An Oracle White Paper
June 2012

Leveraging the Power of Oracle Engineered Systems for Enterprise Policy Automation

ORACLE

Executive Overview

Oracle Engineered Systems deliver compelling return on investment, with highly tuned hardware and software working closely together to achieve maximum reliability, extreme performance and simplified maintenance.

Given the growing needs of enterprises to automate their policies to deliver personalized, timely, accurate, consistent and auditable customer experiences, it is natural to ask how these engineered systems can be leveraged to meet these needs.

Oracle Policy Automation (OPA) is a proven solution for deploying easily managed policies across an organization. Typical uses include:

- Customer self-service for guided assistance
- In a service oriented architecture as part of a CRM of other business process
- High performance batch mode benefit eligibility determinations, benefit payment schedules, employee payment calculations or income or employer tax calculations

To help customers make the right purchasing decisions to meet their current and future needs, this whitepaper gives example results from running Oracle Policy Automation on Oracle Exadata and Oracle Exalogic engineered systems.

Three different deployment scenarios are examined:

1. Oracle Policy Automation Web Determinations on Exalogic
2. Oracle Policy Automation Determinations Server on Exalogic
3. Oracle Policy Automation Batch Processor on Exadata

Oracle Policy Automation Web Determinations on Exalogic

OPA Web Determinations provides a highly scalable solution for deploying customer-facing self-service guidance and agent-facing assisted decision making. It has been designed from the ground up for high volume citizen-facing solutions for national and global scenarios such as tax advice, visa applications, benefit eligibility and payment calculations.

Test Configuration

To see how OPA Web Determinations can benefit from being deployed on Exalogic, a quarter-rack Exalogic X2-2 environment was configured and tested as follows:

- Four load injectors running The Grinder to simulate user load through a 25 page Web Determinations interview with 1 second think time between pages
- 4 Exalogic compute nodes (half the rack), each with standard configuration of Intel Xeon X5670 2.93 GHz processors, and 96GB RAM, running Enterprise Linux 5.5 64-bit
- Each node configured to run 3 instances of Oracle WebLogic Application Server 10.3.6
- 10Gb/s networking between injectors and compute nodes

Application Scenario

To test this configuration, a Social Services benefit screening interview was used with the default look and feel provided by OPA Web Determinations.

Each simulated user completed the interview via 25 separate web page round-trips, most of which required data entry. Each page uses a combination of graphics, a stylesheet and dynamically generated HTML content. Each interview required 158 HTTP requests total to complete.

The policy model contained roughly 200 rules, and processed 50 different data attributes across two separate entities (objects). 30 separate goals (outcomes) were determined by the policy model.

By the end of each simulated interview, the results for each of the policy model goals had been determined by using the supplied data.

Test Method

Each of the four load injectors were configured to run an agent, using between 900 and 3,500 threads per agent. This simulated up to 14,000 concurrent incoming users. The injectors used round-robin scheduling to select the application server for each user session.

A think time was used of one second between each top-level page request. Although shorter than the think time in typical human interaction, this approach had the benefit of completing the largest number of interviews possible using the available injectors.

When each interview completed, the thread would discard the session and start a new benefit screening application – effectively simulating a new independent user.

No caching of responses was performed by the clients.

Results

As the number of injectors increased, completed applications per minute scaled linearly from around 6,000 to over 16,000. See figure 1.

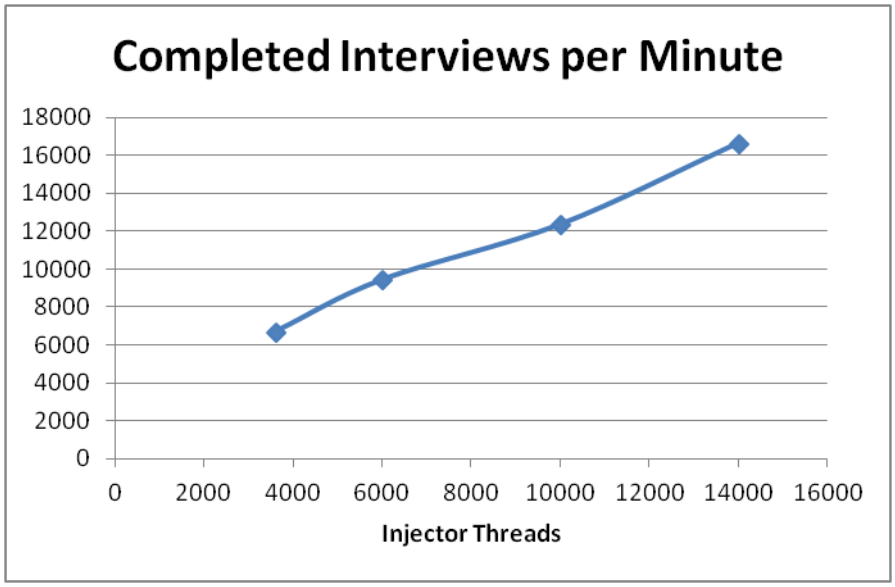

Figure 1: Completed interviews per minute against increasing injector threads, for OPA Web Determinations running on 4 nodes Oracle Exalogic

As the number of injector threads increased, CPU usage remained constant at around 35%, while response time slowed from 250ms to about 1 second. In the tested configuration, the ratio of application servers to CPU cores was 1:8. Effectively 1/3 of the available compute capacity was being utilized. This suggests that even higher applications per minute could be achieved and at a faster response time if more application server instances were servicing requests on the same hardware configuration. At the time of writing, results for alternative application server configurations were not available.

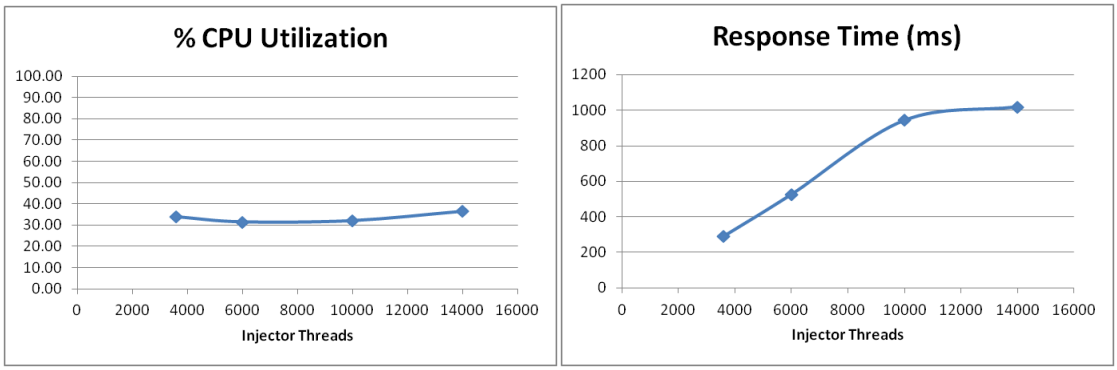

Figure 2: CPU Utilization and Response Time against increasing injector threads, for OPA Web Determinations running on 4 nodes Oracle Exalogic

Conclusion

Exalogic is well suited to running interactive decision making applications using policy models that have been designed and deployed with Oracle Policy Automation. For the scenario tested, one quarter rack of Exalogic can support tens of thousands of interviews being completed by users, in each minute. In a real world situation with increased think time between each call, it is likely that hundreds of thousands of concurrent users could be supported. To maximize performance, consider having up to 12 application server instances on each 24-core Exalogic node.

OPA Determinations Server on Exalogic

Test Configuration

A quarter rack Exalogic X2-2 environment was used to measure the benefits of deploying OPA Determinations Server on Exalogic. It was configured and tested as follows:

- Four load injectors running The Grinder to generate web service load. Think time of 10 seconds between each web service call to Determinations Server.
- 7 Exalogic compute nodes, each with standard configuration of Intel Xeon X5670 2.93 GHz processors, and 96GB RAM, running Enterprise Linux 5.5 64-bit
- Each node configured to run 3 instances of Oracle WebLogic Application Server 10.3.6
- 10Gb/s networking between injectors and compute nodes

Application Scenario

The test this configuration, the aim was to simulate typical scenarios in which business logic is being maintained and managed with OPA, and that logic is deployed within a user-facing application such as a CRM or HR solution.

Four different policy models were used for determinations server performance tests, with varying characteristics. For details of these policy models, see the appendix.

For each scenario, the test simulated an application being used interactively by a business user, with decisions being made by OPA. In this style of integration, each web service call passes business data to OPA, which processes it through a policy model before returning results back to the calling application.

Typically this interaction would be mediated by the application's web services framework or a middleware integration, and use the SOAP API provided by OPA Determinations Server.

Test Method

In each test, all four agents were configured to run a certain number of threads for 20 minutes. The number of threads was increased until sub 1-second response times were no longer being achieved. The think time was selected to simulate an interactive user every ten seconds performing an action that required an answer from OPA Determinations Server.

Each SOAP request to OPA Determinations Server contained a data payload appropriate for the policy model being used in that scenario. OPA Determinations Server then used standard functionality to unpack and process the data through the selected policy model. All applicable decisions that could be reached were made, and the results returned in a standard Determinations Server SOAP response to the caller. The option was turned off to return detailed decision reports to the calling application.

For simplicity of test configuration, each request used identical SOAP requests (and therefore SOAP responses) for every simulated user.

Results

With the FS policy model, scale-up from 30,000 to 60,000 simulated concurrent users saw the number of transactions per second increase from 2,800 to 4,200. Over the same user range, response time increase from approximately 200ms to 1400ms while CPU load increased from 22% to 40%. See figure 3.

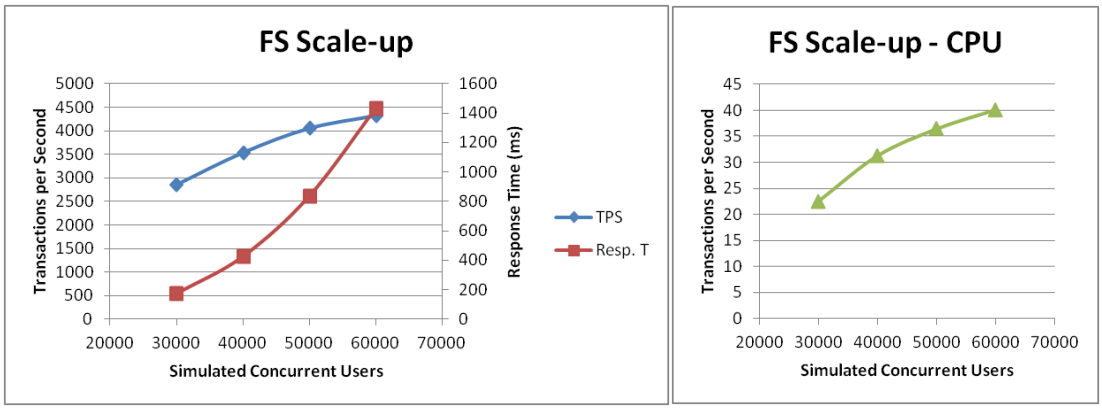

Figure 3: Scale-up figures for FS policy model for OPA Determinations Server running on 7 nodes Oracle Exalogic

With the FK policy model, response time exceeded the 1000ms test threshold with fewer concurrent users. CPU utilization was also much higher. However, the FK policy model is much larger, and many more policy inferences were required to process each request: in fact more than 1000 times more. As a result, the CPUs were kept much busier processing each request, and more overall use was made of the available compute capacity at a given user load.

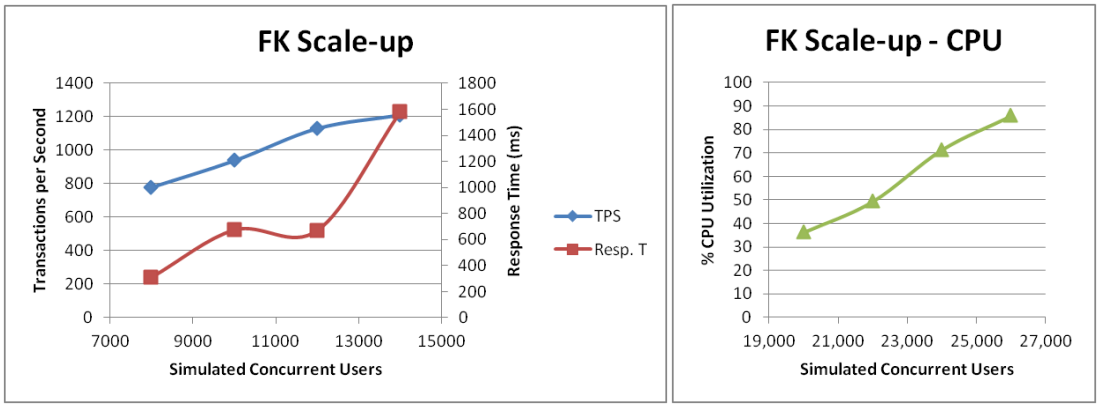

Figure 4: Scale-up figures for FK policy model using OPA Determinations Server running on 7 nodes Oracle Exalogic

Two scenarios that fall between these two extremes were also tested: NY and BC. Each gave similar results. These results are shown in figure 4.

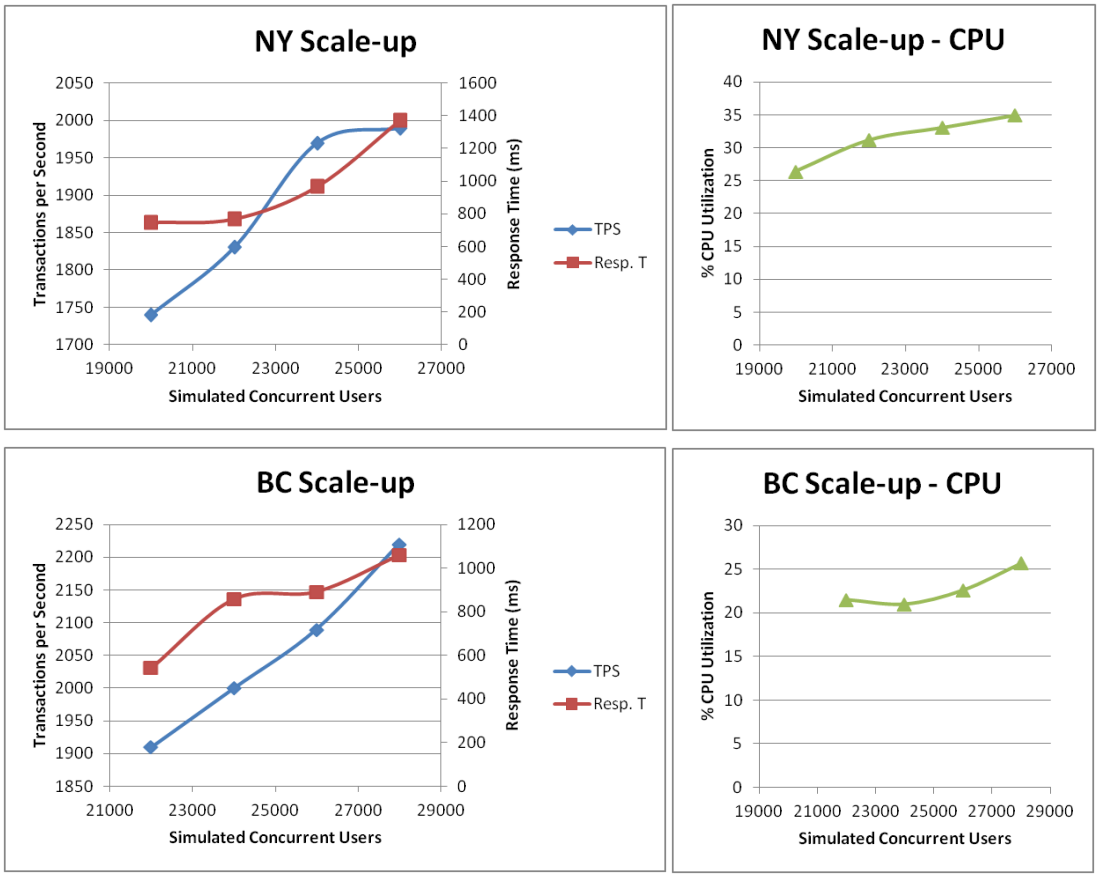

Figure 5: Scale-up figures for NY and BC policy models using OPA Determinations Server running on 7 nodes Oracle Exalogic

Figure 6 shows the number of transactions per second against the complexity of each of the four scenarios tested, while still retaining sub-second response times. Note that the horizontal scale here is logarithmic.

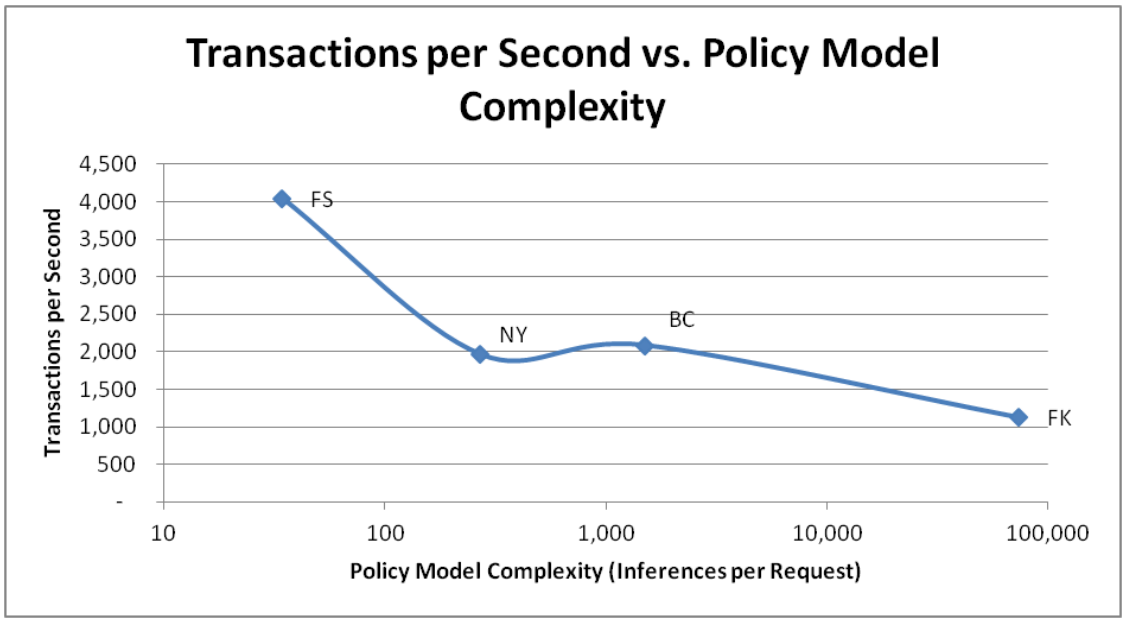

Figure 6: Impact of policy model complexity on the number of transactions per second that can be serviced at sub-second response time

Conclusion

For all four tested policy models, thousands of transactions per second (more than ten thousand concurrent users) were achieved at under 50% CPU utilization. This suggests that a quarter-rack Exalogic could service OPA policy models of varying complexity on behalf of several applications.

To maximize usage of available compute capacity, throughput and user concurrency, consider having up to 12 application server instances on each 24-core Exalogic node.

Policy model complexity has only a logarithmic impact on throughput: 1000 times more policy inferences in each request only caused the number of supportable users to decrease by a factor of 4. This shows that Oracle Policy Automation scales well as policy model complexity increases. It also shows that Exalogic is well suited for using OPA to deliver extremely high performance decision making services within the enterprise.

OPA Batch Processor on Exadata

Test Configuration

A quarter rack Exadata X2-2 environment was used to measure the benefits of deploying OPA Batch Processor on Exadata. It was configured and tested as follows:

- Both database nodes in the quarter rack were used to service database requests
- The OPA batch processor was executed on one of the 2 database nodes
- The batch processor connected to the database using the SCAN listener to automatically balance the database server load
- Standard Exadata 10Gb/s Infiniband networking was used between the node running OPA Batch Processor and the other database nodes

Application Scenario

To test the performance of OPA Batch Processor on Exadata, the scenario selected was to determine whether aged care services provided by agencies were approved for payment.

The information about historical services provided was contained in four related database tables. On average each case had 8 related records in these tables. Results were stored back into columns in two of the tables.

The policy model to calculate these payments used temporal reasoning and other rule logic to determine the time period during which approval was applicable, and to determine if the service period was within that approval period.

Test Method

The database was populated with up to 50 million cases of random data. Each case included between 4 and 12 database rows total, including the top-level case record and all related records. Indexes were placed on the primary and foreign key fields only in each of the three tables. No specific database tuning was performed.

Different test runs of the batch processor used between 1 and 24 separate processes, using the built in capability of OPA Batch Processor to select the number of total processes.

Different data set sizes were also used, varying between 1 million and 50 million cases.

No other load was placed on the database server during the tests.

Results

Figure 6 shows the average number of cases per second that could be processed at different numbers of OPA Batch Processor configured processes. Figure 7 shows the total time taken to process each data set size during these same tests.

In general adding processes allowed more cases to be processed in a shorter period of time. However at lower numbers of cases, adding processes gave no benefit or slowed down overall performance.

Processing speed with one or two processes was already at or close to maximum at 1 million cases, and remained constant for 2 million, 5 million and 10 million cases.

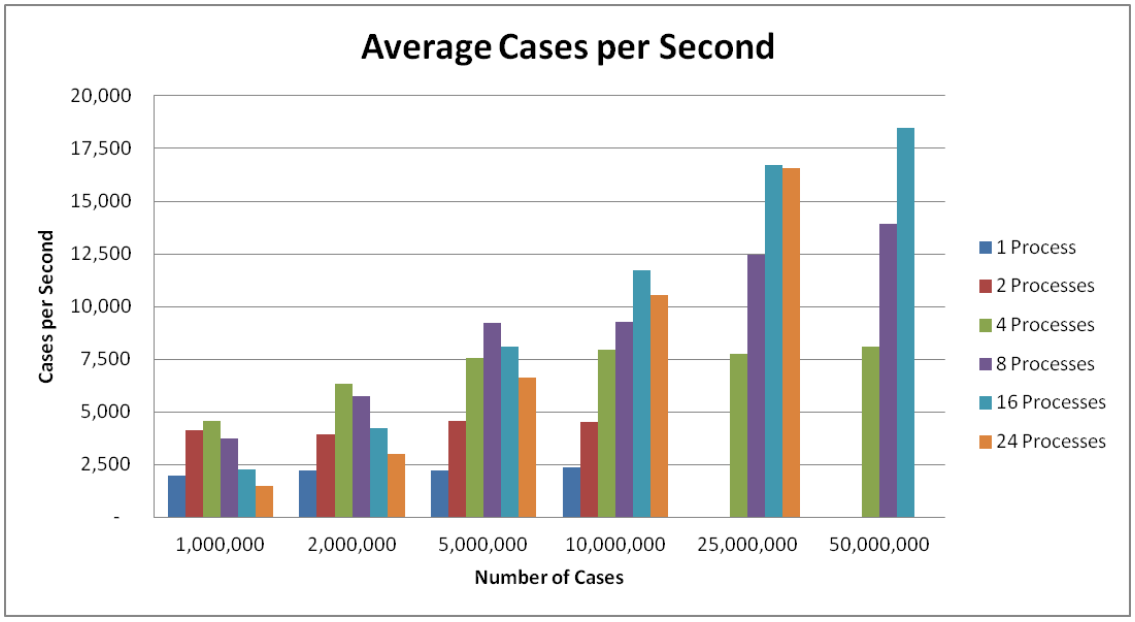

Figure 7: Average cases per second processed by OPA Batch Processor with different numbers of processes.

Processing speed with four processes reached maximum processing speed at 5 million cases, and remained constant for 10 million, 25 million and 50 million cases.

The maximum throughput speed for configurations using 8, 16 and 24 processes was still increasing up to 50 million cases.

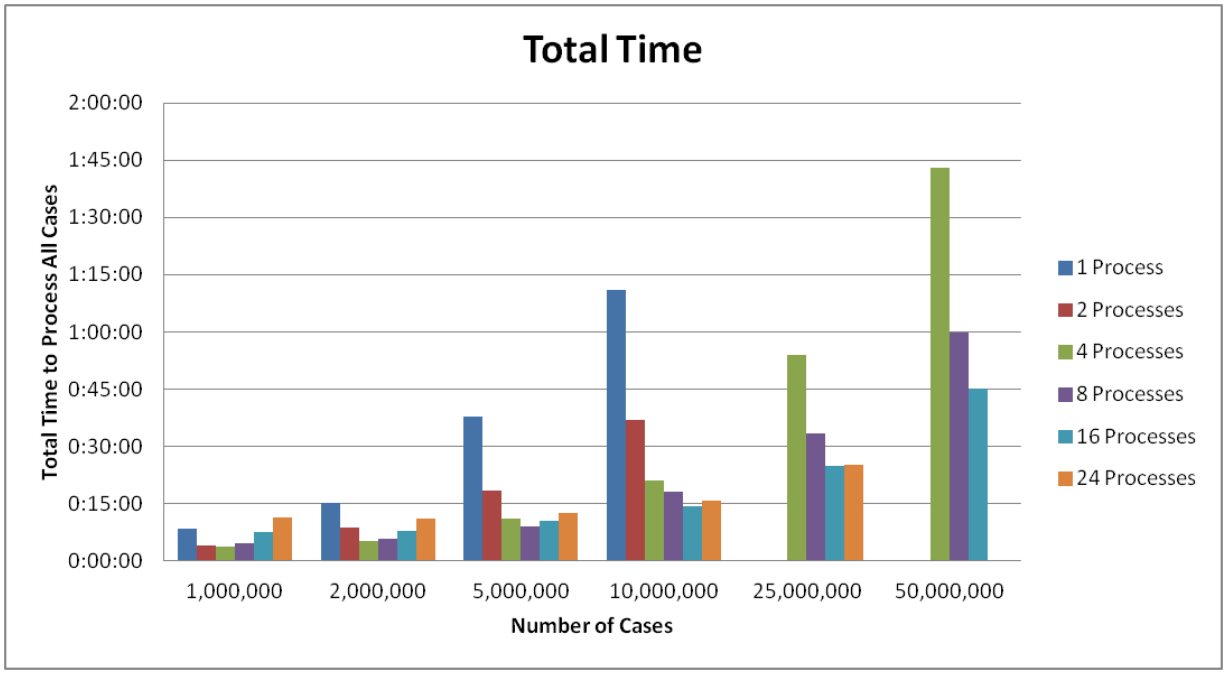

Figure 8: Total time taken by OPA Batch Processor to process different data set sizes, for different numbers of processes[1].

Conclusion

For high volume data enrichment and policy-driven calculation scenarios, OPA Batch Processor on Exadata delivers extremely high throughput with whatever hardware resources are available to dedicate to the batch processing application.

OPA Batch Processor scales linearly on Exadata once maximum processing speed is reached. Therefore, once maximum throughput is achieved at a particular number of processes, the time required to process larger data sets can be simply extrapolated.

For large data sets, configuring OPA Batch Processor to spawn additional processes delivers near linear improvements in overall completion time. It is worth noting that tests were deliberately performed only up to the available number of CPU cores. It is expected that

[1] For data sets of 50 million and 100 million cases, tests were not completed using 24 parallel processes due to high database contention, and are therefore not included in this report. Database tuning is always recommended for any production batch processing scenario. Such tuning was not performed for these tests.

increasing OPA Batch Processor processes beyond the number of CPU cores on the Exadata node will result in slower overall processing.

While tuning the database was beyond the scope of the testing conducted for the purposes of this report, such tuning will likely yield significant additional performance benefits. For example, optimizing for a high volume of parallel UPDATEs against the same database tables would allow all processors on an Exadata node to be used without experiencing database contention.

In summary, Exadata is well-suited to achieving high throughput calculations using Oracle Policy Automation Batch Processor.

Appendix

Policy Models Used for OPA Determinations Server Performance Tests

Four different policy models were used in the above performance tests. Each policy model had different characteristics as noted. These policy models were based on real world business scenarios of Oracle Policy Automation customers.

FS: Calculates the fraud score of an insurance applicant. Policies were predominately captured using an Excel rule table. Request size was 3.4Kb, response size was 3.3Kb. 34 policy inferences in each request.

NY: Calculates the eligible pension payment for an applicant. All policies were captured in Word rule documents. Request size was 28Kb, response size was 251Kb. 270 policy inferences in each request.

BC: Calculates if a household application is eligible for social assistance. All policies were captured in Word rule documents. Request size was 15Kb, response size was 228Kb. More than 1,200 policy inferences in each request.

FK: Calculates if an applicant is eligible to receive reimbursement for medical treatment. Policies are modeled in a mixture of Excel and Word rule documents. Request size was 13Kb, response size was 18Kb. 74,000 policy inferences in each request.

ORACLE

Leveraging the Power of Oracle Engineered Systems for Enterprise Policy Automation
May 2012
Author: Davin Fifield

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com