PNFS/Chimera

Tigran Mkrtchyan

Content

- *What do we need pnfs/chimera for ?*
- *How does dCache use the namespace ?*
- *Where are the various bits and pieces stored ?*
- *Why is PNFS no longer good enough ?*
- *Chimera pros*
  - *Design wise*
  - *Performance wise*
  - *Functional wise*
- *Some SQL examples.*
- *First results from the NDGF migration.*

Quick Reminder

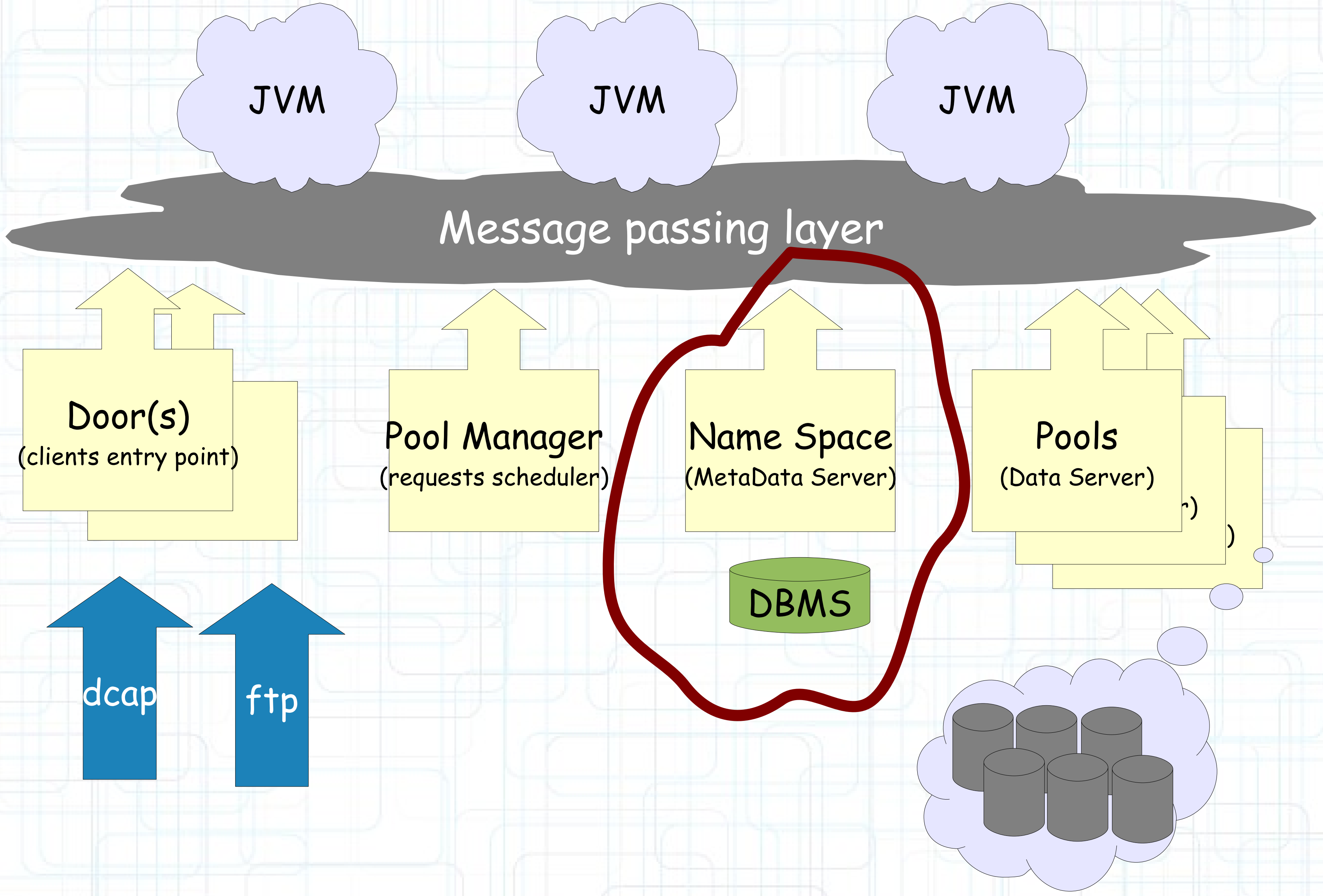
What is PNFS/Chimera ?

And how is it used within dCache ?

PNFS/Chimera provides name space functionality to dCache

- Mapping from path to file/directory object.
- Storage of primary meta data (size, permissions...)
- Storage of dCache specific meta data (location)

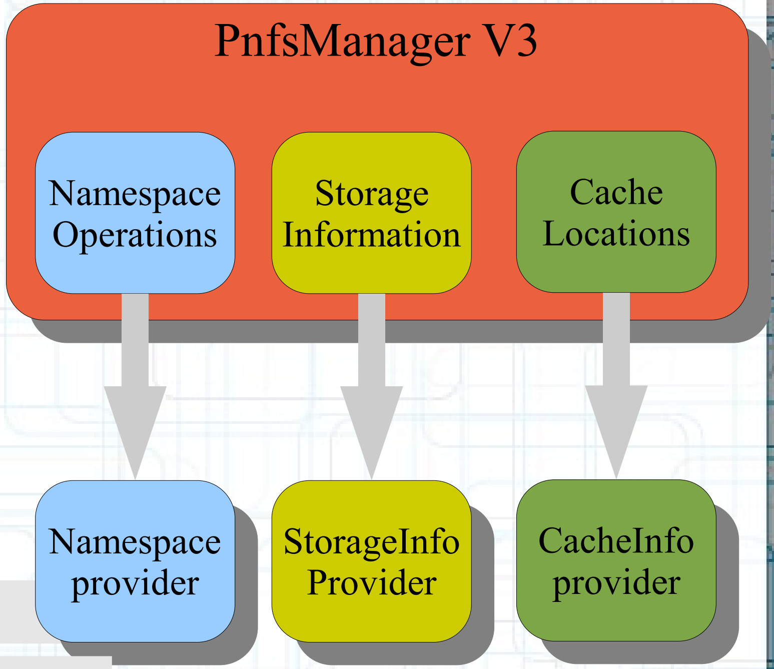
Why does dCache need Chimera/PNFS



Why does dCache need Chimera/PNFS

- Single rooted file system view
- Central container for file metadata (size, owner, atime and so on)
- Central container for file storage informations (AKA storage group)
- Central container for file location(s) within dCache (cacheinfo)
- Central container for file extra metadata (AKA file levels)

Although all containers are called **CENTRAL**, there is not need to store them in a single place.



Former PNFS	Level-0	Level-1	Level-2
Current PNFS	Level-0	Level-1	Companion DB
Chimera		See Later	

Why does dCache need Chimera/PNFS

PNFS and Chimera provide *almost* the same functionality to dCache.

None of the dCache components rely on a particular Namespace service implementation.

What is wrong with PNFS?

Nothing is really wrong ... **however** :

- Only a single way of accessing pnfs : through the NFSv2 stack.
  - Has to be used by dCache plus all mounted clients.
  - Negative side effect : file size limit of 2GB
- Serialized access to individual database(s) through a global DB lock
- Multiple DB transactions per high level operation.
- All metadata stored as BLOB (no SQL queries)
- Internal structure is platform dependent
  - DB can't be moved to an other OS/Platform
- dCache designed to work with PNFS
- Design is based on late 90's technology

Chimera pros (Design)

- Not a daemon – it's an API (and a Library)
- All 'clients' may work in parallel
- Single DB transaction per high level operation
  - Relies on DB transactional model (READ COMMITTED)
- File system view independent of metadata
  - Same objects may be represented by a different tree topology.
  - e.g. : would allow spaces to be represented as file system tree.
- Designed to benefit from the underlaying DB technology.
  - Easy to query.
  - Allows consistency check.
  - Some operations are delegated to *Stored Procedures* and *Triggers*.
- Designed to work with dCache

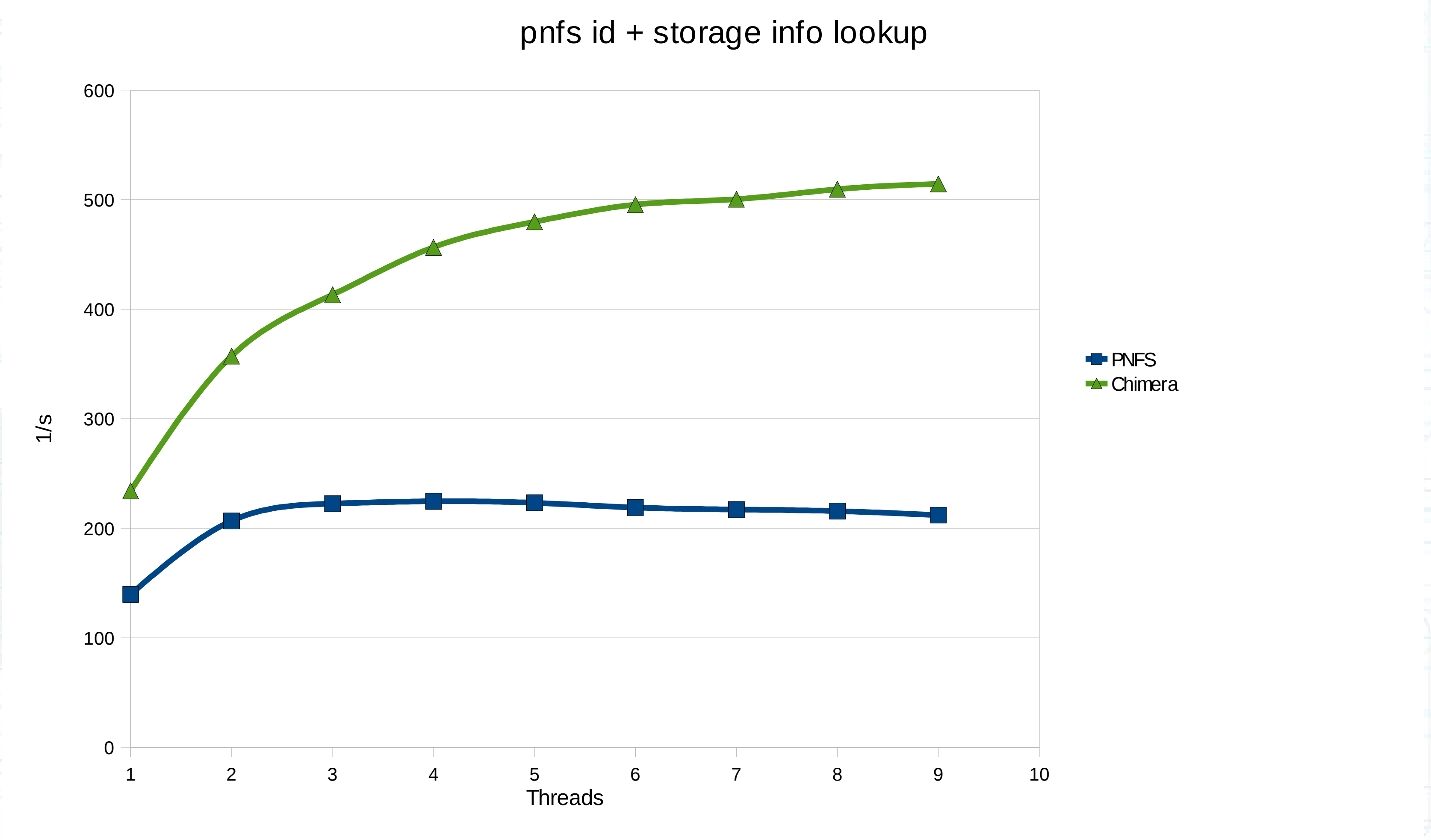
Chimera pros (For you)

- Speed
  - Improves with good data base implementation (Oracle, postgres)
- Scalability
  - Speed improves with more cores, threads.(See next slide)
- Functionality
  - Professional backup (Depends on DB)
  - SQL queries (Examples later)
  - Vendor/platform independent.
- Maintenance
  - Support for PNFS will sooner or later be reduced and discontinued.

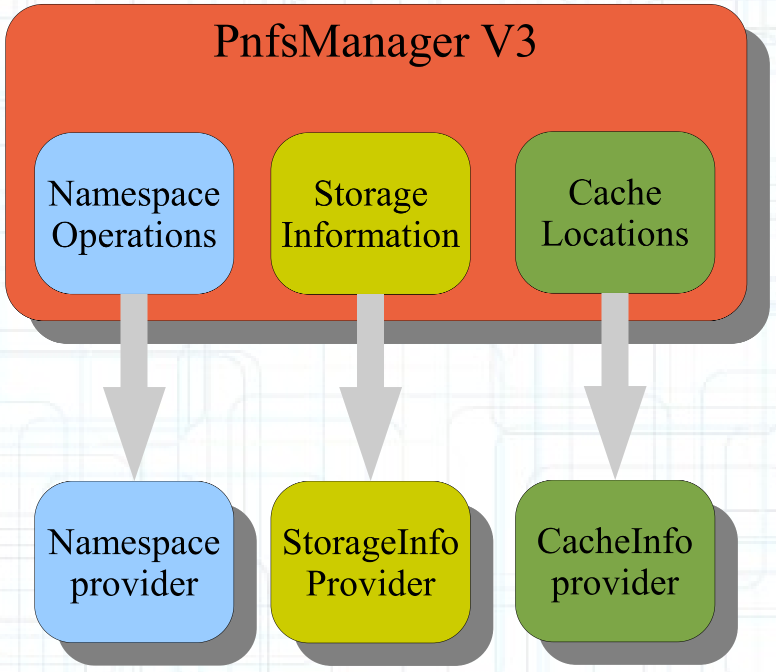
Chimera pros (scalability)

Stolen from Gerd Behrmann (NDGF production system)

Number of 'get storage info by path' per second on a 4 core machine.



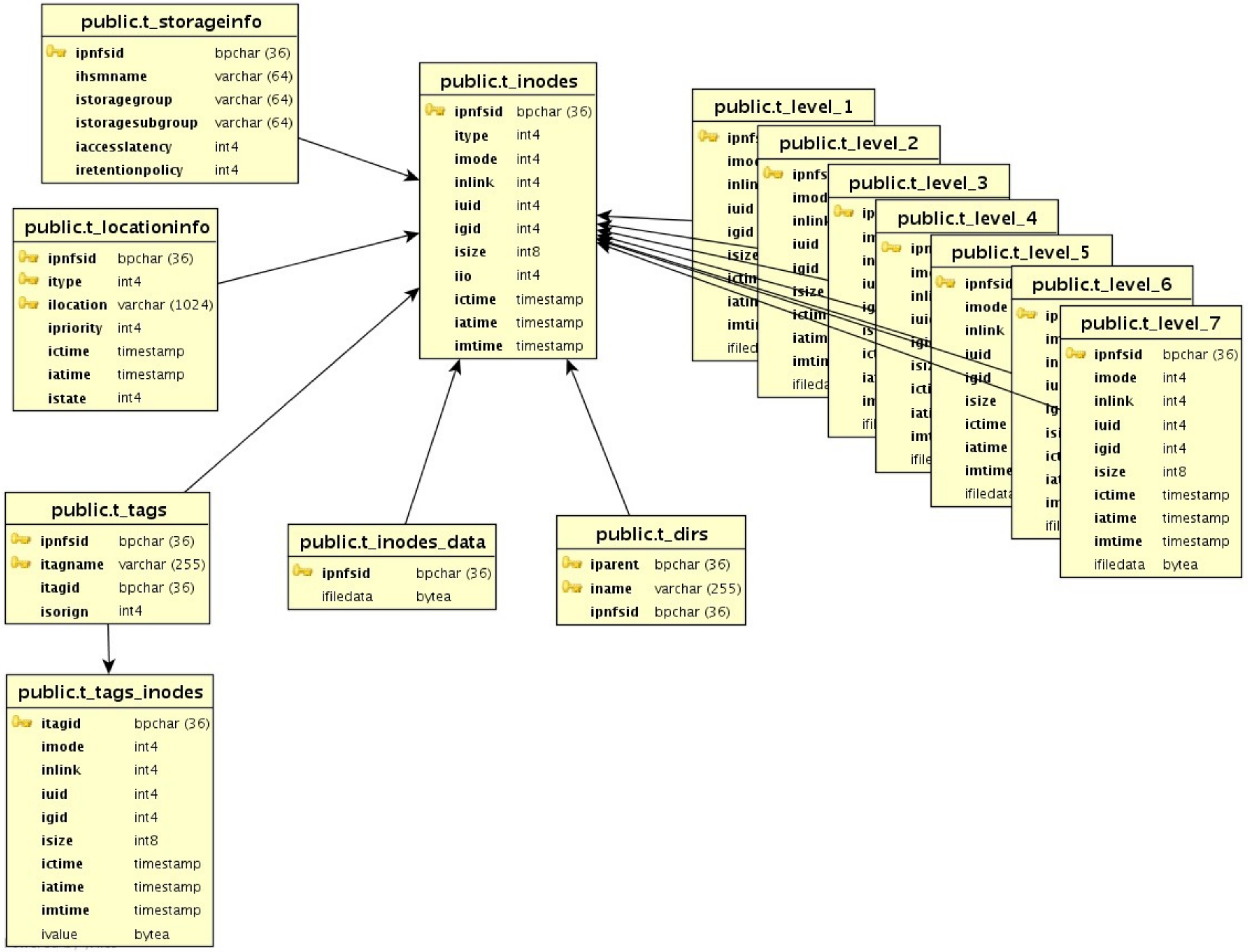
Chimera internals



Former PNFS	Level-0	Level-1	Level-2
Current PNFS	Level-0	Level-1	Companion DB

Chimera	Central Database

Chimera internals



public.t_storageinfo	
🔑 ipnfsid	bpchar (36)
ihsmname	varchar (64)
istoragegroup	varchar (64)
istoragesubgroup	varchar (64)
iaccesslatency	int4
iretentionpolicy	int4

public.t_locationinfo	
🔑 ipnfsid	bpchar (36)
🔑 itype	int4
🔑 ilocation	varchar (1024)
ipriority	int4
ictime	timestamp
iatime	timestamp
istate	int4

public.t_inodes	
🔑 ipnfsid	bpchar (36)
itype	int4
imode	int4
inlink	int4
iuid	int4
igid	int4
isize	int8
iio	int4
ictime	timestamp
iatime	timestamp
imtime	timestamp

public.t_level_7	
🔑 ipnfsid	bpchar (36)
imode	int4
inlink	int4
iuid	int4
igid	int4
isize	int8
ictime	timestamp
iatime	timestamp
imtime	timestamp
ifiledata	bytea

public.t_tags	
🔑 ipnfsid	bpchar (36)
🔑 itagname	varchar (255)
itagid	bpchar (36)
isorign	int4

public.t_inodes_data	
🔑 ipnfsid	bpchar (36)
ifiledata	bytea

public.t_dirs	
🔑 iparent	bpchar (36)
🔑 iname	varchar (255)
ipnfsid	bpchar (36)

public.t_tags_inodes	
🔑 itagid	bpchar (36)
imode	int4
inlink	int4
iuid	int4
igid	int4
isize	int8
ictime	timestamp
iatime	timestamp
imtime	timestamp
ivalue	bytea

Chimera internals

Tables	Description
t_inodes	all file system objects with attributes
t_dirs	file system tree view
t_locationinfo	internal and external locations of the files
t_level_N	'pnfs' levels
t_storageinfo	storage group information
t_checksum	files checksum
t_tags_inodes	all tags with attributes
t_tags	tags heirarchy

Chimera internals (t_inodes)

Attribute	***type***	***Description***
ipnfsid	CHAR(36)	Inode ID
itype	integer	Inode type, e.g. File, directory, symlink and so on
imode	integer	Permission mode in UNIX notation. In future reference to ACL.
inlink	integer	The reference count. For a file number of known references within filesystem, for a directory – number of entries.
iuid	integer	Virtual user id.
igid	integer	Virtual group id.
isize	long	The file size. For a directory 512.
iio	integer	File IO flag. For all files have to be 0. If the file should be stored in the backend database, like config files, then 1.
ictime	long	File creation time in milliseconds.
iatime	long	File last access time in milliseconds. Never updated.
imtime	long	File last modification time in milliseconds.

Chimera internals (t_inodes)

Useful SQL queries using the t_inodes table.

Total Number of files :

```
SELECT COUNT(*) FROM t_inodes WHERE itype=32768;
```

Change the uid for all files from 'old' to 'new' :

```
UPDATE t_inodes SET iuid=new WHERE iuid=old;
```

Space used by all files or by a selected subset.

```
SELECT SUM(isize) AS usedSpace FROM t_inodes WHERE itype=32768;

SELECT SUM(isize) AS usedSpace FROM t_inodes WHERE itype=32768
   AND iuid = ...;
```

Chimera internals (t_inodes)

Useful SQL queries using the t_inodes table.

Largest directories (most entries)

```
SELECT iparent, COUNT(iparent) AS pcount FROM t_dirs
  GROUP BY iparent ORDER BY pcount DESC
```

Number of files created since ...

```
SELECT COUNT(*) FROM t_inodes WHERE ictime > '2009-04-01';
```

Only limited by your fantasy

Chimera internals (t_locationinfo)

Attribute	***type***	***Description***
ipnfsid	CHAR(36)	inode id
itype	integer	type of location e.g. DISK(1), TAPE(0) and so on
ilocation	VARCHAR	the location
ipriority	integer	used in sorting requests
ictime	long	location creation time
iatime	long	location access time, not updated
istate	integer	location state, e.g. ON-LINE(1), OFF-LINE(0)

Chimera internals (t_locationinfo)

Useful SQL queries using the t_locationinfo table.

Find file system entries without corresponding file on pools :

```
SELECT COUNT(*) FROM t_inodes WHERE itype = 32768 AND isize > 0
  AND ipnfsid NOT IN ( SELECT DISTINCT ipnfsid FROM t_locationinfo );
```

Disable the usage of all files on a particular pool.

```
UPDATE t_locationinfo SET istate = 0 WHERE ilocation = 'pool-bla';
```

Summary

Chimera is certainly the next generation name space provider.