Pluggable I/O

Mattijs Janssens

- Why?
  - Inflexible
  - Parallel I/O
  - Lots of directories
  - NFS trashing
- Solution
  - Use run-time selection system
  - At regIOobject level
  - Change format for parallel I/O

regIOobject

Base class for object that

- is registered on a database
- can read/write itself to a stream (serialisation)
- (optional) can read/write itself to disk

Examples:

- p, U, polyMesh/owner

regIOobject (2)

```
template<class T>
class IOList
:
    public regIOobject,
    public List<T>
{

    //- Read from disk
    IOList(const IOobject&);
    ..
    //- Write contents
    virtual bool writeData(Ostream&) const;
};
```

IOobject

- basic description of name and location
  instance : time directory
  local : subdirectory, e.g. polyMesh/
  registry : database
  readOpt : MUST_READ

```
Info<< "Reading field p\n" << endl;
volScalarField p
(
    IOobject
    (
        "p",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
```

Istream/Ostream

- input and output stream
- OFstream : stream data to a file
- IPstream : stream data from another processor
- OStringStream: stream data into memory buffer

Examples:

```
OFstream os("myFile.txt");
os << p << endl;
```

Or

```
p.writeData(os);
```

Read/Write (to disk)

Read from disk:

- search file belonging to IOobject
- open an IFstream
- read header, switch format (binary, compressed)
- call readData(Istream&)

Write to disk:

- obtain file name for IOobject
- open an OFstream
- write header
- switch format (binary, compressed)
- call writeData(Ostream&)

Plug-in

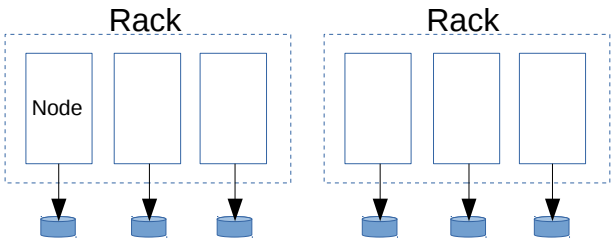
- relay above functionality to plug-in: fileHandler
- plug-in provides:
  - fileName filePath(const IOobject&);
  - bool read(regIObject&);
  - fileName objectPath(const IOobject&);
  - bool write(const regIOobject&)
- etc/controlDict or system/controlDict
- decomposePar -fileHandler collated

Plug-in (2)

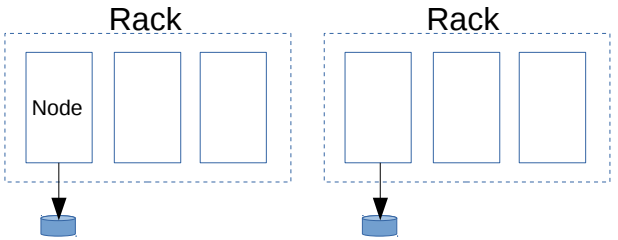
Currently:

- uncollated : existing format
- masterUncollated : all file operations on master processor only
- collated : same but changes output format

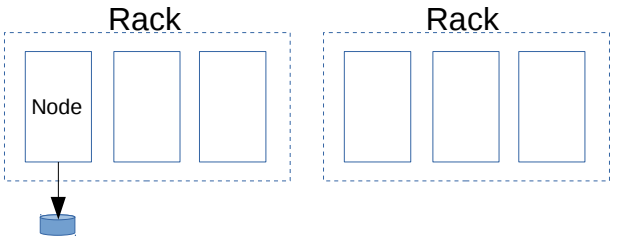
Storage



local



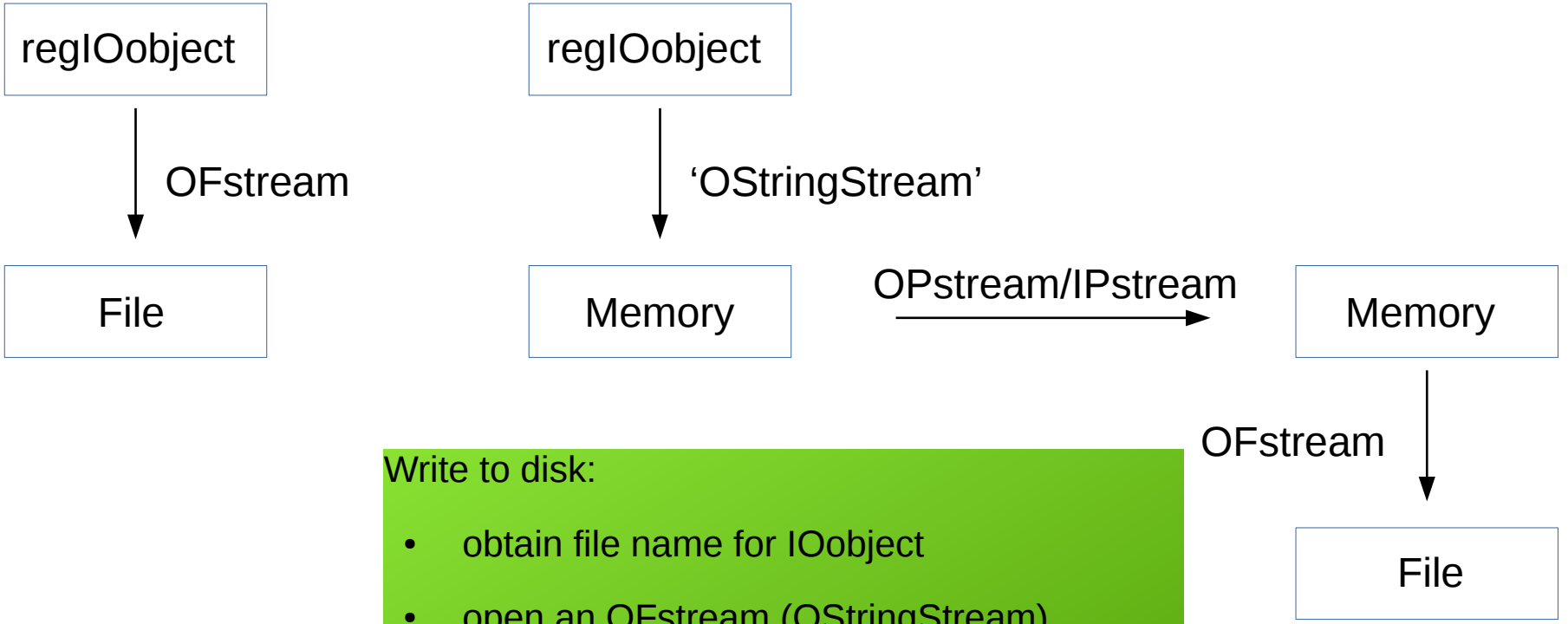
rack



master

masterUncollated

- Use streams!



Collated

- NFS: all processors access same storage
- I/O calls, directories scale with nProcs
- Instead: keep single file but collect all processor-contributions into single file
- processors/ instead of processor0/ .. processorN/
- file type ‘decomposedBlockData’

Collated (2)

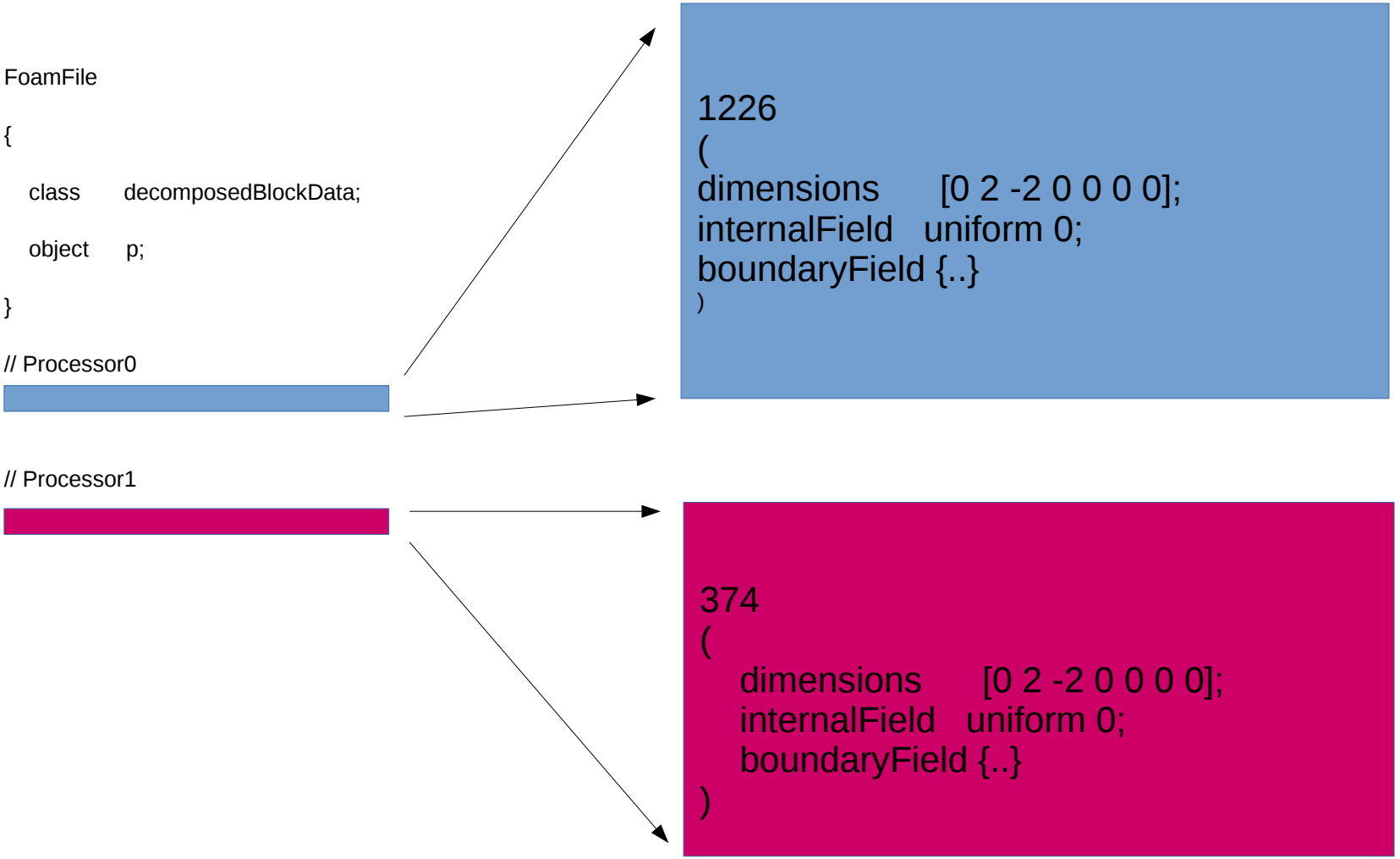
- ls cavity/

0 constant processors system

- ls cavity/processors/

0 0.1 0.2 0.3 constant

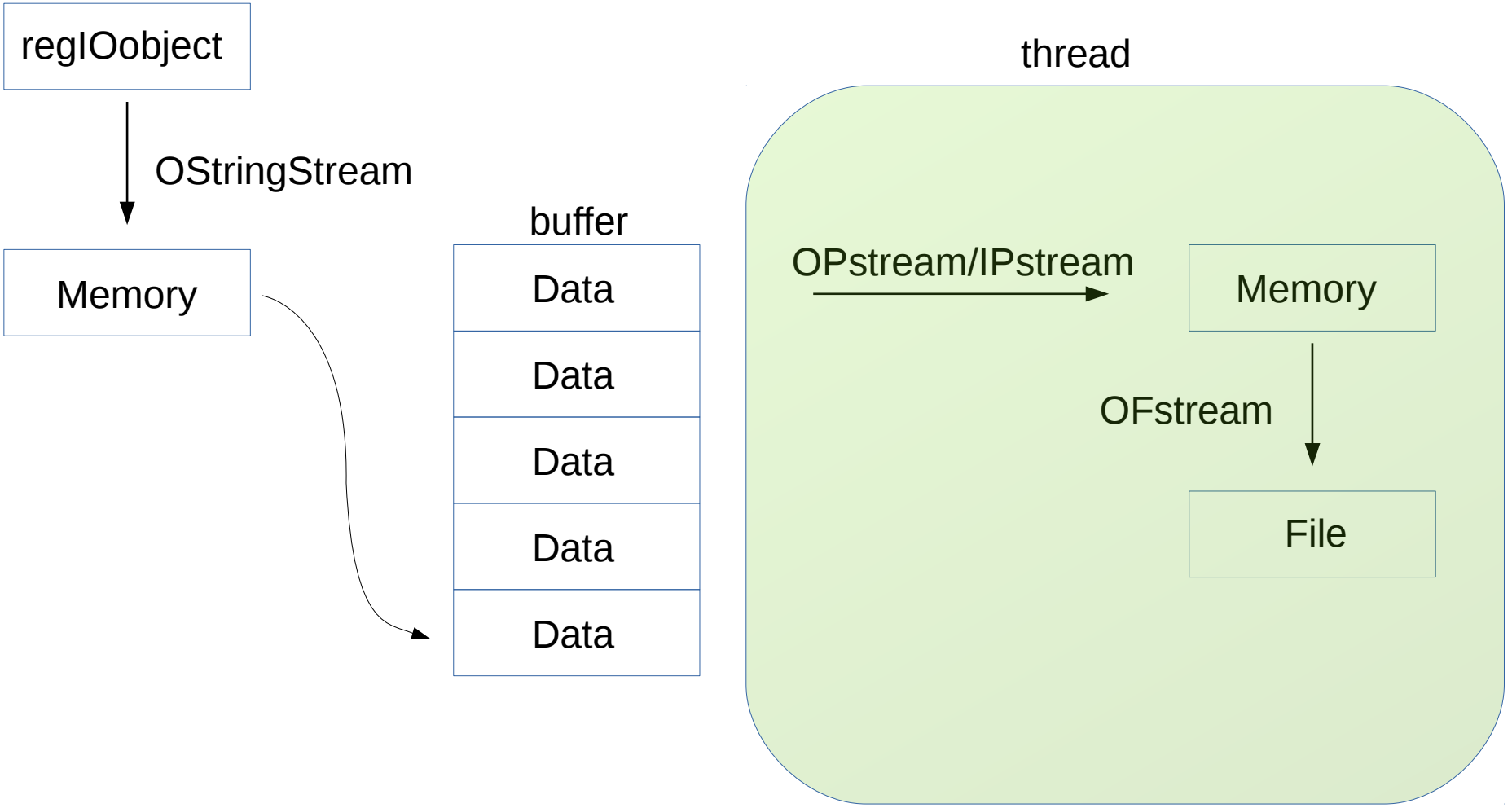
Collated (3)


FoamFile
{
class decomposedBlockData;
object p;
}
// Processor0
// Processor1
1226
(
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField {..}
)
374
(
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField {..}
)

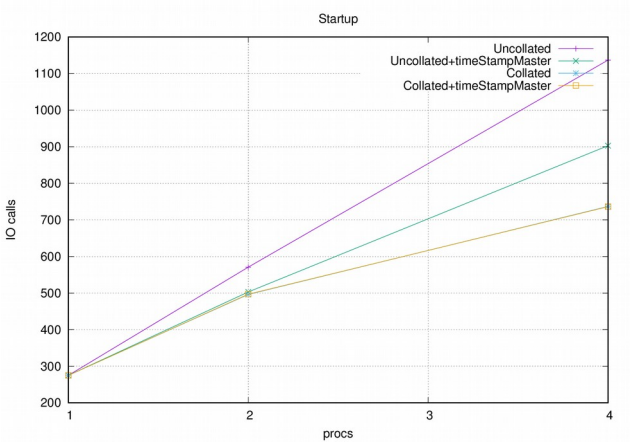
Collated (4)

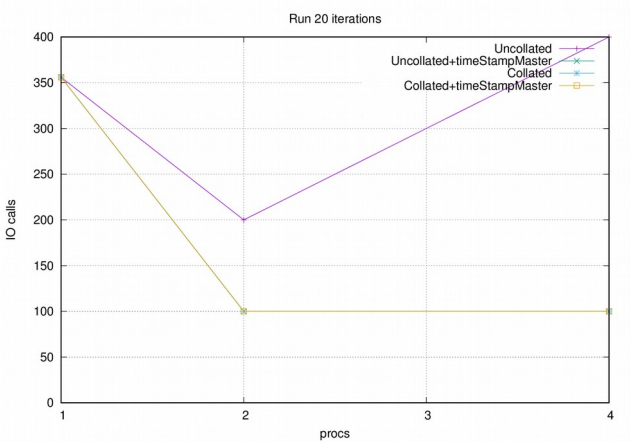
- slaves write to memory (OStringStream)
- at destruction time send over to master
- master processor receives all streamed data
- master processor writes to file
- master is bottleneck → thread!

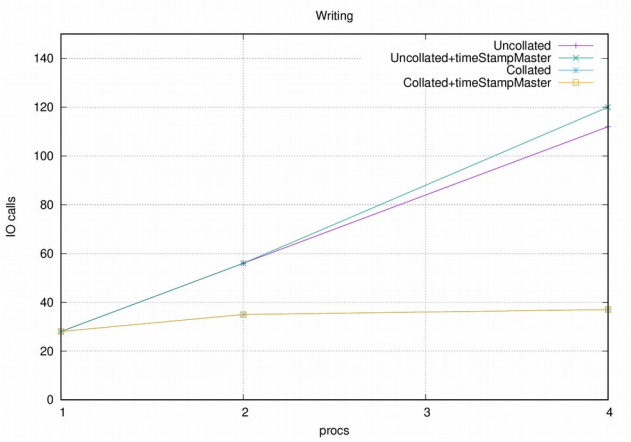
Collated (5)

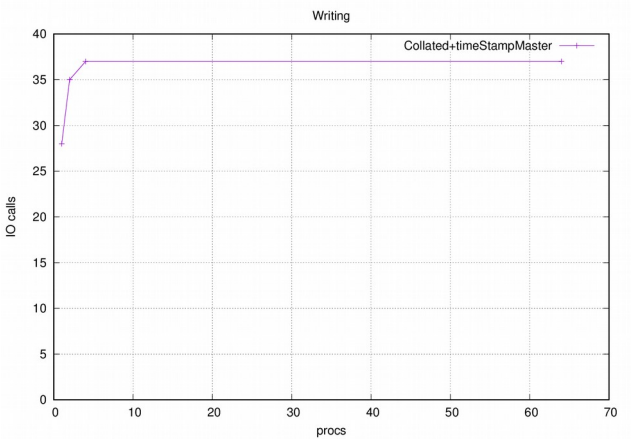

regIOobject
OStringStream
Memory
buffer
Data
Data
Data
Data
Data
thread
OPstream/IPstream
Memory
OFstream
File

Benchmark (simpleFoam)









Design

- at file level, not regIOobject
- no daisy-chaining of reading; hardcoded ‘collated’ format
- collated: master bottleneck
- collated with threading: needs thread-aware mpi
- processors/ : numbers of processors?

  processors256, processors10to19of256?
- parallel input could use optimisation
- user plug-ins

User plug-in

- regionUncollated : special handling for dictionaries (& fields)
- user library, on unpatched OpenFOAM version
- overrides ‘filePath’ and ‘read’
- searches parent directory of region
- loads parent dictionary and searches for region keyword

```
topAir
{
    dimensions      [ 0 0 0 1 0 0 0 ];
    internalField   uniform 300;
    ..
}

heater
{
    dimensions      [ 0 0 0 1 0 0 0 ];
    internalField   uniform 400;
    ..
}
```

```
"(topAir|bottomWater)"
{
    solvers
    {
        p_rgh    {solver GAMG;}
    }
}

"(heater|leftSolid|rightSolid)"
{
    solvers
    {
        h    {solver PCG;}
    }
}
```