DYNAMIC WORKLOAD-AWARE ALGORITHM FOR VM PLACEMENT AND MIGRATION IN OPTICAL DATA CENTER TO MINIMIZE ELECTRICITY COST

K M Sabidur Rahman

May 6, 2015

Friday Group Meeting, Netlab

UC Davis

Previously on my presentations

- What is VM migration and why how it works
- Various reasons behind VM migration
- Motivation behind electricity cost minimization
- Electricity market
- Spatio-temporal properties of electricity cost
- Typical server and rack configuration in DCs
- Literature review

Previously on my presentations

- Power model for such server-rack-DCs (considering PUE)
- Power model for Live VM Migration from one DC to another DC
- Data Center locations of Google/Amazon
- Characterization of workloads
- How we can build a problem statement for a dynamic workload scenario

Today's agenda

- Problem statement
- Heuristic algorithm
- Simulation related considerations

Problem statement

Given:

- Optical backbone network topology
- Set of DCs
- Link capacities
- Dynamic workload
- Hourly prices of electricity at each DC
- Capacity limit of each DC

Problem statement

Provide a **dynamic workload-aware algorithm** which makes decision for placing the new requests and migration of the old VM running, based on:

- Electricity cost at various location
- VM migration cost
- SLA limit for allowable delay
- VM consolidation
- Bandwidth and capacity limit

Heuristic Algorithm

Step 1. At the arrival of each request*, we know:

- How many VMs are currently running in which Rack and which DC
- The topology and capacity limit of racks and DCs (Typical limit in Industry standard for simulation purpose)
- DCs – number of Data Centers and location
- Racks- number of Racks at each DC
- Servers and VMs- typical limit on number of Servers and/or VMs in each DC

(I am looking for references of these numbers)

- Cost of electricity in each DC for next 24 hours

*request: depends on the user request model. Duration and tolerable propagation delay are given for each request

Heuristic Algorithm

Step 2. At the arrival of each request, we calculate the following:

- Cost of running new VMs in each DC.
- Assign request to the DC of the best choice. (According to cost and delay limitation from SLA). VMs already running are taken care of in Step 3, when we migrate VMs.

Heuristic Algorithm

Step 3. After each epoch*:

- Calculate **Cost of migrating**** already running VMs to other DC. DC has to be within the **delay threshold1***** from SLA. **Remaining duration****** of the request has to be considered. Considering VM consolidation too.
- Migrate the VMs to minimize the total running costs of all DCs. (Considering cost from Step 3a + **penalty*******)

Step 4. Go to Step 1 for the next arrival.

*epoch: variable. Can be defined by number of requests. Or by time range.

Considerations

****Remaining duration: Each request will have a required service duration. And we can easily calculate the remaining duration from there.

*****penalty: Additional cost to make sure that we are not moving around lots of VM at every epoch.

Considerations

**Cost of migration: Cost of VM running on source DC + cost of bandwidth over internet + cost of VM running on destination DC

***Delay thresold1: each user request will come with a delay tolerance threshold (mostly tied to the application type). So, we can’t place the request to a VM which is too far from the user. This is user to VM end-to-end delay. We are not considering service/queuing delay.

Thank You