Design Project 2: Virtual Machine Placement in a Data Center Network

Tiffany Yu-Han Chen
Data Center Network

The network you are using in your DP2
Data Center (DC) Networks

- DC networks are organized in a hierarchical way
DC Networks

- Each physical machine has a unique ID
DC Networks

- These machines are divided into groups
DC Networks

- Machines in the same group are connected with a extremely fast network connection.
DC Networks
DC Networks
DC Networks
Virtual Machines (VMs)

- A physical machine is home to many different VMs
- Each physical machine can host 4 VMs
Running Jobs in a DC Network

• It’s common to run large computation tasks in public DCs
  • Amazon EC2, Windows Azure
Running Jobs in a DC Network

• It’s common to run large computation tasks in public DCs
  • Amazon EC2, Windows Azure

• User divides the computation into smaller jobs and puts each job in a VM
  • VMs are placed on various physical machines in the DC
  • VMs communicate with each other to finish the task
Running Jobs in a DC Network

• It’s common to run large computation tasks in public DCs
  • Amazon EC2, Windows Azure

• User divides the computation into smaller jobs and puts each job in a VM
  • VMs are placed on various physical machines in the DC
  • VMs communicate with each other to finish the task

• **Users are paying for each VM (!)**
  • $0.1/min per VM
Summary

• The price of a VM - $0.1/min
VM Placement Problem
VM Placement Problem

• Your app needs multiple VMs, and they can be located anywhere on the network
VM Placement Problem

- Your app needs multiple VMs, and they can be located anywhere on the network
- If your app has to transfer a large amount of data, the network becomes the bottleneck
VM Placement Problem

• Your app needs multiple VMs, and they can be located anywhere on the network
• If your app has to transfer a large amount of data, the network becomes the bottleneck
• Connections become slower if other users are sharing the same path(s) that your traffic is using
VM Placement Problem

• Your app needs multiple VMs, and they can be located anywhere on the network
• If your app has to transfer a large amount of data, the network becomes the bottleneck
• Connections become slower if other users are sharing the same path(s) that your traffic is using
Your Job

• Given
  1. The DC network topology
Your Job

• Given
  1. The DC network topology
  2. The number of VMs the app uses = n
Your Job

• Given

1. The DC network topology
2. The number of VMs the app uses = n
3. The **total** amount of data that each pair of VMs will transfer

\[
B = \begin{bmatrix}
0 & 10MB & 0 \\
2MB & 0 & 3MB \\
0 & 0 & 0 \\
\end{bmatrix}
\]
Your Job

• Given
  1. The DC network topology
  2. The number of VMs the app uses = n
  3. The total amount of data that each pair of VMs will transfer

\[
B = \begin{bmatrix}
0 & 10\text{MB} & 0 \\
2\text{MB} & 0 & 3\text{MB} \\
0 & 0 & 0
\end{bmatrix}
\]

• Place your VMs to minimize the time until the completion of the app
  • Time ↓, Cost ↓ ($0.1/\text{min per VM}$)
Your Job

• Given
  1. The DC network topology
  2. The number of VMs the app uses = n
  3. The **total** amount of data that each pair of VMs will transfer

\[
\begin{bmatrix}
0 & 10\text{MB} & 0 \\
2\text{MB} & 0 & 3\text{MB} \\
0 & 0 & 0
\end{bmatrix}
\]

• Place your VMs to minimize the time until the completion of the app
  • Time ↓, Cost ↓ ($0.1/\text{min per VM}$)

• Adapt the placement in real time to cope with nwk changes
  • Arrival/departure of other clients
  • Completion of existing task
Available Functions

1. `bool place(v, m)`
   - Place virtual machine v on physical machine m

2. `machine_id random_place(v)`
   - Place virtual machine v on a random physical machine, and returns that machine's ID

3. `int progress(u, v)`
   - Returns the number of bytes that virtual machines u and v have left to transfer to each other.

4. `int machine_occupancy(m)`
   - Return the number of VMs currently running on PM m.

5. `double tcp_throughput(v)`
   - Return the throughput of the TCP connection from this VM to VM v over the last 100ms (passive monitoring)
System Design
System Components

• **Measurement** – Learn the properties of the paths between the VMs
  • What to measure?
    • Available bandwidth? App’s throughput?
  • How to measure?
    • Active probing/passive monitoring?
  • How often to measure?
  • Overhead of measurements
    • $, traffic
System Components

• **Measurement** – Learn the properties of the paths between the VMs
  • What to measure?
    • Available bandwidth? App’s throughput?
  • How to measure?
    • Active probing/passive monitoring?
  • How often to measure?
  • Overhead of measurements

• It takes a few RTTs to get an accurate measurement
  • Ex. Measuring the throughput (netperf) takes *multiple seconds*
System Components

- **Placement** – given the measurements, where do you place the VMs
  - How to make the placement decision?
  - How do you interact with other users?
  - Is everything distributed/centralized?
Straw man #1

For v in VMs, \texttt{random\_place}(v)
Straw man #1

For v in VMs, \texttt{random\_place(v)}

- Ignores other paths that may have significantly higher throughput
- Does not consider other clients
Straw man #2

Try to put all the VMs in the same group

For each group
  For m in PMs,
    available_vms += machine_occupancy(m)
    if available_vms >= n
      put all the VMs in this group
Straw man #2

Try to put all the VMs in the same group

For each group
  For m in PMs,
    available_vms += \texttt{machine\_occupancy}(m)
    if available_vms >= n
      put all the VMs in this group

• You may not have enough VMs in one group
Straw man #3 - Straggler

1. Place all VMs randomly

2. Loop repeatedly
   a. Collect progress() values between all pairs
   b. Compute %progress (using matrix B – total to be transferred)
   c. Pick the pair making least %progress
      - Move one of the machines to a different random location
Straw man #3 - Straggler

1. Place all VMs randomly

2. Loop repeatedly
   a. Collect progress() values between all pairs
   b. Compute %progress (using matrix B – total to be transferred)
   c. Pick the pair making least %progress
      - Move one of the machines to a different random location

• Ignores other paths that may have significantly higher throughput.
Straw man #4

while more bytes to transfer between any pair of VMs
pick a pair of VMs that haven't completed their transfers
place them on the same machine
complete their transfers
Straw man #4

while more bytes to transfer between any pair of VMs
pick a pair of VMs that haven't completed their transfers
place them on the same machine
complete their transfers

• You may not have enough VMs in one group
• The path between the VMs that you pick may deliver low throughput.
Some hints...

• Explore better paths by spawning extra VMs
  • Tradeoff: cost
Some hints...

• Explore better paths by spawning extra VMs
  • Tradeoff: cost

• Passive monitoring vs. active probing
  • Passive monitoring
    • Does not provide accurate measurement, (but might be useful?)
  • Active probing
    • Accurate but might take multiple seconds
Some hints...

• Explore better paths by spawning extra VMs
  • Tradeoff: cost

• Passive monitoring vs. active probing
  • Passive monitoring
    • Does not provide accurate measurement, (but might be useful?)
  • Active probing
    • Accurate but might take multiple seconds

• Exploit the DC network topology
Lessons from DP1

• Detailed performance analysis
  • Provide real numbers
  • Get your hands dirty and do some real measurements
• Detailed explanation on the use cases
• Guideline
  • Make reasonable assumptions
  • Try your best to justify your design
  • Persuade your instructor to implement your design