WHITE PAPER | JANUARY 2017

APM Strategies to Increase Cloud Application Resilience

Design for failure with AWS cloud architecture and CA Application Performance Management.

Srikant Noorani
Senior Engineering Architect

Peter Waterhouse
DevOps Solutions Marketing

Table of Contents

Executive Summary

Challenge

Modern cloud applications are inherently complex. So much so in fact, that organizations cannot hope to prevent failures—they're inevitable. But because these systems present an opportunity to conduct digital business at scale, any form of application fragility can and will have dire consequences. Organizations can easily lose sight of this: migrating or developing applications, that while being scalable, lack the resilience and self-recovery needed to remain available and performant—even in adverse situations.

Opportunity

By following explicit guidance from Amazon and utilizing Amazon Web Services (AWS) tools, organizations can better implement one critical design principle of the cloud: *design with failure in mind*. Augment this with application performance management (APM) solutions, and teams gain actionable insights into where and how design-for-failure architectures can be improved—essentially identifying weaknesses in the model to recover from failure and then improving it.

Benefits

The integration between CA Application Performance Management (CA APM) and AWS enables DevOps practitioners to optimize cloud architecture. With advanced visualization surfacing key performance metrics in context of AWS services, and by identifying anomalies, teams can:

- Diagnose problems with AWS self-recovery services.
- Determine the impact of AWS changes on customer experience.
- Increase cost efficiency by diagnosing conditions that cause AWS instances to be overloaded or under-utilized.
- Help identify the cloud designs best suited for different business services.

Section 1:

Cloud Applications: Design for Failure or Prepare to Fail

With the rise of cloud computing, design for failure has become a dominant architectural model. Its strength (although some might argue it's a weakness) is how it places greater control over application performance and availability with development rather than with IT operations staff, on-premises infrastructure or managed service providers.

The rationale behind the thinking is that management responsibility shifts to the application, irrespective of the underlying cloud fabric. With design for failure, applications should be able to achieve high levels of performance and availability, regardless of how elements of the cloud infrastructure are behaving. Even if a virtual instance has an issue or a cloud data center suffers a major meltdown, applications should continue to provide uninterrupted service.

Fundamentally, and as the name implies, design for failure works on the assumption that cloud applications can and will always have problems. Working on this premise, organizations enact strategies during coding, testing and monitoring to not only detect inevitable problems but to learn from them to continuously improve the application. Unfortunately, many organizations have failed to fully embrace this concept, assuming instead that cloud scale will always deliver optimum application performance. Recent events have taught many businesses that this isn't the case.

Take for example the June 2016 floods, which severely impacted cloud services in Sydney, Australia. Torrential rains led to outages across Amazon availability zones, resulting in many websites having degraded service or failing completely.[1, 2] Some businesses, however, fared better than others. By employing multiregion failover, their own native APIs and other cloud vendor services, they could "weather the storm." Essentially, these businesses had embraced design for failure in their application architectures and provided uninterrupted and high-quality services—regardless of cloud infrastructure problems.

With DevOps and cloud, organizations accept that failure is to be expected and leverage many tools to support design-for-failure architectures. AWS provides many supporting services as illustrated in table 1 below:

Table 1:

Sample Amazon AWS services to support design-for-failure architectures.

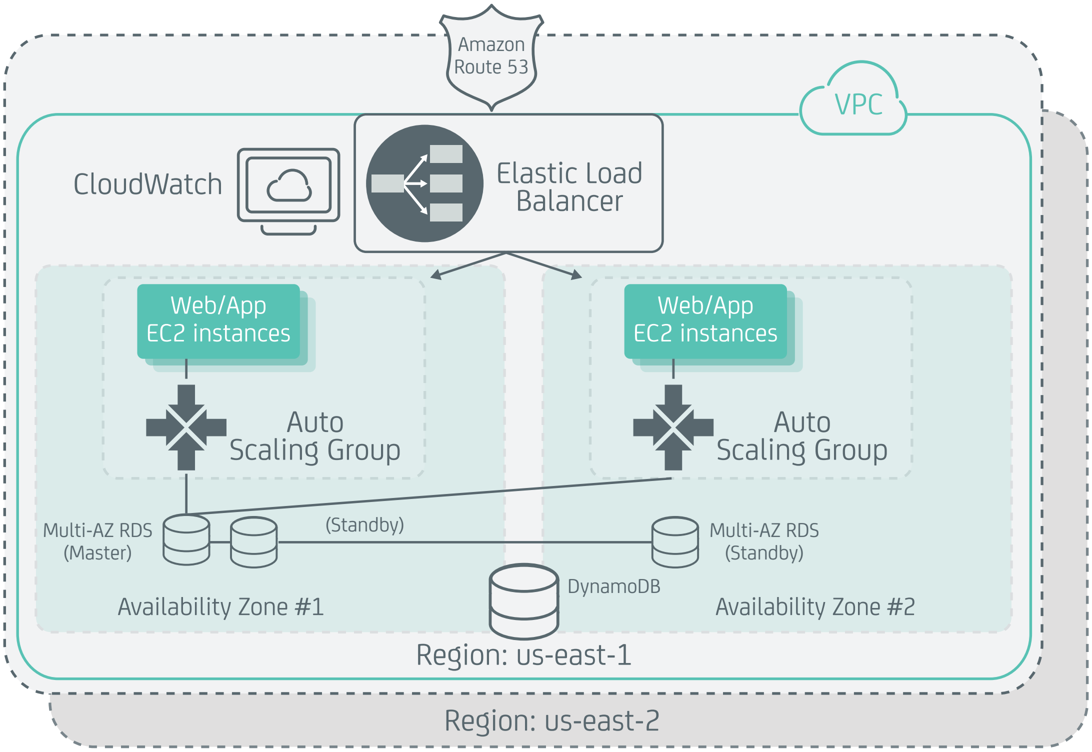
Auto Scaling	Helps business maintain application availability and scales capacity according to predefined conditions.
Elastic Load Balancing	Distributes incoming application across multiple instances, enabling application fault tolerance.
Amazon relational database service (RDS) for multi-availability zones	Increases availability by synchronously replicating data across availability zones.
DynamoDB	A fully managed NoSQL database service providing fast and predictable performance and scalability.
Simple Queuing Service (SQS)	A fast, reliable, scalable, fully managed message queuing service.
Simple Notification Service (SNS)	A managed push notification service to send individual messages or to fan-out messages to large numbers of recipients.
Elastic Block Storage	Highly available storage volumes attached to any running instance in the same availability zone.
CloudWatch	Monitoring services for AWS services and applications.

Amazon provides comprehensive documentation outlining best practices for building design-for-failure cloud architecture.[3,4]

By constructing application architectures comprising many of these elements and eliminating single points of failure, cloud applications should continue to operate—even when problems occur with the cloud infrastructure. Figure 1 illustrates an example of an AWS design-for-failure architecture that incorporates many self-recovery and resilient services.

Figure 1:

Example AWS architecture



As illustrated above, many AWS elements are being leveraged to support multichannel applications and architected to cater for unexpected failure conditions. Instances have been placed across multiple availability zones, while Multi-AZ RDS provides high-availability and failover for databases. To support unanticipated demand, AWS Auto Scaling is used to provision additional EC2 capacity when needed or replace unhealthy EC2 instances. Additionally, DynamoDB is providing automatic replication across availability zones. Although not presented, other services could form part of the design-for-failure architecture, such as Amazon ElastiCache to increase web and mobile application performance, and Elastic block storage for persistent file systems.

Section 2:

Optimizing Cloud Architecture With Modern APM

With design-for-failure architecture, APM takes on additional importance. Since cloud applications should stay performant irrespective of partial failures (e.g., in an availability zone partition), the role of APM is not only to detect these events but to actively inform all stakeholders (development, IT operations, etc.) in order to avert more serious problems, potentially halt application deployments and capture feedback to drive future design improvements. As such, APM's role is completely synergistic with a design-for-failure architecture—that is, it becomes a mechanism to detect failures in an organization's model to plan and recover from failure (which may comprise many of the critical AWS elements presented above). It's also an important tool to continuously test the efficacy of design-for-failure models and prevent teams from making false assumptions or becoming complacent about cloud application resilience.

There are many use cases where APM can be used to enhance design-for-failure architectures. Consider a fairly typical example of a business leveraging AWS cloud services to provide the scale needed to support a new multichannel service (web and mobile). Supplemented by an online marketing and social media campaign, new application functionality may have been released to drive increased customer engagement during a major business event (e.g., the Black Friday shopping season).

During events such as these, the traditional approach of equating reliability with preventing failure falls short. Cloud-based services are inherently complex with many more interdependencies, and unknown demand means that millions of customers could be accessing an application. This results in more pressure on the application architecture and more opportunity for failure. To this end therefore, APM should support the goal of containing inevitable failures and recovering quickly so that business continues uninterrupted. More complete solutions will support the goal of continuous improvement, with monitoring providing teams with valuable feedback into the optimum cloud infrastructure needed for different sets of services operating in a variety of conditions.

To support these goals, modern APM solutions must first be capable of providing cross-functional teams accurate visual representations of the application stack and cloud architecture. Rather than forcing teams to conduct manual topology configuration and element mapping, comprehensive APM will automatically detect and discover cloud applications and create a visual model that is as close to the actual AWS cloud deployment as possible (see Figure 2).

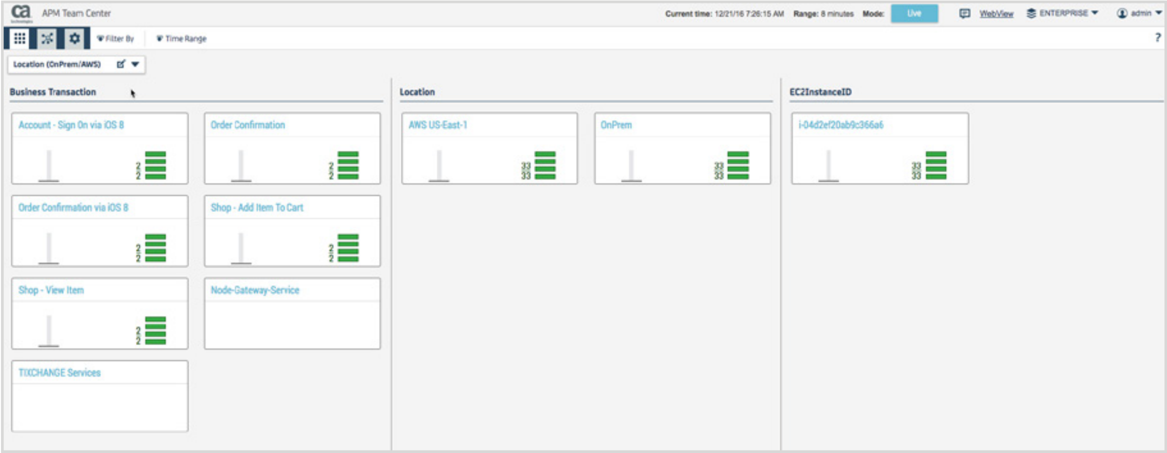


Figure 2:

CA APM Team Center and perspectives—location-based visualization of business services across on-premises and AWS infrastructure.

Here, the perspectives feature of CA APM Team Center is used to simplify and visualize complex cloud topologies into task-relevant views. Using the drop-down box located in the top left, the AWS application deployment has been organized by location. Rather than having to understand the intricacies of APM and mapping topology elements to the actual deployment, the solution has automatically built an accurate representation of the model. This enables teams to have a shared understanding of the architecture and becomes a common reference point to monitor application performance, detect and remediate failures (e.g., in AWS self-recovery services) and identify opportunities to increase quality and resilience.

As also presented in Figure 2, the deployment has been broken down into major microservice elements, including authentication, shopping cart and order confirmation. Since these services are supported by both on-premises infrastructure and AWS cloud services, these are displayed together with general health indicators. Notice how this location-based view has surfaced details of the AWS components, including the region where the application is deployed (us-east-1) and one active instance in operation.

When additional AWS services are employed, they'll be automatically displayed on the CA APM team center dashboard. In Figure 3 for example, an AWS-specific view is enabling support teams to see at a glance a number of elements, including VPC, subnets and availability zones (x 3).

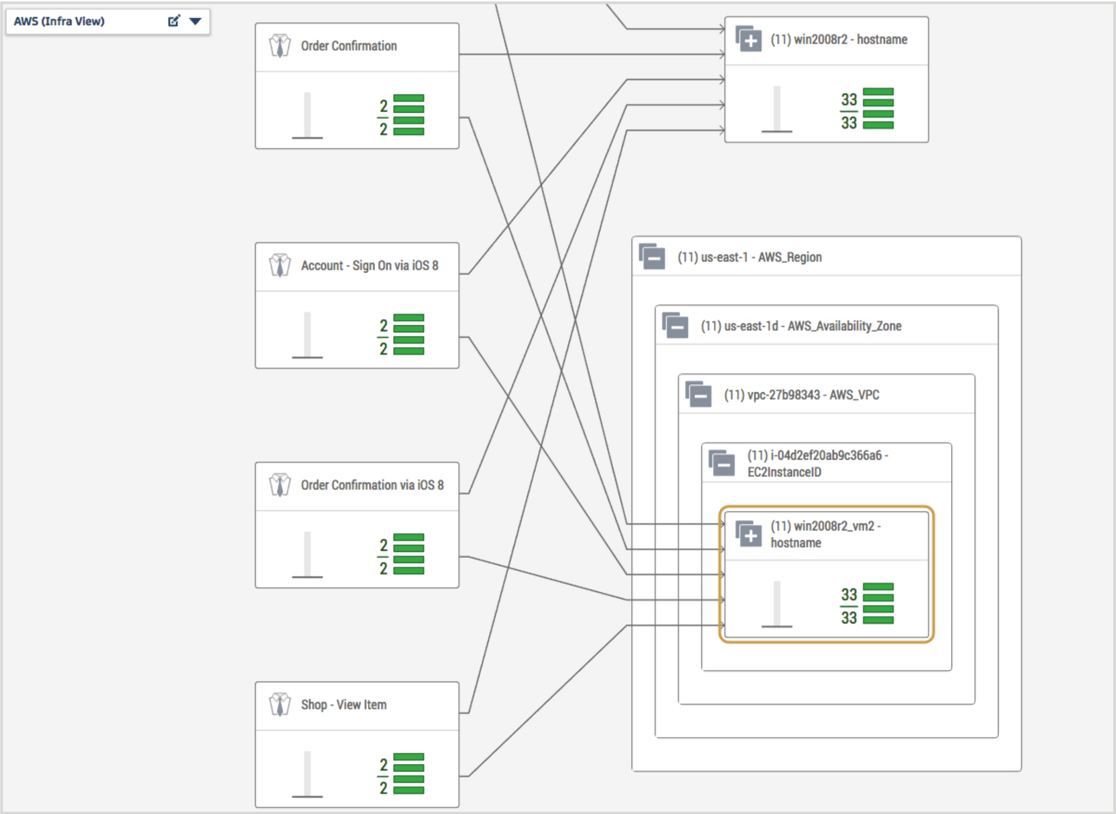

AWS (Infra View)
Order Confirmation
2
2
Account - Sign On via iOS 8
2
2
Order Confirmation via iOS 8
2
2
Shop - View Item
2
2
(11) win2008r2 - hostname
33
33
(11) us-east-1 - AWS_Region
(11) us-east-1d - AWS_Availability_Zone
(11) vpc-27b98343 - AWS_VPC
(11) i-04d2ef20ab9c366a6 - EC2InstanceID
(11) win2008r2_vm2 - hostname
33
33

Figure 3:

AWS infrastructure-based view using CA APM Team Center Perspectives.

Having automatically discovered and modeled the application architecture, modern APM solutions will play a pivotal role in detecting weaknesses in design-for-failure models. Take, for example, a situation where because of increased application load, AWS Auto Scaling has provisioned additional instances to meet demand. While this represents a cloud best practice to cater to increased business activity, it's not immune from problems leading to degraded service. This could include:

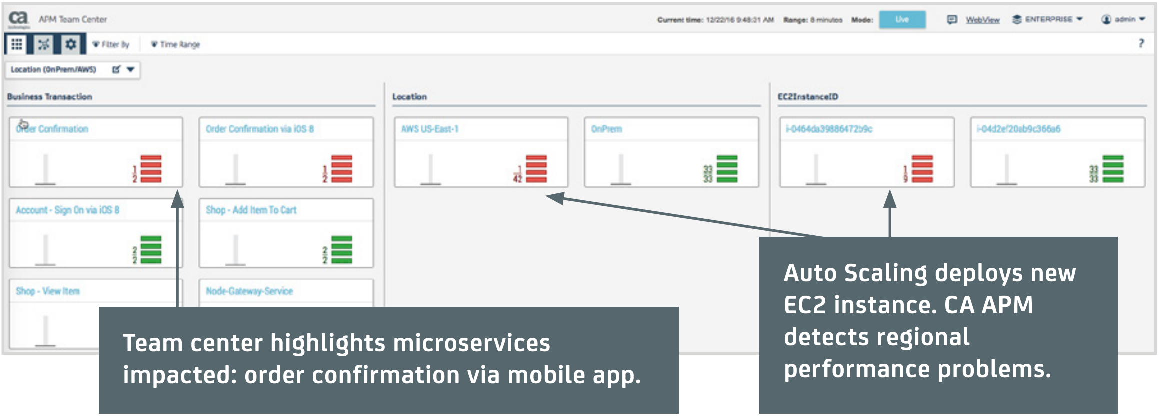
- Failure of the instance itself or supporting elements failing, for example, Elastic Block storage volume
- Incompatible kernel or file system corruption
- Incorrect networking or startup configuration
- Code leaks leading to instance memory exhaustion
- Failure of the physical machine where the EC2 instance resides

Amazon provides many facilities to check the status at both a system and instance level. At a system level, teams can wait for AWS to correct the problem, while for instance related problems teams will need to address the problems themselves. Here, APM solutions can augment the services provided by Amazon, pinpointing problem elements in context of the overarching business services.

Consider the CA APM Team Center view presented in Figure 4 below. During increased business activity, AWS has initiated Auto Scaling to provision additional instance capacity. When this occurs, the team center display is immediately updated to reflect the change, and the new instance is automatically linked to the business service it supports (in this example, an iOS order confirmation service supported by on-premises infrastructure and one EC2 instance). However, and as immediately reflected, a problem occurred that severely impacted business service and the organizations' ability to process mobile phone orders via the order confirmation service. As the view illustrates, the shopping cart is performing normally, but the overall customer experience has been negatively impacted.

Figure 4:

Intelligent EC2 alerts and impact analysis with CA APM Team Center.



Team center highlights microservices impacted: order confirmation via mobile app.

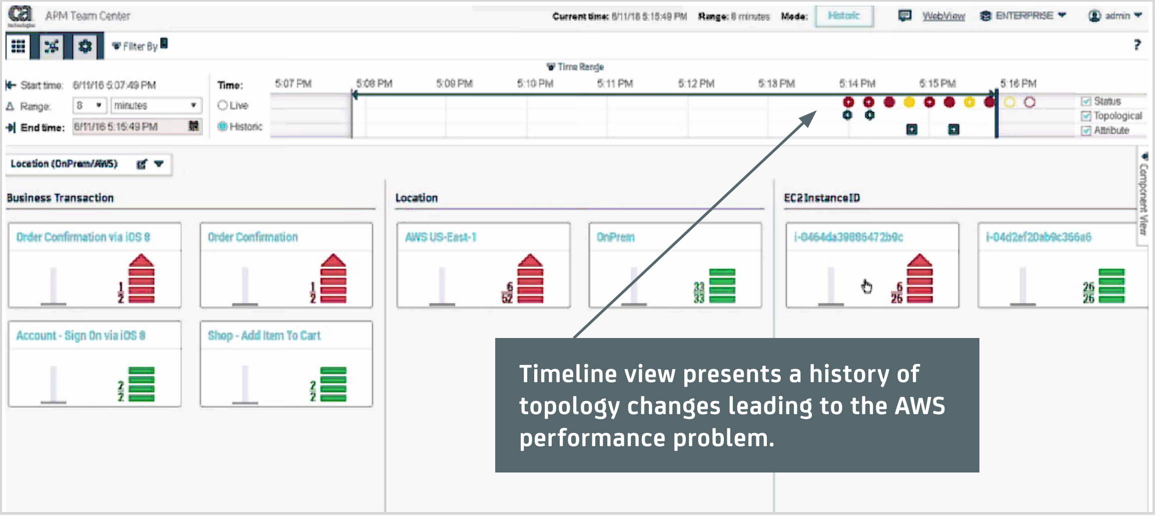
Auto Scaling deploys new EC2 instance. CA APM detects regional performance problems.

Using this interface, cross-functional DevOps teams can organize remediation according to business impact. The interface also pinpoints the problem element (new EC2 instance established from Auto Scaling), which helps avoid lengthy triage time associated with interpreting logs and alarms from multiple sources. Without superfluous and complex maps and alerts, teams see the problem EC2 instance in question, its region (us-east-1), and the business impact.

In situations where unexpected conditions have compromised the design-for-failure model, it's essential that teams quickly identify the root cause of the problem. In this case, APM tools should expose additional information for support analysts or site reliability engineers to determine why the deployment of what should have been an identical EC2 instance deployment is causing problems. As illustrated in Figure 5, the analyst has also invoked CA APM timeline functionality to better understand the series of conditions leading to the performance issue. This functionality is valuable because it enables the analyst to gain change visibility in context of performance.

Figure 5:

CA APM Team Center Timelines—performance impact in context of application changes.

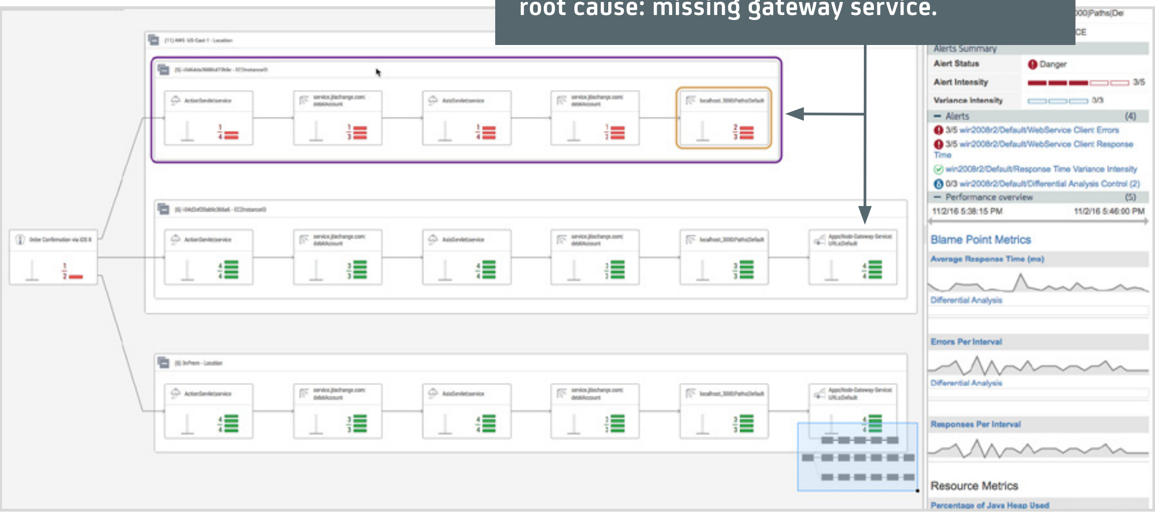


Timeline view presents a history of topology changes leading to the AWS performance problem.

Using this feature, support analysts can see when, where and how the problem started. In this case, a topology change brought about by the EC2 instance deployment is the cause of the performance-related issues. By comparing this against the existing working deployment, teams instantly see that the root cause of the issue is a failure to correctly configure the application when Auto Scaling was initiated. As shown in Figure 6, failure to deploy a node element is the cause of the problem.

Figure 6:

Pinpointing AWS problem root-cause with CA APM Team Center.



Comparing EC2 deployments reveals problem root cause: missing gateway service.

Modern APM solutions can be used to great effect to enhance design for failure models in the problematic area of performance baselining. To date, most monitoring solutions dictate that teams predict acceptable application performance thresholds, set these manually and then receive alerts when thresholds are breached. While resource-intensive, this model has been viable—largely in part due to the static nature of monolithic legacy systems. However, with cloud applications this practice becomes less sustainable.

In distributed systems and microservice-style architectures there are many more performance permutations to consider when establishing monitoring. Defining multiple monitoring approaches may be difficult but still surmountable, however the bigger challenge becomes one of scale. With cloud and microservice applications, the emphasis often shifts to the relationship and communication across distributed services, resulting in a massive increase in the number of alerts to process.

In such cases, old-style performance baselining falls short. Set baselines too low and IT support staff could be inundated with alert storms and false positives. Set them too high and minor but persistent anomalies indicative of more serious and emerging problems can easily be missed. Fortunately, there are statistical analysis methods taken from field-of-quality control that are very applicable in cloud application performance monitoring.

One such approach comes from the groundbreaking statistical work of Walter Shewhart when he worked for Western Electric. By calculating the standard deviation of copper line signal quality, Shewhart demonstrated that simple comparisons against bands of standard deviation could identify points at which the signal exhibited uncontrolled variance; similar to how earthquakes register on a seismometer. Commonly referred to as “how wrong for how long,” these algorithms are useful because they distinguish small nuisance alarms and anomalous trends warranting further action.

This approach that employs standard deviations to establish variance intensities is called differential analysis—and is a core service provided by CA APM.

Unlike traditional baselining that relies on best guess predictions, this technique is purpose-built for highly dynamic cloud applications. By analyzing both severity (how wrong) and time (how long), differential analysis is highly effective in supporting design for failure architectures. Benefits include:

- Rapidly detect any emerging application response time problems across AWS Auto Scaling groups and resources.
- Reduce costs by reliably analyzing performance patterns and trends over periods of time to help match workloads to instance types. This is also a design for failure practice that could help reduce the number of “moving parts”, and hence, opportunity for failure.
- Avoid unnecessary resource provisioning triggered by nuisance events and alarms through Auto Scaling rule optimization based on proven statistical methods.
- Detect slow growing chronic problems and fast acting acute ones through intelligent alerting.
- Triage problems faster by automatically invoking detailed transaction traces when emerging problems are identified.

Section 3:

Summary: Additional Considerations and Benefits

For monitoring, AWS provides rich sets of management APIs and CloudWatch. While possibly enough in a cloud-only scenario, these tools don't measure end-to-end performance and provide more granular troubleshooting.

These potential blind spots become more apparent when aiming to deliver a complete picture of application performance and the user experience. Today's digital services and composite applications rarely operate in isolation and may access both internal and external data sources.

When considering AWS monitoring within a broader context, DevOps practitioners should consider:

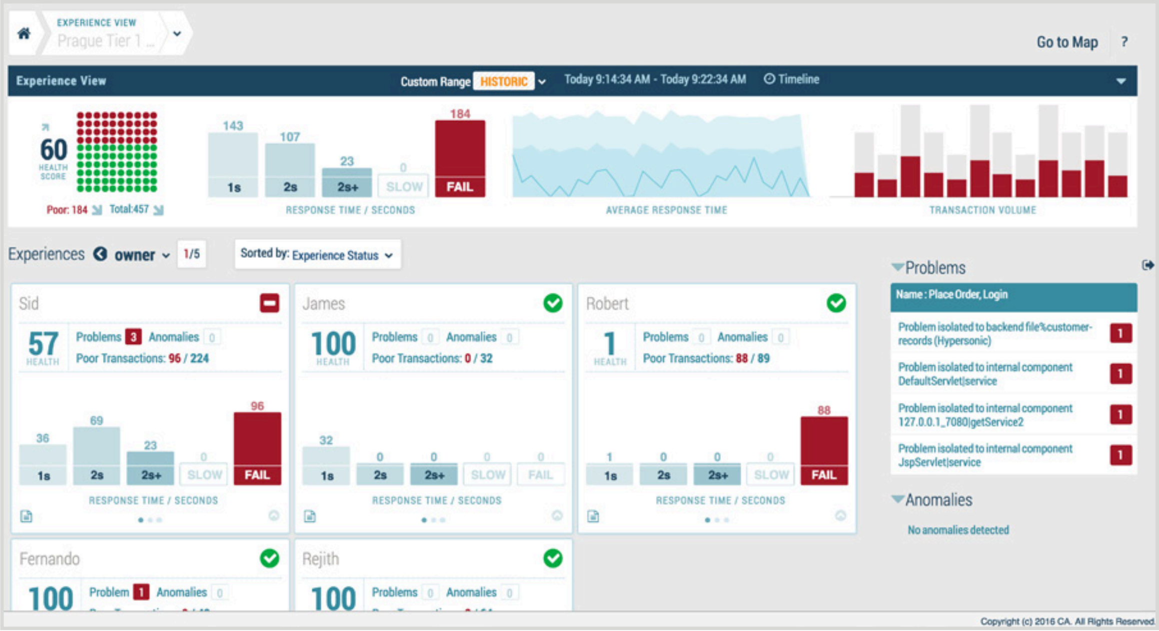
- Integrating AWS monitoring metrics (e.g., via CloudWatch) into a single monitoring platform to track AWS resource usage together with associated on-premises elements.
- Measuring performance from a customer experience for a single version of the truth (see figure 7 on the following page).
- Analyzing trends over time, invoking methods to automatically trigger more actionable alerts based on anomalies and patterns.
- Integrating metrics with network and system infrastructure faults, alarms and logs to gain an aggregated view of performance at a business service level.
- Exploring solutions that incorporate analytics and machine learning, together with assisted triage and workflows.

Additional techniques and practices may include:

- Using APM to help verify that the correct EC2 instances are being used for the right workloads.
- Measuring the performance of common transaction tasks and end-to-end processes (mobile to back end) to better understand the impact of each AWS deployment.
- Moving monitoring into preproduction to help developers identify problematic code; for example, automatically invoking performance monitoring after each build, together with pass-fail conditions.[5]
- Supplementing APM with real-user transaction replication (synthetic monitoring) to ensure complex cloud services are performing optimally across different locations.

Figure 7:

CA APM Experience Views display a health summary of the experiences being delivered across an entire application environment allowing IT Operations to quickly determine poor performance directly impacting customers.



Problem isolated to backend file%customer-records (Hypersonic) 1

Problem isolated to internal component DefaultServlet|service 1

Problem isolated to internal component 127.0.0.1_7080|getService2 1

Problem isolated to internal component JspServlet|service 1

No anomalies detected

Copyright (c) 2016 CA. All Rights Reserved

In conclusion, DevOps practitioners accept that failure is to be expected and leverage the application performance management capabilities discussed in this paper to support modern design for failure models. AWS provides many tools to support these goals, which when integrated with holistic and customer-centric monitoring can help businesses maintain service—even during severe infrastructure events and outages. Taken further, teams can use the learnings from APM analytics to gain critical insights into where architectures can be improved.

To learn more about how to ensure a great customer experience in the cloud with application performance management, visit ca.com/us/why-ca/hybrid-cloud-solutions/ensure-loyalty.html.

About the Authors

Srikant Noorani is a senior technical architect with CA Technologies based out of Toronto, Canada. With more than 18 years of experience in the IT sector, Srikant has worked for both startups and large technology providers, including Sun Microsystems and Fujitsu. Before joining CA, Srikant helped with one of the largest implementations of application performance management at Blackberry.

Peter Waterhouse has been involved in the development, support and marketing of enterprise software solutions for more than 20 years. He has held many management, consulting, technical sales and marketing positions in areas such as cloud computing, DevOps and IT business management. Peter writes and blogs on a range of disruptive business and technology trends, with articles appearing in publications such as Information Week, DevOps.com, Appdevelopemagzine.com, APMdigest and SC Magazine.

[1] Paul Smith, Yolanda Redrup, Financial Review, "Amazon Web Services storm outages serve as a warning of cloud risk to businesses," Jun 6, 2016

[2] Allie Coyne, iTnews, "AWS Sydney outage prompts architecture rethink," Jun 6, 2016

[3] Jinesh Varia, Amazon Web Services, "Architecting for the Cloud: Best Practices," Jan 2011

[4] Jinesh Varia, Amazon Web Services, "Architecting for the Cloud: Best Practices and Design Patterns," Apr 15, 2010

[5] Srikant Noorani, CA Technologies white paper, "Shift-Left Monitoring Strategies for Agile Operations," Nov 2016