CUBRID 9.3 QA Completion Report

This document is the verification report of CUBRID 9.3 in terms of functionality, performance, stability and compatibility.

Table of Contents

1.Test Overview

1.1 Test Objectives

The objectives of this test are to perform functionality, performance, stability, compatibility and other tests for the final release candidate build of CUBRID 9.3 (hereinafter referred to as 9.3), which is under development for release in May 2014, in order to ensure that the product meets the business and user requirements; catch errors that can be bugs or defects; determine user acceptability; ensure and control the product quality before release. To guarantee the stability of CUBRID testing, we have used the test environments configured as below, which could be also adopted as a reference for the further testing in the future. Based on the comparison of function testing results and performance testing results between CUBRID 9.3 and CUBRID 9.2 (hereinafter referred to as 9.2), we could verify whether the performance of 9.3 has improved or not.

- CentOS 5/6 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 9.3.0.0206 (Linux 64-bit/32-bit, Windows 64-bit/32-bit)

1.2 Test Environment

1.2.1 Test Procedures

The test procedures we have used to verify the CUBRID product are shown below. To verify the product functionality, performance, stability and compatibility, the tests have been performed for 4 types of builds. The details of each test suite are described in the appendix of this report.

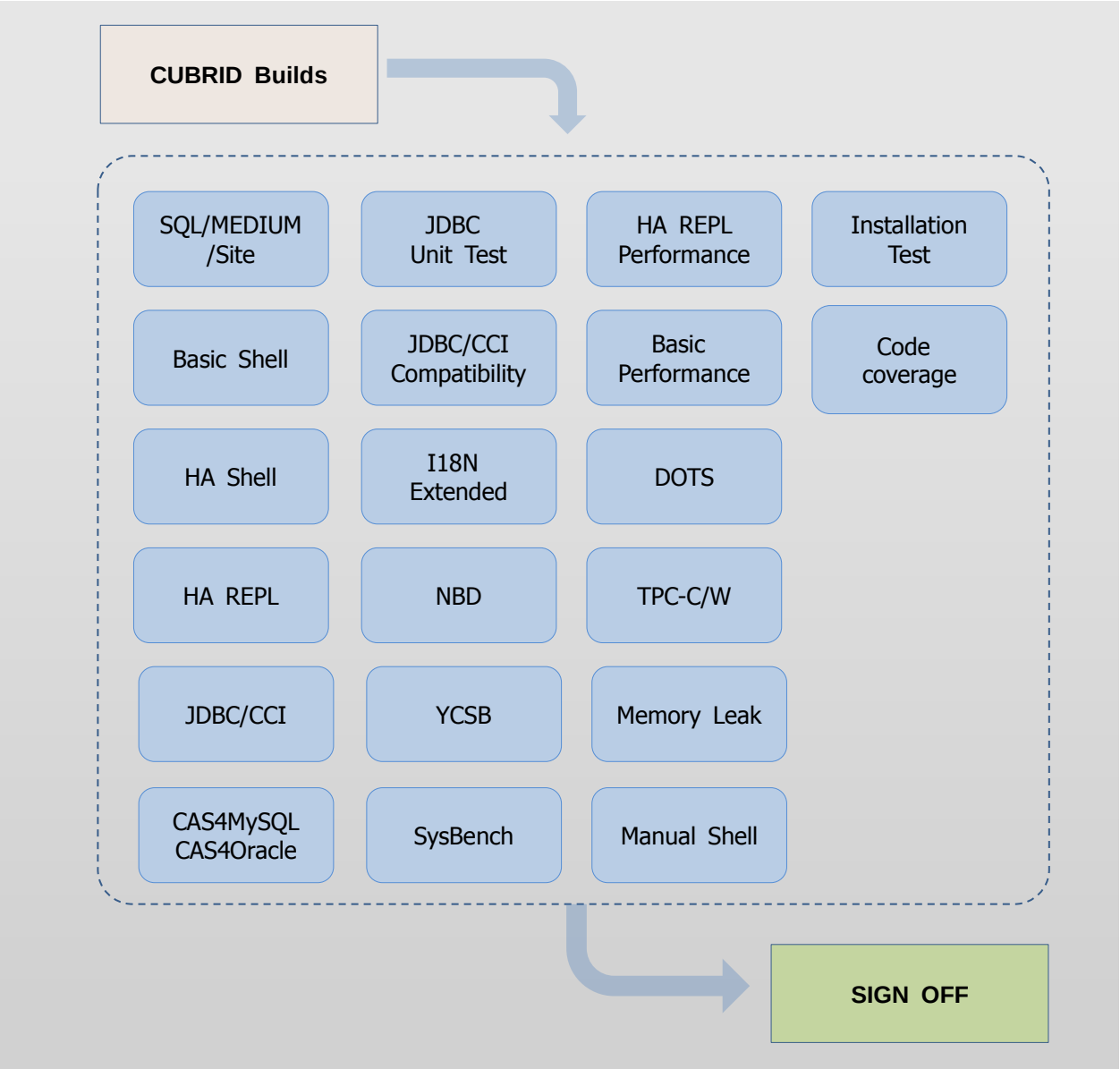


Figure 1. CUBRID Test Procedure

1.2.2 Hardware Test Environment

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

Table 1. Hardware Test Environment

No.	OS	CPU	MEMORY	DISK	Purpose
1	CentOS 6.3 (64-bit)	CPU 2.26 GHz (12 cores)	48 GB	SAS 300G * 6 (Raid1+0)	SQL Test / Stability Test
2	CentOS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores)	32 GB	SAS 600G * 3 (Raid5)	Performance Test
3	CentOS 5.7 (64-bit, VM)	CPU L5520 2.27 GHz (2 cores)	4 GB	50 GB	Shell
4	CentOS 5.3 (64-bit, VM)	CPU E5645 2.4 GHz (12 cores)	5 GB	100 GB	HA Shell
5	Windows Server 2003 (32-bit)	Xeon(R) CPU E5405 2.00GHz *2 (8 cores)	8 GB	SATA 500G * 2 (No Raid)	SQL Test / Basic Performance
6	Windows Server 2003 (64-bit)	Xeon(R) CPU E5506 @ 2.13GHz*2 (8 cores)	24 GB	SATA 500G*2 Raid0	SQL Test / Basic Performance

Note: All the tests are based on these kinds of server configurations to execute.

1.3 Test Category

The following tests have been performed to determine whether 9.3 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
  - SQL query test via CCI Interface
  - CAS4MySQL/Oracle
  - I18n Extended tests
- 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
  - Database volume image compatibility test
- Installation tests
- Other tests
  - Test for checking 9.3 functionalities/bug fixes
  - Memory check (SQL/MEDIUM) by Valgrind
  - Basic functional test on debug build

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 14,774 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 2. Result of Basic Query Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
SQL query test (JDBC and CCI)	12,591	12,591	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. 2,061 scenarios of shell scripts have been executed to verify DBMS conformity.

Table 3. Result of Basic Utility and Other Scenario Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Daily regression shell	1,884	1,884	100%
Manual shell	139	139	100%
Long time shell	38	38	100%

2.1.3 HA Feature Tests

246 scenarios of shell scripts have been executed to verify HA features and the regressions.

Table 4. Result of HA Feature Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Daily regression HA script	180	180	100%
Fault test	8	8	100%
Long time HA script	8	8	100%
Manual HA script	50	50	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. 12,305 scenarios based on SQL files have been executed to verify the data consistency between Master and Slave.

Table 5. Result of HA Replication Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Test Cases migrated from SQL suite	12,224	12,224	100%
Bug regression	81	81	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. 288 scenarios based on shell scripts have been executed on the release build and the debug build to verify all the CCI API features and the BTS issue regressions.

Table 6. Result of CCI Interface Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Basic features	213	213	100%
Bug regression	75	75	100%

2.1.6 JDBC Interface Tests

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

Table 7. 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

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

Table 8. Result of CAS4MySQL/Oracle Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
CAS4MySQL	62	62	100%
CAS4Oracle	62	62	100%

2.1.8 I18n Extended Tests

This test has been performed to verify the I18n functionality under various environments with different DB charsets and client collations. SQL and MEDIUM test suites have been executed with the following 14 combinations of DB charset and client collation. All the test results have been passed.

Table 9. Result of I18n Extended Tests

No.	DB Charset	Client Collation	Number of SQL Files passed	Number of MEDIUM Files passed
1	utf8	utf8_bin	12,593	970
2	euckr	iso88591_bin	12,593	970
3	iso88591	euckr_bin	12,593	970
4	euckr	iso88591_en_ci	12,593	970
5	iso88591	iso88591_en_ci	12,593	970
6	utf8	iso88591_en_ci	12,593	970
7	iso88591	utf8_bin	12,593	970
8	utf8	utf8_en_ci	12,593	970
9	euckr	utf8_bin	12,593	970
10	euckr	euckr_bin	12,593	970
11	utf8	iso88591_bin	12,593	970
12	utf8	euckr_bin	12,593	970
13	iso88591	utf8_en_ci	12,593	970

14	euckr	utf8_en_ci	12,593	970

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 performance of INSERT has significant improvement (around 20%) on 9.3, and the performance of UPDATE, SELECT and DELETE on 4 platforms (Linux 32, 64bit and Windows 32, 64bit) has also improved about 5% ~ 10% comparing with 9.2.

A. Linux: Performance Comparison between 9.2 (64-bit) and 9.3 (64-bit)

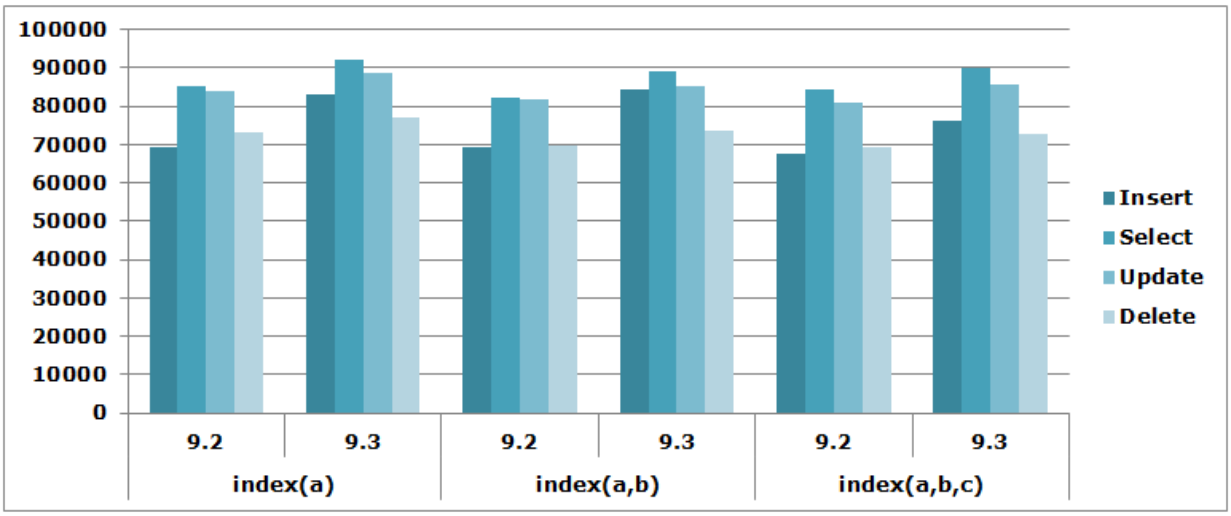


Figure 2. Performance Comparison between 9.2 and 9.3 (Linux 64-bit)

Table 10. Performance Comparison between 9.2 and 9.3 (Linux 64-bit)

Operations	index(a)			index(a,b)			index(a,b,c)		
	9.2	9.3	Ratio	9.2	9.3	Ratio	9.2	9.3	Ratio
Insert	69,134	83,293	120%	69,280	84,235	122%	67,802	76,044	112%
Select	85,082	91,948	108%	82,043	88,949	108%	84,462	89,820	106%
Update	83,765	88,687	106%	81,737	85,311	104%	81,061	85,638	106%
Delete	73,324	77,269	105%	69,960	73,554	105%	69,283	72,567	105%
Total	311,305	341,197	110%	303,020	332,049	110%	302,608	324,069	107%

(Unit: TPS)

B. Linux: Performance Comparison between 9.2 (32-bit) and 9.3 (32-bit)

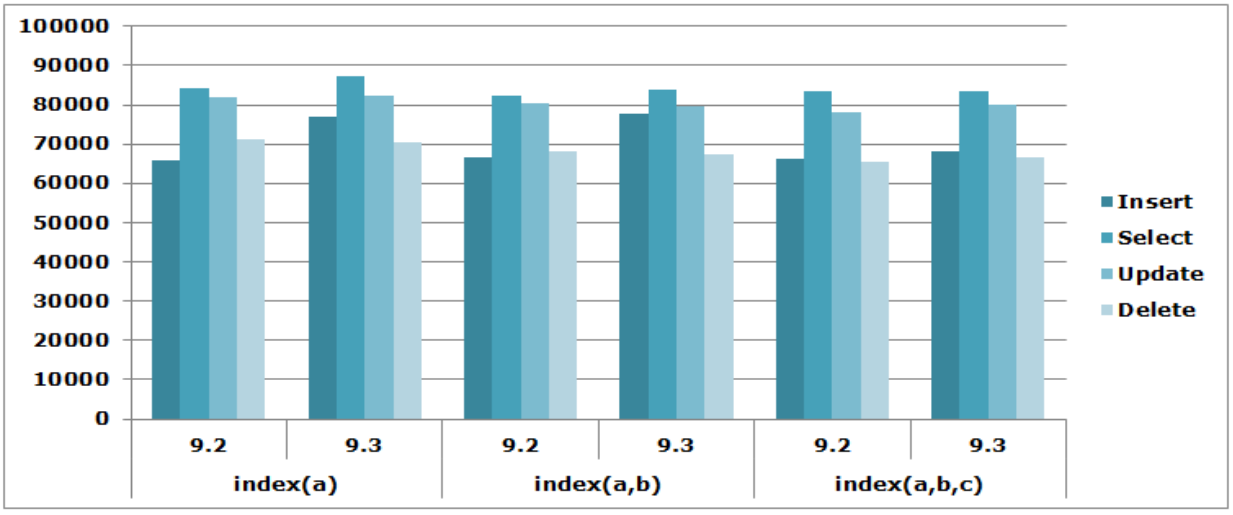


Figure 3. Performance Comparison between 9.2 and 9.3 (Linux 32-bit)

Table 11. Performance Comparison between 9.2 and 9.3 (Linux 32-bit)

Operations	index(a)			index(a,b)			index(a,b,c)		
	9.2	9.3	Ratio	9.2	9.3	Ratio	9.2	9.3	Ratio
Insert	65,912	76,815	117%	66,680	77,697	117%	66,066	68,327	103%
Select	84,303	87,275	104%	82,412	83,710	102%	83,514	83,378	100%
Update	82,094	82,382	100%	80,350	79,600	99%	78,075	80,156	103%
Delete	71,067	70,522	99%	68,212	67,406	99%	65,413	66,479	102%
Total	303,376	316,994	104%	297,654	308,413	104%	293,068	298,340	102%

(Unit: TPS)

C. Windows: Performance Comparison between 9.2 (64-bit) and 9.3 (64-bit)

According to the test result below, we can see that the performance of most operations has significant improvement (around 10% improvement), and a little drop on Delete operation could be regarded as a normal fluctuation.

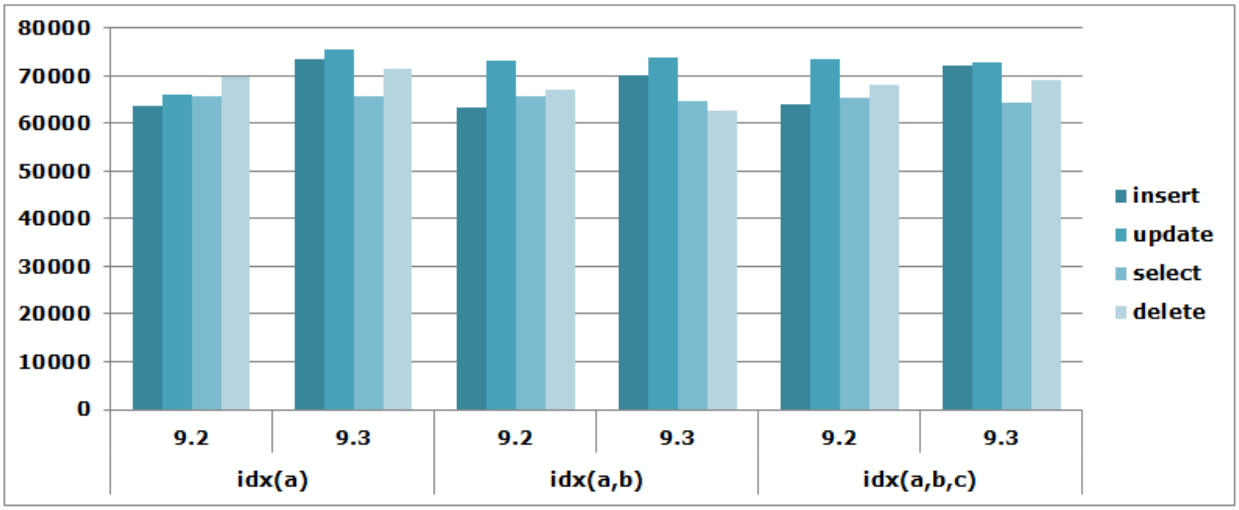


Figure 4. Performance Comparison between 9.2 and 9.3 (Windows 64-bit)

Table 12. Performance Comparison between 9.2 and 9.3 (Windows 64-bit)

Operations	idx(a)			idx(a,b)			idx(a,b,c)		
	9.2	9.3	Ratio	9.2	9.3	Ratio	9.2	9.3	Ratio
Insert	63,755	73,590	115%	63,428	70,145	111%	64,095	72,128	113%
Update	66,123	75,470	114%	73,009	73,964	101%	73,342	72,881	99%
Select	65,702	65,711	100%	65,690	64,828	99%	65,523	64,459	98%
Delete	69,926	71,318	102%	67,222	62,509	93%	68,145	69,205	102%
Total	265,506	286,089	108%	269,349	271,446	101%	271,105	278,673	103%

(Unit: TPS)

D. Windows: Performance Comparison between 9.2 (32-bit) and 9.3 (32-bit)

According to the test result below, we can see the performance of Insert and Update also has nearly 10% of improvement, and other operations are normally better than 9.2.

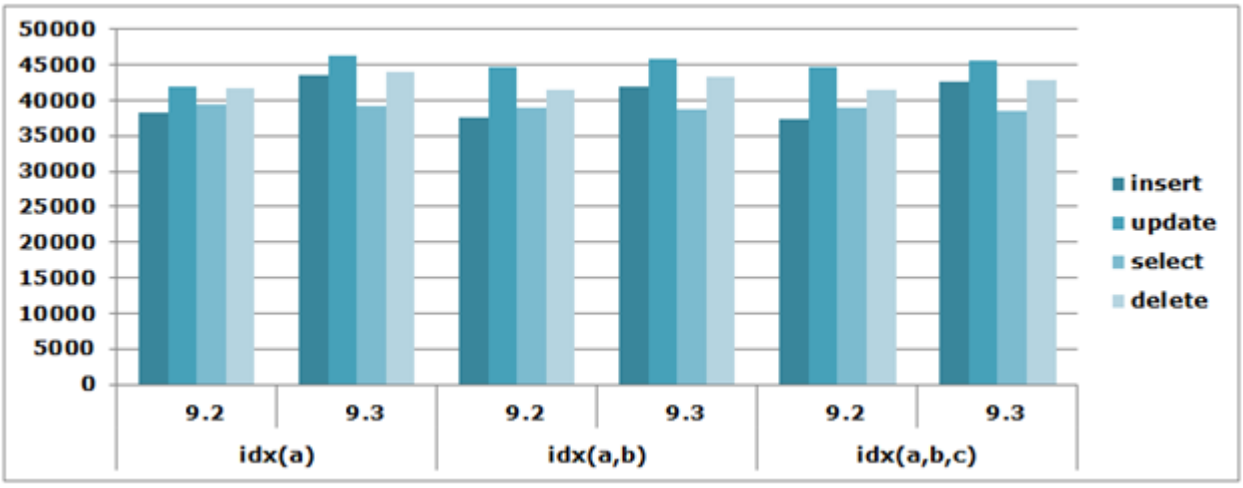


Figure 5. Performance Comparison between 9.2 and 9.3 (Windows 32-bit)

Table 13. Performance Comparison between 9.2 and 9.3 (Windows 32-bit)

Operations	idx(a)			idx(a,b)			idx(a,b,c)		
	9.2	9.3	Ratio	9.2	9.3	Ratio	9.2	9.3	Ratio
Insert	38,328	43,470	113%	37,488	42,041	112%	37,422	42,727	114%
Update	41,988	46,195	110%	44,792	45,728	102%	44,637	45,599	102%
Select	39,343	39,115	99%	39,019	38,653	99%	38,946	38,536	99%
Delete	41,785	43,905	105%	41,359	43,346	105%	41,377	42,928	104%
Total	161,444	172,685	107%	162,658	169,768	104%	162,382	169,790	105%

(Unit: TPS)

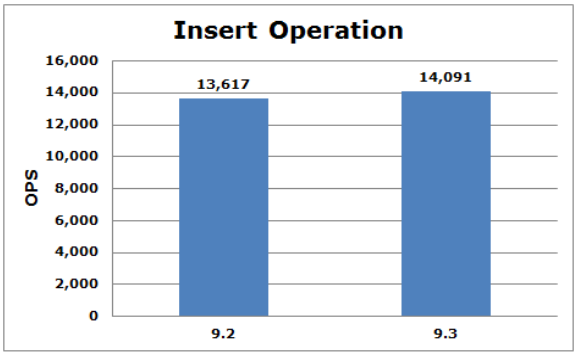
2.2.2 YCSB Performance Test

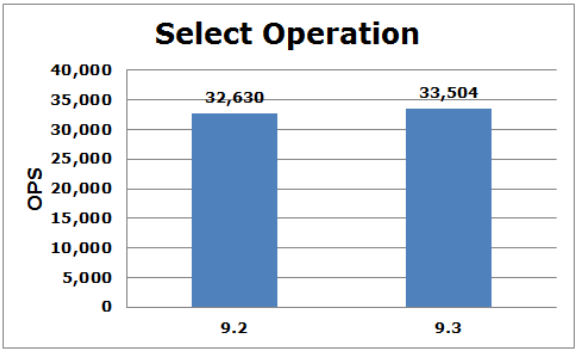
As a framework for benchmarking system, YCSB is popular and widely used in the world nowadays (see also https://github.com/brianfrankcooper/YCSB/wiki). This test has been performed to verify CUBRID performance of not only basic operations but also compositive operations, which are insert, select, scan, update and the mix of them. For more information about the test scenarios, see appendix II. Based on the testing results shown below, the performance of all operations on 9.3 is better than 9.2, the Update operation is more prominent especially; it has nearly 10% of improvement.

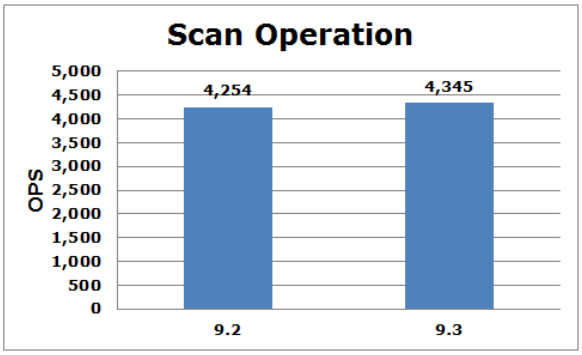
A. Master Server Configuration: Performance Comparison between 9.2 (64-bit) and 9.3 (64-bit)

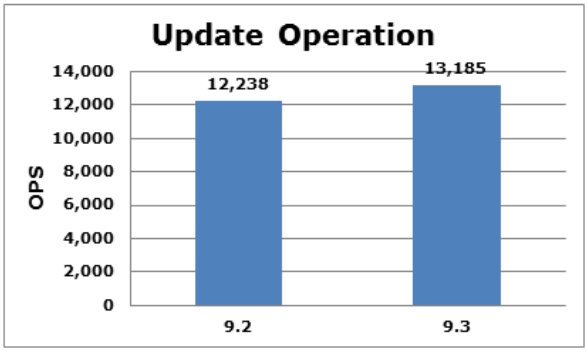
Table 14. Result of YCSB Benchmark (Master Server)

Operations	Throughput(OPS)			Average Latency(ms)		95th Percentile Latency(ms)	
	9.2	**9.3**	**Ratio**	**9.2**	**9.3**	**9.2**	**9.3**
Insert	13,617	14,091	103%	21	20	41	40
Select	32,630	33,504	103%	9	8	34	32
Scan	4,254	4,345	102%	63	62	247	246
Update	12,238	13,185	108%	23	22	21	17
Mix	12,497	12,579	101%	N/A	N/A	N/A	N/A









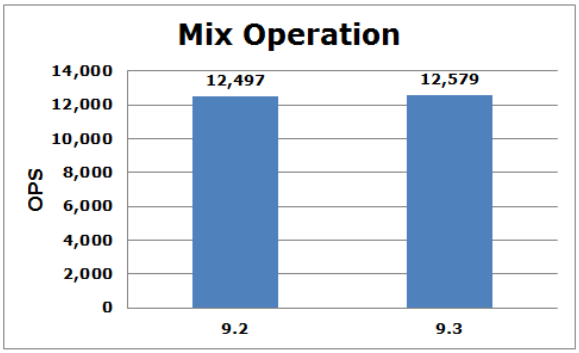
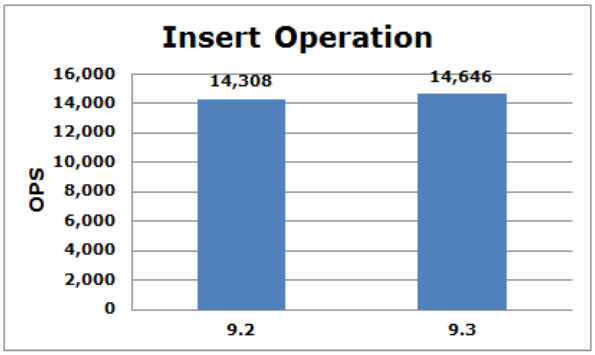


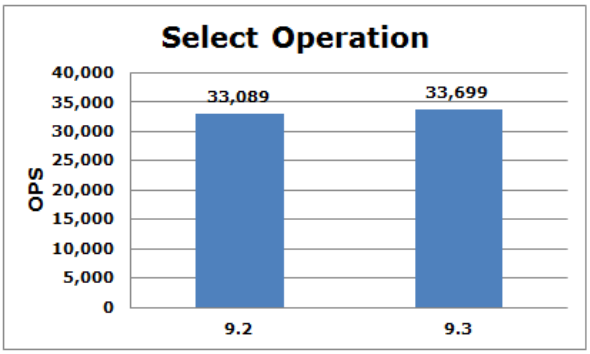
Figure 6. Result of YCSB Benchmark (Master Server)

B. Slave Server Configuration: Performance Comparison between 9.2 (64-bit) and 9.3 (64-bit)

Table 15. Result of YCSB Benchmark (Slave Server)

Operations	Throughput(OPS)			Average Latency(ms)		95th Percentile Latency(ms)	
	9.2	**9.3**	**Ratio**	**9.2**	**9.3**	**9.2**	**9.3**
Insert	14,308	14,646	102%	20	19	47	41
Select	33,089	33,699	102%	8	8	33	32
Scan	4,227	4,407	104%	64	61	248	245
Update	12,917	13,855	107%	22	21	16	15
Mix	12,993	13,277	102%	N/A	N/A	N/A	N/A





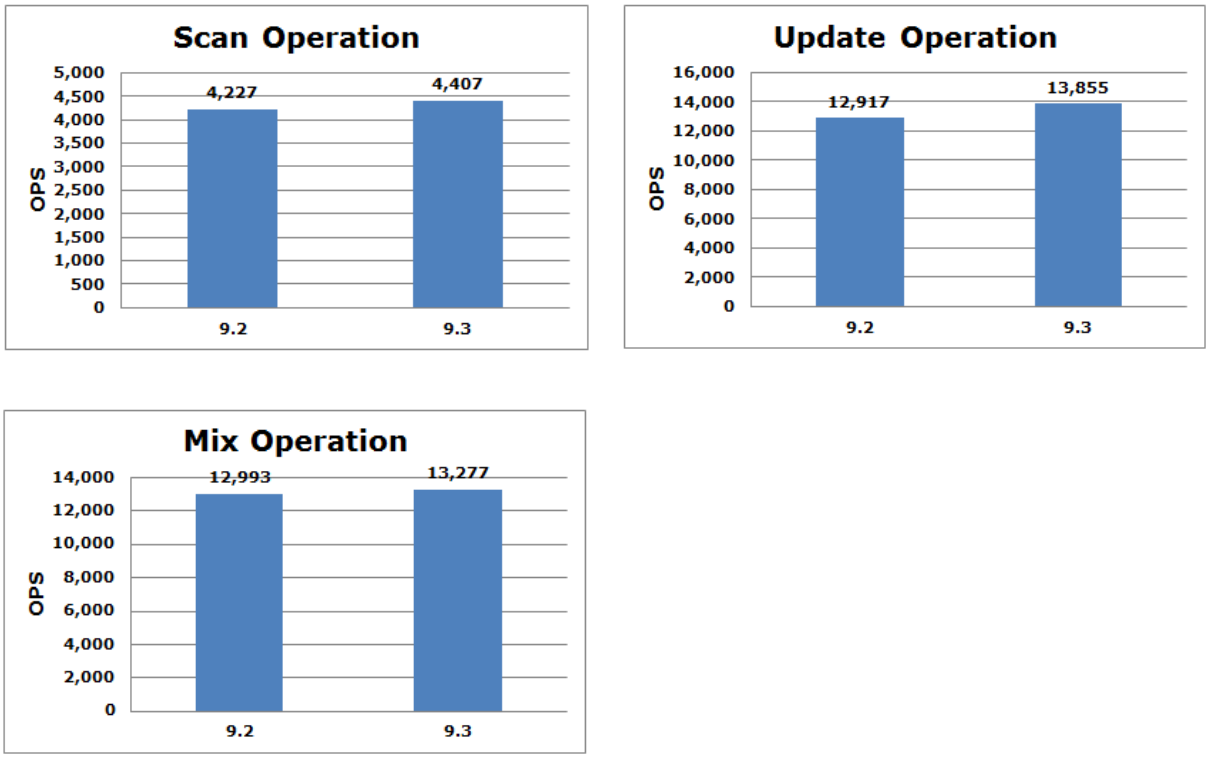


Figure 7. Result of YCSB Benchmark (Slave Server)

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.

Based on the testing results shown below, the performance of SysBench on 9.3 has no significant changes comparing with 9.2.

A. SysBench performance comparison between 9.2 (64-bit) and 9.3 (64-bit)

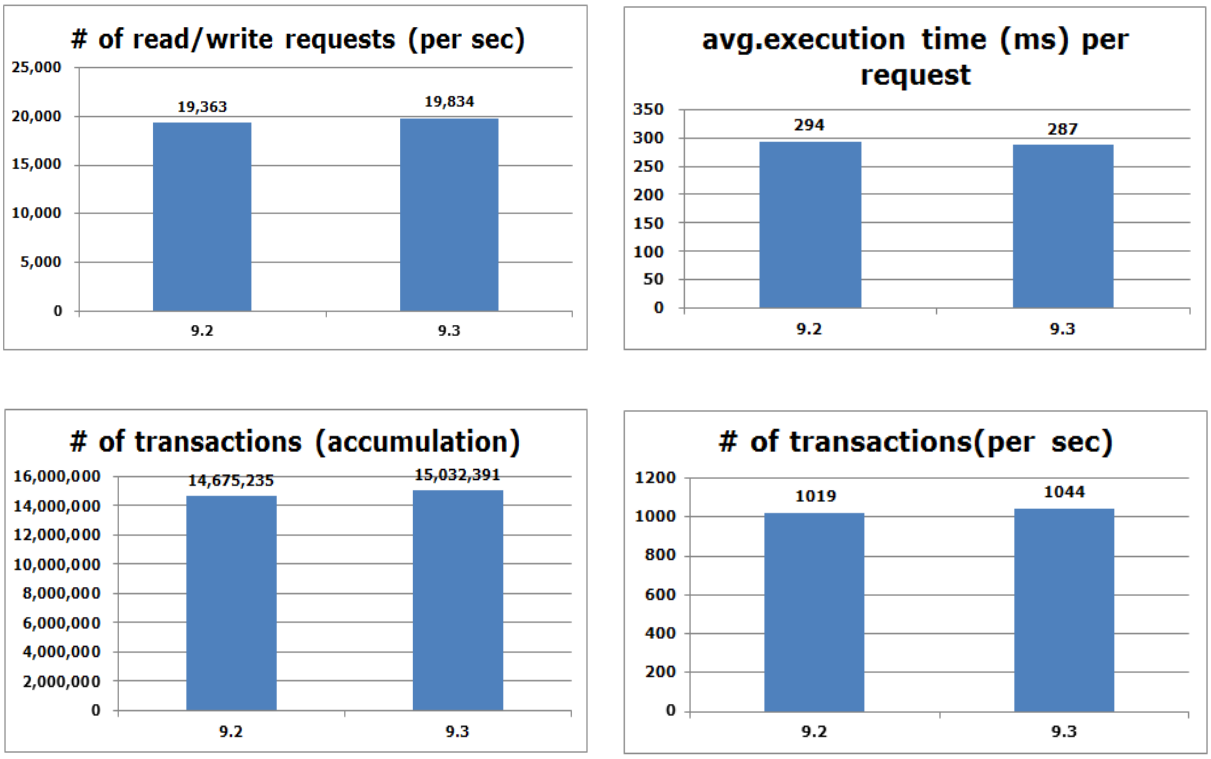


Figure 8. Result of SysBench benchmark

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 9.3 has no significant changes comparing with of 9.2.

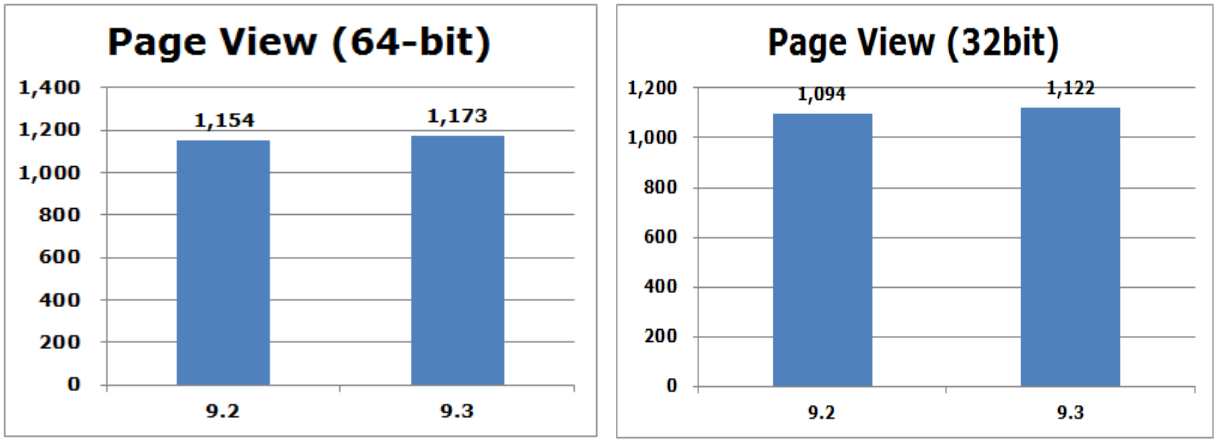


Figure 9. NBD performance comparison

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

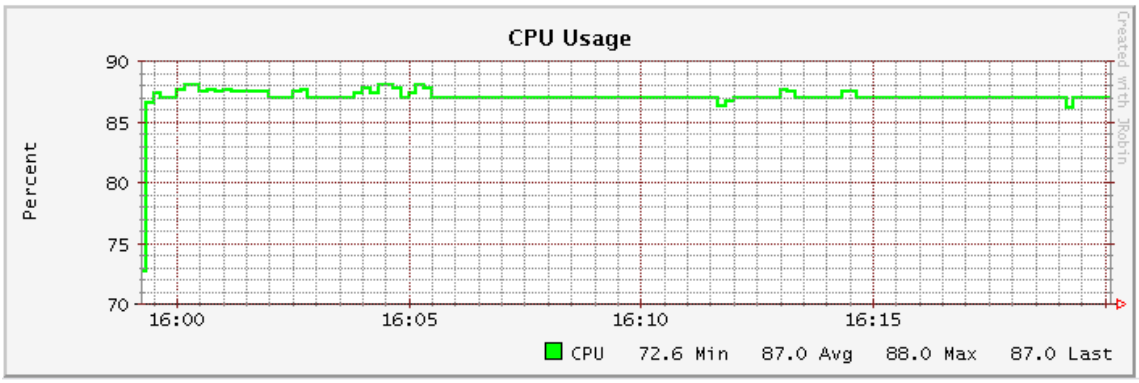


Figure 10. CPU Usage for NBD Benchmark

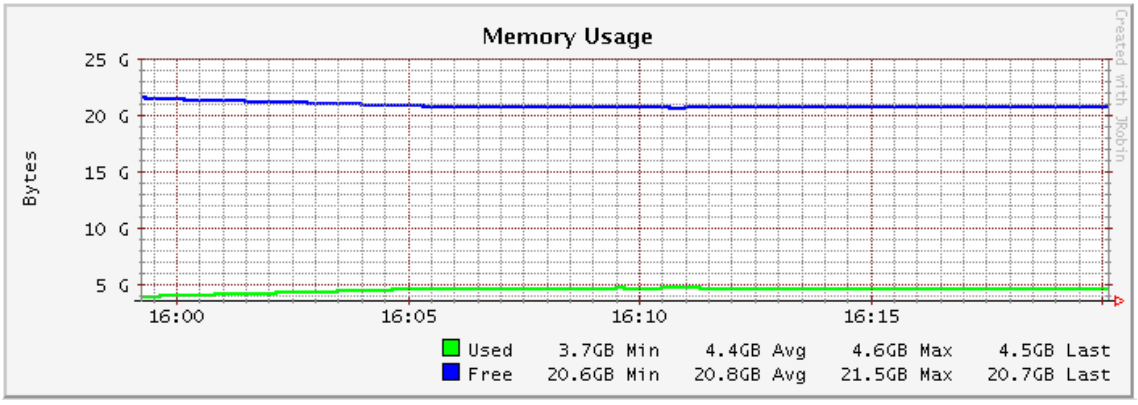


Figure 11. Memory Usage for NBD Benchmark

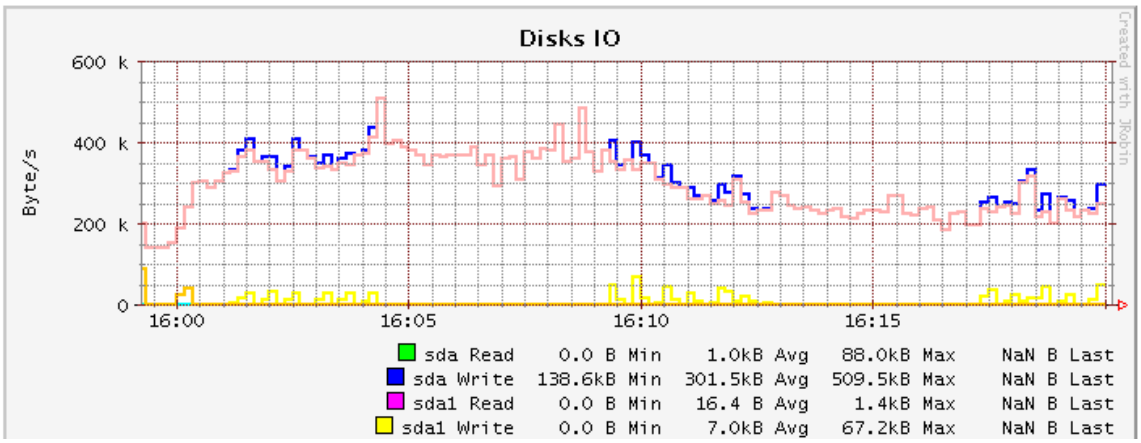


Figure 12. Disks IO status for NBD Benchmark

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 9.3 has slightly improved on tpmC.

A. TPC-C performance comparison between 9.2 (64-bit) and 9.3 (64-bit)

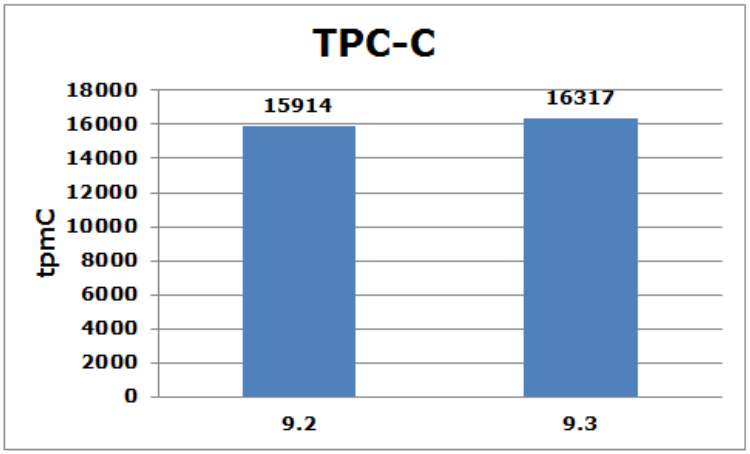


Figure 13. tpmC comparison of TPC-C benchmark

2.2.6 Data Replication Test on HA

This test has been performed to evaluate the performance of data replication under CUBRID HA environment. It uses YCSB to execute INSERT, MIX operations on Master server with the relevant configurations, and checks the delay time of data replication on Slave server. For more details, please refer to appendix II. As shown in the table below, the performance of data synchronization on 9.3 has improved about 50% comparing with 9.2 and has significantly improved comparing with 2008 R4.1.

Table 16. Data replication performance comparison

Version	Delay Time (sec.)
2008 R4.1	4,397
9.2	490
9.3	236

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. According to the test results shown below, the system has operated stably without any abnormalities during 50 hours. You can ignore the failures because they are unique violations due to the modification of duplicated data.

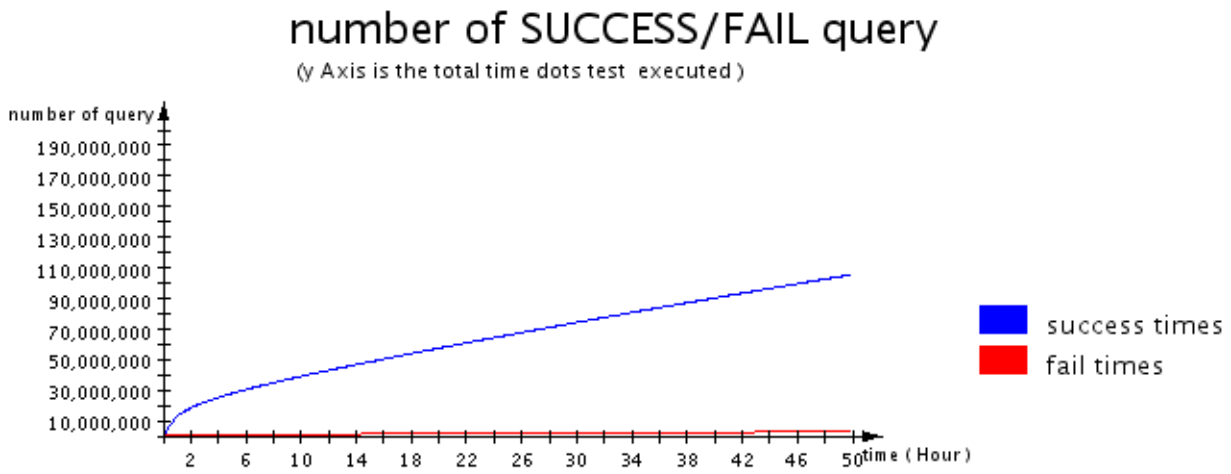


Figure 14. The number of SUCCESS/FAIL Queries of DOTS Test

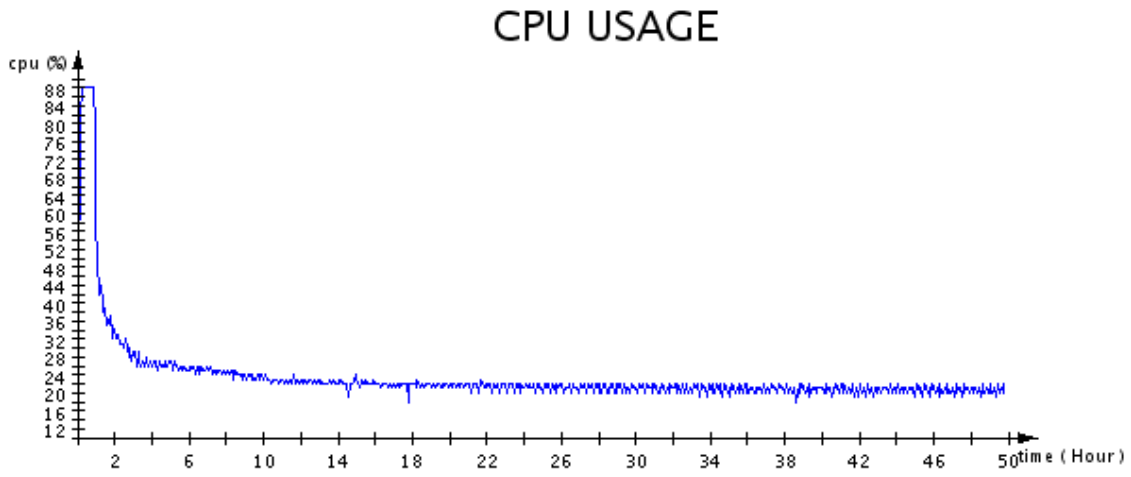


Figure 15. CPU Usage of DOTS Test

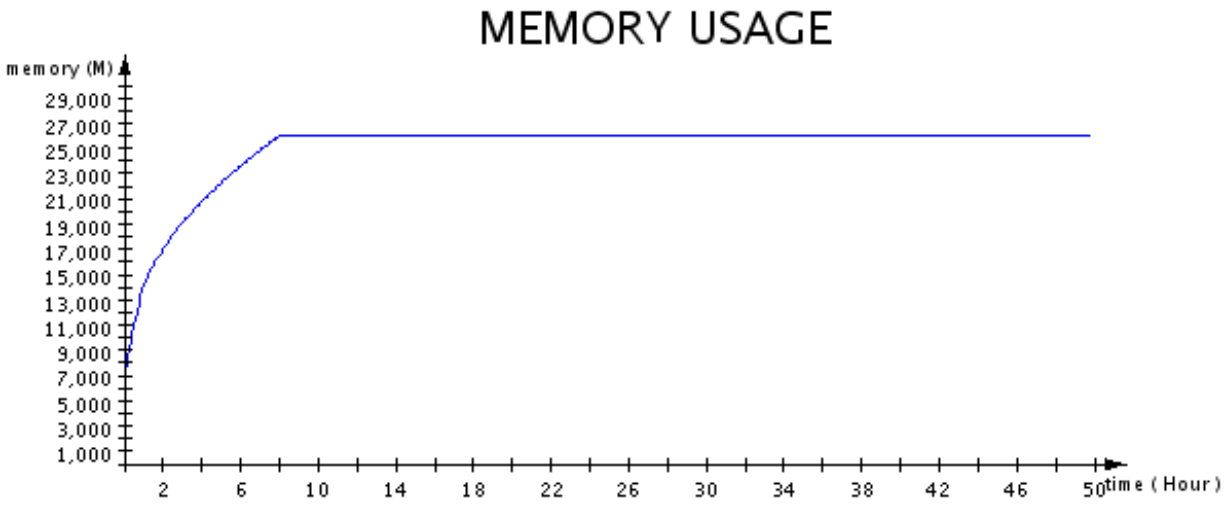


Figure 16. Memory Usage of DOTS Test

2.4 Compatibility Test Results

2.4.1 JDBC and CCI Compatibility Test

This test has been performed to verify the JDBC and CCI compatibility among 2008 R4.1, 2008 R4.3, 2008 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 17. Result of JDBC Compatibility Tests when 9.3 as Driver

Test Component	Scenario	2008 R4.1 Server	2008 R4.3 Server	2008 R4.4 Server	9.1 Server	9.2 Server
9.3 Driver	SQL	8,701	8,737	8,768	12,094	12,374
	Medium	968	970	970	970	970
	Site	1,213	1,213	1,213	1,213	1,213
	Shell_cci	167	187	221	230	253

Table 18. Result of JDBC Compatibility Tests when 9.3 as Server

Test Component	Scenario	2008 R4.1 Driver	2008 R4.3 Driver	2008 R4.4 Driver	9.1 Driver	9.2 Driver
9.3 Server	SQL	12,393	12,392	12,389	12,560	12,591
	Medium	970	970	970	970	970
	Site	1,213	1,213	1,213	1,213	1,213
	Shell_cci	203	211	186	269	272

2.4.2 Database Volume Image Compatibility Test

This test has been performed to verify the database volume image compatibility between 9.2 and 9.3. MEDIUM and Site Tests were executed. All the test results have been passed.

Table 19. Result of Database Volume Image Compatibility Test

Database Volume Image Compatibility	Number of MEDIUM Files passed	Number of Site Files passed
9.3 Server with 9.2 Image	970	1,213
9.2 Server with 9.3 Image	970	1,213

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
- Execute some negative scenarios for installation test

Table 20. 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 Windows 7 on 32-bit and 64-bit Windows XP on 32-bit and 64-bit	PASS
RPM/SH/EXE	64-bit rpm to install on 32-bit Linux 64-bit sh to install on 32-bit Linux 64-bit exe to install on Windows server 2008, 2003, Win7, XP on 32-bit platforms	PASS

2.6 Other Test Results

The entire bug and issue fixes for 9.3 have been confirmed.

2.7 Quality Index

The standard quality index of 9.3 is listed below.

Table 21. Quality Index of 9.3

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.	1,422,245 LOC
			Total number of source code lines in the changed files	1,422,245 LOC
QA Scenario Code Coverage	76%	76.5%	Number of tested statements	235,109
			Total number of statements	307,462
Fault Density Detected by Static Analysis	4 /KLOC	4.80 /KLOC	Number of faults detected by static analysis (Level 1)	350
			Number of faults detected by static analysis (Level 2)	89
			Number of faults detected by static analysis (Level 3)	871
			Number of faults detected by static analysis (Level 4)	0
			Total number of source code lines	1,019,178 LOC
Cyclomatic Code Complexity	3.3%	2.9%	Number of modules whose complexity is over 30	706
			Total number of modules in a project	24,323
	12%	15.6%	Number of modules whose complexity is over 10	3,807
			Total number of modules in a project	24,323

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 9.3. 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 performance of INSERT has significant improvement (around 20%), and the performance of UPDATE, SELECT and DELETE on 4 platforms(Linux 32-bit, 64-bit and Windows server 2003 32-bit and 64-bit) also has slight improvement (around 5% ~ 10%).

For YCSB, we can see that the performance on all YCSB operations on 9.3 is better than 9.2, especially on UPDATE operation; it has nearly 10% of improvement.

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

For TPC-C, there are no significant changes for performance of tpmC, just has slight improvement of about 3%.

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 50 hours of execution, and 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 2008 R4.1, and it also has around 50% of improvement comparing with 9.2.

From the result of compatibility test, we can reach the conclusion that JDBC and CCI on 2008 R4.1, 2008 R4.3, 2008 R4.4, 9.1, and 9.2 have compatibility with 9.3 server, and JDBC and CCI on 9.3 also have compatibility with 2008 R4.1, 2008 R4.3, 2008 R4.4, 9.1, and 9.2 server.

As a conclusion, CUBRID 9.3 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: 12,591		
Case Name	Path	**Description**
object	sql/_01_object	Performs functionality tests of objects supported by CUBRID, and has the largest number of 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. Includes a test of syntax enhancements, system parameters, 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 sql/_20_sql_extension3	Performs a 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 internationalization features, Test of synta x extension 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.
SQL Extension Optimization (CUBRID 9.3 QA scenario)	sql/_28_features_930 sql/_29_features_845	Performs a test of sql extension, optimization and other features. Test for sql extension includes encryption/decryption functions, SELECT ... FOR UPDATE, DROP SERIAL IF EXISTS, Collation inference for host variables etc. Test for optimization includes Hash Aggregate Evaluation, Analytic Functions, Loose Index Scan, Statistics improvement, 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 has been performed to verify the basic DBMS functionalities using shell scripts. In particular, this test is also to verify CUBRID utilities that cannot be tested by SQL statements. Scenarios of shell scripts are executed to verify DBMS conformity.

Total: 2,061		
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 scenario	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, collation of 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.
SQL Extension Optimization Diagnostics statements (CUBRID 9.3 QA scenario)	shell/_31_features_845 shell/_32_features_930	Performs a test for sql extension, optimization and diagnostics statements features. Test for sql extension includes SELECT ... FOR UPDATE, schema locking, base64 function. Test of optimization includes hash aggregation, statistics, etc. In addition, provide SHOW statements for diagnostics: show log header, show volume header, show heap header, show slotted page, show access, etc.
Manual shell	Manually/*	All the manual test cases which can't be automated or need long time to regress

- HA Feature Test

Total: 246		
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: 12,305		
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: 288		
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: 124		
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

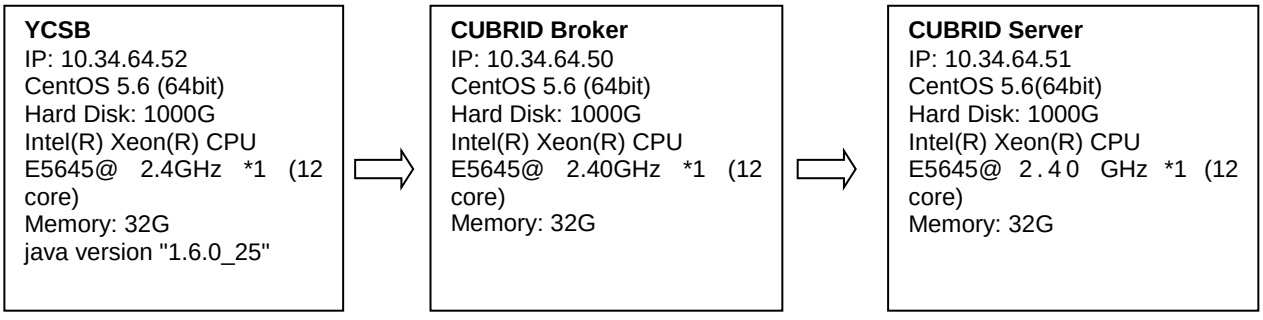
- Measure the number of loops per second.
- For concurrent user tests, add the execution times of all users.

■ 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



✧ 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 INTO usertable(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 = ?;
```

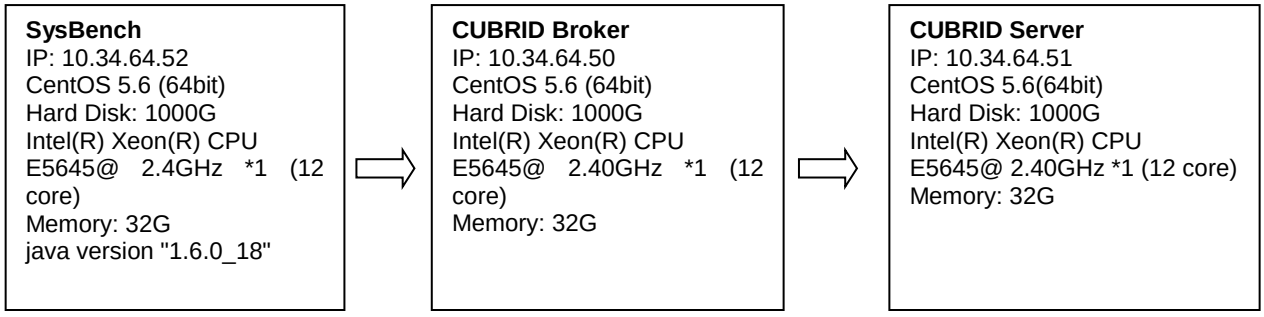
- ✧ Mix operation
  - Select operation: 30%
  - Update operation: 30%
  - Scan operation: 10%
  - Insert operation: 30%

■ SysBench Benchmark

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

■ Test Environment

✧ Test Servers



✧ 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

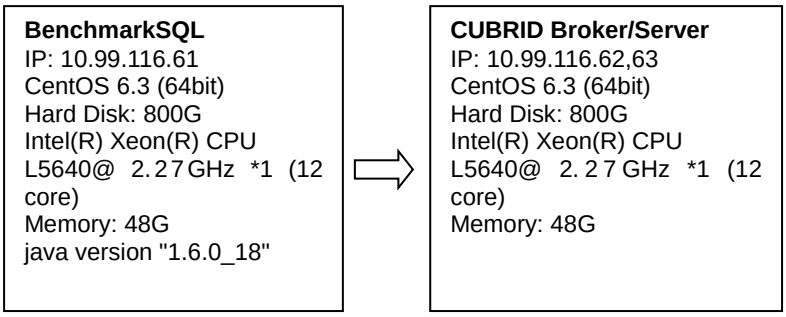
  This test was performed to verify CUBRID performance using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. For more information about NBD Benchmark, see separate documents.

- 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



  - ✧ 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

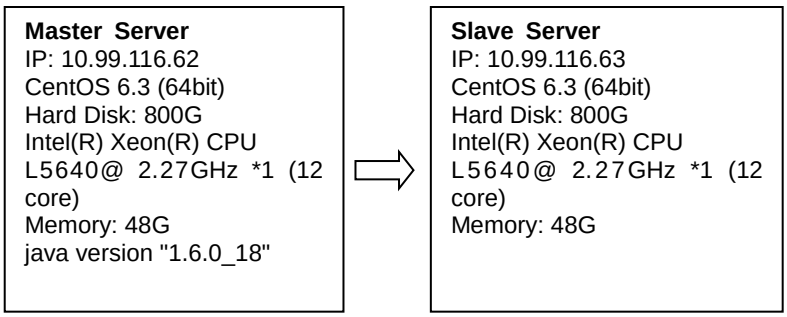
```
Number of warehouses: 10
Number of Terminals: 100
Execute minutes: 30

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



✧ 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_size=1.5G
checkpoint_interval=10min
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

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level	Lines:	235109	307462	76.5 %
Test:	Code Coverage	Functions:	11428	12956	88.2 %
Date:	2014-05-09				
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %				

Directory	Line Coverage		Functions	
/home/bu1/build/src/executables	74.3 %	463 / 623	100.0 %	15 / 15
/home/bu1/build/src/parser	94.0 %	1116 / 1187	100.0 %	10 / 10
external/include	23.8 %	10 / 42	21.1 %	4 / 19
src/base	78.2 %	15229 / 19469	91.0 %	947 / 1041
src/broker	66.6 %	15099 / 22655	83.0 %	964 / 1162
src/cci	75.5 %	5842 / 7735	81.4 %	394 / 484
src/communication	72.9 %	6900 / 9463	79.0 %	328 / 415
src/connection	73.6 %	2884 / 3917	86.6 %	265 / 306
src/executables	72.7 %	13552 / 18653	84.8 %	713 / 841
src/heaplayers	52.5 %	256 / 488	49.2 %	62 / 126
src/heaplayers/util	100.0 %	5 / 5	100.0 %	2 / 2
src/isp	78.2 %	898 / 1148	97.1 %	67 / 69
src/object	76.1 %	23646 / 31077	88.4 %	1729 / 1955
src/optimizer	89.2 %	10118 / 11345	98.5 %	402 / 408
src/parser	84.9 %	47084 / 55445	93.9 %	1509 / 1607
src/query	76.8 %	42948 / 55900	92.4 %	1590 / 1721
src/session	79.3 %	836 / 1054	94.9 %	74 / 78
src/storage	72.7 %	26229 / 36073	88.0 %	1236 / 1405
src/thread	74.4 %	1347 / 1810	90.5 %	105 / 116
src/tools	65.3 %	231 / 354	87.5 %	21 / 24
src/transaction	70.4 %	20416 / 29019	86.0 %	991 / 1152

V. JDBC Code Coverage Results

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	100	79%	8685/10969	67%	2873/4261	3.176
cubrid.jdbc.driver	57	83%	5378/6439	72%	1528/2106	2.588
cubrid.jdbc.jci	37	71%	2920/4078	62%	1262/2034	4.967
cubrid.jdbc.log	2	92%	64/69	92%	12/13	1.393
cubrid.jdbc.net	1	64%	75/117	42%	17/40	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