

**(Draft)**

**PLAN**

**To realize the prototype  
for distributed computing  
on the base of Globus toolkit  
on cluster RAM at  
Stony Brook University**

Andrei E. Chevel

shevel@bnl.gov

June 07, 2002

June 18, 2002

## **Introduction**

Main idea is to realize more or less stable prototype for distributed computing for the Stony Brook University for physics analysis in real environment for the **PHENIX** (<http://www.phenix.bnl.gov/>) collaboration at Brookhaven National Laboratory (**BNL** - <http://www.bnl.gov>). This prototype could be later used in any other university or collaboration if people will find it useful.

## **Problems in remote analysis for relatively small physics group**

### **Introduction**

That is not surprising that physics group for physics analysis is relatively small (5-10 physicists including graduated students). The same could be said about almost any working group where group members responsibilities is difficult (or not possible) to formalize. In other words we could expect many relatively small and relatively independent each other groups will take participation in physics analysis for measured data in the collaboration. All of those groups will have more or less close needs in computing area. Our main aim to prepare the prototype of computing architecture could be used by many groups.

## File replication and file bookkeeping

The term *replica* is used in different context and meanings. Here we would use the term *replica* with meaning "a copy of a file (or collection of files) that is equal to original file (or original collection of files)". We will consider term *equal* as zero completion code after command **diff**.

We need to keep the Replica Catalog where the file has several names:

- Logical file name - logical name of file in spite of location. In this context we would image set of replicas in different places. This file name has one origin location.
- Physical file name - concrete location of the file, for example  
ram0.i2net.sunysb.edu:  
/Data/data02/nanodst/nan2002/ndstsv03/run27000/ndst\_CNT\_run2\_v03\_pro19-27896-018.root

Also a range of files might be included into file collection with some logical name, for instance 'ndstsv03-run27000'. We consider file collection is located in one computing cluster. That means all files in file collection are located on the same computing cluster (on direct disk volumes or in NFS mounted volumes as worth case).

It is very useful to have collection of collections of files. A collection of collections is trivial recursive operation. We just emphasize that collection of collections might be spread out among different clusters.

All the files and collections have to be kept in Replica Catalog (**RC**). **RC** might be replicated (copied into different location) and mirrored (copied in different location and kept in consistency with original RC by automatic regular checking, for instance two times a day).

In simplest case we could use one **RC**.

Several procedures could be used:

- Polling scheme - where client polls on regular base the RC about new files;
- Subscription scheme - where clients will be informed about any change in collection of files. Presumably before client sent the subscription request for concrete collection of file. Hence only those clients would be informed who sent subscription requests.

After client is informed about the change in the collection of files there are several ways how to start the replication process:

- Server can start the replication process (in our case **RCF**);
- Client can start the replication process (i.e. Stony Brook).

Part of operations with **RC** could be performed only by administrator of RC:

- Creation of RC;
- Deletion of RC;
- To setup access permissions (which trusted user account(s) has possibility to read, write files through **RC**).

In conjunction to **RC** a number of user command are required:

- **cpn** physical\_file\_name logical\_file\_name - publish (to store in **RC**) new file name; logical\_file\_name might be in the form
  - o "/CNT\_run2\_v03\_pro19-27000.root"
  - o "CNT\_run2\_v03\_pro19-27000.root"
  - o "/ndstsv03/CNT\_run2\_v03\_pro19-27000.root" - here the file is member of collection of files with name "ndstsv03";

- o `"/nanodst/ndstv03/CNT_run2_v03_pro19-27000.root"` - here `ndstv03` is name of collection of files and `"nanodst"` is name for collection of collections.
- **cdn** `logical_file_name` - delete (remove from **RC**) existing file name; logical file name might be in a range of forms as shown before;
- **ccfc** `name` - create (in **RC**) collection of files with name `"name"`;
- **cccc** `name` - create (in **RC**) collection of collection with name `"name"`;
- **cdfc** `name` - delete (from **RC**) name of collection of files;
- **cdcc name** - delete (from **RC**) name of collection of collections;

Most of previously discussed commands on **RC** could be easily mapped to existing tools from **GDMP**.

## Possible Scenario for data replication

It is possible to discuss several scenario for data replication process for the data analysis.

At first scenario the data are generated at **BNL** (for instance nanoDST). Those data are published when ready to **Replica Catalog (RC)**. All subscribed clients are notified about the data are ready to be replicated.

On client side where information about new data at BNL was received local replica service starts to copy the required portion of data. It is not bad to inform on regular base people on client side about the real status of copying (current transfer speed, average transfer speed from beginning of copy process, how much time will be required to copy remained portion of data in current data transfer).

At second scenario the data could be generated in a range of places (by some simulation program). Later those generated data have to be moved to centralized data store (**BNL**). In this scenario we have data stream directed to BNL. All other technical aspects are the same as in previous scenario.

## Data transfer

One of the underlying technical problems is bulk data transfer. It means transfer of relatively large volumes of data (in context of this project it might be about 1 TB per month). A range of techniques could be followed for the data transfer: **ssh**, **bbftp**, **bbcp**, **Gridftp**. In initial stage we plan to use data transfer technique suggested in **GDMP**.

## Job submission

It is assumed we will have several clusters where we could start up our jobs. Submission of the jobs could be done in different ways:

- To submit the job to concrete cluster (it has to be pointed out in the parameters);
- To submit the jobs without pointing to the concrete cluster. In this case the system will automatically submit the jobs to a default cluster.

We also need to some job submission command

- Job submission on the cluster (host)
  - o `jsub [cluster] [required data from RC] job_script` - to submit the job and make possible to enter next command for an user; if parameters `'required data'` and `'cluster'` have been entered the system will check though **RC** `'are those data available on the cluster?'`; if parameter `'host'` is missing and parameter `'required`

data' is entered the system will start the job on less loaded cluster. Standard output and standard error output will go to the default files like \*.o and \*.e on the cluster where job was performed.

- o jrun [cluster] [required data from RC] job\_script - to run the job and wait till the job will be completed; the same features as for 'jsub'. All standard output from the job will go to the terminal.
- Job status command
  - o jstat [cluster] [job\_number] - if no any parameter given the answer is the list of all jobs submitted from this account name to all cluster names.
- Job deletion
  - o jdel job\_id1 job\_id2 ... - delete pointed job\_ids (cancel them).
- To get job standard output files
  - o jget job\_id - get the output from the job. In principle job has as usual two standard output files (standard output and error output).

Parameter 'cluster' is considered here as fully qualified hostname for main cluster host or as alias name for main cluster host.

All of mentioned commands would be easily mapped to **Globus** toolkit commands.

## Possible scenario for job submission

Fist scenario is very simple - user starts up only job from his own laptop

```
jsub ram0.i2net.sunysb.edu job_script
```

This command will give you job\_id for the job started. The script 'job\_script' will be performed on cluster at Stony Brook. It is assumed that the script invokes a program which does exits on the host. The standard output files will be stored at the host ram0.i2net.sunysb.edu. To get the output from that host to your own laptop you could use on laptop the command

```
jget job_id
```

Output will be performed on the terminal.

Second scenario is little bit more complex. User submits from his laptop large script which in turn will start a series of jobs.

```
jsub large_job_script
```

In this scenario user has as usual some *sand box* or a set of subdirectories where jobs output is intended to be stored. That means job\_ids must be stored in appropriate place to be used later to retrieve the output of jobs.

If jobs have large output (for instance simulated DST files) it could be placed in **Replica Catalog** with commands **ccfc** (create collection of files) before generating of files and **cpn** (publish the file name; see above) immediately after the file creation.

# WWW window for the distributed architecture

Here I try only to emphasize most interesting features what could be browsed.

First of all I would mention **Replica Catalog**. It would nice to have clear view of current replica catalog content, i.e. all our data locations around the World.

Another good thing to browse is current distributed job submission status and job stages.

## Security

I do not discuss any security problems intentionally because it is considered most of security issues were solved in Globus Security Infrastructure (**GSI**).

## Current status

At **RCF** the **Globus** toolkit was installed on stargrid01.rcf.bnl.gov some time ago. Approximately same was done at Stony Brook on the cluster ram0.i2net.sunysb.edu. At Stony Brook the data transfer was tested mainly in direction from ram0.i2net.sunysb.edu to stargrid01.rcf.bnl.gov.

It seems to me now it is the time to create testing infrastructure to test Globus and related tools in real environment.

## Plan for real testing

It is suggested correlated steps which could help us to test prototype.

- To create at Stony Brook (near ram0.i2net.sunysb.edu) the Globus gateway (special Linux/Intel server for all Globus and related stuff).
- To permit for testing purposes to start up the jobs from stargrid01.rcf.bnl.gov to the standard job queues on **RCF**.
- To test at least simplest scenario for data replication and job submission in standard queues on **RCF**.