ISGC'17 - Taipei (ASGC)

Machine Learning analysis of CMS data transfers

V. Kuznetsov (Cornell Univ., US)
N. Magini (Fermilab, US)
D. Bonacorsi (Univ. of Bologna / INFN)
T. Diotalevi (Univ. of Bologna)
A. Repečka (Vilnius Univ., Lithuania)
Z. Matonis (Vilnius Univ., Lithuania)
K. Kančys (Vilnius Univ., Lithuania)

(on behalf of the CMS experiment)

Outline

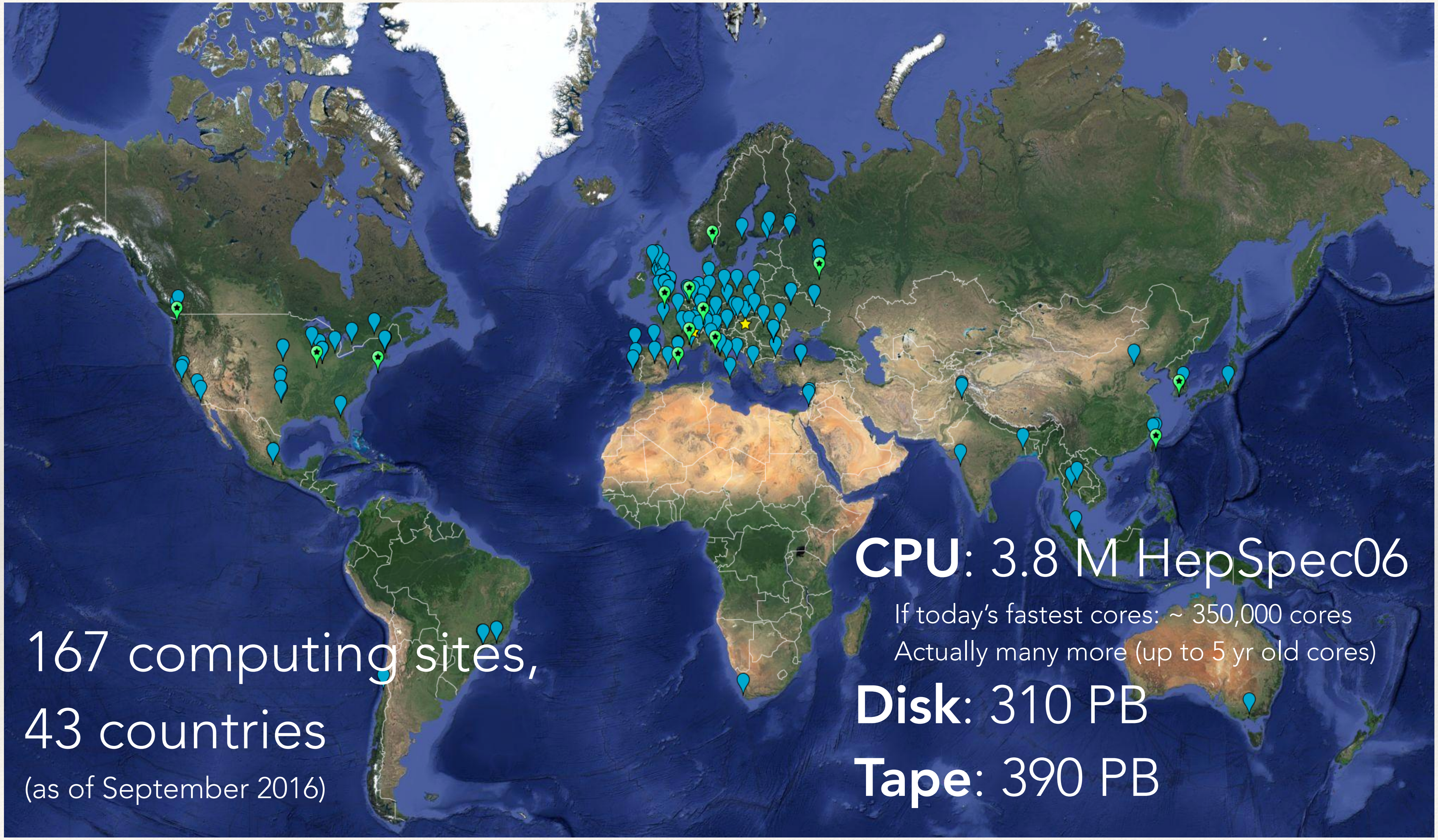
Introduction

Experience on **CMS dataset popularity** studies

Preliminary results on **CMS data transfer latency** studies

Introduction

Worldwide LHC Computing Grid



167 computing sites,

43 countries

(as of September 2016)

CPU: 3.8 M HepSpec06

If today's fastest cores: ~ 350,000 cores

Actually many more (up to 5 yr old cores)

Disk: 310 PB

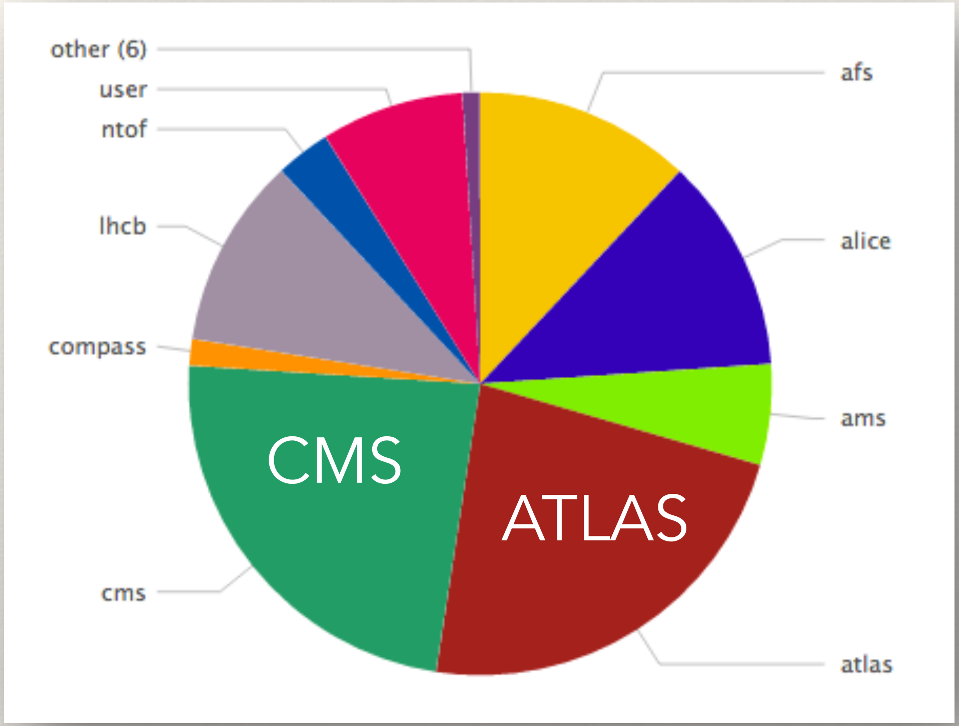
Tape: 390 PB

[credits: I. Bird]

Data collection and distribution

Data taking in 2016 broke many records! Massive volumes of data written to storage

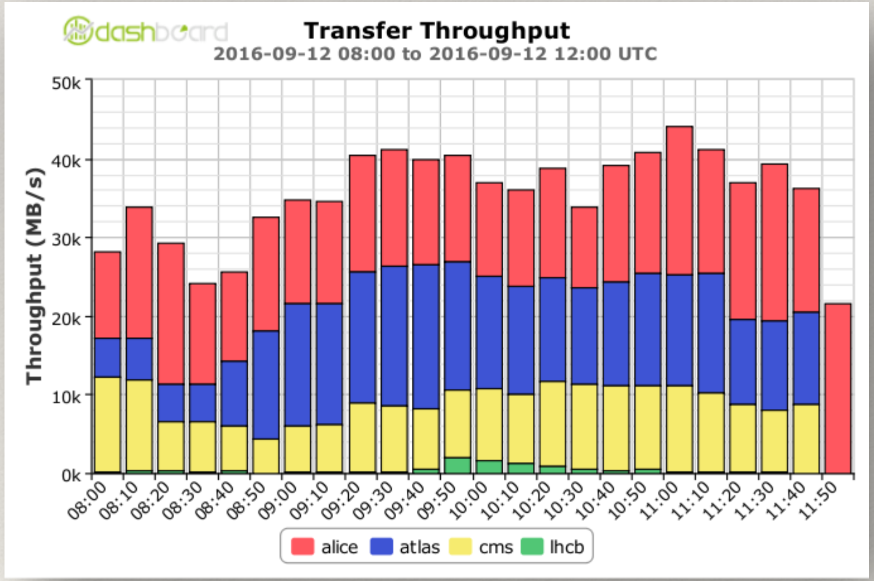
- June-August 2016 at **>500 TB/day**
- **10.7 PB** recorded only in July 2016
- 2016: **>~35 PB** LHC data
- CERN tape archive is **~160 PB**



[credits: I. Bird]

Data distributions over high-performance networks

- global data transfer rates increases to **>40 GB/s** (2x Run-1)
- regular transfers of **80 PB/month**
- many billions of files..



[credits: I. Bird]

CMS: towards **adaptive modelling**

CMS built a computing system that **worked** in LHC Run-1/2.

Do we fully "**understand**" it?

- Can we perform precise modelling of specific workflows / site behaviours / system performances? Can we use this modelling to make predictions?
    - e.g. population vs pollution of Tier disks; TierX - Tier-Y data transfer patterns; ..

Computing operations (meta-)data is **all archived**

- but rarely (or never) accessed
    - e.g. transfers, job submissions, site performances, releases details, infrastructure and services behaviours, analysis accesses, .. you name it!
- we basically monitor to debug in near-time, not to analyse and learn from the past to design and build what's next

Here is where Big Data zoology, Analytics techniques, ML(/DL), come in:

- a complementary "data scientist" approach to CMS computing metadata

Extract **actionable insight**, towards **adaptive modelling of CMS workflows**

This is where ML code lives..

Engineering Effort for Effective ML

- From "Hidden Technical Debt in Machine Learning Systems", D. Sculley at al. (Google), paper at NIPS 2015

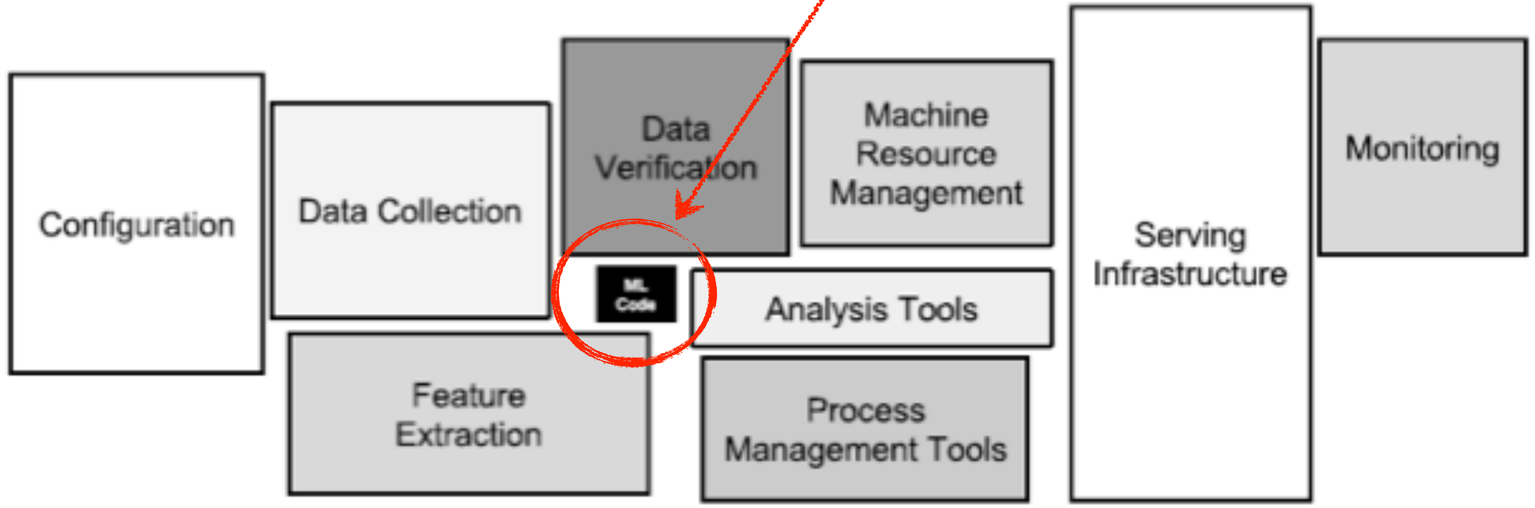


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

[credits: L. Canali, CERN-IT, internal presentation, Dec 2016]

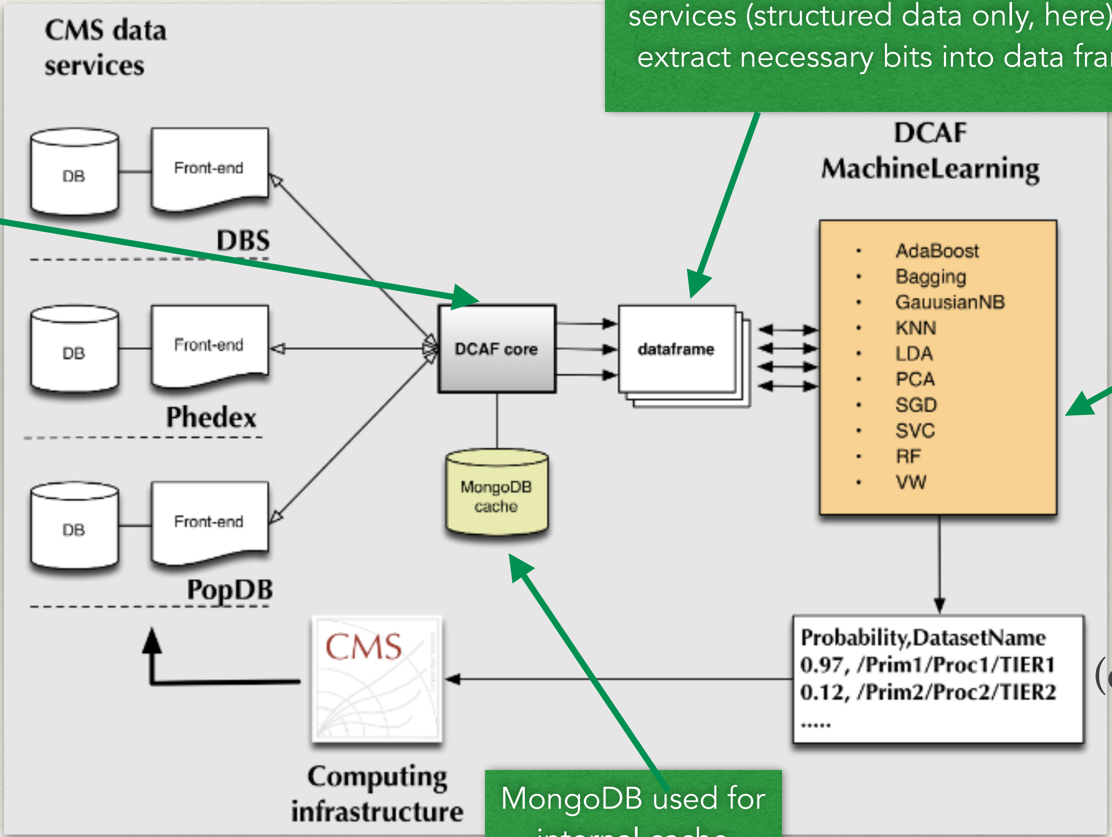
The CMS **DCAF**

Refs: [1]

DCAF = Data and Computing Analytics Framework

Aim is to collect information from CMS data-services and represent it in a form suitable for "analysis" (e.g. ML tools)

- Pilot project finalised and in use



```
Probability,DatasetName
0.97, /Prim1/Proc1/TIER1
0.12, /Prim2/Proc2/TIER2
.....
```

(one example)

Metadata collection in **Go**: gain in speed thanks to concurrency (wrt original Python code base)

Dataframe generator toolkit: collects/transforms data from CMS data services (structured data only, here) and extract necessary bits into data frames

ML algorithms (python / R code) for data analysis

MongoDB used for internal cache

[credits: V. Kuznetsov]

DCAF (continued)

Refs: [1]

Framework consisting of several layers:

- **storage** layer, which can be used to keep information from various CMS data-services.
  - Currently, DCAF uses MongoDB and associated py-mongo driver as a storage solution, but it can be replaced with any other key-value store or RDBMS database
- **mapping** layer to provide mapping between sensitive information and its numerical representation
  - e.g. DNs into set of unique ids
- **aggregation** layer which collects and merge information from various CMS data-sources
- **representation** layer will take care of data representation suitable for analysis framework
  - e.g. represent our dataframe into CSV data-format
- **analysis** layer which will either use a custom analysis algorithm or provide bridge to other learning system libraries
  - The easiest solution would be to use python sklearn library [7] which already provides broad variety of learning algorithms

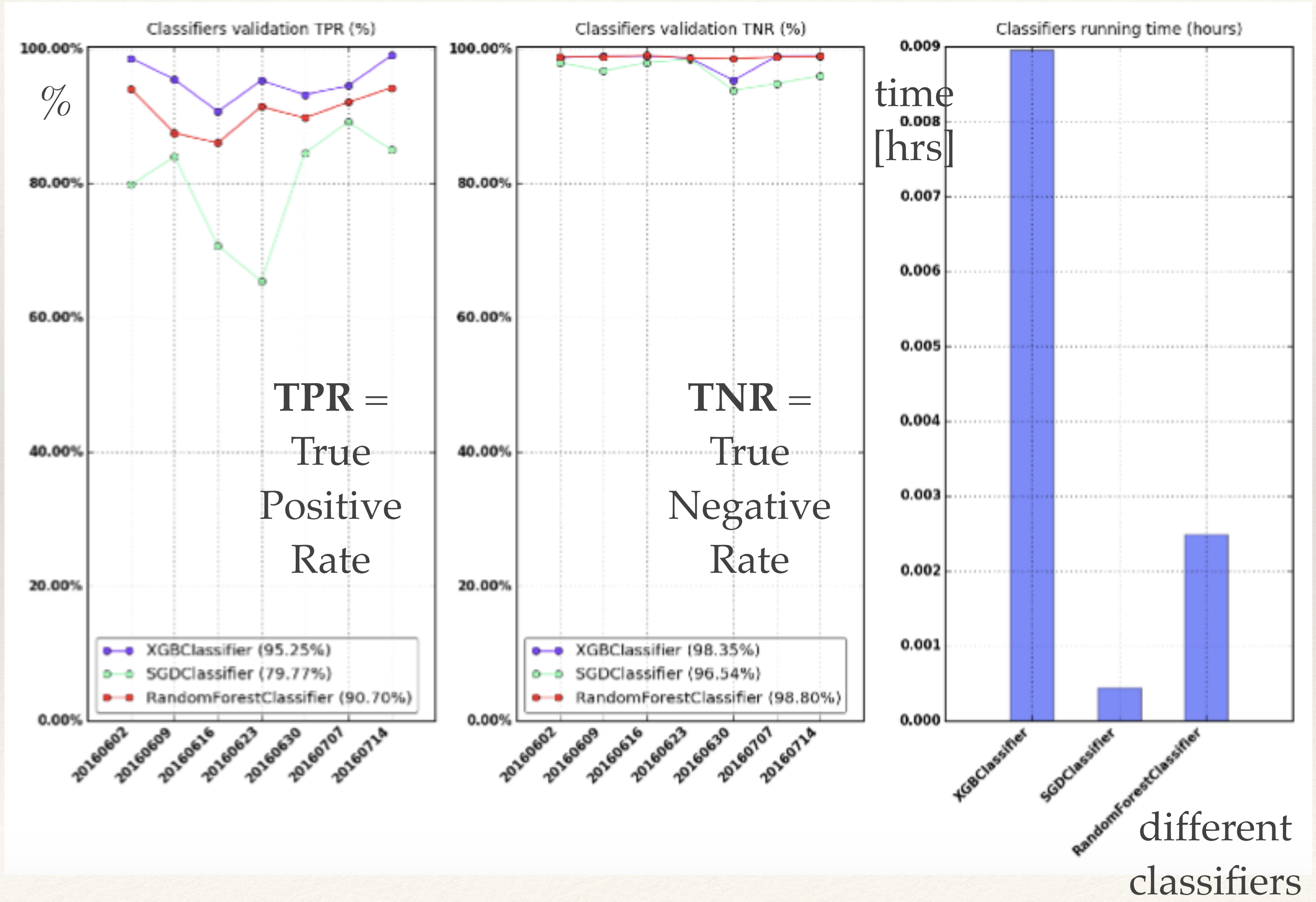
One use-case:

dataset popularity [*]

[*] How “popular” a CMS dataset is for distributed analysis users,
defined by different possible metrics,
e.g. # accesses, # users accessing it, CPU-hrs spent on it, ..

Predictability of CMS dataset popularity

The ability to predict the popularity of a CMS dataset allows to **optimise storage utilisation** (our most expensive computing resource!)



Refs: [2,3,4]

Moving to **Spark**

Most useful is to constantly update the ML-based predictions for CMS data popularity, in “sliding” time windows

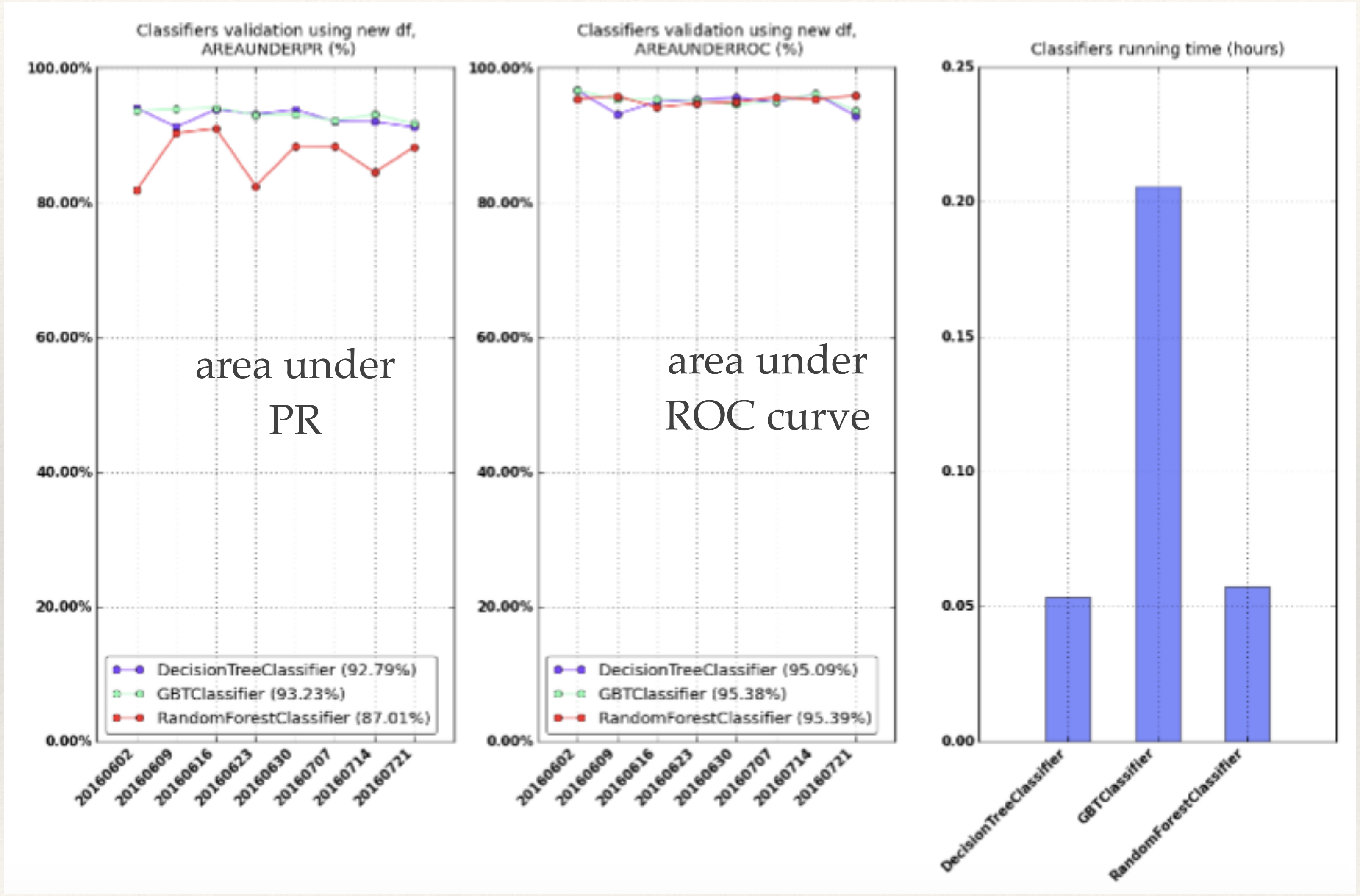
- data usage from previous weeks used to predict usage in following weeks
- several days needed for DCAF to collect up to 12 months of data
    - 6,9,12 months of data used to train classifier; we found that using 6 month is sufficient
    - predictions for up-coming week hence done using 6 months of data, then added this week into training set, re-trained classifier and extract prediction for the next following week, etc..

Decided to move the modelling part to **Apache Spark**

- exploitation of a CERN HDFS cluster
- code written to run in Spark using 3 classifiers available in *Spark+MLlib* libraries
    - RandomForest, DecisionTree, GBT

Refs: [4]

Preliminary results on Spark for **data popularity**



Difference in time spent comes from implementation and from scheduling work on worker nodes in Spark - this overhead will diminish once we will analyse larger data-frames

Refs: [4]

Another use-case:
data transfer latencies

Understanding latencies

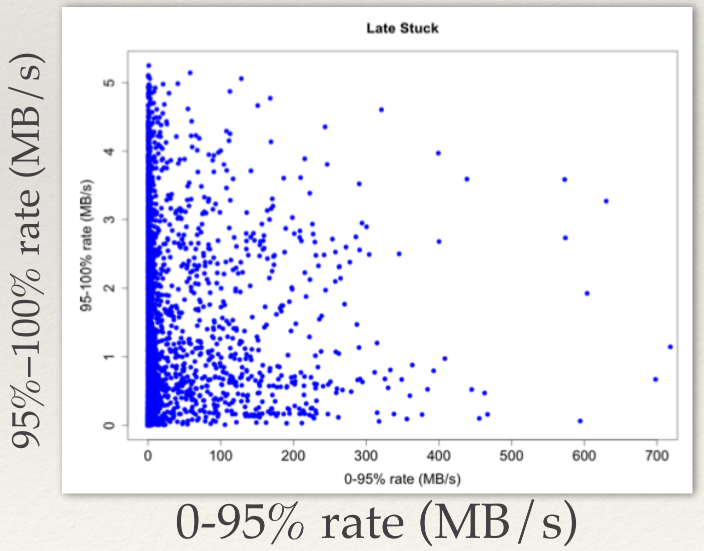
Base ingredients: logs from **PhEDEx** and **FTS**

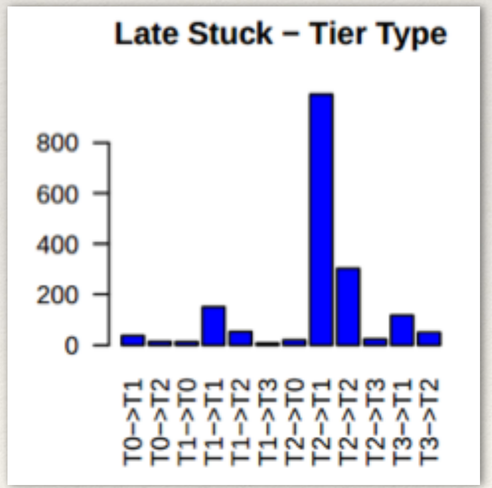
- **PhEDEx**: CMS reliable and scalable dataset replication system
- **FTS**: gLite-originated File Transfer Service used by LHC experiments

Preliminary and preparatory work since long ago (CHEP 2012..)

- CMS PhEDEx equipped to save the relevant data indefinitely (*very important!*)
- R-based study on this data to explore latency types and signatures, introduction of "skew variables" to describe them properly, etc.. Categorisation into 2 main and simple categories: "late" and "early" stuck

Example:
Late Stuck





Refs: [5,6]

Important preliminary exploration to build the next (ML-based) steps

Data preparation [1/2]

Input data needs a **transformation** into a ML suitable form

- FTS raw data collected and injected into a CERN HDFS cluster
- conversion from JSON objects in ASCII files to a flat table format (CSV)

Careful work in data **preparation** (part 1):

- JSON doc structure loosely matched, custom EOF, etc
- each record has nested records, need to be flattened
- hashing algorithms used to convert text→numerical values
- placeholder manually set for all missing attributes
- all described above: done with a script able to process large $\mathcal{O}$(GB) input files

Code is public and has been reused by similar projects in CMS

Refs: [7,8]

Data preparation [2/2]

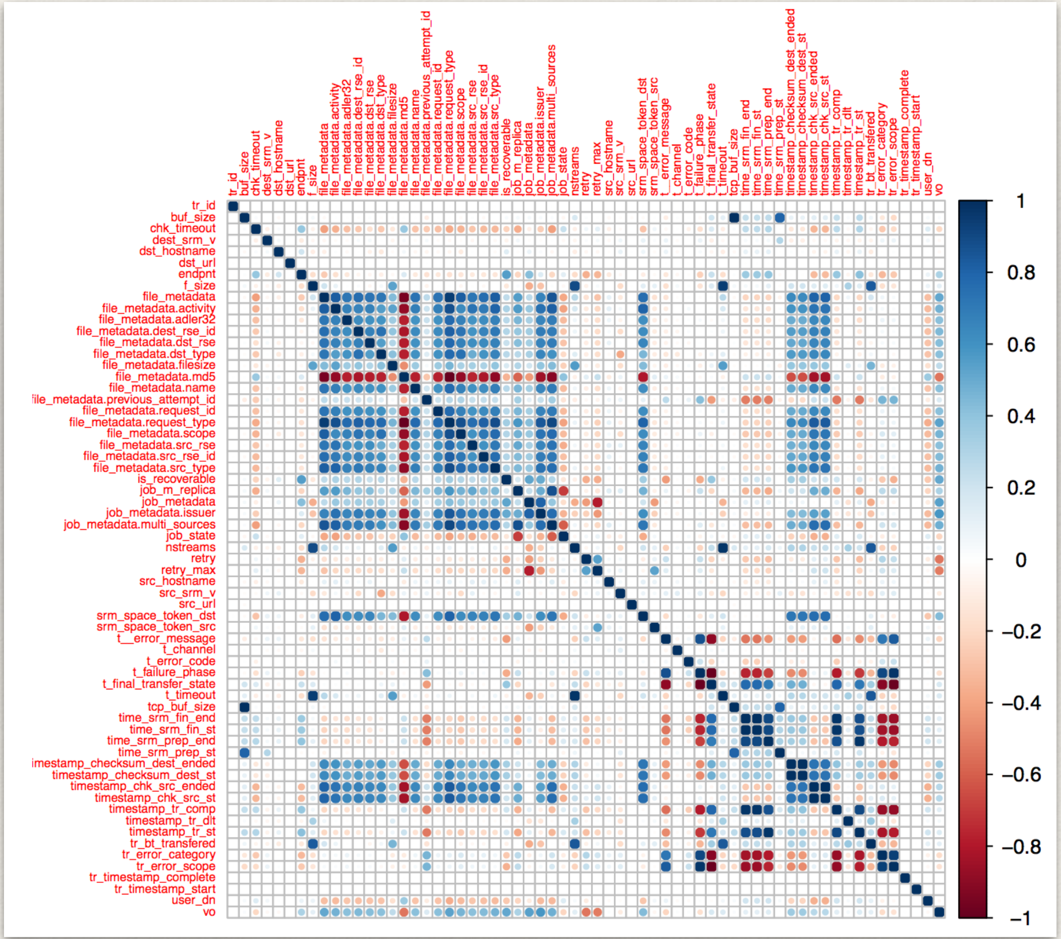
More **preparation** (part 2):

Refs: [7,8]

Data not suitable for a ML approach should be identified and dropped

- attributes regarding the end of a transfer (e.g. occurrence of a failure, or Δt of a transfer phase, ..) are unknown at the start, and can't be used for predictions
- some values are static through all datasets, i.e. uninformative and worthless, waste space and even may mislead algorithms
- obviously correlated attributes (e.g. # files vs file size)

Correlation matrixes of all FTS logs



Reduction to ~50% of the original CMS dataset

ML on **transfer latencies**

Attack the problem as a **supervised ML classification problem**, similarly to what was done for data popularity

Start at a manageable scale

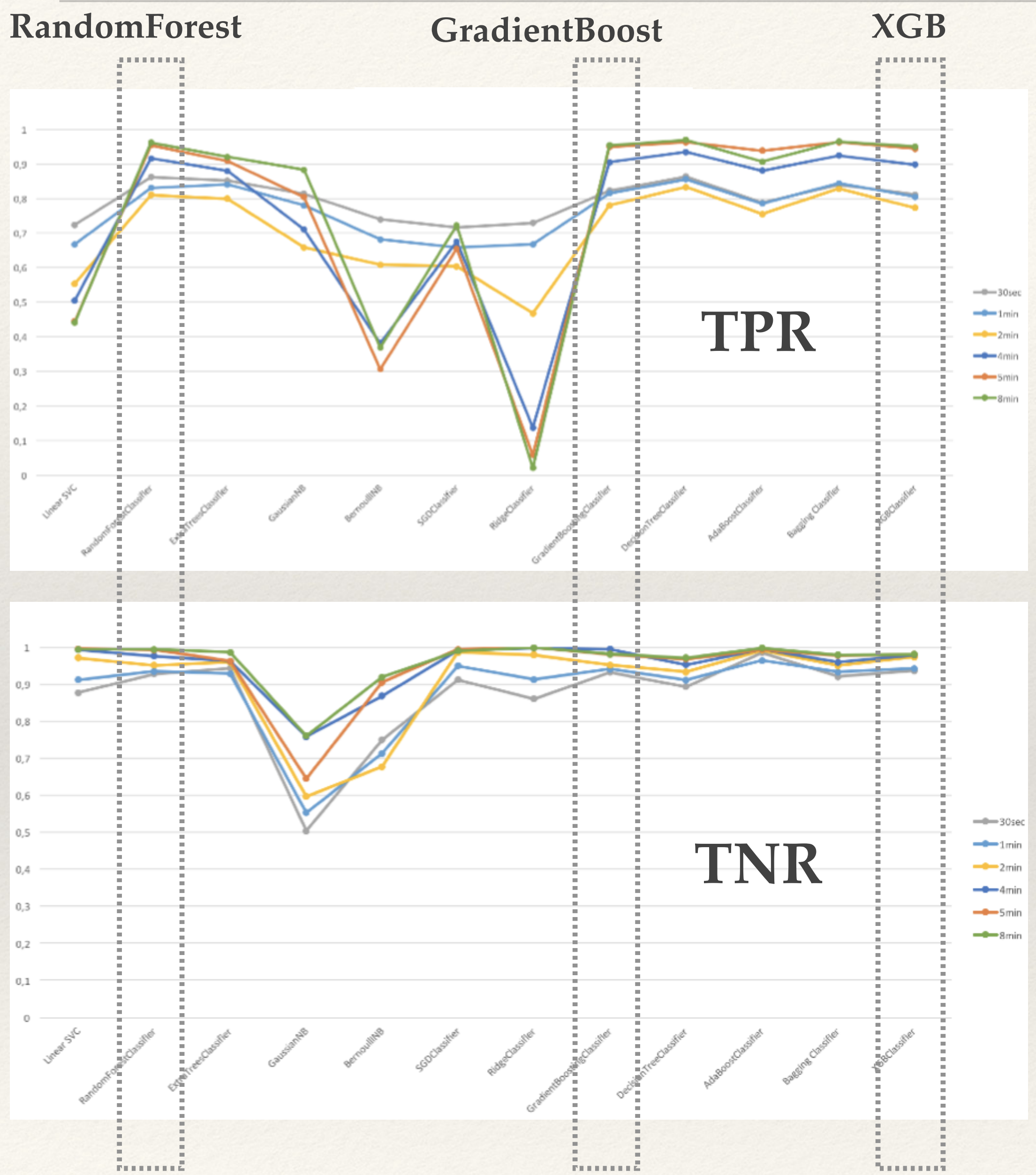
- HDFS cluster contains months of FTS logs → TBs of data
- First prototype done with just a bunch of days in July 2016 as reference

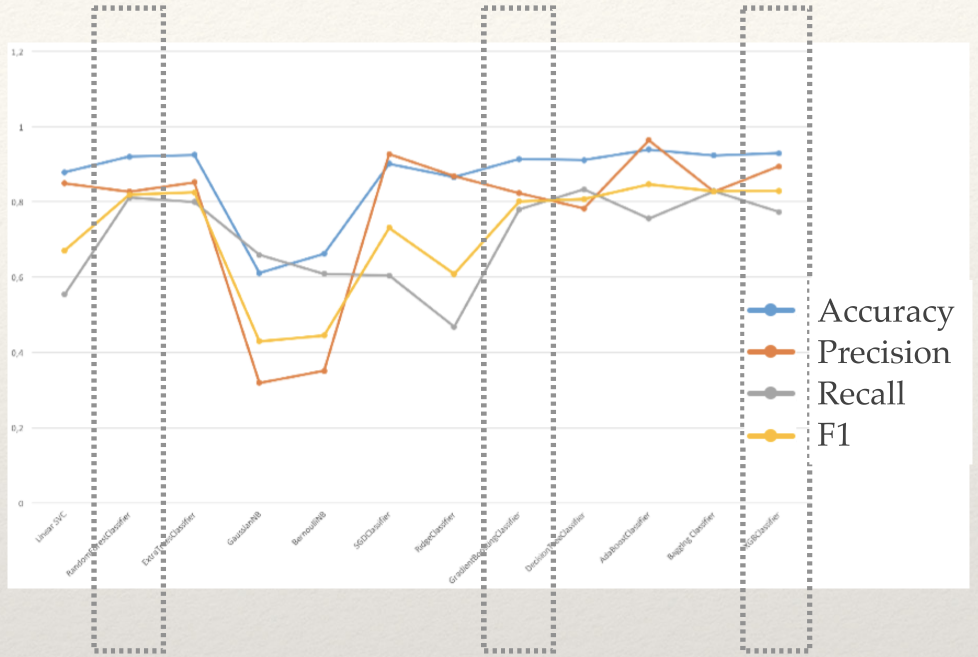
Run the ML machinery via DCAF

- Create/train a model and check predictability
  - Training / validation set at 70% / 30%
- Apply several ML algorithms.
  - Model evaluated with standard scorers (from MAE to accuracy, precision, recall, F1..)
- Some preliminary results in next slide

Refs: [7,8]

Preliminary results





Focus on: **RandomForest Regressor** and **GradientBoost Regressor** from Scikit-learn lib, **XGBRegressor** from XGBoost lib - in terms of scorers (here) but also time and memory consumption (not shown here)

Preliminary: RandomForestRegressor shows a good prediction rate overall, at a considerable cost in time and mem consumption, and may be outperformed by XGBRegressor after proper parameter tuning.. WORK IN PROGRESS

Refs: [4,7,8]

Summary

Big value in analysis CMS computing metadata with big data techniques

- experience on **dataset popularity** is now driving the work on **transfer latencies**

Status on latencies:

- data transformation DONE, feasibility study of ML techniques application DONE, algorithms comparison IN PROGRESS, next is to put it at scale, e.g. Spark jobs on the full FTS log set from HDFS

Manpower

- remarkably, most of the work is done by **students** interested in data science approaches

Value in CMS

- DCAF as an infrastructure that can be easily extended to attack new problems
  - demonstrated to be useful for two distinct use-cases
- Another building block to gain more **actionable insights from CMS computing operations meta-data**

Value beyond CMS

- Potential and easy use by other communities too
  - e.g. FTS usage is shared across experiments, could perform similar studies on other experiments' log

References

[1] https://github.com/dmwm/DMWMAnalytics/tree/master/Popularity/DCAFPilot

[2] "*Exploiting CMS data popularity to model the evolution of data management for Run-2 and beyond*", 2015 J. Phys.: Conf. Ser. 664 032003, http://iopscience.iop.org/article/10.1088/1742-6596/664/3/032003/pdf,

[3] "*Predicting dataset popularity for the CMS experiment*", Feb 2016, https://arxiv.org/pdf/1602.07226.pdf, doi:10.1088/1742-6596/762/1/012048

[4] https://github.com/dmwm/DMWMAnalytics/blob/master/Popularity/Kipras/final_report.pdf, CERN Summer Student 2016 report

[5]"*No file left behind - monitoring transfer latencies in PhEDEx*", proceedings of CHEP 2012, J. Phys.: Conf. Ser. 396 032089, http://iopscience.iop.org/article/10.1088/1742-6596/396/3/032089/pdf

[6] "*Monitoring data transfer latency in CMS computing operations*", proceedings of CHEP 2015, http://stacks.iop.org/1742-6596/664/i=3/a=032033

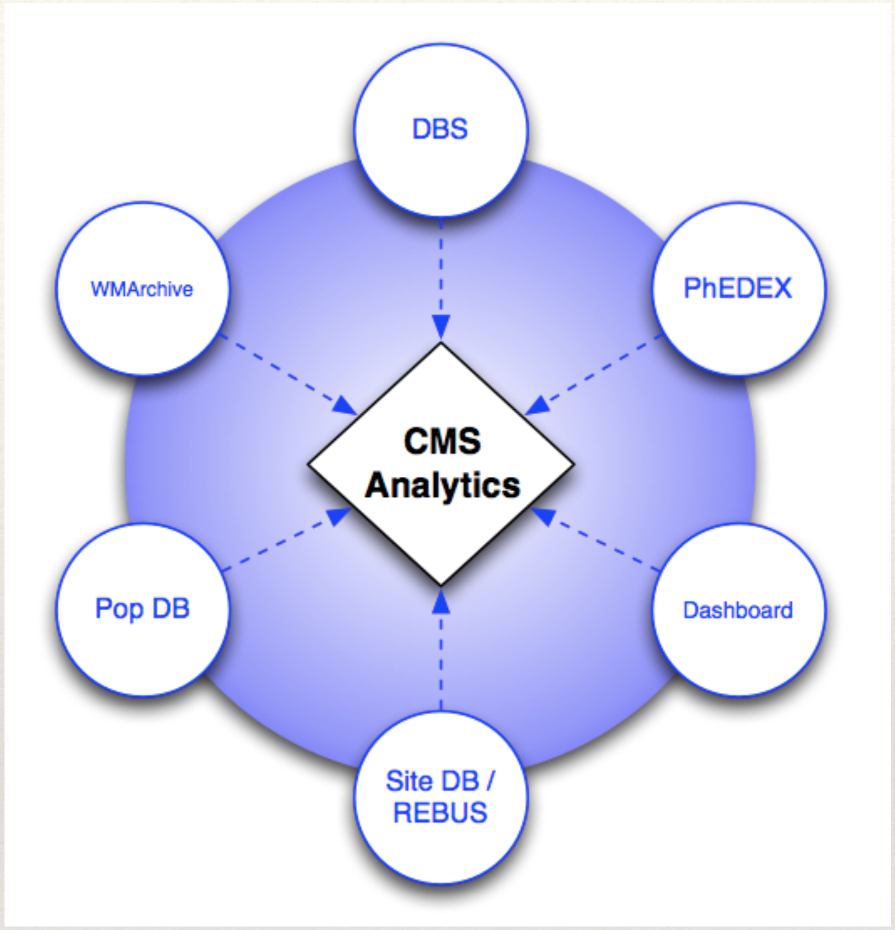
[7] https://cds.cern.ch/record/2209068/files/CERN_REPORT_Z_MATONIS%20docx%20(1).pdf, CERN Summer Student 2016 report

[8] https://cds.cern.ch/record/2218016/files/TommasoDiotalevi_report.pdf, CERN Summer Student 2016 report

For more information: vkuznet@gmail.com, daniele.bonacorsi@unibo.it

BACK-UP

CMS **Structured** data



Most ideas came out of a first CMS R&D workshop in Bologna, back in June 2014

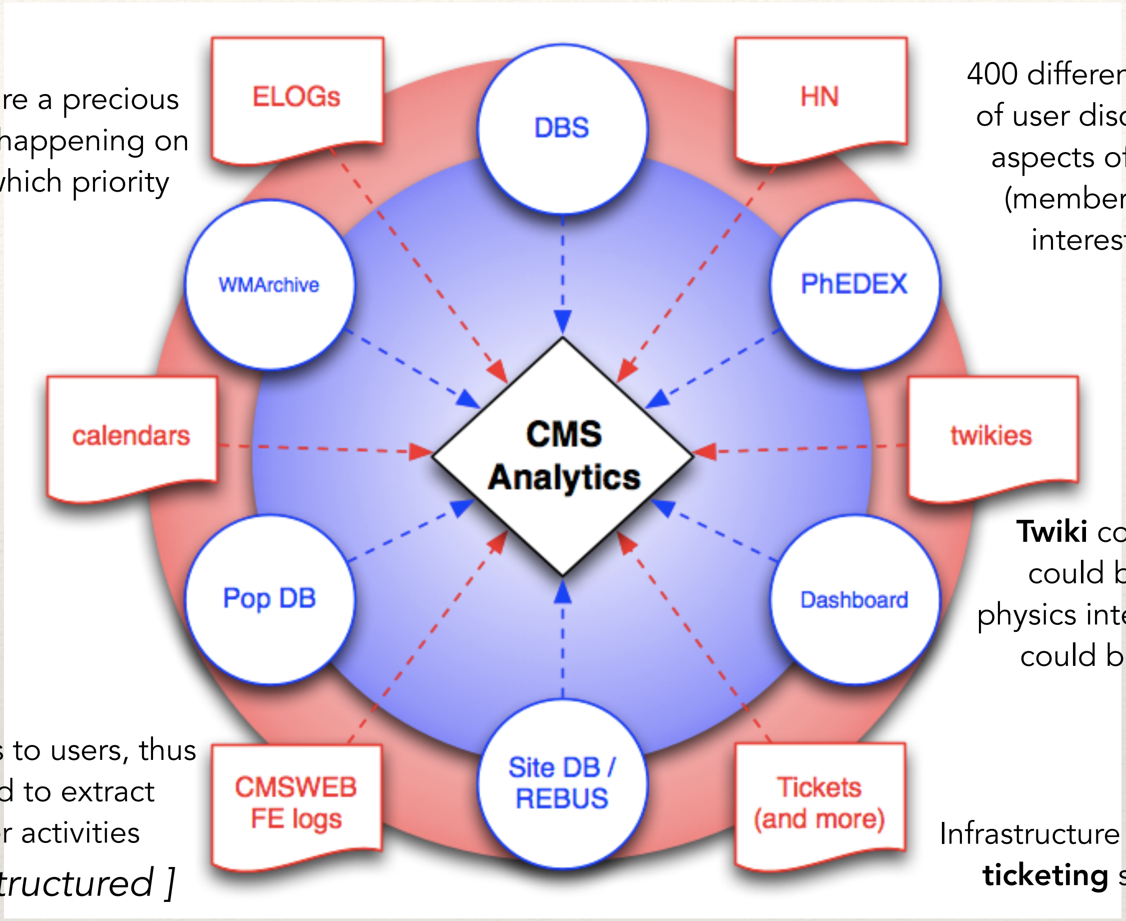
Structured info on a variety of CMS Computing activities are stored across multiple data services

- all info available via CMS data service APIs

CMS Structured and Unstructured data

Activity-based **ELOGs** are a precious source of info on what's happening on which systems and at which priority

400 different **HyperNews** fora, several yrs of user discussions, "social data-mining" aspects of collaboration-level research (membership changes study, physics interests evolution over time, …)



CMS events **calendar**, activity planning docs, list of major conferences and workshops, … could identify cycles within different physics communities

Twiki content as a knowledge graph that could be mapped to user activities and physics interests, and their evolution over time could be studied with appropriate tools.

It serves all data sources to users, thus its logs may be mined to extract valuable info on user activities

[Warning: semi-structured]

Infrastructure issues reporting/tracking, **ticketing** systems (JIRA, GGUS)..

Plenty of **unstructured** information in the CMS Computing (meta-)data ecosystem

- potentially very rich and sensitive predictors of user activities and future needs
- hard to process; manpower shortage; needs careful cost vs gain evaluation

Current focus is mostly on **structured**