Vladimir Adam
Nick Cross
Ryan McGinley
Cesar Polanco
Karanbir Toor

Detailed Design Specifications for vTimeSeries

Version 1.0
March 7th, 2013

\

Table of Contents

1 vTimeSeries

1.1 Application Overview

1.1.1 Definitions

API - Application Programming Interface
DB - Database
GWT - Google Web Toolkit
MVP - Model-View-Presenter
TSA - Time Series Analysis
VM - Virtual Machine
WS - Web Services

1.1.2 Launching the WebApp

The app will be launched by loading the corresponding web page. This will allow the user to access the website without having to download much client-side code. Most of the application logic will be executed on the server, reducing the necessary complexity or capacity of the user’s computer.

1.1.3 Exiting the WebApp

The user will exit the app by closing the corresponding tab on the web browser. Before exiting user will be able to save their current location in the statistical hierarchy, thereby starting in the same location next they open the app.

1.1.4 Document Overview

The remainder of this paper will document the design of the application. The motivation behind this is to make abundantly clear the architecture of the app as well as the thought process that went into each decision. The document will then go into the detail on the different modules that comprise this system, and will describe the user at each stage. The interactions of the various modules will be described in detail. Finally, the document will then show the different users interacting with the system and trace their steps as they perform various actions.

1.1.5 Model-View-Presenter

The user interface is going to be based on the MVP software pattern. This design paradigm consists of three modules. The model, an interface defining the data to be displayed or otherwise acted upon in the user interface. The view, is just a passive interface that displays the data from the model and sends commands to the presenter. Lastly, the presenter acts on both the model and view. The motivation for this is to separate the View from the Model, and have the

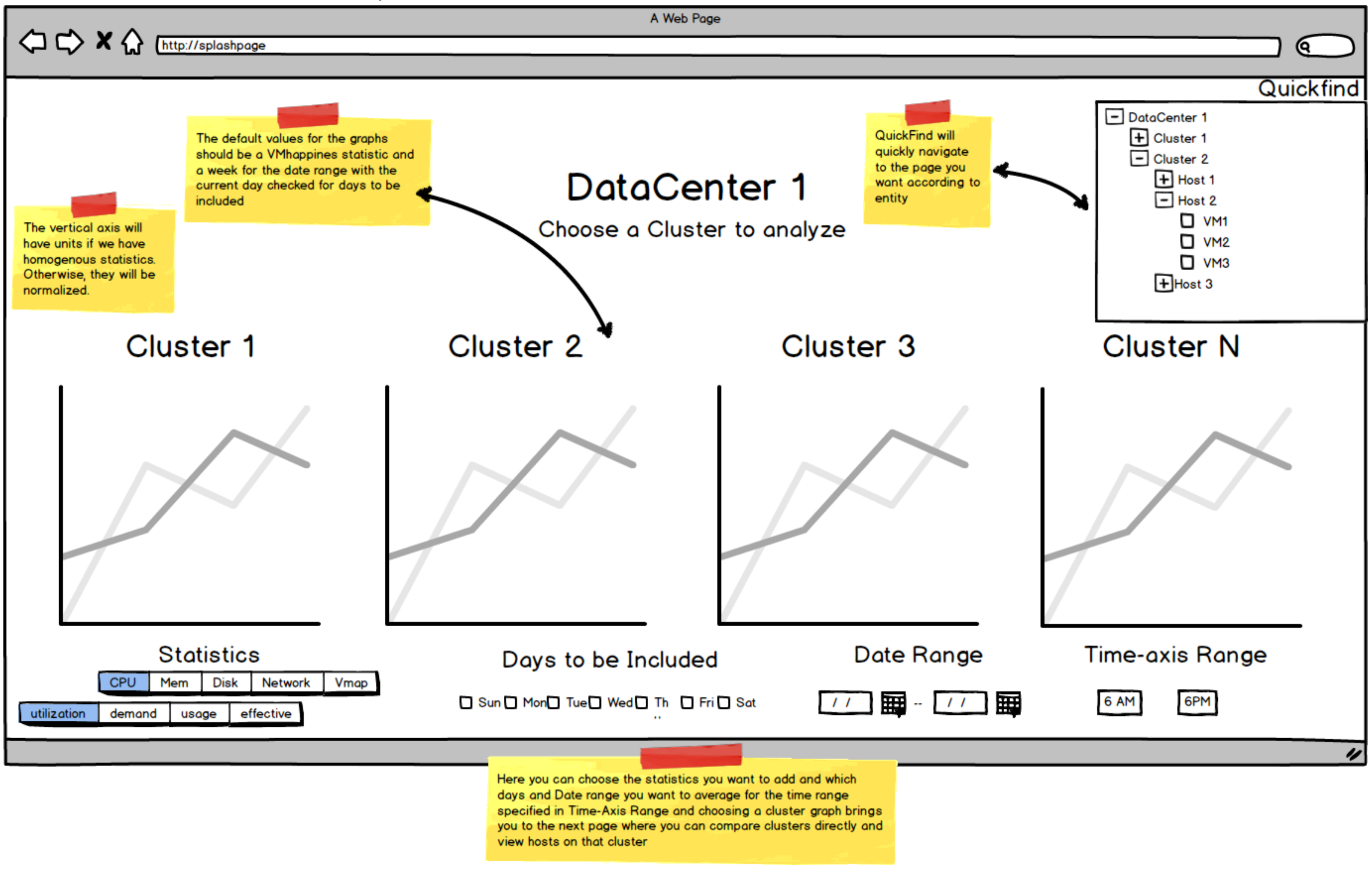
Presenter act as a mediator between the View and the Model. A change in the view will be reflected in the presenter who will notify the model. The model will change accordingly and then notify the presenter which will notify the view. This design

1.2 User Interface

1.2.1 Motivation

The GUI will be designed to provide an easy and intuitive user interface encompassing commonly known combinations of GUI widgets. The GUI will be compact and will allow the user to easily view different configuration of statistics on any number of hosts/cluster that the user desires.

1.2.2 Mockup



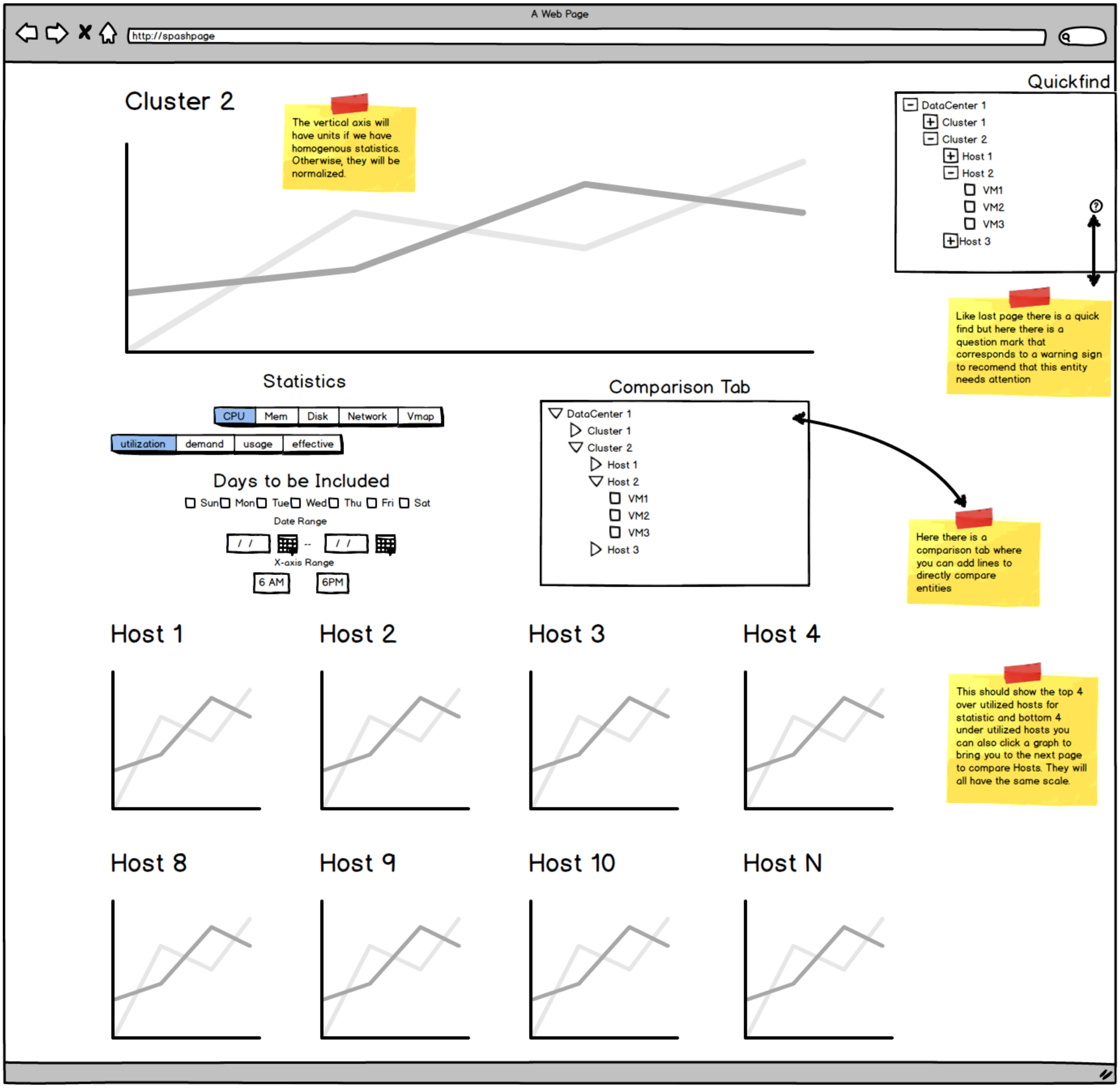
The default values for the graphs should be a VMhappines statistic and a week for the date range with the current day checked for days to be included

The vertical axis will have units if we have homogenous statistics. Otherwise, they will be normalized.

QuickFind will quickly navigate to the page you want according to entity

Here you can choose the statistics you want to add and which days and Date range you want to average for the time range specified in Time-Axis Range and choosing a cluster graph brings you to the next page where you can compare clusters directly and view hosts on that cluster

|
|
V



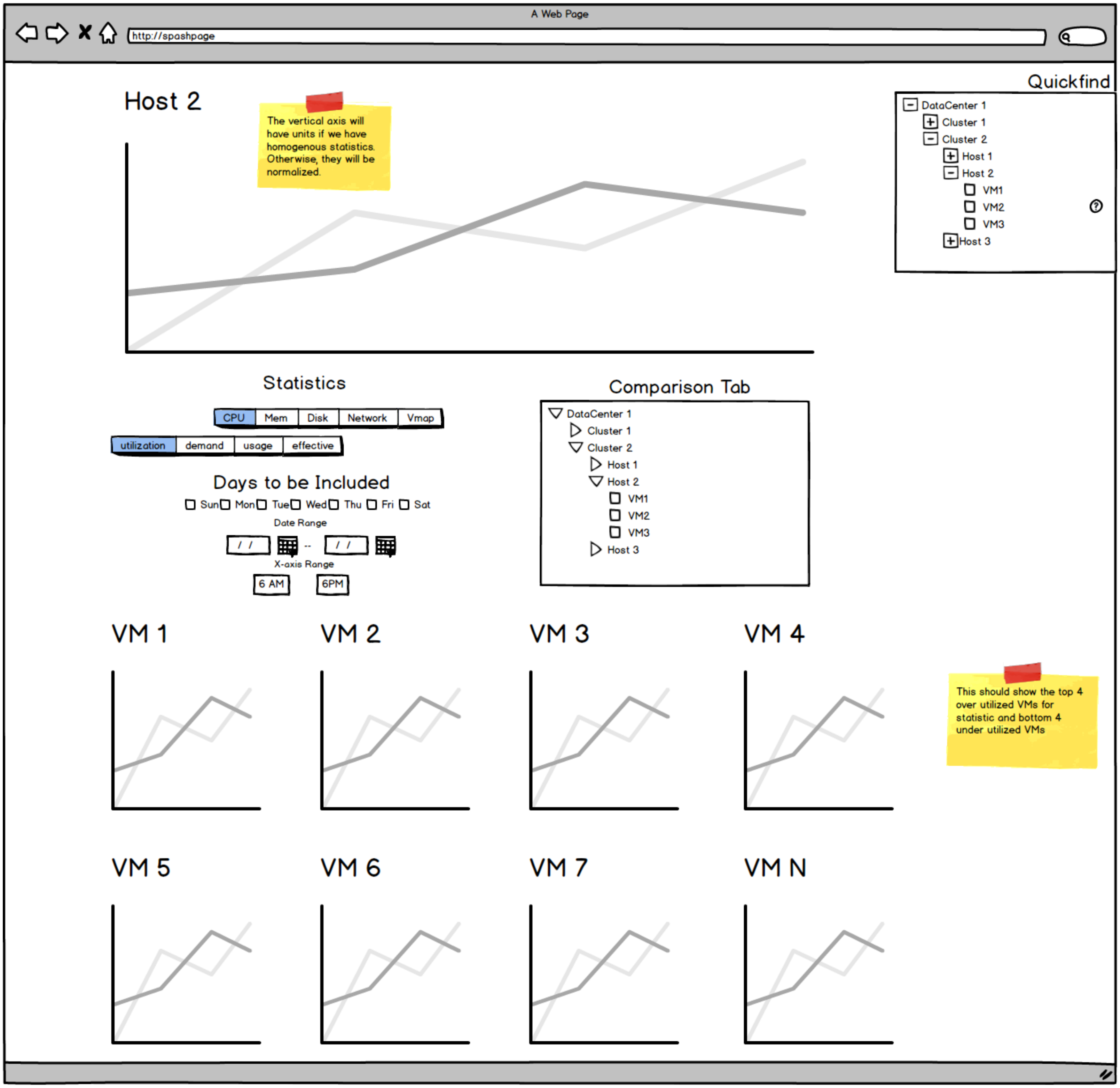
The vertical axis will have units if we have homogenous statistics. Otherwise, they will be normalized.

Like last page there is a quick find but here there is a question mark that corresponds to a warning sign to recomend that this entity needs attention

Here there is a comparison tab where you can add lines to directly compare entities

This should show the top 4 over utilized hosts for statistic and bottom 4 under utilized hosts you can also click a graph to bring you to the next page to compare Hosts. They will all have the same scale.

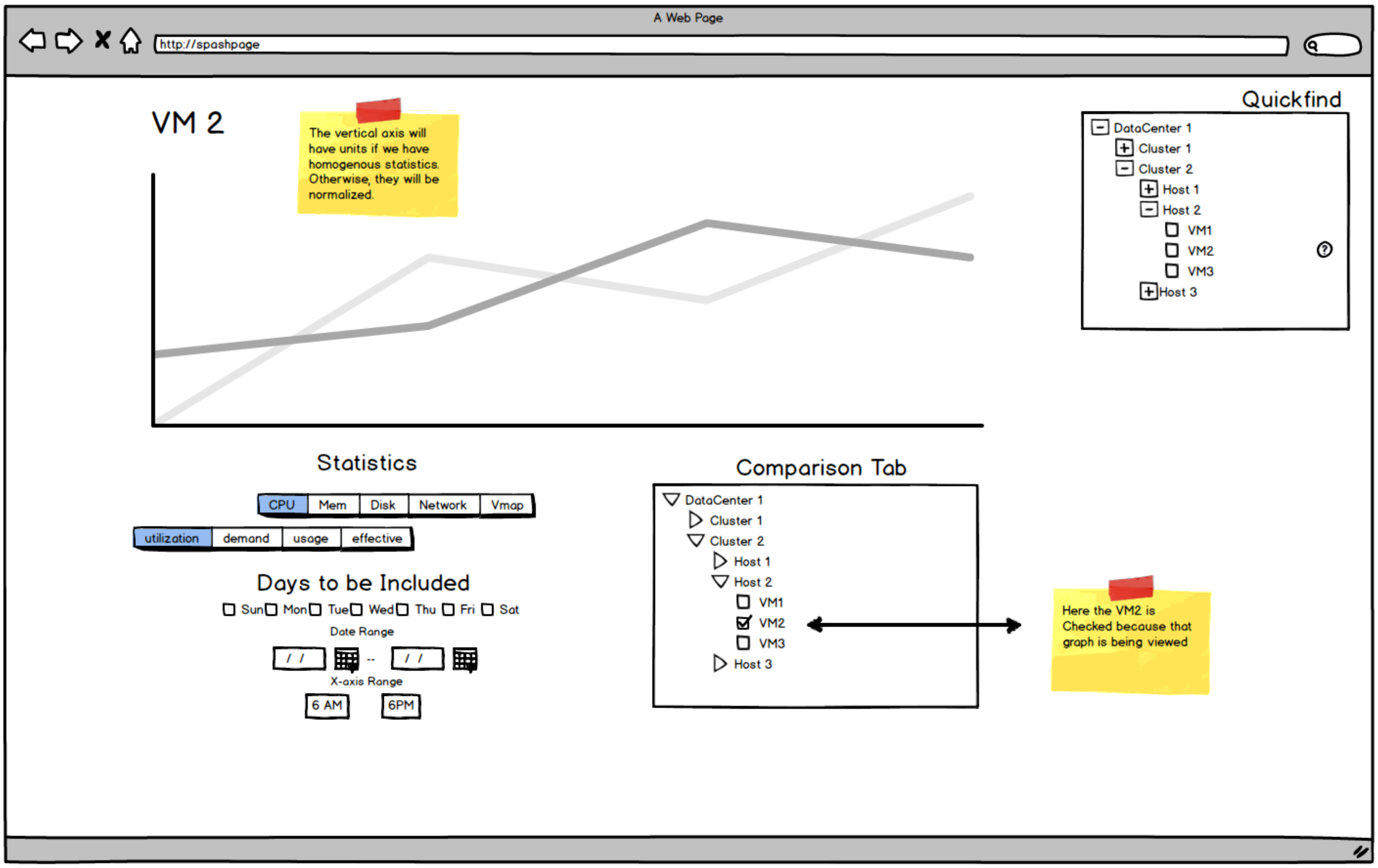
|
|
V



The vertical axis will have units if we have homogenous statistics. Otherwise, they will be normalized.

This should show the top 4 over utilized VMs for statistic and bottom 4 under utilized VMs

|
|
V



The vertical axis will have units if we have homogenous statistics. Otherwise, they will be normalized.

Here the VM2 is Checked because that graph is being viewed

1.3 Basic Functions and Features

1.3.1 Time Series Analysis

We want to identify trends in the data that we can take advantage of. However, real data also comes with noise so we will first need to smooth the data. There are many ways to smooth a time series and the best technique really depends on the type of data.

The simplest smoothing algorithm is a fixed-width sliding average. Where every point on the time series is replaced by the average of some local range. We will do this first to clean up the time series. This technique is a good starting point but leaves many irregularities in the data.

One of the techniques we will experiment with is kernel smoothing. This is like the fixed-width sliding average but the width size is variable and the local range has a Gaussian weight function applied to the average. This technique is good for identifying trends in data, but not good for predicting future values.

A disadvantage of the previously mentioned techniques is they tend to flatten the signal thereby destroying potentially valuable information. The Savitzky–Golay smoothing filter avoids this problem by using polynomial expansions of the data and preserving minima/maxima. This approach will be good for predicting future spikes in the data.

Another trend we will experiment with is exponential smoothing. We believe this technique is especially promising because it is used in performing time-series analysis on financial and market data to identify trends and make predictions. This technique is a moving average with a discrete version of an exponential function as a weight. The major difference in this technique is it uses all previous data to compute the smooth curve, whereas the other techniques mentioned operate in a localized area around each datum in the smooth curve. A potential pitfall for this technique is some data can have convergence issues and give erratic behavior.

After identifying a trend for a particular statistic from a particular VM we can store that trend as a parameter to influence future time-series analysis.

1.3.2 Normalization
The graphs will display units when there is only one statistic selected. However, when there is more than one statistic is selected to compare the coordinates will be normalized. For stats like cpu and mem, normalization will be using the total available resources from the total resource pool. However for stats such as network i/o we will just use the max value.

1.3.3 Hovering Over Plot Point On Graph
When the cursor is hovering over a point on the line it should display the entity type of that line, the stat of the line, and the value at that point.

1.3.4 Legend
When there is only one stat and one entity displayed there will be no legend however when another stat or entity is plotted to the graph a legend for the graph will appear. The legend will show how the graphs are color coded by displaying each color with its corresponding entity and stats.

1.3.5 Displaying More Than One Statistic/Entity On A Graph

For the majority of time while the user is using the application, the user will be viewing graphs utilizing time series analysis. The user will have the option to add statistics to the current graph the user is viewing. This can be done by selecting the desired statistic on a list labeled *Statistics* and clicking the specified statistic then that statistic will remain highlighted. Similarly other entities can be added by checking the box of the specified entity in the *Comparison Tab.* Then the legend will be added to accordingly depending on what is chosen. To get remove an entity all the user must do is uncheck the box and similarly to remove a stat all the user must do is click the stat button again to remove the highlighting.

1.3.6 Quickfind

Because of the vast amount of statistics that can be viewed the user will have the option of using a hierarchical view to quickly navigate to the specified entity page. By using the + expand symbol you can expand clusters to view hosts and expand hosts to view vms. There will also be an indicator image next to an entity that is either being overutilized or underutilized and needs attention.

1.3.7 Manually Selecting a Time Range

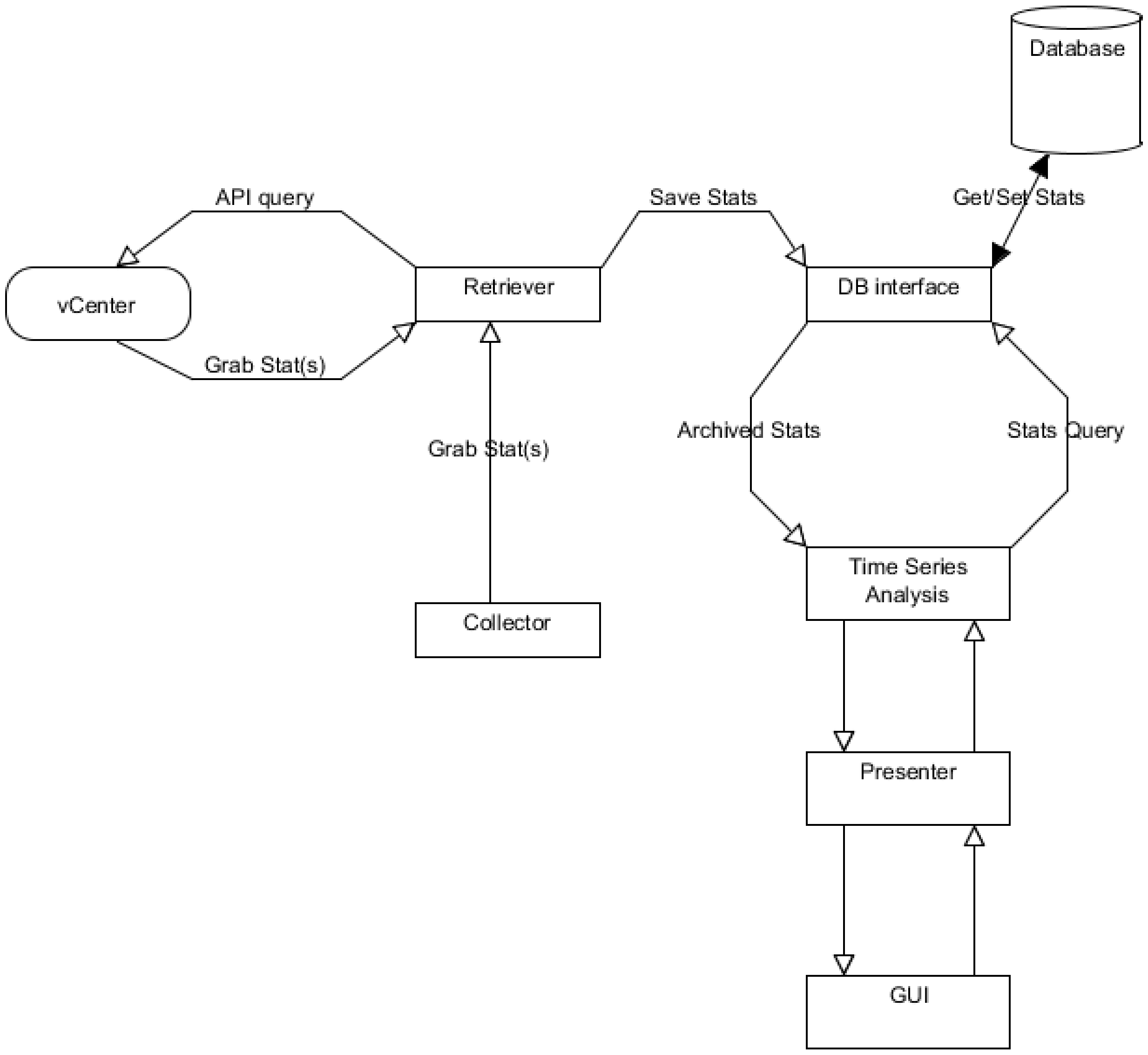
The time range is presented on the X axis of all line graphs and is chosen in the *X-Axis Range* field. This field has a time range for the user to see with a maximum range of 24 hours. The *Days to be Included* field and the *Date Range* field will be the days and dates to be included in averaging for the averaged graph.

1.3.8 Smaller Graphs

For the Cluster page and the Host page there will be a set of 8 smaller graphs showing the top 4 and bottom 4 utilized hosts and VMs respectively. These graphs will need to be continuously updated since new data is always being fed in and the ranking for top 4 and bottom 4 utilized hosts may change with the new data.

2 Modules

2.1 Overview



2.1.1 Application Starting Point

The starting point of our application will be NT_Capstone. This class will implement GWT’s EntryPoint interface, which requires the implementation of the onModuleLoad() function. Here we will initialize our Model and Presenter objects.

2.2 GUI

2.2.1 Purpose

The purpose of the GUI will be to provide a simple yet comprehensive interface where the user can see all the necessary information while not being overwhelmed by clutter of too many unnecessary functionalities. There we will create objects which will constitute our MVP structure. Each view will have its own presenter which. The visual tweaking and finishing touches will be added separately from the code by utilizing Cascading Style Sheets (CSS)

2.2.2 Graphs

We will by using Google Charts Visualization API which lets us display Google Charts alongside other components of GWT. These graphs will act like GWT Widgets and can be integrated into the GWT Layout classes. However, because Google's Visualization API requires an internet connection, the application will require an internet connection in order to display the Graphs and Charts.

2.2.3 Layout

GWT provides the basic Horizontal and Vertical Layout Panels but also contains advanced panels such as FlowPanel and SplitLayoutPanel. However, we are going to use combinations of HorizontalPanels and VerticalPanels to layout all the individual widgets on the screen.

2.2.4 Presenter

All the Views that make up the GUI will be handled in the Presenters for their corresponding views, i.e. each main View should have its own presenter which decouples views from each other and allows for easier event handling.
The presenters will be instantiated in the main application controller, (NTAppController.java). NTAppController.java will contain a reference to the model and to the eventBus which will handle events (some custom events as well) as they are fired by individual presenters. In addition to basic events provided by GWt such as onClick events, we will create our own events for specific situations. For instance, clicking a particular button can fire off AddNewStatistic event.

2.3 Time Series Analysis

The most important module, will use external libraries to process the data collected by the data parser. This data will be averaged here for the time range and will also smooth out the line accordingly.

2.4 Retriever

We have decided that out of the 200 different statistics, we will be using the following for its corresponding hardware layer. This will collect all the available entities and all the available stats for those entities and store them in a DBObject. The DBObject has a array list of entity objects that stores the entity type and maps the stats to the actual values.

VM

● cpu.capacity.usage	mHZ
● cpu.coreUitilization	%
● cpu.demand	mHz
● cpu.usage	mHz
● mem.capacity.usage	KB
● mem.usage	KB
● net.usage	KB/s
● net.throughout.provisioned	KB/s
● disk.capacity	KB
● disk.usage	KB
● vmap.numVMotion	number

Host

● cpu.capacity.usage	mHz
● cpu.coreUtilization	%
● cpu.usage	mHz
● cpu.utilization	mHz
● mem.capacity.usage	KB
● mem.consumed	KB
● mem.usage	KB
● net.usage	KB/s
● net.throughput.provisioned	KB/s

Cluster

● effectivecpu	mHz
● effectivemem	KB
● (cpufairness)	scalar
● (memfairness)	scalar

2.5 Collector

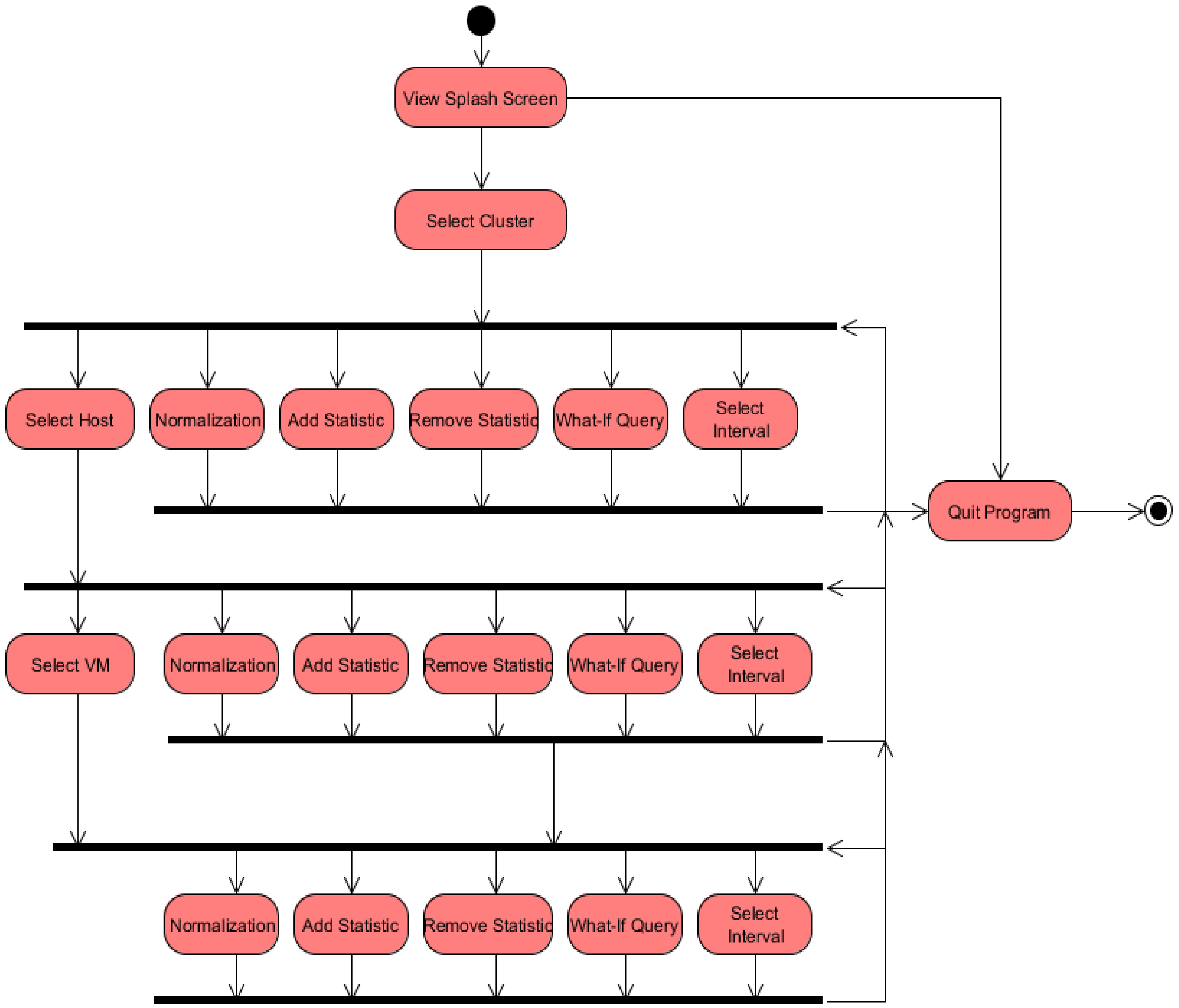
The collector will spawn a new retriever for a specified time interval if none is given it will spawn a new retriever object every 20 seconds which is what VMware considers “real-time stats.”

2.6 Database Interface

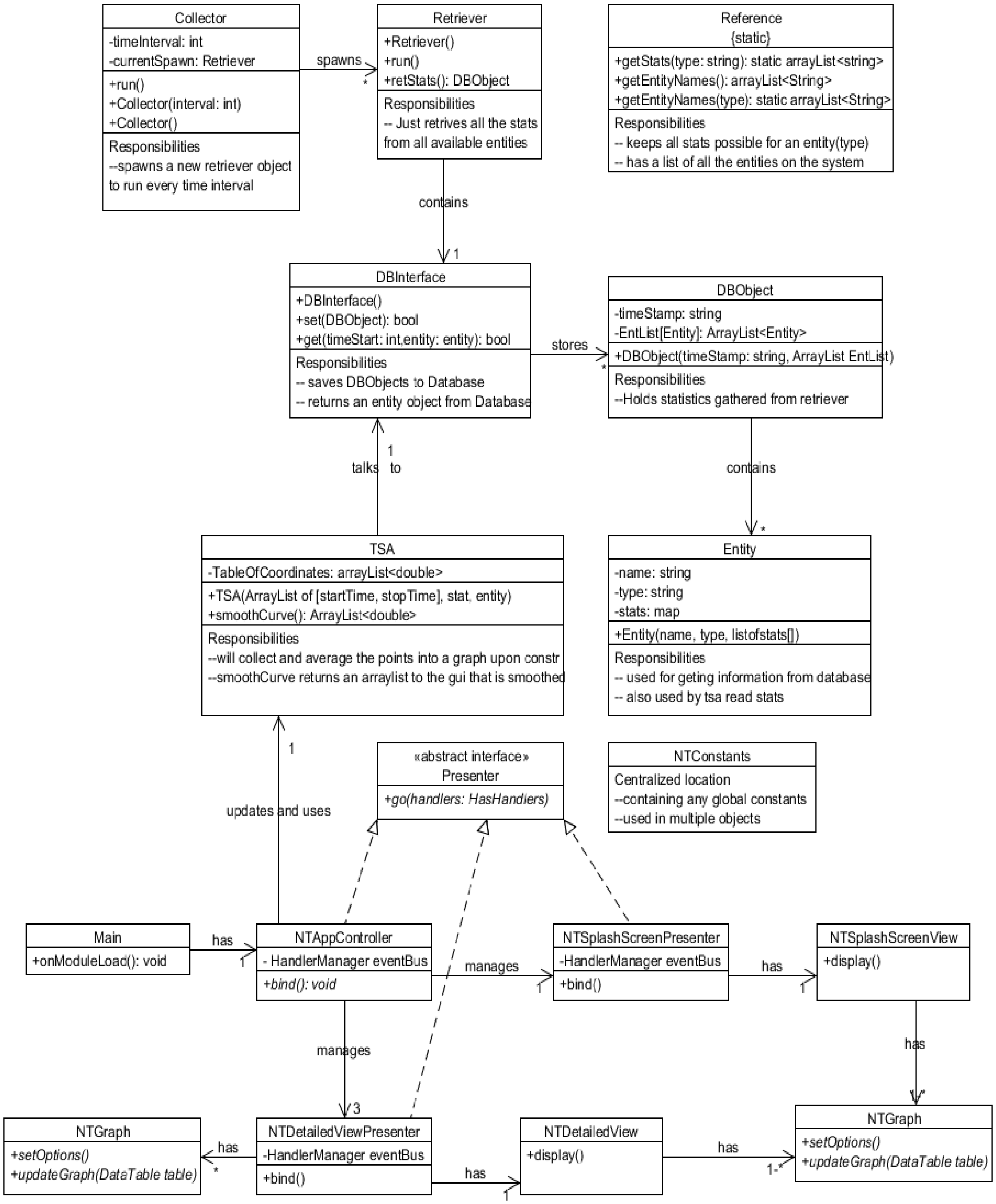
The database will be stored using the set function and will be stored with a key pair that is a time-stamp and entity. Then the values for each of these pair will have a map of stats that will map a stat key to the actual stat value. These will be set using a DBObject that passes a list of entity objects and a timestamp so all entities and all the possible entity's stats are saved to the database.

3 Design

3.1 Activity Diagram



3.2 Class Diagram


Collector
-timeInterval: int
-currentSpawn: Retriever
+run()
+Collector(interval: int)
+Collector()
Responsibilities
--spawns a new retriever object to run every time interval
spawns
*
Retriever
+Retriever()
+run()
+retStats(): DBObject
Responsibilities
-- Just retrives all the stats from all available entities
Reference
{static}
+getStats(type: string): static arrayList<string>
+getEntityNames(): arrayList<String>
+getEntityNames(type): static arrayList<String>
Responsibilities
-- keeps all stats possible for an entity(type)
-- has a list of all the entities on the system
contains
1
DBInterface
+DBInterface()
+set(DBObject): bool
+get(timeStart: int,entity: entity): bool
Responsibilities
-- saves DBObjects to Database
-- returns an entity object from Database
stores
*
DBObject
-timeStamp: string
-EntList[Entity]: ArrayList<Entity>
+DBObject(timeStamp: string, ArrayList EntList)
Responsibilities
--Holds statistics gathered from retriever
1
talks to
contains
*
TSA
-TableOfCoordinates: arrayList<double>
+TSA(ArrayList of [startTime, stopTime], stat, entity)
+smoothCurve(): ArrayList<double>
Responsibilities
--will collect and average the points into a graph upon constr
--smoothCurve returns an arraylist to the gui that is smoothed
Entity
-name: string
-type: string
-stats: map
+Entity(name, type, listofstats[])
Responsibilities
-- used for geting information from database
-- also used by tsa read stats
1
updates and uses
«abstract interface»
Presenter
+go(handlers: HasHandlers)
NTConstants
Centralized location
--containing any global constants
--used in multiple objects
Main
+onModuleLoad(): void
has
1
NTAppController
- HandlerManager eventBus
+bind(): void
manages
1
NTSplashScreenPresenter
-HandlerManager eventBus
+bind()
has
1
NTSplashScreenView
+display()
has
*
manages
3
NTGraph
+setOptions()
+updateGraph(DataTable table)
has
*
NTDetailedViewPresenter
-HandlerManager eventBus
+bind()
has
1
NTDetailedView
+display()
has
1-*
NTGraph
+setOptions()
+updateGraph(DataTable table)

3.3 Advanced Functions and Features

3.3.1 Recommendation

There will be a pop up window that notices on the time series level that 2 VMs have a difference of a certain delta threshold and if the vms reach their maximum and minimum are offset for a certain amount of time a vm migration or affinity recommendation.

3.4 Dialog Box

3.4.1Warnings When Manipulating Statistics or Entities

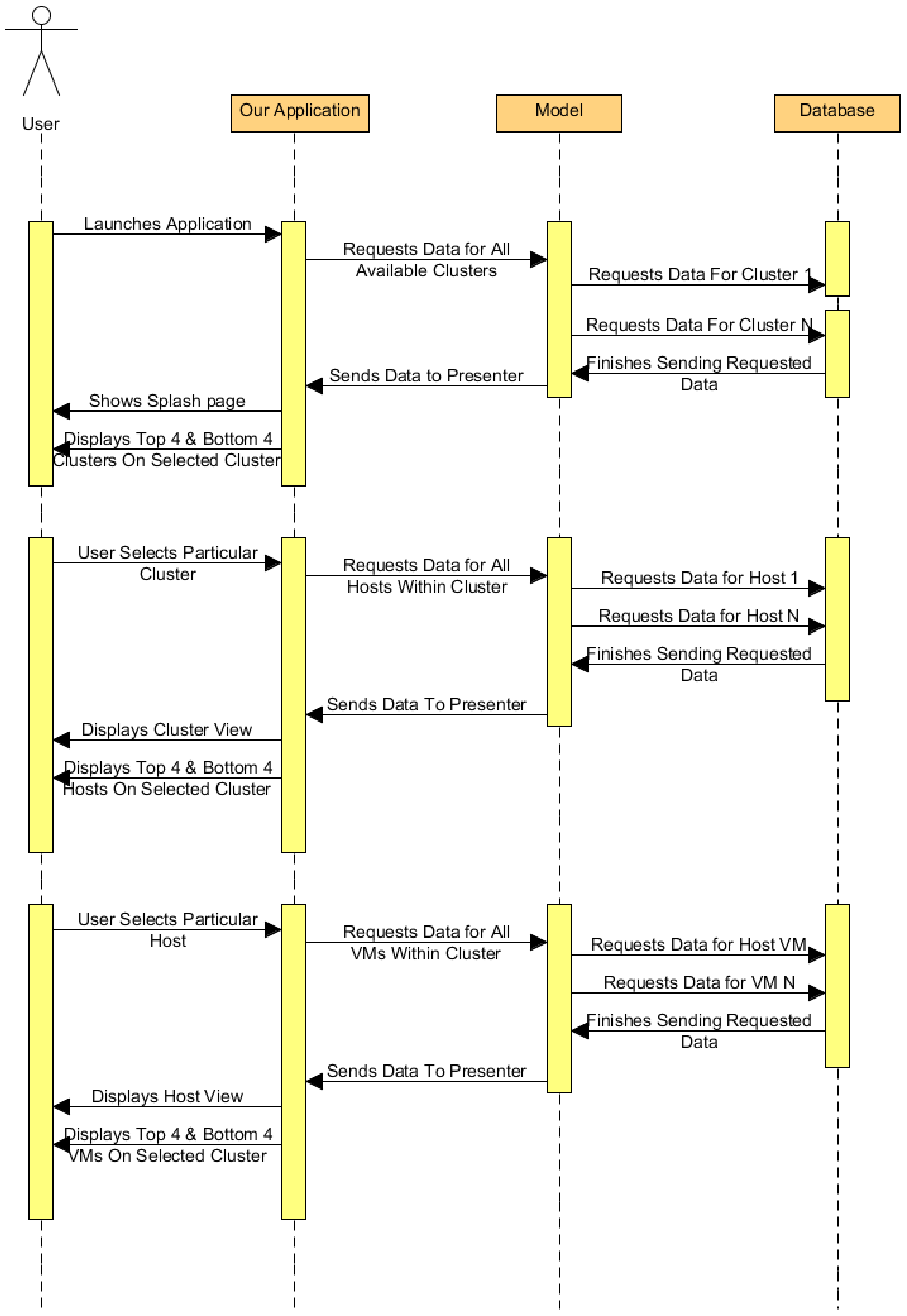
If the user attempts to remove a statistic or entity from being viewed, and it is the last one, we will disallow it/pop up a warning indicating that the user is about to remove the last statistic which will make the graph completely empty.

3.4.2 Warnings When Specifying Date Range

If the user selects a time range that is recording, we will accordingly warn the user that data he will want to view may not be representative of the the actual situation and the user may perceive an incorrect representation of the data. The message will return the date ranges of when there is no data for the specific request.

3.5 Use Case

The following use case shows the high-level overview of interactions between our application, model, and the database containing statistics about individual clusters, hosts, and virtual machines. In it, the user is navigating through the various menus, starting at the data center level, being able to select from multiple clusters. As the user narrows down his choices, he selects a particular cluster, then a particular host from the cluster, and finally, a particular VM from the host.


User
Our Application
Model
Database
Launches Application
Requests Data for All Available Clusters
Requests Data For Cluster 1
Requests Data For Cluster N
Finishes Sending Requested Data
Sends Data to Presenter
Shows Splash page
Displays Top 4 & Bottom 4 Clusters On Selected Cluster
User Selects Particular Cluster
Requests Data for All Hosts Within Cluster
Requests Data for Host 1
Requests Data for Host N
Finishes Sending Requested Data
Sends Data To Presenter
Displays Cluster View
Displays Top 4 & Bottom 4 Hosts On Selected Cluster
User Selects Particular Host
Requests Data for All VMs Within Cluster
Requests Data for Host VM
Requests Data for VM N
Finishes Sending Requested Data
Sends Data To Presenter
Displays Host View
Displays Top 4 & Bottom 4 VMs On Selected Cluster