Towards realistic benchmarks for virtual infrastructure resource allocators

Qin Yin, Timothy Roscoe
Systems Group, Department of Computer Science, ETH Zürich

Abstract

The rise of virtual infrastructures has renewed research interest in the design and implementation of resource allocators which allow users to reserve or use combinations of (virtual) nodes, switches, and network links on a variety of virtual infrastructures ranging from network testbeds to cloud computing facilities. However, since the area is still relatively new, work on resource allocation lacks realistic benchmarks. In this paper, we suggest how to generate realistic benchmarks for virtual infrastructure, and present our workload generator as an example, which is based on a 5-year trace of experiments submitted to the popular Emulab testbed. We demonstrate the importance as well as potential applications of such benchmarks by using the generated test workload in various evaluation scenarios.

1 Introduction

In Infrastructure-as-a-Service (IaaS) systems and networking testbeds, resource requests composed of virtual machines, virtual routers and virtual links are allocated from an underlying shared physical infrastructure. A significant challenge is how to map a virtual network (VN) topology with resource constraints to specific nodes and links in a given shared physical network (PN) infrastructure so as to accurately emulate the network properties requested by users. This may involve virtual network embedding, resource constraint satisfaction, and even concepts from operations research such as yield maximization, in mapping multiple requests for resources to a given physical infrastructure.

The problem has inspired research in a number of related areas, one example being the design of efficient network mapping algorithms. Another is the design and implementation of resource allocators: how users can specify their resource requirements, how resource providers can satisfy many users while optimizing utilization, revenue, power consumption, or some other metric of interest.

To evaluate and compare resource allocators, the research community requires benchmarks to infer system resource allocation behavior, to identify the impact of design choices, and to compare different resource allocation strategies. However, to date there is no clear consensus on what constitutes a realistic workload for such systems, let alone on a common benchmark for comparing ideas.

Existing work uses mostly simulators to generate virtual network requests and substrate network for evaluation. However, randomly generated graphs can not faithfully represent real requests and physical infrastructure. Such synthetic workloads may even miss key aspects in reality that significantly impact system design. Benchmarks based on real trace are less susceptible to this.

We argue that the accuracy of system evaluation results depends on the accuracy and realism of the benchmarks used, and realistic benchmarks should be constructed from real-world workload traces. Of course, this is not a new position. Our contributions for virtual infrastructure resource allocators are as follows:

- A reference model which defines the design

space of resource allocators;

- A general methodology for generating realistic benchmarks, and an example workload generator based on a 5-year trace of experiments submitted to the popular Emulab testbed [12];
- Results from using the generated test workloads in three evaluation scenarios.

2 Virtual infrastructure resource allcoation

Figure 1 depicts a general reference model for resource allocators which includes two stages: *resource reservation* and *resource allocation*. The user submits the resource request specifying the desired elements in the virtual network, constraints on and dependencies between these elements, and the period of time for which the requested resources are desired. If satisfiable, the provider will return a *ticket* as a reservation of a set of resources.

Given a ticket, a user may redeem it for a *lease* before its expiry time (some time before the specified *StartTime*). This redemption confirms the allocation of concrete resource components (activation), and allows access to these resources.

Almost all resource allocation systems are a (non-strict) subset of this model – for example, some choose to conflate leases and tickets. We survey the three main application areas below.

2.1 Testbeds

Early network testbeds such as PlanetLab [23] provide only best-effort VMs to users, but the design is general enough to support brokerage services on top: Sirius [25] allows users to reserve VMs system-wide for an hour at a time. Alternative market-based services like SHARP [14] allow trading of resources.

The Emulab [11] network emulator does not support advance reservation but performs guaranteed-share scheduling. Emulab users may request a set of nodes for experiments, and to specify how they are configured to emulate network topologies. Emulab uses a network mapping algorithm called `assign` which uses simulated annealing to find a near-optimal solution in bounded solving time.

The GENI [16] program aims at both high best-effort utilization and admission-control for guarantees to some clients. It includes several control frameworks: VINI [30] is a PlanetLab-like testbed where users can configure virtual topologies. ProtoGENI [24] is based on Emulab software, but with a different subset of the model in Figure 1 using tickets. ORCA/BEN [22] is control plane (ORCA), which differentiates tickets from leases, for a metro-scale optical testbed (BEN).

2.2 Grid and cloud systems

Grid systems differ from infrastructures like GENI. Firstly, they focus purely on nodes rather than networks. The Globus toolkit [17] distributes large scientific applications across a set of computational resources, and Condor [9] allocates machines to parallel jobs based on a match-making mechanism called ClassAds. Secondly, resource allocation typically uses a batch scheduling job queue: when a job reaches the front of the queue, it gets dedicated access to resources for its run.

Public clouds like Amazon EC2 [10] have many machines, but advertise a small number of VM classes, based on location and approximate computing power. They differ from testbeds in three ways. Firstly, machines are allocated piecemeal and access is granted at time of request – there is no notion of (nor need for) resource reservations. Secondly, in cloud backends, the use of admission control and capacity planning ensure that jobs have enough resource to run with little wait time, something hard to achieve in network testbeds. Finally, until recently allocating shares of the interconnect between nodes was not a feature offered by providers.

Private/hybrid cloud systems like Eucalyptus [13] take the middle ground between public cloud systems and testbeds. They encourage third-party development of the scheduler module. The Haizea [28] lease manager for OpenNebula can allocate VM resources under a variety of terms, including reservations and best-effort queueing.

In this paper, we focus on realistic benchmarks for testbeds. However, the general methodology

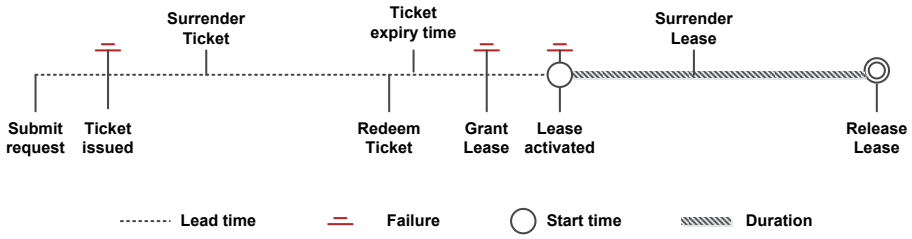

Figure 1: Timeline of a resource request

presented below is also applicable to grid and cloud systems.

3 Benchmark methodology

Benchmarking consists of running a system to evaluate its performance with respect to some predefined benchmark metrics, typically by executing a number of standard well-designed benchmark workloads against it. The generation of the benchmark workloads to be used to test both the current available system and new emerging system requires an accurate workload characterization model. Calzarossa et.al. present a through survey of workload characterization [4] and analyze the issues and methodologies [3]. Workload characterization and benchmarking is considered of key importance in many application areas: I/O systems [6], database management systems [29], Internet web servers [1, 2], etc.

The design of benchmarks for Grid technology has provided a useful input to our work. Chapin et.al. explore creating benchmarks to evaluate parallel job schedulers [5]. Others have proposed a comprehensive model for supercomputer [8] and parallel computers[27] workloads. Recently, Ganapathi et.al. have proposed statistics-driven workload modeling for the cloud [15], and several groups have analyzed a Google compute cluster trace [21, 7]. Workload characterization results are used to predict workload patterns [19] and to synthesize realistic workloads [31].

Virtual infrastructures are still relatively new, particularly those which are distributed in nature and permit complex topological requests, and it is not yet clear what a representative workload for a resource allocator is. Based on a through survey of related work in other fields, we summarize the steps to realistic workload generation: 1) choosing the parameter set which can describe workload behavior; 2) using monitoring tools to collect system workload trace; 3) analyzing the collected trace, applying statistic methods to characterize it and construct workload models; 4) using the constructed model to generate realistic workloads.

In this section, we describe a general approach to generate realistic benchmarks and use our workload generator as an example. Our workload generator is based on a trace of Emulab experiments for the 5 years prior to June 2007, essentially covering every experiment submitted to Emulab before that date. While we make no authoritative claims that this is representative (indeed, the information in the trace is limited and Emulab is unlikely to be representative of other virtual infrastructures), we argue that it is at least based on plausible data and assumptions.

3.1 Parameter selection

The choice of the parameter set to thoroughly describe workload behavior, depends critically on the type of the system. For virtual infrastructure resource allocation reference model, we use the following parameters to characterize VN requests:

- VN request topology with constraints on the requested resources;
- VN request submission time;
- Lead time between request submission and lease activation;
- Duration for which resources are used;
- Probability that a ticket is canceled before it expires;
- Probability that a ticket expires;

- Probability that a lease is surrendered before its specified duration.

3.2 Trace collection

The collection of the trace used to generate realistic benchmarks should ideally use measurment or monitoring tools to collect data, possibly merge information from different sources, ensure a long enough time span, cover details of all the selected parameters, and retain anonymity to ensure users' privacy.

For the resource allocation reference model, its trace should include at least the following information for each request: requested constrained topology, desired duration, timestamps for resource reservation, lease activation and cancellation, as well as all the possible events of ticket surrender, ticket expiry, lease surrender, etc.

Emulab follows only a subset of the reference model, and does not support resource reservation. Emulab logs cover many details of its system execution, however, due to anonymity reasons, the published Emulab trace is limited. Nevertheless, it is a valuable source of information: the trace covers 23818 anonymized experiments, consists of 23GB of `.top` and `.ptop` files. A `.top` file describes a virtual topology requested by an Emulab user, and a `.ptop` file describes the physical network topology that was available at the time the system tried to swap in the experiment.

3.3 Workload characterization

Since the behavior of a real workload is complex and difficult to reproduce, a compact, repeatable and accurate model is needed. Such a model has to capture the statistical characteristics of the real load. More specifically, for the resource allocation reference model, given a complete and detailed workload trace, a generative workload model should be able to describe the frequency distribution of VN requests, the request inter-arrival time distribution, lead time and duration distribution, probabilities of ticket expiration and early ticket/lease surrenders. A more realistic workload model should also capture the relationship among these parameters. These statistical models can later be used to generate realistic workloads.

Jelena et.al.'s comparative study of network testbed usage characteristics [20] shows that distributions of features such as VN request durations and VN sizes are heavy-tailed: they span a wide range of values with most points clustered at small values, and with few points residing in the long tail. This conclusion also holds true for the Emulab trace we use [33]. In Section 3.4 we describe the assumptions we make about the distributions of parameters missing from the trace but important for workload generation.

With a generative workload model, we can generate request workloads by *sampling* these distributions.

3.4 Our workload generator

The Emulab trace includes virtual topology requests which contain information of several dimensions: requested VN size, topology, node and link constraints, etc. In our Emulab trace-base workload generator, we applied a simple method: extracting a request sequence from the available trace by sampling from the complete 5-year request set.

Request arrivals are modelled as Poisson processes with various rate values λ. The Poisson distribution expresses the probability of a given number of events occurring in a fixed interval of time. We vary arrival rate parameter λ to increase or decrease offered load. As a reference, Emulab received 4 requests per hour on average. We model *lead time* and *duration* as Gamma distributions. The Gamma distribution has long been used for modeling demand distribution in queueing systems [26, 18]. It has two parameters: $shape(k)$ and $scale(\theta)$.

Depending on different evaluation scenarios, our workload generator can synthesize workloads with various parameters. For the experiments present in Section 4, we make some further assumptions to simplify the problem: first, *ticket expiry probability* is 0, meaning that all the tickets are redeemed before they are expired; second, *early ticket surrender probability* is 0; third, *early lease surrender probability* is 0, meaning that leases are always released after *duration* time period.

Based on these distribution models, we annotate the Emulab request stream with different time parameters – when to request resources, when to allocate resources and when to release them – and generate our workload from this.

4 Benchmark applications

We briefly show three evaluation scenarios where the generated request workloads are applied. In these evaluations, the physical infrastructure is based on the biggest ProtoGENI site, with 627 nodes (switches and hosts) and 1163 links.

Comparing VN mapping algorithms: By generating various topology requests, we can compare different VN mapping algorithms with respect to the solving time, scalability, revenue-to-cost ratio, etc. by mapping a set of VN requests to the same physical network or by exhausting the physical network. The test workload for this experiment has no temporal attributes, and can be sampled from the set of all Emulab virtual network requests. In [33], we use the generated workload to compare different variants of VF2x algorithm, which is an efficient network mapping algorithm based on the VF2 subgraph isomorphism algorithm.

Investigating dynamic allocation behavior: We can generate sequences of resource requests and releases requests to the resource allocator to investigate its dynamic behavior under continuous requests with respect to resource utilization, number and percentage of successful mappings. In this scenario, the allocator does not support reservation and immediately assigns resources to the request for *duration* time. In this experiment, we set duration Gamma distribution parameters $shape = 0.3$ and $scale = 20$, and vary λ from 4 to 8 and 16 to increase offered load. We use these distributions to annotate the Emulab stream with two parameters: when to request resources and when to release them. Figure 2 shows the physical utilization under different loads. Not surprisingly, we see that the higher the load, the more likely it is that the allocator can fit small requests into the network and achieve higher utilization.

Evaluating allocation strategies: We previously proposed negotiating for resources using late-binding and constraints [32]. Having users specify requests as constraints, and providers reply with commitments also expressed as constraints, gives providers flexibility in late-binding of resources and more potential to optimize metrics like utilization.

The evaluation in [32] used simplistic synthesized data, but Figure 3 revisits the experiment using the more realistic workload generated from the Emulab trace. The parameters for the workload generator are $\lambda = 16$, lead time gamma distribution $shape = 0.8$ and $scale = 20$, duration gamma distribution $shape = 0.3$ and $scale = 20$.

We compare a greedy allocation with a simple late-binding strategy whereby when a ticket request can not be satisfied respecting all prior ticket assignments, the provider picks the unredeemed ticket with the largest overlapping request and resolves both requests together. Figure 3 shows the period between 75 and 100 minutes of a 3-hour experiment, and shows that even this simple strategy can increase utilization most of the time.

5 Discussion

We have presented only the first steps towards a standardized benchmark suite for virtual infrastructure resource allocators, but we argue that such a goal is an important one for the research community to make progress in this field. While the Emulab trace we use has only limited information, it does provide a first cut at creating workloads to test a variety of properties of allocators in a range of different scenarios.

Acknowledgements

We would like to thank Robert Ricci and Fabien Hermenier for sharing the Emulab trace, Jelena Mirkovic for early access to her comparative study of network testbed usage, John Wilkes for many insightful discussions, and the anonymous reviewers for their feedback.

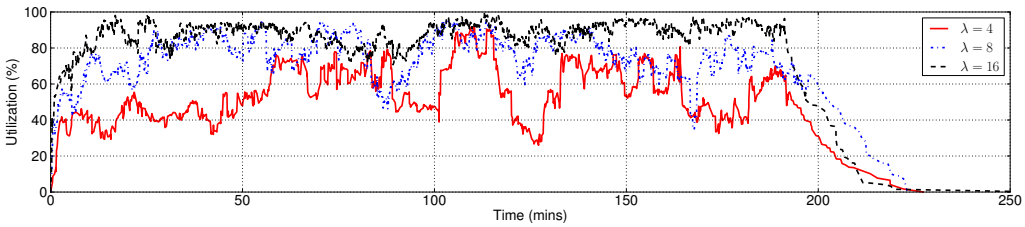

Figure 2: Executing the generated trace of arrival rate $\lambda = 4, 8, 16$

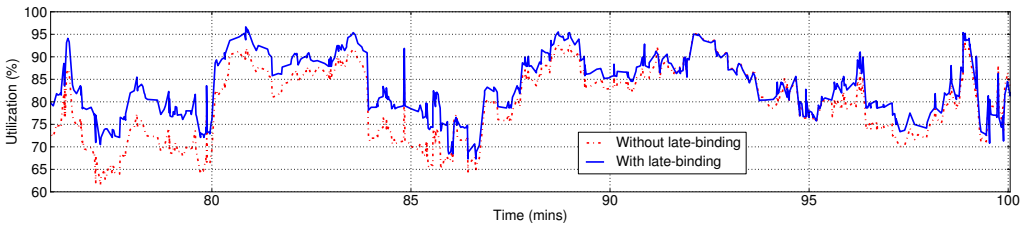

Figure 3: Highlight of the full timeline

References

[1] ARLITT, M. F., AND WILLIAMSON, C. L. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking 5* (1997), 631–645.

[2] BARFORD, P., AND CROVELLA, M. Generating representative web workloads for network and server performance evaluation. *SIGMETRICS Perform. Eval. Rev. 26*, 1 (June 1998), 151–160.

[3] CALZAROSSA, M., MASSARI, L., AND TESSERA, D. Workload characterization issues and methodologies. In *Proc. of Performance Evaluation: Origins and Directions* (2000), pp. 459–481.

[4] CALZAROSSA, M., AND SERAZZI, G. Workload characterization: A survey. In *Proc. of the IEEE* (1993), pp. 1136–1150.

[5] CHAPIN, S. J., CIRNE, W., FEITELSON, D. G., JONES, J. P., LEUTENEGGER, S. T., SCHWIEGELSHOHN, U., SMITH, W., AND TALBY, D. Benchmarks and standards for the evaluation of parallel job schedulers. In *Proc. of JSSPP'99*, pp. 67–90.

[6] CHEN, P. M., AND PATTERSON, D. A. A new approach to I/O performance evaluation: self-scaling I/O benchmarks, predicted I/O performance. *ACM Trans. Comput. Syst.* (1994), 308–339.

[7] CHEN, Y., GANAPATHI, A. S., GRIFFITH, R., AND KATZ, R. H. Analysis and lessons from a publicly available google cluster trace. Tech. Rep. UCB/EECS-2010-95, EECS Department, University of California, Berkeley, Jun 2010.

[8] CIRNE, W., AND BERMAN, F. A comprehensive model of the supercomputer workload. In *Proc. of WWC'01*, pp. 140–148.

[9] Condor. `http://www.cs.wisc.edu/condor/`.

[10] Amazon EC2. `http://aws.amazon.com/ec2/`.

[11] Emulab. `http://www.emulab.net/`.

[12] Emulab trace. `http://www.emulab.net/downloads/topfiles/topfiles-anonymized-2007-06-18.tar.gz`.

[13] Eucalyputus Cloud. `http://www.eucalyptus.com/`.

[14] FU, Y., CHASE, J., CHUN, B., SCHWAB, S., AND VAHDAT, A. SHARP: an architecture for secure resource peering. In *Proc. of SOSP'03*, pp. 133–148.

[15] GANAPATHI, A. S., CHEN, Y., FOX, A., KATZ, R. H., AND PATTERSON, D. A. Statistics-driven workload modeling for the cloud. Tech. Rep. UCB/EECS-2009-160, EECS Department, University of California, Berkeley, Nov 2009.

[16] GENI. `http://www.geni.net/`.

[17] Globus. `http://www.globus.org`.

[18] JAIN, R. K. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, New York, NY, USA, April 1991.

[19] KHAN, A., YAN, X., TAO, S., AND ANEROUSIS, N. Workload characterization and prediction in the cloud: A multiple time series approach. In *Proc. of Cloudman'12*.

[20] MIRKOVIC, J., HUSSAIN, A., AND SHI, H. A comparative study of network testbed usage characteristics. Under submission, USC/Information Sciences Institute, 2012.

[21] MISHRA, A. K., HELLERSTEIN, J. L., CIRNE, W., AND DAS, C. R. Towards characterizing cloud backend workloads: insights from google compute clusters. *SIGMETRICS Perform. Eval. Rev. 37*, 4 (Mar. 2010), 34–41.

[22] ORCA/BEN: A prototype GENI control plane for a metro-scale optical testbed. `https://geni-orca.renci.org/trac/`.

[23] PlanetLab. `http://www.planet-lab.org/`.

[24] ProtoGENI. `http://www.protogeni.net/trac/protogeni`.

[25] Sirius calendar service. `http://www.planet-lab.org/node/114`.

[26] SMITH, J., GOLDEN, P., AND APPLETON, B. *Airline: a strategic management simulation*. Prentice Hall, 1991.

[27] SONG, B., ERNEMANN, C., AND YAHYAPOUR, R. Parallel computer workload modeling with markov chains. In *Proc. of JSSPP'04*, pp. 47–62.

[28] SOTOMAYOR, B., KEAHEY, K., AND FOSTER, I. Combining batch execution and leasing using virtual machines. In *Proc. of HPDC'08*, pp. 87–96.

[29] TPC - Transaction Processing Performance Council. `http://www.tpc.org/`.

[30] VINI. `http://www.vini-veritas.net/`.

[31] WANG, G., BUTT, A. R., MONTI, H., AND GUPTA, K. Towards synthesizing realistic workload traces for studying the hadoop ecosystem. In *Proc. of MASCOTS'11*, pp. 400–408.

[32] YIN, Q., AND ROSCOE, T. A better way to negotiate for testbed resources. In *Proc. of APSys'11*, pp. 19:1–19:5.

[33] YIN, Q., AND ROSCOE, T. Vf2x: Fast, efficient virtual network mapping for real testbed workloads. In *Proc. of TridentCom'12*.