Scalable Architecture on Amazon AWS Cloud

Kalpak Shah
Founder & CEO, Clogeny Technologies
kalpak@clogeny.com

2nd IndicThreads.com Conference On Cloud Computing

3-4 June 2011, Pune, India
Traditional Hardware Model

Customer Dissatisfaction
(Insufficient Hardware)

Large Capital Expenditure

Infrastructure Costs

Time

Predicted Demand
Actual Demand
Traditional Hardware
Opportunity Cost

Architect to scale on-demand and provision as per current requirements

Ideal model for unpredictable and variable loads
Scalability Requirements

- Increase in resources $\rightarrow$ Increase in performance
- Predictability
- Low Latency
- High Reliability
- Dynamism: Number of users, volume of data, skews
- Operational efficiency
- Costs should not scale 😊

{Elasticity, Scalability, Resiliency}
Scalability Perspectives

What needs to scale?
- Compute
- Memory
- Network
- Storage
- Monitoring
- IO Latency
- Provisioning time
- Backup / Restore times
- Failover
- Ops

Vertical scalability

Horizontal scalability

Scale across geographies

HPC workloads

Data Processing workloads
Vertical Scalability

- When scale is predictable and linear
- When you do not want to spend on re-architecting the application or deployment
- Increase instance sizes
  - 1 – 33.5 EC2 Compute Units
  - 613MB memory to 68GB memory
  - Size or number of EBS disks
- HPC Instances
  - 10 Gigabit ethernet
  - Higher IOPS for EBS disks

Limitations....
Scaling multi-tier stacks – 1

✧ Service Oriented Architecture

- Loosely coupled
- Standard service contracts
- Web Services
- Enables independent tiers for deployment & management

✧ Messaging / Queue layer
Scaling multi-tier stacks – 2

- **Amazon SQS**: Reliable, scalable, hosted queue; exposed as web service
- **RabbitMQ**: Open-source HA messaging system, clustering support
- **BeanstalkD**: Simple, fast work queue

◊ **Clustering Application Servers**

- JBoss App Server, IBM WebSphere Application Server
- Add or remove nodes on the fly – automate through scripting
- Stateless behavior can be added when necessary
- VPC does not work across availability zones (AZ) – in the pipeline though
Elastic Load Balancing, Auto Scaling

**Amazon Elastic Load Balancing**
- Distributes incoming traffic to your application across several EC2 instances
- Detects unhealthy instances and reroutes traffic

**Auto-Scaling**
- Enabled by CloudWatch: Monitoring, custom metrics, free tier, graphs and statistics
- Rule-based automatic scaling of your EC2 capacity
- Based on metrics including resource utilization, software stack metrics or custom metrics
- N+1 redundancy
Monitoring & Logging

✧ **Amazon CloudWatch**
  - Monitoring for AWS cloud resources & applications
  - Collect and track metrics – CPU, latency, request counts, custom metrics

✧ **Monitoring with your own tools**
  - Using Hyperic or Nagios for monitoring specific layers of your stack or to leverage existing investments

✧ **Logging**
  - No dependency on instances – copy necessary logs to S3 periodically
Databases - Replication

- Master-Slave Replication (MySQL, Oracle RAC)
- Writes on master
- Reads distributed across slaves
- Works well in read mostly scenarios
- Slave lag issue
Databases – Sharding - 1

- Partition data across masters
- Writes & Reads are distributed
- Application needs modification
- Needs choice of partitioning strategy for uniform data distribution
Databases – Sharding - 2

**Issues**

- Joins cannot be performed across shards
- Application modification can be expensive

**Example**

- Evernote uses database sharding – localized failures, no need for joins
- Each shard handles all data & traffic for about 100,000 users

http://blog.evernote.com/tech/2011/05/17/architectural-digest/#
Databases – Amazon SimpleDB

- Schema-less distributed key-value store
- Highly reliable and scalable (redundancy across geos)
- Automatic indexing of columns
- API based global access
- Supports multiple values for key/attributes
- Eventual consistency or consistency – speed or consistency?

Limitations
- No joins, No transactions, No aggregators, text searches

NoSQL
- MongoDB, Cassandra, Redis
Databases – Amazon RDS

- Relational Database Service (RDS) from AWS
- Scale your DB layer with minimum administration
- MySQL and Oracle supported
- Import existing databases & no changes to applications
- Multi-AZ deployments supported
- Manages backup of your database and enables restore from DB snapshots
Reserving Scalability

- **Reserved Instances**
  - AWS has finite hardware capacity
  - Provisioning times can vary
  - Use few reserved instances to “book” capacity in advance (also take advantage of lower prices)
  - Can be done across availability zones to ensure DR

- **Larger EBS disks**
  - Create larger EBS disks to ensure better performance
  - Netflix creates 1TB disks in this manner
Scalability using PaaS

✧ Amazon Elastic Beanstalk

- Platform-as-a-Service with deployment, capacity provisioning, load balancing, auto-scaling & application health monitoring
- Application versioning support (rollback if needed)
- Uses EC2, S3, RDS, SimpleDB, Load Balancer, CloudWatch
- Retain control of your infrastructure if desired

✧ Other PaaS products

- CumuLogic, CloudBees, DotCloud, PHP Fog
- Java, PHP, RoR, MongoDB, MySQL……
Elastic MapReduce (PaaS)

Hosted Hadoop Framework

- Manages job flows & provisioning of all infrastructure
- AWS Console to create & manage workflows
- Supports custom jars, Hive, Pig, streaming & processing in multiple languages/stacks
- Debug & profile jobs
- Run across geographies

Data processing, analytics

*Scalable Managed MapReduce Platform*
Automation for managing scale

💎 CloudFormation
- Templatize your stack
- Predictable provisioning of your stack

💎 RightScale
- Sophisticated cloud management platform
- Templates, automation, orchestration, portability

💎 Tools, Connectors, Enablers
- Automated orchestration & setup
- Snapshot management
- Monitor security groups and firewalls
There are some limits...

- EC2 has limit of 20 instances
- S3 has limit of 100 buckets
- Simple Email Service (SES) has a daily sending quota

NOTE: All of these limits can be increased or waived by requesting AWS. Ensure to do this before you hit the limits in production.
Scale but minimize costs - 1

◊ **Use of Reserved Instances**
  - Commitment for upto 1-3 years with some upfront payment
  - Actual usage cost is much lower
  - If used for more than 6 months in a year, can be 30-45% cheaper than on-demand instances

◊ **Reduced Redundancy Storage**
  - Reduce costs by storing non-critical data at lesser redundancy and lesser availability/durability of 99.99%

◊ **Instance Sizes**
  - Run some smaller instances as part of clusters
Scale but minimize costs - 2

- **Data Transfer beyond 10TB**
  - Consolidate AWS accounts so that higher usage translates to saved costs. $0.15 upto 10TB and $0.11 beyond 10TB.

- **Identify extra capacity**
  - Use monitoring to identify unused capacity & optimize

- **Spot Instances**
  - Bid for unused capacity – choose your maximum price
  - Get more within your existing budget
Questions?