

# AF DATASET STAGING STATUS AND NEWS

**Dario Berzano** (*INFN & Università - Torino*)

ALICE Offline Week - Genève, 14 jul 2011

# Dataset staging: résumé

# REGISTERING A DATASET

## PROOF PROMPT

/alice/data/ds001

/default/user1/ds004

*Macro, instructions and tips:*  
<http://aaf.cern.ch/node/160>

## DATASET REPOSITORY

/alice/data/ds002

/alice/sim/ds003

/ITS/user2/ds005

/PWG3/user/ds006

- Any user can register datasets using the CreateDataSetFromAliEn() macro
- Datasets are created with files available from the AliEn catalog
- No need to create an intermediate AliEn collection or XML

# REGISTERING A DATASET

## PROOF PROMPT

```
pt [0] TProof::Open("...")  
pt [1] .x CreateDatas
```

*Macro, instructions and tips:*  
<http://aaf.cern.ch/node/160>

## DATASET REPOSITORY

/alice/data/ds001

/alice/data/ds002

/alice/sim/ds003

/default/user1/ds004

/ITS/user2/ds005

/PWG3/user/ds006

- Any user can register datasets using the CreateDataSetFromAliEn() macro
- Datasets are created with files available from the AliEn catalog
- No need to create an intermediate AliEn collection or XML

# REGISTERING A DATASET

## PROOF PROMPT

```
pt [0] TProof::Open("...")  
pt [1] .x CreateDatas
```

*Macro, instructions and tips:*  
<http://aaf.cern.ch/node/160>

## DATASET REPOSITORY

/alice/data/ds001

/alice/data/ds002

/alice/sim/ds003

/default/user1/ds004

/ITS/user2/ds005

/PWG3/user/ds006

## CREATING A DATASET FROM ALIEN:TIPS

- Files to be added chosen with a syntax identical to the **AliEn find** command
- Before committing unwanted results, **preview your search** by using the options:
  - **dryrun** → do not save dataset on PROOF
  - **aliencmd** → output the command to paste into an AliEn shell to view the files
- New **update** option → adds only new files to an existing dataset

# ASYNCHRONOUS STAGING OF FILES

## DATASET REPOSITORY

/alice/data/ds001

/alice/data/ds002

/alice/sim/ds003

/default/user1/ds004

/ITS/user2/ds005

/PWG3/user/ds006

## STAGING QUEUE

alien://.../file1.root

alien://.../file2.root

alien://.../file3.root

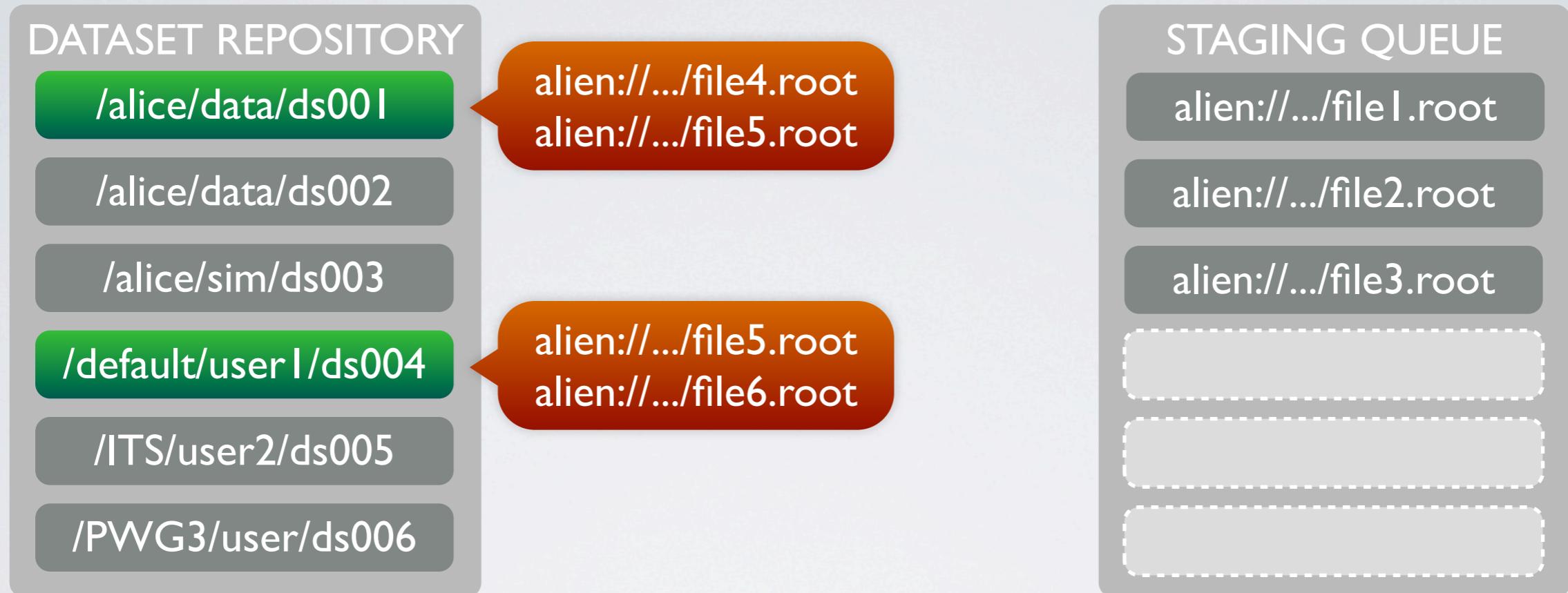


# ASYNCHRONOUS STAGING OF FILES



- The dataset manager daemon continuously scans PROOF datasets in background

# ASYNCHRONOUS STAGING OF FILES



- The dataset manager daemon continuously scans PROOF datasets in background
- Some new files marked as non-staged and non-corrupted are found

# ASYNCHRONOUS STAGING OF FILES



- The dataset manager daemon continuously scans PROOF datasets in background
- Some new files marked as non-staged and non-corrupted are found
- They are placed only once at the end of the daemon's internal staging FIFO queue

# FILE TRANSFER AND ERRORS

now staging {

## STAGING QUEUE

alien://.../file1.root

alien://.../file2.root

alien://.../file3.root

alien://.../file4.root

alien://.../file5.root

alien://.../file6.root

# FILE TRANSFER AND ERRORS

now staging {

## STAGING QUEUE

alien://.../file1.root

alien://.../file2.root

alien://.../file3.root

alien://.../file4.root

alien://.../file5.root

alien://.../file6.root

- Several (configurable) files at once can be transferred

# FILE TRANSFER AND ERRORS

staging finished

## STAGING QUEUE

alien://.../file1.root

alien://.../file2.root

alien://.../file3.root

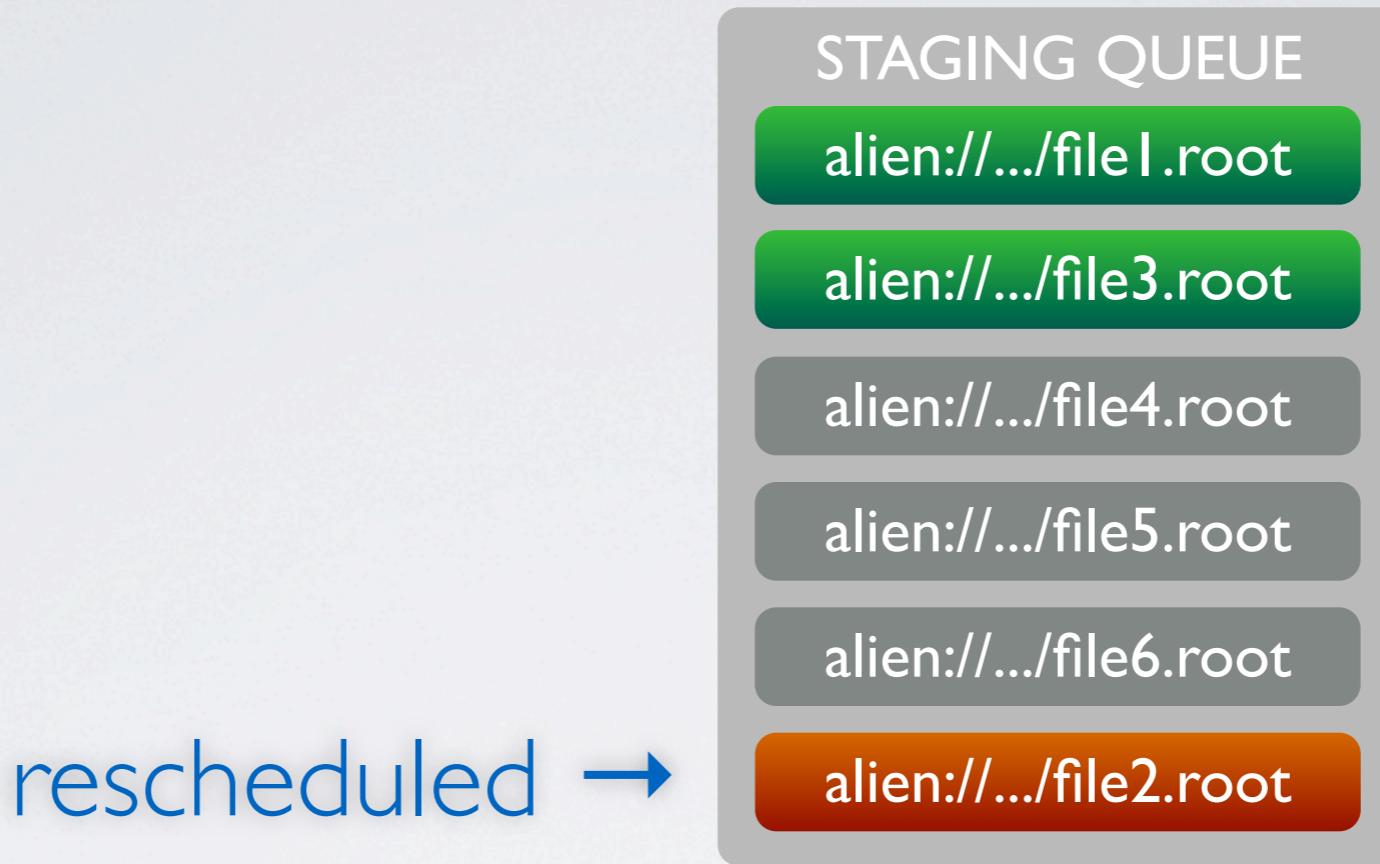
alien://.../file4.root

alien://.../file5.root

alien://.../file6.root

- Several (configurable) files at once can be transferred
- Some transfers are **successful**, others might **fail** instead

# FILE TRANSFER AND ERRORS



- Several (configurable) files at once can be transferred
- Some transfers are **successful**, others might **fail** instead
- When a transfer fails, it is **rescheduled at the end of the queue** many times, and it is considered «corrupted» for good after a (configurable) threshold

# STAGING PROGRESS AND ANALYSIS

Dataset_URI	# Files	Default tree	# Events	Disk	Staged
/alice/data/LHC10h_000139441_p2	3914	/esdTree	7.743e+051	1 TB	82 %
/alice/data/LHC10h_000139441_p2_A0D049	370	/aodTree	7.371e+051	278 GB	99 %
/alice/data/LHC10h_000139465_p2	5098	/esdTree	6.368e+051	1 TB	59 %
/alice/data/LHC10h_000139465_p2_A0D049	518	/aodTree	1.238e+061	403 GB	99 %
/alice/data/LHC10h_000139466_p2	2648	/esdTree	4.527e+051	825 GB	78 %
/alice/data/LHC10h_000139466_p2_A0D049	286	/aodTree	6.426e+051	204 GB	99 %
/alice/data/LHC10h_000139467_p2	2581	/esdTree	5.266e+051	792 GB	92 %
/alice/data/LHC10h_000139467_p2_A0D049	273	/aodTree	6.151e+051	195 GB	99 %
/alice/data/LHC10h_000139470_p2	804	/esdTree	1.804e+051	251 GB	98 %
/alice/data/LHC10h_000139470_p2_A0D049	92	/aodTree	1.72e+051	54 GB	100 %
/alice/data/LHC10h_000139471_p2	1444	/esdTree	3.327e+051	441 GB	99 %
/alice/data/LHC10h_000139471_p2_A0D049	154	/aodTree	2.609e+051	93 GB	100 %

- Everybody can check the **staging progress** from within a PROOF session
- Status can also be checked (for CAF only) **on the web:**  
→ <http://alimonitor.cern.ch/stats?page=CAF2/datasets>
- **Partially-staged** datasets can already be processed

The staging daemon

# AFDSMGRD AND ROOT INTEGRATION

# AFDSMGRD AND ROOT INTEGRATION

- Daemon (namely: **afdsmgrd**) in production on all AAFs since April 2010
- Standalone version available on SVN → <http://afdsmgrd.googlecode.com/>
- Original C++ code (up to v0.4.3) **completely rewritten** to improve performances
- **v0.9.3** is currently in production since Jun 30 → <http://aaf.cern.ch/node/203>

# AFDSMGRD AND ROOT INTEGRATION

- Daemon (namely: **afdsmgrd**) in production on all AAFs since April 2010
- Standalone version available on SVN → <http://afdsmgrd.googlecode.com/>
- Original C++ code (up to v0.4.3) **completely rewritten** to improve performances
- **v0.9.3** is currently in production since Jun 30 → <http://aaf.cern.ch/node/203>
- **The daemon is currently being distributed as part of ROOT v5.30.00 (and backported to v5.28.00e)**

will make sure that the files 'thisheader.h' and 'thatheader.h', needed by 'mymacro.C' are available in the sandbox on the worker machines. Note that 'thisheader.h' and 'thatheader.h' will be available remotely in the sandbox, as 'mymacro.C'; so they should be included directly by 'mymacro.C', e.g. '#include "thisheader.h"'.

- Import the dataset stager daemon 'afdsmgrd' into ROOT; this is used to manage data staging based on the dataset information (see <http://code.google.com/p/afdsmgrd/> for more info). The daemon is located under \$ROOTSYS/proof/afdsmgrd .
- New PROOF bench suite, a framework to run CPU and IO benchmarks with default selectors/data or with user-provided ones. The code is located under proof/proofbench. See <http://root.cern.ch/drupal/content/new-benchmark-framework-tproofbench> .
- Add the possibility to access the files on the workers via the same port used by PROOF. This is useful for cases when it is not possible to start a file server daemon on a different port (because, for example, of a firewall or just inconvenience) and workers

**ROOT v5.30.00 release notes** → <http://root.cern.ch/root/html530/notes/release-notes.html>

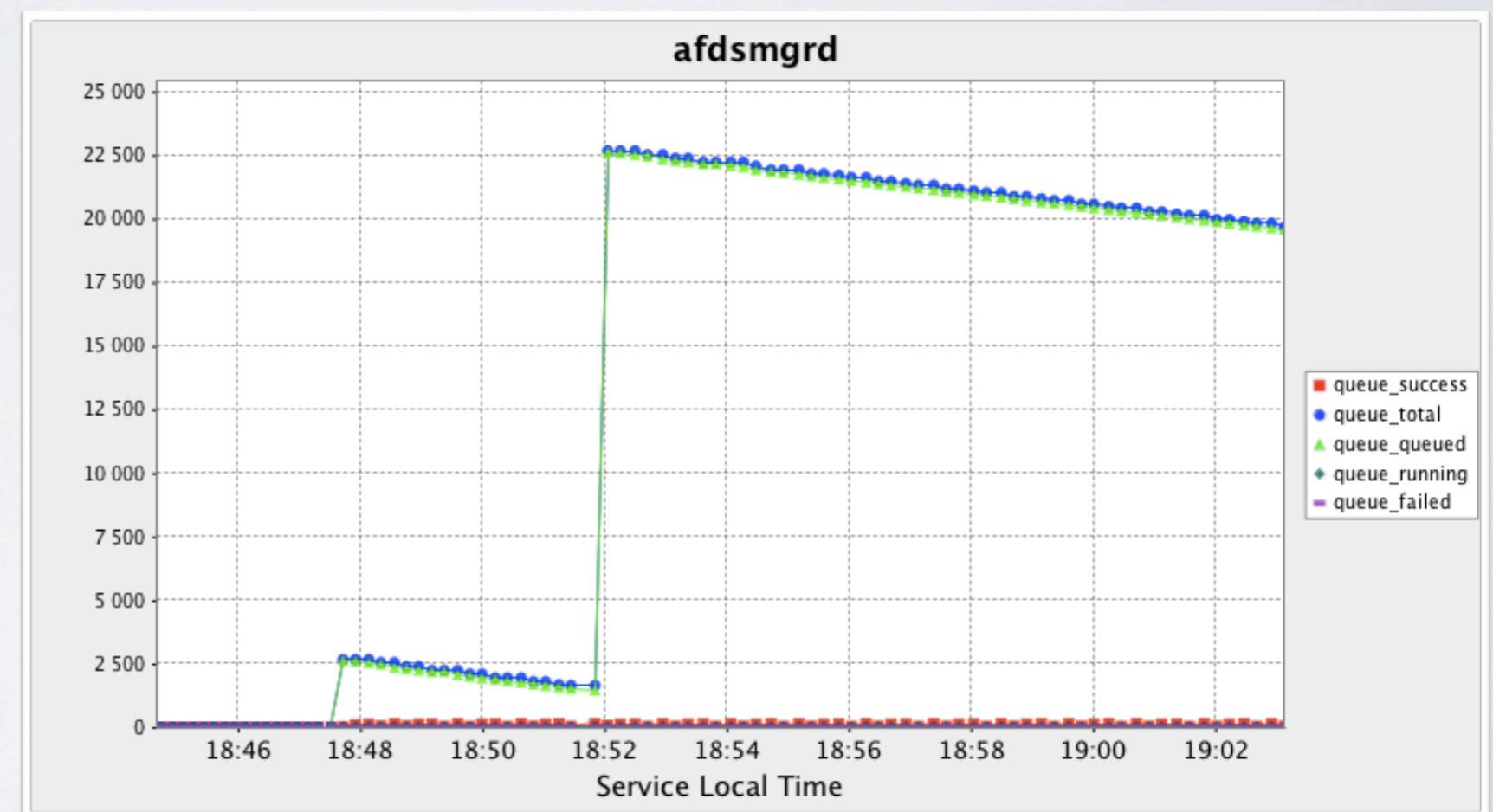
# NEW FEATURES/I

- ROOT threads are no longer used: staging and verification are performed in **separate Unix processes**
  - addresses some *ROOT thread-safety issues difficult to debug* (many ROOT classes are not confirmed to be thread-safe)
- File **integrity check** has been moved to **external processes**
  - ROOT might cause hangups or crashes in some cases if opening a corrupted file: this led to daemon crashes too. Now, ***if the external verification process dies, the daemon does not***, and the abnormal termination is interpreted as «file corrupted»

# NEW FEATURES/2

- No daemon restart to change configuration: **dynamic reconfiguration**
  - the daemon automatically reloads the config file if it changed on disk, and configuration errors are handled gracefully without stopping the daemon
- No daemon restart to flush locked external commands: **timeout**
  - the staging command is killed after a configurable timeout
- Plugin-based monitoring with staging queue, dataset stats and used resources
  - no longer limited to MonALISA (ApMon)

MonALISA realtime monitoring in action



# PERFORMANCE AND STABILITY IMPROVEMENTS

- Internal queue is a **temporary SQLite table**
  - SQLite manages memory swapping very robustly:  
daemon tested with *over a million files* in queue
- File **integrity check** speed improvement, now performed **in parallel**
  - back to v0.4.3 only staging was parallel, while verification was serialized and made all the other transfers to wait if the file was temporarily unavailable



```
# /etc/init.d/afdsmgrd log
I-[20110713-185303] Total elements in queue: 22372 | Queued: 22312 | Downloading: 30 | Success: 30 | Failed: 0
I-[20110713-185303] Not processing datasets now: 4 loops remaining before processing
I-[20110713-185303] Usage statistics: uptime: 4667548.9 s CPU: 3.1%, avg CPU: 3.1%, virt: 454804 KiB, rss: 362704 KiB
I-[20110713-185303] Sleeping 5 seconds
```

actively running for  
**54 days** (and counting):  
**very stable**

~ $\frac{1}{3}$  GiB of **constant**  
resident memory footprint  
(incl. ROOT libs): **no leaks**

# HOW INTEGRITY CHECK WORKS

- Files verification aims to detect corrupted files before running the analysis
- Analyses **might crash** sometimes if running on corrupted files
- Verification is performed by a ROOT macro ran in a **dedicated ROOT session**

## STEPS TO VERIFY THE FILE

- File is opened: if opening fails, file is considered **corrupted**
- If no default tree was indicated when registering, the first tree found is taken
- The default tree is read: if this operation fails, file is considered **corrupted**
- If this point has been reached, **the file is considered OK**
- **If at any point ROOT crashes, the file is considered corrupted**

Dataset manipulation and verification utilities

# AFDSUTILS: DATASET MANIPULATION UTILITIES

*These functions are in AAF package **VO\_ALICE@AFDSUtils::0.9.3***

- afRepairDs()  
*actions (such as deletion) on bad files*
- afShowListOfDs()  
*shows the list of available datasets*
- afShowDsContent()  
*shows the entries of a dataset*
- afFindUrl()  
*shows which datasets have the given file*
- afRemoveUrlsFromDs()  
*remove entries from multiple datasets*
- afDataSetFromAliEn()  
*used by CreateDataSetFromAliEn() macro*
- afMarkUrlAs()  
*change staged/corrupted bit of a file*
- afResetDs()  
*revert dataset to nonstaged/noncorrupted*
- afFillMetaData()  
*scans datasets for num. of entries/trees*
- afDsToPlainText()  
*produce a text file with dataset content*

*For more functions and extensive documentation consult the source code:*

<http://afdsmgrd.googlecode.com/svn/tags/v0.9.3/macros/par/afdsutil/afdsutil.C>

# AFVERIFIER AND DATA REDISTRIBUTION

AAF data model is to propagate uniformly the files of each dataset throughout the nodes of the cluster, yet as time passes by, **data can get no longer evenly distributed**

## WHY SO?

- New PROOF nodes are added
- Disks or nodes get replaced
- Some nodes are unavailable at the time of data staging

## REDISTRIBUTION TOOLS

- **afverifier** to read datasets and retrieve the exact files location
- **pq2 tools** to move data, modified to allow node-to-node transfer without passing by the master (by G.Ganis)

afverifier in action,  
processing 800 files at once

```
0-[20110714-000405] Success: root://pmaster.to.infn.it//alice/data/2010/LHC10h/000138396/ESDs/pass1/10000138396058.60/AliESDs.root
I-[20110714-000408] Total elements in queue: 6824 | Queued: 0 | Processing: 424 | Success: 6400 | Failed: 0 | hours: 0.000000
I-[20110714-000408] Operations summary: Freshly started: 424 | Freshly finished: 800 (800 OK, 0 failed, 266.7 per second)
I-[20110714-000408] Not processing datasets now: 2 loops remaining before processing
I-[20110714-000408] Usage statistics: uptime: 285.5 s, CPU: 7.6%, avg CPU: 7.6%, virt: 128112 KiB, rss: 44216 KiB
I-[20110714-000408] Sleeping 3 seconds
I-[20110714-000411] *** Inside main loop ***
I-[20110714-000411] *** Processing operations queue ***
0-[20110714-000411] Success: root://pmaster.to.infn.it//alice/data/2010/LHC10h/000138396/ESDs/pass1/10000138396058.600/AliESDs.root
```

# AFVERIFIER AND PARALLEL FILE DELETION

*afverifier can also be used to **physically delete files contained in a dataset**: since it processes requests in parallel, it is **the fastest way to remove a dataset***

## INSIDE ROOT

- Load AFDSUtils package:

```
gProof->EnablePackage("VO_ALICE@AFDSUtils::0.9.3")
```

- Mark all the files of a dataset as corrupted:

```
afMarkUrlAs("*", "C", "/group/user/dataset_name")
```

## FROM SHELL

- Configure afverifier based on the example provided in:

```
$ROOTSYS/etc/proof/afverifier.conf.example
```

- Launch afverifier, telling it to consider only corrupted files and remove them:

```
afverifier <other_options> -x -w C
```

# UPCOMING

- Manual for afverifier
- Manual for AFDSUtils
- **Randomization of the transfer queue** instead of processing them in order of arrival (optional): *this would be useful to have at least some data to test analysis on all the newly registered datasets*
- Request restaging of a dataset through a web interface
- **Automatic dataset creation** for centralized sim/reco: **how to do it?**

Thank you!