Test Plan: MSE Project

September 18, 2009

Prepared by Doug Smith
Version 0.1

Table of Contents

Revision History

Version	Date	Changes
0.1	9/18/2009	First draft.

Introduction

This document presents the test plan for my MSE project, and is based on the structure and content defined by IEEE standard for software test documentation [0]. The goal of this test plan is to explain the approach, scope, techniques, and tools used for testing the software constructed during this project.

The primary focus of the testing associated with this project is scale, performance, and stability testing for a workflow execution engine that uses a replicated cache. While some amount of functional testing will performed, performance testing will be emphasized in order to concentrate on system level tested aimed at determining the suitability of various cache configurations for use in an enterprise scale workflow management system.

For functional testing, the use cases enumerated in the Vision document [1] will be tested through their public service interfaces. Additionally, unit tests will be used to validate the correct functionality of several components used to implement the services.

Scale and performance testing will also use the service interfaces for driving the test, but will focus on the architecturally significant use cases enumerated in the Software Architecture document.

References:

[0] IEEE Std 829-1998 IEEE Standard for Software Test Documentation, The Institute of Electrical and Electronics Engineers, 1998.

[1] Vision Document: MSE Project, Doug Smith, http://people.cis.ksu.edu/~dougs/Site/Phase1_Artifacts_files/vision.pdf

[2] Software Architecture: MSE Project, Doug Smith

[3] Formal Requirements Specification: MSE Project, Doug Smith

[4] soapUI WebService Testing Tool - http://www.soapui.org/

[5] MSE Project Plan, Doug Smith

Test Items

The requirements section in the Vision document [1] outlines the functional requirements to be tested, as well as the system quality attributes. Functional and system quality attributes will assessed using tests written against the service interfaces described in the Software Architecture document [2], and the formal requirements specification [3].

Features to Be Tested

The use cases described in the Vision document [1] are in scope for functional testing. The architecturally significant use cases [2] are in scope for system quality attribute testing.

Features Not To Be Tested

Given the system under test is a slimmed down abstraction of an enterprise system, all features will be covered by a functional test, tests designed to establish system quality attributes, or both.

Approach

Cache Configuration

The goal of the project is to establish a cache configuration that offloads processing from the database with no performance degradation, scalable to 16 cache cluster members. Ideally, this goal will be satisfied by targeting the most promising configurations based on analysis, as opposed to testing every possible configuration combination.

There are several configuration options that must be taken into account. At a gross level, configuration options are:

- The selection of the entities and queries to cache, expressed in the Hibernate configuration.
- The region associated with the cached entities and queries, expressed in Hibernate configuration.
- The manner in which sessions interact with the second level cache, expressed in Hibernate configuration.
- At the cache level, selecting replication vs. invalidation.
- At the cache level, selecting a locking strategy.
- At the cache level, selecting an isolation level if pessimistic locking is configured.
- At the cache or cache region level, selecting an eviction policy.
- State transfer configuration at the cache member level.

There are also several tuning configurations available on the underlying multicast protocol used for communication between the cluster members. Protocol tuning options include:

- Transport protocol
- Initial group membership discovery protocol
- Configuration of Merge protocol for recovery from a network partition.
- Failure detection protocol
- Reliable message transmission configuration
- Fragmentation configuration.
- Group membership protocol
- Security configuration
- State transfer configuration

Based on the system under test, I predict the configuration options that will have the greatest impact of performance and scalability are replication vs. invalidation, the locking strategy and related isolation level if applicable, and the state transfer configuration.

My prediction is invalidation (as opposed to replication) will yield the best results, followed by replication using optimistic locking, the replication using pessimistic locking ordered based on locking strategy (favoring policies favoring concurrency over correctness).

The test strategy will be to select reasonable baseline parameters for those items not identified above as having the greatest impact, then testing different configurations of the high impact configuration options.

Functional Testing

The approach for functional testing will be to write test cases expressed in terms of the system's service interfaces, and to execute the tests using the soapUI tool [4].

The following quality attributes shall also be covered by functional tests:

- QA4 – role based access control
- QA5 – basic authentication and identity management

System Quality Attributes

To establish system quality attributes, specific tests are needed to establish system quality attributes not covered in functional testing.

QA1 – No Degradation in Throughput or Response Time

To establish no degradation in throughput or response time when a caching solution is introduced, as well as the characterize performance to compare predicted improvements with actual performance, a workload of 70% reads and 30% writes will be used to drive the system at different numbers of virtual users. This will be done in a systematic way to show scalability of the system as users increase, and will proceed until system performance degrades.

Given that a caching solution is going to be sensitive to the amount of data that is cached for a transaction, as well as the number of items that are cached, the baseline will be created for different data profiles as well.

Once the system performance has been base-lined, the baseline tests for various caching configurations will be repeated, and the results will be compared to the baseline.

QA2 – Functional Correctness Maintained when Caching

No specific tests will be created to establish this quality attribute. This will be a constraint on the baseline performance testing, that will not only measure throughput and response times, but must also check system responses for correctness.

QA3 – Application Availability Maintained when Cluster Member Lost

This will be a variant on performance testing, where in the course of a performance test a cluster member will be killed, with the effects on the system observed. This test will establish if the system is still available when a cluster member is lost.

QA6 – Scale to 16 JVMs

To establish this system quality attribute, the baseline performance test at a high workload will be repeated for multiple system JVM configurations, and will include measuring a 16 JVM cluster configuration.

QA7 – Data Consistency Across Cluster Members

At the end of each performance test, cache contents shall be dumped for each cluster member, and the contents compared.

QA8 – Cluster Members can be Added and Removed without Errors

For this test, during moderate and high workloads, a cluster member will be killed and restarted, and the effect on the system observed. It should be possible for the restarted cluster member to rejoin the cluster and perform the necessary state initialization appropriate for the caching configuration without generating service errors.

QA9 – Use of Persistence Tier Computing Resource are Reduced

The amount of computing resources (CPU utilization, IO activity, etc) used on the database tier will be measured for selected workloads and configurations, with baseline (no caching) results compared to the results obtained using various caching configurations and strategies.

Pass/Fail Criteria

For claiming testing has passed, the following must be true:

- All functional tests must satisfy all relevant requirements, and must conform to the architectural design and formal specification.
- At least one caching configuration must be produced that satisfies all specific system quality attribute test cases.

Suspension Criteria and Resumption Requirements

Given the development and test staff are comprised of a single developer, no formal suspension and resumption criteria are needed.

Test Deliverables

Deliverables include:

- Test plan
- Test logs. Test logs must document the cache parameter configurations associated with the test run.
- Test summary

Testing Tasks

Test tasks are captured in the project plan.

Environmental Needs

To draw valid conclusions about test results, sufficient computing resources are required to produce scale and performance results that characterize limits of the software implementation and cache configuration, without the results being limited artificially by an undersized compute environment.

Details on the deployment and physical architecture of the system can be found in the architecture document [2].

Risks and Contingencies

The biggest risks to testing are insufficient computing resources, and having the scope of the testing extend beyond the amount of work needed to successfully complete this project.

A contingency for insufficient computing resources is to either scale test to the level of hardware that can be enlisted for testing.

For scope management, the focus should be on the most promising cache configurations, with some configurations dropped from testing should too much time be spend in testing.

Approvals

This document is subject to approval by the MSE Project committee members.