Design and implementation of monitoring system

{kapalani, magevadi, shivjana, vsridhar}@indiana.edu

I. **Monitoring Daemon**: Publisherclient.java

1) We captured the CPU and Memory utilization of a local system and Eucalyptus Virtual Machine for a period of 30 seconds using the following commands from the Sigar library:

- sigar.getCpuPerc().getCombined()
- sigar.getMem().getUsedPercent()

2) We then communicated with the Message Broker and published the messages.

Code:

```
//get CPU and MEM utilization
public void output(CpuPerc cpu) throws SigarException {
      cpucom = cpu.getCombined();
      memcom = sigar.getMem().getUsedPercent();
}
```

II. Monitoring frontend: Consumerclient.java

1) We first subscribed to the relevant topic in the Message Broker and receive the messages with monitoring information from the daemons running on the local node. We used Jfree chart to build the UI.

Code:

```
// Receive message from Broker.
public void onEvent(NBEvent message) {
      System.out.println("Received Event {" + message.getContentSynopsis() + "} "
            + new String(message.getContentPayload()));
      combined_perf = new String(message.getContentPayload()).toString(); //Storing Combined Performance
      perf_array = combined_perf.split(",");
      cpu_perf = perf_array[0];
      mem_util = perf_array[1];
      System.out.println("Cpu Performance: " + cpu_perf +"\nMemory Utilization: "+mem_util);
      //custom_render(Double.parseDouble(cpu_perf), Double.parseDouble(mem_util));
      demo.performance_plot(Double.parseDouble(cpu_perf), Double.parseDouble(mem_util));
   }
```

2) **Render.java**

This is the file that plots the graph dynamically as consumer client sends the data to it.

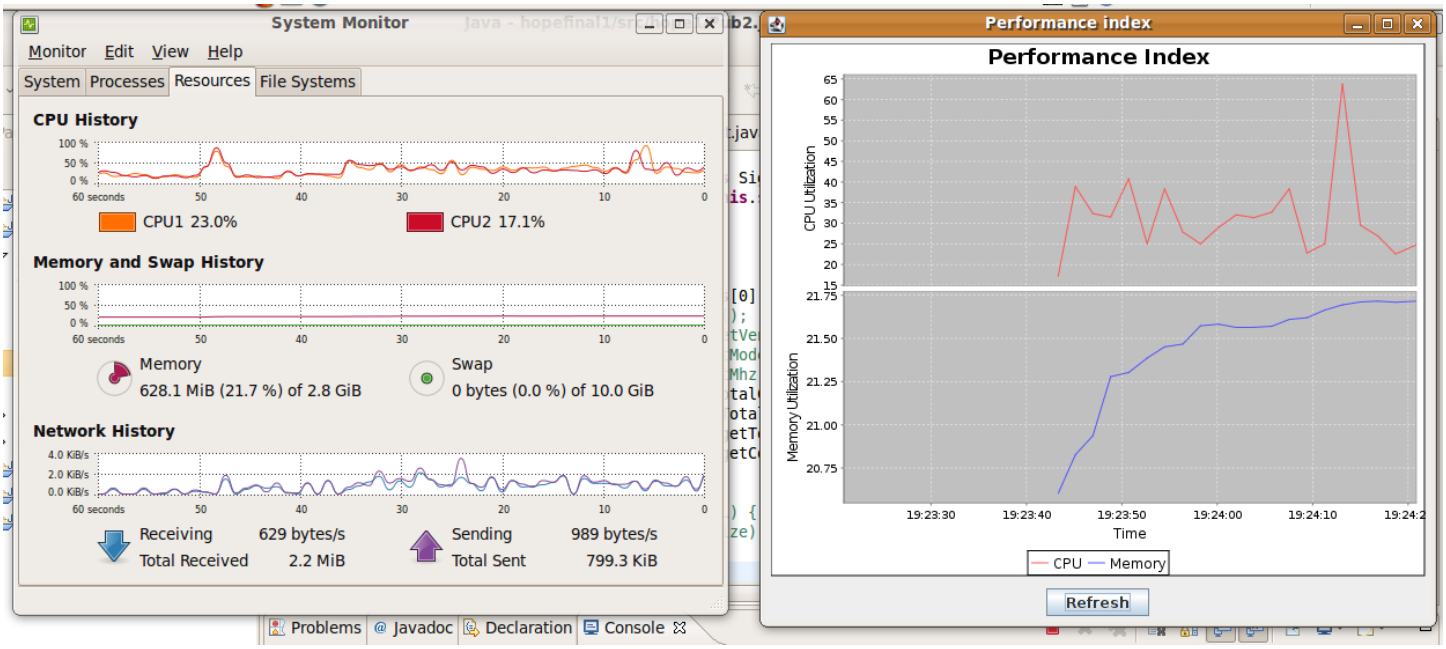
```
//plots graph

public void performance_plot(double cpu_util, double mem_used)
    {
       this.datasets[0].getSeries(0).add(new Millisecond(), cpu_util);
```

```
        this.datasets[1].getSeries(0).add(new Millisecond(), mem_used);
    }
```

Graph snapshot:

Performance Index when daemon run on Local commercial laptop:

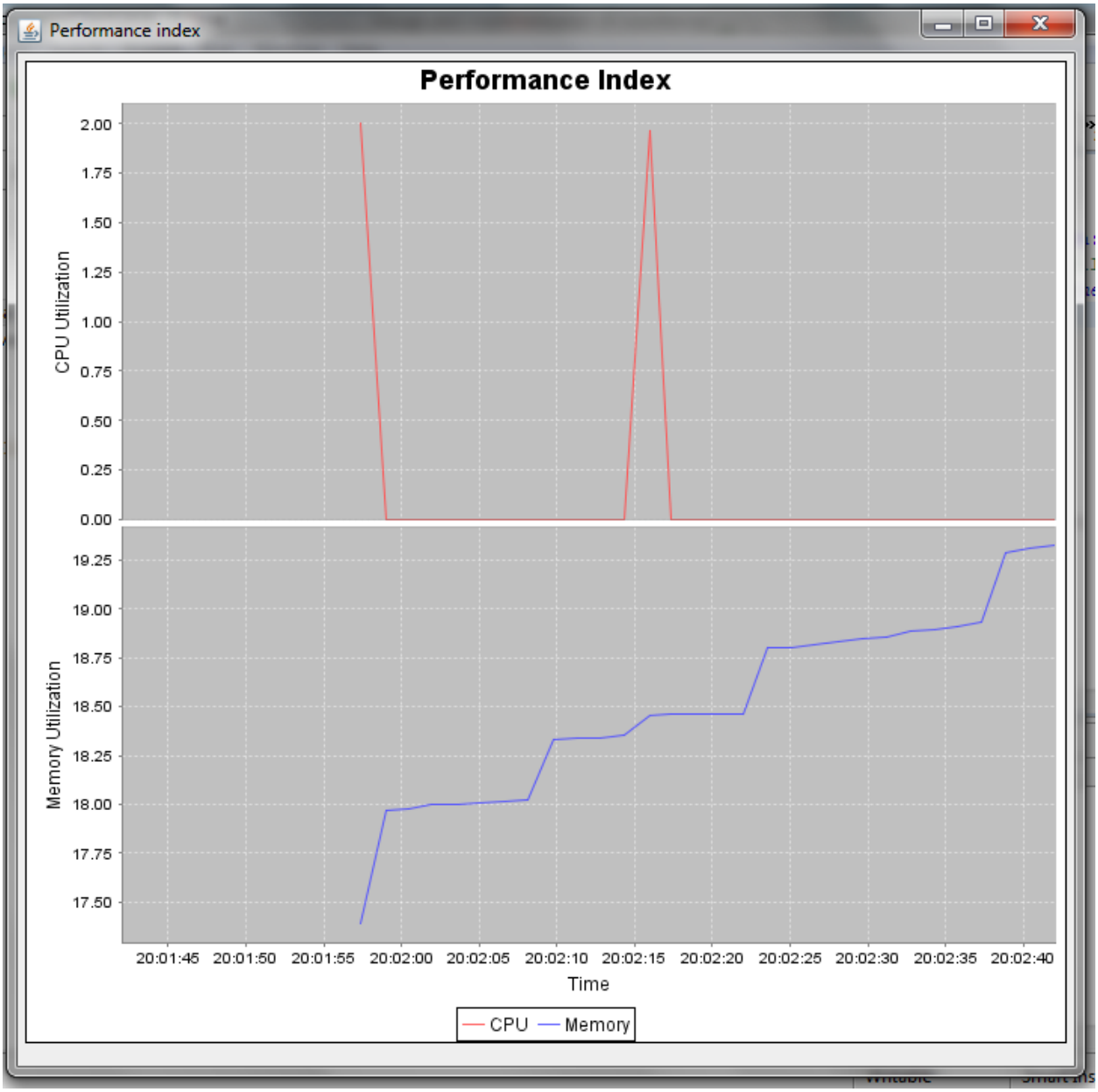


Green-CPU utilization

Red – Memory utilization.

The above picture clearly shows the similarity between local system monitor in Ubuntu and our project performance Index. CPU spikes in system clearly matches with our Jfree chart.

Performance Index when daemon run on Eucalyptus virtual machine:



Green-CPU utilization
Red – Memory utilization.

The above index is the result of running the daemon on single instance of Eucalyptus virtual machine. Our idea about multiple instances is, taking an average between non matching node name's CPU and Memory utilization. For simplicity and accuracy we can ignore groups that have any of the nodes missing(delayed propagation). This way, our graph can be plotted with minimal error, even without synchronization.