CMPE 283 – Virtualization

Project 2

Group project: this project is implemented by group effort.

Due date: May 5th (Monday), 2014

Project Overview

The theme is a large-scale statistics gathering and analysis tool in scalable virtualized environments with an emphasis on resource scheduling.

The goal of this project is:
1. Practice with virtualized environment, managing, load balancing and testing VM.
2. Apply open source tools like Logstash or Scribe
3. Understanding the need of gathering and analyze for large data
4. Visualize the collected data using the tools like google charts

Problem description
Develop a simple DRS and DPM function, and large-scale statistics gathering and analysis in scalable virtualized environments. Specific areas of interest include health models for multi-tier applications, VM and host performance, and detection of anomalies.

Part 1:
1. Simple DRS and DPM implementation

a) DRS1, initial placement: Create 2 vHosts,eachrunning 2 VMs. Each VM should run some programs such as like Prime95 or Folding@Home to keep a vCPU busy.
Create a new VM and place it among one of the two existing vHosts. Placement is based on CPU load. (capture before and after screenshots)

b) DRS2, load balancing: Create new vHost and re-balance the vmload among 3 vHosts. (create multiple VMs and load unbalance for setup, capture before and after screenshots)

c) DPM: Terminate VMs until you reach lower threshold (typically average 30% cpu load), then DPM kicks in by consolidating VMs into two vHosts and shut down one vHost. Repeat the same until you reach one vHost. (capture before and after screenshots)

Part 2:

2. Framework

Build a framework for large scale metrics data collection and analysis and display of the collected and analyzed data

The types of operations that the framework will have are:

1. Performing collection of system data (metrics, logs) to identify workloads on system elements (jobs, hosts, guests etc) as you have done in project 1 and store them in noSQL Database such as Cassandra or Mongo DB. Data rate can reach 100,000 per minutes.

Monitor Hosts and collect the following metrics (Per Guest Level CPU, Memory, Threads, I/O, VMotion)

2. Applying higher-level processing (average per 5 mins intervals, etc) on collected system data to generate abstract views of the system. You can store them in Relational DB such as MySQL, Postgress, BerkeleyDB.
3. Presenting and/or visualizing the outcomes of the above steps in a simple manner.

4. Experiment security options in ESXi such as installing security option for ESXi and experiment with various security options, collect statistics, analyze performance measurements, and report the results.

3. Design Components

The system will have at the minimum the following components:

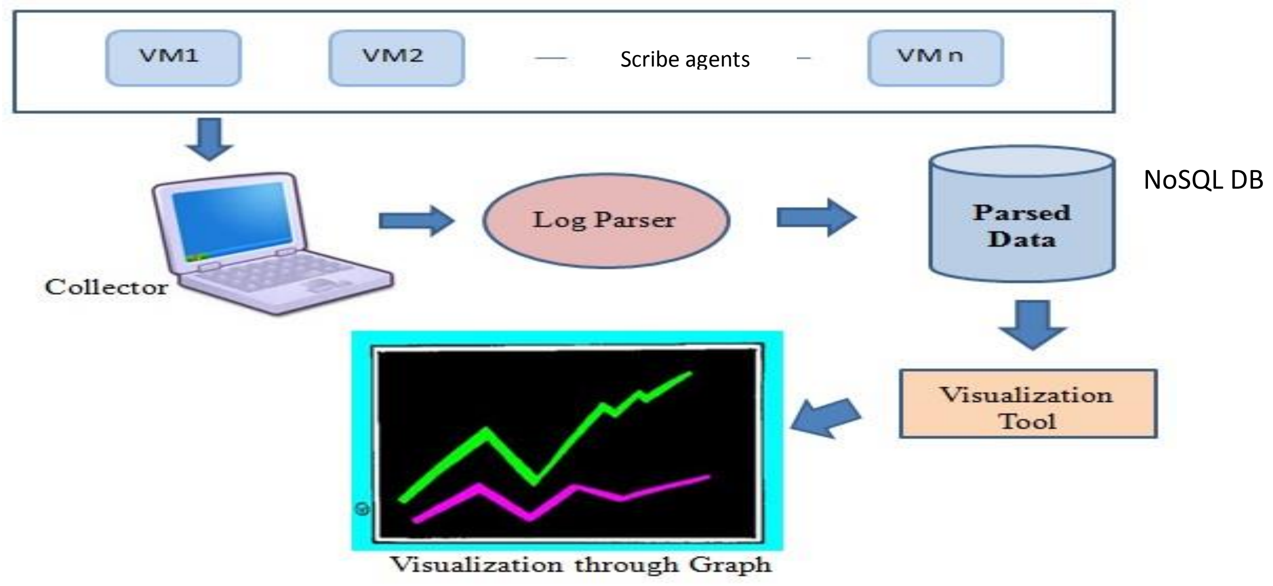
- One or more agent processes sitting at the ESXi kernel level that collect and send monitoring data to the collector
- In order to measure health of application across multiple tiers, access to packets is required.
- Collector Process that will Poll, Process and Archive collected data
- Analysis module to process the collected data
- A Supervisor module that can configure/ manage/ control the various agents
- A simple visualization Tool for above data

4. Configurations

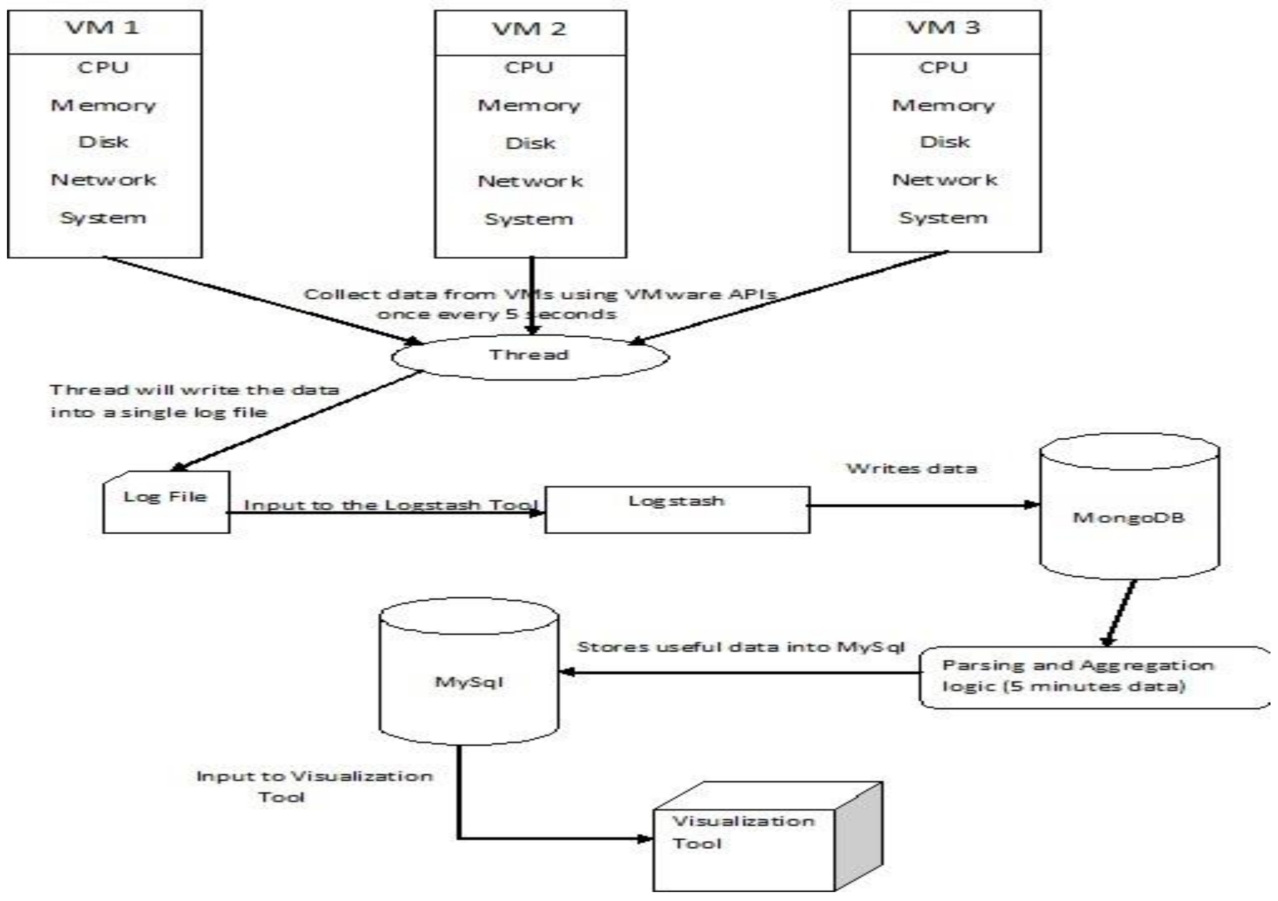
- List of agents and locations
- Environment and configurations for Collectors, port numbers for agents, locations to store data, Polling intervals, etc Startup and Shutdown
- Agents and Collector can be started independently
- Start and stop scripts can also be used to manage them independently

Simulate work load - There needs to be a way to stimulate activity. This could involve some free apps that are designed to stress-test a real CPU, like Prime95 or Folding@Home to keep a vCPU busy. The students could start up one or more instances of these apps and try to force VMware to auto-migrate to another Host with more CPU resources. (Most of these are all platform: Windows, Linux, Mac.)

5. Process and Data Flow



1. Agents will collect the configured metrics and commit as local record tables on each host, every 5 seconds
2. Pollers poll the agents and get record tables back via suitable Transport (TCP/ UDP).
3. After collection is successful, data records will be archived and the engines will be notified.
4. These engines will then analyze and reduce data, and create second set of tables that has analyzed values
5. Hourly Analyzed Record Rollup Process will kick in every hour for rolling up data into 24 sets
6. Daily Analyzed Records Rollup. 24 hour Process will kick in for rolling up data from the 24 sets into daily sets
7. Purge and Cleanup
8. Visualization widgets will run on top of the collected and analyzed data. E.g. Per VM and System wide Histograms, Heat Maps etc, all with events coalesced by time, allowing superimpositions. Latency of Tiered applications co-hosted with per VM Guest/ Network statistics etc

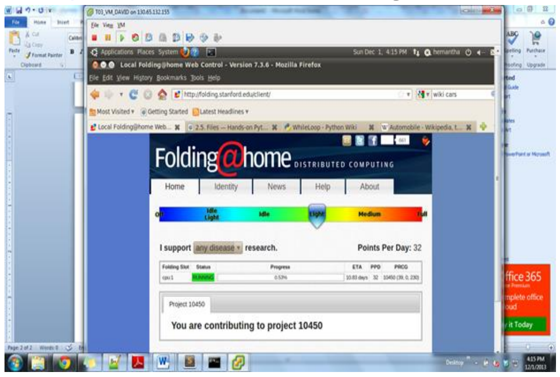

VM 1
CPU
Memory
Disk
Network
System
VM 2
CPU
Memory
Disk
Network
System
VM 3
CPU
Memory
Disk
Network
System
Collect data from VMs using VMware APIs once every 5 seconds
Thread
Thread will write the data into a single log file
Log File
Input to the Logstash Tool
Logstash
Writes data
MongoDB
Stores useful data into MySql
Parsing and Aggregation logic (5 minutes data)
MySql
Input to Visualization Tool
Visualization Tool

6. Other Related Work

There are other systems and frameworks that collect logs and perform analysis.

1. SPLUNK treats each log entry as an event and It is one of the most popular large scale log gathering tools which is available in the market. Splunk tries to view each log entry as an event and tries to process this event depending on the desired outcome of result. Event processing occurs in two stages, parsing and indexing. All data that comes into Splunk enters through the parsing pipeline as large chunks. During parsing, Splunk breaks these chunks into events which it hands off to the indexing pipeline, where final processing occurs. During both parsing and indexing, Splunk acts on the data, transforming it in various ways.
2. Logstash: a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use.
3. SALSA(Analyzing Logs as StAte Machines): SALSA examines system logs to derive state-machine views of the sytem's execution, along with control-flow, data-flow models and related statistics. Exploiting SALSA's derived views and statistics, we can effectively construct higher-level useful analyses.
4. Scribe is a server for aggregating log data streamed in real-time from a large number of servers. It is designed to be scalable, extensible without client-side modification
5. Jquery, Jqplot, HTML5:it is a graphing library integrated with jquery and javascript for plotting dynamic graphs with zooming functionality
6. google chart (Charts4J), high-charts: is a simple tool that lets people easily create a chart from some data and embed it in a web page.

7. Grok made the formatting of the collected data easier
8. Prime95:http://en.wikipedia.org/wiki/Prime95
9. Folding@home (Stanford): http://folding.stanford.edu/English/Guide one can configure how much resources should be allowed to be utilized.


Local Folding@home Web Control - Version 7.3.6 - Mozilla Firefox

http://folding.stanford.edu/client/

I support any disease research. Points Per Day: 32

Folding Slot	Status	Progress	ETA	PPD	PRCG
cpu:1	RUNNING	0.53%	10.83 days	32	10450 (39, 0, 230)

Project 10450

You are contributing to project 10450

7. Deliverables

- Complete source code (do not include object files)
- Screenshots
- Test scripts
- Project report (Why? and Why not?)
    - Introduction (part1 and part 2: goals, objectives, needs, …)
    - Background (part1 and part2: context of project)
    - Requirements (part2: functional and nonfunctional)
    - Design (part1: describe algorithms, part2: architecture, components, key workflows, …)
    - Implementation (part1: how to implement algorithm, part2: environment, tools, approaches, scripting language and tools uses, screenshots, who did what, …)
    - Assumptions: (part1 and part2) What are the assumptions and why?
    - Limitations: (part1 and part2) What are limitations and why?
    - Future Work: (part1 and part2) how this work can be extended?
    - Individual Contribution: (part1 and part2) Who did what?
    - Installation and Execution manual: (part2 only)What and how to execute your codes? Show screenshots.
    - System Testing Results: (part2 only) What/How did you test? Describe all your test methodologies, test cases and show screenshots.
    - Testing (part2 only) (unit, integration, who did what, …)
    - Screenshots with annotations (both part1 and part2)
    - Conclusion (part1 and part2: lessons learned, challenges …)
    - References

8. Submission

- **On-line submission**: Submit all codes and report as one zip file. Submissions shall be made via canvas. **Do not send it to the group mailing**. DO NOT WAIT UNTIL LATE ON THE DUE DATE, as email server lags or delays may result in a late submission. If you are concerned about this, submit your assignment early. Shall include image captures of your program working.
- **Demo and presentations are on/after due date.**
- **No hardcopy submission is required.**