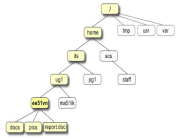


XCFS

an analysis disk pool & filesystem based on FUSE and XROOT protocol

Abstract ID 171

Andreas-Joachim Peters
CERN
IT-DM-DA

Overview

- Introduction
- Architecture
- Characteristics
  - FUSE implementation
  - Features
  - Performance Tests & Comparisons
  - Mass Storage Integration
- Summary & Outlook

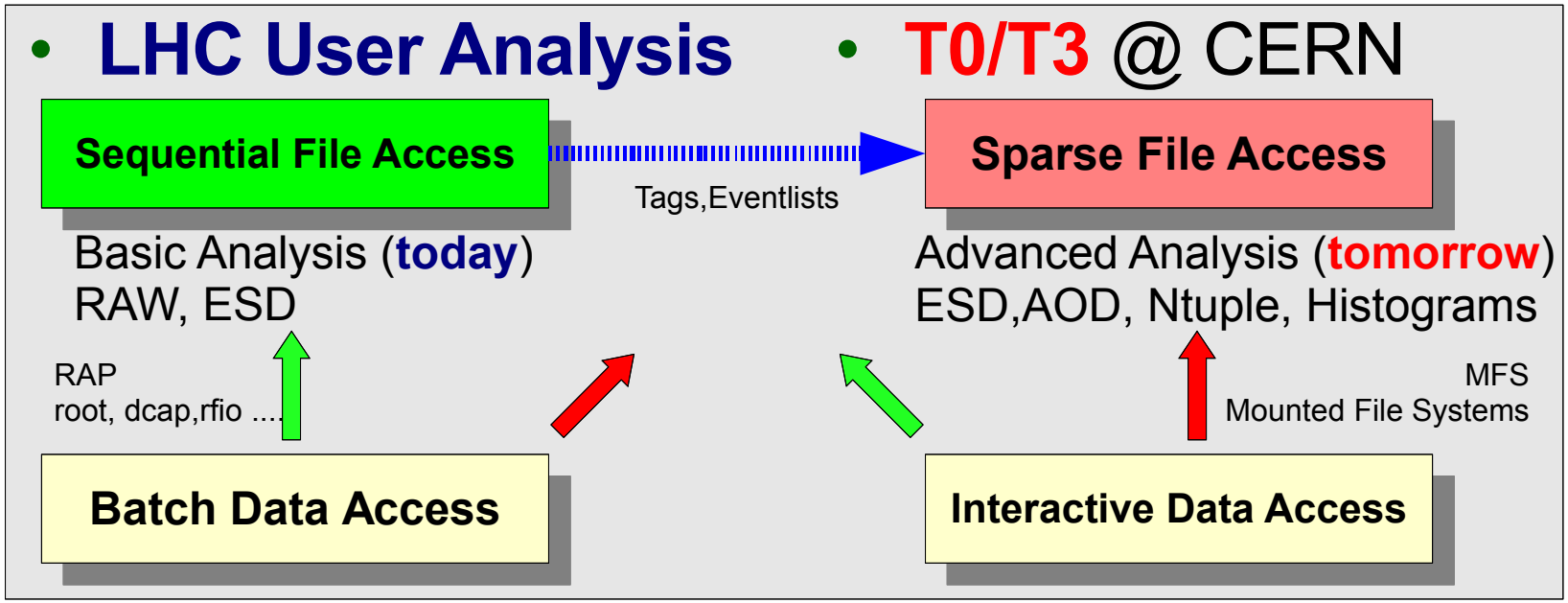
Introduction
Two access 'philosophies'

- **R**emote **A**ccess **P**rotocols vs. **M**ounted **F**ile **S**ystems
  - **RAP**
    - deployed in user space (can be done on the fly)
    - Strong authentication libraries (Krb5/GSI/VOMS)
    - Non-posix optimizations possible
      - e.g. read hints
        - » **asynchronous** read-ahead (for network latency comp.)
        - » **Readv** – compound reads (to reduce #I/O requests)
    - Virtual user model easy to implement
  - **MFS**
    - Kernel implementation & root deployment needed
    - Usually best performance (kernel space)
    - Posix IO model
    - Strong Authentication is tricky (VOMS?)
    - OS supports UID/GID user/group pairs

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

Introduction

The Challenges for LHC analysis



- Boundary Conditions
  - GRID environment
    - GSI authentication
    - User space deployment
  - CC environment
    - Kerberos, - admin deployment
- High I/O load
- Moderate Namespace load
- Many clients ***O(1000-10000)***

- Preferred interface is MFS
  - Easy, intuitive, fast response, standard applications
  - Modreate I/O load
  - High Namespace load
    - + Compilation
    - + Software startup
    - + searches
- Less Clients ***O***(#users)

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

Introduction
History

- **2008: CERN** needs **T3 analysis facility**
  - **CERN**'s **DM system** [Castor] **needs extension** for analysis requirements
    - Need for better support of advanced user analysis
      - » Currently high file open latency
      - » Currently low file open concurrency
      - » No interactive access via a mounted file system
  - Need to **study** possible solutions in **short time scale** (for October 2008)
  - ⇒ **R&D project in DM group** studying existing solutions
      - » DPM
      - » Castor2
      - » Lustre
      - » Scalla/xrootd

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

Introduction

R&D Prototype Studies

- **R&D** development project 2008 as a feasability study for enduser analysis
  - **DPM/Castor2/Scalla** didn't fullfill requirements
  - 1st prototype was based on Lustre using an xrootd-lustre interface
    - **tight integration** (no gateway model!)
      - » xrootd on each lustre OST–xrootd server reads local file
      - » large read-ahead on disk server and client
    - **very good performance** for **RAP** & **MFS**, but
      - » Strong authentication in Lustre is not yet available
      - » Fault tolerance in Lustre only via redundant hardware (costs!)
      - » No replication
      - » No MSS interface (yet) in the production version
      - » No straight forward procedures for OST loss/replacement
  - Adding features to Lustre quite difficult
    - » still kernel space implementation

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

Introduction

Why developing XCFS ?

- The idea behind: take the best of Lustre and take advantage of xroot plugin model for a pure xroot implementation
  - Adapt MDS schema
  - Adapt Quota schema
  - Provide MFS via FUSE!
  - Make few compromises in 'posix-ness'
  - Compromise between performance and features
    - Extra user-space copy but HA features
    - Easy to add new features
    - Easy to deploy

⇒XCFS Prototype

Architecture
Requirements

- **Simplicity**
- **High performance**
- **Scalability** (parallel FS approach)
- **Fault Tolerance**
- **Policies** (fair resource sharing/Quota)
- Access with both **RAP** & **MFS**
  - Batch data access via **RAP**
    - » **Non-posix optimizations**
  - Interactive data access via **MFS**
    - » **posix**
- Strong Authentication
  - **Krb5/ GSI / VOMS**

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

Architecture

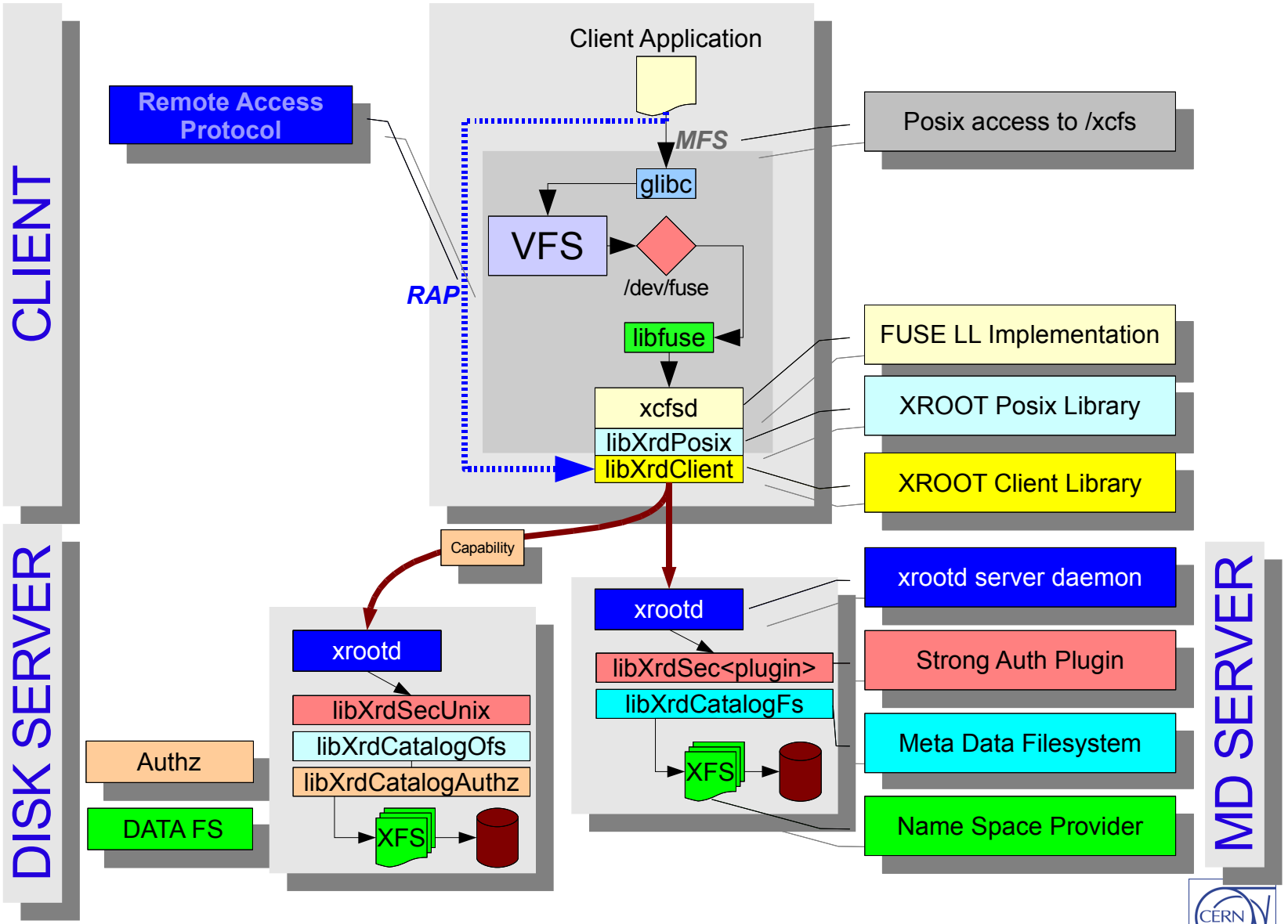
Key Elements – xrootd & FUSE

- **xrootd** client & server + xroot protocol
  - Low latency and high bandwidth data access
    - Client is distributed with ROOT
      - Standard application for HEP
    - Plugin architecture for flexible extensions
      - Protocol, Authentication,Authorization, File System, Name-To-Name Plugins
    - Protocol features (redirection, failure recovery, retry etc.)
- **FUSE** Low-Level API
  - allows filesystem daemon with multi-user support
    - Passes UID/GID for each request
    - One shared mount (using mount option *allow_other*) done by 'root'
    - Allows to set kernel caching for files individually

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

Architecture

XCFS Overview - xroot + FUSE



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

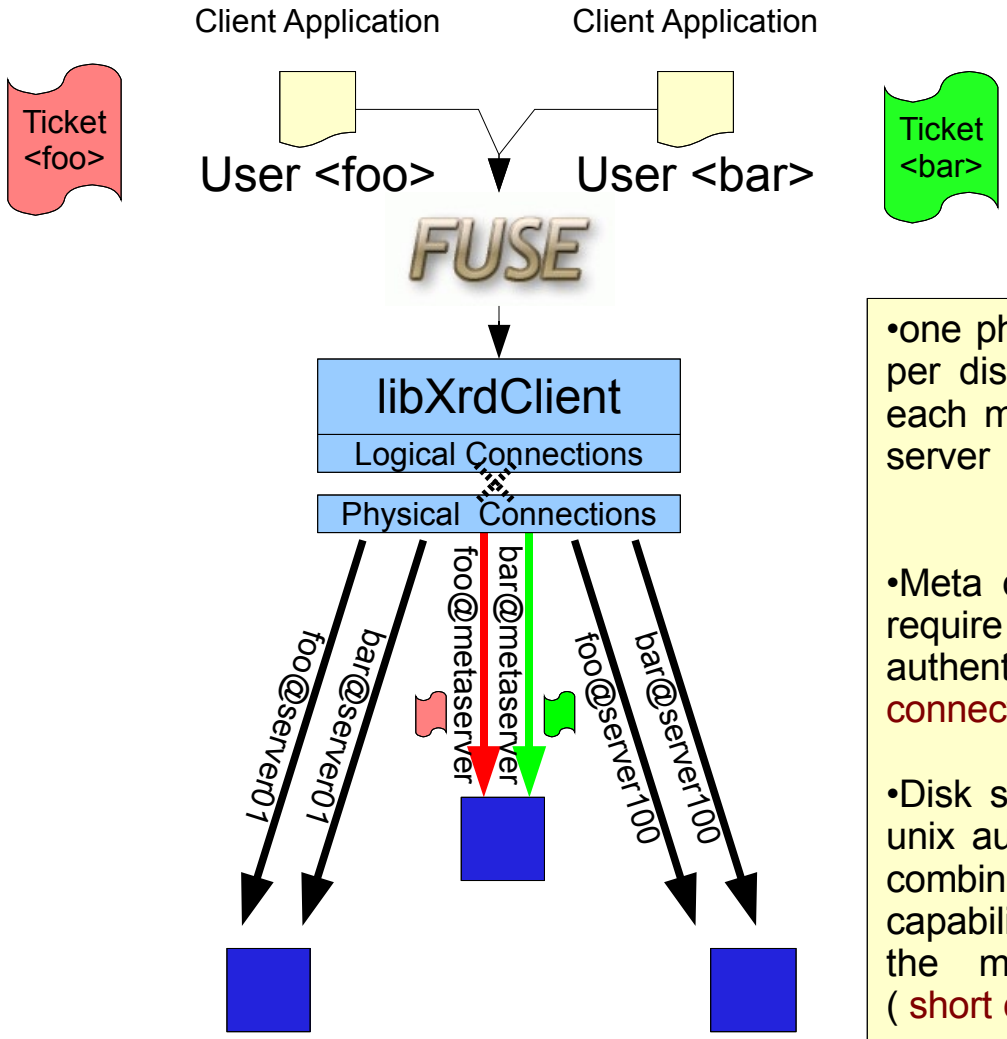
Implementation & Features

Overview

- Strong Authentication krb5/GSI+VOMS
  - Group role support
- MFS via FUSE
- RAP via xroot protocol
- Distributed Quota System
- /proc-like interface for realtime parameters
- HA features
  - Synchronous replication
  - MDS failover
- CLI for users & adminstrators
- Policy interface
  - Automatic Space Selection
  - Automatic Replica Creation
  - Automatic Garbage Collection
  - Automatic Archiving/Backup/Aggregation

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

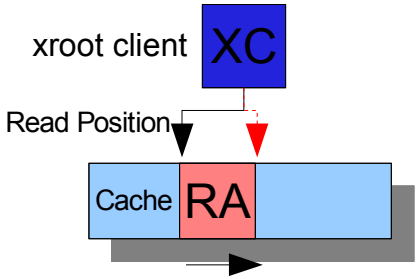
- Multi-User Handling & Authentication



- one physical connection per distinguished user to each meta data and disk server
- Meta data server require strong authentication (long connection TTL)
- Disk server require unix authentication in combination with capabilities issued by the meta data server (short connection TTL)

Implementation & Features
XROOT Client

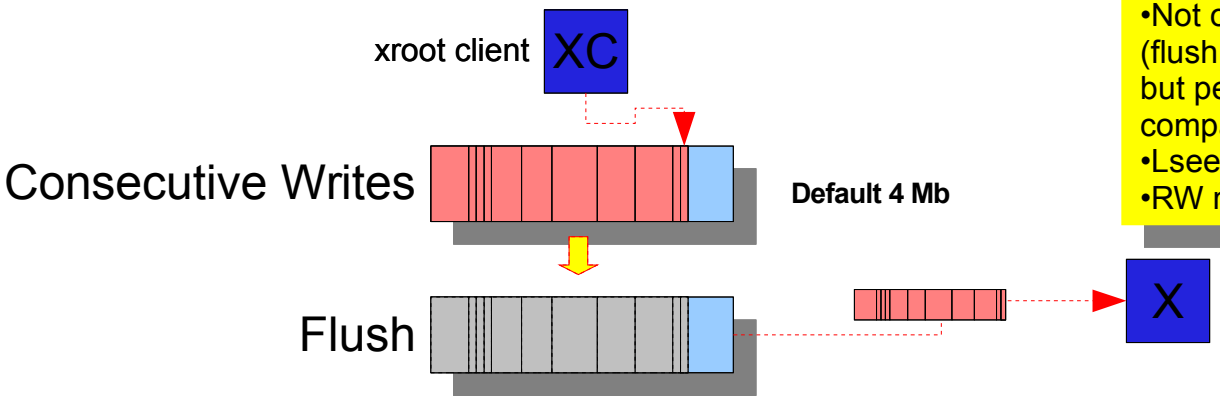
- Read-ahead & Read Cache



•Reduces latencies
•Excellent for streaming
•Less good for sparse io

* latest xrootd client uses same RA-cache for writes

- XCFS Write Cache
  - aggregates writes of 4k/64k pages



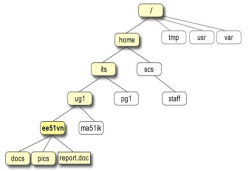
•Not optimal (flush is sync) but performance boosts compared to 4k writes
•Lseek forces flush
•RW mode disables cache

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

Implementation & Features

Namespace FS Plugin

- Namespace Provider
  - Visible namespace is stored in a file system using standard file system meta data (e.g. ownership, permissions, ACLs, Ext.Attr.) and sparse files



- The sparse files carry file attributes & size
  - the physical files are stored on disk server filesystems
- With the creation of a new file a GUID is created and tagged to the sparse file as a trusted extended attribute
- The physical location is stored as a bit into a location bitmap on the sparse file as a trusted extended attribute
- Currently only XFS was tried as namespace filesystem, EXT3 et. al. could also be used

Implementation & Features

Namespace Trusted Attributes

- **# file: datafile1**
- **trusted.fid**=0s40WWHI73Ed2yWQAVF1WDzA==
- **trusted.grouplable**="collection1"
- **trusted.loc**=0sAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
- AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA==

16 byte binary GUID

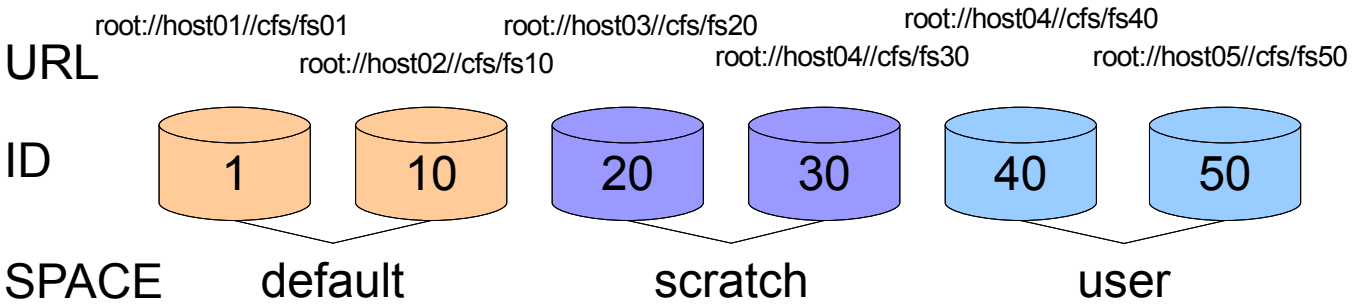
User defined or automatic aggregation lable

Binary Location Bitmap (default 8192 bits)

Implementation & Features

File Location and Space Logic

- The storage space is made up by many file systems residing on many disk servers
- Every disk server file system has a unique location bit (ID) assigned and an xroot access URL
  - File systems can be moved around hosts without problem by changing the bit/URL combination
- Filesystems can be grouped into spaces



xrdcp /tmp/higgs.root **root://mds.cern.ch//xcfs/cern.ch/higgs.root?space=scratch**

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

Implementation & Features

Inventory & Quota

- Inventory directory on MDS for each target FS storing a GUID⇒LFN link for reverse lookup
  - Used for fast consistency checks (fsck)
- Distributed Quota
  - Local XFS quota on the disk server filesystems
  - MDS keeps global quota table with the quota status for each user/group per filesystem
    - » Used for creation [file placement]
  - Local quota enforcement on target FS using standard soft/hard-limits for space and inodes
  - Configured distributed per space with admin tools

```
-------------------------------------------------------------------------------------------------------------------------------------
user:
-------------------------------------------------------------------------------------------------------------------------------------
quota: fs=/cfs/cfs1     user=ahorvath      9 used=45.81 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20907   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=airibarr      9 used=45.13 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20565   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=allcdr        9 used=45.85 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20921   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=anovais       9 used=45.90 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20925   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=apeters       9 used=42.59 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20266   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=atlonl        9 used=45.61 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20666   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=atlsprod      9 used=45.89 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20891   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=baud          9 used=46.01 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20948   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=bclement      9 used=46.16 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20923   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=bcouturi      9 used=45.92 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20762   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=ccfpro        9 used=45.85 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20894   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=ccorreia      9 used=46.15 Gb   soft=465.66 Gb  hard=558.79 Gb  files=21073   ino-soft=499998   ino-hard=600000   space=default
quota: fs=/cfs/cfs1     user=cguiller      9 used=45.84 Gb   soft=465.66 Gb  hard=558.79 Gb  files=20906   ino-soft=499998   ino-hard=600000   space=default
```

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

Implementation & Features

File System Selection

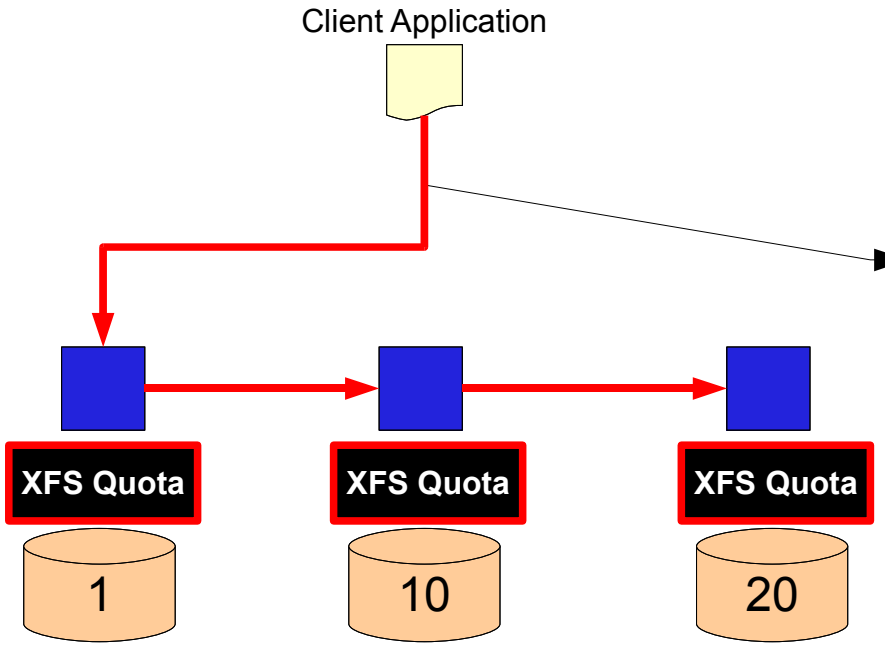
- 1st principle is round-robin scheduling over a suitable subset of filesystems
  - **read**
    - All filesystems having the location bit set can be selected in round-robin fashion
  - **write**
    - All filesystems which have space within the soft quota for the client (either user or group quota)
    - At 80%(default) fillling status filesystems are weighted according to their filling status for selection
- The client can always enforce a certain space or filesystem for a request
  - » if the file or space is available there

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

Implementation & Features

Synchronous Replication

- For each file creation the client or a server policy can enforce the creation of <N> replicas per file
- Replicas are generated and written synchronously in the following manner



Performance Write:

N<replica>	MFS [Mb/s]	RAP
1	60	100
2	48	50
3	26	33
4	21	25

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

Implementation & Features

Replica File Information

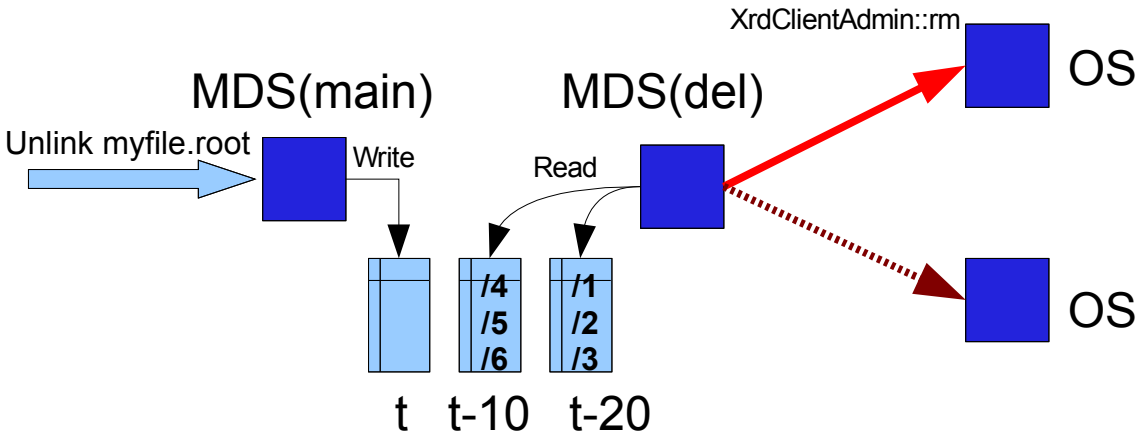
- xcfs fileinfo

```
[root@l          ~]# xcfs fileinfo /xcfs/cern.ch/ha/higgs1.root
==================================================================================
  File: `/cfs/cfs1/namespace/xcfs/cern.ch/ha/higgs1.root'
  Size: 34603008        Blocks: 8          IO Block: 4096   regular file
Device: 9301h/37633d    Inode: 50357150    Links: 1
Access: (0700/-rwx------)  Uid: (  755/ apeters)   Gid: ( 1338/      vl)
Access: 2008-10-01 16:37:09.269028735 +0200
Modify: 2008-10-01 16:37:12.672153772 +0200
Change: 2008-10-01 16:37:12.672153772 +0200
==================================================================================
File ID                  : 6d667626-8fc6-11dd-ae6a-0015175583cc
Location                 : 0x00041000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
NREP=01 ID=10   SPACE=default      URL=root://lxfsrc2802.cern.ch:1094//cfs/fs10 [drained]
NREP=02 ID=20   SPACE=default      URL=root://lxfsrc2803.cern.ch:1094//cfs/fs20
```

Implementation & Features

File Deletion - avoid blocking operations...

- Namespace deletions are synchronous
- Physical removals are delayed by 10s
  - Deletions are written to a spool file on the MDS and asynchronously executed by a deletion thread on the remote disk servers
    - As long as a disk server is down, deletion is rescheduled 300s later again and again
    - Physical file deletion is very fast in XFS

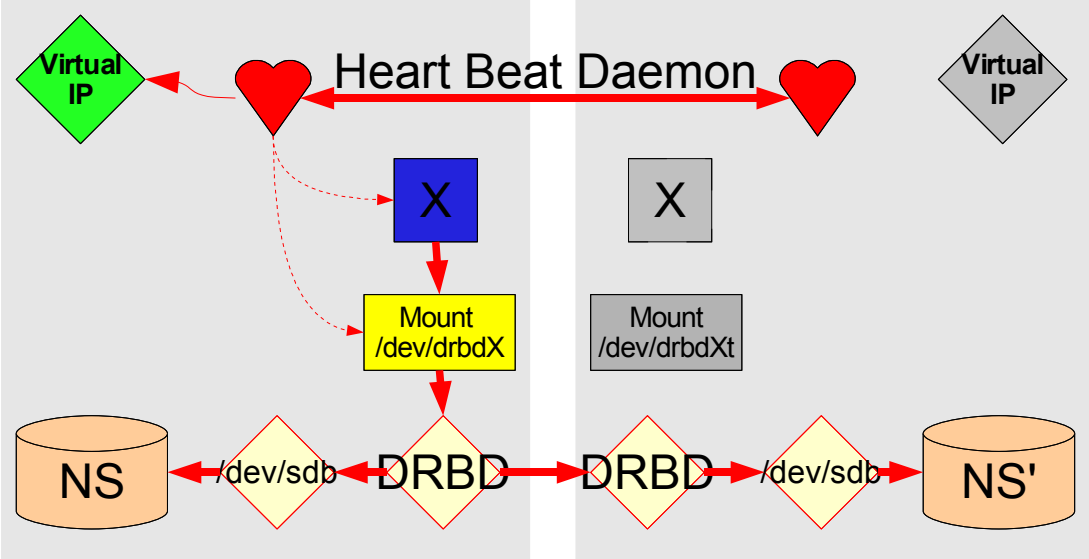


CERN - IT Department
CH-1211 Genève 23
Switzerland
www.cern.ch/it

Implementation & Features

Meta Data Redundancy - How do we get it?

- MDS is single point of failure
  - HA setup provides realtime copy and allows service failover within few seconds



  - Very easy to setup
  - Alternatives or additionally:
    - Mount NS via LVM and do regular snapshots
    - Introduce *changelog* protocol (like MySQL replication or others) – would allow read-only replicas of MDS

Implementation & Features

XCFS Command Line Interface

- **User Commands**
  - xcfs
    - ls,stat,find,getfacl,setfacl,mkdir, rmdir,rm,chown,chmod
  - xcfs cp
  - xcfs register
  - xcfs listquota
  - xcfs filesystems
  - xcfs whoami
  - xcfs fileinfo
  - xcfs dropreplica
  - xcfs shrinkreplica
  - xcfs addreplica
  - xcfs stagein
  - xcfs stageout
  - xcfs mount
  - xcfs umount
- **Administrator Commands**
  - xcfs filelist
  - xcfs filesystems
  - xcfs quota get
  - xcfs quota set
  - xcfs quota list
  - xcfs register
  - xcfs fsck
  - xcfs fileinfo
  - xcfs drainfs
  - xcfs undrainfs
  - xcfs enablefs
  - xcfs disablefs
  - xcfs freefs

Performance Measurements

Prototype Instance

CERN IT Department

Clients 80 x

8 core
16 GB
2.3 Ghz
Xeon
GBit

MDS 2 x

8 core
16 GB
2.3 Ghz
Xeon
Gbit
RAID10
(4x500Gb)

- Namespace 1 **TB** formatted for:
  4 x 200 Mio inodes ≃
  4 x 60 Mio files (1 replica)
  ≃ XCFS uses **~4k / file**

OST/
DiskServer 10 x

8 core
8 GB
2.3 Ghz
Xeon
Gbit
RAID5
(20x500Gb-2xspare)
SW RAID0 over 4xRAID5

default
readahead
parameters

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

Performance Measurements

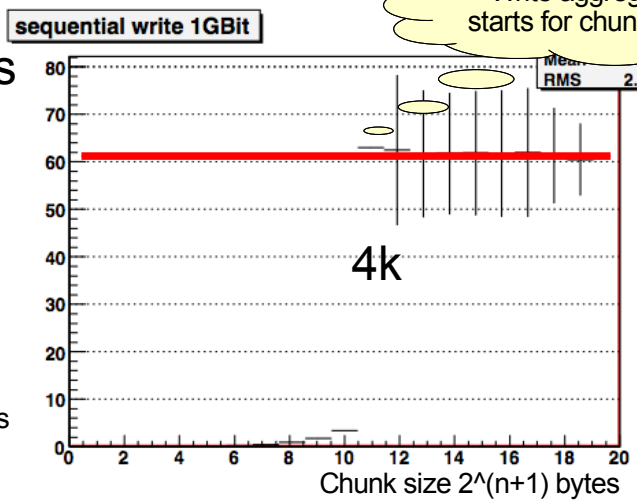
XCFS client sequential read/write 1Gbit ethernet

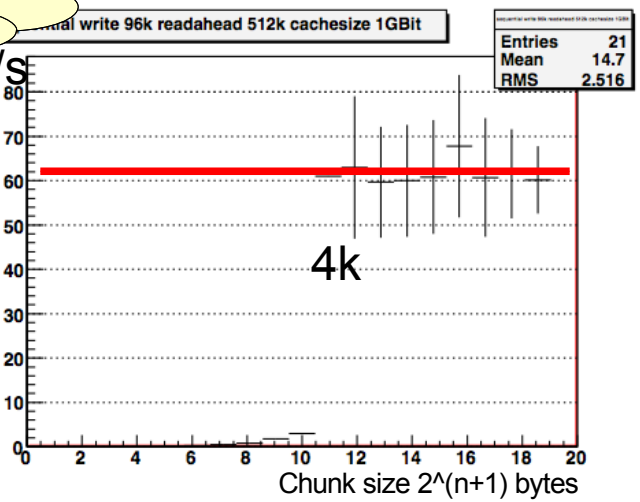
CERN IT Department

Write aggregation starts for chunks > 4k

WRITE

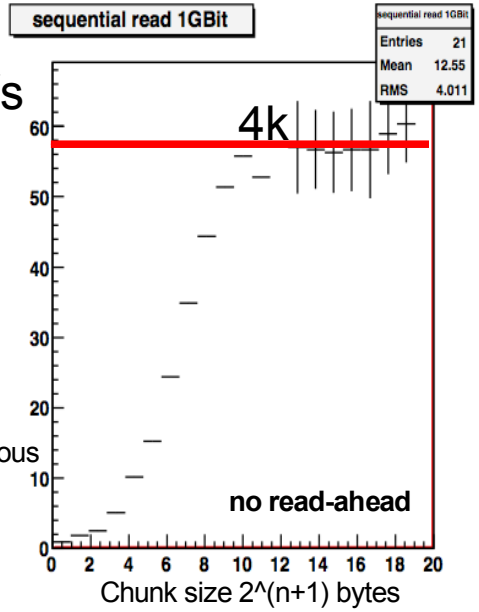
Buffered/ synchronous

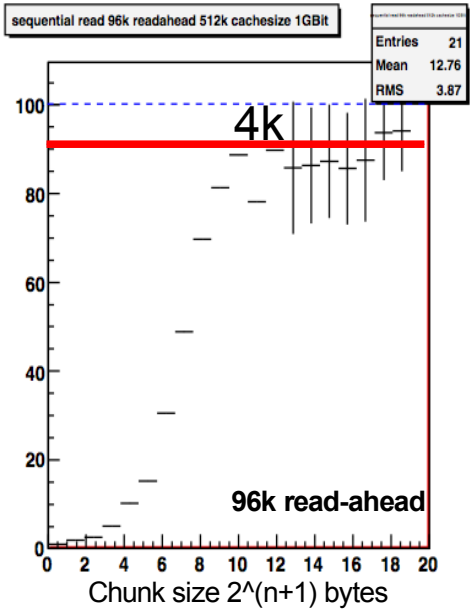


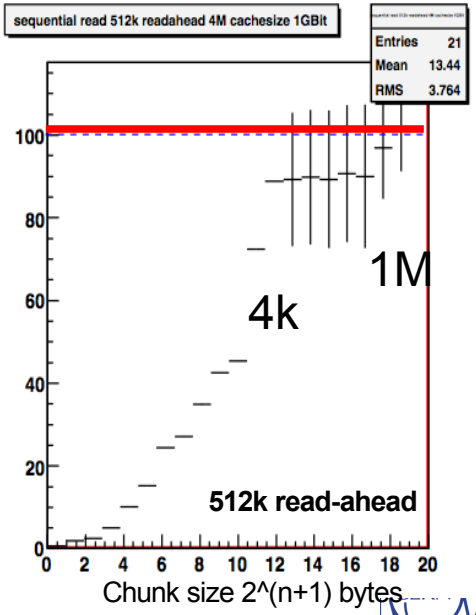


READ

Buffered/ asynchronous







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

Performance Measurements

Sparse Access

- Evaluation of a typical analysis scenario
  - ROOT h1 analysis macro [http://root.cern.ch]
    - 1 file 267Mb – 10.5 Mb have effectivly to be read to run analysis macro (~ 4% of the file)
  - Try to understand client differences
    - file is always cached in memory on server side
  - Measure performance with file cached/uncached on client side if available for the protocol
  - Comparison with Lustre, AFS, NFS4!

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

Performance Measurements

Sparse Access

	RAP xroot (rv)	RAP xroot (as)	MFS xcfs (0k)	MFS xcfs (96/512k)	AFS	Lustre (40M)	Lustre (0k)	NFS4
Empty client cache								
Real/Cpu sec	3.4 / 3.0	3.3 / 3.5	5.7 / 3.4	6.0 / 3.42	11.2 / 3.1	5.6 / 3.4	28.4 / 3.1	5.8/ 3.4
MB on wire	12.5 Mb	12.5 Mb	25 Mb	267 Mb	275 Mb	277 Mb	25 Mb	25 Mb
Filled client cache (2nd execution)								
Real/Cpu sec	3.4 / 2.9	3.3 / 3.5	2.9 / 2.9	2.9 / 2.9	2.9 / 2.9	3.0 / 3.0	2.9 / 2.9	3.2 / 3.2
MB on wire	12.5 Mb	12.5 Mb	0 Mb	0 Mb	0 Mb	0 Mb	0 Mb	0 Mb

No big differences. XCFS is competetive.
Main difference between solutions is network I/O!

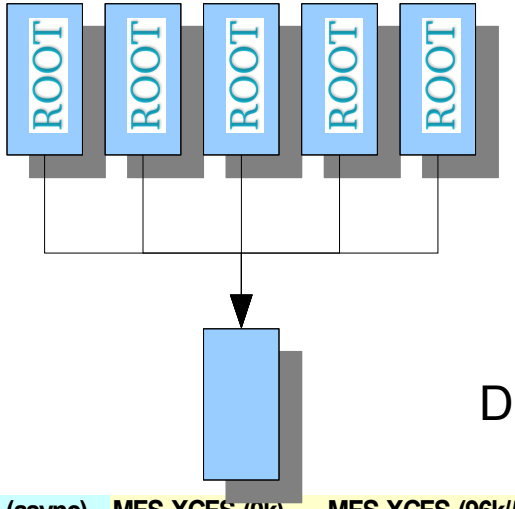
- (...) = read ahead size / cache size
  σ < 0.2 s for time measurements,

- **Considerations**
  - Network traffic is an important factor – it has to match the ratio IO(CPU Server) / IO(Disk Server)
  - Lustre doesn't disable readahead during forward-seeking access and transfers the complete file if reads are found in the buffer cache (readahead window starts with 1M and scales upto 40 M)
  - XCFS/LUSTRE/NFS4 network volume without read-ahead is based on 4k pages in Linux
    - Most of the requests are not page aligned and result in additional page to be transfered (avg. read size 4k)
  - 2nd execution plays no real role for analysis since datasets are usually bigger than client buffer cache

Performance Measurements

Sparse Access

- 5 clients per disk server analysing 4% of a chain with 9 files like in the previous test (9*267Mb*5 = 11.7 Gb total data volume)



5 analysis applications on 5 client machines

Disk server (uncached data)

	RAP xroot	RAP xroot (async)	MFS XCFS (0k)	MFS XCFS (96k/512k)	MFS XCFS(512k/	Lustre (40M)	Lustre (0k)
files not buffer cached on disk server / 1 client process per client machine							
Real /CPU sec	**131 s / 37s**	138 s / 37 s	150 s / 28 s	139 s / 28 s	134 s / 28 s	**134 s / 31 s**	324 s / 29 s
RAID Util.	60% - **11 Mb/s**	60% - 11 Mb/s	55% - 8 Mb/s	60 % - 105 Mb/s	60 % - **110 Mb/s**	60 % - **110 Mb/s**	

Again no big differences. XCFS is competetive.
Main difference between solutions is network I/O!

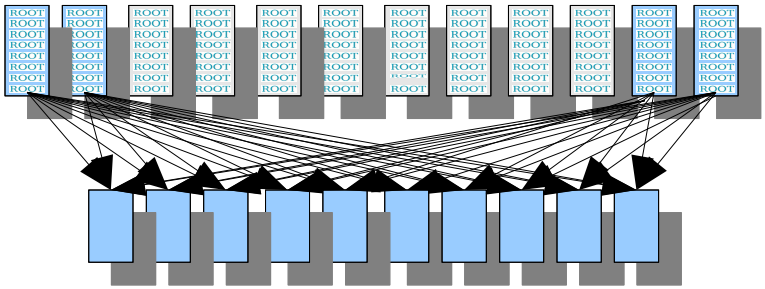
Running 8 applications on a single client box against a single disk server reveals a difference between **RAP** & **MFS**

72 files are analyzed via **RAP xroot** vs 40 files with **MFS** (XCFS or Lustre) in the same time window (Gbit saturation)

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

Performance Measurements

- Full picture: **many applications per client box** (>=1/core)
  - round-robin access creates with certain probability **accidental overload** of a single machine
    - Dangerous for small **N(Clients)/DiskServer –** averages out well for many clients
  - Favours approach **not** to use any **client read-ahead** for forward-seeking sparse access
    - read-ahead on server side is good for <u>our</u> hardware since io(Network) = io(HD) / 7
      - side remark: the xrootd-lustre plugin implemented it by design, but not XCFS



Result Analysis:

- **No** huge **differences between protocols** – difference based on readahead!
- **RAP smallest network I/O volume** but more seeking on server side
  - tradeoff – total throughput can but mustn't be better
- **Results** are **sensitive to data format and sparseness** of reads
- **Lustre** often wins in performance comparisons with linear access
  - main reason: aggressive and large readahead

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

Performance Measurements

File Streaming

- Some analysis applications do effectivly streaming file access of entire files
  (e.g. tested an example LHCb and CMS analysis)

- For file streaming there is only a %-level performance difference in the output rate per disk server between xroot (**115** Mb/s) and Lustre (**117** Mb/s) if large read-ahead is used for both.

xroot 10Gb server
118 Mb/s single 1Gb client
from server disks -
20 clients **1050 Mb/s** from
buffer cache

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

Performance Measurements

XCFS Meta Data - I/O

CERN IT Department

	RAP xroot	MFS XCFS	
30 write streams per disk server (267 Mb per file) – blocksize 1M			
Write rate per disk server	112 Mb/s	112 Mb/s	
30 read streams per disk server – blocksize 1M			
Read rate per disk server	110 Mb/s	110 Mb/s	
Sequential ROOT raw file creations/s (single client)	175 files/s	85 files/s	Latency 5.7/11.7 ms
Sequential ROOT raw file read opens/s (single client)	385 files/s	90 files/s	Latency **2.8**/11.0 ms
Parallel ROOT raw file read open/s (80 clients)	**2500** files/s	2100 files/s	Latency 52//62.0 ms
Parallel ROOT raw file creation/s (80 clients)	**250** files/s	180 files/s	With DRBD-C
Stat Performance (80 clients)		30.000 stats/s	500% CPU
Stat Performance (1 client)		1.000 stats/s	Latency **1.0 ms**
Cmd Performance (readdir + stat)		55.000 cmds/s	600% CPU

- MDS can **cache ~ 4 Mio** files of the namespace in memory (16 GB)
  - Actively used namespace expected to be even less
    - Most operations read from buffer cache
- File Creation performance **limited by DRBD & RAID-10** (4 disks)
- tested with **748 file systems** and **5000 clients** / **80** different **user identities**
  - works – but request timeouts to be tuned for many client connections
- Tests with iozone, configure/make, tar, random shell function tests etc. ok
  - XCFS doesn't reach the response time during compilation or software startup like AFS or Lustre because of missing stat optimizations
  - Still many (exotic) tests in the posix test suite fail (f.e. chown currently deactivated, no xattrs)

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

Filesystem Comparison

Basic Features

	XCFS	Lustre (1.6.5)	NFS4.0	AFS
Unix Auth	√	√	√	-
Krb5 Auth	√	-	√	√
GSI Auth	√	-	(√)	-
VOMS Auth	√	-	-	-
Kernel Module Client	√	√	√	√
Userspace Impl. Client	√	-	-	-
Kernel Impl. Server	-	√	√	√
Round Robin Distribution	√	√	-	-
Dynamic Space Assignment	√	√	-	-
File Redundancy	SW/sync	HW/sync/offline	HW/sync	SW/async/ro
File Replication	SW/sync	-	-	-
Stripe Mode	-	√	-	-
Quota System	√	√	√	√
Userspace Client API	√	(√)	-	-
Mac OSX Client	√	-	√	√
Windows Client	-	-	√	√

Mass Storage Integration
What is needed?

- Requires call-out mechanims for
  - Staging
  - Migration
- Requires bandwidth prioritization
  - tape streams need to get most of disk server bandwidth once they run
- Options/Question of Namespace
  - Shared Namespace with MSS
    - Expect meta data slow down – but global view
  - Separate Namespace with MSS
    - Good for performance – only local view
    - Permission synchronization? Master-Slave or Master-Master model?

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

XCFS Prototype Status

- **Code Size small**
  - Server Plugins 13.000 lines
  - FUSE Plugin 2.000 lines / CLI 1.500 lines
- **Few instabilities** in **MFS** client found
  - based on race conditions
- **Security** (Credential) Handling needs **consolidation** in XrdClient
  - currently creates copy of client credentials to authenticate + global mutex
- **Update** to 10/2008 version of xrootd
  - provides caching of writes in XrdClient
  - Avoids patch for SFS_PLUGIN calls
  - 1st try to port XCFS to this version didn't work
    - Problems with client message queue
- **Misses** Documentation

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

Summary

- **xroot** as **RAP**
  - has simple deployment model, allows flexibility and gives excellent performance for sparse and sequential file access for the tested use cases/hardware
- **xroot** via **FUSE** as **MFS**
  - performance for stream access
    - excellent read (100%) i/o
    - reasonable write (60%) i/o
  - most user friendly user interface
  - FUSE performance is better than expected
- **XCFS** framework provides
  - Strong authentication
  - HA features (replication + failover)
  - Namespace performance
  - Policies

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

Outlook

DM Strategy – Outcome of prototype studies

- **RAP** & **MFS** seems very good approach to satisfy broad user (analysis) requirements
  - **xroot as solution for RAP**
    - ! Already in production in CASTOR2 v2.1.8 T3 pools
      - Part of XCFS was (re-)used to do new xrootd implementation for CASTOR2 **'xCastor2'**
  - **MFS still under consideration and open questions**
    - Which one to use ?
    - Integrate or seperate ?
    - Affordable for MSS ?
    - How to deploy ?
    - When ?
-

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

Outlook

DM Strategy – Usage of XCFS prototype in MSS

- Try modified **XCFS** as base for CERN MSS CASTOR2 diskpool implementation
  - Most of the current stager functionality is already part of **XCFS**
    - Stager would be only responsible for file im- and export to/from the pool
    - File access unscheduled – only per diskserver prioritization for migration/staging streams
  - Needs evalutation of best namespace model
    - Where do we see tape files?
  - Prototype test in 2009
  - Follow generic implementation approach
    - Adapt the stager model to support any parallel FS

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

The End!

Thank you for your attention!

Questions or Comments?

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