



# HOW TO for Running LIGO examples through Chimera VDS



**Ewa Deelman, Gaurang Mehta, Karan Vahi**

deelman, gmehta,vahi@isi.edu

Version 1.0

## Overview

The Chimera Virtual Data System (VDS) provides a catalog that can be used by application environments to describe a set of application programs ("transformations"), and then track all the data files produced by executing those applications ("derivations"). Chimera contains the mechanism to locate the "recipe" to produce a given logical file, and then to create a "DAG" of program executions that will produce the requested file.

This Chimera "preview release" does not yet contain all features necessary for production application deployment, but it will serve the valuable purpose of providing a hands-on laboratory in which to try virtual data applications. [1]

Following is a "How To" document for running a generic LIGO example as well as the LIGO pulsar search example.

## Glossary of Terms

Abstract dag	An abstract directed cyclic graph with logical names of executables and files
Concrete dag	A concrete condor style dag which can be submitted to the condor system
DAG	A directed acyclic graph that expresses the order in which a sequence of jobs is to run, and the dependencies between the jobs
DAX	An abstract DAG expressed in XML
Transformation	Any executable or script
Derivation	An instantiation of a transformation
DAGman	Condor Directed acyclic graph manager
Condor	University of Wisconsin High Throughput Job Scheduler
Condor-G	Globus supported version of Condor
VDL	Virtual data language
VDLx	Virtual data language xml format
VDLt	Virtual data language textual format
VDS	Virtual data system
VDC	Virtual data catalog
Chimera	fancy name for virtual data system
Planner	The decision making engine in the VDS
Abstract planner	The first decision making engine which generates abstract dags
Concrete planner	The second decision making engine which takes a dax and generates Dags
Storage element	Gridftp enable machine sharing file system with one or more Compute Element
Computer element	Grid enabled machine and sharing file system with a Storage Element
Properties	The set parameters for the VDS

## Installation

Download the chimera binary distribution from [www.griphyn.org/chimera](http://www.griphyn.org/chimera)

Untar the distribution

Set VDS\_HOME environment variable to your VDS distribution directory

Set PATH=\$VDS\_HOME/bin

Set CLASSPATH=\$CLASSPATH:\$VDS\_HOME/lib

## Properties

Create a properties file in \$VDS\_HOME/etc directory with this syntax. The properties file tells the chimera system where to put and look the database files for the virtual data catalog (vdc) and the transformation catalog. (tc). Also it holds information about the replica catalog connection.

vds.rc.host	replica-catalog host url
vds.rc.password	replica-catalog password
vds.rc.collection	replica-catalog collection where the o/p files will be registered
vds.rc.manager_dn	replica-catalog manager dn
vds.db.tc	location of file to use as transformation catalog <i>default location \$VDS_HOME/var/tc.data</i>
vds.db.pool	location of file which has pool configuration <i>default location \$VDS_HOME/etc/pool.config</i>
vds.db.vds	location of file which will be the vds database <i>default location \$VDS_HOME/var/vds.nr/(0-9)</i>
vds.cplanner.work_dir	directory which will store all your working files (mount point +this dir) used only for head option
vds.cplanner.transfer_pool	not used currently
vds.schema.dax	points to the schema location for dax

Sample properties file

vds.rc.host	ldap://dc-user.isi.edu:9129/rc=Chimera_LIGO_Catalog, ou=isi.edu,o= NetscapeRoot
vds.rc.password	GIGadmin
vds.rc.collection	testchimera
vds.rc.manager_dn	cn=Directory Manager,ou=isi.edu,o=NetscapeRoot
vds.db.tc	/nfs/asd2/gmehta/chimeratestdir/ligo.tc
vds.db.pool	/nfs/asd2/gmehta/chimeratestdir/pool.config
vds.db.vds	/nfs/asd2/gmehta/chimeratestdir/ligo.vds
vds.cplanner.work_dir	/GriphynData/

## **Pool.config**

Create a pool.config file in the \$VDS\_HOME/etc/ directory ( you can create the file in some other directory and add the corresponding path in the \$VDS\_HOME/etc/properties file. The pool.config file contains the universe a pool supports, the host string for the globus job-manager, url-prefix for the grid-ftp server (optional will override one provided by replica catalog if specified) se mount point for file locations ( appended with working dir property vds.cplanner.work\_dir for getting the full paths to files) , startup scripts (cleanup scripts to be provided by Argonne , not used currently)

### Syntax of pool.config file.

#Poolname	Universe	job-mg-string	url-prefix	se-mount-point	startup-script
-----					
<pool>	<universe>	<hostname/jobmanager>	<null>	</some/se/path>	<path/to/startscript>
-----					
pool	-	any valid location registered in the replica catalog providing computational and/or storage facilities. Special pool name "local" denotes the storage/compute element is the local machine and not a remote gatekeeper or pool.			
universe	-	supported universes for the pools .			
	vanilla	condor	universe	:	job manager string has to point to a condor-jobmanager
	standard	condor	universe	:	job manager string has to point to a condor-jobmanager
	globus	universe	:	job manager string has to point to a fork jobmanager	
	transfer	universe	:	job manger string can point to any jobmanager depending on your system pool settings.	
url-prefix	-	needs to be null for the moment			
se-mount-point	-	the mount point for the storage element			
startup script	-	points to location of a start up script can be null for the moment ( not supported in this version)			

### Example pool.config file

#Poolname	Universe	job-mg-string	url-prefix	se-mount-point	startup-script
isi	vanilla	smarty.isi.edu/jobmanger-condor	null	/nfs/asd2/gmehta/se	/usr/bin/start
isi	standard	smarty.isi.edu/jobmanger-condor	null	/nfs/asd2/gmehta/se	null
isi	globus	smarty.isi.edu/jobmanger-fork	null	/nfs/asd2/gmehta/se	null
isi	transfer	smarty.isi.edu/jobmanger-fork	null	/nfs/asd2/gmehta/se	null
local	vanilla	null	null	/nfs/asd2/gmehta/se	null
local	standard	null	null	/nfs/asd2/gmehta/se	null
cal	vanilla	ldas.cal.edu/jobmanger-condor	null	/ligo/isi/gmehta/se	null
cal	standard	ldas.cal.edu/jobmanger-condor	null	/ligo/isi/gmehta/se	null
cal	globus	ldas.cal.edu/jobmanger-fork	null	/ligo/isi/gmehta/se	null
cal	transfer	ldas.cal.edu/jobmanger-fork	null	/ligo/isi/gmehta/se	null

**Transformation File.**

By default the transformation file is picked up from \$VDS\_HOME/tc.data but u can set it to any file in the \$VDS\_HOME/etc/properties file.

The syntax of the tc.data file is as follows

#poolname	logical transformation	physical transformation	environment string
<pool>	<transformation name>	<transformation location>	<env variables required>
pool	-	any valid location registered in the replica catalog providing computational and/or storage facilities. Special pool name "local" denotes the storage/compute element is the local machine and not a remote gatekeeper or pool.	
logical transformation-		the logical name of the transformation or executable	
physical transformation	-	the actual path to the transformation which is mapped to the logical tx.	
environment string	-	the environment variables required to run the transformation. Multiple env variables can be defined by separating <name>=<value> pairs by using“;”	

Example tc.data file

#poolname	logical transformation	physical transformation	environment string
isi	extract	/usr/local/bin/FrCopy	LIGO=/usr/ligo;PATH=/usr/bin
isi	concat	/usr/local/bin/FrCopy	LIGO=/usr/ligo;PATH=/usr/bin
isi	gsincftpget	/usr/globus/bin/gsincftpget	GLOBUS=/usr/globus2
isi	gsincftpget	/usr/globus/bin/gsincftpget	GLOBUS=/usr/globus2
local	griphynrc	/usr/chimera/bin/griphynrc	VDS_HOME=griphynrc
isi	resample	/usr/local/bin/resample	LIGO=/usr/ligo;PATH=/usr/bin
isi	sft	/usr/local/ldas/bin/sft	LDAS=/usr/local/ldas
isi	deft	/usr/local/ldas/bin/sft	LDAS=/usr/local/ldas

**Writing a vdl file to populate the VDS database.**

**(Example 1)** ligo.vdl

------(start of file)

```
TR extract( input a, none b="*", output c, none d="6")
{
  argument = "-i "${input:a}";
  argument = "-a "${b}";
  argument = "-o "${output:c}";
  argument = "-c "${d}";
  profile hints.pnfUniverse = "vanilla";
}
```

```
TR concat( input a[], output b, none c="6" )
{
  argument files = "-i "${a}";
  argument = "-o "${output:b}";
  argument = "-c "${c}";
  profile hints.pnfUniverse = "vanilla";
}
```

```
DV left->extract( a=@ {input:"H-658020145.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020145-op.F"}, d="-1");
```

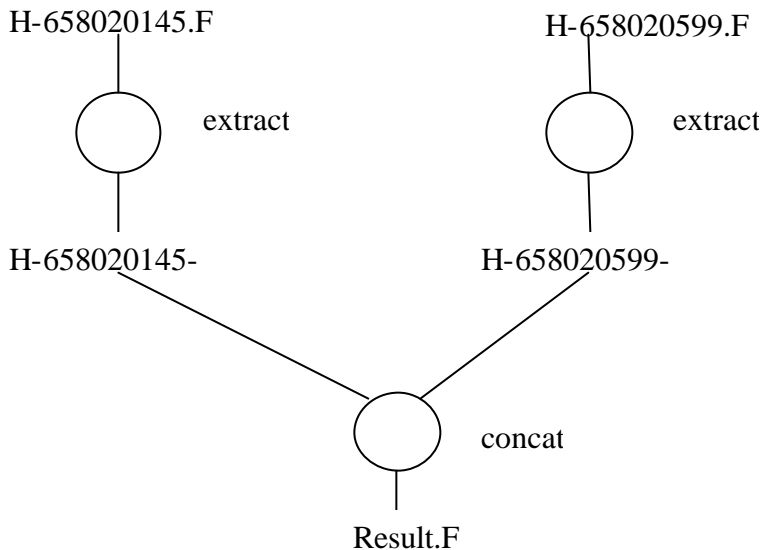
```
DV right->extract( a=@ {input:"H-658020599.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020599-op.F"},d="-1");
```

```
DV bottom->concat( a=[ @ {input:"H-658020145-op.F"},@ {input:"H-658020599-op.F"} ]$
  b=@ {output:"result.F"},c="-1");
```

----- (end of file)

*(please refer to the VDLT language reference in the \$VDS\_HOME/doc directory for how to write VDLT)*

The derivation defined above results in a “ V “ shaped dag



## **Generate VDLX from VDLT**

Run the following command

```
vdlt2vdlx ligo.vdl > ligo.vdlx
```

To store this vdlx in the vds database type the following command

```
insertvdc ligo.vdlx
```

(note your vds.db.vdc variable in \$VDS\_HOME/etc/properties should be set to a file u want to store the database in.)

## **Generate a DAX :**

Run this command to generate an abstract dax with the abstract planner.

```
gendax -l ligotest -f result.F -o ligo.dax
```

The resultant dax would look like this

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- generated: 2002-07-22T17:34:11-07:00 -->
<!-- generated by: gmehta [US] -->
<adag xmlns="http://www.griphyn.org/chimera/DAX"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.griphyn.org/chimera/DAX http://www.griphyn.org/chimera/dax-
1.5.xsd" count="1" index="0" name="test">
  <filename file="H-658020599.F" link="input" isTemporary="false"/>
    <snip snip .....
      .....>
  <job name="extract" id="ID00003">
    <argument> -i <filename file="H-658020599.F" link="input" isTemporary="false"/> -o <filename
file="H-658020599-op.F" link="output" isTemporary="false"/></argument>
    <profile namespace="hints" key="pfUniverse">vanilla</profile>
    <uses file="H-658020599-op.F" link="output" isTemporary="false"/>
    <uses file="H-658020599.F" link="input" isTemporary="false"/>
  </job>
  <snip snip .....
    .....>
  <child ref="ID00001">
    <parent ref="ID00003"/>
    <parent ref="ID00002"/>
  </child>

```

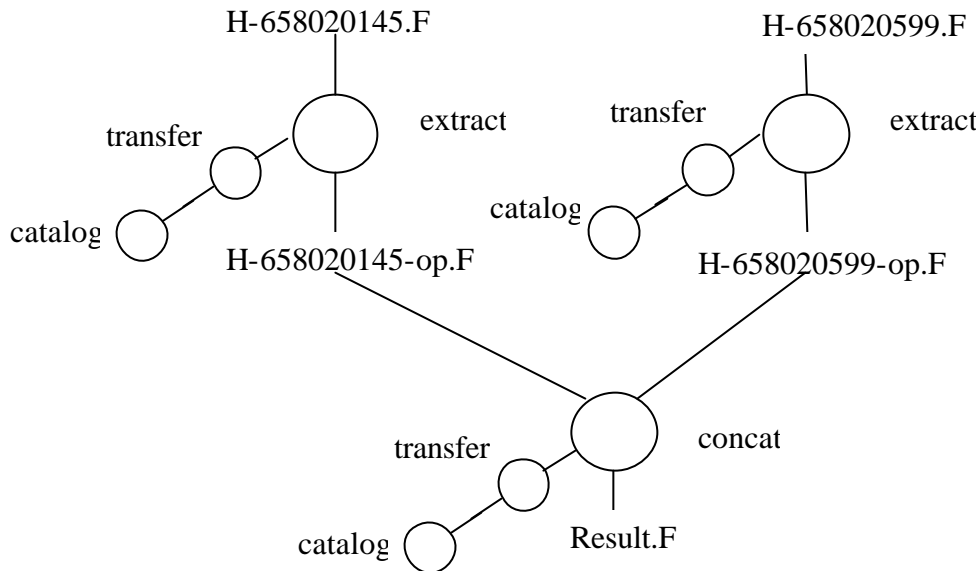
## Generate the concrete dags

Run the following command to generate condor runnable dag and submit files

```
gencdag --dax ligo.dax --dir ./testdags --donotrun --p isi --o isi
```

This command will generate all the required dags in the testdags directory

Each node will have a transfer as well as a replica catalog registering node attached to it.



Type the following command to submit the dag into Condor-G

```
condor_submit_dag <generated dag name>
```

The jobs in the dag will execute on pool isi and the output data will be transferred to the isi pool

**Example 2 : mini ligo pulsar search (ligopulsar.vdl)**

```
TR extract( input a, none b="*", output c, none d="6")
```

```
{
  argument = "-i "${input:a}";
  argument = "-a "${b}";
  argument = "-o "${output:c}";
  argument = "-c "${d}";
  profile hints.pnfUniverse = "vanilla";
}
```

```
TR concat( input a[], output b, none c="6" )
```

```
{
  argument files = "-i "${a}";
  argument = "-o "${output:b}";
  argument = "-c "${c}";
  profile hints.pnfUniverse = "vanilla";
}
```

```
TR resample( input a, output b, none sf1="0", none sf2="16384", none lpf="8192")
```

```
{
  argument = "-i "${input:a}";
  argument = "-o "${output:b}";
  argument = "-sf1 "${sf1}";
  argument = "-sf2 "${sf2}";
  argument = "-lowpass "${lpf}";
  profile hints.pnfUniverse = "globus";
}
```

```
TR sft( input a, output b, none c="SAMPLE8")
```

```
{
  argument = "-i "${input:a}";
  argument = "-o "${output:b}";
  argument = "-normalized -t "${c}";
  profile hints.pnfUniverse = "globus";
}
```

```
TR deft( input a[], output b, none c="argument")
```

```
{
  argument files = "-i "${a}";
  argument = "-o "${output:b}";
  argument = "-searchpulsar -c "${c}";
  profile hints.pnfUniverse = "globus";
}
```

```

DV first1->extract( a=@ {input:"H-658020145.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020145-op.F"}, d="-1");

DV first2->extract( a=@ {input:"H-658020146.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020146-op.F"}, d="-1");

DV first3->extract( a=@ {input:"H-658020599.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020599-op.F"}, d="-1");

DV first4->extract( a=@ {input:"H-658020600.F"}, b="IFO_DCDM_1",
  c=@ {output:"H-658020600-op.F"}, d="-1");

DV second1->resample(a=@ {input:"H-658020145-op.F"},
  b=@ {output:"H-658020145-res.F"}, sf1="0", sf2="2048", lpf="1024");

DV second2->resample(a=@ {input:"H-658020146-op.F"},
  b=@ {output:"H-658020146-res.F"}, sf1="0", sf2="2048", lpf="1024");

DV second3->resample(a=@ {input:"H-658020599-op.F"},
  b=@ {output:"H-658020599-res.F"}, sf1="0", sf2="2048", lpf="1024");

DV second4->resample(a=@ {input:"H-658020600-op.F"},
  b=@ {output:"H-658020600-res.F"}, sf1="0", sf2="2048", lpf="1024");

DV third1->concat( a=[@ {input:"H-658020145-res.F"}, @ {input:"H-658020146-res.F"}],
  b=@ {output:"H-result1.F"}, c="-1");

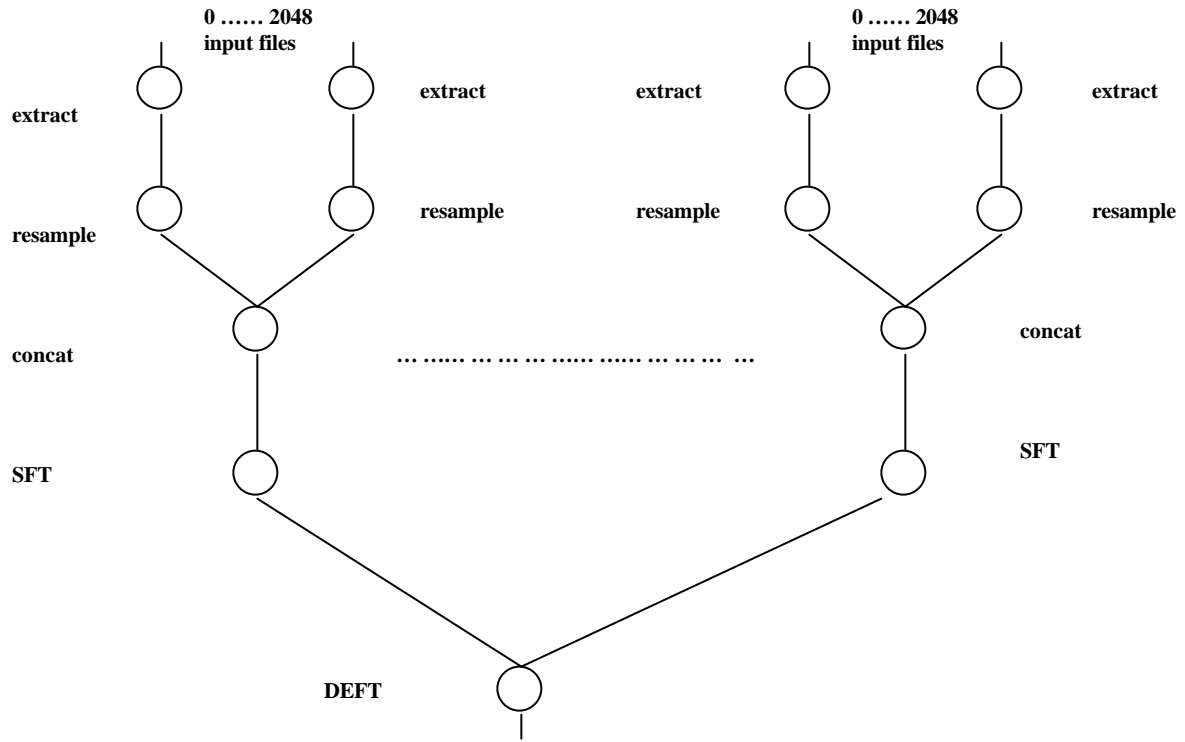
DV third2->concat( a=[@ {input:"H-658020599-res.F"}, @ {input:"H-658020600-res.F"}],
  b=@ {output:"H-result2.F"}, c="-1");

DV fourth1->sft(a=@ {input:"H-result1.F"}, b=@ {output:"H-SFT1.F"}, c="SAMPLE16");

DV fourth2->sft(a=@ {input:"H-result2.F"}, b=@ {output:"H-SFT2.F"}, c="SAMPLE16");

DV fifth1->deft(a=[@ {input:"H-SFT1.F"}, @ {input:"H-SFT2.F"}], b=@ {output:"pulsar.F"}),
  c="someargument");

```



Ligo Pulsar Search Dag .