Resource Allocation Simulator for V-NetLab

Mingwei Zhang

Feburary 20, 2011

1 System Structure

1.1 OBJECTIVE

The resource allocation simulator is an evaluation tool in V-NetLab system. Its purpose is to quantitatively evaluate the performance of the online resource allocator, a component in V-NetLab that allocate the virtual networks to physical hosts while satisfying the CPU, memory and network request in an optimized way. In particular, we use an offline allocator that generates the best allocation to compare the result from the online allocator. A schedule generator is designed to give the same schedules to both online and offline allocator and their allocation result will be evaluated by a performance evaluator.

1.2 SYSTEM STUCTURE & TERMS

Figure 1 shows the structure of the resource allocation simulator.

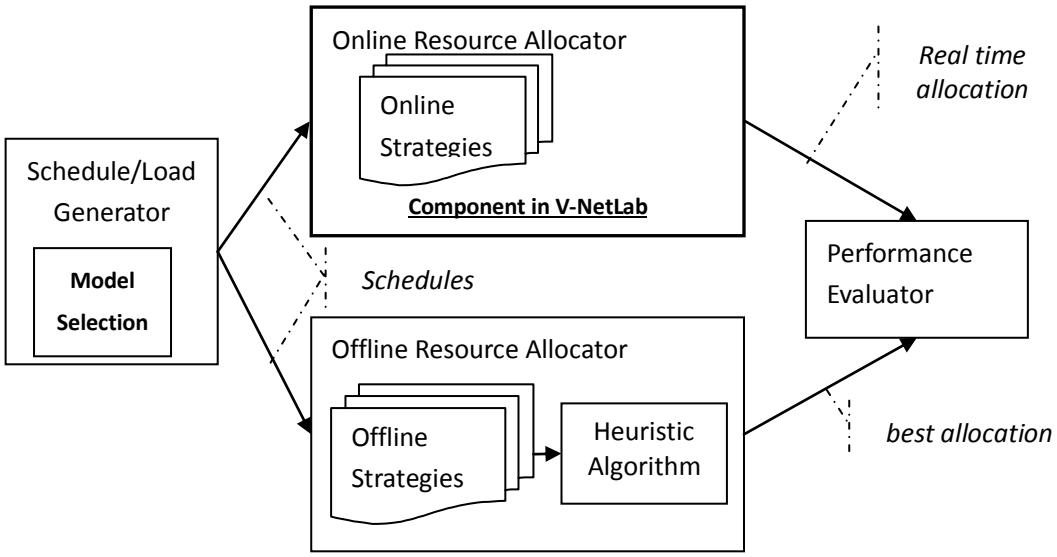

Fig. 1 Structure of Resource Allocation Simulator in V-NetLab

TERMS:

Schedule: schedule temporal sequence of task request. In particular, it is a sequence of two value tuples: <time, networkID>.

Allocation: allocation is a mapping of network with physical host as <networkID, host>. Note that a network could be separated into several parts; it's possible that one networkID may be mapped to several hosts.

Best allocation: best allocation is the allocation generated by offline algorithm. It represents the "best allocation" that we could find.

Real time allocation: real time allocation is the result given by online algorithm.

Strategy: strategy is the method or algorithm for allocator to make decisions that which host the request network should be assigned to.

Online Strategy: strategy that used by online allocator.

Offline Strategy: strategy that used by offline allocator.

Heuristic Algorithm: In particular, in this system, heuristic algorithm means simulated annealing algorithm.

2 Schedule Generator

2.1 Model Selection

This chapter will focus on the work of the load generation which is regarded as **a substitution of the previous wok (Munya 2008).**

In order to accurately measure the performance of online allocator, the arrival time of incoming requests should be "randomized", since the requests are generated by several users independently and unpredictably. Mathematically, this "randomized" feature will demonstrate the characteristics of long-range dependency, and infinite variance. And because of this character (Uhlig and Bonaventure 2001; Horn, Kvalbein et al. 2007), the load schedule should indicate a feature of burstiness, a.k.a self-similarity(Leland, Taqqu et al. 1993)

In this system, we decide to use the pareto distribution with ON/OFF model, a mature model for network traffic generation and simulation (Becchi 2006). The reason for pareto distribution is because of its features of **long-range dependency, and infinite variance**(Wikipedia), while ON/OFF model is the mature model for network traffic generation(PARK and WILLINGER 2000; Becchi 2006; Horn, Kvalbein et al. 2007).

2.2 Specifications

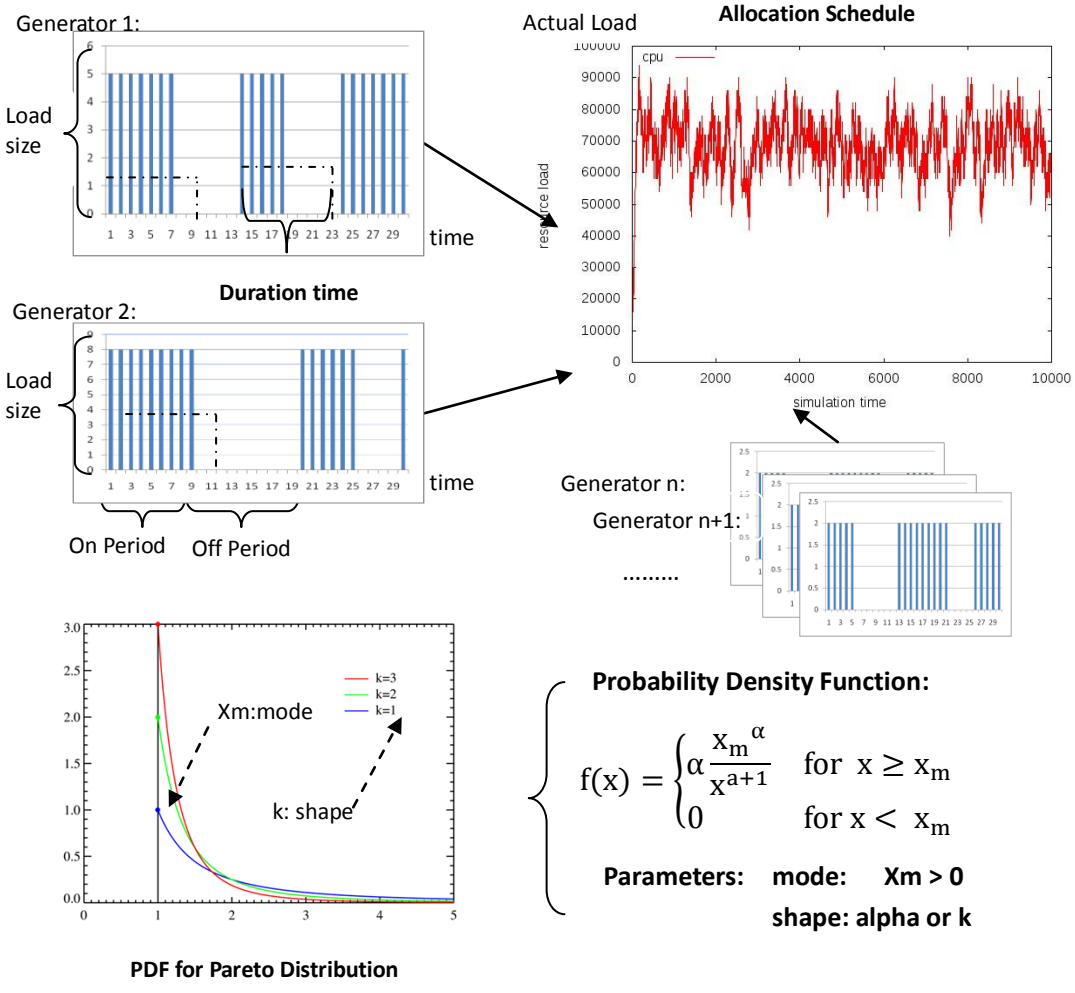

Fig. 2 Self-Similar Load Generator

Critical Parameters:

Number of generators:

Mean (μ_{on}) & Variance ($\varepsilon_{on}{}^2$) time for On Period: In pareto distribution, the mean and the variance are interpreted by mode and shape with the following equation:

$$u_{on} = \frac{\alpha x_m}{\alpha - 1} \qquad for\ \alpha > 1 \qquad (1.1)$$

$$\varepsilon_{on}{}^2 = \frac{\alpha {x_m}^2}{(\alpha-2)(\alpha-1)^2} \qquad for\ \alpha > 2 \qquad (1.2)$$

In particular, when shape is less than 2, then the variance is infinite.

Mean (μ_{off}) & Variance (($\varepsilon_{off}{}^2$) time for OFF Period: these two parameters are the same as above.

Size of Load (μ_l): size of load is the mean value of the network resource request as CPU request, memory request or network bandwidth request. For each generator, the load size is fixed to some value as we should decide.

Duration (μ_d): specifically means the duration of one request. **In our system, duration of requests follows normal distribution with variance 1 .**

Target Load (Lt_i): target load is the overall load that the generator is expected to generate.

How to compute Target Load:

The following is the formula to compute the target load:

$$Lt_i = \mu_d * \mu_l * {}^{\mu_{on}}/_{(\mu_{on} + \mu_{off})} \qquad (1.3)$$

$$Lt = \sum_{i=1}^{n} Lt_i \qquad (1.4)$$

Where, Lt_i is the target load of generator i; Lt is the total target load. μ_d is mean value of duration; μ_l is the mean value of load size; μ_{on} is the mean value of on period and μ_{off} is the mean value of off period.

Note that the above computation is just an approximation of the real target load in that parameters as variance are not considered. The actual target load could vary from the expected Lt because the length of off and on period could be far away from the given mean value.

3 Online and Offline Strategies

The FUNDAMENTAL difference between online strategies and offline ones is whether it can see the "future". For offline algorithm, it has the whole view of the schedule; moreover, it can change the allocation that has been made before. However, online strategy can only make decisions on the current situation and they can never see the incoming requests.

3.1 online strategies

Online strategies could have several metrics that may help them to make a decision. These metrics includes the following:

- current available CPU capacity:
- current available mem capacity:

- current available nw capacity:
- number of networks each host currently have
- the guess value of average duration time of each request
- the guess value of average load size of each request
- the current completion rate of each allocated network

Some Simple Strategies:

First fit: always start from the beginning of the Host List and find the first host that fits a part of the request or a whole request. If it only fits a part, then split the request, do the partial allocation and move on to next host.

Round-Robin: same as the first fit except that it starts from the last host that accepts the allocation. When searching to end, the algorithm will come back to the beginning of the host chain.

Memory-First: only consider only the memory request and use First fit strategy to do the job. This means it never move on to the next host until the memory on that host was used up.

Bottleneck Minimum Strategy

Basic idea: Good strategies should tradeoff the average completion rate and the fairness in an appropriate way.

In order to control the fairness, the strategy will try to average the load among all the hosts by balancing the bottleneck ratio* (see section 4.1 for bottleneck). If every host has the load that is proportional to their capacity, the virtual networks will be less possibly suffered from bottlenecks of resource limitation on each host.

The strategy will the all the host within range above or below the average bottleneck (*mean*) ratio. The value of the range (*vary*) is a parameter that could change.

In particular, when doing one specific allocation, the strategy will have several tactics as follows:

- **Always start from the lightest loaded host.** Using this tactics will assign a portion until the new bottleneck of that host reach average value. And then move to the heavier loaded one.
- **Always start from the heaviest loaded host,** do the allocation until bottleneck value reach to *mean+vary*
- **Setting up some "special hosts" for small networks**. Divide the allocation place of small networks from larger ones will increase the average completion rate (*mean*), since small networks are easy to satisfy.

3.2 offline strategies

Offline strategies may have many extra "clues" to find a good allocation. These clues include the following:

- the exact time when a specific request stops,
- the number of incoming requests and their
- load sizes in the next period of time.

Besides the "clues", offline strategies have one more advantage that they can change reallocate the virtual networks they have already made.

3.3 Simulated Annealing

4 Performance Evaluation

This chapter will clarify the method and metrics that we use to evaluate an allocation.

4.1 Cost Model

Terms:

- **Virtual Machine (vm):** is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. In V-NetLab, a vm is the atomic unit of a user request.
- **Virtual link (vlink):** a link that connects two vms. The link could spans two physical hosts or within one host.
- **Virtual Network (vnet):** a group of vms that are connected by virtual links. A vnet will serve as one user request. Note that a vnet is a connected graph.
- **CPU Time Slice Request (Cr):** the CPU slice that an individual vm requests. The value is generated by the user.
- **Actual CPU Time Slice(Ca):** the actual CPU time slice that an individual vm could use.
- **Network Bandwidth Request (Nr):** the network bandwidth that an individual vm requests. Note that the Bandwidth Request may be consist of several Link Requests.
- **Actual Network Bandwidth (Na):** the actual network bandwidth that an individual vm could use.
- **Network Link Request (Lr):** the network bandwidth request of one virtual link.
- **Actual Link Bandwidth (La):** the actual network bandwidth that one virtual link could use.
- **CPU Capacity (Hc):** refers to the CPU capacity of a physical host.
- **Network Capacity (Hw):** refers to the network bandwidth of a physical host.
- **Bottleneck**: the bottleneck refers to the physical hosts that have the heaviest load compared with its capacity. Specifically, there are two kinds of bottlenecks: CPU Bottleneck and network bottleneck.
- **CPU Bottleneck Value (Bc)**: the value of CPU bottleneck is represented as $\mathrm{Bc} = \sum \mathrm{Ca}/\mathrm{Hc}$, where $\sum \mathrm{Ca}$ represents all the vms in this host.
- **Theoretical CPU Completion Rate:** interpreted as 1/Bc.
- **Network Bottleneck Value (Bw):** the value of network bottleneck is represented as $\mathrm{Bw} = \sum \mathrm{La}/\mathrm{Hw}$, where $\sum \mathrm{La}$ represents all the vlinks that connect this host with some other ones.
- Theoretical Network Completion Rate: interpreted as 1/Bw.

Assumptions:

- The overall completion rate is the lowest one of the vms inside the network according to the barrel effect.
- If memory request cannot be satisfied, then the completion rate of the vm is 0% and thus the overall completion rate is 0%
- Since CPU and network are shared resources, all the vms in one physical host will have a proportional request of one resource if the other one could be fully satisfied in this host. (ie. Vm1 requests 200 unit cpu; vm2 requests 100 unit cpu. If the host has only 150 cpu unit and network is fully satisfied, then vm1 will get 100 and vm2 will get 50)
- The network usage will be linearly proportional to CPU usage. (ie. Vm1 requests 200 cpu unit and 100 network bandwidth unit. If vm1 get only 100 cpu unit, it will get 50 network unit and this is true vice versa)
- Individual vm will always have identical completion rates of CPU request and network request because of the rule above

The following scenario is given to help computing the completion rate of all the virtual networks in a given allocation:

Scenario 1:

For a given allocation, a mapping of vms and hosts, if all the vms stop working and then start gradually by increasing CPU usage linearly and synchronously with one percentage each period. This process will proceed until no vm could increase its resource usage.

Lemma 1:

If we put <hostID, Bc> and <hostID, Bw> of each host into a list and sort it in descending order by value, eliminate duplicate hostID with smaller value. And we get a time order that the hosts will stop increasing resource usage due to its capacity limit.

Prove:

Lemma 2:

Once a host has stopped increasing resource usage due to the capacity limit, ***the resource usage of vnets that contains the vms on that host will never increase****, while the rest could still increase.*

Prove:

Theorem 1:

For a given vnet, the completion rate is decided by the first stopped host that contains the vms of that vnet.

Prove:

Algorithm:

Step 1: Is there vnets available? If so, goto Step 2, otherwise goto Step 6

Step 2: Compute Bc and Bw for each physical host.

Step 3: Put both the Bc and Bw into a list and sort it. (Lemma 1)

Step 4: get the host with the highest value (Lemma 2)

Step 5: compute the completion rate of all related vnets (Theorem 1)

Step 4: deletes these vnets and the physical host.

Step 5:update the Hc and Hw for the rest of hosts. Goto step one

Step 6: Finish and return.

4.2 Goodness of Allocation

The goodness of an allocation will be measured by both the average completion rate and their variance. The following gives the formula to compute the goodness of an allocation:

$$G_t = \frac{\mu_t}{1+\frac{\varepsilon_t{}^2}{\alpha}} \qquad (4.1)$$

$$G = \sum_{t=0}^{T} Gt \qquad (4.2)$$

Where:

In formula 4.1, Gt is the instantaneous goodness μ_t is the mean value of instantaneous completion rate among the networks at time t, $\varepsilon_t{}^2$ is the corresponding variance. α is a variable that describe the sensitiveness toward the change of variance. Specifically, a large α will make the goodness less affected by the variance.

In formula 4.2, G is the quantitative measurement of goodness over the period of duration [0, T]; T is the simulation time.

This formula 4.1 will fully represents by the following charts, where alpha is 0.01.

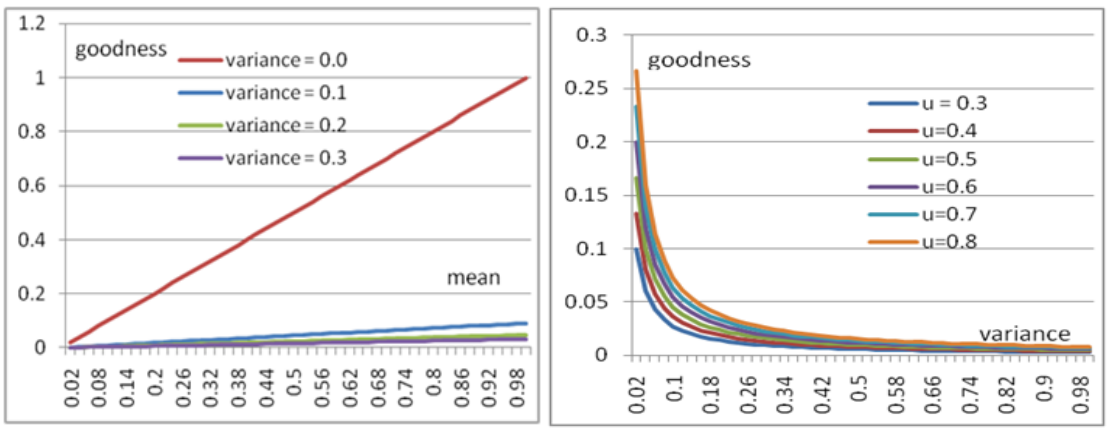

Fig 3. a) when the variance is fixed b) when the mean is fixed

Reference

Becchi, M. (2006). From Poisson Processes to Self-Similarity: a Survey of Network Traffic Models. cse567-06-ftp. Washington Washington University in St. Louis.

Horn, G., A. Kvalbein, et al. (2007). "An empirical comparison of generators for self similar simulated traffic." Perform. Eval. **64**(2): 162-190.

Leland, W. E., M. S. Taqqu, et al. (1993). On the self-similar nature of Ethernet traffic. Conference proceedings on Communications architectures, protocols and applications. San Francisco, California, United States, ACM**:** 183-193.

Munya (2008). VNetLab Resource Allocator.

PARK, K. and W. WILLINGER (2000). SELF-SIMILAR NETWORK TRAFFIC AND PERFORMANCE EVALUATION. New York, NY, USA, John Wiley & Sons, Inc.

Uhlig, S. and O. Bonaventure (2001). Understanding the Long-Term Self-Similarity of Internet Traffic. Proceedings of the Second International Workshop on Quality of Future Internet Services, Springer-Verlag**:** 286-298.

Wikipedia. "Pareto distribution." from http://en.wikipedia.org/wiki/Pareto_distribution