EXPLOITING APACHE SPARK PLATFORM FOR CMS COMPUTING ANALYTICS

Marco MEONI, INFN (presenter)
Luca MENICHETTI, CERN
Valentin KUZNETSOV, CORNELL
Tommaso BOCCALI, INFN
Daniele BONACORSI, UNIBO

08/22/2017 – ACAT 2017 – Seattle

In a Nutshell

- Hadoop clusters @ CERN feature ~5 PB of raw storage
  - 39 nodes with 64GB of RAM and 32 cores/node
  - Mix of Intel/AMD and CentOS7/SLC6
- Since 2015 CMS has stored large sets of computing logs
- Evaluation of Apache Spark
  - Scalable and efficient processing of metadata on Hadoop platform
  - Leverage in-memory and persistence API
  - Scala vs Python
- How an experiment can benefit from Spark
  - Reduce processing time on large dataset (DS) and metadata
  - Perform M.L. studies/analytics over large DS from multiple data sources

Use Cases

- Site Utilization
  - Which DS occupy my T[1-3] site?
- Users Activities and Daily Stats
  - Most accessed Datatier or DS, job throughput, failure distributions
- Machine Learning (M.L.) on DS Popularity
  - Fast queries for reprocessing of monitoring time intervals
  - LFN-to-DS aggregation and correlation
  - Assessment of predictive models that can learn DS utilization profiles

Computing Logs on Hadoop

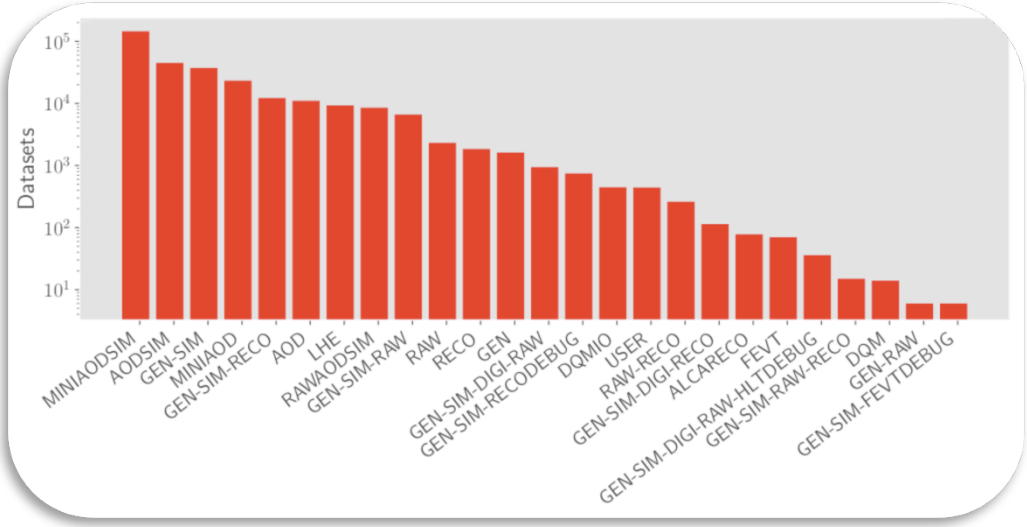
- *Billions* of 2015-17 file access logs, heterogeneous sources
- *Millions* of samples after file aggregation by dataset
- *Thousands* dataframes built on common attributes for M.L.

Source	Items	Type	Description
EOS	786,934,116	structured	Disk storage system at CERN
AAA	2,370,570,956	structured	CMS XrootD federation for Grid data
CRAB	1,177,951	structured	Grid infrastructure for job submission
DBS3	5,193,522	structured	Global dataset/fileblocks catalogue
Block-Replicas	805,614,541	structured	Global replica catalogue
PhEDEx	58,227,786	structured	Fileblock locator and export service
CADI	1,791	semi-struct	CMS conference database

Site Stats by Datatier

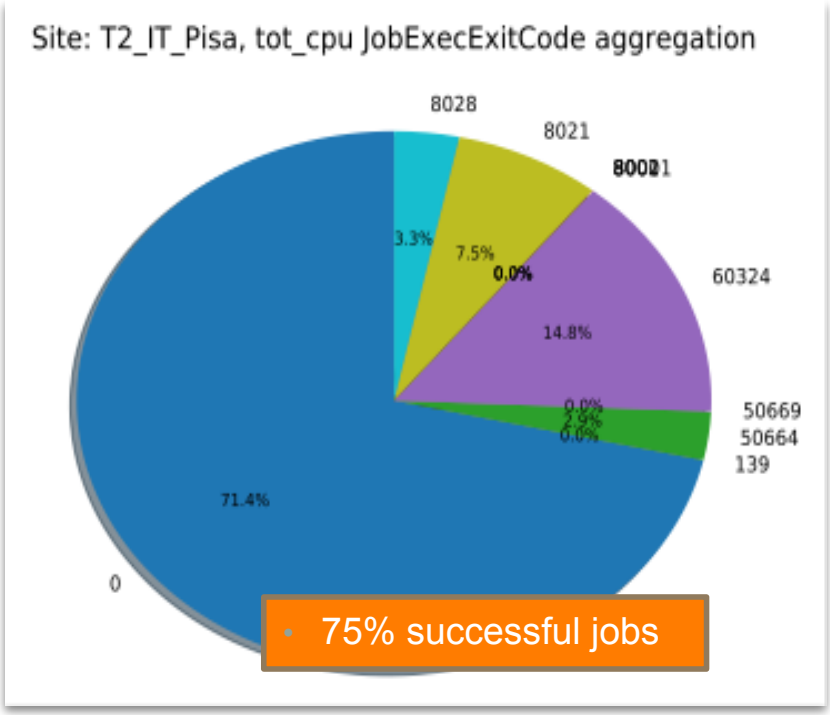
- Stats for all CMS sites provide insight on user activities
- Easily merge DBS+PhEDEx in Spark job, not possible with ORACLE
- Process billions of logs, e.g. AAA DS, in reasonable time

```
T2-sites           dataset size=1.283e+17 (113.989 PB)
evts=250809312370   replica size=7.446e+16 (66.137 PB)
-------------------------------------------------------
                        evts          size  replica size
tier
AOD           162782404863  2.031304e+16  1.959890e+16
RAW          1683921651218  4.498240e+16  1.722459e+16
AODSIM         34936712479  9.451795e+15  9.058594e+15
RECO           55951414059  1.714168e+16  7.169092e+15
MINIAOD       246201671780  4.792439e+15  4.780326e+15
MINIAODSIM    132260740917  4.145671e+15  4.075050e+15
GEN-SIM        35102730685  1.502151e+16  3.213296e+15
ALCARECO       95433911221  3.302962e+15  3.095442e+15
```

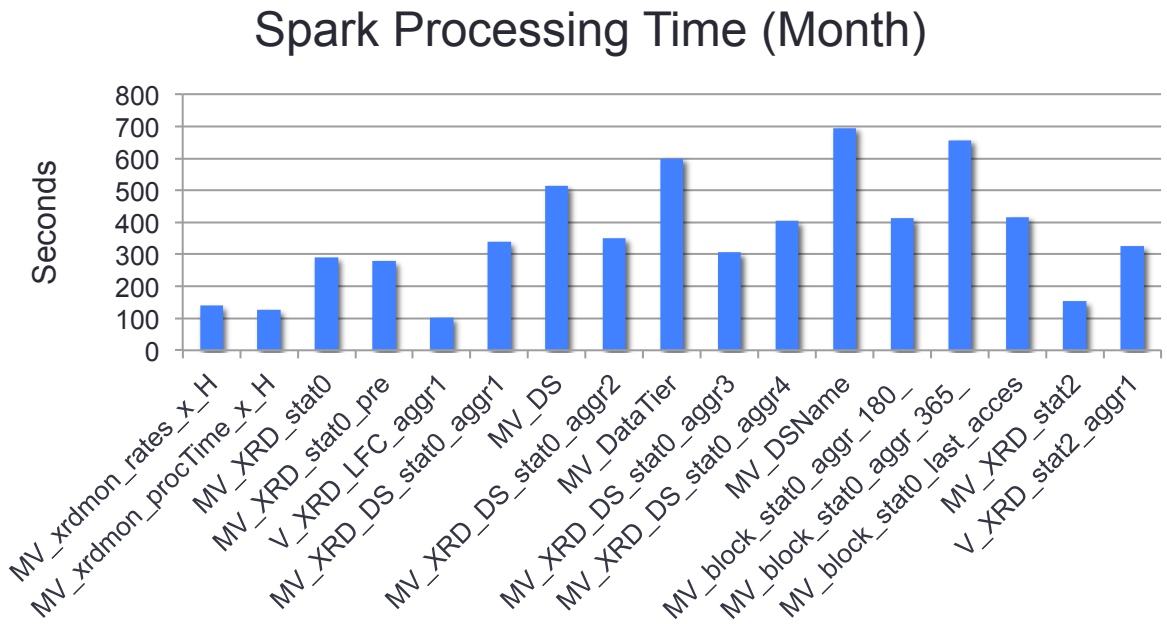


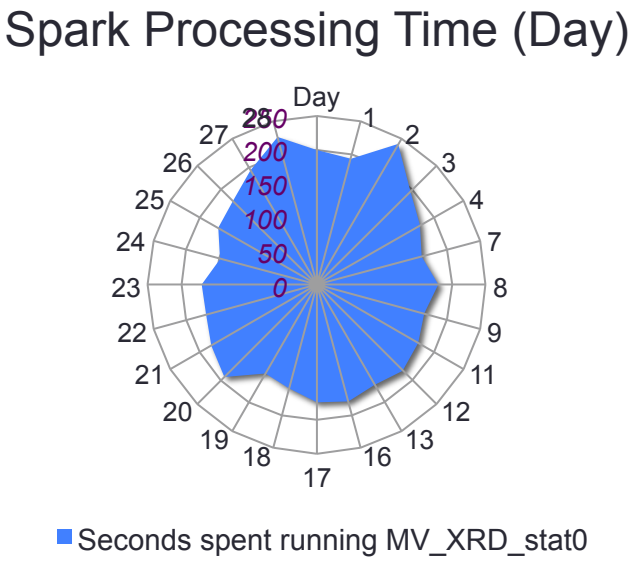
Job Throughput by Daily Stats

- Spark allows to look at different angles of CMS data by aggregating streams
  - Almost impossible with distributed DB
  - See job throughput or identify failures
  - Easy to look at specific sites or slice data in different dimension
  - Provide control and flexibility for data monitoring



Monitoring DS Popularity





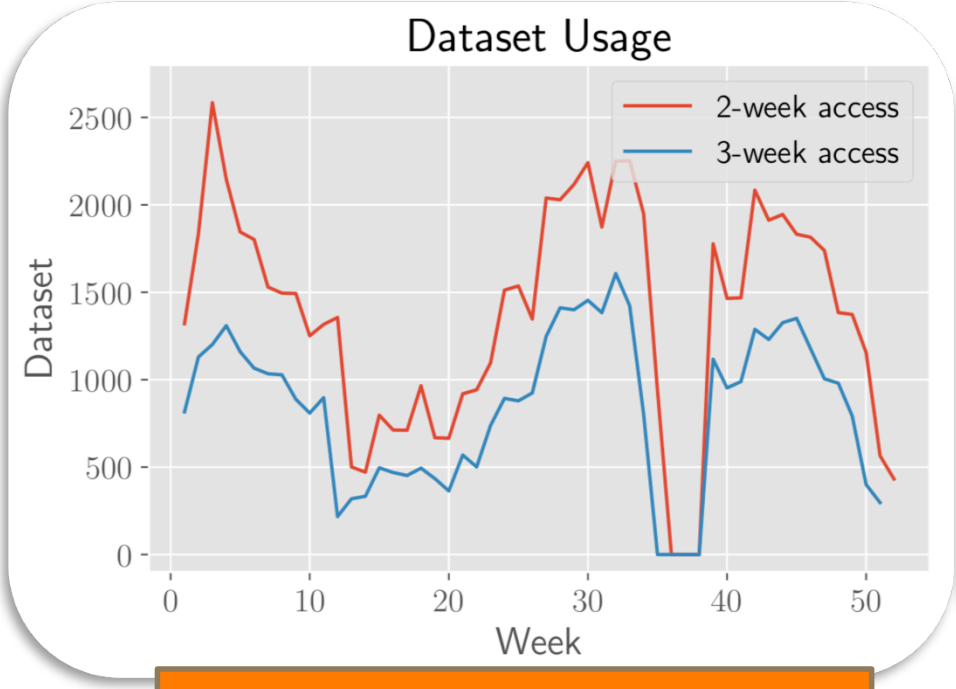
- Oracle: continuous running of incremental MV update
- Hadoop: re-processing of popularity queries for any time interval is fast

Machine Learning on DS Popularity

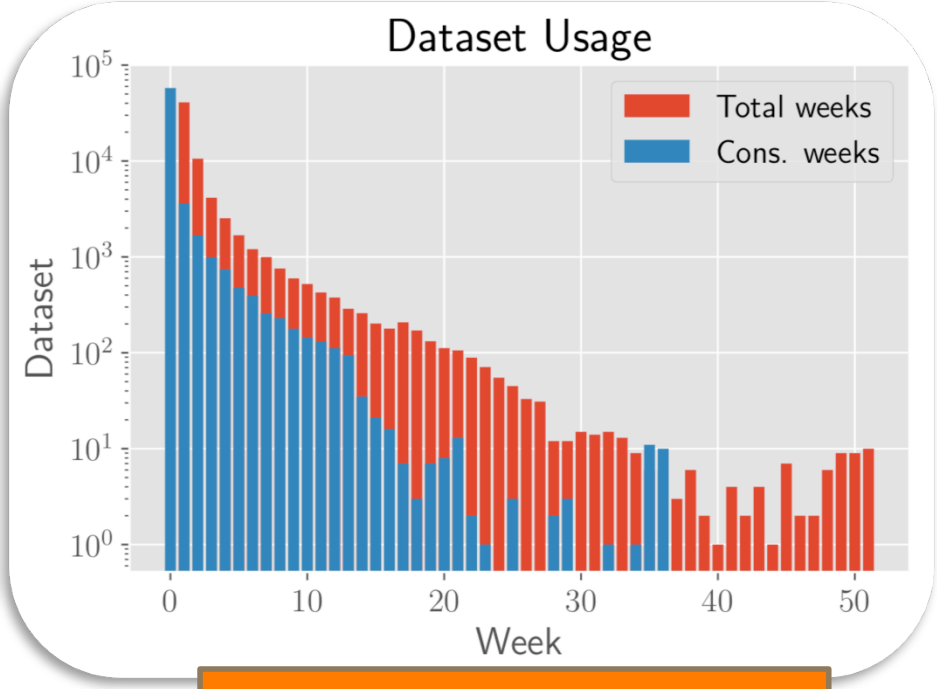
- Data placement at most WLCG experiments is based on DS popularity
  - need to make optimal choice of replication to maximize data availability for processing and analysis
- Common questions
  - Can we *predict* the popularity of a new dataset?
  - Pre-compute the proper number of replicas?
  - Delete unpopular replicas?

Understanding Popularity Timespan

- 1 week is the ideal timespan for DS Popularity Prediction



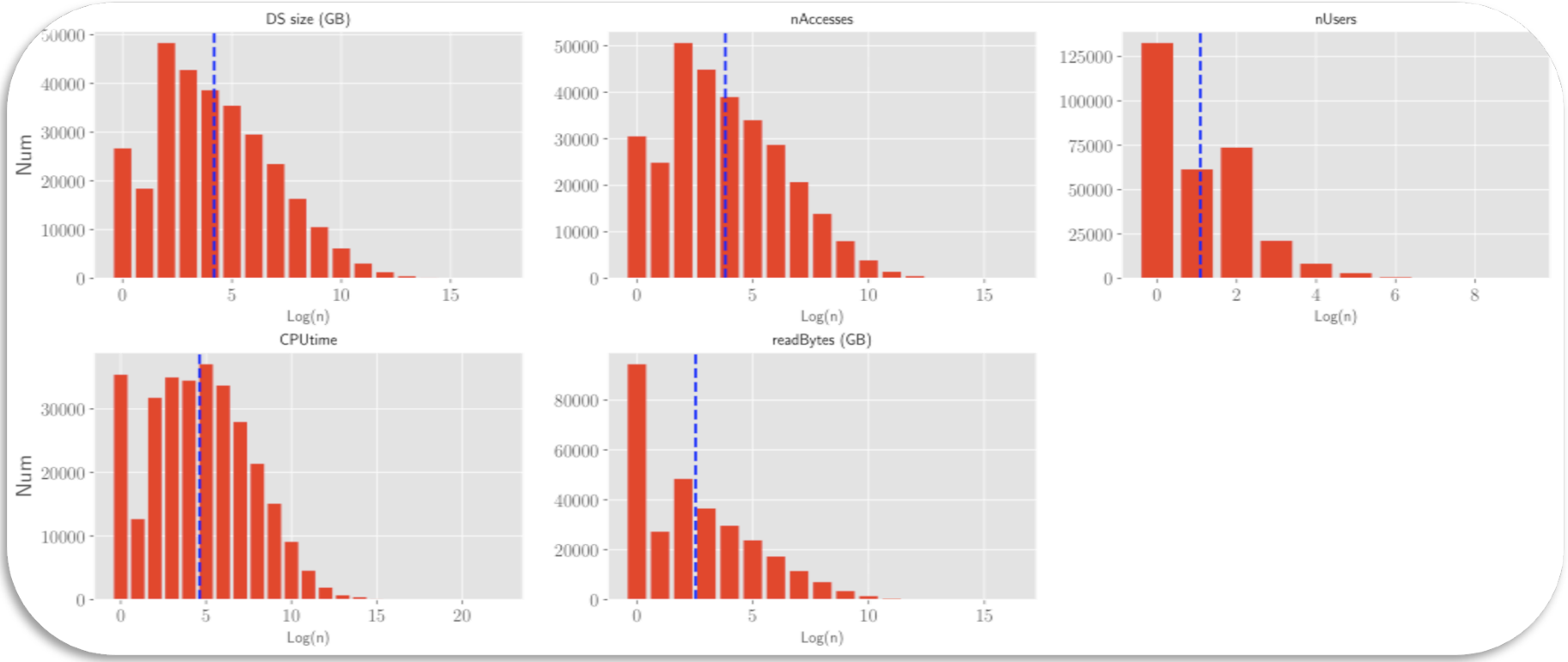
Accesses on 3 weeks are ~50% less frequent



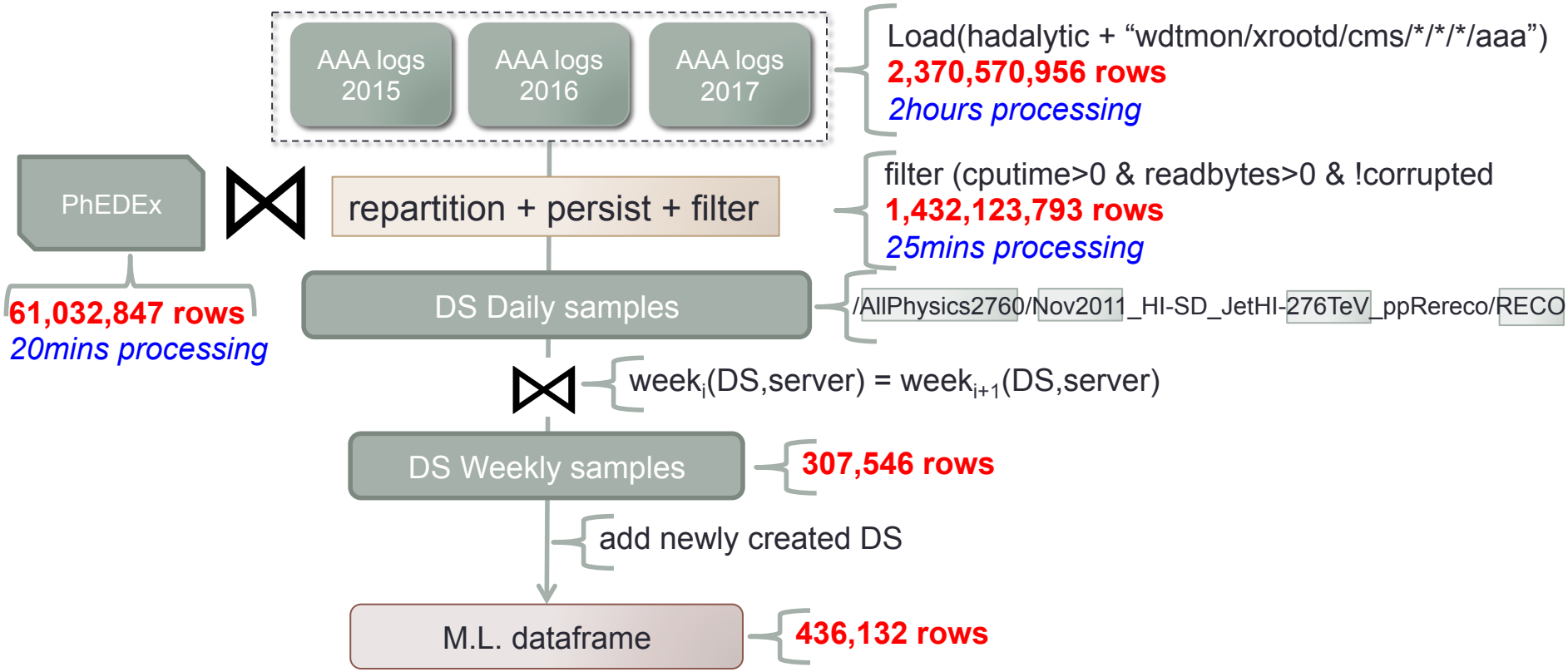
DS have relatively short access pattern

Ideal Popularity Cutoffs

- Different cutoffs on the usage features bring different popularity definitions

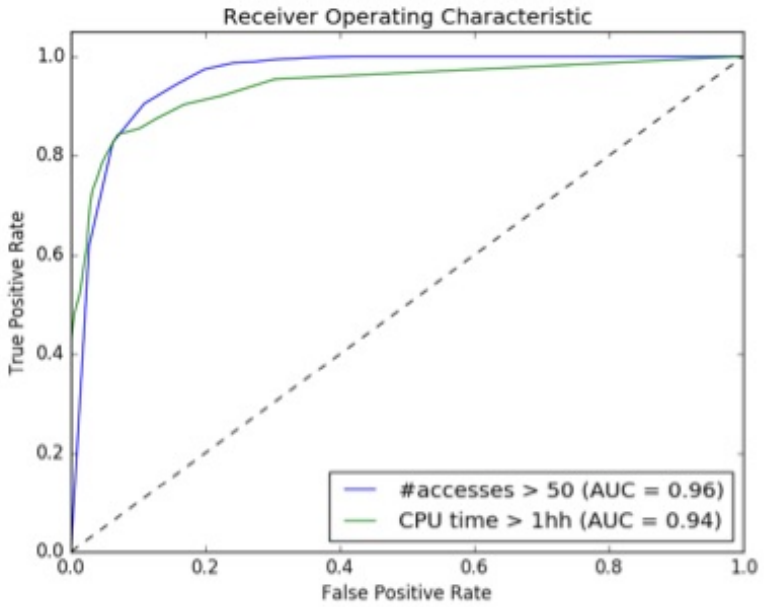


Generation of M.L. Dataframe



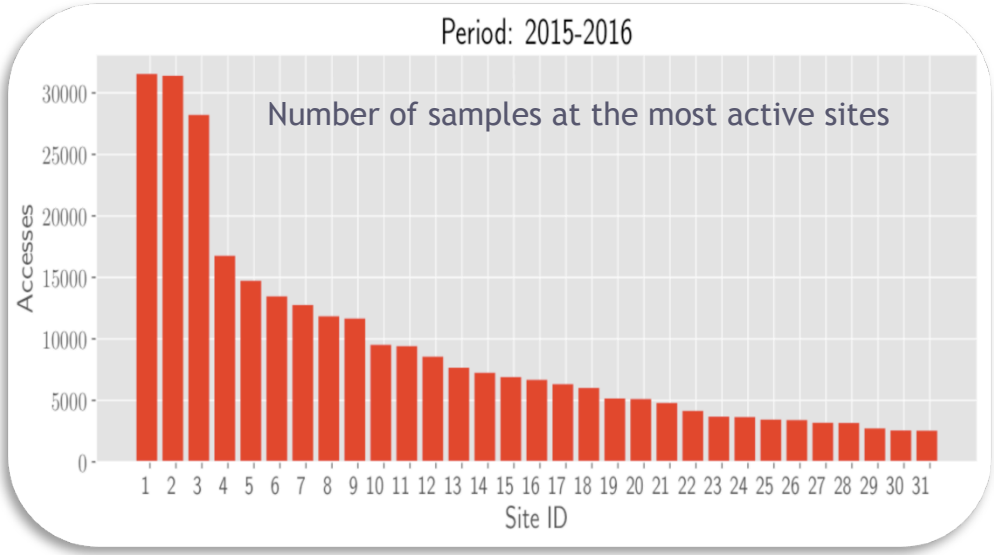
Model Training on Spark

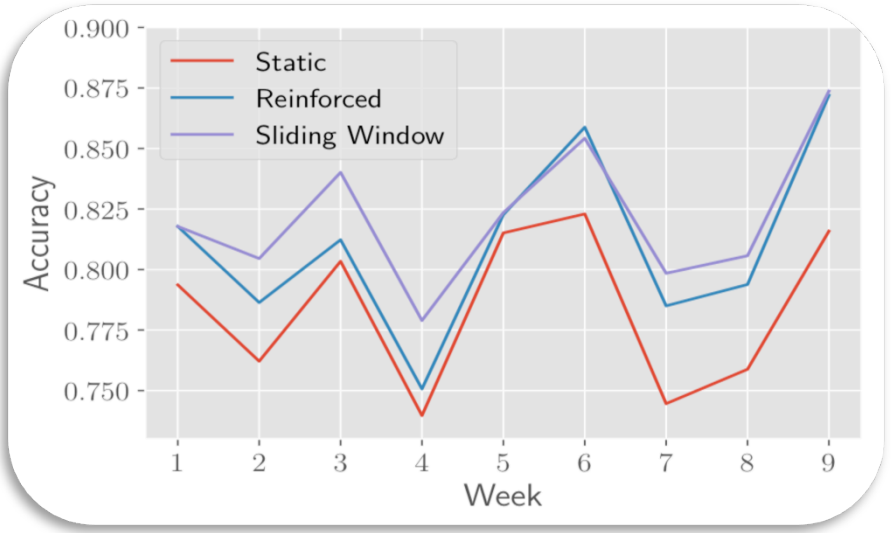
- Load Input Samples
  - Weekly CSV samples (create maps of categorical features)
  - Weekly SVM samples
- Define the Features Array
  - Make LabeledPoint of features
- Train and Score Classifiers
  - Python SkLearn
  - Scala MLlib
    - Compute classification probabilities



Enhancing the Predictive Models

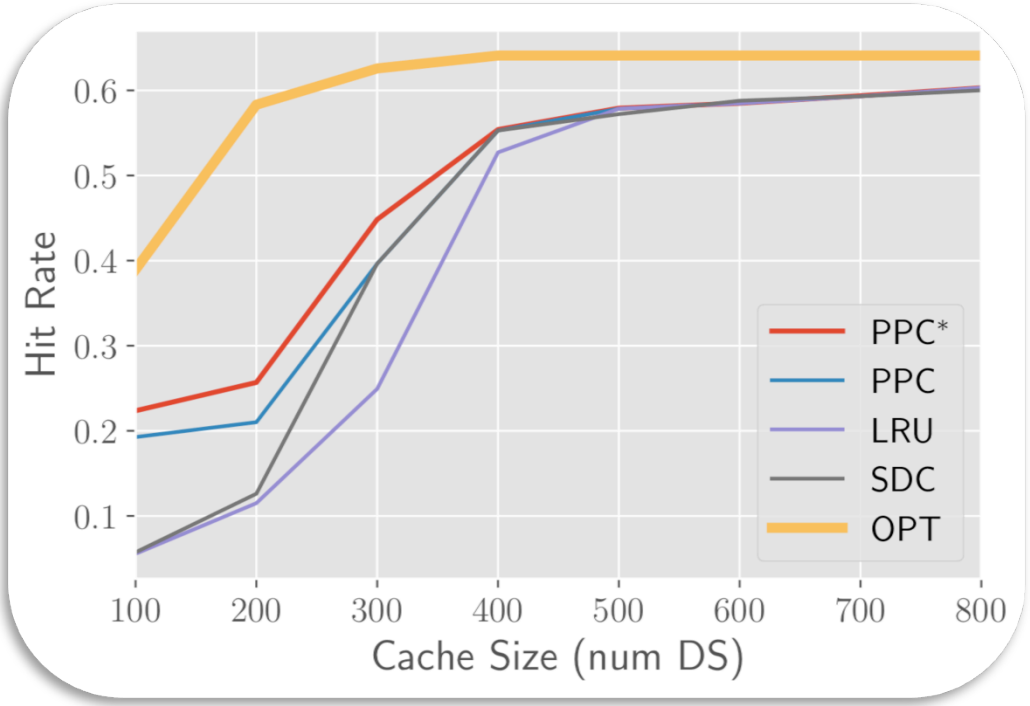
- Feature Engineering
- Global vs Site-level Models
  - Evaluate training of site-classifiers as locality could exist in DS accesses
- Model aging and refreshing





DS Caching based on Popularity

- Strategy: do not evict cache elements if popular next week



- PPC always outperforms LRU
- PPC is 2x better with limited cache
- $\text{HitRate}_{max} = \frac{1 - \text{Misses}_{compulsory}}{\text{numDS}}$

CMS Computing Analytics on Spark

- Optimization of Spark cluster settings
  - spark.core.max, spark.executor.cores, spark.dynamicAllocation.maxExecutors
  - repartition(), persist(MEMORY_AND_DISK)
- Scala
  - Emerging language, Spark is written in Scala, Dataframe/RDD parallelism available out-of-the-box
- Python
  - Language known in HEP, but PySpark requires additional training
  - PySpark is a wrapper around Java libraries (slower versus native libs, need to handle memory issue not known in Python, etc.)
- Program throughput highly depends on its structure
  - Use Dataframe operations instead of iterations
  - Apply functions from Spark API instead of language specific (PySpark)

Conclusions and Outlook

- Spark is an extremely useful platform to crunch large DS
  - Quick reprocessing of CMS statistics for all sites, daily job stats, etc.
- XrootD DS popularity is very important to CMS operations
  - Leverage computing analytics on Hadoop/Spark, scalability, M.L.
- Best approach
  - Run analytics with Scala (RDD, dataframe)
  - Run M.L. with PySpark (reduced sample set, SkLearn state of the art)
- Work in progress
  - Application of DS predictive models to DS replicas
- Acknowledgement
  - CERN-IT for the cluster availablility, maintenance and support