Final Project

Azure Log Analytics

Stromberg, Patrick

Deep Azure@McKesson

Dr. Zoran B. Djordjević

Problem Statement

- When moving solutions to the cloud, selecting appropriate sizes for your virtual machines is an issue from both a cost and performance perspective. Choose one too big, and you're paying more than you need to. Choose one too small, and your application will not scale properly. In this project, I use Log Analytics to collect various types of data during performance tests to help determine appropriate sizing of your VM.

Background / Assumptions

- The software being deployed to the cloud is a directory server, specifically ForgeRock's OpenDJ
- OpenDJ will run in a Linux VM using OpenJDK. I used CentOS, but you could use any major linux distro that is supported.
- The directory server needs to support 5,000 search requests per second
- Log Analytics will be used to analyze and visualize both
  - OS level data (think things like CPU utilization of the VM)
  - Application level data (timing specific to the searches being done against OpenDJ)

High Level Steps

- Provision 4 Linux CentOS Virtual Machines of varying strengths (either 2 or 8 CPUs and either a hard disk drive or a solid state drive)
- Set up Log Analytics in the Azure Portal and connect to the VMs to collect OS data such as CPU and disk utilization
- Install OpenDJ and sample users
- Run performance test using built-in OpenDJ tools and have custom Java code transmit results to Log Analytics
- Analyze results via Azure dashboard and queries in Azure Portal using Log Analytics

Operating System Data

- The OMS agent is installed on each VM
- It collects system-level performance data (think CPU and Disk Utilization) and submits them to Log Analytics

Performance Test Data

- Here is an example of the application-specific performance data

[dirservd@vm-pls-8core-ssd bin]$ ./searchrate -D "cn=Directory Manager" -w Test1234 -b "dc=example,dc=com" -F
-M 5000 -c 4 -t 4 -p 1389 -g "rand(0,50000)" "(uid=user.%d)"

Throughput		Response Time				
(ops/second)		(milliseconds)				
recent	average	recent	average	99.9%	99.99%	99.999%
5002.6	5002.6	0.709	0.709	13.11	22.54	22.81
4999.4	5001.0	0.498	0.603	11.73	20.19	22.68
5000.8	5000.9	0.488	0.565	10.88	19.53	22.68

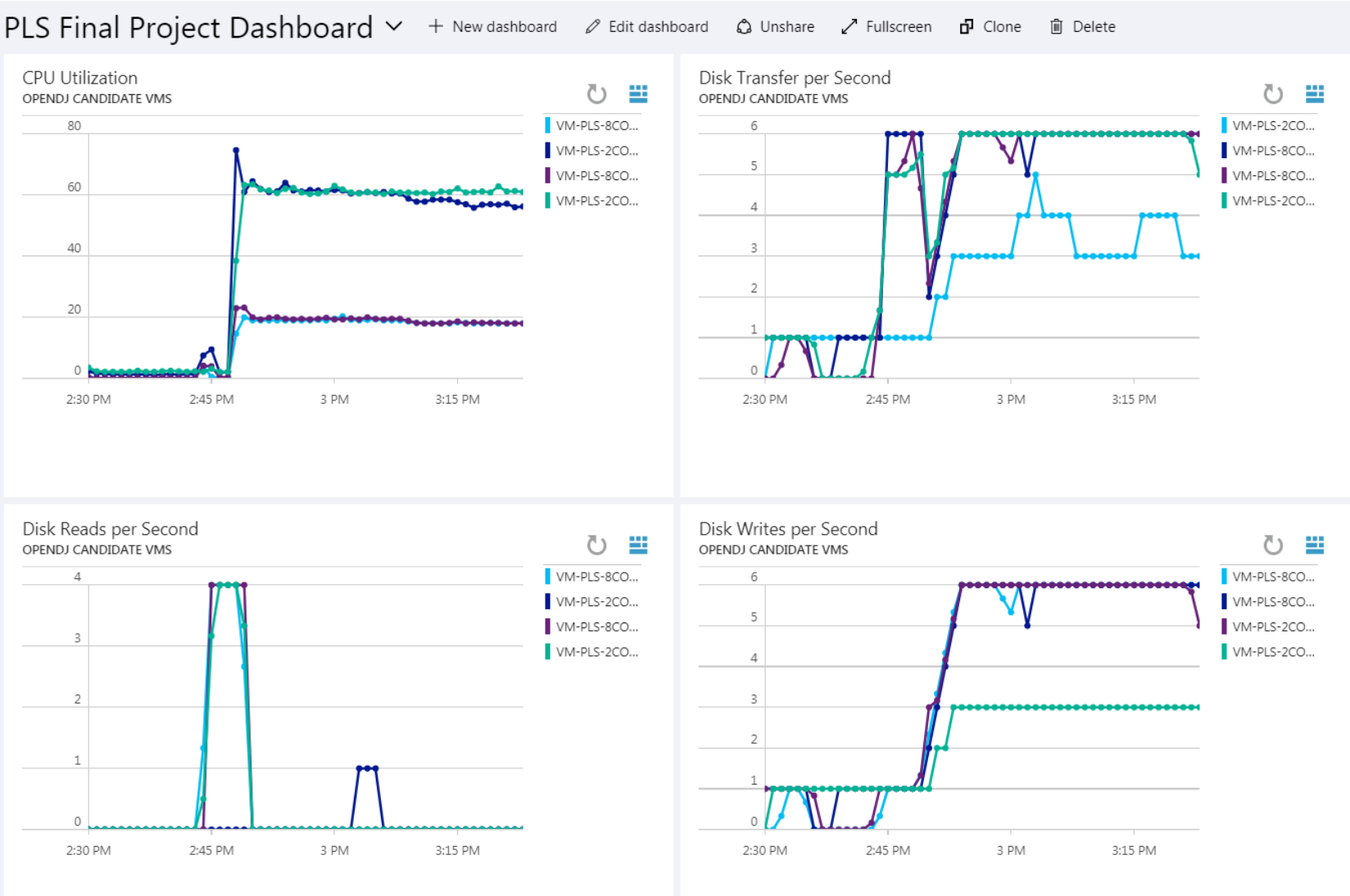
Data Transformation

- Custom Java code will transform these performance test results into a JSON object and submit them to the Data Collector API

```
JSONObject jo = new JSONObject();
if (verbose.equals("Y")) System.out.println("line = " + s);
StringTokenizer st = new StringTokenizer(s," ");
int colNum = 1;
while (st.hasMoreTokens()) {
    String value = st.nextToken();
    if (value.indexOf("|") < 0) {
        // this is not a column break - it is a number
        Double dVal = new Double(value);
        switch (colNum) {
        case 1:
            jo.put("throughput_recent", dVal);
        case 2:
            jo.put("throughput_average", dVal);
        case 3:
            jo.put("response_recent",dVal);
        case 4:
            jo.put("response_average", dVal);
        case 5:
            jo.put("reponse_999", dVal);
        }
        colNum++;
    }
}
jo.put("machine_name", description);
if (transmit.equals("Y")) {
    if (verbose.equals("Y")) System.out.println("posting " + jo.toString());
    postMessage(jo.toString());
}
```

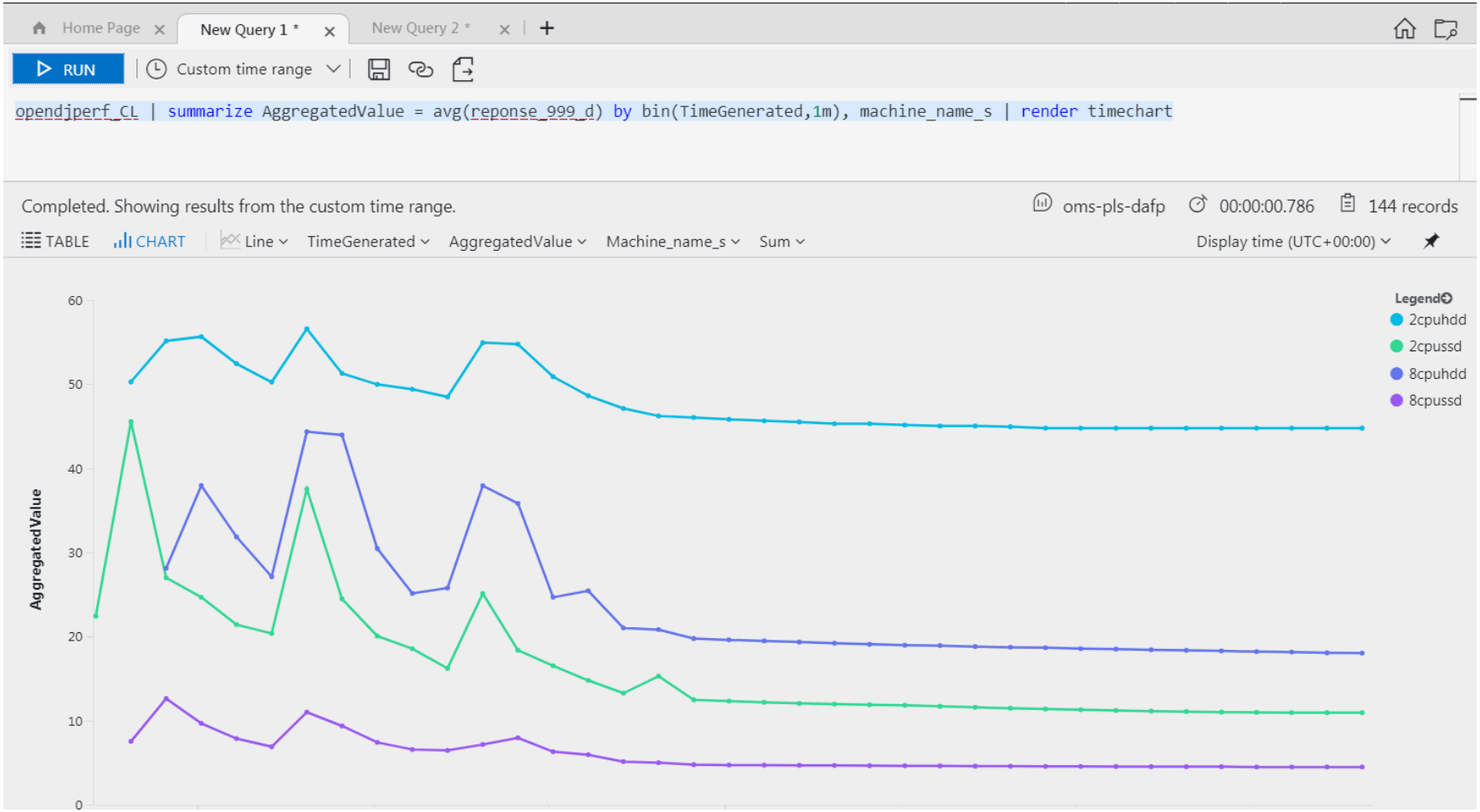
Analysis of results

- Dashboard using OS statistics



Analysis of results

- Queries using application statistics



Performance Testing Results

- The 2 CPU HDD box is not viable. It will not support the required load.
- The 2 CPU SSD box is an interesting low-cost alternative. It might require some amount of warmup before being placed into service, but it was able to outperform the 8 CPU HDD
- If cost is not an issue, or high levels of service are required, the 8 CPU SSD is the way to go.
- I'd recommend further analysis with other configurations (4 CPU?)

Lessons Learned

- I would change the directory tree structure in OpenDJ to more closely model the desired production use-case. It is not common that all users are thrown into a big pile in a single location in the directory tree. I'd also test the limits of group membership and attempt to strain the indexing strategy. There was no time to accomplish this for this assignment.
- I would automate the provisioning of the software onto the VMs using a tool like Ansible or Puppet. This would allow you to take just the binaries for OpenDJ and the custom Java code and run an automated setup against the VM. You could likely even tie the VM creation into the automated setup.
- The cloud is a fantastic performance testing lab, providing you're able to pay the bills. The ability to create n-many configurations and test them quickly and easily is priceless.
- I'm not sold on the use of Dashboards in Azure for things like this. They're nice and flashy, but you can likely get the job done using ad-hoc queries.

YouTube URLs, GitHub URL, Last Page

- Two minute (short): https://youtu.be/wIE2RjrE-WM
- 15 minutes (long): https://youtu.be/FeINq6da5II
- GitHub Repository with all artifacts: https://github.com/patstromberg/da-final