Data Handling Jamboree report

- WLCG meeting in June, Amsterdam
  - and follow up from collab. Meeting, SW week,...
- General theme and assumptions
- Demonstrators

General Theme and Assumptions

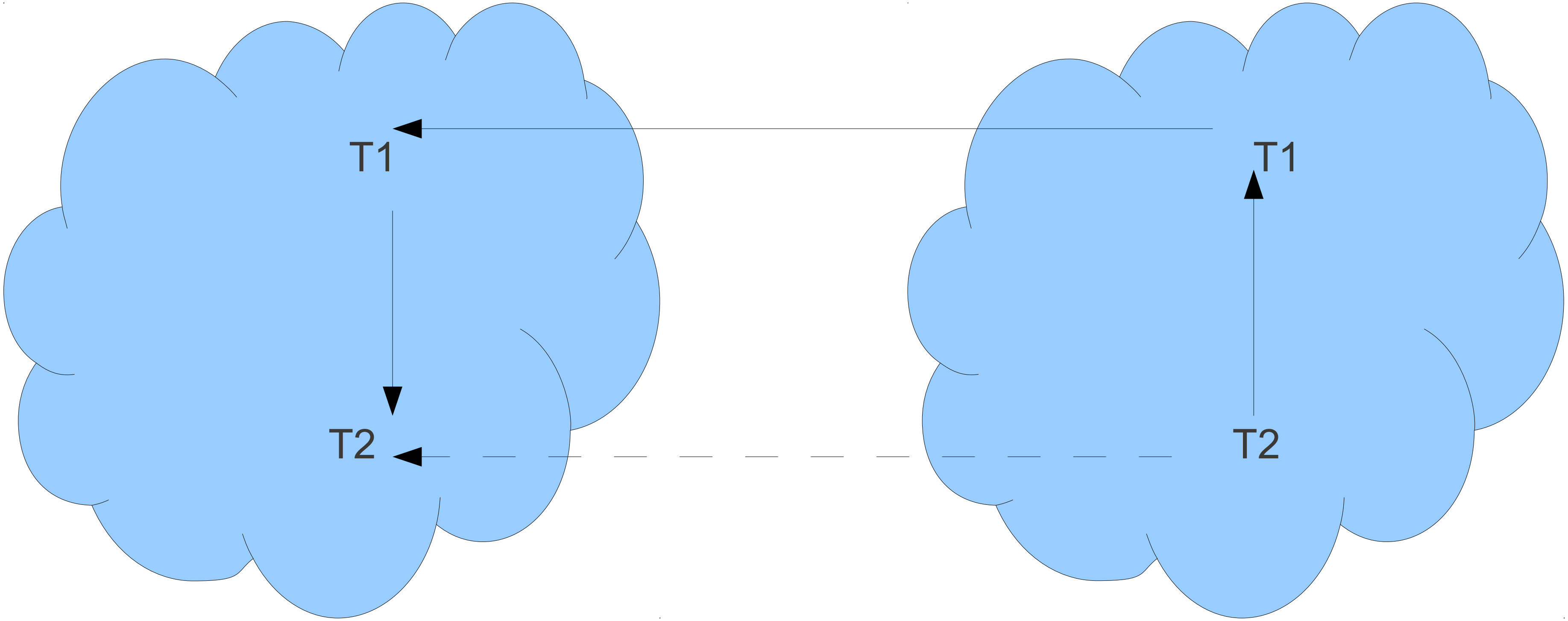
- MONARC model, of hierarchical Tiers, is 10 years old
  - based on the assumption that network bandwidth would be primary limitation
    - minimize WAN traffic: keep within national research net
    - jobs go to data: write once-read many
- In practice, network bandwidth was never a constraint
  - and will even improve: industry driven
  - constraint is own middleware
    - SRM, FTS, LFC.....poor reliability, performance and no failover
- Standard protocols and solutions preferred
  - funding for development/maintenance dries up

MONARC in action

- OPN for T0 – T1 is probably correct
  - only added T1-T1 connections
- After much work and expt. tools takeover
  - we can move lots of data reliably
    - within MONARC model, i.e.T0-T1-cloud T2s
    - between T2s of different clouds, restriction is only FTS
      - current hop via T1s has no basis in network topology

T2-T2 between clouds

Via T1
SCRATCHDISK



FTS redesign

- Channels designed for network limitation
- Typically control inbound traffic
  - but no coordination between FTS instances
  - SRMs apparently still need protection
- Allow anywhere to anywhere and let routing do its job
  - optimizing N-to-N channels is not easy(ask CMS)

PD2P

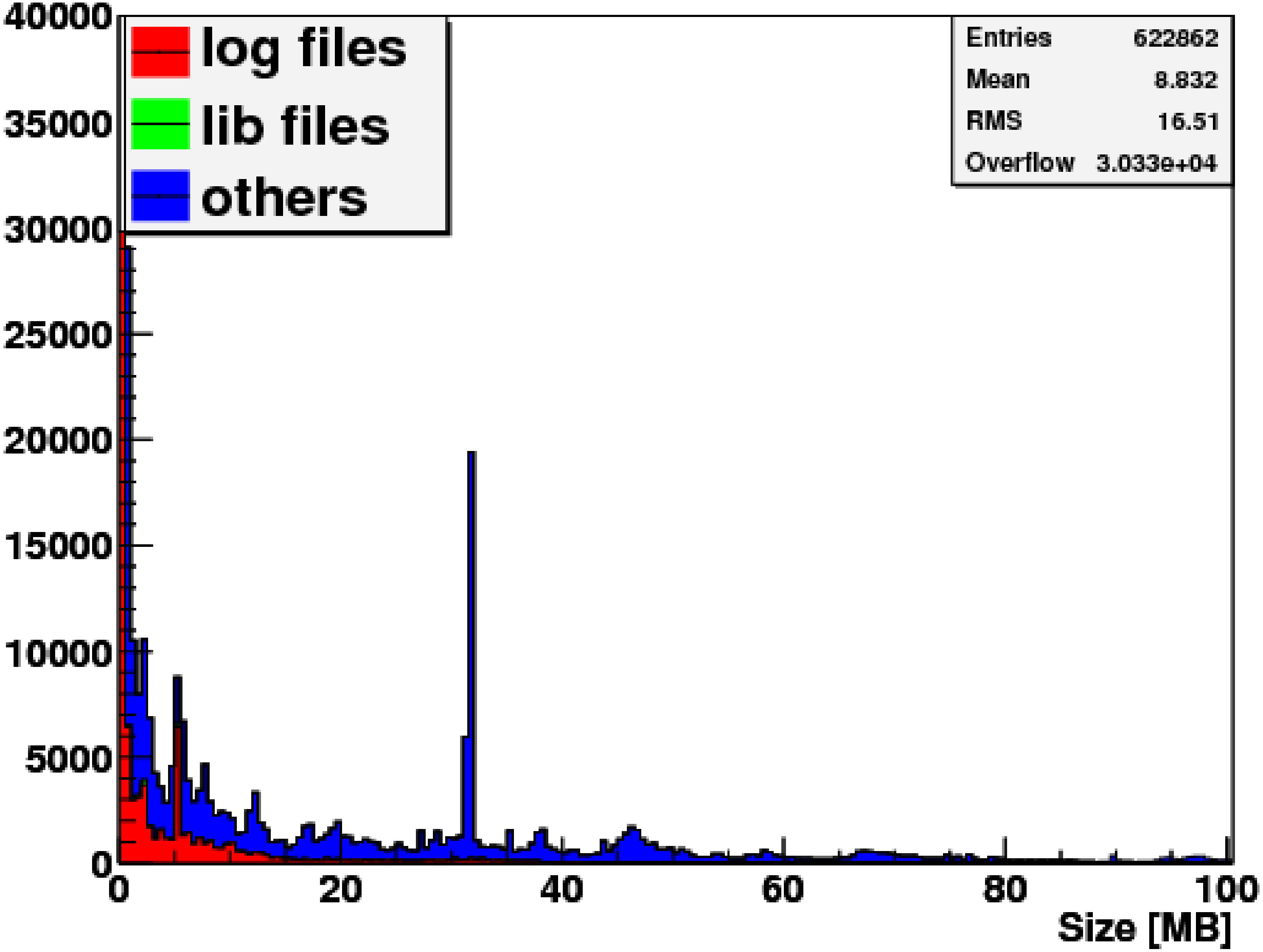
- Currently pre-place data based on expected access pattern
  - much of the data is not touched
  - users access data which is not distributed
    - e.g. ESD only at T1s
- Panda places datasets based on real user jobs
  - subscription made, when job queued at T1
  - subsequent jobs can go to subscribed T2
    - or re-broker initial job
  - only distribute data which is used

Chirp Demonstrator

- “A file system for Grid computing”
    - shared FS with full acls and x509 authentication
    - simple: users can easily deploy servers
- Several features of global file system
    - access it from user job and client
        - workshop assumption of WAN usage allows this
    - keep small files out of DDM,LFC,SRM,FTS,...
- May be replaced by NFS4.1 or webdav(http) when functional
    - not anti-standard protocols

http://www.cse.nd.edu/~ccl/software/chirp/

User File sizes



9MB

Server and Client

- Start your own server (then it trivially scales)
  - chirp_server -r /data &
  - needs 1 port open
- Access the server from anywhere
  - chirp_put localFile host.lmu.de /mydir/theFile
  - puts file in host:/data/mydir/theFile
- Or FUSE the file system
  - $ mkdir chirp
  - $ chirp_fuse chirp
  - $ ls chirp/host.lmu.de/mydir/theFile
  - And all other posix operations
- All of the above are x509 authenticated

Auth and acls

- Each directory has hidden file .__acl

```
globus:/C=DE/O=GermanGrid/OU=LMU/CN=Rodney_Walker rwlda
globus:/DC=ch/DC=cern/OU=Organic_Units/OU=Users/CN=hanawa/CN=678589/CN=Keita_Hanawa rl
globus:/C=AT/O=AustrianGrid/OU=UIBK/OU=astro/OU=HEPHY/CN=Brigitte_Epp rl
globus:/C=CA/O=Grid/OU=westgrid.ca/CN=Roghaiyeh_Dastranj_Tabrizi_42 rl
```

- User controls acl for their directory
  - e.g. can give access to analysis group
    - share files produced on NAF
    - aggregate output from jobs running on Grid

Status

- Chirp server at CERN configured with write access for ATLAS VO
- Panda pilot writes output and log files
- Ganga config for user to set chirp host and path
    - config.Panda.chirpconfig='chirp^voatlas92.cern.ch^/RodWalker^'
- Chirp client available on CERN afs
- Ready for beta users
    - https://twiki.cern.ch/twiki/bin/view/Atlas/ChirpForUserOutput

Caching via xrootd

- xrootd instances on different sites can get files from one another, and cache them
  - 1PB test system at CERN
  - to be loaded with ATLAS/CMS data
- Currently asking for volunteer test sites at 1,10,100ms RTT
- xrootd enthusiasts seem to have found support from CERN
  - Castor is particularly bad for direct io

Block caching

- TTreeCache intelligent read-ahead
  - learning phase of few events
  - 30MB vector read of only the blocks needed
- Xrootd caches sparse file
  - only fills in the bocks requested in a vector read
  - blocks available to other jobs on the site
- root caches sparse file on 'local' FS
  - local can be shared FS
  - provide api for for any cache mechanism, e.g. xrootd

Standard Protocols

- NFS4.1 preferred for direct access
  - uses file protocol and buffer cache
  - no vector read, but posix fadvise (asynchronous load of vector)
  - sparse file caching and global filesystem capabilities unknown (to me at least)
  - Dcache and DPM will have NFS4.1 access
    - disgard rfio and dcap, all is file
- NFS4.1 not industry standard yet

Standard Protocols(2)

- http is industry standard with many tools
  - vector read supported, root can read http files
- Caching and replica discovery
  - Squid or dedicated appliances
  - partial file caching not yet supported(blocks)
    - is this only reason to use xrootd(not industry standard)
- not aware of large scale http access tests
  - concern that xrootd enthusiasts lead us to yet another HEP only solution

Conclusions

- “MONARC is 10 years old”
  - or it was just wrong
- ATLAS data access is ok, and better with minor changes
  - better guess/enforcement of user access pattern
  - on-demand distribution and FTS N-to-N
- CERN Castor direct access driving wilder revolution
  - installing dCache would probably suffice
  - danger of another HEP solution to standard problem