コンピュータ科学特別講義Ⅳ

Parallel Algorithm Design（#4）

Masato Edahiro

June 1, 2018

Please download handouts before class from
http://www.pdsl.jp/class/utyo2018/

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.

PDSL

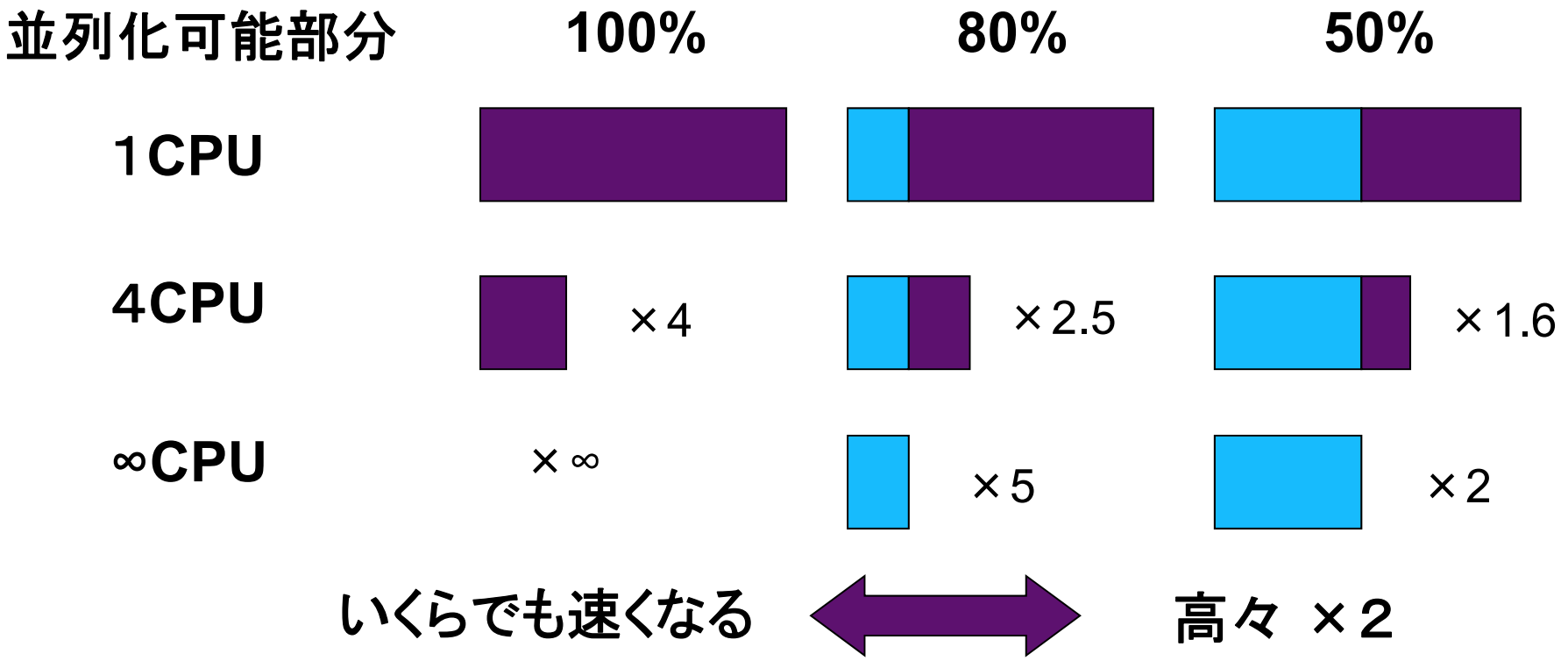
Contents of This Class

- Our Target
  - Understand Systems and Algorithms on “Multi-Core” processors
- Schedule (Tentative)
  - #1 April 6 (=Today) What’s “Multi-Core”?
  - #2 April 13 : Parallel Programming Languages (Ex. 1)
  - April 20, 27, May 4, 11, 18: NO CLASS
  - #3 May 25 : Parallel Algorithm Design
  - #4 June 1 (Fri) : Laws on Multi-Core
  - #5 June 8 : Examples of Parallel Algorithms (1) (Ex. 2)
  - June 15: NO CLASS
  - #6 June 22 : Examples of Parallel Algorithms (2)
  - #7 June 29 : Examples of Parallel Algorithms (3)
  - #8 July 6 : Examples of Parallel Algorithms (4)
  - #9 July 13 : Examples of Parallel Algorithms (5) (Ex. 3)
  - (July 20)

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

Amdahlの法則

- プログラムの中で、並列化可能部分の割合が大きくなければ意味がない



Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

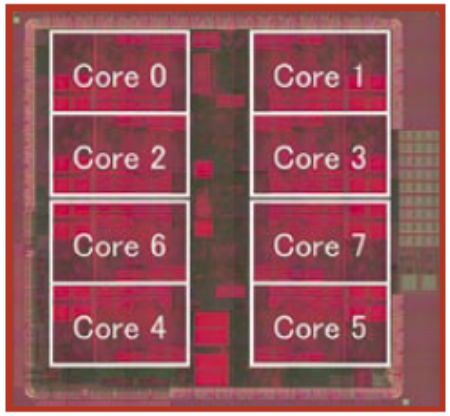
パレートの法則（80:20の法則）

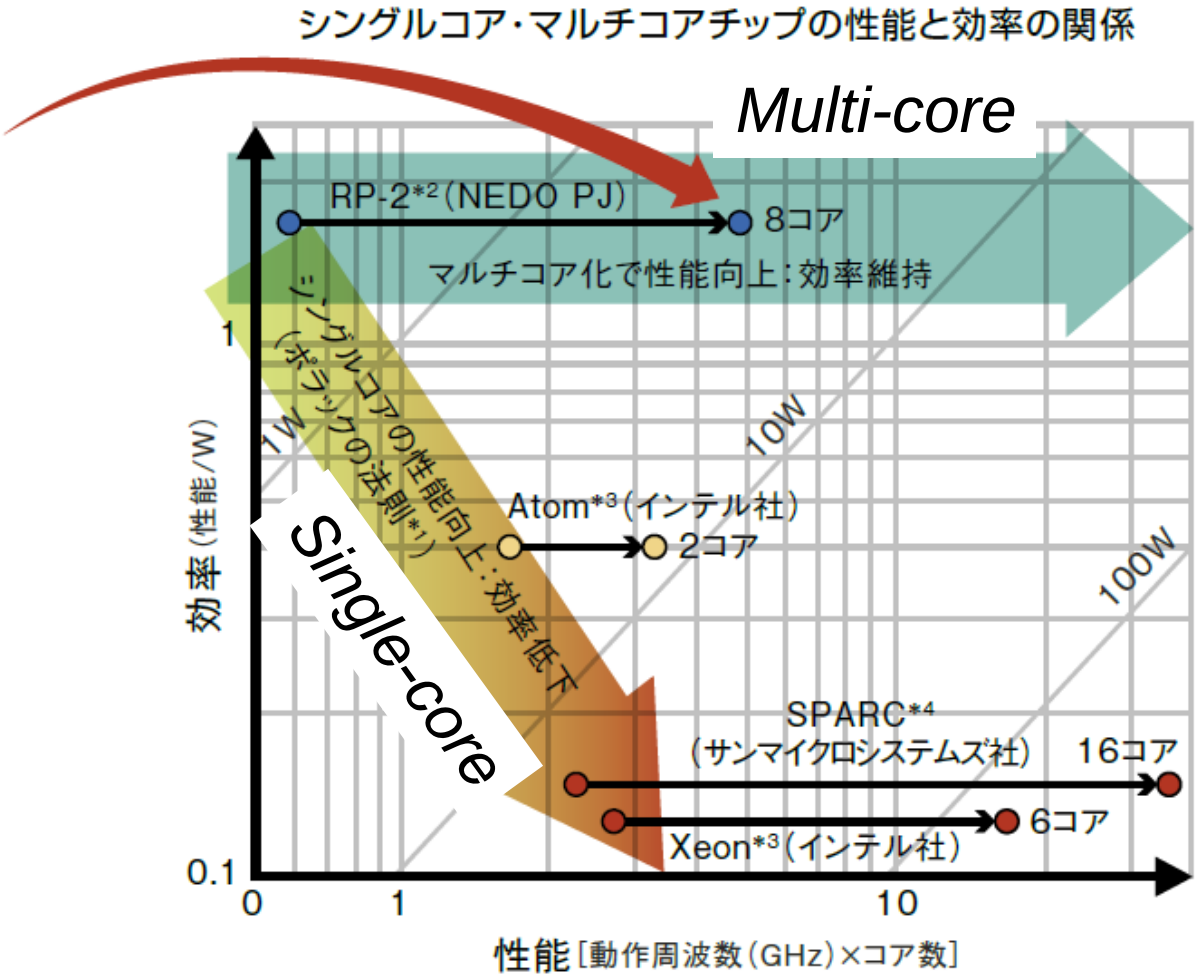
- 「プログラムの処理にかかる時間の80%はコード全体の20%の部分が占める」という経験則
- 注：パレートの法則自体は世の中の様々な現象に使われる法則であり、プログラムは一例

- Amdahlの法則において、プログラム実行の80%を並列化したいとき、パレートの法則に則っていればコードの20%に関する並列化検討をすればよいことになる

Pollackの法則

- プロセッサの性能はその複雑性の平方根に比例する（経験則）

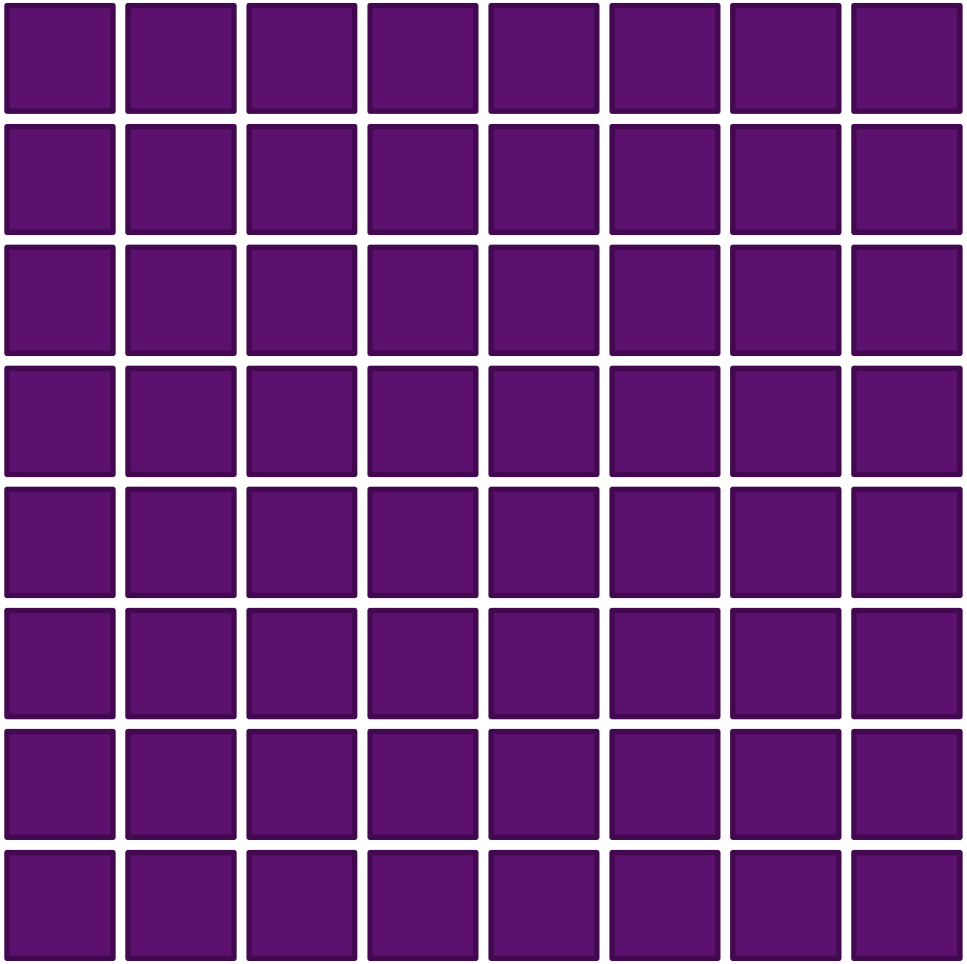




メニーコア向けAmdahlの法則の拡張

- 並列化可能部分はメニーコア実行がよいが、並列化不可能部分は高性能な単一コア実行がよい
- メニーコアの３シナリオ
  - 対称型
    - すべてのコアが同じ能力を持つ（通常のAmdahlの法則と同じ）
  - 非対称型
    - 一つの高性能コアと複数の軽いコア。ただし命令セットはどちらのコアも同じ（同じバイナリが動く）
  - 動的変更型（現在の所理想論）
    - あるときには高性能コア、あるときにはメニーコア
- ４個のコアと同じ面積で４倍性能の単一コアができるわけではない
  - *Perf(r)*: r個のコアと同じ面積を単一コアで使ったときに出せる性能→ここではsqrt(r) （Pollackの法則）

メニーコア（対称型）

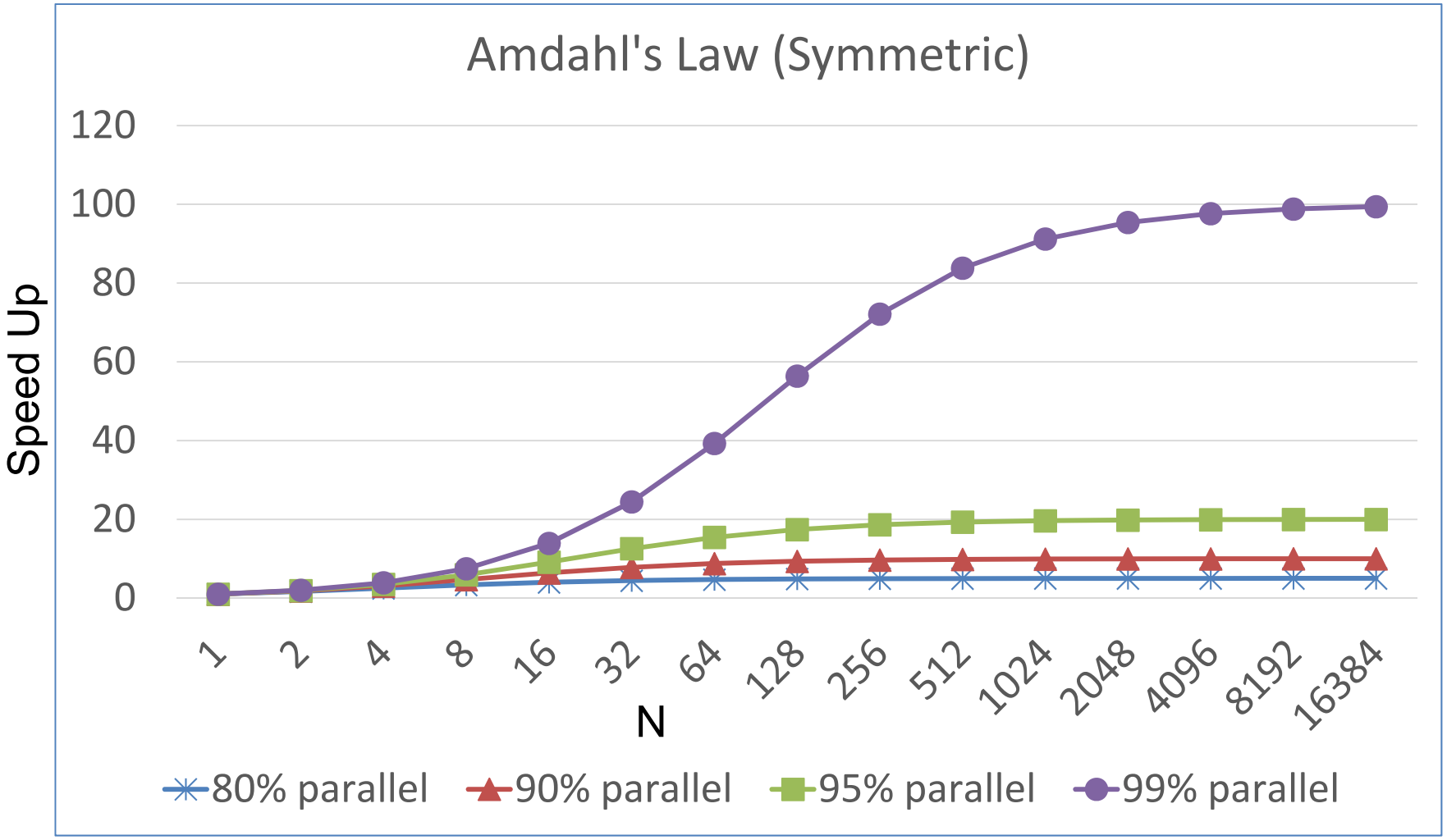


Amdahlの法則（対称型）

- P: 並列化可能な割合
- N: プロセッサ数
- Speed Up?

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

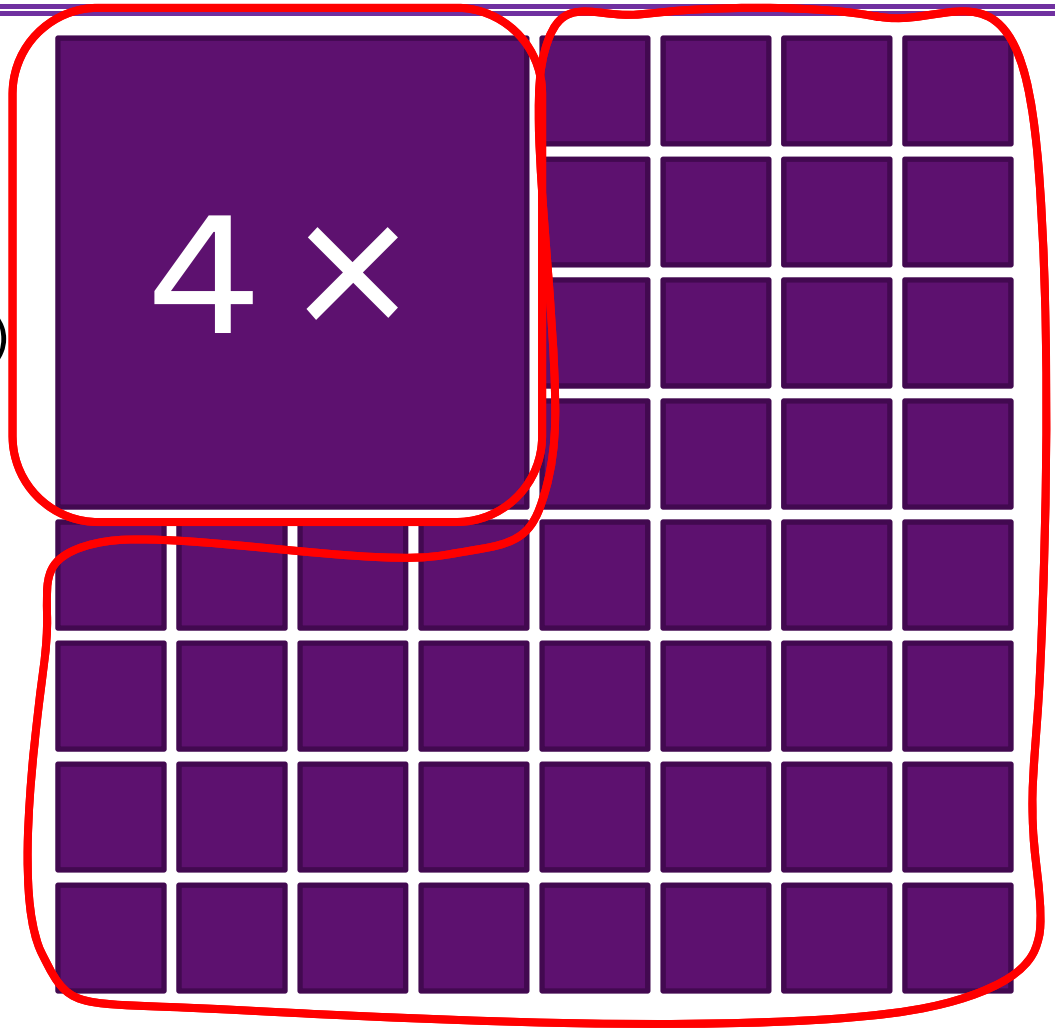
アムダールの法則により飽和してメリットがなかなか活かせない



非対称型（逐次処理のために高速CPUを）

Perf(r) x 1

(Use for all part)



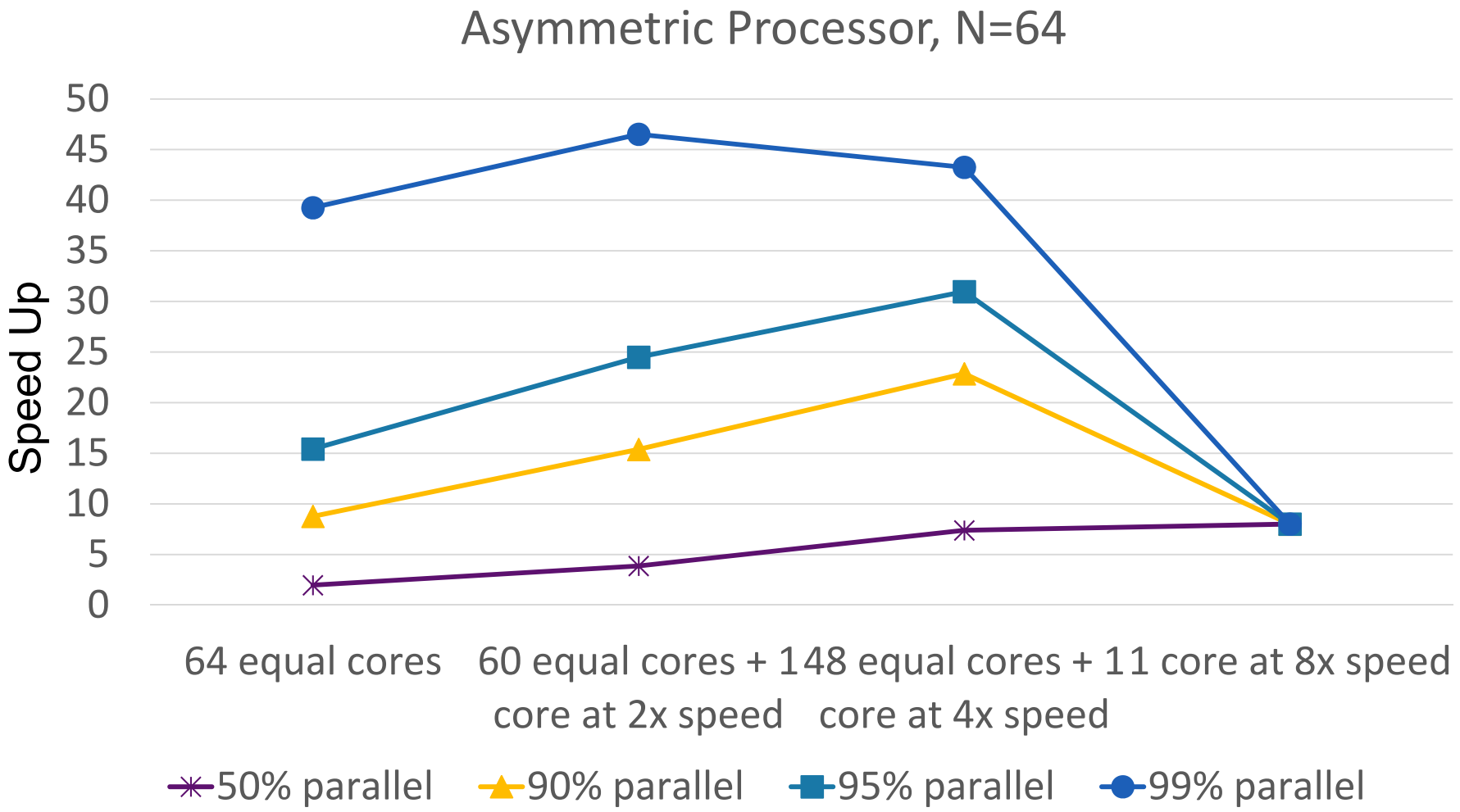
1 x (N – r)

(only for parallelizable part)

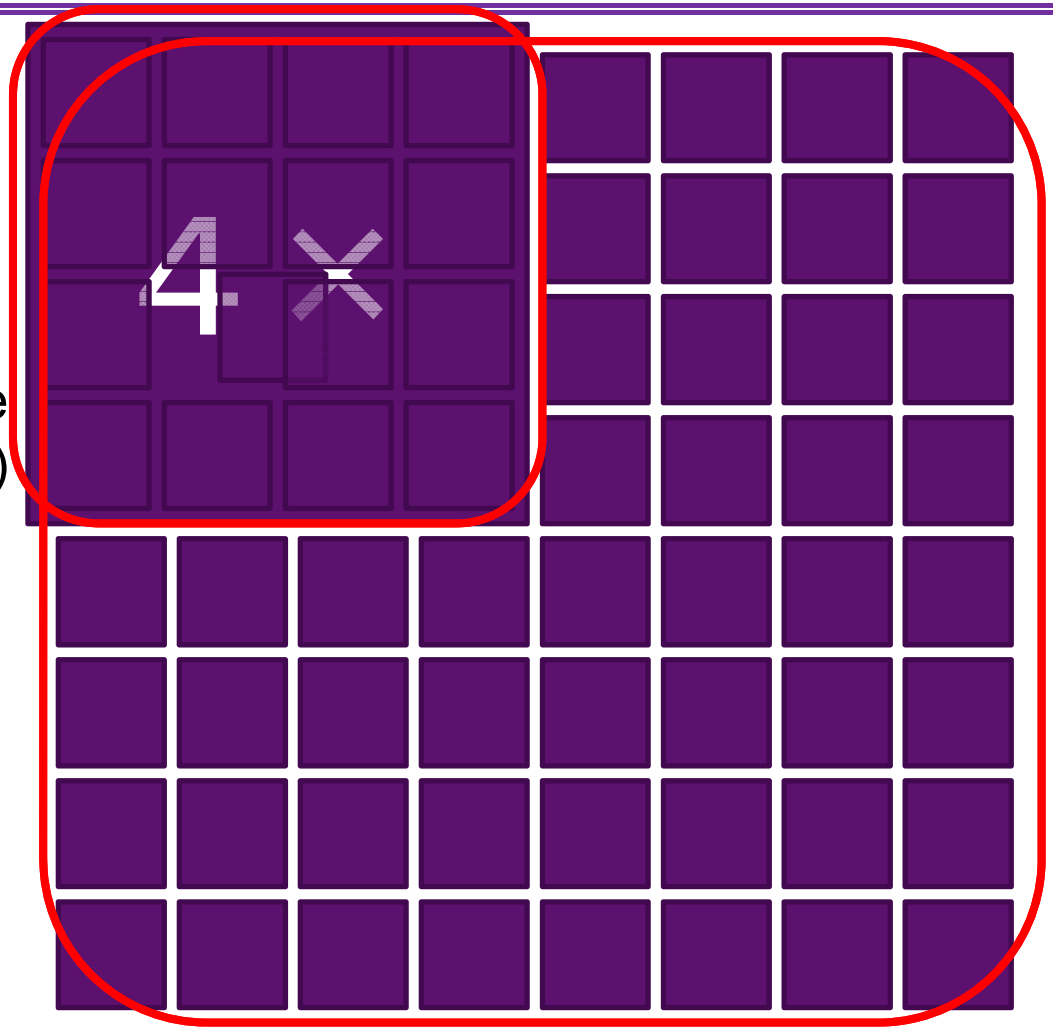
Speed Up?

Speed Up?

性能が最も高くなる構成がアプリ依存になる



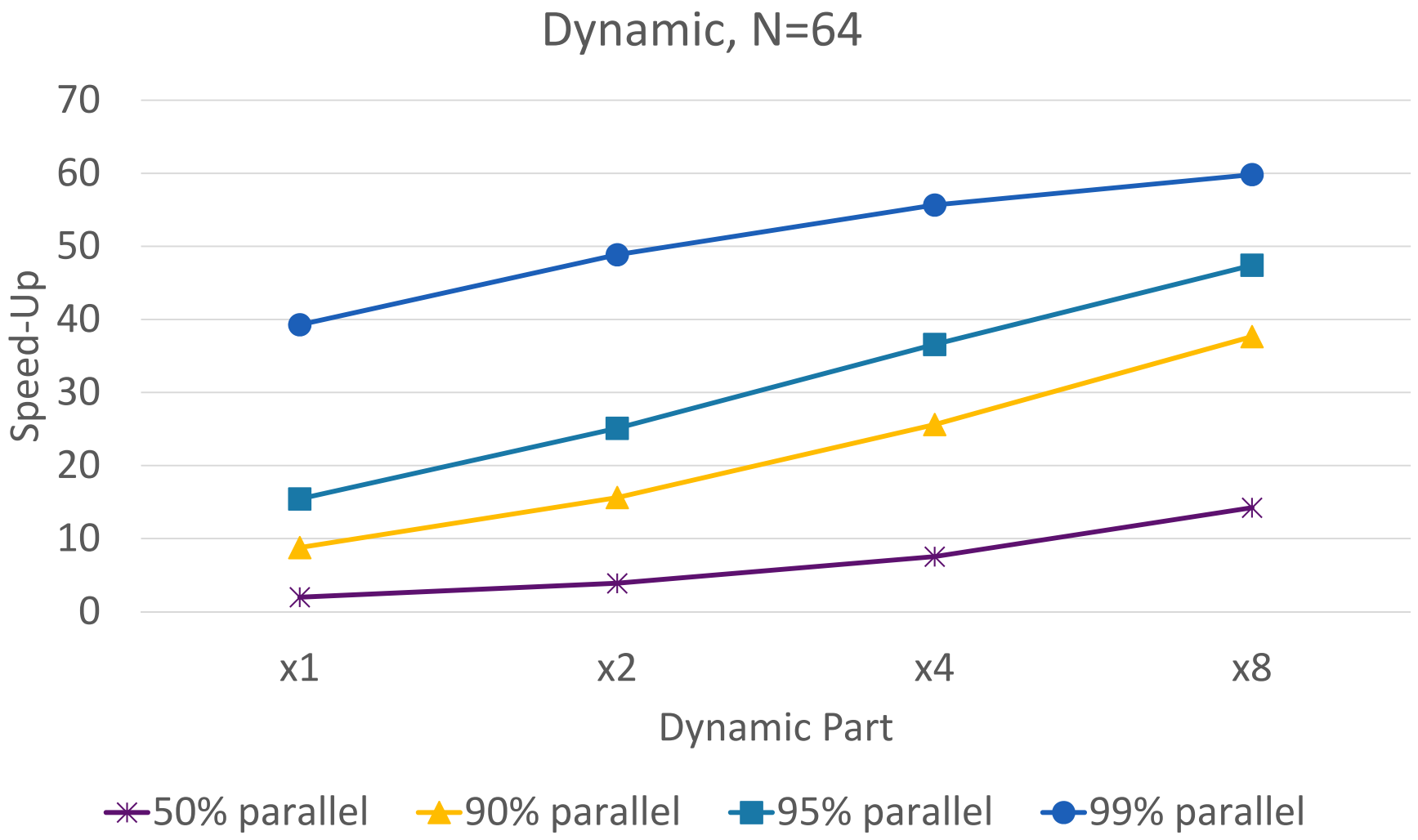
動的に変えられるようにすると、、、



Speed Up?

Speed Up?

スケーラビリティがある



Guntherの予想

- 競合や干渉を考慮。性能向上C(N,s,k)は、
  C(N,s,k) = N / (1 + s * (N-1) + k * N * (N-1))

- パラメータ
  - s: 競合パラメータ (プログラムの逐次部分など。0から1の間, Amdahlの法則と同じ意味)
  - k: 干渉パラメータ（同期オーバーヘッドなど、本来のアルゴリズムとは関係のないコスト）

- s=0, k=0：理想的な場合
- s>0, k=0：Amdahlの法則と同様
- kが０でないとき、Nの最適値がある (NLimit)（＝性能向上が飽和する）

Graduate School of Information Science, Nagoya Univ.
Parallel & Distributed Systems Lab.
PDSL

Karp-Flattの指標

- （対称型アーキテクチャ向け）Amdahlの法則から、並列化不能部分(np=1-P)を計算することによって求まる