

EasyISTR／固有値ベンチマーク (その2)

Windows 10、Linux (VMwareゲストと実マシン)

前回からFrontISTR 5.1.1とIntel-MKLの結果を追加
第10世代Intel CPUでの結果を参考に追加

概要

ベンチマーク 実施理由

- Windows用公式バイナリ（MS-MPIバージョン）は遅いと感じた
- Ubuntu, Debianではリポジトリからintel-mklをインストールできる
- MKL組み込みでどのくらい速くなるか試した

結果

- 並列計算をしない場合、Windows版はSerial ver.が良い
- MUMPSはCGに比べて、20～40倍高速
- MKLを組み込んだMUMPSが最も高速
- CGはMKL組み込みの効果なかった

ベンチマークに使用したPCのスペック

- CPU : Intel Core i5 6600K 3.5GHz 4コア
デスクトップPCです
- RAM: 32GB
- ドライブ
Cドライブ: SSD(480GB)、Eドライブ: HDD(3TB) . . . EasyISTRとSalome
他に1TBのHDDがあり、Debian 10をインストールしています
(BIOSからブートドライブを選べばネイティブ環境で立ち上がります)
- グラフィック : NVIDIA GTX 960
- 参考にCore i7 (第10世代、第4世代) のノートパソコンのデータも加えました

Linux環境でのFrontISTRの Intel-mkl組み込み

DebianとUbuntuで確認しました

FrontISTRへのMKL組み込み方法（その1）

小川氏のver.4.6バイナリを使用する方法^[1]を基本とし、intel-mkl関係で工夫した。

1. リポジトリからライブラリの導入

```
$ sudo apt install build-essential
```

```
$ sudo apt install libtrilinos-ml-dev
```

```
$ sudo apt install libtrilinos-aztecoo-dev libtrilinos-zoltan-dev
```

```
$ sudo apt install libmumps-dev libmetis-dev
```

```
$ sudo apt install intel-mkl # Debian 10では、non-free
```

2. REVOCAP_Refinerをコンパイル

```
$ tar xvf REVOCAP_Refiner-1.1.04.tar.gz
```

```
$ cd REVOCAP_Refiner-1.1.04
```

```
$ make
```

FrontISTRへのMKL組み込み方法（その2）

3. FrontISTRのビルド(その1)

CMAKEではうまくいかなかったので、Makefile.confでビルドした。

```
$ tar xvf FrontISTR-v5.1.1.tar.gz
```

```
$ cd FrontISTR-v5.1.1
```

4. FrontISTRのビルド（その2）

Makefile.conf.orgを編集して、Makefile.confとして保存する。

Intel-mklに関する設定を次ページに示す。

FrontISTRへのMKL組み込み方法（その3）

Makefile.con抜粋（Intel-mklが関係する部分）

MKL PARDISO

```
MKLDIR    = /usr  
MKLINCDIR = $(MKLDIR)/include/mkl  
MKLLIBDIR = $(MKLDIR)/lib/x86_64-linux-gnu
```

Fortran compiler settings

```
F90      = mpif90 -fopenmp  
F90FLAGS = -m64 -I/usr/include/mkl  
F90LDFLAGS = -lstdc++ -L/usr/lib/x86_64-linux-gnu -lmkl_gf_lp64 -lmkl_gnu_thread -lmkl_core -lgomp  
-lpthread -lm -ldl  
F90OPTFLAGS = -O2  
F90FPP      = -cpp  
F90LINKER   = mpif90 -fopenmp
```

FrontISTRへのMKL組み込み方法（その4）

5. FrontISTRのビルド(その3)

```
$ ./setup.sh -p --with-tools --with-refiner --with-metis  
--with-mumps --with-mkl --with-ml
```

6. FrontISTRのビルド（その4）

コンパイルしてバイナリを作成する

```
$ make
```

```
$ make install
```

Makefile.confで設定したフォルダにバイナリが保存される

Windows上のビルドについて

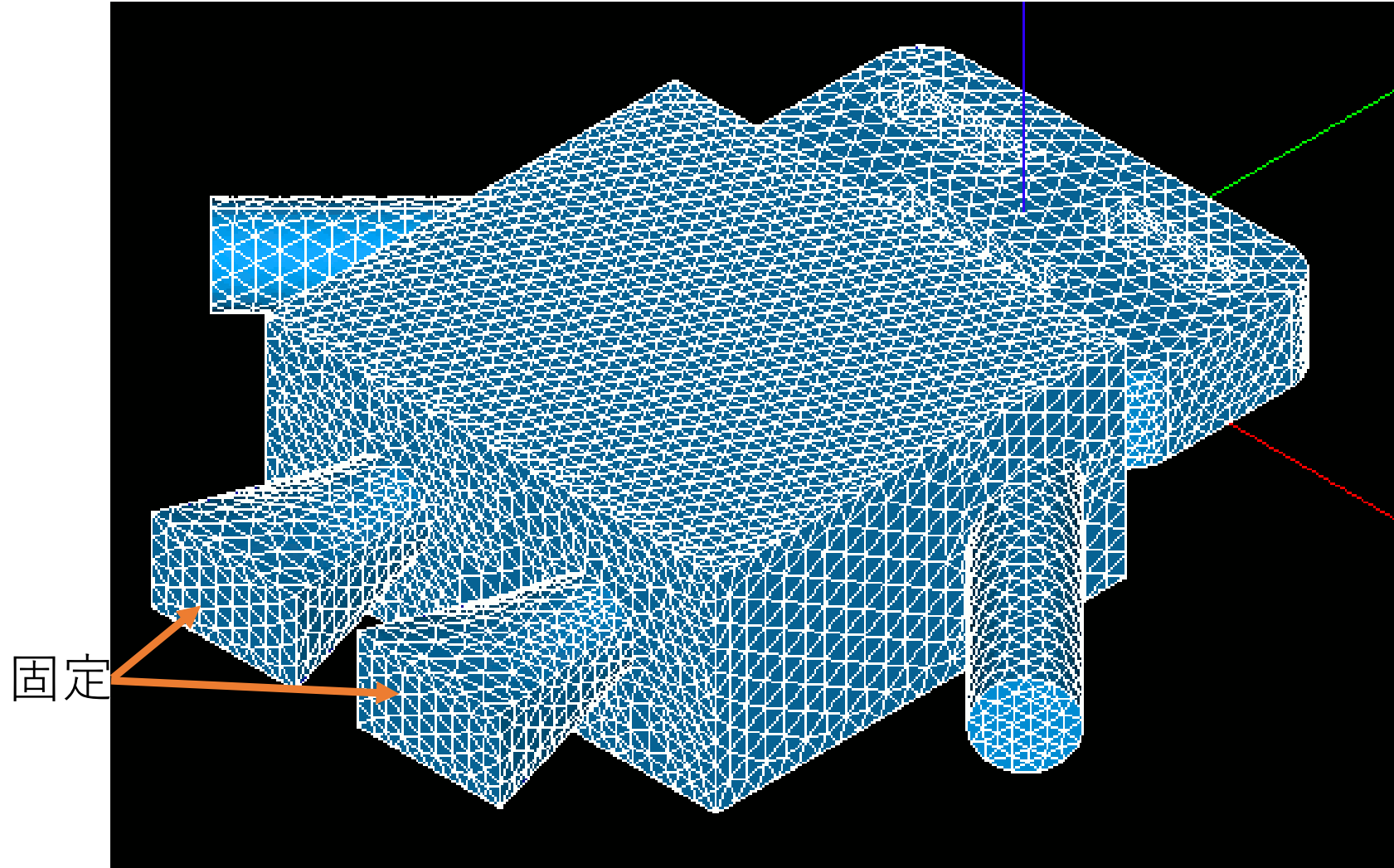
- 公式サイトの情報では、MSYS2を使う環境
- リポジトリには、Intel-mklとtrilinosのバイナリがないので、敷居が高い
- MSYS2は、ローリングリリースなので、ライブラリのメジャーアップデートが入ったときに壊れる可能性がある
- 以上の理由でビルドはあきらめました

固有解析ベンチマーク結果

解析に用いたメッシュ

四面体2次要素 (前回発表した^[2]のと同じモデルです。)

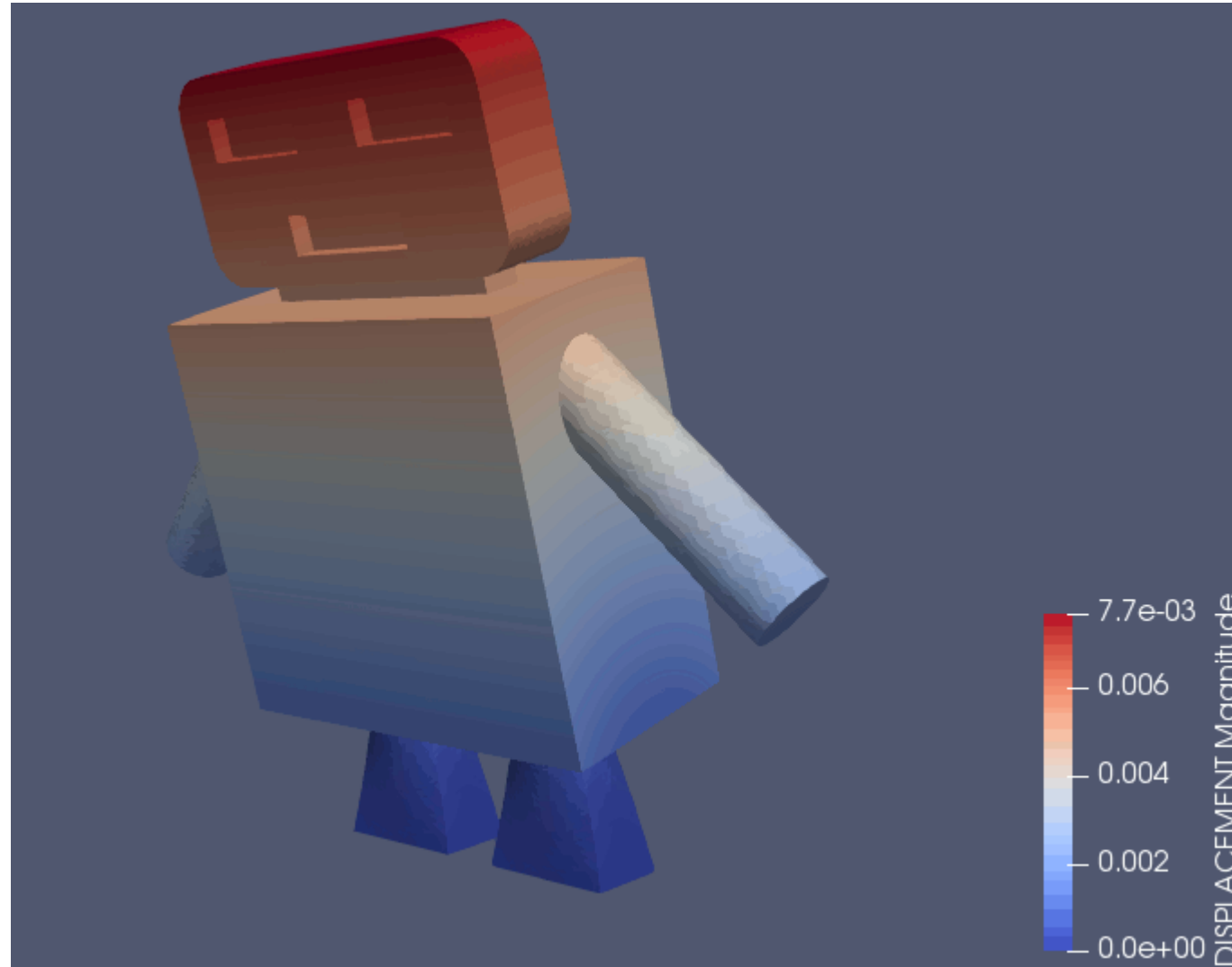
Nodes 60786 elements type:342 37317



FrontISTRのソルバー

- FrontISTRの線形ソルバーと前処理^[3]によると、よく使うのは「CG」と「MUMPS」とのことです
- Windows 10、仮想マシンのUbuntuと実マシンのDebian 10の環境でこの2種類のソルバーでの解析時間を比較しました
- メッシュ分割による並列計算は、適用していません

デフォルトの5次モードまで解析（図は1次モードの変位）



Windows バイナリでのベンチマーク結果 (4コア)

| | FrontISTR 4.6 Windows 10 Pro 公式バイナリを使用 | | FrontISTR 5.1.1、Windows 10 Pro、公式バイナリを使用 | | | | | |
|--------------|--|---------------|--|---------------|------------------|---------------|-----------------|---------------|
| | | | MS-MPI ver. | | | | Serial ver. | |
| | 旧MPI | | 新MPI | | | | | |
| | CG | MUMPS | CG | MUMPS | CG | MUMPS | CG | MUMPS |
| Total (秒) | 603.48 (21倍) | 28.77 (1倍) | 977.03 (20倍) | 48.06 (1倍) | 1054.88 (35倍) | 29.73 (1倍) | 551.29 (20倍) | 27.59 (1倍) |
| Pre (秒) | 2.41 | 2.41 | 2.31 | 2.31 | 2.07 | 2.07 | 2.11 | 2.14 |
| Solve (秒) | 601.07 | 26.40 | 974.73 | 45.75 | 1052.81 | 27.66 | 549.18 | 25.45 |

- MS-MPIのバージョンに注意が必要
(バージョンにより、MUMPS、CGの速度が変わる)
- CG (MS-MPI ver.は特に遅い) と比べて、MUMPSは20~35倍高速
- 並列計算しない場合は、Serial ver.が良い

Debian 10実マシン環境 (4コア、BIOSで切り替え)

| | Frontistr 4.6 openblas | | Frontistr 5.1.1 openblas | | Frontistr 5.1.1 Intel-mkl | |
|--------------|---------------------------|---------------|-----------------------------|---------------|------------------------------|---------------|
| | CG | MUMPS | CG | MUMPS | CG | MUMPS |
| Total (秒) | 556.14 (35倍) | 15.74 (1倍) | 490.57 (31倍) | 15.86 (1倍) | 490.89 (36倍) | 13.59 (1倍) |
| Pre (秒) | 1.66 | 1.72 | 1.78 | 1.78 | 1.77 | 1.76 |
| Solve (秒) | 554.49 | 14.02 | 488.79 | 14.08 | 489.12 | 11.82 |

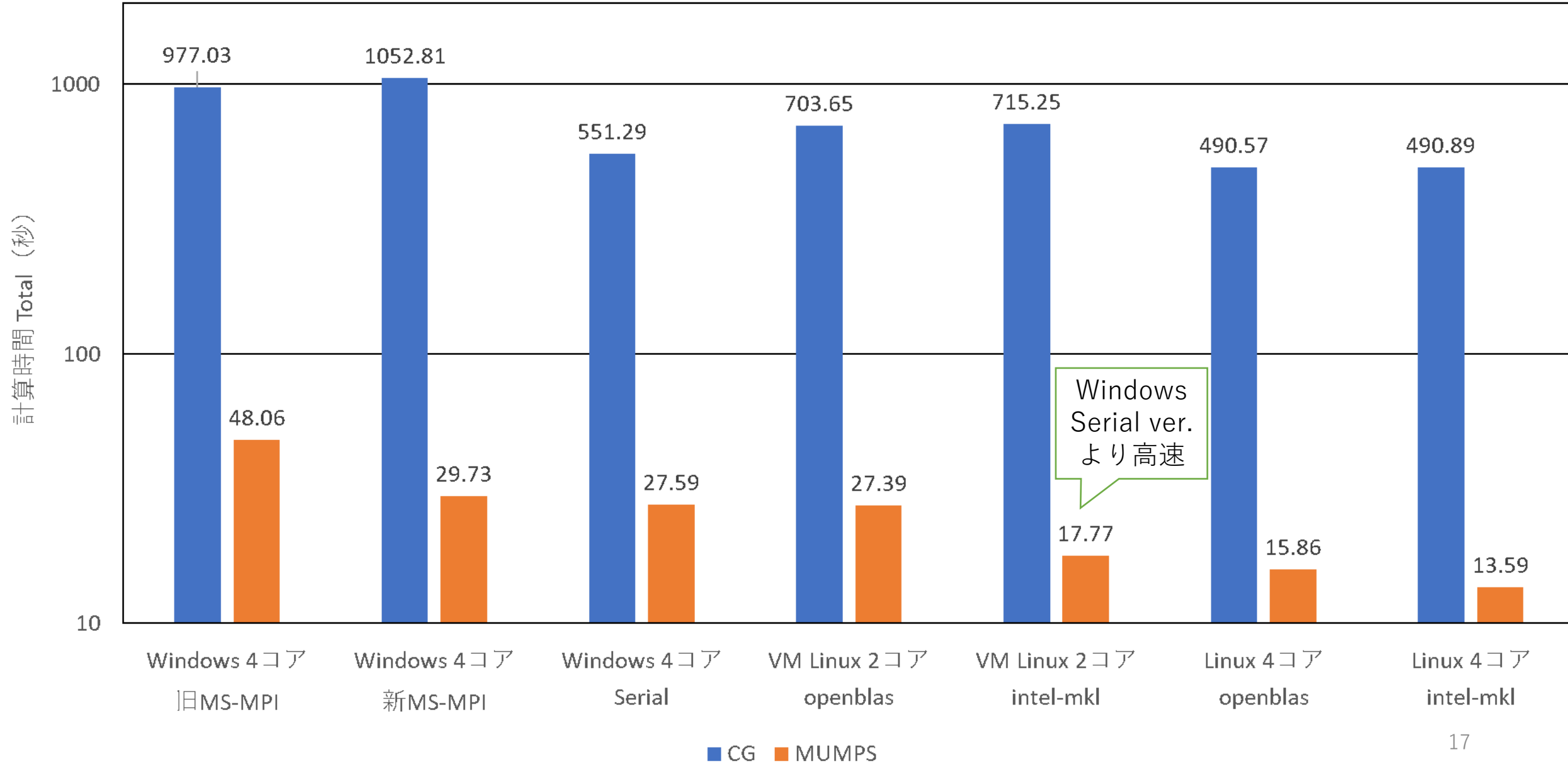
- CGと比べて、MUMPSが31～36倍高速
- Intel MKLを組み込んだMUMPSが最も高速
- CGは大きな差はない

Windows上のVMwareのゲスト(2コア)

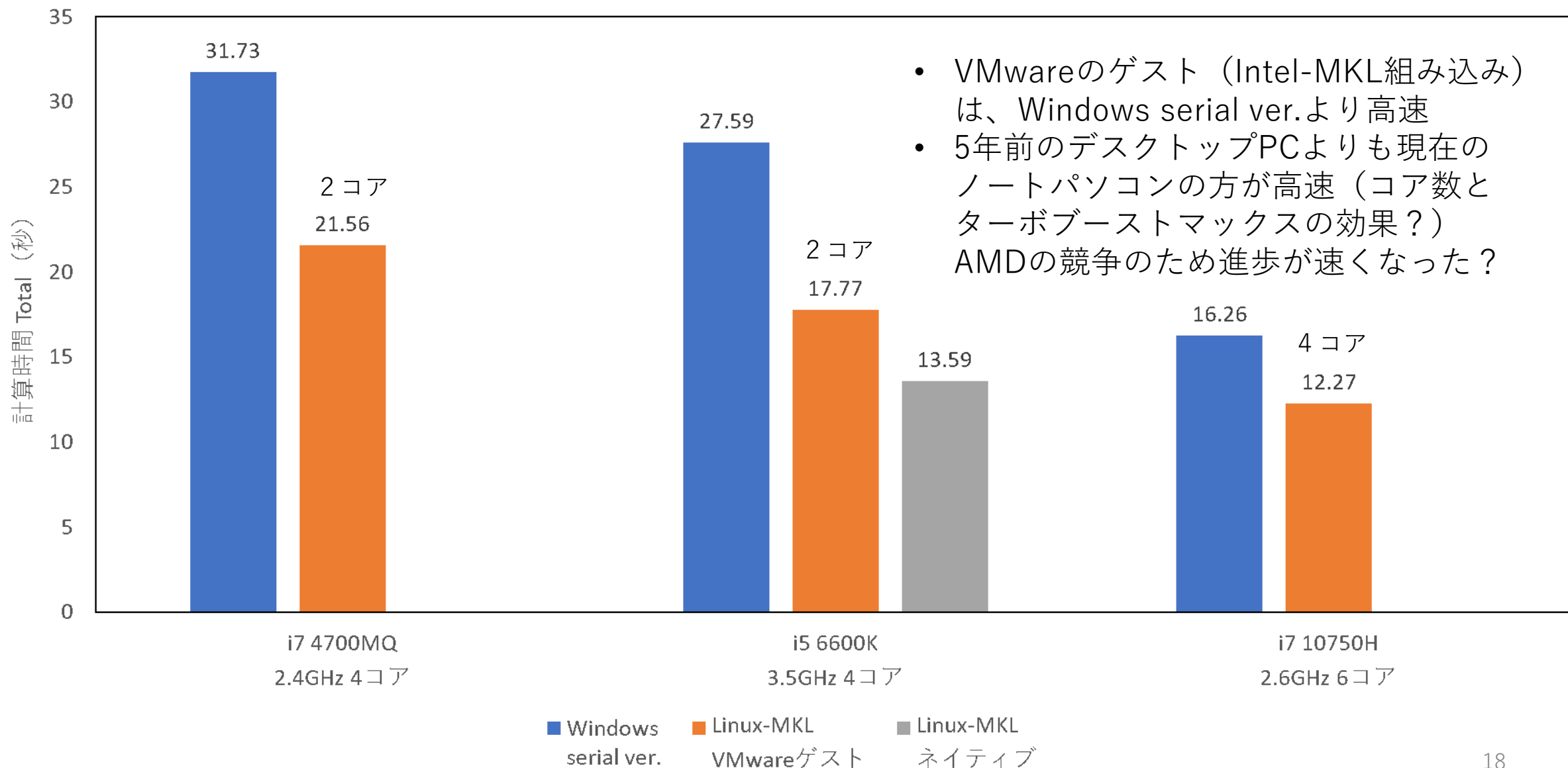
| | Frontistr 4.6 Openblas Ubuntu 18.04 | | Frontistr 5.11 Openblas Ubuntu 20.04 | | Frontistr 5.11 Intel-mkl Ubuntu 20.04 | |
|--------------|---|---------------|--|---------------|---|---------------|
| | CG | MUMPS | CG | MUMPS | CG | MUMPS |
| Total (秒) | 775.78 (30倍) | 25.99 (1倍) | 703.65 (26倍) | 27.39 (1倍) | 715.25 (40倍) | 17.77 (1倍) |
| Pre (秒) | 2.01 | 2.01 | 2.19 | 3.12 | 3.21 | 2.23 |
| Solve (秒) | 773.77 | 23.97 | 701.45 | 24.26 | 712.03 | 15.55 |

- CGと比べて、MUMPSは26倍～40倍高速
- Intel MKLを組み込んだMUMPSが最も高速
- CGは大きな差はない

FrontISTR-V5.1.1でのベンチマーク結果



他のマシンでのベンチマーク結果 (FrontISTR 5.1.1, MUMPS)



VMwareゲスト（Linux/MKL組み込み）の評価

メリット

- Windows版より高速（MUMPSソルバー）
- DirectMKLが使える（FrontISTR ver. 5.Xの場合。4.6はNG）
- イメージのコピーにより他のマシンへ環境を移植できる

デメリット

- ファイルを消してもディスクの使用量が減らない（SSDの場合、VMwareのディスク圧縮は危険？）
- これを避けるため、共有フォルダで作業するとディスクアクセスが極端に遅くなる（open-vm-tools-desktop）
- ゲストOS（Linux）からマルチディスプレイが使えない

結果のまとめ

- 並列計算を実施しない場合、Windows版はSerial ver.が良い
- MUMPSは、CGと比べて20～40倍高速
現在のPCであれば、RAMを64GB（ノート）、128GB（デスクトップ）まで増設できるので、MUMPSでかなりの範囲を解析できると思われる
- Intel MKLを組み込んだMUMPSソルバーが最も高速（Linux環境で確認した）
仮想マシンでもWindows Serial ver.よりも高速
- 仮想環境でのLinux（mkl組み込み）は、速度やDirectMKLを使用できるメリットがあるが、デメリットもある
マシンや解析内容に応じて、環境を選定することを推奨します

参考／最近のIntel製CPUの仕様

- 第10世代Core i7
 - 購入したノートパソコン（Core™ i7-10750H）の場合
 - 6コア12スレッド
 - 高負荷の場合でもクロックアップするインテル® **ターボ・ブースト・マックス・テクノロジー 3.0** に対応
- 第11世代Core i7
 - 最近発売されたCore™ i7-11370Hの場合
 - このCPUを搭載したVAIO Zは、RAM 32GB, 1TB SSDで約40万円と高価
 - 4コア8スレッド
 - ターボ（4コア）：4.6GHz
 - Intel® **AVX-512** に対応
- デスクトップパソコンなら第11世代が良い。ノートパソコンは時間があればコア数の多い第11世代のCPUが出荷されるまで待った方がよいと思います。

参考資料

1. オープンソース大規模並列FEM非線形構造解析プログラム FrontISTR v4.6のインストール (ubuntu 16.04 LTS)
<https://qiita.com/michioga/items/b9bc6d5c04318b107714>
2. EasyISTR固有値解析ベンチマーク
<http://opencae.gifu-nct.ac.jp/pukiwiki/index.php?%C2%E8%A3%B6%A3%B9%B2%F3%CA%D9%B6%AF%B2%F1%A1%A720190511>
3. FrontISTRの線形ソルバーと前処理
https://www.frontistr.com/seminar/160318/LINEQ_Solver_fixed.pdf