2019年2月22日

PCクラスタワークショップin柏2019

PCクラスタコンソーシアム実用アプリケーション部会報告

OpenFOAMによる流体解析ベンチマークテスト
スパコン・クラウド・FOCUSでのチャネル流れ解析および
Oakforest-PACSにおけるOptunaを用いたHBM設定のベイズ最適化

今野　雅

PCクラスタコンソーシアム実用アプリケーション部会 委員

オープンCAE学会V&V小委員会 委員長

東京大学情報基盤センター 客員研究員

株式会社OCAEL

オープンCAE学会共通OpenFOAMベンチマーク

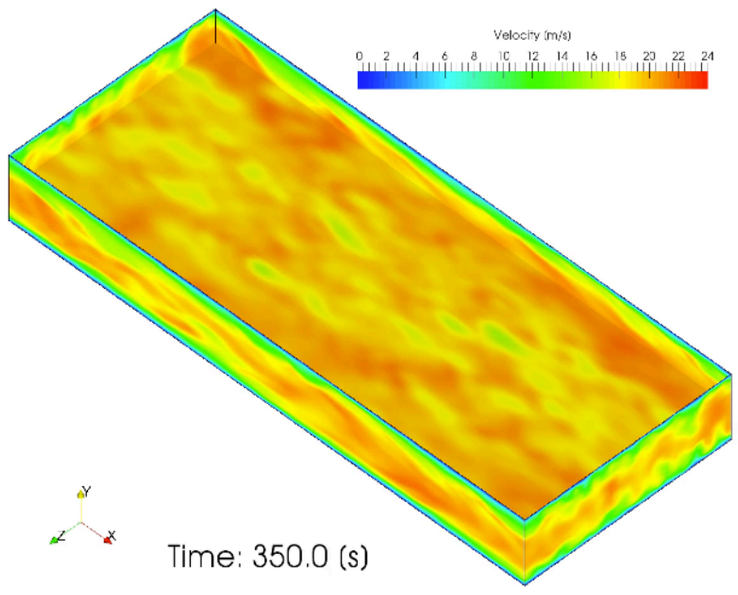
- **大学のスーパーコンピュータ**
  - ✓ 近年は産業利用など教育・公共機関以外でも利用可能なシステム有り
  - ✓ 通常，課題審査が必要．通常1ヶ月～1年単位での課金
- **産業界専用のスーパーコンピュータFOCUS**
  - ✓ 法人の場合，課題の審査無く利用可能
  - ✓ 1円単位で使った分だけ課金．ただし1年毎にアカウント発行料も必要
- **クラウドサービス**
  - ✓ 通常，誰でも課題の審査無く利用可能で，使った分だけ課金(分または時間単位)
- **CPU，メモリ，インターコネクトに違いがあり，性能比較が難しい．**

▼

- **オープンCAE学会V&V小委員会で，チャネル流れによる共通OpenFOAMベンチマークを作成し，解析速度と並列化効率を比較した．**
- **測定結果は学会のリポジトリで公開している．**
- **なお，昨年までは費用比較も行なっていたが，価格体系が複雑であったり，定額制であるシステムも含まれる事から，今回は費用比較を行わない．**

オープンCAE学会チャネル流ベンチマーク

チャネル流れ ($Re_\tau = 110$)



格子数約3M (0.37, 24M 等他の格子数の結果はレポジトリ参照)

- ✓ メッシュ生成に時間を要しない
- ✓ 構造格子のため，格子数変更が容易
- ✓ 圧力と速度のみ解くので，「圧力線形ソルバの解析時間が支配的」という非圧縮性流体解析の特性を素直に示す

解析条件

$L_x \times L_y \times L_z = 5\pi \times 2 \times 2\pi$

$Re_\tau = u_\tau \, \delta / \mu = 110$ [-]

ここで

L_x, L_y, L_z：各方向のチャネル幅 [m]

u_τ：壁面摩擦速度 [m/s]

δ：チャネル半幅 [m] ($=L_y / 2$)

μ：動粘性係数 [m^2/s^2]

主流方向(x): 一定の圧力勾配

主流方向(x), スパン方向(z): 周期境界

ソルバ: OpenFOAM pimpleFoam

乱流モデル: 無し(laminar)

速度線型ソルバ: BiCG (前処理DILU)

圧力線型ソルバ: PCG (前処理DIC)

領域分割手法：scotch(周期境界面は同領域)

- **2~51ステップの平均CPU時間(Execution time)から1時間あたりのステップ数を算出**

計測システム(黄色：追加，オレンジ：再測定)

機関	システム (略称)	CPU [Intel Xeon] (周波数[GHz])	CPU (コア)	倍精度性能 [GFlops]	メモリ[GiB] (帯域幅[GB/s])	インターコネクト (帯域幅[Gbps])
JCAHPC	Oakforest-PACS (OFP)	Phi 7250 (1.4)	1(68)	3046	96(115.2), MCDRAM 16(490)	Intel Omni-Path (100)
東京大学	Reedbush-U (RBU)	E5-2695 v4(2.1-3.3)	2(36)	1210	256(76.8×2)	Infiniband EDR(100)
FOCUS	A(FA)	L5640(2.26)	2(12)	108	48(25.6×2)	Infiniband QDR(40)
	D(FD)	E5-2670 v2(2.5)	2(20)	400	64(51.2×2)	Infiniband FDR(56)
	F(FF)	E5-2698 v4(2.2)	2(40)	1152	128(76.8×2)	
	H(FH)	D-154(2.1)	1(8)	205	64 (34.1)	10GbE(10)×2 or 4
Amazon	c4.8xlarge(c48x)	E5-2666 v3(2.9)	2(18)	310	60(不明)	10GbE(10)
Microsoft	Azure A9(A9)	E5-2670(2.6)	2(16)	333	112(不明)	Infiniband QDR(40)
	Azure H16r(H16r)	E5-2667 v3(3.2)	2(16)	691	112(不明)	Infiniband FDR(56)
大阪大学	OCTOPUS(OCT)	6126 (2.6)	2(24)	1997	192(255.9)	Infiniband EDR(100)
九州大学	ITO A(ITO)	6154 (3.0-3.7)	2(36)	3456	192(255.9)	Infiniband EDR(100)
Oracle	BM HPC 2.36(ORA)	6154 (3.0-3.7)	2(36)	3456	384(255.9)	RoCE v2 (100)

注) 倍精度性能は，機関のWEBページに明記された値，または，AVXの動作周波数を考慮して算出したWEBページから引用

使用OpenFOAMバージョン・コンパイラ・MPI

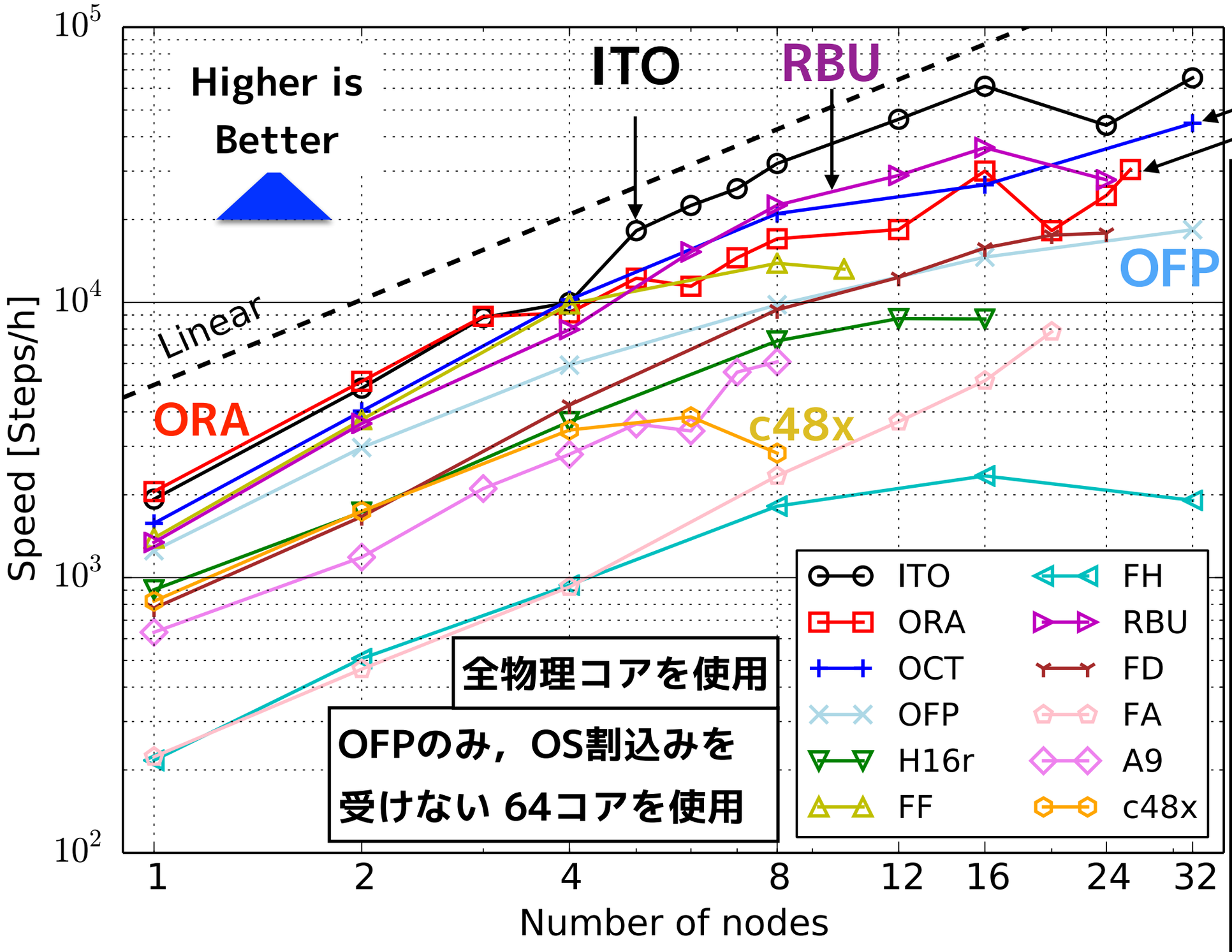
機関	システム	バージョン	コンパイラ	MPI
JCAHPC	OFP(Flatモード)	OpenFOAMv1612+	Icc 2019.1	Intel MPI 2019.1
大阪大学	OCT		Icc 2018.2	Intel MPI 2018.2
東京大学	RBU	OpenFOAM 2.3.0	Gcc-4.8.5	SGI MPI 2.14
九州大学	ITO			Intel MPI 2018.1
Oracle	ORA			Intel MPI 2018.1
Microsoft	H16r			Intel MPI 5.0.3
	A9		Gcc 4.8.3	Intel MPI 5.1.1
Amazon	c48x			OpenMPI 1.8.5
FOCUS	FOCUS			OpenMPI 1.6.5

実行時の主な設定

OFP) WM_COMPILER=IccKNLでコンパイル．unset KMP_AFFINIY KMP_HW_SUBSET=1T I_MPI_PIN_PROCESSOR_EXCLUDE_LIST=0,1,68,69,136,137,204,205
I_MPI_PIN_DOMAIN=4 MPI_BUFFER_SIZE=6291456 HBM_THRESHOLD=4 HBM_SIZE=240 LD_PRELOAD=libhbm.so
OCT) I_MPI_DYNAMIC_CONNECTION=0 I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
ITO) I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
ORA) I_MPI_DAPL_TRANSLATION_CACHE=0 I_MPI_DAPL_UD_TRANSLATION_CACHE=0
H16r) I_MPI_FABRICS=shm:dapl I_MPI_DAPL_PROVIDER=ofa-v2-ib0 I_MPI_DYNAMIC_CONNECTION=0 I_MPI_DAPL_DIRECT_COPY_THRESHOLD=655360
A9) I_MPI_FABRICS=shm:dapl I_MPI_DAPL_PROVIDER=ofa-v2-ib0 I_MPI_DYNAMIC_CONNECTION=0
c48x) mpirun -bind-to core FOCUS) mpirun -bind-to-core -mca btl openib,sm,self

OFPではMC-DRAMの設定が必要だが，後述するグリッドサーチによる最適値を用いた．

各システムの解析速度比較



最高速

・1～3ノード：

ORA (Skylake 6154, RoCE v2)

・4～32ノード：

ITO (Skylake 6154, IB EDR)

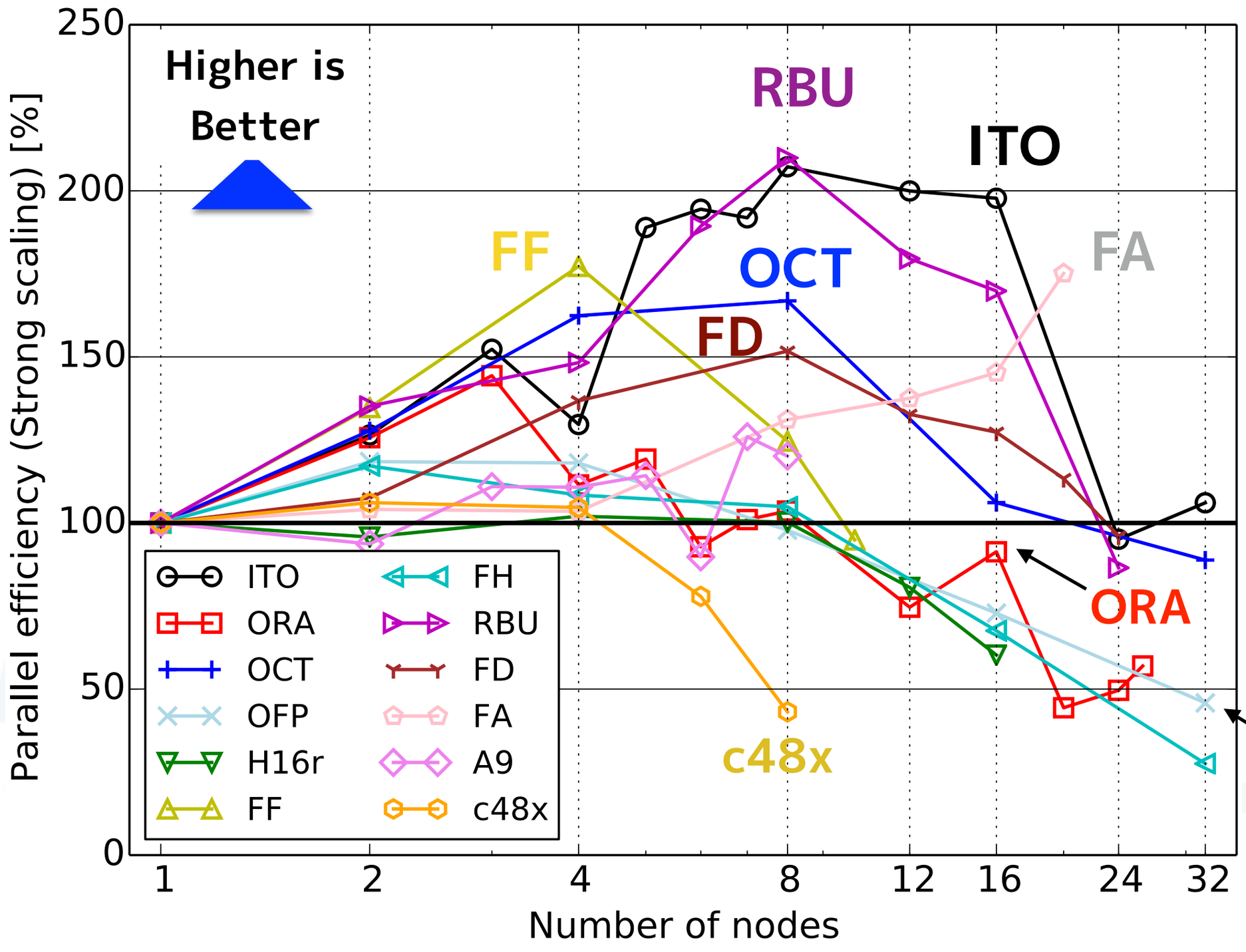
次点

・OCT (Skylake 6126, IB EDR)

・RBU (Broadwell, IB EDR)

10GbEであるc48xは6ノードで飽和

各システムのStrong scaling並列化効率比較



- Infinibandの RBU，ITO，OCT，FD，FAは16ノードまでスーパリニア
- RoCE v2のORAは16ノードまで概ねスケール
- OFPは12ノード以上で悪化
- 10GbEのc48xは6ノードから劣化

OFPフラットモードにおけるlibhbmの設定

- OFPのFlatモードにおけるOpenFOAMの実行には，libhbmを用いている．
- libhbmは，memkindのautohbmと同様にHBM(High-Bandwidth Memory)のメモリ確保をするが，アプリケーション開始時に，プロセス毎に固定サイズのヒープを確保し，アプリケーションが終了するまで開放しない点が異なる．
- libhbmを用いた実行時には，以下の表に示す環境変数を適切に指定する必要があり，libhbmの配布先(レポジトリ)では推奨値が示されている．

環境変数	説明	推奨値	推奨値の根拠
HBM_SIZE	ヒープサイズ	256MB	HBMサイズ(16GB) / プロセス数(今回64)
HBM_THRESHOLD	HBMに割り当てる下限サイズ	16KB	HBMの断片化を避けると共に，2つのメモリのバンド幅を活かす(サイズが小さい場合，レイテンシが小さいDDR4を使う)．
MPI_BUFFER_SIZE	MPIバッファのサイズ	1MB	大抵のソルバでは大きな値を必要としないので，MC-DRAMを浪費しないように1MB程度に小さくする．

Optunaを用いたlibhbm設定のベイズ最適化

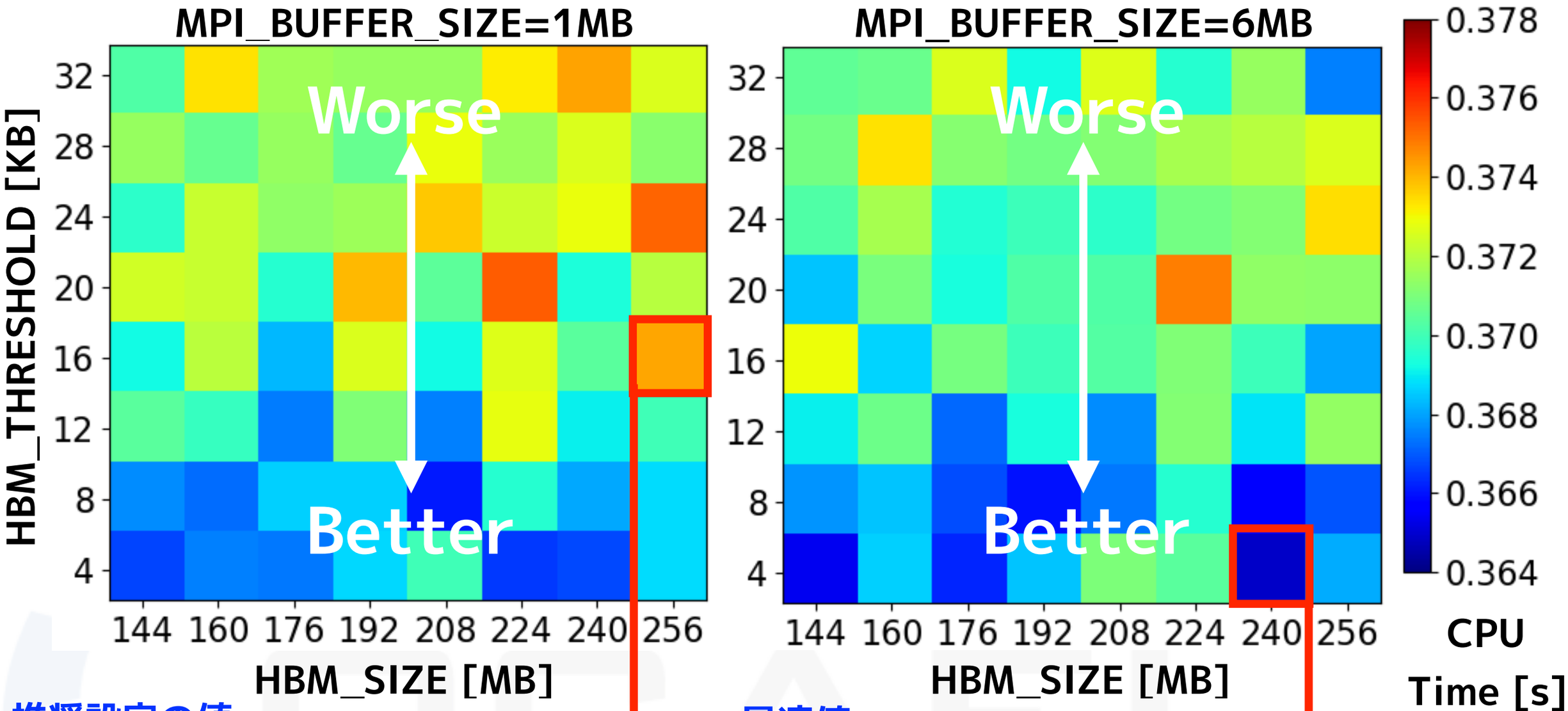
- MPIバッファサイズやlibhbm設定の最適値は問題に依存し，チャネル流での最適値は推奨値と異なる可能性があるので，グリッドサーチの最適化を行った.
- 一方，CFD解析の試行時間は長いので，よりパラメータ数が多く，パラメータ空間が広い場合，グリッドサーチの最適化コストは莫大となる可能性がある.
- そこで，ハイパーパラメータ自動最適化ツールOptunaを用いたベイズ最適化を行い，以下に示すパラメータ空間におけるグリッドサーチとの比較により，その収束性や効率を検討した.

表：グリッドサーチおよびベイズ最適化のパラメータ空間 (総数512)

環境変数	設定値	サンプル数
HBM_SIZE	144MB～256MB，16MB刻み	8
HBM_THRESHOLD	4～32KB，4KB刻み	8
MPI_BUFFER_SIZE	1MB～8MB，1MB刻み	8

- 同パラメータでの解析を5回以上行い，CPU時間の平均値を最適化した.
- なお，並列化効率が1に近い8ノード・512MPI並列計算のみを対象とした.

グリッドサーチによる平均CPU時間ヒートマップ



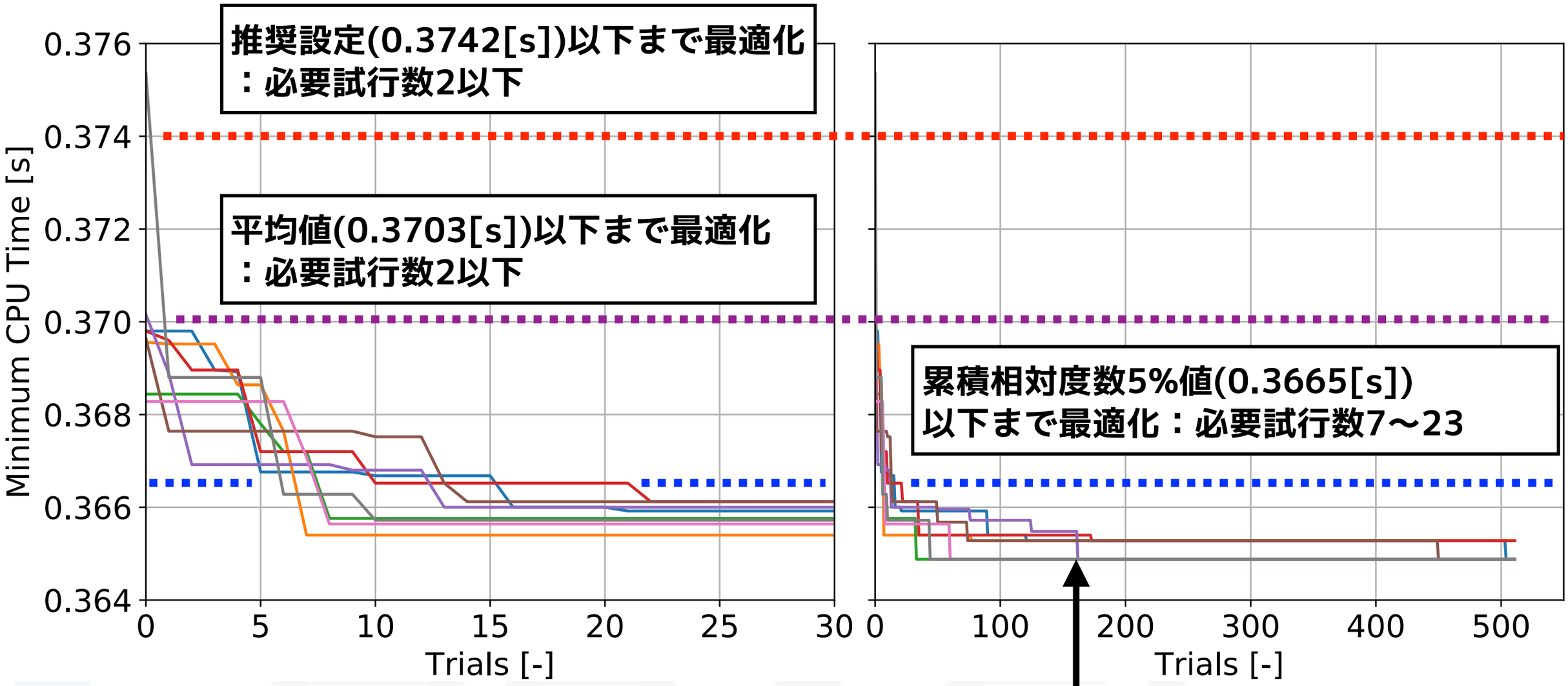
推奨設定の値

CPU Time: 0.3742[s]
累積相対度数=97%(非常に遅い)
MPI_BUFFER_SIZE=1MB
HBM_SIZE=256MB, THRESHOLD=16KB

最適値

CPU Time: 0.3649[s]
推奨設定より約2.5%高速
MPI_BUFFER_SIZE=6MB
HBM_SIZE=240MB, THRESHOLD=4KB

試行数512×8回のベイズ最適化結果



8回中6回は最適値0.3649[s]まで概ね160試行で到達するが，2回は最適値にならない．
ただ，その到達値0.3653[s]は最適値より0.1%大きいだけであり，累積相対度数は0.6%である．
なお，MPI_BUFFER_SIZE=5MB，HBM_SIZE=160MB，HBM_THRESHOLD=4KBであった．

- ベイズ最適化により累積頻度5%の値まで最適化するのに必要な試行数は7～23．
- 必要試行数は全サンプル数512の4.5%以下であり，大変効率的に最適化される．

まとめ

- FOCUSや大学のスパコンおよびクラウドにおいて，オープンCAE学会V&V小委員会作成のチャンネル流れによるOpenFOAMベンチマークテストを実行し，各システムでの解析速度や並列化効率を比較した．
  - ✓ 理論演算性能やメモリバンド幅が高い最新のシステムは高い性能が得られた．
  - ✓ インターコネクトがInfinibandのシステムでは，16ノードまでスーパーリニアとなるが，10GbEのシステムでは6ノード以上でスケールしなかった．一方，RoCE v2のシステムは16ノードまで概ねスケールした．
- KNLのフラットモードでlibhbm等のHBM用ライブラリを用いる場合，新たにHBMの設定やMPIバッファサイズの調整が必要となるが，Optunaを用いたベイズ最適化により，グリッドサーチに比べ大変少ない試行数で最適化できた．
  - ✓ 今回のベンチマークではCG法の圧力線形ソルバの前処理としてICに固定しているが，AMGも加えて最適化すると多くのパラメータが増えるので，OptunaのDefine-by-Runや枝刈り機能を用いて，線形ソルバの最適化まで試したい．

謝辞：電通国際情報サービス(計測当時)の住友様からAmazon EC2，日本Microsoft(計測当時)の佐々木様からMicrosoft Azure A9，日本Microsoftの五十木様からMicrosoft Azure H16rでのベンチマークの結果をご提供頂いた．ここに深く感謝する．

附録：OpenFOAM自動ビルドスクリプト

- 実用アプリケーション部会でOpenFOAM講習会を実施したシステムでは，原則として本OpenFOAMベンチマークを実施し，公開している(例：大阪大学 OCTOPUS).
- OpenFOAMをビルドするには，各種依存ソフトウェアが必要であり，依存ソフトウェアがシステムに無い時，ソースからビルドする必要が生じる.
- 種々の依存ソフトウェアのバージョン制限を考え，適切なバージョンのソースをダウンロードし，適切な順序でビルドしていくのは面倒.
- ビルド作業を自動化する installOpenFOAM を作成.
  - bashスクリプトなので，大抵のLinuxシステムでそのまま実行可能.
  - 依存ソフトウェアのバージョンを指定可能.
  - 必要なソースを自動でダウンロード.
  - ビルドした依存ソフトウェアの再利用.
  - 複数のコンパイラバージョン，MPIバージョンの組み合わせを自動的にビルド.
- FOCUS，Reedbush-U，Oakforest-PACS，ITO，OCTOPUS等のシステムについて，環境設定済み．並列ビルド用のジョブスクリプトも有り.

オープンCAE学会のレポジトリで公開中