

# 本書の構成と読み方

本書は全9章と付録で構成されています。

1章ではまず最初に、高速なWebサービスとは何かを概説します。「高速なWebサービス」という漠然としたイメージから、高速とは具体的にどのような状態のことを指すのかを高い解像度で理解します(著者：馬場俊彰)。

2章ではWebサービスのモニタリングについて学びます。高速であることを確認するためには、実際のパフォーマンスを観測可能にする必要があります(著者：中西建登。9章も担当)。

3章ではISUCONで実際に出题されるものと同様のWebサービスを例にして、1ステップずつ手を動かして高速化の第一歩を踏み出す手順を説明します(著者：藤原俊一郎。4章、付録Aも担当)。

4章ではWebサービスに負荷を与えるためのツールの使い方を学びます。Webサービスに対して、実際のアクセスパターンに近いシナリオを持った負荷を与えることで、実用的な負荷試験が可能になります。

5章ではWebサービスには必須の存在であるデータベースのパフォーマンスについて、主にMySQLを例にして解説します。データベースの速度は、Webサービスの性能に密接に関わります(著者：長野雅広)。

6章ではリバースプロキシについて、nginxを例にして学びます。ある程度以上の規模のWebサービスを高速に配信するためには、リバースプロキシを適切に扱うことが重要です(著者：金子達哉。7、8章も担当)。

7章ではWebサービスにおけるキャッシュ戦略について説明します。キャッシュを適切に利用すると大きな高速化の効果がありますが、不適切な使い方をすると深刻な問題が発生することもあります。適切なキャッシュの使い方を学びましょう。

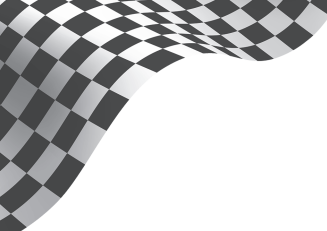
8章ではWebアプリケーションの実装について、特にISUCONや実務において問題が発生しがちな、パフォーマンス上重要なポイントを解説します。

9章ではWebサービスを動作させる基盤となる、OSのチューニングについて学びます。普段Webアプリケーションを運用しているだけでは意識しづらいOSのレイヤーにも、パフォーマンスに重要なポイントは多くあります。

付録Aでは、本書内で例として取り上げたWebサービスに対して、本書内で紹介したテクニックを適用してパフォーマンスチューニングを行い、性能がどのように改善したのかを順を追って解説します。

付録Bでは、実際にISUCONで使用されるようなベンチマーカ(負荷を与えるプログラム)を作成するためには、どのような点を考慮する必要があるのかを解説します(著者：草野翔)。

Webサービスの高速化はひとつの技術で成り立つものではなく、複数の技術を組み合わせて行う総



はじめに ..... iii

本書の構成と読み方 ..... v

著者プロフィール ..... vii

**Chapter 1 チューニングの基礎知識 1**

**1-1 “高速であること”は現代のWebサービスの必須要件 ..... 2**

    “高速”はWebサービスの競争力を直接的に左右する ..... 2

    “高速”はSEOに効果あり ..... 2

    “高速”は高コスト効率を実現する ..... 3

**1-2 高速なWebサービスとは ..... 4**

    どうなっていると高速なWebサービスなのか ..... 4

    Webサービスの速さの単位 ..... 5

    Webサービスの構造を把握する ..... 6

**1-3 Webサービスの負荷 ..... 8**

    Webサービスの負荷が高い状態 ..... 8

    速さとキャパシティ ..... 9

    パフォーマンスチューニング ..... 10

**1-4 必要十分なキャパシティを用意するには ..... 10**

    必要十分なキャパシティとは ..... 11

    必要十分なキャパシティの見積りかた ..... 12

**1-5 パフォーマンスチューニング“きほんのき” ..... 13**

    推測せず計測する ..... 13

    公平に比較する ..... 13

    1つずつ比較する ..... 14

<b>1-6</b>	パフォーマンスチューニング“きほんのほ”	15
	ボトルネックだけにアプローチする	15
	ボトルネックの特定は外側から順番に	16
	ボトルネック対処の基本3パターン	17
<b>1-7</b>	パフォーマンスチューニング“きほんのん”	18
	負荷試験の各工程の概要	18
<b>1-8</b>	まとめ	22

## Chapter 2 モニタリング

23

---

<b>2-1</b>	モニタリングとは - インフラにおけるテスト	24
<b>2-2</b>	モニタリングに対する考え方	26
<b>2-3</b>	モニタリングの種類	27
	外形監視	27
	内部監視	28
<b>2-4</b>	手動でのモニタリング	29
<b>2-5</b>	モニタリングツール	31
<b>2-6</b>	モニタリングツールのアーキテクチャ	32
	エージェント node_exporter	35
	node_exporter で取得できるメトリクス	37
<b>2-7</b>	実際にモニタリングを行う	38
<b>2-8</b>	モニタリングの注意点	40
	正しい計測結果の見極め	41
	2つのグラフを比較するときは他の条件を合わせる	43
	高負荷状態のモニタリング	44
	モニタリングの解像度	45
<b>2-9</b>	ログに対するモニタリング	47
<b>2-10</b>	まとめ	48

---

<b>3-1</b>	本書で扱う Web サービス private-isu .....	51
	private-isu の仕様と動作環境 .....	51
	手元で private-isu を動作させる .....	51
	Amazon EC2 で private-isu を起動する .....	52
	Docker で private-isu を起動する .....	53
	実際に private-isu を触ってみる .....	55
<b>3-2</b>	負荷試験の準備 .....	57
	負荷試験環境を用意する .....	58
	nginx のアクセスログを集計する .....	58
	アクセスログを JSON 形式で出力する .....	59
	JSON 形式のアクセスログを集計する .....	62
	alp のインストール方法 .....	63
	alp を使ったログ解析方法 .....	63
<b>3-3</b>	ベンチマーカによる負荷試験の実行 .....	65
	ab コマンドのインストール .....	66
	ab コマンドの使用法 .....	66
	ab の結果と alp の結果を比較する .....	67
	アクセスログのローテーション .....	68
<b>3-4</b>	パフォーマンスチューニング 最初の一步 .....	70
	負荷試験実行 - 最初の結果を把握する .....	70
	負荷試験中の負荷を観察する .....	74
	MySQL のボトルネックを発見する準備 .....	75
	スロークエリログを解析する .....	77
	チューニングの成果を確認する負荷試験 .....	82
	あらたなボトルネックを見つける .....	83
<b>3-5</b>	ベンチマーカ の並列度 .....	85
	サーバーの処理能力を全て使えているか確認する .....	86
	なぜ CPU を使い切れていないのか .....	87
	複数の CPU を有効に利用するための設定 .....	88
	サーバーの並列度を上げて負荷試験を実行する .....	90
<b>3-6</b>	まとめ .....	92

<b>4-1</b>	負荷試験ツールk6	94
	k6をインストールする	95
<b>4-2</b>	k6による単純な負荷試験	96
<b>4-3</b>	k6でシナリオを記述する	99
	シナリオ内で共通で使用する関数を定義する	99
	Webサービスの初期化処理シナリオを記述する	100
	sleep()関数：一定時間待機する	101
	ユーザーがログインしてコメントを投稿するシナリオを記述する	102
	check()関数：レスポンスの内容をチェックする	104
	parseHTML()関数：HTML内の要素を取得する	105
	ファイルアップロードを含むフォームを送信する	106
	シナリオで使用する外部データを用意する	107
<b>4-4</b>	複数のシナリオを組み合わせた統合シナリオを実行する	109
	統合シナリオの実行結果の例	110
<b>4-5</b>	負荷試験で得られたアクセスログを解析する	111
<b>4-6</b>	まとめ	113

<b>5-1</b>	データベースの種類と選択	116
	一貫性を重視するRDBMS	116
	アプリケーションニーズに合わせたNoSQL	118
	一貫性と分散を両立するNewSQL	119
	データベースの選択	120
<b>5-2</b>	データベースの負荷を測る	120
	OSから負荷を観察する	120
	MySQLのプロセスリストを見ている	121
	pt-query-digestによるスロークエリログの分析	123
	query-digesterを利用したプロファイリングの自動化	126
	pt-query-digestの結果の見方	126

<b>5-3</b>	<b>インデックスでデータベースを速くする</b>	130
	データベースから結果を高速に得るには	130
	データベースにおけるインデックスの役割	131
	インデックスで検索が高速になる理由	131
	MySQLにおけるインデックスの利用	132
	複合インデックス・並び替えにも使われるインデックス	135
	クラスターインデックスの構成と	
	クラスターインデックスでのインデックスチューニング	136
	多すぎるインデックスの作成によるアンチパターン	140
	MySQLがサポートするその他のインデックス	141
<b>5-4</b>	<b>N+1 とは</b>	143
	クエリ数増大によりアプリケーションが遅くなる理由	143
	N+1の見つけ方と解決方法	145
	データベース以外にもあるN+1問題	154
<b>5-5</b>	<b>データベースとリソースを効率的に利用する</b>	155
	FORCE INDEXとSTRAIGHT_JOIN	155
	必要なカラムだけ取得しての効率化	159
	プリペアドステートメントとGo言語における接続設定	160
	データベースとの接続の永続化と最大接続数	161
<b>5-6</b>	<b>まとめ</b>	164

## Chapter 6 リバースプロキシの利用

165

<b>6-1</b>	<b>アプリケーションとプロセス・スレッド</b>	167
<b>6-2</b>	<b>リバースプロキシを利用するメリット</b>	169
<b>6-3</b>	<b>nginxとは</b>	170
<b>6-4</b>	<b>nginxのアーキテクチャ</b>	173
<b>6-5</b>	<b>nginxによる転送時のデータ圧縮</b>	174
<b>6-6</b>	<b>nginxによるリクエスト・レスポンスのバッファリング</b>	178
<b>6-7</b>	<b>nginxとアップストリームサーバーの接続管理</b>	179
<b>6-8</b>	<b>nginxのTLS通信を高速にする</b>	180
<b>6-9</b>	<b>まとめ</b>	181

---

<b>7-1</b>	キャッシュデータ保存に利用されるミドルウェア	184
<b>7-2</b>	キャッシュをKVSに保存する際の注意点	187
<b>7-3</b>	いつキャッシュを利用するか	188
	十分短いTTLを設定する	189
<b>7-4</b>	具体的なキャッシュ実装方法	190
	キャッシュにデータがなければキャッシュを生成して生成結果を保存する手法	190
	キャッシュがなければデフォルト値や古いキャッシュを返し、 非同期にキャッシュ更新処理を実行する	194
	バッチ処理などで定期的にキャッシュを更新する	198
	private-isuで実際にキャッシュを利用する	199
<b>7-5</b>	キャッシュを監視する	201
<b>7-6</b>	まとめ	202

---

<b>8-1</b>	外部コマンド実行ではなく、ライブラリを利用する	204
<b>8-2</b>	開発用の設定で冗長なログを出力しない	208
<b>8-3</b>	HTTPクライアントの使い方	209
	同一ホストへのコネクションを使い回す	209
	適切なタイムアウトを設定する	211
	同一ホストに大量のリクエストを送る場合、 対象ホストへのコネクション数の制限を確認する	212
<b>8-4</b>	静的ファイル配信をリバースプロキシから直接配信する	213
<b>8-5</b>	HTTPヘッダーを活用してクライアント側にキャッシュさせる	215
<b>8-6</b>	CDN上にHTTPレスポンスをキャッシュする	218
	CDNで世界中どこからアクセスしても高速なサービスを提供する	218
	Cache-Controlを活用してCDNやProxy上にキャッシュさせる	220
<b>8-7</b>	まとめ	222

<b>9-1</b>	流れを見極める	224
<b>9-2</b>	Linux Kernelの基礎知識	224
<b>9-3</b>	Linuxのプロセス管理	228
<b>9-4</b>	Linuxのネットワーク	230
	ネットワークのメトリクス	230
	Linux Kernelにおけるパケット処理の効率化	231
<b>9-5</b>	LinuxのディスクI/O	234
	ストレージの種類	234
	ストレージの性能とは - スループット、レイテンシ、IOPS	235
	ストレージの性質を調査	237
	ディスクマウントのオプション	239
	I/Oスケジューラ	241
<b>9-6</b>	CPU利用率	243
	us - User：ユーザ空間におけるCPU利用率	244
	sy - System：カーネル空間におけるCPU利用率	245
	ni - Nice：nice値（優先度）が変更されたプロセスのCPU利用率	245
	id - Idle：利用されていないCPU	247
	wa - Wait：I/O処理を待っているプロセスのCPU利用率	247
	hi - Hardware Interrupt：ハードウェア割り込みプロセスの利用率	247
	si - Soft Interrupt：ソフト割り込みプロセスの利用率	247
	st - Steal：ハイパーバイザによって利用されているCPU利用率	247
<b>9-7</b>	Linuxにおける効率的なシステム設定	248
	ulimit	249
<b>9-8</b>	Linuxカーネルパラメータ	253
	net.core.somaxconn	253
	net.ipv4.ip_local_port_range	255
<b>9-9</b>	MTU (Maximum Transmission Unit)	260
	その他のカーネルパラメータ	263
<b>9-10</b>	まとめ	263



<b>A private-isuの攻略実践</b> .....	266
<b>A-1</b> 用意した競技用環境 .....	266
<b>A-2</b> ベンチマーカの実行方法 .....	267
<b>A-3</b> 各章の技法を適用する .....	268
初期状態 (約650点) .....	268
comments テーブルにインデックスを追加する (約7,000点) .....	269
unicorn worker プロセスを4にする (約15,000点) .....	270
静的ファイルをnginxで配信する (17,000点) .....	272
アップロード画像を静的ファイル化する (約22,000点) .....	273
GET / を解析する .....	276
posts と users を JOIN して必要な行数だけ取得する (約90,000点) .....	277
ベンチマーカが使用するファイルディスクリプタ上限を増加させる .....	281
プリペアドステートメントを改善する (約110,000点) .....	282
comments テーブルヘインデックスを追加する (約115,000点) .....	283
posts からのN+1クエリ結果をキャッシュする (約180,000点) .....	284
適切なインデックスが使えないクエリを解決する (約200,000点) .....	287
外部コマンド呼び出しをやめる (約240,000点) .....	290
MySQLの設定を調整する (約255,000点) .....	291
memcachedへのN+1を解消する (約300,000点) .....	293
RubyのYJITを有効にする (約320,000点) .....	294
最初に作成したインデックスを削除してみる (約10,000点) .....	295
<b>A-4</b> まとめ .....	296
<b>B ベンチマーカの実装</b> .....	297
<b>B-1</b> ISUCONのベンチマーカは何をするのか .....	297
負荷試験ツールとしてのベンチマーカ .....	298
Webサービス実装のE2Eテストとしてのベンチマーカ .....	298
スコアとエラーを提供する情報源としてのベンチマーカ .....	300
ベンチマーカに求められる振る舞いに気をつける .....	301
<b>B-2</b> ベンチマーカに頻出する実装パターン .....	301
context.Contextを知る .....	301
timeとcontextによるループのパターン .....	306
syncパッケージの利用 .....	310
sync/atomicパッケージの利用 .....	315
Functional Optionパターン .....	316



<b>B-3</b>	private-isu を対象としたベンチマーカの実装	316
	入出力を設計する	317
	データを持つ	321
	初期化処理を実装する	325
	ログインする処理を作る	331
	画像投稿する処理を作る	334
	トップページを検証する	335
	得点を計算する	336
	実際に動かしてみる	337
<b>B-4</b>	まとめ	338
	索引	339