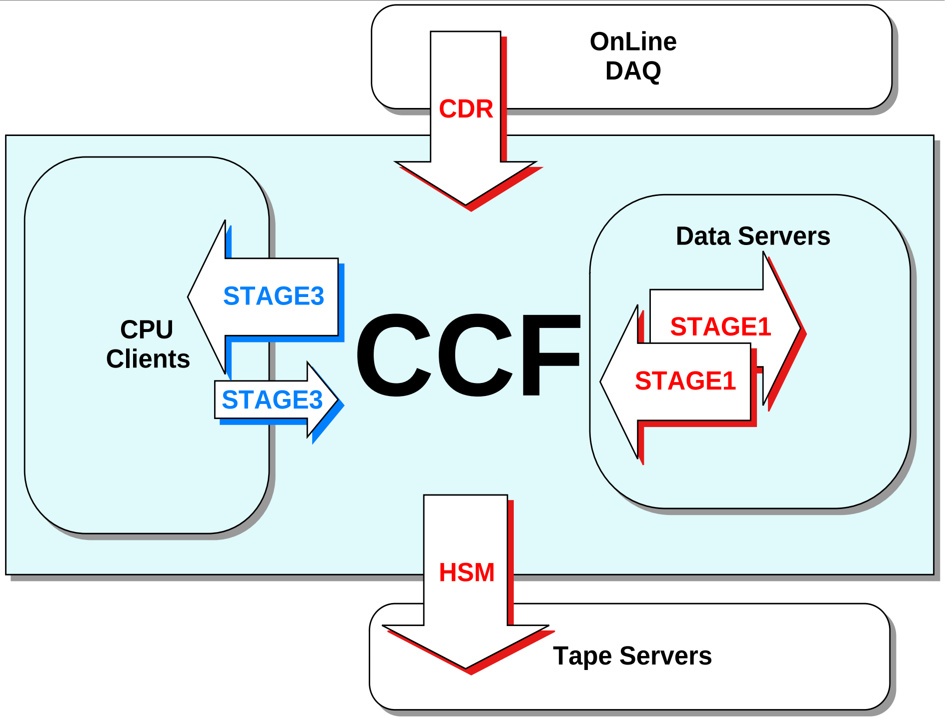
COMPASS Computing Farm Project

Index

- The CCF project
- The CCF software
- DAQ mode at 35 MB/s tests
- Condition Data Bases studies
- Compass pilot run (Summer 2000)
- Conclusions

CCF project



- 35 MB/s (300 TB/y) of data from the experiment to the CCF for recording (Central Data Recording: at CERN: NA48, NA45, TestBeam) and reconstruction.
- 20,000 CU (~2k SpecInt95, ~ 100 PCs computing power) needed to reconstruct the events at the same speed of the DAQ: quasi-online processing.
- PC hardware to provide the bulk of the computing power (1997: WNT, now Linux). Disk servers: SUN and DEC servers have been tested. Now all data servers run Linux. Network technology is Ethernet (Fast/Gigabit).
- C++ reconstruction program (Coral framework: http://coral.cern.ch)
- C++ objects stored as such in a hierarchical way (Objectivity/DB). New technology to be understood
- Hierarchical Storage Manager (HSM) layer (1998-99:HPSS, now: CASTOR)
- Data flow model: data set driven, every run -> 10-20 streams of parallel transfers -> parallel population of DBs and reading (for reconstruction)

CCF software

The CCF control software (written in Perl) should:

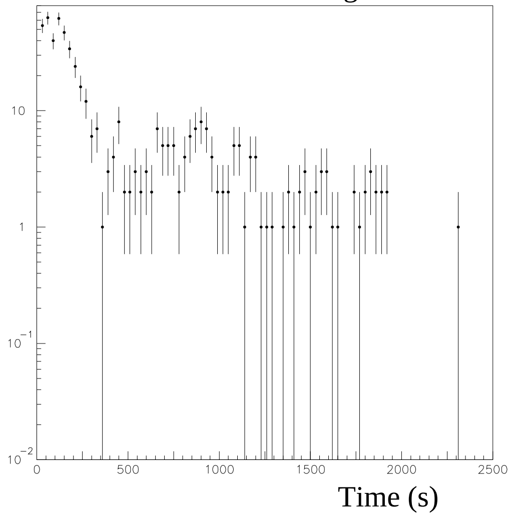
- Operate and monitor the CCF (from the hardware selection to the error recovery)
- Steer the reconstruction programs
- Interact with the data storage technologies: Objectivity/DB and the HSM (CASTOR)

Data Servers:
Real Time: AMS status and CPU (each thread)
Real Time: Disk status for load balancing
Log files: CDR (Transfer performances)
Log files: Stage1 (DB input performances)

Lock Server:
Real time: LS status and CPU
Real time: number of locks (r/u/mrow)
Real time: histogram transaction duration
Real time: alarm on too long transaction
client, server, transaction #,
PID#, DB path...

CPU clients:
Log files: reconstruction output

Transaction duration histogram



Operations are “run based” (“file based”)

- job-based output (log files)

“Real-time” tools complement this picture

- Run on Linux and Solaris
- simple UDP communication system
- Use Objy tools (e.g. `oolockmon`), kernel informations (e.g. CPU usage) and allow correlation (via time)

CCF administration

The deployment of a system with more than 100 major components, require automatic procedure and a deep understanding of the dependencies among all the building blocks (hard and software).

The farm is managed by IT/PDP within the existing control infrastructure (SURE) implementing the missing functionalities, wherever possible; independent daemons provided by the CCF software otherwise. Standard procedures to install consistently a large number of machines have been set-up (e.g. CERN SUE and ASIS mechanisms under CCF control); in production, all software is installed on each node (cvs check and parallel build on all nodes or simply rcp; no critical applications in AFS).

Quality assurance and benchmarking:

- Pragmatic approach: e.g. install Objectivity/DB and run tests on all nodes to test local and network access; measure the disk speed on a server with combinations of read and write streams; measure the network connectivity among different hosts...
- Build official software on all nodes.

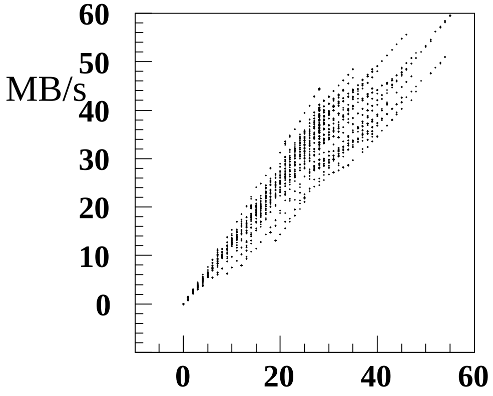
35 MB/s test in DAQ mode

- Data server stage: 11 PCs with ~ 1.5 TB disk space
- Data server hardware: dual PII PCs with Fast ethernet and UW SCSI tray. Each data server holds ~100-200 GB of JBOD.
- About 35MB/s mock data sent from COF to the CCF data servers (CDR stage) in ~30 concurrent parallel streams using the RFIO software
- CCF data servers convert the data in Objectivity/DB data base (Stage1). This stage is performed locally (using the Data servers CPU: no AMS involved). This is a choice to leave the AMS free to serve the CPU clients
- Data bases sent to some machines simulating the tape servers (HSM); the original data deleted when the corresponding events finished HSM stage
- The data bases are erased from disk (and the corresponding federation database updates are performed)
- As a result of these tests, we (plan to) have about 20 PCs to be the data servers sharing a total of few TB of disk

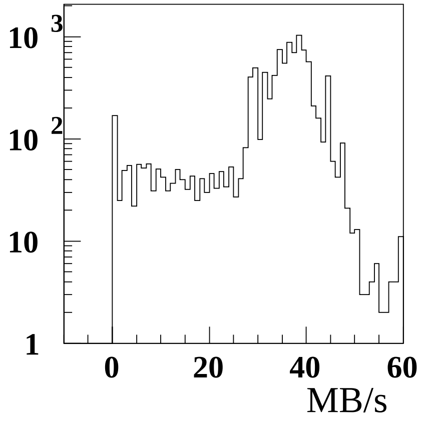
35 MB/s results

- The CDR system during 6 hours is shown. The CDR has a negligible idle time

Transfer rate vs no. of streams

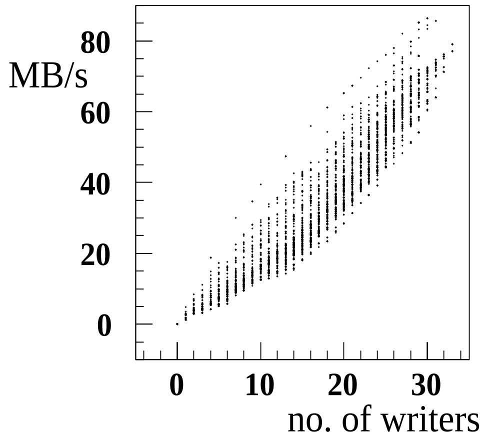


Transfer rate

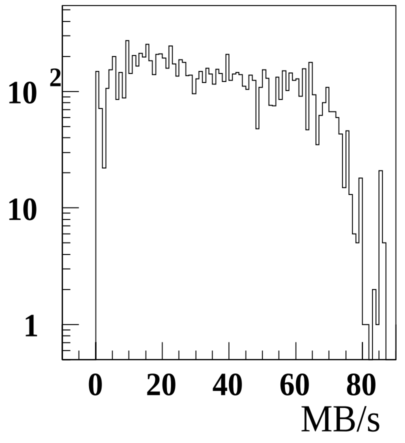


- The stage1 system: 17 Objectivity/DB clients (mean value) write concurrently into the same federation. Typical rate of a writer (local I/O) is 2.4 MB/s (up to 4 streams per PC). Idle time < 2%.

DB speed vs no. of processed (Objjy writers)

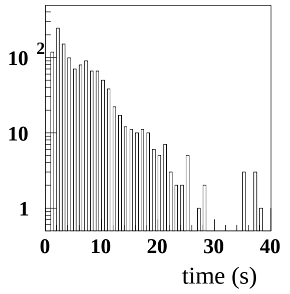


DB speed (Objy writers)

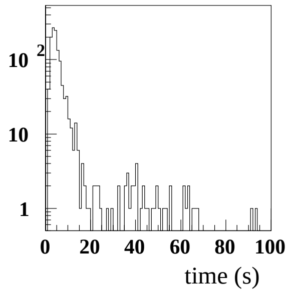


- Monitor DBs in realistic conditions

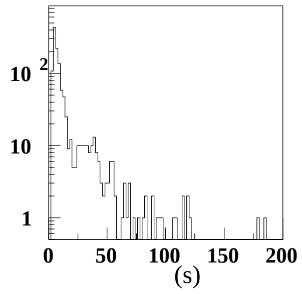
Data base file creation



Containers creation

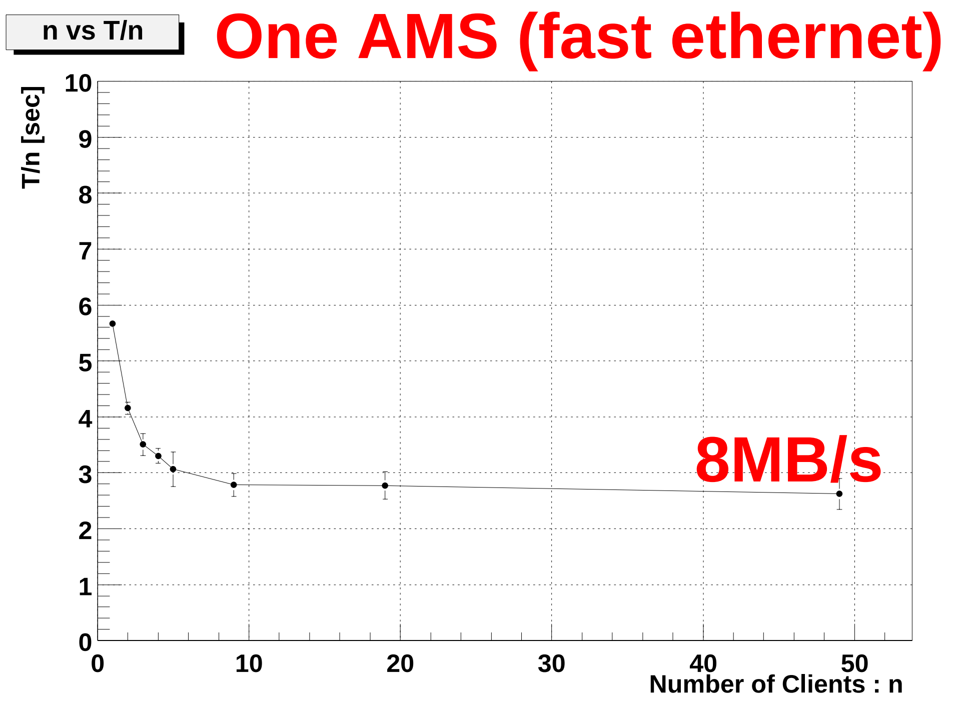


Commit time



Condition Data Base studies

- Use CERN port of the CDB (originated by Babar; unfortunately now two independent branches)
- No users when we started (end 1999). Efficient bug fixes and improvements from IT/DB; now a new version is being defined (back to User Requirements).
- Study possible DB access congestion at run startup (multiple loading of conditions); cross check with monitoring tools (AMS CPU, LockServer CPU, data base locks, etc...).
- Job just started; test to be done on the CDB with real data (sizes, #versions) and with some new hardware (Linux PCs with mirrored EIDE disks and Gigabit ethernet).
- The present tests are done with a "standard"



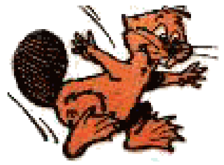
CCF data servers (SCSI disks and Fast ethernet network interface).

Compass pilot run (Summer 2000)

- Pilot run
  - muon and hadron beams up to nominal intensities
  - all detector types on the floor (although partially equipped)
  - debug the whole chain (detectors, front-end electronics, data acquisition, CCF)
  - collect data to study detector performances and tracking algorithm (CORAL)
- CCF status
  - CDR all the time active (10 weeks); all data to tape.
  - Objectivity/DB data bases created in the last weeks in parallel (also DB to tape)
  - Last week up to 100 clients able to run CORAL (event scan only) in parallel with CDR and data base population
  - Two main new ingredients (compared to previous tests): the tape system is managed by a new HSM system (CASTOR; in 1999 we used HPSS for the test beams) and the AMS on the data servers with the raw data contains the HSM interface to CASTOR (in Spring CMS used the AMS-HPSS interface).
- Hardware performances
  - In general OK
  - Some strange hang-up in data servers (understood, new kernel upgrade scheduled 2.2.12 -> 2.2.15)
  - Some scsi-disk problems: no data loss because of software architecture... but!
  - Some (3-4) clients experienced a crash

CCF and CASTOR

- CASTOR main motivations:
  - Focussed on HEP requirements
  - Short term goal: handle NA48 and COMPASS data in a fully distributed environment
  - Long term goal: handle LHC data
  - Sequential and random access should both be supported with good performance
  - Good scalability
  - Easy to administer, clone, and deploy
  - High modularity to easily replace components
  - Available on most Unix systems and WindowsNT



Cern Advanced STORage Manager

http://wwwinfo.cern.ch/pdp/castor

- CASTOR status (Phase 1)
  - Remote access to disk files
  - Disk pool management
  - Indirect access to tapes (staging)
  - Implement HSM functionalities
  - Stress tests (ALICE MDC early 2000)
  - HSM in production (COMPASS)
- Hot points (for CCF):
  - Robust and very well supported
  - Plain files and Data Base sent to CASTOR as such (entered in CASTOR namespace)
  - Objectivity/DB AMS (CERN version IT/DB) now available with the CASTOR backend (originally HPSS only)
  - Very positive experience! No data loss.

AMS-HSM interface

- Original version from Babar (Solaris, HSM=HPSS)
- CERN version developped in IT/DB group
  - Linux version (AMS running on Linux data server)
  - HSM = CASTOR
- Experience and outlook
  - HPSS->CASTOR possible!
  - Stable running possible
  - Prototype version (due to its HPSS origin, now the AMS-HSM requests data to CASTOR, makes a copy in its own managed private pool, and then it serves the data to clients: double copy)
  - Final version before 2001 running (AMS-HSM optimised to work also with CASTOR, i.e. no double copies, no private pool: the AMS-HSM should read the DB directly out of the CASTOR namespace, i.e. directly out of the CASTOR pool using RFIO)
  - Very promising (no crash during the tests due to the HSM part). Very few crashes of the AMS part (known problems, not connected to the interface, already improved against AMS 5.2.0)

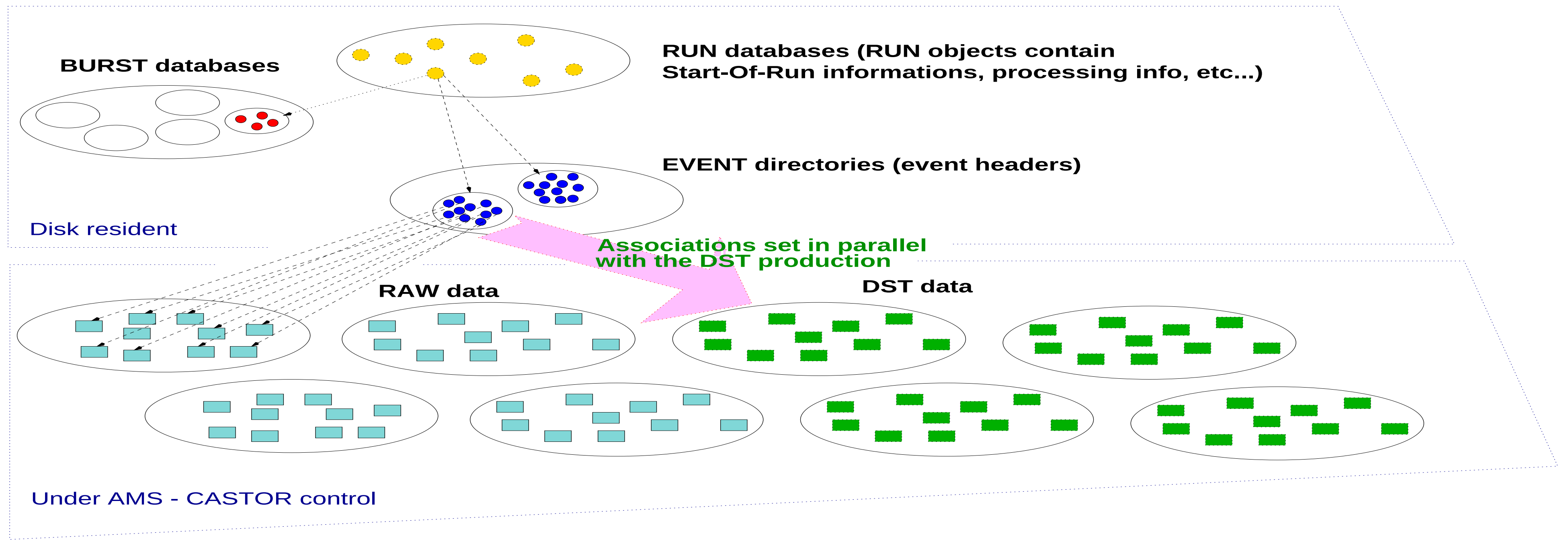
Data organisation

The data base organisation derives directly from the data structure created by the data acquisition (set of parallel files).

Frequently accessed quantities are left on disk if possible (Conditons, Run objects, etc...).

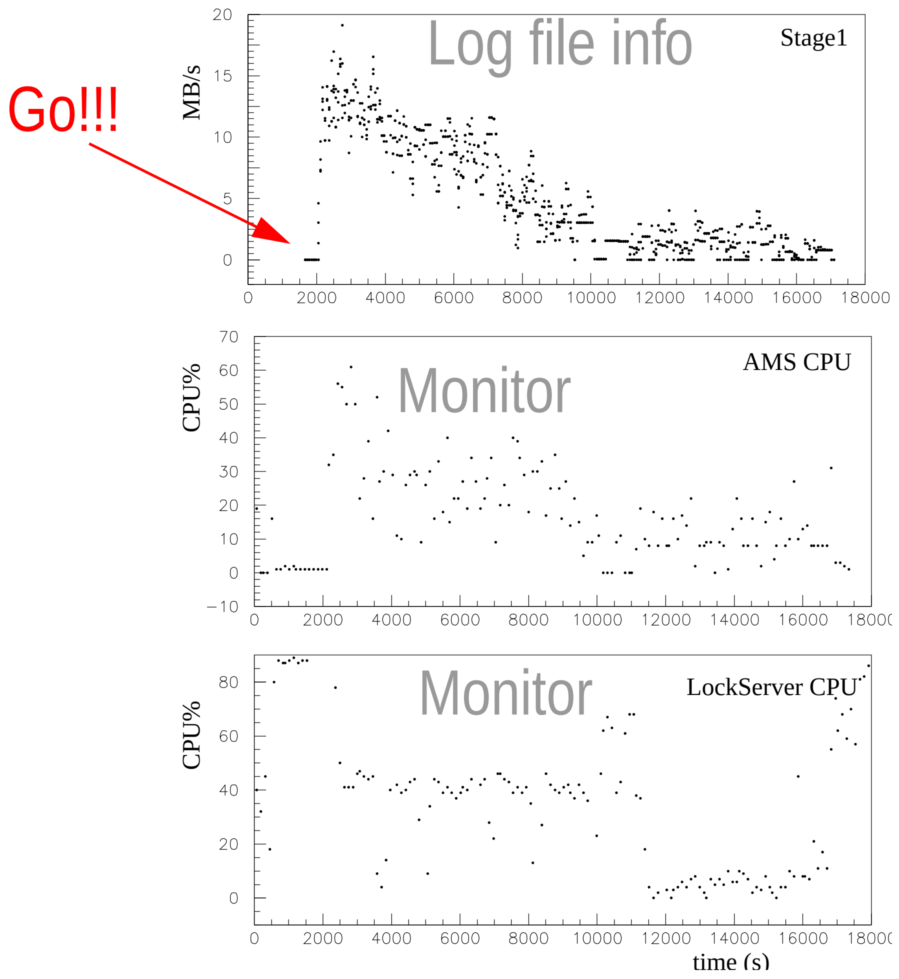
The logical structure is very light (one-one and one-to-many associations) done with Objectivity/DB.

The disk/tape system is powered by an HSM (CASTOR) interfaced to the Objectivity/DB AMS


BURST databases
RUN databases (RUN objects contain
Start-Of-Run informations, processing info, etc...)
EVENT directories (event headers)
Disk resident
Associations set in parallel
with the DST production
RAW data
DST data
Under AMS - CASTOR control

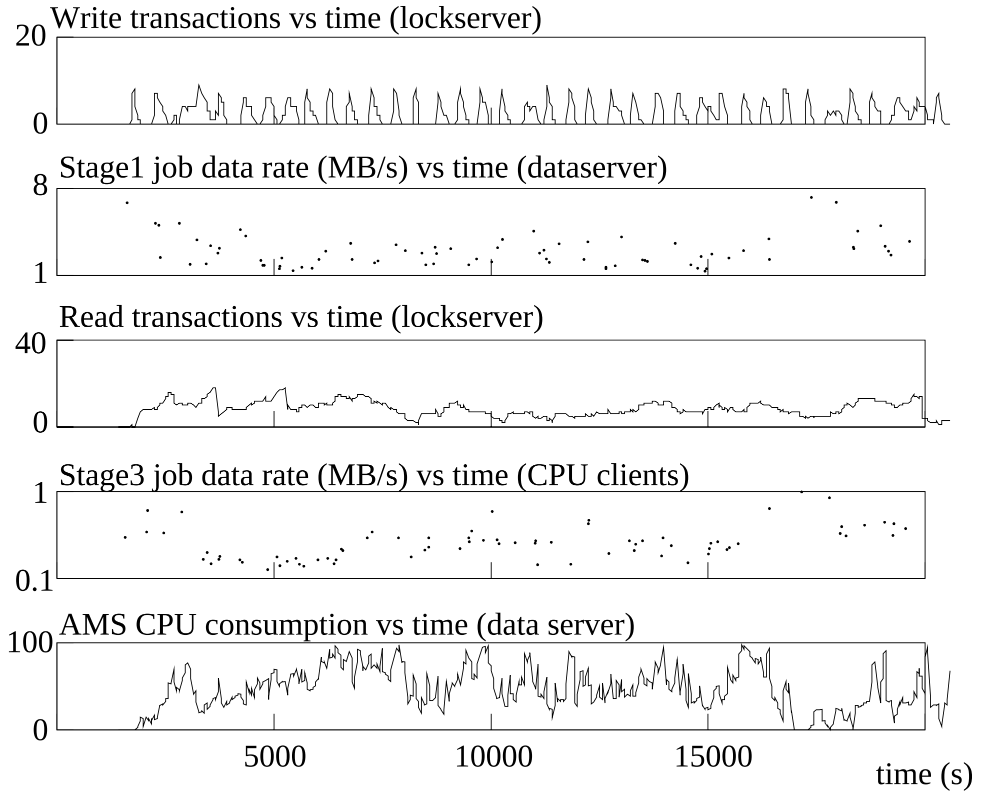
DAQ mode with real data

- Switch on the stage1 system (input data into Objectivity/DB) with real data. Not optimized for the actual data size (3 kB instead of 30 kB)
- Accumulate for hours and then run to catch up data acquisition rate (~1 MB/s)
- Confirmation of the "mock data challenges": speed as expected, no problems.
- In particular, the run data bases were at the same time scanned to avoid double insertion (just for test, other protections exist), scanned for the "stage2" and "stage3" activities (see next slides), and updated with the new run entering the DB federation: no bad lock)



Reconstruction

- After “stage1” all data are on data bases, but not all associations (to avoid locking). “stage2” enters into the game: successfully tested (practically no data are written, only run to event directories are set, no lock observed)
- The “stage3” is the reconstruction stage (running Coral). This could be done, with Coral scanning the events (no online reconstruction)
- No real clash between the reading and the concurrent writing into the run data bases
- Reading congestion problems observed (already observed in MDCs). The aggregate I/O (reading) drops in respect single stream value.
- Many reasons: bad DB physical layout (Objy bug fixed but not tested); some strange network overhead; possible inefficient multithreading performances; too many clients on one AMS (>>10 clients, the design value)
- Mock data results (25% of the farm)



Conclusions

- The Compass Computing Farm is (at the 99% level...) a Linux PC farm
- Tests (aka Mock Data Challenges)
  - we find nice agreement on the behaviour of our system with real or mock data
  - DAQ mode; Quasi-online reconstruction; Condition data base access
- CCF Software and monitor tools
  - Successfully tested on battle field. Monitors crucial to make sensible tests and commissioning: you cannot have too many!
  - All components demonstrated to be robust and error resilient: disk (servers) problem, CPU client or user program crash, lock server crash... No DAQ stops due to Objectivity/DB.
- Very nice results in DAQ mode (test up to 35 MB/s test)
  - Most important mode of operation completely demonstrated
  - Similar performances (i.e. MB/server) observed with real data
- Quasi-online mode (AMS/HSM/CORAL)
  - Prototype system works, more tests and optimisation needed