Alternative data storage solution for mobile messaging services

David C.C. Ong^{a,*}, Rytis Sileika^a, Souheil Khaddaj^{a,b} and Radouane Oudrhiri^b

^a*Faculty of Computing, Information Systems and Mathematics, Kingston University, Kingston upon Thames KT1 2EE, UK*

^b*Systonomy Ltd, 2nd Floor, Berkeley Square House, Berkeley Square, London W1J 6BD, UK*

Abstract. In recent years, mobile devices have become relatively more powerful with additional features which have the capability to provide multimedia streaming. Better, faster and more reliable data storage solutions in the mobile messaging platform have become more essential with these additional improvements. The existing mobile messaging infrastructure, in particular the data storage platform has become less proficient in coping with the increased demand for its services. This demand especially in the mobile messaging area (i.e. SMS – Short Messaging Service, MMS – Multimedia Messaging Service), which may well exceed 250,000 requests per second, means that the need to evaluate competing data management systems has become not only necessary but essential. This paper presents an evaluation of SMS and MMS platforms using different database management systems – DBMS and recommends the best data management strategies for these platforms.

Keywords: Database management systems, database reliability, mobile communication

1. Introduction

Designing a high performance mobile messaging platform is the main mission for today's mobile applications architects. Creating a more efficient messaging protocol or platform code often has improved the overall performance of the platform, but major problems relating to the efficiency of use of the platform's data management systems still remain. The Lightweight Directory Access Protocol – LDAP is commonly used as the data storage solution in the mobile messaging platform, but it is not efficient in storing large amounts of data such as MMS (which can reach over 100 KB in size). In fact, LDAP is not designed to handle this massive amount of data (as a result of the string-based encoding of names and attribute values in LDAP design [15]) but rather small and static data such as Simple Network Management Protocol – SNMP. Alternatively, Database Management System – DBMS could offer a solution as it has reached maturity for its data storage and maintainability solutions.

Although, there are various benchmarks and test results available [5,7,8], more investigation is still required in order to produce a convincing and precise evaluation model in terms of consistency and throughput of data over a given period of time. Also, there is both a lack of accurate representation of distributions in the actual system operations, and a lack of credible assessment of those non-complex data structures which resemble mobile messaging data structures, as most of them tend to concentrate on the

*Corresponding author: 3 Thames Meadow, West Moseley, Surrey KT8 1TQ, UK. Tel.: +44 7791 343 069; E-mail: david.ong@bcs.org.

performance of the larger database where the life span of the data stored is long and bulky compared with the mobile data. There is also a tendency toward popular areas such as On-Line Transaction Processing and Web Database Application [3], Extreme Database Test [1,5,8], etc. which do not satisfy the criteria required for the mobile data management systems.

Recently Sleepycat Software Inc. has published a white paper entitled “Managing Data within Billing, Mediation and Rating System” [7], this database performance model deals only with a small test sample, thus there were no statistically acceptable results to support the claim of the superiority of the chosen database.

The data structure of the mobile messaging platforms is simple, consisting of one or two tables with an estimated three columns each. It is considered unproductive and expensive to recreate a custom designed data management system, as current available data management systems have reached maturity in terms of their robustness and efficiency in handling and storing data. In view of this, most of the mobile messaging platforms use a data management system such as a DBMS to handle data manipulation and storage of the platform.

It should also be stated that related work which focuses on the quality of services (QoS) of the mobile devices in the wireless environment using DBMS as its data storage solutions, especially in the field of spatial location and query optimisation [16], sets out to discover how a workload of multiple queries in a real-time environment could be optimised and be incorporated into system performance for spatial [18, 20] or mobile [17] database. Such works have contributed to the on-going improvement of the QoS in the wireless environment. In addition, the creation of the evaluation framework in this paper has benefited from the performance benchmarks resulting from improvements in queries of syntax, semantics and algorithms that have also played a vital role in enhancing the overall QoS [19,21,22].

This paper aims to investigate and evaluate several database management systems that are commonly used for mobile messaging services. In the quest to find the best data management strategies for mobile messaging platforms, a critical evaluation was carried out to assess current available data management systems. This evaluation focused on the SMS and MMS platforms by observing performance and quality of service in the message handling under minimum and maximum workload, where data size remains consistent throughout the evaluation period.

In this work we start by considering a number of data management systems together with the software and hardware environment. Then, an evaluation framework was constructed together with some initial evaluation of the various platforms. Several real time experiments using SMS and MMS data structures were then conducted. Finally, we present some conclusions and suggestions for future works.

2. Data management systems

2.1. Database classification

It was considered unproductive to evaluate all of the data management systems available, as this would require huge resources and time, so data management systems were categorised into three main classes based on the storage engine provided. There are “Main Memory based DBMS”, “Disk based DBMS” and “Disk based data store without DBMS”. DBMS products such as Oracle, MySQL, TimesTen, Microsoft SQL Server, MaxDB, etc. have offered various data management solutions under these classifications.

“Main Memory based DBMS” manages, caches and stores data in the main memory [4]. It is the fastest in transaction management and storage but the data stored will be completely lost in the event of

a system failure. DBMS products such as TimesTen or MySQL also utilise a main memory engine for their data store engine.

“Disk based DBMS” provides a persistent storage to disk. It uses data cached in a memory buffer to handle transaction before storage to disk [4]. It is considered safe because data that have been successfully stored to the disk will not be lost in case of system failure. “Disk based DBMS” like Microsoft SQL Server and Oracle incorporates persistent data storage engines to provide a persistent storage solution. Unlike other established DBMS products, MySQL utilises third party data storage engines besides its own (e.g. InnoDB and BerkeleyDB) for the persistent storage solution.

Functionalities provided in DBMS are often considered a waste of resources if the data structures like mobile message services are simple and small. In “Disk based DBMSs”, data storage engines (e.g. InnoDB, BerkeleyDB, etc) are used to store and retrieve data from the disk. Hence, “Disk based data store without DBMS” could be considered a solution for managing a simple mobile messaging data structure.

2.2. Database selection

For each classification, the database that was considered the best for this evaluation was selected based on cost, efficiency, performance, reliability, popularity and their general availability.

In the mobile messaging industry, operators cannot afford data loss in the event of system failure. TimesTen was selected to represent a “Main Memory based DBMS” [6] because it provides an automatic secondary persistent data management disk solution. In the event of system failure, data loss will be minimal.

There are various “Disk based DBMS” products on the market. These products have gained popularity due to their high reliability and integrity, as there is a very small amount of data loss in the event of system failure. Popularity of MySQL in the data management solutions has grown recently [10,13] because it offers a cheaper alternative to the established “Disk based DBMS” product such as Oracle and Sybase. MySQL with MyISAM engine was therefore recommended to represent the “Disk based DBMS” class.

BerkeleyDB is a transaction safe storage engine with a page locking facility. It is viewed as safest as a data storage solution, as it only requires minimal processing overhead before data is safely stored. It was therefore chosen in the “Disk based DBMS” (i.e. MySQL) to provide a transaction safe solution to meet the data management requirement. It is considered easy to fit onto practically any data management system and efficient in handling simple data structures. Recently several of the established IT corporations such as Amazon.com, Cisco Systems, Google, Teligent, Mitel, Motorola and Sun Microsystems have looked into BerkeleyDB for their data management solutions [9,11,12]. For these reasons, BerkeleyDB has been chosen to represent the “Disk based data store without DBMS” class.

3. Construction of evaluation framework & platform

3.1. Hardware and software environment

Performance may depend on the server and database environment configuration and specification. Evaluation was conducted in a fixed server environment, where databases involved in the evaluation were installed. A cluster with 2×2.4 GHz (Intel Xeon HT) processor, 1.5 GB of RAM, 36GB U-SCSI 10 k internal HDD and a disk array (3×4 disk volumes (RAID 1 + 0)) 10 k for the shared HDD, which

housed Linux RedHat ES 3 with the 2.4 kernel as the operating system was used as the base evaluation platform. Java 2 Platform, Standard Edition (J2SE) 1.5 was adopted to execute the evaluation model.

It is acknowledged that the performance of the “Main Memory based DBMS” or “Disk based DBMS” may improve dramatically, simply by the installation of more memory or faster HDD in the server. For the purpose of this evaluation, basic hardware and software specifications had to be fixed to ensure that an even assessment could be performed. Optimisation of individual databases was done to the best of our knowledge and experience.

3.2. *Data structure*

The data structure for SMS consists of a single table, which has 10 bytes of index and 1,200 bytes for data. This platform gives a simple and easy way to handle and store conventional short messages. In order to handle long messages embedded with audio, graphics, video and data, MMS was devised. MMS has two tables, one of which is designed for the index and metadata of the message and the other was created for storing the actual data of the message. The index table consisted of 10 bytes of index and 1,200 bytes of metadata. The data table for the MMS has 10 bytes of index and 100,000 bytes of message data.

The mixture of tasks is based on the proportional distribution of tasks in the actual system observed in the SMS and MMS platforms. The evaluation model would replicate the actual distribution of these tasks. This is to ensure the model is able to mimic the processes in the actual platform environment.

Each process in the SMS evaluation model consists of a number of tasks; typically, it has 4 insertion, 12 selection, 1 updating and 4 deletion tasks of the SMS data. The index table for the MMS has a similar proportion of tasks as described in the SMS. The proportion of tasks for the data table consisting of 4 “insertion”, 8 “selection” and 4 “deletion” tasks was adopted. Distribution tasks for SMS and MMS platforms presented above were based on the functional requirements described in the ETSI Standard titled “Technical realization of the Short Message Service” for the GSM 3.40 [14].

3.3. *Evaluation framework*

The purpose of this evaluation was not to test the system under extreme conditions, to breaking point, but rather to evaluate an optimal level of performance for the system when it reaches a consistent level. The challenge was to control the state of the database during testing and to order the test runs in such a way that a measurable figure could be observed at the same time maintaining an optimal operation state [2]. Thus, the evaluation framework must achieve three identified aims to fulfil this purpose. These are:

- To define and achieve a consistent level in the system before any measurement is taken.
- To reach and maintain maximum data throughput to the database.
- To maintain the same mixture of tasks executed and at the same time ensuring randomness of the tasks.

The evaluation framework was created using JAVA utilising its threading functionalities to ensure that maximum throughput and randomness of task requests could be achieved. It first starts with the establishment of the database connectivity and then commences to send requests to the database (Fig. 1). The manner of the sent requests is controlled by four threads within the framework. True measurement of the database performance is taken when consistent throughput to the database is established. The measurement ends when the database has successfully executed a specific number of requests. The number of requests is determined by the level of request sent, which is divided into low, medium and high throughput. Further explanation of the framework is given in the following sections.

Table 1
Evaluation Framework for SMS Platform

Block (n)	Select (n - 1) 1: 1 of n	Insert (n) 1: 1 of n	Select (n - 1) 1: 1 of n	Delete (n - 2) 1: 1 of n	Select (n - 1) 1: 1 of n	Update (n - 1) 1: 4 of n	Select (n - 1) 1: 1 of n
1	X	✓	X	X	X	X	X
2	✓	✓	✓	X	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓
7	✓	X	✓	✓	✓	✓	✓
8	X	X	X	✓	X	X	X

Fig. 1. Evaluation Framework Overview.

3.3.1. Evaluation processes

The evaluation framework (Table 1) consists of 8 blocks of tasks. Each block consists of the proportion of distribution of tasks obtained in the actual system and repeated individually according to the specified recurrences. The proportion of distribution for tasks in the SMS and MMS evaluations is kept as discussed earlier (Section 3.2). The division of the framework into blocks is crucial to ensure each task is successfully executed and returns the anticipated value from the database, rather than returning null or not found. For the example in block 5, the SELECT task is based on the unique id inserted in block 4, INSERT task with unique id for the block 5, DELETE task is based on the unique id inserted in block 3 and finally the UPDATE task which is executed once every four times of recurrence of block 5, is based on the unique id inserted in block 4.

3.3.2. Defining and achieving a consistent level

Maintaining consistency throughout the evaluation is imperative. Before the measurement is started, the database will be initialised first (i.e. establishing connection, purging data, etc.). There is usually inconsistency behaviour observed at the point where the database begins execution of the process requested. To ensure an optimal level can be achieved where the data size of the database remains constant and handles maximum workload at a stable processing rate; warming up sequences (i.e. blocks 1 and 2) are introduced to build up workload and the processing rate of the database. Once the database reaches optimal consistency level, the measurement will be started and sequences of blocks (i.e. block 4 to 6) will be executed. These sequences are based on the distribution of tasks defined in Section 3.2. After the measurement is taken, blocks 7 and 8 are carried out to ensure the database performance is slowly brought down to standstill.

Table 2
Summary of 10 Test Results

	BerkeleyDB 4.3		BerkeleyDB JE 1.7	
	<i>Local Disk</i>	<i>Shared Disk</i>	<i>Local Disk</i>	<i>Shared Disk</i>
Total number of tasks/test	210,000	210,000	210,000	210,000
Average number of tasks/millisecond executed in 10 tests	105	105	6	0.5
Time spent/test (minute)	0:02	0:02	0:35	6:58
Standard deviation, σ	0	0	0	1

3.3.3. Maintaining maximum data throughput

It is considered that the true performance of the database could not be observed, if the size of the database were rapidly expanding and decreasing by a large margin throughout the evaluation period. Thus, measurement is taken only when the system starts to execute block 3 and ends after executing block 6. During this period, variation in the database size can be maintained within very small limits. The measurement taken is then considered the best to represent actual performance of the database under a live environment, where the database is constantly running.

3.3.4. Maintaining same mixture and randomness of the tasks

A single test framework was not considered a justifiable evaluation, as a maximum throughput of the database and true randomness in the mixture of the tasks could not thus be achieved. Multi-threading therefore was introduced to the evaluation system to ensure maximum throughput of the database and to ensure that randomness was achieved. Four threads were therefore introduced to the framework. Without such multi-threading, measurements taken from repeated evaluations would be invariable. Conversely, introducing the threading in the system meant that there could be only slight variation in the measurements taken should the same evaluation be repeated. Variation would exist mainly due to the randomness of multiple task requests created by the threading. There is a difference in time of the transitional period from one task to another. The difference in task combinations in this random environment contributes to the overall differences in the measurement. Observations could be based on these variations, when performance of the database might be considered inconsistent and unstable if huge differences among these variations were to be observed.

3.3.5. Evaluation examination

The examination of the evaluation result is based on the ten tests carried out for each of the test categories. The speed of the database to execute requested tasks is measured based on the average time it takes to complete the 10 tests. The sequence of tasks executed in each of the 10 tests is different due to the randomness introduced into the framework by the multi-threading. The standard deviation is taken for the 10 tests conducted to measure the level of variation of the performance of the database under different sequences of the tasks to check for the database consistency. If the standard deviation is high, we may conclude that consistency and reliability of the database is low, as the performance will have varied over the different mixture of tasks. Conversely, if the standard deviation is low, we may conclude that consistency and reliability of the database is high (e.g. Table 3).

3.4. Initial evaluation

3.4.1. Evaluation aspects consideration: BerkeleyDB JE 1.7 and BerkeleyDB 4.3

Sleepycat Software Inc. has released two versions of BerkeleyDB engines. BerkeleyDB 4.3 written in C++ with JNI interface to JAVA and BerkeleyDB JE 1.7 written in JAVA. A series of ten tests was

Table 3
Summary of 10 Test Results

	MySQL		TimesTen	
	<i>Non-prepared</i>	<i>Prepared</i>	<i>Non-prepared</i>	<i>Prepared</i>
Total number of tasks/test	420,000	420,000	420,000	420,000
Average number of tasks/millisecond executed in 10 tests	18	12	5	88
Time spent/test (minute)	0:23	0:34	1:15	0:05
Standard deviation, σ	0	0	52	11,921

Fig. 2. Performance evaluation for BerkeleyDB 4.3.

Fig. 3. Performance evaluation for BerkeleyDB JE 1.7.

conducted on each to assess performance of these engines when they reside in a local disk and a shared disk. These tests were based on the proportion of tasks for the SMS model in Section 3.2.

Table 2 shows the summary of 10 test results conducted for each test category (i.e. local and shared disks). The total number of tasks sent to the database was 210,000 for a test.

BerkeleyDB 4.3 performs 105 tasks in a millisecond for both disk locations, but BerkeleyDB JE 1.7 is only able to perform 6 tasks per millisecond in local disk and 0.5 tasks per millisecond in shared disk. Since standard deviation of each category of tests is between 0 and 1, it is shows that performance for the test using 210,000 tasks is consistent. The results of each test are shown in Figs 2 and 3.

BerkeleyDB JE 1.7 fails to maintain consistency and reliability in local disk and shared disk tests. In addition, the performance is a concern as the number of tasks executed by BerkeleyDB JE 1.7 is

Fig. 4. Performance evaluation for MySQL.

significantly lower than BerkeleyDB 4.3. In the network environment, databases are most likely to reside on a shared disk than local disk. This finding has confirmed the previously reported result [7]. Therefore, BerkeleyDB 4.3 should be selected among these engines and will be used as standard BerkeleyDB engine for SMS and MMS evaluations.

It is acknowledged that time taken for each category of test is too short to make a comprehensive comparison, but it is enough to demonstrate a large performance discrepancy between these engines.

3.4.2. Evaluation aspects consideration: Prepared and Non-prepared SQL statements for DBMS databases in JAVA

There are two ways to send a SQL statement in JAVA via JDBC – “prepared” and “non-prepared” SQL statements. A “non-prepared” statement is a SQL statement that needs to be validated before it is executed on the database server. A “prepared” statement will be validated and stored in the database memory in advance. Input parameters will be sent and executed without need to revalidate the whole SQL statement. A series of tests was conducted to assess performance of these statements for both MySQL and TimesTen Databases. These tests were based on the distribution of tasks for the SMS model in Section 3.2.

Table 3 shows the summary of 10 test results conducted for each test category (i.e. “prepared” and “non-prepared” SQL statements). The total number of tasks sent to the database was 420,000 per test.

For the “prepared” SQL statement, 12 tasks per millisecond were measured for MySQL and 88 tasks per millisecond for TimesTen. For the “non-prepared” SQL statement, 18 and 5 tasks per millisecond for MySQL and TimesTen. The standard deviation for both SQL statements in MySQL is 0, which is considered reliable. Meanwhile for the TimesTen, there is large standard deviation especially for the “prepared” SQL statement. This performance differences may due to the external overhead as TimesTen depends heavily on memory, which is shared by the operating system.

Performance of MySQL for both SQL statements was consistent throughout the duration of the evaluation with only minor differences among them (Fig. 4). TimesTen performs poorly with the “non-prepared” SQL statement but for the “prepared” statement it outperformed its own “non-prepared” SQL statement (Fig. 5) and both types in SQL statements of MySQL. Therefore, the “prepared” SQL statement should be adopted as a template for DBMS query instructions.

It is acknowledged that the time taken for each category of test is too short to make a comprehensive comparison, but it is enough to demonstrate a large performance discrepancy between these SQL statements. Selection for the “prepared” SQL statement is seen to be in favour of TimesTen, as MySQL

Fig. 5. Performance evaluation for TimesTen.

Table 4
Low Volume SMS Test, Summary of 10 Test Results

	MySQL	TimesTen	BerkeleyDB
Total number of tasks/test	1,050,000	1,050,000	1,050,000
Average number of tasks/millisecond executed in 10 tests	12	84	23
Time spent/test (minute)	1:24	0:12	0:45
Standard deviation, σ	84	3,476	2,951

performs better for the “non-prepared” SQL statement, but high performance of “prepared” SQL statement for TimesTen outscored the minor performance improvement observed in “non-prepared” SQL statement for MySQL. Future evaluation could be conducted to consider these differences.

4. SMS and MMS experiments

Observations and recommendations from the Sections 3.4.1 and 3.4.2 are implemented in the SMS and MMS experiments. Each experiment was divided into three critical areas, based on the different volume of traffic. There are; “Low Volume Test” to observe performance of the databases in low levels of data intensity and database size; “Medium Volume Test” to examine performance of the databases when they handle medium volumes of data intensity and database size; and “High Volume Test” to study databases performances under high data intensity and database size. These experiments are based on a proportion of distribution of tasks described in Section 3.2.

4.1. Evaluation of the SMS platform

Throughout this evaluation, data sizes inserted into the database were the maximum allowed by the length of each column. Tables 4, 5 and 6 show the summary results of 10 concurrent tests conducted for each database under different classifications of data intensity. 1,050,000 task requests were sent to databases for the low volume test, 2,100,000 tasks for the medium volume test and 4,200,000 requests were executed for the high volume test.

MySQL and TimesTen maintained around 12 and 85 tasks per millisecond respectively throughout three different data intensity tests. For the BerkeleyDB, performance was approximately 23 tasks per millisecond for the low volume test, 27 tasks per millisecond for the medium volume test and 10 tasks

Table 5
Medium Volume SMS Test, Summary of 10 Test Results

	MySQL	TimesTen	BerkeleyDB
Total number of tasks/test	2,100,000	2,100,000	2,100,000
Average number of tasks/millisecond of 10 tests	12	89	27
Time spent/test (minute)	2:50	0:23	1:16
Standard deviation, σ	80	2,738	2,870

Table 6
High Volume SMS Test, Summary of 10 Test Results

	MySQL	TimesTen	BerkeleyDB
Total number of tasks/test	4,200,000	4,200,000	4,200,000
Average number of tasks/millisecond executed in 10 tests	12	88	10
Time spent/test (minute)	5:48	0:47	6:33
Standard deviation, σ	160	2,000	158

Fig. 6. Performance evaluation for Low Volume SMS Test.

per millisecond for the high volume test. MySQL had the lowest standard deviation compared with the other two databases. It scored 84 for the low volume test, 80 for the medium volume test and 160 for the high volume test. TimesTen had the highest standard deviation in low, medium and high volume tests with a figure of 3,476, 2,738 and 2,000 respectively. Meanwhile, BerkeleyDB presented 2,951 and 2,870 for low and medium volume tests, but only 158 in the high volume test (as shown in Tables 4, 5 and 6).

Based on the results shown in Figs 6, 7 and 8, performance of BerkeleyDB degraded dramatically as the volume of tasks increased. There is a reliability issue for BerkeleyDB and TimesTen as the standard deviation for each test is high. But, TimesTen has overcome this problem by delivering a far superior service compared with other databases. Going on this evidence, “Main Memory based DBMS” is considered the most desirable choice as the data management strategy for SMS services, provided it has a third party or its own plug-in secondary persistent storage.

It is acknowledged that the standard deviation score is insignificant in relation to the high number of tasks executed by databases but it is enough to carry some weight when the time taken to complete the tasks among the different databases is considered.

4.2. Evaluation of the MMS platform

The MMS platform consists of two tables, the index and the data tables. It is considered a good strategy to find various database combinations in search of the best solution to manage index and data

Fig. 7. Performance evaluation for Medium Volume SMS Test.

Fig. 8. Performance evaluation for High Volume SMS Test.

tables for the MMS platform, rather than just keep to one type of the database for both tables. The SMS evaluation observations from Section 4.1 are considered essential in this selection process.

“Main Memory based DBMS” is considered unsuitable for the data table, owing to the limitation of the hardware (i.e. RAM) to store such a huge amount of data. “Main Memory based DBMS” or “Disk based DBMS” for index table will be combined with “Disk based DBMS” or “Disk based data store without DBMS”.

There are four viable combinations to consider, which are; MySQL for both index and data tables; TimesTen for the index table with MySQL for the data table; MySQL for the index table with BerkeleyDB for the data table; and finally TimesTen for the index table with BerkeleyDB for the data table.

As the message size (i.e. 100,000 bytes) to be inserted was not viable for the evaluation platform, data size inserted into the database for this table was reduced to 30,000 bytes.

Tables 7, 8 and 9 show the summary of 10 concurrent tests with the results conducted for each database under different classifications of data intensity. 370,000 tasks request were sent to databases for the low volume test, 1,100,000 tasks for the medium volume test and 1,850,000 requests were executed for the high volume test.

In the low volume test, the combination of TimesTen: BerkeleyDB (i.e. index: data table) was the fastest with 7 tasks executed in one millisecond. This was followed by TimesTen: MySQL and MySQL: BerkeleyDB in which both scored about 5 tasks per millisecond. MySQL: MySQL was the slowest

Table 7
Low Volume MMS Test, Summary of 10 Test Results

<i>Index Data</i>	MySQL MySQL	TimesTen MySQL	MySQL BerkeleyDB	TimesTen BerkeleyDB
Total number of tasks/test	370,000	370,000	370,000	370,000
Average number of tasks/millisecond executed in 10 tests	4	5	5	7
Time spent/test (minute)	1:26	1:18	1:11	0:54
Standard deviation, σ	119	43	223	631

Table 8
Medium Volume MMS Test, Summary of 10 Test Results

<i>Index Data</i>	MySQL MySQL	TimesTen MySQL	MySQL BerkeleyDB	TimesTen BerkeleyDB
Total number of tasks/test	1,110,000	1,110,000	1,110,000	1,110,000
Average number of tasks/millisecond executed in 10 tests	4	4	4	4
Time spent/test (minute)	5:07	4:14	5:23	4:40
Standard deviation, σ	140	386	1,021	1,415

Table 9
High Volume MMS Test, Summary of 10 Test Results

<i>Index Data</i>	MySQL MySQL	TimesTen MySQL	MySQL BerkeleyDB	TimesTen BerkeleyDB
Total number of tasks/test	1,850,000	1,850,000	1,850,000	1,850,000
Average number of tasks/millisecond executed in 10 tests	3	3	2	2
Time spent/test (minute)	10:44	9:38	17:57	19:07
Standard deviation, σ	178	296	46	103

with only 4 tasks per millisecond. Results under medium volume test have shown all the combinations recorded about 4 tasks per millisecond, but the standard deviation for the combination with BerkeleyDB as data table was high (over 1000). Combinations that used MySQL as the data table successfully executed 3 tasks per millisecond in the high volume test, but those using BerkeleyDB as the data table, where only able to complete 2 tasks in a millisecond.

Based on the results shown in Figs 9, 10 and 11, the implementation of MySQL as a data table is seen as faster than using BerkeleyDB. Although both MySQL and TimesTen managed to execute almost the same number of tasks as an index table, TimesTen is the fastest in executing all the tasks both using MySQL and BerkeleyDB as the data table, when compared with MySQL. Thus, using “Main Memory based DBMS” as an index table and “Disk based DBMS” as a data table is the most desirable combination data management strategy for MMS services, but it must be kept in mind that using two different databases as a solution is not always a wise choice.

It is acknowledged that the standard deviation score is insignificant in relation to the high number of tasks executed by databases but it is enough to carry some weight when the time taken to complete the tasks among the different databases is considered.

4.3. Special consideration: Evaluation of small storage systems

It has been seen that BerkeleyDB performs well when there is only a small volume of data involved (Sections 4.1 and 4.2). Although it is not generally desirable for data management strategies for SMS and MMS platforms, it is highly recommended for BerkeleyDB in handling small volumes of data where the need for data integrity is top priority. A test was devised to evaluate BerkeleyDB against “Main

Fig. 9. Performance evaluation for Low Volume MMS Test.

Fig. 10. Performance evaluation for Medium Volume MMS Test.

Fig. 11. Performance evaluation for High Volume MMS Test.

Memory based DBMS” – TimesTen and “Disk based DBMS” – MySQL. This test was based on the proportion of distribution of tasks for the SMS model (Section 3.2).

Table 10 shows the summary of 10 test results conducted for each database. The total number of tasks sent to the database was 210,000 for each test.

TimesTen was the fastest with 105 tasks executed per millisecond, BerkeleyDB with 101 tasks per millisecond, and finally MySQL was able to execute the requested tasks at the rate of 12 tasks per millisecond.

BerkeleyDB matched the performance of the “Main Memory based DBMS” (i.e. TimesTen) as shown in Fig. 12. For critical systems that require small persistent data storage, BerkeleyDB was the most

Table 10
BerkeleyDB Test, Summary of 10 Test Results

	MySQL	TimesTen	BerkeleyDB
Total number of tasks/test	210,000	210,000	210,000
Average number of tasks/millisecond executed in 10 tests	12	105	101
Time spend/test (minute)	0:17	0:02	0:02
Standard deviation, σ	17	0	11068

Fig. 12. Performance evaluation for BerkeleyDB Test.

suitable for the data management strategy.

5. Conclusion and future work

The observations produced from the various tests could be viewed as a guideline in selecting the best data management strategies that meet this design requirement. Selection of the data management platform often depends on the customer. DBMS has offered a few attractions as it has reached maturity for its data storage and maintainability solutions. In addition, it could be integrated into existing mobile messaging platforms with fewer man-hours. Thus, integration and maintainability cost will remain minimal.

For the customer, cost is often a top priority in the selection process. “Main Memory based DBMS” are best in terms of performance but not in terms of pricing, it is expensive to upgrade the memory in the system and license fee costs are high. The cost of database licences might also need to be taken into account, here BerkeleyDB license fees cost less compared with those of other database licenses. Upgrading the disk to a high specification HDD is a cheap option that may solve the performance issue with BerkeleyDB and “Disk based DBMS”. When only a small storage footprint is required, it is observed that BerkeleyDB is a good choice in terms of cost and performance when compared with “Main Memory based DBMS” and “Disk based DBMS”. The customer may not always need a high performance data management system and may be more concerned with the consistency, reliability and integrity of the system. “Disk based DBMS” is seen to present the best choice for this requirement.

Regarding the prospect of advancing mobile technologies, further review of the data management strategies should be conducted with consideration given to live video streaming for the mobile devices and clustering solutions for data management systems. Since recommendations given in this paper are aimed at high performance systems they may not be valid in other circumstances. New proposals,

therefore, should be made based on the results of these evaluations in order to meet any new system design requirement.

References

- [1] V. Ercegovac, *The TEXTURE Benchmark: Measuring Performance of Text Queries on a Relational DBMS*, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.
- [2] F. Haftmann, *Parallel Execution of Test Runs for Database Application Systems*, Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005.
- [3] M. Vieira, *A Dependability Benchmark for OLTP Application Environments*, Proceedings of the 29th VLDB Conference, Berlin, 2003.
- [4] M. Altýnel, *Cache Tables: Paving the Way for an Adaptive Database Cache*, Proceedings of the 29th VLDB Conference, Berlin, 2003.
- [5] T. Bourke, *Using MySQL to benchmark OS performance*, NewsForge, OSTG Inc., <http://software.newsforge.com/software/04/12/27/1238216.shtml?tid=152&tid=72&tid=29>, 2005.
- [6] Team, TimesTen, Mid-tier Caching: The FrontTier Approach, SIGMOD'02, Madison, 2002.
- [7] Sleepycat Software Inc., Managing Data within Billing, Mediation and Rating Systems, Sleepycat Software Inc, 2005.
- [8] T. Dyck, *Server Databases Clash*, eWEEK. Ziff Davis Publishing Holdings Inc. <http://www.eweek.com/article2/0,4149,293,00.asp>, 2002.
- [9] N. Hedlund, *Berkeley DB Transactional Data Store – Teligent*, Sleepycat Software Inc, 2003.
- [10] Jupitermedia Corporation, Developer.com Product of the Year 2004 Results Jupitermedia Corporation. <http://www.developer.com/lang/other/print.php/3307861>, 2005.
- [11] A. Okmianski, *Berkeley DB Transactional Data Store – Cisco*, Sleepycat Software Inc., 2003.
- [12] R. Anzarouth, *Berkeley DB Transactional Data Store – Mitel*, Sleepycat Software Inc, 2003.
- [13] Red Herring, Inc., RH-100 Europe: The Ikea of Databases, Red Herring Inc. <http://www.redherring.com/Article.aspx?a=11919&hed=RH-100+Euro+pe%3a+The+Ikea+of+Databases>, 2005.
- [14] ETSI, Technical realization of the Short Message Service, GSM 3.40 Vs. 3.5.0. European Telecommunications Standards Institute, 2003.
- [15] S. Lim, *Design and Implementation of LDAP Component Matching for Flexible and Secure Certificate Access in PKI*, IBM, 2006.
- [16] D. Lee, When location-based services meet databases, *Mobile Information Systems*, IOS Press, 1(2) (2005), 81–90.
- [17] R. Malladi, *Applying Multiple Query Optimization in Mobile Databases*, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), 2003.
- [18] H. Hu, *Towards Real-time Parallel Processing of Spatial Queries*, Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03), 2003.
- [19] K. Vorwerk, On Implicate Discovery and Query Optimization, Proceedings of the International Database Engineering and Applications Symposium (IDEAS'02), 2002.
- [20] J. Wu, *Scheduling of Query Execution Plans in Symmetric Multiprocessor Database Systems*, Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2004.
- [21] J. Pisharath, *A Window-Based Approach to Retrieving Memory-Resident Data for Query Execution*, Proceedings of the International Database Engineering and Applications Symposium (IDEAS'04), 2004.
- [22] A. Gog, Evolutionary Tuning for Distributed Database Performance, Proceedings of the 4th International Symposium on Parallel and Distributed Computing (ISPD'05), 2005.

David C.C. Ong has more than six years industrial experience in the fields of web-based services, application development, database design and mobile messaging services. He has worked as researcher and developer in the telecommunication industry specialising in the design of mobile messaging infrastructure for the mobile phone operators. Currently he works as a researcher at Kingston University under a joint research programme with one of the University's industrial partners. He has been a member of the British Computer Society (MBCS) since 2003. He holds a BSc (Hons) in Computer Science and an MSc in Data Communications. At present, he is reading for a PhD in Computer Science at Kingston University, London.

Rytis Sileika has worked in the IT industry for more than nine years. For the past five years, he has specialized in system installation and integration as a senior system integrator and platform architect in the telecommunications industry. He also has experience in planning and designing the configuration and installation of a wide variety of operating systems, clustering software, database software, and storage and networking subsystems. He has a BSc in IT with major in IT infrastructure design and implementation.

Dr. Souheil Khaddaj is a Reader and is leading the Component & Distributed Systems Research Group (CODIS) at Kingston University. He completed his PhD in Computer Science at Queen Mary and Westfield College, University of London. He has been active in research and development using novel computer architectures and technologies for numerous applications since 1990. His research interests cover diverse environments ranging from scalable high performance clusters to mobile computing devices with dynamic interaction of computing and information resources. He has been involved in a number of national and international research projects and has authored/co-authored over 50 technical papers.

Dr. Radouane Oudrhiri is a CTO of Systonomy. He has more than 18 years of teaching, research and consulting in software engineering, architecture, process and quality improvement methods and strategies, including Six Sigma, DFSS and SW-CMM. He acts as a Strategic Consultant and Architecture Authority. Radouane has wide experience of heterogeneous and Distributed Architectures, architecture standards, styles and methods. He designed and delivered end-to-end and led the production of multiple software packages within start-up and corporate environments in general and Telecom in particular. Radouane has an M.S. in Operations Research from ENSAE Paris, an MBA from ESSEC Paris and a Ph.D. in Information Systems and Theory from ESSEC and Universite d'Aix-Marseille. He is also a lecturer in software engineering at the French Engineering Telecommunication School, Telecom-Paris and an evaluator at the European Commission on Mobile Applications, Distributed Architectures and Software Process Improvement.