Certified Tester

Performance Testing Syllabus

CT-PT

versão 1.0 (2018)

International Software Testing Qualifications Board

Provided by American Software Testing Qualifications Board and German Testing Board

Versão traduzida, adaptada e disponibilizada na Língua Portuguesa pelo *WGT BSTQB* - Grupo de Traduções do BSTQB

Direitos autorais

Grupo de Trabalho de Testes de Performance:

Graham Bath

Rex Black

Alexander Podelko

Andrew Pollner

Randy Rice

Histórico da Revisão

Versão	Data	Observações
Alpha V04	13 December 2016	Version for NYC Meeting
Alpha V05	18 December 2016	After NYC Meeting
Alpha V06	23 December 2016	Restructure Ch 4 Renumbering and adjusting LOs as agreed in NYC
Alpha V07	31 December 2016	Add author comments
Alpha V08	12 February 2017	Pre-Alpha version
Alpha V09	16 April 2017	Pre-Alpha version
Alpha Review V10	28 June 2017	For the Alpha Review
V2017v1	27 November 2017	For Alpha Review
V2017v2	15 December 2017	Alpha updates
V2017v3	15 January 2018	Technical edit
V2017v4	23 January 2018	Glossary review
V2018 b1	1 March 2018	Beta Candidate for ISTQB
V2018 b2	17 May 2018	Beta Release for ISTQB
V2018 b3	25th August 2018	Beta review comments incorporated for release version
Version 2018 (1.0)	9 December 2018	ISTQB GA Release

Histórico da versão BSTQB

Versão	Data	Observações
1	14/06/2023	Padronização do layout com o ISTQB
2	29/06/2024	Adequação visual

Índice

Agradecimentos

Este documento foi produzido pelo *American Software Testing Qualifications Board* (ASTQB) e pelo *German Testing Board* (GTB):

Graham Bath (GTB, Working Group co-chair)
Rex Black
Alexander Podelko (CMG)
Andrew Pollner (ASTQB, Working Group co-chair)
Randy Rice

A equipe principal agradece à equipe de revisão por suas sugestões. O ASTQB gostaria de reconhecer e agradecer ao *Computer Measurement Group* (CMG) por suas contribuições no desenvolvimento deste syllabus.

As seguintes pessoas participaram da revisão, comentários ou votação deste programa ou de seus predecessores:

Dani Almog
Sebastian Chece
Todd DeCapua
Wim Decoutere
Frans Dijkman
Jiangru Fu
Matthias Hamburg
Ágota Horváth
Mieke Jungebload
Beata Karpinska
Gábor Ladányi
Kai Lepler
Ine Lutterman
Marek Majernik
Stefan Massonet
Judy McKay
Gary Mogyorodi
Joern Muenzel
Petr Neugebauer
Ingvar Nordström
Meile Posthuma
Michaël Pilaeten
Filip Rechtoris
Adam Roman
Dirk Schweier
Marcus Seyfert
Péter Sótér
Michael Stahl
Jan Stiller
Federico Toledo
Andrea Szabó
Yaron Tsubery
Stephanie Ulrich
Mohit Verma
Armin Wachter
Huaiwen Yang
Ting Yang

Este documento foi formalmente lançado pelo ISTQB em 9 de dezembro de 2018.

O BSTQB® agradece aos voluntários do WGT BSTQB® pelo empenho e esforço na tradução e revisão deste material: Eduardo Medeiros Rodrigues, George Fialkovitz Jr., Paula B. de Oliveira, Osmar Higashi, Rogério Athaide Almeida, Stênio Viveiros.

0 Introdução

0.1 Finalidade deste documento

Este syllabus forma a base para a *Certified Tester Foundation Level – Performance Testing* (CT-PT). O ASTQB® e o GTB® fornecem este syllabus da seguinte forma:

1. Para os Conselhos Nacionais, para traduzirem a seu idioma local e credenciar provedores de treinamento. Os Conselhos Nacionais podem adaptar o plano de estudos às suas necessidades linguísticas específicas e modificar as referências para se adaptarem às suas publicações locais.
2. Para os Conselhos de Exames, para derivar questões de exame em sua língua local adaptadas aos objetivos de aprendizagem de cada syllabus.
3. Aos Provedores de Treinamento, para produzir material didático e determinar métodos de ensino apropriados.
4. Para os Candidatos à certificação, para se prepararem para o exame (como parte de um curso de treinamento ou de forma independente).
5. Para a Comunidade Internacional de Software e Engenharia de Sistemas, para promover a profissão de software e testes de sistemas, e como base para livros e artigos.

O ASTQB e o GTB podem permitir que outras entidades utilizem este programa para outros fins, desde que procurem e obtenham permissão prévia por escrito.

0.2 A certificação de nível fundamental em Teste de Performance

A certificação *Certified Tester Foundation Level – Performance Testing* (CT-PT) é destinada a qualquer pessoa envolvida em teste de software que deseje ampliar seus conhecimentos de testes de performance ou qualquer pessoa que deseje iniciar uma carreira de especialista em testes de performance. A qualificação também é destinada a qualquer pessoa envolvida na engenharia de performance que deseja obter um melhor entendimento desses testes.

O syllabus considera os seguintes aspectos do teste de performance:

- Aspectos técnicos.
- Aspectos baseados no método.
- Aspectos organizacionais.

As informações sobre o teste de performance descritas no syllabus ISTQB® *Advanced Level Technical Test Analyst* [ISTQB_ALTTA_SYL] são consistentes e desenvolvidas por este syllabus.

0.3 Resultados de negócios

Esse capítulo lista os Resultados de Negócios esperados de um candidato que tenha obtido a certificação do nível fundamental em Teste de Performance.

PTFL-1 Entender os conceitos básicos de eficiência de performance e teste de performance.

PTFL-2 Definir riscos, metas e requisitos de performance para atender às necessidades e expectativas dos stakeholders.

PTFL-3 Entender as métricas de performance e como coletá-las.

PTFL-4 Desenvolver um plano de teste de performance para atingir metas e requisitos estabelecidos.

PTFL-5 Conceitualmente, projetar, implementar e executar testes básicos de performance.

PTFL-6 Analisar os resultados de um teste de performance e as implicações para os stakeholders.

PTFL-7 Explicar o processo, a lógica, os resultados e as implicações dos testes de performance para os stakeholders.

PTFL-8 Entender as categorias e usos de ferramentas de performance e critérios para sua seleção.

PTFL-9 Determinar como as atividades de teste de performance se alinham com o ciclo de vida do software.

0.4 Objetivos de aprendizagem examináveis

Os Objetivos de Aprendizagem apoiam os Resultados de Negócios e são usados para criar exames para se obter a Certificação de Teste de Performance do Nível Fundamental. Os Objetivos de Aprendizagem são classificados para um nível cognitivo de conhecimento (nível K).

Um nível K, ou nível cognitivo, é utilizado para classificar os objetivos de aprendizagem de acordo com a taxonomia revisada de Bloom [Anderson01]. O ISTQB® usa essa taxonomia para projetar seus exames.

Este syllabus considera quatro níveis-K diferentes (K1 a K4):

K1 (Lembrar): O candidato deve lembrar ou reconhecer um termo ou um conceito.

K2 (Entender): O candidato deve selecionar uma explicação para uma declaração relacionada ao tópico da pergunta.

K3 (Aplicar): O candidato deve selecionar a aplicação correta de um conceito ou técnica e aplicá-lo a um determinado contexto.

K4 (Analisar): O candidato pode separar informações relacionadas a um procedimento ou técnica em suas partes constituintes para melhor compreensão e pode distinguir entre fatos e inferências.

Em geral, todas as seções desse syllabus são examináveis em um nível K1. Ou seja, o candidato reconhecerá, lembrará e recordará um termo ou conceito. Os objetivos de aprendizagem nos níveis K2, K3 e K4 são mostrados no início do capítulo pertinente.

0.5 Tempos de treinamento recomendados

Um tempo mínimo de treinamento foi definido para cada objetivo de aprendizagem neste programa. O tempo total de cada capítulo é indicado no título do capítulo.

Os provedores de treinamento devem observar que outros syllabi do ISTQB aplicam uma abordagem de tempo padrão que aloca horários fixos de acordo com o nível cognitivo. O syllabus de testes de performance não aplica estritamente este esquema. Como resultado, os provedores de treinamento recebem uma indicação mais flexível e realista dos tempos mínimos para cada objetivo de aprendizado.

0.6 Requisitos de entrada

O *Cetified Tester Foundation Level* (CTFL) deve ser obtido antes de fazer o exame *Certified Tester Foundation Level - Performance Testing* (CT-PT).

0.7 Fontes de informação

Os termos usados neste Syllabus estão definidos no Glossário de termos do ISTQB usado no teste de software [ISTQB_GLOSSARY].

O último capítulo contém uma lista de livros e artigos recomendados sobre testes de performance.

1 Conceitos básicos [60 min]

Palavras-chave

teste de capacidade de concorrência, teste de resistência, geração de carga, teste de carga, teste de performance, teste de escalabilidade, teste de pico, teste de estresse

Objetivos de Aprendizagem

1.1 Princípios do Teste de Performance

PTFL-1.1.1 (K2) Compreender os princípios do teste de performance.

1.2 Tipos de Teste de Performance

PTFL-1.2.1 (K2) Compreender os diferentes tipos de testes de performance.

1.3 Testando os tipos de Teste de Performance

PTFL-1.3.1 (K1) Chamar os tipos de teste no teste de performance.

1.4 O Conceito de geração de carga

PTFL-1.4.1 (K2) Entender o conceito de geração de carga.

1.5 Modos comuns de falha de eficiência de performance e suas causas

PTFL-1.5.1 (K2) Dar exemplos de modos de falha comuns de testes de performance e suas causas.

1.1 Princípios do Teste de Performance

A eficiência da performance (ou simplesmente desempenho) é uma parte essencial de fornecer uma boa experiência aos usuários quando eles usam seus aplicativos em uma variedade de plataformas fixas e móveis. O teste de performance desempenha um papel crítico no estabelecimento de níveis de qualidade aceitáveis para o usuário final e, muitas vezes, está intimamente integrado a outras disciplinas, como engenharia de usabilidade e engenharia de performance.

Além disso, a avaliação da adequação funcional, usabilidade e outras características de qualidade sob condições de carga, como durante a execução de um teste de performance, pode revelar problemas específicos de carga que afetam essas características.

O teste de performance não se limita ao domínio baseado na Web, no qual o usuário final é o foco. Também é relevante para diferentes domínios de aplicações com uma variedade de arquiteturas de sistemas, como o clássico cliente-servidor, distribuído e embarcado. Tecnicamente, a eficiência de performance é categorizada na ISO-25010 [ISO25000] *Product Quality Model* como uma característica de qualidade não funcional, com as três subcaracterísticas descritas abaixo. O foco e a priorização adequadas dependem dos riscos avaliados e das necessidades dos vários stakeholders. A análise dos resultados dos testes pode identificar outras áreas de risco que precisam ser abordadas.

Comportamento do tempo: geralmente, a avaliação do comportamento do tempo é o objetivo mais comum dos testes de performance. Esse aspecto examina a capacidade de um componente ou sistema responder às entradas do usuário ou do sistema dentro de um tempo específico e sob condições específicas. As medições do comportamento do tempo podem variar do período de ponta-a-ponta usado pelo sistema para responder à entrada do usuário, até o número de ciclos de CPU requeridos por um componente de software para executar uma tarefa específica.

Utilização de recursos: se a disponibilidade de recursos do sistema for identificada como um risco, a utilização desses recursos (p. ex., a alocação de memória RAM limitada) poderá ser investigada por meio da realização de testes específicos de performance.

Capacidade: se as questões de comportamento do sistema nos limites da capacidade exigida do sistema (p. ex., números de usuários ou volumes de dados) forem identificadas como um risco, os testes de performance podem ser realizados para avaliar se a arquitetura do sistema é adequada.

O teste de performance geralmente assume a forma de experimentação, que permite a medição e a análise de parâmetros específicos do sistema. Esse pode ser conduzido iterativamente no apoio a análise, desenho e implementação do sistema para permitir que sejam tomadas decisões de arquitetura e para ajudar a definir as expectativas dos stakeholders.

Os seguintes princípios de testes de performance são muito importantes.

- Os testes devem estar alinhados às expectativas definidas dos diferentes grupos de stakeholders, em particular dos usuários, arquitetos de sistemas e da equipe de operações.
- Os testes devem ser reprodutíveis. Os resultados estatisticamente idênticos (dentro de uma tolerância específica) devem ser obtidos repetindo-se os testes em um sistema inalterado.
- Os testes devem produzir resultados que sejam compreensíveis e possam ser prontamente comparados com as expectativas dos stakeholders.

- Os testes podem ser realizados, onde os recursos permitem, em sistemas completos ou parciais ou em ambientes de teste que são similares ao sistema de produção.
- Os testes devem ser, de forma prática, acessíveis e executáveis dentro do prazo definido pelo projeto.

Os livros de [Molyneaux09] e [Microsoft07] fornecem uma base sólida para os princípios e aspectos práticos dos testes de performance.

Todas as três subcaracterísticas de qualidade referenciados acima afetarão na capacidade de dimensionamento do sistema em teste (SUT).

1.2 Tipos de Teste de Performance

Podem ser definidos diferentes tipos de testes de performance. Cada um pode ser aplicável a um determinado projeto, dependendo dos objetivos do teste.

Teste de Performance

O teste de performance é um termo abrangente que inclui qualquer tipo de teste focado na performance (responsivo) do sistema ou componente sob diferentes volumes de carga.

Teste de Carga

O teste de carga se concentra na capacidade de um sistema em lidar com níveis crescentes de cargas reais, de forma antecipada, resultantes de solicitações de transação geradas por quantidades de usuários ou processos controlados ou concorrentes.

Teste de Estresse

O teste de estresse se concentra na capacidade de um sistema ou componente em lidar com picos de cargas que estão no limite ou além dos limites das cargas de trabalho previstas ou especificadas. O teste de estresse também é usado para avaliar a capacidade de um sistema em lidar com a disponibilidade reduzida de recursos, como capacidade de processamento, largura de banda e memória disponíveis.

Teste de Escalabilidade

O teste de escalabilidade se concentra na capacidade de um sistema em atender a requisitos futuros de eficiência, que podem estar além dos requisitos exigidos atualmente. O objetivo desses testes é determinar a capacidade do sistema de expandir (p. ex., com mais usuários, grandes quantidades de dados armazenados) sem violar os requisitos de performance especificados no momento ou em sua falha. Uma vez que os limites de escalabilidade são conhecidos, os valores limites podem ser definidos e monitorados na produção para fornecer um alerta de problemas que podem estar prestes a acontecer. Além disso, o ambiente de produção pode ser ajustado com configurações adequadas de hardware.

Teste de Pico

O teste de pico foca na capacidade de um sistema em responder corretamente a rajadas súbitas de cargas de pico e retornar depois a um estado estável.

Teste de Resistência

O teste de resistência concentra-se na estabilidade do sistema ao longo de um período específico para o contexto operacional do sistema. Esse tipo de teste verifica se não há problemas de capacidade de recursos (p. ex., vazamentos de memória, conexões de banco de dados, conjuntos de encadeamentos) que podem, eventualmente, degradar a performance e/ou causar falhas nos pontos de interrupção.

Teste de Concorrência

O teste de concorrência foca no impacto das situações em que ações específicas ocorrem simultaneamente (p. ex., quando muitos usuários fazem login ao mesmo tempo). Os problemas de concorrência são notoriamente difíceis de encontrar e reproduzir, especialmente quando o problema ocorre em um ambiente em que o teste tem pouco ou nenhum controle, como na produção.

Teste de Capacidade

O teste de capacidade determina quantos usuários e/ou transações um sistema suportará atendendo aos objetivos declarados de performance. Estes objetivos podem também ser indicados em relação aos volumes de dados resultantes de transações.

1.3 Testando os tipos do Teste de Performance

Os principais tipos de testes usados nos testes de performance incluem testes estáticos e testes dinâmicos.

1.3.1 Teste Estático

As atividades de testes estáticos geralmente são mais importantes para os testes de performance do que para os testes de adequação funcional. Isso ocorre porque muitos defeitos críticos de performance são introduzidos na arquitetura e na modelagem do sistema. Esses defeitos podem ser introduzidos por mal-entendidos ou falta de conhecimento por parte dos analistas e arquitetos. Esses defeitos também podem ser introduzidos porque os requisitos não capturaram adequadamente o tempo de resposta, a taxa de transferência, as metas de utilização de recursos, a carga e o uso esperados do sistema ou restrições.

As atividades de testes estáticos para performance podem incluir:

- Revisões de requisitos com foco em aspectos e riscos de performance.
- Revisões de esquemas de banco de dados, diagramas de relacionamento de entidade, metadados, procedimentos armazenados e consultas.
- Revisões do sistema e arquitetura de rede.

- Revisões de segmentos críticos do código do sistema (p. ex., algoritmos complexos).

1.3.2 Teste Dinâmico

À medida que o sistema é construído, o teste de performance dinâmico deve começar o mais rápido possível.

Oportunidades para testes dinâmicos de performance incluem:

- Durante o teste de unidade, incluindo o uso de informações de perfil para determinar possíveis gargalos e análise dinâmica para avaliar a utilização de recursos.
- Durante o teste de integração de componentes, nos principais casos de uso e fluxos de trabalho, especialmente ao integrar diferentes recursos de caso de uso ou integrar-se à estrutura de *backbone* de um fluxo de trabalho.
- Durante o teste do sistema de comportamentos de ponta-a-ponta sob várias condições de carga.
- Durante o teste de integração do sistema, especialmente para fluxos de dados e fluxos de trabalho entre as principais interfaces dos sistemas. No teste de integração de sistemas, não é incomum que o usuário seja outro sistema ou máquina (p. ex., entradas das entradas de sensores e outros sistemas).
- Durante o teste de aceite, para criar confiança do usuário, cliente e operador, na performance adequada do sistema e para ajustar o sistema sob condições do mundo real (mas geralmente não encontrar defeitos de performance no sistema).

Em níveis de teste mais altos, como testes de sistema e testes de integração de sistemas, o uso de ambientes, dados e cargas realistas é crítico para resultados precisos (consulte o Capítulo 4). No Ágil e em outros ciclos de vida iterativos incrementais, as equipes devem incorporar testes de performance estáticos e dinâmicos nas iterações iniciais, em vez de aguardar as iterações finais para abordar os riscos de performance.

Se o hardware customizado ou novo fizer parte do sistema, os testes iniciais de performance dinâmico podem ser realizados usando simuladores. No entanto, é uma boa prática começar a testar o hardware real o mais rápido possível, pois os simuladores geralmente não capturam adequadamente as restrições de recursos e os comportamentos relacionados à performance.

1.3.3 O conceito de geração de carga

Para realizar os vários tipos de testes de performance descritos no capítulo 1.2, as cargas representativas do sistema devem ser modeladas, geradas e submetidas ao sistema em teste. As cargas são comparáveis às entradas de dados usadas para casos de teste funcionais, mas diferem nas seguintes formas principais:

- Uma carga de teste de performance deve representar muitas entradas do usuário, não apenas uma.
- Uma carga de teste de performance pode exigir hardware e ferramentas dedicados para geração.

- A geração de uma carga de teste de performance depende da ausência de quaisquer defeitos funcionais no sistema em teste que possam afetar a execução do teste.

A geração eficiente e confiável de uma carga específica é um fator-chave para o sucesso durante a execução de testes de performance. Existem diferentes opções para geração de carga.

Geração de carga através da interface do usuário

Essa pode ser uma abordagem adequada se apenas um pequeno número de usuários tiver que ser representado e se os números necessários de clientes de software estiverem disponíveis para inserir as entradas necessárias. Essa abordagem também pode ser usada em conjunto com ferramentas de execução de testes funcionais, mas pode rapidamente se tornar impraticável à medida que o número de usuários a serem simulados aumenta. A estabilidade da interface do usuário (UI) também representa uma dependência crítica. Mudanças frequentes podem afetar a repetibilidade dos testes de performance e podem afetar significativamente os custos de manutenção. Testes por meio da interface do usuário podem ser a abordagem mais representativa para testes de ponta a ponta.

Geração de carga usando grupos

Essa abordagem depende da disponibilidade de muitos testadores que representarão usuários reais. No teste de grupos, os testadores são organizados de forma que a carga desejada possa ser gerada. Este pode ser um método adequado para testar aplicações que podem ser acessadas de qualquer lugar do mundo (p. ex., baseado na web), e pode envolver os usuários gerando uma carga a partir de uma ampla gama de diferentes tipos de dispositivos e configurações. Embora essa abordagem possa permitir que um número muito grande de usuários seja utilizado, a carga gerada não será tão reprodutível e precisa quanto as outras opções e é mais complexa de organizar.

Geração de carga por meio de API (*Application Programming Interface*)

Essa abordagem é semelhante ao uso da interface do usuário para entrada de dados, mas usando uma API do aplicativo para simular a interação do usuário com o sistema em teste. A abordagem é, portanto, menos sensível a alterações na interface do usuário (p. ex., atrasos) e permite que as transações sejam processadas da mesma maneira como se fossem inseridas diretamente por um usuário através de uma interface. Podem ser criados scripts dedicados que repetidamente chamam rotinas API específicas e permitem que mais usuários sejam simulados em comparação com o uso de entradas da interface do usuário.

Geração de carga usando protocolos de comunicação capturados

Essa abordagem envolve a captura da interação do usuário com o sistema em teste no nível do protocolo de comunicação e a reprodução desses scripts para simular um número potencialmente grande de usuários de uma maneira repetível e confiável. Esta abordagem baseada em ferramentas é descrita nas Seções 4.2.6 e 4.2.7.

1.4 Modos comuns de falha de eficiência de performance e suas causas

Embora existam muitos e diferentes modos de falha de performance que podem ser encontrados durante os testes dinâmicos, são apresentados a seguir alguns exemplos mais comuns (incluindo falhas no sistema), juntamente com suas causas típicas:

Resposta lenta sob todos os níveis de carga

Em alguns casos, a resposta é inaceitável, independentemente da carga. Isso pode ser causado por problemas de performance subjacentes, incluindo, mas não se limitando a, modelagem ou implementação de banco de dados inválido, latência de rede e outras cargas de segundo plano. Esses problemas podem ser identificados durante testes funcionais e de usabilidade, não apenas em testes de performance, portanto, os analistas de teste devem ficar atentos para eles e relatá-los.

Resposta lenta sob níveis de carga moderada a pesada

Em alguns casos, a resposta degrada-se inaceitavelmente com carga moderada a pesada, mesmo quando essas cargas estão inteiramente dentro dos intervalos permitidos, normais e esperados. Defeitos subjacentes incluem a saturação de um ou mais recursos e variações de cargas pesadas.

Resposta degradada ao longo do tempo

Em alguns casos, a resposta degrada-se gradual ou severamente ao longo do tempo. Causas subjacentes incluem vazamentos de memória, fragmentação de disco, aumento da carga de rede ao longo do tempo, crescimento do repositório de arquivos e crescimento inesperado do banco de dados.

Tratamento inadequado de erros sob carga pesada ou acima do limite

Em alguns casos, o tempo de resposta é aceitável, mas o tratamento de erros diminui em níveis altos de carga e além do limite. Os defeitos subjacentes incluem pools de recursos insuficientes, filas e pilhas subdimensionadas e configurações de tempo limite muito rápidas.

Exemplos específicos dos tipos gerais de falhas listados acima incluem:

- Um aplicativo baseado na Web que fornece informações sobre os serviços de uma empresa não responde às solicitações do usuário em sete segundos (uma regra prática geral do setor). A eficiência de performance do sistema não pode ser alcançada sob condições de carga específicas.
- Um sistema trava ou não consegue responder a entradas do usuário quando submetido a um grande número súbito de solicitações do usuário (p. ex., vendas de ingressos para um grande evento esportivo). A capacidade do sistema para lidar com esse número de usuários é inadequada.
- A resposta do sistema é significativamente reduzida quando os usuários enviam solicitações para grandes quantidades de dados (p. ex., um relatório grande e importante é postado em

um site para download). A capacidade do sistema para lidar com os volumes de dados gerados é insuficiente.

- O processamento batch não pode ser concluído antes que o processamento on-line seja necessário. O tempo de execução do processamento batch é insuficiente para o período permitido.
- Um sistema em tempo real fica com memória RAM insuficiente quando os processos paralelos geram grandes demandas de memória dinâmica que não podem ser liberadas a tempo. A memória RAM não é dimensionada adequadamente ou as solicitações de memória RAM não são priorizadas adequadamente.
- Um componente do sistema em tempo real A, que fornece entradas para o componente B do sistema em tempo real, não consegue calcular as atualizações no tempo necessário. O sistema geral não responde a tempo e pode falhar. O código das funcionalidades do componente A devem ser avaliados e modificados (perfil de performance) para garantir que os tempos de respostas da atualização necessários possam ser alcançados.

2 Fundamentos da medição da performance [55 min]

Palavras-chave

medição, métrica

Objetivos de Aprendizagem

2.1 Métricas típicas coletadas no Teste de Performance

PTFL-2.1.1 (K2) Entender as métricas mais comuns coletadas nos Testes de Performance.

2.2 Agregação de resultados do Teste de Performance

PTFL-2.2.1 (K2) Explicar por que os resultados dos testes de performance são agregados.

2.3 Principais fontes de métricas de performance

PTFL-2.3.1 (K2) Compreender as principais fontes de métricas de performance.

2.4 Resultados comuns do Teste de Performance

PTFL-2.4.1 (K1) Recordar os resultados mais comuns dos testes de performance.

2.1 Métricas típicas coletadas no Teste de Performance

2.1.1 Por que as métricas de performance são necessárias?

Medições precisas e suas métricas derivadas são essenciais para definir os objetivos do teste de performance e para avaliar seus resultados. O teste de performance não deve ser realizado sem primeiro entender quais medidas e métricas serão necessárias. Os seguintes riscos do projeto se aplicam se este aviso for ignorado:

- Não se sabe se os níveis de performance serão aceitáveis para atender aos objetivos operacionais.
- Os requisitos de performance não são definidos em termos mensuráveis.
- Pode não ser possível identificar tendências que prevejam níveis mais baixos de performance.
- Os resultados reais de um teste de performance não podem ser avaliados comparando-os com um conjunto de medidas que definam se a performance é aceitável e/ou inaceitável.
- Os resultados dos testes de performance são avaliados com base na opinião subjetiva de uma ou mais pessoas.
- Os resultados fornecidos por uma ferramenta de teste de performance não serão compreendidos.

2.1.2 Coletando medições e métricas de performance

Como acontece com qualquer forma de medição, é possível obter e expressar métricas de maneiras precisas. Portanto, qualquer uma das métricas e medições descritas nesse capítulo pode e deve ser definida para ser significativa em um contexto particular. Esta é uma questão de realizar testes iniciais e aprender quais métricas precisam ser refinadas e quais precisam ser adicionadas.

Por exemplo, a métrica do tempo de resposta provavelmente estará em qualquer conjunto de métricas de performance. No entanto, para ser significativa e acionável, a métrica de tempo de resposta precisará ser definida em termos de hora do dia, número de usuários simultâneos, quantidade de dados sendo processados e assim por diante.

As métricas coletadas em um teste de performance específico variam de acordo com:

- O contexto de negócios (processos de negócios, comportamento de clientes e usuários e expectativas dos stakeholders).
- O contexto operacional (tecnologia e como é usado).
- Os objetivos de teste.

Por exemplo, as métricas selecionadas para o teste de performance de um site de comércio eletrônico internacional serão diferentes daquelas selecionadas para o teste de performance de um sistema embarcado utilizado para controlar a funcionalidade de um dispositivo médico.

Uma maneira comum de categorizar as medições e métricas de performance é considerar os ambientes técnico, de negócios ou o operacional em que a avaliação da performance é necessária.

As categorias de medições e métricas incluídas abaixo são comumente obtidas dos testes de performance.

Ambiente técnico

As métricas de performance variam de acordo com o tipo de ambiente técnico, conforme mostrado na lista a seguir:

- Baseado na Web.
- Móvel.
- Internet das coisas (IoT).
- Dispositivos clientes de desktop.
- Processamento do lado do servidor.
- Mainframe.
- Bancos de Dados.
- Redes.
- A natureza do software em execução no ambiente (p. ex., embarcado).

As métricas incluem o seguinte:

- Tempo de resposta (p. ex., por transação, por usuário simultâneo, tempos de carregamento da página).
- Utilização de recursos (p. ex., CPU, memória, largura de banda de rede, latência de rede, espaço em disco disponível, taxa de E/S, processamentos distribuídos ociosos e ocupados).
- Taxa de transferência da transação-chave (ou seja, o número de transações que podem ser processadas em um determinado período).
- Tempo de processamento Batch (p. ex., tempos de espera, tempos de processamento, tempos de resposta da base de dados, tempos de conclusão).
- Números de erros que afetam a performance.
- Tempo de conclusão (p. ex., para criar, ler, atualizar e excluir dados).
- Carregamento em segundo plano em recursos compartilhados (especialmente em ambientes virtualizados).
- Métricas de software (p. ex., complexidade de código).

Ambiente de negócios

Do ponto de vista comercial ou funcional, as métricas de performance podem incluir o seguinte:

- Eficiência do processo de negócios (p. ex., a velocidade de execução de um processo geral de negócios, incluindo fluxos de casos de uso normais, alternativos e excepcionais).
- Taxa de transferência de dados, transações e outras unidades de trabalho executadas (p. ex., pedidos processados por hora, linhas de dados adicionadas por minuto).
- Cumprimento de *Service Level Agreement* (SLA) ou taxas de violação (p. ex., violações de SLA por unidade de tempo).
- Escopo de uso (p. ex., porcentagem de usuários globais ou nacionais que realizam tarefas em um determinado momento).
- Concorrência de uso (p. ex., o número de usuários que executam simultaneamente uma tarefa).
- Tempo de uso (p. ex., o número de pedidos processados durante os horários de pico de carga).

Ambiente operacional

O aspecto operacional do teste de performance se concentra em tarefas que geralmente não são consideradas voltadas para o usuário por natureza. Esses incluem:

- Processos operacionais (p. ex., o tempo necessário para a inicialização do ambiente, backups, desligamento e reinício).
- Restauração do sistema (p. ex., o tempo necessário para restaurar dados de um backup).
- Alertas e avisos (p. ex., o tempo necessário para o sistema emitir um alerta ou aviso).

2.1.3 Selecionando métricas de Performance

Deve-se notar que coletar mais métricas do que o necessário não é necessariamente a coisa certa. Cada métrica escolhida requer um meio de coleta e geração de relatórios consistentes. É importante definir o conjunto de métricas que suporte os objetivos do teste de performance.

Por exemplo, a abordagem *Goal-Question-Metric* (GQM) é uma maneira útil para alinhar as métricas com metas de performance. A ideia é, primeiro estabelecer os objetivos, depois fazer perguntas para saber quando os objetivos foram alcançados. As métricas estão associadas a cada pergunta para garantir que a resposta possa ser mensurável (ver capítulo 4.3 do *Expert Level Syllabys - Improving the Testing Process* [ISTQB_ELTM_ITP_SYL] para obter uma descrição mais completa da abordagem do GQM). Deve-se observar que a abordagem do GQM nem sempre se encaixa no processo de teste de performance. Por exemplo, algumas métricas representam a integridade de um sistema e não estão diretamente vinculadas a metas.

É importante perceber que após a definição e a captura das medidas iniciais, podem ser necessárias medidas e métricas adicionais para entender os verdadeiros níveis de performance e determinar onde as ações corretivas podem ser necessárias.

2.2 Agregação dos resultados do Teste de Performance

O objetivo de agregar métricas de performance é poder compreendê-las e expressá-las de uma maneira que transmita com precisão a imagem total da performance do sistema. Quando as métricas de performance são vistas apenas no nível detalhado, chegar à conclusão correta pode ser difícil, especialmente para os stakeholder no negócio.

Para muitos stakeholders, a principal preocupação é que o tempo de resposta de um sistema, site, ou outro objeto de teste esteja dentro dos limites aceitáveis.

Uma vez que a compreensão mais profunda das métricas de performance foi alcançada, as métricas podem ser agregadas para que:

- Os stakeholders de negócio e de projeto possam ter uma visão mais abrangente da performance do sistema.
- Possam identificar as tendências na performance.
- As métricas de performance sejam relatadas de uma maneira compreensível.

2.3 Principais fontes de métricas de performance

A performance do sistema não deve ser mais que minimamente afetada pelo esforço da coleta de métricas (conhecido como o efeito de teste). Além disso, o volume, a precisão e a velocidade com que as métricas de performance sejam coletadas tornam o uso da ferramenta um requisito. Embora o uso combinado de ferramentas não seja incomum, elas podem introduzir redundância no uso das ferramentas de teste e outros problemas (ver capítulo 4.4).

Existem três fontes principais de métricas de performance:

Ferramentas de teste de performance

Todas as ferramentas de teste de performance fornecem medições e métricas como resultado de um teste. As ferramentas podem variar no número de métricas exibidas, na maneira como são mostradas e na capacidade do usuário de personalizá-las para uma situação específica (ver capítulo 5.1).

Algumas ferramentas coletam e exibem métricas de performance no formato de texto, enquanto ferramentas mais robustas coletam e exibem graficamente as métricas de performance em um formato de dashboard. Muitas ferramentas oferecem a capacidade de exportar as métricas para facilitar a avaliação e os relatórios de testes.

Ferramentas de monitoramento de performance

As ferramentas de monitoramento de performance são frequentemente empregadas para complementar os recursos do relatório das ferramentas de teste de performance (ver capítulo 5.1). Além disso, as ferramentas de monitoramento podem ser usadas para monitorar continuamente a performance do sistema e para alertar os administradores de sistemas sobre os níveis reduzidos de performance e os níveis mais elevados de erros e alertas do sistema. Essas ferramentas também podem ser usadas para detectar e notificar nos casos de comportamento suspeito (como ataques de *Denial of Service* (DoS) e *Distributed Denial of Service* (DDoS).

Ferramentas de análise de log

Existem ferramentas que examinam e compilam as métricas a partir dos registros do servidor. Algumas dessas ferramentas podem criar mapas para fornecer uma visualização gráfica dos dados.

Os erros, alertas e avisos são normalmente registrados nos logs do servidor. Esses incluem:

- Alto uso de recursos, como: utilização da CPU, níveis de armazenamento em disco consumidos e largura de banda insuficiente.
- Erros de memória e avisos, como exaustão de memória.
- Problemas de deadlocks e multi-threading, especialmente ao executar operações de banco de dados.
- Erros de banco de dados, como exceções de SQL e tempos limite de SQL.

2.4 Resultados comuns de um teste de performance

Em testes funcionais, particularmente ao verificar requisitos funcionais especificados ou elementos funcionais de histórias de usuários, os resultados esperados geralmente podem ser claramente definidos e os resultados dos testes interpretados para determinar se o teste foi aprovado ou não. Por exemplo, um relatório mensal de vendas mostra um total correto ou incorreto.

Enquanto os testes que verificam a adequação funcional geralmente se beneficiam de roteiros de teste bem definidos, o teste de performance muitas vezes carece dessa fonte de informação. Além dos stakeholders serem notoriamente ruins em articular os requisitos de performance, muitos analistas de negócios e Product Owners são ruins em extrair tais requisitos. Os testadores geralmente recebem orientação limitada para definir os resultados esperados do teste.

Ao avaliar os resultados do teste de performance, é importante observar os resultados de perto. Os resultados brutos iniciais podem ser enganosos, com falhas de performance escondidas sob resultados gerais aparentemente bons. Por exemplo, a utilização de recursos pode estar bem abaixo de 75% para todos os principais recursos potenciais de gargalos, mas o tempo de resposta ou o rendimento das principais transações ou casos de uso é uma ordem de magnitude muito lenta.

Os resultados específicos a serem avaliados variam de acordo com os testes executados e geralmente incluem os discutidos no capítulo 2.1.

3 O teste de performance no ciclo de vida do software [195 min]

Palavras-chave

métrica, risco, ciclo de vida de desenvolvimento de software, log de teste, registro de teste

Objetivos de Aprendizado

3.1 Principais atividades no teste de performance

PTFL-3.1.1 (K2) Compreender as principais atividades no Teste de Performance.

3.2 Categorias de riscos de performance para diferentes arquiteturas

PTFL-3.2.1 (K2) Explicar as categorias mais comuns de riscos de performance para diferentes arquiteturas.

3.3 Riscos de performance em todo o ciclo de vida de desenvolvimento de software

PTFL-3.3.1 (K4) Analisar os riscos de performance de um determinado produto em todo o ciclo de vida de desenvolvimento de software.

3.4 Atividades no teste de performance

PTFL-3.4.1 (K4) Analisar um projeto qualquer para determinar as atividades de teste de performance apropriadas para cada fase do ciclo de vida de desenvolvimento de software.

3.1 Principais atividades no teste de performance

O teste de performance é de natureza iterativa. Cada teste fornece informações valiosas sobre a performance do aplicativo e do sistema. As informações coletadas de um teste são usadas para corrigir ou otimizar os parâmetros do aplicativo e do sistema. A próxima iteração de teste mostrará os resultados das modificações e assim por diante até que os objetivos do teste sejam atingidos.

As atividades de teste de performance se alinham com o processo de teste do ISTQB [ISTQB_FL_SYL].

Planejamento de teste

O planejamento de teste é particularmente importante para testes de performance devido à necessidade de alocação de ambientes de teste, dados de teste, ferramentas e recursos humanos. Além disso, essa é a atividade na qual o escopo do teste de performance é estabelecido.

Durante o planejamento do teste, as atividades de identificação de risco e análise de risco são concluídas e as informações relevantes são atualizadas em qualquer documentação de planejamento de teste (p. ex., plano de teste, plano de teste de nível). Assim como o planejamento de testes é revisitado e modificado conforme necessário, os riscos, os níveis de risco e o status de risco são modificados para refletir as mudanças nas condições de risco.

Monitoramento e controle dos testes

As medidas de controle são definidas para fornecer planos de ação, caso sejam encontrados problemas que possam afetar a eficiência da performance, como:

- O aumento da capacidade de geração de carga se a infraestrutura não gerar as cargas desejadas, conforme planejado para testes de performance específicos.
- O hardware modificado, novo ou substituído.
- As alterações nos componentes de rede.
- As mudanças na implementação de software.

Os objetivos do teste de performance são avaliados para verificar a realização dos critérios de saída.

Análise de teste

Os testes de performance efetivo são baseados em uma análise de requisitos de performance, objetivos de teste, *Service Level Agreement* (SLA), arquitetura de TI, modelos de processo e outros itens que compõem a base de teste. Essa atividade pode ser suportada pela modelagem e análise de requisitos de recursos do sistema e/ou comportamento usando planilhas ou ferramentas de planejamento de capacidade.

São identificadas as condições de teste específicas, como níveis de carga, condições de tempo e transações a serem testadas. Então, são decididos os tipos de teste de performance necessários (p. ex., carga, estresse, escalabilidade).

Modelagem de teste

São projetados os casos de teste de performance. Estes são geralmente criados de forma modular para que possam ser usados como blocos de construção de testes de performance maiores e mais complexos (ver capítulo 4.2).

Implementação de teste

Na fase de implementação, os casos de teste de performance são ordenados nos procedimentos de teste de performance. Esses procedimentos devem refletir as etapas normalmente tomadas pelo usuário e outras atividades funcionais que devem ser cobertas durante o teste de performance.

Uma atividade de implementação de teste é estabelecer e/ou redefinir o ambiente de teste antes de cada execução de teste. Como o teste de performance é tipicamente orientado por dados, é necessário um processo para estabelecer dados de teste que sejam representativos dos dados reais de produção em volume e tipo, de modo que o uso da produção possa ser simulado.

Execução de teste

A execução do teste ocorre quando o teste de performance é realizado, geralmente usando ferramentas de teste de performance. Os resultados dos testes são avaliados para determinar se a performance do sistema atende aos requisitos e outros objetivos declarados. Quaisquer defeitos são relatados.

Conclusão dos testes

Os resultados dos testes de performance são fornecidos aos stakeholders (p. ex., arquitetos, gerentes, Product Owners) em um relatório de resumo de teste. Os resultados são expressos através de métricas que são frequentemente agregadas para simplificar o significado dos resultados do teste. Os meios visuais de geração de relatórios, como *dashboards*, costumam ser usados para expressar os resultados dos testes de performance de maneiras mais fáceis de entender do que as métricas baseadas em texto.

O teste de performance é geralmente considerado uma atividade contínua, pois é realizado sucessivas vezes e em todos os níveis de teste (componente, integração, sistema, integração do sistema e teste de aceite). No encerramento de um período definido de testes de performance, um ponto de encerramento do teste pode ser alcançado onde testes projetados, recursos de ferramentas de teste (casos de teste e procedimentos de teste), dados de teste e outros *testwares* são arquivados ou passados para outros testadores para uso posterior durante as atividades de manutenção do sistema.

3.2 Categorias de riscos de performance para diferentes arquiteturas

Como mencionado anteriormente, a performance do aplicativo ou do sistema varia consideravelmente com base na arquitetura, no aplicativo e no ambiente de hospedagem. Embora

não seja possível fornecer uma lista completa dos riscos de performance para todos os sistemas, a lista abaixo inclui alguns tipos típicos de riscos associados em arquiteturas específicas:

Sistemas de computadores únicos

Estes são sistemas ou aplicativos que são executados inteiramente em um computador não virtualizado.

A performance poderá degradar devido:

- Ao consumo excessivo de recursos, incluindo vazamentos de memória, atividades em segundo plano, como software de segurança, subsistemas de armazenamento lentos (p. ex., dispositivos externos de baixa velocidade ou fragmentação de disco) e mal gerenciamento do sistema operacional.
- Na implementação ineficiente de algoritmos que não fazem uso dos recursos disponíveis (p. ex., memória principal) e, como resultado, executam mais lentamente do que o necessário.

Sistemas multicamadas

Esses são sistemas de sistemas executados em vários servidores, cada um executando um conjunto específico de tarefas, como servidor de banco de dados, servidor de aplicativos e servidor de apresentação. Cada servidor é, obviamente, um computador e está sujeito aos riscos dados anteriormente. Além disso, a performance poderá degradar devido ao projeto de banco de dados ruim ou não escalável, aos gargalos de rede e largura de banda ou a capacidade inadequada de qualquer servidor único.

Sistemas distribuídos

Estes são sistemas de sistemas, semelhantes a uma arquitetura multicamadas, mas os vários servidores podem mudar dinamicamente, como um sistema de comércio eletrônico que acessa diferentes bancos de dados de estoque, dependendo da localização geográfica da pessoa que faz o pedido. Além dos riscos associados a arquiteturas multicamadas, essa arquitetura pode enfrentar problemas de performance devido a fluxos de trabalho ou fluxos de dados críticos através de servidores remotos imprevisíveis ou não confiáveis, especialmente quando esses servidores sofrem problemas de conexão periódica ou períodos intermitentes de carga intensa.

Sistemas virtualizados

Estes são sistemas em que o hardware físico hospeda vários computadores virtuais. Essas máquinas virtuais podem hospedar sistemas e aplicativos de um único computador, bem como servidores que fazem parte de uma arquitetura de várias camadas ou distribuída. Os riscos de performance que surgem especificamente da virtualização incluem carga excessiva no hardware em todas as máquinas virtuais ou configuração inadequada da máquina virtual do host, resultando em recursos inadequados.

Sistemas dinâmicos/baseados em nuvem

Estes são sistemas que oferecem a capacidade de escalar sob demanda, aumentando a capacidade conforme o aumento do nível da carga. Esses sistemas são tipicamente distribuídos e sistemas

multicamadas virtualizados, embora com recursos de dimensionamento automático projetados especificamente para mitigar alguns dos riscos de performance associados a essas arquiteturas. No entanto, há riscos associados a falhas na configuração adequada desses recursos durante a configuração inicial ou atualizações subsequentes.

Sistemas cliente-servidor

Estes são sistemas em execução em um cliente que se comunicam por meio de uma interface com um único servidor, servidor de várias camadas ou servidor distribuído. Como há código em execução no cliente, os riscos de um único computador se aplicam a esse código, enquanto os problemas do lado do servidor mencionados acima também se aplicam. Além disso, existem riscos de performance devido a problemas de velocidade e confiabilidade da conexão, congestionamento de rede no ponto de conexão do cliente (p. ex., Wi-Fi público) e possíveis problemas devido a firewalls, inspeção de pacotes e balanceamento de carga do servidor.

Aplicativos móveis

São aplicativos executados em um smartphone, tablet ou outro dispositivo móvel. Tais aplicativos estão sujeitos aos riscos mencionados para aplicativos cliente-servidor e baseados em navegador (aplicativos da web). Além disso, os problemas de performance podem surgir devido aos recursos e conectividades limitados e variáveis disponíveis no dispositivo móvel (que podem ser afetados pelo local, duração da bateria, estado da carga, memória disponível, e temperatura). Para aqueles aplicativos que usam sensores, acelerômetros ou *bluetooth*, os fluxos dos dados lentos originários dessas fontes podem gerar problemas. Por fim, os aplicativos móveis geralmente têm interações pesadas com outros aplicativos móveis locais e serviços remotos da Web, e qualquer um deles pode se tornar um gargalo de eficiência da performance.

Sistemas embarcados em tempo real

Estes são sistemas que funcionam dentro ou até mesmo controlam coisas cotidianas, como carros (p. ex., sistemas de entretenimento e sistemas inteligentes de frenagem), elevadores, sinais de trânsito, sistemas de aquecimento, ventilação e ar-condicionado (HVAC) e muito mais. Esses sistemas geralmente apresentam muitos dos riscos dos dispositivos móveis, incluindo (cada vez mais) problemas relacionados à conectividade, uma vez que esses dispositivos estão conectados à *internet*. No entanto, a diminuição da performance de um videogame móvel geralmente não representa um risco à segurança do usuário, embora essa lentidão em um sistema de frenagem de veículo possa ser catastrófica.

Aplicativos de mainframe

São aplicativos - em muitos casos, aplicativos de décadas - que suportam, muitas vezes, funções de negócios de missão crítica em um data center, às vezes por meio de processamento *batch*. A maioria é bastante previsível e rápida quando usada originalmente, mas muitas delas agora podem ser acessadas via APIs, serviços da Web ou por meio de seu banco de dados, o que pode resultar em cargas inesperadas que afetam a taxa de transferência de aplicativos estabelecidos.

Observe que qualquer aplicativo ou sistema específico pode incorporar duas ou mais das arquiteturas listadas acima, o que significa que todos os riscos relevantes serão aplicados a esse

aplicativo ou sistema. Na verdade, considerando a Internet das Coisas (IoT) e a difusão de aplicativos móveis - duas áreas em que níveis extremos de interação e conexão são a regra - é possível que todas as arquiteturas estejam presentes de alguma forma em um aplicativo e, portanto, todos os riscos devem ser aplicados.

Embora a arquitetura seja claramente uma decisão técnica importante com um impacto profundo nos riscos de performance, outras decisões técnicas também influenciam e criam riscos. Por exemplo, os vazamentos de memória são mais comuns em linguagens que permitem o gerenciamento direto de memória de *heap*, como C e C ++, e os problemas de performance são diferentes para bancos de dados relacionais versus não relacionais. Tais decisões se estendem até a modelagem de funções ou métodos individuais (p. ex., a escolha de um algoritmo recursivo em oposição a um algoritmo iterativo). Como testador, a capacidade de conhecer ou influenciar essas decisões varia, dependendo das funções e responsabilidades dos testadores dentro da organização e do ciclo de vida de desenvolvimento de software.

3.3 Riscos de performance em todo o ciclo de vida de desenvolvimento de software

O processo de análise de riscos para a qualidade de um produto de software em geral é discutido em vários programas do ISTQB (p. ex., ver [ISTQB_FL_SYL] e [ISTQB_ALTM_SYL]). Também é possível encontrar discussões sobre riscos e considerações específicas associadas às características específicas de qualidade (p. ex., [ISTQB_UT_SYL]) e de uma perspectiva comercial ou técnica (p. ex., ver [ISTQB_ALTA_SYL] e [ISTQB_ALTTA_SYL], respectivamente). Nesse capítulo, o foco está nos riscos relacionados à performance para a qualidade do produto, incluindo as formas de como o processo, os participantes e as considerações mudam.

Para riscos relacionados a performance para a qualidade do produto, o processo é:

1. Identificar os riscos para a qualidade do produto, concentrando-se em características como comportamento no tempo, utilização de recursos e capacidade.
2. Avaliar os riscos identificados, assegurando que as categorias de arquitetura relevantes (ver capítulo 3.2) sejam abordadas. Avaliar o nível geral de cada risco, identificado em termos de probabilidade e impacto, e usando critérios claramente definidos.
3. Tomar as medidas apropriadas de mitigação de risco para cada item com base na sua natureza e no nível de critérios de risco.
4. Gerenciar os riscos em uma base contínua para garantir que sejam adequadamente mitigados antes do lançamento.

Tal como acontece com a análise de risco de qualidade em geral, os participantes neste processo devem incluir os *stakeholders* comerciais e técnicos. Para a análise de risco relacionada a performance, os participantes do negócio devem incluir aqueles com uma percepção específica de como os problemas de performance na produção afetarão realmente os clientes, os usuários, a empresa e outros stakeholders. Os participantes do negócio devem avaliar que o uso pretendido, negócios, societário ou de segurança crítica, possíveis danos financeiros e de reputação, responsabilidade legal civil ou criminal e fatores similares afetam o risco de uma perspectiva de negócios, criando riscos e influenciando o impacto das falhas.

Além disso, os *stakeholders* técnicos devem incluir aqueles com um profundo entendimento das implicações de performance de requisitos relevantes, arquitetura, modelagem e decisões de implementação. Os *stakeholders* técnicos devem entender que as decisões de arquitetura, modelagem e implementação afetam os riscos de performance de uma perspectiva técnica, criando riscos e influenciando a probabilidade de defeitos.

O processo de análise de risco específico escolhido deve ter o nível apropriado de formalidade e rigor. Para os riscos relacionados à performance, é especialmente importante que o processo de análise de riscos seja iniciado com antecedência e seja repetido regularmente. Em outras palavras, o testador deve evitar confiar inteiramente nos testes de performance realizados no final do nível de teste do sistema e no nível de teste de integração do sistema. Muitos projetos, especialmente sistemas maiores e mais complexos de projetos de sistemas, tiveram surpresas desagradáveis devido à descoberta tardia de defeitos de performance que resultaram de decisões de requisitos, modelagem, arquitetura e implementação feitas no início do projeto. Portanto, a ênfase deve ser em uma abordagem iterativa para identificação, avaliação, mitigação e gerenciamento dos riscos de performance durante todo o ciclo de vida de desenvolvimento de software.

Por exemplo, se grandes volumes de dados forem manipulados por meio de um banco de dados relacional, a performance lenta de junções muitos-para-muitos devido a uma modelagem de banco de dados ruim só será revelada durante o teste dinâmico com um conjunto de dados de teste em grande escala, como os usados durante os testes de sistema. No entanto, uma revisão técnica cuidadosa que inclua analistas de banco de dados experientes pode-se prever os problemas antes da implementação do banco de dados. Após essa revisão, em uma abordagem iterativa, os riscos são identificados e avaliados novamente.

Além disso, a mitigação e o gerenciamento dos riscos devem abranger e influenciar todo o processo de desenvolvimento de software, não apenas o teste dinâmico. Por exemplo, quando decisões críticas relacionadas à performance, como o número esperado de transações ou usuários simultâneos, não podem ser especificadas no início do projeto, é importante que as decisões de arquitetura e modelagem permitam uma escalabilidade altamente variável (p. ex., recursos de computação baseados em nuvem sob demanda). Isso permite a antecipação da tomada de decisão sobre a mitigação dos riscos.

Um bom analista de performance pode ajudar as equipes do projeto a evitar tardiamente a descoberta de defeitos críticos de performance durante os níveis mais altos de teste, como no teste de integração do sistema ou no teste de aceite do usuário. Os defeitos de performance encontrados em um estágio final do projeto podem ser extremamente caros e podem até levar ao cancelamento de projetos inteiros.

Como com qualquer tipo de risco de qualidade, os riscos relacionados a performance nunca podem ser evitados completamente, ou seja, algum risco de falha de produção relacionada a performance sempre existirá. Portanto, o processo de gerenciamento de riscos deve incluir o fornecimento de uma avaliação realista e específica do nível residual de risco para os stakeholders do negócio e técnicas envolvidas no processo. Por exemplo, dizer simplesmente: “Sim, ainda é possível que os clientes sofram longos atrasos durante o *check-out*” não é útil, pois não dá ideia de qual quantidade de mitigação de risco ocorreu ou do nível de risco que permanece. Em vez disso, fornecer

informações claras sobre a porcentagem de clientes que provavelmente sofrerão atrasos iguais ou superiores a determinados limites ajudará as pessoas a entender o status.

3.4 Atividades no teste de performance

As atividades no teste de performance serão organizadas e executadas de forma diferente, dependendo do tipo de ciclo de vida de desenvolvimento de software em uso.

Modelos de desenvolvimento sequencial

A prática ideal dos testes de performance em modelos de desenvolvimento sequencial é incluir os critérios de performance como parte dos critérios de aceite definidos no início do projeto. Reforçando a visão do ciclo de vida dos testes, as atividades de teste de performance devem ser conduzidas durante todo o ciclo de vida de desenvolvimento do software. À medida que o projeto avança, cada atividade de teste de performance subsequente deve ser baseada nos itens definidos nas atividades anteriores, conforme mostrado abaixo.

- **Conceito**: verificar se as metas de performance do sistema estão definidas como critérios de aceite para o projeto.
- **Requisitos**: verificar se os requisitos de performance estão definidos e representam corretamente as necessidades dos stakeholders.
- **Análise e modelagem**: verificar se a modelagem do sistema reflete os requisitos de performance.
- **Codificação/implementação**: verificar se o código é eficiente e reflete os requisitos e a modelagem em termos de performance.
- **Teste de componentes**: conduzir os testes de performance no nível do componente.
- **Teste de integração de componentes**: conduzir os testes de performance no nível da integração de componentes.
- **Teste de sistema**: conduzir os testes de performance no nível do sistema, que incluem hardware, software, procedimentos e dados representativos do ambiente de produção. As interfaces do sistema podem ser simuladas desde que forneçam uma representação real da performance.
- **Teste de integração de sistema**: realizar os testes de performance com todo o sistema, que é representativo do ambiente de produção.
- **Teste de aceite**: validar se a performance do sistema atende às necessidades do usuário e aos critérios de aceite especificados originalmente.

Modelos iterativos e incrementais de desenvolvimento

Nesses modelos de desenvolvimento, como o Ágil, o teste de performance também é visto como uma atividade iterativa e incremental (consulte [ISTQB_FL_AT]). O teste de performance pode ocorrer como parte da primeira iteração ou como uma iteração dedicada inteiramente ao teste de performance. No entanto, com esses modelos de ciclo de vida, a execução desse tipo de teste pode ser executada por uma equipe separada.

A Integração Contínua (IC) é comumente executada em ciclos de vida de desenvolvimento de software iterativo e incremental, o que facilita uma execução altamente automatizada dos testes. O

objetivo mais comum do teste na IC é realizar os testes de regressão e garantir que cada construção seja estável. O teste de performance pode fazer parte dos testes automatizados realizados na IC, se forem projetados de modo a serem executados no nível de construção. No entanto, ao contrário dos testes automatizados funcionais, existem outras preocupações, como as seguintes:

- *A configuração do ambiente de teste*: isso geralmente requer um ambiente de teste disponível sob demanda, como um ambiente de teste de performance baseado em nuvem.
- *Determinar quais testes de performance sejam automatizados na IC*: devido ao curto período disponível para os testes de IC, os testes de performance podem ser um subconjunto de testes de mesmo tipo mais extensivos, que são realizados por uma equipe de especialistas em outros momentos durante uma iteração.
- *Criar os testes de performance para a IC*: o principal objetivo destes testes como parte da IC é garantir que uma alteração não tenha impacto negativo na performance. Dependendo das alterações feitas para qualquer compilação, novos testes podem ser necessários.
- *Execução de testes de performance em partes de um aplicativo ou sistema*: isso geralmente exige que as ferramentas e os ambientes de teste sejam capazes de realizar testes rápidos, incluindo a capacidade de selecionar os subconjuntos aplicáveis de testes.

O teste de performance nos ciclos de vida de desenvolvimento de software iterativo e incremental também podem ter suas próprias atividades de ciclo de vida:

- **Planejamento do release**: nesta atividade, o teste de performance é considerado da perspectiva de todas as iterações em um release, da primeira iteração até a iteração final. Os riscos são identificados e avaliados e as medidas de mitigação são planejadas. Isso geralmente inclui o planejamento de qualquer teste de performance final antes do lançamento do aplicativo.
- **Planejamento de iteração**: no contexto de cada iteração, o teste de performance pode ser executado dentro da iteração e a cada uma concluída. Os riscos de performance são avaliados em mais detalhes para cada história de usuário.
- **Criação de história de usuário**: as histórias de usuário geralmente formam a base dos requisitos de performance nas metodologias Ágeis, com os critérios de performance específicos descritos nos critérios associados de aceite. Eles são chamados de histórias de usuários não funcionais.
- **Modelagem de testes de performance**: os requisitos de performance e critérios descritos em histórias específicas de usuários são usados no projeto de testes (ver capítulo 4.2)
- **Codificação/implementação**: durante a codificação, o teste de performance pode ser realizado no nível do componente. Um exemplo disso seria o ajuste de algoritmos para uma ótima eficiência de performance.
- **Teste/avaliação**: embora o teste seja normalmente realizado próximo às atividades de desenvolvimento, o teste de performance pode ser realizado como uma atividade separada, dependendo do escopo e dos objetivos do teste de performance durante a iteração. Por exemplo, se o objetivo do teste de performance for testar a performance da iteração como um conjunto completo de histórias de usuário, será necessário um escopo mais amplo de testes de performance do que o observado no teste de performance de uma única história de usuário. Isso pode ser agendado em uma iteração dedicada para testes de performance.

- **Entrega**: Como a entrega introduzirá o aplicativo no ambiente de produção, a performance precisará ser monitorada para determinar se o aplicativo atinge os níveis desejados de performance em uso real.

Software comercial de prateleira (COTS) e outros modelos de fornecedores

Muitas organizações não desenvolvem aplicativos e sistemas, mas estão em posição de adquirir software de fontes de fornecedores ou de projetos de código aberto. Em tais modelos de fornecedor/comprador, a performance é uma consideração importante que requer testes das perspectivas do fornecedor (fornecedor/desenvolvedor) e do comprador (cliente).

Independentemente da origem do aplicativo, geralmente é responsabilidade do cliente validar se a performance atende aos requisitos. No caso do software personalizado desenvolvido por um fornecedor, dos requisitos de performance e dos critérios de aceite associados que devem ser especificados como parte do contrato entre o fornecedor e o cliente. No caso de aplicativos COTS, o cliente tem a responsabilidade exclusiva de testar a performance do produto em um ambiente de teste realista antes da implantação.

4 Tarefas de teste de performance [475 min]

Palavras-chave

simultaneidade, perfil de carga, geração de carga, perfil operacional, aceleração, desaceleração, sistema de sistemas, taxa de transferência do sistema, plano de teste, *think time*, usuário virtual

Objetivos de Aprendizado

4.1 Planejamento

PTFL-4.1.1 (K4) Derivar os objetivos do Teste de Performance a partir de informações relevantes.

PTFL-4.1.2 (K4) Delinear um Plano de Teste de Performance que considere os objetivos de performance para um determinado projeto.

PTFL-4.1.3 (K4) Criar uma apresentação que permita que vários stakeholders compreendam a lógica por trás do teste planejado.

4.2 Análise, modelagem e implementação

PTFL-4.2.1 (K2) Dê exemplos de protocolos comuns encontrados em testes de performance.

PTFL-4.2.2 (K2) Compreender o conceito de transações nos testes de performance.

PTFL-4.2.3 (K4) Analisar os perfis operacionais para uso do sistema.

PTFL-4.2.4 (K4) Criar os perfis de carga derivando-os dos perfis operacionais para determinados objetivos de performance.

PTFL-4.2.5 (K4) Analisar a taxa de transferência e a simultaneidade ao desenvolver testes de performance.

PTFL-4.2.6 (K2) Compreender a estrutura básica de um script de teste de performance.

PTFL-4.2.7 (K3) Implementar os scripts de teste de performance consistentes com os perfis do plano de carga.

PTFL-4.2.8 (K2) Compreender as atividades envolvidas na preparação para execução de testes de performance.

4.3 Execução

PTFL-4.3.1 (K2) Compreender as atividades principais na execução dos scripts de teste de performance.

4.4 Analisando os resultados e relatórios

PTFL-4.4.1 (K4) Analisar e relatar os resultados dos testes de performance e suas implicações.

4.1 Planejamento

4.1.1 Derivando os objetivos do teste de performance

Os stakeholders podem incluir usuários e pessoas com experiência comercial ou técnica. Eles podem ter objetivos diferentes relacionados ao teste de performance. Os stakeholder definem os objetivos, a terminologia a ser usada e os critérios para determinar se o objetivo foi alcançado.

Os objetivos dos testes de performance estão relacionados a esses diferentes tipos dos stakeholders. Uma boa prática é distinguir entre objetivos técnicos e de usuários. Os objetivos baseados no usuário se concentram principalmente na satisfação do usuário final e nos objetivos de negócio. Geralmente, os usuários estão menos preocupados com os tipos de recursos ou com a forma como um produto é entregue. Eles só querem ser capazes de fazer o que precisam fazer.

Os objetivos técnicos, por outro lado, concentram-se nos aspectos operacionais e fornecem respostas a perguntas relacionadas à capacidade de escalonamento de um sistema ou sob quais condições a degradação da performance pode se tornar aparente.

Os principais objetivos do teste de performance incluem a identificação de riscos potenciais, a descoberta de oportunidades de melhoria e a identificação das mudanças necessárias.

Ao coletar informações dos vários stakeholders, as seguintes perguntas devem ser respondidas:

- Quais transações serão executadas no teste de performance e qual tempo médio de resposta é esperado?
- Quais métricas do sistema devem ser capturadas (p. ex., uso de memória, taxa de transferência da rede) e quais valores são esperados?
- Quais melhorias de performance são esperadas desses testes em comparação com ciclos de testes anteriores?

4.1.2 O Plano de Teste de Performance

O Plano de Teste de Performance (PTP) é um documento criado antes da ocorrência de qualquer teste de performance. O PTP deve ser referido no Plano de Teste (consulte [ISTQB_FL_SYL]), que também incluirá informações relevantes de programação. O PTP continua a ser atualizado assim que o teste de performance é iniciado.

As seguintes informações devem ser fornecidas em um PTP:

Objetivo

O objetivo do PTP descreve os objetivos, estratégias e métodos para o teste de performance. Permite uma resposta quantificável à questão central da adequação e da prontidão do sistema para executar sob uma carga.

Objetivos de teste

Os objetivos gerais do teste de performance a serem alcançados pelo Sistema em Teste (SUT) são listados para cada tipo de *stakeholder* (ver capítulo 4.1.1)

Visão geral do sistema

Uma breve descrição do SUT fornecerá o contexto para a medição dos parâmetros do teste de performance. A visão geral deve incluir uma descrição de alto nível da funcionalidade que está sendo testada sob carga.

Tipos de testes de performance a serem realizados

Os tipos de testes de performance a serem realizados são listados (ver capítulo 1.2), juntamente com uma descrição da finalidade de cada tipo.

Critérios de aceite

O teste de performance destina-se a determinar a capacidade de resposta, rendimento, confiabilidade e/ou escalabilidade do sistema sob uma determinada carga de trabalho. Em geral, o tempo de resposta é uma preocupação do usuário, a taxa de transferência é uma preocupação comercial e a utilização de recursos é uma preocupação do sistema. Os critérios de aceite devem ser definidos para todas as medidas relevantes e relacionados ao seguinte, conforme aplicável:

- Objetivos gerais do teste de performance.
- *Service Level Agreement* (SLAs).
- Valores de Baseline: um baseline é um conjunto de métricas usadas para comparar as medidas atuais de performance e as alcançadas anteriormente. Isso permite que melhorias específicas de performance sejam demonstradas e/ou que os critérios de aceite sejam confirmados. Pode ser necessário primeiro criar um baseline usando dados mascarados, quando possível, de um banco de dados.

Dados de teste

Os dados de teste incluem uma ampla variedade de dados que precisam ser especificados para um teste de performance. Esses dados podem incluir o seguinte:

- Dados da conta do usuário (p. ex., contas de usuário disponíveis para login simultâneo).
- Dados de entrada do usuário (p. ex., os dados que um usuário incluiria no aplicativo para executar um processo de negócios).
- Banco de dados (p. ex., o banco de dados com conteúdo que é complementado com dados para uso em testes).

O processo de criação de dados de teste deve abordar os seguintes aspectos:

- Extrair os dados a partir dos dados de produção.
- Importar dados para o SUT.
- Criar dados novos.
- Criar backups que serão usados para restaurar os dados quando novos ciclos de testes forem executados.
- Mascarar os dados. Essa prática é usada em dados de produção que contêm informações de identificação pessoal e é obrigatória de acordo com as *General Data Protection Regulations* (GDPR). No entanto, no teste de performance, o mascaramento de dados aumenta o risco

dos testes de performance, pois pode não ter as mesmas características de dados que as usadas no mundo real.

Configuração do sistema

A seção de configuração do sistema no PTP inclui as seguintes informações técnicas:

- Uma descrição específica da arquitetura do sistema, incluindo servidores (p. ex., web, banco de dados, balanceador de carga).
- Definição de múltiplos níveis.
- Detalhes específicos do hardware de computação, incluindo versões (p. ex., núcleos de CPU, RAM, discos de estado sólido (SSD), discos rígidos (HDD)).
- Detalhes específicos do software, incluindo versões (p. ex., aplicativos, sistemas operacionais, bancos de dados, serviços usados para dar suporte à empresa).
- Sistemas externos que operam com o sistema em teste, e suas configurações e versões (p. ex., sistema de comércio eletrônico com integração ao *NetSuite*).
- Identificação de release/versão do sistema em teste.

Ambiente de teste

O ambiente de teste geralmente é um ambiente separado que imita em menor escala o ambiente de produção. Esta seção do PTP deve incluir como os resultados do teste de performance serão extrapolados para serem aplicados ao ambiente de produção maior. Com alguns sistemas, o ambiente de produção se torna a única opção viável para testes, mas neste caso os riscos específicos desse tipo de teste devem ser discutidos.

Às vezes, as ferramentas de teste residem fora do próprio ambiente de teste e podem exigir direitos de acesso especiais para interagir com os componentes do sistema. Esta é uma consideração para o ambiente de teste e configuração.

Os testes de performance também podem ser realizados em uma parte que compõe o sistema que é capaz de operar sem outros componentes. Geralmente, isso é mais barato do que o teste com todo o sistema e pode ser executado assim que o componente é desenvolvido.

Ferramentas de teste

Esta seção inclui uma descrição de quais ferramentas de teste (e versões) serão usadas na criação de scripts, execução e monitoramento dos testes de performance (consulte o Capítulo 5). Esta lista normalmente inclui:

- As ferramentas usadas para simular transações de usuários.
- As ferramentas que fornecem carga a partir de vários pontos dentro da arquitetura do sistema (pontos de presença).
- As ferramentas que monitoram a performance do sistema, incluindo as descritas acima na configuração do sistema.

Perfis

Os perfis operacionais fornecem um fluxo passo a passo repetível através do aplicativo para um uso específico do sistema. Agregar esses perfis operacionais resulta em um perfil de carga (comumente chamado de cenário). Ver capítulo 4.2.3 para mais informações sobre perfis.

Métricas relevantes

Muitas medições e métricas podem ser coletadas durante a execução dos testes de performance (consulte o Capítulo 2). No entanto, o excesso de medições pode dificultar a análise e influenciar negativamente a performance real do aplicativo. Por essas razões, é importante identificar as medidas e métricas mais relevantes para atingir os objetivos do teste de performance.

A tabela a seguir, detalhada no capítulo 4.4, apresenta um típico conjunto de métricas para teste e monitoramento de performance. Os objetivos de teste para performance devem ser definidos para essas métricas, quando necessário, para o projeto:

Monitorando a Performance	
Tipo	Métrica
Status de usuários virtuais	quantidade_OK, quantidade_NOK
Tempo de resposta de transação	mínimo, máximo, média, 90º percentil
Transações por segundo	quantidade_OK/s, quantidade_NOK/s, total/s
Hits (p. ex., no banco de dados ou no servidor Web)	hits/s (mínimo, máximo, média, total)
Taxa de transferência	bits/s (mínimo, máximo, média, total)
Respostas de http	respostas/s (mínimo, máximo, média, total) classificadas por código
Utilização de CPU	% de utilização
Utilização de memória	% de utilização

Riscos

Os riscos podem incluir áreas não mensuráveis como parte do teste de performance, bem como limitações ao próprio teste (p. ex., interfaces externas que não podem ser simuladas, carga insuficiente, incapacidade de monitorar servidores). Limitações do ambiente de teste também podem produzir riscos (p. ex., dados insuficientes, ambiente reduzido). Consulte as Seções 3.2 e 3.3 para mais tipos de risco.

4.1.3 Comunicação sobre o Teste de Performance

O testador deve ser capaz de comunicar a todos os stakeholders a lógica por trás da abordagem de teste de performance e as atividades a serem realizadas (conforme detalhado no Plano de Teste de Performance). Os assuntos a serem abordados nesta comunicação podem variar consideravelmente entre os *stakeholders*, dependendo se eles têm um interesse voltado para negócios/usuário ou um foco mais voltado para tecnologia/operações.

Stakeholders com foco nos negócios

Os fatores a seguir devem ser considerados ao se comunicar com os *stakeholders* com foco nos negócios:

- Os *stakeholders* com foco nos negócios estão menos interessados nas distinções entre características de qualidade funcionais e não funcionais.
- Questões técnicas relacionadas às ferramentas, scripts e geração de carga geralmente são de interesse secundário.
- A conexão entre os riscos do produto e os objetivos do teste de performance deve ser claramente indicada.
- Os *stakeholders* devem estar cientes do equilíbrio entre o custo dos testes de performance planejados e a representatividade se seu resultado, em comparação com as condições de produção.
- A repetibilidade dos testes de performance planejados deve ser comunicada. O teste será difícil de repetir ou pode ser repetido com um mínimo de esforço?
- Os riscos do projeto devem ser comunicados. Isso inclui restrições e dependências relativas à configuração dos testes, requisitos de infraestrutura (p. ex., hardware, ferramentas, dados, largura de banda, ambiente de teste, recursos) e dependências da equipe principal.
- As atividades de alto nível, devem ser comunicadas (ver capítulos 4.2 e 4.3), juntamente com um plano amplo contendo custos, cronograma e marcos.

Stakeholders com foco em tecnologia

Os seguintes fatores devem ser considerados ao se comunicar com os *stakeholders* com foco em tecnologia:

- A abordagem planejada para gerar os perfis de carga exigidos deve ser explicada, e o envolvimento dos *stakeholders* técnicos deve ser esclarecido.
- O detalhamento das etapas na configuração e execução dos testes de performance deve ser realizado para mostrar a relação dos testes com os riscos de arquitetura.
- As etapas necessárias para tornar os testes de performance repetíveis devem ser comunicadas. Estes podem incluir aspectos organizacionais (p. ex., participação do pessoal-chave), bem como questões técnicas.
- Onde houver compartilhamento dos ambientes de teste, o agendamento da realização dos testes de performance deve ser comunicado para garantir que os resultados não sejam afetados negativamente.
- Mitigações de impactos potenciais em usuários reais devem ser comunicadas e aceitas se o teste de performance necessitar ser executado no ambiente de produção.
- Os *stakeholders* técnicos devem ser claros sobre suas tarefas e seus agendamentos.

4.2 Análise, modelagem e implementação

4.2.1 Protocolos típicos de comunicação

Os protocolos de comunicação definem um conjunto de regras de comunicação entre computadores e sistemas. Projetar adequadamente os testes para direcionar partes específicas do sistema requer protocolos de entendimento.

Os protocolos de comunicação são frequentemente descritos pelas camadas do modelo *Open Systems Interconnection* (OSI) (ver ISO/IEC 7498-1), embora alguns protocolos possam ficar fora desse modelo. Para testes de performance, os protocolos da Camada 5 (Camada de Sessão) para a Camada 7 (Camada de Aplicação) são mais comumente usados para testes de performance. Protocolos comuns incluem:

- **Banco de dados**: ODBC, JDBC, outros protocolos específicos do fornecedor
- **Web**: HTTP, HTTPS, HTML
- **Serviço da web**: SOAP, REST

De um modo geral, o nível da camada OSI que está mais focado no teste de performance está relacionado ao nível da arquitetura que está sendo testada. Ao testar alguma arquitetura incorporada de baixo nível, por exemplo, as camadas com números mais baixos do modelo OSI estarão principalmente em foco.

Protocolos adicionais usados no teste de performance incluem:

- **Rede**: DNS, FTP, IMAP, LDAP, POP3, SMTP, Windows Sockets, CORBA
- **Móvel**: TruClient, SMP, MMS
- **Acesso remoto**: Citrix ICA, RTE
- **SOA**: MQSeries, JSON, WSCL

É importante entender a arquitetura geral do sistema porque os testes de performance podem ser executados em um componente individual do sistema (p. ex., servidor da Web, servidor de banco de dados) ou em todo um sistema por meio de testes de ponta a ponta. Os aplicativos tradicionais de 2 camadas criados com um modelo cliente-servidor especificam o cliente como a GUI e a interface de usuário principal, e o servidor como o banco de dados de *backend*. Esses aplicativos exigem o uso de um protocolo como o ODBC para acessar o banco de dados. Com a evolução dos aplicativos baseados na Web e das arquiteturas de várias camadas, muitos servidores estão envolvidos no processamento de informações que são renderizadas no navegador do usuário.

Dependendo da parte do sistema que é direcionado para o teste, é necessário um entendimento do protocolo apropriado a ser usado. Portanto, se a necessidade for realizar testes de ponta a ponta emulando a atividade do usuário a partir do navegador, um protocolo da Web como http/httpS será empregado. Dessa forma, a interação com a GUI pode ser ignorada e os testes podem se concentrar na comunicação e nas atividades dos servidores *backend*.

4.2.2 Transações

As transações descrevem o conjunto de atividades realizadas por um sistema desde o ponto de início até quando um ou mais processos (solicitações, operações ou processos operacionais) foram concluídos. O tempo de resposta das transações pode ser medido com o objetivo de avaliar a performance do sistema. Durante um teste de performance, essas medições são usadas para identificar quaisquer componentes que exijam correção ou otimização.

As transações simuladas podem incluir o tempo de interação do usuário com o sistema para refletir melhor o tempo de um usuário real realizando uma ação (p. ex., pressionando o botão ENVIAR). O tempo de resposta da transação mais o tempo de interação é igual ao tempo decorrido para essa transação.

Os tempos de resposta da transação coletados durante o teste de performance mostram como essa medida muda sob diferentes cargas impostas ao sistema. A análise pode não mostrar degradação sob carga, enquanto outras medidas podem mostrar degradação severa. Aumentando a carga e medindo os tempos de transação subjacentes, é possível correlacionar a causa da degradação com os tempos de resposta de uma ou mais transações. Os tempos de resposta da transação coletados durante o teste de performance mostram como essa medição modifica sob diferentes cargas impostas ao sistema. A análise pode não mostrar degradação sob carga, enquanto outras medições podem mostrar degradação severa. Aumentando a carga e mensurando os tempos de transação subjacentes, é possível correlacionar a causa da degradação com os tempos de resposta de uma ou mais transações.

As transações também podem ser aninhadas para que atividades individuais e agregadas possam ser medidas. Isso pode ser útil, por exemplo, ao entender a eficiência da performance de um sistema de pedidos on-line. O testador pode querer medir as etapas discretas no processo de pedido (p. ex., pesquisar item, adicionar item ao carrinho, pagar pelo item, confirmar pedido), bem como o processo de pedido como um todo. Ao aninhar transações, os dois conjuntos de informações podem ser reunidos em um teste.

4.2.3 Identificando perfis operacionais

Os perfis operacionais especificam padrões distintos de interação com um aplicativo, como de usuários ou outros componentes do sistema. Vários perfis operacionais podem ser especificados para um determinado aplicativo. Eles podem ser combinados para criar um perfil de carga desejado para alcançar objetivos de teste de performance específicos (ver capítulo 4.2.4).

As principais etapas a seguir para identificar perfis operacionais são descritas nesta seção:

1. Identificar os dados a serem coletados
2. Reunir os dados usando uma ou mais fontes
3. Avaliar os dados para construir os perfis operacionais

Identificar dados

Quando os usuários interagem com o sistema em teste, os dados a seguir são reunidos ou estimados para modelar seus perfis operacionais (p. ex., como eles interagem com o sistema):

- Diferentes tipos de usuários e suas funções (p. ex., usuário padrão, membro registrado, administrador, grupos de usuários com privilégios específicos).
- Tarefas genéricas diferentes executadas por esses usuários/funções (p. ex., navegar em um site da Web para obter informações, pesquisar um site em um determinado produto, realizar atividades específicas da função). Observe que essas tarefas geralmente são modeladas em um alto nível de abstração (p. ex., no nível de processos de negócios ou principais histórias de usuários).
- Número estimado de usuários para cada função/tarefa por unidade de tempo durante um determinado período. Essas informações também serão úteis para a construção posterior de perfis de carga (ver capítulo 4.2.4).

Reunir dados

Os dados mencionados acima podem ser coletados de várias fontes diferentes:

- Realização de entrevistas ou workshops com os *stakeholders*, como Product Owners, gerentes de vendas e (potenciais) usuários finais. Essas discussões geralmente revelam os principais perfis operacionais dos usuários e fornecem respostas para a pergunta fundamental para quem este aplicativo é destinado.
- Especificações funcionais e requisitos (quando disponíveis) são uma fonte valiosa de informações sobre os padrões de uso pretendidos que também podem ajudar a identificar os tipos de usuários e seus perfis operacionais. Quando as especificações funcionais são expressas como histórias do usuário, o formato padrão permite que os tipos de usuários sejam identificados (isto é, *como [tipo de usuário], quero [algum recurso] para que [algum benefício]*). Da mesma forma, os diagramas e descrições de casos de uso da UML identificam o ator para o caso de uso.
- A avaliação de dados de uso e métricas obtidas de aplicativos semelhantes pode oferecer suporte à identificação de tipos de usuários e fornecer algumas indicações iniciais do número esperado de usuários. O acesso a dados monitorados automaticamente é recomendado (p. ex., de uma ferramenta de administração do webmaster). Isso incluirá logs de monitoramento e dados obtidos do uso do sistema operacional atual, onde uma atualização para esse sistema é planejada
- O monitoramento do comportamento dos usuários ao executar tarefas predefinidas com o aplicativo pode fornecer informações sobre os tipos de perfis operacionais a serem montados para testes de performance. Coordenar essa tarefa com qualquer teste de usabilidade planejado é recomendado (especialmente se um laboratório de usabilidade estiver disponível).

Construir perfis operacionais

As etapas a seguir são usadas para identificar e construir perfis operacionais para usuários:

- É adotada uma abordagem de cima para baixo. Perfis operacionais amplos e relativamente simples são inicialmente criados e só são divididos se isso for necessário para atingir os objetivos do teste de performance (ver capítulo 4.1.1).
- Perfis de usuários específicos podem ser destacados como relevantes para testes de performance se, envolverem tarefas que são executadas com frequência, requererem

transações críticas (alto risco) ou frequentes entre diferentes componentes do sistema ou potencialmente exigem grandes volumes de dados a serem transferidos.
- Os perfis operacionais são revisados e refinados com os principais *stakeholders* antes de serem usados para a criação de perfis de carga (ver capítulo 4.2.4).

O sistema em teste nem sempre é sujeito a cargas impostas pelo usuário. Perfis operacionais também podem ser necessários para testes de performance dos seguintes tipos de sistema (observe que esta lista não é extensa):

Sistemas de processamento batch off-line

O foco aqui está principalmente no rendimento do sistema de processamento batch (ver capítulo 4.2.5) e sua capacidade de concluir dentro de um determinado período. Os perfis operacionais concentram-se nos tipos de processamento exigidos dos processamentos batch. Por exemplo, os perfis operacionais de um sistema de negociação de ações (que normalmente inclui processamento de transações on-line e batch) podem incluir aqueles relacionados a transações de pagamento, verificação de credenciais e verificação da conformidade de condições legais para tipos específicos de transações de estoque. Cada um desses perfis operacionais resultaria em diferentes caminhos percorridos através do processamento batch geral de um estoque. As etapas descritas acima para identificar os perfis operacionais de sistemas on-line baseados no usuário também podem ser aplicadas no contexto de processamento batch.

Sistemas de sistemas

Os componentes dentro de um ambiente com vários sistemas (que também podem ser embarcados) respondem a diferentes tipos de entrada de outros sistemas ou componentes. Dependendo da natureza do sistema em teste, isso pode exigir a modelagem de vários perfis operacionais diferentes para representar efetivamente os tipos de entrada fornecidos por esses sistemas de fornecedores. Isso pode envolver análise detalhada (p. ex., de buffers e filas) junto com os arquitetos do sistema e com base nas especificações do sistema e da interface.

4.2.4 Criar perfis de carga

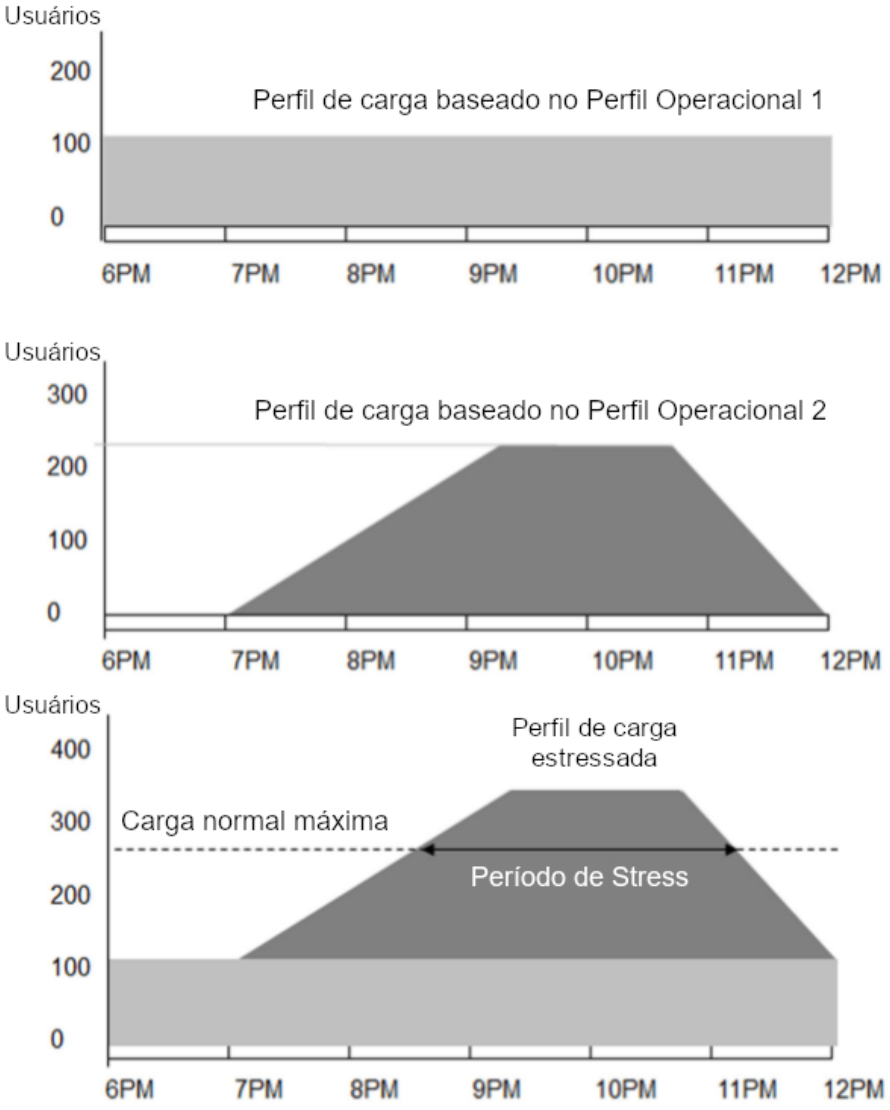
Um perfil de carga especifica a atividade que um componente ou sistema que está sendo testado pode experimentar na produção. Ele consiste em um número modelado de instâncias que executará as ações de perfis operacionais predefinidos em um período específico. Onde as instâncias são usuários, o termo "*usuários virtuais*" é comumente aplicado.

As principais informações necessárias para criar um perfil de carga realista e repetitivo são:

- O objetivo do teste de performance (p. ex., avaliar o comportamento do sistema sob cargas de tensão).
- Perfis operacionais que representam com precisão os padrões de uso individuais (ver capítulo 4.2.3).
- Problemas conhecidos de taxa de transferência e simultaneidade (ver capítulo 4.2.5)
- A quantidade e a distribuição de tempo com os quais os perfis operacionais devem ser executados, de modo que o SUT experimente a carga desejada. Exemplos típicos são:

- **Aceleração**: aumento constante da carga (p. ex., adicionar um usuário virtual por minuto)
- **Desaceleração**: carga decrescente constante.
- **Etapas**: alterações instantâneas no carregamento (p. ex., adicionar 100 usuários virtuais a cada cinco minutos).
- **Distribuições predefinidas** (p. ex., o volume simula ciclos de negócios diários ou sazonais).

O exemplo a seguir mostra a construção de um perfil de carga com o objetivo de gerar condições de tensão (no máximo ou acima do máximo esperado para um sistema manipular) para o sistema em teste.



Na parte superior do diagrama, é mostrado um perfil de carga que consiste em um passo com uma entrada de 100 usuários virtuais. Esses usuários executam as atividades definidas pelo Perfil Operacional 1 em toda a duração do teste. Isso é típico de muitos perfis de carga de performance que representam uma carga de segundo plano.

O diagrama do meio mostra um perfil de carga que consiste em um aumento de até 220 usuários virtuais que é mantido por duas horas antes da redução. Cada usuário virtual executa atividades definidas no Perfil Operacional 2.

O diagrama inferior mostra o perfil de carga que resulta da combinação dos dois descritos acima. O sistema sob teste é submetido a um período de três horas de estresse.

Para mais exemplos, consulte [Bath14].

4.2.5 Analisar a taxa de transferência e concorrência

É importante entender diferentes aspectos da carga de trabalho: taxa de transferência e simultaneidade. Para modelar perfis operacionais e de carga adequadamente, ambos os aspectos devem ser levados em consideração.

Taxa de transferência do sistema

A taxa de transferência do sistema é uma medida do número de transações de um determinado tipo que o sistema processa em uma unidade de tempo. Por exemplo, o número de pedidos por hora ou o número de solicitações http por segundo. A taxa de transferência do sistema deve ser diferenciada da taxa de transferência da rede, que é a quantidade de dados trafegados pela rede (ver capítulo 2.1).

A taxa de transferência do sistema define a carga no sistema. Infelizmente, muitas vezes o número de usuários simultâneos é usado para definir a carga de sistemas interativos em vez de taxa de transferência. Isso é parcialmente verdadeiro porque esse número é mais fácil de encontrar, mas parcial porque é a maneira como as ferramentas de teste de carga definem a carga. Sem definir perfis operacionais - o que cada usuário está fazendo e com que intensidade (que também é taxa de transferência para um usuário) - o número de usuários não é uma boa medida de carga. Por exemplo, se houver 500 usuários executando consultas curtas a cada minuto, teremos uma taxa de transferência de 30.000 consultas por hora. Se os mesmos 500 usuários estiverem executando as mesmas consultas, mas uma por hora, a taxa de transferência será de 500 consultas por hora. Portanto, existem os mesmos 500 usuários, mas uma diferença de 60x entre cargas e pelo menos uma diferença de 60x nos requisitos de hardware para o sistema.

A modelagem de carga de trabalho geralmente é feita considerando o número de usuários virtuais (processamento distribuído de execução) e o tempo de interação do usuário com o sistema (atrasos entre as ações do usuário). No entanto, o rendimento do sistema também é definido pelo tempo de processamento, e esse tempo pode aumentar conforme a carga aumenta.

$$\text{Taxa de transferência do sistema} = \frac{número\ de\ usuários\ virtuais}{tempo\ de\ processamento + tempo\ de\ interação}$$

Assim, quando o tempo de processamento aumenta, a taxa de transferência pode diminuir significativamente, mesmo que todo o resto permaneça o mesmo.

O rendimento do sistema é um aspecto importante ao testar sistemas de processamento em lote. Neste caso, o rendimento é tipicamente medido de acordo com o número de transações que podem ser realizadas dentro de um determinado período (p. ex., uma janela de processamento de lote noturna).

Concorrência

A simultaneidade é uma medida do número de encadeamentos de execução simultâneos/paralelos. Para sistemas interativos, pode ser um número de usuários simultâneos/paralelos. A concorrência é geralmente modelada em ferramentas de teste de carga, definindo o número de usuários virtuais.

Concorrência é uma medida importante. Representa o número de sessões paralelas, cada uma das quais pode usar seus próprios recursos. Mesmo que a taxa de transferência seja a mesma, a quantidade de recursos usada pode variar dependendo da simultaneidade. As configurações típicas de teste são sistemas fechados (do ponto de vista da teoria de enfileiramento), em que o número de usuários no sistema é definido (população fixa). Se todos os usuários estiverem aguardando a resposta do sistema em um sistema fechado, nenhum novo usuário poderá chegar. Muitos sistemas públicos são sistemas abertos - novos usuários chegam o tempo todo, mesmo que todos os usuários atuais estejam aguardando a resposta do sistema.

4.2.6 Estrutura básica de um script de Teste de Performance

Um script de teste de performance deve simular uma atividade de usuário ou componente que contribua para a carga no sistema em teste (que pode ser todo o sistema ou um de seus componentes). Ele inicia com solicitações para o servidor em uma ordem apropriada e em um determinado ritmo.

A melhor maneira de criar scripts de teste de performance depende da abordagem de geração de carga usada (ver capítulo 4.1).

- A maneira tradicional é gravar a comunicação entre o cliente e o sistema ou componente no nível do protocolo e reproduzi-lo depois que o script foi parametrizado e documentado. A parametrização resulta em um script escalável e de fácil manutenção, mas a tarefa de parametrização pode ser demorada.
- A gravação no nível da GUI geralmente envolve a captura de ações GUI de um único cliente com uma ferramenta de execução de teste e a execução desse script com a ferramenta de geração de carga para representar vários clientes.
- A programação pode ser feita usando solicitações de protocolo (p. ex., solicitações de http), ações de GUI ou chamadas de API. No caso de scripts de programação, a sequência exata de solicitações enviadas e recebidas do sistema real deve ser determinada, o que pode não ser trivial.

Geralmente, um script é uma ou várias seções de código (escritas em uma linguagem de programação genérica com algumas extensões ou em uma linguagem especializada) ou um objeto, que pode ser apresentado a um usuário pela ferramenta em uma GUI. Em ambos os casos, o script incluirá solicitações de servidor criando carga (p. ex., solicitações http) e alguma lógica de programação em torno delas especificando como exatamente essas solicitações seriam invocadas (p. ex., em que ordem, em que momento, com quais parâmetros, o que deve ser verificado). Quanto mais sofisticada a lógica, mais será necessária a utilização de linguagens mais poderosas de programação.

Estrutura geral

Muitas vezes, o script tem uma seção de inicialização (onde tudo é preparado para a parte principal), seções principais que podem ser executadas várias vezes e uma seção de limpeza (onde as etapas necessárias são tomadas para concluir o teste corretamente).

Coleção de dados

Para coletar tempos de resposta, os cronômetros devem ser adicionados ao script para mensurar quanto tempo uma solicitação ou uma combinação de solicitações demora. As solicitações cronometradas devem corresponder a uma unidade significativa de trabalho lógico - por exemplo, uma transação comercial para adicionar um item a um pedido ou enviar um pedido.

É importante entender o que exatamente é mensurado: no caso de scripts em nível de protocolo, é o tempo de resposta do servidor e da rede apenas, enquanto os scripts da GUI mensuram o tempo de finalização (embora o que é exatamente mensurado dependa da tecnologia usada).

Verificação de resultados e tratamento de erros

Uma parte importante do script é a verificação de resultados e o tratamento de erros. Mesmo nas melhores ferramentas de teste de carga, o tratamento de erros padrão tende a ser mínimo (como a verificação do código de retorno da solicitação http), portanto, recomenda-se adicionar verificações adicionais para verificar o retorno real das solicitações. Além disso, se qualquer limpeza for necessária em caso de erro, provavelmente será necessário implementá-la manualmente. Uma boa prática é verificar se o script está fazendo o que é suposto fazer usando métodos indiretos (p. ex., analisando o banco de dados para verificar se as informações apropriadas foram adicionadas).

Os scripts podem incluir outras regras de especificação de lógica referentes à quando e como as solicitações do servidor serão feitas. Um exemplo é a configuração de pontos de sincronização, o que é feito especificando que o script deve aguardar um evento nesse ponto antes de continuar. Os pontos de sincronização podem ser usados para garantir que uma ação específica seja chamada simultaneamente ou para coordenar o trabalho entre vários scripts.

Scripts de teste de performance são software, portanto, criar um script de teste de performance é uma atividade de desenvolvimento de software. Deve incluir a garantia de qualidade e testes para verificar se o script funciona como esperado com toda a gama de dados de entrada.

4.2.7 Implementando scripts de teste de performance

Os scripts de teste de performance são implementados com base no PTP e nos perfis de carga. Embora os detalhes técnicos da implementação sejam diferentes, dependendo da abordagem e da(s) ferramenta(s) usada(s), o processo geral permanece o mesmo. Um script de performance é criado usando um *Integrated Development Environment* (IDE) ou editor de script, para simular um comportamento de usuário ou componente. Normalmente, o script é criado para simular um perfil operacional específico (embora geralmente seja possível combinar vários perfis operacionais em um script com instruções condicionais).

Conforme a sequência de solicitações é determinada, o script pode ser gravado ou programado dependendo da abordagem. O registro geralmente garante que ele simule exatamente o sistema real, enquanto a programação depende do conhecimento da sequência de solicitação apropriada.

Se a gravação no nível do protocolo for usada, uma etapa essencial após a gravação na maioria dos casos é a substituição de todos os identificadores internos registrados que definem o contexto. Esses identificadores devem ser transformados em variáveis que podem ser alteradas entre execuções com valores apropriados que são extraídos das respostas da solicitação (p. ex., um identificador de usuário que é adquirido no login e deve ser fornecido para todas as transações subsequentes). Essa é uma parte da parametrização de script, às vezes chamada de correlação. Nesse contexto, a palavra correlação tem um significado diferente do usado na estatística (onde significa relação entre duas ou mais coisas). As ferramentas avançadas de teste de carga podem fazer alguma correlação automaticamente, portanto, pode ser transparente em alguns casos, mas em casos mais complexos, a correlação manual ou a adição de novas regras de correlação podem ser necessárias. A correlação incorreta ou sua ausência é a principal razão pela qual os scripts gravados falham na reprodução.

Executar vários usuários virtuais com o mesmo nome de usuário e acessar o mesmo conjunto de dados (como geralmente acontece durante a reprodução de um script gravado sem qualquer outra modificação além da correlação necessária) é uma maneira fácil de obter resultados enganosos. Os dados podem ser completamente armazenados em cache (copiados do disco para a memória para acesso mais rápido) e os resultados seriam muito melhores do que na produção (onde esses dados podem ser lidos de um disco). Usar os mesmos usuários e/ou dados também pode causar problemas de simultaneidade (p. ex., se os dados estiverem bloqueados quando um usuário estiver atualizando-os) e os resultados seriam muito piores do que na produção, pois o software aguardaria a liberação do bloqueio antes do próximo usuário que poderia bloquear os dados para atualização.

Portanto, os scripts e as configurações de teste devem ser parametrizados (isto é, os dados fixos ou gravados devem ser substituídos por valores de uma lista de possíveis escolhas), para que cada usuário virtual use um conjunto de dados adequado. O termo apropriado aqui significa diferente o suficiente para evitar problemas com armazenamento em cache e simultaneidade, que é específico para o sistema, dados e requisitos de teste. Essa parametrização adicional depende dos dados no sistema e da forma como o sistema trabalha com esses dados, portanto, isso geralmente é feito manualmente, embora muitas ferramentas forneçam alguma assistência.

Há casos em que alguns dados devem ser parametrizados para que o teste funcione mais de uma vez, por exemplo, quando um pedido é criado e o nome do pedido deve ser exclusivo. A menos que o nome do pedido seja parametrizado, o teste falhará assim que ele tentar criar um pedido com um nome existente (gravado).

Para corresponder aos perfis operacionais, os tempos de interação do usuário com o sistema devem ser inseridos e/ou ajustados (se registrados) para gerar um número adequado de solicitações/taxa de transferência, conforme discutido no capítulo 4.2.5.

Quando scripts para perfis operacionais separados são criados, eles são combinados em um cenário que implementa todo o perfil de carga. O perfil de carga controla quantos usuários virtuais são iniciados usando cada script, quando e com quais parâmetros. Os detalhes exatos da implementação dependem da ferramenta de teste de carga específica ou da configuração.

4.2.8 Preparando-se para a execução do teste de performance

As principais atividades para preparar a execução dos testes de performance incluem:

- Configuração do sistema em teste.
- Implantação do ambiente.
- Configuração das ferramentas de geração de carga e monitoramento garantindo que todas as informações necessárias sejam coletadas.

É importante garantir que o ambiente de teste seja o mais próximo possível do ambiente de produção. Se isso não for possível, deve haver um entendimento claro das diferenças e como os resultados do teste serão projetados para o ambiente de produção. Preferencialmente, o ambiente de produção e os dados reais seriam usados, mas o teste em um ambiente reduzido ainda pode ajudar a atenuar vários riscos de performance.

É importante lembrar que a performance é uma função não-linear do ambiente, portanto, quanto mais longe o ambiente for do padrão de produção, mais difícil serão precisas as projeções da performance da produção. A falta de confiabilidade das projeções e o aumento do nível de risco aumentam à medida que o sistema de teste se parece menos com a produção.

As partes mais importantes do ambiente de teste são dados, configuração de hardware e software e a configuração da rede. O tamanho e a estrutura dos dados podem afetar drasticamente os resultados do teste de carga. Usar um pequeno conjunto de dados de amostra ou um conjunto de amostras com uma complexidade de dados diferente para testes de performance pode gerar resultados enganosos, especialmente quando o sistema de produção usará um grande conjunto de dados. É difícil prever quanto o tamanho dos dados afeta a performance antes que o teste real seja realizado. Quanto mais próximos os dados de teste estiverem dos dados de produção em tamanho e estrutura, mais confiáveis serão os resultados do teste.

Se os dados forem gerados ou alterados durante o teste, pode ser necessário restaurar os dados originais antes do próximo ciclo de teste para garantir que o sistema esteja no estado adequado.

Se algumas partes do sistema ou alguns dos dados não estiverem disponíveis para testes de performance por qualquer motivo, uma solução alternativa deve ser implementada. Por exemplo, um simulador pode ser implementado para substituir e emular um componente de terceiros, responsável pelo processamento do cartão de crédito. Esse processo é geralmente chamado de virtualização de serviço e existem ferramentas especiais disponíveis para ajudar nesse processo. O uso de tais ferramentas é altamente recomendado para isolar o sistema em teste.

Existem muitas maneiras de implantar ambientes. Por exemplo, as opções podem incluir o uso de qualquer um dos itens a seguir:

- Laboratórios de teste internos (e externos) tradicionais.
- Nuvem como um ambiente usando *Infraestrutura as a Service* (IaaS), quando algumas partes do sistema ou todo o sistema é implantado na nuvem.
- Nuvem como um ambiente usando o *Software as a Service* (SaaS), quando os fornecedores fornecem o serviço de teste de carga.

Dependendo dos objetivos específicos e dos sistemas a serem testados, um ambiente de teste pode ser preferido em detrimento de outro. Por exemplo:

- Para testar o efeito de uma melhoria de performance (otimização de performance), usar um ambiente de teste isolado pode ser a melhor opção para notar até mesmo pequenas variações introduzidas pela mudança.
- Para carregar o teste de todo o ambiente de produção de ponta-a-ponta para garantir que o sistema manipulará a carga sem maiores problemas, os testes da nuvem ou de um serviço podem ser mais apropriados (observe que isso só funciona para os SUTs que podem ser acessados a partir de uma nuvem).
- Para minimizar os custos quando os testes de performance são limitados pelo tempo, a criação de um ambiente de teste na nuvem pode ser uma solução mais econômica.

Seja qual for a abordagem usada para implantação, o hardware e o software devem ser configurados para atender ao objetivo e ao plano de teste. Se o ambiente corresponder à produção, ele deverá ser configurado da mesma maneira. No entanto, se houver diferenças, a configuração poderá ser ajustada para acomodar essas diferenças. Por exemplo, se as máquinas de teste possuírem menos memória física do que as máquinas de produção, os parâmetros de memória do software (como o tamanho de *heap* no Java) talvez precisem ser ajustados para evitar paginação de memória.

A configuração/emulação adequada da rede é importante para sistemas globais e móveis. Para sistemas globais (ou seja, um que tenha usuários ou processamento distribuído em todo o mundo), uma das abordagens pode ser implantar geradores de carga em locais onde os usuários estão localizados. Para sistemas móveis, a emulação de rede continua a ser a opção mais viável devido às variações nos tipos de rede que podem ser usados. Algumas ferramentas de teste de carga têm ferramentas de emulação de rede incorporadas e existem ferramentas independentes para emulação de rede.

As ferramentas de geração de carga devem ser implementadas corretamente e as ferramentas de monitoramento devem ser configuradas para coletar todas as métricas necessárias para o teste. A lista de métricas depende dos objetivos do teste, mas é recomendável coletar pelo menos as métricas básicas para todos os testes (ver capítulo 2.1.2).

Dependendo da carga, da abordagem específica da ferramenta/geração de carga e da configuração da máquina, mais de uma máquina de geração de carga pode ser necessária. Para verificar a configuração, as máquinas envolvidas na geração de carga também devem ser monitoradas. Isso ajudará a evitar uma situação em que a carga não seja mantida adequadamente porque um dos geradores de carga está sendo executado lentamente.

Dependendo da configuração e das ferramentas usadas, as ferramentas de teste de carga precisam ser configuradas para criar a carga apropriada. Por exemplo, parâmetros específicos de emulação do navegador podem ser definidos ou IP dinâmicos (simulando que cada usuário virtual tem um endereço IP diferente) pode ser usado.

Antes dos testes serem executados, o ambiente e a configuração devem ser validados. Isso geralmente é feito pela realização de um conjunto controlado de testes e pela verificação de seus resultados, bem como pela verificação de que as ferramentas de monitoramento estão rastreando as informações importantes.

Para verificar se o teste funciona conforme projetado, várias técnicas podem ser usadas, incluindo análise de log e verificação do conteúdo do banco de dados. A preparação para o teste inclui verificar

se as informações necessárias são registradas, se o sistema está no estado adequado, entre outros. Por exemplo, se o teste alterar significativamente o estado do sistema (adicionar ou alterar informações no banco de dados), pode ser necessário retornar o sistema ao estado original antes de repetir o teste.

4.3 Execução

A execução do teste de performance envolve a geração de uma carga submetendo-a ao sistema em teste de acordo com um perfil de carga (geralmente implementado pelos scripts de teste de performance invocados de acordo com um determinado cenário), monitorando todas as partes do ambiente e coletando e mantendo todos os resultados e informações relacionados ao teste. Geralmente, as ferramentas/configurações de teste de carga avançados realizam essas tarefas automaticamente (após, é claro, a adequada configuração). Eles geralmente fornecem um console para permitir que os dados de performance sejam monitorados durante o teste e permitem que os ajustes necessários sejam feitos (ver capítulo 5.1). No entanto, dependendo da ferramenta usada, algumas etapas manuais podem ser necessárias, quando o sistema em teste e os testes específicos estão sendo executados.

Os testes de performance geralmente são focados em um estado estável do sistema, ou seja, quando o comportamento do sistema é estável. Por exemplo, quando todos os usuários/processamentos distribuídos simulados são iniciados e estão executando o trabalho conforme projetado. Quando a carga está mudando (p. ex., quando novos usuários são adicionados), o comportamento do sistema está mudando e fica mais difícil monitorar e analisar os resultados dos testes. O estágio de chegar ao estado estacionário é geralmente chamado de aceleração, e o estágio de término do teste é geralmente chamado de desaceleração.

Às vezes, é importante testar estados transitórios quando o comportamento do sistema está mudando. Isso pode se aplicar, por exemplo, ao registro simultâneo de muitos usuários ou testes de pico. Ao testar estados transitórios, é importante entender a necessidade da cuidadosa monitoração e análise dos resultados, já que algumas abordagens comuns podem ser enganosas (como as médias de monitoramento).

Durante a aceleração, é aconselhável implementar estados de carga incremental para monitorar o impacto da carga crescente na resposta do sistema. Isso garante que o tempo suficiente seja alocado para a aceleração e que o sistema seja capaz de lidar com a carga. Uma vez que o estado de estabilidade tenha sido alcançado, é uma boa prática monitorar que tanto a carga quanto as respostas do sistema são estáveis e que as variações aleatórias (que sempre existirão) não são substanciais.

É importante especificar como as falhas devem ser tratadas para garantir que nenhum problema no sistema seja introduzido. Por exemplo, pode ser importante que o usuário efetue logout quando ocorrer uma falha para garantir que todos os recursos associados a esse usuário sejam liberados.

Se o monitoramento estiver integrado à ferramenta de teste de carga e estiver configurado corretamente, ele geralmente será iniciado ao mesmo tempo que a execução do teste. No entanto, se as ferramentas de monitoramento independentes forem usadas, o monitoramento deve ser iniciado separadamente e as informações necessárias coletadas para que a análise subsequente

possa ser realizada junto com os resultados do teste, valendo o mesmo para a análise de log. É essencial sincronizar todas as ferramentas usadas no tempo, para que todas as informações relacionadas a um ciclo de execução de teste específico possam ser localizadas.

A execução do teste é frequentemente monitorada usando o console da ferramenta de teste de performance e a análise de registro em tempo real para verificar problemas e erros no teste e no SUT. Isso ajuda a evitar a continuação desnecessária da execução de testes em grande escala, o que pode até causar impacto em outros sistemas se algo der errado (p. ex., se ocorrer falha, os componentes falharem ou as cargas geradas estiverem muito baixas ou altas). Esses testes podem ser caros de se executar podendo ser necessário interromper o teste ou fazer alguns ajustes imediatos no teste de performance ou na configuração do sistema se o teste se desviar do comportamento esperado.

Uma técnica para verificar os testes de carga que estão se comunicando diretamente no nível do protocolo é executar vários scripts (funcionais) no nível da GUI ou até mesmo executar manualmente perfis operacionais semelhantes em paralelo ao teste de carga em execução. Isso verifica se os tempos de resposta relatados durante o teste diferem apenas dos tempos de resposta medidos manualmente no nível da GUI pelo tempo gasto no lado do cliente.

Em alguns casos, ao executar testes de performance de maneira automatizada (p. ex., como parte da Integração Contínua, conforme discutido no capítulo 3.4), as verificações devem ser feitas automaticamente, uma vez que o monitoramento manual e a intervenção podem não ser possíveis. Nesse caso, a configuração do teste deve ser capaz de reconhecer quaisquer desvios ou problemas e emitir um alerta (geralmente ao concluir o teste corretamente). Essa abordagem é mais fácil de implementar para testes de performance de regressão quando o comportamento do sistema é geralmente conhecido, mas pode ser mais difícil com testes de performance exploratório ou testes caros de performance em grande escala que podem precisar de ajustes dinâmicos durante o teste.

4.4 Analisando os resultados e relatórios

O capítulo 4.1.2 discutiu as várias métricas em um plano de teste de performance. Definindo estes itens inicialmente determina-se o que deve ser medido para cada teste. Após a conclusão de um ciclo de teste, os dados devem ser coletados para as métricas definidas.

Ao analisar os dados, ele é comparado primeiro ao objetivo do teste de performance. Uma vez que o comportamento é entendido, podem ser tiradas conclusões que fornecem um relatório de resumo significativo que inclui as ações recomendadas. Essas ações podem incluir a alteração de componentes físicos (p. ex., hardware, roteadores), alteração de software (p. ex., otimização de aplicativos e chamadas de banco de dados) e alteração da rede (p. ex., balanceamento de carga, roteamento).

Os dados a seguir são comumente analisados:

- **Status de usuários simulados** (p. ex., virtuais). Isso precisa ser examinado primeiro. Normalmente, é esperado que todos os usuários simulados tenham conseguido realizar as tarefas específicas de seu perfil operacional. Qualquer interrupção nessa atividade imitaria o que um usuário real pode experimentar. Isso torna muito importante primeiro ver que toda

a atividade do usuário é concluída, pois os erros encontrados podem influenciar os outros dados de performance.

- **Tempo de resposta da transação**. Isso pode ser medido de várias maneiras, incluindo mínimo, máximo, médio e percentil (p. ex., 90º). As leituras mínima e máxima mostram os extremos da performance do sistema. A performance média não é necessariamente um indicativo de outra coisa senão a média aritmética e muitas vezes pode ser distorcida por fatores externos. O 90º percentil geralmente é usado como meta, pois representa a maioria dos usuários que atingem um limite específico de performance. Não é recomendado exigir 100% de conformidade com os objetivos de performance, pois os recursos necessários podem ser muito grandes e o efeito resultante para os usuários será geralmente menor.
- **Transações por segundo**. Isso fornece informações sobre quanto trabalho foi feito pelo sistema (taxa de transferência do sistema).
- **Falhas de transação**. Esses dados são usados ao analisar transações por segundo. As falhas indicam que o evento ou processo esperado não foi concluído ou não foi executado. Quaisquer falhas encontradas são motivo de preocupação e a causa raiz deve ser investigada. As transações com falha também podem resultar em transações inválidas por segundo, uma vez que uma transação com falha levará muito menos tempo que uma concluída.
- **Ocorrências (ou solicitações) por segundo**. Isso fornece uma noção do número de acessos a um servidor pelos usuários simulados durante cada segundo do teste.
- **Taxa de transmissão de rede**. Isso geralmente é medido em *bits* por intervalo de tempo, como *bits/s*. Isso representa a quantidade de dados que os usuários simulados recebem do servidor a cada segundo. (ver capítulo 4.2.5)
- **Respostas de http**. Eles são medidos por segundo e incluem possíveis códigos de resposta, como: 200, 302, 304, 404, o último indicando que uma página não foi encontrada.

Embora muitas dessas informações possam ser apresentadas em tabelas, as representações gráficas facilitam a visualização dos dados e a identificação de tendências.

As técnicas usadas na análise de dados podem incluir:

- Comparando os resultados aos requisitos declarados.
- Observando as tendências nos resultados.
- Utilizando técnicas estatísticas de controle de qualidade.
- Identificando erros.
- Comparando os resultados esperados e reais.
- Comparando os resultados atuais com os de testes anteriores.
- Verificando o funcionamento adequado dos componentes (p. ex., servidores, redes).

Identificar a correlação entre métricas pode nos ajudar a entender em que ponto a performance do sistema começa a se degradar. Por exemplo, "*Qual é número de transações por segundo que foram processadas quando a CPU atingiu 90% da capacidade e o sistema ficou lento?*".

A análise pode ajudar a identificar a causa raiz da degradação ou falha da performance, o que, por sua vez, facilitará uma correção. O teste de confirmação ajudará a determinar se a ação corretiva resolveu a causa raiz.

Relatórios

Os resultados da análise são consolidados e comparados com os objetivos declarados no plano de teste de performance. Eles podem ser relatados no relatório de situação geral do teste juntamente com outros resultados de teste ou incluídos em um relatório dedicado para testes de performance. O nível de detalhes relatado deve corresponder às necessidades dos *stakeholders*. As recomendações baseadas nesses resultados geralmente abordam os critérios de liberação de software (incluindo o ambiente de destino) ou as melhorias de performance necessárias.

Um relatório de teste de performance típico pode incluir:

Sumário executivo

Esta seção é concluída quando todos os testes de performance tiverem sido executados e todos os resultados tiverem sido analisados e compreendidos. O objetivo é apresentar conclusões, descobertas e recomendações concisas e compreensíveis para o gerenciamento com o objetivo de obter-se um resultado com ações específicas.

Resultado de testes

Os resultados do teste podem incluir algumas ou todas as informações a seguir:

- Um resumo fornecendo uma explicação e elaboração dos resultados.
- Resultados de um teste de *baseline* que serve como instantâneo da performance do sistema em um determinado momento e forma a base de comparação com os testes subsequentes. Os resultados devem incluir a data/hora em que o teste foi iniciado, a meta do usuário simultâneo, a taxa de transferência medida e as principais conclusões. As principais conclusões podem incluir a taxa de erro global medida, o tempo de resposta e o rendimento médio.
- Um diagrama de alto nível mostrando todos os componentes de arquitetura que poderiam (ou tiveram) impacto nos objetivos do teste.
- Uma análise detalhada (tabelas e gráficos) dos resultados do teste, mostrando tempos de resposta, taxas de transação, taxas de erro e análise de performance. A análise também inclui uma descrição do que foi observado, como em que ponto um aplicativo estável se tornou instável e a origem das falhas (p. ex., servidor da Web, servidor de banco de dados).

Análise de log e dados armazenados

Um log de cada teste deve ser gravado. O log normalmente inclui:

- Data/hora do início do teste.
- Duração do teste.
- Scripts usados para teste (incluindo uma variedade de scripts se vários forem usados) e dados de configuração dos mais relevantes.
- Arquivos de dados de teste usados pelo teste.
- Nome e localização dos arquivos de dados/log criados durante o teste.
- Configuração de HW/SW testada (especialmente quaisquer alterações entre execuções).
- Média e pico de utilização de CPU e RAM em servidores da Web e de banco de dados.
- Notas sobre a performance alcançada.

- Defeitos identificados.

Recomendações

As recomendações resultantes dos testes podem incluir o seguinte:

- Alterações técnicas recomendadas, como a reconfiguração de hardware ou software ou infraestrutura de rede.
- Áreas identificadas para análise posterior (p. ex., análise de registros do servidor da web para ajudar a identificar as causas-raiz de problemas e/ou erros)
- Monitoramento adicional necessário de *gateways*, servidores e redes para que dados mais detalhados possam ser obtidos para medir-se as características e tendências de performance (p. ex., degradação)

5 Ferramentas [90 min]

Palavras-chave

gerador de carga, gerenciamento de carga, ferramenta de monitoramento, ferramenta de teste de performance.

Objetivos de Aprendizado

5.1 Ferramentas de Suporte

PTFL-5.1.1 (K2) Compreender como as ferramentas suportam testes de performance.

5.2 Avaliação de Ferramentas

PTFL-5.2.1 (K4) Avaliar a adequação de ferramentas de teste de performance em um determinado cenário de projeto.

5.1 Ferramentas de suporte

As ferramentas de teste de performance incluem os seguintes tipos de ferramentas para seu suporte.

Geradores de carga

O gerador, por meio de um IDE, editor de scripts ou conjunto de ferramentas, pode criar e executar várias instâncias do cliente que simulam o comportamento do usuário de acordo com um perfil operacional definido. Criar várias instâncias em curtos períodos causará carga em um sistema em teste. O gerador cria a carga e coleta métricas para relatórios posteriores.

Ao executar os testes de performance, o objetivo do gerador de carga é imitar o mundo real tanto quanto o é na prática. Isso geralmente significa que as solicitações do usuário são oriundas de vários locais e não apenas do local de teste. Ambientes configurados em vários pontos diferentes distribuirão a carga nesses pontos, de modo que nem tudo será proveniente de uma única rede. Isso fornece realismo ao teste, embora os resultados, às vezes, possam ficar distorcidos em casos de picos de rede intermediários criarem atrasos.

Console de gerenciamento de carga

A console de gerenciamento de carga fornece o controle para iniciar e parar o(s) gerador(es) de carga. A console também agrega métricas das várias transações definidas nas instâncias de carga usadas pelo gerador. A console permite que relatórios e gráficos das execuções de teste sejam visualizados e suporta a análise de resultados.

Ferramentas de monitoramento

As ferramentas de monitoramento são executadas simultaneamente com o componente ou sistema sob teste e supervisionam, registram e/ou analisam o comportamento do componente ou sistema. Os componentes típicos monitorados incluem filas do servidor da web, memória do sistema e espaço em disco. As ferramentas de monitoramento podem suportar efetivamente a análise da causa raiz da degradação da performance em um sistema em teste e podem ser usadas para monitorar um ambiente de produção quando o produto for lançado. Durante a execução do teste de performance, os monitores também podem ser usados no próprio gerador de carga.

Os modelos de licença para ferramentas de teste de performance incluem a licença tradicional baseada na instalação em uma rede local com propriedade total, um modelo de licença baseado em nuvem e licenças de código aberto que podem ser usadas em um ambiente definido ou por meio de nuvem. Cada modelo implica em uma estrutura de custos diferente e pode incluir manutenção contínua. O que está claro é que, para qualquer ferramenta selecionada, entender como essa ferramenta funciona (por meio de treinamento e/ou estudo individual) exigirá tempo e orçamento.

5.2 Avaliação de ferramentas

Os fatores a seguir devem ser considerados ao selecionar uma ferramenta de teste de performance:

Compatibilidade

Em geral, uma ferramenta é selecionada para a organização e não apenas para um projeto. Isso significa considerar os seguintes fatores na organização:

- **Protocolos**: conforme descrito no capítulo 4.2.1, os protocolos são aspectos muito importantes para a seleção de ferramentas de performance. Entender quais protocolos um sistema usa e quais deles serão testados fornecerá as informações necessárias para avaliar a ferramenta de teste apropriada.
- **Interfaces para componentes externos**: interfaces para componentes de software ou outras ferramentas podem ser consideradas como parte dos requisitos completos de integração para atender aos requisitos de processo ou outros requisitos de interoperabilidade (p. ex., integração no processo de IC).
- **Plataformas**: a compatibilidade com as plataformas (e suas versões) dentro de uma organização é essencial. Isso se aplica às plataformas usadas para hospedar as ferramentas e as plataformas com as quais as ferramentas interagem para monitoramento e/ou geração de carga.

Escalabilidade

Outro fator a considerar é o número total de simulações de usuários simultâneos que a ferramenta pode manipular. Isso incluirá vários fatores:

- Número máximo de licenças necessárias.
- Requisitos de configuração da estação de trabalho de geração de carga/servidor.
- Capacidade de gerar carga a partir de vários locais (p. ex., servidores distribuídos).

Compreensibilidade

Outro fator a considerar é o nível de conhecimento técnico necessário para usar a ferramenta. Isso geralmente é ignorado e pode levar os testadores não qualificados a configurar incorretamente os testes, o que, por sua vez, fornece resultados imprecisos. Para testes que exigem cenários complexos e um alto nível de programação e personalização, as equipes devem garantir que o testador tenha as habilidades necessárias, experiência e treinamento.

Monitoramento

O monitoramento fornecido pela ferramenta é suficiente? Existem outras ferramentas de monitoramento disponíveis no ambiente que podem ser usadas para complementar o monitoramento pela ferramenta? O monitoramento pode ser correlacionado às transações definidas? Todas essas perguntas devem ser respondidas para determinar se a ferramenta fornecerá o monitoramento exigido pelo projeto.

Quando o monitoramento é um programa separado/ferramentas/pilha acumulada, ele pode ser usado para monitorar o ambiente de produção quando o produto é liberado.

Referências

Normas e padrões

[ISO25000] ISO/IEC 25000:2005, Software Engineering - Software Product Quality Requirements and Evaluation (SQuaRE)

Documentos ISTQB/BSTQB

[ISTQB_UT_SYL] ISTQB Foundation Level Usability Testing Syllabus, Version 2018

[ISTQB_ELTM_ITP_SYL] Expert Level Syllabys - Improving the Testing Process

[ISTQB_ALTA_SYL] ISTQB Advanced Level Test Analyst Syllabus, Version 2012

[ISTQB_ALTTA_SYL] ISTQB Advanced Level Technical Test Analyst Syllabus, Version 2012

[ISTQB_ALTM_SYL] ISTQB Advanced Level Test Manager Syllabus, Version 2012

[ISTQB_FL_SYL] ISTQB Foundation Level (Core) Syllabus, Version 2018

[ISTQB_FL_AT] ISTQB Foundation Level Agile Tester Syllabus, Version 2014

[ISTQB_GLOSSARY] ISTQB Glossary of Terms used in Software Testing, (http://glossary.istqb.org)

Literatura

[Anderson01] Lorin W. Anderson, David R. Krathwohl (eds.) “A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom’s Taxonomy of Educational Objectives”, Allyn & Bacon, 2001, ISBN 978-0801319037

[Bath14] Graham Bath, Judy McKay, “The Software Test Engineer’s Handbook”, Rocky Nook, 2014, ISBN 978-1-933952-24-6

[Molyneaux09] Ian Molyneaux, “The Art of Application Performance Testing: From Strategy to Tools”, O’Reilly, 2009, ISBN: 9780596520663

[Microsoft07] Microsoft Corporation, “Performance Testing Guidance for Web Applications”, Microsoft, 2007, ISBN: 9780735625709