CloudCAP: A Case Study in Capacity Planning Using the Cloud

Joan A. Smith[1], John F. Owen[1], and James R. Gray[2]

[1] Emory University Atlanta, Georgia USA
[2] Kronos, Inc. Chelmsford, Massachusetts USA

Abstract. Emory University Library teamed with a commercial firm to develop a prototype system for using Amazon's EC2 to properly size web application server deployment environments. This approach has been successfully applied to both high-transaction commercial environments with hundreds of thousands of users and to lower transaction digital library environments with hundreds of users. Starting with the same EC2-based product, our goal was to assess whether a similar strategy is practical for an academic library as well as for commercial systems. We examined cloud configuration and deployment costs, test preparation and analysis, and overall feasibility of this approach. Typically, for digital libraries, the user levels are significantly lower, the deployment costs are lower, and the return on investment (ROI) is not as immediately obvious. We conclude that the effort is worth the investment only (a) when there are significant repercussions from under-sizing a newly deployed digital library and (b) sufficient engineering staff are on hand to develop and debug the deployment scenarios.

1 Background

Emory University Library operates over 150 scholarly websites on a dozen small servers ranging from dual-core to quad-core hardware, with average daily visitor (non-robot) numbers in the low to mid-100's according to our Google Analytics reports. Except for rare internet outages and occasional scheduled maintenance, the sites are available and responsive 365x24x7. Using inexpensive, off-the-shelf hardware running a typical "LAMP stack" (Linux, Apache, MySQL and Perl, Python, or PHP), our servers easily handle the low-volume visitor rates typical of academic sites. Our capacity planning has therefore primarily focused on consolidation of services to reduce total hardware investment and administrative overhead for an ever-growing number of sites.

The situation changed dramatically with the launch of the Transatlantic Slave Voyages website[3] (in 2008 and again in 2009), and the new African-Origins website[4] (in 2011). Highly-popular, these digital scholarship sites are actually web applications, more complex than the digital library webs we typically host.

[3] $http://slavevoyages.org$
[4] $http://african-origins.org$

News coverage by agencies like the New York Times ramp up site visitor counts by several orders of magnitude: from hundreds per day to tens of thousands per day. Pre-publicity beta deployment proved that our existing approach would lead to embarrassing server failure and site crash. With limited funds, few hardware resources, and a shortage of engineers, we needed to accurately plan for capacity and responsive site performance.

Sophisticated tools to properly size deployment environments have been developed by academic and commercial researchers to ensure satisfactory performance and sufficient capacity of the institutional infrastructure [2],[3]. But using these planning tools is problematic for a small operation like Emory Libraries, in part because we lack the on-site availability of a variety of hardware with which to test candidate configurations. Even where license costs are not prohibitive, effective operation of these tools requires a significant investment in user-training and deployment time, which has so far not been practical for our small engineering team.

A 2009 UC Berkeley report highlighted the Cloud's "elasticity of resources" which allow businesses to meet variability in performance needs without investing in high-cost hardware [1]. However, Cloud-base deployments may be prohibited (e.g., government) or simply unaffordable for 24x7 operations (e.g., Emory University Library). In this case, the Cloud can be used to test potential physical configurations before hardware is purchased and the system deployed.

We spent several years in the commercial sector using the Cloud to size infrastructure for non-Cloud deployments and to identify and eliminate bottlenecks in the application environment. Our strategy, CloudCAP, was based on Amazon's EC2 because of its broad configuration options for computing hardware and operating systems, its ready availability, and Amazon's inexpensive operational fees. From 2008 through 2011 we applied the CloudCAP approach to Emory web applications that had abnormally high visitor rates, i.e., tens of thousands per day instead of just hundreds. This paper discusses our findings.

2 Designing CloudCAP

CloudCAP has a very specific role to play when it comes to capacity planning and performance testing: Provide the most rapid turnaround time possible for evaluating a proposed feature in the environment. Just as improving the edit-compile-debug time improves programmer productivity, reducing the time required to configure, deploy, test and evaluate a system improves product quality through increased tester productivity. Where previously a few basic load tests might be completed, CloudCAP allows more testing to occur, driving the solution towards a configuration that is both economical and responsive. CloudCAP extends the Infrastructure as a Service (IaaS) model to create a Testing as a Service (TaaS) model. It is a web-based application that interfaces with Amazon's storage (S3) and compute Clouds (EC2) to enable testers to rapidly deploy and configure both the application under test (AUT) and the test environment.

2.1 Cloud Definition

The term "Cloud" has been used to describe a wide variety of services ranging from mainframe computing centers to monthly subscription-based services. From our perspective, the Cloud has very specific characteristics. (1) It is sold on demand, not by subscription. (2) It is elastic, i.e., a user can purchase as little or as much as is needed at that time. (3) The hardware is managed by the provider, and (4) it is characterized by rapid provisioning and deployment of systems, near real-time for some Cloud services. These features are what make the Cloud attractive as a platform for capacity planning and performance testing.

2.2 CloudCAP Architecture

CloudCAP has three responsibilities. First, it facilitates rapid application deployment of both the AUT and the test agents. All software components required for both AUT and testing are preinstalled in the operating systems images we create for use by the solution. The image contains preinstalled copies of all software required by any of the nodes, such as database engine, application server, web server, etc, allowing us to maintain a single Amazon Machine Image (AMI) per operation system, independent of the type of node the image will become. All node types launch from the same image. The second responsibility is configuration of the nodes in the cluster. This configuration cannot happen until the nodes have been booted and are connected to the network. To avoid excessive polling, this configuration control is inverted, with each node asking CloudCAP for configuration data at two distinct stages in its boot process. The node's configuration scripts are hooked into the *rc.local* script on Unix style operating systems, and in the startup group policy on Windows operating systems. Finally, the tool is responsible for monitoring and aggregating performance data from the cluster.

Figure 2 gives an overview of the process from the perspective of a sequence diagram. The entire sequence takes only minutes to complete, whereas deploying physical hardware with a new configuration takes us several hours or longer. Stage-1 configuration has the nodes requesting from CloudCAP the node's specific configuration data. Once all nodes report to CloudCAP that Stage-1 configuration is complete, Stage-2 begins. Nodes learn the IP addresses of their peers enabling an application server to initialize a connection pool back to the database.

Consider a two machine cluster with a MySQL database server and a Java Servlet container, both on Windows 2003 Server. CloudCAP would first instruct EC2 to launch two copies of CloudCAP's custom Windows 2003 Server image. Once the first machine boots, its Startup Group Policy instructs it to connect to CloudCAP, and retrieve its Stage-1 instructions. The node starts configuring MySQL, and loading any customer specific data it needs. Similarly, the second node can start configuring Apache Tomcat, but it cannot actually launch it until Stage-2, when it has the address of the database connection. Once finished with Stage-1 processing, both nodes begin polling CloudCAP for peer data.

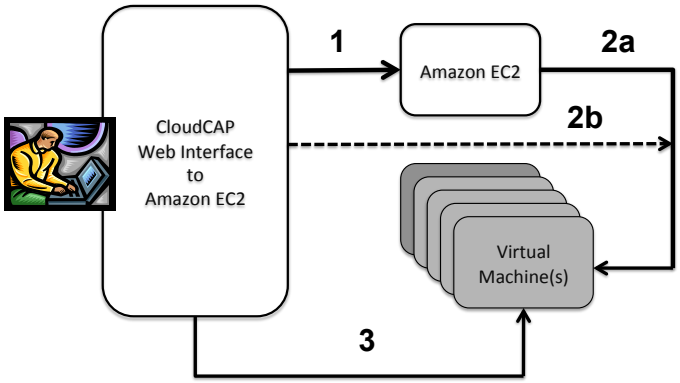

Fig. 1. CloudCAP is a front-end for Amazon's EC-2 service. A single tester can configure & instantiate a variety of tests from a single CloudCAP instance. Step (1) User initiates CloudCAP via the web interface. Step (2a) EC2 launches the virtual machines configured by the user's selection of options (2b). In Step (3) the user monitors the machines and gathers data from the performance tests.

Once peer data is provided, cluster configuration concludes with the second node configuring the database connection pool and starting Apache Tomcat.

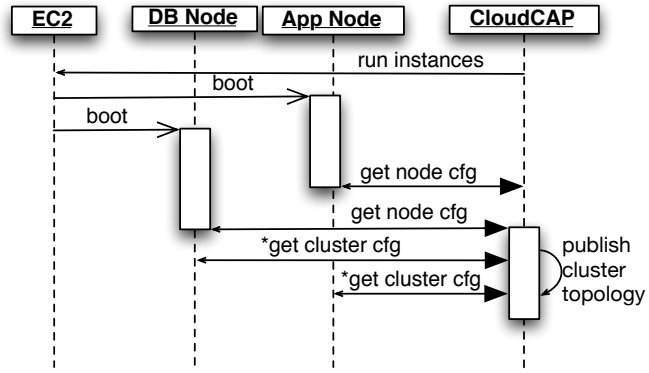

Fig. 2. Abbreviated sequence diagram of CloudCAP instantiation & operation.

2.3 Using CloudCAP

CloudCAP can be used for both performance and functional testing. It also serves to prove the deployability of the packaged software: If it does not install

properly or components are missing, this fact is immediately discovered during CloudCAP initialization. See Figure 3 for a screen shot of the simple initialization interface. The effort of writing the test scripts is the same in both traditional and cloud-based testing. However, with CloudCAP an additional up front effort is required to script the deployment of the AUT into the CloudCAP framework. The primary benefit of CloudCAP is that once the deployment has been scripted, the full range of tests can be conducted by a single test engineer, and the updates and retests can be accomplished in a matter of minutes or hours instead of days or weeks. For the Library, with a project queue far longer than its staff capacity, thorough product testing is highly resource constrained. Off-loading the test setup, operation, and results reporting to a one-button system like CloudCAP has helped us prepare more complex sites for successful launch.

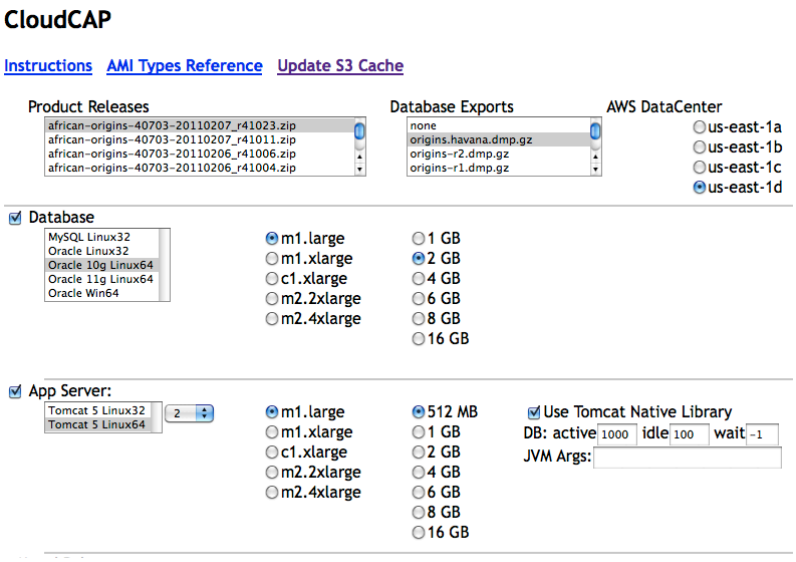

Fig. 3. Part of the CloudCAP web interface to configuration options. Product release list comes from the software's version control repository.

2.4 Case Study

The Africans-Origins website launch was planned to include major news media coverage, meaning the site would experience abnormally high traffic volume, estimated at 12,000 to 20,000 users per day based on our previous experience with the Transatlantic Slave Voyages website. We use page response time and page throughput as key metrics to measure performance since this has proven a reliable indicator of our corporate and government customers' satisfaction.

The page response times are measured in seconds per page, and throughput in pages per second. For African-Origins, our target performance level was a page response of no more than 0.5 seconds per page and a throughput of ≥ 20 pages per second at peak unique visitor levels. We used CloudCAP both to estimate the capacity of our best in-house server and to identify the minimum configuration that would meet our performance goals.

The Library is not equipped with the infrastructure to perform comparative hardware stress-tests, so we relied on CloudCAP to identify performance bottlenecks and recommend server configurations. The key to the evaluation was CloudCAP's ability to quickly configure machine images and clusters with the target environment, and the ability to rapidly deploy, and re-deploy, the African-Origins site into the test environment so as to support quick assessment of environment and application changes. Using CloudCAP, machine images and a single cluster were configured to support the application environment and the test agent environment. Test agent scripts were created that simulated unique visitor access, and emphasis was placed on the sections of the application that were expected to be areas of high-traffic. Initial testing was conducted to evaluate the proper configuration of the test environment and the deployment of the African-Origins application within the environment. After verifying that CloudCAP was properly configuring the test environment, an initial performance test was run for the purpose of establishing a baseline set of metrics.

The initial test run revealed that the application server CPU utilization was nearly 100% and that system saturation occurred at a level of only 30 unique visitors, far below acceptable limits. Further tests indicated that the Levenshtein search process was the primary bottleneck. A refactoring of the search implementation to improve its efficiency increased site performance to acceptable levels. Additional tests revealed a potential saturation point with respect to database access and ongoing refinements to the application were made to minimize its impact on system performance. We continued to conduct "what-if" analysis on the site, adjusting allocated resources (CPU, RAM, etc.) until we reached an optimal configuration. Each "what-if" session took only minutes of the tester's time, whereas performing those same hardware adjustments and redeployments on systems in our labs would have taken many hours per configuration change.

3 Results

We found a number of positive results arising from CloudCAP. It allows for what-if scenarios in both topology and in tuning parameters including specific details like RAM allocations, and number of CPUs. Options can be configured to meet domain-specific needs, allowing for more flexible scenario testing. CloudCAP supports not only a LAMP stack but also Java Enterprise stacks (Tomcat, Oracle, Apache), and Windows environments (Server 2008, IIS, SQL Server). Cloud-based capacity planning and assessment provides a cost-effective means to:

– Quickly configure varying combinations of hardware and operating systems that simulate the production environment
– Quickly deploy load-test agents that can simulate varying levels and types of customer usage patterns
– Quickly conduct "what-if" analysis to determine if variances in system configuration can improve overall application performance.

Once the test environment has been configured, robust testing of the product can be conducted with minimal support from QA staff, resulting in a faster turn-around time between testers and developers and culminating in an improved end-product. The on-demand nature of the Cloud is a perfect match for the bursty nature of load testing and for overall capacity planning whether the eventual deployment will be local or in the Cloud itself. On the other hand, using CloudCAP requires a substantial engineering investment by the institution. In part this is because the API for integrating tests into the CloudCAP framework is still immature, and configuring CloudCAP to test a new product may outweigh relative benefits. In a commercial environment, the benefit accrues over a span of several product releases, encompassing the entire lifecycle of the product. In the library environment, a product may have only a single release. New sites often see a very heavy load during the initial deployment which then subsides into the more typical low-level page hits, so the institution might find it cheaper to simply "throw hardware at the problem." After the initial surge completes, engineering focus shifts to consolidation onto a shared service environment, where many applications co-exist on common hardware. CloudCAP can be used to validate that the hardware configuration of the shared environment has sufficient capacity to support consolidation.

One challenge is mapping the EC2 hardware to equipment that can be purchased from commercial vendors. Amazon characterizes compute power in terms of a synthetic benchmark, *compute units*. Amazon defines one EC2 Compute Unit as providing the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.[5] However, because there is no standard benchmark to evaluate more modern equipment in terms of compute units, translating an EC2 environment into practical hardware is largely guess work.

The next difficulty centers around the variability in hardware provided by Amazon. While in theory a test might be running on a 12-compute-unit box, other virtual machine tenants on the same physical hardware might coincidentally also be compute-intensive. Although Amazon attempts to virtualize equally, they have no predefined knowledge of the usage scenarios of the many tenants they colocate on the same physical hardware. In our experience, multiple test runs are necessary to even-out the random effects of colocation with other tenants.

[5] Cf. Amazon's description at `http://aws.amazon.com/ec2/instance-types/`.

4 Future Work

The Cloud-based testing approach has proven invaluable for integrating load testing into the workflow, but its use has highlighted some shortcomings. A high initial investment is required for each new application introduced into the CloudCAP environment. The application plugs into the CloudCAP service through a still evolving API. This API needs to be matured so that it provides a flexible and stable interface by which an application can specify its full installation process and implement its own configuration into the clustered environment. Even though some standardization of application installations exist such as the Java WAR and EAR specifications, many of the details of deploying applications in a clustered environment remain painfully manual.

Performance monitoring is an important aspect of any load testing effort. In the CloudCAP environment it is still very ad-hoc, and is based on Amazon's Cloud Watch monitoring tool and a collection of operating system specific tools (Windows Performance Monitor, top, ps etc). An ideal system would aggregate tools from the Cloud provider together with operating system tools running on the individual nodes similar to the Ganglia tool.[6] Such an aggregation would allow rapid assessment of performance, further reducing the total testing time of a particular permutation.

A considerable amount of work still needs to be done before CloudCAP can be released as an Open Source product. In part the problem is one of practicality, since each institution's environment needs to be customized within the CloudCAP framework. In other words, there is not as much general reusability as we had hoped. Nonetheless, the Cloud-based testing approach shows promise for organizations with strong in-house engineering expertise and the need to support a wide range of performance requirements or deployment environments.

References

1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, University of California at Berkeley (February 2009)
2. Bagchi, S., Hung, E., Iyengar, A., Vogl, N., Wadia, N.: Capacity planning tools for web and grid environments. In: Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools. ACM (2006)
3. Smit, M., Nisbet, A., Stroulia, E., Edgar, A., Iszlai, G., Litoiu, M.: Capacity planning for service-oriented architectures. In: Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds. pp. 11:144–11:156. ACM (2008)

[6] $http://ganglia.info$