CUBRID 2008 R4.1 Patch1 QA Completion Report

This document is a verification report of CUBRID 2008 R4.1 Patch1 in terms of functionality, performance, 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 R4.1 Patch1 (hereinafter referred to as R4.1 Patch1), which is under development for release in February 2012 and to determine its release based on the test results. To test the stability of CUBRID, a test environment was configured as described below. Based on a comparison between the performance test result of CUBRID 2008 R4.1 Patch1 and that of CUBRID 2008 R4.0 Patch2 (hereinafter referred to as R4.0 Patch2), we tested to determine whether the performance of R4.1 Patch1 was regressed or improved.

- CentOS 5.5 (32/64-bit) or compatible
- CentOS 5.3 (32/64-bit) or compatible
- Windows 2003 (32/64-bit) or compatible
- Final test build: 8.4.1.1018 (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 different from the one described here. To verify product stability and functionality, performance, functionality, stability and other tests were performed for 4 types of builds as shown in the figure below. The details of each test are described in the appendix of this report.

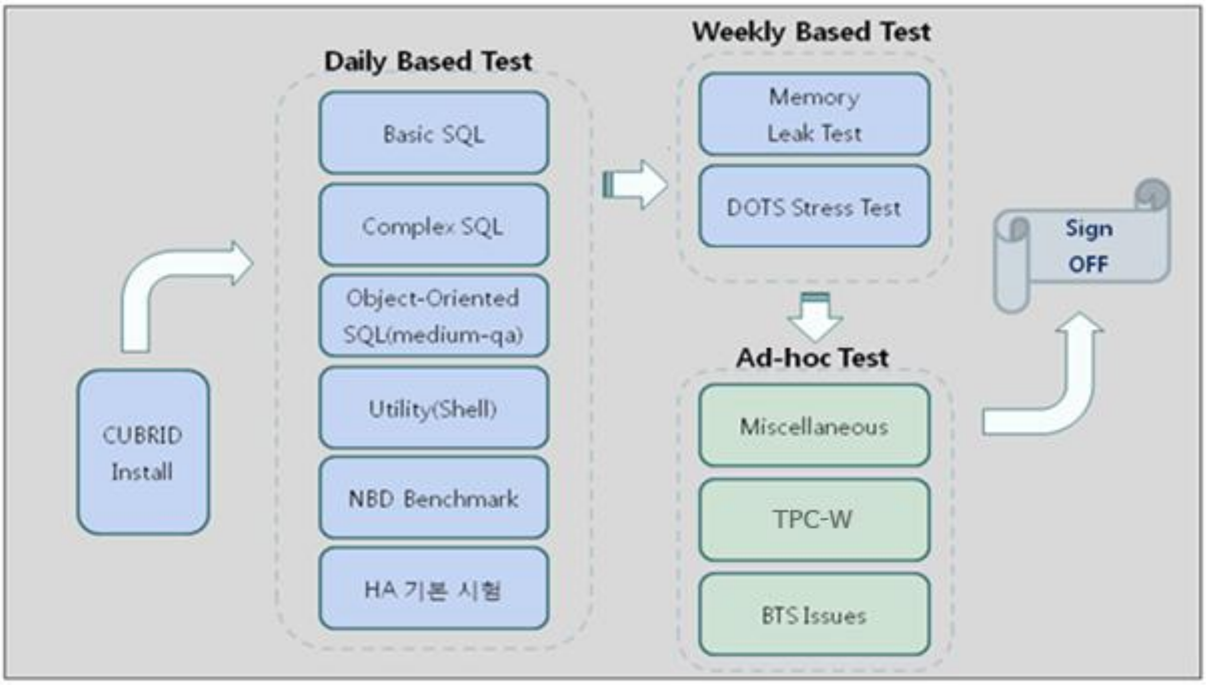


Figure 1. CUBRID Test Procedure

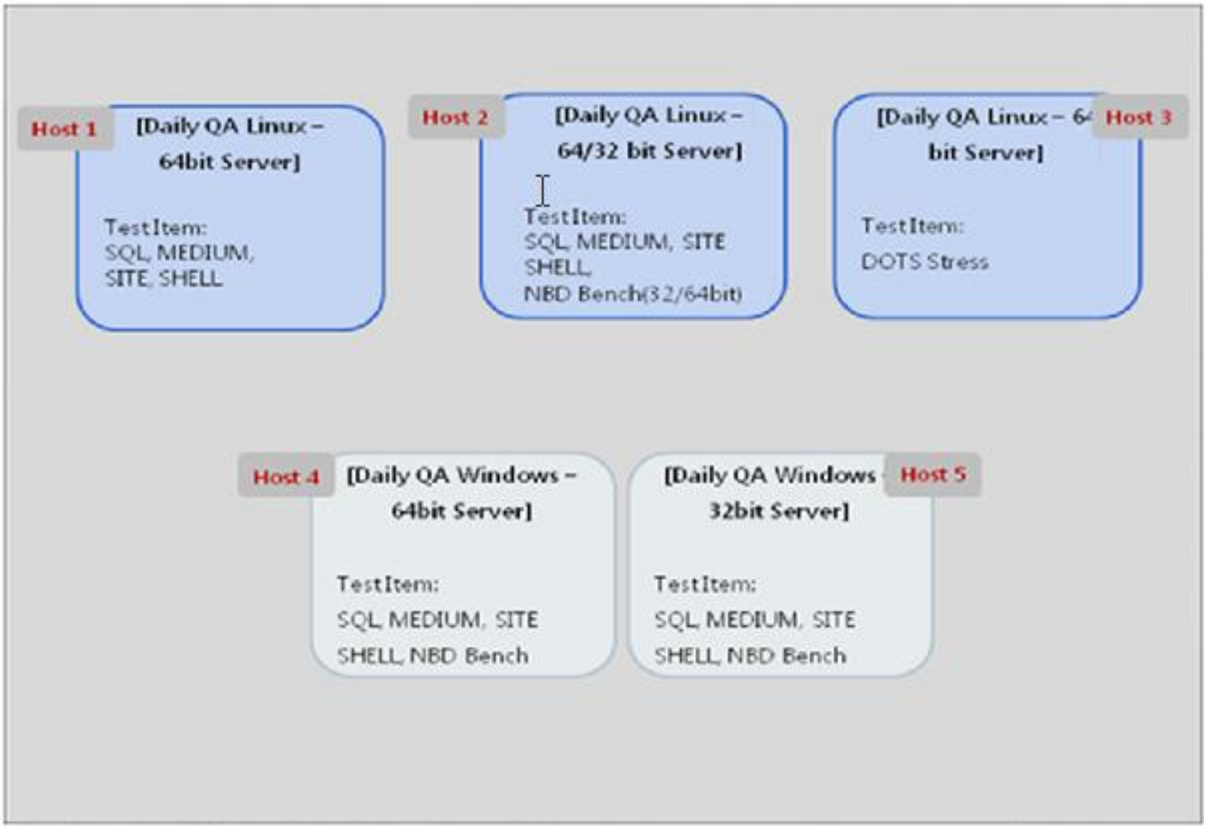

Host 1
[Daily QA Linux – 64bit Server]
TestItem:
SQL, MEDIUM, SITE, SHELL
Host 2
[Daily QA Linux – 64/32 bit Server]
TestItem:
SQL, MEDIUM, SITE SHELL, NBD Bench(32/64bit)
Host 3
[Daily QA Linux – 64 bit Server]
TestItem:
DOTS Stress
Host 4
[Daily QA Windows – 64bit Server]
TestItem:
SQL, MEDIUM, SITE SHELL, NBD Bench
Host 5
[Daily QA Windows 32bit Server]
TestItem:
SQL, MEDIUM, SITE SHELL, NBD Bench

Figure 2. System diagram for basic test

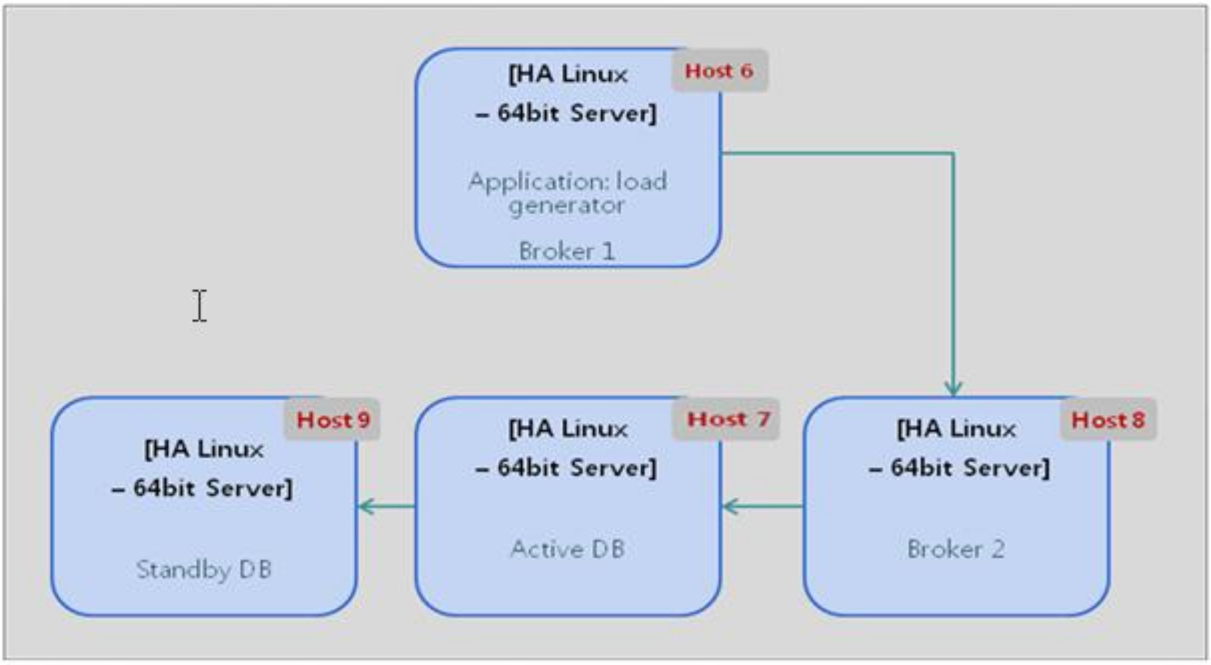


Figure 3. System diagram for HA test

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 core) * 1	16 GB	SAS 600G * 3 (Raid5)
Host 2	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 core) * 1	16 GB	SAS 600G * 3 (Raid5)
Host 3	Cent OS 5.3 (64-bit)	Xeon(R) 2.4 GHz (12 core) * 1	16 GB	SAS 600G * 3 (Raid5)
Host 4	Windows 2003 (64-bit)	Xeon 2.10 GHz (quadcore) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 5	Windows 2003 (32-bit)	Xeon 2.10 GHz (quadcore) * 1	4 GB	SATA 500G * 2 (No Raid)
Host 6	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 core) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 7	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 core) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 8	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 core) * 2	8 GB	SATA 500G * 2 (No Raid)
Host 9	Cent OS 4.7 (64-bit)	Xeon 2.00 GHz (8 core) * 2	8 GB	SATA 500G * 2 (No Raid)

1.3 Test Category

The following tests were performed to determine whether CUBRID can be released. The details of each test are described in the appendix of this report.

- Functionality test
  - SQL query test
  - MEDIUM query test
  - SITE query test
  - Utility (Shell) test
  - Basic HA feature test
  - CCI/PHP/JDBC Interface test
- Performance test
  - Performance test for basic DBMS functions
  - YCSB Benchmark
  - NBD Benchmark
- Stability test
  - DOTS stress test
- HA Enhancement
  - TPC-W test

  - SQL/MEDIUM with valgrind test
  - Dots on HA test
- Other tests
  - Test for checking CUBRID 2008 R4.1 Patch1 functionalities/bug fixes
  - Memory check 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 by using SQL statements. SQL statements stored in 10928 files were tested 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.

Table 1. Result of Basic Query Tests

Test Category	Number of Scenarios	Number of Scenarios passed	Pass Rate
SQL query test	8745	8745	100%
MEDIUM query test	970	970	100%
SITE query test	1213	1213	100%

2.1.2 Basic Utility and Other Scenario Tests

This test was 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 using SQL statements. We ran scenarios written by 582 shell scripts to verify DBMS conformity.

Table 2. Result of Basic Utility and Other Scenario Tests

Test Category	Number of Scenarios	Number of Scenarios passed	Pass Rate
Utility	197	197	100%
Bug regression	290	290	100%
Environment variable	5	5	100%
Other	90	90	100%

2.1.3 HA Feature Tests

Table 3. Result of HA Feature Tests

Test Category	**Number of Scenarios**	**Number of Scenarios passed**	**Pass Rate**
Data replication test	5	5	100%

Bug regression	69	69	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%

2.2 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 appendix. For all environment variables, except for SQL_LOG=OFF in cubrid_broker.conf, default configuration values were used. As shown in the table below, we have found the overall performance in Linux(64-bit/32-bit) has been improved slightly, but for Windows 32-bit, the performance in all operations are not significant improvement. It is needed to investigate it more in the future.

A. Linux: Performance Comparison between CUBRID 2008 R4.0 Patch2 and CUBRID 2008 R4.1 Patch1 (64-bit)

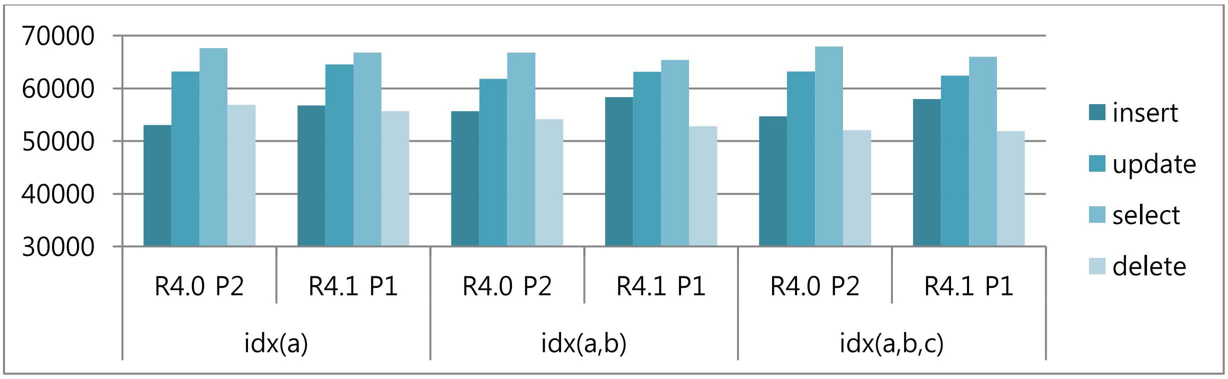


Figure 4. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Linux 64-bit)

Table 4. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Linux 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.0 P2	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	53043	56736	107%	55652	58357	105%	54674	57969	106%
Update	63194	64515	102%	61797	63126	102%	63203	62429	99%
Select	67632	66787	99%	66809	65374	98%	67936	65992	97%
Delete	56850	55654	98%	54144	52811	98%	52055	51880	100%
Total	240719	243692	101%	238402	239668	101%	237868	238270	100%

(Unit: TPS)

B. Linux: Performance Comparison between CUBRID 2008 R4.0 Patch2 (32-bit) and CUBRID 2008 R4.1 Patch1 (32-bit)

We can find the performance has been improved slightly.

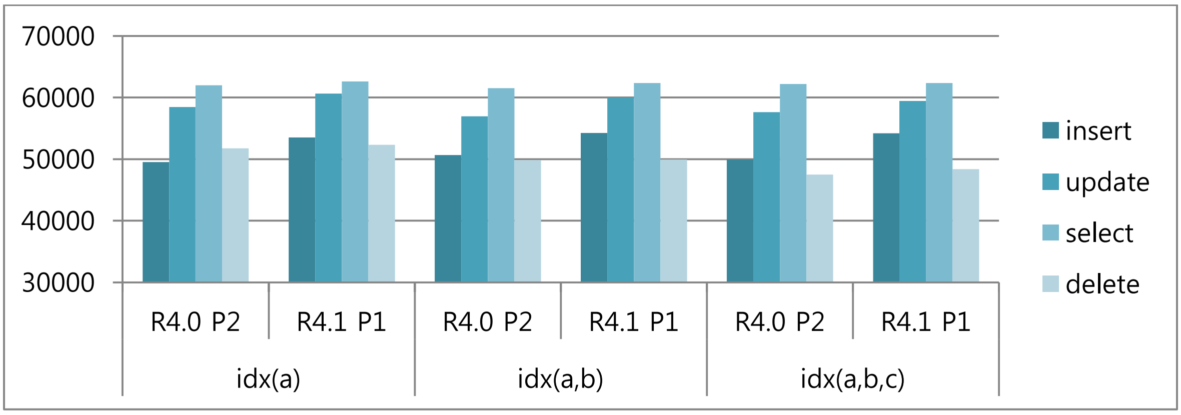


Figure 5. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Linux 32-bit)

Table 5. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Linux 32-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.0 P2	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	49499	53517	108%	50659	54276	107%	49943	54207	109%
Update	58458	60656	104%	56956	60012	105%	57628	59476	103%
Select	62005	62605	101%	61530	62382	101%	62208	62386	100%
Delete	51751	52323	101%	49821	49924	100%	47476	48359	102%
Total	221713	229101	103%	218966	226594	103%	217255	224428	103%

(Unit: TPS)

C. Windows: Performance Comparison between CUBRID 2008 R4.0 Patch2(64-bit) and CUBRID 2008 R4.1 Patch1 (64-bit)

We found that the performance of 64-bit R4.1 Patch1 is higher about 20% than that of 64-bit R4.0 Patch2.

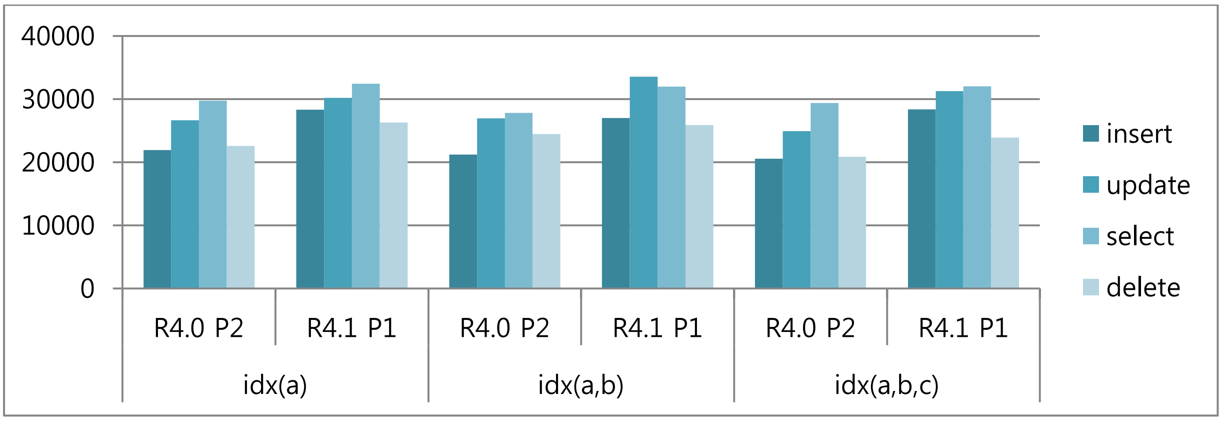


Figure 6. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Windows 64-bit)

Table 6. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Windows 64-bit)

	idx(a)			idx(a,b)			idx(a,b,c)		
	R4.0 P2	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	21935	28321	129%	21223	27004	127%	20543	28353	138%
Update	26661	30215	113%	26953	33574	125%	24948	31259	125%
Select	29740	32435	109%	27830	31962	115%	29402	32030	109%
Delete	22611	26273	116%	24463	25890	106%	20838	23913	115%
Total	100947	117244	116%	100469	118430	118%	95731	115555	121%

(Unit: TPS)

D. Windows: Performance Comparison between CUBRID 2008 R4.0 Patch2 (32-bit) and CUBRID 2008 R4.1 Patch1 (32-bit)

We have found that there was no significant change in performance between 32-bit R4.0 Patch2 and 32-bit R4.1 Patch1.

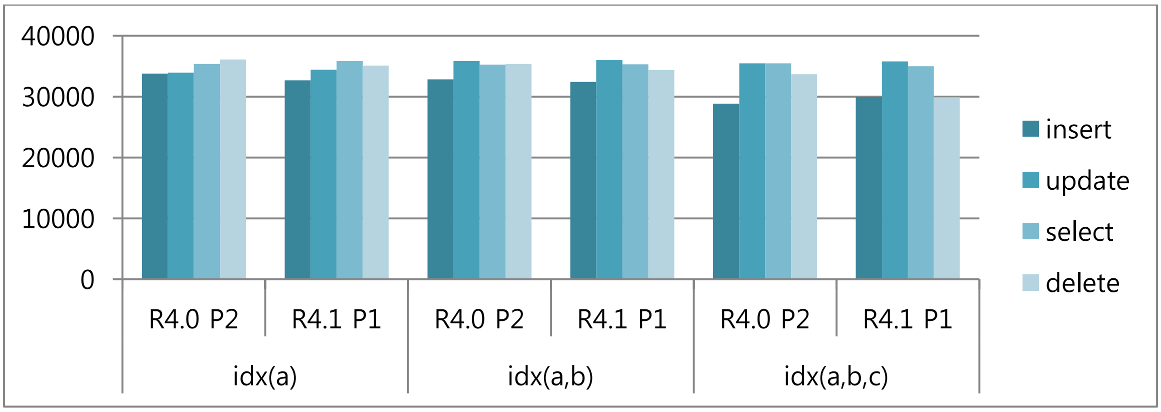


Figure 7. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Windows 32-bit)

Table 7. Performance Comparison between R4.0 Patch2 and R4.1 Patch1 (Windows 32-bit)

	idx(a)			**idx(a,b)**			**idx(a,b,c)**		
	R4.0 P2	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	33803	32692	97%	32866	32432	99%	28855	29949	104%
Update	33960	34432	101%	35874	36012	100%	35505	35771	101%
Select	35394	35828	101%	35286	35345	100%	35464	35030	99%
Delete	36136	35105	97%	35354	34360	97%	33685	29892	89%
Total	139293	138057	99%	139380	138149	99%	133509	130642	98%

(Unit: TPS)

2.2.2 YCSB Performance Test

YCSB as a framework for benchmarking system is popular in the world (see also https://github.com/brianfrankcooper/YCSB/wiki). This test was performed to verify CUBRID performance not only basic but also compositive operations, which are insert, select, scan, update and mix for them. For more information about test scenarios, see the appendix II. As shown in the results below, we have found that under master server configuration, the performance for update and insert operations got remarkable improvement, and exceeded more than 60%. Under another slave server configuration, they also had improvement but slightly. In addition, we also found that, for master server configuration, scan operation regressed slightly. It is needed to investigate it more in the future.

A. Master Server Configuration: Performance Comparison between R4.0 Patch2 (64-bit) and R4.1 Patch1 (64-bit)

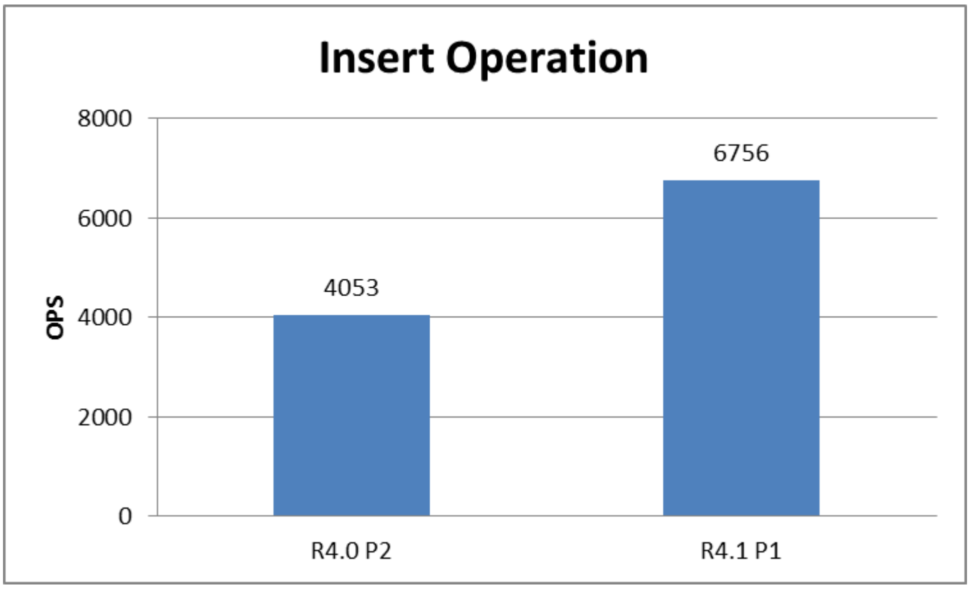


Figure 8. Result of Insert Operation of YCSB Benchmark (Master Server)

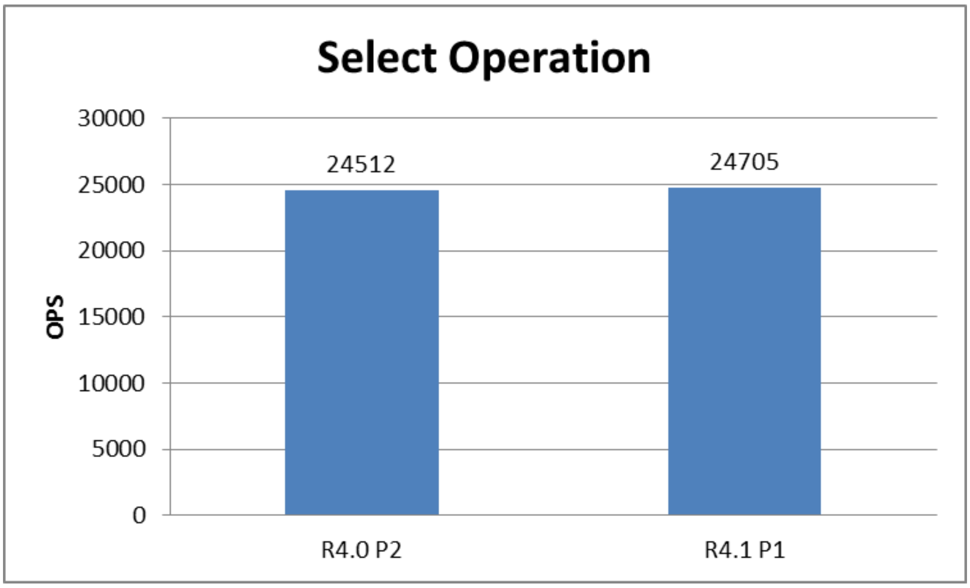


Figure 9. Result of Select Operation of YCSB Benchmark (Master Server)

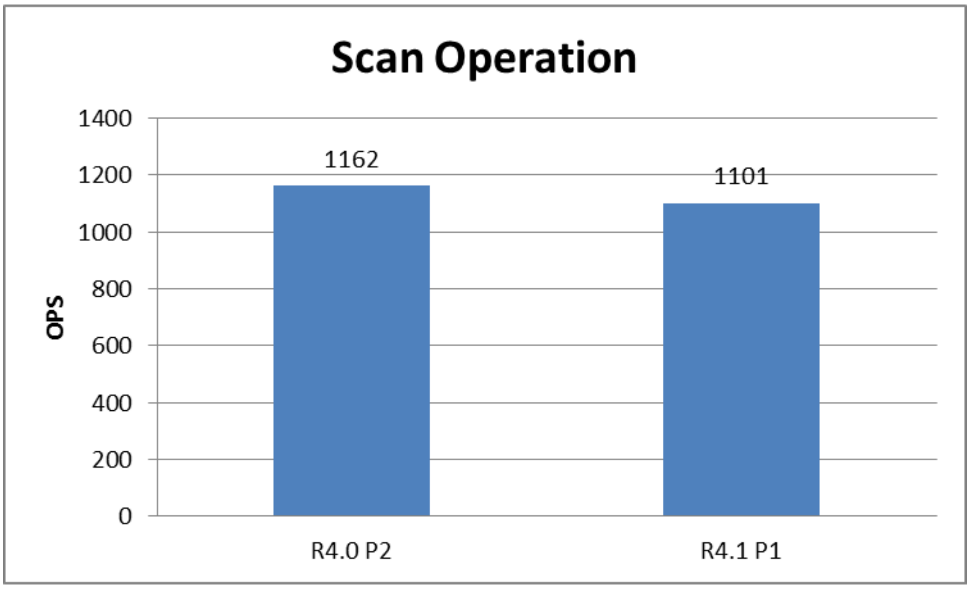


Figure 10. Result of Scan Operation of YCSB Benchmark (Master Server)

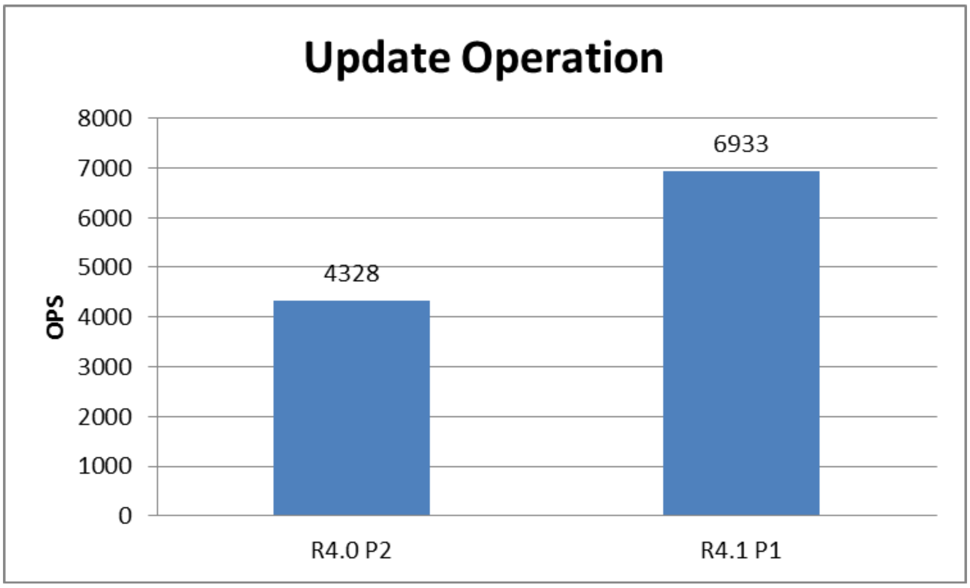


Figure 11. Result of Update Operation of YCSB Benchmark (Master Server)

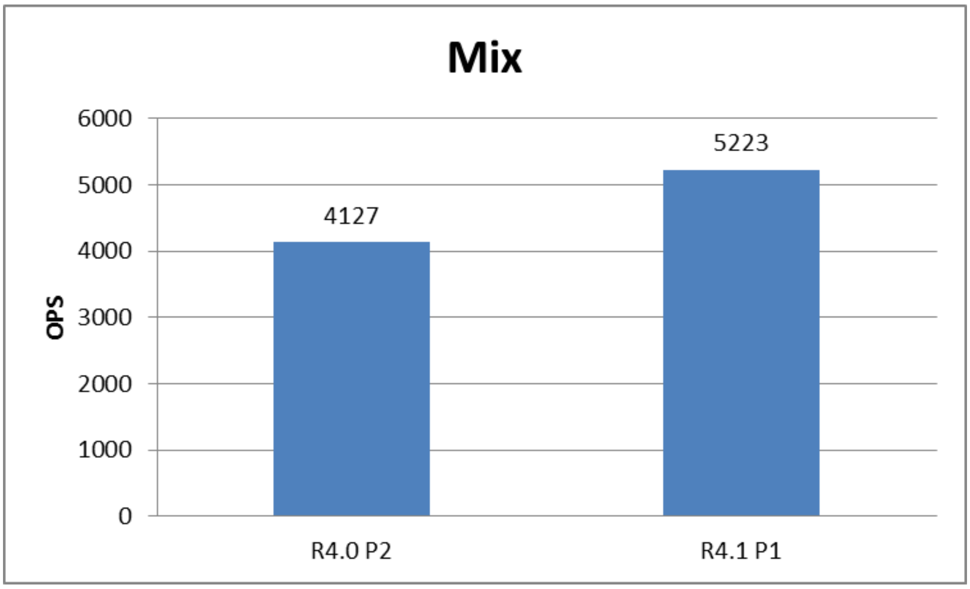


Figure 12. Result of Mixed of YCSB Benchmark (Master Server)

Table 8. Result of YCSB Benchmark (Master Server)

	Throughput (ops/sec)			**99th Percentile Latency (ms)**		
Operations	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	4053	6756	167%	33	13	253%
Select	24512	24705	101%	3	3	100%
Scan	1162	1101	95%	68	67	101%
Update	4328	6933	160%	32	13	246%
Mix	4127	5223	126%	223	65	343%

B. Slave Server Configuration: Performance Comparison between R4.0 Patch2 (64-bit) and R4.1 Patch1 (64-bit)

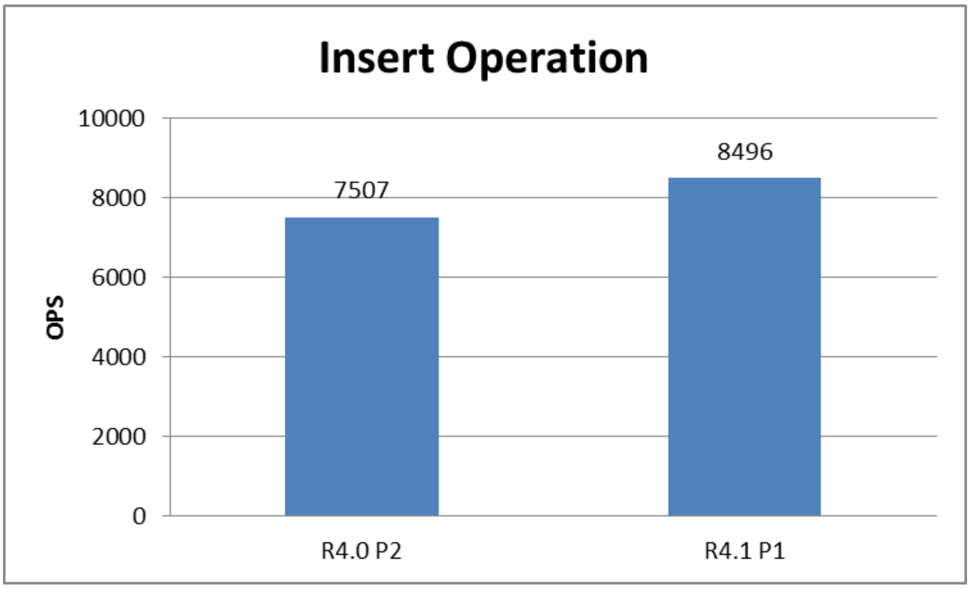


Figure 13. Result of Insert Operation of YCSB Benchmark (Slave Server)

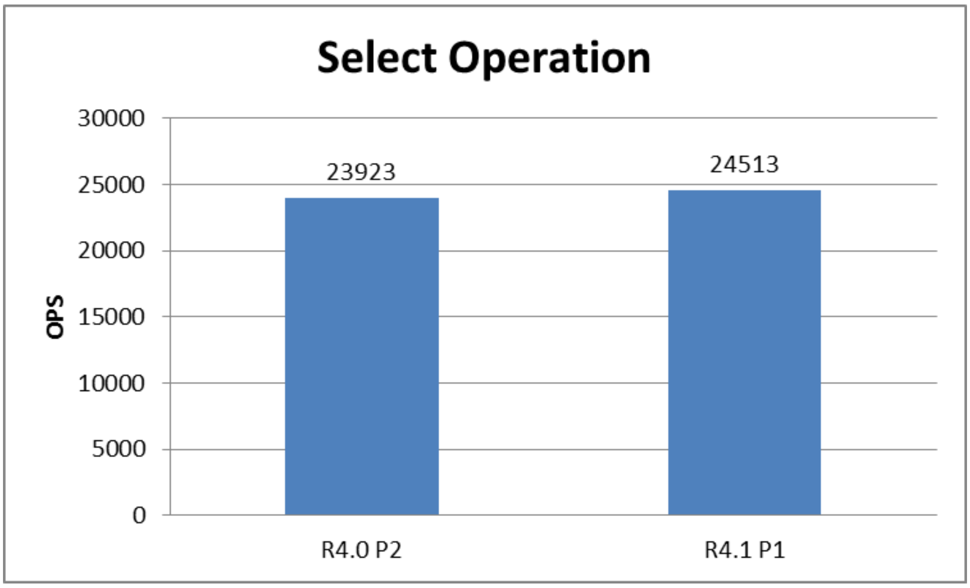


Figure 14. Result of Select Operation of YCSB Benchmark (Slave Server)

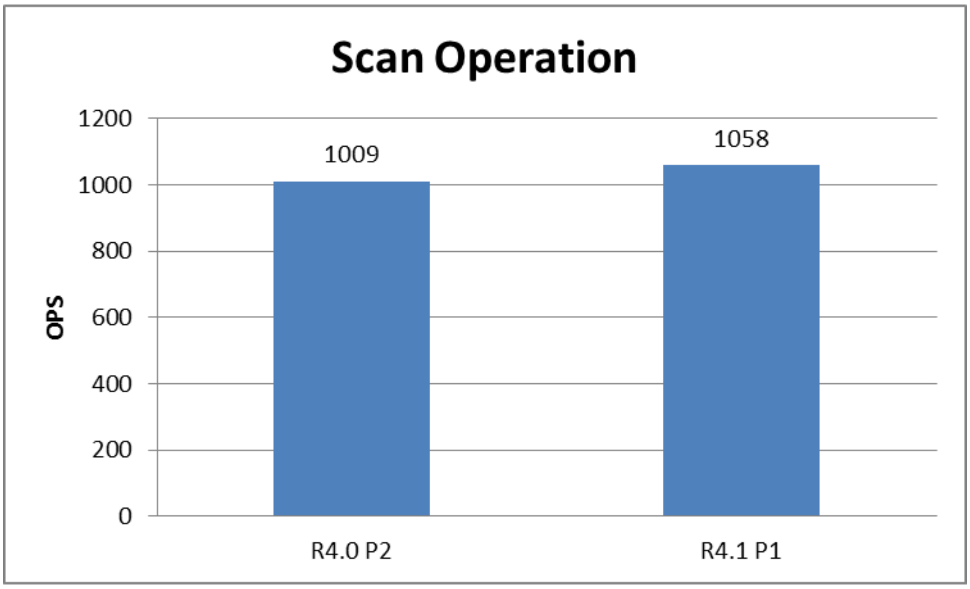


Figure 15. Result of Scan Operation of YCSB Benchmark (Slave Server)

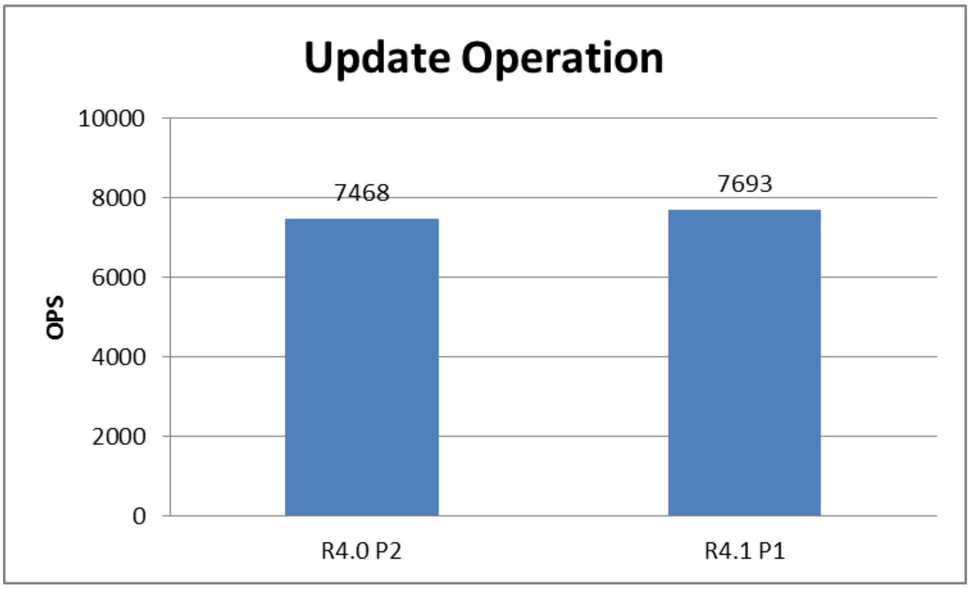


Figure 16. Result of Update Operation of YCSB Benchmark (Slave Server)

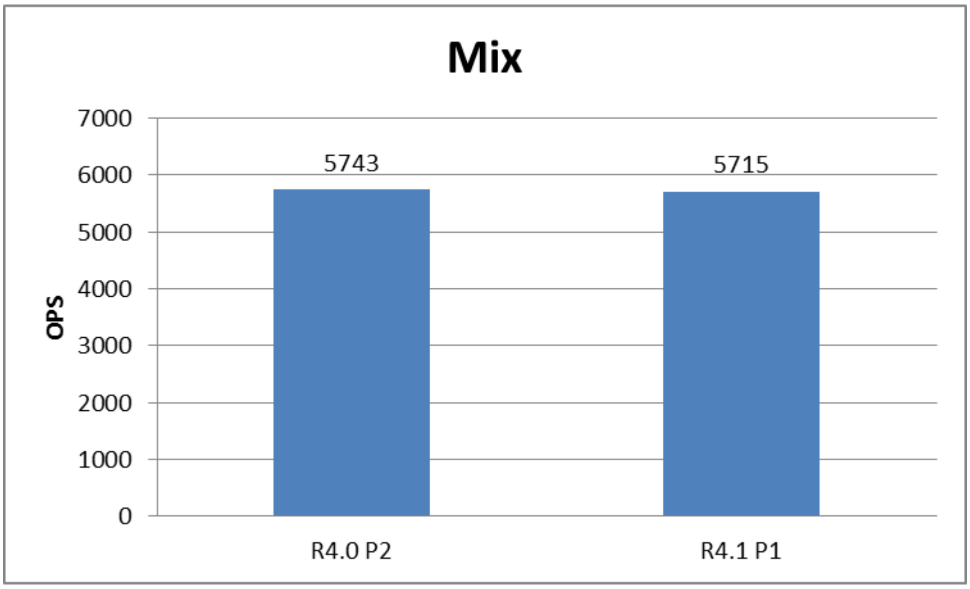


Figure 17. Result of Mixed of YCSB Benchmark (Slave Server)

Table 9. Result of YCSB Benchmark (Slave Server)

	Throughput (ops/sec)			99th Percentile Latency (ms)		
Operations	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**	**R4.0 P2**	**R4.1 P1**	**Performance Ratio**
Insert	7507	8496	113%	21	8	262%
Select	23923	24513	102%	3	3	100%
Scan	1009	1058	105%	78	73	106%
Update	7468	7693	103%	23	18	127%
Mix	5743	5715	100%	24	18	133%

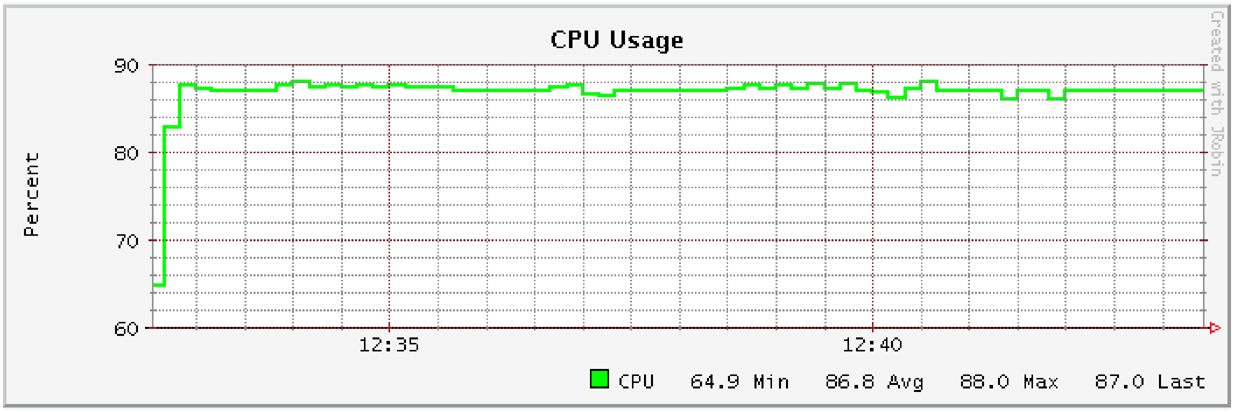
2.2.3 NBD Benchmark Performance Test

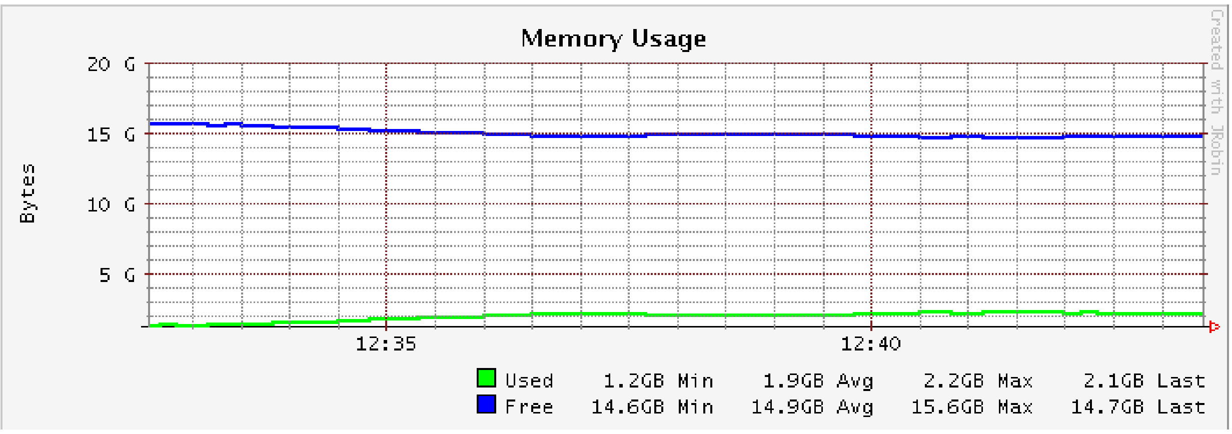
This test was performed to verify CUBRID performance by using the NBD Benchmark tool, which has been developed to verify the performance of the general bulletin board application framework. The scalability of the test DB was Level 1. As shown in the results below, there was not remarkable improvement in NBD Benchmark test loads between R4.0 Patch2 and R4.1 Patch1. CUBRID R4.1 Patch1 is slightly higher than CUBRID R4.0 Patch2. The Page View improved about 5% in CUBRID 2008 R4.1 Patch1 than CUBRID 2008 R4.0 Patch2.

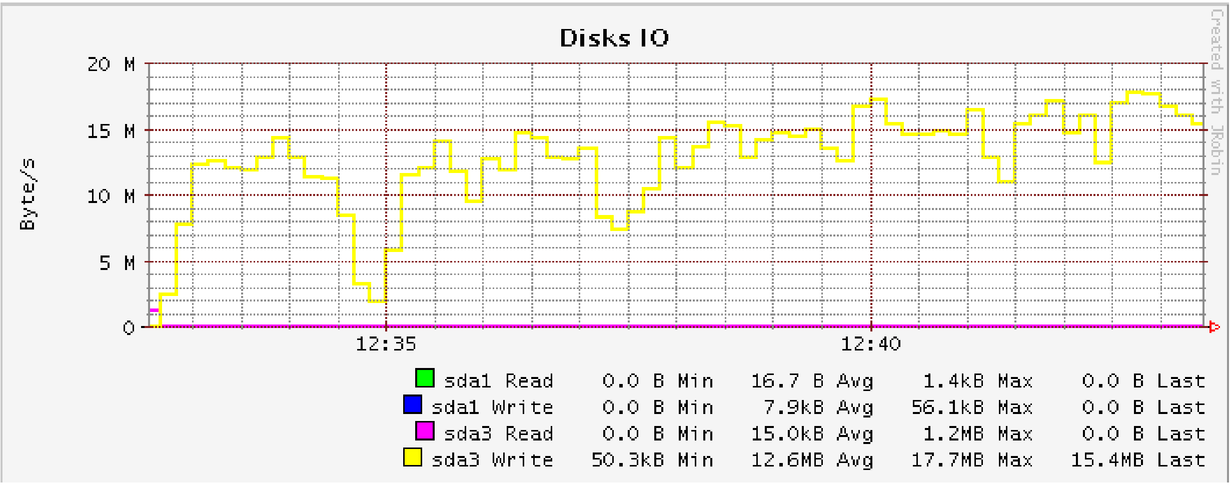
Table 10. Result of NBD Benchmark

Platform	Version	BIT	Page View
Linux 64	CUBRID 2008 R4.0 P2 (Default parameter configured)	64-bit	1267
Linux 32	CUBRID 2008 R4.0 P2 (Default parameter configured)	32-bit	1287
Linux 64	CUBRID 2008 R4.1 P1 (Default parameter configured)	64-bit	1325
Linux 32	CUBRID 2008 R4.1 P1 (Default parameter configured)	32-bit	1291

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

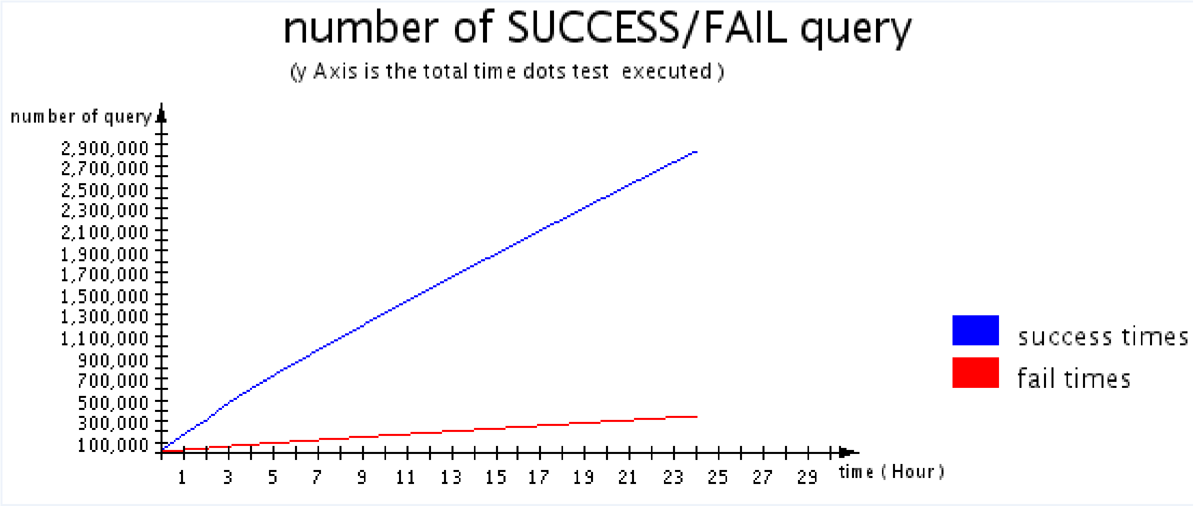


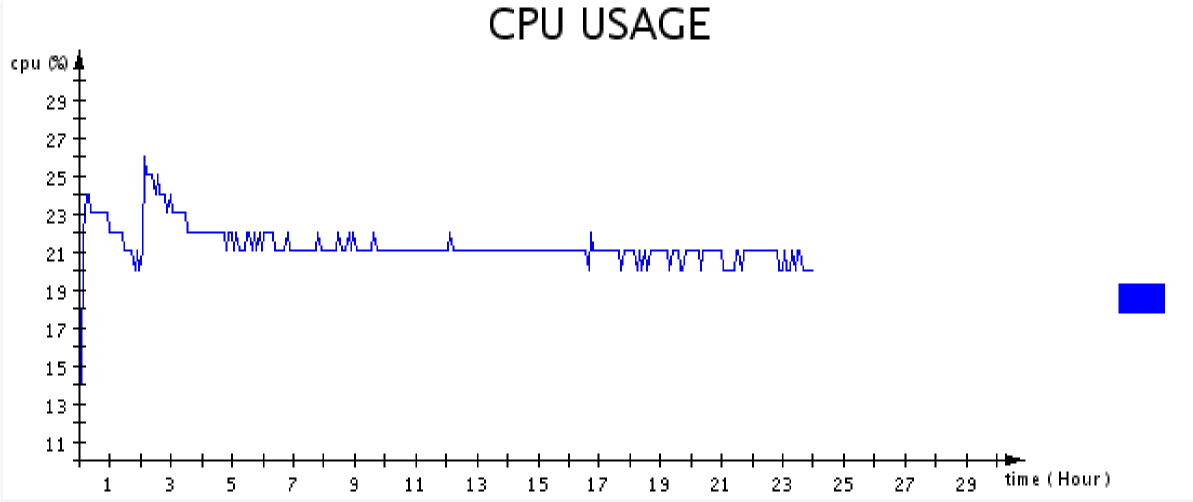


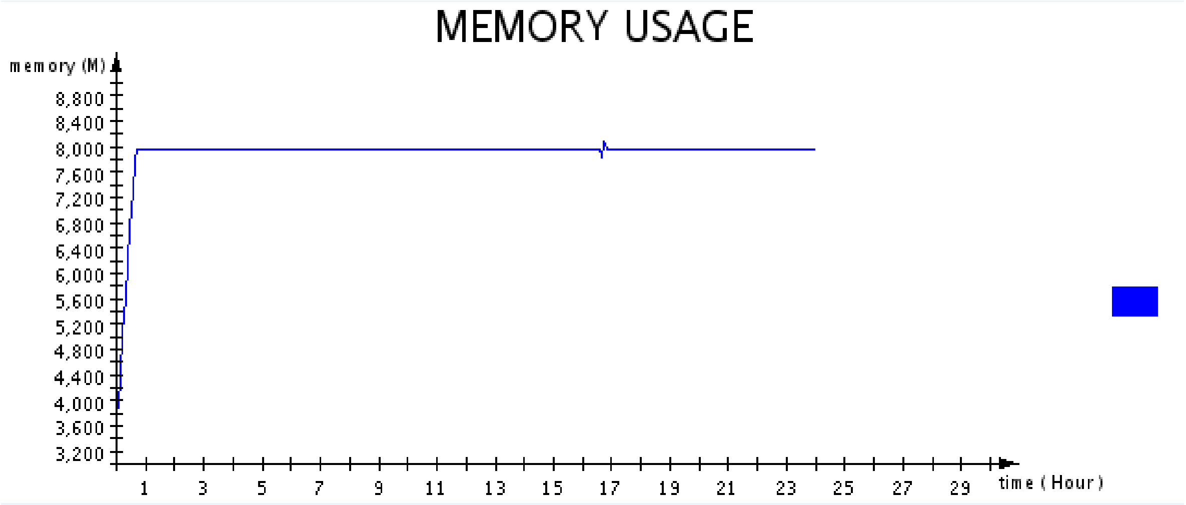


2.3 Stability Test Results

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. As shown in the test results below, the system operated stably without any abnormalities during the 24-hour load period. You can ignore the fails because they are unique violations due to the modification of duplicate data.







2.4 Other Test Results

All bug fixes resolved in CUBRID 2008 R4.1 Patch1 have been confirmed.

2.5 Quality Index

The standard quality index of CUBRID 2008 R4.1 Patch1 is listed below.

Table 11. Quality Index of R4.1 Patch1

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.	827,066 LOC
			Total number of source code lines in the changed files	827,066 LOC
QA Scenario Code Coverage	75%	73.5%	Number of tested statements	170,523
			Total number of statements	232,099
Fault Density Detected by Static Analysis	4 /KLOC	2.67 /KLOC	Number of faults detected by static analysis (Level 1)	161
			Number of faults detected by static analysis (Level 2)	7
			Number of faults detected by static analysis (Level 3)	424
			Number of faults detected by static analysis (Level 4)	0
			Total number of source code lines	809,205LOC
Cyclomatic Code Complexity	3.3%	2.9%	Number of modules whose complexity is over 30	563
			Total number of modules in a project	19,121
	12%	16.2%	Number of modules whose complexity is over 10	3,093
			Total number of modules in a project	19,121

3. Conclusions

As described in Chapters 1 and 2, CUBRID 2008 R4.1 Patch1 has been tested in terms of its functionality, performance, stability and other issues before its release.

The tests have been performed in the Linux 32-bit, Linux 64-bit, Windows 32-bit and Windows 64-bit environments. All tests were executed, and the related defects have been logged into BTS for (32/64-bit) on Linux/Windows.

Based on the results obtained through the basic performance test, we have found that the overall basic performance of CUBRID 2008 R4.1 Patch1 was slightly improved than that of CUBIRD 2008 R4.0 Patch2 in 32/64bit Linux/Windows.

Based on the YCSB performance results, we have found that, under master server configuration, the performance for update and insert operations got very remarkable improvement, and exceeded more than 60%. We can certainly say that this is the most significant changes in performance that CUBRID 2008 R4.1 Patch1 bring us. We are very excited that our customers can clearly feel the notable performance improvement for massive insert and update operations because this is also default configuration after installation. In addition, the other operations' performance also show progressive result.

Based on the results obtained through the NBD benchmark test, we have found that there was slightly improvement in performance between CUBRID 2008 R4.0 Patch2 and CUBRID 2008 R4.1 Patch1. The Page View of CUBRID 2008 R4.1 Patch1 improved about 5% than that of CUBRID 2008 R4.0 Patch2.

Appendix

I. Functionality Test Scenarios

This test was performed to verify the basic DBMS functionalities by using SQL statements. SQL statements stored in files were tested 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: 8745		
Case Name	Path	**Description**
object	sql/_01_object	Performs functionality tests of objects supported by CUBRID, and has the largest number of scenarios (3332 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 an 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 an 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 te st of limit optimizing, using index clause enhancement, descending index scan, covering index, ordering index, optimizing group by clause, Index scan with like predic ate, next key locking,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: 1213		
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 by using shell scripts. In particular, this test was also performed to verify CUBRID utilities that cannot be tested by using SQL statements. We ran scenarios written using shell scripts to verify DBMS conformity.

Total: 582		
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.

- HA Feature Test

Total: 221		
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

II. Performance Test Scenario

- 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

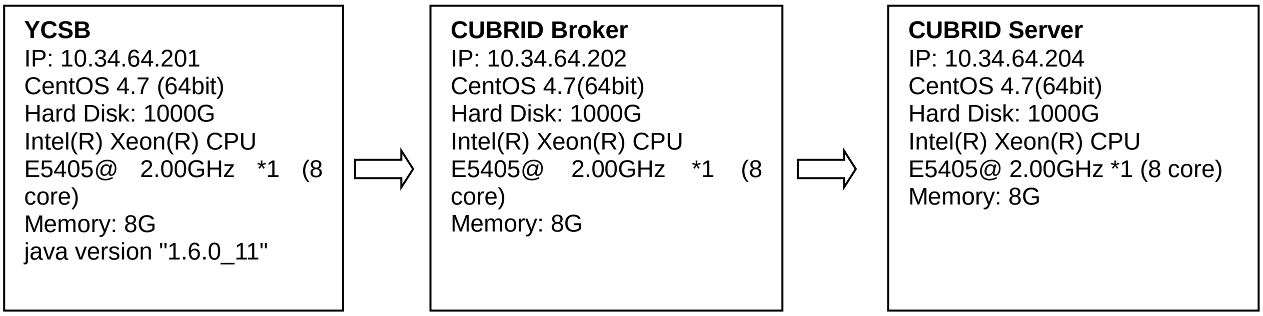
- Measure the number of loops per second.
- For concurrent user tests, add the execution times of all users.

■ YCSB Benchmark

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

■ Common Test Environment

✧ Test Servers



YCSB
IP: 10.34.64.201
CentOS 4.7 (64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5405@ 2.00GHz *1 (8 core)
Memory: 8G
java version "1.6.0_11"

CUBRID Broker
IP: 10.34.64.202
CentOS 4.7(64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5405@ 2.00GHz *1 (8 core)
Memory: 8G

CUBRID Server
IP: 10.34.64.204
CentOS 4.7(64bit)
Hard Disk: 1000G
Intel(R) Xeon(R) CPU E5405@ 2.00GHz *1 (8 core)
Memory: 8G

✧ 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
```

- Use default CUBRID broker configuration except below:
  - cubrid_broker.conf: sql_log=OFF

- 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=40
    - 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=40

- fieldlength=10

- Scan operation

  - recordcount=10000000

  - operationcount=100000

  - workload=com.yahoo.ycsb.workloads.CoreWorkload

  - readallfields=true

  - readproportion=0

  - updateproportion=0

  - scanproportion=1

  - insertproportion=0

  - requestdistribution=zipfian

  - threads=40

  - fieldlength=10

- Update operation

  - recordcount=10000000

  - operationcount=1000000

  - workload=com.yahoo.ycsb.workloads.CoreWorkload

  - readallfields=true

  - readproportion=0

  - updateproportion=1

  - scanproportion=0

  - insertproportion=0

  - requestdistribution=zipfian

  - threads=40

  - fieldlength=10

- Mix operation

  - recordcount=10000000

  - operationcount=1000000

  - workload=com.yahoo.ycsb.workloads.CoreWorkload

  - readallfields=true

- readproportion=0.3
- updateproportion=0.3
- scanproportion=0.1
- insertproportion=0.3
- requestdistribution=zipfian
- threads=40
- fieldlength=10

- ■ Test data on master server configuration
  - ✧ CUBRID server configuration
    - ♦ async_commit=no
    - ♦ group_commit_interval_in_msecs=0
    - ♦ data_buffer_size=4G
  - ✧ 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 slave server configuration
  - ✧ CUBRID server configuration
    - ♦ async_commit=yes
    - ♦ group_commit_interval_in_msecs=1000
    - ♦ data_buffer_size=4G
  - ✧ 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)
)
CREATE INDEX ink2_usertable ON usertable (userkey, field1);
CREATE INDEX ink3_usertable ON usertable (userkey, field1, field2);
```

- 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%

- NBD Benchmark

This test was performed to verify CUBRID performance by 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.

III. Stability Test Scenario

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
);
```

- 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

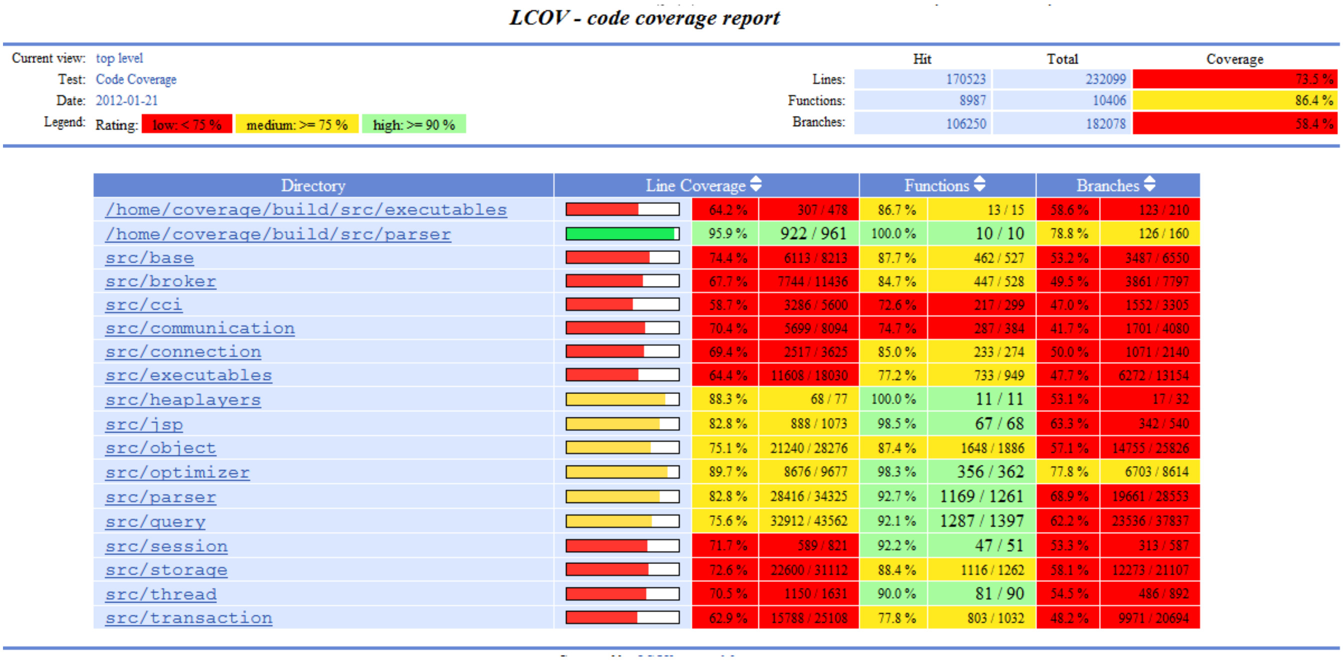
```
SELECT SELLERID,DESCRIPTION,BID_PRICE,START_TIME,END_TIME,BID_COUNT
FROM ITEM WHERE ITEMID =
UPDATE ITEM SET DESCRIPTION = ?,BID_PRICE = ?,START_TIME = ?,END_TIME = ? WHERE ITEMID = ?
```

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				
Test:	Code Coverage	Lines:	170523	232099	73.5 %
Date:	2012-01-21	Functions:	8987	10406	86.4 %
Legend:	Rating: low: < 75 % medium: >= 75 % high: >= 90 %	Branches:	106250	182078	58.4 %

Directory	Line Coverage			Functions		Branches	
/home/coverage/build/src/executables		64.2 %	307 / 478	86.7 %	13 / 15	58.6 %	123 / 210
/home/coverage/build/src/parser		95.9 %	922 / 961	100.0 %	10 / 10	78.8 %	126 / 160
src/base		74.4 %	6113 / 8213	87.7 %	462 / 527	53.2 %	3487 / 6550
src/broker		67.7 %	7744 / 11436	84.7 %	447 / 528	49.5 %	3861 / 7797
src/cci		58.7 %	3286 / 5600	72.6 %	217 / 299	47.0 %	1552 / 3305
src/communication		70.4 %	5699 / 8094	74.7 %	287 / 384	41.7 %	1701 / 4080
src/connection		69.4 %	2517 / 3625	85.0 %	233 / 274	50.0 %	1071 / 2140
src/executables		64.4 %	11608 / 18030	77.2 %	733 / 949	47.7 %	6272 / 13154
src/heaplayers		88.3 %	68 / 77	100.0 %	11 / 11	53.1 %	17 / 32
src/jsp		82.8 %	888 / 1073	98.5 %	67 / 68	63.3 %	342 / 540
src/object		75.1 %	21240 / 28276	87.4 %	1648 / 1886	57.1 %	14755 / 25826
src/optimizer		89.7 %	8676 / 9677	98.3 %	356 / 362	77.8 %	6703 / 8614
src/parser		82.8 %	28416 / 34325	92.7 %	1169 / 1261	68.9 %	19661 / 28553
src/query		75.6 %	32912 / 43562	92.1 %	1287 / 1397	62.2 %	23536 / 37837
src/session		71.7 %	589 / 821	92.2 %	47 / 51	53.3 %	313 / 587
src/storage		72.6 %	22600 / 31112	88.4 %	1116 / 1262	58.1 %	12273 / 21107
src/thread		70.5 %	1150 / 1631	90.0 %	81 / 90	54.5 %	486 / 892
src/transaction		62.9 %	15788 / 25108	77.8 %	803 / 1032	48.2 %	9971 / 20694