CHEP 2013

Business as Unusual

Summary Overview

Posters and CHEPers in The Grot Zaal

Summary Overview

- Introduction
- Buzzword Glossary
- General Themes
- Interesting Topics

Introduction

- CHEP – Computing In High Energy and Nuclear Physics
- Held every 18months, international
- 2013 in Amsterdam - ~500 participants
- Plenaries and 6 parallel tracks (Data Acquisition, Event Processing, Distributed Processing, Data Storage, Software Engineering, Production Infrastructures)

Introduction

- Also ~230 posters
- 20th CHEP – 28 years since last held in Amsterdam
- Interesting to see what has changed in that time…

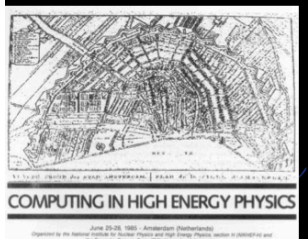
Things have changes since 1985 …

… have completely gone away …

- "Portability Aspects of MODULA-2"
- "Using the 3081/E as a VAX Emulator"
- "A LAN with Real-Time Facilities *based on OSI Standards*"

… or have just changed a lot …

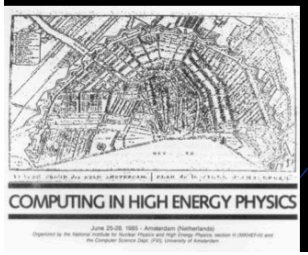
- "Satellite Communication"
- "LAN with an Experiment Command Interpreter and 2.5 MBaud Interfaces"



http://www.chep2013.org/1985

… but not all that much!

- Multi-processor, multi-core & 'GPU'
  - "Loosely and Tightly Coupled Parallel Processors for High Energy Physics"
  - "Parallelism in Scientific Engineering Computation"
  - "Use of SIMD—SPMD Machines for Simulation in Particle Physics"
  - *Panel discussion*: "Vector and Parallel Processing in HEP"



http://www.chep2013.org/1985

Buzzword Glossary

- “Cloud” (as compared to “Grid”)
- Grid Computing
  - Dedicated computer systems
  - Physical systems at validated sites
- Cloud Computing
  - Infrastructure As A Service (IAAS) (Cue Car Analogy)
  - Virtualised nodes created/destroyed on demand
  - Can use any provider with resources

Buzzword Glossary

- Grid Storage
  - Dedicated storage systems co-located at Grid Computing sites (eg DPM at Liverpool)
  - Owned by the Grid sites
  - Accessible anywhere with authorisation
- Cloud Storage
  - Virtualised storage systems
  - Leased from third parties
  - No capital expense, scales as required
  - Eg Dropbox, Google Cloud Storage

Buzzword Glossary

- GPGPU
  - General Purpose Graphics Processing Units
  - Hundreds of simple Cores
  - nVidia/AMD, CUDA
- Multi Core
  - A chip with more than one Core/CPU on it
- Many (Integrated) Core
  - Intel Xeon Phi
  - Dozens/Hundreds of Cores

Buzzword Glossary

- Big Data
  - Traditionally defined as
    - Volume
      - Amount of data
    - Velocity
      - Speed of processing
    - Variety
      - Different types of data
- Not just about sheer size…

General Themes

- GPGPUs and other animals
  - Getting more for less the hard way
- C++11+
  - C++ grows up, it'll be 14 before you know it
- Clouds Are Gathering
  - Even if they're our Own Clouds
- The Federation
  - It's cheaper to move Data than store it

getting more for less the hard way

GPGPUS AND OTHER ANIMALS

Processing

- GPGPUs
  - Nvidia (CUDA framework)
  - AMD (OpenCL)
- MIC
  - Intel Xeon Phi (x86, icc)
- CPUs
  - x86 (Intel, AMD)
  - ARM

Processing

New architectures

- Over the past ten years processors have hit power limitations which place significant constraints on "Moore's Law" scaling.
- The first casualty was scaling for single sequential applications, giving birth to multi-core processors.

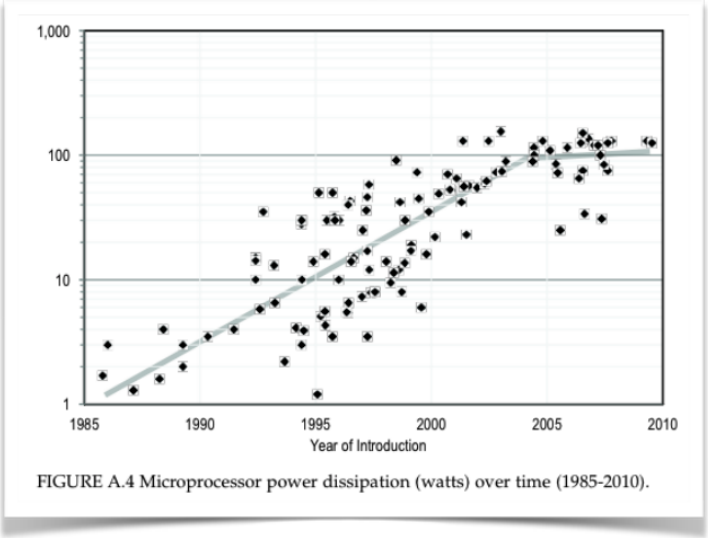


FIGURE A.4 Microprocessor power dissipation (watts) over time (1985-2010).

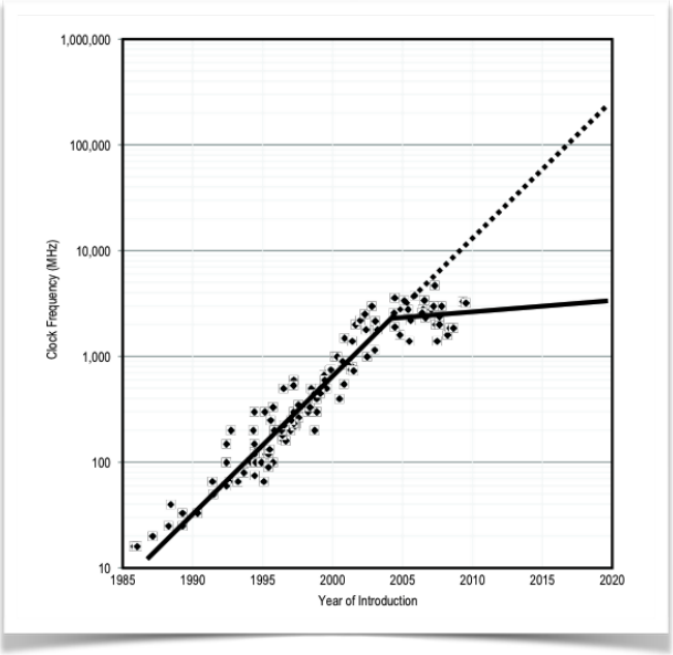
From: "The Future of Computing Performance: Game Over or Next Level?"

"Explorations of the viability of ARM and Intel Xeon Phi for Physics Processing"
- presented by Dr. Peter Elmer

Processing

New Architectures

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.



From: "The Future of Computing Performance: Game Over or Next Level?"

"Explorations of the viability of ARM and Intel Xeon Phi for Physics Processing"
- presented by Dr. Peter Elmer

Processing – Tesla & Xeon Phi

Why are they interesting ?

	Xeon E5-2687	Tesla K20X	Xeon-Phi 7120P
#physical-cores	8	14 SMX	61
#logical-cores	16	2688	244
clock (GHz)	3.1	0.735	1.238
GFLOPS (DP/SP)	198.4/396.8	1.317/3.950	1.208/2.416
SIMD	AVX 64-bit	N/A	AVX2 512-bit
cache (MB)	20	1.5	30.5
#Mem. Channels	4	–	16
Max Memory (GB)	256	6	16
Mem BW (GB/s)	51.2	250	352
ECC	YES	YES	YES

- 1 Tflops in one device ✔
- nothing is for free ✘
  - manage high number of threads
  - exploit several levels of parallelism
  - hide latency host-device (Amdhal law)

Processing – Tesla & Xeon Phi

- Nvidia CUDA dominant in HPC
  - More mature platform
  - Better performance
  - Development work continuing
- Xeon Phi (MIC) emerging
  - Very early days for HEP
  - Shows promise
  - Issues with Intel compiler (icc)

Processing – ARM

Power efficiency

Compare power consumption to HS06 values

	HS06	power [W]	HS06/W
Calxeda/Viridis	10.4	~5	2.1
HP dc7900 i7	95	~150	0.63
HS22 E5620	130	~250	0.52
HS22 E5645	179	~250	0.72
HS23 E5-2670	339	~360	0.94
DELL C6145	558	~600	0.93

Power consumption values are estimates

Observations

Power efficiency advantage for ARM by factor 2-4

480 HS06 in 2U enclosure possible (1.1 GHz SoC)

Calxeda EnergyCore SoC with 1.4 GHz (~13 HS06?)

1 GB/core, would need multithreading in applications

ARM A15 (PAE) and A53/A57 (64bit) in 2014/15

Cost of ARM servers not yet competetive (€/HS06)

Ubuntu (and now Fedora) Linux OS available

No port of HEP or experiment software attempted, but "should be straightforward"

Running SPEC 2006 somewhat cumbersome

Dedicated optimization for FPU or GPU?

Conclusions

Large power and possibly cost savings potential with ARM based servers

Linux (Ubuntu, Fedora) established on ARM

Should invest in HEP and experiment software ports

Ability to use different CPU architectures puts pressure on vendors

"HS06 benchmark values for an ARM based server"
Presented by Stefan Kluth

Processing – ARM

- Slower cores, but much more power-efficient
- More events per W
- Not quite competitive on events per £/$/€ yet
- Typically 32-bit, not traditionally aimed at HPC and data centers
- HPC-targeted CPUs and System-on-Chip (SoC) coming
  - Project Denver from Nvidia
  - HieroFalcon from AMD

c++ grows up

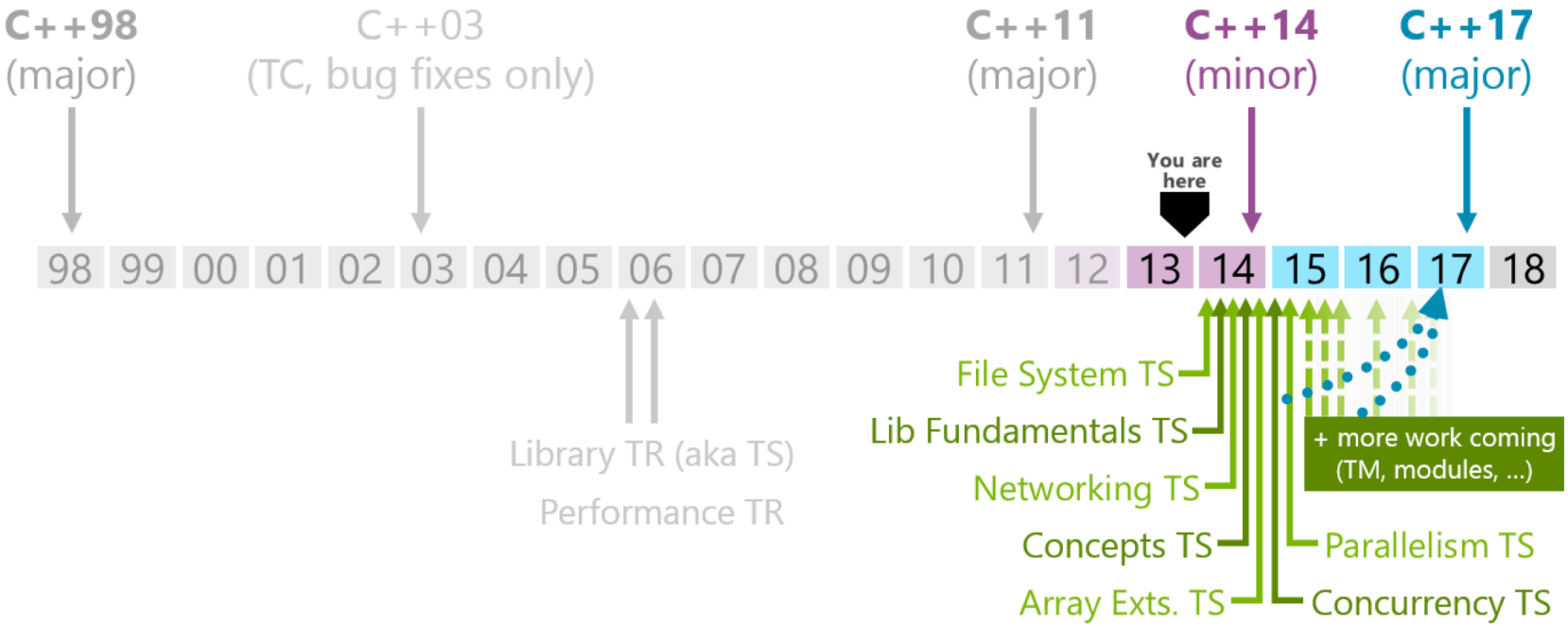
C++11+

C++11

- Significant update to the language
- New compilers
- No mixing of C++98 and C++11
- Auto vectorisation
- Simplified
- ROOT moving to C++11 in v6

- **C++ evolves!** Presented by **Axel NAUMANN**

C++ Standardisation



Relevance of C++11 to HEP

- Bjarne Stroustrup: “C++11 feels like a new language”
- Simpler code
- More expressive code
- Ability to write robust code
- Increased performance

C++11: Simple Code

```
for (std::map<std::string, std::vector<MyClass> >::const_iterator
   i = m.begin(), e = m.end(); i != e; ++i) {
```

- auto

```
for (auto i = begin(m), e = end(m); i != e; ++i) {
```

- Range-based for

```
for (auto i: m) {
```

Auto-Vectorization

- Compilers combine simple loop iterations into vector operations
- Function call, pointer access etc prevent auto-vectorization
- Advantage: needs no extra code; leverages compiler knowledge and optimization
- Disadvantage: rarely possible; needs intrusive code refactoring; gets easily broken also because the vectorization is not explicitly visible (except for “ugliness” of code)

From C++03 to C++11

- 100% supported by GCC 4.8, clang 3.3 with flag -std=c++11; largely by GCC 4.7, clang 3.2, ICC 14, MSVC 2013
- Old C++ code usually compiles in C++11 “mode”, ROOT had about 8 changes on 3 million lines of code:
  - token#pasting CPP macros
  - x={...} initializers
- Object file compiled with C++11 should not be linked against old C++: all C++11 or none

- Modern compilers solves frequent user complaint: diagnostics!

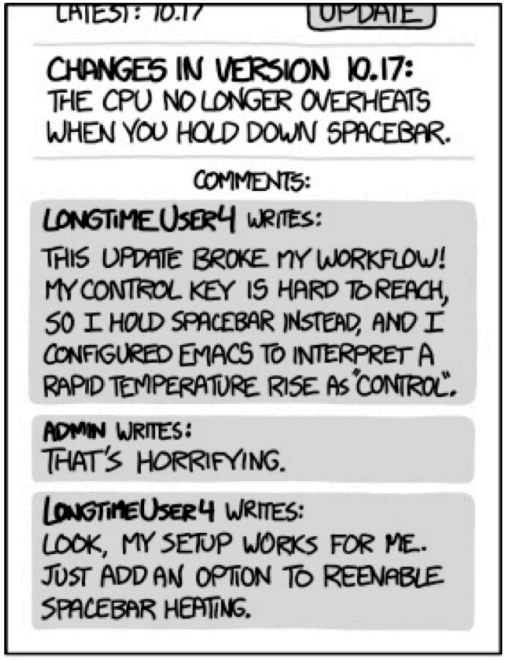
```
std::find(vec.begin(), ConstVec.end(), 12);
```

```
T.C: In function 'void f()':
T.C:9: error: no matching function for call to 'find(__gnu_cxx::__normal_i
terator<double*, std::vector<double, std::allocator<double> > >, __gnu_cxx
::__normal_iterator<const double*, std::vector<double, std::allocator<doub
le> > >, int)'
```

```
T.C:9:4: error: no matching function for call to 'find'
   std::find(Vec.begin(), ConstVec.end(), 12);
   ^~~~~~~~~
/usr/include/c++/4.6/bits/stl_algo.h:4394:5: note: candidate template
      ignored: deduced conflicting types for parameter '_InputIterator'
      ('__normal_iterator<double *, [...]>' vs.
      '__normal_iterator<const double *, [...]>')
    find(_InputIterator __first, _InputIterator __last,
    ^
1 error generated.
```

Language Summary

- The language has changed dramatically
- Many benefits especially for casual coders: safe, simple, expressive code
  - ownership clarification
  - concise constructs for common idioms
- Improved standard library
- It saves time!



EVERY CHANGE BREAKS SOMEONE'S WORKFLOW.

from http://xkcd.com

clouds are gathering

CLOUDS

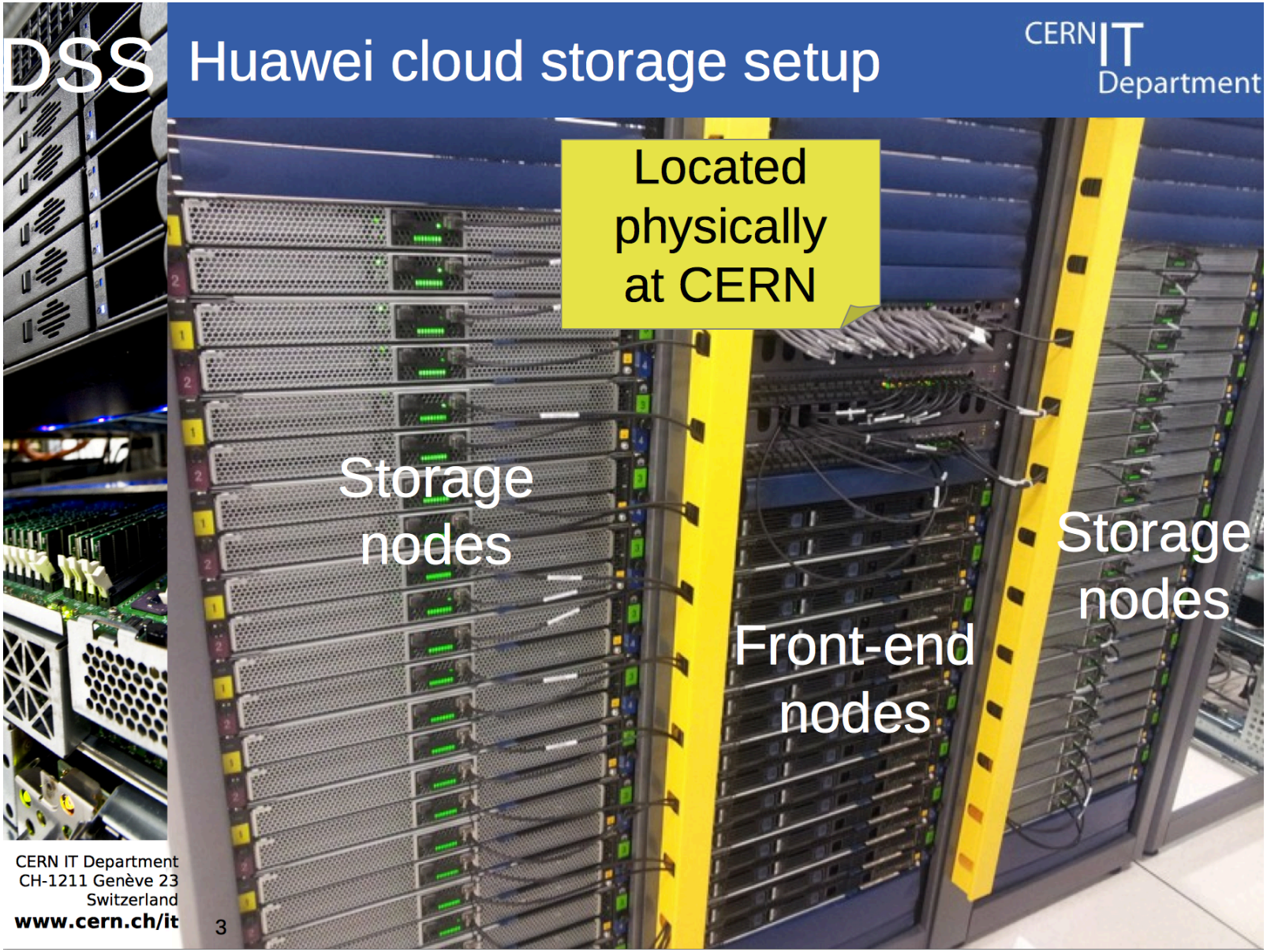
Clouds - Storage

- Standard interfaces
- Can scale well
- Lots of commercial interest and support
- Great for opportunistic or short term needs
- Ubiquitous access

Clouds - Storage

- Cern cloud storage tests
- S3 compatible
- linear scaling with number of frontends
- low maintenance
- ROOT-plugin soon

Huawei cloud storage setup



CERN IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Conclusion

- Raw performance
  - Upload and download scalability demonstrated
  - Additional front-end nodes increased linearly the performance
- Fault tolerance: powering off a chassis
  - Transparent disk failure recovery demonstrated
- File system with cloud storage back-end
  - Full publishing procedure tested
  - Uploading of only new files feature tested

Clouds - Storage

- Cloud storage for BES II experiment
- s3fs fuse interface
- Standard POSIX filesystem (cp/ls/rm etc)
- Small performance loss but very scalable

Cloud storage

- Object storage system

  well documented interface

  on top of standard protocols (HTTP)

  accessible through wide area network

- Advantages

  elasticity, standard protocols, tunable durability by redundancy, scalability, possibility of using lower cost hardware, private or public

- Significant development over the last few years

  Amazon S3: 2 trillion objects, 1.1M requests/sec (as of April 2013)

- Typical use cases

  well suited for "write-once read-many" type of data: images, videos, documents, static web sites, …

Extending ROOT for cloud storage (cont.)

```
[12:58 | lxslc509] ROOT (0)> root
  ****************************************
  *                                      *
  *       W E L C O M E  to  R O O T     *
  *                                      *
  *   Version  5.24/00b   11 October 2009*
  *                                      *
  *  You are welcome to visit our Web site*
  *         http://root.cern.ch          *
  *                                      *
  ****************************************

ROOT 5.24/00b (tags/v5-24-00b@30662, Sep 03 2013, 14:03:59 on linuxx8664gcc)

CINT/ROOT C/C++ Interpreter version
Type ? for help. Commands must be C
Enclose multiple statements between
root [0]
root [0] .L drawCloudHisto.cxx
root [1]
root [1] drawCloudHisto("swift://fsc.ihep.ac.cn:8080/root/gaussHistogram.root")
<TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2]
```

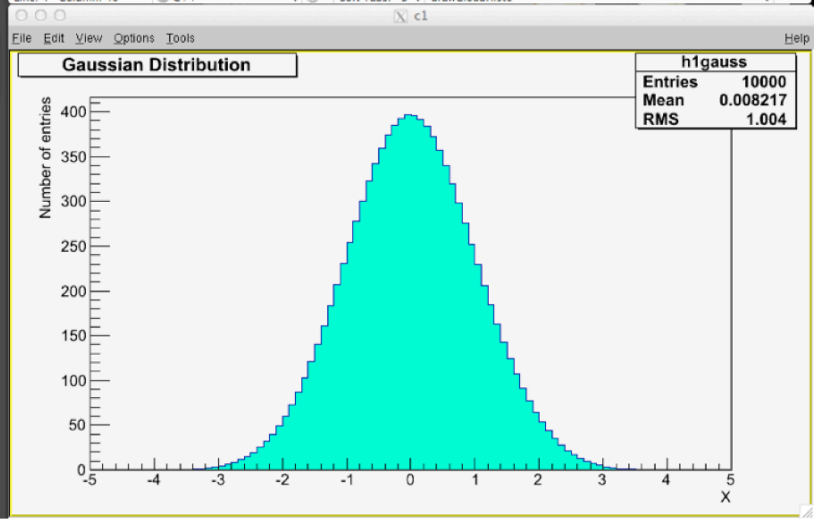
Backwards compatible

Load ROOT C++ macro

Draw the histogram contained specified in the remote Swift file

```
1  void drawCloudHisto(const char* fileName)
2  {
3     // Open the remote file which contains the histogram
4     TFile* inputFile = TFile::Open(fileName);
5
6     // Load the histogram
7     TH1F* histogram = (TH1F*)inputFile->GetObjectChecked("h1gauss", "TH1F");
8
9     // Draw the histogram
10    histogram->Draw();
11 }
12
```

No cloud-specific code



With this extension, BES III can transparently use cloud storage

Filesystem interface to cloud storage

- Useful to expose cloud storage as a local file system

  usual Unix file manipulation commands work transparently (e.g. cp, ls, tar, …)

  POSIX-based applications work (almost) unmodified

- Evaluated S3fs, a FUSE-based file system designed for Amazon S3 backend

  https://code.google.com/p/s3fs

- Features

  files and directories have their corresponding objects named with their full path in S3fs

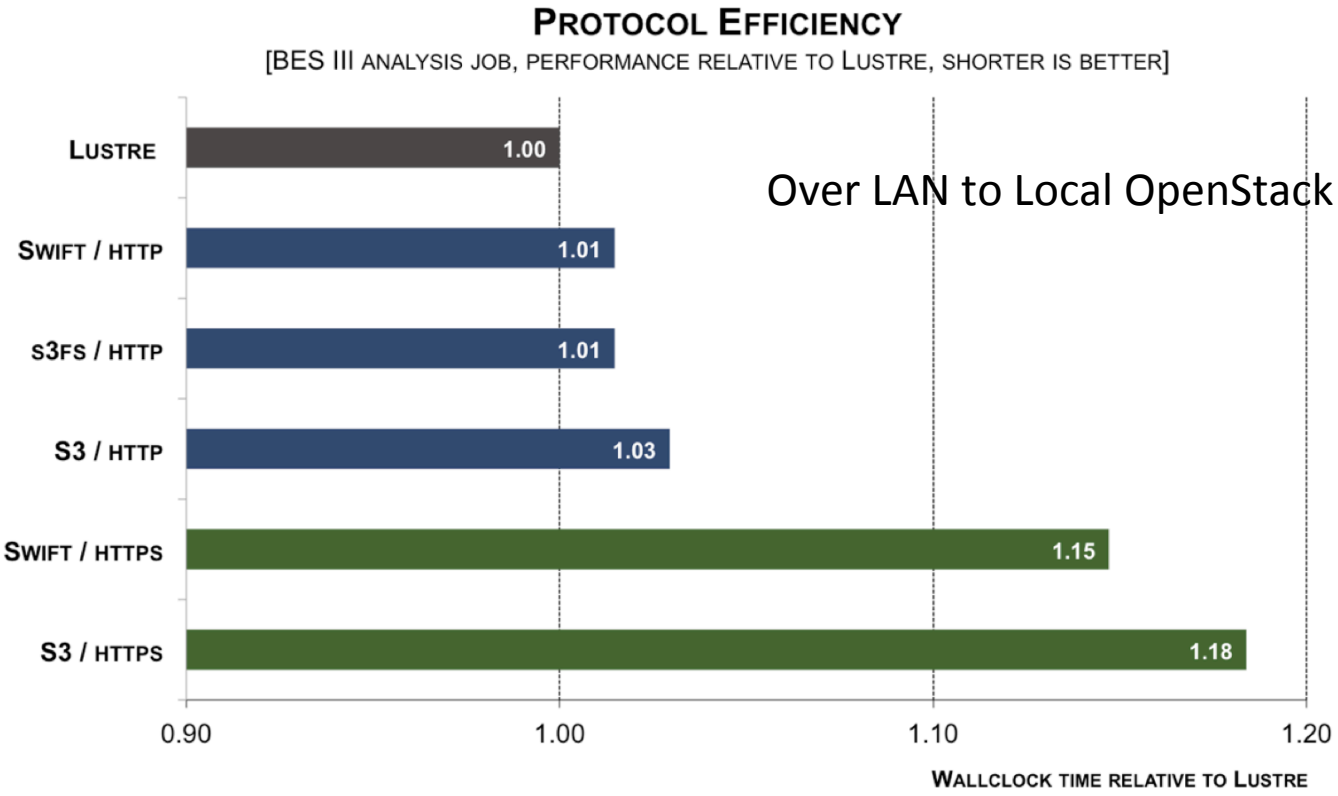
  directories implemented as empty objects to store their metadata

  download whole file to local cache on open(), subsequent operations act on the local copy

  new or modified files are uploaded on close()

See backup slides for details

Efficiency with real jobs



Over LAN to Local OpenStack

Low overhead of both native Swift and S3 over HTTP
Noticeable penalty when using HTTPS

Clouds - Storage

- owncloud: cernbox alterative to dropbox

The origins of the **cernbox** project

- We need a competitive alternative to Dropbox for CERN users
  - Reasons
    - SLAs: availability, confidentiality
    - integration into IT infrastructure
    - archival & backup policies
  - The scale of the problem is unknown but we have some indications
    - 4500 distinct IPs in DNS from cern.ch to *.dropbox.com (daily…)
  - We also want to adapt to user expectations
  - We manage large-scale online-storage systems
    - …and we can leverage on them

EOS (RAW)	CEPH (RAW)	Other services
~62 PB	~3.5 PB	~30 PB

Bulk of disk storage operated by IT/DSS

Clouds - Processing

- Scalability
- Cheaper than capital equipment for short bursts
- Soak up peak CPU demand

- ATLAS and Clouds
- Evaluation of PROOF on Google Compute Engine

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

Summary

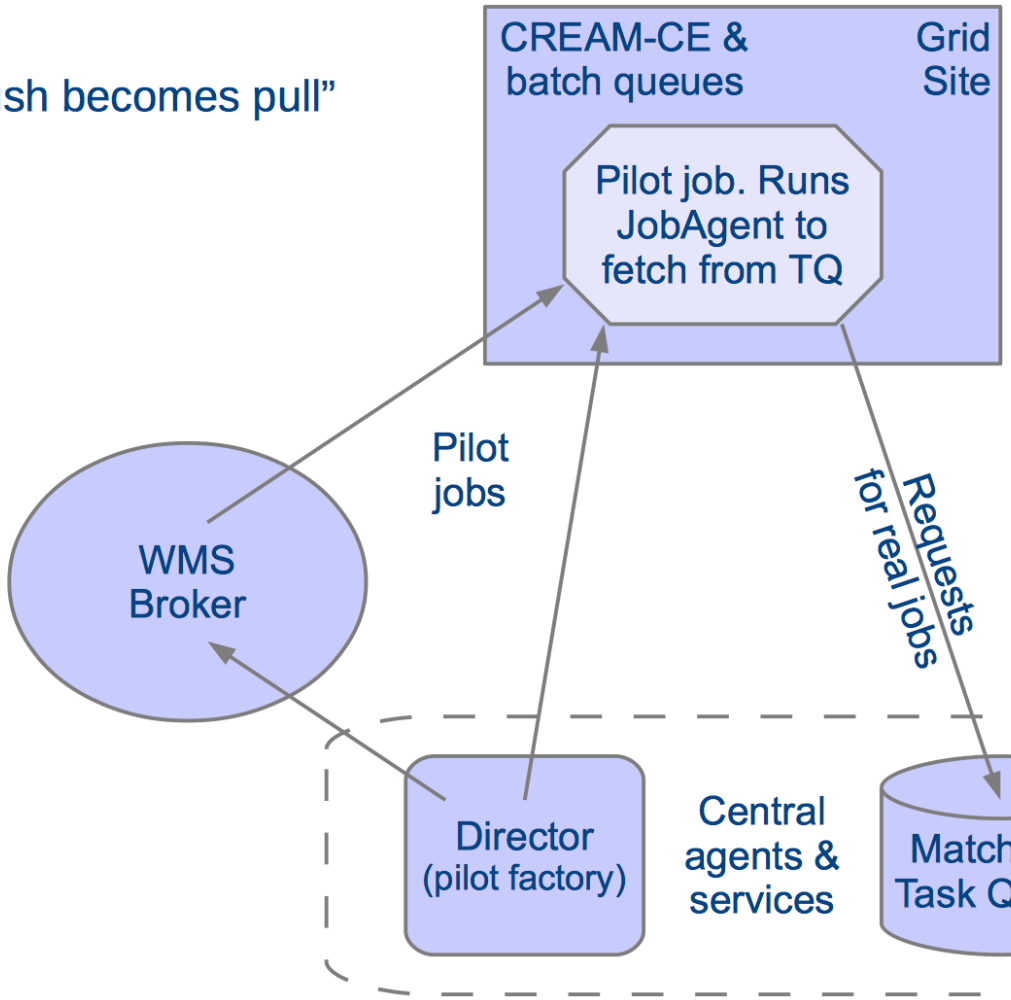
- Positive experience with PROOF @ GCE
- Hardware very stable
  - Restarts only required for changing conf
- Good absolute and scaling performances
  - CPU perf compares well to real CPU
  - 100 MB/s / node
- Viable solution to cope with spikes in demand for computational resources for analysis

Clouds - Processing

- Beyond IaaS to Infrastructure As A Client
- “Vacuum” system for grid jobs
- Pull rather than push job payloads
- No local batch/CE system needed
- Useful for sites with majority work for a few large VOs

The Grid

"Push becomes pull"

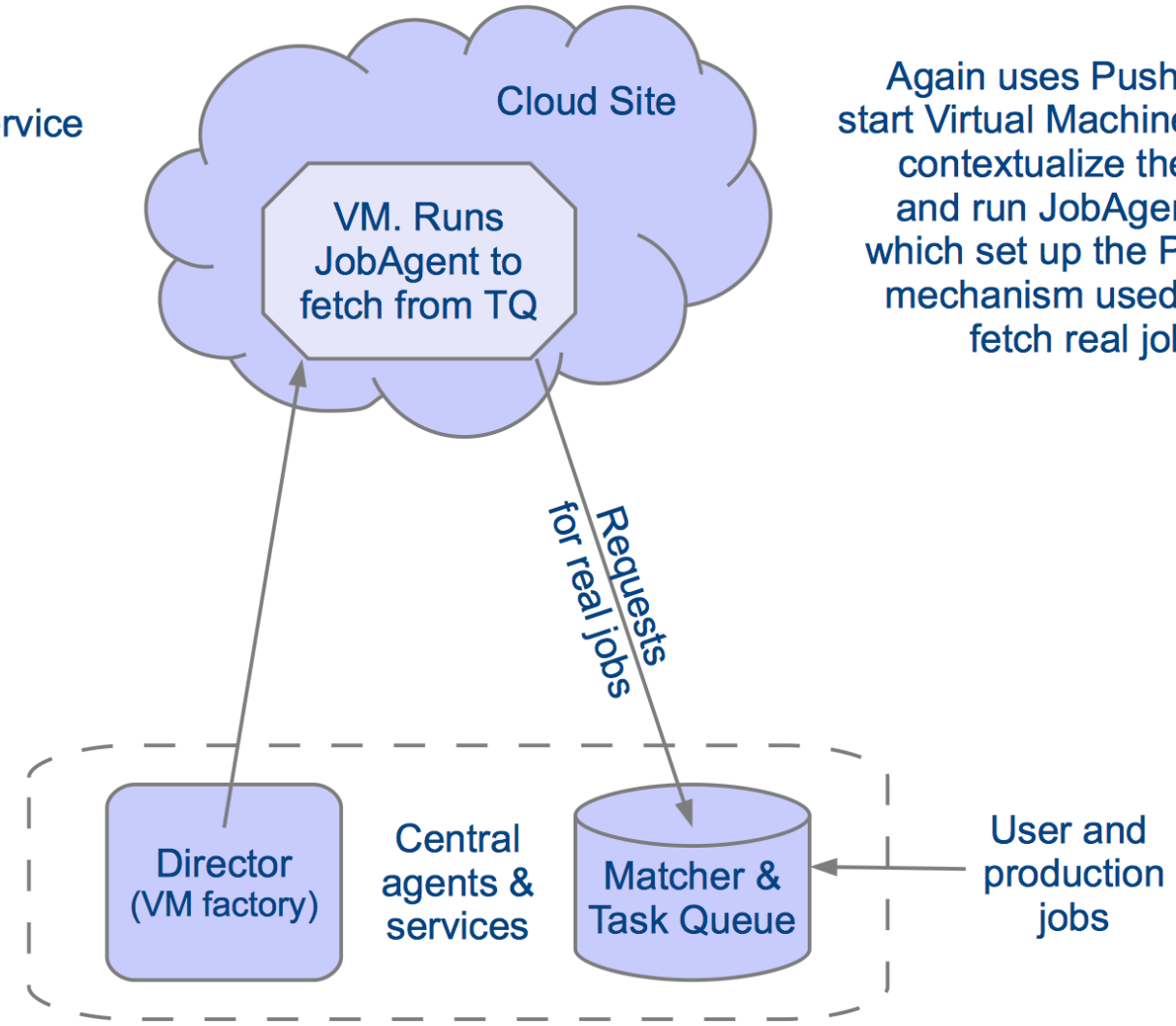


Originally a push model using brokers, but VOs switched to a pull model by using pilot jobs and central task queue.

(These diagrams use LHCb terminology but other experiments have equivalent components.)

The Cloud

Infrastructure-as-a-Service (IaaS)



Again uses Push to start Virtual Machines, contextualize them and run JobAgents which set up the Pull mechanism used to fetch real jobs.

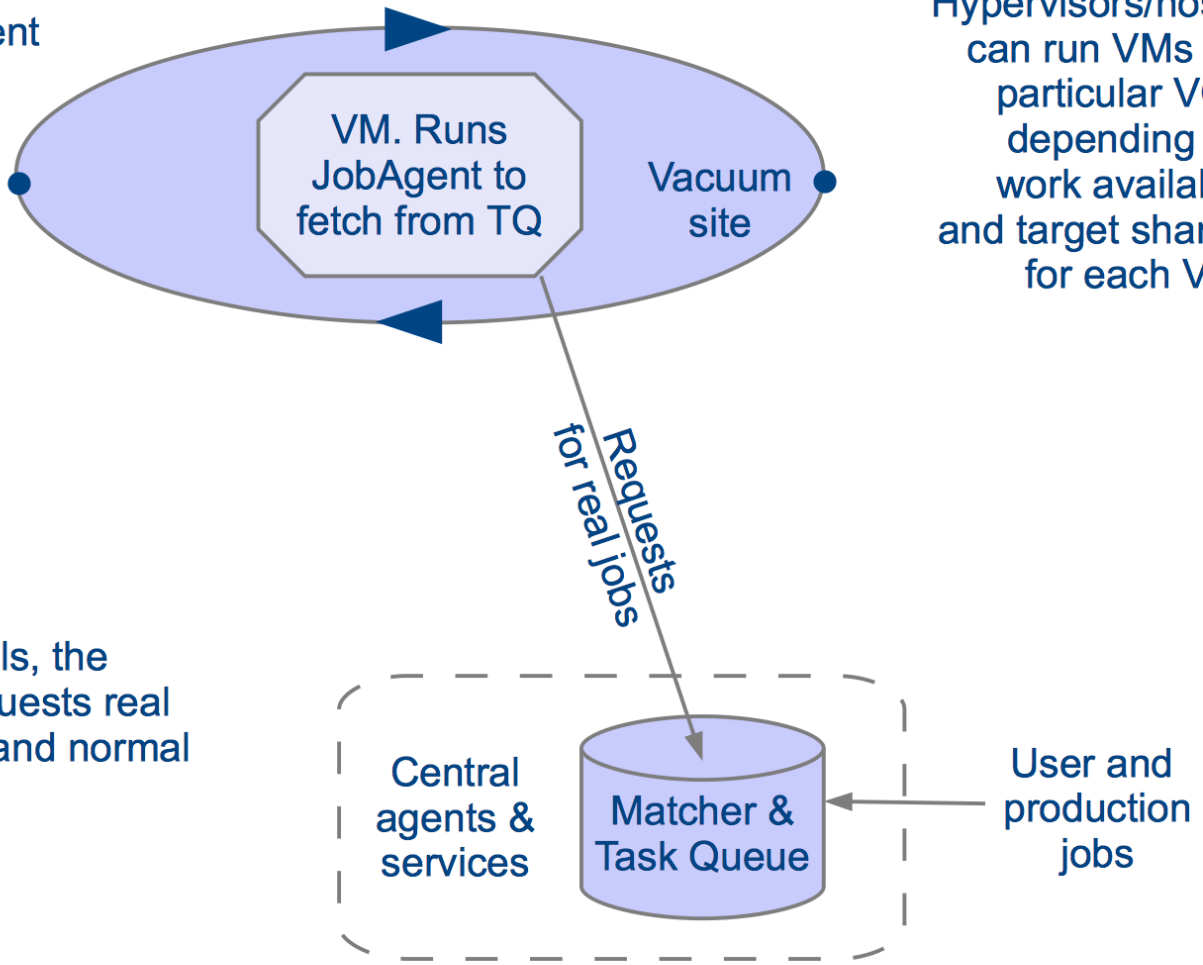
In LHCb, use the same TQ as for Grid and direct DIRAC execution of jobs.

"The Vacuum"

Infrastructure-as-a-Client (IaaC)

Instead of being created by VOs, the Virtual Machines appear spontaneously "out of the vacuum" at sites.

As with the other models, the JobAgent runs and requests real jobs from the Matcher and normal Task Queue.



Hypervisors/hosts can run VMs for particular VOs depending on work available and target shares for each VO.

it’s cheaper to move data than it is to store it

THE FEDERATION

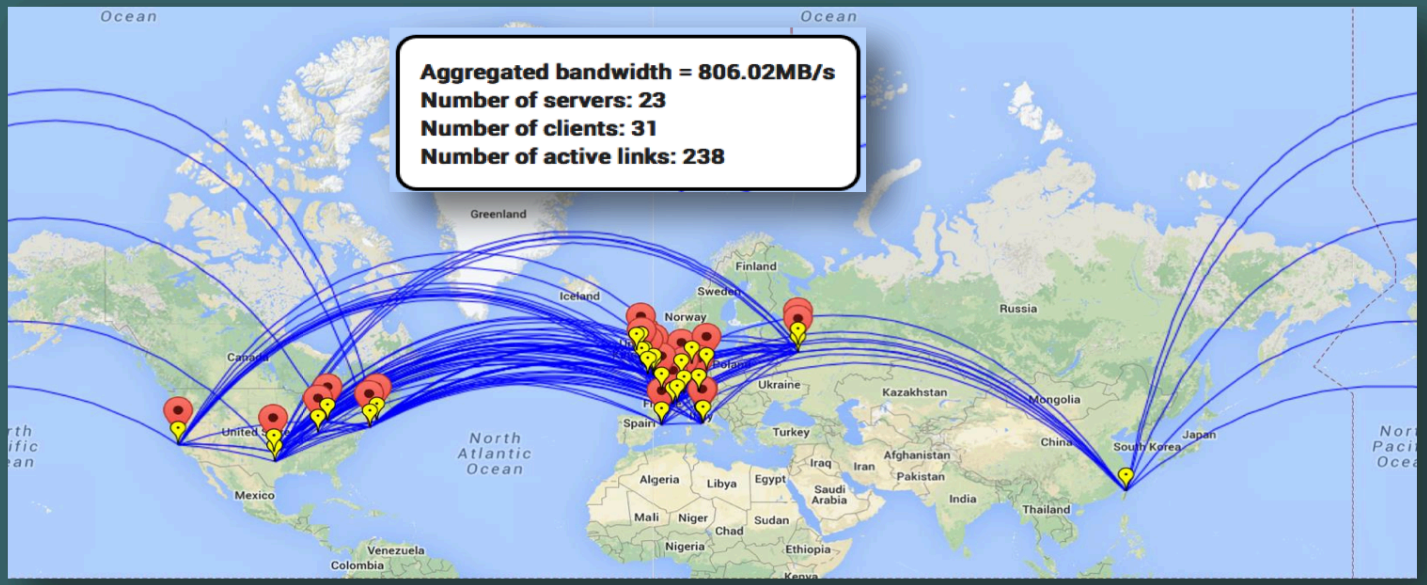
Federated Storage - ATLAS

- FAX – Federated ATLAS Xrootd, Read-Only
- Can run jobs on sites with no local storage or use local storage as cache
- rucio will simplify filename/file lookups
- xrootd used because it fits with HEP access (ie ROOT) plus scalable
- Ultimate goal for site to be able to still run jobs with no local storage (eg during upgrades)
- WAN can be as good as LAN (but not always).

What is FAX?

FAX (Federated ATLAS Xrootd) is a way to unify direct access to a diversity of storage services used by ATLAS

- Read only access
- Global namespace
- Currently 42 federated sites
- Regions covered: US, DE, UK, ES, and CERN



But Not only that!

- Initial use cases
  - Failover from stage-in problems with local storage
  - Gain access to more CPUs using WAN direct read access
    - Allow brokering to Tier 2s with partial datasets
    - Opportunistic resources without local ATLAS storage
  - Use as caching mechanism at sites to reduce local data management tasks
    - Eliminate cataloging, consistency checking, deletion services
- WAN data access group formed in ATLAS to determine use cases & requirements on infrastructure

Federated Storage - CMS

- Traditionally move jobs to data (data transfers were slow) but this is becoming problematic
- Goal is Any data Anytime Anywhere (AAA)
- Local access still better but not by much (6%), trading off latency with bandwidth
- Can redirect jobs away from busy sites with data to another site then remote access the data

Any Data, Anytime, Anywhere (AAA)

- Goal: make all data even more straightforwardly available to any CMS physicist, anywhere
  - Reliably: no access failures
  - Transparently: never notice where the data actually reside
  - Easily: no operational burdens for physicists to have local access
  - Universally: fulfill the promise of opportunistic grid computing
- Technical solution is federated storage: a collection of disparate storage resources transparently accessible across a wide area via a common namespace
- NSF-funded US CMS effort based at Nebraska/UCSD/Wisconsin to achieve these goals and propagate to CMS as a whole

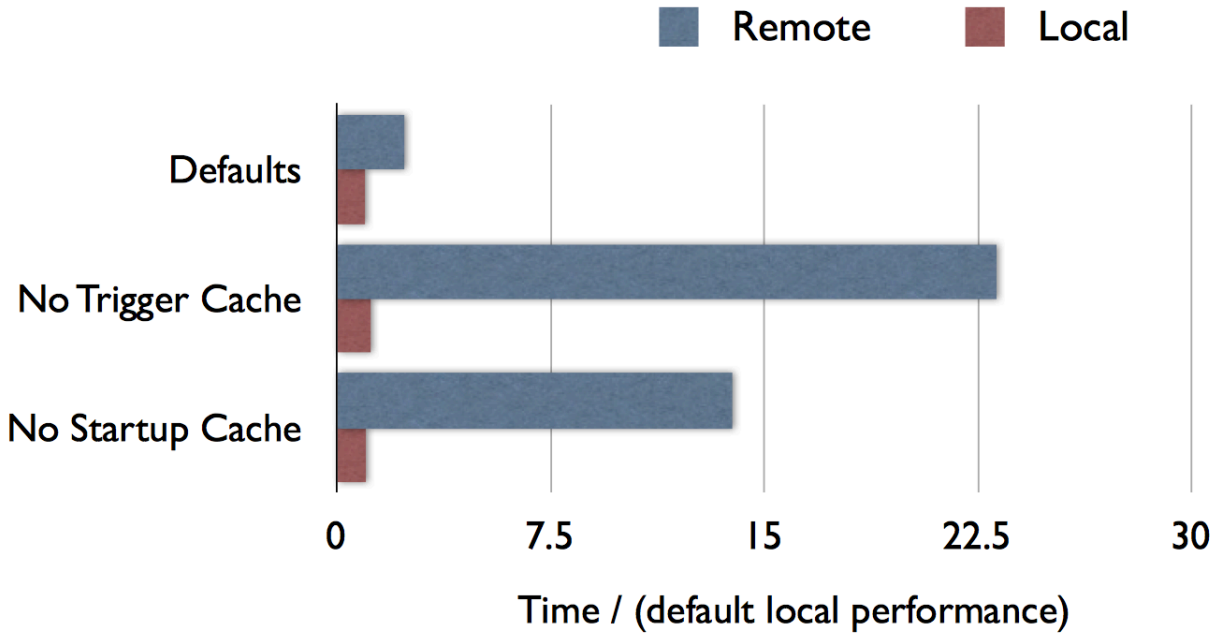
Federated Storage - CMS

- Increasing data federation over high latency links (WAN, wireless, desktops etc)
- Trying to reduce number of reads + reduce data read + parallise access, to increase efficiency
- Ttreecache essential to get good performance on high latency links
- Very successful

High-Latency is the future!

- We have seen increased interest in data federations within the WLCG.
- I thoroughly believe that this model is appropriate for HEP outside LHC.
  - It is important to identify approaches we can feed back into ROOT.
- If we continue to target smaller computing resources, departmental clusters, and individual laptops, the network will only get worse!

Summary - Avoiding Network Round Trips



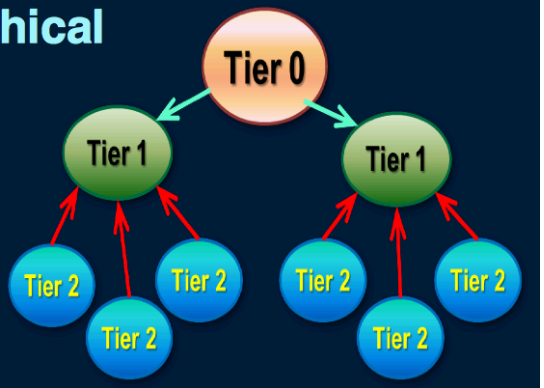
Not shown: ROOT defaults (no TTreeCache) reading remotely is 177x slower than CMSSW's defaults reading locally!

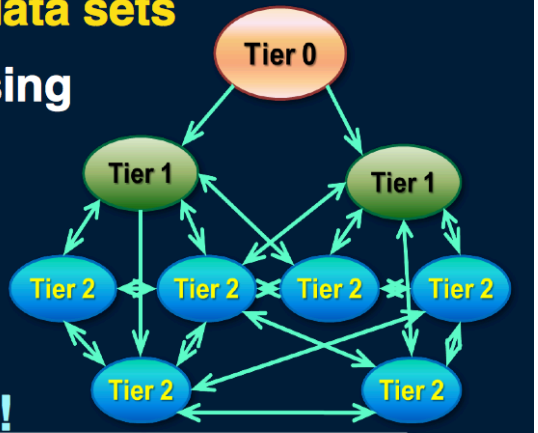
Federated Storage - Network

- Error-free WANs are essential for high volume data transfers
- NO FIREWALLS
- Remote IO != flexible data if the framework isn't efficient

LHC Computing Model Evolution

- The original MONARC model was (largely) hierarchical
- Main evolutions introduced since 2010:
  - Meshed data flows: Any site can use any other site as source of data
  - Dynamic data caching: Analysis sites pull datasets from other sites "on demand", including from Tier1s and Tier2s in other regions
    - Combined with strategic pre-placement of data sets
  - Remote data access: jobs executing locally, using data cached at a remote site in quasi-real time
    - Possibly in combination with local caching
- Federated Data Systems: FAX, PhEDEx, Alien
- Variations by experiment; but a common element is: Increased reliance on network performance !

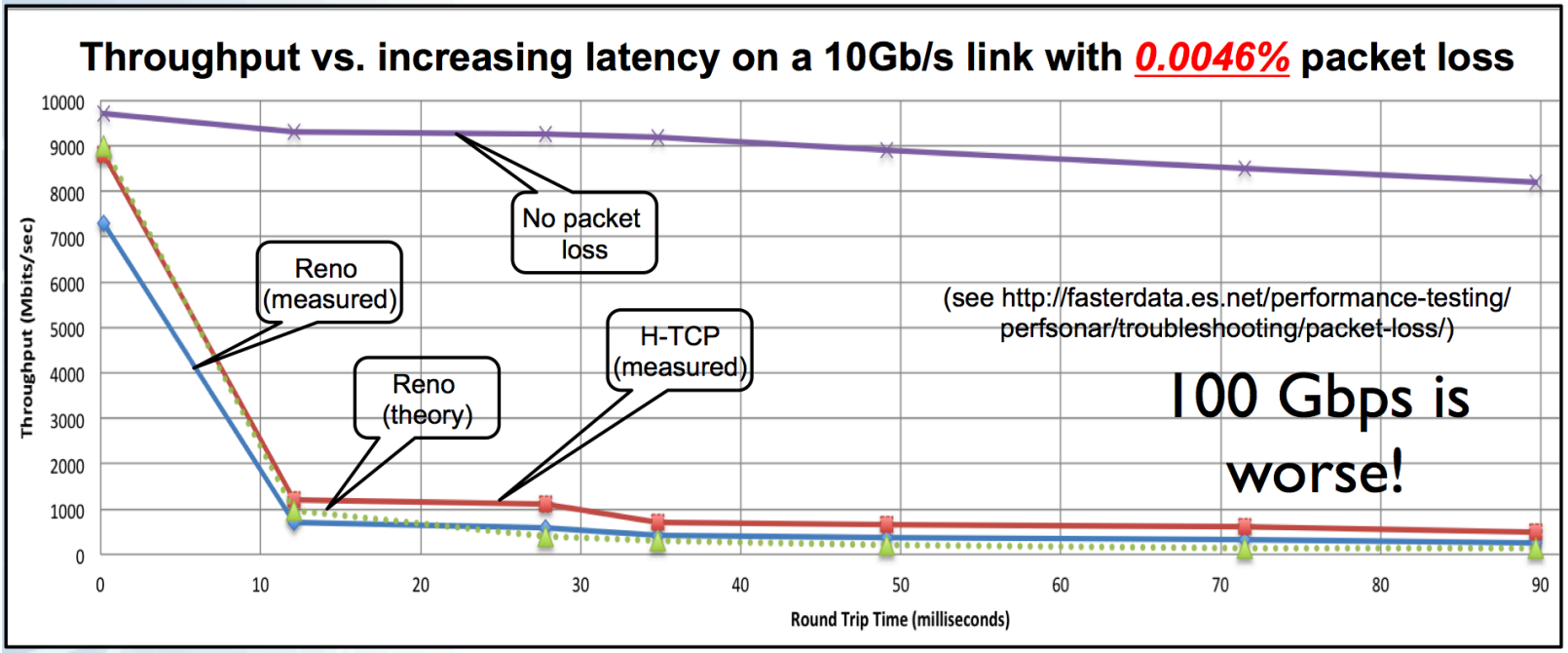




Network Inflexibility?!?

- Networks are the only entity listed as both helping and hindering flexible data. The culprit? Humans and TCP!
  - **TCP is a glass workhorse**. At high-bandwidth and high latency TCP is extraordinarily sensitive to networking problems.
  - Great networks don't live in isolation. For a given flow, one must consider all the pieces involved - endpoint hosts, campus networking, regional networking, and backbone networks. **The humans who run these networks must collaborate closely to fix problems**.
  - To achieve great TCP rates, all must work without a single error or misconfiguration. Error free end-to-end paths are not easily achieved.
    - All our network operators are great, but we place them in an impossible situation.
- Recent trends - such as performance monitoring (perfSonar) and Science DMZs - have made errors easier to spot and less likely to occur.
  - Yet TCP dictates we must have precisely zero errors!

A small amount of packet loss makes a huge difference in TCP performance



- On a 10 Gb/s LAN path the impact of low packet loss rates is minimal
- On a 10Gb/s WAN path the impact of low packet loss rates is enormous

- ***Implications***: error-free paths are essential for high-volume data transfers

Slide courtesy of Jason Zurawski

perfSONAR powered

The Others

- Lots of other stuff going on:
- Cluster Filesystems (CEPH almost perfect)
- SSDs (Caching)
- Data Preservation
- Software Build Systems
- Using Databases for Analysis
- Configuration Management with Puppet
- Software-defined Networking
- ROOT/Geant4 Updates
- Service Monitoring
- …

Thank You.

Questions?