Fedora on AWS

Fedora on Amazon Web Services (AWS)

What is AWS?

Amazon Simple Storage Service (S3)

What is S3?
A system which allows storage and retrieval of data using web services interfaces (SOAP and REST). The data is stored on Amazon servers which perform backup and replication activities to assure no data loss.

Benefits

- Scalable storage (no limitations on number of files stored)
- Reliable storage (SLA specifies 99.99% availability)
- No file system limitations (like number of files in a folder, etc)
- No up-front costs, pay only for what you use

Limitations

- Speed is limited by the network making I/O much slower than local disk
- File size is limited to 5GB

Cost

- Charges are based on amount of space used, bandwidth used, and number of requests
  $0.15 per GB-month of storage space
  $0.10 per GB / month data transfer in
  $0.17 per GB / month data transfer out (cost goes down with scale)
  $0.01 per 1,000 PUT, POST, or LIST requests
  $0.01 per 10,000 GET requests

Using S3

- Authentication to S3 is done through use of an Access Key ID and Secret Access Key pair, or through use of an X.509 certificate
- Connecting to S3 is done via SOAP or REST. Amazon provides libraries in several languages to make use of S3 easier. Third-party libraries are also available; these tend to be more full-featured.
- All files stored in S3 are considered objects. Every object can have metadata associated with it as a list of name-value pairs.
- All objects are stored in buckets, which are similar to folders on a file system. Bucket names must be unique across all of S3. This is necessary to allow access to buckets using subdomains (i.e. if you have a bucket named fedora, the content in that bucket will be available at https://fedora.s3.amazonaws.com/)
- The name of each object is its key. Keys must be unique within a bucket. Keys can be named in such a way as to suggest a folder structure.

Resources
Main page - http://www.amazon.com/gp/browse.html?node=16427261
Getting Started Guide - http://docs.amazonwebservices.com/AmazonS3/2006-03-01/gsg/
Developer Guide - http://docs.amazonwebservices.com/AmazonS3/2006-03-01/
Technical FAQ - http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1109&categoryID=55
Java REST library for S3 - http://developer.amazonwebservices.com/connect/entry.jspa?externalID=132&categoryID=47
JetS3t, open source toolkit for S3 (includes a Swing GUI for working with S3 files) - https://jets3t.dev.java.net/

Amazon Elastic Compute Cloud (EC2)

What is EC2?
A service which provides computing capacity in a highly customizable environment running on Amazon hardware. EC2 provides servers which can be customized and saved, allowing any number of instances of that server to be started and stopped as needed. The processing power, memory capacity, and storage space of each server instance is selectable at boot time, providing a flexible computing environment.

Benefits

- Server images allow starting/stopping pre-configured server instances
- Server instances can be started/stopped as needed, requiring only slightly more time than is required to boot/shutdown the OS
- Root access to running server instances
- Provides the ability to run server instances in multiple locations by specifying the "Availability Zone" of the instance on startup

- Elastic IP addresses allow you to map a static IP to any running instance, so your IP stays consistent even as server instances come and go
- No up-front costs, pay only for what you use

Limitations

- Linux is the only supported OS at the moment, though others could be used through emulation.
- Storage on a server instance goes away when the instance is stopped (this can be resolved by using EBS)

Cost

- There are 5 instance types, 2 of which are considered high-CPU instances. Prices vary based on the instance type used.
  $0.10 per instance-hour for small instance - 1.7 GB memory, 1 compute unit, 32-bit
  $0.20 per instance-hour for high-CPU medium instance - 1.7 GB memory, 5 compute units, 32 bit
  $0.40 per instance-hour for large instance - 7.5 GB memory, 4 compute units, 64-bit
  $0.80 per instance-hour for xlarge instance - 15 GB memory, 8 compute units, 64-bit
  $0.80 per instance-hour for high-CPU xlarge instance - 7 GB memory, 20 compute units, 64-bit

- Data transfer rates
  $0.10 per GB / month data transfer in
  $0.17 per GB / month data transfer out (cost goes down with scale)
  $0.00 per GB / month data transferred between instances in the same availability zone

Using EC2

- Server images are called Amazon Machine Images (AMIs). AMIs are stored on S3, then registered with EC2.
- A server Instance is an AMI which has been started and is running. All instances of an AMI are identical at startup. All changes to an instance and any information stored on an instance are lost when an instance fails or is shut down.
- Amazon provides a set of command line tools for working with AMIs and instances. Instances can be accessed via SSH.
- AMIs can be created from scratch on a local machine or by modifying an existing AMI. There is a listing of publicly available AMIs, some provided by Amazon, others provided by various companies or individuals.
- To create an AMI from an existing AMI you create an instance of the existing AMI, log in and modify it as required, create an AMI bundle (i.e. produce the image based on the instance), upload it to S3, then register the AMI with EC2.
- Parameters can be passed to an instance on startup, either as a text string or as a file. These parameters can be used to start or configure an instance in a certain way.

Resources
Main page - http://www.amazon.com/gp/browse.html?node=201590011
Getting Started Guide - http://docs.amazonwebservices.com/AWSEC2/2008-02-01/GettingStartedGuide/
Developer Guide - http://docs.amazonwebservices.com/AWSEC2/2008-05-05/DeveloperGuide/
Technical FAQ - http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1145&categoryID=100
EC2 Setup Video (Windows) - http://s3.amazonaws.com/AmazonEC2Tutorial/AmazonEC2Tutorial.wmv
ElasticFox, excellent Firefox Extension for EC2 - http://developer.amazonwebservices.com/connect/entry.jspa?externalID=609
List of public AMIs - http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=101

Amazon Elastic Block Store (EBS)

What is EBS?
A persistent storage solution for Amazon EC2 instances. EBS provides block level storage which can be mounted directly to an EC2 instance and used as local disk.

Benefits

- Up to 1TB disk storage per volume, unlimited volumes
- Persist beyond the life of an EC2 instance
- Designed to perform faster than EC2 instance storage
- Can be disconnected and moved between instances
- Automatically disconnects when an EC2 instance fails or is shut down
- Many EBS volumes can be mounted to a single instance at one time
- Reliable: EBS instances are automatically replicated (within its availability zone) - on the order of 10x more reliable than typical disk drives
- Snapshots of an EBS volume can be made to S3, allowing creation of new EBS volumes with a known dataset
- Snapshots of a volume after the first only store deltas, keeping the S3 footprint small.

Limitations

- Can only be connected to one EC2 instance at a time
- Not as reliable as S3 (should be backed up to S3 periodically)
- No apparent access to data on an EBS volume outside of EC2

Cost

$0.10 per allocated GB per month
$0.10 per 1 million I/O requests

Using EBS

- Amazon provides command line tools for using EBS along with the EC2 tools. These tools allow you to create, attach, detach, and delete volumes as well as create and delete snapshots.

Resources
Main page - http://www.amazon.com/gp/browse.html?node=689343011
Feature Guide - http://developer.amazonwebservices.com/connect/entry.jspa?externalID=1667&ref=featured

Amazon SimpleDB

What is it?
A simplified relational database service which removes the need for a schema by limiting queries to a single table (called a domain) and allowing variable columns per row.

Resources
Main page - http://www.amazon.com/gp/browse.html?node=342335011

Amazon Simple Queue Service (SQS)

What is it?
A simple queue service made available to handle very basic queueing needs. The only functions are CreateQueue, SendMessage, ReceiveMessage, and DeleteMessage. There is no publish/subscribe support.

Resources
Main page - http://www.amazon.com/Simple-Queue-Service-home-page/b?ie=UTF8&node=13584001

What has been done so far using AWS?

Fedorazon

- Created a private AMI which starts up with a running Fedora.
- Data is stored to S3 via a a mount provided by using JungleDisk.
- Created tutorials that allow people to try out Fedora by creating an instance once they are given permissions to access the AMI
- More information:
  http://www.jisc.ac.uk/whatwedo/programmes/programme_rep_pres/repositories_sue/fedorazon.aspx
  http://www.ukoln.ac.uk/repositories/digirep/index/Fedorazon_How_to_Guides
  http://docs.google.com/View?docid=dfccf5n6_51dwmk8rdb

Integration Challenge

- Created a plug-in for LLStore which stores all object and datastream content to S3
  - Amazon S3 Storage project in the incubator
- Created two private AMIs

1. ami-69ff1b00: A 32-bit image using Fedora Core 6 OS which includes a checkout of the current Fedora trunk and all of the tools necessary to build /install/run a Fedora server.
2. ami-73fa1e1a: Starting from image above, adds scripting to update and build the code, install the server based on an install.properties file included with instance startup, and start the server.

- Data is stored using instance storage, which is ideal for automated testing, but not for running a real repository.
- Performed testing with storing content on EBS, but no significant integration with Fedora.
- Conducted a simple performance measure by running the ConfigB system tests on Fedora servers which were identical except for the storage location, using instance storage, S3, or EBS as the location to write objects and datastreams. Results specify the time required to run the ConfigB test suite.
  - Instance Storage: 7 min 12 sec
  - S3 Storage: 17 min 59 sec
  - EBS Storage: 6 min 45 sec

Scalability Testing

- Created a simple test to perform a continuous ingest of objects into Fedora. The goal of this test is to determine how many objects can be ingested before the system fails, and determine what that failure point is.
- The test was run on a medium-size, high-CPU EC2 instance. The MySQL database was on the 10GB OS partition, Fedora ran on the 350GB /mnt partition, and the results file was written to a mounted EBS volume. The test ended when MySQL ran out of disk space. About 16 million objects were ingested during the test.

Where do we want to go with AWS?

Ideas so far...

- Automated Testing / Nightly Builds
  - Create an AMI which can be started with a particular configuration on which unit and system tests can be run using Bamboo or another test automation tool.
  - Current AMI is a good start toward this. Would need to add capabilities to determine database preference from install.properties and then start and configure that database. May also want to determine a way to choose a JVM version other than 1.5.

- Performance Testing
  - Parameterized EC2 instances provide a good candidate for simplifying the setup and cleanup necessary for running test instances. Tests can also be run in parallel by simply starting more instances.

    - Tests will likely need to be vetted against a standard server to determine if there are any unexpected effects caused by running on EC2.

- FedoraShare
    - Create an AMI which runs a Fedora instance with content/DB/triplestore data saved to an EBS volume. Include an application sitting on top of Fedora which allows people to use Fedoa in interesting ways.

- Mediation (DSpace/Fedora)
    - Mediate between multiple repository systems
    - New repository service which takes ideas from DSpace/Fedora
    - Simple services (example: MrSID image conversion)

- Akubra
    - Store to S3
    - Using EBS if running on EC2

- Simple startup of Fedora server
    - Basic public AMI with Fedora
    - Having pre-configured Fedora servers that can be made available quickly
    - Easy way to get up and running with Fedora
    - How to handle billing issues?
    - Adding value with durable service overlay

- Federation/scaling/high availability experimentation
    - Allow starting new EC2 instances to provide higher availability, better performance, etc.

- Information modeling support

- Local use
    - Website on EC2, data on S3, etc