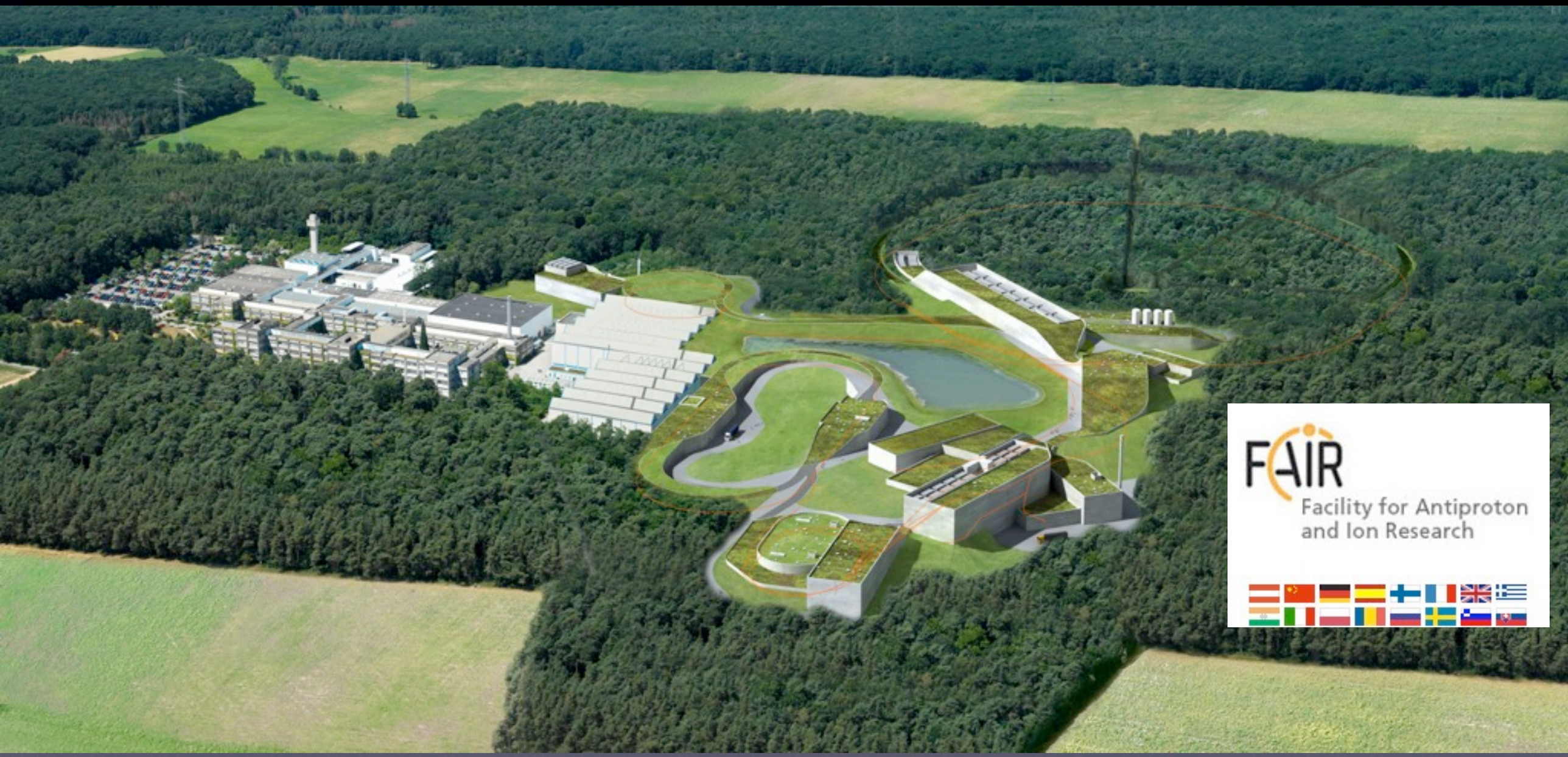
Computing On Demand: Analysis in the Cloud

Anar Manafov, GSI Darmstadt

GSI: a German National Lab for Heavy Ion Research

FAIR: Facility for Ion and Antiproton Research ~2018



GSI computing 2011

- ALICE T2/T3
- HADES
- 2800 (LSF) + 1800 (SGE) Cores
- ~10000 Cores at the end of 2011
- 2.1 PByte Lustre

FAIR computing 2018

- CBM
- PANDA
- NuSTAR
- APPA
- LQCD
- 300000 Cores
- 40 PB Disk and 40 PB Archive

Computing On Demand

resources with or without RMS

Hardware

Computing On Demand

Software

tools

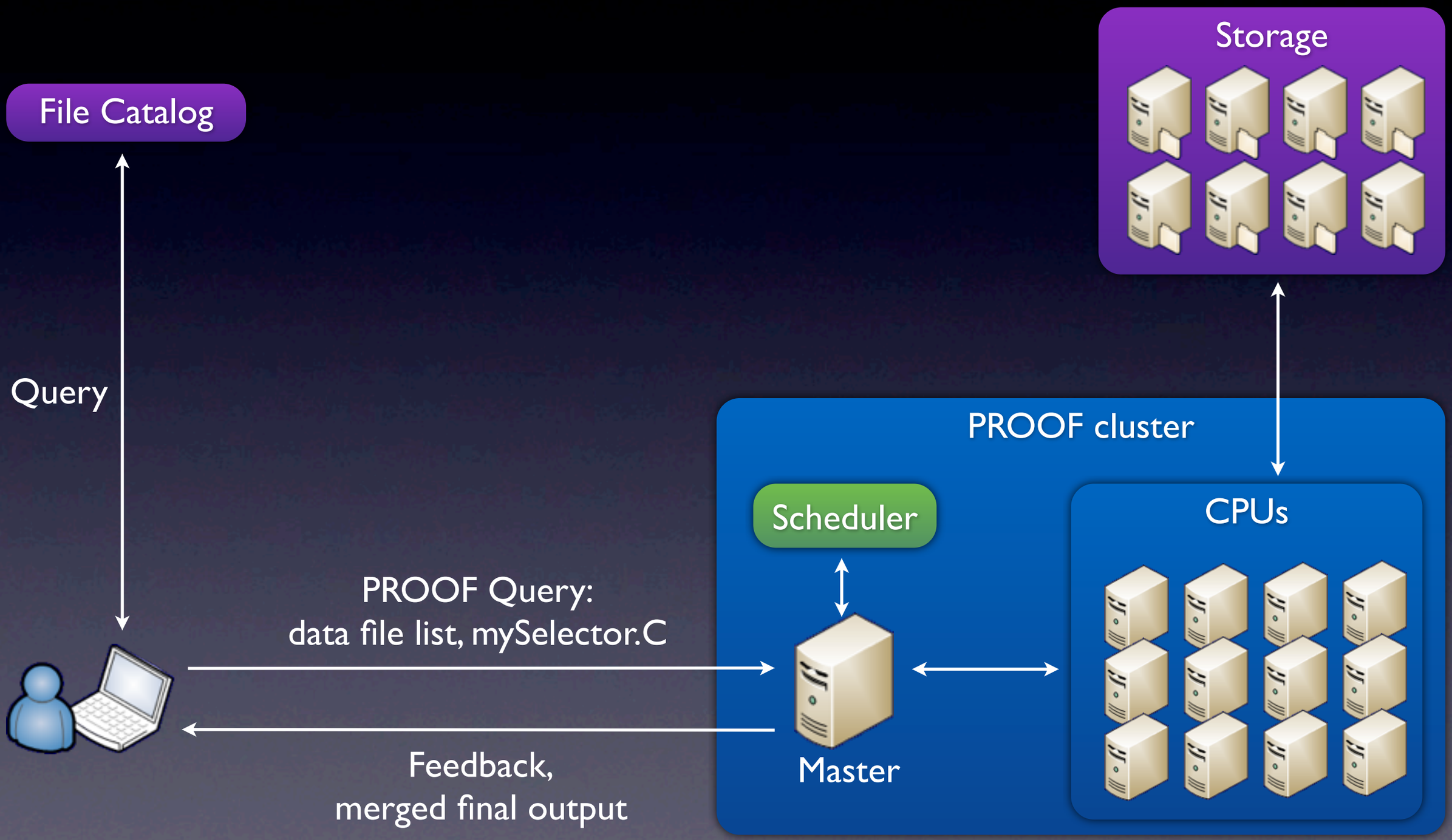
analysis software

Tools

HEP Data Analysis

Typical HEP analysis needs a continuous algorithm refinement cycle

PROOF


Storage
File Catalog
Query
PROOF cluster
Scheduler
CPUs
PROOF Query:
data file list, mySelector.C
Master
Feedback,
merged final output

Cons of pre-installed approach

- One user can disturb other users.
- From time to time admin. interventions are needed.
- There is only one ROOT/xrootd version for PROOF services.
- There is a master node limitation.

Is PROOF developing in a wrong direction?

instead of concentrate on real PROOF features, it tries to substitute a batch system

Motivation

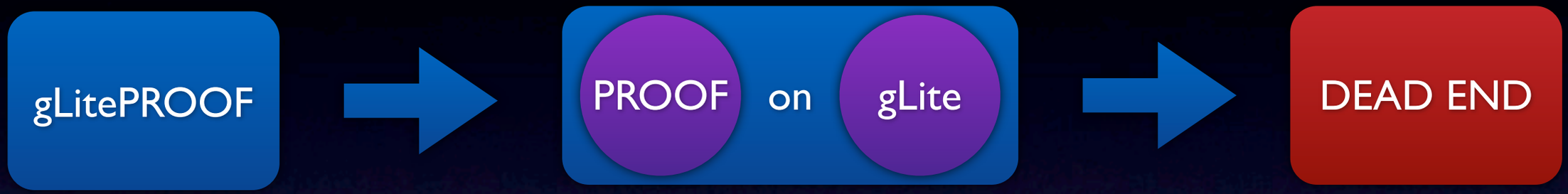
The main idea is to have our resources shared between local jobs (“batch”), Grid jobs and PROOF.
We don’t want to waste our resources and to let machines stay without a load.

The final product also should:

- not require administration,
- not require su privileges,
- be able to work on different RMS,
- provide automation and control of PROOF,
- be stable and private.

History

History



History


gLitePROOF
PROOF
on
gLite
DEAD END


by manipulating processes priority with nice
"static" PROOF cluster
PROOF
+
LSF
Too many admin. problems

PoD

PROOF on Demand

3 steps to set your private PROOF cluster up

Start
PoD Server

3 steps to set your private PROOF cluster up

3 steps to set your private PROOF cluster up

3 steps to set your private PROOF cluster up

step #1

pod-server start

step #2

pod-submit -r [lsf | ge | pbs | condor] -q my_queue -n 100

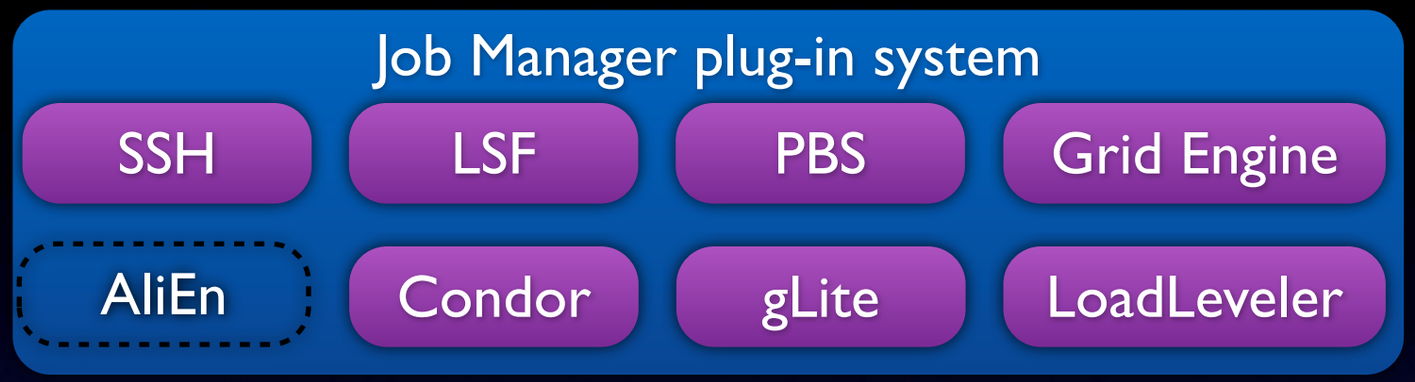
or

pod-ssh -c pod_ssh.cfg --submit

step #3

pod-info -n

Different job managers



PoD is shipped with a number of plug-ins, which cover all major RMSs, such as local cluster systems and Grid.

If you don't have any RMS, then the SSH plug-in can be used.
The SSH plug-in is also used to setup PROOF clusters on Clouds.

PoD vs “Static” approach

a User

- can entirely control his/her dynamic cluster,
- can setup and use it on demand,
- can dynamically change an amount of workers,
- can select a preferable master host,
- doesn’t need admins to take an action,
- doesn’t disturb other users,
- is free to choose a ROOT version for services.

PoD vs VM based “static”

PoD

- can use firewall protected WNs,
- can combine WNs from different sub-nets,
- doesn’t require additional amin./config. machinery,
- provides managed PROOF cluster.

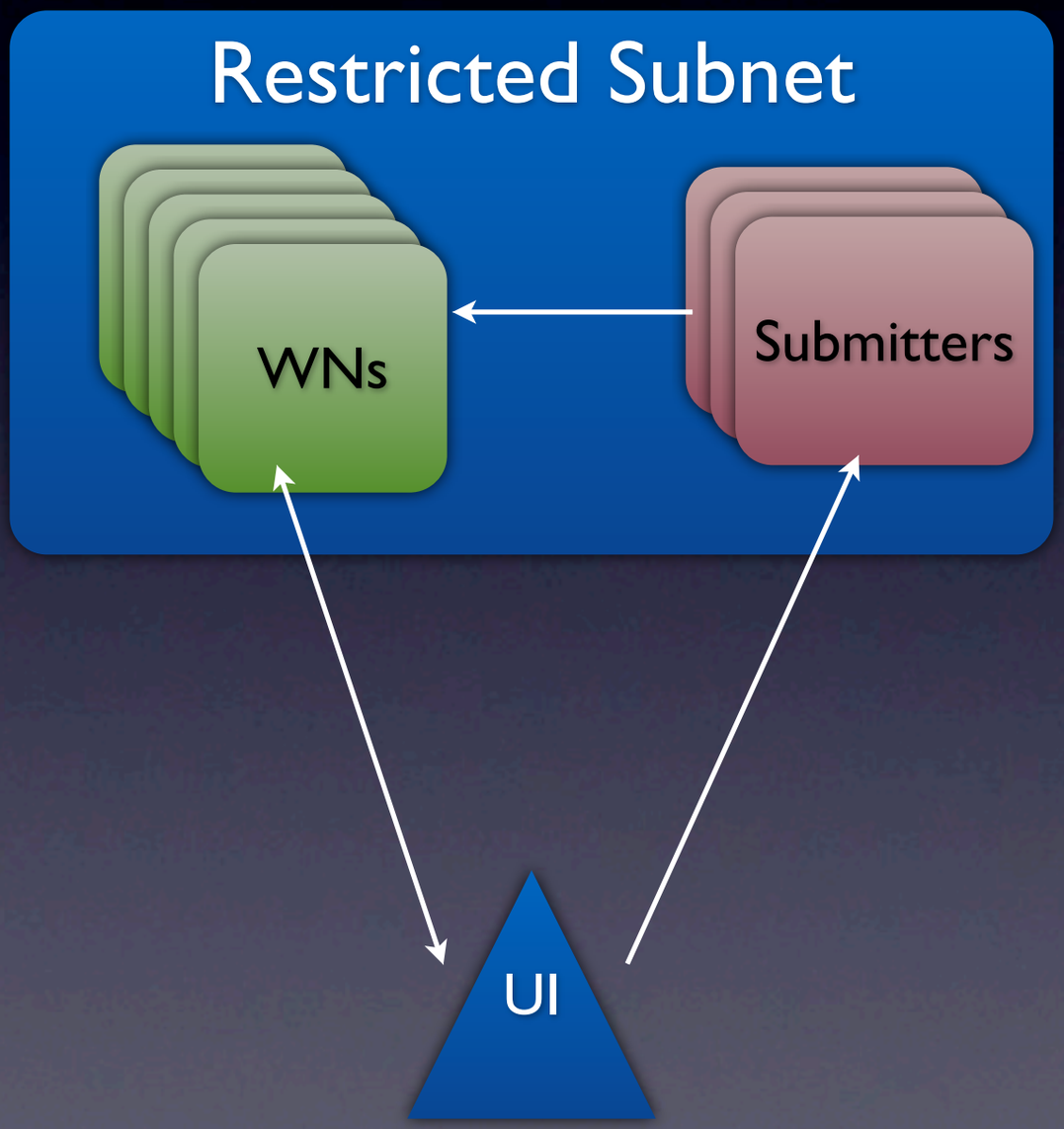
pod-remote: a thin client concept

pod-remote

- helps to control remote PoD instances,
- uses only secured channels for communications,
- automatically creates and maintains SSH tunnels for PROOF and PoD connections.

New PoD development:

Dynamic Master architecture



- use pod-remote to submit PoD jobs,
- all PoD WNs connect back to PoD UI,
- PoD UI dynamically assigns a suitable WN to be a PoD server and a PROOF master,
- all WNs connect to that dynamic PROOF master,
- PoD automatically creates its packet-forwarding tunnel to redirect PROOF traffic between UI and the Master.

PoD: a “Green-PC” concept

The PoD SSH plug-in helps to consume workstations around you.

PoD could be perfectly combined with CernVM, for example, to let users intentionally share cores of their workstations.

Data location

For the dynamic analysis model is the best to keep data remote from WNs, but local for the site.

Interesting study:

ISGC2011, "Investigation of storage options for scientific computing on Grid and Cloud facilities" (Fermi National Accelerator Laboratory, US)

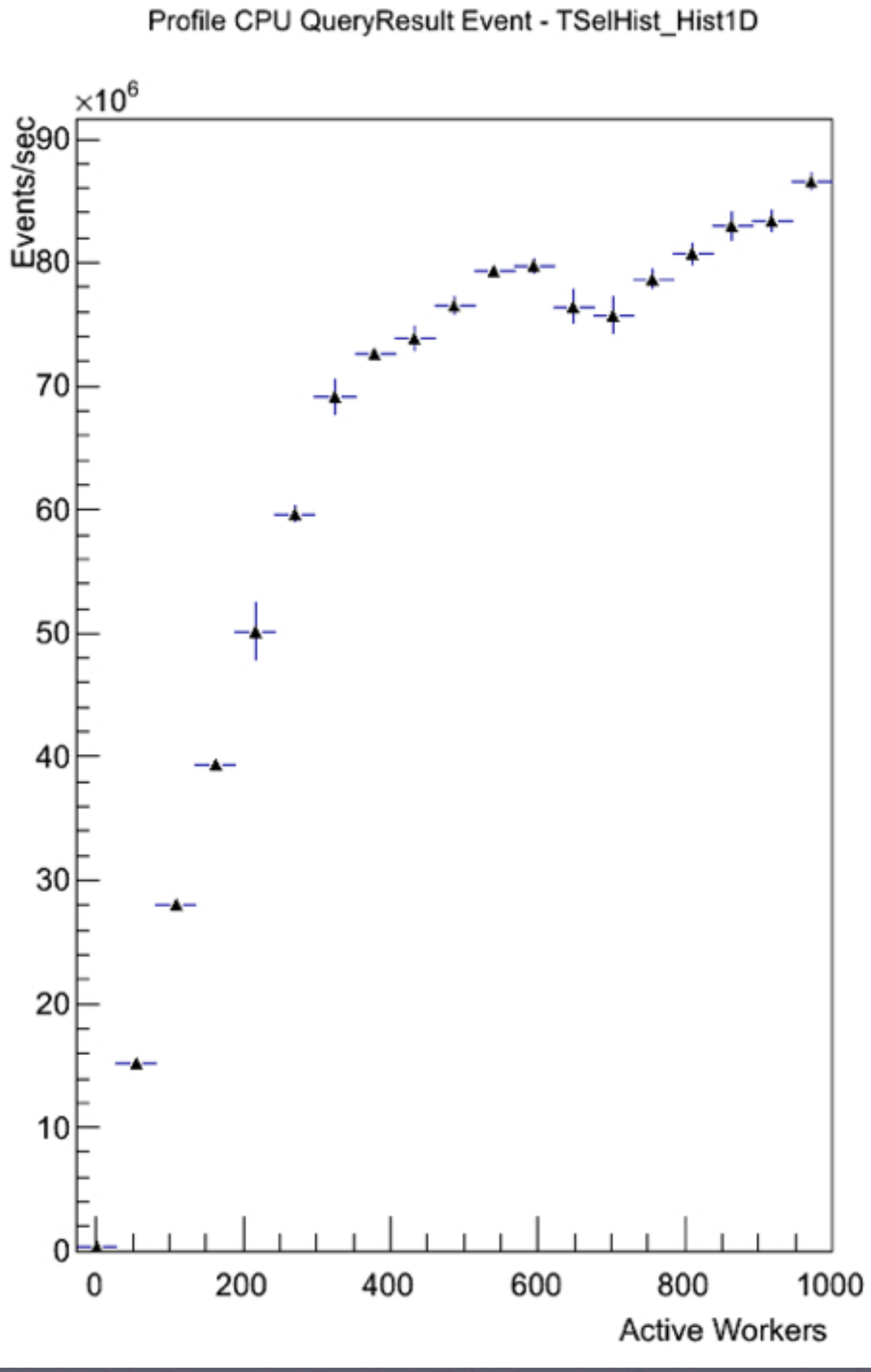
http://www2.twgrid.org/event/isgc2011/slides/GridsandClouds/2/Storage-Evaluation-ISGC-2011.pdf

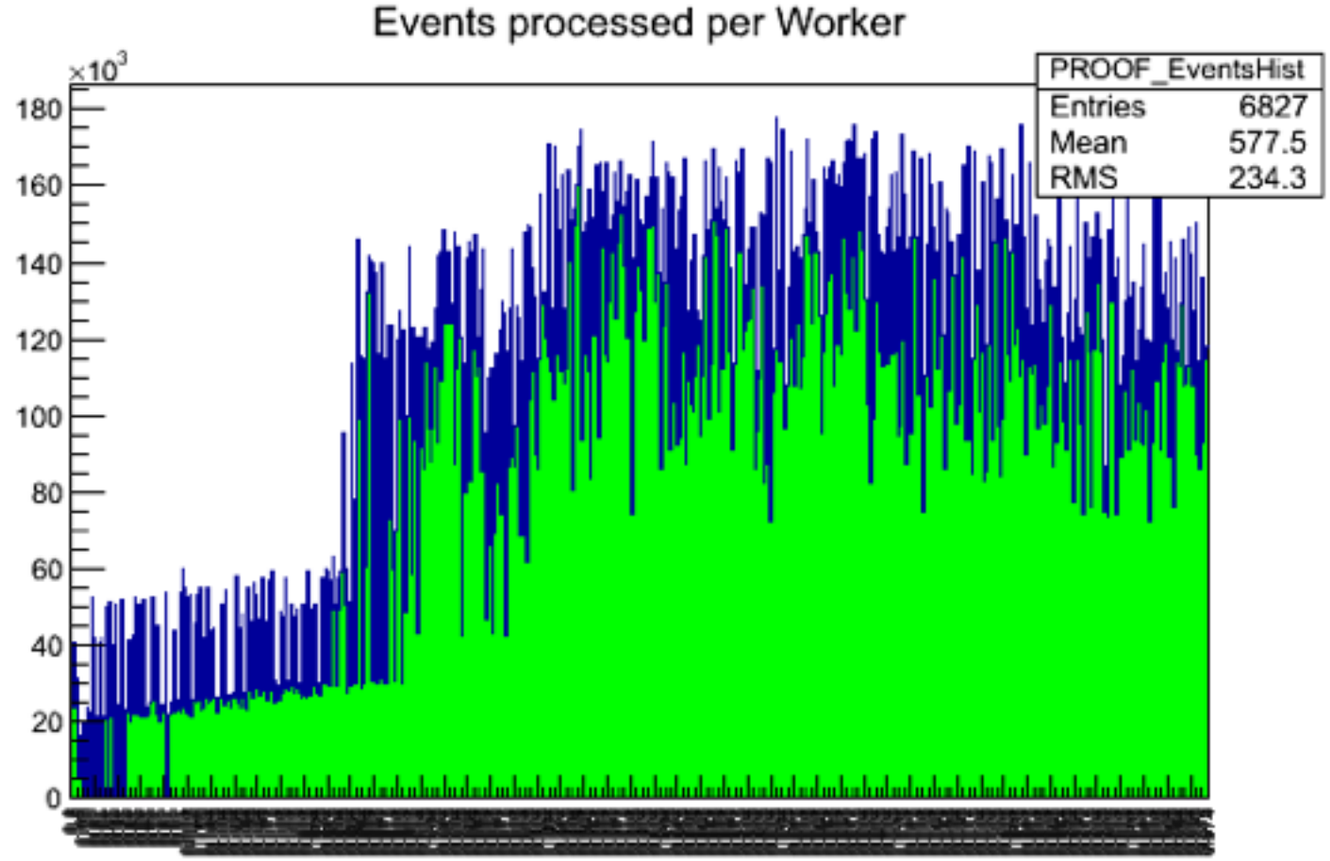
Fitness

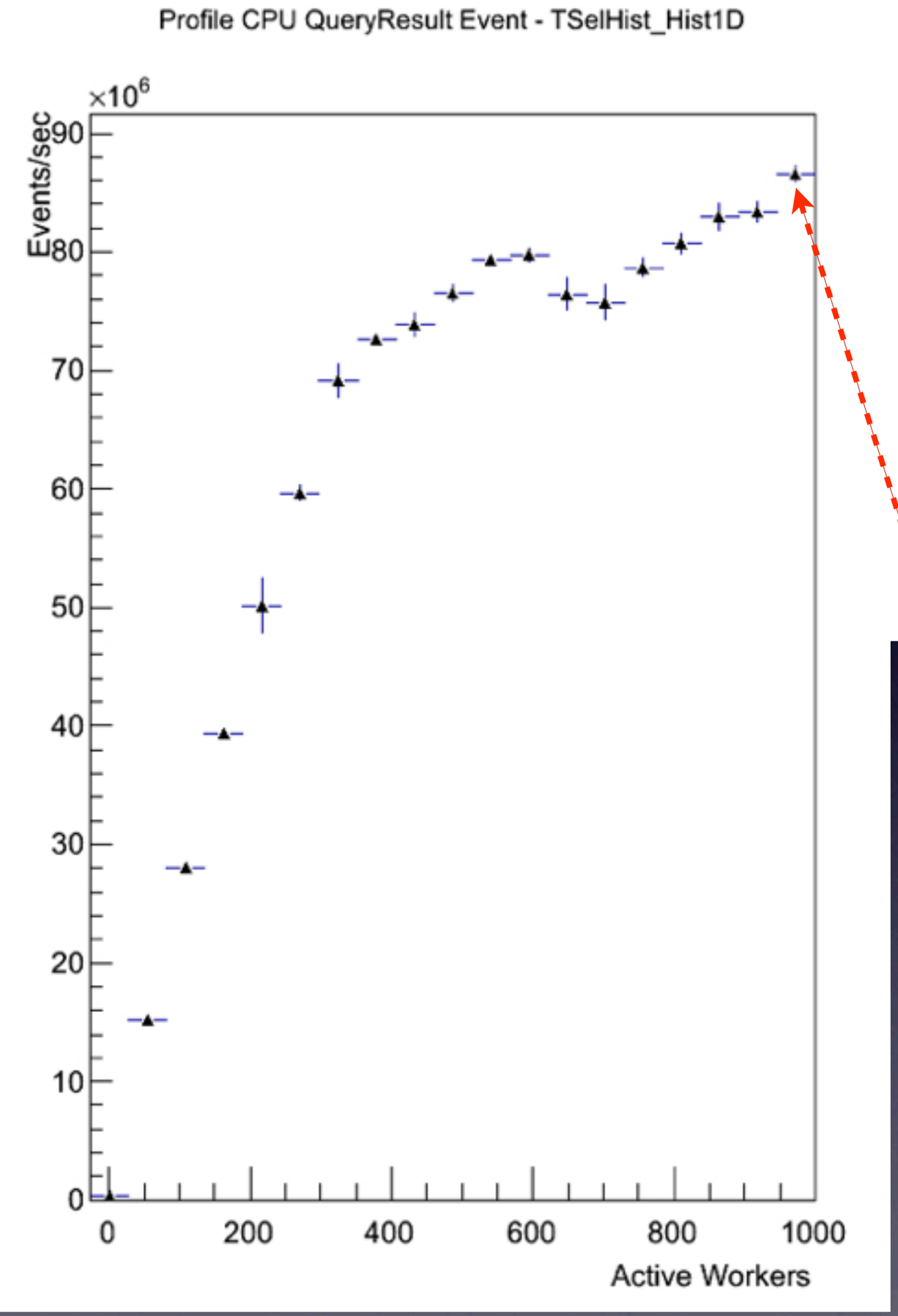
These kind of exercises you can't easily make without PoD!

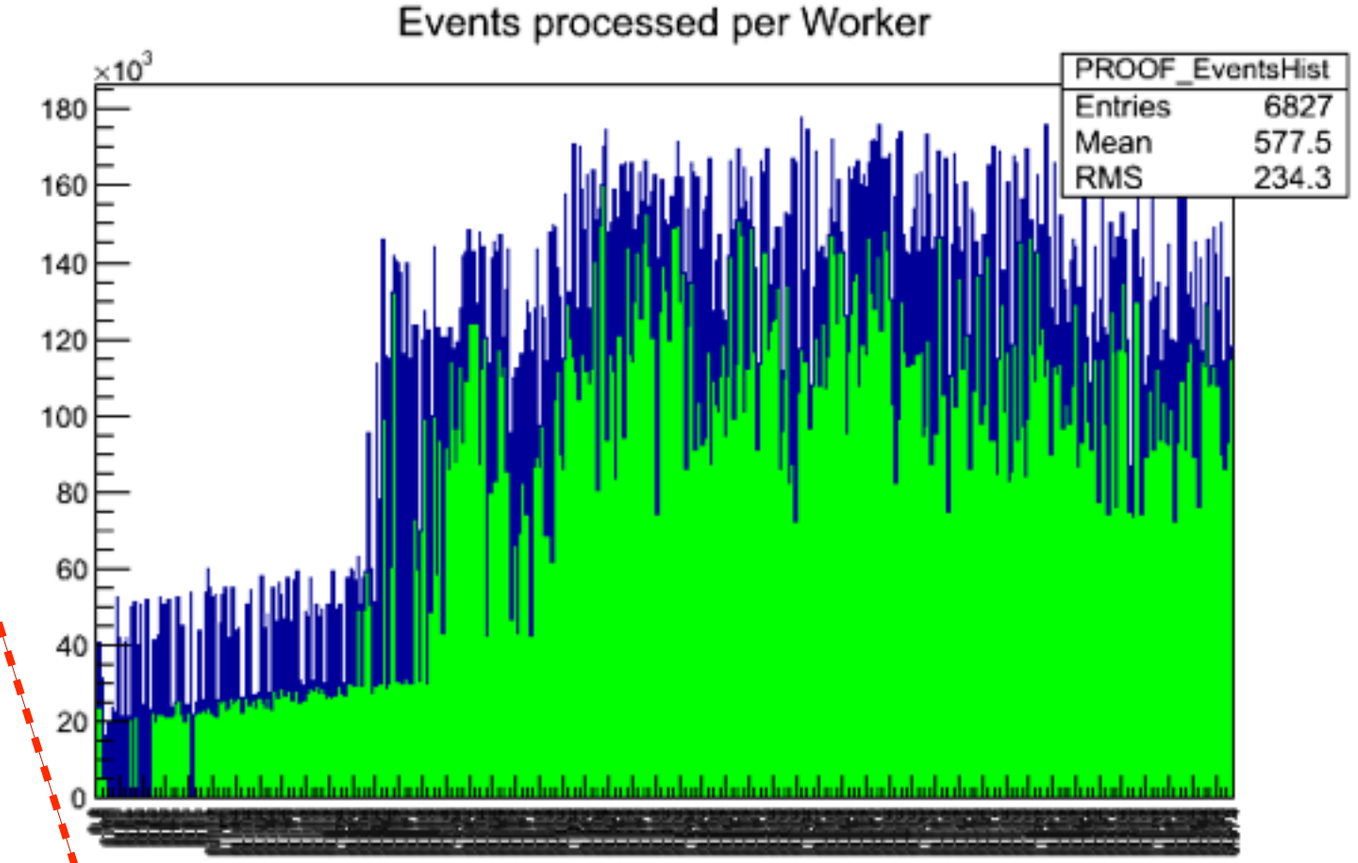
PoD has been used to setup a dynamic PROOF cluster of 1000 WNs on Cloud.

For tests we use
the new PROOF Benchmark Suite.

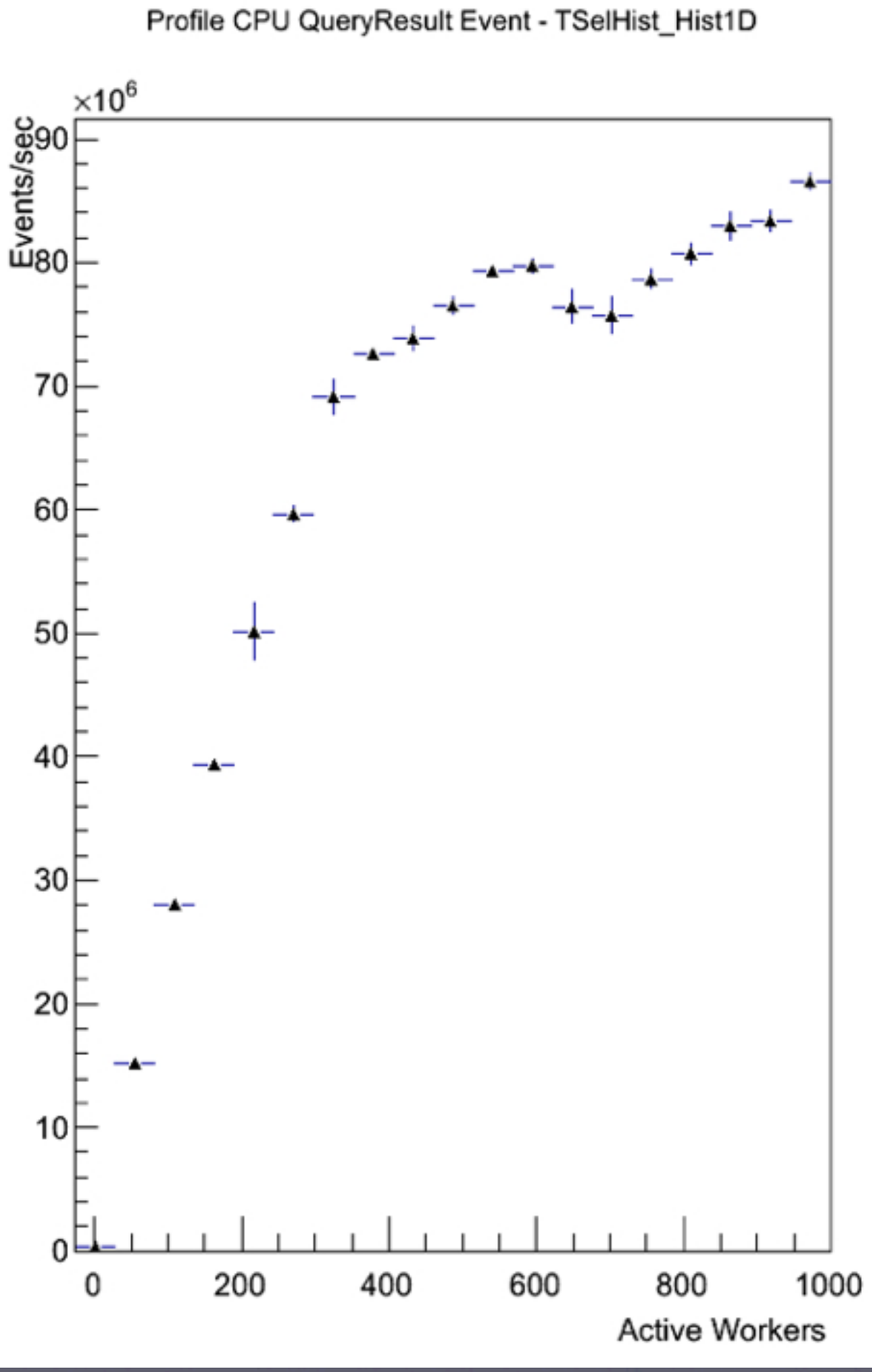

Profile CPU QueryResult Event - TSelHist_Hist1D
×10^6
Events/sec
90
80
70
60
50
40
30
20
10
0
0
200
400
600
800
1000
Active Workers

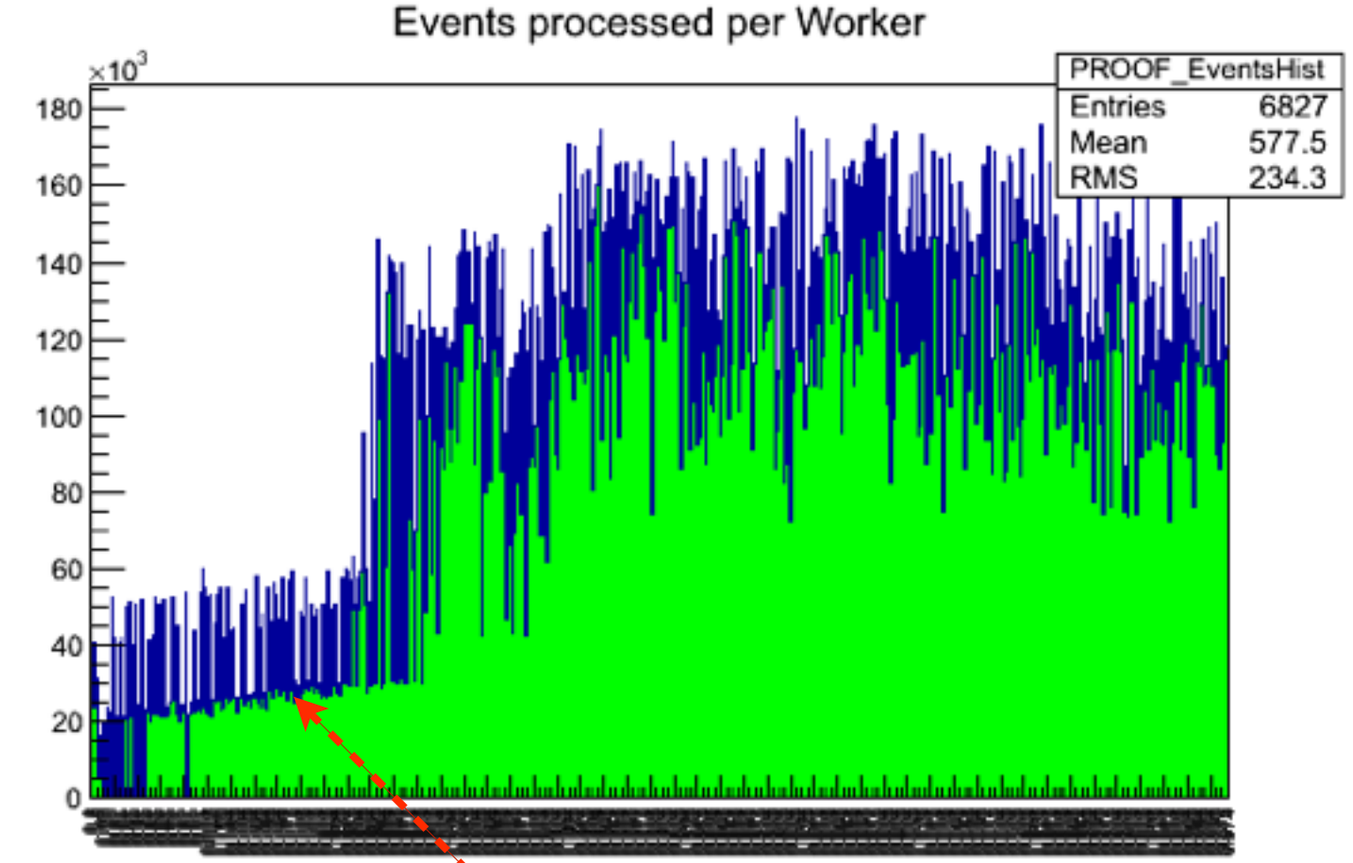

Events processed per Worker
×10^3
180
160
140
120
100
80
60
40
20
0
PROOF_EventsHist
Entries 6827
Mean 577.5
RMS 234.3



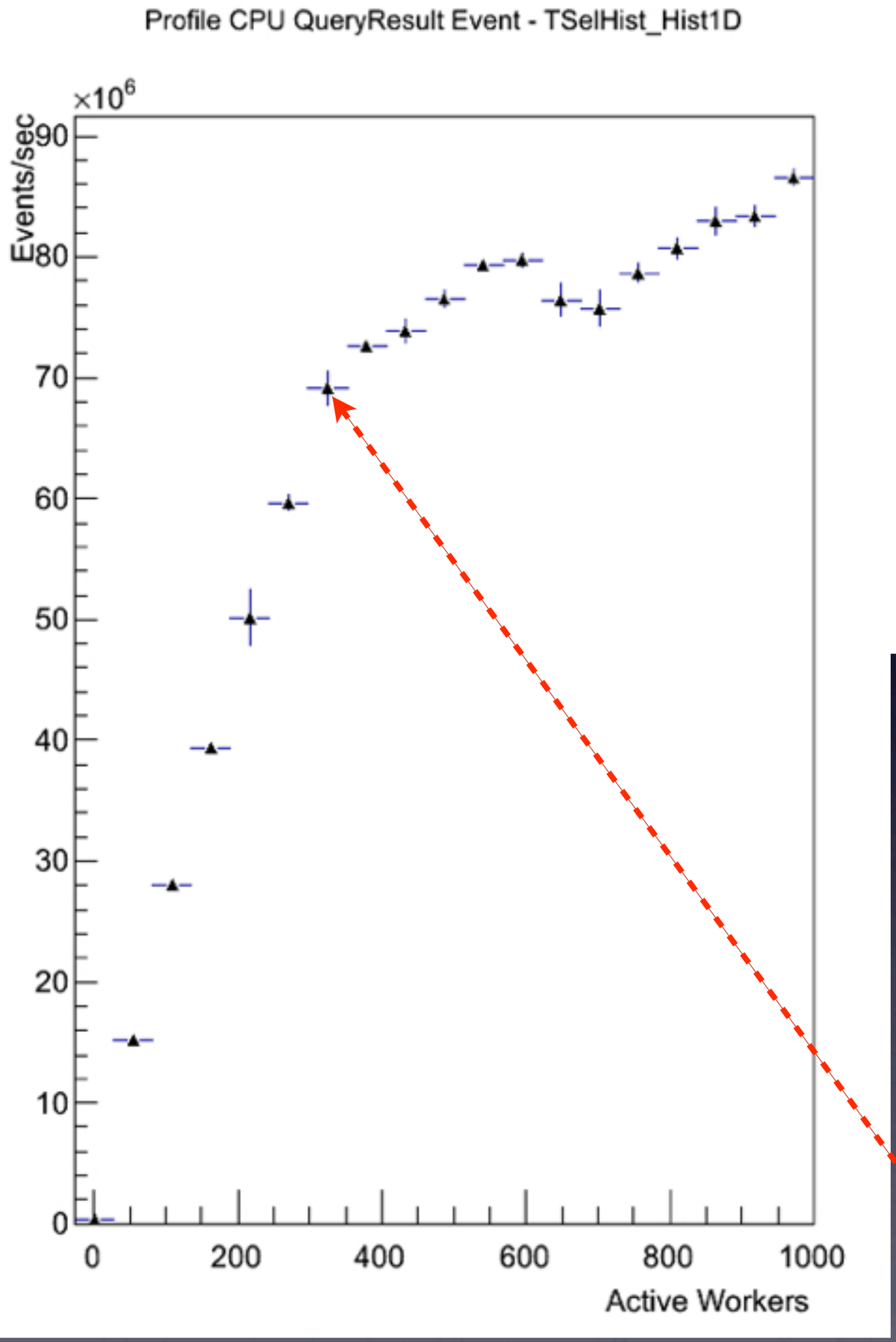


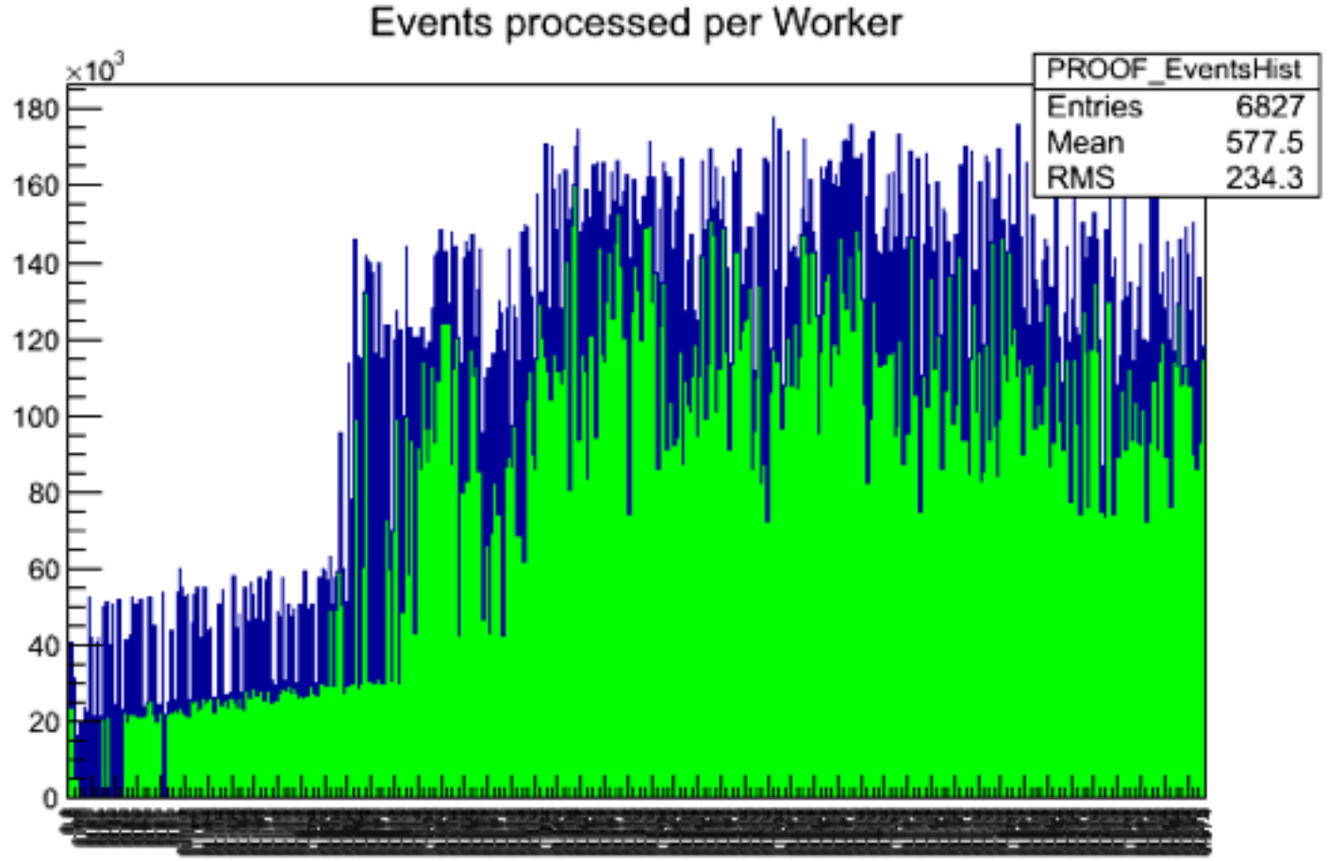
975 WNs/Master is the limit



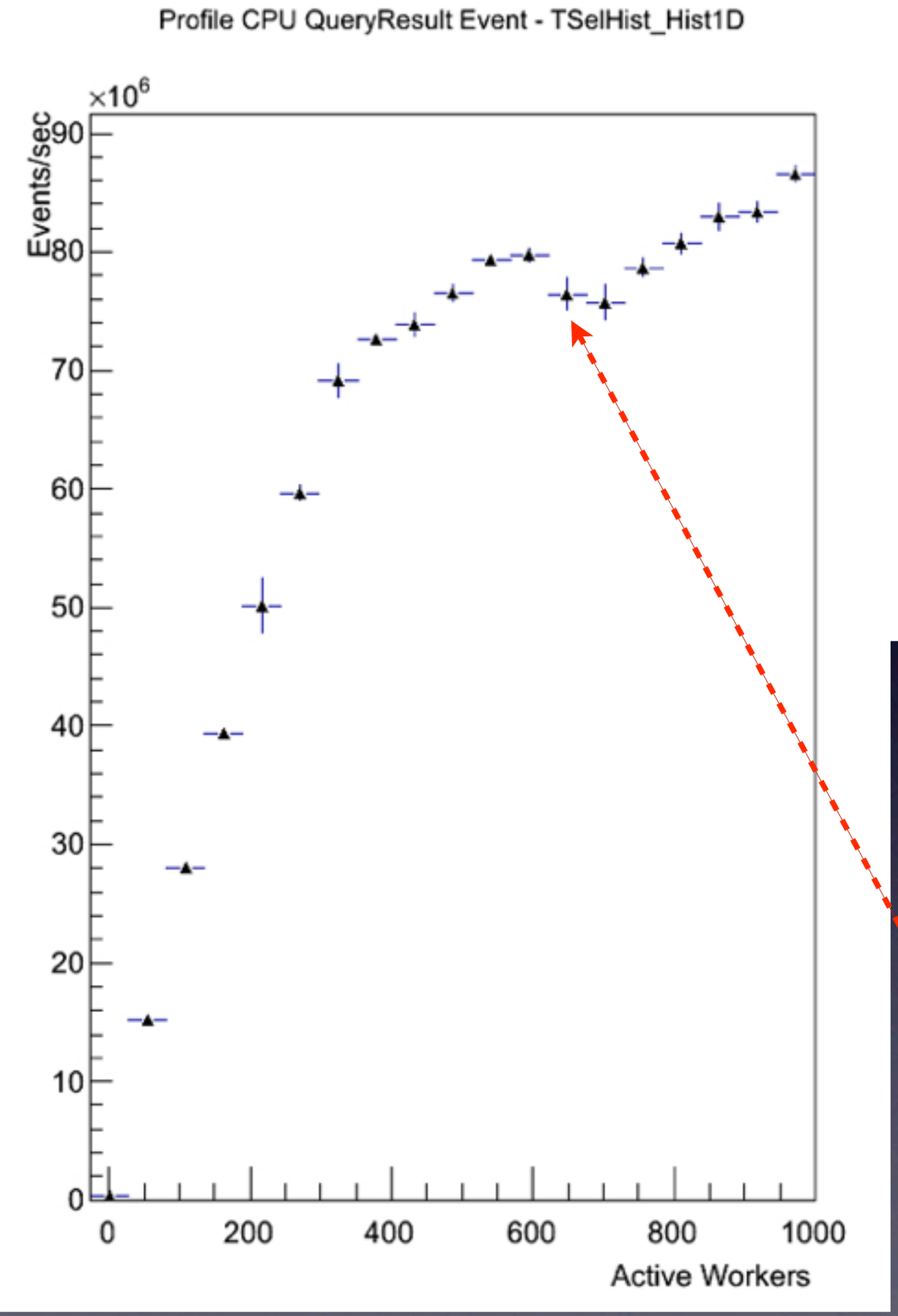


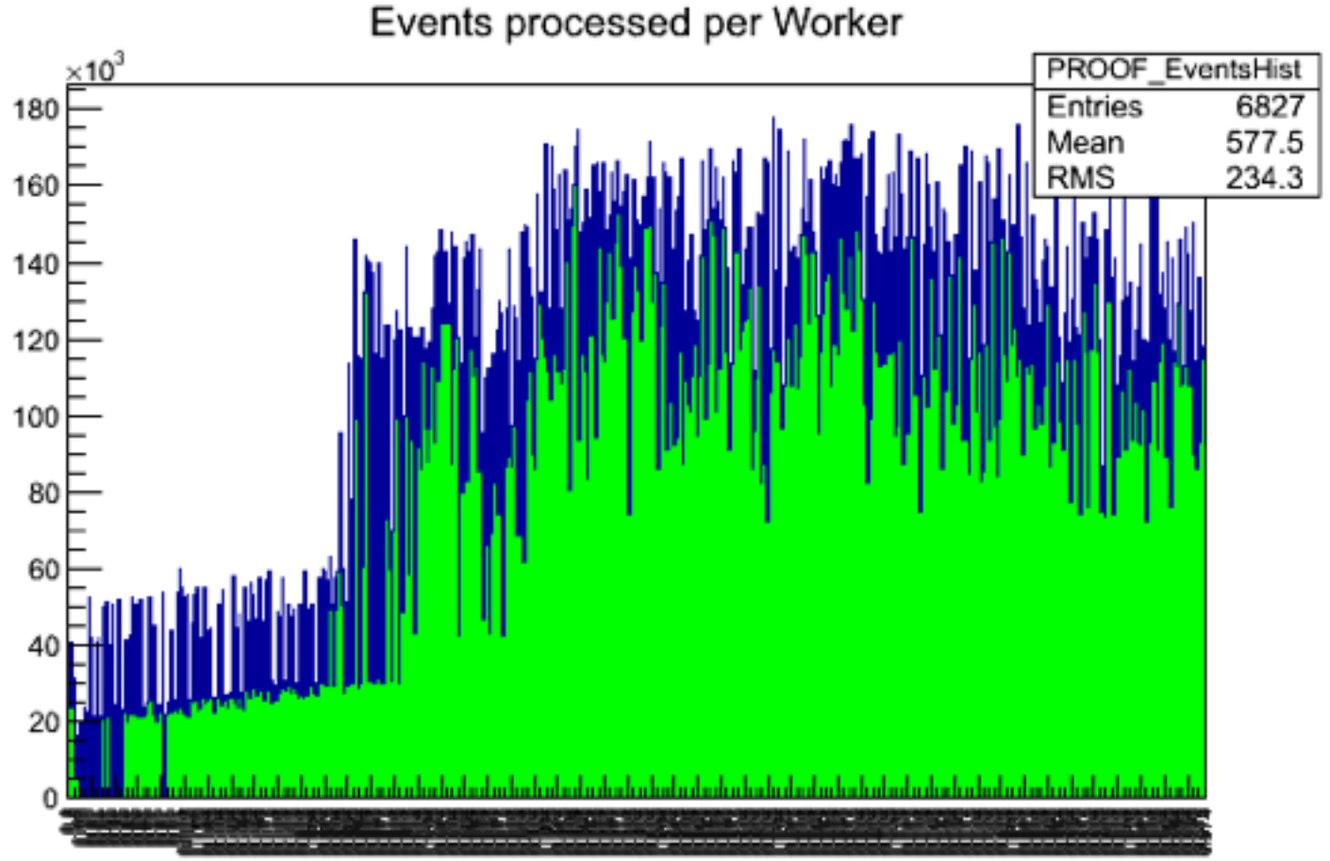
Slow single-threaded packetizer: some workers are idling





~350WNs per Master is the efficiency border





What is this? Is it a "Slow Packetizer" effect?
Perfectly reproducible, when going 500+ WNs...

Fitness: conclusions

PoD:

- is a scalable and a perfect tool for PROOF on the Cloud,
- more optimization is required to distribute WNs faster via SSH (when going over 600+ WNs).

PROOF:

- ~350WNs per Master is the efficiency border,
- a single-threaded PROOF packetizer is a bottleneck,
  - a new TThreadPool is in development. Can that help?
- should not be limited by only ~975 WNs per Master
  - can be solved by changing select() call to poll().

Analysis Software:

dynamic, on-demand Analysis Model

Essential Properties

- portability,
- simplicity,
- confidence (trust),
- lightweightness.

Portability

How can we achieve portability of our analysis code?

Develop portable analysis frameworks:

- Don’t be irresponsible! Just use ROOT!
- Constantly revise all dependencies.

Use open standards.

even more - be a part of standardization committees...
For example, HEP is a big C++ customer.
Are we presented well in the C++ standards committee?

Simplicity

Keep it simple!

- supportable code,
- simple interfaces,
- task based analysis.

Confidence

Trust your code!

- unit/functional tests.

  We should write testable code.

Code must fit for tests and not the other way around.

```
TDD
all code is guilty
until proven innocent
```

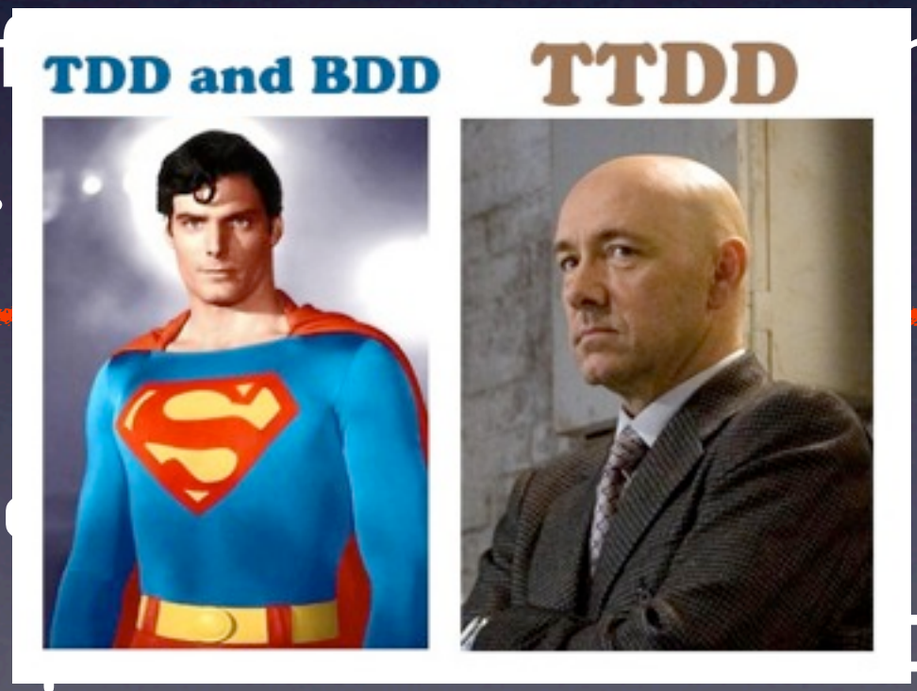
Confidence

Trust your code!

- unit/functional tests.

  We should write testable code.

Code must f he other way around.




```
all
until
```

ty

tent

http://fabiopereira.me/blog/2010/05/27/ttdd-tautological-test-driven-development-anti-pattern/

Lightweightness

Minimal dependency of business logic code to infrastructural code.

(aka separation of business logic and infrastructural stuff)

Summary

Hard-/software infrastructure operations are getting even more complicated (VMs, GPU, Cloud...).

If you want to harness all possible resources you can't prescribe OS, MW, and RMS.

At FAIR, for example, we try to be as independent as possible by using ROOT and PROOF as basis.

If you sit on ROOT and PROOF: It's important that experiments support PROOF.