# オープンソース可視化ソフトウエアのための高速並列リーダコードの開発

## OpenFOAM native reader for ParaView 3

大嶋拓也(新潟大学)

# 自己紹介

NIIGATA UNIVERSITY

## 所属

新潟大学工学部建設学科　建築学コース

## 所属学会

日本建築学会、日本音響学会、日本流体力学会、日本騒音制御工学会

## 専門

建築音響学 (音響数値解析)

# 開発動機(1)

NIIGATA UNIVERSITY

・WindowsベースのCAE演習用教材の製作

- OpenFOAM (Cygwinベース) + ParaView (Windowsネイティブ)

・OpenFOAM付属のParaView (当時はParaView2)用リーダ

- OpenFOAMライブラリに依存
- CygwinライブラリとWindowsネイティブ(MSVC++)ライブラリは非互換

# 開発動機(2)

NIIGATA UNIVERSITY

・Windows上でParaViewからOpenFOAMデータを読み込むための選択肢

1. foamToVTK

    ◯: プログラミング不要

    ×: データ変換の手間と容量、解析上のタイムステップ情報が失われる

2. ParaViewをCygwin上でビルド、OpenFOAMライブラリとリンク

    ×: 非現実的(無理でした)、性能が貧弱(特にI/O)

3. OpenFOAMライブラリに依存しない(ネイティブ)リーダコードを作成

    ◯: ParaView単体で動作、全てを開発者の好みで作成できる

    ×: 開発労力、実現可能性

➡ 演習用教材以上の発展も見越して、3.のネイティブリーダ開発に決定

# Introduction

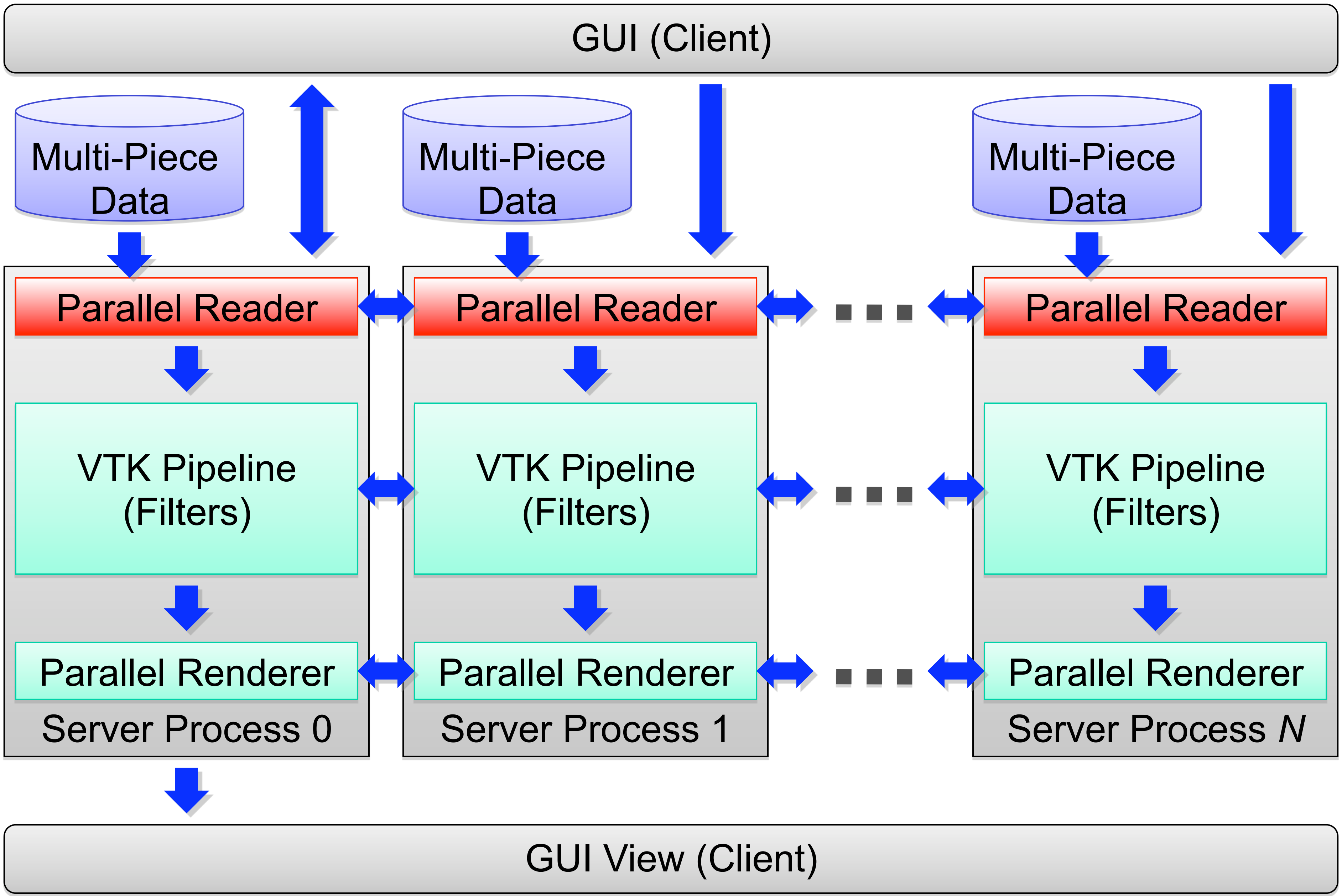


- ParaView does efficient rendering even for large (> 10 million cells) cases
- It's I/O performance that defines user experience in real situations
- We want a fast reader for OpenFOAM!

- NB: The following explanations about the reader implementation is not meant to be exhaustive.  Please consult the actual source code for details.

# ParaView server-client model

NIIGATA UNIVERSITY

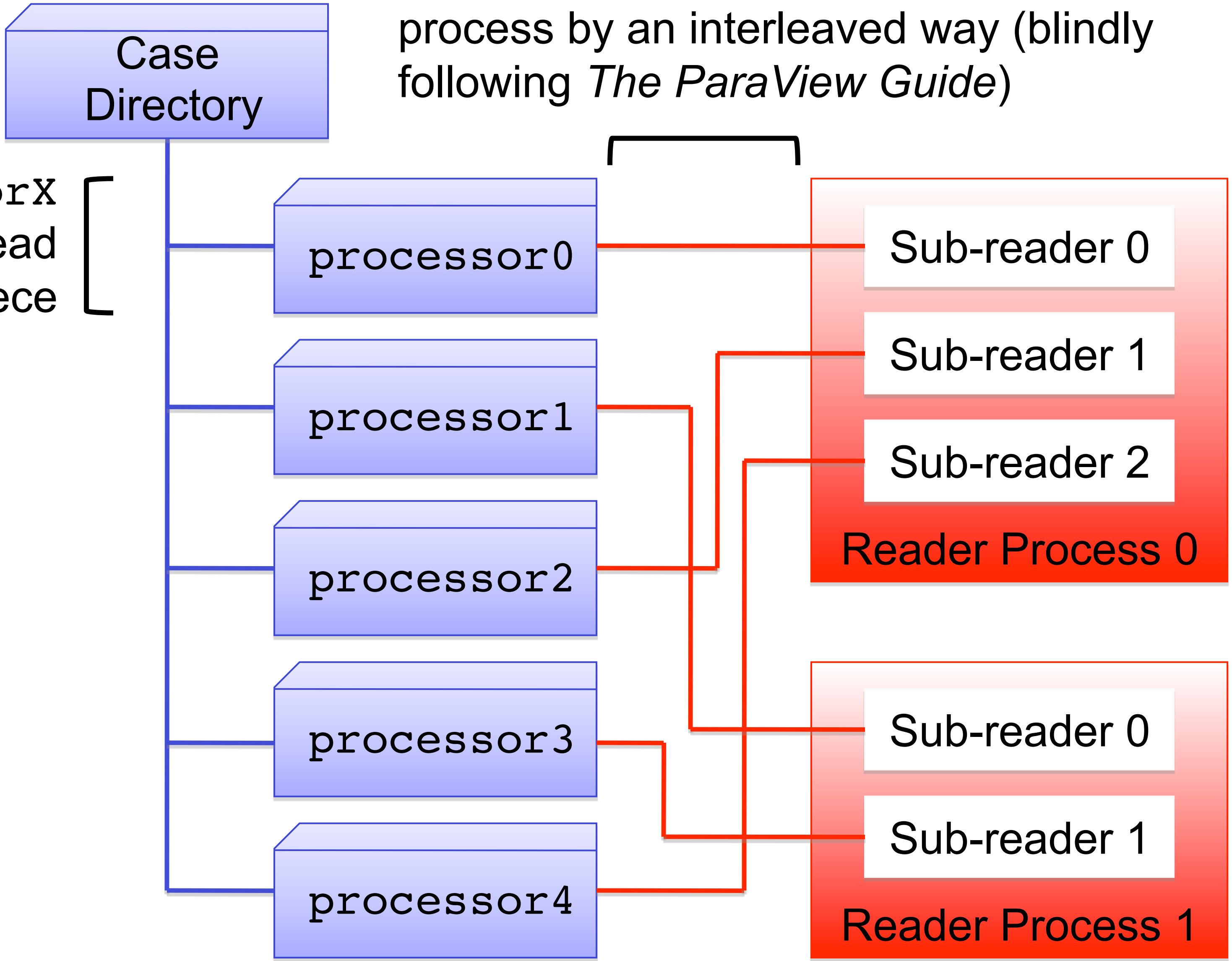


# Piece treatment in the reader

NIIGATA UNIVERSITY

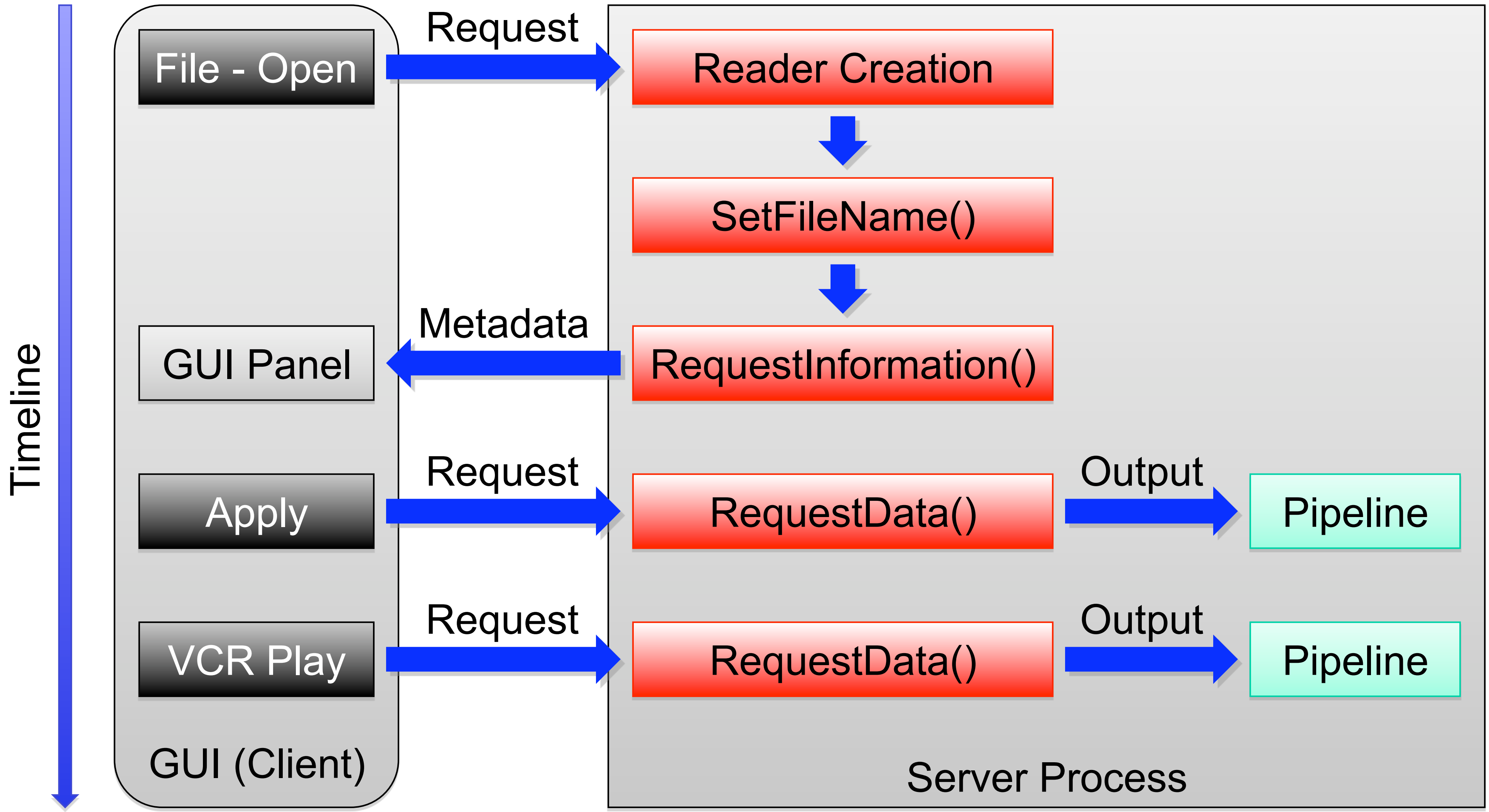
Each piece is assigned to a reader process by an interleaved way (blindly following *The ParaView Guide*)

Each `processorX` subdirectory is read as a piece



# ParaView reader request sequence (outline)

NIIGATA UNIVERSITY



# Processing information request



**RequestInformation()**

Input information

- Number of server processes
- My process number

► Determine which `processorX` subdirectories to read

Output metadata

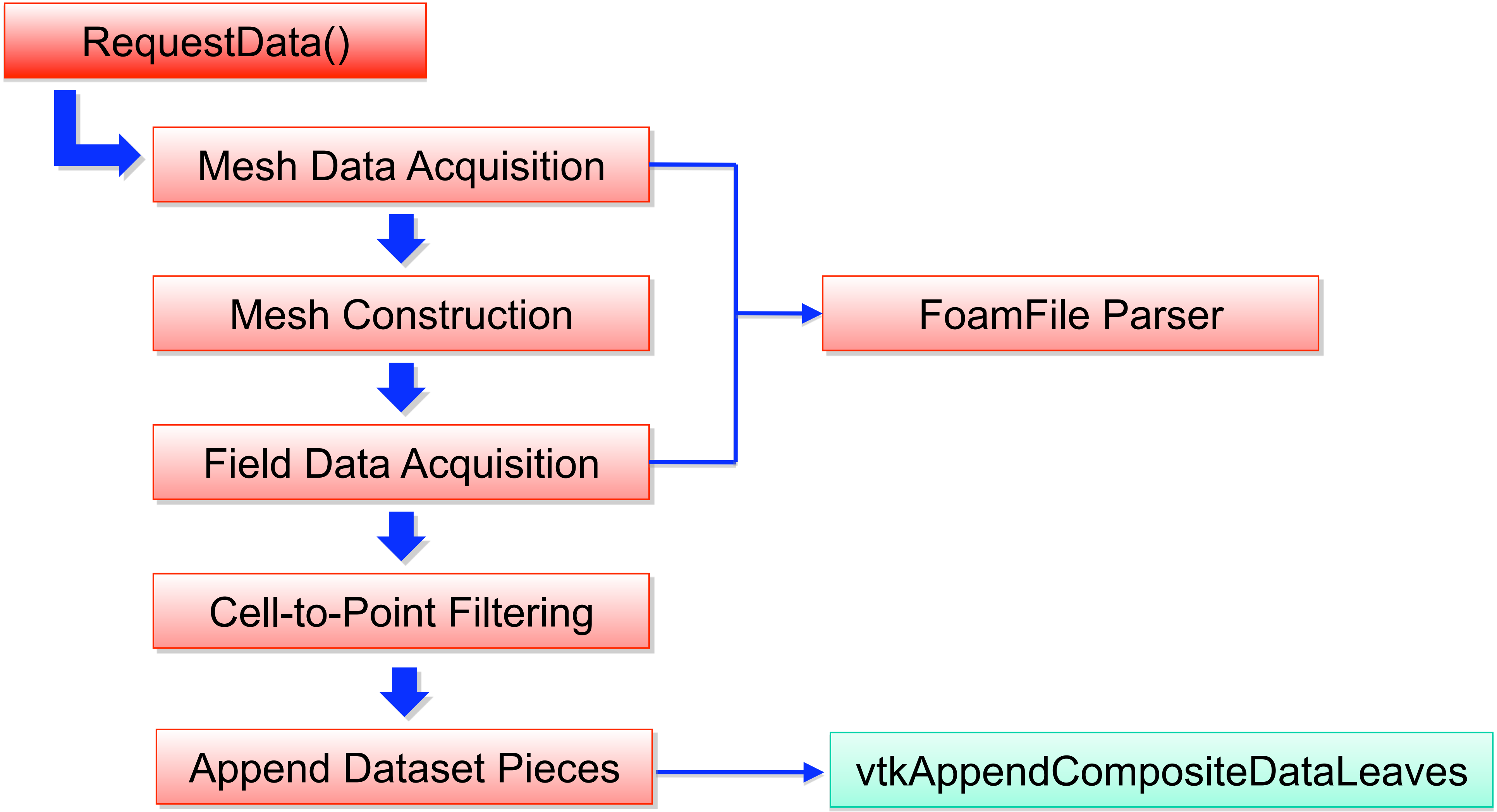
- Number of data pieces ◄ Count the number of `processorX` subdirectories
- Number of timesteps
- List of timesteps

] List time directories

- List of boundary patches ◄ Obtain from `polyMesh/boundary` file
- List of cell/point/lagrangian arrays ◄ List field objects under a time directory
- Collect metadata to server process 0 (the only interprocess communication)

# Processing data request: Overview

NIIGATA UNIVERSITY




- Implementation details follow

# Processing Data Request: FoamFile Parser



## FoamFile Parser

- Dedicated parser that handles C-like syntax of OpenFOAM file format
  - Covers many undocumented exceptional syntaxes
- Directly interacts with zlib for gzip-compressed format support
- Hijacks crc32() by an empty dummy function when possible (+5% performance)
- Uses own string-to-float conversion routine as a replacement to system strtod()
  - The key part that defines the reader performance for ascii cases
  - Fast!
  - Omits overflow/underflow handling
  - *Not* meant to be accurate until the last bit of mantissa
  - ... but proven to be reasonably accurate for postprocessing purposes

# Processing Data Request: Mesh construction (1)



**Mesh Construction**

- Convert OpenFOAM face-oriented polyMesh data structure to VTK cell-oriented unstructured grid
- The key part that determines initial case loading time

1. OpenFOAM polyMesh format

| Face number | Owner Cells | Neighbour Cells | Face-Points | | | |
|---|---|---|---|---|---|---|
| 0 | Cell #0 | Cell #0 | Point #0 | Point #1 | Point #2 | |
| 1 | Cell #1 | Cell #1 | Point #0 | Point #1 | Point #2 | Point #3 |
| 2 | Cell #2 | Cell #2 | Point #0 | Point #1 | Point #2 | |

# Processing Data Request: Mesh construction (2)

NIIGATA UNIVERSITY

2. Create intermediate cell-face list from owners and neighbours

Cell number

Cell-Faces

| Cell number | | | | |
|---|---|---|---|---|
| 0 | Face #0 | Face #1 | Face #2 | |
| 1 | Face #0 | Face #1 | Face #2 | Face #3 |
| 2 | Face #0 | Face #1 | Face #2 | |

3. Create ordered cell-point list (VTK unstructured grid)

Cell number

Cell-Points

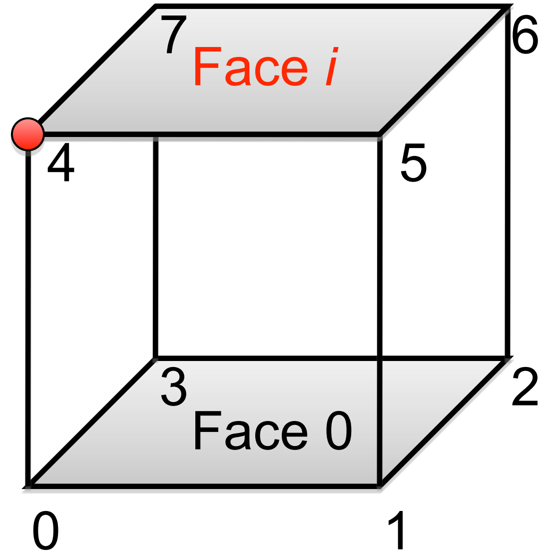
| Cell number | | | | | |
|---|---|---|---|---|---|
| 0 | Point #0 | Point #1 | Point #2 | Point #3 | |
| 1 | Point #0 | Point #1 | Point #2 | Point #3 | Point #4 |
| 2 | Point #0 | Point #1 | Point #2 | Point #3 | |

# Processing Data Request: Mesh construction (3)



Creation of cell-point list from cell-face/face-point list (a rough sketch)

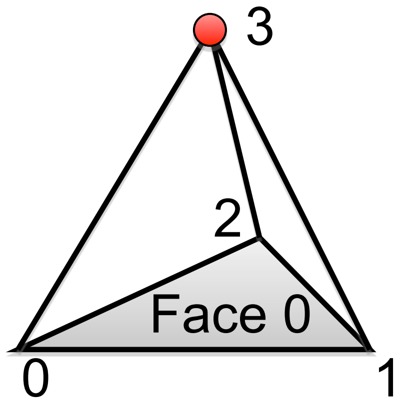
Hexahedron and prism:




a. Search for face *i* that does not share any of its vertices with Face 0

b. Search for a pivot point which is the opposite point of the edge that starts from point 0 of face 0 and that does not belong to face 0

Tetrahedron and pyramid:




a. Search for a point that does not belong to face 0

# Processing Data Request: Cell-to-point filtering



## Cell-to-Point Filtering

- Does roughly what volPointInterpolation in OpenFOAM does or what vtkCellDataToPointData in VTK does
- The filter stands at the middle of the two from accuracy point of view:
  - Does *not* do inverse distance weighting (following vtkCellDataToPointData)
    - Saves extra memory required to hold weighting factors
    - Saves extra computational load to do IDW
  - Does account for boundary values (following volPointInterpolation)
    - Overrides filtered values at boundary points by boundary values
    - Also accounts for all neighboring boundary values at patch-edge points

# Timing tests (1): Setup

NIIGATA UNIVERSITY

Testing environment

- Mac Pro 3.0 GHz 4-core, 16GB RAM 1.0TBx3 RAID0, OS X 10.5.5
- OpenFOAM 1.5.x OS X Port 2008-10-08
- ParaView 3.5-CVS 2008-11-11

► Not an officially supported platform of OpenFOAM, take as a rough indication. Also note the benchmarks are meant to show difference in design philosophies, not to judge absolute technical superiorities.

Timing instrument

- "Tools" -> "Timer Log"

Enabled fields

- p, U

# Timing tests (2): Simple serial case

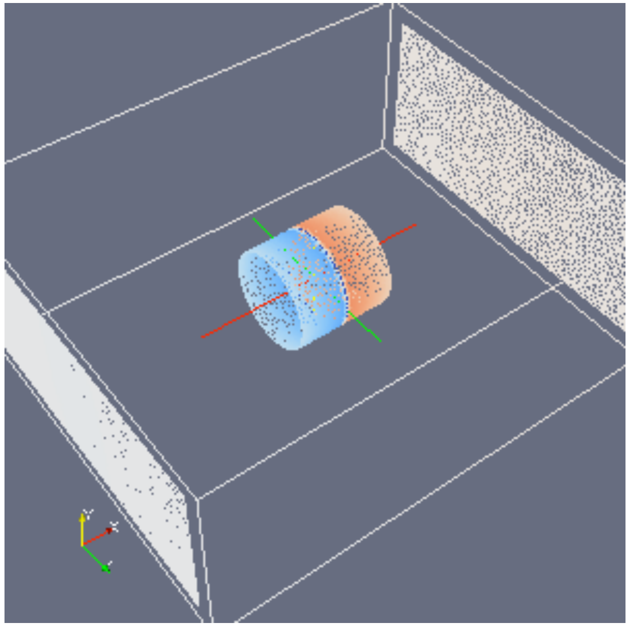
NIIGATA UNIVERSITY

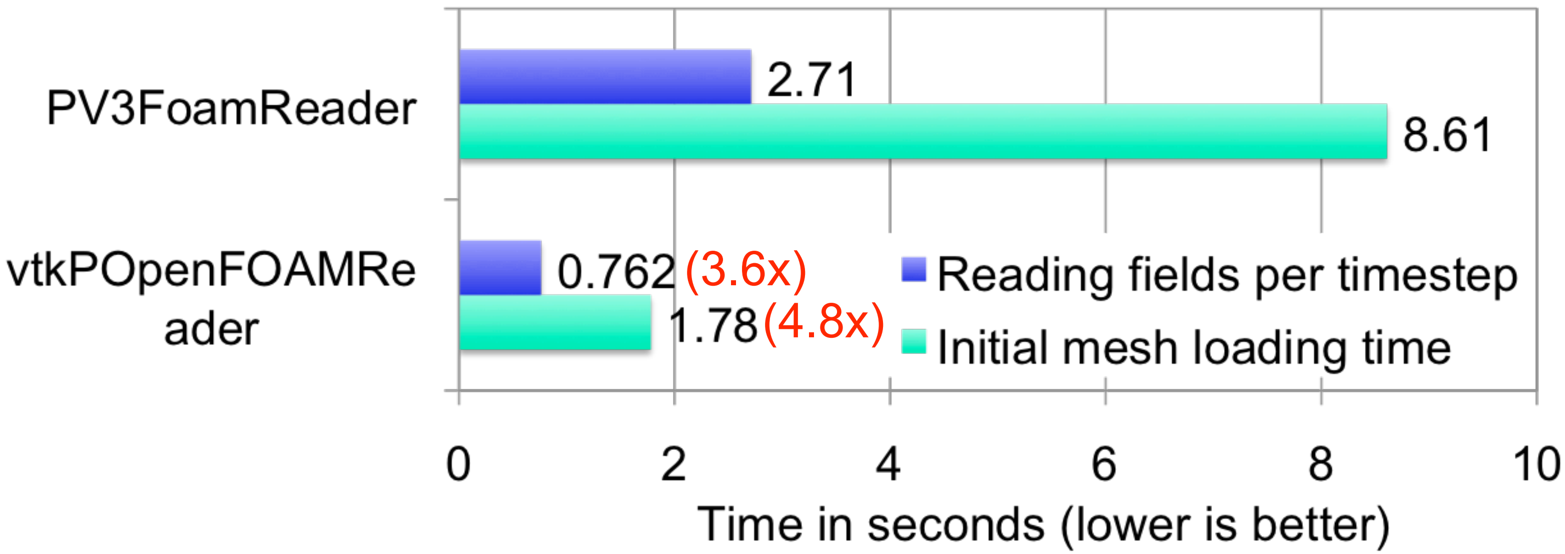
Parallelepiped geometry meshed with tetrahedra

File format: Gzipped-Ascii

Case type: Serial case

Number of cells: 773,543 cells





# Timing tests (3): Parallel case

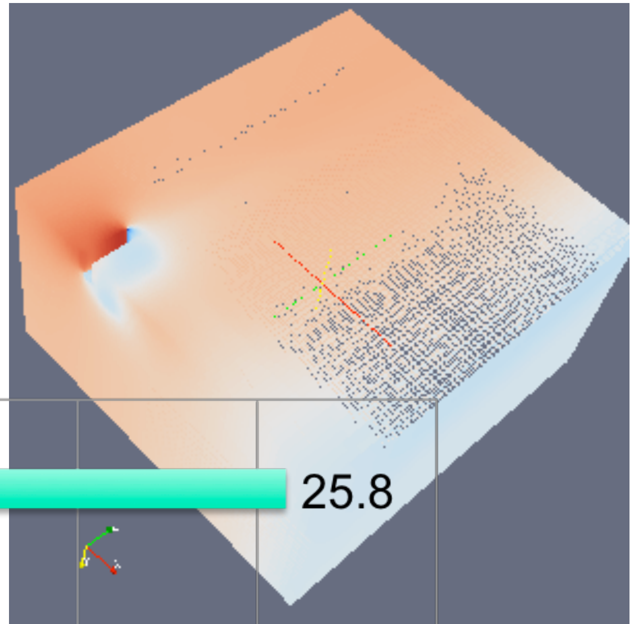


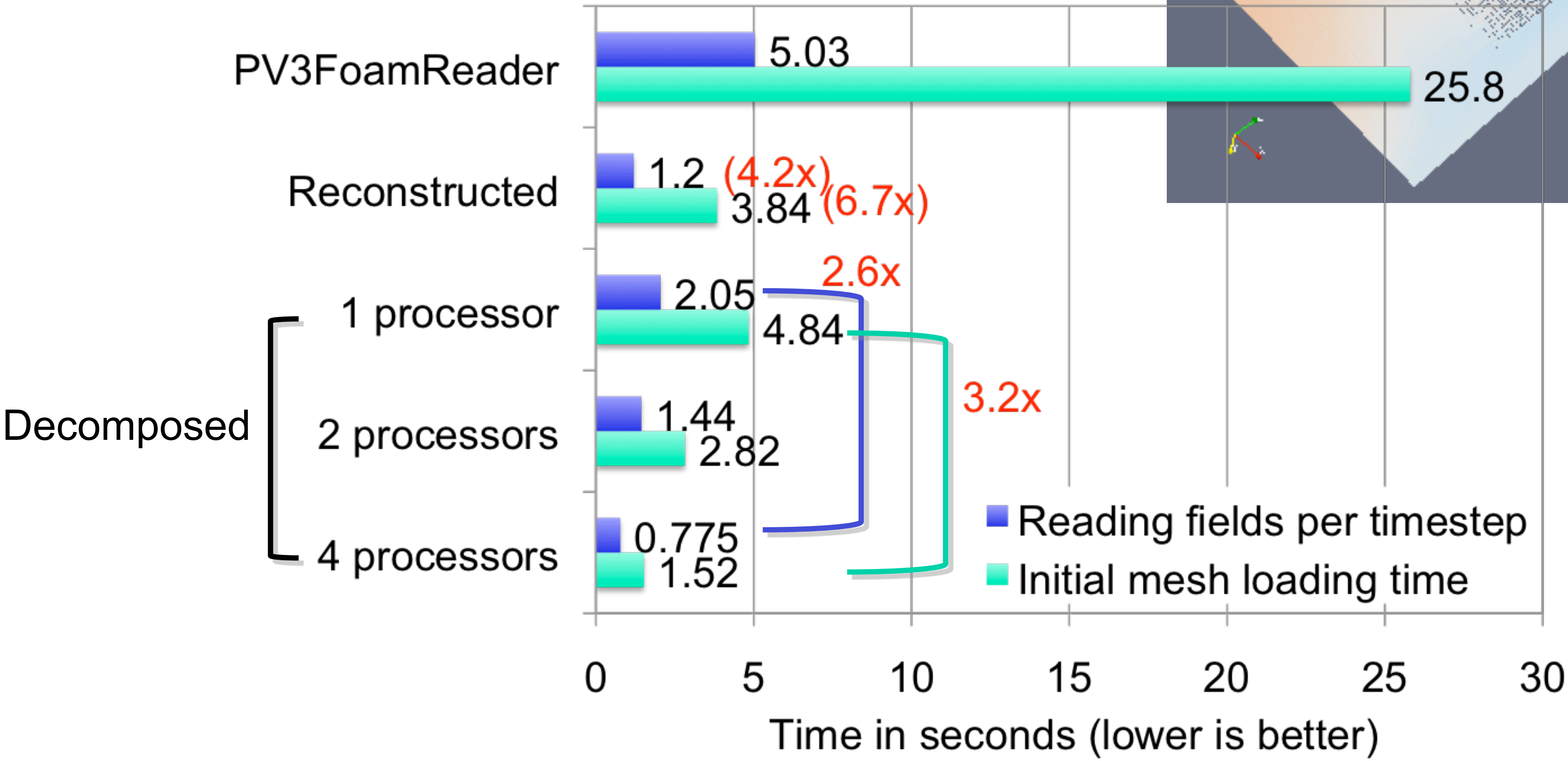
Parallelepiped geometry meshed with hexahedra

File format: Gzipped-Ascii

Case type: Serial / parallel-decomposed cases

Number of cells: 1,291,208 cells





# Timing tests (4): Parallel large case

NIIGATA UNIVERSITY

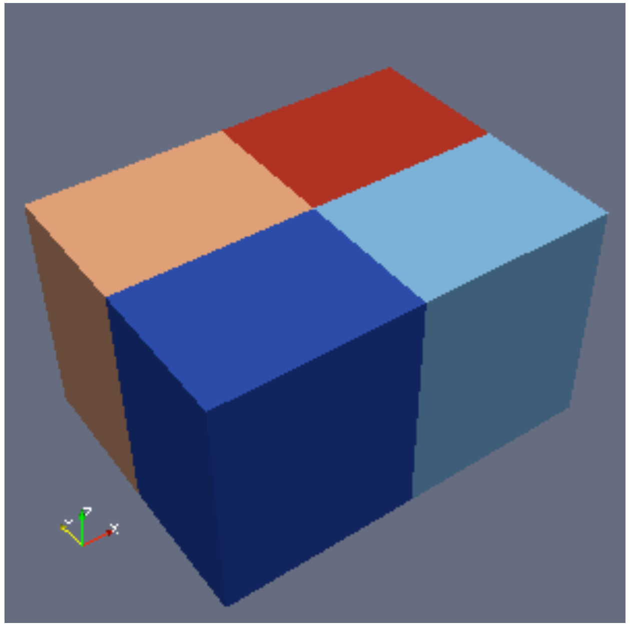
Parallelepiped geometry meshed with hexahedra

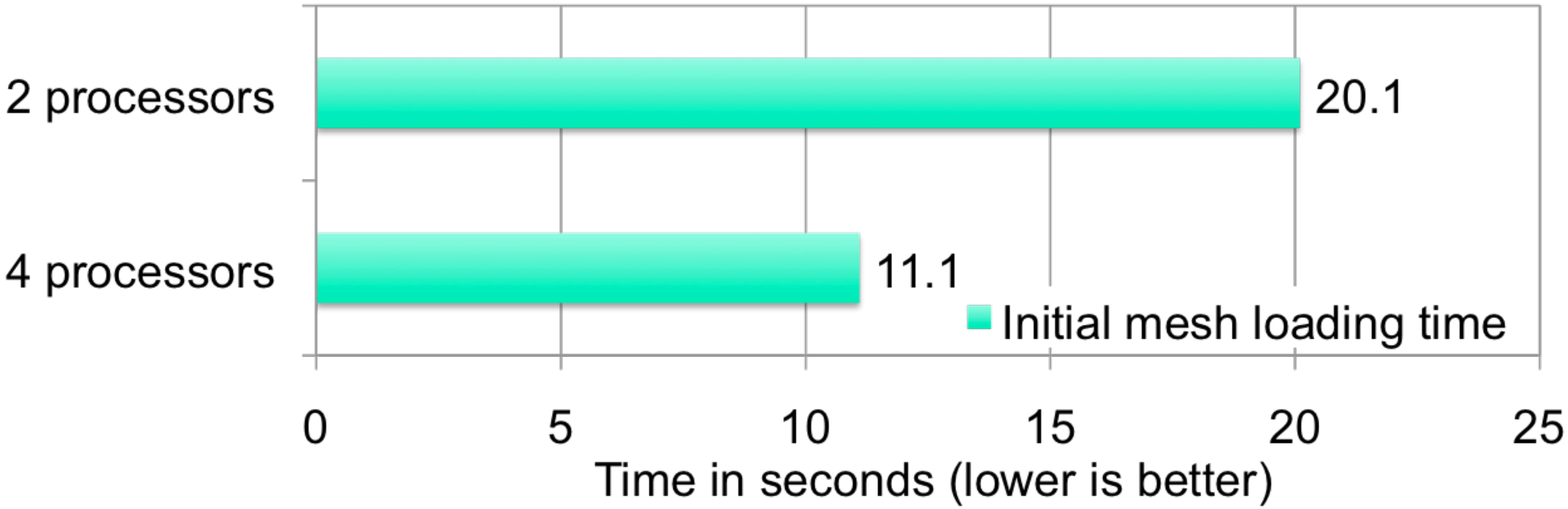
File format: Gzipped-Ascii

Case type: Parallel-decomposed case

Number of cells: 12,150,000 cells

(about 10x of the previous case)





# Summary and future works



## Summary

- Implemented an OpenFOAM parallel reader for ParaView
- Found to be 3x – 7x (typically 4x – 5x) faster than PV3FoamReader in serial
- Parallel tests showed 2.6x – 3.2x speedup for 4 processors

## Future works

- Make the reader a part of official ParaView/VTK distribution (involves politics)
- Geometry filter optimization (rather a matter of ParaView itself than the reader)

# Thanks for listening!

The reader code is available at

http://openfoamwiki.net/index.php/Contrib_Parallelized_Native_OpenFOAM_Reader_for_ParaView