

# CloudCAP: Case Study in Capacity Planning Using the Cloud

Joan A. Smith, PhD  
Emory University  
Atlanta, GA  
[mail@joanasmith.com](mailto:mail@joanasmith.com)



John F. Owen, MS-CS  
Owenworks, Inc.  
Blacksburg, VA  
[frank@owenworks.biz](mailto:frank@owenworks.biz)



Owenworks, Inc.

James R. Gray, MS  
Kronos, Inc.  
Washington, DC  
[jim.gray@kronos.com](mailto:jim.gray@kronos.com)



# Background

- Commercial foundation:
  - Responsible for performance assurance of web-based employee time-tracker
  - Mission-critical US government software
  - Annual leave, hours worked, holidays, overtime, etc.
  - Many versions for special-case installations
- 1 Million+ Users
  - Majority of US government departments, bureaus
  - Agency-specific customizations
- Custom, local installations
  - Customer Data Center deployment (not Cloud – security regs)
  - Dedicated hardware purchased for each installation
  - High system performance requirements
  - Top-Quality software assurance essential

# Problem Statement:

## Performance Tuning & Capacity Planning

- Ensure scalability & estimate deployment requirements given:
  - Frequent product releases
  - Multiple optional feature sets
  - Various deployment topologies
  - Installed-user base from 300 to 300,000+ users at any single site
- Detect & evaluate performance regressions
  - Determine resource constraints
  - Disk I/O, CPU usage, RAM usage, Network I/O
  - Correct & retest prior to release

# Solution: CloudCAP

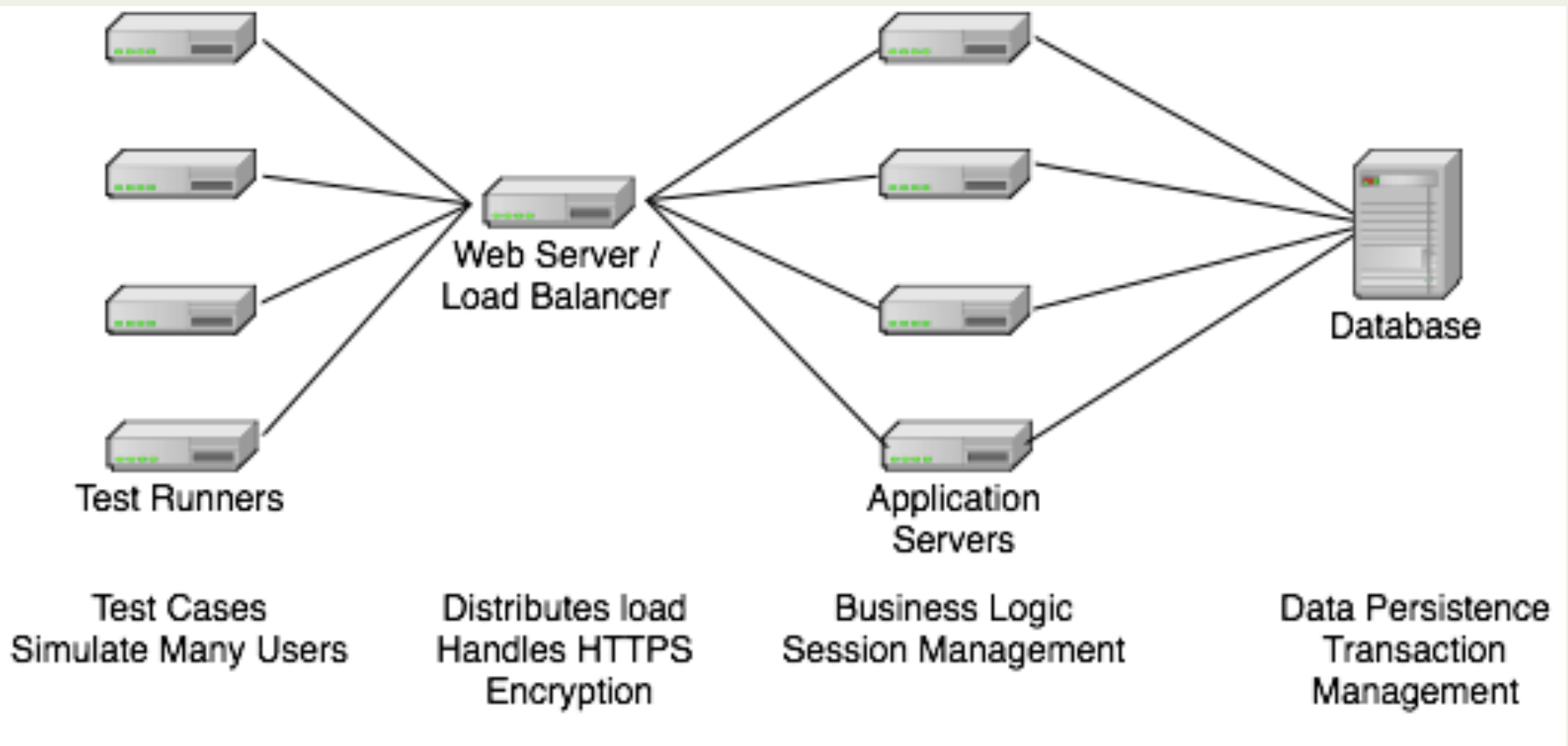
## *Two key innovations:*

1. Web interface for cluster configuration & life-cycle management
  - Consolidates AWS hardware options
  - Streamlines host software installation, configuration (Oracle, Apache, Tomcat, JMeter, etc.)
  - Connects to version control repository for Product Release selection, auto-installation, & configuration
2. Auto-discovery & inter-nodal communication
  - Sequencing of node state transitions
  - Peer discovery: What is the topology of the cluster
  - Synchronization: App servers cannot start until database is servicing requests

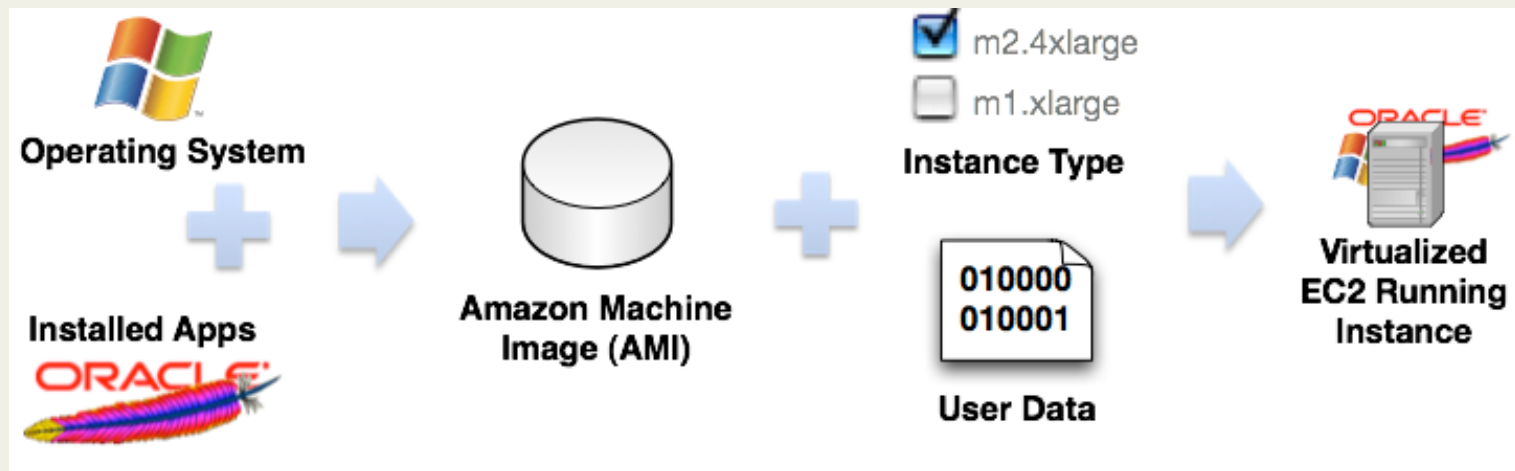
# EC2: What it is & How it works

- EC2 = Elastic Compute Cloud
  - A virtual infrastructure
  - Highly configurable with OS, RAM, CPU features
- Instance Creation
  - Amazon Machine Image: Hard Drive image of an OS stored in Amazon Cloud (S3 or EBS)
  - Instance Type: Selectable Hardware
  - Running copy of an image on an instantiation of an instance type
  - Max 16K user data
- Costing
  - Cost begins upon powering up
  - Instance Price per Hour: \$0.02-\$2.48
  - Additional \$\$ for network, storage usage
- Management
  - Web interface for administration
  - Query API (URL based query strings)
  - SOAP API (Web Service)

# Typical large app cluster



# Launching a Virtual Instance



# Cluster Management: Web UI

**CloudCAP**

[Instructions](#) [AMI Types Reference](#) [Update S3 Cache](#)

**Software Releases**

- emory-voyages-20101115\_r39547.zip
- emory-voyages-20101112\_r39528.zip
- emory-voyages-20101022\_r39120.zip
- emory-voyages-20100922\_r38602.zip
- emory-voyages-20101129\_r39796.zip
- emory-voyages-20101124\_r39769.zip
- emory-voyages-20101124\_r39767.zip
- emory-voyages-20101118\_r39662.zip

**Database Exports**

- none
- voyages-v1.exp.dmp.gz
- voyages-v2.exp.dmp.gz

**AWS DataCenter**

- us-east-1a
- us-east-1b
- us-east-1c
- us-east-1d

**Database**

MySQL Linux32  
Oracle Linux32  
Oracle 10g Linux64  
Oracle 11g Linux64  
Oracle Win64

**App Server:**

Tomcat 5 Linux32 4  
Tomcat 5 Linux64

**Load Balancer:**

**Test Runner:**

**Instance Specifications**

<input checked="" type="radio"/> m1.large	<input type="radio"/> 1 GB
<input type="radio"/> m1.xlarge	<input type="radio"/> 2 GB
<input type="radio"/> c1.xlarge	<input type="radio"/> 4 GB
<input type="radio"/> m2.2xlarge	<input checked="" type="radio"/> 6 GB
<input type="radio"/> m2.4xlarge	<input type="radio"/> 8 GB

**Use Tomcat Native Library**

DB Conn: active 1000 idle 100 wait -1

JVM Args:



# JMeter Report Example

**Summary Report**

Name: Summary. 76K DB m2.4xlg DB 32G; Four c1.xlg App 6G, m1.xlg Load, Ten m1.xlg JM 8GB 250 EMP/250 SUP

Comments: default JVM ARGS

Write results to file / Read from file

Filename   Log/Display Only:  Errors  Successes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
TiTo Employees:WebTA Start	4000	1745	59	10451	1258.98	0.00%	9.5/sec	3184.84	345089.0
TiTo Employees:Employee Login	4000	775	52	7693	824.66	0.00%	9.5/sec	91.72	9922.0
TiTo Employees:Click Timesheets	4000	619	49	6690	472.41	0.02%	9.5/sec	411.41	44471.3
TiTo Employees:Change Pay Period	3999	476	53	6377	351.03	0.00%	9.5/sec	422.15	45532.4
TiTo Employees:Select Day of Week	39990	643	51	10633	622.31	0.00%	93.2/sec	4533.99	49796.8
TiTo Employees:Enter Time, Click Save	39990	842	61	12883	784.11	0.00%	93.3/sec	5188.39	56939.2
TiTo Employees:Enter Time and Click Validate	3999	1918	95	14375	1821.15	0.00%	9.6/sec	351.91	37607.0
TiTo Employees:Click Affirm	3999	953	81	11618	893.43	0.00%	9.7/sec	562.76	59617.2
TiTo Employees:Click Logout	3999	181	16	6227	226.12	0.00%	9.7/sec	83.07	8782.0
TOTAL	107976	797	16	14375	855.29	0.00%	248.8/sec	14565.81	59942.1

# Amazon Management Console Example

The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo, navigation links (Products, Developers, Community, Support, Account), and user options (Settings, Sign Out). The main content area is divided into a left-hand navigation pane and a central workspace.

**Navigation Pane:**

- Region: US East
- EC2 Dashboard
  - Instances
  - Spot Requests
- IMAGES
  - AMIs
  - Bundle Tasks
- ELASTIC BLOCK STORE
  - Volumes
  - Snapshots
- NETWORKING & SECURITY
  - Elastic IPs
  - Security Groups
  - Placement Groups
  - Load Balancers
  - Key Pairs

**My Instances:**

Viewing: All Instances | All Instance Types

Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name
<input type="checkbox"/>	i-b6efdbdb	ami-447b8f2d	ebs	m1.large	running	zgroup, database	jim.gray.a
<input type="checkbox"/>	i-b0efdbdd	ami-447b8f2d	ebs	m1.large	running	zgroup, app server	jim.gray.a
<input checked="" type="checkbox"/>	i-b2efdbdf	ami-447b8f2d	ebs	m1.large	running	zgroup, load balanc	jim.gray.a
<input checked="" type="checkbox"/>	i-8cefdbe1	ami-447b8f2d	ebs	m1.large	running	zgroup, jmeter mas	jim.gray.a
<input checked="" type="checkbox"/>	i-24e8dc49	ami-447b8f2d	ebs	m1.large	running	zgroup, database	jim.gray.a
<input type="checkbox"/>	i-26e8dc4b	ami-447b8f2d	ebs	m1.large	running	zgroup, app server	jim.gray.a
<input type="checkbox"/>	i-22e8dc4f	ami-447b8f2d	ebs	m1.large	running	zgroup, load balanc	jim.gray.a

**EC2 Instances: i-24e8dc49, i-b2efdbdf, i-8cefdbe1**

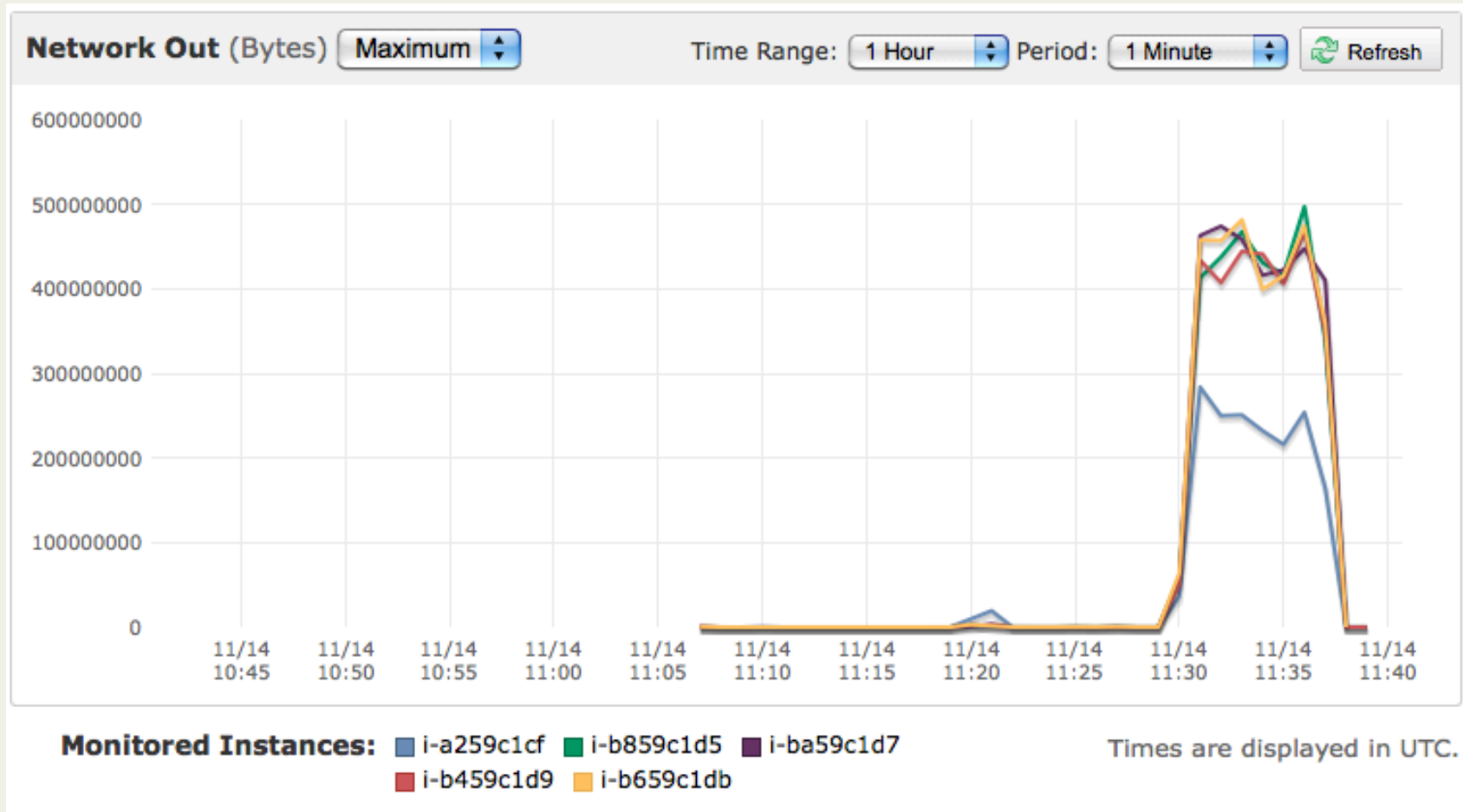
Monitoring | Description | Tags

Time Range: Last Hour | Refresh

Graphs are for 3 instances with detailed monitoring enabled. Times are displayed in UTC.

- Avg CPU Utilization (Percent):** Line graph showing a spike to approximately 50% at 01:30 on 12/12.
- Avg Disk Reads (Bytes):** Line graph showing a spike to approximately 250,000 bytes at 01:30 on 12/12.
- Avg Disk Writes (Bytes):** Line graph showing a spike to approximately 2,500,000,000 bytes at 01:30 on 12/12.
- Max Network In (Bytes):** Line graph showing a spike to approximately 80,000,000 bytes at 01:30 on 12/12.
- Max Network Out (Bytes):** Line graph showing a spike to approximately 250,000 bytes at 01:30 on 12/12.

# Amazon CloudWatch: Example Graph Detail



# Challenges: Results Analysis

- Instance types are ***approximate***
  - Need multiple runs to even out results
  - *Cluster Luck*:  
The likelihood Amazon EC2 gives you a fast set of machines.
- Mapping test results to real hardware is difficult
  - How much CPU is a CU worth?
  - Close approximations are possible
- EC2 network topology differs from deployment topology
- Disk I/O performance is poor on EC2
  - Disk bound apps not a good match for EC2 testing
  - But: Few apps these days are disk bound
- Lack of comparable historical data from similar sites

# Problem Statement: Academic Library

- Peak user loads at launch, tapers off rapidly
  - Typical scenario: 100+ users/month
  - Rare scenario: 10,000+ users/month
  - No good predictors for launch-time load
  - Few academic sites hit the user jack-pot
- Infrequent releases
  - Unique digital scholarship projects
  - Relatively static content
  - Usually one-off: Not baseline for other projects
- Performance regressions are unlikely to be caught before deployment
  - Little QA performed
  - Deadline/research driven, not QA/customer driven

# Adapting CloudCAP to Library Web

- Some Reusable Parts
  - *Reuse*: Web UI/node lifecycle management
  - *Reuse*: Startup state management
  - *Reuse*: Peer discovery, registration, interaction
- Several New Parts
  - *New*: Application-specific configuration
  - *New*: Application-specific load scripts/test scripts
  - *New*: Application-specific performance settings
  - *New*: Application-specific tuning parameters

# Cost-Benefit Comparison

- Commercial App:
  - + High reusability
  - + Risk-justified (high cost of software failure)
  - + Regression-testing produces critical information
  - + Cost amortized through reduced deployment expenses, increased sales, customer satisfaction
- Academic Scholarly App:
  - Limited reusability
  - Risk-unjustified (low cost of software failure)
  - Regression-testing uninformative or not critical
  - Cost not recoverable in sales, license fees
- BUT:
  - + Potentially useful for university-wide IT App deployments where risk of failure is high

# QUESTIONS & ANSWERS

Thank you for attending our presentation.

Joan A. Smith, PhD  
Emory University  
Math & CS Dept.  
Atlanta, GA 30322  
[mail@joanasmith.com](mailto:mail@joanasmith.com)

John F. Owen, MS-CS  
Owenworks, Inc.  
Blacksburg, VA  
[frank@owenworks.biz](mailto:frank@owenworks.biz)