Study of Database Reliability and Performance in VCL

By Yeshwanth Kumaraswamy , North Carolina State University

Abstract—**This Independent study Report aims to solve certain problems experienced by the current VCL system, including response slowdowns when multiple users query the system for statistical information, Section 2 handles this part of the problem. We also address the problem related to backing up large amounts of data at frequent intervals of time, and as to how to decrease the backup turn around time of the database component of the system.**

The solution to these are discussed in Section 3.This report also highlights the major differences between MYSQL and PostgreSQL, and tries to arrive at a conclusion as to which one of these is best suited for VCL environment. The solution to this is presented in Section 4 of the report.

Finally, the problem of whether to distribute database services in a single VCL system is addressed in Section 5 of the report. I have tried to address these problems with the help of simulations, code and theoretical solutions. The actual code has been replaced by plain textual description, and the simulation results are included in form of a table.

I. INTRODUCTION

This part of the report aims to describe the functionality of the VCL system. The VCL system is a complex system which allows the user to reserve a computer with desired set of applications and remotely access the reserved system. This provides great deal of flexibility. VCL allows user to use popular applications like MATLAB,SAS etc and provides many simulation environments like OPNET. All that is needed to make use of this facility is a valid ID and an Internet connection.

Thousands of student access VCL daily, load on the System seems to increase significantly, and hence the ever increase need for performance enhancement. The future plans to extend the access to VCL to students who are not part of NC State University rings an alarm, and has raised several performance related issues which are discussed in the following sections. There are some major and minor changes as suggested in the rest of the report.

We aim to enhance the performance by a considerable extent. The current VCL system doesn't make use of stored procedure and hence increases the system load as the entire result set has to be returned. Hence, increases the memory demand and caused certain scripts to terminate prematurely. Many queries currently in use can be modified to perform better by making use of better operators and paraphrasing the query based on the nature of the query. The engine used with each of the table has also been modified depending on the nature of the tables and based on the nature of the data stored in them. Significant concurrency can be achieved using specific type of table, and decreases the response time.

A detailed comparison between the two leading open source database system is carried out on the basis of several parameters that are relevant to VCL system and a logical conclusion is derived as to which database system is better for VCL environment and if a transformation is desired is it feasible. Additional complexity is introduced as the database system is maintained by a third party and hence convincing them for such a transformation could be difficult.

Since the VCL is going to expand, by being open to other community colleges, the number of users will significantly increase in the near future. Hence it would make sense to distribute the database services. This part of the solution is discussed in Section 5 of the report. The objective of this report is to address all these problems and provide satisfactory solutions to each of the aforementioned problems. Section 6 of this report also predicts the possible outlook of VCL in the future, where the services is made open to different regions (divided into quadrants). This schematic diagram differs significantly from the cluster based model. This model involves lot of complication as presented in section 6, but it can also result in a very robust and efficient system designed to cater the needs of number of users.

Section 7 gives the conclusion and future work in this domain. Section 8 gives the list of all references both Web and bibliographical.

II. How to avoid and mitigate database reponse slowdowns when multiple concurrent requests on statistics are sent to the system.

A. The Problem

There are restrictions on the size of the script that can be in the memory at a particular point of time (say 32 megs), which results in termination of certain scripts hence resulting in an additional delay. The use of stored procedure obviates this problem, by ensuring only the result sets are returned.

One scenario that takes place in the VCL System involves an application that may need to accept input from the end user, read some data in the database, decide what statement to execute next, retrieve a result, make a decision, execute some SQL, and so on.

If the application code is written entirely outside of the database, each of these steps would require a network round trip between the database and the application. The time taken to perform these network trips can easily dominate overall user response time.

B. The solution

Stored Procedures, allowing to query to be executed on the server and the results to be sent back , thus avoiding the entire table from being loaded into the memory. Client/Server applications typically have to balance the load between the client PC and the relatively more powerful server machine. Using stored program is one of the ways to reduce the load on the client, which might otherwise get overloaded [14].

Network bandwidth is often a serious constraint on client/server applications, execution of multiple server-side operations in a single stored program could reduce network traffic. Also, maintaining correct versions of client software in a client/server environment was often problematic. Centralizing at least some of the processing on the server allowed a greater measure of control over core logic.

Stored programs offered clear security advantages, because in the client/server paradigm, end users typically connected directly to the database to run the application. By restricting access to stored programs only, users would not be able to perform ad hoc operations against tables and other database structures. Hence, stored programs can improve the security of your database server[6] .

Stored procedures add a extra *layer of Abstraction to the existing software application.* Which means that as long as the interface remains the same the underlying tables can change any number of times , without any noticeable change in the application. [1]

By using this layer of abstraction we can prevent a potential attacker from knowing the structure of the underlying tables. Hence data become safe from being exposed to the outside world[1]

Stored programs offer a mechanism to abstract data access routines, hiding your implementation behind a procedural interface and making it easier to evolve your data structures over time.

Stored programs can be used to implement functionality that is needed and can be called from multiple applications, and from multiple places within a single application.

Stored programs will reduce network traffic considerably. The evidence is this presented in the figure below. Another security advantage inherent in stored programs is their resistance to SQL injection attacks.

Carefully written stored procedures may allow for fine grained security permissions to be applied to a database. Each stored procedure written for mysql can contain security level as either definer or invoker, the former allows the stored procedure to be executed with the permissions set for the invoker, the latter allows the procedure to be executed with the permissions of the definer who wrote the code, so the procedure can access all the tables and data to which the definer is entitled. Hence, client programs might be restricted from accessing the database via any means except those that are provided by the available stored procedures [14].

Stored procedures allow for business logic to be embedded as an API in the database, which can simplify data management and reduce the need to encode the logic elsewhere in client programs. This may result in a lesser likelihood of data becoming corrupted through the use of faulty client programs. Thus, the database system can ensure data integrity and consistency with the help of stored procedures.

Some experts claim that databases should be for storing data only, and that business logic should only be implemented by writing a *business layer* of code, through which client applications should access the data. However, the use of stored procedures does not preclude the use of a *business layer*.

One scenario involves an application that may need to accept input from the end user, read some data in the database, decide what statement to execute next, retrieve a result, make a decision, execute some SQL, and so on. If the application code is written entirely outside of the database, each of these steps would require a network round trip between the database and the application. The time taken to perform these network trips can easily dominate overall user response time.

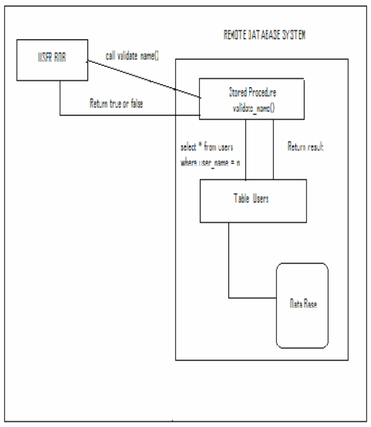
C. Figure

The figure below shows abstract functioning of a stored procedure here a user named Bob calls a stored procedure called validate_name. The stored procedure accesses the table users to find out if the name exists and return only the result hence doesn't eat up the memory.

It also mitigates the memory requirement as all the execution takes place on the server side and only the result set is being retuned. Hence no more premature termination of script will take place as the memory limit placed on the size of script will not increase. And since the only results are returned the network will be kept to minimum.

The procedure validate_name() can run directly within the database engine. In a production system, this typically means that the procedures run entirely on a specialized database server, which has direct access to the data being accessed. The benefit here is that network communication costs can be avoided completely. This becomes particularly important for complex series of SQL statements.

The diagram also depicts an important property of stored procedures that is the user is completely abstracted from the behavior of the code. This layer of abstraction is useful because it allows the underlying database to be changed as long as the interface is maintained. And it also allows security levels to be defined where the execution of the procedure takes place under invoker or definer rights.



D. Sample Implementation of Stored Procedure

The stored procedure can be called from our VCL code (which is written using PHP) using the mysqli interface. There are several methods available with the mysqli interface. Mysqli_query() is used to call the stored procedure from the PHP code. For example
$result = mysqli_query($link , “call view_stat(‘param’)”) .

The above result can be fetched row wise or into an array using mysqli_fetch_row() or mysqli_fetch_array().

The existing code has to be modified to use mysqli ineterface instead of mysql interface. Sample stored procedure in Statistics.php file

```
delimiter ##
create procedure view_stat (in userid char(10),pass varchar(225))
    reads sql data
   SQL Security Invoker
Begin

      declare done  int default 0;
      declare logged_in int default 0;
      declare uid char(9) default ‘ ‘
      declare pword varchar(225);
     declare err_cond int default 0;
declare continue handler for not found
begin
       set done  = 1;
end;

declare exit handler for 1048
begin
     set err_cond = 1;
 end

declare yash_cursor cursor for select  unityid , password from user;
open yash_cursor;

loop1 : LOOP
begin
fetch yash_cursor into uid,pword;

start transaction;
SAVEPOINT yashpoint;
if strcmp(uid,userid) = 0 and strcmp(pass,pword)=0
begin

      set logged_in = 1;
      leave loop1;

end;
end if;

if done == 1 then

leave loop1;

end if;

if err_cond  == 1 then
begin
     select concat(“Script Terminating”);
     leave loop1
     rollback to yash_point;;

end;
end if;
```

end loop1;

close yash_cursor;

select logged_in;

commit;

end##

The above code shows a simple procedure that will check if the unity id and password entered match that in the user table. A simple cursor is defined to loop through the user table extracting user name and password , then a simple comparison is made to determine if the user is present in the table or not an whether his password matches or not. A continue handler assists in breaking out of the loop when the condition not found is met (i.e. end of data).A exit handler has been defined for some error condition 1048 and sets the variable, and a suitable action is taken. A status message is printed and the transaction is roll backed and the script is terminated. The variable is set to indicate that the error has occurred.

This code can be called using the syntax shown in start of part D. Many of the queries can be turned into calls to stored procedure. As we see the security level has been set to that of the invoker, hence the procedure executes with the privilege of the account executing the procedure.

E) Other Ways of improvement:

i) Database Engine

Using different engines for different tables depending on the usage of the respective tables. The two main engines used are InnoDB and MyISAM , apart from these we have another engine type which finds specific relevance in VCL system is MyISAM merge table.

Tables with select can be transformed into MyISAM engine . If the application requires only fast row counts then MyISAM is more efficient. Tables which involve many insertions (example **Query log table**) can be transformed into MyISAM to avoid bottle necks.

Also the **myisampack** utility compresses MyISAM tables. **myisampack** works by compressing each column in the table separately. Usually, **myisampack** packs the data file 40%-70% .When you later have to use the table , the server reads the table into the memory the information needed to decompress the columns.

This results in better performance as only those selected rows have to be decompressed . You can use SELECT, DELETE, UPDATE, and INSERT on MERGE tables. You must have SELECT, UPDATE, and DELETE privileges on the MyISAM tables that you map to a MERGE table.

Obtaining more speed is one of the major advantages of. You can split a big read-only table based on some criteria, and then put individual tables on different disks. Also, performing repairs become easier, in comparison to one large table (Log Table in this example).

I have concluded that if the following changes are introduced in the table engine types, then performance and concurrency can be significantly enhanced.

Contenders for MyISAM – Image table , OS Table , query log ,User Table

Contenders for MyISAM merge table – Log Table

Mr. Peeler and Mr. Thompson have promised to implement the this particular feature in the near future.

ii) Index and view Usage

Usage of index to quicken search , to modify the existing indexes in order to maximize the efficiency of the querying process. The solutions presented here will be corroborated with the help of experiment results which ran against sample data. I have used using the output of explain select command (run using phpMyAdmin) to determine if the changes did really bring in any improvement in terms of ***space or time*** .

The query can be executed faster if the optimizer knows how to eliminate all the unwanted rows and just look into a small subset of total rows which would match the search condition , because more faster it eliminates as many rows as possible ,so that it will be able to return rows that match your criteria. This remains the primary purpose of the optimizer. Queries can be executed much fast if the most restrictive tests are performed first.

Lets take an example to drive this point through considering the table `BlockComputers`. This table has two columns both of which are of type INT.
And we have a combined Primary index defined on the two columns of this table .

Now if we execute the query

Select comp_name from `BlockComputers` where blockTimeid = value1 and computerid = value2;

Now if the blockTimeid column matches 700 columns and computerid column matches with 300 values and together these two test produce only 30 columns as the final output. Now lets look how the optimizer would go about it , if it had chosen to compare blockTimeid it would had to make 700 tests which produce 670 failures and if it had chosen to start off by comparing computerid it would have just had 270 failures tests. Hence the optimizer would choose to go about comparing the second column (computerid).

We can help the optimizer make such decisions by using index wisely. If we plan to set up a combined index as mentioned in the example above make sure that both the columns are of the same type , this would ensure better performance than using two columns with dissimilar types.
Another way to enhance the performance query of the index, is to ensure that if any queries involving arithmetic operations uses any indexed column as a part of the query then they should be made stand alone , the following example should make things clear.

If we had to make a decision as to whether the blockTimeid is less than 2 (sample query doesn't exists in the database)

Select blockTimeid from `BlockComputers` where computerid < 4 / 2 ;

In the above query the optimizer correctly fetches only those rows where the computerid is less than 2. If the query had been written as follows then

*Select blockTimeid from `BlockComputers` where computerid * 2 < 4 ;*

Then the optimizer would have fetched all tuples and multiplied it by 2 and then made a comparison to extract all those rows with value less than 4. The purpose of defining the index is defeated.

Hence the way we frame the query plays an important role even if we have the right indexes defined on them, the index's purpose may be nullifies as the optimizer cannot use them.

The same concept applies when making date comparison as well. Consider the following query defined in function getStatGraphHourData

SELECT l.start AS start, l.finalend AS end
FROM log l, sublog s, user u
WHERE ***year(l.start) < '$endyr'*** *AND*
year(l.finalend) > '$startyr' *AND*
s.logid = l.id AND
s.computerid IS NOT NULL AND
l.userid != $reloadid AND
l.userid = u.id AND
u.affiliationid = {$user['affiliationid']}

The query is used to find reservations made by a particular user with a specific affiliation ID. We extract the year from start and end date and make comparisons of all the reservations made by him/her.

Here we have indexes defined on the two columns *start* and *end.* Since we have an operation being performed on it (i.e. year()) the optimizer will not apply the index for this query , as first all the rows will be retrieved and then the operation will be performed on it (i.e year(start)) then only the comparison will be made . A remedy for such a query would be to use date instead of year only.

Indexes use tends to reduce to null and void if the query involves regex. Hence we should try to use the regex operators prudently if we want the optimizer to use the index effectively.

Another possible optimization is to use the features that the optimizer has been best tuned for. For example as we are using MySQL ,and its optimizer has been more optimized joins as sub queries are recent additions , hence the optimizer has not been tuned for it.

The use of index can also be nullified if the query involves usage of auto type conversion.
For example

*i) Select * from BlockComputers where blockTimeid = '4' ;*

*ii)Select * from BlockComputers where blockTimeid = 4 ;*

The first version involves the usage of auto type conversion hence optimizer will not use the index , the second one doesn't use the auto type feature , hence more effective.

We can use explain to determine if the indexes defined on the columns are effective in producing the output of the query.
Say we use the following queries as a replacement for the part of the query marked in bold

i) Select TO_DAYS(l.finalend) into constant – TO_DAYS(now())

ii) WHERE TO_DAYS(l.start) - TO_DAYS(CURDATE()) < *constant*

iii) WHERE TO_DAYS(l.start) < *cutoff* + TO_DAYS(CURDATE())

iv)WHERE l.start < DATE_ADD(CURDATE(), INTERVAL *constant* DAY)

Here the part ii) of the query cannot use index as optimizer cannot use the index as TO_DAYS(l.start) which requires all the rows to be pulled out before the comparison has been performed. The output of the select would look like

explain select * from BlockComputers where TO_DAYS(l.start) - TO_DAYS(curdate()) < *constant*

```
id: 1
select_type: SIMPLE
table: BlockComputers
type: ALL
possible_keys: NULL
key: NULL
key_len: NULL
ref: NULL
rows: 5
Extra: Using where
```

The type field says 'all' which means index is not being used (all records are being examined). The key field shows NULL meaning no index is being used.

The output of explain select for part iii)

Explain select * BlockComputers from
where to_days(expiration) < constant + to_days(curdate())

```
id: 1
select_type: SIMPLE
table: BlockComputers
type: ALL
possible_keys: NULL
key: NULL
```

```
key_len: NULL
ref: NULL
rows: 5
Extra: Using where
```

The first two outputs shows that indexes were not used for the reasons specified above.

The output of explain select for part iv) looks like

```
Explain  select  * from BlockComputers
where constant  < date_add(cur_date(), INTERVAL
30 day)

         id: 1
  select_type: SIMPLE
       table: Block
        type: range
possible_keys: comb_index
         key: comb_index
     key_len: 4
         ref: NULL
        rows: 3
       Extra: Using where
```

Let us apply the concepts discussed above in using indexes prudently in VCL.

The optimizer tries using the indexes whenever possible

a)
The primary index in table Block Computers can be replaced with combined index to facilitate faster searching.

After we execute the statement

```
drop index block_comp_index on
BlockCompouters;


create index new_block_comp primary
on
blockComputers(`blockTimeid` , `computerid`)
```

the efficiency of such a change was proven with a sample explain select on `BlockComputers` table on Page 7.

b)
Transforming the indexes found in `BlockRequest` Table into Foreign Key indexes. This index will also serve to maintain the foreign key constraint as efficient as possible.

Views can be defined on tables that are queried frequently and updated rarely, and stored procedure can be made to query these views based on certain conditions. For example the following view spreads across four tables.

CREATE ALGORITHM=UNDEFINED VIEW `viewstat`.`v1` AS
SELECT

l.userid,l.nowfuture,l.start,l.initialend,l.finalend,l.wasavailable,
l.ending,l.ending,i.prettyname,o.prettyname AS OS

FROM log l,image i, user u,OS o WHERE i.id = l.imageid AND i.OSid = o.id AND l.userid = u.id

ORDER BY i.prettyname;

iii) Changes to system variables

Another Performance enhancing features associated with MyIsam tables is the ability to delay writing the index data back to the disk. Mysql usually flushes the modified key blocks back to the disk after making changes to them, but there is a flexibility to modify this default behavior , this can prove advantageous where there are excessive insert , delete or update activities. This facility is available via the modification of the tristate variable (**delay_key_writes**) This facility is only advantageous only if we have ***clustered database servers , or else the data may be out of sync , in which case repair table has to be used.***

iv) Query Modification

Many Queries which aren't replaced by stored procedures can be replaced by queries which perform better after having undergone some simple modification. The results of these sample modification will be provided in form of execution time for different input sets.

Query optimization for a stream of queries using query graph has been described in [9], which is used to support common sub-expression detection in a stream of request and it tries to optimize it as a group rather than individual query.

Before we suggest any changes pertinent to code changes in the existing VCL code , we have to look at the way the Query is optimized by MySQL,

This was described in the previous part i.e. pp 6-8 The following code snippet shows how the performance of a query can be improved by replacing queries with Cartesian products with join operators (like natural, left outer, right outer etc…),this modification is applicable to number of queries in the actual VCL code. For example the query in getStatGraphDayData can be replaced by

```
SELECT count(l.id) FROM log l JOIN user u ON
l.userid = u.id where l.start BETWEEN '$startdate'
and '$enddate'
AND
l.userid = u.id AND
u.affiliationid = "{$user['affiliationid']}";
AND
l.userid != $reloadid ;
```

Simulation Run	*Run time (in sec) Before modification*	*Run time (in sec) After modification*
1	*0.0008*	*0.0005*
2	*0.0008*	*0.0006*
3	*0.0007*	*0.0004*

Usage of built in function also serves as an improvement. For example the query in `getStatGraphHourData` can be improved as follows

```
Select l.start as start , l.finalend as end
From ( log l NATURAL JOIN user ( userid ,
unityid …) as u )
OUTER JOIN sublog s on s.logid = l.id
WHERE l.start > '$enddt' and l.finalend >
'$startdate'
and nullif (s.computerid,NULL) and u.affliationid
= "{$user['affiliationid']}";
```

On similar lines many queries can be modified.

Simulation Run	*Run time (in sec) Before modification*	*Run time (in sec) After modification*
1	*0.016*	*0.07*
2	*0.016*	*0.08*
3	*0.015*	*0.07*

Another obvious improvement to execution of a query is to use the mysql **query_cache** (appropriate modification has to be done to my.cnf (**query_cache_type**), indicating the exact size of the query cache).This can be overridden (in case you don't want to cache the query) , by explicitly adding **SQL_NO_CACHE** to your SQL statements. When appropriate modification has been made to the configuration file indicating you want cache queries, the mysql first looks at the cache for an exact match without trying to analyze or execute it, if found the result is immediately returned. The format in which the result is returned from the cache is similar to the way the query is returned to the client , hence very little additional efforts is required.Mysql uses the hashed value of the query when looking for a match in the cache.

[Note : The solution specified above couldn't be tested as I don't have access to the my.cnf file , but I am sure that modifying the size of cache will definitely avoid frequent re-fetching of instructions (there will be a cache hit when it comes to frequently executed queries) Hence there will be a significant improvement in response time.]

III. A WHICH PROCEDURES MUST BE IN PLACE FOR BACKING UP STORED DATA .

1) **Backing up Stored Data**

DUMPS

When the database it set up , and the tables have been created (data has been inserted) , **mysqldump** can be performed by the administrator , it will be more efficient to create a **shell script** which will execute mysqldump and names the back up files based on the current date and time. The following shell script will be executed every hour by manipulating **crontab**.

```
#!/usr/bin/env bash
dbname="vclDBresearch"
user="root"
password=" "
mysql_bin="/usr/local/mysql/bin"

 #Generate a random file name every time  using
#the date function

filename="/Users/yash/Everything/EverythingShe
ll/yash"` date          +%D%H%M%S`".sql"

#call the dump routine by supplying appropriate
#parameters each time

${mysql_bin}/"mysqldump  $dbname  -u  $user
password=$password    > $filename
```

The crontab can be scheduled to run this shell script (**BackupDB.sh**) say *every hour* .

The changes required to the **crontab**

```
@hourly  /Users/yash/BackupDB.sh
```

This creates the backup of the database identified by db_name into a file very hour .This file would contain all the table definitions, along with all the commands required to lock the tables.

If we find ourselves running **out of space** we can always delete the dumps. Rather than always deleting the oldest back up we can delete those dumps that fall on odd days.

It also promotes Auditing of the previous dump files, which will be required in case we want to analyze the cause for ***Data Corruption*** , to examine the ***rate of growth*** of your database or any such reasons.

We can perform dumps **over the network** so that your backups are created on a host other than our database server.

A sample raw dump of a database named 'Wolf Express'

```
-- MySQL dump 10.13
--
-- Host: localhost    Database: WolfExpress
-- ------------------------------------------------------
-- Server version    5.1.22-rc-community

/*!40101                                    SET
@OLD_CHARACTER_SET_CLIENT=@@CHAR
ACTER_SET_CLIENT */;
/*!40101                                    SET
@OLD_CHARACTER_SET_RESULTS=@@CH
ARACTER_SET_RESULTS */;
/*!40101                                    SET
@OLD_COLLATION_CONNECTION=@@COL
LATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103                                    SET
@OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
```

```
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `categories`
--

DROP TABLE IF EXISTS `categories`;
SET @saved_cs_client = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `categories` (
 `id` int(6) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
SET character_set_client = @saved_cs_client;

--
-- Dumping data for table `categories`
--

LOCK TABLES `categories` WRITE;
/*!40000 ALTER TABLE `categories` DISABLE KEYS */;
INSERT INTO `categories` VALUES (1,'Snacks'),(2,'Beverages');
/*!40000 ALTER TABLE `categories` ENABLE KEYS */;
UNLOCK TABLES;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

    Dump completed on 2007-10-12 20:21:17
```

The **disadvantages** of this method lies in the fact that the dumps produced takes up more space than the actual table and data. The second reason is that all the numeric data is converted to ASCII contributing to the increase in space usage. Dumps are more CPU intensive hence they take more time than other methods.

Raw Backups

Raw backups are much faster than dumps because the data is presented in the native form and doesn't involve conversion to ASCII. MySQL offers the following commands for raw backups namely mysqlhotcopy , mysqlsnapshot.

Mysqlhotcopy is a perl script that is included in standard mysql distribution. It is efficient in backing up large databases (consisting of InnoDB and MyIsam tables).It is very popular and is used for producing online raw backups. It works by obtaining read locks on all tables it is supposed to backup. The **disadvantage** of using this is that it doesn't scale very well when the traffic increases.
Restoring the files backed up (.MYI , .MYD , .FRM) using mysqlhotcopy , involves copying the appropriate files into the correct subdirectories.

If we have large number of InnoDB tables then **ibbackup** is the best tool. http://innodb.com/hot-backup/

MysqlSnapShot is yet another tool used for raw backup. The draw back is that it can only be used for MyIsam tables.

```
$ mysqlsnapshot -u root -p Password -s /tmp/snap --split -n
```

This produces separate tar file for each database in the system.

There are tools to perform **offline backups** but that would be irrelevant in our case as the mysql server is not managed by us.

As we have discussed various ways by which back up can be performed , but each method has its own pros and cons. We can improve this by developing a script which traverses every table in a database , determines the kind of engine it utilizes and call appropriate routine.

If the *log table* is implemented as a MyISAM merge table , then the size of those tables can grow very fast in which case it may be desired to free up some old obsolete data or move them to external storage, the following pseudo code (actually a perl script transformed into textual description).

The following textual description computes the table sizes , and performs a check to see if the size of the table is exceeding a threshold. If so it alerts the administrator to take appropriate action.

```
#definition of variables

cmd = " ";  #The mysql back up command
tname, ttype; #holds table name and type
table_sizes; #array that holds the table sizes
dbh;  #database handle

#the file which holds the old table names and sizes

temp_file = “/tmp/temp/yash”;

## Contains alert message for the Admin

Pseudocode subroutine
connect(db="VclDbresearch", host = "localhost" ,
user="root",  password=" " )

#connect to database
Begin
      dbh=connect(db , host , user , password)
end

Pseudocode subroutine  disconnect(dbh)
Begin
#disconnect  from database

dbh.disconnect or warn "Disconnection error:
end

Pseudo code subroutine  mail_admin()
Begin

# Mails the table sizes and warning message to the
administrator

sendmail = ‘/usr/sbin/sendmail –t’;

if (send == “ “) then
begin

  print “Enter the to_address “ ;
  exit;

end if

Verify if the address is valid

Set up the variables required to mail the admin

rep_to = “Reply-to:admin@ncsu.edu\n”;

send_to = “To:” . $send . “\n”;

subject = “Subject :” . $sub . “\n”;

open (SENDMAIL,” | $sendmail”)
```

```
print  SENDMAIL $reply_to;
print  SENDMAIL $subject;
print  SENDMAIL $send_to;
print SENDMAIL "Content-type:text/plain\n\n";
print SENDMAIL $content;
close SENDMAIL;

end

sub sort_table_res( )
begin

  Sort the tables result

end

# Find out tables names and their Engines ,
ExecuteQuery("USE INFORMATION_SCHEMA");

Prepare("Select table_name,ENGINE from TABLES where table_schema like 'vclDBresearch');

# Execute the query

table_in = ExecuteQuery();

foreach table tb (table_in)
begin

     System(cmd);

 End for

#Ignore table with Engine Heap As they don't reside in the disk

 Else if ( tb  eq "Hash")
 Begin

    Next;

 end


# flush and lock all tables
# All MyISAM tables
ExecuteQuery("FLUSH TABLES WITH READ LOCK");

ExecuteQuery("USE $db");

table_info = ExecuteQuery("SHOW TABLE STATUS");

#Open the file and dump the contents in it
#These values can be used as comparision
# to see % growth in  the table size
#if it has exceeded a threshold then alert the admin

 foreach table tb in(table_info)
begin

      name = table.Name;
      size =   table.Data_length;

end for

# The file temp which hold the data of table names and sizes is created

Open the temp file and write the information of table name and sizes into it

Read in the old table size for each table in the file

Read in the new table size for each table in the array

  # I assume the number of tables don't increase for simplicity

Find the change in size of table

per_change = int ( $sizenew - $sizeold) / $sizeold;

if ($perchange > $threshold)
begin
      #Prepare the content for admin

       text = text + "The table " + tabnew .
       + "Has a size of " + sizenew . "\n";

End if
```

Call mail_admin (text , send_to , $sub , $reply_to);

#Execute the Backup

system(cmd);

# unlock the tables

ExecuteQuery("UNLOCK TABLES");

# sort by size and print

sort(table_sizes);

_ _END_ _

IV. B SYNCHRONIZING DATABASE IN MULTIPLE DATABASE SYSTEMS, AND DECREASING BACKUP TURN AROUND TIME OF DATABASE COMPONENT OF THE SYSTEM.

The first two solutions are theoretical and hence were not actually considered as viable solutions. The third solution recommends the usage of MySQL Cluster which takes care of all the requirements mentioned in the topic. Its features are mentioned clearly in the part c).

a) Replication

Master Slave model can be used where the slave holds the exact copy , and can take over in case the master fails. This works as follows, the master maintains a binary log of all queries which modify the data in the database ,and this is sent across the network to the slave . The slave executes these set of queries against its local copy of data , hence allows consistency to be maintained between the local copy and the server copy.

Using a creative DNS setup , the applications can be insulated from knowing whether they are being served by the master or the slave , and minimizes the effort involved in switching from master to slave in case of failure. Say we have our mysql master server running at say www.mysqlserver1.com and slave running at www.mysqlserver2.com , instead of hardcoding the address of the master into our PHP applications , we can set modify the DNS records in the server , by adding www.mysqlserver.com as the **Canonical Name** and by using a very small value for TTL , we can ensure that the information is not cached in the clients machine for a long duration. In case the mysql server in the master node crashes, we can have a script which automatically modifies the DNS records so that www.mysqlserver.com now points to the slave server. When the TTL expires the client would automatically start referring to the new slave server.

b) Load Balancing

This involves having one **master** server and several **slave** servers. The catch here is finding a way to efficiently distribute the query across several slave servers, so that all them get equal workload. The simplest approach will be to use Round-Robin DNS using which we can assign multiple IP addresses to one hostname, and make sure that our application connects to one random slave server.

c) MySQL Cluster

Most of the requirements discussed above namely High Availability , reduction of downtime, automatic detection of failure and recovery from it etc … are provided by **MySQL Cluster**, which combines ***mysql*** with a ***fault tolerant database clustering architecture***.

Some of the major features of **MySQL Cluster** are

- It ensures high availability and guarantees of less than 5 minutes of down time a year, including scheduled maintenance. MySQL Cluster implements automatic node recoverability to ensure an application automatically fails over to another database node that contains a consistent data set, if one or more database ndes fail. Should all nodes fail due to hardware faults for example, MySQL Cluster ensures an entire system can be safely recovered in a consistent state by using a combination of checkpoints and log execution. Furthermore, MySQL Cluster ensures systems are available and consistent across geographies by enabling entire clusters to be replicated across regions.

- MySQL Cluster provides the response time and throughput to meet the most demanding high volume enterprise applications. MySQL Cluster achieves its performance advantage by being a main memory clustered database solution, which keeps all data in memory and limits IO bottlenecks by asynchronously writing transaction logs to disk. It also enables sharing of processing within a cluster.

- MySQL delivers extremely fast failover time with sub-second response so your applications can recover quickly in the event of application, network or hardware failure. MySQL Cluster uses synchronous replication to propagate transaction information to all the appropriate database nodes so applications can automatically fail over to another node extremely quickly.

- The parallel server architecture combines database nodes, management server nodes, and application nodes that can be distributed across computers and geographies to ensure there is no single point of failure. Any node can be stopped or started without stopping the applications that use the database.

- MySQL Cluster is designed to be largely self-governing so very few system parameters actually need fine-tuning, further reducing the risk of costly errors. As a result, there are typically fewer conflicts with other software and hardware, and less need for manual intervention.

- MySQL Cluster includes easy to use and powerful tools for administering your clustered environment. Command line tools enable you to monitor database nodes, control access to applications, and create and restore backups.

d) Decrease back up turn around time

Two solutions were initially proposed but only the second was accepted.

i) Server crashes is a possibility in which case it is essential to get the system back in working state as soon as possible. One of the popular ways of doing that is via the use of **hot – swappable RAID disks** ,not supporting the RAID means that the server can handle one disk crash , but the system has to be shut down to replace the bad disk.The best way to overcome this problem would be to use the spare disk (RAID) in case of failure. But one disadvantage associated it with that you are sacrificing performance efficiency as well as redundancy.

ii) Mysql Cluster can be used for the reasons mentioned below :

a) It ensures high availability and guarantees of less than 5 minutes of down time a year, including scheduled maintenance. MySQL Cluster implements automatic node recoverability to ensure an application automatically fails over to another database node that contains a consistent data set, if one or more database nodes fail. Should all nodes fail due to hardware faults for example, MySQL Cluster ensures an entire system can be safely recovered in a consistent state by using a combination of checkpoints and log execution. Furthermore, MySQL Cluster ensures systems are available and consistent across geographies by enabling entire clusters to be replicated across regions.

b) MySQL Cluster is designed to be largely self-governing so very few system parameters actually need fine-tuning, further reducing the risk of costly errors. As a result, there are typically fewer conflicts with other software and hardware, and less need for manual intervention.

c) MySQL delivers extremely fast failover time with sub-second response so your applications can recover quickly in the event of application, network or hardware failure. MySQL Cluster uses synchronous replication to propagate transaction information to all the appropriate database nodes so applications can automatically fail over to another node extremely quickly.

V. WHAT ARE THE ADVANTAGES IF ANY OF USING POSTGRESQL INSTEAD OF MYSQL IN VCL ARCHITECTURE .

Open Source database programmer have had a lot of options when it comes to choosing their database management systems. But, the two major open source relational databases are PostgreSQL and MySQL. Both of these relational management systems that are available for free for download. The features provided by postgreSQL are slightly more advanced than that provided by MySQL these will be listed in the following section. There has been an ever raging battle between these two open source giants.

The database system for the VCL system is maintained by a third party. Hence migration has to be justified strongly to convince them for such an act.

I shall bring out some of the basic differences between postgreSQL and MySQL and then we shall discuss which one of them is best suited for VCL environment.

It would be very difficult to generalize which one of these two would be better, but if the exact scenario in which the application will operate is given , then we will be able to draw some valid conclusion regarding this dilemma.

I will enumerate the basic differences between the two based on the following parameters.

Let us take these parameters and some code specific aspects to arrive at a conclusion regarding which one of them is the best for our VCL application and discuss about the feasibility of migration (if we decide to choose PostgreSQL).

PostgreSQL offers some advanced features that are absent from mysql we will examine each of these in detail ,this will aid us in arriving at a conclusion.

i) SQL standard and compliance :

PostgreSQL understands a good subset of SQL92/99 plus some object-oriented features to these subsets. Postgres is capable of handling complex routines and rules as declarative SQL queries, subqueries, views, multi-user support, transactions, query optimization, inheritance, and arrays .Does not support selecting data across different databases[13].

MySQL uses SQL92 as its foundation. Runs on countless platforms. Mysql can construct queries that can join tables from different databases. Supports both left and right outer joins using both ANSI and ODBC syntax. As of MySQL 4.1 from that release on, MySQL will handle subqueries. Views supported as of release 5.

Outcome

Mysql query optimizers performs an extensive search of all the query evaluation plan to determine the best plan. As far as join queries are concerned the number of possible plans investigated by the MYSQL grows exponentially. This fact shouldn't deter us as the number of tables involved in join queries are kept minimal in our queries.
Mysql supports Stored Procedures , Functions and Triggers as well , like PostgreSQL.

Here postgreSQL is marginally better in terms of features like object-oriented aspects , handling complex routines etc... .As our application doesn't involve any sequel code referring tables spread across several databases, the constraint of Postgres shouldn't bother us.

ii) Platforms :

PostgreSQL lacks binary distribution for all the supported plataforms. One of the problems is that

PostgreSQL doesn't run properly on NT as a service by default, you need something like firedaemon to start it. The PgAccess GUI is available on windows as well, but it lacks a few features that psql supports. Non-supported platforms: Windows9x/Me, NextStep, Ultrix.

MYSQL has binary distribution for most of the supported platforms. MySQL works better on Windows than PostgreSQL does. MySQL runs as a native Windows application (a service on NT/Win2000/WinXP), while PostgreSQL is run under the cygwin emulation.

Outcome

Mysql is better than Postgres in this regard due to it versatility.

iii) Speed

PostgreSQL slower on low-end but has some options for improving. Postgres forks on every incoming connection - and the forking process and backend setup is a bit slow, but one can speed up PostgreSQL by coding things as stored procedures. But postgres is very good at holding large loads[12].

MySQL is very fast on both simple and complex SELECTs, but might require changing the database type from MyISAM to InnoDB for UPDATE intense applications.Mysql proponents claim that the time to execute read only task is very less in mysql than postgres.Hence they are very efficient in application which involve large amount is read. MySQL handles connections very fast, thus making it suitable to use MySQL for Web - if you have hundreds of CGIs connecting/disconnecting all the time you'd like to avoid long startup procedures.

Outcome

Here again I feel MYSQL is better than PostgreSQL as our application will cross the 2 million users mark (and the number is expected to increase as our application is going to be made public with website opening up to community schools etc.) Hence speed becomes an important factor.Mysql is versatile in handling incoming connection requests quickly and hence making it ideal for number of users intensive application like VCL.

iv) Stability

PostgreSQL 6.x series and earlier were much worse in this aspect. Random disconnects, core dumps and memory leaks are usual. PostgreSQL 7.x series was a big improvement. Expect PostgreSQL 8.x to continue this trend.

MySQL does very good job even on the busiest sites; it certainly has some problems handling hundreds of connections per second, but these problems are resolvable. Random disconnects and core dumps are exceptionally rare. MySQL has a much larger user base than PostgreSQL, therefore the code is more tested and has historically been more stable than PostgreSQL and more used in production environments.

Outcome

Mysql has another clear advantage over PostgreSQL. http://www.vcl.ncsu.edu clearly qualifies as a busy site. Hence a more stable database system like mysql is essential , rather than postgres which suffers from random disconnects, and since mysql has been in use more widely in various environments , hence it is clearly better in terms of stability than Postgres.

v) Data Integrity

A very preliminary quality that must be satisfies by a good database management system. It is the basic task of any system to guarantee safe storage and retrieval of data i.e. the contents shouldn't be manipulated by the database in an unexpected way. PostgreSQL pays a lot of attention to see through that the ACID properties are maintained, and hence ensures a consistent state of the database all the time. Postgres also supports the concept of WAL (write ahead logs) , this feature enable it to recover the database to a to any point in history[12].

Postgres does very good job supporting referential integrity, has transactions and rollbacks, foreign keys ON DELETE CASCADE and ON UPDATE CASCADE.

Mysql has a indifferent view towards ACID properties. Mysql has some basic provisions for referential integrity and transactions/rollbacks. CHECK clause is allowed for compatibility only and has no effect on database operation. InnoDB tables have FOREIGN KEYs for relational or multi-table delete, and support transaction processing. In MySAM tables FOREIGN KEY is for compatibility only and has no effect on database operation[12].

Outcome

PostgreSQL is better than mysql in this regard, primarily because of its strict adherence to ACID properties and Postgres is little more strict when it comes to acceptance of data than MySQL. This can prove to be very harmful. The following example shows how this leniency of Mysql can result in it accepting invalid data.

Example [13]

```
mysql> create table foo(a date);
Query OK, 0 rows affected (0.08 sec)

mysql> insert into foo (a) values('2007-feb-30');
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> select * from foo;
+------------+
| a          |
+------------+
| 0000-00-00 |
+------------+
1 row in set (0.00 sec)
```

The same scenario in postgres would have produced the following results

```
atf=# insert into foo (a) values('2007-feb-30');
ERROR:  date/time field value out of range: "2007-feb-30"
```

Comparing all the cases in defense of data integrity postgreSQL is a clear winner.

vi) Special Server Side Features

Postgres has rules, triggers, server-side functions that can be written in C, pgsql, python, perl and tcl languages. INSTEAD OF rules can be used for updating data through views. PostgreSQL has schemas that allow users to create objects in separate namespaces, so two people or applications can have tables with the same name. There is also a public schema for shared tables. Table/index creation can be restricted by removing permissions on the public schema.

MySQL has simple (and probably inconvenient) mechanism for server-side shared libraries with C functions. Rudimentary support for triggers was included beginning with MySQL 5.0.2. An external development implemented in perl can be used as stored procedures in Mysql. MySql has more powerful admin tools included in the distribution (mysqladmin allows you to watch processes and queries in-progress), including hot backup, a file corruption recovery tool and couple of others. Command-line tools - you can see database and table structures using describe and show commands. Postgres' commands are less obvious (\d to show a list of tables for instance).

Outcome

In our application triggers are not used, hence it doesn't convince us to migrate from mysql to postgres. Even though mysql doesn't offer the depth in the afore mentioned features but is no way incompetent. Moreover, keeping the nature of our application and code in mind, we can safely assume that Mysql is a good compromise.

vii) Locking and Conurrency support

PostgreSQL has a mechanism called MVCC (MultiVersion Concurrency Control), comparable or superior to best commercial databases. It can do

row-level locking, can lock rows for writing in one session but give these rows unaffected in another session. MVCC is considered better than row-level locking because a reader is never blocked by writer. Instead, PostgreSQL keeps track of all transactions and is able to manage the records without waiting to become available.
MySQL can do table locking for ISAM / MyISAM and HEAP tables, page level locking for BDB tables. InnoDB has full row level locking support.

Outcome

PostgreSQL is a clear winner when it comes to transaction control, and especially taking the number of simultaneous operations being performed on the same table. But InnoDB offers a good compromise by providing complete row level locking.

viii) Large Objects

In Postgres, Large Objects are very special beasties. You need to create them using lo_create function and store the result of the function - OID - in a regular table. Later you can manipulate the LOB using the OID and other functions - lo_read/lo_write, etc. Large object support is broken in Postgres - pg_dump cannot dump LOBs; you need to develop your own backup mechanism. The team is working on implementing large rows; this will replace current LOB support.

In MySQL, text and binary LOBs are just fields in the table. Nothing special - just INSERT, UPDATE, SELECT and DELETE it the way you like. There are some limitations on indexing and applying functions to these fields.

Outcome

Mysql is better than Postgresql in this aspect but sadly large objects aren't used in our VCL code.

ix) Alter Table

Postgres supports ALTER TABLE to some extent. You can ADD COLUMN, RENAME COLUMN and RENAME TABLE.

MySQL has all options in ALTER TABLE - you can ADD column, DROP it, RENAME or CHANGE its type on the fly - very good feature for busy servers, when you don't want to lock the entire database to dump it, change definition and reload it back.

Outcome

This is a very essential feature which mysql fully supports and provides. Provides great flexibility and avoids unnecessary locking.

x) National Language Support

Postgres compiled with --enable-locale does some jobs based on its locale settings, and can change locale settings per client (not per database), which is a bit more flexible. Compiled --with-mb (Multibyte support) Postgres can translate on-the-fly between many predefined character sets.
MySQL does some tasks based on its locale settings, but not many. *.Both databases now support Unicode characters; MySQL supports both UCS-2 and UTF-8 encoding, and PostgreSQL supports only the latter.*

Outcome

Again a compromise to be made in regards to this feature if we choose to mysql, but it is not much of a concern as our tables use the default latin1 charset.

xi) Features

Both MySQL and postgreSQL provide and array of built in functions that provide functions ranging from string manipulation functions to date time functions. There are lot of common data types supported by both postgreSQL and MySQL , they also contain certain data types that are native to only there system , for example the interval datatype.

Postgres has certain features that are unique to it , and mysql either doesn't have these features built-in or they are being planned for future release.One such feature is the use of subselect anywhere in a query. Both postgres and mysql are extremely easy to use , recently the GUI for mysql named query browser was released which makes the task of *writing Stored Procedure , Functions , writing query , executing backups , administering the performance of the system , modifying performance variables* etc… a interactive and simplified operation. Such a GUI tool is available for postgres

as well.

Administration is extremely simple in both postgres and mysql , this is completely different from database systems like oracle .As mentioned in the last para with the help GUI tools the task becomes even simpler.

Outcome

I feel the features provided by MYSQL are good enough for most of the application especially VCL. Hence this point of features doesn't strongly encourage a migration from mysql to postgres.
In terms of administration both are equally good , but mysql has a slight edge in terms of good experience and replication integration[13].

xii) Support

MySQL is much more widely used, so many more applications support MySQL, and there is a larger community ready to assist you with problems, as well as more books and resources on MySQL. MySQL AB, the commercial company guiding MySQL, and who employ most of the developers, offer various levels of support contracts.

Of course, PostgreSQL has active mailing lists, and there are commercial companies offering support as well, so you are not likely to go too far wrong with either.

Outcome

Mysql definitely proves to be better than post gres in this case .

xiii) Security

This feature being a very important one , we shall analyze it considering the factors like the existing security features , shortcomings , vulnerabilities ,

attacks , prevention applicable to both mysql and postgreSQL.

MySQL has exceptionally good fine-grained access control. You can GRANT and REVOKE whatever rights you want, based on user name, table name and client host name.

Mysql's simplicity is one of its major strength , extremely useful features of Microsoft SQL Server is the ability to execute queries on remote database servers using openrowset using the following syntax

PostgreSQL has similar features, but a little less fine-grained. For example, if user can connect to a database, user can CREATE TABLE, thus running Denial-of-Service. On the other hand PostgreSQL can limit logins based on different criteria - network segment, indent string, etc.

Select * from openrowset(… ; MySQL host , root , password ; 'Select * from mysql.user'...)

This command allows user to execute a query on a remote server running possibly a different DBMS – in middle of an SQL server query , this could be used as a port scanner to scan the network that the server currently is in. This illustrates one of the strengths of Mysql as openrowset statement doesn't exists.

SQL injection is a technique that exploits a security vulnerability occurring in the database layer of an application. The vulnerability is present when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and thereby unexpectedly executed. It is in fact an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another (*definition courtesy Wikipedia*).

SQL Injection attacks happens only in interactive web applications where the user is asked to enter his/her input at runtime. The attacker may provide a carefully structured malicious query which would result in different database request than the intended one [1] .
Sql injection attacks happens at the application layer as where dynamic sql queries are generated , this can be avoided by checking the query semantics at run time. We have to pay attention to SQL injection attacks that happens at the database layer.

Since we aim on replacing all queries in the php code of VCL with calls to *stored procedures* it would be good to focus on trying to defend attacks directed towards stored procedure in the database layer.

The stored procedure provides an extra layer of abstraction at the database layer as described in Section 1 of the report.

Technique for detecting SQLIA (SQL injection attacks) have been described in *"Preventing SQL injection attacks in Stored Procedure-" (Source 1 in References).* A method described in that paper shows how to prevent the malicious statements from accessing the database.

i) *Sample Code to demonstrate SQL injection in Stored Procedure.[1]*

```
Create procedure employee. RetrieveProfile ( in
name varchar(50), in Passwd varchar(50) )
Sql security invoker
as
begin
declare  @SQL varchar(200);

set @SQL=‘select PROFILE from EMPLOYEE
where ‘;
if len(@Name) > 0 AND len(@Passwd) > 0

Begin

 select @SQL=@SQL+‘NAME=“‘+@Name+“‘ and
‘;
select
@SQL=@SQL+‘PASSWD=“‘+@Passwd+““;

end
else
begin

select @SQL=@SQL+‘NAME=“Guest“‘;
end
exec(@SQL)
end
```

The above procedure presents a possible scenario when SQLIA can take place with a stored procedure.

The user is expected to provide a user name and password, if he doesn't then he is granted guest access.

If the malicious attacker enter the value of user name as follows enters

@name = y ‘ or ‘f‘ = ‘f

@pass = r ‘ or ‘r’ = ‘r

Then the query evaluates to

Select profile from employee where @name=’y’ or ‘f’=’f’ and @passwd = ‘r’ or ‘s’=’s’ ;

This condition would always succeed and hence will result in the attacker seeing all the data present in the table employee which he is not supposed to see.

SQL Injection can be reduced but not completely avoided in mysql and php by using ***mysql_real_escape_string***. Use of stored procedures give us a great deal of flexibility of enforcing some amount of insulation against SQL injection, this is one of the forte of using stored procedures in mysql. Specific permissions are required before a user can create a stored program, and, similarly, specific permissions are needed in order to execute a program. A solution for avoiding such attacks is by using *static analysis* and *runtime validation. This method is described in [1].*

What sets the stored program security model apart from that of other database objects and from other programming languages is that stored programs may execute with the permissions of the user who created the stored program, rather than those of the user who is executing the stored program. This model allows users to execute operations via a stored program that they would not be privileged to execute using straight SQL.

We can also create stored programs that execute with the privileges of the calling user, rather than those of the user who created the program. This mode of security is sometimes called invoker rights security, and it offers other advantages beyond those of definer rights.

Since our application makes use of stored procedure it adds a layer of insulation against SQL injection (all the stored procedures should use security level of invoker so that no malicious user will be able to access the tables .We can code our stored procedures in such a way that tables are completely locked to the users from unauthorized access.

So now to the big question Postgres or Mysql ? Mysql seems to be meeting the requirement of our application (VCL) ,it may not provide data integrity in the same scale or quality as that of postgres, but still does a decent job. Moreover stability is very essential, mysql is much more stable than postgres which suffers random disconnects ,which may not be a good sign for a busy system like VCL.

And mysql is faster in execution of queries that are read oriented , and moreover mysql is very versatile as it is supported on number of platforms. When it comes to the issue of migrating to a different database system ,especially for application of this magnitude a lot of thought has to be given so as justify whether such a change is warranted.

After having weighed all the pros and and cons , I would affirm that MySQL should continue to operate as our back end.

VI. WHETHER IT MAKES SENSE TO DISTRIBUTE THE DATABASE SERVICES IN A SINGLE VCL SYSTEM.

A. Introduction

Highly parallel database systems have displaced conventional mainframe computers for handling large databases and transaction processing tasks. Before two decade or so the future and realization of parallel database system seemed very bleak, this was primarily due to the fact that most of the research was directed towards hardware (that researchers vouched on for the realization of parallel database system. But none of these devices were able to deliver the goods.

Another problem which the researchers faced was the increasing at rate much slower than that of the processor, which increased by several factors. Hence the researchers faced the problem of I/O Bottleneck unless , some way to negotiate the speed difference between the processor and the disk was found out.

The arrival of relational database model changed the entire face of parallel database system. The queries written in relational model were inherently cable of being executed in small chunks simultaneously at different locations ,each location possessing own copy of part of the data , i.e. the each location holds certain number of rows of each table in the schema.

Once the data has been split across different heterogeneous systems, the task of distributed data processing takes over. This task is by no means simple as the parallel system is composed of several thousands of heterogeneous systems, and every system which is a part of the distributed system will undergo changes at random intervals, this would depend on the load at that particular point of time. The new systems which are currently a part of the distributed system have to interact with legacy system although the legacy system do not have to process the distributed data.

B. *The Proposition*

I feel it does make sense to distribute the database services , as it makes the system more efficient , fault tolerant , can accommodate many users , support users from various domains (other than NC state Students) , reduce response time , keep the number of users per blade server to an optimal number , avoid response slowdowns , will be more robust i.e. crashing of one quadrant will not deny any user access to resources

The request will automatically be guided on to the next quadrant , hence the there will never be a case where the user is kept waiting or where the user is forced to retry after some time .
The figures 1.1. and 1.2 show the general nature of a clustered system as well as a distributed system. The structure of each will be explained in the next section.

C. *Figure*

The figure 1.1 shows the basic structure of a clustered system. Here we are under the assumption that the facilities provided by VCL (making a reservation , requesting statistical data , ending a reservation etc…) are available to students and professors outside NC State University . Here I have assumed the system to be open to UNCB , WCU , NCSU and UNCW.

The four colleges are considered to be four different quadrants. In figure 1.1 the user is assumed to make a reservation via a web application through a root database which aids in *distributing the services.* Each of the four quadrants do not maintain their own local copy of the database. The master database also serves as the root database (i.e. all the information regardless of which quadrant the information is applicable , it will be stored in the root database only.

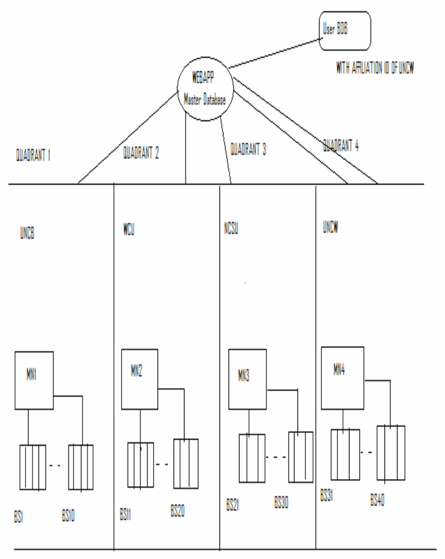


Figure 1.1 – Clustered System

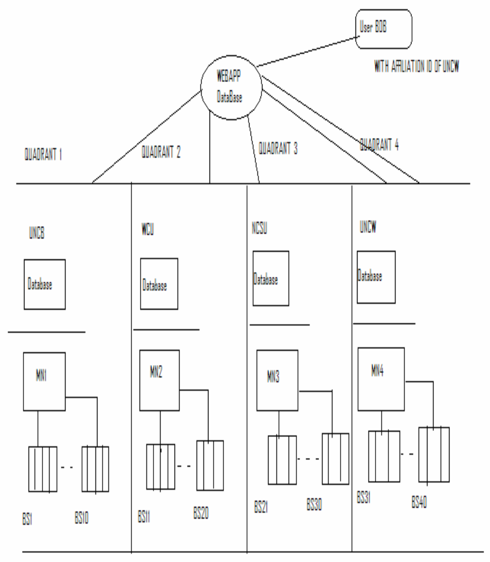


Figure 1.2 – Distributed System

This doesn't seem to make sense as to why should we interleave data from different sources in one place even though we use the same schema everywhere. This results in additional complication in terms of maintaining regular back ups , crash recovery , maintaining several secondary copies of the up to date data in several slave nodes etc…

Moreover we also observe in figure 1.1 that there are only management nodes and Blade servers within each quadrant. When a request arrives say from user Bob , who belongs to UNCB , the data for UNCB students is not localized in the database, hence the response will be slower in comparison to a system which maintains separate copies of the database for each quadrant. After the request was successfully redirected to an appropriate node in the UNCB quadrant , the user has made an reservation and he interacts with the allotted node via the master node. We have determined lots of drawbacks in a clustered system like the one shown in figure 1.1.

Now let us consider figure 1.2 which shows the distributed system. Here each of the four quadrants maintains it own local copy of the database which holds all the information pertinent to that quadrant like user information , log tables , preloaded images, etc… the databases in different quadrant though they maintain only local information all of them share the same schema. Here the root database directs information to the appropriate quadrant. The content of the master database will be such that it will be able to determine the ***exact affiliation of the user*** from his input.

Now what if the entire quadrant comes down then the master node should be able to redirect the request to the nearest quadrant (based on criteria like network proximity, network hops etc…). Hence the system proves to be very robust, but this implementation is still in the rudimentary stages. Once we dwell more into it, we may discover additional challenges in it.

Now, if the master node which has the same schema as the all the four databases in the quadrants, then we have to introduce robustness by having some sort of master slave architecture, which would allow the slave to take over in case the master fails , this would require the slave to maintain the exact copy of the master all the time , so that it can continue from where the master left off. This is not the only concern we have also keep the back up turn around time to the minimum i.e. the restoration or the taking over of the slave should be done as soon as possible.

The problem of keeping the local copy in the slave up to date can be taken care of by using the technical described in the section 4 of the report. Master Slave model can be used where the slave holds the exact copy, and can take over in case the master fails. This works as follows, the master maintains a binary log of all queries which modify the data in the database, and this is sent across the network to the slave. The slave executes these set of queries against its local copy of data, hence allows consistency to be maintained between the local copy and the server copy.

We can even use the method described by jung hoo choo and Hector Garcia Molina [11] in their paper on "Synchronization database to improve freshness". The problem is focused on the web data, but this method can be applied to our domain as well.

D. Challenges and propositions

In distributed database system as the one depicted in the part C of this section raises several issues one of them is robustness, which was discussed in the last few passages, the next is the performance issues which is mainly Parallelism and Concurrency control and quick response time to users who access the system and fast redirection to the closest quadrant (depends on the network proximity using metrics like hop count) in case the quadrant to which the user belongs has crashed. This would require the news of the crash to be communicated to the master database as soon as possible so that it can

take an action as soon as possible. We are using a shared nothing architecture for our reference model of VCL [3]. Each memory and disk is owned by some processor that acts as a server for that data. Mass storage in such an architecture is distributed among the processors by connecting one or more disks. The Teradata, Tandem, and nCUBE machines typify this design. Shared-nothing architectures minimize interference by minimizing resource sharing.
They also exploit commodity processors and memory without needing an incredibly powerful interconnection network. As Figure 4 suggests, the other architectures move large quantities of data through the interconnection network. The shared-nothing design moves only questions and answers through the network. Raw memory accesses and raw disk accesses are performed locally in a processor, and only the filtered (reduced) data is passed to the client program. This allows a more scaleable design by minimizing traffic on the interconnection network. Another advantage of using the Shared nothing approach is the fact that we can use hundreds and thousands of processors which do not interfere with each other. [3]

Apart from considering the approach we are using (i.e. Shared nothing approach) . We also have to decide how the inter connection between the various quadrants and the master node is going to take place. Several options including tree structured communication network, three level duplexed network and are mentioned in [3] .Our architecture doesn't require every nodes to communicate with each other.

Performance of the distributed system is a composite term which depends on several factors like response time, throughput and speed up over varying range of loads. We must also take into account the impact of communication overhead, initiation costs on the performance. We must also be able to determine how the system concurrency control is ensured and what effect will concurrency control have on the performance of the entire system. We have to analyze how scaling the system affects the system performance and how degree of partitioning the database improves the performance.

Most of commercial available systems allow for inter transaction parallelism (allowing more than one transaction to execute simultaneously) and some provide both inter and intra transaction parallelism which improves the response time of individual transaction[2], we want our VCL distributed system to support both inter and intra transaction level parallelism.

We will transaction using both the master process (which runs on the master node) and several cohort processor (runs in one of the nodes in the quadrant) these two in tandem are responsible for the successful execution of the transaction , but the cohort processor in a particular quadrant is responsible for the execution of the task. The process running in the master node will also be responsible for controlling the concurrency. Various locking protocols are discussed in [2].

In order to increase the intra transaction parallelism we can partition relation existing in a single quadrant by placing the partitions across different nodes. There are several ways to partition the relation of a single database system (within a particular quadrants) namely 1-Way partitioning and 8 way partitioning [2]. The former uses no partitioning and the entire relation may be kept in one single node. The latter has the relation partitioned across eight nodes. Hence the performance in VCL system can be increased using such partitioning schemes which will lead to intra level parallelism [2].

E. Conclusion

We can conclude certain things safely from the discussion of distributed system and distributing the services

i) Performance scaling takes place linearly as the size of the system increases.

ii) Performance of the system scales depending on the degree of partitioning.

VII CONCLUSION AND FUTURE WORK

The VCL as already mentioned is going to undergo a major change in terms of number of users, increase in load and hence additional complexity, unforeseen challenges, performance degradation etc… are a direct consequence of the former. I hope the changes suggested in this report are robust enough to handle all the current problems faced by VCL system.

In the future, the potential for a good research problem in the field of distributing the database services in the VCL system is very high. The solution to that problem will hold the key about the success of VCL system in the future. I hope this will lead to a robust and efficient VCL system which is ready to face the challenges of the future and meet all the requirements of the users.

The process of allowing non-NCSU students to make a reservation and use applications of their choice is already under way by opening VCL to ***Wake tech community college*** , the system is open for 8 courses currently , with each course having around 20 – 25 students.

With Intel ready to donate one thousand Blade Servers the signs are good for a healthy future of VCL. With the presence of thousand blade servers the VCL can support a maximum of twenty five thousand users simultaneously. Hence it would be safe to vouch that VCL is growing far and beyond just the NCSU realm and has the potential in the future to reach massive number of users.

VIII REFERENCES

[1] Ke Wei,M. Muthuprasanna,Suraj Kothari "*Preventing SQL Injection Attacks in Stored Procedure*" pp. 1-8.

[2] Michael J.Carey, Miron Livny "*Parallelism and Concurrency control Performance in Distributed Database System*" pp. 122–125, 127-128.

[3] David J.DeWitt,Jim Gray, "*Parallel Database Systems,The future of high performance Database Processing.*

[4] Donald kosssman, "State of art in distributed Query processing".

[5] Stephen M.Thomas "Using Automated fix generation to mitigate SQL Injection Vulnerabilities" pp .

[6] Ivan Zoratti MySQL AB , "MySQL Security best practice.

[7] Matthias Jarke, Jurgen Koch,"Query Optimization in database system". pp 111-114

[8] Philip A.Bernstein , Nathan Goodman , "Concurrency control in distributed database system".

[9] Sheldon Finkelstein , "Common expression analysis in Database Applications" pp 235 - 244

[10] UV Ramana , TV Prabhakar , "Some Experiments with the performance of LAMP Architecture".

[11] Junghoo Cho, Hector Garcia Molina "Synchronizing a database to improve freshness

[12] Title "Open Source Databases , Part III : Choosing a database" Author Reuven M.Lerner June 2007,Linux Journal,Volume 2007 Issue 158 Available: http://ieeexplore.ieee.org/ie15/4123728/4123729/04123759.pdf?tp=&arnumber=4123759&isnumber=4123729

[13] Title "Open Source Databases , Part II : Choosing a database" Author Reuven M.Lerner June 2007,Linux Journal,Volume 2007 Issue 157 Available: http://delivery.acm.org/10.1145/1250000/1243947/9618.html?key1=1243947&key2=8608116911&coll=GUIDE&dl=GUIDE&CFID=43771622&CFTOKEN=88303657

[14] Mysql 5.1 official Documentation - http://dev.mysql.com/doc/refman/5.1/en/

[15] InnoDB official Site - http://www.innodb.com/

[16] Jeremy Zawodny Official Mysql site - http://jeremy.zawodny.com/mysql/

[17] Michael Kofler , Apress : A definitive guide to Mysql – Second Edition

Yeshwanth Kumaraswamy is from Raleigh , North Carolina .He received his bachelors in computer science (2003 – 2007) from Visvesvaraya technological University , Bangalore INDIA. He enrolled in for a masters program at the NC state

University in fall of 2007.He will receive his masters by spring of 2009.