2020年8月31日 (WEB公開版: 2020年9月11日)

第1回 スーパーコンピュータ「不老」ユーザ会
「名古屋大学スーパーコンピュータ不老のOpenFOAMベンチマークテスト」

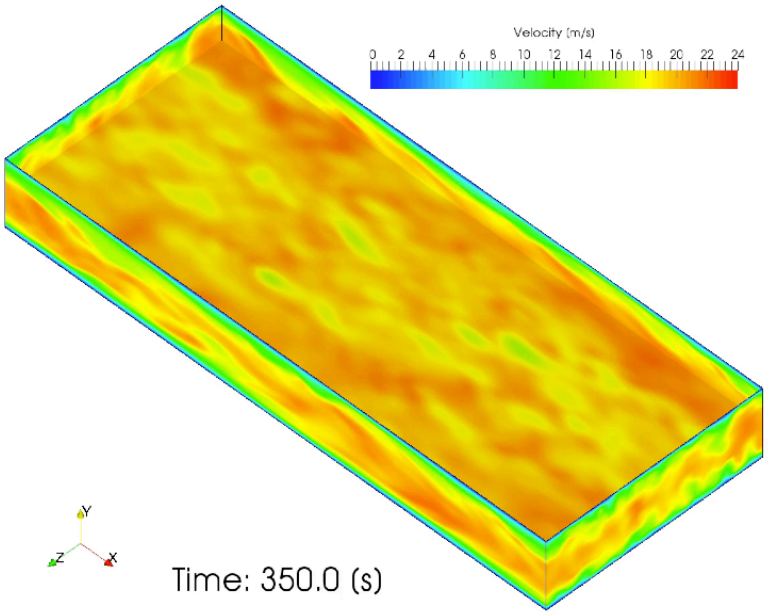
今野 雅 / IMANO Masashi

(株式会社OCAEL・東京大学情報基盤センター客員研究員)

1　はじめに

- 学際大規模情報基盤共同利用・共同研究拠点(JHPCN)の2020年度一般共同研究課題「機械学習を用いた風環境予測精度の向上と防災技術への応用」(課題代表者:高木洋平氏)での共同研究では，以下の計算機資源が供与されている．
  - 名古屋大学情報基盤センター 不老
  - 東京大学情報基盤センター Oakbridge-CX (以下，OBCX)
- 本課題では，建物設計時の支援となるよう，機械学習を用いて，建物表面のピーク風圧や歩行者高さでの風環境解析を短時間で予測するための基礎検討を行なっている．
- この機械学習を行うには，多数の建物形状に対して，教師データとなる非定常LES解析(格子数3〜24M程度)と，入力データとなる定常RAS解析(格子数1〜3M程度)を行う必要がある．
- 効率的な計算機資源の利用のため，不老Type I，Type II，CloudとOBCXでOpenFOAMベンチマークテストを行い，多数のLES，RAS解析を効率的に行うことができるシステムや計算条件を検討した．

2　オープンCAE学会チャネル流ベンチマーク



レイノルズ数 Re_τ	110
主流方向	一定の圧力勾配
主流・スパン方向	周期境界
ソルバ	pimpleFoam
乱流モデル	無し (laminar)
領域分割手法	scotch(周期境界面は同領域)
速度線型ソルバ	BiCG (前処理DILU)
圧力線型ソルバ	**PCG (前処理DIC)**

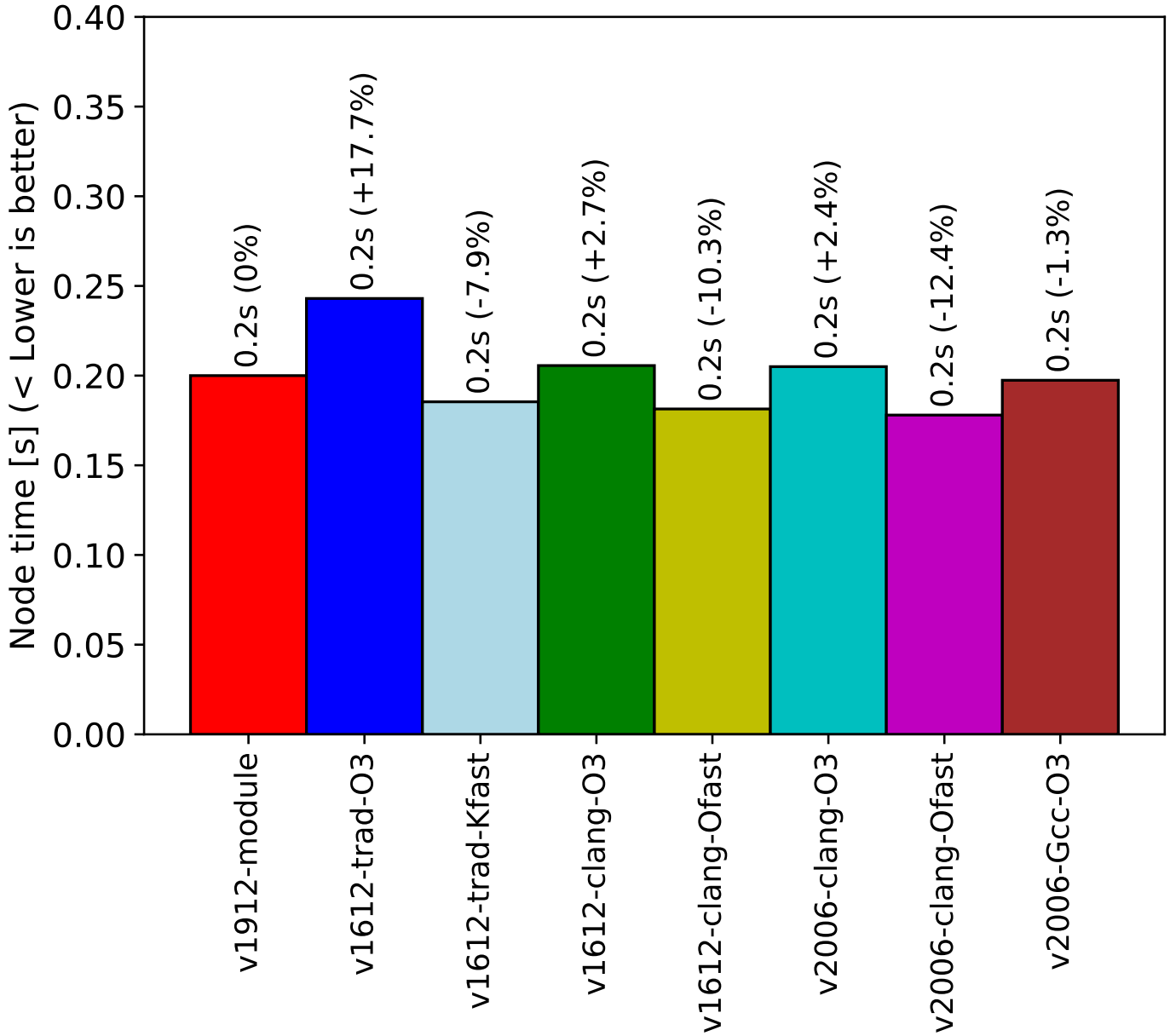
- 多くのスパコンやクラウドで計測結果があり，他のシステムとの比較が可能なオープンCAE学会のチャネル流ベンチマークで計測.
- 計測結果やジョブスクリプトは，チャネル流ベンチマークのレポジトリ [3] 参照.
- 0.37M，3M，24Mの格子数で検討.
- 0.37M，3Mの格子数は1〜51ステップ，24Mは1〜9ステップの解析時間(Execution Time)の平均値を算出し，さらに使用ノード数を掛けて，ノード時間[s]とした.
- PPN: Type I=48，Cloud=80, OBCX=56とフルコア，Type II=使用GPU数.

3　Type Iの実行ケース

ケース名	OpenFOAM	コンパイラ	主なオプション
v1912-module	v1912	Fcc 4.2.0	-Nclang -O3(*1)
v1612-trad-O3	v1612+	“	-Ntrad -O3
v1612-trad-Kfast	v1612+	“	-Ntrad -Kfastから-Kfp_relaxedを除外(*2)
v1612-clang-O3	v1612+	“	-Nclang -O3 -march=armv8.2-a+fp16+nosve(*3)
v1612-clang-fast	v1612+	“	-Nclang -Ofast -march=armv8.2-a+fp16+nosve
v2006-clang-O3	v2006	“	-Nclang -O3 -march=armv8.2-a+fp16+nosve
v2006-clang-fast	v2006	“	-Nclang -Ofast -march=armv8.2-a+fp16+nosve
v2006-Gcc-O3	v2006	Gcc 10.2.0	-O3(*4)

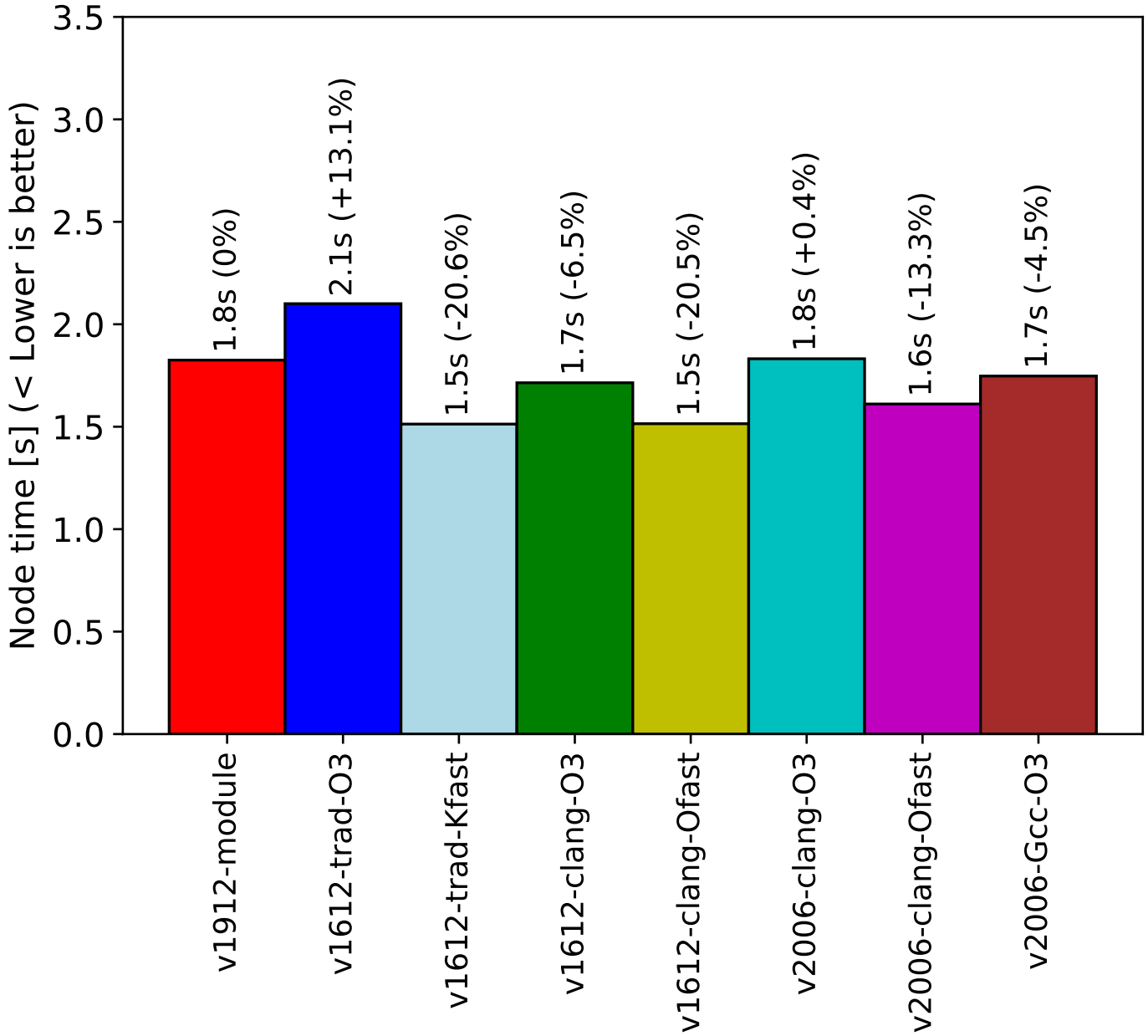
- (*1) spackでビルドされたType Iのopenfoam/1912_200506 module．ただし，sigFpe が発生する場合があるため，moduleの設定でtrapFpe を無効化.
- (*2) -Kfp_relaxedを指定するとソルバが実行不可のため.
- (*3) sveを有効するとソルバ並列実行時にsigFpeが発生するため.
- (*4) Gccでは-march=armv8-aや，fp，simd，lse，rdma，dotprodの有効化で，有意に高速化されることは無かった．またsveの有効化で並列計算不可.

4　0.37M格子での1ノード解析時のノード時間の比較



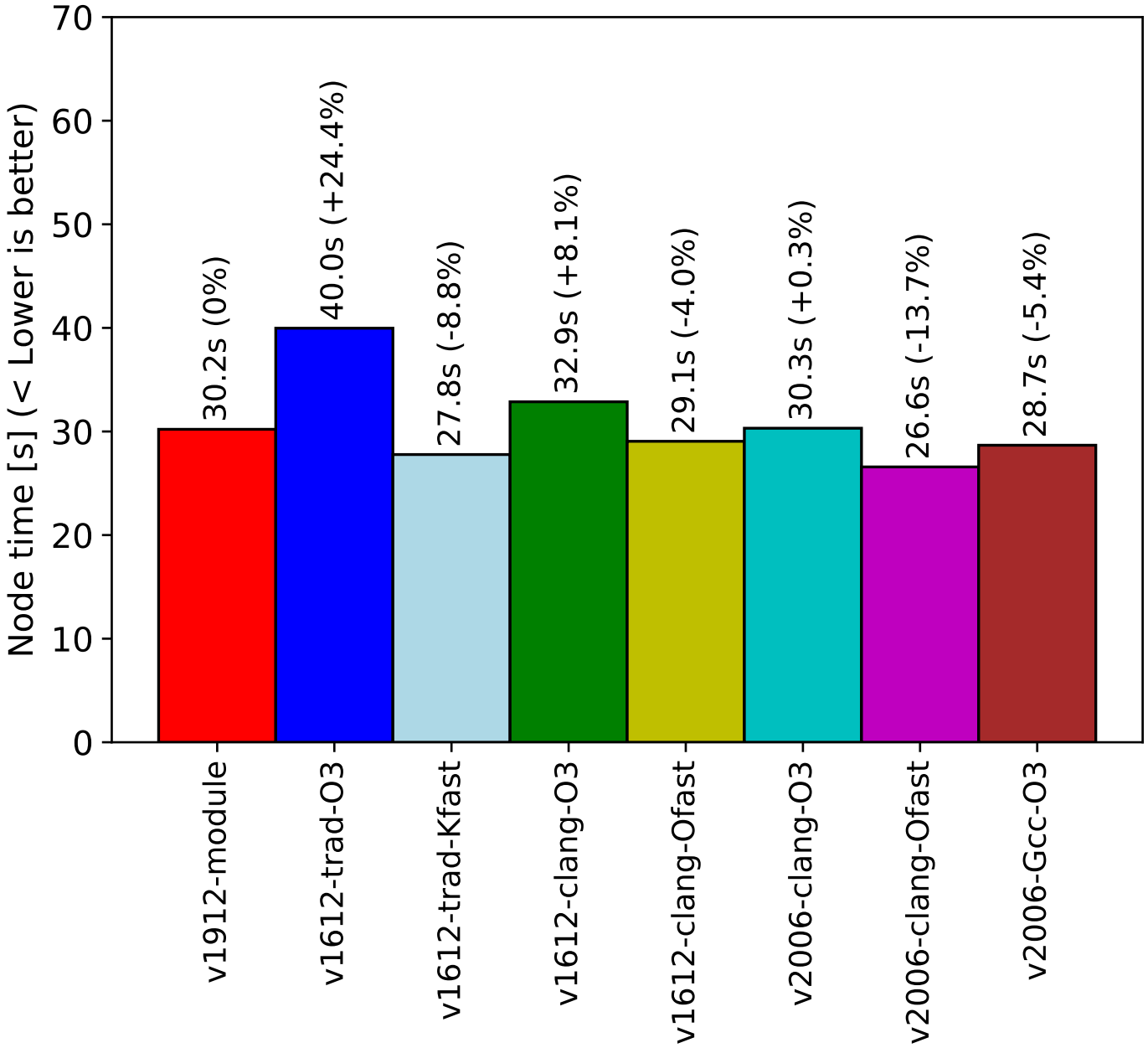
- -O3よりfastレベルの最適化のほうが高速.
- Fccのtradモードより，clandモードのほうが高速.
- Gccはclangモードの-O3最適化と同程度.

5　3M格子の1ノード解析時のノード時間の比較



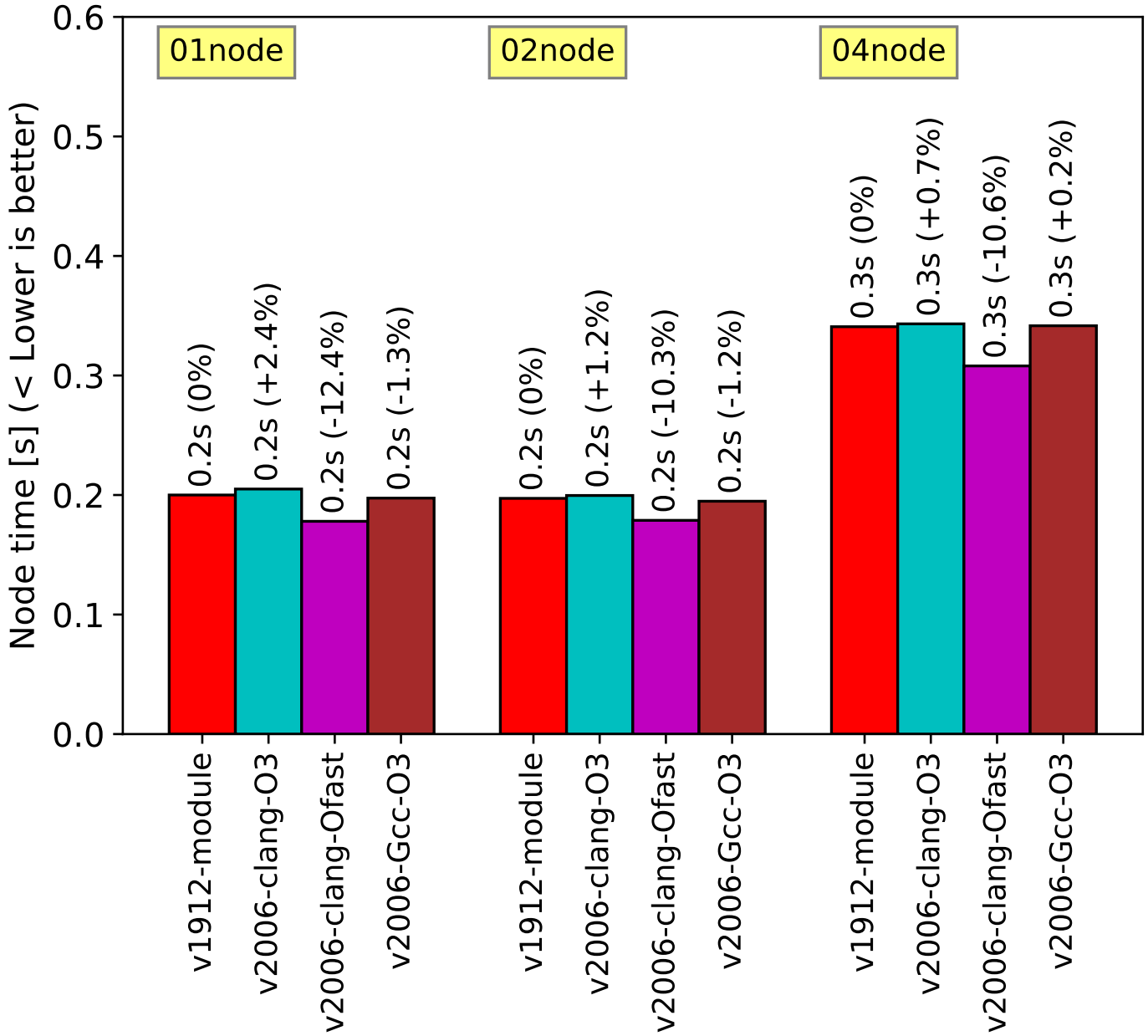
- 0.37M格子と同様の傾向.

6　24M格子の1ノード解析時のノード時間の比較



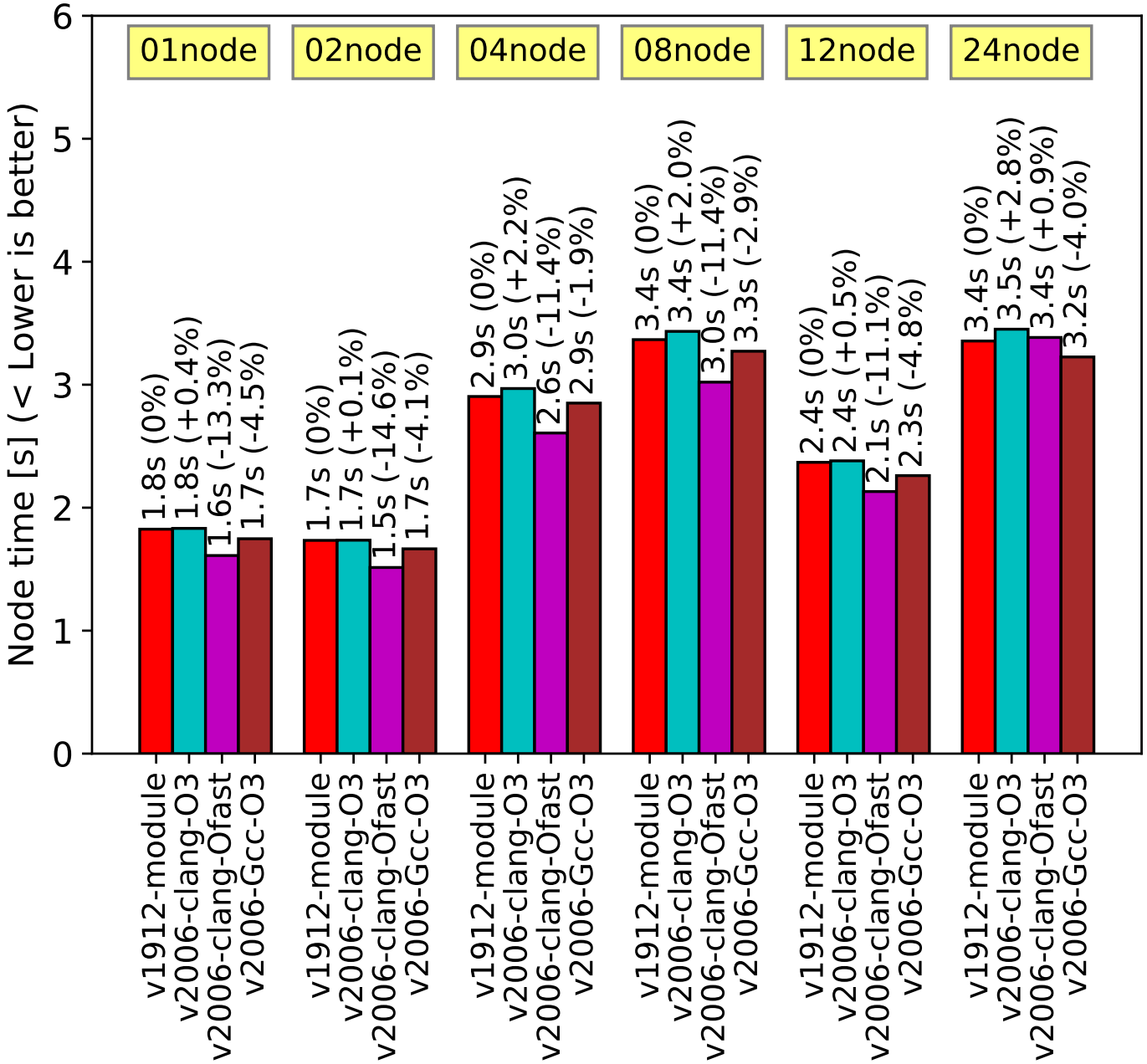
- 0.37M格子と同様の傾向.

7　0.37M格子でのノード時間の比較



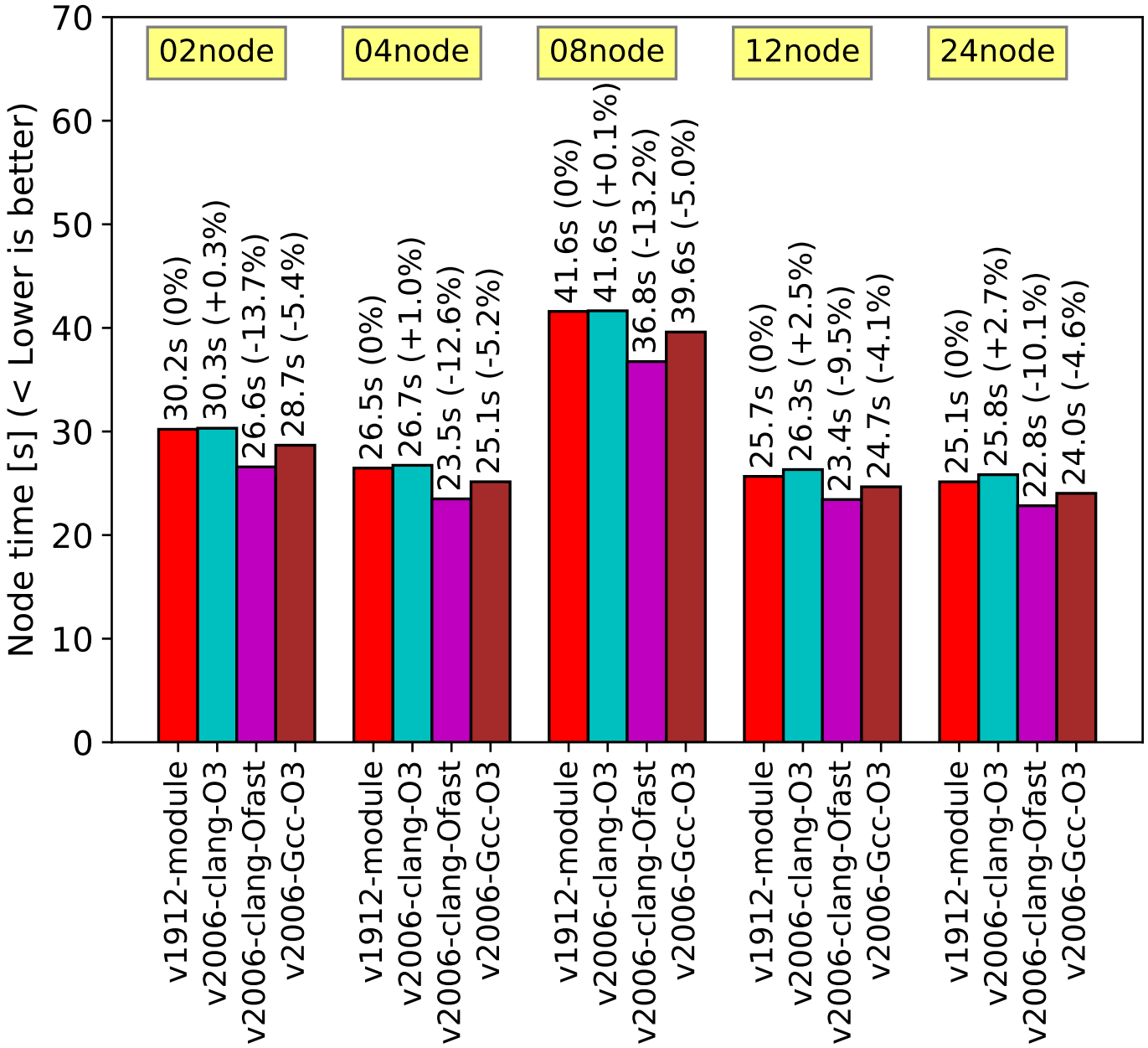
- 新しいバージョンのv1912とv2006のみで比較.
- 多ノードでもFccのclangモード-O3最適化はGccと同程度.
- Fcc clangモードのfast最適化が最速で，moduleより10%程度速い.
- nodeのshapeは無指定だが，default allocation modeはtorus.

8　3M格子でのノード時間の比較



- 0.37M格子と同様の傾向.

9　24M格子でのノード時間の比較



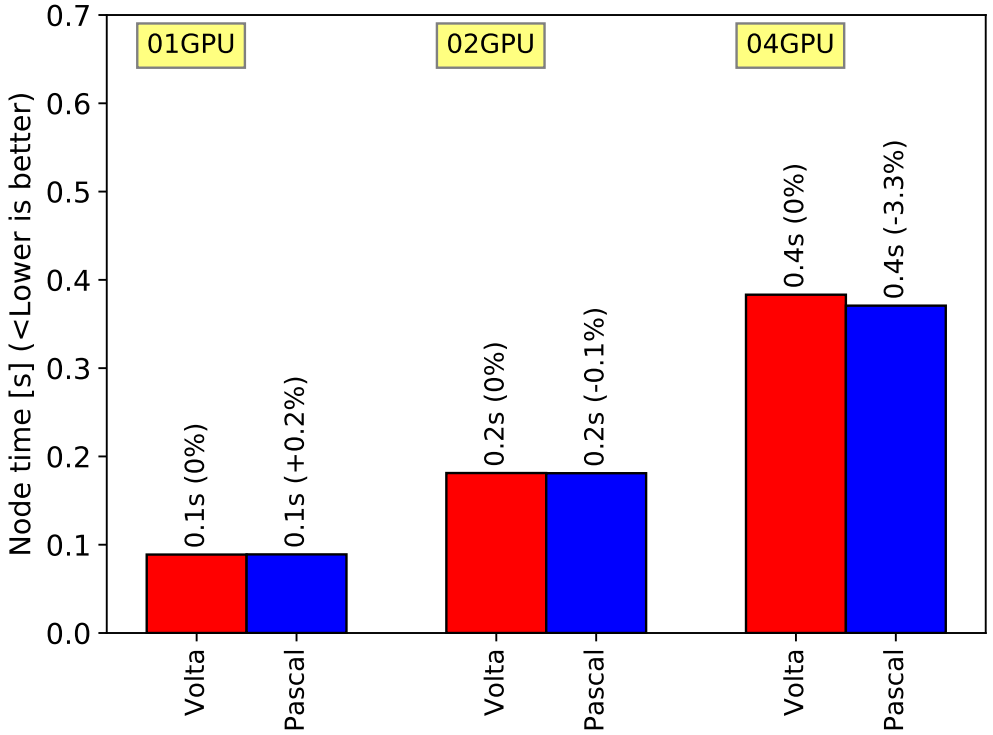
- 0.37M格子と同様の傾向.
- Type Iについては，今後は最速のFcc clangモードのfast最適化されたOpenFOAM v2006の結果のみを示す.

10　Type IIの実行ケース

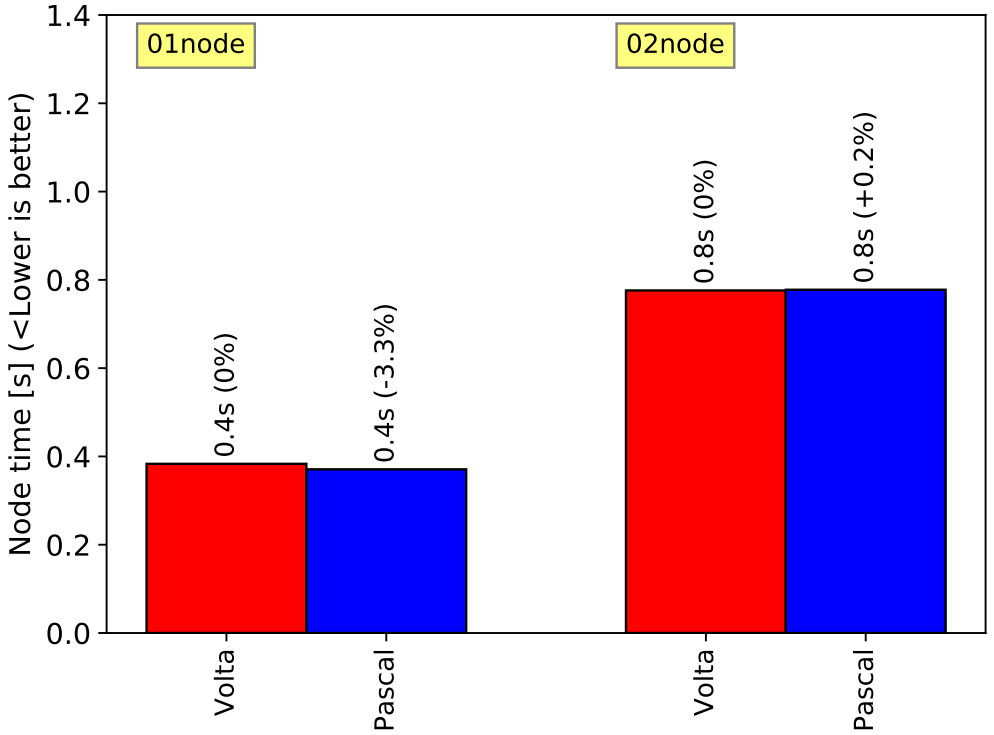
ケース名	OpenFOAM	コンパイラ	主なオプション
Volta	rapidCFD-dev	Nvcc 10.1.243	-O3 -gencode=arch=compute_70,code=sm_70
Pascal	“	“	-O3 -gencode=arch=compute_60,code=sm_70

- Type IIはGPU機なので，OpenFOAM-2.3.1のThrustライブラリによるGPU実装であるRapidCFDを用いた.
- PCGの前処理は，DICではなく，AINVが用いられる[2].
- Volta向けビルドとPascal向けビルドの比較も行なった.
- MPIライブラリは全てのmoduleのOpenMPI 4.0.3を使用.
- しかし，moduleのOpenMPI 4.0.3では，今回ビルドしたrapidCFD-dev(コミット:0fc3af285e3551234ffff9f9402c4c42f088af66)において，GPUDirect RDMAの機能を有効にするとソルバが実行できなかったので無効とした.
- なお，RISTの山岸らの既往の研究[4, 5]では，TSUBAME 3.0においてOpenMPI 2.1.1を用い，rapidCFDのGPUDirect RDMAの有無を比較しているが，1〜64GPUの範囲では特に有意な差が無い結果であった.

11　0.37M格子のノード時間の比較



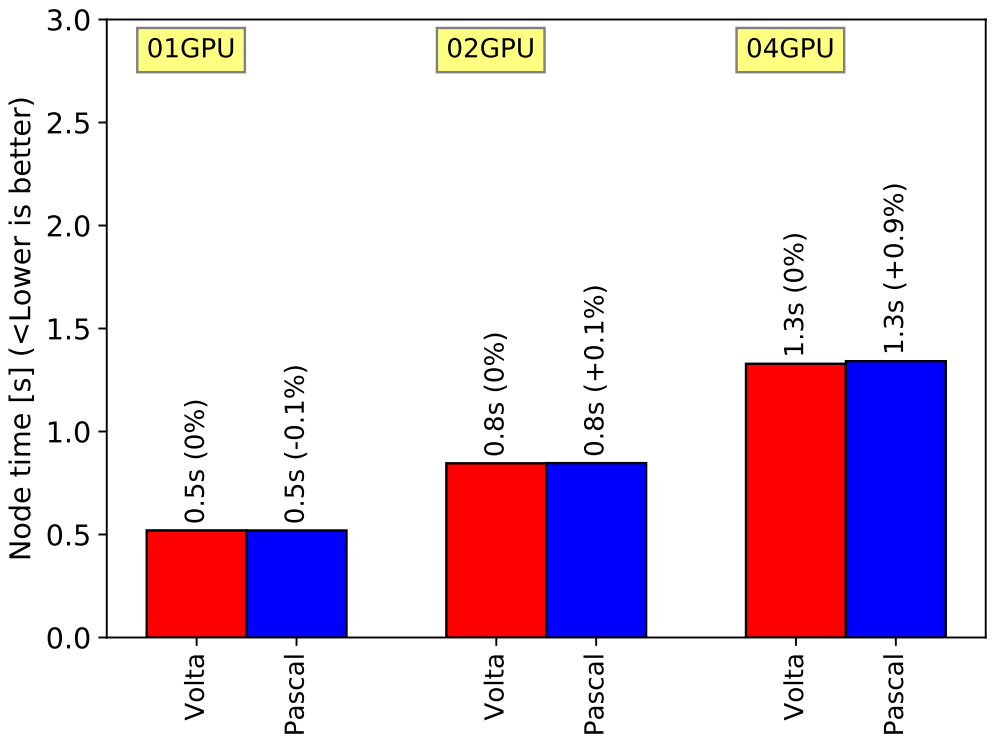
1ノードで GPU 数を変更



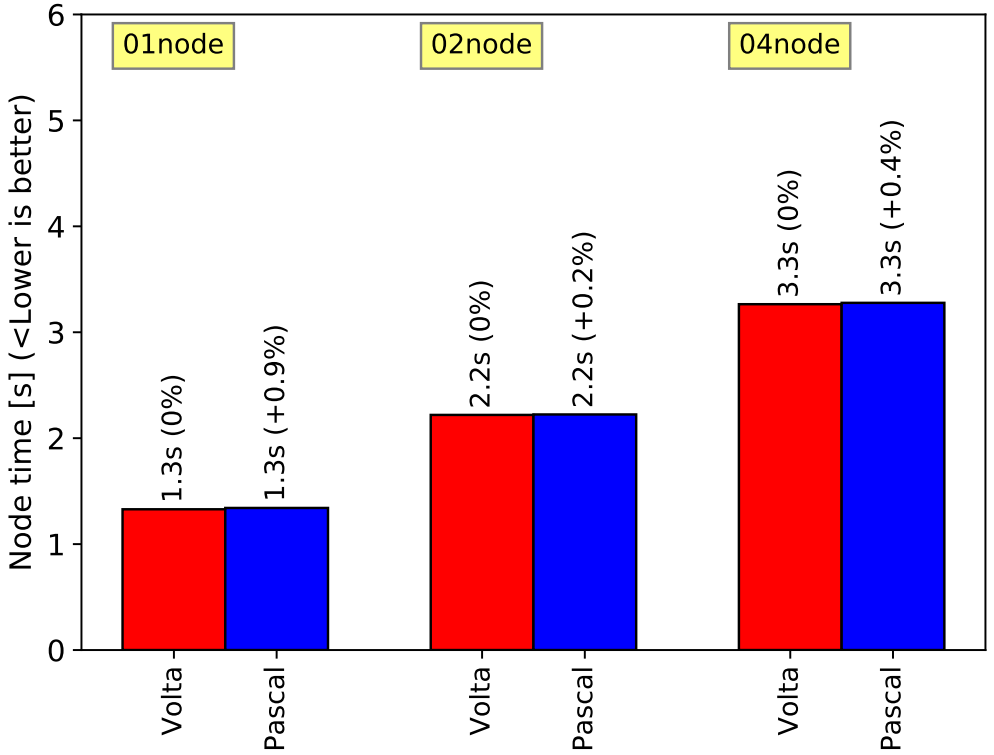
1ノードで 4GPU でノード数を変更

- 1GPUのジョブはノードを共有するcx-shareリソースグループで実行し，その他のジョブはノードを占有.
- Volta向けとPascal向けビルドで有意な差は無い.
- ノード内でGPUを増した場合でも計算時間が概ね一定で，ノード時間がGPU数に比例.
- ノード数が増えると，ノード時間が大きく増加し，スケールしない.
- 格子数が少ないため，ほぼ通信に費されている.

12　3M格子のノード時間の比較



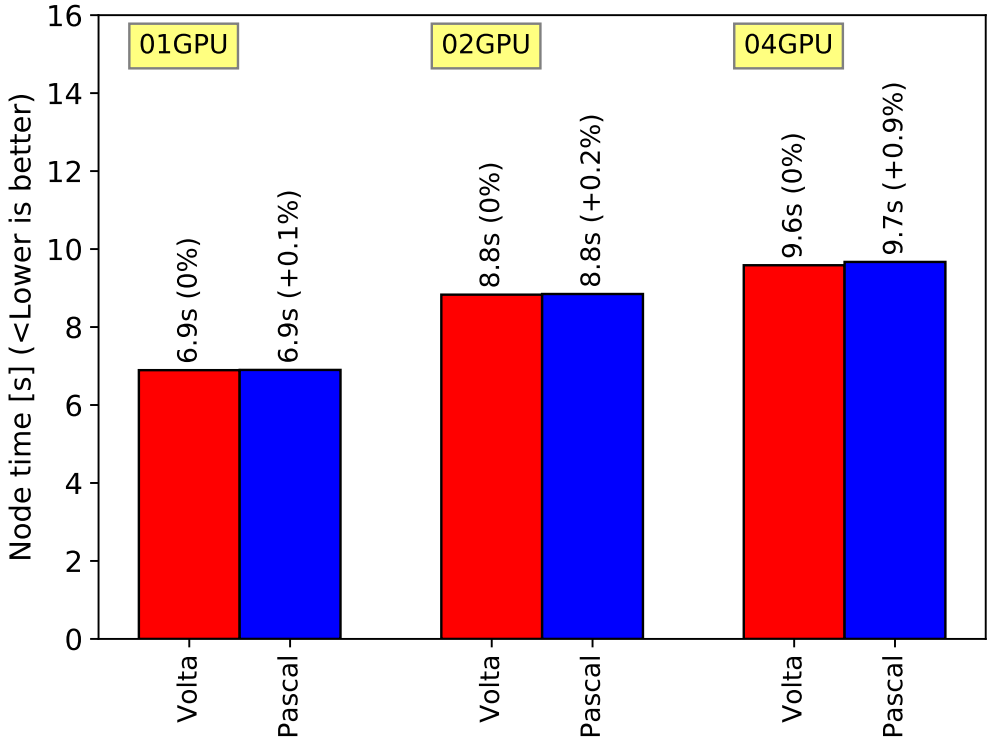
1ノードでGPU数を変更



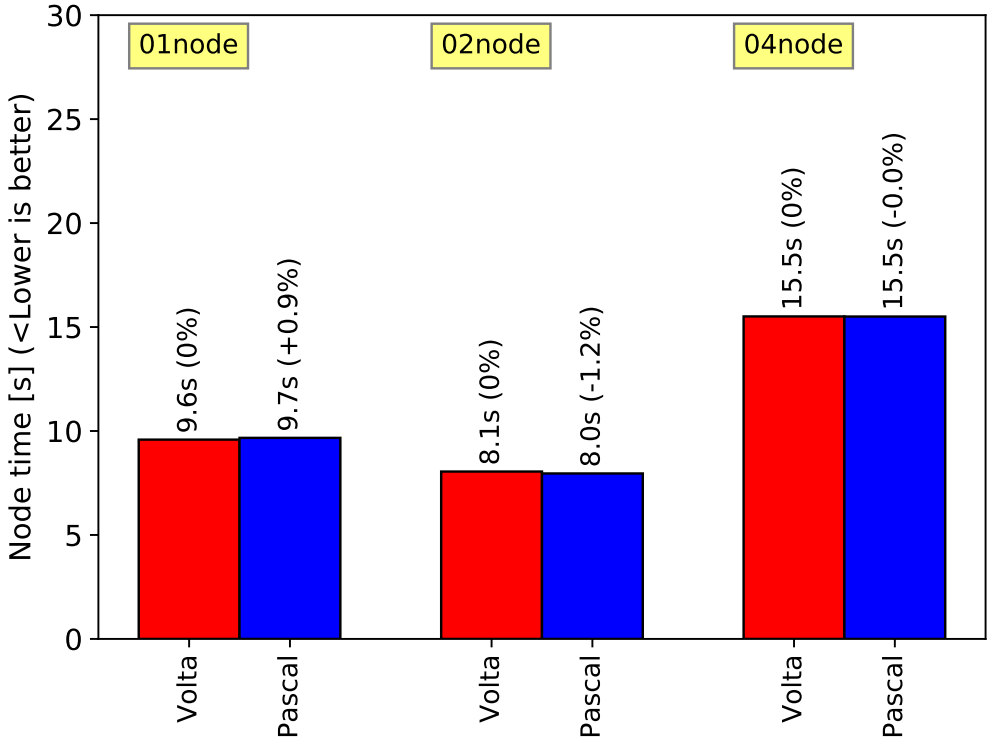
1ノードで4GPUでノード数を変更

- 0.37M格子と概ね同様の傾向.
- まだ格子数が少なく，概ね通信に費されている.

13　24M格子のノード時間の比較



1ノードで**GPU**数を変更



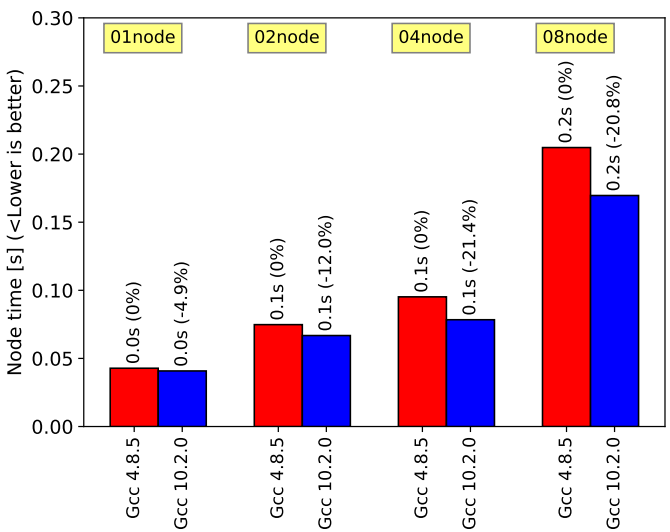
1ノードで**4GPU**でノード数を変更

- ノード内でGPUを増した場合，1GPUベースでの並列化効率は，2GPUで約77%，4GPUで約70% であり，ある程度スケール.
- 1ノードより2ノードのほうがノード時間が減少しスーパースケールするが，4ノードではノード時間が大幅に増加.
- Votlta向けとPascal向けビルドで有意な性能差が無いので，今後はVotlta向けビルドの結果のみを示す.

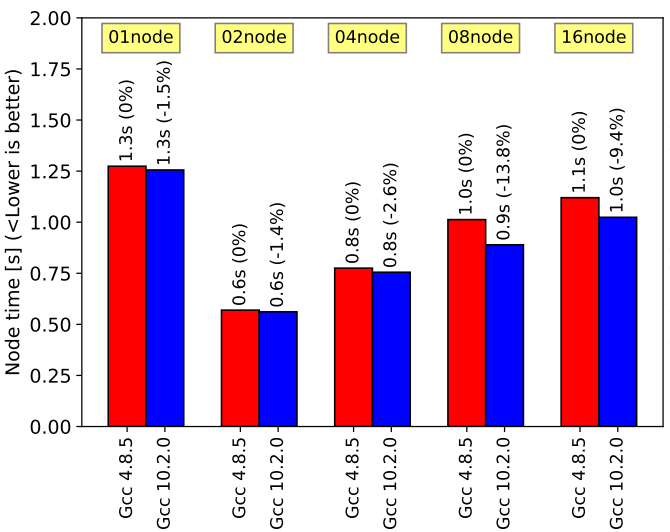
14　CloudおよびOBCXの実行ケース

ケース名	OpenFOAM	コンパイラ	主なオプション	システム
Flow Cloud	v2006	Gcc 10.2.0	-O3	Cloud
OBCX	“	“	“	OBCX

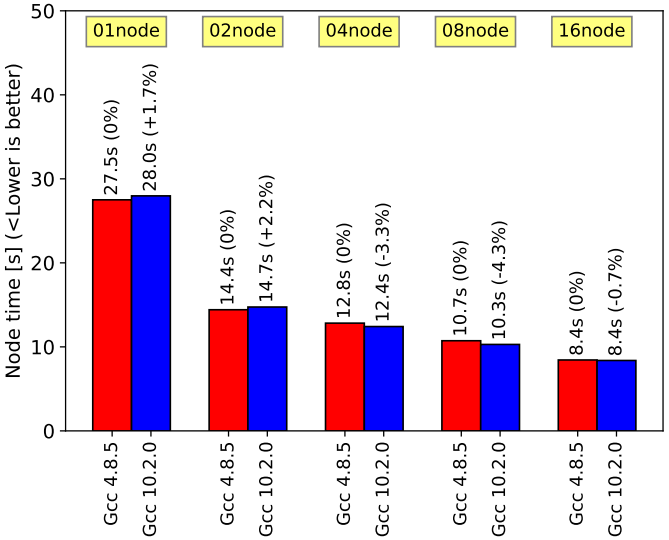
- CloudやOBCXでは，Gcc 10.2を用いてビルドしたv2006を用いたが，Gccの最適化オプションについても，デフォルトの最適化オプションである-O3を用いた.
- Cloudのシステムコンパイラである Gcc 4.8.5でもビルドを行なったが，以下のCloudでの結果の通り，概ね10.2のほうが高速だったので，今後は10.2の結果のみ示す.



0.37M格子

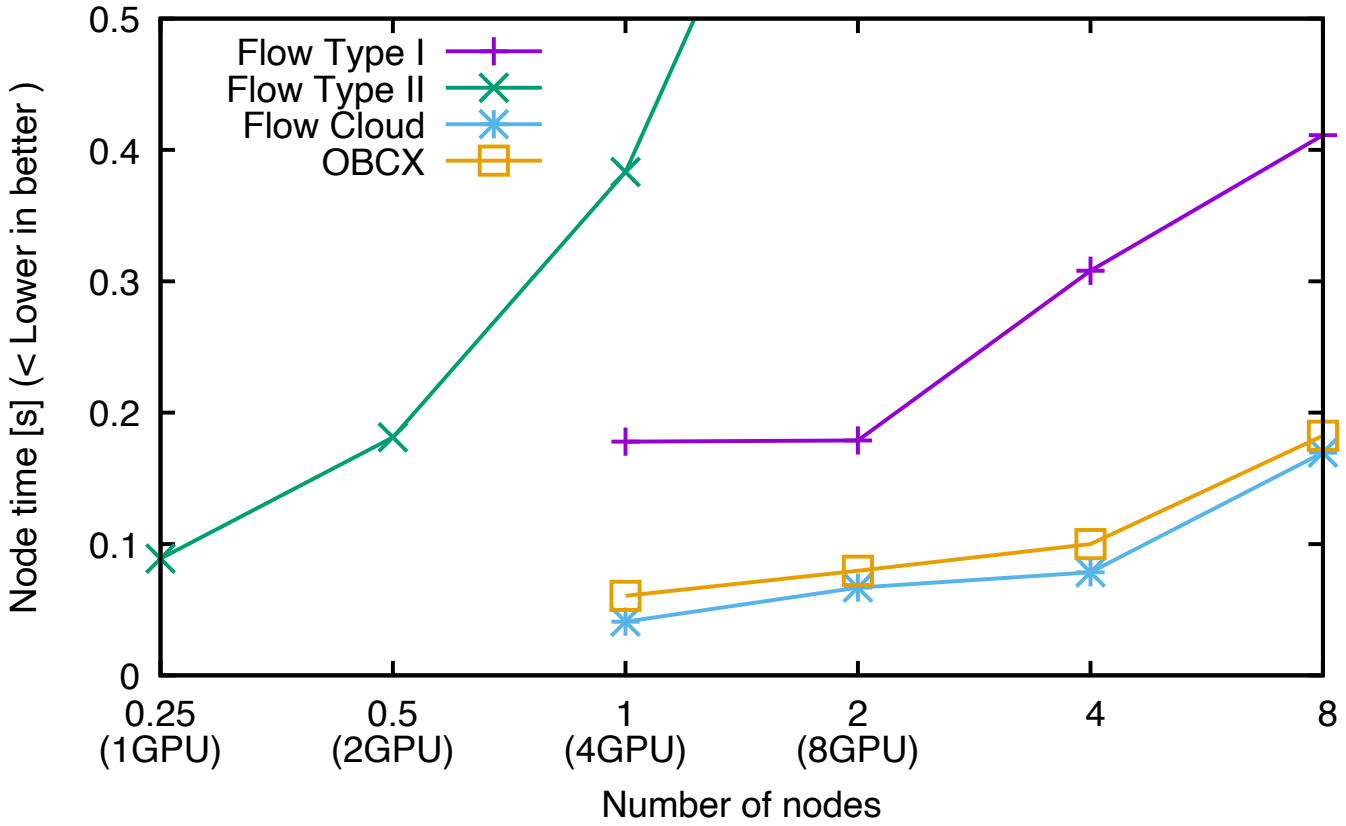


3M格子



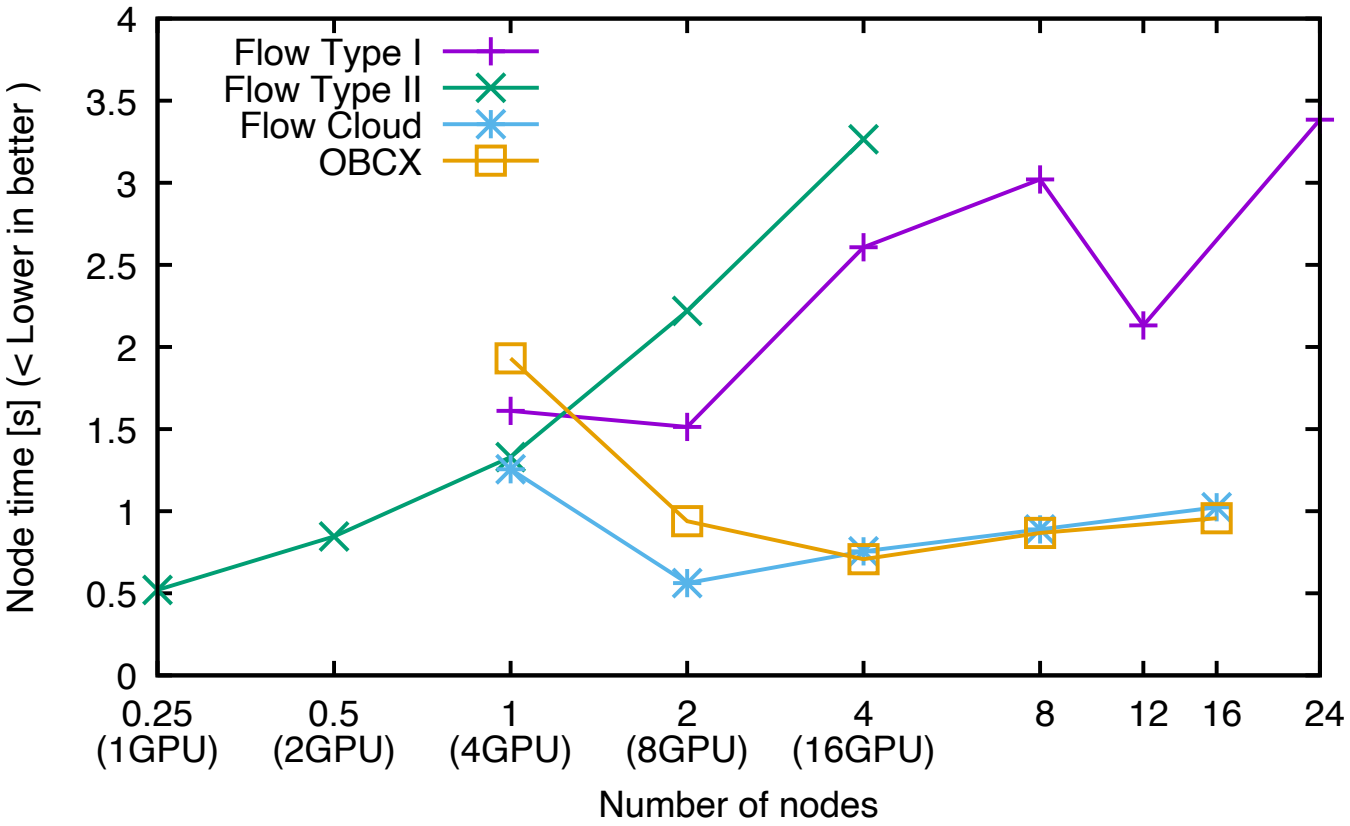
24M格子

15　各システムでの0.37M格子のノード時間の比較



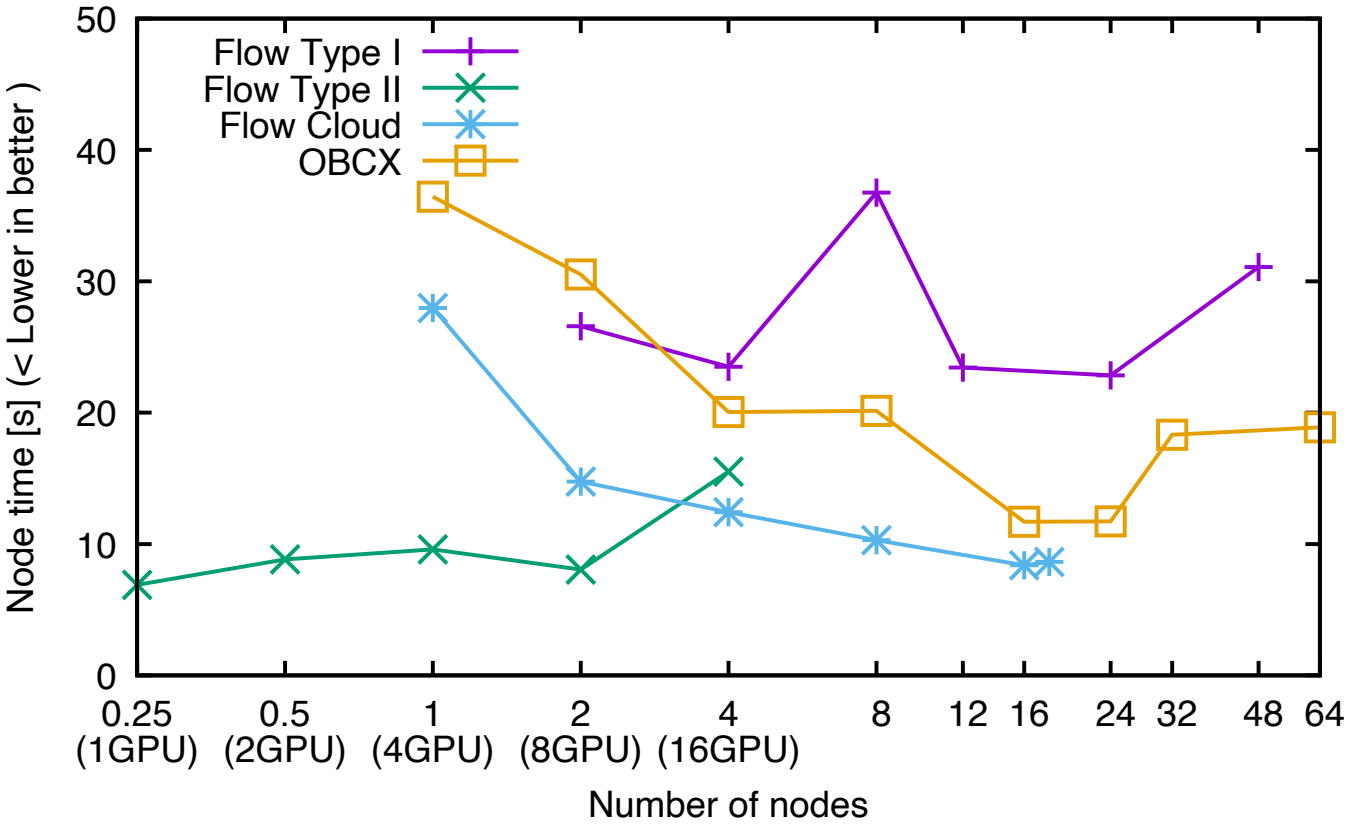
- Type I，Cloud，OBCXともに，2ノードまでノード時間がほぼ一定でスケール.
- Cloud, OBCX, Type IIの1GPU，Type Iの順にノード時間が増加.
- Type IIの1GPU，または，Cloud・OBCXの1ノードが効率的.
- 定常解析では領域分割のコストが高いので，領域分割が不要な1GPUが最も効率的.

16　各システムでの3M格子のノード時間の比較



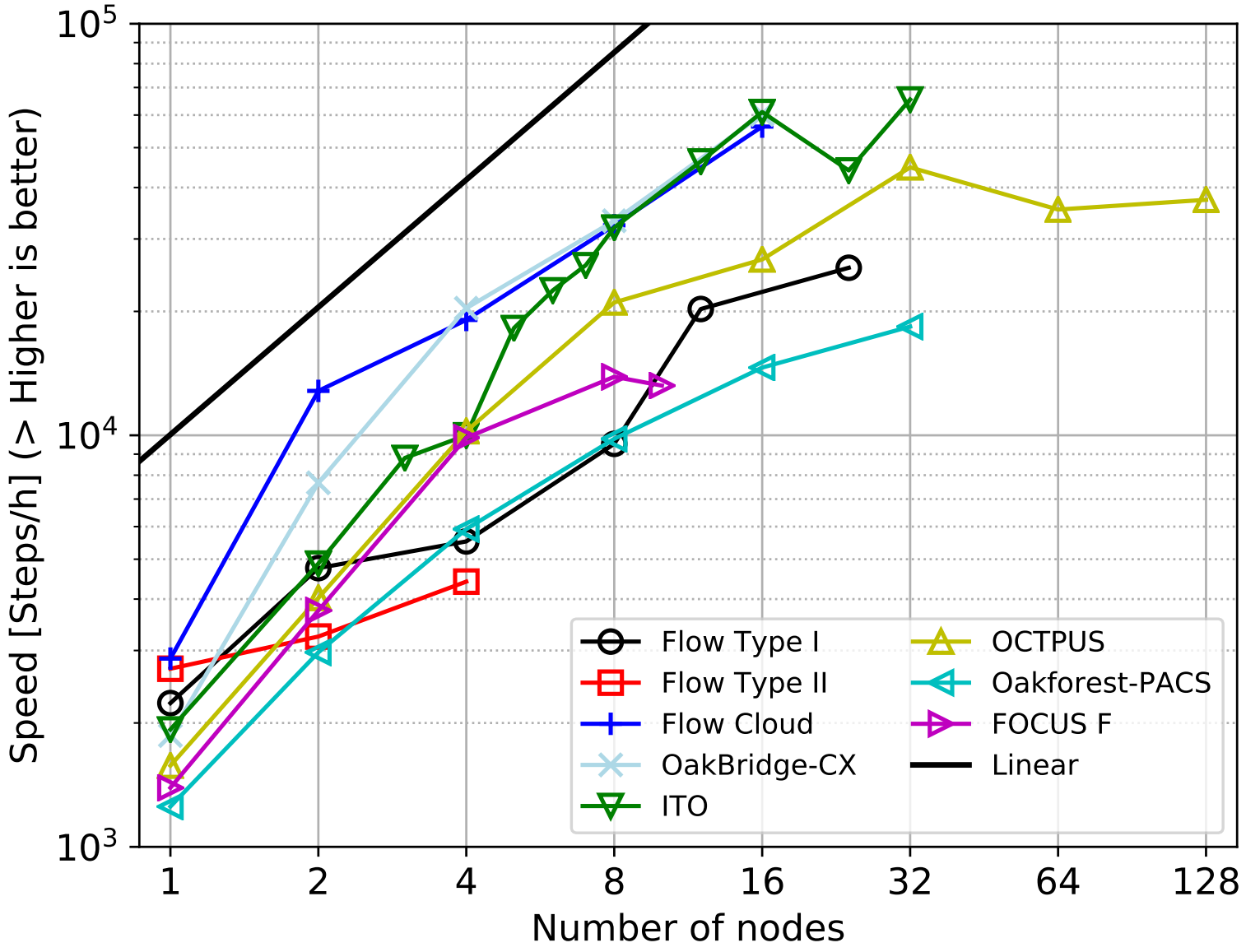
- Type Iは，2ノードまでスケール．1Tofu単位の12ノードで一旦減少．
- CloudとOBCXは，2〜8ノードまで概ねスケール．
- Type IIの1GPUは，Cloud 2ノードと同程度の最小値．
- 定常解析ならType IIの1GPU，非定常LES解析ならCloud・OBCXの2〜8ノードが効率的．

17　各システムでの24M格子のノード時間の比較



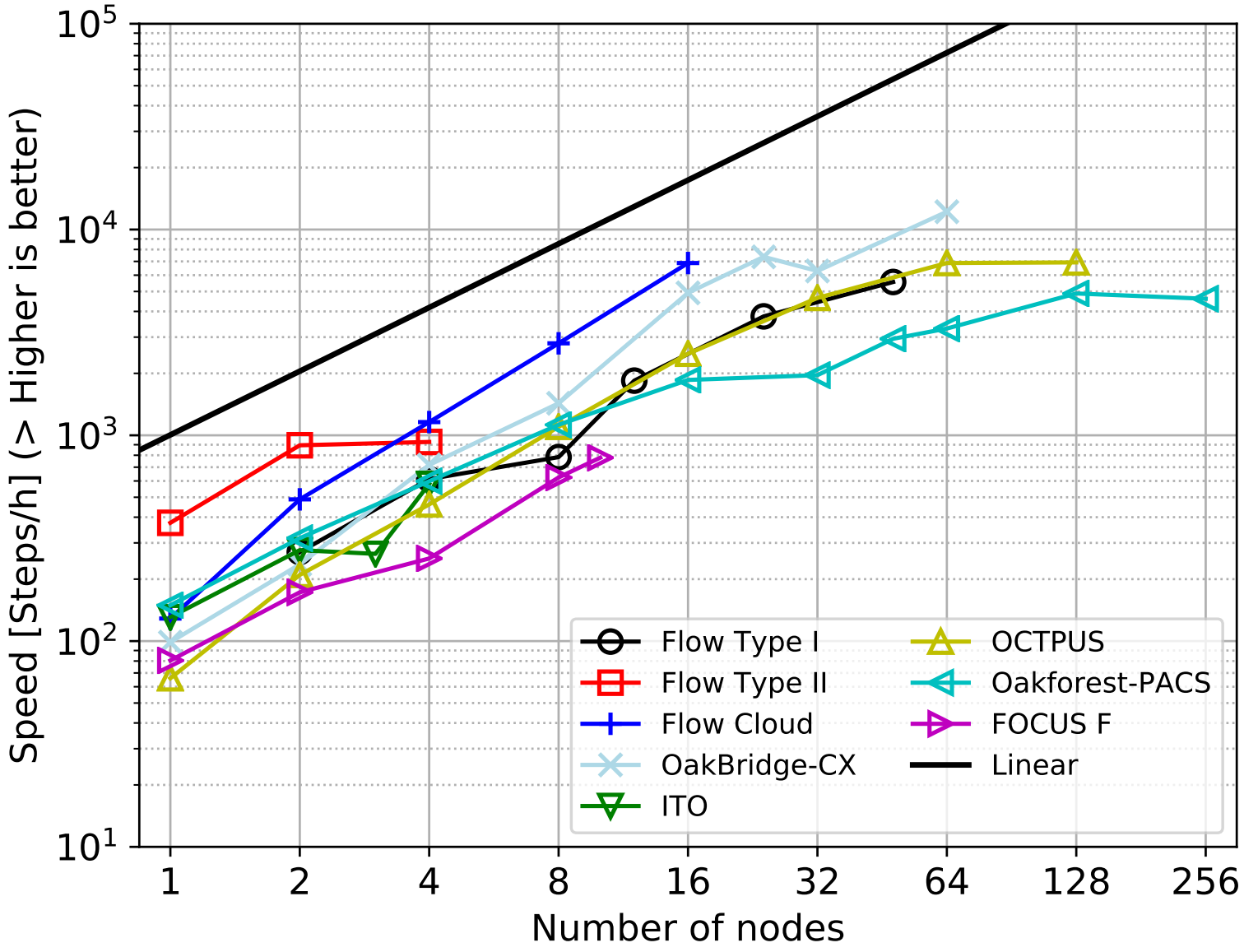
- CloudとOBCXは，16ノードまでノード時間が減少し，スーパースケール．
- Type Iは8ノードを除き，24ノードまでスケールするが，CloudやOBCXより遅い．
- Type IIは2ノードまではスケールし，Cloudよりもノード時間が大幅に小さい．
- 定常解析はType IIの1GPUまたは2ノード，非定常LES解析はCloud 16ノード，OBCX 16〜24ノードが効率的．Cloud(100ノード)とOBCX(1368ノード)の混雑も要考慮．

18　各システムでの3M格子の解析スピードの比較



- Cloudが最速
- Type IIは1ノードではCloudと同程度だが，多ノードでは遅くなる．
- Type Iは，概ね大阪大学OCTOPUS(Intel Xeon 6126[Skylake] 24コア/ノード)と同程度．

19　各システムでの24M格子の解析スピードの比較



- 1，2ノードを除けばCloudが最速.
- Type IIは1，2ノードで最速だが，4ノードではCloudより少し劣る.
- Type Iは，概ね大阪大学OCTOPUS(Intel Xeon 6126[Skylake] 24コア/ノード)と同程度.

20　まとめ

- JHPCN課題で供与されている計算資源の不老のType I，II，Cloud，および，東京大学OBCXについて，オープンCAE学会のチャネル流ベンチマークを行なった.
- 機械学習のために実行する必要がある多数の建物形状に対する非定常LES解析や定常RAS解析に対して，効率的なシステムやノード数等を選定した.
- 今後はより大きな格子数でチャネル流ベンチマークを行い，Type I，II，Cloud，OBCXの特性を検討したい.

謝辞

- Fccのtradモードによる OpenFOAMのビルドは非常に時間がかかる上，tradモードでのコンパイルはメモリ使用量が多いため，Type Iの計算ノード1台を用いてビルドした場合，メモリ不足で異常終了しない並列数の上限は3並列であった．そこで，本研究では，多数のビルドオプションを変更したベンチマークテストを効率良く行うため，`auto_wmake`[1]を用いて，ベンチマークテストで用いるソルバと関数，および，その依存ライブラリのみをビルドした．ここに`auto_wmake`の作者のkurenaif氏に感謝する．
- 本研究は，学際大規模情報基盤共同利用・共同研究拠点の支援による(課題番号: jh200064-NAH).

Copyright

参考文献

[1] auto_wmake. Accessed: 2020-09-11. URL: https://github.com/kurenaif/auto_wmake/.

[2] Ilya B Labutin and Irina V Surodina. Algorithm for sparse approximate inverse preconditioners in the conjugate gradient method. *Reliable Computing*, 19:121, 2013. URL: https://hgpu.org/?p=11066.

[3] オープンCAE学会OpenFOAMベンチマーク. Accessed: 2020-09-11. URL: https://gitlab.com/OpenCAE/OpenFOAM-BenchmarkTest/.

[4] 山岸孝輝, 井上義昭, 青柳哲雄, and 浅見曉. GPU・メニーコアにおけるOpenFOAMの高度化支援紹介. In 第*1*回*CAE*ワークショップ～流体・構造解析アプリケーションを中心に～. 高度情報科学技術研究機構, Dec 2017. URL: https://www.hpci-office.jp/pages/ws_cae_171206.

[5] 山岸孝輝, 井上義昭, 青柳哲雄, and 浅見曉. OpenFOAMの GPUによる高速化事例紹介. In 平成*29*年度 高速化ワークショップ～「京」を中核とする*HPCI*におけるメニーコアの活用を目指して～. 高度情報科学技術研究機構, Mar 2018. URL: https://www.hpci-office.jp/pages/ws_hpc_180323.