Software Bundling Selection for Cloud Virtual Machine Images

Marco A. S. Netto, Marcos D. Assunção
Lakshminarayanan Renganarayana, Chris Young

IBM Research

{marcosda, mstelmar}@br.ibm.com, {lrengan,ccyoung}@us.ibm.com

***Abstract*—Organisations and end-users are increasingly using Cloud resources to take advantage of the anticipated benefits of a more cost effective and agile IT infrastructure. Virtual machines are provisioned based on a selection of available images, which often contain the operating system and the software stack required by applications. When managing an image library, some of the challenges faced by a resource provider include (i) identifying the optimal number of virtual machine images that satisfy most user requirements, and (ii) bundling software systems into images to minimise the time to provision virtual machines and ease the selection of images from an end-user's perspective. Using a traditional data centre workload, this paper proposes an algorithm for selecting software bundles for virtual machine images and examines the impact of bundle selection on the number and characteristics of resulting images. The main finding is that creating a small set of virtual machine images packed with the most popular software systems is enough to drastically reduce the time to deploy the software stack required by applications, and hence minimise the time for provisioning virtual servers in a Cloud infrastructure.**

I. Introduction

Under a cloud [1] model that delivers Infrastructure as a Service (IaaS), virtual machines are provisioned from a selection of available virtual machine images, which often contain the operating system and the software stack required by user applications. Users can start from basic images offered by the cloud provider and customise them to their needs or, in certain cases, upload their own images to the cloud. Although over time this model allows for users to select from a large pool of images and reuse images that better match their requirements, it can easily lead to image sprawl, which may exacerbate the problem of selecting suitable images [2].

The ability for a customer to rapidly deploy new workloads via new cloud instances is a key selling point of cloud and virtualisation. However, for this to be a reality customers must find an appropriate set of images that meet the minimum requirements of their application workload; including non-functional requirements such as speed, robustness, and scalability. This selection process is largely manual and expects customers to tediously iterate through many choices to find the one most suitable for their needs.

Existing solutions attempt to address the challenge of finding suitable virtual machine images by providing means to annotate existing images and offering mechanisms for image discovery [3], [4], [5]. One alternative method is to define a small set of virtual machine images that have the software bundles that satisfy most application requirements. Limiting the choices to a well defined set of virtual machine images is key to standardisation and reducing the cost of managing the servers created from these images. The savings come from the extreme automation of server management enabled by the standardised set of images.

Our previous paper described a framework to assist the selection of *virtual resource templates*, focusing on resource attributes such as CPU, memory, and disk capabilities [6], and did not account for the software stack. This paper builds on previous work and, by using the workload of an existing data centre, proposes an image creation algorithm considering the software stack required by user applications. The workload represents an entire stack including the operating system, software, application, and the supporting non-functional requirements (*e.g.* ability to perform *x* transactions per second). We examined requests for hundreds of servers within a production enterprise data centre and used this information to derive bundles of software that should be pre-installed with the images to be offered by a cloud infrastructure. The main contributions of this work are:

- A software bundle selection algorithm for deriving a set of images required to satisfy the majority of user requirements;
- Characterisation of a production enterprise data centre workload considering software stack requirements;
- Evaluation of the software bundling algorithm and its impact on virtual machine provisioning time and the number of required virtual machine images.

Our evaluation shows that creating a small set of virtual machine images packed with the most popular software systems is enough to drastically reduce the time to deploy the software stack required by applications, thus minimising the time for provisioning Cloud virtual servers. The results, techniques, and ideas presented in this paper can be leveraged by researchers and practitioners working on image management. Such topic is particularly relevant for large data centres as provisioning of virtual machines depends on carefully created, tested, and validated images; and these steps at present may not be fully automated.

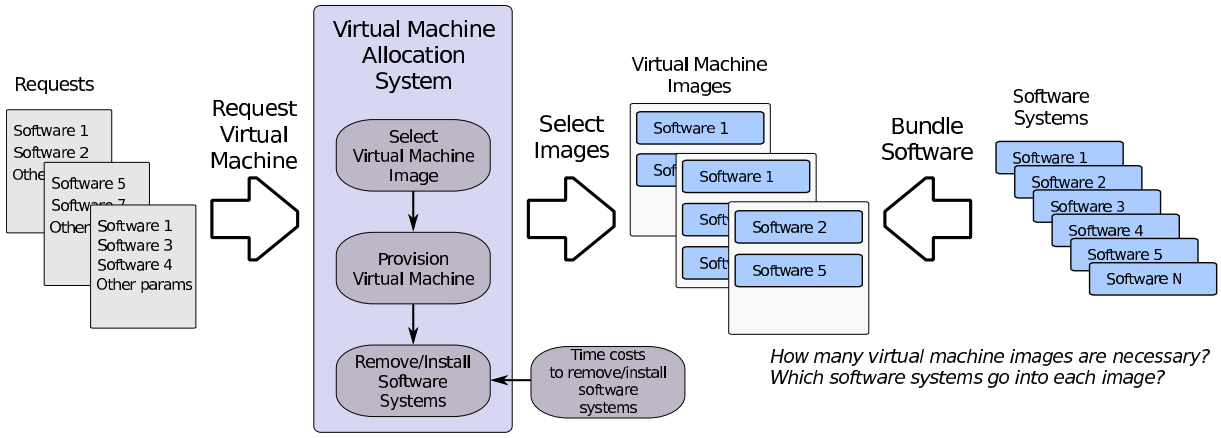


Fig. 1. Requests for virtual machines contain, among other parameters, the software stack that needs to be in the virtual machine. From the resource provider's perspective, the challenge is to identify the optimal number of VM images with the right software bundles.

II. Problem Description and Metrics

The ability to quickly create VMs with different software stacks is important for realising the elastic and on-demand capacity scaling promised by cloud computing. The open community has approached this problem by creating and sharing VM images, as in Amazon EC2's Amazon Machine Images (AMI). To provision VMs, users choose from these publicly available AMIs and add to or remove software from them to make them match their needs. Although this approach is attractive, it does not scale to large enterprises that need to create a large number of standardised VMs, which is the scenario we explore in this paper.

Figure 1 depicts a typical enterprise scenario where users request VMs by specifying, among other parameters, the required software stack to run their services and applications. The cloud provider then maps the requirements to the available images. The images may already contain a set of pre-defined software stack. Once an image is selected, a virtual machine is created from it and adapted (software packages are added or removed) to meet user requirements, and then deployed for use. Note that software systems may need to be removed not only to save disk space or to speed-up provisioning time, but also due to licensing issues.

The provider has a trade-off between having a large number of images that enable fast creation of new VMs versus the cost of supporting and maintaining them up-to-date. Solutions that help with this trade-off are essential for providers to maximise the reuse of images and minimise the cost of supporting them. Besides, such solutions can help users that have access to many images in the cloud. Such images, with appropriate software stacks are hereafter also termed as *bundles*.

The effectiveness of the proposed software bundling algorithm is measured with the following metrics:

- **Deployment time:** time to provision a virtual machine, including time to add or remove software packages;
- **Number of images:** number of images required to meet a given set of virtual machine requests, where a VM image is a combination of operating system together with a set of software bundles.

A. Workload overview

The data set used in this paper consists of 747 requests for enterprise server configurations, which include specifications for CPU, memory, disk, and most importantly, required software packages. All server requirements, collected over a period of three years, are based on the same operating system and hardware platform to rule out differences not attributable to the workload. In this section, we characterise the data set considering multiple dimensions, viz., (i) the software packages requested; (ii) number of requests per software; (iii) number of software packages per request; and (iv) pairs of dependencies between software systems.

We identified a set of 23 software packages (henceforth also referred as middleware M1,...,M23) in these server requests. Databases and application servers comprise more than 50% of the packages, which is a typical of enterprise workloads. In addition, more than 80% of the requests required only two software systems, 10% required three or four software systems, and the remaining requests required up to nine software systems. Four software systems were required by more than 200 requests. The remaing software systems were required by fewer than 50 requests in average.

Another interesting data relates to software dependencies. Figure 2 presents the number of requests that required each pair of middleware M1 to M23. Although we observe that most middleware pair dependencies have fewer than 20 requests, there is a small set of dependencies that have more than 60 requests. These dependencies must be investigated further in order to create the images with appropriated software bundles.

In addition to considering software dependencies, it is important to take into account the time required to install each software. The workload used shows that the time required to install a middleware varies from only a few seconds to over 2,500 seconds. Although this time does not seem much for certain deployment scenarios, one must consider that software deployment is only part of the process of provisioning a server. Software installation times over 600 seconds could impact negatively the deployment of applications that rely on the instantiation of virtual clusters with multiple workers [7].

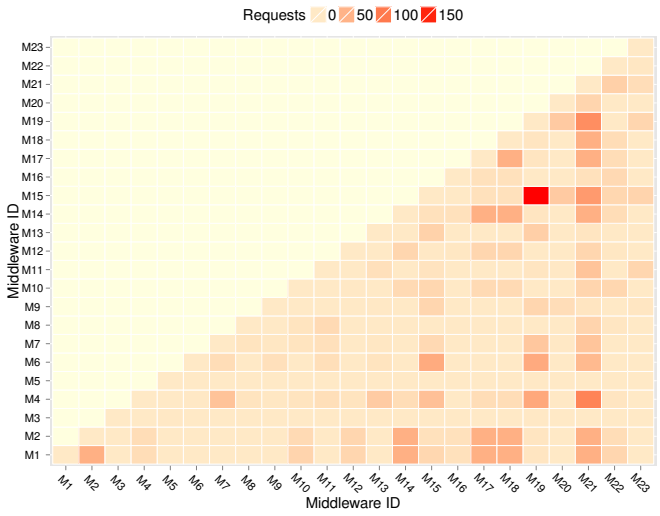


Fig. 2. Pairs of middleware depedencies.

III. Software Bundling Algorithm

The proposed software bundling algorithm uses a metric called *Software Utility* to model how important a given software is to a bundle. It is defined as:

$$SU_i = \sum_{j=1}^{ri} instTime(i) - \sum_{k=1}^{nri} remTime(i)$$

Where, ri is the number of requests that require software i, and nri is the number of requests that do not require software i. $instTime(i)$ and $remTime(i)$ are functions that represent the time for installation and removal of software i, respectively. Variable SU is used to measure how important is a software to be part of a bundle, considering the installation cost savings and the removal cost lost when the software is in a bundle.

Algorithm 1 describes the pseudo-code for creating the bundles. The main input parameters of the algorithm are:

- *reqList:* List of server requests, where each request has a list of software systems;
- *softList:* List of available software systems, where each software contains its installation time, removal time, and software utility value;
- *maxBundles:* Maximum number of bundles;
- *maxSoftSys:* Maximum number of software systems per bundle. Note that the algorithm can easily be modified to add other constraints such as maximum disk space or licenses used per bundle.

The algorithm starts by creating a list of empty bundles (Line 1) and then sorts the list of software systems by the decreasing order of their software utility values (Line 2). The initial best provisioning cost is set to the value of provisioning the requests using no bundles (Line 3). It then moves into a loop to create each software bundle (Line 4–19) considering the *maxSoftSys* value (Lines 5–19) to constrain the number of software systems for each bundle. A bundle creation involves iterating through the list of software systems (Lines 8–19), where each software system is included in the bundle (Line 11), testing whether the provisioning cost of the requests decreases (Line 12). If the cost is reduced, the algorithm keeps it in a temporary variable (Line 15) along with the software responsible for decreasing the cost (Line 16). In the end of the *softList* iteration, the best software and provisioning cost are stored (Lines 18–19). The algorithm iterates until all bundles are created, and finally the bundle set is returned (Line 20).

Algorithm 1: Pseudo-code for generating a bundle set.

```
   Input: reqList, softList, maxBundles, maxSoftSys
   Output: bundleList
1  bundleList ← createEmptyBundles(maxBundles)
2  Sort softList by decreasing order of SU
     // Evaluate with no bundles
3  bestCost ← evaluate(bundleList, reqList, softList)
4  foreach bundle in bundleList do
5      for k ← 1 to maxSoftSys do
6          bestSoftware ← null
             // Temporary best cost
7          tmpCost ← bestCost
8          foreach s in softList do
9              if bundle contains s then
10                 Ignore s for this bundle
11             bundle.addSoftware(s)
12             cost ← evaluate(bundleList, reqList,
               softList)
13             bundle.removeSoftware(s)
14             if cost < tmpCost then
15                 tmpCost ← cost
16                 bestSoftware ← s
17         if bestSoftware ≠ null then
18             bundle.addSoftware(bestSoftware)
19             bestCost ← tmpCost
20 return bundleList
```

IV. Evaluation

We evaluated the proposed software bundling algorithm using the workload from a production data centre as described in Section II-A. We developed a simulator to perform the bundling process and investigated the impact on software deployment time according to various input values and scenarios.

A. Setup and Input Parameters

The workload trace used in the experiments contains information about server requests, their software stack requirements and software installation and removal times. The simulation parameters include the number of bundles to meet the software requirements and the maximum number of software systems per bundle. In addition to analysing deployment time, we also show results on the characteristics of the software bundles when varying the set of input parameters.

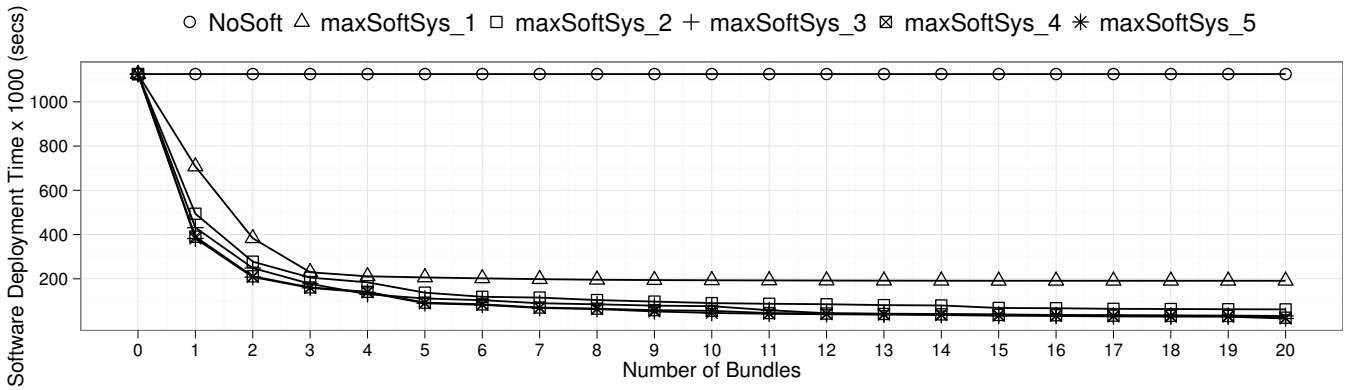


Fig. 3. Total software deployment time for all requests as a function of the maximum number of bundles (0 to 20 bundles) and maximum number of software systems per bundle (0 to 5 software systems).

B. Result Analysis

Figure 3 presents the main results on deployment time required to meet the software requirements of all requests in the workload. We also included an upper-bound to show the deployment time considering empty bundles (*NoSoft*). The number of bundles has higher impact on the deployment time than the number of software systems per bundle, with exception of *maxSoftSys_1*, which results in higher deployment time in most cases compared to the other maximum number of software systems per bundle. This happens because the overall deployment time is highly dependent on a set of software systems with high installation time but low inter-dependency, *i.e.* such software systems are not required together by the same requests. Therefore, including them separately into multiple bundles notably reduces the overall deployment time.

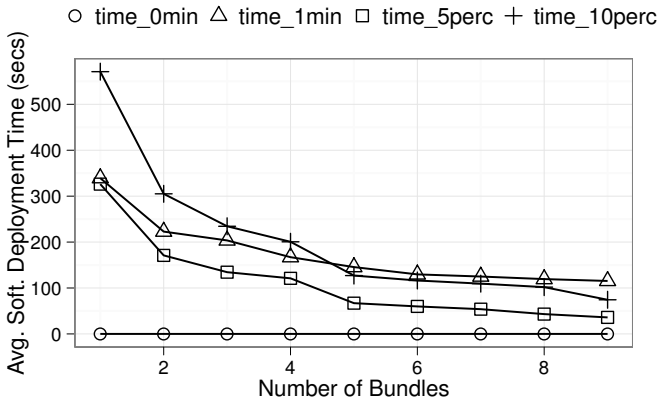
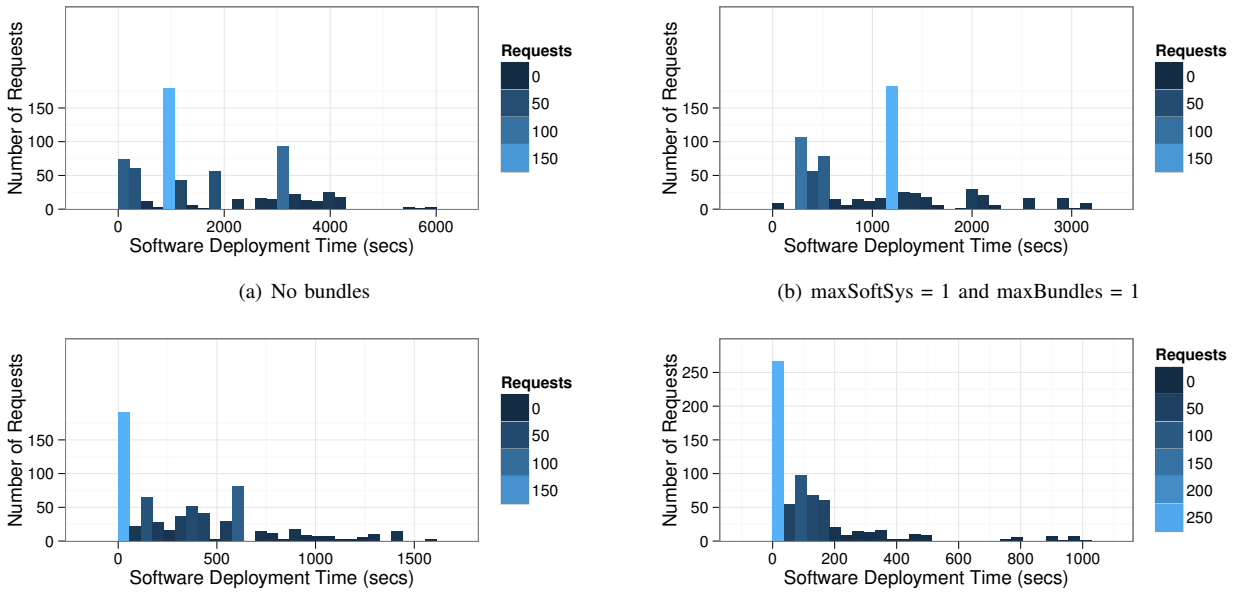


Fig. 4. Average software deployment time as a function of number of bundles and software removal time: 0 minutes, 1 minute, 5 percent of the installation time, and 10 percent of the installation time. The bundles can be configured with all software systems (*i.e.* maxSoftSys=23).

Another important result from Figure 3 is that after a certain threshold, increasing the number of bundles has little impact on software deployment time. This occurs because after a certain number of bundles, only requests with small software utility benefit from new bundles, as the first bundles benefit requests with high software utility. In addition, the impact of bundle configuration and number of bundles depends not only on the installation time but also on the software removal time. For this reason, we also evaluated the deployment time when modifying the software removal times, as illustrated in Figure 4. If removal time is zero, it makes sense to put all software systems into a single bundle so that the software deployment time for all requests is zero. However, by doing so, the image increases in size and the provisioning time of the virtual machine increases accordingly. The investigation on disk space constraints for software bundles requires further analysis, which we leave for future work.

By analysing the deployment time per request it is possible to observe how many requests can use the bundles with zero or close to zero software deployment time. Figure 5 (a) shows that there are very few requests with deployment time close to zero seconds and that the deployment time for the requests varies within a large range—from close to zero to approximately 6000 seconds. When we use a single bundle with a single software, although the number of requests with deployment time close to zero does not increase (it actually decreases), the requests with deployment time greater than 3000 are eliminated (Figure 5 (b)). Therefore, eliminating a single installation provides great benefits. If we go further, by including a few bundles and software systems into them, we observe a real shift towards no deployment time, as illustrated in Figures 5 (c) and (d). The parameters used for the cases presented in Figure 5 were selected based on the results and conclusions drawn from Figure 3.

Until now we have presented results for the number of bundles and software systems per bundle. Figures 6, 7, and 8 show the software bundles and software utility as a function of the number of bundles having maxSoftSys=1, maxSoftSys=3, and maxSoftSys=5 respectively. These figures how *how* these bundles look like. A general aspect of these figures is that software systems with high software utility tend to be present in the bundles since the number of bundles is small, and they remain in the bundles as the number of bundles increases. As the number of bundles increases, the bundling algorithm adds software systems with low software utility into the bundles. This is the expected behaviour due to the bundling algorithm



(a) No bundles

(b) maxSoftSys = 1 and maxBundles = 1

(c) maxSoftSys = 2 and maxBundles = 3

(d) maxSoftSys = 5 and maxBundles = 5

Fig. 5. Software deployment time for all requests and a subset of number of bundles and maximum software systems per bundle.

strategy of sorting the software systems by their decreasing value of the software utility values. However, there are two interesting aspects in the results of these figures.

The first interesting aspect of the bundling process illustrated in Figures 6, 7, and 8 is that there are software systems with negative utility that are bundled before software systems with positive utility. Intuitively, one would expect all software systems with positive utility values to be bundled before the software systems with negative utility values. But the opposite happens because the utility value relies on the installation and removal times considering that the bundle in which the software is placed is empty. As the bundles start to be filled with software systems, the utility evolves and therefore, software systems with low utility start to fit better in the bundles than software systems with high utility. Another reason is that, as the number of bundles increases, the software systems with low utility are used in isolated bundles, and these software systems do not reduce the quality of the early created bundles.

The second interesting aspect of the bundling process depends on the maximum number of software systems allowed per bundle. As we increase the number of bundles, the software systems do not remain in the same bundle. This can be seen already when maxSoftSys=5 (Figure 8). This follows the same explanation as above regarding the fact that the utility of the software systems evolves as the bundles are modified in the bundling process.

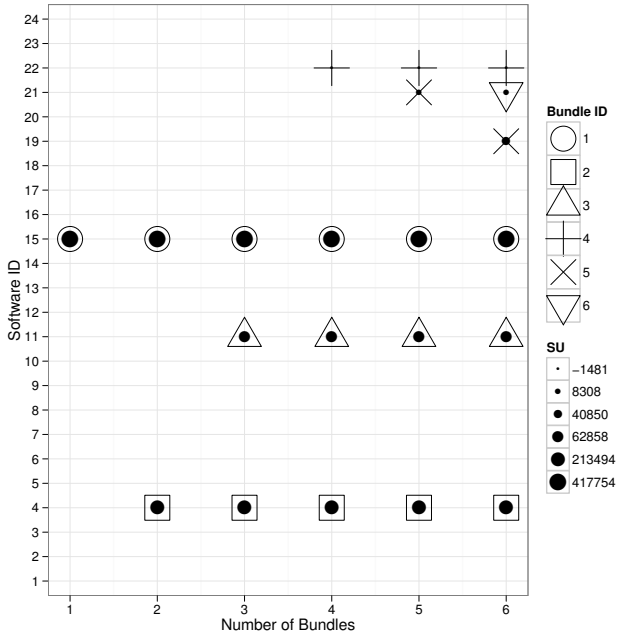


Fig. 6. Software bundles and software utility as a function of the number of bundles having maxSoftSys=1.

V. Related Work

Existing work has focused on minimising the time required by operations involved in the process of server provisioning; important factors in scenarios such as the creation of virtual clusters [7] and minimisation of energy consumption [8], [9]. For example, Snowflock provides a mechanism for fast VM cloning that allows for swift instantiation of workers on Cloud environments [10]. Ganguly *et al.* [11] propose techniques to minimise the hassle of system configuration by deploying virtual machines with pre-configured system components followed by the deployment of delta configurations. Sethi *et al.* [12] address a similar issue by providing a scheme that discovers and preserves software dependencies that are then stored along with virtual machine images and reused for later deployments. Wartel *et al.* [13] propose a mechanism for trust

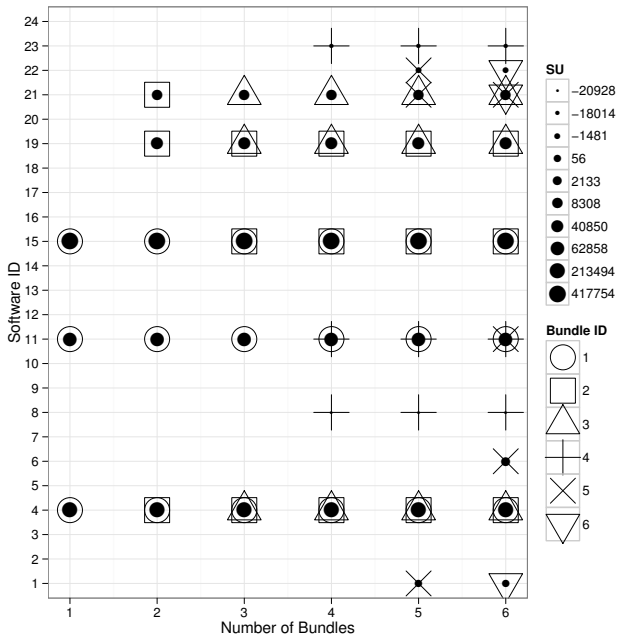


Fig. 7. Software bundles and software utility as a function of the number of bundles having maxSoftSys=3.

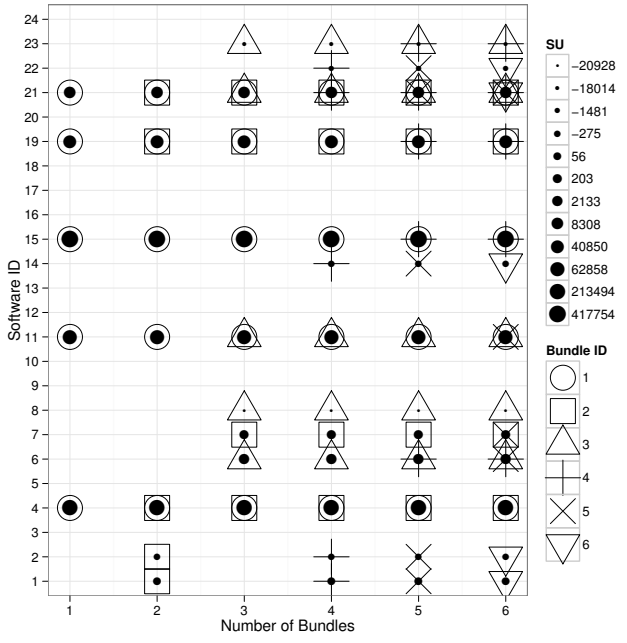


Fig. 8. Software bundles and software utility as a function of the number of bundles having maxSoftSys=5.

and distribution of virtual machine images across sites whereas Pfitzmann and Joukov [14] present and assess a technique for mapping existing IT systems to multi-image templates. Kochut and Karve [15] propose and evaluate a model that explores virtual machine image similarity to reduce the data volume transferred from a storage server to the hypervisor on which virtual machines are instantiated. Assunção *et al.* provide a technique for deriving VM templates that match the majority of user requests in a Cloud infrastructure [6].

The concept of software bundling predates the personal computer era [16]. The challenge of software bundling for virtual machines has been attacked mainly from two fronts: (i) by allowing users to pack the software they require into the virtual machine images and providing mechanisms for image discovery for new deployments based on these images; and (ii) by discovering the software bundles that satisfy most users and providing virtual machine images with these bundles. Filepp *et al.* [3] tackle the problem of choosing an image for a VM by selecting the image that yields the smallest software migration cost (*i.e.* the cost for installing the required software and removing software that is not necessary). The Mirage project attempts to provide an image library and means for searching and fastly retrieving images [5] and techniques to avoid image sprawl [2]. Dastjerdi *et al.* [4] propose a service that matches instances that users provide to appropriate offerings of cloud providers.

Striking a balance between number of software bundles and provisioning performance is important as image sprawl can raise several issues, including difficulties in software patching [17] and security [18]. Sapuntzakis *et al.* [19] attack the problem of patching software systems by using the concept of appliances dissociated from the software configuration and data. Appliances can be maintained and patched irrespective of the user data, which can later be applied to a patched appliance.

Our paper focuses on assisting system administrators in selecting the software stack to be included in virtual machine images, whereas existing work focuses mainly on searching images that can reduce provisioning time.

VI. Concluding Remarks

In this paper we studied the problem of selecting software bundles for creating virtual machine images. We introduced an algorithm that helps cloud providers and users in selecting the right number of virtual machine images with appropriate software bundles. In addition, using workload traces from a production data centre, we evaluated the impact of software bundles on the server provisioning time when varying parameters such as maximum number of bundles and maximum number of software systems per bundle.

One of the main findings of our work is that creating a small set of virtual machine images packed with the most popular software systems is enough to drastically reduce the time to deploy the software stack required by applications, and hence minimise the time to provision virtual machines in a Cloud. For instance, with a single bundle and a single software system, we were able to observe savings on total deployment time of approximately 40%. Reducing the server provisioning time leads to several advantages such as enabling faster deployment of applications that require virtual clusters with multiple workers and minimising peaks of power consumption during software installation.

As a future work, we intent to evaluate how the software bundling impacts the scheduling of virtual machines and the energy consumption of physical servers [9], [20].

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computing System*, vol. 25, no. 6, pp. 599–616, 2009.

[2] D. Reimer, A. Thomas, G. Ammons, T. Mummert, B. Alpern, and V. Bala, "Opening black boxes: Using semantic information to combat virtual machine image sprawl," in *Proceedings of ACM SIGPLAN/SIGOPS VEE*, 2008, pp. 111–120.

[3] R. Filepp, L. Shwartz, C. Ward, R. D. Kearney, K. Cheng, C. C. Young, and Y. Ghosheh, "Image selection as a service for cloud computing environments," in *Proceeding of the IEEE Service-Oriented Computing and Applications (SOCA'10)*, Perth, Australia, Dec. 2010, pp. 1–8.

[4] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, "An effective architecture for automated appliance management system applying ontology-based cloud discovery," in *IEEE/ACM CCGrid*, Melbourne, Australia, May 2010, pp. 104–112.

[5] G. Ammons, V. Bala, T. Mummert, D. Reimer, and X. Zhang, "Virtual machine images as structured data: the mirage image library," Proceedings of the USENIX HotCloud 2011, Jun. 2011.

[6] M. D. Assuncao, M. A. S. Netto, B. Peterson, L. Renganarayana, J. Rofrano, C. Ward, and C. Young, "Cloudaffinity: A framework for matching servers to cloudmates," in *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS'12)*, 2012.

[7] M. Stillwella, D. Schanzenbacha, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *Journal of Parallel and Distributed Computing*, vol. 70, no. 9, pp. 962–974, September 2010.

[8] G. Valentini, W. Lassonde, S. Khan, N. Min-Allah, S. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, pp. 1–13, 2011.

[9] H. Viswanathan, E. K. Lee, I. Rodero, D. Pompili, M. Parashar, and M. Gamell, "Energy-aware application-centric vm allocation for hpc workloads," in *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS'11)*, 2011.

[10] H. A. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, E. de Lara, M. Brudno, and M. Satyanarayanan, "Snowflock: rapid virtual machine cloning for cloud computing," in *Proceedings of the 4th ACM European conference on Computer systems (EuroSys'09)*. ACM, 2009, pp. 1–12.

[11] A. Ganguly, J. Yin, H. Shaikh, D. Chess, T. Eilem, R. Figueiredo, J. Hansom, A. Mohindra, and G. Pacifici, "Reducing complexity of software deployment with delta configuration," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*, 2007, pp. 729 –732.

[12] M. Sethi, K. Kannan, N. Sachindran, and M. Gupta, "Rapid deployment of soa solutions via automated image replication and reconfiguration," in *Proceedings of the IEEE International Conference on Services Computing (SCC'08)*, 2008.

[13] R. Wartel, T. Cass, B. Moreira, E. Roche, M. Guijarro, S. Goasguen, and U. Schwickerath, "Image distribution mechanisms in large scale cloud providers," in *Proceedings of the IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom'10)*, 2010.

[14] B. Pfitzmann and N. Joukov, "Migration to multi-image cloud templates," in *Proceedings of the IEEE International Conference on Services Computing (SCC'11)*, 2011.

[15] A. Kochut and A. Karve, "Leveraging local image redundancy for efficient virtual machine provisioning," in *Proceedings of the 13th IEEE/IFIP Network Operations and Management Symposium (NOMS'12)*, 2012.

[16] E. Pugh, "Origins of software bundling," *IEEE Annals of the History of Computing*, vol. 24, no. 1, pp. 57 –58, Jan-Mar 2002.

[17] T. Garfinkel and M. Rosenblum, "When virtual is harder than real: Security challenges in virtual machine based computing environments," in *10th USENIX HotOS*, Berkeley, USA, 2005, pp. 20–20.

[18] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS'09)*, 2009.

[19] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. S. Lam, and M. Rosenblum, "Virtual appliances for deploying and maintaining software," in *Proceedings of the 17th USENIX conference on System administration (LISA'03)*, 2003.

[20] A.-C. Orgerie, L. Lefèvre, and I. G. Lassous, "On the energy efficiency of centralized and decentralized management for reservation-based networks," in *Proceedings of the IEEE Global Communications Conference (GLOBECOM'11)*, 2011.