Analysis of Overhead and Profitability in Nested Cloud Environments

Josef Spillner*, Andrey Brito†, Francisco Brasileiro†, Alexander Schill*
* Faculty of Computer Science
Technische Universität Dresden
01062 Dresden, Germany
Email: {josef.spillner,alexander.schill}@tu-dresden.de
† Laboratório de Sistemas Distribuídos
Universidade Federal de Campina Grande
CEP 58429-900, Campina Grande - PB, Brazil
Email: {andrey,fubica}@dsc.ufcg.edu.br

***Abstract*—The on-demand provisioning of computing and storage resources and the corresponding pay-as-you-go billing have made cloud computing a popular paradigm to achieve a high technological utility. While most use cases can adequately be covered by the offers of existing public cloud providers, the granularity in provisioning and pricing is not high enough for several task execution scenarios. Nested cloud environments are among the concepts to circumvent these problems. They let consumers manage their allocations with a higher degree of flexibility, including the reselling or repurposing to other subconsumers. In this paper, we take a critical look at the state of the art of nesting technologies and reason about the resource consumption overhead and the resulting economic profitability of employing a nested cloud. The results are validated within a cloud resource broker.**

I. Background

A. Introduction

Convenient, affordable and scalable distributed computing through public cloud computing infrastructure providers represents one of the strongest shifts in today's institutions and enterprises. Building upon parallelisation, virtual machines equipped with scalability-aware software are hosted with a chosen infrastructure provider and automatically scaled out horizontally on demand by launching additional instances. This procedure has several drawbacks for the consumer in the commercial reality, including a reliance on providers and a consumer-side overprovisioning due to coarse-grained resource and billing slices. The first contribution of this paper is hence an analysis of the technical and economical losses and the potential gains from alternative procedures with and without overhead consideration. The second contribution is a nested cloud architecture and its integration into a novel fiduciary highly-virtualising resource service broker which lets users resell surplus slices [1].

B. Infrastructure Service Characteristics and Limitations

Cloud Computing is a fuzzy term referring to new computing paradigms which combine scalable grid and cluster designs with on-demand provisioning through Everything-as-a-Service layers for free or with pay-per-use utility billing. The Infrastructure-as-a-Service layer (IaaS) is particularly interesting because the computing, storage and communication resources delivered through it are dependencies for almost all other layers. The scalability of software in the cloud is high when both horizontal (scale-out) and vertical (scale-up) mechanisms are supported by the infrastructure services, although in practice most providers limit their offers to scale-out despite the higher management costs and hence omit fine-grained reservation and control facilities. Further limits exist which damp the advantages for various reasons, especially for small remote calculations whose idle allocations add up considerably over time.

The observed limitations can be divided into three dimensions: a temporal one, a spatial one and a structural one. Furthermore, they can be divided into two directions: a minimum as lower bound, and a maximum as upper bound. Here, only the minimum is of interest, as infinite scalability is out of scope. Temporal restrictions apply to the billing by the hour and other fixed-minimum time intervals. In contrast, there is no noticable fixed minimum on the billing for the consumption of quantities of electrical energy or water. Spatial (copial, capacitive) restrictions apply to the reservation by the Gigabyte, the Gigahertz and other fixed-minimum amounts of discrete resource slices. These are rather needless consumption restrictions which could be resolved by a more fine-grained allocation policy aligned with the needs of the resource service consumer. In particular, vertical scalability for letting the consumer control the exact resource allocation at any time would contribute to a solution. Finally, structural restrictions apply to the flat model of running a single virtual machine on top of the Infrastructure-as-a-Service layer. This can be contrasted with a hierarchical recursive model in which nested subordinate virtual machines can be launched and controlled from the layer beneath them.

The aim of this paper is the presentation of our approach to analyse and systematically overcome the limitations through an appropriately designed intermediate nested cloud broker which nested leverages nested virtualisation support in IaaS. In the next section, we first present existing works which

978-1-4673-5163-8/12/$31.00 © 2012 IEEE

have a similar goal. Then, we describe solution methods with quantitative results on virtualisation and brokering techniques which determine the design of our approach. The fourth section explains and discusses the achieved architectural and practical results which include prototypes and experiments. The text concludes with suggestions for future resource service characteristics.

II. Related Work Analysis

The authors of xClouds [2] argue for hardware exposure and mutable hypervisors to overcome shortcomings in the deployment of portable and innovative applications which maximise the allocated resources at infrastructure service providers. The solution preferred by xClouds is nested virtualisation, which is found to be feasible performance-wise. However, xClouds remains at this virtualisation level and doesn't propose a suitable market or broker integration. It also remains vague about how the performance penalties contribute to general overhead costs when running higher-level virtualisation.

On the other hand, financial broker models have already been proposed in the cloud computing community. A broker acting as intermediate resource coordinator and assisting the provider with usage forecasts is expected to be financially beneficial to the provider [3]. Since our interest is with reducing or compensating consumer loss instead of increasing provider income, these results are not directly applicable to our work. Still, we argue that a broker model can also be beneficial to the consumer if operated independently from providers.

Recent research on precise resource allocation calculated by a mapping of the needs of the software running on this set of resources contributes to decreased consumer-side overprovisioning [4]. Hence, it becomes easier for infrastructure service consumers to cut out in advance the allocation part not needed for themselves and to reuse it for other purposes.

III. Solution Methods

Suitable techniques for higher consumer utility with minimal provider cooperation have been suggested based on the subdivision of resource reservations into either serial or parallel segments (or slices) [1]. Fig. 1 shows a sketch of both parallel and serial slicing of a cloud resource allocation and the expected overhead factors for both, which we will later determine numerically. Our analysis and realisation work concentrates on parallel sharing and hence on nested virtualisation on the operating system and hypervisor levels beneath the cloud computing stack. It encompasses four methods.

The first method is an experimental trial of nested virtualisation capabilities in current operating systems and hypervisors. Given the recent additions, documentation on this topic is still unreliable and many factors including hardware architecture and software versions contribute to feasible combinations. The second method is then for the best successful combination to determine the overhead in time (temporal) and in resources (spatial): CPU usage, memory consumption, disk space. The third method is a translation of this technical overhead into an economic one by evaluating current commercial infrastructure

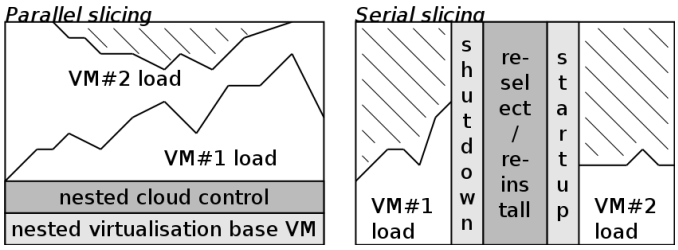

Fig. 1. Conceptual parallel and serial slicing of a cloud resource allocation

service terms and conditions expressed as IaaS ontologies. As a result, we will determine the potential gain of using our approach from which we then derive the net gain by substracting the necessary cost to operate a broker which implements the approach. The fourth method, which follows the design science methodology, is an architectural realisation of a highly-virtualising cloud resource broker with a user-friendly marketplace frontend and a cloud execution backend with nested virtualisation and vertical scaling capabilities. We report about it in the results section.

A. Virtualisation and Vertical Scaling Considerations

Nested virtualisation refers to a running host system at level $L0$ with a virtual machine at $L1$ which itself is the host to a virtual machine at $L2$. Deeper nesting levels $L3...Ln$ lead to recursively nested virtualisation. They are not of concern for our resource sharing concept due to high management complexity but we still consider them in our performance analysis in order to find out about future use cases.

We have assembled a representative choice of hardware-assisted and software-emulated virtualisation and container isolation software to cross-check the nested virtualisation of $L2$ on $L1$ with $L0$ being a stock 64-bit Linux system. Fig. 2 summarises our findings. The diagonal axis represents the recursively nested virtualisation with the same software on all levels. A success (v) or failure (x) sign without a background colour represents untested theoretic combinations whereas a background means that a test has been performed. The matrix complements our own tests (source c) with previously published results (source b [5]) and ongoing experiments of colleagues (source a contributed via the xen.user mailing list).

Due to the good nesting support, we have chosen a KVM-on-KVM for the experimental gathering of performance and resource usage overhead data. According to the authors of the KVM nesting code for Intel processors with the VMX flag, the overhead at $L2$ should be around 6-8% [6]. The authors of NestCloud, a nested virtualisation approach, come to similar conclusions with about 5-6% of overhead for both processing and memory usage [7].

Vertical scaling is also supported by KVM through an attachable monitor application. The relevant mechanisms are CPU hot-plugging and memory ballooning together with kernel same-page merging to reduce the memory usage even more when running identical executables and libraries in parallel VMs.

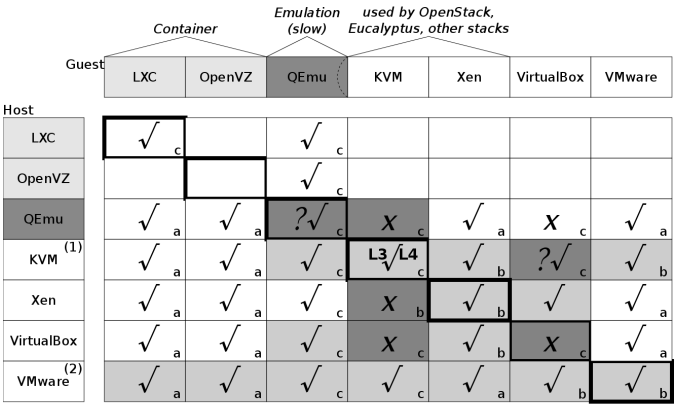

Fig. 2. Nested virtualisation matrix with L2 guests running on L1 hosts

B. Nested Virtualisation Runtime Performance

The experimental setup to verify and render these findings more precisely consists of a recursively nested operating system, on each level running with a Kernel with activated nested virtualisation. The operating system image is derived from a given image by wrapping the next higher level image into it and adding scripts to automatically run the experiments, start the inner VM and retrieve the results from it. Fig. 3 visualises the setup. At the VM core, a three-pass benchmark determines representative metrics for each level:

1) CPU-intensive: 100 million trigonometric formula calculations of the type $sin(i) + cos(j) + tan(i + j)$
2) RAM-intensive: 10 million pipe (user-space socket pairs) open/close requests
3) SYS-intensive: 10 thousand executions of the trivial system command `/bin/true`

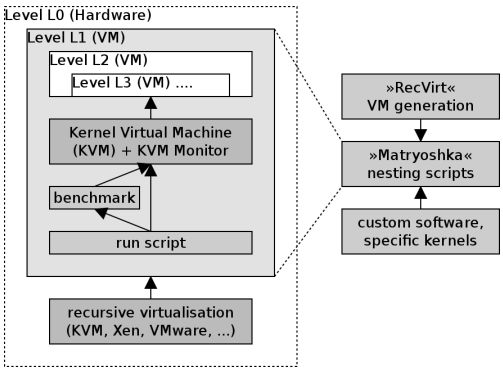

Fig. 3. Experimental setup through a nested virtualisation stack

The three following figures visualise the performance of nested virtualisation with the same operating system and hypervisor, but with different processors, through all currently possible nesting levels. The contenders are a dual-core AMD E-450 processor (Fig. 4), a dual-core AMD Athlon II X2 processor (Fig. 5) and a quad-core Intel i7 M620 CPU (Fig. 6), all running the Linux kernel 3.2 and KVM 1.0. The speed-up for the SYS-intensive task in L1, which invokes privilege-changing system calls, can be explained by the highly-optimised kernel routines for single-processor systems as compared to the multi-processor (SMP) hardware which needs to schedule the benchmark along with other software. Another interesting obervation is that for all practical purposes, $L_{max} = 2$ for the nested VMX routines applicable to Intel processors which cause a kernel failure when entering $L3$, and $L_{max} = 3$ for the nested SVM routines applicable to AMD processors which cause a kernel boot hang when entering $L4$. Supplementary, it should be noted that running a VM at $L3$ will cause significant performance penalties for certain kinds of software, but not for applications mostly operating in userspace, including numerical calculations, bag-of-tasks jobs and so forth.

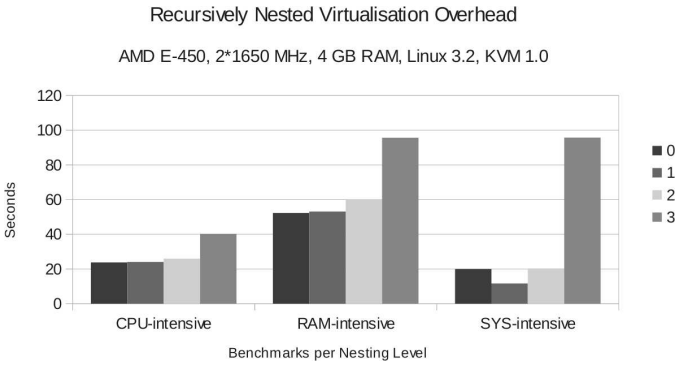

Fig. 4. Nested virtualisation performance on AMD E-450

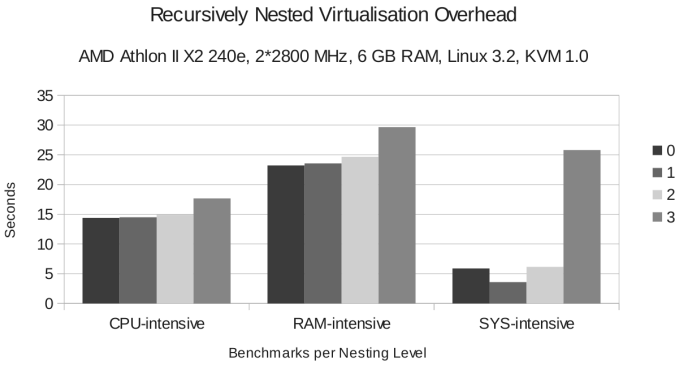

Fig. 5. Nested virtualisation performance on AMD Athlon II X2

C. Nested Virtualisation and Vertical Scaling Slicing Performance

On the AMD Athlon II X2 machine as used in the runtime performance benchmarks, the start of the VM takes 7.8 seconds for the boot into $L1$ compared to 26.9s into $L2$ and 167.3s into $L3$. The shutdown is unsurprisingly faster, with 14.2s from $L3$ back into $L2$, 4.8s back into $L1$ and 3.3s back into $L0$.

CPU hot-plugging and memory ballooning are basically instantaneous operations on all levels.

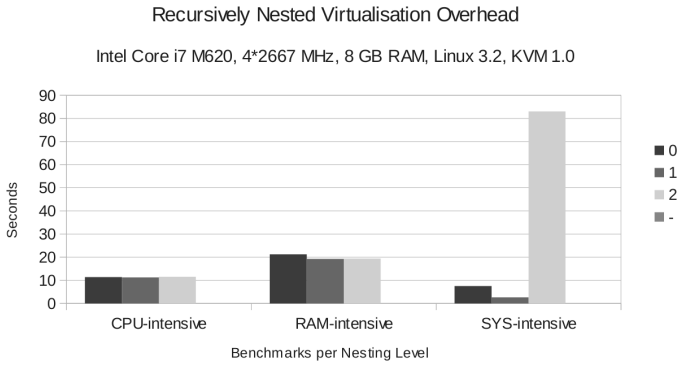

Fig. 6. Nested virtualisation performance on Intel Core i7

D. Economic and Service Science Considerations

The short-time allocation of resource services to cover load peaks beyond in-house infrastructure capacities is a often-discussed tactics with high utility [8]. In order to make resource slicing useful in an economic context, we will suggest a market-oriented approach of well-described and tradeable resources offered as services through a broker.

Service value networks are a business construct which combines service science foundations with practical service trading aspects like service level agreements [9]. Most services can be offered as value-added services and customisations, as well as bundles (economically motivated compositions) and orchestrations (technically motivated compositions). Cloud resource services add decomposition, also referred to as splitting or slicing, as variation category without excluding the other ones. Hence, resource service brokers should offer service consumers to become resellers or value-added service providers by applying the variations including decomposition. Table I summarises all suitable service offering variations for cloud resource services and existing languages to express them through formal means. It becomes obvious that generic service description languages like for instance the Unified Service Description Language (USDL) are suitable for composed services, but not for decomposed ones. Specialised resource description languages fill this gap. The Flexible Resource Description Language (FleRD) is one such language and a very promising one with support for resource splits called mapping, although it is currently still in its testing phase and oriented more towards networked resources [10]. Hence, we will attempt to enter a compromise by relying on both generic and specialised resource service descriptions in our architecture.

Table II summarises the technical overhead and the resulting economic loss when applying this to a typical offer from an infrastructure service provider.

IV. Results and Discussion

Following the experimental retrieval of findings about operating system level virtualisation techniques, we verify and realise a previously proposed cloud resource broker design which technically hosts services through a nested cloud and economically acts as a market-in-the-middle between resource providers and consumers [1].

TABLE I
Service variation and reselling models

Service variation	Explanation	Languages
Customisation	The service is enhanced with value-added components.	-
Bundling	Several services are sold together.	USDL
Orchestration	A service takes its functionality mainly from other services.	USDL, BPEL
Decomposition	A service is split into parts which are sold separately.	CloudNet-FleRD

TABLE II
Overhead Measurements

Overhead per virtualisation level			
	L0	L1	L2
Storage	0 MB	454 MB	454 MB
Memory	0 MB	34 MB	34 MB
Processing	0%	67%	1802%
Monthly overhead cost in the cloud			
S.Cost	0c	8.3-15.8c	8.3-15.8c
M.Cost	0c	0.9-1.3c	0.9-1.3c
P.Cost	0c	15.3-21.9c	410.6-590.3c

Cost ranges estimated according to Amazon EC2 availability region and on-demand plan pricing. Storage cost is shared, others per instance. All costs in Brazilian centavos.

A. A Market-in-the-Middle Architecture

In analogy to the recursively nested virtualisation experimental setup, a base virtual machine is dedicated to enabling the nested cloud with other VMs referred to as sub-VMs running at a higher virtualisation level. The nested cloud VM is to be deployed by the broker and offers control facilities through a configurator which turn it into a light-weight infrastructure manager. Fig. 7 visualises the nested cloud setup. KVM is used to switch on and off the respective sub-VMs. The KVM monitor, which is interfaced with through the serial communication tool minicom, allows for fine-grained vertical scalability, including dynamic memory assignments and CPU hotplugging into the sub-VMs.

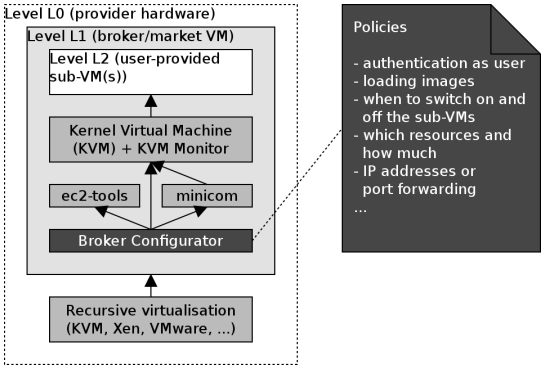

Fig. 7. Nested cloud with base virtual machine

The control facilities are supposed to be bound to a service configuration created by a preparation dialogue. In addition to regular service configuration options, the dialogue asks the potential user of a virtual machine instance about the utilisation and sharing of otherwise unused resource capacities. The option to share serially or in parallel is orthogonal to this kind of sharing. A third option would be to ask for manual (user-managed) or automatic (broker-managed) control of the sub-VMs, either startup and shutdown for the serial sharing, or dynamic resource provisioning for the parallel sharing. An automatic control requires reliable usage forecasting which is not yet part of our concept.

For the higher-level networking, there are two choices: The dynamic reservation of IP addresses, which requires this capability at the cloud provider, or port forwarding, which is less suitable because it requires each sub-VM to specify its relevant ports and may lead to unpredictable port numbers in the case of conflicts. Elastic IP address assignments are increasingly available at commercial cloud providers which will make this choice the preferred one for deployments in public clouds.

B. Experimental Deployment in HVCRB

In order to verify the proposed nested cloud approach to yield more efficient resource usage at infrastructure service providers, we have implemented it and reproduced the overhead measurements as refined highly-virtualising cloud resource broker (HVCRB) [1] on top of SPACEflight, an experimentation system for service and cloud computing [11], on an AMD64 machine with 8 GB RAM. The system was made bootable with a custom Linux 3.5 kernel, hence looping through the nested virtualisation capabilities to all VMs deployed in Eucalyptus. Also, we have added the nested cloud base VM to the demonstrator and modified the platform services to support the sharing variation on resource services. As a scenario service, we deployed the social network software Noosfero as a SaaS offering inside a VM.

Fig. 9 summarises the experimental architecture and the most important workflows therein: (1) The providers of complex software offered as services and packaged as virtual machines deploy the VM images at the broker. In SPACE, a web-based provisioning tool then takes over the package. The artefacts are extracted and registered at a service registry and contract manager while the VM image itself is recognised as to be handled by a cloud stack, which is Eucalyptus. (2) The VM images are deployed into the cloud stack. To turn the service broker into a HVCRB, the nested cloud VM image also needs to be deployed there. (3) Service consumers select a resource-dependent service through a market interface and instantiate it. The use of the service is prepared by a negotiation of a Service Level Agreement and a specific configuration with business objectives, such as the number of participants in the social network scenario service. HVCRB adds a configuration option about the utilisation and sharing of otherwise unused resource capacities according to possible sharing schemes [1]. The service configuration along with the agreement negotiation is shown in Fig. 8. (4) A configurator picks up the configuration document and instructs the cloud stack to allocate a certain amount of resources depending on the business objectives for the scenario services. With HVCRB, the cloud stack instantiates the nested cloud VM whose controller is then instructed to launch the actual service VM. (5) Final service consumers interact with the service VM without knowledge about the existence of the nested cloud layer. (6) Consumers looking for resource services to instantiate resource-dependent services through them find them at the market. With HVCRB, the instantiation of VMs can use the resources in the form of sub-VMs running on the nested cloud VM.

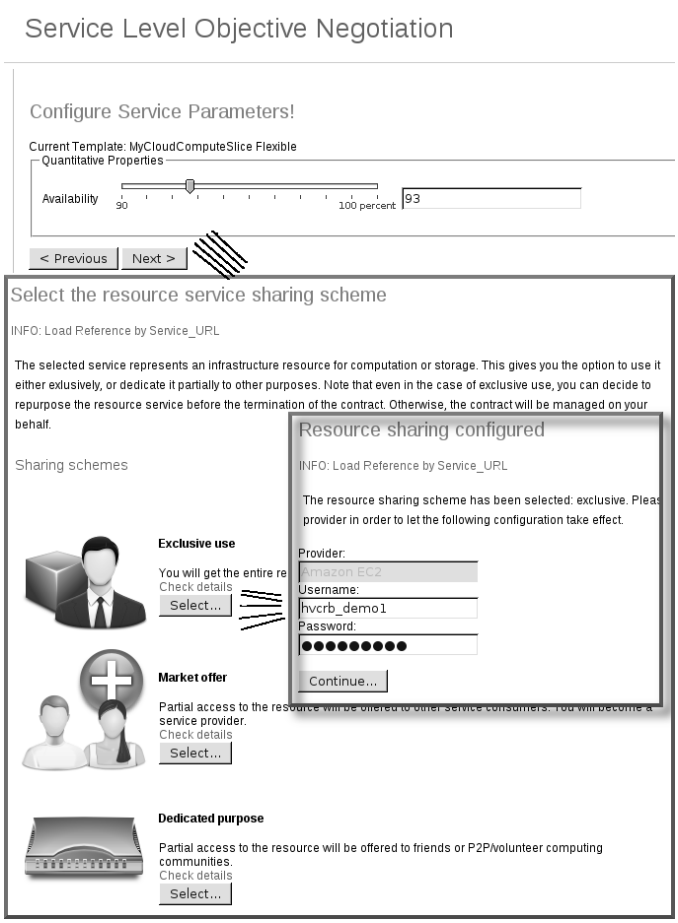

Fig. 8. Service configuration along with the SLA negotiation in the contract wizard provided by SPACE (screenshots)

Due to limitations of publicly offered cloud environments, we could not yet verify the approach with such offers. The reasons include outdated host kernels which lack nested virtualisation and vertical scaling capabilities, the lack of elastic IP assignments to connect the sub-VMs, and missing control facilities to start up and shut down the sub-VMs from arbitrary image files. We expect that the evolutionary adoption of newer software versions will make a deployment at public cloud providers feasible, although we also see how this might rise legal issues similar to used software reselling debates in the past.

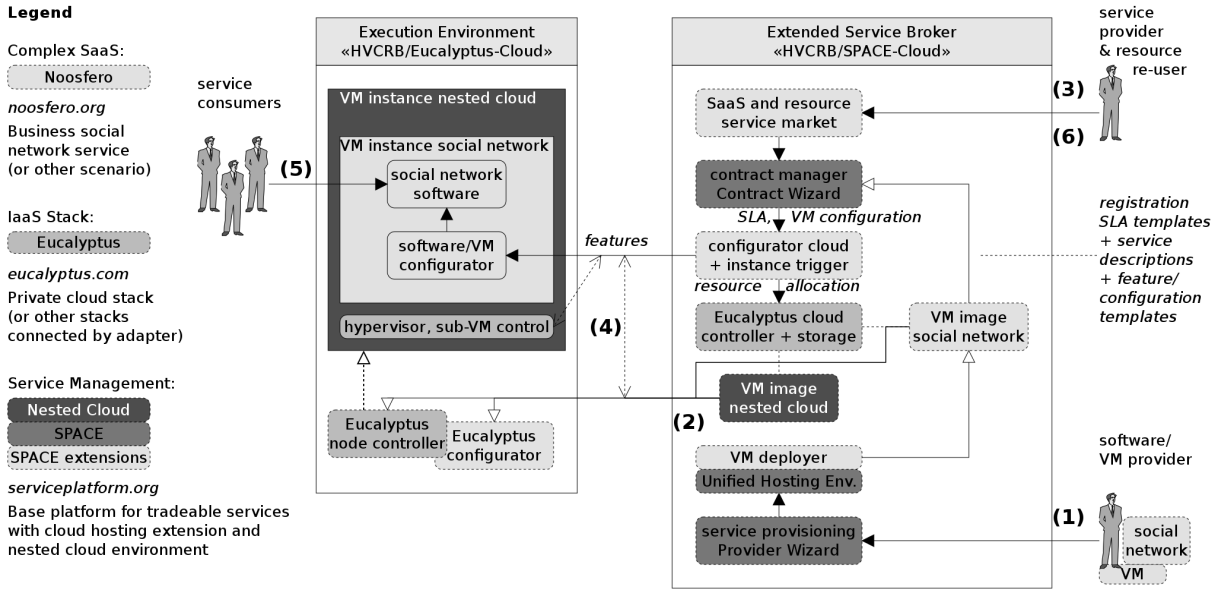

Fig. 9. Nested cloud hosting architecture beneath a highly virtualising cloud resource broker with scenario SaaS

V. Conclusions

This paper analysed and realised a previously suggested layered approach towards highly virtualising cloud resource brokers for efficient allocation and utilisation of resources procured from public cloud providers. We qualitatively and quantitatively analysed recursively nested virtualisation as a means to achieve especially parallel sharing. A two-level nested cloud layer was then derived from this technology in order to integrate it with existing cloud computing environments, which we performed through a highly-virtualising broker integrated with the service platform SPACE. The results suggest that by introducing this approach into public clouds, resources can be allocated more efficiently and with high utility while consumer-side overprovisioning can be compensated through open resource service markets.

Acknowledgements

This work has received funding under project number 080949277 by means of the European Regional Development Fund (ERDF), the European Social Fund (ESF) and the German Free State of Saxony. It has also been funded by a scholarship from the Brazilian National Council for Scientific and Technological Development - CNPq (PDJ 159716/2011-0).

References

[1] J. Spillner, A. Brito, F. Brasileiro, and A. Schill, "A Highly-Virtualising Cloud Resource Broker," in *5th International IEEE/ACM Conference on Utility and Cloud Computing (UCC)*, November 2012, Chicago, Illinois, USA.

[2] D. Williams, E. Elnikety, M. Eldehiry, H. Jamjoom, H. Huang, and H. Weatherspoon, "Unshackle the cloud!" in *Proceedings of the 3rd USENIX Conference on Hot topics in Cloud computing (HotCloud)*, June 2011, Portland, Oregon, USA. [Online]. Available: http://xcloud.cs.cornell.edu

[3] O. Rogers and D. Cliff, "A financial brokerage model for cloud computing," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 2, April 2012.

[4] L. Wu, S. K. Garg, and R. Buyya, "SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments," in *11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, May 2011, pp. 195–204, Los Angeles, California, USA.

[5] O. Berghmans, "Nesting Virtual Machines in Virtualization Test Frameworks," Ph.D. dissertation, Universiteit Antwerpen, 2010.

[6] M. Ben-Yehuda, M. D. Day, Z. Dubitzky, M. Factor, N. Har'El, A. Gordon, A. Liguori, O. Wasserman, and B.-A. Yassour, "The Turtles Project: Design and Implementation of Nested Virtualization," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, October 2010, pp. 423–436, Vancouver, British Columbia, Canada.

[7] Z. Pan, Q. He, W. Jiang, Y. Chen, and Y. Dong, "NestCloud: Towards practical nested virtualization," in *International Conference on Cloud and Service Computing (CSC)*, December 2011, pp. 321–329, Hong Kong, China.

[8] P. D. M. Jr., F. Brasileiro, R. A. Santos, D. C. M. Maia, R. Lopes, M. W. A. de Carvalho, R. M. C. Ribeiro, N. Andrade, and M. Mowbray, "Business-Driven Short-Term Management of a Hybrid IT Infrastructure," *Journal of Parallel and Distributed Computing*, vol. 72, no. 2, pp. 106–119, February 2012.

[9] B. S. Blau, "Coordination in Service Value Networks – A Mechanism Design Approach," Ph.D. dissertation, Universität Karlsruhe (TH), 2009.

[10] G. Schaffrath, S. Schmid, I. Vaishnavi, A. Khan, and A. Feldmann, "A Resource Description Language with Vagueness Support for Multi-Provider Cloud Networks," in *International Conference on Computer Communication Networks (ICCCN)*, July 2012, Munich, Germany.

[11] J. Spillner, "SPACEflight - A Versatile Live Demonstrator and Teaching System for Advanced Service-Oriented Technologies," in *21st International Crimean Conference on Microwave and Telecommunication Technology (CriMiCo/*КрыМиКо*)*, September 2011, pp. 455–456, Sevastopol, Ukraine. [Online]. Available: http://serviceplatform.org/