International Software Testing Qualifications Board

Testeur certifié

Syllabus, spécialiste du Niveau Fondation
Tests de performance

Version 2018

Fourni par

American Software Testing Qualifications Board

et

German Testing Board

Avis de copyright

Groupe de travail sur les tests de performance :

Graham Bath Rex
Black
Alexander Podelko
Andrew Pollner
Randy Rice

Historique de révision

Version	Date	Remarks
Alpha V04	13 décembre 2016	Version pour NYC Meeting
Alpha V05	18 décembre 2016	Après NYC Meeting
Alpha V06	23 décembre 2016	Restructuration du Ch 4 Renumérotation et ajustement des LOs comme approuvé à NYC
Alpha V07	31 décembre 2016	Ajout des commentaires d'auteurs
Alpha V08	12 février 2017	Version Pre-Alpha
Alpha V09	16 avril 2017	Version Pre-Alpha
Alpha Review V10	28 juin 2017	Pour revue Alpha
V2017v1	27 novembre 2017	Pour revue Alpha
V2017v2	15 décembre 2017	Mise à jour Alpha
V2017v3	15 janvier 2018	Edition technique
V2017v4	23 janvier 2018	Revue pour glossaire
V2018 b1	1 mars 2018	Candidat Bêta pour l'ISTQB
V2018 b2	17 mai 2018	Version Bêta pour l'ISTQB
V2018 b3	25 août 2018	Commentaires de la revue Bêta ajoutés à la version finale
Version 2018	9 décembre 2018	Version ISTQB GA

Table des matières

Remerciements

Ce document a été produit par l'American Software Testing Qualifications Board (ASTQB) et le German Testing Board (GTB):

Graham Bath (GTB, Coprésident du groupe de travail)
Rex Black
Alexander Podelko (CMG)
Andrew Pollner (ASTQB, Coprésident du Groupe de travail)
Randy Rice

L'équipe des auteurs remercie l'équipe de revue pour ses suggestions et ses commentaires. ASTQB tient à remercier le Computer Measurement Group (CMG) pour leur contribution à l'élaboration de ce syllabus.

Les personnes suivantes ont participé à l'examen, aux commentaires ou au vote de ce syllabus ou de ses prédécesseurs :

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

Ce document a été officiellement publié par l'ISTQB le 9 décembre 2018.

0. Introduction à ce syllabus

0.1 Objectif de ce document

Ce syllabus constitue la base de la qualification des tests de performance au Niveau Fondation. L'ASTQB® et le GTB® fournissent ce syllabus comme suit :

1. Aux bureaux nationaux de l'ISTQB, pour le traduire dans leur langue locale et accréditer les prestataires de formation. Les bureaux nationaux de l'ISTQB peuvent adapter le syllabus à leurs besoins linguistiques particuliers et modifier les références pour s'adapter à leurs publications locales.
2. Aux organismes de certification, afin d'obtenir des questions d'examen dans leur langue locale adaptées aux objectifs d'apprentissage de chaque syllabus.
3. Aux organismes de formation, afin de produire des cours et déterminer les méthodes d'enseignement appropriées.
4. Aux candidats à la certification, afin de se préparer à l'examen (dans le cadre d'une formation ou de manière indépendante).
5. À la communauté internationale de l'ingénierie des logiciels et des systèmes, pour faire progresser la profession du test des logiciels et des systèmes, et comme base pour les livres et les articles.

L'ASTQB® et le GTB® peuvent permettre à d'autres entités d'utiliser ce syllabus à d'autres fins, sous condition qu'elles aient obtenu une autorisation écrite préalable.

0.2 La certification de Niveau Fondation des tests de performance

La certification de Niveau Fondation s'adresse à toute personne impliquée dans les tests logiciels qui souhaite élargir ses connaissances en tests de performance ou toute personne souhaitant commencer une carrière de spécialiste dans les tests de performance. La qualification s'adresse également à toute personne impliquée dans l'ingénierie de la performance qui souhaite mieux comprendre les tests de performance.

Le syllabus traite des principaux aspects suivants des tests de performance :

- Aspects techniques
- Aspects basés sur la méthode
- Aspects organisationnels

Les informations sur les tests de performance décrites dans le syllabus ISTQB® Analyste de Test Technique de Niveau Avancé [ISTQB_ALTTA_SYL] est compatible avec ce syllabus et y est développé.

0.3 Objectifs-Métier

Cette section énumère les objectifs-métier attendus d'un candidat qui a obtenu la certification de test de performance de Niveau Fondation.

PTFL-1 Comprendre les concepts de base de l'efficacité de la performance et des tests de performance
PTFL-2 Définir les risques, les objectifs et les exigences en matière de performance pour répondre aux besoins et aux attentes des parties prenantes
PTFL-3 Comprendre les mesures de performance et la façon de les recueillir
PTFL-4 Élaborer un plan de test de performance pour atteindre les objectifs et les exigences énoncés
PTFL-5 Concevoir, mettre en œuvre et exécuter des tests de performance de base
PTFL-6 Analyser les résultats d'un test de performance et établir les implications pour diverses parties prenantes
PTFL-7 Expliquer le processus, la justification, les résultats et les implications des tests de performance à diverses parties prenantes
PTFL-8 Comprendre les catégories et les utilisations des outils de performance et leurs critères de sélection
PTFL-9 Déterminer comment les activités de test de performance s'alignent avec le cycle de vie du logiciel

0.4 Objectifs d'apprentissage examinables

Les objectifs d'apprentissage soutiennent les objectifs-métier et sont utilisés pour créer l'examen d'obtention de la certification de test de performance de Niveau Fondation. Les objectifs d'apprentissage sont associés à un niveau cognitif de connaissances (Niveau K).

Un niveau K, ou niveau cognitif, est utilisé pour classer les objectifs d'apprentissage en fonction de la taxonomie révisée de Bloom [Anderson01]. ISTQB® utilise cette taxonomie pour concevoir les examens de ses syllabi.

Ce syllabus tient compte de quatre niveaux K différents (K1 à K4):

Niveau K	**Mot-Clé**	**Description**
1	Se souvenir	Le candidat doit se souvenir ou reconnaître un terme ou un concept.

2	Comprendre	Le candidat doit choisir une explication pour une déclaration liée au sujet de la question.
3	Appliquer	Le candidat doit choisir l'application correcte d'un concept ou d'une technique et l'appliquer à un contexte donné.
4	Analyser	Le candidat peut séparer les informations relatives à une procédure ou à une technique dans ses parties constitutives pour mieux comprendre et faire la distinction entre les faits et les inférences.

En général, toutes les parties de ce programme sont examinables à un niveau K1. C'est-à-dire que le candidat reconnaîtra, se remémorera et se souviendra d'un terme ou d'un concept. Les objectifs d'apprentissage aux niveaux K2, K3 et K4 sont affichés au début du chapitre pertinent.

0.5 Temps de formation recommandés

Un temps de formation minimum a été défini pour chaque objectif d'apprentissage dans ce syllabus. Le temps total pour chaque chapitre est indiqué dans le titre du chapitre.

Les formateurs doivent noter que d'autres syllabus de l'ISTQB appliquent une approche de « temps standard » qui alloue des temps fixes selon le niveau K. Le syllabus Tests de Performance n'applique pas strictement cette règle. Par conséquent, les formateurs reçoivent une indication plus souple et réaliste des temps de formation minimaux pour chaque objectif d'apprentissage.

0.6 Exigences d'entrée

La certification de base au Niveau Fondation est requise avant de passer l'examen de certification de Test de Performance de Niveau Fondation.

0.7 Sources d'information

Les termes utilisés dans le syllabus sont définis dans le Glossaire des termes utilisés dans les tests logiciels de l'ISTQB [ISTQB_GLOSSARY].

La section 6 contient une liste de livres et d'articles recommandés sur les tests de performance.

1. Concepts de base – 60 mins.

Mots-clés

Tests de capacité, efficacité des tests de concurrence, tests d'endurance, génération de charge, tests de charge, tests de performance, tests de passage à l'échelle, test de pic (de charge), test de stress

Objectifs d'apprentissage pour les concepts de base

1.1 Principes et concepts
PTFL-1.1.1 (K2) Comprendre les principes des tests de performance

1.2 Types de tests de performance
PTFL-1.2.1 (K2) Comprendre les différents types de tests de performance

1.3 Types de tests dans les tests de performance
PTFL-1.3.1 (K1) Rappeler les types de tests (utilisés) dans les tests de performance

1.4 Le concept de génération de charge
PTFL-1.4.1 (K2) Comprendre le concept de génération de charge

1.5 Échecs courants dans les tests de performance et leurs causes
PTFL-1.5.1 (K2) Donner des exemples de modes d'échec courants des tests de performance et leurs causes

1.1 Principes des tests de performance

L'efficacité des performances (ou simplement la « performance ») est un élément essentiel de la « bonne expérience » fournie aux utilisateurs lorsqu'ils utilisent leurs applications sur une variété de plates-formes fixes et mobiles. Les tests de performance jouent un rôle essentiel dans l'établissement de niveaux de qualité acceptables pour l'utilisateur final et sont souvent étroitement intégrés à d'autres disciplines telles que l'ingénierie de l'utilisabilité et l'ingénierie de la performance.

En outre, l'évaluation de l'adéquation fonctionnelle, de la convivialité et d'autres caractéristiques de qualité dans des conditions de charge, comme lors de l'exécution d'un test de performance, peut révéler des problèmes spécifiques à la charge qui ont une incidence sur ces caractéristiques.

Les tests de performance ne se limitent pas au domaine Web où l'utilisateur final est au centre de l'attention. Ils sont également pertinents pour différents domaines d'application avec une variété d'architectures système, telles que le client-serveur classique, les architectures distribuées ou embarquées. Techniquement, l'efficacité de la performance est classée dans le modèle de Qualité des Produits ISO 25010 [ISO25000] en tant que caractéristique de qualité non fonctionnelle et comprend les trois sous-caractéristiques décrites ci-dessous. L'orientation et la priorisation adéquates dépendent des risques évalués et des besoins des diverses parties prenantes. L'analyse des résultats des tests peut identifier d'autres domaines de risque qui doivent être abordés.

Comportement dans le temps : En général, l'évaluation du comportement dans le temps est l'objectif le plus courant des tests de performance. Cet aspect des tests de performance examine la capacité d'un composant ou d'un système à répondre aux entrées de l'utilisateur ou du système dans un délai déterminé et dans des conditions spécifiées. Les mesures du comportement dans le temps peuvent varier du temps pris par le système, de bout en bout, pour répondre à l'entrée de l'utilisateur, au nombre de cycles de processeur (CPU) requis par un composant logiciel pour exécuter une tâche particulière.

Utilisation des ressources : Si la disponibilité des ressources du système est identifiée comme un risque, l'utilisation de ces ressources (p. ex., l'attribution de RAM limitée) peut être étudiée en effectuant des tests de performance spécifiques.

Capacité : Si les problèmes de comportement du système aux limites de capacité requises (p. ex. le nombre d'utilisateurs ou des volumes de données) sont identifiés comme un risque, des tests de performance peuvent être effectués pour évaluer la pertinence de l'architecture du système.

Les tests de performance prennent souvent la forme d'expérimentations, ce qui permet de mesurer et d'analyser des paramètres spécifiques du système. Ceux-ci peuvent être menés de façon itérative à l'appui de l'analyse, de la conception et de la mise en œuvre du système afin de permettre de prendre des décisions en matière d'architecture et d'aider à façonner les attentes des parties prenantes.

Les principes suivants de test de performance sont particulièrement pertinents.

- Les tests doivent être alignés sur les attentes définies des différents groupes de parties prenantes, en particulier les utilisateurs, les concepteurs de systèmes et le personnel des opérations.
- Les tests doivent être reproductibles. Des résultats statistiquement identiques (dans une tolérance spécifiée) doivent être obtenus en répétant les tests sur un système inchangé.

- Les tests doivent donner des résultats qui sont à la fois compréhensibles et qui peuvent être facilement comparés aux attentes des parties prenantes.
- Les tests peuvent être effectués, lorsque les ressources le permettent, soit sur des systèmes complets ou partiels, soit sur des environnements de test représentatifs du système de production.
- Les tests doivent être pratiquement abordables et exécutables dans les délais fixés par le projet.

Les livres de [Molyneaux09] et [Microsoft07] fournissent un contexte solide aux principes et aux aspects pratiques des tests de performance.

Les trois sous-caractéristiques de qualité ci-dessus auront une incidence sur la capacité du système sous test (SST) d'être mis à l'échelle.

1.2 Types de tests de performance

Différents types de tests de performance peuvent être définis. Chacun d'eux peut s'appliquer à un projet donné, selon les objectifs du test.

Tests de performance

Le test de performance est un terme générique comprenant tout type de test axé sur les performances (réactivité) du système ou du composant sous différents volumes de charge.

Test de charge

Les tests de charge mettent l'accent sur la capacité d'un système à répondre aux niveaux croissants de charges, réalistes et prévues, résultant des demandes de transactions générées par un nombre contrôlé d'utilisateurs ou de processus simultanés.

Tests de stress

Les tests de stress mettent l'accent sur la capacité d'un système ou d'un composant à traiter les pics de charges qui atteignent la limite ou sont au-delà des limites des charges de travail prévues ou spécifiées. Les tests de stress sont également utilisés pour évaluer la capacité d'un système à gérer la disponibilité réduite des ressources telles que la capacité de calcul, la bande passante et la mémoire disponibles.

Tests de passage à l'échelle (NDT : scalabilité ou évolutivité)

Les tests de passage à l'échelle mettent l'accent sur la capacité d'un système à répondre aux exigences futures en matière d'efficacité qui pourraient aller au-delà de

celles actuellement requises. L'objectif de ces tests est de déterminer la capacité de croissance du système (p. ex., avec un plus grand nombre d'utilisateurs, de plus grandes quantités de données stockées) sans violer les exigences de performance spécifiées actuellement ou générer une défaillance. Une fois que les limites du passage à l'échelle sont connues, les valeurs seuils peuvent être fixées et surveillées en production afin de fournir un avertissement sur les problèmes éventuellement sur le point de se poser. En outre, l'environnement de production peut être ajusté avec des quantités appropriées de matériel.

Test de pics

Le test de pics de charge se concentre sur la capacité d'un système à répondre correctement aux pics de charges soudains et de revenir par la suite à un état stable.

Tests d'endurance

Les tests d'endurance se concentrent sur la stabilité du système durant un laps de temps spécifique au contexte opérationnel du système. Ce type de test vérifie qu'il n'y a pas de problèmes de capacité des ressources (p. ex., fuites de mémoire, connexions de base de données, groupes d'unités d'exécution (NDT : pools de threads)) qui peuvent éventuellement dégrader les performances et/ou causer des défaillances aux points de rupture.

Tests de concurrence

Les tests de concurrence se concentrent sur l'impact des situations où des actions spécifiques se produisent simultanément (p. ex., lorsqu'un grand nombre d'utilisateurs se connectent en même temps). Les problèmes de concurrence sont notoirement difficiles à trouver et à reproduire, en particulier lorsque le problème se produit dans un environnement où les testeurs ont peu ou pas de contrôle, comme la production.

Tests de capacité

Les tests de capacité déterminent le nombre d'utilisateurs et/ou de transactions qu'un système donné supportera tout en respectant les objectifs de performance énoncés. Ces objectifs peuvent également être énoncés en ce qui concerne les volumes de données résultant des transactions.

1.3 Types de tests dans les tests de performance

Les principaux types de tests utilisés dans les tests de performance comprennent les tests statiques et les tests dynamiques.

1.3.1 Tests statiques

Les activités de tests statiques sont souvent plus importantes pour les tests de performance que pour les tests de pertinence fonctionnelle. Cela se justifie parce que de nombreux défauts de performance critiques sont introduits dans l'architecture et la conception du système. Ces défauts peuvent être introduits par les concepteurs et les architectes en raison de malentendus ou un manque de connaissances. Ces défauts peuvent également être introduits parce que les exigences n'ont pas suffisamment saisi les objectifs de temps de réponse, de débit ou d'utilisation des ressources ; la charge et l'utilisation prévues du système ; ou encore les contraintes.

Les activités de test statiques pour la performance peuvent inclure :

- Revues des exigences en mettant l'accent sur les aspects et les risques liés à la performance
- Revues des schémas de base de données, diagrammes d'entités-relations, métadonnées, procédures et requêtes stockées
- Revues du système et de l'architecture réseau
- Revues des segments critiques du code du système (p. ex., algorithmes complexes)

1.3.2 Tests dynamiques

Au fur et à mesure que le système se construit, les tests de performance dynamiques devraient commencer, dès que possible.
Les opportunités de tests de performance dynamiques comprennent :

- Pendant les tests unitaires, y compris l'utilisation de l'information de profilage pour déterminer les goulots d'étranglement potentiels et l'analyse dynamique pour évaluer l'utilisation des ressources
- Pendant les tests d'intégration des composants, au travers des cas d'utilisation clés et des flux de travail, en particulier lors de l'intégration de différentes caractéristiques de cas d'utilisation ou de l'intégration à la structure de base (NDT : backbone structure) d'un flux de travail
- Pendant les tests système des comportements globaux de bout en bout dans diverses conditions de charge
- Pendant les tests d'intégration des systèmes, en particulier pour les flux de données et les flux de travail au travers des interfaces entre systèmes clés. Dans les tests d'intégration du système il n'est pas rare que l' «utilisateur» soit un autre système ou une machine (par exemple, des entrées provenant de capteurs et d'autres systèmes)
- Pendant les tests d'acceptation, pour renforcer la confiance de l'utilisateur, du client et de l'opérateur, dans les performances appropriées du système et pour ajuster le système dans des conditions réelles (mais généralement pas pour trouver des défauts de performance dans le système)

Dans les niveaux de test plus élevés tels que les tests système et les tests d'intégration du système, l'utilisation d'environnements, de données et de charges réalistes sont essentielles pour des résultats précis (voir le chapitre 4). Dans les cycles de vie Agile et autres cycles de vie itératifs, les équipes devraient intégrer des tests de performance statiques et dynamiques dans les itérations précoces plutôt que d'attendre les itérations finales pour faire face aux risques de performance.

Si du matériel personnalisé ou nouveau fait partie du système, les premiers tests de performance dynamique peuvent être effectués à l'aide de simulateurs. Cependant, il est bon de commencer à tester sur le matériel réel dès que possible, car souvent les simulateurs ne saisissent pas adéquatement les contraintes de ressources et les comportements liés aux performances.

1.4 Le concept de génération de charge

Pour effectuer les divers types de tests de performance décrits à la section 1.2, les charges représentatives du système doivent être modelisées, générées et soumises au système sous test. Les charges sont comparables aux entrées de données utilisées pour les cas de test fonctionnels, mais diffèrent principalement comme suit :

- Un test de charge doit représenter de nombreuses entrées d'utilisateurs, et pas seulement une seule
- Un test de charge peut nécessiter du matériel et des outils dédiés à sa génération
- La génération d'd'un test de charge dépend de l'absence de défauts fonctionnels dans le système sous test, qui peuvent avoir un impact sur l'exécution du test

La génération efficace et fiable d'une charge spécifiée est un facteur de réussite clé lors des tests de performance. Il existe différentes options pour la génération de tests de charge.

Génération de tests de charge via l'interface utilisateur

Il peut s'agir d'une approche adéquate si seulement un petit nombre d'utilisateurs doivent être représentés et si le nombre requis de clients logiciels pour entrer les entrées requises est disponible. Cette approche peut également être utilisée en conjonction avec des outils fonctionnels d'exécution des tests, mais peut rapidement devenir peu pratique à mesure que le nombre d'utilisateurs à simuler augmente. La stabilité de l'interface utilisateur (UI) représente également une dépendance critique. Des changements fréquents peuvent avoir une incidence sur la répétabilité des tests de performance et peuvent impacter de manière importante les coûts de maintenance. Les tests effectués par l'interface utilisateur peuvent être l'approche la plus représentative pour les tests de bout en bout.

Génération de tests de charge utilisant la participation de masse (NDT : Crowd)
Cette approche dépend de la disponibilité d'un grand nombre de testeurs qui représenteront de vrais utilisateurs. Lors des tests de masse, les testeurs sont organisés de manière que la charge désirée puisse être générée. Il peut s'agir d'une méthode appropriée pour tester des applications accessibles de n'importe où dans le monde (p. ex., sur le Web) et peut impliquer que les utilisateurs génèrent une charge à partir d'un large éventail d'appareils de différents types et configurations. Bien que cette approche puisse permettre d'utiliser un très grand nombre d'utilisateurs, la charge générée ne sera pas aussi reproductible et précise que d'autres options et est plus complexe à organiser.

Génération de charge via l'interface de programmation d'applications (API)
Cette approche est similaire à l'utilisation de l'interface utilisateur pour la saisie des données, mais utilise l'API de l'application au lieu de l'interface utilisateur pour simuler l'interaction de l'utilisateur avec le système sous test. L'approche est donc moins sensible aux modifications (p. ex., retards) de l'interface utilisateur et permet de traiter les transactions de la même manière que si elles étaient saisies directement par un utilisateur via l'UI. Des scripts dédiés peuvent être créés qui appellent, de manière répétée, des routines spécifiques d'API et permettent à plus d'utilisateurs d'être simulés par rapport à l'utilisation d'entrées d'interface utilisateur.

Génération de tests de charge utilisant des protocoles de communication capturés
Cette approche consiste à capturer l'interaction de l'utilisateur avec le système sous test au niveau du protocole de communication, puis à rejouer ces scripts pour simuler un nombre potentiellement très important d'utilisateurs d'une manière répétable et fiable. Cette approche fondée sur des outils est décrite dans les sections 4.2.6 et 4.2.7.

1.5 Problèmes courants de performance et leurs causes

Bien que l'on puisse certainement trouver de nombreux modes d'échec de performance différents lors des tests dynamiques, les quelques exemples qui suivent représentent des défaillances courantes (y compris les crashes du système), ainsi que des causes typiques :

Réponse lente avec tous les niveaux de charge
Dans certains cas, la réponse est inacceptable, quelle que soit la charge. Cela peut être causé par des problèmes de performance sous-jacents, y compris, mais non limité à, la mauvaise conception ou mise en œuvre de la base de données, la latence du réseau, et d'autres charges d'arrière-plan. Ces problèmes peuvent être identifiés lors

des tests fonctionnels et d'utilisabilité, et donc pas seulement durant les tests de performance, de sorte que les analystes de test devraient rester vigilants et les signaler.

Réponse lente avec des niveaux de charge modérés à lourds
Dans certains cas, la réponse se dégrade de façon inacceptable avec une charge modérée à lourde, même lorsque ces charges sont entièrement dans les plages normales, attendues et autorisées. Les défauts sous-jacents comprennent la saturation d'une ou plusieurs ressources et des charges variables en arrière-plan.

Réponse dégradée au fil du temps
Dans certains cas, la réponse se dégrade progressivement ou sévèrement au fil du temps. Les causes sous-jacentes comprennent les fuites de mémoire, la fragmentation du disque, l'augmentation de la charge réseau au fil du temps, la croissance du répertoire de fichiers et la croissance inattendue de la base de données.

Gestion d'erreur inadéquate ou inappropriée avec une charge lourde ou dépassant les limites
Dans certains cas, le temps de réponse est acceptable, mais le traitement des erreurs se dégrade à des niveaux de charge élevés et au-delà de la limite. Les défauts sous-jacents comprennent l'insuffisance des pools de ressources, des files d'attente et des piles sous-dimensionnées, et des paramètres de temporisation trop courts.

La liste qui suit reprend des exemples spécifiques de types d'échecs généraux énumérés ci-dessus :

- Une application Web qui fournit des informations sur les services d'une entreprise ne répond pas aux demandes des utilisateurs dans les sept secondes (une règle générale de base de l'industrie). L'efficacité des performances du système ne peut pas être atteinte dans des conditions de charge spécifiques.
- Un système se bloque ou n'est pas en mesure de répondre aux entrées des utilisateurs lorsqu'il est soudainement soumis à un grand nombre de demandes d'utilisateurs (p. ex., la vente de billets pour un événement sportif majeur). La capacité du système à traiter ce nombre d'utilisateurs est inadéquate.
- La réponse du système est considérablement dégradée lorsque les utilisateurs soumettent des demandes de grandes quantités de données (p. ex., un rapport volumineux et important est affiché sur un site Web pour téléchargement). La capacité du système à gérer les volumes de données générés est insuffisante.
- Le traitement par lots (NDT : batch) n'est pas en mesure de se terminer avant que le traitement en ligne ne soit requis. Le temps d'exécution des processus par lots est insuffisant pour la période autorisée.

- Un système en temps réel est à court de RAM lorsque les processus parallèles génèrent de grandes demandes de mémoire dynamique qui ne peuvent pas être libérées à temps. La RAM n'est pas dimensionnée de manière adéquate, ou les demandes de RAM ne sont pas correctement priorisées.
- Un composant système en temps réel A qui fournit des entrées au composant B du système en temps réel n'est pas en mesure de calculer les mises à jour à la vitesse requise. L'ensemble du système ne répond pas à temps et peut échouer. Les modules de code dans le composant A doivent être évalués et modifiés (« profilage des performances ») afin de s'assurer que les taux de mise à jour requis puissent être atteints.

2. Fondamentaux de mesure de la performance - 55 mins.

Mots-clés

Mesure, métrique

Objectifs d'apprentissage pour les fondamentaux des mesures de la performance

2.1 Métriques typiques recueillies dans les tests de performance
PTFL-2.1.1 (K2) Comprendre les métriques typiques recueillies dans les tests de performance

2.2 Agréger les résultats des tests de performance
PTFL-2.2.1 (K2) Expliquer pourquoi les résultats des tests de performance sont agrégés

2.3 Sources clés de métriques de performance
PTFL-2.3.1 (K2) Comprendre les principales sources de métriques de performance

2.4 Résultats typiques d'un test de performance
PTFL-2.4.1 (K1) Rappeler les résultats typiques d'un test de performance

2.1 Métriques typiques recueillies dans les tests de performance

2.1.1 Pourquoi des métriques de performance sont nécessaires

Des mesures précises et les métriques qui sont dérivées de ces mesures sont essentielles pour définir les objectifs des tests de performance et pour évaluer les résultats des tests de performance. Les tests de performance ne doivent pas être effectués sans d'abord comprendre quelles mesures et métriques sont nécessaires. Les risques suivants s'appliquent au projet si ce conseil est ignoré :

- On ne sait pas si les niveaux de performance sont acceptables pour atteindre les objectifs opérationnels
- Les exigences de performance ne sont pas définies en termes mesurables
- Il n'est peut-être pas possible d'identifier les tendances qui peuvent prédire des niveaux de performance plus faibles
- Les résultats réels d'un test de performance ne peuvent pas être évalués en les comparant à un ensemble de mesures de base qui définissent une performance acceptable et/ou inacceptable

- Les résultats des tests de performance sont évalués en fonction de l'opinion subjective d'une ou de plusieurs personnes
- Les résultats fournis par un outil de test de performance ne sont pas compris

2.1.2 Collecte des mesures et des métriques de performance

Comme avec n'importe quelle forme de mesure, il est possible d'obtenir et d'exprimer des métriques de manière précise. Par conséquent, l'une ou l'autre des métriques et mesures décrites dans cette section peut et doit être définie comme significative dans un contexte particulier. Il s'agit d'effectuer des tests initiaux et d'apprendre quelles métriques doivent être encore affinées et qui doivent être ajoutées.

Par exemple, la métrique du temps de réponse sera probablement présente dans n'importe quel ensemble de métriques de performance. Toutefois, pour être significative et réalisable, la métrique du temps de réponse devra être mieux définie en précisant l'heure de la journée, le nombre d'utilisateurs simultanés, la quantité de données traitées et ainsi de suite.

Les métriques recueillies dans le cas d'un test de performance spécifique varieront en fonction :

- Du contexte métier (processus métier, comportement des clients et des utilisateurs, attentes des parties prenantes),
- Du contexte opérationnel (technologie et comment elle est utilisée)
- Des objectifs de test

Par exemple, les métriques choisies pour les tests de performance d'un site Web d'e-commerce international seront différentes de celles choisies pour le test de performance d'un système embarqué utilisé pour contrôler la fonctionnalité des dispositifs médicaux.

Une façon courante de classer les mesures et les métriques de performance est de tenir compte de l'environnement technique, de l'environnement métier ou de l'environnement opérationnel dans lequel l'évaluation de la performance est nécessaire.

Les catégories de mesures et de métriques décrites ci-dessous sont celles couramment obtenues à partir des tests de performance.

Environnement technique

Les métriques de performance varient selon le type d'environnement technique, comme le montre la liste suivante :

- Sur le Web

- Mobile
- Objets connectés (Internet-of-Things (IoT))
- Périphériques de bureau
- Traitement côté serveur
- Mainframe
- Bases de données
- Réseaux
- La nature des logiciels en cours d'exécution dans l'environnement (p. ex., embarqué)

Les métriques comprennent :

- Temps de réponse (p. ex., par transaction, par utilisateur concurrent, temps de chargement de page)
- Utilisation des ressources (p. ex., processeur, mémoire, bande passante réseau, latence réseau, espace disque disponible, taux I/O, tâches (threads) inactives et occupées)
- Capacité de débit (NDT : throughput rate) de la transaction clé (c.-à-d. le nombre de transactions qui peuvent être traitées dans un délai donné)
- Temps de traitement par lots (p. ex., temps d'attente, temps de traitement (NDT : throughput time), temps de réponse de base de données, temps d'achèvement)
- Nombre d'erreurs ayant une incidence sur les performances
- Temps d'achèvement (p. ex., pour la création, la lecture, la mise à jour et la suppression des données)
- Charge d'arrière-plan sur les ressources partagées (en particulier dans les environnements virtualisés)
- Métriques logicielles (p. ex., complexité du code)

Environnement métier

Du point de vue fonctionnel ou métier, les métriques de performance peuvent inclure :

- Efficacité des processus métier (p. ex., la rapidité d'exécution d'un processus métier global, y compris les flux de cas d'utilisation normaux, alternatifs et exceptionnels)
- Débit (NDT : throughput) de données, transactions et autres unités de travail effectuées (p. ex., commandes traitées par heure, lignes de données ajoutées par minute)
- Taux de conformité ou de violation du Service Level Agreement (SLA) (p. ex., violations du SLA par unité de temps)
- Étendue d'utilisation (p. ex., pourcentage d'utilisateurs mondiaux ou nationaux effectuant des tâches à un moment donné)

- Concurrence de l'utilisation (p. ex., le nombre d'utilisateurs effectuant simultanément une tâche)
- Timing d'utilisation (p. ex., le nombre de commandes traitées pendant les périodes de pointe)

Environnement opérationnel
L'aspect opérationnel des tests de performance se concentre sur des tâches qui ne sont généralement pas considérées comme étant, par nature, destinée aux utilisateurs finaux. Il s'agit notamment des :

- Processus opérationnels (p. ex., le temps requis pour le démarrage de l'environnement, les sauvegardes, les arrêts et les temps de reprise)
- Restauration du système (p. ex., le temps nécessaire pour restaurer les données d'une sauvegarde)
- Alertes et avertissements (p. ex., le temps nécessaire pour que le système émette une alerte ou un avertissement)

2.1.3 Sélection des métriques de performance

Il convient de noter que la collecte de plus de métriques que nécessaire n'est pas forcément une bonne chose. Chaque métrique choisie nécessite un moyen de collecte et de rapports cohérents. Il est important de définir un ensemble de métriques que l'on peut obtenir et qui appuient les objectifs de test de performance.

Par exemple, l'approche objectif-question-métrique (NDT : Goal-Question-Metric ou GQM) est un moyen utile d'aligner les métriques sur les objectifs de performance. L'idée est d'abord d'établir les objectifs, puis de poser des questions pour savoir quand les objectifs ont été atteints. Les métriques sont associées à chaque question pour s'assurer que la réponse à la question est mesurable. (Voir Section 4.3 de Expert Level Syllabus – Improving the Testing Process [ISTQB_ELTM_ITP_SYL] pour une description plus complète de l'approche GQM.) Il est à noter que l'approche GQM ne correspond pas toujours au processus de test de performance. Par exemple, certaines mesures représentent la santé d'un système et ne sont pas directement liées aux objectifs.

Il est important de se rendre compte qu'après la définition et la capture des mesures initiales, d'autres mesures et métriques peuvent être nécessaires pour comprendre les niveaux de performance réels et déterminer où des actions correctives peuvent être nécessaires.

2.2 Agréger les résultats des tests de performance

Le but de l'agrégation des mesures de performance est d'être en mesure de les comprendre et de les exprimer d'une manière qui traduit avec précision l'image globale des performances du système. Lorsque les mesures de performance ne sont considérées qu'au niveau détaillé, il peut être difficile de tirer la bonne conclusion, surtout pour les parties prenantes issues du métier.

Pour de nombreuses parties prenantes, la principale préoccupation est que le temps de réponse d'un système, d'un site Web ou d'un autre objet de test se situe dans des limites acceptables.

Une fois qu'une meilleure compréhension des métriques de performance a été atteinte, celles-ci peuvent être agrégées de sorte que :

- Les parties prenantes issues du métier et des projets peuvent voir l'état général de la performance du système
- Les tendances de performance peuvent être identifiées
- Les mesures de performance peuvent être rapportées de façon compréhensible

2.3 Sources clés de métriques de performance

Les performances du système ne devraient être que peu impactées par l'effort de collecte des métriques (connu sous le nom d'« effet de sonde »). En outre, le volume, la précision et la vitesse avec lesquelles les métriques de performance doivent être recueillies font de l'utilisation des outils une exigence. Bien que l'utilisation combinée d'outils ne soit pas rare, elle peut introduire une redondance dans l'utilisation d'outils de test et d'autres problèmes (voir Section4.4).

Il existe trois sources clés de métriques de performance :

Outils de test de performance
Tous les outils de test de performance fournissent des mesures et des métriques à la suite d'un test. Les outils peuvent varier en fonction du nombre de métriques indiquées, de la façon dont les métriques sont affichées et de la capacité de l'utilisateur à personnaliser les métriques à une situation particulière (voir aussi Section 5.1).

Certains outils recueillent et affichent des métriques de performance au format texte, tandis que des outils plus performants recueillent et affichent des mesures de performance graphiquement en forme de tableau de bord. De nombreux outils offrent la possibilité d'exporter des métriques pour faciliter l'évaluation des tests et les rapports.

Outils de surveillance de la performance
Des outils de suivi de la performance sont souvent utilisés pour compléter les capacités de reporting des outils de tests de performance (voir aussi Section 5.1). En outre, des outils de surveillance peuvent être utilisés pour surveiller les performances du système de façon continue et pour alerter les administrateurs du système sur la baisse de niveau des performances et les niveaux plus élevés d'erreurs et d'alertes du système. Ces outils peuvent également être utilisés pour détecter et informer en cas de comportement suspect (comme les attaques par déni de service et les attaques par déni de service distribué).

Outils d'analyse de logslogs
Ce sont des outils qui analysent les logs du serveur et compilent les métriques à partir de ceux-ci. Certains de ces outils peuvent créer des diagrammes pour fournir une vue graphique des données.

Les erreurs, les alertes et les avertissements sont normalement enregistrés dans les logs du serveur. Il s'agit notamment de :

- Utilisation élevée des ressources, comme l'utilisation élevée du processeur, les niveaux élevés de stockage sur le disque et l'insuffisance de la bande passante
- Erreurs de mémoire et avertissements, tels que l'épuisement de la mémoire
- Impasses et problèmes multi-threading, en particulier lors de l'exécution des opérations de base de données
- Erreurs de base de données, telles que les exceptions SQL et les délais d'attente SQL

2.4 Résultats typiques d'un test de performance

Dans les tests fonctionnels, en particulier lors de la vérification des besoins fonctionnels spécifiés ou des éléments fonctionnels des user stories ; les résultats attendus peuvent généralement être définis clairement et les résultats des tests interprétés pour déterminer si le test a réussi ou échoué. Par exemple, un rapport de vente mensuel montre un total correct ou incorrect.

Alors que les tests qui vérifient la pertinence fonctionnelle bénéficient souvent d'oracles de test bien définis, les tests de performance manquent souvent de cette source d'information. Non seulement les parties prenantes sont notoirement mauvaises lorsqu'il s'agit de définir des exigences de performance, mais encore, de nombreux analystes métier et Product Owners sont tout aussi mauvais en la matière. Les testeurs reçoivent souvent des directives limitées pour définir les résultats attendus des tests.

Lors de l'évaluation des résultats des tests de performance, il est important d'examiner les résultats de près. Les premiers résultats bruts peuvent être trompeurs avec des échecs de performance cachés sous des résultats globaux apparemment bons. Par exemple, l'utilisation des ressources peut être bien inférieure à 75 % pour toutes les ressources clés constituant un potentiel goulet d'étranglement, mais le débit ou le temps de réponse des transactions clés ou des cas d'utilisation sont d'un ordre de grandeur trop lents.

Les résultats spécifiques à évaluer varient selon les tests exécutés, et comprennent souvent ceux discutés dans la Section 2.1.

3. Tests de performance dans le cycle de vie du logiciel – 195 mins.

Mots-clés

Métrique, risque, cycle de vie de développement logiciel, log de test

Objectifs d'apprentissage

3.1 Activités principales des tests de performance
PTFL-3.1.1 (K2) Comprendre les principales activités de test de performance

3.2 Risques de performance pour différentes architectures
PTFL-3.2.1 (K2) Expliquer les catégories typiques de risques de performance pour différentes architectures

3.3 Risques de performance tout au long du cycle de vie du développement logiciel
PTFL-3.3.1 (K4) Analyser les risques de performance d'un produit donné tout au long du cycle de vie du développement logiciel

3.4 Activités de test de performance
PTFL-3.4.1 (K4) Analyser un projet donné pour déterminer les activités de test de performance appropriées pour chaque phase du cycle de vie du développement logiciel

3.1 Activités principales des tests de performance

Les tests de performance sont de nature itérative. Chaque test fournit des informations précieuses sur les performances de l'application et du système. Les informations recueillies à partir d'un test sont utilisées pour corriger ou optimiser les paramètres de l'application et du système. L'itération de test suivante montrera ensuite les résultats des modifications, et ainsi de suite jusqu'à ce que les objectifs de test soient atteints.

Les activités de test de performance s'alignent sur le processus de test de l'ISTQB [ISTQB_FL_SYL].

Planification des tests
La planification des tests est particulièrement importante pour les tests de performance en raison de la nécessité d'allocation d'environnements de test, de données de test,

d'outils et de ressources humaines. En outre, il s'agit de l'activité dans laquelle la portée des tests de performance est établie.

Pendant la planification des tests, les activités d'identification des risques et d'analyse des risques sont achevées et l'information pertinente est mise à jour dans toute documentation de planification des tests (p. ex., plan de test, plan de test de niveau). Tout comme la planification des tests est réexaminée et modifiée au besoin, les risques, les niveaux de risque et l'état du risque sont modifiés pour tenir compte des changements dans les conditions.

Surveillance et contrôle des tests
Les mesures de contrôle sont définies pour fournir des plans d'action si l'on rencontre des problèmes qui pourraient avoir une incidence sur l'efficacité de la performance, comme :

- Augmentation de la capacité de production de charge si l'infrastructure ne génère pas les charges souhaitées comme prévu pour des tests de performance particuliers
- Matériel modifié, nouveau ou remplacé
- Modifications apportées aux composants réseau
- Modifications à la mise en œuvre de logiciels

Les objectifs de test de performance sont évalués pour vérifier la réalisation des critères de sortie.

Analyse des tests
Des tests de performance efficaces sont basés sur une analyse des exigences en matière de performance, d'objectifs de test, de Service Level Agreements (SLA), d'architecture informatique, de modèles de processus et d'autres éléments qui constituent la base de test. Cette activité peut être soutenue par la modélisation et l'analyse des besoins en ressources du système et/ou du comportement à l'aide de feuilles de calcul ou d'outils de planification de la capacité.

Des conditions de test spécifiques sont identifiées telles que les niveaux de charge, les conditions de timing et les transactions à tester. Le type de test de performance requis (p. ex., charge, stress, passage à l'échelle) est alors décidé.

Conception des tests
Les cas de test de performance sont conçus. Ceux-ci sont généralement créés sous forme modulaire afin qu'ils puissent être utilisés comme éléments constitutifs de tests de performance plus grands et plus complexes (voir la section 4.2).

Implémentation des tests
Au cours de la phase d'implémentation, les cas de test de performance sont classés dans les procédures de test de performance. Ces procédures de test de performance doivent refléter les étapes normalement prises par l'utilisateur et d'autres activités fonctionnelles qui doivent être couvertes pendant les tests de performance.

Une activité d'implémentation de test consiste à établir et/ou à réinitialiser l'environnement de test avant chaque exécution de test. Étant donné que les tests de performance sont généralement pilotés par les données, il est nécessaire de définir un processus afin d'établir des données de test représentatives des données de production réelles, en volume et en type, afin que l'utilisation de la production puisse être simulée.

Exécution des tests
L'exécution du test se produit lorsque le test de performance est effectué, souvent à l'aide d'outils de test de performance. Les résultats des tests sont évalués afin de déterminer si la performance du système répond aux exigences et aux autres objectifs énoncés. Tous les défauts sont signalés.

Clôture des tests
Les résultats des tests de performance sont fournis aux parties prenantes (p. ex., architectes, managers, Product Owners) dans un rapport de synthèse de test . Les résultats sont exprimés par des métriques qui sont souvent agrégées pour simplifier le sens des résultats des tests. Des moyens de reporting visuels tels que les tableaux de bord sont souvent utilisés pour exprimer les résultats des tests de performance d'une manière plus facile à comprendre que les métriques textuelles.

Les tests de performance sont souvent considérés comme une activité continue en ce qu'ils sont effectués à plusieurs reprises et à tous les niveaux de test (composants, intégration, système, intégration du système et tests d'acceptation). À la fin d'une période définie de tests de performance, un point de clôture des tests peut être atteint lorsque des tests conçus, des ressources d'outils de test (cas et procédures de test), des données de test et d'autres éléments du testware sont archivés ou transmis à d'autres testeurs pour une utilisation ultérieure pendant les activités de maintenance du système.

3.2 Catégories de risques de performance pour différentes architectures

Comme mentionné précédemment, les performances de l'application ou du système varient considérablement en fonction de l'architecture, de l'application et de l'environnement d'accueil. Bien qu'il ne soit pas possible de fournir une liste complète

des risques de performance pour tous les systèmes, la liste ci-dessous comprend certains types typiques de risques associés à des architectures particulières :

Systèmes sur un seul ordinateur
Il s'agit de systèmes ou d'applications qui s'exécutent entièrement sur un ordinateur non virtualisé. Les performances peuvent se dégrader en raison de :

- Consommation excessive de ressources, y compris les fuites de mémoire, les tâches de fond telles que les logiciels de sécurité, les sous-systèmes de stockage lents (p. ex., les appareils externes à basse vitesse ou la fragmentation du disque) et la mauvaise gestion par le système d'exploitation.
- Mise en œuvre inefficace d'algorithmes qui n'utilisent pas les ressources disponibles (p. ex., mémoire principale) et, par conséquent, exécutent plus lentement que nécessaire.

Systèmes multi-tiers
Il s'agit de systèmes qui s'exécutent sur plusieurs serveurs, chacun effectuant un ensemble spécifique de tâches, telles que le serveur de base de données, le serveur d'application et le serveur de présentation. Chaque serveur est, bien sûr, un ordinateur et est soumis aux risques indiqués précédemment. En outre, les performances peuvent se dégrader en raison d'une conception de base de données médiocre ou non évolutive, des goulots d'étranglement du réseau, et d'une bande passante ou d'une capacité inadéquate sur n'importe lequel des serveurs.

Systèmes distribués
Ce sont des systèmes, semblables à une architecture multi-tiers, mais les différents serveurs peuvent changer dynamiquement, comme un système de commerce électronique qui accède à différentes bases de données d'inventaire en fonction de l'emplacement géographique de la personne qui passe la commande. En plus des risques associés aux architectures multi-tier, cette architecture peut rencontrer des problèmes de performances dus à des flux de travail critiques ou des flux de données vers, de, ou à travers de serveurs distants peu fiables ou imprévisibles, en particulier lorsque ces serveurs souffrent de problèmes de connexion périodiques ou de périodes intermittentes de charge intense.

Systèmes virtualisés
Ce sont des systèmes où le matériel physique héberge plusieurs ordinateurs virtuels. Ces machines virtuelles peuvent héberger des systèmes et applications mono-ordinateurs ainsi que des serveurs qui font partie d'une architecture multi-tiers ou distribuée. Les risques de performance qui découlent spécifiquement de la virtualisation comprennent une charge excessive sur le matériel au travers de toutes les machines virtuelles ou une configuration incorrecte de la machine virtuelle hôte avec pour résultat des ressources insuffisantes.

Systèmes dynamiques/cloud
Ce sont des systèmes qui offrent la possibilité d'évoluer à la demande, en augmentant la capacité à mesure que le niveau de charge augmente. Ces systèmes sont généralement des systèmes distribués et multi-tiers virtualisés, mais avec des fonctionnalités de mise à l'échelle automatique conçues spécifiquement pour atténuer certains des risques de performance associés à ces architectures. Cependant, il existe des risques associés à la configuration incorrecte de ces fonctionnalités lors de leur déploiement initial ou lors des mises à jour ultérieures.

Systèmes client –serveur
Il s'agit de systèmes fonctionnant sur un client qui communique via une interface utilisateur avec un seul serveur, un serveur multi-tiers ou un serveur distribué. Étant donné que le code s'exécute sur le client, les risques décrits pour un ordinateur unique s'appliquent, tandis que les problèmes côté serveur mentionnés ci-dessus s'appliquent également. De plus, des risques de performance existent en raison de problèmes de vitesse et de fiabilité de connexion, de congestion du réseau au point de connexion client (p. ex., Wi-Fi public) et de problèmes potentiels dus aux pares-feux, à l'inspection des paquets et à l'équilibrage de la charge du serveur.

Applications mobiles
Il s'agit d'applications fonctionnant sur un smartphone, une tablette ou un autre appareil mobile. Ces applications sont soumises aux risques mentionnés pour les applications client-serveur et basées sur un navigateur (applications Web). En outre, des problèmes de performances peuvent survenir en raison des ressources limitées et variables et de la connectivité disponibles sur l'appareil mobile (qui peuvent être affectées par l'emplacement, la durée de vie de la batterie, l'état de charge, la mémoire disponible sur l'appareil et la température). Pour les applications qui utilisent des capteurs ou des radiofréquences telles que des accéléromètres ou Bluetooth, les flux de données lents provenant de ces sources pourraient créer des problèmes. Enfin, les applications mobiles ont souvent de lourdes interactions avec d'autres applications mobiles locales et des services Web distants, toutes choses pouvant potentiellement devenir un goulet d'étranglement de l'efficacité des performances.

Systèmes embarqués en temps réel
Il s'agit de systèmes embarqués aux objets quotidiens voire même qui les contrôlent telles que les voitures (p. ex. les systèmes de divertissement et les systèmes de freinage intelligents), les ascenseurs, les feux de circulation, le chauffage, la ventilation et la climatisation (CVC), et plus encore. Ces systèmes présentent souvent nombre de risques communs aux appareils mobiles, y compris des problèmes (de plus en plus) liés à la connectivité puisque ces appareils sont connectés à Internet. Toutefois, la diminution des performances d'un jeu vidéo mobile n'est généralement pas un danger

pour la sécurité de l'utilisateur, tandis que de tels ralentissements dans un système de freinage de véhicule pourrait s'avérer catastrophique.

Applications Mainframe
Il s'agit d'applications, dans de nombreux cas, vieilles de plusieurs décennies, qui prennent en charge dans un centre de données, des fonctions métier souvent essentielles à la mission, parfois par le biais du traitement par lots. La plupart sont tout à fait prévisibles et rapides lorsqu'elles sont utilisées comme prévu initialement, mais beaucoup d'entre elles sont maintenant accessibles via des API, des services Web, ou par le biais de leur base de données, ce qui peut entraîner des charges inattendues qui affectent le débit d'applications établies.

Notez qu'une application ou un système particulier peut incorporer deux ou plusieurs des architectures énumérées ci-dessus, ce qui signifie que tous les risques pertinents s'appliqueront aussi à cette application ou à ce système. En fait, compte tenu de l'Internet des objets et de l'explosion des applications mobiles - deux domaines où les niveaux extrêmes d'interaction et de connexion sont la règle - il est possible que toutes les architectures soient présentes sous une forme ou une autre dans une application, et donc tous les risques peuvent s'appliquer.

Bien que l'architecture soit clairement une décision technique importante ayant un impact profond sur les risques de performance, d'autres décisions techniques influencent et génèrent des risques. Par exemple, les fuites de mémoire sont plus courantes avec les langues qui permettent la gestion directe de la mémoire de tas (NDT : heap memory), telles que C et C++, et les problèmes de performance sont différents pour les bases de données relationnelles et les bases de données non relationnelles. De telles décisions s'étendent jusqu'à la conception de fonctions ou de méthodes individuelles (p. ex., le choix d'un algorithme récursif par opposition à un algorithme itératif). En tant que testeur, la capacité à connaître ou même à influencer de telles décisions variera en fonction des rôles et des responsabilités des testeurs au sein de l'organisation et du cycle de vie du développement de logiciels.

3.3 Risques de performance tout au long du cycle de vie du développement logiciel

Le processus d'analyse des risques pour la qualité d'un produit logiciel en général est discuté dans divers syllabi ISTQB (par exemple, voir [ISTQB_FL_SYL] et [ISTQB_ALTM_SYL]). Vous pouvez également trouver des discussions sur des risques spécifiques et des considérations associés à des caractéristiques particulières de la qualité (p. ex., [ISTQB_UT_SYL]), et d'un point de vue métier ou technique (p. ex., voir [ISTQB_ALTA_SYL] et [ISTQB_ALTTA_SYL], respectivement). Dans cette

section, l'accent est mis sur les risques liés à la performance pour la qualité des produits, y compris les façons dont le processus, les participants et les considérations changent.

Pour ce qui est des risques liés à la performance pour la qualité du produit, le processus est le suivant :

1. Identifier les risques pour la qualité du produit, en mettant l'accent sur des caractéristiques telles que le comportement du temps, l'utilisation des ressources et la capacité.
2. Évaluer les risques identifiés, en veillant à ce que les catégories d'architecture pertinentes (voir la section 3.2) soient traitées. Évaluer le niveau global de risque pour chaque risque identifié en termes de probabilité et d'impact à l'aide de critères clairement définis.
3. Prendre les mesures appropriées d'atténuation des risques pour chaque élément de risque en fonction de la nature de l'élément de risque et du niveau de risque.
4. Gérer les risques sur une base continue pour s'assurer que les risques sont adéquatement atténués avant la livraison.

Comme pour l'analyse des risques de qualité en général, les participants à ce processus devraient inclure les parties prenantes métier et techniques. Pour l'analyse des risques liés au rendement, les parties prenantes métier doivent inclure ceux qui sont particulièrement conscients de la façon dont les problèmes de performance en production affecteront réellement les clients, les utilisateurs, l'entreprise et d'autres parties prenantes en aval. Les parties prenantes métier doivent comprendre que l'utilisation prévue, les dommages pour l'entreprise, la société ou les dommages critique pour la sécurité, les dommages financiers et/ou de réputation potentiels, la responsabilité juridique civile ou pénale et des facteurs similaires influent sur le risque du point de vue métier, ce qui crée des risques et influence l'impact des défaillances.

De plus, les parties prenantes techniques doivent inclure ceux qui ont une compréhension approfondie des implications en matière de performance des exigences pertinentes, de l'architecture, de la conception et des décisions de mise en œuvre. Les parties prenantes techniques doivent comprendre que les décisions d'architecture, de conception et de mise en œuvre affectent les risques de performance d'un point de vue technique, ce qui crée des risques et influence la probabilité de défauts.

Le processus spécifique d'analyse des risques choisi devrait avoir le niveau approprié de formalité et de rigueur. En ce qui concerne les risques liés à la performance, il est particulièrement important que le processus d'analyse des risques soit lancé tôt et se répète régulièrement. En d'autres termes, le testeur doit éviter de s'appuyer

entièrement sur les tests de performance effectués vers la fin du niveau de test du système et le niveau de test d'intégration du système. De nombreux projets, en particulier des projets de systèmes de systèmes plus grands et plus complexes, ont rencontré des surprises malheureuses en raison de la découverte tardive de défauts de performance provenant des exigences, de la conception, de l'architecture et des décisions de mise en œuvre prises tôt dans le projet. L'accent devrait donc être mis sur une approche itérative de l'identification, de l'évaluation, de l'atténuation et de la gestion des risques de performance tout au long du cycle de vie du développement logiciel.

Par exemple, si de grands volumes de données sont traités via une base de données relationnelle, les performances lentes de jointure plusieurs à plusieurs dues à une mauvaise conception de la base de données, ne peuvent se révéler que lors de tests dynamiques avec des jeux de données de test à grande échelle, tels que ceux utilisés lors des essais système. Cependant, une revue technique minutieuse effectuée par des ingénieurs de base de données expérimentés peut prédire les problèmes avant la mise en œuvre de la base de données. Après une telle revue, dans une approche itérative, les risques sont identifiés et évalués à nouveau.

En outre, l'atténuation et la gestion des risques doivent s'étendre et influencer l'ensemble du processus de développement de logiciels, et pas seulement les tests dynamiques. Par exemple, lorsque des décisions critiques liées à la performance, comme le nombre prévu de transactions ou d'utilisateurs simultanés, ne peuvent pas être spécifiées au début du projet, il est important que les décisions de conception et d'architecture permettent une très grande évolutivité (par exemple, des ressources informatiques à la demande dans le cloud). Cela permet de prendre des décisions précoces en matière d'atténuation des risques.

Une bonne ingénierie des performances peut aider les équipes de projet à éviter la découverte tardive de défauts de performance critiques lors de niveaux de test plus élevés, tels que les tests d'intégration du système ou les tests d'acceptation des utilisateurs. Les défauts de performance constatés à un stade avancé du projet peuvent être extrêmement coûteux et peuvent même entraîner l'arrêt de projets entiers.

Comme pour tout type de risque de qualité, les risques liés à la performance ne peuvent jamais être évités complètement, c'est-à-dire qu'il existera toujours un risque de défaillance de la production liée à la performance. Par conséquent, le processus de gestion des risques doit comprendre une évaluation réaliste et spécifique du niveau résiduel de risque pour les parties prenantes techniques et métier impliquées dans le processus. Par exemple, il n'est pas utile de simplement dire : « Oui, il est toujours possible pour les clients d'éprouver de longs retards pendant le règlement (NDT : du

panier) », car cela ne donne aucune idée de la quantité d'atténuation des risques qui s'est produite ou du niveau de risque qui subsiste. Au lieu de cela, fournir un aperçu clair du pourcentage de clients susceptibles d'éprouver des retards égaux à ou dépassant certains seuils, aidera à comprendre le statut.

3.4 Activités de test de performance

Les activités de test de performance seront organisées et effectuées différemment, selon le type de cycle de vie du développement logiciel utilisé.

Modèles de développement séquentiel

La pratique idéale des tests de performance dans les modèles de développement séquentiel est d'inclure des critères de performance dans le cadre des critères d'acceptation qui sont définis au début d'un projet. Venant renforcer la vision du cycle de vie des tests, les activités de test de performance devraient être menées tout au long du cycle de vie du développement logiciel. Au fur et à mesure que le projet progresse, chaque activité de test de performance successive doit être fondée sur des éléments définis dans les activités antérieures comme indiqué ci-dessous.

- Concept - Vérifiez que les objectifs de performance du système sont définis comme des critères d'acceptation pour le projet.
- Exigences - Vérifier que les exigences en matière de performance sont définies et représentent correctement les besoins des parties prenantes.
- Analyse et conception - Vérifiez que la conception du système reflète les exigences de performance.
- Codage/implémentation - Vérifiez que le code est efficace et reflète les exigences et la conception en termes de performances.
- Tests de composants - Effectuer des tests de performance au niveau des composants.
- Test d'intégration des composants - Effectuer des tests de performance au niveau de l'intégration des composants.
- Test système - Effectuer des tests de performance au niveau du système, qui comprend le matériel, les logiciels, les procédures et les données qui sont représentatifs de l'environnement de production. Les interfaces système peuvent être simulées à condition qu'elles donnent une véritable représentation des performances.
- Test d'intégration de systèmes - Effectuer des tests de performance avec l'ensemble du système qui est représentatif de l'environnement de production.
- Tests d'acceptation - Valider que les performances du système répondent aux besoins et aux critères d'acceptation énoncés à l'origine.

Modèles de développement itératif et incrémental

Dans ces modèles de développement, tels qu'Agile, les tests de performance sont également considérés comme une activité itérative et incrémentale (voir [ISTQB_FL_AT]). Les tests de performance peuvent se faire dans le cadre de la première itération, ou comme une itération entièrement dédiée aux tests de performance. Toutefois, avec ces modèles de cycle de vie, l'exécution des tests de performance peut être effectuée par une équipe distincte chargée de tester les performances.

L'intégration continue (IC) est couramment effectuée dans les cycles de vie itératifs et incrémentaux de développement de logiciels, ce qui facilite une exécution hautement automatisée des tests. L'objectif le plus courant des tests dans l'IC est d'effectuer des tests de régression et de s'assurer que chaque build est stable. Les tests de performance peuvent faire partie des tests automatisés effectués dans l'IC si les tests sont conçus de manière à être exécutés à un niveau de build. Toutefois, contrairement aux tests automatisés fonctionnels, il existe d'autres préoccupations telles que :

- La configuration de l'environnement de test de performance - Cela nécessite souvent un environnement de test qui est disponible sur demande, comme un environnement de test de performance basé sur le cloud.
- Déterminer les tests de performance à automatiser dans l'IC - En raison du court laps de temps disponible pour les tests d'IC, les tests de performance d'IC peuvent être un sous-ensemble de tests de performance plus étendus qui sont effectués par une équipe spécialisée à d'autres moments au cours d'une itération.
- Création des tests de performance pour l'IC - L'objectif principal des tests de performance dans le cadre de l'IC est de s'assurer qu'un changement n'a pas d'impact négatif sur la performance. Selon les modifications apportées pour un build donné, de nouveaux tests de performance peuvent être nécessaires.
- Exécution de tests de performance sur des parties d'une application ou d'un système - Cela nécessite souvent que les outils et les environnements de test soient capables de tests de performance rapides, y compris la possibilité de sélectionner des sous-ensembles de tests applicables.

Les tests de performance dans les cycles de vie itératifs et incrémentaux du développement logiciel peuvent également avoir leurs propres activités de cycle de vie :

- Planification de la version - Dans cette activité, les tests de performance sont considérés du point de vue de toutes les itérations dans une version, de la première itération à l'itération finale. Les risques liés à la performance sont identifiés et évalués, et les mesures d'atténuation sont prévues. Cela comprend souvent la planification de tout test de performance final avant la livraison de l'application.

- Planification de l'itération - Dans le contexte de chaque itération, des tests de performance peuvent être effectués dans le cadre de l'itération et, à mesure que chaque itération est terminée.
  Les risques de performance sont évalués plus en détail pour chaque User Story.
- Création de User Story – Les User Stories constituent souvent la base des exigences en matière de performance dans les méthodologies Agile, avec les critères de performance spécifiques décrits dans les critères d'acceptation associés. Celles-ci sont appelées user stories « non fonctionnelles ».
- Conception des tests de performance – les exigences de performance et les critères qui sont décrits dans des User Stories particulières sont utilisés pour la conception de tests (voir section 4.2)
- Codage/Implémentation – Pendant le codage, les tests de performance peuvent être effectués au niveau des composants. Un exemple de ceci serait l'ajustement des algorithmes pour l'efficacité optimale de performance.
- Test/Evaluation – Bien que les tests soient généralement effectués à proximité des activités de développement, les tests de performance peuvent être effectués en tant qu'activité distincte, selon la portée et les objectifs des tests de performance pendant l'itération. Par exemple, si l'objectif des tests de performance est de tester les performances de l'itération en tant qu'ensemble complet de User Stories, une plus grande portée de tests de performance sera nécessaire que celle observée dans le test de performance d'une seule User Story. Cela peut être programmé dans une itération dédiée pour les tests de performance.
- Livraison – Étant donné que la livraison introduira l'application dans l'environnement de production, les performances devront être surveillées pour déterminer si l'application atteint les niveaux de performance souhaités dans l'utilisation réelle.

Logiciel commercial sur étagère et autres modèles fournisseurs/acquéreurs

De nombreuses organisations ne développent pas elles-mêmes des applications et des systèmes, mais sont plutôt en mesure d'acquérir des logiciels auprès de fournisseurs ou de projets open-source. Dans ces modèles fournisseurs/acquéreurs, la performance est une considération importante qui nécessite de tester à la fois selon les perspectives du fournisseur (fournisseur/développeur) et celles de l'acquéreur (client).

Quelle que soit la source de l'application, il est souvent de la responsabilité du client de valider que la performance répond à ses exigences. Dans le cas des logiciels personnalisés développés par le fournisseur, des exigences en matière de performances et des critères d'acceptation connexes doivent être spécifiés dans le cadre du contrat entre le fournisseur et le client. Dans le cas des applications sur

étagère, le client a la responsabilité exclusive de tester les performances du produit dans un environnement de test réaliste avant le déploiement.

4. Tâches de test de performance – 475 mins.

Mots-clés

Concurrence, profil de charge, génération de charge, profil opérationnel, descente de charge, montée en charge, système de systèmes, débit du système, plan de test, temps de réflexion, utilisateur virtuel

Objectifs d'apprentissage

4.1 Planning

PTFL-4.1.1 (K4) Dériver les objectifs des tests de performance à partir d'informations pertinentes
PTFL-4.1.2 (K4) Définir un plan de test de performance qui tient compte des objectifs de performance d'un projet donné
PTFL-4.1.3 (K4) Créer une présentation qui permet à diverses parties prenantes de comprendre la raison d'être des tests de performance prévus

4.2 Analyse, Conception et Implémentation

PTFL-4.2.1 (K2) Donner des exemples de protocoles typiques rencontrés dans les tests de performance
PTFL-4.2.2 (K2) Comprendre le concept de transactions dans les tests de performance
PTFL-4.2.3 (K4) Analyser les profils opérationnels pour l'utilisation du système
PTFL-4.2.4 (K4) Créer des profils de charge dérivés de profils opérationnels pour les objectifs de performance donnés
PTFL-4.2.5 (K4) Analyser le débit et la concurrence lors de l'élaboration de tests de performance
PTFL-4.2.6 (K2) Comprendre la structure de base d'un script de test de performance
PTFL-4.2.7 (K3) Mettre en œuvre des scripts de test de performance conformes au plan et aux profils de chargement
PTFL-4.2.8 (K2) Comprendre les activités liées à la préparation à l'exécution des tests de performance

4.3 Exécution

PTFL-4.3.1 (K2) Comprendre les principales activités dans l'exécution des scripts de test de performance

4.4 Analyse des résultats et des rapports

PTFL-4.4.1 (K4) Analyser et rapporter les résultats et les implications des tests de performance

4.1 Planning

4.1.1 Dériver les objectifs de test de performance

Les parties prenantes peuvent inclure des utilisateurs et des personnes ayant une expérience technique ou métier. Ils peuvent avoir des objectifs différents en ce qui concerne les tests de performance. Les parties prenantes fixent les objectifs, la terminologie à utiliser et les critères pour déterminer si l'objectif a été atteint.

Les objectifs des tests de performance sont liés à ces différents types de parties prenantes. Il est une bonne pratique de faire la distinction entre les objectifs d'utilisation et les objectifs techniques. Les objectifs d'utilisation se concentrent principalement sur la satisfaction des utilisateurs finaux et les objectifs métier. En général, les utilisateurs sont moins préoccupés par les types d'entités ou la façon dont un produit est livré. Ils veulent juste être en mesure de faire ce qu'ils doivent faire.

Les objectifs techniques, d'autre part, se concentrent sur les aspects opérationnels et aux questions concernant la capacité de passage à l'échelle d'un système, ou dans quelles conditions des performances dégradées peuvent devenir apparentes.

Les principaux objectifs des tests de performance comprennent l'identification des risques potentiels, la recherche de possibilités d'amélioration et l'identification des changements nécessaires.

Lors de la collecte d'informations auprès des différentes parties prenantes, il faut répondre aux questions suivantes :

- Quelles transactions seront exécutées lors des tests de performance et quel temps de réponse moyen est prévu ?
- Quelles métriques du système doivent être capturées (p. ex., utilisation de la mémoire, débit réseau) et quelles valeurs sont attendues ?
- Quelles améliorations de performance sont attendues de ces tests par rapport aux cycles de test précédents ?

4.1.2 Le plan de test de performance

Le Plan de Test de Performance (PTP) est un document créé avant tout test de performance. Le PTP devrait être mentionné dans le plan de test (voir [ISTQB_FL_SYL]) qui comprend également des informations pertinentes sur l'horaire. Il est constamment mis à jour une fois que les tests de performance ont commencé.

Les informations suivantes doivent être fournies dans un PTP :

Objectif
L'objectif du PTP décrit les buts, les stratégies et les méthodes des tests de performance. Il permet une réponse quantifiable à la question centrale de l'adéquation et de la disponibilité du système à tester sous charge.

Objectifs de test
Les objectifs globaux de test pour l'efficacité de la performance à atteindre par le système sous test (SST) sont énumérés pour chaque type d'intervenant (voir Section 4.1.1)

Aperçu du système
Une brève description du SST fournira le contexte de la mesure des paramètres des tests de performance. L'aperçu devrait inclure une description de haut niveau de la fonctionnalité testée sous charge.

Types de tests de performance à effectuer
Les types de tests de performance à effectuer sont énumérés (voir la section 1.2) ainsi qu'une description de l'objet de chaque type.

Critères d'acceptation
Les tests de performance visent à déterminer la réactivité (NDT responsiveness), le débit, la fiabilité et/ou la capacité de passage à l'échelle du système dans le cadre d'une charge de travail donnée. En général, le temps de réponse est une préoccupation de l'utilisateur, le débit est une préoccupation du métier, et l'utilisation des ressources est une préoccupation du système. Les critères d'acceptation devraient être fixés pour toutes les mesures pertinentes et liés aux points suivants, le cas échéant :

- Objectifs globaux de test de performance
- Service Level Agreements (SLAs)
- Valeurs de base - Une ligne de base (NDT baseline) est un ensemble de mesures utilisées pour comparer les mesures de performance actuelles et précédemment réalisées. Cela permet de démontrer des améliorations particulières à la performance et/ou de confirmer la réalisation de critères d'acceptation des tests. Il peut être nécessaire, dans la mesure du possible, de créer d'abord la ligne de base à l'aide de données assainies à partir d'une base de données.

Données de test
Les données de test comprennent un large éventail de données qui doivent être spécifiées pour un test de performance. Ces données peuvent inclure :

- Données de compte utilisateur (p. ex., comptes d'utilisateurs disponibles pour la connexion simultanée)

- Données d'entrée utilisateur (p. ex., les données qu'un utilisateur saisirait dans l'application afin d'effectuer un processus métier)
- Base de données (p. ex., la base de données pré-peuplée qui est remplie de données à utiliser dans les tests)

Le processus de création de données de test doit aborder les aspects suivants :

- Extraction de données à partir de données de production
- Importation de données dans le SST
- Création de nouvelles données
- Création de sauvegardes qui peuvent être utilisées pour restaurer les données lorsque de nouveaux cycles de tests sont effectués
- Masquage ou anonymisation des données. Cette pratique est utilisée sur les données de production qui contiennent des informations personnelles identifiables et sont obligatoires en vertu du Règlement Général sur la Protection des Données (RGPD). Cependant, dans les tests de performance, le masquage des données ajoute des risques aux tests de performance car ils peuvent ne pas avoir les mêmes caractéristiques que les données utilisées dans le monde réel.

Configuration du système

La section de configuration du système du PTP comprend les informations techniques suivantes :

- Une description de l'architecture spécifique du système, y compris les serveurs (p. ex., web, base de données, équilibrage de charge)
- Définition des multiple-tiers
- Détails spécifiques du matériel informatique (p. ex. cœurs CPU, RAM, disques à semi-conducteurs (SSD), disques durs (HDD)) y compris les versions
- Détails spécifiques des logiciels (p. ex. applications, systèmes d'exploitation, bases de données, services utilisés pour soutenir l'entreprise), y compris les versions
- Systèmes externes qui fonctionnent avec le SST et leur configuration et leur version (p. ex., système de commerce électronique avec intégration à NetSuite)
- Build du SST / identificateur de version

Environnement de test

L'environnement de test est souvent un environnement distinct qui imite la production, mais à plus petite échelle. Cette section du PTP devrait inclure la façon dont les résultats des tests de performance seront extrapolés pour s'appliquer à l'environnement de production plus vaste. Avec certains systèmes, l'environnement de

production devient la seule option viable pour les tests, mais dans ce cas, les risques spécifiques de ce type de test doivent être discutés.

Les outils de test résident parfois en dehors de l'environnement de test lui-même et peuvent nécessiter des droits d'accès spéciaux afin d'interagir avec les composants du système. Il s'agit d'une considération à prendre en compte pour l'environnement de test et la configuration.

Des tests de performance peuvent également être effectués avec une partie des composants du système qui est capable de fonctionner sans les autres composants. C'est souvent moins cher que de tester avec l'ensemble du système et cela peut être effectué dès que le composant est développé.

Outils de test
Cette section comprend une description des outils de test (et des versions) qui seront utilisés dans le scripting, l'exécution et le suivi des tests de performance (voir le chapitre 5). Cette liste comprend normalement :

- Outil(s) utilisé(s) pour simuler les transactions utilisateur
- Outils pour fournir la charge à partir de plusieurs points dans l'architecture du système (points de présence)
- Outils pour surveiller les performances du système, y compris ceux pour la configuration du système décrits ci-dessus

Profils
Les profils opérationnels fournissent un flux répétable, étape par étape, à travers l'application, pour une utilisation particulière du système. L'agrégation de ces profils opérationnels se traduit par un profil de charge (communément appelé scénario). Voir la section 4.2.3 pour plus d'informations sur les profils.

Métriques pertinentes
Un grand nombre de mesures et de métriques peuvent être recueillies lors de l'exécution d'un test de performance (voir le chapitre 2). Cependant, la prise de trop de mesures peut rendre l'analyse difficile et avoir un impact négatif sur les performances réelles de l'application. Pour ces raisons, il est important d'identifier les mesures et les métriques les plus pertinentes pour atteindre les objectifs des tests de performance.

Le tableau suivant, expliqué plus en détail dans la section 4.4, montre un ensemble typique de mesures pour les tests de performance et le suivi. Les objectifs de test pour la performance doivent être définis pour ces mesures, le cas échéant, pour le projet :

Métriques de performance	
Type	**Métrique**
Statut d'utilisateur virtuel	# Passé # Échoué
Temps de réponse à la transaction	Minimum Maximum Moyenne 90% Percentile
Transactions par seconde	# Passé / seconde # Échoué / seconde # Total / seconde
Accès (NDT : hits) (p. ex., sur la base de données ou le serveur Web)	Accès / seconde ▪ Minimum ▪ Maximum ▪ Moyenne ▪ Total
Débit	Bits / seconde ▪ Minimum ▪ Maximum ▪ Moyenne ▪ Total
HTTP Réponses par seconde	Réponses / second ▪ Minimum ▪ Maximum ▪ Moyenne ▪ Total Réponse par HTTP Codes de réponse

Suivi des performances	
Type	**Métrique**
Utilisation du processeur (CPU)	% de CPU disponible utilisé
Utilisation de la mémoire	% de la mémoire disponible utilisée

Risques
Les risques peuvent inclure les zones non mesurées dans le cadre des tests de performance ainsi que les limitations des tests de performance (p. ex., interfaces externes qui ne peuvent pas être simulées, charge insuffisante, incapacité à surveiller les serveurs). Les limites de l'environnement de test peuvent également entraîner des risques (p. ex., données insuffisantes, environnement réduit). Voir Sections 3.2 and 3.3 pour plus de types de risque.

4.1.3 Communiquer sur les tests de performance

Le testeur doit être capable de communiquer à toutes les parties prenantes la raison d'être de l'approche de test de performance et les activités à entreprendre (comme indiqué dans le Plan de Test de Performance). Les sujets à traiter dans cette communication peuvent varier considérablement d'une partie à l'autre selon qu'ils ont un intérêt « métier / utilisateur » ou une orientation plus « technologique / opérationnelle ».

Parties prenantes ayant une orientation métier
Les facteurs suivants doivent être pris en considération lors de la communication avec les parties prenantes orientées métier :

- Les parties prenantes ayant une orientation métier s'intéressent moins aux distinctions entre les caractéristiques de qualité fonctionnelles et non fonctionnelles.
- Les questions techniques concernant l'outillage, le scripting et la génération de charge sont généralement d'intérêt secondaire.
- Le lien entre les risques produits et les objectifs de test de performance doit être clairement établi.
- Les parties prenantes doivent être sensibilisés à l'équilibre entre le coût des tests de performance planifiés et la représentativité des résultats des tests de performance, par rapport aux conditions de production.
- La répétabilité des tests de performance prévus doit être communiquée. Le test sera-t-il difficile à répéter ou peut-il être répété avec un minimum d'effort ?
- Les risques du projet doivent être communiqués. Il s'agit notamment de contraintes et de dépendances concernant la mise en place des tests, les exigences en matière d'infrastructure (p. ex., le matériel, les outils, les données, la bande passante, l'environnement de test, les ressources) et les dépendances vis-à-vis du personnel clé.
- Les activités de haut niveau doivent être communiquées (voir les sections 4.2 et 4.3) ainsi qu'un vaste plan contenant les coûts, les délais et les jalons.

Parties prenantes techniques

Les facteurs suivants doivent être pris en considération lors de la communication avec les parties prenantes ayant une orientation technique :

- L'approche prévue pour générer les profils de charge requis doit être expliquée et la participation attendue des parties prenantes techniques clarifiée.
- Les étapes détaillées de la configuration et de l'exécution des tests de performance doivent être expliquées pour montrer la relation entre les tests et les risques architecturaux.
- Les étapes requises pour rendre les tests de performance reproductibles doivent être communiquées. Il peut s'agir d'aspects organisationnels (p. ex. participation d'un personnel clé) ainsi que de questions techniques.
- Lorsque les environnements de test doivent être partagés, la planification des tests de performance doit être communiquée pour s'assurer que les résultats des tests ne seront pas affectés négativement.
- Les mesures d'atténuation de l'impact potentiel sur les utilisateurs réels si les tests de performance doivent être effectués dans l'environnement de production doivent être communiqués et acceptés.
- Les parties prenantes techniques doivent clairement être au fait de leurs tâches et de quand elles doivent être effectuées.

4.2 Analyse, Design et Implémentation

4.2.1 Protocoles de communication typiques

Les protocoles de communication définissent un ensemble de règles de communication entre les ordinateurs et les systèmes. Concevoir des tests correctement pour cibler des parties spécifiques du système nécessite de comprendre les protocoles.

Les protocoles de communication sont souvent décrits par les couches du modèle Open Systems Interconnection (OSI) (Voir ISO/IEC 7498-1), bien que certains protocoles puissent sortir de ce modèle. Pour les tests de performance, les protocoles de la couche 5 (couche de session) à la couche 7 (couche d'application) sont les plus couramment utilisés. Les protocoles courants comprennent :

- Base de données - ODBC, JDBC, autres protocoles spécifiques aux fournisseurs
- Web - HTTP, HTTPS, HTML
- Web Service - SOAP, REST

D'une manière générale, dans les tests de performance, le niveau de la couche OSI auquel on s'intéresse le plus dépend de l'architecture testée. Lors du test d'une

architecture à faible niveau, une architecture embarquée par exemple, les couches inférieures du modèle OSI seront principalement concernées.

D'autres protocoles utilisés dans les tests de performance comprennent :

- Réseau - DNS, FTP, IMAP, LDAP, POP3, SMTP, Windows Sockets, CORBA
- Mobile - TruClient, SMP, MMS
- Accès à distance- Citrix ICA, RTE
- SOA - MQSeries, JSON, WSCL

Il est important de comprendre l'architecture globale du système car les tests de performance peuvent être exécutés sur un composant individuel du système (p. ex., serveur Web, serveur de base de données) ou sur l'ensemble d'un système via des tests de bout en bout. Les applications traditionnelles 2-tier construites avec un modèle client-serveur spécifient le « client » comme interface utilisateur et interface utilisateur primaire, et le « serveur » comme base de données backend. Ces applications nécessitent l'utilisation d'un protocole tel que ODBC pour accéder à la base de données. Avec l'évolution des applications web et des architectures multi-tier, de nombreux serveurs sont impliqués dans le traitement d'informations qui sont finalement restituées par le navigateur de l'utilisateur.

Selon la partie du système qui est ciblée pour les tests, une compréhension du protocole approprié à utiliser est requise. Par conséquent, si le besoin est d'effectuer des tests de bout en bout émulant l'activité de l'utilisateur depuis un navigateur, un protocole Web tel que HTTP/HTTPS sera utilisé. De cette façon, l'interaction avec l'interface graphique peut être contournée et les tests peuvent se concentrer sur la communication et les activités des serveurs backend.

4.2.2 Transactions

Les transactions décrivent l'ensemble des activités effectuées par un système du point d'initiation au moment où un ou plusieurs processus (requêtes, opérations ou processus opérationnels) ont été achevés. Le temps de réponse des transactions peut être mesuré aux fins de l'évaluation du rendement du système. Lors d'un test de performance, ces mesures sont utilisées pour identifier les composants qui nécessitent une correction ou une optimisation.

Les transactions simulées peuvent inclure le temps de réflexion afin de mieux refléter le moment où un utilisateur réel prend des mesures (p. ex., appuyer sur le bouton « ENVOYER »). Le temps de réponse de la transaction plus le temps de réflexion équivaut au temps écoulé pour cette transaction.

Les temps de réponse des transaction collectés pendant les tests de performance montrent comment cette mesure change sous différentes charges imposées au système. L'analyse peut ne montrer aucune dégradation sous charge tandis que d'autres mesures peuvent montrer une dégradation grave. En augmentant la charge et en mesurant les temps de transaction sous-jacents, il est possible de corréler la cause de la dégradation avec les temps de réponse d'une ou de plusieurs transactions.

Les transactions peuvent également être imbriquées de sorte que les activités individuelles et agrégées puissent être mesurées. Cela peut être utile, par exemple, lorsque l'on comprend l'efficacité de performance d'un système de commande en ligne. Le testeur peut vouloir mesurer les étapes pas à pas du processus de commande (p. ex., recherche d'article, ajout d'élément au panier, payer l'article, confirmer la commande) ainsi que le processus de commande dans son ensemble. En imbriquant les transactions, les deux ensembles d'informations peuvent être recueillis en un seul test.

4.2.3 Identifier les profils opérationnels

Les profils opérationnels spécifient des modèles distincts d'interaction avec une application comme des utilisateurs ou d'autres composants du système. Plusieurs profils opérationnels peuvent être spécifiés pour une application donnée. Ils peuvent être combinés pour créer un profil de charge souhaité pour atteindre des objectifs de test de performance particuliers (voir Section 4.2.4).

Les principales étapes suivantes pour identifier les profils opérationnels sont décrites dans cette section :

1. Identifier les données à recueillir
2. Recueillir les données à l'aide d'une ou plusieurs sources
3. Évaluer les données pour construire les profils opérationnels

Identifier les données

Lorsque les utilisateurs interagissent avec le système sous test, les données suivantes sont recueillies ou estimées afin de modéliser leurs profils opérationnels (c.-à-d. comment ils interagissent avec le système) :

- Différents types de personae utilisateur et leurs rôles (p. ex., utilisateur standard, membre enregistré, administrateur, groupes d'utilisateurs ayant des privilèges spécifiques).
- Différentes tâches génériques effectuées par ces utilisateurs/rôles (p. ex., navigation sur un site Web pour obtenir de l'information, recherche sur un site Web pour un produit particulier, exécution d'activités spécifiques aux rôles). Notez que ces tâches sont généralement mieux modélisées à un niveau élevé d'abstraction (p. ex., au niveau des processus métier ou desuser stories).

- Nombre estimatif d'utilisateurs pour chaque rôle/tâche par unité de temps sur une période donnée. Ces informations seront également utiles pour la construction ultérieure de profils de charge (voir Section 4.2.4).

Recueillir des données

Les données mentionnées ci-dessus peuvent être recueillies à partir d'un certain nombre de sources différentes :

- Mener des entrevues ou des ateliers avec des parties prenantes, telles que les Product Owners, les directeurs des ventes et les utilisateurs finaux (potentiels). Ces discussions révèlent souvent les principaux profils opérationnels des utilisateurs et fournissent des réponses à la question fondamentale « À qui s'adresse cette application ».
- Les spécifications et les exigences fonctionnelles (lorsqu'elles sont disponibles) sont une source précieuse d'information sur les modèles d'utilisation prévus qui peuvent également aider à identifier les types d'utilisateurs et leurs profils opérationnels. Lorsque les spécifications fonctionnelles sont exprimées comme des User Stories, le format standard permet directement aux types d'utilisateurs d'être identifiés (c.-à-d., En tant que « *type d'utilisateur* », je veux « *une certaine capacité » ;* de sorte que *« un bénéfice »).* De même, les diagrammes et les descriptions de cas d'utilisation de l'UML identifient l'« acteur » pour le cas d'utilisation.
- L'évaluation des données d'utilisation et des métriques obtenues à partir d'applications similaires peut prendre en charge l'identification des types d'utilisateurs et fournir des indications initiales du nombre prévu d'utilisateurs. L'accès à des données surveillées automatiquement (p. ex., à partir d'un outil d'administration d'un web master) est recommandé. Cela inclura la surveillance des logs et des données provenant de l'utilisation du système opérationnel actuel où une mise à jour de ce système est prévue
- Le suivi du comportement des utilisateurs lors de l'exécution des tâches prédéfinies avec l'application peut donner un aperçu des types de profils opérationnels à modéliser pour les tests de performance. Il est recommandé de coordonner cette tâche avec tous les tests d'utilisabilité prévus (surtout si un laboratoire d'utilisabilité est disponible).

Construire des profils opérationnels

Les étapes suivantes sont suivies pour identifier et construire des profils opérationnels pour les utilisateurs :

- Une approche descendante est adoptée. Des profils opérationnels généraux relativement simples sont initialement créés et ne sont détaillés que si cela est nécessaire pour atteindre les objectifs de test de performance (voir Section 4.1.1)

- Les profils d'utilisateurs particuliers peuvent être désignés comme pertinents pour les tests de performance s'ils impliquent des tâches qui sont exécutées fréquemment, nécessitent des transactions critiques (à haut risque) ou fréquentes entre différents composants du système, ou exigent potentiellement de gros volumes de données à transférer.
- Les profils opérationnels sont examinés et affinés avec les principales parties prenantes avant d'être utilisés pour la création de profils de charge (voir Section 4.2.4).

Le système sous test n'est pas toujours soumis à des charges imposées par l'utilisateur. Des profils opérationnels peuvent également être nécessaires pour tester les performances des types de système suivants (veuillez noter que cette liste n'est pas exhaustive) :

Systèmes de traitement des lots (NDT : batch) hors ligne
L'accent est mis ici principalement sur le débit du système de traitement par lots (voir Section 4.2.5) et sa capacité à terminer dans un délai donné. Les profils opérationnels se concentrent sur les types de traitement qui sont exigés des processus de lot. Par exemple, les profils opérationnels d'un système de négociation d'actions (qui comprend généralement le traitement des transactions en ligne et par lots) peuvent inclure ceux relatifs aux transactions de paiement, à la vérification des informations d'identification et à la vérification de la conformité des conditions légales pour certains types de transactions boursières. Chacun de ces profils opérationnels se traduirait par des chemins différents qui passeraient par le processus global de lot pour une action. Les étapes décrites ci-dessus pour identifier les profils opérationnels des systèmes basés sur les utilisateurs en ligne peuvent également être appliquées dans le contexte du traitement par lots.

Systèmes de systèmes
Les composants d'un environnement multi systèmes (qui peuvent également être embarqués) répondent à différents types d'entrée provenant d'autres systèmes ou composants. Selon la nature du système sous test, cela peut nécessiter la modélisation de plusieurs profils opérationnels différents pour représenter efficacement les types d'intrants fournis par ces systèmes qui les fournissent. Cela peut impliquer une analyse détaillée (p. ex. des tampons et des files d'attente) avec les architectes du système et en fonction des spécifications du système et de l'interface.

4.2.4 Création de profils de charge

Un profil de charge précise l'activité qu'un composant ou un système testé peut connaître en production. Il se compose d'un nombre désigné d'instances qui effectueront les actions des profils opérationnels prédéfinis sur une période

déterminée. Lorsque les instances sont des utilisateurs, le terme « utilisateurs virtuels » est couramment employé.

Les principales informations requises pour créer un profil de charge réaliste et reproductible sont :

- L'objectif de test de performance (p. ex., évaluer le comportement du système sous les charges de stress)
- Profils opérationnels qui représentent avec précision les modèles d'utilisation individuels (voir Section 4.2.3)
- Problèmes connus de débit et de concurrence (voir Section 4.2.5)
- La quantité et la répartition du temps avec lesquelles les profils opérationnels doivent être exécutés de telle sorte que le SST connaît la charge désirée. Les exemples typiques sont :
    - Montée en charge progressive : Augmentation constante de la charge (p. ex., ajouter un utilisateur virtuel par minute)
    - Décroissance de charge progressive : Charge en baisse constante
    - Étapes : Changements instantanés de charge (p. ex., ajouter 100 utilisateurs virtuels toutes les cinq minutes)
    - Distributions prédéfinies (p. ex., le volume imite les cycles d'affaires quotidiens ou saisonniers)

L'exemple suivant montre la construction d'un profil de charge avec l'objectif de générer des conditions de stress (au-delà du maximum prévu pour un système à manipuler) pour le système sous test.

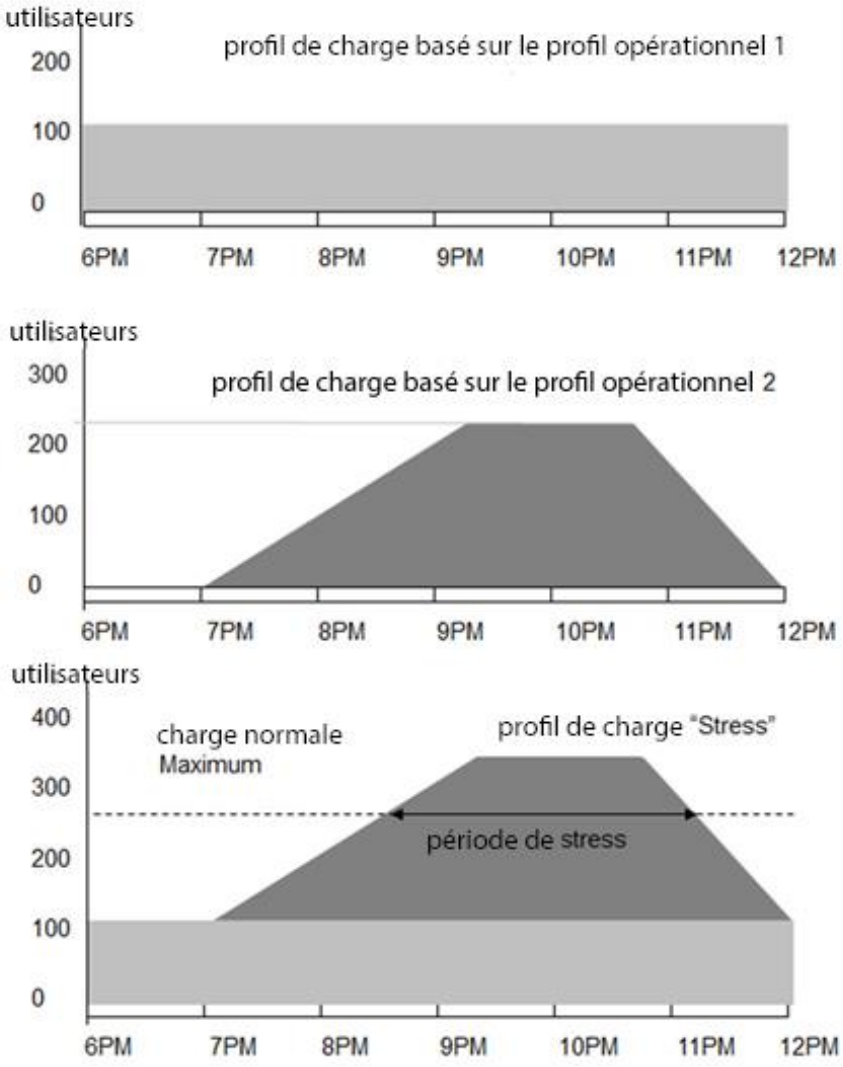

Diagramme 1 : Exemple de construction d'un profil de charge « stress »

En haut du diagramme, le profil de charge affiche l'entrée de 100 utilisateurs virtuels en une étape. Ces utilisateurs effectuent les activités définies par le profil opérationnel 1 sur toute la durée du test. Ceci est typique de nombreux profils de charge de performance qui représentent une charge d'arrière-plan.

Le diagramme du milieu montre un profil de charge qui se compose d'une rampe de montée en charge jusqu'à 220 utilisateurs virtuels, maintenue pendant deux heures avant de redescendre. Chaque utilisateur virtuel effectue les activités définies dans le profil opérationnel 2.

Le diagramme inférieur montre le profil de charge qui résulte de la combinaison des deux. Le système sous test est soumis à une période de stress de trois heures.

Pour d'autres exemples, se référer à [Bath14].

4.2.5 Analyse du débit et de la concurrence

Il est important de comprendre les différents aspects de la charge de travail : le débit et la concurrence. Pour modéliser correctement les profils opérationnels et les profils de charge, ces deux aspects doivent être pris en considération.

Débit du système

Le débit du système est une mesure du nombre de transactions d'un type donné que le système traite par unité de temps. Par exemple, le nombre de commandes par heure ou le nombre de requêtes HTTP par seconde. Le débit du système doit être distingué du débit du réseau, qui est la quantité de données déplacées sur le réseau (Section 2.1).

Le débit du système définit la charge sur le système. Malheureusement, très souvent, le nombre d'utilisateurs simultanés est utilisé, au lieu du débit, pour définir la charge pour les systèmes interactifs. Cela est partiellement vrai parce que ce nombre est souvent plus facile à trouver, et en partie aussi parce que c'est la façon dont les outils de test de charge définissent la charge. Sans définir les profils opérationnels - ce que chaque utilisateur fait et avec quelle intensité (qui représente le débit pour un utilisateur) - le nombre d'utilisateurs n'est pas une bonne mesure de la charge. Par exemple, s'il y a 500 utilisateurs qui lancent des requêtes courtes chaque minute, nous avons un débit de 30 000 requêtes par heure. Si les mêmes 500 utilisateurs exécutent les mêmes requêtes, mais une par heure, le débit est de 500 requêtes par heure. Ce sont donc les mêmes 500 utilisateurs, mais cela représente une différence de 60x entre les charges et au moins une différence de 60x en besoins matériels pour le système.

La modélisation de la charge de travail se fait généralement en tenant compte du nombre d'utilisateurs virtuels (threads d'exécution) et du temps de réflexion (retards entre les actions des utilisateurs). Cependant, le débit du système est également défini par le temps de traitement, et ce temps peut augmenter à mesure que la charge augmente.

Débit du système = [nombre d'utilisateurs virtuels] / ([temps de traitement] + [temps de réflexion])

Ainsi, lorsque le temps de traitement augmente, le débit peut diminuer considérablement même si tout le reste est inchangé.

Le débit du système est un aspect important lors du test des systèmes de traitement par lots. Dans ce cas, le débit est généralement mesuré en fonction du nombre de transactions qui peuvent être effectuées en un temps donné (p. ex., une fenêtre de temps allouée à un traitement par lots tournant de nuit).

Concurrence

La concurrence est une mesure du nombre de tâches (NDT : threads) simultanées/ parallèles d'exécution. Pour les systèmes interactifs, il peut s'agir d'un certain nombre d'utilisateurs simultanés/parallèles. La concurrence est généralement modélisée dans les outils de test de charge en définissant le nombre d'utilisateurs virtuels.
La concurrence est une mesure importante. Elle représente le nombre de sessions parallèles, chacune pouvant utiliser ses propres ressources. Même si le débit est le même, la quantité de ressources utilisées peut différer selon la concurrence. Les configurations de test typiques sont des systèmes fermés (du point de vue de la théorie de la file d'attente), où le nombre d'utilisateurs dans le système est défini (population fixe). Si tous les utilisateurs attendent la réponse du système dans un système fermé, aucun nouvel utilisateur ne peut arriver. De nombreux systèmes publics sont des systèmes ouverts : de nouveaux utilisateurs arrivent tout le temps même si tous les utilisateurs actuels attendent la réponse du système.

4.2.6 Structure de base d'un script de test de performance

Un script de test de performance doit simuler une activité d'un utilisateur ou d'un composant, qui contribue à la charge sur le système sous test (qui peut être l'ensemble du système ou l'un de ses composants). Il envoie des requêtes au serveur dans un ordre approprié et à un rythme donné.

La meilleure façon de créer des scripts de test de performance dépend de l'approche de génération de charge utilisée (Section 4.1).

- La façon traditionnelle est d'enregistrer la communication entre le client et le système ou le composant au niveau du protocole, puis de la rejouer après que le script a été paramétré et documenté. La paramétrisation se traduit par un script pouvant passer à l'échelle et maintenable, mais la tâche de paramétrisation peut prendre du temps.
- L'enregistrement au niveau de l'interface graphique implique généralement la capture des actions effectuées sur l'interface graphique d'un seul client, avec un outil d'exécution de test, que l'on exécute ensuite avec l'outil de génération de charge pour représenter plusieurs clients.
- La programmation peut être effectuée à l'aide de requêtes de protocole (p. ex., requêtes HTTP), d'actions via l'interface utilisateur ou d'appels API. Dans le cas des scripts de programmation, la séquence exacte des requêtes envoyées et reçues du système réel doit être déterminée, ce qui peut ne pas être trivial.

Habituellement, un script consiste en une ou plusieurs sections de code (écrites dans un langage de programmation générique avec quelques extensions ou dans un langage spécialisé), ou un objet, qui peut être présenté à un utilisateur par l'outil au travers d'une interface graphique. Dans les deux cas, le script comprendra des requêtes de serveur créant une charge (p. ex., des requêtes HTTP) et une certaine logique de programmation les accompagnant afin de spécifier comment exactement ces requêtes seraient invoquées (p. ex., dans quel ordre, à quel moment, avec quels paramètres, ce qui devrait être vérifié). Plus la logique est sophistiquée, plus il est nécessaire d'utiliser des langages de programmation puissants.

Structure globale
Souvent, le script comporte une section d'initialisation (où tout est préparé pour la partie principale), des sections principales qui peuvent être exécutées plusieurs fois, et une section de « nettoyage » (où les mesures nécessaires sont prises pour terminer le test correctement).

Collecte de données
Pour recueillir les temps de réponse, des compteurs doivent être ajoutées au script pour mesurer combien de temps prend une requête ou une combinaison de requêtes. Les demandes chronométrées doivent correspondre à une unité de travail logique significative, par exemple, une transaction commerciale pour l'ajout d'un élément à une commande ou la soumission d'une commande.

Il est important de comprendre ce qui est exactement mesuré : dans le cas des scripts au niveau du protocole, il s'agit uniquement du temps de réponse du serveur et du réseau, tandis que les scripts GUI mesurent le temps de bout-en-bout (bien que ce qui est exactement mesuré dépend de la technologie utilisée).

Vérification des résultats et traitement des erreurs
Une partie importante du script est la vérification des résultats et la gestion des erreurs. Même dans les meilleurs outils de test de charge, la gestion d'erreur par défaut a tendance à être minimaliste (comme de vérifier le code de retour d'une requête HTTP), il est donc recommandé d'ajouter des indicateurs supplémentaires pour vérifier ce que les requêtes renvoient réellement. En outre, si un « nettoyage » est nécessaire en cas d'erreur, il devra probablement être mis en œuvre manuellement. Une bonne pratique consiste à vérifier que le script fait ce qu'il est censé faire en utilisant des méthodes indirectes, par exemple, en vérifiant dans la base de données que les informations appropriées ont bien été ajoutées.

Les scripts peuvent inclure d'autres règles pour spécifier la logique définissant quand et comment les requêtes au serveur seront faites. Un exemple consiste à définir des

points de synchronisation, en spécifiant que le script doit attendre un événement à ce moment donné avant de continuer. Les points de synchronisation peuvent être utilisés pour s'assurer qu'une action spécifique est invoquée simultanément ou pour coordonner le travail entre plusieurs scripts.

Les scripts de test de performance sont des logiciels, donc la création d'un script de test de performance est une activité de développement de logiciels. Il devrait inclure l'assurance de la qualité et des tests pour vérifier que le script fonctionne comme prévu avec toute la gamme de données d'entrée.

4.2.7 Implémentation des scripts de test de performance

Les scripts de test de performance sont implémentés en fonction du PTP et des profils de charge. Bien que les détails techniques de la mise en œuvre diffèrent selon l'approche et (les) / l'outil(s) utilisés, le processus global reste le même. Un script de performance est créé à l'aide d'un environnement de développement intégré (EDI) ou d'un éditeur de script, pour simuler un comportement d'un utilisateur ou d'un composant. Habituellement, le script est créé pour simuler un profil opérationnel spécifique (bien qu'il soit souvent possible de combiner plusieurs profils opérationnels dans un script avec des déclarations conditionnelles).

Au fur et à mesure que la séquence des requêtes est déterminée, le script peut être enregistré ou programmé en fonction de l'approche. L'enregistrement garantit généralement qu'il simule exactement le système réel, tandis que la programmation repose sur la connaissance de la séquence de requêtes appropriée.

Si l'enregistrement est effectué au niveau du protocole, dans la plupart des cas, une étape essentielle après l'enregistrement consiste à remplacer tous les identificateurs internes enregistrés qui définissent le contexte. Ces identificateurs doivent être transformés en variables qui peuvent être modifiées, entre les exécutions, avec des valeurs appropriées extraites des réponses à la requête (p. ex., un identifiant utilisateur qui est acquis lors de la connexion et doit être fourni pour toutes les transactions ultérieures). Il s'agit d'une partie de la paramétrisation du script, parfois appelée « corrélation ». Dans ce contexte, le mot corrélation prend une signification différente de celle utilisée dans les statistiques (où il désigne la relation entre deux ou plusieurs choses). Les outils avancés de test de charge peuvent automatiquement faire une certaine corrélation, de sorte qu'elle peut être transparente dans certains cas, mais dans des cas plus complexes, une corrélation manuelle ou l'ajout de nouvelles règles de corrélation peuvent être nécessaires. Une corrélation incorrecte ou un manque de corrélation constituent la principale raison pour laquelle les scripts enregistrés ne sont pas rejouables.

L'exécution de plusieurs utilisateurs virtuels avec le même nom d'utilisateur ayant accès au même ensemble de données (comme c'est habituellement le cas lors du rejeu d'un script enregistré sans autre modification au-delà de la corrélation nécessaire) est un moyen simple d'obtenir des résultats trompeurs. Les données pourraient être entièrement mises en cache (copiées depuis le disque dans la mémoire pour un accès plus rapide) et les résultats seraient alors beaucoup meilleurs qu'en production (où ces données peuvent être lues à partir d'un disque). L'utilisation des mêmes utilisateurs et/ou données peut également causer des problèmes de concurrence (p. ex., si les données sont verrouillées lorsqu'un utilisateur les met à jour) et les résultats seraient alors bien pires qu'en production, car le logiciel attendrait le déverrouillage des données avant que le prochain utilisateur puisse, à son tour, les verrouiller pour la mise à jour).

Ainsi, les scripts et les harnais de test doivent être paramétrés (c.-à-d. que les données fixes ou enregistrées doivent être remplacées par des valeurs provenant d'une liste de choix possibles), de sorte que chaque utilisateur virtuel utilise un ensemble approprié de données. Le terme « approprié » signifie ici assez différent pour éviter les problèmes de mise en cache et de concurrence, ce qui est spécifique pour le système, les données et les exigences de test. Cette paramétrisation supplémentaire dépend des données dans le système et de la façon dont le système fonctionne avec ces données, de sorte qu'elle est généralement faite manuellement, bien que de nombreux outils fournissent une assistance en la matière.

Il y a des cas où certaines données doivent être paramétrées pour que le test fonctionne plus d'une fois, par exemple, lorsqu'une commande est créée et que le nom de la commande doit être unique. À moins que le nom de la commande ne soit paramétré, le test échouera dès qu'il tentera de créer une commande avec un nom existant (enregistré).

Pour correspondre aux profils opérationnels, les temps de réflexion doivent être insérés et/ou ajustés (s'ils sont enregistrés) pour générer un nombre approprié de requêtes/ de débit comme nous l'avons expliqué dans la Section 4.2.5.

Lorsque des scripts sont créés pour des profils opérationnels distincts, ils sont combinés dans un scénario mettant en œuvre l'ensemble du profil de charge. Le profil de charge contrôle le nombre d'utilisateurs virtuels qui commencent à utiliser chaque script, quand et avec quels paramètres. Les détails exacts de l'implémentation dépendent de l'outil spécifique de test de charge ou du harnais de test.

4.2.8 Préparation à l'exécution des tests de performance

Les principales activités de préparation à l'exécution des tests de performance sont :

- Mise en place du système sous test

- Déploiement de l'environnement
- Mise en place les outils de génération et de surveillance de la charge en s'assurant que toutes les informations nécessaires seront collectées

Il est important de s'assurer que l'environnement de test est aussi proche que possible de l'environnement de production. Si cela n'est pas possible, il doit y avoir une compréhension claire des différences et de la façon dont les résultats des tests se projetteront sur l'environnement de production. Idéalement, l'environnement de production et les données réelles devraient être utilisés, mais les tests dans un environnement réduit peuvent encore aider à atténuer un certain nombre de risques de performance.

Il est important de se rappeler que la performance est une fonction non linéaire de l'environnement, de sorte que plus l'environnement est éloigné de la norme de production, plus il devient difficile de faire des projections précises quant à la performance en production. Le manque de fiabilité des projections et l'augmentation du niveau de risque augmentent à mesure que le système de test s'écarte de la production.

Les parties les plus importantes de l'environnement de test sont les données, la configuration matérielle et logicielle, et la configuration du réseau. La taille et la structure des données pourraient affecter considérablement les résultats des tests de charge. L'utilisation d'un ensemble de données réduit ou d'un ensemble de données ayant une complexité différente, lors des tests de performance, peut donner des résultats trompeurs, en particulier quand le système de production utilise un grand ensemble de données. Il est difficile de prédire dans quelle mesure la taille des données affecte les performances avant que les tests réels ne soient effectués. Plus les données de test, en taille et en structure, sont proches des données de production, plus les résultats des tests seront fiables.

Si les données sont générées ou modifiées pendant le test, il peut être nécessaire de restaurer les données d'origine avant le prochain cycle de test pour s'assurer que le système est dans l'état approprié.

Si, pour une raison quelconque, certaines parties du système ou certaines données ne sont pas disponibles pour les tests de performance, une solution de contournement doit être mise en œuvre. Par exemple, un bouchon peut être mis en œuvre pour remplacer et émuler un composant tiers responsable du traitement des cartes de crédit. Ce processus est souvent appelé « virtualisation des services » et il existe des outils spéciaux disponibles pour le faciliter. L'utilisation de ces outils est fortement recommandée pour isoler le système sous test.

Il existe de nombreuses façons de déployer des environnements. Par exemple, les options peuvent inclure l'une ou l'autre des pratiques qui suivent :

- Laboratoires de test traditionnels internes (et externes)
- Le cloud comme environnement utilisant « Infrastructure as a Service » (IaaS), lorsque certaines parties ou la totalité du système est déployée dans le cloud
- Le cloud comme environnement utilisant « Software as a Service » (SaaS), lorsque les fournisseurs fournissent le service de test de charge

Selon les objectifs spécifiques et les systèmes à tester, un environnement de test peut être préféré à un autre. Par exemple,

- Pour tester l'effet d'une amélioration des performances (optimisation des performances), l'utilisation d'un environnement de laboratoire isolé peut être une meilleure option pour voir même de petites variations introduites par le changement.
- Pour charger l'ensemble de l'environnement de production de bout en bout pour s'assurer que le système gérera la charge sans aucun problème majeur, les tests à partir du cloud ou d'un service peuvent être plus appropriés. (Notez que cela ne fonctionne que pour les SST accessibles à partir d'un cloud).
- Pour minimiser les coûts lorsque les tests de performance sont limités dans le temps, la création d'un environnement de test dans le cloud peut être une solution plus économique.

Quelle que soit l'approche de déploiement utilisée, le matériel et le logiciel doivent être configurés pour atteindre l'objectif et le plan du test. Si l'environnement correspond à la production, il doit être configuré de la même manière. Toutefois, s'il y a des différences, la configuration peut devoir être ajustée pour tenir compte de celles-ci. Par exemple, si les machines de test ont moins de mémoire physique que les machines de production, les paramètres de mémoire logicielle (tels que la taille du tas Java – NDT : heap) peuvent devoir être ajustés pour éviter le « paging » de mémoire.

Une configuration /émulation appropriée du réseau est importante pour les systèmes mondiaux et mobiles. Pour les systèmes mondiaux (c'est-à-dire un système qui a des utilisateurs ou un traitement distribué dans le monde entier), l'une des approches peut être de déployer des générateurs de charge dans les endroits où les utilisateurs sont situés. Pour les systèmes mobiles, l'émulation du réseau reste l'option la plus viable en raison des variations existant dans les types de réseau qui peuvent être utilisés. Certains outils de test de charge incluent des outils d'émulation du réseau et il existe aussi des outils autonomes pour ce faire.

Les outils de génération de charge doivent être correctement déployés et les outils de surveillance doivent être configurés pour recueillir toutes les métriques nécessaires pour le test. La liste des métriques dépend des objectifs du test, mais il est recommandé de recueillir au moins des métriques de base pour tous les tests (voir Section 2.1.2).

Selon la charge, l'approche spécifique de génération d'outils/charge, et la configuration de la machine, plus d'une machine de génération de charge peut être nécessaire. Pour vérifier la configuration, les machines impliquées dans la génération de charge doivent également être surveillées. Cela permettra d'éviter une situation où la charge n'est pas maintenue correctement parce que l'un des générateurs de charge fonctionne lentement.

Selon la configuration et les outils utilisés, les outils de test de charge doivent être configurés pour créer la charge appropriée. Par exemple, des paramètres spécifiques d'émulation du navigateur peuvent être définis ou l'usurpation d'IP (simulant que chaque utilisateur virtuel a une adresse IP différente) peut être utilisée.

Avant l'exécution des tests, l'environnement et la configuration doivent être validés. Cela se fait généralement en effectuant un ensemble contrôlé de tests et en vérifiant les résultats de même qu'on vérifie que les outils de surveillance suivent l'information importante.

Pour vérifier que le test fonctionne tel qu'il est conçu, une variété de techniques peut être utilisées, y compris l'analyse des logs et la vérification du contenu de la base de données. La préparation du test comprend la vérification que les informations requises sont journalisées, que le système est dans l'état approprié, etc. Par exemple, si le test modifie considérablement l'état du système (ajouter/modifier les informations dans la base de données), il peut être nécessaire de remettre le système dans l'état d'origine avant de répéter le test.

4.3 Exécution

L'exécution des tests de performance implique la génération d'une charge appliquée au SST selon un profil de charge (généralement mis en œuvre par des scripts de test de performance invoqués selon un scénario donné), la surveillance de toutes les parties de l'environnement, et la collecte et la conservation de tous les résultats et informations liés au test. Habituellement les outils de test de charge avancés / les harnais effectuent ces tâches automatiquement (après, bien sûr, une configuration appropriée). Ils fournissent généralement une console pour permettre de surveiller les données de performance pendant le test et de procéder aux ajustements nécessaires

(voir Section 5.1). Toutefois, selon l'outil utilisé, le SST et les tests spécifiques exécutés, certaines étapes manuelles peuvent être nécessaires.

Les tests de performance sont généralement axés sur un état stable du système, c'est-à-dire lorsque le comportement du système est stable. Par exemple, lorsque tous les utilisateurs/threads simulés sont lancés et exécutent des travaux tels que prévus. Lorsque la charge change (par exemple, lorsque de nouveaux utilisateurs sont ajoutés), le comportement du système change aussi et il devient plus difficile de surveiller et d'analyser les résultats des tests. L'étape pour arriver à l'état stable est souvent appelée la montée en charge, et l'étape de la fin du test est souvent appelée la décroissance de charge.

Il est parfois important de tester les états transitoires, lorsque le comportement du système change. Cela peut s'appliquer, par exemple, à l'ouverture de session simultanée d'un grand nombre d'utilisateurs ou des tests de pics. Lors du test des états transitoires, il est important de comprendre la nécessité d'une surveillance et d'une analyse minutieuses des résultats, car certaines approches standard, comme la surveillance des moyennes, peuvent être complètement erronées.

Pendant la montée en charge, il est conseillé de mettre en œuvre des états de charge incrémentaux pour surveiller l'impact de la charge croissante sur la réponse du système. Cela garantit que suffisamment de temps est alloué à la montée en charge et que le système est en mesure de gérer la charge. Une fois que l'état stable a été atteint, une bonne pratique consiste à surveiller que la charge et les réponses du système sont stables et que les variations aléatoires (qui existent toujours) ne sont pas substantielles.

Il est important de préciser comment les défaillances doivent être traitées pour s'assurer qu'aucun problème n'est introduit au niveau du système. Par exemple, il peut être important pour l'utilisateur d'interrompre sa session quand un échec se produit pour s'assurer que toutes les ressources associées à cet utilisateur sont libérées.

Si la surveillance est intégrée à l'outil de test de charge et qu'elle est correctement configurée, elle commence habituellement en même temps que l'exécution du test. Toutefois, si des outils de surveillance autonomes sont utilisés, la surveillance doit être démarrée séparément et les informations nécessaires recueillies de manière que l'analyse ultérieure puisse être effectuée en même temps que les résultats des tests. Il en va de même pour l'analyse des logs. Il est essentiel de synchroniser tous les outils utilisés, afin que toutes les informations relatives à un cycle spécifique d'exécution des tests puissent être localisées.

L'exécution des tests est souvent surveillée à l'aide de la console de l'outil de test de performance et de l'analyse des logs en temps réel pour vérifier les problèmes et les erreurs dans le test et le SST. Cela permet d'éviter de continuer inutilement à exécuter des tests à grande échelle, ce qui pourrait même avoir un impact sur d'autres systèmes si les choses tournent mal (p. ex., si une défaillance se produit, des composants échouent ou que les charges générées sont trop faibles ou élevées). Ces tests peuvent être coûteux à exécuter et il peut être nécessaire d'arrêter le test ou de faire quelques ajustements à la volée au test de performance ou à la configuration du système si le test s'écarte du comportement attendu.

Une technique pour vérifier les tests de charge qui communiquent directement au niveau du protocole consiste à exécuter plusieurs scripts de niveau GUI (fonctionnels) ou même à exécuter manuellement des profils opérationnels similaires en parallèle au test de charge en cours d'exécution. Ceci contrôle que les temps de réponse rapportés pendant le test ne diffèrent des temps de réponse mesurés manuellement au niveau de l'interface graphique que par le temps passé du côté client.

Dans certains cas, lorsqu'on exécute des tests de performance d'une manière automatisée (par exemple, dans le cadre de l'intégration continue, comme cela a été discuté dans Section 3.4) les contrôles doivent être effectués automatiquement, car la surveillance manuelle et l'intervention peuvent ne pas être possibles. Dans ce cas, le test mis en place devrait être en mesure de reconnaître les écarts ou les problèmes et d'émettre une alerte (généralement tout en remplissant correctement le test). Cette approche est plus facile à mettre en œuvre pour les tests de performance de régression lorsque le comportement du système est généralement connu, mais peut être plus difficile avec des tests de performance exploratoires ou des tests de performance coûteux à grande échelle qui peuvent avoir besoin d'ajustements dynamiques pendant le test.

4.4 Analyse des résultats et reporting

La section 4.1.2 a discuté des différentes métriques d'un plan de test de performance. La définition de ces éléments à l'avance détermine ce qui doit être mesuré pour chaque exécution de test. Une fois le cycle de test terminé, les données doivent être recueillies pour les métriques définies.

Lors de l'analyse des données, on compare d'abord à l'objectif de test de performance. Une fois le comportement compris, des conclusions peuvent être tirées qui fournissent un rapport sommaire significatif qui inclut les actions recommandées. Ces mesures peuvent inclure la modification des composants physiques (p. ex., matériel, routeurs), la modification de logiciels (p. ex., l'optimisation des applications et des appels de base de données) et la modification du réseau (p. ex., l'équilibrage des charges, le routage).

Les données suivantes sont généralement analysées :

- **Statut des utilisateurs simulés (p. ex., virtuels)**. Cela doit être examiné en premier. On s'attend normalement à ce que tous les utilisateurs simulés aient été en mesure d'accomplir les tâches spécifiées dans le profil opérationnel. Toute interruption de cette activité peut ainsi imiter l'expérience d'un utilisateur réel. Il est donc très important de voir d'abord que toute activité utilisateur est terminée car toute erreur rencontrée peut influencer les autres données de performances.
- **Temps de réponse à la transaction**. Cela peut être mesuré de plusieurs façons, y compris le minimum, le maximum, la moyenne et un percentile (p. ex., 90^{e}). Les lectures minimales et maximales montrent les extrêmes des performances du système. Le rendement moyen n'est pas nécessairement indicatif d'autre chose que la moyenne mathématique et peut souvent être biaisé par des valeurs aberrantes. Le 90^{e} percentile est souvent utilisé comme un objectif car il représente la majorité des utilisateurs atteignant un seuil de performance spécifique. Il n'est pas recommandé d'exiger une conformité à 100 % avec les objectifs de rendement, car les ressources requises peuvent être trop importantes et l'effet net pour les utilisateurs sera souvent mineur.
- **Transactions par seconde**. Cela fournit des informations sur la quantité de travail effectué par le système (débit du système).
- **Échecs transactionnels**. Ces données sont utilisées lors de l'analyse des transactions par seconde. Les défaillances indiquent que l'événement ou le processus prévu n'a pas été terminé ou qu'il ne s'est pas exécuté. Les défaillances rencontrées sont préoccupantes et la cause racine doit être investiguée. L'échec des transactions peut également entraîner des transactions invalides par seconde, car une transaction échouée prendra beaucoup moins de temps qu'une transaction réussie.
- **Accès (ou requêtes) par seconde**. Cela donne une idée du nombre d'accès à un serveur par les utilisateurs simulés au cours de chaque seconde du test.
- **Débit réseau.** Ceci est généralement mesuré en bits par intervalle de temps, comme en bits par seconde. Cela représente la quantité de données que les utilisateurs simulés reçoivent du serveur chaque seconde. (Voir Section 4.2.5)
- **Réponses HTTP.** Ceux-ci sont mesurés par seconde et comprennent des codes de réponse possibles tels que : 200, 302, 304, 404, ce dernier indiquant qu'une page n'est pas trouvée.

Bien qu'une grande partie de ces informations puissent être présentées dans les tableaux, les représentations graphiques facilitent la visualisation des données et l'identification des tendances.

Les techniques utilisées dans l'analyse des données peuvent inclure :

- Comparaison des résultats aux exigences énoncées
- Observation des tendances des résultats
- Techniques de contrôle de la qualité statistique
- Identification des erreurs
- Comparaison des résultats attendus et réels
- Comparaison des résultats aux résultats antérieurs des tests
- Vérification du bon fonctionnement des composants (p. ex. serveurs, réseaux)

L'identification de la corrélation entre les mesures peut nous aider à comprendre à quel moment les performances du système commencent à se dégrader. Par exemple, quel nombre de transactions par seconde ont été traitées lorsque le processeur a atteint une capacité de 90 % et que le système a ralenti ?

L'analyse peut aider à identifier la cause racine de la dégradation ou de l'échec des performances, ce qui facilitera la correction. Les tests de confirmation aideront à déterminer si les mesures correctives ont bien pris en compte la cause racine.

Reporting

Les résultats de l'analyse sont consolidés et comparés aux objectifs énoncés dans le plan de test de performance. Ceux-ci peuvent être rapportés dans le rapport global d'état des tests, ainsi que d'autres résultats des tests, ou inclus dans un rapport dédié aux tests de performance. Le niveau de détail rapporté doit correspondre aux besoins des parties prenantes. Les recommandations fondées sur ces résultats portent généralement sur les critères de livraison de logiciels (y compris l'environnement cible) ou les améliorations requises des performances.

Un rapport typique de test de performance peut inclure :

Résumé

Cette section est achevée une fois que tous les tests de performance ont été effectués et que tous les résultats ont été analysés et compris. L'objectif est de présenter des conclusions, des constatations et des recommandations concises et compréhensibles pour le management dans le but d'obtenir un résultat réalisable.

Résultats des tests

Les résultats des tests peuvent inclure une partie ou la totalité des informations suivantes :

- Un résumé fournissant une explication et une élaboration des résultats.
- Les résultats d'un test de référence qui sert d'« instantané » de la performance du système à un moment donné et constitue la base de la comparaison avec les tests ultérieurs. Les résultats doivent inclure la date/heure du début du test, l'objectif de l'utilisateur concurrent, le débit mesuré et les principales constatations. Les principales constatations peuvent inclure le taux d'erreur global mesuré, le temps de réponse et le débit moyen.
- Un diagramme de haut niveau montrant tous les composants architecturaux qui pourraient (ou ont effectivement) impacté les objectifs de test.
- Une analyse détaillée (tableaux et graphiques) des résultats des tests montrant les temps de réponse, les taux de transaction, les taux d'erreur et l'analyse des performances. L'analyse comprend également une description de ce qui a été observé, par exemple à quel moment une application stable est devenue instable et la source de défaillances (p. ex., serveur Web, serveur de base de données).

Logs de test / informations enregistrées

Un log de chaque exécution de test doit être enregistré. Le log comprend généralement :

- Date/heure du début du test
- Durée du test
- Scripts utilisés pour le test (y compris le mix de scripts si plusieurs scripts sont utilisés) et les données pertinentes de configuration de script
- Fichier(s) de données de test utilisé(s)
- Nom et localisation des fichiers de données/logs créés lors du test
- Configuration HW/SW testée (en particulier tout changement entre les exécutions)
- Utilisation moyenne et maximale du processeur et de la RAM sur les serveurs Web et bases de données
- Notes sur les performances obtenues
- Défauts identifiés

Recommandations

Les recommandations résultant des tests peuvent inclure :

- Modifications techniques recommandées, telles que la reconfiguration du matériel ou des logiciels ou de l'infrastructure réseau

- Domaines identifiés pour une analyse plus approfondie (p. ex., analyse des logs de serveurs Web pour aider à identifier les causes racines des problèmes et/ou des erreurs)
- Surveillance supplémentaire requise des passerelles, des serveurs et des réseaux afin d'obtenir des données plus détaillées pour mesurer les caractéristiques et les tendances des performances (p. ex., dégradation)

5. Outils – 90 mins.

Mots-clés

Générateur de charge, gestion de charge, outil de surveillance, outil de test de performance

Objectifs d'apprentissage

5.1 Prise en charge par les outils

PTFL-5.1.1 (K2) Comprendre comment les outils prennent en charge les tests de performance

5.2 Pertinence des outils

PTFL-5.2.1 (K4) Évaluer la pertinence des outils de test de performance dans un scénario de projet donné

5.1 Prise en charge par les outils

Les outils de test de performance comprennent les types d'outils suivants pour soutenir les tests de performance.

Générateurs de charge

Le générateur, grâce à un IDE, un éditeur de scripts ou une suite d'outils, est capable de créer et d'exécuter plusieurs instances client qui simulent le comportement de l'utilisateur selon un profil opérationnel défini. La création de plusieurs instances en de courtes périodes de temps entraînera une charge sur un système sous test. Le générateur crée la charge et recueille également des métriques pour les rapports ultérieurs.

Lors de l'exécution des tests de performance, l'objectif du générateur de charge est d'imiter le monde réel autant qu'il est pratique. Cela signifie souvent que les demandes des utilisateurs provenant de divers endroits sont nécessaires, et pas seulement à partir du lieu où les tests sont effectués. Les environnements qui sont configurés avec plusieurs points de présence répartiront la charge selon sa provenance afin qu'elle ne vienne pas toute d'un seul réseau. Cela rend le test réaliste, mais cela peut aussi parfois fausser les résultats si les points de réseau intermédiaires créent des retards.

Console de gestion de la charge

La console de gestion de la charge fournit le contrôle pour démarrer et arrêter le générateur de charge(s). La console agrège également les métriques issues des différentes transactions qui sont définies dans les instances de charge utilisées par le

générateur. La console permet de visualiser les rapports et les graphiques des exécutions de test et prend en charge l'analyse des résultats.

Outil de surveillance
Les outils de surveillance fonctionnent en même temps que le composant ou le système sous test et supervisent, enregistrent et/ou analysent le comportement du composant ou du système. Les composants typiques qui sont surveillés comprennent les files d'attente des serveurs Web, la mémoire du système et l'espace sur le disque. Les outils de surveillance peuvent prendre en charge efficacement l'analyse des causes racines de la dégradation des performances dans un système sous test et peuvent également être utilisés pour surveiller un environnement de production lorsque le produit est livré. Pendant l'exécution des tests de performance, les moniteurs peuvent également suivre le générateur de charge lui-même.

Les modèles de licence pour les outils de test de performance comprennent la licence traditionnelle basée par utilisateur/site avec pleine propriété, un modèle de licence payante basé sur l'utilisation dans le cloud et des licences open source qui sont libres d'utilisation dans un environnement défini ou au travers d'offres basées sur le cloud. Chaque modèle implique une structure de coûts différente et peut inclure une maintenance continue. Ce qui est clair, c'est que pour tout outil sélectionné, la compréhension de son fonctionnement (par la formation et/ou l'étude libre) exigera du temps et du budget.

5.2 Pertinence de l'outil

Les facteurs suivants doivent être pris en considération lors de la sélection d'un outil de test de performance :

Compatibilité
En général, un outil est sélectionné pour une organisation et pas seulement pour un projet. Cela signifie que l'on doit tenir compte des facteurs suivants dans l'organisation :

- Protocoles : Comme décrit à la section 4.2.1, les protocoles sont un aspect très important dans la sélection des outils de performance. Comprendre quels protocoles un système utilise et lequel d'entre eux sera testé fournira les informations nécessaires afin d'évaluer l'outil de test approprié.
- Interfaces vers des composants externes : Les interfaces vers des composants logiciels ou d'autres outils peuvent devoir être envisagées dans la formulation des exigences complètes d'intégration afin de répondre aux exigences relatives au processus ou à toute autre forme d'interopérabilité (p. ex., intégration dans le processus IC).

- Plates-formes : La compatibilité avec les plates-formes (et leurs versions) au sein d'une organisation est essentielle. Cela s'applique aux plates-formes utilisées pour héberger les outils et les plates-formes avec lesquelles les outils interagissent pour la surveillance et/ou la génération de charge.

Passage à l'échelle

Un autre facteur à considérer est le nombre total de simulations utilisateur simultanées que l'outil peut gérer. Cela inclura plusieurs facteurs :

- Nombre maximum de licences requises
- Exigences de configuration de poste de travail/serveur employé pour générer la charge
- Capacité de générer une charge à partir de plusieurs points de présence (p. ex., serveurs distribués)

Facilité de compréhension

Un autre facteur à considérer est le niveau de connaissances techniques nécessaires à l'utilisation de l'outil. Ceci est souvent négligé et peut conduire des testeurs non qualifiés à configurer incorrectement des tests, qui à leur tour fourniront des résultats inexacts. Pour les tests nécessitant des scénarios complexes et un niveau élevé de capacité de programmation et de personnalisation, les équipes doivent s'assurer que le testeur possède les compétences, le contexte et la formation nécessaires.

Surveillance

La surveillance fournie par l'outil est-elle suffisante ? Existe-t-il d'autres outils de surveillance disponibles dans l'environnement qui peuvent être utilisés pour compléter la surveillance par l'outil ? La surveillance peut-elle être corrélée aux transactions définies ? Les réponses à toutes ces questions doivent être entendues pour déterminer si l'outil pourra fournir la surveillance requise par le projet.

Lorsque la surveillance est constituée par un programme/outils distinct/ pile entière, il peut être utilisé pour surveiller l'environnement de production lorsque le produit est livré.

6. Références

6.1 Standards

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

6.2 Documents ISTQB

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

6.3 Livres

[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

7 index