The Self-Managing Database: Automatic Health Monitoring and Alerting

Daniela Hansell & Gaja Krishna Vaidyanatha
Product Managers, Server Technologies, Oracle Corporation
Agenda

- Introduction
- Why do you care?
- The Uncertainty Principle – A Required Detour
- Problems with currently available monitoring systems
- Automatic Database Health Monitoring in 10g
- Benefits of Oracle 10g’s server-generated alerts
- Benefits of 10g’s Server Performance Management
- EM 10g Database Control
- PL/SQL APIs for 3rd Party Integration
- Conclusion
- Q & A
Introduction

- Databases are sprouting all over in number
- Databases are growing in size
- Effective Management is key
  - Self-managing
  - Increased efficiency
  - Lower costs
  - Proactive problem detection, diagnosis and resolution
- Server-generated alerts provide the foundation for self-managing performance and database management
Why do you care?

- Solution to all life’s problems is 42
  - Douglas Adams’ “The Hitchhiker’s Guide to the Galaxy” (Check out the “ultimate” version)
- Too much time spent in performance management
  - Sometimes for the wrong reasons!
- Need to empower you to be more productive
- Need to know about “real problems” before they arise
  - Can tolerate only so many bad hair days?
- Server-generated alerts provide monitoring functionality at miniscule levels of overhead
The Uncertainty Principle – A Required Detour

- Proposed by Werner Heisenberg in 1927
- Discovered that the velocity and distance of a subatomic object or particle cannot be accurately measured
- If you shine a light source on an object or particle, the photons affect (move) subatomic particles in a significant fashion
  - Can’t get accurate readings for both distance and velocity
  - Even if distance is measured, velocity measurements are rendered useless
- Conclusion: The Observer Affects the Observed
  - This is not true at macroscopic levels
Problems with currently available monitoring systems - I

- The Observer Affects the Observed
  - At the **Macroscopic** level
- Databases are affected when monitored
  - Contention (Latching, I/O)
  - Potential Downtime (CUI)
  - Resource Overhead
- Excessive Resource Overhead
  - No pain, no gain…unless it is too much pain
  - Anything > 1% is too much
  - Data pings (pull-based mechanism)
- Visit to the Family Physician’s Office
Problems with currently available monitoring systems - II

- Complexity of Setup & Required Customization
  - Scripts
  - Schema Objects
  - Deployment Inflexibilities
    - Manual Configuration
    - Task Repetition
  - Lack of Scalability
    - Issues with Mass Deployment

- Lack of transition from Problem Detection → Diagnosis → Resolution
Database Health Monitoring: *Before* - Alert Polling

- Notify Management Console
- Notify DBA (Page/E-Mail)
- V$ Views
- Poll Metrics
- SGA Structures
- Oracle Server
Automatic Database Health Monitoring: Now - Alert Pushing

Notify Management Console

Process/Agent

Automatic Notification

Server Pushes Alerts

Oracle Server (SGA)

MMON

Alert Notification

ALERT_QUEUE
Advanced Queue
(Automatic Persistence)

Notify DBA
(Page/E-Mail)

Automatic Workload Repository (AWR)
Automatic Database Health Monitoring

- Server-generated Alerts
  - Proactive
  - Out-of-the-box
  - Push instead of Pull
  - Just-in-time

Advisory Infrastructure

Server-generated Alert Infrastructure

Automatic Maintenance Task Infrastructure

Automatic Workload Repository

ORACLE
Benefits of 10g’s Server-Generated Alerts

- No Pinging (Data Push, Proactive)
- Extremely Low Overhead (< 0.1% resource consumption)
- Minimal Configuration
- The Observer Affects the Observed (Subatomic)
- Alert history persisted to AWR
Benefits of 10g’s Server Performance Management

- Comprehensive Advisory Framework
- Integrated drilldowns from Diagnosis to Resolution

Automatic Database Diagnostic Monitor (ADDM)

  SQL Tuning
  Access
  Memory
  Space

  PGA
  SGA
  Buffer Cache
  Shared Pool
  Segment Advisor
  Undo Advisor
Use Case – Where the Rubber Meets the Road

AWR

Oracle Server (SGA)

AWR Snapshots

ADDM diagnoses the problem
(Identifies SQL statements)

SQL-Tuning Advisor tunes the query

Alerts detect problem condition
(Unacceptable Database Wait Time Ratio)

10g Advisors
Type of Alerts

- **Threshold-based:**
  - Associated with a condition such as a threshold being exceeded.
  - Example: Tablespace fullness.
  - Behavior: Alerts automatically cleared when alert condition clears.

- **Non-threshold:**
  - Associated with an occurrence of an event such as an error.
  - Example: ORA-1555 - Snapshot too old.
  - Behavior: Cleared by user acknowledgment.
Alert Types Examples

• Threshold-based
  – Active Sessions Waiting for I/O
  – Active Sessions Waiting for CPU
  – Active Sessions Waiting for Non-I/O-related events

• Non-threshold-based
  – Snapshot too old
  – Resumable Session Suspended
Threshold Configuration

• Most server-generated alerts are configured by setting two threshold values on database metrics
  – Warning Threshold
  – Critical Threshold

• Other Server Alerts correspond to specific database events such as “Snapshot too old”
  – There are NO thresholds associated with these
Out-of-Box Alerts

- Enabled by default in the server:
  - Tablespace Space Usage (warning 85%, critical 97%)
  - Snapshot Too Old
  - Recovery Area Low On Free Space
  - Resumable Session Suspended
- Other alerts may be enabled out of box by Enterprise Manager alert framework.
- Automatic threshold settings in upcoming releases
  - Based on database workload not black magic
Monitoring Architecture

Server Components
- Space Management
- Recovery
- Undo Management
- Metrics Monitoring

Server Alert Infrastructure
- Advanced Queue
- DBA Views

Subscribing Clients
- OEM EMD
- 3rd Party Tools
- End User

ORACLE
Related Technical Details

- Alerts on 10g Server Wait Classes
  - Application
  - Concurrency
  - User I/O
  - System I/O
  - ...
- Measurement done within the Oracle process every minute
- Avoid False Peaks – Calculate # of occurrences
- Persistence maintained with flushes to AWR
- Alerts have Actions to facilitate corrective measures
**New Views**

- **Recent Metrics**
  - V$SYSMETRIC_HISTORY
  - V$SYSMETRIC
  - V$SVCMETRIC
  - ...

- **Metric History**
  - DBA_HIST_SYSMETRIC_HISTORY
  - ...

- **Server-generated Alerts**
  - DBA_THRESHOLDS
  - V$ALERT_TYPES
  - DBA_OUTSTANDING_ALERTS
  - DBA_ALERT_HISTORY
  - ...

- **AWR**

---

Orac
Enterprise Manager 10g
Database Control
Usage Model

- Optionally customize alerts thresholds
- Set up notification rules (Paging, E-mail)
  - Receive notification
  - Review alert details and advice
  - Use Advisors for problem diagnosis and correction
ADDM Finding Details

Period Start Time: Aug 11, 2003 12:00:17 PM
Duration (minutes): 29.9
Finding: Time spent on the CPU by the instance was responsible for a substantial part of database time.

Database Time (minutes): 181.73
Impact (minutes): 57.27
Impact (%): 31.3

Recommendations

- **Details / Category**
  - **SQL Text**: DELETE WAIT_OBJECTS WHERE OBJECT_NAME LIKE "% Bottlenecks%"
    - **Action**: Run Advisor Now
  - **SQL Text**: SHOW SQL Tuning
    - **Run SQL Tuning Now**

Findings Path

Expand All | Collapse All
Findings:

- **Finding**: Time spent on the CPU by the instance was responsible for a substantial part of database time.
  - **Impact (%):** 31.3

Database Time Breakdown For Analysis Period

- **Administrative**: 0%
- **Application**: 0.3%
- **Commit**: 0.3%
- **Concurrency**: 0.6%
- **Configuration**: 0.3%
- **Network**: 1.2%
- **Other**: 6.1%
- **System I/O**: 5.9%
- **User I/O**: 74.2%

Copyright © 1995, 2003, Oracle. All rights reserved.
Database Performance

The image shows a graph from Oracle Enterprise Manager. The graph illustrates database performance metrics such as run queue length and paging rate over time. The graph also displays active sessions waiting and working, indicating the utilization of CPU and other resources. The instance throughput is represented, showing transactions per second.
User I/O Wait Class Drilldown
### SQL Details (SQL ID: asbx90h55nt6g)

```sql
DELETE FROM wait_objects
WHERE object_name LIKE 'IO Bottlenecks%'
```

### Execution Plan

<table>
<thead>
<tr>
<th>Operation</th>
<th>Order</th>
<th>Rows</th>
<th>KBytes</th>
<th>Cost</th>
<th>Time</th>
<th>CPU Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>No explain plan available</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Current Statistics

<table>
<thead>
<tr>
<th>Current Statistics</th>
<th>Execution History</th>
<th>Recommendation History</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PL/SQL APIs for 3rd Party Integration
Setting Thresholds

DBMS_SERVER_ALERTS

SET_THRESHOLD

GET_THRESHOLD

SQL> exec DBMS_SERVER_ALERT.SET_THRESHOLD
(9000, \* Metric Identifier, METRIC_ID in V$METRICNAME
DBMS_SERVER_ALERT.OPERATOR_GE, \* Operator for Warning
Threshold
'60', \* Warning Threshold
DBMS_SERVER_ALERT.OPERATOR_GE, \* Operator for Critical
Threshold
'80', \* Critical Threshold
1, \* Observation Period in Minutes
1, \* Occurrences
NULL, \* Instance Name (NULL implies ORACLE_SID value)
DBMS_SERVER_ALERT.OBJECT_TYPE_TABLESPACE, \* Object Type
'KITCHEN' \* Name of the Object);
Configuring Metrics for Alerting – 3rd Party Use Case

1. Select the alert you are interested in
   1. Query `V$METRICNAME` for all available metrics

2. Set Warning and Critical Thresholds using `dbms_server_alerts.set_threshold()`

3. Alternatively you can use EM’s Alert Configuration pages to do just that.

4. You are done!!!
Consuming Server-generated Alerts – 3rd Party Use Case

1. Subscribe to ALERT_QUE using `dbmsaqadm.add_subscriber()`
2. Create the agent for the subscribing user of the alerts using the `dbmsaqadm.create_aq_agent()`
3. Associate the db user with the AQ agent using the `dbmsaqadm.enable_db_access()`
4. Grant the dequeue privilege using the `dbms_aqadm.grant_queue_privilege()`
5. Register for alert enqueue notification (optional) using `aq$reg_info()` and `dbms_aq.register()`
Consuming Server-generated Alerts – 3rd Party Use Case

6. Configure e-mail and http proxy using various procedures in `dbms_aqelms.set*()`
7. Dequeue the alert using `dbms_aq.dequeue()`
8. Reveal the entire alert message using `dbms_server_alert.expand_message()`

This is what EM does automatically!!!
Conclusion

- Databases have grown in number and size
- Management needs to be automated and self-healing to the extent possible
- The name of the game in 10g is: AUTOMATE
- Server-generated alerts provide the foundation for self-managing performance and database management
Reminder –

Please complete the OracleWorld Online Session Survey.

This was Paper# 40169.

Thank you.