Assessment Evaluation: MSE Project

November 5, 2010

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 summarizes the testing done for my MSE project, covering unit testing, functional testing, and testing related to establishing the system quality attributes. The original test plan was set forth in [1], this document captures testing done relative to the plan, as well as deviations from the original plan.

References:

[1] MSE Test Plan, Doug Smith, http://people.cis.ksu.edu/~dougs/Site/Phase_2_Artifacts_files/test_plan.pdf

[2] Sonar Code Quality Platform, http://www.sonarsource.org/

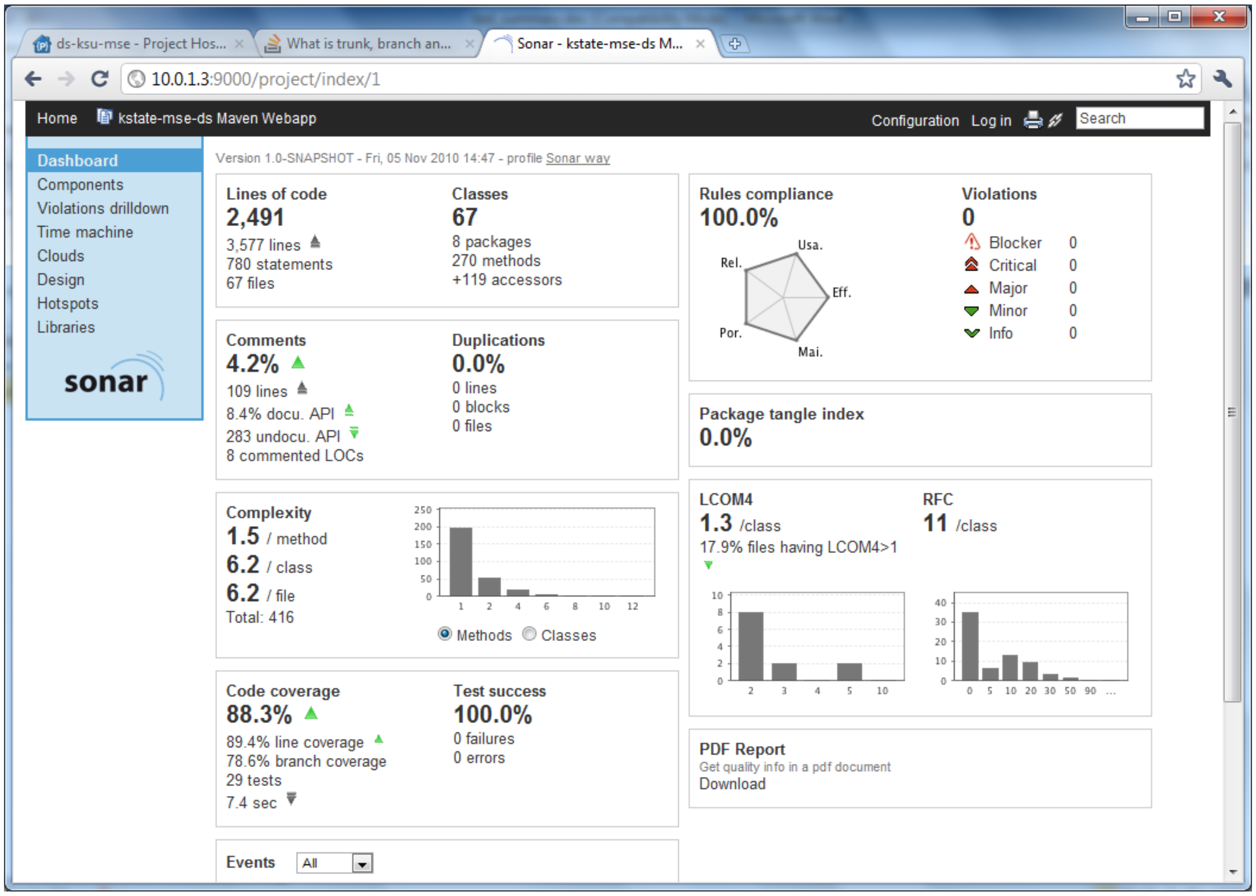
[3] Cobertura Test Coverage Tool, http://cobertura.sourceforge.net/

[4] MSE Software Architecture Document, http://people.cis.ksu.edu/~dougs/Site/Phase_2_Artifacts_files/sad.pdf

Unit Testing and Functional Testing Summary

As good development practice, unit tests were written to verify the correct functioning of the classes comprising the implementation of the system. The sonar dashboard [2] was used to monitor code coverage, measured using the Cobertua test coverage tool [3].

As of 11/5/2010, test coverage was at 88.3%, based on 89.4% line coverage, and 78.6% branch coverage.



Version 1.0-SNAPSHOT - Fri, 05 Nov 2010 14:47 - profile Sonar way

Lines of code
2,491
3,577 lines
780 statements
67 files

Classes
67
8 packages
270 methods
+119 accessors

Rules compliance
100.0%

Violations
0

Blocker	0
Critical	0
Major	0
Minor	0
Info	0

Comments
4.2%
109 lines
8.4% docu. API
283 undocu. API
8 commented LOCs

Duplications
0.0%
0 lines
0 blocks
0 files

Package tangle index
0.0%

Complexity
1.5 / method
6.2 / class
6.2 / file
Total: 416

LCOM4
1.3 /class
17.9% files having LCOM4>1

RFC
11 /class

Code coverage
88.3%
89.4% line coverage
78.6% branch coverage
29 tests
7.4 sec

Test success
100.0%
0 failures
0 errors

PDF Report
Get quality info in a pdf document
Download

Examples of items not covered in test cases are:

- All instances of exceptions thrown when database access fails. This is acceptable as there is some test coverage of this that validates the unchecked exceptions generated in these scenarios are caught and handled by the service level fault handler.
- Coverage of static String constants defined in a class. The fact this is flagged as not covered is more a quirk of the coverage measurement tool, as opposed to a risk accepted as part of the project.
- Coverage of all methods defined on Hazelcast Loader and Store interfaces. The functional tests associated with the application did not cause these methods to be invoked (invocation of the methods is under the control of the Hazelcast IMDG). Note the code these Loader and Store interface implementations call is covered by other unit tests; some of the integration between Hazelcast and the Loader and Store implementations was not covered.

Functional testing (beyond that at the outer layers of the unit tests) was performed via single thread runs of the scale test drivers, and invocation of the web services using soapUI with manual inspection of the results. In terms of the overall testing, most of the functional testing was covered in unit tests at the service interface/transaction script level.

Assessment of System Quality Attributes

Quality Attribute 1 – No Degradation in Throughput or Response Time

A requirement of the caching solution was that it would not degrade throughput or response times relative to a non-caching enabled solution. In other words, achieving scalability at the price of significant reductions in performance is unacceptable.

This was proven in the architecture assessment of the initial and final architectures used for this system. The initial architecture showed increased performance with caching enabled for data access, the second architecture showing an additional increase over the initial architecture when database writes were performed asynchronously relative to the web service transactions. See [4] for details.

Quality Attribute 2 – Functional Correctness Maintained With Caching Enabled

This was demonstrated both via the unit tests, and via the scale tests.

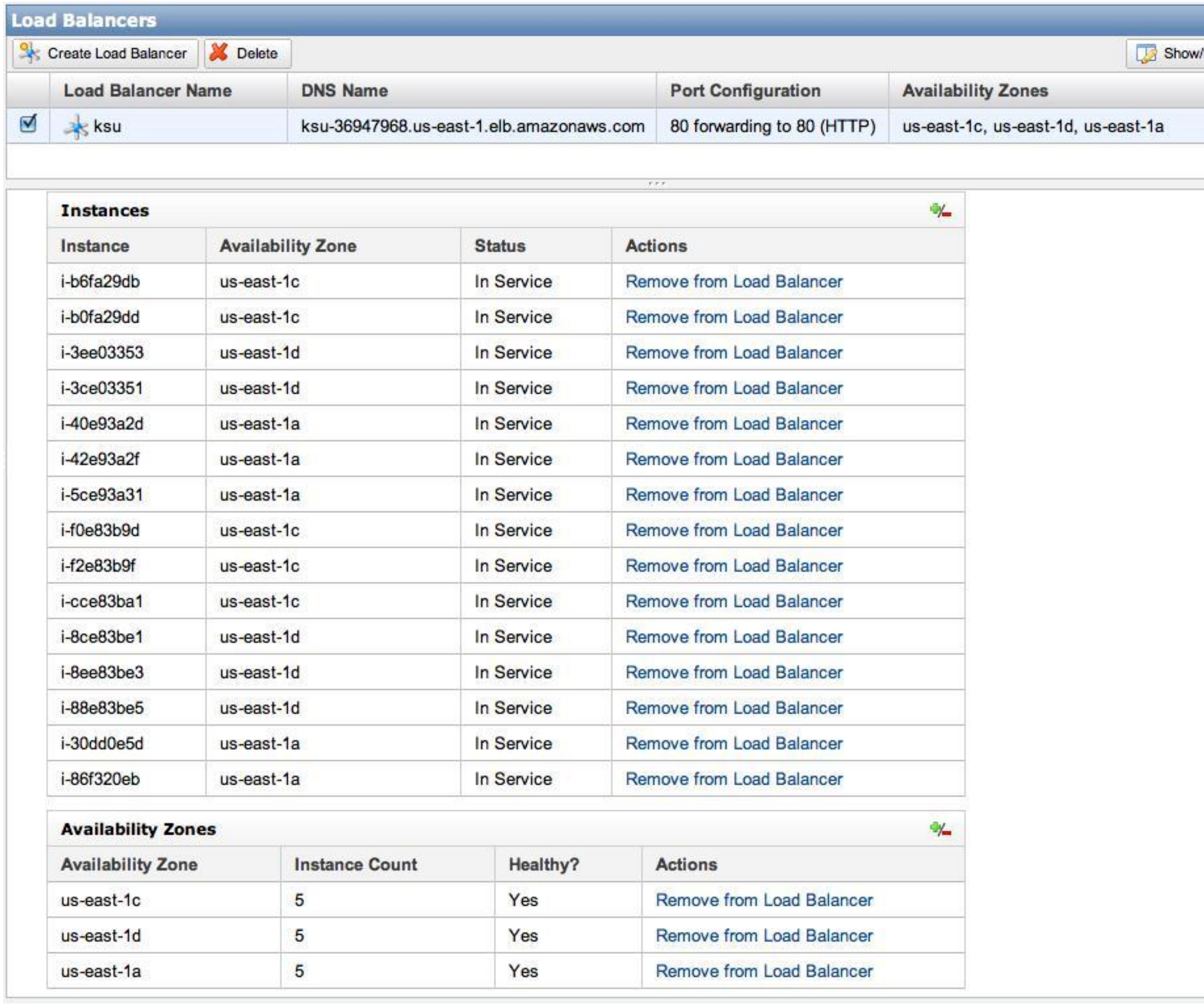
Quality Attribute 3 – Application Availability Maintained when Cluster Member Lost

This was demonstrated during scale testing when adjusting cluster member sizes. When shutting down the application or when removing cluster mebers, unavailability of cluster members was detected by the EC2 load balancer, removing them from the cluster and routing clients away from failed nodes. Additionally, using the Hazelcast monitoring application, it was observed that data stored on nodes removed from the cluster were backed up on others nodes in the cluster in response to cluster members being removed.

Quality Attribute 6 – Scale to 16 Nodes

Scalability is the ability to handle another unit of work by adding an additional unit of computing resources.

To test the scalability of the system, I used Amazon EC2 to procure computing environments. The database ran on a dedicated EC2 server, and application servers were run across three availability zones, with a load balancer distributing load between the servers.



Load Balancers

	Load Balancer Name	DNS Name	Port Configuration	Availability Zones
✓	ksu	ksu-36947968.us-east-1.elb.amazonaws.com	80 forwarding to 80 (HTTP)	us-east-1c, us-east-1d, us-east-1a

Instances

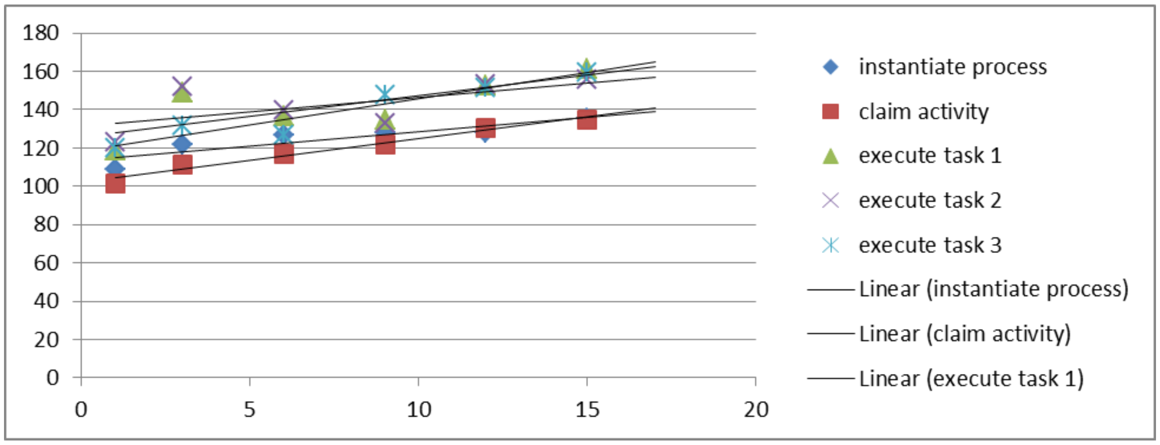
Instance	Availability Zone	Status	Actions
i-b6fa29db	us-east-1c	In Service	Remove from Load Balancer
i-b0fa29dd	us-east-1c	In Service	Remove from Load Balancer
i-3ee03353	us-east-1d	In Service	Remove from Load Balancer
i-3ce03351	us-east-1d	In Service	Remove from Load Balancer
i-40e93a2d	us-east-1a	In Service	Remove from Load Balancer
i-42e93a2f	us-east-1a	In Service	Remove from Load Balancer
i-5ce93a31	us-east-1a	In Service	Remove from Load Balancer
i-f0e83b9d	us-east-1c	In Service	Remove from Load Balancer
i-f2e83b9f	us-east-1c	In Service	Remove from Load Balancer
i-cce83ba1	us-east-1c	In Service	Remove from Load Balancer
i-8ce83be1	us-east-1d	In Service	Remove from Load Balancer
i-8ee83be3	us-east-1d	In Service	Remove from Load Balancer
i-88e83be5	us-east-1d	In Service	Remove from Load Balancer
i-30dd0e5d	us-east-1a	In Service	Remove from Load Balancer
i-86f320eb	us-east-1a	In Service	Remove from Load Balancer

Availability Zones

Availability Zone	Instance Count	Healthy?	Actions
us-east-1c	5	Yes	Remove from Load Balancer
us-east-1d	5	Yes	Remove from Load Balancer
us-east-1a	5	Yes	Remove from Load Balancer

A work load for a single cluster member was established, then multiples of the workload corresponding to cluster size was used to establish scalability, cluster size referring to the number of application server machines. Specifically, for each of the 4 test scenarios, the baseline workload was 1 thread running the scenario transactions 250 times. With the cluster at its maximum size of 15 application server nodes (plus the database node for a total of 16 nodes), 15 client threads invoking the test scenario 250 times was used.

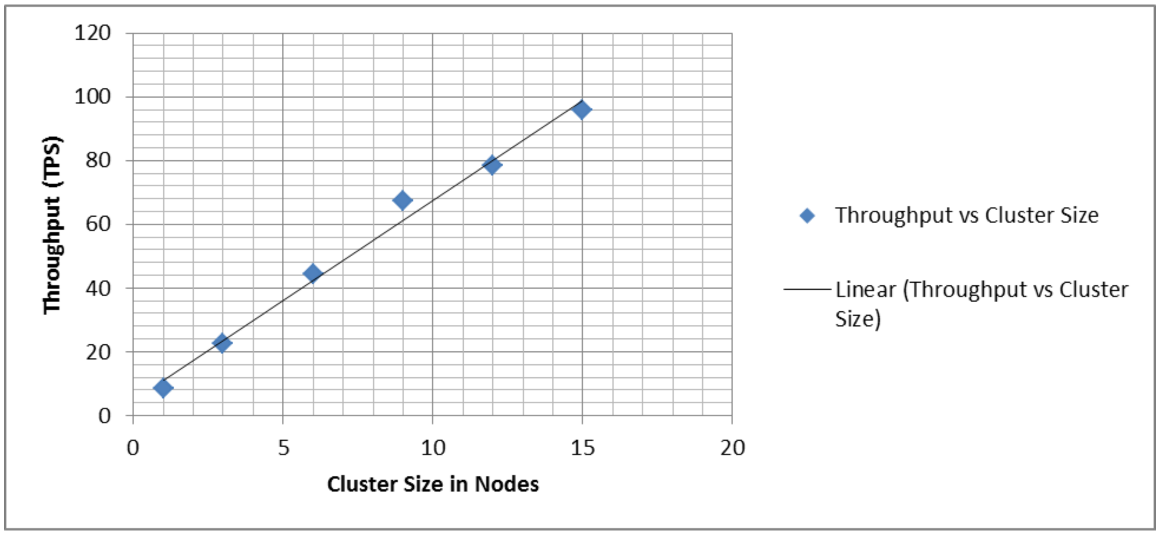
The following chart shows the response times of each transaction for different cluster sizes.



Note that while the response times are trending up as the number of cluster members are added, the absolute increase in response time is low (30 – 40 ms difference between 1 and 15 cluster members). Also note this increase may be more indicative of resource limitations from a test client perspective than real increases in response times from the server environments.

The above chart shows that response times are maintained within a reasonable tolerance as the workload and cluster size is multiplied, which is an important characteristic of scalable systems.

The following chart shows the throughput of the system as compute resources are added.



The important result illustrated by this chart is the linear relationship between the cluster size in nodes and the throughput: adding a unit of computing resources, specifically additional EC2 nodes, produces the ability to handle another unit of workload.

Quality Attribute 7 – Data Consistency Across Cluster Members

This quality attribute was included based on the initial solution, which involved replication to all cluster members, in some cases asynchronously on transactions commit. With the revised architecture, data replication is performed synchronously when writing through the cache, and is replicated to a single backup. Given the solution adopted, data consistency across cluster members is no longer problematic from an application development perspective as it is a function of the correctness of the IMDG software

Quality Attribute 8 – Cluster Members Can Be Added and Removed Without Errors

This was demonstrated during assessment of Quality Attribute 3 (Application Availability Maintained when Cluster Member Lost)

Quality Attribute 9 – Use of Persistence Tier Resources are Reduced

This was observed during testing and in comparison of the old and new architectures. With the IMDG solution configured as a write behind cache, writes to the database were batched up; instead of writes trickling in and requiring a transaction per modification, writes were pushed to the database as a batch and committed as a batch, making more efficient use of the database tier computing resources.

Test Disposition

The Pass/Fail criteria outlined in the test plan is stated as follows:

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.*

Based on the above criteria and the test results summarized in this document, it is asserted that a test disposition of pass has been established.