# The Cost of the Cloud

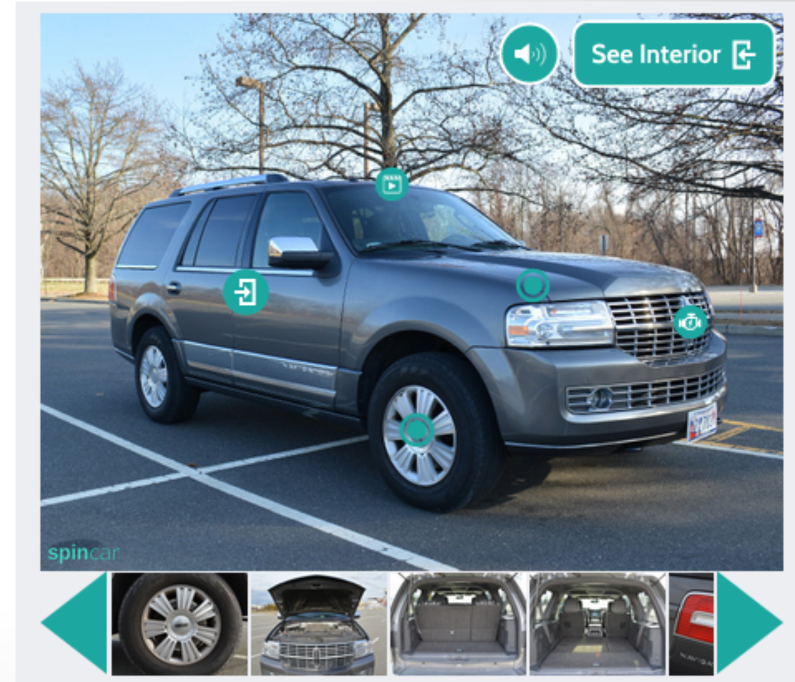
Steve Saporta
CTO, SwipeToSpin
Mar 20, 2015

# The SwipeToSpin product

SpinCar 360 WalkAround

- JPEG images
- HTML
- JavaScript
- CSS

“WA” for short



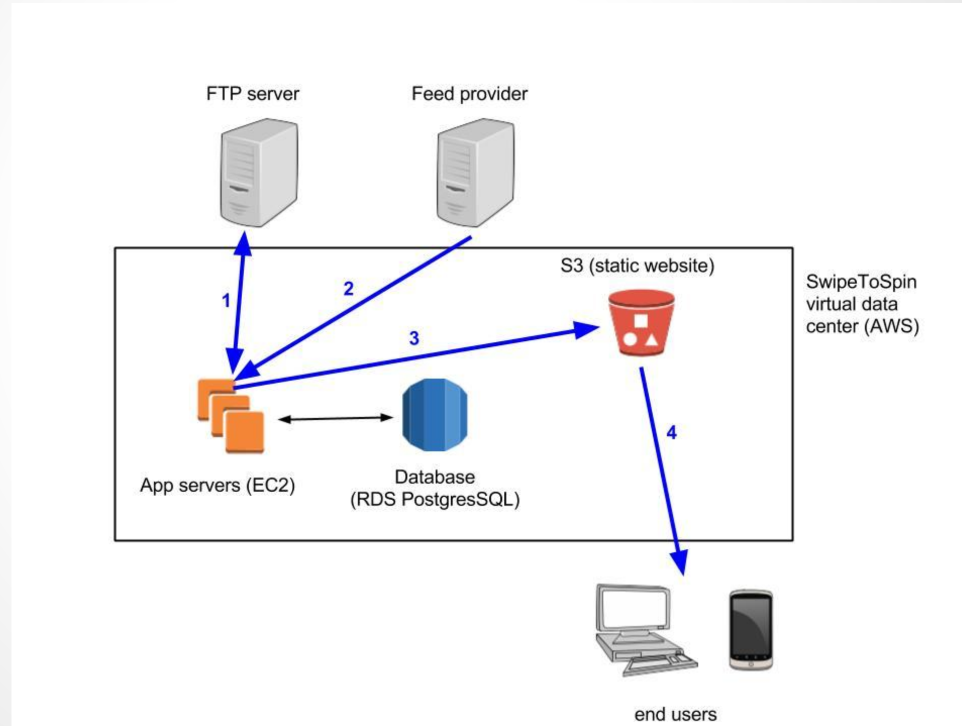
# Creating a WA

1. Download and parse CSV file
2. Download and process images
3. Upload to Amazon S3

# Serving a WA

4. Serve from S3 static website to end users

# 360 WalkAround system diagram



# Capacity planning (cont’d)

Initial assumptions (several of which turned out to be wrong)

- 100 active WAs / dealer
- 10 new WAs / dealer / day
- 10,000 website visits / dealer / mo
- 10 WAs viewed per visit
- 25 2400x1600 source photos per WA
- Size of a WA: ~50MB (~95% zoom tiles)
- Zoom enabled but used lightly

# Capacity planning: creation

- Creation = steps 1, 2 and 3
- How many servers are needed to create WAs?
- A few big servers, or many small ones?

# Capacity planning: creation (cont'd)

- How many WAs can one server create?
- What type of server?

| Step | Time (t1.micro) | Time (m3.xlarge) |
| --- | --- | --- |
| Download photos from FTP server | 4 sec | 4 sec |
| Process the photos | 13 sec | 5 sec |
| Upload photos to S3 | 4 sec | 4 sec |
| **Total** | **21 sec** | **13 sec** |

- M3.xlarge costs 14x as much, not even 2x as fast

# Capacity planning: creation (cont’d)

Is it more efficient to create multiple WAs in parallel?

- Creating *n* WAs in parallel takes about *n* times as long as creating one WalkAround
- Conclusion: queue the requests and use several low-end servers

# Capacity planning: creation (cont’d)

How many dealers’ feeds can a t1.micro instance process?

- ~10 cars / dealer / day
- (10 cars / dealer / day) x (21 sec / car) = 210 sec / dealer / day
- (84,400 sec / day) / (210 sec / feed) = ~400 feeds / day
- Might be bursty, so perhaps 100 dealers / day with acceptable queueing times
- ~0.01 t1.micro instances per dealer

# Capacity planning: creation (cont’d)

Creating WAs is cheap

- (0.01 instance / dealer) x ($0.02 / hr / instance) x (720 hrs / mo) = $0.144 / dealer / mo
- That’s right: 14 cents!

# Cost to serve WAs

Amazon S3 cost has 3 components

- Storage
- Requests
- Bandwidth

# Cost to serve WAs (cont'd)

S3 storage cost

- (50 MB / WA) x (0.001 MB / GB) x ($0.03 / GB / mo) = $0.0015 / WA / mo
- (100 WAs / dealer) x ($0.0015 / WA / mo) = $0.15 / dealer / mo

# Cost to serve WAs (cont'd)

## Requests

- Tier-2 (GET) requests cost $0.004 / 10,000 requests
- It can be difficult to estimate how many requests will occur
    - Each image, JavaScript file, etc counts as a request
    - Caching may reduce the number of requests
- I behaved as a typical user viewing one WA, then examined an AWS usage report
- ~200 requests per WA viewed

# Cost to serve WAs (cont’d)

Requests (cont’d)

- (200 requests / WA) * ($0.004 / 10,000 requests) = $0.00008 / WA
- (10,000 visits / dealer / mo) x (10 WAs / visit) x ($0.00008 / WA) = $8 / dealer / mo

# Cost to serve WAs (cont’d)

Bandwidth

- A visitor doesn’t download the entire WA. I estimated 5 MB per WA viewed.
- (5 MB) x (0.001 MB / GB) x ($0.12 / GB) = $0.0006 / WA
- (10,000 visits / dealer / mo) x (10 WAs / visit) x ($0.0006 / WA) = $60.00 / dealer / mo

# Cost to serve WAs (cont'd)

Summary

- Storage: $0.15 / dealer / mo
- Requests: $8.00 / dealer / mo
- Bandwidth: $60.00 / dealer / mo
- Total: ~$70.00 / dealer / mo

# Imperfect assumptions

The initial assumptions were educated guesses based on an unproven business model.

We can now compare to real-world figures based on several months of analytics data.

# Imperfect assumptions (cont'd)

- 100 active WAs / dealer
  - dealer != website != customer (but close enough)
  - mean = 182
  - median = 57
- 10 new WAs / dealer / day
  - mean = 4
  - median = 2

# Imperfect assumptions (cont'd)

- 10,000 website visits / dealer / mo
- 10 WAs viewed per visit
  - VDPs viewed
    - mean = 13,248
    - median = 7,356
  - WAs viewed
    - mean = 4,208
    - median = 1,959

# Imperfect assumptions (cont’d)

- 25 2400x1600 source photos per WA
    - 640x480 is more typical
- Size of a WA: ~50MB (~95% zoom tiles)
- Zoom enabled but used lightly
    - Zoom eliminated. More like 2MB.

# Amazon price decreases

EC2 t1.micro instances

- Budgeted $0.02 / hr
- Replaced with t2.micro instances at $0.013 / hr

Bandwidth

- Budgeted $0.12 / GB
- Current price is $0.09 / GB

# Actual cost to serve WAs

- As of Feb, 2015, ~100 active dealers
- Based on actual AWS bill, actual cost was much lower than the $70 / dealer / mo estimate
  - Eliminated zoom feature
  - Fewer visits to dealer websites than anticipated
  - Smaller source photos (640x480)
  - Reserved Instance pricing

# Database

- Already had a small RDS instance
- Performance was slowing as the database grew

# Database research questions

- Upgrade to a larger RDS instance? How much larger?
- Indices
  - Would adding certain indices speed up queries?
  - Would it slow down inserts?
- Does production load affect query speed?
- RDS vs EC2 + PostgreSQL
  - RDS takes care of patching, backups, etc
  - EC2 costs less

# RDS capacity planning

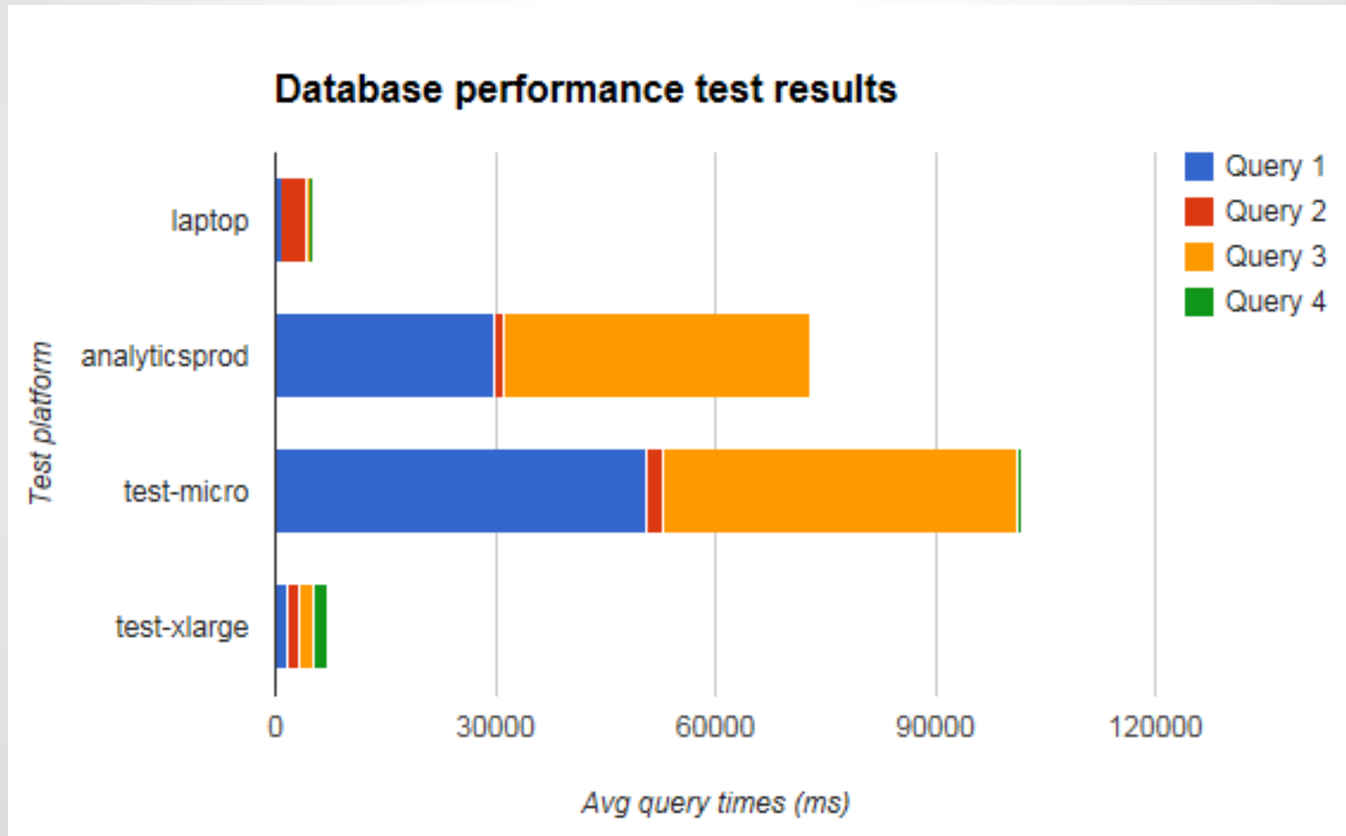
- Tested with a copy of the production database
- db.t1.micro vs db.m3.xlarge
  - $0.036/hr vs $0.780/hr
- 4 test queries based on some slow real-world queries
- Ran each query 5 times and averaged the times

# RDS capacity planning (cont'd)

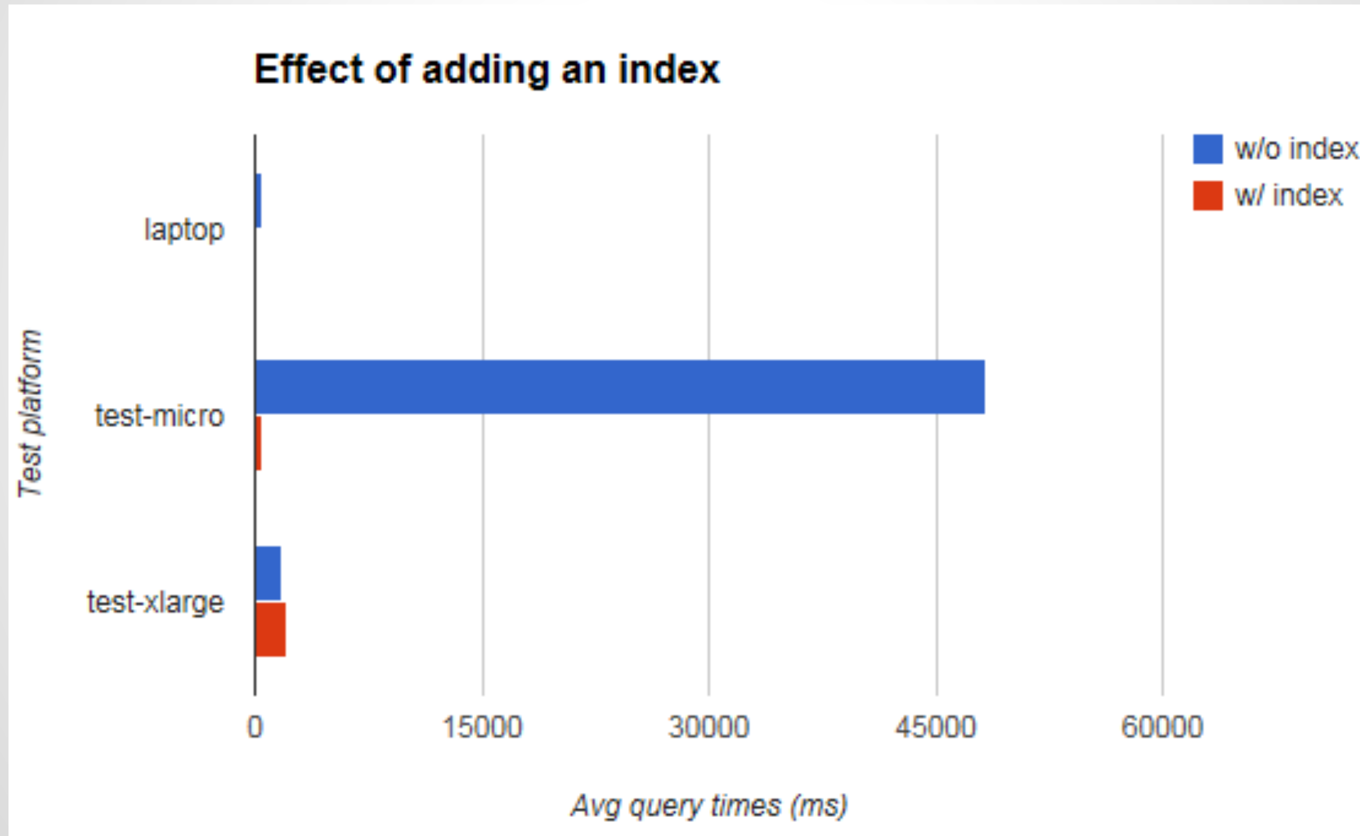
Caching can muddy the results

- Example: a query took 6,006 ms on the first iteration and less than 300 ms on iterations 2 - 5
- Defeating caching isn't a valid test, because caching exists in the real world
- Caching helped some queries more than others
- Caching helped some platforms more than others
- Conclusion: average over several iterations and hope for the best

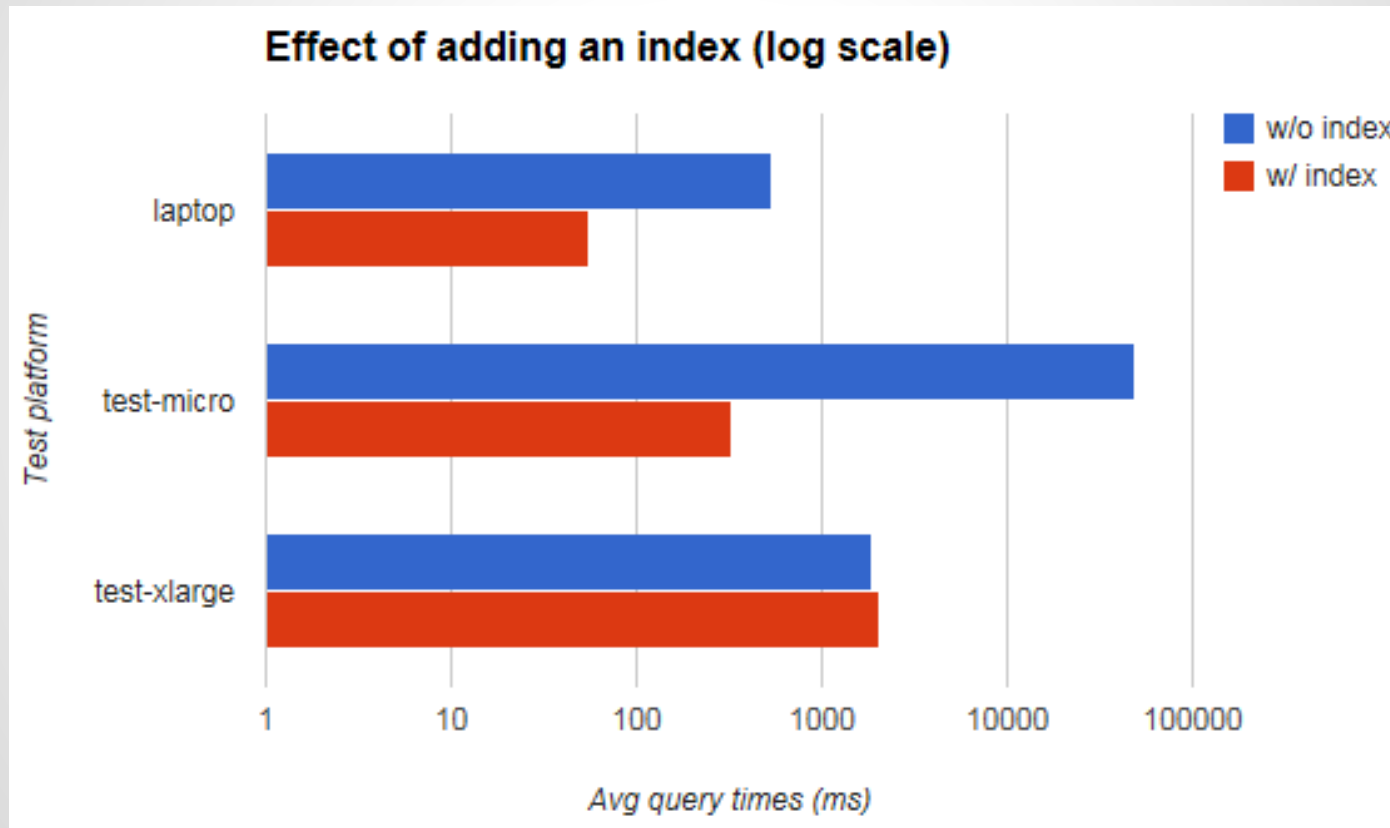
# RDS capacity planning (cont’d)



# RDS capacity planning (cont'd)



# RDS capacity planning (cont'd)



# RDS capacity planning (cont’d)

Conclusions

- m3.xlarge is an order of magnitude faster than t1.micro
- Adding an index
  - Improved read performance hugely (1 - 2 orders of magnitude), but not consistently
  - Didn’t harm write performance
- Production load had little effect
  - m3.xlarge comparable to laptop
- EC2 m3.large instance with PostgreSQL is comparable to RDS db.m3.large instance
  - Didn’t test formally, but similar queries execute in similar time

# RDS capacity planning (cont'd)

Decisions

- Added the index (and eventually several other indices)
- Upgraded to RDS db.m3.large
  - Cheaper than db.m3.xlarge ($0.39/hr vs $0.78/hr) and fast enough
- Used an EC2 m3.large instance for the reporting database
  - Patching and backups not important for this database
  - Lower cost ($0.14/hr)
  - Easier to set up cron job for nightly updates

# Other AWS costs

- Load balancers
- Extra EC2 instances for crawler, customer web UI, etc.
- Database I/O requests -- $0.20 / million, but it adds up

# Strategies to reduce cost

- Reserved Instances
  - Long-term commitment and/or up-front fee in return for a lower per-hour cost
  - One- or three-year term
  - EC2
    - All upfront, partial upfront or no upfront
  - RDS
    - Heavy Utilization only

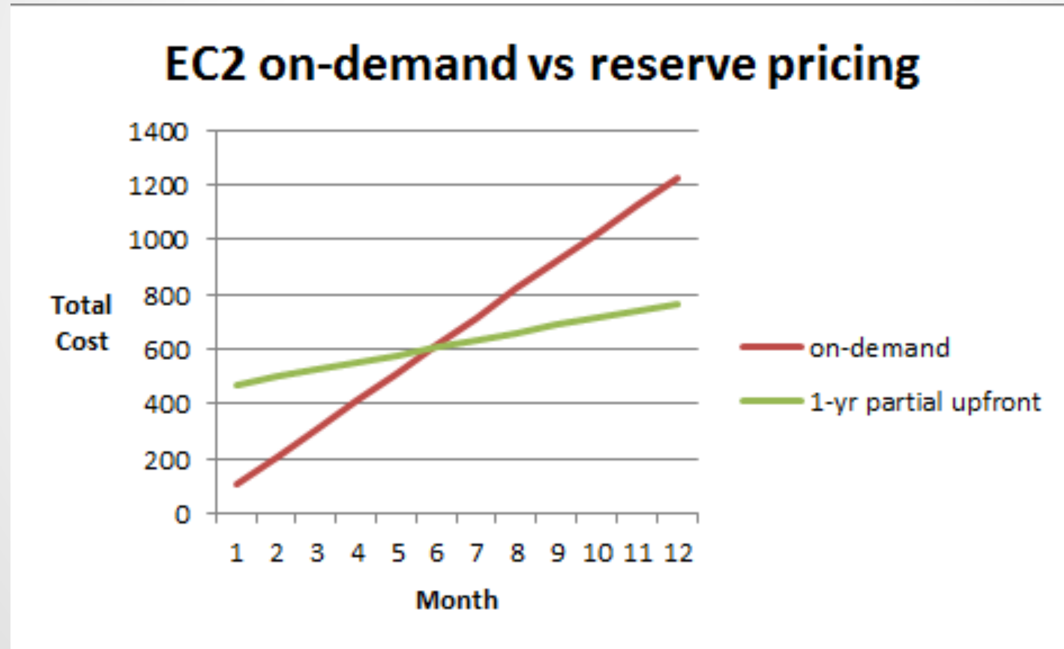
# Strategies to reduce cost (cont'd)

Reserved Instances - EC2 example - Linux m3. large

| Pricing option | Upfront | Hourly/monthly cost | Effective hourly cost |
|---|---|---|---|
| On-demand | $0 | $0.14/hr | $0.1400 |
| 1 yr no upfront | $0 | $73.00/mo | $0.1000 |
| 1 yr partial upfront | $443 | $27.01/mo | $0.0876 |
| 1 yr all upfront | $751 | $0 | $0.0857 |
| 3 yr partial upfront | $673 | $21.90/mo | $0.0556 |
| 3 yr all upfront | $1373 | $0 | $0.0522 |

# Strategies to reduce cost (cont’d)

EC2 reserved pricing breakeven example



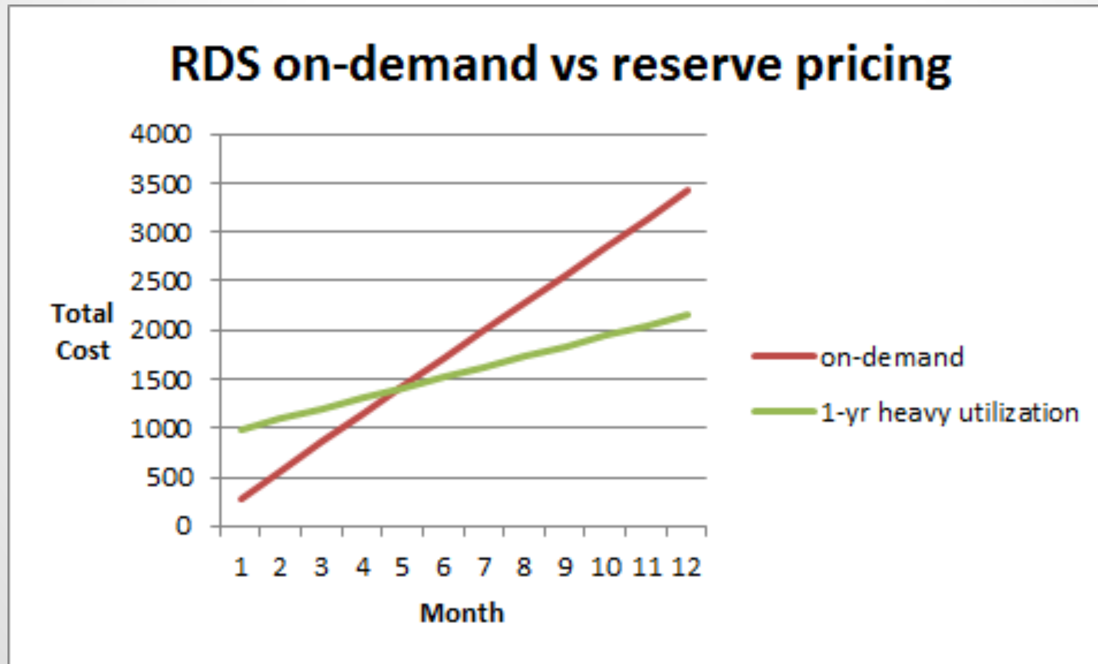
# Strategies to reduce cost (cont'd)

Reserved Instances - RDS example - PostgreSQL db.m3.large multi-AZ

| Pricing option | Upfront | Hourly cost | Effective hourly cost |
|---|---|---|---|
| On-demand | $0 | $0.390/hr | $0.390 |
| 1 yr reserved | $886 | $0.144/hr | $0.245 |
| 3 yr reserved | $1345 | $0.104/hr | $0.155 |

# Strategies to reduce cost (cont’d)

RDS reserved instance breakeven example

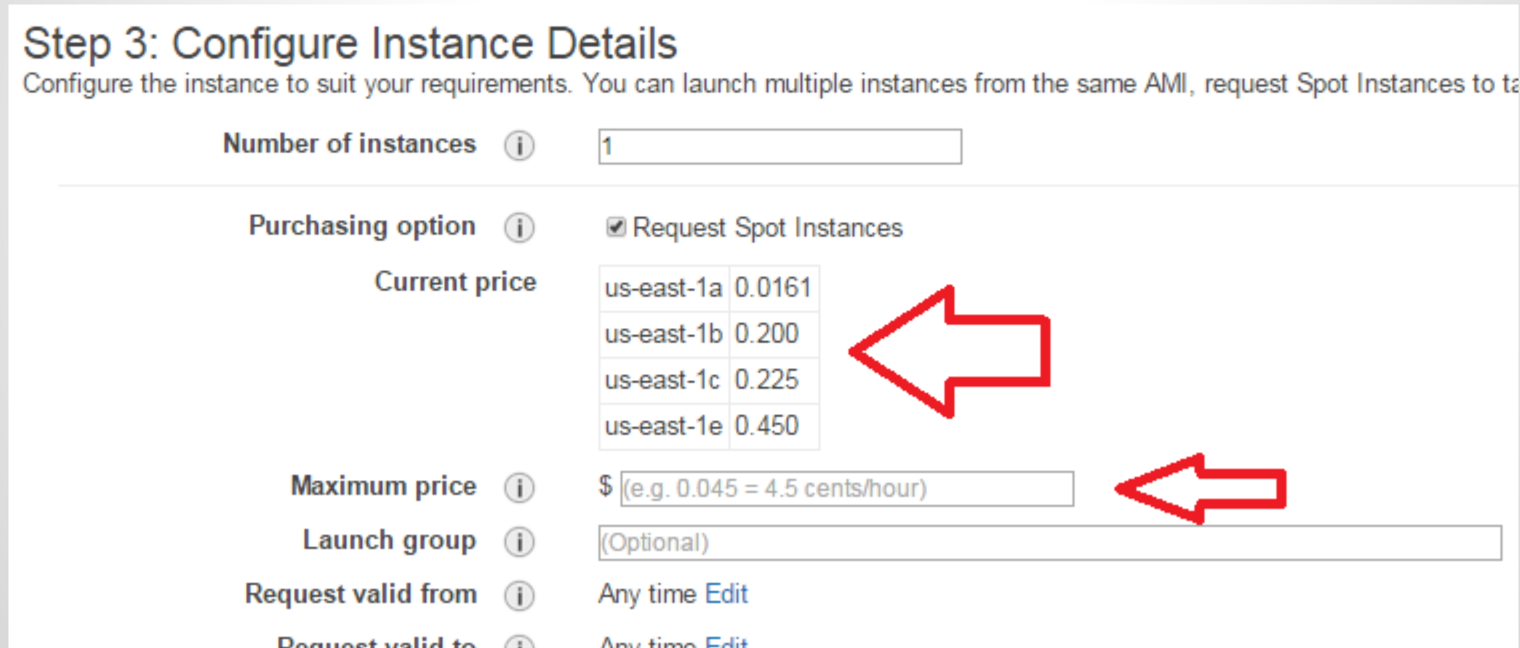


# Strategies to reduce cost (cont'd)

- Spot Instances
  - Like an auction, matching buyers and sellers of spare EC2 instances
  - Bid how much you're willing to pay per hour
  - Instances are allocated to you if available at or below your price
  - Instances vanish if there are no longer any that meet your price
  - Not good if your application needs to save state
  - Great for applications like web crawling
  - Price varies from hour to hour and from AZ to AZ
  - Recently, spot instances for Linux m3.large instance (on-demand price $0.14 / hr) were as low as $0.0161 / hr

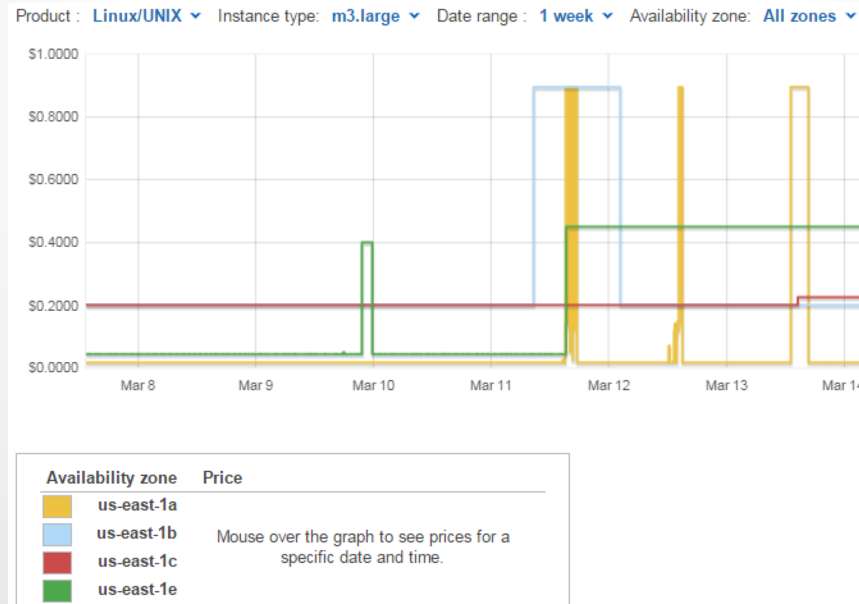
# Strategies to reduce cost (cont'd)

## Requesting spot instances



# Strategies to reduce cost (cont'd)

One-week history of Linux m3.large spot prices



# Strategies to reduce cost (cont’d)

- Auto-scaling
  - Number of instances increases or decreases automatically in response to load
  - You define what constitutes high load based on network utilization, CPU utilization, etc
  - Don’t store state on the instance
  - Requires a strategy to configure new instances
    - AMI
    - Elastic Beanstalk
    - Configuration management tool like Chef or Puppet

# Strategies to reduce cost (cont’d)

- Stop instances you’re not using
  - Not an option for load balancers or RDS instances
  - Can be manual, or automated (e.g. with a cron job)
  - Example: shut down a test environment on weekends

# Competitive pricing

Comparing 3 of the major cloud providers

- Amazon AWS
- Microsoft Azure
- Rackspace Cloud Servers

# Competitive pricing (cont’d)

All 3 vendors’ pricing models are similar

- Cloud compute charges based on
  - number of virtual CPUs
  - RAM
  - storage
- Cloud storage charges based on
  - GB stored per month
  - outbound GB transferred

# Competitive pricing (cont’d)

Be aware of differences in the details

- I/O request charges
- How redundant is the storage?
- Minimum “service level” charges
- Etc

# Competitive pricing (cont'd)

| Vendor | Compute product | Compute specs | Compute $/hr | Storage $/GB/mo | Storage $/GB xfer |
|---|---|---|---|---|---|
| AWS | Linux m3. large | 2 cores, 7.5 GB RAM, 32 GB storage | $0.14 | $0.03 | $0.09 |
| Azure | Linux D2 | 2 cores, 7 GB RAM, 100 GB storage | $0.19 | $0.03 | $0.09 |
| Rackspace | Linux Compute1-4 | 2 vCPUs, 3.75 GB RAM, no storage | $0.10 | $0.10 | $0.12 |

# Thank you!

steve@spincar.com