Quasi-online accounting
and
monitoring system for distributed clouds

R. Seuster and R. Sobie

F.Berghaus, K.Casteels, C.Driemel
M. Ebert, C.Leavett-Brown, M.Paterson
University of Victoria

CHEP 2018, Sofia

Introduction

- We run HEP workload for ATLAS + Belle II on distributed clouds:
  - 15 clouds, general/HEP research and commercial (Azure, Amazon, Google) in North America & Europe
  -
- We are motivated by two key elements:

  (1) As we use clouds of other HEP sites, they require accurate and timely accounting information

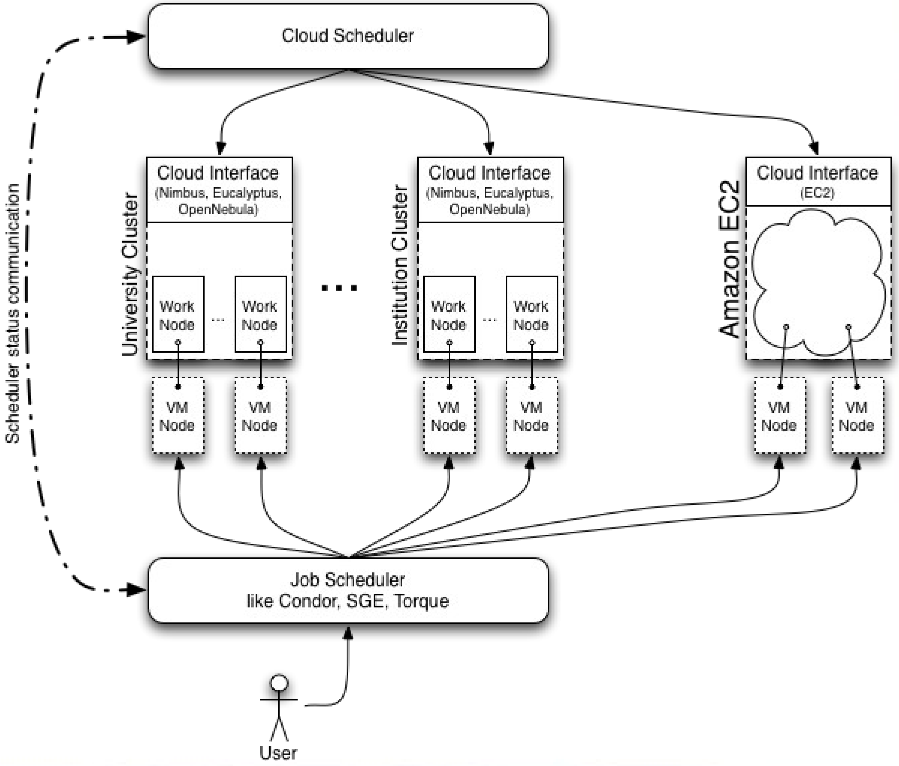
  (2) We use our framework and hardware to provide ATLAS and Belle II job monitoring

Intermezzo: CloudScheduler

How do we run distributed clouds ?

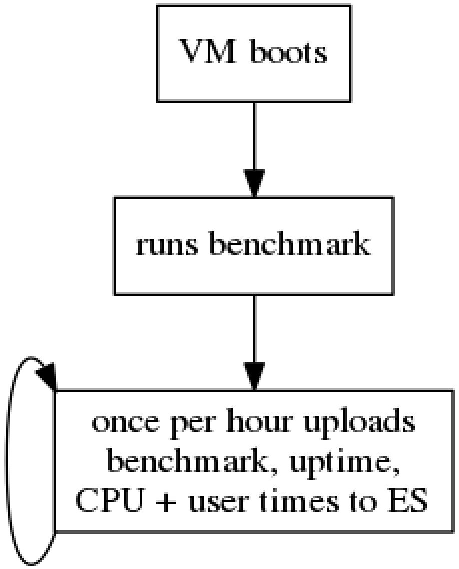
- We run 3 instances of cloudscheduler (CS):
  - 2 for ATLAS (Canada and at CERN) and 1 for Belle II (Canada)
- CS checks queue for idle jobs and boots VMs
  - contextualization of VM registers VMs in condor on CS server
  - jobs will then start on new VMs
  - CS will retire VMs if no workload left

http://cloudscheduler.org

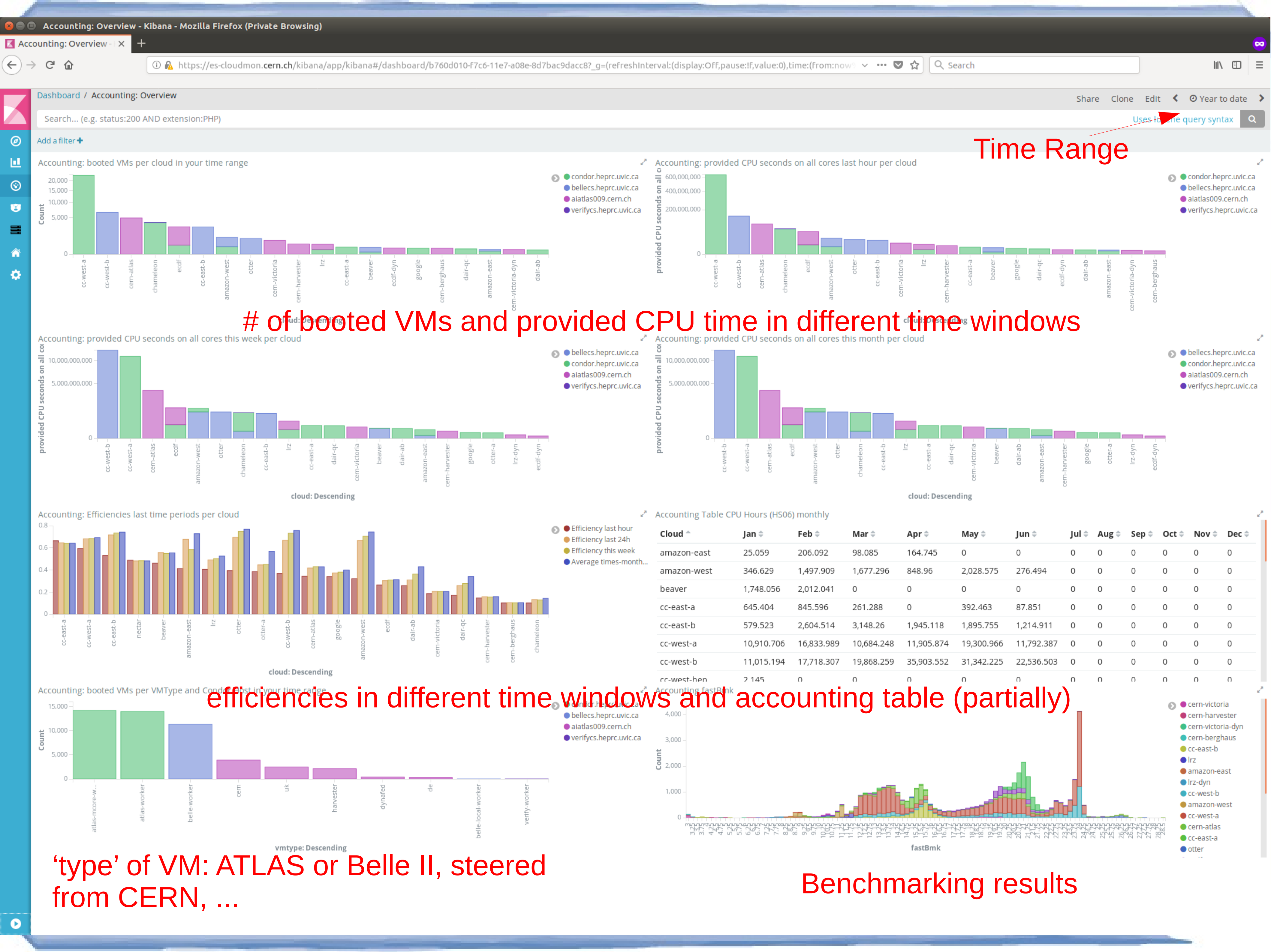


Accounting Framework

- ElasticSearch(ES)/Kibana instance at CERN + pycurl(*) on VMs to upload data
    - one document in ES per VM per month
- “Fast-HS” benchmark run at VM boot time
- once an hour all VMs upload benchmark and “uptime, CPU and user times” to ES
    - plots and tables automatically updated
- several displays: last hour, last day, last week, last month
- upload updates existing documents in ES (new document in ES for each VM at beginning of month(**))


VM boots
runs benchmark
once per hour uploads benchmark, uptime, CPU + user times to ES

(*) no additional install on VMs required: pycurl uploads in-memory json documents to ES
(**) month is part of the name of document in ES



Accounting: booted VMs per cloud in your time range

Accounting: provided CPU seconds on all cores last hour per cloud

Accounting: provided CPU seconds on all cores this week per cloud

Accounting: provided CPU seconds on all cores this month per cloud

Accounting: Efficiencies last time periods per cloud

Accounting Table CPU Hours (HS06) monthly

Cloud	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
amazon-east	25.059	206.092	98.085	164.745	0	0	0	0	0	0	0	0
amazon-west	346.629	1,497.909	1,677.296	848.96	2,028.575	276.494	0	0	0	0	0	0
beaver	1,748.056	2,012.041	0	0	0	0	0	0	0	0	0	0
cc-east-a	645.404	845.596	261.288	0	392.463	87.851	0	0	0	0	0	0
cc-east-b	579.523	2,604.514	3,148.26	1,945.118	1,895.755	1,214.911	0	0	0	0	0	0
cc-west-a	10,910.706	16,833.989	10,684.248	11,905.874	19,300.966	11,792.387	0	0	0	0	0	0
cc-west-b	11,015.194	17,718.307	19,868.259	35,903.552	31,342.225	22,536.503	0	0	0	0	0	0
cc-west-hep	2,145	0	0	0	0	0	0	0	0	0	0	0

Accounting: booted VMs per VMType and Condor host in your time range

Accounting fastBmk

Time Range

# of booted VMs and provided CPU time in different time windows

efficiencies in different time windows and accounting table (partially)

‘type’ of VM: ATLAS or Belle II, steered from CERN, ...

Benchmarking results

Enhancing Stability of Accounting Information

- We rely on stability of ES instance at CERN
- VMs repeatedly update existing documents in ES
  - failed uploads from VMs to ES will be corrected by next successful upload of accounting information (1h later)
  - most interest in monthly breakdown → at beginning of month, all running VMs create new documents in ES
    - monthly accumulation in plots and tables almost trivial
    - document name based on name and boot time of VM and have current month appended

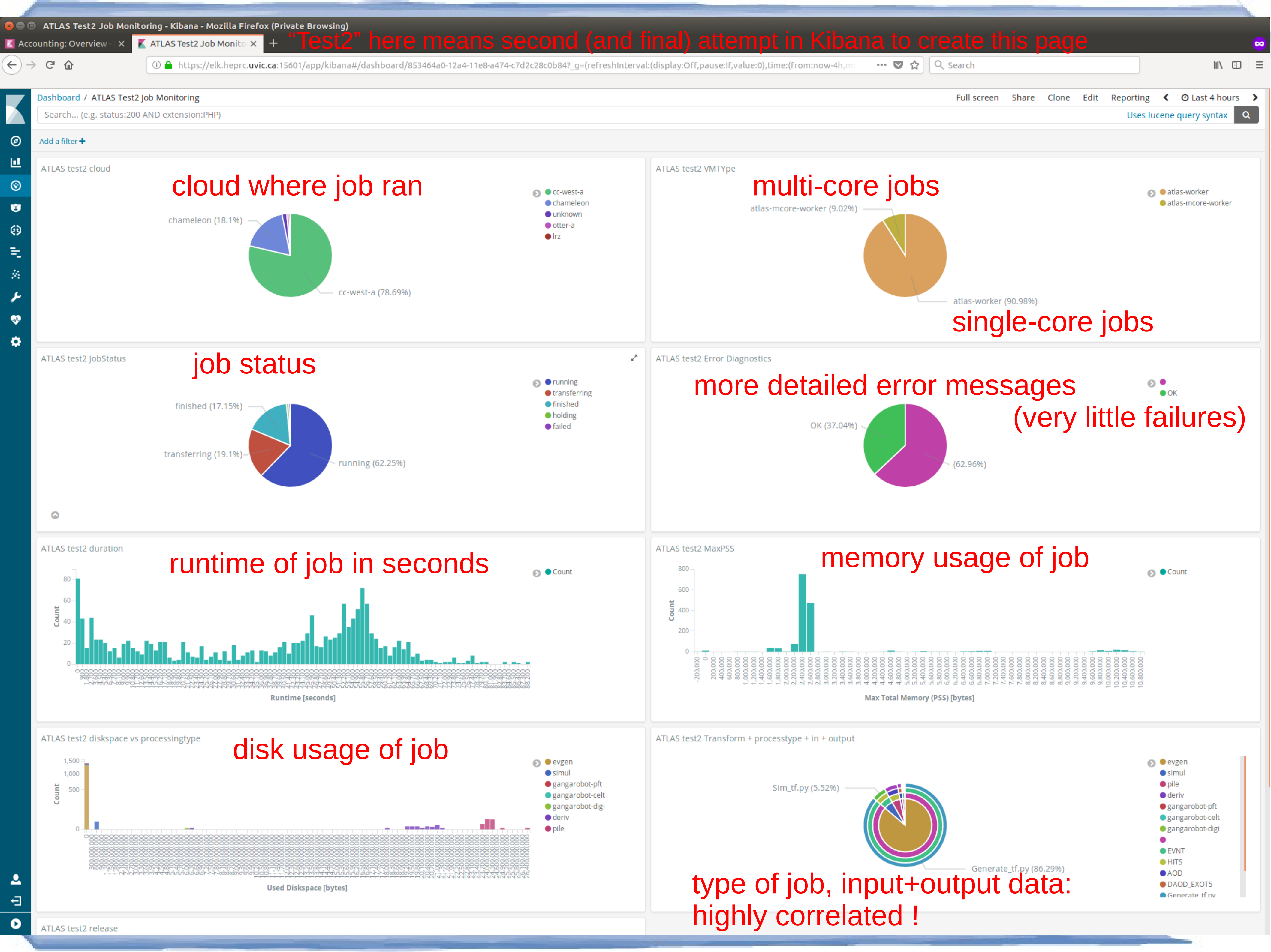
      → also allows for simple retrieval from scripts
- To ensure accuracy of accounting information, we performed extensive cross checks

Re-using Frameworks: Job Monitoring

- ElasticSearch/Kibana at UVic used to additionally monitor Paylod Job Successes/Failures Monitoring
  - needed to identify quickly faulty clouds, e.g. in case of connection problems for up-/download of data
- Scripts for accumulation of information runs on dedicated VMs, collects information from queueing system (HTCondor), experiments job database (Panda/DIRAC) and on VMs (in case of Belle II)
- different approaches needed for ATLAS / Belle II

Job Monitoring for ATLAS

- Panda DB main source of information, inquired once an hour for all jobs that finished in our two queues
  - this results in fine grained request to Panda via curl
    → very small load on DB
- HTCondor job ID part of returned information
  - Combined with info from condor and cloudscheduler
    → we know on the cloud where the job ran
    → cloud dependent job monitoring
  - wealth of other information available
    → detailed monitoring possible



"Test2" here means second (and final) attempt in Kibana to create this page

cloud where job ran

multi-core jobs

single-core jobs

job status

more detailed error messages

(very little failures)

runtime of job in seconds

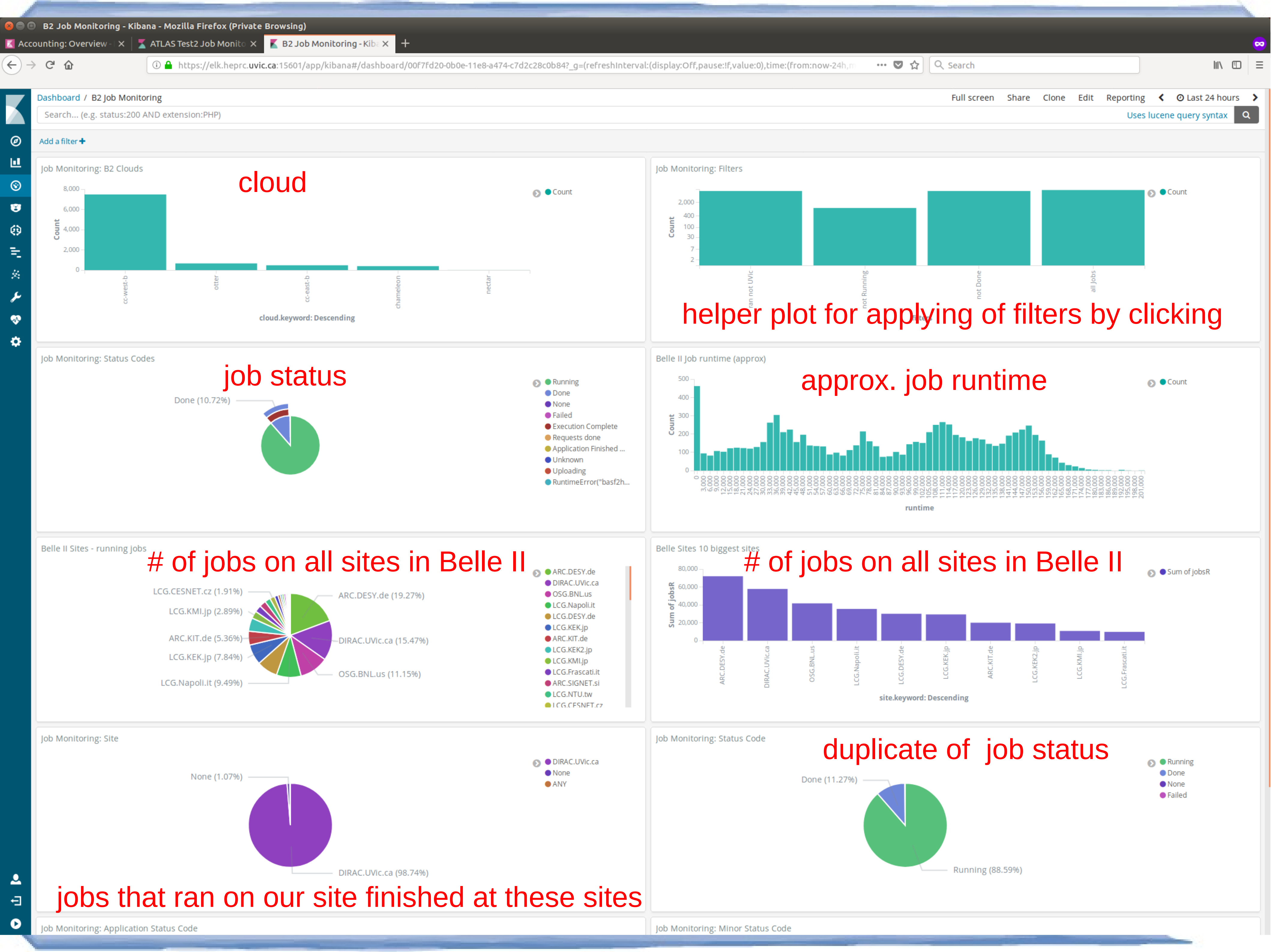
memory usage of job

disk usage of job

type of job, input+output data:
highly correlated !

Job Monitoring for Belle II

- Belle II’s DIRAC-DB does not allow for easy data mining like ATLAS’ Panda DB
  → small script on all worker nodes collects every 15 mins all DIRAC job IDs and reports them back into ES with state “running”
- “Collector scripts” asks ES for all jobs “running” and last update older than 1h (either because job stopped or missed 4 times in a row to update in ES, unlikely)
- for all those job IDs ask DIRAC DB via CLI interface for update, and store updated information in ES
  - failed jobs at our site can be resubmitted to other site and would continue to be monitored



cloud

helper plot for applying of filters by clicking

job status

approx. job runtime

# of jobs on all sites in Belle II

# of jobs on all sites in Belle II

duplicate of job status

jobs that ran on our site finished at these sites

How to transfer Secrets onto VMs

- VMs used can be on a public cloud with public IP addresses.

  Need to transfer GSI keys, ES username/passwords securely onto VMs

- Also, How can we ensure that
  - our pool of VM only contain ‘our’ VMs
  - our VMs run only our workload → HTCondor and GSI
- Secrets could be certificates (GSI) used for condor communication between services, ssh keys, passwords for other services (e.g. ElasticSearch)
- once GSI/SSL authentication for condor established can use that – but how to establish that securely ?
  - Openstack API not encrypted

How to transfer Secrets onto VM II

0. Preparation on CS:

-CS boots new VM, generates random blob and encrypts payload with random blob

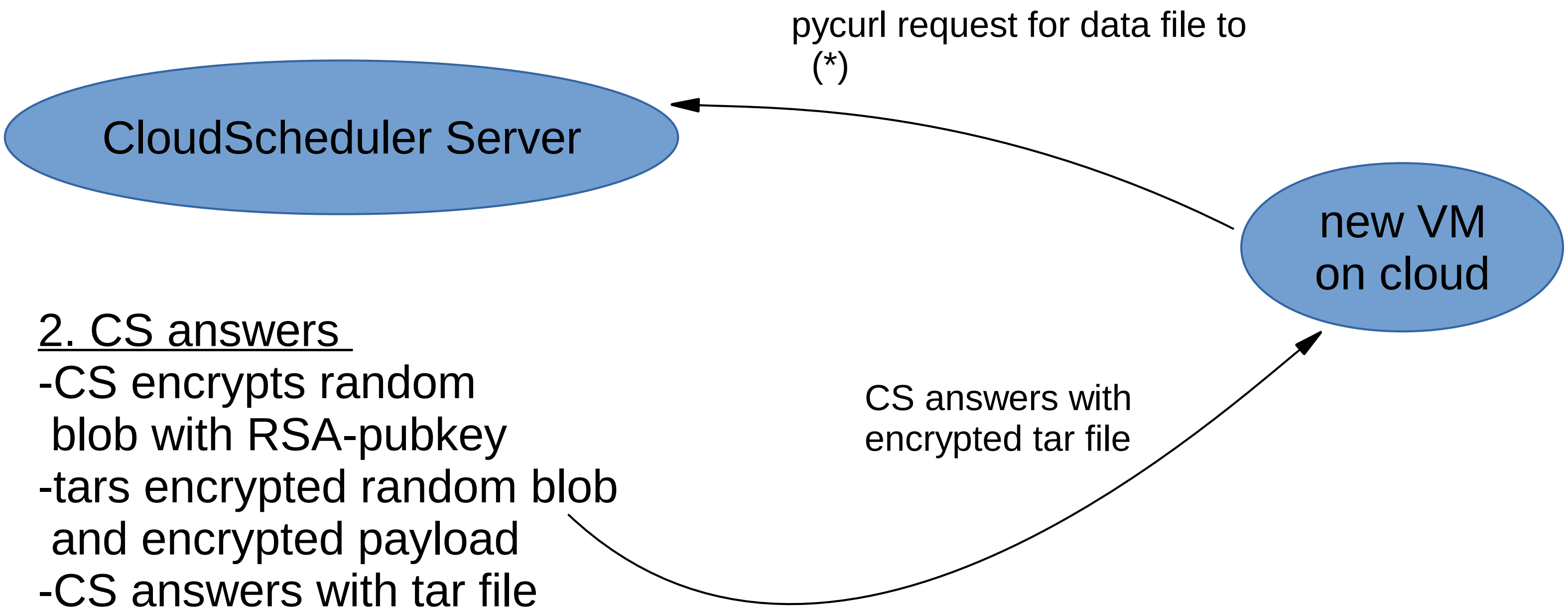
-shared secret, e.g. requestID for booting new VM will be used for further encryption

→ requires small code changes to CS

1. Preparation on new VM:

-VM boots and generates RSA pub and priv keys for secure communication with CS

-VM requests secret from CS with pubkey part of http-request



pycurl request for data file to (*)

CloudScheduler Server

new VM on cloud

CS answers with encrypted tar file

2. CS answers

-CS encrypts random blob with RSA-pubkey

-tars encrypted random blob and encrypted payload

-CS answers with tar file

-CS won’t repond to same request again

-CS also won’t answer after certain time to requests for this VM

FIXME: Future Improvements: handle properly concurrent requests, implement apache module

Note: ssh pub key also contains hostname of VM, which could be used as additional cross check.

(*) pubkey here is in fact gzipped + base64 encoded ssh public key

Summary and Conclusion

- Accounting information stored in ElasticSearch and vizualized in Kibana as plots and tables
  - system very reliable with accurate numbers
- Job Success/Failure monitoring also in ElasticSearch and Kibana
  → almost online monitoring of job successes/failures
- Transfer of Secrets into VMs with industry standard tool: openssl, ssh keys results in ssh-like encryption

Backup

Encryption with ssh keys

- similar to ssh connections, ssh keys can be used to encrypt data, see e.g. https://bjornjohansen.no/encrypt-file-using-ssh-key
  - generate random bits R
  - encrypt payload P with random bits R to get P’
  - encrypt random bits R with public key to get R’
  - tar R’ and P’ and store on web server where VMs will download ‘their’ tar file
    - CS runs slightly modified python simplehttpserver [(*)] for comunication between CS and VMs
  - untar and decrypt with private key on VM

[(*)] https://docs.python.org/2/library/simplehttpserver.html