# Operational experiences

Castor deployment team

# Outline

- Operational model
- Castor-2 usage characteristics by the different experiments
- What works, what doesn't, what can be improved?
  - Database operations
  - Services (dlf, lsf, stager, rmmaster)
  - Grid related complications

    *just to give you a flavour ☺*

# Current operational model

- Many distinct but similar Remedy/GGUS flows…
  - Castor.support
  - Castor2.support
  - WAN data operations
  - Data service interventions
- … plus additional requests/alarms/information by e-mail…
  - Castor.operations, castor-deployment
  - Experiment and Service Challenge mailinglists
  - Private e-mails…
- … but always the same group of people ☺
  - currently 4 service managers (but <4 FTE's) + 2 core developers
- Drawbacks:
  - As things go wrong on many different levels *(hardware failures, network problems, system crashes, configuration problems)*, ***debugging*** is not trivial. Neither is ***problem reporting***…
  - Often many-to-many reporting leads to silence…
  - SLA: best effort, no formal 24 x 7 coverage, no measurable QoS

# Apply **Standard** operational model

- Re-factor the support area's
  - tape-related problems will be handled by a separate flow
- Rely more on CC operators and SysAdmin team
  - **to provide 24 x 7 coverage**
  - already used for H/W and OS level problems
  - we will need to provide procedures *or fix bugs* ☺
- 3-level support structure, with escalations
  - **1st level: FIO Service Manager on Duty**
    - *one person + backup from a group of 5 Service Managers*
    - *"simple cases", responsible for **follow-up** of difficult cases*
    - *As for lxplus/lxbatch*
  - **2nd level: Castor Service Manager**
    - *Castor Operations team*
  - **3rd level: Developers**
    - *Castor, SRM, Gridftp, rootd, FTS, LFC, …*

# Castor usage

- Group activities in diskpools
  - T0: one user, large files, limited number of streams
  - Durable data: `write once, read many`
    *need to manage additional SRM endpoints*
  - User analysis:
    - many users
    - many small files
      - Atlas, CMS: average filesize <100 MB
      - 40K files per server!
      - Overhead of tapewriting
    - many (looooong lived!) open filehandles
      *exception: Alice copy datafiles before opening them*
  - There will be more…
- Very different access patterns!
- Castor is handling these access patterns in production

# Workarounds

- We have workarounds for various bugs
  - cleanup procedures in databases
  - Lemon actuators, cron jobs
  - throw hardware at the problem
- Good
  - service continues to function
  - operational flexibility, big improvement over Castor-1!
- Bad
  - bug fixes take time, as does their rollout
  - workarounds tend to lower priority of bug fixes
  - “temporary fixes” become harder to remove as time goes by…

# Stager database operations

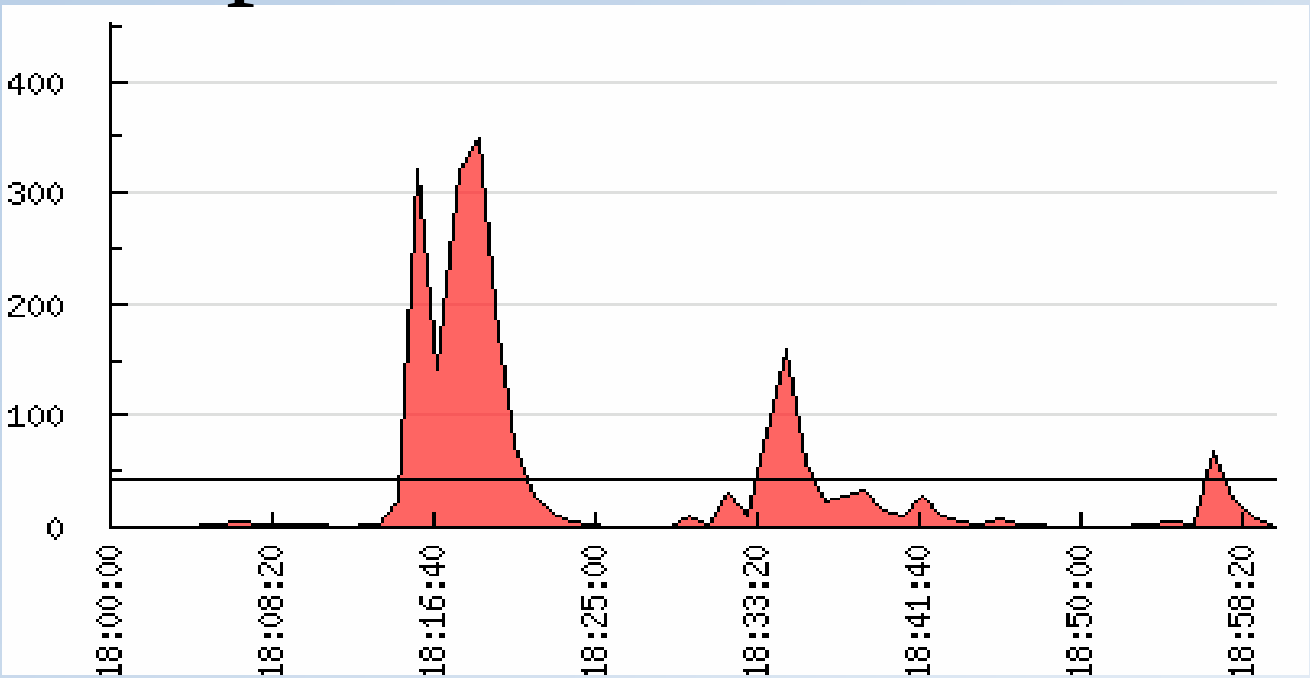
## Stuck DISK2DISKCOPY, repair

```
SQL> select concat(diskserver.name,concat(':',concat(filesystem.mountpoint,diskcopy.path))) \
       from diskserver,filesystem,diskcopy where diskserver.id=filesystem.diskserver and filesystem.id=diskcopy.filesystem \
       and diskcopy.status=0 and diskcopy.castorfile in (select castorfile from diskcopy where status=1);

CONCAT(DISKSERVER.NAME,CONCAT(':',CONCAT(FILESYSTEM.MOUNTPOINT,DISKCOPY.PATH)))
--------------------------------------------------------------------------------
lxfsrk4108:/srv/castor/03/63/68476963@cnsgrid.cern.ch.255418639
lxfsrk5901:/srv/castor/01/52/69424452@cnsgrid.cern.ch.268274193
lxfsrk5902:/srv/castor/03/73/69424473@cnsgrid.cern.ch.268274300
lxfsrk3902:/srv/castor/02/87/69424487@cnsgrid.cern.ch.268274365

  SQL> select id from filesystem where mountpoint='/srv/castor/01/' and diskserver in \
         (select id from diskserver where name='lxfsrk4108');

         ID
  ----------
  54762992

  SQL> update diskcopy set filesystem=54762992 where id=268274193;

  1 row updated.

  SQL> commit;

  Commit complete.
```

# LSF issues

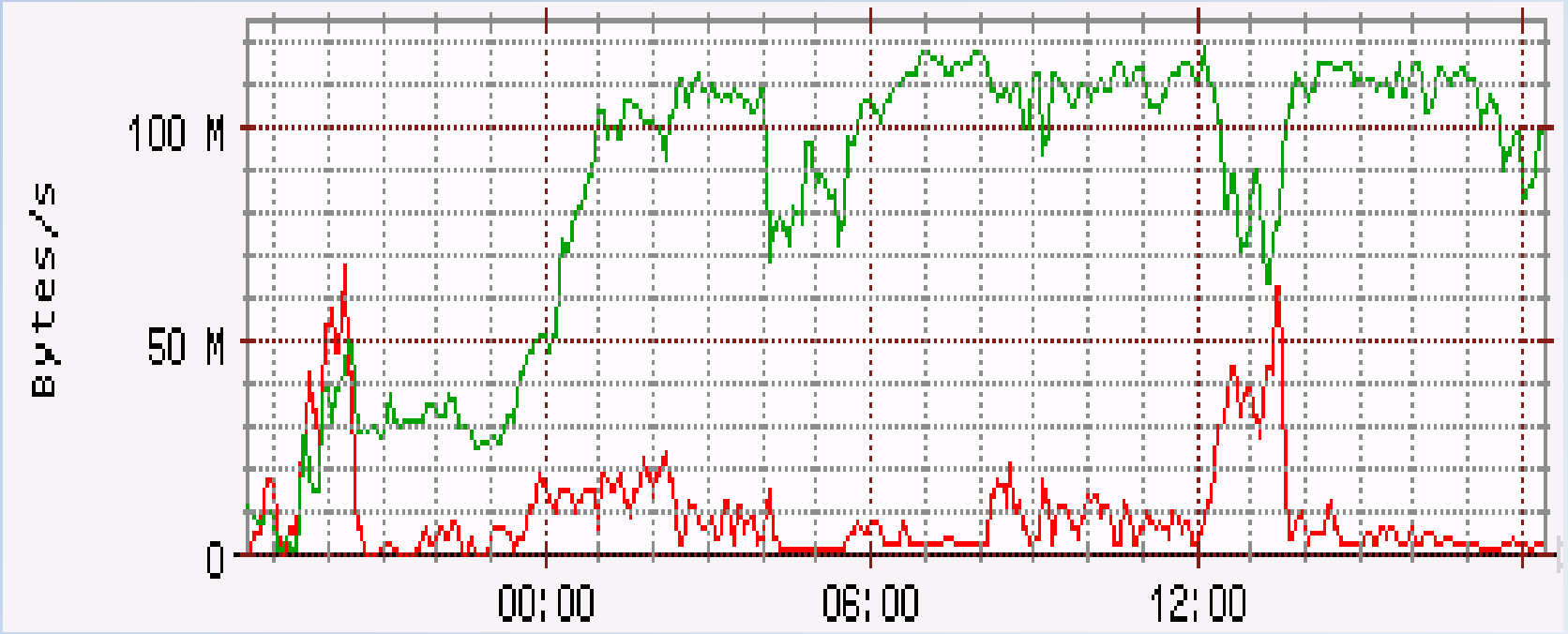
- we suffer (infrequently…) from LSF meltdowns
  - very tight scheduling required (mbatchd is basically
  - overhead in the CASTOR LSF plugin
- Castor schedules LSF jobs to the diskservers for fileaccess
  - scheduling is complicated *(over-complicated?)*
  - scheduling takes time
  - each running job involves 5 processes
  - non-negligible overhead
- We observe load spikes on diskservers…

# Load spikes on diskservers




- Not fully understood yet… *but clearly related to bursts of user activity*
- To be studied:
  - Minimize footprint of individual job *(reduce number of forked processes)*
- Workarounds:
  - Limit number of jobs per server *(inefficient use of disk space, need more servers…)*
  - Temporarily disable server *(automatic, based on Lemon alarms)*

# …but the system performs!




**Diskserver from CMS user pool running at NIC speed**

# Access protocols

- all Castor-2 diskservers have 3 access protocols configured
- rfio
  - dangling connections when clients die *(fix to be deployed)*
- rootd
  - no big problems
  - good response from developers
  - already worked on Castor-1
  - does not log to DLF
- gridftp
  - all nodes have (managed!) grid certificates, monitoring, etc
  - well-behaved *(CPU-hungry, but the diskservers have spare cycles)*
  - does not log to DLF
  - expertise will be needed
    - *gridftp not 'native' protocol yet*
    - *cope with evolving environment (SLC4, 64bit, VDT)*
    - *castor-gridftp v2 (when required)*

# DLF

- Distributed Logging Facility: logs events from instrumented components into Oracle
- Queries take too long *(lack of partitioning)*
  *fixed, but not yet deployed*
- Database full? Drop the tables!
- stager needs dlfserver up and running *(bug!)*
  - **Stager leaks memory** *cron job to restart it every 3 hours*
  - **dlfserver dies** *Lemon actuator restarts it*

*But to stop the service for a DB upgrade, we need to stop the cron job and deactivate the actuator first!*

# Configuration consistency

- Castor components are often individually configured, and store the information independently
  - LSF
  - Stager database
  - rmmaster
- Inconsistencies lead to confusion *(at best…),* inefficiencies *(unused diskservers),* problems…
- We derive configurations as much as we can from our Quattor Configuration DataBase
- Workaround: run regular consistency checks, correct configurations by hand

# rmmaster…

- **Is not stateless at all!!!**
- Looses state when daemon is restarted
- Administrator must store and restore by hand when upgrading
- IP renumbering of diskservers are not properly handled
- rmmaster writes filesystems/diskserver information into stager DB, once and for all…

- This component needs a re-design!

# Castor and the grid

- we run multiple SRM endpoints
  - **`castorgrid.cern.ch`** (aka **`castorsrm.cern.ch`**)
  - **`srm.cern.ch`** (aka **`castorgridsc.cern.ch`**)
  - soon: 4 endpoints to “durable” experiment data
  - later: SRM v2.1
- **`castorgrid.cern.ch`** and **`srm.cern.ch`** provide similar functionality…
  - **`castorgrid.cern.ch`** proxies data, necessary for access Castor-1 stagers
  - but with different connectivity *should no longer be the case*
- ...but software packages and their configurations are very different
  - historical reasons…
  - leading to different grid-mapfiles
  - rationalizing this is slow, painful, and necessary
- We have very little monitoring of SRM and GridFTP problems

# Conclusions

- Castor instances are large *(LHC expts: 1.7M files, 463 TB)* and complex *(very different filesizes/access patterns)*
  - **And they are up-and-running!**
- To run the Castor services requires workarounds
  - We have workarounds for many problems (good!)
  - We have workarounds for many problems (bad!)
- It takes us too long to get bugs fixed and deployed *(and bugs are exported to external institutes…)*
- To be able to scale the operability
  - **We are streamlining our support structure**
  - **We need to replace workarounds by fixes!**