WindPerfectの高速処理

OpenMPソルバーの概要

2013年8月

株式会社環境シミュレーション

WindPerfectの高速処理 "OpenMPとは"

主に共有メモリ型並列計算機で用いられる標準化された基盤。
並列計算でのプログラミング手法。

◆OpenMP
・ディレクティブを挿入しメッセージ交換を実現。 MPIの方が速い。
・並列環境と非並列環境でほぼ同一のソースコードを使用可能。
・SMP環境向き。 並列化の効率はコンパイラに依存。

◆スレッドプログラミング
・1つの保護の単位としてのプロセス(スレッド)を複数動作。
◆MPI(MessagePassingInterfaceの略)
・マシン間でメッセージを授受できるライブラリインタフェース

どんな環境で並列化するか？

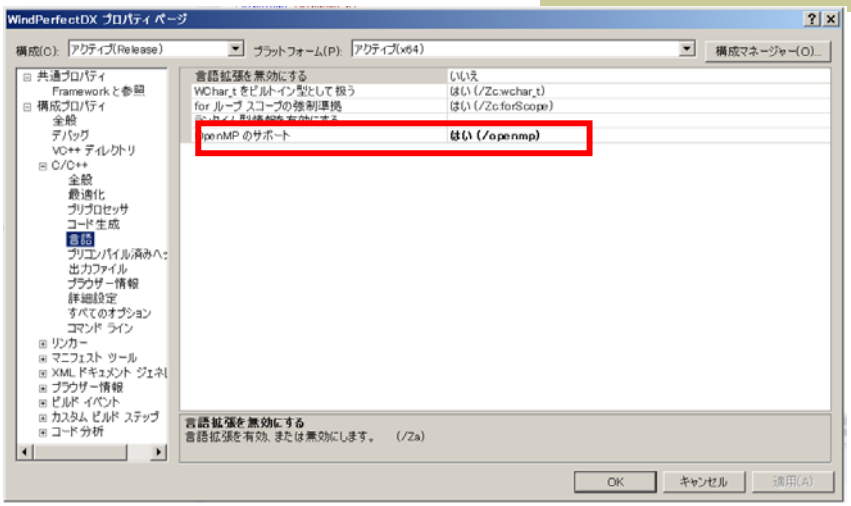
◆Linux ／ Windows
Linuxの方が並列計算の実績・資源が豊富　cf. GPGPU

◆WorkStation ／ PC
WorkStationのコア数：8～128，PCのコア数：1～8

◆Intel ParallelStudio / Microsoft VisualStudio2012
計算負荷の解析・データ依存評価が可能なParallelStudio
サポートをはじめて4年目のVisualStudio

VisualStudioの環境設定

◆プロジェクト属性設定



◆ヘッダー定義

◆OpenMP指示文

```
#include <omp.h>
#include <stdio.h>  ・・・

#pragma omp parallel
{
#pragma omp for private( I, J, K, IJK )
for(K=2;K<=KM1_Fine;K++)
for(J=2;J<=JM1_Fine;J++)
for(I=2;I<=IM1_Fine;I++)
{
IJK = I+IMAX_Fine*(J-1)+IJMAX_Fine*(K-1);
・・・・・
```

プログラミングの実際1

◆Navier-Stokes式の解法
　陽解法：Jacob法

◆圧力方程式の解法
　速度ー圧力同時解法：改良Gauss-Seidel法

◆温度/濃度/湿度移流拡散方程式の解法

データ依存が
極めて高い

◆輻射計算式(Stefan-Boltzmann式)の解法
　Radiosity計算なのでデータ依存性低い

プログラミングの実際2

◆Red-Black(Multi Color) 法

メモリ衝突を起こさないプログラミング技法

添字の和が奇数(odd)か偶数(even)かでループを分ける。

```
#pragma omp parallel //*
{
#pragma omp for private( N, IJK )
for(N=1;N<=NlocMatFluid_mod1;N++)
                {
                IJK = locMAT1_mod1_Fine[N];
                ・・・・・
                }
}
```

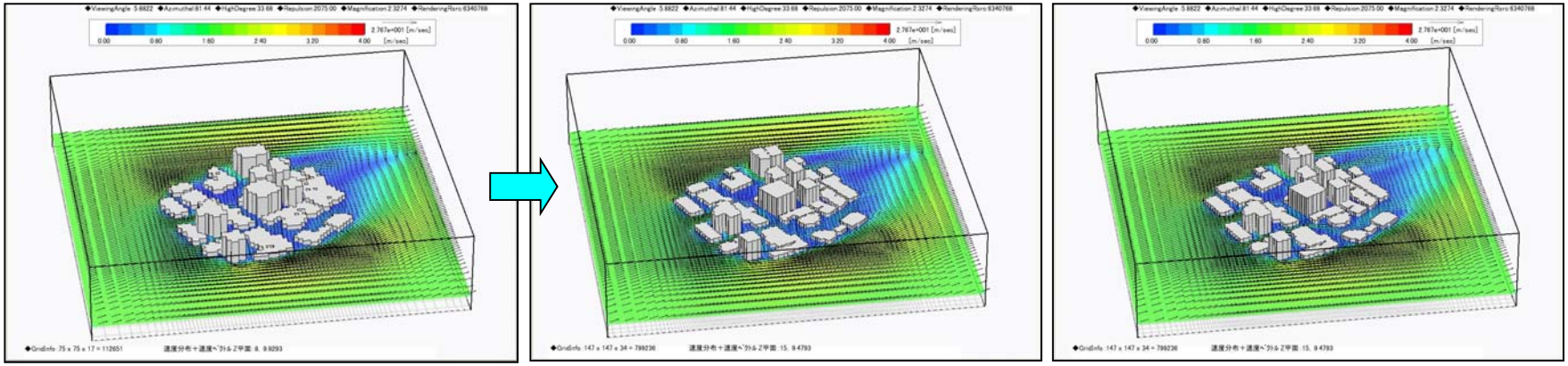
添字の和が奇数のループ

```
#pragma omp parallel //*
{
#pragma omp for private( N, IJK )
for(N=1;N<=NlocMatFluid_mod0;N++)
                {
                IJK = locMAT1_mod0_Fine[N];
                ・・・・・
                }
}
```

添字の和が偶数のループ

Nestingソルバーの概要

◆粗格子　→　細格子　で場の値を交換しながら計算を進行
　風環境解析でテスト計算



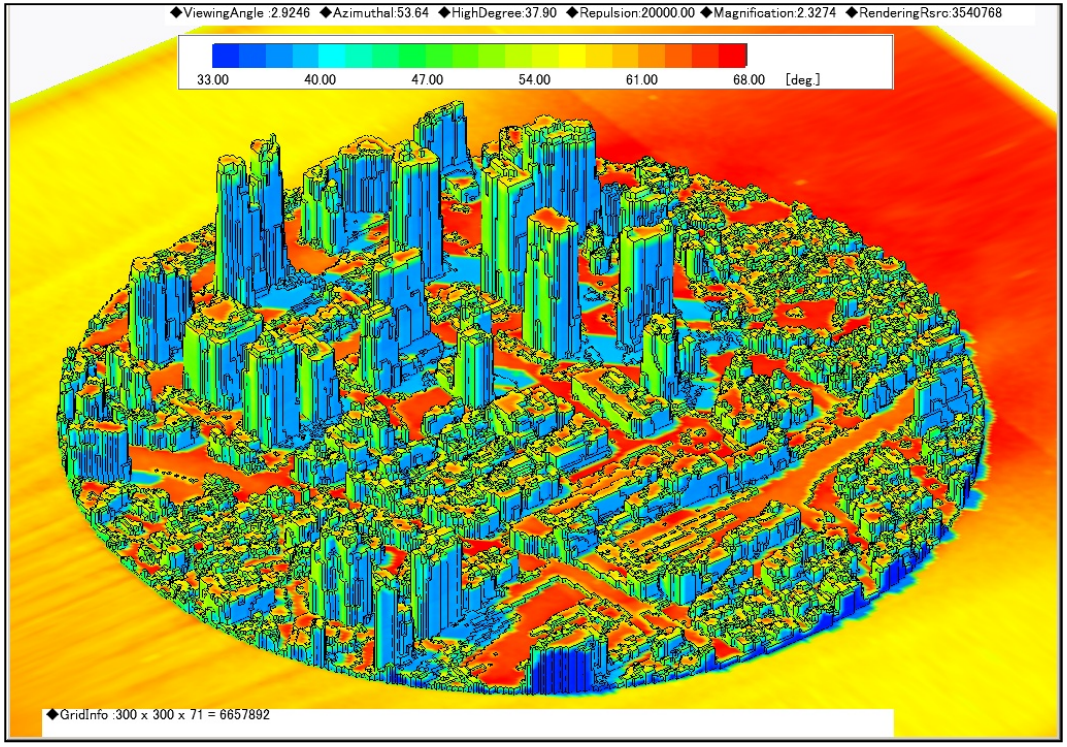
粗格子計算　　細格子計算　　非Nesting計算

格子数:　　147×147×34 ＝ 799236　on Corei7-3520M
計算時間:　　Singleソルバー　　:541.0sec
　　OpenMPソルバー　　:　84.9sec　→　6.37倍

OpenMPで約2倍、Nestingで3倍。但しOpenMP単独で6倍以上も。

ベンチマーク結果1

◆ヒートアイランド解析　西新宿



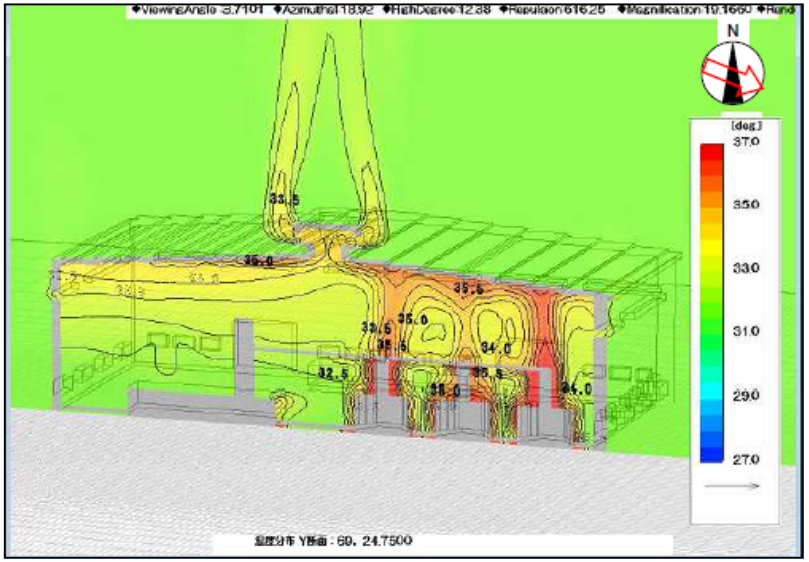
格子数:　300×300×71 ＝ 6657892　on Corei7-3520M

計算時間:　Singleソルバー　:1468.0min

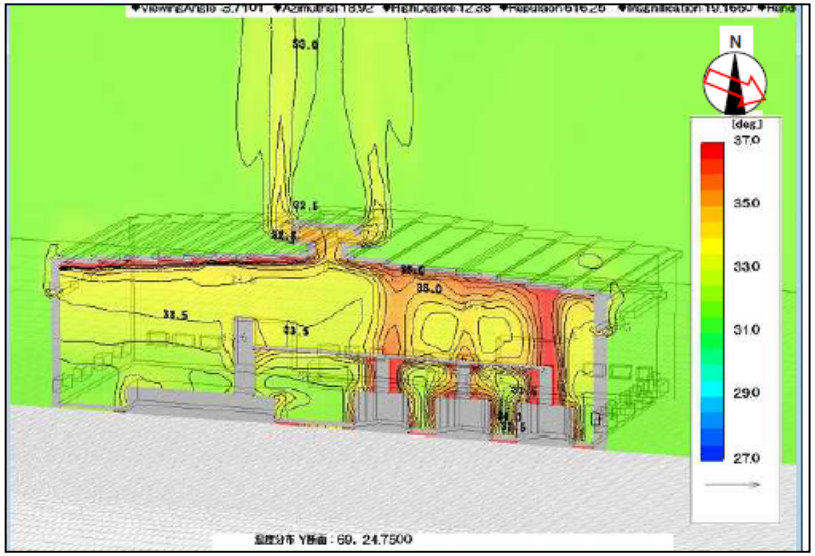
OpenMPソルバー　: 223.1min → 6.58倍

ベンチマーク結果2

◆自然換気解析



Singleソルバー結果



OpenMPソルバー結果

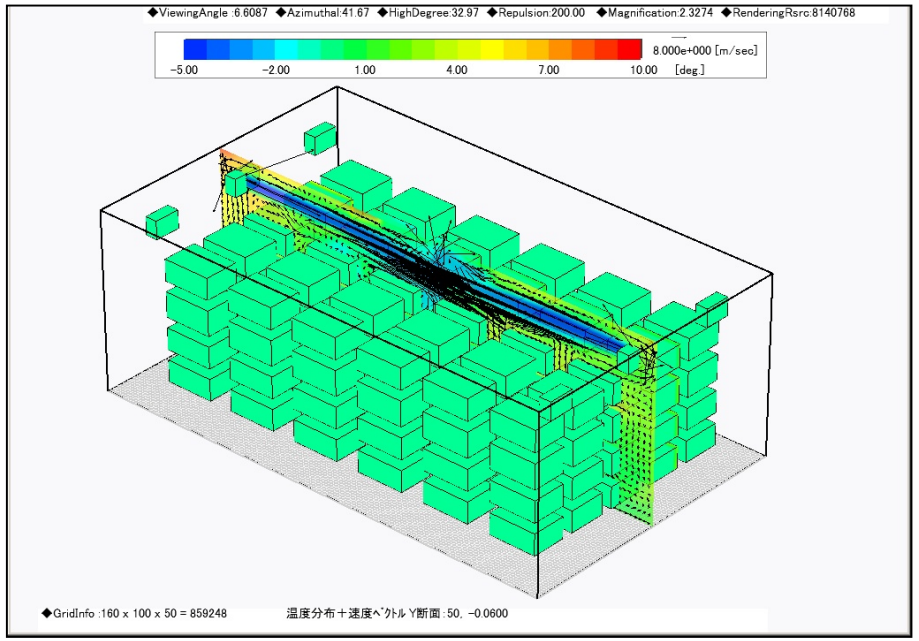
格子数: 119×118×76 ＝ 1132560 on Corei7-3520M

計算時間: Singleソルバー :196.0min

OpenMPソルバー : 37.0min → 6.58倍

ベンチマーク結果3

◆空調解析
冷蔵庫



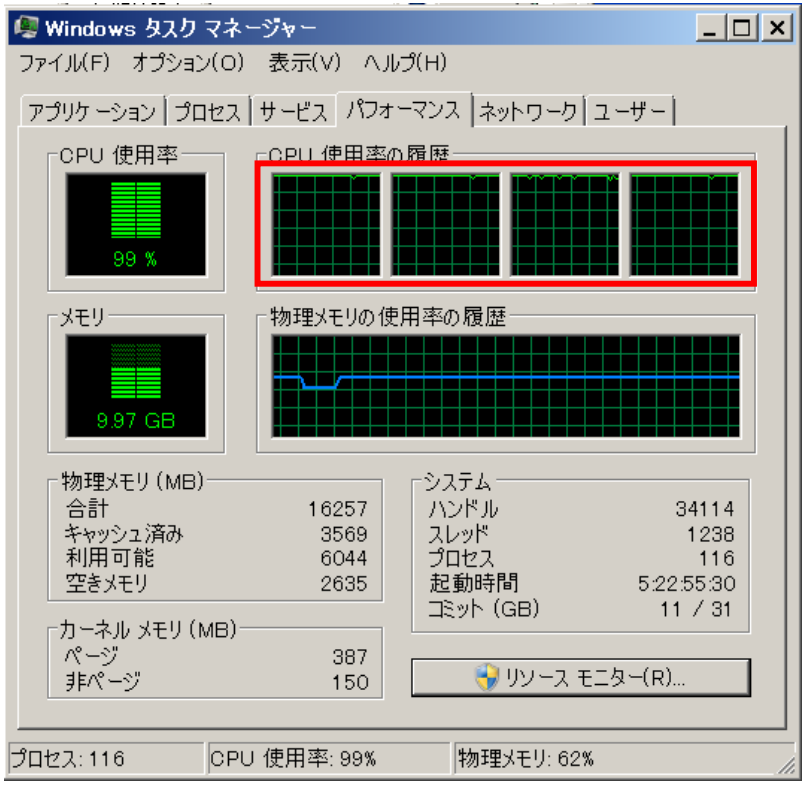
Singleソルバー結果　　OpenMPソルバー結果

格子数:　160×100×50 ＝ 859248　on Corei7-3520M

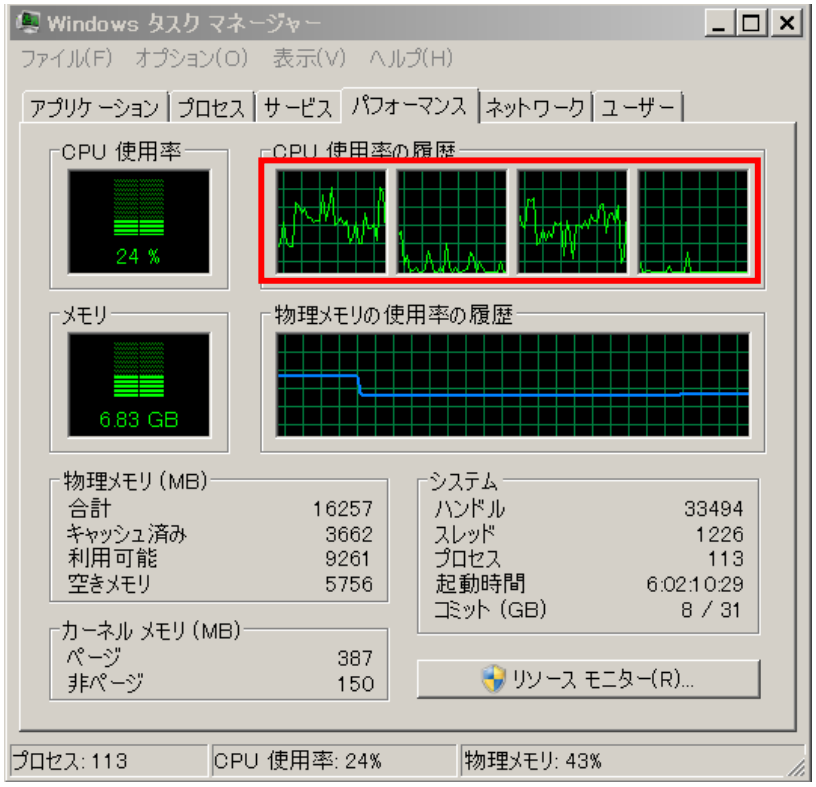
計算時間:　Singleソルバー　:632.0min

OpenMPソルバー　: 99.7min → 6.34倍

OpenMPソルバー利用時の注意点



OpenMPソルバー利用時



Singleソルバー利用時

OpenMPソルバーは計算機リソースを大きく費消。
同時にいくつもソルバーを走らせるのは不可。

今後の課題

◆8コア以上のマルチコアへの対応

◆並列計算時のパフォーマンス解析

◆データ依存の解消

◆課金： 8コアまでは加算はありません

ソルバーの高速化　1

○ターンアラウンドの速さが重視される時代

より良い設計条件の探索には速いことが必須

シームレスな思考過程がより付加価値を生む

○解析精度とのトレードオフ

非線形性の強い問題、熱対流の卓越した問題は、計算速度の速さだけでは評価出来ない。

ソルバーの高速化　２

○ネスティングソルバー　（Nesting, Multi Grid）

・全体ネスティング

　風環境解析等では一定の高速化が図れる

・部分ネステイング

　空調・機械系の問題で、吹出し口付近近傍の（Courant数が小さい）ネステイングを行う

・計算負荷が解析規模に比例

ソルバーの高速化　３

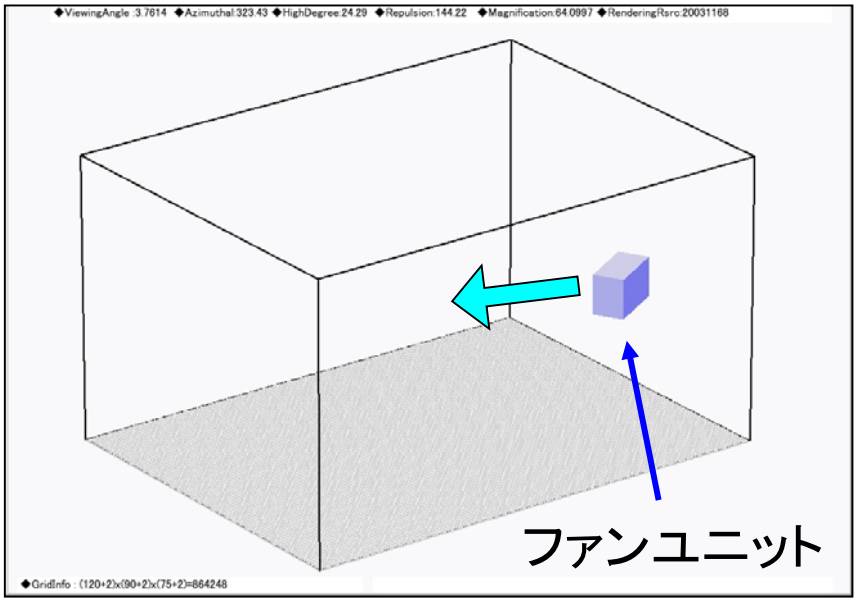
○解法パラメータの最適化

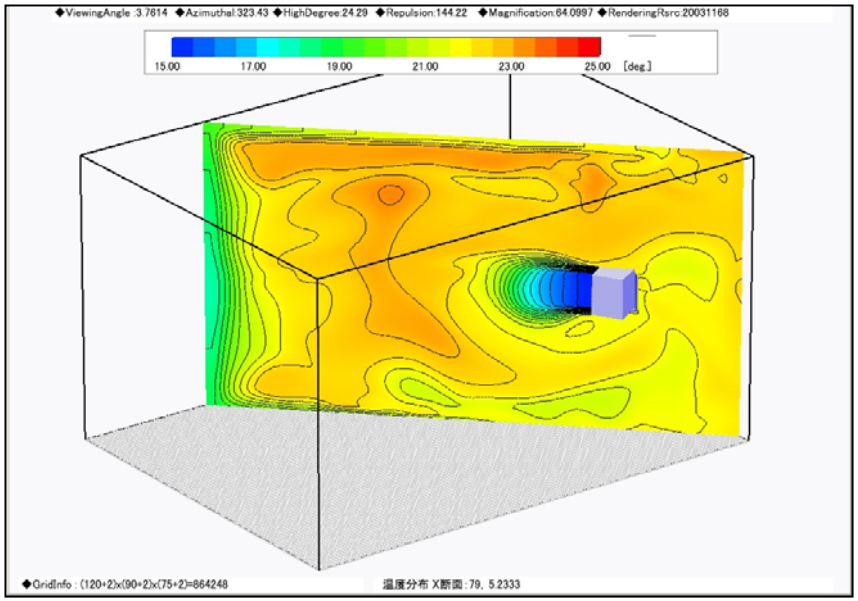
同じ建築分野であっても、解析パラメータの設定は問題によって異なる。

- 風環境・風荷重解析　：　渦・シアーの再現を重視
- 外部熱気流解析　：
  風量・熱量バランスの維持を重視
- 空調解析　：　流れ場の進展を重視
- 換気解析　：　内外気流の連続性維持を重視

ソルバーの高速化　4

解析例：ファンユニットによる室内空調



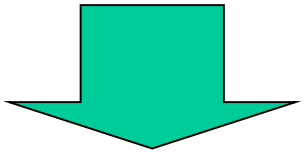


最適化無し：1．00倍　　最適化有り：4．23倍

最適化有り＋TVD：5．10倍

バイナリRead/Write　1
バイナリ形式の結果ファイル

結果ファイル（.rst）の形式を
アスキー（テキスト）方式から
バイナリ（内部表現）形式に変更



- 結果ファイル入出力の高速化
  （読み込み速度は従来の約7倍）
- 結果ファイルサイズの軽量化
  （従来の2分の1以下）

バイナリRead/Write　2
結果ファイル出力の高速化

結果ファイル出力速度の比較例

•総格子数360万の場合

	出力時間
アスキー形式	23秒
バイナリ形式	9秒

約2.6倍

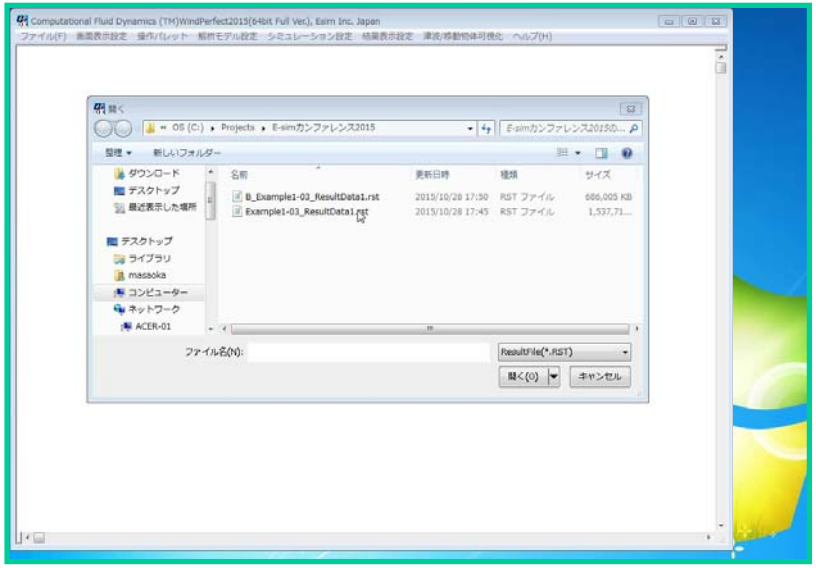
•総格子数1440万の場合

	出力時間
アスキー形式	97秒
バイナリ形式	42秒

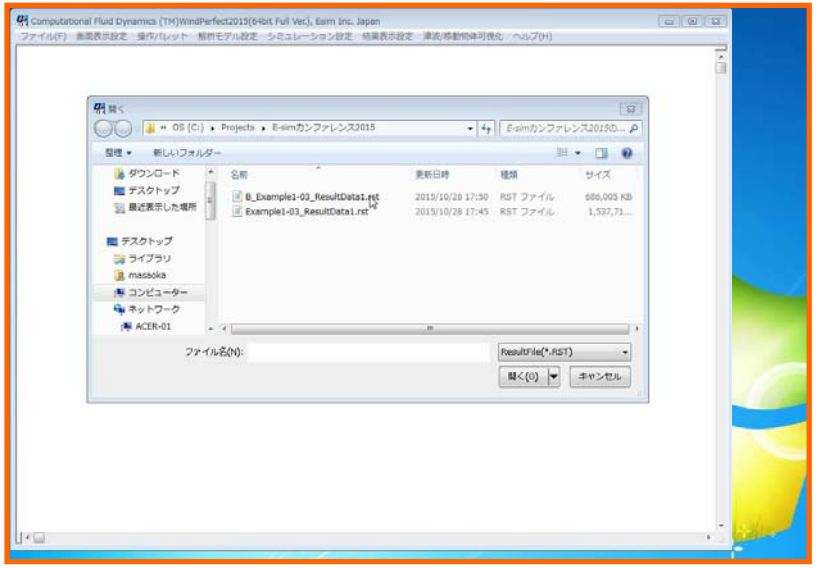
約2.3倍

バイナリRead/Write　3
結果ファイル読み込みの高速化

結果ファイル読み込み速度の比較



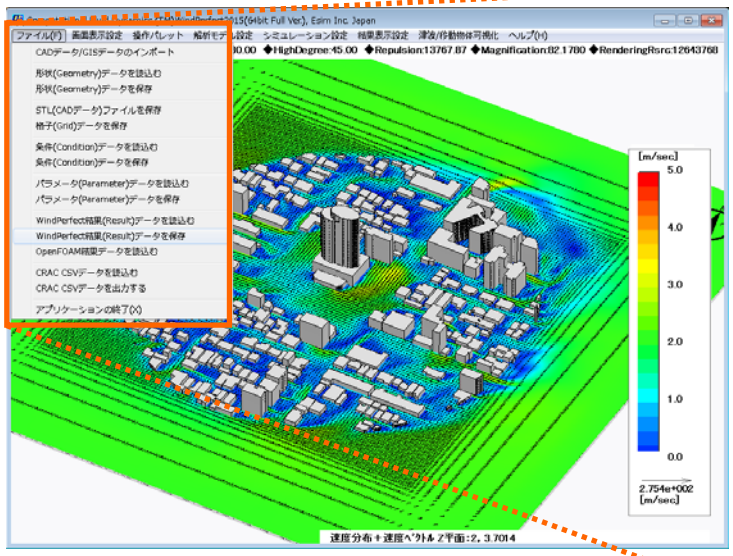
アスキー形式
1分48秒



バイナリ形式
15秒

バイナリRead/Write　4
バイナリ形式出力

従来の結果ファイル(.rst)をバイナリ形式に出力することが可能(ファイルサイズ軽量化)



従来の結果ファイルを読み込み、
”WindPerfect結果データを保存”で出力