Performance Testing, Modeling and Piloting – All at Once!

By Dave Jewell (jewell@us.ibm.com)
Contributed for use by the Workshop on Performance and Reliability
Session: WOPR15, Campbell, CA
Date: October 27 – 30, 2010

Abstract

We are accustomed to doing performance testing of a new application or release prior to deployment, but what happens when the application being tested is already being deployed as a production pilot? This experience report discusses a situation where one enterprise's aggressive deployment schedule for a multi-store labor management application required creative use of performance test scenarios plus performance information coming from both pilot experience and performance modeling to rectify potential batch window problems with its sales associate work schedule generation process.

Business Context

A major North American clothing retailer was seeking to improve the efficiency and cost-effectiveness of the way in which it managed the time of its sales associate labor force across its 2,500+ stores. Sales associate wages constitute a significant portion of its overall costs, and the retailer's management team felt that in a challenging business climate, centralizing the management of sales associate labor would provide financial benefits.

The retailer's requirements included the ability to track the hours of sales associates clocking in and out, and the ability to automatically generate optimized work schedules that took into account labor requirements based on projected store traffic plus various constraints on the sales associate labor force at each store (e.g. availability, applicable state/local labor laws), with the intent of improving store-level profitability.

Technical Context

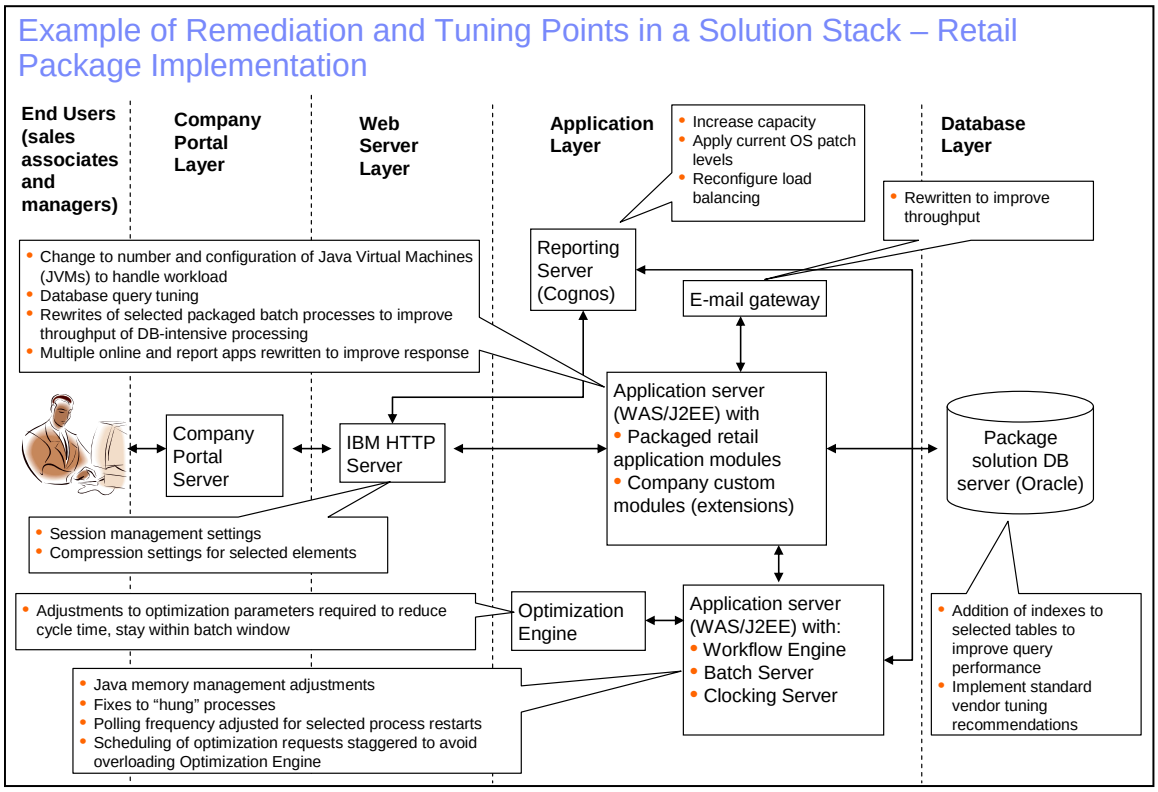
The overall project is an example of what IBM refers to as a "package integration" project, in that a vendor packaged software solution was being integrated with the customer's existing IT systems. Part of the project involved replacing store-based instances of its legacy labor management package, while another part involved the typical package development and tailoring activities commonly referred to as "RICEF" (Reports, Interfaces, Conversions, Extensions, and Forms).

The package chosen was a labor management solution developed under the Workbrain brand now owned and supported by Infor. The bulk of the application was developed on

a J2EE platform, using IBM Websphere Application Server and Oracle databases. Due to the size and technical complexity of the project, personnel from both the customer's organization and multiple vendors were involved in the development, testing and deployment of the solution.

Project Considerations

Because of the complexity of the solution and the fact that the package was being tailored extensively for this retail company's situation, performance testing revealed a number of opportunities for tuning and performance improvement.



Adding urgency to the situation was the fact that due to previous delays, production pilot deployment began months before the overall performance test effort and some stages of development were able to finish. Because of this, the retailer mandated a performance test strategy in which performance test data and activity volumes in the performance test environment would precede those volumes being deployed into production by several weeks, so that the performance test results would disclose any performance, reliability, capacity and scalability issues before they happened in production.

This strategy had a number of implications:

- In addition to developing performance test scripts, significant effort had to be put into creating synthetic performance test system database data, representative of the future deployment scenario in question (e.g. number and mix of stores deployed, store forecast activity, store employee populations, etc.).
- If necessary, deployment could be temporarily delayed to rectify such problems, but with the intent of resuming and catching up so that overall deployment would complete on time before standard year-end production "freeze" went into effect.

Batch Generation of Sales Associate Work Schedules

Of particular interest from a performance test scenario perspective was the generation of weekly sales associate work schedules. Nightly batch processing in support of schedule generation and upstream/downstream processes proceeded according to the following schedule (major tasks shown):

- Sunday a.m. – Historic Sales and Traffic Data, Availability Data
- Sunday p.m. – Team Data
- Monday p.m. / Tuesday a.m. – Budget Update, Forecast Generation
- **Tuesday p.m. / Wednesday a.m. – Schedule Generation**
- Wednesday p.m. – Schedule Regeneration
- Thursday p.m. – Schedule Publishing and E-mail Distribution
- Friday p.m. / Saturday a.m. – Payroll Data Export

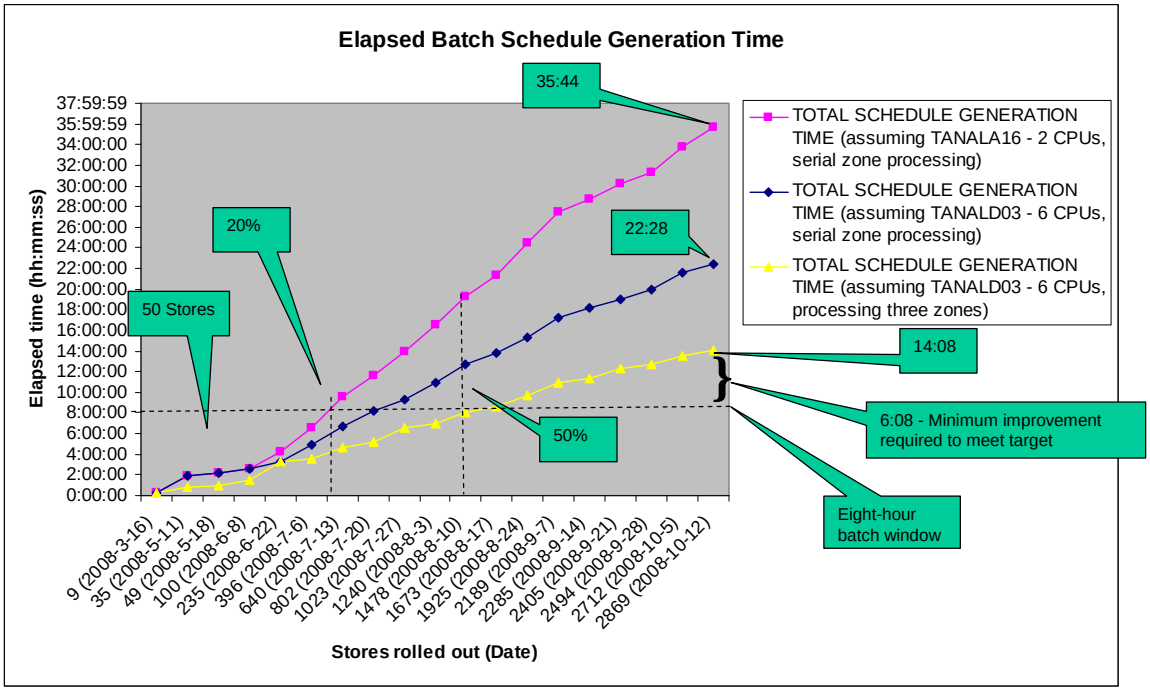
The Tuesday overnight processing window for schedule generation was critical because schedule generation needed to complete for all 2,500+ plus stores. Because schedule generation is a complex optimization problem, the Workbrain solution employed dedicated servers, a specialized modeling language (Mosel) and a sophisticated optimization solution (DASH) to support schedule generation.

Interrelationship of Pilot, Test and Modeling Scenarios for Schedule Generation

The Workbrain labor management solution partitioned the stores into zones based mostly on geographic and brand groupings. In the initial solution, only one zone could undergo scheduling processing at a time, although multiple stores in the same zone could undergo scheduling processing as long as DASH optimization engines (instances) were available. The figure below shows the results of ***early performance testing*** for three different performance test zones using different capacity and parallelism configurations.

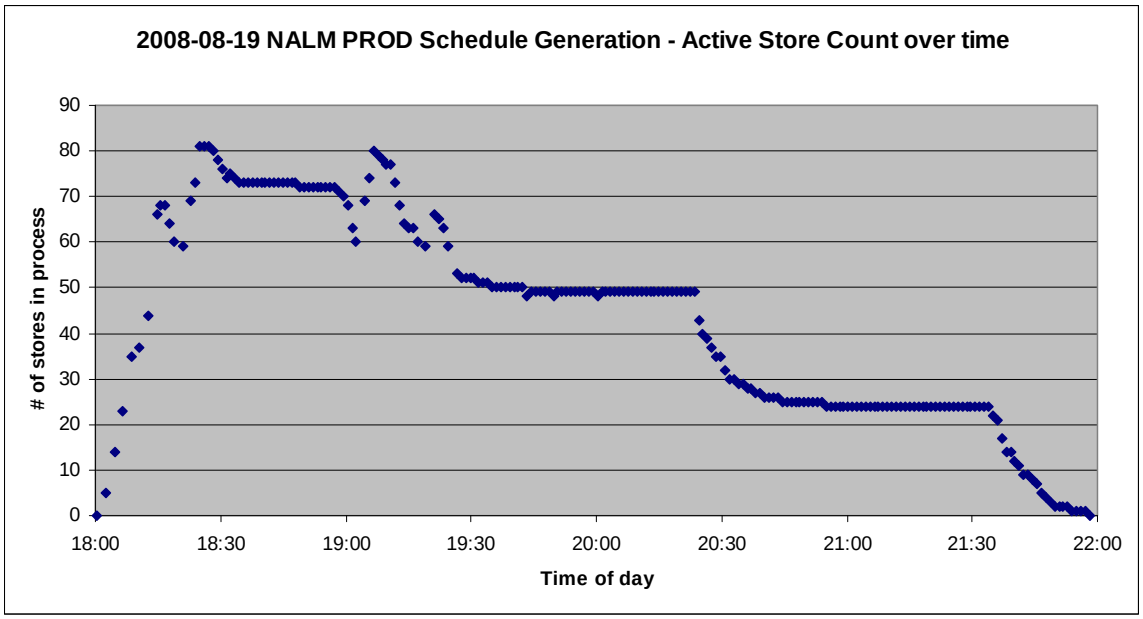
Scheduling Scenario #	Scheduling Scenario Description	Average # of Concurrent DASH processes	Average Elapsed DASH Time per Store	Overall Elapsed Time	Overall Elapsed Time
1	Test of Zone 96128 - 2 Batch CPUs (5/12)	12	0:05:18	1:31:00	Had to throttle back batch maximum tasks to avoid overrunning server capacity.
2	Test of Zone 96128 - 6 Batch CPUs (5/13)	20	0:05:23	1:09:00	IMPROVEMENT # 1 - Increase capacity of Batch server from 2 to 6 CPUs, 8G to 24G memory, by utilizing TANALD03 for schedule generation.
3	Test of Zone 90036 - 6 Batch CPUs (5/19)	TBD - (Assuming ~20)	0:10:24	1:39:00	Note higher average DASH scheduling time than for zone 96128.
4	Test of Zone 92357 - 6 Batch CPUs (5/19)	TBD - (Assuming ~20)	0:14:08	3:04:00	Note higher average DASH scheduling time than for zone 96128.
5	Test of Overlapping Zones (96128 / 90036 / 92357 running concurrently) - 6 Batch CPUs (5/19)	33	0:08:47	3:52:00	IMPROVEMENT # 2 - Run up to three zones concurrently at a time instead of strictly serializing zones.

Using the performance test results, ***simple performance models*** were developed to make projections on how elapsed time could be expected to grow in production as the solution was rolled out to more stores over time, again using different capacity and parallelism assumptions.



Schedule generation was scheduled to start at 6 pm US Pacific time, with a goal of completing in eight hours to minimize impact to the next morning's batch and online processing. Since the performance model showed significant problems in being able to do that, a number of corrective actions were taken to address the problem.

Since pilot production deployment had already started, ***production performance results*** were taken to help validate progress. The following chart shows the number of concurrent DASH optimization engines (0 – 80) active during the batch schedule generation cycle, with 1,000 stores completing their processing in around four hours. This result was measured three months after the initial batch performance testing and projections had been made, when a subset of the performance and reliability improvements had been deployed.



Even with some of the reliability improvements in place, part of the challenge of improving performance was due to the fact that shortly after the first two hours of the test, the problem was not one of needing more DASH optimization engines and server capacity, but the fact that the few remaining stores to be run had elapsed times longer than what was typical for a store (e.g. 5 minutes or less). The variances in processing times were in part due to the different sizes, labor pools and forecasts being managed, although in a number of cases it is possible that data setup errors for certain stores caused them to have longer elapsed schedule generation times than necessary.

Actions Taken and Outcomes

- The package vendor fixed reliability problems with the batch processes used to manage DASH optimization that occasionally led to stalled or terminating schedule generation tasks.
- Resource, memory, process start/restart and Java tuning measures were taken to ensure DASH optimization could perform well for the number of concurrent DASH optimization engines assigned.
- Custom Unix shell scripts were written which allowed new zones to be launched in parallel to keep 60 – 80 DASH optimization engines active at any given time, as long as there were zones that had not yet been launched.
- Continuing attention was given to data clean-up, and in some cases optimization assumptions, pertaining to stores with unusually long running schedule generation elapsed times.
- After negotiation with the retailer's business stakeholders, tuning parameters specific to the retailer's scheduling model were change to reduce processing time.
- By the time the performance test had ended, a credible path to attaining the eight-hour batch window had been demonstrated.

Lessons Learned

Obviously the situation where an application has begun pilot deployment prior to completion of development and performance test is not an ideal or recommended one, but it probably happens more than performance and reliability testers would prefer. The experience from this project suggests that in this situation, we can use what we are learning from both test and production at the same time, and in the case of a staged deployment, make selected use of modeling to give our stakeholders as much of an idea as we can about what the future holds.

In this particular case, performance testing, modeling and pilot deployment results all played a part in contributing to the success of the project and our understanding of schedule generation performance, since each had its own "test scenarios" of sorts.

- ***Performance test results*** allowed us to see what a live system would do in various test data, capacity and parallelism scenarios, although in the early stages of testing, not enough data was present to do a live test of the 100% deployment scenario.
- ***Performance modeling results*** allowed us to make projections into the future based on what had been learned from performance testing.
- ***Pilot production results*** allowed us to get weekly feedback (on Wednesday mornings) on the results of live production schedule generation to guide our ongoing testing, modeling and tuning efforts.