# SRM Usage Monitoring

Dirk Duellmann

(many slides by Giuseppe Lo Presti with input from Shaun De Witt)

CERN IT

SRM Workshop @ DESY

18th May 2009

# Motivation

- SRM will be used for STEP09 and 2009/2010 run
  - Workload will still ramp-up for the run period
- Do we have the monitoring in place to answer questions like:
  - How much headroom does the system provide for increased experiment requests?
  - Can we quickly spot current bottlenecks?
    - SRM processing (front-end vs back-end)
    - Storage back-end
  - Do we have an end-to-end measure of failures and retry counts at the different levels?
  - Can we spot unexpected use and originating users?
- Can we connect the usage numbers from experiment production systems down to storage
  - at Tier 0, Tier 1 and Tier 2
- Do we even have suitable metrics logged?

# First Step - collect status quo

- Discussions so far between
  - FTS, CASTOR, dCache, DPM
- Main focus so far - Tier 0 processing
  - and Tier 1 logs from FTS, CASTOR
  - starting to collect & process DPM logs

# Present deployment and statistics gathering

- At CERN
  - 5 endpoints, one per LHC VOs + general public instance
  - 3 nodes each
- At RAL
  - 5 endpoints; looking at ATLAS and CMS only
  - 2 nodes each, except ATLAS (4 nodes)
- At CNAF
  - General endpoint + CMS dedicated endpoint
  - 3 nodes each

- Statistics gathered on a **single node** of each endpoint, for a 2-month time interval
  - From March 1st to April 30th

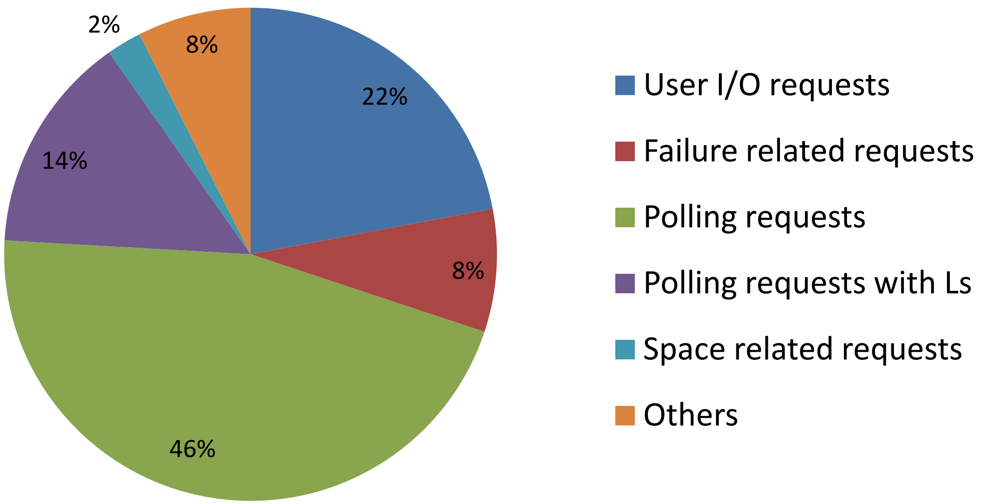
# SRM methods: a classification

- To ease the breakdown analysis, the following categories have been defined:
- User I/O requests
  - srmPrepareTo, srmCopy, srmBringOnline, ...
- Failure related requests
  - srmAbortRequest, srmAbortFiles, srmReleaseFiles
- Polling/query requests
  - srmPing, srmStatusOf, srmLs
- Space related requests
  - srmGetSpaceTokens, srmReserveSpace, ...
- Others
  - **12** more methods (the specs include **39** methods)

# Breakdown by SRM method 1/3

**srm-cms @CERN**

**1.2 reqs/sec**

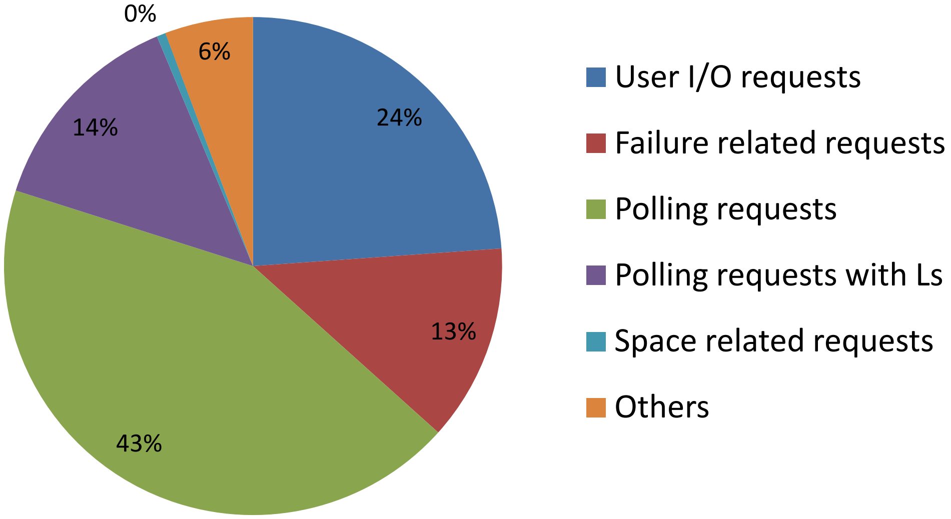


- Observations
  - Fair ratio of polling vs. I/O (prepare) requests
    - but note the amount of srmLs, also used for polling
  - Failure/success ratio not taken into account
    - These are all the incoming requests
    - “Failure related” requests are normally issued to clean up after a failure has occurred at either ends

# Breakdown by SRM method 2/3

**srm-public @CERN**

**0.4 reqs/sec**

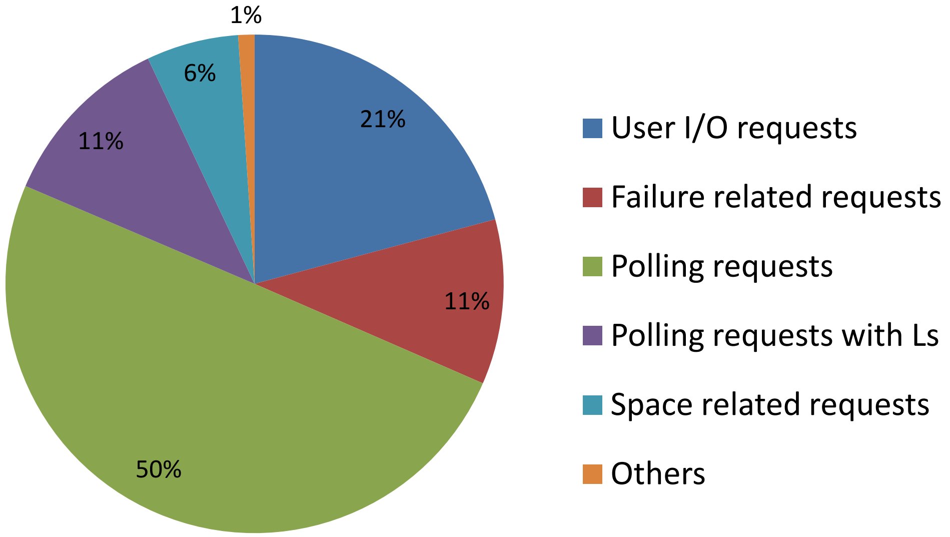


- Observations
  - The “others” category for srm-public includes the whole set of existing SRM methods
    - whereas only a fraction of them is effectively used elsewhere
    - srm-public serves the DTEAM VO, and many SRM tests (e.g. S2) run as DTEAM...

# Breakdown by SRM method 3/3

**srm-cms @RAL**

**0.5 reqs/sec**



- Observations
  - At a Tier1 the ratio polling/prepare requests is slightly worse
  - And the number of “other” requests is negligible
    - Only **14** SRM methods used, out of the **39** in the specs

# Breakdown by client

- No detailed data yet...
- But main SRM client @ CERN is **FTS** by far
  - 80-90% of the total load, depending on the endpoint
- Clients at T1 sites typically just follow

# A word on throughput

- A clear evidence from this exercise is the different behavior depending on the VO
  - ATLAS ran at **8** requests/s, 5 times more than LHCb or CMS, whereas ALICE ran at 2 orders of magnitude less
    - The ATLAS average file size played a role here
    - To be still checked whether over the observation period all VOs ran at any constant load
    - **STEP'09** will hopefully provide a baseline
  - The load at T1s is of the same order of the load at the T0
    - T1 storage activity is much more "Grid-oriented", thus it mostly goes through SRM

# Conclusions

- General request-to-poll ratio looks OK
  - ping rate from FTS should be reviewed
  - don’t expect large load reduction once “client back-off” is in place
    - but feature was meant for overload conditions anyway
  - should look at ls vs req status
    - once req status is fixed!
- Categorisation of call rates suggested by Giuseppe looks useful (also to experiments?)
  - Now being implemented into CASTOR monitoring as automated collection from DLF records
    - available for site fabric monitoring as other metrics

# More conclusions

- Rate monitoring will also allow for detection of unused API calls
  - apart from public SRM endpoint were S2 tests probe “full” API
  - this alone does not mean much though!
- First results not fully understood yet
  - eg ATLAS rate expectations differ from FTS and SRM by one order
  - eg RAL rates look high
- Should try to close loop with experiments now and put automated usage monitoring in place before the run

# Topics for discussion

- Can we agree on call and error rate metrics here?
  - For error rates a minimal subset could:
    - how many user perceived errors came from SRM component itself?
    - how many from the storage back-end?
    - how many from remote end of a transfer?
    - Focus on spotting costly or deep retrials
      - transfers failing at the end
      - repeat request which go more than one level into the s/w stack
- Rate metrics per node and per VO?
  - probably need both
    - for site diagnostic of h/w utilisation
    - for experiment cross check with total load from production system
- Latency metrics (eg time to turl)
  - so far obtained via FTS
  - intrinsic calculations and regular summaries?