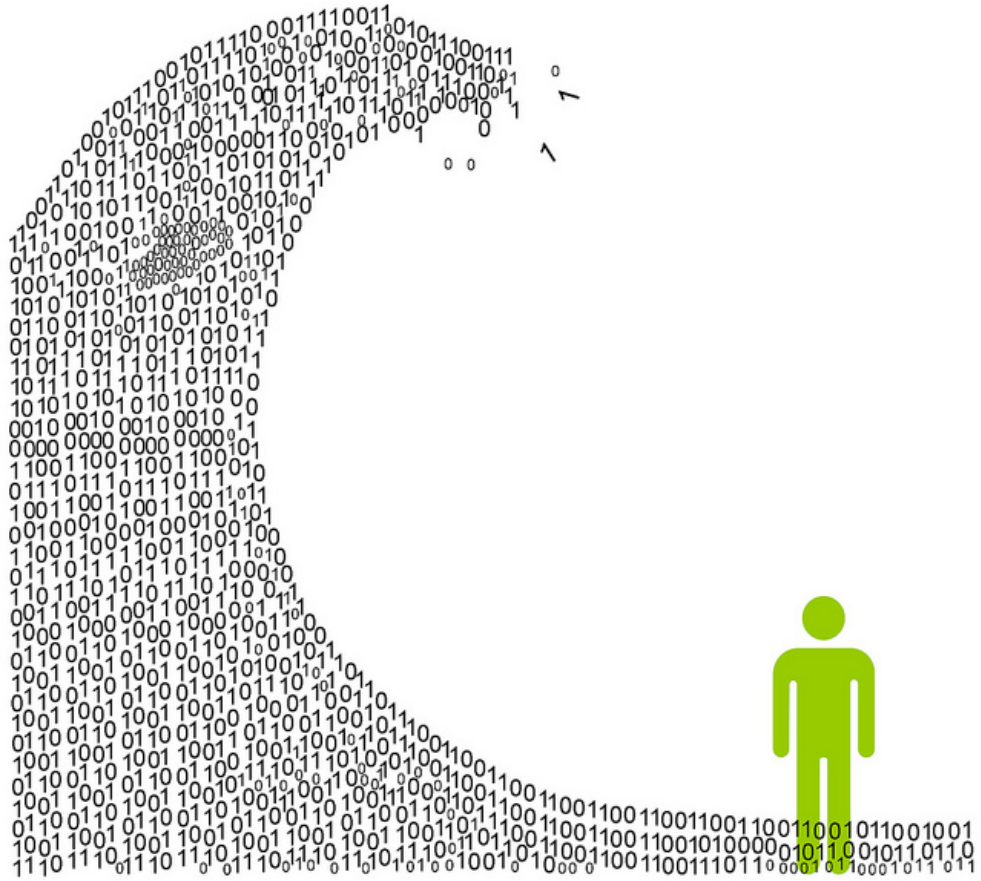
Apache Spark v/s scikit learn

- Real time results
- Mesos Cluster manager
- Interfaced with Hadoop Distributed File System (HDFS)
- Aggressive caching in memory
- Faster and Scalable
- RDD level parallelism
- Version Used:
  - Release 1.4.0

- Works directly as a library
- User-friendly
- Benchmarked models already in use
- Deployed over:
  - Python 2.7.5
  - Scikit version 0.16.0

Python scikit-learn

- High CPU usage ☹
- High RAM usage ☹
- More Time ☹



Apache Spark

- CPU usage looks Good ☺
- RAM usage looks Good ☺
- Less Time ☺

Thank You

References

1. Siddha Ganju, 2015, Evaluation of Apache Spark as Analytics as framework for CERN's Big Data Analytics, CERN Zenodo network
2. Toby Segaran, 2007, Programming Collective Intelligence, O' Reilly
3. Sandy Ryza, Uri Laserson, Sean Owen, Josh Wills, Advanced Analytics with Spark, Patterns for Learning from Data at Scale, O' Reilly
4. Holden Karau, Andy Konwinski, Patrick Wendell, Matei Zaharia, Learning Spark, Lightning-Fast Big Data Analysis, O'Reilly
5. Apache Spark Documentation. Retrieved from http://spark.apache.org/
6. Apache Spark Programming Guide. Retrieved from http://spark.apache.org/docs/latest/programming-guide.html
7. Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux, 2011. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30, DOI:10.1109/MCSE.2011.37
8. Wes McKinney, 2010. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56
9. Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay, 2011. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830
10. Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001. Retrieved from http://www.scipy.org/