O'REILLY®

Compliance at Speed

Achieving Performance in Enterprise Applications

Mark Lustig

4 Easy Ways to Stay Ahead of the Game

The world of web ops and performance is constantly changing. Here's how you can keep up:

1. **Download free reports** on the current and trending state of web operations, dev ops, business, mobile, and web performance. http://oreil.ly/free_resources
2. **Watch free videos and webcasts** from some of the best minds in the field—watch what you like, when you like, where you like. http://oreil.ly/free_resources
3. **Subscribe** to the weekly O'Reilly Web Ops and Performance newsletter. http://oreil.ly/getnews
4. **Attend the O'Reilly Velocity Conference**, the must-attend gathering for web operations and performance professionals, with events in California, New York, Europe, and China. http://velocityconf.com

For more information and additional Web Ops and Performance resources, visit **http://oreil.ly/Web_Ops**.

O'REILLY®

Compliance at Speed

Achieving Performance in Enterprise Applications

Mark Lustig

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

Compliance at Speed

by Mark Lustig

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Editors: Mike Loukides and Brian Anderson

October 2014: First Edition

Revision History for the First Edition:

2014-10-30: First release

2015-05-01: Second release

ISBN: 978-1-491-90987-4

[LSI]

Table of Contents

Introduction

In many industries today, adhering to regulations is not optional; it is mandatory. As information technology professionals, we are constantly challenged with tight timelines for building and enhancing information systems, not just to provide new functionality, but also to ensure our systems meet the guidelines and standards for each industry.

Compliance Affects Everyone, Not Just the Big Banks

Compliance impacts all industries, and is becoming more important every day. Highly regulated industries including financial services and health care must meet strict standards for compliance. For online retailers, privacy and security standards must also be met. The social networking industry is facing regulations specific to consumer protection and the use of customer information.

No industry is immune to meeting compliance requirements, and emerging regulations create more challenges to achieving performance objectives each year, both domestically and internationally. Any website that uses, stores, or processes personal or payment information must address these challenges, notably for security and the payment card industry (PCI), but also for accessibility,access controls, confidentiality, and audit purposes.

Staying abreast of techniques to meet performance goals and compliance regulations is an emerging trend within both performance engineering (PE) and DevOps. Conferences such as Velocity are addressing these topics both tactically and strategically. Tactical, cutting-edge

techniques are taking into account the needs of high-tech and web-facing companies as well as large Fortune® 500 enterprises. Strategically, the emerging cultural paradigm of DevOps is becoming more prominent at larger companies, across complex architectures that include legacy systems.

Performance Is Mandatory for Competitiveness and Business Success

Today's complex system architectures include rich user interfaces, the ability to execute complex business transactions quickly, and the need to provide critical information to users in a variety of formats, both desktop and mobile. How do you ensure you can meet business goals when the system is made up of a combination of web servers, application servers, and multiple middleware layers, including interfaces to web services, databases, and legacy systems? How do you achieve performance goals while meeting regulatory requirements such as multifactor authentication, encryption, and storing years' worth of online transactional data? System designers and architects must understand and manage the performance impacts of mandated features to ensure that service levels can be maintained.

In an effort to accelerate the timelines in providing new systems and enhancing functionality, we're moving from the classic software development methodologies of the past to methodologies based on continuous deployment. Adoption of agile and continuous integration and deployment models enables system functionality to be released more quickly, without sacrificing quality. Regulated industries are struggling to adopt these methodologies, as long-standing release management and testing processes are slow to adapt to accelerated delivery models.

The trend of ubiquitous access is putting more pressure on system performance. Access patterns and user behavior are changing. The mix of concurrent types of users and concurrent access is also forcing a change in how systems are designed to support these emerging trends. We must build systems to achieve performance for all users executing business-critical transactions, regardless of whether a particular user is coming from a desktop PC, a mobile device, or a kiosk. When designing and building the system, we must test to ensure good performance for all users, at the same time.

Case Studies in Performance and Compliance

Throughout this report, we'll highlight various real-world examples. The examples span industries and identify some of the performance challenges created by adhering to regulatory requirements, and the strategies used to address those challenges. Some of these case studies followed the process outlined in this report proactively, while others required addressing the performance issues reactively. The examples have been anonymized to protect the innocent.

To Minimize Reputational Risk, Performance and Compliance Objectives Must Both Be Met

Solving these challenges is not trivial. Business users demand systems that perform well *and* meet regulatory compliance requirements. Often the consequence of complying with mandatory regulations is a reduction of system performance.

Key tenets of performance engineering—workload characterization (e.g., types of transactions, users, volumetrics), disciplined PE processes applied across the software development life cycle, and architectural considerations of performance (load time, throughput/bandwidth)—are required for success.

Through a combination of system optimization techniques at every tier and integration point and the cooperation and commitment of the business to support performance improvement as a critical success factor, performance goals can and will be achieved.

This report outlines a disciplined process that can be followed to achieve your performance goals, while meeting compliance objectives.

Performance Engineering

Performance engineering is not merely the process of ensuring that a delivered system meets reasonable performance objectives; rather, PE emphasizes the "total effectiveness" of the system, and is a discipline that spans the entire software development life cycle. By incorporating PE practices throughout an application's life cycle, scalability, capacity, and the ability to integrate are determined early, when it is still relatively inexpensive to tailor a solution specific to business needs.

Key activities occur at different stages of the life cycle. Notably, these include:

Platform/environment validation: Determine if a particular technical architecture will support an organization's business plan, by employing workload characterization and executing stress, load, and endurance tests.

Workload characterization: A successful performance test requires a workload that simulates actual online and batch transactions as closely as possible. Workshops at which key business and technical professionals agree on representative user profiles help characterize workloads. If batch processing is required, representative messages must be defined. Online profiles are defined by the transactions each one performs.

Capacity planning for performance: Understanding the point at which hardware resources are optimally utilized to support the system's performance goals (e.g., response time, concurrency, and throughput) is critical. Balancing the number of resources while providing resiliency may require horizontal scaling to ensure continuity during failover.

Performance benchmarking: Execute sets of client-specific workloads on a system to measure its performance and its ability to scale. Also execute tests to determine an application's performance limits.

Production performance monitoring: Proactively troubleshoot problems when they occur, and develop repairs or "workarounds" to minimize business disruption.

Challenges to Consider

In today's competitive landscape, business must always consider the performance challenges involved in meeting user expectations. Notably, you must minimize the cost of performance-related outages and enforce service-level agreements (SLAs).

Quantifying the Cost of Poor Performance/ Outages

Understanding the costs of an outage aids in understanding the return on investment (ROI) of proactive performance engineering. Remember, *operational costs "hide" the true cost of system development.* Costs of downtime in production (post-deployment) include the following:

Recovery costs
: These include costs incurred during problem identification, analysis and resolution, and validation testing, as well as external support costs and data recovery costs.

Productivity costs
: These are calculated as duration of outage × total persons affected × average percentage of productivity lost × average employee costs.

Lost revenue
: This is calculated as duration of outage × percentage of unrecoverable business × average revenue per hour.

Consider the example of a company that spends millions of dollars on application support instead of new application development. In this case, each 15-second timeout in the enterprise application integration

(EAI) infrastructure could result in a $5 call to an outsourced contact center. Over the course of six months, this could result in unanticipated support costs of almost $3 million—funds that could otherwise have been used for new development efforts.

In addition to the costs of an outage, it is important to understand the scope of the impact—specifically, who is impacted. For example, an outage that affects the top customers, responsible for the majority of revenue leveraged by the system, carries a much higher weight than one that affects only the smallest customers. When defining service levels for availability and transactions, consider *which* customers are impacted and *when* they're impacted, especially in the context of business "events"(dates, time frames) where access to systems is more crucial.

Service-Level Agreement (SLA) Enforcement

Service-level agreements help organizations meet business objectives. By clearly defining and measuring against goals, organizations can monitor progress internally and in relation to competitors.

SLAs are critical because they provide business metrics and key performance indicators for organizations to manage against. SLAs specific to response time (e.g., a web page must render within three seconds) can be effectively measured with application performance management (APM) technologies. The primary goal of IT is to service the business, and well-defined SLAs provide a clear set of objectives, identifying the activities that are most appropriate to monitor, report, and build incentives around. Few organizations clearly define SLAs.

Service-level agreements should be designed with both organizational costs and benefits in mind. When set too low, business value is negatively affected. When set too high, additional and unnecessary costs may be incurred. Establishing and agreeing on the appropriate service levels requires IT and the business groups to work together to set realistic, achievable SLAs.

Performance Goals

> Measurement is the first step that leads to control and eventually to improvement. If you can't measure something, you can't understand it. If you can't understand it, you can't control it. If you can't control it, you can't improve it.
>
> — H. James Harrington

All systems must strive to avoid the culture of *it's not a problem until users complain*. The non-functional goals for system performance must be part of the overall business requirements. Business requirements are the output of the inception (requirements and analysis) phases of any system development initiative.

Many business initiatives do not effectively define and track the non-functional requirements (NFRs) of response time, throughput, and scalability. Non-functional requirements are not limited specifically to performance, though all have an effect on a system's ability to scale and perform. For reference, common NFRs include:

- Usability
- User and application documentation
- Security
- Transition
- Data conversion
- System capacity and scalability (resource utilization)
- Interoperability
- Robustness
- Performance (response time, throughput, concurrency)
- Reliability
- Availability
- Flexibility
- Maintainability
- Portability[1]

1. Definitions for many of these NFRs, often referred to as Quality Attributes, can be found here.

Key performance objectives and internal incentives should ideally support defining and reporting against service level compliance and regulatory compliance. This can be best accomplished by ensuring there are clearly defined regulatory compliance requirements and well-defined service-level agreements. Managing to these requirements and SLAs can then be enabled by identifying and executing activities to monitor, report, and build incentives around. Keep in mind that any undocumented requirements will likely be missed, and managing these requirements will not be possible.

A critical first step toward defining and implementing SLAs is the identification of the key business transactions, key performance indicators (KPIs), and system transaction volumetrics. Development and PE teams should begin the discussion of service-level agreements and deliver a draft at the end of each analysis phase within each iteration. For example, these may include transaction response times, batch processing requirements, and data retention requirements.

Regulatory requirements including access control, confidentiality, and logging should also be addressed at this time. The requirements will help determine if a performance test and proof-of-concept design validation test is required in order to verify that specific service levels are achievable while meeting these requirements.

Large organizations frequently don't define service-level objectives, and find it difficult to meet these objectives when they're added later, during analysis phases; enforcing them is therefore a challenge.

Performance goals and non-functional requirements must be defined to ensure that a system can be effectively managed.

Effective Searching at a SaaS Digital Storage Provider

To meet compliance goals the system was architected to process emails upon ingestion to facilitate quick retrieval when needed.

At a digital storage records provider, a software-as-a-service (SaaS) email archiving offering was created to support the Sarbanes-Oxley compliance required of financial services institutions. The SaaS provider's customers used the solution to fulfill their compliance requirements for storing every email for at least seven years, with availability for searching and retrieval. The challenge of building custom solutions (including infrastructure), staying current with regulations and technologies, and ensuring adequate capacity was always avail-

able was met by using the SaaS provider. The SaaS provider had to provide the capability to search through more than one billion emails to meet its customers' goals.

This performance challenge was solved by using third-party searching tools from Oracle, which implemented full-text searching. Emails, including bodies and attachments, were indexed on ingestion, in batches and in realtime. Thus, this was a time-space tradeoff, incurring large amounts of storage needed to support the indexing design, with the benefit of performance. Implementation of specific partitioning design patterns also allowed the SaaS provider to meet performance requirements, usually separating data by dates, allowing for parallel searching across large date ranges.

Regulatory Compliance

The term *regulatory compliance* refers to the adherence of an organization to the laws, specifications, regulations, and standards required for an industry. Companies in each industry face unique criteria specific to their industry, and must meet those conditions. Enforcement of standards varies by industry and situation, though penalties for failing to meet them can be severe.

Many regulatory standards exist to protect individuals' and companies' data. Examples of protected data include driver's license numbers, social security numbers, account numbers, credit card numbers, medical records, claims submissions, and any other private information.

Federal Regulations

If you are doing business in the US, here are some of the most important regulations, described in relation to their impact on performance:

Gramm-Leach-Bliley Act (GLBA), 1999
: GLBA is focused on protecting the privacy of consumer information held by financial institutions. It requires companies to provide consumers with privacy notices that explain the financial institutions' information-sharing practices. Consumers have the right to limit some sharing of their information. User access to systems must be recorded and monitored for potential abuse of that data. This requires logging and access controls, which can impact performance.

Health Insurance Portability and Accountability Act (HIPAA), 1996
: HIPAA includes a few key goals. The act requires the protection and confidential handling (encryption) of protected health infor-

mation (PHI), gives American workers the ability to transfer and continue health insurance coverage for themselves and their families when they change or lose their jobs, and mandates industry-wide standards for health care information for electronic billing and other processes. User access to systems must be monitored, and data must be secure throughout all transactions. The requirements for confidentiality and access control can impact performance.

Sarbanes-Oxley (SOX), 2002
: The purpose of the SOX Act is to oversee financial reporting processes for finance professionals. It includes reviewing legislative audit requirements and protecting investors through more accurate corporate disclosures. The act established a public company accounting oversight board and deals with issues of auditor independence, corporate responsibility, and enhanced financial disclosure. User access, including login and transactions, must be recorded and monitored, adding overhead to all activity.

Children's Online Privacy Protection Act (COPPA), 1998
: COPPA prohibits websites from collecting personally identifiable information from children under 13 without parental consent. It mandates website operators to collect only "reasonably necessary" personal information for an online activity. Recent revisions (2013) to this act address changes in the way children use and access the Internet, including the increased use of mobile devices and social networking. The modified rule widens the definition of children's personal information to include persistent identifiers such as cookies that track a child's activity online, as well as geolocation information, photos, videos, and audio recordings. Requiring an online "permission slip" adds system activity to check if permission has been granted, in addition to the overhead of the transactions required to obtain the authorization initially. Rules for captured data must also be configured to support this data access. This requires access controls, which can impact performance, as the authentication and authorization requirements require additional system activity for each request.

Family Educational Rights and Privacy Act (FERPA), 1974 and 2011
: FERPA is intended to protect the rights of students and to ensure the privacy and accuracy of education records. The act applies to all institutions that are recipients of federal aid administered by the Secretary of Education. It prevents the disclosure of personally

identifiable information (PII) in a student's education record without the consent of a parent or eligible student. As with COPPA, the checks for permission to access data—including rules, access controls, and authorization checks for each system request—result in additional system activity.

International Laws and Regulations

Globally accessible applications may need to comply with multiple laws and regulations from other countries. An example of this is the European Union (EU) Data Protection Initiative (Directive 95/46/EC), which requires protecting the privacy of all personal data collected for or about citizens of the EU. In these cases the application architect must consider if it makes sense for the application to adhere to a superset of regulations, if one can be found (e.g., use the highest encryption level that is required across all the countries), or to selectively implement different regulations based on each country. Multiple code bases may be practical, with the goal of achieving optimal performance for the user base.

For example, the security requirements for 10% of users may impact performance severely for those users; the other 90% of users may require a lower level of encryption, and implementing a two-tiered system can rsult in increased performance for the vast majority of users. The trade-off is based on the performance impacts of implementing different levels of regulations versus the operational impact of managing the diverse implementations. The latter may require multiple deployments of some components based on country, or additional code complexity to handle the country differences.

The Foreign Corrupt Practices Act (FCPA) is also worth noting. FCPA prohibits companies from paying bribes to foreign political figures and government officials for the purpose of obtaining business. Many companies may use third-party vendors as representatives in foreign countries. This isn't as much of a technical issue but may hinder a company's ability to choose vendors.

The Primary Challenge

Considering these well-known regulations, which represent a subset of federal regulations, it quickly becomes apparent that systematic controls must be put in place when building systems to ensure com-

pliance. The primary challenge and objective is achieving the non-functional goals of performance while meeting key regulatory requirements with regard to access control, confidentiality, and logging.

Aligning Performance Objectives with Compliance Regulations

Meeting both compliance and performance objectives requires structure and discipline. Compliance is usually a functional requirement while performance is most often a non-functional requirements. A structured process will achieve better overall performance by defining and tracking both functional and non-functional requirements together. Meeting both objectives can be accomplished by following the process outlined in the remainder of this report. This process includes the following steps:

1. Define the business goals for performance.
2. Identify constraints. These include:
    a. Business constraints
    b. Regulatory and compliance constraints
3. Design and develop for performance goals.
4. Execute performance measurement and testing.
5. Implement performance monitoring.
6. Mitigate risks.

1. Define the Business Goals for Performance

Ultimately, the goal of system development is to meet the business goals of your organization. Business goals include meeting compliance objectives. Without the business, information technology is irrelevant.

Understanding the business goals must be the first step, and understanding the system performance goals and expectations is of primary importance. For example, if the business goals of a financial services provider include executing more financial transactions (i.e., money transfers) in a shorter period of time, the business problem translates to clear performance expectations. The transactions per second (TPS) rate required from the system can be calculated. The process of defining the business performance goals must be disciplined and thorough and include the business partners.

The business motivation for visibility into performance goals must also be captured. This will translate into the reporting requirements and metrics used by the IT department and the corporate internal business users, and external customers. The metrics and reporting requirements can be used to define the reports and dashboards used when monitoring the system and business transactions.

2. Identify Constraints

Once performance goals have been established, project constraints must be well understood. Constraints typically include resource (i.e., hardware, software, network), geography (i.e., location of users and the infrastructure), and time (i.e., operating windows) constraints, and regulatory compliance requirements regarding access control, confidentiality, and logging.

Understanding, defining, and documenting constraints requires communication with business partners. As constraints are constantly changing, staying current with emerging regulations is also critical. Depending on the size of the organization, an internal compliance team may be responsible for identifying and auditing systems for compliance. In other cases, outside agencies can be used.

2a. Identifying Business Constraints

As part of the business requirements phase, functional and non-functional requirements are defined and documented. Functional requirements define *what* the system must do. Non-functional requirements define *how* the system must do it. Business constraints may be subtle. For example, marketing campaigns can affect the way a system is implemented. Consider the scenario of a marketing campaign banner image that is presented to a user upon logging in to a secure home page. The image for the banner may be selected from multiple cam-

paigns depending on rules defined by the business. The retrieval of the image requires selection of the campaign by the rules implemented by the business. This flexibility results in targeted marketing campaigns based on user characteristics and behavior. The constraint is the need to process the business rules during the page rendering process. This constraint requires additional processing and must be considered in the design of the system.

2b. Identifying Regulatory and Compliance Constraints

Access control, confidentiality, and logging are the primary compliance requirements that must be defined, documented, and implemented in such a way as to minimize performance impact.

Access control is often implemented using role-based access models. Depending on the implementation model, achieving robust performance may be difficult. Access control must be enforced at both the authentication layer and the services layer. In many cases, back-end transactions are required to verify access to the service being called. This level of access control must be implemented with performance in mind, reducing the overall number of transactions to ensure compliance. This can be a challenging model to implement.

Confidentiality is typically addressed via encryption, both for passwords and for confidential data. Confidential data cannot be stored or transmitted in clear text. Regulations dictate the security policies that must be followed to ensure compliance.

Logging may be required to ensure compliance. Synchronous logging implementations can slow down performance. A common technique to reduce the performance impact is to leverage asynchronous logging and auditing techniques.

Security Compliance for a Large Hardware Provider

Corporate security implemented the best practice configuration rules to limit Distributed Denial of Service (DDoS) attacks but did not consider scenarios where the configuration would prohibit certain use cases.

A large portal-based application used Apache web servers for the front-end presentation tier. Corporate security manadated security requirements which caused performance issues, as the settings restricted transaction duration. These settings were based on regula-

tions interpreted incorrectly by corporate security to minimize transaction duration and vulnerability to DDoS attacks. The timeout was set for less than 30 seconds to tighten security as much as possible and ensure compliance. These settings were applied globally and affected all system transactions. Unfortunately not all transactions were able to execute within this time frame, and responses for some reporting functions exceeded the threshold.IT revisited the standards and worked with business representatives and was able to increase the timeout and receive an exemption for these reporting transactions.

3. Design and Develop for Performance Goals

When designing a system, performance must be a priority. Understanding the demands that may be placed on them—particular functions, batch jobs, or components—should be at the top of a developer's to-do list when designing and building systems. Early in the design process, developers should test code and components for performance, especially for complex distributed architectures. For example, if 50 services are going to be built using a framework including web services, middleware, databases, and legacy systems, a proof-of-concept (POC) performance test should be part of the design process. After building out two or three key transactions based on the proposed architecture, run the test. This will help determine if the design will scale to support the expected transaction load *before* the entire system is built.

Many strategies can be designed into the system to ensure optimal performance. Some examples include asynchronous logging and caching of user attributes and shared system data. Being judicious is always recommended if there's a requirement that only affects certain customers. It's worth considering multiple code sets depending on the requirements of key customers. For example, if 90% of users won't see a benefit from preloading data, the code to pre-load/cache data should be built in such a way as to only support the 10% of users that will see the performance benefit.

4. Execute Performance Measurement and Testing

Performance measurement requires discipline to ensure accuracy. In order to identify and establish specific tests, the PE team must model, via a workload characterization model, real-world performance expectations. This provides a starting point for the testing process. The team can modify and tune the model as successive test runs provide additional information. After defining the workload characterization model, the team needs to define a set of user profiles that determine the application pathways that typical classes of users will follow. These profiles are used and combined with estimates from business and technical groups throughout the organization to define the targeted performance behavior criteria. These profiles may also be used in conjunction with predefined performance SLAs as defined by the business.

Once the profiles are developed and the SLAs determined, the performance test team needs to develop the typical test scenarios that will be modeled and executed. In addition, the performance test environment must be identified and established. This may require acquiring hardware and software, or can be leveraged from an existing or shared environment. At a minimum, the test environment should closely represent the production environment, though it may be a scaled-down version.

The next critical part of performance testing is identifying the quantity and quality of test data required for the performance test runs. This can be determined through answering different questions: Are the test scenarios destructive in nature to the test bed of data? Can the database be populated in such a way that it's possible to capture a snapshot of the database before any test run and restored between test runs? Can the test scenarios create the data that they require as part of a setup script, or does the business complexity of the data require that it be created one time up front and then cleaned up as part of the test scenarios? One major risk to the test data effort, if using an approach leveraging actual test scripts, is that one of the test scripts may fail during the course of the test runs and the data will have to be recreated anyway, using external tools or utilities.

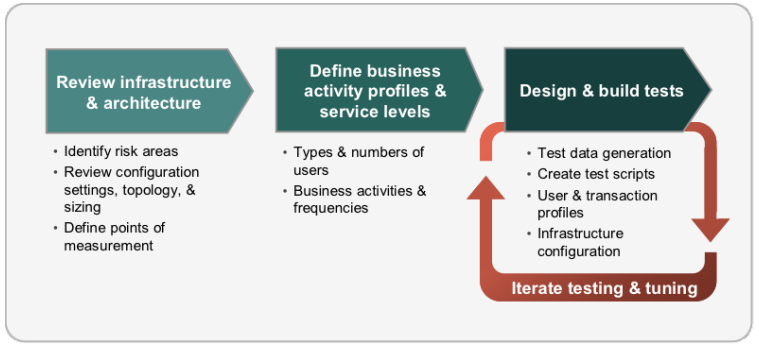
As soon as these test artifacts have been identified, modeled, and developed, the performance test can begin with an initial test run, mod-

eling a small subset of the potential user population. This is used to shake out any issues with the test scripts or test data used by the test scripts. It also validates the targeted test execution environment including the performance test tool(s), test environment, system under test (SUT) configuration, and initial test profile configuration parameters. In effect, this initial test is a "smoke test" of the performance test runtime environment.

At the point when the PE smoke test executes successfully, it is time to reset the environment and data and run the first of a series of test scenarios. This first scenario will provide significant information and test results that can be used by the performance test team defining the performance test suites.

The performance test is considered complete when the test team has captured results for all of the test scenarios making up the test suite. The results must correspond to a repeatable set of system configuration parameters as well as a test bed of data.

The following diagram outlines the overall approach used for assessing the performance and scalability of a given system. These activities represent a best-practices model for conducting performance and scalability assessments.



Each test iteration attempts to identify a system impediment or prove a particular hypothesis. The testing philosophy is to vary one element, then observe and analyze the results. For example, if results of a test are unsatisfactory, the team may choose to tune a particular configuration parameter and then rerun the test.

Proactive Vulnerability Testing for Enterprise Systems

IT security scans may impact system availability. IT security needs to partner with application teams to balance coverage without impacting systems.

At many large corporations, regulations are enforced by running automated security scans. These scans can run continuously and have adverse effects on performance and availability. The scans either slow down performance dramatically or, even worse, cause faults within running processes requiring their restart. Interpretation of regulations must be carefully implemented to ensure compliance and balance the performance impacts. A recent *Wall Street Journal* editorial criticized Federal Trade Commission monitoring of IT departments at companies that had security breaches, causing overreactions at times. Adjusting the schedule minimized the impact of these automated scans as well as ensuring adequate system resources were available.

5. Implement Performance Monitoring

The increased complexity of today's distributed and web-based architectures has made it a challenge to achieve reliability, maintainability, and availability at the levels that were typical of traditional systems implementations. The goal of systems management and production performance monitoring is to enable measurable business benefits by providing visibility into key measures of system quality.

To be proactive, companies need to implement controls and measures that either enable awareness of potential problems or target the problems themselves. Application performance monitoring (APM) not only ensures that a system can support service levels such as response time, scalability, and performance, but, more importantly, proactively enables the business to know when a problem will arise. When difficulties occur, PE, coupled with APM, can isolate bottlenecks and dramatically reduce time to resolution. Performance monitoring allows proactive troubleshooting of problems when they occur and facilitates developing repairs or "workarounds" to minimize business disruption.

Organizations can implement production performance monitoring to solve performance problems, and leverage it to inhibit unforeseen

performance issues. It establishes controls and measures to sound alarms when unexpected issues appear, and isolates them. Unfortunately, the nature of distributed systems development has made it challenging to build in the monitors and controls needed to isolate bottlenecks, and to report on metrics at each step in distributed transaction processing. In fact, this has been the bane of traditional systems management. However, tools and techniques have matured to provide end-to-end transactional visibility, measurement, and monitoring.

Aspects of these tools include dashboards, performance monitoring databases, and root cause analysis relationships allowing tracing and correlation of transactions across the distributed system. Dashboard views provide extensive business and system process information, allowing executives to monitor, measure, and prepare based on forecasted and actual metrics. By enabling both coarse and granular views of key business services, they allow organizations to more effectively manage customer expectations and business process service levels, and plan to meet and exceed business goals. In short, they deliver the right information to the right people, at the right time. It is important to define what needs to be measured based on the needs of the business and IT.

Understanding application performance and scalability characteristics enables organizations to measure and monitor business impacts and service levels, further understand the end user experience, and map dependencies between application service levels and the underlying infrastructure. The integration of business, end user, and system perspectives enables management of the business at a service and application level.

6. Mitigate Risk

As risks are identified through analysis of test results and application performance monitors, the impact of these risks must be categorized. Sample categories include:

- Business impact
  - — Regulatory impacts for outages
  - — High financial impact for outages
  - — Application supports multiple lines of business
  - — Application classified as business critical

  - Application supports contractual SLAs
- User population
  - Application has geographically diverse users (domestic, international)
  - High rate of user population or concurrency growth expected
- Transaction volumes
  - "Flash" events may dramatically increase volumes

As risks are identified, specific solutions and recommendations must be developed to minimize and resolve these issues. The release and deployment model will influence how and when a particular solution or change is implemented. For example, if caching is going to be added, will this be implemented in a single release or will components be deployed in successive releases? Less code-invasive changes such as hardware configuration or changes isolated to a single tier (i.e., additional database indexes) may be able to be handled in minor or emergency releases.

Security Compliance for a Large Financial Services Provider

Meeting compliance requirements to store seven years' worth of data can lead to challenges in database table design to efficiently accommodate large data sets.

Financial services compliance applications consist of very complex functionality, often relying heavily on the database layer to store metadata and configuration information for multiple financial plan and benefits combinations. This results in the need for a stable and performant data model. Compliance often requires storage of transactional data for a period of seven years, in an online manner, resulting in potentially very large tables. Without accurate statistics for the database optimizer to rely upon, large table sizes can result in slow-running SQL and stored procedures. IT created a purging strategy and table partitioning strategies to limit the amount of data fetched in each request to enable fast and consistent data access response. In addition, the application tier was experiencing slow response times due to large amounts of computations for each request. Performance was improved through load balancing across multiple application

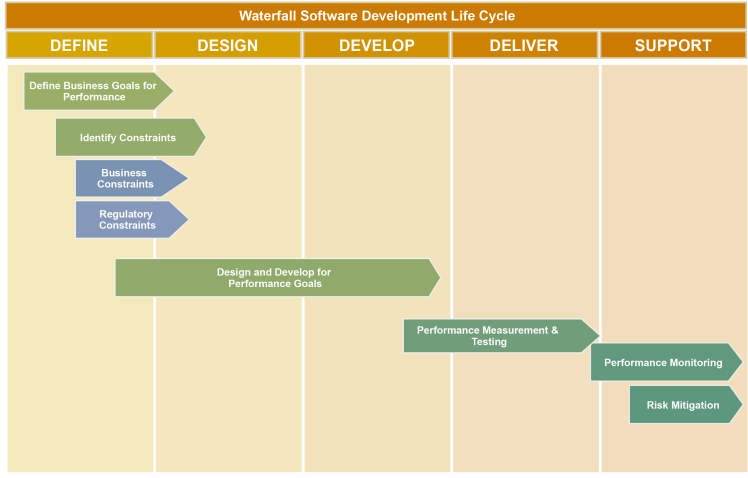
servers and increasing the number of application threads to leverage more CPU resources.

Development Methodology Considerations

Software development methodologies vary by implementation and framework. Depending on the standards defined for an organization, the methodology followed may be dictated by the enterprise, or, if multiple methodologies are supported, it may depend on the requirements/demands of the project. The process for achieving performance goals while addressing compliance requirements is applicable to and consistent across multiple methodologies, as portrayed in the diagrams that follow.

Waterfall

The waterfall model is still followed by very large organizations for many critical system implementations. This progressive development process provides a disciplined structure, as well as checkpoints, to support a predictable set of requirements and releases. This disciplined and rigid methodology requires both functional and non-functional requirements to be captured during the requirements phase and applied to the full development life cycle. Compliance requirements are typically captured as functional requirements, while the non-functional requirements include performance and scalability.



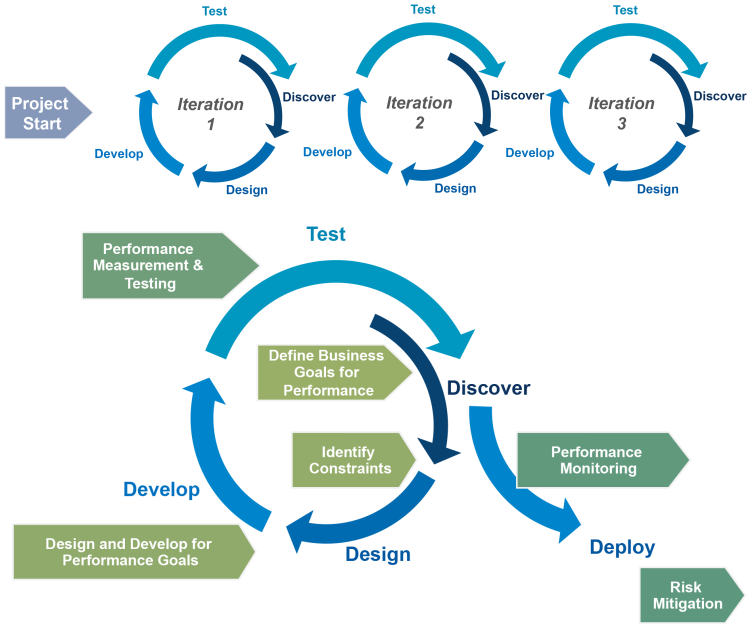
Iterative Development: Agile and Scrum

Functional compliance requirements and performance can also be effectively addressed when following agile and Scrum methodologies.

Many companies, including high-tech organizations and startups, have adopted agile as their primarily development methodology. Flexible and iterative development allows functional and non-functional requirements to be addressed in multiple iterations. Ideally, compliance requirements are captured as functional requirements in the early iterations.

Iterative and agile methods allow building of software in the form of completed, finished, and ready-for-use iterations or blocks, beginning with the blocks perceived to be of the highest value to the customer.

Scrum is an agile development model based on multiple small teams working independently. Within each iteration, certain steps must be followed to ensure the performance goals are defined, tested, and monitored.



Iteration Detail

Following the disciplined process discussed above will enable you to meet both performance and compliance objectives. This process is

applicable to multiple development methodologies. By understanding the business needs, the system workload, and the reporting requirements, you'll be able to measure and monitor real world performance. This will ensure meeting the goals of performance and compliance requirements, providing visibility into key measures of system quality, all while proactively mitigating risks.

Conclusion

Greenfield solutions rarely exist in highly regulated industries. Achieving enterprise performance requires navigating regulatory compliance and systems constraints. The goal is to meet compliance requirements while minimizing any reductions in system performance.

Though many highly regulated industries are slow to adopt continuous integration and deployment models, addressing performance across the development life cycle and within each iteration will ensure reaching performance goals. Across all industries, regulations and requirements affect performance; maintaining performance as a primary objective will enable success.

The primary objective for organizations is to ensure that they are aware of and take steps to comply with relevant laws and regulations while minimizing any impact on system performance. Addressing this challenge takes discipline and an understanding of existing and emerging regulations. Following the process outlined in this paper can and will enable success.

References for This Report

- GLBA
- HIPAA
- Sox
- COPPA
- FERPA

About the Author

Mark Lustig leads the Performance Engineering Practice for Collaborative Consulting. Mark has solved challenging performance issues for numerous Fortune® 500 companies, across a breadth of industries, notably financial services, insurance, and healthcare. In addition to being a hands-on performance engineer, Mark specializes in application and technology architecture for multi-tiered Internet and distributed systems. His 20+ years of experience in the high-technology arena includes systems architecture and performance engineering expertise, particularly in designing, implementing, and tuning applications and solving large-scale performance problems for enterprise systems.