

非機能要件定義

TIS株式会社

テクノロジー & イノベーション本部

テクノロジー & エンジニアリングセンター



- ApacheはApache Software Foundation の米国およびその他の国における登録商標もしくは商標です。
 - BIG-IPはF5 Networks, Inc.の登録商標です。
 - OracleおよびJava、WebLogic、JavaScript、MySQLは、オラクルおよびその関連会社の登録商標です。
 - Linuxは米国およびその他の国におけるLinus Torvaldsの登録商標です。
 - Red Hatは米国およびその他の国における Red Hat, Inc. またはその子会社の登録商標です。
 - DebianはSoftware In The Public Interest Incorporatedの登録商標です。
- その他の社名、製品名などは、一般にそれぞれの会社の商標または登録商標です。
なお、本文中では、TMマーク、Rマークは明記しておりません。

本コンテンツには、読者としてベンダー企業（SIer）の方を想定した記述があります。このため、ユーザ企業の方が読まれる場合は、適宜以下のような読み替えを行ってください。

- ベンダー：ユーザ企業内でシステム開発を担う部門
- お客様、顧客：対象システムの利用者

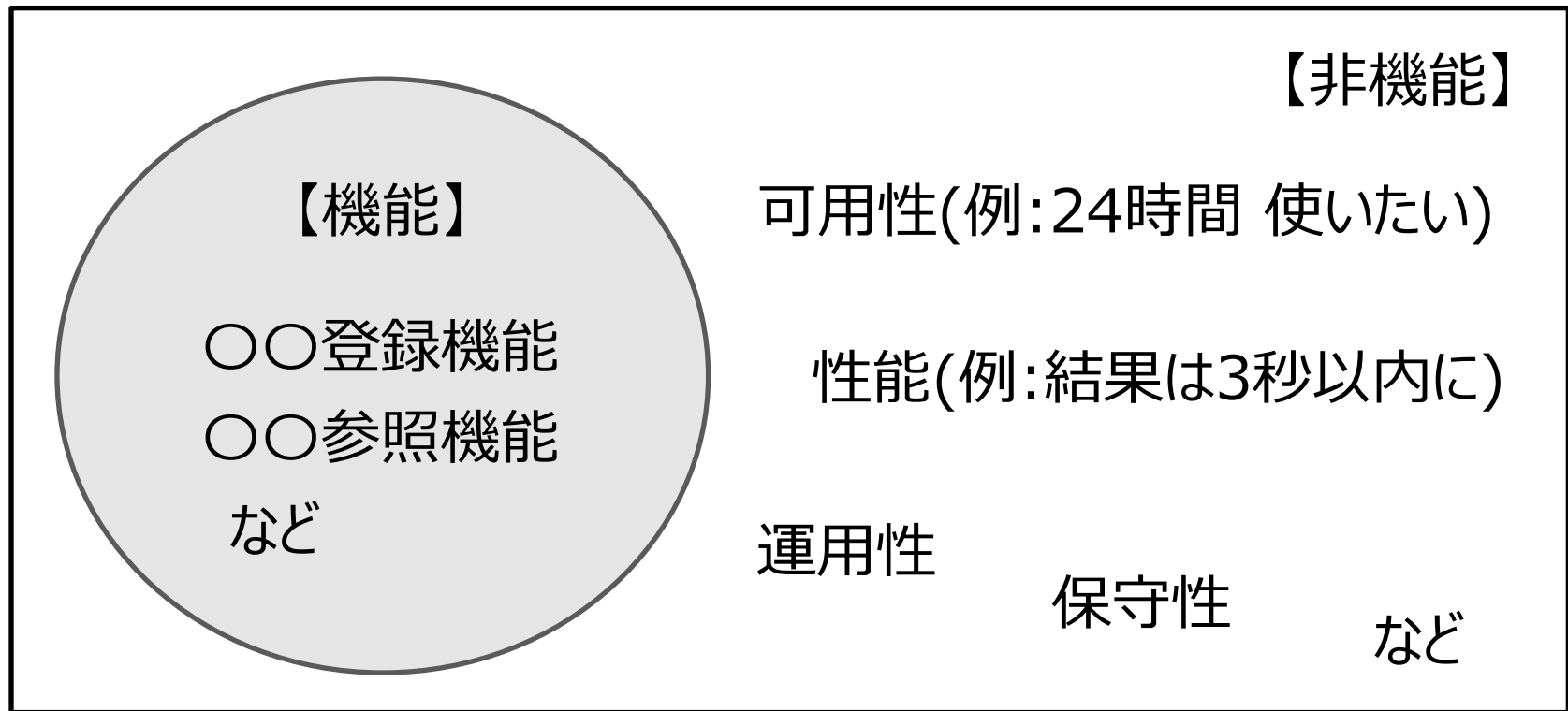
- 非機能要件について学ぶ意味
- 非機能要件を網羅的に検討する方法
～非機能要求グレード～

- 非機能要件についての重要なトピック
 - インフラについての基礎知識
 - 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
 - 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

非機能要件について学ぶ意味

機能以外の側面に求められる要件のこと

システムの側面



項目数がとても多い

可用性

運用スケジュール
業務継続性
目標復旧水準（業務停止時）
目標復旧水準（大規模災害時）
稼働率
サーバ
端末
ネットワーク機器
ネットワーク
ストレージ
データ
システム
外部保管データ
付帯設備
復旧作業
可用性確認

性能・拡張性

通常時の業務量
業務量増大度
保管期間
オンラインレスポンス
バッチレスポンス
オンラインスループット
バッチスループット
帳票印刷能力
CPU拡張性

メモリ拡張性
ディスク拡張性
ネットワーク
サーバ処理能力増強
帯域保証機能の有無
HWリソース専有の有無
性能テスト
スパイク負荷対応

運用・保守性

運用時間
バックアップ
運用監視
時刻同期
計画停止
運用負荷削減
パッチ適用ポリシー
活性保守
定期保守頻度
予防保守レベル
復旧作業
障害復旧自動化の範囲
システム異常検知時の対応
交換用部材の確保
開発用環境の設置
試験用環境の設置
マニュアル準備レベル
リモートオペレーション

外部システム接続
保守契約（ハードウェア）
保守契約（ソフトウェア）
ライフサイクル期間
メンテナンス作業役割分担
一次対応役割分担
サポート要員
導入サポート
オペレーション訓練
定期報告会
内部統制対応
サービスデスク
インシデント管理
問題管理
構成管理
変更管理
リリース管理

移行性

移行のスケジュール
システム展開方式
移行設備
移行データ量
移行媒体
変換対象（DBなど）
移行作業分担
リハーサル
トラブル対処

セキュリティ

情報セキュリティのコンプライアンス
セキュリティリスク分析
セキュリティ診断
セキュリティリスクの見直し
セキュリティリスク対策の見直し
セキュリティパッチ適用
認証機能
利用制限
管理方法
データ暗号化
不正監視
データ検証
ネットワーク制御
不正検知
サービス停止攻撃の回避
マルウェア対策
Web実装対策
セキュリティインシデント対応/復旧

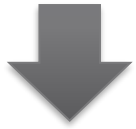
システム環境・エコロジー

構築時の制約条件
運用時の制約条件
ユーザ数
クライアント数
拠点数
地域的広がり
特定製品指定

システム利用範囲
複数言語対応
製品安全規格
環境保護
電磁干渉
耐震/免震
スペース
重量
電気設備適合性
温度（帯域）
湿度（帯域）
空調性能
環境負荷を抑える工夫
エネルギー消費効率
CO2排出量
低騒音

など(まだある)

項目数がとっても多い



{ 検討漏れが起きやすい

{ 学ぶべきことが多岐にわたる

学びづらい

現場で役立つ知識を効率的に学ぶのは難しい

(適した書籍や研修はあまり無い)

あまり意識されない、知られていない。

機能要件に目が行きがち
お客様も、ベンダー自身も...

しかし、
非機能について理解をしておかないと・・・

性能テストやリリース後になってから、性能が出ないことが発覚。

リリース後にシステム停止が頻発。

その他、検討もれによる見積過少 → 要員負荷増大／利益減

関係者と対等に話をできない。
（用語レベルでわからない）

調査・学習に多大な時間をとられる。
（1から調べる必要がある）

結果、精神的に疲弊してしまう。

学び初めのハードルを突破しましょう。

そこさえ突破すれば、**他の内容を学ぶ「とっかかり」**ができ、先に進むことができます。

そして、先に説明したような事態を回避し
現場で活躍していきましょう！

非機能要件を網羅的に検討する方法

～ 非機能要求グレード ～

非機能要件の項目は多い

可用性

運用スケジュール
業務継続性
目標復旧水準（業務停止時）
目標復旧水準（大規模災害時）
稼働率
サーバ
端末
ネットワーク機器
ネットワーク
ストレージ
データ
システム
外部保管データ
付帯設備
復旧作業
可用性確認

性能・拡張性

通常時の業務量
業務量増大度
保管期間
オンラインレスポンス
バッチレスポンス
オンラインスループット
バッチスループット
帳票印刷能力
CPU拡張性

メモリ拡張性
ディスク拡張性
ネットワーク
サーバ処理能力増強
帯域保証機能の有無
HWリソース専有の有無
性能テスト
スパイク負荷対応

運用・保守性

運用時間
バックアップ
運用監視
時刻同期
計画停止
運用負荷削減
パッチ適用ポリシー
活性保守
定期保守頻度
予防保守レベル
復旧作業
障害復旧自動化の範囲
システム異常検知時の対応
交換用部材の確保
開発用環境の設置
試験用環境の設置
マニュアル準備レベル
リモートオペレーション

外部システム接続
保守契約（ハードウェア）
保守契約（ソフトウェア）
ライフサイクル期間
メンテナンス作業役割分担
一次対応役割分担
サポート要員
導入サポート
オペレーション訓練
定期報告会
内部統制対応
サービスデスク
インシデント管理
問題管理
構成管理
変更管理
リリース管理

移行性

移行のスケジュール
システム展開方式
移行設備
移行データ量
移行媒体
変換対象（DBなど）
移行作業分担
リハーサル
トラブル対処

セキュリティ

情報セキュリティのコンプライアンス
セキュリティリスク分析
セキュリティ診断
セキュリティリスクの見直し
セキュリティリスク対策の見直し
セキュリティパッチ適用
認証機能
利用制限
管理方法
データ暗号化
不正監視
データ検証
ネットワーク制御
不正検知
サービス停止攻撃の回避
マルウェア対策
Web実装対策
セキュリティインシデント対応/復旧

システム環境・エコロジー

構築時の制約条件
運用時の制約条件
ユーザ数
クライアント数
拠点数
地域的広がり
特定製品指定

システム利用範囲
複数言語対応
製品安全規格
環境保護
電磁干渉
耐震/免震
スペース
重量
電気設備適合性
温度（帯域）
湿度（帯域）
空調性能
環境負荷を抑える工夫
エネルギー消費効率
CO2排出量
低騒音

など(まだある)

もれなく検討するための業界標準

IPAが公開している**非機能要求グレード**が
非機能要件をもれなく検討するための業界標準として使われています。

項番	大項目	中項目	小項目	小項目説明	重複項目	重要項目	マトリクス(指標)	レベル						運用コストへの影響	備考
								0	1	2	3	4	5		
A1.1.1	可用性	継続性	運用スケジュール	システムの稼働時間や停止運用に関する情報。			運用時間(通常)	規定無し	定時内(9時~17時)	夜間のみ停止(9時~21時)	1時間程度の停止有り(9時~翌朝8時)	若干の停止有り(9時~翌朝8時55分)	24時間無停止		<p>【重複項目】 01.1.1. 運用時間は、システムの可用性の実現レベルを表す項目であると共に、運用・保守性に関する開発コストや運用コストを検討する上でも必要となる項目であるため、可用性と運用・保守性の両方に含まれている。</p> <p>【オキウス】 運用時間は、オンライン/バッチを含みシステムが稼働している時間帯を指す。</p> <p>【レベル】 0内の時間は各レベルの一例を示したもので、レベル選定の条件とはしてない。規定無しは、固定のサービス時間が存在しないことを示し、基本的にシステムは停止して、必要に応じてユーザがシステムを起動するようなケースを想定している(例:障害発生に備えた予備システム、開発・検証用システム等)。定時内や夜間のみ停止は、一般的な業務形態を想定したもので、業務が稼働する時間帯が異なるシステムにおいては、時間帯をスライシさせるなどの調整が必要である。停止有りとは、システムを停止しなければならない時間帯ではなく、システムを停止できる可能性のある時間帯を指す。24時間無停止は、オンライン業務が稼働していない時間帯にバッチを稼働させる必要がありシステムを停止することができないようなケースも含まれる。</p>
A1.1.2							運用時間(特定日)	規定無し	定時内(9時~17時)	夜間のみ停止(9時~21時)	1時間程度の停止有り(9時~翌朝8時)	若干の停止有り(9時~翌朝8時55分)	24時間無停止		<p>【重複項目】 01.1.2. 運用時間は、システムの可用性の実現レベルを表す項目であると共に、運用・保守性に関する開発コストや運用コストを検討する上でも必要となる項目であるため、可用性と運用・保守性の両方に含まれている。</p> <p>【オキウス】 特定日とは、休日/祝祭日や月末月初など通常の運用スケジュールとは異なるスケジュールを定義している日のことを指す。特定日が複数存在する場合は、それぞれに対してレベル値を調整する必要がある(例:「月~金はレベル2だが、土日はレベル0」、「通常はレベル5だが、毎月1日にリポートをするためその日はレベル3」など)。 また、ユーザの休日だけでなく、ベンダの休日についても特定日として認識し、運用保守体制等を整合すること。</p>
A1.1.3							計画停止の有無	計画停止有り(運用スケジュールの変更可)	計画停止有り(運用スケジュールの変更不可)	計画停止無し					<p>【重複項目】 02.1.1. 計画停止の有無は、システムの可用性の実現レベルを表す項目であると共に、運用・保守性に関する開発コストや運用コストを検討する上でも必要となる項目であるため、可用性と運用・保守性の両方に含まれている。</p> <p>【運用コストへの影響】 計画停止が「有り」の場合、事前のバックアップや、システム構成に応じた手順準備など、運用時のコストがかかる。</p>
A1.2.1			業務継続性	可用性を保证するにあたり要求される業務の範囲とその条件。			対象業務範囲	内部向けバッチ系業務	内部向けオンライン系業務	内部向け全業務	外部向けバッチ系業務	外部向けオンライン系業務	全ての業務		<p>【オキウス】 ここでの対象業務範囲とは、稼働率を算出する際の対象範囲を指す。</p>

非機能要求グレード(2018)より

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

6つの大項目から構成されます。

- ・可用性
- ・性能・拡張性
- ・運用・保守性
- ・移行性
- ・セキュリティ
- ・システム環境・エコロジー

非機能要求 大項目	説明	要求の例
可用性	システムサービスを継続的に利用可能とするための要求	<ul style="list-style-type: none">・運用スケジュール（稼働時間、停止予定など）・障害、災害時における稼働目標
性能・拡張性	システムの性能、および将来のシステム拡張に関する要求	<ul style="list-style-type: none">・業務量および今後の増加見積り・システム化対象業務の特性（ピーク時、通常時、縮退時など）
運用・保守性	システムの運用と保守のサービスに関する要求	<ul style="list-style-type: none">・運用中に求められるシステム稼働レベル・問題発生時の対応レベル

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

非機能要求大項目	説明	要求の例
移行性	現行システム資産の移行に関する要求	<ul style="list-style-type: none"> ・新システムへの移行期間および移行方法 ・移行対象資産の種類および移行量
セキュリティ	情報システムの安全性の確保に関する要求	<ul style="list-style-type: none"> ・利用制限 ・不正アクセスの防止
システム環境・エコロジー	システムの設置環境やエコロジーに関する要求	<ul style="list-style-type: none"> ・耐震/免震、重量/空間（※）、温度/湿度、騒音など、システム環境に関する事項 ・CO₂排出量や消費エネルギーなど、エコロジーに関する事項

※空間：要求の一例として「機器の搬入経路と放熱のための十分な隙間を含めた設置場所を確保できること」

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

これらの大項目は、以下のとおり細分化されています。

大項目： 6 個



中項目： 3 5 個



小項目： 1 1 8 個



メトリクス（指標）

具体的にお客様と合意する部分

： 2 3 8 個（内：重要項目 9 2 個）

例えば・・・

項番	大項目	中項目	小項目	小項目説明	重複項目	重要項目	メトリクス (指標)
A.1.1.1	可用性	継続性	運用スケジュール	システムの稼働時間や停止運用に関する情報。	○	○	運用時間(通常)
A.1.1.2					○	○	運用時間(特定日)
A.1.1.3					○	○	計画停止の有無
A.1.2.1		業務継続性	業務継続性	可用性を保証するにあたり、要求される業務の範囲とその条件。		○	対象業務範囲
A.1.2.2						○	サービス切替時間
A.1.2.3						○	業務継続の要求度

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

現場ではこの表を利用して、お客様とすり合わせを行います。

- ・各項目について、どのくらいのレベルを実現すべきか？

それはなぜか？

なぜか、が重要

- ・選択するレベルはコストに見合うのか？

- ・どうやって実現するのか？

- ・双方の宿題の整理。

お客様でないとわからないこともある

(例：業務量の予測はお客様が担当)

非機能要求グレード 「レベル」

項番	大項目	中項目	小項目	小項目説明	重複項目	重要項目	メトリクス(指標)	レベル					
								0	1	2	3	4	5
A.1.1.1	可用性	継続性	運用スケジュール	システムの稼働時間や停止運用に関する情報。	○	○	運用時間(通常)	規定無し	定時内(9時～17時)	夜間のみ停止(9時～21時)	1時間程度の停止有り(9時～翌朝8時)	若干の停止有り(9時～翌朝8時55分)	24時間無停止

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

- ・メトリクスごとに、実現のコストや難易度が大きく変わるポイントで「レベル」が設けられています。
 - ・レベルに示されている値は、あくまで出発点です。
お客様と議論することで、実際の値を決めていきます。
- ➡ どのレベルを選ばいいのでしょうか？
- ➡ 「モデルシステム」を参考にします。

非機能要求グレード 「モデルシステム」

社会的影響が殆ど無いシステム	社会的影響が限定されるシステム	社会的影響が極めて大きいシステム
<p>企業の特定期間が比較的限られた範囲で利用しているシステムで、機能が低下または利用不可能な状態になった場合、利用部門では大きな影響があるが、その他には影響しないもの。</p> <p>例えば、<u>ごく小規模のインターネット公開システム</u>など。</p>	<p>企業活動の基盤となるシステムで、その機能が低下又は利用不可能な状態に陥った場合、当該企業活動に多大の影響を及ぼすと共に取引先や顧客等の外部利用者にも影響を及ぼすもの。</p> <p>例えば、<u>企業の基幹システム</u>など。</p>	<p>国民生活・社会経済活動の基盤となるシステムで、その機能が低下又は利用不可能な状態に陥った場合、国民生活・社会経済活動に多大な影響を与えるもの。</p> <p>例えば、<u>不特定多数の人が利用するインフラシステム</u>など。</p>

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

非機能要求グレード レベルの選択

項番	メトリクス (指標)	レベル						社会的影響が殆ど無いシステム			社会的影響が限定されるシステム		社会的影響が極めて大きいシステム			
		0	1	2	3	4	5	選択レベル		選択時の条件		選択レベル		選択時の条件		
A.1.1.1	運用時間(通常)	規定無し	定時内 (9時～17時)	夜間のみ 停止 (9時～21時)	1時間程度の 停止有り (9時～翌朝8時)	若干の停止 有り (9時～翌朝8時55分)	24時間無 停止	2	夜間のみ 停止 (9時～21時)	夜間を実施する 業務はなく、システムを停止可能。	4	若干の停止 有り (9時～翌朝8時55分)	24時間無停止での 運用は必要ないが、極力システムの稼働は継続させる。	5	24時間無 停止	システムを停止できる時間帯が存在しない。

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

レベルに示されている値を出発点として、お客様と議論をします。
その結果、実際の値が決まっていきます。

もちろん、「レベルを決めて終わり」ではありません。
具体的にいつなら停止して良いか、等の詳細化が必要です。
(レベル4でも停止時間は5分なのか10分なのか、停止時刻は、等)

「お客様の要求」を整理するための

コミュニケーション・ツール

漏れがないという利点もありますが、コミュニケーションツールとしての側面が強いです。

お客様にとっては、「非機能」について何を考慮すればよいのか、そのきっかけを与えてくれるものになります。

非機能要件についての重要なトピック

以降のスライドでは

非機能要求グレードの項目のうち
特に優先度の高い、以下の6つの小項目にしばって要件の合意、設計、実装を
適切に行なうための知識をお伝えします。

「優先度が高い」とは・・・

- ・現場で考慮する頻度が特に高い
- ・顧客ビジネスへの影響が特に大きい

大項目	中項目	小項目
可用性	継続性	運用スケジュール
		業務継続性
		稼働率
性能・拡張性	業務処理量	通常時の業務量
	性能目標値	オンラインレスポンス
		バッチレスポンス (ターンアラウンドタイム)

非機能要求グレード(2018)より作成

<https://www.ipa.go.jp/archive/digital/iot-en-ci/ent03-b.html>

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

非機能の検討では、システム全体のサーバ構成を描ける必要があります。

これができないと
社内での議論やお客様との会話についていきません。

そのために、一般的なWebアプリケーションの構成で登場する
「主要なインフラ」について解説します。

業務アプリケーションの開発を主に担当する場合であっても

「常識」として必要になる知識です。

しっかりと身につけましょう。

サーバ構成、描けますか？

一般的なW e bアプリケーションでは、
サーバ構成の要素として以下のようなものが登場します。

アプリケーションサーバ（A P）

W e bサーバ（W E B）

D Bサーバ（D B）

ロードバランサ（L B）

【演習】 サーバ構成を描く

「一般的なW e b アプリケーション」では、これらのサーバがどのような構成となるか知っていますか？

手元のメモ用紙などに、図を描いてから次のスライドに進みましょう。

A P

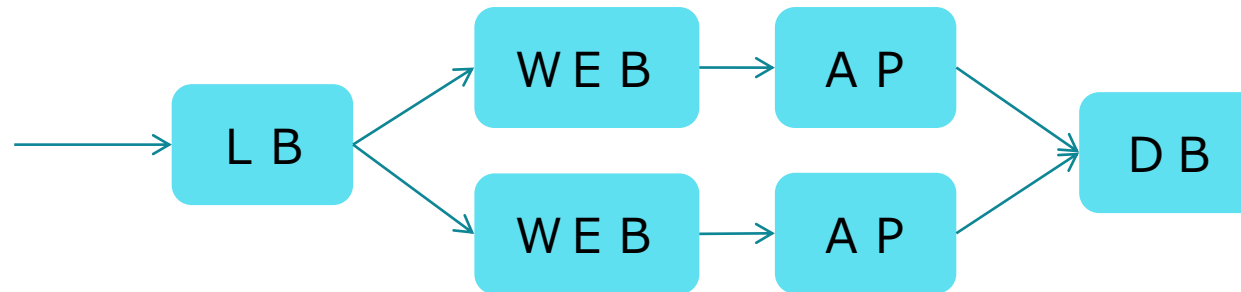
W E B

L B

D B

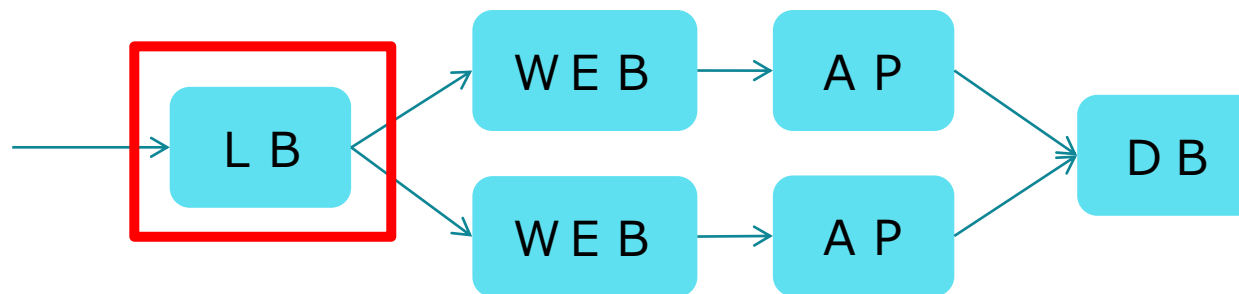
【演習回答】 サーバ構成を描く

一般的なW e b アプリケーションの構成



各要素について見ていきます。

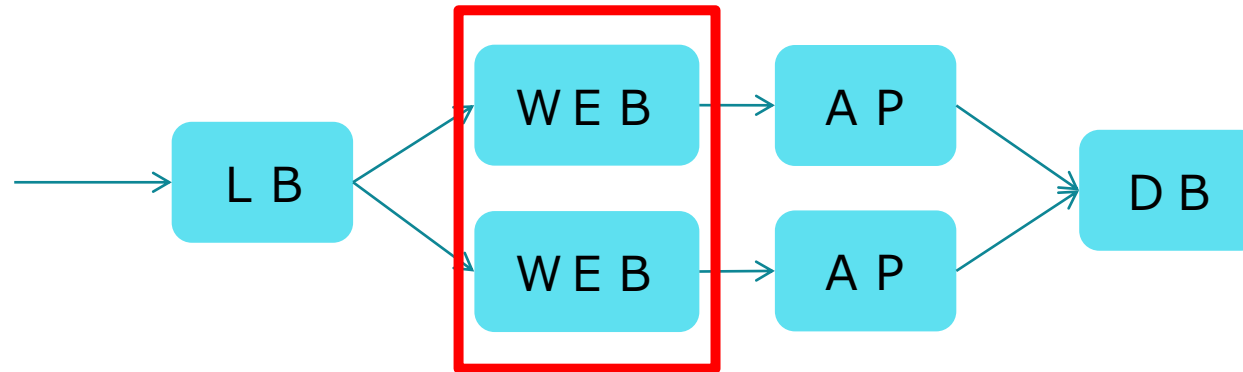
ロードバランサ（LB）



システム外からの要求を一手に受けて
いずれかのWEBサーバにその要求を振り分けます。

これにより2つの事が実現できます・・・

- ・後続サーバ群の負荷を分散できます。（負荷分散）
- ・後続サーバ群を冗長化できます。（可用性UP）



静的リソースを保持します。

静的リソースとは・・・？

ユーザーに送り返すもののうち
状況によって内容が変わらないもの

例) アプリケーションで使用するCSS, JavaScript, 画像

WEBサーバ (WEB)

ロゴ画像



TISの特長 サービス 導入事例 ニュース 企業情報 IR情報 サステナビリティ

HOME > TISインテックグループ > TISインテックグループのサービス

TISインテックグループのサービス

キーワードで探す

ソリューション名、キーワードを入力ください



課題から探す

7件ご紹介できます



ビジネスモデルを
変革したい

☒ 決済/請求/回収

☐ グローバル

☐ 開発基盤

☐ コンサルティング

☐ Fintech

☐ エネルギー

☐ プロジェクトマネジメント

☐ IFRS

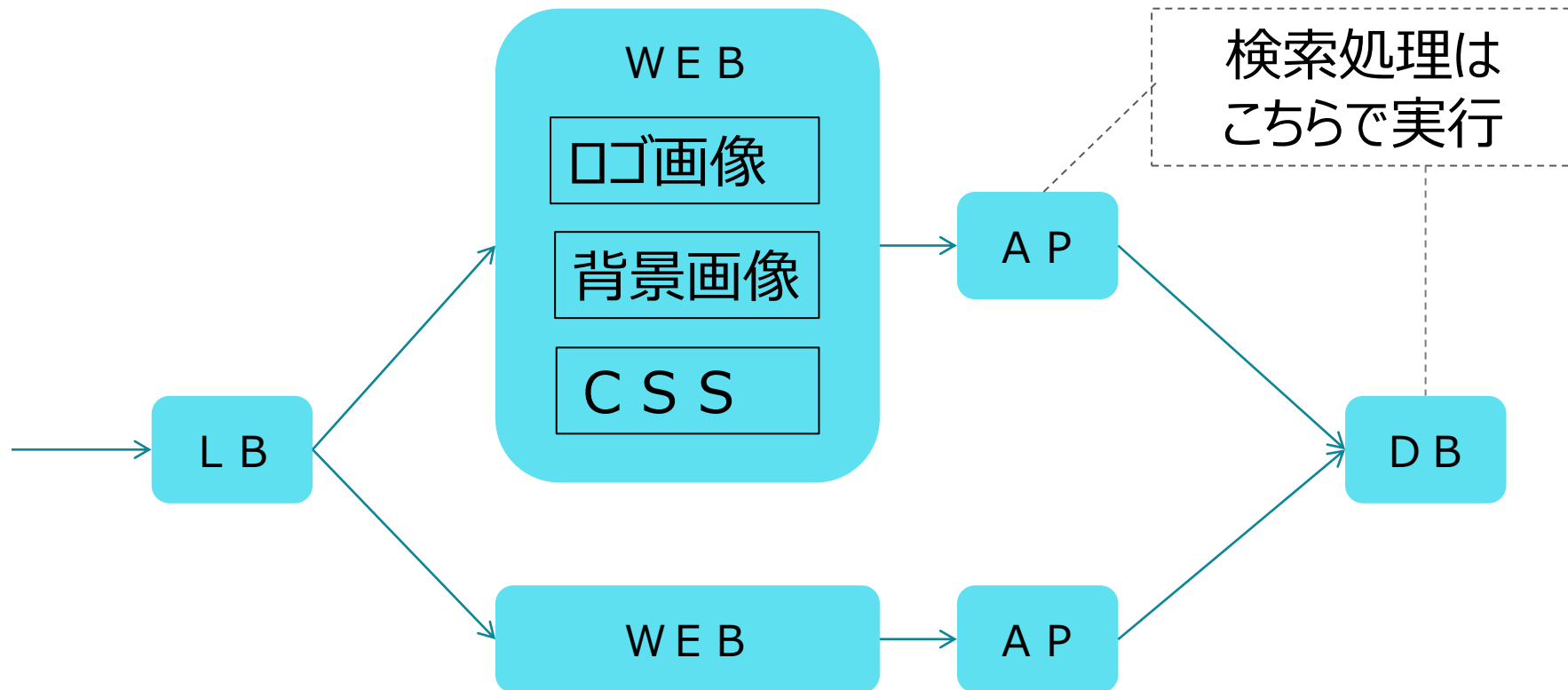
7件ご紹介できます

× 検索条件をクリア

検索する

背景画像

WEBサーバ (WEB)

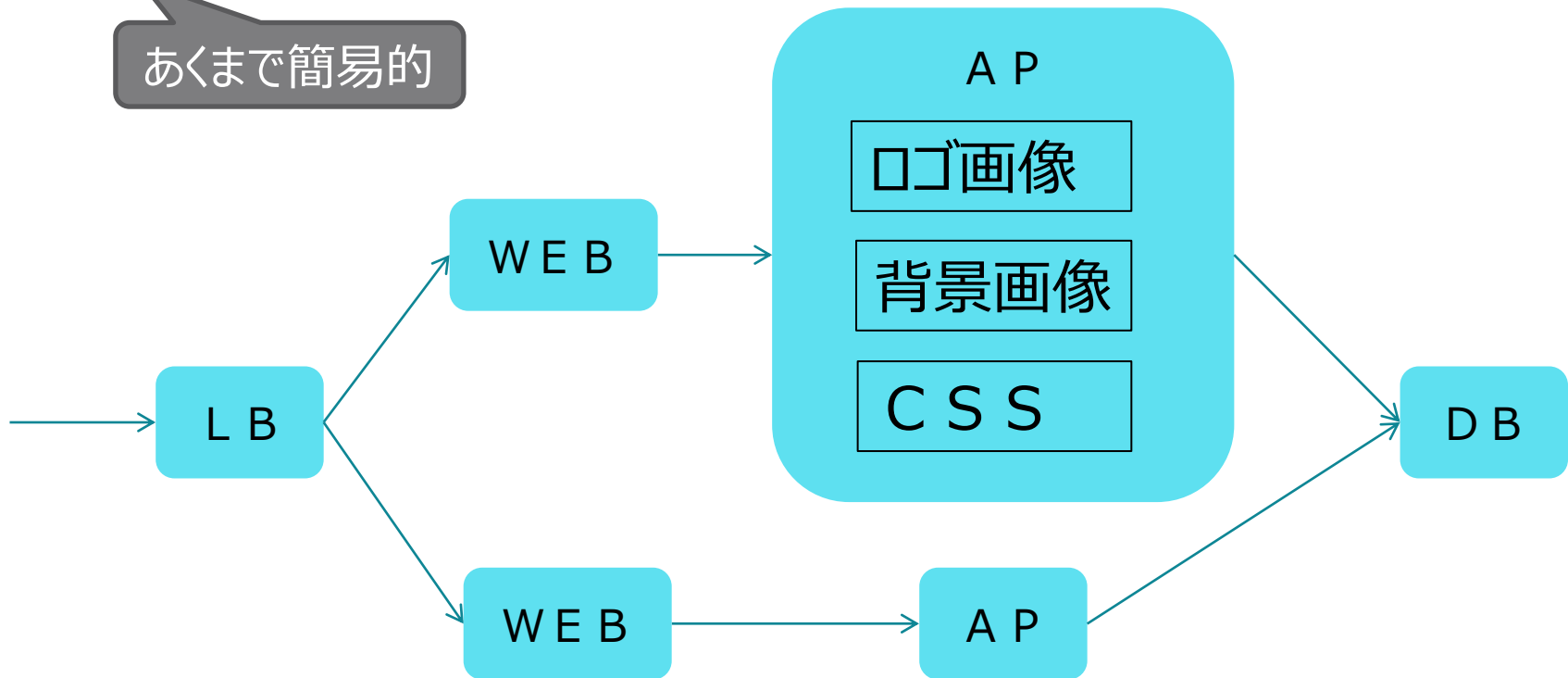


静的リソースをWEBサーバに持たせると
何がうれしいのでしょうか？

WEBサーバ (WEB)

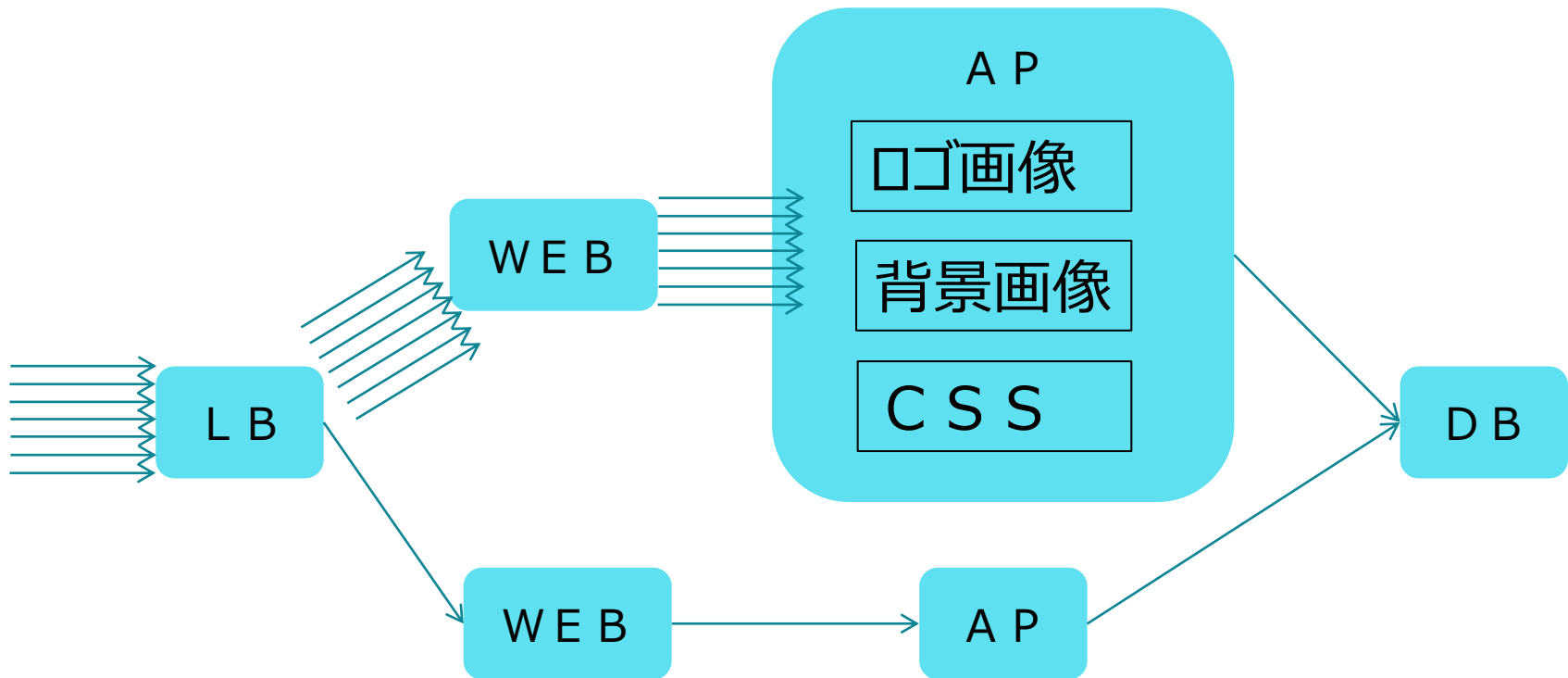
A Pサーバにも
簡易的なWEBサーバ機能
がありますが...

あくまで簡易的



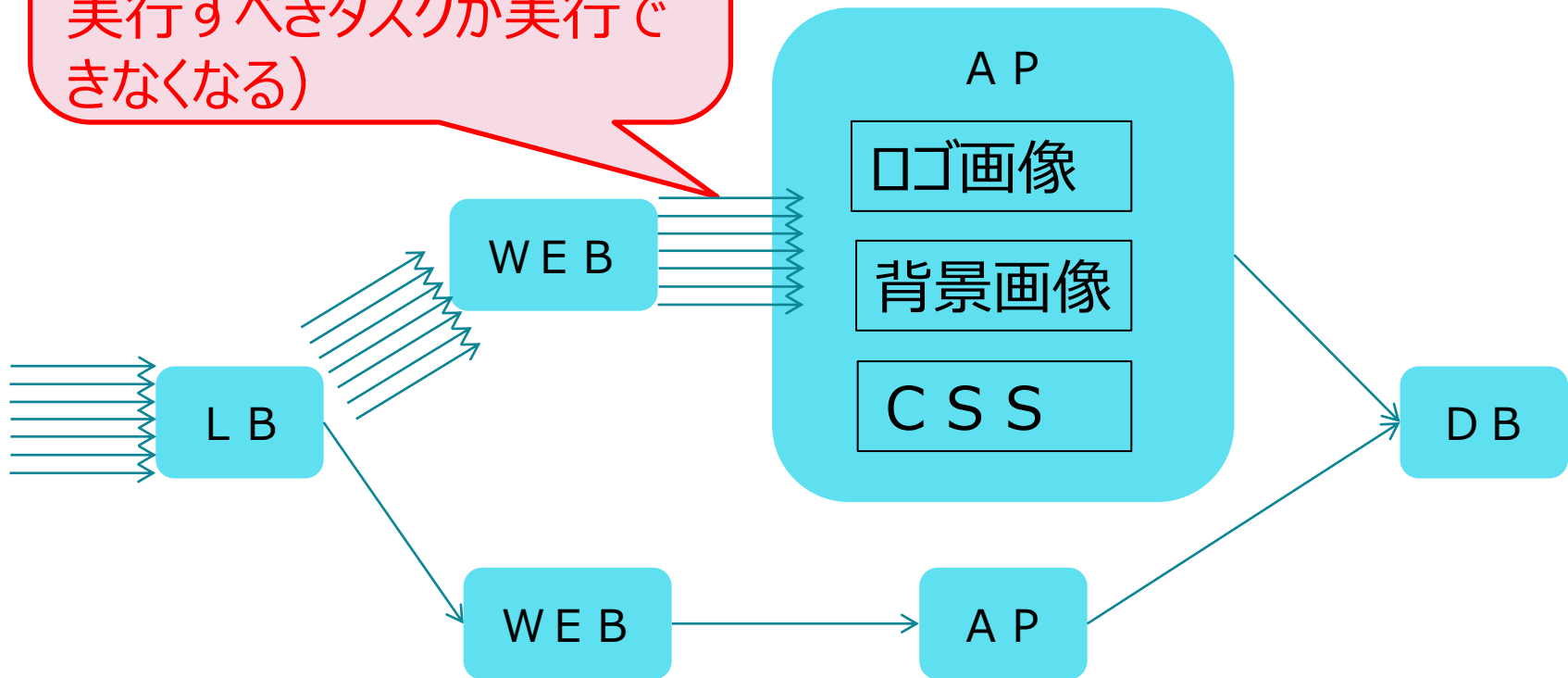
WEBサーバ (WEB)

大量のアクセスがあると...

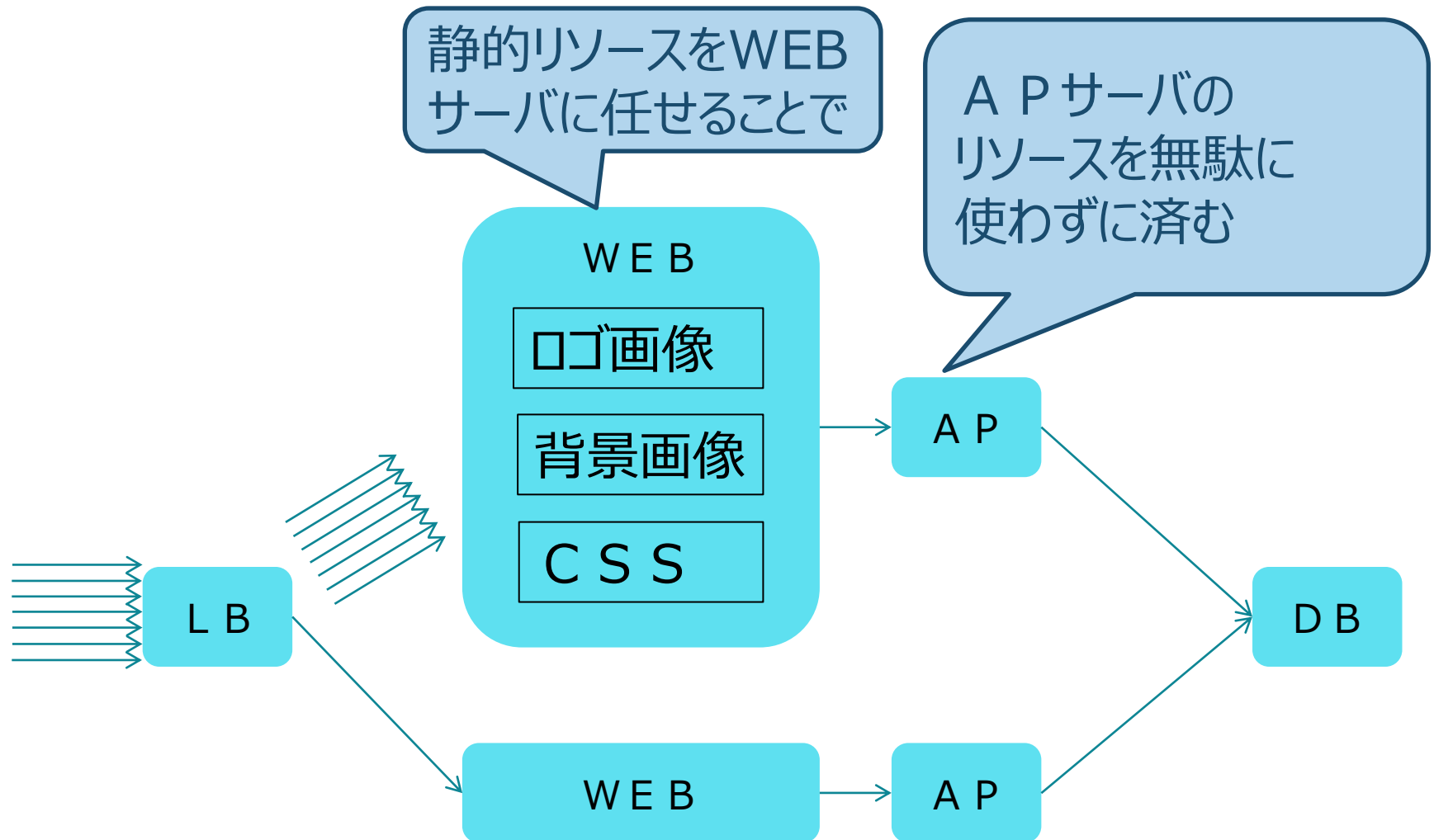


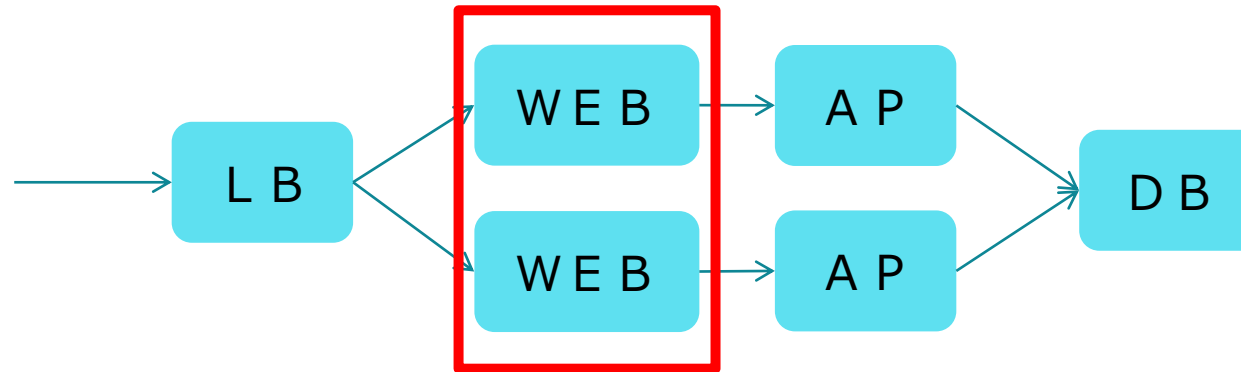
WEBサーバ (WEB)

APサーバのリソースが
無駄に使われてしまい
さばききれなくなる（本来
実行すべきタスクが実行で
きなくなる）



WEBサーバ (WEB)



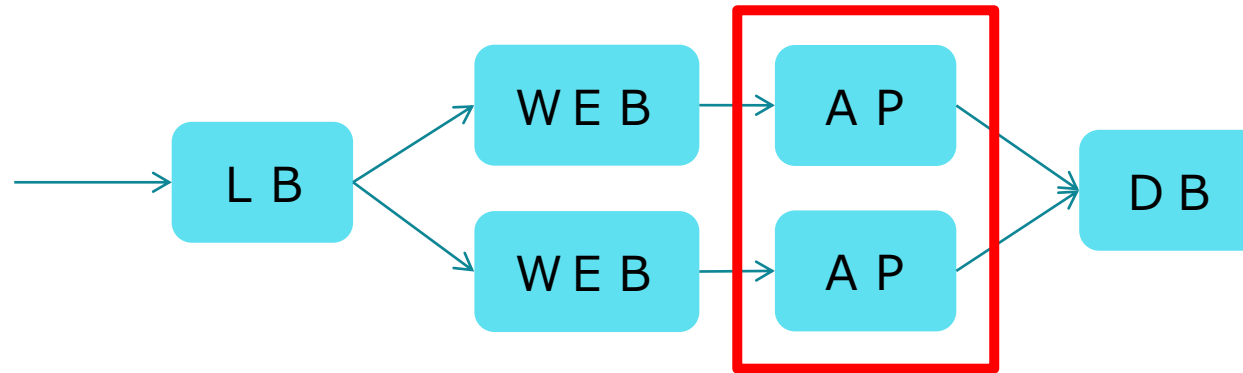


Apache HTTP ServerなどのソフトウェアをLinuxマシンなどにインストールすることが多いです。

A Pサーバの前に置くことで、セキュリティを向上させます。
たとえば、D O S 対策をここに組み込むことがあります。

特定IPからのリクエストをシャットアウトする

A Pサーバ (A P)



アプリケーションが稼働するサーバです。

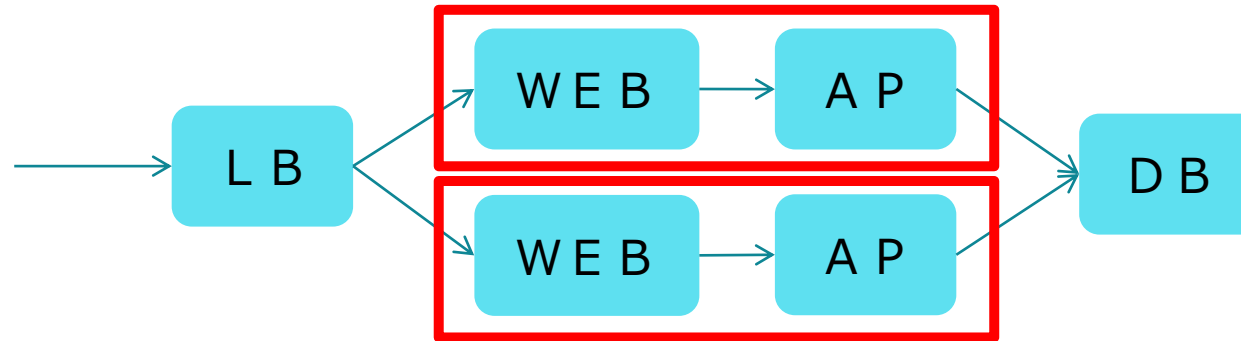
A Pサーバ (A P)

アプリケーションが稼働するサーバ、とは？

A P

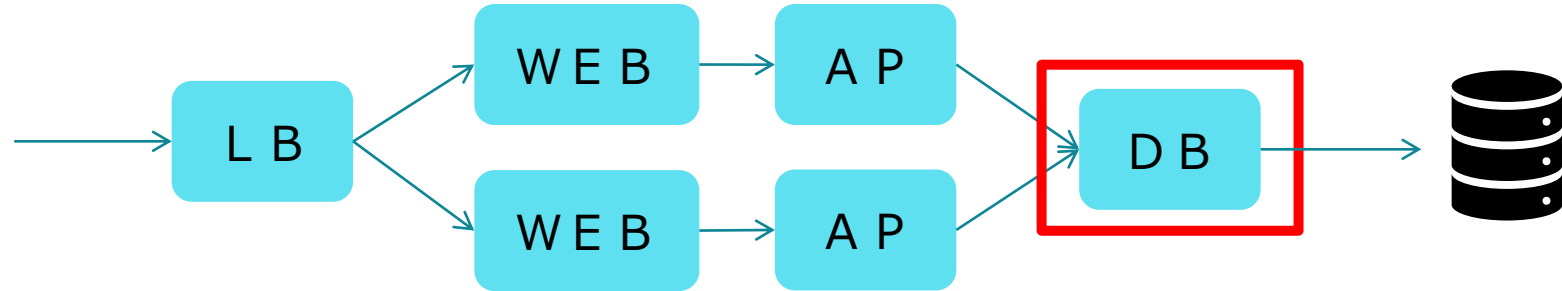
層	実際の例
業務アプリケーション	● ● 登録機能 ● ● 一覧機能
アプリケーションサーバ	Tomcat, WebLogic
アプリケーションサーバ製品の動作に必要なライブラリ	Java SE Java EE
OS層	Red Hat Enterprise Linux

WEBとAPは複数系列



可用性の確保や、負荷分散の実現のため
WEBサーバとAPサーバは通常複数の系列で
構成されます。

DBサーバ (DB)



DBMS (Oracle, MySQLなど) が稼働します。

ディスクに対して、業務データの読み書きをします。

➡ メモリに比べれば時間のかかるディスクI/Oが発生します。

「ちょっと全体像を書いて整理してみて」と言われたときに
こういった図をササッと描けるようにしておきましょう。

社内での議論や、お客様との会話についていくために
とても重要なことです。

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

「可用性」とは、
システムが利用できる状態が継続されている度合い。

この度合いを数値化したものが**「稼働率」**。

可用性を高める基本方針は
「冗長化」を施し、**「単一障害点」**をなくすこと。

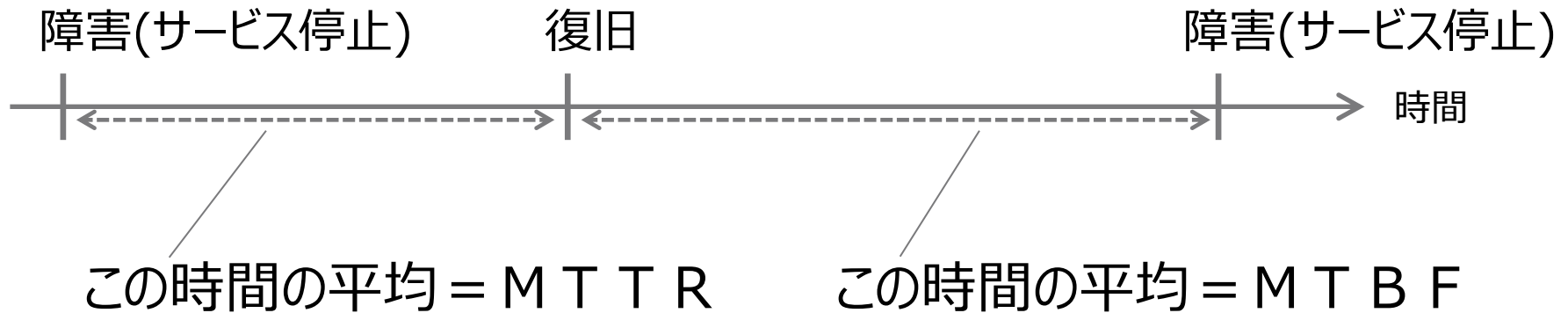
「計画停止」がどれだけ許容されるか、も論点となる。

システムを計画的に停止すること

停止させている間に、以下をおこないます。

- ・アプリケーションのリリース
- ・インフラへのパッチ適用（OS, ミドルウェア） など

$$\text{稼働率} = \frac{\text{MTBF (平均故障間隔)}}{\text{MTBF (平均故障間隔)} + \text{MTTR (平均復旧時間)}}$$



※ MTTR に計画停止の時間を含めて考えるお客様もいます。しかし、MTTR の意味（不測の事態から復旧するまでの時間）と異なりますし、求められる稼働率の実現が困難となってしまうため、MTTR に計画停止の時間を含めないように、お客様と合意しましょう。

稼働率 = **明示された利用条件**の下で、システムが要求されたサービスを提供できる割合。

明示された利用条件とは・・

- ・システムをユーザーに開放する時間は何時～何時？
- ・24時間365日の稼働が必要？

など、お客様と事前に合意しておいた利用条件

【参考】

24時間365日の稼働の場合、1年間で業務が中断する時間の合計は以下のとおりです。

95%.....18.3日

99%.....87.6時間

99.9%..... 8.76時間

99.99%..... 52.6分

99.999%..... 5.26分

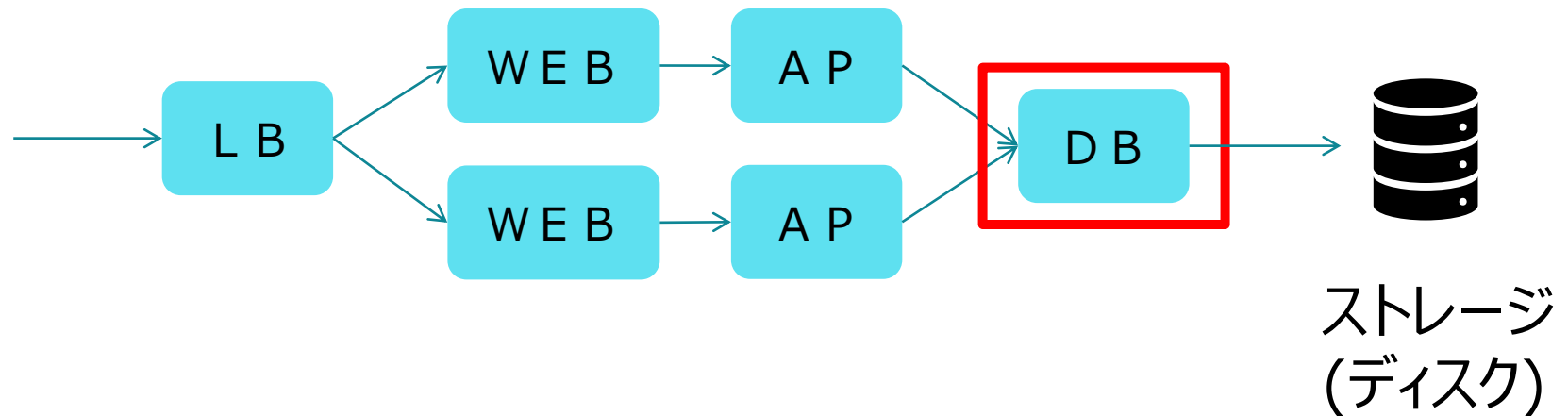
実際の稼働率は通常の機能と異なり、長期間にわたって実際に運用しない限り検証できないため、リリース前の検証はほぼ実現不可能です。

そこで、多くの現場では以下を行います。

- ・机上でシステム全体の稼働率を算出する
- ・障害テストを行う(待機系に切り替わるか等)

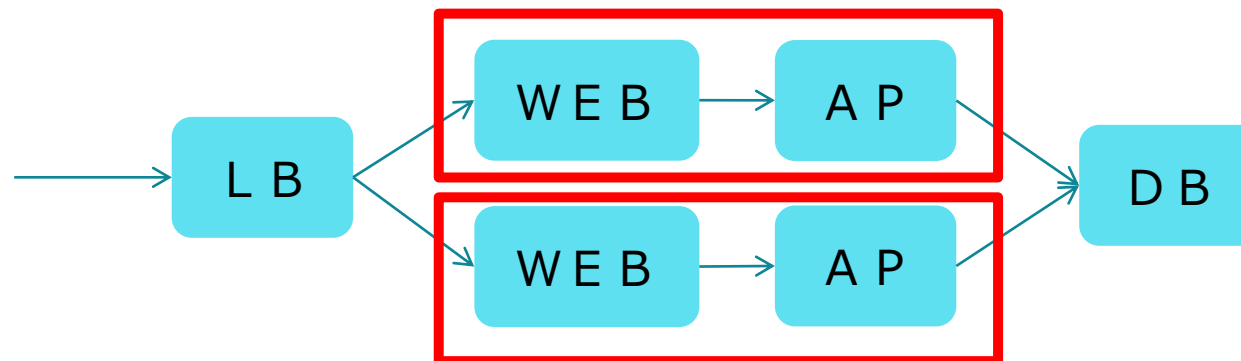
その結果をお客様に検収いただくことになります。

単一障害点 = その単一箇所が働かないと
システム全体が障害となるような箇所。
シングルポイントとも呼ばれる。



典型的にはD Bサーバが該当。
D Bサーバの可用性向上策については、後述します。

冗長化 = 障害発生後でもシステム全体の機能を維持し続けられるように、予備装置を平常時からバックアップとして配置し運用しておくこと



「可用性」とは、
システムが利用できる状態が継続されている度合い。

この度合いを数値化したものが**「稼働率」**。

可用性を高める基本方針は
「冗長化」を施し、**「単一障害点」**をなくすこと。

「計画停止」がどれだけ許容されるか、も論点となる。

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

要求や要件はどこに書いてある？

典型的には、R F Pに記載してあります。

R F P（Request For Proposal）は
お客様が作成する資料です。

お客様がベンダーに対して
提案にあたっての要求事項を示すものです。

ベンダーはこれをインプットに、提案書を作成します。

背景や要求を正しく理解しましょう

ビジネス上の背景



昨今、カードの不正利用が多いので、カードが利用されたら、本人に即時でメール通知をしたい。業界的にはやるのが当たり前になってきているので、他社に追随したい。

要求

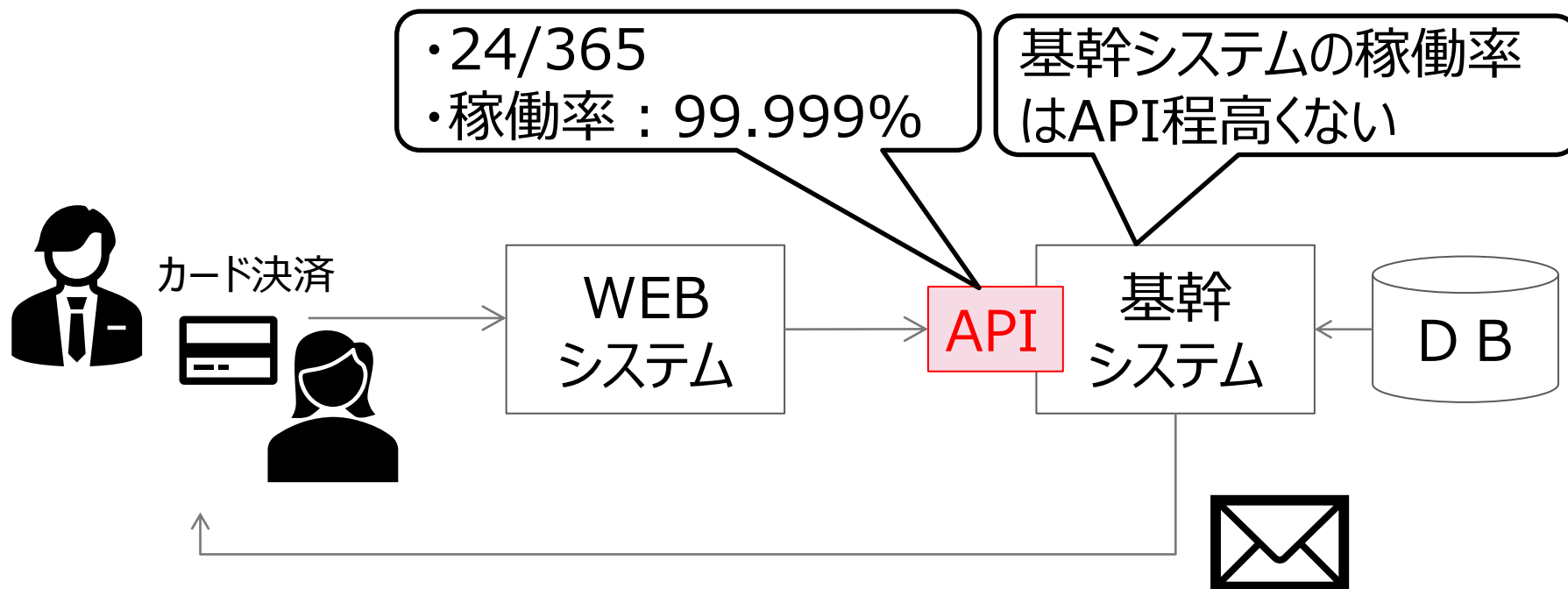


既存の基幹システムに24/365で利用できるAPIを**追加**して、WEBのシステムから利用できるようにしてほしい。

要件

基幹システム（に追加したAPI）の稼働率は99.999%とする。
（年間の停止時間上限：約5分）

背景や要求を正しく理解しましょう



APIの稼働率を実現するには基幹システムのインフラ構成の抜本的変更が必要。
しかし、既存の基幹システムのインフラ構成を変えるのは、非現実的・・・
(高コスト、高難易度、長期間)

背景や要求を正しく理解しましょう

あなたが「ビジネス上の背景」を知らなかったとしたら
どうやって現状を打開できるでしょうか・・・？

ここにだけ
注目すればよいので
しょうか？

要求



要件

既存の基幹システムに24/365で利用できる
A P I を**追加**して、W E B のシステムから利
用できるようにしてほしい。

基幹システム（に追加したAPI）の稼働率は
99.999%とする。
（年間の停止時間上限：約 5 分）

背景や要求を正しく理解しましょう

ビジネス上の背景



要求



要件

昨今、カードの不正利用が多いので、カードが利用されたら、本人に即時でメール通知をしたい。業界的にはやるのが当たり前になってきているので、他社に追随したい。

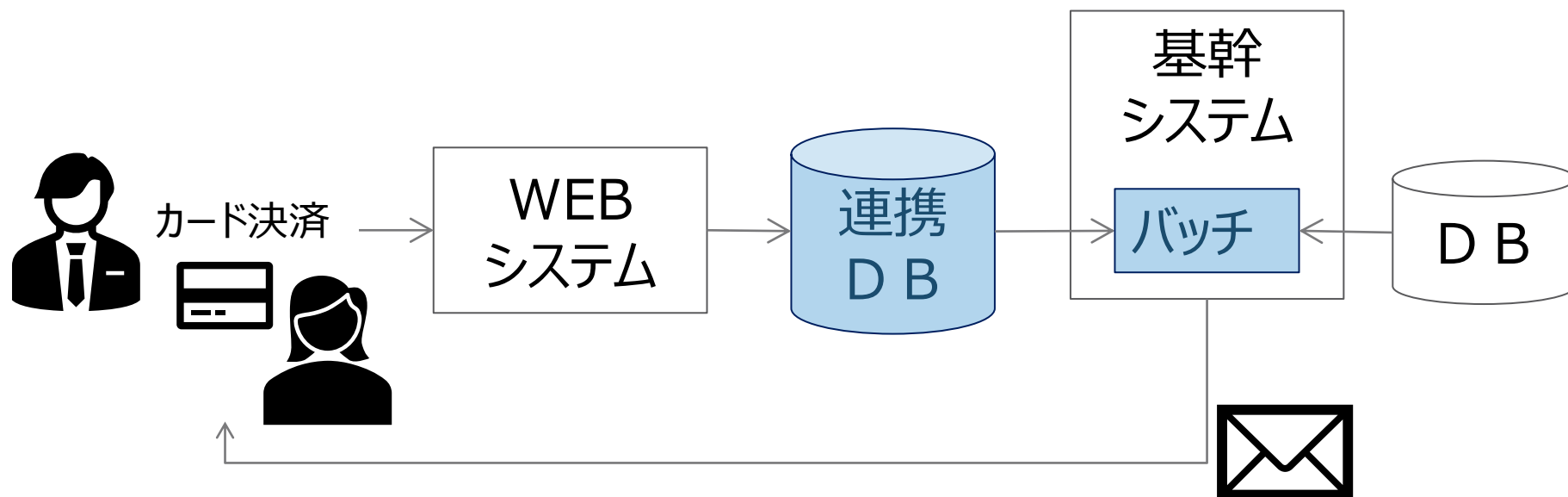
この情報をお客様から引き出し、ここまで立ち返る。

→

お客様は**APIを作りたいのではない**。
真にやりたいのは**メール通知**。

メール通知を実現するのであれば、APIを作る以外の現実的な方式を提案できる！

背景や要求を正しく理解しましょう



この構成なら

- ・ビジネス上の背景にマッチする。
- ・当初のレベルの可用性は必要ない。

背景や要求を正しく理解しましょう

このように
お客様の提示する「要件」に固執するのではなく
「要求」や「ビジネス上の背景」を正しく理解することが重要です。
(可用性に限ったことではありません。)

そうしなければ、現実的な解決策を導けません。

結果、お客様に無駄な投資をさせることになり、
お客様のビジネスを失敗に導いてしまいます。

(お客様の予算感に合わず、失注になることもあります)

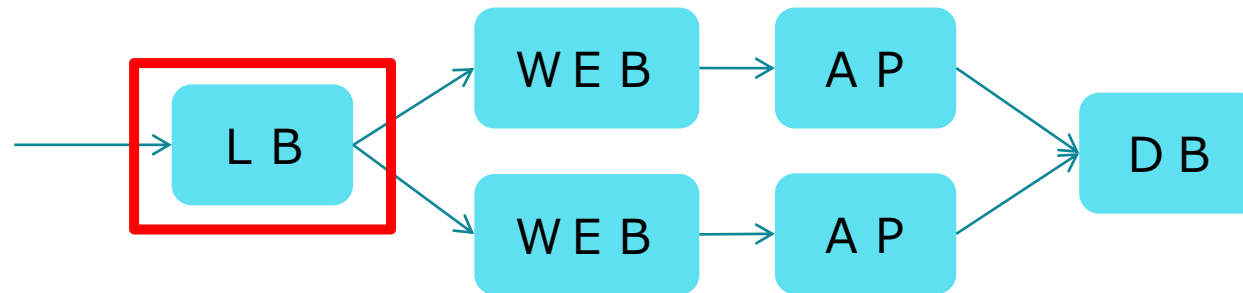
- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

では、可用性要件の各レベルを実現するためにどのような実現方式があるのでしょうか？

典型的なサーバ構成を例に、説明していきます。

まずは、「Webアプリケーション」でのサーバ構成を見ていき、次に「バッチ」を見ていきましょう。

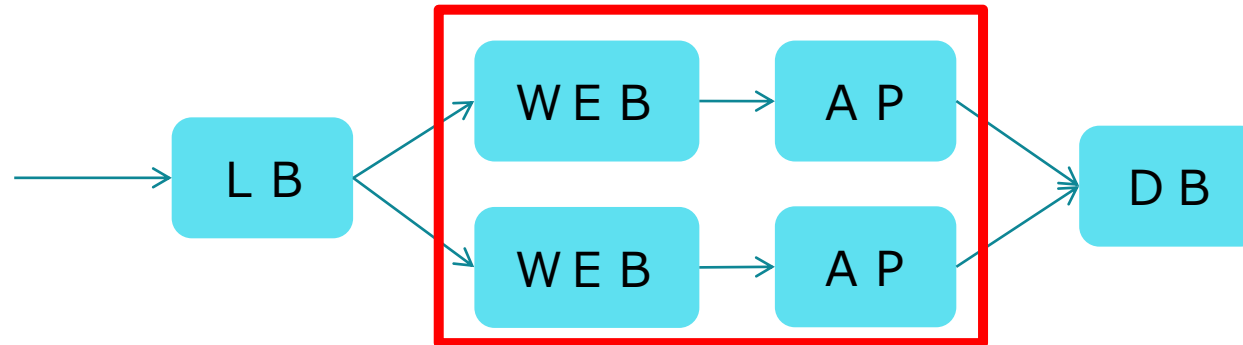
実現方式（L B）



可用性の 要件レベル	実現方式	
	ハードウェア／ミドルウェア	系列
高い （非常に高い。以降も同じ）	ハードウェア L B 極めてハイスペックな製品を購入する。ハードウェアから作り込まれており、かなり高価。（F5ネットワークスジャパン社のBIG-IPシリーズなど）	2系列 Active/Activeで構成でき、空白時間無しで切り替え可能。
低い （それ程高くない。以降も同じ）	ソフトウェア L B オープンソースのソフトウェア L B ツールを、Linuxマシンなどにインストールする。安価で実現可能。（Apacheのモジュールなど）	2系列(Activeな系列は1つのみ) Active/Stand-byで冗長化できるが、お互いの死活監視が一定の時間間隔で行われる等の理由で、 <u>切り替え時にある程度の空白時間が生まれてしまう。</u> = 単一障害点となり得る

※あくまでも典型例です。状況によって異なる場合があります。

実現方式 (WEB, A P)



実現方式（WEB, AP）

可用性の 要件レベル	実現方式
	系列
高い	3 系列以上 多重障害（複数系列で同時に障害が発生してしまうこと）が許容されない場合は、さらに系列を増やす
低い	2 系列 （多くの場合、1 系列とはしない）

※あくまでも典型例です。状況によって異なる場合があります。

ちなみにWEBサーバ, APサーバでは

- ・OSに、どのLinuxディストリビューションを使うか。
例) Red Hat Enterprise Linux、Debian
- ・APサーバに、ベンダー製品とオープンソースのどちらを使うか。
例) WebLogic, Tomcat

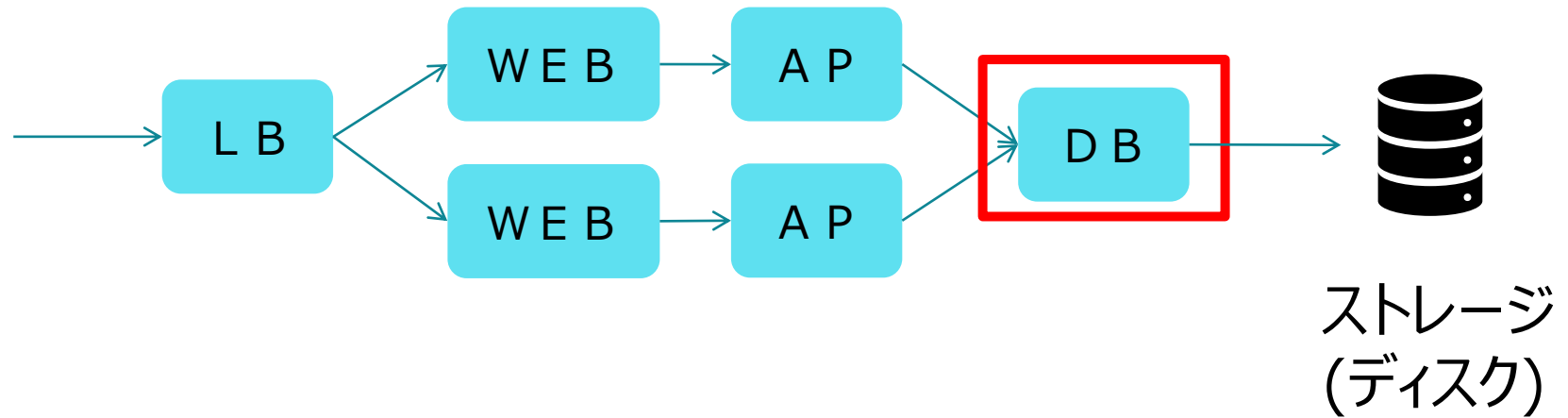
といった選択肢があります。

どれを選ぶかは、
可用性の要件とは無関係に決まります。

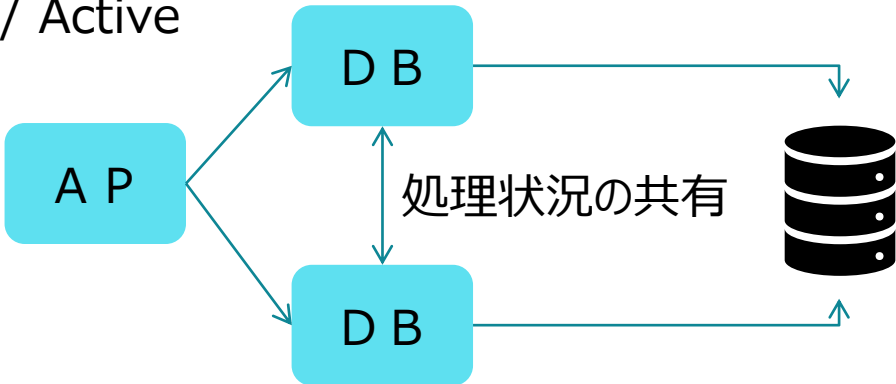
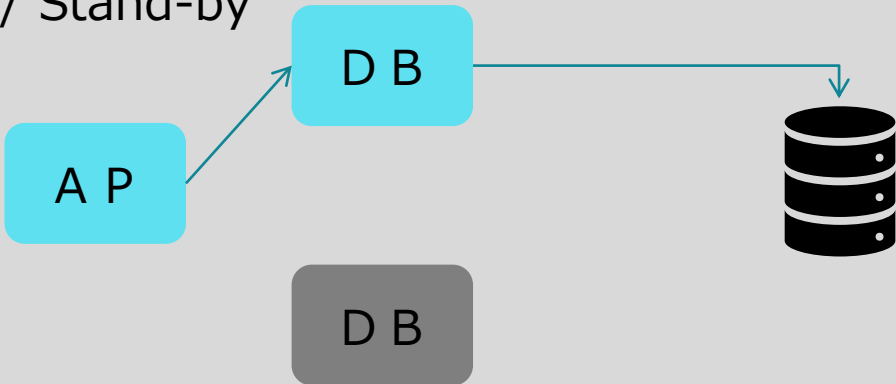
- ・アプリケーションの機能を実現できるか？
（昔は、製品でないと実現できない機能があった。EJBなど）
- ・PJで採用予定のサーバ構成やソフトウェアで
動作検証されているか？（相性は良いか？）
- ・OSパッチ適用の早さは十分か？（製品 ⇒ OSSの順番）

このため、可用性の要件が高いからといって、必ずしもベンダー製品が
利用される、というわけではありません。

実現方式 (D B)



実現方式（D B）

可用性の 要件レベル	実現方式	備考
高い	Active / Active 	Oracle RAC など。 かなり高価。
低い	Active / Stand-by 	比較的安価。

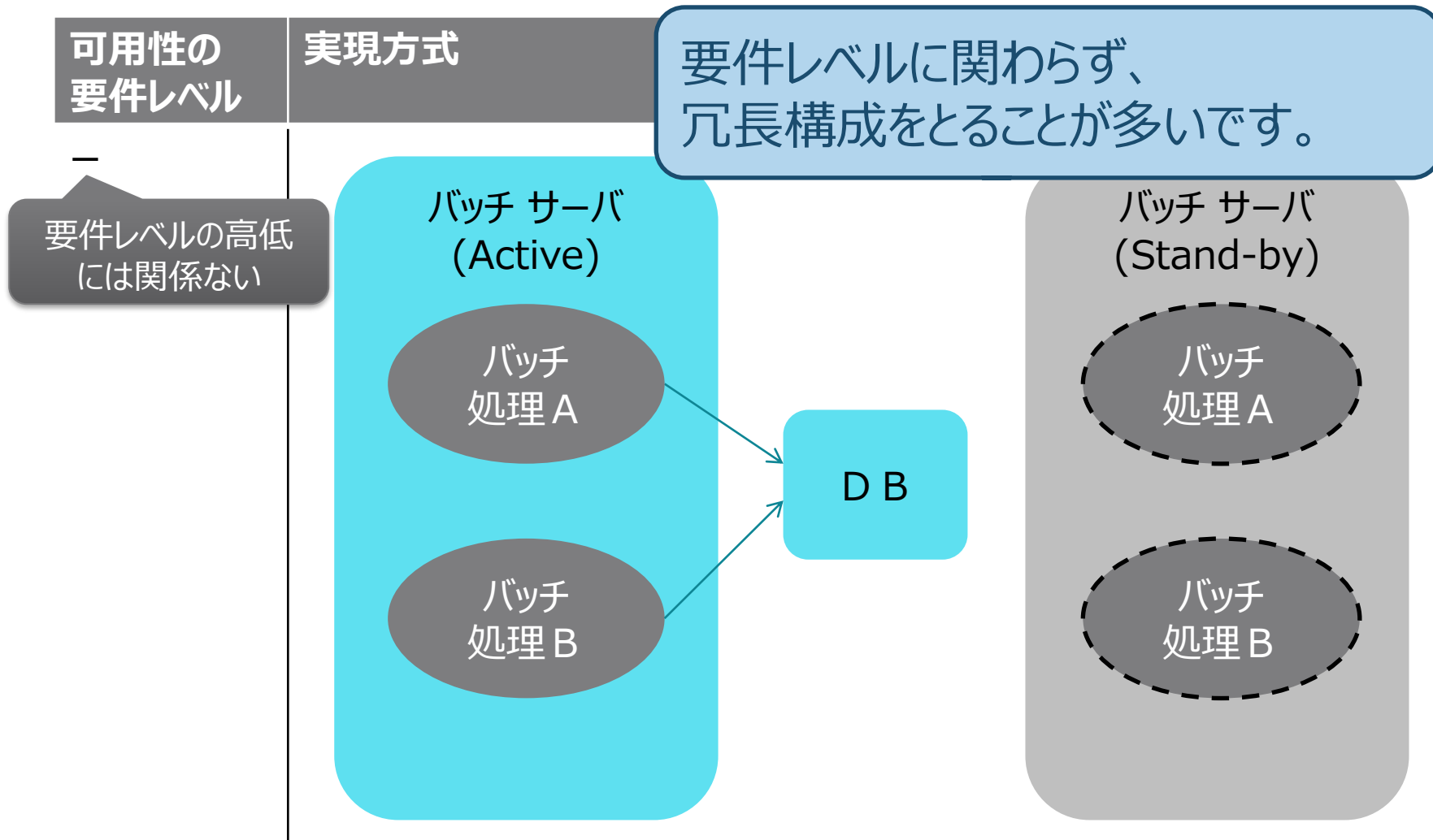
※あくまでも典型例です。状況によって異なる場合があります。

続いて、「バッチ」の構成を見ていきましょう。

バッチでは、バッチプロセス稼働するサーバを冗長化します。

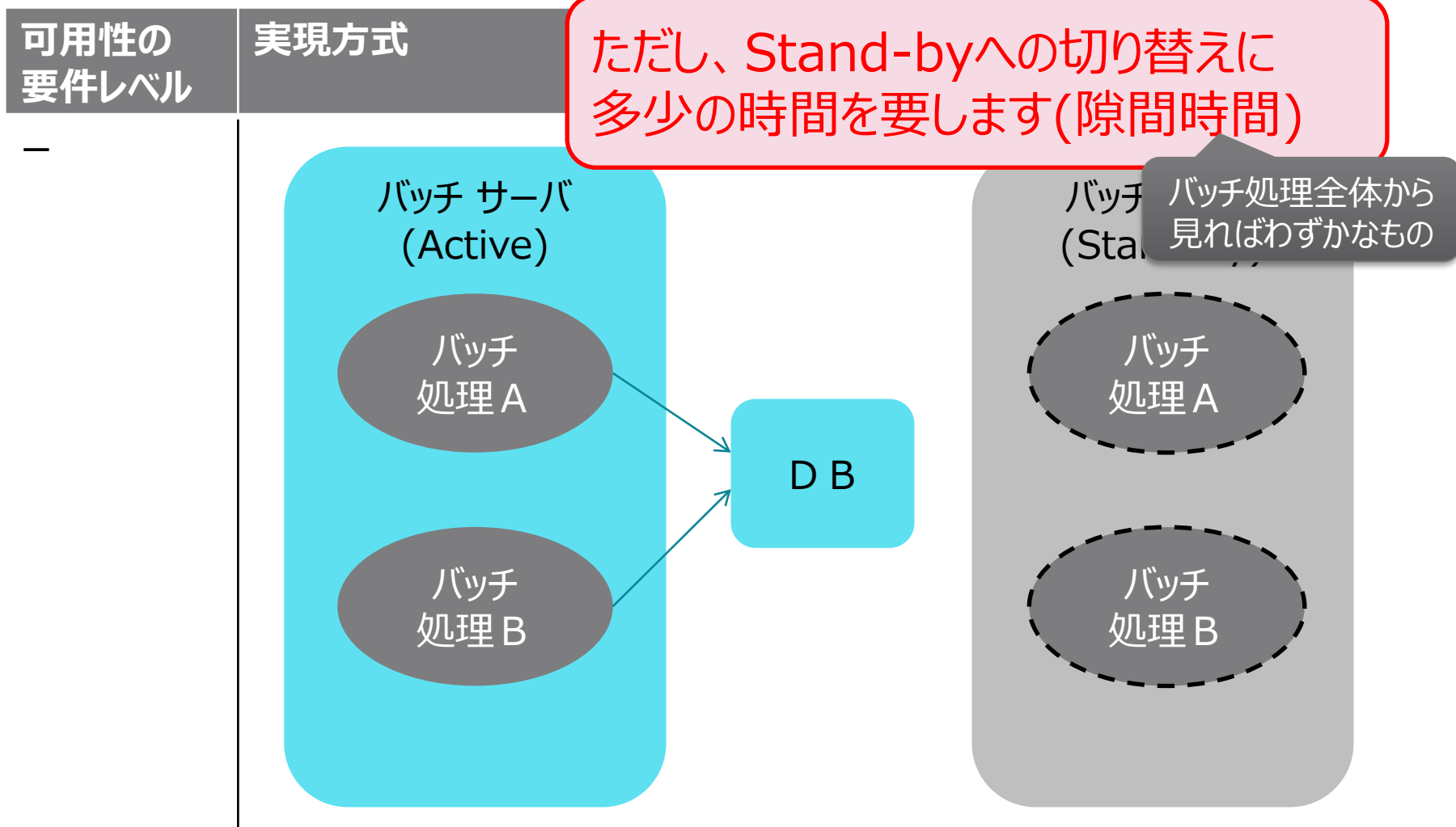
実現方式（バッチ）

都度起動バッチ（単発バッチ／時刻起動バッチ）



実現方式（バッチ）

都度起動バッチ（単発バッチ／時刻起動バッチ）



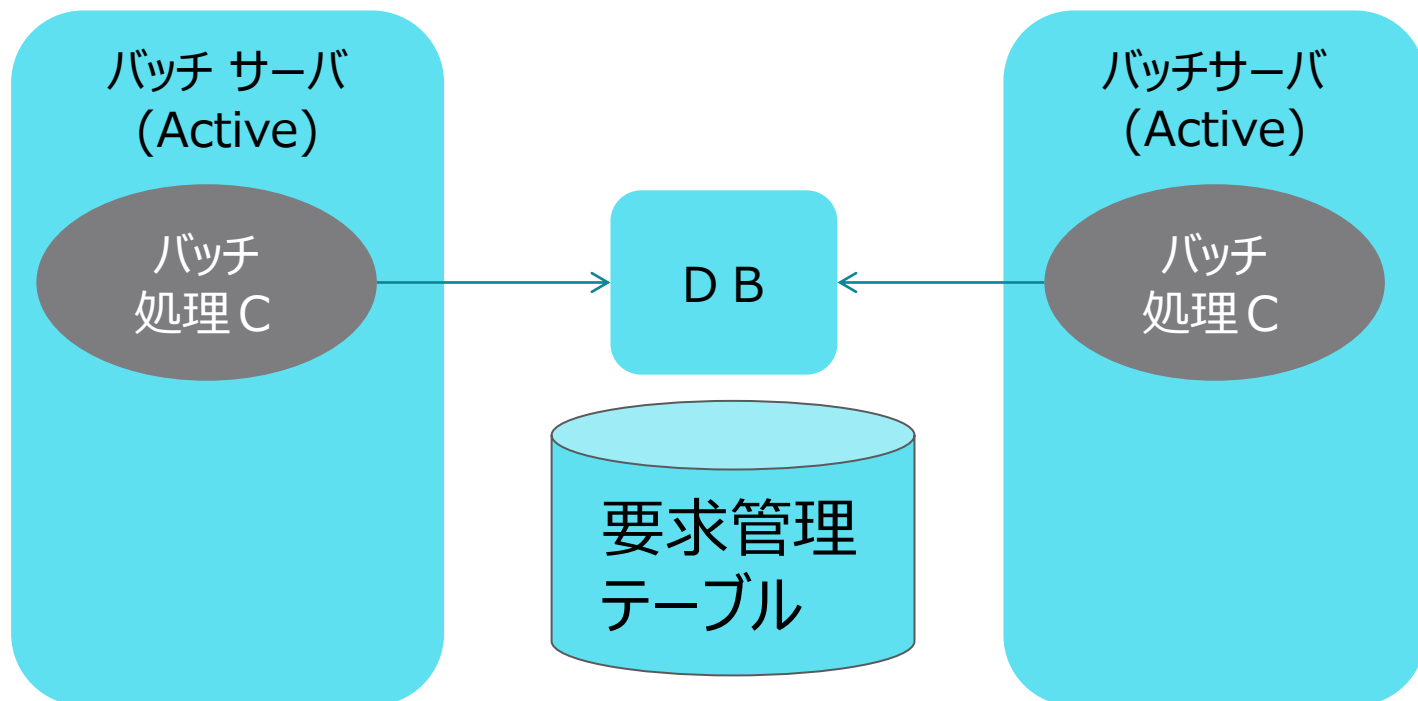
実現方式（バッチ）

常駐バッチ

可用性の
要件レベル

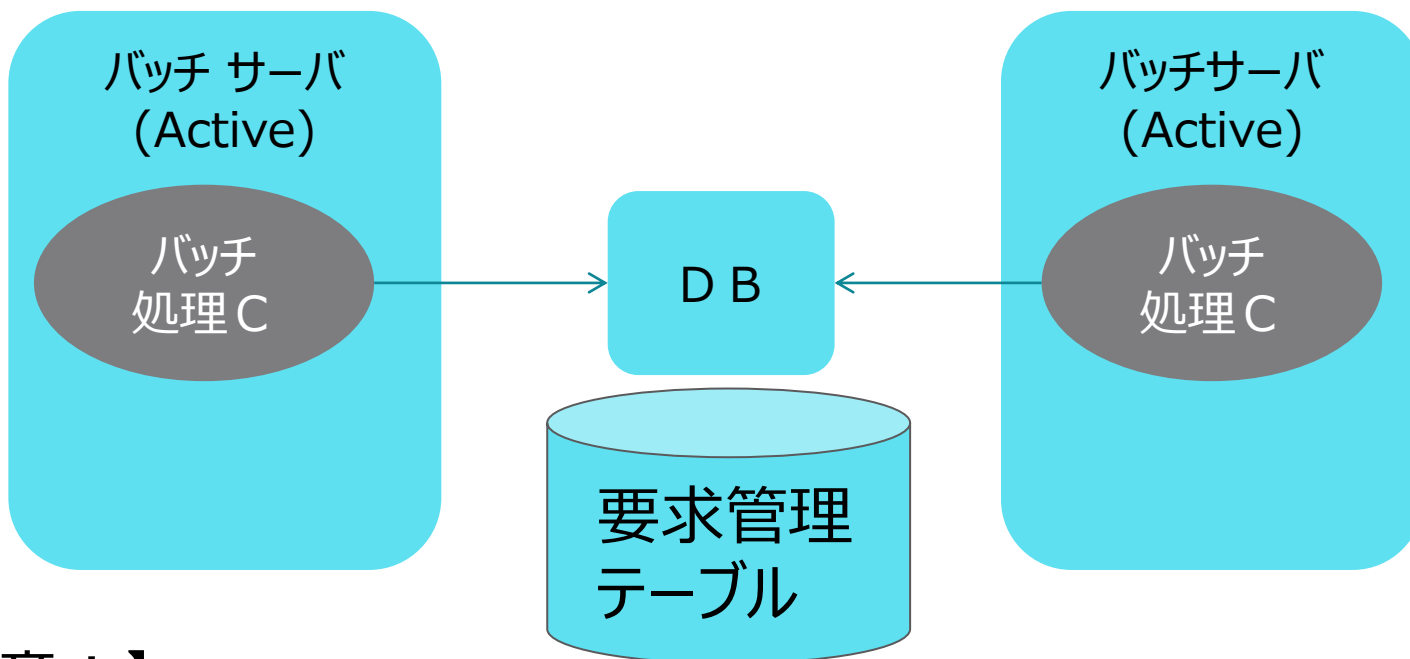
高い

両系ともActiveのため、片系がダウンしても
隙間時間なく処理を継続できます。



低い

都度起動バッチと同じ。
Stand-byへの切り替えに多少の時間を要します(隙間時間)。



【注意！】

同じデータを重複して処理しないような仕掛けが必要です。

※実現方法について詳しく知りたい方は以下を参照ください。

Fintan > アプリケーションアーキテクチャの学習コンテンツ > データベースの排他制御

<https://fintan.jp/?p=8376>

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

タイムアウトとは？

タイムアウト

待ち時間が閾値を超えた場合

待ちの状態を打ち切って、システムの資源を解放すること

システムの資源：CPU、メモリ、DBのロックなど

なぜ必要？

システム資源の過消費の防止。

特定の処理がリソースを占有してしまうと

他の処理がそれらのリソースを使えない。

→ **可用性の低下**

【参考】他にも、以下の目的があります。

- ・必要以上にシステムの利用者を待たせない。
- ・システム障害、ハングアップを早期に検知する。

タイムアウトが無いとこうなる

業務アプリ

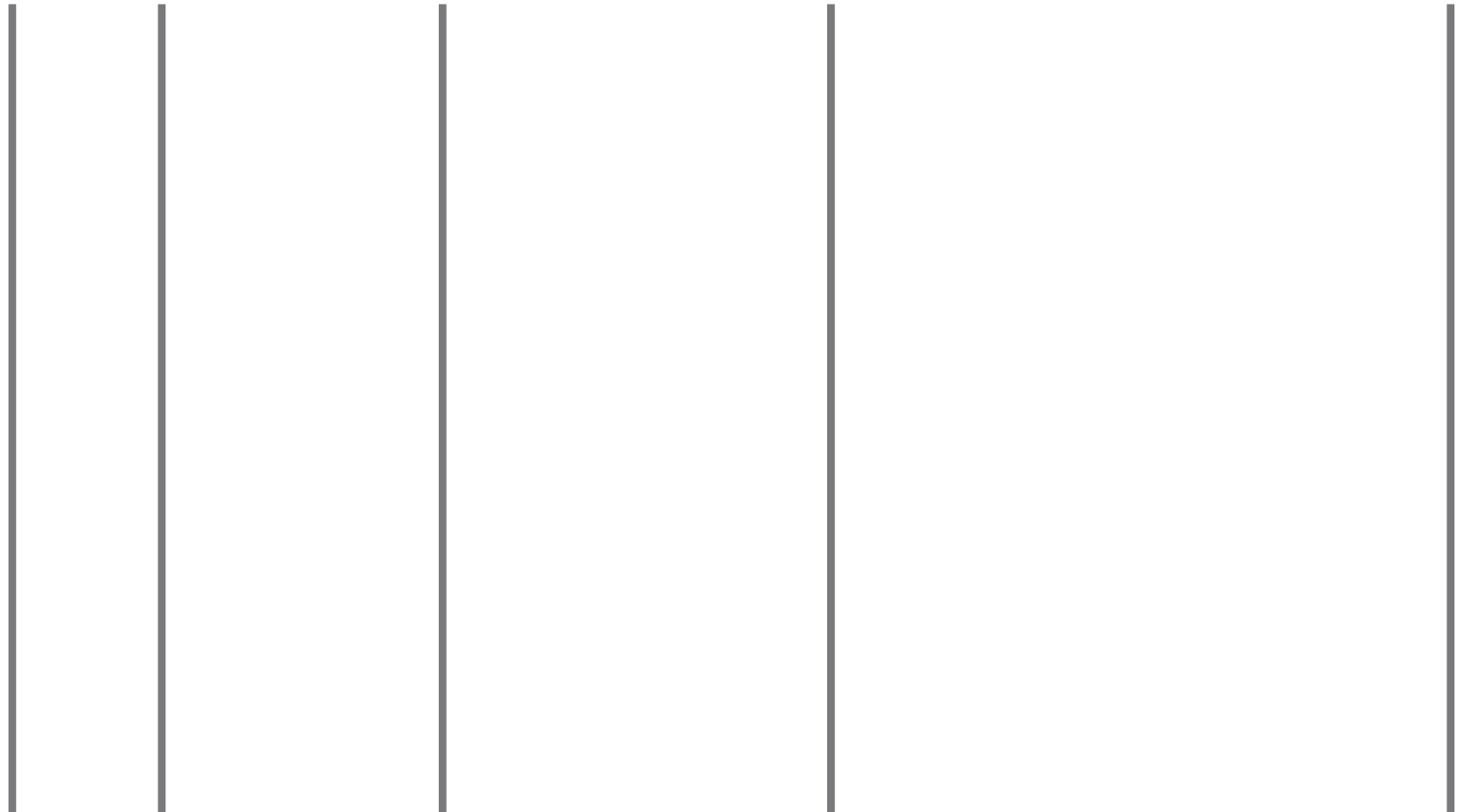
①

②

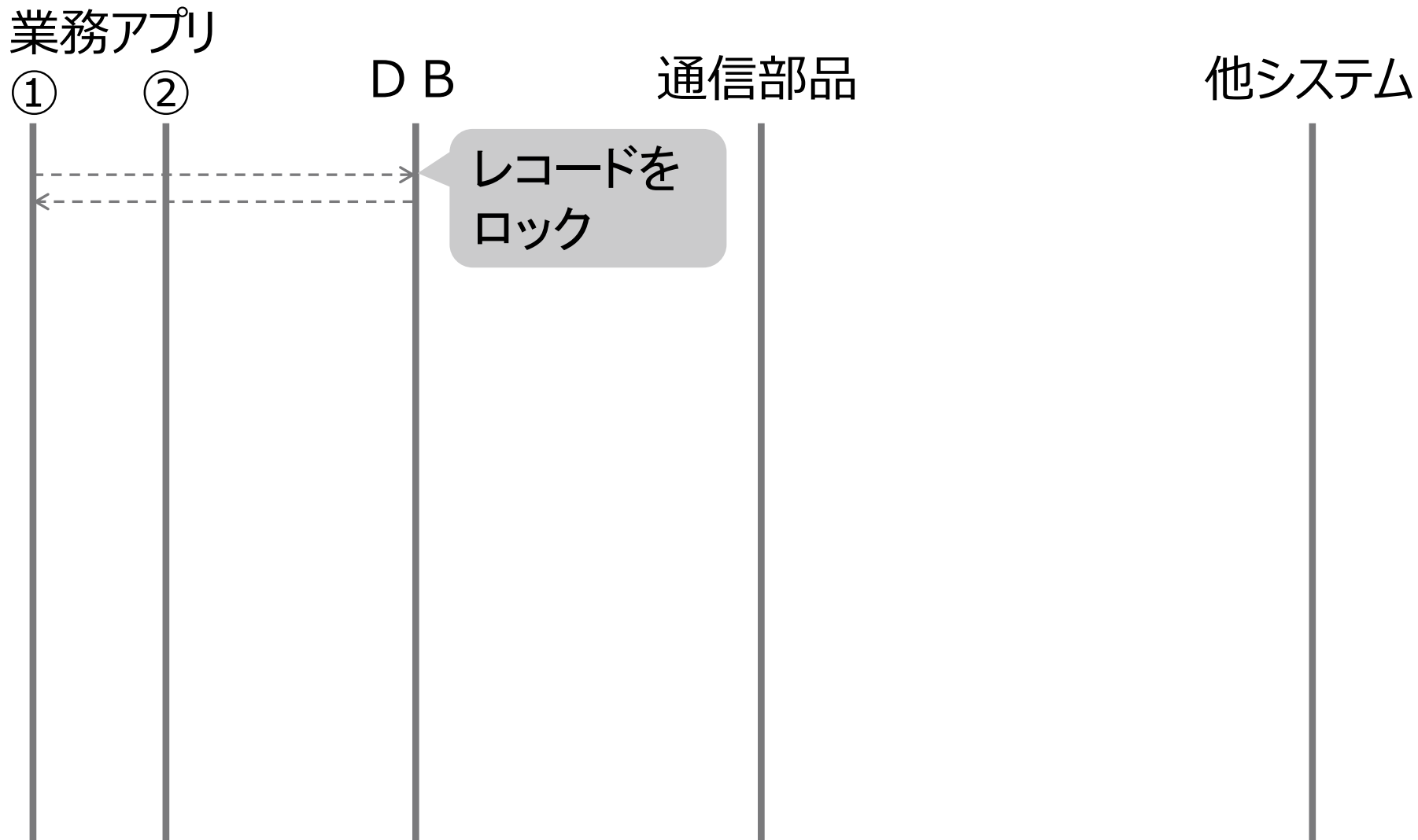
D B

通信部品

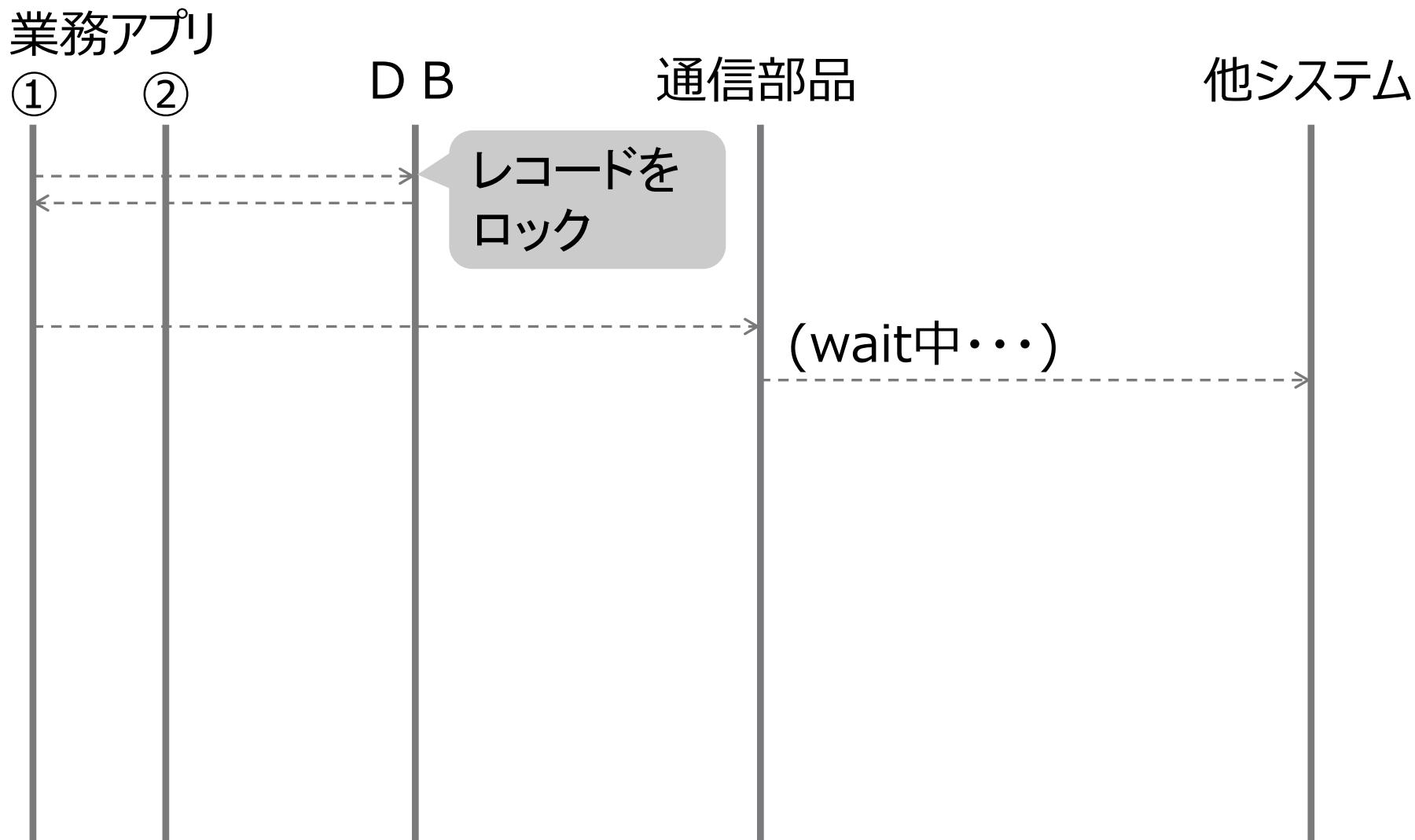
他システム



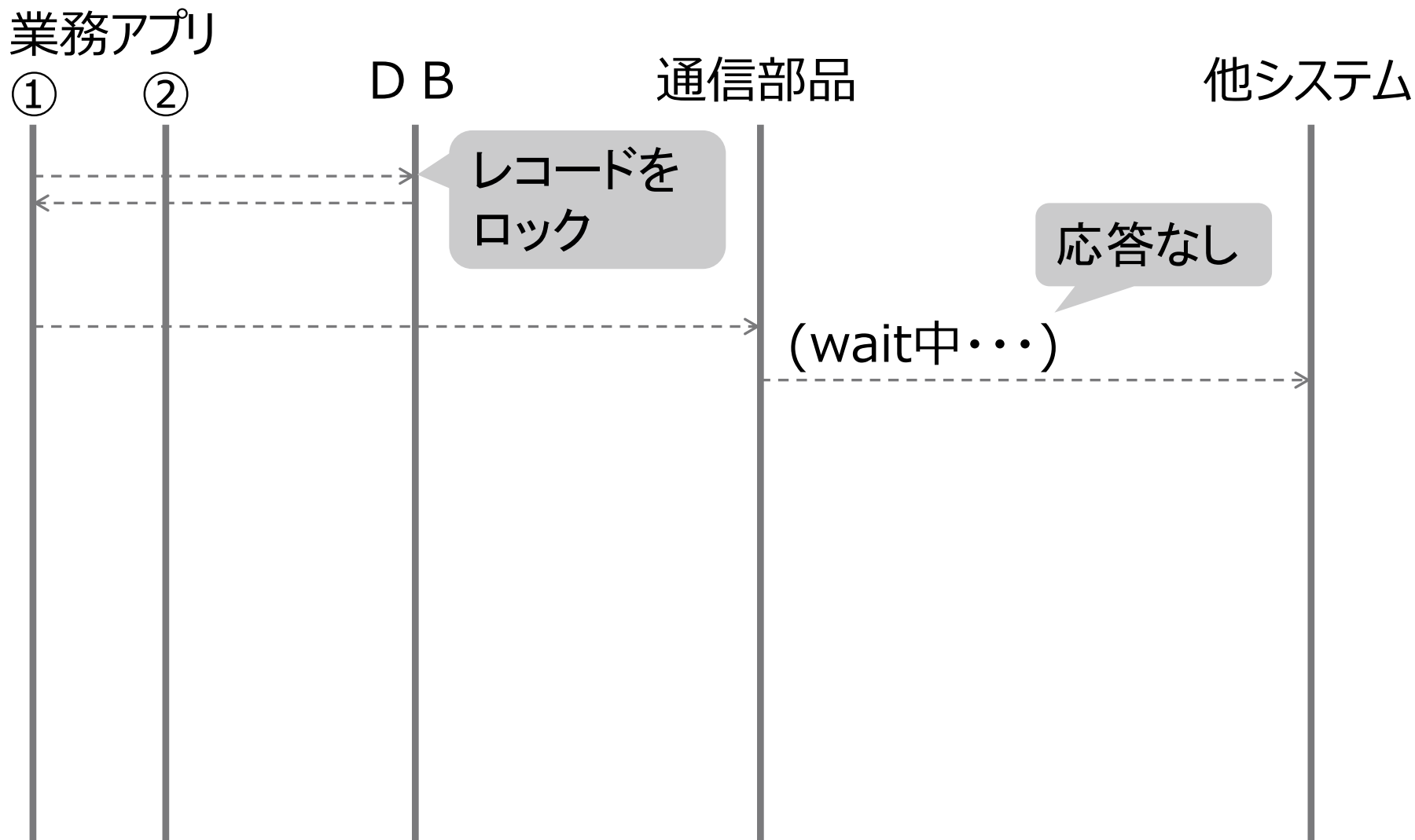
タイムアウトが無いとこうなる



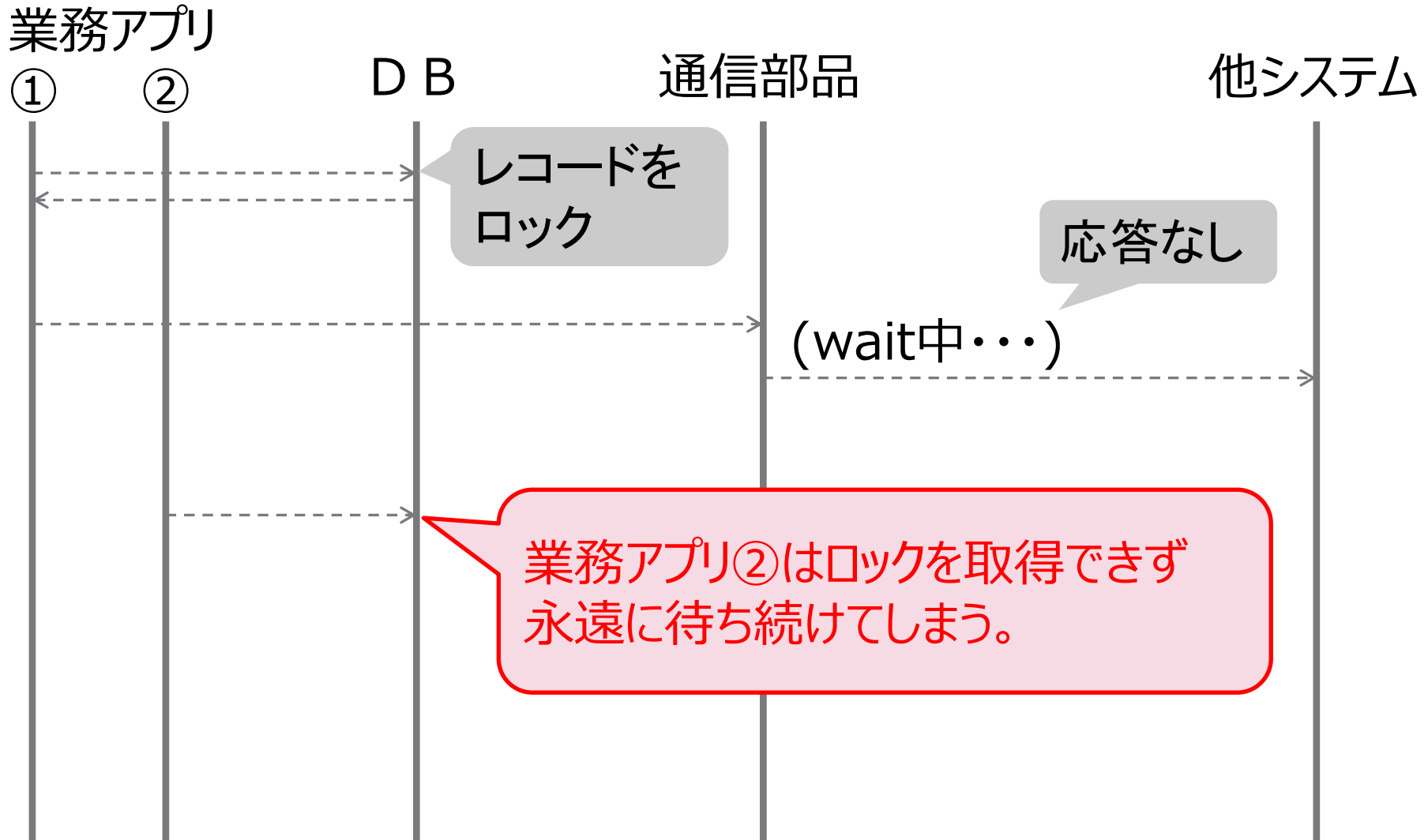
タイムアウトが無いとこうなる



タイムアウトが無いとこうなる

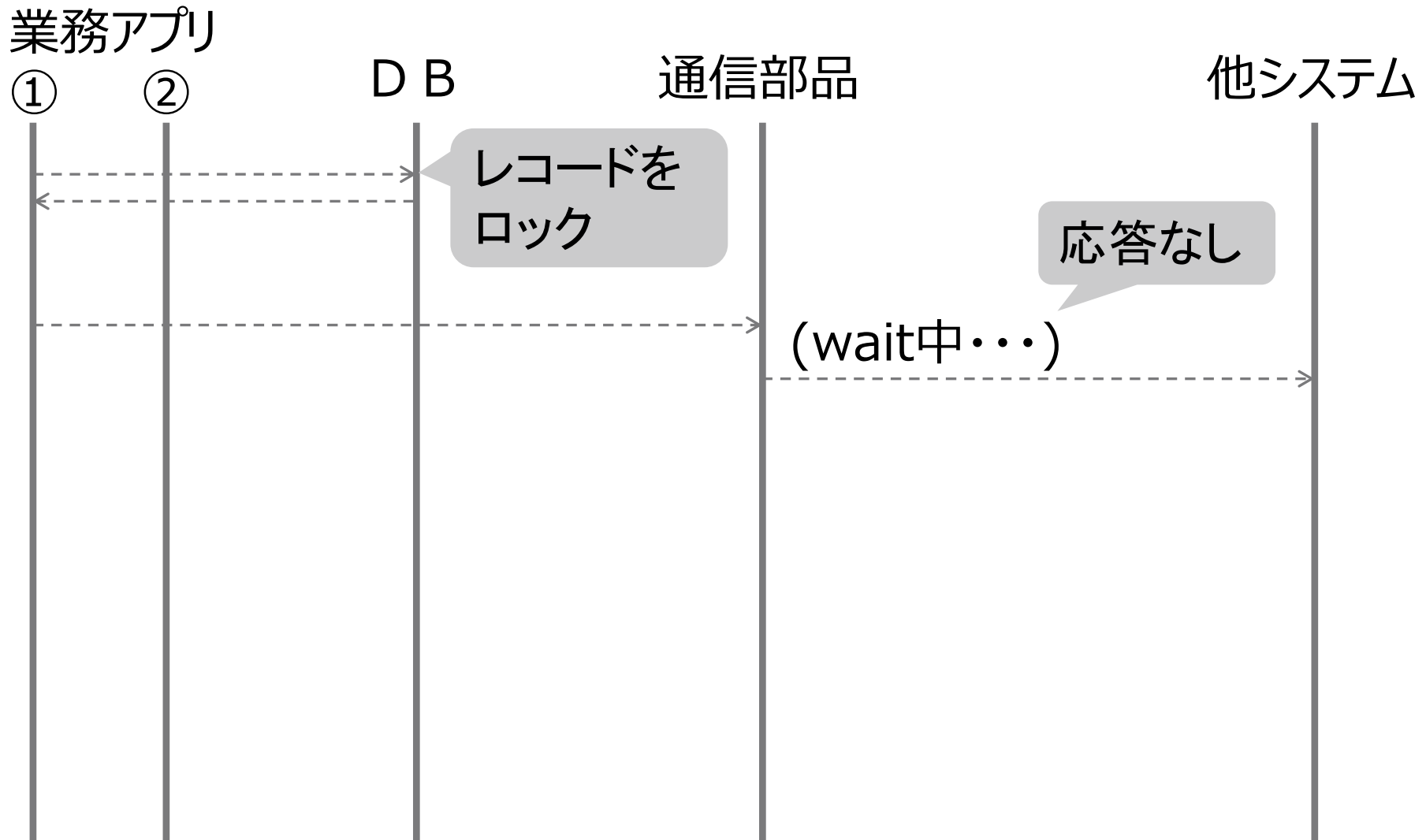


タイムアウトが無いとこうなる

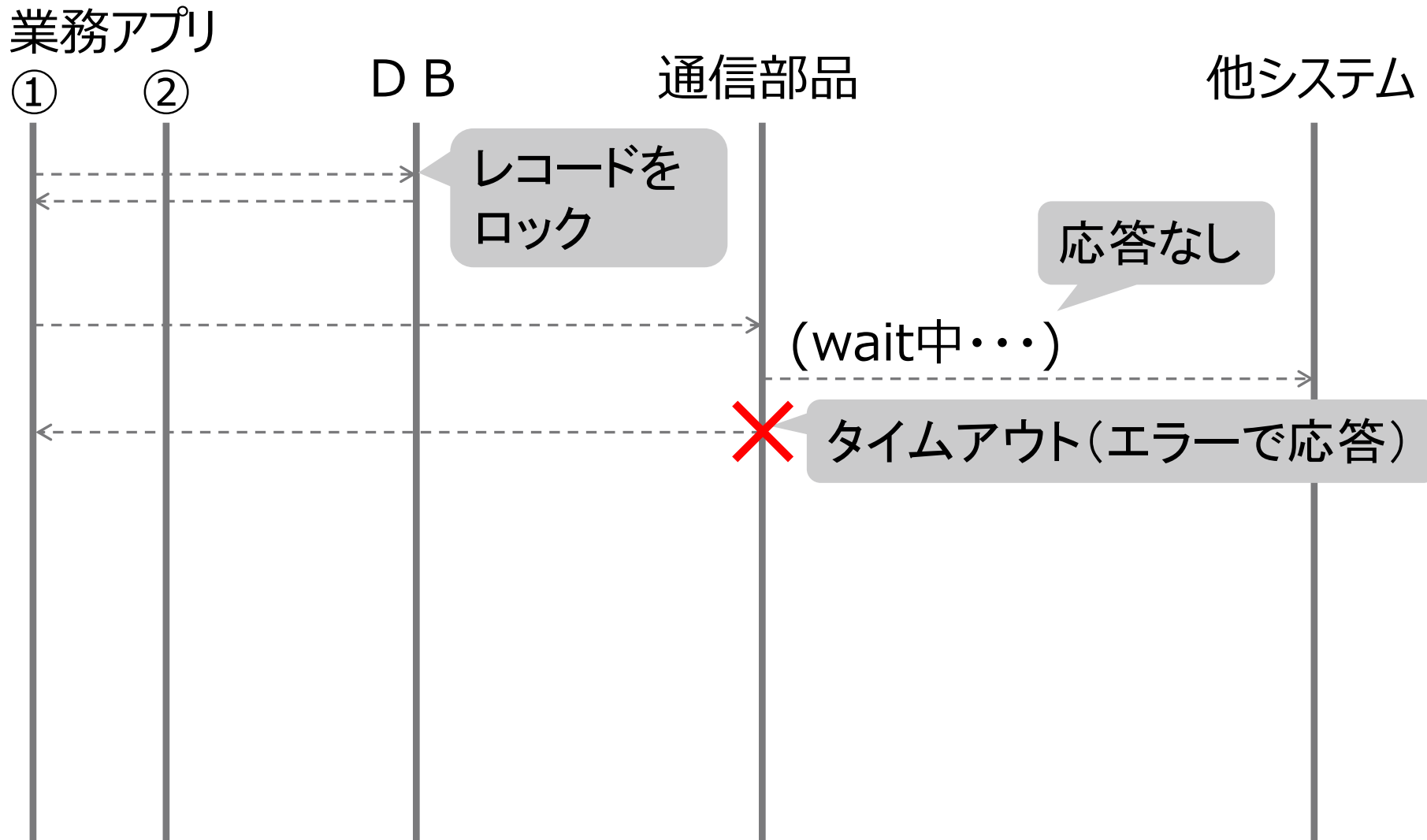


タイムアウトがあると・・・

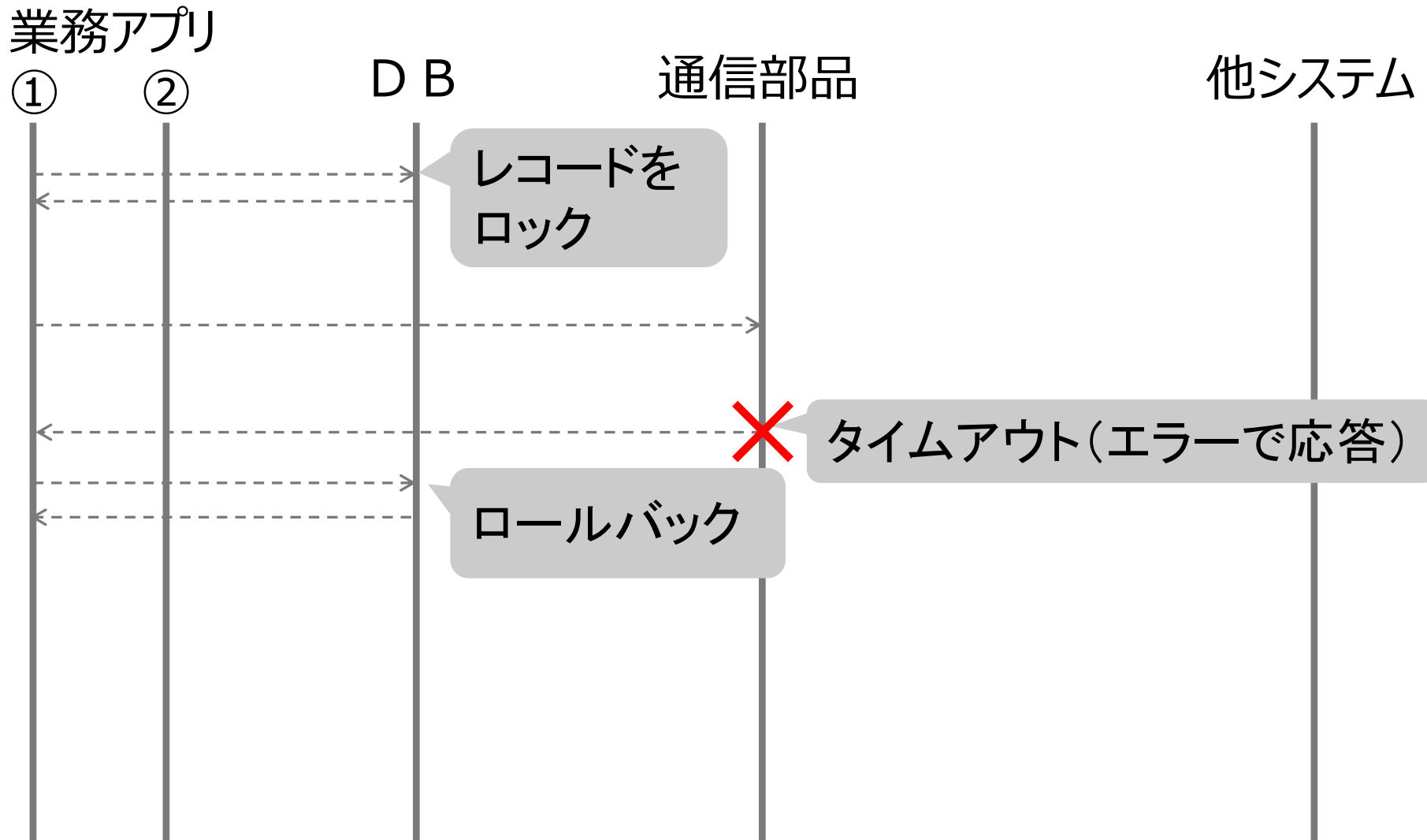
タイムアウトがあるとこうなる



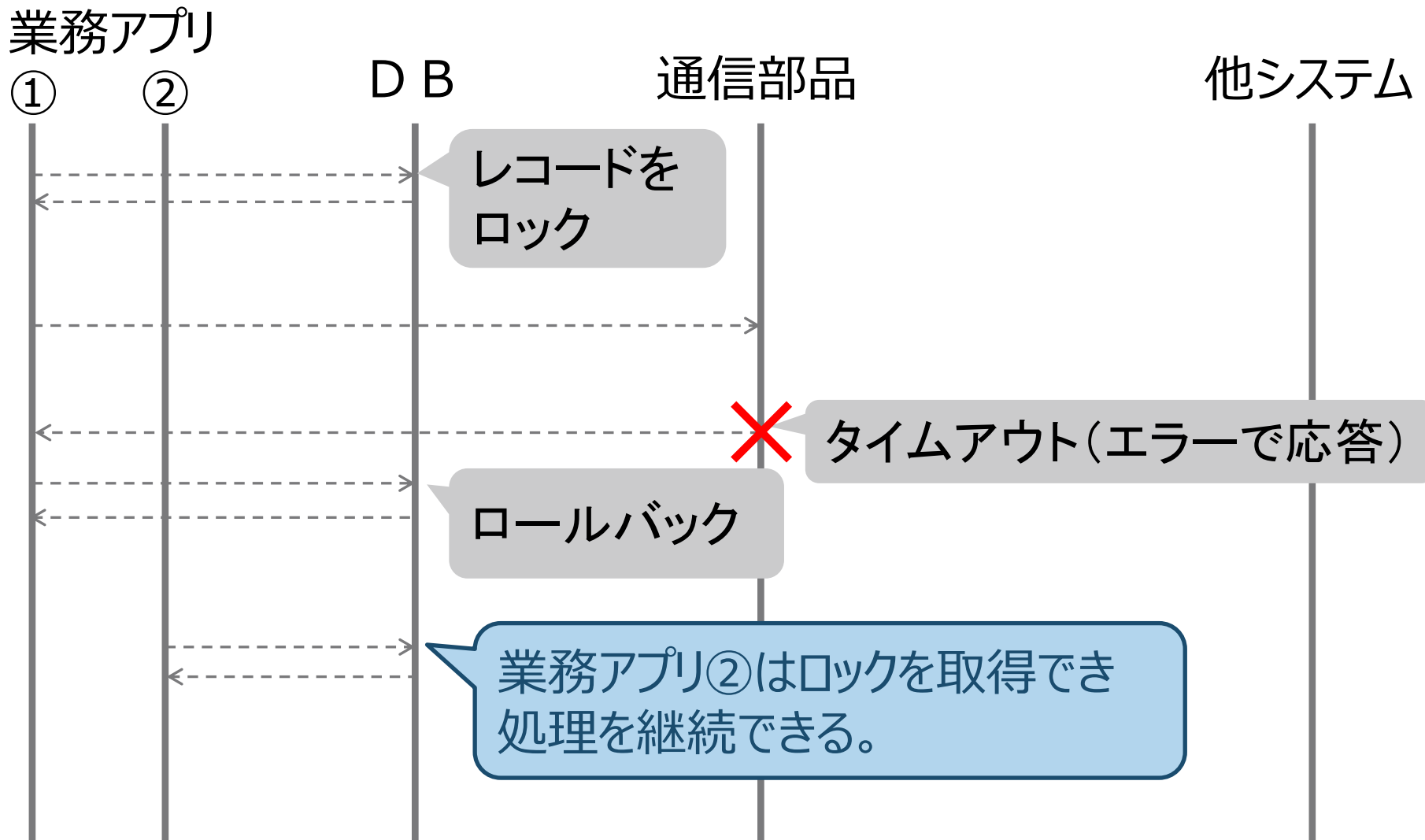
タイムアウトがあるようになる



タイムアウトがあるとこうなる

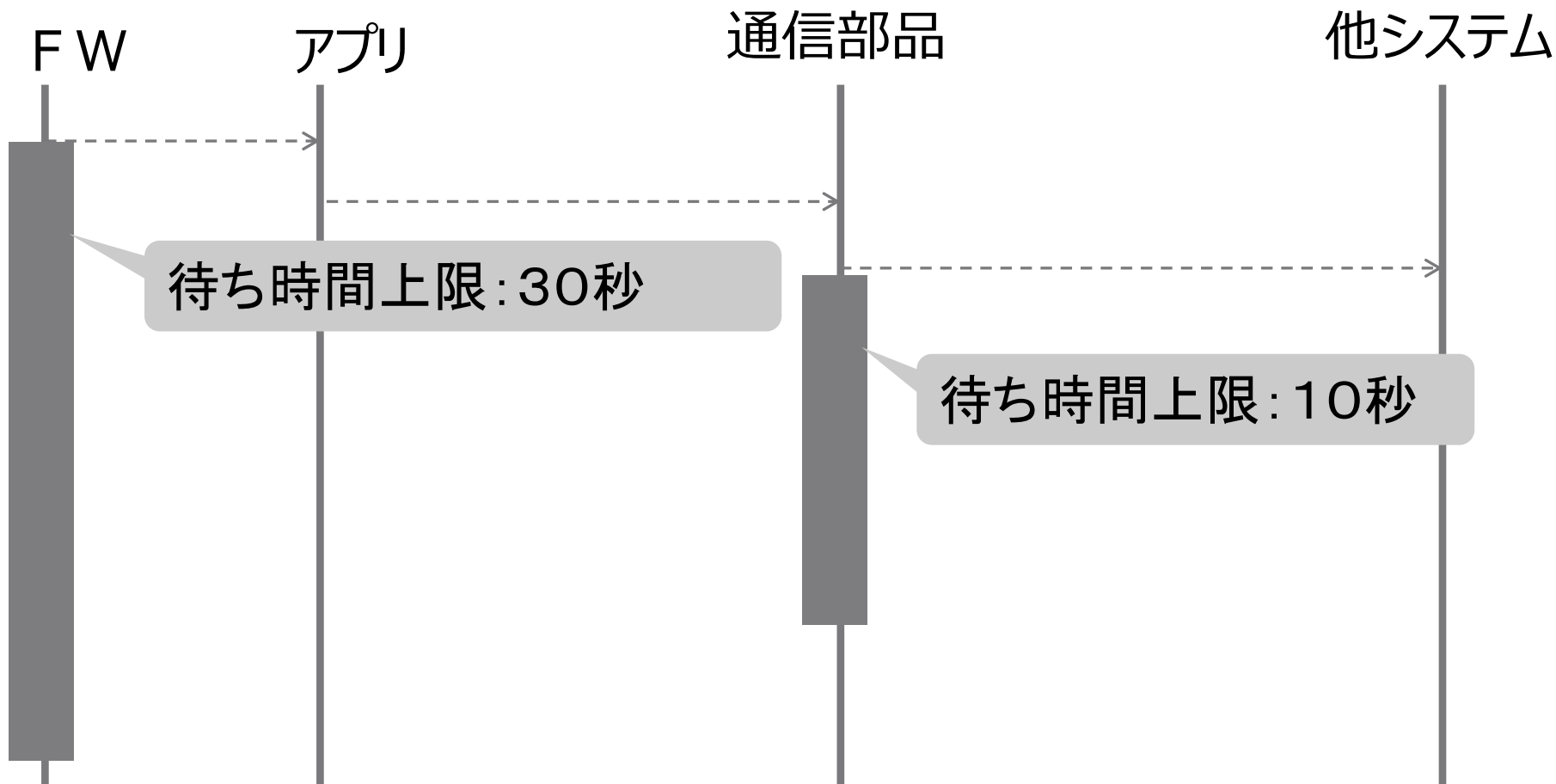


タイムアウトがあるとこうなる

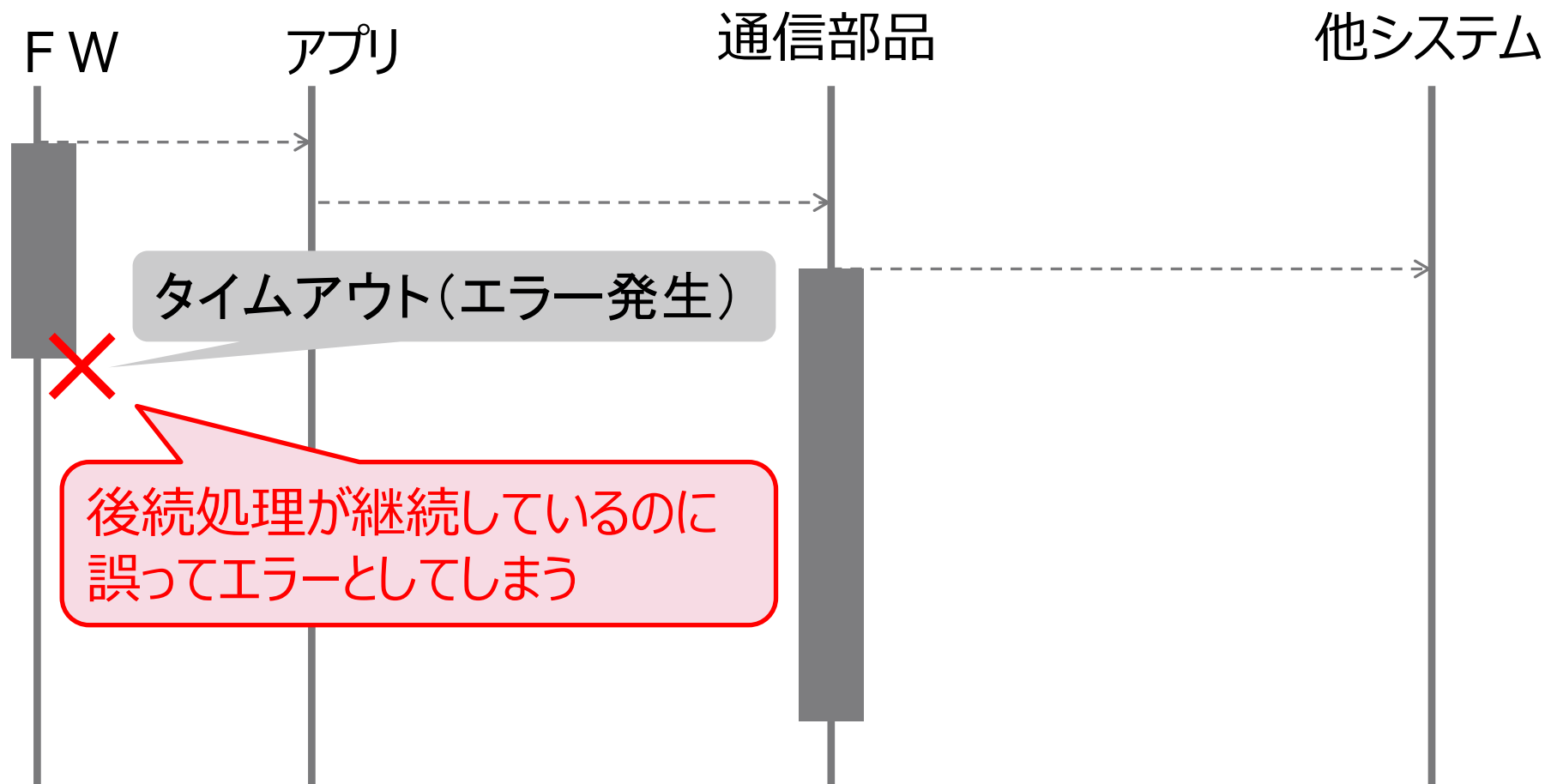


タイムアウト設定の基本

呼び出し元から離れるにつれて、待ち時間上限を短くしていく



そうしないと・・・



その他、可用性の実現方式について
特に重要となる知識は「セッション管理」の方式です。

※セッション管理について詳しく知りたい方は以下を参照ください。

Fintan > アプリケーションアーキテクチャの学習コンテンツ > Webアプリケーションのセッション管理
<https://fintan.jp/?p=8376>

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

様々な定義がありますが、「速さ」のことです。

画面オンライン：ボタンを押下してから、●秒で応答が返る。

バッチ：

- ・ 1 時間あたり●件のデータを処理する。(= スループット)
- ・ ●万件のデータを●時間で処理する。

など

どのような形式で性能要件を合意すべきでしょうか？

よくある例（N G）

「画面オンラインでは、平均 3 秒以内にユーザーに応答を返す。」

全ての処理で、平均 3 秒で応答できるでしょうか？

本当にお客様と約束できますか？

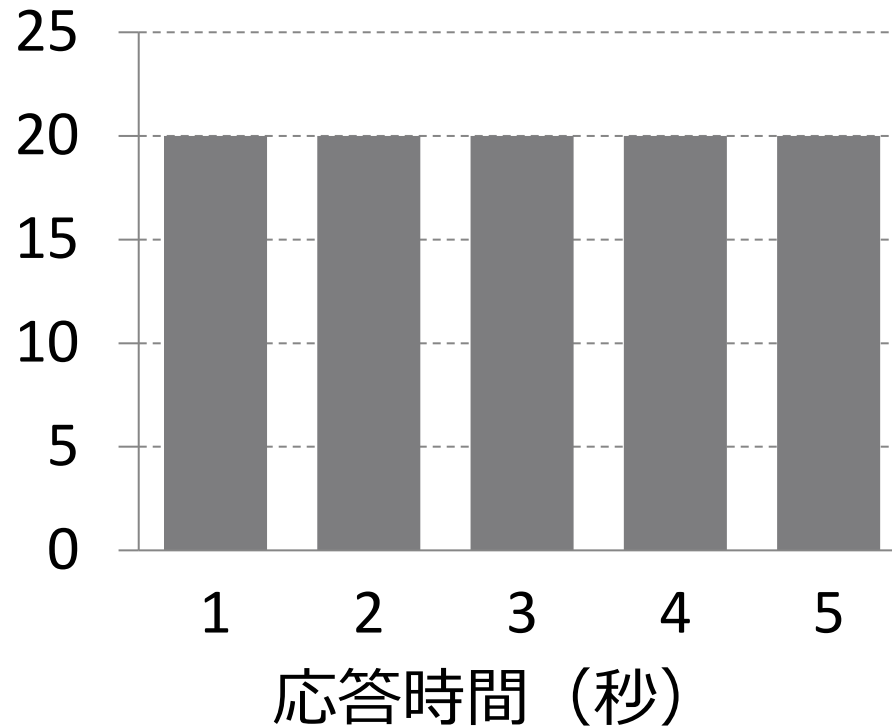
よくある例（N G – ただし前ページよりは良い）

「画面オンラインでは、
原則、照会は平均3秒以内、登録は平均5秒以内に
ユーザーに応答を返す。

ただし、帳票出力／一括処理／ファイルダウンロード／
大量データ処理など、上記指標の遵守が困難な業務処理に
ついては、個別に目標設定を行う。」

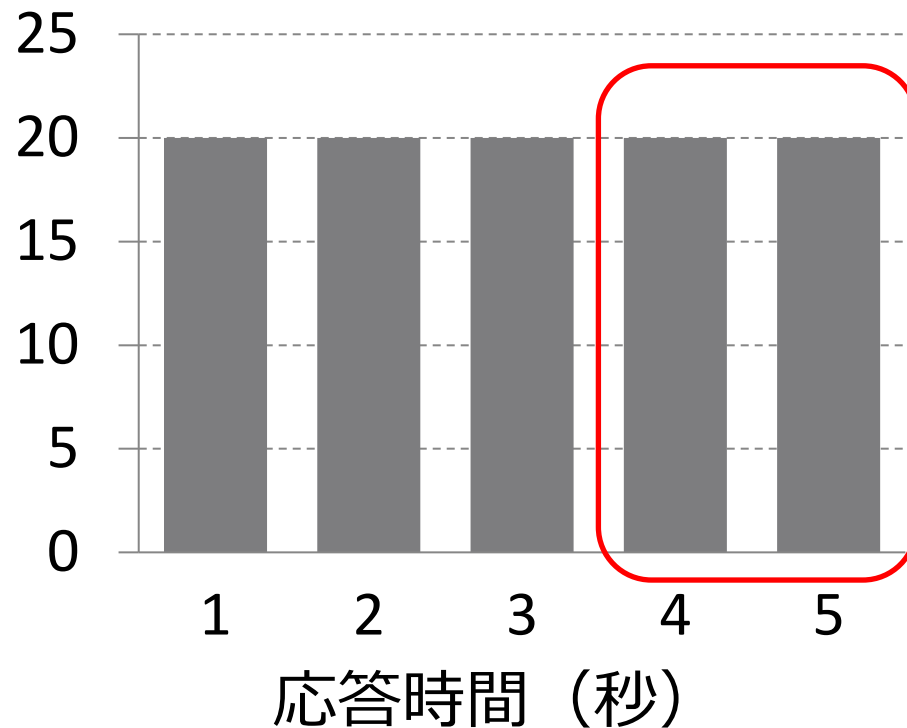
処理の「重さ」によって色分けをしている点は良いですが
ある問題をはらんでいます。

件数



応答時間の平均値を
求めてみましょう。

件数



応答時間の平均値は
「3 秒」です。

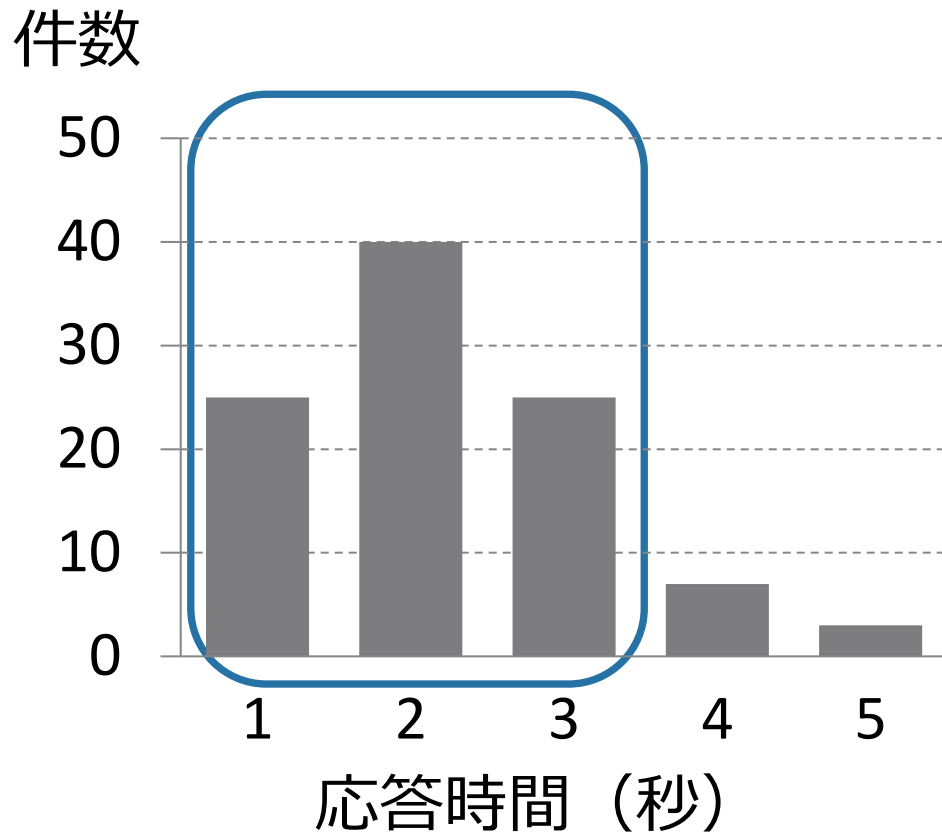
4 秒や 5 秒の件数が全体の
40%を占めています。

お客さまに納得いただけそう
でしょうか？

では、どうすれば良いのでしょうか？

 分布で表現して合意すべきです。

分布で表現する



このグラフは

「90%のリクエストで
3秒以内に応答する」

という要件を表します。

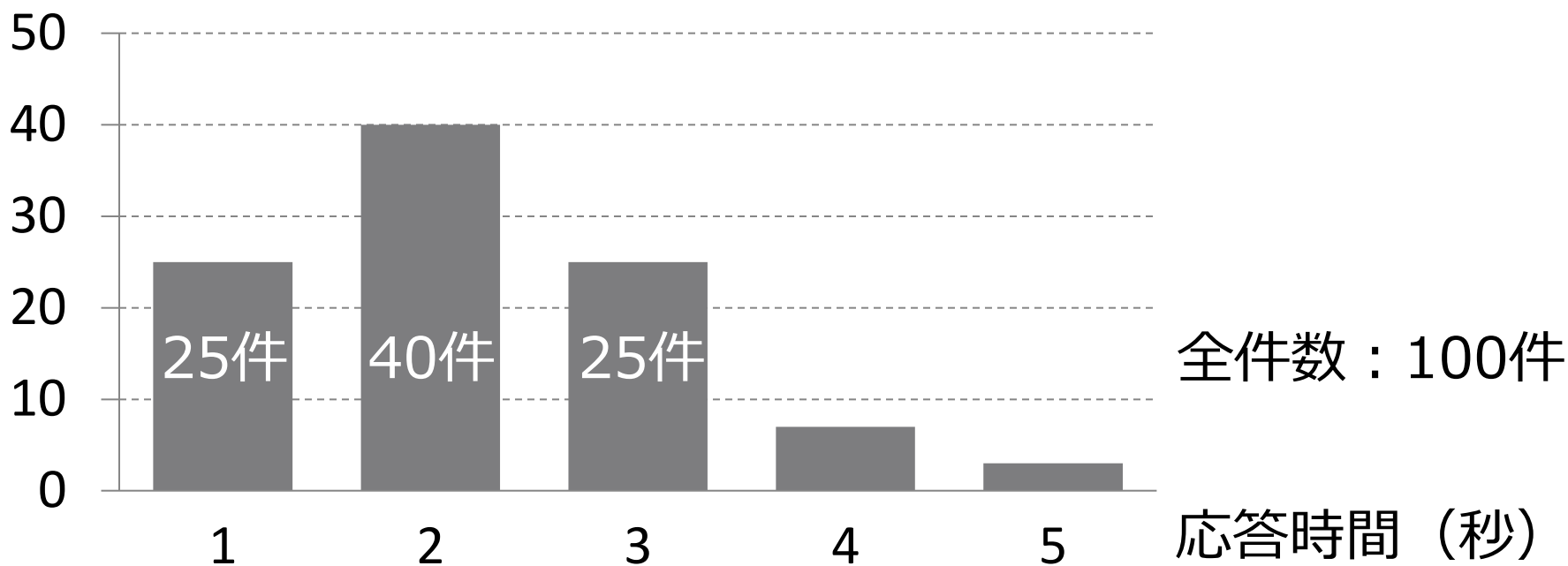
お客さまにとって納得しやすい
妥当な要件といえます。

※案件によって合意すべき値は
異なります。

現場では「90パーセンタイル 3秒以下」と表現されます。

●●パーセンタイルとは、最小値からその値までのデータ数が全体のデータ数の●●%となる「位置」を表します。

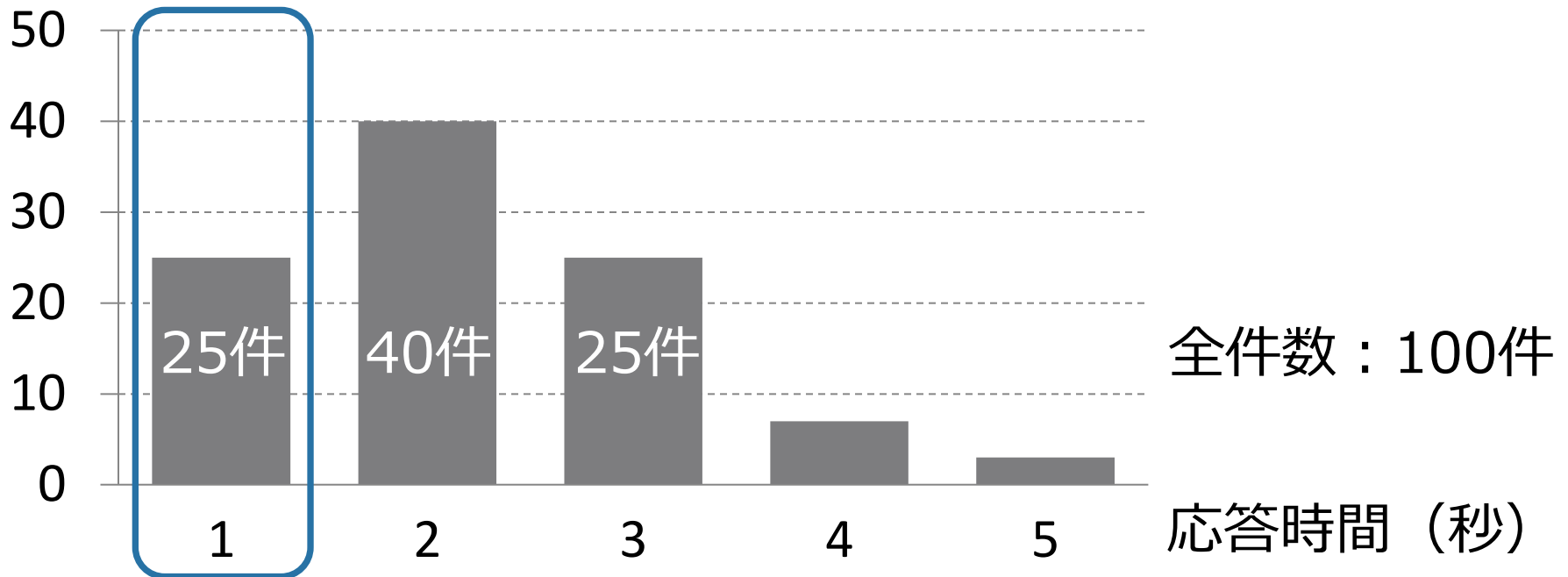
件数



応答時間がこのような分布の場合・・・

分布で表現する

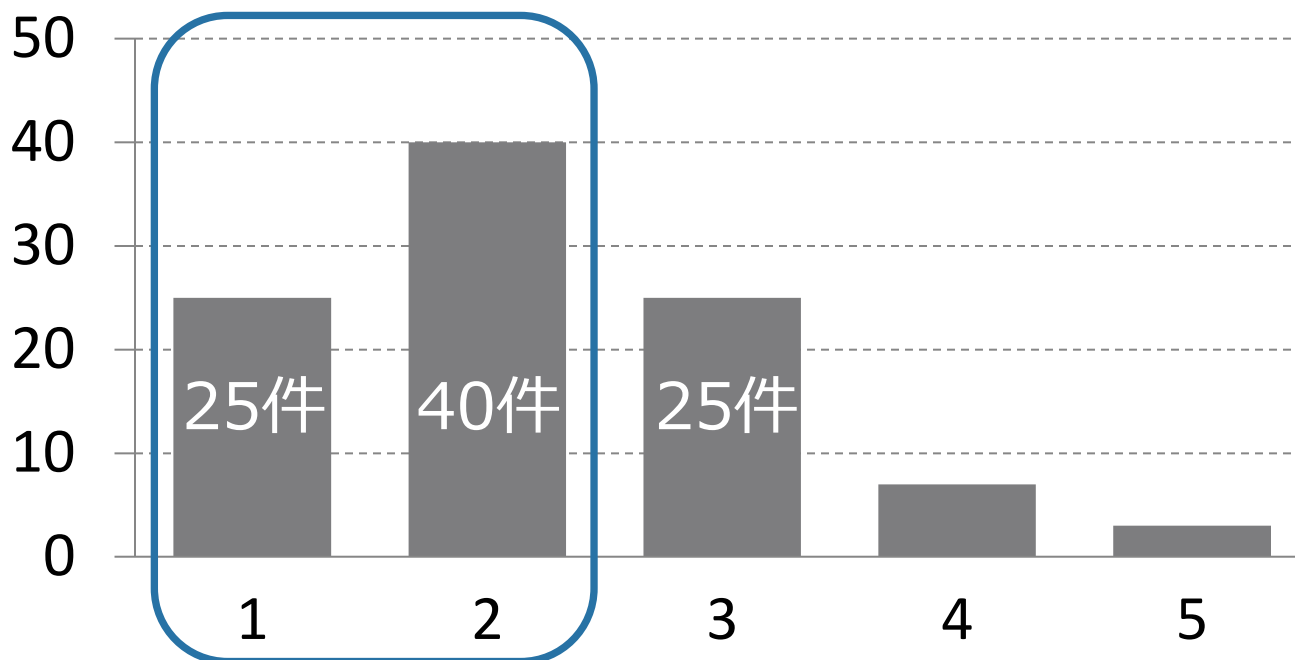
件数



25パーセンタイルとなるのは・・・ 1 秒

分布で表現する

件数



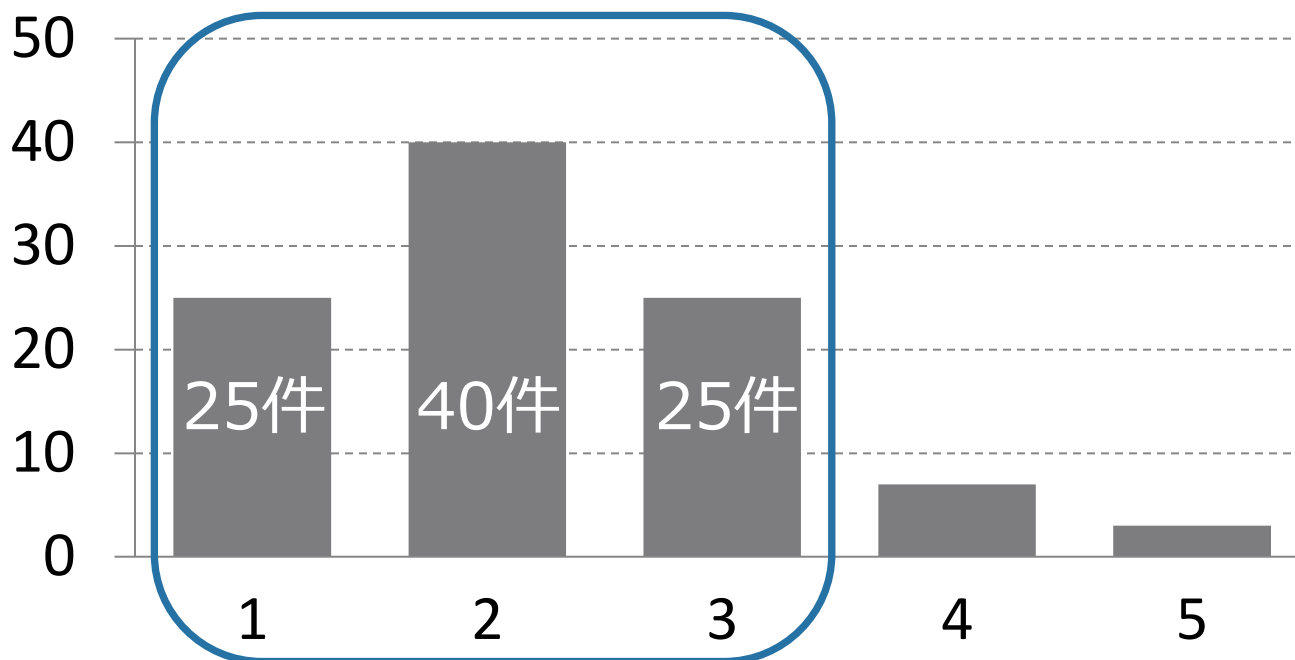
全件数：100件

応答時間 (秒)

65パーセンタイルとなるのは・・・ 2 秒

分布で表現する

件数



全件数：100件

応答時間 (秒)

90パーセンタイルとなるのは・・・ 3 秒

現場では多くの場合（慣例的に）
「90パーセンタイル」の位置をどこで合意するのか？
について、お客様と議論のうえ合意します。

その結果
「90パーセンタイル 3秒以下」などの要件が定まります。

このように合意した性能要件については
要件を満たしているかを「性能テスト」で確認します。

JMeterなどのテストツールを使用して
一定数のリクエストをサーバに送信することで
要件どおりの分布となるかを確認します。

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

業務処理量を明確にしましょう

業務処理量とは？

お客様の業務で扱う「要求」や「データ」の量

画面オンラインだと・・・

ユーザー数、同時ログインユーザー数、
単位時間あたりのトランザクション数（リクエスト数）、
参照するデータの件数、
同時実行トランザクション数、など

バッチだと・・・

入力ファイルなどのデータ量、バッチ本数、など

業務処理量を明確にしてどうするのでしょうか？

その業務処理量を捌けるように、システムを作りこんでいきます。

例えば、ミドルウェア（MW）の設定値を決めます。

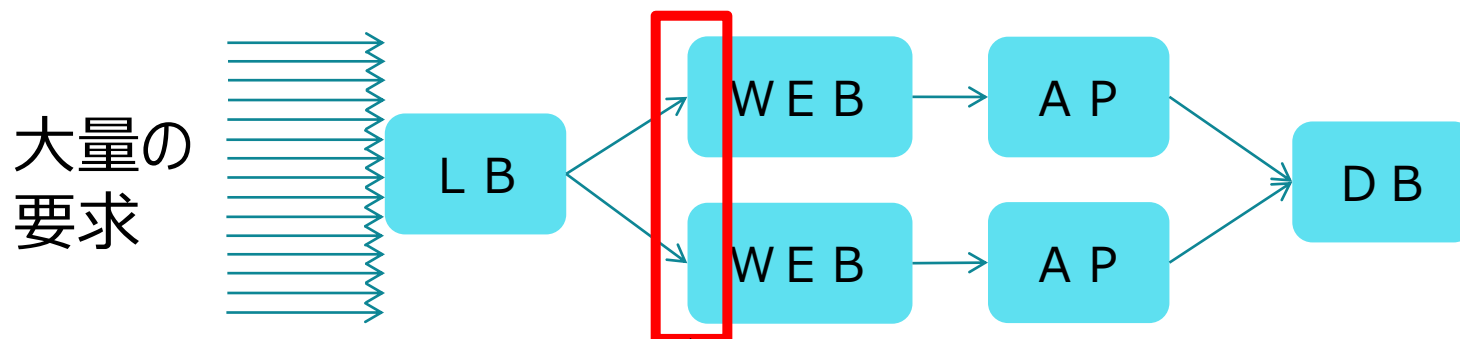
今回は、画面オンラインの「単位時間あたりのトランザクション数」をインプットに、MWを設定する流れについて解説します。

多くの場合・・・

ハードウェア、MWの選定、MWの設定 → インフラ担当者

MWの設定（一部） → アプリ担当者（アーキテクト）

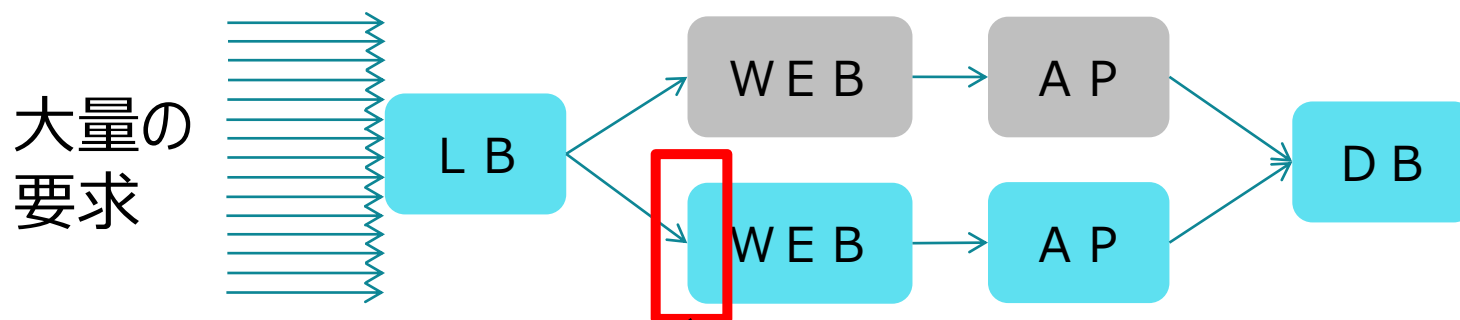
MWの設定の多くは、インフラ担当者が担当します。
しかし、アプリ担当者（アーキテクト）でなければ設定値を導出できない項目も存在します。
アプリ担当者だからといって、MWの設定に無関係という訳ではないのです。



今回は、ここでどれだけの要求数を受け入れるか？
という上限を設定します。（典型的にはApacheの設定）

設定値が大きすぎると
W E Bサーバ以降が過負荷となってしまいます。

設定値が小さすぎると
性能要件を満たせなくなってしまいます。



片系のみの稼働でも（縮退稼働などと言います）、
ピーク時の要求数に耐えられるようにすべきです。

この前提を置いて、設定を施していきます。

適切な設定値を導き出すために
「最大同時実行リクエスト数」を算出していきます。
算出方法は以下の通りです。

最大同時実行リクエスト数

$$\begin{aligned} &= \text{ピーク時の T P S (Transaction Per Second)} \\ &\quad \times \text{1 P V あたりの平均処理時間} \\ &\quad \times \text{余裕率} \end{aligned}$$

ピーク時の T P S

T P S = Transaction Per Second
1 秒あたりのトランザクション数（要求数）

この業務量はお客様でなければわかりませんから、
お客様に聞きましょう。
（お客様の理解できる表現で、聞く必要があります。）

例えば、お客様から「**4 件**」との回答を得られたとします。

1 P Vあたりの平均処理時間

照会処理の数：登録処理の数の比率を仮定します。
例えば、4：1 とします。

処理全体を、処理時間が大きく異なる
塊に分け、その比率を仮定する。

また、パーセンタイルで定めた秒数は、平均値よりも遅いはずです。



このため、パーセンタイルで定めた秒数を用いて平均処理時間を計算すれば、「悪い方に倒した」余裕のある結果を得ることができます。

「平均値」を算出するのは難しいため、パーセンタイルの秒数を「代用」する。

例えば、90パーセンタイルが「照会：3秒」「登録：5秒」の場合、それぞれの秒数を用いて、1 P Vあたりの平均処理時間を計算しましょう。

以上を踏まえると、1 P Vあたりの平均処理時間は

$$\begin{aligned} & (\text{照会処理 } 3 \text{ 秒} \times 4 \text{ 件} + \text{登録処理 } 5 \text{ 秒} \times 1 \text{ 件}) \div 5 \text{ 件} \\ & = \underline{\underline{3.4 \text{ 秒}}} \text{ となります。} \end{aligned}$$

今回の例では・・・

最大同時実行リクエスト数

= ピーク時のTPS

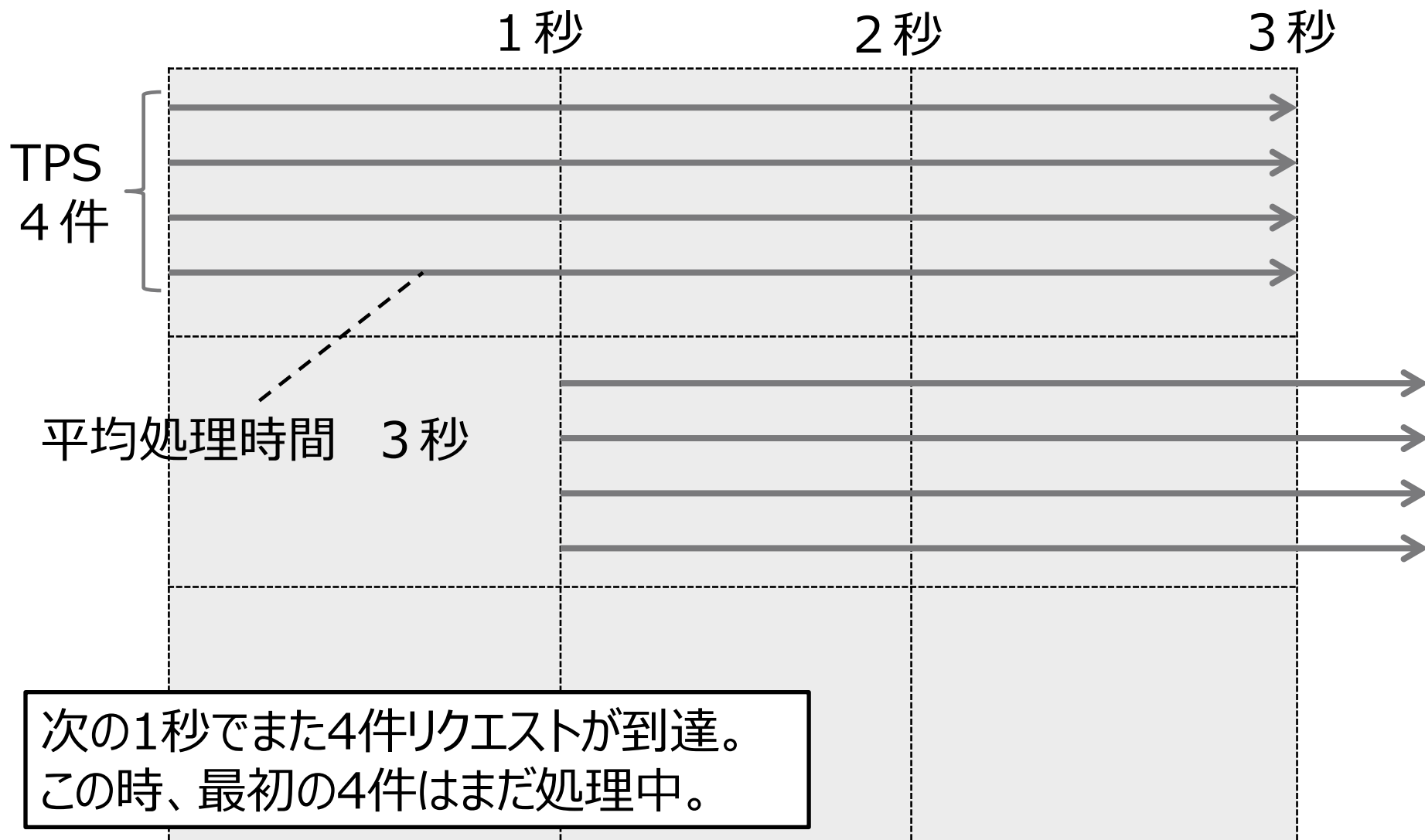
× 1PVあたりの平均処理時間 × 余裕率

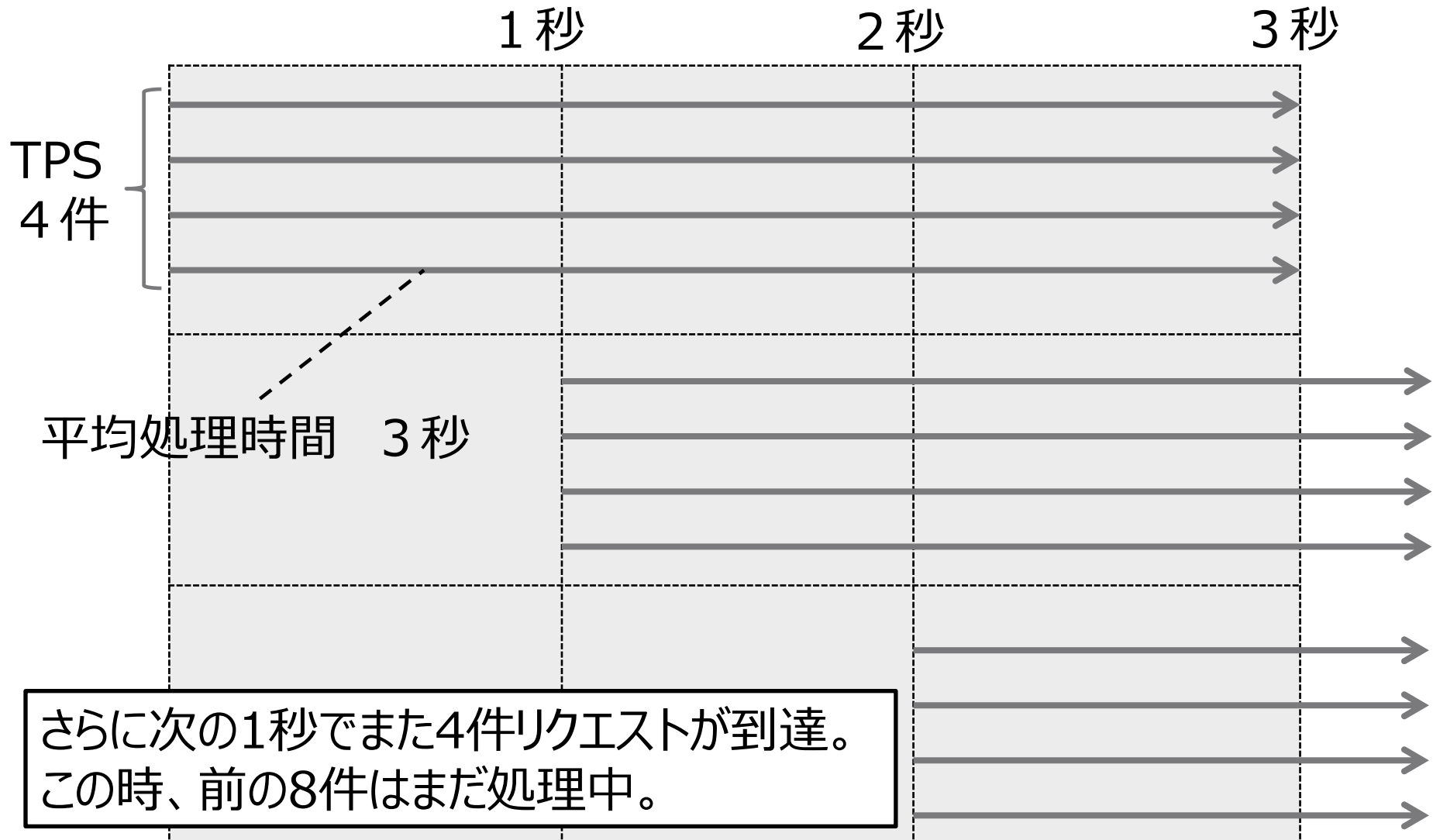
= 4件

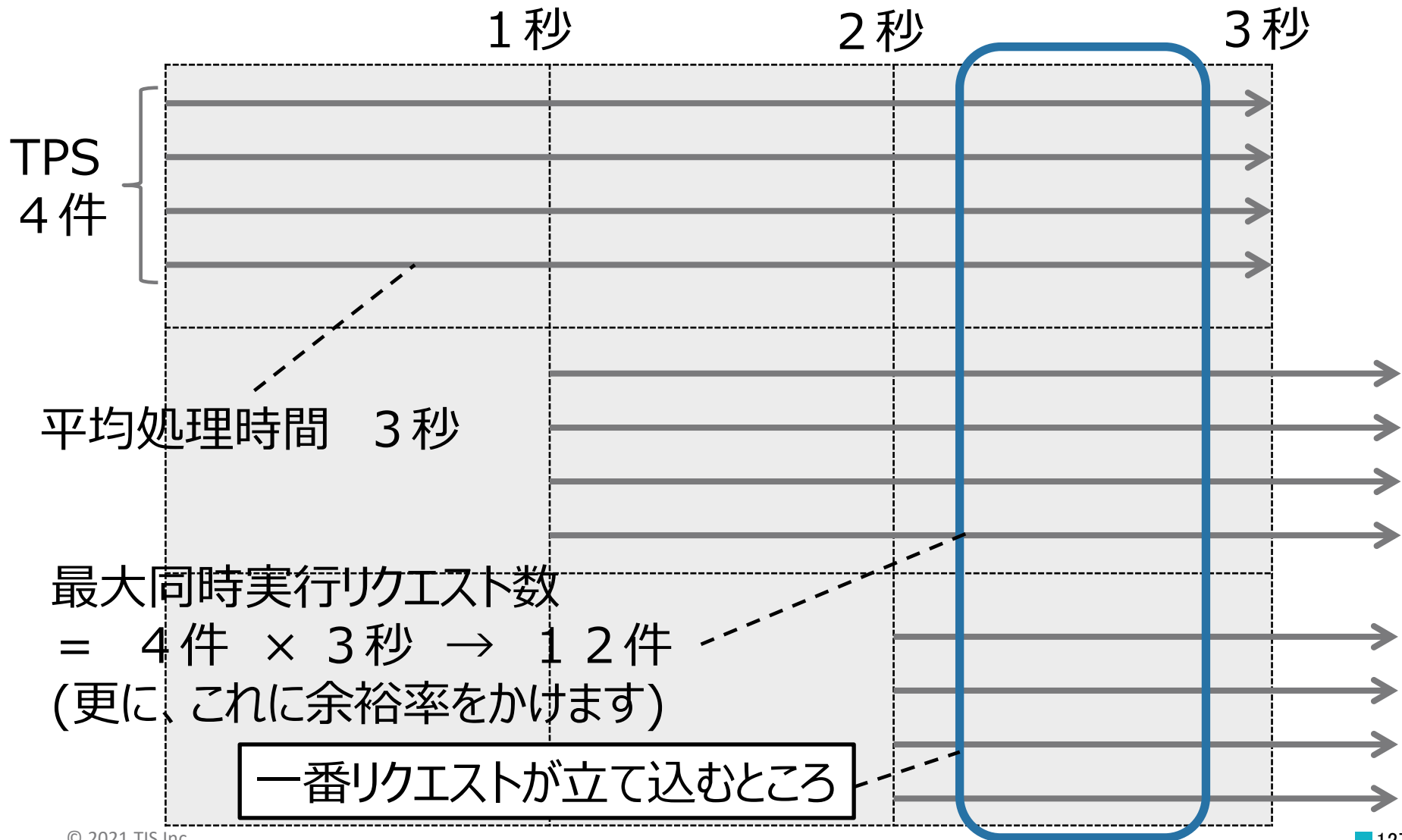
× 3.4秒 × 1.2 (20%を余裕分とします)

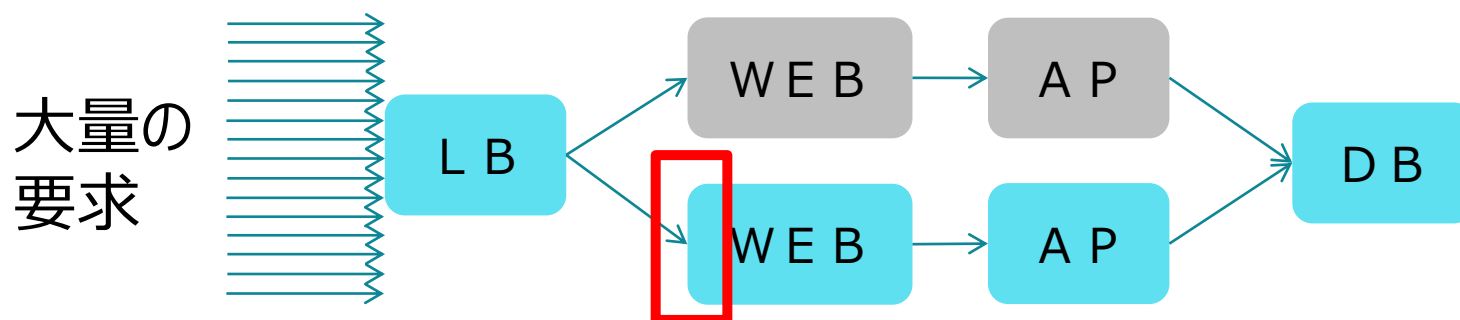
≒ 16件









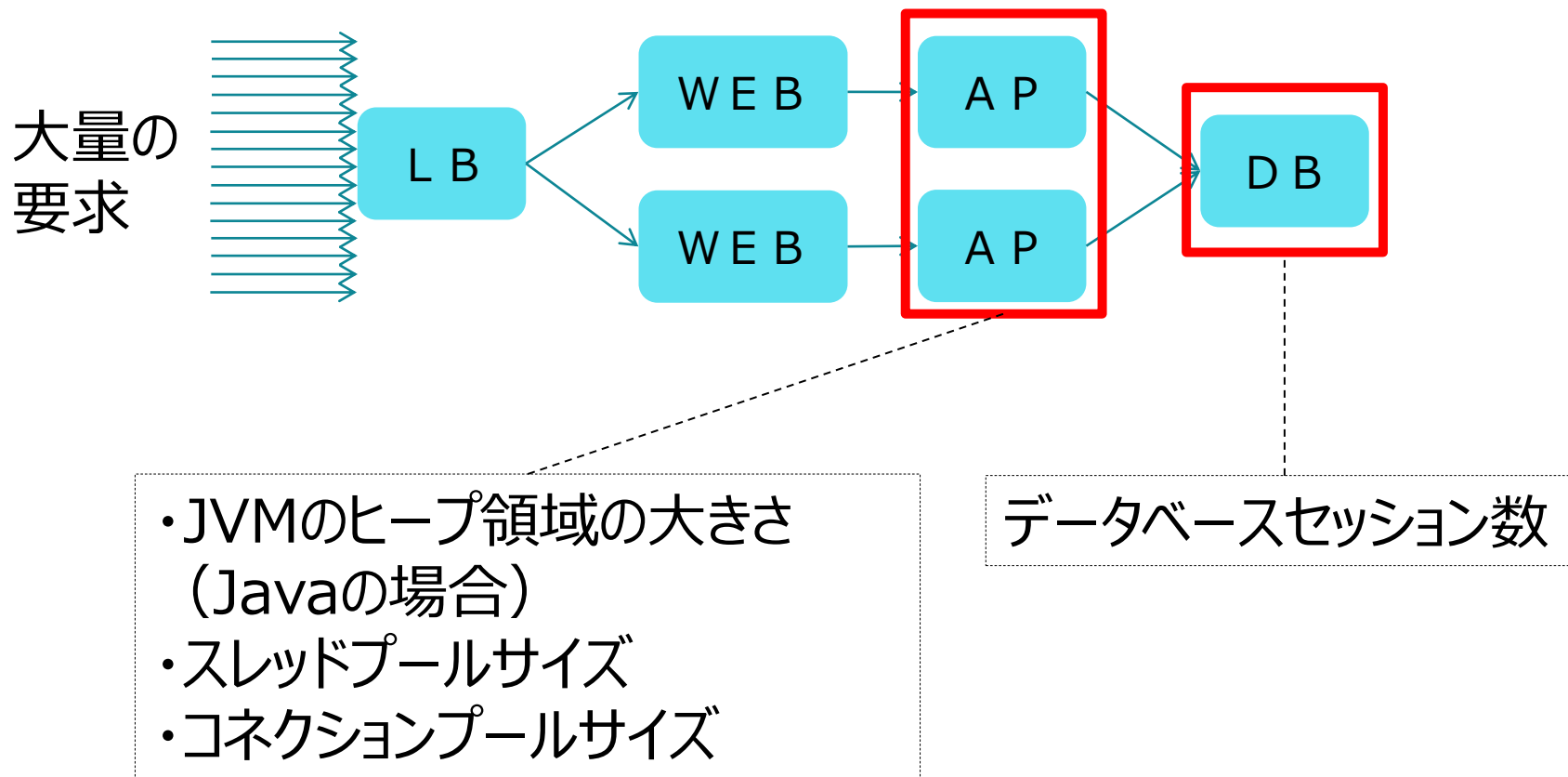


今回の例では、少なくとも同時に16件以上を受け入れ可能な設定を施します。

典型的にはApache HTTP Serverの「MaxClients」ディレクティブを、16以上に設定します。

※サーバのCPUコア数の観点から、上記の件数を受け入れ可能か否か、についても確認が必要です。

MW設定の例（その他）



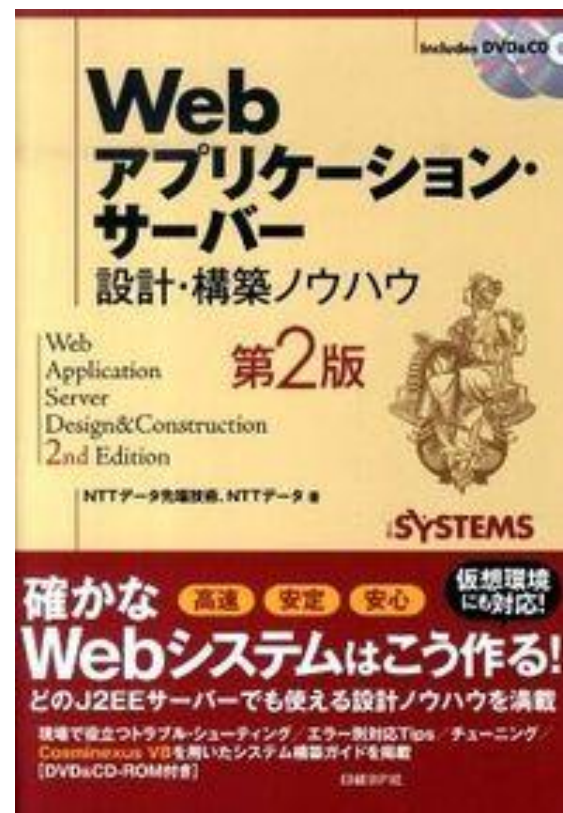
上記は一例です。

これらのような設定値を的確に算出して設定することで、お客様の要求を満たすことができます。

MW設定の例（その他）

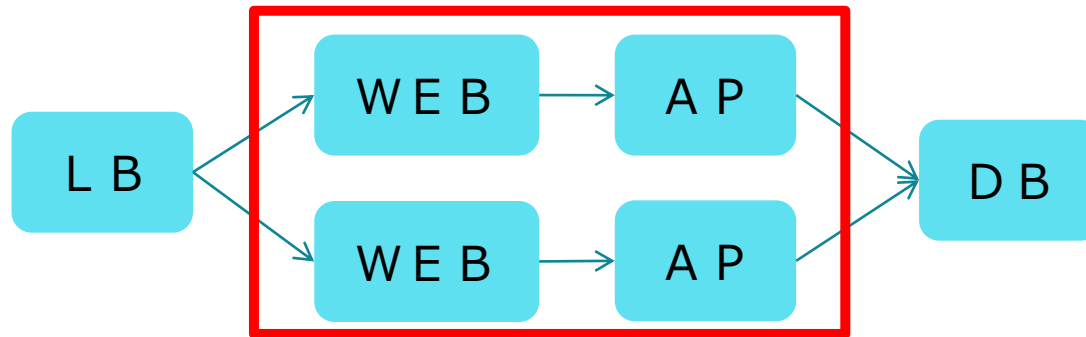
その他の設定値も含めた、MW設定の設計については、以下の書籍が参考になります。

Webアプリケーション・サーバー
設計・構築ノウハウ 第2版
日経BP
NTTデータ先端技術、NTTデータ 著
ISBN 978-4-8222-2994-8

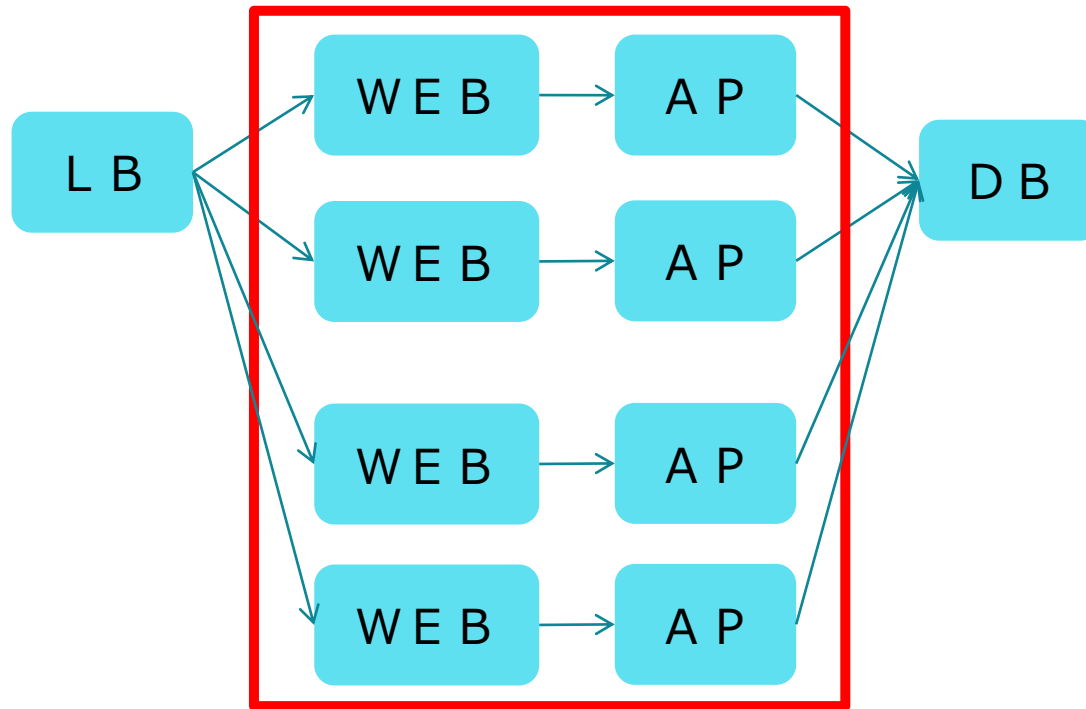


<https://www.hanmoto.com/bd/isbn/9784822229948>
2021年12月10日11時の最新情報を取得

- インフラについての基礎知識
- 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
- 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）



WEB/APサーバを増強する場合はどうするのでしょうか。



スケールアウトが基本です。（＝ 台数／系列数を増やします）

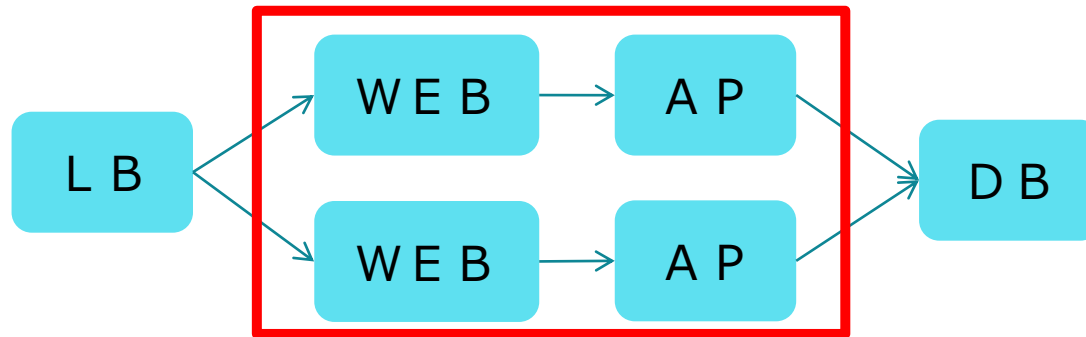
なぜかというと

工夫すれば、WEB/APサーバに状態（データなど）を持たせないように、アプリを構築できるからです。

工夫の例)

- ・セッションをD Bで管理する
- ・設定値を設定ファイルではなく、環境変数で管理する

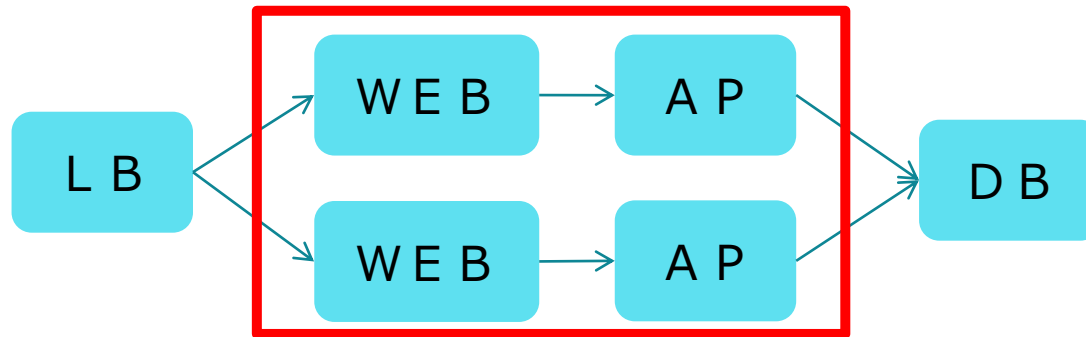
その他、詳しくは「Twelve Factor App」で調べてみましょう。



ただし、WEB/APサーバのスケールアウトには注意が必要です。

- ・DBサーバのCPU・メモリに余裕が無かったり
- ・アプリのDBアクセスが非効率だと

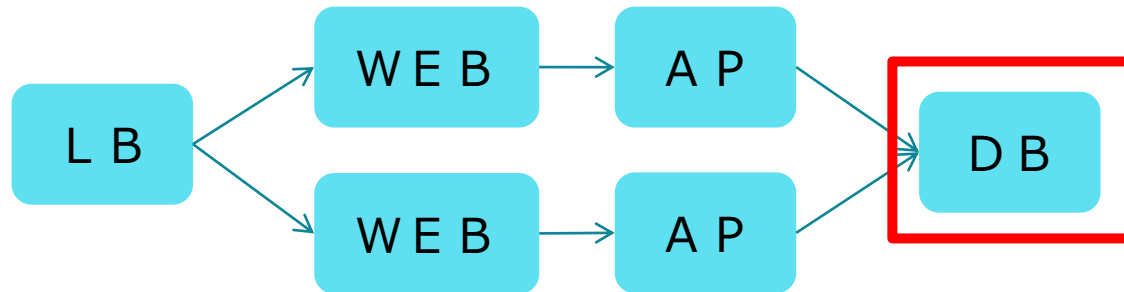
次はDBサーバがボトルネックになってしまいます。



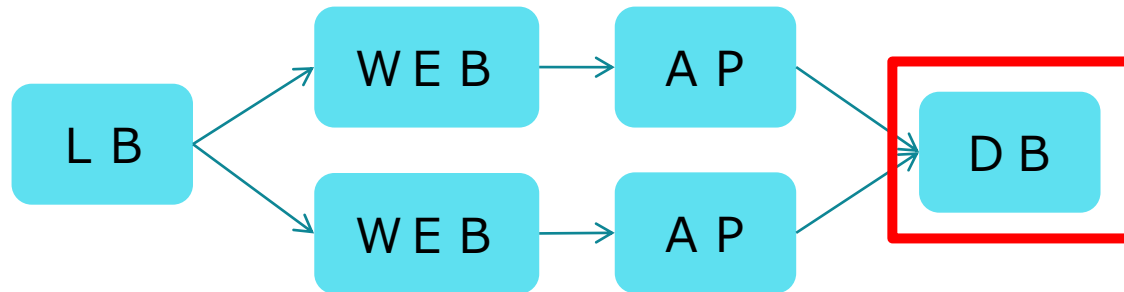
WEB/APサーバの負荷状況といった局所的な情報だけではなく

DBサーバも含めたサーバ構成 全体を見回して

スケールアウトの是非を判断しましょう。



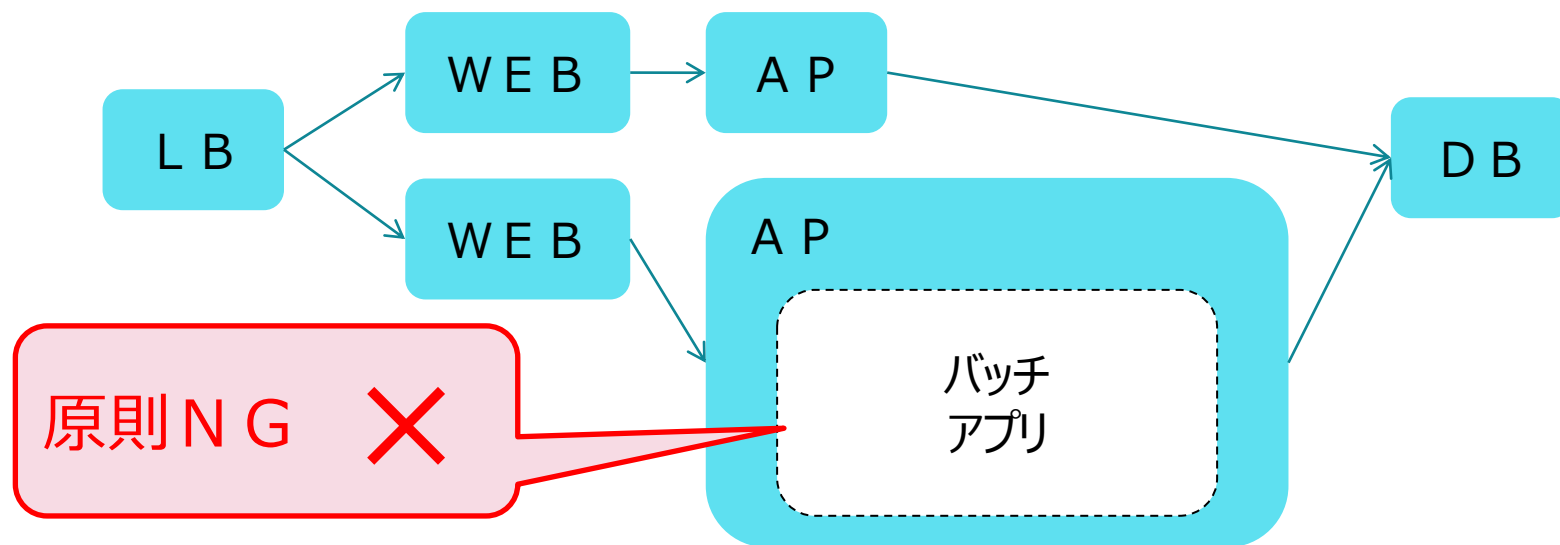
DBサーバの場合はどうでしょうか。



どうしても状態（データ）を持ってしまうため
スケールアップが基本です。（＝マシンをパワーアップさせます）

典型的には、空きスロットに、CPUやメモリを追加します。
後から追加できるように、最初から「空き」を用意しておくのが
ポイントです。

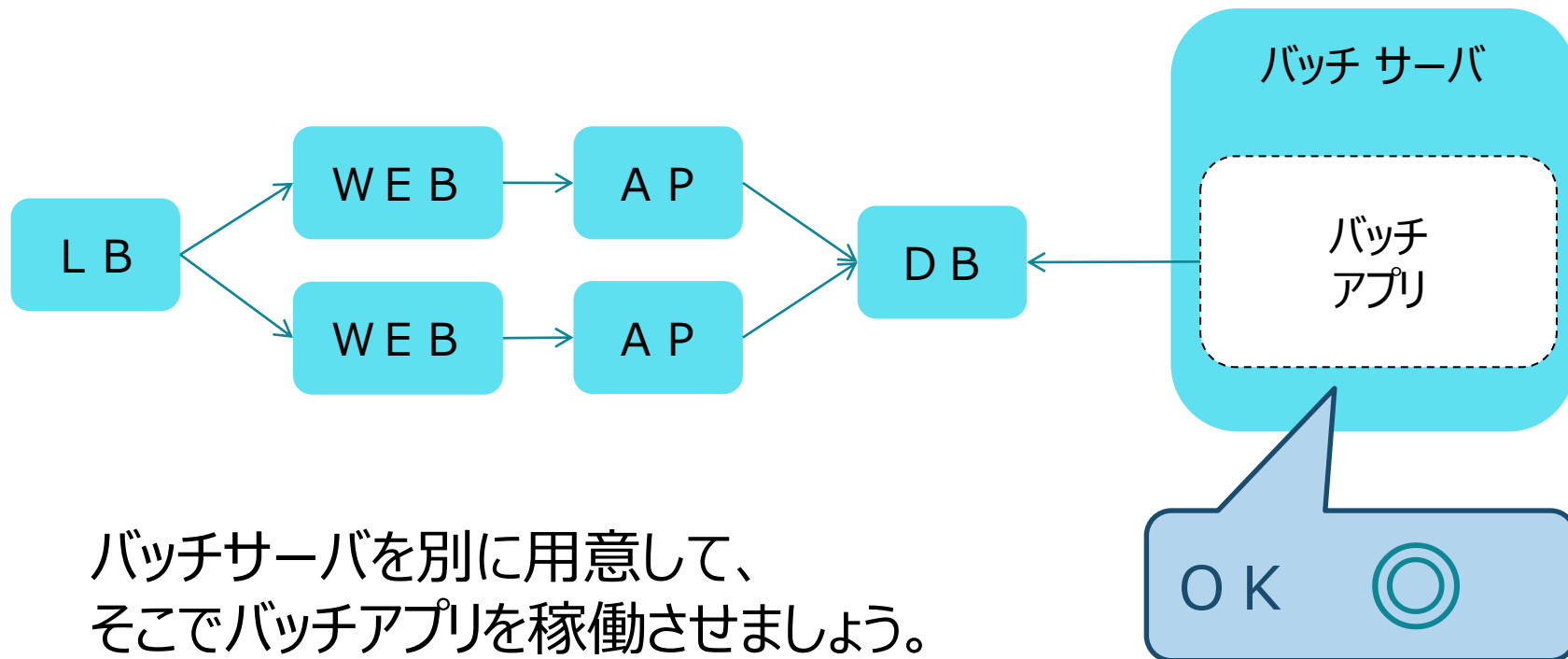
では、バッチについてはどうでしょうか？



バッチアプリを画面オンラインが動く A P サーバに配置するのは原則としては N G です。

画面オンラインとバッチで、A P サーバのリソース（C P U、メモリ）を消費してしまうためです。結果、お互いの性能が出ません。
[ただし、バッチの稼働時間に、画面オンラインが停止する場合は、O K です。]

バッチのサーバ構成



おわりに

- 非機能要件について学ぶ意味
- 非機能要件を網羅的に検討する方法
～非機能要求グレード～

- 非機能要件についての重要なトピック
 - インフラについての基礎知識
 - 可用性
 - 基本的な用語
 - 要件を合意するときのポイント
 - 実現方式（サーバ構成, タイムアウト）
 - 性能・拡張性
 - 要件を合意するときのポイント
 - 実現方式（MW設定, サーバ構成）

本コンテンツで取り上げなかったこと

取りあげなかった非機能要件のうち、
「セキュリティ」の項目がとくに重要です。

- ・可用性
- ・性能・拡張性
- ・運用・保守性
- ・移行性
- ・セキュリティ
- ・システム環境・エコロジー

このテーマは単独でも膨大な内容となるため
本コンテンツでは取り上げることができませんでした。

本コンテンツで取り上げなかったこと

「セキュリティ」については
まずは以下資料での学習をおすすめします。



IPA「安全なウェブサイトの作り方」

<https://www.ipa.go.jp/security/vuln/websecurity/ug65p900000196e2-att/000017316.pdf>

<https://www.ipa.go.jp/security/vuln/websecurity/about.html>
2023年4月4日11時の最新情報を取得

その他、本コンテンツに関連する内容として、本コンテンツと一緒に公開している以下をオススメします。

Webアプリケーションのセッション管理

- <https://fintan.jp/?p=8376>
- Fintan > アプリケーションアーキテクチャの学習コンテンツ > Webアプリケーションのセッション管理

データベースの排他制御

- <https://fintan.jp/?p=8376>
- Fintan > アプリケーションアーキテクチャの学習コンテンツ > データベースの排他制御

本コンテンツでは「可用性」のセクションでこれらの内容が登場しました。

更なる理解のために、お役立てください。

学び初めのハードルを突破できましたか？

本コンテンツの内容を「とっかかり」として

現場でご活躍されることを期待しております！

ITで、社会の願い叶えよう。



TIS INTEC
Group