

システムの信頼性、スケーラビリティ、効率性を実現する
パフォーマンステストサービス



ATLAS

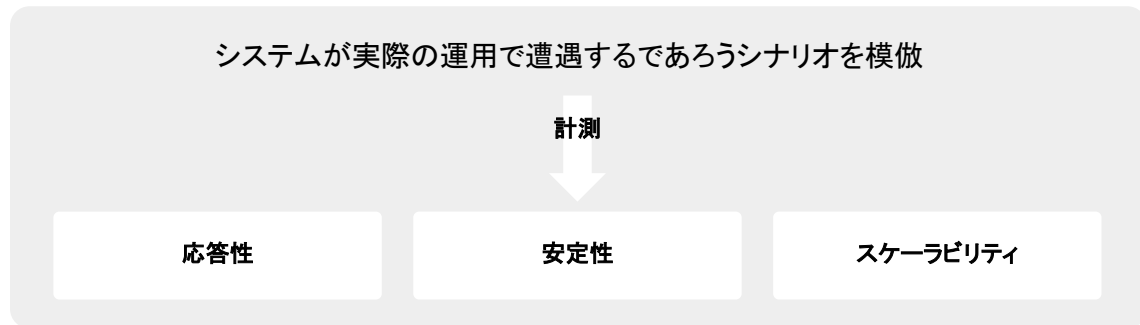
Performance Testing

はじめに

近年、ウェブサービスやアプリケーションのパフォーマンスは、ユーザーエクスペリエンスとビジネス成功に直接的な影響を与えるようになっていっています。多くの企業が高度な機能を持つアプリケーションを開発している一方で、**そのパフォーマンスがしばしば見落とされがち**です。

このような状況から、Sun*では、お客様が直面しているパフォーマンスの課題を解決し、サービスやアプリケーションの品質を向上させるための包括的なテストソリューションとして「ATLAS」を開発いたしました。

性能テストは、ソフトウェア、アプリケーション、またはシステムが **特定の負荷や条件下** **でどのように動作するかを評価**します。



**Performance
& load testing**

パフォーマンステストの重要性

パフォーマンステストのメリット

1	ユーザーエクスペリエンスの向上 パフォーマンステストは、アプリケーションやサービスが迅速に処理できるかを検証し、ユーザーエクスペリエンスを向上させます。
2	スケーラビリティ システムが増加する負荷にどれだけ対応できるかを把握し、将来におけるサービスの成長を計画することができます。
3	信頼性の向上とリスク軽減 異なる条件下でシステムがどのように動作するかを把握することで、障害を未然に防ぎ、信頼性を高めることができます。
4	コスト効率の向上 開発の早い段階でパフォーマンスのボトルネックを特定することで、長期的にコストを節約できます

パフォーマンステストを行わないことのリスク

1	貧弱なユーザーエクスペリエンス 遅いまたは応答しないアプリケーションは、ユーザーをイライラさせ、ビジネスの機会損失につながる可能性があります。
2	システム障害 パフォーマンステストを行わないと、ピーク使用時にシステムの停止が発生するリスクがあります。
3	ブランドの毀損 頻繁なダウンタイムや遅いパフォーマンスは、ユーザーの離脱を招いたり、ブランド毀損につながり、顧客の信頼を失います。
4	増加するコスト パフォーマンスのボトルネックを早期に特定しないと、将来高いコストがかかる可能性があります。

パフォーマンステストは単なるベストプラクティスではなく、**リスクを軽減し、高品質なサービスを提供するための重要な要素**です。

パフォーマンスに関するインシデントを防ぐ6つの問い

1. サイトの**応答速度が遅くありませんか？**その原因と対策は把握していますか？
2. サイトが**同時に何人のユーザーに対応**できますか？
3. **同時に処理できるリクエスト数**を把握していますか？
4. ダウンタイムやエラーが発生した場合、その**影響を最小限に抑えるための対策**をされていますか？
5. サイトのパフォーマンスが**ビジネスに与える影響**について、**具体的な数値やデータ**を把握していますか？
6. 今後のビジネス拡大を見据えて、**現在のシステムが負荷に耐えられる設計**になっていますか？

ATLASについて

パフォーマンステスト・サービス「Atlas」は、ウェブサイトまたはウェブ・アプリケーションの以下の能力を評価します。

- さまざまなレベルのユーザー・トラフィックの処理
- ユーザーのリクエストに対する応答速度
- パフォーマンスの問題の早期検出とユーザー体験の向上

Sun*のエンジニアが「Altals」を活用し、高トラフィック、低トラフィック、ピークトラフィックなど、さまざまなユーザーシナリオをシミュレートし、パフォーマンスのボトルネック、エラー、ユーザーエクスペリエンスに影響を与える可能性のあるその他の問題を特定します。

優れたパフォーマンス・テスト戦略に対応するため、スモークテスト、負荷テスト、ストレステスト、ソーステスト、スパイクテストなど、さまざまなテストをサポートし、お客様の戦略に合わせて設定を変更することが可能です。



主なパフォーマンス指標

	VU(仮想ユーザー)数 アプリケーション上のユーザートラフィックをシミュレートするための仮想ユーザ数
	HTTPリクエスト時間 全リクエストのエンド・ツー・エンド時間(総レイテンシ)
	パーセンタイル 50、90、99など、様々なパーセンタイルにわたる応答時間の分布
	リクエスト/秒(RPS) 1秒あたりのユーザーリクエスト数
	失敗率 失敗または誤ったリクエストの割合
	その他 転送されたデータ、チェック、送受信されたデータなど、多くのメトリクス

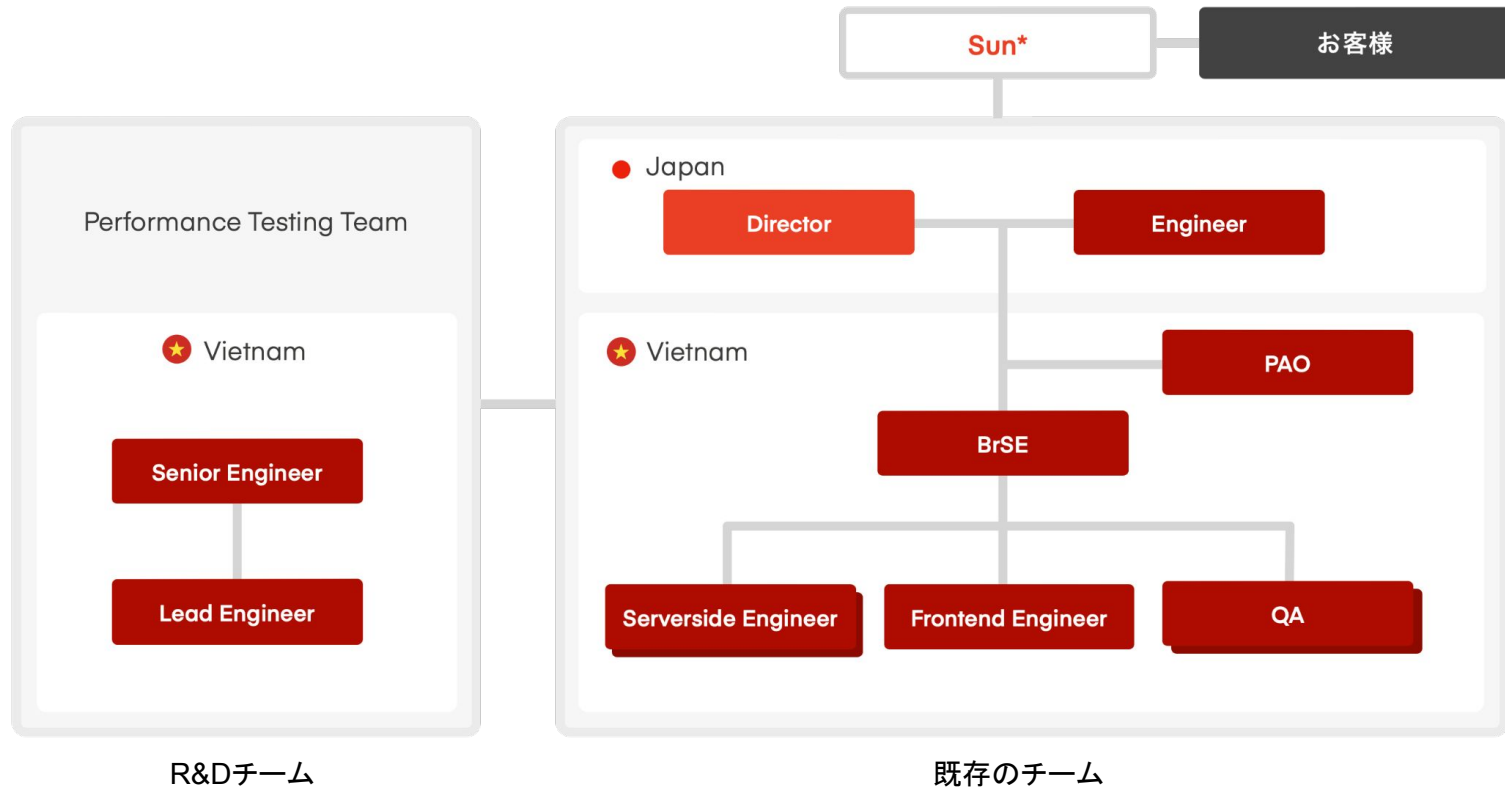
ATLASで可能なテストの種類

1	スモークテスト 新しく開発されたソフトウェアが基本的な機能を正しく実行できるかを確認する初期テストです。このテストは「煙が出ないかどうかを確認する」という意味で名付けられました。主な目的は、基本的な問題がないかを素早く確認し、詳細なテストが行える状態であるかを判断することです。	スパイクテスト 短期間に急激に増減する負荷に対するシステムの耐性を評価するテストです。突然のトラフィックの増加（例：セールやプロモーションイベント）がシステムに与える影響を調査します。これによって、システムが急な負荷の変化にどれだけ迅速に対応できるかを評価できます。
2	負荷テスト システムが特定の負荷レベルで正常に動作するかを確認するテストです。このテストは、システムが設計通りの負荷を処理できるか、またはそれを超えた場合にどのように動作するかを評価します。負荷テストは、システムのスケーラビリティと耐久性を測定するために行われます。	ソークテスト(エンデュランステスト) システムが長期間にわたって一定の負荷で安定して動作するかを確認するテストです。メモリリークやリソースの枯渇など、時間とともに発生する可能性のある問題を特定する目的で行われます。
3	ストレステスト システムが設計された容量以上の負荷を処理する能力を評価するテストです。目的は、システムが極端な条件下でどのように動作するかを理解し、その限界点を特定することです。これによって、システムが高負荷状態でどのように失敗するか、またはどのように回復するかを確認できます。	これらのテストは、それぞれ異なる側面の性能評価に重点を置いていますが、システムの性能と耐久性を確保するために非常に重要です。それぞれのテストが特定のシナリオや状況に対応しており、組み合わせて使用することで、より全面的な性能評価が可能になります。

サービスの流れ



体制イメージ



主な特徴

アプリケーション・パフォーマンスに関するレポート

- テスト結果のエグゼクティブ・サマリー、最も関連性の高いメトリクスとパフォーマンス・データは、深刻になる前に忍び寄るパフォーマンス問題を特定するのに役立ちます。
- パフォーマンス・テスト結果の詳細なレポートを作成します。
- テスト結果と分析を利害関係者、経営陣、および他のチーム・メンバーと簡単に共有できます。

◀ Scenarios & Options	REQUESTS(3) - Stress test - Login
SCENARIOS (1) +	GET Access Homepage https://atlas.sun-asterisk.vn
STRESS TEST - LOGIN 📄 🗑	POST Login Test User https://atlas.sun-asterisk.vn/login
Options	SLEEP 1 seconds
Requests (3)	GET Get user's info https://atlas.sun-asterisk.vn/api/web-init



テストの
スケジュール

Create schedule ×

Select test

Test by script (2023-05-18 10:14:08) ▾

Start at (Local time GMT+7)

05/18/2023 📅 10:18 AM ⌚

☒ Repeat

Repeat every

Interval Frequency

1 Day ▾

End

☐ Never

☒ On

Date Time

05/18/2023 📅 01:01 AM ⌚

Times

0 🔍

ATLASを利用してできること

アプリケーション・パフォーマンスの改善活動

- 結果を分析し、パフォーマンス問題に関するパターンを検出
- パフォーマンス調査のコンサルティング
- アプリケーションに対する深い洞察を得るために、データ収集、追跡、トレースを行うための最適なツールを推奨



End-to-end distributed
tracing



Distributed tracing backend

性能テスト結果のサンプル

テストでアクセス実行
しHTTPリクエスト数

400VUs | 4m | Load zone: Tokyo | Started by: dao.thai.son@sun-asterisk.com

Performance Overview

REQUESTS MADE
37288 reqs

テスト中のリクエスト
失敗数

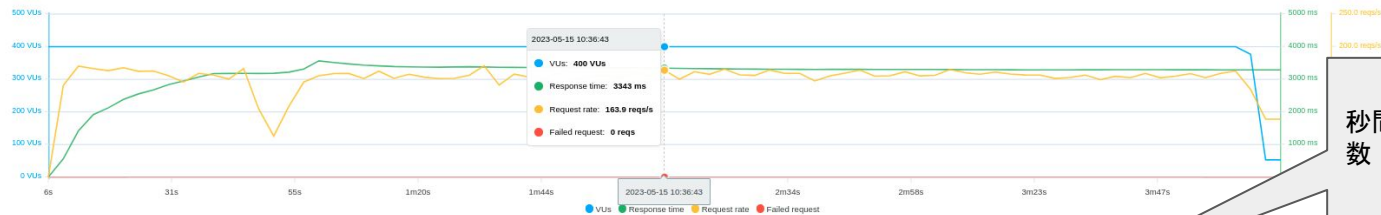
HTTP FAILURES
0 reqs

秒間最大リクエスト数

PEAK RPS
170.66 reqs/s

下位95%のリクエ
ストの応答時間

P95 RESPONSE TIME
3191 ms



秒間平均レスポンス
数

実行したテストシナリ
オ

Executor	Request Count	Failed request	P95 Response Time	AVG RPS	Performance Chart
Stress test Web Init	Constant vus	37288 reqs	0 reqs	3191 ms	150.70 reqs/s

リクエスト先毎の詳細

HTTP	Thresholds	Checks	Scenario	Method	Status	Count	Min	AVG	Max
288/37288	(0/0)	(0/0)	Stress test Web Init	GET	200	37288	77ms	1508ms	6969ms

テストビルダーでテストスクリプトを簡単に作成

TEST BUILDER

Projects > Sample Test > Sun* CI Performance Test > Test builder

Sun* CI Performance Test

Load zone: Tokyo

Save

Save & Run

<< Scenarios & Options

REQUESTS(1) - Stress test Web Init

SCENARIOS (1)

STRESS TEST WEB I...

Options

Requests (1)

GET Web Init

https://ci.sun-asterisk.com/api/web-init

SLEEP 1 seconds

Request

Name: Web Init

Method: GET URL: https://ci.sun-asterisk.com/api/web-init

Headers Params Checks Variables Body

Name: Value:

+ Add new

Thresholds (0)

OPTIONS

+ Group + Request + Sleep

スクリプト・エディターでテスト・スクリプトを簡単に作成

SCRIPT EDITOR

Projects > Sun*CI > Script Editor

Sun* CI Performance Test

Load zone: Tokyo

Save

Save & Run

✓ PERFECT Script looks fine and is ready to run

script.js

```
1 import { sleep } from 'k6'
2 import http from 'k6/http'
3
4 export const options = {
5   thresholds: {
6     http_req_duration: ['p(95)<1000'],
7   },
8   scenarios: {
9     scenario_xaeo7: {
10       executor: 'ramping-vus',
11       gracefulStop: '30s',
12       stages: [
13         { target: 20, duration: '30s' },
14         { target: 20, duration: '30s' },
15         { target: 0, duration: '30s' },
16       ],
17       startVUs: '1',
18       gracefulRampDown: '30s',
19       exec: 'scenario_xaeo7',
20     },
21   },
22 }
23
24 export function scenario_xaeo7() {
25   http.get('https://ci.sun-asterisk.com/api/web-init')
26   sleep(1)
27 }
```

テストをスケジュール化と自動実行

Create schedule ×

Select test

Sun* CI Performance Test ▼

Start at (Local time GMT+7)

08/15/2023 📅 02:45 PM 🕒

☒ Repeat

Repeat every

Interval

1

Frequency

Hour ▼

End

☐ Never

☒ On

Date

08/19/2023 📅

Time

02:45 PM 🕒

☐ After

Times

0 runs

× Close 💾 Save

メールまたはWebhookでテスト実行イベントの通知を受ける

PROJECT SETTINGS Projects > Sun*CI > Settings

General Notifications

Project Notifications

Email notifications

Events: Test started, Test finished, Test failed, Test timed out, Test aborted

Webhook notifications

Events: Test started, Test finished, Test failed, Test timed out, Test aborted

+ Create notification

Create notification

Type

webhook

Events

Test started, Scheduled test started, Test finished successfully, Test failed, Test finished, Test timed out, Test aborted (by user), Test aborted (by system), Test aborted (by threshold), Test aborted (by limit)

Webhook URL

https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXX

+ Add new header

X-API-TOKEN XXXXXXXXXXXXXXXXXXXXXXX

Close Send test event Save

まとめ

ATLASは、さまざまな種類のお客様に適した設計になっています。

パフォーマンステストに関する経験やノウハウが豊富なプロジェクトから、シンプルにパフォーマンステストを実施したいプロジェクト



どの規模のプロジェクトでもSun*で開発を支援しているプロジェクトは、**ATLASを無料ご利用**いただけます。

パフォーマンス問題の検出と修正の経験はあるが、パフォーマンステストの設計と実行に十分なリソースがない。



Sun*のエンジニアがパフォーマンステストを実行いたします。

明らかなパフォーマンス問題があるが、改善・修正方法が明確でないプロジェクト



Sun*の開発チームへパフォーマンス・チューニングの改善をご依頼いただけます。



実績

お客様の期待

Project	1秒あたりの最大リクエスト数	95%のリクエスト応答時間	失敗率	求めること
A	40リクエスト/秒	1,000ms未満	0%	システムの安定稼働
B	100リクエスト/秒	5,000ms未満	0%	システムの安定稼働
C	150,000 リクエスト/秒	3,000ms未満	0%	システムの安定稼働

実際の値

Project	1秒あたりの最大リクエスト数	95%のリクエスト応答時間	失敗率	要因
A	~16リクエスト/秒	7,412ms - 14,218ms	100%	システムクラッシュ、クエリ遅延...
B	~53リクエスト/秒	13,782ms - 22,858ms	100%	システムクラッシュ、クエリ遅延、キューの輻輳..
C	~2000 リクエスト/秒	2,700ms - 3,000ms	80%	システムクラッシュ、クエリ遅延...

実績

コンサルティングとソリューション

Project	提案した改善点	導入したもの
A	● 最大接続数の増加 ● 遅いSQLクエリを修正 ● アプリケーションのCPUを増やす ● PHP-FPMのチューニング	● 最大接続数の増加 ● 遅いSQLクエリを修正 ● アプリケーションのCPUを増やす ● PHP-FPMのチューニング
B	● PHP-FPM のチューニング ● 遅いSQLクエリの修正 ● キューの最適化	● PHP-FPM のチューニング ● 遅いSQLクエリの修正 ● キューの最適化
C	● FEに静的サイト生成を使用 ● 遅いSQLクエリを修正 ● キューの最適化 ● キャッシュの適用	● FEに静的サイト生成を使用 ● 遅いSQLクエリを修正 ● キューの最適化 ● キャッシュの適用

実績

改善後の結果(同じインフラコストを維持)

Project	改善前	改善後
A	- リクエスト/秒:16回/秒 - P95応答時間:7,412ms~14,218ms - 失敗率 ~95%	- リクエスト/秒:32回/秒 - P95応答時間 <3,000ミリ秒 - 失敗率 0%
B	- リクエスト/秒: ~53回/秒 - P95応答時間:13,782ms~22,858ms - 失敗率 ~26%	- リクエスト/秒:100回/秒 - P95応答時間 < 5,000ms - 失敗率 ~4%
C	- リクエスト/秒: ~2000回/秒 - P95応答時間:2700ms~3000ms - 失敗率 ~50%	- リクエスト/秒: ~10000回/秒 - P95応答時間 < 3,000ms - 失敗率 0%



シナリオのサンプル

スモークテストの例 - シナリオ

このシナリオでは、ベースライン性能値を収集するためにスモークテストを実施します。

スモーク・テストはスループットを小さく(2-5 VU または 3-10 反復の間)、時間を短く(30-60 秒の間)する必要があります。

次の図のシナリオ構成は、1 分間に 5 人の仮想ユーザーがターゲットサイトにアクセスすることを意味します。

Scenarios & Options

Options - Smoke_Test

SCENARIOS (1) +

SMOKE_TEST  

Options

Requests (3)

Thresholds (0)

OPTIONS ^

General

Name

Executors 

Constant vus

Start time 

Graceful stop 

Constant VUs

A fixed number of VUs execute as many iterations as possible for a specified amount of time.

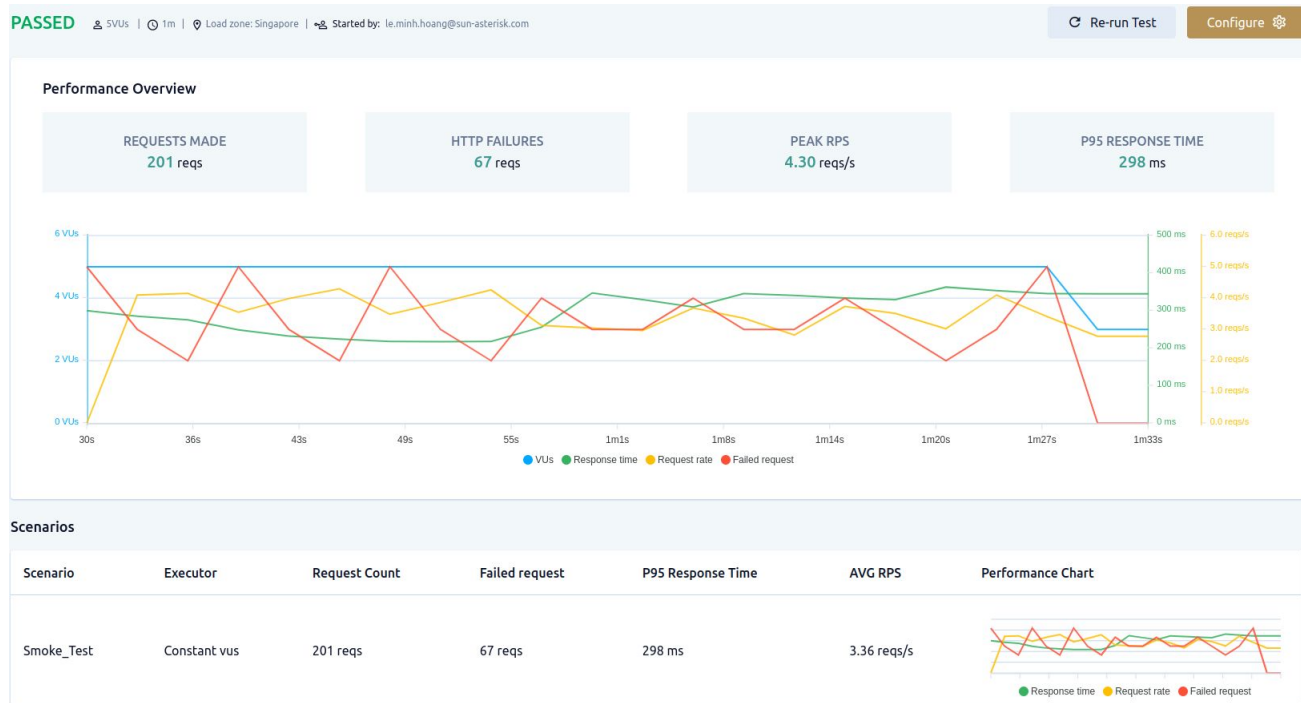
VUs  VUs

Duration 

スモークテストの例 - テスト結果

67のリクエストでHTTPエラーが発生。

レスポンスタイム(緑の線)は5人のバーチャルユーザーを維持しながら比較的安定しています。このスモークテストの結果は、さらなるテストができるようになるまで、システムのデバッグと修正に使用されます。



負荷テストの例 - シナリオ

このシナリオでは、システムが典型的な日にパフォーマンス目標を満たすかどうかを理解するために、負荷テストを実施。典型的な日は、平均的な数のユーザーが同時にアプリケーションにアクセスし、通常の作業を行う場合を意味します。システムにアクセスする VU 数を、5 分以内に 1 から 100 VU まで徐々に増やし、次の 10 分間はその 100 VU を維持し続け（これが平均的な負荷であると仮定）、最後の 5 分間は徐々にトラフィックを 0 VU まで減らします。

<< Scenarios & Options

Options - Load_Test

SCENARIOS (1) +

LOAD_TEST  

Options

Requests (1)

Thresholds (0)

OPTIONS ^

General

Name

Executors 

Ramping vus

Start time 

Graceful stop 

Ramping VUs

A variable number of VUs execute as many iterations as possible for a specified amount of time.

Start VUs  VUs

Graceful Ramp Down 

Target VUs  VUs

Duration  

VUs



VUs



+ Add new stage

複雑なシナリオ例

単純なシナリオだけでなく、複雑な処理フローシナリオにも対応しています。
下記の例のように、複数のステップを持つアプリケーションテストフローを作成します。

一般的なサイト	ベーシック認証でログイン→APIトークンを取得→APIトークンを使ってAPIにアクセス。
Eコマースサイト	登録 → ログイン → 商品を見る → 商品をカートに入れる → 注文をチェックアウトする → ログアウト
ブログサイト	ログイン → ブログ記事表示 → 新規記事作成 → 記事更新 → 記事削除



よくある質問と回答

FAQ

Q	他のチーム（QA、BE、Infra）はテストの進捗に関与する必要がありますか？
A	はい。テストのスピードと精度を高めるためのベストプラクティスとして、他のチームもテストの進行に関与する必要があります。QA チームは、テストシナリオやテストケースの作成やレビューに参加すべきです。また、BEチームは、システムのログを監視し、不健全な動作などを把握することに参加すべきです。 インフラチームは、最適なインフラソリューションを見つけるために、インフラ構成の提案や変更に参加する必要があります。そのため、パフォーマンス・テストの計画時には、他のチーム・メンバーの余分な労力を事前に考慮する必要があります。
Q	テストはどの環境で実施されますか？
A	新規開発PJの場合：テストは本番環境で、リリース時と同じ設定で実行します。保守PJの場合：正確なテスト結果を得るために、本番環境と同じスペック、同じ設定のステージング環境でテストを実施するため、スベックアップを考慮する必要があります。

Q	ATLASは他のパフォーマンステストツールと何が違うのですか？
A	Atlas: 高性能、軽量、ユーザーフレンドリー、パフォーマンステストに特化している。Javascriptによるscript-as-codeをサポート。簡単なクラウド統合、シームレスなスケーリング。プロフェッショナルなサポートチーム。オープンソースツール（JMeter、Gatling、.etc）：多様だが複雑な場合が多く、深い専門知識が必要。ローカルでのセットアップ、インフラ管理、スケーリングに課題があり、より大きなコンピューティングリソースが必要。 クラウドベースのソリューション（LoadRunner、BlazeMeterなど）：導入が容易で、インフラ管理が不要。ただし、設定が制限される場合があり、コストが高い。
Q	ブラウザテストをサポートしていますか？
A	2023年8月の時点では、まだブラウザテストをサポートしていません。しかし、近々この機能を開発する予定です。最新情報をお待ちください。

FAQ

Q	Google PageSpeed Insightとの違いはなんですか
A	Google PageSpeed Insightは負荷テストツールではなく、一つのリクエストを送って、そのレスポンスを分析します。 負荷テストのツールは同時に大量のリクエストを送って、システム全体のパフォーマンスが落ちるかどうかを確認します。 PageSpeed Insightでテストし結果が良好でも、負荷テストをすると、悪い結果になる可能性があります。 1人のユーザーがアクセスでは速くとも、1000人がアクセスするとシステムだダウンしてしまう、というケースを負荷テストで調査します。
Q	
A	



Make awesome things that matter.