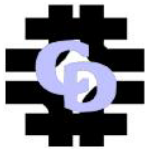
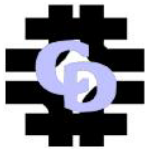


Fermilab

Database Applications Deployment Case Study

Anil Kumar
CD/CSS/DSG

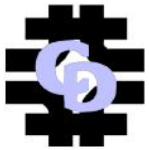
Outlines



Fermilab

- Background
- Challenge
- Solution
- Case Study Details

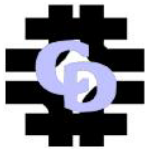
Background



Fermilab

- CSS-DSG develops, deploys and supports databases, and does system administration for hardware for Experiment and Lab Users.
- Developments are structured as projects - invariably involving participants from more than one department in the Computing Division and usually in collaboration with the end users and often with Project Leaders outside of the Department.
- We focus on Data Handling and Management: databases for information and event data management; distributed data management and access; system administration of multiplatform experiment hardware.
- CSS-DSG is working on projects with CDF, D0, CMS, BTeV, MINOS and beyond.

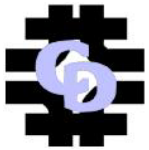
Challenge



Fermilab

- To deploy tuned and integral database schemas to production ensuring consistency and integrity in data model design.
- Proactive approach in understanding the requirement and fine tune the data design.
- Performant databases.
- 24*7 Support for Databases.

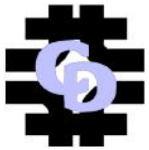
Solution



Fermilab

- Setting the hardware infrastructure including disk layouts for database storage.
- Data Modeling
- Server Tuning
- Query Tuning
- Load balance via replication
- Tracking of access and resources used to help troubleshot tuning
- Exiting the inactive sessions to db
- Capacity planning

Case Study Details



Fermilab

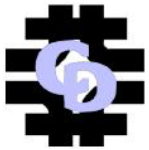
- Specific Experiments for Case Study
- Hardware Infrastructure for Database(s)
- Data Modeling
- Deployment
- Tuning
- Load Balance Via Replication
- Monitoring
- Special Features
- Summary

Specific Experiments for Case Study

Fermilab

- D0
- CDF
- CMS
- MINOS

Hardware Infrastructure for Database(s)



Fermilab

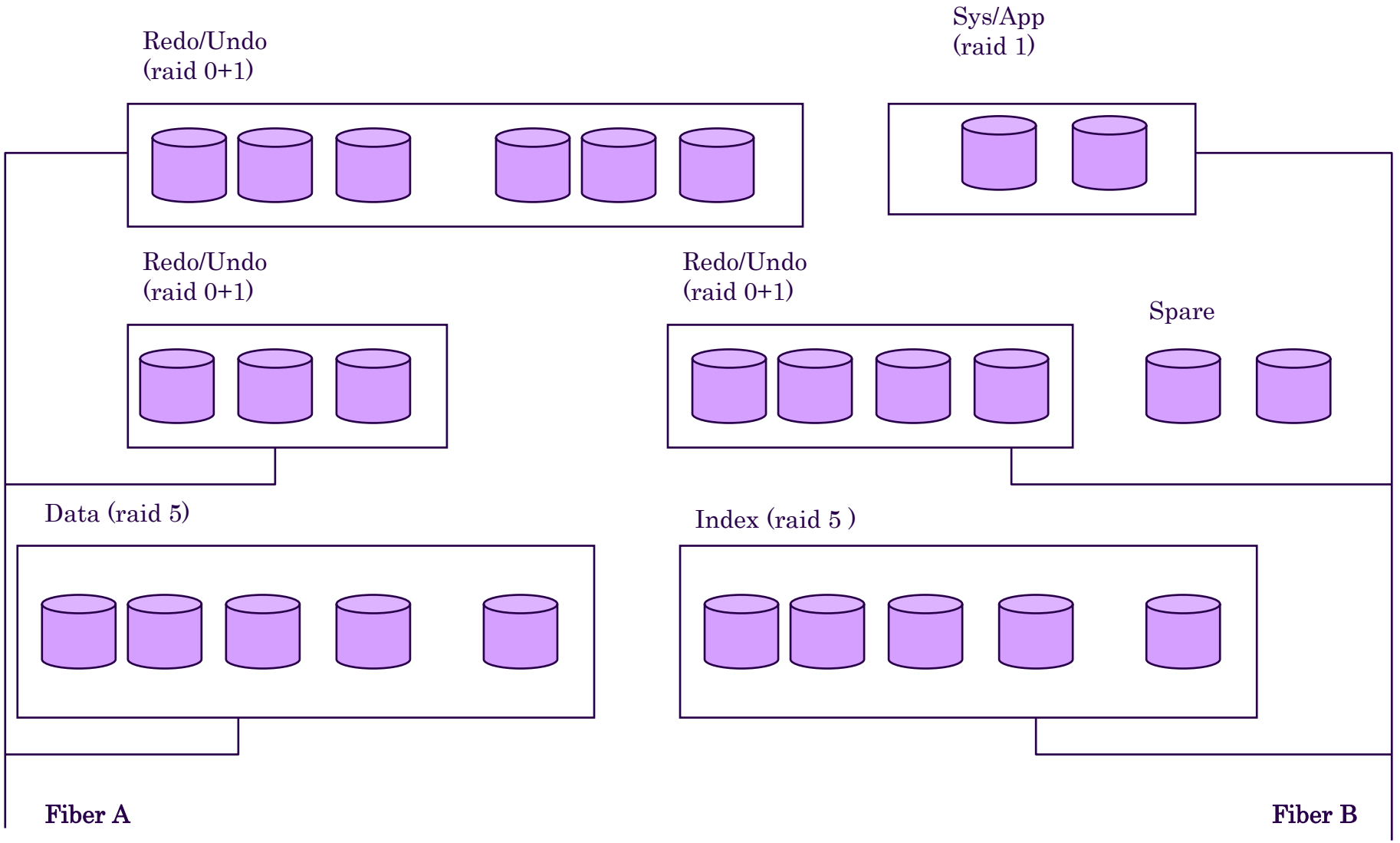
- Hardware is selected based upon the application's size , growth and performance requirement.

- Database is deployed keeping in mind the OFA

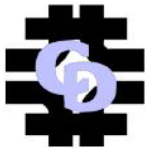
(Optimal Flexible Architecture of Oracle)

- See D0 Off-line Database Disk Array Diagram Fig 1.

D0 Off-line Database Disk Array Diagram



Data Modeling



Fermilab

- Data Modeling tool being used is Oracle Designer.
- There is central repository for design of all applications across different experiments.
- Each application owner goes through a tutorial for Oracle Designer.
- Data Model evolves per reviews.
- Once Final Design is done. Space report is generated for the application.
- DDLs are generated through Design Editor tool of Oracle Designer. DDLs are reviewed with Data Base group as per standards defined. Application owner CVS the ddls.

Standards for Data Modeling

Fermilab

- Review DDL statements for:
  - Tables: Correct tablespace placement i.e. indexes in index tablespaces, data in data tablespaces
  - Tables: Ensure that no table has 0 start or end rows, otherwise it's not counted in space report.
  - Tables: Every table must have a corresponding Primary Key in the constraints file.
  - Tables: Every table(and view,and sequence) must have a synonym)
  - Tables: Check every object for storage definitions; match DDL with storage report, initial, next must be defined, it is easy to miss one

Standards for Data Modeling

Fermilab

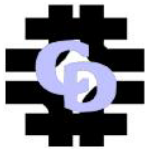
- Review DDL statements for:
  - Tables: Check for pct increase of 0%
  - Tables:Check that all NOT NULL fields are on top
  - Tables:Check date fields are defined as date vs varchar2 or other, unless required by the application
  - Constraints: FK's do not require storage or tablespace definitions, but PK's do. Make sure every ER table with crow-foot coming into it has a FK in the .con file.
  - Indexes: Every FK constraint **must** have a corresponding FK index. Also compare initial,next,min,max on space report to each FK in the .ind file.

Standards for Data Modeling

Fermilab

- Review DDL statements for:
  - Triggers: Ensure that there are not multiple "Before Insert" triggers on the same table. They need to be rolled into one trigger.
  - Sequences: Every sequence needs a synonym
  - Synonyms: Due to bug in Designer, make sure table owner is removed from synonym
  - Synonyms: must be given to public
  - Grants: Every table and sequence must have at least 'SELECT' granted to it

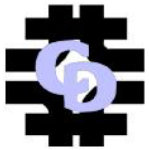
Standards for Data Modeling



Fermilab

- Review Space Report:
  - Check to make sure Database Block Size matches the db_block_size. Default value is 2K, but our databases generally are created with db_block_size of 8K.(8192)
- Make sure the CLOBS and BLOBS go into their own tablespaces. Example
  - Make sure to update the Database Application section off the Support DB Project References page

Deployment

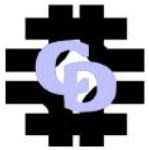


Fermilab

- A development environment exists for each application. This environment will house the application's tablespaces and the owner of the application.
- The development environment contains logons for non dba individuals. These logons are used for development and testing at the lowest levels, before applications are user-ready.
- An integration environment exists and used to test the cutting scripts and application code before declaring production.
- Once schema is deployed on integration and APIs are tested are deployed on production.
- The integration and production databases will contain logons for the application logons and dbas only. Individual users without application-specific roles will not have access.
- Application Deployment on dev/int/prd Philosophy Details :

http://www-css.fnal.gov/dsg/external/oracle_admin/run2_standards.html

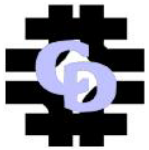
Tuning



Fermilab

- Server Tuning
- SQL Tuning
- Partitioning

Server Tuning



Fermilab

- Server Statistics are monitored on a weekly basis.
- Servers are tuned on the basis of recommendations from monitoring reports.
- Example :

After upgrade of D0 databases to 9.2.0.5

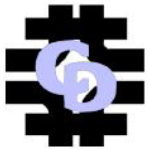
performance degraded. Action taken to tune :

- Tuned the server by setting the automatic memory management.
- Analyzed the whole database using DBMS_STATS pack. Database was much performant.

  Average Response time per execute on d0 is 0.01 secs to 0.07 secs. Monitoring graphs are posted on web on a monthly basis :

https://fncduh12.fnal.gov/cp/

SQL Tuning



Fermilab

- Bad Performant queries are tracked . And tuned as needed.
- Find the explian plan of query. Also trace of query can be submitted to “tkprof” utility to find details of query as per CPU and parse counts goes.

. Cartesian Joins are performance Killer. In fact result in bad query

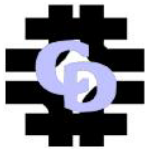
- Ad hoc queries are first run on integration db before submiiting to production.

Example:

The following query cause cartesian join since data_files table is not joined with other tables.

```
select pt.param_type ParamType, pv.param_value ParamValue,
pv.param_value_id, ParamValueID,pc.param_category
ParamCategory
from   data_files df, param_values pv, param_types pt,
param_categories pc  where   (pc.param_category like 'global%'
and pv.param_type_id=pt.param_type_id
and pt.param_category_id=pc.param_category_id)
order by pt.param_type;
```

SQL Tuning Contd



- Query with inner joins is going be faster as compared to SUB SELECT queries. Since inner join doesn't have overhead of sort operation.

Example :

```
select df.file_id from project_files pf, data_files df where
pf.proj_snap_id=136780 and pf.file_id=df.file_id and
df.file_content_status_id=1;
```

is faster than

```
select file_id from data_files where file_id in (select
file_id from project_files where proj_snap_id=136780)
and file_content_status_id=1;
```

SQL Tuning Contd

Fermilab

- Execution Plan for First Query :

```
----------------------------------------
0 SELECT STATEMENT Optimizer=CHOOSE
(Cost=305 Card=150 Bytes=3
000)

1 0 NESTED LOOPS (Cost=305 Card=150
Bytes=3000)
2 1 TABLE ACCESS (BY INDEX ROWID) OF
'PROJECT_FILES' (Cost=5
Card=150 Bytes=1650)

3 2 INDEX (RANGE SCAN) OF 'PF1_PK'
(UNIQUE) (Cost=3 Card=1
50)

4 1 TABLE ACCESS (BY INDEX ROWID) OF
'DATA_FILES' (Cost=2 Ca
rd=1 Bytes=9)

5 4 INDEX (UNIQUE SCAN) OF 'FI_PK' (UNIQUE)
(Cost=1 Card=2
)
```

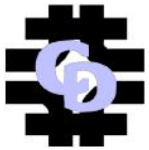
- Execution Plan for Second Query

```
----------------------------------------------
```

-

```
0 SELECT STATEMENT Optimizer=CHOOSE
(Cost=307 Card=150 Bytes=3
000)

1 0 NESTED LOOPS (Cost=307 Card=150
Bytes=3000)
2 1 SORT (UNIQUE)
3 2 TABLE ACCESS (BY INDEX ROWID) OF
'PROJECT_FILES' (Cost
=5 Card=150 Bytes=1650)

4 3 INDEX (RANGE SCAN) OF 'PF1_PK'
(UNIQUE) (Cost=3 Card
=150)

5 1 TABLE ACCESS (BY INDEX ROWID) OF
'DATA_FILES' (Cost=2 Ca
rd=1 Bytes=9)

6 5 INDEX (UNIQUE SCAN) OF 'FI_PK' (UNIQUE)
(Cost=1 Card=2
)
```

Partitioning



Fermilab

- Partitioning has been implemented for very large table(s) in the database. e.g. The D0 Event table, which records trigger information
- The Event table is count partitioned with each partition containing 50M events. Each partition's data and index are stored in its own tablespaces.
- Benefits of partitioning are twofold,
  - It improves query optimization. The optimizer, knowing that the table has been partitioned, directs the server to only access data from that partition.
  - Increases backup performance. Backup performance is achieved because once a partition is "rolled over", its data and index tablespaces are converted to READ ONLY. READ ONLY tablespaces are only backed up twice a month, and are excluded from READ WRITE tablespaces backup, thus reducing the time for daily database and tape backups. Currently, over 1 billion events are distributed over 25 partitions in the database, and a new partition is added roughly once every three weeks. The growth of partitions in the D0 Event table is shown in Figure 1.

Partitioning – Contd

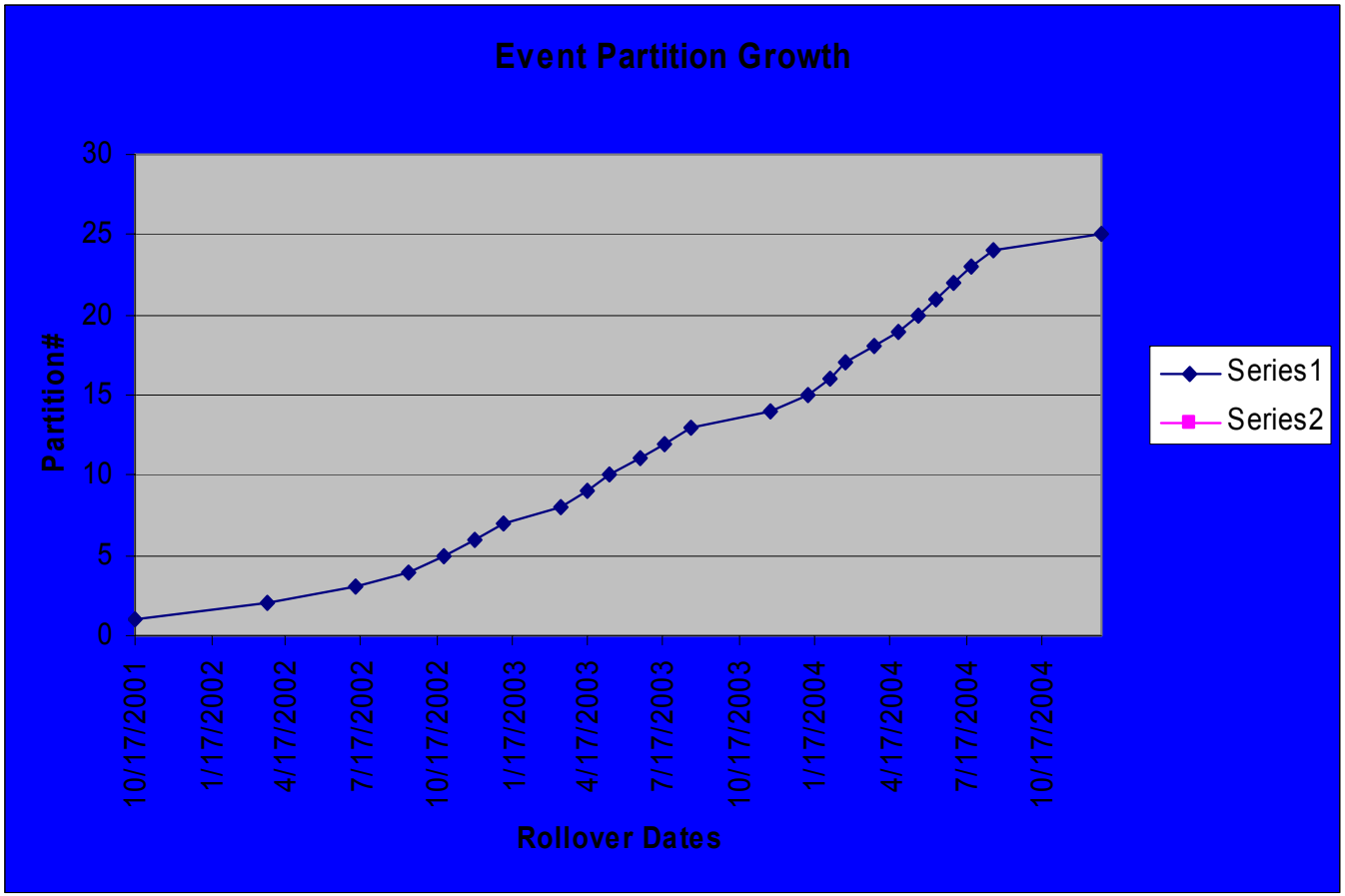


Figure 1

Load Balance Via Replication

Fermilab

- CDF employed oracle replication to copy the online instances of the database to the offline, is shown in Figure 2.
- CDF is also pursuing a web caching technique through the Frontier project that will greatly improve the performance of their read-only database access, especially to clients located geographically distant to Fermi Lab.

CDF Db Current Scenario

Fermilab

CDF Basic Replication

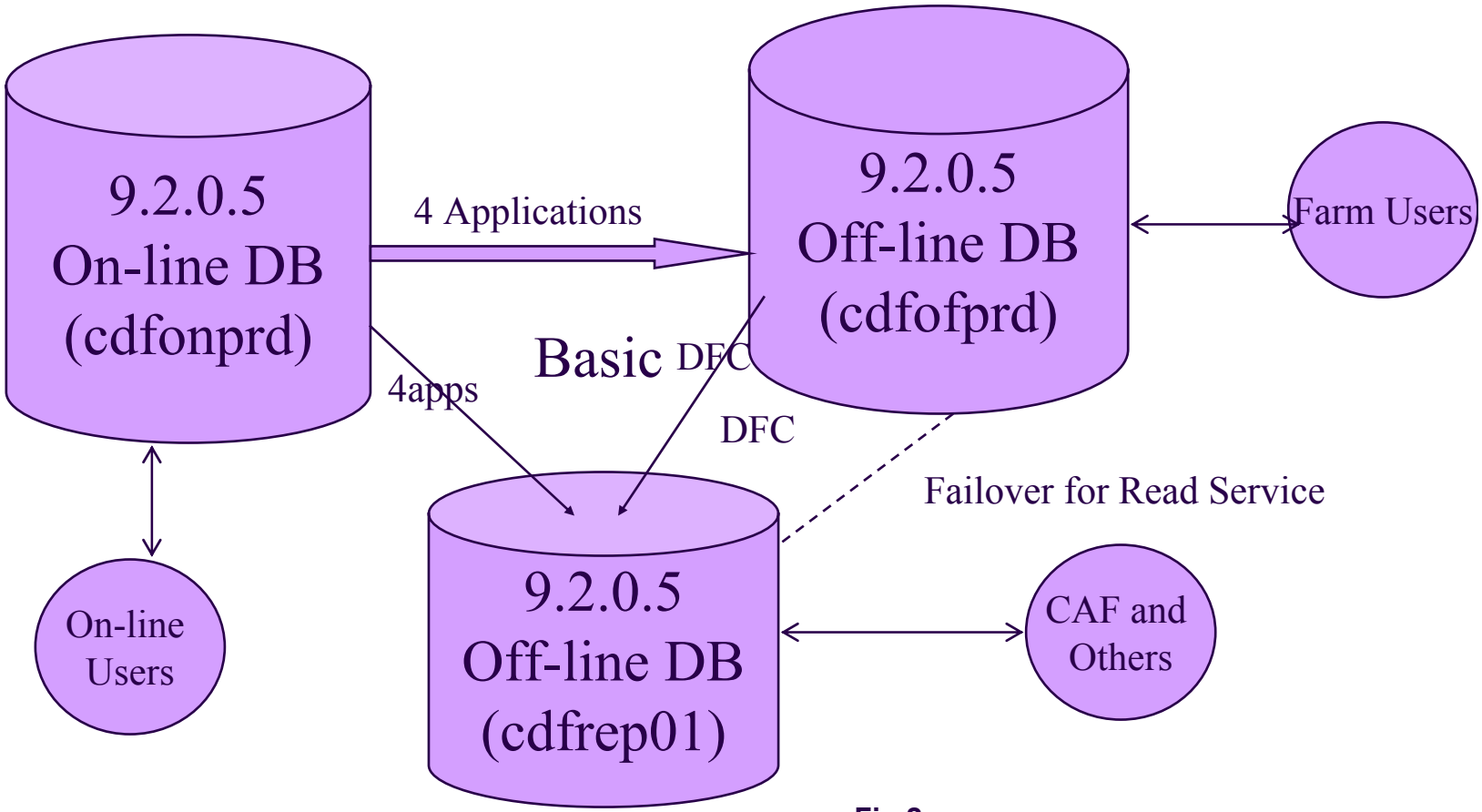
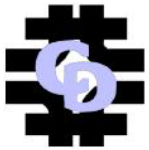


Fig 2

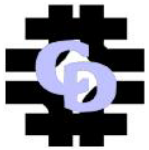
Monitoring



Fermilab

- **Oracle Enterprise Manager (OEM)**
- OEM is tool by Oracle Corp. to monitor the oracle databases.
- Monitor the a) Node up and down. b) Database Listener down
- c) Intelligent Agent
- d) Number of storage extents.
- e) Space
- f) Database Alerts – Db down , file corruption
- g) Monitor number of concurrent sessions
- h) Monitor the space usage
- i) Monitor the CPU usage
- k) Monitor the Memory Usage
- l) Monitor the hit ratios for Library , Buffer Cache.
- Plots the monitoring graphs.
- Graphs can be posted on the web.

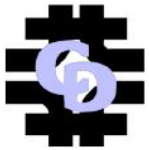
Monitoring



TOOLMAN/DBATOOLS/TOOLMAN DB

- The toolman utility is designed to provide an alternative method for monitoring Oracle databases that does not share a common point of failure with the Oracle
- Enterprise Manager. Unlike OEM, it has no GUI and is implemented as shell and SQL scripts. It is hoped that this will provide a higher level of database
- monitoring, perhaps at the expensive of redundant alerting.
- Toolman also provides ongoing routine maintenance for a database by executing a series of cron jobs that automatically perform basic Oracle maintenance activities.
- Further, toolman provides a source of historical data and ongoing monitoring should a machine be isolated from the network and otherwise unavailable to OEM.
- Customization of Toolman
- Toolman can be customized in several ways for the machine and databases it monitors.
- The toolman product contains two configuration files those describe the characteristics of the machine and databases to be monitored. All of the toolman scripts look up this information using the toolman_config command to determine how to operate. By
- tailoring these two files, one can tailor the environment in which toolman operates.
- The format of the configurations files is a colon-separated list of tailoring characteristics. The first line of the file defines the contents of each field. Temporary modifications can be edited directly on the machine; permanent modifications need to be included in the toolman product.

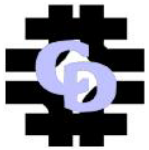
Special Features



Fermilab

- Sniping
- Login Traces
- Capacity Planning

Sniping



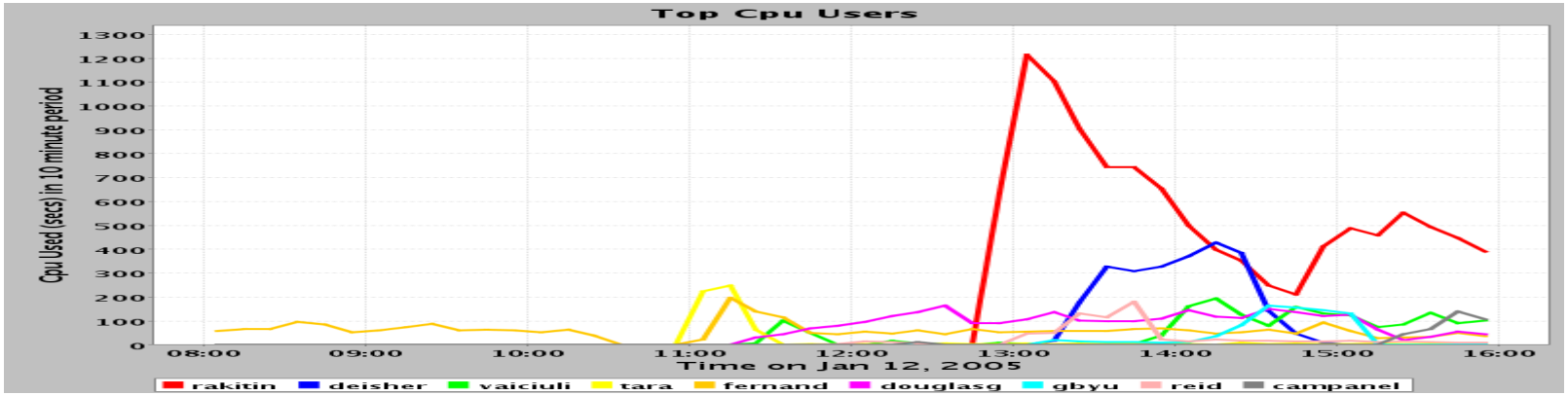
Fermilab

- Specific inactive sessions since a specific time are terminated and e-mail notification with complete session info is sent to user that session is terminated.
- This feature helped in lowering the number of sessions to db. Some times db used to become unusable due to limited number of processes.
- There are rules for each database which user's session and which program need to be sniped.
- Also this helped in tune the code so that API will connect to db if data from database is needed, otherwise disconnects.

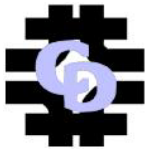
Login Trace

Fermilab

- On the database level each logon and logout is tracked with session info including CPU used by the session.
- This information is used to track top CPU users of the database and number of connections over a given period of time.



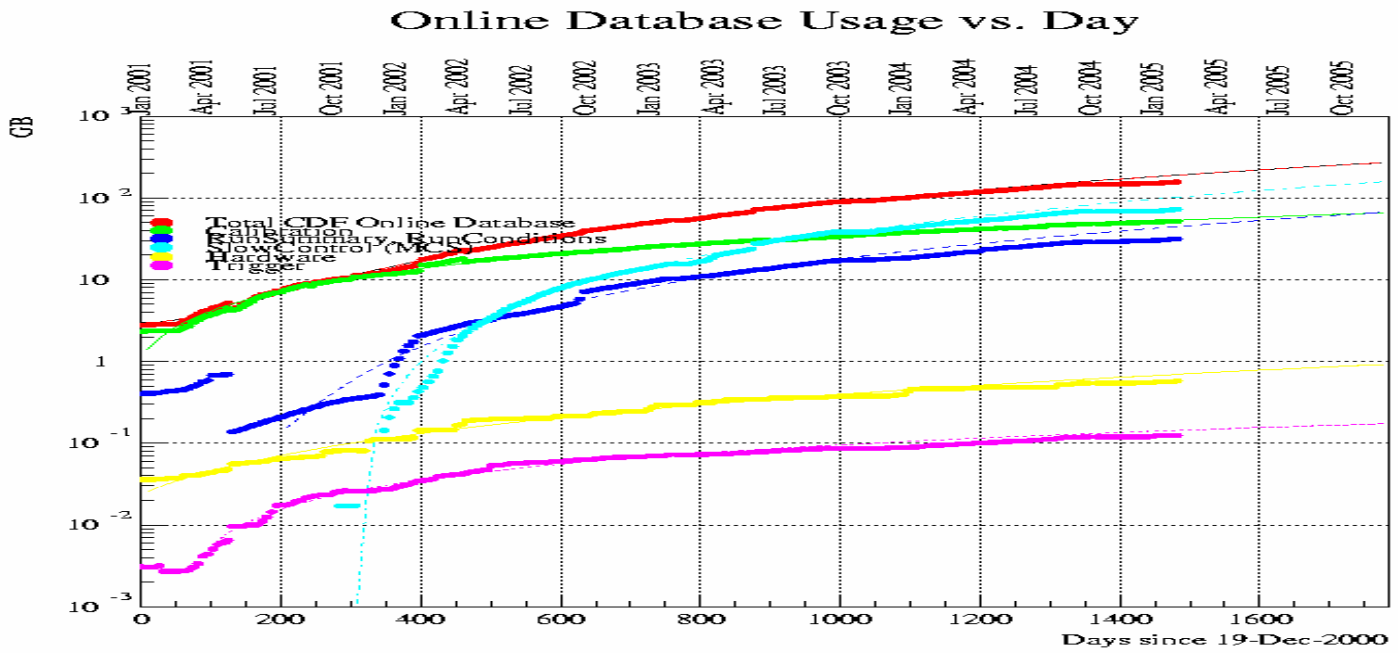
Capacity Planning



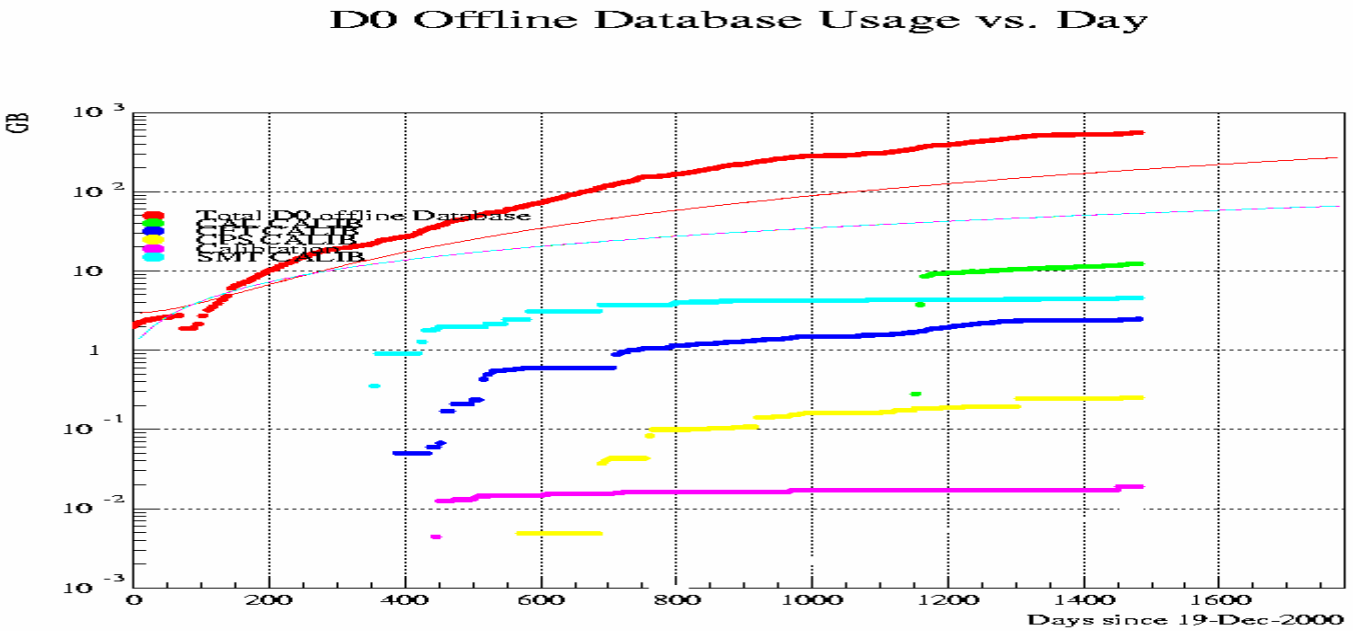
Fermilab

- On weekly basis, a cron job tracks the space usage by each object in the database.
- This data is plotted to find the growth of each application over a period of time.
- This data is analyzed to project the future growth of the databases.

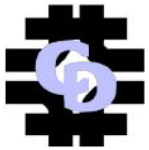
CDF Db Capacity Planning



D0 Db Capacity Planning



Summary



Fermilab

- System Admin should work together with DBAs to define the disk layout as per OFA.
- Application Developers should work together with DBAs to define integral Data Model for the application.
- Tuned Code is important Key for tuned Servers.
- DBAs and Application owner should play a proactive role in capacity planning.
- DBAs should be proactive in monitoring and tuning the Server.