次世代メニーコア型スパコンのためのシステムソフトウェア

堀　敦史
システムソフトウェア技術部会 部会長（理研 AICS）
清水　正明
システムソフトウェア技術部会 副部会長（日立）

2017/12/15

- IHK/McKernel
  - 最新情報
- Process in Process (PiP)
  - 新しい並列実行環境の紹介


McKernel

背景

- HPCシステムの複雑化への対応
  - 並列性の増大
    - コア数とノード数
  - メモリ階層数の増大
  - 電力制約

少数のコアをOS専用としても，そのオーバヘッドは無視できる（1/32コアで3パーセント）

スケーラブルで安定した性能の提供と新ハードウェアへの迅速な対応がカギ

- アプリケーションの複雑化
  - 新しいプログラミングモデルの導入
    - 例：In-situデータアナリティクス、ワークフロー
  - 周辺ソフトの複雑化

Linux API で開発されることが多いため、Linux との ABI 互換性が重要

既存の膨大なツールチェインがそのまま使えることが重要

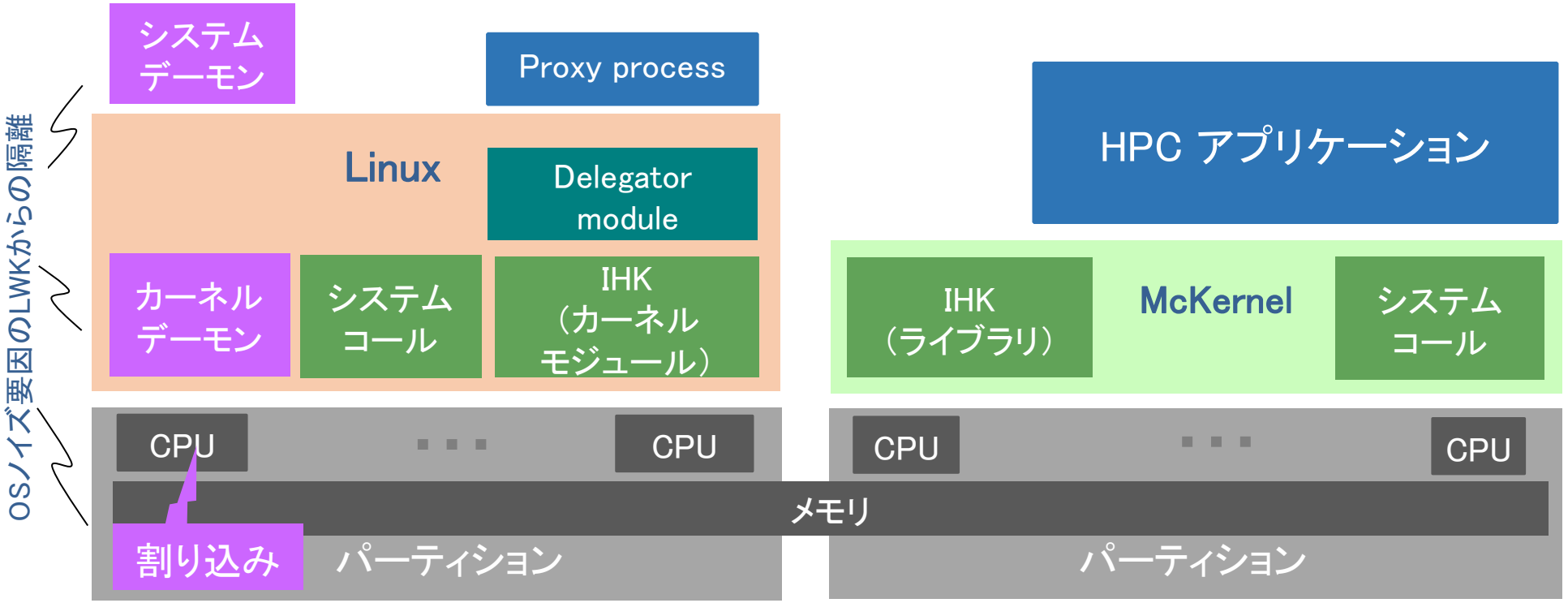
これらの要件を同時に満たせるか？

Linuxカーネルの問題点

- **Linux はころころ変わる**
  - 頻繁かつ大胆に変わる
    - Linux カーネルを改変し，かつ，catch-up するのは非常に大変
  - 多様なハードウェア
    - 次々に現れるデバイスハードウェア
    - デバイスドライバの作成にはH/Wの詳細な情報が不可欠
- **Linux カーネルを改造するのは非常に大変**
  - プロダクトとしては不可能に近い
- **しかしながら，言語環境，実行環境は Linux を継承したい**
  - 環境も作るのは大変
  - Linux API/ABI との互換性をどのように実現するか？

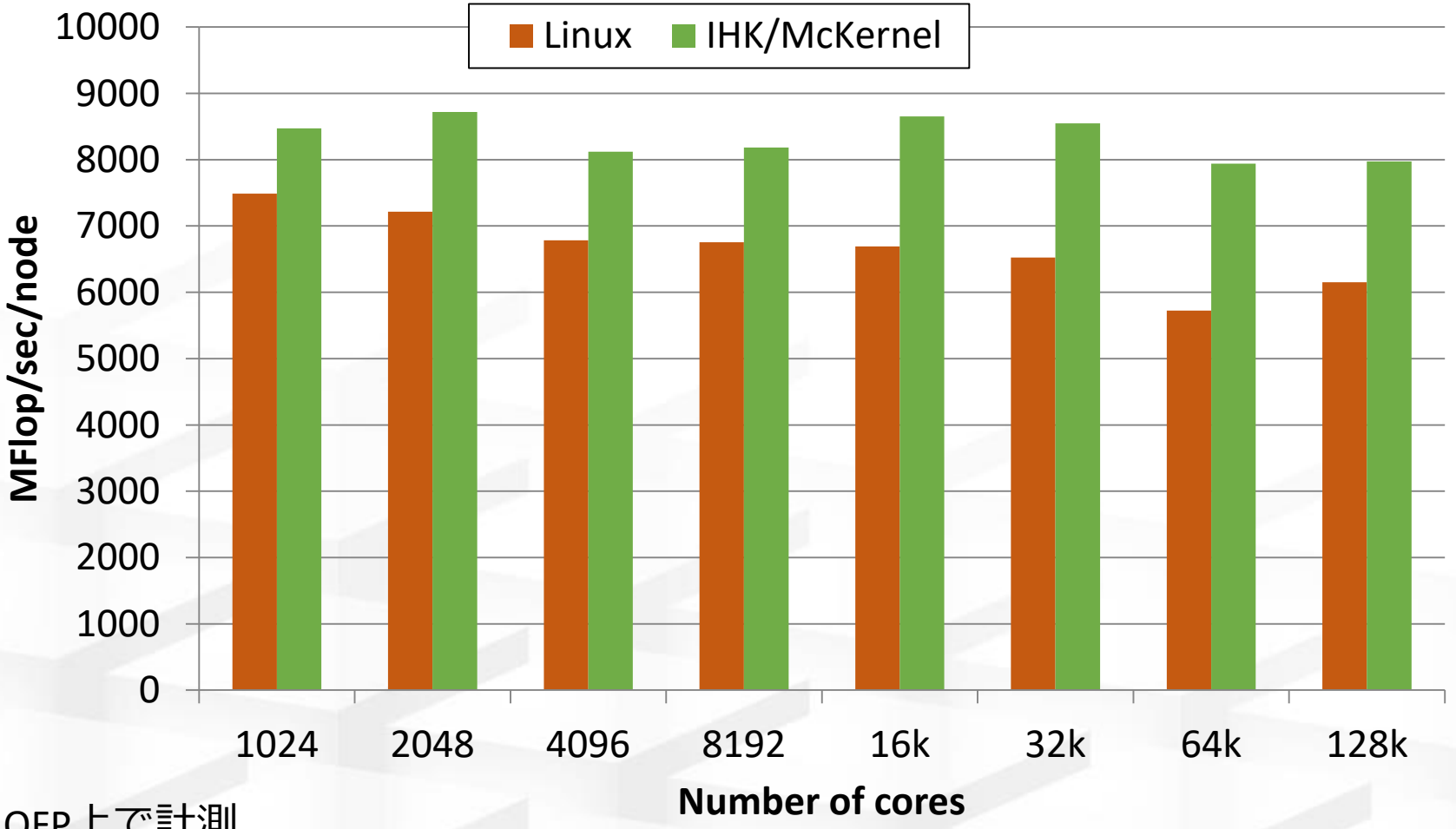
McKernel のアーキテクチャ

1. Interface for Heterogeneous Kernels (IHK): 複数の異種OSを持つためのフレームワーク
   - ノード資源のパーティショニング
   - LWKの管理(例:カーネルの起動、停止)
   - LWKとLinuxの間の通信機構(Inter-kernel communication, IKC)の提供
2. McKernel: スクラッチから開発されたHPC向けLWK
   - ノイズレス
   - 性能クリティカルなシステムコールのみMcKernelで動作、他はLinuxにオフロード



CCS-QCD (Hiroshima University)

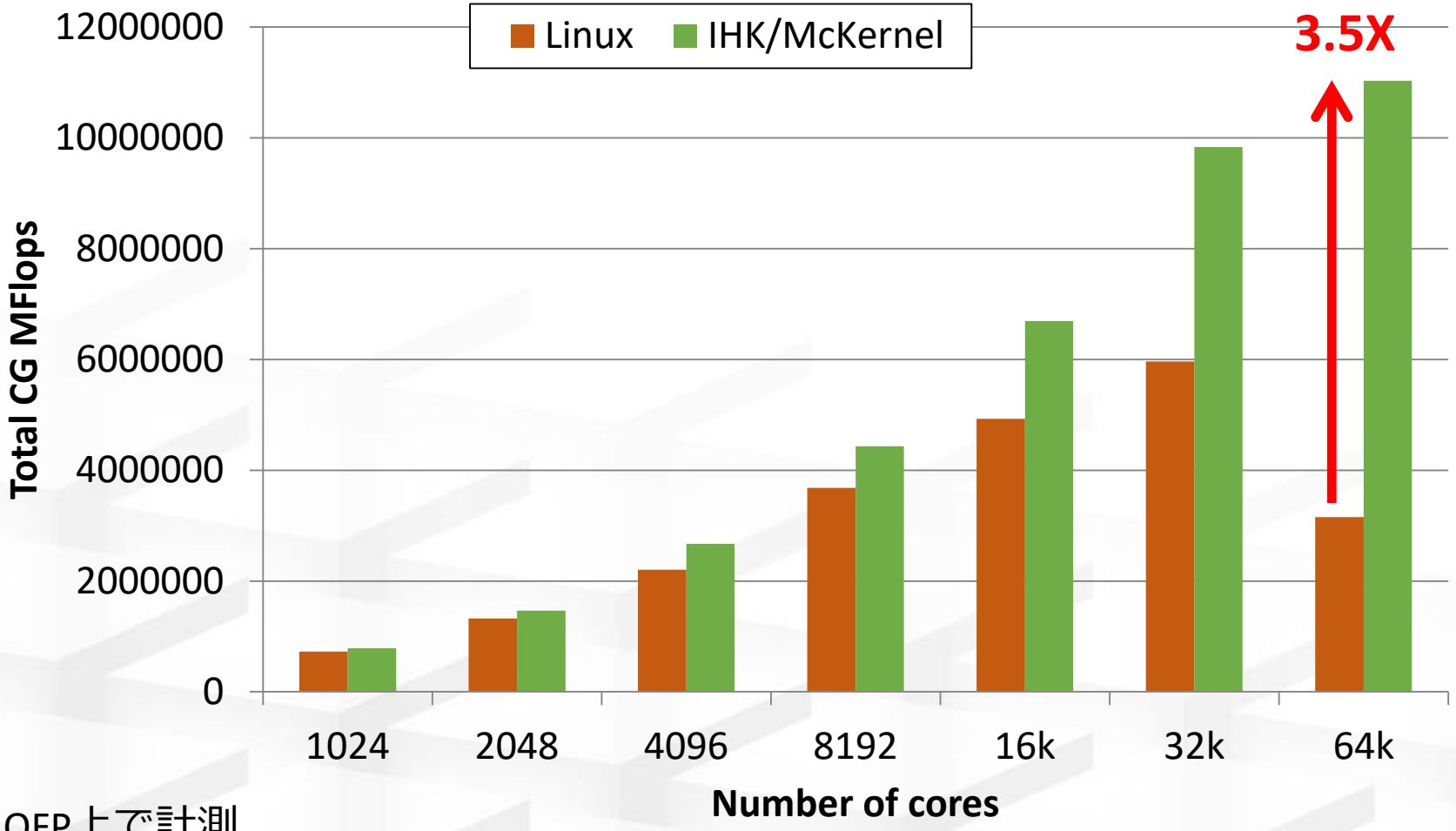
- Lattice quantum chromodynamics code – weak scaling
- Up to 38% improvement



OFP上で計測

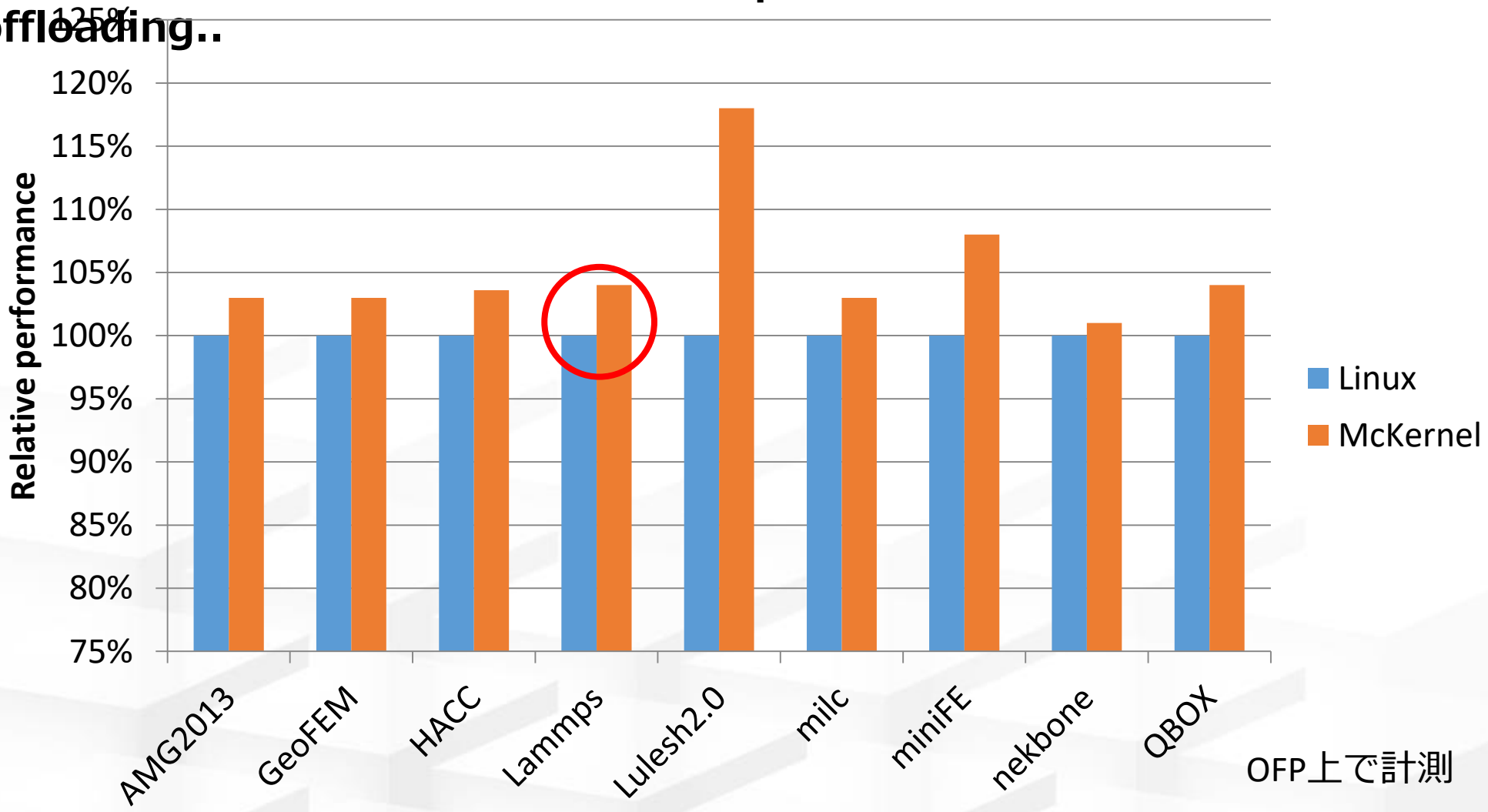
miniFE (CORAL benchmark suite)

- Conjugate gradient - strong scaling
- Up to 3.5X improvement (Linux falls over..)



OFP上で計測

Single node: McKernel outperforms Linux across the board → multi-node Lammps suffers from network offloading..



OFP上で計測

- lammps, HACC, QBOX ~4% better, as opposed to being slower than Linux on 8 nodes
- OmniPath offload overhead??

IHK/McKernel - まとめ

- **Xeon Phi 上でも動作**
  - Knights Landing (KNL)　　最新メニーコアCPU
- **Linux+IHK/McKernel**
  - LinuxコアでMapReduce，IHK/McKernelコアで並列アプリという動作でも，Linux だけよりも性能独立性が高い
  - Linuxよりはるかに単純なので，変更，機能追加が容易
    - IHK/McKernel：約 8 万行
    - Linux：2100万行！！
      （http://news.mynavi.jp/news/2016/04/11/130/）
- **より高いLinuxコンパチビリティ**
  - Procfs および numa 機能の実装など
- **実用性**
  - Linux を reboot せずに IHK/McKernel を boot できる
- **実戦配備**
  - Oakforest-PACS（東大・筑波大）で採用
  - ポスト京コンピュータでも採用される予定

Process in Process (PIP)

背景

- **メニーコアの台頭**
  - これまでのノード内並列モデルのままで良いのか？
  - マルチプロセス（MPI）、マルチスレッド（OpenMP）

		Address Space	
		Isolated	Shared
Variables	Privatized	Multi-Process (MPI)	*3rd Exec. Model*
	Shared	??	Multi-Thread (OpenMP)

アイデア

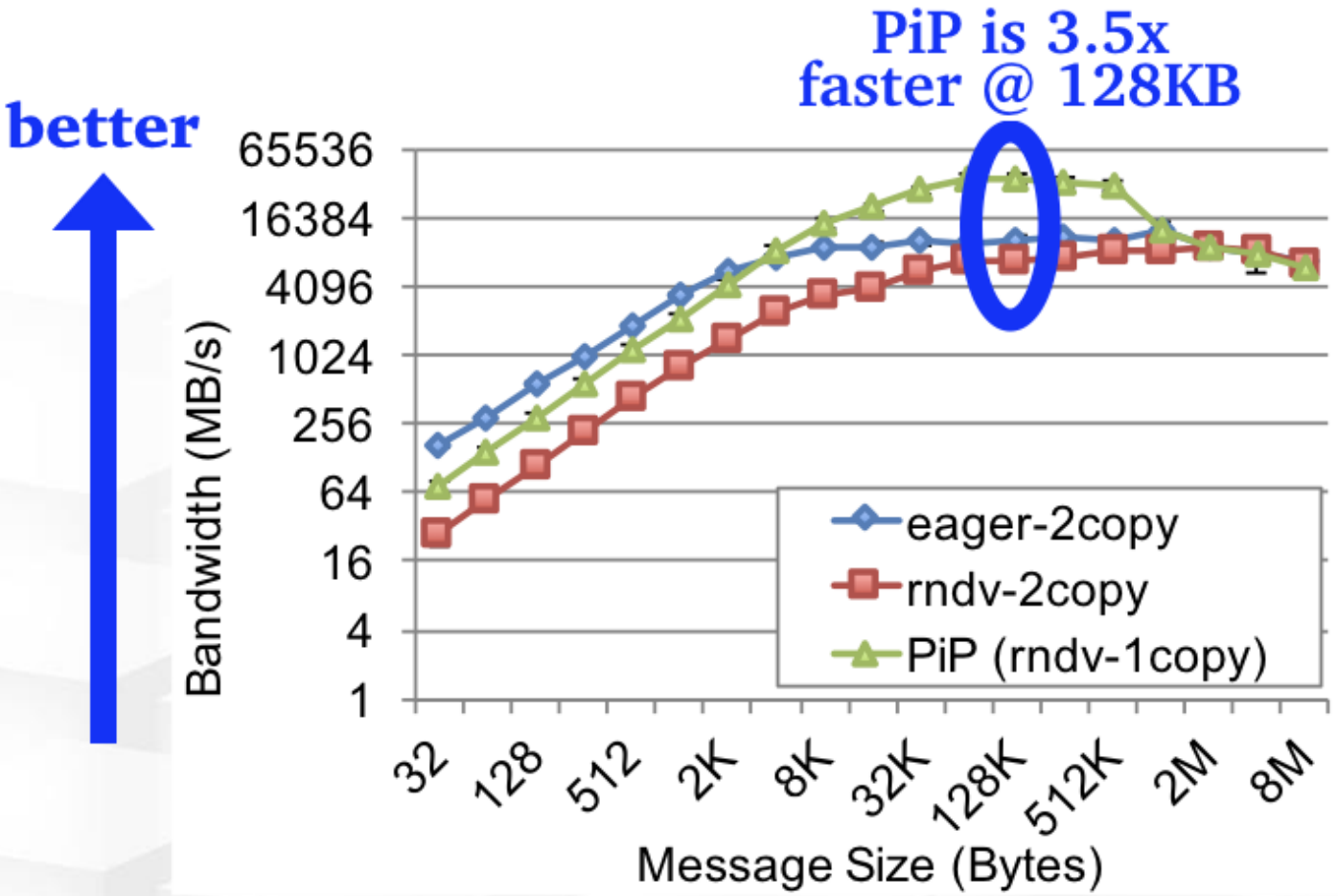
- **プロセス間でデータを互いに直接アクセスできるようにしよう！**
- **既存研究**
  - Smartmap (米SNL)
    - **Kitten OS**
    - X86 アーキテクチャに依存
  - Xpmem (SGI ?)
    - **Linux カーネルモジュール**
    - 直接アクセスするためにシステムコールが必要
  - PVAS (理研)
    - **Linux カーネルに約 5,000 行のパッチ**
  - MPC (CEA@仏)
    - **独自のコンパイラが必要**
- **PiP はユーザレベルの普通のライブラリ**

HPDC'18 で採択


The 27th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC)
Tempe, Arizona, June 11 - June 15, 2018
KARSTEN SCHWAN
BEST PAPER AWARD
presented to
Atsushi Hori (RIKEN); Min Si (ANL); Balazs Gerofi,
Masamichi Takagi (RIKEN); Jai Dayal (Intel);
Pavan Balaji (ANL); Yutaka Ishikawa (RIKEN)
For their paper: Process-in-Process: Techniques for Practical Address-Space Sharing

MPICHでの評価

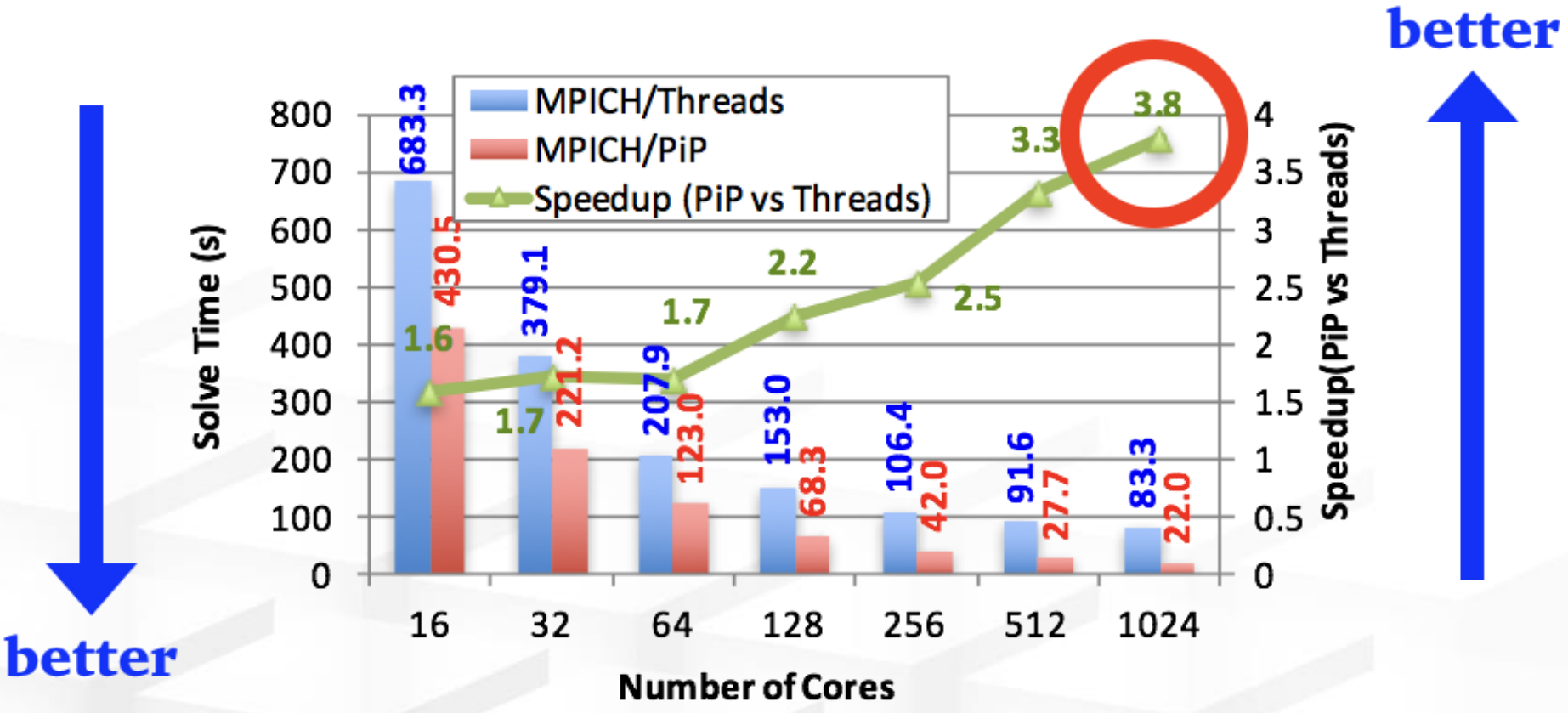
- Current　Eager/Rndv. 2 Copies
- PiP　Rndv. 1 Copy



(Xeon/Linux)

SNAPでの評価

- (MPI + OpenMP) ⇨ (MPI + PiP)



PiP V.S. threads in hybrid MPI+X SNAP strong scaling on OFP (1-16 nodes, flat mode).

PiP – まとめ

- **従来の２つのノード内並列実行方式、プロセス並列とスレッド並列のいいとこ取り**
- **これまでに提案された方式と異なり、ユーザレベルで実装される**
- **PiP により、以下の利点が期待される**
  - より高速なノード内通信
  - より低メモリ消費な並列実行環境
- **オープンソース**
  - MPICH、Open MPI の PiP による実装
- **共同研究**
  - アルゴンヌ研究所、テネシー大学、イリノイ大学、仏CEA など