データ工学特論（3）

情報基盤センター
天野　浩文

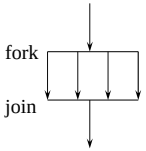
この講義に関するwebサイト:
http://isabelle.cc.kyushu-u.ac.jp/~amano/data-engineering/

最適なストライピングユニットサイズの選択

- RAIDの性能を上げる…相反する2つの目標
  - 各ディスクにできるだけ多くのデータを転送させる
  - できるだけ多くのディスクを入出力に参加させる
- **ストライピングユニット**
  あるストライプの一部として各ディスクに保存される連続したデータ領域⇒このサイズが RAID の性能を大きく左右する
  - トレードオフの問題
    - 小⇒少量のデータでも多くのディスクに分散する
    - 大⇒少量のデータなら少ないディスクにまとまる
- 最適なストライピングユニットサイズの選択は解析が困難
  - 限定された単純なモデルでの推定
  - シミュレーション・実験による推定

RAID の解析的評価を行う上での問題点

- ブロック系 RAID にアクセスする場合には
  - 個々のディスクに対するリクエストに fork (それぞれキューにつながれる)
  - 処理が終了したら join (同期の処理が入る)



- 待ち行列理論による解析
  - 厳密解は2サーバでのfork-join問題までしか得られていない
- アプリケーションの入出力要求のモデル化も難しい
  - 待ち行列理論では，負荷が一時的に高くなった場合の過渡的な挙動をモデル化しにくい
  - 定常的な状態（リクエストがたまり続けることなく処理できる状態）での挙動を調べるのがせいぜい

Chen と Patterson による経験則(1990)　ー(1)

- 16台のディスクが同期して回転する場合の最適ストライピングユニットサイズ
  - 平均リクエストサイズと，同時に発生するリクエストの多重度とを変化させてシミュレーション
  - 最適なスループットを与えるストライピングユニットサイズ
- 結果
  - 最も大きな影響を与えるパラメータは多重度である
- 多重度があらかじめわかっているとき，任意のリクエスト分散に対して最大スループットの 95%を与える式

$$1\text{セクタ} + \frac{1}{4} \times \text{平均位置決め時間} \times \text{データ転送速度} \times (\text{多重度} - 1)$$

（平均位置決め時間＝平均シーク時間＋平均回転遅延）

Chen と Patterson による経験則(1990) ー(2)

- 直観的な解釈

$$1\text{セクタ}+\frac{1}{4}\times\text{平均位置決め時間}\times\text{データ転送速度}\times(\text{多重度}-1)$$

  - 多重度が増えれば，サイズは大きくしたほうがよい
  - 多重度が下がれば，サイズは小さくしたほうがよい
  - 平均位置決め時間とデータ転送速度の積はストライピングのコストと利益のバランスを表している
    - コスト:ストライピングしたことによる位置決め時間の増加
    - 利益:ストライピングによる転送速度の増大
- 多重度がわからないときの近似式

$$\frac{2}{3}\times\text{平均位置決め時間}\times\text{データ転送速度}$$

LeeとKatzの解析(1991)

- 冗長ディスクのないRAIDの解析的モデルを設定し，最適ストライピングユニットサイズを与える式を導出

$$\text{最適ストライピングユニットサイズ}=\sqrt{\frac{PX(L-1)Z}{N}}$$

  P: 平均位置決め時間
  X: データ転送速度
  L: 多重度
  Z: リクエストサイズ
  N: ディスク数

- この式でも，平均位置決め時間とデータ転送速度の積*PX*が出てきている

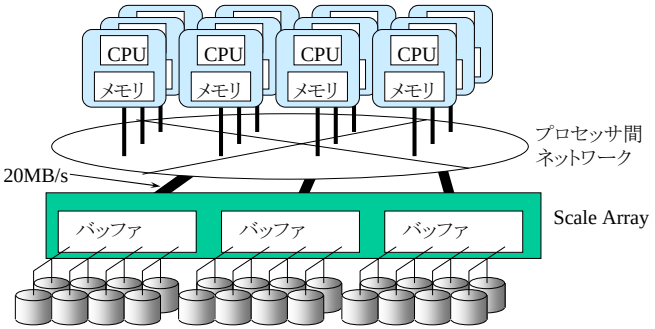
ChenとLeeの研究(1993)

- RAID-5 の最適ストライピングユニットサイズ
  - 読み出しが多い場合は RAID-0 と同じ
  - 書き込みが多い場合は，read-modify-write を行うよりもフルストライプアクセスのほうが速い
    ⇒ストライピングユニットサイズは小さくするとよい
- ディスク数と最適ストライピングユニットサイズ
  - 読み出し:ディスク数が増えるほど小さく
  - 書き込み:ディスク数が増えるほど大きく
- 読み書きが混在するときは

$$\frac{1}{2}\times\text{平均位置決め時間}\times\text{データ転送速度}$$

RAIDの例(1)ーScale Array

- Thinking Machines社 Scale Array （1992）
  同社の並列計算機 CM-5 用 RAID-3，大規模数値計算用



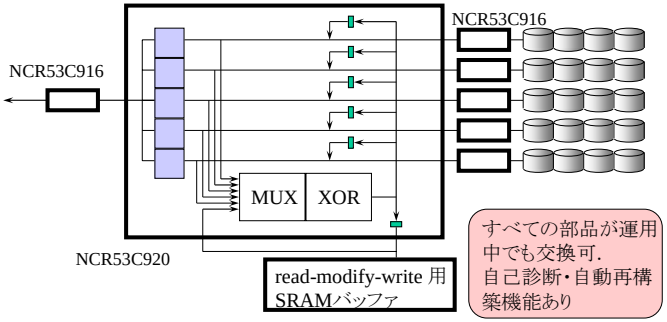
Scale Array（続き）

- 最小構成…
  - （(SCSI ディスク×2）×4）×3＝24ディスク
    22データディスク，1パリティ，1スペア
- プロセッサ間ネットワークにディスクアレイを直付け
  - 接続できるディスクアレイの数にほとんど制限なし
  - 最大120ディスク（＝5セット）までテスト済み
- ピーク性能（120台のとき）
  - 読み出し：185MB/sec
  - 書き込み：135MB/sec

RAIDの例（2）－NCR6298（1992年発表）

- NCR社 6298 ディスクアレイサブシステム…RAID-0, 1, 3, 5
  - 電源・冷却系も多重化
  - 専用コントローラで構成

商用システムなので，
耐故障性・保守性を重視

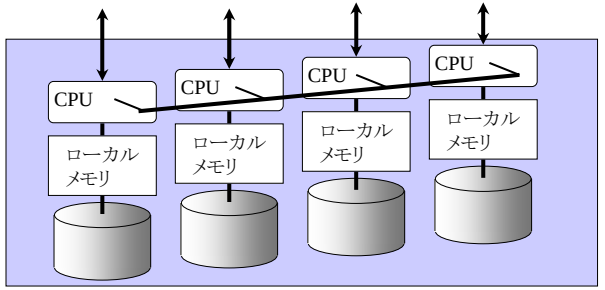


NCR6298（続き）

- RAID-5の書き込み以外の操作
  コントローラの各チャネルがロックステップ（lock-step）動作をするので，バッファは不要
- RAID-5の書き込み操作
  - 旧データ・旧パリティを読み込んでexclusive-OR を取り，SRAM バッファに格納する
  - ホストから新データが届くと，ただちに SRAMバッファのデータと exclusive-OR される
- 最大転送性能
  - 10MB/sec (RAID-0, 1, 5)
  - 14MB/sec (RAID-3)

RAIDの例（3）－TickerTAIP/DataMesh

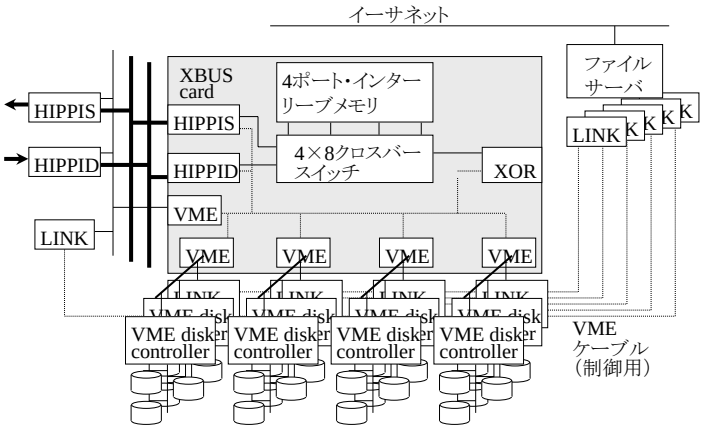
- Hewlett-Packard 研究所の プロトタイプ(1993年発表)
  - CPU付き「インテリジェントディスク」のアレイ
  - CPU間に内部接続ネットワーク
  - 複数の外部接続インタフェース



RAIDの例（4）－UCB RAID-II

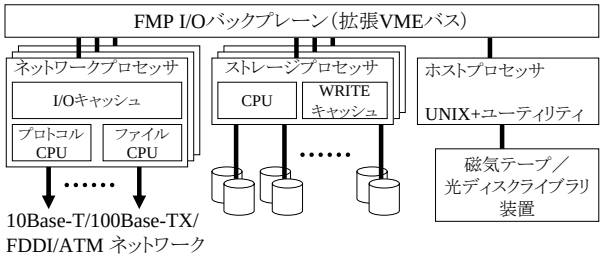
- UCBの ストレージサーバプロトタイプ（1993年発表）
  - SCSIディスクによるアレイを HIPPIインタフェース（100MB/sec）に接続
  - XBUSカードと呼ばれる専用ボードで実装
  - ファイルサーバは通常のワークステーションであるが，実際のデータ転送はサーバの遅いメモリを経由しない
    - サーバはVMEバスによりディスクをコントロールする
    - データの転送は XBUS - HIPPI インタフェース経由で直接行う

UCB RAID-II（続き）



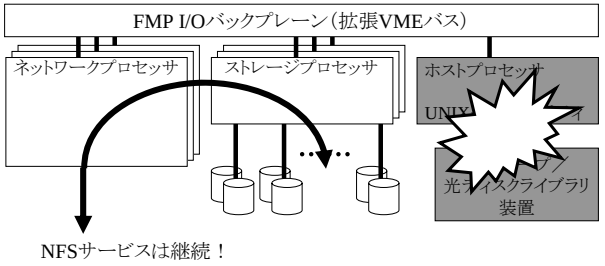
RAIDの例（5）－Auspex Netserver

- Auspex社NetServerファミリ
  - RAIDのデータを，NFS (Network File System) により複数の計算機で共有するためのファイルサーバ専用機
  - RAIDコントローラ，NFSサーバ，バックアップサーバをそれぞれ専用プロセッサで実装 (Functional Multiprocessing)



Auspex Netserver（続き）

- Data Guard
  - ホストプロセッサのUNIXがクラッシュしても，NFSサービスは継続可能になるようなオプションソフトウェア



Auspex Netserver（続き）

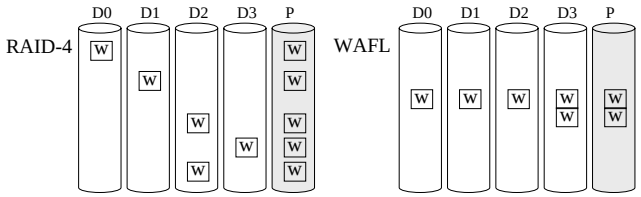
- Drive Guard
  - RAID 5のサポート（パリティのみハードウェアで演算）
  - Normal/Degraded/Rebuildモード
  - 1つのストレージプロセッサ内で複数のレベルのRAIDを混在させることができる
- NeTservices
  - UNIX, Windows のファイルを一元管理
    - NFS，CIFS (Common Internet File System) プロトコル＊の両方をサポート
      ＊CIFS…Windows で用いられるファイルシェアリングプロトコル
    - 同一ディレクトリ内でUNIX/Windowsファイルの混在も可能

RAIDの例（6）－NetApp Multiprotocol Filer

- Network Appliance 社 Multiprotocol Filer
  - NetServerと同様の専用機
    ただし，単一プロセッサで，UNIXも搭載せず
  - NFS, CIFS, HTTP (Hypertext Transfer Protocol) の3種類のプロトコルをサポート
  - RAIDには珍しく，レベル4のみサポート
    …運用中でもディスク追加による拡張が可能
- WAFL (Write Anywhere File Layout)
  - UNIXファイルシステムと同様のファイルシステム
  - ただし，メタデータ（データブロックの所在を記録するデータ）をファイルに格納し，これ自体もディスク上の任意の場所に格納できるようにしている。
  - NVRAMにNFSリクエストをログとして記録

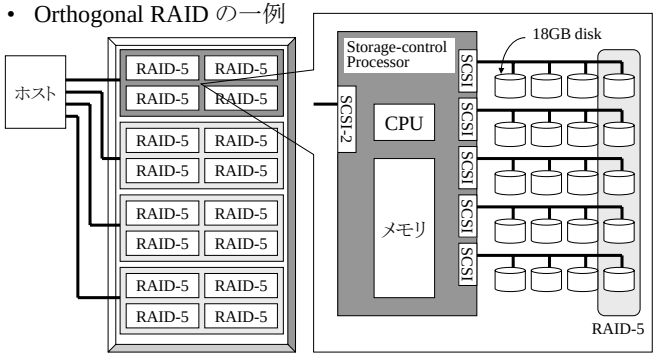
NetApp Multiprotocol Filer（続き）

- WAFLとRAID-4
  - RAID-5が考案された理由は，RAID-4ではRead-Modify-Write操作が多数あるとパリティディスクがボトルネックになるためだった。
  - WAFLでは，Read-Modify-Writeを行わない
    - 数百のリクエストを，近接するいくつかのフルストライプにまとめて書きこみ，ディスクアクセス回数を減らす。



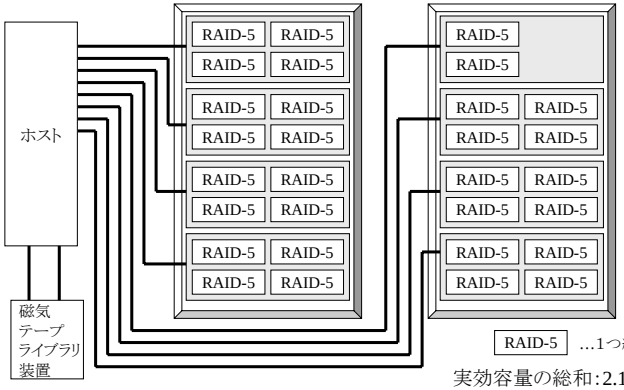
RAIDの例（7）－SiliconGraphics Challenge RAID

- Orthogonal RAID の一例



SiliconGraphics Challenge RAID（続き）

- 九州大学情報基盤センターでの構成例



実効容量の総和：2.1TB

RAIDの例（8）－Sun Microsystems StorEDGE



ディスクアレイのまとめ（1）

- RAIDの重要性
- ハードディスクの基礎知識
- RAIDのパターン
  - Level 0 (Non-Redundant)
  - Level 1 (Mirrored)
  - Level 2 (Memory-Style ECC)
  - Level 3 (Bit-Interleaved parity)
  - Level 4 (Block-Interleaved Parity)
  - Level 5 (Block-Interleaved Distributed Parity)

  Level 0+1, Level 6 といったカテゴリもある

ディスクアレイのまとめ（2）

- RAIDの比較
  - RAID-0 と RAID-1
  - RAID-2 と RAID-3
  - RAID-3 と RAID-5
    - 大きな読み出し，小さな読み出し
    - 大きな書き込み，小さな書き込み
- RAIDの信頼性を左右する要因
  - 2台のディスクの故障
  - システムクラッシュ
  - 訂正不能なビットエラー
- MTTDL (Mean Time to Data Loss)

ディスクアレイのまとめ（3）

- RAIDの信頼性を上げるための方策
  - スペアディスク
  - 状態情報
  - Orthogonal RAID
- RAIDの性能を上げるための方策
  - バッファリングとキャッシング
  - フローティングパリティ
  - パリティロギング
  - パリティデクラスタリング，
  - 分散スペアリング，パリティスペアリング
  - 仮想ストライピング

ディスクアレイのまとめ（4）

- 最適なストライピングユニットサイズの選択
  - 大きくするか，小さくするか，トレードオフの問題
  - 解析的に求めるのが難しく，今後の課題となっている
- RAIDの例
  - Thinking Machines 社 Scale Array
  - NCR社 6298
  - Hewlett-Packard研究所 TickerTAIP/DataMesh
  - UCB　RAID-II
  - Auspex NetServer
  - NetApp Multiprotocol Filer
  - SiliconGraphics Challenge RAID
  - Sun Microsystems StorEDGE

さて，ディスクアレイの次のお題は…

- ディスクアレイ，RAID
  ⇒ディスクを複数台並べて，これを並列に駆動する仕組み
  非並列のアプリケーションからの要求を処理するための仕組み
- それでは…
  - 並列計算機が複数のディスクを持っているときにはどうしたらよいのか？
  - 並列プログラムに大量のデータを高速に供給するためにはどうしたらよいのか？

並列ファイルシステム
並列入出力機構

次のお題は

- **並列ファイルシステム**
  - 並列アプリケーションプログラムに大容量のデータを読み書きさせるためのファイルシステム
- **並列入出力機構**
  - 並列アプリケーションプログラムの各プロセスが同時に読み書きを行うための入出力機構（インタフェース，ライブラリ）
- しかし，その前に，並列計算機とはどんなものか，簡単に見ておくことにしよう
  - Flynn の分類
  - 共有メモリ型，分散メモリ型
  - 超並列計算機
  - SMPとSMPクラスタ

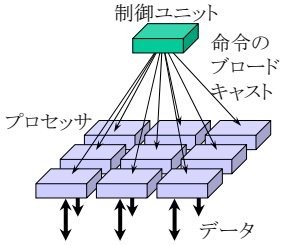
並列計算機のアーキテクチャ（1）

- Flynnによる分類(1966)
  - データと命令の流れによる有名な4つの分類
  - 簡潔ではあるが，すべての計算機がこれで分類できるかどうかは，意見が分かれるところでもある

		data stream	
		single	multiple
instruction stream	single	SISD	SIMD
	multiple	MISD	MIMD

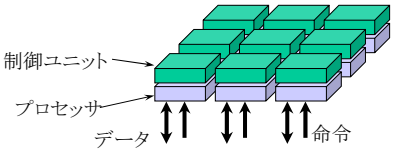
並列計算機のアーキテクチャ（2）

- SISD (Single instruction stream, single data stream)
  - 通常の 1CPUの計算機
  - 命令パイプラインを持つスカラープロセッサ（RISCチップ）をこれに含める考え方もある
- SIMD (Single instruction stream, multiple data stream)
  - 単一の制御ユニットから全プロセッサに命令をブロードキャスト
  - 各プロセッサは1命令（または1クロック）ごとに同期をとりながら実行
  - ベクトルプロセッサをこの分類に含める考え方もある



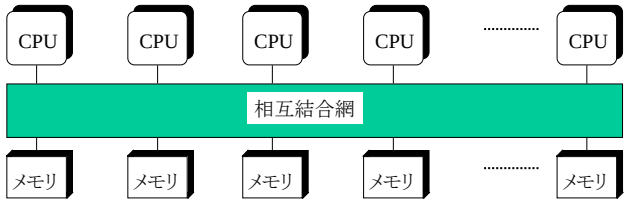
並列計算機のアーキテクチャ（3）

- MISD (Multiple instruction stream, sigle data stream)
  - 多重命令流・単一データ流
  - 用語としては存在するが実在しない，という考え方と，ベクトルプロセッサをこれに含める考え方とがある
- MIMD (Multiple instruction stream, multiple data stream)
  - 各プロセッサに制御ユニットが付属
  - 各プロセッサは独立に動作（別の命令を実行できる）



並列計算機のアーキテクチャ（4）

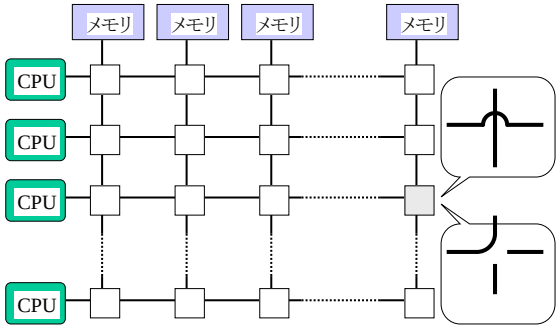
- **共有メモリ型並列計算機**
  multiprocessor と呼ばれることもある



  - 相互結合網にはいろいろな機構がありうる
    - バス
    - クロスバースイッチ

クロスバースイッチ

- プロセッサどうし，プロセッサとメモリを接続する



並列計算機のアーキテクチャ(5)

- **分散メモリ型並列計算機**
  multicomputer と呼ばれることもある

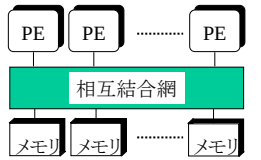
  - 相互結合網にはいろいろなトポロジがありうる
    - オメガ網
    - ハイパーキューブ
    - メッシュ
    - トーラス

超並列計算機

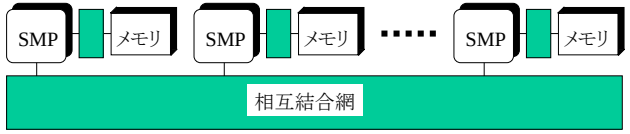
- 「超」がつくほど多数のプロセッサを搭載した計算機
  - 「超」の厳密な定義はないが，最近では1024台くらいのプロセッサを搭載したものも珍しくない
  - そうなると，CPU(central processing unit)という呼称はあまりなじまないので，1つ1つのプロセッサを**PE(processing element)**と呼ぶこともある
- スケーラビリティ(scalability，規模適応性)
  - プロセッサの数やメモリの容量を増やすのが容易なこと
  - 予算の制約を除き，搭載できるプロセッサ数に制限が(あまり)ないこと

SMPとSMPクラスタ

- SMP (symmetric multiprocessor)
  - 共有メモリ型並列計算機の一種
  - すべてのPEが対等
  - メモリのどこも同等にアクセス可

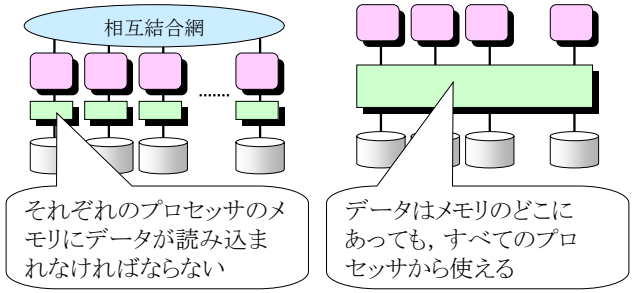


- SMPクラスタ
  - SMPを分散メモリ型並列計算機のようにつないだもの



並列計算機におけるディスク入出力の問題点

- 分散メモリ型の並列計算機におけるディスク入出力
  - 共有メモリではないので，それぞれのメモリにデータを読み込む必要がある

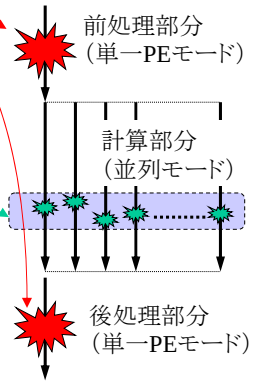


並列入出力の必要性

- 分散メモリ型並列計算機
  - 多数の**PE(processing element)**と大容量の主記憶によって大量のデータを用いた大規模な計算を行うことができる
  - 複数のディスクを搭載し，それを並列に駆動できる
- 高度な科学技術計算では，計算の精度をあげようとすると必要なデータの量が急激に増大することが多い
  - そのようなプログラムは，計算時間を短縮するため，並列化されることが多い
  - 並列プログラムにディスク入出力を行わせる必要が出てくる

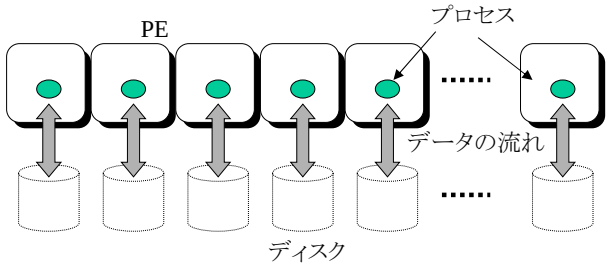
並列プログラムのデータ入出力

- 単一のPEが代表で入出力を行う
  ⇒大容量のデータの入出力には時間がかかる
- 複数のPEが同時に入出力を行う
  ⇒プログラマが制御するのは面倒



並列入出力のイメージ

- 複数のPE上で動作している各プロセスが，自分の必要とするデータを並列に入出力
- あたかも，それぞれがローカルディスクにアクセスしているかのように(物理的にはそうでなくてもよい)



並列入出力のためのさまざまな手法

- 並列ファイルシステム
  - どちらかというと，複数のディスクをまとめて1つのファイルシステムのイメージを提供
- 並列入出力ライブラリ
  - どちらかというと，ディスクはバラバラで，それを並列に使うイメージを提供
- 並列プログラミングライブラリの入出力インタフェース
  - MPI (message passing interface) のようなライブラリのなかの並列入出力インタフェース