Amazon Web Services

Power-Tools on a Micro-Budget

Agenda

- Introduction of the services
- Ideas
- Demo
- Q&A

one simple goal :

to make you think

Why Amazon ?

Running the type of infra we're interested in

Since 1994

At a scary scale

They know web services better than most

Using them before they were called
web services

talk is Amazon centric

solutions are generic

free to pick some ideas

and re-implement them

Services

S3

SQS

EC2

won't even mention the rest

S3

Simple Storage Service

probably already read about it

mostly seen as **backup storage**

cool as such, but cooler than it seems

S3

Pay as you go **storage**

write / read / delete

unlimited objects

in unlimited buckets

from 1 byte to 5 gigabytes

S3

API through REST and SOAP

Objects served using HTTP and Bittorrent

Public / Private, using ACLs

Friendly URLs, vhosts are supported

S3

Dirt Cheap

$0.15 / gig stored / month

$0.10 / gig moved to S3

$0.18 / gig fetched from S3 (digressive)

“peanuts per hits”, but not free

transfer fee is waved to / from EC2

S3 Billing

calculating the actual fee is complex...

...but mostly pointless, it's that cheap

S3 Billing

High Volume :

50% to 90% savings

(disks, servers, colocation, staff, etc.)

Low Volume :

Lower than your electricity bill

Simple Queue Service

not much noise about it

looks rather uninteresting

SQS

Pay as you go **Messaging**

Unlimited messages

Unlimited queues

Stored for up to 15 days

REST : 8 kB / message

SOAP : 256 kB / message

SQS producer

Sends a message

Forgets it

(can keep track, but that's not the idea)

SQS consumer

Polls a queue, asking for **new message**(s)

Gets the message, which **locks** it

Acts on the payload

Deletes the message

SQS lock expiration

If a message is **read**

But **not deleted**

Something went **wrong** somewhere

SQS makes the message **visible again**

For the next consumer to read / lock / delete

SQS lock expiration

Lock has a “Time To Live”

Default for a queue

Set at read time

Tweaked over time

EC2

Elastic Compute Cloud

EC2

Pay as you go **CPU**

Dirt cheap

$0.10 / instance / hour

To save you calculating :

$2.4 / day

~ $72 / month

EC2

Linux based (Xen)

Images based on Fedora Core or Debian
Plenty of base images provided

Root access, install anything you want

Feels like a VPS, except for the specs

EC2 specs

equiv. 1.7 Ghz x86 CPU

1.75 GB Ram

160 GB drive

EC2 network

250Mb/s, said to be bursted to 1Gb/s

eth0 has a **private IP address**

NAT'd to the outside world (or not)

Boot defined **external firewall policy**

VLAN for instances sharing a policy

EC2 is unique

No setup costs

No commitment

No delay beyond the boot process

Start 1-20 instances in one command

Stop them when finished

Arrange with Amazon if you need more

⚠ Big Gotcha

Instances are **ephemeral**

No reboot

Disk will always be **freshly cleaned up**

Keep that in mind, or cry...

S3 makes a comeback

Cheap Analogy

Think of it as a cheap hotel room

All you expect from a hotel room

But don’t expect to feel at home

Leave valuables behind and they’re gone

Between guests, they clean the room back to their standard room

Short Lived...

That's where the analogy ends...

You can customise your room

But you need to think about what you always want to find, how you want to find it, etc.

Describe that to the hotel manager

Agree on a name

Custom Instances

When you book, don't just ask for
"**a** hotel room"

Ask for N rooms
configured as "**your** hotel room"

VPS like use is possible

But limited, unless you really think about it

elastic compute cloud

not “cheap on demand VPS”

</salesman>

<marketer>

What Amazon got right

Price

Simple to get started

Simple building blocks

Taking you 80% of the way

Few imposed design choices

BYOT : Build Your Own Tools

Lessons in API design

Basic, rock solid, services

Open API

“Broken” libs

Cause an itch to make them better

Listen, learn, make things better over time

Only when power-users have real problems

Don’t try to please the Blogosphere

</marketer>

<techie>

How it could apply to you

US based, latency is painful

(~120ms from cablecom.ch)

Not suitable as surfer facing infrastructure

unless it's for particularly painful tasks

or for a limited time (during upgrades, etc.)

Seeing how Amazon evolved, seeing EU based DCs is not impossible

How it could apply to you

Reduce hardware needs for data crunching

Need 35gigs of ram for a few hours ?

With 20x the disk IO of a local server ?

How it could apply to you

Reduce “lab” costs

No need to have a local lab anymore

Boot a entire lab when you need it

Concurrent access to the “lab”, boot 2...

Perfect to design and test scaling strategies

How to get started

Apply for S3

Start using S3

Apply for EC2

Read the docs while waiting for approval

Start simple

“Hot Files” Caching

Squid Proxy as a Web Accelerator (RAM only)

Cache misses falling back to S3

Static assets pointed to Squid

Leave the cache management to Squid

Store all assets to S3

Only handle dynamic content locally

Demo Background

only a small part of a much bigger picture

dirty feeds (XML / CSV / ...)

variety of sources

turned into sqlite DBs, common schema

variable size, some too big to handle

several gigs of feeds, in different languages

Challenge

index the content into Lucene

Hard Requirements

Easy to trace indexing problems

Partial re-indexing when

- the schema is fine tuned
- a feed is updated

Can’t burn a hole in my desk ...

... or my wallet

Soft requirements

No technological tie in

Ruby preferred, but Java friendly if the libs justify it (Lucene / Solr)

No financial investment, custom built cluster is out of the question

As little time to learn as possible

Being able to freeze the project while working on other things is a big plus

Solution

Sqlite DBs stored on S3

Queue with a pointer to the DB we want to split

Splitter fetches the DB

Splitter generates XML chunks with 500 entries

Splitter stores the chunks on S3

Splitter notifies pool of *Indexers* of a split DB

Indexer fetches the chunks

Indexer creates and populate a Lucene index

Indexer stores the index on S3

possibly the wrong choice of words, but...

its **maps** a DB to XML chunks

and **reduces** them

to a Lucene index

closing words

Changing the landscape

Taken separately, nothing revolutionary

Together, they open a lot of possibilities

With a very low barrier of entry

- Financially
- Technically

Don’t ignore it!

Gives potential competitors a serious edge

No commitment, flexible by design

“Cost of failure” is minimal, so a lot of mental barriers to starting a company are removed

Adsense seed money becomes realistic

3 geeks operating out of a Starbucks can start something real big