CPU/IO Usage Summary Project Report

Sponsor: Shane Griffith/Acxiom

301 E Dave Ward Dr.

Conway, AR 72032

University of Arkansas at Little Rock

Capstone 1

Instructors: Elizabeth Pierce and Bruce Bauer

Alexander Sohn

12/8/2021

TABLE OF CONTENTS

Overview

The CPU/IO Usage Summary Project is a simple yet complicated project. To explain, our team at Acxiom has a client that sends us data in which we fix/complete and overall make better for their business purposes. The “solution” that does this runs 24/7 and is constantly updating tables and sending out files for the client to utilize. The client has teams/groups across the world that run campaigns and reports to gather information from the database to see the success of marketing campaigns, individual store locations, and future business decisions. The whole goal for the project is to provide our Production Support team with the proper tool to look and compare CPU and IO usage between the different groups that query from our server, which houses our database. This will include a months’ worth of data to look at. The importance of this capture time is if we miss our SLA (must finish certain processing by certain time), then we can go back and see if a certain group had over average server usage (that caused our processing to be delayed and non-efficient).

Problem

The current problem with how we capture this data is it is manually done through a SQL query that takes 1 to 2 minutes to process, which is then needed to be moved to excel and made into a usable graph. This only allows for one person to view it and does not provide historical data.

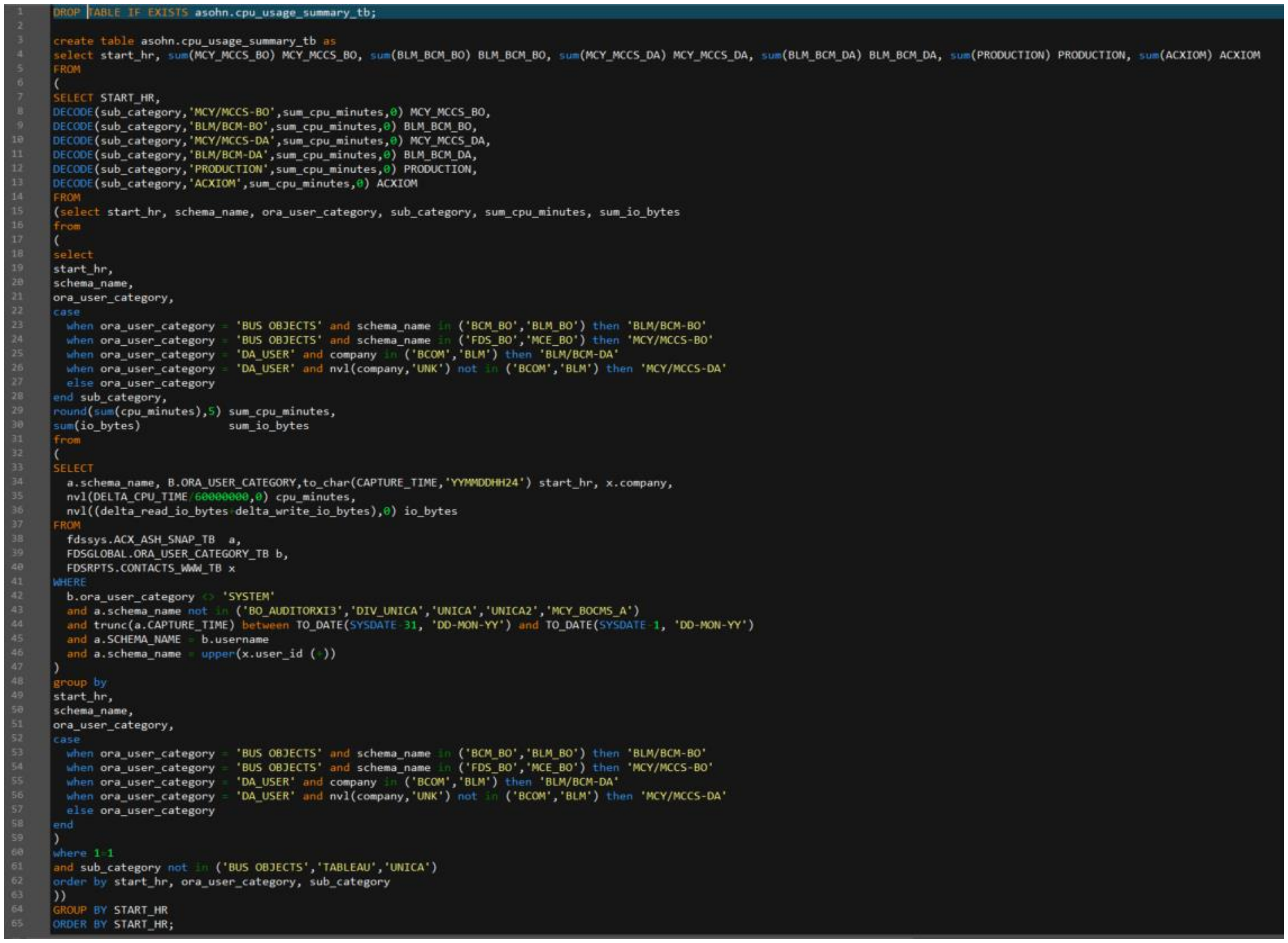
Solution

In all, our solution consists of a webpage that contains two graphs, one for CPU usage and one for IO usage, and a date selection of some sort (drop down or list) that will update the graph. This webpage will be housed on our current inhouse built website, that contains other tools we use daily to help with production support.

Current State

Our solution to this problem is as follows and shown in the current state section. We have built a new SQL query using the old one and techniques from an old graph tool to make the data we capture compatible for our website that we are hosting the data on. First, we put the groups that utilize our server into groups to consolidate them by their type (Direct Access, Business Objects, Acxiom, and Production). Then once the query was built and working, we put it in a create table statement and into a .sql file. Above the create table, we have a delete table statement(important later on).

SQL Code to build table of one months’ worth of CPU Data.

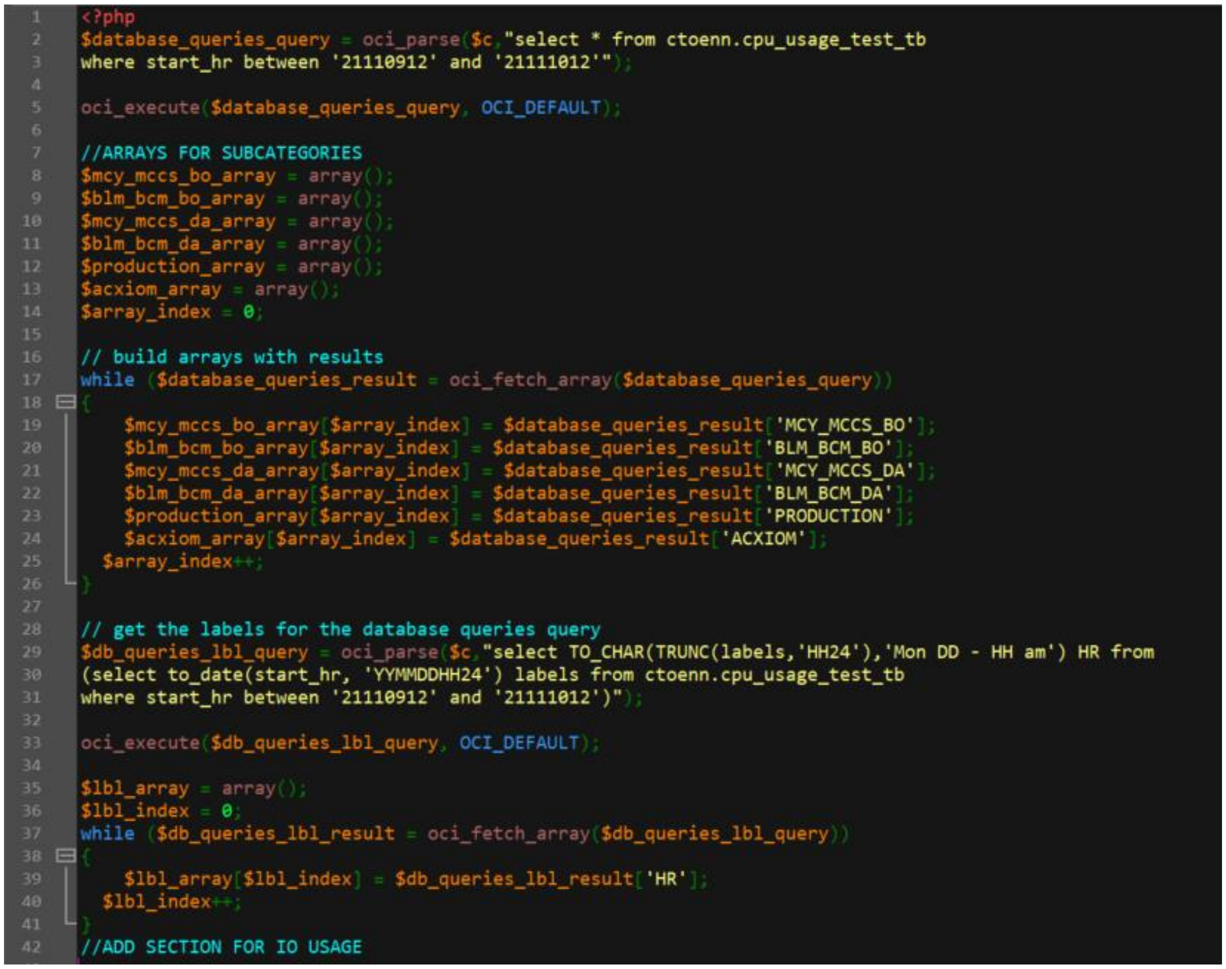


```sql
DROP TABLE IF EXISTS asohn.cpu_usage_summary_tb;

create table asohn.cpu_usage_summary_tb as
select start_hr, sum(MCY_MCCS_BO) MCY_MCCS_BO, sum(BLM_BCM_BO) BLM_BCM_BO, sum(MCY_MCCS_DA) MCY_MCCS_DA, sum(BLM_BCM_DA) BLM_BCM_DA, sum(PRODUCTION) PRODUCTION, sum(ACXIOM) ACXIOM
FROM
(
SELECT START_HR,
DECODE(sub_category,'MCY/MCCS-BO',sum_cpu_minutes,0) MCY_MCCS_BO,
DECODE(sub_category,'BLM/BCM-BO',sum_cpu_minutes,0) BLM_BCM_BO,
DECODE(sub_category,'MCY/MCCS-DA',sum_cpu_minutes,0) MCY_MCCS_DA,
DECODE(sub_category,'BLM/BCM-DA',sum_cpu_minutes,0) BLM_BCM_DA,
DECODE(sub_category,'PRODUCTION',sum_cpu_minutes,0) PRODUCTION,
DECODE(sub_category,'ACXIOM',sum_cpu_minutes,0) ACXIOM
FROM
(select start_hr, schema_name, ora_user_category, sub_category, sum_cpu_minutes, sum_io_bytes
from
(
select
start_hr,
schema_name,
ora_user_category,
case
  when ora_user_category = 'BUS OBJECTS' and schema_name in ('BCM_BO','BLM_BO') then 'BLM/BCM-BO'
  when ora_user_category = 'BUS OBJECTS' and schema_name in ('FDS_BO','MCE_BO') then 'MCY/MCCS-BO'
  when ora_user_category = 'DA_USER' and company in ('BCOM','BLM') then 'BLM/BCM-DA'
  when ora_user_category = 'DA_USER' and nvl(company,'UNK') not in ('BCOM','BLM') then 'MCY/MCCS-DA'
  else ora_user_category
end sub_category,
round(sum(cpu_minutes),5) sum_cpu_minutes,
sum(io_bytes)             sum_io_bytes
from
(
SELECT
  a.schema_name, B.ORA_USER_CATEGORY,to_char(CAPTURE_TIME,'YYMMDDHH24') start_hr, x.company,
  nvl(DELTA_CPU_TIME/60000000,0) cpu_minutes,
  nvl((delta_read_io_bytes+delta_write_io_bytes),0) io_bytes
FROM
  fdssys.ACX_ASH_SNAP_TB  a,
  FDSGLOBAL.ORA_USER_CATEGORY_TB b,
  FDSRPTS.CONTACTS_WWW_TB x
WHERE
  b.ora_user_category <> 'SYSTEM'
  and a.schema_name not in ('BO_AUDITORXI3','DIV_UNICA','UNICA','UNICA2','MCY_BOCMS_A')
  and trunc(a.CAPTURE_TIME) between TO_DATE(SYSDATE-31, 'DD-MON-YY') and TO_DATE(SYSDATE-1, 'DD-MON-YY')
  and a.SCHEMA_NAME = b.username
  and a.schema_name = upper(x.user_id (+))
)
group by
start_hr,
schema_name,
ora_user_category,
case
  when ora_user_category = 'BUS OBJECTS' and schema_name in ('BCM_BO','BLM_BO') then 'BLM/BCM-BO'
  when ora_user_category = 'BUS OBJECTS' and schema_name in ('FDS_BO','MCE_BO') then 'MCY/MCCS-BO'
  when ora_user_category = 'DA_USER' and company in ('BCOM','BLM') then 'BLM/BCM-DA'
  when ora_user_category = 'DA_USER' and nvl(company,'UNK') not in ('BCOM','BLM') then 'MCY/MCCS-DA'
  else ora_user_category
end
)
where 1=1
and sub_category not in ('BUS OBJECTS','TABLEAU','UNICA')
order by start_hr, ora_user_category, sub_category
))
GROUP BY START_HR
ORDER BY START_HR;
```

Current State Cont.

After this SQL was built out, we looked at the existing system of selecting data, creating arrays, building arrays with the data, and then using a query to create labels for the graph and modified to our own. These are the arrays the webpage uses to make the graph with the selected days’ worth of data.

PHP and SQL that will build arrays with data

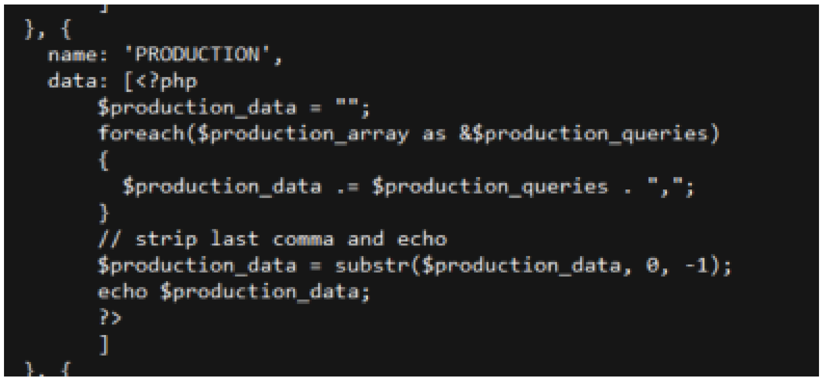


```
<?php
$database_queries_query = oci_parse($c,"select * from ctoenn.cpu_usage_test_tb
where start_hr between '21110912' and '21111012'");

oci_execute($database_queries_query, OCI_DEFAULT);

//ARRAYS FOR SUBCATEGORIES
$mcy_mccs_bo_array = array();
$blm_bcm_bo_array = array();
$mcy_mccs_da_array = array();
$blm_bcm_da_array = array();
$production_array = array();
$acxiom_array = array();
$array_index = 0;

// build arrays with results
while ($database_queries_result = oci_fetch_array($database_queries_query))
{
    $mcy_mccs_bo_array[$array_index] = $database_queries_result['MCY_MCCS_BO'];
    $blm_bcm_bo_array[$array_index] = $database_queries_result['BLM_BCM_BO'];
    $mcy_mccs_da_array[$array_index] = $database_queries_result['MCY_MCCS_DA'];
    $blm_bcm_da_array[$array_index] = $database_queries_result['BLM_BCM_DA'];
    $production_array[$array_index] = $database_queries_result['PRODUCTION'];
    $acxiom_array[$array_index] = $database_queries_result['ACXIOM'];
  $array_index++;
}

// get the labels for the database queries query
$db_queries_lbl_query = oci_parse($c,"select TO_CHAR(TRUNC(labels,'HH24'),'Mon DD - HH am') HR from
(select to_date(start_hr, 'YYMMDDHH24') labels from ctoenn.cpu_usage_test_tb
where start_hr between '21110912' and '21111012')");

oci_execute($db_queries_lbl_query, OCI_DEFAULT);

$lbl_array = array();
$lbl_index = 0;
while ($db_queries_lbl_result = oci_fetch_array($db_queries_lbl_query))
{
    $lbl_array[$lbl_index] = $db_queries_lbl_result['HR'];
  $lbl_index++;
}
//ADD SECTION FOR IO USAGE
```

Current State Cont.

Using JavaScript and PHP, we used the arrays to load the data into the proper sections of the graph on the webpage. We also used JavaScript to create the names for the axis's and title.

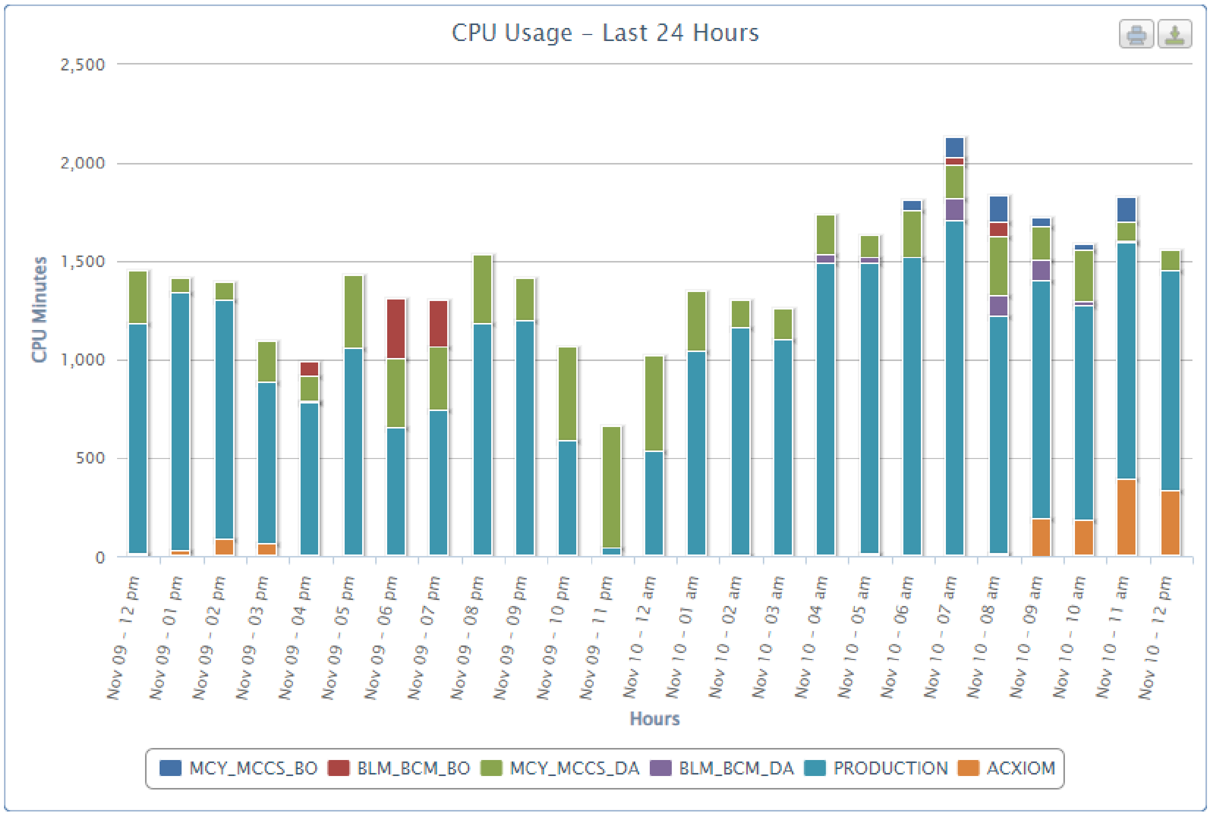
JavaScript and PHP that implements arrays into webpage



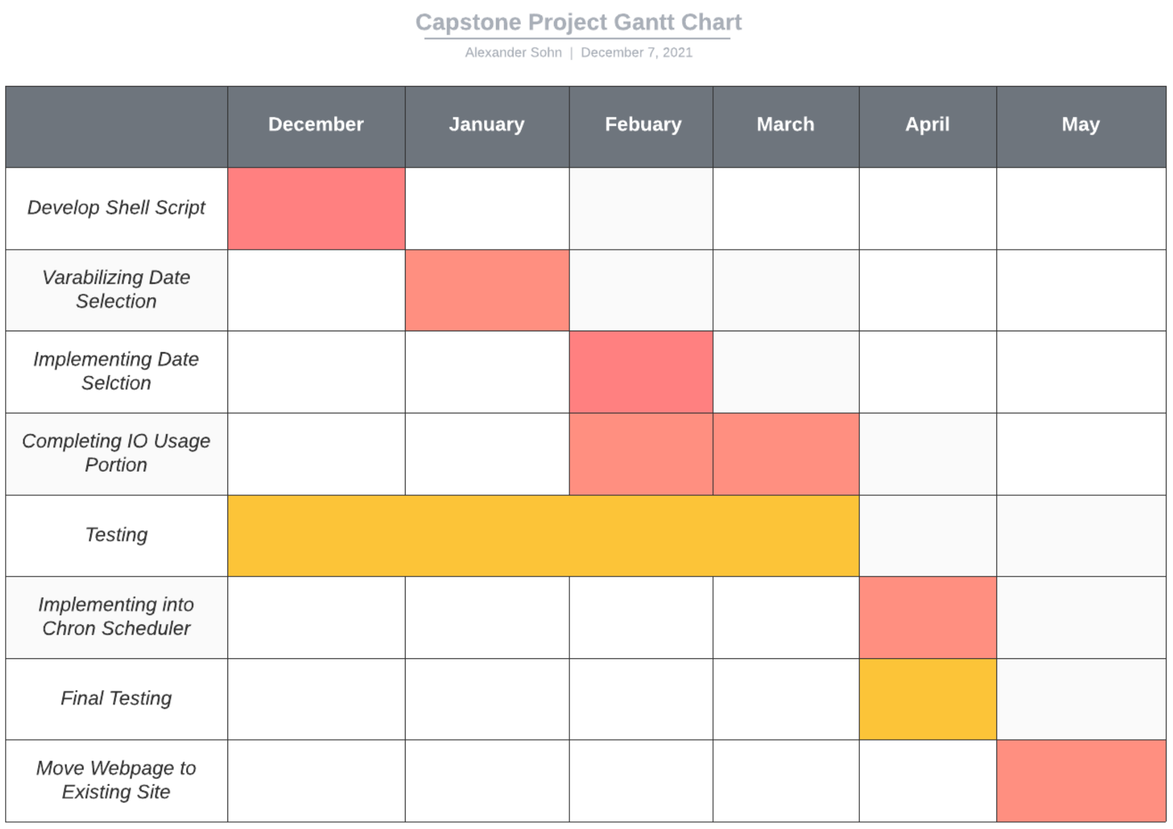
```
}, {
  name: 'PRODUCTION',
  data: [<?php
      $production_data = "";
      foreach($production_array as &$production_queries)
      {
        $production_data .= $production_queries . ",";
      }
      // strip last comma and echo
      $production_data = substr($production_data, 0, -1);
      echo $production_data;
      ?>
      ]
}, {
```

Current State Final

Finally, this is a screenshot of our current graph. As of now, we only have the CPU graph displayed, as getting the IO graph to work is simply changing the SQL a tad. For the CPU Usage, we have it displaying from Noon to Noon, this way our overnight production(the most important processing time) is shown without getting cut off.



Project Plan



Capstone Project Gantt Chart

Alexander Sohn | December 7, 2021

	December	January	Febuary	March	April	May
Develop Shell Script	🟥					
Varabilizing Date Selection		🟥				
Implementing Date Selction			🟥			
Completing IO Usage Portion			🟥	🟥		
Testing	🟨	🟨	🟨	🟨		
Implementing into Chron Scheduler					🟥	
Final Testing					🟨	
Move Webpage to Existing Site						🟥

Testing Methods

We have already tested the current state, so before future state goals are implemented, we will test thoroughly. As we begin to reach the final states of the PHP, JavaScript, SQL, and Shell Script that we have written, we will send individual rough drafts to be looked over by our Solution Architect: Scott Travis. Once we start to put all the pieces, Scott will also test and try to break our webpage, to make sure it functions properly. After Scott is done testing all of the pieces and the prototypes are turned into final product, we will implement our solution into the existing tool website, where Scott will test for bugs and performance. When bugs or problems are found, Scott will reach out and they will be fixed and tested once again. This process will repeat until the webpage works smoothly, at which time we will announce to the rest of the team of the new tool added to the website.

Future State

In the future, we will have 2 graphs one above the other, containing CPU Usage and IO Usage. To the left of these graphs, there will be a drop down or list of dates (EX: 12-NOV-21 / 12-NOV-21) in which a user can click on and update the graphs data. In regards to the shell script, we are going to implement it to run every 3 hours using a chron scheduler (This will keep the data up to date). The main .sql file containing creation of the table will work like this: A wip table will be built, this wip table will then be renamed into the main table, and the old table will be deleted. Other than this, pulling the dates that are selected will have to be worked on and figured out (going to variablize the selection on the webpage to be input into the query and update what data is pulled into the arrays).