Clowder: A Software for Managing Cluster Of High Performance Computers

by

© *Samson Ugwuodo*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of *Science*

Scientific Computing Program

Memorial University of Newfoundland

*A*pril 2017

St. John's Newfoundland

Abstract

Clowder is a system designed to help researchers in the Faculty of Engineering and Science to manage and control clusters of high performance computers. Some of the features that are missing in this system is the ability to reserve a computer for a certain period of time, a function that allows users to add more computers and manage their properties , and also a user interface where users can have interactive access to the system. Therefore, there is a need to upgrade and complete this system by developing software that provides the missing functionality. A web interface addresses the issue of user accessibility. Having a dynamic database system provides the functionality of keeping record of user activities, storing computer information and managing all reservations. These design approaches are achieved using SQL models, HTML templates and the Go programming language. Instead of performing several command line prompt statements, which is the current method in the use of Clowder system, the software provides flexible user access to the cluster of computers, a dynamic system management in a single software system as a research tool in the Faculty of Engineering and Science.

Acknowledgements

I will like to express my humble thanks to my supervisor Dr. Jonathan Anderson for his overwhelming support, guidance, professional advise as a professor and his contribution through out this project.

I want to express my sincere appreciation to the School of Graduate Studies, the Department of Scientific Computing and Computer Engineering for their academical and financial support.

I would also like to thank the head of Department Dr Ronald Haynes , and Gail kenny for their subsequent advice and support through out my in class academic works and other wise.

Finally, I am grateful to my family and God almighty for the love and encouragement that i have been blessed with through out this journey.

Contents

List of Tables

List of Figures

1 Introduction

The use of high performance computers for research in the Faculty of Engineering and Scientific has increase, because of increase in research tasks, such as testing new operating systems. As researchers demand large data storage and robust computer architectures, these computer systems and their properties also need to be expanded to accommodate this large tasks. As we expand the systems access and usage becomes more complicated to manage. Therefore we are faced with the issue of system management, system accessibility, and storage. These issues for example, are the inability to provide proper record of each computer system and their usage, the inability to provide flexible users access to the computers, and also the problem of monitoring the number of computers that are reserved by users. As a result of these challenges it is highly necessary to develop software that solves these problems for Faculty of Engineering and Science.

Clowder is a system designed to provide Preboot Execution Environment (PXE) for testing new operating systems through a Network File System (NFS) sever on the high performance computers. Previous work has been started by researchers in Engineering department to complete these system for several years, but yet there was more improvement yet to be made in order to complete it, as it lacks some necessary modern functionality and features, such as database system, interactive user interface and restrictions. As an example, it takes extra effort and manual command line statements to access a computer in the cluster during research. Another reason for improvement is to provide flexible and dynamic user interface rather than current one (SSH). Therefore the current state of Clowder is not flexible and convenient enough. So the main goal of this project is to develop software that addresses the issue of user accessibility, system management and automatic control protocol. Addressing all these issues will improve the performance of Clowder in speed , access and robustness, and therefore making it a completed software tool.

We have accomplished this goal by using a design approach that provides a web interface, a databases system and automatic command protocols. The database system is designed to store and keep record of the computer systems and user activities. The web interface enable users to remotely log on to several machine from different locations simultaneously via a web page, and also provide user with the system inventory and other user activities. This software provide the ability to adding new computer system to the cluster, and to modify their properties individually. Also it provides the ability to make reservations: to allow users to reserve a computer for a certain period of time, and able to end reservations as well. A function to allow user search the inventory with some specifications according to their demands.

As this software serves as a tool to manage the cluster of computers and user activities, it is important to know that reservations made, computer details, network interface cards and disks are stored as data. So all the computer system installed in the cluster with their names, vendors, memory size, architecture, and micro architecture is stored in the database. And same is applicable to the disks, network interface cards and any other devices that could be part of the cluster. Data is represented as variables in their various data types in the database scheme. All these information stored in the database tables servers as input data for the program and out put for the inventory. This database scheme allows user to add or update new machines installed in the cluster, and therefore provide dynamic access to the machines.

The web interface serves as a platform for users to interact with the system and overview other users activities via a web designated web page. This interface has also replaced the command line prompts which was the previous user interface for Clowder system. This choice provides flexible access and control to the system, by allowing users to log on to the system and make request of the inventory at any time through the web server.

2 Design and Specifications

2.1 Specifications

The objective of this software is to a dynamic user interface and a functional database system. With regard to that, the specification is made up of necessary features and functionality that appear on the user interface. The following are the major specification of this software:

- Data Inventory
- Searching Inventory(SI)
- Make Reservations
- Cancel Reservations
- Add Data
- Update Data

Registered users have access to the system with unique user name, so the user interface provides a log in form on the web page where user can submit their details to be stored in the database. The user interface is able to list the inventory of data stored in the database in different categories. This is one of the main features of the user interface because it presents the content of the database. As there are data inventory on the user interface, it is necessary to have a function that allow users to search information from the inventory. Another requirement of this software is the ability to make reservations: users are able to choose a machine from the inventory and reserve it for a fixed date and time. This requirement on the high level is meant to allow user to a specific machine for running a particular task on that machine without interruption. When a reservation made is expired, the machine that was reserved becomes free for other users. As new machines are added to the cluster, the Add-data functionality allows users to add details of those machines to the database and update the inventory. Likewise, when the user want to change a certain properties of the machines such as, memory size or disk, the Update-data functionality helps to perform this action.

2.2 Design

The main idea of the software is to solve the issue of user interface, and system management. So we have designed this software with the approach to provide functional solution to the specifications. The design structure comprises the various segments and element of the user interface (front end) combined with the database (back end) and the control protocols. Below is a figure that describe the main structure of the design in terms of operation:

We designed a web interface that provides flexible and dynamic access to the system with interactive functionality. This web page contains inputs and output features. The reason for choosing a web page instead of other options is to provides online dynamic and simultaneous access to the system rather than performing manual command prompts (example: checking the usage of machine, properties of machine, and status of machine on Unix interface). Another reason is to have a functional real time user interface that present the activities of the Clowder. The web server is responsible for serving the web page all the resources requested by the users, and it serves as a channel between the user and the database. We designed an operational (event recording and system referencing) database that address the issue of system management. The database stores data from the web page and also retrieves data on the web page via the server. As the user log on to the their account on the web page the server will pass this information to the database which is controlled by the main program (command protocol). The command protocols is responsible for handling the user interface activities and executing the database queries.

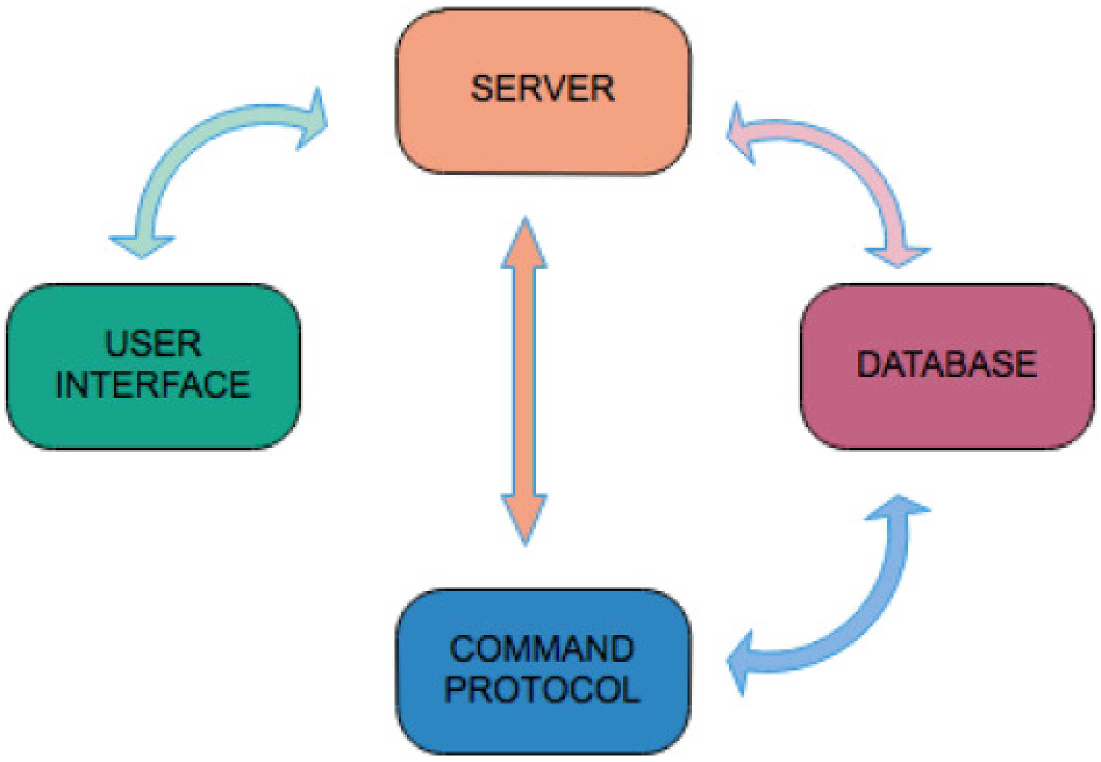

Figure 2.1: Design

2.2.1 User Interface

The web interface represents the user interface (UI) where users get access to the system. It is part of the design that represents the front end of the software.The web interface elements controls the basic functionality for data input and out put. It represent each feature as mentioned in the specification. This web interface structure is designed using HTML template as the front end and Go programming language as the back end. Below figure discribes the web interface avtivity; The

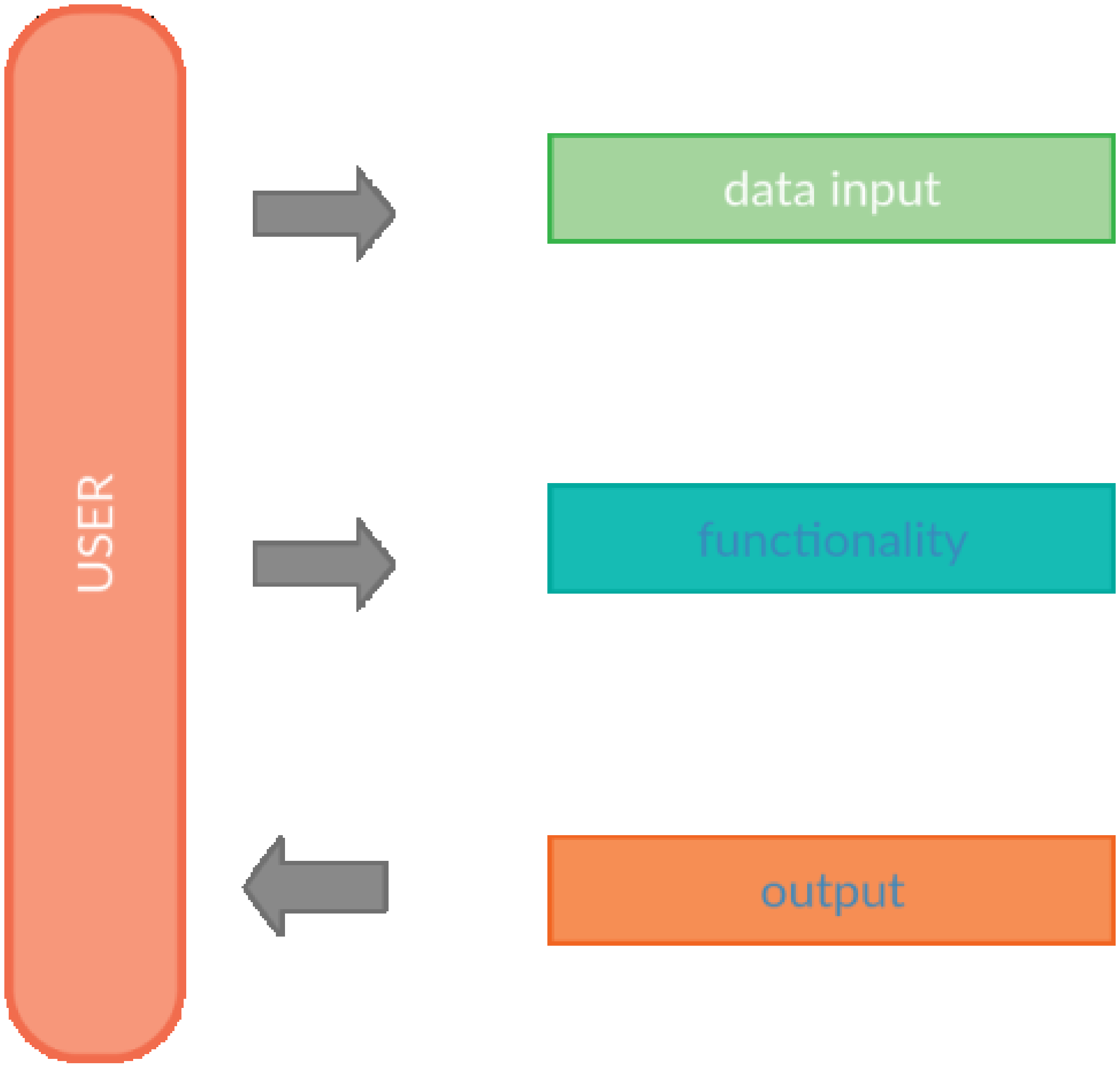

Figure 2.2: User Interface Description

web interface contains all the necessary elements required to have dynamic and flexible access to the system. These elements includes:

- Forms
- Input controls

Forms

Forms are one the elements that provides the ability to input data to the database.The data is submitted through a HTML post request [4]. The post forms are used for submitting data such as machines and reservations details. The form is designed with elements which includes; id and name attributes for identifying data, text fields for collecting data, select menus for selecting data options and submit button for submitting the data. When data is submitted with HTML form, the Go HTTP framework in the web server call the Go function handler to process and parse this data to the database. After the data (example: a database query) is processed, the Go program generates an HTML page (such as machines inventory), which the server returns to the web interface. In this system design, the forms are used for creating new machines, disks, NICs and making reservation.

Input Controls

The input control in this web interface provides the necessary components that supports the functionality of the software. There is array of HTML input control, but we have chosen a few that satisfy our specifications. These input controls includes text field, buttons, date fields, drop down list and list box. They are used for both input and output functions like listing the inventory, inserting texts, to update data, to send queries, viewing selection options and searching inventory. These controls are designed with HTML tags and HTML template.

2.2.2 Server

The server is responsible for parsing all request from the web interface to the database and command protocol. When request is made on the web interface (for instance available machine on database), the server calls the specific function of command protocol to handle this request by providing the necessary resources. Likewise the server interchange communication between the user interface and the database system for executions and requests.

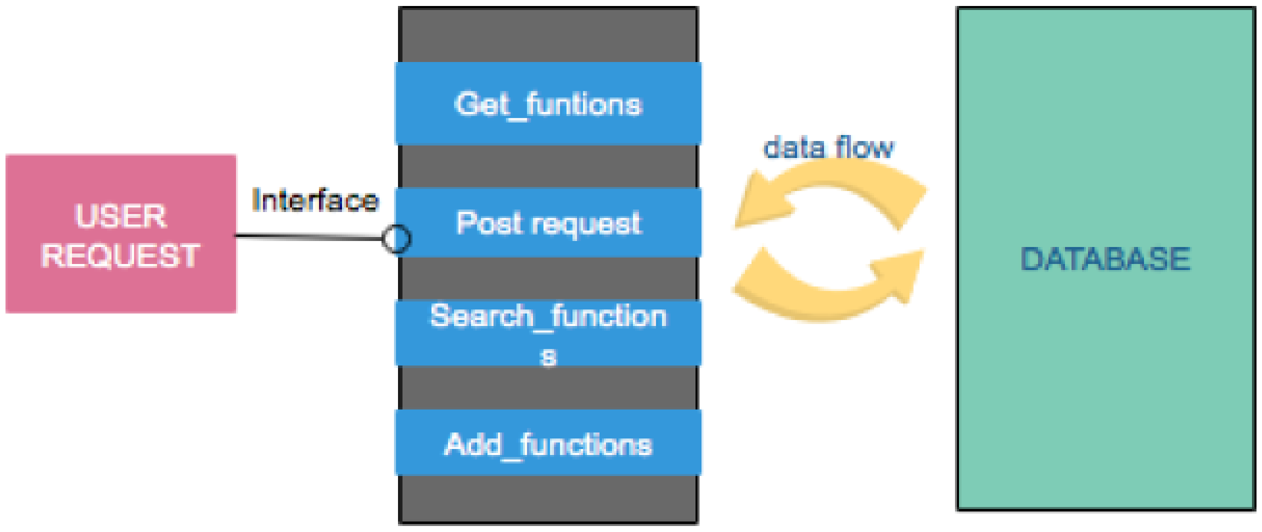

Figure 2.3: Server Activity Description

2.2.3 Database

The database is another major component of this system, because it is provides storage and resource for the system. The data includes reservation records, machine details, disks, and NICs information. The database is designed with SQLite model using Go programming language as the back end tech. SQlite was used here because its a fast open source SQL engine. We created a functional database system that suites the specification of the software. The database scheme contains tables of machines, user record, reservation, NICs (network interface cards) and disks. One of the function of the software is the ability to make reservations. Below show a typical model of the database design.

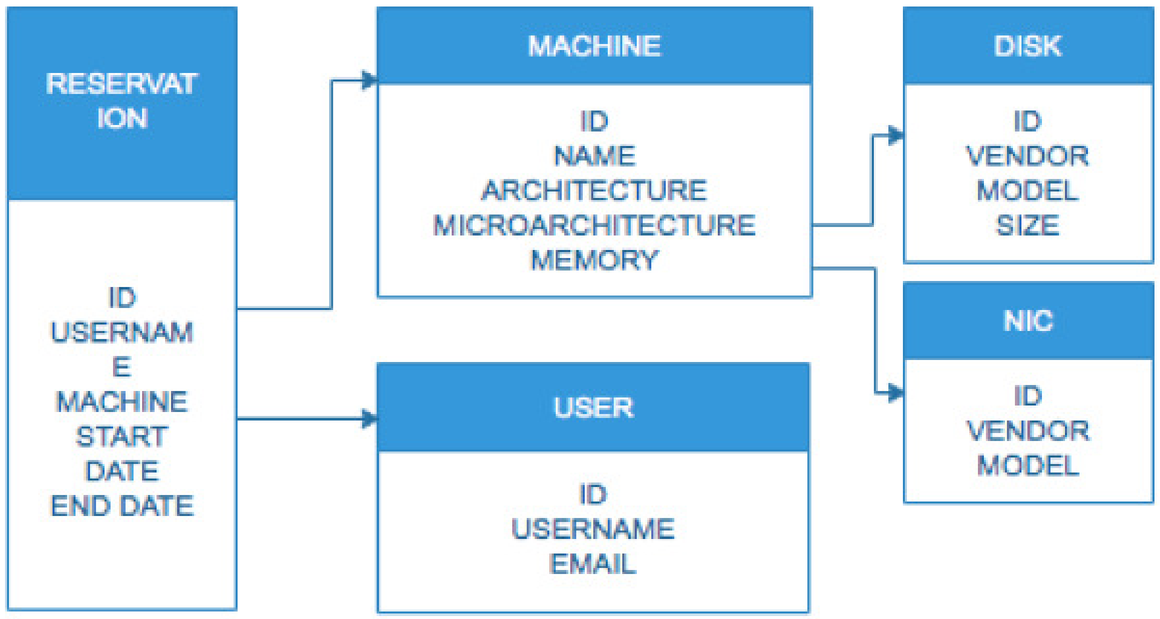

Figure 2.4: Database design

2.2.4 Data flow and protocols

The data flows from the user interface via the server to the database. We used Go HTTP handler to process data recieved from the HTML form to be stored in the database. The client server initiate each method (function) when the HTTP handler sends a request. For example, when users search for list of information such as machines details, the request is sent through the server as a query, and the specific Go fuction is called to fetch the data from the disegnated database. This process is the protocol that control the sequence of data flow in the system. It controls data input, handling and output. This protocol improves data circulation in the system and provides automatic command rather than manual query routin perfomed on commad line.

3 Implementation

3.1 Review

The software implementation showcase the functionality of all the requirements as described previously. This functionality include the ability to input data and retrieve data through the system. Proper measures has been be considered to obtained maximum result as required. This measure includes the choice of tools used for the developement and the design approach. It was implemented to demonstrate the realization of the proposed specifications especialy features such as making reservations in the system.

3.2 Implementation tools

The user interface is implemented on the internet web browser using apeche local server port as web address. The web browser serves as the environment for testing the frontend (user interface) of the software, and while the HTML template and Go library is used for the development. The database was developed using the SQLite model schema which is coded with Go language. This platform provides robust query funtionality for creating several data schema. The HTTP server provides connections between the frontend and backend implementation. It server as a channel for parsing command protocol through user interface and database. We used local port to represent web address for this implemeting. These tools provided all the necessary component to address the requirements for this software.

3.3 Program Structure

The Go programming language has a fundamental developemtn structure that is categorized into packages. The packages contains a group of program file with dependencies that links them together. For this software design, we have used two main package to actualize the develpment goal. These packages inludes:

- Database package
- HTTP package

The database package contains all the Go files with database related method (struct) and functions. Each Go class or struct of hardware represented in the system depends on this package for communicationg with other classes and database resources. The HTTP package also contain some Go file relates to the user interface (for example the funtion Handler) and HTML files. the HTML files contians several templates for the web interface content and elements. These file depends on the HTTP package for serving the web contents and post requests. Below is a table describing the list of header files in each packages.

No	Package	Files
1	Database (pkg)	Machine.go
		Reservation.go
		Disk.go
		User.go
		nic.go
2	HTTP (pkg)	Handler.go
		Server.go
		Templates(.html)

Table 3.1: Program package

3.4 List of Funtions

The functions contains group of statements that perform the tasks specified in the software requirement. It contains the algorithms that process different queries and schema actions. The method fields contains the entities of the hardware represented in the system, and this feilds are used as argumet in the various functions.

No	Struct	Functions
1	machine()	initMachines()
		CreateMachine()
		UpdateMachine()
		GetMachines()
		GetAvailableMachine()
2	reservation()	initReservation
		CreateReservation()
		GetReservation()
		FilterReservation()
		SortReservation()
3	disks()	intiDisks
		CreateDisks()
4	nics()	initNICs
		CreateNICs

Table 3.2: List of Funtions

3.5 Description of Functions

Adding Machines to the system

Adding machines and other devices to the system is one of the software requirements. On the user interface, the software provides a text feild and forms where users can enter details of machines to be saved as data in the database. The SQLite Init funtion create a new database schema with defined table feilds where data is stored. When the user submit a AddMachine form, the InitMachine() create the database schema where the machine details is stored. It uses type (Machine) feilds as argument for creating the schema feilds. Below is the code listing for creating database:

Listing 3.1: Creating Database for machine

```
func initMachines(tx *sql.Tx) error {
        _, err := tx.Exec(`
        CREATE TABLE Machines(
                ...........
        );`)
        return err
}
```

The CreateMachine function is reponsible for parsing the details of machine to the database schema. It uses the SQLite insert query to parse a give data using the arguments to the database. When the user submit a form (AddMachine), the Go HTTP handler process this form by converting the plan text to a formdata accroding to their data types defined in the schema, and then call the CreateMachine to insert this data to the database. Below is the code list:

Listing 3.2: Adding machines details

```
func (d DB) CreateMachine(name string, arch string,
        microarch string, cores int, memoryGB int) error {
        _, err := d.sql.Exec(`
                        INSERT INTO Machines(name, arch, microarch,
                         cores, memory)
                        VALUES (
                                //feild values
                        )`,
                name, arch, microarch, cores, memoryGB)
        return err
}
```

Making a Reservation

The reservation process is similar to creating a machine but here it required the UserID and MachineID as froeign keys in creating the database schema. The UserID is used for selecting the user making the reservation, and MachineID for selecting the machine to be reserved. On the web interface (reservation page), the form has a drop down list where users can select their username and the machine they want to reserve. When users open the reservation page to create new revervation, the HTML form use the post request to get list of machines and username on the dropdown list.

Listing 3.3: Storing Reservation details

```
_, err := d.sql.Exec(`
        INSERT INTO Reservations(machine,user,start,end,
        pxepath,nfsroot)
        VALUES (
                (SELECT id from Machines where name=?),
                (SELECT id from Users where username=?),
                //other fields
        )`,
```

To make a reservation, users will select their username and a machine from the drop down list which is loaded by GetMachine and GetUser function. The selection query gets the UserId and the MachineId of the selected username and machine as foriegn key to the reservation database shema. Also, start and end time/date of the reservation is entered on th text feild , and this plan text is formated to match the database using a time package in Go library. After submiting the form, the Go HTTP handler handles the Insert query by taking all the arguement values and saving them to database.

Checking Available Machines

The user can search the inventory for specific imformation. One of the example is the ability to search for available machines in the system. Available machines are those machine that are either not reserved at the moment or that are free for a certain period. This feature helps users to get list of machine free to be reserved. In this process of checking available machine, the user enters a specific start date and end date in the provided text feild, and submit the query. After submiting the query, the HTTP handler process this request, and the server calls the GetAvailableMachine method (function) to fetch this request from the database. Go HTTP handler uses a Request functon to call the server and ResponseWriter to respond to the html request [2]. The server uses ListernAndServe method to listen to incoming request and to sall the requested method to handle the request. In the GetAvalaibleMachine method, we used a SQLite sub query tech to combine the machine and reservation database table. This combination logic allows the query to scan through both table rows using reservationID and machineID to indentify the machines that are not reserved for that time range the user entered. The logic contains SQL WHERE cluase, OR and AND operator to perform comparisons between reservation dates and time. [1]The AND operator is used to allow multiple conditions in the statement, the OR operator is used to combine condition that are true, and the WHERE clause is used to specify condition while fetching data from database[3]. Below code listing shows the function and query for this operation:

Listing 3.4: Searching available

```
func(d DB) Filter_M_By_Dates(from time.Time, to time.Time) ([]Machine, error){


        rows, err := d.sql.Query(`
                SELECT m.id, name, arch, microarch, cores, memory
                FROM Machines m
                WHERE m.id NOT IN (SELECT r.machine FROM Reservations r
                WHERE (? BETWEEN r.start AND r.end)
                OR (? BETWEEN r.start AND r.end)
                AND ( r.start BETWEEN ? AND ?) OR (r.end BETWEEN ? AND ?)


        );
        `, from, to, from, to, from, to)
```

Updating Machines details

As part of the software requirements, users can change the information stored in the database. This is done on the wen interface using edit box that is attached to every feild with that requirement. Example is updating the information of a particular computer or disks when the hardware is upgraded. In this case, the UpdateMachine method performs this action by using the SQLite update query. This update query opens the database schema of the machine and replaces the existing data with a new one. Below code listing shows the update function:

Listing 3.5: Function for Updating data

```

        _, err := d.sql.Exec(`
                        UPDATE Machines
```

```
                    SET arch = ?, microarch = ?, cores = ?, memory = ?
                    WHERE '+row+' = id

        ',arch, microarch, cores, memory)
```

3.6 Get Funtions

The Get function is a mothod used for fetching list of machines and other devices stored in the database. Each time the user opens the web interface, the HTTP handler send server a request and server calls the GetMachines method to fetch the data from database. This information is listed in the inventory on the web interface through the HTTP ResponseWriter. The Get futction uses the SQLite selection query to scan throught the database and select the required data. Another usage for this Get function is fetching the details of a machine when the user click on a particular machine. It uses a SELECT query and WHERE condition in the logic, to speciy the machine clicked used its unique ID. These mehtods is applicable to other classes (type) such as, disks, nices and reservations.

3.7 Problems Encounted

During the implemetations, we encountered problems such as formating plain text to database data types, and combining two database table for search queries.

4 Evaluation

4.1 Test cases

We have evaluated this software by testing different functionality as specified in the requirement. This was achieved by creating multiple cases that proves the performance of the software. This evalution was performed in real time and directly online. We have inlcuded images and table showing the test cases and results.

Checking available machines

In the software specification, it is designed to have search option on the user interface. One of this search requirement is to check for available machines (unreserved machines). In the below table is list of machines reserved for different date. The test is to enter a search query with a specific date interval and ask the software to provide any machine available withing that date.

No	Machines	July	Aug	Sep	Oct	Nov	Dec
1	Machine A	free	free	free	free	free	free
2	Machine B	free	free	free	free	free	free
3	Machine C	free	free	free	free	free	free
4	Machine D	free	free	free	free	free	free
5	Machine E	free	free	free	free	free	free
6	Machine F	free	free	free	free	free	free

Table 4.1: List of Reservations

No	Date: start — end	Available machines
1	4/07 to 24/07	A C D E F
2	4/07 to 24/07	A C D E F
3	4/07 to 24/07	A C D E F
4	4/07 to 24/07	A C D E F
5	4/07 to 24/07	A C D E F
6	4/07 to 24/07	A C D E F

Table 4.2: Test case and Result

Reserving a machine

Reserving machine is another requirement we have implemented in this software. This is where users choose a machine and create a reservation with definit time and date.

Updating a machine

filtering inventory

5 Future Work

In my future work I intend to implement this software concept in a mobile application platform. Using a schedulling technology and a reservation capability concept, we can develope a software application that can manage a parking lot payment and space reservation. The purpose of this idea is to reduce the time people spent hovering around the parking lot in search of space. The application will allow user to pre-book a space and pay for time online before driving to the parking lot. With that ability, the designated parking lot will only have enough space for people that have pre-booked a space. this idead can be implemented using any programming language that support object oriented programming.

6 Conclusion and Discussion

Bibliography

[1]

[2]

[3]

[4] M. LLC. MS Windows NT kernel description, 1999.