Database Performance

Eric Grancher - Nilo Segura

Oracle Support Team

IT/DES

Agenda

- Service
- Server
- Client
- Schema Design
- To be done
- Conclusion

Service

- Not the right hardware for testing/debugging
  - Several database corruptions caused by hardware problems
  - Single CPU machines - system was so loaded that we could not connect to it during the heavy load
- Not enough time for proper testing : firefighting mode!!
  - SC not the right time to stop/start databases to explore new settings/chances
- Critical service => requires a full time DBA : 13 databases right now to control!! Not counting dev/test databases
  - DB Consulting
  - Performance and Tuning
  - Monitoring
  - Backup
- Resources allocated in the resource plan for production

Server

- Running now latest Oracle RDBMS 32bits release available 10.2.0.2
  - Upgrades have not harmed either the performance (10gR1, 10gR2) or stability.
    - One problem (10.2.0.1) caused one index to be corrupted (under certain circumstances) - patch installed with current 10.2.0.2
- Conservative approach
  - Some cutting-edge technology not enabled
    - Stability was/is our goal. We do the experiments somewhere else..
- Generic tuning of the DB parameters based on experience
  - Buffer cache, Shared pool area, PGA area.
  - Internal advisories enabled
- Asynchronous I/O enabled
  - Direct I/O does not work together with Asynch I/O on current Red Hat 3.0
    - This is a desired feature - Available by default in our DB/Solaris for a number of years
- Segment space management automatic
- Databases are backed using Oracle Recovery Manager tool

Server

- Heavy use of Automatic Workload Repository to generate server side performance reports
  - Data collected regularly by the Oracle kernel
- Some problems with the optimizer for one particular SQL stmt
  - Use of optimizer hints alleviated the problem
  - But not optimal - single most expensive query in the current Castor release!!
- Oracle kernel collects automatically statistics for the optimizer to generate the execution plans for the SQL stmt
  - Blocked the automatics data gathering once we obtained stable execution plans
  - Use OUTLINES to save the execution plans - to be explored

Client

- DB centric application
  - Stager code calls directly SQL and PL/SQL procedures
  - Oracle C++ Call Interface API
    - OO interface, bulk operations, bind variables
- Use of PL/SQL stored procedures was indeed a very good idea
  - Much easier to modify the SQL code inside the DB than touching the C++ code
    - Modify the code and see immediately their positive or negative effects
  - Better performance and DB integration for complex data manipulation logic
- Help with best DB client coding practices fixed some initial problems
  - Lack of bind variables in some places
  - Excessive COMMIT activity
    - Discovered with the first performance report
  - Bulk operations added in some places
    - Used heavily in the PL/SQL code
    - More relevant for the DLF

Schema design

- Only the Stager part
  - DLF not touched yet, just generic recommendations
    - Partitioning, Bulk operations…
- No real involvement during database schema design
  - Very general validation
- Recently some re-engineering effort using Oracle Designer to obtain E/R diagrams of the schema
  - Found some missing NOT NULL and FK constraints
    - Some already fixed
    - Also found that adding a certain FK constraint break some Castor initialization steps :(
  - Could add some Check constraints in some places
- Effort aiming at letting the database to know more about the semantics of the application

Optimization

- First rule: for databases, you can not increase performance beyond a certain level by “being generic”
  - Use the minimum set of common features just to be compatible with other RDBMSs is a mistake.
  - There is never “enough performance” for the user :)
- Second rule: do the same with less database resources
  - If I can get the same results with less I/O… let’s do it!!

Optimization

- Major effort to optimize the application since August 2005
  - Identifying and Analyzing hot SQL stmts
    - Numer of executions + Logical I/O
  - Adding missing Indices
    - Normal B-trees + Function Based
  - Analyzing access patterns to hot tables triggered a change in the physical implementation of the tables
    - From normal Heap to Partitioned Index Organized Tables
  - Rewrote parts of the PL/SQL code to reduce its complexity
    - Reducing the number of SQL stmt to be executed
    - No real PL/SQL code profiling yet
- All changes validated by the database performance reports

Optimization

- So far we have optimized all the problematic SQL code but one big SELECT
  - A problem in the optimizer?
  - A problem with the design of the database that forces that SQL stmt to be “that complex”?
  - It will certainly require a re-write/wording.. But we will prevail!! :)
- There is a serialization (“thelock” table) point in the code
  - Introduced to achieve certain results that otherwise looked pretty difficult and expensive to obtain
  - After some optimizations, it does not cause any problem
    - It does not appear in the database performance report :)
  - Checking alternative ways so that it could be eventually removed
    - It does not look “nice” from a strict database point of view and it is an intellectual challenge :)

To be done (I)

- Implement missing constraints
  - NOT NULL, Check and FOREIGN KEY
  - Castor Team aware of these issues and working on a new version :)
    - Some are already fixed
  - The fix will increase the consistency of the data
  - It may change query plans due to missing indices required by the Foreign Keys
- Further profiling of the PL/SQL code may lead to a code rewrite to increase its efficiency/scalability

To be done (II)

- Removal of serialization point
  - It will lead to PL/SQL code changes
- Further optimization of the physical structure of the tables when required
  - IOT, Partitions, Table Clustering
- No changes in the Castor client code..
  - The stored procedures api do not change
  - The table columns are still the same
- Work on DLF and in the SRM part (when available)

Conclusion

- Excellent collaboration with the Castor Team
  - Many thanks to Sebastien, Giuseppe and Olof for their kind help
- Major effort of DB optimization complete
  - At least 80 Castor Stager requests processed per second
- DB is not the bottleneck of the system
  - Confident that the DB can easily sustain the incoming workloads
    - Provided that we do not run into HW limits.. :)
- We can still increase the performance of the DB working on the application side.
  - Last improvement took place last Saturday on the Castor2 SC4 Stager DB - added an index..
- It would be benefitial to perform more regular stress tests on the whole system