Processing Tevatron Data Around the World: *Lessons Learned*

Adam Lyon (lyon@fnal.gov)
Fermilab Scientific Computing Division
Data Handling Group Leader
DØ and Muon g-2 Experiments

BES Detector Workshop
August 2, 2012

U.S. DEPARTMENT OF ENERGY

Fermilab

What's this about?

The Tevatron experiments have been processing enormous amounts of data world wide for over 12 years

There are many lessons we've learned

The most important: The difficulties of data management are easy to underestimate; they still surprise us!

Outline of this talk

Some context

Requirements

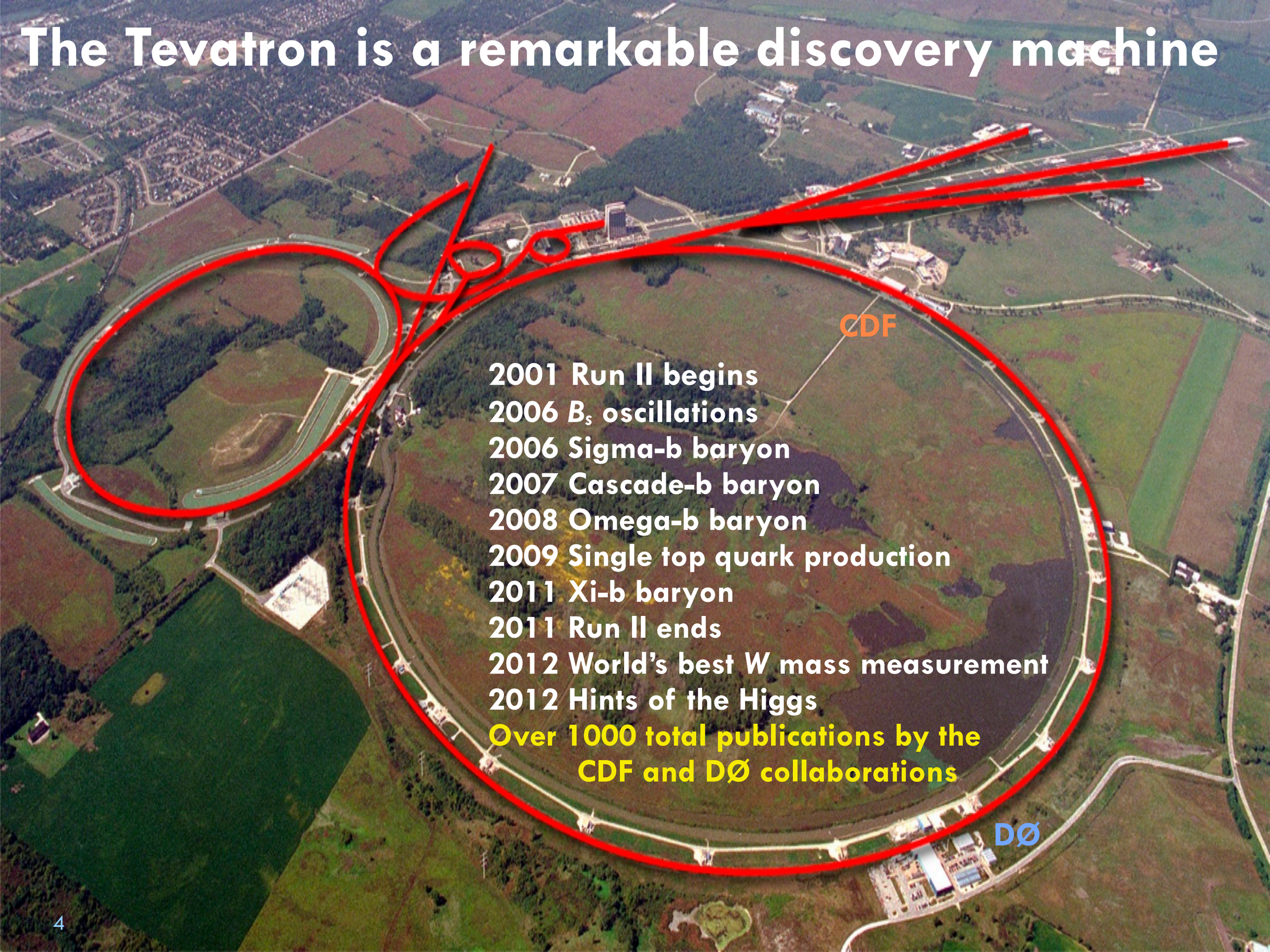
Implementation

Evolution

Future

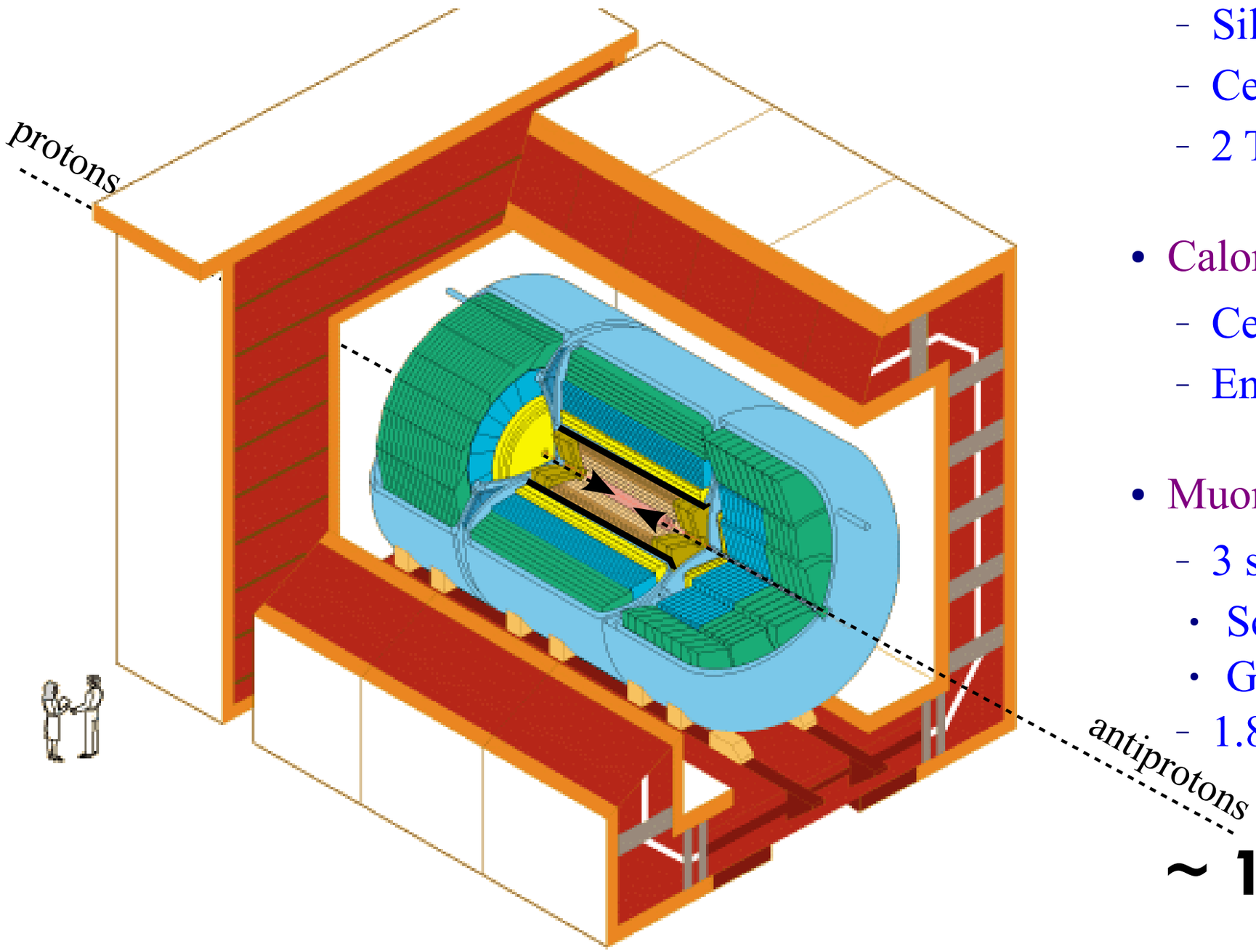
Reveal lessons learned along the way

The Tevatron is a remarkable discovery machine



2001 Run II begins
2006 B_s oscillations
2006 Sigma-b baryon
2007 Cascade-b baryon
2008 Omega-b baryon
2009 Single top quark production
2011 Xi-b baryon
2011 Run II ends
2012 World's best *W* mass measurement
2012 Hints of the Higgs
Over 1000 total publications by the CDF and DØ collaborations

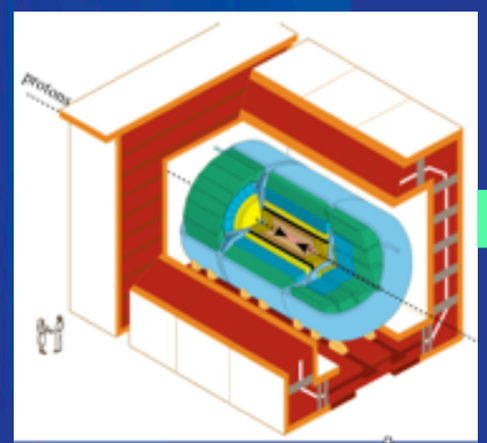
DØ is one of two large detectors at the Tevatron



- Central Tracking System
  - Silicon Micro-strip Tracker
  - Central Fiber Tracker
  - 2 T Solenoid Magnet
- Calorimeters
  - Central Calorimeter (CC)
  - End Calorimeters (EC)
- Muon System
  - 3 sets of detectors
    - Scintillating tiles
    - Gas Drift Tubes
  - 1.8 T Toroid Magnets

~ 1M channels

The processing steps from detector to analysis



Detector

raw

Tape storage

raw

thumbnail (tmb)

tmb

tmb

tmb

root-tree

root-tree

root-tree

Reconstruction

Correct & Skim

Make CAF Trees

Select data

Analysis

Plots, Results, Thesis, Paper, Trip to Sweden

Other work steps

<u>Reprocessing</u> – go back to raw data and reprocess again with new reconstruction software

<u>Monte Carlo production</u> – Produce simulated events and overlay real detector noise

DØ typically does reconstruction and analysis on Fermilab resources and Monte Carlo production offsite

Big Reprocessings done offsite

Behind the scenes of a processing step

Tape storage

Disk

Compute farm

Move to disk cache

Move to worker node

Hold small output files on disk

Process data

Transfer for merge

Merge small files

Store big files back to tape

Requirements of a data management system

Catalog data files with metadata

Query and group files into datasets

Maintain file status

Track locations of files (tape & disk)

Manage disk cache

Reliably transfer files from disk to tape and disk to disk

Store files onto tape

Track file consumption

Implementation: SAM

Sequential Access to data via Metadata

A comprehensive data management system

Started as a CD project for DØ in 1998 and continues today

Lots of developers:
2 database, 4 original DØ developers, +3 when added grid, +3 when added CDF … now down to ~1.5

Shifters from the experiment for operations

Code in C++ and Python

Close integration with experiment framework C++ code

One code base for everyone <-- has worked out well

Place focus on metadata

De-emphasize the file name

Store metadata: Data tier, skim name, application name/ version, monte carlo production details, data collection details, dates, # of events

Lesson learned: It's hard to keep metadata out of the filename

Create dataset definition with a metadata query

Datasets are dynamic (grow with new files, shrink when files go "bad")

A project is a set of jobs processing files in a dataset

The snapshot is the list of files when the project starts and is static

File status is extremely important

Availability status: Bad means that file is not available anywhere (not cached and tape is out of service). Automated

Content status: Bad means that the contents of the file is, for some reason, invalid (usually pilot error when configuring an application). Completely manual

Lesson learned: The two part status has been extremely successful

Never "delete" files from the system

Lesson learned: Being unable to recycle file names was a hinderance. Added file "retirement" to cope

Throttling is a crucial component

In the early days, systems were fragile. Too many users requesting files caused big problems

SAM introduced many throttles to file access:
- # of files a node can have requested
- # of files a project can have requested
- Careful pre-staging of files ("excess satisfaction")

Lesson learned: Integrating throttling with efficient tape operations is difficult; some inefficiency will remain

Lesson learned: Even as systems become robust, fragility can remain and throttling remains necessary

Data integrity is crucial

Tape system generates a CRC when file is stored. CRC is recorded in SAM database

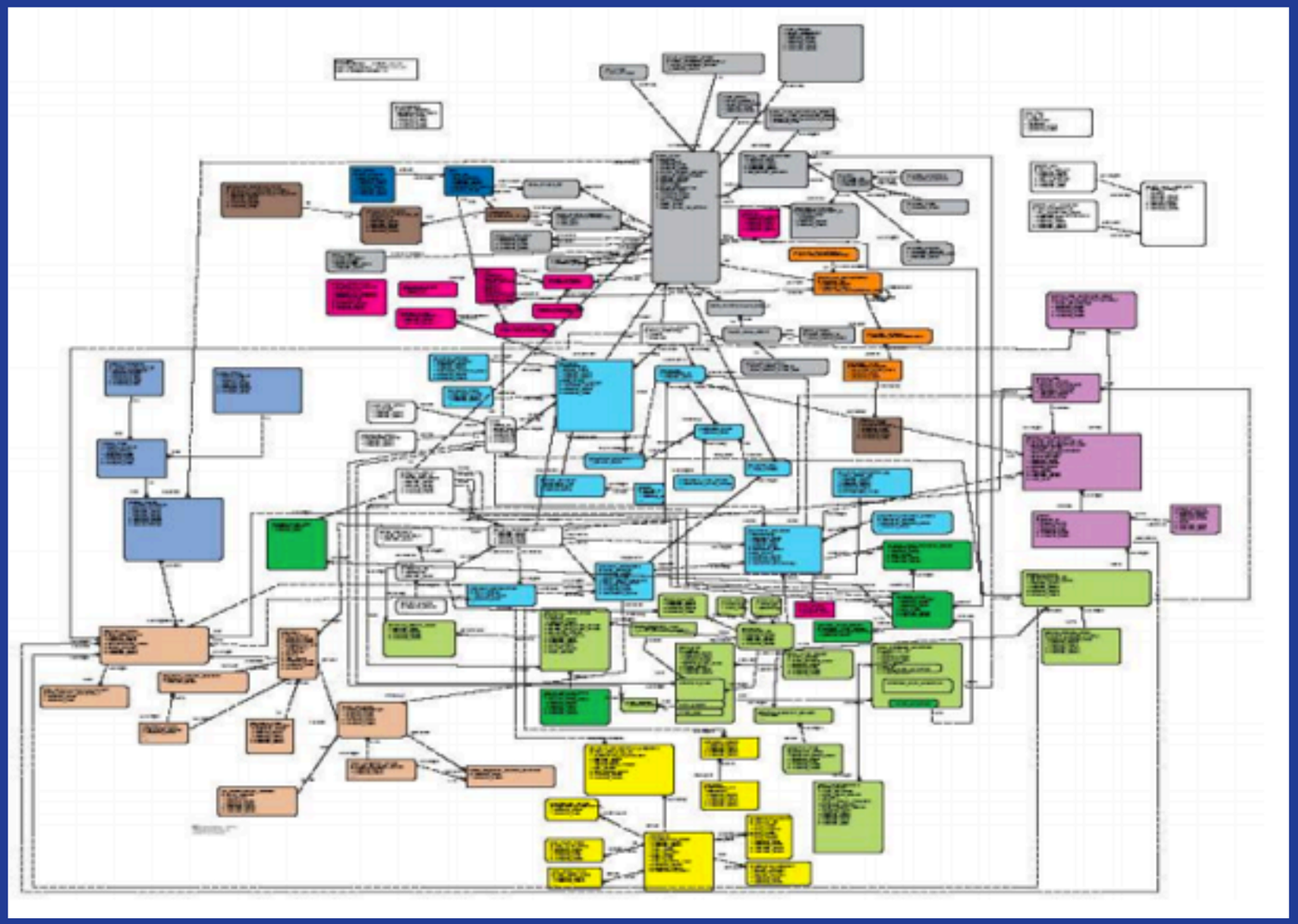
SAM checks the CRC after every file transfer

We catch a few corrupted transfers every year

Lesson learned: CRC checks add time, but checking is absolutely critical for data integrity

The "SAM" Project implements the data management requirements

Oracle database for metadata catalog, replica catalog, cache status, "project" tracking, "process" tracking, consumption status, configuration



Lesson learned: Exclusive use of one database has plusses (query tuning, support) and minuses (hard to port)

A short glossary

Job - running an application on a batch system in a batch slot

SAM Project - A group of possibly parallel jobs requesting files from the same snapshot

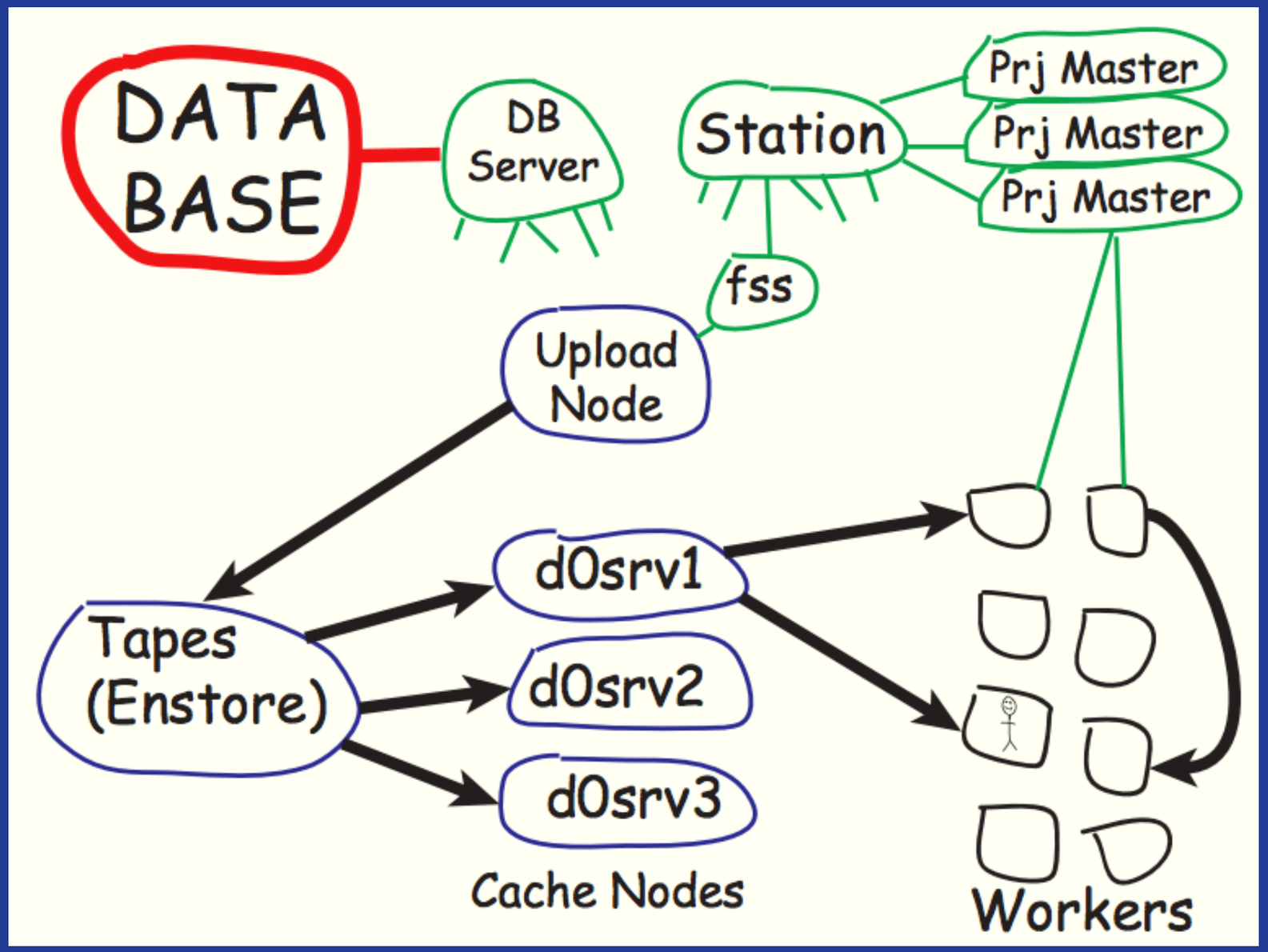
SAM Project Master - A service that manages file delivery for a project

SAM Process - corresponds to an individual job receiving and processing files

SAM Station - A service that manages all cache and projects (an experiment can have many stations)

Lesson learned: Grouping jobs into projects has been very successful

SAM System cartoon



Interprocess communication all with CORBA (was the best choice in 1999)

Lesson learned: Protocol choices are hard to change later, but possible to "wrap"

Data Handling specs for DØ

Plan from 1999

Entire dataset:	**0.5 - 1 PB**
Tapes:	**50 GB, 6 MB/s**
Cache size:	**20 TB**
# files in catl:	**1 million**
# events in catl:	**few billion**

Data Handling specs for DØ

	Plan from 1999	Actual in 2012
Entire dataset:	0.5 - 1 PB	8.2 PB
Tapes:	50 GB, 6 MB/s	800 GB, 120 MB/s
Cache size:	20 TB	760 TB
# files in catl:	1 million	140 million
# events in catl:	few billion	abandoned (too big)

Lesson learned: Think big and prepare for more!
Scalability is crucial!

Be ready for eco-system changes

Introduction of "Big Cache"

Introduction of "file spreading"

Lesson learned: Monitoring is crucial (and hard to get right). Be ready for conditions to change over time, necessitating action

Lesson learned: Manipulating the cache by hand (e.g. pinning files) leads to headaches; leave it automated

Lesson learned: Automate everything!

The "SAM" Project Events

Started as a Fermilab Computing Division project for DØ in 1998

Introduced to CDF in 2003

DØ Reprocessing in 2005 (first large-scale use of the "Grid") 1.5B events, 82K files, ~82 TB

DØ Reprocessing in 2007 (use of Open Science Grid and LHC Computing Grid) 1.2B events

Monte Carlo production on OSG and LCG

SAM makes a Grid for offsite processing (SAMGrid)

Run offsite with a consistent and central interface -- sounds like the Grid, but before the Grid was real

We wrote our own middleware and "batch adapters"

circa 2004

<u>Very</u> painful to operate and maintain. Software very fragile. Needed students to help. We had to debug site issues.

Lesson learned: Be pioneers only when you have no other options.

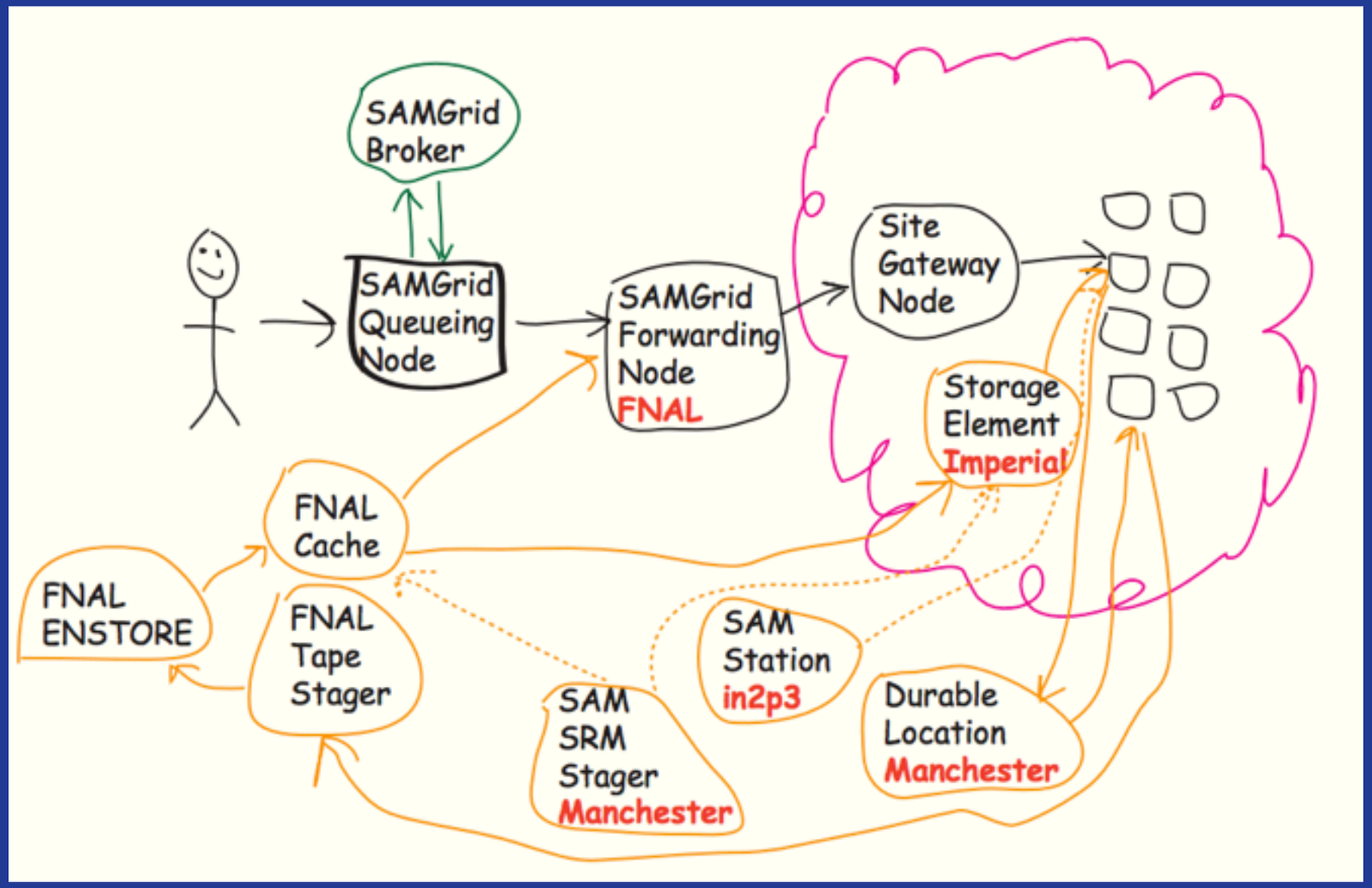
Adopted standard Grid tools

We still have a central interface, but now to OSG and LCG. Adapted SAM to use Grid Storage.

We use GlideinWMS

OSG and LCG oversee site maintenance

Use polling to overcome worker node access restrictions



Recovering when things go wrong

Jobs accumulate output on the worker node
(usually an output file is the result of reading several input files)

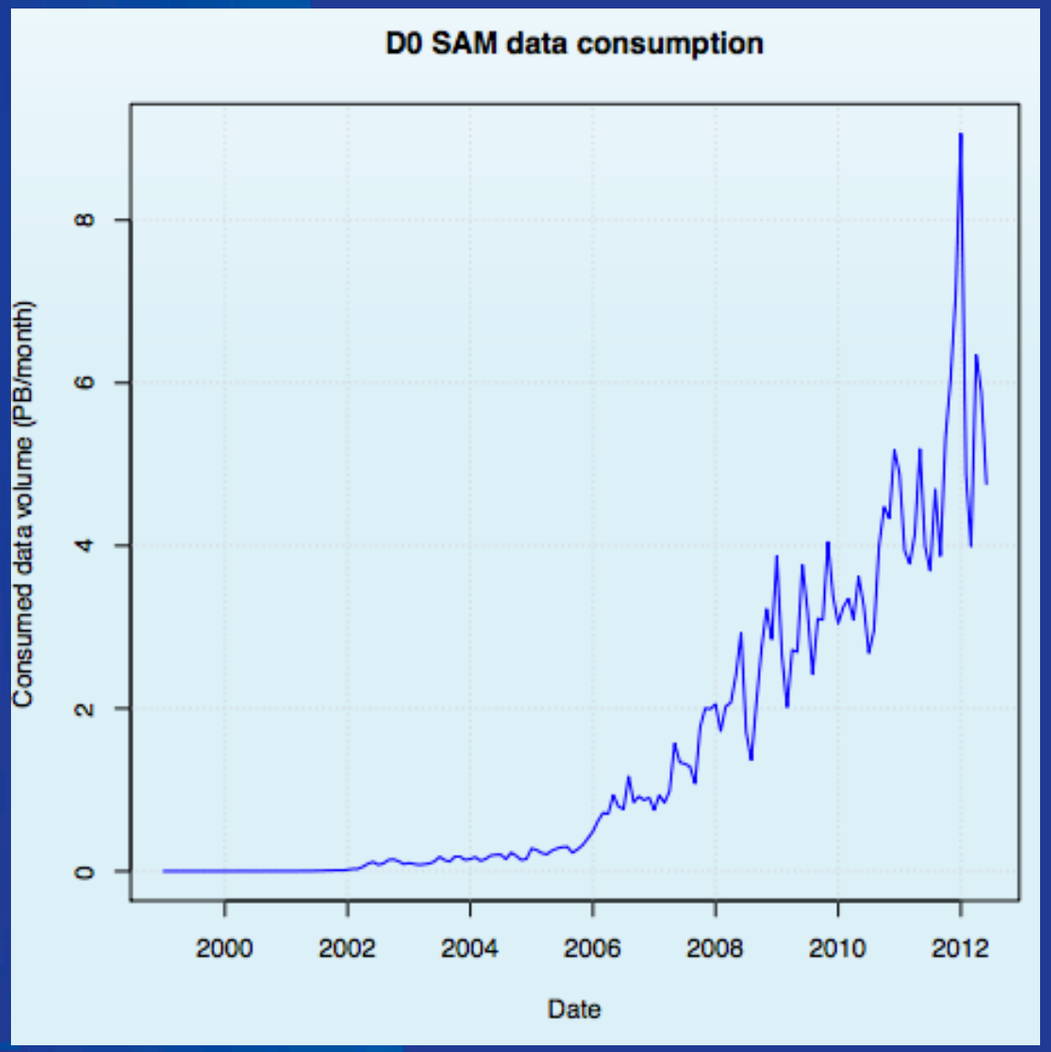
What happens if some jobs are successful but others fail (crashes, dies, lost output)? What happens if a file can't be delivered?

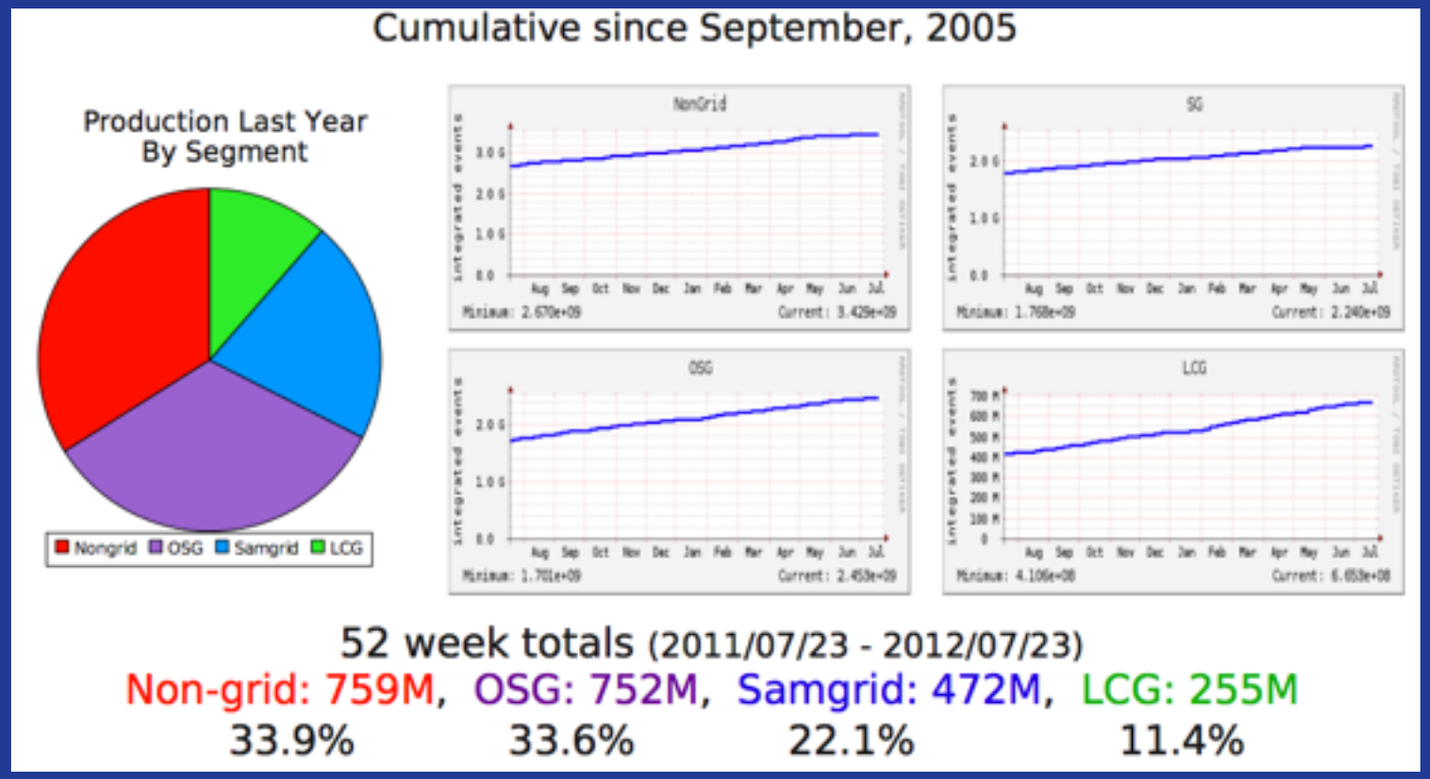
SAM keeps track of all unfulfilled transfer requests (unconsumed files). Can run a recovery project to try these files again

SAM also tracks if a job reports successful completion. For those that don't, users can run a recovery project to re-process the input files corresponding to lost output

Lesson learned: Users appreciate these features. Important to think about recovery when failures occur

Some statistics





All time total of ~ 250 TB consumed

Monte Carlo Production
8.75B events produced total

~40% on OSG+LCG

We still have a powerful old-style SAMGrid site in the Czech Republic

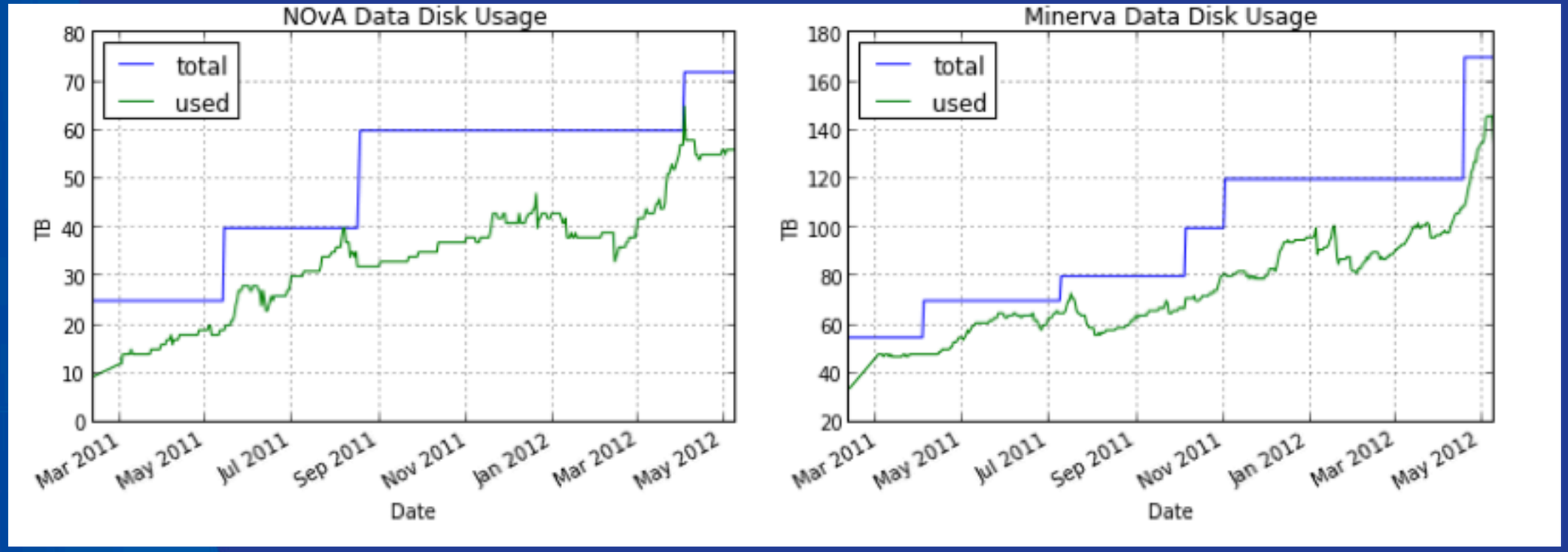
France hosts a big compute farm for special MC requests (non-grid)

SAM's future

Fermilab is transitioning to smaller specialized experiments at the *Intensity Frontier (IF)*

New experiments are 100 TB - PB scale data production

Remember underestimating data management? We see this from the new experiments ("We'll just put everything on disk" - having a big SAN yields false security).



Introducing SAM, but with improvements to make it more palatable

Data Handling for IF experiments

We soul searched whether to continue with SAM, adopt something else, or start writing something new

<u>Decided</u> that there's no other product like SAM (LHC data handling systems are not generic enough and use a different data handling philosophy)

<u>Decided</u> it would be too much work to embark on something new, especially since SAM is working extremely well

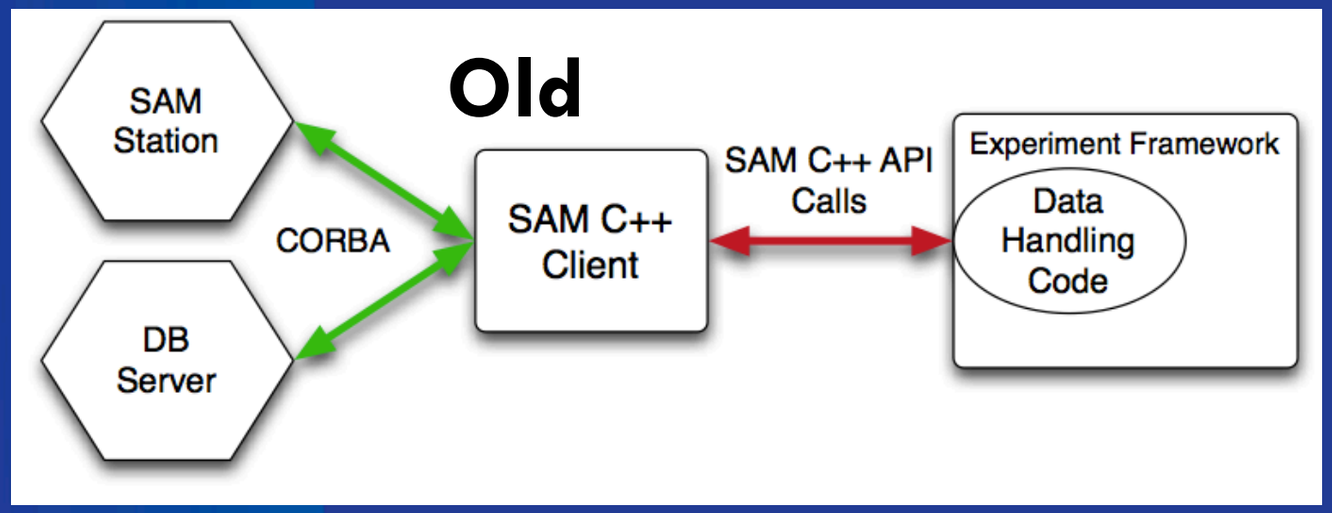
<u>Decided</u> to leave the SAM internals as they are, but wrap over the warts

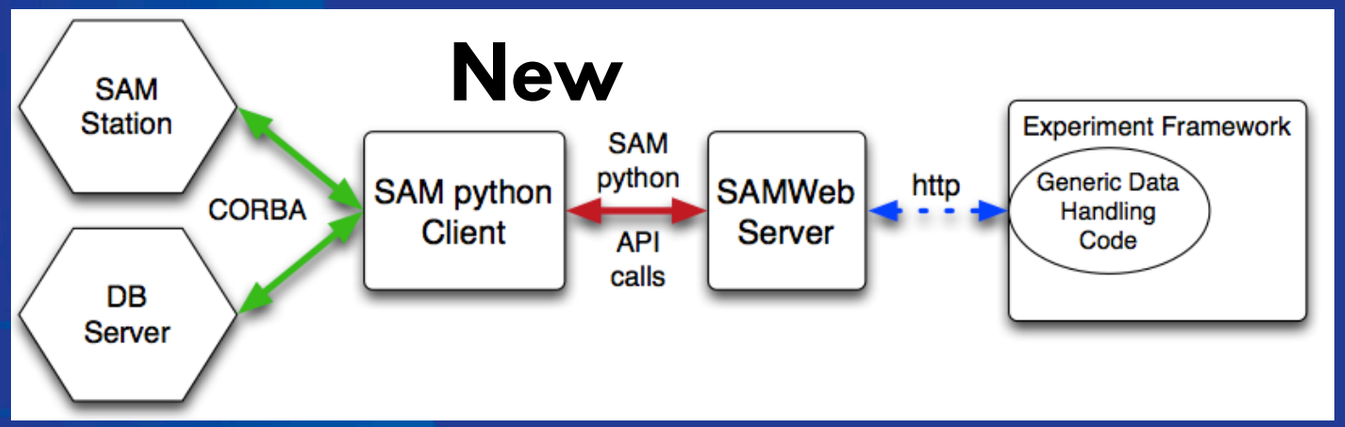
Two big warts:

1) Integration with experiment frameworks
2) User interface, especially for file transfers to your desktop

SAMWeb & SAMfs

Dramatically improve SAM + Experiment Framework Integration and our deployment burden





Make SAM behave like a file system with FUSE

- Navigate metadata as directories

```
ls -F /samfs/run
ERROR-supply-a-run-number-like-200000@

> ls -F /samfs/run/268211  # will list sub-runs
111/   112/   113/

> ls -F /samfs/run/268211/111 # will list data-tiers
raw/   reconstructed/

> ls -F /samfs/run/268211/111/raw  # will list files
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root*
raw_nu_268211_111_3.root*
```

- Navigate pre-defined datasets

```
> ls -F /samfs/dataset
ERROR-supply-a-user-name@

> ls -F /samfs/dataset/lyon  # list my datasets
my-special-run-15593/  oscillated-nu-bars-5/
bad-tracking-examples/

> ls -F /samfs/dataset/lyon/my-special-run-15593/
reconstructed_nubar_15593_112_4.root*
reconstructed_nubar_15593_112_5.root*
```

- Load file to cache with "get"

```
> ls -F /samfs/run/268211/111/raw
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root*
raw_nu_268211_111_3.root*

> ls -F /samfs/get/run/268211/111/raw # prohibit ls with get
ERROR-trying-to-list-directory-with-a-GET-path@

> cat /samfs/get/run/268211/111/raw/raw_nu_268211_111_2.root
# binary output here

> root  # Open file directly in root
TFile* f = new TFile("/samfs/get/run/268211/111/raw
268211_111_2.root")
# ... back to shell

> ls -F /samfs/run/268211/111/raw  # @ == soft link == cached
raw_nu_268211_111_1.root* raw_nu_268211_111_2.root@
raw_nu_268211_111_3.root*
```

Lesson we hope to learn: Improve by wrapping instead of by scrapping (we hope this doesn't make maintenance difficult, we'll see)

Summary and sales pitch

SAM is a very successful comprehensive data management system

But it hasn't been easy - many lessons learned

Many lessons applicable to others

Fermilab SCD is interested exporting SAM to other labs & experiments

BACKUPS

SAMGrid Workflow Management

"D0RunJob" is our home written workflow manager

Not very exciting – knows how to generate Monte Carlo data, usually 200 events per job

Scripts for generation, geant, simulation, reconstruction, and Root-tree making

Has hooks for declaring meta-data to SAM

Requests noise overlay files from SAM

Stores output files on "durable locations" for merging

Tape System (Enstore)

For D0...

7 tape libraries; 70K slots

~100 LTO4 drives, 53 T10000C drives

47K LTO4 tapes, 2.5K T10000T2 tapes

Over $6M investment

Why our own tape system? In 1999, tried HPSS and had poor experience. Not clear if problems could be resolved for Run II