投稿日 2022/04/05

性能テスト計画ガイド

概要

本書は、アプリケーション開発プロジェクトの性能テスト計画で検討すべきトピック、及び性能テストを推進・実施する上で考慮すべきポイントを解説するものです。 性能テストを計画・実施する意義への理解を促進するとともに、性能テスト計画の属人化を回避することを目的としています。 本書はウォーターフォールによるWebアプリケーション開発を行うプロジェクトで活用できます。 なお、本書の一部内容は参考文献『全体テスト計画ガイド』と『テスト種別＆観点カタログ』、『非機能要求グレード』を利用することを前提としています。

また、『性能テスト計画書サンプル』にて、本書をもとに作成した性能テスト計画書のサンプルを用意しています。こちらも合わせてご参照下さい。

目次

1. 本書について

本書（性能テスト計画ガイド）は、性能テスト作業全体の計画を検討するためのガイドです。全体テスト計画では取り扱われない、性能テスト計画固有のトピックを中心に解説を行うとともに、性能テストの実施におけるポイントについても触れていきます。
本書で取り扱う「性能テスト計画」とは以下の作業を指します。

- プロジェクトにおける性能テストの方針を決定する。
- 性能テストを行うための環境の要件を決定する。

1.1. 目的

本書の目的は以下の通りです。

- 性能テストを計画・実施する意義が理解されること。
- 性能テスト計画で検討するべき事項、及び性能テストで実施する作業内容を定義するとともに、それらについて考慮すべきポイントを解説することで、性能テスト計画の属人化を回避すること。

1.2. コンセプト

本書のコンセプトは以下の通りです。

- プロジェクト固有の事情に依存せず、横断的に利用できること。
- テストの中でも性能テスト計画固有のトピックを中心に解説すること。

1.3. 対象

1.3.1. 想定読者

主要なターゲットとして、下記を想定しています。

- 性能テスト計画の策定、またはレビューを担当する人。
- 性能テストで実施する作業を担当する人。

ただし、テストケース作成やテスト実行の経験（性能テスト以外でも可）があることや、アプリケーションプログラムの実装、及びサーバ作業の経験があることを前提としています。

1.3.2. 対象とするプロジェクト

本書はウォーターフォールによる大規模Webアプリケーションの新規開発プロジェクトを前提としています。

1.3.3. 取り扱わない内容

本書では下記についてはスコープ外として取り扱わないものとします。

- 「プロジェクト計画書」に詳細が記載される傾向にあるトピックの詳細。 例えば、進捗管理、品質管理などの管理方針は「プロジェクト計画書」に記載されることが多いため、本書では詳細な説明は行わない。
- 見積りや品質の指標についてはプロジェクト毎に過去実績値などを参考に検討するものとし、本書には記載しない。
- 『全体テスト計画ガイド』に記載のある内容については冗長となるため、本書には記載しない。

1.4. 活用シーン

本書は以下のようなシーンで活用できます。

- 全体テスト計画で検討した性能テスト方針が問題ないか、実施可能かどうか評価するとき。
- 実施可能な性能テスト計画を考えるために必要な検討事項や、あらかじめ収集が必要な情報を確認するとき。
- 性能テスト計画書の構成や内容を決定するとき。
- 性能テスト計画書の作成後、セルフチェックやレビューにて、記載内容の過不足を確認するとき。
- 性能テストに関わったことのない人が内容について学習するとき。

2. 性能テストについて

2.1. 性能テストとは

性能テストとは、広義の意味では言葉の通りシステムの性能を検証するテストです。しかし、性能と言っても様々な要素があり、性能テストが何を指すのかはあいまいになりがちです。それゆえに、人によって性能テストの定義が異なるということが起こり得ます。
本書における性能テストとは、『テスト種別＆観点カタログ』にて定義している狭義の意味での性能テストを指しています。『テスト種別＆観点カタログ』では性能・拡張性に関わるいくつかのテスト種別が定義されており、それぞれを簡単に表現すると以下のようになります。

テスト種別	概要
性能テスト	性能要件で決められた業務量を処理した時に性能要件通りの処理速度が出るかどうかを検証するテスト。
ストレステスト	外部からシステムの限界負荷となるアクセスをした時の動作を検証するテスト。主に以下の種類がある。 • システムの限界負荷となるまで徐々にアクセスを増やした時の動作を検証する。 • システムの限界負荷となるように短い時間で急激にアクセスをした時の動作を検証する。 • システムの限界を超えた負荷となるようにアクセスをした時の動作を検証する。
ボリュームテスト	システムに限界負荷となる量のデータを投入した時の動作を検証するテスト。
ロングランテスト	システムに長時間の負荷をかけた時の動作を検証するテスト。

上記の通り、非機能要件として定められた性能要件（想定負荷、処理速度等）をシステムが満たしているかどうか検証するのが本書で取り扱う性能テストです。

2.2. 性能テストで実施すること

性能テストでは、大きく分けて以下の作業が発生します。

- 性能テスト計画策定
- テスト環境の準備
- 外部連携システムとの調整
- 性能テストシナリオ作成
- 性能テストケース作成
- 性能テストデータ作成・投入
- 負荷ツールの準備
- 性能テスト実施
- 性能テスト結果の評価
- 性能のボトルネック解消
- テストの再実施と再評価

上記のうち、負荷ツールの準備、性能のボトルネック解消が性能テスト固有のものとなります。後述しますが、他のテストでも実施することになるテスト環境の準備、テストデータ作成・投入、テスト結果の集計・評価でも性能テスト固有で考慮すべき観点があります。そのため、性能テスト計画では多くのトピックで性能テスト固有の観点が要求され、属人化しやすい傾向にあります。

本書ではそういった観点を記載することで、属人化を回避することを目的としています。より具体的なタスクについては『3.5. タスクと役割分担』にて一覧化しています。

2.3. 性能テストの意義

性能テストを行う意義は、本番稼働時に近い状況下でシステムの動作確認をすることです。

機能テストにより、機能要件通りにアプリケーションが動くことは保証されます。しかし、機能テストは性能に関するテストではないため、システムに性能要件通りの負荷がかかった時の動作保証までは出来ません。本番稼働時のようにシステムに高い負荷がかかっている場合には、CPUやメモリ等のリソース枯渇や、同一のDBレコードアクセスによる競合が発生したり、処理が性能要件の処理時間内に完了しなかったりと、機能テストでは起こらなかったような不具合が発生することがあります。このような不具合は実際に本番相当の環境で本番相当の負荷をかけなければ検出できません。また、このような不具合が本番稼働後に発生してしまうと、画面がまともに動かなかったり、締め日までに処理が完了しなかったりと、業務に多大な影響が出てしまうことが多くあります。そのため、性能テストを実施し、本番稼働時にもシステムが正常に動くことを検証する必要があります。

2.4. 性能テスト計画の意義

性能テストを計画的に進めていかないと以下のような問題が発生する可能性があります。これらの問題により性能テストが十分に実施できなくなってしまうと、最悪の場合システムをリリースできなくなくなったり、リリース後に性能問題が発生したりする可能性があります。

発生する問題	説明
性能テスト実施に必要な作業が漏れてしまう	性能テスト固有の作業はどれも時間がかかるものが多く、検知が遅れるとリカバリが効かなくなってしまうリスクが高い。

性能テスト実施に必要な大量データが用意できない	性能テストでは本番相当の負荷をかけるために大量のデータを環境へ投入する必要がある。データは業務特性を踏まえたものでなくてはならないため、簡単には作成することが出来ない。手作業でできる規模ではないため、どうやって大量データを用意するのか、計画を立てて進めていく必要がある。
テスト実施時に本番相当の負荷をかけられる環境やツールが用意されていない	性能テストでは本番と同等の環境に本番相当の負荷をかける必要があるため、環境の設計や負荷をかけるためのツールの準備が必要となる。どちらも時間のかかる作業であり、計画的に進める必要がある。
性能テストの実施結果を評価する方針がぶれてしまう	性能テストの実施結果はサーバリソースの使用状況や、多数のリクエストに関する応答時間や応答結果等、膨大なデータとなる。これらを生データのままで評価することは難しく、統計をとったり、可視化したりすることが多い。ツールの導入や作成等の準備が必要となるため、計画的に進める必要がある。
性能テストの実施方針に関してステークホルダー間で認識齟齬が発生してしまう	性能テストで誰がどこまで検証するのか、どうやって検証するのか、実施や評価のタイミングで認識齟齬が発生してしまうと、テストの実施ができなくなったり、テストをやり直さなければならなくなったりといった問題が発生する。性能テストの実施にあたり、その意義と実施方針をステークホルダー間で合意する必要がある。

このような問題を未然に防ぐため、性能テストの計画を立案し、計画的に作業を進めるとともに、ステークホルダーと合意形成を行っていくことが重要となります。

2.5.検討時期

性能テスト計画は、各作業を実施するよりも早く立案する必要があります。性能テストで実施すべき作業は『3.5. タスクと役割分担』にて解説しますが、性能テストをするための環境構築が必要となるため、要件定義や設計工程から作業が発生します。そのため、それに伴って性能テスト計画も要件定義時点から始めていく必要があります。具体的には、性能要件が決まった段階から計画の立案を始めるのが理想的です。

2.6.検討事項

具体的な全体テスト計画の検討事項については、以下をご参照ください。

章番号	トピック	概要
3.1.	性能テスト方針	性能テスト計画を立案する上での、基本的な考え方・方針および前提事項を検討する。
3.2.	性能テストの実施方針	性能テストの実施や準備における方針を検討する。
3.3.	テスト環境	性能テストを実施する環境の要件を検討する。
3.4.	体制	性能テストの推進体制・役割を検討する。
3.5.	タスクと役割分担	性能テストのタスクと役割分担を検討する。
3.6.	スケジュール	性能テストのスケジュールを検討する。

3. 検討が必要なトピックの解説

3.1. 性能テスト方針

このトピックでは、性能テストに関わる要件、制約を踏まえて、性能テスト全体に関わる方針を検討します。 検討においては、全体テスト計画で検討した内容に加え、性能テスト固有の制約を考慮します。 このテスト方針をベースに詳細な内容を検討します。

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。

3.1.1. 性能テストに関わる初期条件

プロジェクト計画書や非機能要件定義書、全体テスト計画書などをインプットに、性能テスト計画の立案に必要な性能要件・制約事項を初期条件として収集・整理します。
全体テスト計画にて収集した初期条件（『全体テスト計画ガイド』の『4.1.テスト方針』をご参照下さい。）に加え、性能テストでは要件定義で定められた性能要件がインプットとして必要になってきます。具体的には、参考文献『非機能要求グレード』にて記載されているような項目が漏れなく定義されている必要があります。性能要件が要件定義にて決まっていない場合、性能テスト計画が立案できないので、早急に定義して下さい。

また、性能要件が決まっていたとしても、定義が曖昧な場合は性能テスト計画の立案時のインプットとして不十分と

なってしまいます。性能要件が以下のような観点で明確に決まっていなかった場合も早急に定義し直すようにして下さい。

観点	説明
業務量	性能要件として、想定される業務量が定義されている必要がある。Webアプリケーションにおいては単位時間あたりにシステムへ飛んでくるリクエスト量を業務量として定義することが多い。
ターンアラウンドタイム (処理にかかる時間)	性能要件として、許容されるアプリケーションの処理時間が定義されている必要がある。 特に、システムにおけるどこからどこまでの範囲に対する処理時間なのかまで含めて決めておかないと、評価するための測定指標が決められなくなるので注意が必要。外部システムとの連携がある場合、外部システムの処理も含めての処理時間なのか、自システムと外部システムそれぞれでの処理時間なのかも要件定義時点で明確になっている必要がある。 また、ターンアラウンドタイムを「一律で○秒以内」のように決めてしまうと、たまたま処理が遅くなってしまったリクエストを許容できなくなるため、順守率やパーセンタイルを用いて定義することが多い。この場合、計測範囲が大きいほどネットワーク起因で順守率が下がる傾向にあるため、計測範囲に合わせた順守率となるように注意が必要。 以下にオンライン機能の性能要件に関する決め方の一例を示す。 • ユーザ端末からリクエストを送信してから3秒以内にユーザ端末のブラウザ上でレンダリングが完了する。順守率は80%とする。※ • ロードバランサにリクエストが到達してからアプリケーションの処理が完了し、ロードバランサからレスポンスを返すまでを１秒以内とする。順守率は90%とする。 ※フロントエンドがリッチに作り込まれている場合、フロントエンドの性能も考慮する必要があるため、レンダリングまで含めた性能要件を例としている。
スループット (単位時間あたりに処理できる量)	処理時間だけでなく、性能要件として、アプリケーションの単位時間あたりの処理量も合わせて定義されている必要がある。 多くの場合、バッチ処理に対する単位時間あたりの処理件数が指標となる。この場合は単位時間を明確にするとともに、バッチ処理ごとにスループットを定義する必要がある。 ターンアラウンドタイムと同様に、どこからどこまでの範囲に対するスループットなのかも含めて定義されている必要がある。
データ量	データ量によってもシステムの負荷は大きく変わるため、ユーザ数など、どのテーブルにどれだけのデータ量を格納する想定なのかについても性能要件で決まっている必要がある。 以下に性能要件の決め方の一例を示す。 • 稼働時はユーザ100万人程度を想定し、5年後にはユーザが500万人に増加する想定とする。
インフラ	もしアプリケーションが要件通りの性能を出せたとしても、そのときのサーバリソースが限界ギリギリでは、長期間の運用に耐えることができない。そのため、インフラ面でもサーバリソースの使用率について、性能要件として定める必要がある。 具体的には以下のようなサーバリソースが性能要件で決める対象となる。

	• CPU使用率 • メモリ使用率 • ディスク使用率 また、CPUやメモリについては、瞬間的なリソース使用率の高騰が正常に稼働している場合でも起こりうるため、インフラの性能要件ではリソース使用率の閾値を決めるとともに、瞬間的なリソース使用率の高騰を許容するように定めることが多い。
スケーラビリティ	将来的に見込まれる業務量やデータ量の増加、及びその増加に対するシステムの拡張性についても性能要件として定める必要がある。ターンアラウンドタイムやスループット、インフラについて将来の想定を決めて、それらに対し、例えば以下のような要件を決めていく。 • 何年後かに想定される業務量やデータ量に耐えうるリソースをあらかじめ用意する。 • 業務量やデータ量の増加に従ってリソースを増強する計画を立てる。（段階的に増強することもある。） • 業務量やデータ量に合わせて自動的にリソースを増強する。 また、業務量やデータ量の増加に伴ってリソースを増強していく場合は、増強後のシステム構成でも性能要件を満たしていなければならないため、増強後のシステム構成での性能テストも実施する必要がある。

これらの要件については、以下のようなパターンそれぞれについても一律で定義するのか、それともパターンごとに定義するのか、合わせて要件定義で決定する必要があります。

- テスト対象機能毎に定義するのかどうか
- 負荷が平常時の場合とピーク時の場合それぞれで定義するのかどうか
- 冗長構成をとる構成要素について、縮退状態の場合と冗長構成を維持している状態、それぞれで定義するのかどうか

全ての機能について性能テスト対象とするのかについても非機能要件として定義する必要があります。ここが明確になっていない場合、テスト実施範囲に関する方針の定義ができなくなります。

また、性能テストの制約事項として、どれだけのコストが割り当てられているのかについても確認する必要があります。「2.2. 性能テストで実施すること」にも記載したとおり、性能テスト固有で考慮すべき観点は多いうえに、方針によるコスト変動が大きく、他のテストよりも準備に時間がかかる傾向があります。そのため、性能テストのコストを正しく見積もるのは難易度が高く注意が必要です。割り当てられたコストが現実的なものであるかどうか確認し、そのコスト内で性能テストを実施することが困難だと判断される場合には、プロジェクト内でのエスカレーションが必要となります。

他にも、後述の『3.3. テスト環境』にて性能テストに必要な環境の要件を検討する際のインプットとなる、テスト環境に関する制約事項も併せて確認するようにして下さい。具体的には、例えば以下について確認する必要があります。

- 各環境はどの時期から使えるようになるのか
- 各環境はどのような構成となっているのか

3.1.2. 品質・コスト・スケジュール・テスト実施範囲・網羅性に関する方針の定義

品質・コスト・スケジュール・テスト実施範囲・網羅性に関する方針を検討します。基本的な考え方は『全体テスト計画ガイド』の『4.1.テスト方針』をご参照下さい。

ここでは、性能テスト固有の考慮点について解説します。システムの性能品質や、どの機能についてテストするのかについては既に性能要件として決まっているので、ここでは性能要件に対して、どこまで性能を保証するのかを検討します。

テスト品質としては100%の網羅率とするのが理想的ですが、コスト・スケジュールとの兼ね合いから、リスクの低いパターンについては目標とする網羅率を100%から下げることがあります。

具体的には以下のような網羅性についてどこまでテストするのかを検討します。

テスト対象機能に対する網羅性

性能要件にて、全機能をテスト対象としている場合は特に問題はないですが、一部機能をテスト対象としている場合、計画時点で機能単位でテスト対象を決定する必要があります。対象とする機能を選ぶ観点としては、例えば以下があげられます。

- 処理方式（登録、更新、検索、etc.）ごとに代表機能を選ぶ。
- ユーザからのアクセスが多い機能を選ぶ。
- 性能リスクが高い機能を選ぶ。
- 性能を重要視する機能を選ぶ。

また、連携する外部システムが存在する場合、外部システムの処理もテスト実施範囲に含めるかどうかについても合わせて検討する必要があります。検討する際に考慮すべき点として以下があげられます。

- 連携する外部システムの性能が保証されているかどうか
  外部システムが既に稼働中のシステムで性能が保証されている場合は性能テストの実施範囲から外す選択肢も出てきます。
- 連携する外部システムの性能テストが実施可能かどうか
  外部システムが既に稼働中のシステムで、テスト環境が存在しない、あるいはテスト環境が性能テストに必要な要件を満たさない場合、そもそも外部システムを含めた性能テストが実施できなくなってしまいます。このような場合はシミュレーターによって外部システムとの処理をダミー化することも選択肢の一つとなります。

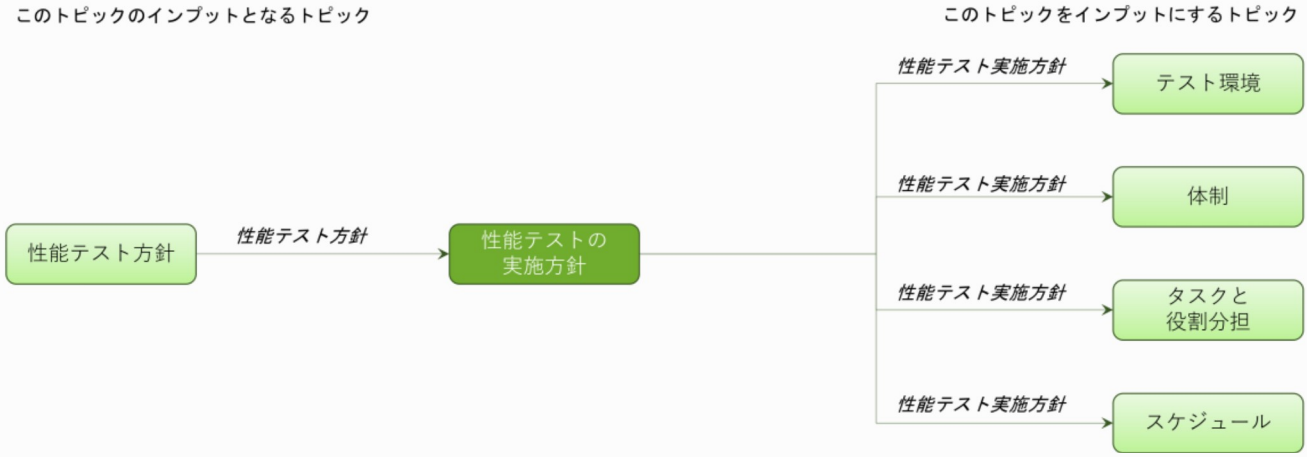
3.2. 性能テストの実施方針

このトピックでは性能テストの実施方針について検討していきます。
『3.1. 性能テスト方針』で検討した方針を踏まえ、どのように性能テストを実施していくのか検討していきます。
具体的には以下のトピックについて検討してきます。

- 性能テスト観点
- 性能テストシナリオの方針
- 性能テストデータ方針
- 負荷ツールの選定

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。



3.2.1. 性能テスト観点

性能テストでは、システムが性能要件通りの性能を出しているかどうかという観点で検証を行います。

このトピックでは、何を持って性能通りの性能が出ていると評価するのか検討します。性能要件にて測定指標やその目標値は定義されていますが、性能テストの実施時にどうやって測定指標を測定するのか、どうやって目標値に収まっていることを評価するのかについては性能テスト計画にて検討する必要があります。

以下に性能要件ごとの測定方法の例を示します。実際のプロジェクトでは以下の例を参考に、システムの構成や使用する製品に合わせて確認方法を検討して下さい。また、生データのまま目視で確認することは難しく、確認漏れの原因となるため、評価方法として、測定したデータを可視化して評価することも併せて検討して下さい。

性能要件	測定方法の例
業務量	• システムに対して単位時間あたりに送信したリクエスト数で評価する。
ターンアラウンドタイム	• クライアントから見てリクエストを送信してからレスポンスを受け取るまでをターンアラウンドタイムとして定義している場合 使用する負荷ツール※から見たターンアラウンドタイムに当たるメトリクスを収集して評価する。 • サーバ上での処理時間としてターンアラウンドタイムを定義している場合 ロードバランサにて、処理開始（リクエストを受け取った瞬間）や処理終了（レスポンスを返した瞬間）が分かるようなログを出力するように設定し、実行後のログから処理時間を算出する。算出には自動計算ツールを作成して使用する。
スループット	• バッチの実行ログから処理件数と処理時間を取得し、単位時間あたりの処理数を算出する。
データ量	• 性能テストの実施に当たって投入したデータをエビデンスとして残し、評価する。 • 性能テスト実施時にDBのテーブルに対してレコード数を確認した結果も合わせてエビデンスに残すことで評価する。
インフラ	• サーバリソースを取得するOSコマンドを１分ごとに実行し、その結果からCPU、メモリ、ディスク使用率が性能要件を満たしているかどうか評価する。

※負荷ツールについては『3.2.4. 負荷ツールの選定』をご参照下さい。

また、性能テストでは評価のための情報だけでなく、性能問題が発生した際の解析に使う情報も収集し、確認する必要があります。確認項目と確認観点、測定方法の例を以下に示します。これらが性能テスト実施時に確認出来る状態でないと、性能問題を解決する際に必要な情報が足りなくなってしまうので注意が必要です。

確認項目	確認観点	測定方法の例
CPU処理待ち行列	CPUの処理性能やコア数不足がボトルネックになっていないか	サーバリソースを取得するOSコマンドを１分ごとに実行し、ロードアベレージを取得する。
ページング発生有無	大量のページングが発生していないか	サーバリソースを取得するOSコマンドを１分ごとに実行し、ページインとページアウトの情報を取得する。
ネットワーク帯域使用量	ネットワーク帯域使用量・使用率がボトルネックになっていないか	• サーバリソースを取得するOSコマンドを１分ごとに実行し、サーバ側でのネットワーク帯域使用量・使用率を取得する。 • ネットワーク機器側で、製品に合わせた方法によりネットワーク帯域使用量・使用率を取得する。
ディスクI/O	ディスクI/Oがボトルネックになっていないか	サーバリソースを取得するOSコマンドを１分ごとに実行し、その結果からディスクI/Oの情報を取得する。
OS	エラーが発生していないか	OSのログにより、エラーの発生有無を確認する。
共有ストレージ	共有ストレージのディスクI/Oがボトルネックになっていないか	製品に合わせた方法でディスクI/Oの情報を取得する。
	ディスク容量が枯渇していないか	製品に合わせた方法でディスク容量を確認する。
ハードウェア共通	エラーが発生していないか	製品の機能によりエラーが検知されていないか確認する。

3.2.2. 性能テストシナリオの方針

このトピックでは性能テストで使用するテストシナリオの方針を検討します。性能テストでは負荷のかけ方も含めてテストシナリオを検討する必要があります。

性能テストで重要となってくるのは、本番の運用時に性能要件通りに性能が出ることです。そのため、本番でかかる負荷を再現するために、実際のユーザの動きを再現するようなシナリオにすることが基本方針となります。

オンライン機能

オンライン機能に対するテストシナリオでは、ユーザの動きを再現するために、例えば以下のような点に注意が必要です。

- ログインしてから該当機能へアクセスする。
  ログインが必要なシステムであれば、ユーザはログイン後に各種機能へアクセスします。そのため、テストシナリオでもログインから実行することで実際のユーザの動きが再現できます。
- 運用時と各機能へのアクセス数割合が同じになるようにする。
  実際の運用時には、テスト対象機能だけにアクセスがくるのではなく、トップ画面やメインメニュー、システムの主要機能へアクセスが集中します。テスト実施時にかかる負荷もこれらの機能へ集中的にアクセスするようにテストシナリオを検討して下さい。
- 単一ユーザによるシナリオではなく、複数ユーザによるシナリオにする。

  本番運用時は一人のユーザから大量にアクセスがくるわけではなく、多数のユーザから同時にアクセスが来ます。この状況を再現するために、テストシナリオでは複数のユーザを用いるように注意が必要です。

バッチ機能

バッチ機能でも実際の運用と同じ負荷となるようにバッチ実行のテストシナリオを検討する必要があります。
テスト対象となるバッチ機能の実行について、例えば以下のような点に注意が必要です。

- ジョブ管理ツールの定義に従って実行するシナリオとする。

  ジョブ定義上で先行後続の関係や、並列実行を定義している場合、その通りに実行するシナリオとする。

3.2.3. 性能テストデータ方針

このトピックでは性能テスト実施で使用するテストデータの方針について検討します。
投入するデータ量は性能要件を満たす必要があります。『3.1. 性能テスト方針』で初期条件として収集したデータ量に関する性能要件通りとなるよう、各テーブルに投入するデータ量を検討して下さい。
性能テスト実施時に本番通りの負荷をかけるためには、投入するデータの内容も本番運用時のものと同等となるようにすることが重要となってきます。例えば以下のような観点でデータの内容を検討します。

- データの内容
  本番運用時のデータでも性能が出ることを検証するために、ダミーの値を使用する際には、運用時に想定される値を入れるようにします。
  また、大量投入時にエラーとならないよう、テーブル間のリレーションシップについても注意が必要です。
- データの分布
  データの分布は検索実行時の性能に大きく影響します。実際の運用時に想定されるようなデータ分布となるようにデータの内容を検討します。例えば、以下のような点に注意が必要です。
  - ツールによって大量データを作成した結果、本番と異なるような分布になってしまう。
    例えば、ユーザ名は本番運用時には様々な値が入りますが、テストデータで全て同じダミーの値を入れてしまうと、検索時の性能が本番とは変わってしまいます。

    - 自動採番される値について、テストデータの採番方法自に考慮が漏れて、業務仕様と異なる分布となってしまう。
    - 日付データが特定の期間に偏ってしまう。
    - 業務観点で特殊な分布となっている項目を再現できていない。
      例えばサービス開始時に登録が集中し、ユーザの登録日時がサービス開始時期に集中するといったことが起こり得ます。

また、連携する外部システムを性能テスト実施範囲に含める場合は、外部システムのテスト環境にもテストデータの投入が必要となる場合があります。自システムのテスト環境に投入するテストデータと整合性を保つ必要があるので、外部システムに投入するテストデータの方針も計画時点で検討する必要があります。
このように性能テストにおけるデータ作成は考慮すべき観点が多くあり、他のテストに比べてデータ作成に時間がかかります。そのため、性能テストが割り当てられたコスト内に収まるかどうか確認する際にはデータ作成にかかる工数の見積もりが過小とならないよう注意が必要です。

3.2.4. 負荷ツールの選定

性能テストでは本番相当の負荷をかけるために、システムに対して相当数のアクセスをする必要があります。人力で実施するのは現実的ではないため、システムへの大量アクセスを自動的に行ってくれる負荷ツールを用いて性能テストをすることがほとんどです。
このトピックでは性能テストの実施時にどの負荷ツールを使用するのか検討します。検討時には以下のような点を考慮します。

観点	説明
導入のしやすさ	コストやスケジュールの面から、導入のしやすさが重要になる。以下のような観点で、プロジェクト特性と照らし合わせながら検討する。 • プロジェクト内に該当ツールの有識者がいるかどうか。 • どのぐらい学習コストがかかるのか。 • 環境構築がしやすいかどうか。
シナリオの柔軟性	負荷ツールによって、テストシナリオの方針で検討した方針通りのシナリオが作成できるかどうか確認する必要がある。具体的には以下のような点に注意が必要。 • セッションを維持できるかどうか。 ログインが必要なシステムでは必須の機能となる。 • クッキーから値を抽出できるかどうか。 セッション維持や何らかの機能でクッキーを使用する場合は必須の機能となる。 • レスポンスデータからCSRFトークン等を抽出できるかどうか。 トークンを使用する機能では必須の機能となる。 • ログインしてから該当機能へアクセスするような複雑なシナリオが組めるかどうか。 簡単なシナリオでの使用のみを想定したツールもあるため、注意が必要。
かけられる負荷	負荷ツールも無制限に負荷をかけられるわけではない。ツール自体の限界や、ツールを使用する端末のスペックによってもかけられる負荷の上限が決まってしまうため、注意が必要。 場合によっては、並列実行可能なツールの使用や、複数端末から負荷ツールを同時実行することを検討する。

結果の評価しやすさ	テスト観点で検討した評価方針の通りに実施結果が取得できない場合、テストが実施できても評価することできなくなってしまうので、検討時には注意が必要。評価に支障が出ないようにするため、以下のような観点で検討する。 • 評価に必要な情報が負荷ツールから取得できるかどうか。 • 評価ツール単独で可視化までできるのか、それとも可視化ツールとの組み合わせが必要かどうか。 可視化ツールとの組み合わせが必要な場合、可視化ツールも合わせて選定する必要がある。

注釈

準備作業時に考慮できていないと、後々性能テストの実施ができなくなるような問題について、以下に例を示します。
※性能テスト計画の範疇を超えていますが、計画時点で考慮することによって対処できるものもあるため、参考として記載します。

- 多要素認証が必要で、負荷ツールによる自動化ができない
  テスト対象機能の性能測定を手動で行うか、画面操作を自動化するようなツールの導入を検討する必要があります。
  あるいは、性能テストの実施時に多要素認証を無効とするような仕掛けをあらかじめ作り込んでおくという手もありますが、本番運用時に多要素認証を有効化し忘れるリスクがあるため注意が必要です。
- 登録機能や削除機能等で、単純な連続実行ができない
  登録機能や削除機能では、すべてのリクエストで同じ入力値を使用してしまうと、重複登録や削除対象がないことで入力チェックによりはじかれてしまい、連続実行ができなくなってしまいます。そのため、例えば以下のようにして連続実行可能なテストシナリオとなるように検討する必要があります。
  - 登録機能で、重複不可項目について連番等を使って重複不可項目が重複しないようにする。
  - 削除機能で、事前に十分な量の削除対象データを用意するか、登録してから削除を行うシナリオにする。

3.3. テスト環境

このトピックでは、『3.1. 性能テスト方針』や、『3.2. 性能テストの実施方針』にて検討した内容を踏まえ、性能テストに必要なテスト環境の要件を検討します。 基本的な考え方については『全体テスト計画ガイド』の『4.5.テスト環境』をご参照下さい。

ここでは性能テストに必要な環境に求められる要件について解説します。

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。

検討方法

まず、大前提として、性能テストでは本番同等のテスト環境を用意する必要があります。本番と同等の環境で検証しなければ、本番運用時に性能要件を満たすことを確認出来ないためです。具体的には以下の要件を満たしている必要があります。

- 本番と同等のシステム構成であること。
- 本番と同等のサーバリソースであること。
- 本番と同等のデータ量を投入できること。
- 本番と同等の通信帯域であること。

スケーラビリティとして将来的にリソースを増強する要件となっている場合には、増強後と同等な環境でもテストをする必要があります。増強前と増強後、両方のテストが出来るようにテスト環境の調整をして下さい。

また、性能テスト実施時の評価に必要な情報が収集可能であることも要件の一つです。具体的には以下のような要件を満たしている必要があります。

- サーバリソースを収集するためのパッケージやツールがインストールされていること。

負荷ツールを実行できる環境であるかどうかについても注意が必要です。負荷ツールの実施時にはクライアント端末が高負荷となる他、通信帯域を圧迫するため、以下の要件を満たしている必要があります。

- 負荷ツールの実行端末が十分な負荷をかけられるリソースを持っていること。
- 負荷ツールで発生する通信量に耐え得る通信帯域であること。

上記を満たす方法として、以下が例としてあげられます。

- 負荷ツールが実行できるだけのリソースを持った端末を用意する。
- 負荷ツールで発生する通信量に耐えうるだけの通信帯域を確保する。

ツール実施用の専用端末を用意する場合は、テスト環境の外に配置するのか、テスト環境内の同一ネットワーク上に配置するのかについても明確にするようにして下さい。

プロジェクトの特性を考慮し、上述の要件を満たすように性能テストで使用する環境を設計します。

また、連携する外部システムを性能テスト実施範囲に含める場合は、外部システムのテスト環境にも同様の要件が求められます。自システムのテスト環境だけでなく、外部システムのテスト環境についても要件を検討するようにして下さい。

3.4. 体制

このトピックでは、テスト実施に関与するステークホルダーと、各ステークホルダーの責務、指示系統、トレーニングについて検討します。 基本的な考え方については『全体テスト計画ガイド』の『4.6.体制』をご参照下さい。

ここでは性能テストの体制を考える上で重要となる観点について解説します。

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。

検討方法

性能テストの実施では、以下のメンバーが体制に含まれている必要があります。

- アプリケーションの開発チームメンバー
- システムのインフラ開発メンバー

理由は以下の通りです。

アプリケーションの開発チームメンバー

性能テストでは多くの作業において、業務要件の知識が必要となります。そのため、アプリケーションの開発チームメンバーを性能テストの体制に含める必要があります。具体的には以下の作業で業務要件の知識が求められることになります。

- テストシナリオ作成

  実際のユーザの動きを意識したシナリオとするためには業務要件の知識は不可欠です。

- テストデータ作成

  業務要件の知識が無ければ、テーブル間のリレーションシップを意識しつつ、本番データと類似した内容のデータを大量に作成することは難しいです。

- 負荷ツールの準備

  検討した方針に従って負荷ツールでテストシナリオを作成する際も、入力項目の仕様等、業務要件の知識が要求されます。

システムのインフラ構築メンバー

性能テストの検証対象はアプリケーションプログラムだけではなく、システム全体となります。そのため、性能テストでは開発しているシステムのインフラに関する知識も必要となります。それに加え、『3.3. テスト環境』にて検討した要件を満たすテスト環境の設計・構築が必要となるので、インフラ開発チームメンバーを性能テストの体制に含める必要があります。具体的には以下の作業でインフラの知識が求められることになります。

- 性能テストで使用する環境の設計・構築

  テスト環境の構築となるため、インフラ開発チームメンバーは体制上必須となります。

- 負荷ツールの準備

  負荷ツールの実施方法によっては、サーバへの負荷ツールのインストール等、環境構築作業の中に負荷ツールの準備も含まれることになります。

また、性能テストで発生する不具合の多くは、アプリケーションとインフラの双方がボトルネックとなり得ます。そのため、アプリケーションプログラムとインフラの双方の観点による解析・解消をすることになるため、不具合解析・解消時にもアプリケーション開発メンバーとインフラ開発メンバーの協力が必要となります。

3.5. タスクと役割分担

このトピックでは性能テストに必要なタスクを洗い出し、その役割分担を検討します。 基本的な考え方については『全体テスト計画ガイド』の『4.6.体制』をご参照下さい。

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。

検討方法

以下に性能テストで発生するタスクの例を記載します。タスクを洗い出す際の参考にして下さい。

分類	タスク
性能テスト環境の準備	• 性能テスト環境の設計 • 性能テスト環境の構築

外部連携システムとの調整	• 性能テスト時に接続できる環境の調整 • どこまで負荷をかけてよいかの調整
性能テストシナリオ作成	• オンライン機能のテストシナリオ作成 • バッチ機能のテストシナリオ作成
性能テストケース作成	• オンライン機能のテストケース作成 • バッチ機能のテストケース作成
性能テストデータ準備	• テストデータ作成 • テストデータ投入
負荷ツールの準備	• 負荷ツールの導入 • 負荷ツールのシナリオ作成 • 負荷ツールの疎通確認 ※1
性能テスト実施	• 負荷ツールの実行 • サーバリソースの監視・取得
性能テスト結果の評価	• 評価対象となる情報の収集 • 収集した情報の可視化
性能のボトルネック解消 ※2	• ボトルネックの特定 • アプリケーションやインフラの改修
性能テストの再実施と再評価 ※2	• テスト再実施 • テスト結果の再評価

注釈

※1：高負荷状態でのテスト実施はほとんどの場合性能テストが初となるため、初回実施時にトラブルになりやすいです。あらかじめ負荷ツールの事前疎通確認をしておくと、初回実施時のリスクを減らすことができます。

※2：性能テストで検知された不具合はアプリケーションとインフラ双方からのアプローチが必要となるため、対応に時間がかかることが多いです。また、システムの基盤部分へ修正が入ることもあるので、テストが中止となる場合もあります。そのため、ボトルネック解消と、性能テストの再実施と再評価をあらかじめ必要なタスクとして想定しておくことで、余裕を持ったスケジュールにすることができます。

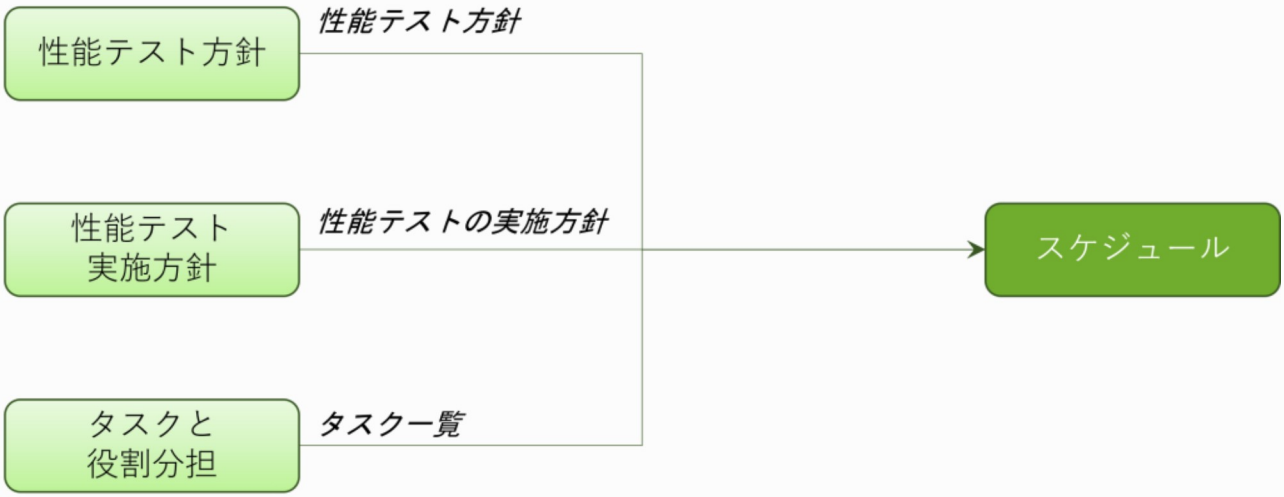
3.6. スケジュール

このトピックでは、『3.5. タスクと役割分担』にて検討したタスクの実施スケジュールを検討していきます。基本的な考え方については『全体テスト計画ガイド』の『4.8.スケジュール』をご参照下さい。

性能テスト計画で検討する他のトピックとの関係

このトピックと関係がある主要なトピックは、以下の通りです。

このトピックのインプットとなるトピック



検討方法

性能テストのスケジュールを検討する際は以下の点に注意が必要です。

- テスト環境の設計から始める必要がある。
- テストの初回実施時にトラブルが起きやすい。
- 性能テストを止めてのボトルネック解消と再実施が必要となることが多い。
- 性能テストは実施が遅くなるほどリスクが高い。
  性能テストはそもそもの実施時期がプロジェクト終盤のため、実施が遅れてしまうと性能問題を検知するのがリリース直前になってしまう可能性があります。性能問題は一朝一夕には解決しないものが多く、こうなってしまうとほとんどの場合リリースが出来なくなるので、非常にリスクが高くなります。

上記の観点から、早い時期に準備を始める必要があり、性能テストの実施期間も長めに確保する必要があるほか、性能テストの開始時期も早い方が望ましいです。

また、性能テストを他のテストと並列で実施すると、性能テスト実施時のシステムに対する負荷によって、他のテストへ影響が出てしまう可能性があります。そのため、性能テスト実施時はテスト環境を占有するスケジュールにすることが望ましいです。

上記を踏まえ、以下に『3.5. タスクと役割分担』のタスク例に対し、どのタスクをどの工程で実施するかを検討した例を示します。〇のついている工程で作業を実施します。
各工程の略称やそれぞれの実施タイミングが以下となっている理由については注釈をご参照ください。

	要件定義	外部設計	内部設計	PG・UT	IT	ST
テスト環境の設計		○	○			
テスト環境の構築				○		
外部連携システム との調整	○	○	○	○	○	○
性能テスト シナリオ作成		○	○			
性能テストケース 作成		○	○			
性能テストデータ 準備			○	○	○	○
負荷ツール準備				○	○	○
性能テスト実施						○
性能テスト結果の 評価						○
性能の ボトルネック解消						○
性能テストの 再実施と再評価						○

注釈

工程の略称は以下の通りです。

- PG：プラグラミング
- UT：単体テスト
- IT：結合テスト
- ST：システムテスト

例として挙げている各タスクの実施スケジュールを設定した意図は以下の通りです。

- テスト環境の設計、テスト環境の構築
  設計と構築作業は他の環境構築作業と並行して進めるスケジュールとしています。
- 外部連携システムとの調整
  外部システムとの連携部分も性能テスト範囲に含める場合、外部システム側でもテスト環境の準備が必要であるため、要件定義時点から調整を行っていくスケジュールとしています。
- 性能テストシナリオ作成、性能テストケース作成
  外部設計が終わった機能のテストシナリオから着手するスケジュールとしています。
  また、機能によっては内部設計の内容を踏まえてバリエーションを検討する必要があるため、

てはテストシナリオとテストケースの作成を内部設計で実施するスケジュールとしています。

- 性能テストデータ準備
  データモデルの設計が完了してからの着手となるほか、テストシナリオやテストケースの内容を踏まえての実施となるため、内部設計からの着手としています。
- 負荷ツール準備
  負荷ツールのテストシナリオ作成はアプリケーションを動かしながら実施する必要があるため、PG・UTからの着手としています。また、事前疎通の実施はIT～STで実施するスケジュールとしています。
- 性能テスト実施、性能テスト結果の評価、性能のボトルネック解消、性能テストの再実施と再評価
  どれも性能テストの工程で実施する作業となります。

4. 参考文献

- 全体テスト計画ガイド
- テスト種別＆観点カタログ
- 性能テスト計画書サンプル
- 川口真一・下村剛士(2017), “性能要件を実現する性能品質確保戦略“, ユニシス技報, 日本ユニシス, Vol.36 No.4 通巻131号, P39～54 (参照 2022-03-10).
- 仲川 樽八・森下 健 著(2017),”Amazon Web Services負荷試験入門―クラウドの性能の引き出し方がわかる”, 技術評論社.
- 独立行政法人情報処理推進機構 ソフトウェア・エンジニアリング・センター編集(2008), “SEC BOOKS：ソフトウェアテスト見積りガイドブック”.https://www.ipa.go.jp/archive/publish/secbooks20080919.html （参照 2023-04-04）.
- 独立行政法人情報処理推進機構（IPA）・ソフトウェア・エンジニアリング・センター（SEC）編集(2011), “高信頼化ソフトウェアのための開発手法ガイドブック”.https://www.ipa.go.jp/archive/publish/secbooks20110601.html （参照 2023-04-04）.
- 独立行政法人情報処理推進機構（IPA）, “システム構築の上流工程強化（非機能要求グレード）”, IPA 独立行政法人 情報処理推進機構, 2019, https://www.ipa.go.jp/sec/softwareengineering/std/ent03-b.html(参照 2022-03-10).

/* Recommend */

「ソフトウェアテスティング」 のおすすめ記事はこちら

この記事に関連する記事もお読みください。

ソフトウェアテスティング
システムの耐障害性を高めるためのアプローチ

ソフトウェアテスティング
障害テスト計画書サンプル

ソフトウェアテスティング
障害テスト計画ガイド

2022/12/22 6

2022/12/22 7

2022/12/22 13

最近投稿された記事も用意しました。

ソフトウェアテスティング

TIS AIChatLab：MagicPodをつかった自動テスト導入戦略

2025/02/26 8

ソフトウェアテスティング

DialogPlay：AIテスト自動化プラットフォーム『MagicPod』の活用

2024/10/30 22

ソフトウェアテスティング

Sales Driveの安定リリースを支える自動テスト：MagicPodの活用状況

2023/12/22 14

「ソフトウェアテスティング」で最も読まれている記事を以下にまとめています。

ソフトウェアテスティング

全体テスト計画ガイド

2018/10/01 35

ソフトウェアテスティング

テスト種別＆テスト観点カタログ

2018/10/01 19

ソフトウェアテスティング

性能テスト計画ガイド

2022/04/05 21