Issues and Perspectives in LHC Computing

Federico Carminati

LHC Days in Split 2010

October 9, 2010

Acknowledgments & disclaimer

- Thanks to ATLAS and CMS for the excellent material for this talk
- All the opinions expressed in this talk are my own, and so are all the mistakes...

Computing @ LHC

- ~15 years of preparation of the software
- ~10 years of preparation of the Grid
- "Epic" fights, prophetic statements, heroic efforts, promises of revolutions…
- Does not really matter who was right and who was wrong…
- But it really does matter what was right and what was wrong

Experience gives you the ability to immediately recognise a mistake the second time you make it.

General David Petraeus

The idea is to be able to do a bit better than that

Before we start

- Despite many technical difficulties, the LHC computing is a success
  - All experiments have been able to show very quickly results
  - The improvement rate in the quality of the analysis presented is impressive
- This is the first time in the HEP history that interesting results of such quality have been shown so rapidly
  - This is a proof of the maturity of the simulation, reconstruction, calibration and analysis
- We must have been doing something right
  - And we clearly have used well the "extra time" we had

From: Federico Carminati
Sent: 24 November 2009 12:31
To: Ian Bird; Frederic Hemmer
Subject: Thanks!

Dear Ian and Frederic,

proper recognitions and thanks will be given in due time and place. I just want to drop you a short note to thank IT and WLCG for the exceptional support. Grid, Batch, Castor, CAF and Networking worked flawlessly. The first collisions seen in ALICE were reconstructed with the standard system, no hacks no patches, one hour after being taken, at CERN via the Grid. A big thanks to you all for your professionalism and dedication. Please pass this down as appropriate. Best regards,

Federico Carminati

The Language

- The move to C++ did not fulfil most of the prophecies
  - Physicists still write the code, they do not "ask Computer Scientists" to do that
  - Physicists do not program in UML
  - Templates blow up the code, virtual tables slow it down
  - The code is NOT simpler (quite the contrary) or more efficient (or only after careful optimisation)
  - Compilation is slower, memory footprint larger, memory fragmentation a curse, debugging a nightmare
  - FORTRAN compilers did not disappear
- However this is far from being the most serious drawback
  - Only some "senior physicists" have migrated to C++
  - The others are left with Excel….

What the alternatives?

- C++ is a very complex language
  - The FORTRAN standard is 50 pages, the C++ one 700 (!)
  - But the problem we have to solve is complex
- Will F9x have been any better?
  - Users would have had to learn it anyhow
  - Beware of "faux amis"… moving from F77 to F9x could have been a problem
- We could have tried to implement LHC software in F77 + Zebra
  - Not sure it would have scaled to O(10k) data structures (classes)
  - LHC software is ~10x the last F77+Zebra packages
- Some things were lost
  - Some aspects of the ZEBRA I/O with "self describing" structures are not entirely matched even by root
  - The difference between structural and reference link

What the alternatives?

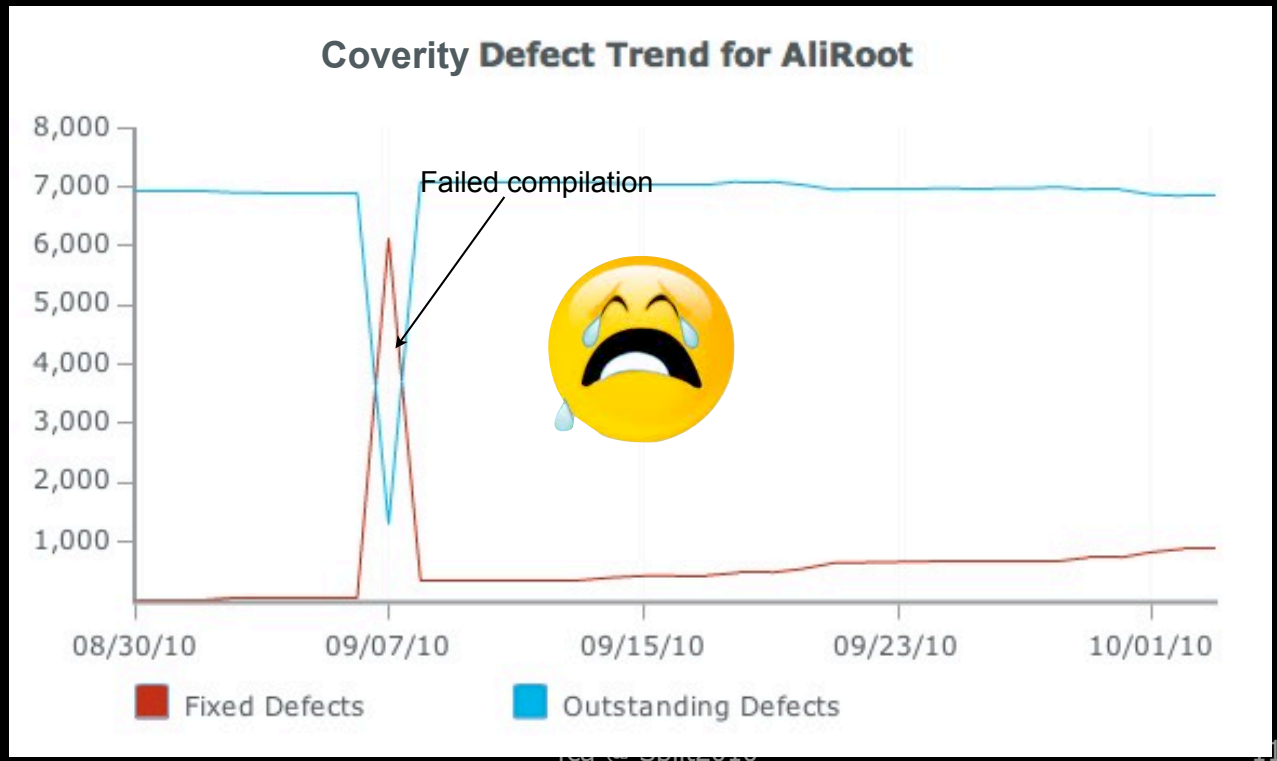
- Moving to Java would not have simplified the problem itself
  - Plus all the intrinsic weaknesses of Java: speed, math lib…
  - Most of Java VMs are written in C++
- One of the main problem is the richness of C++
  - Training and coding discipline have been lacking
  - Coding conventions are very difficult to enforce
- Proper encapsulation of complexity turned out to be a real experts' job
  - It takes vision, experience and strong leadership
  - Committees usually produce C++amels

C++ a huge mistake?

- Probably not
  - Shared libs, polymorphism & inheritance, plugins
  - Integration with graphics, OS and Grid
  - Support for very large codes
- However
  - Most of us were grossly incompetent
  - We had to learn "on the job" (and we are still learning)
  - Some mistakes were very difficult to root out
  - Others will stay there till we will rewrite everything
- A whole generation (1996-2001) of common projects has been the victim of this process
  - MOOSE, LHC++, Objectivity, SPE, SEAL, PI…
- Design errors are very costly and difficult to correct
  - XP "merciless refactoring"

Code quality

- AliRoot had 240,000 TClonesArray deletion / event…
  - Now this is down to few hundreds / run
- We are still far from understanding what the compiler does to the code
  - And we have already to care about GPUs, multicores…

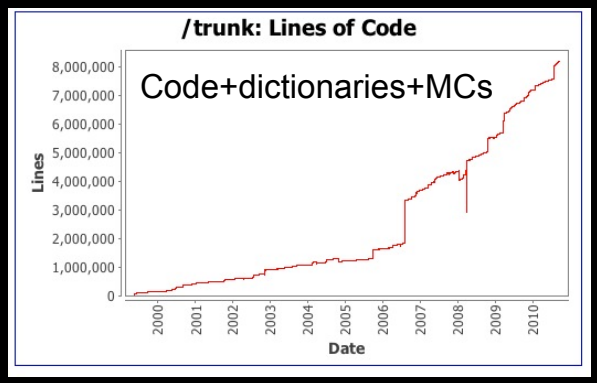


The common projects

- Out of the committee-started ones only two survived
  - Geant4 and JCOP – two real successes?
- Some "spontaneous" ones survived
  - Gaudi, MonALISA
- Salvation came from the "garage"
  - root, xrootd, athena, phedex ...
- A lot of questions
  - What did we learn? How can we be sure to be lucky again in the future? Which of the ones who bit the dust we are missing most?
- Software process and Software Engineering deserves a talk of its own

Code maintenance

- Beyond svn/cvs, library management tools are minimally used
  - Whole framework releases are the norm
  - Single library/package release seems to be extremely hard
- Balancing rapid evolution and stability is a unmatched challenge
  - In ALICE we have weekly release and trunk tags, we can update-tag-deploy in a matter of hours if needed
  - Still we are blamed for a cumbersome release cycle AND for lack of stability



The Grid

- It all started for us in 2001 (CHEP Padova)
- Parallel efforts at least in three places
  - US: PPDG, GriPhyn, VDT, OSG
  - EU: EDG, EGEE
  - Nordic Countries: NorduGrid, ARC
- Gigantic effort O(1B$)
- No (real!) critical review of the outcome
  - But to have an idea just read the EDG technical proposal
  - … with good glass of your preferred liquor

The Grid – the bad news

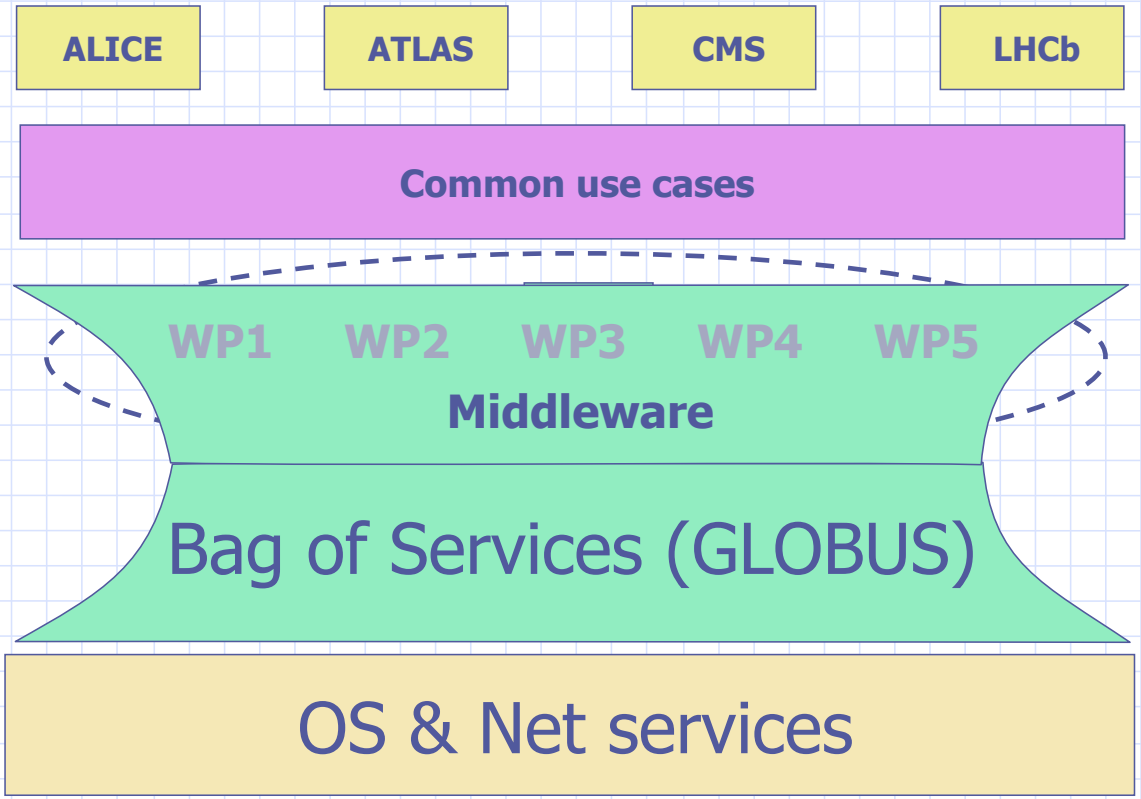
- Failed to address (recognise?) the data-centric nature of our problem
  - Too much time spent on the WMS, too little on data management
  - Objectivity was addressing the right problem, but way too naive and with the wrong tools
- Failed to join forces in a common effort between the different projects
  - The commonality (painfully) stopped at Globus for EGEE / OSG
- Failed to join forces between experiments
  - Who remembers HEPCAL?
  - Why? Several partial answers, but not a real good one

A proposal

If we manage to define a set of common use cases we will be easier for them to arrive at

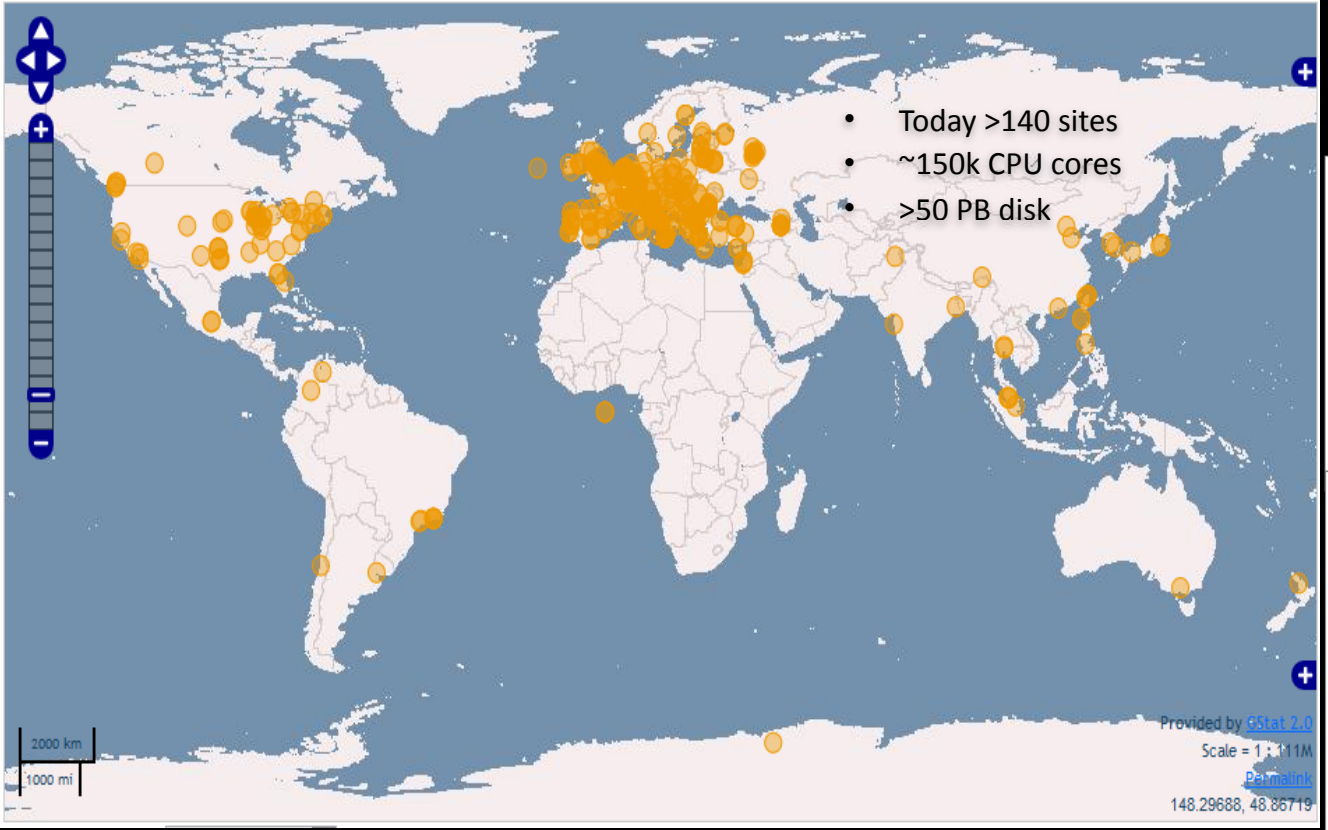
Specific application layer

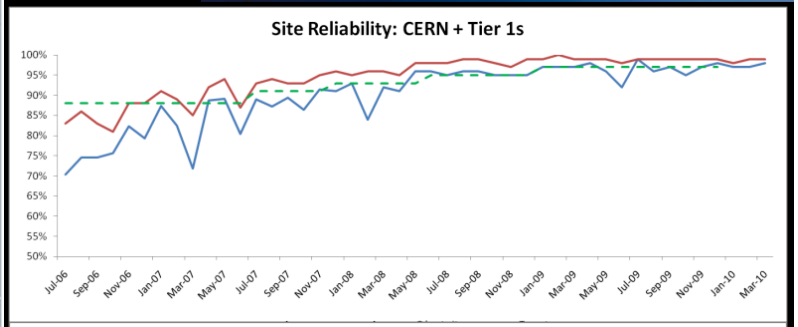
VO common application layer

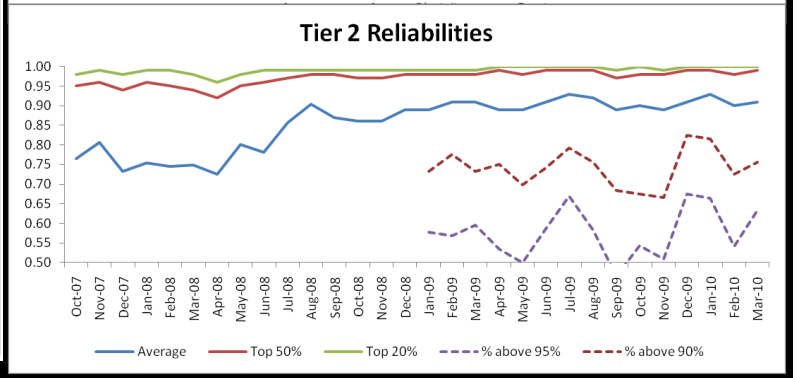


The Grid – the good news

- Experiments are using it routinely (!)
  - It works, simply
- The establishment of a world-wide community of centres working together is an indisputable triumph
- A very large number of young scientist are now familiar with the Grid
- WLCG is a stable platform where to experiment new ideas
- The Grid has really enable digitally challenged countries







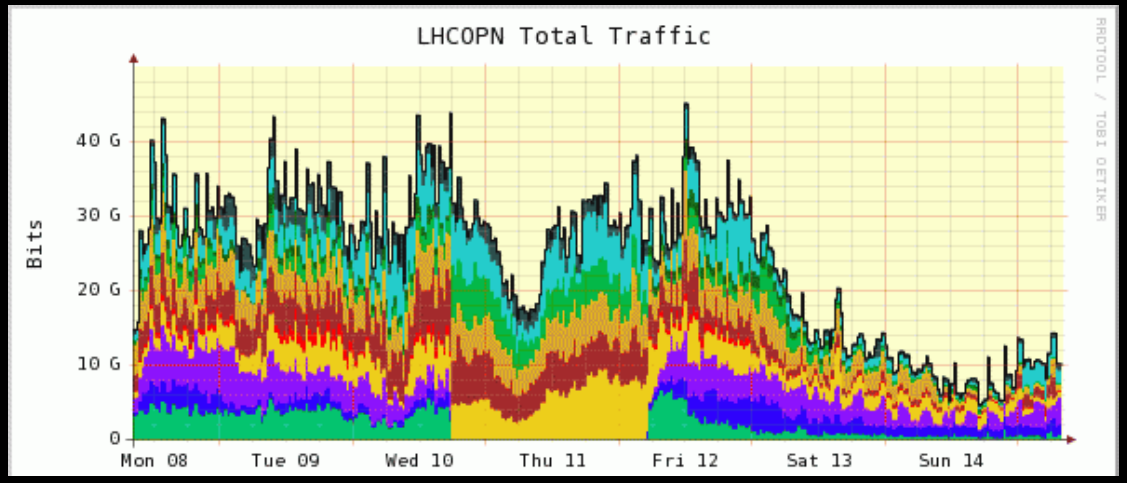
- Site readiness very good
- However the most common reason for job failure is indeed site misconfiguration
  - Apart of course from user errors
- Instabilities coming from BDII are less frequent now
- Good news is that availability is constantly improving
  - Even if the human cost can be very high

CASTOR

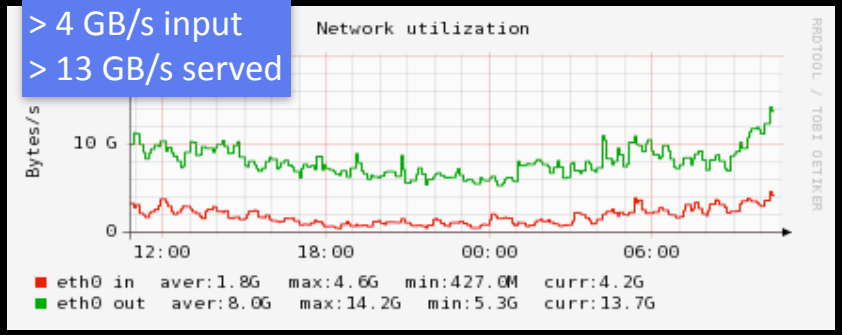
- Ten years of data challenges… and then we simply changed the product at the last minute
  - But it was a courageous and successful move
  - The accumulated experience was an invaluable asset to a successful change
- Data collection at T0 is one of the things that worked best
  - Processing infrastructure steady and reliable
  - T0 is a limit to data taking, and dreams are allowed (ALICE + CMS multiplying HI rate by factors days before the run!)

The Grid – data transfer

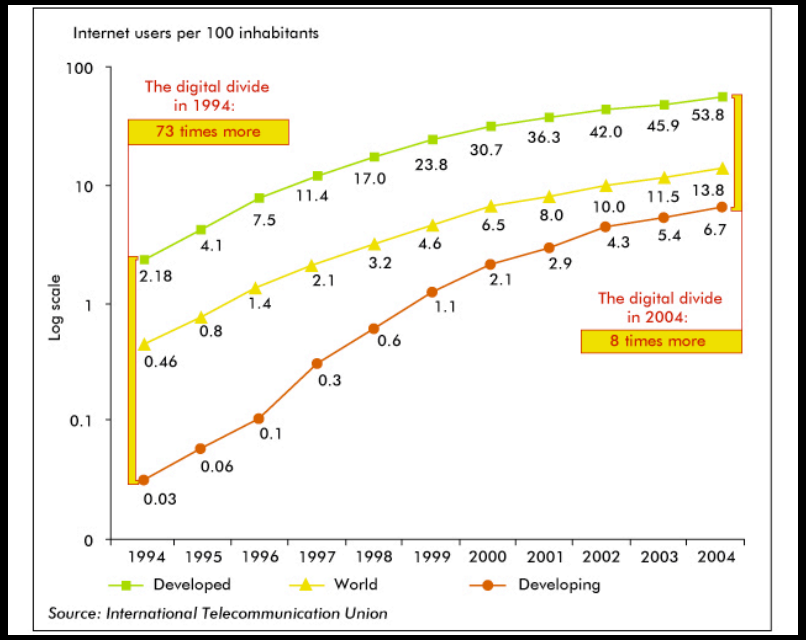
- Data transfer has been especially successful
  - Out of CERN has peaked above 1GB
- Transfer between centres also very good

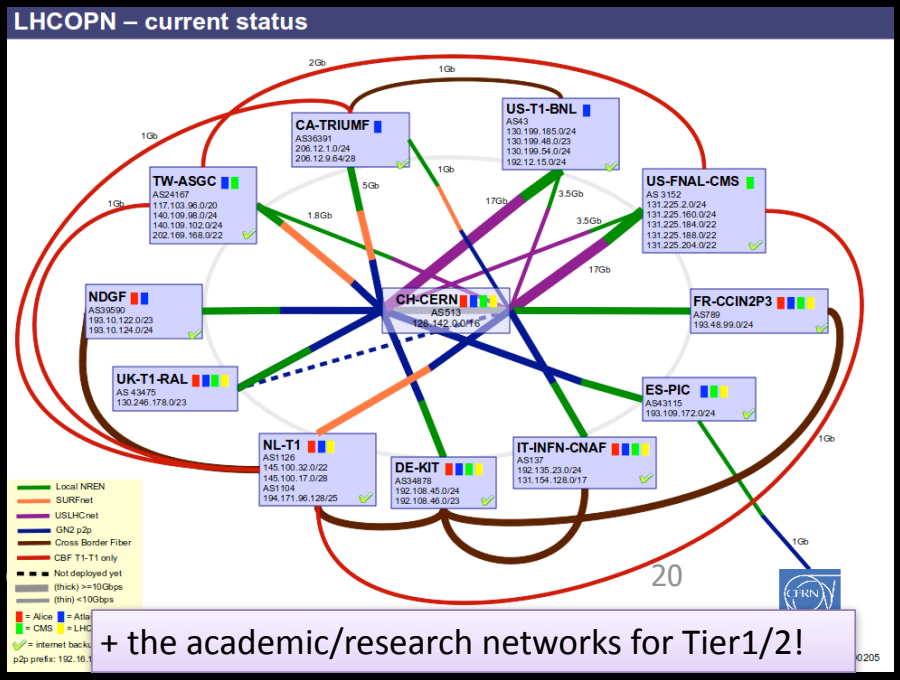


Tier 0 traffic:
> 4 GB/s input
> 13 GB/s served



- The network is probably the best surprise here
- Still the least oversubscribed resource we have





+ the academic/research networks for Tier1/2!

At present, institutes in Europe typically have a ~ 1 Mb/s access to CERN. In some places the available bandwidth is already as high as 622 Mb/s. We expect Gb/s networks to be available by the beginning of LHC operation. This assumes an increase of a factor of ~ 100, which is typical of the improvements in the technology over a ten-year period. However, current price trends would imply that achieving this performance would require an increase in network funding.

CERN/LHCC/94 43
LHCC/P2
15 December 1994

Technical Proposal

for a

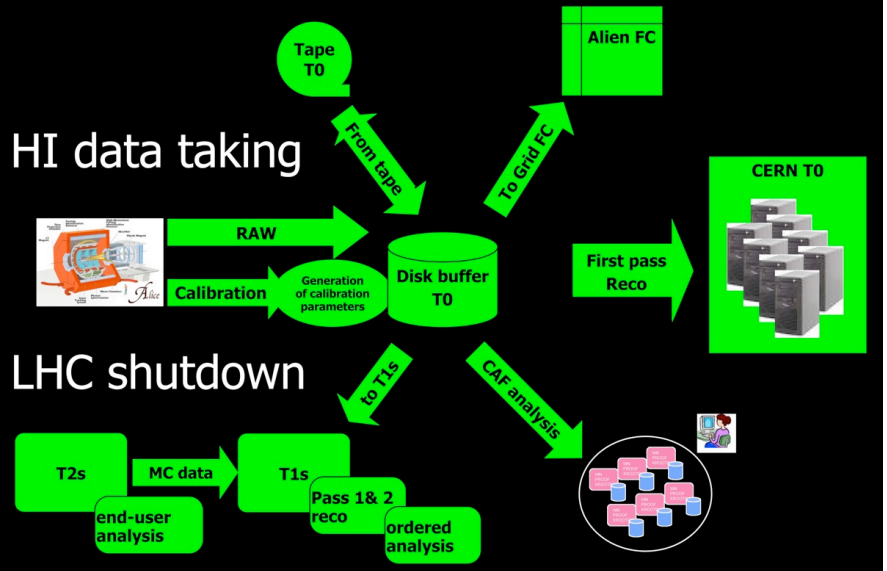
General-Purpose pp Experiment

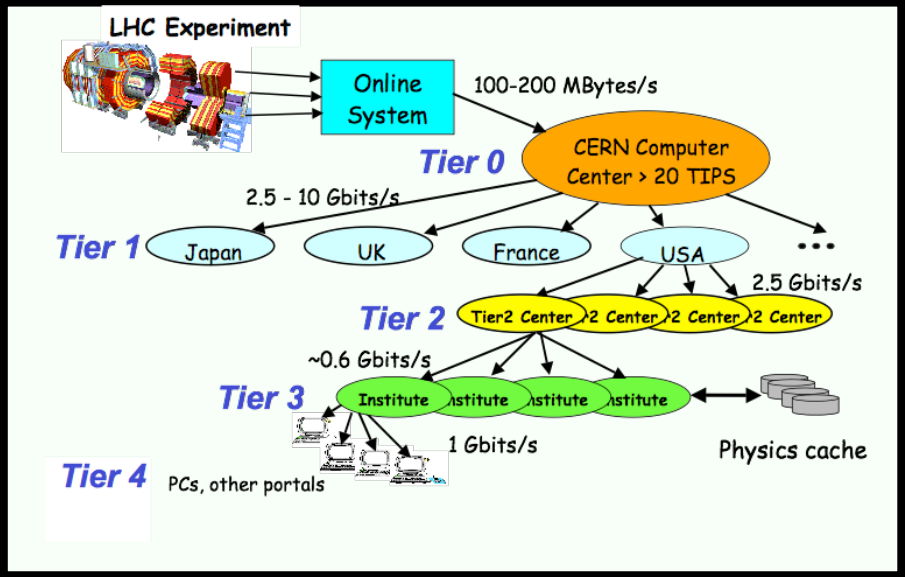
at the

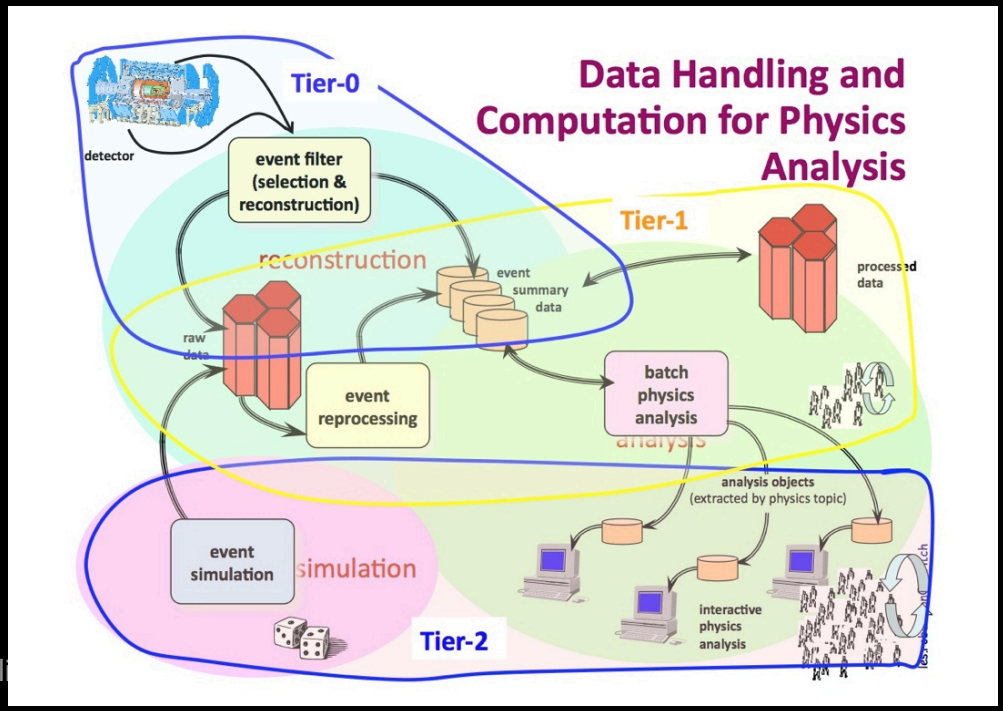
Large Hadron Collider at CERN

The Grid – it all started with MONARC

- The computing models are remarkably similar
- T0: First reco, data storage
- T1: Subsequent recos, MC reco, ordered analysis
- T2: MC, Chaotic analysis
- T3: End user analysis
- ALICE does not have (or admit having…) T3s

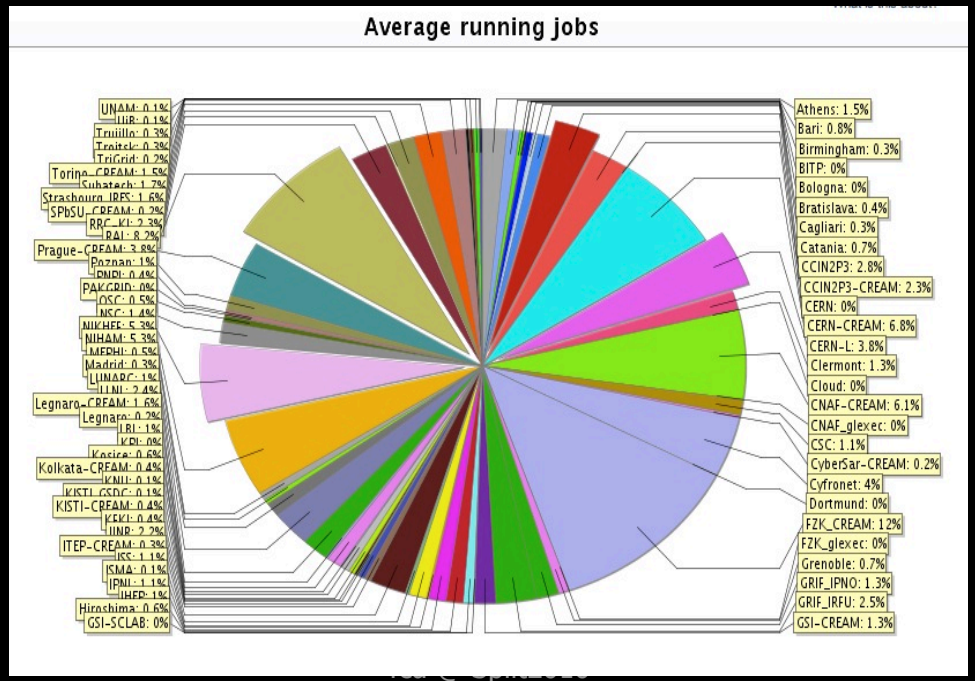






The Grid – relations T1-T2-T3

- T2 have been a very good surprise
  - More than 50% of the work in ALICE is done by T2
- The Grid is becoming more and more "cloudy"
  - Not really clear what is the difference between T1s and T2s apart from data custodial and better network
    - but the latter is about to change - OPNng



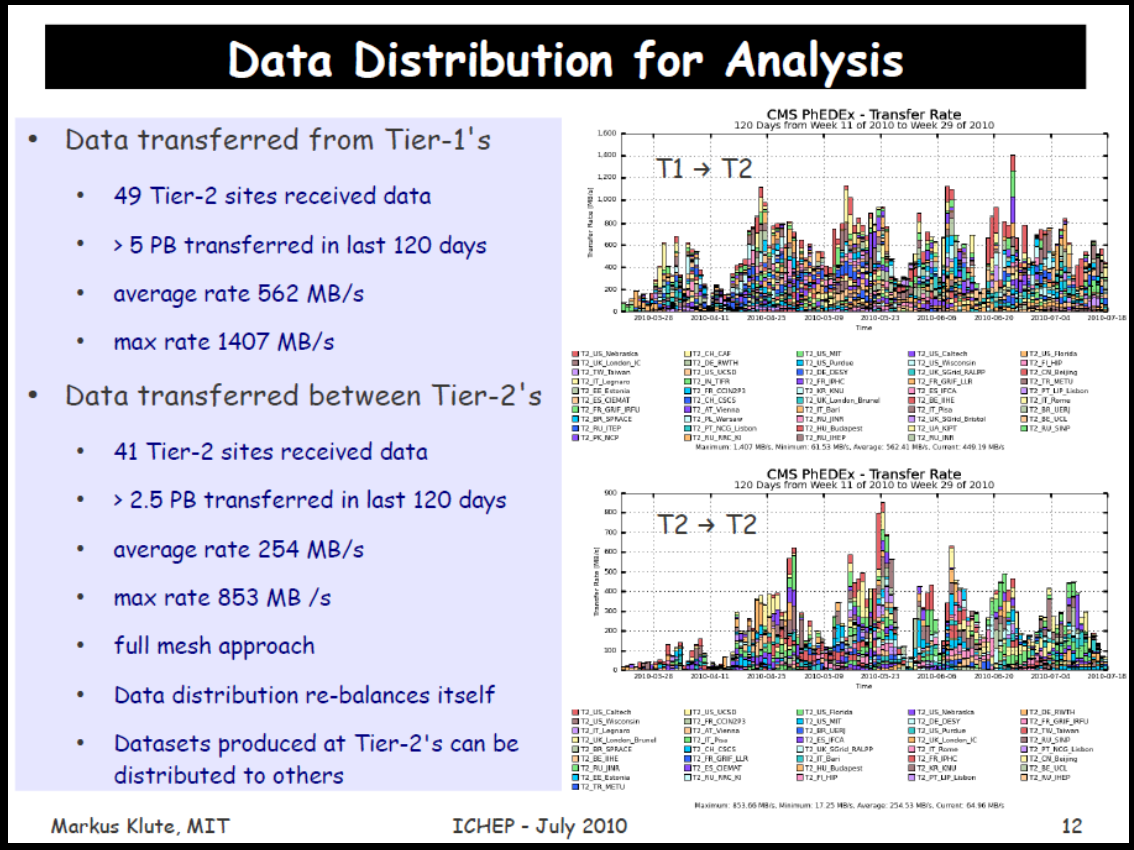
The Grid – Job management

- The priority and quota mechanism is hard to implement Grid-wide
  - Central queues (ATLAS, ALICE) are a single point of failure / bottleneck
  - Distributed queues (CMS) makes it more difficult to manage priorities
- Permissions and quotas on files are also a problem
  - See above for central vs distributed catalogues
- "Upgrading" the Grid is a very long process
  - CREAM
  - SL5
  - glexec

Data is still the problem

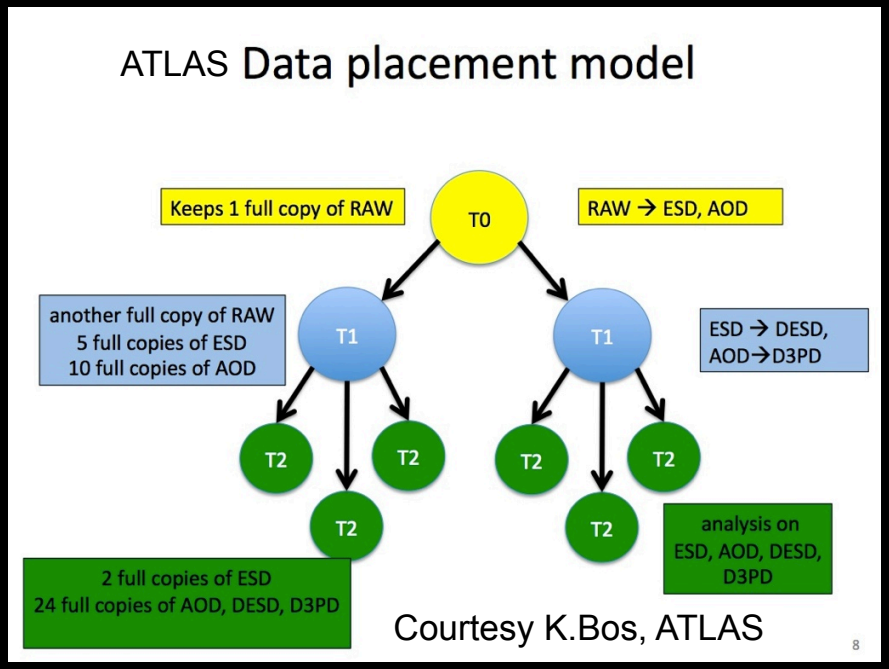
- Data placement is the main problem, particularly for analysis
  - "predictive" data placement for ATLAS and CMS
  - "opportunistic" data placement for ALICE
- Data distribution "per se" works very well

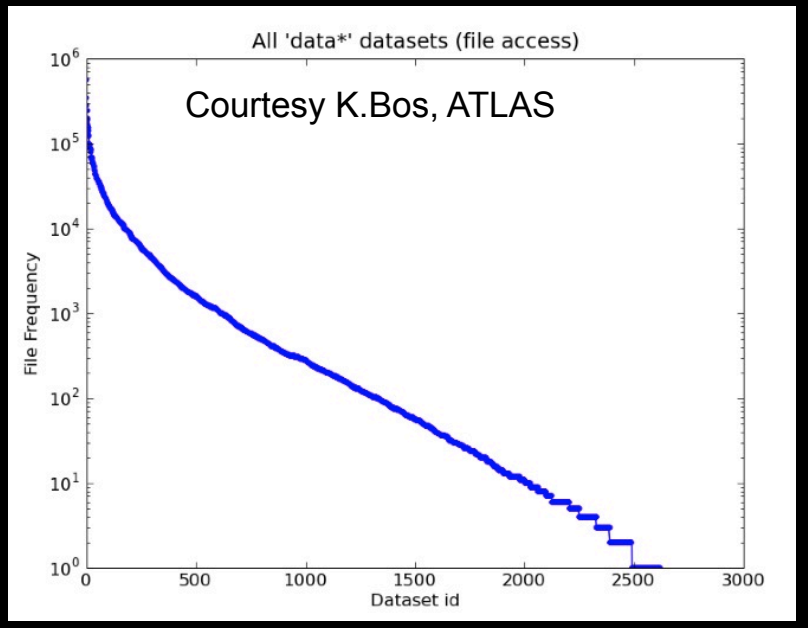
- With "infinite" disk space the two would be equivalent
- "opportunistic" data distribution depends on a single central catalogue
  - It took ALICE 10 years to get there!



Data placement

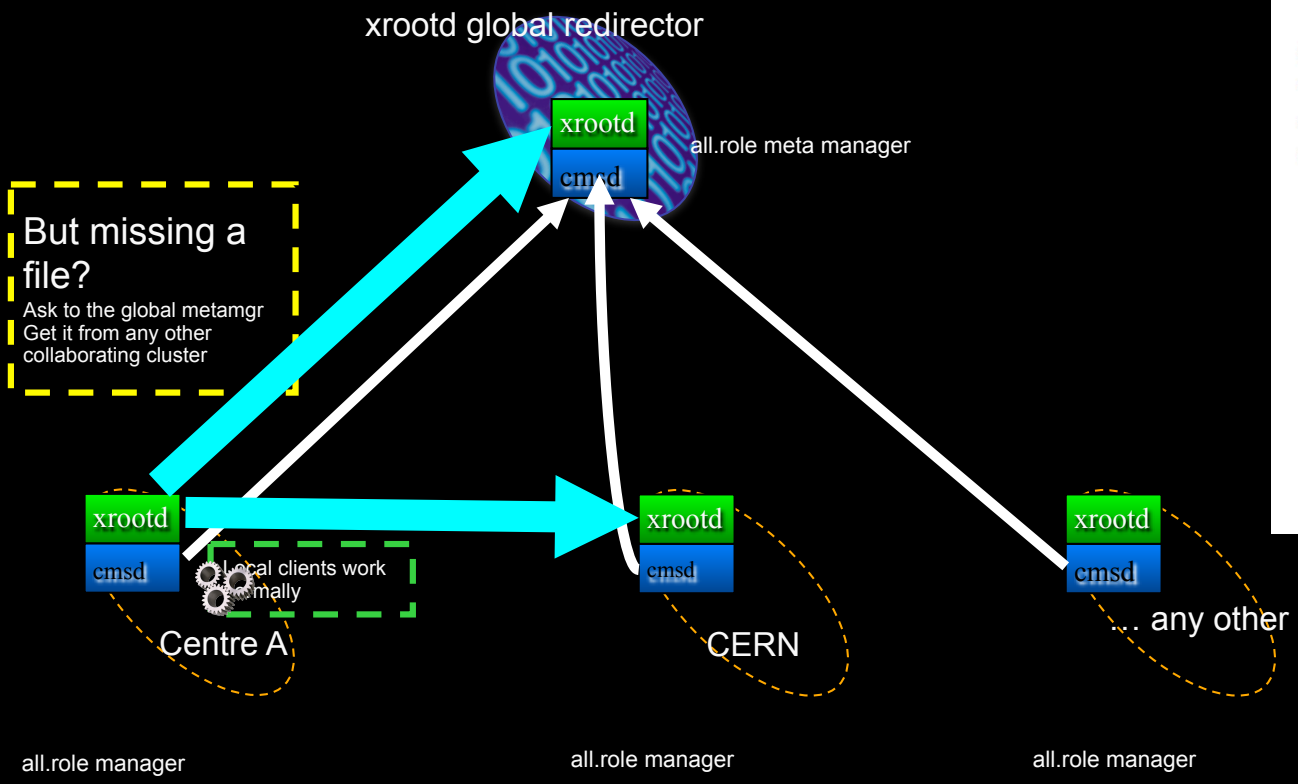
- Difficult to predict when a dataset is popular and when not
  - CMS has 8 orders of magnitude in access between popular and non-popular files!
- Dynamic partial file caching would probably help

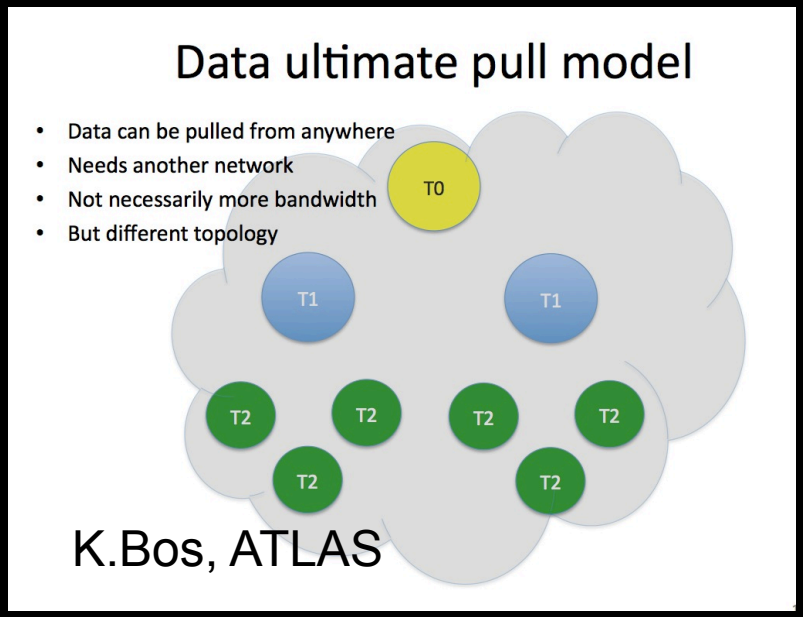




- For ALICE and CMS pretty much the same
- Data duplication increase disk space requirements
- It is now clear that online tapes are not the solution

Complete "pull" model?





- With careful caching and overlapping access over the network can be slower by a factor 2-3
- xrootd offers this now
  - Will other products go the same way soon?

Evolution of requirements

- No longer T0-1-2-3 hierarchy
- T1 and T2 will become equivalent in the network (OPNng)
- Traffic between countries as much as within
- No longer disk space but network bandwidth will scale with #users and #data
- More analysis-user demand driven traffic pattern rather than the DC of pre-placement
- We must have intelligent layer for data brokering
- Bandwidth between each of the sites must be at least what we now have between a T1 and T2 site within a cloud

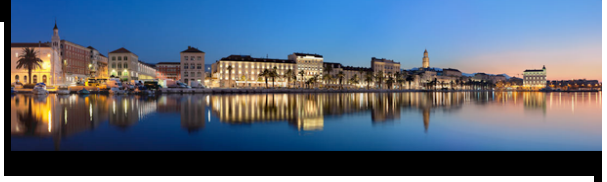
K.Bos, June 2010

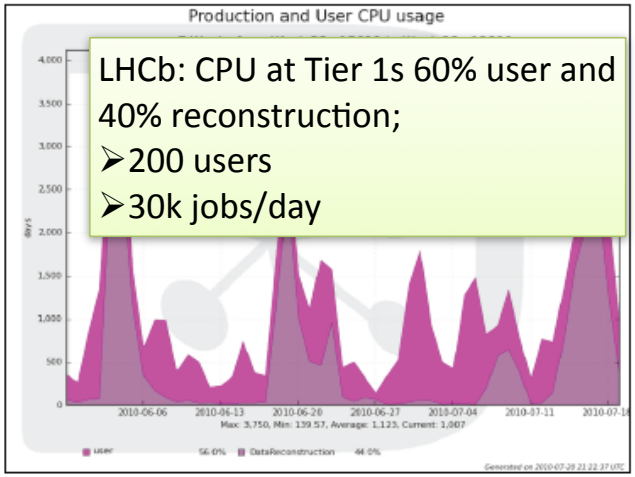
Other data woes

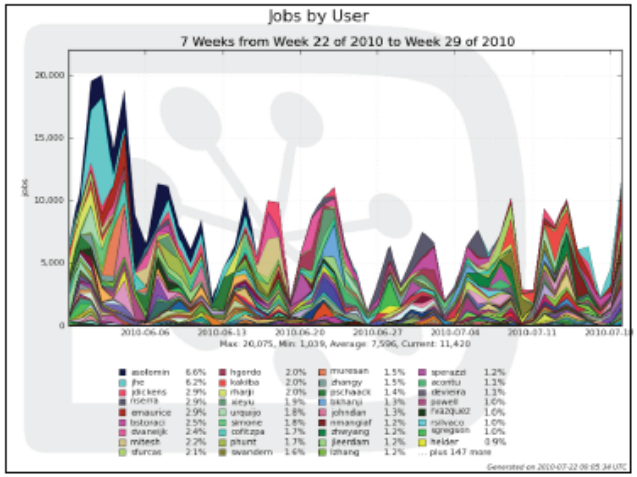
- If we distributed "generously" the data, deleting them can be a real trouble
  - This is still sorely true for CASTOR!
- Increasing the disk space at the centres is more difficult than increasing CPU size
- Global data monitoring and quotas are more difficult to implement than CPU quotas
  - Understanding the data access pattern will be a must
  - But do we have the tools for that?
- ATLAS has "factorised" the problem introducing "clouds"
  - But they are already thinking to "bridge" clouds

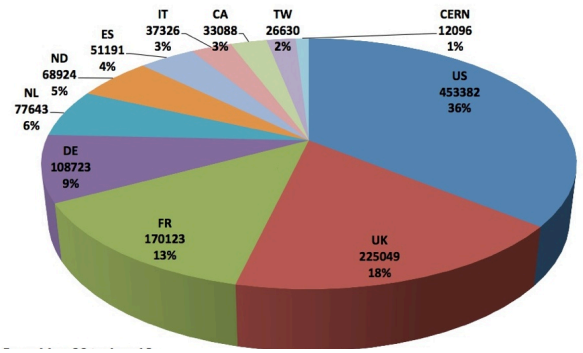
Data processing

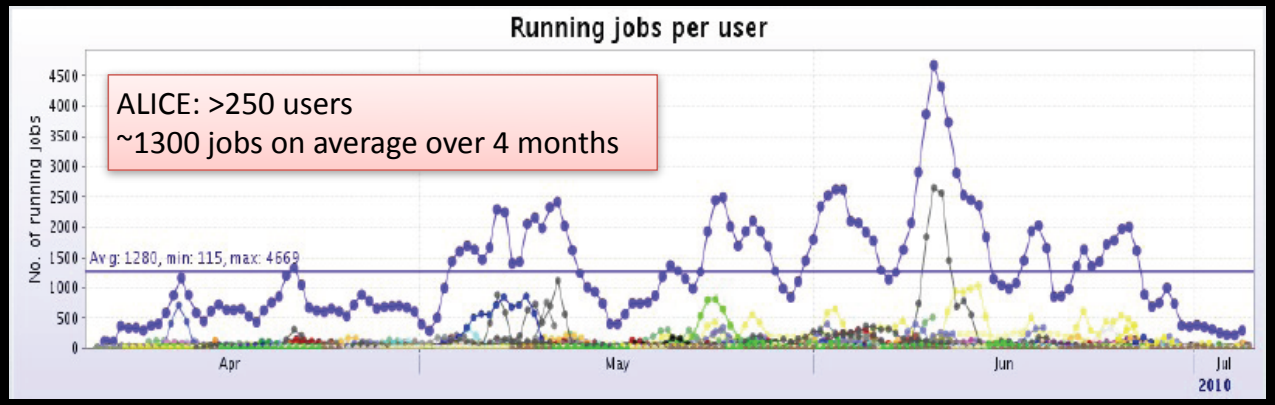
- Data processing @T0 and reprocessing @T1 successful for all experiments
  - Early data has been available for analysis for all experiments
- However it is a very people-intensive exercise
  - The emphasis will be on automation
  - Not having a common approach here will hurt, we will have to invent four different wheels to do the same thing
  - Again data management is at the heart of the problem

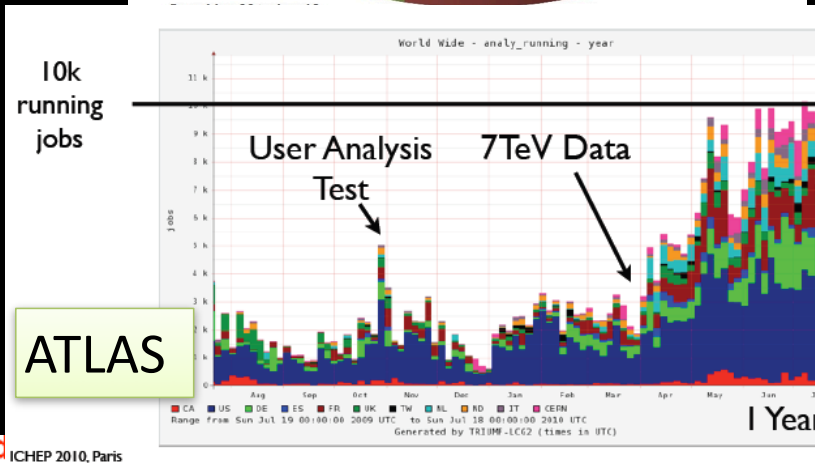


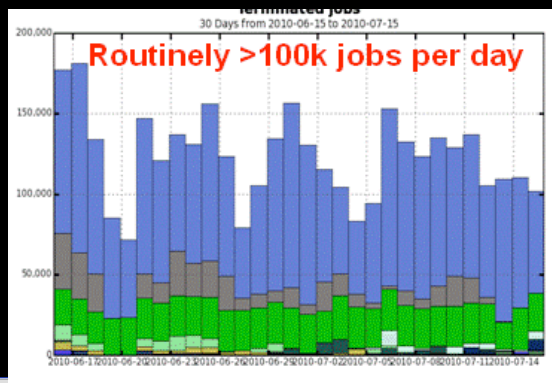


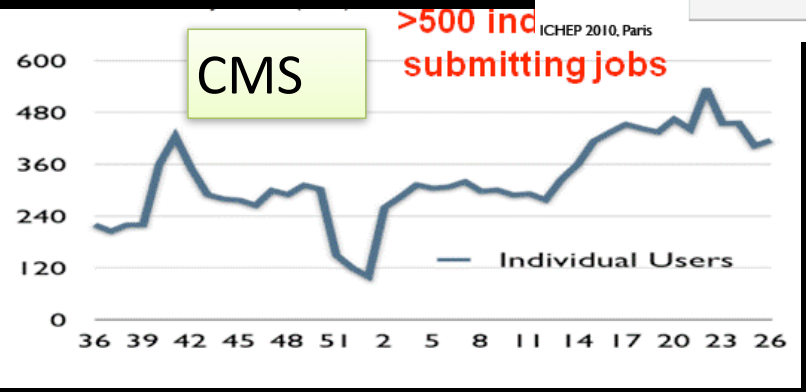












GRID-based analysis in June-July 2010:
>1000 different users, ~ 11 million analysis jobs processed

Analysis

- Much more successful than anticipated
  - At least by ALICE
  - We can really do analysis on the Grid
- In some sense analysis is victim of its own success
  - In ALICE users are "abusing" the "par file" system
    - Local compilation of code fragments
  - The access to the calibration database from analysis jobs is overloading the AliEn catalogue
  - In ATLAS the Data Distribution Model is running way above the design values
- Multiplication of data formats and reduction in the file size is a common curse

Analysis objects

- Some problems in the reconstruction can to be corrected in the ESDs
  - So the ESDs are the "preferred" format to get the best results "before next reco pass"
- The global size of replicated ESDs gets prohibitively large
  - It can exceed the size of raw by factors
- Smaller and "filtered" analysis objects are more efficient
  - However the "inflation" of analysis formats can be a big problem for management of the code
- Part of the problem is fast code evolution
  - But this cannot be changed for the moment, we are learning
- The other solution would be a more flexible data distribution strategy
  - From push to pull mode

Conclusions & perspectives

The process

- Exploring different roads was unavoidable, enriching and very useful
  - Our current success is the result of it
  - Harsh and frank criticism was also good
- But
  - Doing it along experiment lines was wrong
  - Too much animosity prevented any real conclusion to be drawn to avoid (good) people to lose face and (good) work to be bashed
- This is a doubtful legacy for the future projects

The code

- Move to C++ was probably inevitable
  - But it made a lot of "collateral damage"
  - Learning process was long, and it is still going on
  - Very difficult to judge what would have happened "had root not been there"
- The most difficult question is now "what next"
  - A new language? there is none at the horizon
  - Different languages for different scopes (python, java, C, CUDA…) just think about debugging
  - A better discipline in using C++ (in ALICE no STL / templates)
- Code management tools, build systems, (c)make, autotools
  - Still a lot of "glue" has to be provided, no comprehensive system "out of the box"

The Grid

- A half empty glass
  - We are still far from the "Vision"
  - A lot of tinkering and hand-holding to keep it alive
  - 4+1 solutions for each problem
  - We are just seeing now some light at the end of the tunnel of data management
- The half full glass
  - We are using the Grid as a "distributed heterogeneous collection of high-end resources", which was the idea after all
  - LHC physics is being produced by the Grid
- Due to the LHC schedule change we are possibly overprovisioned at the moment
  - But it won't last...

The Grid

- If
  - Some of the basic MW is "pushed down" into the OS
  - Some "home grown" (EGEE/OSG/ARC) MW "moves up" in functionality to include some of the experiment-specific MW
- We have a virtuous cycle
  - However there is no sign of this happening
- Commonality would still be the best approach but it seems very difficult to achieve
  - Do not ask me why

Data management

- We are just ahead of the technology and of the data models offered by industry
  - We really need a single global namespace, ubiquitous access, intelligent replication and caching, redundancy and resilience
- The alternative is
  - Use home-grown solutions (I see only xrootd there but...)
  - Wait for "standard" solutions: lustre, NFS-4, hadoop (the wait can be long before we have a complete solution)
- Probably we should do the first and watch attentively the second
  - And plan carefully the transition
- The load of analysis will be the "reality check" that will introduce pragmatism

Catalogues!

- What we want is a single name space
  - A "single" catalogue becomes quickly overloaded
  - Distributed catalogues are hard to sync, the more you add the farther you go from the target…
- So we should do less
- Make a single catalogue as light as you can
  - Take out of the catalogue everything that can be found dynamically
  - Put all that into a self-updating cache, with good error recovery
  - Go for LFNs only and forget GUIDs
- I know it sounds terribly like what ALICE is trying to do with xrootd, but
  - I have no better idea here
  - I am open to suggestions...

Grid need-to-have

- Far more automation and resilience
  - Make the Grid less manpower intensive
- More integration between workload management and data placement
- Better control of upgrades (OS, MW)
  - Or better transparent integration of different OS/MW
- Integration of the network as an active, provisionable resource
  - "Close" storage element, file replication / caching vs remote access
- Better monitoring
  - Or perhaps simply more coherent monitoring...

Things I did not talk about

- Virtuality
  - A solution in search of a problem?
  - Safety from the "porting hell" or temporary patch for unportable code?
  - It will probably become "commodity", maximally successful when nobody will talk about it
- Parallel processing
  - proof is a good workhorse for a small and well behaved community
  - Integration of Proof and Grid (POD) seems promising
  - A more "general" approach would be needed, but can there be one in parallel processing?
- Impact of multicore / GPUs
  - Back to the future… remember the vectorisation and the arithmetic coprocessor?

Experience is what is left when you have done all the possible mistakes in a restricted domain.

We certainly have worked hard to gain experience...


"That's
all
folks!"