Amazon Holiday Traffic

This is only a 12-day outlook! The peak is likely much higher compared to March traffic.

Amazon Web Services

Amazon Web Services

EC2
Compute services

- Amazon maintains many thousands of servers. Each server hosts many virtual machines
- You can sign up for EC2 and rent virtual machines with a certain number of CPU cores and a certain amount of memory
Amazon Web Services

EC2
Compute services

EBS
Block storage
(like a local filesystem, but accessed over a network)

- Amazon maintains large network of storage arrays
- Disk arrays are networked so that even if one array fails, the system will stay up
- You can mount any EBS volume from any EC2 instance in the same datacenter
- The EBS volume appears as if it’s a normal hard drive. An EBS volume can only be mounted to one EC2 instance at a time
Amazon Web Services

- **EC2**: Compute services
- **EBS**: Block storage (like a local filesystem, but accessed over a network)
- **S3**: Object storage (sort of like Google Drive)
Amazon Web Services

EC2
Compute services

EBS
Block storage
(like a local filesystem, but accessed over a network)

S3
Object storage
(sort of like Google Drive)

Glacier
Archive storage
(like S3, but cheap and glacially slow)
# Amazon Web Services

## Amazon Web Services

<table>
<thead>
<tr>
<th>Compute</th>
<th>Developer Tools</th>
<th>Internet of Things</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2 Virtual Servers in the Cloud</td>
<td>CodeCommit Store Code in Private Git Repositories</td>
<td>AWS IoT Beta Connect Devices to the cloud</td>
</tr>
<tr>
<td>EC2 Container Service Run and Manage Docker Containers</td>
<td>CodeDeploy Automate Code Deployments</td>
<td>Mobile Services</td>
</tr>
<tr>
<td>Elastic Beanstalk Run and Manage Web Apps</td>
<td>CodePipeline Release Software using Continuous Delivery</td>
<td>Mobile Hub Beta Build, Test, and Monitor Mobile apps</td>
</tr>
<tr>
<td>Lambda Run Code in Response to Events</td>
<td></td>
<td>Cognito User Identity and App Data Synchronization</td>
</tr>
</tbody>
</table>

## Storage & Content Delivery

| S3 Scalable Storage in the Cloud | CloudWatch Monitor Resources and Applications | Internet of Things |
| CloudFront Global Content Delivery Network | CloudFormation Create and Manage Resources with Templates | AWS IoT Beta Connect Devices to the cloud |
| Elastic File System PREVIEW Fully Managed File System for EC2 | CloudTrail Track User Activity and API Usage | Mobile Services |
| Glacier Archive Storage in the Cloud | Config Track Resource Inventory and Changes | Mobile Hub Beta Build, Test, and Monitor Mobile apps |
| Import/Export Snowball Large Scale Data Transport | OpsWorks Automate Operations with Chef | Cognito User Identity and App Data Synchronization |
| Storage Gateway Integrates On-Premises IT Environments with Cloud Storage | Service Catalog Create and Use Standardized Products | Device Farm Test Android, Fire OS, and iOS apps on real devices in the Cloud |

## Database

| RDS Managed Relational Database Service | Identity & Access Management Manage User Access and Encryption Keys | Elastic Transcoder Easy-to-use Scalable Media Transcoding |
| DynamoDB Predictable and Scalable NoSQL Data Store | Directory Service Host and Manage Active Directory | SES Email Sending Service |
| ElastiCache In-Memory Cache | Inspector Analyze Application Security | SQS Message Queue Service |
| Redshift Managed Petabyte-Scale Data Warehouse Service | WAF Filter Malicious Web Traffic | SWF Workflow Service for Coordinating Application Components |

## Networking

| VPC Isolated Cloud Resources | Analytics | Enterprise Applications |
| Direct Connect Dedicated Network Connection to AWS | EMR Managed Hadoop Framework | WorkSpaces Desktops in the Cloud |
| Route 53 Scalable DNS and Domain Name Registration | Data Pipeline Orchestration for Data-Driven Workflows | WorkDocs Secure Enterprise Storage and Sharing Service |

## Security & Identity

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identity</td>
<td>Directory Service Host and Manage Active Directory</td>
<td>SES Email Sending Service</td>
</tr>
<tr>
<td>Access Management</td>
<td>Inspector Analyze Application Security</td>
<td>SQS Message Queue Service</td>
</tr>
<tr>
<td>Manage User Access and Encryption Keys</td>
<td>WAF Filter Malicious Web Traffic</td>
<td>SWF Workflow Service for Coordinating Application Components</td>
</tr>
</tbody>
</table>

## Analytics

<table>
<thead>
<tr>
<th>Analytics</th>
<th>Enterprise Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMR Managed Hadoop Framework</td>
<td>WorkSpaces Desktops in the Cloud</td>
</tr>
<tr>
<td>Data Pipeline Orchestration for Data-Driven Workflows</td>
<td>WorkDocs Secure Enterprise Storage and Sharing Service</td>
</tr>
</tbody>
</table>

## Amazon Web Services

https://codentrick.com/aws-amazon-web-services-overview/
Amazon Web Services

- Estimated 1.3 million servers\(^1\) in 68 datacenters\(^2\)
- Custom routers. 100 Gbps interconnects between data centers, 25Gbps connections to each server
- Custom server design, custom motherboard chipsets, custom GPUs and FPGAs
- Custom storage servers. Each rack contains 1110 hard drives, 8.8 petabytes of storage

Benefits of “cloud computing”

- **Benefits to AWS users:**
  - No huge up-front infrastructure investment
  - No need to hire dedicated systems administrators
  - Stability benefits of globally distributed infrastructure
  - Flexibility in handling load… Pay only for what you need and avoid getting slammed in a high-load event

- **Benefits to Amazon:**
  - Rent out unused storage capacity, make lots of money
  - Infrastructure investments benefit Amazon as well
  - $$$$$$$
<table>
<thead>
<tr>
<th>Segment Information</th>
<th>Three Months Ended December 31,</th>
<th>Twelve Months Ended December 31,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2016</td>
<td>2017</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>North America</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$26,240</td>
<td>$37,302</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>25,424</td>
<td>35,610</td>
</tr>
<tr>
<td>Operating income</td>
<td>$ 816</td>
<td>$1,692</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>International</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$13,965</td>
<td>$18,038</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>14,452</td>
<td>18,957</td>
</tr>
<tr>
<td>Operating income (loss)</td>
<td>$(487)</td>
<td>$(919)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$ 3,536</td>
<td>$ 5,113</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>2,610</td>
<td>3,759</td>
</tr>
<tr>
<td>Operating income</td>
<td>$  926</td>
<td>$1,354</td>
</tr>
</tbody>
</table>

### AMazon.com, Inc.
#### Segment Information
#### (in millions)

<table>
<thead>
<tr>
<th></th>
<th>Three Months Ended December 31,</th>
<th>Twelve Months Ended December 31,</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2016</td>
<td>2017</td>
</tr>
<tr>
<td><strong>North America</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$26,240</td>
<td>$37,302</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>25,424</td>
<td>35,610</td>
</tr>
<tr>
<td>Operating income</td>
<td>$ 816</td>
<td>$1,692</td>
</tr>
<tr>
<td><strong>International</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$13,965</td>
<td>$18,038</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>14,452</td>
<td>18,957</td>
</tr>
<tr>
<td>Operating income (loss)</td>
<td>$(487)</td>
<td>$(919)</td>
</tr>
<tr>
<td><strong>AWS</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Net sales</td>
<td>$ 3,536</td>
<td>$ 5,113</td>
</tr>
<tr>
<td>Operating expenses</td>
<td>2,610</td>
<td>3,759</td>
</tr>
<tr>
<td>Operating income</td>
<td>$ 926</td>
<td>$1,354</td>
</tr>
</tbody>
</table>

Users of AWS

If we were to rethink filesystems built for cloud services, what would they look like?
Cloud-Native File Systems

Remzi H. Arpaci-Dusseau
Andrea C. Arpaci-Dusseau
University of Wisconsin-Madison

Venkat Venkataramani
Rockset, Inc.
How And What We Build Is Always Changing

Earliest days

- Assembly programming on single machines

Big single-machine advances

- Unix: A standard (and good) OS!
- C: A systems language!

Same thing, one level up: Distributed systems

- Collect group of standard machines, build something interesting on top of them
Commonality: New System on Fixed Substrate

Whether a single machine/distributed, we tend to build **new systems** on a **fixed set of resources** with **fixed (sunk) cost**

- Machine: X CPUs, Y GB memory, Z TB storage
- Buy many such machines
- Build new system of interest on those machines

But the world is changing...
Welcome To Cloud

Cloud is a reality

• Can rent cycles or bytes as needed
• Per-unit cost is defined and known
• Not just raw resources: services too

Many new systems are being realized only in cloud

• Excellent example: Snowflake elastic warehouse [sigmod ’16]
Thus, Questions

Cloud-native thinking:
How should we build systems given the cloud?

• What new opportunities are available?
• What new systems can we realize?
• What can we stop worrying about?
In This Talk

Cloud-native principles
• Guidelines for how to think about building systems in the era of the cloud

Cloud-native file system
• Case study: How to transform a local file system into a cloud-native one
Principles

**Storage** principles

**CPU** principles

**Overarching** principle

(just highlights; more in paper)
Storage Reliability

Storage reliability principle:
Highly replicated, reliable, and available storage can (should?) be used (The “S3” principle)

- 11 “9s” of durability!

Implication: Build on top of this, don’t build YARSS (Yet Another Replicated Storage System)

- Example (kind of): BigTable on GFS
Storage Cost and Capacity

**Storage cost principle:**
Storage space is generally inexpensive
- At cheapest, $4 / month / TB

**Storage capacity principle:**
A lot of storage space available
- “The total volume of data and number of objects you can store are unlimited” (Amazon)

**Implication:** Use space as needed to improve system
- Example: Indices for added lookup performance
Storage Hierarchy

**Storage hierarchy principle**: Storage is available in many forms, with noticeable differences in performance and cost across each level

- Example: Amazon Glacier vs S3

**Implication**: Must manage data across levels

- Can improve performance, reduce costs
CPU Parallelism

**CPU parallelism principle** (or $A \times B = B \times A$): It should cost roughly the same to execute on $A$ CPUs for $B$ seconds as it does to execute on $B$ CPUs for $A$ seconds

- Granularity of accounting might limit you…

**Implication**: Do everything you can in parallel
CPU Capacity

**CPU capacity principle:**
Large numbers of CPUs are available
- As with storage, essentially "unlimited"

**Implication:** Use as many CPUs as you need
- Scale up to solve tasks quickly
CPU Scale-Up/Down

CPU scale-up/scale-down principle:
One should only use as many CPUs as needed for a task, and not more

• While cheap, CPUs are not free either

Implication: Must monitor usage, turn off CPUs when unused
CPU Remote Work

CPU remote-work principle:
When possible, use remote CPU resources to do needed work

  • Shared data store makes this easier

Implication: Can separate foreground/background

  • Improve predictability of former, use parallelism for latter
CPU Hierarchy

**CPU hierarchy principle**: CPU is available in different forms, with differences in performance, cost, and reliability across each level

- Normal vs. spot instance for example

**Implication**: CPU types must be managed

- Pick CPU right for given task
Overarching Principle

**Overall performance/cost principle:** Every decision in cloud-native systems is ultimately driven by a cost/performance trade-off

- Can’t make decisions without cost/perf knowledge
- Extremes are interesting: highest performance, or lowest cost
- But middle ground is important too: “reasonable” cost/performance

**Implication:** Cost must be fundamental part of systems (and even applications above)
Implications

Replicated storage: Don’t reinvent the wheel
Extra space is cheap: Use for performance?
Massive parallelism: Use for background tasks
Hierarchy: Continuous data migration to lower cost while keeping performance high?
Cost: Have to know how much is OK to spend

Overall: Proper utilization of the cloud requires rethinking of how we build the systems above them
Case Study: CNFS
Case Study: CNFS

Case Study: **Cloud-Native File System (CNFS)**

**Classic**
- File System

**Cloud-Native**
- CNFS
  - Cloud Block Service (e.g., EBS)
CNFS Architecture

- **App**
- **VM**
- **CNFS**
- **Snap**
- **Snap**
- **Snap**
- **Snap**
- **Snap**
- **Snap**

- **Amazon EBS High-Performance**
- **Amazon EBS Low-Cost**

**Functions:**
- **Communicate**
- **Demote**
- **Compress**

**Actions:**
- **Read/Write**

**CNFS Manager**

**Workers**
CNFS: Key Points

Copy-on-write (COW): Natural fit for cloud
  • Enables background work on immutable storage

Storage work naturally offloaded from front end
  • Enables predictable low-latency for foreground
  • Adds massive parallelism for background

Can optimize for cost or performance or mix
  • Need hints from above on what is important
  • New APIs too

But, still needs help from cloud providers
  • Example: Can’t access EBS volumes from many clients (now)
Conclusions

Cloud Native

• New way to build systems upon substrate provided by Cloud

Principles: New guidelines for design

• Higher-level services: Don’t reinvent the wheel
• Flexible resources: Can use a lot or a little
• Different types of resources: Costly/Fast vs. Cheap/Slow
• Cost awareness: Nothing is free

Case study: CNFS

• A local COW file system built to run on EBS (not a disk)
• Early prototype: Modified ext4 can migrate files across cloud volumes (but much still to be done)

Cloud-native thinking: How does it change your next system?
End