Evaluating Google Compute Engine with PROOF

G Ganis CERN
S Panitkin BNL

CHEP 2013, Amsterdam, 14 October 2013

Outline

- ATLAS and clouds
- Google Compute Engine
- PROOF benchmarks
- Summary

ATLAS and clouds

- R&D project to explore clouds to cope with spikes in demand for computational resources
  - See R. Sobie et al., ATLAS Cloud Comp. R&D, Facilities, Infrastructures, Networking track
- Experience with variety of cloud platforms
  - EC2, hybrid commercial / academic
- Trial project on Google Compute Engine (GCE) from August 2012-April 2013
  - ~5M core-hours allocated

ATLAS and GCE

- Tried
  - Cloud Storage and data management
  - Xroot for Cloud storage aggregation and interaction with ATLAS Xroot federation
  - PanDA queue for Monte Carlo Simulations
  - Analysis Facility mode
    - PROOF
- Positive experience as a whole
  - Large scale production run on GCE for about 2 months (500 CPU, ~4k cores, 214 M events)

Google Compute Engine

- Google's IaaS product
- Public since beginning 2013
- KVM, linux images
  - Debian 6 & 7, CentOS 6.2
  - No private images
- Five geographical zones
  - europe-west1-a, europe-west1-b
  - us-central1-a,us-central1-b, us-central2-a
- Competitive pricing scheme

GCE: what you get

- 1,2,4,8 core machines
- Standard, HighCPU, HighMem profiles
- Disks (NAS)
  - Ephemeral (tied to the life of the machine)
  - Persistent
- Possibility of snapshots
- Interfaces
  - WEB Portal, command line tool (gcutil)
  - RESTful API
  - Metadata server for contextualization

GCE: for this run

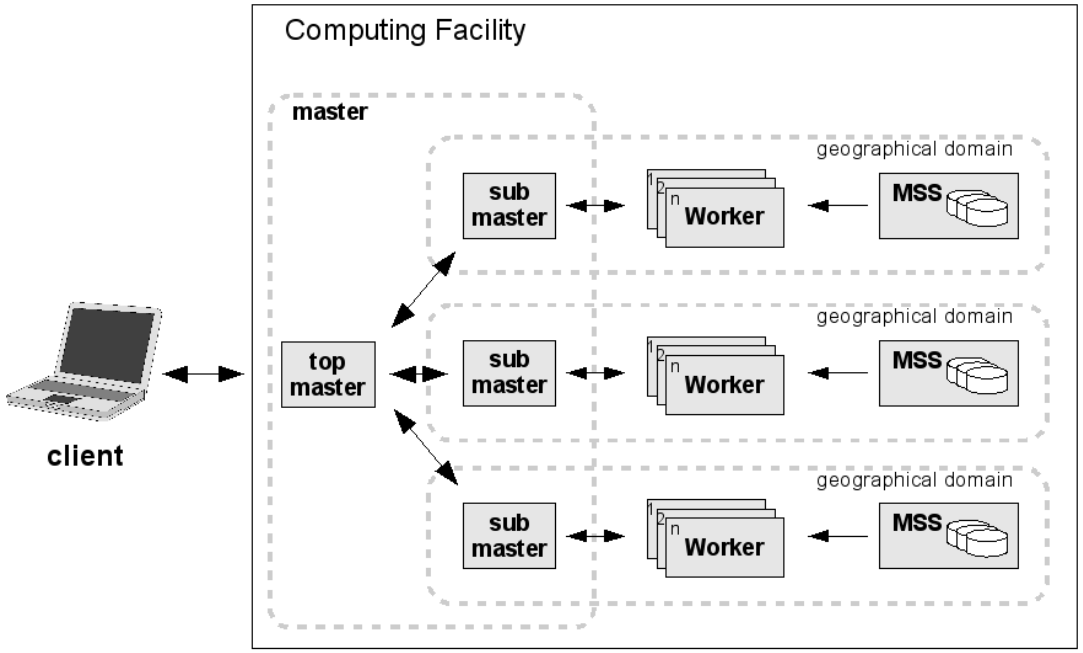
- Single machine contextualized ‘by hand’
  - Deployed via snapshot
- gcutil to manage the machines and disks
  - Gives full control
- Startup latency very good
  - Typically less than 1 minute almost independent on the number of machines

What could help here:

- Use of contextualization tools
- Possibility to use own images

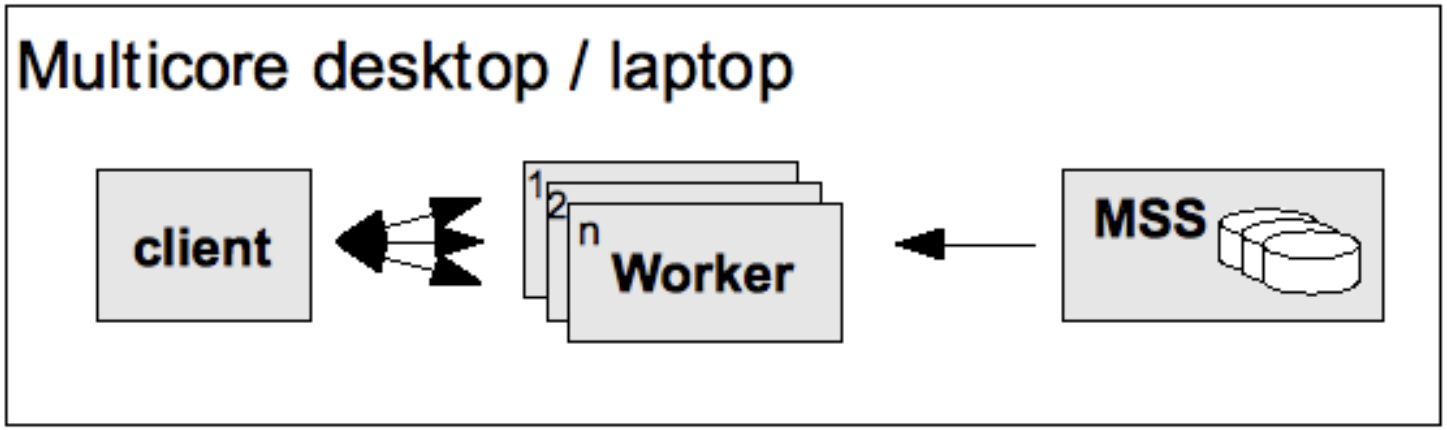
PROOF - Parallel ROOT Facility

- Multi-process parallelism
- Pull architecture
- Output Merging



PROOF Lite

- Multi-process parallelism on multicore



TProofBench

- CPU intensive
  - *Cycle*: generation of random numbers, fill histos
  - *Cycle/s* versus *# of workers*
- I/O intensive
  - *Cycle*: read entry from a TTree + some filtering
  - *Mbytes/s* versus *# of workers / node* or *# of workers*
  - **Cold read**: reset RAM cache for all files before each measurement
- {Average, RMS} of 4 measurements / point
- Record max rate and average rate
  - Average includes init and term (PROOF overhead)

Single 8-core machine

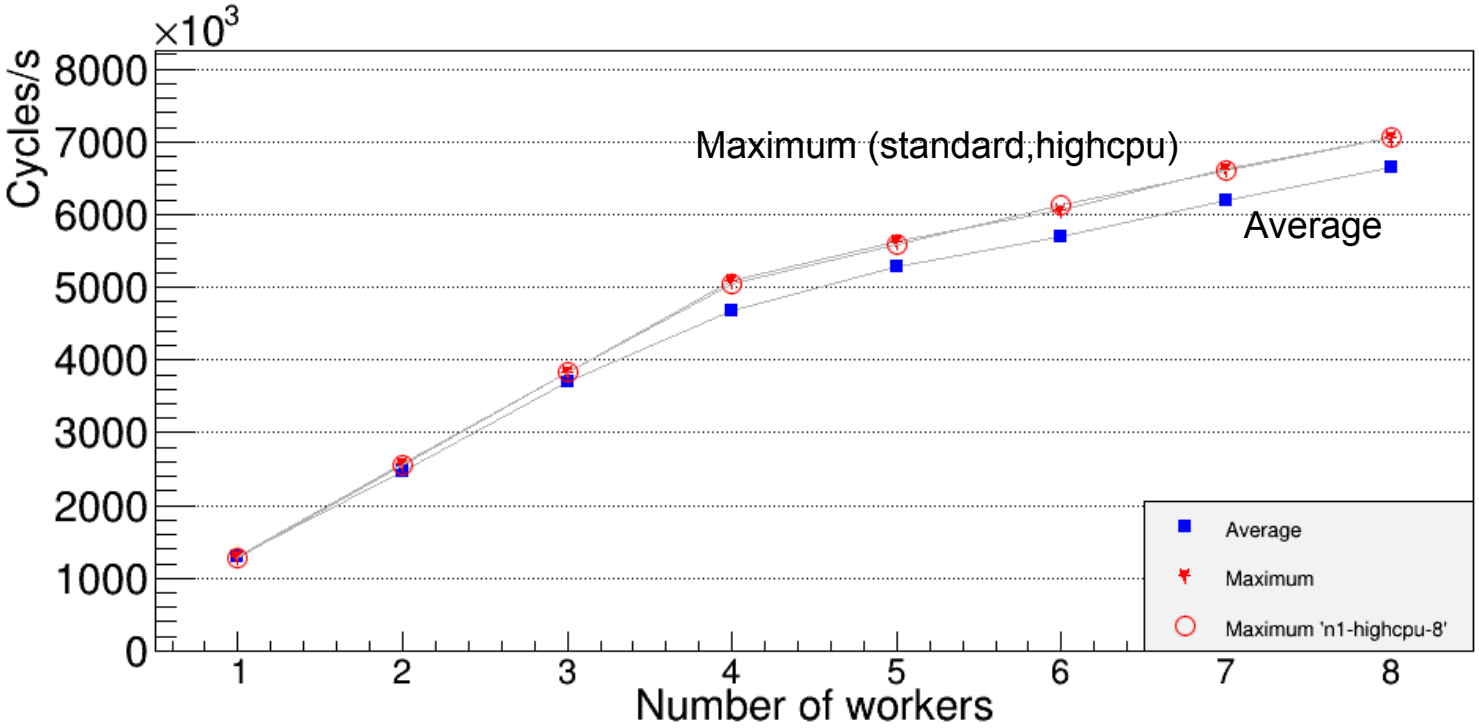
n1-standard-8-d

- 8 vCPU
- 30 GB RAM
- 1.7 TB ephemeral disk

PROOF-Lite

- ROOT v5-34-10

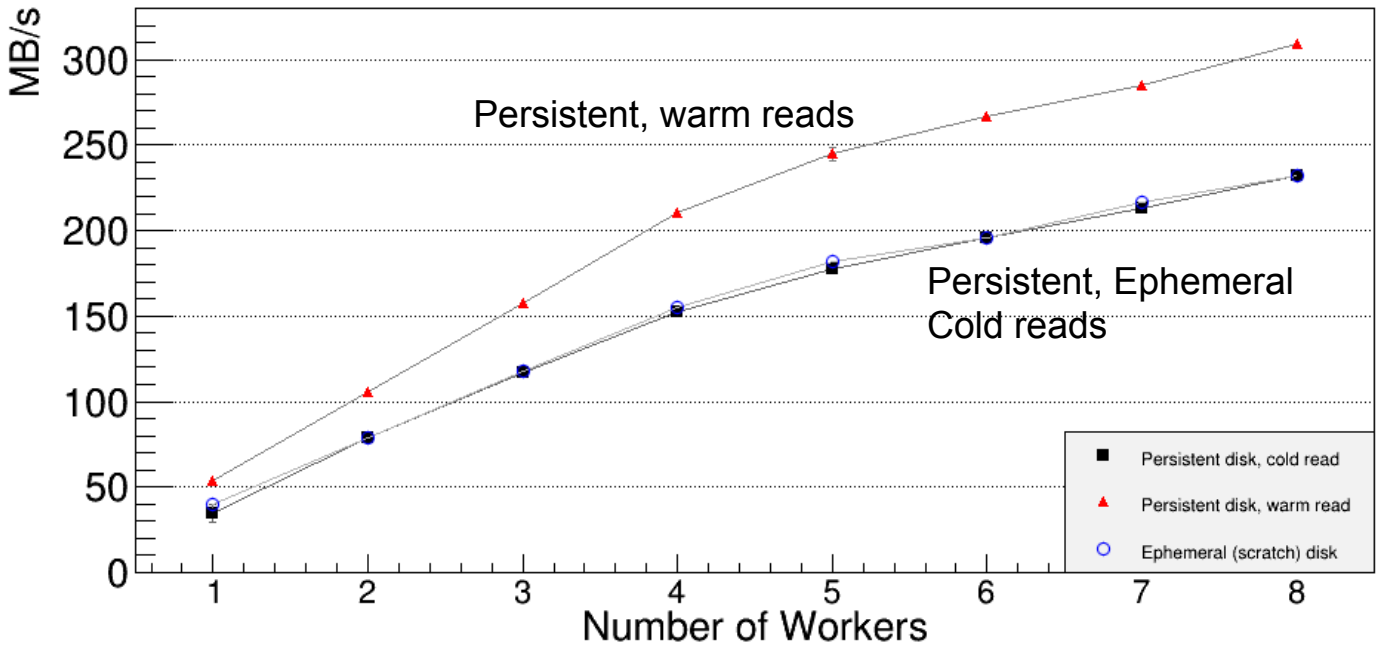
Single 8-core: CPU



Single 8-core: CPU (2)

- Hyperthreaded machine behaviour
- Very good performance / worker compared to real machines
  - 1.27 MCycles/s/wrk Xeon(R)
  - 0.93 MCycles/s/wrk Xeon(R) X7460
  - 1.00 MCycles/s/wrk i7-3632QM
- No difference ‘standard’ and ‘highcpu’ (claimed +10%)

Single 8-core: I/O



dataset: 16 files, 2.8 GB

- Up to 230 MB/s
- Persistent ~= ephemeral

60 node cluster

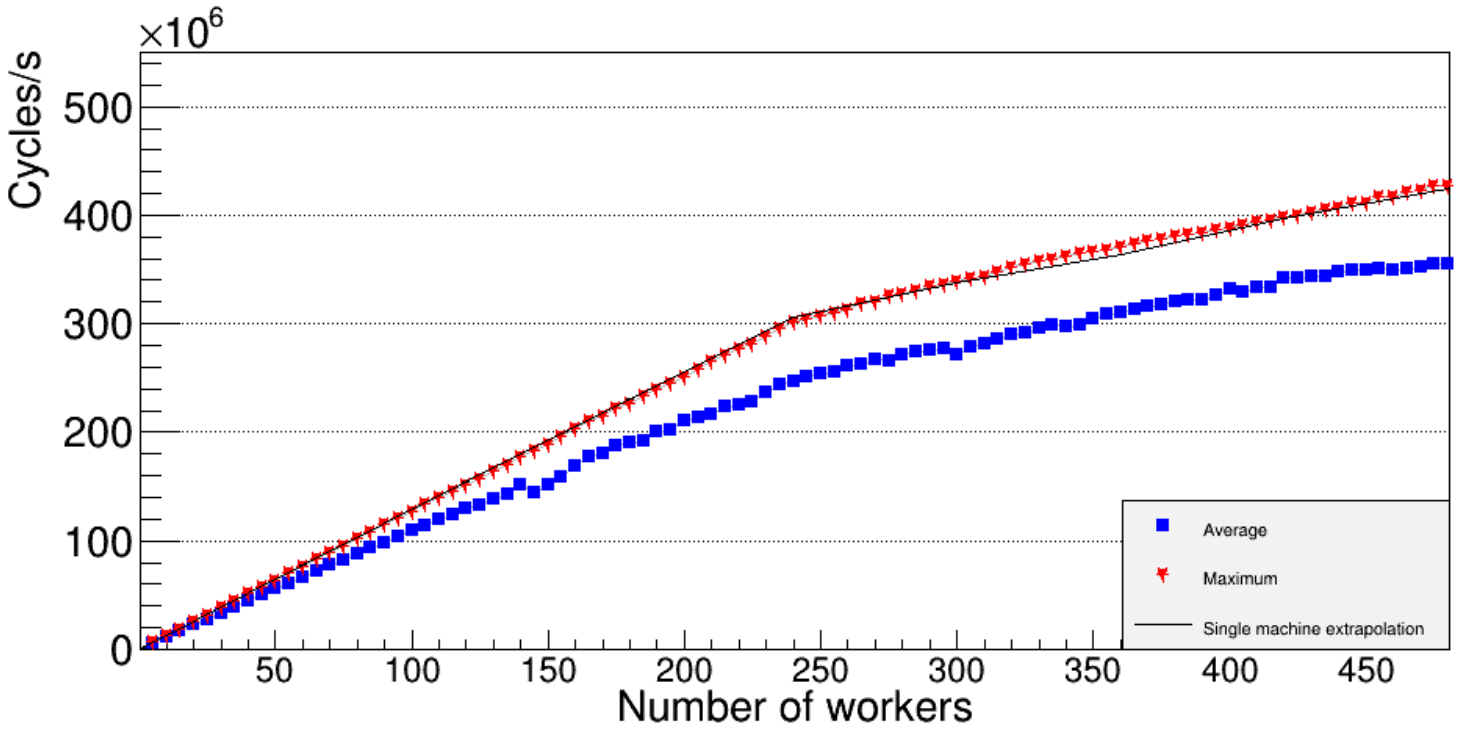
1 + 60 n1-standard-8-d

- 8 + 480 cores
- 30 + 1800 GB RAM
- 1.7 + 102 TB ephemeral disk

Standard PROOF cluster

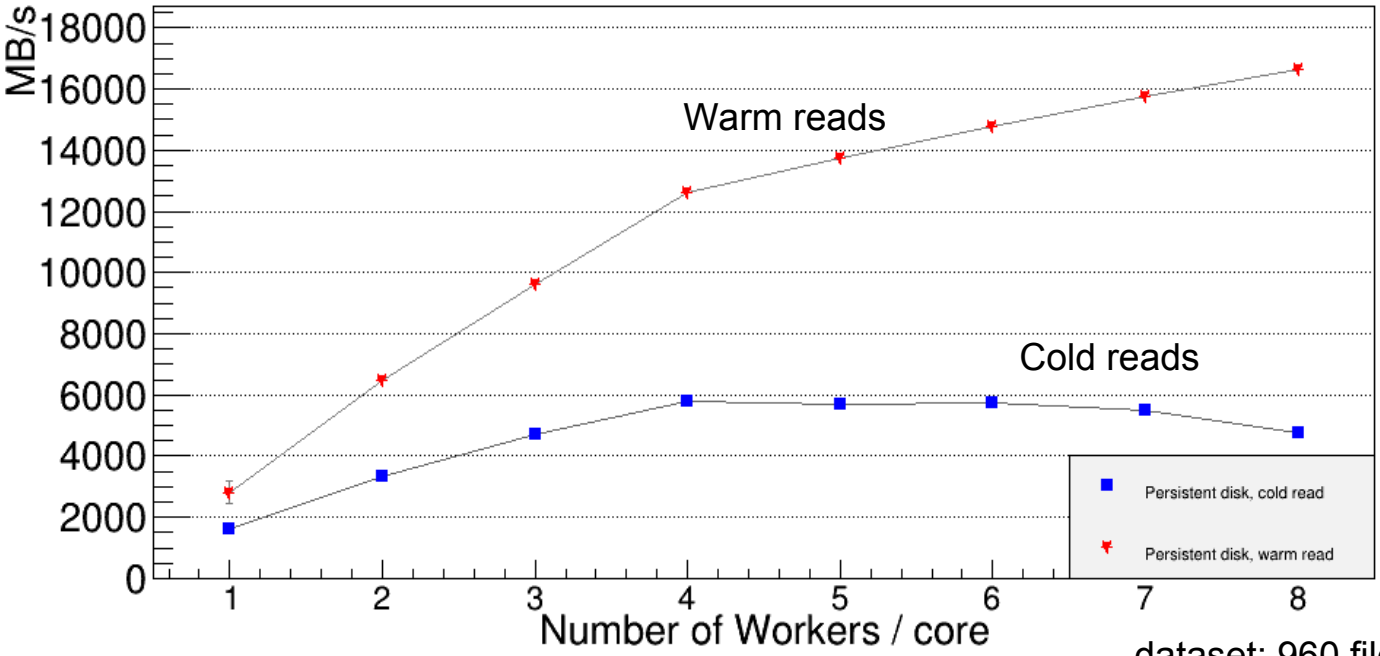
- ROOT v5-34-10

60 node / 480 core: CPU



- Very good scaling of maximum
  - PROOF not limited by work distribution

60 node / 480 core: I/O



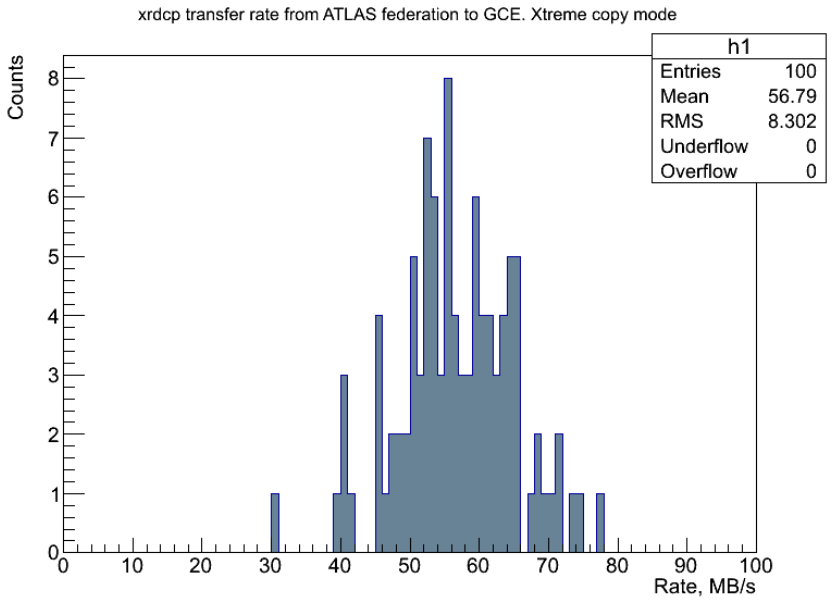
dataset: 960 files, 168 GB

- 6 GB/s @ 4 workers/node (100 MB/s node)
- Warm rate scales from single node

60 node / 480 core: I/O (2)

- Saturation effects due to network topology ?
- PROOF clusters with 4 workers / node seems to give optimal results

Network I/O



- Federated ATLAS Xroot to GCE
- Multi-{source,stream}
- Xroot cluster on GCE ephemeral storage
  - 1.7 TB / node
- Average transfer rate: 57 MB/s
  - Single source xrdcp rate 40 MB/s
- Note, this is over public networks
  - Direct network peering with academic and LHC networks under discussion (100 Gb/s)

Summary

- Positive experience with PROOF @ GCE
- Hardware very stable
  - Restarts only required for changing conf
- Good absolute and scaling performances
  - CPU perf compares well to real CPU
  - 100 MB/s / node
- Viable solution to cope with spikes in demand for computational resources for analysis

Thank you!

Questions?