Enhancing Security In Cloud Computing Through Virtual Machine Placement

Advisor: Prof. Ravi Sandhu and Prof. Meng Yu
By: Jin Han

Outline

- Introduction
- Threat Model and Security Assumption
- Quantify the Security Risk
- Objectives in VM Placement
- Secured Multi-Objectives Oriented virtual machine Placement algorithm (SMOOP)
- Quantify Co-residency Risk through Machine Learning
- Conclusion

Introduction

- **Virtual machine placement strategies can significantly affect the overall performance, security risks, energy cost of the entire cloud.**
- **We present a Secured Multi-Objective Optimization Virtual Machine Placement algorithm (SMOOP) to seek an overall improved solution.**
- **We present a machine learning based framework to better quantify Co-Residency Risk with Large Scale Dataset**

Threat Model and Security Assumption

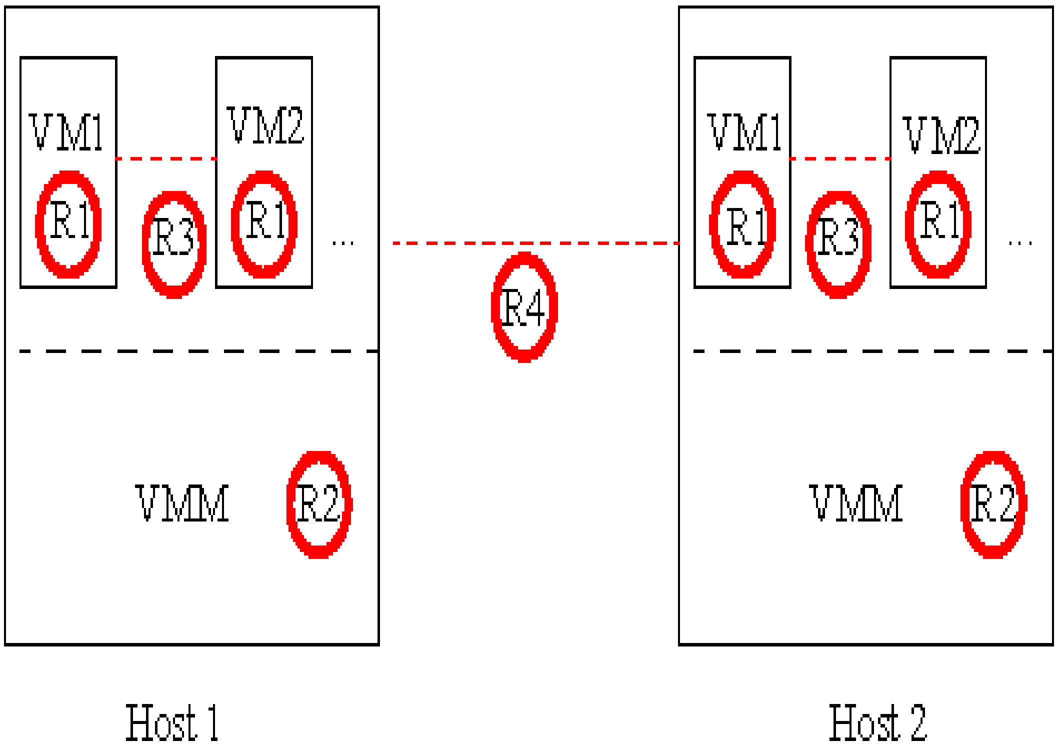
- ❖ **Co-residency and Network based Attacks were considered.**
- ❖ **Attacker are capable of utilizing vulnerabilities in both VMs and virtual machine monitors (VMMs, or hypervisor) of the clouds.**
- ❖ **Attacker need to deploy their own VMs into the same physical server to co-locate with Target.**

Security Assumptions

- The cloud service management, placement related software components, and the migration process are all secure.
- For simplicity, each migration of a VM will result in affordable cost in terms of service interruption and consume the same amount of resources.
- The cloud service provider has enough CPU, network bandwidth, and other resources to perform arbitrary migration of VMs.
- The cloud service provider has sufficient resources as the reward, e.g., extra memory or CPUs, to motivate VM migration.

Above assumptions ensure that change of VM placement is both acceptable and affordable for cloud provider and clients.

Security Assessment in the Cloud



- R1: VM Risk
- R2: VMM/Hypervisor Risk
- R3: Co-Residency Risk
- R4: Network Risk

Quantify the Security Risk

- Quantify the values of each types of security risks, and calculate the overall security risk of the entire cloud.

- R1(VM Risk) is not affected by a specific Placement.
- R2(VMM/Hypervisor Risk), R3(Co-residency Risk) and R4(Network Risk) are affected by a specific Placement.

R1 - VM Risk

- **R1, VM risk, is based on CVSS (Common Vulnerability Scoring System).**
- **Assumption : the security level of a VM is determined by the worst vulnerability of that VM.**
- **For a VMi, R1 = f_1 (<S1, S2,…Sj>) = Max(CVSS Scores of VMi) / 10, which is in (0,1).**

R2 - Hypervisor Risk

- **R2, Hypervisor Risk, is determined by two factors: its own vulnerability and the VMs running on it.**
- **$Risk_{hypervisor}$ = Max(CVSS Score) / 10, which is in (0,1).**
- **In our paper, we only considered the VM with the highest risk.**
- **For a Host i, R2 = f_2 ($Risk_{hypervisor}$, {$R1_i$}) = $Risk_{hypervisor}$ * (1 + max(R1 in Host i)) / 2 , which is in (0, 1) too.**

R3 - Co-residency Risk

- **A malicious VM can deploy co-resident attack to its target if they are collocated on the same Host.**
- **In SMOOP work, For a VM i on the Host K, its co-residency risk R3 is calculated with f_3 ({R1$_i$}) as:**

$$R_3^i = 1 - \prod_{j=1}^{N} (1 - R_1^j p_{jK})$$

where Pj = 1 if VM j (other then i) is placed on the Host K.

- **In our subsequent work, we proposed a machine learning based framework to quantify the Risk R3.**

R4 - Network Risk

- **We assume attacker could find a path to attack the target through Network Connection.**
- **In this paper, we only considered the risk caused by only direct network connections for simplicity.**
- **For a VM i, its network risk R4 is calculated as:**

$$R_4^i = 1 - \prod_{j=1}^{N} (1 - R_1^j)$$

where VM j is sending packet to VM i directly and they are not at the same Host.

Risk Level of a VM

- **After R1 to R4 are all quantified, we could have the risk level of a VM i with *f*({R<j>}) as:**

  $$R_i = 1 - (1-R1_i)(1-R2_i)(1-R3_i)(1-R4_i)$$

- **A VM is safer with lower risk level value.**
- **Question: How is the risk level of cloud determined by the set of R value within a specific placement?**

Secured Multi-Objectives Oriented Virtual Machine Placement

- ❖ **In our experiment, we setup three objectives to optimize: Security Risk(SR), Resource Wastage(RW) and Network Traffic(NT).**
- ❖ **Users could add more objectives, based on their preferences.**

Security Risk

- **Within a specific placement, the set of R value of VM is confirmed.**
- **The security level of cloud is determined by the distribution of the set of R value.**
- **We used the median value of the set of R value as the risk level of the cloud within a specific placement.**
- **The security risk of cloud f_{SR} = median(R) in our work so far.**

Resource Wastage

- **In this paper, we consider the wastage of multiple resources, including CPU, memory, and disk.**
- **Within a specific placement, the wastage percentage of CPU, memory, and disk in Host K could be calculated as W_{CPU}, W_{MEM}, W_{Disk}.**
- **The resource wastage $f_{WS} = \sum \max(W_{CPU}, W_{MEM}, W_{Disk})$. Weight could used here per user's preferences.**
- **Capacity constraints in each host are applied.**

Network Traffic

- Two VMs with high amount of data exchanging should be placed into same Host, to reduce the network traffic.
- The network traffic from VM i to VM j: T_{ij} = Packet_N_{ij} / t
- The total network traffic in cloud is

  $f_{NT} = \sum_{j=1}^{N} \sum_{i=1}^{N} (T_{ij} * p_{ij})$, where $p_{ij} = 0$ if VM i, j are in the same Host, otherwise it is 1.

SMOOP Design

- **Challenge: It is impractical to directly find the optimal solution minimizing all objectives. At the same time, the security metrics can only be evaluated after the placement is specified.**

- **SMOOP is proposed to search improved solutions on specific target objective or balancing on multiple objectives.**

- **New crossover and mutation operation are designed with security related strategies.**

Prioritize the Objectives

- **Our algorithm tries to provide a improved solution which can be as good as possible in Service Provider's perference.**

- **To enable users to prioritize the objectives according to their business preference, we can add weight factors into the fitness function as:**

  $\boldsymbol{f} = w1^* f_{SR} + w2^* f_{RW} + w3^* f_{NT}$, **where** $\sum wi = 1$.

Algorithm 1 SMOOP

```
Ensure: Canditate = init() by Strategies
  for G = 1 → N_G do
     for i = 1 → N_E do
        Elite[i] = Elite_choosing(Canditate)
     end for
     for j = 1 → N_C do
        (X, Y) = Random_select(Canditate)
        Off_C[j] = Crossover(X, Y)
     end for
     for k = 1 → N_M do
        X = Random_select(Canditate)
        Off_M[k] = Mutation(X)
     end for
     Temp = fitness_sorting(Elite, Off_C, OFF_M)
     for i = 1 → N_G do
        Candidate[i] = temp[i]
     end for
  end for
```

Security related Strategies

- **Placement strategy I: Put a VM into a physical machine which has network connections with it.**
- **Placement strategy II: high risk VMs should be put into the isolated zones.**
- **Placement strategy III: low risk VM without any connection with VMs in isolated zones should be put into low risk Host.**
- **Placement strategy IV: marked lowest and highest hypervisor risk physical machines should have a higher probability to be kept during crossover operation.**
- **Strategy V: If a VM on one physical machine has connection with a VM on a different physical machine, we should migrate them together.**

Strategies Implementation

- **When VM is deployed, <VM, PM> pair will be generated. Each pair would have a associated “strategy fit set”.**
- **The set will be tested to determine whether it satisfies each strategy.**
- **Higher priority strategy come first.**
- **Multiple strategies should be satisfied as more as possible.**
- **Priority list of strategy will keep evolving in the runtime environment and new strategy will be added too.**

Algorithm 2 Crossover(X, Y)

```
Temp = Blank_Placement_Object
Rank_A = Rank(X)
Rank_B = Rank(Y)
for i = 1 → P_N do
    if Rank_A[i] > Preset_value then
        temp ← Rank_A[i]
    end if
    if Rank_B[i] > Preset_value then
        temp ← Rank_B[i]
    end if
end for
Remove_duplicate_VM(temp)
Ran = Gen_Random_list(VM)
for i = 1 → V_N do
    if Ran[i] is not in temp then
        temp ← Ran[i] by Strategies
    end if
end for
Return Temp
```

Algorithm 3 Mutation(X)

```
Temp = Blank_Placement_Object
temp ← X
for i = 1 → Preset_Maximum_number do
    temp ← Switch(temp) by Strategy of Switching VM
end for
Return Temp
```


- **No guarantee the child generated would be better than parents.**

Incremental Task Handling

- When a new VM is deployed or re-activated, the new placement should be generated based on the current one to achieve the multi-objective optimization, while keeping the low migration cost in mind.

- We improved our SMOOP algorithm to better handle incremental deployment task by collaborating with latest Virtual Machine Allocation Policies:
    1. Co-Location Resistant (CLR)
    2. Previous-selected-Server-First (PSSF)

Modified SMOOP

To better handle the incremental task, we improved the SMOOP by removing crossover operation and mutation operation is modified to adapt the latest VM allocation policies.

Algorithm 5.1 SMOOP-phase2

Ensure: Canditate $\leftarrow$ Insert(Current, New_VM) with Pre-set VM allocation Policy

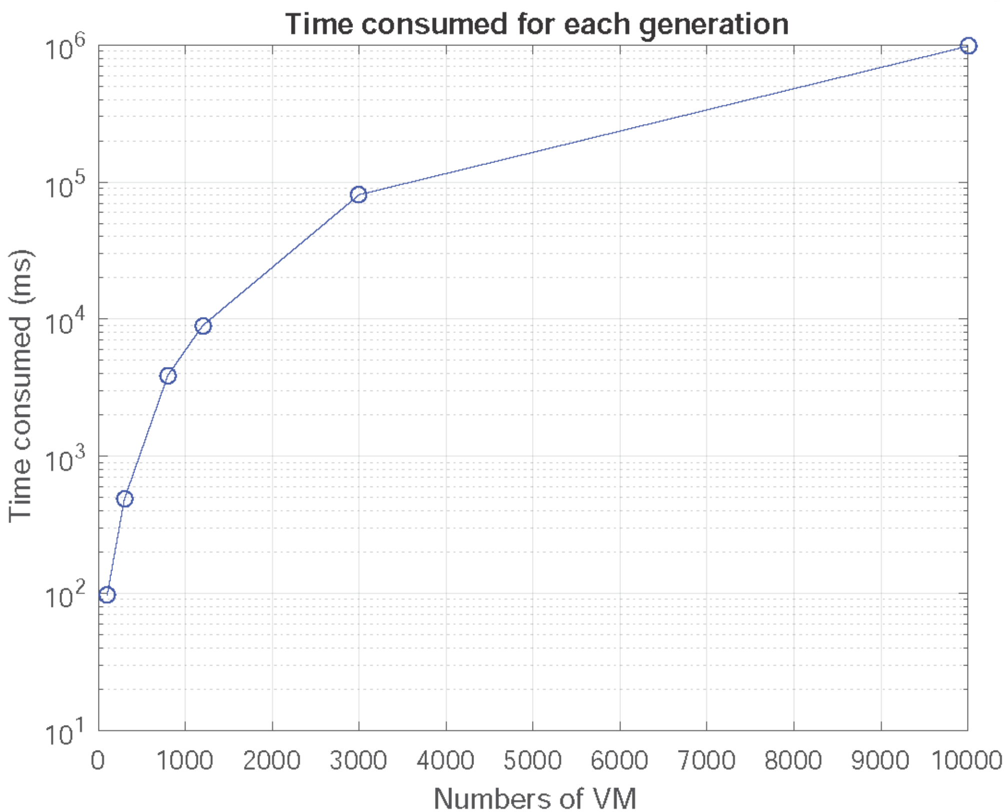
```
for G = 1 → N_G do
    for i = 1 → N_E do
        Elite[i] = Elite_choosing(Canditate)
    end for
    for k = 1 → N_M do
        X = Random_select(Canditate)
        Off_M[k] = Mutation(X)
    end for
    Temp = fitness_sorting(Elite, Off_C, OFF_M)
    for i = 1 → N_G do
        Candidate[i] = temp[i]
    end for
end for
```

Evaluation

- **Computing Complexity**
- **Risk Reduction**
- **Effectiveness of Multi-objective Optimization**
- **Comparison with Random-FFD Algorithm**

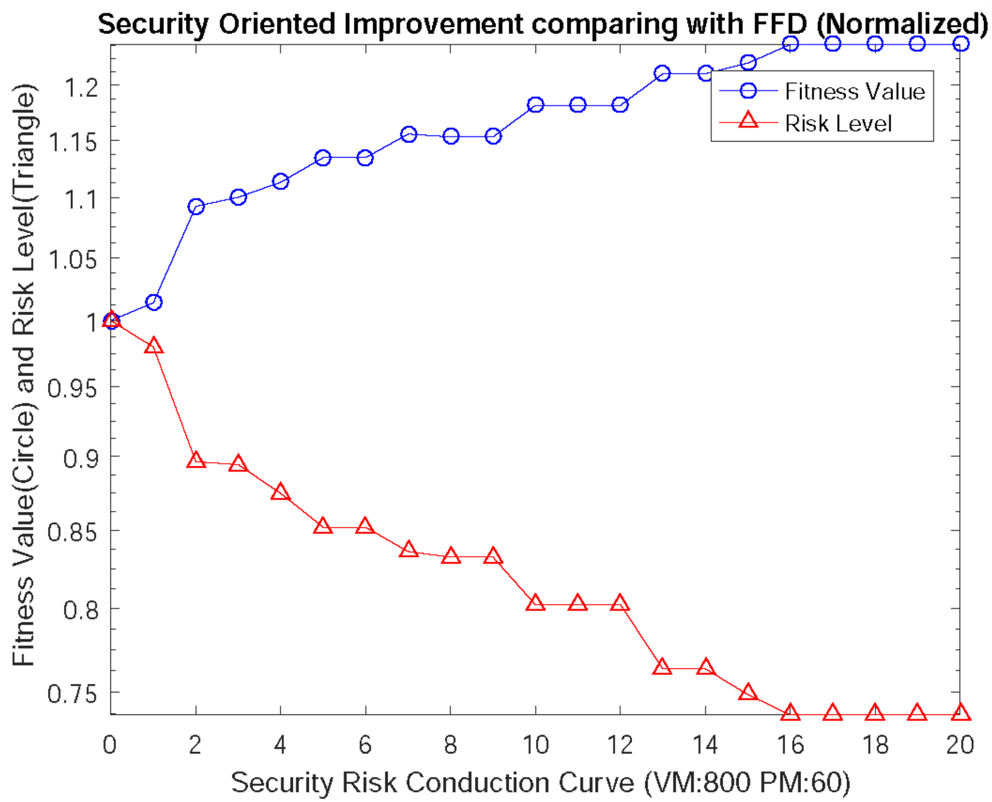
Computing Complexity

- **Timing is key factor here. Our algorithm should provide a solution within acceptable time period.**

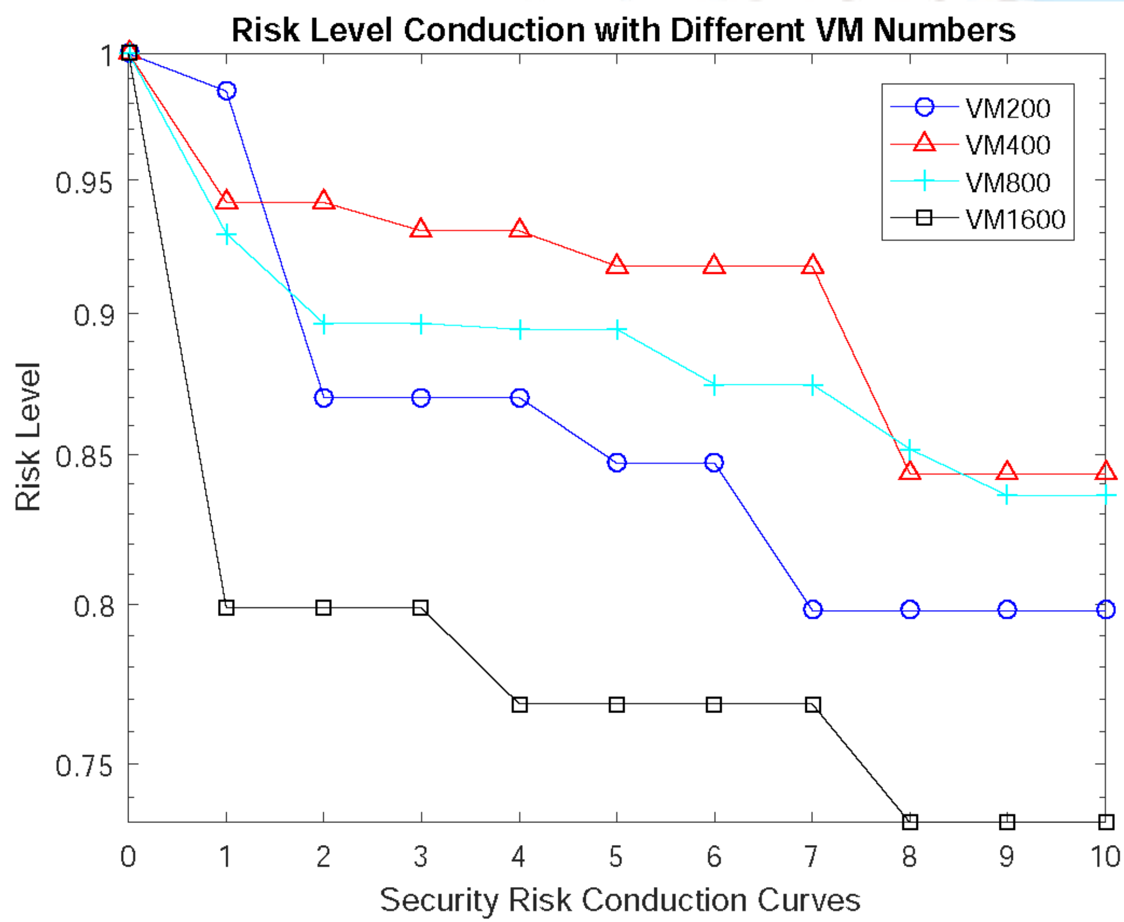


Risk Conduction

- At the beginning of each simulation, we always generate 100 placement with the random-FFD algorithm and use the lowest risk level as the baseline reference.

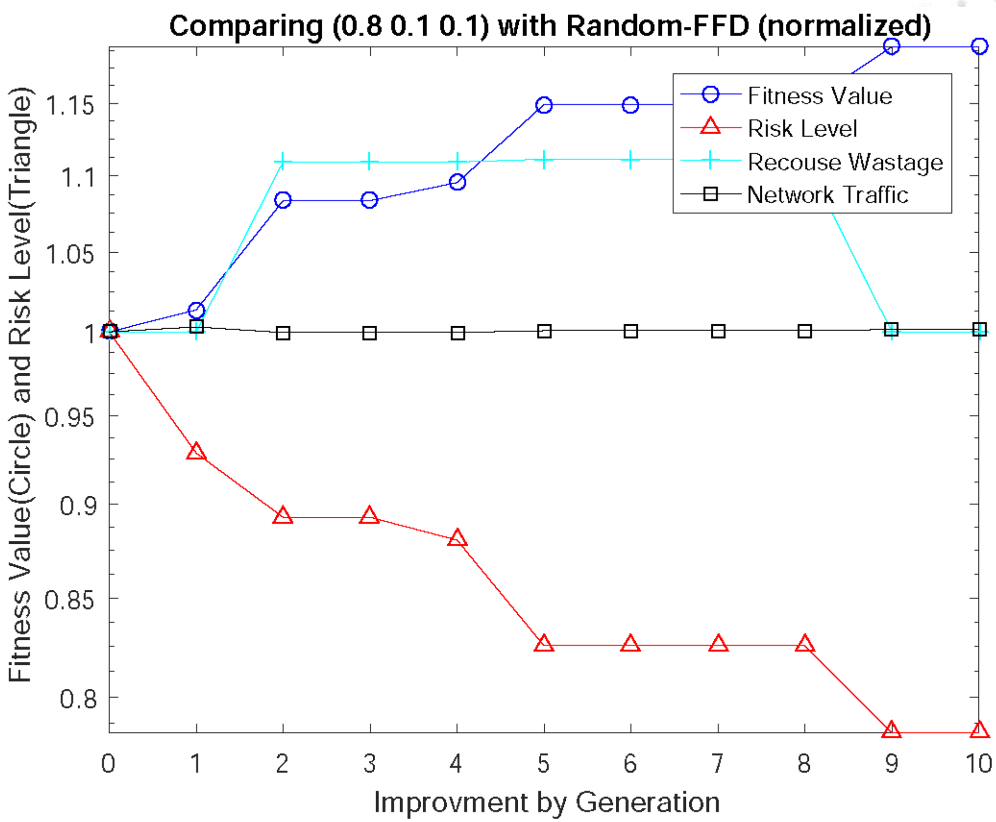


Risk Conduction



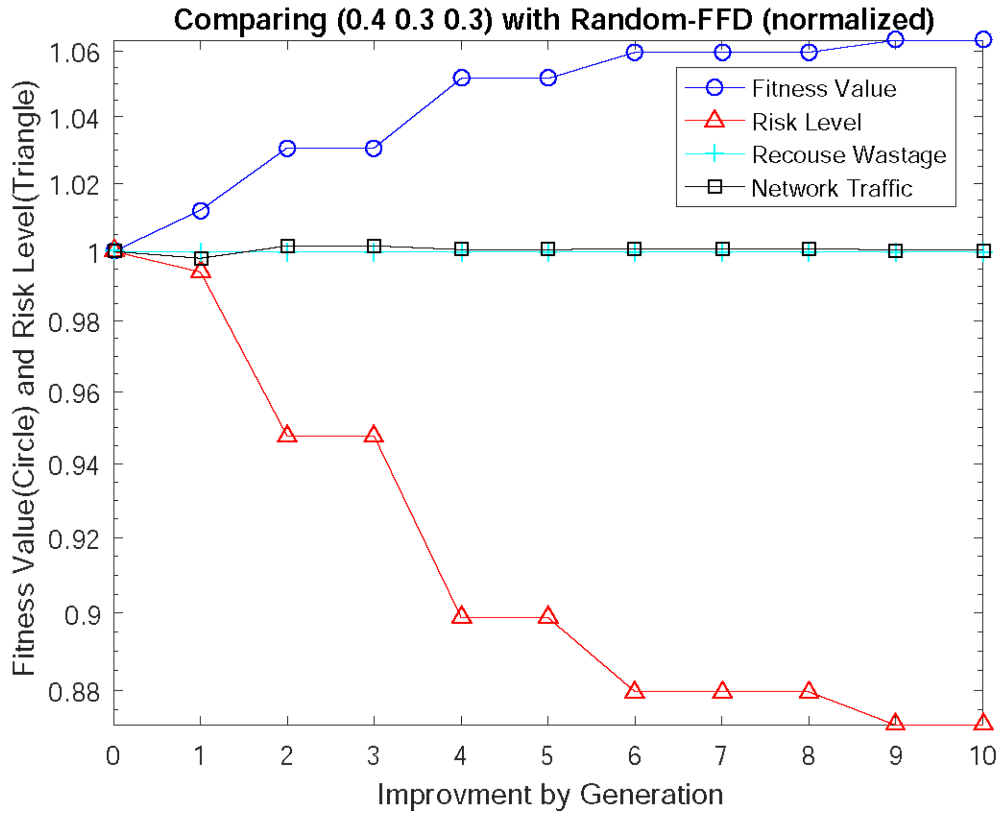
Effectiveness of Multi-objective Optimization

❖ **Experimental results with weight setting (0.8, 0.1, 0.1) in an environment of 800 VMs and 60 physical machines.**



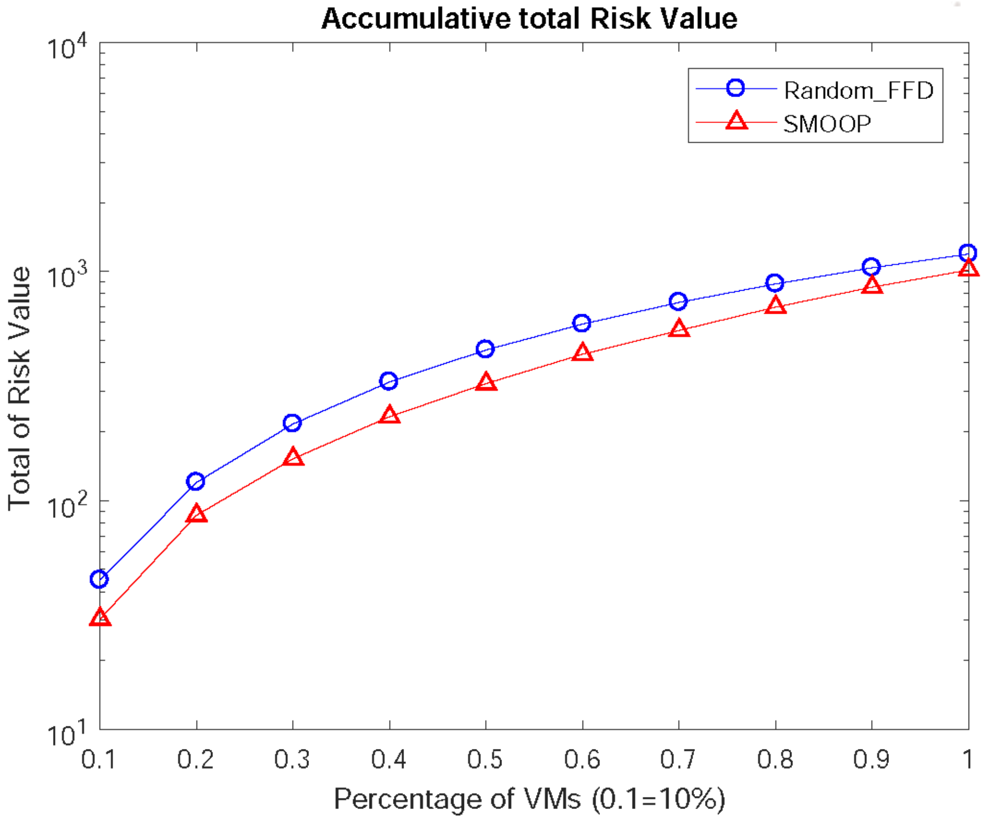
Effectiveness of Multi-objective Optimization

❖ **Experimental results with weight setting (0.4, 0.3, 0.3) in an environment of 3000 VMs and 200 physical machines.**



Comparison with Random-FFD Algorithm

- **By checking accumulative total risk value of whole VM set, SMOOP could more efficiently reduce the risk level of whole cloud.**



Quantify Co-Residency Risk through Machine Learning

- **Profile normal service subscriber's behavior pattern by large scale Microsoft Azure Dataset.**

- **Co-resident Risk Rate of a VM is mainly determined by the owner**

- **Need a dynamic adapted framework to better quantify Co-Resident Risk Rate for SMOOP**

Additional Assumptions

- ❖ **Service Provider has no knowledge about attacker's appearance**
- ❖ **Service Provider has no or limited knowledge about attacker's behavior pattern**
- ❖ **Attacker's behavior pattern could be evolved and different**
- ❖ **Service Provider could verify limited amount of normal service subscriber (mark as legal)**
- ❖ **Attacker has no way to compromise the data collected by service provider for profiling purpose**
- ❖ **Attacker can't compromise the detection system implemented by service provider**

Above additional assumptions ensure our proposed quantify system act practical in real world.

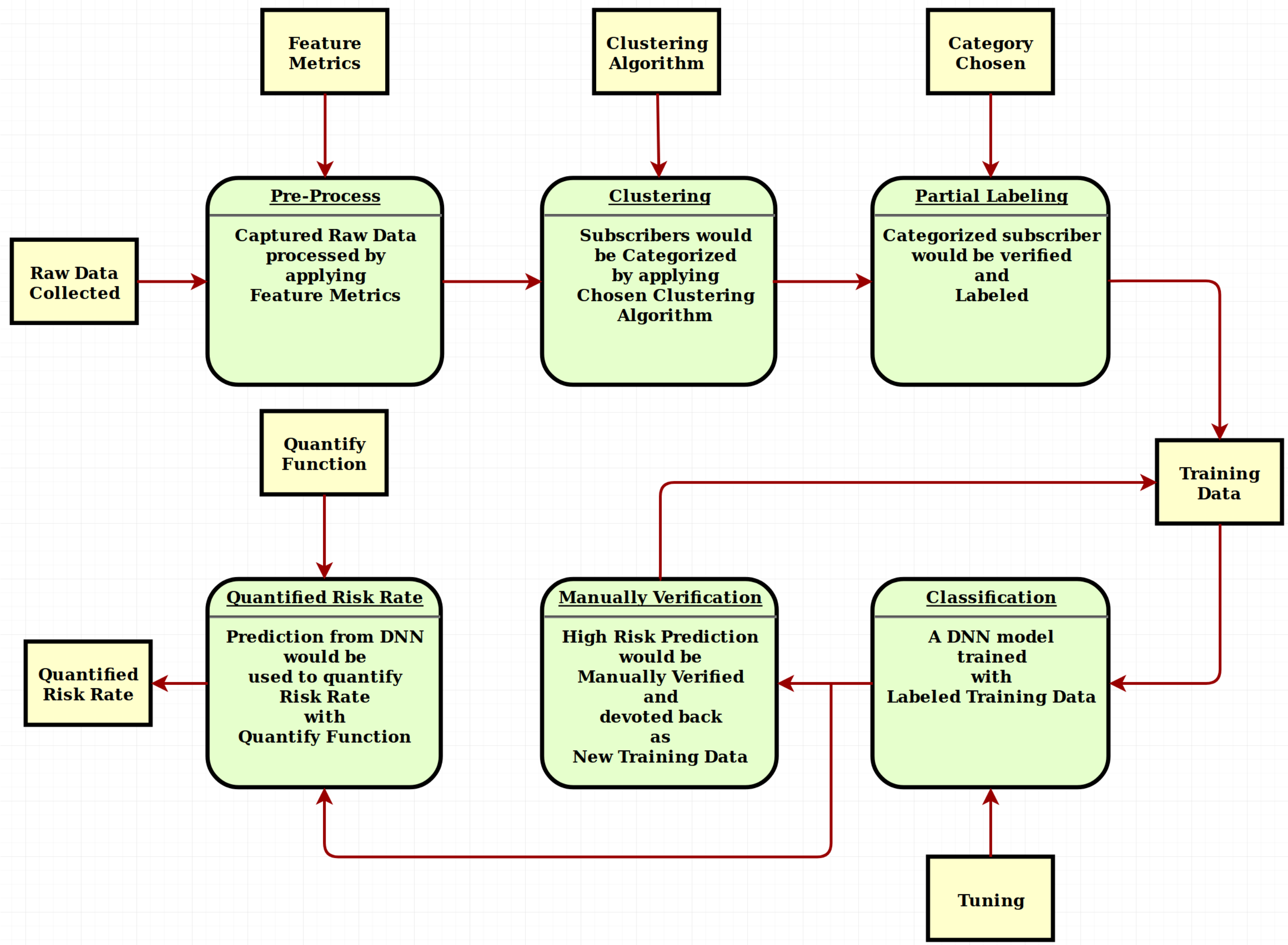
Co-Resident Attacker's Potential Behavior Pattern

Attacker must achieve the co-residency with their target first

1. **A number of VMs will be started simultaneously or independently**
2. **Check if any these started VMs were deployed into the same PM with Target**
3. **Stop failed VMs to save the cost (Optional)**

Above steps could be repeated several times until co-residency is achieved.

Overview of Framework



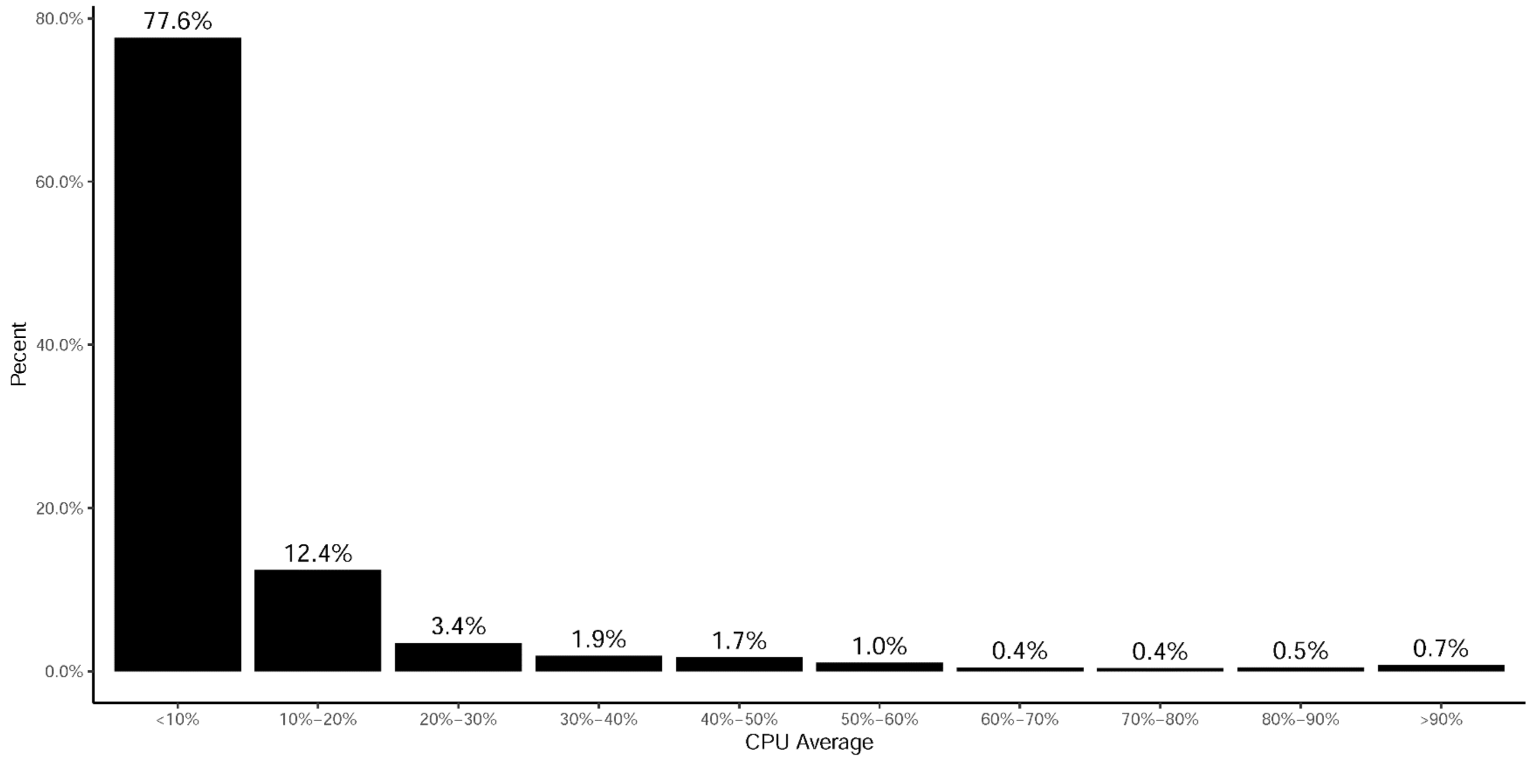
Clustering Subscriber by Feature Metrics

Proposed a six dimension Feature Metrics to profile service subscriber:

1. **N – Total amount of VMs deployed by a subscriber**
2. **T – Average interval time between starting two VMs of a subscriber**
3. **M – Median memory size of VMs of a subscriber**
4. **A – Overall active rate of a subscriber**
5. **W – Average amount of active VM in each time stamp of a subscriber**
6. **I – Median of average CPU utilization rate among all VMs in each time stamp of a subscriber**

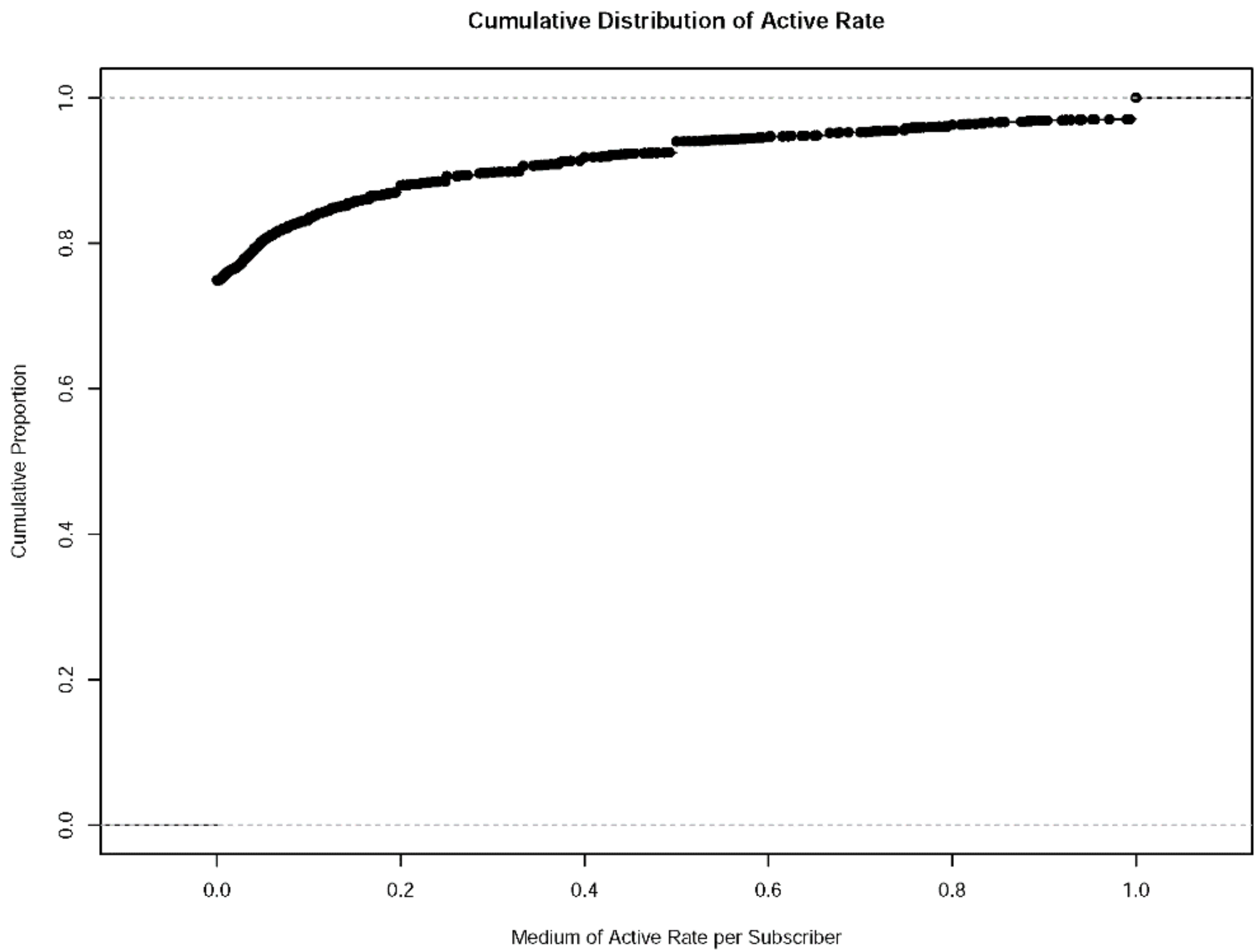
Insight of Azure Dataset

❖ **Below diagram is the overall active rate of all VMs. It is clear to see that around 90% VMs is under 15% Average CPU utilization**



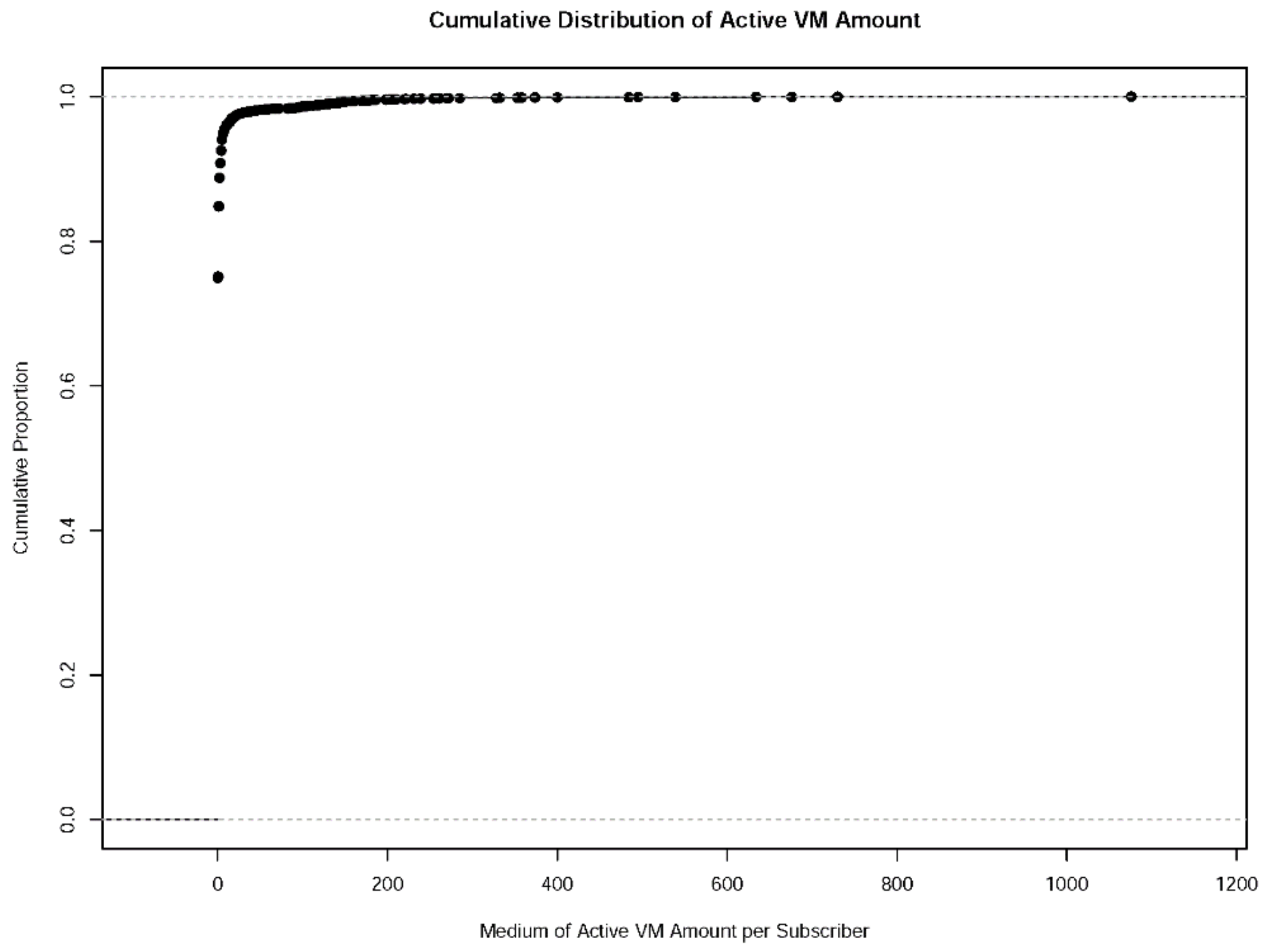
Median of Active Rate

Over 80% subscriber remained less than 10% active rate in each time stamp.



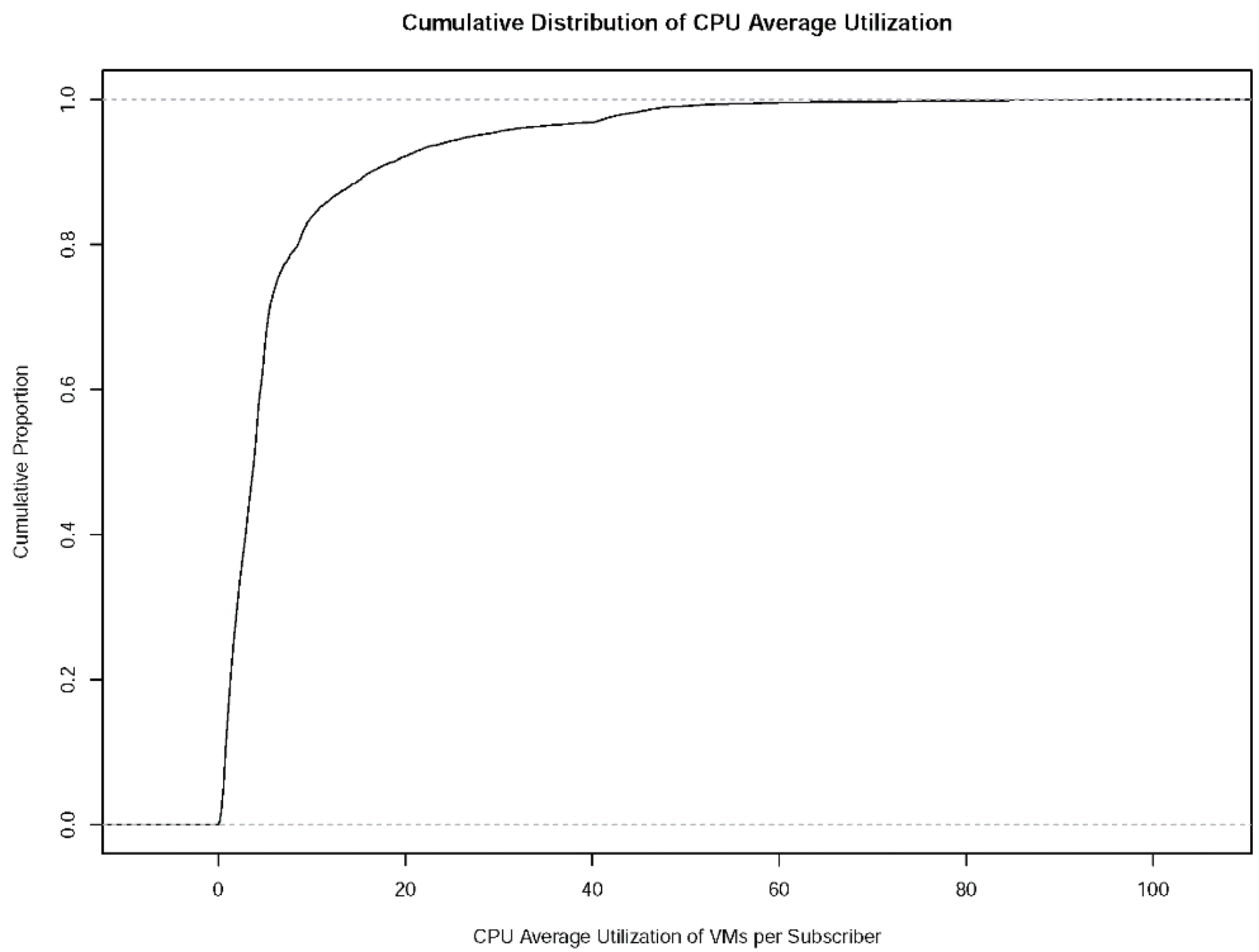
Average Active VM Amount

Over 95% subscriber only maintained equal or less than one active VM in each Sampling time spot. Combined with Median of Active rate below, we could filter out extreme active subscriber



Median of Avg. CPU Util. Rate

Then we obtained a brief idea about the average CPU util. rate among subscribers in each Sampling Time Spot. We could conclude that over 85% subscriber should be clustered into one category.



Clustering Algorithm

In the end, we used DBSCAN to cluster the service subscriber by comparing with other clustering algorithm, it handle noises better

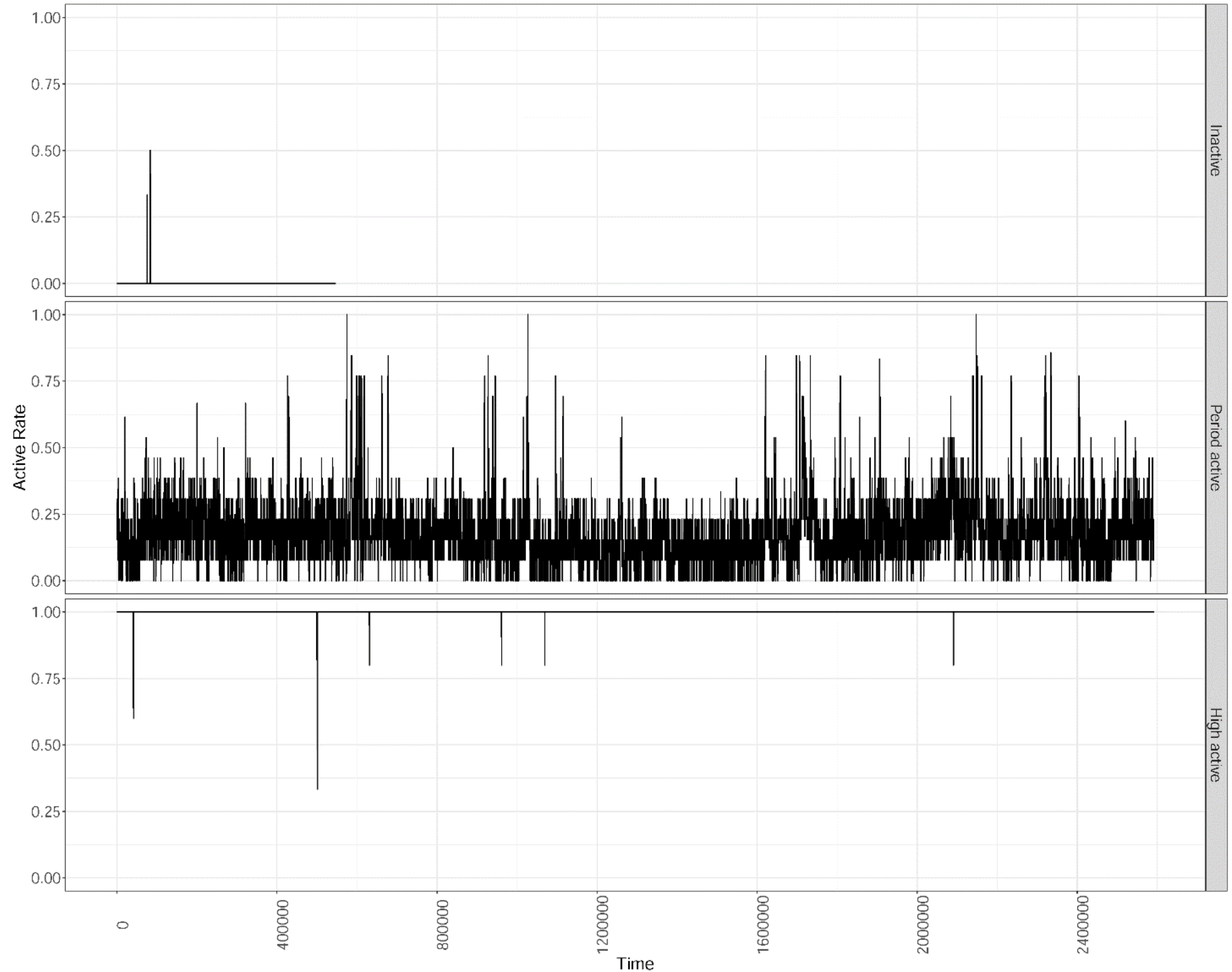
1. **Using DBSCAN initially cluster out categories of subscribers**
2. **Manually check every categories**
3. **Partially label them into three major type: Inactive(Normal), Period Active, Extreme Active**
4. **Labeled data will be used as the input for the training of classification component**

Three type of Active Level

We finished the detail active rate curve for every subscriber

Three typical type of curves are demonstrated on the right diagram.

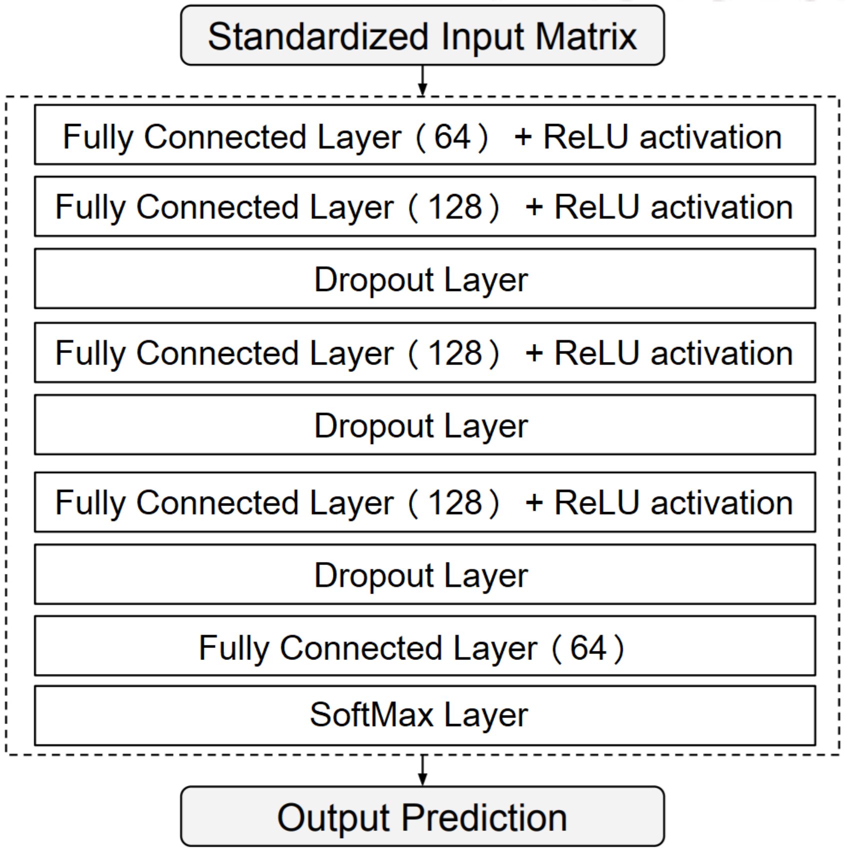
Over 11000+ subscriber have the similar curve with top category.



Classification Component

- Use collected history to classify new seen data
- New classified data could be optional verified and used as new training history data
- In our implementation, a Deep Neural Network finished the classification task
- Output probability will be used to calculate co-resident risk rate

DNN Architecture



Quantify the Co-resident Risk

Since the we labeled the DBSCAN output into three categories, we setup the output of our DNN in this way:

- **For inferenced subscriber, three predicted probability number will be output to represent three major category**

- **The number represented the normal category will be used to calculate the co-resident risk. In other word, we believe it represent the derivation rate from normal behavior pattern**

- **In our implementation, R3 = 1 – P(normal subscriber)**

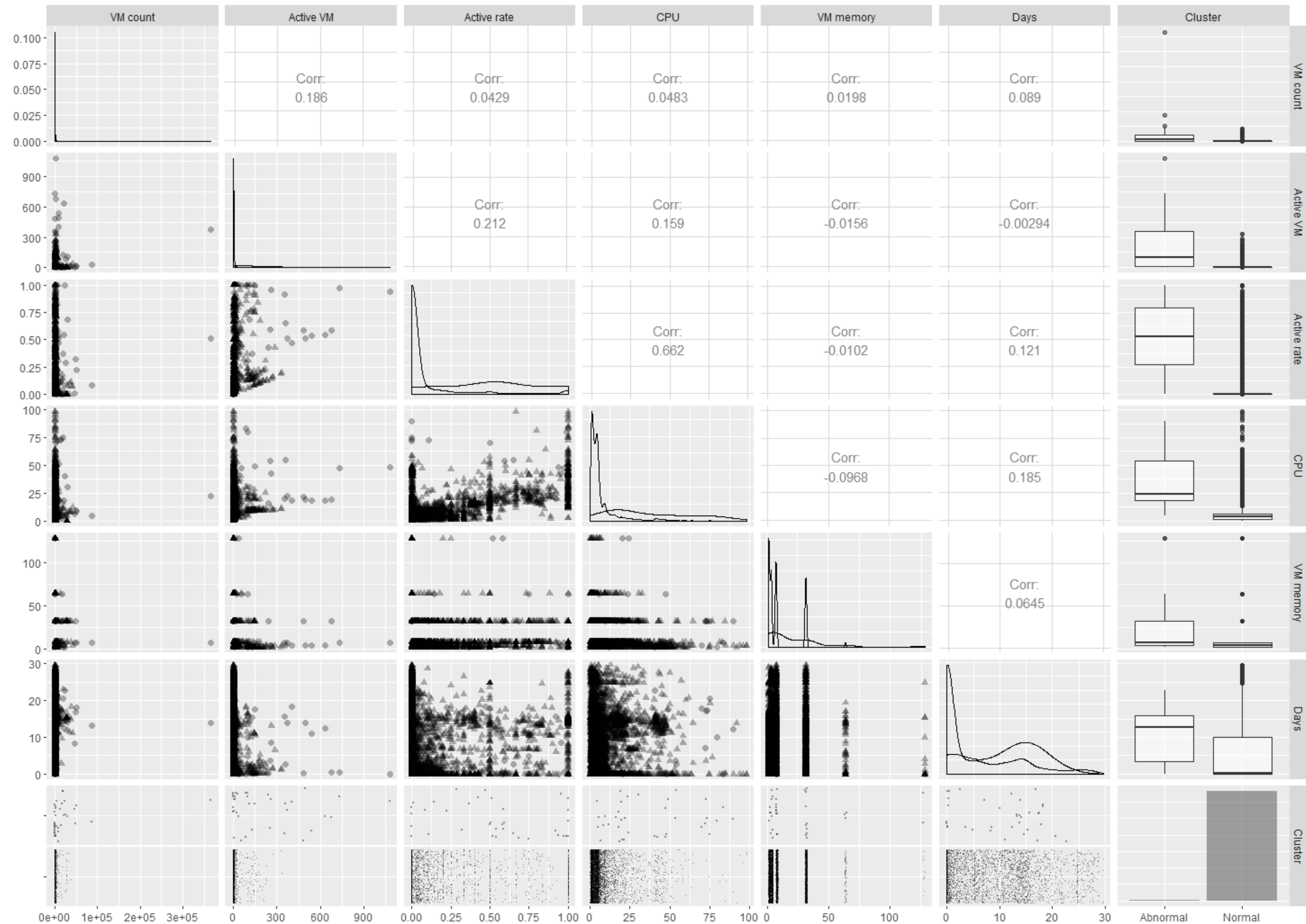
Evaluation

- ❖ **Clustering Subscriber**
- ❖ **Classification Evaluation**

Clustering of Subscribers

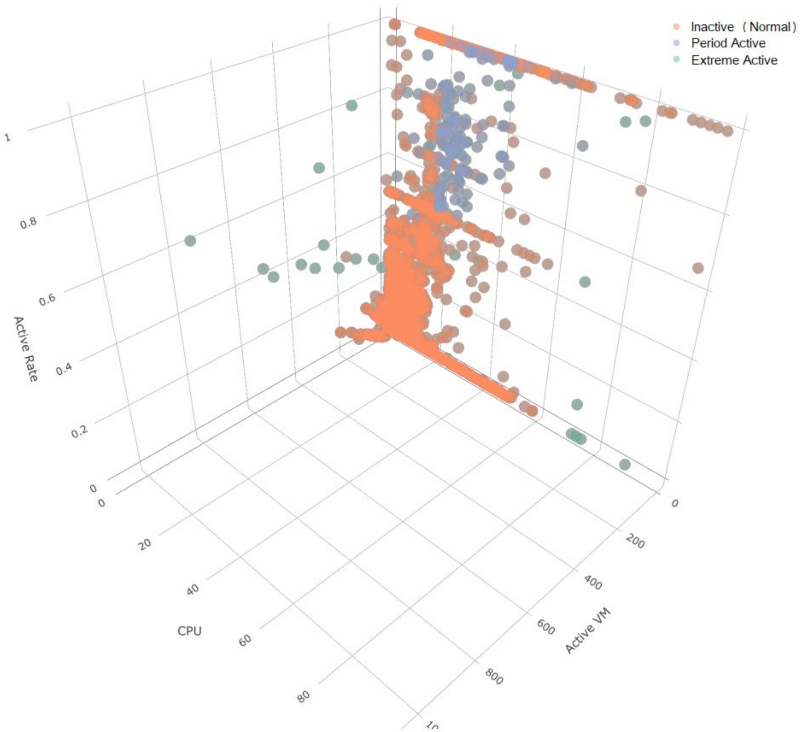
Clustering paring diagram among normal subscriber and others.

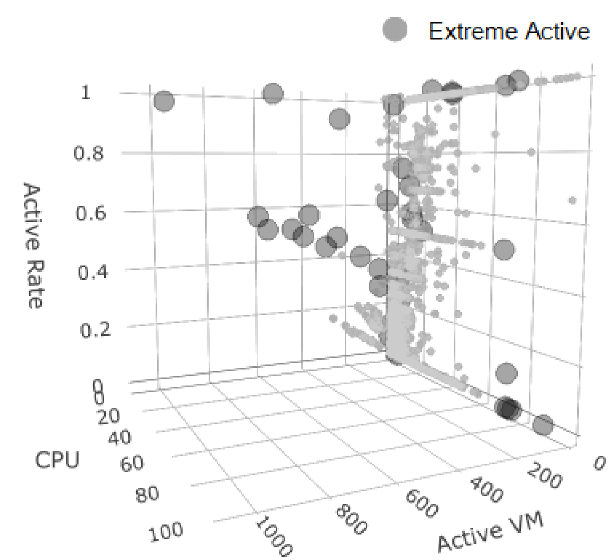
Clustering finished by DBSCAN algorithm.



Clustering Result

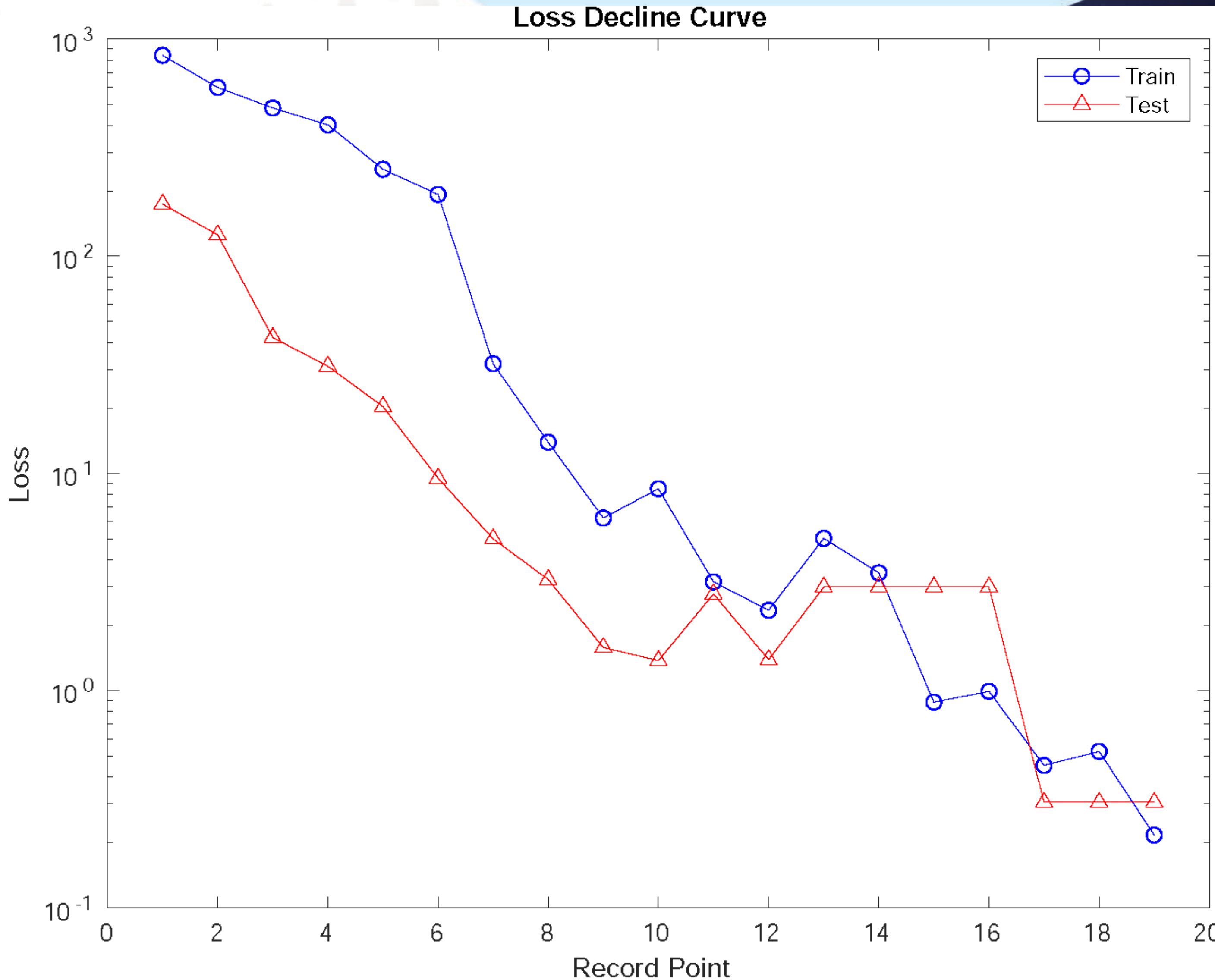
Based on DBSCAN clustering output, we labeled them into three major category: Inactive(Normal), Period Active, Extreme Active. These labeled data would be used for our training.





Training of DNN

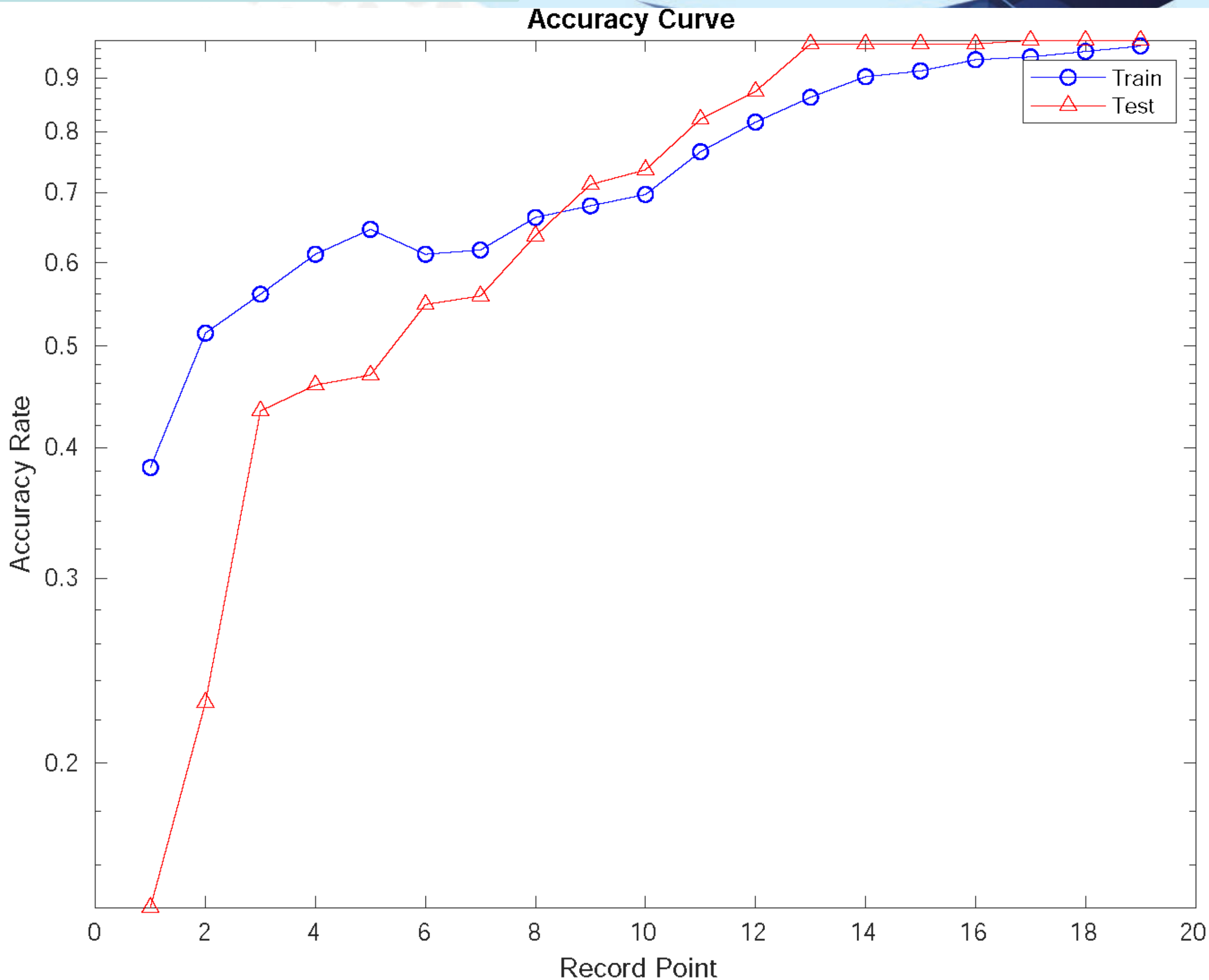
Decline curve of Loss Function with handling extreme imbalanced training data and conquered the over-fit problem



Accuracy Rate for Test Set

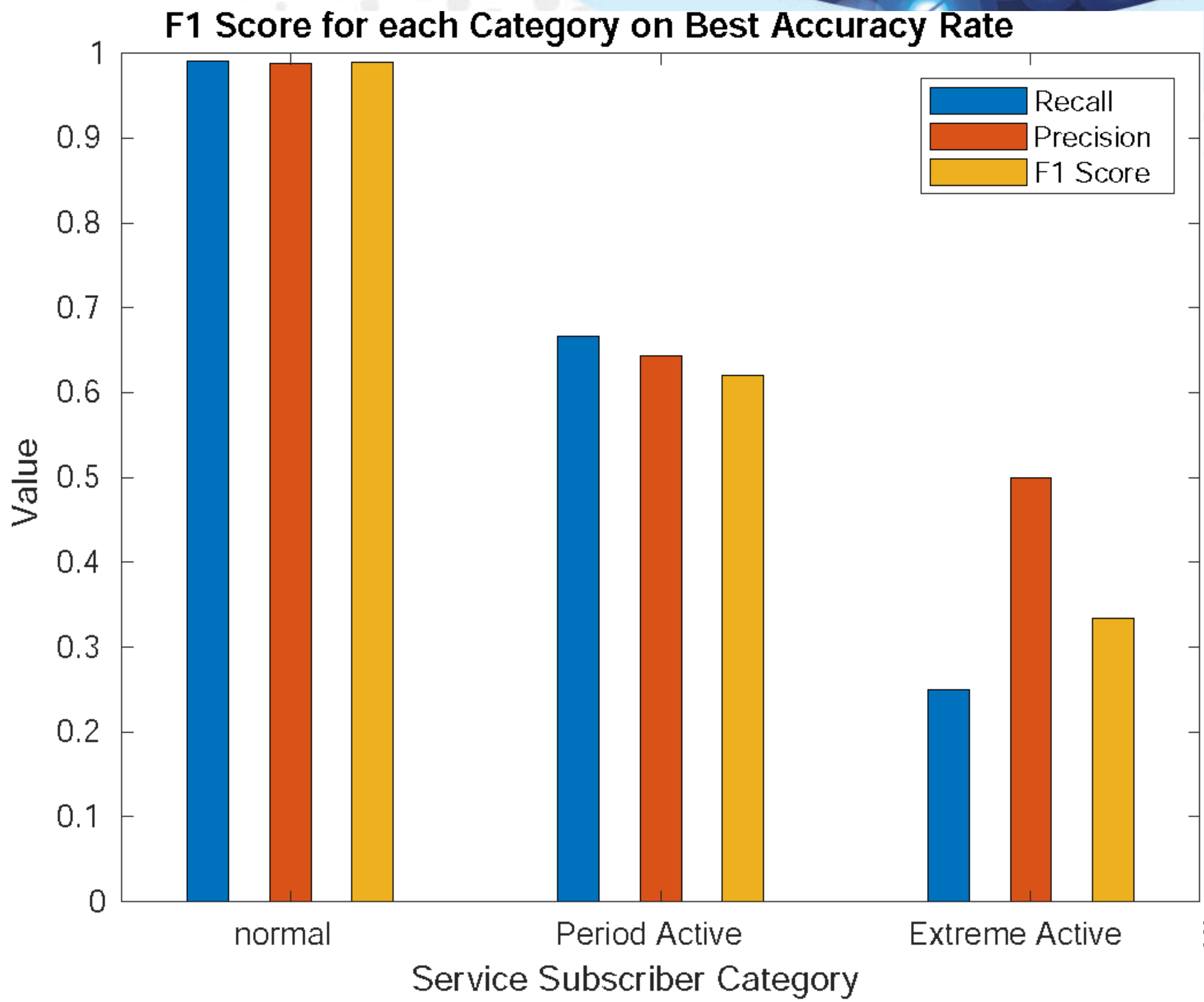
Accuracy rate for Test set demonstrated our training is quite robust with new seen data.

The test set is 15% reserved and each test is finished by randomly drafted 50% from test set



F Measuring Matrix

F measuring Matrix result demonstrates that there is still a lot improve space for identify the rare event, in our case, the attacker’s appearance.



Conclusion

- We conduct security assessment of the cloud and Security Risk of cloud was quantified.
- We present a Secured Multi-Objective Optimization Virtual Machine Placement algorithm (SMOOP) to seek an overall improved solution.
- Latest VM allocation polices were integrated to better handle incremental deploy situation.
- Based on the large scale Microsoft Azure dataset, we finished the task to profile the behavior pattern of normal service subscriber within our proposed feature metrics.
- A dynamic adapted framework was proposed to more accurately quantify the co-residency risk rate.

Planned Work

- Too much detail information of service subscriber were lost during the data abstract procedure. We need to build a Deep Conventional Neural Network as a sub-module to handle the detail feature diagram directly.

- General adaption of our framework should be tested with new large scale dataset.

- Abnormal event could be detected by profiling subscriber's behavior pattern.

Publication

- Jin Han, Wanyu Zang, Songqing Chen, and Meng Yu. Reducing security risks of clouds through virtual machine placement. In Giovanni Livraga and Sencun Zhu, editors, Data and Applications Security and Privacy XXXI (Dbsec), pages 275–292, Cham, 2017. Springer International Publishing.
- Jin Han, Wanyu Zang, Li Liu, Songqing Chen, and Meng Yu. Risk-aware multi-objectives optimized virtual machine placement in the cloud. Journal of Computer Security, Vol 26, Issue 5, pages 707-730, Published: Aug 7,2018. IOS Press Publishing.

Appendix

- Incremental implantation of placement was finished with integrating with latest VM allocation policies.
- User constraints were discussed within incremental implantation
- Fuzz sorting is discussed to replace single fitness function
- AI engine was used to better quantify co-residency risk in our framework (From Proposal)
- Possible better equation for Ri values is discussed
- The security evaluation is hard to do before the placement was confirmed

Thank You!

- ✓ Question?