imaginea white paper

Building Applications for the Cloud – Challenges, Experiences and Recommendations

Web applications need to be highly reliable. They must scale dynamically, as users and data volumes increase. Scaling up offers the biggest challenge, and many have taken it on by adding more resources. However, this can be a very expensive proposition for many organizations. This paper examines the issues involved in building such applications and also explores alternate technologies such as master-slave replication, sharding and denormalization; and also looks at datastore implementations used by commercial players, which can be used without resorting to expensive vertical scaling techniques. In the light of the current attempts to move applications to the Cloud, these technologies hold great promise. The paper also briefly describes the relevant experience of engineers at Imaginea.

Overview

Web applications need to be highly reliable with services running without any disruption. They must scale dynamically, as users and data volumes increase. Scaling up offered the biggest challenge, and many have taken it on by adding more resources. However, this can be a very expensive proposition for many organizations. This paper examines the issues involved in building such applications, and also explores alternate technologies such as master-slave replication, sharding and denormalization; and also looks at other datastore implementations used by commercial players, which can be used without resorting to expensive vertical scaling techniques. In the light of the current attempts to move applications to the Cloud, these technologies hold great promise. The paper also briefly touches upon the experience of Imaginea engineers in these areas.

Introduction

Building applications for the cloud is has become a specialized art. Rapid approaches towards adoption of cloud computing infrastructure, the pay per use models and ability to leverage the platforms as a service benefits have made applications to be cloud deployable applications a substantial requirement. However technical considerations and challenges that encircle the effort to have an application cloud scalable, deployable and usable are plenty which requires an expert at work.

Practical insights into database issues, security loopholes, performance management, resource optimization, memory management, load balancing and ease of use are some of the key areas to be carefully considered when building a cloud scalable application. To leverage the advantages of the cloud post application deployment, one needs to have a deep understanding of cloud mechanics, orchestration and architecture.

Considerations

When an application is to be built for operating over the cloud, some technical aspects that are of paramount importance, and which require serious consideration are:

- High reliability
- Dynamic scalability
  - Millions of Users
  - Volumes of data
- Low Latency
- Across all layers
  - Database
  - Messaging
  - Web

Scaling up the database is the most critical part of the effort when developing a cloud-based application.

Challenges

Practical insights into database issues, security loopholes, performance management, resource optimization, memory management,

load balancing and ease of use are some of the key areas to be carefully considered when building a cloud scalable application. To leverage the advantages of the cloud post application deployment, one needs to have a deep understanding of cloud mechanics, orchestration and architecture.

However, vertical scaling has limitations, and is also expensive when it is considered for cloud servers. Since one hits a wall of 1gbps maximum speed with vertical scaling, it does not serve the purpose. Moreover, as you scale vertically, the number of page faults and disk seeks increase rapidly due to the increase in the number of reads.

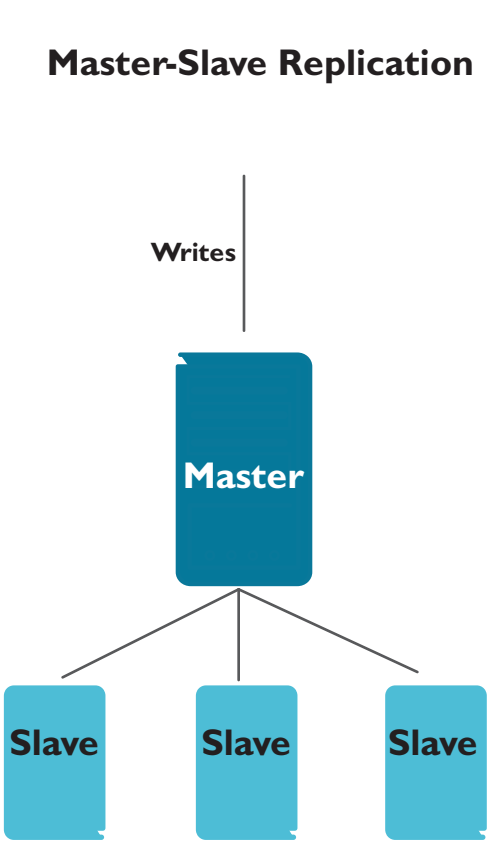
Another key challenge is ensuring there is no single point of failure (if the main server crashes, what can one do?)

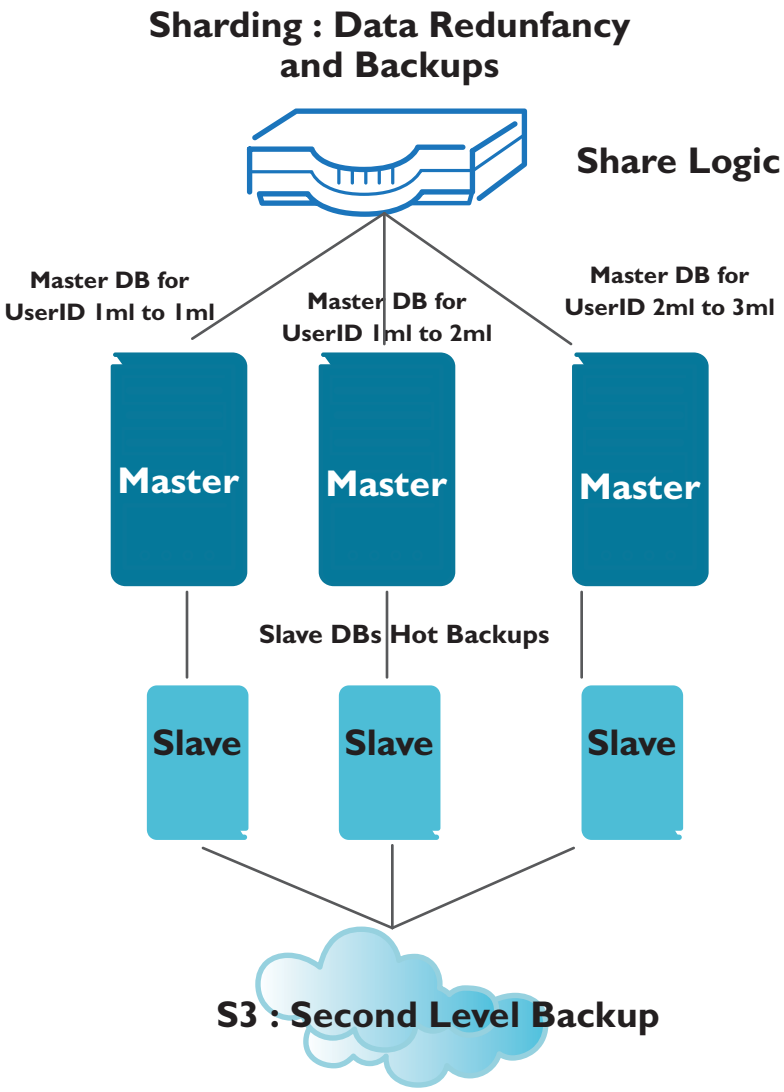
Experiences and Recommendations

On the web, applications need to be highly reliable with services running without any disruption. They must scale dynamically, as users and data volumes increase. Scaling up offered the biggest challenge, and means we have to add more resources. Scaling vertically by adding more CPUs and Disks is an option. A dating website scaled up their application to handle over a billion requests per month by moving to a 512GB RAM, 32 CPU machine. But this is an extremely expensive option, with machines of such high-end configurations costing anywhere around 100K USD. The company could have used another option, i.e. run their applications on commodity hardware, scaling horizontally by adding more boxes, as the need to scale up arose. But an even better option could have been to move the applications to the Cloud.

Applications need to scale on all the tiers- web, messaging and database. Scaling databases is the most challenging as there are hardly any databases built for scale, and which can transparently provide for application scalability with minimal changes. As disk I/O performance is the primary factor driving database scalability, scaling vertically by adding high-end disks with greater speeds and replication is worth the effort. Affordable RAID 6 or RAID 10 disks can be used to improve disk performance. However, RAID disks have an upper limit on disk transfer speeds- between 200MBps to 1GBps, which limits scaling.

The other option is to add more database instances, with master-slave replication strategy where a master handles writes, and replicates data to multiple slaves. MySQL supports master-slave scale-out configuration where the data gets replicated transparently to the slaves. When the application spends most of the time in reads, the application scales as the reads can be served from any of the slaves. An example of such a deployment is seen during user registration at dekoh.com, where all new user sign-up requests, which involve write to the database, are routed to the master, and the login requests to the slaves. Since user login occurs frequently as compared to user registration, the configuration mentioned here scales well. But the master doesn't scale when there are more writes





and when it also adds a slave lag as the data gets replicated. To overcome the slave lag, data partitioning based on a certain criteria known as sharding, has been carried out here.

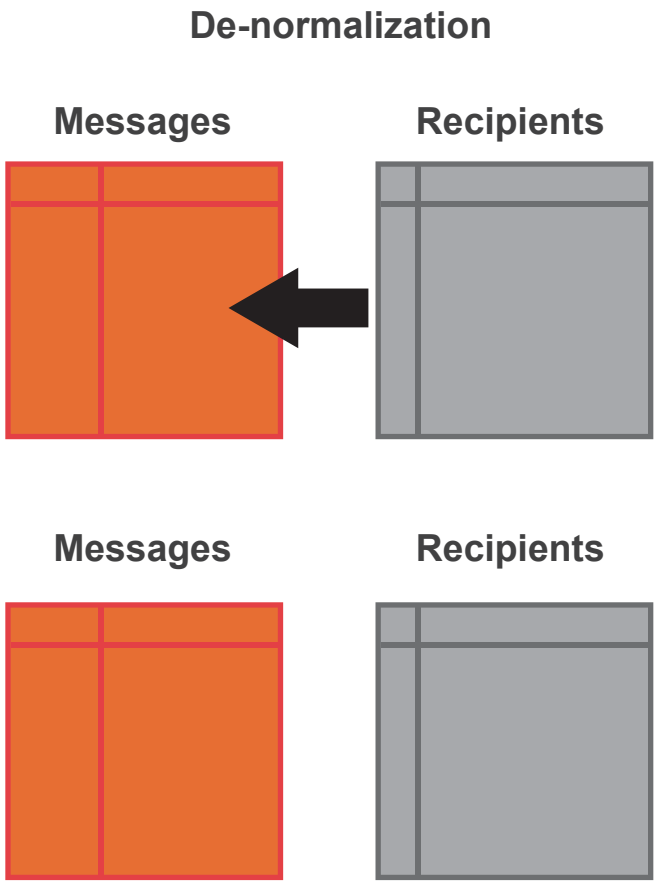
Sharding and De-Normalization

Partitioning data across masters would distribute writes to different instances and both the reads and writes scale well. Flickr moved from pure replication to the sharding mechanism to be able to scale. To achieve reliability, replication should be used along with sharding with lesser number of slaves to overcome the slave lag. Both, replication and partitioning schemes require certain level of changes to the application's architecture, as the application should be programmed to handle the reads and writes to database instances differently. Choosing an appropriate sharding scheme would be the key in determining performance. There are different sharding schemes, which can be categorized as Vertical sharding, Range based, Key or Hash-based, Directory sharding, and so on.

Sharding schemes are chosen based on the type of data that needs to be scaled. For example, if we have a customer centric application with customers from different regions, data pertaining to the customers can be sharded into regions and stored into different database instances. This type of partitioning is known as vertical sharding. But, when the customers of a particular region grow, there would be more load on that particular instance. Each type of sharding scheme discussed can be adopted in different situations and they have their own pros and cons. Say, key based or hash based partitioning works with a pre-configured number of instances and if we need to add more instances, requires re-partitioning the entire data which would be very time-consuming. Sometimes, more than one partitioning scheme can be used in congruence to achieve the needed scale. In our enterprise collaboration portal, we shard data into different corporates (a vertical shard scheme) and the corporate data is further sharded based on the type of activity i.e. posts, bookmarks etc.

However, partitioning data into shards adds more complexity in terms of maintaining the integrity of the data, application architecture and Joins. Any change in the partitioning scheme would require re-organizing the entire data, which would be very expensive. Joins are not possible as the data is broken up into different shards. How do we overcome the absence of joins? The trade-off is to introduce some amount of de-normalization, making fundamentals go for a toss.

De-normalization is basically bringing related, often redundant data together, to improve the query execution performance. In one of our applications, we have messaging infrastructure where messages are sent to recipients (users). Messages are stored in a 'messages' table and the 'recipients' table contains message-to-recipient mapping. To obtain recent 10 messages sent to a particular recipient, all message ids for the recipient are obtained from the 'recipients' table and a join is performed on the 'messages' table filtering the recent 10 timestamps. If there are a lot of messages sent



to a particular recipient, a join is performed on the 'messages' table with lot of rows from the 'recipients' table and then the timestamp filter is applied. Instead, if the timestamp of the message is duplicated in the recipients table, it is easy to filter out the first 10 message ids and then perform a join. With this approach the query takes lesser time to execute and minimizes disk I/O.

Approaches by Major Players

Major players like Google™, Amazon™, Yahoo™, Facebook™ and others, addressed issues related to data scalability and have come up with their own datastore implementations. Google's Bigtable™ is a distributed 'schema-less' key-value store, which was developed for the web search engine, and later was adopted as a datastore for Orkut™, Docs™, Google Maps™, Earth™ and others. Bigtable runs on top of Google File System and provides the needed scalability at its core, supporting replication and high availability at the file system level. Google App Engine offers Bigtable as the primary datastore for application developers.

Amazon's SimpleDB is a distributed key-value store which supports a SQL-like syntax for retrieving data and exposes REST API for all the operations. SimpleDB is available as a paid service from Amazon and can be very effective when working with huge amount of data. However, not many applications have yet adopted SimpleDB due to a lot of limitations enforced by the service on the size of results, comparisons, predicates used in the query etc. There is no support for transactions, aggregate functions, data types and full-text search, which are a set back for adopting the service.

Apache™ Hive, a Facebook's initiative, is a data warehouse that runs on top of the Hadoop™ Distributed file system. Facebook uses Hive to analyze historical data of users and content using brute force mechanism. Hive is not a datastore but is only used for analytics on large amounts of data. There are many more datastores like HBase™, HyperTable™, Cassandra™, CouchDB™, Voldermort™ developed for different purposes, which address the problem of scalability, faced by users, in various ways.

Making the Right Choice

- Different data-stores address different problem spaces
- Identify what best suits your application

Imaginea specializes in ensuring the right solution is adopted for the need. Cloud scaling and design of cloud apps has been our strength. In the sections below, we share our experiences gathered over the years designing different cloud applications.

Imaginea- Building Applications for the Cloud

Imaginea has been one of the early adopters of Elastic Compute Cloud (EC2) web service powered by Amazon. We have helped build products for the cloud. As early adopters of cloud computing, we have successfully resolved challenging technical problems that are commonly encountered when building cloud-ready applications.

Imaginea has firsthand experience in developing products for social media that have handled significantly large number of users, with sizeable data being transacted by these users. We have also designed systems to scale using technologies like sharding and optimized social media applications for scalability, fault tolerance and other important performance characteristics.

Imaginea has a rich history, as a part of Pramati Technologies, the leader in building software products in India since 1998. Along with the flagship Application Server product for the Enterprises, Pramati also developed Web 2.0 and Enterprise 2.0 products, Dekoh™, SocialTwist™ and Qontext™. Imaginea teams have extensive experience in building these applications.

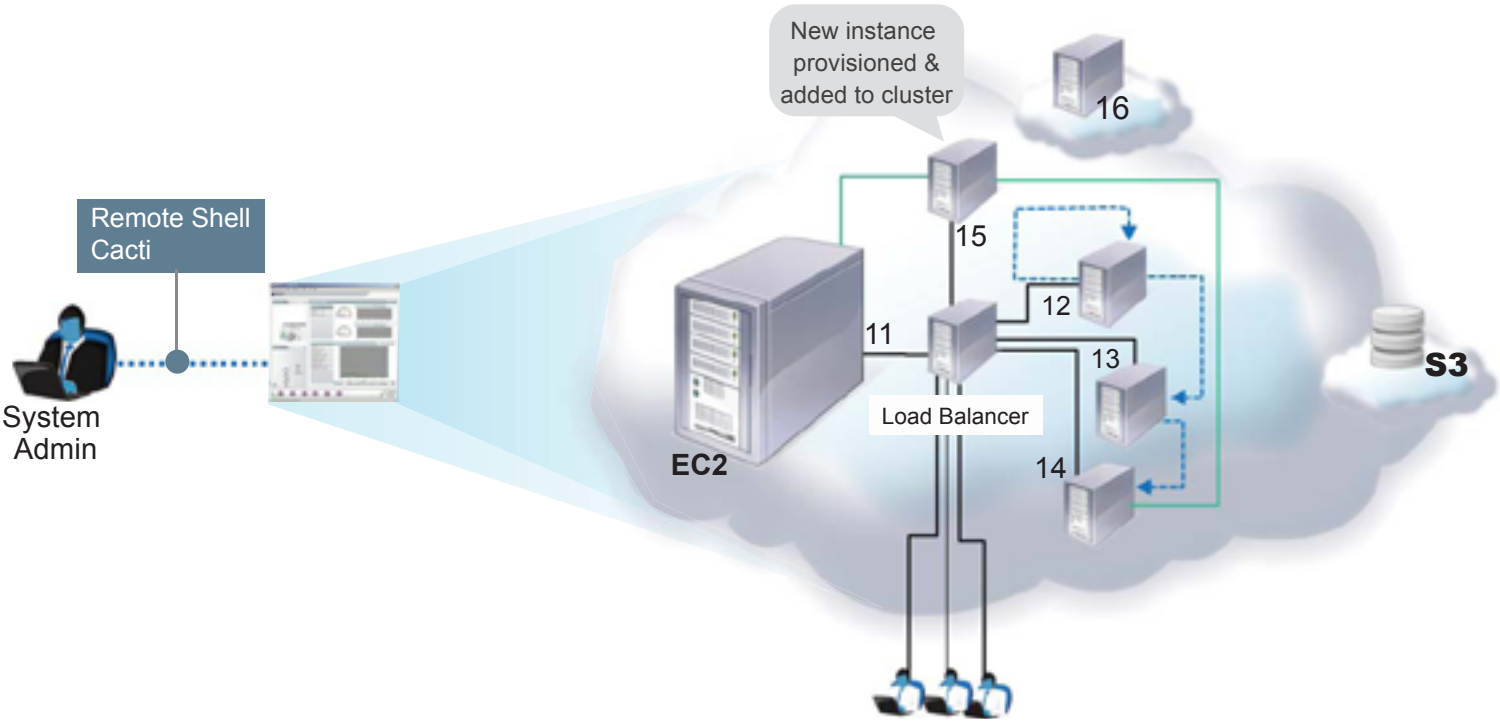
Scaling the Web-Tier – Some Key suggestions

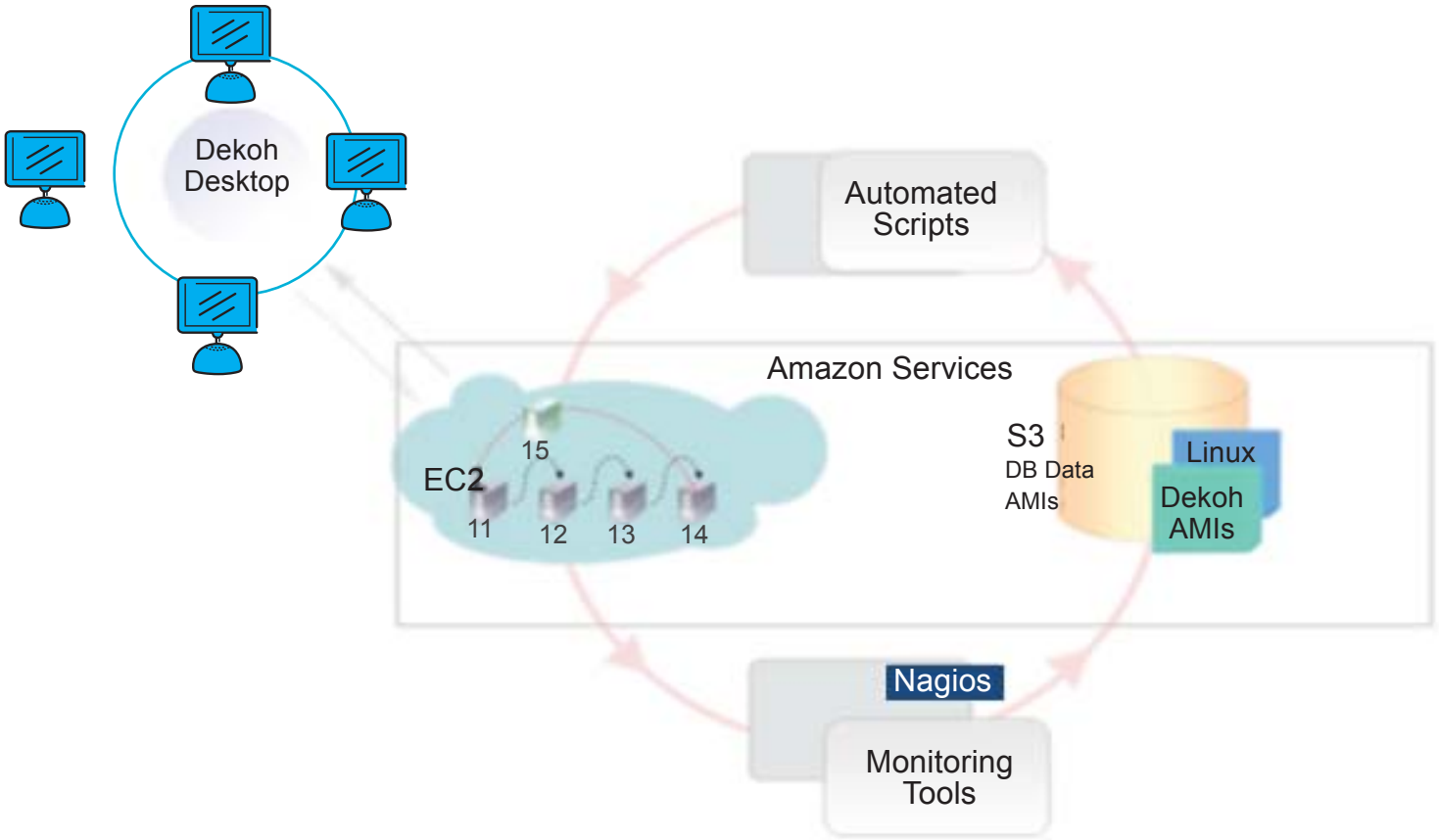
- Avoid sessions
  - Easier with AJAX UIs
- Run only data services
  - Static content requests delegated to S3 or CDN
- Load balancing
  - DNS based
    - Use Elastic IPs
  - Web Loadbalancer (LB)
- PUT requests are idempotent, POST request aren't
  - LB will retry idempotent methods

Dekoh – Web and Desktop Integration

Dekoh, is a free desktop platform, which allows users to organize all media in one place, on a desktop and share it with their contacts. Dekoh Desktop is a cross-OS desktop platform that combines and offers the best of the desktop and the web to users. It enables secure sharing of local content and applications deployed on the Dekoh Desktop with assured privacy. Pramati used Amazon's Elastic Compute Cloud (EC2), to build the Dekoh Network that can scale on-demand so that when thousands of potential Dekoh applications access the hosted architecture, they can benefit from the combined power of EC2 and Dekoh. These users now can access rich, desktop-quality applications, even offline, without worrying about application performance even when

Applications built on Dekoh Desktop are accessible via browser locally and remotely





Key Considerations

Clustering on EC2 is of absolute importance when building a scalable application on the cloud. Dynamic load balancing is an important activity that needs to be carefully managed to ensure a scalable application on the cloud is available.

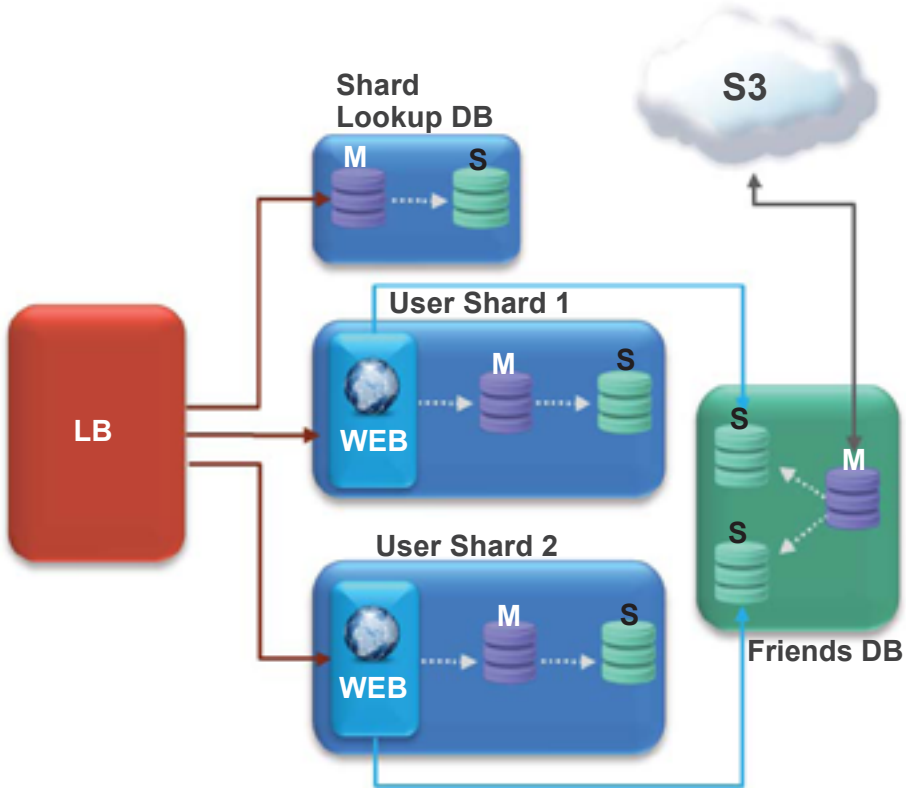
Key Learnings

- Run only data services
  - We use images to keep the binaries
  - App upgrades- app+db integrated
    - Our apps are ALL just war files
    - DB schemas managed a la dekoh callbacks
    - Predefined handing available for sql scripts
    - Other dependencies - we can use servlet app-start callbacks
      - Example: mysql version, config files, other libraries
    - All other binaries (like mysql) managed explicitly
    - All required config files are in this image
    - Image can be one or more apps
- Backup scripts (local to s3)
- Monitoring - nagios is again, script-driven
  - Script can be anything "invokable" from Linux
  - We use- sh, or sh-launch-java
- Cacti- to see trends

  - System trends
    - CPU, MEM, Network, Disc, etc
  - App trends
- S3 used for backups alone
- 160GB access for all app data including mysql files
- No NFS mounts or shared data files used today
- Database used is MYSQL
  - Using MYSQL db replication for performance on read access
  - Use separate read and write data sources
    - Abstracted by our own java API

myPicks2008 – Social Gaming on the Cloud

myPicks2008 is an online game for predicting winners of the 2008 Olympics that was built as a Facebook application on Sun's Zembly platform:



Architecture and Key Learnings :

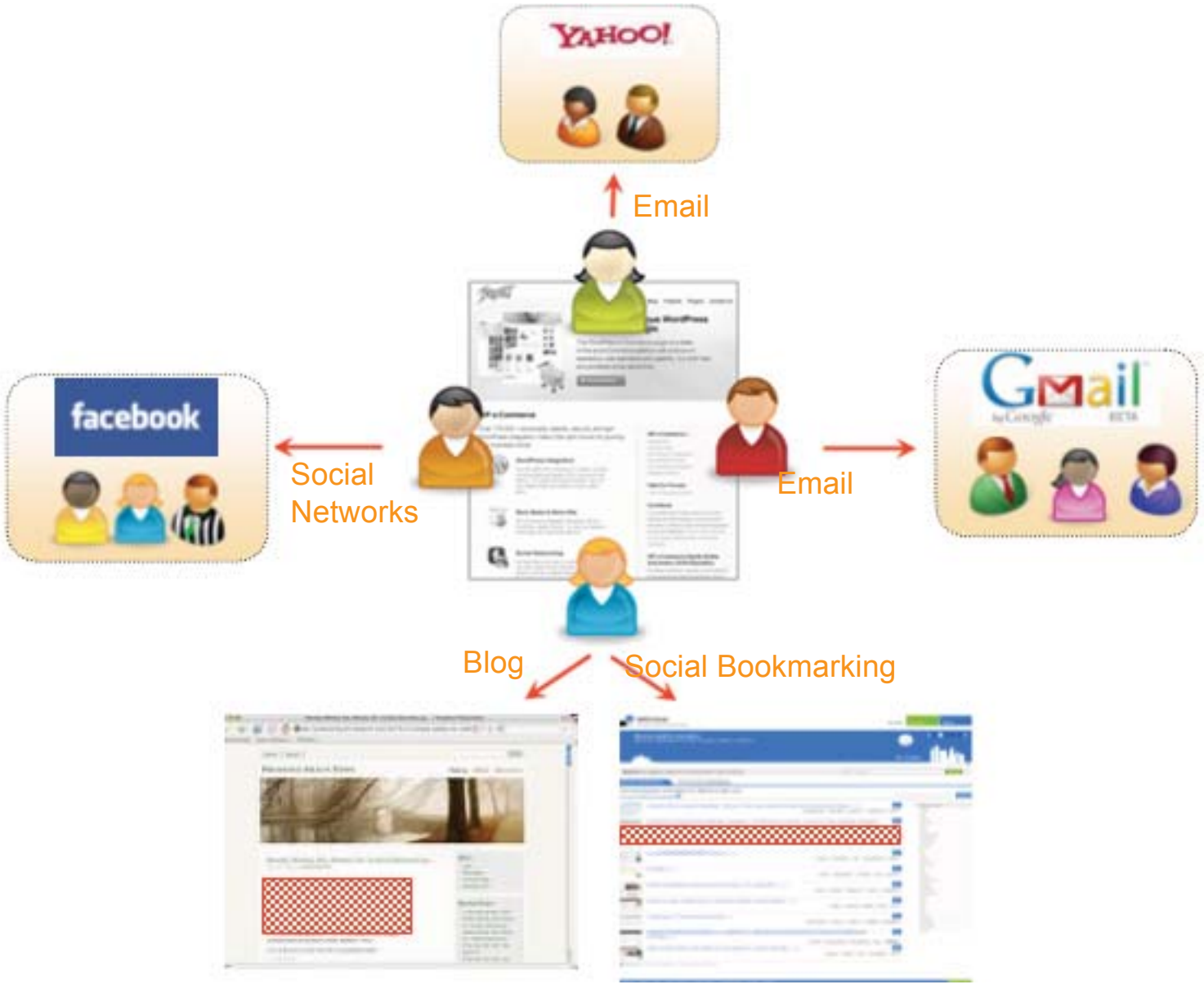
- User DB is sharded (with hot standby)
- We used lookup table for sharding
  - Cached in memory instead of memcached
  - 10MB cache size for million users
  - Sharding logic plugged-in as NodeChooser in Pramati Web Loadbalancer
- Friends DB uses replication instead of Map-Reduce
- Attaching the webservers to a shard reduced the number of connections to the User DB

SocialTwist: Word-of-Mouth Marketing

SocialTwist brings widget applications for use by communities and corporations. Tell-a-Friend™, from SocialTwist, is a word-of-mouth marketing tool used over the web. It is the only word-of-mouth solution that enables context-rich word-of-mouth email, IM, social, blog and bookmark referrals. This marketing tool is powered by Amazon's EC2 and S3 for high availability.

Why on the Cloud

- Unpredictable users and loads
  - Need to provision and de-provision hardware programmatically
  - Need to control hardware costs
  - Dynamic Scalability across layers
- Unpredictable users and loads
- Control over infrastructure

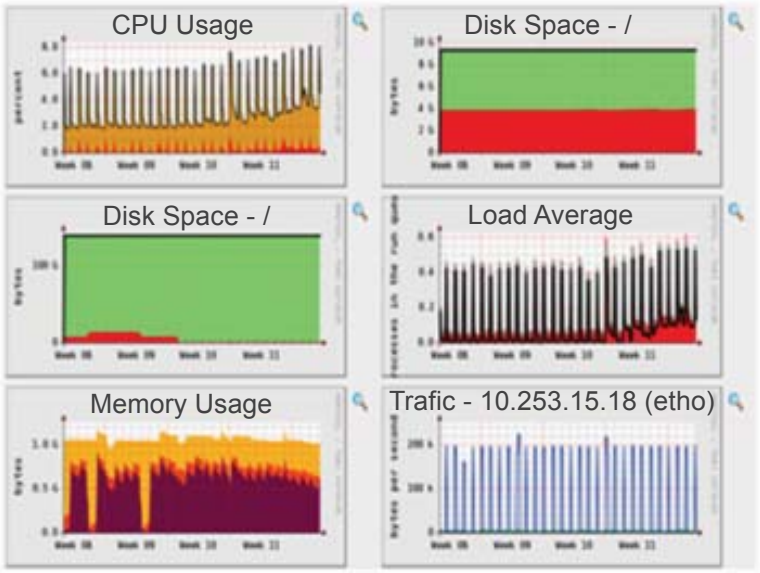


Key considerations [Webserver and Database Scaling]

- WebServer - More than 1 million widgets served till date
- Database - Usage statistics for each user
- Dynamic Content Caching
- Pramati Web Load Balancer - Minimal sessions
- Amazon S3 for static content
- Read – mostly data (config etc) - memcached
- High concurrent writes (stats)- sharding based on user id
- Connection pooling

Key Learning

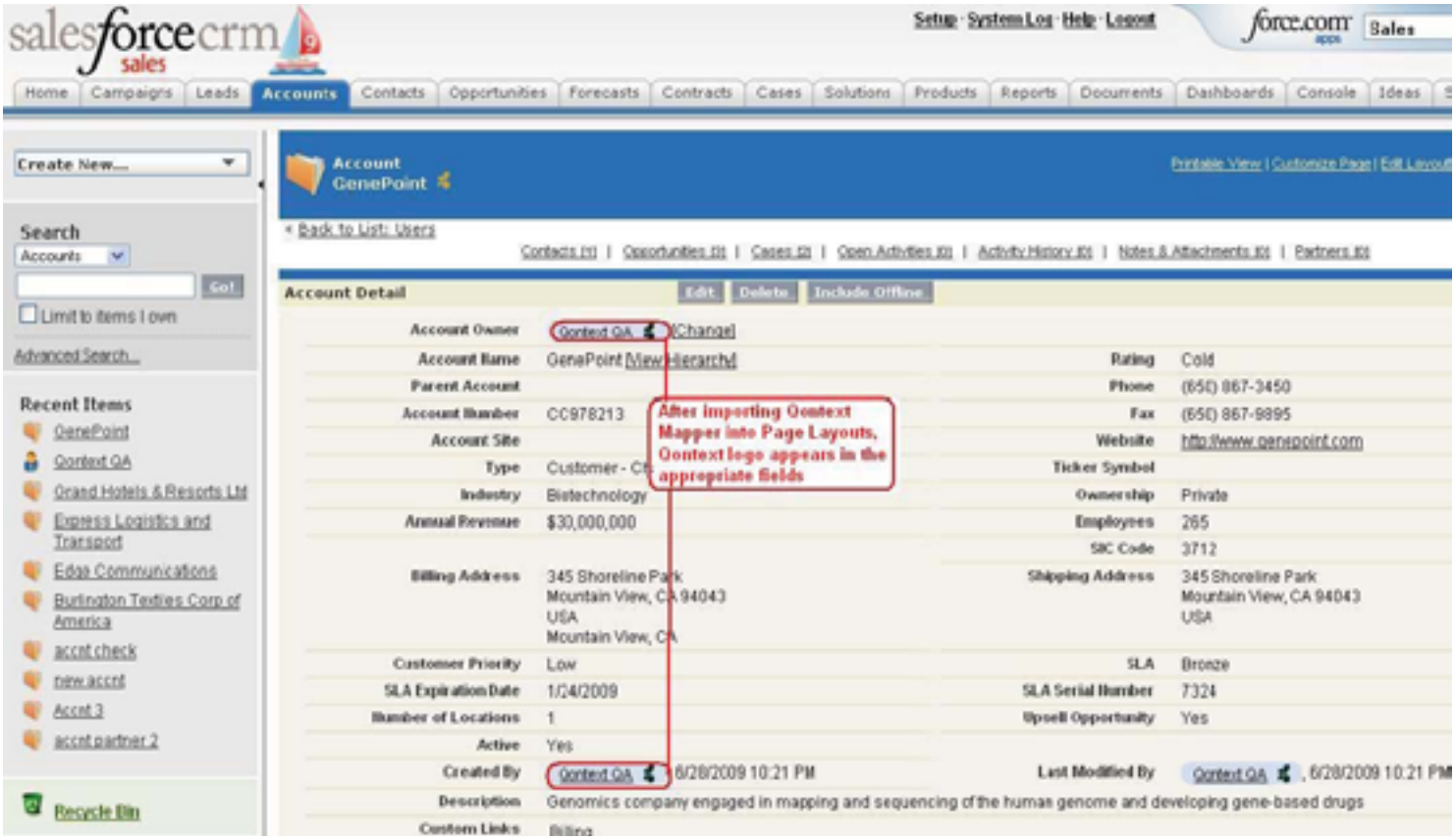
- Base AMI Type
  - LB
  - Web
  - Database
- Base AMI Features
  - Backup
  - Join the monitoring system
  - Join the cluster
  - Publish health stats
  - Preconfigured services
- Create AMI for each instance type in the application
- Monitor health of instances
- Create instances on demand using the AMIs
- Instances join cluster



Qontext – Enterprise 2.0

Qontext, is a portal for enterprise collaboration with widgets for discussions, posts, documents, bookmarks. Qontext is built to provide Enterprise 2.0, social networking capabilities on a SaaS model. Qontext works on the Widget model, whereby widgets can be triggered to collaborate with other users, based on the context. This enhances collaboration, making it more meaningful, and keeping it more relevant and contextual.

Qontext is integrated with Salesforce™ to enhance the CRM application, so that it works in sync with an enterprise social networking tool. This integration of a CRM with a contextual collaboration tool, has created an integrated application that can break the boundaries of such enterprise applications, which used to be more functional. It helps their users to be more proactive and collaborative.



A human factor has been added to the enterprise functionality, where decisions and discussions on an issue have become more social for an enterprise tool. The confluence of enterprise social networking and a CRM with a widget-based, contextual, collaboration model brings a revolutionary approach in enterprise knowledge sharing. This ushers in a collaborative work style, where collective decision making aided by technology, brings success to teamwork.

The Road Ahead

Scalable datastores are still evolving and there is no one datastore that best suits all purposes. Recently, a lot of new, non-relational databases have cropped up both inside and outside the cloud. One key message from this is that if you want vast, on-demand scalability, you need a non-relational database. A cloud platform without a scalable data store is not much of a platform at all.

Hence, to provide customers with a scalable place to store application data, vendors have only one real option. They have to implement a new type of database system that focuses on scalability, at the expense of the other benefits that come with relational databases.

Conclusion

The arrival of Cloud Computing has completely changed the face of computing and how software is used. Cloud Computing has ensured the era of software a service instead of it being treated like a commodity, sold to users. A new way of selling software as a service evolved, which is popularly referred to as SaaS. Along with Cloud Computing, Web 2.0 evolved, which facilitated higher levels of user interaction, leading to a growth in social networking portals. The obvious impact of social networking can be seen by the number of people using it. Social networking portals have infused new approaches to associations in society and have created an area where conversations can take place beyond geographical limits in an asynchronous mode. Building applications for the Cloud requires a specialist, who can understand all these aspects and deliver a world class application.

If you want to build an application that cloud-scales, leverage our expertise. Please get in touch with us by contacting us at sales@imaginea.com

imaginea