CUBRID 2008 R4.3 QA Completion Report

This document is theverification report of CUBRID 2008R4.3 in terms of functionality, performance and stability.

Table of Contents

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 2008 R4.3 (hereinafter referred to as R4.3), which is under development for release in November2012, and to determine its release based on the test results. To test the stability of CUBRID, test environmentswere configured as described below. Based on comparisons between the performance test results of CUBRID 2008 R4.3 and those of CUBRID 2008 R4.1 Patch 7 (hereinafter referred to as R4.1 P7), we have tested to determine whether the performance ofR4.3hasimproved or not.

- CentOS 5.6 (32/64-bit) or compatible
- CentOS 5.3 (32/64-bit) or compatible
- CentOS4.7 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 8.4.3.0150(Linux 64-bit/32-bit, Windows 64-bit/32-bit)

1.2 Test Environment

1.2.1 Test Procedures

Tests to verify the CUBRID product are shown below. The test sequence used may be different from the one described here. To verify 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.

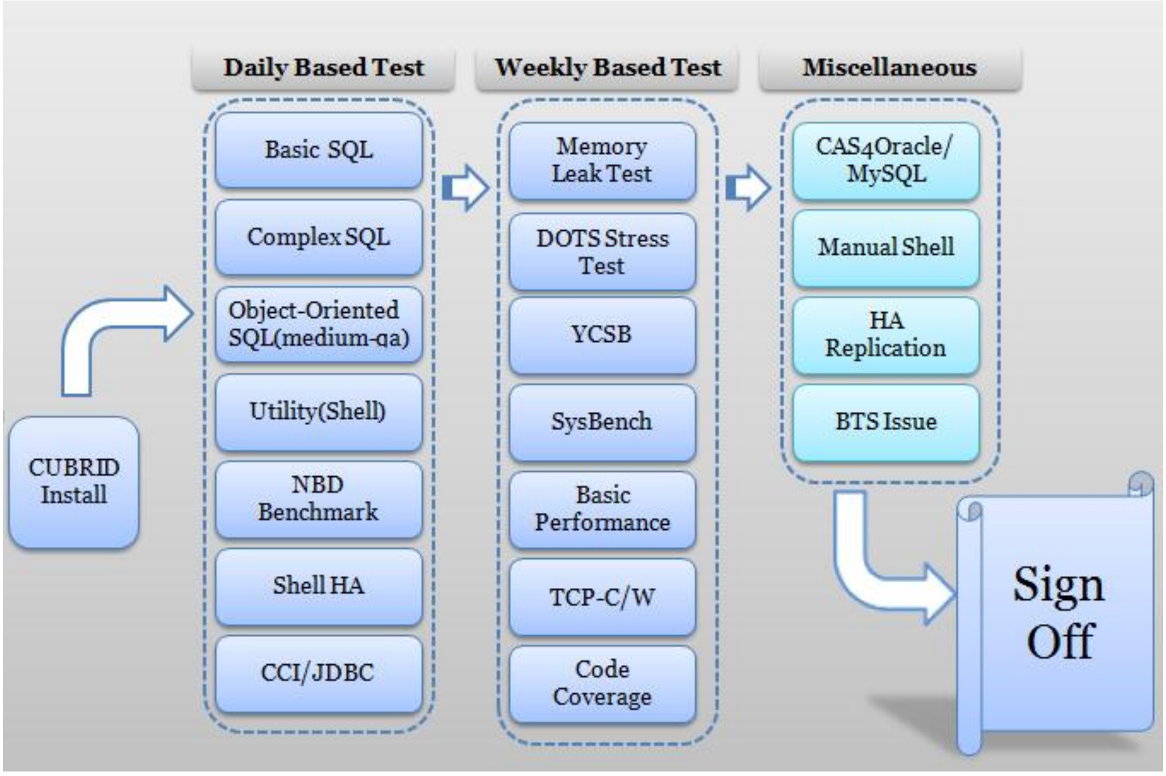


Figure 1. CUBRID Test Procedure

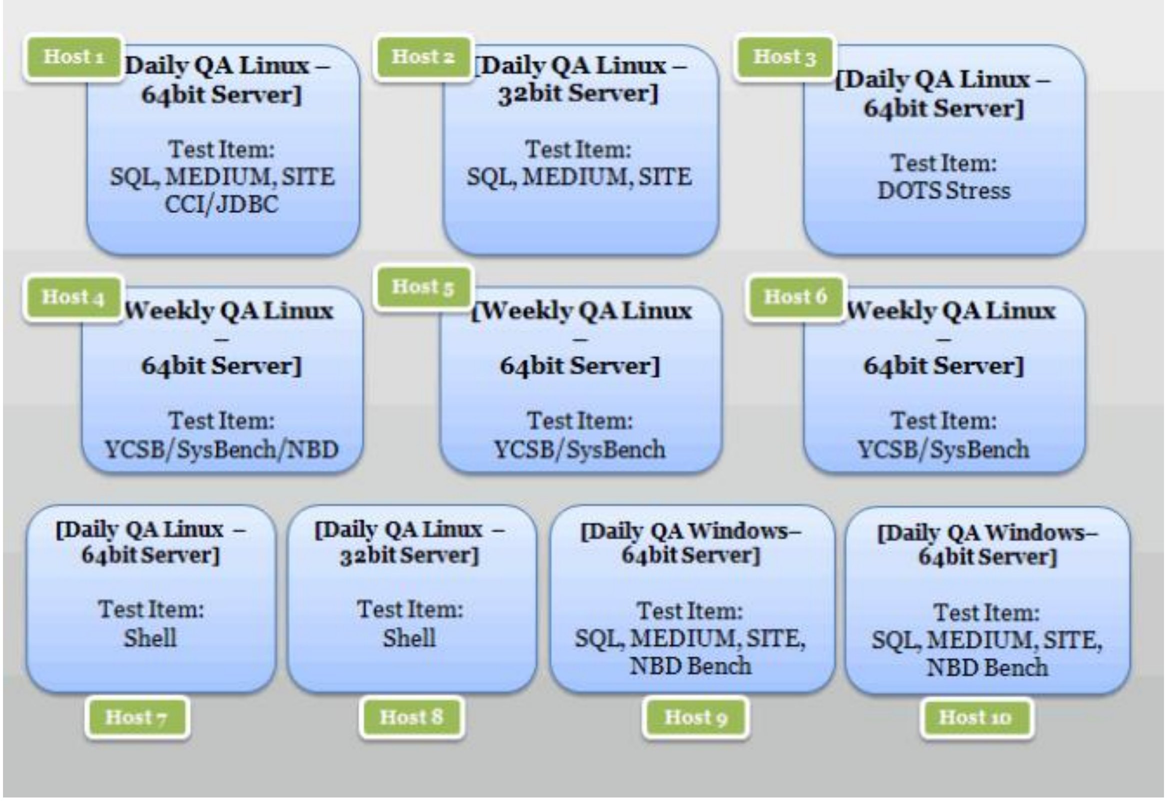

Host 1
Daily QA Linux – 64bit Server]
Test Item:
SQL, MEDIUM, SITE
CCI/JDBC
Host 2
[Daily QA Linux – 32bit Server]
Test Item:
SQL, MEDIUM, SITE
Host 3
[Daily QA Linux – 64bit Server]
Test Item:
DOTS Stress
Host 4
Weekly QA Linux – 64bit Server]
Test Item:
YCSB/SysBench/NBD
Host 5
[Weekly QA Linux – 64bit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

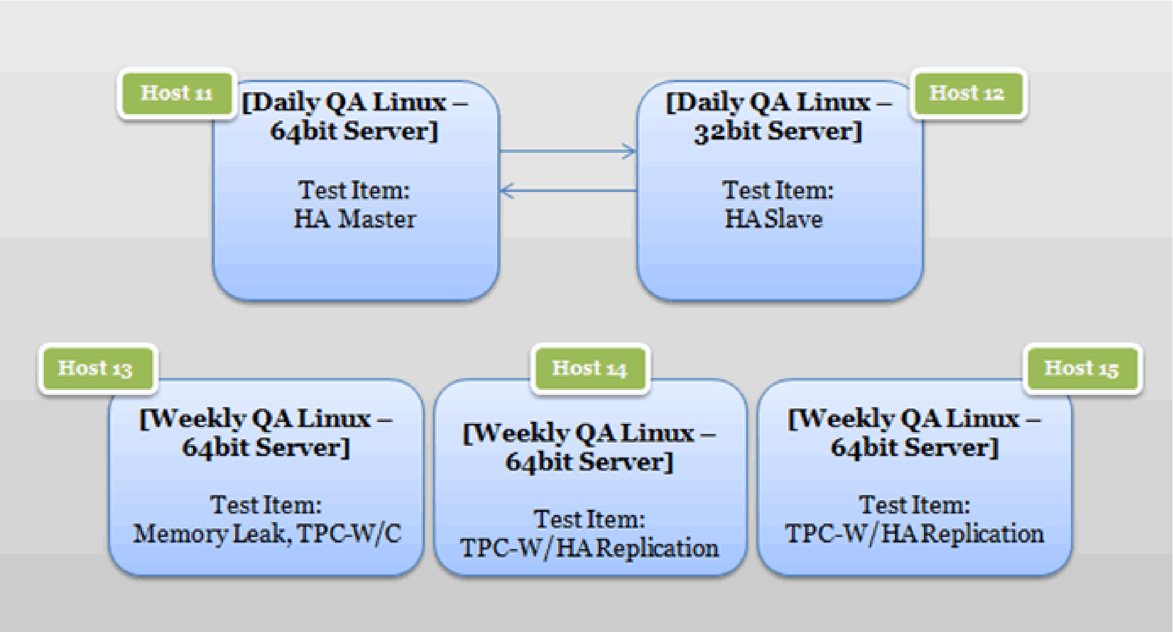


Figure 3. System Diagram for HA Test

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	24 GB	SAS 600G * 3 (Raid5)
Host 5	Cent OS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 GB	SAS 600G * 3 (Raid5)
Host 6	Cent OS 5.6 (64-bit)	Xeon(R) 2.4 GHz (12 cores) * 1	24 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 (quadcores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 10	Windows 2003 (32-bit)	Xeon 2.0 GHz (quadcores) * 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 4.7 (64-bit)	Xeon 2.00 GHz (8 cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 14	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 cores) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 15	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 cores) * 2	8 GB	SATA 500G * 2 (No Raid)

1.3 Test Category

The following tests were performed to determine whether R4.3 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 Replicationtest
    - CCIInterface test
    - JDBC Interface test
    - CAS4MySQL/Oracle
- Performance tests
    - Basic Performance Test
    - YCSB Benchmark
    - SysBench
    - NBD Benchmark
    - TPC-C
- 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 R4.3 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 was performed to verify the basic DBMS functionalities using SQL statements. SQL statements stored in 10,970 files have been executed to verify DBMS conformity. We have executed the stored SQL statements in a JDBC-based application and compared the results withthe 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	8,787	8,787	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 was performed to verify the basic DBMS functionalities using shell scripts. In particular, this test was also performed to verify CUBRID utilities that could not be tested by SQL statements. Scenarios of713shell 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	141	141	100%
Bug regression	401	401	100%
Environment variable	7	7	100%
Other	164	164	100%

2.1.3 HA Feature Tests

Scenarios of 267 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
Data replication test	5	5	100%
Bug regression	115	115	100%
Node fault test	16	16	100%
Process fault test	8	8	100%
Broker fault test	8	8	100%
Run replication test scenarios	115	115	100%

2.1.4 HA Replication Tests

HA Replication Test is a new QA tool which runs SQL test cases on HA Master, and then verifies data consistency between Master and Slave. Scenarios of 8,787SQL files have been executed to verify 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	8,787	8,787	100%
Bug regression	0	0	100%

2.1.5 CCI Interface Tests

CCI Interface Test is to verify if all the CCI APIsof CUBRID can work well as described in the CUBRID manual. Scenarios of 208 shell scripts have been executed to verify all the CCI APIs and the regressions.

Table 5.Result of CCI Interface Tests

Test Category	Number of Scenario Files	Number of Scenario Filespassed	Pass Rate
Basic features	188	188	100%
Bug regression	20	20	100%

2.1.6 JDBC Interface Tests

Scenarios of 1,476 shell scripts have been executed to verify all the JDBC APIs and the regressions.

Table 6.Result of JDBC Interface Tests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
Features test	1,476	1,476	100%

2.1.7 CAS4MySQL/Oracle Tests

Scenarios of 64 shell scriptshave been executed to verify the each features of CAS4MySQL and CAS4Oracle.

Table 7.Result of CAS4MySQL/Oracle Tests

Test Category	Number of Scenario Files	Number of Scenario Filespassed	Pass Rate
CAS4MySQL	30	30	100%
CAS4Oracle	34	34	100%

2.2 Performance Test Results

2.2.1 CUBRID Basic Performance Test

This test was performed to check the performance of the CUBRID DBMS basic operations, which are select, insert, update and delete. For more information about test scenarios, see the appendixII. For all configuration variables, except for SQL_LOG=OFF in cubrid_broker.conf, default configuration values were used. As shown in the table below,we can find that the performance of UPDATE and DELETE on Linux 64-bit has shown significant improvement of over 60%. In Linux 32-bit, the performance of all the operations has increased more than 30%. We can certainly say that this is the most significant changes in performance that R4.3 has brought us.

We also identify the performance enhancement of the UPDATE and DELETE operation is strongly related with the workspace of CAS process and it is highly affected by the previous INSERT operations. That means there's no performance enhancement of UPDATE and DELETE operation itself. If the INSERT, UPDATE and DELETE are mixed, there's significant performance improvement.

A. Linux: Performance Comparison between R4.1 P7 and R4.3(64-bit)

We can find that the performance of UPDATE and DELETE operations has shown significant improvement. The performance of UPDATE has increased about 72%, and DELETE increased about 52%. The other operations for INSERT and SELECT have also shown slight improvement.

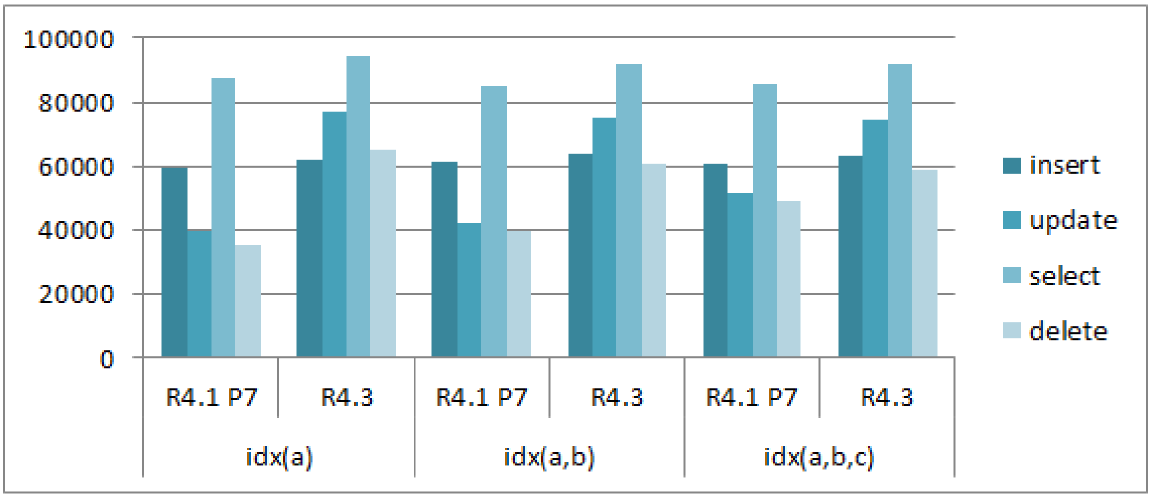


Figure 4. Performance Comparison between R4.1 P7 and R4.3 (Linux 64-bit)

Table 8.Performance Comparison between R4.1P7and R4.3 (Linux 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.1 P7	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**
Insert	59,886	62,219	104%	61,202	64,139	105%	60,627	63,589	105%
Update	39,927	77,047	193%	42,423	75,441	178%	51,739	74,505	144%
Select	87,805	94,630	108%	85,064	92,163	108%	85,557	91,784	107%
Delete	35,380	64,945	184%	39,642	60,576	153%	49,028	58,757	120%
Total	222,998	298,841	134%	228,331	292,319	128%	246,951	288,635	117%

(Unit: TPS)

B. Linux: Performance Comparison between R4.1 P7 (32-bit) and R4.3(32-bit)

We can find that the performance of all the operations has shown great improvement (around 30%). The performance of the UPDATE operationhas increased over 50%.

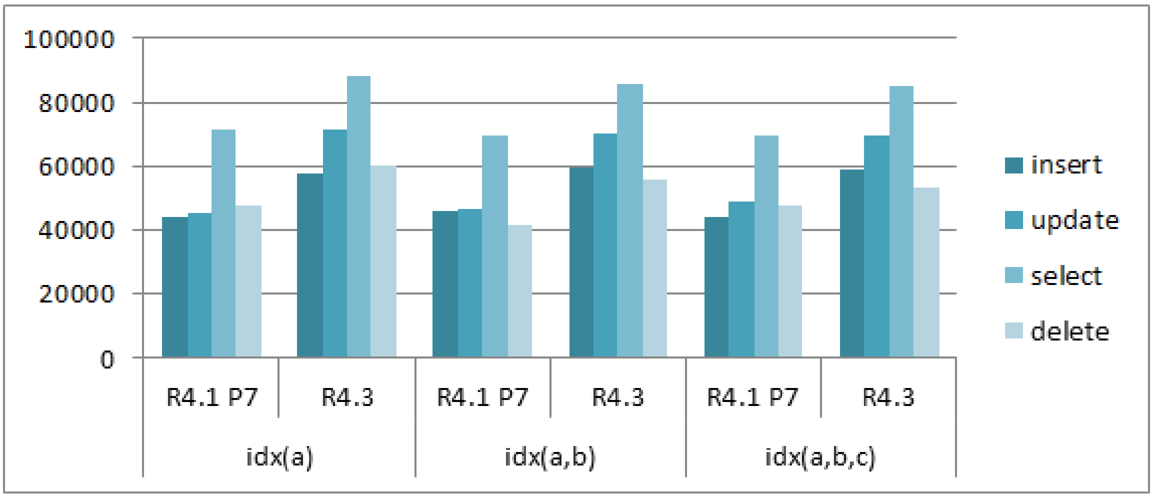


Figure 5. Performance Comparison between R4.1 P7 and R4.3 (Linux 32-bit)

Table 9.Performance Comparison between R4.1 P7 and R4.3 (Linux 32-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.1 P7	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**
Insert	43,995	57,741	131%	45,983	59,475	129%	44,202	59,034	134%
Update	45,240	71,699	158%	46,338	70,361	152%	48,870	69,559	142%
Select	71,343	88,036	123%	69,589	85,581	123%	69,461	85,056	122%
Delete	48,008	60,096	125%	41,853	55,759	133%	47,815	53,583	112%
Total	208,586	277,572	133%	203,763	271,176	133%	210,348	267,232	127%

(Unit: TPS)

C. Windows: Performance Comparison between R4.1P7(64-bit) and R4.3 (64-bit)

According to the test result, we can see that the performance of most operations has shown positive changes.

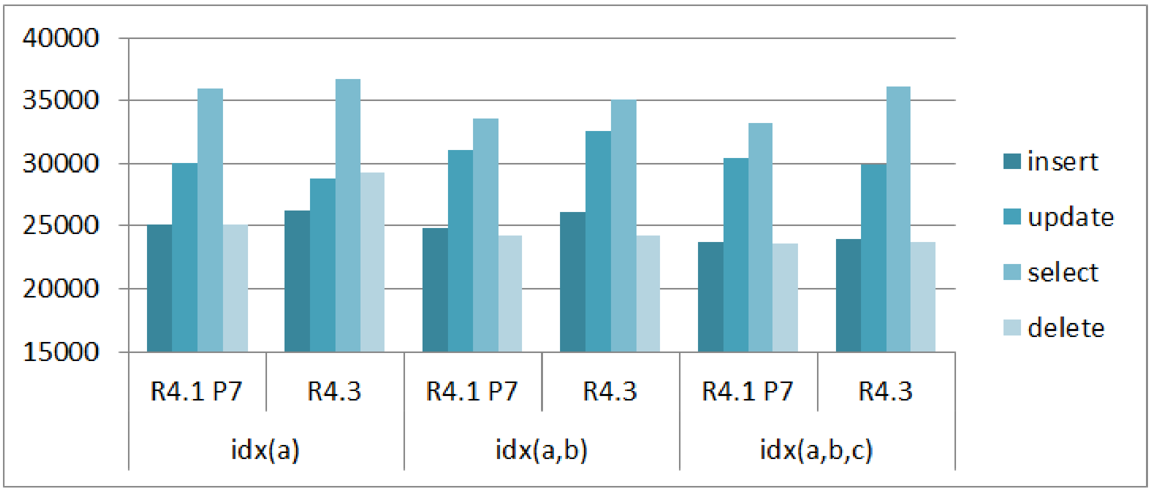


Figure 6. Performance Comparison between R4.1 P7 and R4.3 (Windows 64-bit)

Table 10.Performance Comparison between R4.1 P7 and R4.3 (Windows 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.1 P7	R4.3	Ratio	R4.1 P7	R4.3	Ratio	R4.1 P7	R4.3	Ratio
Insert	25,061	26,320	105%	24,812	26,157	105%	23,688	23,946	101%
Update	30,083	28,739	96%	31,051	32,576	105%	30,401	29,942	98%
Select	35,968	36,730	102%	33,604	35,110	104%	33,221	36,139	109%
Delete	25,156	29,325	117%	24,282	24,200	100%	23,640	23,709	100%
Total	116,268	121,114	104%	113,749	118,043	104%	110,950	113,736	103%

(Unit: TPS)

D. Windows: Performance Comparison between R4.1 P7 (32-bit) and R4.3 (32-bit)

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

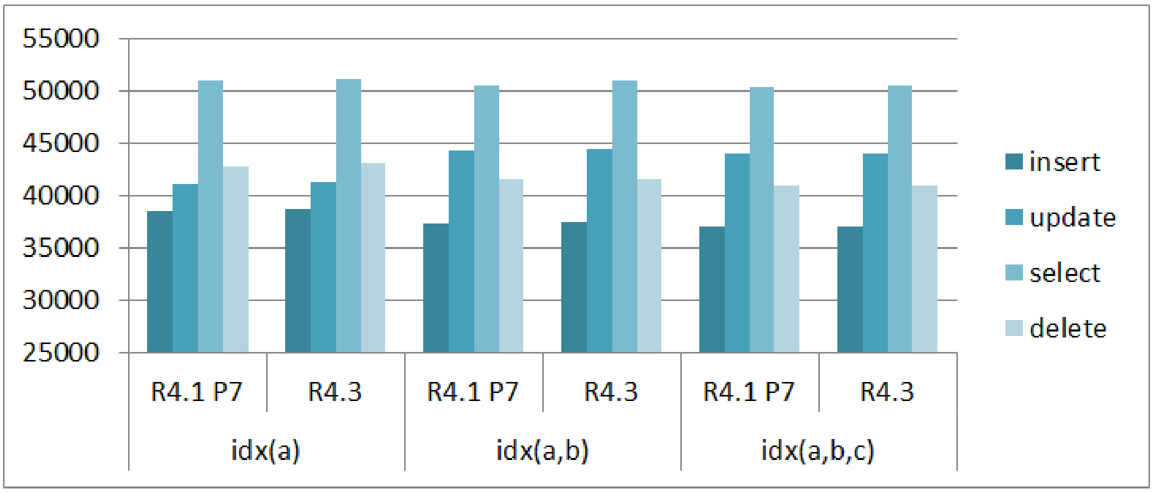


Figure 7. Performance Comparison between R4.1 P7 and R4.3 (Windows 32-bit)

Table 11.Performance Comparison between R4.1 P7 andR4.3 (Windows 32-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.1 P7	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**Ratio**
Insert	38,568	38,666	100%	37,341	37,522	100%	36,977	37,047	100%
Update	41,162	41,211	100%	44,248	44,400	100%	44,038	43,930	100%
Select	50,986	51,151	100%	50,463	50,933	101%	50,309	50,488	100%
Delete	42,837	43,091	101%	41,610	41,539	100%	40,872	40,884	100%
Total	173,553	174,119	100%	173,662	174,394	100%	172,196	172,349	100%

(Unit: TPS)

2.2.2 YCSB Performance Test

YCSB as a framework for benchmarking system is popular in theworld (see also https://github.com/brianfrankcooper/YCSB/wiki). This test wasperformed to verify CUBRID performance of not only basic operations but also compositive operations, which are insert, select, scan, update and the mix ofthem. For more information about test scenarios, see the appendix II. As shown in the results below, the performance for most operations has improved(nearly5%)except scan operation in slave server configuration which reduce slightly.

A. Master Server Configuration:Performance Comparison between R4.1 P7 (64-bit) and R4.3 (64-bit)

Table 12.Result of YCSB Benchmark (Master Server)

	Throughput(OPS)			Average Latency(ms)		95thPercentile Latency(ms)	
Operations	**R4.1 P7**	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**R4.1 P7**	**R4.3**
Insert	13,221	14,261	108%	21	19.54	35	35
Select	24,811	25,066	101%	11.53	11.4	28	28
Scan	4,453	4,510	101%	56	55.8	244	244
Update	12,593	13,452	107%	22.8	21.3	41	21
Mix	12,469	13,617	109%	31.45	28.27	51	45

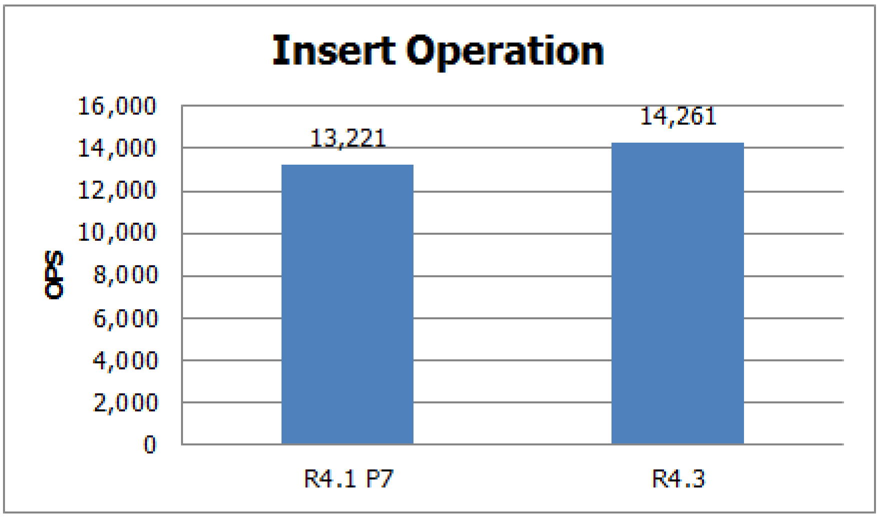


Figure 8. Result of Insert Operation of YCSB Benchmark (Master Server)

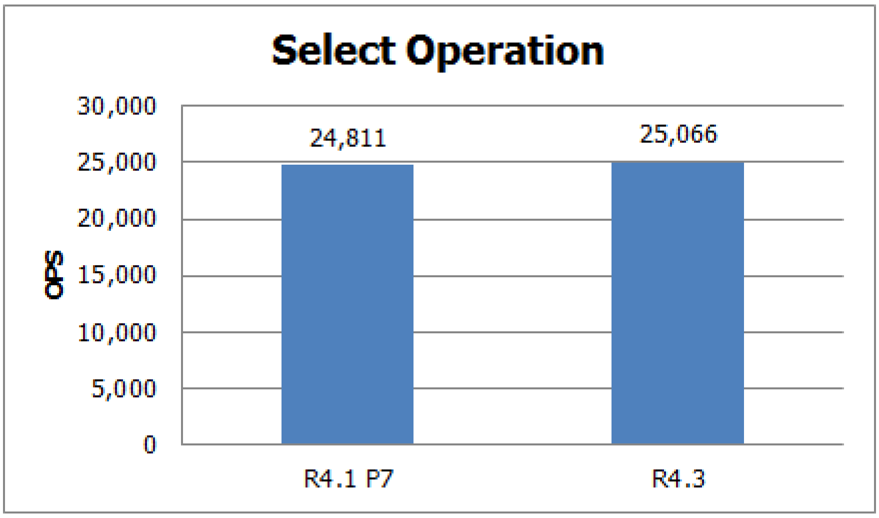


Figure 9. Result of Select Operation of YCSB Benchmark (Master Server)

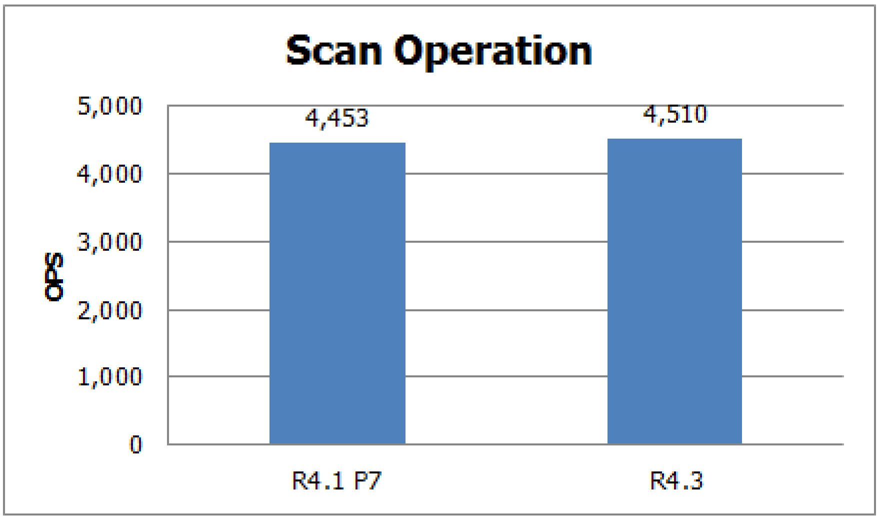


Figure 10. Result of Scan Operation of YCSB Benchmark (Master Server)

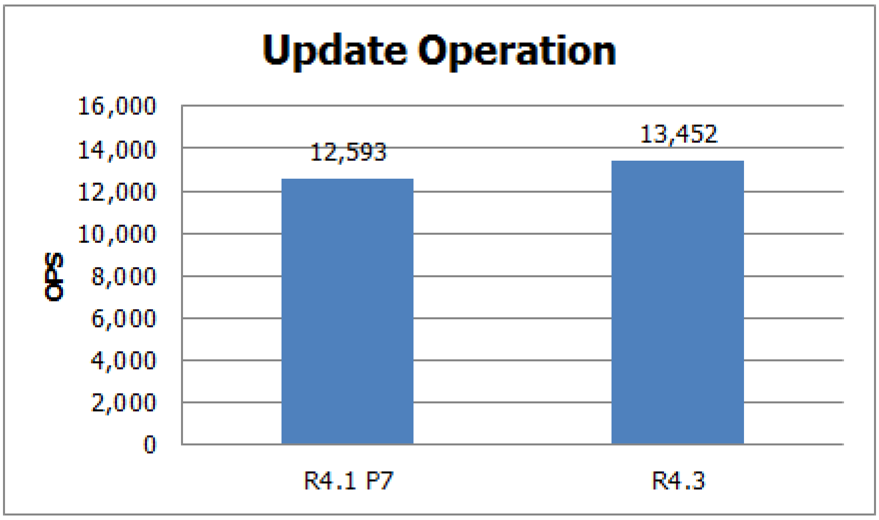


Figure 11. Result of Update Operation of YCSB Benchmark (Master Server)

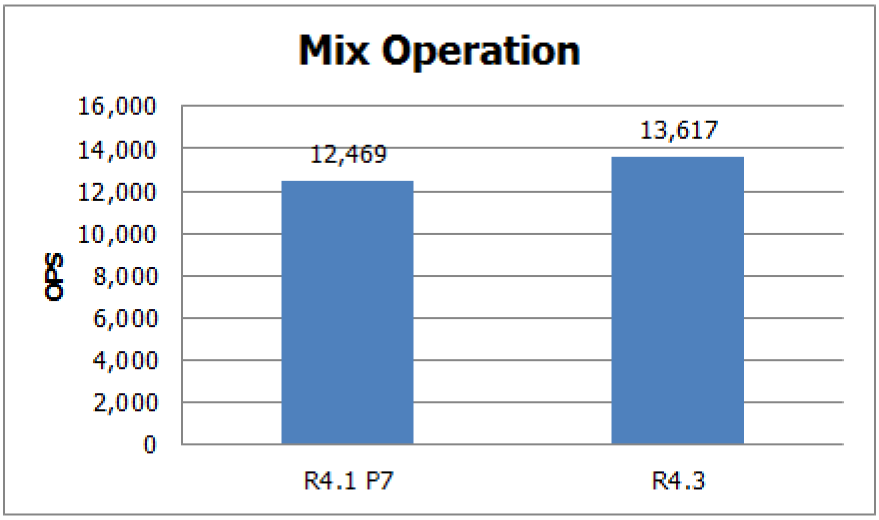


Figure 12. Result of Mixed of YCSB Benchmark (Master Server)

B. Slave Server Configuration: Performance Comparison between R4.1P7 (64-bit) and R4.3(64-bit)

Table 13.Result of YCSB Benchmark (Slave Server)

	Throughput(OPS)			Average Latency(ms)		95thPercentile Latency(ms)	
Operations	**R4.1 P7**	**R4.3**	**Ratio**	**R4.1 P7**	**R4.3**	**R4.1 P7**	**R4.3**
Insert	13,756	14,589	106%	20.28	19	39	37
Select	24,654	25,167	102%	11.6	11.29	28	27
Scan	4,446	4,287	96%	56.5	58.64	243	247
Update	13,460	13,951	104%	21	20.478	17	16
Mix	12,941	13,884	107%	34	27.8	112	45

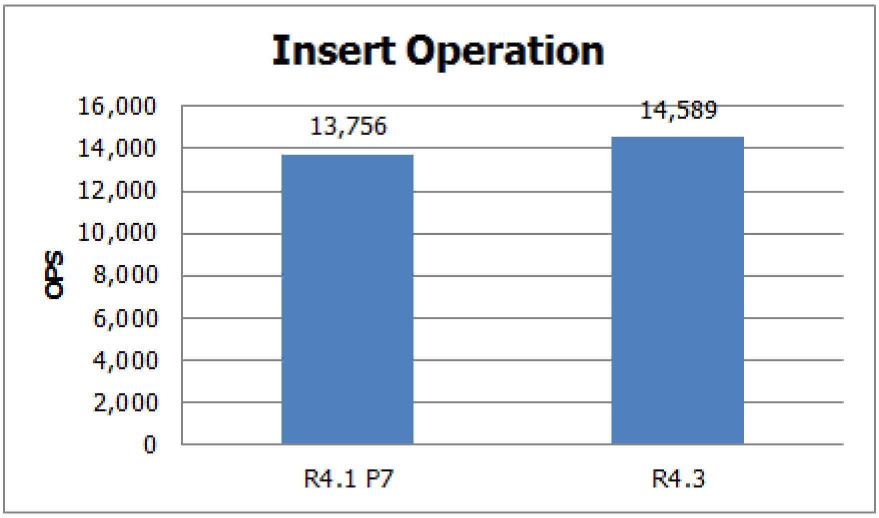


Figure 13. Result of Insert Operation of YCSB Benchmark (Slave Server)

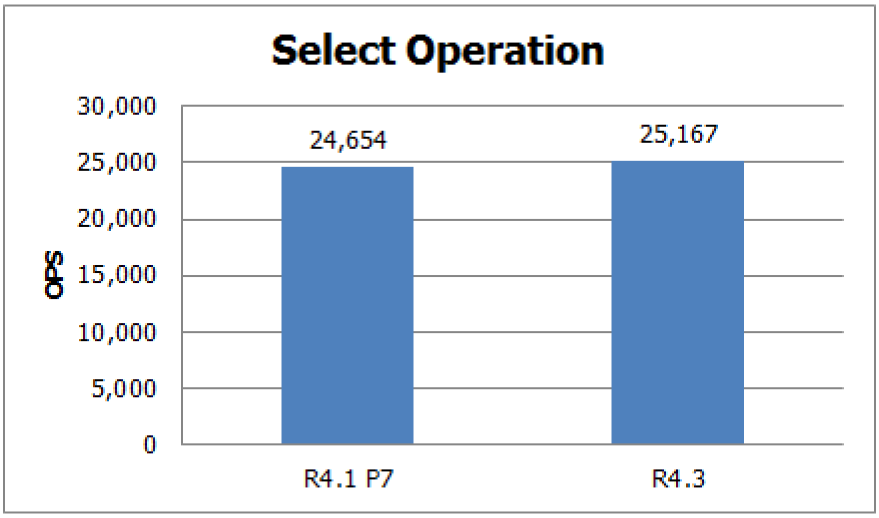


Figure14. Result of Select Operation of YCSB Benchmark (Slave Server)

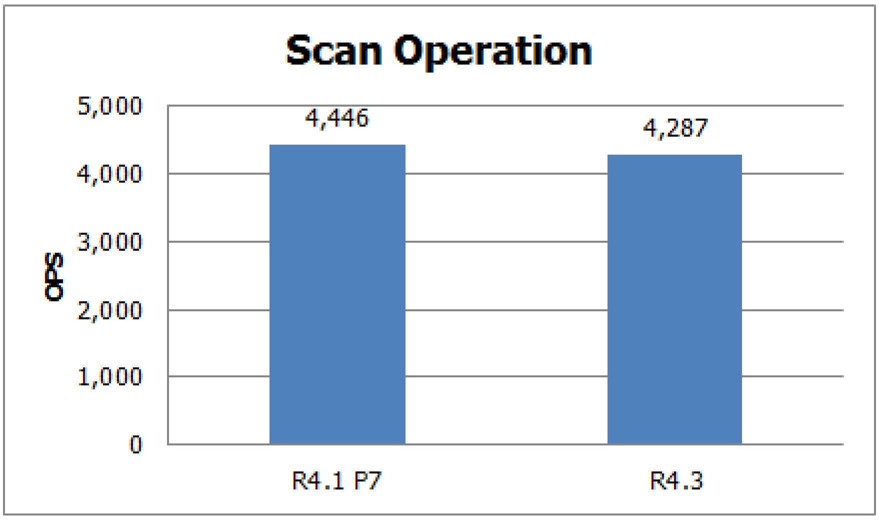


Figure 15. Result of Scan Operation of YCSB Benchmark (Slave Server)

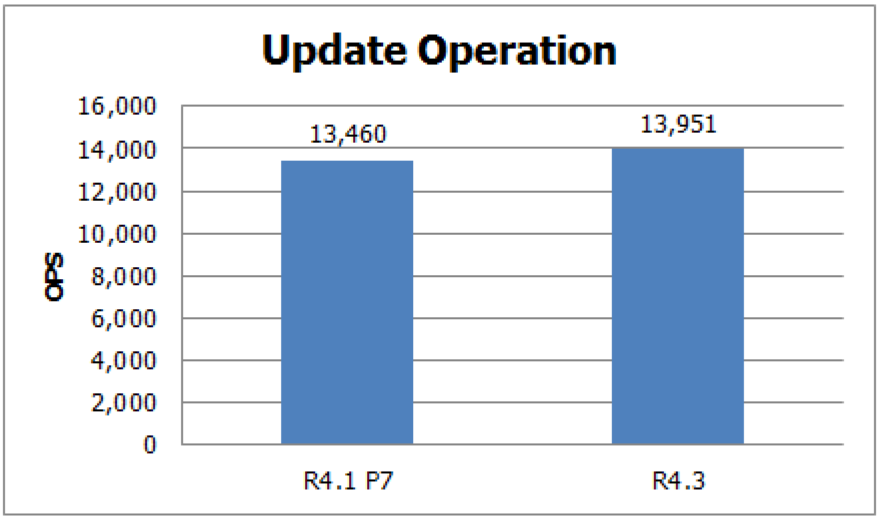


Figure 16. Result of Update Operation of YCSB Benchmark (Slave Server)

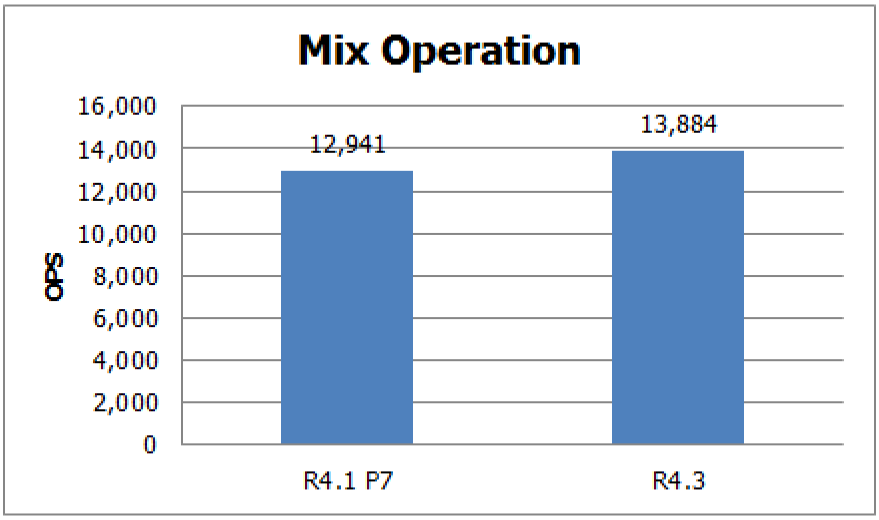


Figure 17. Result of Mixed of YCSB Benchmark (Slave Server)

2.2.3 SysBenchPerformance 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 alsohttp://sysbench.sourceforge.net/).SysBench runs a specified number of threads and they all 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 test scenarios, see the appendix II.

As shown in the results below, the performance of SysBench on R4.3 has slightly improvedon the number of read/write requests(per sec), the average execution time of per request and the number of transactions.

A. SysBench performance comparison between R4.1P7 (64-bit) and R4.3 (64-bit)

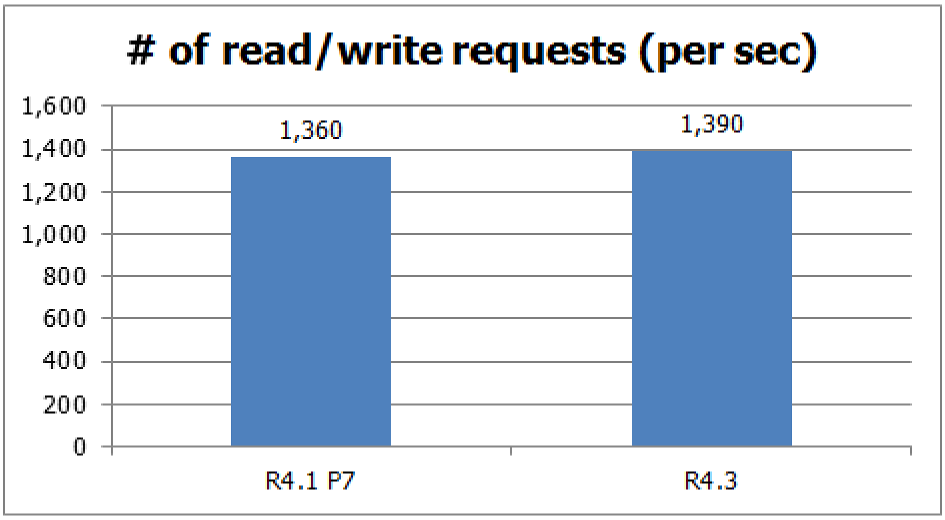


Figure 18.The number of read/write requests per second of SysBench benchmark

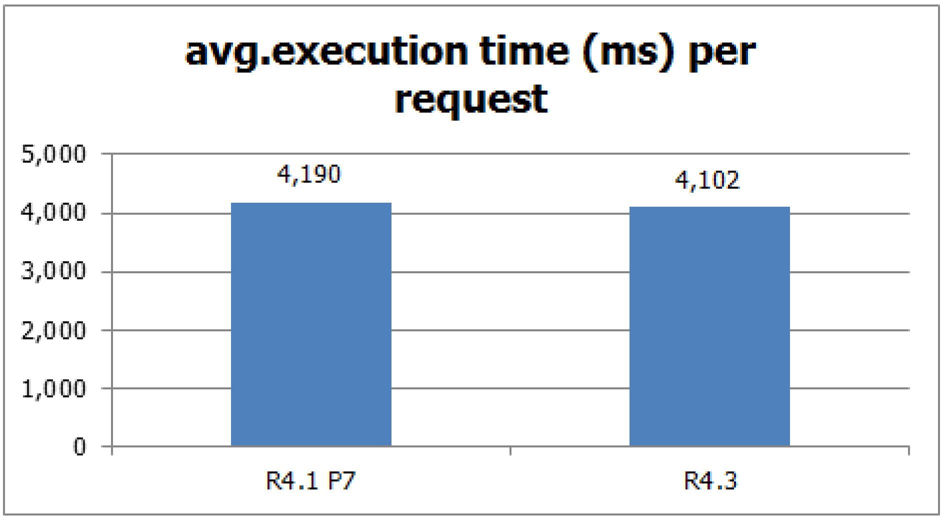


Figure 19.The average execution time per request of SysBench benchmark

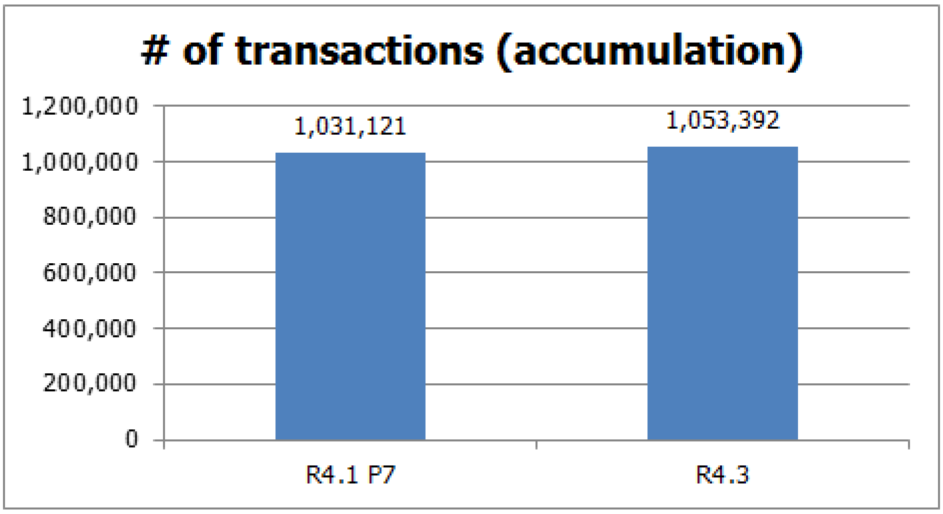


Figure 20. Theaccumulatednumber of transactionsof SysBench benchmark

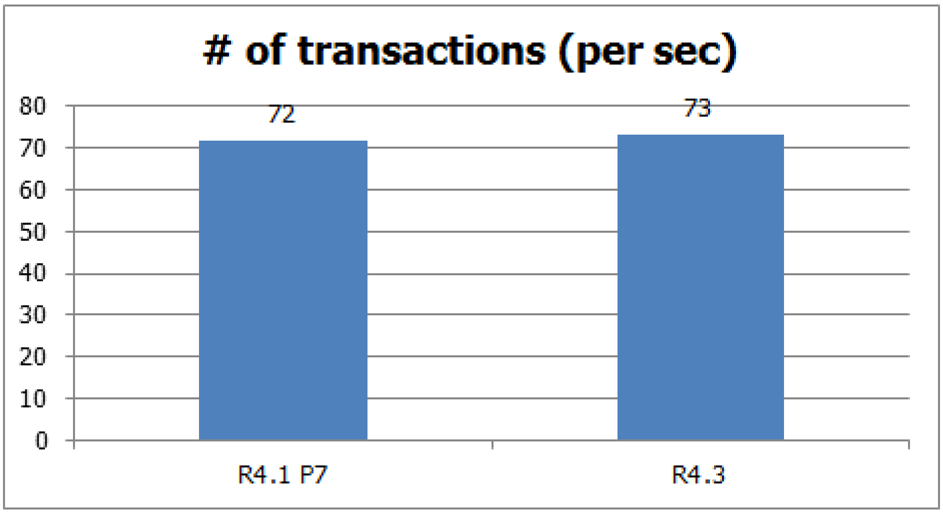


Figure 21.The number of transactions per secondofSysBench benchmark

2.2.4 NBD Benchmark Performance Test

This test was performed to verify 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 Viewsof R4.3has no significant changethan that of R4.1 P7.

A. NBD performance comparison between R4.1 P7 (64-bit)and R4.3(64-bit)

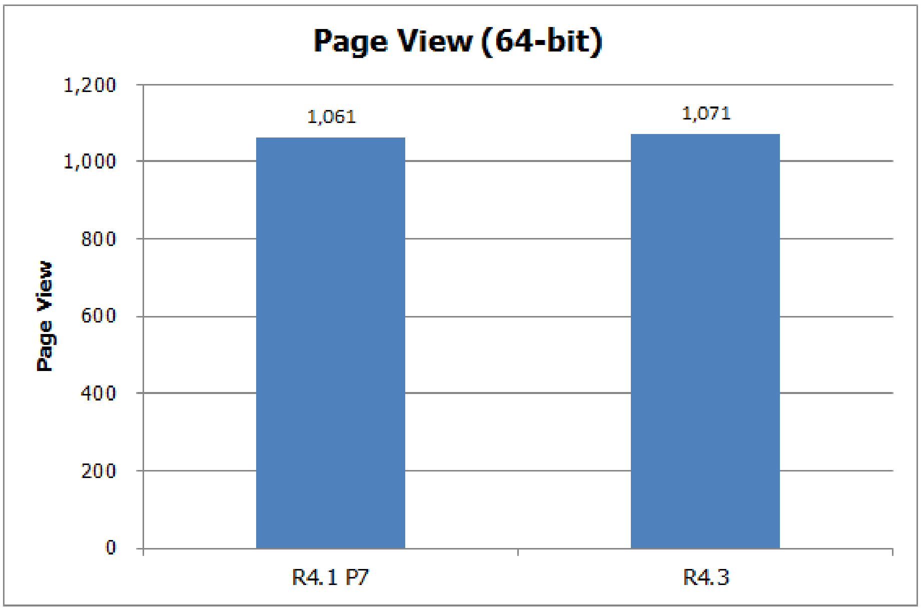


Figure 22.NBD performance comparison (64-bit)

B. NBD performance comparison between R4.1P7 (32-bit) and R4.3(32-bit)

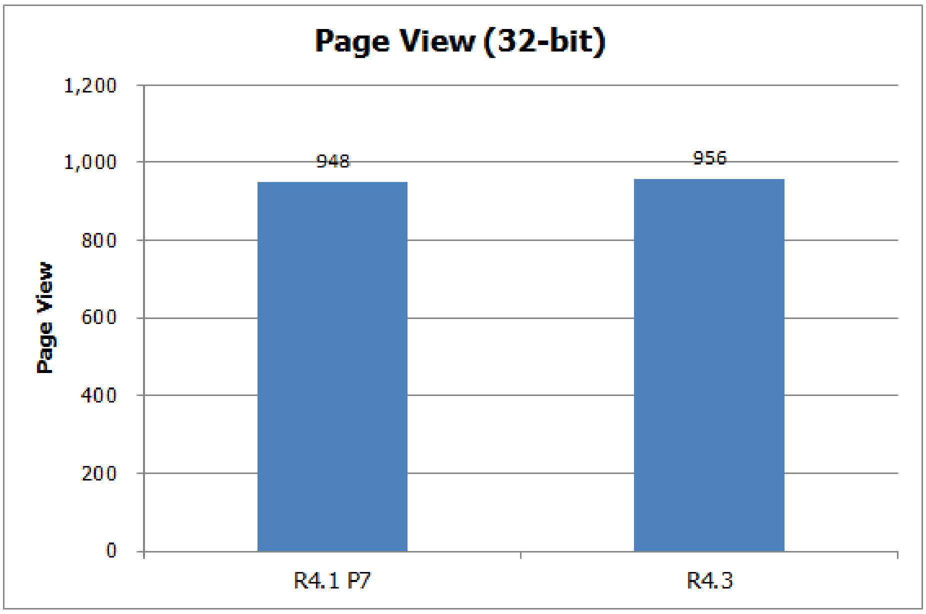


Figure 23.NBD performance comparison (32-bit)

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

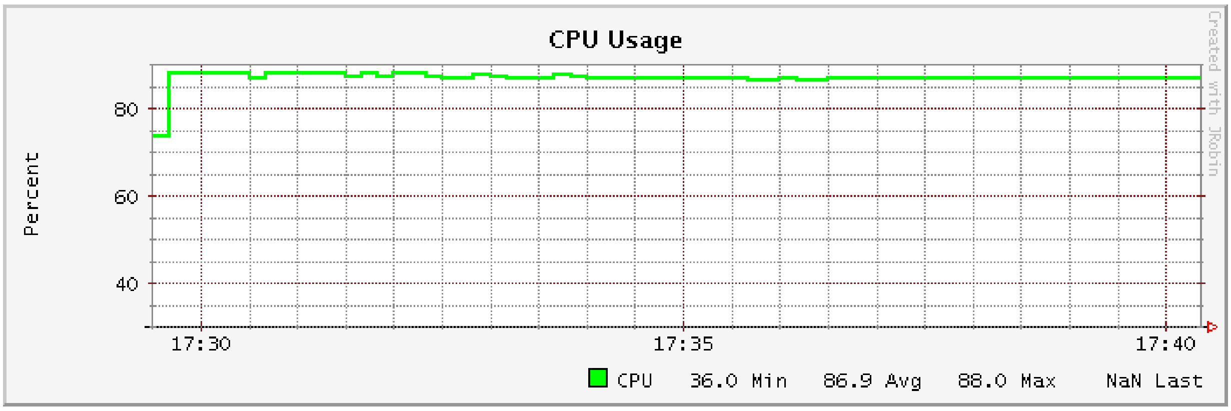


Figure 24.CPU Usage for NBD Benchmark

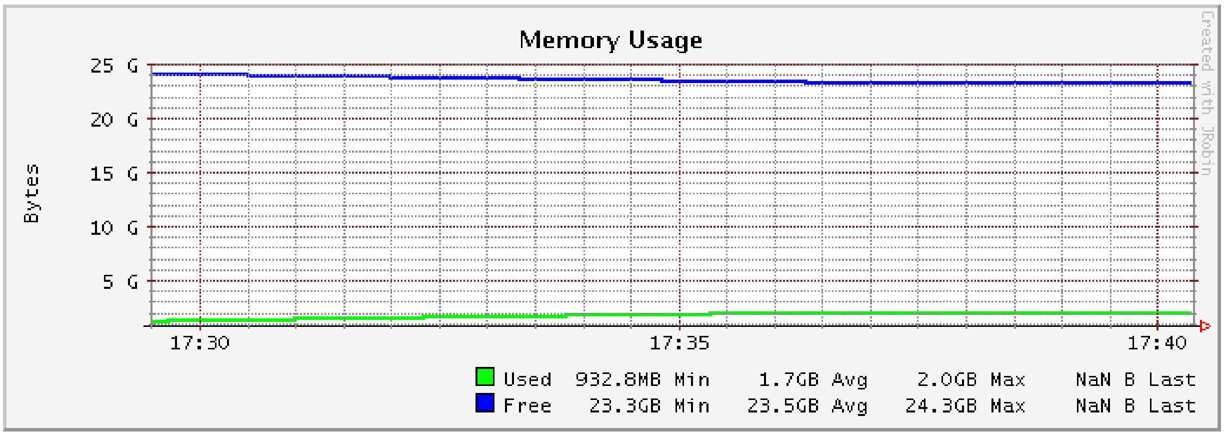


Figure 25.Memory Usage for NBD Benchmark

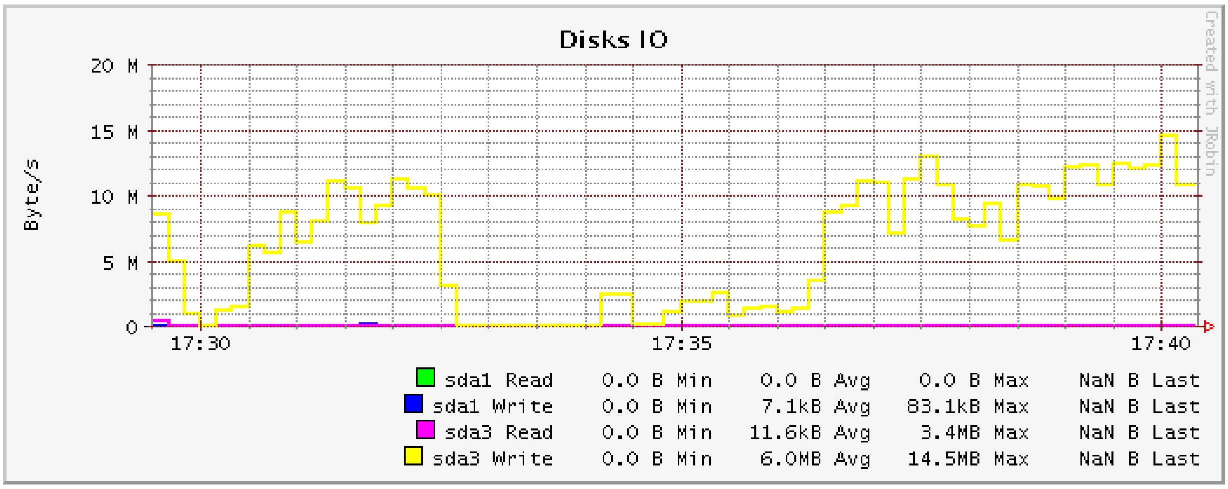


Figure 26.DisksIO status for NBD Benchmark

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 alsohttp://www.tpc.org/tpcc/)is more complex than previous OLTP benchmarks such as TPC-A because of its multiple transaction types, more complex 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 R4.3 has slightly improvedon tpmC.

A. TPC-C performance comparison between R4.1 P7 (64-bit)and R4.3 (64-bit)

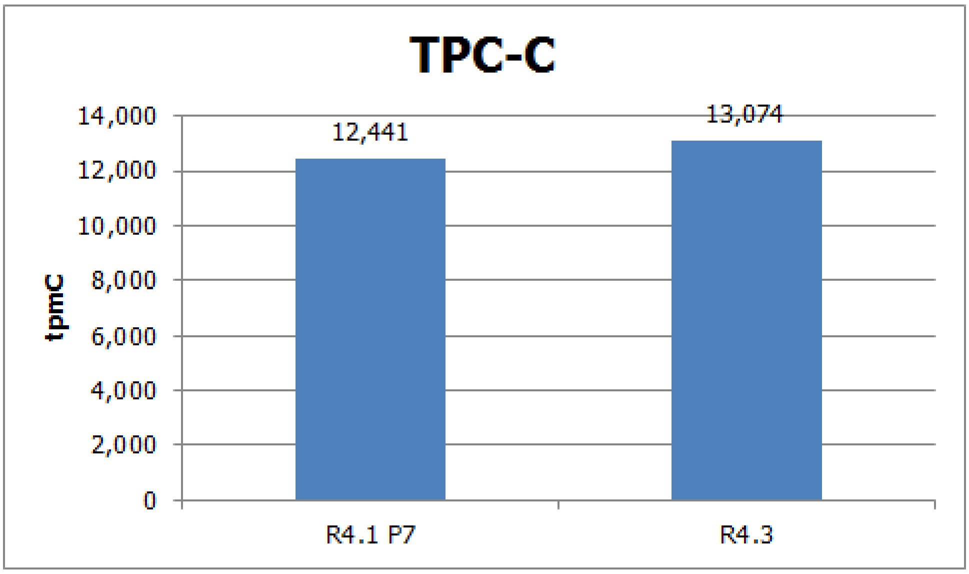


Figure 27.tpmC comparison of TPC-C benchmark

2.3 Stability Test Results

DOTS, a sub-project of an open project called "Linux Test Project," is an open test tool for testing the DBMS. For more information about DOTS, see the appendix III. As shown in the test results below, the system operated stably without any abnormalities during 24hours. You can ignore the failures because they are unique violations due to the modification of duplicated data.

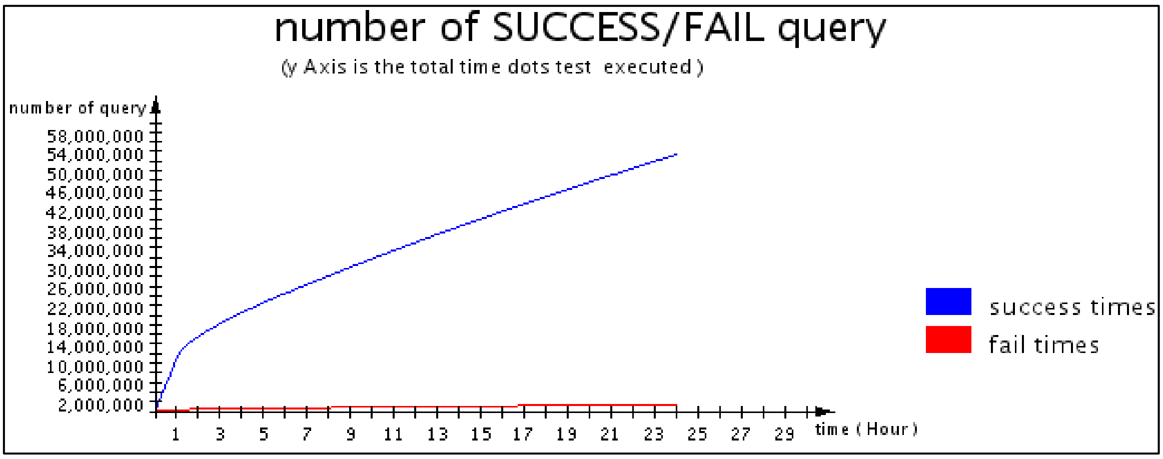


Figure 28.The number of SUCCESS/FAIL Queries of DOTS Test

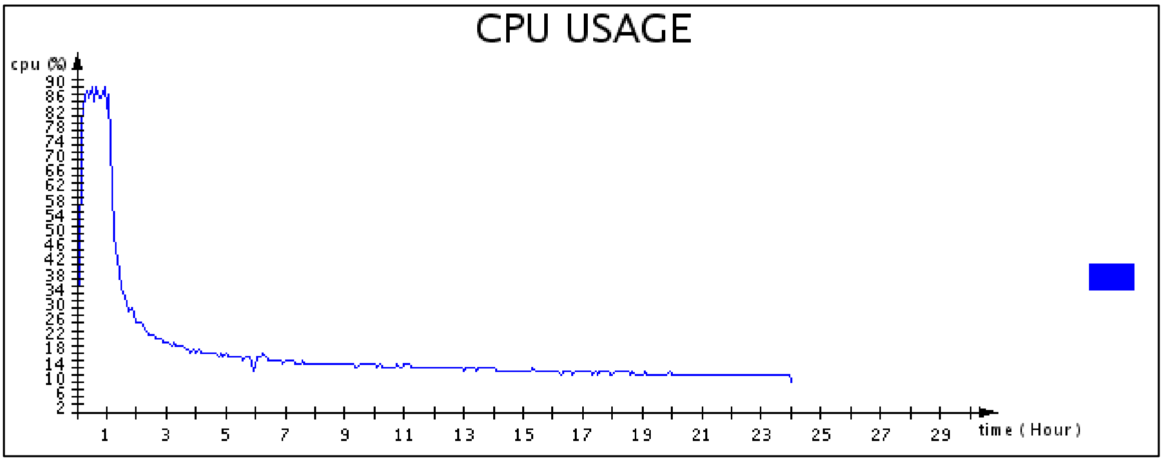


Figure 29. CPU Usage of DOTS Test

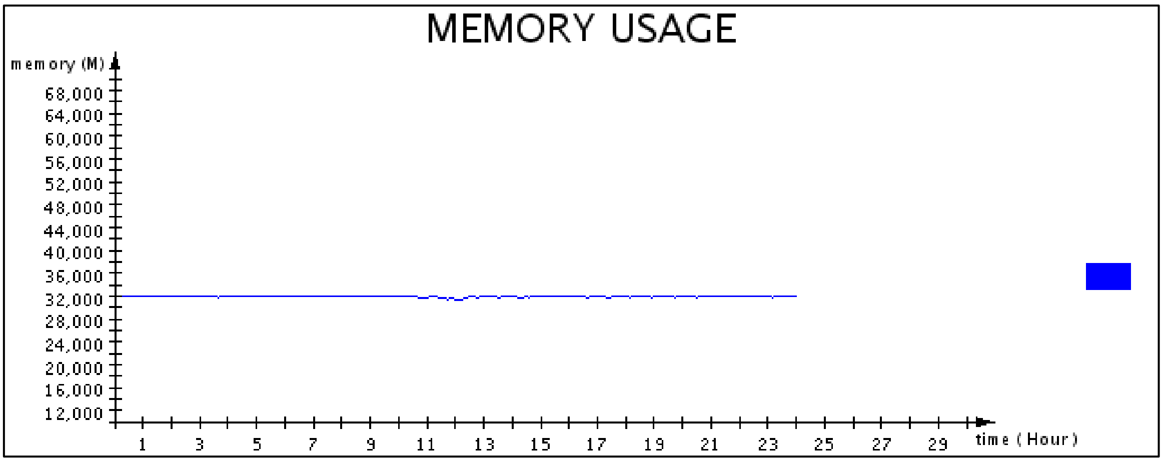


Figure 30.Memory Usage of DOTS Test

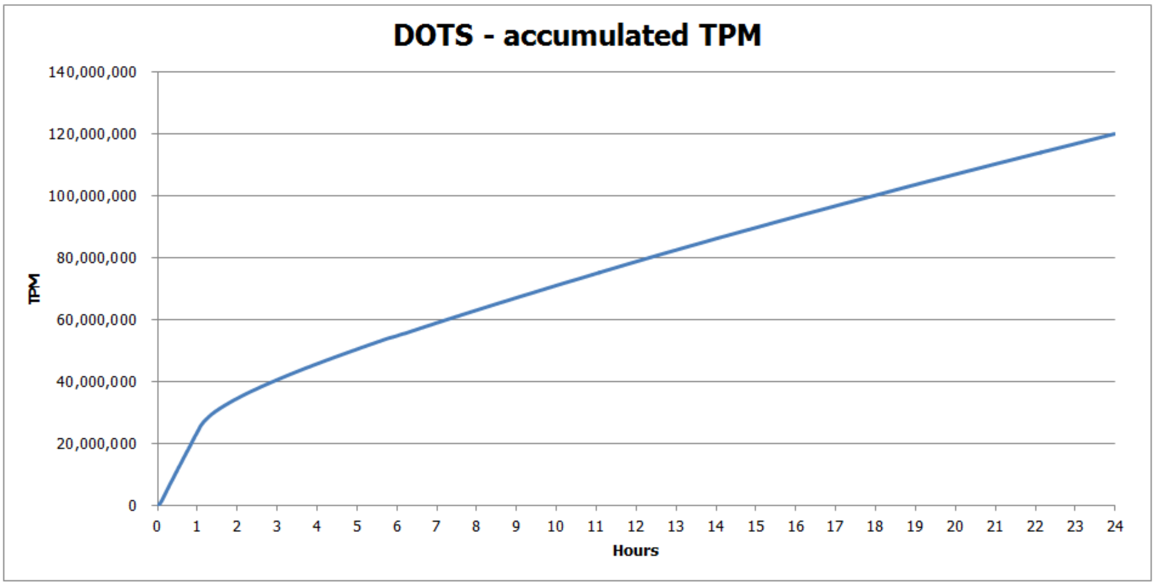


Figure 31.The accumulated TPM of DOTS Test

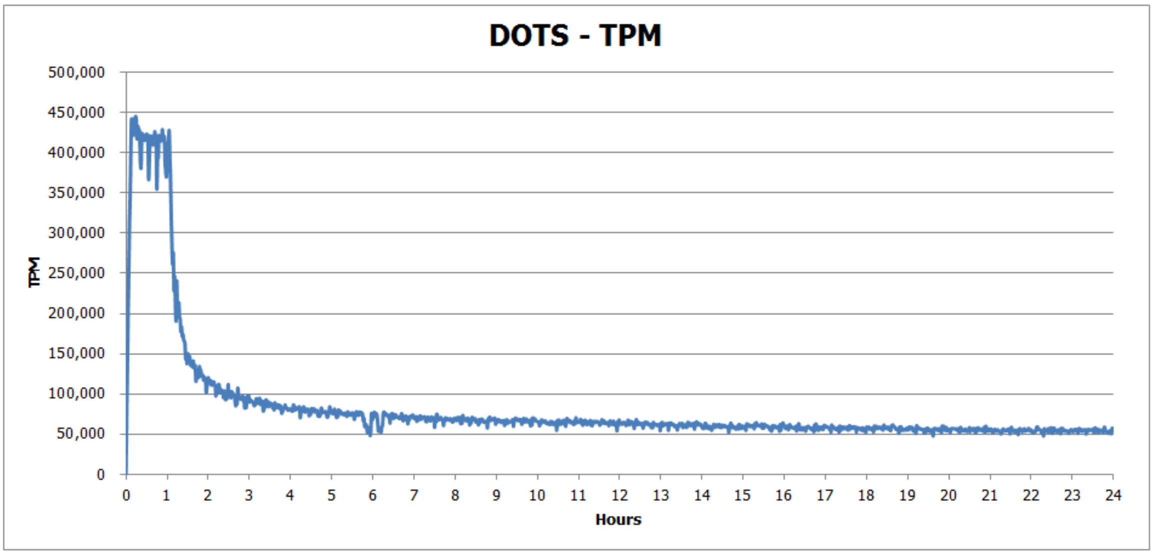


Figure 32. TPM of DOTS Test

2.4 Compatibility Test Results

This test was performed to verify the JDBC and CCI compatibility between R4.1 P7 and R4.3.
SQL, MEDIUM and JDBC Unit Testswere executed to verify JDBC compatibility. Shell test cases for CCI were executed to verify CCI compatibility.

Table 14.Result of JDBC CompatibilityTests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
R4.1 P7 JDBC -> R4.3 Server	11,233	11,233	100%
R4.3 JDBC -> R4.1 P7 Server	11,199	11,199	100%

Table 15.Result of CCI CompatibilityTests

Test Category	Number of Scenario Files	Number of Scenario Files passed	Pass Rate
R4.1 P7 CCI -> R4.3 Server	208	208	100%
R4.3 CCI -> R4.1 P7 Server	194	194	100%

2.5 Installation Test Results

Installation test was performed based on 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 16.Result of Installation Test

Package Type	Test OS	Result
RPM/SH/TAR.GZ	LinuxCentOS 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 7on 32-bit and 64-bit Windows XP on 32-bit	PASS

2.6 Other Test Results

The entire bug and issue fixes for R4.3 have been confirmed.

2.7 Quality Index

The standard quality index of R4.3is listed below.

Table 17.Quality Index of R4.3

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.	1,200,254 LOC
			Total number of source code lines in the changed files	1,200,254 LOC
QA Scenario Code Coverage	76%	74.9%	Number of tested statements	183,008
			Total number of statements	244,483
Fault Density Detected by Static Analysis	4 /KLOC	3.74 /KLOC	Number of faults detected by static analysis (Level 1)	252
			Number of faults detected by static analysis (Level 2)	17
			Number of faults detected by static analysis (Level 3)	549
			Number of faults detected by static analysis (Level 4)	0
			Total number of source code lines	848,847LOC
Cyclomatic Code Complexity	3.3%	2.7%	Number of modules whose complexity is over 30	573
			Total number of modules in a project	20,923
	12%	15.3%	Number of modules whose complexity is over 10	3,201
			Total number of modules in a project	20,923

3.Conclusions

As describedin Chapter 1 and 2, all the test cases for functions have been regressed, and the scenarios forperformance, stability, compatibility, installation and other testshave also been successfully executed before the release of R4.3. 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 fromthe basic performance test, we can find that the performance of UPDATE and DELETE on Linux 64-bit has shownsignificant improvement of over60%. In Linux 32-bit, the performance of all the operations has increased more than30%. We can certainly say that this is the most significant changes in performance that R4.3 has brought us.

We also identify the performance enhancement of the UPDATE and DELETE operation is strongly related with the workspace of CAS process and it is highly affected by the previous INSERT operations. That means there's no performance enhancement of UPDATE and DELETE operation itself. If the INSERT, UPDATE and DELETE are mixed, there's significant performance improvement.

For YCSB, we can see thatthe performance for most operations has improved(nearly 5%)except scan operation in slave server configuration which has slightly reduced.

For SysBench, according to the test result,the performance has very slightimprovement (nearly 2%).

For NBD, there are no significant change for performance of Page View.

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

For stability test with DOTS, according to the two graphs with TPM and its accumulation, it looks quite stable even after 24 hours of execution, and by examining the resource usage and other health data, no notable issues have been found.

From the result of compatibility test, we can reach the conclusion that JDBC and CCI on R4.1 P7have compatibility with R4.3server, and JDBC and CCI on R4.3 also have compatibility with R4.1 P7 server except some known issues.

As a conclusion, CUBRID 2008 R4.3 meets the criteria of release.

Appendix

I. Functionality Test Scenarios

This test was performed to verify the basic DBMS functionalities using SQL statements. SQL statements stored in files have be en executed to verify DBMS conformity. We executed the stored SQL statements in a JDBC-based application, and compared the results to the stored reference file 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: 10,970		
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
Codecoverage	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 aunit test of the syntax extension 2.
FBO	sql/ _15_fbo	Performs a test of the FBO feature.
Index enhancement	sql/ _16_index_enhancement	Performs aunit 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.
MySQLcompatibility for NEWS service	sql/_22_news_service_mysql_compatibility	Performs a test of several functions, regular expression and hint rewriting.

- 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: 713		
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.
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.
MySQL compatibility	shell/_23_mysql_compatibility	Includes scenarios that verify syntax extension, which has been added in CUBRID 2008 R3.1.
Unstable	shell/_25_ unstable	Includes scenarios that are not very stable
Manual shell	Manually/*	All manual test cases which can't be automized or need long time to regress

- HA Feature Test

Total: 267		
Case Name	**Path**	**Description**
Data replication test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed in a normal state with no fault.
Node fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a node fault occurs during insert/update/delete operations.
Process fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a process fault occurs that causes the database process to stop during insert/update/delete operations.
Broker fault test	execp/UsualCase	Includes scenarios that check whether HA replication is properly performed when a broker fault occurs during insert/update/delete operations.
Replication scenario	scripts/sql	Includes scenarios that test whether HA is working properly for each CUBRID transaction type, and has two sub directories: random_case and special_case

Bug regression	HA/shell/	Includes scenarios that verify there is no regression by checking the HA bug fixes in CUBRID

- HA Replication test

Total: 8,787		
Case Name	**Path**	**Description**
Test Cases migrated fr om SQL suite	N/A	Migrated existing SQL suite into HA environment. Execute them on m aster node, then check whether be replicated to slave or not.
Bug Regression	HA/shell/_24_fun ctional_repl/	Includes scenarios that verify there is no regression by checking the HA bug fixes in CUBRID

- CCI Interface test

Total: 208		
Case Name	**Path**	**Description**
Features test	Interface/shell/_2 0_cci	Which contains CCI all APIs, each APIs are mentioned in manual are tested in shell scripts
Bug Regression	Interface/shell/_2 0_cci/_12_issue	Includes shell scripts which are written when verify CCI bts issues

- JDBC Interface test

Total: 1,476		
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: 64		
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

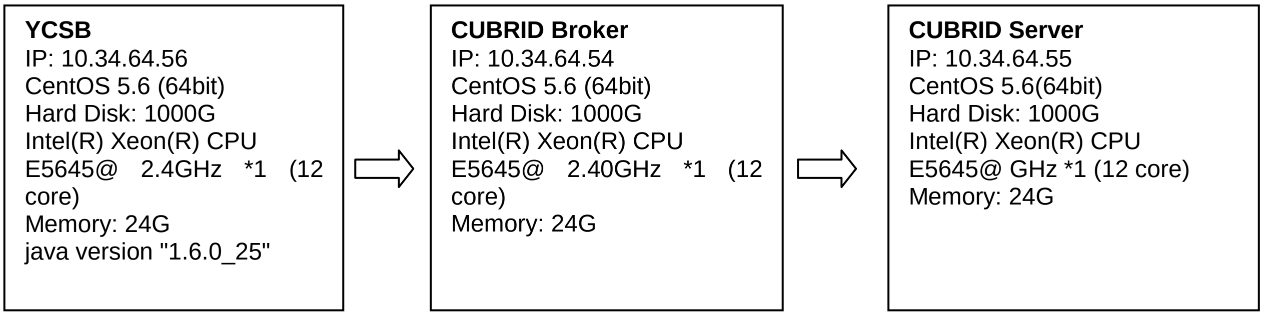
- Measure the number of loops per second.
- For concurrent user tests, add the execution times of all users.

■ YCSB Benchmark

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

■ Common Test Environment

✧ Test Servers



YCSB
IP: 10.34.64.56
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.4GHz *1 (12 core)
Memory: 24G
java version "1.6.0_25"

CUBRID Broker
IP: 10.34.64.54
CentOS 5.6 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ 2.40GHz *1 (12 core)
Memory: 24G

CUBRID Server
IP: 10.34.64.55
CentOS 5.6(64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5645@ GHz *1 (12 core)
Memory: 24G

✧ CUBRID database volume configuration

```
cubridcreatedbycsb
cubridaddvoldb -p data --db-volume-size=2G ycsb -S
cubridaddvoldb -p data --db-volume-size=2G ycsb -S
cubridaddvoldb -p index --db-volume-size=2G ycsb -S
cubridaddvoldb -p index --db-volume-size=2G ycsb -S
cubridaddvoldb -p temp --db-volume-size=2G ycsb –S
```

✧ Configuration for CUBRID

- cubrid_broker.conf:

```
SERVICE                         =ON
BROKER_PORT                     =33000
MIN_NUM_APPL_SERVER             =5
MAX_NUM_APPL_SERVER             =300
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  =  ?;
```

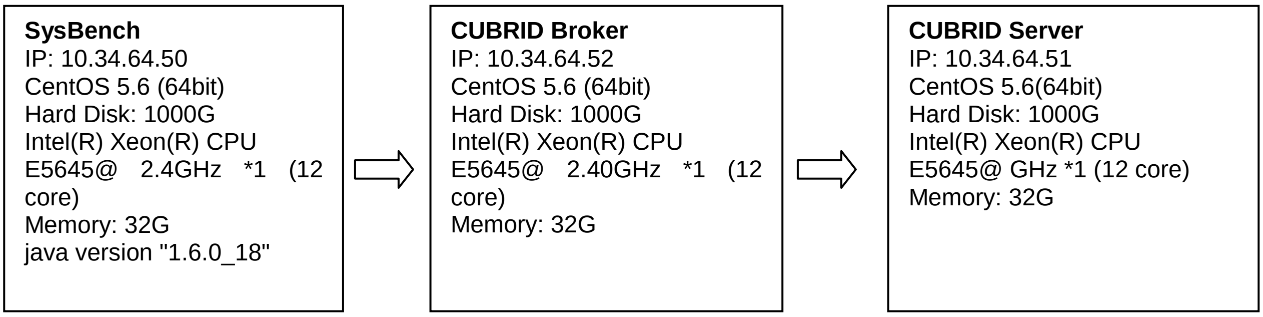
- ✧ Mix operation
  - Select operation: 30%
  - Update operation: 30%
  - Scan operation: 10%
  - Insert operation: 30%

■ SysBench Benchmark

This test wasperformed to verify CUBRID performance based on OLTP business.

■ Test Environment

✧ Test Servers



SysBench
IP: 10.34.64.50
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: 10.34.64.52
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@ GHz *1 (12 core)
Memory: 32G

✧ CUBRID database volume configuration

```
cubridcreatedbsysbench
cubridaddvoldb -p data --db-volume-size=2G sysbench -S
cubridaddvoldb -p data --db-volume-size=2G sysbench -S
cubridaddvoldb -p index --db-volume-size=2G sysbench -S
cubridaddvoldb -p temp --db-volume-size=2G sysbench  -S
```

✧ Configuration for CUBRID

- cubrid_broker.conf:

```
SERVICE                        =ON
BROKER_PORT                      =33000
MIN_NUM_APPL_SERVER                =350
MAX_NUM_APPL_SERVER                =350
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,
      kINTEGER 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=10.34.64.52 □
              --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

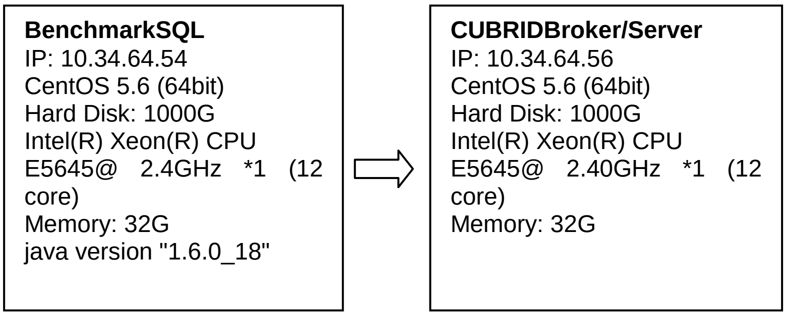
This test was performed to verify CUBRID performance using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. For more information about NBD Benchmark, see separate documents.

- 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 toexecute 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 (Revision: 22,174)

- Test Environment
  - ✧ Test Servers



BenchmarkSQL
IP: 10.34.64.54
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"

CUBRIDBroker/Server
IP: 10.34.64.56
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 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          =5
MAX_NUM_APPL_SERVER          =200
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%
```

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
```

- 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

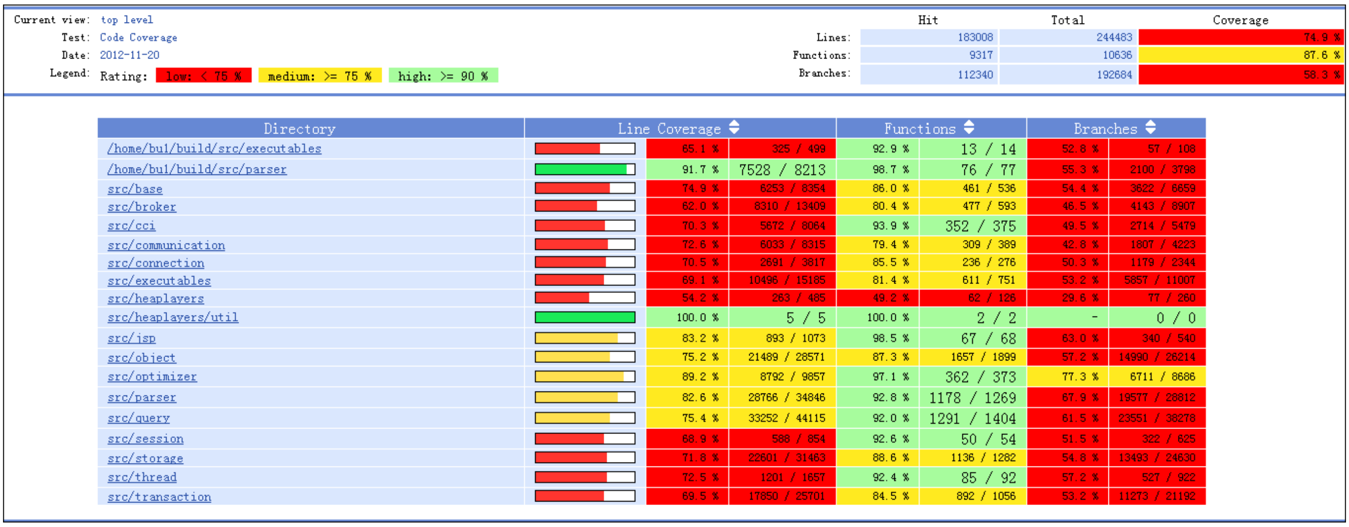
```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID =
UPDATE ITEM SET DESCRIPTION = ?,BID_PRICE = ?,START_TIME = ?,END_TIME = ? WHERE ITEMID = ?
```

- 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.

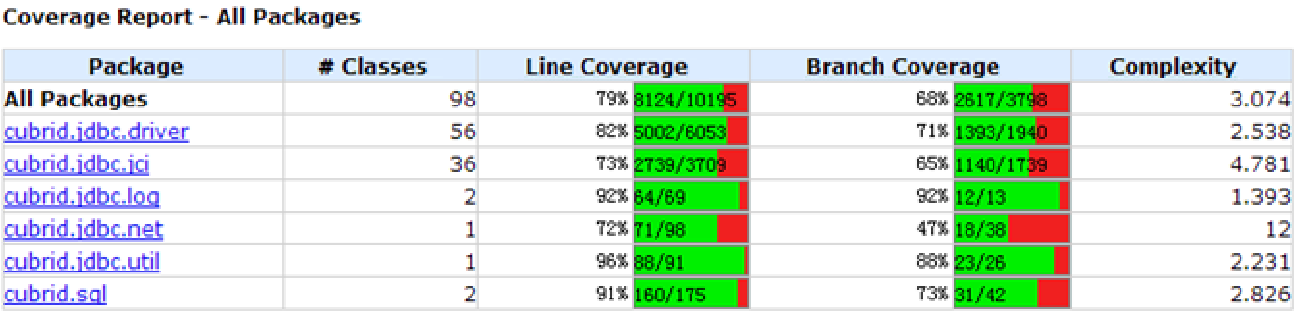
IV. Scenario-based Code Coverage Results



			Hit	Total	Coverage
Current view:	top level				
Test:	Code Coverage	Lines:	183008	244483	74.9 %
Date:	2012-11-20	Functions:	9317	10636	87.6 %
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %	Branches:	112340	192684	58.3 %

Directory	Line Coverage			Functions		Branches	
/home/bu1/build/src/executables		65.1 %	325 / 499	92.9 %	13 / 14	52.8 %	57 / 108
/home/bu1/build/src/parser		91.7 %	7528 / 8213	98.7 %	76 / 77	55.3 %	2100 / 3798
src/base		74.9 %	6253 / 8354	86.0 %	461 / 536	54.4 %	3622 / 6659
src/broker		62.0 %	8310 / 13409	80.4 %	477 / 593	46.5 %	4143 / 8907
src/cci		70.3 %	5672 / 8064	93.9 %	352 / 375	49.5 %	2714 / 5479
src/communication		72.6 %	6033 / 8315	79.4 %	309 / 389	42.8 %	1807 / 4223
src/connection		70.5 %	2691 / 3817	85.5 %	236 / 276	50.3 %	1179 / 2344
src/executables		69.1 %	10496 / 15185	81.4 %	611 / 751	53.2 %	5857 / 11007
src/heaplayers		54.2 %	263 / 485	49.2 %	62 / 126	29.6 %	77 / 260
src/heaplayers/util		100.0 %	5 / 5	100.0 %	2 / 2	-	0 / 0
src/jsp		83.2 %	893 / 1073	98.5 %	67 / 68	63.0 %	340 / 540
src/object		75.2 %	21489 / 28571	87.3 %	1657 / 1899	57.2 %	14990 / 26214
src/optimizer		89.2 %	8792 / 9857	97.1 %	362 / 373	77.3 %	6711 / 8686
src/parser		82.6 %	28766 / 34846	92.8 %	1178 / 1269	67.9 %	19577 / 28812
src/query		75.4 %	33252 / 44115	92.0 %	1291 / 1404	61.5 %	23551 / 38278
src/session		68.9 %	588 / 854	92.6 %	50 / 54	51.5 %	322 / 625
src/storage		71.8 %	22601 / 31463	88.6 %	1136 / 1282	54.8 %	13493 / 24630
src/thread		72.5 %	1201 / 1657	92.4 %	85 / 92	57.2 %	527 / 922
src/transaction		69.5 %	17850 / 25701	84.5 %	892 / 1056	53.2 %	11273 / 21192

V. JDBC Code Coverage Results



Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	98	79%	8124/10195	68%	2617/3798	3.074
cubrid.jdbc.driver	56	82%	5002/6053	71%	1393/1940	2.538
cubrid.jdbc.jci	36	73%	2739/3709	65%	1140/1739	4.781
cubrid.jdbc.log	2	92%	64/69	92%	12/13	1.393
cubrid.jdbc.net	1	72%	71/98	47%	18/38	12
cubrid.jdbc.util	1	96%	88/91	88%	23/26	2.231
cubrid.sql	2	91%	160/175	73%	31/42	2.826