Event Service

How to exploit opportunistic (and other) resources

1ª Reunión Presencial del Proyecto VMGRID
(ATLAS TIER2 ESPAÑA)
@FAC. CIENCIAS (UAM)

Esteban Fullana (IFIC UV-CSIC)

...antes que nada disculparme por no poder estar presente físicamente en la UAM

Why Event Service

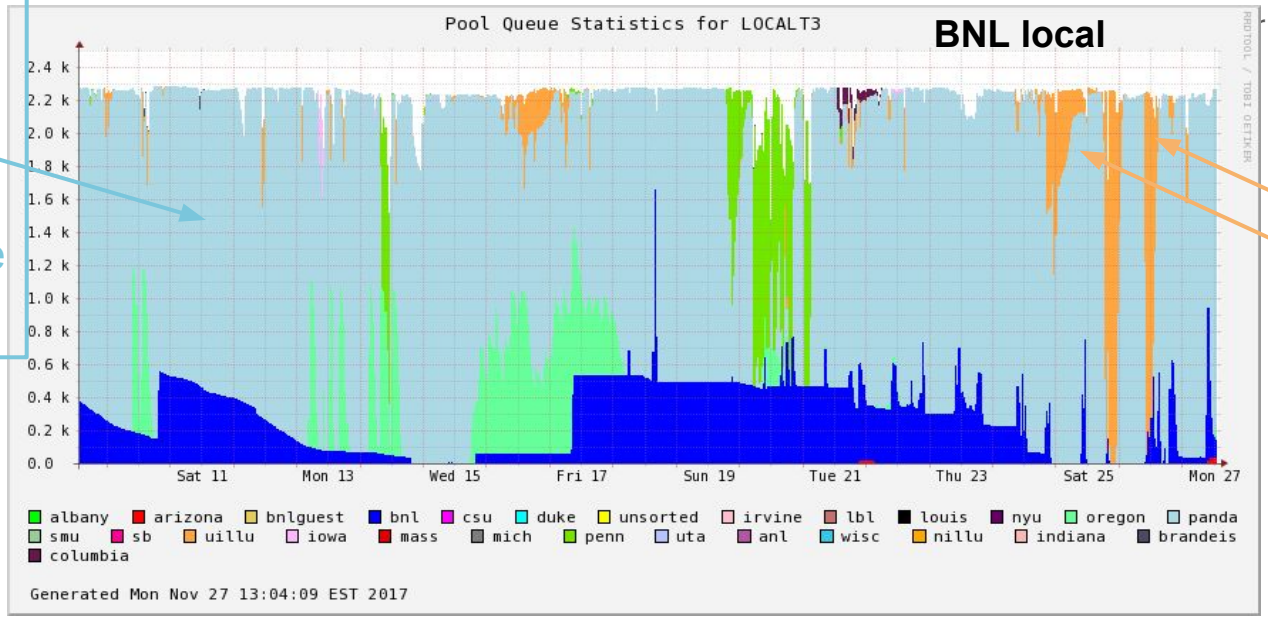
- **Efficiently and flexibly exploit any CPUs available**
  - Efficient use of **opportunistic** and volatile resources
    - Dynamically making use of available CPUs as they appear
    - Robust against being killed at any time when preempted by local jobs or higher priorities

Why Event Service

- **Efficiently and flexibly exploit any CPUs available**
  - Efficient use of **opportunistic** and volatile resources
    - Dynamically making use of available CPUs as they appear

higher priorities

When CPU is not used by local users, EventService can use it



ES jobs killed, restarted, re-killed, etc...

See other example by Rod

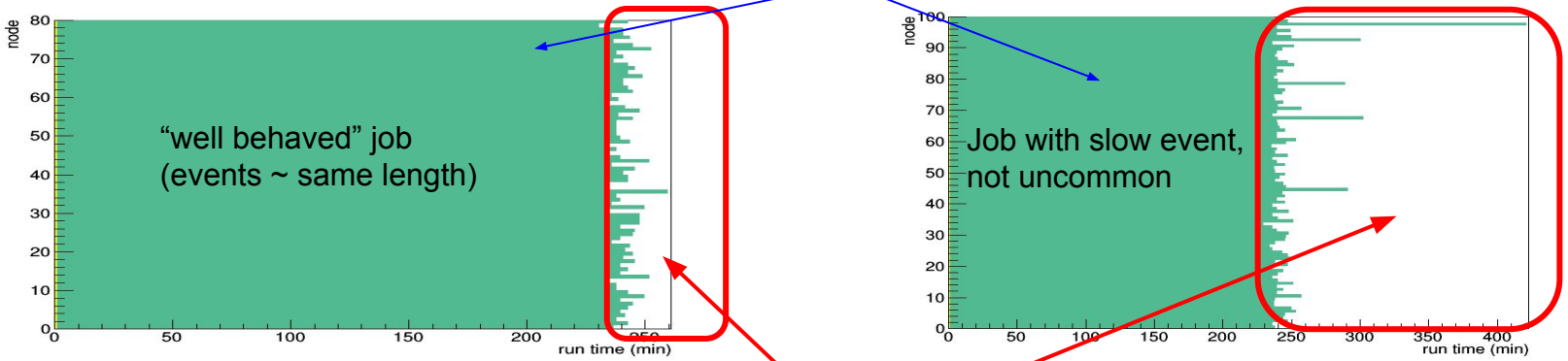
Why Event Service

- **Efficiently and flexibly exploit any CPUs available**
  - Efficient use of **opportunistic** and volatile resources
    - Dynamically making use of available CPUs as they appear
    - Robust against being killed at any time when preempted by local jobs or higher priorities
  - Better utilization of **all** resources
    - Full AthenaMP utilization of cycles, using multiple consumers to speed up a job, a task...
  - Filling scheduling gaps with backfill on **HPCs** and grid
    - Easy to fill scheduling holes where normal jobs cannot run
  - Efficient use of volunteer **ATLAS@Home** cycles
    - Exploiting background cycles with minimal waste when suspended/killed
- Profit from every minute of CPU with minimal waste
  - When killed, previous CPU work is preserved thanks to fine-grained outputs

The Event Service leverages volatile and opportunistic resources efficiently, and exploits conventional resources to the brim

Event Service on HPCs: filling the holes

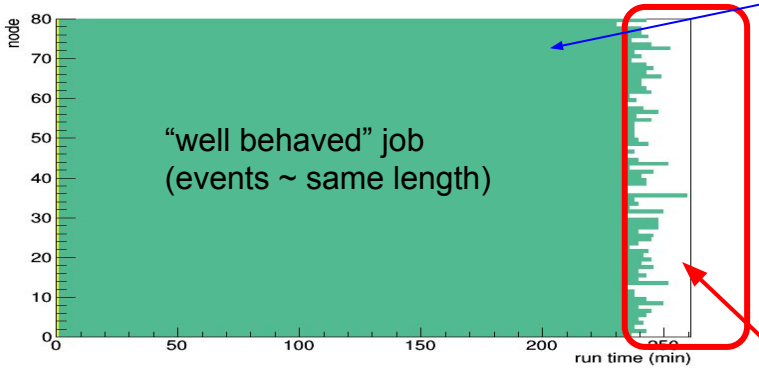
NERSC utilization per node, no ES

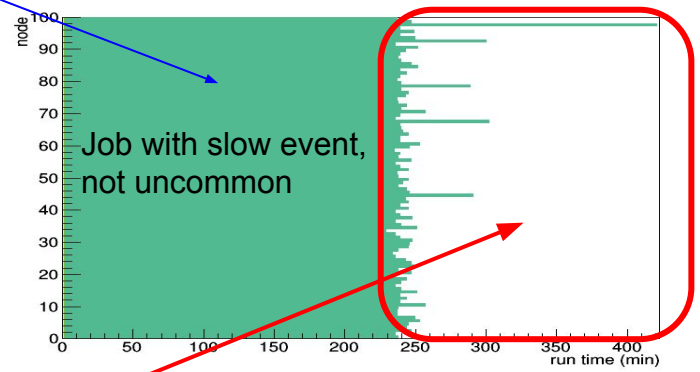


Taylor Childers

Event Service on HPCs: filling the holes

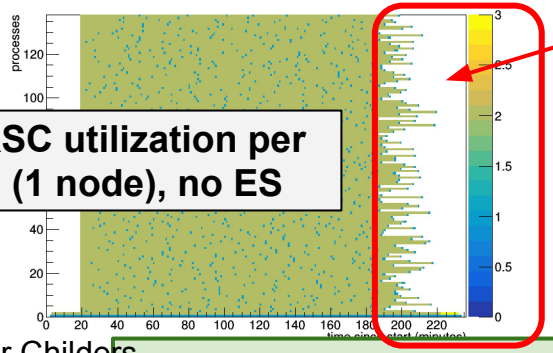
NERSC utilization per node, no ES





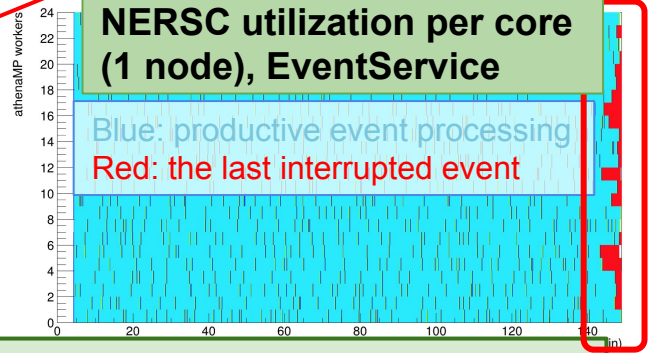
White: empty cycles

NERSC utilization per core (1 node), no ES



NERSC utilization per core (1 node), EventService

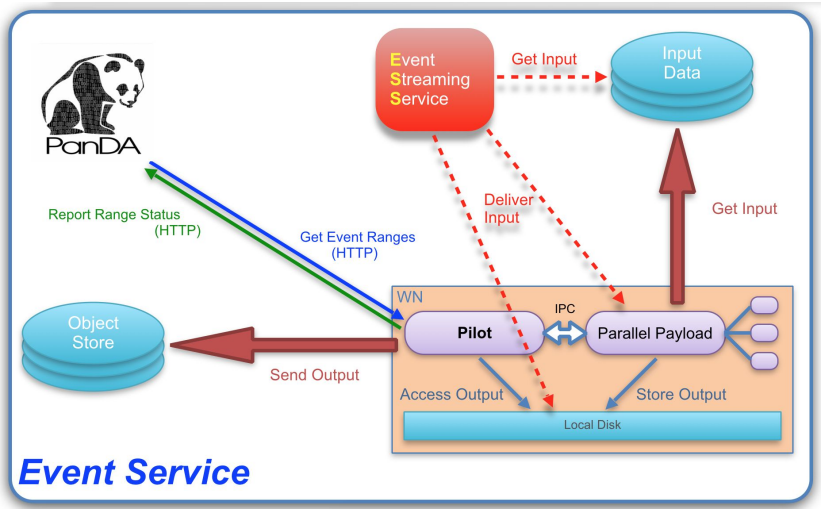
Blue: productive event processing
Red: the last interrupted event



Taylor Childers

Parallel processing leaves wasted cycles due to uneven event processing times (HPCs and all athenaMP). ES fills the holes with events

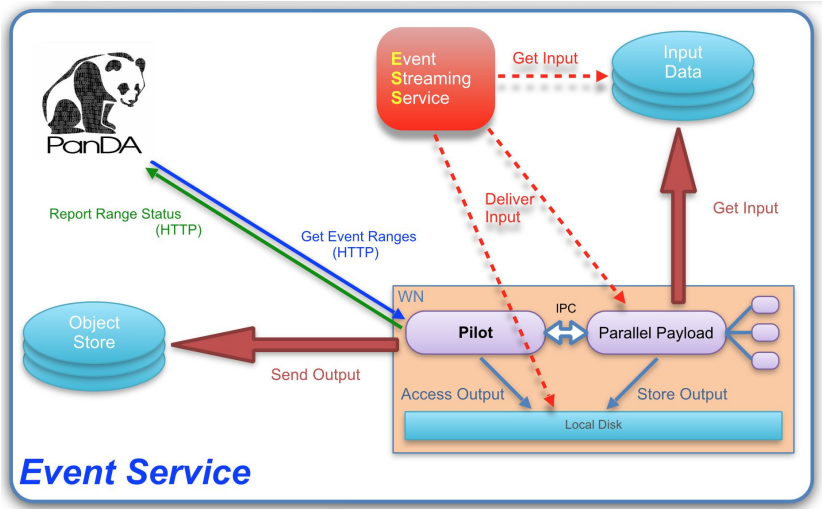
What is Event Service (ES)



- Pilot delivers **fine-grained workload to running payload application** in real time
  - **Workload:** Event Ranges
  - **Event Range:** identifier string with Input File ID, event positions within the file, etc.
- Payload application**:** AthenaMP with a special configuration for the ES
- Payload directly reads input files for the event data
  - Either **local or remote** inputs
- The payload makes a **sequence of fine-grained output files**
  - One output per input event range

- Event Service outputs are first stored into **Object Stores** (or to normal storages)
  - Final outputs are produced later by **dedicated merge jobs**
- Missing component: Event Streaming Service. Intelligent asynchronous delivery of the input data to worker nodes (in early design/prototype phase)

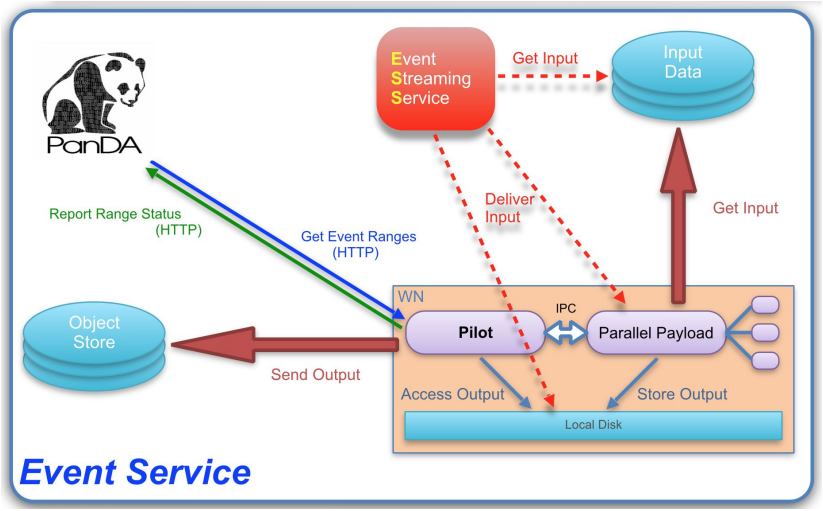
What is Event Service (ES)



- Pilot delivers **fine-grained workload to running payload application** in real time
  - **Workload:** Event Ranges
  - **Event Range:** identifier string with Input File ID, event positions within the file, etc.
- Payload application**:** AthenaMP with a special configuration for the ES
- Payload directly reads input files for the event data
  - Either **local or remote** inputs
- The payload makes a **sequence of fine-grained output files**
  - One output per input event range

- Event Service outputs are first stored into **Object Stores** (or to normal storages)
  - Final outputs are produced later by **dedicated merge jobs**
- Missing component: Event Streaming Service. Intelligent asynchronous delivery of the input data to worker nodes (in early design/prototype phase)

What is Event Service (ES)



- Pilot delivers **fine-grained workload to running payload application** in real time
  - **Workload:** Event Ranges
  - **Event Range:** identifier string with Input File ID, event positions within the file, etc.
- Payload application**:** AthenaMP with a special configuration for the ES
- Payload directly reads input files for the event data
  - Either **local or remote** inputs
- The payload makes a **sequence of fine-grained output files**
  - One output per input event range

- Event Service outputs are first stored into **Object Stores** (or to normal storages)
  - Final outputs are produced later by **dedicated merge jobs**
- Missing component: Event Streaming Service. Intelligent asynchronous delivery of the input data to worker nodes (in early design/prototype phase)

How much Event Service (wrt Normal)

- Simul only for now
  a. “Simple”, well suited case
  b. Highest return: largest part of our CPU usage
- EventService vs normal:
  a. Number of jobs: up to 6% are ES
  b. Walltime: ES is ~3% of total Simul
  c. Ramping ES slowly

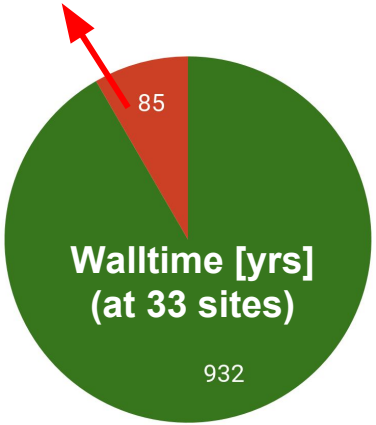
- ES framework is scaling well (running jobs are comparable to the “standard” one)
- **We are grabbing a lot of single core resources which might have been wasted otherwise**

Wall time : Event Service Failures

Simul with EventService

Failures: 8% walltime:

- of which 68% merging failures

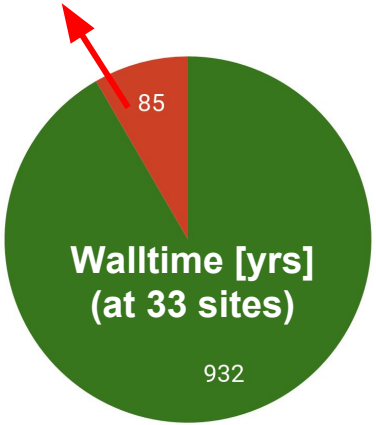


Event Service : Failures analysis

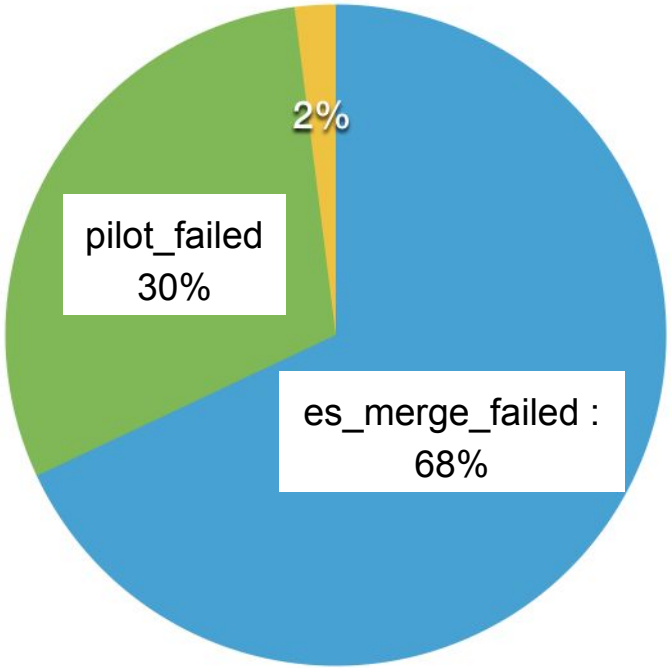
Simul with EventService

Failures: 8% walltime:

- of which 68% merging failures



Event Service Failures

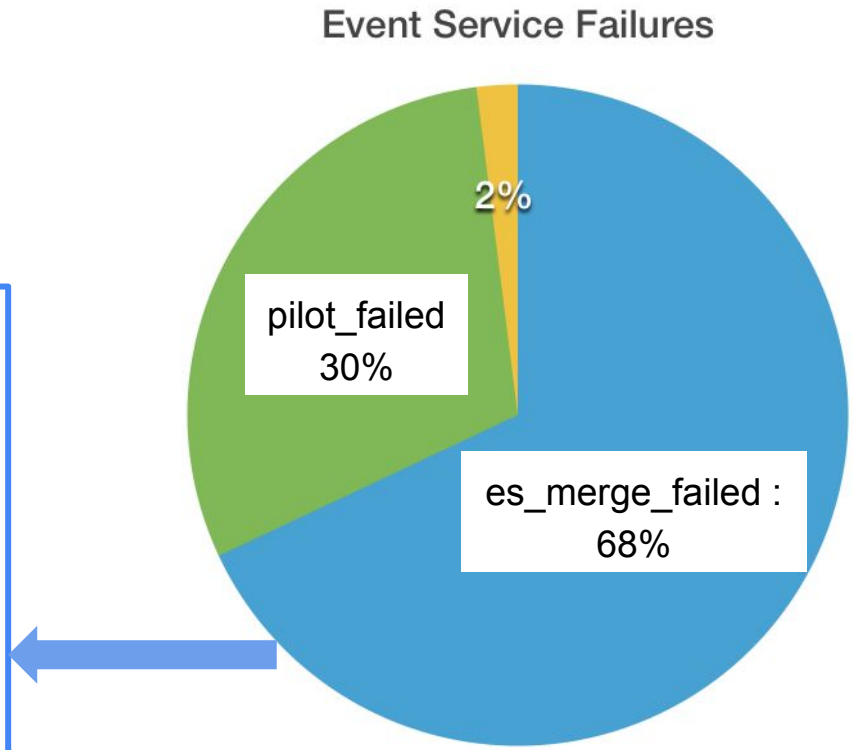


Event Service : Failures analysis

es_merge_failed : Mainly stageout errors proposed actions

Postpone/sleep esmerge jobs if a site with pre-merge files is not online

When an ES merge job fails, it can tar all successfully downloaded files, upload the new tar files and use it to replace all small files----may only reproduce files cannot be downloaded



Event Service : Failures analysis

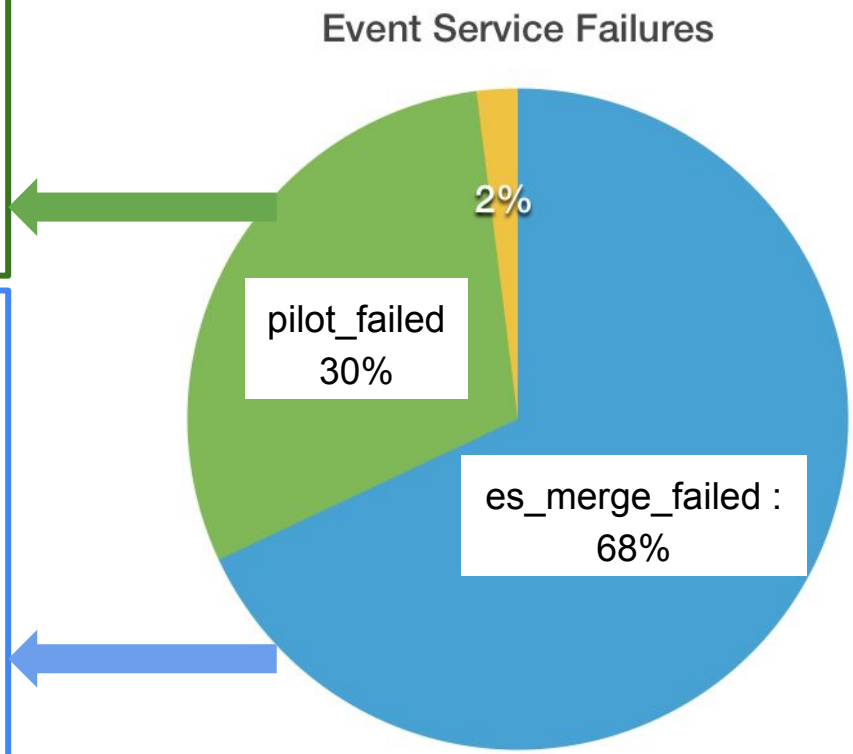
pilot_failed : Mainly stageout errors

proposed actions :
In case of primary storage failures, stageout to failover storagesAlready implemented, in prod testing on few sites

es_merge_failed : Mainly stageout errors
proposed actions

Postpone/sleep esmerge jobs if a site with pre-merge files is not online

When an ES merge job fails, it can tar all successfully downloaded files, upload the new tar files and use it to replace all small files----may only reproduce files cannot be downloaded



Event Service : Failures analysis

pilot_failed : Mainly stageout errors

proposed actions :
In case of primary storage failures, stageout to failover storagesAlready on few sites

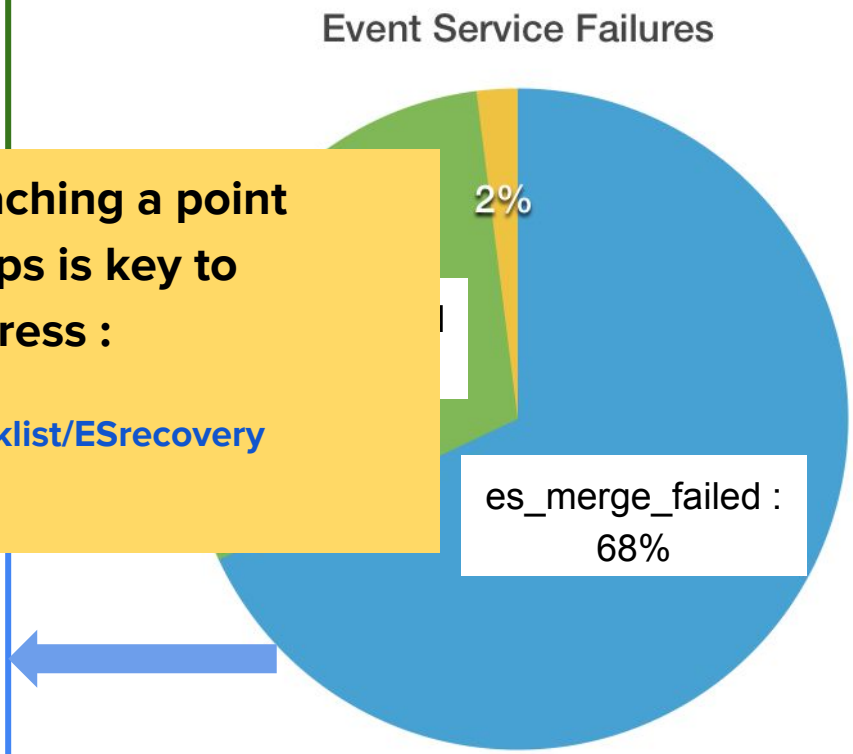
es_merge_failed : Ma
proposed actions

Postpone/sleep esmerg
pre-merge files is not or

When an ES merge job fails, it can tar all successfully downloaded files, upload the new tar files and use it to replace all small files----may only reproduce files cannot be downloaded

In any case we are reaching a point where automate ES Ops is key to continue making progress :

⇒ **Automatic ESblacklist/ESrecovery**



From jaroslava.schovancova@cern.ch

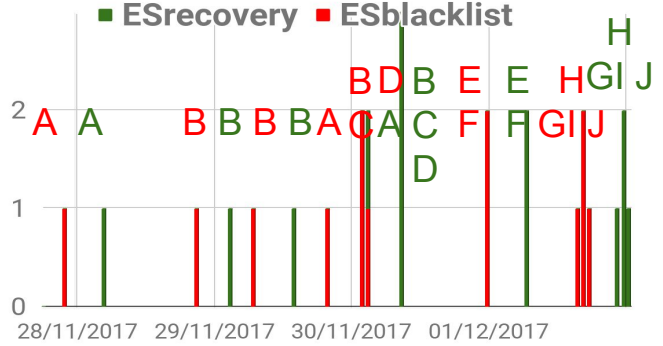
Automatic ESblacklist/ESrecovery

- ESblacklist: ESfailures ⇒ disable ES for the resource

ESblacklist: >50% jobs ESfailed in the past 2 hrs ⇒ disable PQ for ES jobs
ESrecovery: re-enable ES after 6 hours
EXHAUSTED: ESblacklist 4x within 24 hrs: no recovery, alert ES ops team

Transient vs. permanent issue

- ESblacklist/ESrecovery in action
  - A-J ... anonymized PanDA queues that run&fail ES jobs
  - Red line: ES was disabled for a PQ
  - Green line: ES was re-enabled for a PQ
  - Magenta line (absent): ES was disabled 4x within 24 hrs, so instead of auto-recovering, we alert ES ops team: an expert has to check.
  - ES Ops experts do not have to deal with transient errors anymore, can focus on permanent issues!



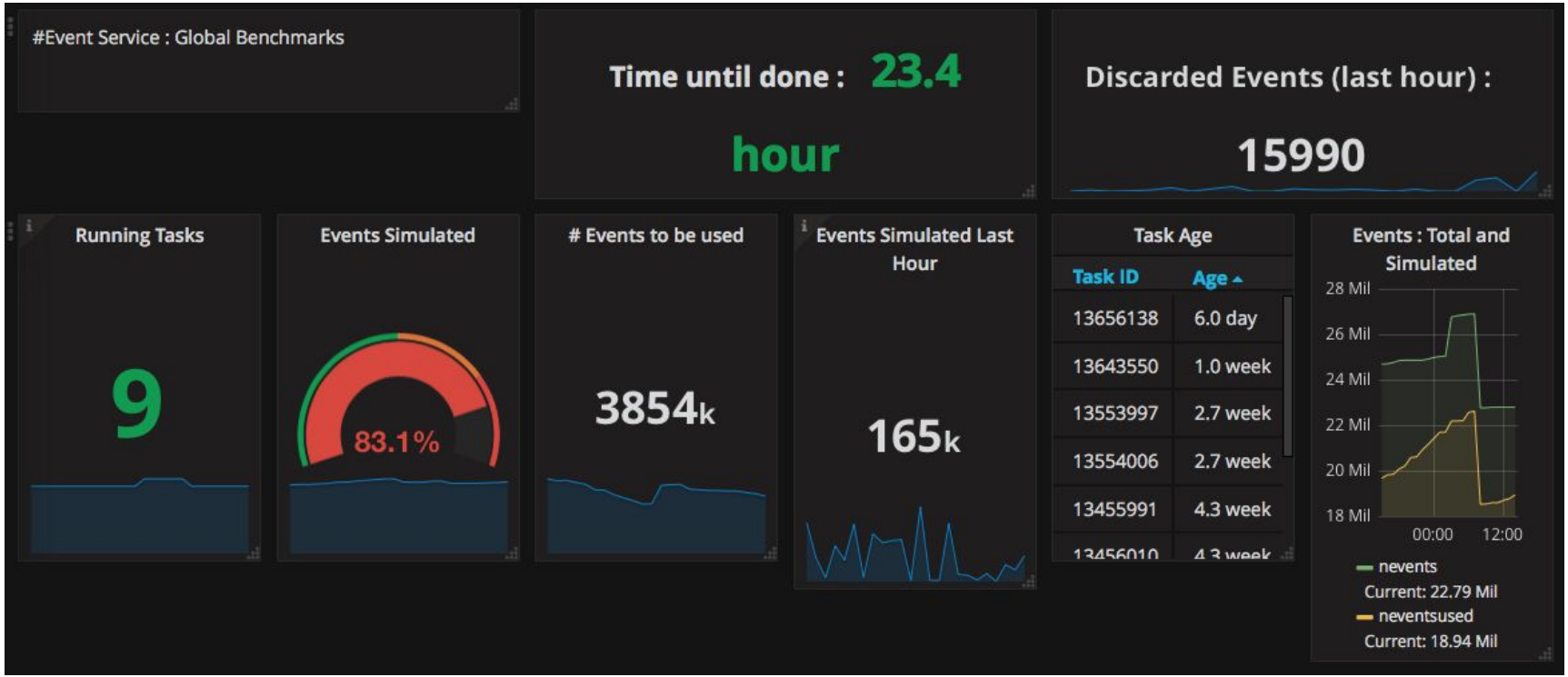
- All ESblacklist/ESrecovery based on ES job counting and study error categories

⇒ we can re-use this type of automation for other workflows! *#GreenComputing*

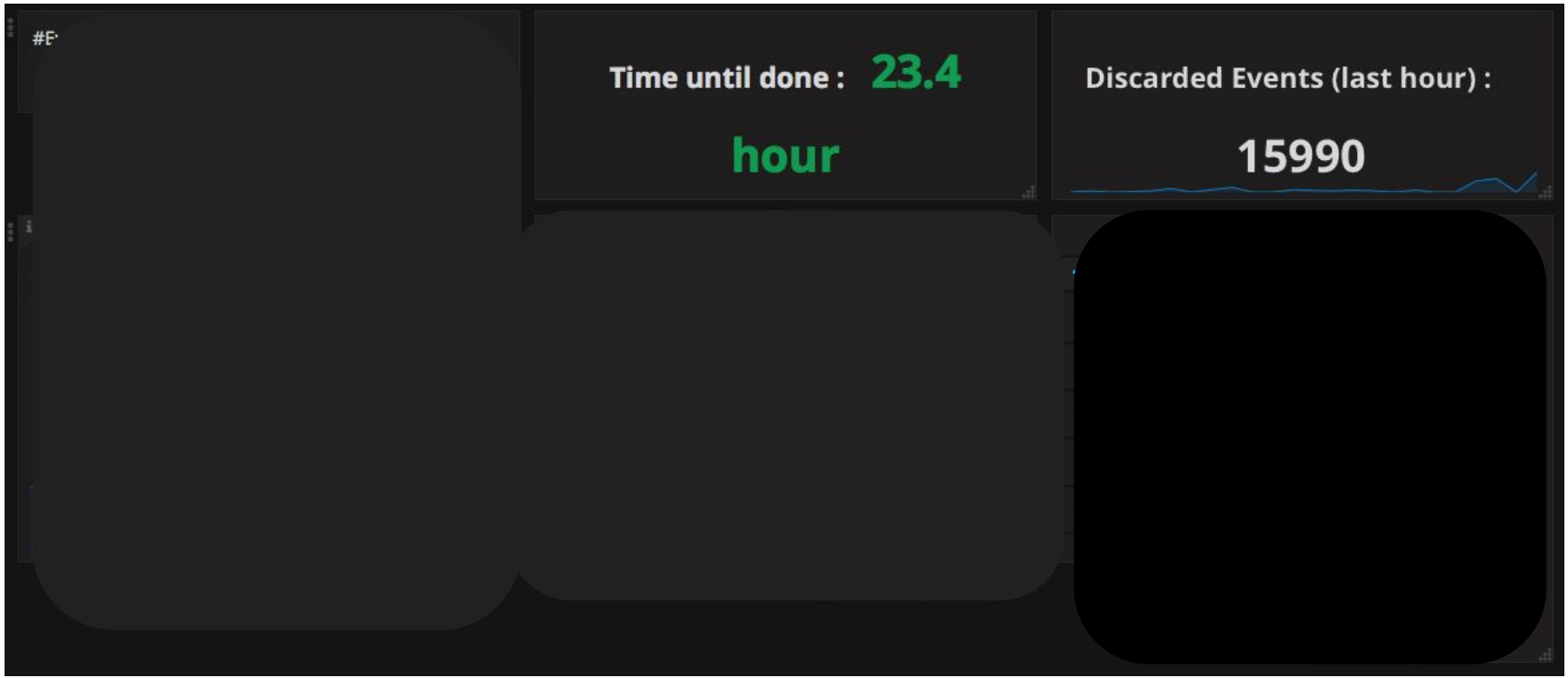
Monitoring tools in Bigpanda

- A lot of functions have been implemented for ES in bigpanda.
- World view
  - https://bigpanda.cern.ch/dash/production/?cloudview=world&es=true
- All tasks
  - https://bigpanda.cern.ch/tasks/?eventservice=eventservice&display_limit=300&days=60
  - https://bigpanda.cern.ch/runningprodtasks/?processingtype=simul&eventservice=1&days=30
- For every tasks
  - Jobset retries
  - In a jobset, consumers retries

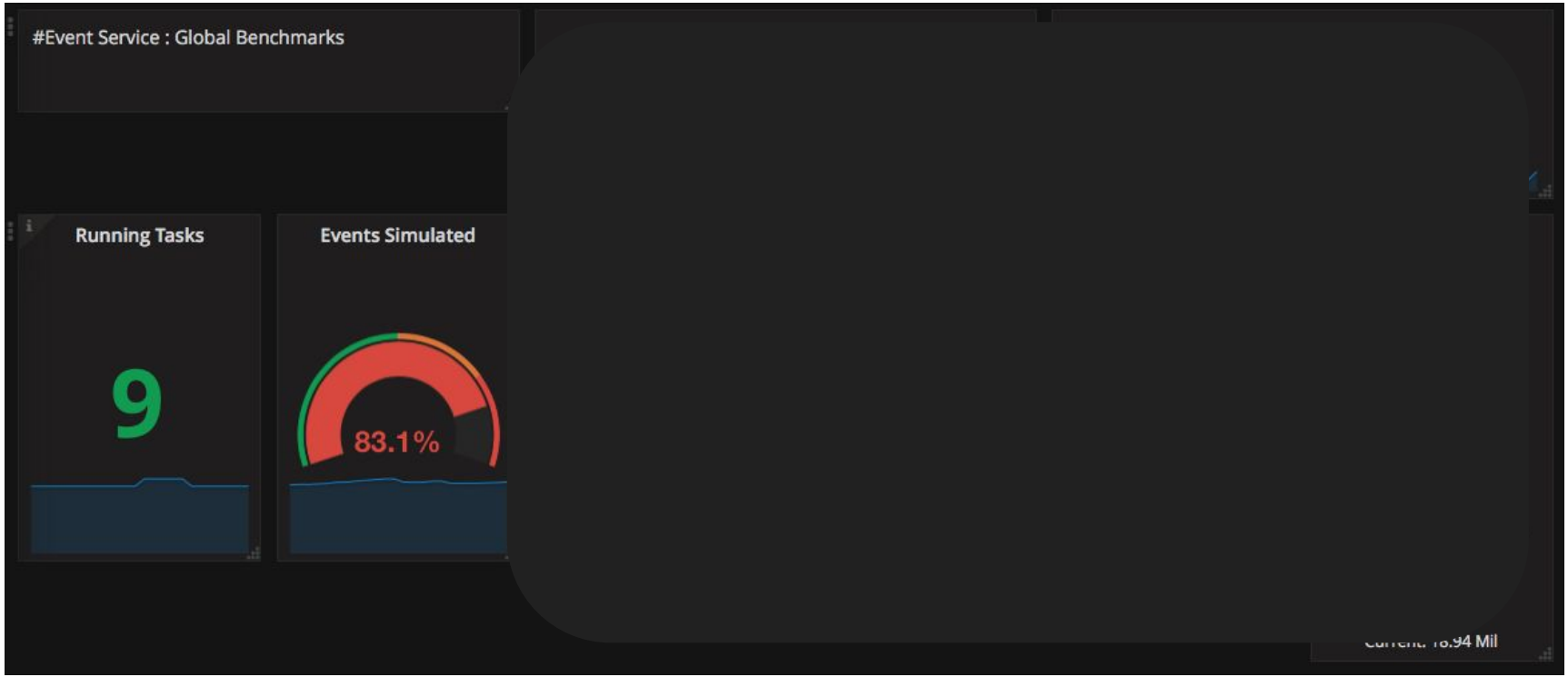
Monitoring tools : taylored dashboard



Monitoring tools : taylored dashboard

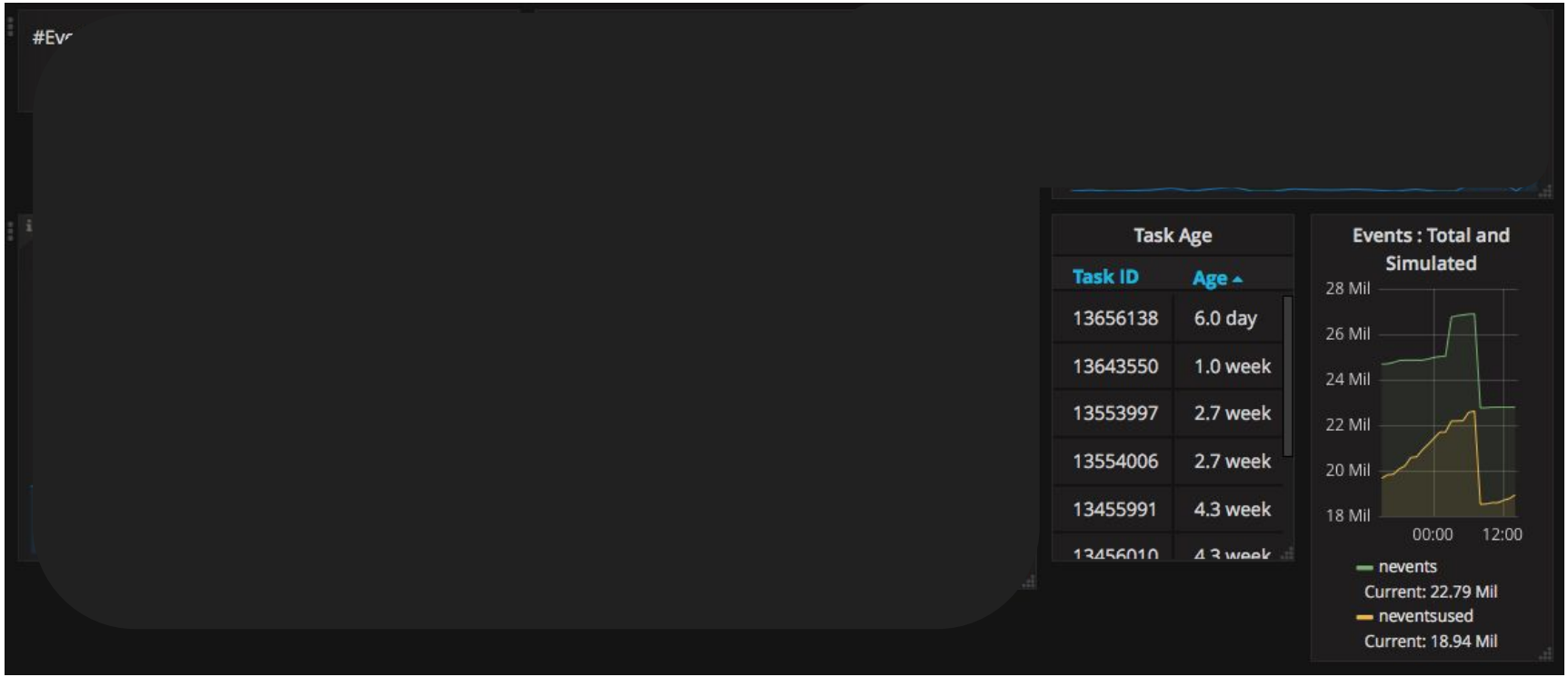


Monitoring tools : taylored dashboard

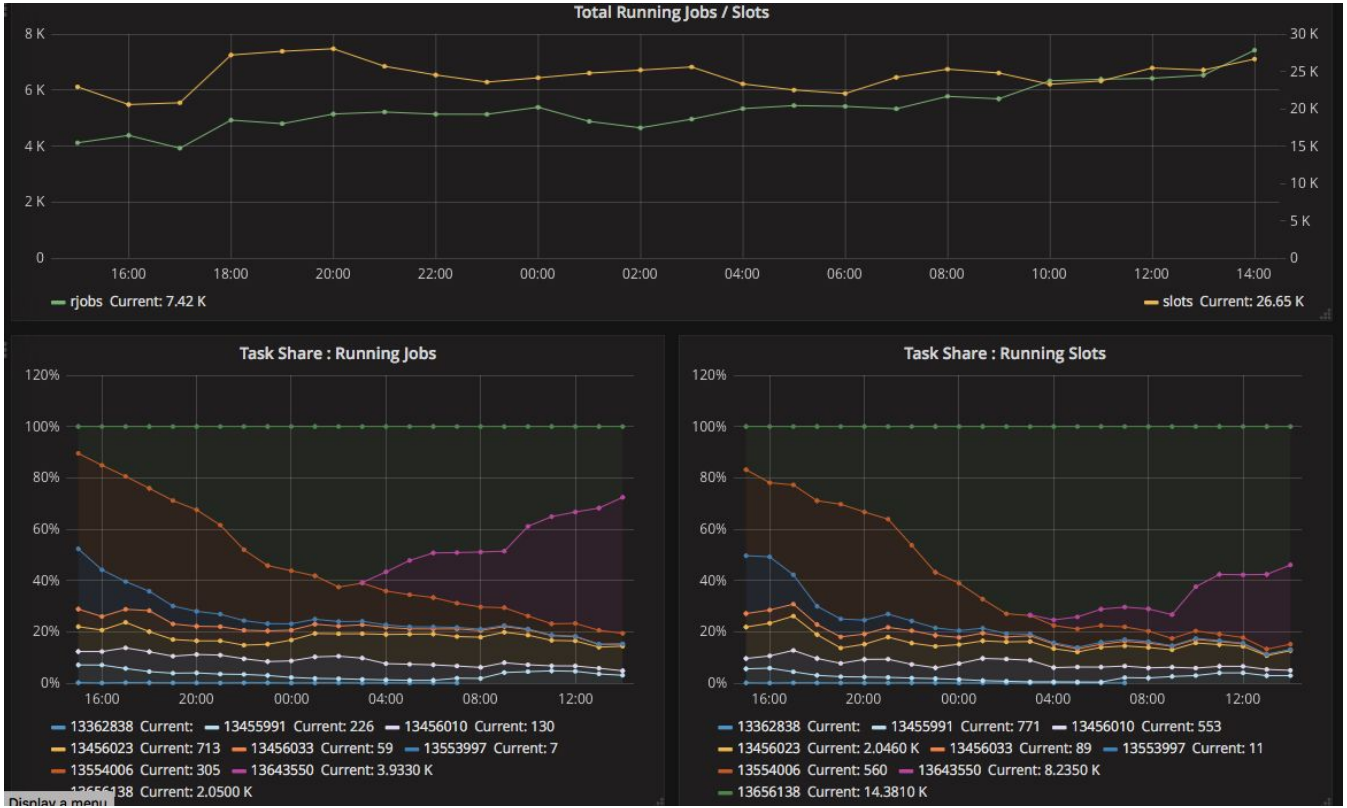


Monitoring tools : taylored dashboard

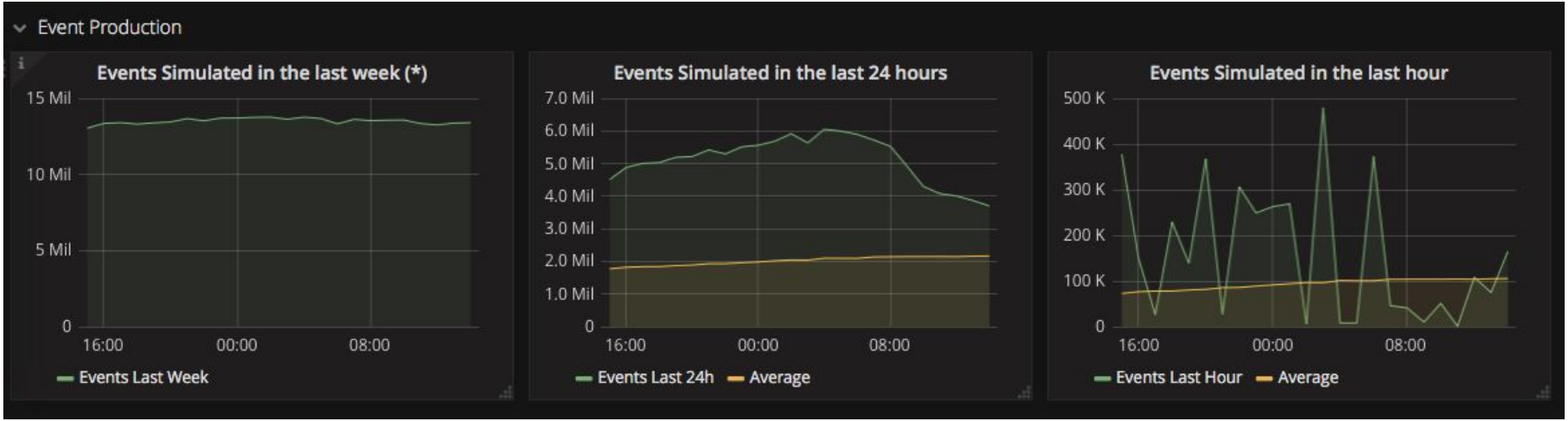
Monitoring tools : taylored dashboard



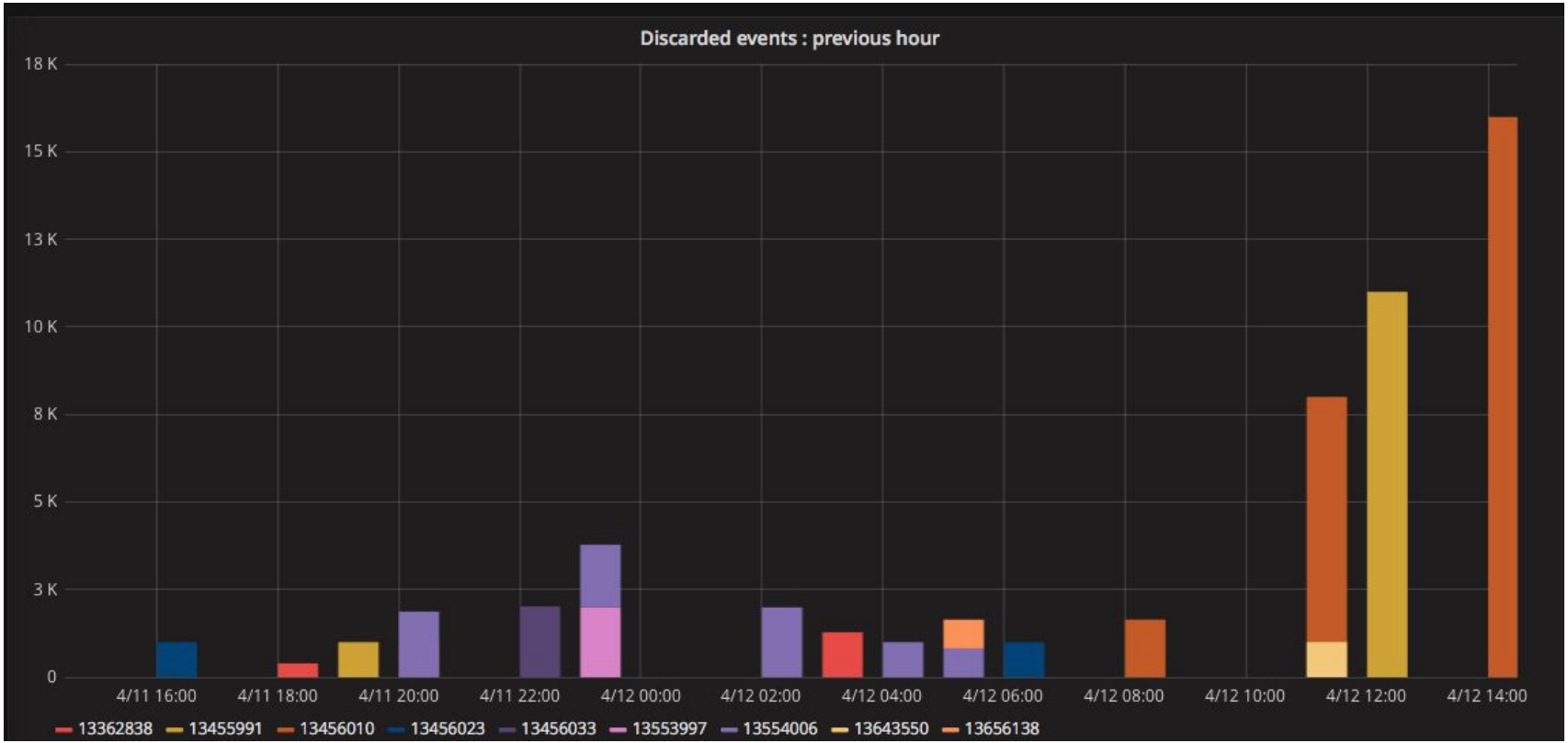
Monitoring tools : taylored dashboard



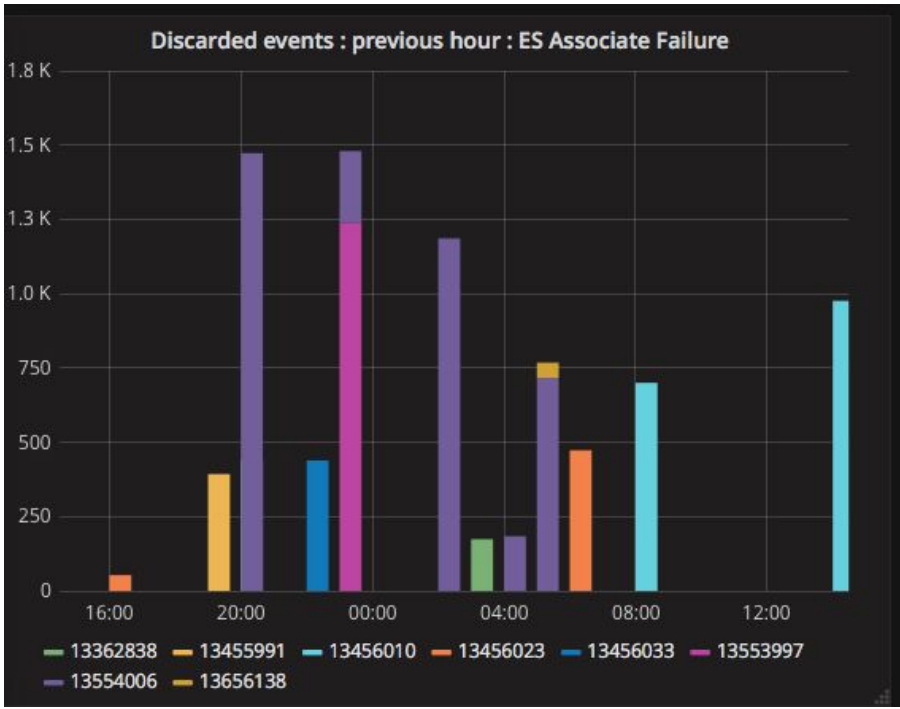
Monitoring tools : taylored dashboard

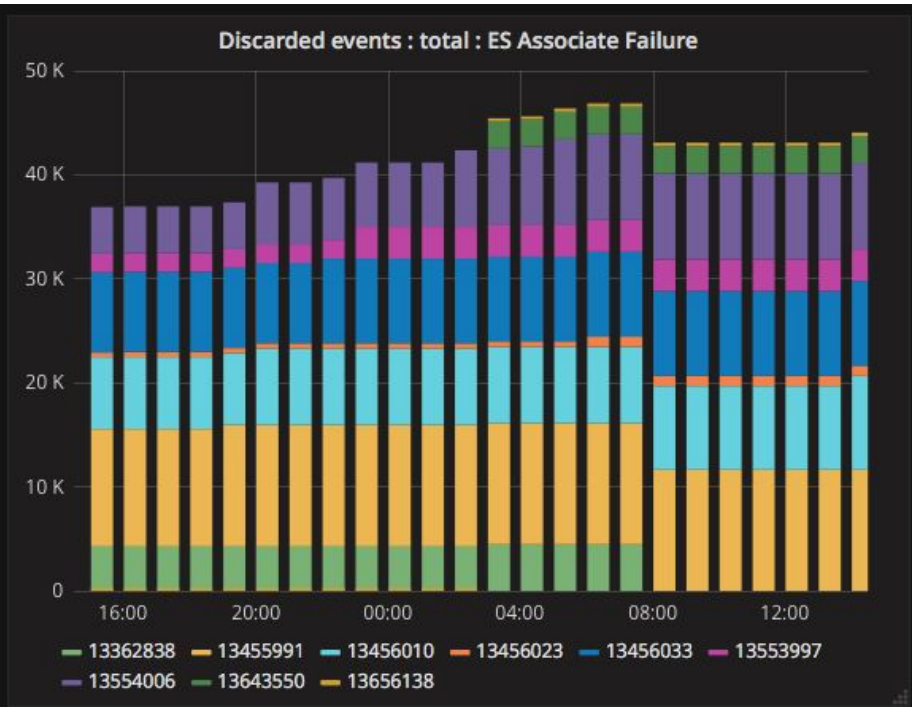


Monitoring tools : taylored dashboard



Monitoring tools : taylored dashboard





Monitoring tools : taylored dashboard

We have seen:

Event service benefits

How event service works

How event service compares with Normal Simulation Jobs

Where it fails and how we want to solve it

So, what’s next?

What's next

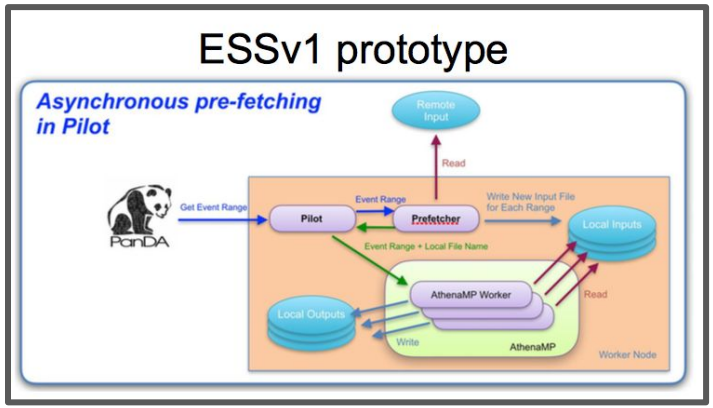
- Keep expanding grid roll-out
  - Improving on storage interactions, operational load, monitoring, ...
- Task lifetime improvements
  - To improve the lifetime from task start to task end.
- ES is being (re)implemented at major HPCs as their workflows are refined
  - NERSC (Cori), ANL (Theta), Oak Ridge (Titan)
- Explore remote WAN access to avoid input pre-placement
  - ES for inputs currently uses the same production workflow as normal jobs, doing input pre-placement
- The next big step presently entering R&D: Event Streaming Service
  - Efficient asynchronous WAN data delivery for simul and ultimately more I/O intensive workloads

Event Streaming Service

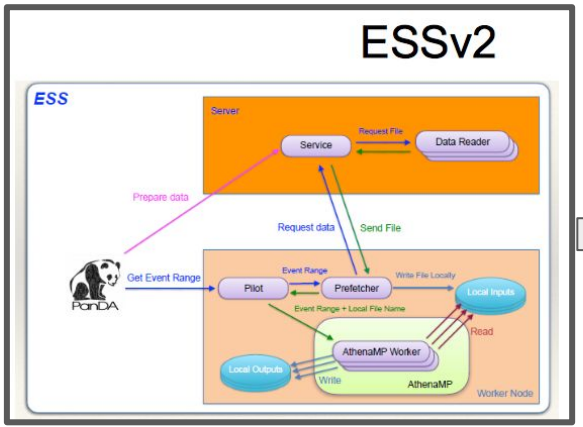
More information : see the interesting talk by Nicolo yesterday

What Event Service is to computing,
Event Streaming Service is to input data transfer

i.e. Instead of transferring the whole input file, it only transmits the range of events needed



First succesfull job sent this week:
https://bigpanda.cern.ch/task/12810722



Disentangles Event Streaming Service from Event Service

More user cases than simulation jobs

Todo lo anterior es donde estamos, pero adonde vamos?

- Event Service es operativo
  - Pero muy limitado y no está claro el camino para crecer...
  - ...parte del crecimiento depende de los Objects Stores disponibles
  - A veces vale la pena dar un paso atras para avanzar mas rapido
- Evolucion :
  - Esta aceptado que Event Service es una prioridad para el futuro
  - Pero para llegar a el es necesario rediseñar PANDA
    - Hacerlo mas “*event aware*”
    - Tadashi pone una timescale de un año
- Pero a corto termino?
  - Arreglar los problemas que tenemos ahora : e.g. *discarded events*
  - Y sacar lecciones para el rediseño de Panda

The END

Event Service Schematic

Implementations realizing the architecture are in green