

アーキテクチャ

分散システム

2011年12月13日

建部修見

はじめに

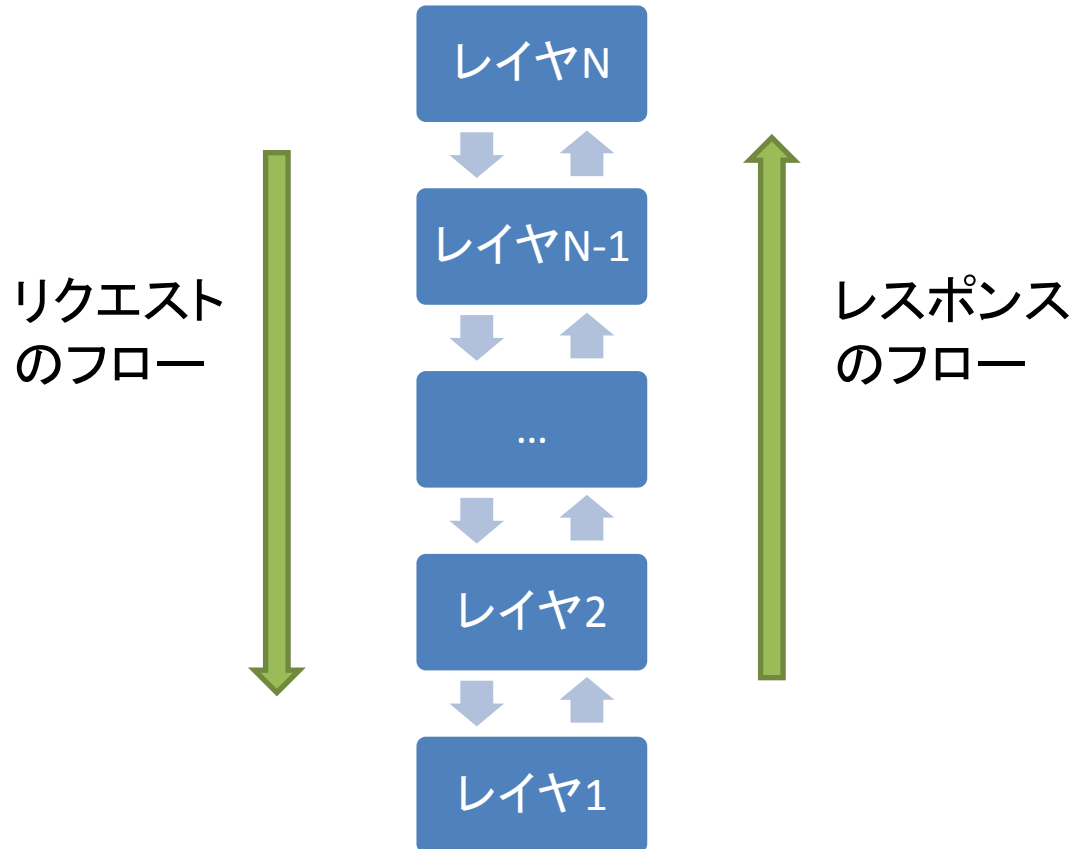
- ソフトウェアアーキテクチャ
 - どのようなソフトウェアコンポーネントで構成され、どのように相互作用が行われるか
 - アーキテクチャのスタイル
- システムアーキテクチャ
 - 集中アーキテクチャ
 - 分散アーキテクチャ
 - ハイブリッドアーキテクチャ
- 自立的システム (autonomic systems)
 - フィードバック制御

用語

- コンポーネント (component)
 - **明確に定義**された (well-defined) インターフェースを持つ **交換可能**な (ソフトウェアの) 構成単位
- コネクタ (connector)
 - コンポーネント間の通信, 調整, 協力を伝えるメカニズム
 - 遠隔手続き呼出し (RPC), メッセージパッシング, データストリーミングなど

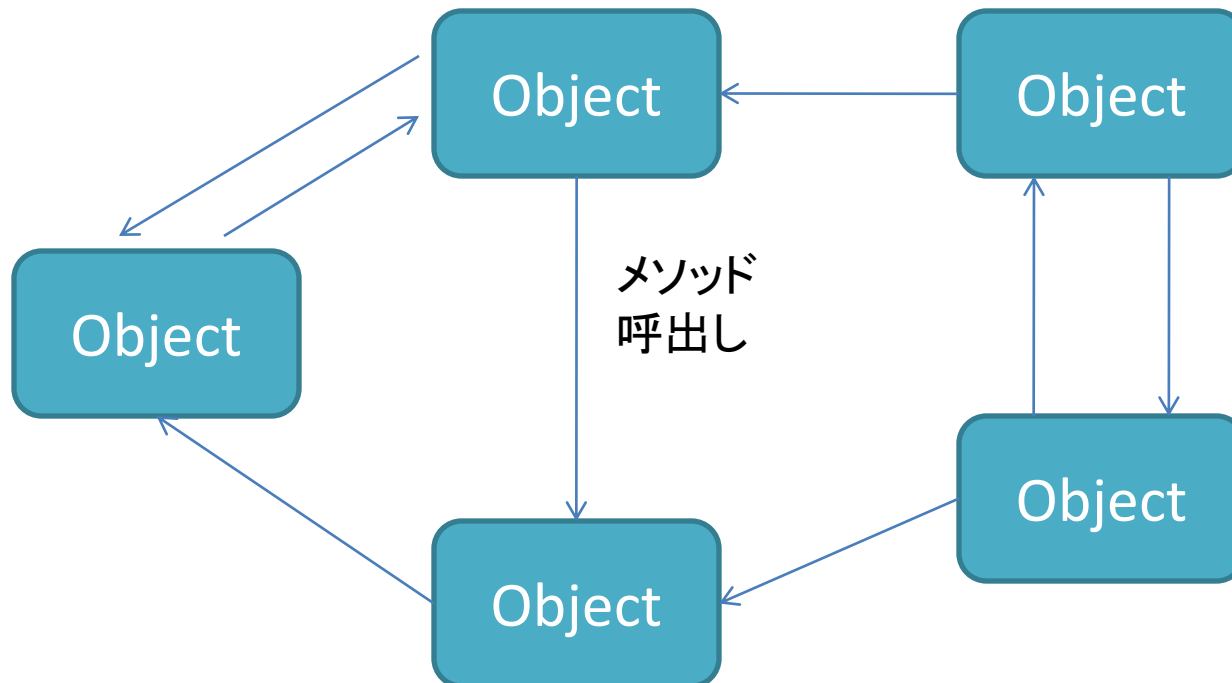
レイヤーアーキテクチャ (Layered Architectures)

- 層状アーキテクチャ, 階層アーキテクチャ
- レイヤ i はレイヤ $i-1$ を呼び出せる
- ネットワークのコンポーネントでよく利用される



オブジェクトベースアーキテクチャ (Object-based Architectures)

- より疎な構成
- オブジェクトがコンポーネント
- 遠隔手続き呼出し

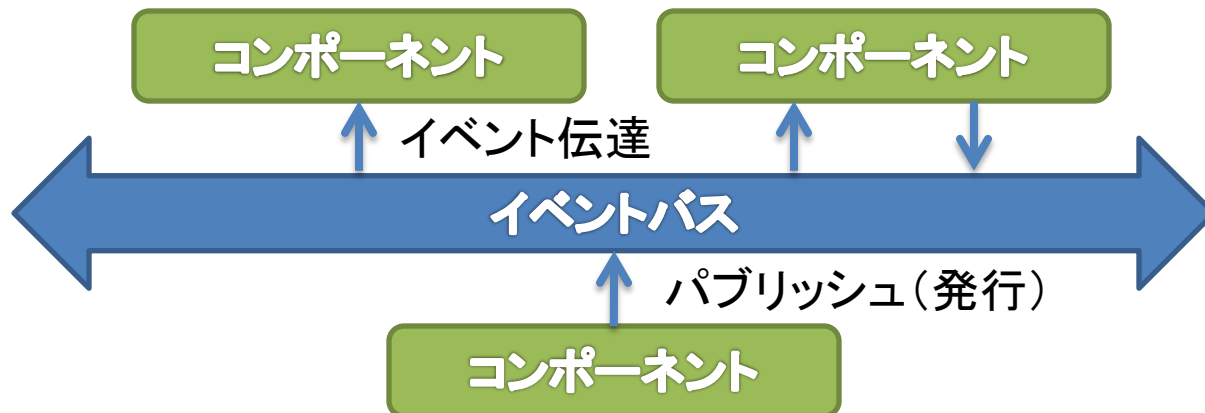


データセンタアーキテクチャ (Data-centered Architectures)

- **共有レポジトリ**により通信を行う
- 多くのネットワークアプリケーションは、**共有分散ファイルシステム**の**ファイル**を利用して通信を行う
- Webベースのアプリケーションは、Webベースの**共有データサービス**を利用する

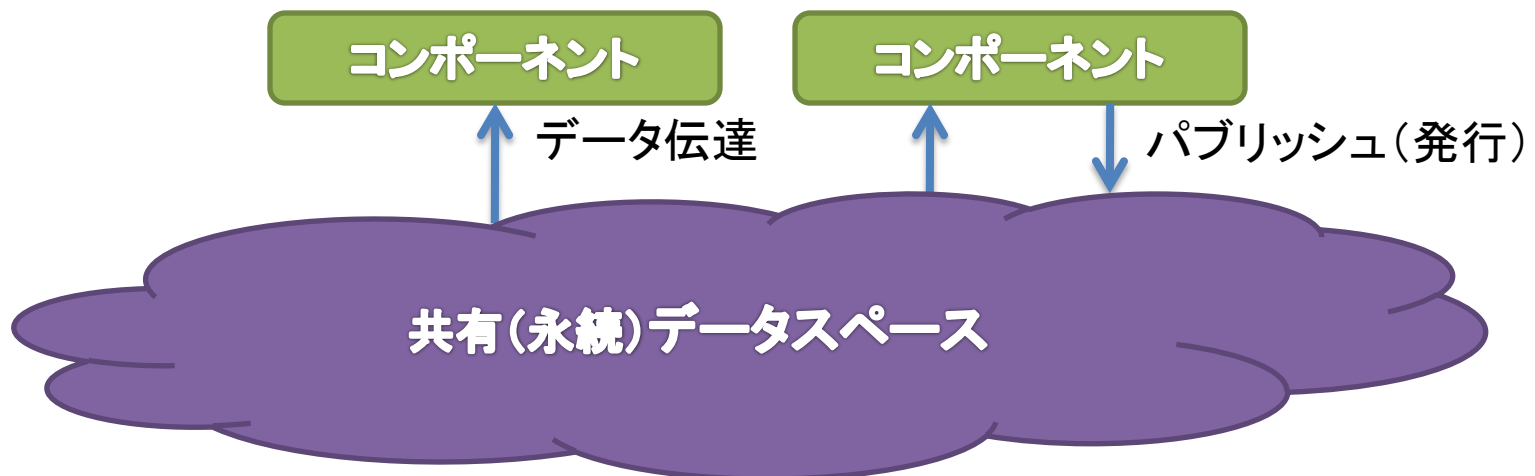
イベントベースアーキテクチャ (Event-based Architectures)

- イベントの伝搬で通信する
- 発行・購読 (Publish/subscribe) システム
 - 購読しているプロセスにイベントを発行する
 - 疎結合 (loosely coupled) 型プロセス
 - 参照分離 (Referentially decoupled)
 - お互いに参照する必要はない



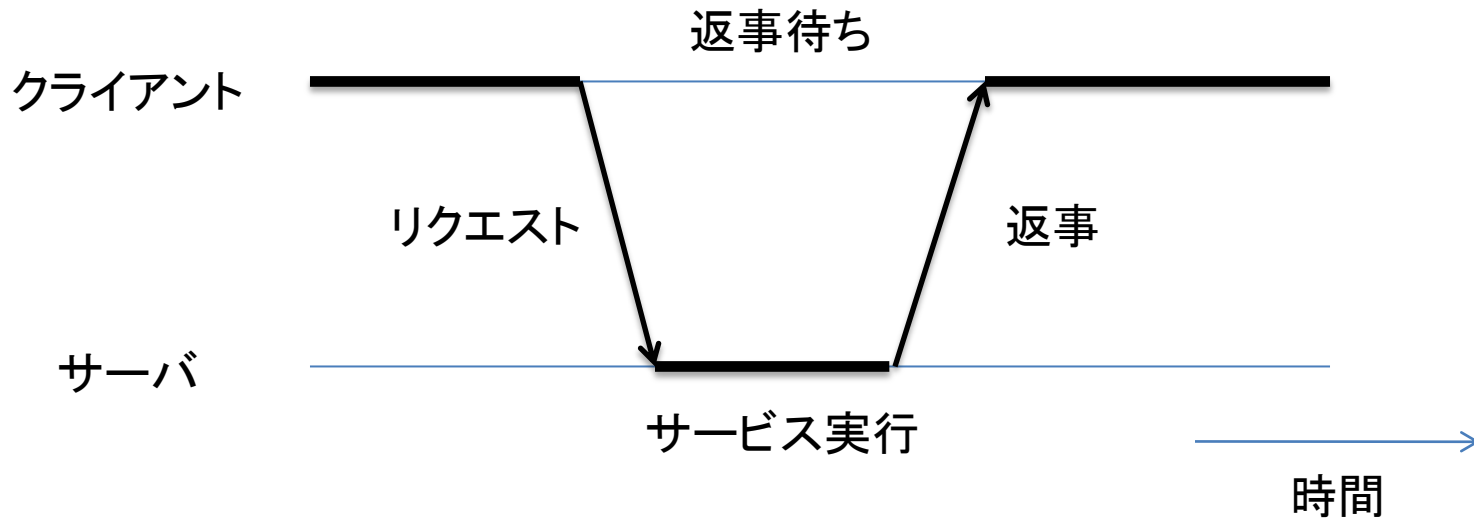
共有データスペース (Shared Data Spaces)

- データセンタアーキテクチャとイベントベースアーキテクチャの組合せ
- プロセスは時間的にも分離
 - 通信中にアクティブでなくてもよい
- SQL, ファイル



システムアーキテクチャ

- システムアーキテクチャ
 - コンポーネントの相互作用と配置の方法
- クライアントサーバモデル



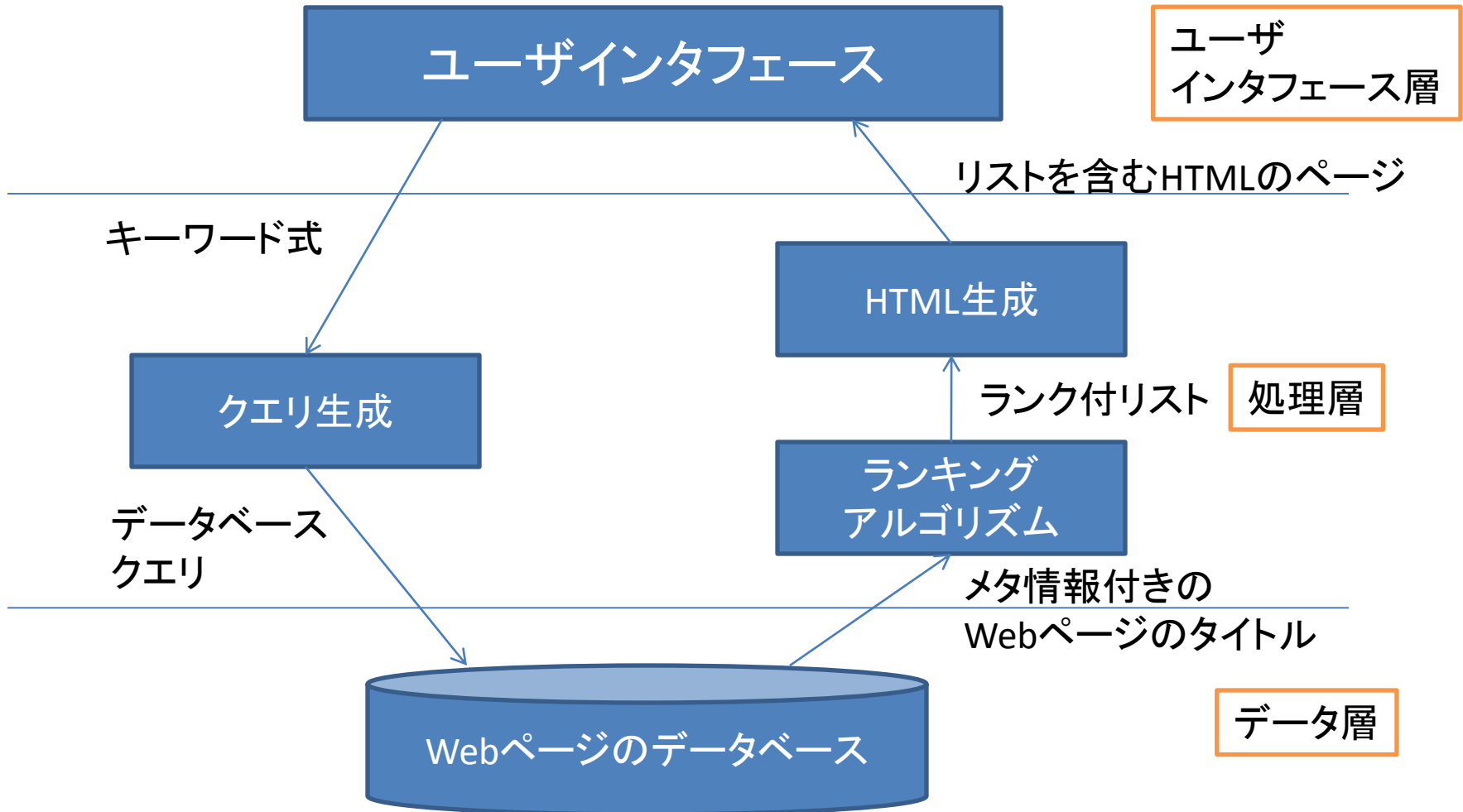
クライアントサーバモデル

- コネクションレスの通信 (例: UDP, user datagram protocol)
 - LANなど高信頼な環境では効率的
 - クライアントはメッセージ(サービスと引数)をサーバに送信,サーバは返事を送信
 - 信頼性のない環境, リクエストorレスポンスが失われる可能性
 - リクエストの再送信→サービスを二度実行する可能性
 - 「銀行口座から100万円引き出す」などは困る
 - 「残高照会」などは何度実行してもよい = **idempotent**な操作
- 信頼性のあるコネクション指向の通信 (例: TCP, transmission control protocol)
 - 広域環境のような低信頼な環境
 - コネクションを確立してリクエストを発行
 - コネクション(再)接続のコスト

アプリケーションのレイヤリング

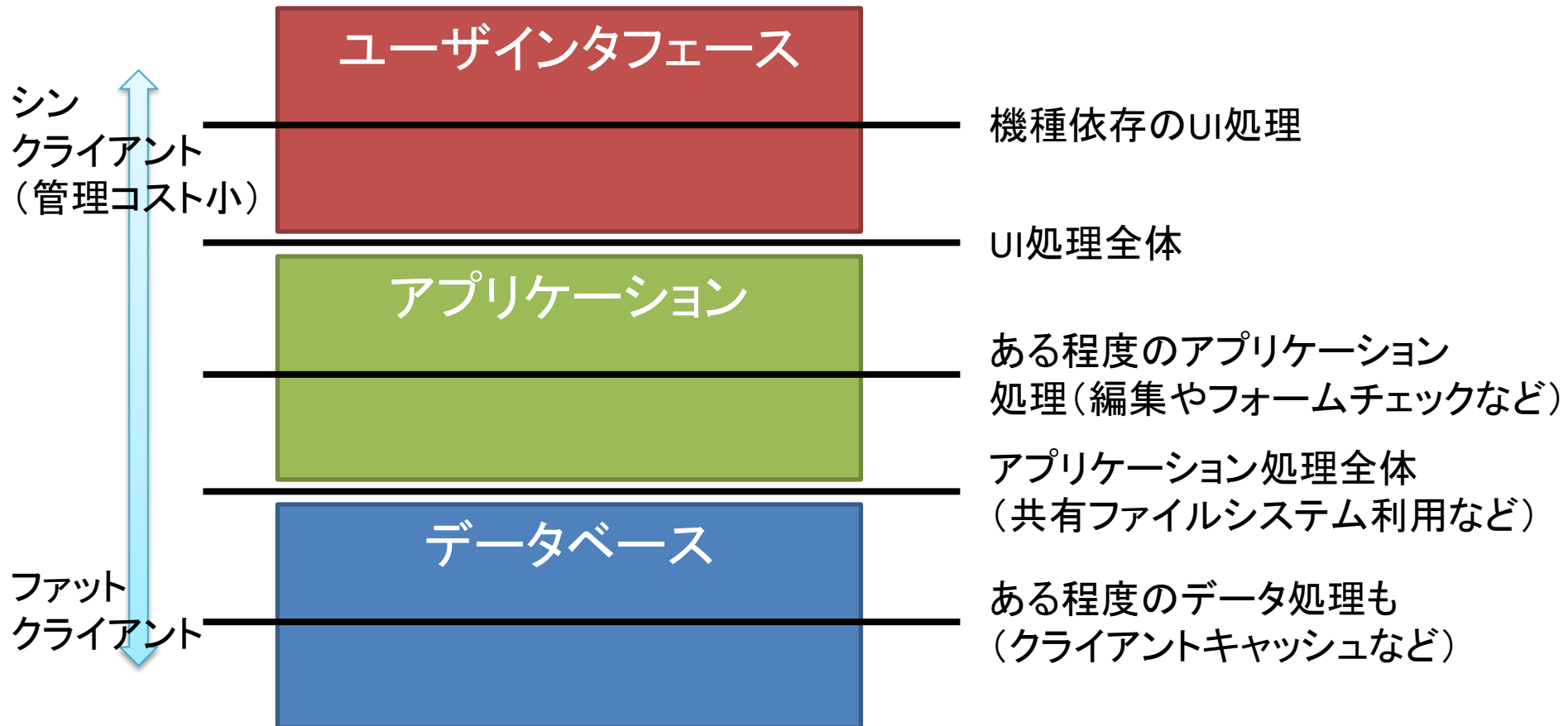
- (データベースをアクセスする)クライアントサーバアプリケーションは三層の階層からなる
 - ユーザインタフェース層 (user-interface level)
 - クライアント (キャラクタ, グラフィックス)
 - 処理層 (processing level)
 - それぞれのアプリケーション処理
 - データ層 (data level)
 - ファイルシステム, データベース
 - 永続性 (persistency) をもつ

インターネット検索エンジンの例



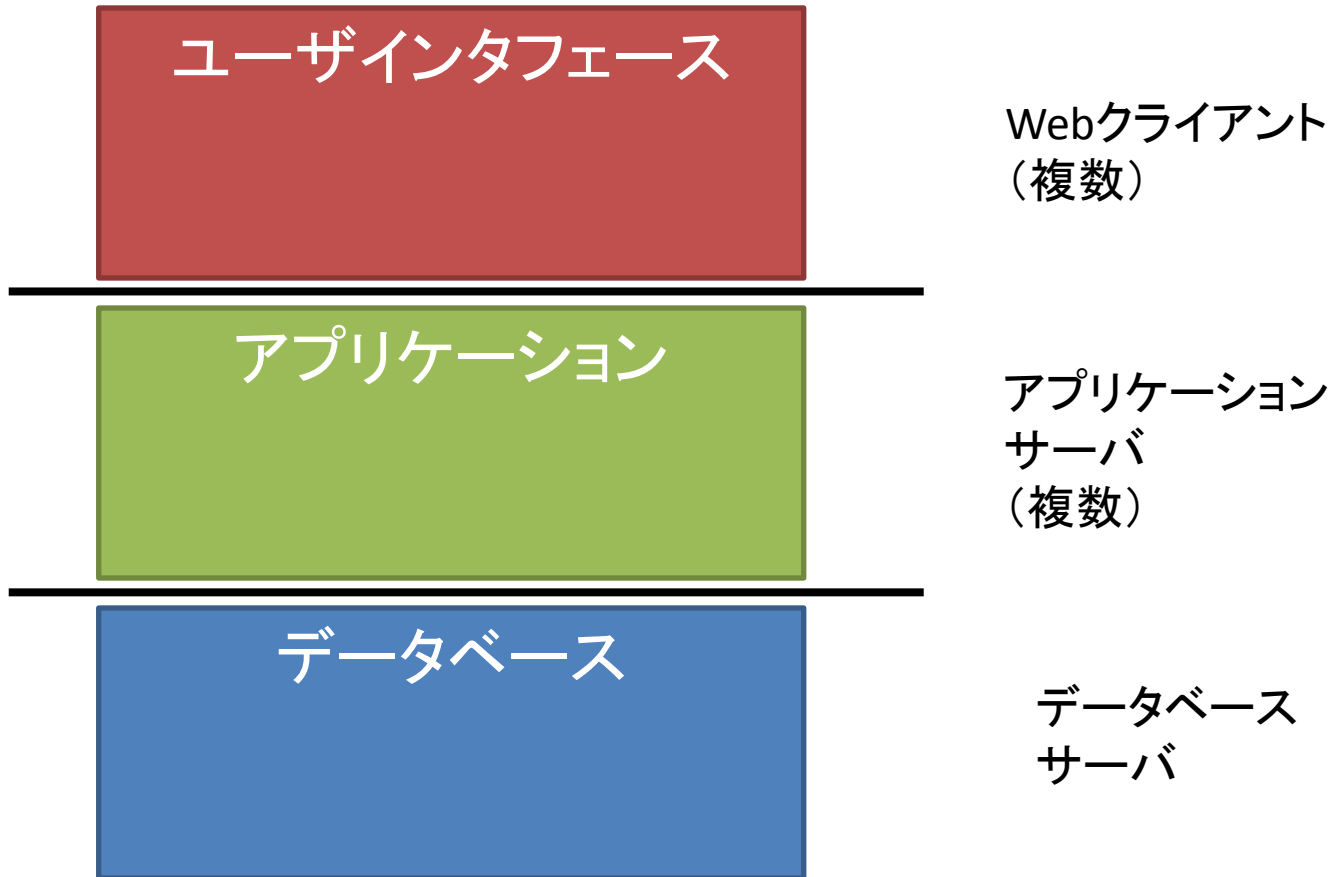
二層アーキテクチャ

- 三層レイヤをクライアントとサーバに分ける



多層アーキテクチャ

(例)



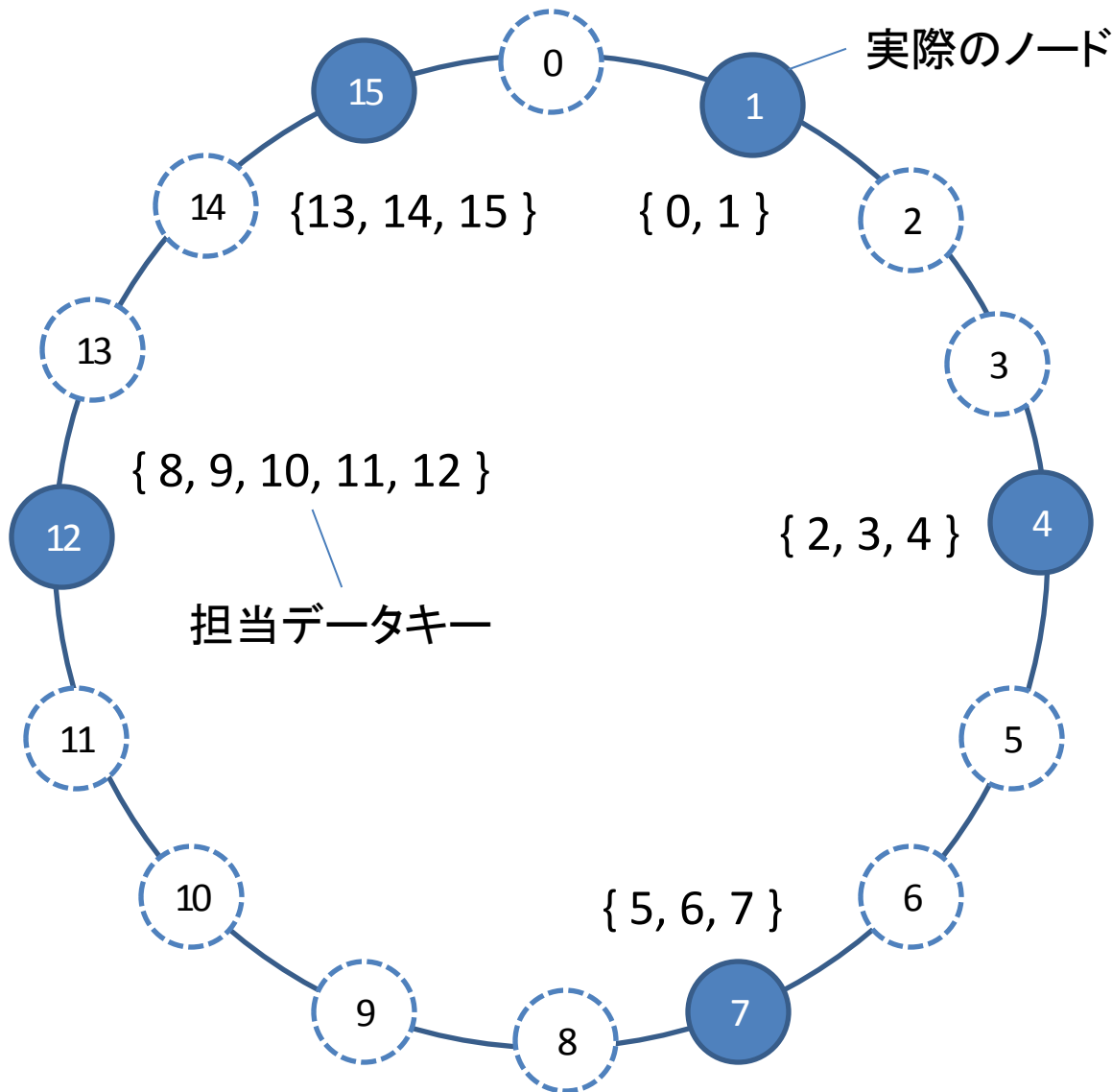
分散アーキテクチャ

- 垂直分散 (vertical distribution)
 - 機能単位を複数マシンで分散
- 水平分散 (horizontal distribution)
 - 同一機能を複数マシンで分散
 - 複数マシンで負荷を分散
 - Cf. P2P (peer-to-peer) システム
- P2P システム
 - (概念的には) P2P を構成するプロセスは同一
 - プロセス間の相互作用は対称的, クライアントでもありサーバでもある (サーバント, servent)
 - オーバレイネットワーク (overlay network)
 - プロセス間のネットワーク。ルーティングしてプロセス間でメッセージ通信

構造化P2Pアーキテクチャ

- オーバレイネットワークを決定的手続きで構成
- 分散ハッシュ表 (distributed hash table, DHT) に基づく
 - データは128ビット (MD5), 160ビット (SHA1) などの広いID空間のランダムなキーに割当てられる
 - 距離に基づきキーをノードのIDに割当てる
 - ハッシュ表をハッシュ値で分割
- データをLOOKUPするとき, そのデータが割当てられているノードを返す
 - データが割当てられているノードにルーティングする

Chord [Stoica et al., 2003]



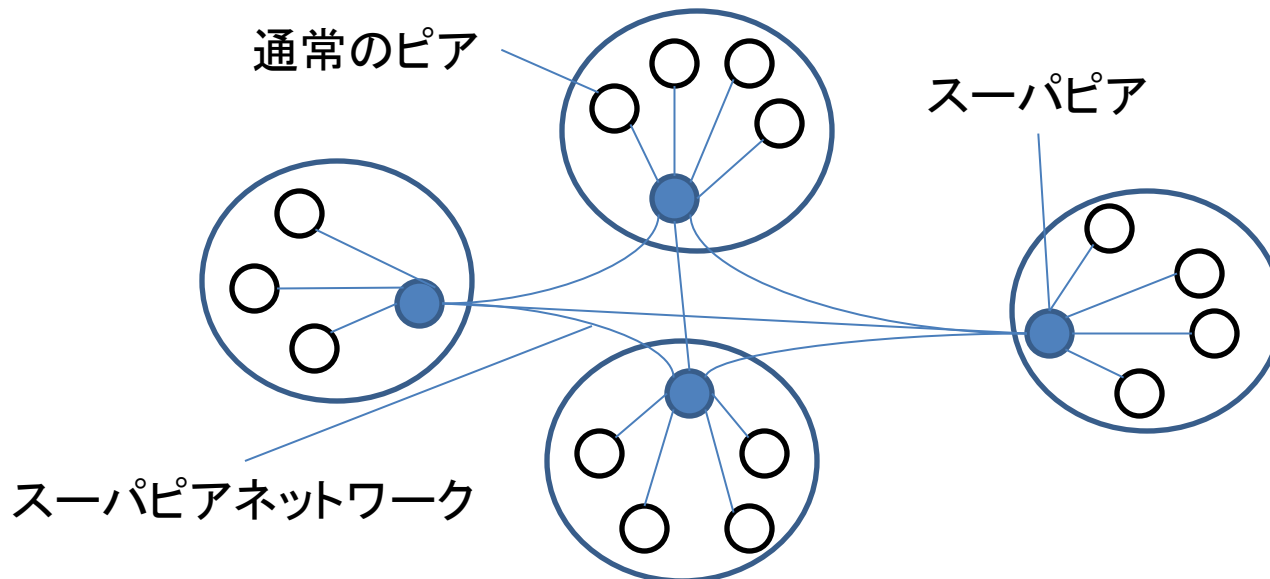
- $\text{succ}(k)$ は最小のノード $\text{id} \geq k$
- $\text{LOOKUP}(k)$ で $\text{succ}(k)$ を返す
(アルゴリズムは後の講義だが、 $O(\log N)$ ステップで検索)
- メンバシップ管理
(前後のノードに知らせる)

非構造化P2Pアーキテクチャ

- 乱数アルゴリズムでオーバレイネットワークを構築
- データもランダムに配置
- 検索はリクエストをフラッディング（ブロードキャスト）
- ランダムグラフの生成が目標
 - それぞれのノードが、生きているノードの内ランダムにcノードの情報を知っている

スーパーピア (Superpeers)

- 非構造P2Pではデータ検索は基本フラッディングなため、ピア数の増加に対し問題がある
- CDN (コンテンツデリバリネットワーク) をP2Pで実装する場合、コンテンツを高速に発見したい
- インデックスを保持し、ブローカ (仲介) となるノード = **スーパーピア** の導入 (cf. Sun JXTA)



ハイブリッドアーキテクチャ

- 協力的 (collaborative) 分散システム
- BitTorrentファイル共有システム[Cohen, 2003]
 - 協力的なP2Pファイルダウンロード
 - ファイルのダウンロードは, コンテンツを提供するノードだけが可能
 - .torrentファイルはトラッカ (tracker) を示す。トラッカはファイルのチャンクを保有するアクティブなノードを保持
 - アクティブノードは現在ほかのファイルをダウンロードしているノード

まとめ

- ソフトウェアアーキテクチャ＝ソフトウェアの論理的な構成
- システムアーキテクチャ＝コンポーネントがどのように異なるマシンに配置されるか
- アーキテクチャのスタイル
 - レイヤ, オブジェクト指向, イベント指向, データスペース指向
アーキテクチャ
- クライアントサーバモデル
 - 集中アーキテクチャとなりやすい
- P2Pシステム
 - プロセスは等しく振る舞う
 - オーバレイネットワーク＝ほかのピアの局所リストを持つ論理的なネットワーク
 - 構造化P2Pと非構造化P2P

演習問題

- ソフトウェアアーキテクチャ、システムアーキテクチャとは何か？
- 国際標準化機構 (ISO) が定めた OSI (Open Systems Interconnection) 参照モデルを調べよ
- 構造化 P2P における DHT は Chord の他にどのようなものが提案されているか？それぞれの特徴をまとめよ。