CERN IT Department

AFS Backup System at CERN

Jakub T. Moscicki

(for the CERN AFS team)

CERN IT-DSS

EAKC 2014

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

Content

- Current status
- Evolution of AFS (Backup) at CERN
- Recent developments (aka new backup system)
- Outlook

Current AFS backup in numbers

- /afs/cern.ch
  - 300TB total
  - 80K volumes
- Daily backup
  - ~10K volumes dumped
  - 10-15TB traffic to tapes
- Total backup
  - ~2.5 PB on tape (x 2 for dual-copy)
  - ~2.5 millions object in the backup archive
  - keep daily snapshots for 180 days
- Restores
  - ~60 volume restores from tapes monthly

- we are using the official TSM backup service – managed by another team

TSM backup service

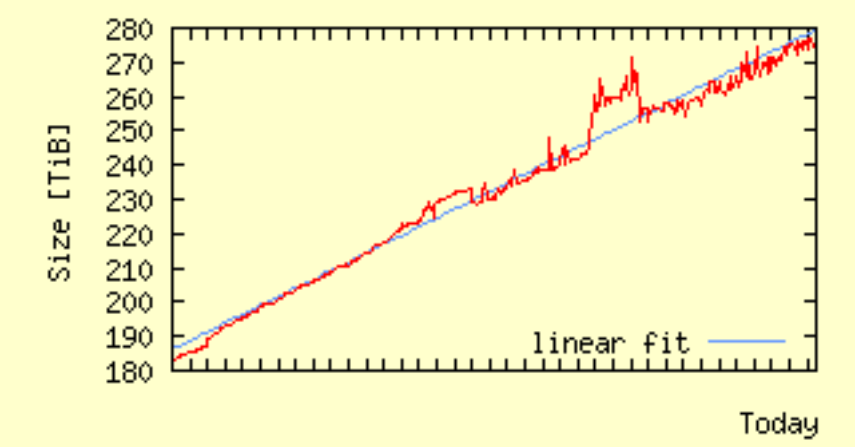
- TSM Server
  - nice “black box” (shared with other users)
  - IBM tape library behind
- Running smoothly
  - but we need to be careful to comply with the requirements
    - backup window and disk cache size
- Details
  - 18 GB of disk cache
  - 6pm-10am backup window
  - 10am-6pm migration to tape window
  - migration to disk kicks-in when disks at 60% full
    - Best if backup and migration windows are separated

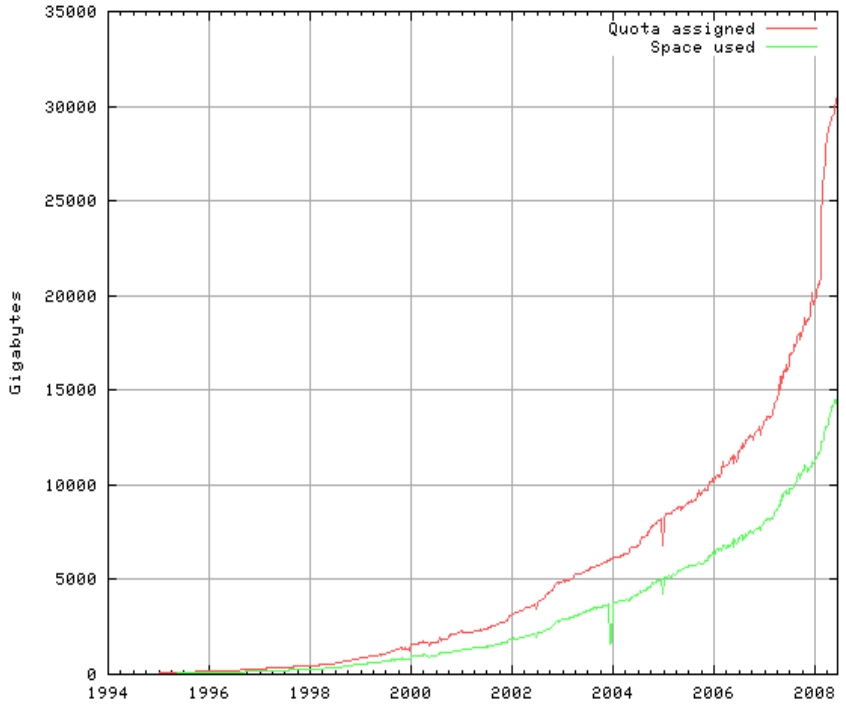
Operational experience

- We backup ~50 servers in parallel
- Slowest backup sessions finishing in 12-16 hours
  - (trying to) respect the TSM tape migration schedule
- Backup “touches” all the volumes
  - problems with ~10-15 volumes a day (0.015% of all volumes)
    - e.g volumes LOCKED or dead corpses after failed vol moves
  - “competing” with another volserver user
    - automatic volume balancer (“pool monitor”)
    - ~50 volumes moved daily
- Recent, subtle 1.6 change in the vos backupsys output broke parsing!

The only constant is change

- AFS data growth





- N.B. the daily backup traffic in 2014 ~ the total cell size 6 years ago

The only constant is change (2)

- Tape backends
  - 2014: towards CASTOR2
  - 2013: TSM 6
  - 2007: TSM 5
  - 2000: CASTOR1
  - *20th c. : STORAGETEK DLT ROBOT*
  - *20th c. : DLT STACKER*
  - *20th c. : EXABYTE STACKER*

The only constant is change (3)

- Backup Policy Changes (2011)
  - Remove distinction between backed up and non-backed up volume types
  - Shorten the retention period: from 18 months to 6 months
- Motivation
  - Simplification: easier for the user and less likely to confuse them
  - Harmonization across services (AFS, DFS)
  - Analysis of the real usage and needs (backup restore frequency in time)

- Consequence: changes in the load on the backup system

Evolving the AFS backup system

- Until 2013 we had the “old” AFS backup system at CERN
  - good service but accumulated 20 years of history
  - original AFS backup system design dates back to antiquity
    - … when tapes were mounted manually / tape devices directly attached
  - eventually became very complex and difficult to maintain
- Juggling several server-side processes:
  - backup database
  - butc (backup coordinator) process on each server
  - expect wrapper script for to automate the functioning of butc
  - home-made xbsa.so plugin for butc to talk to TSM via an abstract “standard” API
  - a lot of static configuration
    - volume sets
    - statically defined backup cycles in the backup database (but with a hack “local perl Catalog” to allow differential dumps)
    - tape port offset numbers, …
- Backup has been the concern in the community for some time already (EAKC 2011)
  - https://indico.desy.de/contributionDisplay.py?contribId=11&sessionId=3&confId=4756

New AFS Backup System (ABS)

- Exchanged ideas with DESY
  - Wolfgang Friebel and Peter van der Reest
- Drastically simplify the architecture
  - Treat the underlying archival service as an object store (with a simple catalog)
    - PUT, GET, DELETE, LIST
- “just a bunch of scripts”
  - high-level driver in Python
  - module (plugin) to connect to the backend archive service

ABS in a nutshell

- dump

  for each partition:
  vos backupsys
  for each volume:
  vos dump [-time yesterday] | PUT

- restore

  for each dump in LIST:
  GET && vos restore [-overwrite i]

- We put and get from archive named volume dumps (files)
  - PUT = send a voldump file to archive
  - GET = retrieve a voldump file from archive
  - LIST = list all voldump files for a given volume

Keeping track of dumps

- metadata encoded in a dump name
  - **(volid, volname, backup_timestamp, previous_timestamp, ...)**
  - previous_timestamp=0 → FULL BACKUP
- list of dumps stored in the catalog of the archive backend
  - linked list of incrementals rooted at the full dump
  - if volume unchanged then the incremental may be skipped
    - e.g. then simply previous_timestamp = day – 2

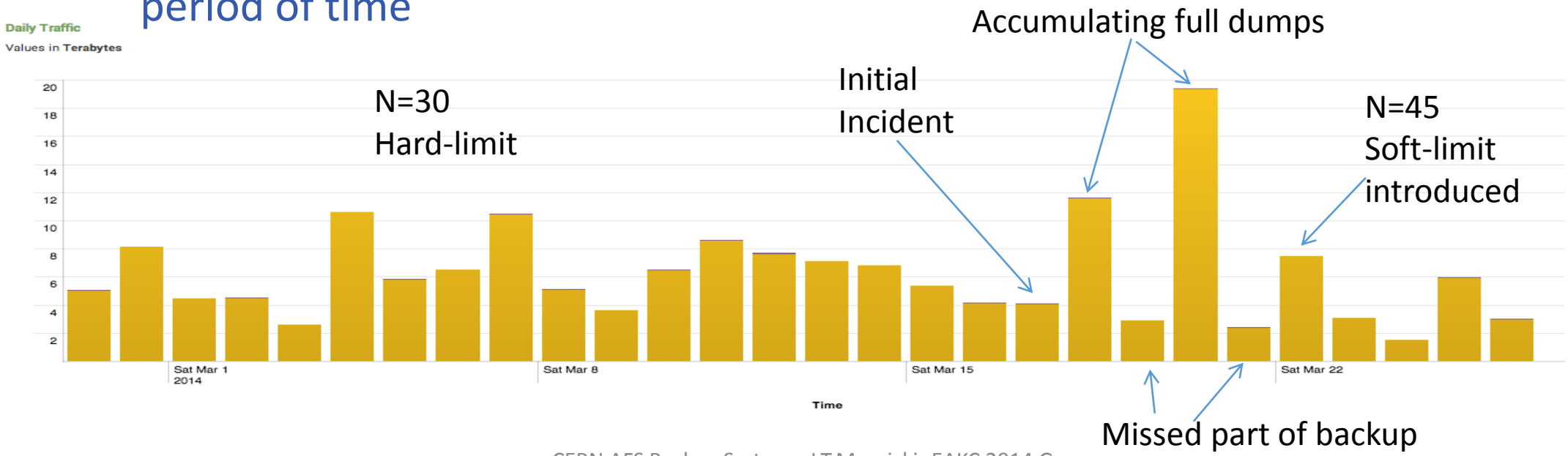
Backup cycle of a volume

- Full backup cycle (N days)
  - FULL – INCREMENTAL – INCREMENTAL – …
    FULL – INCREMENTAL – INCREMENTAL – …
  - Presently N=45
  - Ideally full dumps should be uniformly distributed in time
    - determine the full backup day by computing the volume id hash uniform in range [0,N)

- N is easily adjustable without creating full dump storms
  - if you change N then full dumps will still be distributed evenly
- The longer the cycle:
  - the more incremental dumps to recall
  - the longer effective retention period (more accumulated data)

Handling excess of daily traffic

- If for any reason backups are missed then the full dumps accumulate
  - a hard-limit on the maximum timespan between the full backups
  - daily traffic increased beyond what we can currently support with TSM
  - a one-liner fix: introduced soft limit smearing out any full dump excess over a period of time



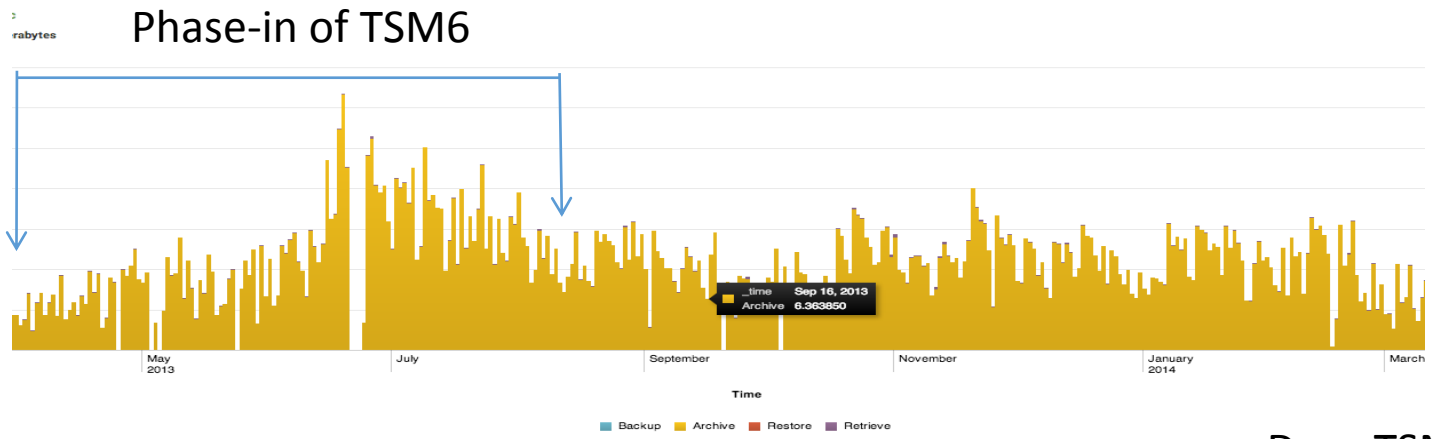
ABS with TSM

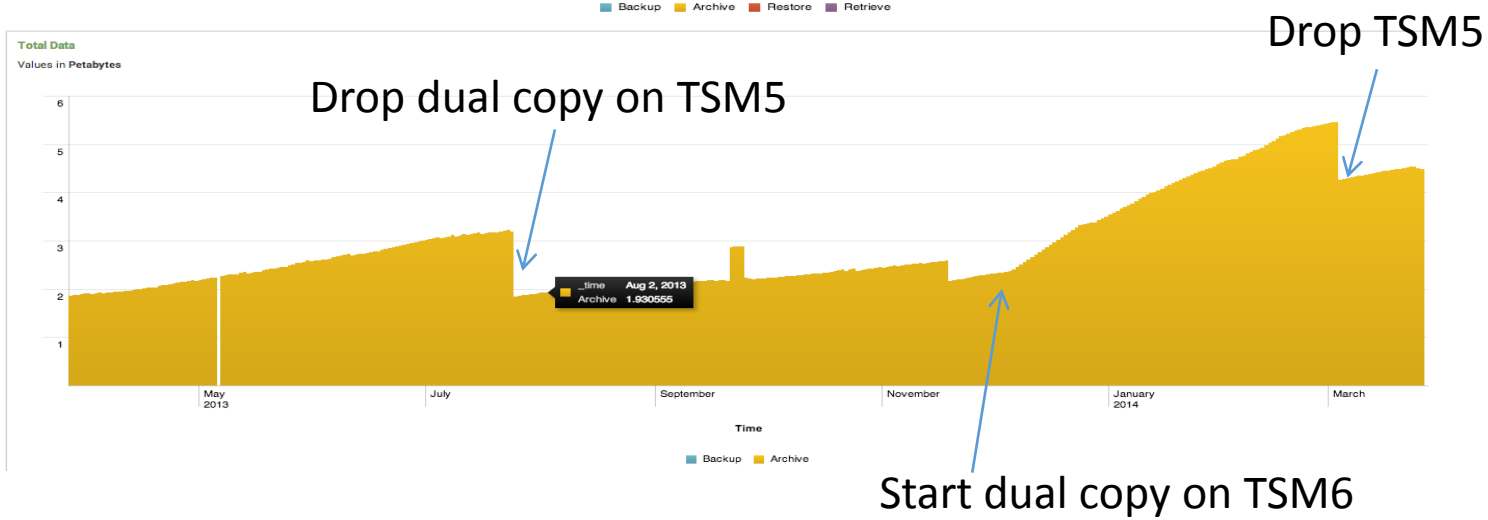
- TSM-specific functionality encapsulated in a separate module
- We rely on the catalog provided by the archive system
  - But the TSM db query is not very efficient
  - So we have a catalog cache on AFS directly
    - cache entries stored in a balanced tree in the filesystem
    - cache consistency is protected by md5 checksums
- We skip unchanged volumes (vos exa → Last Update)
  - Do not send „empty” incremental dumps
- We have a TSM archive client capable of reading input from stdin
  - implemented in-house in C using TSM API
  - rationale: dsmc archive requires input files to be on disk

Migration experience

- Last year we migrated from old backup system to ABS
  - …and from TSM5 to TSM6 at the same time…
  - TSM6 servers on much bigger hardware and designed to cope with a higher load (10-15TB daily traffic)
- Transition was a smooth but long running process
  - run both in parallel until get confidence in the new system
  - keep data in the old system to respect the retention period
- Need to control the volume of data
  - Cannot do full backup dumps on day one -> 300TB would be sent in 24 hours
  - ABS allows to define flexible schedules for a gradual phase-in partition-by-partition and server-by-server

Migration in numbers





N.B numbers cumulative:
TSM5+TSM6

Towards Castor2

- Concern: total data stored on tape
  - we are approaching the license ceiling with IBM for the TSM
  - TSM model will not scale further beyond current daily traffic figures
    - one server bundles namespace with the disk cache migration to tape
- Castor2
  - Currently stores ~100PB
  - Global namespace but direct access to multiple data servers (better scaling)
  - Needs client side encryption

ABS with Castor2

- ABS with Castor2
  - data encryption
    - **vos dump | openssl enc | PUT_TO_CASTOR**
    - encoding type (with possible key version number):
      - easy: another metadata attribute in the dump name
  - efficient, hierarchical catalog of files
    - should be able to drop our TSM catalog cache
  - Castor2 plugin prototype module
    - sufficiently easy, 150 lines of python

Some scaling considerations

- Scaling limits (as the servers grow larger)
  - daily backup == we have 24 hours to get the data out of the server
  - processing partitions in parallel
    - easy: this is adjustable parameter in ABS
    - but: volserver may become the next bottleneck limiting the total throughput from a single server
  - possible congestion of volserver
    - we cannot keep it busy all the time with just doing backups and nothing else
- ABS CASTOR2 prototype benchmarks (preliminary)
  - aes256 encryption enabled
  - 3 partitions in parallel: 4.7TB backed up in 23.5 hours
    - volserver at 40-50% CPU busy
    - disks not saturated
    - openssl processes at 5-10% CPU each
  - 8 partitions in parallel: 10TB backed up in 24 hours
    - volserver at 120% CPU busy

Summary

- We have a flexible backup system for our growing AFS cell
  - Fully operational and handling ever increasing loads
  - We can relatively easily adapt it to changing requirements

- It is much easier to operate and manage
  - less processes to handle
  - less static configuration
  - simple-minded model using a minimal feature set of our archive systems