Hardware as a service - enabling dynamic, user-level bare metal provisioning of pools of data center resources.

Jason Hennessey*, Chris Hill†, Ian Denhardt*, Viggnesh Venugopal‡, George Silvis*, Orran Krieger*, Peter Desnoyers ‡
*Boston University. {henn,isd,gsilvis,okrieg}@bu.edu
†M.I.T. {cnh}@mit.edu
‡Northeastern University. {ve0028,pjd}@ccs.neu.edu

***Abstract*—We describe a "Hardware as a Service (HaaS)" tool for isolating pools of compute, storage and networking resources. The goal of HaaS is to enable dynamic and flexible, user-level provisioning of pools of resources at the so-called "bare-metal" layer. It allows experimental or untrusted services to co-exist alongside trusted services. By functioning only as a resource isolation system, users are free to choose between different system scheduling and provisioning systems and to manage isolated resources as they see fit. We describe key HaaS use cases and features. We show how HaaS can provide a valuable, and somehwat overlooked, layer in the software architecture of modern data center management. Documentation and source code for HaaS software are available at: https://github.com/CCI-MOC/haas.**

I. Introduction

Todays's massive scale-out data center facilities typically operate under one of two models:

1) a **bare-metal** paradigm in which a single OS environment is deployed and end-users typically must fit their workflow to the environment provided. This paradigm is dominant in many high-performance scientific and technical computing enterprises. It provides full hardware performance to end users and can readily accomodate hardware innovations such as co-processors, specialized networks and massively parallel file system architectures.
2) a **virtualized/"cloud"** paradigm which supports end-user selectable OS environments. In this paradigm the full software environment can be tailored to a particular workflow. This paradigm is dominant in public and private cloud initiatives. For a variety of reasons this model does not, in general, provide the same level of performance guarantees or specialized hardware support that a **bare-metal** environment provides.

In developing the HaaS system we are motivated by an interest in providing end-users the performance and capability characteristics of **bare-metal** systems alongside the software environment flexibility that comes with **vitualized/"cloud"** environments.

Prior research has investigated this sort of capability, with a number of groups having created automated systems for both research [1]–[6] and commercial [7], [8] purposes. These systems currently (or plan to soon) implement three core functions: 1) *scheduling:* determining the specific set of nodes that will be given in response to a request, 2) *provisioning:* installing/configuring software onto bare nodes and 3) *isolation:* allowing a user to control which nodes can communicate with his nodes. While valuable, these systems are not general purpose because they couple their scheduling and provisioning functionality whereas some use cases require alternative schedulers [9]–[11] and provisioning systems [12], [13]. As a result, the existing systems are only deployed in environments dedicated to their specialized use cases.

A. HaaS Use Cases

We introduce a new system called HaaS (Hardware as a Service) which only performs isolation. With it, a mixture of user-specified provisioning and scheduling systems can be used to address a wide set of use cases. This flexibility means that HaaS can be deployed as a fundamental layer in a multi-tenant general purpose data center. Specific use cases we target include:

- Deploying a customized OS (like neurodebian) onto bare metal nodes for specific HPC workloads
- Running bare metal HPC clusters with isolated networks to minimize jitter [14]–[16].
- Enabling commercial, production and research-based users to deploy their own clouds using the provisioning platform that best meets their needs.
- Safely giving experimental cloud, networking and OS researchers access to raw compute nodes without risk to production systems.
- Enabling isolated bare-metal acceptance testing, development, pre-production and production deployments for mission critical applications.
- Running competing schedulers in the same data center which can take into account different properties such as hardware accelerators.

By installing HaaS as the basic isolation layer, data centers can realize increased flexibility, potentially allowing organizations to cooperatively own and operate equipment while giving user communities flexible control.

II. Description

Key concepts and their descriptions for the HaaS model are shown in Figure 1. The main difference between HaaS and related systems is that, rather than building them in, HaaS enables multiple schedulers and provisioning systems to be used. For scheduling, while other systems decide the nodes to be used, HaaS enables users or schedulers to directly determine which node to use via a REST API. This allows different schedulers to be used for different environments or even to have multiple schedulers serving different use cases out of the same larger pool of nodes. For provisioning, other systems

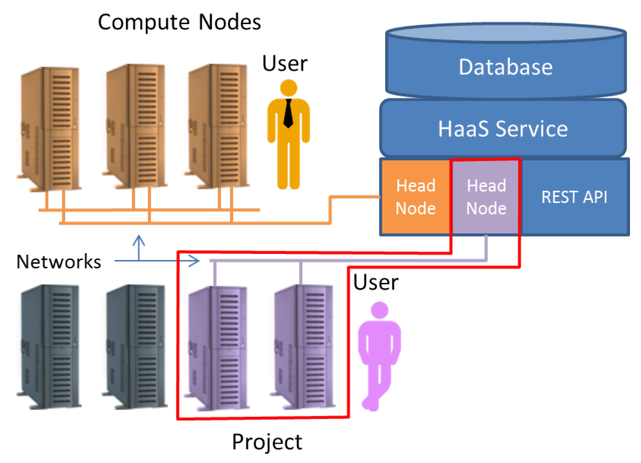

Fig. 1. Diagram of major HaaS components showing two isolated projects. One is managed by a professional (top; orange) while the other is managed by someone more casual (bottom; purple). A *project* represents a collection of resources and is managed by *groups* which contain individual *users*. Projects can contain allocated *compute nodes*, a *head node* and *networks*. *Compute nodes* represent bare metal systems, whereas a *head node* can either be a virtual or physical system from which provisioning or other management software may be run. *Networks* are generic methods of communications that connect compute nodes through a node's *NIC* to a particular *switch*. The *HaaS Service* tracks state and processes client requests to configure resources.

require users to provide to them an image to be deployed. With HaaS, the user is given a head node that is connected to isolated provisioning and management networks of the allocated resources. The user can then deploy any provisioning software they would like on that head node since they are given full control of it. End users are free to deploy network broadcast services and other intrusive network tools, since HaaS ensures necessary network isolation.

HaaS uses its knowledge of compute nodes and network connectivity to enforce isolated configurations. To accomplish connectivity, it maintains a list of network switch ports, the NICs of compute nodes, and how the two are connected. Switches are supported via model-specific drivers. Currently, OSI-model layer 2 networks using VLANs are supported for connecting and isolating networks, though the model could conceivably accommodate other mechanisms such as OpenFlow, Infiniband or Storage Area Networks.

III. Demonstration and Evaluation

With HaaS, we have observed that moving resources takes place in minutes, rather than days or more for equivalent manual system (often the norm today). We plan to demonstrate and evaluate HaaS with examples from the bulleted list in subsection I-A to illustrate its scope and functionality. We also plan to explore how complexity metrics can be used to illustrate how the HaaS abstractions allow end-users to easily express compute and network configuration steps succinctly and unambiguously.

IV. Acknowledgements

Partial support for this work was provided by the MassTech Collaborative Research Matching Grant Program, National Science Foundation award #1347525 and several commercial partners of the Mass Open Cloud who may be found at http://www.massopencloud.org. We are also grateful to the many contributors towards the HaaS project, including Zhaoliang Liu, Andrew Mohn, Jonathan Bernard, Valerie Young, Jonathan Bell, Rohan Garg, Ritesh Singh and Abhishek Raju.

References

[1] D. Anderson, M. Hibler, L. Stoller, T. Stack, and J. Lepreau, "Automatic online validation of network configuration in the emulab network testbed," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, June 2006, pp. 134–142.

[2] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, no. 0, pp. 5 – 23, 2014, special issue on Future Internet Testbeds Part I. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613004507

[3] J. S. Chase, D. E. Irwin, L. E. Grit, J. D. Moore, and S. E. Sprenkle, "Dynamic virtual clusters in a grid site manager," in *High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on*. IEEE, 2003, pp. 90–100.

[4] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, pp. 59–64, Jan. 2003. [Online]. Available: http://doi.acm.org/10.1145/774763.774772

[5] S. Averitt, M. Bugaev, A. Peeler, H. Shaffer, E. Sills, S. Stein, J. Thompson, and M. Vouk, "Virtual computing laboratory (vcl)," in *Proceedings of the International Conference on the Virtual Computing Initiative*, 2007, pp. 1–6.

[6] G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Voeckler, R. J. Figueiredo, J. Fortes, K. Keahey, and E. Delman, "Design of the futuregrid experiment management framework," in *GCE2010 at SC10*, IEEE. New Orleans: IEEE, 11/2010 2010, Paper.

[7] D. Van der Veen *et al.*, "Openstack ironic wiki." [Online]. Available: https://wiki.openstack.org/wiki/Ironic

[8] Canonical, "Metal as a Service." [Online]. Available: http://maas.ubuntu.com/

[9] A. Yoo, M. Jette, and M. Grondona, "SLURM: Simple linux utility for resource management," in *JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING*, ser. LECTURE NOTES IN COMPUTER SCIENCE, vol. 2862. BERLIN, GERMANY: SPRINGER-VERLAG BERLIN, 2003, Article; Proceedings Paper, pp. 44–60.

[10] I. Foster, "Globus toolkit version 4: Software for service-oriented systems," *Journal of Computer Science and Technology*, vol. 21, no. 4, pp. 513–520, 2006. [Online]. Available: http://dx.doi.org/10.1007/s11390-006-0513-y

[11] A. Reuther, J. Kepner, W. Arcand, D. Bestor, B. Bergeron, C. Byun, M. Hubbell, P. Michaleas, J. Mullen, A. Prout *et al.*, "Llsupercloud: Sharing hpc systems for diverse rapid prototyping," in *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*. IEEE, 2013, pp. 1–6.

[12] T. Lange, "Fully automatic installation (fai)," 2006.

[13] "Cobbler." [Online]. Available: http://www.cobblerd.org/

[14] F. Petrini, D. Kerbyson, and S. Pakin, "The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of asci q," in *Supercomputing, 2003 ACM/IEEE Conference*. IEEE, 2003, pp. 55–55.

[15] D. Tsafrir, Y. Etsion, D. G. Feitelson, and S. Kirkpatrick, "System noise, os clock ticks, and fine-grained parallel applications," in *Proceedings of the 19th annual international conference on Supercomputing*. ACM, 2005, pp. 303–312.

[16] K. B. Ferreira, P. Bridges, and R. Brightwell, "Characterizing application sensitivity to os interference using kernel-level noise injection," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 19.