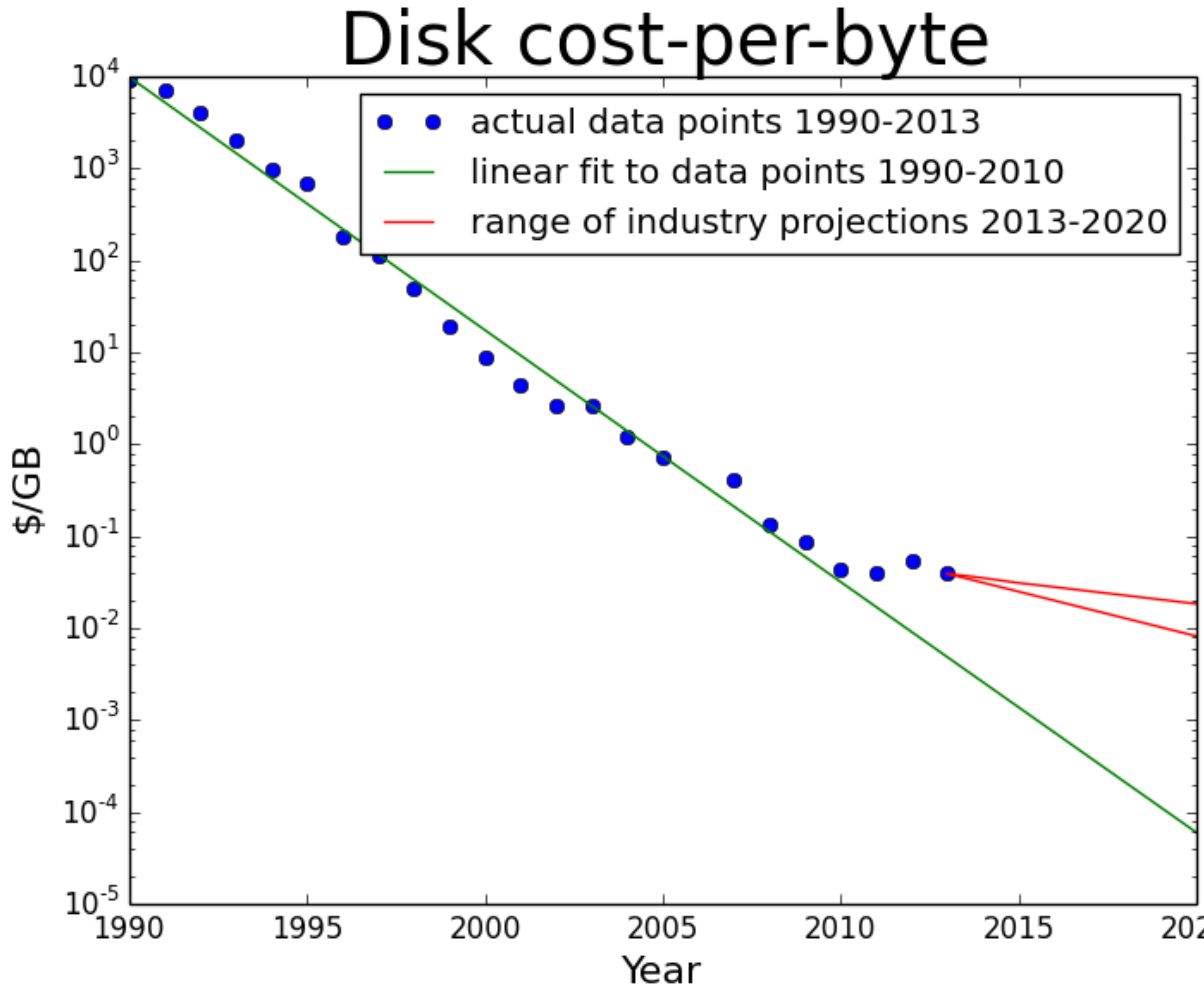
Infrastructure Analytics for Optimisation

Dirk Duellmann, CERN IT, storage group

Why?

- 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 -> model impact commercial clouds, disk->flash->NV memory
- experiments and IT are accountable to funding bodies: throughput per investment?
    - quantitatively instead of just qualitatively
    - absolute (not just relative) numbers

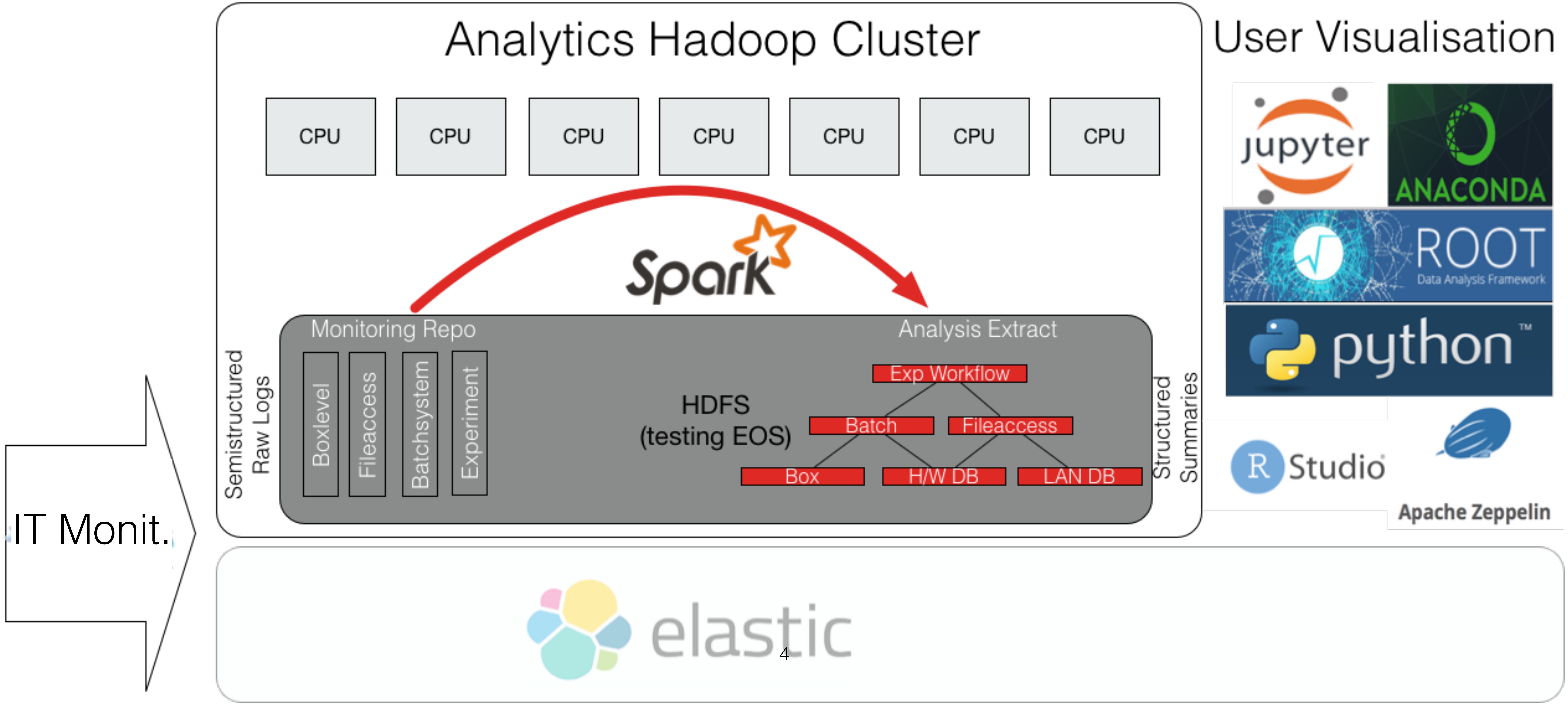


Analysis Input Data

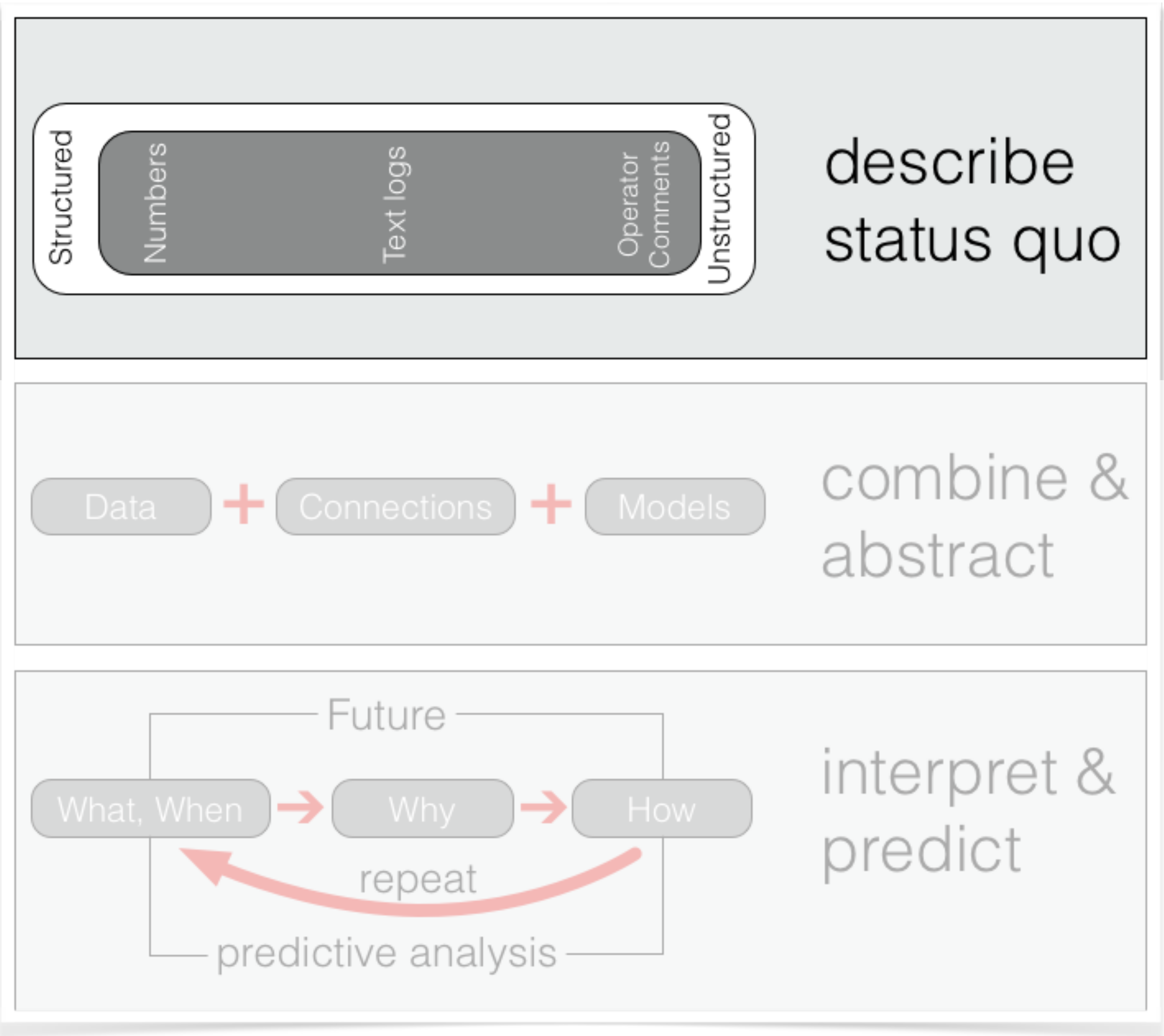
(bulk items collected by IT monitoring project)

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

(2016)

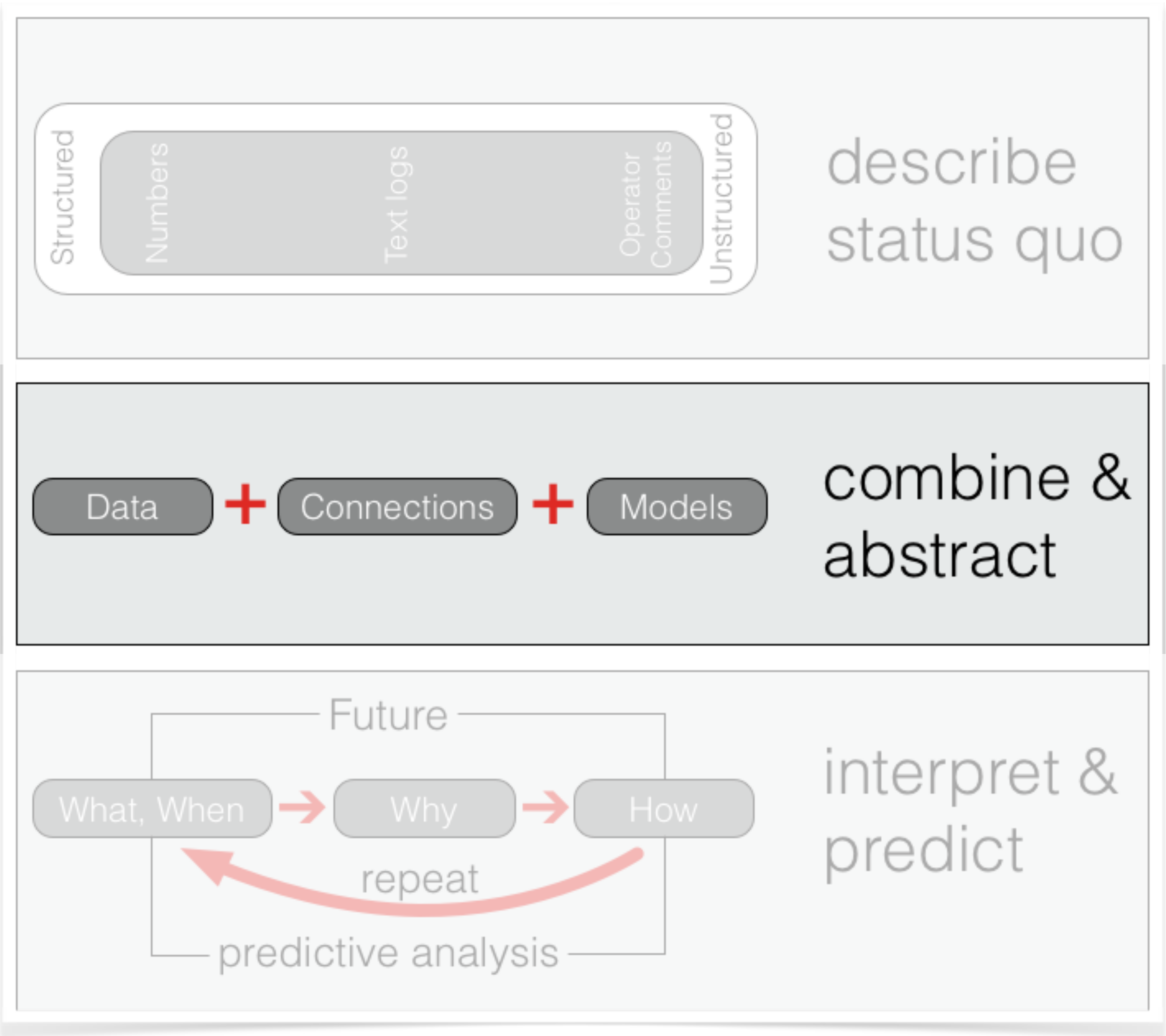

Analytics Hadoop Cluster
CPU
CPU
CPU
CPU
CPU
CPU
CPU
Spark
Semistructured
Raw Logs
Monitoring Repo
Boxlevel
Fileaccess
Batchsystem
Experiment
HDFS
(testing EOS)
Analysis Extract
Exp Workflow
Batch
Fileaccess
Box
H/W DB
LAN DB
Structured
Summaries
User Visualisation
jupyter
ANACONDA
ROOT
Data Analysis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

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

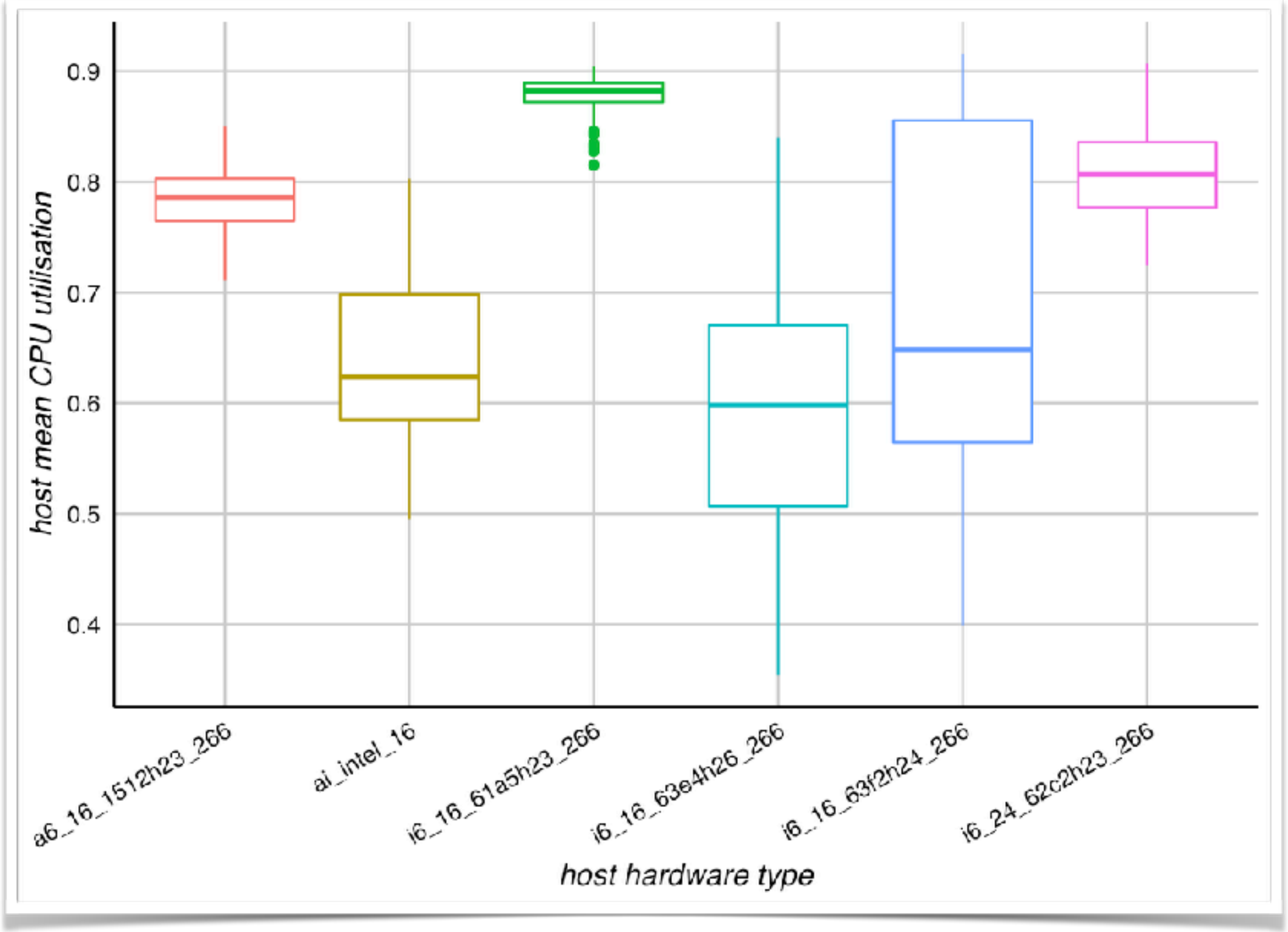


- Involved in several experiment performance studies
  - Starting point: why do users/service providers see:
    - **slow** file **access**? **inefficient CPU** usage?
    - differences: Wigner vs CERN, CERN vs T1, etc..
  - **where is** the **bottleneck? where should be?**
- Connected data from experiment,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

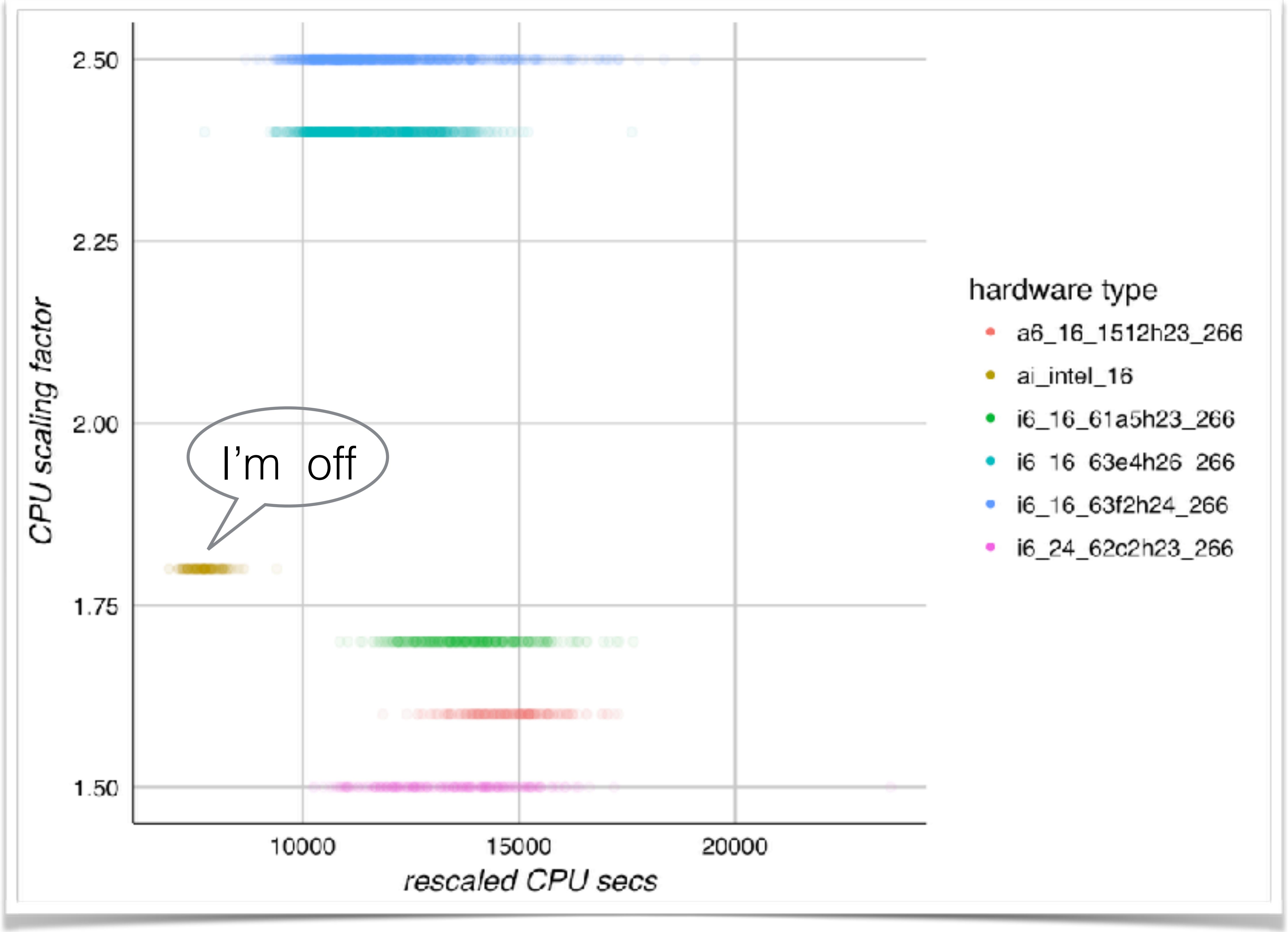
data from Alice production in 2016

Examples: One production task

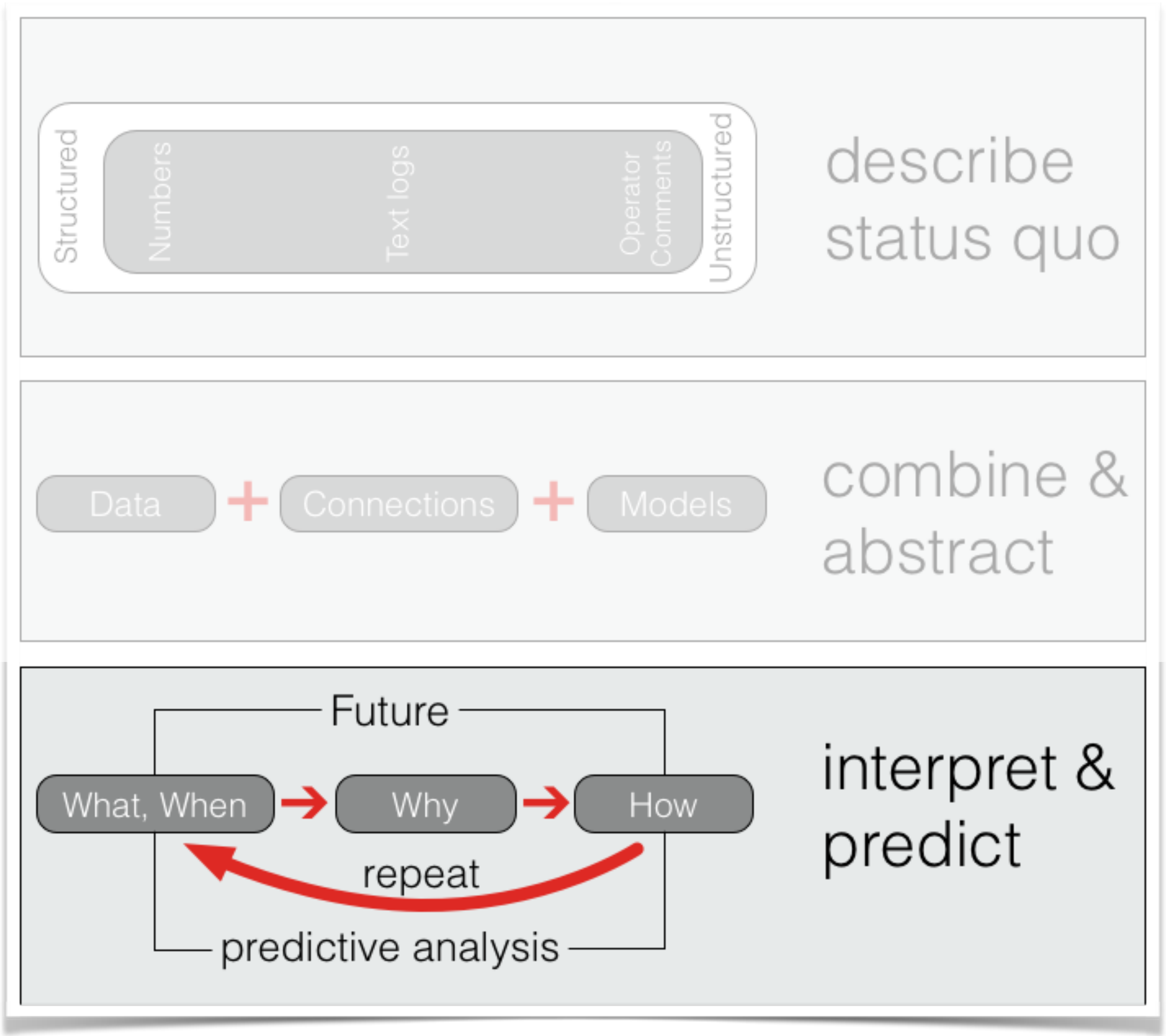
CPU “Efficiency” versus H/W types



CPU Performance Calibration check



Model Predictions

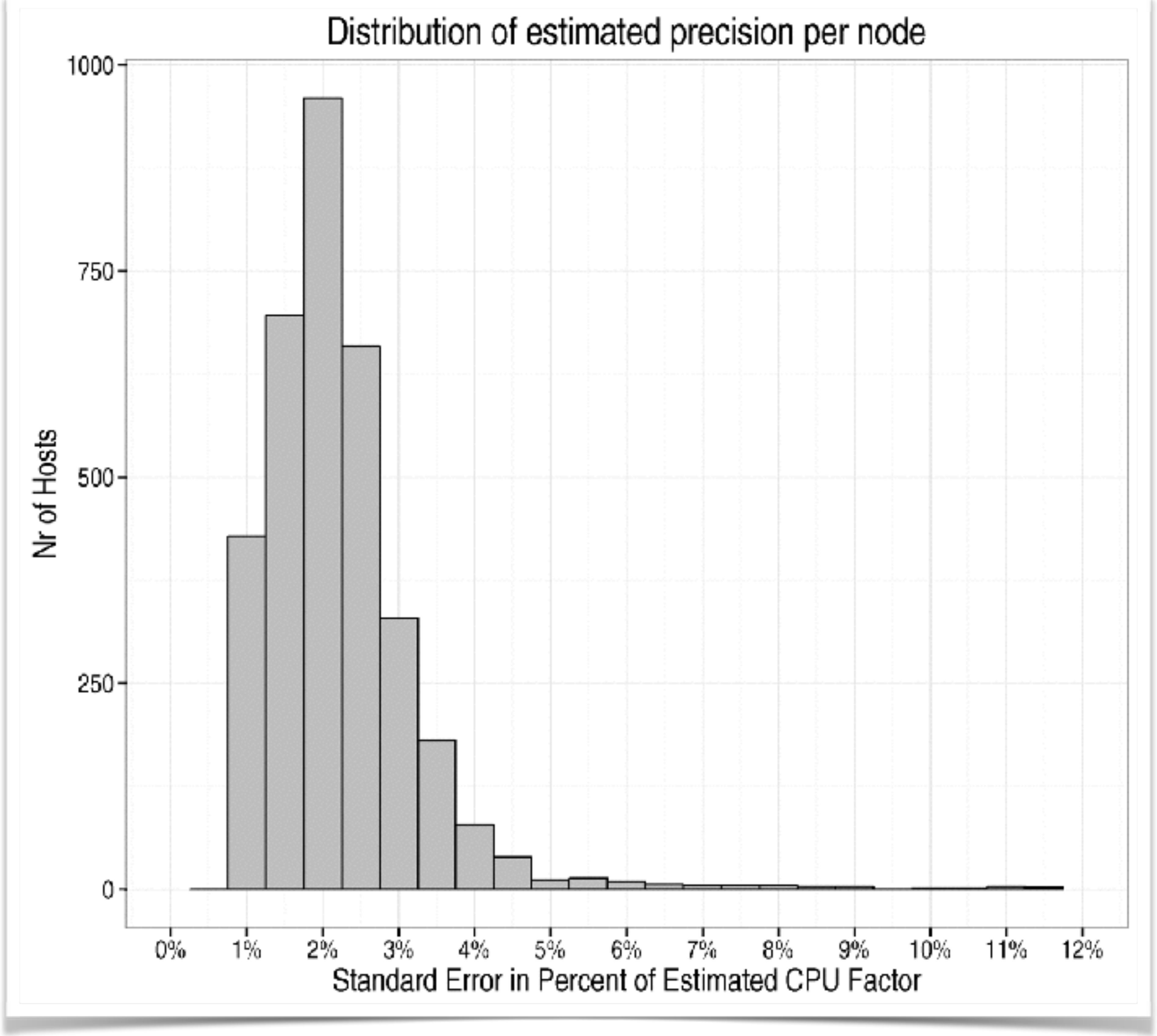


- Answer via predictive models:
  can we construct a more performant system for the same price?
- Simplest case: CPU-bound jobs
  - **CPU & RAM speed** => MC throughput
- More balanced case:
  - need to consider:
    CPU, local I/O, LAN I/O, WAN I/O, network speeds

PhD th. C. Nieke, TU Braunschweig
accepted @ IEEE Cloud 2017

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



Established collaboration,
SSRC, UC Santa Cruz

Next steps:
Analysis of Disk Failures

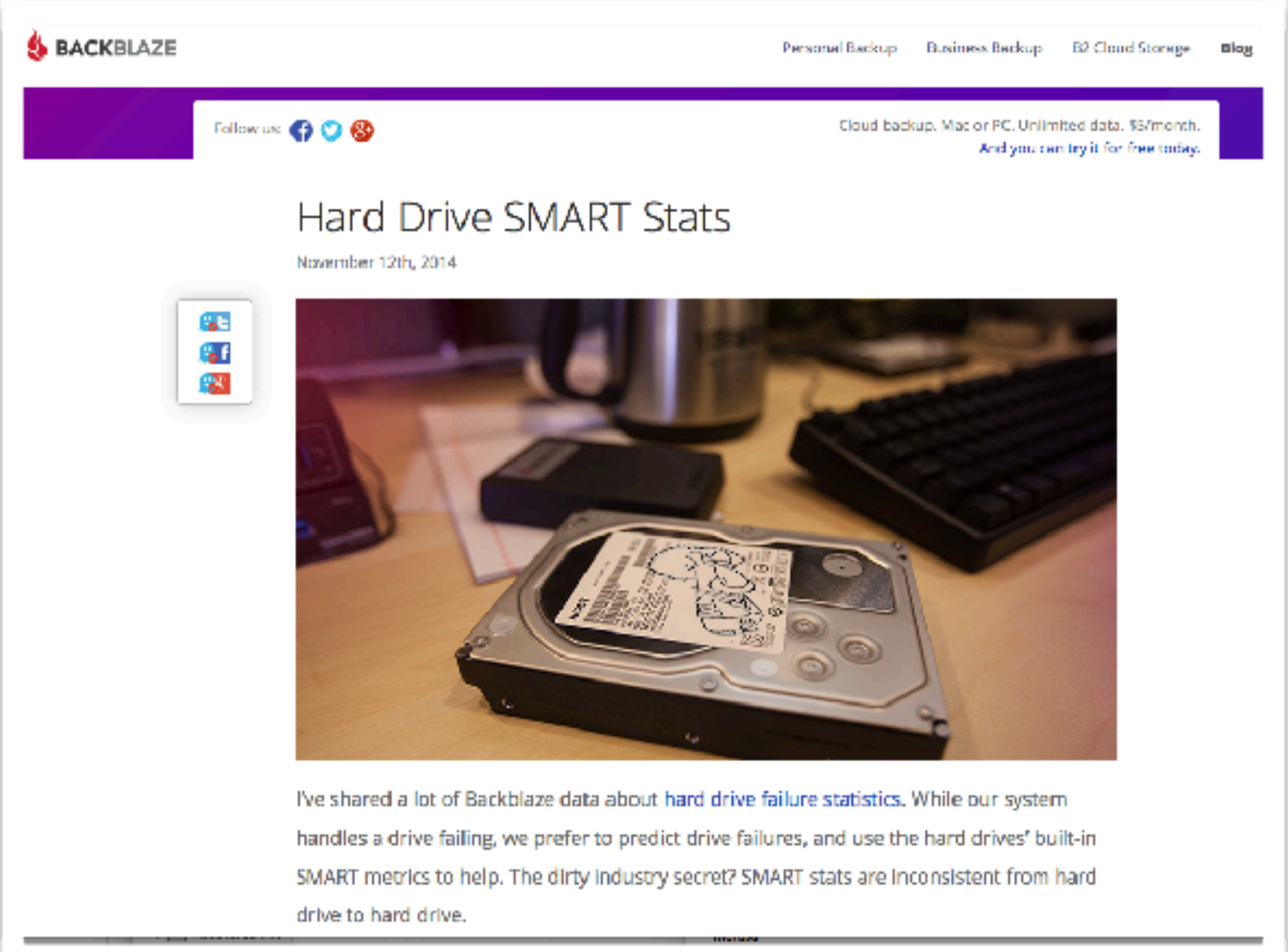
- Failures on some 70 k disks (similar O(backblaze))

  1: failure impact on service performance

  2: comparison of enterprise and consumer disks

  3: predictive maintenance

- Using data from:
  - ~~existing~~ smart sensors (no systematic collection)
  - disk replacement ~~logs~~ (logs are not complete enough)
  - disk hardware ~~status~~ repository (does describe purchase but not status quo)
  - logs from EOS & Hadoop clusters



ML job classifier

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

Typical Analysis Pattern: Scatter-Gather

- 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

Challenges?

Current resource limit

- **People** with analysis experience and understanding of end-to-end computing goals
  - black box ML may help with some problems
- Analysis software: there is **plenty -** almost too much - **choice**
  - for statistical analysis - a **quality plot** and fitting package **is still a challenge**
  - specifically for semi/un-structured infrastructure data
    - **good language string support**: eg factors, regexp, json, jquery
    - **column store** and **performant join implementations**
  - **functional languages** greatly **simplify parallelisation**, but can greatly **reduce** set of **contributors**.
- **Active workbooks** with import from spark and export to pdf/html are nice
  - but a smooth **boundary between interactive and batch** mode is still an issue