



Enabling Grids for E-scienceE

gLite FiReMan

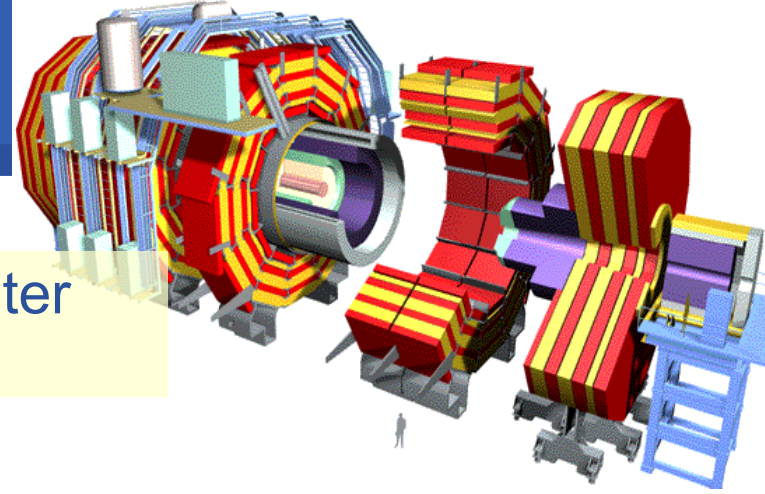
*Krzysztof Nienartowicz, CERN
On behalf of EGEE JRA1*

VLDB 2006

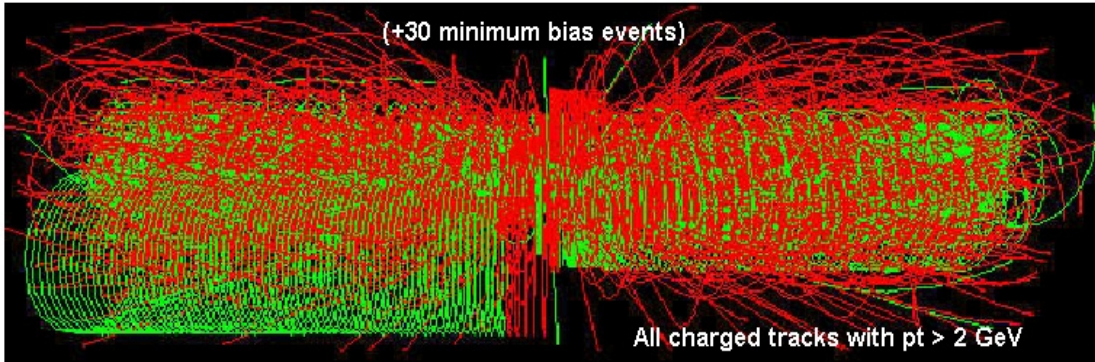
www.eu-egee.org
www.glite.org



- **What and why Oracle FiReMan was built for**
- **Interface driven functional requirements**
- **Hardware and use case driven speed requirements**
 - Feasibility study
- **Resulting approach and internal functionality**
- **Selected database techniques and features used**
- **Problems encountered**
- **Distribution**
- **Results**
- **Lessons for the future**



This is reduced by online computers that filter out a few hundred “good” events per sec.

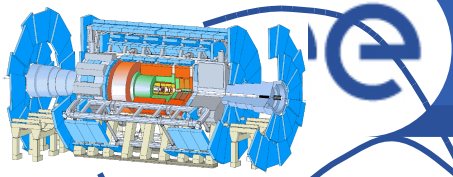


Which are recorded on disk and magnetic tape at 100-1,000 MegaBytes/sec → ~15 PetaBytes per year for all four experiments



Data Handling and Computation for Physics Analysis

Enabling Grids for E-science

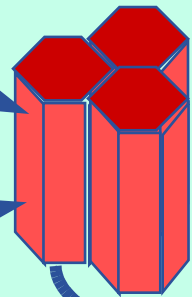


detector

event filter
(selection & reconstruction)

reconstruction

raw data

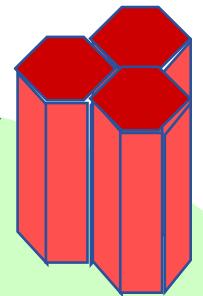


event reprocessing

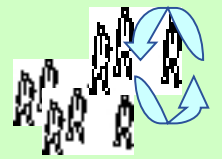


analysis

batch physics analysis



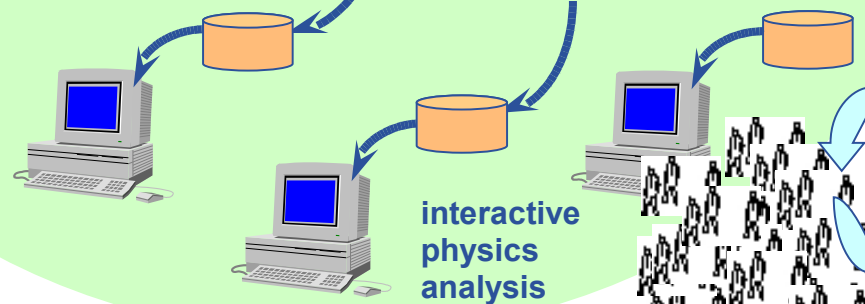
processed data



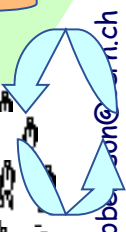
analysis objects
(extracted by physics topic)

event simulation

simulation



interactive physics analysis



- - FiReMan = File and Replica Management Catalog
 - EGEE aim to provide holistic catalog solution for Grid
 - Initially based on cross of HEP and biomedical requirements
 - Logical (LFN, GUID) <-> (SURL) physical mapping space as in EDG
 - Hierarchy as in Alien, AFS, GNS or other Grid cataloging solutions
 - Built on top of experiments' and industry experience
 - Hierarchical space as
 - LDAP
 - MS Active Directory
 - Ease of Distribution
 - Big and Fast fast fast

- **File Catalog and StorageIndex**

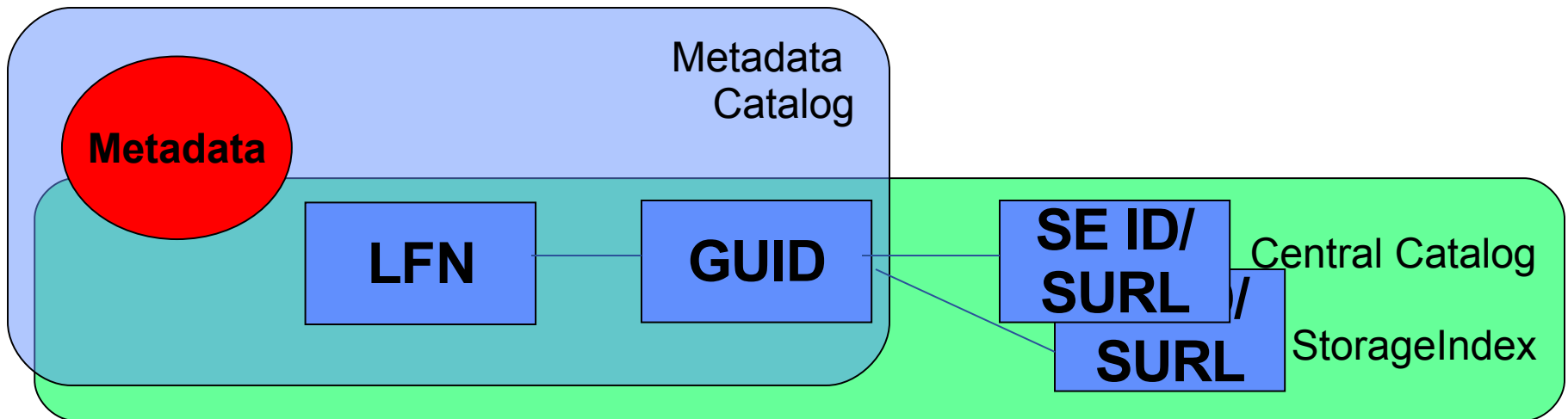
- Filesystem-like view on logical file names
- Keeps track of sites where data is stored
- Conflict resolution

- **Replica Catalog**

- Keeps information at a site

- **Meta Data Catalog**

- Attributes of files on the logical level
- Boundary between generic middleware and application layer



Redefinition of VLDB.

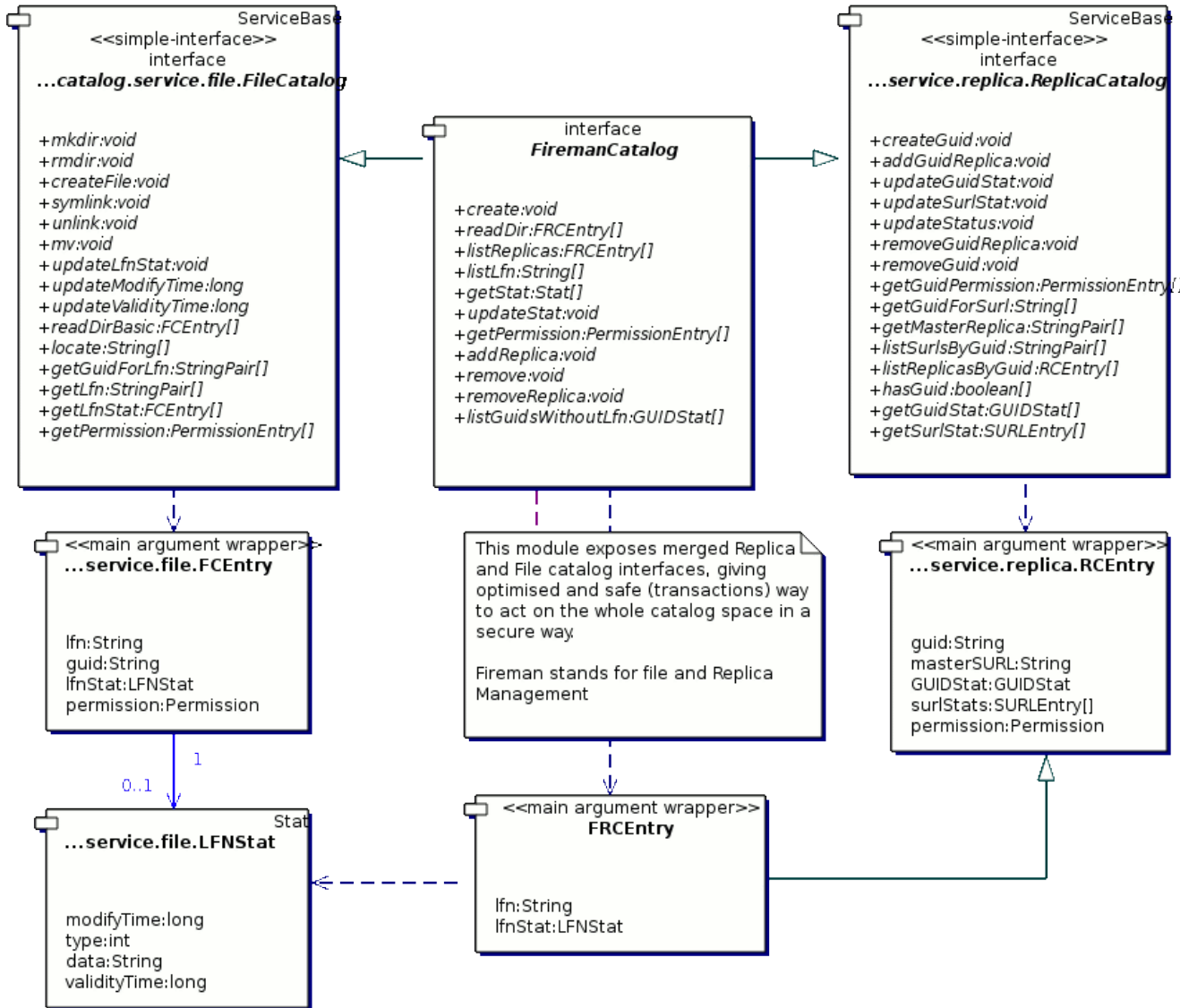
EGEE VLDB...

That is fast and scalable.

On our hardware.

*Say 10^{11} entries and thousands of users.
Simultaneously accessing database.*

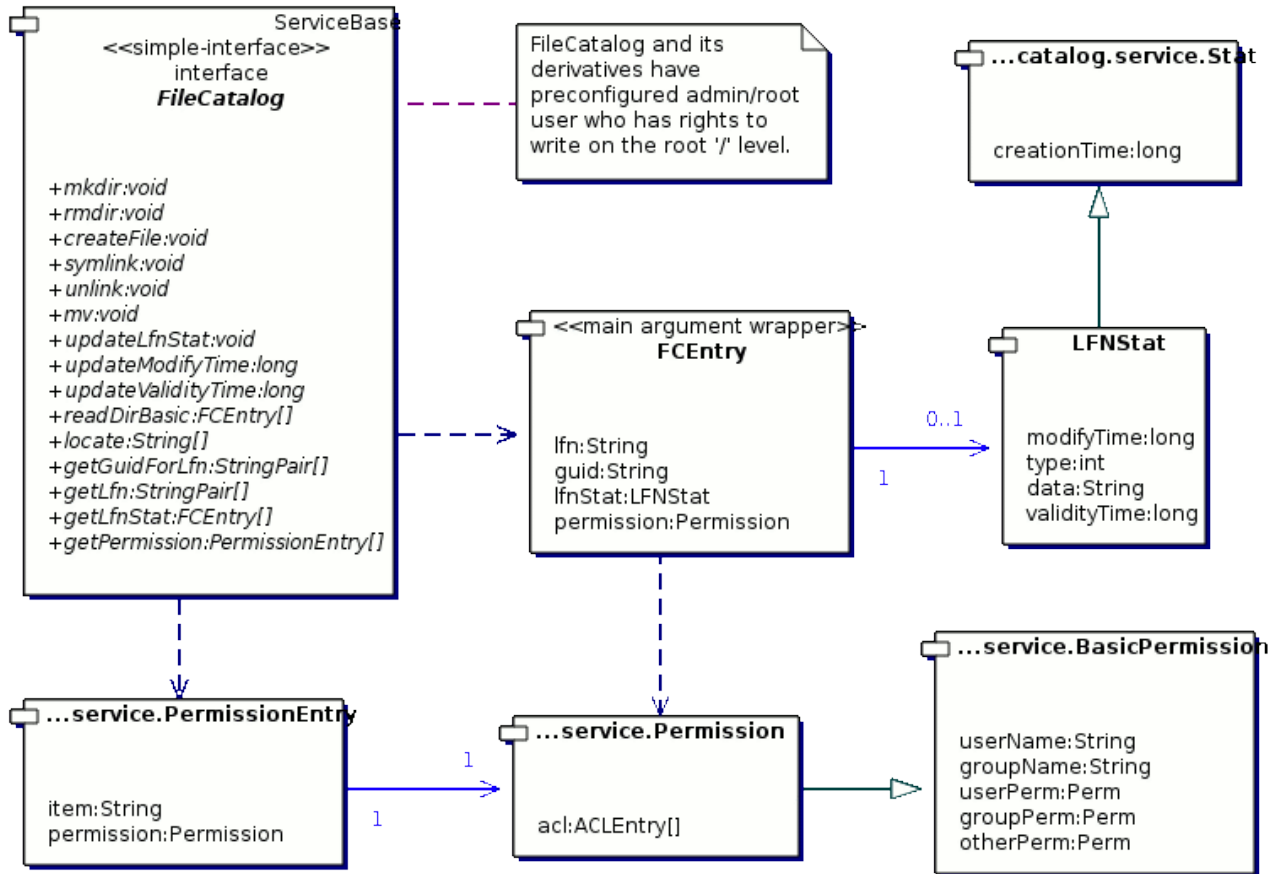
- 13. There is no magic to make MySQL originating things working faster on commercial databases, including Oracle. Formula is to use advanced, often proprietary features.*
- 15. Generic = Slower but Proprietary != Fast*
- 17. Gains from hardware upgrades are very costly, and surely not linear.
See TPC-C winning machines.*
- 19. Volume of data increased order of magnitude but not that much the number of users/transactions served by the commodity hardware.*



- Multiple shared types among different interfaces.
- FiReMan links LFN, Replica and Metadata worlds
- Lists of objects are both parameters and are being returned

- **50 methods**
- **16 types**
- **Dynamic hierarchical space**
 - Create entries
 - Remove
 - Update
 - Rename
 - Readfor LFN or replica/security/metadata dimensions
 - Nested symlink support with checking for cycles
- **Administrative or on the fly creation of users – VOMS dependency**
- **Superusers**
- **Ownership and permission context part of the core system**

- **Bulk operations.**
 - Transactionally safe
 - Stateless
 - Windowing operations for big results

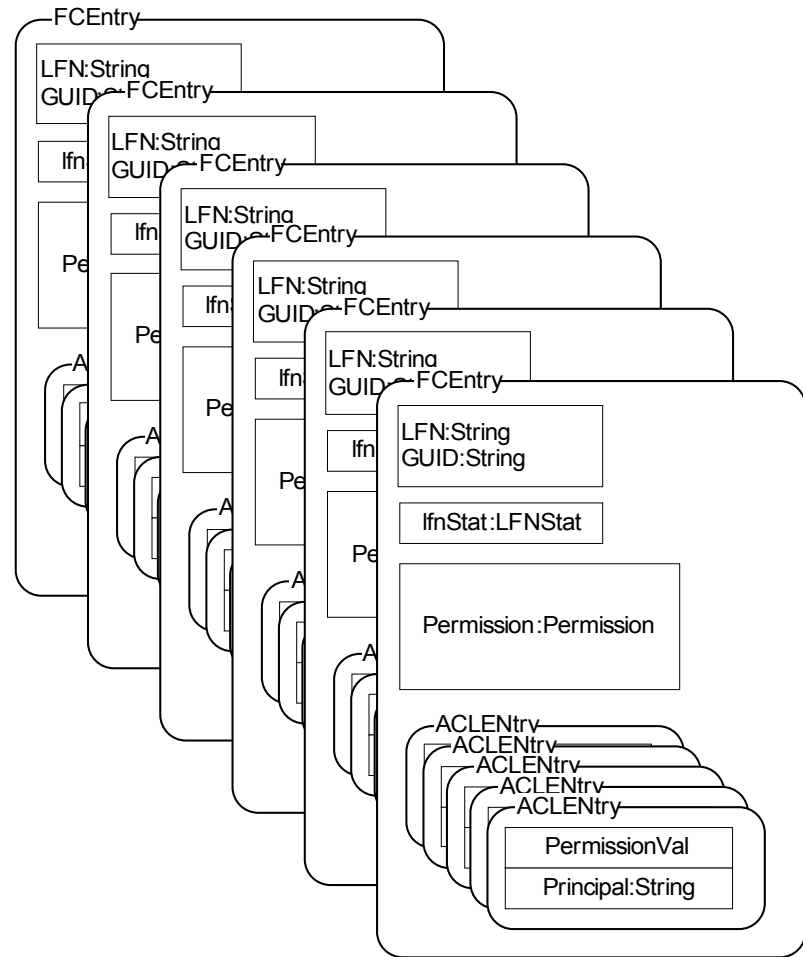


- No constraints on ACL length
- No constraints other than system capacity on input/output parameters
 - i.e. LFN max ~32KB
 - No levels constraint
 - No constraints on metadata attributes number
- Surely not relational/dataset model

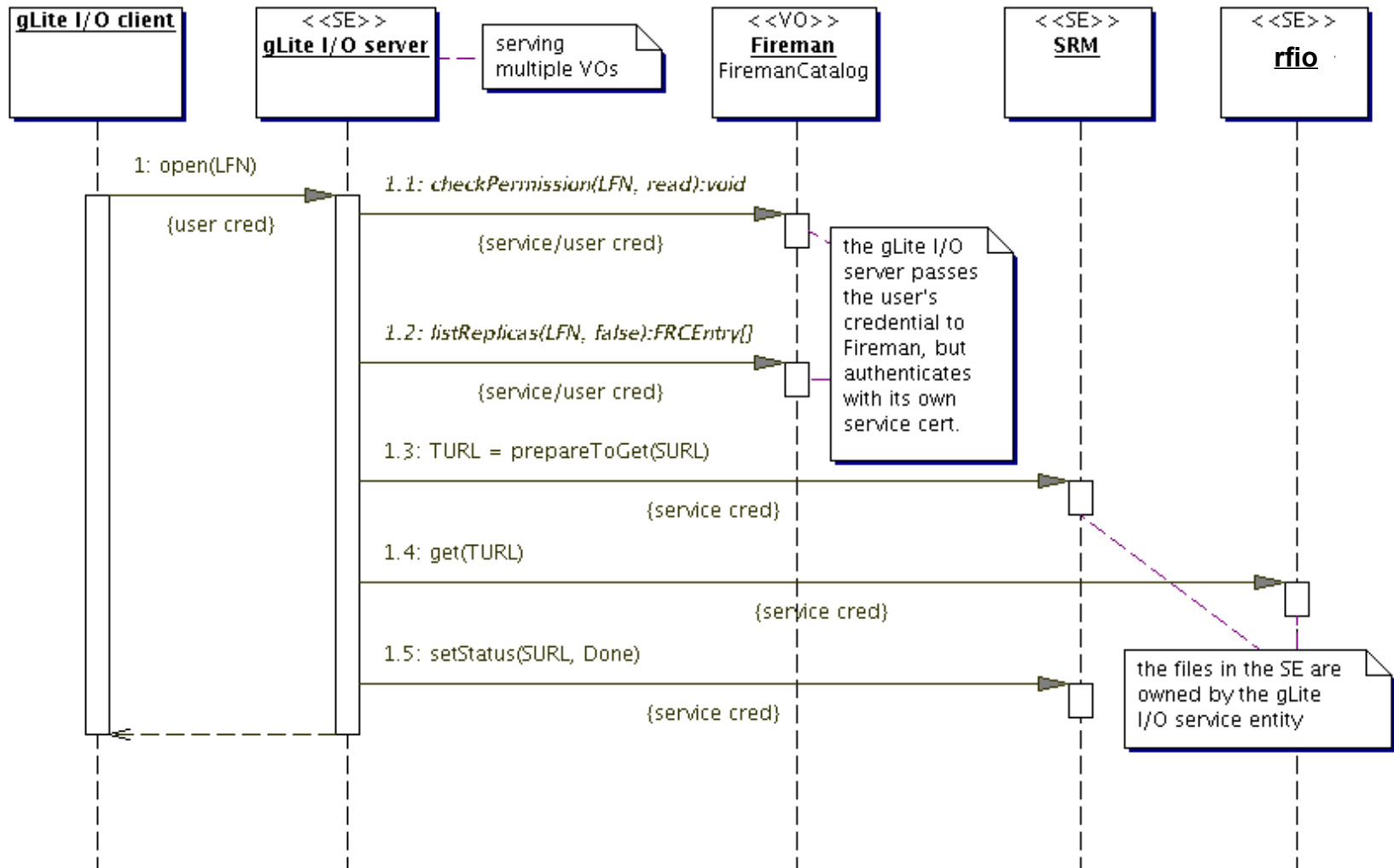
- Old impedance mismatch problem
 - Especially if we want to have bulk operations
- We do not want to do JOIN by ourselves – it is work for database
 - Flattening is bad either way
- Solution: Propagate complex objects down to the database and process them there naturally

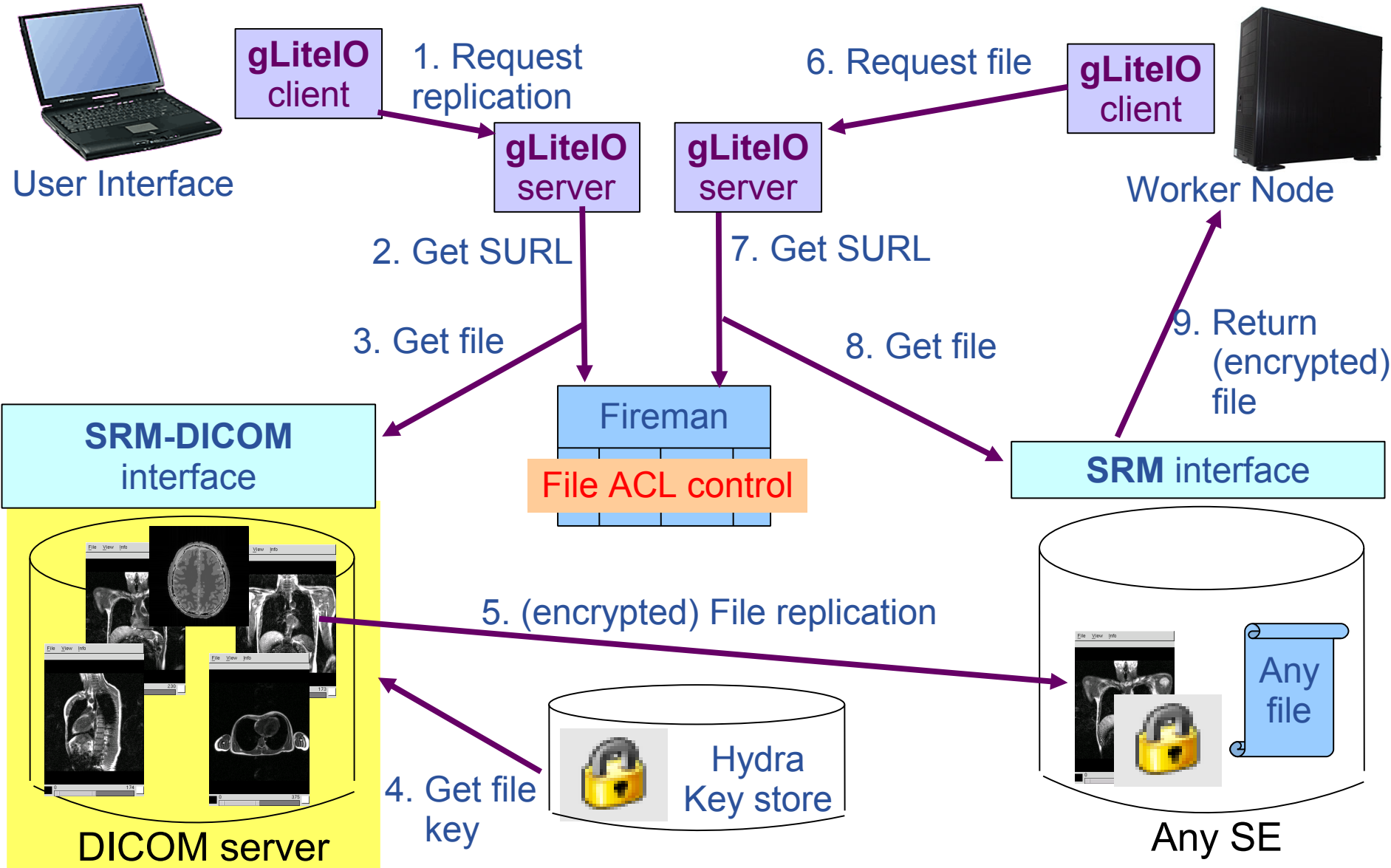
- Why interface (API) we define is important for persistent data modeling?
 - List of complex objects
 - With complex objects embedded
 - With lists of complex objects embedded
 - With list of objects

Because it may complicate our lives....



File access when the files are owned by a single entity in the Storage Element.
The access control is enforced by the gLite I/O server.





Database machine

CERN diskserver

- **Dual Intel(R) Xeon(TM) CPU 2.4 GHz, 512kB cache**
- **2GB**
- **Roughly 1000GB available for database**
 - RAID0 – 9 mirrored disks
 - *Almost* standard CERN Oracle configuration
 - 16k db_block_size preferred, with buffers db_16k_cache_size set
 - some tuning on db_file_multiblock_read_count
 - Still space for low level DB process tuning
- **Orders of 10^8 – hierarchical entries in the catalog possible volume-wise**
 - Plus related security info and replica info, min. additional single entry doubles the number $5 \cdot 10^8$ to over 10^9 .

- **Feasibility study**

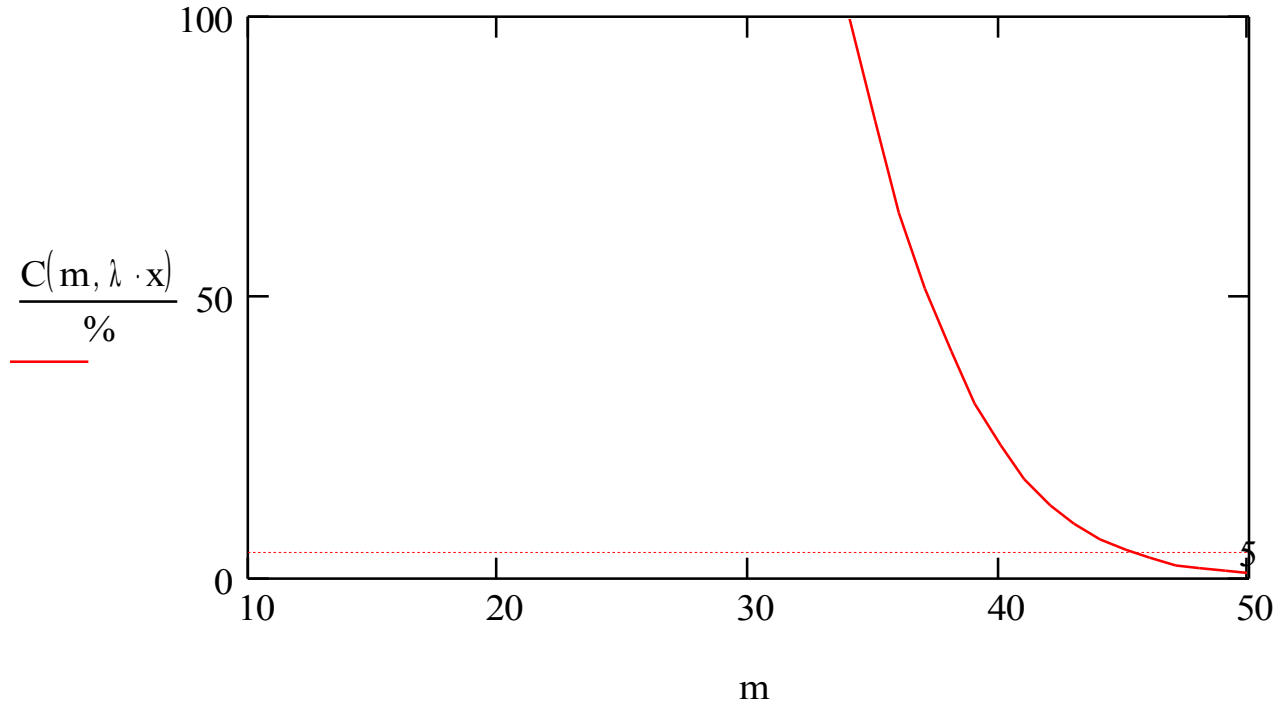
- Massive System Servicing performance evaluation
- M/M/1 queueing model for simplicity just for database backend
 - Look for limiting service figures based on experience and requirements
 - *CMS document*
 - *Compass experience*
 - *Volumes of data expected*
 - *Guessing, intuition... ☹*
 - Only database considered as application servers easier to multiply
 - *Real bottleneck in a longer term is a database not a protocol or security overhead*



number of DB processes

$$m := 10..50$$

Probability of timeouts



DB processes with stable response time

Average users call rate during the heaviest load:

$$\lambda := \frac{200}{\text{sec}}$$

Average time spent on servicing DB call, assumption: bulk calls, 10-250 items per call:

$$x := 170\text{ms}$$

Average time spent on servicing DB call, assumption: bulk calls, 10-250 items per call:

number of database processing slots

$$m := 50$$

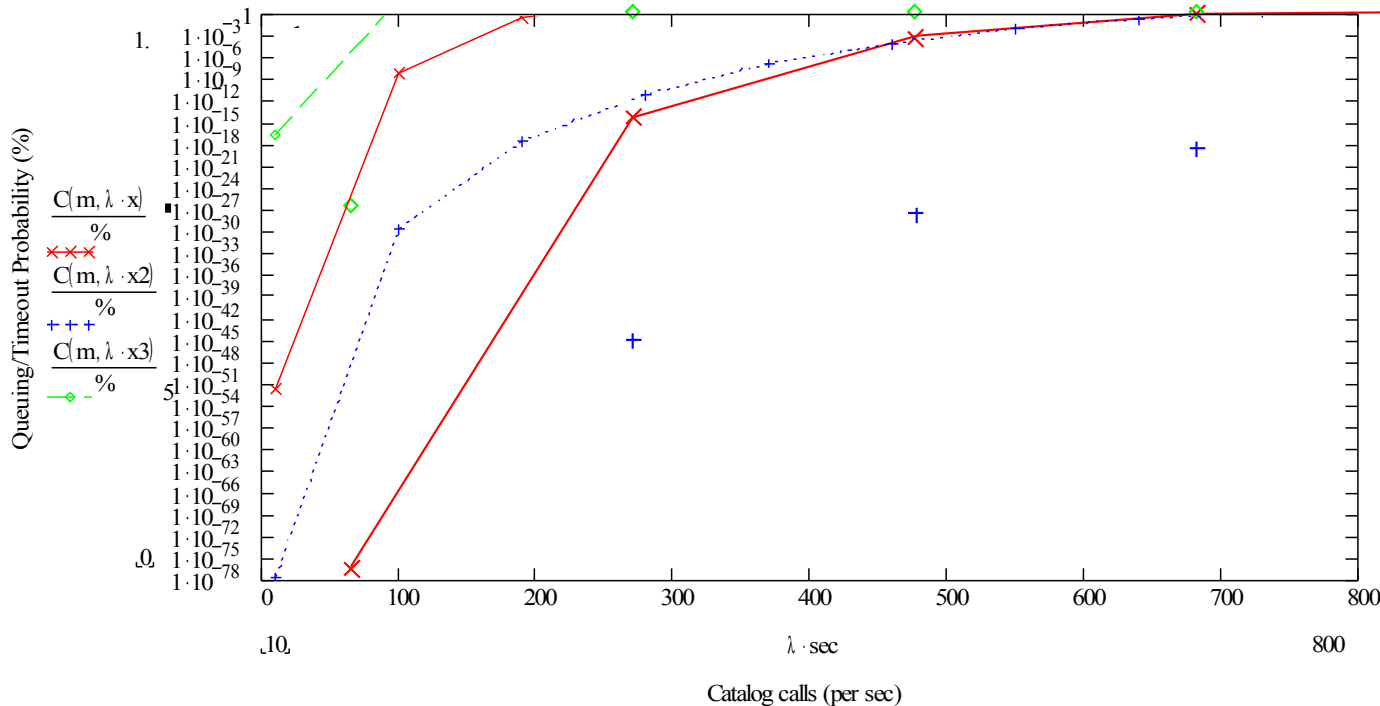
average users call intensity during the busiest time

$$\lambda := \frac{10}{\text{sec}}, \frac{100}{\text{sec}}, \frac{500}{\text{sec}}$$

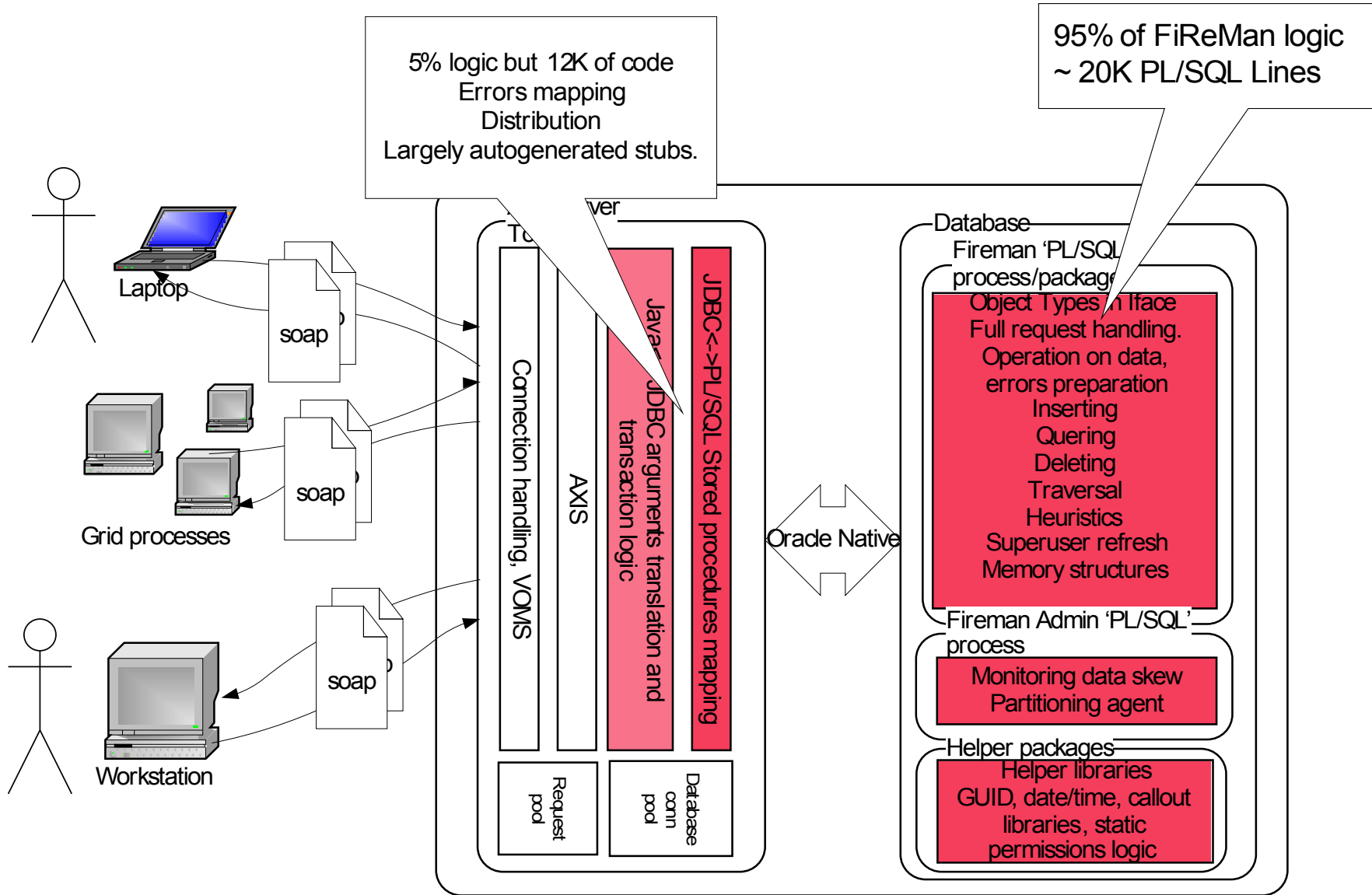
$x_1 := 170\text{ms}$ for ~50-500 bulk calls

$x_2 := 50\text{ms}$ for ~1-50 bulk calls

$x_3 := 1\text{sec}$ heavy load, 500-..bulk calls



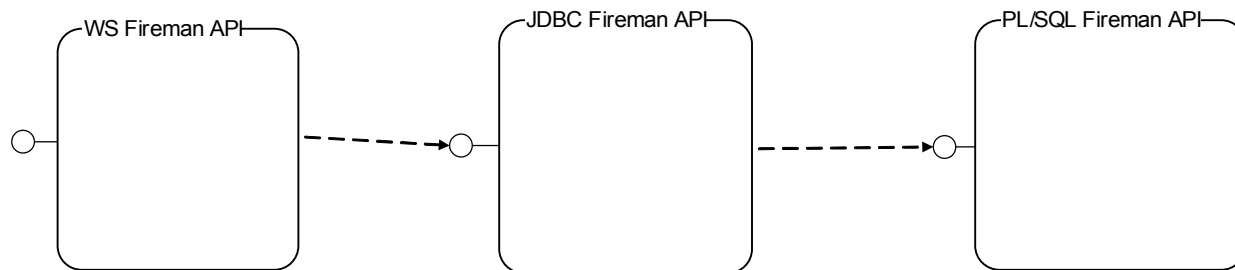
- ××× Average servicing speed 170ms
- +++ Fast servicing for small calls 1-15 items, 50ms
- ◇ - Slow processing speed while DB is overloaded or v. big in/out 1



- **Unique features**

- Internally implemented partial errors reporting - no speed loss.
 - *If 3 items out of 1000 failed during one call, no rollback is necessary.*
 - *Remaining 997 items can be committed or read at user will.*
- Fast LFN renaming with negligible cost regardless of position in the hierarchy
- **Flat behaviour – same results with $10^6..10^8$ items**
 - Very big number of catalog principals—thousands feasible
 - Unlimited size of ACL
 - Unlimited size of metadata attributes
 - Unlimited number of superusers
 - Consistency of users, items and permissions
 - *Fast removal of all pertaining ACLs if user/group removed*
 - Easy reporting on ACL layout
 - Ability to amend default behaviour by users
 - *Ownership of newly created items, home directory, dangling GUIDs handling*
- Functionality separated from protocol
 - Web Service, pure-JDBC, C++ possible

- **Three libraries**
 - AXIS-WS SOAP
 - 50 methods
 - JDBC Side
 - Web Service complex types mapped onto JDBC representation
 - 50 methods
 - PL/SQL Side
 - Web Service complex types mapped onto SQL representation
 - 50 methods
 - Client libraries, often with autogenerated custom wrappers WSDL->(Fireman XSLT)->client stub
 - Java
 - C/C++
 - Perl, Python
 - JavaScript

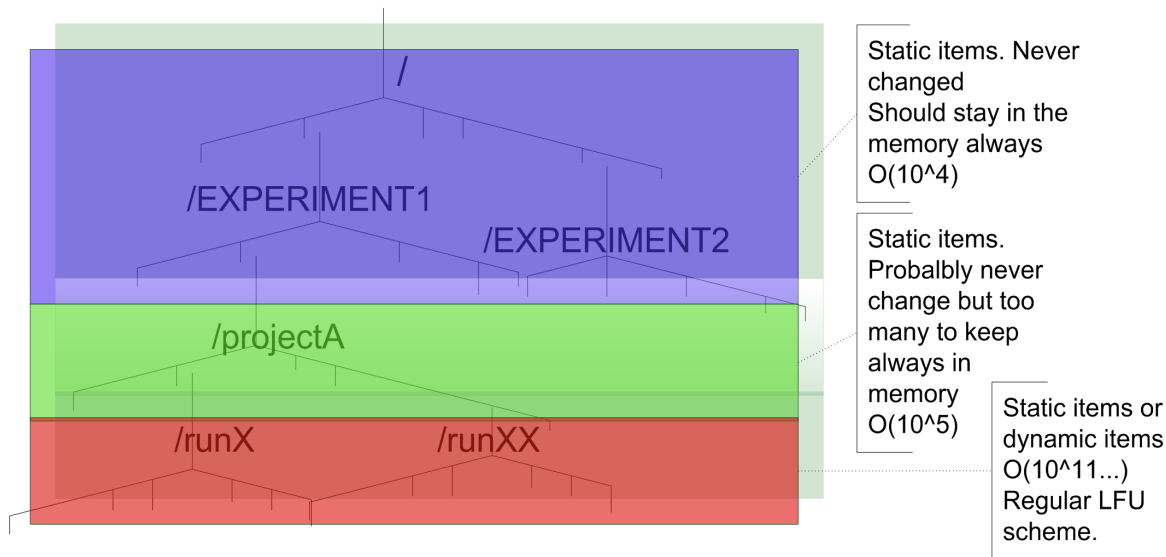


- **How to achieve these speeds and functionality**

- Hierarchy slicing
- Hierarchy traversal
- Hierarchy clustering

- Database cache utilisation/invalidation during traversal
- Clustering during DB reads

- /VO1/experiment1/subexperimentA/run2
- /VO1/experiment1/subexperimentA/run2
- ...
- /VO1/experiment1/subexperimentA/run2/user1
- ...
- ...



- Clustering
 - LFN traversal is much faster
 - Wise caching utilisation
 - Index organized table
 - Plus compression

	parent	id	name	Some data	
Block 1	-	1	glite	Other data	100%
	1	20	myVO.org	Other data	
	20	300	production	Other data	
	300	4000	run	Other data	
Block l	4000	50000	07	Other data	15%-100%
	50000	56000	123456	Other data	
	56000	57000	calibration	Other data	
Block k	57000	800000	cal	Other data	? – but all fetched at once
	800000	100000000	cal-table100	Other data100	
	800000	100000001	cal-table101	Other data101	
	800000	100000002	cal-table102	Other data102	

- **To get functionality with minimal CPU, Disk activity**

- **CONNECT BY**

```
select sys_connect_by_path(name,'/') path, f.*
  FROM
    FC_FILE f
  START WITH parent_id=0 AND file_id=1
    CONNECT BY PRIOR file_id = parent_id
              AND NAME=fireman.getpathtoken(LEVEL-1,path)
```

- **To diminish size of joins or memory usage**

- Functions in query

- In-memory collections

- TABLE() operand

- Known heuristics to limit selectivity

- *chdir* behaviour within single operation

- *Token read clustering* so no need to traverse same path in a single operation

- **Both CPU and memory**

- Non-relational resultsets

- Conditional paths

- **Use bulk PL/SQL**

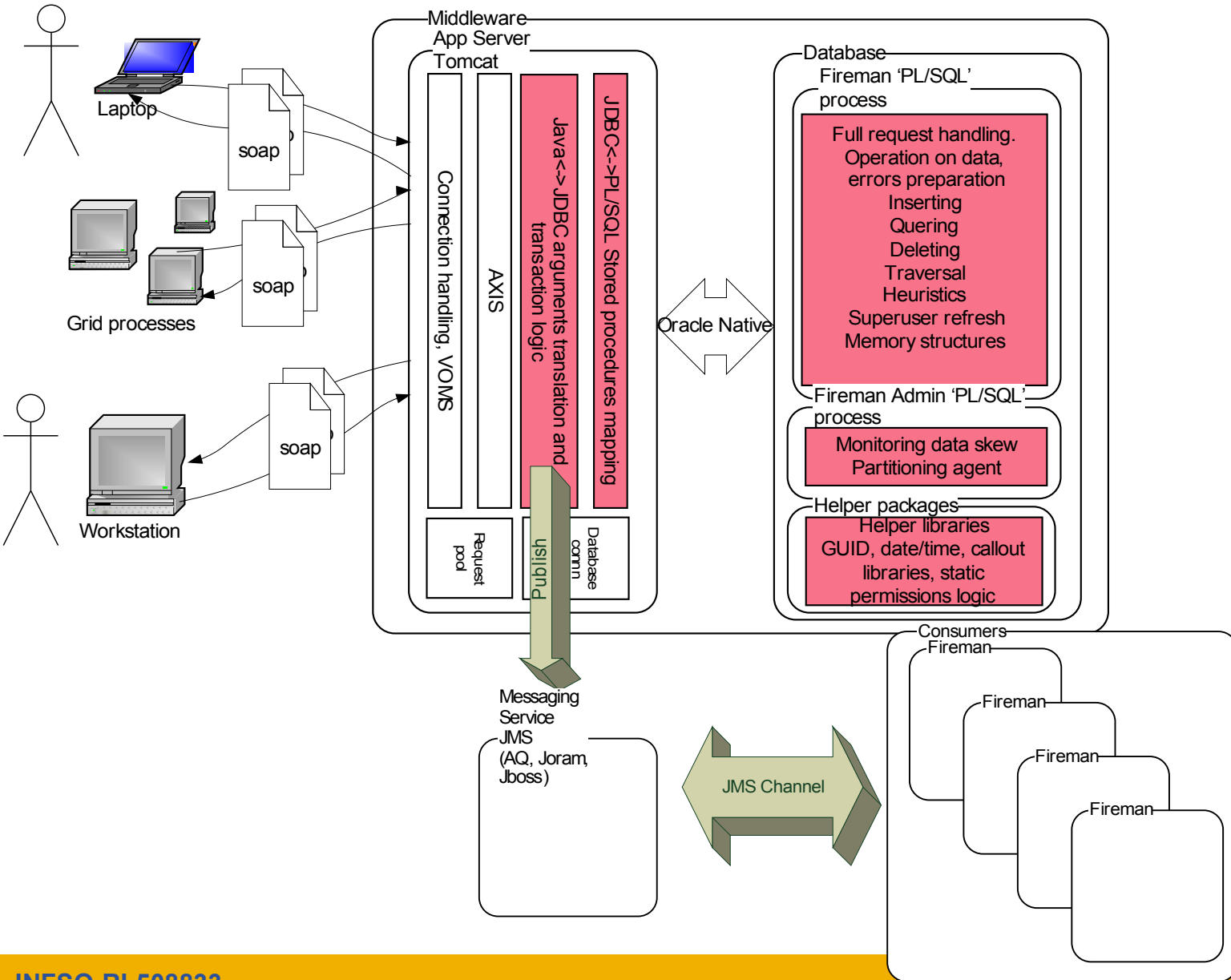
- Its error reporting


```
select * from TABLE(getDirectoriesLFN(inputCollection));
```

- **Good choice if we don't care about CPU stress**
 - They introduce object or procedural feeling to data flow withing queries and ease code maintability
- **They are not for free**
 - They cost a SQL<-> PI/SQL context switch
 - 5-50% slower then subqueries
 - There are Oracle bugs...

- **Thanks to Index Organized Tables data is naturally clustered**
 - Partitions are based on adjacency and traversal level level
 - Partitions can be created by administrative process running in the background
 - 300M entries tested
 - ~800M database entries (*LFN, Replica, ACL, Metadata*)
 - ~10K..10M file items in one partition
 - For the hierarchical space - *list partitioning by value*
 - parent<->child relationship being a key
 - Clusters of keys (parent_id,file_id)
 - Some data reorganization plausible
 - *For GUID driven spaces*
 - Hash partitioning

- **Architecture**
 - Publisher
 - Publishing code embedded in the application server
 - *Needs common transaction boundary*
 - Consumer
 - Specialized process that subscribed to the Topic
 - *Single Topic currently*
 - Based on standard JMS
 - Freedom of topologies, vendors
 - Publishers tested on GNU Joram and Oracle AQ



- Infamous ORA-600**

Problem	Present in	Bug reference	Fixed in (as of Jan 2006)
Internal errors when debugging pl/sql from JDeveloper on 9iR2	9.2.0.x		10g
ORA-600 [VOPRVL1] IN QUERY HAVING AN LEFT OUTER JOIN USING OBJECT TYPES	10.1.0.3	4260205	patch 4260205
Recycle bin feature causes the database to crash the machine	10.1.0.3		
ORA-7445 [KOKQAGO] BEING HIT IN QUERY USING CAST/MULTISET	9.2.0.6	4288071	9.2.0.7
ORA-600 [KKXMCPLS2] BEING HIT DURING SELECT	9.2.0.6 10.1.0.3	4312416	no fix available
ORA-600 [KOKEEAF14] BEING HIT IN OBJECT ORIENTED QUERY	10.1.0.3	4327752	10.1.0.4
PSRC: ORA-600 [QKAFFSINDEX3] AFTER APPLYING 10.1.0.4	10.1.0.4	4457034	patch 4457034
WRONG RESULT WITH ALIASES IN NESTED SQL WITH A FUNCTION CALL	10.1.0.x 10.2.0.x	4695283	no fix available

Source Michał Kwiatek

WRONG RESULT WITH ALIASES IN NESTED SQL WITH A FUNCTION CALL

*** 10/24/05 01:32 am ***

PROBLEM:

 Wrong result with aliases in nested sql with a function call.
 Gives expected result in 9.2.0.7 but not on 10.1.0.4 and 10.2.0.1.
 Customer needs a fix on 10.2.0.1.

DIAGNOSTIC ANALYSIS:

 Results got with query below :

```
SQL> select a per, a p, a p2 from (
      2 SELECT (dummyBug2051014(1,2)) as a
      3 FROM
      4 (
      5 SELECT /*+cardinality (i 20) */ *
      6 FROM
      7 TABLE(genLfns())
      8 ) i);
```

9.2.0.7 :

 PER P P2

PER	P	P2
3	3	3
3	3	3
3	3	3

Execution Plan

 0 SELECT STATEMENT Optimizer=CHOOSE (Cost=11 Card=8168 Bytes=16336)
 1 0 COLLECTION ITERATOR (PICKLER FETCH) OF 'GENLFNS'

10.1.0.4 :

```

-----
.
  PER    P    P2
-----
      3
      3
      3
.
  
```

Execution Plan

```

-----
 0  SELECT STATEMENT Optimizer=CHOOSE (Cost=25 Card=8168)
 1  0  COLLECTION ITERATOR (PICKLER FETCH) OF 'GENLFNS'
.
  
```

Idem with CHOOSE mode
(...)

WORKAROUND:

```

-----
None found
.
  
```

RELATED BUGS:

```

-----
None found
.
  
```

REPRODUCIBILITY:

```

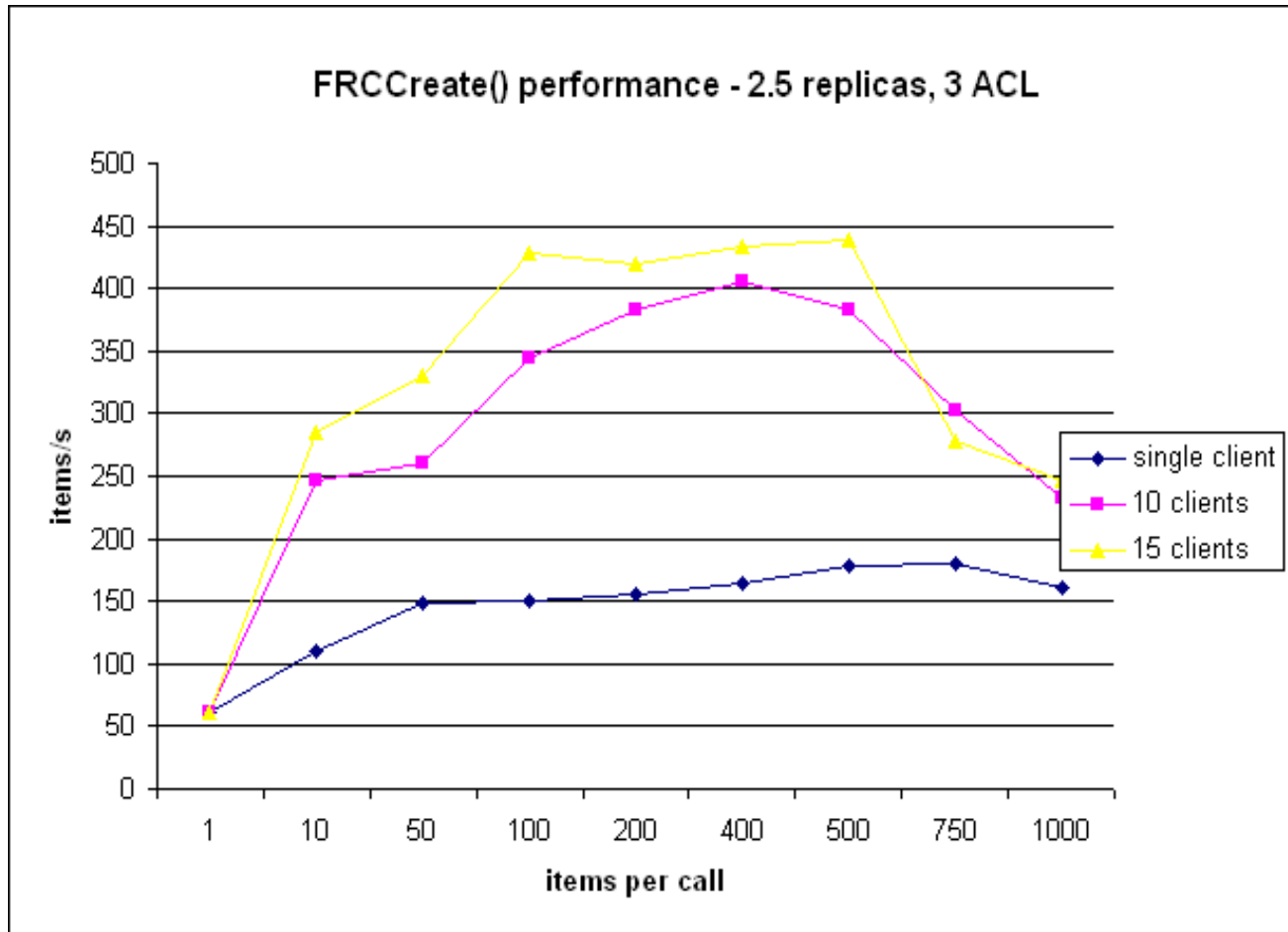
-----
9.2.0.7 : No
10.1.0.4 : Yes
10.2.0.1 : Yes
.
  
```

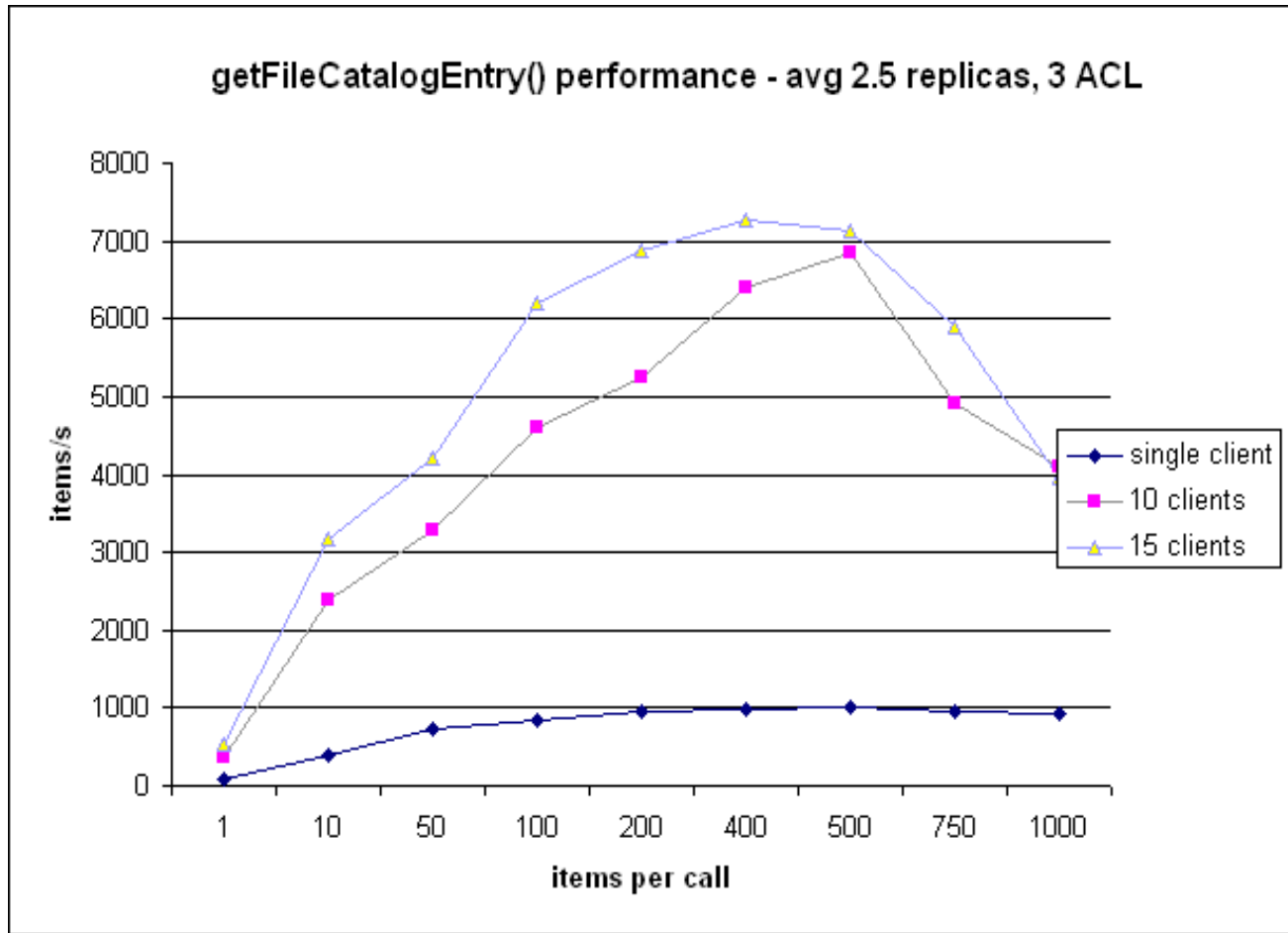
TEST CASE: (...) and oracle changelog follows (**no fixes to report** ...⊗⊗⊗)

Raw database speeds without Application Server wrappers via JDBC

LFN dependent – volume independent – same results for 1M and 250M entries

- **getFileCatalogEntry of bulk 10 items**
 - 400 full FRCEntries/sec per client
 - ACL retrieval, permissions check, symlink traversal
 - Aggregated by 10 clients, up to 3000 items/sec
- **getFileCatalogEntry of bulk 500 items**
 - 1000 full FRCEntries/sec per client
 - ACL retrieval, permissions check, symlink traversal
 - Aggregated by 15 clients, up to 7000 items/sec
- **Insertion stable at around 100-400 creations/sec**
- **Security overhead – 10KB per call, four extra messages**





- **Even Grid protocols/SOA could be expensive the real cost is in the DB, as usual**
- **Try to limit *impedance mismatch***
 - By designing flat interface, easily transformable on the way to the database
 - or go for object model while talking to DB.
 - It's a standard defined by SQL99 and supported by number of other DBs (DB2, PostgreSQL,...)
 - *Not all bindings are supported – always Java, usually C++*
 - It saves memory and CPU on both client and DB ends
 - Not always standard JDBC mapping is the best. (see jPublisher)
- **Cluster persistent data**
 - Use data model that takes fully what Oracle gives when it comes to clustering
 - Index organized tables - IOTs
 - *If suitable*
 - Partition data and indices
 - Not always *natural partitioning* solves all problems (*timestamp not a access pattern key*)

- **Use selectivity knowledge**
 - Cardinality and other hints
 - NESTED_LOOPS vs HASH_JOIN
- **Scalability facts**
 - Development, testing volume
 - Use real volumes, even this is not enough (data skew triggered errors)
 - Test coverage
 - Volume-wise, user#, DB version, data distribution characteristics
- **Demystified Oracle**
 - Tool with thousands of unique features
 - And same bunch of problems
 - some very painful to track down and to alleviate with elegant workaround (on time)
 - OO and PL/SQL toolset below expectations



Lightweight Middleware for
Grid Computing