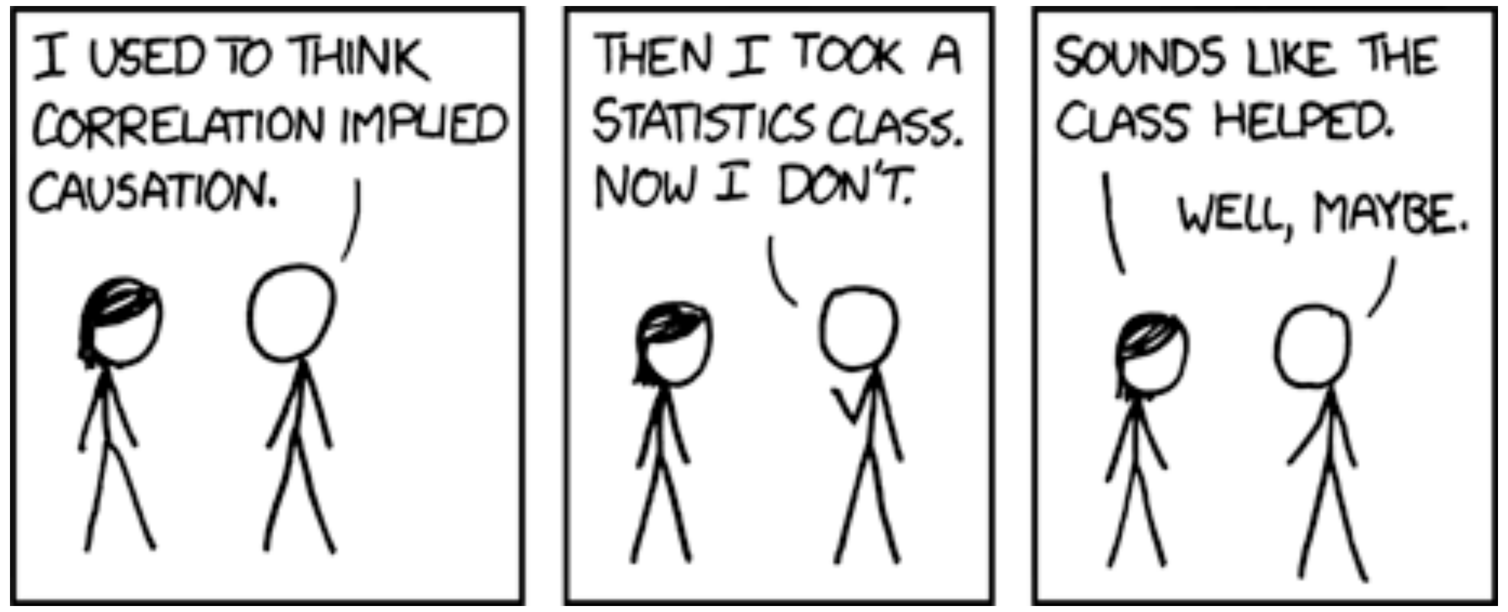
Infrastructure Analytics and Optimisation

Dirk Duellmann, IT-ST

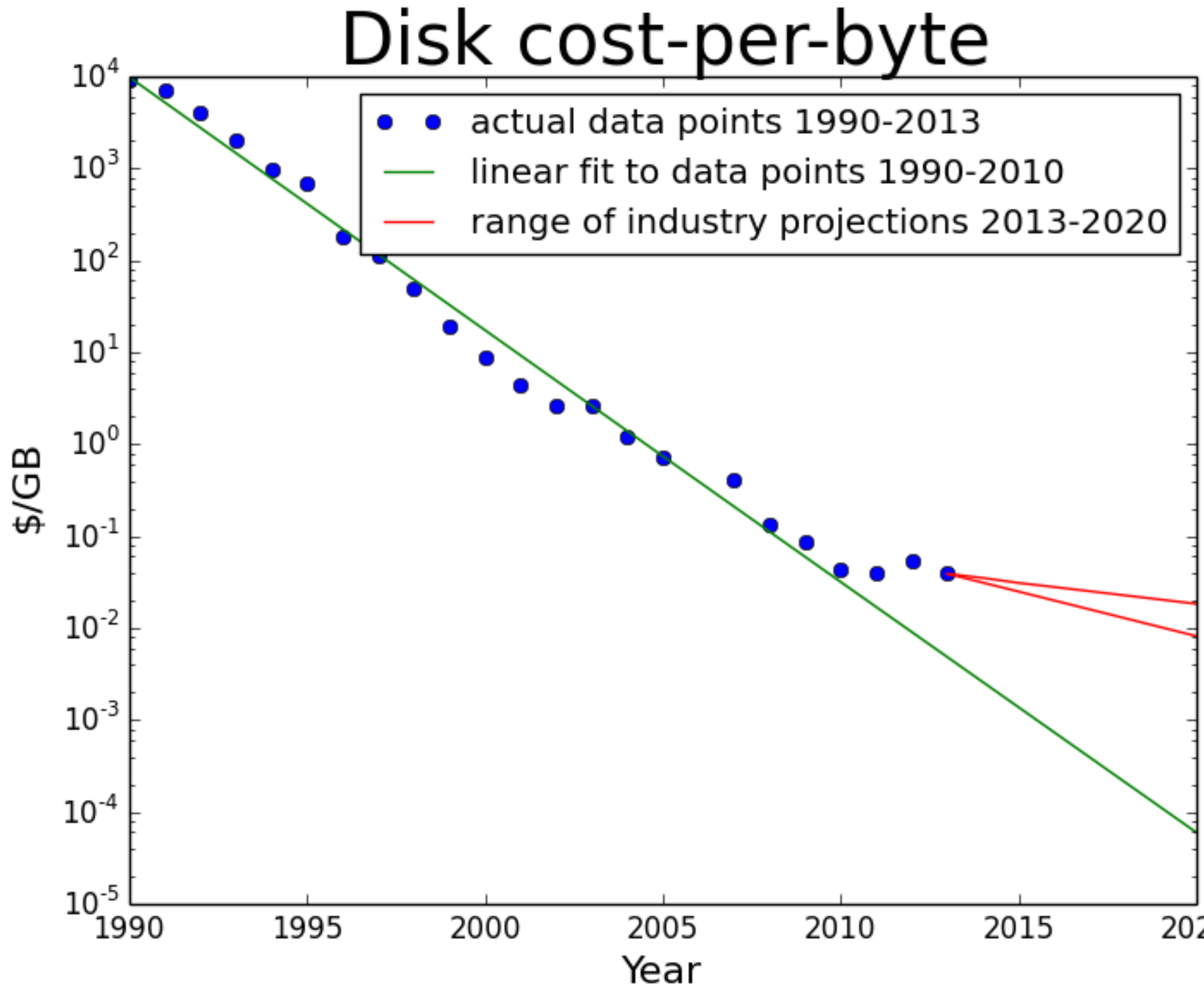
Infrastructure Analytics

- Goal: end-to-end optimisation
  - increase experiment visible computing outcome per resource investment
- Infrastructure
  - eg the CERN IT computing infrastructure spread across individual services and sites
- Analytics
  - := analysis, but somehow without all the effort and frustration…



Why the effort?

- LHC performance is excellent and increasing
- Budgets are expected to stay constant (at best)
    - Moore's and Kryder's "law" are slowing down
    - Several disruptive changes ahead
      (eg commercial clouds, disk->flash->NV memory)
- experiments and IT are increasingly accountable
  -> funding bodies: throughput per investment?
    - quantitatively instead of just qualitatively
    - absolute (not just relative) numbers



How? Analysis Steps

- **Measuring**
  - a lemon entry, LSF job, EOS file access
- **Collecting**
  - for all boxes, jobs, disk servers
- **Monitoring**
  - display timelines and simple aggregates

- **Accounting**
  - authoritative sums (consolidated, corrected, x-checked)
- **Correlation**
  - Is CPU utilisation impacted by lack of I/O ops? network latency? local RAM?
- **Modelling / Prediction**
  - “black box” that calculates a *similar* outcome as the real system from a given set of input parameters
- **Evaluation / Conclusion**
  - decide&validate an actual change!

Similarities with physics chain?

- **Measuring <-> DAQ**
  - a lemon entry, LSF job, EOS file access
- **Collecting <-> Event Building**
  - for all boxes, jobs, disk servers
- **Monitoring <-> Online Event Display**
  - display timelines and simple aggregates
- **Accounting <-> Calibration**
  - authoritative sums (consolidated, corrected, x-checked)
- **Correlation <-> Reconstruction**
  - Is CPU utilisation impacted by lack of I/O ops? network latency? local RAM?
- **Modeling / Prediction <-> Simulation**
  - “black box” that calculates a *similar* outcome as the real system from a given set of input parameters
    - similar := close enough to answer a service question
    - useful := less effort than just “trying” the real thing
- **Evaluation / Conclusion**
  - decide&validate an actual change

Input Data from IT monitoring project

Subsystem	Location	Amount	
lemon	hdfs	78 TB	box level
squid	hdfs	110 GB	http cache access
openstack	hdfs	12 TB	agile infrastructure
syslog	hdfs	23 TB	unstructured box logs
eos	hdfs	12 TB	file access metrics
castor	hdfs	55 TB	tape archive access
LANdb	hdfs	small O(100 MB)	host,ip,hypervisor, location
perfsonar	hdfs	small O(10 GB)	network link status
exp. dashboard	hdfs	small (< 1TB)	job summaries
exp. file popularity	hdfs	small O(200GB)	user data access
batch	hdfs	500 GB	accounting & queue-config
hw specs	afs	100MB	h/w rating per model

(6/2016)

Data vs Information

- In contrast to physics data analysis
  - often unstructured
  - no up-front, designed data model
- Medium volume
  - usually several tens of TB per analysis dataset
  - not “Big Data”, but processing times can be large enough (hours to days) to disrupt interactive analysis
- Prepare data extracts suitable for analysts
  - keep people focussed on understanding the data
  - … not just on waiting for batch jobs to select data

COMMUNICATIONS OF THE ACM

CERN Library

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE

Home / Blogs / BLOG@CACM / Why the 'Data Lake' is Really a 'Data Swamp' / Full Text

BLOG@CACM

Why the 'Data Lake' is Really a 'Data Swamp'

By Michael Stonebraker
December 22, 2014
Comments (2)

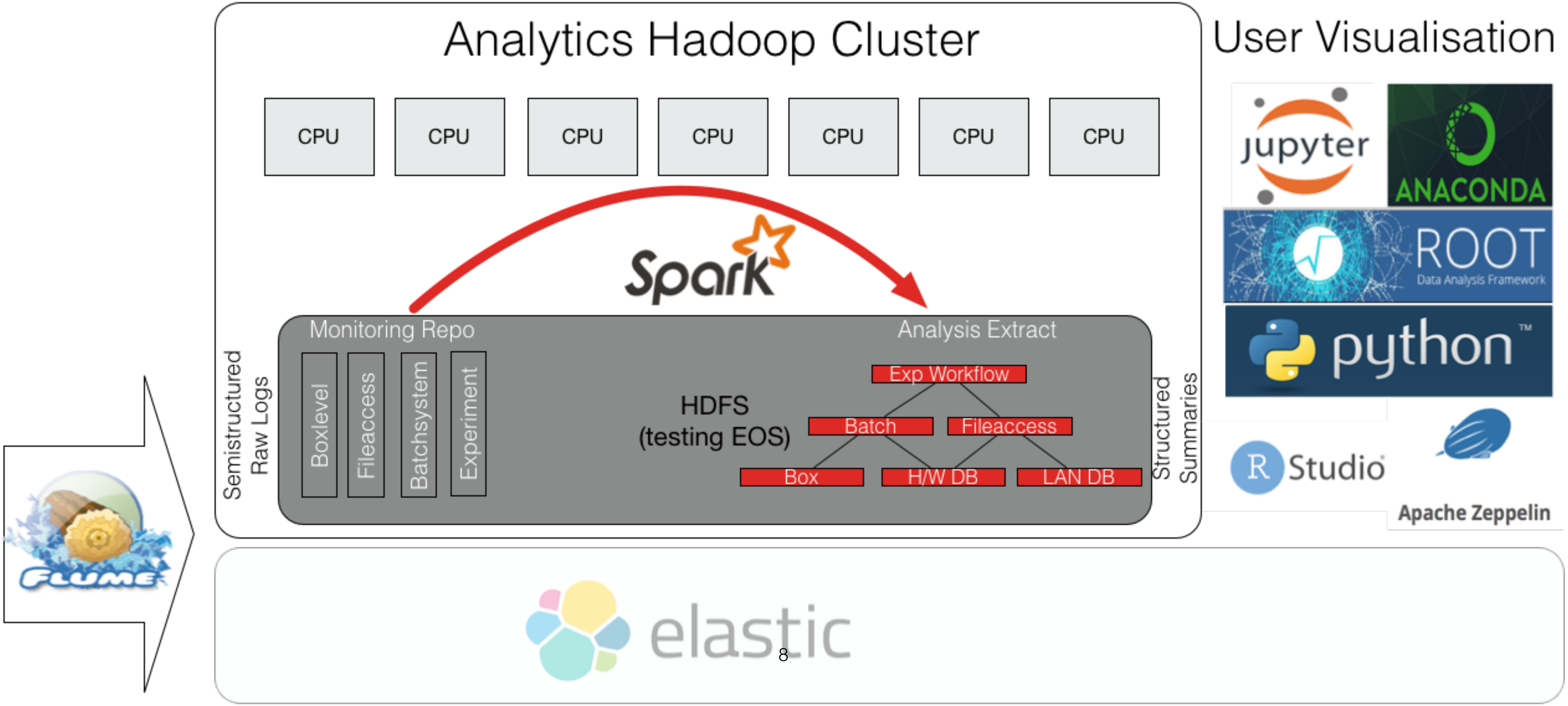
VIEW AS: SHARE:

A popular refrain I hear these days is "I am planning to put all of my data into a data lake so my employees can do analytics over this potential treasure trove of information." This point of view is also touted by several vendors selling products in the Hadoop ecosystem. Unfortunately, it has a serious flaw, which I illustrate in this posting using (mostly fake) data on one of my M.I.T. colleagues.

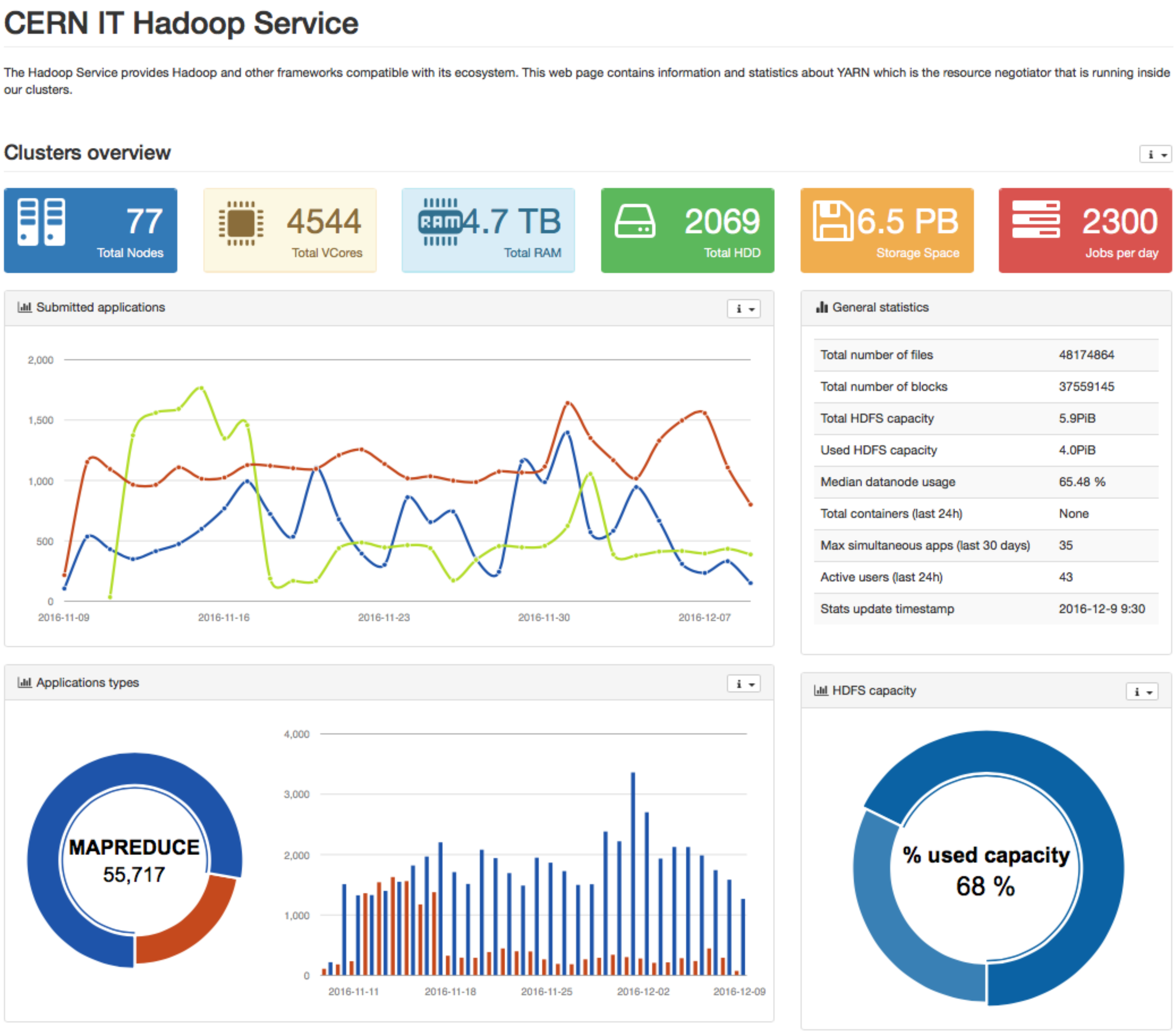
Consider two very simplistic data sources containing data on employees. The first data source has record of the form:

Employee (name, salary, hobbies, age, city, state)

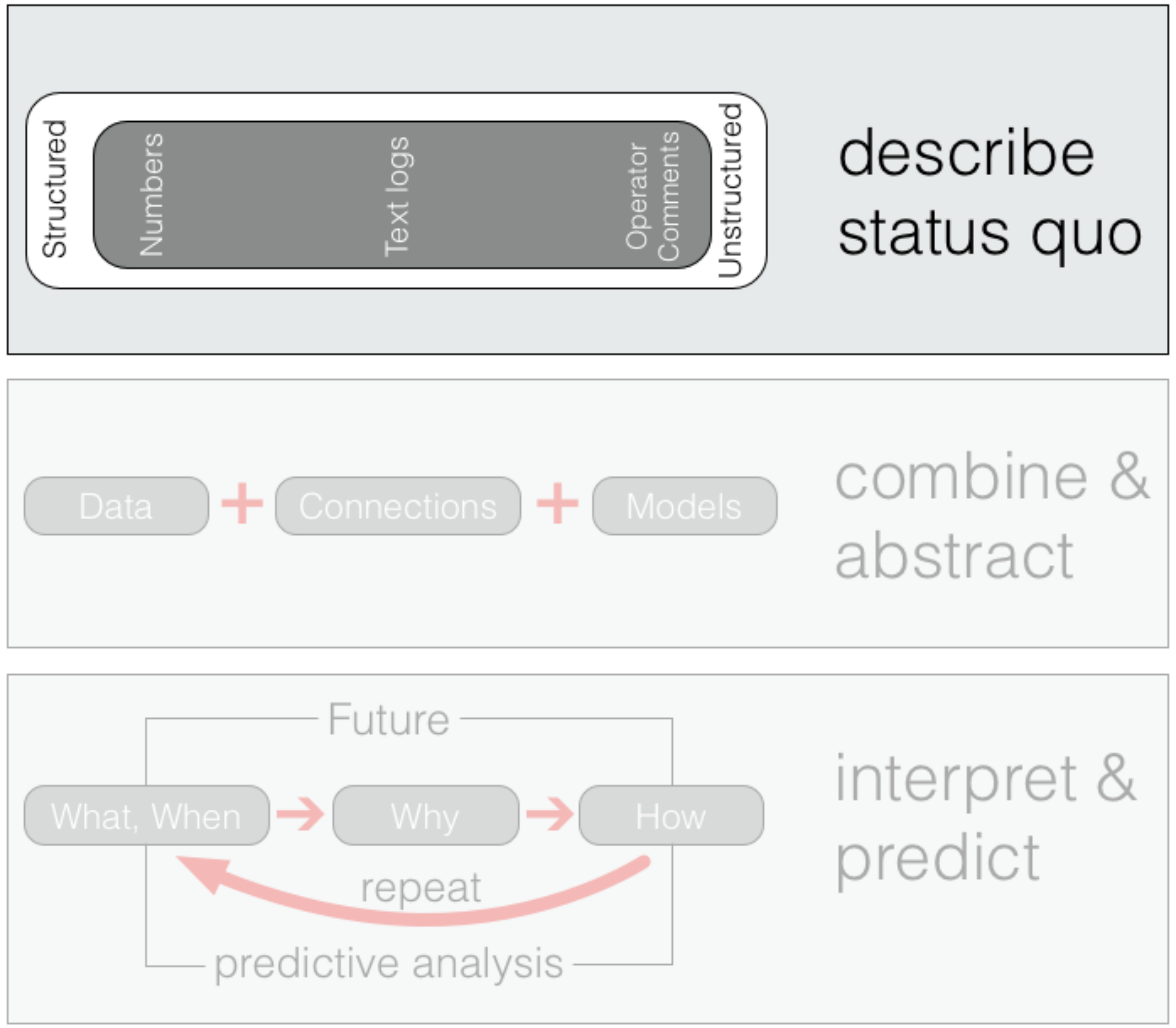
while the second contains data with a layout of:



Analytics Hadoop Cluster
CPU
CPU
CPU
CPU
CPU
CPU
CPU
Spark
Monitoring Repo
Analysis Extract
Semistructured
Raw Logs
Boxlevel
Fileaccess
Batchsystem
Experiment
HDFS
(testing EOS)
Exp Workflow
Batch
Fileaccess
Box
H/W DB
LAN DB
Structured
Summaries
User Visualisation
jupyter
ANACONDA
ROOT
Data Analysis Framework
python
R Studio
Apache Zeppelin
FLUME
elastic

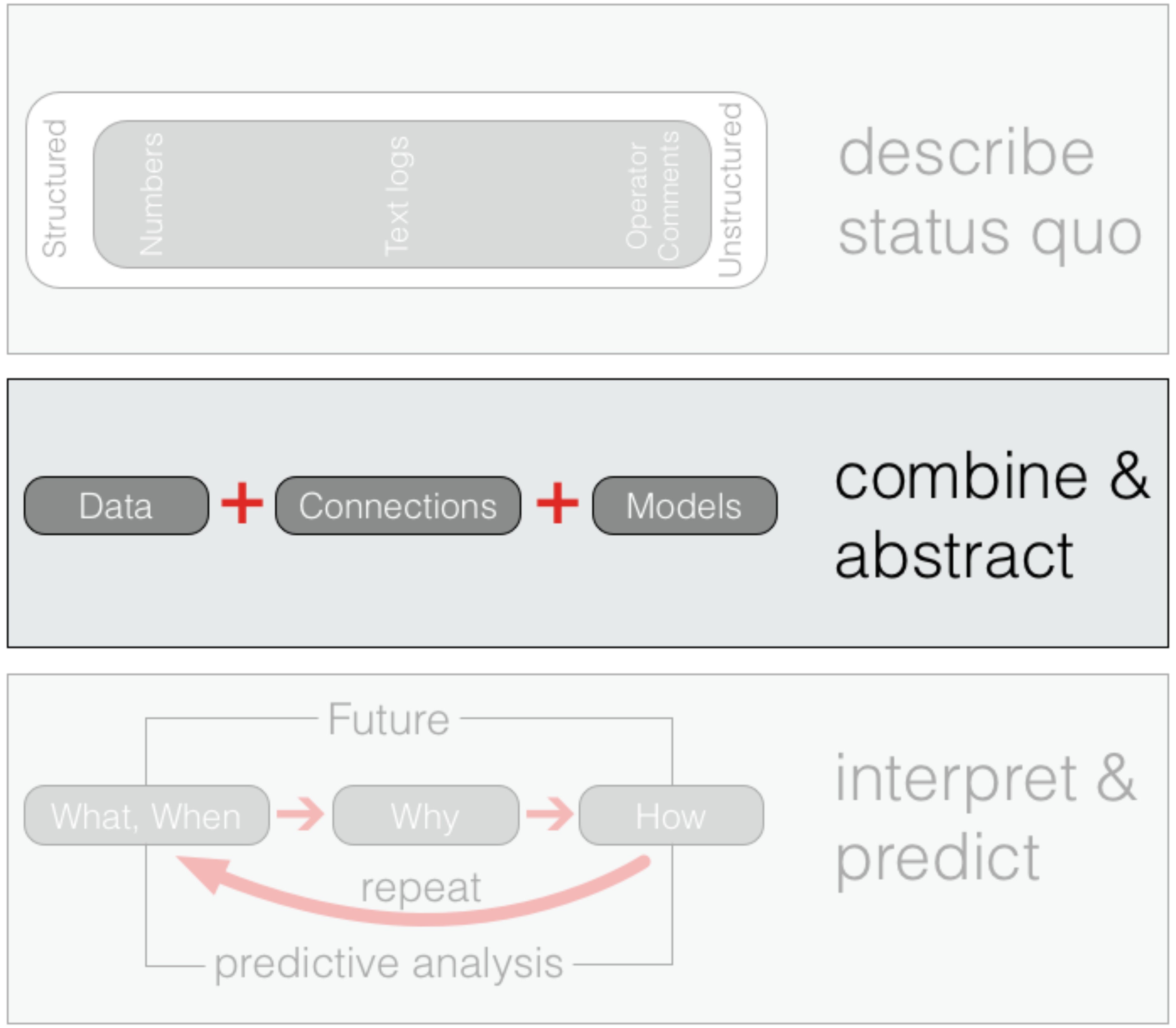

CERN IT Hadoop Service
The Hadoop Service provides Hadoop and other frameworks compatible with its ecosystem. This web page contains information and statistics about YARN which is the resource negotiator that is running inside our clusters.
Clusters overview
77
Total Nodes
4544
Total VCores
4.7 TB
Total RAM
2069
Total HDD
6.5 PB
Storage Space
2300
Jobs per day
Submitted applications
2,000
1,500
1,000
500
0
2016-11-09
2016-11-16
2016-11-23
2016-11-30
2016-12-07
General statistics
Total number of files 48174864
Total number of blocks 37559145
Total HDFS capacity 5.9PiB
Used HDFS capacity 4.0PiB
Median datanode usage 65.48 %
Total containers (last 24h) None
Max simultaneous apps (last 30 days) 35
Active users (last 24h) 43
Stats update timestamp 2016-12-9 9:30
Applications types
MAPREDUCE
55,717
4,000
3,000
2,000
1,000
0
2016-11-11
2016-11-18
2016-11-25
2016-12-02
2016-12-09
HDFS capacity
% used capacity
68 %

Metric Collection



- **Collection** via *IT monitoring project*
- select and summarise **relevant metrics**
  - Find & remove unexpected / unintended access patterns
- **To what level** can we trust our metrics & assumptions?
  - Evaluate data quality: eg accuracy, units(!)
    - data that has not been used quantitatively yet has likely problems
- Simple **quantitative cross-checks**:
  - eg for CPU
    - $\sum job_{cpu} \sim \sum sched_{cpu} \sim \sum host_{cpu}$ (any significant losses?)
  - eg for disk
    - ∑ disk I/O ~ ∑ user I/O + ∑ internal I/O (ratio expected?)

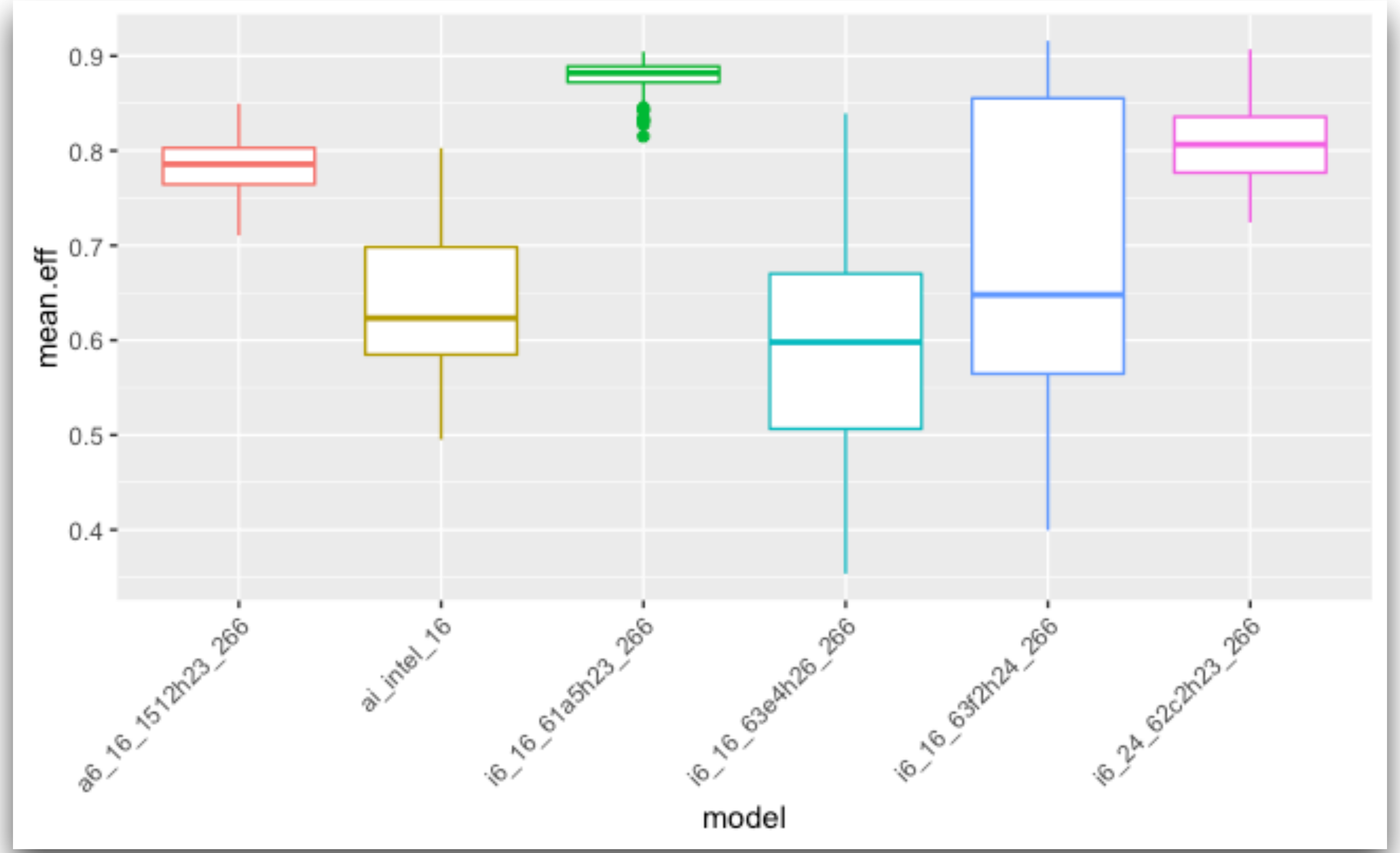
Connecting Data



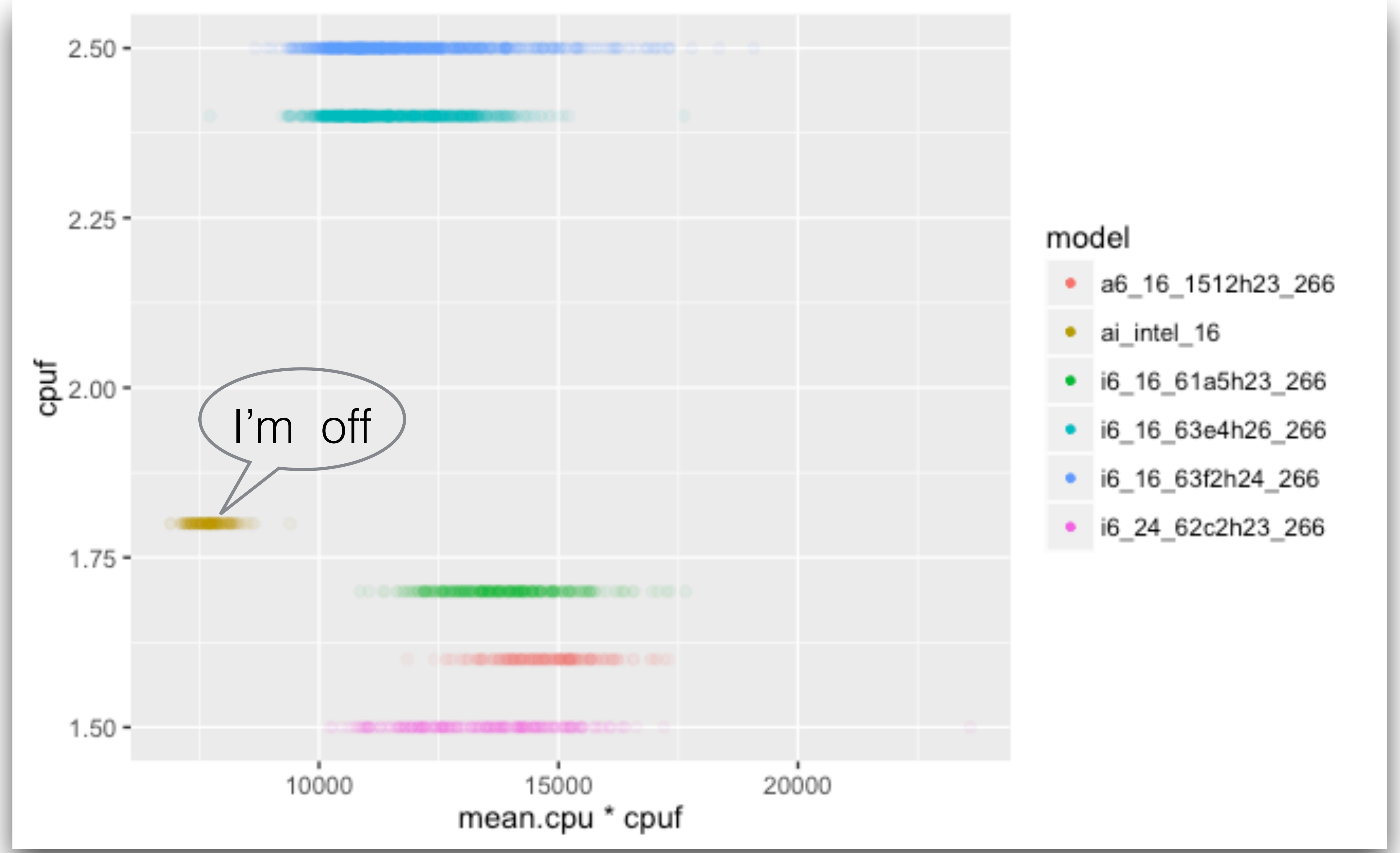
- Involved in several experiment performance studies
  - Starting point: why do users/service providers see:
    - **slow** file **access**? **inefficient CPU** usage?
    - differences: Wigner vs CERN, CERN vs T1, etc..
  - **where is** the **bottleneck? where should be?**
- Connected data from experiment, storage, batch
  - connected infrastructure data: LAN db, hardware db
  - enables correlation with location, hw type, HEPSPEC

Examples: One production task

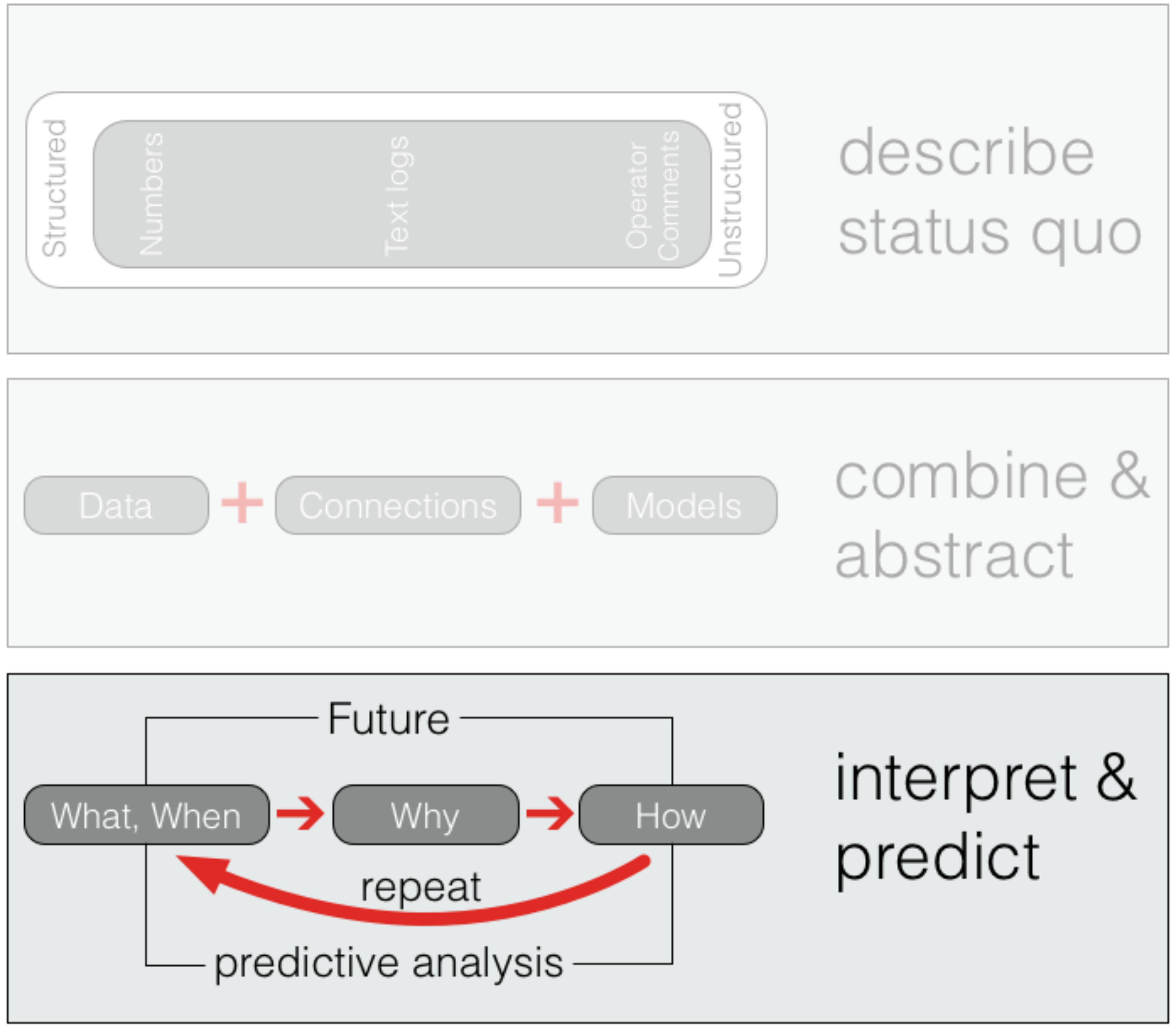
CPU "Efficiency" versus H/W types



CPU Performance Calibration check



Model Predictions



- Evaluate (simple) predictive models: Can we construct a more performant system for the same price?
- Simplest case: CPU bound
  - **CPU benchmarks** + memory optimisation => MC throughput
- Not CPU-bound case - balance between
  - CPU, WN storage, LAN storage, WAN storage, network

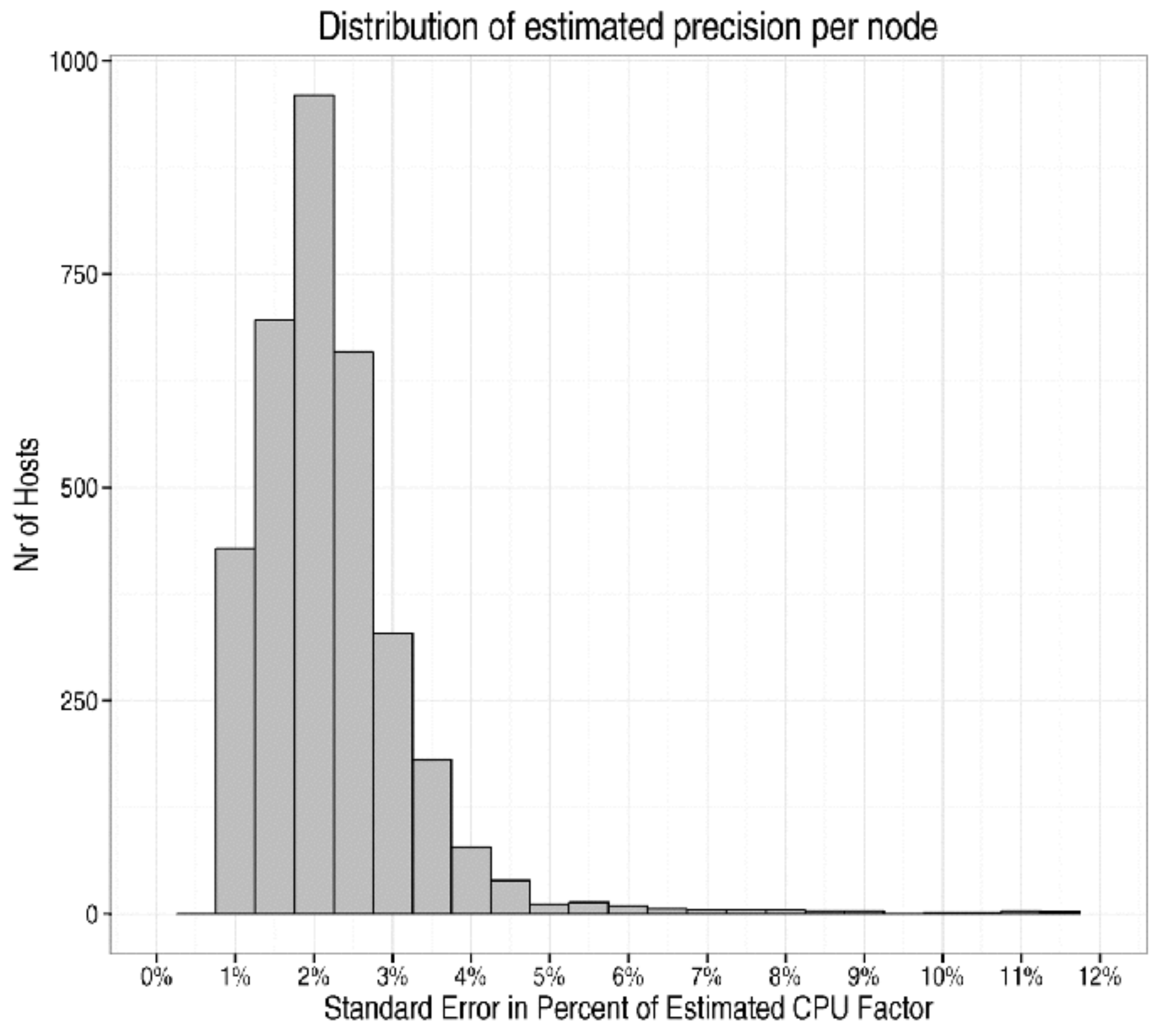
Passive Benchmark

- Basic Idea:
  - Take the workload as set of benchmarks
    - Assume jobs per task are equal, compare runtime
    - Based on existing monitoring logs

- Advantages:
  - Zero intrusion, basically no overhead
  - Always representative (the benchmark **is** the workload)

- Application:
  - Observe performance during operation
  - Compare configurations by performance on the actual workload

- Accuracy / Precision
  - Experiment on LSF dataset: ATLAS and CMS, 3 months
  - Equal or better prediction of performance than HepSPEC06
  - Precision per node is below 5% error for 98% of nodes



A Typical Analysis Pattern

- Preselection/reformat batch (goal: max. throughput)
  - “horizontal” scaling allows to skim for useful data -> input for repetitive analysis steps
  - “standard” Hadoop chain with Spark works very well
- Interactive analysis & visualisation (goal: min. latency)
  - big memory sometimes helps more than many boxes
  - analysis language support for parallelisation helps even more
- Ideally both above systems (many boxes - big memory) are integrated

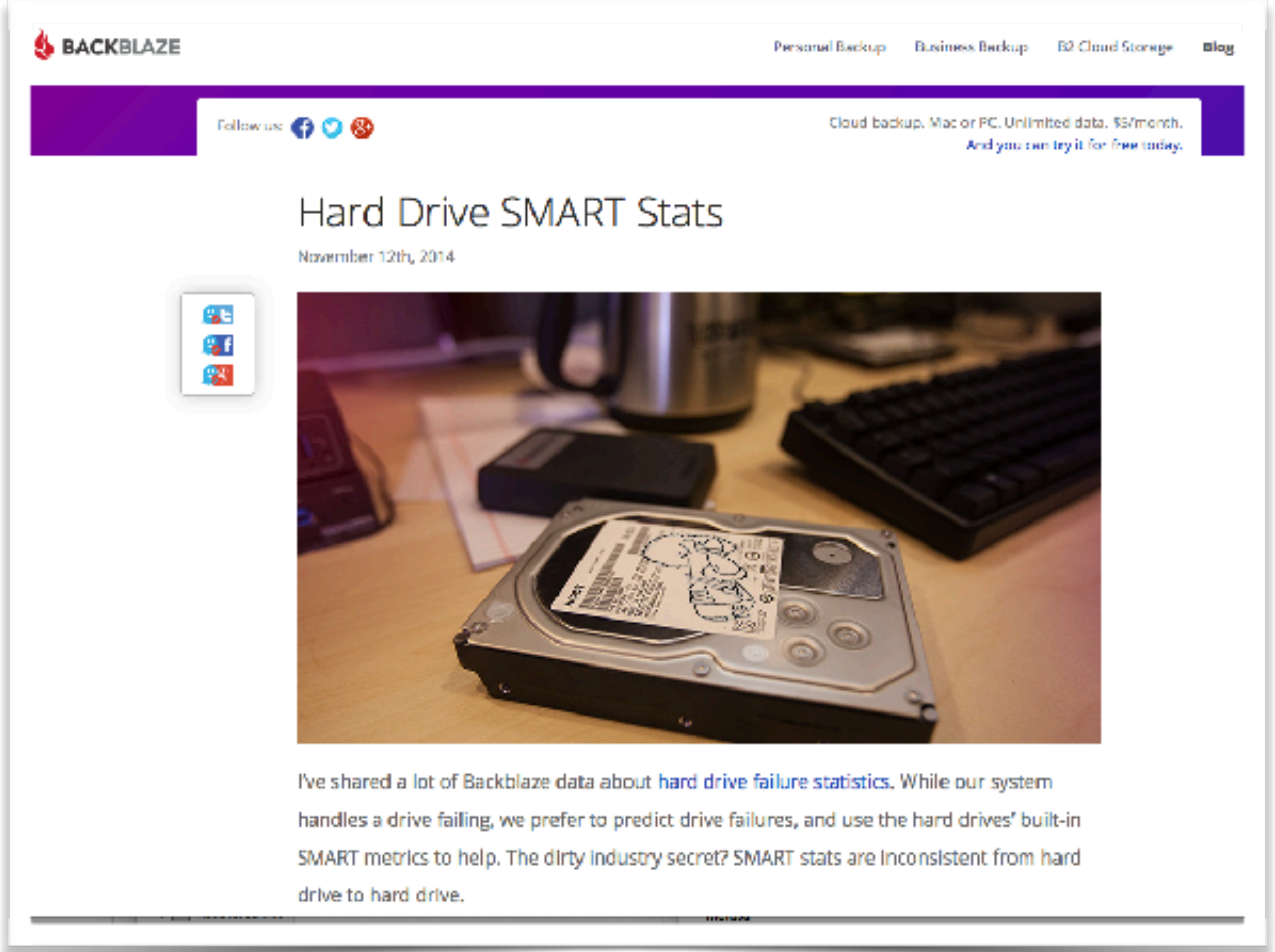
Next Steps: Analysis Disk Failures

- Analysis of disk failures in EOS, Hadoop, …
  - existing smart sensors
  - logs from kernel / EOS / hadoop

1: quantitative comparison of kinetic and consumer disk failure rates

2: predictive maintenance

3: impact on service performance



ML job classifier

- Can we automatically classify jobs to be
  - into: CPU-bound, file-I/O bound, box I/O-bound, site I/O-bound
- Metrics used: experiment (task), lemon (local disk), batch (cpu), EOS (site disk)
  - Evaluating: **simple cut model** and **random forrest**
- Classifier output: optimisation hints
  - file replication: eg these files (don't) need additional replicas
  - job placement: eg these jobs (don't) need a local SSD