Exploiting CMS data popularity to model the evolution of data management for Run-2 and beyond

V. Kuznetsov, T. Wildish, D. Giordano, N. Magini,
T. Boccali, M. Neri, M. Girone, D. Bonacorsi

April 13th, 2015

21st International Conference on Computing in High Energy and Nuclear Physics **CHEP2015** Okinawa Japan: April 13 - 17, 2015

Introduction

CMS collects data on datasets "**popularity**"

- i.e. most frequently accessed replicas
- in terms of # accesses and CPU hours used

The **data placement** is evolving towards a **less static** model

- add replicas of *existing* datasets that appear to be most popular
- remove replicas of *existing* datasets that appear to be least popular

See another CHEP'15 talk

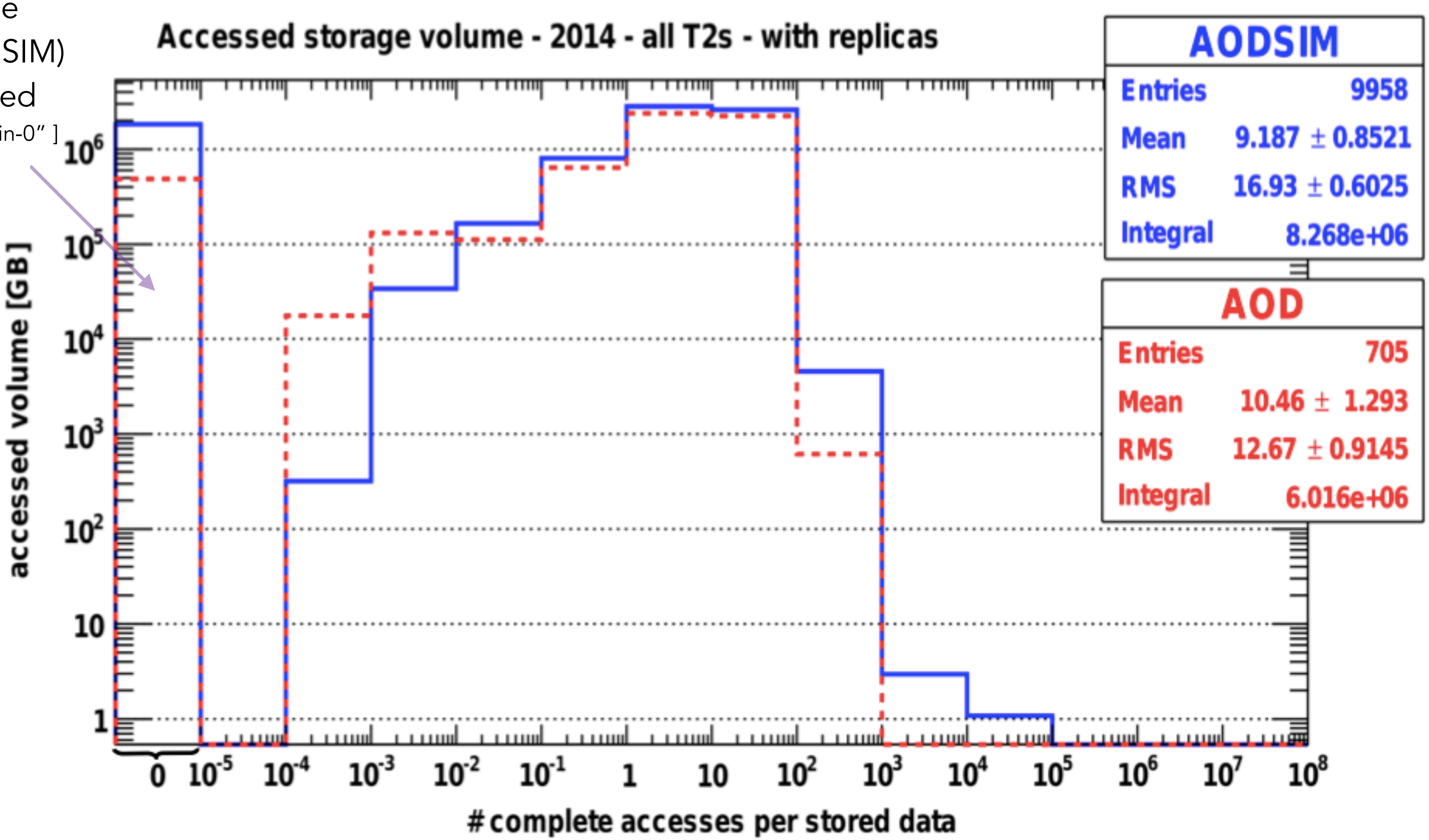
- C. Paus et al, "Dynamic Data Management for the Distributed CMS Computing System" (earlier in this same session)

We discuss here a complementary, looking-forward approach

- problem formulation: predict which datasets *will* become popular once they *will* be available on the Grid for distributed analysis

CMS data popularity in 2014

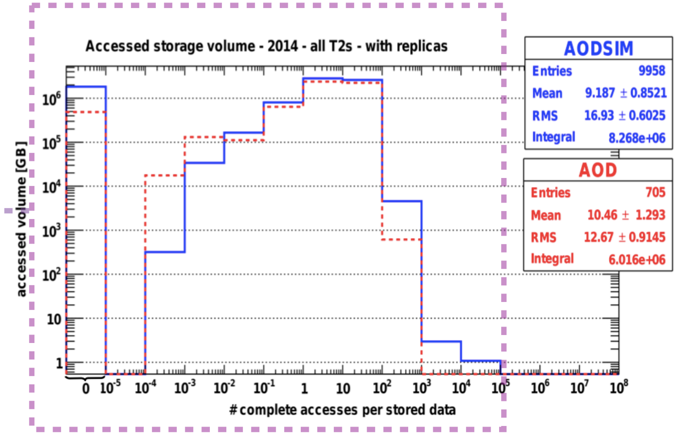
~16% (~29%) of
total storage
for AOD (AODSIM)
is not accessed
[in the following: "bin-0"]



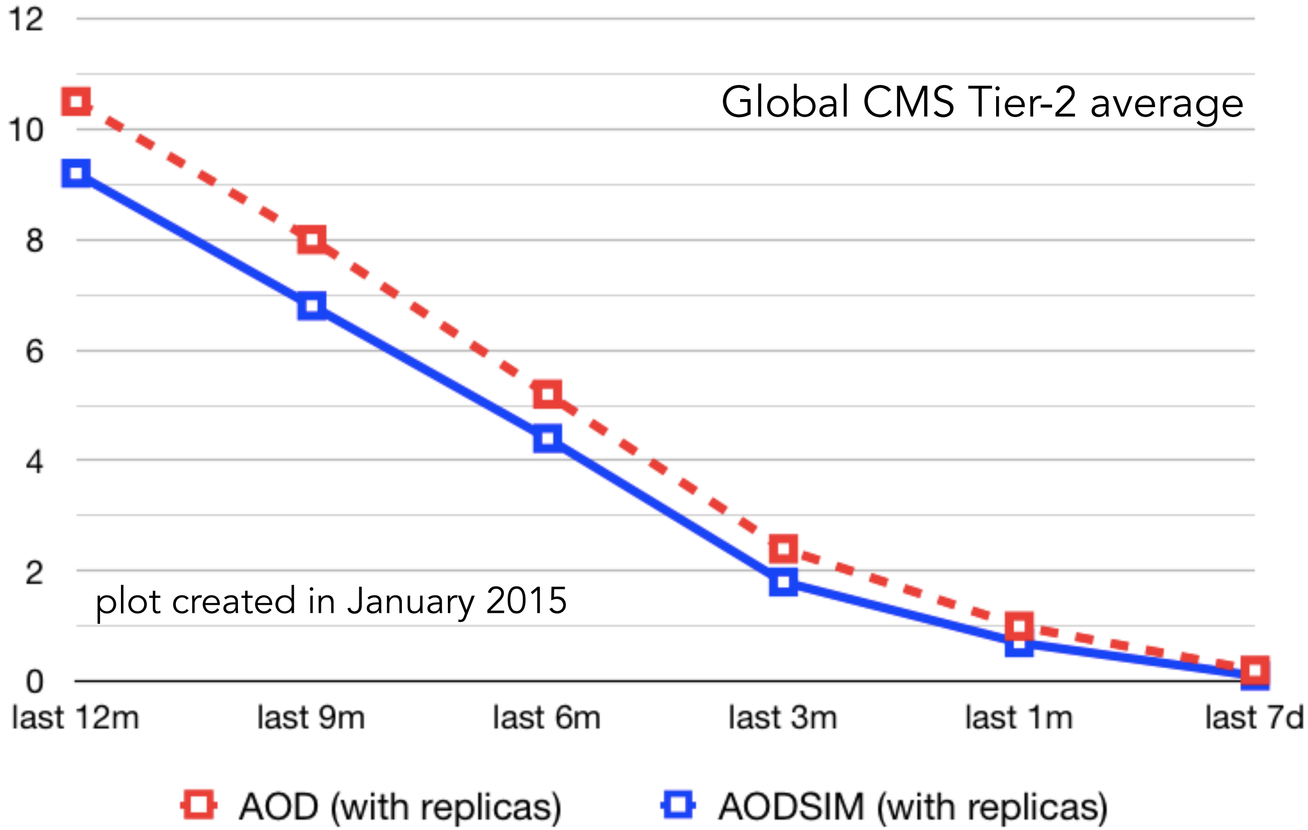
Data popularity information is rich in content and in potential correlations

- in particular the "unpopular" fraction is most interesting

A blind average



< # complete accesses per stored bytes >

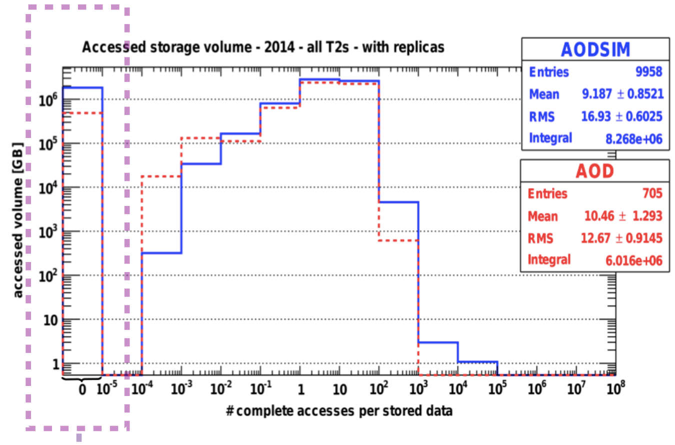


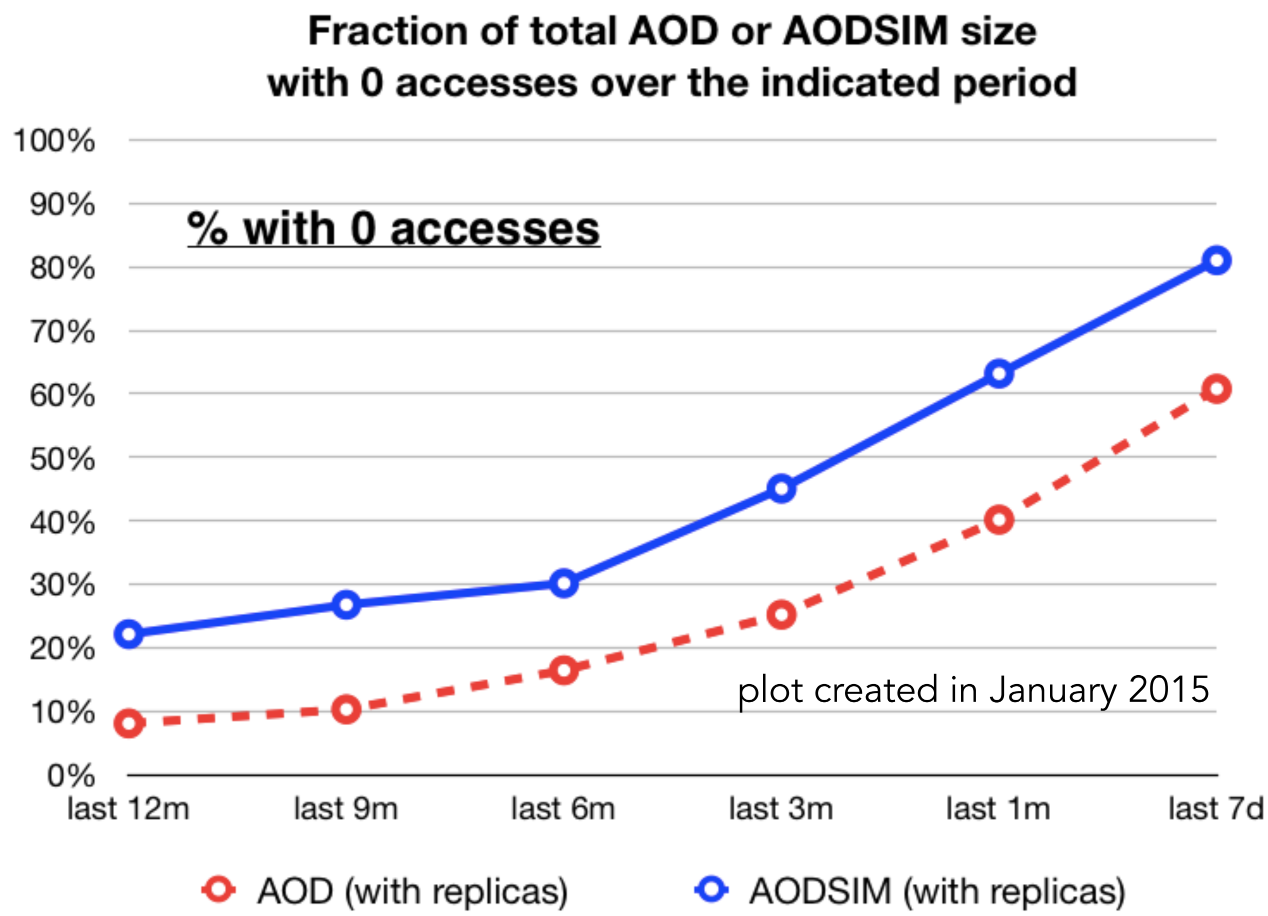
Not too bad?

- Maybe..
- But there is a "really unpopular data" bin whose content has been just averaged..
- What's in it?

Really unpopular data

[“bin-0” = data volume with 0 accesses]





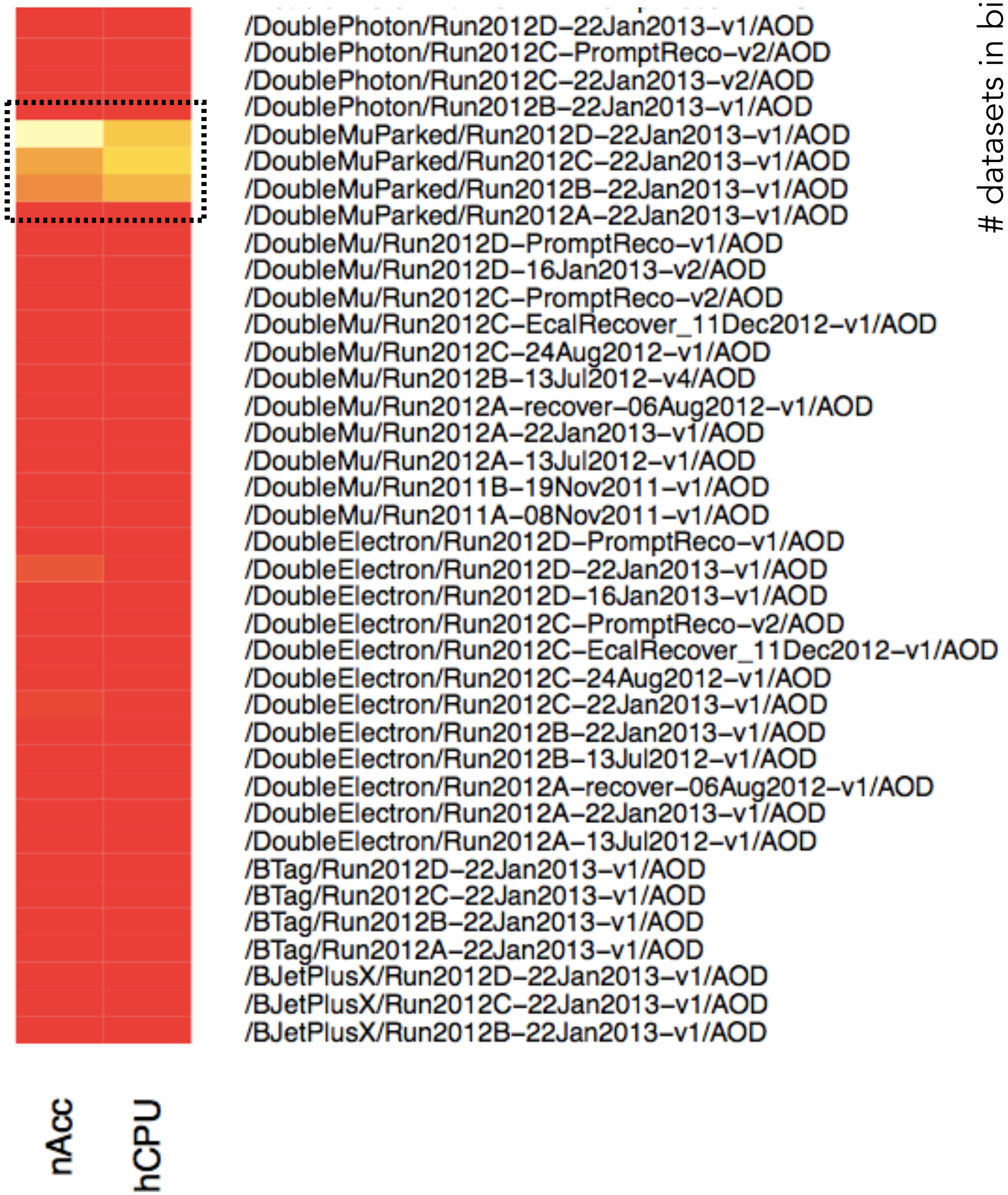
This instead has the potential to tell a story..

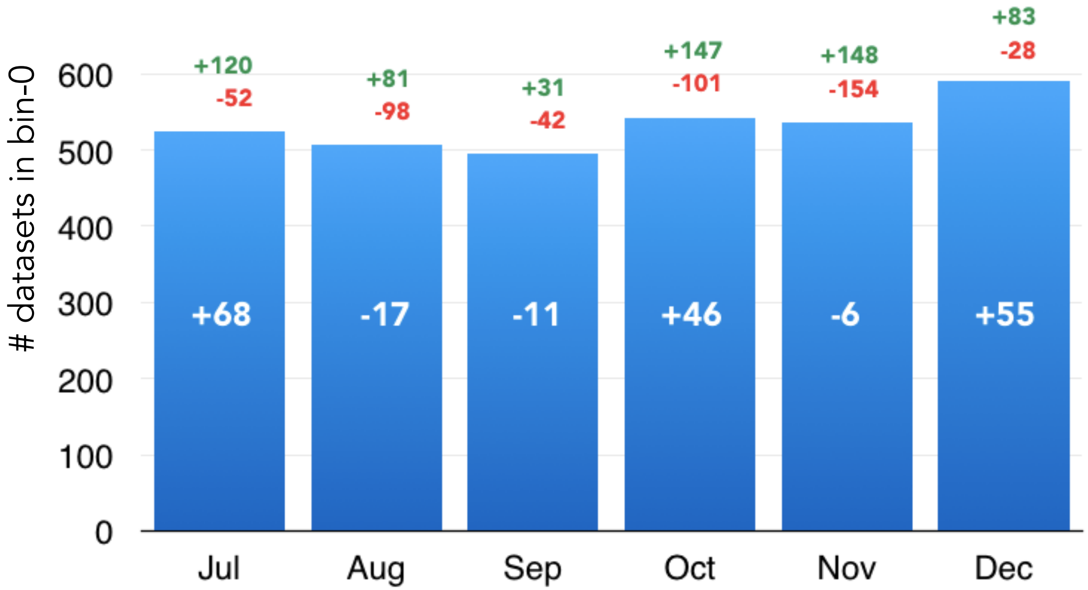
- but you need to inspect different time windows, and gain dataset-level details, and…

Dataset-oriented and incremental view

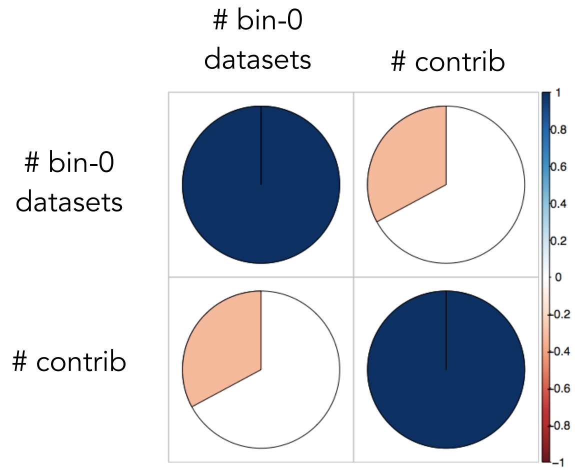
[“bin-0” = data volume with 0 accesses]

Colder datasets are easy to spot





Anti-correlation (despite tepid) between # unpopular datasets and # CMS contributions to conf/ws



What if…

All this can be done on data from the *past* to act in the *present*.

What if we could learn from the *past* and perform predictions for *future* datasets?

This e.g. will tell us which data to fill fast caches in front of disk systems with

- in computing system in O(5-10) yrs from now

Plenty of data from computing operations in Run-1 and LS1 are available.

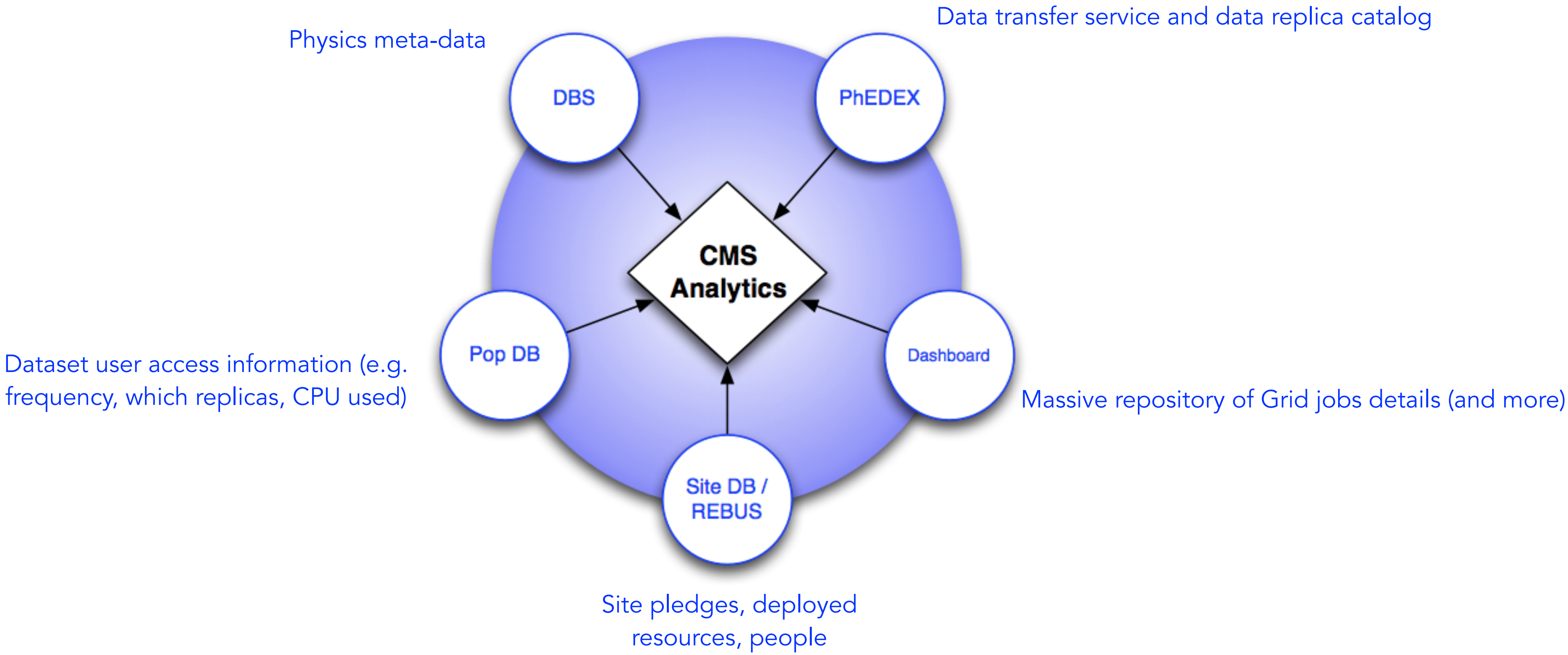
This data is all **archived**, but **rarely** (or never) **accessed** by anyone

- e.g. transfers, job submissions, site performances, releases logs, analysis performances,
  - PhEDEx DB, WMAgent logs, Dashboard, SiteDB, SSB, etc…
- we basically monitor to debug in near-time, not to analyse what happened in the past
- we never fixed holes in monitoring data, never validated (most of) them with a decent care
  - not polished and not complete/coherent ⟹ not explorable ⟹ not exploitable in its current form

Variety (and veracity) are the Big Data V's that matter most here

- (Volume not negligible, but manageable - Velocity: real-time is not a must)
- Variety: very irregular data set: structured, semi-structure and unstructured data
- Veracity: data integrity and the ability to trust them to make decisions is important

Structured data...



Structured data on a variety of CMS Computing activities

- stored across multiple data services, available via data service APIs

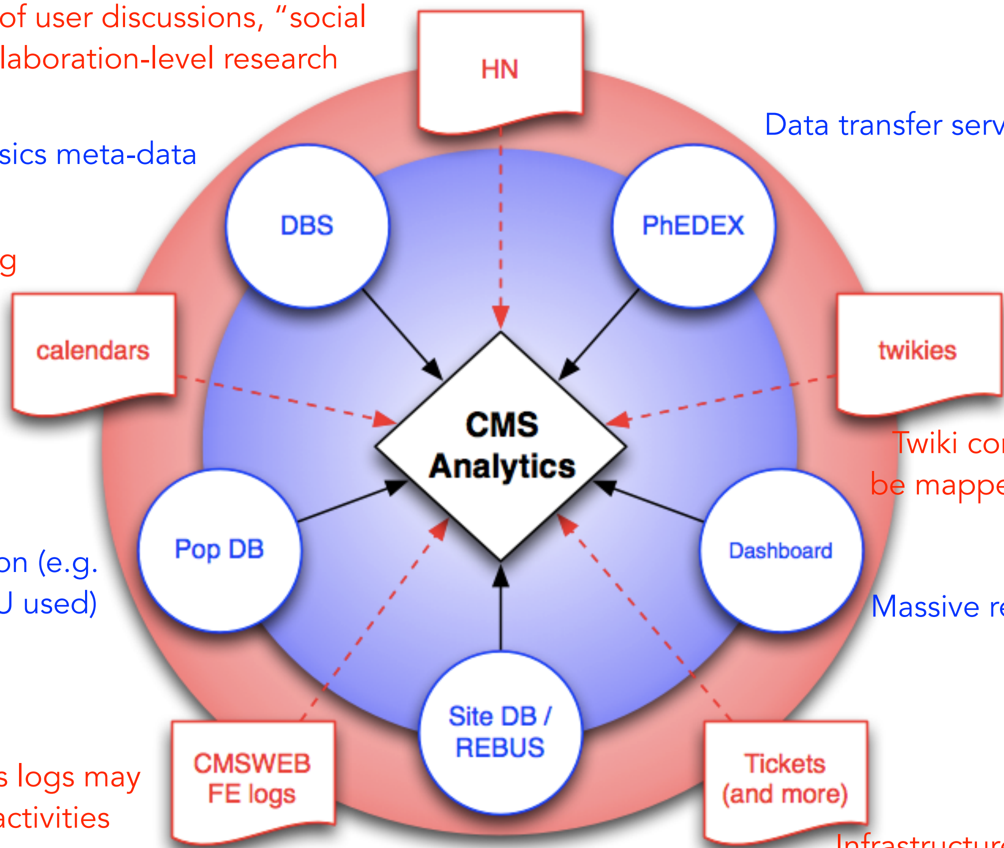
Structured and **Unstructured** data

400 different fora, several yrs of user discussions, "social data-mining" aspects of collaboration-level research

Physics meta-data

Data transfer service and data replica catalog

CMS events calendar, activity planning docs, list of major conferences, etc



Twiki content as a knowledge graph that could be mapped to user activities and physics interests

Dataset user access information (e.g. frequency, which replicas, CPU used)

Massive repository of Grid jobs details (and more)

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

[semi-structured]

Site pledges, deployed resources, people

Infrastructure issues reporting/tracking, Savannah, GGUS, activity-based ELOGs, topical e-groups, ..

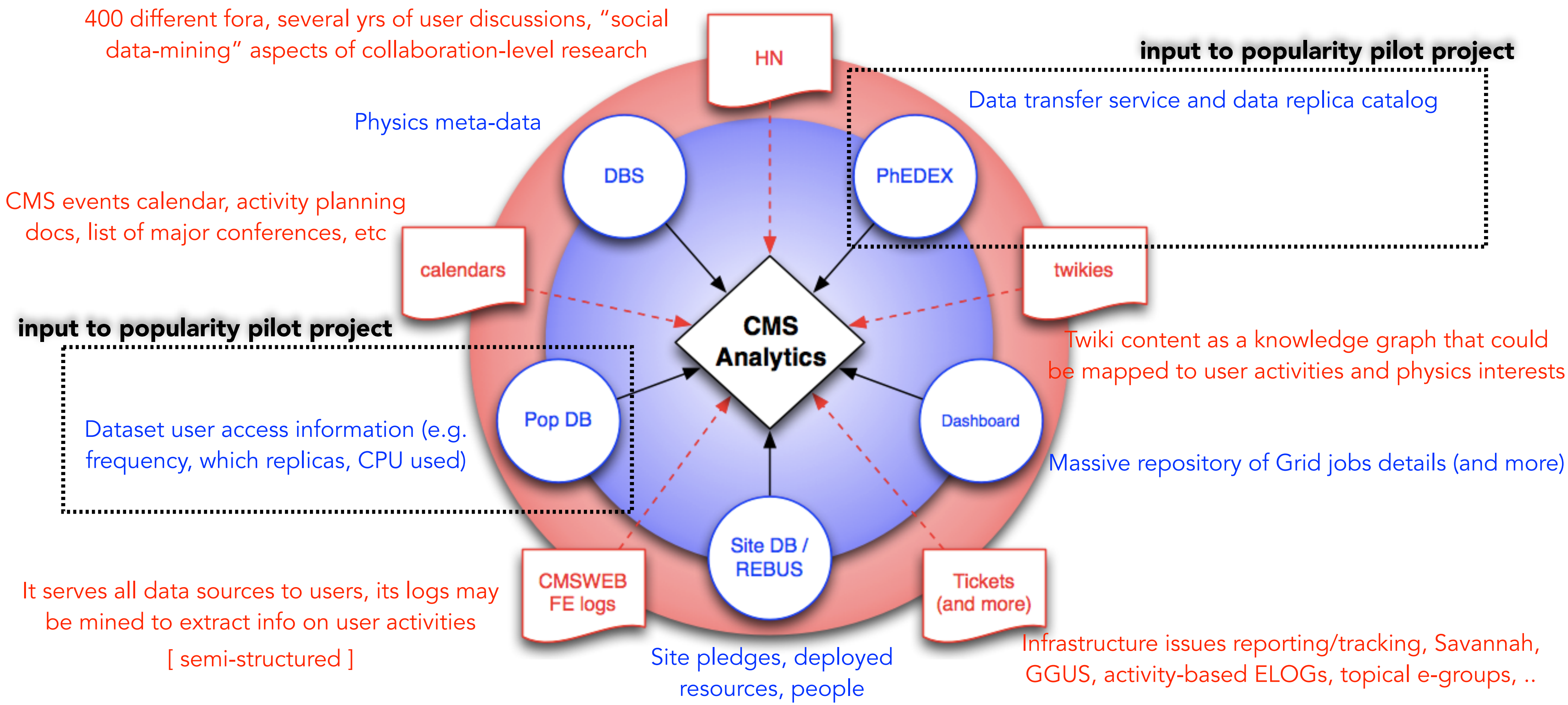
Structured data on a variety of CMS Computing activities

- stored across multiple data services, available via data service APIs

Plenty of **unstructured** information in the CMS Computing ecosystem

- hard to process but very diverse and potentially very rich!

Structured and Unstructured data



Structured data on a variety of CMS Computing activities

- stored across multiple data services, available via data service APIs

Plenty of **unstructured** information in the CMS Computing ecosystem

- hard to process but very diverse and potentially very rich!

CMS Analytics

Long-term goal (2-3 years)

- build **adaptive data-driven models of CMS {Data/Workflow} Management**
- **make predictions**: predict future behaviours from measurements of past performances

Short-term goal (in Run-2)

- support **CMS Computing operations**
  - e.g. **improvements in the use of computing resources**

How?

- **deeper understanding of CMS "data" from computing operations** in Run-1/LS1
  - a by-product , but it has a huge value in itself

Why **adaptive** modelling?

- models of the past aren't going to apply to the future for long..
- only adaptive modelling will give us confidence and predictive power in the long term

Bottom-up approach in selecting the **pilot projects**

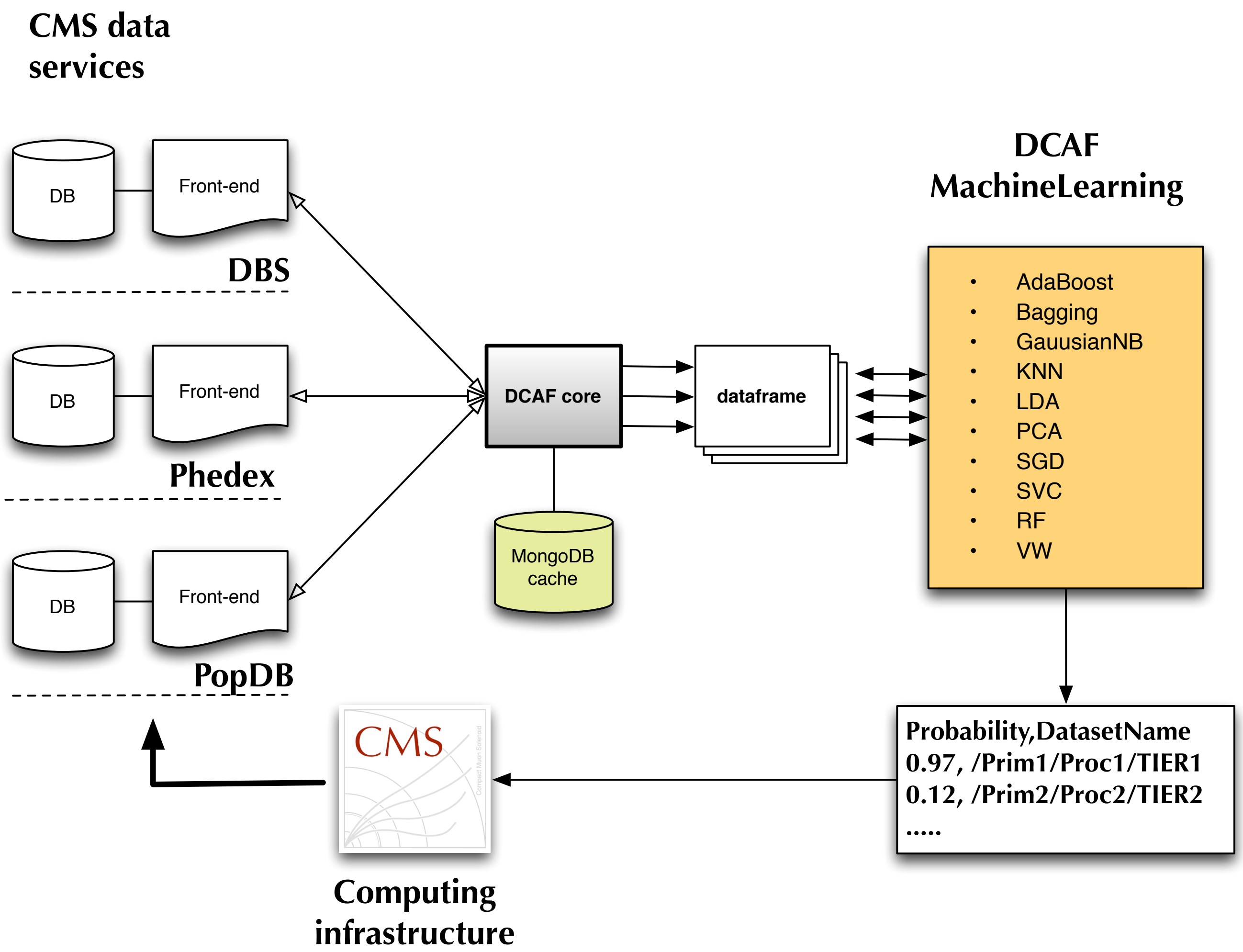
- focus on clear problem(s) formulation
- well-defined, self-contained, independent pilot projects

One example:

Predict popularity of new datasets

DCAFPilot (Data and Computing Analysis Framework)

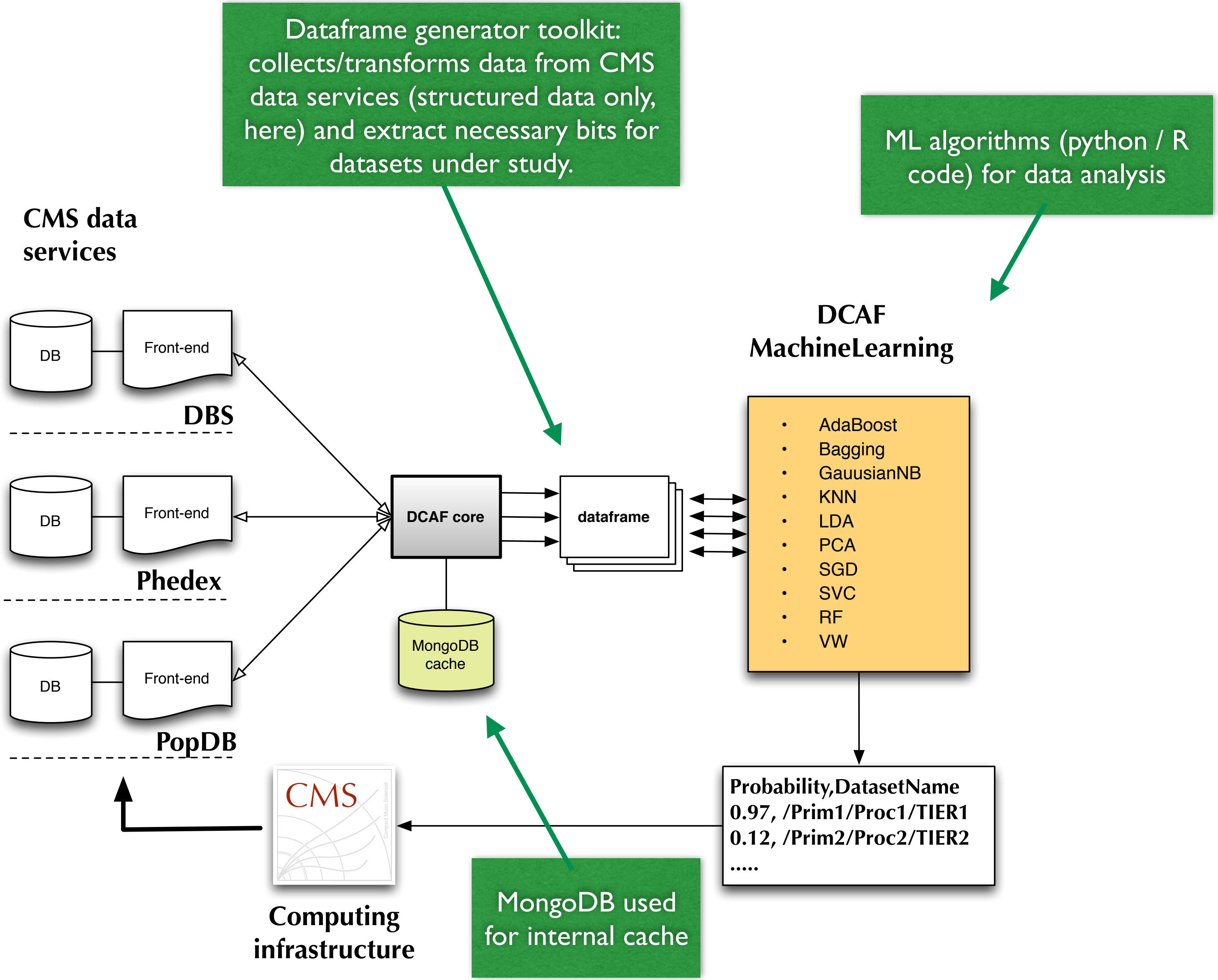
- a pilot project to understand metrics, analysis workflow, necessary functionalities (and possible technology choices) of the machinery needed to attack this problem



One example:

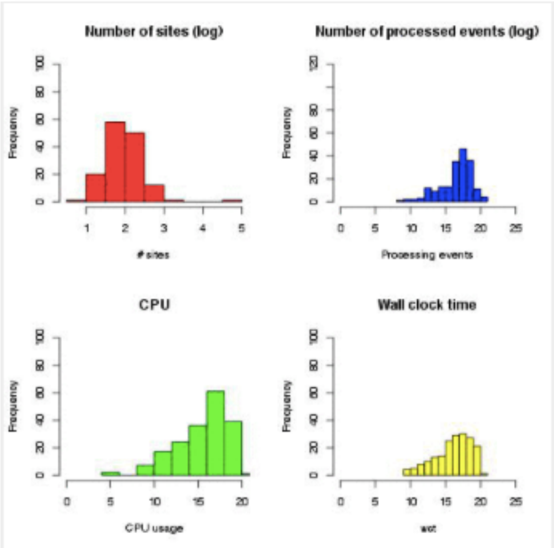
Predict popularity of new datasets

DCAFPilot

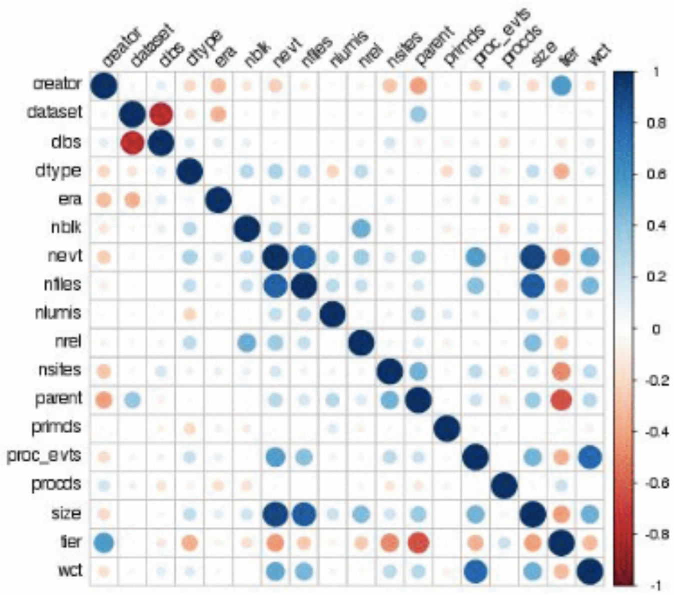


Visualise the data-frame

Live data



Correlations



Different dataset popularity metrics

A. by # users accessing it

B. by total CPU used to process it

A

B

From data collection to prediction

1. Data collection
2. Data transformation into suitable format for ML
3. The ML model:
   - use classification or regression techniques
     - the former allows to predict real values of metrics (e.g. # accesses)
     - the latter allows only to classify into categories (e.g. popular or unpopular)
   - train and validate your ML model
     - split data into train and validation sets
     - ~600K rows in the 2014 dataset: Jan-Nov used as a train set, Dec used as the validation set
     - estimate your predictive power on the validation set
4. Generate new data and transform it (similar to step #2)
5. Apply your best model to new data to make prediction
6. Verify prediction with PopDB once metrics become available

DCAFPilot in numbers

Some figures from a dry run of the machinery:

Data collection:

- Queried 5 data services (4 DBS instances used), 10 APIs used
- Internal cache fed with ~220k datasets, ~900 release names, 500+ SiteDB entries, 5k people's DNs
- ~800k queries placed overall
- Anonymisation and factorisation via internal cache

Data frame:

- constructed out of 78 variables, made of 52 data-frame files, ~600k rows
- each file is worth 1 week of CMS meta-data (~600kB gzipped) and has ~1k popular datasets with a ~1:10 ratio of popular vs unpopular randomly mixed)

Preliminary observations

statistical variables:
accuracy, **precision**, **recall** and **F1** scorers

scikit-learn classifiers (python)

online-learning algorithm by Yahoo

eXtreme Gradient Boosting, a parallel gradient boosting tree solution

Classifier	# accesses > 10			
	accuracy	precision	recall	F1
Random Forest	0.98	0.86	0.98	0.92
SGDClassifier	0.96	0.98	0.62	0.76
Linear SVC	0.95	0.68	1.00	0.81
Vowpal Wabbit	0.96	0.98	0.69	0.74
xgboost	0.98	0.82	0.98	0.90

A good start!

DCAFPilot stands as a good proof-of-concept

- Caution in drawing conclusion is a must, of course
- Plenty of work to do (avoid known ML obstacles, work on defining metrics, etc)

More projects are being cooked under similar approaches..

- **Popularity** is just one starting example
- Other aspects of CMS Computing may benefit from **analytics** approaches
- Updates at CHEP'16?

Remembering the goal:

- short-term: understand our "data" from Run-1/LS1 and improving our way to do computing operations and use distributed resources
- long-term: a **data-driven adaptive model of CMS Computing**