



DRAFT: COMMENTS SOLICITED

Transformation Catalog Design for GriPhyN

Prototype of Transformation Catalog Schema.

Ewa Deelman, Carl Kesselman, Gaurang Mehta

10/10/2001

1. Introduction

As part of the GriPhyN project (www.griphyn.org) we are developing a transformation catalog, which will be used to store information about the transformations which need to be used in order to process data as requested by the user. The catalog can be used to process raw or derived data products. Data derived using the transformations will be possibly placed in the Metadata catalog or the Replica Catalog[1].

The transformation catalog will be evaluated in the context of the LIGO prototype. The existing prototype is able to take an input of channel names and starting and ending time and then can query the replica catalog for the location of the requested data. If the files are not found, computation used to produce them is scheduled on LIGO's computational resources. The granularity of LIGO data frames is set to 50sec time frames. Frames can be concatenated to form larger time frames. The results are provided to the user as a URL. The newly produced 50 sec frames are then entered into the replica catalog for future reference. The concatenated file returned to the user is not reflected in the replica catalog.

Currently, the prototype uses only two transformations. By using the transformation catalog, the user will be able to ask for a rich set of derived data. This will be another step towards the realization of the concept of Virtual Data, where the user/application can request data whether it is materialized or not. If the materialized data is not available or the cost of accessing the processed data is greater than the cost of applying the transformation then we need obtain information about where the appropriate transformation is located.

This document aims to specify a schema for a transformation catalog, which holds information regarding the transformation files and their location.

2. Catalog Requirements

A major factor in deciding on the design is the need for information about the architecture and the platform of the machine where the computation is to be done. Another important factor is whether a binary is available or not for that architecture. If binaries are available the shared libraries are assumed to be included in the tarball distribution as they may be required to run the executable. We also need to provide compiler information (which compiler was used to generate the code) as well as the flags used. If the source is distributed we need to provide a makefile describing how to build the software. Factors such as cost of running or accessing the transformation as well as the minimum resource requirements have to be taken into consideration.

The transformation catalog can then be searched for the transformation with a variety of parameters, such as transformation name, architecture type, creator etc ... We believe that the catalog will be mostly accessed via frequent read/search operations and less frequent writes. Ultimately, the catalog will need to be distributed to provide efficient and fault-tolerant access from various locations on the grid. To support these requirements, we have decided to implement the Transformation Catalog using an LDAP server with a LDBM backend and support the LDAP query language for catalog access (the issue of catalog distribution still needs to be addressed.)

3. General Catalog Design

The Transformation Catalog (TC) fits into the GriPhyN catalog structure as depicted in Figure 1. TC records information about the transformations used to create derived data: executable name, arguments, etc ...

The following attributes are required:

- Transformation name.
- Program file name.
- The name of the author.
- Version number.
- Type (binary or source)
- Supported architecture and OS: the architecture and OS for which the program has been compiled.
- URL: repository(s) where program may be found, and path in the repository.
- Port number and protocol to get into the repository

In addition, we can specify the following optional attributes

- Language the program is written in.
- Compiler Flags used for optimization.
- Configuration file (name and location) containing the parameters for the program and their default values.
- The compiler used to generate the program.
- Transformation description.
- Cost of running program.
- Email of the Creator.
- CRC and file size for error checking
- Logical expression specifying the resource requirements, such as min of processors, maximum number of processors (by default we can assume a single processor system), use only a square number of processors etc...

- Valid field: since we provide only write once capability for this catalog, the entry is used to invalidate erroneous or transformations.

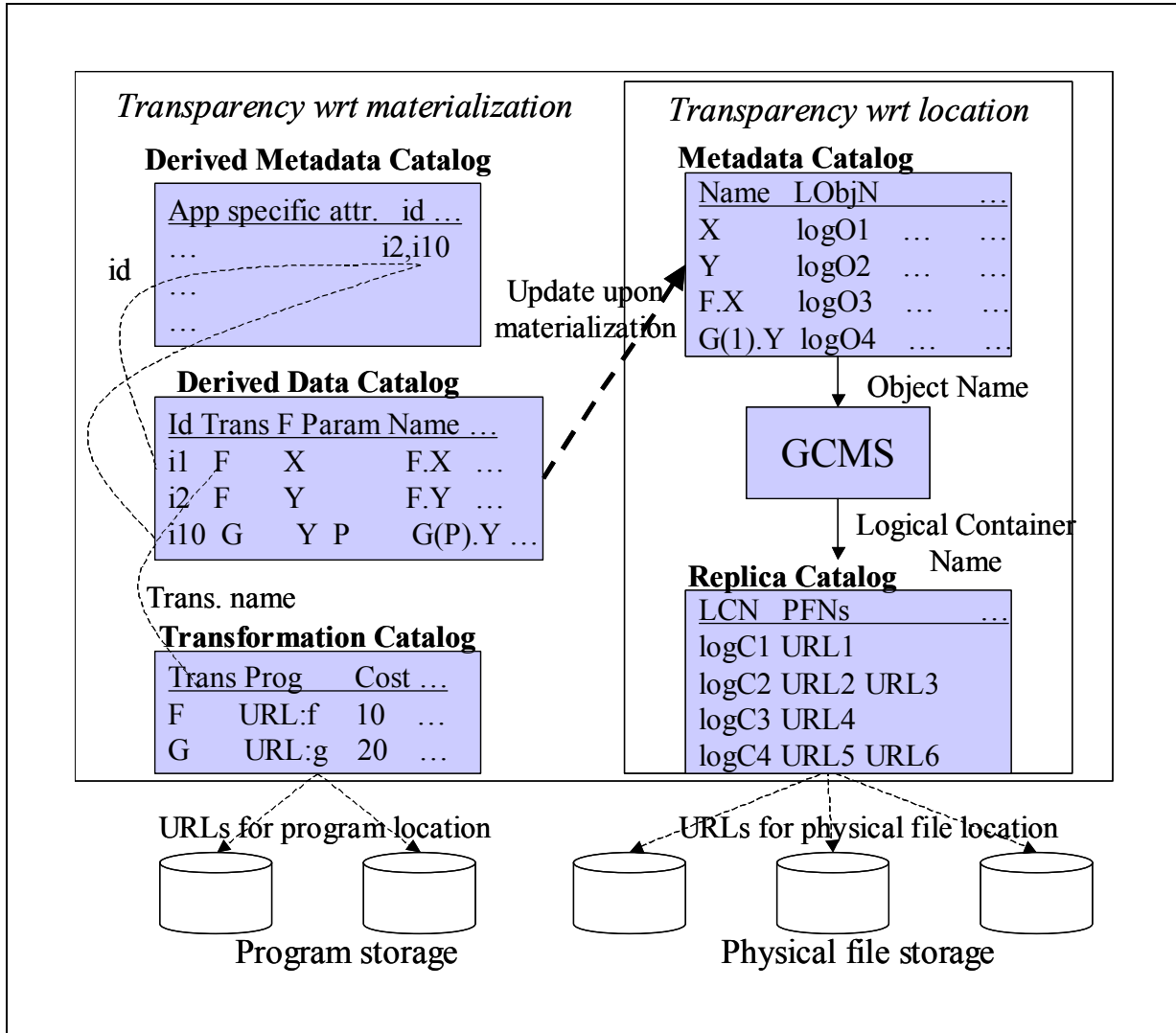


Figure 1: Catalogs in GriPhyN

4. Transformation Catalog Schema

The schema for the transformation catalog is designed to address the needs of the project as described in Section 2. The data in the catalog is being structured based on the type of transformation, the architecture of the machine for which the binary is available and the location of the executable.

We propose to create four object classes :

- | | |
|----------------------------|--------|
| 1. TRANSFORMATION CATALOG | Rdn tc |
| 2. TRANSFORMATION INFO | Rdn ti |
| 3. TRANSFORMATION ARCH | Rdn ta |
| 4. TRANSFORMATION LOCATION | Rdn tl |

The Transformation Catalog is the main root node for the catalog. The information about the transformations is stored below this root. The structure that defines the transformation is the Transformation Info (“TI”), which maps from the Id and gives details about what the transformation does. It also has information about whether the transformations available in that sub tree are available as binary distributions or source distributions or both.

The valid description field in the TI allows for the enabling or disabling of access to a particular transformation and stores the reason for the disabling the transformation. This field can be implemented in all the lower structures as well thus allowing for more control over the data in the transformation catalog.

The Transformation info is then categorized based on the architecture and platform the transformation can run on. The structure used to define that is called the Transformation Architecture (“TA”). The required attribute of the TA is the OS type. An additional optional field is provided for patch version if a particular transform requires a specific patch.

The Transformation Architecture is finally categorized into locations. This structure is called the Transformation Location (“TL”). The TL holds all the details about the location of the binary distribution or the source distribution. The paths to these distributions are given as a URL, explicitly specifying the access protocol, path and port.

Some of the assumptions we have made are that the binary distributions are available as tarballs, which contain the transformation binary, the shared objects required for executing the transformation as well as the setup script needed to configure the distribution.

The Source Distributions are similarly assumed to be available in GCVS repositories. The appropriate URL pointing to the distribution is provided.

TL also stores the creators name and the version number of the distribution. The contact info of the creator is provided as an optional attribute. Along with the distributions an attribute pointing to a security certificate is provided for validation of the distribution as well as for error checking.

We also need to provide information about the cost of running or obtaining the transformation, the minimum resources required for running this transformation, compiler used as well as optimization flags. More research needs to be done as to the best way to represent these attributes and hence have not been included in the design at this stage.

The details of each object class and their attributes are as follows:

A. TRANSFORMATION CATALOG - OBJECT CLASS

Rdn = tc

Attributes:

- Required [None]
- Optional

Creatorname	cname	dn
Createtimestamp	cstamp	numeric
Modifyname	mname	dn
Modifytimestamp	mstamp	numeric

E.g. :

```
tc          = Griphyn
dn          : tc=Griphyn
cname       : gmehta@isi.edu
cstamp      : 2001-13-07
mname       : deelman@isi.edu
mstamp      : 2001-14-07
```

B. TRANSFORMATION INFO - OBJECT CLASS

Rdn = ti

Child of TRANSFORMATION CATALOG

Attributes:

- **Required**

ProgramName	prog	cis
Type	type	cis

Where type is binary distribution or src distribution or both.

- **Optional**

Valid	valid	cisboolean
ValidDesc	vdesc	cis
Description	desc	cis

E.g. :

ti = FFT
dn : ti = FFT, tc = Griphyn
prog : Fast Fourier Transform
type : binary
valid : true
vdesc : valid transformation type
desc : To compute fast Fourier on the raw data
email : deelman@isi.edu

C. TRANSFORMATION ARCHITECTURE - OBJECT CLASS

Rdn = ta

Child of TRANSFORMATION INFO

Attributes:

Required

OperatingSystem os ces

Optional

Patch patch cis

E.g. :

ta = Griphyn
dn : ta = sparc, ti = FFT, tc = Griphyn
os : Solaris2.6
patch : Solaris 2.6.1.3

D. TRANSFORMATION LOCATION- OBJECT CLASS

Rdn = tl

Child of TRANSFORMATION ARCHITECTURE

Attributes:

Required

Executable fname ces

Version ver cis

Creator creator cis

Optional

CreatorEmail email cis

ConfigurationFile	conf	ces
Source	srcname	ces
Signature	sign	cis
CreateTimeStamp	cstamp	numeric

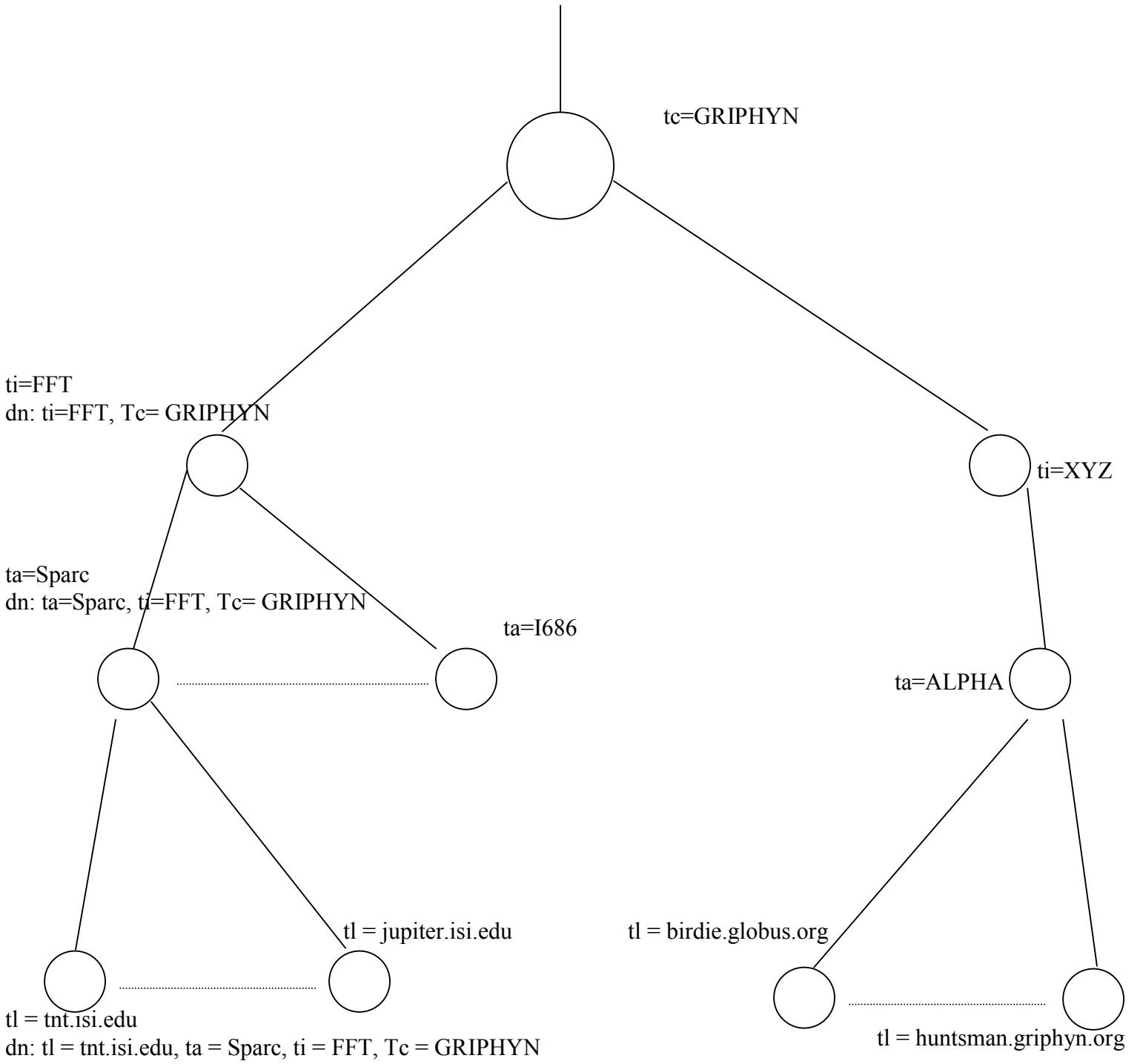
E.g. :

```
tl          = tnt.isi.edu
dn         : tl = abc.isi.edu, ta = sparc, ti = FFT, tc = Griphyn
fname      : gftp://tnt.isi.edu:8012/nfs/asd2/gmehta/transforms/fft/fourier.tar
srcname    : gcvs://tnt.isi.edu/cvs/griphyn/transforms/fourier
conf       : gftp://tnt.isi.edu:8012/nfs/asd2/gmehta /transforms/fourier.conf
ver        : 1.0.1
creator    : ISI
sign       : gftp://tnt.isi.edu:8012/nfs/asd2/gmehta/keys/verify.key
cstamp     : 7-12-01 12:00:01
```

Acknowledgments

We would like to thank Ann Chervenak and Laura Pearlman for their comments on the catalog design.

Visual Representation of the Example :



BIBLIOGRAPHY:

1. Deelman E., Foster I, Kesselman C., Livny M. Representing Virtual Data: A Catalog Architecture for Location and Materialization Transparency. Draft of January 26, 2001. (www.griphyn.org/news/meetings/2001apr09/foster_representing_virtual_data_jan01.doc)
2. Allen, B., Deelman, E., Kesselman, C., Lazzarini, A., Prince, T., Romano, J. and Williams, R. LIGO's Virtual Data Requirements, California Institute of Technology T000135-00-D, 2000. (www.griphyn.org/news/meetings/2001apr09/williams_ligo_requirements.doc)