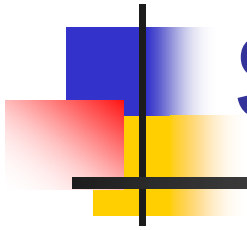


StoRM: status report

A disk based SRM 2.1.1 server

StoRM: status report

Result of collaboration between:

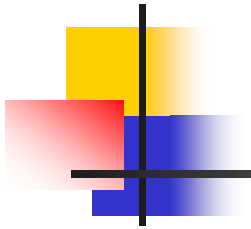
INFN - Grid.IT Project from the Physics community

+

ICTP - EGRID Project: to build a pilot national grid facility for research in Economics and Finance (www.egrid.it)

StoRM: status report

- Summary:
  - Objectives to achieve
  - Implementation strategies
  - SRM v2.1.1 functionality currently available
  - Release process
  - Simple use cases



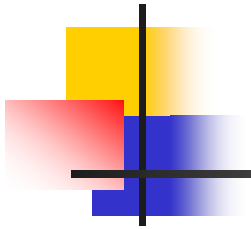
I. Objectives to achieve

StoRM objectives

- StoRM's implementation of SRM 2.1.1 meant to meet three important requirements from Physics community:
  - Large volumes of data exasperating disk resources: **Space Reservation** is paramount.
  - Boosted performance for data management: **direct POSIX I/O call.**
  - Security on data as expressed by VOMS: strategic integration with **VOMS proxies**.

StoRM objectives

- EGRID Requirements:
  - Data comes from Stock Exchanges: very strict legally binding disclosure policies. **POSIX-like ACL access from grid environment**.
  - **Promiscuous file access**: existing file organisation on disk seamlessly available from the grid + files entering from the grid must blend seamlessly with existing file organisation. *Very challenging – probably only partly achievable!*
- StoRM: disk based storage resource manager... allows for controlled access to files – major opportunity for low level intervention during implementation.



II. Implementation strategies

StoRM implementation: behind SRM v2.1.1

- VOMS/Security
  - ACLs on disk's filesystem seen as natural mechanism for enforcement
  - StoRM **requires** ACL capable filesystem (ext3, RaiserFS, GPFS, ...)
  - Physics community access patterns set at the beginning: natural to partition access rights into blocks of special local-users which grid credentials get mapped to.
  - ACLs tend to be set up earlier on: Ahead Of Time approach.
  - Easily supports present day naïve file access based on VO membership.

StoRM implementation: behind SRM v2.1.1

- Boosted POSIX I/O
  - Performant parallel filesystem distributed over all WNs of farm: **GPFS**, **Lustre**, ...
  - StoRM's SRM logic decoupled from specific filesystem chosen: requires only to write/use specific filesystem module.

StoRM implementation: behind SRM v2.1.1

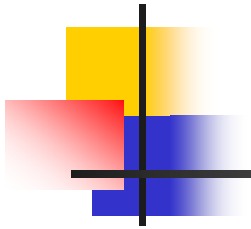
- Filesystem with **native support for space reservation (GPFS, ...)**

  Native support offers high robustness as no metadata catalogue is used: no need for critical synchronisation with underlying filesystem state.

StoRM implementation: behind SRM v2.1.1

Important feature to meet EGRID requirements: support for **JustInTime** approach to ACL set-up.

- ACLs absent on filesystem; applied on the fly for the particular pool account user to whom grid credentials get mapped; removed once data management completes.
- No need for initial partitioning of local Unix accounts based on required access rights.
- Tackles head on scalability issue on security + aids in promiscuous file access.

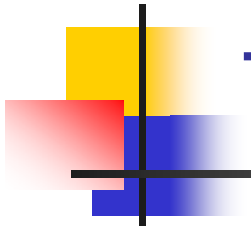


III. SRM v2.1.1 functionality currently available

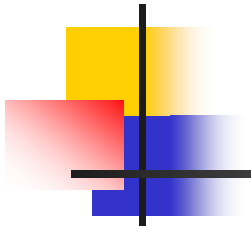
StoRM's SRM v2.1.1 functionality

- Presently available:
  - srmPrepareToGet
  - srmPrepareToPut
  - srmCopy in Push Mode
  - srmReserveSpace + supporting functionality
  - srmXXXRequestStatus
  - Volatile + Permanent file storage type
  - SRM clients
  - Simple access rights

StoRM's SRM v2.1.1 functionality



- Within the next couple of weeks:
  - srmLs
  - srmCopy in Pull Mode
  - Finish off sorting out of security issues between StoRM's different tiers of architecture



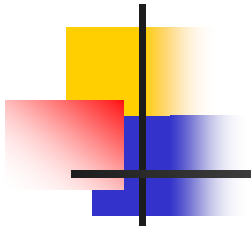
IV. Release process

StoRM's release process

- Development machine at CNAF where new features are tested, debugged and integrated.
- Since June there is an expanded co-operation with CNAF: testbed of several machines with GPFS – functionality test of all features listed before

StoRM's release process

- By second half of November there will be official release to select users



V. Simple use cases

StoRM simple use case scenarios

- Use case 1: POSIX I/O usage
  - StoRM presides over files on a SE: GPFS Filesystem spread over all WN. Access to data is granted simply on VO membership basis.
  - Grid user submits job; job reaches WN; job first executes SRM client for getting the file directly.
  - StoRM verifies grid user has right permissions; StoRM returns a TURL with file handle; if StoRM is using JiT: it sets up an ACL for local user to which grid credential is being mapped.
  - Job processes the file. If JiT: StoRM removes ACL when job finishes.

StoRM simple use case scenarios

- Use Case 2: moving large dataset from source StoRM to destination StoRM (generally applies to all SRM servers)
  - SRM client issues srmReserveSpace on destination StoRM; destination StoRM checks requesting user has permissions; destination StoRM returns SpaceToken.
  - SRM client issues srmCopy to source StoRM for pushing data set to destination StoRM, given space token.
  - Source StoRM checks permissions + negotiates with destination server a GSIFTP transfer.

StoRM simple use case scenarios

- Use case 3: computing centre wishes to join an existing grid infrastructure; no/little impact on users' organisation of files – users continue to organise files as they have always done.
  - Centre's user submits job to the grid; output file gets saved on centre's StoRM SE.
  - Another centre's user wishes to perform local computation on newly created file: user has no need to be aware of special arrangements for data produced from the grid.

Acknowledgements

- On going technical partnership with J.P.Baud of DPM
- StoRM team: Alessio Terpin, Ezio Corso, Flavia Donno, Heinz Stockinger, Luca Magnoni, Riccardo Murri, Riccardo Zappi. Project leader: Antonia Ghiselli.