New Oracle 10g Features
by Ken Jacobs, Oracle  “The release of 10g has historical implications”

- Performance enhancement – processing speed
- Designed to work with Grid
- Regular expressions in SQL queries
- native floating point numbers
  - New data types – BINARY_FLOAT, BINARY_DOUBLE, and conversions with Oracle NUMBER data bype. Better computation performance; space savings
- enhanced CONNECT BY – hierarchical tree queries
- enhanced support for Collections (VARRAYS, nested tables..)
- Faster PL/SQL and packages (UTL_MAIL, UTL_COMPRESS)
- Statistical functions in SQL - BLAST algorithm (DNA pattern matching)
- UNDROP TABLE
- Faster, bigger LOBs (to terabytes)
- TABLESPACE
- Query optimizer (industry’s best). Use Cost Base Optimizer (COB), not Rule Based Optimizer (ROB) for SQL Tuning (ROB being deprecated)
- OLAP new analytic features
- Data Mining – new gui
- SQL MODEL clause – permits spreadsheet-like computations in a single SQL statement, individually addressable cells
- HTML DB – built-in RAD environment which includes Application Builder, the SQL Workshop, and the Data Workshop for building db-centric web applications. Can build web-based applications that meet departmental information needs.
- Auto diagnostic monitor – auto SQL tuning
- Self optimizing SQL – finds slow SQL statements
- Flashback Query – see data back in time (primarily for recovery purposes) – see all versions of a row between 2 times and the transactions affecting them
- Automatic dynamic query optimization
- “Data Pump”, an enhanced import/export facility, lots faster.
- RAC architecture (Real Application Clusters – free feature with 10g)
- Automatic “striping” for raids
- Incremental backups 10x faster
- Enhanced CONNECT BY
- ANSI join, full outer JOIN, CASE statement (9i)

NOTE : 8i support ends 12/04; Rule Based optimizer support ends after version 9i
Forms to J2EE, Not Java: Architectural Challenges and Strategies for Migration of Oracle Forms to J2EE, by Sri Rajan, Churchill Software

Existing Oracle forms on 2-tier client/server architecture when migrated to J2EE present challenges for developers. Client side business logic must move to server side (where it belongs).

Jautomator maps various parts of the monolithic Oracle Forms application to these J2EE business logic and presentation tiers, and splits client-server code across the Model (business logic), View (presentation UI), and Controller (event handling) layers. Value added over JDeveloper – forms are integrated into their product.

Forms authors may not possess the skills necessary to migrate to J2EE:
- OO Analysis / Design
- UML
- Design Patterns
- Frameworks (IDEs, Oracle BC4J, ADF ..)

Note: ADF is the Application Developer Framework provided in JDeveloper, previously called BC4J. This is a framework providing Data Binding and Data Access classes, which fit in as part of the Control Layer. It uses Struts architecture and especially supports JSP in a Browser. This set of classes is available as a jar file for any IDE, but JDeveloper has excellent built-in support for developing with these classes.

4 Categories of Enterprise Applications:
- Business to Business (inventory mgmt, supply chain …)
- Business to Consumers (amazon.com)
- Self Service (status, …)
- ERP (Enterprise Resource Planning: business, order entry, accounts payable …)

View technologies:
- Web client - java client/applets with JFC (Swing) best for running rich clients on all client platforms
- JSP
- JavaServer Faces
- Flash (internal caching – good for broken connections)
- SWT (IBM’s emerging tech. more like Windows UI …)

The notion of "faceless" business services becomes a very interesting area for designing a stack of common "service-oriented" business logic completely independent of the user interface. They are looking into ways to expose this level of capability in the coming months.
Building a Poor Performing J2EE Application is Not Your Developer’s Fault
By Dave Martin, Wiley Technology

Slides available from Wiley for our internal use only. Please contact drogind@slac.stanford.edu for the complete set.

Topics of presentation include:
True stories of performance gone bad
  • Does Java have resource leaks?
  • What are the ten or twelve most common production problems?
  • How might you identify them?

Planning for performance
  • Do performance and resource monitoring from the very beginning of the project.

Possible Symptoms
  — Consistent slowness
  — Slower and slower against some variable
    • Time
    • Load
  — Sporadic hangs / random errors
  — Foreseeable lockups
  — “Sudden chaos”
  — High utilization of resource R (CPU, memory, network, etc.)

Memory Leaks : Linear & Exponential
  — Causes:
    • Objects added to linear structures without being removed (e.g., linked lists)
    • Other API misuse (addListener() without corresponding removeListener(), etc.)
    • Objects added to most data structures without being removed (e.g., vectors, hashtables)
  — Aggregate detection:
    • Linear / exponential growth in heap
  — Specific detection:
    Figure out object types being leaked
    Find related APIs and search code for misuse

Resource Leak
  • Causes:
    — API misuse of Java objects with resource-style lifecycle (create->use->destroy)
  • Aggregate detection:
    — Specific symptoms
  • Specific detection:
    — Audit code for API misuses – use try catch finally clauses

Bad Coding: Infinite Loop
  • Causes:
    — Infinite loop in code
  • Aggregate detection:
    — Stalled threads / permanently high concurrent usage
  • Specific detection:
    — Thread dumps as needed
Bad Coding: CPU-Bound Component

- Causes:
  - Idiot with a “Learn Java in 24 Hours” book
- Aggregate Detection:
  - method timers
  - aggregate CPU utilization
- Specific Detection:
  - detailed CPU utilization
- Typical Cure:
  - Cache of data or of performed calculations

Layer-it is (slides 14-16)

- Causes:
  - Poorly implemented data bridge layer, or simply too many of them
  - DB -> XML -> XSLT -> More XML -> “Custom Data Management Layer” -> Consumer
- Aggregate Detection:
  - method timers
- Specific Detection:
  - Ask for a design or architecture document
  - Add more method tracers

The Unending Retry (20)

- Causes:
  - Continual attempts to call backend + unavailable backend
- Aggregate Detection / Specific Detection:
  - Same as Infinite Loop
- Note:
  - Differentiating successful calls from failed ones helps with availability management of the backend system

Threading: Deadlock / Livelock

- Causes:
  - Fundamental error in threading / lock acquisition strategy
- Aggregate Detection:
  - Stalled threads / permanently high concurrent usage – pay atten to dependencies
- Specific Detection:
  - Thread dump

Threading: Chokepoint

- Causes:
  - Many threads bottlenecked waiting for one lock
- Aggregate Detection:
  - Stalled threads / high concurrent usage
- Specific Detection:
  - Thread dump

Internal Resource Bottleneck

- Causes:
  - Overusage of internal resource (threads, database connections, etc.)
- Aggregate Detection:
  - Stalled threads / high concurrent usage
- Call rate and average response time of internal resource
- Specific Detection:
  - Also compare with methods from Resource Leak, External Bottleneck, and Overusage of External System
External Bottleneck

— Causes:
  • External system (database, authentication server) is slow
  • Compare with Overusage of External System
— Aggregate Detection:
  • Method timers (particularly on bridge layers)
— Specific Detection:
  • Generally need deeper visibility into details of JVM’s interaction with backend system (e.g., SQL Agent)
  • Also need tools native to said backend system

The Goldilocks Problem (26)

— Interesting problem occurs when the size of transactions with backend systems needs to be tuned
— Can be intertwined with / exacerbated by Layer-itis and Overusage of External System

What is Performance Measurement and Analysis?

Different definitions for different parts of your project...
— Development
  • Profiling
  • Logging
  • Unit Testing
— QA/Staging
  • Functional Testing
  • Performance/Load Testing
  • Defect/Performance Tracking
— Production
  • Audit Trails
  • Availability
  • Service Level Agreements (SLAs)
  • Sitemining

So how do you analyze? (31)

Think in terms of user transactions or use cases for your application
In our example, a single transaction might be...
— Logging in
— Logging out
— Looking Up Account Balance
— Editing Account Information
— Transferring Funds
Stay focused on what your customers do with your application... why tune something unrelated?

“Repeatable” and “automated” are your project’s two watchwords

• The key to a succesful J2EE Application deployment is constant vigilance.
• Build repeatable and automated scripts to test and generate data for weekly, daily builds.

Approach #1: Profiling

• How does it work? (JVMPI)– Java Virtual Machine Profiler Interface
• What can it get you?
  — CPU usage
    • Clock time
    • CPU time
  — Memory usage
    • Navigating a Java Heap Reference Graph
    • Watching the Garbage Collector (temporaries, etc..)
• Pros:
  — Lots of Information
— Detailed Memory Data

- Cons:
  - Lots of Information
  - Lots of overhead (not usually possible under load)

Approach #2: Logging

- How does it work?
  - You log (Log4J, JDK 1.4 Logging, ..)
  - They log (EPM, PMI, JMX, ..)
- How can you use it?
  - In a file...
  - Into a monitor...
    - Resource Analyzer, Introscope, etc..
- Pros (of you log):
  - Produce any statistic
  - Send stat anywhere
- Cons (of you log):
  - You write it yourself
  - Can not log into components you did not write (e.g. WebSphere)
  - Hard or impossible to change once application is deployed

Approach #3: Bytecode Instrumentation

- How does it work?
  - In-memory or Out-of-memory post-processing of bytecode (e.g JAR or ZIP files)
    - "Logging without source code changes"
- How can you use it?
  - Like you do logging or profiling, but...
    - Without the same level of detail about memory utilization
  - For example:
    - average response times,
    - call count,
    - call rate,
    - stalled thread counts,
- Pros:
  - Add or remove on demand
  - Low overhead
  - Watch anything (even components you don’t own, like WebSphere)
- Cons:
  - Limited options on what kind of tracing can be done
  - Doing it for free would be hard

Making sense of your data...

- Wall Clock Time vs. CPU Time
  - CPU Time interesting mostly during development: how much computational resources must be spent to complete a given task?
  - (Response) Clock Time interesting in Staging and Production: how long does someone actually have to wait to get an answer?
    - For example, stalled or starving backend system
- Memory Consumption
  - GC Heap vs. OS-level
  - Leaks
    - Common Causes of Leaks
      - Object lifecycle
      - Per-user Session Caches
      - Caches in general
- Your applications resources... file descriptors, VSAM records, etc..
Putting it all together…

What does a good use of these technologies look like in a real project?

— Development:
  • Developers have automated functional and performance testing harness. Any problems arrive in their mailbox the following morning.
  • Testing harness built on the core use cases of the application – as soon as one is functional for the first time, it should get added.
  • Developers use Profilers to diagnose and isolate any issues that are detected by the automated framework.
  • If audit is required, a full logging strategy should be implemented at Alpha testing time (but probably not before) – logging to debug can be done anytime.

— QA:
  • Mostly, their job is to build and maintain the above-mentioned testing harness!
  • Any edge case style problems should be added to the harness on an ongoing basis

And in Production?

• Who is using what portions of your application?
• How long is it taking them?
• How much load do you need to support now? Next week? Next month?
  — Dealing with seasonal or singular load “events”
  — Planning for a successful deployment
Oracle 10g Manageability
By Michael Sit, Oracle

Oracle 10g – A Revolution in Database Technology, Geared for the Grid!

Oracle 10g
- designed to make use of Grid computing, a new movement in industry
- coordinated use of servers and storage acting as ONE
- infrastructure is used to its full advantage.
  1. Problem: Islands of storage –“My storage is 50% utilized and growing 30% a year
  2. Solution: Oracle 10g automatic storage management groups standard, modular disks - consolidated, networked storage, dynamically provision
  3. Problem: Islands of computation –“ My servers are 15% utilized”
  4. Solution: Oracle 10g integrated clusterware groups industry standard servers into one

Features:
- Highly scalable, available and reliable
  1. Push-button add/drop server to cluster
  2. Automatic routing of service requests to appropriate server with lowest load
  3. On server failure, automatic re-allocation of surviving servers to services

- Low cost in management
  1. 50% DBA cost (comparing to 9i)
  2. automatic maintenance tasks
    - Easy installation (one CD, < 1 hour)
    - Automatic Database Diagnostic Monitoring
    - Automatically provides database-wide performance diagnosis
    - Servers generated alerts (proactively)
    - Pinpoints root cause
    - Flash Backup and recovery
    - Easy – single command: SQL> Flashback database to ‘2:05PM’
    - Like a “Rewind” button for the Database
    - Self optimizing SQL management (automatic SQL tuning, analysis and etc)
Automated Path Management (Real-time discovery of new patches and applying)

Reality of today (a survey)
- 10% Oracle 9i, 90% Oracle 8i and Oracle 8.
- Oracle 10g – targeted for a new trend of enterprise grid computing
- “Keeping your feet on the ground and reaching for the Stars”
Vendors

**Embarcadero Technologies**
has a product called ER studio. You can pull a database structure into a nice gui picture plus all kinds of data on each object. A web page can be generated of the picture, with live links into object data. (!!!!!) We have a couple of fully functional demo disks, each good for 2 weeks.

Also on the demo disks are gui tools for database design, and for SQL development. Plus some other things, yet to be explored.

**Leccotech**
has a product called SQL Expert, which helps you build optimized SQL queries. On the demo disk is also something called DB Expert – not sure what this is yet.

**Churchill**
has a tool for converting Oracle Forms applications into J2EE applications (not that we have any Oracle Forms, but it looked pretty cool!)

**Princeton:** database performance, “active archive” to retrieve Oracle archived data.

**Info Cyclone:** database performance, retrieval

**Appworks:** job scheduling, app automation (?)

**Quest:** TOAD and other application dev and db management tools

**DB Appliances:** backup and recovery

**Solix:** Enterprise archiving, migration

**Database Specialists:** performance tuning, backup and recovery, remote DBA services

**EMC:** database and application optimization