CUBRID 9.2 QA Completion Report

This document is the verification report of CUBRID 9.2 in terms of functionality, performance and stability.

Table of Contents

CUBRID 9.2 QA Completion Report 1

1. Test Overview 4

- 1.1 Test Objectives 5
- 1.2 Test Environment 5
  - 1.2.1 TEST PROCEDURES 5
  - 1.2.2 HARDWARE TEST ENVIRONMENT 8
- 1.3 Test Category 9

2. Test Results 10

- 2.1 Functionality Test Results 11
  - 2.1.1 BASIC QUERY TESTS 11
  - 2.1.2 BASIC UTILITY AND OTHER SCENARIO TESTS 11
  - 2.1.3 HA FEATURE TESTS 11
  - 2.1.4 HA REPLICATION TESTS 12
  - 2.1.5 CCI INTERFACE TESTS 12
  - 2.1.6 JDBC INTERFACE TESTS 12
  - 2.1.7 CAS4MYSQL/ORACLE TESTS 13
- 2.2 Performance Test Results 14
  - 2.2.1 CUBRID BASIC PERFORMANCE TEST 14
  - 2.2.2 YCSB PERFORMANCE TEST 18
  - 2.2.3 SYSBENCH PERFORMANCE TEST 23
  - 2.2.4 NBD BENCHMARK PERFORMANCE TEST 24
  - 2.2.5 TPC-C PERFORMANCE TEST 28
  - 2.2.6 DATA REPLICATION TEST ON HA 28
- 2.3 Stability Test Results 29
- 2.4 Compatibility Test Results 30

2.5 Installation Test Results ____ 31
2.6 Other Test Results ____ 32
2.7 Quality Index ____ 33

3. Conclusions ____ 34

Appendix ____ 36

I. Functionality Test Scenarios ____ 37
II. Performance Test Scenarios ____ 42
III. Stability Test Scenarios ____ 52
IV. Scenario-based Code Coverage Results ____ 54
V. JDBC Code Coverage Results ____ 54

1.Test Overview

1.1 Test Objectives

The objectives of this test are to perform functionality, performance and stability tests for the final release candidate build of CUBRID 9.2 (hereinafter referred to as 9.2), which is under development for release in September 2013, in order to determine the product release afterwards based on the test results. To guarantee the stability of CUBRID testing, we have used the test environments configured as below, which could also be adopted as a reference for the further testing. Based on the function results and the comparisons between the performance test results of CUBRID 9.2 and those of CUBRID 9.1 (hereinafter referred to as 9.1), we could verify whether the performance of 9.2 has been improved or not.

- CentOS 5.6 (32/64-bit) or compatible
- CentOS 5.3 (32/64-bit) or compatible
- CentOS 4.7 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 9.2.0.0155 (Linux 64-bit/32-bit, Windows 64-bit/32-bit)

1.2 Test Environment

1.2.1 Test Procedures

Test procedures we have used to verify the CUBRID product are shown below. The actual test sequence used may be different from the one described here. To verify the product stability, functionality, performance and other tests were performed for 4 types of builds as shown in the figure below. The details of each test suite are described in the appendix of this report.

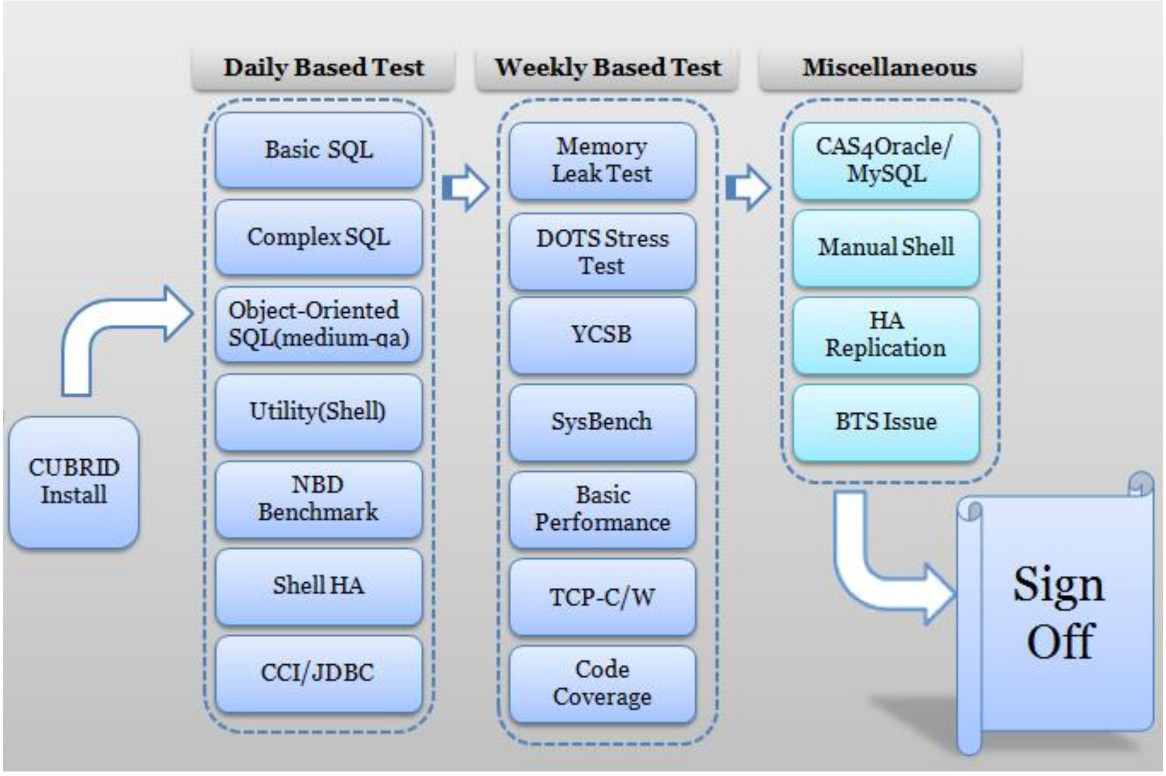


Figure 1. CUBRID Test Procedure

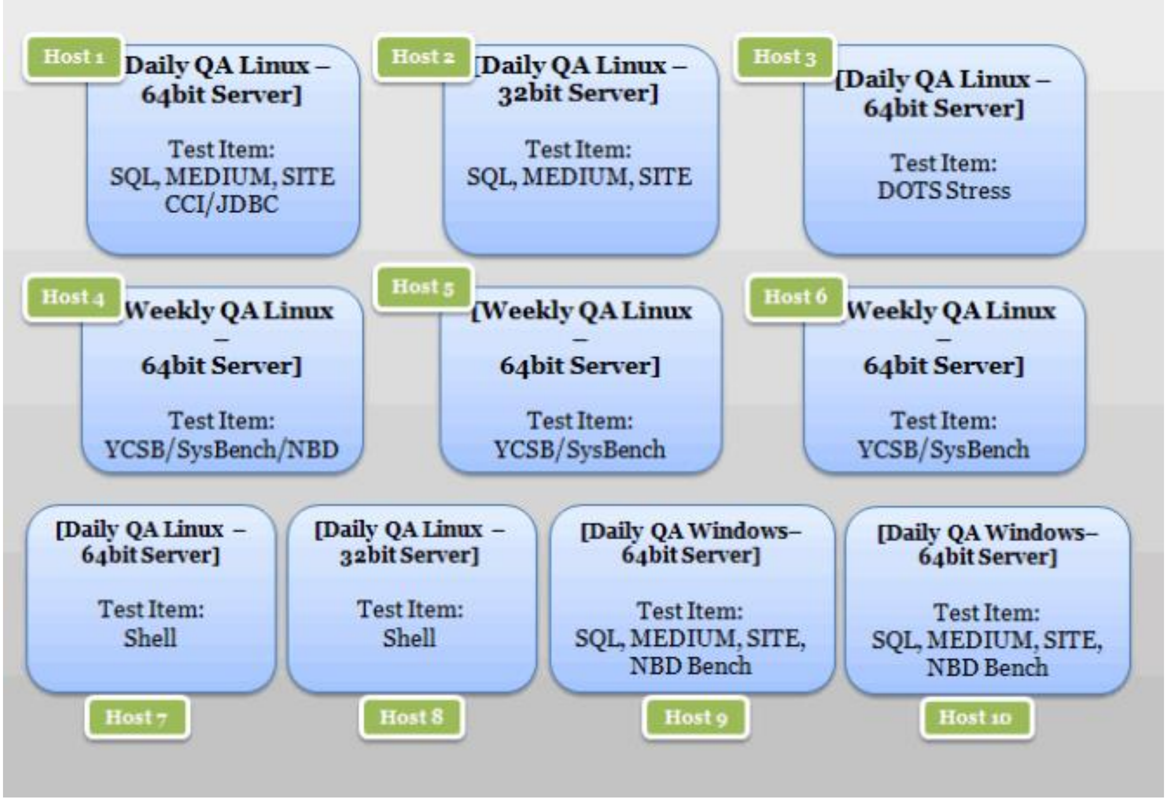

Host 1
Daily QA Linux – 64bit Server]
Test Item: SQL, MEDIUM, SITE CCI/JDBC
Host 2
[Daily QA Linux – 32bit Server]
Test Item: SQL, MEDIUM, SITE
Host 3
[Daily QA Linux – 64bit Server]
Test Item: DOTS Stress
Host 4
Weekly QA Linux – 64bit Server]
Test Item: YCSB/SysBench/NBD
Host 5
[Weekly QA Linux – 64bit Server]
Test Item: YCSB/SysBench
Host 6
Weekly QA Linux – 64bit Server]
Test Item: YCSB/SysBench
[Daily QA Linux – 64bit Server]
Test Item: Shell
Host 7
[Daily QA Linux – 32bit Server]
Test Item: Shell
Host 8
[Daily QA Windows– 64bit Server]
Test Item: SQL, MEDIUM, SITE, NBD Bench
Host 9
[Daily QA Windows– 64bit Server]
Test Item: SQL, MEDIUM, SITE, NBD Bench
Host 10

Figure 2. System Diagram for Basic Test

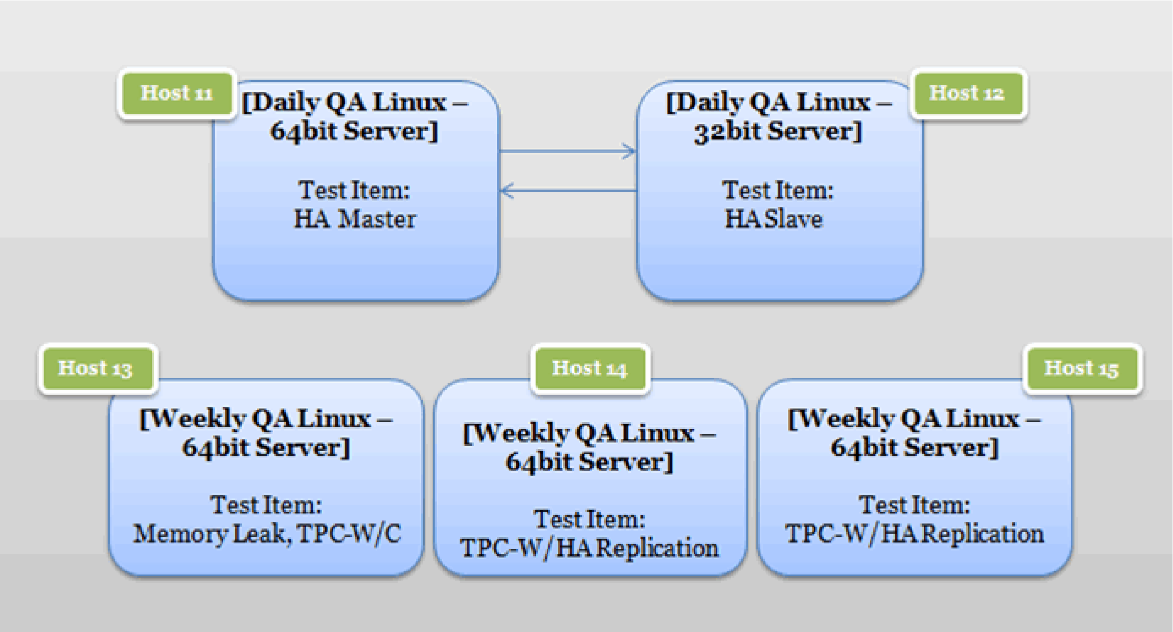

Host 11
[Daily QA Linux – 64bit Server]
Test Item: HA Master
Host 12
[Daily QA Linux – 32bit Server]
Test Item: HA Slave
Host 13
[Weekly QA Linux – 64bit Server]
Test Item: Memory Leak, TPC-W/C
Host 14
[Weekly QA Linux – 64bit Server]
Test Item: TPC-W/HA Replication
Host 15
[Weekly QA Linux – 64bit Server]
Test Item: TPC-W/HA Replication

Figure 3. System Diagram for HA Test

1.2.2 Hardware Test Environment

Servers for the CUBRID test and their usage are listed in the table below.

Name	OS	CPU	MEMORY	DISK
Host 1	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 GB	SAS 600G * 3 (Raid5)
Host 2	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 GB	SAS 600G * 3 (Raid5)
Host 3	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 GB	SAS 600G * 3 (Raid5)
Host 4	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	32 GB	SAS 600G * 3 (Raid5)
Host 5	Cent OS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	32 GB	SAS 600G * 3 (Raid5)
Host 6	Cent OS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	32 GB	SAS 600G * 3 (Raid5)
Host 7	Cent OS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 GB	SAS 600G * 3 (Raid5)
Host 8	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	32 GB	SAS 600G * 3 (Raid5)
Host 9	Windows 2003 (64-bit)	Xeon 2.33 GHz (quad cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 10	Windows 2003 (32-bit)	Xeon 2.0 GHz (quad cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 11	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 12	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 13	Cent OS 6.3 (64-bit)	Xeon 2.27 GHz (12 cores) * 1	48 GB	SAS 300G * 6 (Raid1+0)
Host 14	Cent OS 6.3 (64-bit)	Xeon 2.27 GHz (12 cores) * 1	48 GB	SAS 300G * 6 (Raid1+0)
Host 15	Cent OS 6.3 (64-bit)	Xeon 2.27 GHz (12 cores) * 1	48 GB	SAS 300G * 6 (Raid1+0)

1.3 Test Category

The following tests have been performed to determine whether 9.2 meets the criteria of release. The details of each test are described in the appendix of this report.

- Functionality tests
  - SQL query test
  - MEDIUM query test
  - SITE query test
  - Utility (Shell) test
  - HA Feature test
  - HA Replication test
  - CCI Interface test
  - JDBC Interface test
  - CAS4MySQL/Oracle
- Performance tests
  - Basic Performance Test
  - YCSB Benchmark
  - SysBench
  - NBD Benchmark
  - TPC-C
  - Data Replication Test on HA
- Stability tests
  - DOTS stress test
  - TPC-W on HA test
- Compatibility tests
  - JDBC compatibility test
  - CCI compatibility test
- Installation tests
- Other tests
  - Test for checking 9.2 functionalities/bug fixes
  - Memory check (SQL/MEDIUM) by Valgrind

2.Test Results

2.1 Functionality Test Results

2.1.1 Basic Query Tests

This test has been performed to verify the basic DBMS functionalities by using SQL statements. SQL statements stored in 14,557 files have been executed to verify DBMS conformity. We have executed all the stored SQL statements in both JDBC-based and CCI-based applications on the release build and the debug build, and then compared the results with the stored reference files for verification.

Table 1. Result of Basic Query Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
SQL query test (JDBC and CCI)	12,374	12,374	100%
MEDIUM query test	970	970	100%
SITE query test	1,213	1,213	100%

2.1.2 Basic Utility and Other Scenario Tests

This test has been performed to verify the basic DBMS functionalities by using shell scripts. In particular, this test was also performed to verify CUBRID utilities that could not be tested by SQL statements. 1,754 scenarios tested by shell scripts have been executed to verify DBMS conformity.

Table 2. Result of Basic Utility and Other Scenario Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Utility	226	226	100%
Bug regression	955	955	100%
Environment variable	7	7	100%
Other	566	566	100%

2.1.3 HA Feature Tests

179 scenarios tested by shell scripts have been executed to verify HA features and the regressions.

Table 3. Result of HA Feature Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Bug regression	171	171	100%
Fault test	8	8	100%

2.1.4 HA Replication Tests

HA Replication Test is a new QA tool which runs SQL test cases on HA Master, and verifies the data consistency between Master and Slave. 12,423 scenarios based on SQL files have been executed to verify the data consistency between Master and Slave.

Table 4. Result of HA Replication Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Test Cases migrated from SQL suite	12,360	12,360	100%
Bug regression	63	63	100%

2.1.5 CCI Interface Tests

CCI Interface Test aims to verify if all the CCI APIs of CUBRID work well as described in the CUBRID manual. 254 scenarios based on shell scripts have been executed on the release build and the debug build to verify all the CCI APIs basic features and the BTS issue regressions.

Table 5. Result of CCI Interface Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Basic features	202	202	100%
Bug regression	52	43	100%

2.1.6 JDBC Interface Tests

1,530 scenarios tested by java jUnit have been executed to verify all the JDBC API features and the BTS issue regressions.

Table 6. Result of JDBC Interface Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Features test	1,530	1,530	100%

2.1.7 CAS4MySQL/Oracle Tests

108 scenarios tested by shell scripts have been executed to verify the features of CAS4MySQL and CAS4Oracle respectively.

Table 7. Result of CAS4MySQL/Oracle Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
CAS4MySQL	54	54	100%
CAS4Oracle	54	54	100%

2.2 Performance Test Results

2.2.1 CUBRID Basic Performance Test

This test has been performed to check the performance of the CUBRID DBMS basic operations, which are select, insert, update and delete. For more information about the test scenarios, see the appendix II. All the default configuration values are adopted except SQL_LOG=OFF in cubrid_broker.conf. As shown in the table below, we can see that the performances of INSERT, UPDATE, SELECT and DELETE on 4 platforms(Linux 32, 64bit and Windows 32, 64bit) are same as 9.1.

A. Linux: Performance Comparison between 9.1 (64-bit) and 9.2 (64-bit)

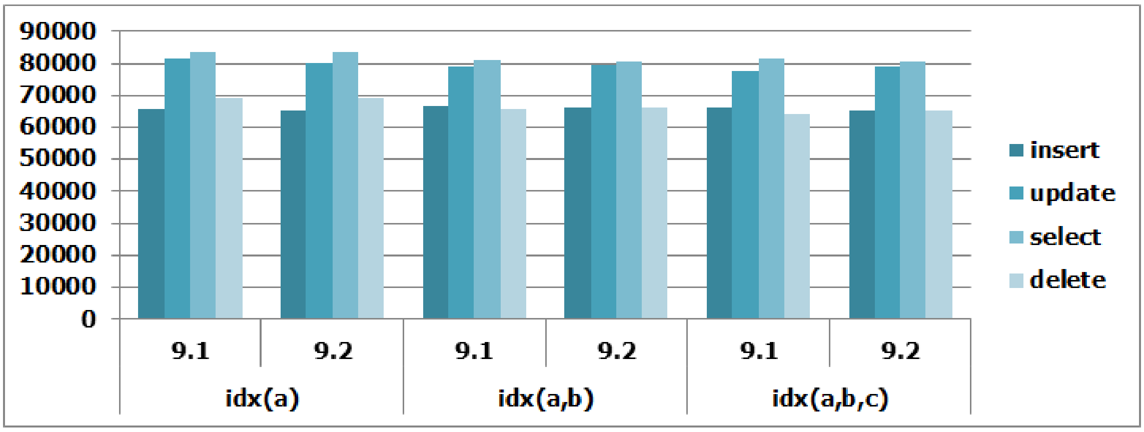


Figure 4. Performance Comparison between 9.1 and 9.2 (Linux 64-bit)

Table 8. Performance Comparison between 9.1 and 9.2 (Linux 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	9.1	9.2	Ratio	9.1	9.2	Ratio	9.1	9.2	Ratio
Insert	65,827	65,048	99%	66,454	66,263	100%	66,422	65,421	98%
Update	81,302	80,014	98%	79,095	79,626	101%	77,712	78,886	102%
Select	83,515	83,629	100%	80,969	80,370	99%	81,498	80,502	99%
Delete	69,072	69,078	100%	65,521	66,383	101%	64,150	65,411	102%
Total	299,716	297,769	99%	292,039	292,642	100%	289,782	290,220	100%

(Unit: TPS)

B. Linux: Performance Comparison between 9.1 (32-bit) and 9.2 (32-bit)

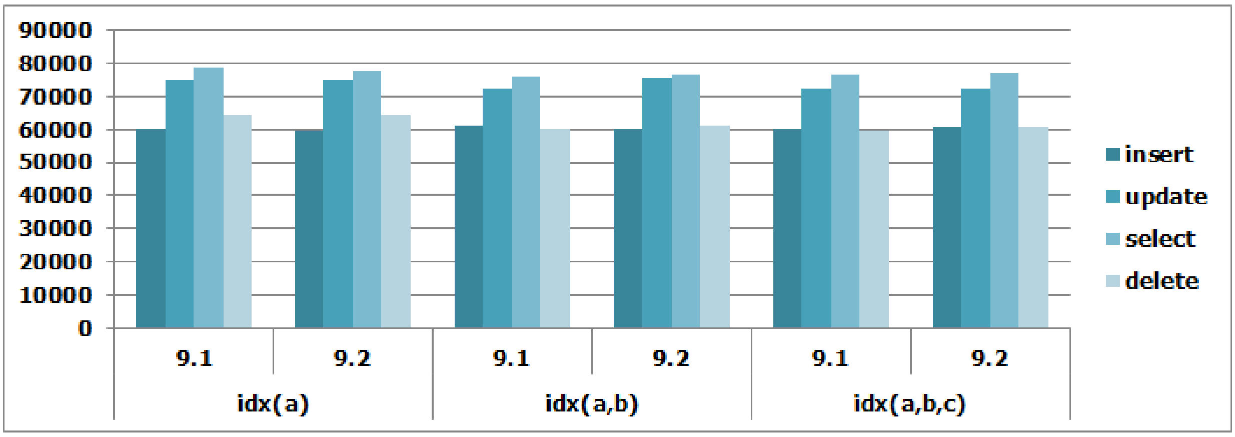


Figure 5. Performance Comparison between 9.1 and 9.2 (Linux 32-bit)

Table 9. Performance Comparison between 9.1 and 9.2 (Linux 32-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	9.1	9.2	Ratio	9.1	9.2	Ratio	9.1	9.2	Ratio
Insert	60,263	59,422	99%	61,200	60,373	99%	60,099	60,594	101%
Update	75,208	75,055	100%	72,224	75,467	104%	72,105	72,496	101%
Select	78,496	77,809	99%	75,993	76,695	101%	76,582	77,127	101%
Delete	64,328	64,509	100%	60,118	61,241	102%	59,640	60,838	102%
Total	278,295	276,795	99%	269,535	273,776	102%	268,426	271,055	101%

(Unit: TPS)

C. Windows: Performance Comparison between 9.1 (64-bit) and 9.2 (64-bit)

According to the test result below, we can see that the performance of some operations has a little drop, that is normal fluctuation.

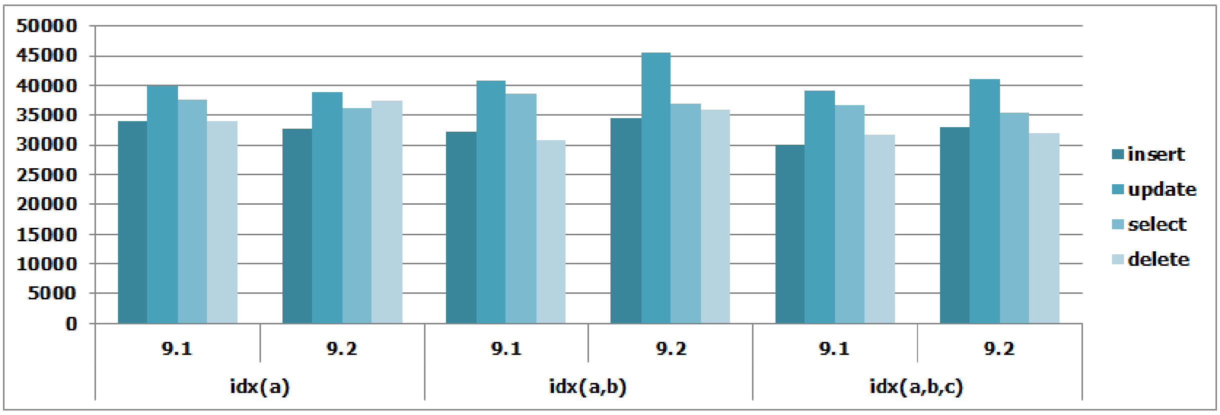


Figure 6. Performance Comparison between 9.1 and 9.2 (Windows 64-bit)

Table 10. Performance Comparison between 9.1 and 9.2 (Windows 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	9.1	**9.2**	**Ratio**	**9.1**	**9.2**	**Ratio**	**9.1**	**9.2**	**Ratio**
Insert	34,047	32,715	96%	32,322	34,409	106%	30,073	33,079	110%
Update	39,893	38,951	98%	40,723	45,490	112%	39,079	41,051	105%
Select	37,670	36,256	96%	38,594	36,897	96%	36,661	35,363	96%
Delete	33,910	37,421	110%	30,703	35,813	117%	31,685	31,963	101%
Total	145,520	145,343	100%	142,342	152,609	107%	137,498	141,456	103%

(Unit: TPS)

D. Windows: Performance Comparison between 9.1 (32-bit) and 9.2 (32-bit)

According to the test result below , we can see that there is no performance change based on Windows 32-bit OS testing.

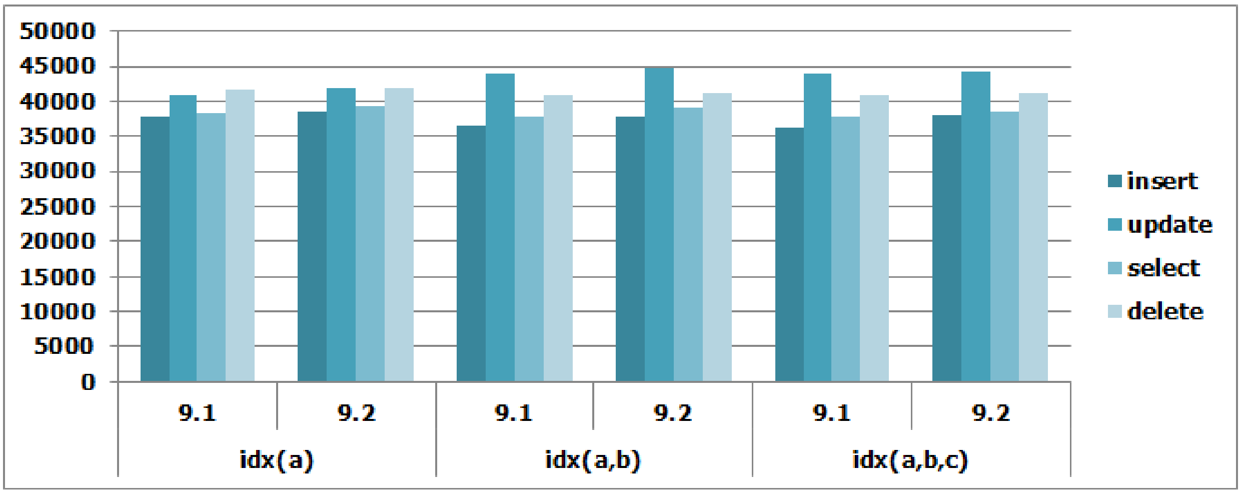


Figure 7. Performance Comparison between 9.1 and 9.2 (Windows 32-bit)

Table 11. Performance Comparison between 9.1 and 9.2 (Windows 32-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	9.1	**9.2**	**Ratio**	**9.1**	**9.2**	**Ratio**	**9.1**	**9.2**	**Ratio**
Insert	37,857	38,529	102%	36,609	37,896	104%	36,302	37,987	105%
Update	41,008	41,909	102%	44,014	44,751	102%	43,872	44,362	101%
Select	38,406	39,445	103%	37,902	38,997	103%	37,752	38,683	102%
Delete	41,721	41,888	100%	40,849	41,247	101%	40,999	41,130	100%
Total	158,992	161,771	102%	159,374	162,891	102%	158,925	162,162	102%

(Unit: TPS)

2.2.2 YCSB Performance Test

As a framework for benchmarking system, YCSB is popular and widely used in the world nowadays (see also https://github.com/brianfrankcooper/YCSB/wiki). This test has been performed to verify CUBRID performance of not only basic operations but also compositive operations, which are insert, select, scan, update and the mix of them. For more information about the test scenarios, see appendix II. As shown in the results below, the performance on SELECT has significant improvement comparing with 9.1, and all other operations are same as 9.1.

A. Master Server Configuration: Performance Comparison between 9.2 (64-bit) and 9.1 (64-bit)

Table 12. Result of YCSB Benchmark (Master Server)

	Throughput(OPS)			Average Latency(ms)		95thPercentile Latency(ms)	
Operations	**9.1**	**9.2**	**Ratio**	**9.1**	**9.2**	**9.1**	**9.2**
Insert	13,945	13,577	97%	20	21	37	34
Select	26,398	32,593	123%	10	9	29	33
Scan	4,235	4,297	101%	63	63	248	247
Update	12,399	12,262	99%	23	24	18	18
Mix	12,929	12,585	97%	N/A	N/A	N/A	N/A

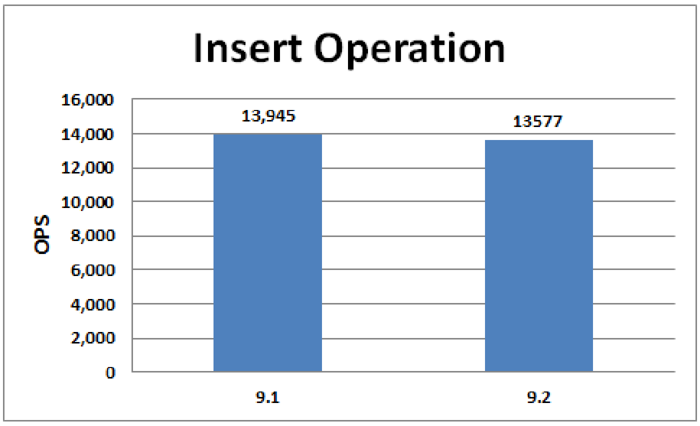


Figure 8. Result of Insert Operation of YCSB Benchmark (Master Server)

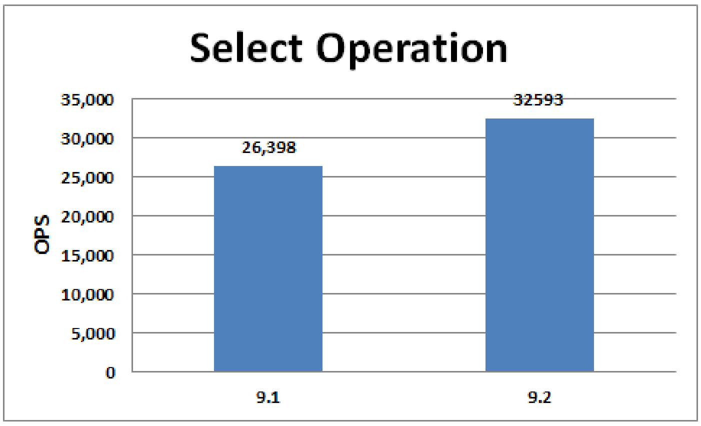


Figure 9. Result of Select Operation of YCSB Benchmark (Master Server)

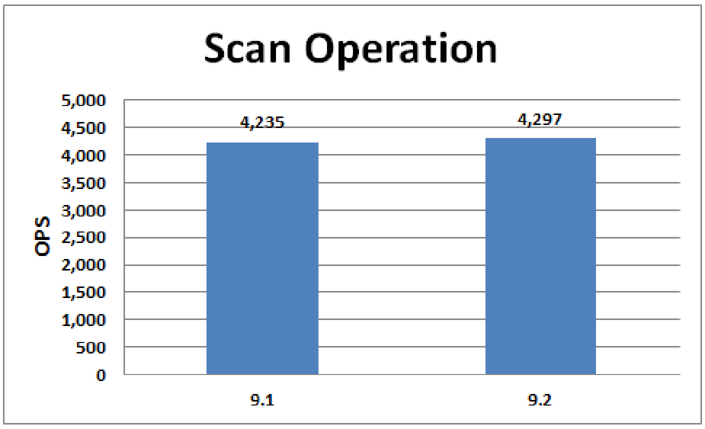


Figure 10. Result of Scan Operation of YCSB Benchmark (Master Server)

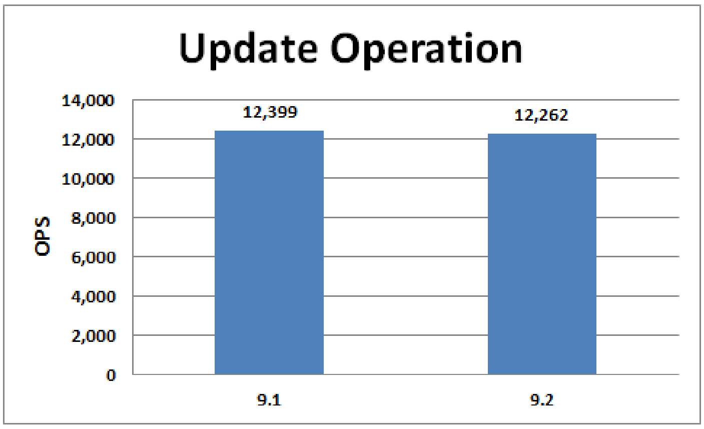


Figure 11. Result of Update Operation of YCSB Benchmark (Master Server)

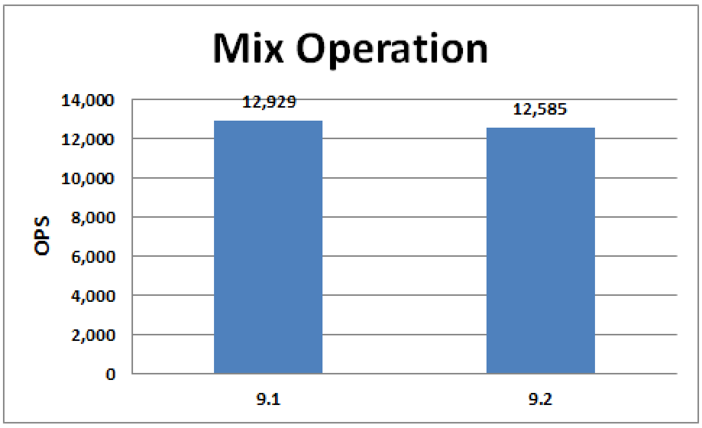


Figure 12. Result of Mixed of YCSB Benchmark (Master Server)

B. Slave Server Configuration: Performance Comparison between 9.1 (64-bit) and 9.2 (64-bit)

Table 13. Result of YCSB Benchmark (Slave Server)

	Throughput(OPS)			Average Latency(ms)		95th Percentile Latency(ms)	
Operations	**9.1**	**9.2**	**Ratio**	**9.1**	**9.2**	**9.1**	**9.2**
Insert	14,496	14,018	97%	19	21	38	42
Select	28,224	32,783	116%	9	9	31	34
Scan	4,307	4,244	99%	64	63	248	247
Update	12,971	12,829	99%	22	23	15	16
Mix	13,334	12,732	95%	N/A	N/A	N/A	N/A

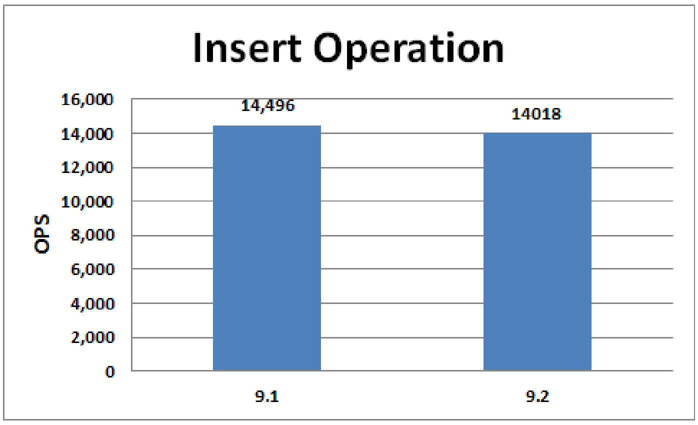


Figure 13. Result of Insert Operation of YCSB Benchmark (Slave Server)

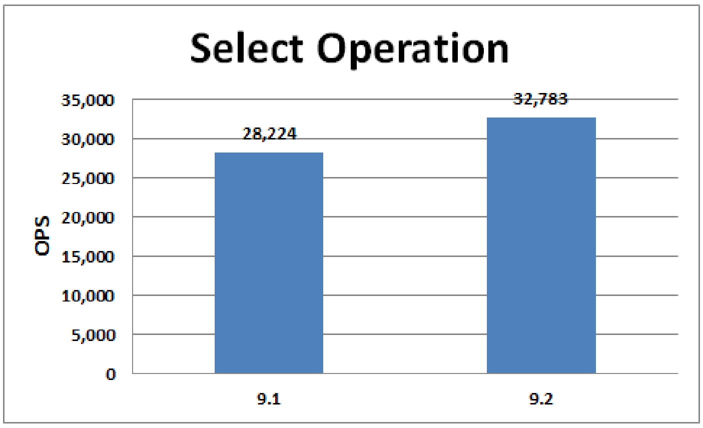


Figure 14. Result of Select Operation of YCSB Benchmark (Slave Server)

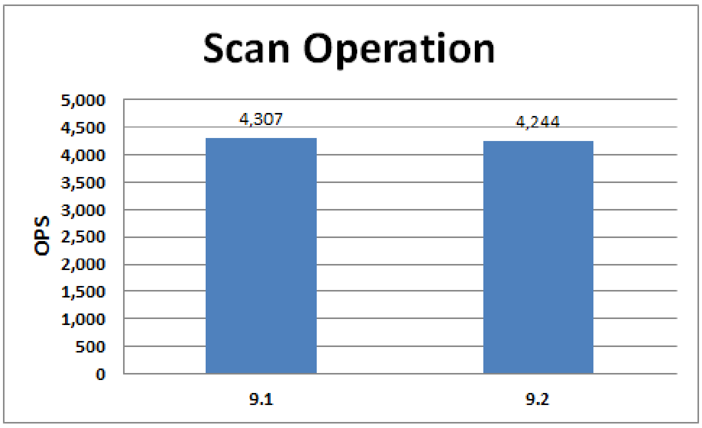


Figure 15. Result of Scan Operation of YCSB Benchmark (Slave Server)

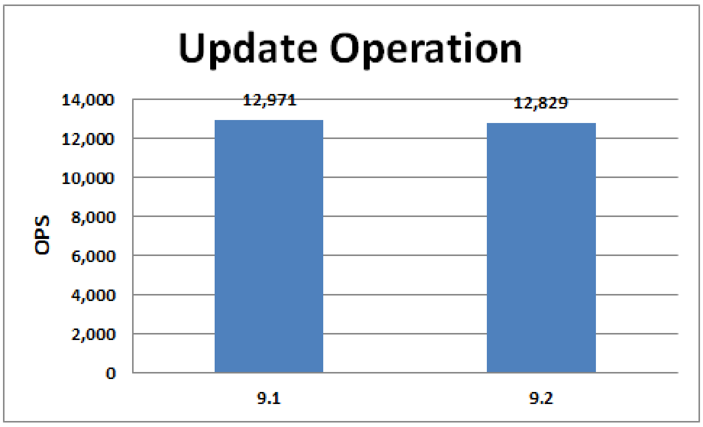


Figure 16. Result of Update Operation of YCSB Benchmark (Slave Server)

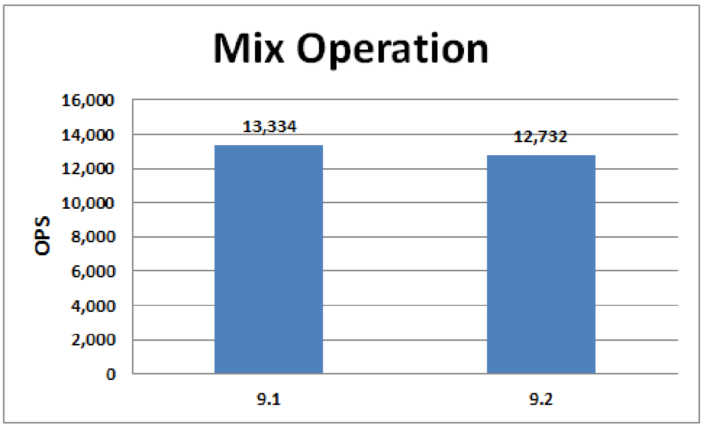


Figure 17. Result of Mixed of YCSB Benchmark (Slave Server)

2.2.3 SysBench Performance Test

SysBench is a modular, cross-platform and multi-threaded benchmark tool for evaluating OS parameters that are important for a system running a database under intensive load (see also http://sysbench.sourceforge.net/). SysBench runs a specified number of threads which could execute requests in parallel. The actual workload produced by requests depends on the specified test mode. You can limit either the total number of requests or the total time for the benchmark, or both. Available test modes are implemented by compiled-in modules, and SysBench was designed to make adding new test modes an easy task. Each test mode may have additional (or workload-specific) options. For more information about the test scenarios, see appendix II.
As shown in the results below, the performance of SysBench on 9.2 has no changes comparing with 9.1, a little drop should belong to is the normal fluctuation.

A. SysBench performance comparison between 9.1 (64-bit) and 9.2 (64-bit)

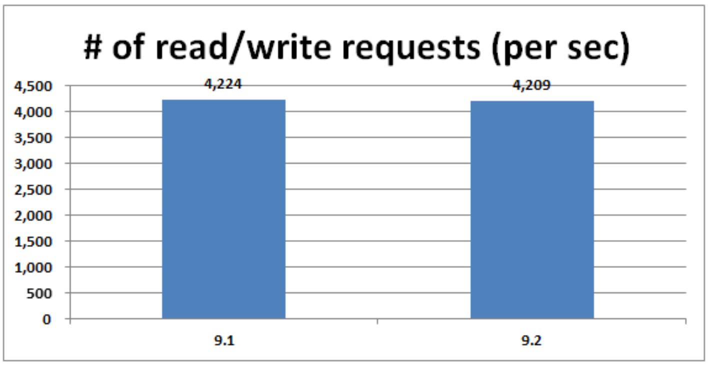


Figure 18. The number of read/write requests per second of SysBench benchmark

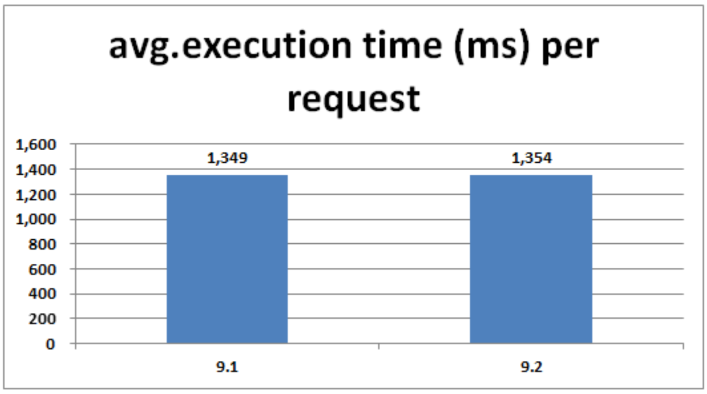


Figure 19. The average execution time per request of SysBench benchmark

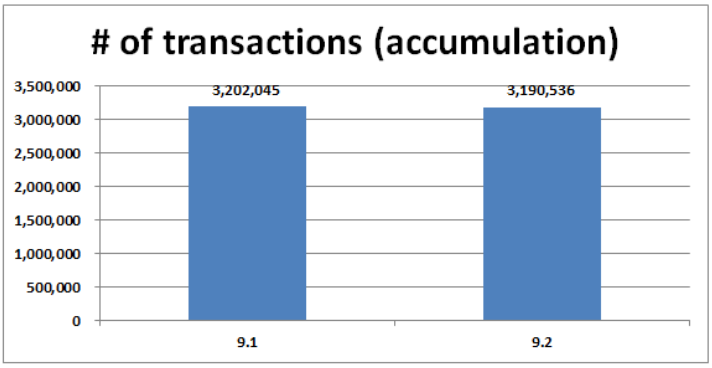


Figure 20. The accumulated number of transactions of SysBench benchmark

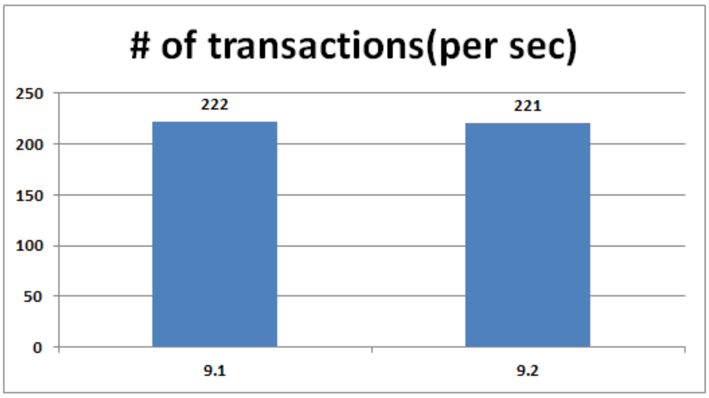


Figure 21. The number of transactions per second of SysBench benchmark

2.2.4 NBD Benchmark Performance Test

This test has been performed to verify the CUBRID performance with the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. The scalability of the test DB was Level 1. The number of Page Views of 9.2 has no significant changes comparing with of 9.1.

A. NBD performance comparison between 9.1 (64-bit) and 9.2 (64-bit)

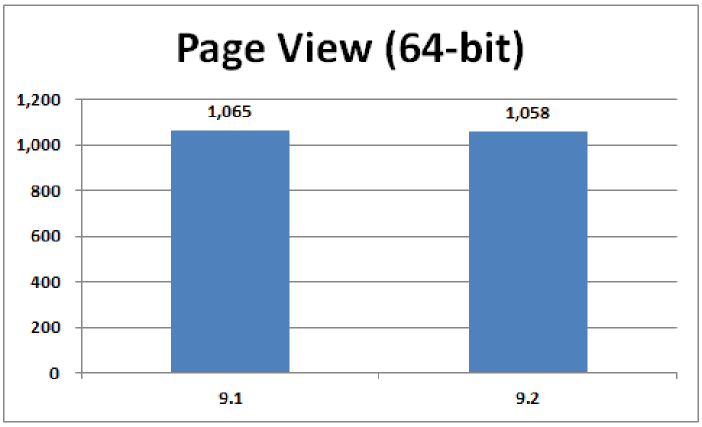


Figure 22. NBD performance comparison (64-bit)

B. NBD performance comparison between 9.1 (32-bit) and 9.2 (32-bit)

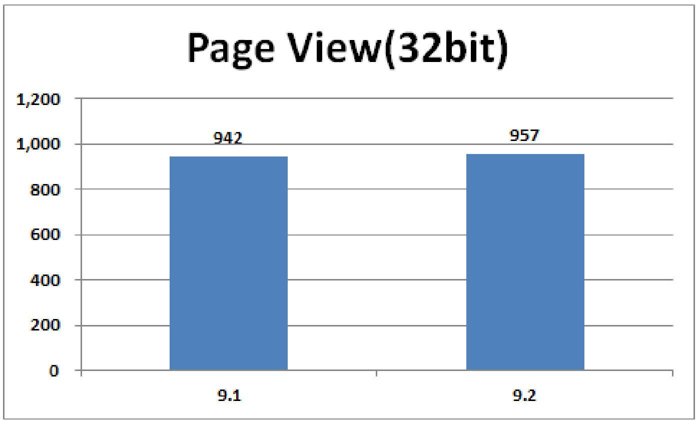


Figure 23. NBD performance comparison (32-bit)

The following graphs represent the usage rate of each resource while processing the NBD benchmark test on Linux 64-bit.

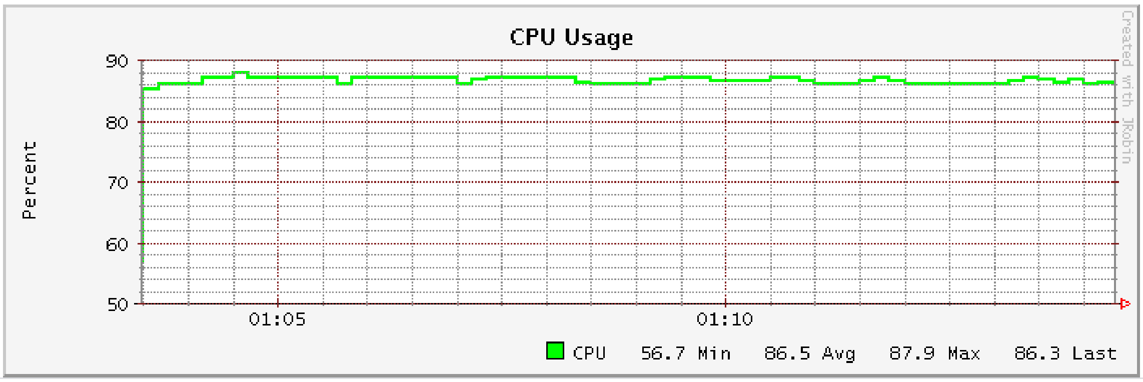


Figure 24. CPU Usage for NBD Benchmark

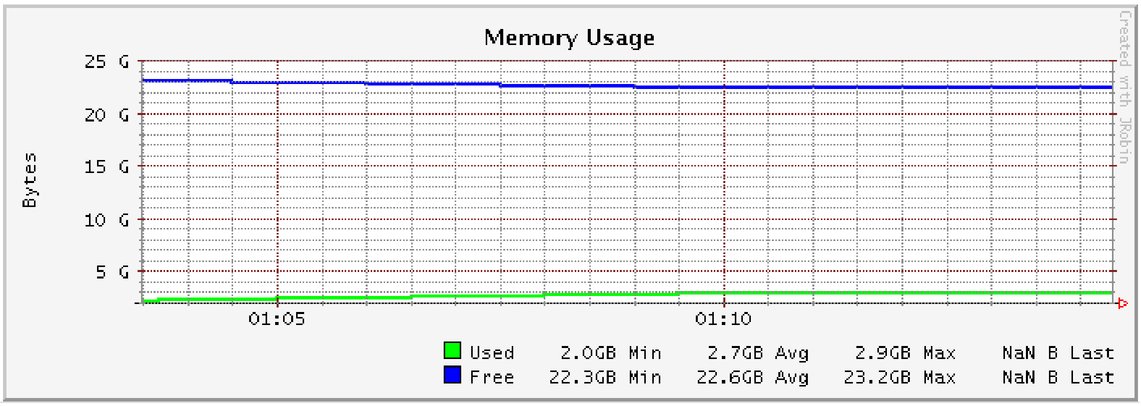
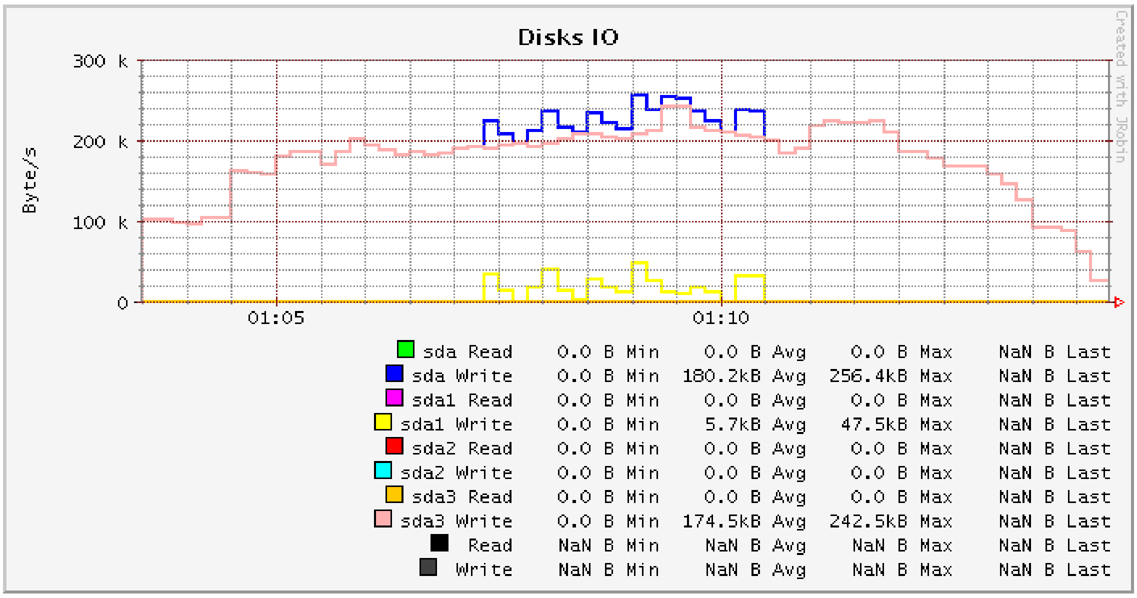


Figure 25. Memory Usage for NBD Benchmark



🟩 sda Read	0.0 B Min	0.0 B Avg	0.0 B Max	NaN B Last
🟦 sda Write	0.0 B Min	180.2kB Avg	256.4kB Max	NaN B Last
🟪 sda1 Read	0.0 B Min	0.0 B Avg	0.0 B Max	NaN B Last
🟨 sda1 Write	0.0 B Min	5.7kB Avg	47.5kB Max	NaN B Last
🟥 sda2 Read	0.0 B Min	0.0 B Avg	0.0 B Max	NaN B Last
🟦 sda2 Write	0.0 B Min	0.0 B Avg	0.0 B Max	NaN B Last
🟧 sda3 Read	0.0 B Min	0.0 B Avg	0.0 B Max	NaN B Last
🟥 sda3 Write	0.0 B Min	174.5kB Avg	242.5kB Max	NaN B Last
⬛ Read	NaN B Min	NaN B Avg	NaN B Max	NaN B Last
⬛ Write	NaN B Min	NaN B Avg	NaN B Max	NaN B Last

Figure 26. Disks IO status for NBD Benchmark

2.2.5 TPC-C Performance Test

TPC Benchmark C, approved in July of 1992, is an on-line transaction processing (OLTP) benchmark. TPC-C (see also http://www.tpc.org/tpcc/) is more complex than the previous OLTP benchmarks such as TPC-A because of its multiple transaction types, complicated database and overall execution structure. TPC-C involves a mix of five concurrent transactions of different types and complexity either executed on-line or queued for deferred execution. The database is comprised of nine types of tables with a wide range of record and population sizes. TPC-C is measured in transactions per minute (tpmC).
As shown in the results below, the performance of TPC-C on 9.2 has slightly improved on tpmC.

A. TPC-C performance comparison between 9.1 (64-bit) and 9.2 (64-bit)

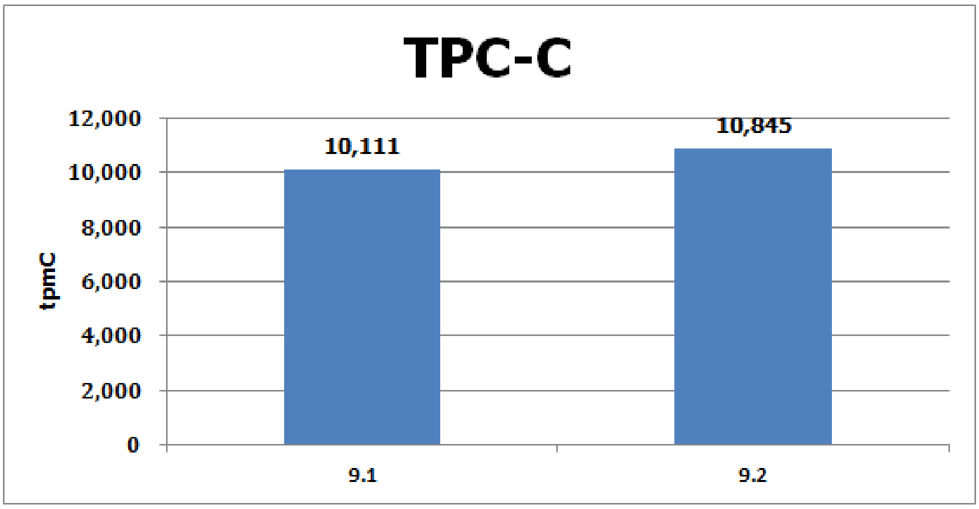


Figure 27. tpmC comparison of TPC-C benchmark

2.2.6 Data Replication Test on HA

This test has been performed to evaluate the performance of data replication under HA environment by using YCSB to execute INSERT, MIX operations on Master server with the related configurations, and check the delay time of data replication on Slave by CUBRID SQL statement. For more details, please refer to appendix II. As shown in the table below, the performance of data synchronization on 9.2 is basic same with 9.1, but 9.2 has been significantly improved comparing with R4.1.

Table 14. Data replication performance comparison

Version	Delay Time (sec.)
R4.1	4,397
9.1	267
9.2	256

2.3 Stability Test Results

DOTS, a sub-project of an open source project "Linux Test Project," is an open source test tool for the DBMS testing. For more information about DOTS, see appendix III. As shown in the test results below, the system operated stably without any abnormalities during 54 hours. You can ignore the failures because they are unique violations due to the modification of duplicated data.

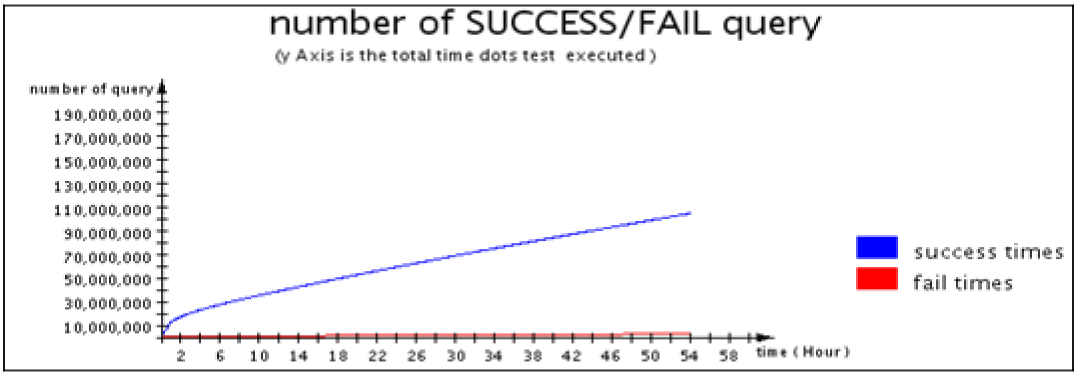


Figure 28. The number of SUCCESS/FAIL Queries of DOTS Test

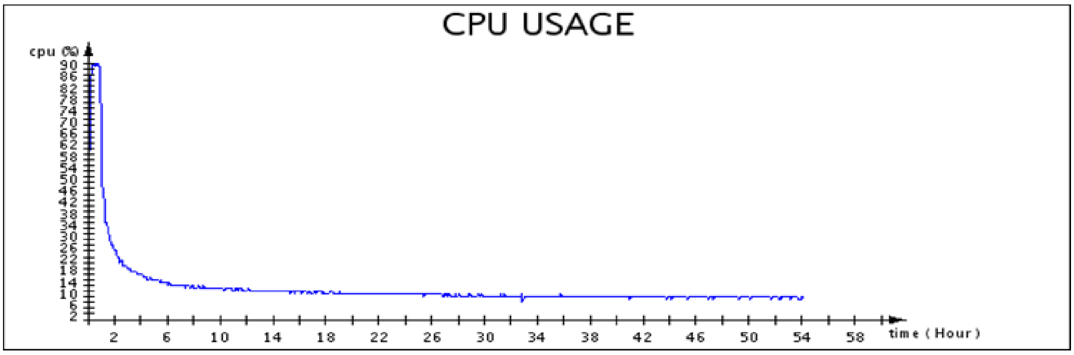


Figure 29. CPU Usage of DOTS Test

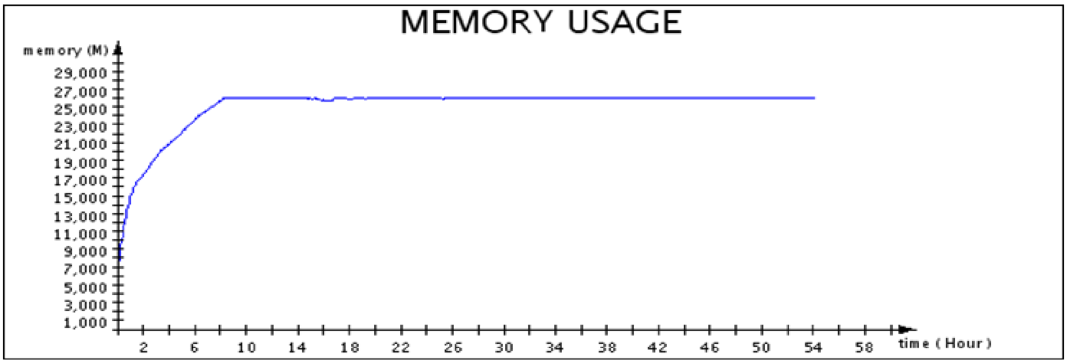


Figure 30. Memory Usage of DOTS Test

2.4 Compatibility Test Results

This test has been performed to verify the JDBC and CCI compatibility among R4.1, R4.3, R4.4, 9.1 and 9.2. SQL, MEDIUM and Site Tests were executed to verify JDBC compatibility. Shell test cases for CCI were executed to verify CCI compatibility, all the test results have been passed.

Table 15. Result of JDBC Compatibility Tests when 9.2 as Driver

Test Component	Scenario	8.4.1 Server	8.4.3 Server	8.4.4 Server	9.1.0 Server
9.2.0 Driver	SQL	8,704	8,740	8,763	12,100
	Medium	970	970	970	970
	Site	1,213	1,213	1,213	1,213
	Shell_cci	176	190	222	251

Table 16. Result of JDBC Compatibility Tests when 9.2 as Server

Test Component	Scenario	8.4.1 Driver	8.4.3 Driver	8.4.4 Driver	9.1.0 Driver
9.2.0 Server	SQL	12,193	12,191	12,188	12,343
	Medium	970	970	970	970
	Site	1,213	1,213	1,213	1,213
	Shell_cci	193	200	169	258

2.5 Installation Test Results

Installation test has been performed based on the below basic scenarios:

- Install and uninstall package
- Start and stop service/server/broker and manager
- Create and delete database
- Execute a simple query in csql

Table 17. Result of Installation Test

Package Type	Test OS	Result
RPM/SH/TAR.GZ	Linux CentOS on 32-bit and 64-bit	PASS
SH	Ubuntu 11 on 64-bit SULinux on 64-bit Fedora 15 64-bit	PASS
EXE/ZIP	Windows Server 2008/2003 on 32-bit and 64-bit	PASS
EXE/ZIP	Windows 7 on 32-bit and 64-bit Windows XP on 32-bit	PASS

2.6 Other Test Results

The entire bug and issue fixes for 9.2 have been confirmed.

2.7 Quality Index

The standard quality index of 9.2 is listed below.

Table 18. Quality Index of 9.2

Quality Index Name	Project Quality Standard	Approved Quality Index during Implementation	Measurement Target	
Coding Standards Compliance Rate	100%	100%	Number of coding conventions observed in a project	56
			Number of coding conventions applied to each team	56
Code Review Execution Rate	100%	100%	Number of source code lines for which code review is performed.	1,372,432 LOC
			Total number of source code lines in the changed files	1,372,432 LOC
QA Scenario Code Coverage	76%	75.1%	Number of tested statements	220,851
			Total number of statements	294,235
Fault Density Detected by Static Analysis	4 /KLOC	4.73 /KLOC	Number of faults detected by static analysis (Level 1)	329
			Number of faults detected by static analysis (Level 2)	75
			Number of faults detected by static analysis (Level 3)	850
			Number of faults detected by static analysis (Level 4)	0
			Total number of source code lines	978,969 LOC
Cyclomatic Code Complexity	3.3%	3.0%	Number of modules whose complexity is over 30	694
			Total number of modules in a project	23,631
	12%	15.5%	Number of modules whose complexity is over 10	3,679
			Total number of modules in a project	23,631

3.Conclusions

As described in Chapter 1 and 2, all the test cases for functions have been regressed, and the scenarios for performance, stability, compatibility, installation and other tests have also been successfully executed before the release of 9.2. The tests have been performed on Linux 32-bit, Linux 64-bit, Windows 32-bit and Windows 64-bit environments. The related defects have been logged into BTS.

Based on the results obtained from the basic performance test, we can see that the performances of INSERT, SELECT,UPDATE and DELETE on Linux 64-bit are same as 9.1.

For YCSB, we can see that the performance of SELECT has significant improvement comparing with 9.1, approximately 20% increase. But other operations such as INSERT, UPDATE and DELETE on Linux 64-bit are same as 9.1.

For NBD and SysBench, there are no significant changes for performance.

For TPC-C, there are no significant changes for performance of tpmC.

For stability test with DOTS, according to the graphs of CPU usage, Memory usage and the number of SUCCESS/FAIL queries of Dots , it looks quite stable even after 54 hours of execution, no notable issues have been found.

According to the result of data replication test on HA mode, the performance of data synchronization has significantly improved from R4.1, and it is same as 9.1 release.

From the result of compatibility test, we can reach the conclusion that JDBC and CCI on R4.1, R4.3, R4.4 and 9.1 have compatibility with 9.2 server, and JDBC and CCI on 9.2 also have compatibility with R4.1, R4.3, R4.4 and 9.1 server.

As a conclusion, CUBRID 9.2 meets the criteria of release.

Appendix

I. Functionality Test Scenarios

This test has been performed to verify the basic DBMS functionalities by using SQL statements. SQL statements stored in the files have been executed to verify DBMS conformity. We have executed all the stored SQL statements in JDBC-based and CCI-based applications, and compared the results to the stored reference files for verification. The scenario files included in the basic functionality test are stored in the SQL and MEDIUM directories of the CUBRID QA tool.

- SQL Query Test

Total: 12,374		
Case Name	Path	**Description**
object	sql/_01_object	Performs functionality tests of objects supported by CUBRID, and has the largest number of scenarios (3,332 scenarios).
user_authorization	sql/_02_user_authorization	Performs functionality tests of user and authorization management.
object_oriented	sql/_03_object_oriented	Performs tests for the object-oriented concept. CUBRID is an object-relational database management system (DBMS).
operator_function	sql/_04_operator_function	Performs functionality tests of basic functions and operators supported by CUBRID.
manipulation	sql/_06_manipulation	Performs tests of the insert, update, delete, and select statements, which are the most commonly used SQL statements in DML. Basic statements, subqueries and various join queries are tested.
misc	sql/_07_misc	Performs functionality tests of DCL (Data Control Language), including statistics update or other functionalities.
javasp	sql/_08_javasp	Performs functionality tests of Java stored procedures.
64-bit	sql/_09_64bit	Performs basic functionality test scenarios of the bigint and datetime types
Connect_by	sql/_10_connect_by	Performs a test of the hierarchical query feature
Code coverage	sql/_11_codecoverage	Performs a test of uncovered codes based on the code coverage results.
Syntax Extension	sql/_12_mysql_compatibility	Performs a test of the syntax extension.
BTS issues	sql/_13_issues	Performs a test of known issues, which comes from issue management system.
MySQL compatibility	sql/ _14_mysql_compatibility_2	Performs a unit test of the syntax extension 2.
FBO	sql/ _15_fbo	Performs a test of the FBO feature.
Index enhancement	sql/ _16_index_enhancement	Performs a unit test of the index enhancement.
SQL Extension	sql/ _17_sql_extension2	Performs a test of the syntax extension 2. Include s a test of syntax enhancements, system parame ters, show statements, date/time functions, string functions, aggregate functions, other functions.

Index enhancement	sql/ _18_index_enhancement_qa	Performs a test of the index enhancement. Includes a test of limit optimizing, using index clause enhancement, descending index scan, covering index, ordering index, optimizing group by clause, Index scan with like predicate, next key locking, etc.
SQL Extension 3 Index Enhancement Internationalization (CUBRID 9.0 Beta unit test)	sql/_19_apricot	Performs an unit test of syntax extension 3, performance and internationalization features. Includes multi-table UPDATE/DELETE, pseudo column, analytic functions, MERGE statements, ENUM type, filtered index, function based index, index skip scan, partition and collation.
MySQL compatibility for NEWS service	sql/_22_news_service_mysql_compatibility	Performs a test of several functions, regular expression and hint rewriting.
SQL Extension 3 Index Enhancement Internationalization (CUBRID 9.0 Beta QA scenario)	sql/_23_apricot_qa	Performs a test of syntax extension 3, performance and internationalizztion features, Test of syntax exetension 3 includes multi-table UPDATE/DELETE, pseudo column, analytic functions, MERGE statements, ENUM type, and other functions, Test of performance includes filtered index, function based index, index skip scan and partition enhancement. Test of internationalization includes tests of 11 languages.
SQL Extension 3 Internationalization (CUBRID 9.1 QA scenario)	sql/_24_aprium_qa	Performs a test of syntax extension, internationalization features. Test for syntax extension includes TRUNC, WIDTH_BUCKEY, ROUND, NTILE functions, LEAD analytic function, and direct access to partitions in INSERT/UPDATE statements. Test for internationalization includes collation per table, SHOW COLLATION, COLLATE modifier applied to expressions, etc.
844 feature enhancement	sql/_25_features_844	Performs a test of alter table to add columns when table already contains data
SQL Extension Internationalization (CUBRID 9.2 QA scenario)	sql/_26_features_920	Performs a test of sql extension, internationalization features. Test for sql extension includes NULLS order syntax, and some functions: FIRST_VALUE, LAST_VALUE, NTH_VALUE, CUME_DIST, PERCENT_RANK,MEDIAN. Test for internationalization includes new collations added, hash partition on columns with any collation, etc.

- MEDIUM Query Test

Total: 970		
Case Name	**Path**	**Description**
01_fixed	medium/_01_fixed	Performs regression test scenarios for bug fixes that have been implemented since the initial version.
02_xtests	medium /_02_xtests	Performs test scenarios for functionalities supported by CUBRID,

		but not by other DBMSs.
03_full_mdb	medium /_03_full_mdb	Performs test scenarios for sequential/index scan queries with an index.
04_full	medium /_04_full	Performs test scenarios that include testing queries for limit values of CUBRID.
05_err_x	medium /_05_err_x	Performs negative test scenarios for functionalities that are supported by CUBRID, but not by other DBMSs.
06_fulltests	medium /_06_fulltests	Performs test scenarios for search queries with OIDs.
07_mc_dep	medium /_07_mc_dep	Includes a query that gives various conditions to a WHERE clause in the SELECT query, and tests whether or not a correct result has been selected.
08_mc_ind	medium/_08_mc_ind	Includes scenarios that test queries performing schema change.

- SITE Query Test

Total: 1,213		
Case Name	**Path**	**Description**
k_count_q	site/k_count_q	Retrieves count (*) results of a query that is included in the kcc_q query.
k_merge_q	site/k_merge_q	Forces to give a hint to the kcc_q queries allowing merge joins.
k_q	site/k_q	Performs tests for OID reference, collection type, and path expression that are part of the object-oriented concept supported by CUBRID with different scalabilities. In addition, it performs functionality tests while increasing the number of join participating tables.
n_q	site/n_q	Performs tests for a complex query in which subqueries, outer/inner joins or group-by queries are combined, and checks whether correct results are retrieved.

- Utility (Shell) Test

  This test was performed to verify the basic DBMS functionalities using shell scripts. In particular, this test was also performed to verify CUBRID utilities that cannot be tested by SQL statements. Scenarios of shell scripts are executed to verify DBMS conformity.

Total: 1,754		
Case Name	**Path**	**Description**
utility	shell/_01_utility	Includes a script that tests the database management commands supported by CUBRID.
sqlx_init	shell/_02_sqlx_init	Includes scenarios that change the configuration of CUBRID DBMS parameters, and checks whether they are working correctly.
itrack	shell/_03_itrack	Includes scenarios that verify there is no regression by checking the bug fixes in CUBRID, and stores scenarios that cannot be tested by SQL.
miscellaneous	shell/_04_misc	Includes miscellaneous items, such as jdbc cache, query cache and async commit test
addition	shell/_05_addition	Includes scenarios added to improve code coverage and

		mainly tests the options of CUBRID utilities.
BTS issues	shell/_06_issues	Includes scenarios that verify there is no regression by checking the bug fixes in CUBRID, and stores scenarios that cannot be tested by SQL.
Index enhancement	shell/_07_index_enhancement	Includes scenarios that verify next key lock and change the configuration of CUBRID DBMS related to index enhancement, which has been added in CUBRID 2008 R4.0 Beta.
64bit scenario	shell/_09_64bit	Includes file size on linux 64 bit
improve coverage sceneario	shell/_11_codecoverage	Includes shell cases to improve coverage, all the cases are related to the system parameter test
xa datasource	shell/_21_xa	Includes scenarios to cover xa DataSource features
MySQL service compatibility	shell/_22_news_service_mysql_compatibility	Includes scenarios to test CUBRID compatibility with MySQL service
MySQL compatibility	shell/_23_mysql_compatibility	Includes scenarios that verify syntax extension, which has been added in CUBRID 2008 R3.1.
CUBRID 9.0 Beta QA	shell/_24_apricot	Includes scenarios that verify CUBRID 9.0 Beta functions such as i18n, enum, etc.
Unstable	shell/_25_ unstable	Includes scenarios that are not very stable
CUBRID 9.0 Beta QA	shell/_26_apricot_qa	Includes scenarios that added by QA to verify CUBRID 9.0 Beta functions such as i18n, cursor holdability, etc.
CUBRID 9.1 QA	shell/_27_aprium_qa	Includes scenarios that added by QA to verify prefix key, enum, collationof CUBRID 9.1 i18n function.
New feature and feature enhancement	shell/_28_ features_844	Includes error message enhancement, server statistic update and query profiling features
SQL Extension Internationalization (CUBRID 9.2 QA shells script scenario)	shell/_29_features_920	Performs a test for sql extension, internationalization features.Test for sql extension includes NULLS order syntax, and some functions: FIRST_VALUE, LAST_VALUE, NTH_VALUE, CUME_DIST, PERCENT_RANK,MEDIAN.Test of internationalization includes new collations added, hash partition on columns with any collation, etc.
Manual shell	Manually/*	All the manual test cases which can't be automated or need long time to regress

- HA Feature Test

Total: 179		
Case Name	**Path**	**Description**
Fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a node/process/broker fault occurs during insert/update/delete operations.
Bug regression	HA/shell/	Includes scenarios that verify there is no regression by checking the HA bug fixes in CUBRID

- HA Replication test

Total: 12,423		
Case Name	**Path**	**Description**
Test Cases migrated from SQL suite	N/A	Migrated existing SQL suite into HA environment. Execute them on master node, then check whether be replicated to slave or not.
Bug Regression	HA/shell/_24_functional_repl/	Includes scenarios that verify there is no regression by checking the HA bug fixes in CUBRID

- CCI Interface test

Total: 254		
Case Name	**Path**	**Description**
Features test	Interface/shell/_20_cci	Which contains CCI all APIs, each APIs are mentioned in manual are tested in shell scripts
Bug Regression	Interface/shell/_20_cci/_12_issue	Includes shell scripts which are written when verify CCI bts issues

- JDBC Interface test

Total: 1,530		
Case Name	**Path**	**Description**
Features test	N/A	Which include unit test for jdbc, jdbc spec 3.0 test, and other open source databases jdbc case migration

- CAS4MySQL/Oracle test

Total: 108		
Case Name	**Path**	**Description**
CAS4MySQL	N/A	Cas4MySQL test and CAS4MySQL BTS issues automation scripts
CAS4Oracle	N/A	Cas4Oracle test and Cas4Oracle BTS issues automation scripts

II. Performance Test Scenarios

- CUBRID Basic Performance Test

To evaluate the basic performance of DBMS, the following 5 variables were used. Database Server, Broker, and Load Generator were run on a single server.

- Number of data (or number of program loops)
  - ✧ Total number of data: 900,000 items
  - ✧ Number of program loops: 100,000 loops/program (900,000 items)
    - COMMIT Interval
      - After every execution
      - After 100 executions
      - After 1,000 executions
    - Number of concurrent users
      - 5 users
      - 10 users
    - Number of index attributes
      - create index idx1 on xoo(a)
      - create index idx2 on xoo(a,b)
      - create index idx3 on xoo(a,b,e)
    - Interface
      - JDBC (Dynamic SQL): Prepared statements were used.

- Test data
  - ✧ Test schema

```
CREATE TABLE xoo (
        a         int,
        b         int,
        c         int,
        d         int,
        e         char(10),
        f         char(20),
        g         char(30)
)
```

```
CREATE INDEX idx1 on xoo(a);
CREATE INDEX idx2 on xoo(a,b);
CREATE INDEX idx3 on xoo(a,b,e);
```

- ✧ Test data

  Enter data from 1 to 450,000; total number of data is 900,000.

- ✧ How to perform a test
  - Insert/update/select/delete data from a specific number.
  - For concurrent user tests, the start and end numbers are defined to prevent data from overlapping, in order to ensure that there is no competition between the concurrent clients.
  - For concurrent user test programs, a JDBC test program is tested with a multi-threaded program, and a C program is tested with a multi-process program.
  - If the number of loops is 10,000, a user repeats execution 10,000 times in the case of the 1-user test, and each user repeats execution 2,000 times in the case of the 5-user test. Similarly, if the number of loops is 100,000, a user repeats execution 100,000 times in the case of the 1-user test, and each user repeats execution 20,000 times in the case of the 5-user test.

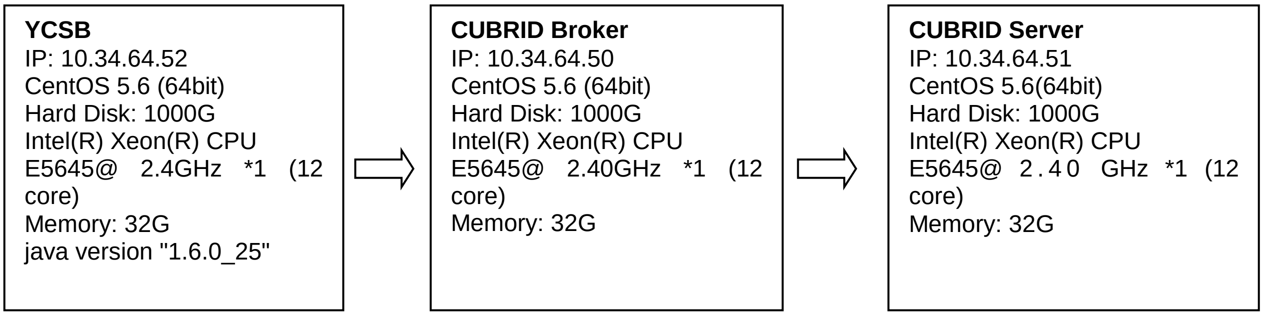
- ✧ How to measure test results
  - Measure the number of loops per second.
  - For concurrent user tests, add the execution times of all users.

■ YCSB Benchmark

This test was performed to verify CUBRID performance of not only basic operations but also compositive operations, which are insert, select, scan, update and mix of them.

■ Common Test Environment

- ✧ Test Servers



YCSB
IP: 10.34.64.52
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.4GHz *1 (12 core)
Memory: 32G
java version "1.6.0_25"

CUBRID Broker
IP: 10.34.64.50
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.40GHz *1 (12 core)
Memory: 32G

CUBRID Server
IP: 10.34.64.51
CentOS 5.6(64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.40 GHz *1 (12 core)
Memory: 32G

- ✧ CUBRID database volume configuration

```
cubrid  createdb  ycsb
cubrid  addvoldb -p data --db-volume-size=2G ycsb -S
cubrid  addvoldb -p data --db-volume-size=2G ycsb -S
cubrid  addvoldb -p index --db-volume-size=2G ycsb -S
cubrid  addvoldb -p index --db-volume-size=2G ycsb -S
cubrid  addvoldb -p temp --db-volume-size=2G ycsb –S
```

- ✧ Configuration for CUBRID
  - ⬩ cubrid_broker.conf:

```
SERVICE                    =ON
BROKER_PORT                =33000
MIN_NUM_APPL_SERVER        =320
MAX_NUM_APPL_SERVER        =320
APPL_SERVER_SHM_ID         =33000
LOG_DIR                    =log/broker/sql_log
ERROR_LOG_DIR              =log/broker/error_log
SQL_LOG                    =OFF
TIME_TO_KILL               =120
SESSION_TIMEOUT            =300
KEEP_CONNECTION            =AUTO
CCI_DEFAULT_AUTOCOMMIT     =ON
```

  - ⬩ cubrid.conf:

```
data_buffer_size=4G
sort_buffer_size=2M
cubrid_port_id=1523
max_clients=500
db_volume_size=512M
log_volume_size=512M
```

- ✧ Workload configuration on YCSB
  - ⬩ Insert operation (load)

```
recordcount=10000000
operationcount=10000000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=0
updateproportion=0
scanproportion=0
insertproportion=1
requestdistribution=zipfian
threads=300
fieldlength=10
```

  - ⬩ Select operation

```
recordcount=10000000
operationcount=10000000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=1
```

```
updateproportion=0
scanproportion=0
insertproportion=0
requestdistribution=zipfian
threads=300
fieldlength=10
table=usertable
```

- Scan operation

```
recordcount=10000000
operationcount=10000000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=0
updateproportion=0
scanproportion=1
insertproportion=0
requestdistribution=zipfian
fieldlength=10
table=usertable
maxscanlength=200
threads=300
```

- Update operation

```
recordcount=10000000
operationcount=10000000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=0
updateproportion=1
scanproportion=0
insertproportion=0
requestdistribution=zipfian
fieldlength=10
table=usertable
threads=300
```

- Mix operation

```
recordcount=10000000
operationcount=10000000
workload=com.yahoo.ycsb.workloads.CoreWorkload
readallfields=true
readproportion=0.3
updateproportion=0.3
scanproportion=0.1
insertproportion=0.3
requestdistribution=zipfian
fieldlength=10
table=usertable
maxscanlength=200
```

```
threads=300
```

- ✧ Test schema

```
Create table usertable (
userkey                          CHARACTER VARYING(100) PRIMARY KEY,
field1              CHARACTER VARYING(100),
field2              CHARACTER VARYING(100),
field3              CHARACTER VARYING(100),
field4              CHARACTER VARYING(100),
field5              CHARACTER VARYING(100),
field6              CHARACTER VARYING(100),
field7              CHARACTER VARYING(100),
field8              CHARACTER VARYING(100),
field9              CHARACTER VARYING(100),
field10             CHARACTER VARYING(100)
)
```

- ■ Test data on master server configuration
  - ✧ CUBRID server configuration
    - ⬧ async_commit=no
    - ⬧ group_commit_interval_in_msecs=0
- ■ Test data on slave server configuration
  - ✧ CUBRID server configuration
    - ⬧ async_commit=yes
    - ⬧ group_commit_interval_in_msecs=1000
- ■ Statements to be tested
  - ✧ Insert operation

```
INSERT INTOusertable(userkey, field1, field2, field3, field4, field5, field6, field7, field8, field9, field10)
VALUES (?, ?, ?, ?, ?, ?,?, ?, ?, ?, ?);
```

- ✧ Select operation

```
SELECT * FROM usertable WHERE userkey= ?;
```

- ✧ Scan operation

```
SELECT * FROM usertable WHERE userkey>= ?LIMIT ?;
```

- ✧ Update operation

```
UPDATE usertable set field1=?, field2=?, field3=?, field4=?, field5=?, field6=?, field7=?, field8=?, field9=?, field10=? WHERE
```

```
userkey = ?;
```

✧ Mix operation

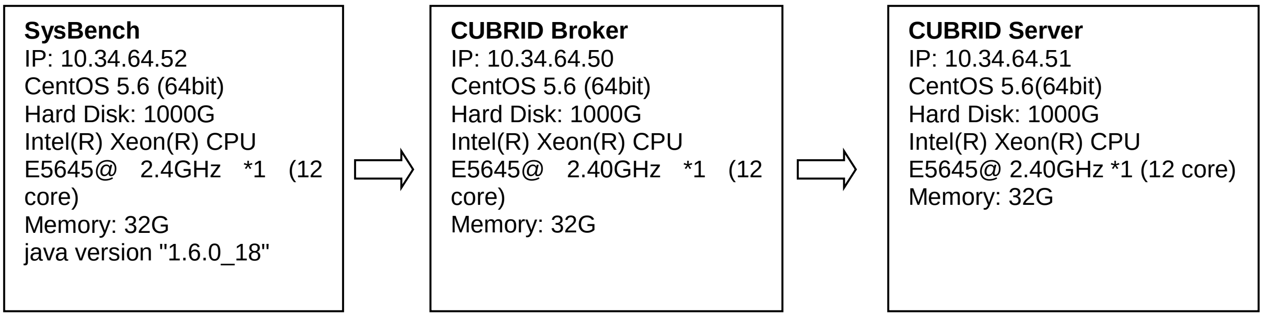
- Select operation: 30%
- Update operation: 30%
- Scan operation: 10%
- Insert operation: 30%

■ SysBench Benchmark

This test was performed to verify CUBRID performance based on OLTP business.

■ Test Environment

✧ Test Servers



SysBench
IP: 10.34.64.52
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.4GHz *1 (12 core)
Memory: 32G
java version "1.6.0_18"

CUBRID Broker
IP: 10.34.64.50
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.40GHz *1 (12 core)
Memory: 32G

CUBRID Server
IP: 10.34.64.51
CentOS 5.6(64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.40GHz *1 (12 core)
Memory: 32G

✧ CUBRID database volume configuration

```
cubrid  createdb  sysbench
cubrid  addvoldb -p data --db-volume-size=2G sysbench -S
cubrid  addvoldb -p data --db-volume-size=2G sysbench -S
cubrid  addvoldb -p index --db-volume-size=2G sysbench -S
cubrid  addvoldb -p temp --db-volume-size=2G sysbench  -S
```

✧ Configuration for CUBRID

- cubrid_broker.conf:

```
SERVICE                        =ON
BROKER_PORT                      =33000
MIN_NUM_APPL_SERVER               =320
MAX_NUM_APPL_SERVER               =320
APPL_SERVER_SHM_ID              =33000
LOG_DIR                         =log/broker/sql_log
ERROR_LOG_DIR                     =log/broker/error_log
SQL_LOG                         =OFF
TIME_TO_KILL                    =120
SESSION_TIMEOUT                   =300
KEEP_CONNECTION                   =AUTO
```

```
CCI_DEFAULT_AUTOCOMMIT    =ON
```

- cubrid.conf:

```
data_buffer_size=4G
log_buffer_size=4M
sort_buffer_size=2M
max_clients=500
cubrid_port_id=1523
db_volume_size=512M
log_volume_size=512M
async_commit=no
group_commit_interval_in_msecs=0
```

✧ Test schema

```
create table sbtest(
      id        INTEGER AUTO_INCREMENT PRIMARY KEY,
      k         INTEGER DEFAULT 0 NOT NULL,
      c         CHAR(120) NOT NULL DEFAULT '',
      pad       CHAR(60) NOT NULL DEFAULT '',
      INDEX i_sbtest_k ON sbtest (k)
)
```

✧ Configuration to start SysBench

```
./sysbench --test=oltp \
              --db-driver=cubrid \
              --cubrid-host=10.34.64.50 \
              --cubrid-port=33000 \
              --cubrid-db=sysbench \
              --num-threads=300 \
              --max-requests=0 \
              --max-time=14400 \
              --oltp-skip-trx=off \
              --oltp-read-only=off \
              --oltp-table-size=1000000 \
run
```

■ NBD Benchmark

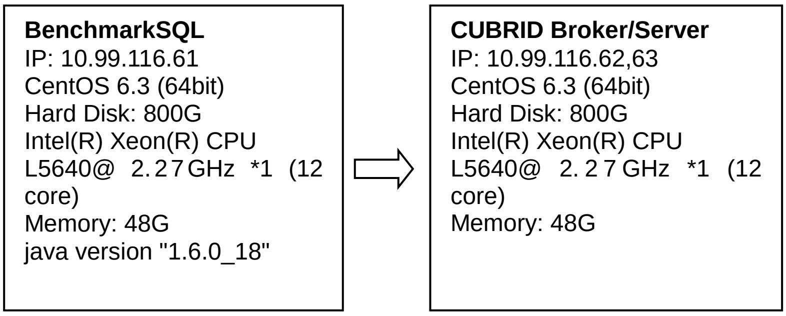
This test was performed to verify CUBRID performance using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. For more information about NBD Benchmark, see separate documents.

■ TPC-C Benchmark

BenchmarkSQL is a implementation of TPC-C standard. We can get more information in website http://sourceforge.net/projects/benchmarksql/. For this performance test, we just use this BenchmarkSQL tool to execute on CUBRID. In order to support CUBRID very well, we made some modification. See below for location:

SVN URL: http://svn.bds.nhncorp.com/xdbms/qatools/trunk/benchmarksql

- Test Environment
  - ✧ Test Servers



BenchmarkSQL
IP: 10.99.116.61
CentOS 6.3 (64bit)
Hard Disk: 800G
Intel(R) Xeon(R) CPU L5640@ 2.27GHz *1 (12 core)
Memory: 48G
java version "1.6.0_18"

CUBRID Broker/Server
IP: 10.99.116.62,63
CentOS 6.3 (64bit)
Hard Disk: 800G
Intel(R) Xeon(R) CPU L5640@ 2.27GHz *1 (12 core)
Memory: 48G

✧ CUBRID database volume configuration

```
cubrid createdb tpcdb10
cubrid addvoldb -p data --db-volume-size=2G tpcdb10 -S
cubrid addvoldb -p data --db-volume-size=2G tpcdb10- S
cubrid addvoldb -p index --db-volume-size=2G tpcdb10 -S
cubrid addvoldb -p temp --db-volume-size=2G tpcdb10 -S
```

✧ Configuration for CUBRID

- cubrid_broker.conf:

```
SERVICE                    =ON
BROKER_PORT                  =33000
MIN_NUM_APPL_SERVER           =120
MAX_NUM_APPL_SERVER           =120
APPL_SERVER_SHM_ID           =33000
LOG_DIR                    =log/broker/sql_log
ERROR_LOG_DIR                =log/broker/error_log
SQL_LOG                    =OFF
TIME_TO_KILL               =120
SESSION_TIMEOUT              =300
KEEP_CONNECTION              =AUTO
CCI_DEFAULT_AUTOCOMMIT   =ON
```

- cubrid.conf:

```
data_buffer_size=4G
max_clients=300
```

✧ BenchmarkSQL configuration

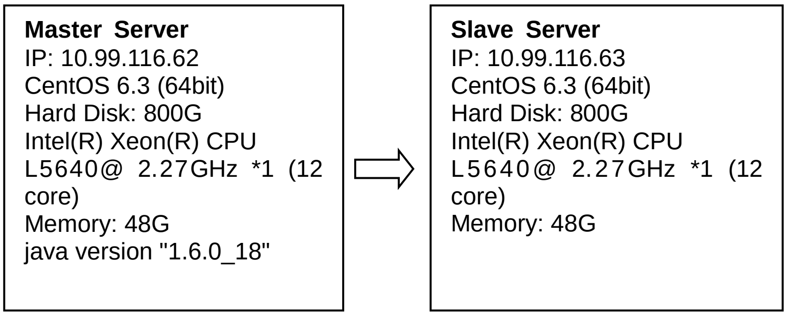
```
Number of warehouses: 10
Number of Terminals: 100
Execute minutes: 30

Payment : 43%, Order-Status: 4%,  Delivery: 4% , Stock-Level: 4% ,New-Order:45%
```

- Data Replication Test on HA

This test was performed to evaluate the performance of data replication on HA environment, by using YCSB to execute Insert, Mix operations on Master server with the related configurations, and check the delay time of data replication on Slave by CUBRID SQL statement.

✧ Test Servers



Master Server
IP: 10.99.116.62
CentOS 6.3 (64bit)
Hard Disk: 800G
Intel(R) Xeon(R) CPU L5640@ 2.27GHz *1 (12 core)
Memory: 48G
java version "1.6.0_18"

Slave Server
IP: 10.99.116.63
CentOS 6.3 (64bit)
Hard Disk: 800G
Intel(R) Xeon(R) CPU L5640@ 2.27GHz *1 (12 core)
Memory: 48G

✧ Table scheme

```
csql> ;sc usertable
=== <Help: Schema of a Class> ===
 <Class Name>
      usertable
 <Attributes>
      userkey                    CHARACTER VARYING(100) NOT NULL
      field1                     CHARACTER VARYING(100)
      field2                     CHARACTER VARYING(100)
      field3                     CHARACTER VARYING(100)
      field4                     CHARACTER VARYING(100)
      field5                     CHARACTER VARYING(100)
      field6                     CHARACTER VARYING(100)
      field7                     CHARACTER VARYING(100)
      field8                     CHARACTER VARYING(100)
      field9                     CHARACTER VARYING(100)
      field10                    CHARACTER VARYING(100)
 <Constraints>
      PRIMARY KEY pk_usertable_userkey ON usertable (userkey)
```

✧ Configuration for CUBRID

- cubrid_broker.conf:

```
SERVICE                  =ON
BROKER_PORT              =33000
MIN_NUM_APPL_SERVER      =320
MAX_NUM_APPL_SERVER      =320
APPL_SERVER_SHM_ID       =33000
LOG_DIR                  =log/broker/sql_log
ERROR_LOG_DIR            =log/broker/error_log
SQL_LOG                  =OFF
TIME_TO_KILL             =120
SESSION_TIMEOUT          =300
KEEP_CONNECTION          =AUTO
CCI_DEFAULT_AUTOCOMMIT   =ON
```

- cubrid.conf:

```
data_buffer_size=5G
max_clients=200
ha_mode=on
```

- cubrid_ha.conf

```
ha_copy_sync_mode=sync:sync
```

✧ YCSB configurations

- cubrid_load

```
recordcount=20000000
operationcount=10000000
readallfields=true
readproportion=0
updateproportion=0
scanproportion=0
insertproportion=1
requestdistribution=zipfian
threads=100
fieldlength=10
```

- cubrid_mix

```
recordcount=20000000
operationcount=10000000
readallfields=true
insertproportion=0.6
updateproportion=0.3
deleteproportion=0.1
requestdistribution=zipfian
fieldlength=10
table=usertable
maxscanlength=200
```

III. Stability Test Scenarios

DOTS, a sub-project of an open project called "Linux Test Project", is an open test tool for testing the DBMS.

- Test Related Schema (the Number of Data in Each Table)

```
CREATE TABLE REGISTRY (
    USERID              CHAR(15) NOT NULL PRIMARY KEY,
    PASSWD              CHAR(10),
    ADDRESS             CHAR(200),
    EMAIL               CHAR(40),
    PHONE               CHAR(15)
);

CREATE TABLE ITEM (
    ITEMID              CHAR(15) NOT NULL PRIMARY KEY,
    SELLERID            CHAR(15) NOT NULL,
    DESCRIPTION         VARCHAR(250) ,
    BID_PRICE           FLOAT,
    START_TIME          DATE,
    END_TIME            DATE,
    BID_COUNT           INTEGER
);

CREATE TABLE BID (
    ITEMID              CHAR(15) NOT NULL PRIMARY KEY,
    BIDERID             CHAR(15) NOT NULL,
    BID_PRICE           FLOAT,
    BID_TIME            DATE
);
```

- CUBRID configuration
  - cubrid_broker.conf

```
MIN_NUM_APPL_SERVER=20
MAX_NUM_APPL_SERVER=100
APPL_SERVER_MAX_SIZE=100
SQL_LOG=OFF
```

  - cubrid.conf

```
log_max_archives=150
async_commit=yes
group_commit_interval_in_msecs=10
checkpoint_every_npages=100000
checkpoint_interval_in_mins=10
max_clients=200
data_buffer_size=1G
```

- DOTs configuration

```
DURATION=24:00
CONCURRENT_CONNECTIONS= 20
AUTO_MODE = no
SUMMARY_INTERVAL = 5
MAX_ROWS= 900000000
```

- Data Size and How to Create Data

The initial number of data when starting the test is 0. Enter 1000 of data in the REGISTRY table. Next, enter 100 of data in the ITEM table as well as in the bid table. Then, update 100 times.

- Transaction types

✧ INSERT transaction 1

```
INSERT INTO ITEM (ITEMID,SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT)
VALUES (?, ?, ? ,?, ?, ?, ?)
```

✧ INSERT transaction 2

```
INSERT INTO BID (ITEMID,BIDERID,BID_PRICE,BID_TIME)
VALUES (?, ?, ?, ?)
```

✧ SELECT transaction 1

```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID = ?
```

✧ SELECT transaction 2

```
SELECT BIDERID, BID_PRICE, BID_TIME FROM BID WHERE ITEMID = ?
SELECT BIDERID, BID_PRICE, BID_TIME FROM BID WHERE ITEMID = ?
```

✧ UPDATE transaction 1

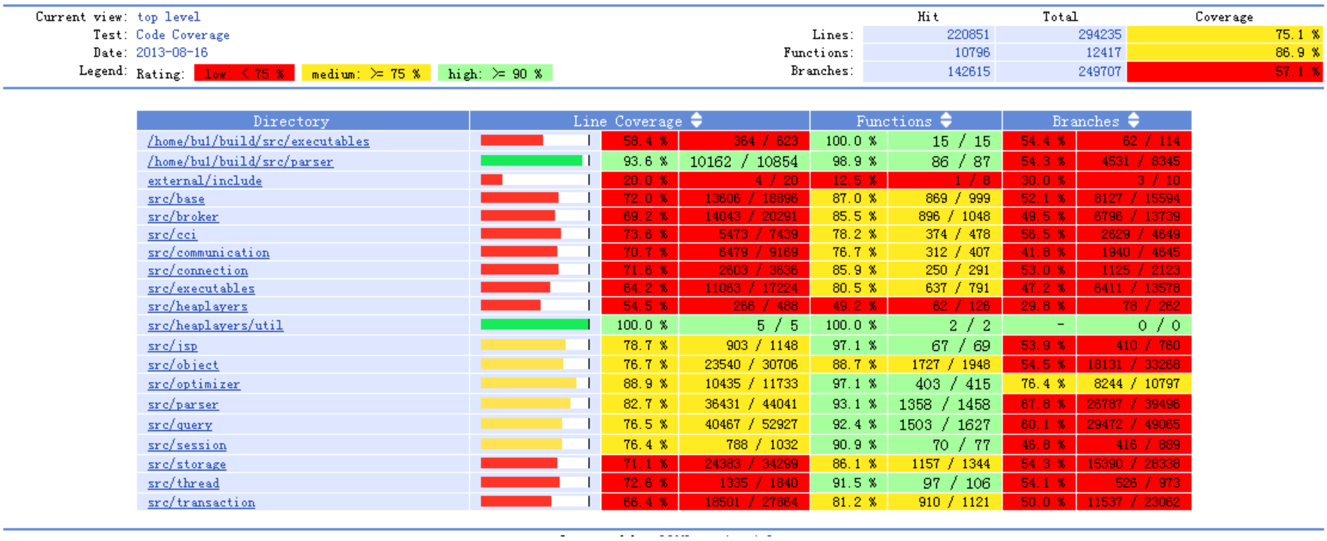
```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID =
UPDATE ITEM SET DESCRIPTION = ?,BID_PRICE = ?,START_TIME = ?,END_TIME = ? WHERE ITEMID = ?
```

■ How to Generate Load

✧ How to generate load

Use two threads to generate the initial load. Each thread repeats the insert/select/update queries mentioned above. The DOTS program checks CPU usage every 5 minutes. If the Peak CPU usage does not exceed 100%, the test continues, by adding two more threads.

IV. Scenario-based Code Coverage Results

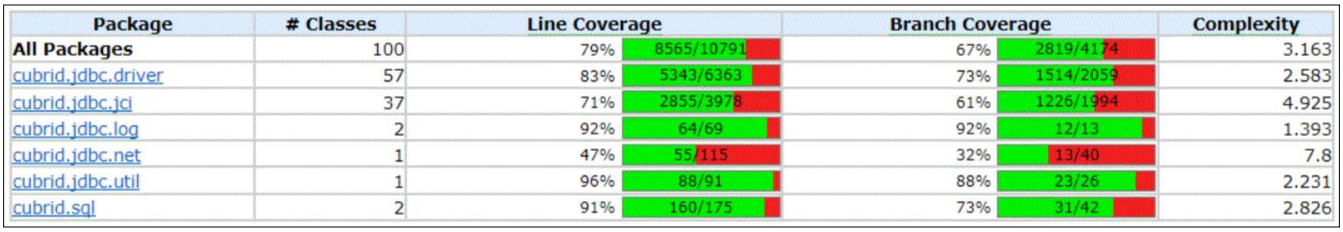


Current view: top level
Test: Code Coverage
Date: 2013-08-16
Legend: Rating: low: < 75 % medium: >= 75 % high: >= 90 %

	Hit	Total	Coverage
Lines:	220851	294235	75.1 %
Functions:	10796	12417	86.9 %
Branches:	142615	249707	57.1 %

Directory	Line Coverage		Functions		Branches	
/home/bu1/build/src/executables	58.4 %	364 / 623	100.0 %	15 / 15	54.4 %	62 / 114
/home/bu1/build/src/parser	93.6 %	10162 / 10854	98.9 %	86 / 87	54.3 %	4531 / 8345
external/include	20.0 %	4 / 20	12.5 %	1 / 8	30.0 %	3 / 10
src/base	72.0 %	13606 / 18896	87.0 %	869 / 999	52.1 %	8127 / 15594
src/broker	69.2 %	14043 / 20291	85.5 %	896 / 1048	49.5 %	6796 / 13739
src/cci	73.6 %	5473 / 7439	78.2 %	374 / 478	56.5 %	2629 / 4649
src/communication	70.7 %	6479 / 9169	76.7 %	312 / 407	41.8 %	1940 / 4645
src/connection	71.6 %	2603 / 3636	85.9 %	250 / 291	53.0 %	1125 / 2123
src/executables	64.2 %	11063 / 17224	80.5 %	637 / 791	47.2 %	6411 / 13578
src/heaplayers	54.5 %	266 / 488	49.2 %	62 / 126	29.8 %	78 / 262
src/heaplayers/util	100.0 %	5 / 5	100.0 %	2 / 2	-	0 / 0
src/jsp	78.7 %	903 / 1148	97.1 %	67 / 69	53.9 %	410 / 760
src/object	76.7 %	23540 / 30706	88.7 %	1727 / 1948	54.5 %	18131 / 33268
src/optimizer	88.9 %	10435 / 11733	97.1 %	403 / 415	76.4 %	8244 / 10797
src/parser	82.7 %	36431 / 44041	93.1 %	1358 / 1458	67.8 %	26787 / 39496
src/query	76.5 %	40467 / 52927	92.4 %	1503 / 1627	60.1 %	29472 / 49065
src/session	76.4 %	788 / 1032	90.9 %	70 / 77	46.8 %	416 / 889
src/storage	71.1 %	24383 / 34299	86.1 %	1157 / 1344	54.3 %	15390 / 28338
src/thread	72.6 %	1335 / 1840	91.5 %	97 / 106	54.1 %	526 / 973
src/transaction	66.4 %	18501 / 27864	81.2 %	910 / 1121	50.0 %	11537 / 23062

V. JDBC Code Coverage Results



Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	100	79%	8565/10791	67%	2819/4174	3.163
cubrid.jdbc.driver	57	83%	5343/6363	73%	1514/2059	2.583
cubrid.jdbc.jci	37	71%	2855/3978	61%	1226/1994	4.925
cubrid.jdbc.log	2	92%	64/69	92%	12/13	1.393
cubrid.jdbc.net	1	47%	55/115	32%	13/40	7.8
cubrid.jdbc.util	1	96%	88/91	88%	23/26	2.231
cubrid.sql	2	91%	160/175	73%	31/42	2.826