Providing Full Isolation in Cloud System Software

John Lange
University of Pittsburgh
jacklange@cs.pitt.edu

1 Introduction

An ongoing challenge to deploying cloud based applications, particularly at scale, has been the inability of the underlying infrastructure to effectively provide performance isolation to users and their applications. While a significant amount of existing work has addressed this issue in multiple ways [3, 4], most of these solutions have focused on the allocation and isolation of the underlying hardware resources for concurrent workloads. Unfortunately, while these approaches do provide measurable benefits in consolidated environments, we have shown that they are incomplete because they do not consider interference effects at the *system software layer* [2, 1]. For many applications, operating system (OS) level services and resources have just as prominent an effect on performance as the underlying hardware they provide access to. Therefore, in order to provide a fully isolated execution environment, it is necessary to address performance interference at both the hardware *and software* layers of the system.

Providing fully isolated environments to cloud workloads requires support in the underlying system software which in many cases can only be achieved through modifying (or possibly replacing) the OS and runtime environment. One example of our work focuses on providing software level performance isolation through the use of *lightweight co-kernels*. In our architecture, multiple heterogeneous operating system instances co-exist on a single server platform, and directly manage independent sets of hardware resources. Each co-kernel executes as a fully independent OS environment and does not rely on any other OS instance for system level services, thus avoiding cross workload contention on OS resources. Each co-kernel is capable of providing fully isolated system software environments, or *enclaves*, to local consolidated workloads. This approach allows a system administrator to dynamically compose independent enclaves from arbitrary sets of local hardware resources at runtime based on applications' resource and isolation requirements. An example of such a system configuration is shown in Figure 1.

The foundation of our work is based on previous work in specialized operating systems for High Performance Computing (HPC) and supercomputing environments. HPC applications are notable for both their large scalability requirements as well as their strong inter-node dependencies and synchronization behaviors. This poses a significant challenge to an OS architecture as single node performance issues propagate throughout the entire system due to the lock step nature of the computation. For this reason HPC OS kernels target performance consistency as the paramount design goal, even above overall performance. This is in contrast to most commodity systems which primarily focus on system performance and are willing to sacrifice occasional overheads and slow downs in order to improve it. To achieve performance consistency, specialized HPC OS environments tend to follow a *lightweight* philosophy that restricts the number of OS features and severely limits the sharing of hardware resources. We claim that the same design decisions that allow strong performance consistency in an HPC environment can be leveraged to provide strong performance isolation in a consolidated cloud environment.

Our co-kernel architecture allows on-demand runtime deployment of independent lightweight co-kernels on the same local node to meet the workload requirements presented by an application. In our model, a cloud service will be able to offer specialized execution environments that provide improved SLA metrics based on isolation guarantees provided by the underlying system software. Such approaches are already in use at many cloud service providers, where customers can select the degree of resource contention they are willing to be subjected to in return for increased usage fees. These lightweight environments would be capable of providing Platform-as-a-Service environments through the use of specialized runtimes or more general purpose Infrastructure-as-a-Service functionality through the use of a an available lightweight virtualization layer. These isolated environments would not require dedicated servers, racks, or datacenters, but would instead rely on the partitioning of local hardware resources among a set of independent OS environments.

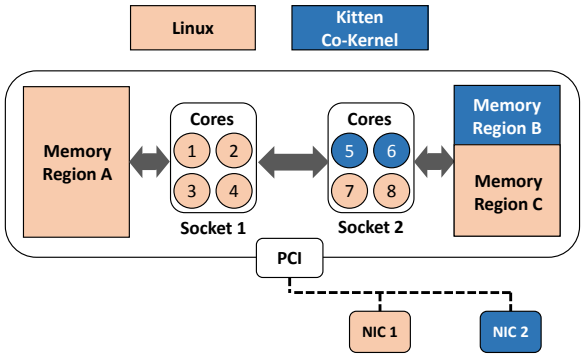

Figure 1: A co-kernel environment on a partitioned cloud environment

This approach allows the local resource management service to make localized layout decisions dynamically based on either a global policy or by monitoring system performance.

2 Testbed Requirements

To further our research agenda there are two capabilities that would be beneficial from a cloud testbed. First is the ability to evaluate research on system software architectures directly in the environment, and second is the ability to monitor the system on a large scale and collect system performance measurements from the system during normal operation. These capabilities would allow us (and other systems researchers) to not only evaluate their own work in the system, but also to gain a greater understanding of system software behavior from an actual cloud architecture (something that is very difficult if not impossible to do currently).

Experimental Testbed The first requirement we have of a testbed is the ability to easily and rapidly deploy customized kernel images on the testbed systems. At minimum our work would require the enabling of a set of standard Linux configuration options in an otherwise unmodified kernel, however the ability to load a fully custom kernel image could prove to be necessary in the future for our work. Furthermore, it is likely that other systems researchers would need the ability to make kernel modifications as well. Second, we would need the ability to gain root level privileges on the systems themselves. Our work is done in the context of Linux kernel modules, which require administrative permissions to both load and manage during run time. Therefore, broad root level access or at least sudo access with suitable file permissions would be required.

In addition to the administrative requirements our work is based on the ability to dynamically partition a local compute node's resources. This capability needs to be supported at both the software (administrative) level as well as in the hardware itself. Therefore, any testbed architecture would ideally include hardware that allows hardware-level partitioning to allow both performance and administrative isolation. For example, SR-IOV capable PCI devices, IOMMUs, and large scale multi-core NUMA systems provide this capability (to varying degrees) on current generation platforms.

Workload Monitoring The second main feature we would like to see on a cloud testbed is the ability to efficiently monitor the underlying system software. Ideally this capability would allow us to consistently monitor the OS + runtime as the system is used by a wide range of users and workloads. These measurements are necessary in order to gain a better understanding of the characteristics of workload interference and how system software architecture impacts it. Such access would include the ability to access the measurements collected from hardware performance monitoring features as well as kernel level measurements. These capabilities are currently available via a number of mechanisms (perf, ftrace, etc), so at minimum having these features enabled should be a requirement. Ideally the testbed would also include an integrated and centralized infrastructure to enable both the collection and storage of measurement data in a way that is easily gathered, accessed, and shared.

References

[1] HPMMAP: Lightweight Memory Management for Commodity Operating Systems. In *Proceedings of the 28th IEEE International Parallel and Distributed Processing Symposium, (IPDPS)* (2014)).

[2] KOCOLOSKI, B., OUYANG, J., AND LANGE, J. A Case for Dual Stack Virtualization: Consolidating HPC and Commodity Applications in the Cloud. In *Proceedings of the third ACM Symposium on Cloud Computing (SOCC)* (2012).

[3] POPA, L., KUMAR, G., CHOWDHURY, M., KRISHNAMURTHY, A., RATNASAMY, S., AND STOICA, I. Faircloud: sharing the network in cloud computing. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication* (2012).

[4] SHUE, D., FREEDMAN, M., AND SHAIKH, A. Performance Isolation and Fairness for Multi-Tenant Cloud Storage. In *Proc. 10th Symposium on Operating Systems Design and Implementation (OSDI)* (2012).