Scalable Infrastructure for Web Applications

Systems and Scalability Team

Alex Martineau, Shawel Negussie, JianJeng Tan, Sonny Zhao

Abstract

As an application grows in compute & data footprint as well as criticality, administrators need to consider how their infrastructures can be scaled to match these needs. There are many ways to achieve scaling. These depend on the design & architecture of an operating environment, with decisions spanning the spectrum from traditional approaches to the cloud. When architecting the environment, it is important to understand the needs of your application in terms of criticality & availability. Currently, Heroku is able to adequately support the Art Gallery web application. Heroku leverages Amazon EC2, which has built-in availability & business continuity. A key advantage to this model is that there is no need to have expertise outside of the application layer. However, it may be necessary to revisit the design decision in the future. For example, Tufts may eventually achieve an economy of scale internally to the point where it may be more cost-efficient to host the Art Gallery internally. Other design choices that may be revisited in the future include frameworks, platforms, & business continuity methods.

Introduction

Today, web applications can support hundreds of millions of users. They respond rapidly to requests and exchange an immense amount of data with users from moment to moment. Social networking companies like Facebook run their web sites as their businesses. For this reason, above all else, their overriding goal will always be to ensure their offerings (i.e. their web applications) are available. This is also true for any large company or institution that cannot afford to take outages to their online services, which is effectively all of them. As more users come to depend on an application, it will need to keep up with demand. Software optimizations and new software frameworks help make the application more powerful and efficient, but that is not the whole picture. The architecture of the entire system must be designed with scaling in mind. If your application is one out of countless to become widely used, you may find yourself growing faster than you can react. For example, between 2008 and 2009, Facebook grew by 200 million active users. For that purpose, you have to have an infrastructure that is designed to scale easily and cost-effectively. Today, there are a number of ways to build a scalable environment, ranging from traditional to new approaches. There are merits to each. In this paper, we will explore these options.

Audience

This paper is intended for the IT staff (architects, administrators) at Tufts University, who will be responsible for supporting and maintaining the Tufts art gallery web site. This is also intended for the Tufts Computer Science department students in Comp120, who will be architecting and implementing the project. As the project is already entered implementation, the information in this paper may not influence the first iteration of the art gallery. However, it can be used to inform them on the tradeoffs associated with each approach. In the future, as the next generation art gallery web site is being planned, this paper can be used to assist with decision making.

Computing Platforms

The platform for an application is defined as the hardware architecture and software framework on which it runs. As such, it is a generic term that is used to describe everything from server hardware & processor architectures to operating systems, programming languages, programming interfaces, network protocols, & storage protocols. For example, IBM sells Mainframes as well as open systems OS's, along with specialized hardware, processor architectures, protocols, and software interfaces. The idea of the platform is application-centric in that it is about delivering the capabilities needed by an application to run effectively. Because of the breadth of components that could potentially be part of a platform, for the scope of this paper, we will not be looking at all of the possible platforms available. Instead, we'll explore some trends in modern scalable platforms and the different approaches.

Traditional "from the ground up"

In many cases, the infrastructure for a company is built from the ground up. This tends to be off-the-shelf servers, switches, & storage, which are architected together to serve a number of applications. Older environments tend to have specialized, vendor-specific hardware (and software) for individual use cases. In many cases, this meant paying the premium for proprietary protocols and applications. Some of this still persists today, such as the use of Mainframes for mission critical applications in banking, financials, and healthcare. However, in many cases, the modern datacenter largely consists of generic, x86-compatible platforms that run on industry-standardized protocols and models. That is, even though an environment may be heterogeneous in terms of vendor-specific hardware (e.g. Dell vs. HP servers), it will at large be able to support the same applications because of shared standards. As a result, one advantage to this model is that you have a variety of vendors to choose from depending on how their products address particular needs of your application. Competition keeps the pressure on the vendors to continue to innovate and improve upon their products.

Traditional IT infrastructure vendors tend to specialize in particular layers in the stack. For example, Dell and HP dabble in storage but are primarily sell servers. Cisco and Brocade are in turn largely switch vendors. One problem is that, in trying to preserve their relevance, at times these vendors and subsequently their products don't work very well together. Sometimes, technically feasible configurations are not supported. For example, Oracle was slow to support running its database products in VMware-virtualized environments due to competition with its own Oracle VM server virtualization product. Another issue is the cost of acquisition and maintenance. Vendors spend a lot of

R&D to make their products relevant in relation to not only their direct competitors, but also to other layers in the stack. For example, server virtualization technology threatens server vendors, who could potentially see a lighter bottom line because virtualization allows users to squeeze more from the hardware. As such, many vendors differentiate by adding “special sauce” features to their offerings. While technically beneficial, this allows the vendors to justify the premium they charge customers. Secondly, most products have their own management interfaces. As a general rule, the more critical the use case, the more sophisticated (read: complicated) the user interfaces. There are standards such as SNMP and SMI-S that standardize management interfaces, but in many cases, to gain access to advanced controls, it is preferable to operate through native management. In most shops, there are dedicated teams who manage single layers in the stack. In a relatively small team, this poses a challenge.

At the application layer, the traditional development frameworks for creating web applications do not offer the tools to scale to the needed for a modern web application out of the box. In order to do so, custom libraries need to be written.

Cloud Platforms

Recently, many software frameworks have surfaced to help address some of the problems with traditional web applications. These frameworks range in scope and functionality, but at large are there to create a scalable, available, and well-performing software infrastructure. They provide access to features through programming language-specific libraries which developers can then use in their applications. Some ambitious companies, like Google, have developed and implemented their own software frameworks which allow their applications to scale to utilize hundreds of thousands of hosts efficiently. Due to their ability to address scale effectively, these frameworks reduce the need to have “top of the line” hardware. Google, for example, utilizes commodity white-box servers. With this model, to respond to increased demand, simply add more compute nodes on the go into a running cluster. Once demand stabilizes, the extra capacity can be removed from the cluster and powered down or added to another cluster. This elastic infrastructure can reduce acquisition, operational (e.g. power and cooling), and management costs.

One prominent platform example is Google App Engine. It allows any developer to write Java or Python code incorporating Google’s APIs. Once uploaded into Google’s infrastructure, the application will be able to take on the performance, availability, and scaling qualities of the Google infrastructure. Similarly, Heroku and Engine Yard also offer scaling for Ruby on Rails applications. Although there are advantages to using an externally hosted platform-as-a-service (PaaS), there are cases where this is not an option. In the next section, we will further explore these options.

It is also possible to use software frameworks to scale in-house. There are a number of companies who develop software frameworks intended to do just this. One is Appistry’s CloudIQ platform. CloudIQ contains libraries that enable distributed computing and caching, and includes a distributed file system. It is also compatible with Apache Hadoop. For programming languages, it supports Java, .NET, and C/C++. Once you have developed your code, it can be packaged and uploaded into the CloudIQ cluster. CloudIQ creates a layer that allows for an arbitrary number of hosts to work together on a number of applications. Workload from an application can be distributed for load balancing, high availability, and

fault tolerance. The nodes are added and removed elastically from the cluster with CloudIQ Manager, which is accessible from any node in a cluster. It also tracks applications, node membership, and cluster utilization to keep you informed on when additional nodes might be needed.

SaaS

While it is possible to develop your own web applications, whether using a traditional approach or with a cloud platform, for a generalizable use case, there is likely to be a great number of existing products that can fit our needs. Looking to the cloud again, one approach is via Software-as-a-Service. With SaaS approaches, the web application is hosted externally by a company who has already developed the software stack. Fine controls, customizations, and administration are enabled via simple web GUIs. Scaling for your application is abstract and occurs under the covers (within the host's infrastructure). The features you have access to and the scale of your environment depends on the details of your monthly subscription. The advantage is that this is by far the most application-centric approach. The large degree of abstraction from hardware assets and code development means that administrators are performing simple, logical tasks that have a direct link to the experience of the end-users. For a small team relying on limited technical resources – as is the case with the Tufts art gallery staff – SaaS may be the ideal model. SaaS does share a number of stipulations with external PaaS, in that it removes visibility from the underlying infrastructure. As such, you're counting on the developer of the service to be able to optimize utilization of the physical resources (CPUs, hosts, network, storage) and to scale well for your specific use case. If your workload characteristics (e.g. access patterns, read/write distribution) fall outside of what the service is designed to handle, you may be disappointed by performance and scalability. Fortunately, since SaaS cater to specific use cases (e.g. SalesForce.com is used for Customer Resource Management), the SaaS developers are generally experts within their space and understand what is needed to handle applications that fall into their supported use cases. Interestingly, there are also companies that are enabling other companies to develop SaaS offerings, either internally or externally. For example, SaaSGrid develops middleware that allow SaaS developers to manage customers, metering, entitlements, etc, and other relevant aspects of delivering a SaaS. SalesForce.com also enables an ecosystem of Independent Software Vendors (ISVs) to build SaaS offerings. Even though certain use cases have benefited greatly from SaaS, it is still a relatively new field (even though Service-Oriented Architecture isn't new) that has yet to be widely adopted across IT as a whole. I believe we'll see SaaS continually enriched in capability and scope over the near future.

Research has yielded a number of art gallery SaaS products. For further research, see: GallerySoft, ArtApp, and Art Center Canvas.

Hosting Options

In many ways, the architecture of a platform is directly related to the hosting options. In this paper, we simply define hosting as the physical location of the underlying infrastructure. Considerations and implications include the level of control and the relative cost of each approach.

Internal hosting

Most IT departments own hardware infrastructure that can be purposed for various applications. As such, hosting the web application internally is in some ways the “default” approach. This is especially compatible with the “from the ground up” approach to building a platform; in combination, these approaches offer the maximum level of control. For use cases with specialized workloads, this allows for the greatest flexibility to customize the infrastructure accordingly. From a security standpoint, this is also potentially the safest, as sensitive information is kept in-doors. However, this is also the most heavy-weight approach. For a truly scalable infrastructure, this approach requires expertise across a wide range of domains: electrical, power, physical infrastructure, storage & storage networking, server hardware, operating systems, software & application development, administration, lab/data center management processes, etc. Power, footprint, & management costs add up. Internally hosted infrastructures are able to achieve an economy of scale that begins to offset these costs at some point. This threshold varies depending on the particular components within the infrastructure & their associated costs (acquisition, management, power, etc). For a relatively small shop, it may make sense to host a web site outside of their internal infrastructure.

Managed hosting

Managed hosting is still considered a traditional approach. In this model, your infrastructure is hosted outside of your datacenter. This is an advantage for when physical logistics pose a problem to scalability. For example, in cases where data center space is limited, migrating a web application to a managed hosting site allows the application to scale further. In general, managed hosters can achieve a high economy of scale by housing multiple customers. This can lead to cost savings when compared to the power and footprint costs for an internal deployment. Depending on the particular service, managed hosters may also offer varying degrees of on-site, infrastructure support. Generally, this support is for the physical infrastructure only. From the operating system layer and up, responsibility still lies with the hosted organization. This allows the team to retain more control over their environment. There are limits to situations where managed hosting is appropriate. For example, there may be compliance policies to sensitive data. However, some managed hosters have credentials that certify them for handling sensitive information. For example, Terremark, who does managed hosting for many US federal government agencies, has Top Secret clearance and has implemented a number of processes to secure their data centers.

“The Cloud”

Cloud infrastructures, such as Amazon and Rackspace are a logical extension to managed hosting. Many managed hosters (e.g. Terremark, Rackspace) also tend to have cloud infrastructure offerings. They fall into three buckets: Infrastructure-as-a-Service, Platform-as-a-Service, and Software-as-a-Service. For example, Amazon’s EC2 is an IaaS offering, whereas Google App Engine is a PaaS. Successful cloud computing vendors have reached sweet spots for their offerings and in many cases are cheaper than both in-house and managed hosting approaches. Generally, these environments are highly virtualized in one form or another. As such, these offerings are elastic and flexible, with variable cost structures that scale with footprint. Management also tends to be simplified.

When exploring different cloud offerings, be mindful of the Service Level Agreements for their offerings. The cloud spans the spectrum of IT needs, with offerings ranging from commodity & public to hybrid and private clouds. These generally map to the needs of companies of varying sizes, from startups to Enterprises. Google App Engine for Business is able to provide an SLA with 3x9's of availability (99.9% uptime), which may cater to small to medium businesses and business support applications, whereas Enterprises often require 5x9's of availability. Depending on these SLAs, costs associated with cloud offerings may vary dramatically. Thus, understanding the needs of the application in question is very important. For example, for the Tufts art gallery today, having 3x9's or even 2x9's may be sufficient. However, as demand (and criticality) increases, it may be necessary to re-assess the use of these cloud offerings.

Service Levels

Service levels define the characteristics of an operating environment as well as the metrics by which these qualities are measured. At the highest level, service level objectives encapsulate the expected levels in terms of availability & performance. In detail, these objectives are defined by metrics such as response time, IOPS, throughput, etc. These are formalized into service level agreements which describe the ramifications of not meeting the service levels. In a traditional environment, especially for internally hosted environments, SLAs are loosely defined & tend to be "softer". However, meeting SLAs is especially important for externally hosted environments that represent business transactions between companies. In these situations, not meeting the agreed-upon service levels will generally mean penalties for revenue in one form or another. The type of service level required will also affect the available options in terms of hosting & platform.

Criticality and Tiering

One perception is that it is always more advantageous to get the highest available service level for an application. However, this must be measured against the cost associated with service levels. The cost associated with service levels can scale rapidly & quickly reach a point that is not cost-efficient for what is actually needed. Thus, it is very important to understand the criticality of your application. There are a number of tiers for an application. In general, these can be classified as mission critical, business critical, and business support. Mission critical applications are core to the organization & directly tied to revenue generation. In the case of Tufts University, this may be the eBill tuition payment system or the main University website. Business critical applications are those that ensure the continued operation of the organization. An example of this is the Trumpeter Email servers (although for larger corporations, email can also be considered mission critical). Business support applications are an ecosystem of apps. In our case, Tuftslife.com may be considered a support application. Depending on the scope of the Tufts Art Gallery website, it could fall within a spectrum. Factors to consider include: Expected number of users & specific use cases. For example, if use cases include tracking donor information or perhaps even facilitating online donations, then the application could be considered business. However, if it is only used by a relatively small set of users, then it can be placed on a lower tier. If the focus is on the art gallery & exhibition use cases only, then this may be downgraded to business support.

The mapping of tier to the specific service level objectives is dependent on the organization (Tufts) & its budget.

Availability

One of the metrics by which service level can be measured is availability. Generally, this is put in the framework of 9's. For example, 5x9's of availability represents 99.999% availability. In a given year, 5x9's represents roughly 5 minutes of downtime. 5x9's is usually reserved for the most mission critical applications for an Enterprise. For Tufts, the mark for mission critical is likely lower e.g. 4 or perhaps even 3x9's (52 minutes and 8.76 hours of downtime, respectively). For the Art Gallery, depending on scope, may land between 2 to 3x9's. Generally, adding a single 9 of availability will dramatically increase the cost of the solution. For example, building an environment to support 5x9's of availability could cost millions of dollars in hardware & infrastructure alone. In contrast, services like Google App Engine for Business or Amazon's EC2 are significantly cheaper because they only guarantee 3x9's of availability. As a consequence, the application is less protected (see Amazon's recent down time).

Generally, availability is increased by adding additional, redundant components into the operating environment. There are many layers to ensuring availability: Adding additional VMs into an application cluster protects against logical failure; additional servers protects against hardware failures; redundant storage & app-level network fabrics to ensure that continued communication between components in the stack; redundant storage devices, data backup & RAID protection ensure the availability of information. Other considerations for availability include failover scenarios & procedures for planned & unplanned downtime, failure domains & geographical separation between areas for disaster recovery, and the number of copies kept for each data device.

As a note, the current plan of record for the Art Gallery, in which the site will be hosted on Heroku, will offer 3x9's of availability (Heroku is built on Amazon's EC2).

Business Continuity Methods

There are a number to ensure the availability of data. Depending on the platform & hosting options, business continuity is abstracted away from the application administrators.

RAID & Data Mirrors

Fundamentally, the availability of an application will come down to the availability of the data. For that reason, for almost all applications, multiple copies of the same data are kept. This is done through a number of methods. One of the prominent approaches is through RAID. RAID stands for Redundant Array of Independent Disks. RAID is the general name for a number of disk-level data protection schemes that ensures that data is not lost despite failure of disk drives. RAID is abstracted away from the front-end, SCSI representation of the storage devices: For any SCSI initiator, a RAID'ed storage device appears as any other disk. Here are a number of popular RAID protection types:

RAID 1

RAID1 is effectively an identical copy of the data on one or more disks. This means that despite failure of n-1 disks, the data will still be available. Obviously, the effective logical to raw capacity utilization depends on the number of copies. For example, a classic RAID 1 configuration with 2 disks has 50%

utilization. RAID 1 is often used when performance & availability of data is more important than cost-effectiveness. Thus, it is often reserved for top-tier applications. At a more granular level, you may place critical data, such as database logs, on RAID 1 storage devices to maximize the availability of data.

RAID 5

RAID 5 utilizes striping & parity calculations for availability. Parity data is computed via XOR operations. In the event of a disk failure, data is reconstructed from the extra parity data. RAID 5 can tolerate the failure of a single disk, the data for which can be reconstructed. However, during the rebuild time, it is vulnerable to a second disk failure, which will then lead to data loss. The cost-efficiency of RAID 5 depends on the ratio between data and parity. For example, in a RAID 5 configuration involving 4 disk drives, the logical, usable capacity will occupy ¾ of the actual consumed capacity. The more disks involved in a RAID 5 RAID group, the more cost efficient in that the ratio between logical to raw capacity will continue to rise. However, as more disks are added, the failure domain for the RAID group also rises, thus creating a greater chance for data loss. RAID 5 is suitable for a large number of use cases on standard, Fibre Channel or SAS drives. However, the relatively lower availability of data limits of the practicality RAID 5 on slower drive media such as SATA – because of slower rebuild times of these drives, RAID 5 can leave create a larger window of vulnerability to data loss.

RAID 6

RAID 6 builds on top of RAID 5 by adding a second parity. In doing so, RAID 6 devices are able to tolerate dual disk failures. Parity calculations do add overhead to IO workload, however. Thus, compared to RAID 5 and RAID 1, RAID 6 is caters least to performance. However, it is able to achieve greater availability & in many cases, equal cost-effectiveness as RAID 5.

Picking the right RAID

For the Art Gallery, RAID 5 is likely sufficient. It represents a middle ground between the performance of RAID 1 and the cost effectiveness of RAID 6. RAID 5 would be suitable for content retrieval & the general web site, where optimum performance is needed. For archival purposes, where a large number of high definition JPEGs need to be stored for long durations, RAID 6 storage devices can be utilized. The fact that they're rarely accessed means that the relatively slower performance will not have a significant impact to end-user experience.

Data Mirrors

RAID tends to be rigid design choices made at the beginning of deploying an application. Though newer storage technology has allowed data to be easily and non-disruptively moved between RAID protection types, the use of higher level data copies is also prominent today. In today's cloud infrastructures such as Amazon & Google, the number of copies kept for logical data can be adjusted on the fly. The number of copies kept defines the criticality of the use case and as an implication will have variable cost & performance characteristics. For example, data can be kept in geographically dispersed locations. Not only does this allow a greater resiliency to disasters, but also allows for the lower latency access to data.

Snapshots & Clones

For the purpose of this paper, clones are defined as full copies of data, whereas snapshots are space-efficient logical copies. Both represent the logical data from a point in time for the application.

Snapshots and clones are generally used for business continuity & as “hot backups”, ready for use to restore back to recent point in time in case of corruption & need for immediate recovery. A common strategy is to have a fixed number of rolling clones or snaps spanning the duration of a number of days. That way, administrators can restore data back to known states at various times of the day/week. They are also used for business intelligence. For example, a clone of a customer database may be taken, to be used to run analytics against to better understand patterns across these customers.

The chief difference between clones & snapshots is that clones are full copies, whereas snapshots only use a fraction of the capacity of the original device. Snapshots often utilize the copy-on-first-write (COFW) mechanism: At the creation of a snapshot, it would actually consume no real capacity, but is ready for host access immediately. The way this works is that snapshots’ logical block addresses are actually pointed to the data on the original device. As a host writes to the disk, data is copied at write-time from the original device to the snapshot. As this occurs, the capacity consumed by the snapshot will increase over time. Another difference between clones and snapshots is that, should the original device fail, full, recoverable backup is only available through clones. This is because snapshots may consist largely of pointers back to the original device, which in this scenario has suffered a failure.

Another concept is Continuous Data Protection, or CDP. CDP works by journaling host writes against a set of storage devices (In concept, similar to database logs). In so doing, CDP enables administrators to “turn back time” for data over a virtual continuum.

For the purpose of the Art Gallery, a small number of such point-in-time copies of data may be appropriate for emergency needs.

Backups

Backups are kept to ensure that, should production data be corrupted or lost, that recovery is still possible for a time-consistent copy of the information.

Backups tend to be space efficient and cheap, because generally it is compressed, deduplicated, & placed on cheaper media. Traditionally, archival backup has been placed on cheap media, such as tape. However, tape is unwieldy and slow to recover from. Today, an alternative is slow, cheap SATA disk drives. After archival, drives can be spun down to save on power (Massive Array of Idle Disks).

Disaster Recovery

Disaster Recovery (DR) represents methodologies that mitigate the effects of datacenter- or geographical-level disruptions. DR represents some of the highest cost components to ensuring application uptime. Traditionally, this means having multiple “passive” duplicates for every hardware component at the primary site. This, in addition to the networking infrastructure between sites, leads to very high costs. In today’s larger clouds where infrastructure & applications are scalable over large distances, additional infrastructure investments can be made “active”, which allows for greater saving. Since this is already taken care of by these clouds, applications hosted on the cloud do not generally need to worry about disaster recovery.

Often, effective Disaster Recovery strategies dictate large distances between sites to ensure availability – it would be useless to have additional sites if they're all taken down by the same disaster! At larger distances, applications need to consider performance (latency) & data consistency.

For the current Heroku deployment of the Art Gallery application, Disaster Recovery is built-in to the base infrastructure as provided through Amazon EC2. However, even if the Art Gallery is moved off in the future, Disaster Recovery will likely not be needed for the application.

Conclusion

There are many options to scaling the infrastructure for a web application, spanning the spectrum from traditional infrastructures to the cloud. It is important to understand the needs of your application when architecting the operating environment. While there are many aspects to this, one of the core value propositions of the cloud is in its ability to encapsulate these needs. This frees up the end-customer to focus on application-level capabilities & reduces the need for specialized expertise in individual areas. For this reason, Heroku is a suitable platform for the current scope of the Art Gallery. However, should this scope grow in the future, there may be a need to revisit this design choice, which will have implications on the frameworks, platforms, hosting options & business continuity methods employed.