Hardness of Approximation and Greedy Algorithms for the Adaptation Problem in Virtual Environments

Ananth I. Sundararaj Manan Sanghi John R. Lange Peter A. Dinda
{ais,manan,jarusl,pdinda}@cs.northwestern.edu
Department of Electrical Engineering and Computer Science
Northwestern University

***Abstract*—**
Over the past decade, wide-area distributed computing has emerged as a powerful computing paradigm. Virtual machines greatly simplify wide-area distributed computing by lowering the abstraction to benefit both resource users and providers. A virtual execution environment consisting of virtual machines (VMs) interconnected with virtual networks provides opportunities to dynamically optimize, at run-time, the performance of existing, unmodified distributed applications without any user or programmer intervention. We have formalized the adaptation problem in virtual execution environments, and shown that it is NP-hard to both, solve and approximate within a factor of $m^{1/2-\delta}$ for any $\delta > 0$, where m is the number of edges in the virtual overlay graph. We also designed and evaluated greedy adaptation algorithms and found them to work well in practice.

I. Introduction

We have been developing a middleware system, Virtuoso, for virtual machine grid computing that, for a user, very closely emulates the existing process of buying, configuring, and using a computer or a collection of computers from a web site. Instead of a physical computer, the user receives a reference to the virtual machine which he can then use to start, stop, reset, and clone the machine.

The nature of the network presence that the virtual machine receives depends solely on the policies of the remote site. To deal with this network problem we developed VNET [2], a simple layer two virtual network tool. VNET is ideally placed to monitor the resource demands of the VMs. The VTTIF (Virtual Topology and Traffic Inference Framework) component of Virtuoso achieves this [2]. Parts of Virtuoso that are salient to this work are:

Measurement and inference: This involves (a) measuring the traffic load and communication topology of applications running inside the virtual machines, (b) monitoring the underlying network and inferring its topology, bandwidth and latency characteristics, and (c) measuring host and VM characteristics such as their size, compute capacities and demands. In previous work [1] we have shown how to successfully accomplish these tasks.

Adaptation mechanisms: A wide variety of adaptation mechanisms are possible in the context of virtual execution environments, such as (a) VM migration, (b) overlay topology and routing changes, and (c) network and CPU resource reservations. These have been described previously [2].

Adaptation algorithm: Most importantly, we need an efficient algorithm that drives the adaptation mechanisms and is guided by the measured and inferred data.

II. Adaptation problem formulation

We describe a constrained version of the generic adaptation problem that includes only the routing and mapping components. VNET monitors the underlying network and provides a directed VNET topology graph, $G = (H,E)$, where H are VNET nodes (hosts running VNET daemons and capable of supporting one or more VMs) and E are the possible VNET links. Wren (integrated with VNET [1]) provides estimates for the available bandwidth over each link in the VNET topology graph. These estimates are described by a bandwidth capacity function, $\text{bw} : E \rightarrow \mathbb{R}$.

We are also given an initial mapping of virtual machines to hosts, $\mathcal{M}$, which is a set of 3-tuples, $M_i = (\text{vm}_i, h_i, y_i)$, $i = 1,2\ldots n$, where $\text{vm}_i \in \text{VM}$ is the virtual machine in question, $h_i \in H$ is the host that it is currently mapped onto and $y_i \in \{0,1\}$ specifies whether the current mapping of VM to host can be changed or not. A value of 0 implies that the VM can be remapped and a value of 1 implies that it cannot.

VTTIF infers the application communication topology in order to generate the traffic requirements of the application, $\mathcal{A}$, which is a set of 3-tuples, $A_i = (s_i, d_i, b_i)$, $i = 1,2\ldots m$, where s_i is the source VM, d_i is the destination VM, b_i is the bandwidth demand between the source destination pair.

The goal is to find an adaptation algorithm that uses the measured and inferred data and drives the adaptation mechanisms at hand to improve application throughput. We wish to find

- a mapping from VMs to hosts, vmap : $VM \to H$, meeting the specified constraints.
- a routing, $R : \mathcal{A} \to \mathcal{P}$, where $\mathcal{P}$ is the set of all paths in the graph $G = (H,E)$, i.e. for every 3-tuple, $A_i = (s_i, d_i, b_i)$, allocate a path, $p\big(\text{vmap}(s_i), \text{vmap}(d_i)\big)$, over the overlay graph, G, meeting the application demands while satisfying the bandwidth constraints of the network.

Once the mappings and paths have been decided, each VNET edge will have a residual capacity, rc_e, which is the bandwidth remaining unutilized on that directional edge. We define $\text{rc}_e = \text{bw}_e - \sum_{e \in R(A_i)} b_i$.

For each mapped path, $R(A_i)$, we define its bottleneck bandwidth, $\text{bb}\big(R(A_i)\big) = \min_{e \in R(A_i)} \big\{\text{rc}_e\big\}$. The aim of the adaptation algorithm is to maximize the sum of residual bottleneck bandwidths over each mapped path.

Problem 1 (Mapping and Routing Problem In Virtual Execution Environments (MARPVEE))
INPUT:

- A directed graph $G = (H,E)$
- A function bw : $E \to \mathbb{R}$
- A set, $\text{VM} = (\text{vm}_1, \text{vm}_2 \ldots \text{vm}_n)$, $n \in \mathbb{N}$
- A set of ordered 3-tuples $\mathcal{A} = \{(s_i, d_i, b_i) \mid s_i, d_i \in \text{VM};\ b_i;\ i = 1, \ldots, m\}$
- A set of ordered 3-tuples $\mathcal{M} = \{(\text{vm}_i, h_i, y_i) \mid \text{vm}_i \in \text{VM};\ h_i \in H;\ y_i \in \{0,1\};\ i = 1, \ldots, n\}$

OUTPUT: vmap : $VM \to H$ and $R : \mathcal{A} \to \mathcal{P}$ such that

- $h_i = \text{vmap}(\text{vm}_i) \quad \forall\, M_i = (\text{vm}_i, h_i) \in \mathcal{M}$ if $y_i = 1$
- $\text{rc}_e \geq 0,\ \forall e \in E$
- $\sum_{i=1}^{m} \big(\min_{e \in R(A_i)} \big\{\text{rc}_e\big\}\big)$, where $\text{rc}_e = (\text{bw}_e - \sum_{e \in R(A_i)} b_i)$, is maximized

III. Computational complexity of the adaptation problem

Theorem 1: MARPVEED (decision version) is NP-complete.

The NP-hardness for this problem is established by reduction from the Edge Disjoint Path Problem (EDPP) which is shown to be NP-complete.

We also used EDPP to investigate the approximability of MARPVEE. If m is the number of edges in the virtual overlay graph we have the following result.

Theorem 2: For any $\delta > 0$, it is NP-hard to approximate MARPVEE within $m^{1/2-\delta}$ unless P=NP.

IV. Greedy adaptation algorithms

We have devised two greedy algorithms for mapping VMs to hosts. One finds all the mappings in a single pass (GreedyMapOne), while the other takes two passes over the input data (GreedyMapTwo). We have also adapted Dijkstra's shortest path algorithm that now finds the widest path for an unsplittable network flow. Since MARPVEE involves both, mapping and routing network flows we can first apply the mapping algorithm (either one) followed by the routing algorithm (GreedyRouting). Alternatively we can interleave the two.

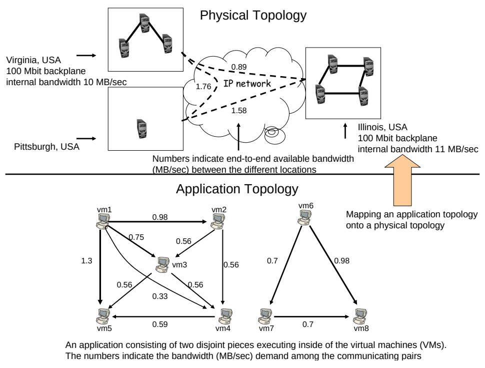
Physical Topology
Virginia, USA
100 Mbit backplane
internal bandwidth 10 MB/sec
Pittsburgh, USA
0.89
1.76
IP network
1.58
Numbers indicate end-to-end available bandwidth (MB/sec) between the different locations
Illinois, USA
100 Mbit backplane
internal bandwidth 11 MB/sec
Application Topology
vm1
vm2
vm6
0.98
0.75
0.56
1.3
vm3
0.56
0.7
0.98
0.56
0.56
0.33
vm5
0.59
vm4
vm7
0.7
vm8
Mapping an application topology onto a physical topology
An application consisting of two disjoint pieces executing inside of the virtual machines (VMs).
The numbers indicate the bandwidth (MB/sec) demand among the communicating pairs

Fig. 1. Experimental setup.

We implemented an evaluator that was used to calculate the residual bandwidth for multiple test cases. We evaluated the four algorithm variations in three different settings, randomly generated topologies using BRITE, smaller topologies created by hand and a real world scenario. Figure 1 illustrates our setup. All the algorithms were found to perform well in most scenarios. For randomly generated topologies we do not see any differences between the different variations. However for the topology created by hand and for the real world scenario that result in a clustered setting, the one-pass variation outperforms the two-pass algorithm. Further, we did not notice in difference between the interleaved and non-interleaved variations.

V. Conclusion

We formalized the adaptation problem that arises in virtual execution environments environments. We have shown that the adaptation problem is NP-hard and proven that it is NP-hard to approximate within a factor of $m^{1/2-\delta}$ for any $\delta > 0$, where m is the number of edges in the virtual overlay graph. We evaluated a variety of greedy algorithms and found them to perform well in practice. We are currently focusing on researching the feasibility of a single optimization metric that would be effective for a range of distributed applications.

References

[1] A. Gupta, M. Zangrilli, A. I. Sundararaj, A. Huang, P. Dinda, and B. Lowekamp. Free network measurment for adaptive virtualized distributed computing. In *Proceedings of IPDPS*, April 2006. To Appear.

[2] A. I. Sundararaj, A. Gupta, and P. A. Dinda. Increasing application performance in virtual environments through run-time inference and adaptation. In *Proceedings of HPDC*, July 2005.