Scalable Elastic Systems Architecture

Dan Schatzberg
Boston University
dschatz@bu.edu

Jonathan Appavoo
Boston University
jappavoo@bu.edu

Orran Krieger
VMware
okrieger@vmware.com

Eric Van Hensbergen
IBM Austin Research Lab
ericvanhensbergen@us.ibm.com

1. Introduction

Elasticity should be treated as a first class system parameter. Particularly in large cloud environments, elastic applications would benefit if the underlying infrastructure provided primitives for elasticity and were themselves elastic. If you want to provide an elastic service and the cloud does not provide good primitives for the degree of elasticity you require, then you are forced to over-provision – acquire more resources than you instantaneously need and subsequently hoard them. Doing so hinders the cloud's ability to optimize global system utilization. Free or idle resources become hidden. If however, each cloud layer provides appropriate primitives that permit resources to be acquired and released at a scale that is equal to or better than what is required, then hoarding is less likely to occur. This permits the cloud infrastructure to collectively migrate resources to the real demand. To achieve this in a multi-layer system, demand must be transparently reflected from top to bottom. We must focus on the design and evaluation of primitives for expressing and managing elasticity at all levels, across nodes, and potentially across data centers.

If research focuses on pushing the boundaries of elasticity, new classes of applications can be developed. For example, if a cloud would permit an application to grow and shrink the use of thousands of processors between mouse clicks, then *High Performance Interactive Applications* would be viable. Consider a medical imaging and analysis application. Using a raw megapixel image with an algorithm requiring quadratic memory in the size of the input, this requires roughly 14 terabytes of memory, putting it well outside the reach of the ram capacities of desktop computers. However, a "small" supercomputer today ($1/10$ of the largest current IBM BlueGene P System), capable of approximately 10^{14} operations per second, can not only contain the data, but can perform an operation on each data value in under a second. All of a sudden, operating on the image not only becomes viable, but we can even do it at interactive speeds.

While an interactive version of this application has large value, it is not feasible today. Suppose a doctor's office had the necessary software and wanted to use Amazon's EC2 HPC offering for an 8 hour work day. To operate on the image would require 623 compute instances[1]. Given pricing at the time of writing, this translates to approximately $\$8000.00$ per day. Due to the interactive nature of the application, the actual utilization of the instances will be a small fraction of the time that is being paid for. This is likely a cost prohibitive proposition. If, however, it was possible to acquire and release the resources at interactive time scales, then the instances could be reallocated to other EC2 users and the doctor's cost would more closely reflect the usage. Researching dramatically higher degrees of elasticity with respect to the scale of the resources and duration they are held would enable such high performance interactive applications.

If we develop effective ways of exporting the elasticity via designed and usable primitives, then we can not only ease the burden of developing elastic applications and services, but also we can foster and encourage them. We can reduce the application development burden by providing suptport for representing and reflecting dynamic demand and translating it into dynamic requests for resources. Similar to how a traditional operating system transparently manages memory via mappings and pages faults, one can explore how systems can enable primitives for elasticity.

In summary we argue that elasticity is an important area of research and hypothesize that research in this area will lead to more efficient systems with less hoarding, new applications that exploit massive cloud resources elastically, and system software and libraries that will simplify the task of developing elastic applications.

In this talk we will present our goals for a system that supports extreme elasticity. Motivated by these goals, we will present our Scalable Elastic Systems Architecture (SESA).

2. Goals

Based on our observations, we posit the following goals for a systems architecture for elasticity:

Top-Down Demand The system should enable demand on services to flow from high level layers as transparently as possible to the lowest layers of the system. Hoarding should be discouraged or at least made transparent. Event driven interfaces and services should be supported and encouraged by the system.

Bottom-Up Support We advocate that elasticity should be an explicit characteristic that should be supported in the lowest layers of a system and, if possible, all the way into the hardware. The construction of layers that are explicit about the elasticity they provide with respect to the base elasticity of the system should be encouraged via systems support.

Exploit Modularity Use modularity to enable applications to map elasticity in their demand as closely to the capabilities of the system when desired. Embrace the layering of cloud computing as an abstract architecture but ensure that all but the lowest layer can have varying implementations or be over-ridden. Further, provide some form of support for a component model that enables the base elasticity to be exploited by new and advanced applications.

3. SESA

Motivated by these observations, we are exploring a Scalable Elastic System Architecture (SESA) which aims to enable extreme degrees of elasticity across all system layers, aiding in the construction of elastic software. Along with an abstract layering, we propose a distributed elastic component model that can be used to construct layers of the system that are tuned to exploiting fine grain elasticity.

SESA defines four meta layers to a system that can easily be mapped to current and future cloud computing environments. At the bottom is the physical Elastic Node/HW Layer that represents a data center's underlying computer resources. We assume a model in which the resources of the data center are decomposed into *nodes* that form the basic unit of resource allocation and thus elasticity. The next layer up, the Elastic Partition Layer, provides groups or partitions of node resources that can be associated with a consumer or principal. A partition is the basic unit by which a principal can elastically aggregate node resources. To enable applications to scale and exploit the elasticity of a partition, the next layer provides an Elastic Building Block model. Elastic Building Blocks (EBBs) are the primary way in which application software is structured so that it is scalable and changes in demand can be converted into elastic consumption of node resources for a particular application of a principal. The top layer is the Elastic Service and Runtime Layer. The specific service and or runtime code of the application is written as a dynamically allocated set of EBBs.

Our systems software focus is on the top two layers. Our goal is to introduce a distributed library OS runtime that enables Elastic Building Blocks (EBBs) for the construction of system and application layers that express and exploit the maximum elasticity of the hardware to meet the demands of interactive workloads. Our model attempts to achieve the goals of top-down demand, bottom-up support and exploiting modularity. Specifically, it uses an event model that maps events to method invocations of EBB's. EBB construction and destruction are by default to be triggered by event access and quiescence. The components will have an associated IPC like model so that invocations can cross layers. The library model will enable EBB constructed services to be constructed and deployed in conjunction with existing implementations.

4. Concluding remarks

Systems in the past were designed to efficiently share fixed resources among a mix of different applications. We are currently building our clouds and elastic cloud applications using the HW and SW systems that arose from this legacy.

In this talk we will propose a new research agenda focused on elasticity. We argued that elasticity is an important area of research and hypothesized that research in this area will lead to more efficient systems with less hoarding, new applications that exploit massive cloud resources elastically, and system software and libraries that will simplify the task of developing elastic applications.

We will discuss some of our thoughts on a top-to-bottom cloud-scale system focused on elasticity. We argue that such a system will require: 1) a HW/IaaS layer that can quickly reallocate resources to different applications, 2) an event driven model where resource demand flows from the high level layers as transparently as possible to the lowest level of the system, and 3) a model of modularity that allows layers to be overridden as necessary and provides applications with a component model that enables the base elasticity to be exploited by new and advanced applications.

These requirements have led to the design of the SESA system briefly described in the previous section. We are just starting to build this system. The elastic building blocks extend our previous work on building blocks from K42[5] and draw additional inspiration from Fragmented Objects[7], Distributed Shared Objects[4], and Distributed Shared Abstractions[3]. We will incorporate EBB into a library OS inspired by our experience with Libra [2]. This library OS will be able to automatically and quickly extend across and release VMs or HW nodes as the application's demands change. Our prototype will initially focus on a MATLAB like environment built utilizing SAGE[8].

We expect to rapidly have an end-to-end simple operational implementation of SESA focused on a medical imaging application on top of SAGE running on both BG/Q systems and vCloud[6]. Focusing this work on one relevant application domain, a particular programing model, and two interesting large scale systems will give us good experience on the applicability of these ideas.

Cloud computing makes whole new classes of elastic applications, applications that can exploit thousands of nodes for minutes or even seconds, possible. The nature of cloud computing makes it practical to develop whole new systems, targetting elasticity, and have them be useful for some workload while those system co-exist with legacy systems and applications. Research in elastic systems software is both urgent and can quickly be made relevant to a broad community.

Acknowledgments

This material is based upon work supported in part by the Department of Energy Office of Science under its agreement number DE-SC0005241 and DE-SC0005365.

References

[1] Amazon. Amazon EC2 Pricing. `http://aws.amazon.com/ec2/pricing/`.

[2] G. Ammons, R. W. Wisniewski, J. Appavoo, M. Butrico, D. Da Silva, D. Grove, K. Kawachiya, O. Krieger, B. Rosenburg, and E. Van Hensbergen. Libra. In *Proceedings of the 3rd international conference on Virtual execution environments - VEE '07*, 2007.

[3] C. Clmenon, B. Mukherjee, and K. Schwan. Distributed shared abstractions (dsa) on multiprocessors. *IEEE Transactions on Software Engineering*, 1993.

[4] P. Homburg, M. V. Steen, and A. S. Tanenbaum. Distributed shared objects as a communication paradigm. In *In Proc. of the Second Annual ASCI Conference*, 1996.

[5] O. Krieger, M. Mergen, A. Waterland, V. Uhlig, M. Auslander, B. Rosenburg, R. W. Wisniewski, J. Xenidis, D. Da Silva, M. Ostrowski, J. Appavoo, and M. Butrico. K42. In *ACM SIGOPS Operating Systems Review*, 2006.

[6] O. Krieger, P. McGachey, and A. Kanevsky. Enabling a marketplace of clouds: Vmware's vcloud director. *SIGOPS Oper. Syst. Rev.*, 2010.

[7] M. Makpangou, Y. Gourhant, and J. pierre Le Narzul. Fragmented objects for distributed abstractions. In *Readings in Distributed Computing Systems*, 1992.

[8] Sage. Sage: Open source mathematics software. `http://www.sagemath.org/`.