BlockQoS: Storage-aware placement of distributed deployments in data centers.

Bart Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

Homer Simpson
Twentieth Century Fox
Springfield, USA
Email: homer@thesimpsons.com

James Kirk
and Montgomery Scott
Starfleet Academy
San Francisco, California 96678-2391
Telephone: (800) 555–1212
Fax: (888) 555–1212

***Abstract*—[abstract]**

I. Introduction

II. Cloud Admission

A. Example

The provisioning of topologies from Infrastructure service with unknown QoS result in deployments that prevent or conflict with matching application requirements.

- Performance requirements are not met: heterogeneous performance characteristics for different VM or performance degradation for single VM for synchronous hypervisor replication (backup)
- Durability: Replicas are stored on the same physical entity. VMs on different physical machines run process that store redundant data for durability. Virtual volumes are placed on the same physical storage device.
- Availability: Shared network

B. Cloud admission state of the art

Cloud admission vs. reallocation and migration.

- Admission: Goals is low rejection rate, fast placement.
- Migration/reallocation: Goal is convergence against load balanced state under minimum migration costs. Reallocation is migration under available service.

C. Cloud admission development methodology

Roles

- Virtual Entity Engineer: Builds abstraction of data center that results in data center model. Data center model abstracts a data center into physical entities with resources and resource capacities and deployment options for virtual entities. Identifies resource feature interactions.
- Topology Engineer: D
- Topology requester:

III. Data Center and Application Model

A. Data Center Model

A *data center model* (DCM) represents an abstraction of a data center (DC). A DCM describes a set of *physical entities* and their communication capabilities. *Virtual entities* are assigned (placed) to physical entities during *placement*, *migration* and *reallocation*. A graphical representation of a DCM is a non-directional graph $G^d = (V^d, E^d)$ with verticies $\{v_i^d \mid v_i^d \in V^d \wedge 1 \leq i \leq \left|V^d\right|\}$ and edges $\{e_i^d \mid e_i^d \in E^d \wedge 1 \leq i \leq \left|E^d\right|\}$ with V^d denoting the set of physical entities and E^d denoting the set of *communication capabilities* between physical entities. We use the following notations to reference edges: $e^d = (v_i^d, v_j^d)$. We use the notation pel_i to refer to an element of the unified set of verticies and edges $\{pel_i \mid pel_i \in V^d \cup E^d \wedge 1 \leq i \leq \left|V^d \cup E^d\right|\}$.

$$G^d = (V^d, E^d) \tag{1}$$

$$\{v_i^d \mid v_i^d \in V^d \wedge 1 \leq i \leq \left|V^d\right|\} \tag{2}$$

$$\{e_i^d \mid e_i^d \in E^d \wedge 1 \leq i \leq \left|E^d\right|\} \tag{3}$$

$$e^d = (v_i^d, v_j^d) \tag{4}$$

$$\{pel_i \mid pel_i \in V^d \cup E^d \wedge 1 \leq i \leq \left|V^d \cup E^d\right|\} \tag{5}$$

1) Physical Entities: We distinguish between three sets of physical entities: (i) *physical machines* (PMs), (ii) *physical storage locations* (PSs) and (iii) *physical switches* (PWs). We refer to a *physical machine* by $pm_i \in PM$ with PM denoting the set of physical machines and to a *physical storage location* by $ps_i \in PS$ with PS denoting the set of physical storage locations. A physical storage location can represent a single storage device, e.g., HDD or SSD, or a managed storage appliance of multiple storage devices. We refer to a *physical switch* (switch) by $pw_i \in PW$ with PW denoting the set of switches in the DC.

$$V^d = PM \cup PS \cup PW \tag{6}$$

$$PM = \{pm_i \mid 1 \leq i \leq |PM|\} \tag{7}$$

$$PS = \{ps_i \mid 1 \leq i \leq |PS|\} \tag{8}$$

$$PW = \{pw_i \mid 1 \leq i \leq |PW|\} \tag{9}$$

2) Physical network: Physical links represent the physical network that connects physical entities. We refer to a physical link that connects two physical entities by a communication capability. If a physical link connects a PM and a PS, the PS represents a disk that is locally attached to a PM. We assume that two PMs or two PSs connect to each other through a physical switch.

$$((v_i^d, v_j^d) \mid v_i^d \in PM) \rightarrow v_j^d \notin PM \tag{10}$$

$$((v_i^d, v_j^d) \mid v_i^d \in PS) \rightarrow v_j^d \notin PS \tag{11}$$

We refer to a sequence of verticies in which all elements besides the first vertice v_i^d and the last verticie v_j^d are PWs and two following elements are connected through physical links as *communication path* $CP_{v_i^d, v_j^d}$.

$$\begin{aligned} CP_{v_i^d, v_j^d} = ((a_k)_{k=1}^{l} \mid 2 \leq l \leq |PW| + 2 \wedge \\ \{a_1 = v_i^d, a_l = v_j^d\} \subseteq V^d \wedge \\ \{(a_k)_{k=1}^{l}\} \cap \{a_1, a_l\} \subseteq SW \wedge \\ \{(a_m, a_{m+1}) \in E^d \mid 1 \leq m \leq l-1\}) \end{aligned} \tag{12}$$

3) Resource and property offers: Physical entities and physical links offer resources to and determine properties of virtual entities. Resources are limited by a capacity and consumed by usage while properties are not. Virtual entities consume resources of the physical entities they are placed on. A *resource offer* has a resource type $rt_i \in RT$ with RT denoting the set of resource types.

$$RT = \{rt_i \mid 1 \leq i \leq |RT|\} \tag{13}$$

We refer to a resource offer $ro_{pel_i}(rt_j) \in RO_{pel_i}$ by a resource type rt_j with pel_i denoting a physical entity or link and RO_{pel_i} denoting the set of resources offers of pel_i.

$$RO_{pel_i} = \{ro_{pel_i}(rt_j) | 1 \leq j \leq |RO_{pel_i}|\} \tag{14}$$

A resource offer has a capacity $c(ro_{pel_i}(rt_j))$. Examples of resources offers and metrics are:

- PM: number of cores, memory in GB
- PS: storage size in GB, random/sequential throughput in IOPS/Mb per s
- PL: bandwidth in Mb
- PS: bandwidth in Mb

A *property offer* characterizes a physical entity. Similar to a resource offer, a property offer has a property type. However, properties do not have a capacity that is consumed by placing a virtual entity on a physical entity. We refer to a property offer $po_{pel_i}(pt_j) \in PO_{pel_i}$ by a property type pt_j with pel_i denoting a physical entity or link and PO_{pel_i} denoting the set of property offers of pel_i. A property type defines the domain of valid values for a resource offer.

$$PT = \{pt_i \mid 1 \leq i \leq |PT|\} \tag{15}$$

$$PO_{pel_i} = \{po_{pel_i}(pt_j) \mid 1 \leq j \leq |PO_{pel_i}|\} \tag{16}$$

Examples of properties and their domain for different physical entities are:

- pm: physical location [?] [2DO: ref], processor type [intel, AMD] [2DO: ref]
- ps: disk type [HDD, SSD] [2DO: ref]
- pw:

B. Application Model

An *application model* (AM) specifies functional and non-functional requirements of an application with regards to infrastructure that a Cloud provider provisions for the deployment of an application. A graphical representation of an AM is a non-directional graph $G^a = (V^a, E^a)$ with verticies $\{v_i^a \mid v_i^a \in V^a \wedge 1 \leq i \leq |V^a|\}$ and edges $\{e_i^a \mid e_i^a \in E^a \wedge 1 \leq i \leq |E^a|\}$ with V^a denoting the set of virtual entities and E^a denoting the set of *communication requirements* between virtual entities. We use the notation vel_i to refer to an element of the unified set of verticies and edges $\{vel_i \mid vel_i \in V^a \cup E^a \wedge 1 \leq i \leq |V^a \cup E^a|\}$.

1) Virtual entities and network: We distinguish between two sets of virtual entities in V^{app}: (i) *virtual machines* and (ii) *virtual volumes*. We refer to a virtual machine by $vm_i \in VM$ with VM denoting the set of virtual machines and to a virtual volume by $vv_i \in VV$ with VV denoting the set of virtual volumes.

$$VM = \{vm_i \mid 1 \leq i \leq |VM|\} \tag{17}$$

$$VV = \{vv_i \mid 1 \leq i \leq |VV|\} \tag{18}$$

We express communication requirements between tupels of virtual entities by edges $e^a = (v_i^a, v_j^a)$.

2) Resource and property demands: Virtual entities and communication demands express functional and non-functional requirements of an application deployment. For example, functional requirements of a virtual volume are capabilities for snapshots or backups. Non-functional requirements are, e.g., storage capacity and performance requirements. Functional requirements are captured by properties and non-functional requirements by resources demands or properties.

Functional requirements of a vel_i are expressed by *property demands*. We refer to a property demand by $pd_{vel_i}(rt_j) \in PD_{vel_i}$ with PD_{vel_i} denoting the set of property demands of vel_i.

$$PD_{vel_i} = \{pd_{vel_i}(pt_j) \mid 1 \leq j \leq |PD_{vel_i}|\} \tag{19}$$

Non-functional requirements of a vel_i are expressed by property demands or *resource demands*. We refer to the resource demand of vel_i for a resource of resource type rt_j by $rd_{vel_i}(rt_j) \in RD_{vel_i}$ with RD_{vel_i} denoting the set of resource demands of vel_i.

$$RD_{vel_i} = \{rd_{vel_i}(rt_j) | 1 \leq j \leq |RD_{vel_i}|\} \tag{20}$$

IV. Placement - Mapping rules

A feasible solution to a placement problem requires to find an admissible mapping of an instance of an application model G^{app} to an instance of a data center model G^{dc}. Therefore, virtual entities in an AM are assigned to physical entities in a DM. We present a set of mapping rules to constraint the placement problem.

We refer to a vel_i that is mapped to a subset of physical entities $H_j \subseteq V^{dc}$ as *guest* and to H_j as *host* of vel_i, respectively. We use the notation $\phi(vel_i, H_j)$ to express an existing mapping relationship between a guest vel_i and a set of hosts $(H_j \mid j \in N \wedge |H_j| \leq |V^d|)$. We present a number of mapping rules in form of implications that constraint admissible mappings of a vel_i to match application requirements.

A. Generic mapping of virtual entities

Entity placement constraints restrict feasible mappings for single virtual entities to physical entities. Virtual machines are guests of physical machines, virtual volumes are guests of physical storage locations and virtual links are guests of a communication paths.

$$vm_i \rightarrow \phi(vm_i, pm_j) \tag{21}$$

$$vv_i \rightarrow (\phi(vv_i, H_j) \mid H_j \subseteq PS) \tag{22}$$

Furthermore, two virtual entities v_i^a, v_j^a with communication demands are guests of physical entities that are connected.

$$(v_i^a, v_j^a) \rightarrow (\phi(v_i^a, v_k^d) \wedge \phi(v_j^a, v_l^d) \mid CP(v_k^d, v_l^d)) \tag{23}$$

A virtual entity vel_i with a property demand $pd_{vel_i}(pt_j)$ is guest of physical entities that each provide a property offer of the same property type pt_j.

$$pd_{vel_i}(pt_j) \rightarrow \phi(vel_i, \{pel_k \mid po_{pel_k}(pt_j)\}) \tag{24}$$

A virtual entity vel_i with a resource demand $rd_{vel_i}(rt_j)$ is guest of physical entities with a resource offer of the same resource type $ro_{pel_k}(rt_j)$.

$$rd_{vel_i}(rt_j)) \rightarrow \phi(vel_i, \{pel_k \mid ro_{pel_k}(rt_j)\}) \tag{25}$$

The aggregated resource demand of all guests for each resource offer of a host does not exceed the resource capacity of the resource offer. A guests with a resource demand of a resource type are mapped to hosts that provide enough capacity of the same resource type for the aggregated demand of all their guests.

$$\begin{aligned} rd_{vel_i}(rt_j) \rightarrow (\phi(vel_i, pel_k) \mid \\ c(ro_{pel_k}(rt_j)) \geq (\sum_{l \in \{vel_l \mid \phi(vel_l, pel_k)\}} rd_{vel_l}(rt_j)) \end{aligned} \tag{26}$$

B. Specific mapping of virtual volumes

We present a set of storage specific mapping rules to account for storage specific application requirements in placement decisions.

1) Replication: We assume that virtual volumes can be replicated with a *replication factor* of $rf(vv_i) = \alpha \in \mathrm{N}$. The set of *virtual volume replicas* (replicas) of the virtual volume vv_i is defined by $(R_{vv_i} \mid 1 \leq |R_{vv_i}| \leq \alpha)$. A replicated virtual volume is guest of a set of physical storage locations. We assume that multiple replicas are guests of different physical storage locations.

$$rf(vv_i) = \alpha \rightarrow (\phi(vv_i, \{ps_j\}) \mid |\{ps_j\}| = \alpha) \tag{27}$$

2) Property offer similarity: We assume that an operating system of a virtual machine vm_i observers a write performance for a mounted virtual volume vv_j that synchronously propagates updates to all replicas in R_{vv_j} through hypervisor-level replication. We further assume that replicas are guests of physical storage locations with different write performance characteristics, e.g., through the usage of SSD and HDD [**?**]. In this case, the physical storage location with the lower write performance can degrade the write performance observed by vm_i. We propose to account for such cases by enabling the placement of a set of replicas R_{vv_j} on physical storage locations with similar performance characteristics. We assume that the performance characteristics of a physical storage location is continuously captured by the Cloud provider in a property offer. Examples of recent approaches to capture such performance characteristics are presented in [**?**], [**?**], [**?**]. The provisioning of multiple virtual volumes with similar storage performance characteristics for multiple virtual machines can simplify the management of a distributed application. Therefore, we allow the specification of a property requirement $r_{prop}(\{vv_i\}) = (\beta \mid \beta \in PT)$ for a set of virtual volumes $\{vv_i\}$. A property requirement ensures that all virtual volumes in $\{vv_i\}$ are placed on physical entities with the same property offer without specifying the property offer itself.

$$r_{prop}(\{vv_i\}) = \beta \rightarrow \{\phi(vv_i, \{ps_k \mid po_{ps_k}(\beta)\})\} \tag{28}$$

3) Physical storage colocation: We assume that replicas R_{vv_i} of a virtual volume vv_i are *colocated*, i.e., guests of the same physical storage location, or *anti-colocated*, i.e., guests of different physical storage locations. Cloud provider can benefit from transparent placement of colocated and anti-colocated replicas through increased opportunities for load-balancing and data center-wide reduction of bandwidth usage. We argue that transparent co-location of replicas can degrade the durability of a virtual volume. A virtual volume vv_i can specify a colocation requirement $r_{col}(vv_i) = \gamma$ with γ denoting the number of collocated replicas.

$$r_{col}(vv_i) = \gamma \rightarrow (\phi(vv_i, H_j) \mid \gamma \leq |H_j| \leq rf(vv_i)) \tag{29}$$

Furthermore, we account for the case that applications store data redundantly on multiple virtual volumes that are potentially not replicated and/or mounted by different virtual machines. Therefore, we allow to specify colocation requirements $r_{col}(\{vv_i\}) = \gamma$ on a set of virtual volumes.

$$r_{col}(\{vv_i\}) = \gamma \rightarrow \{\phi(vv_i, H_j) \mid \gamma \leq |\cap H_j|\} \tag{30}$$

4) Communication path colocation: We account for the case that a virtual machine mounts multiple virtual volumes to increase the availability of block storage accessible from the operating system. Therefore, we try to increase the *overall availability* of a number of mounted block storage volumes visible to the operating system of a virtual machine during placement. We assume that the overall availability degrades with overlapping communication paths from the host of a VM to the hosts of mounted virtual volumes. We measure the degree of overlap in terms of shared switches on a set of communication paths. We allow the specification of an *availability requirement* $r_{avb}(vm_i, \{vv_j\}) = \delta$ for a virtual machine vm_i and a set of virtual volumes $\{vv_j \mid (vm_i, vv_j)\}$. The availability requirement β describes an upper bound for the number of shared switches on the set of communication paths that connect the host of vm_i with the hosts of $\{vv_j\}$.

$$r_{avb}(vm_i, \{vv_j \mid (vm_i, vv_j)\}) = \delta \tag{31}$$

$$r_{avb}(vm_i, \{vv_j\}) = \delta \rightarrow \delta \leq |\{sw_k \mid sw_k \in (CP_{pm_l,ps_m} \mid \sigma(vm_i, pm_l) \wedge \sigma(vv_j, \{ps_m\})\}| \tag{32}$$

V. Placement - Optimization Problem

A. Preferences

Application Expert preferences

- Functional requirements
- Non-functional requirements

Cloud provider preferences

- Load-balancing
- Rejection rate
- Efficient resource usage

B. Objective Function

VI. Algorithm

[2Do: Algorithm goes here]

VII. Evaluation

[2Do: Evaluation goes here]

A. Implementation

B. Performance

- Evaluate solver time for standard data center sizes

C. Scalability

VIII. Conclusion