分散システム 第2回
― アーキテクチャ ―

大連理工大学・立命館大学 国際情報ソフトウェア学部

大森 隆行

講義内容

- アーキテクチャ
  - 概説
  - ソフトウェアアーキテクチャ
  - システムアーキテクチャ
    - クライアントサーバ

分散システムとアーキテクチャ

- 分散システムは複雑になりやすいため、適切に設計することが重要
  - 様々なソフトウェア部品から構成される
  - それらが様々なコンピュータ上に配置される

- アーキテクチャが重要
  - ソフトウェアアーキテクチャ
  - システムアーキテクチャ

ソフトウェアアーキテクチャ (software architecture)

- ソフトウェアの全体的な構造であり、システムの概念的な一貫性を提供する手段
  - モジュールの構造や構成
  - コンポーネント間の相互作用
  - コンポーネントが使用するデータの構造 等
- (広義には、システムの主要な構成要素とそれらの相互作用)
- 開発を方向付ける
  - 基本構造から大きく外れる改造は困難
- 構成要素
  - コンポーネント：明確なインタフェースを持つソフトウェア部品
  - コネクタ：コンポーネント間の通信、協調、協力のための仕組み

システムアーキテクチャ

- コンピュータシステムの全体的な構成や動作原理
  - システムがどのような要素で構成されるか
  - 構成要素間の関係
  - システム外の環境との関係

- コンピュータシステム構築の際に重要
  - アーキテクチャはハード、ソフト両面を考慮した設計を含む

講義内容

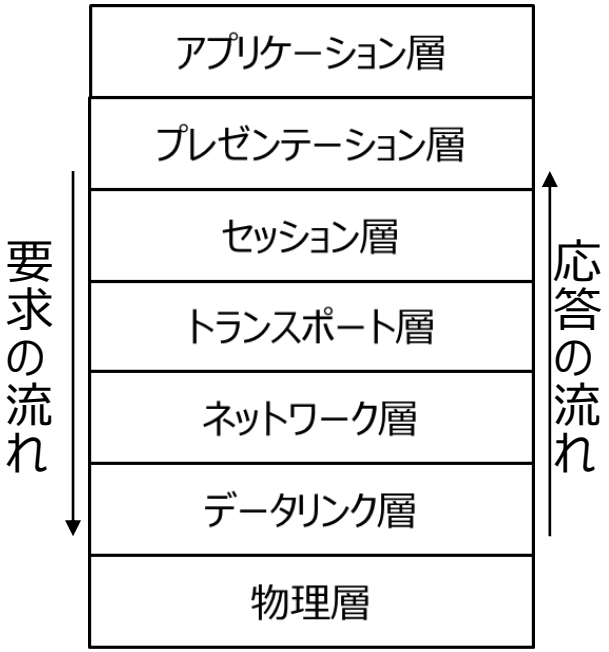
- アーキテクチャ
  - 概説
  - ソフトウェアアーキテクチャ
  - システムアーキテクチャ
    - クライアントサーバ

ソフトウェアアーキテクチャの例

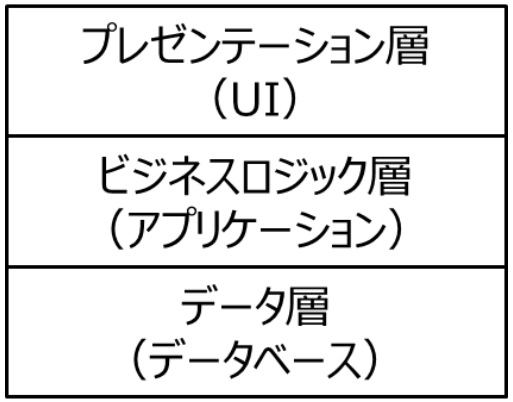
- 階層型アーキテクチャ
- オブジェクトベースアーキテクチャ
- データ中心型アーキテクチャ
- イベントベースアーキテクチャ

階層型アーキテクチャ

- i番目の層は、直下の層（i-1番目の層）に対して、サービスの要求を行う



ISO 7層モデル

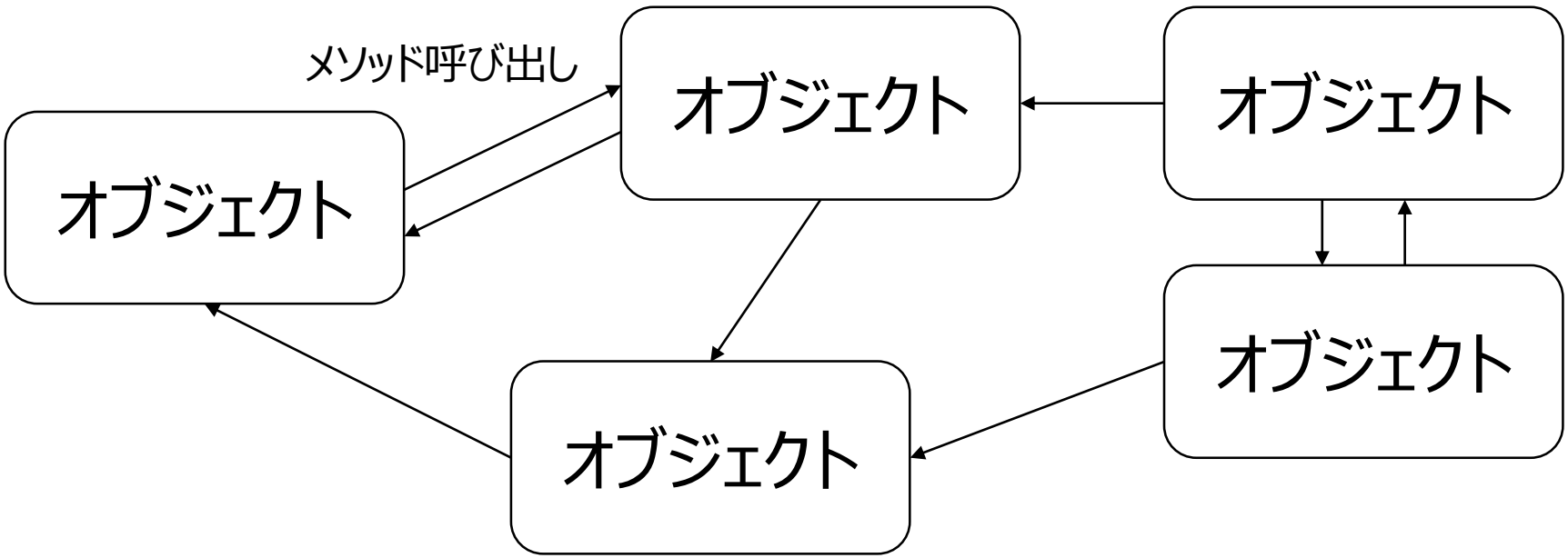


Webアプリケーション
3層モデル

仮想マシンモデル

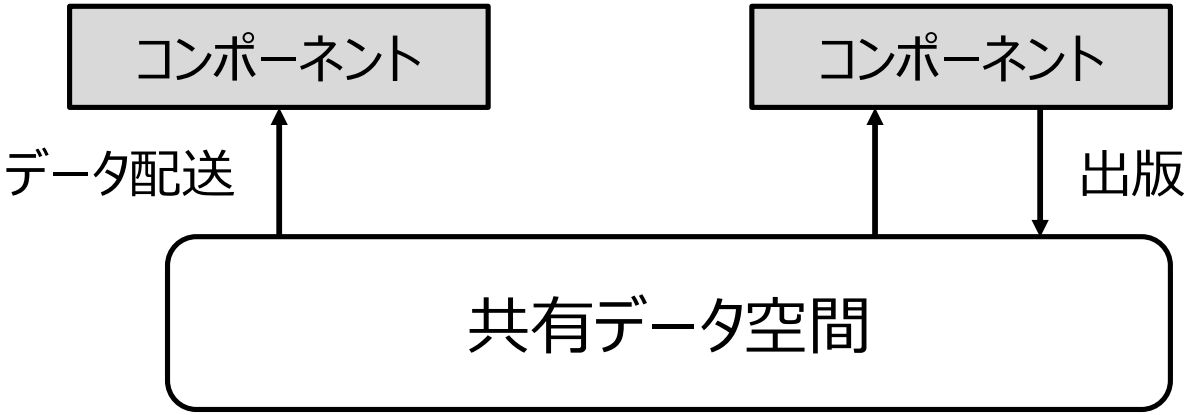
オブジェクトベースアーキテクチャ

- オブジェクトをコンポーネントとする
  - 上下関係はない
- クライアントサーバともよくマッチする



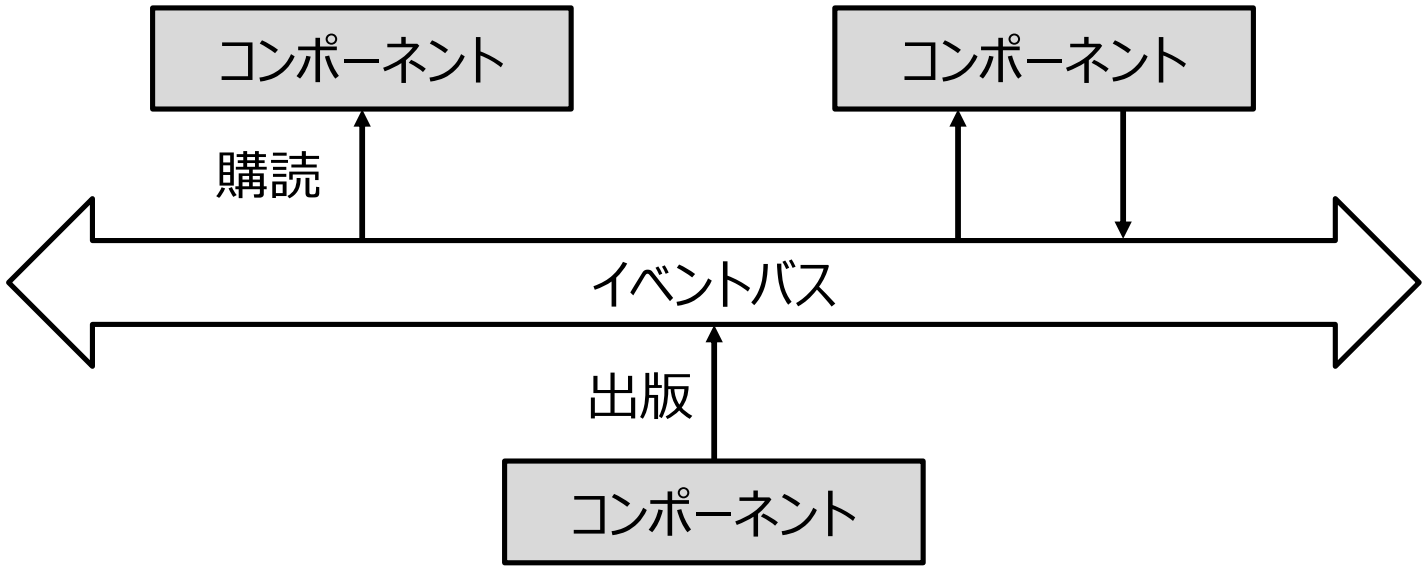
データ中心型アーキテクチャ

- 共有リポジトリ(データベース)への書き込みや参照によって通信を行う
- 一貫性を持ってデータを管理するシステムとよく適合する



イベントベースアーキテクチャ

- コンポーネント同士がイベントを介して通信
  - 互いを明示的に参照する必要がない (疎結合)



確認問題

- 以下の各文は正しいか。○か×で答えよ。
  - 分散システムを構成するコンピュータの種類は一様である。
  - システムのアーキテクチャを考える際はソフト、ハード両面から考えなければならない。
- 各説明に合うアーキテクチャの種類を答えよ。
  - あるコンポーネントがイベントを発行する。発行されたイベントはイベントバスにより伝達される。他のコンポーネントはイベントバス上のイベントを購読する。
  - 複数のコンポーネントが共有のデータ空間を持ち、データの一貫性を保ったままデータの更新・取得を行うことができる。
  - コンポーネントがオブジェクトとして実現されており、オブジェクト同士はメソッド呼び出しにより連携する。
  - 複数の階層により構成される。各層において、他層の実装の内容を考慮する必要がない。

選択肢：イベントベースアーキテクチャ、階層型アーキテクチャ、
データ中心型アーキテクチャ、オブジェクトベースアーキテクチャ

講義内容

- アーキテクチャ
  - 概説
  - ソフトウェアアーキテクチャ
  - システムアーキテクチャ
    - クライアントサーバ

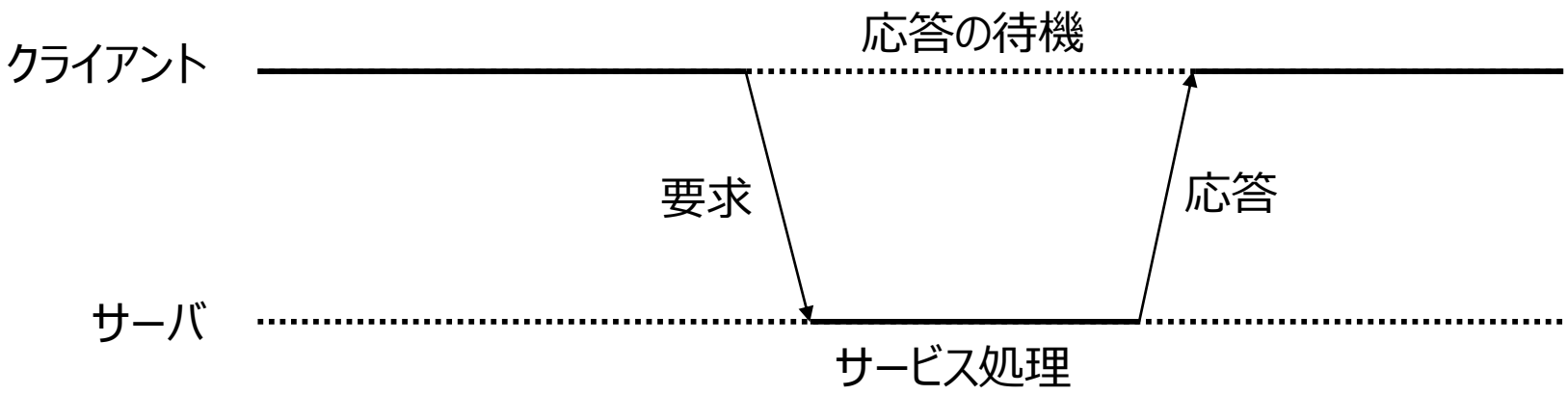
システムアーキテクチャの例

- 集中型(centralized)
  - クライアントサーバ
    - 役割の異なるグループ(クライアント・サーバ)により構成
- 非集中型(decentralized)
  - ピアツーピア(peer-to-peer)
    - 原則として各ノードが対等

クライアントサーバモデル

- システムを2つの異なる役割を持つグループ
  クライアントとサーバに分ける

- サーバ(server)
  - サービスを提供する端末やアプリケーション
    - (例) ファイルサーバ、印刷サーバ、Webサーバ…
- クライアント(client)
  - サービスを要求する端末やアプリケーション

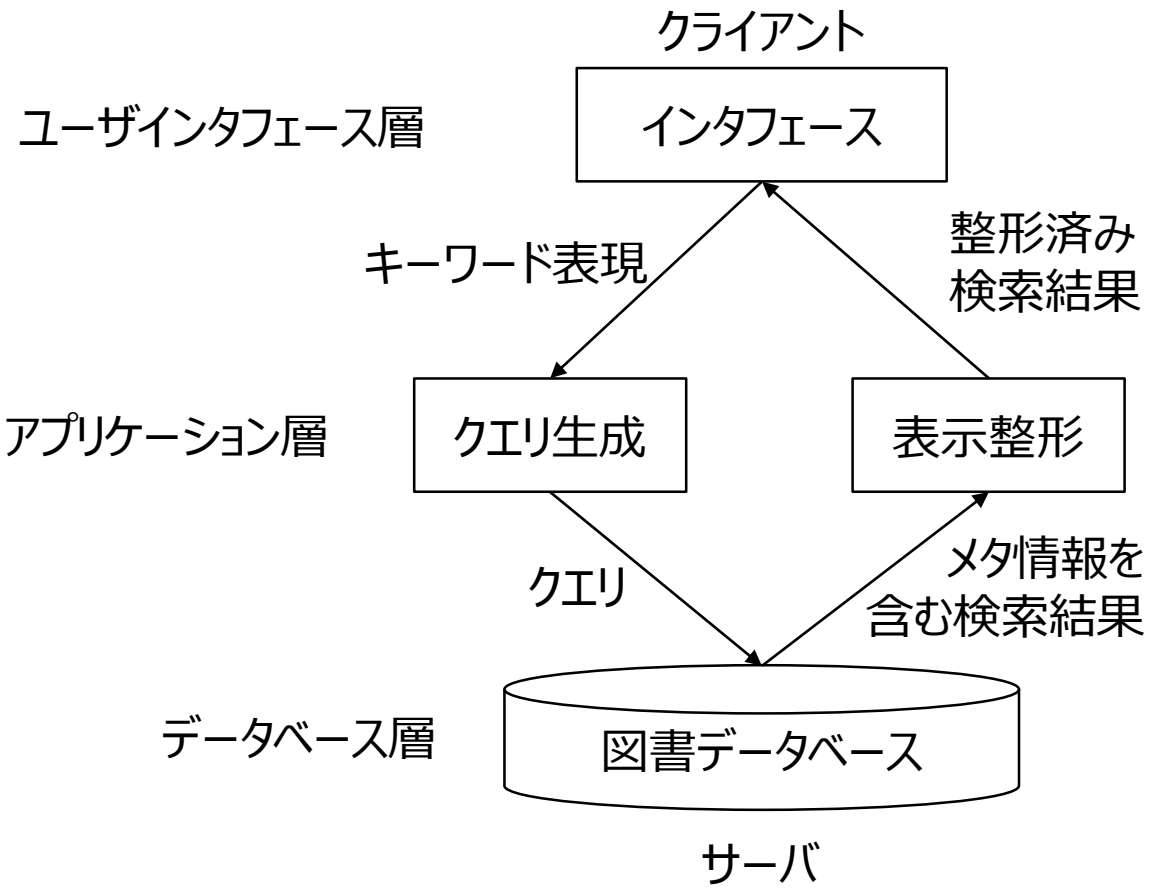


クライアントサーバモデル

- クライアント・サーバ間のプロトコルが明確に定義されていなければならない
  - どのように要求し、どのように回答するか
- クライアント・サーバ間はメッセージの交換で接続
  → 独立性が高い
  → 実現が比較的容易
  - ただし、単一のコンピュータがクライアント兼サーバになることもある

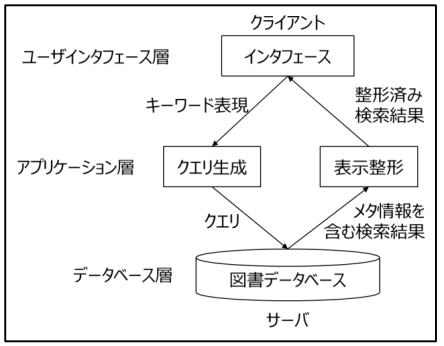
クライアントサーバモデルの例

図書検索システム

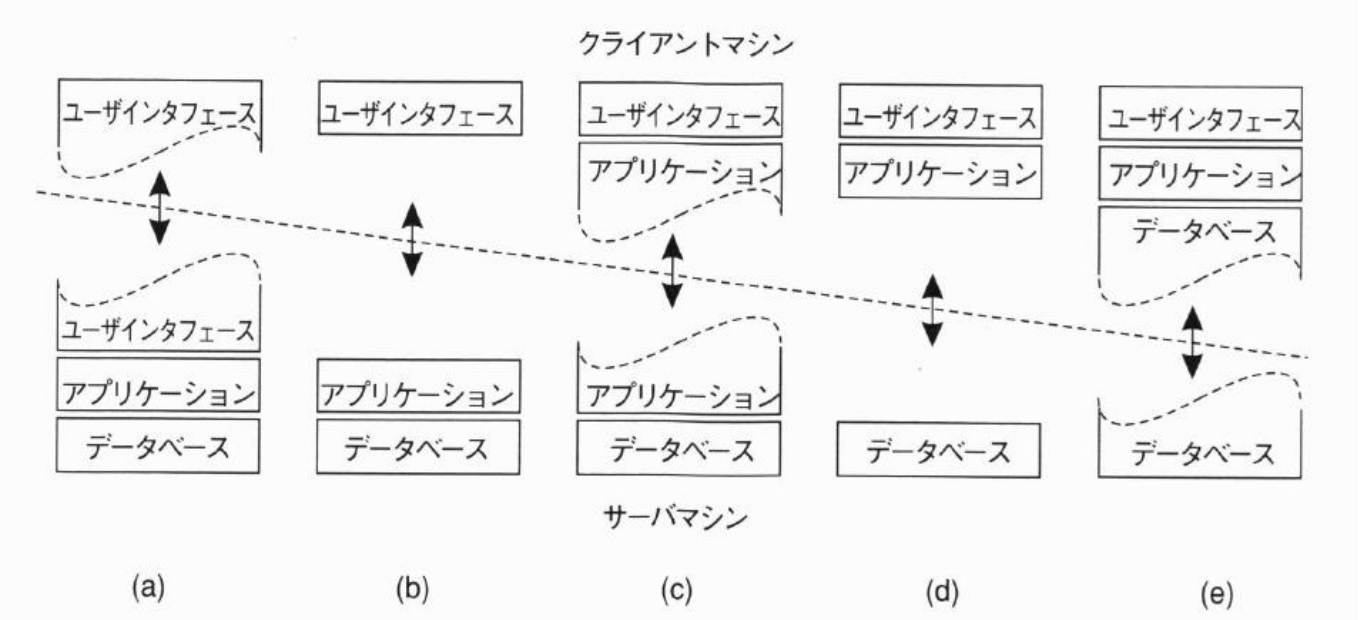


- ユーザ側はデータベースの詳細は知らなくても良い
  ＝内部を隠蔽可能
- システム構成としては、必ずしもクライアント・サーバの2層で捉える必要はない

クライアントサーバモデル



- クライアント・サーバの分け方も様々



シンクライアント
(thin client)

- クライアント性能が低いとき有効
- 通信量増大

ファットクライアント
(fat client)

- クライアント性能が高いとき有効
- 通信量減少
- データの一貫性やセキュリティの問題

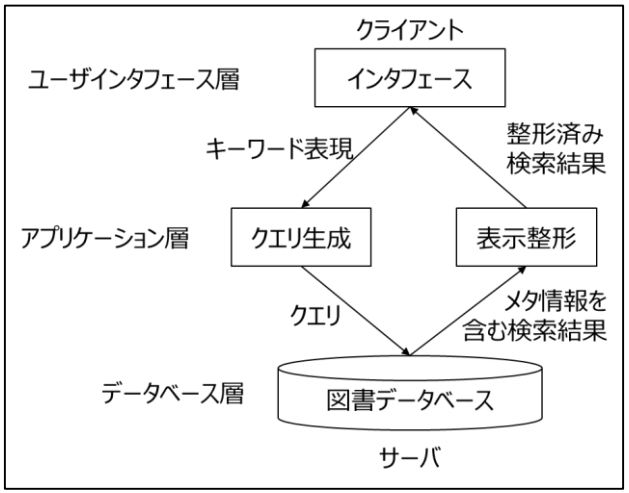
クライアントサーバモデル

- クライアント・サーバの分け方も様々
  - 3層以上に分割する方法もあり得る

クライアント ・・・

アプリケーションサーバ ・・・

データベースサーバ ・・・

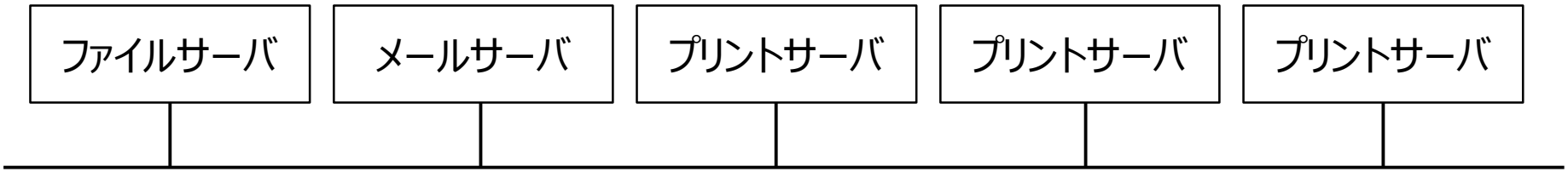


- 常に明確にクライアントとサーバを区別できるとは限らない
  - この例では、アプリケーション"サーバ"は
    データベースサーバに対しては"クライアント"となる

垂直分散・水平分散

- 垂直分散 (vertical distribution)
  - 役割が異なる構成要素に分割
  - クライアントサーバモデルが該当
- 水平分散 (horizontal distribution)
  - 対等な構成要素に分割
  - ピアツーピアモデルが該当
  - (例) 大規模Webシステム
    同一のコンテンツを持ったWebサーバを複数立ち上げ、負荷が集中しないように接続要求を振り分ける

垂直分散・水平分散



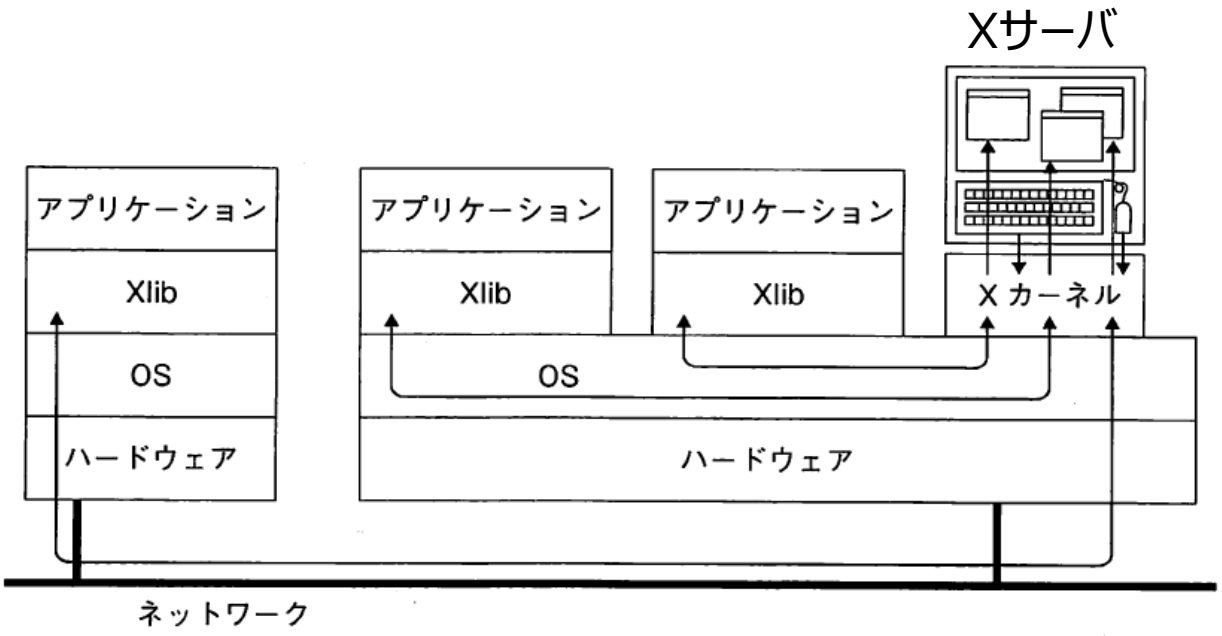
- 垂直分散 兼 水平分散 もあり得る
  - 垂直分散：機能ごとにサーバを分割
    - 各サービスに応じた性能の計算機を使えば良い
    - 無駄なコストを省ける可能性
  - 水平分散：同一の機能を持つサーバを複数提供
    - (上記の例の場合) 大量の印刷要求に対応できる

確認問題

- ()内から正しいものを選べ。
  - クライアントサーバ型は(集中型・非集中型)のシステムアーキテクチャに属する。
  - ピアツーピア型は(集中型・非集中型)のシステムアーキテクチャに属する。
  - クライアントに比較的多くの役割を持たせるアーキテクチャを特に(シンクライアント・ファットクライアント)と呼ぶ
  - クライアントの機能を比較的限定したアーキテクチャを特に(シンクライアント・ファットクライアント)と呼ぶ
- 以下の各文は正しいか。○か×で答えよ。
  - ピアツーピア型のシステムでは、各ノードが原則として対等である。
  - クライアントサーバ型のシステムでは、クライアントがサービスを提供し、サーバからの接続を待ち受ける。
  - 垂直分散では、役割が異なる構成要素に分割する。
  - 水平分散では、対等な役割を持つ構成要素に分割する。

クライアントサーバモデルの例(2)

- Xウインドウシステム
  - シンクライアント方式
  - ネットワークを介してウインドウ環境(GUI)での操作を提供
  - Xサーバ：入出力機能を提供するサーバ
    (アプリケーションプロセスは別途存在)



サーバの分類

- 反復サーバ (iterative server)
  1つのクライアントからの要求を処理し終えると次の要求を受け付け可能となる
  - 短時間の接続で済む場合には有効
- 並行サーバ (concurrent server)
  複数のクライアントからの要求を同時に(並行して)受け付け可能
  - 複数のプロセス(orスレッド)で対応する
  - 無限に受け付け可能なわけではない

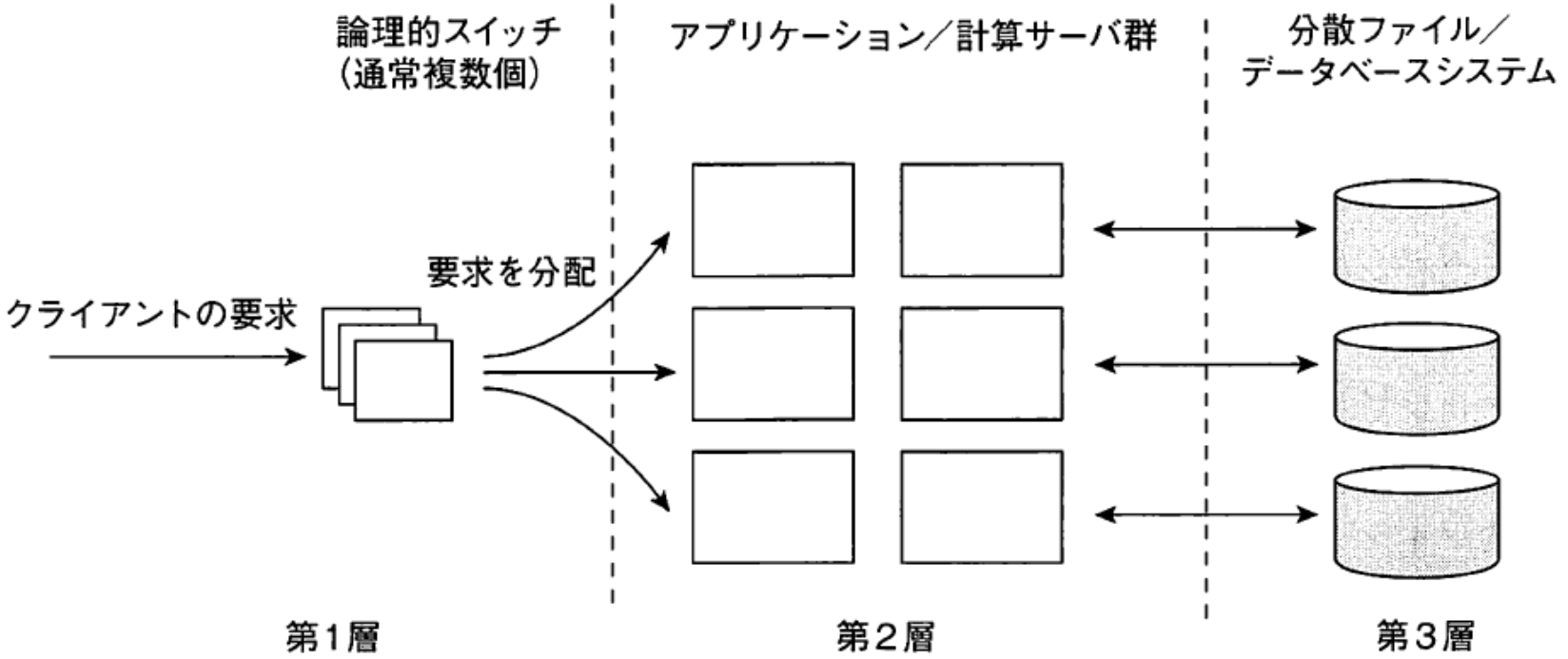
サーバの分類

- エンドポイント(endpoint)：
  クライアントからの要求の伝達先
  →ポート(port)とも呼ばれる
- クライアントからの接続の方法による分類
  - ウェルノウンポート (well-known port)を使用
    - 広く知られたサービスに対して
      標準的に割り当てられたポート
    - クライアントに明示的に通知しなくても接続可能
    - (例) FTP：TCPの21番、HTTP：TCPの80番
  - 窓口となるサーバに問い合わせる
    - 例えば、問い合わせ時にプロセスを用意し、
      エンドポイントの対応付けもその時点で行う

サーバの分類

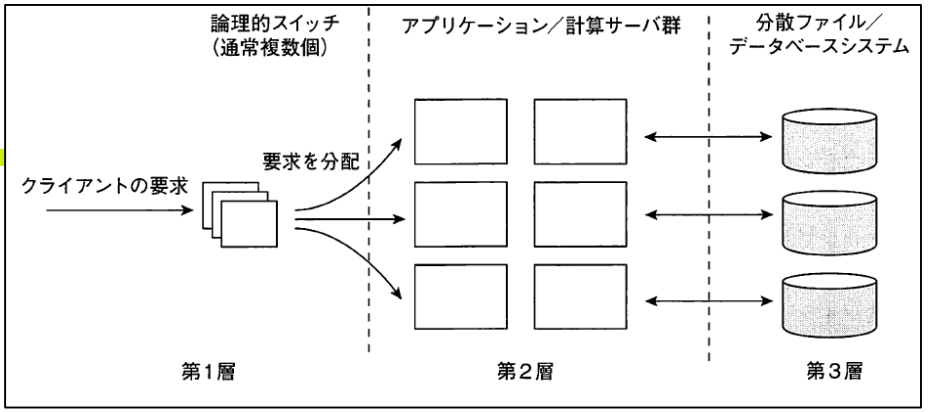
- ステートフル (stateful)
  - サーバが接続に関する状態管理を行う
  - 状態に関する情報のやり取りを減らすことができるが、クライアント増加の際の負荷増大が大きい(スケーラビリティで劣る)
- ステートレス (stateless)
  - サーバが接続に関する状態管理を行わない
  - (例) HTTPサーバ
    - Cookieを使えば状態管理可能

サーバクラスタ



- 論理的スイッチによって適切なサーバに計算を依頼
- 各層で必要な性能に特化したマシンを使用
  - 性能要求、使用できるマシンの制約等により2層運用もありうる

サーバクラスタ



- 第1層は複数のサーバの存在を隠蔽
  - TCPハンドオフ
    - クライアントはスイッチに対して接続要求
    - スイッチが選定したサーバが応答
    - OSによるサポートが必要
  - スイッチによるサーバの選定
    - 負荷分散
      - e.g., ラウンドロビン方式 (round robin)

確認問題

- 次の各文は正しいか。○か×で答えよ。
  - 反復サーバでは、クライアントからの要求を処理している間は次の要求を受け付けられない。
  - 並行サーバでは、複数のクライアントからの処理を同時に受け付けられる。
- 空欄にあてはまる語句を答えよ。
  - (1)は、コンピュータネットワークにおいて通信を行うためのエンドポイントを意味する。
  - (1)のうち、よく知られたサービスについては固定的な番号が割り振られており、これを(2)と呼ぶ。
  - (3)なサーバは、接続に関する状態管理を行う。

参考文献

- 「分散システム」
  水野 忠則 監修、共立出版、2015
- 「分散システム 原理とパラダイム 第2版」
  アンドリュー・S・タネンバウム 他 著、2009