Case studies – CMS Dashboard

Jacek Wojcieszuk, IT-DM

Database Developers' Workshop

July 8th, 2008

Dashboard application

- A home-developed application for monitoring LHC experiments’ activity in the GRID
- Separate flavors for each experiment
- 3-tiers: client, application server, database
- Mixed workload:
  - Some OLTP
  - And some batch processing
- Many performance issues with CMS Dashboard
  - Very high load generated by the application
  - Very poor response times of some queries executed from the Dashboard’s web interface
  - Occassional deadlocks and processing glitches

CMS Dashboard and high load

- Findings:
  - Multiple jobs scheduled with the DBMS_JOB package
    - Some of them re-processing the same piece of data many times
    - Some of them failing from time to time in the middle of processing for various reasons
  - Hundreds of thousands of database sessions per week
    - Some of them very short
  - Lots of indices – severly affecting DML performance
  - Bad data organization:
    - All the data in one big table while only the most recent data is often accessed

CMS Dashboard and high load

- Solutions:
  - Review of scheduled jobs implementation, use DBMS_SCHEDULER instead of DBMS_JOB:
    - DBMS_SCHEDULER gives more control on when and where the jobs are running
    - DBMS_SCHEDULER provides access to some logging information, that facilitates debugging
  - Use of a database connection pool – especially easy in case of multi-tier applications
    - check if the connection pool is sized/configured properly
  - Review of the schema and indexing strategies
    - Remove overlaping indices
    - Denormalize the schema where it is justified
  - Use of range partitioning to separate recent and old data.

Very poor response times of some queries

```
select * from (
SELECT JOB."SchedulerJobId", SITE."DisplayName",
   GRID_STATUS_REASON."GridStatusReason", GRID_STATUS."StatusName",
   JOB."JobMonitorId", JOB."DboardStatusId" , JOB."DboardJobEndId",
   JOB."DboardGridEndId", JOB."DboardStatusEnterTimeStamp",
   JOB."DboardFirstInfoTimeStamp", JOB."DboardLatestInfoTimeStamp",
   JOB."SubmittedTimeStamp", JOB."StartedRunningTimeStamp",
   JOB."FinishedTimeStamp", NODE."IpValue", TASK."TargetCE",
   TASK."TaskMonitorId", TASK_JOB."NoEventsPerRun" ,
   TASK_JOB."EventRange", JOB."JobExecExitCode",
   TASK."SubmissionType", rownum as rnum FROM JOB, SITE, TASK,
   TASK_JOB, GRID_STATUS, SUBMISSION_TOOL, USERS,
   GRID_STATUS_REASON, INPUT_COLLECTION, SUBMISSION_TYPE, NODE where
   (rownum < (50+50)) and (("DboardLatestInfoTimeStamp" >= :bv_date1
   OR "DboardStatusId" in ('P','R'))) and (("DboardGridEndId"='U'
   and "DboardStatusId"='T')) and ((TASK."TaskTypeId" in (select
   "TaskTypeId" from task_type where "Type" = :bv_activity))) and
   ((JOB."SiteId" in (select "SiteId" from site where "DisplayName"
   = :bv_site) or JOB."SiteId" in (select "SiteId" from site where
   "SiteName" = :bv_site))) and ( JOB."TaskId" = TASK."TaskId" and
   JOB."TaskJobId" = TASK_JOB."TaskJobId"
and JOB."SiteId" = SITE."SiteId" and JOB."GridStatusId" =
   GRID_STATUS."StatusId" and TASK."SubmissionToolId" =
   SUBMISSION_TOOL."SubmissionToolId" and TASK."UserId" =
   USERS."UserId"
and TASK."InputCollectionId"=INPUT_COLLECTION."InputCollectionId" and
   GRID_STATUS_REASON."GridStatusReasonId" =
   JOB."GridStatusReasonId" and NODE."NodeId" = JOB."WNIp" and
   TASK."SubmissionType"=SUBMISSION_TYPE."SubmissionType"
)) where (rnum >= 50)
```

Very poor response times of some queries

- Findings:
  - – All afected queries of monstrual size, practically unmanageable
  - – Often more than 10 tables joined
- Solution:
  - – Schema review and reorganization to have most often executed queries as simple as possible
    - Adjust data model to predominant application use cases
    - Denormalization when benefits surpass related danger
    - Materialized views when relevant
  - – Stop the query execution if nobody is waiting for the results

Deadlocks and processing glitches

- Findings:
  - Sessions sometimes failing due to deadlocks
  - Processing stops sometimes due to locking issues
    - All spotted cases related to DML operations executed manually and not commited or rolled back
- Solution:
  - Review application’s workflow to avoid deadlocks
  - Protect production data from being modified manually
    - Access to the data through reader/writer accounts only
    - Access to the data owner schema only on request