1. 並列処理機構

並列プログラミングの概念
並列コンピュータシステム
並列処理による速度向上

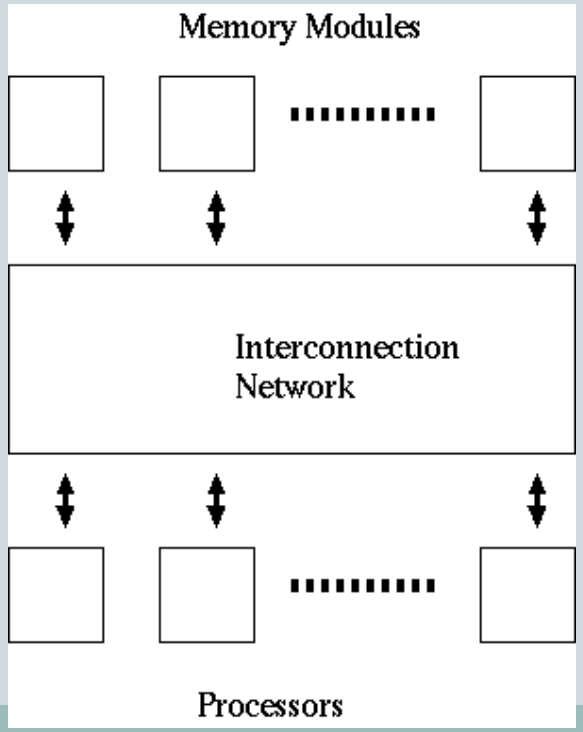
1.1 計算速度の需要

- 計算速度
  - VLSIアーキテクチャの進化によって計算速度が飛躍的に向上
  - 応用問題も飛躍的に複雑化
  - 常に計算速度向上の要望
- 並列処理
  - 速度向上の一つの方法
  - 一つのプロセッサの速度向上には限界
- 用語
  - 並列プログラミング
    - 並列処理のためのプログラムを書くこと
  - 並列コンピュータ
    - 並列処理を行うための計算プラットフォーム

1.2 並列コンピュータの種類

共有記憶マルチプロセッサシステム

- 共有記憶マルチプロセッサシステム
  - 単一プロセッサモデルの自然な拡張
  - 複数のプロセッサが単一アドレス空間のメモリ装置にアクセスできる



マルチコア技術では
プロセッサコアと呼ぶ

1.2 並列コンピュータの種類

- 並列プログラミング
  - 特殊な並列構文と共有変数等を持つ並列言語を利用する。
  - コンパイラが並列処理用の実行可能コードを生成する。
  - 指示子を用いて並列処理可能部分を指定し、暗黙的にスレッドを割当る。
  - スレッドを明示的に定義し各プロセッサに割当てる。

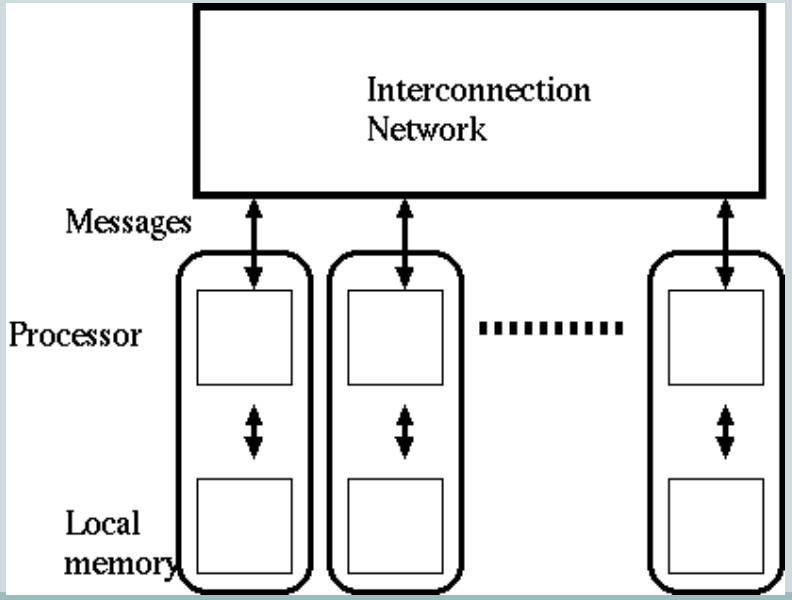
- 特徴
  - メッセージ通信型に比べてプログラミングが比較的容易である。
  - ハードウエアの構成が困難である(大規模化が困難)。
  - 階層構造による工夫がある。

マルチコア技術の革新によりコア数の増加

1.2 並列コンピュータの種類

メッセージ通信マルチコンピュータ

- メッセージ通信マルチコンピュータ
  - 複数のコンピュータを結合網によって結合したシステム
  - 各コンピュータは自分のアドレス空間を有する。
  - プロセッサは自分のメモリのみアクセスできる。
  - 情報のやりとりはメッセージ通信。



1.2 並列コンピュータの種類

- 並列プログラミング（分散メモリプログラミング）
  - 逐次プログラムにメッセージ通信用のライブラリルーチンを挿入する。
  - 問題を分割し、部分問題を一つのプロセスとしてプログラミング。
  - プロセスを各コンピュータに割当てる。
  - 一台のコンピュータに割当てられた複数のプロセスは時分割処理。
  - 異なるコンピュータに割当てられたプロセス間通信はメッセージ通信。

- 特徴
  - 大規模システムを構築しやすい。
  - ネットワーク接続されたワークステーション(PC)も利用可能。
  - プログラミングは共有記憶パラダイムと比較して困難である。

1.2 並列コンピュータの種類

MIMDとSIMDの分類

- Flynnの分類
  - SISD(Single Instruction stream-Single Data stream)
    - 単一プロセッサのコンピュータ
  - MIMD(Multiple Instruction stream-Multiple Data stream)
    - 各プロセッサは別のプログラムをもち、異なるデータに対して作用する。
  - SIMD(Single Instruction stream-Multiple Data stream)
    - 一つのプログラムの命令が複数のプロセッサに放送され、それぞれのデータに対して同一の処理が行われる。
    - 応用範囲は広い。例えば画像処理。

1.3 メッセージ通信マルチコンピュータのアーキテクチャ

静的結合網によるメッセージ通信マルチコンピュータ

- 静的結合網
  - ノード間に固定された物理的リンクをもつ。
  - 各ノードは、プロセッサとメモリと他のノードへのリンクを持つ。
- リンク
  - 線が一本で情報を一ビットずつ転送する。
  - ワードの各ビット毎に線を準備しワード全体を同時に転送する。
- 結合網の評価基準
  - ネットワークバンド幅：単位時間に転送できるビット数で、bit/秒。
  - ネットワークレイテンシー：メッセージを送信する時間。
  - 通信レイテンシー：ソフトウエアオーバーヘッド、インターフェイス遅延も含めたものをいう。
  - 直径：2個のノード間の経路中の最小リンク数。メッセージ遅延に大きく影響する。

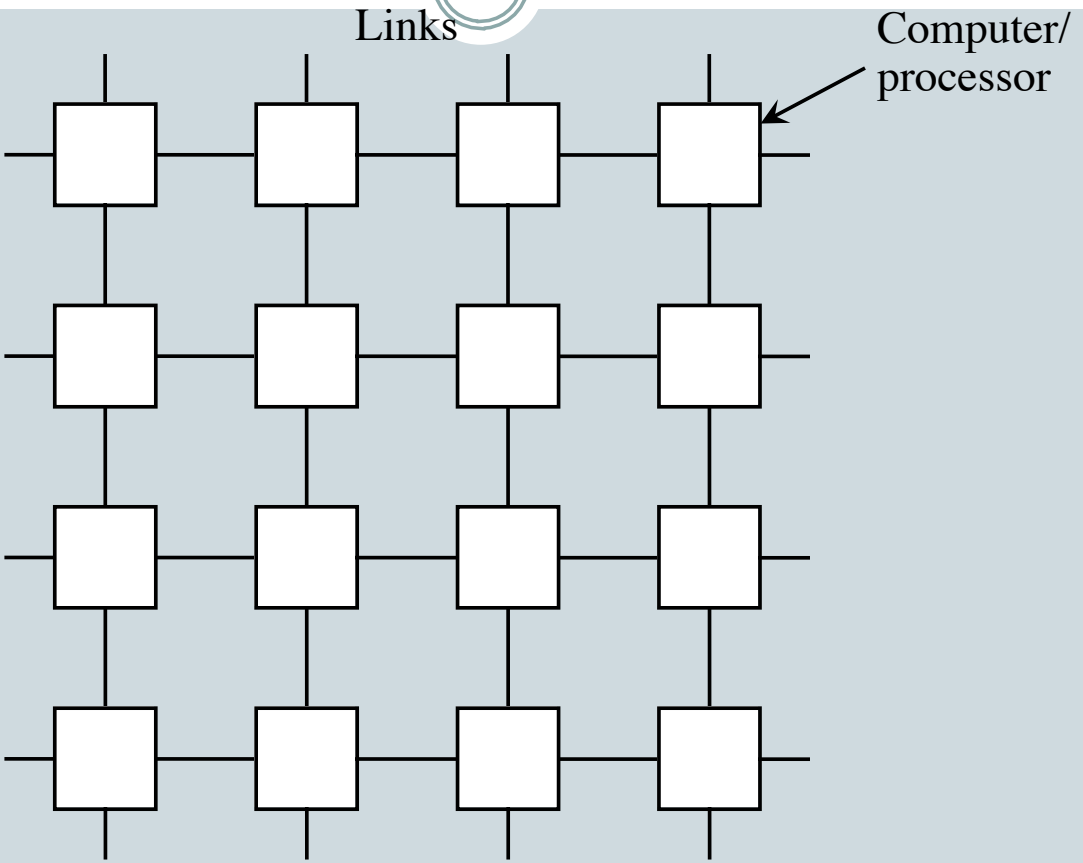
1.3 メッセージ通信マルチコンピュータのアーキテクチャ

- 完全結合網
  - 各ノードは他の全てのノードに対してリンクを持っている。
  - nノードの場合、各ノードはn-1本のリンクを持つ。全体でn(n-1)/2本。
  - nが小さいときは理にかなっているが、nが大きくなると非現実的。

- ライン/リング
  - ライン(line)の両端を閉じればリング(ring)となる。
  - 各ノードは2個の隣接ノードとリンクで結ばれている。
  - ラインの直径はn-1で、リングは[n/2]となる。
  - ルーティング（経路制御）が簡単。

- 二次元アレイ（メッシュ）
  - ¥sqrt{n} X ¥sqrt{n}のメッシュの直径は2(¥sqrt{n}-1)である。

Two-dimensional array (mesh)



Also three-dimensional - used in some large high performance systems.

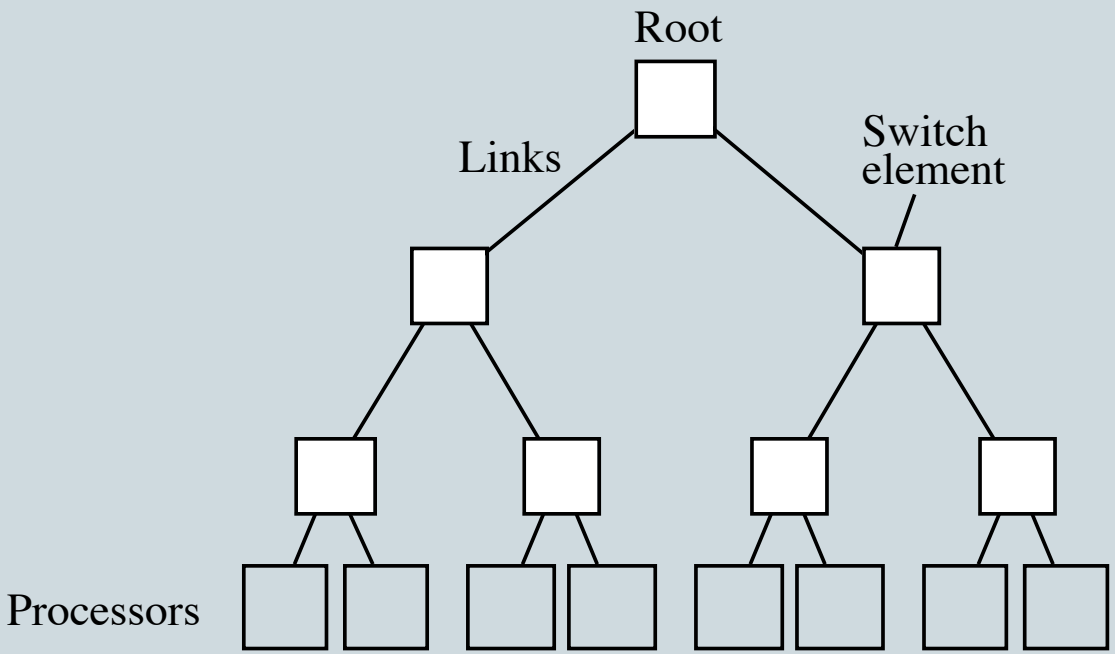
1.3 メッセージ通信マルチコンピュータのアーキテクチャ

- トーラス
  - メッシュの端を反対側とつなぐとできる。
  $\sqrt{n} \times \sqrt{n}$ のトーラスの各ノードは4本のリンクを持つ。
  - 全体では2n本。

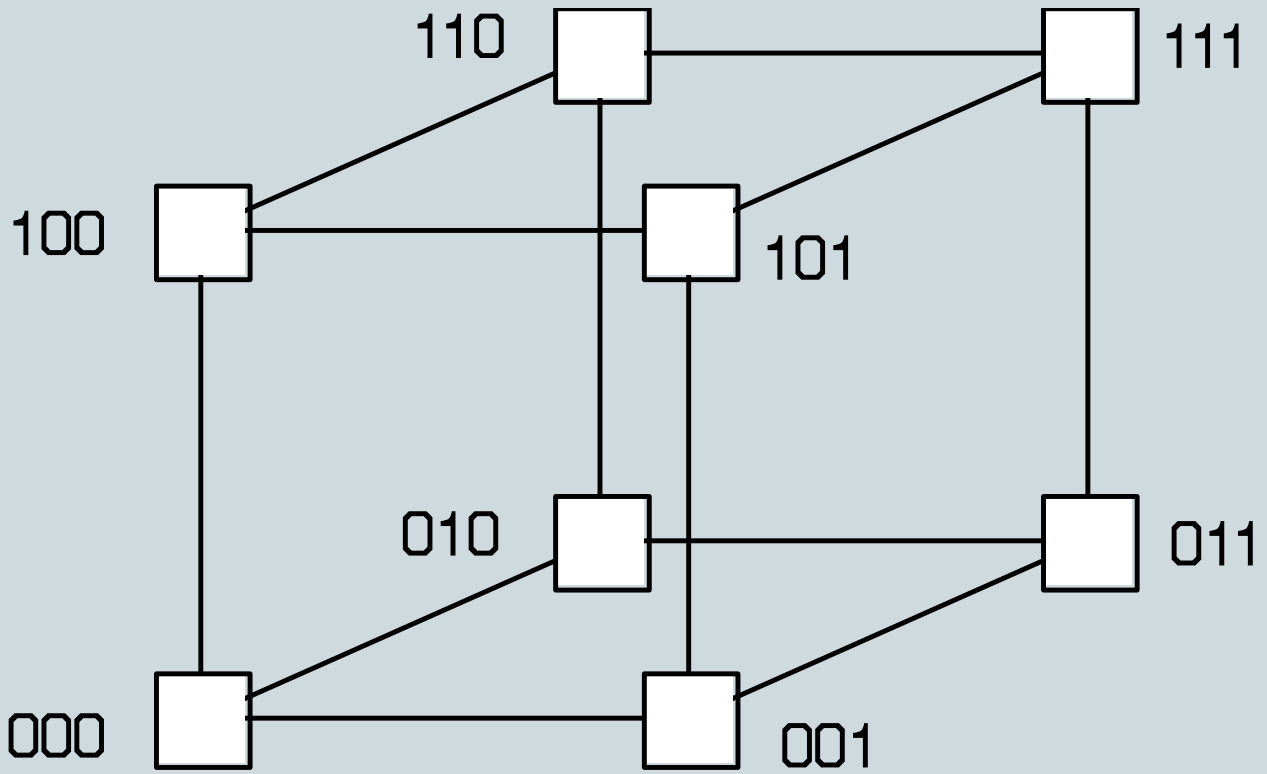
- ツリー網（2分木網）
  - 最初のノードはルートと呼ばれる。
  - 各ノードはその下の2個のノードとリンクを持つ。
  - 分割統治型アルゴリズムと相性がよい。

- ハイパーキューブ網（d次元の(2進）ハイパーーキューブ網）
  - 各ノードはネットワークのそれぞれの次元の1個のノードと結合。
  - d次元であれば、各ノードにdビットの2進アドレスを割当て、1ビットのみ異なるノード間にリンクを繋ぐ。
  - 直径は $\log_2 n$ 。

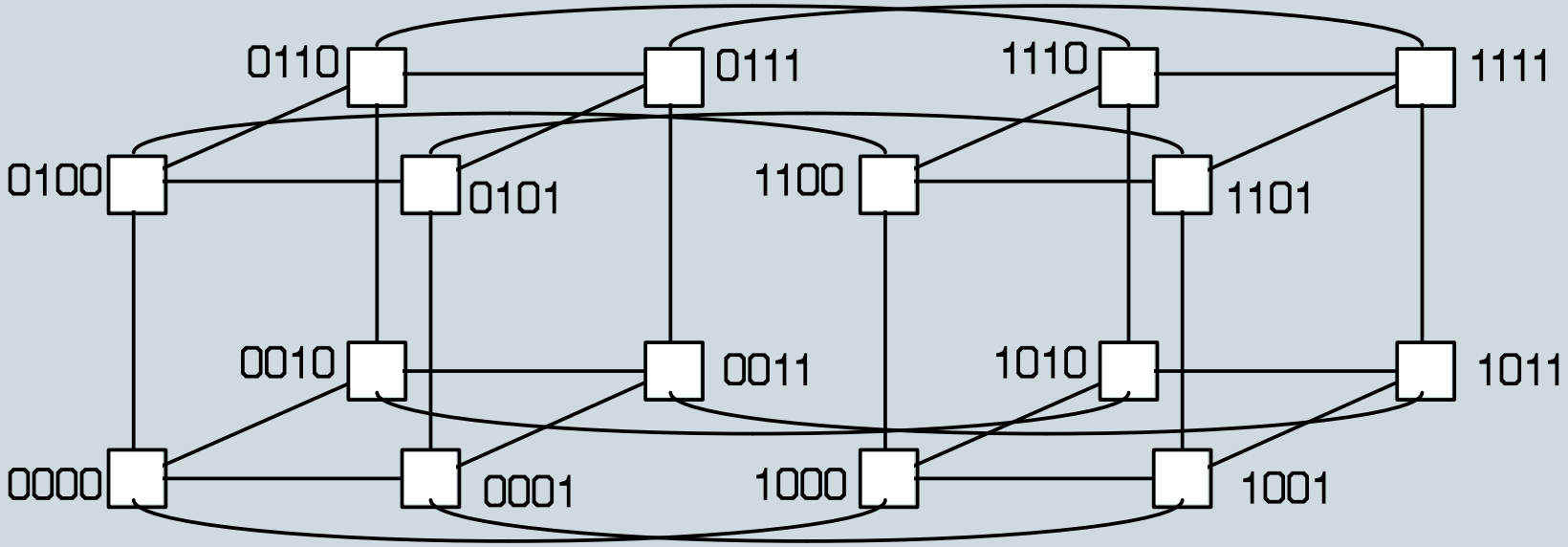
ツリー網（Tree）



三次元ハイパーキューブ
Three-dimensional hypercube



Four-dimensional hypercube



Hypercubes popular in 1980's - not now

1.4 計算速度向上の可能性

- プロセス
  - 逐次に処理されるプログラムの単位で、大きさは粒度(granularity)と呼ばれる。
  - 粒度が大きい
    - プロセス生成とプロセス間通信のコストが小さくなる。
  - 粒度が小さい
    - 並列度はあがるが、プロセス生成とプロセス間通信のコストが大きくなる。
  - 並列コンピュータのアーキテクチャに応じて、最適な粒度を決定する必要がある。
- 速度向上度(加速率)S(p)
  - 単一プロセッサシステムに対する相対性能の尺度
    - S(p) =(1プロセッサによる実行時間)/(pプロセッサによる実行時間)
    - S(p) = t_s/t_p
    - 通常は、S(p) < p

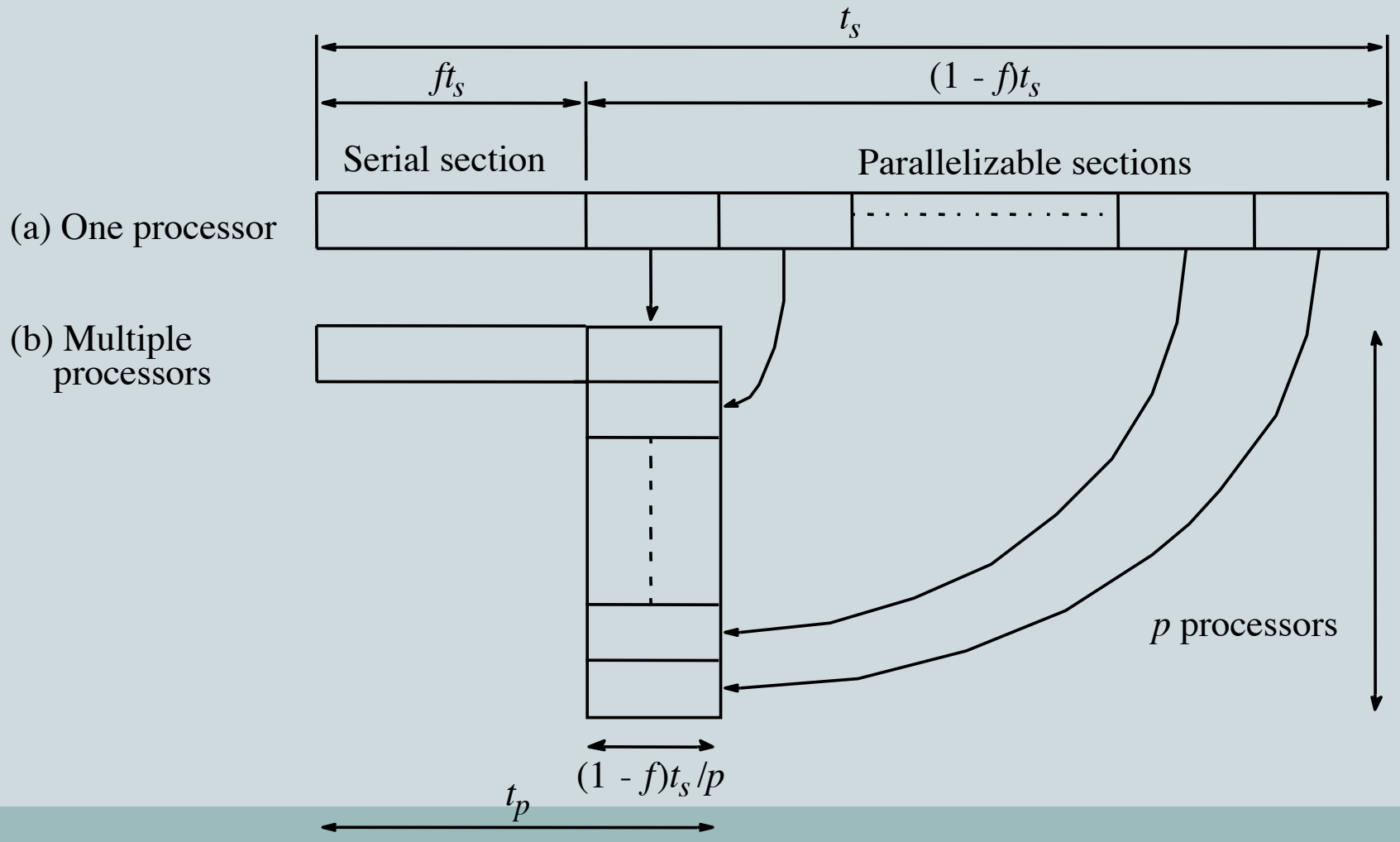
1.4 計算速度向上の可能性

- オーバーヘッド
  - 一部のプロセッサのみしか有用な仕事をしておらず、残りはアイドルである期間
  - 定数を局所的に再計算する場合、逐次計算には現れない余分な計算。
  - メッセージ送信のための通信時間。

- 最大速度向上度はp
  - S(p) = t_s/t_p = p
  - だけど通常はそうはならない-> アムダールの法則

- 超線形速度向上(スーパーリニア): S(p) ＞ p
  - メモリ, キャッシュの効果
  - 探索アルゴリズムでは可能性有り

最大速度向上率 Amdahl's law



Amdahl's law（アムダールの法則）

Speedup factor is given by:

$$S(p) = \frac{t_s}{ft_s + (1-f)t_s/p} = \frac{p}{1 + (p-1)f}$$

This equation is known as Amdahl's law

Speedup against number of processors