Experimental Evaluation of SDN-Controlled, Joint Consolidation of Policies and Virtual Machines

Wajdi Hajji∗  Posco Tso∗  Lin Cui†  Dimitrios Pezaros‡

∗Loughborough University, UK
†Jinan University, China
‡Glasgow University, UK
1 Movitation

2 Sync algorithms and system architecture

3 Conclusion

4 Experimental Evaluation
Middlebox challenges

IPS_{1} 

IPS_{2} 

\textbf{Overload!} 

\textbf{Policy Migration} 

\textbf{S_{1}} 
\textbf{S_{2}} 

\textbf{v_{1}} \rightarrow \textbf{S_{1}} \rightarrow \textbf{IPS_{1}} \rightarrow \textbf{IPS_{2}} \rightarrow \textbf{v_{2}} 

Posco Tso
Middlebox challenges
Motivation
Sync algorithms and system architecture
Conclusion
Experimental Evaluation

Middlebox challenges

- Overloaded link
- Policy Migration
- VM Migration
- IPS
- S
- v
- Link overloaded
We have proposed Sync: *Synergistic Policy and Virtual Machine Consolidation in Cloud Data Centers*\(^1\).

Plain lanuage: Sync migrates virtual machines and network policies at the same time.

---

We have proposed Sync: *Synergistic Policy and Virtual Machine Consolidation in Cloud Data Centers*\(^1\).

Plain language: Sync migrates virtual machines and network policies at the same time.

Does it scale in a real data centre environment?

---

1. Motivation

2. Sync algorithms and system architecture

3. Conclusion

4. Experimental Evaluation
How does $\textit{Sync}$ work? – $\textit{Sync}$ algorithms

**Get Communicating VM Groups**

The algorithm partitions all VMs into isolated groups in which VMs do not communicate with a VM outside their group. These VM groups will be the input of other algorithms.

**Policy Migration**

This algorithm focuses on migrating the policies, in other words defining again the MBs; replace them with the same type of MBs as the deployed ones.

**VM Migration**

The VM migration algorithm, for a given VM group, initialises and obtains the preference list (where no policy violation or overused server capacity) of all servers.
The topology and the controller communicate through OpenFlow to add rules to switches and via out-of-band control channel.
Source code available on GitHub
https://github.com/wajdihajji
1. Movitation

2. Sync algorithms and system architecture

3. Conclusion

4. Experimental Evaluation
1. Movitation

2. Sync algorithms and system architecture

3. Conclusion

4. Experimental Evaluation
We have run our experiments on two identical servers, each has 8 Cores/1.2Ghz CPU and 8GB Memory. Both servers have Ubuntu 14.04 is running atop.

In server A, we have installed Mininet version 2.3.0d1, OpenFlow 1.35 and Python 2.7.6.

In server B, we have installed Ryu controller 4.10.

Two servers are connected through a 1Gbps switch.
Group Formation

A graph showing the relationship between group size and the average time to identify the group, as well as the group ratio. The graph has two axes: the x-axis represents the group size, ranging from 0 to 1000, and the y-axis represents the group ratio and average time to identify the group, ranging from 0 to 0.9. The data shows an increase in average time and group ratio as the group size increases.
Sync runtime with growing number of VMs – VM groups

CDF of groups vs Runtime (s) for 2k, 6k, and 10k VMs.
Sync runtime with growing number of VMs – Policy Migration

The diagram shows the cumulative distribution function (CDF) of groups over runtime for different numbers of VMs: 2k, 6k, and 10k VMs. The x-axis represents the runtime in seconds, while the y-axis represents the CDF of groups.
Sync runtime with growing number of VMs – VM Migration

CDF of groups

Runtime (s)

2k VMs
6k VMs
10k VMs

0  100  200  300  400  500  600  700

0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1

0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1

Posco Tso

Sync
Sync runtime with growing number of flows – VM groups

CDF of groups vs. Runtime (s) for 20k, 60k, and 100k flows.

- 20k flows
- 60k flows
- 100k flows
Sync runtime with growing number of flows – Policy Migration

![Graph showing the CDF of groups vs Runtime (s) for different flow counts: 20k flows, 60k flows, and 100k flows. The graph illustrates the distribution of runtime with an increasing number of flows.]
Sync runtime with growing number of flows – VM Migration

CDF of groups

Runtime (s)

20k flows
60k flows
100k flows

0 10 20 30 40 50 60 70 80 90 100

Posco Tso
Sync runtime with growing number of MBs – VM groups

The diagram shows the cumulative distribution function (CDF) of groups over runtime (s) for different sizes of data sets (20 MBs, 50 MBs, 80 MBs). The X-axis represents the runtime in seconds, and the Y-axis represents the CDF of groups. The curves indicate how the proportion of groups completes within a given runtime, with each curve representing a different size of data set.
**Sync runtime with growing number of MBs – Policy Migration**

The graph shows the CDF (Cumulative Distribution Function) of groups over runtime (s) for different numbers of MBs: 20 MBs, 60 MBs, and 80 MBs. The runtime increases with the number of MBs, indicating a higher demand on processing time as the dataset grows.
Sync runtime with growing number of MBs – VM Migration
The number of VMs has a measurable effect on *Get communicating VM Groups* and *Policy migration* on one hand, and *VM migration* on the other hand.

The three factors have a different impact on the *Sync* algorithms, flows impacts more *Get Communicating VMs* and *Policy Migration* algorithms, while the number of VMs can significantly alter the time needed by *VM migration* algorithm.

The number of MBs has a known effect on *Get communicating VM Groups* and *Policy migration*, whereas, in *VM migration*, its impact becomes unpredictable because VM migration decision depends more on policy violation prevention strategy.

Because of its fractional use of CPU resources, *Sync* is very resource efficient and has room to scale to much bigger topologies.
Thank you! Questions?