

Network-aware Virtual Machine Consolidation for Large Data Centers

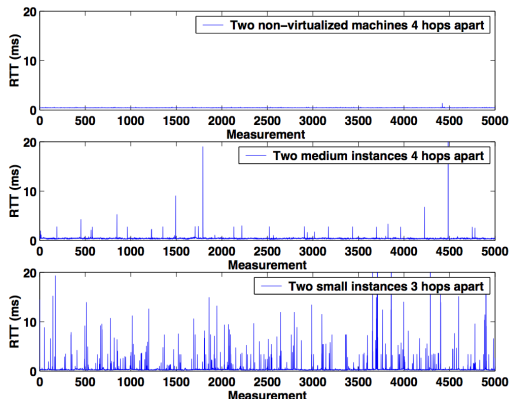
Dharmesh Kakadia¹, Nandish Kopri² and Vasudeva Varma¹

¹IIT-Hyderabad, India

²Unisys Corp., India

Network Performance in Cloud

- ▶ In Amazon EC2, TCP/UDP throughput experienced by applications can fluctuate rapidly between 1 Gb/s and zero.
- ▶ abnormally large packet delay variations among Amazon EC2 instances.



1

¹G. Wang et al. *The impact of virtualization on network performance of amazon ec2 data center.* (INFOCOM'2010)

Scalability

- ▶ Scheduling algorithm has to scale to millions of requests
- ▶ Network traffic at higher layers pose significant challenge for data center network scaling
- ▶ New applications in data center are pushing need for traffic localization in data center network

Problem

VM placement algorithm to consolidate VMs using
network traffic patterns

Subproblems

- ▶ *How to identify?* - cluster VMs based on their traffic exchange patterns
- ▶ *How to place?* -placement algorithm to place VMs to localize internal datacenter traffic and improve application performance

How to identify?

VMCluster is a group of VMs that has large communication cost (c_{ij}) over time period T .

How to identify?

VMCluster is a group of VMs that has large communication cost (c_{ij}) over time period T .

$$c_{ij} = \text{AccessRate}_{ij} \times \text{Delay}_{ij}$$

AccessRate_{ij} is rate of data exchange between VM_i and VM_j and Delay_{ij} is the communication delay between them.

VMCluster Formation Algorithm

$$AccessMatrix_{n \times n} = \begin{bmatrix} 0 & c_{12} & \cdots & c_{1n} \\ c_{21} & 0 & \cdots & c_{2n} \\ \vdots & \vdots & & \vdots \\ c_{n1} & c_{n2} & \cdots & 0 \end{bmatrix}$$

c_{ij} is maintained over time period T in moving window fashion and mean is taken as the value.

```
for each row  $A_i \in AccessMatrix$  do  
  if  $\maxElement(A_i) > (1 + opt\_threshold) * avg\_comm\_cost$   
  then  
    form a new VMCluster from non-zero elements of  $A_i$   
  end if  
end for
```


How to place ?

How to place ?

- ▶ Which VM to migrate?

How to place ?

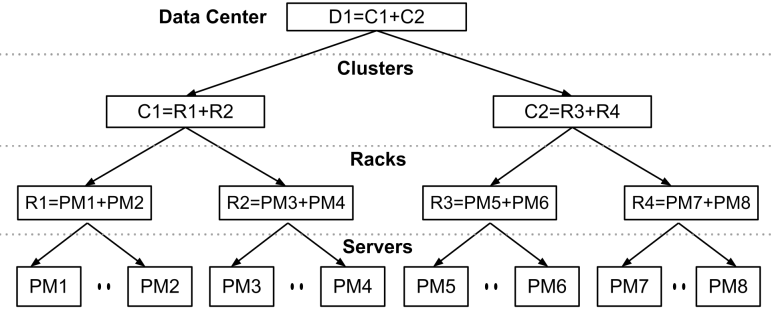
- ▶ Which VM to migrate?
- ▶ Where *can* we migrate?

How to place ?

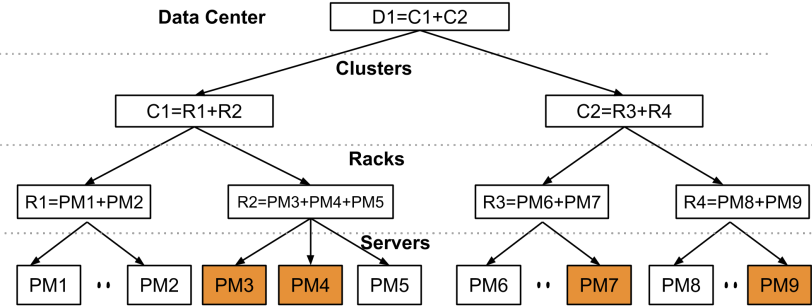
- ▶ Which VM to migrate?
- ▶ Where *can* we migrate?
- ▶ Will the the effort be worth?

Communication Cost Tree

- ▶ Each node represents cost of communication of devices connected to it.

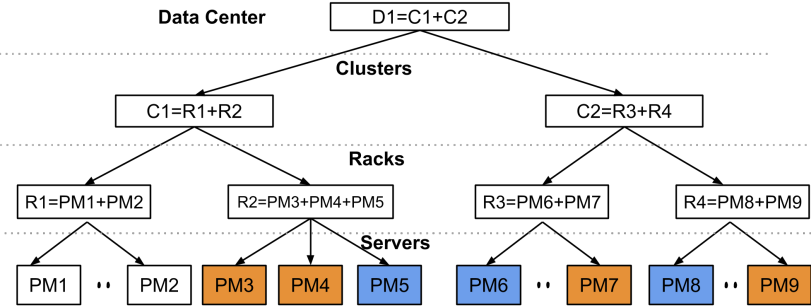


Example : VMCluster



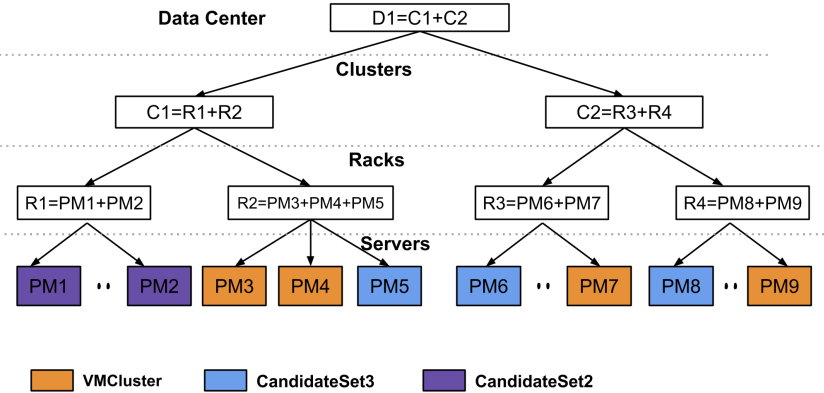
 VMCluster

Example : *CandidateSet3*



 VMCluster  CandidateSet3

Example : *CandidateSet₂*



How to place ?

How to place ?

Which VM to migrate?

$$VMtoMigrate = \arg \max_{VM_i} \sum_{j=1}^{|VMCluster|} C_{ij}$$

How to place ?

Which VM to migrate?

$$VMtoMigrate = \arg \max_{VM_i} \sum_{j=1}^{|VMCluster|} c_{ij}$$

Where *can* we migrate?

$CandidateSet_i(VMCluster_j) = \{c \mid \text{where } c \text{ and } VMCluster_j \text{ have a common ancestor at level } i\}$
– $CandidateSet_{i+1}(VMCluster_j)$

How to place ?

Which VM to migrate?

$$VMtoMigrate = \arg \max_{VM_i} \sum_{j=1}^{|VMCluster|} c_{ij}$$

Where *can* we migrate?

$CandidateSet_i(VMCluster_j) = \{c \mid \text{where } c \text{ and } VMCluster_j \text{ have a common ancestor at level } i\}$
– $CandidateSet_{i+1}(VMCluster_j)$

Will the the effort be worth?

$$PerfGain = \sum_{j=1}^{|VMCluster|} \frac{c_{ij} - c'_{ij}}{c_{ij}}$$

Consolidation Algorithm

- ▶ Select the VM to migrate
- ▶ Identify CandidateSets
- ▶ Select destination PM
 - ▶ Overload the destination
 - ▶ Gain is significant

Consolidation Algorithm

```
for  $VMCluster_j \in VMClusters$  do  
  Select  $VMtoMigrate$   
  for  $i$  from leaf to root do  
    Form  $CandidateSet_i(VMCluster_j - VMtoMigrate)$   
    for  $PM \in candidateSet_i$  do  
      if  $UtilAfterMigration(PM, VMtoMigrate)$   
         $< overload\_threshold$  AND  $PerfGain(PM, VMtoMigrate)$   
         $> significance\_threshold$  then  
          migrate VM to PM  
          continue to next VMCluster  
        end if  
      end for  
    end for  
  end for
```

Trace Statistics

Traces from three real world data centers, two from universities (uni1, uni2) and one from private data center (prv1) [4].

Property	Uni1	Uni2	Prv1
Number of Short non-I/O-intensive jobs	513	3637	3152
Number of Short I/O-intensive jobs	223	1834	1798
Number of Medium non-I/O-intensive jobs	135	628	173
Number of Medium I/O-intensive jobs	186	864	231
Number of Long non-I/O-intensive jobs	112	319	59
Number of Long I/O-intensive jobs	160	418	358
Number of Servers	500	1093	1088
Number of Devices	22	36	96
Over Subscription	2:1	47:1	8:3

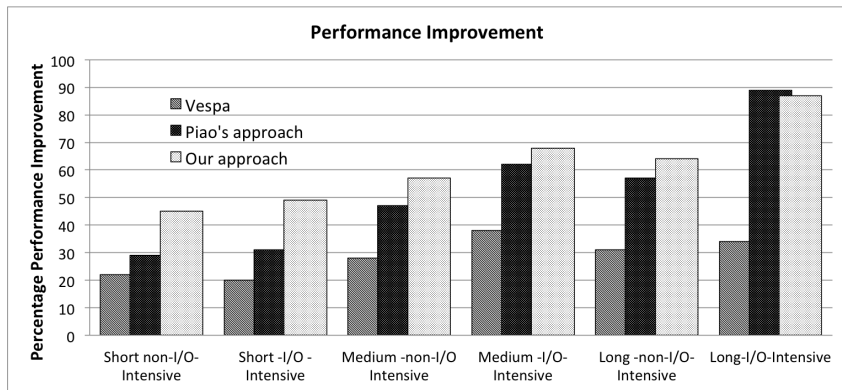
Experimental Evaluation

We compared our approach to traditional placement approaches like Vespa [1] and previous network-aware algorithm like Piao's approach [2].

- ▶ Extended NetworkCloudSim [3] to support SDN.
- ▶ Floodlight² as our SDN controller.
- ▶ The server properties are assumed to be HP ProLiant ML110 G5 (1 × [Xeon 3075 2660 MHz, 2 cores]), 4GB) connected through 1G using HP ProCurve switches.

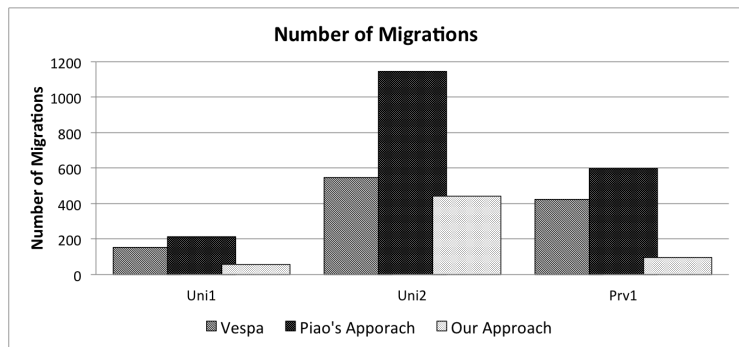
²<http://www.projectfloodlight.org/>

Results : Performance Improvement



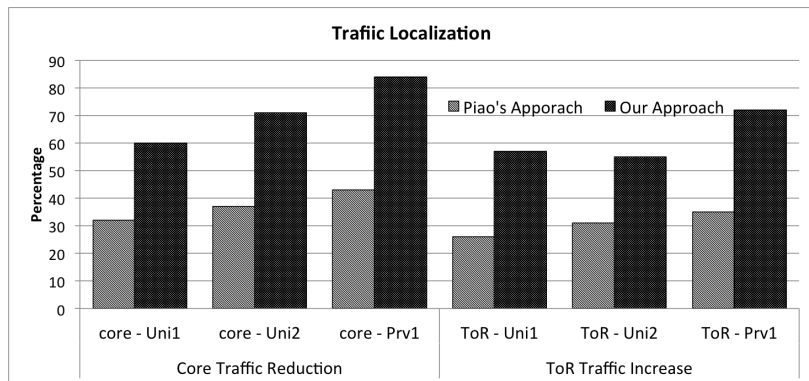
- ▶ I/O intensive jobs are benefited most, but others also share the benefit
- ▶ Short jobs are important for overall performance improvement

Results : Number of Migrations



- ▶ Every migration is not equally beneficial

Results : Traffic Localization



- ▶ 60% increase ToR traffic (vs 30% by Piao's approach)
- ▶ 70% decrease Core traffic (vs 37% by Piao's approach)

Results : Complexity – Time, Variance and Migrations

Measure	Trace	Vespa	Piao's approach	Our approach
Avg. scheduling Time (ms)	Uni1	504	677	217
	Uni2	784	1197	376
	Prv1	718	1076	324

Results : Complexity – Time, Variance and Migrations

Measure	Trace	Vespa	Piao's approach	Our approach
Avg. scheduling Time (ms)	Uni1	504	677	217
	Uni2	784	1197	376
	Prv1	718	1076	324
Worst-case scheduling Time (ms)	Uni1	846	1087	502
	Uni2	973	1316	558
	Prv1	894	1278	539

Results : Complexity – Time, Variance and Migrations

Measure	Trace	Vespa	Piao's approach	Our approach
Avg. scheduling Time (ms)	Uni1	504	677	217
	Uni2	784	1197	376
	Prv1	718	1076	324
Worst-case scheduling Time (ms)	Uni1	846	1087	502
	Uni2	973	1316	558
	Prv1	894	1278	539
Variance in scheduling Time	Uni1	179	146	70
	Uni2	234	246	98
	Prv1	214	216	89

Results : Complexity – Time, Variance and Migrations

Measure	Trace	Vespa	Piao's approach	Our approach
Avg. scheduling Time (ms)	Uni1	504	677	217
	Uni2	784	1197	376
	Prv1	718	1076	324
Worst-case scheduling Time (ms)	Uni1	846	1087	502
	Uni2	973	1316	558
	Prv1	894	1278	539
Variance in scheduling Time	Uni1	179	146	70
	Uni2	234	246	98
	Prv1	214	216	89
Number of Migrations	Uni1	154	213	56
	Uni2	547	1145	441
	Prv1	423	597	96

Conclusion

- ▶ Network aware placement (and traffic localization) helps in Network scaling.
- ▶ VM Scheduler should be aware of migrations.
- ▶ Think like a scheduler and think *rationally*. You may not want *all* the migrations.

Thank you

Send your queries to

@DharmeshKakadia

dharmesh.kakadia@research.iiit.ac.in

References

1. C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici. A scalable application placement controller for enterprise data centers. (WWW'2007)
2. J. Piao and J. Yan. A network-aware virtual machine placement and migration approach in cloud computing. (GCC'2010)
3. S. K. Garg and R. Buyya. Networkcloudsim: Modelling parallel applications in cloud simulations. (UCC'2011)
4. T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. (IMC'2010)