Minimizing Service Interruption during Live Multiple VM Migration

Prasad P. Netalkar
Department of Electrical and Computer Engineering
Rutgers, State University of New Jersey
pnetalka@winlab.rutgers.edu

***Abstract*—Cloud computing is a platform allowing users to request for various resources such as compute, storage on demand basis and are basically fulfilled by large cloud providers such as Amazon and Microsoft.The challenge to efficiently migrate these virtual resources both within and between the cloud has gained lot of attention in recent years. VM (virtual machine) migration could be triggered due to the change in resource demand or decrease in user perceived QoS (Quality of Service) and migrating it to a different location will help alleviate the issue. With the advent of 5G, these migrations will be so common due to the large deployment of edge clouds and hence triggering frequent migration as the user moves. If multiple VMs must be migrated simultaneously, providing a seamless migration with low downtime is very challenging. In this work, a geometric programming model is formulated where the main moto is to allocate optimal bit rates for the migrating VMs such that total time for migration is minimized.The results provide further insights in designing an efficient multi VM seamless migration scheme with low downtime.**

***Index Terms*—VM, dirty rate, pre-copy, convex**

I. Introduction

Next generation 5G networks are expected to support large numbers of users and providing resources to these users have become a key topic of interest these days. Network operators and researchers are demonstrating the feasibility of mmWave (28GHz initially) which can provide a very high bandwidth of around 1Gbps for radio access network. But the backhaul or fibre network still operating with some centralized architecture has become bottleneck these days. Cloud computing is gaining lot of attention due to its distributed architecture allowing wide varieties of applications to be hosted on its platform and the capability to aggregate resources and providing it to users on demand. Cloud is nothing but a data centre which can be considered as having infinite storage and compute resources. These resources can be accessed from anywhere with the power of internet. It's always better to place it closer to users for lower latency and higher throughput. Cloud providers like AWS and Microsoft have multiple datacentre around the world to support its customers. In cloud, resource virtualization is a vital requirement. Virtualization allows for efficient resource utilization as each cloud must support millions of users and allocating a dedicated resource to everyone is practically impossible. VMs can be easily cloned, instantiated, migrated or rolled back to previous state without hardware intervention making it a good candidate to deploy end user services. Virtual machine resides on top of hypervisor and is completely transparent to physical hardware. A hypervisor is a software or a firmware that helps in creating and running VMs. It also allows one host machine or server to support multiple guest VMs by efficiently sharing storage and compute resources between guest VMs.

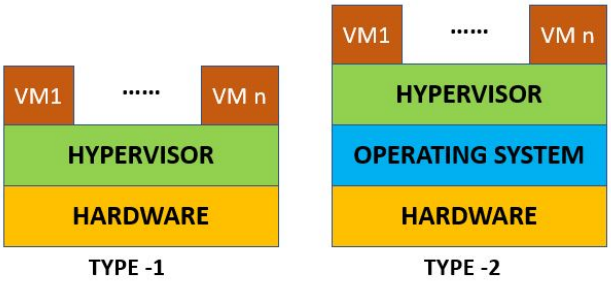


Fig. 1. Types of Hypervisor.

There are mainly two categories of hypervisor. As shown in Fig. 1, type 1 hypervisor runs directly on top of hardware and controls the guest VMs whereas type 2 runs on top of host OS and hypervisor separates guest OS from host OS. It's with the help of hypervisor VM mobility is possible. Live VM migration is a key feature which allows migrating VM with least service disruption to end user QoS. Migration mainly occurs due to the change in user demand or location. Migrating VM closer to user or to a better place which can support higher demands will improve QoS and might as well help in increasing revenue of cloud providers. For live migration of VM from one server to another, we must maintain network, storage and memory states such that user experiences minimum downtime. A well-known algorithm called pre-copy is used to perform live migration. It is an iterative push-based migration algorithm which consist of 3 phases. At first, (called as round 0) all the static memory pages allocated to that VM is transferred. In the next phase all the dirty memory pages i.e pages modified during previous copy phases are transferred iteratively until certain timeout interval or the dirty memory rate is less than the available transfer rate. Finally, we have the stop and copy phase where the original VM is suspended and all the remaining dirty memory contents are transferred to the destinations [1]. Fig. 2 and Fig. 3 provide a system overview of migration of a single and multiple VM respectively. At the

destination we integrate all the above memory contents and the VM is powered on with a fixed resume time based on hardware configuration. The problem of transferring VM to a different

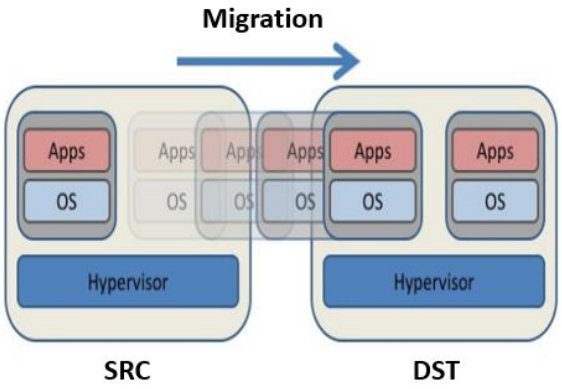


Fig. 2. VM Migration.

location such that the user experiences minimum downtime is a key issue that must be solved. Having multiple VM will further make the problem further interesting as we need to allocate optimal bit rate between multiple VMs. Migration of VMs inherently has many challenges few of which are bandwidth between source and destination locations, VM memory size, dirty memory rate etc. A convex optimization problem can be formulated where the main goal is to allocate optimal bit rate between various VMs and also on deciding the optimal number of rounds required for the above phases such that total time for migration is minimized.

II. Mathematical Formulation

A. Problem

As discussed above in data centre networks we have a problem of VM migration. Having multiple VMs and providing seamless migration for these machines such that user doesn't experience any drop in QoS is a very interesting problem by itself. Pre-copy algorithm used by VM and executed by hypervisor helps in having this seamless migration. The algorithm needs multiple rounds of iteration until the VM is transferred to the destined location. If we consider n_j as the total number of rounds required for migration of VM j, round 0 is for static memory copy phase, round 1 to n_j-1 is for dirty memory phase and finally round n_j for stop and copy phase. The optimization model and parameters are given in the following section.

B. Parameters and VM Model

The precopy time and downtime which constitute total migration time are the two key parameters which has to be optimized for VM migration scenarios. The following parameters are used for VM model formulation [1], [2].

- M number of VMs to be migrated
- V_{memj} memory size of jth VM to be migrated
- D_{ij} memory dirtying rate during round i for VM j

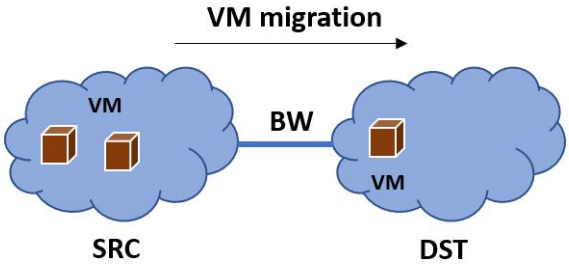


Fig. 3. High level concept diagram.

- V_{ij} amount of dirty memory copied during round i for VM j
- T_{ij} time to transfer V_{ij}
- n_j number of rounds for VM j
- r_{ij} channel bit rate reserved in round i to migrate VM j
- R total bit rate of migration channel (BW)

The equation for live migration are-

$$V_{0j} = V_{memj} = r_{0j}T_{0j} \tag{1}$$

$$V_{ij} = D_{i-1j}T_{i-1j} = r_{ij}T_{ij} \tag{2}$$

$$V_{n_jj} = D_{n_j-1j}T_{n_j-1j} = r_{n_jj}T_{n_jj} \tag{3}$$

$\forall$i =1 .. n_j and $\forall j = 1..M$, we can write (1) and (2) recursively and obtain these following equations-

$$T_{0j} = \frac{V_{memj}}{r_{0j}} \tag{4}$$

$$T_{ij} = \frac{V_{ij}}{r_{ij}} = \frac{V_{memj}}{r_{0j}} \prod_{h=1}^{i} \frac{D_{h-1j}}{r_{hj}} \tag{5}$$

Here, we consider $n_j = \bar{n}$ as a control parameter which we vary in our simulation to get optimal $\bar{n}$. From the above equations we get the equations for total pre-copy time and total down-time

- Total pre-copy time

$$T_{pre}(\bar{n}, r) = \sum_{j=1}^{M} \frac{V_{memj}}{r_{0j}} + \sum_{j=1}^{M} \frac{V_{memj}}{r_{0j}} \sum_{i=1}^{\bar{n}-1} \prod_{h=1}^{i} \frac{D_{h-1j}}{r_{hj}} \tag{6}$$

- Total Downtime

$$T_{down}(\bar{n}, r) = \sum_{j=1}^{M} \frac{V_{memj}}{r_{0j}} \prod_{h=1}^{\bar{n}} \frac{D_{h-1j}}{r_{hj}} \tag{7}$$

For M machines, in eq (6) the first term represents the time required for static memory copy phase and the next term represents time required for dirty memory copy phase which is iterative.In general $T_{pre}(\bar{n}, r)$ determines the time when the stop and copy phase will occur.Eq (7) represents time required for stop and copy phase which is basically the last step in VM migration process.It's always better to have $T_{pre(\bar{n},r)}$ and $T_{down}(\bar{n}, r)$ as low as possible.

C. Optimization Problem

At each round i of migration process, we have to allocate optimum bit rate r_{ij} given V_{memj} and D_{i-1j} for VM j.The problem can be rewritten as follows [1],

$$minimize \;\; T_{pre}(\bar{n}, r) + T_{down}(\bar{n}, r) \tag{8a}$$

subject to

$$D_{i-1j} < r_{ij} \;\; \text{(Sustainable migration)} \tag{8b}$$

$$\sum_{j=1}^{M} r_{ij} \leq R \;\; \text{(BW constraint)} \tag{8c}$$

$$r_{ij} > 0 \;\; \text{(Rate constraint)} \tag{8d}$$

$\forall i = 1..n_j$ and $\forall j = 1..M$, also the optimization variable is r_{ij} and control parameter is $\bar{n}$.
This problem can be interpreted as minimization of total migration time i.e the sum of total precopy time and the total downtime subject to migration,bandwidth and rate constraint. By solving the above problem we get the optimum bit rate required per round and per machine such that the migration time is minimum and user experiences a very low service interruption.The constraint (8b) ensures that during the entire migration process the memory produced is less than the transfer rate which guarantees migration is feasible. Also, (8b) is the constraint pertaining to bandwidth which limits any access allocation against available resources. And finally we have the rate constraint (8c) which should always be positive to have a viable migration (implicit constraint).

D. Geometric Programming Approach

The above optimization problem (8a) is a geometric program having a standard form [3]

$$minimize \;\; f_0(x) \tag{9a}$$

subject to

$$\begin{aligned} f_i(x) &\leq 1, \quad \text{i = 1,..m} \\ g_i(x) &= 1, \quad \text{i = 1,..p} \end{aligned} \tag{9b}$$

where f_i are posynomial,g_i are monomials, and x_i are the optimization variables.The variables have to be positive ($x_i > 0$).In standard form, the objective function should be posynomial which has to be minimised, equality constraint should have monomials equal to one and inequality constraint should have posynomials less than or equal to one.
The monomial and posynomial are defined as below

- Monomial

$$f(x) = c x_1^{a_1} x_2^{a_2} ... x_n^{a_n} \quad c > 0 \, , \, a_i \in R \tag{10}$$

- Posynomial: sum of monomials

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} ... x_n^{a_{nk}} \quad c_k > 0, \, a_{ik} \in R \tag{11}$$

where , x_1,....,x_n denote n real positive variables.

III. Results

To validate the model, simulation were carried out using MATLAB GGPLAB solver. It is a primal-dual interior point solver for GP (geometric program) to solve convex version of the original geometric program. VM memory size is modeled using uniform distribution with different mean to include randomness and currently the dirty rate is assumed fixed during each round with rate of 2500 pages per second which maps to 10.24 MBytes/s (considering linux standard page size of 4096 bytes) [1].The impact of transfer rounds, number of VMs and also bandwidth are studied. A brief overview of the performance of pre-copy algorithm with respect to the above parameters are discussed in the following subsection.

A. Impact of transfer rounds $\bar{n}$ on VM migration

From Fig. 4 it is clear that we only need fewer number of rounds for VM migration as the total time ie. sum of precopy and downtime saturates after 4-5 rounds which in turn depends on Vmem size. Higher the Vmem size higher the total time for migration. For an average Vmem size of 0.5GB, dirty rate (D) of 10.24MB/s, bandwidth (R) of 1Gbps and M=3 we only need 4 rounds for precopy algorithm to converge as after 4 rounds there is no significant improvement in total time.Similarly for Vmem size of 1GB and 4GB we need 5 rounds to have low migration time.An interesting case is of Vmem size 4GB which requires same number of rounds as Vmem size 1GB but has a total time which is much higher as compared to Vmem size of 1GB.

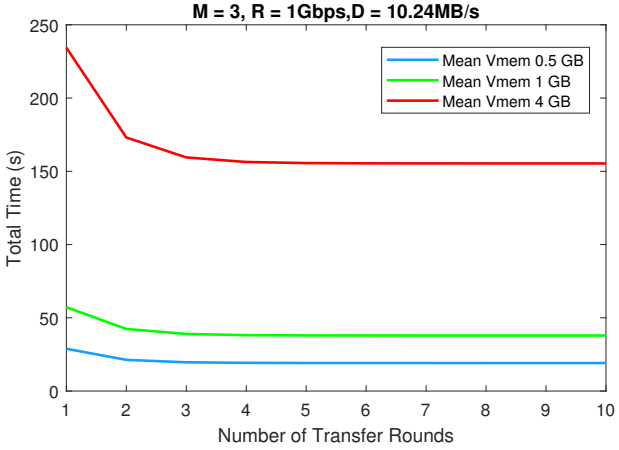


Fig. 4. Vmem Size vs Transfer Rounds

B. Impact of number of VMs for migration

For a fixed mean Vmem size of 1GB as the number of VMs increases the total time for migration increases. Migrating 1 VM takes less than 4s as compared to 150s for migrating 5 VMs simultaneously. Due to the low availability of bandwidth it's not feasible to migrate large number of VMs as the migration time increases rapidly. Also, lot of network as well as compute resources are wasted during this process. A sample rate allocation of the optimization problem for different

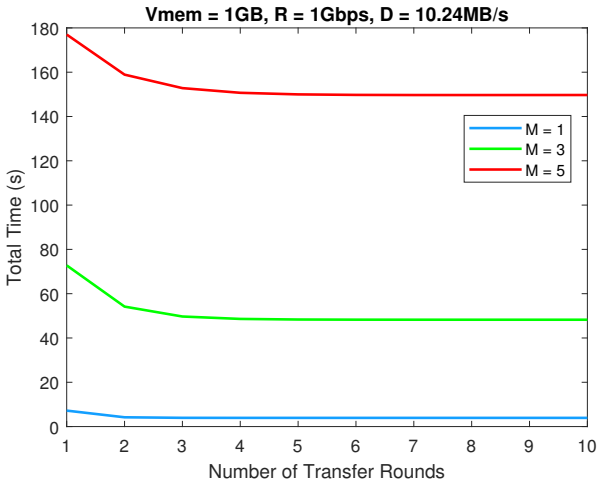


Fig. 5. Number of VMs vs Transfer Rounds

number of VMs with average size 1GB and bandwidth 1 Gbps (125 MB/s) is given in the table below. Based on the individual VM memory size optimal rate is allocated by GP solver such that it provides low migration time. Also, as the number of VMs increases optimal rate allocation converges to static R/n allocation.

Optimal Rate Allocation

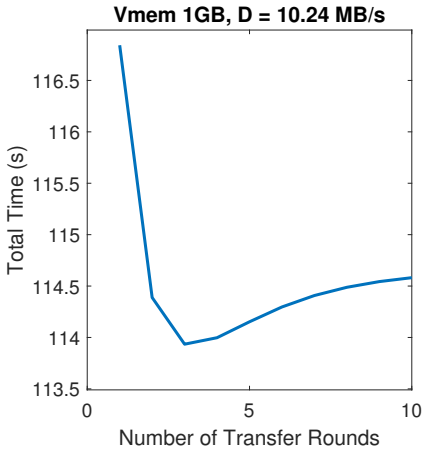
Number of VMs	Vmem Size	Rate (r_{ij})
1	744.07 MB	125 MB/s
3	500.02 MB	38.48 MB/s
	479.92 MB	37.88 MB/s
	904.72 MB	48.62 MB/s
5	609.86 MB	24.02 MB/s
	617.66 MB	24.12 MB/s
	859.44 MB	26.92 MB/s
	805.48 MB	26.34 MB/s
	576.72 MB	23.58 MB/s

C. Perturbation analysis

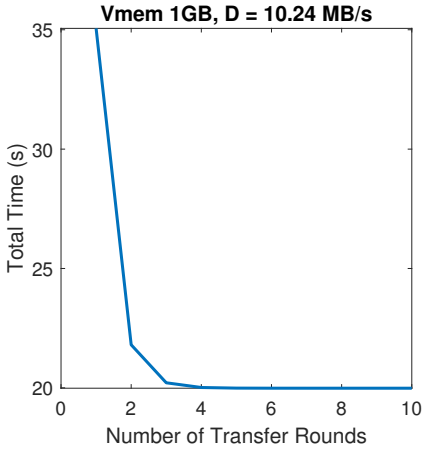
From Fig. 6(a), reducing the bandwidth to half (or tightening the rate constraint) the total time for migration decreases at first with the increase in transfer rounds. But after 3 rounds the total time increases which means the algorithm is not going to converge and if we have a very low bandwidth it's always better to migrate as early as possible with very few dirty memory copy phase rounds. On the other hand if we have a larger bandwidth Fig. 6(b) (loosing the rate constraint) we see a very low total time which converges after 4 rounds [3]. Having a large bandwidth always facilitates faster and efficient migration.

IV. Conclusion

Live migration of virtual machines was studied, a requirement for present day data center networks. A GP based optimization problem was formulated for allocating optimal bit rates between various machines such that when migrated



(a) R = 0.5 Gbps



(b) R = 2 Gbps

Fig. 6. Impact of bandwidth $\mathbf{R}$ on VM migration

simultaneously results in low total migration time. A detailed system analysis on pre-copy algorithm was carried out with respect to different parameters settings like transfer rounds,bandwidth and dirty rate. With the help of optimization results we can quantify that fewer number of transfer rounds are enough as the total time saturates after sometime and also if the bandwidth is too low it's always better to migrate as early as possible avoiding the iterative dirty memory copy phase.

Acknowledgment

I would like to thank Prof. Kristin Dana for her feedback and support all throughout the course.

References

[1] Cerroni, Walter, and Flavio Esposito. "Optimizing live migration of multiple virtual machines." IEEE Transactions on Cloud Computing (2016).

[2] Liu, H., Xu, C. Z., Jin, H., Gong, J., Liao, X. (2011, June). Performance and energy modeling for live migration of virtual machines. In Proceedings of the 20th international symposium on High performance distributed computing (pp. 171-182). ACM.

[3] Boyd, S., Kim, S. J., Vandenberghe, L., Hassibi, A. (2007). A tutorial on geometric programming. Optimization and engineering, 8(1), 67.

APPENDIX

A. Main Code File

```
addpath C:\Rutgers-GA\CONVEX\ggplab;
%close all
clear all
global QUIET;
QUIET =1;

trails=10;
rounds=10; % rounds= Vmem+dirty pages+stop
    phase=nj

plotta =1;
Cmig =1; %indicator variables
Cdown =1;%indicator variables

M = 3;% number of VMs
avg = 1000;%Vmem mean

Z = [125/2,125,125*2];% 1 Gbps = 125 Mbytes/s
    - standard-BW
D = 10.24; %2500 pps = 81.92 Mbps = 10.24
    MBytes/s -standard

for a=3 %1=0.5Gbps,2=1Gbps,3=2Gbps
   Rmax=Z(a);
   for t = 1:trails
      p = ['Trail Number = ', num2str(t)];
      disp(p);
      for j = 1:M
         Vmem(j) = avg*rand(1); %uniform
             distribution
         Vmatrix(j,t,a)=Vmem(j);
      end
      for nj=1:rounds %number of rounds
         gpvar r(M); %gp variables - optimal
             rate allocation to minimize
             total migration time

         %objective 1- down time
         Tdown = posynomial;
         Tdown = objTdown(Vmem,r,D,nj);

         %objective 2- migration time
         Tmig = posynomial;
         Tmig = objTmig(Vmem,r,D,nj);

         % objective 1 + objective 2 = Total
             time
         objective = posynomial;
         objective = Cmig*Tmig + Cdown*Tdown;

         con1=D<=r(1); %constraint 1
         con2=0.00001<=r(1);%contraint 2
         for j=2:M
            con1=[con1;D<=r(j)];
            con2=[con2;0.00001<=r(j)];
         end
         rtemp=0;
         for j=1:M
            rtemp=rtemp+r(j);
         end
         con3=rtemp<=Rmax; %contraint 3
         constraint = [con1;con2;con3];
         % solver
         [tottime,solution,status] =
             gpsolve(objective,
             constraint,'min');
         assign(solution)

         % Store input
         for k=1:size(r)
            Rmatrix(k,nj,a)=r(k);
         end
         TotTime(nj,t,a) = tottime;
      end
   end
end
```

B. Objective Function I

```
function Tdown = objTdown(Vmem,R,D,nj)
M=size(Vmem,2);%number of VMs
Tdown=posynomial;
Ttemp=posynomial;
for j=1:M
   if(nj>=2)
     Ttemp=(Vmem(j)*D^(nj-1))/R(j)^nj;
   else
     Ttemp=Vmem(j)/R(j);
   end
   Tdown=Tdown+Ttemp;
end
return
end
```

C. Objective Function II

```
function Tmig = objTmig(Vmem,R,D,nj)
M=size(Vmem,2);
Tmig=posynomial;
for j=1:M
   Ttemp1=posynomial;
   Ttemp2=posynomial;
   Tmigtemp=posynomial;
   for i=1:nj-1
      Ttemp1=(D/R(j))^i;
      Ttemp2=Ttemp2+Ttemp1;
   end
   Tmigtemp=(Vmem(j)/R(j))*(1+Ttemp2);
   Tmig=Tmig+Tmigtemp;
end
return
end
```