dCache

LCG Storage Element and HSM optimizer

Patrick Fuhrmann, DESY
for the dCache Team

dCache is a joint effort between the Deutsches
Elektronen Synchrotron (DESY)
and the Fermi National Laboratory (FNAL)

The Team

Jon Bakken, FNAL

Rob Kennedy, FNAL

Alex Kulyavtsev, FNAL

Timur Perelmutov, FNAL

Don Petravick, FNAL

Vladimir Podstavkov, FNAL

Michael Ernst, DESY

Patrick Fuhrmann, DESY

Martin Gasthuber, DESY

Tigran Mkrtchyan, DESY

Mathias de Riese, DESY

Sven Sternberger, DESY

Acknowledgments

CERN : Jean-Philipp Baud, Maarten Litmaath, Andreas Unterkircher

Karlruhe (gridKa) : Doris Ressmann

BNL : Scott O'Hare, Ofer Rind

Vanderbilt : Matthew T. Calef

Basic dCache System

Basic Specification

- Single 'rooted' file system name space tree
- Data may be distributed among a huge amount of disk servers.
- Supports multiple internal and external copies of a single file

Scalability

- Distributed Movers AND Access Points (Doors)
- Automatic load balancing using cost metric and inter pool transfers.
- Pool 2 Pool transfers on pool hot spot detection

Basic dCache System (cont.)

Configuration

- Fine grained configuration of *pool attraction scheme*
- Pool to pool transfers on configuration of *forbidden transfers*
- Fine grained tuning : Space vs. Mover cost preference

Tertiary Storage Manager connectivity

- Automatic HSM migration and restore
- Convenient HSM connectivity for
  enstore, osm, tsm, preliminary for Hpss by BNL.

Administration

- Using standard 'ssh' protocol for administration interface.
- First version of graphical interface available for administration
- Large set of options per module
  (due to different use pattern DESY <> FERMI)

Miscellaneous

- CRC checksum calculation and comparison (partially implemented yet)
- Pluggable door / mover pairs
- Data removed only if space is needed

LCG Storage Element

- DESY dCap lib incorporates with CERN s GFAL library
- gsiFtp support
- SRM version ~ 1 (1.7) in production
- limited GRIS functionality (using workaround)

Resilient dCache

- Controls number of copies for each dCache dataset
- Makes sure n < copies < m
- Adjusts replica count on pool failures
- Adjusts replica count on scheduled pool maintenance

Not yet in official distribution, but in production

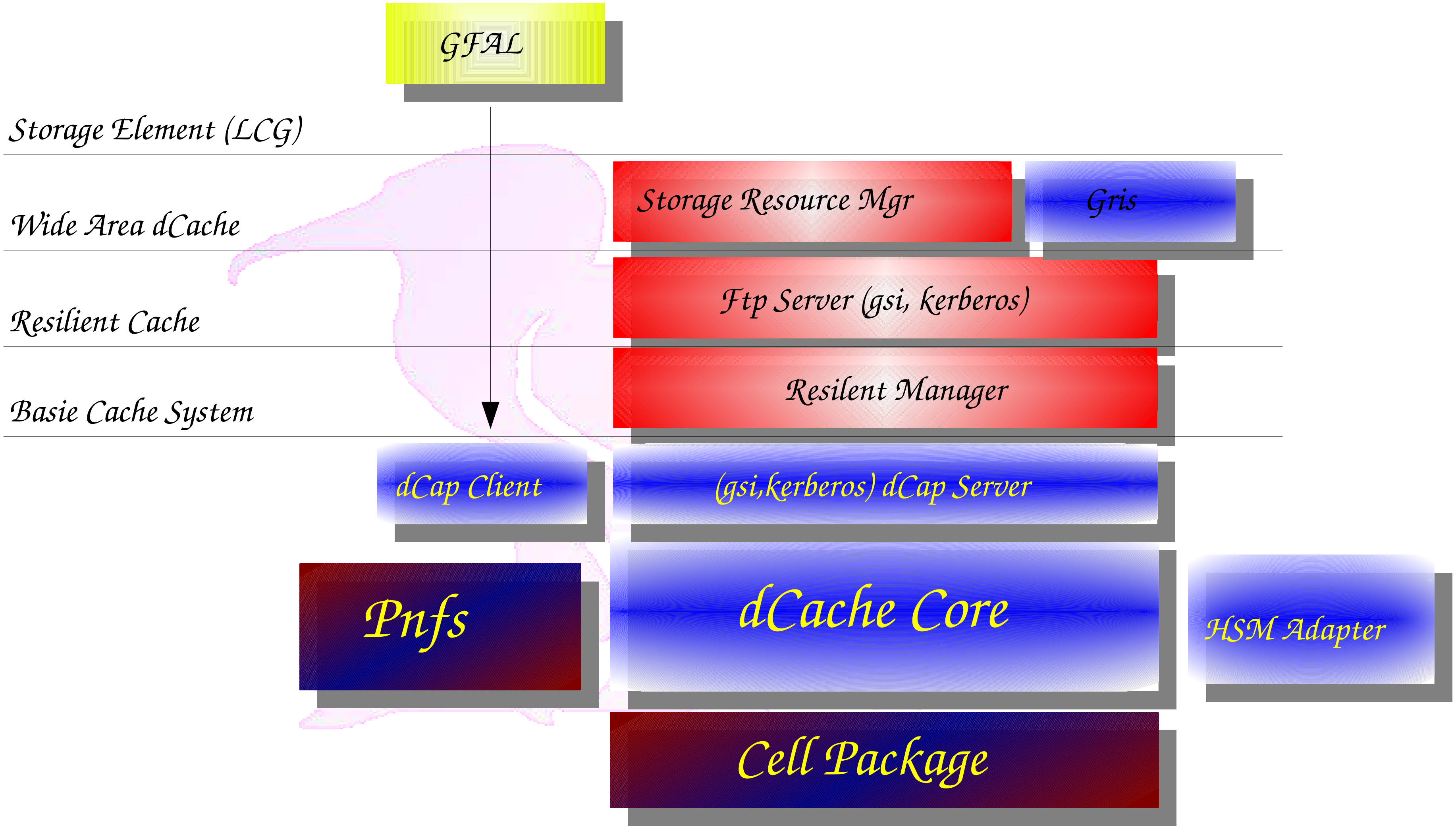
dCap Protocol and Implementation

- implements I/O and name space operations including 'readdir'
- works on mounted *pnfs* and URL like syntax
- available as standard shared object and preload library

```
ls -l dcap://dcachedoor.desy.de/user/patrick
```

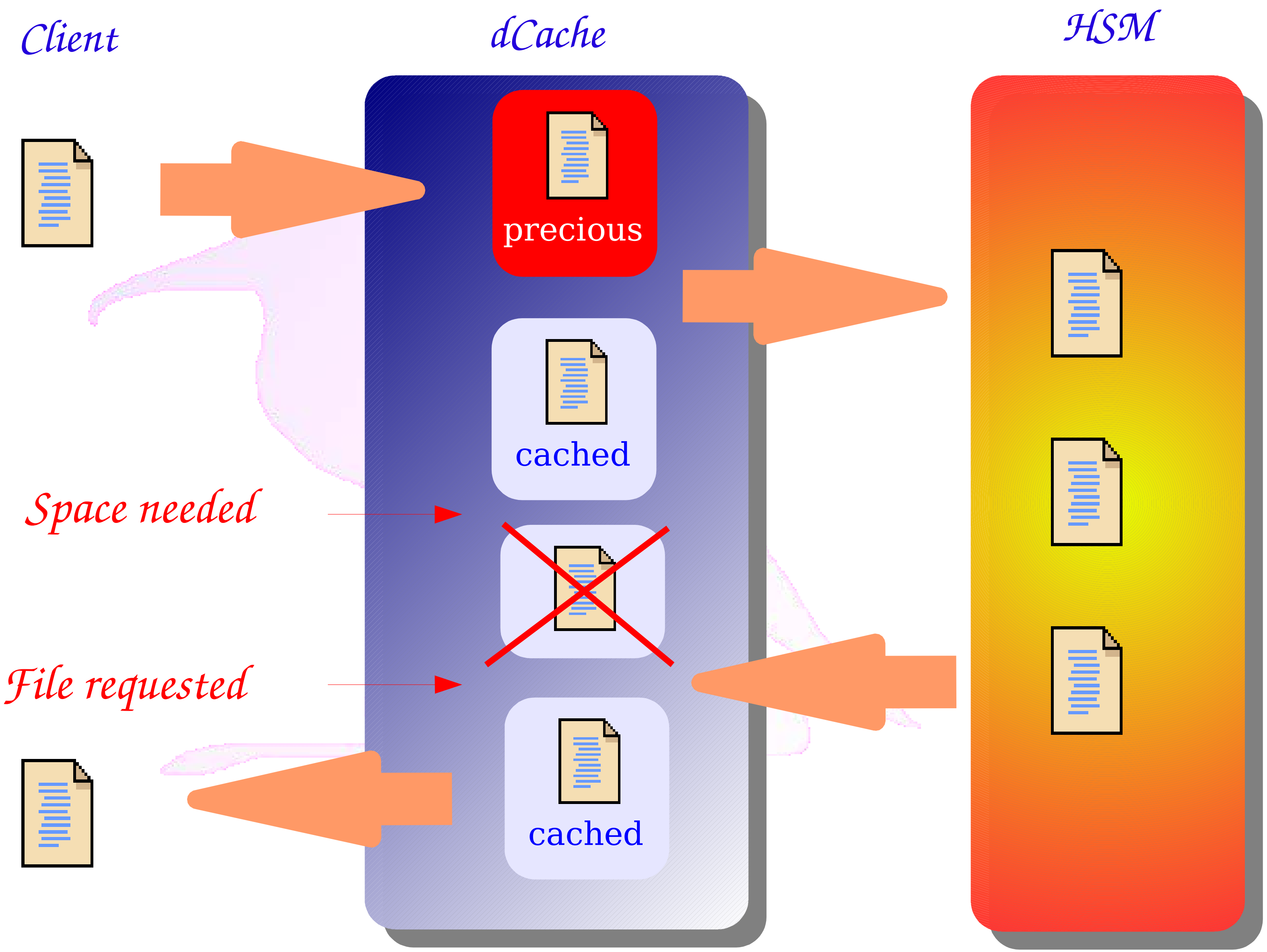
- positive tested for Linux, Solaris, Irix (partially for XP)
- automatic reconnect on server door and pool failures
- supports read ahead buffering and deferred write
- supports ssl, kerberos and gsi security mechanisms
- Thread safe
- Interfaced by *ROOT* ®

dCache Functionality Layers



dCache
The HSM Interface

dCache - HSM Interaction



Deferred HSM flush

Precious data is separately collected per storage class

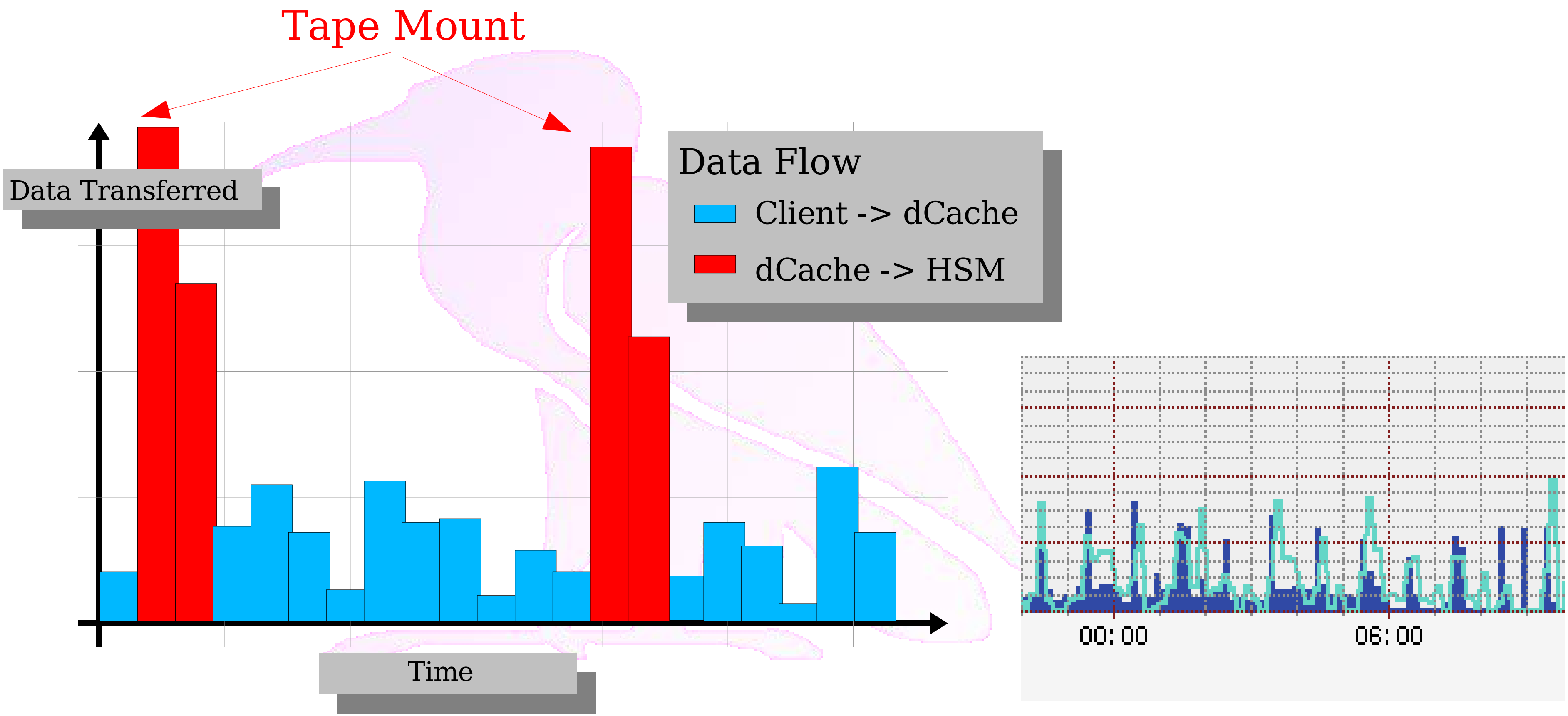
Each 'storage class queue' has individual parameters, steering the HSM flush operation.

- Maximum time, a file is allowed to be 'precious' per 'storage class'.
- Maximum number of precious bytes per 'storage class'
- Maximum number of precious files per 'storage class'

Maximum number of simultaneous 'HSM flush ' operations can be configured

Multiple HSMs instances and HSM classes are supported simultaneously

Deferred HSM flush (cont.)



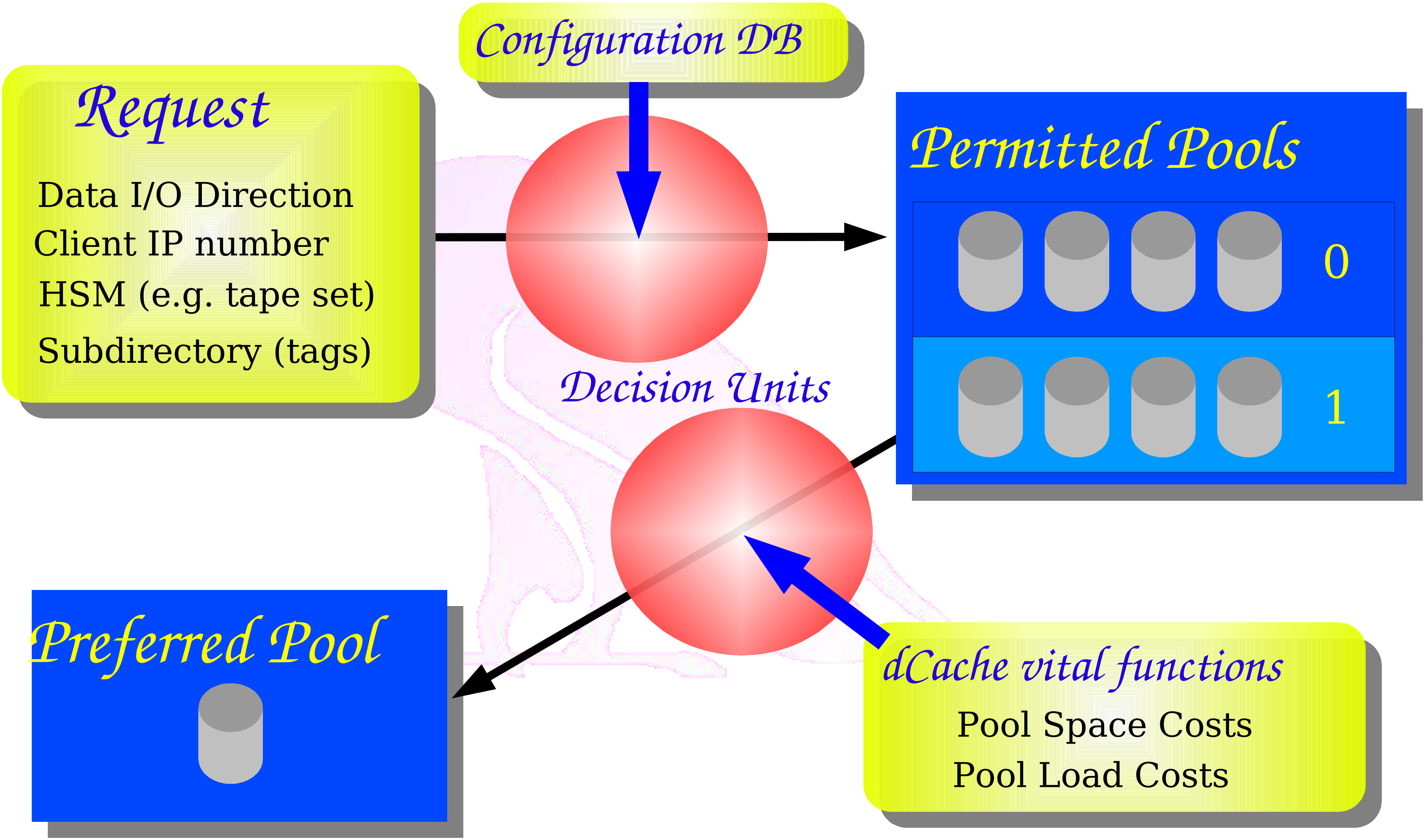
The Pool Selection Mechanism

Static Configuration

Dynamic Behavior

Tuning ...

Pool Selection Mechanism



Configuration DB

Request

Data I/O Direction
Client IP number
HSM (e.g. tape set)
Subdirectory (tags)

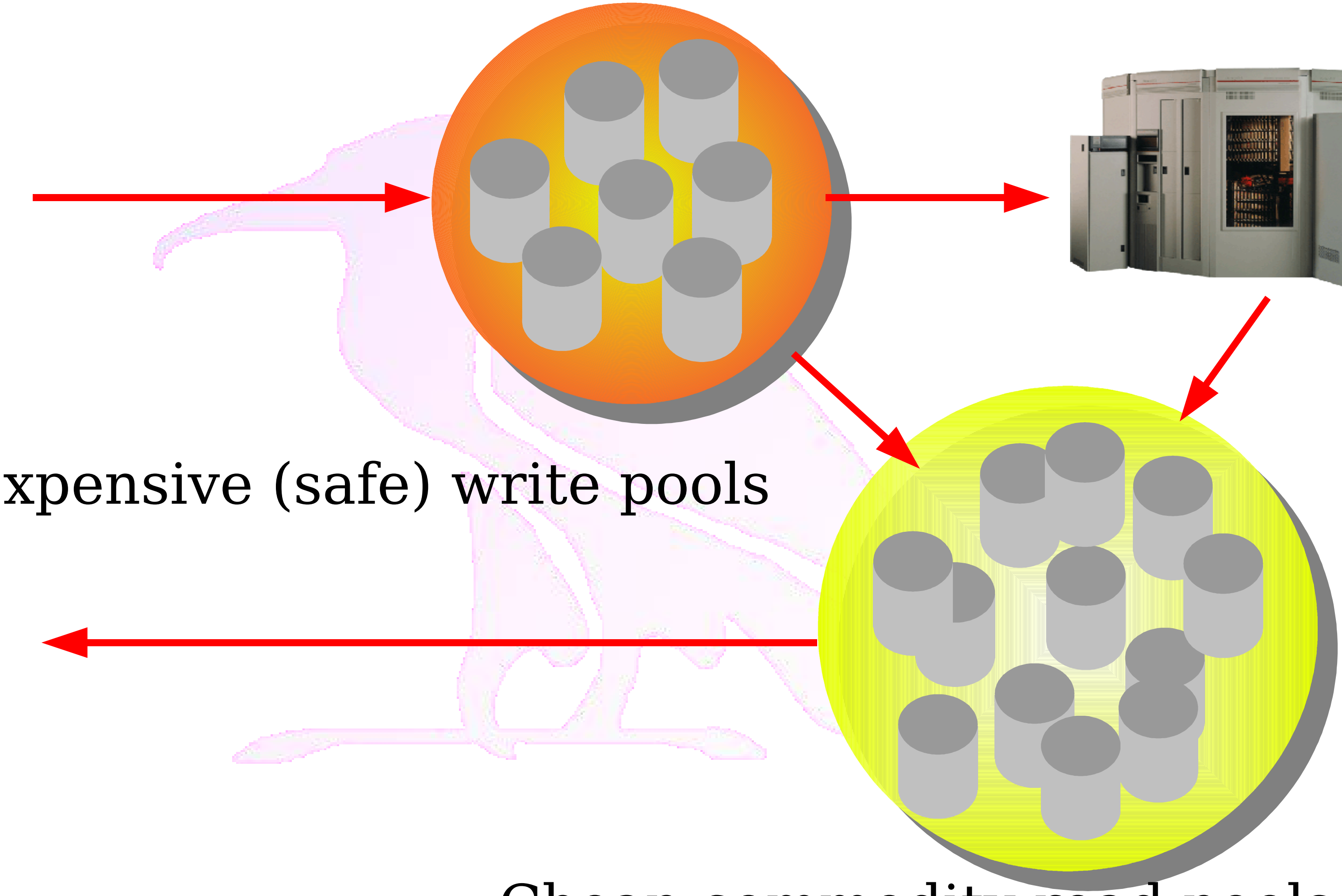
Permitted Pools

Decision Units

Preferred Pool

dCache vital functions

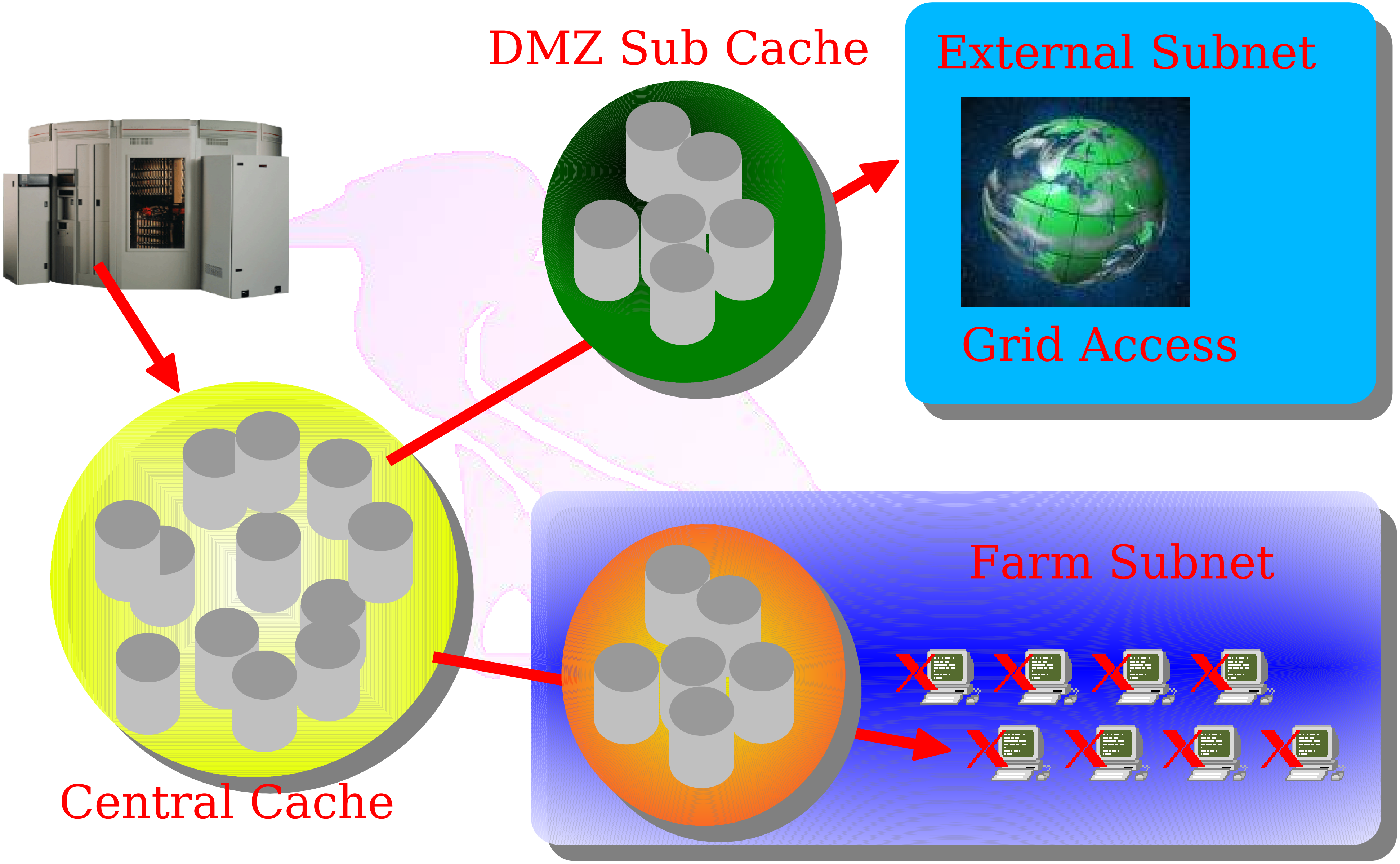
Pool Space Costs
Pool Load Costs



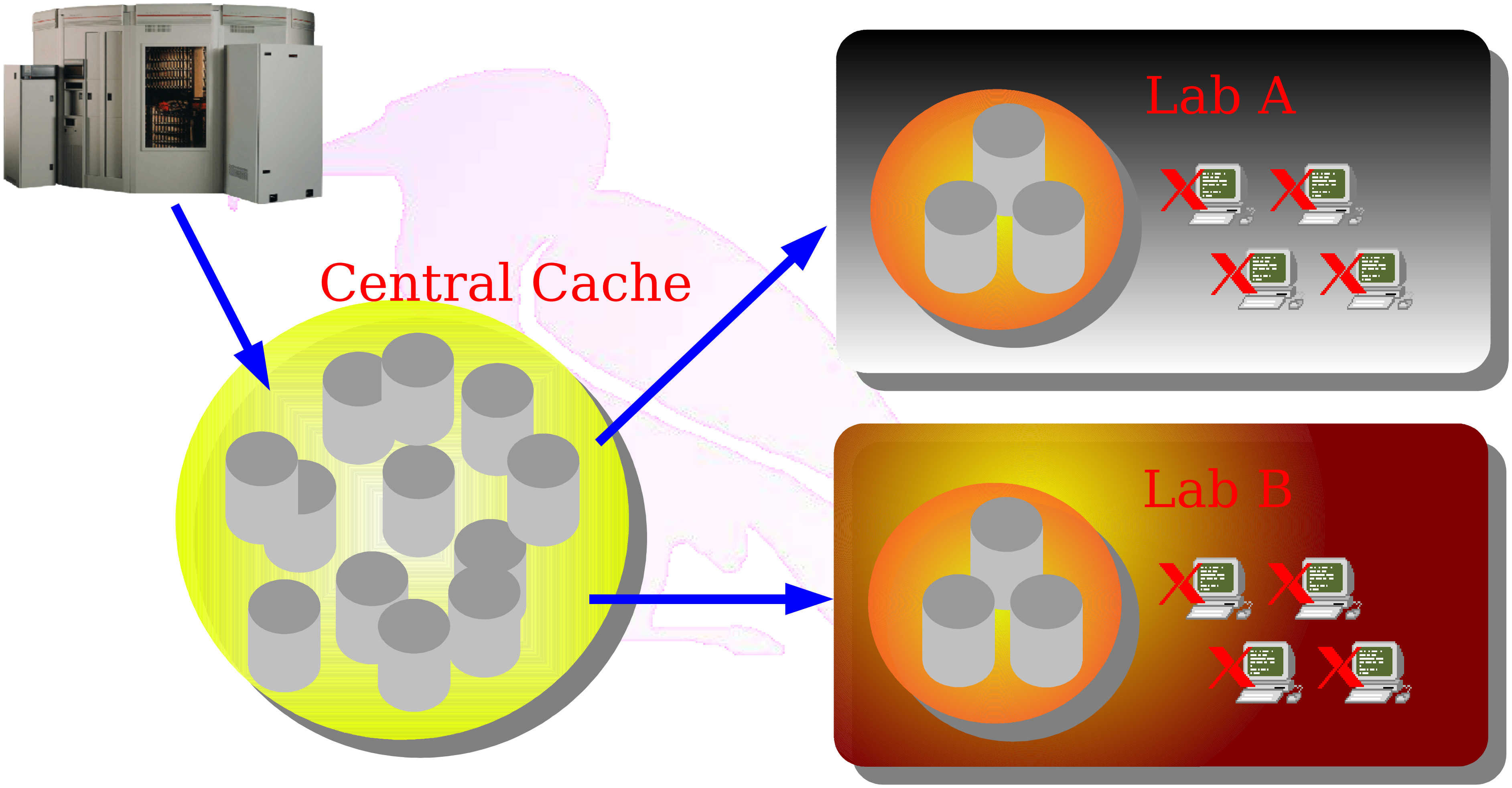
Expensive (safe) write pools

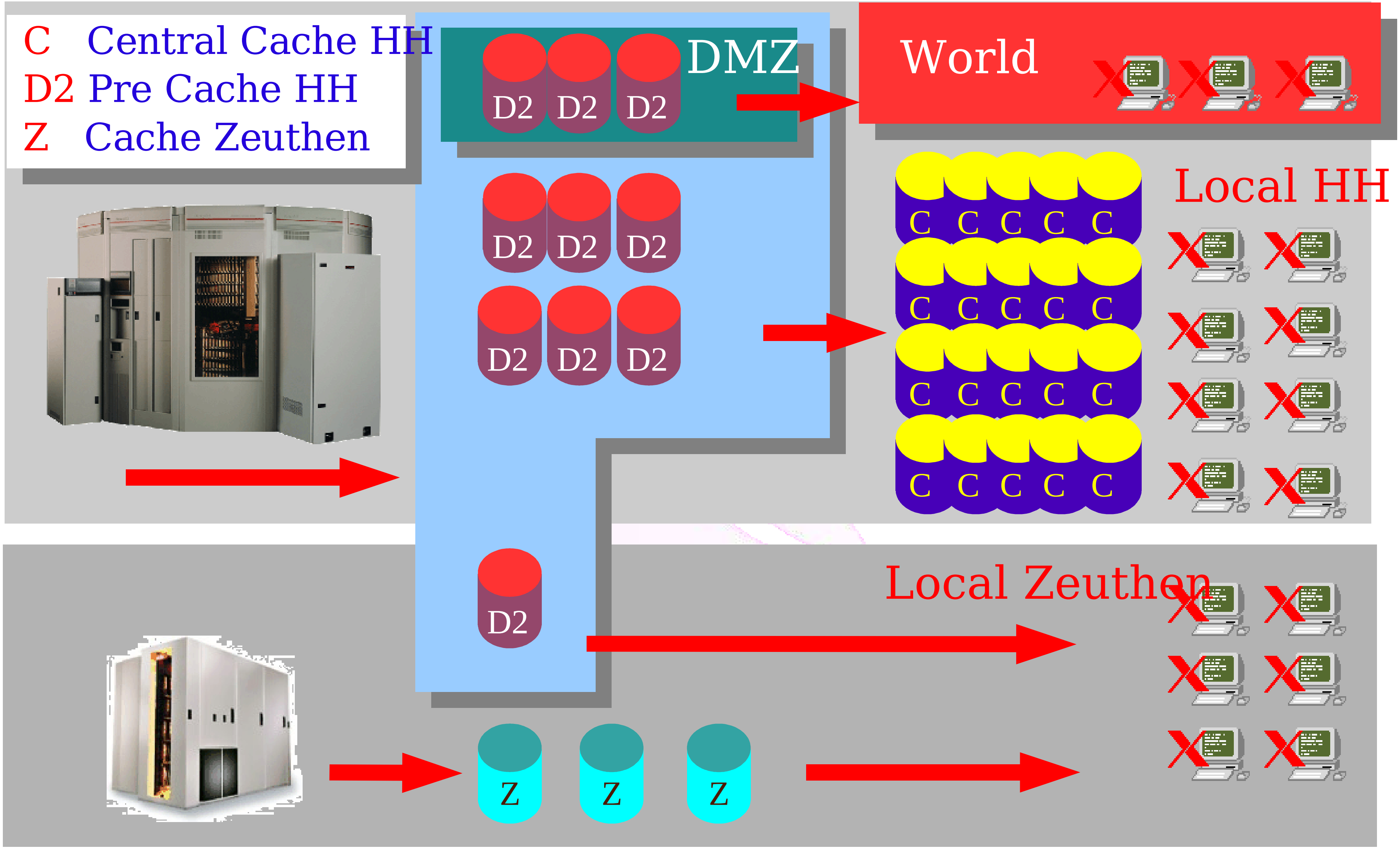
Cheap commodity read pools

Grid Pool Selection



Pool Selection





C Central Cache HH
D2 Pre Cache HH
Z Cache Zeuthen

Pool Selection : Tuning (1)

Space vs. Load

For each request, the central cost module generates two cost values for each pool :

Space : Cost based on available space or LRU timestamp

CPU : Cost based on the number of different movers (in,out,...)

The final cost, which is used to determine the best pool, is a linear combination of Space and CPU cost.

The coefficients needs to be configured.

Space coefficient << Cpu coefficient

Pro : Movers are nicely distributed among pools.

Con : Old files are removed rather than filling up empty pools.

Space coefficient >> Cpu coefficient

Pro : Empty pools are filled up before any old file is removed.

Con : 'Clumping' of movers on pools with very old files or much space.

dCache Basic Design : checksum

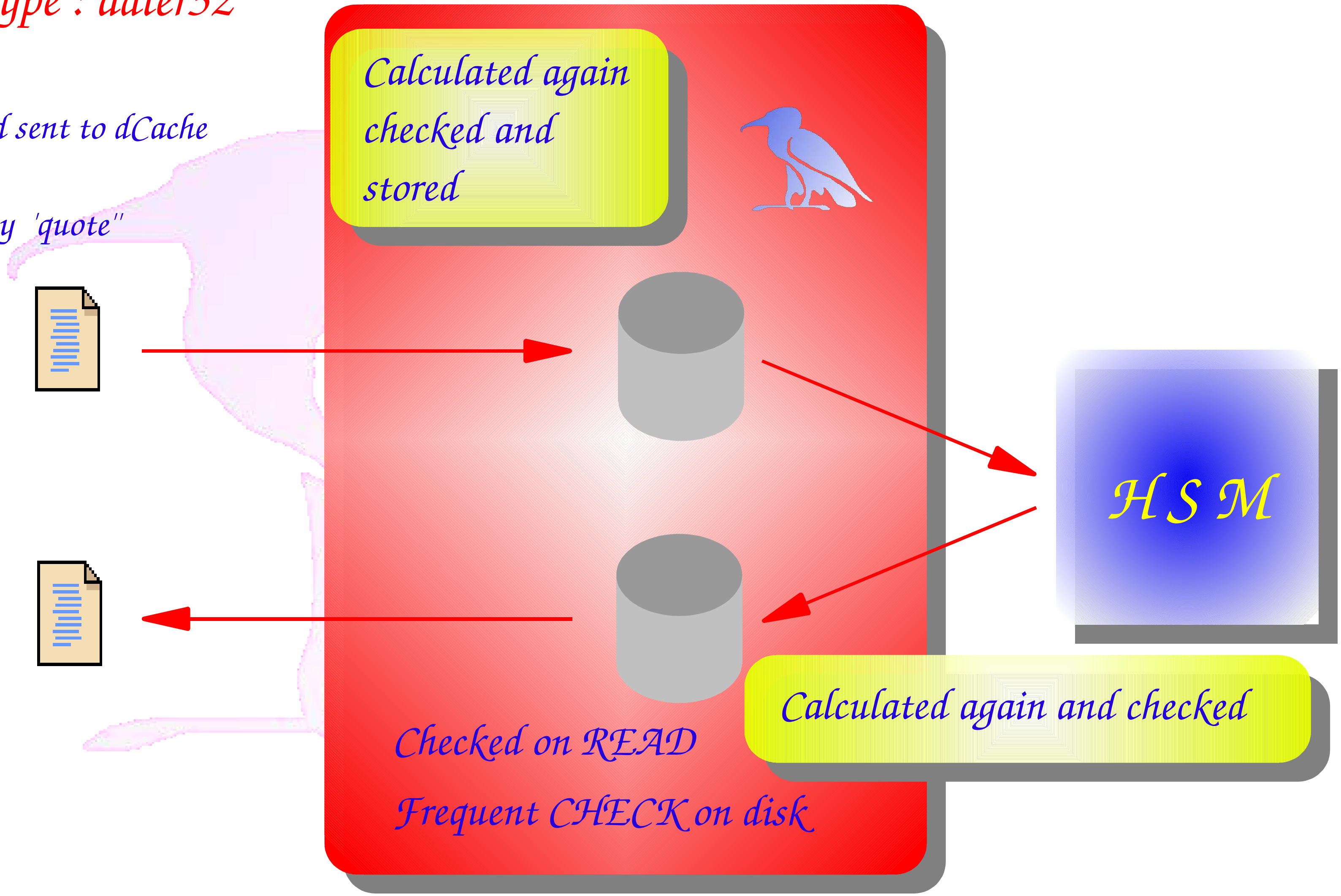
Supported Type : adler32

dCap

calculated and sent to dCache

(x) FTP

can be sent by 'quote''



Calculated again checked and stored

HSM

Calculated again and checked

Checked on READ

Frequent CHECK on disk

Future

The scalable Storage Element

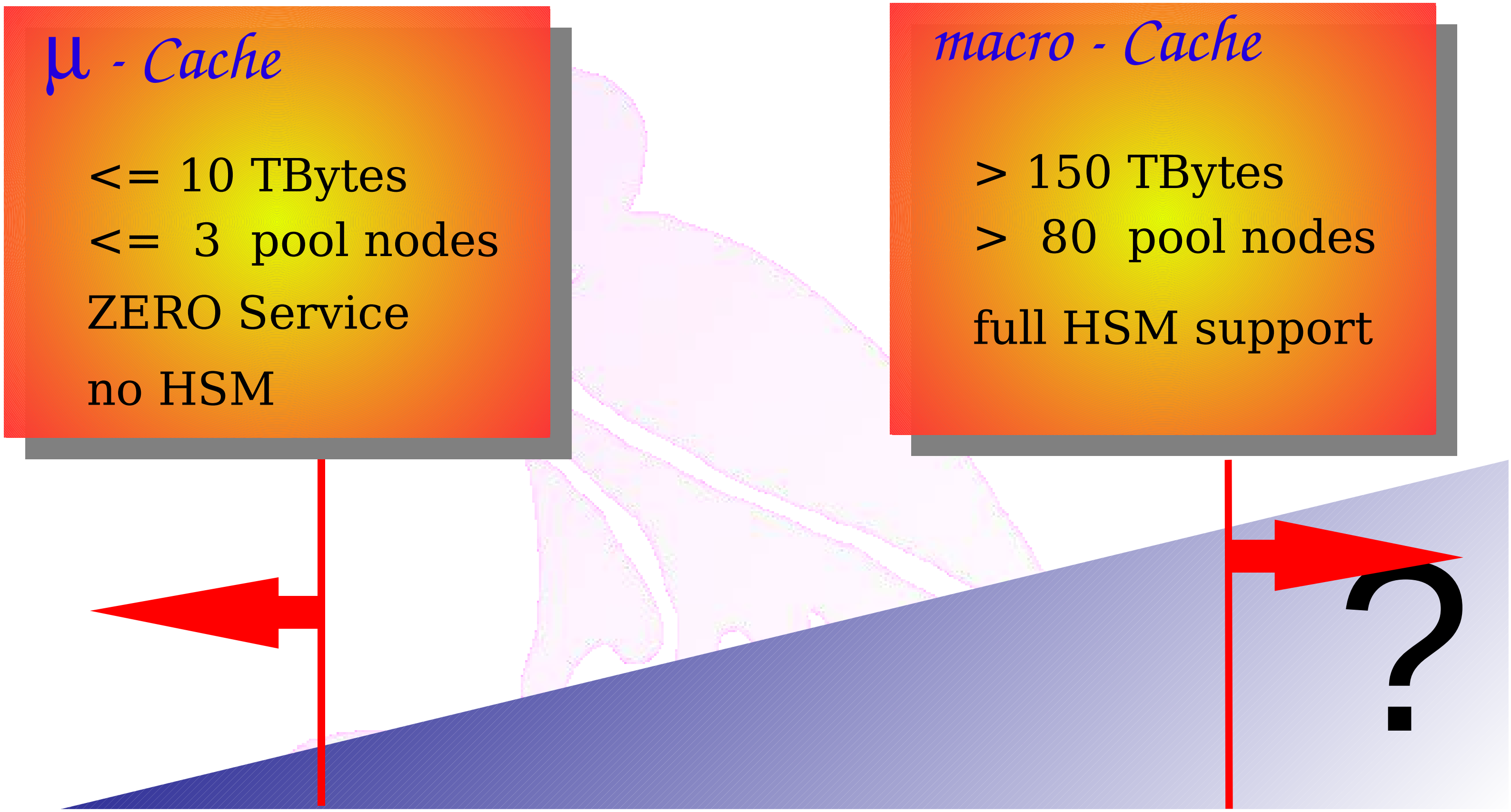
Improved Packaging and Documentation

Smart Prestager

High Performance Name Space

dCache as mirror HSM

Scalable Storage Element



µ - Cache

<= 10 TBytes
<= 3 pool nodes
ZERO Service
no HSM

macro - Cache

> 150 TBytes
> 80 pool nodes
full HSM support

Scalable Storage Element (cont.)

µ - Cache

How to achieve a 'zero service' micro Cache System ?

Possible partners and funding from D-Grid initiative

macro - Cache

Find 'non scalable components' in dCache.

First candidate : name service (pnfs)

Improved *Prestager*

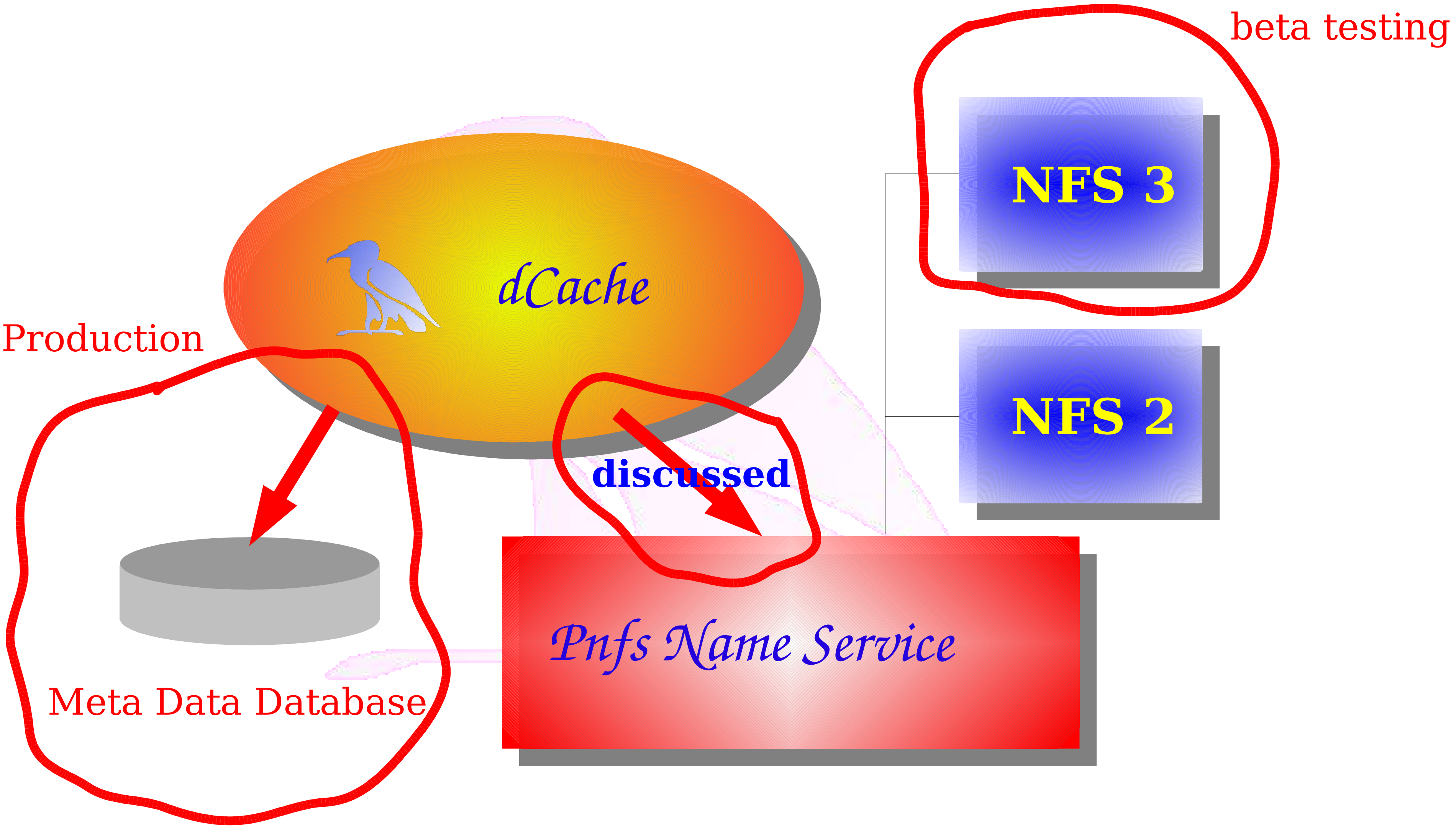
Collects requests in time bins

Submits requests per 'tape'

Knows about HSM load

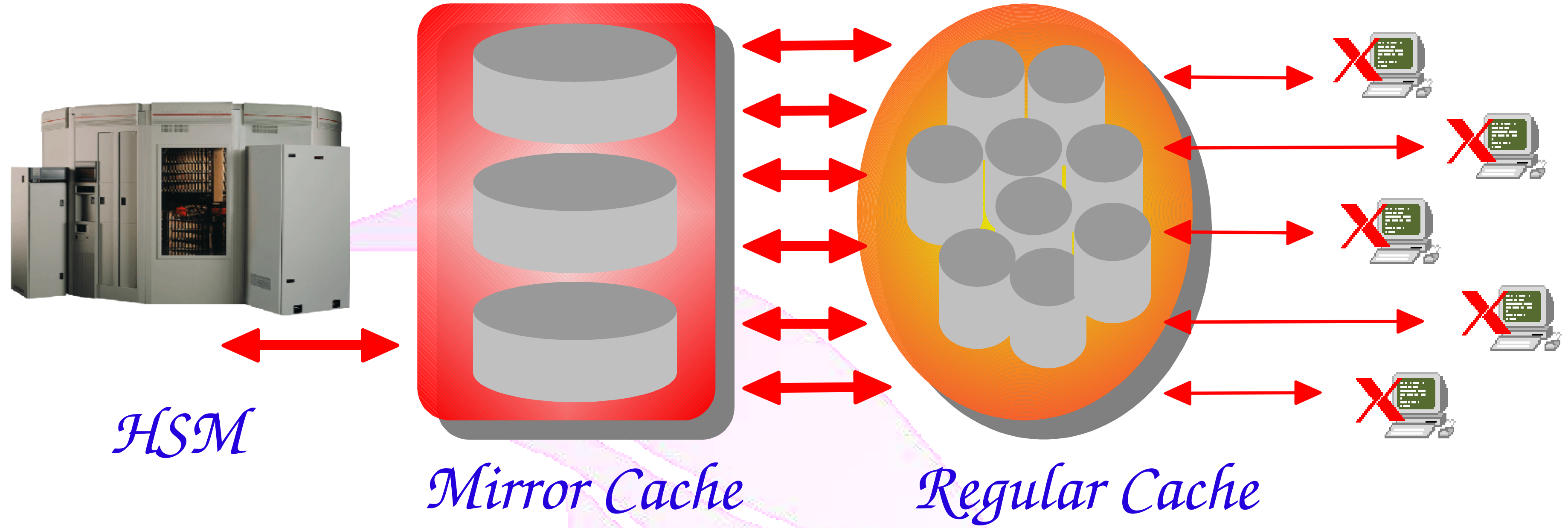
Improved dCache name service

Stolen from Tigrans talk



dCache as HSM mirror

Project managed by Martin Gasthuber, DESY



nearly all *HSM* data on *Mirror Cache*

Mirror Cache has highest possible data density (lowest dollars/Tbyte)

Controlled number of high speed streams between *Mirror Cache* and *Regular Cache*

Mirror Cache behaves like HSM (except for mount/dismount delays)

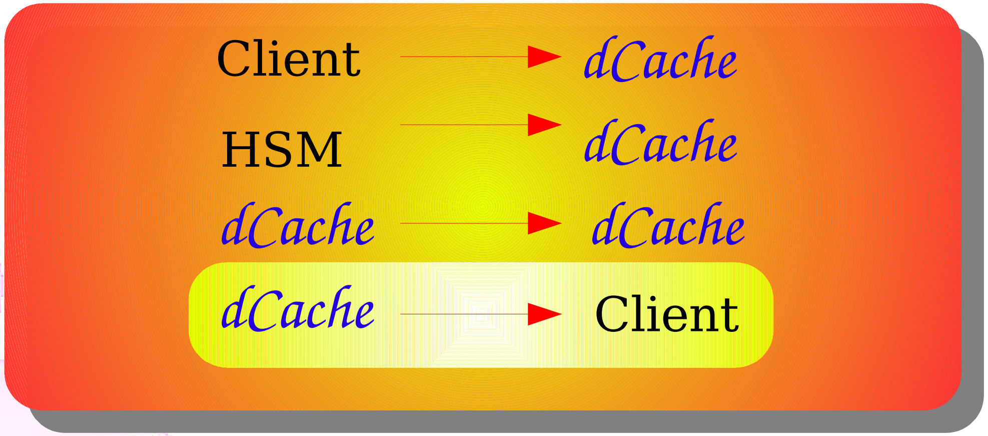
Mirror Cache disks switched OFF if not accessed

HSM to *Mirror Cache* transfers only after disk replacement

dCache
End of official presentation

Pool Selection Mechanism

Pool Selection required for



Client → dCache

HSM → dCache

dCache → dCache

dCache → Client

Pool selection is done in 2 steps

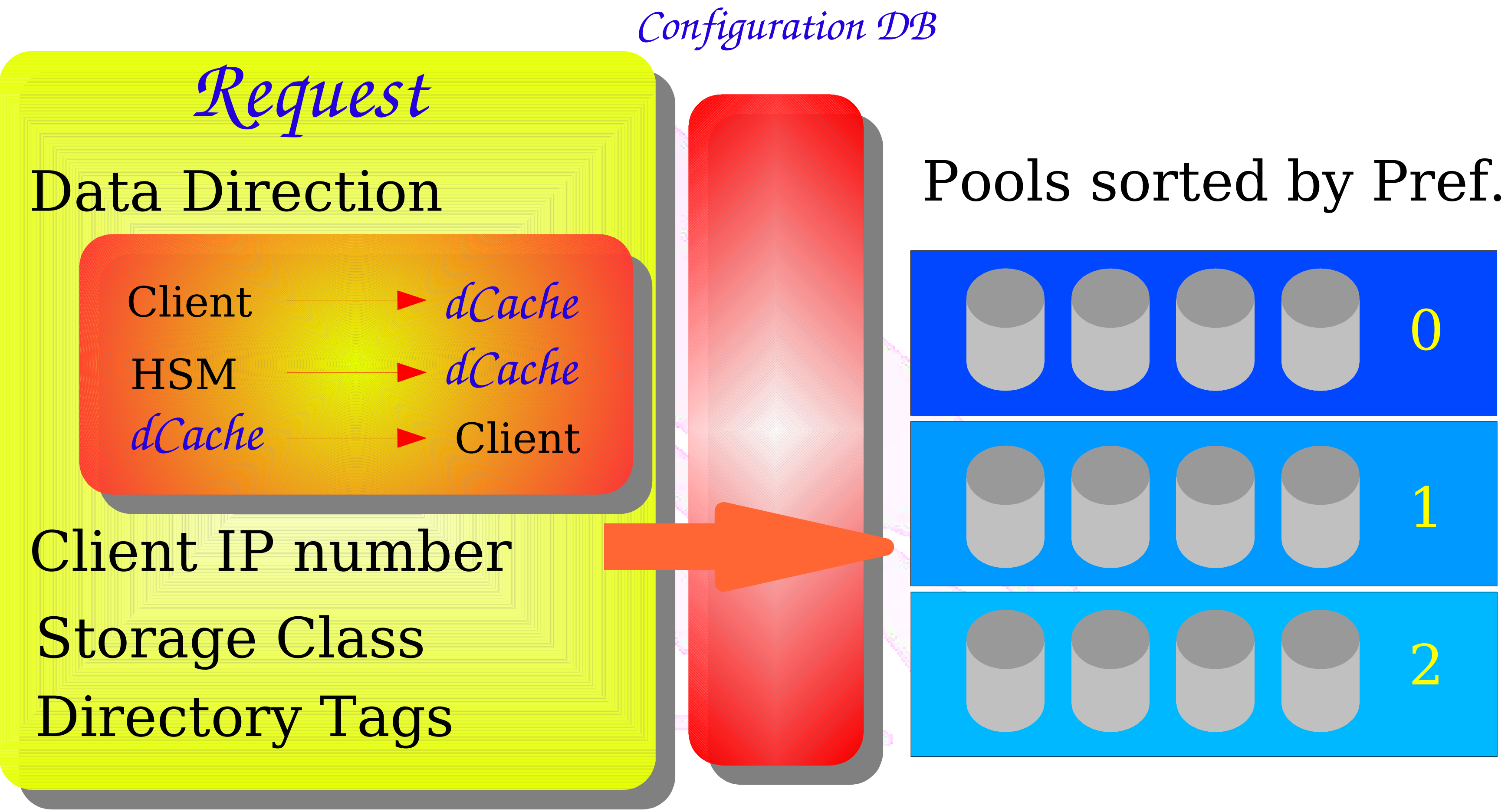
I) Query configuration database :

which pools are allowed for requested operation

II) Query 'allowed pool' for their vital functions :

find pool with lowest cost for requested operation

Pool Selection Mechanism



Configuration DB

Request

Data Direction

Client → dCache

HSM → dCache

dCache → Client

Client IP number

Storage Class

Directory Tags

Pools sorted by Pref.

0

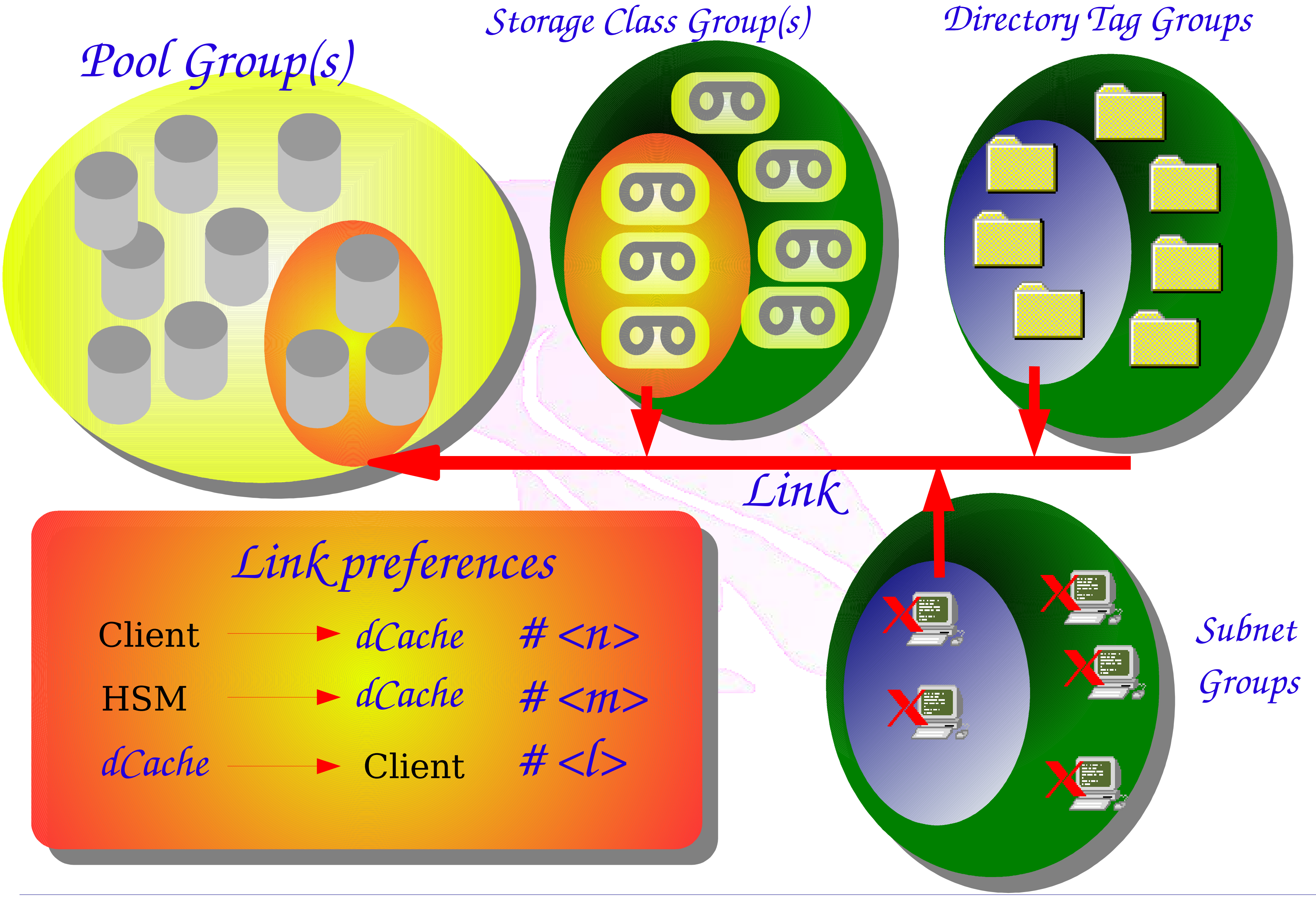
1

2

Mode A : fall-back only if all pools of pref. <x> are down.

Mode B : fall-back if cost of pools of pref. <x> is too high.

Pool Selection Mechanism



Link preferences

Client → dCache # <n>

HSM → dCache # <m>

dCache → Client # <l>

Pool Selection Mechanism

Goals / Use cases

Dedicated write pools (select by data direction)

Allow 'precious' files on secure disks only.
Read requests will trigger p2p to cheap disks. (e.g. datataking)

Support multiple HSMs (select by storage class)

Assign different pool set to different HSMs (e.g. HSM migration)

Support 'group owned' pool sets (select by storage class or tag)

Assign 'experiment data' to 'experiment owned pools'
BUT have 'fallback' pools common to all experiments.

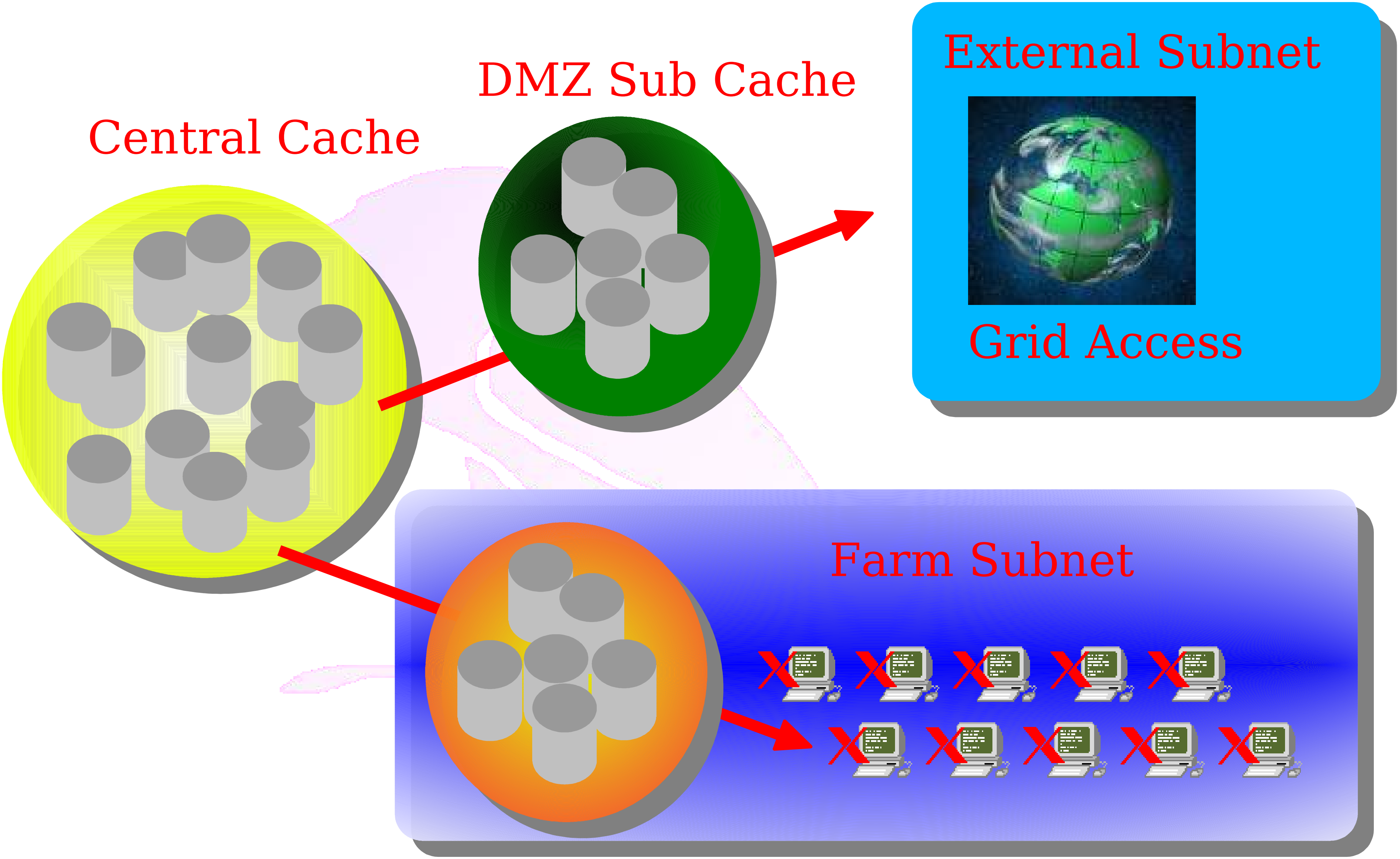
Support 'working group' quotas (select by storage class or tag)

Assign different number of pools to different working groups resp. 'data types' (raw,dst...)

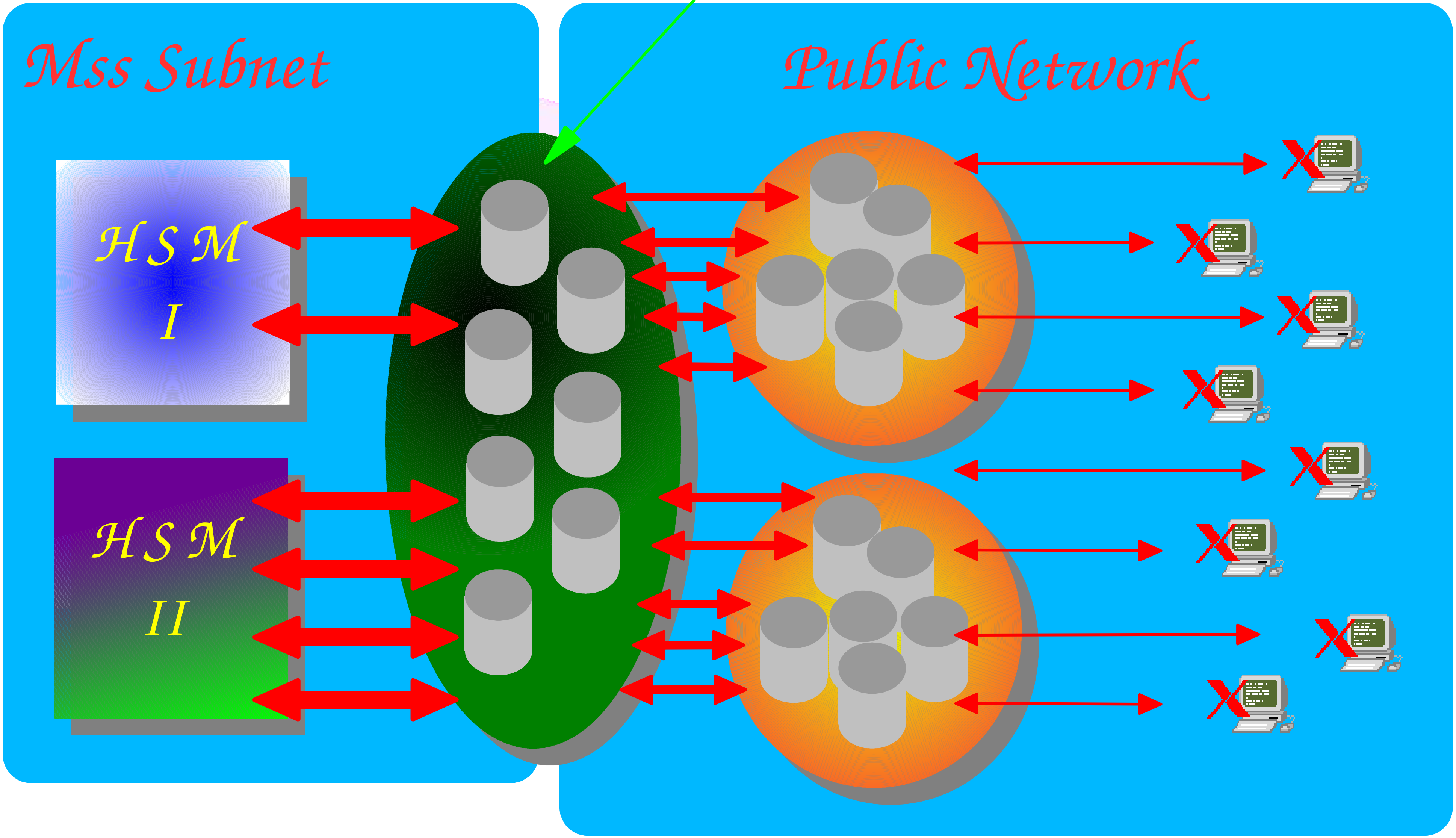
Special pools for farm subnets or external subnets

e.g. : Grid users vers. Internal users.

Selection by subnet



HSM decoupling



Dual Interface

One high speed link per drive

Dynamic pool selection

Method

Frequent update of 'pools vital functions'

- available space
- least recently used 'timestamp'
- number of movers (in,out,store,stage,p2p)

Performing 'smart' guess between updates.

Goal

Uniform (even) distribution of requests per pool
for requests coming 'in bunches'.

Pool Selection : Tuning (1)

Space vs. Load

For each request, the central cost module generates two cost values for each pool :

Space : Cost based on available space or LRU timestamp

CPU : Cost based on the number of different movers (in,out,...)

The final cost, which is used to determine the best pool, is a linear combination of Space and CPU cost.

The coefficients needs to be configured.

Space coefficient << Cpu coefficient

Pro : Movers are nicely distributed among pools.

Con : Old files are removed rather than filling up empty pools.

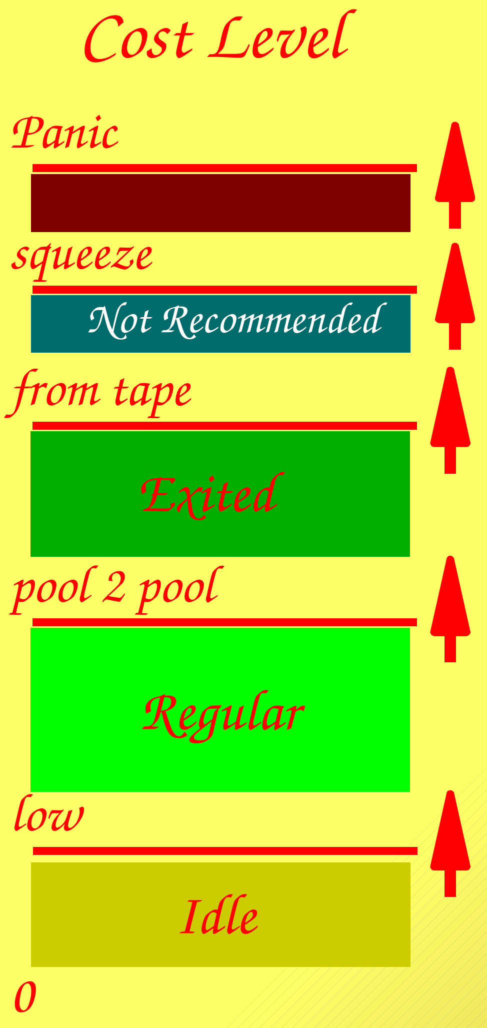
Space coefficient >> Cpu coefficient

Pro : Empty pools are filled up before any old file is removed.

Con : 'Clumping' of movers on pools with very old files or much space.

Pool Selection Tuning (2)

Pool to pool transfers etc. ...



PANIC

Take file from pool with lowest cost.

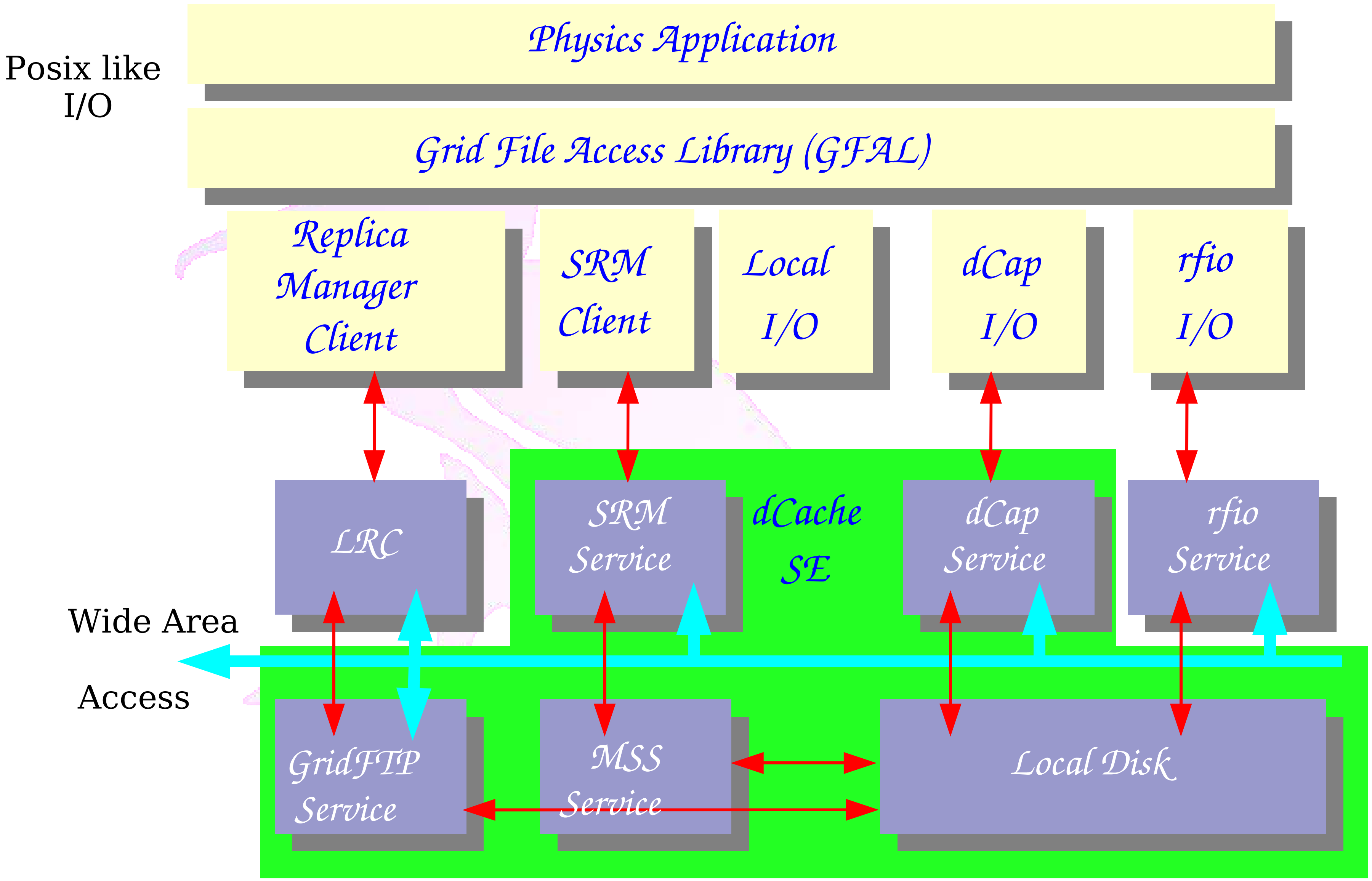
Fetch file from Hsm to cheap pool, before delivering to it to client.

Copy file to cheap pool first, before delivering it to client.

Take file from pool with lowest cost, otherwise try to get rid of duplicates.

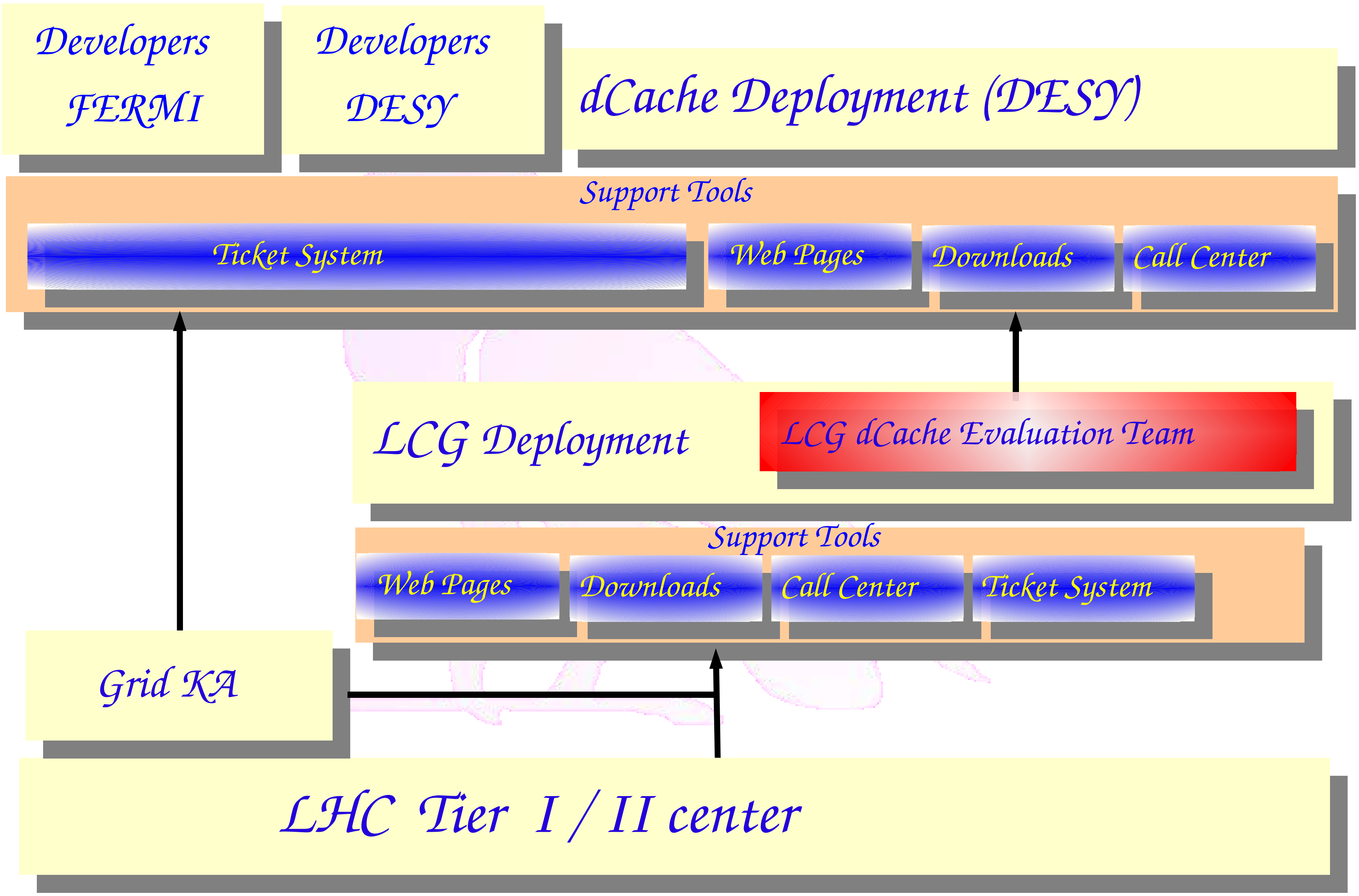
dCache
End of official presentation

LCG Storage Element : File access



Source : Michael Ernst 18/5/2004

dCache - LCG support model



Developers FERMI

Developers DESY

dCache Deployment (DESY)

Support Tools

Ticket System

Web Pages

Downloads

Call Center

LCG Deployment

LCG dCache Evaluation Team

Support Tools

Web Pages

Downloads

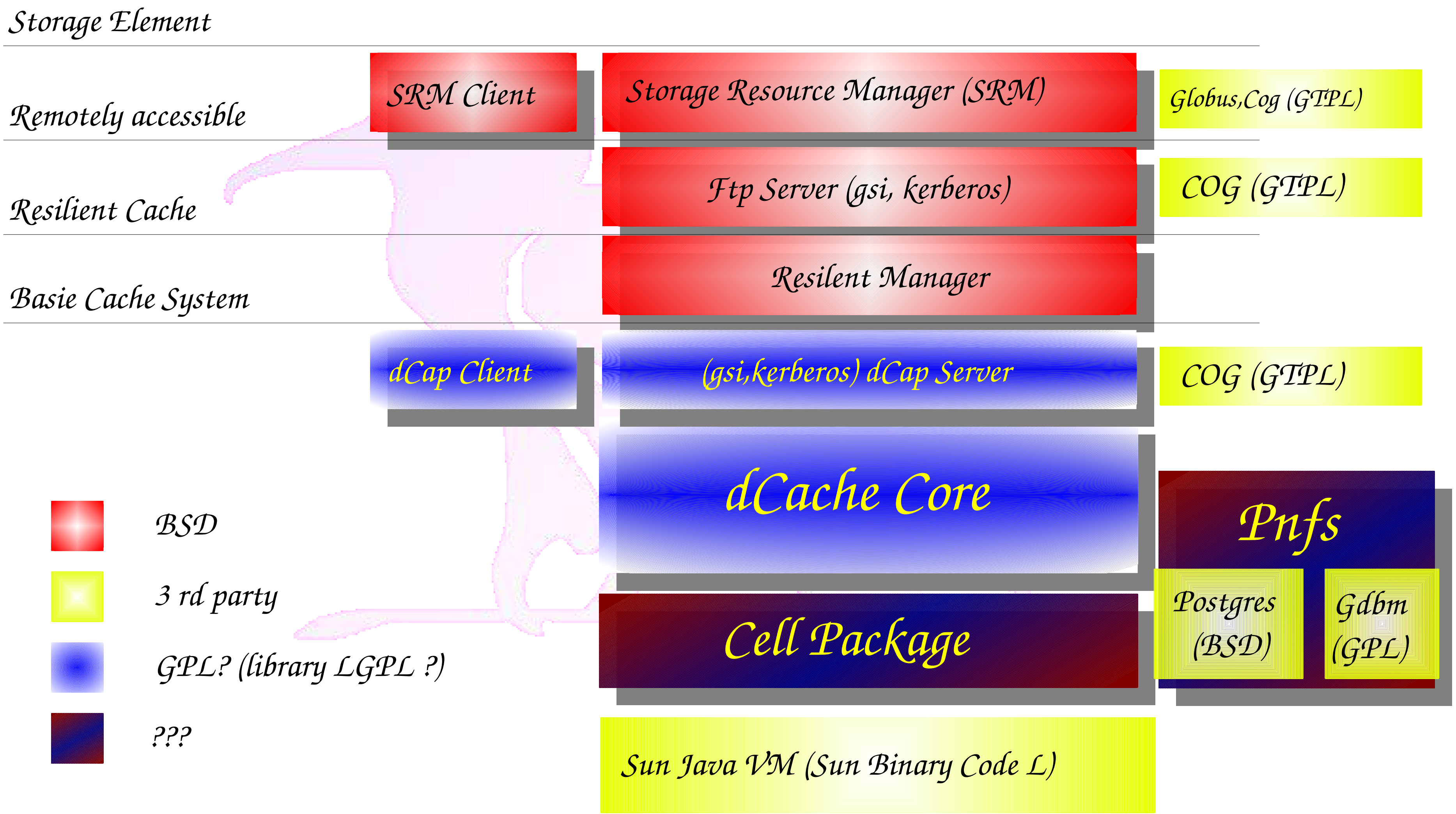
Call Center

Ticket System

Grid KA

LHC Tier I / II center

dCache Component License Model



Storage Element

Remotely accessible

Resilient Cache

Basie Cache System

SRM Client

Storage Resource Manager (SRM)

Globus,Cog (GTPL)

Ftp Server (gsi, kerberos)

COG (GTPL)

Resilent Manager

dCap Client

(gsi,kerberos) dCap Server

COG (GTPL)

dCache Core

Pnfs

Postgres (BSD)

Gdbm (GPL)

Cell Package

Sun Java VM (Sun Binary Code L)

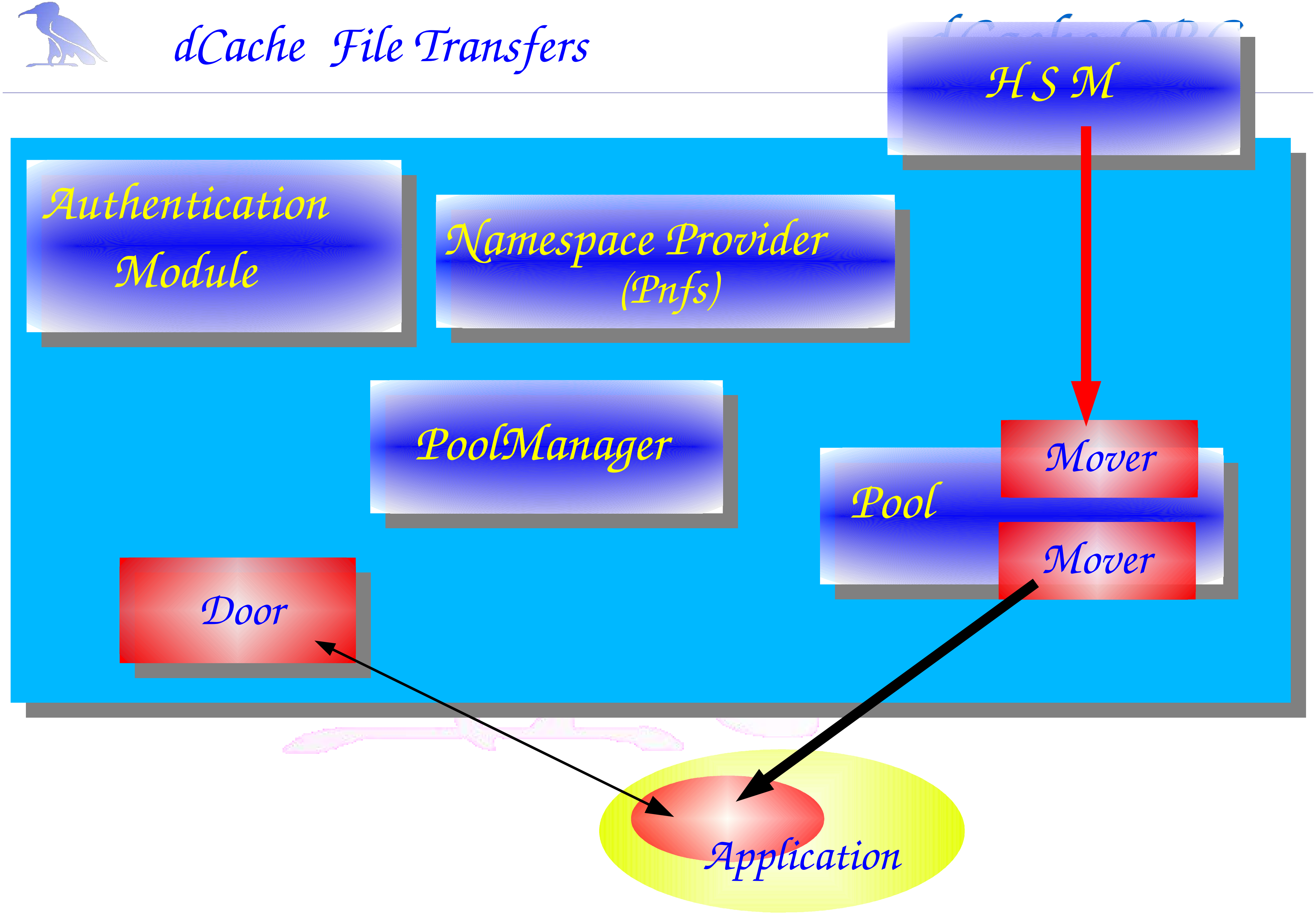
🟥 BSD

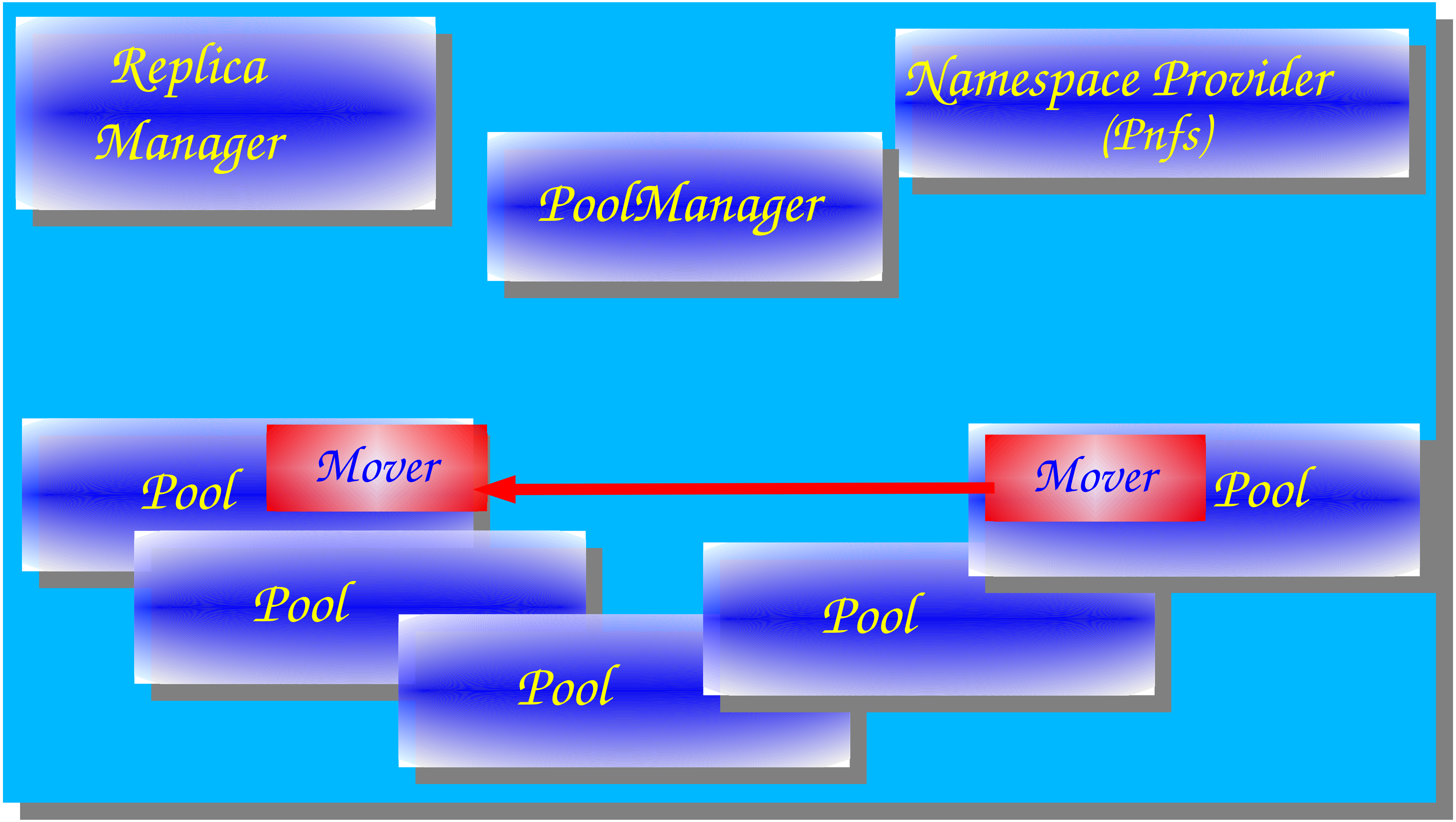
🟨 3 rd party

🟦 GPL? (library LGPL ?)

🟪 ???

dCache File Transfers




Replica Manager
PoolManager
Namespace Provider (Pnfs)
Pool
Mover
Mover
Pool
Pool
Pool
Pool