INFORMATION TECHNOLOGY CENTER
THE UNIVERSITY OF TOKYO

次世代スパコンに向けた演算加速・利用者支援

塙 敏博 （東京大学情報基盤センター）

THE UNIVERSITY OF TOKYO

- 次世代スパコンに向けた演算加速
- 次世代スパコン活用のための利用者支援

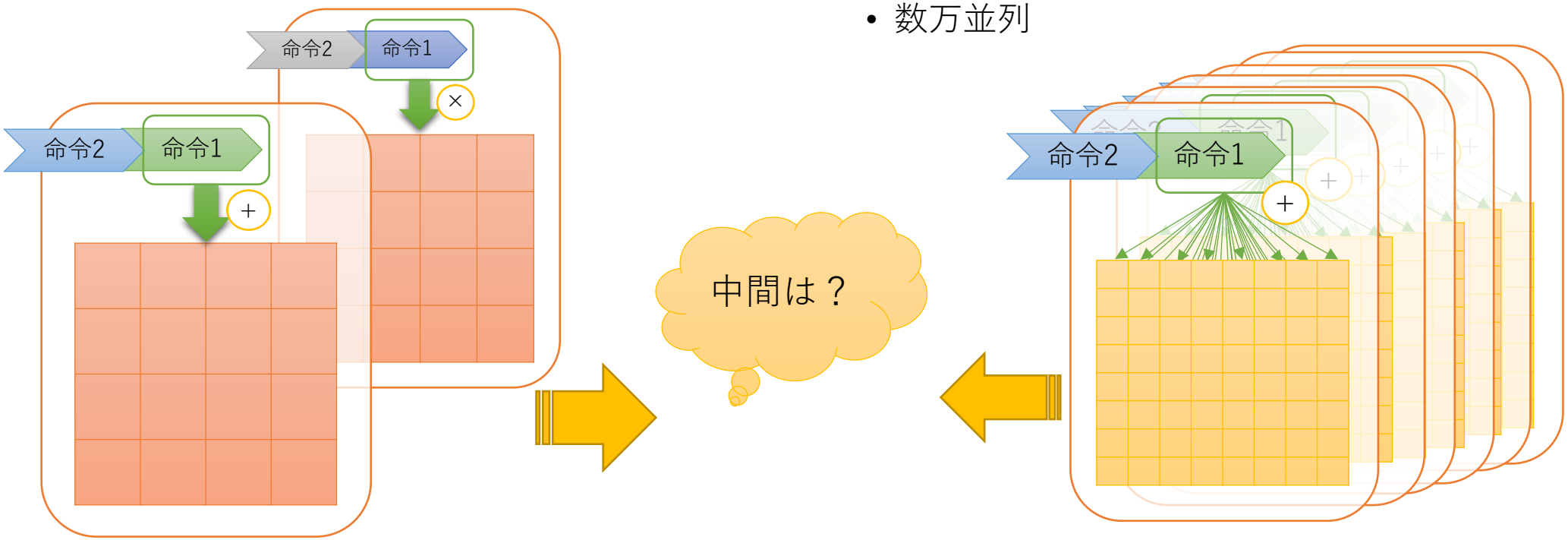
GPUによる演算加速

CPU（マルチコア/メニーコア）

- 複雑な命令を持つ、高速な数十個程度のコア
- 少数のベクトル要素を同時に演算可能
  - Intel Xeonの例：AVX512 … 倍精度浮動小数点 8 要素同時演算可能 x 2個コアに内蔵
- 独立して動作が可能
- 劇的に性能が上げられない：後半を参照

GPU

- 単純な命令だけを持つ、比較的遅い数千個規模のコア
- 大きな単位(32など)の単位で同じ命令を処理
  - NVIDIA GPUの例：ワープ…命令を処理する単位=32
- 自立して動作できない、CPUは必ず必要
  - アクセラレータ、オフローディング処理
- 非常に多くの並列性がないと性能を出し切れない
  - 数万並列



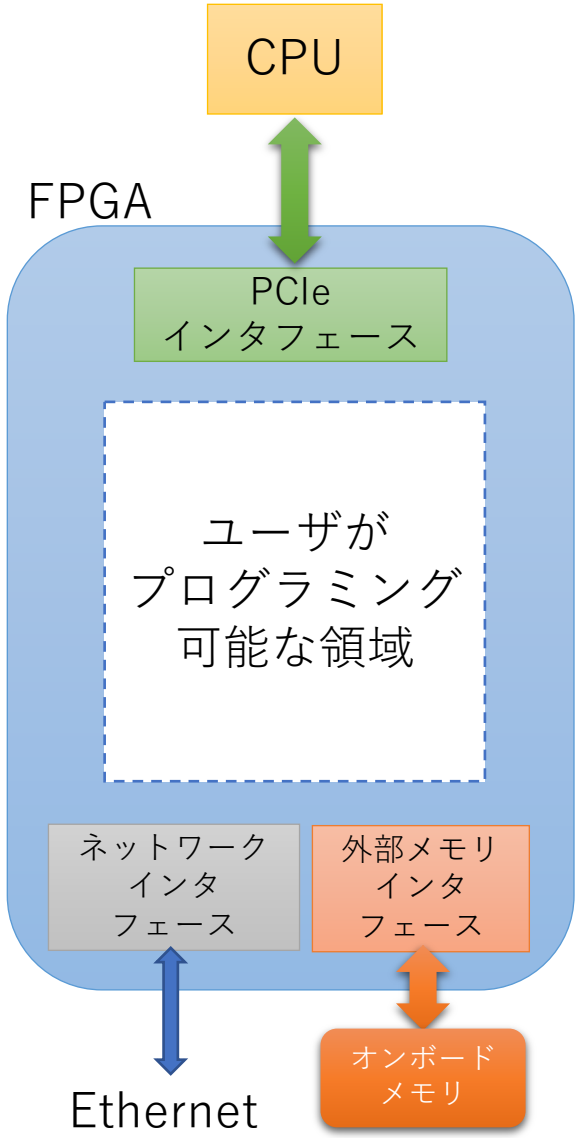
もっと性能を上げたい？

- それなら専用回路を持ったチップを作るしかないでしょう！
  - 深層学習用：Google TPU, プリファードネットワークス MN-Core, …
- でもスパコンは専用ではないし…
  - 様々なアプリケーションが動作する汎用システム

FPGA (Field Programmable Gate Array)

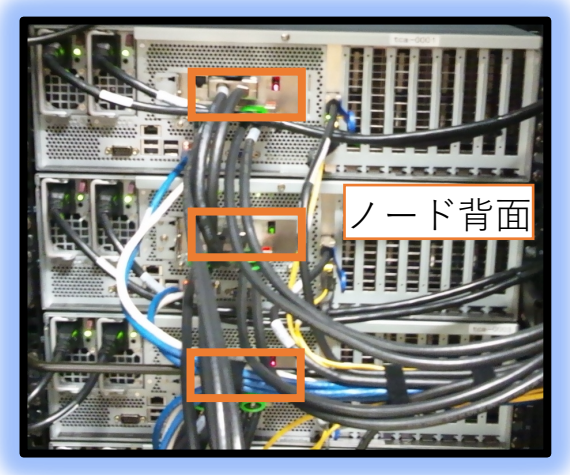
- GPUのための通信機構を実現 (PEACH2/PEACH3, HA-PACS/TCA)
- 実行ごとに回路を入れ替え可能
- GPUと同様アクセラレータとして使用可能
- GPUと違い、複雑な処理を連続して行うことに向く

	性能の目安 (単精度: FP32)
CPU	~5 TFLOPS
GPU	~20 TFLOPS
FPGA	~10 TFLOPS (実際は1桁以下)

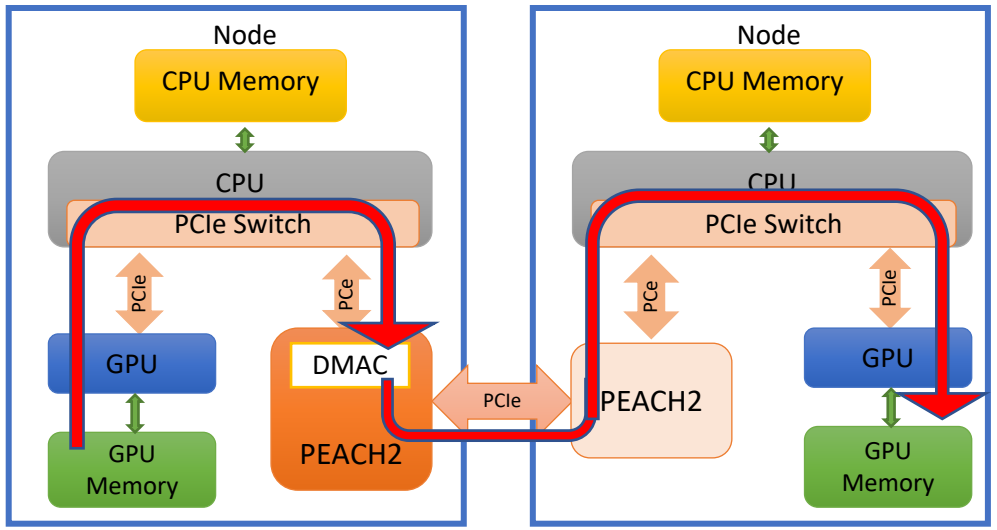


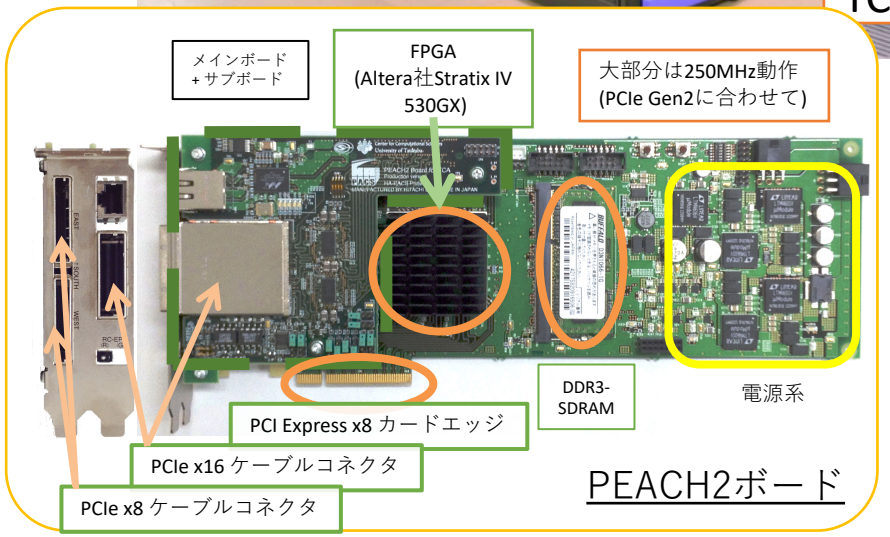
HA-PACS/TCA (2013年11月～2018年10月)@筑波大

FPGAによるノード間GPU直接通信を実現



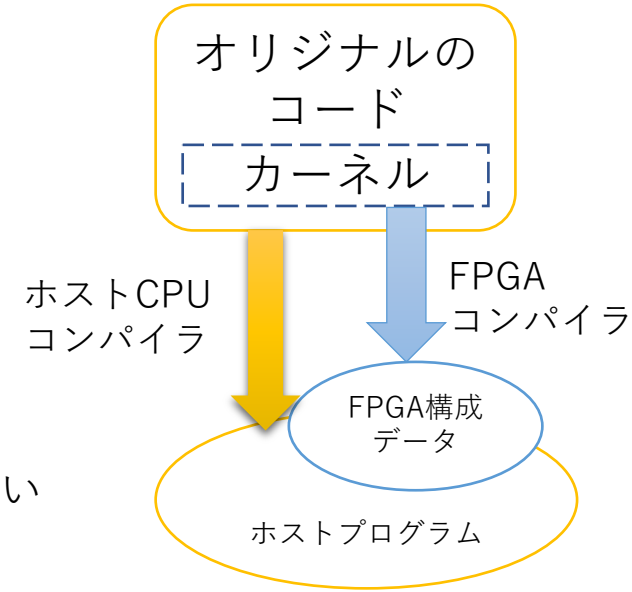
16ノード x 4 グループ，合計10ラック



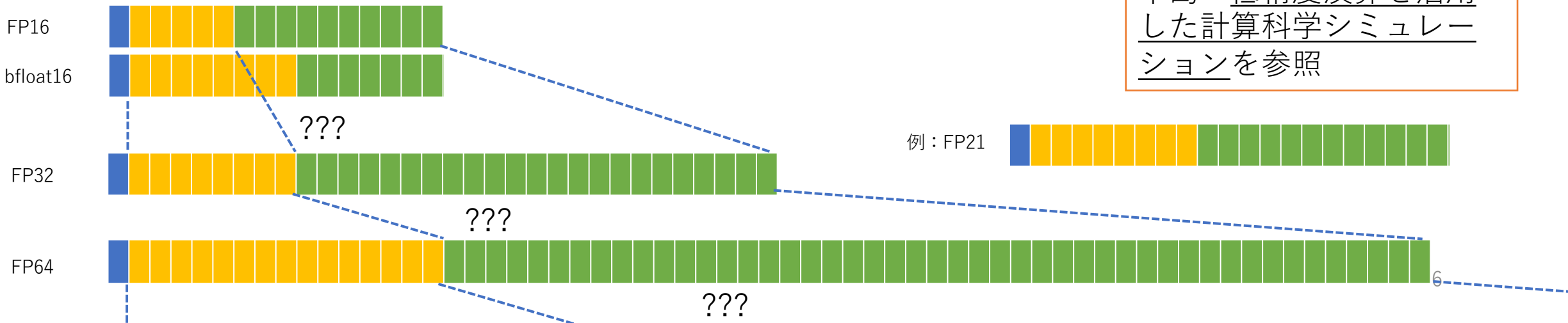


FPGA (Field Programmable Gate Array)

- OpenCLやC++言語(HLS)で設計できる
  - 以前は、ハードウェアを直接記述する必要があった
    - ハードウェア記述言語: Verilog HDL, VHDL, SystemVerilog, …
  - 主要部分（カーネル）をFPGAにオフロード
- 進行中：任意精度演算器を組み合わせた数値計算
  - 現在のCPU, GPUには, FP16/32/64しか演算器がない
    - それ以外のビット数のものは、近い精度の演算器を使って計算して丸めるしかない
  - FPGAなら、直接、任意精度の演算器が作成できる!!
    - 有効性の比較：性能、電力
    - 現実的な時間で実アプリケーションの精度検証が可能



中島：低精度演算を活用した計算科学シミュレーションを参照



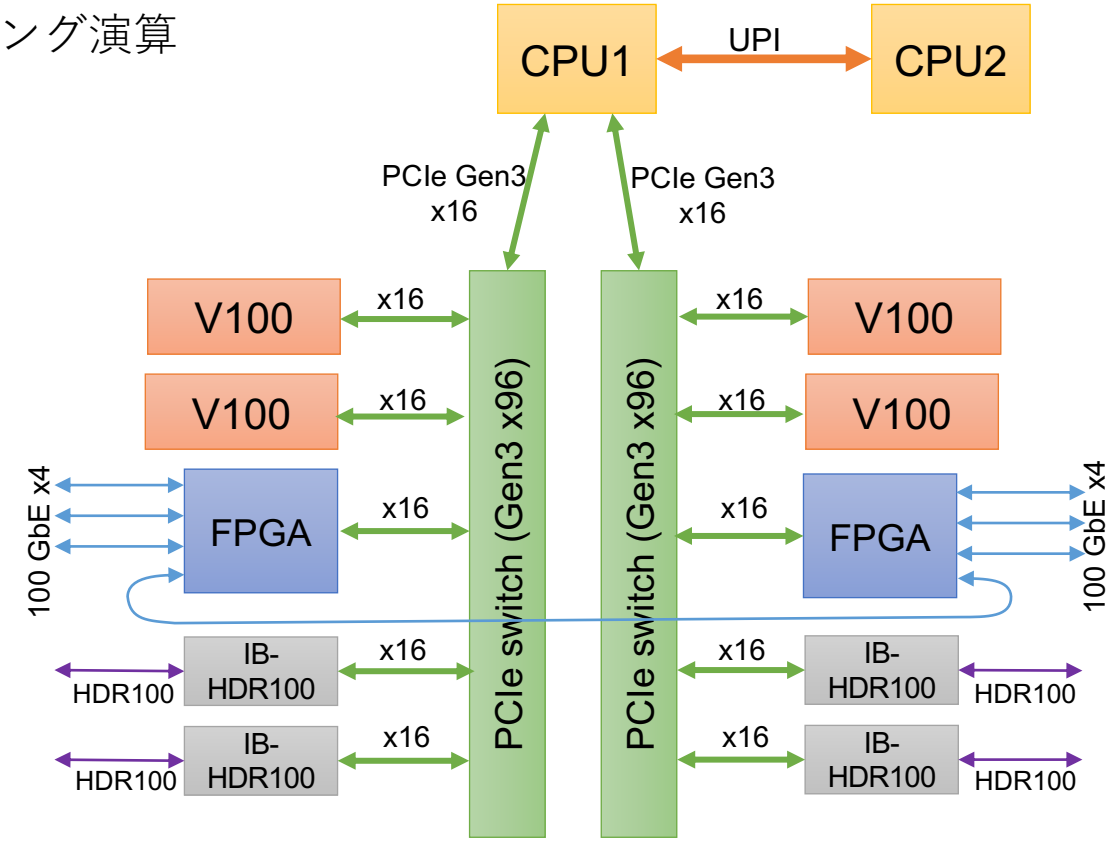
GPU, FPGAマルチハイブリッド

筑波大学計算科学研究センター Cygnus

- CPU, GPU, InfiniBand, FPGA+100 Gbit Ethernet複合システム: AiS (Accelerators in Switch)
- FPGA制御、FPGA-GPU間の連携等にも貢献
  - GPUの得意な部分、FPGAの得意な部分を細かく連携
  - FPGA間の独自ネットワークも利用可能、ストリーミング演算

「再構成可能システムとGPUによる複合型高性能計算プラットフォーム」, 科研費基盤B, 2018–20年度, 代表：朴 泰祐(筑波大)

出典：筑波大学計算科学研究センター



Cygnus Albireoノードの構成

まとめ（次世代スパコンに向けた演算加速）

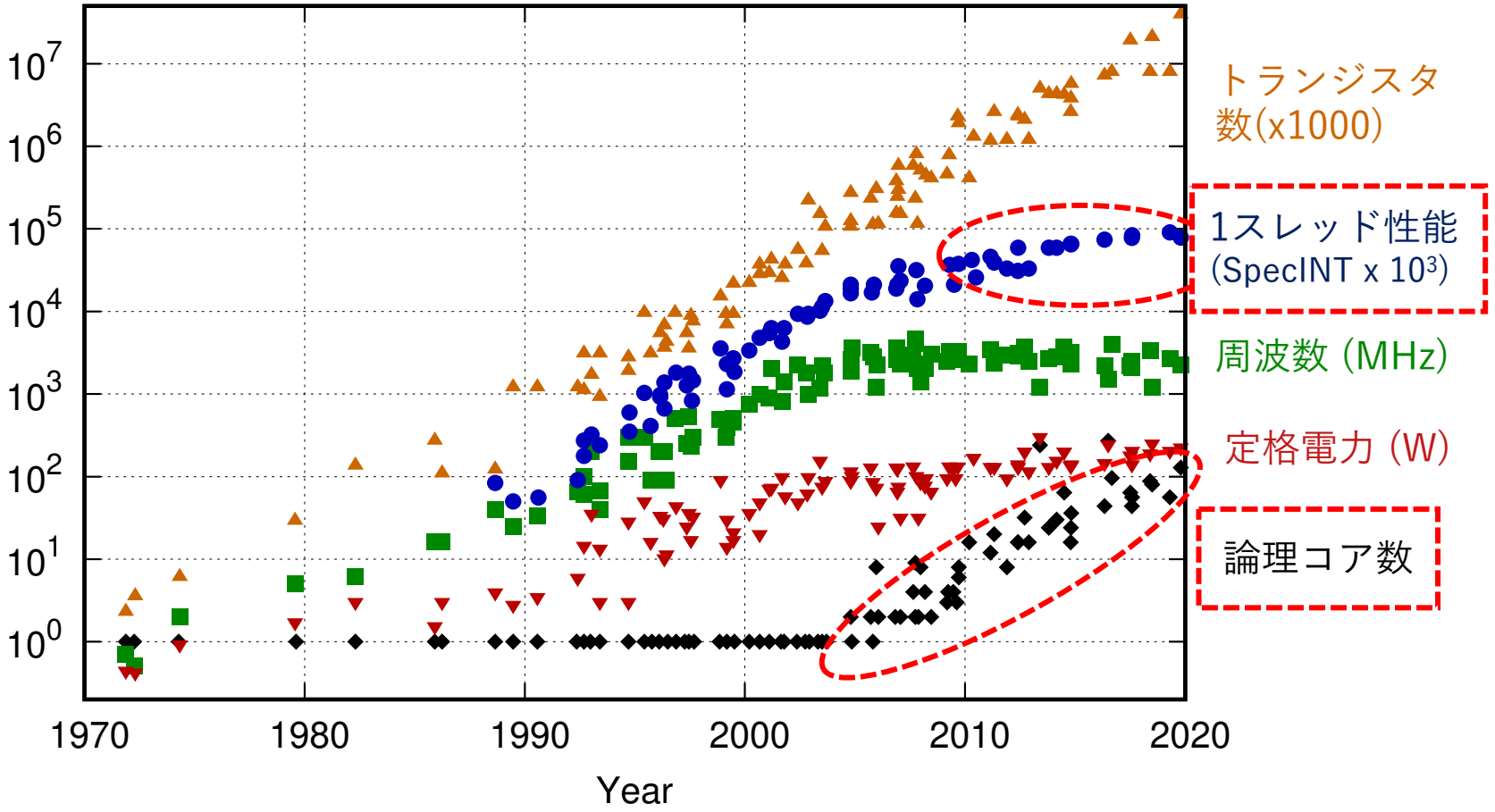
- CPU, GPUに、FPGAも加えた、マルチヘテロアクセラレータ
  - GPUとCPUの間のギャップを補完する
- FPGAは次世代ノード間通信のテスト環境としても有用
  - 通信と演算の融合

- 次世代スパコン性能向上の鍵になる（かも）
  - 実証実験環境としても有望

- 次世代スパコンに向けた演算加速
- 次世代スパコン活用のための利用者支援

将来のスパコンに向けた課題 (1/2)

マイクロプロセッサの性能トレンド



Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten
New plot and data collected for 2010-2019 by K. Rupp

- 近年、1コアあたりの性能は、ほぼ変わらないか、むしろ低下
- コア数の増加だけで性能向上を支える
  - Intel Xeon Phi 7250 (KNL, OFP):
    68コア, 272スレッド
  - Intel Xeon Platinum 8280 (CascadeLake, OBCX):
    28コア/ソケット
    56コア/ノード
  - AMD EPYC Rome:
    64コア, 128スレッド/ソケット
    128コア, 256スレッド/ノード
  - A64FX (富岳):
    48コア+4アシスタントコア

将来のスパコンに向けた課題 (2/2)

電力、熱の制約

- 現状ですでにCPU中全てのコアを全力で動かすことはできない
  - そもそも電力が足りない
  - コアの動作は全て熱になるため、放熱が間に合わない
- 全コアを使って動かそうとすると、全体が遅くなる
  - IntelのAVX512

ダークシリコン問題

メモリバンド幅性能の制約

- 全コアで共有、コア数が少ない方が性能が良くなる場合も
  - Streamベンチ: Xeon Platinum 8280 (CascadeLake)では
    56コア中32コアで最高性能

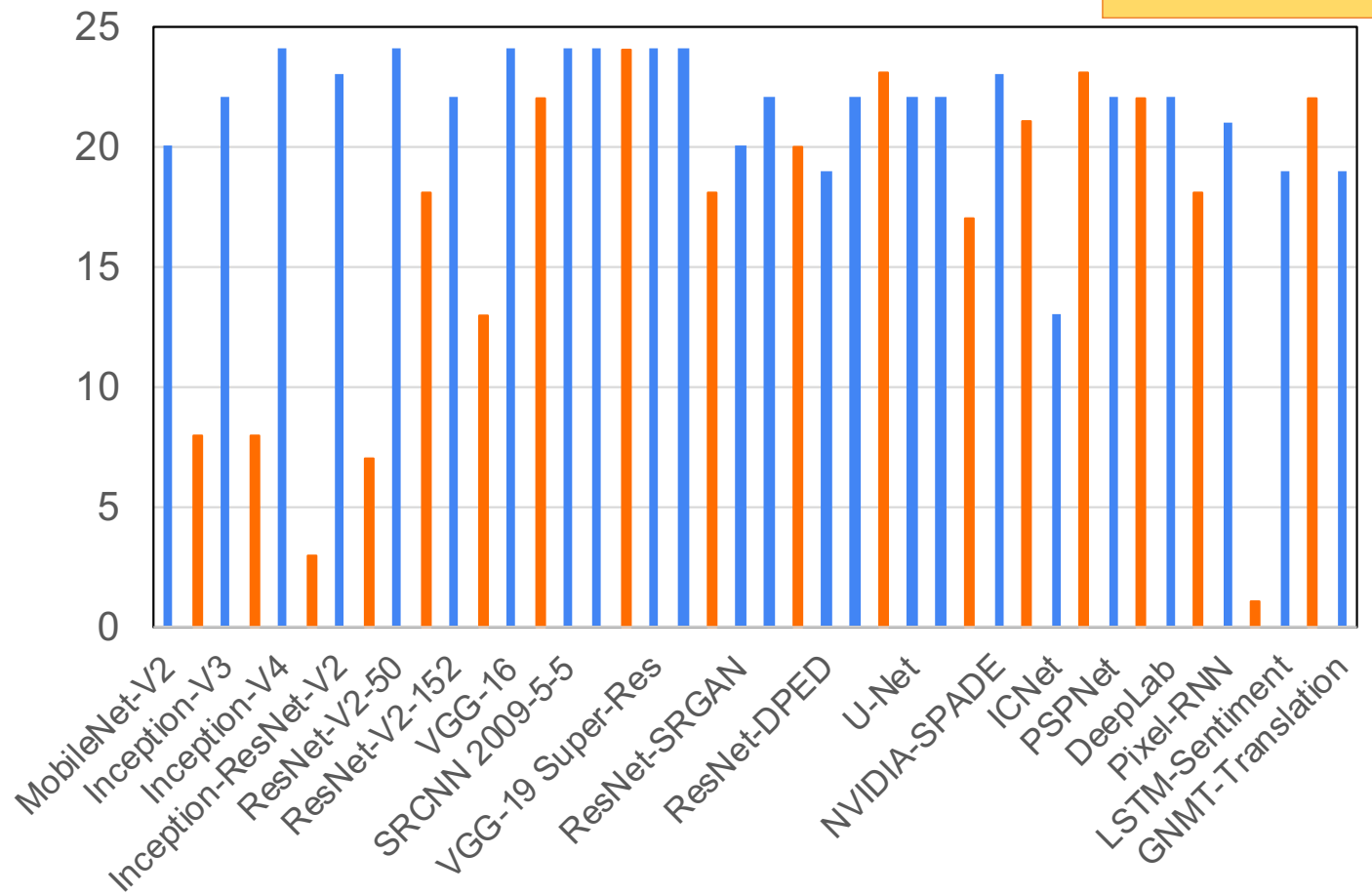
- 現在は、プログラムを使う人が何並列で動かすかを決めている
- しかし、結構難しい場合もある

実測：最適コア数の違い (AI Benchmark)

- コア数を変えて性能を測定
  - 実際には1回のベンチマークで連続して走る
- 24コア(全部)~1コアまで、様々な特性

http://ai-benchmark.com/alpha.html

途中で条件を変えられるようにする必要、
最適な実行条件を決めるのが大変



推論 (inference)
学習 (training)

測定環境：
Intel Xeon Platinum 8260L (CascadeLake) 1ソケット
24コア, 2.4 GHz

余剰コアを活用する高性能計算・データ解析支援

科研費基盤A, 2020–22, 代表

- プログラムに対して、各スレッドのコアに対する最適な割り当て
  - コアの割り当て自動化
  - ヒントの提示

使いきれず余るコア
＝余剰コア

余剰コアを使って何ができるか

- 制約下で如何に有効活用するか
  - 但し主たる計算には（あまり）影響を与えない範囲で
- ユーザーランドで実現する
  - 特別な権限無く使える

- 実行中性能プロファイリング
  - ハードウェア性能カウンタ
  - 追加コード無し
- キャッシュプリフェッチ
- 電力バジェットの自動調整
- 並列数の自動調整
- 通信・ファイルIO隠蔽
- In-Situ 可視化

- ライブラリとツール群などからなるフレームワーク **UTHelper**の開発

まとめ（次世代スパコン活用のための利用者支援）

- 性能を上げるにはコアを増やすしかない
- しかし大量のコアが使い切れず余ってしまうケースも

- 余剰コアの有効利用
  - ものすごく頑張って作り込みをしなくても、
    ほどほどに良い性能が得られるような環境の提供
  - ツール、ライブラリ等の開発

Thank you for watching