Analysing Parallel I/O using Allinea Tools

Introduction to I/O profiling at scale

Keeran Brabazon, ARM

Parallel I/O Tutorial, ISC,18th June 2017

allinea
Now part of ARM

Allinea – What is it?

- Help the HPC community develop and design the best applications and make the most use of HPC clusters
- HPC Tools company 2002 – 2016
- Part of ARM since December 2016
  - Continue to improve tools for new uses in HPC
  - Support for all HPC applications and architectures

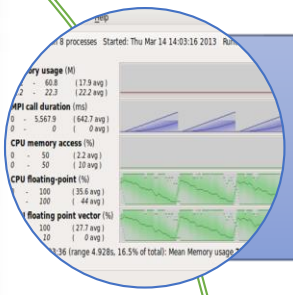
Allinea Products

- Allinea Forge
  - Combined debugging (DDT) and profiling (MAP) in the same interface
  - Designed for application developers

- Allinea Performance Reports
  - Summary of application performance
  - Designed for system administrators

- For more information see www.allinea.com

Performance Analysis with Allinea MAP

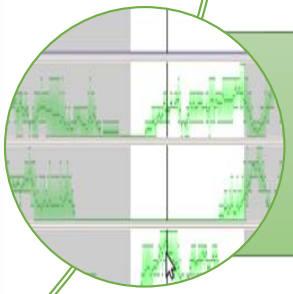


Low overhead measurement

- Accurate, non-intrusive application performance profiling
- Seamless – no recompilation or relinking required

Easy to use

- Source code viewer pinpoints bottleneck locations
- Zoom in to explore iterations, functions and loops



Deep

- Measures CPU, communication, I/O and memory to identify problem causes
- Identifies vectorization and cache performance

Scalability

- Allinea tools are to be used at scale
  - Use on 100k+ cores
  - Profiles remain on the order of 10s of MB whether running on 100 processes for 30 seconds or 100k processes for 10 hours
  - When evaluating performance of an HPC application you should run at HPC scales

Running Allinea MAP

- Program should be compiled with compiler optimisations left *on*
  - Optimisation should be done on production run
- Debug symbols need to be included
  - -g flag in GNU and Intel compilers
  - -Gfast compilation flag for Cray compiler

Running Allinea MAP

- Simply prefix `map --profile` to execution command
  - Also in batch submission script

- A `.map` is generated with filename including time-stamp in working directory
  - Use `-o <output_fname>.map` to write to named file

Running Allinea MAP

```
mpirun -n 1024 ./my_mpi_exe command line args …
```

```
map --profile mpirun -n 1024 ./my_mpi_exe \
command line args …
```

Running Allinea MAP

```
mpiexec -n 1024 ./my_mpi_exe command line args …
```

```
map --profile mpiexec -n 1024 ./my_mpi_exe \
command line args …
```

Running Allinea MAP

```
aprun -n 1024 ./my_mpi_exe command line args …
```

```
map --profile aprun -n 1024 ./my_mpi_exe \
command line args …
```

Viewing Performance Data in Allinea MAP

- Using the Allinea MAP GUI to view a single file

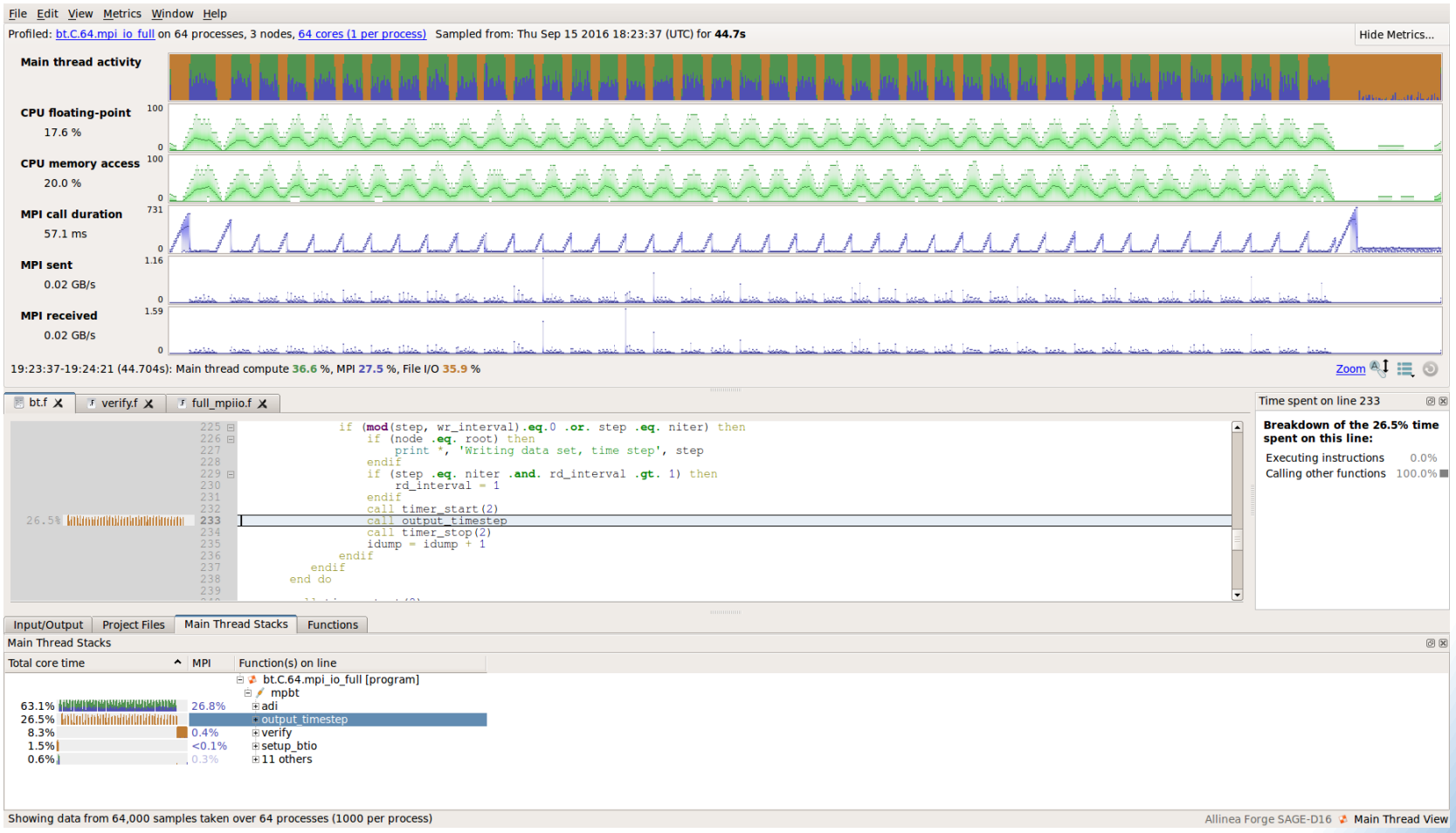
```
map my_map_file.map
```

- Using scripts to view data from multiple profiles
  1. Export to JSON

```
map --export=my_json_file.json /
        my_map_file.map
```

  2. Extract and view data. Example scripts at https://github.com/arm-hpc/allinea_json_analysis

Example MAP Profile



Visualising I/O Performance with Allinea MAP

Visualising I/O Performance

- Consider the application `benchio` from the earlier tutorial session

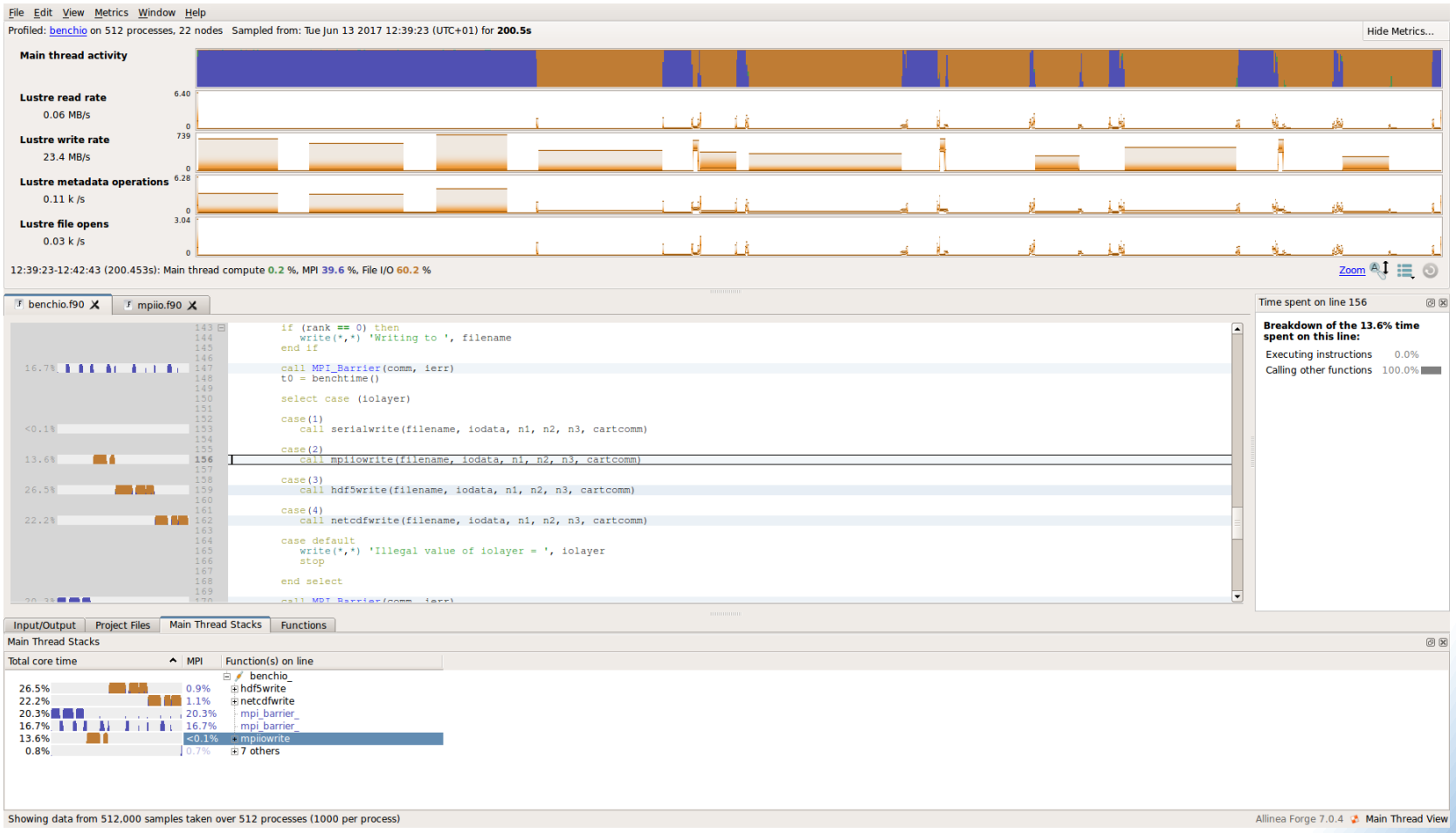
- Writes to file with:
  - 1 stripe
  - Lustre selected striping
  - Default striping (4 on Archer)

Visualising I/O Performance

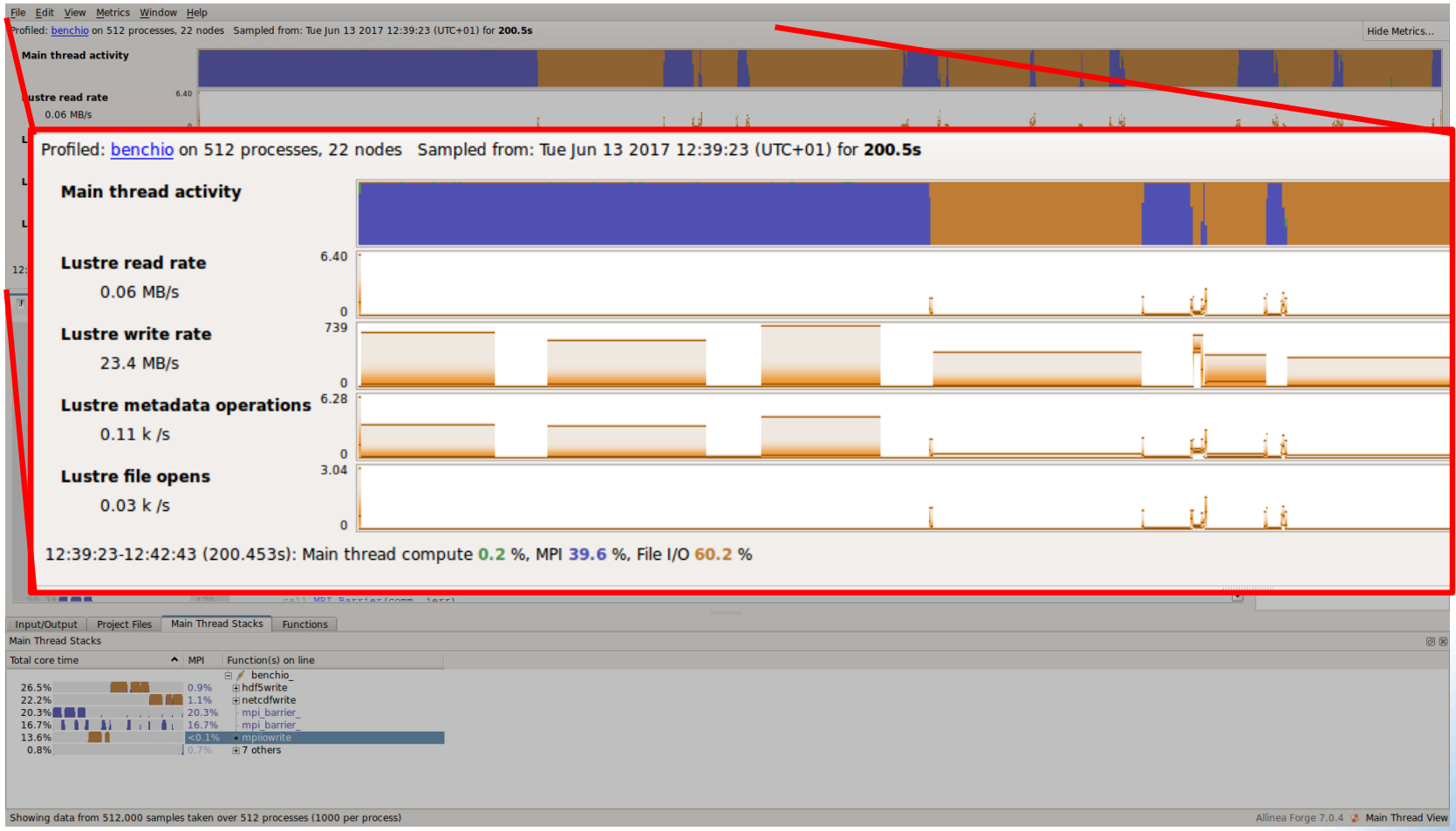
- Consider the application `benchio` from the earlier tutorial session

- Writes to file using:
  - Serial I/O
  - MPI I/O
  - HDF5
  - NetCDF

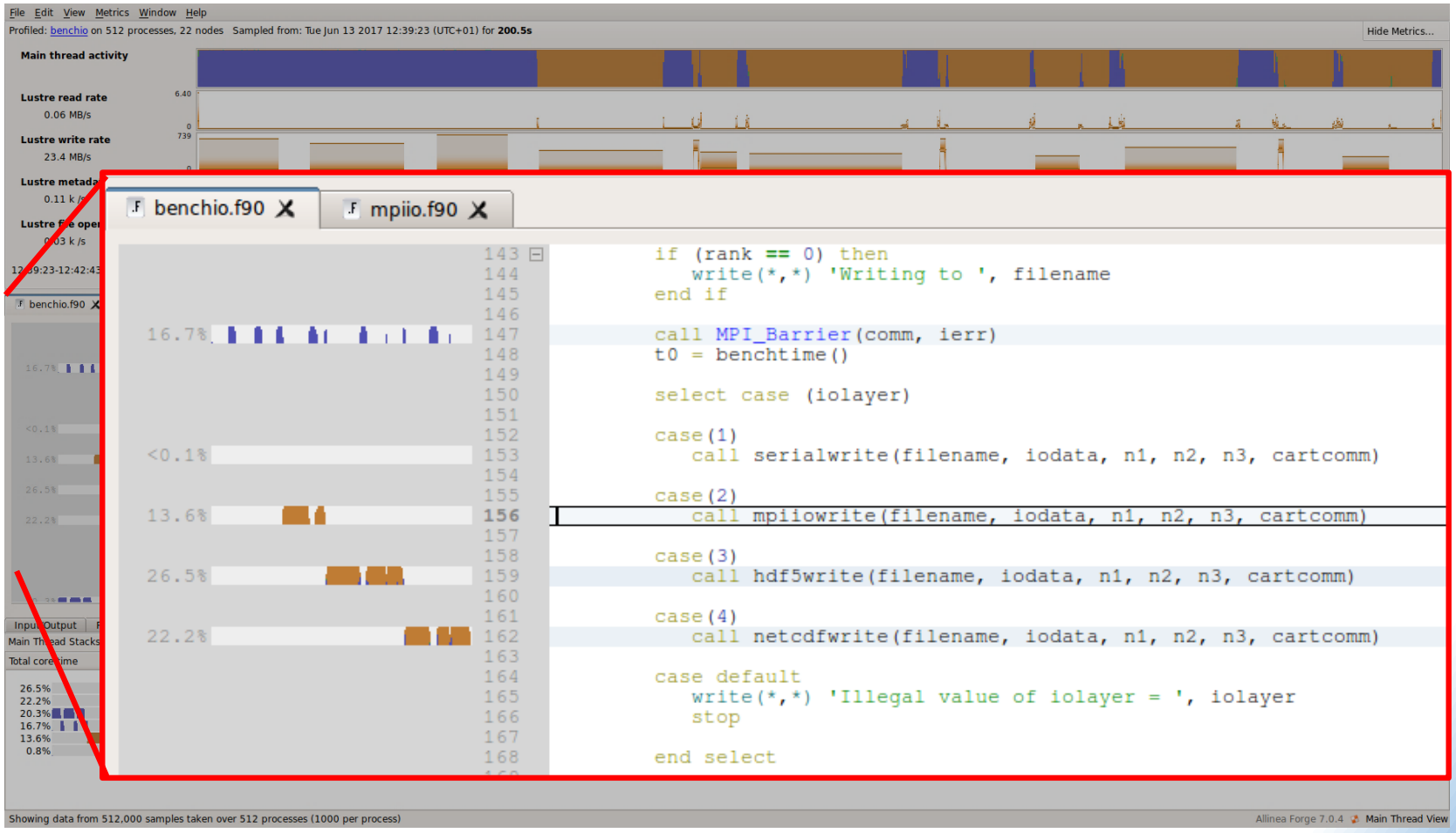
Visualising I/O Performance



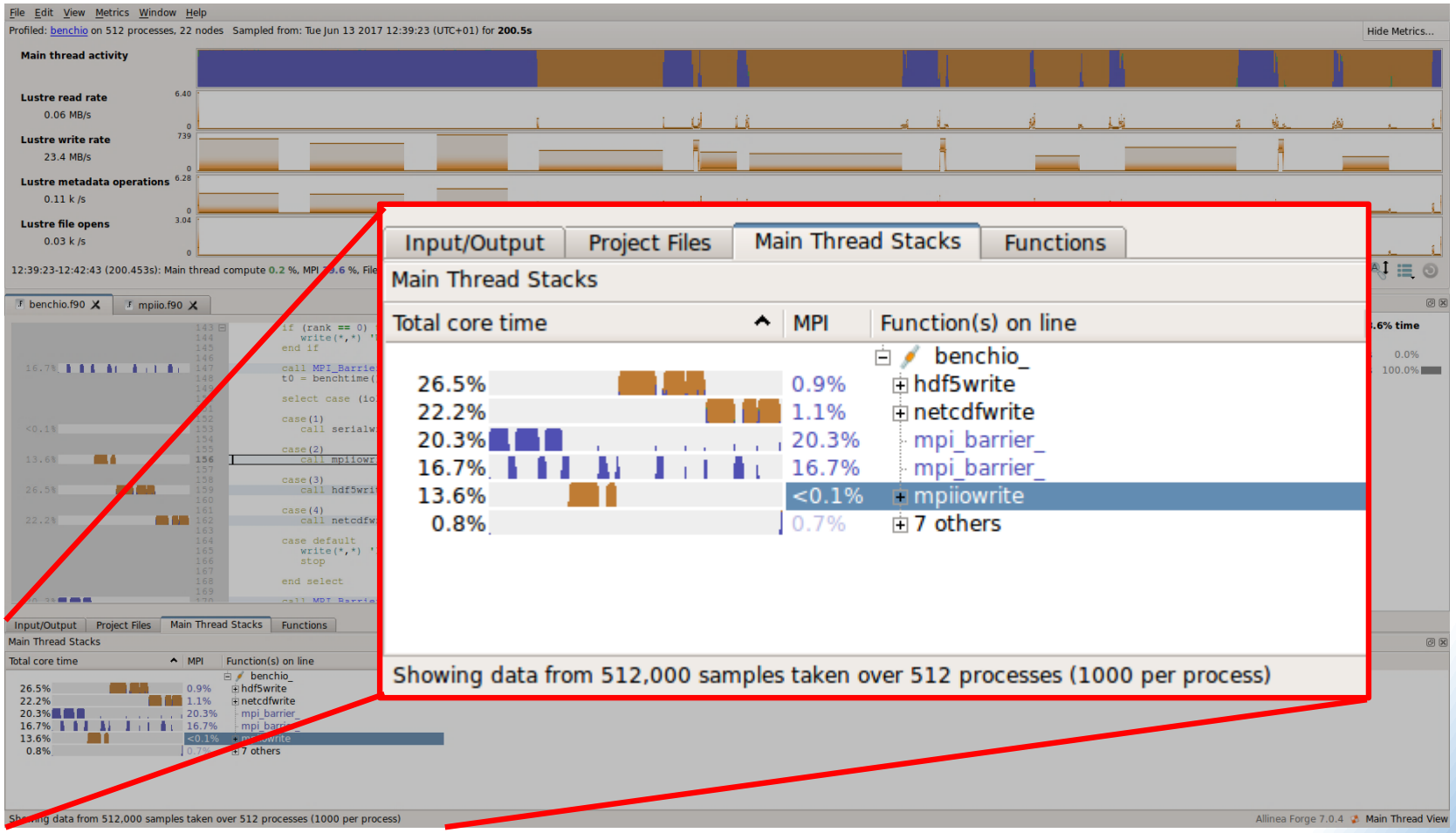
Visualising I/O Performance



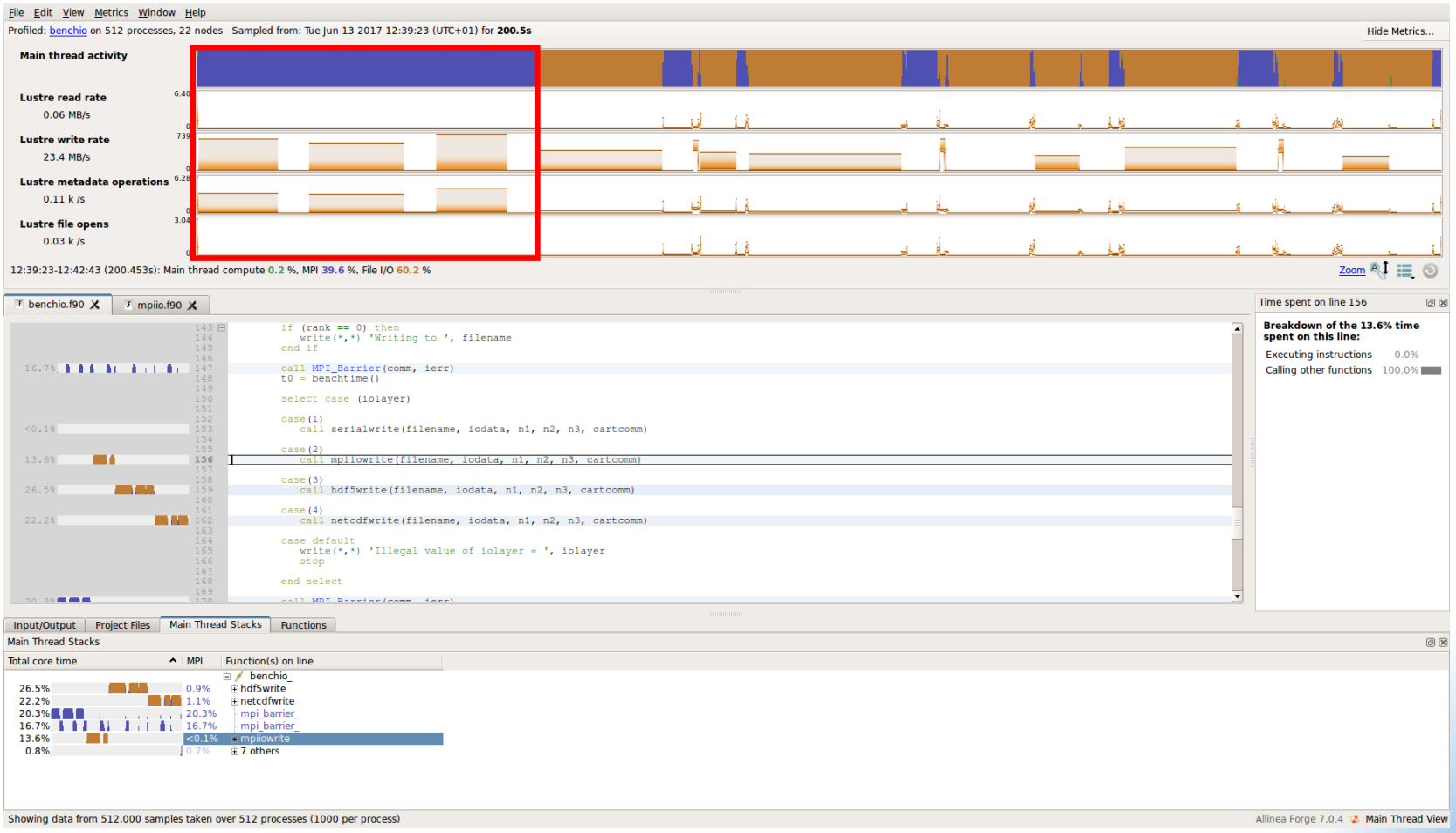
Visualising I/O Performance



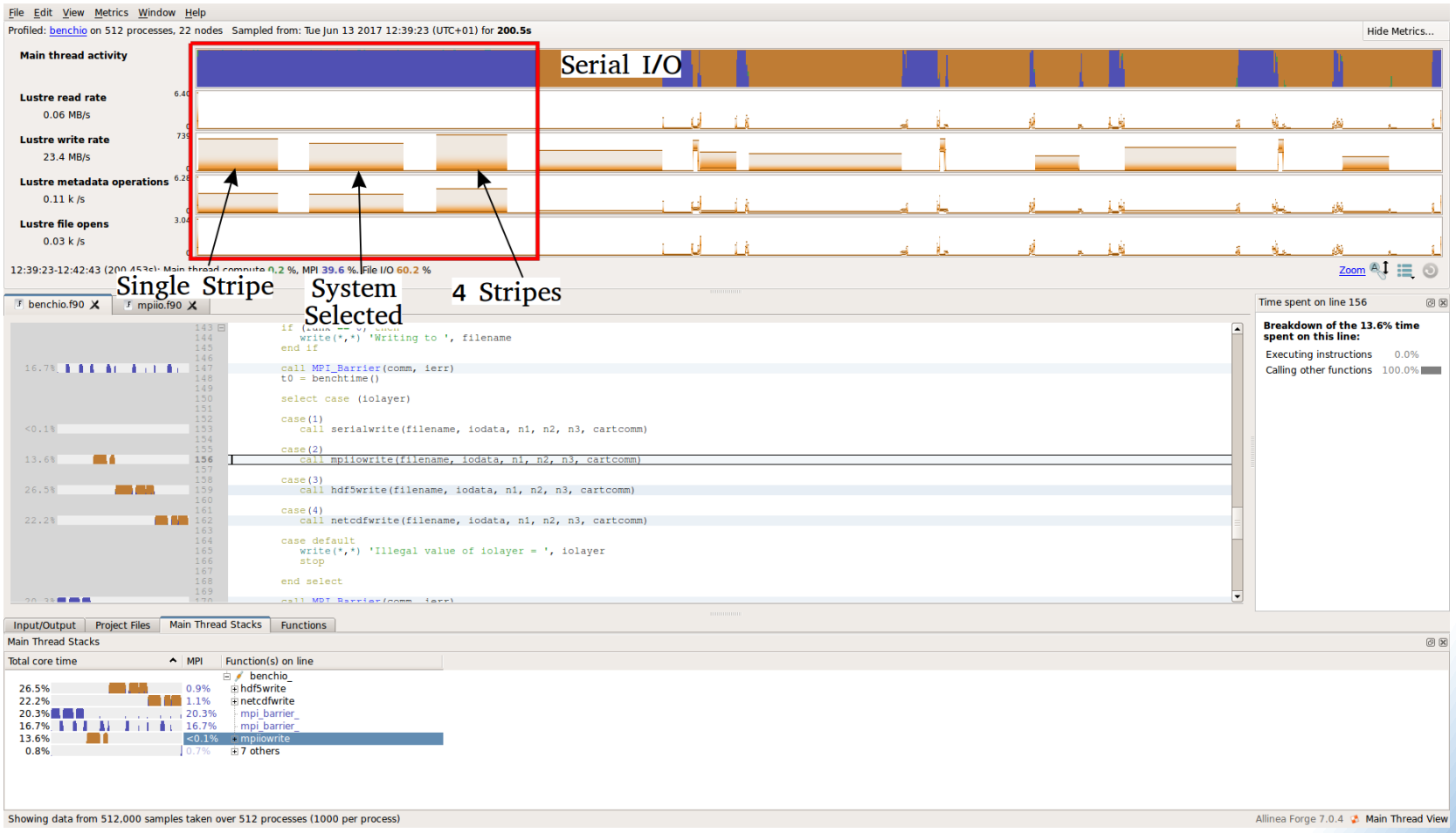
Visualising I/O Performance



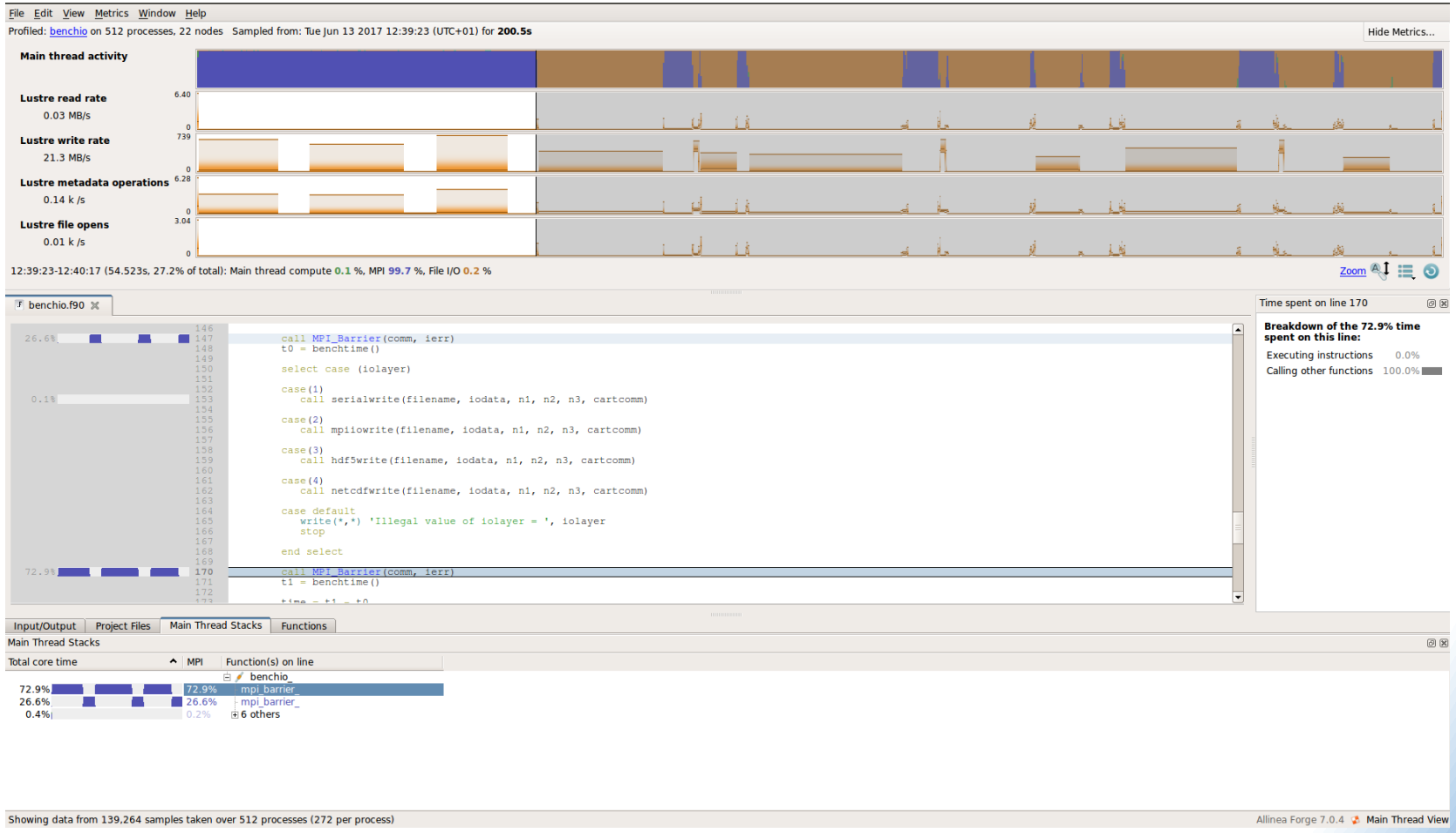
Visualising I/O Performance



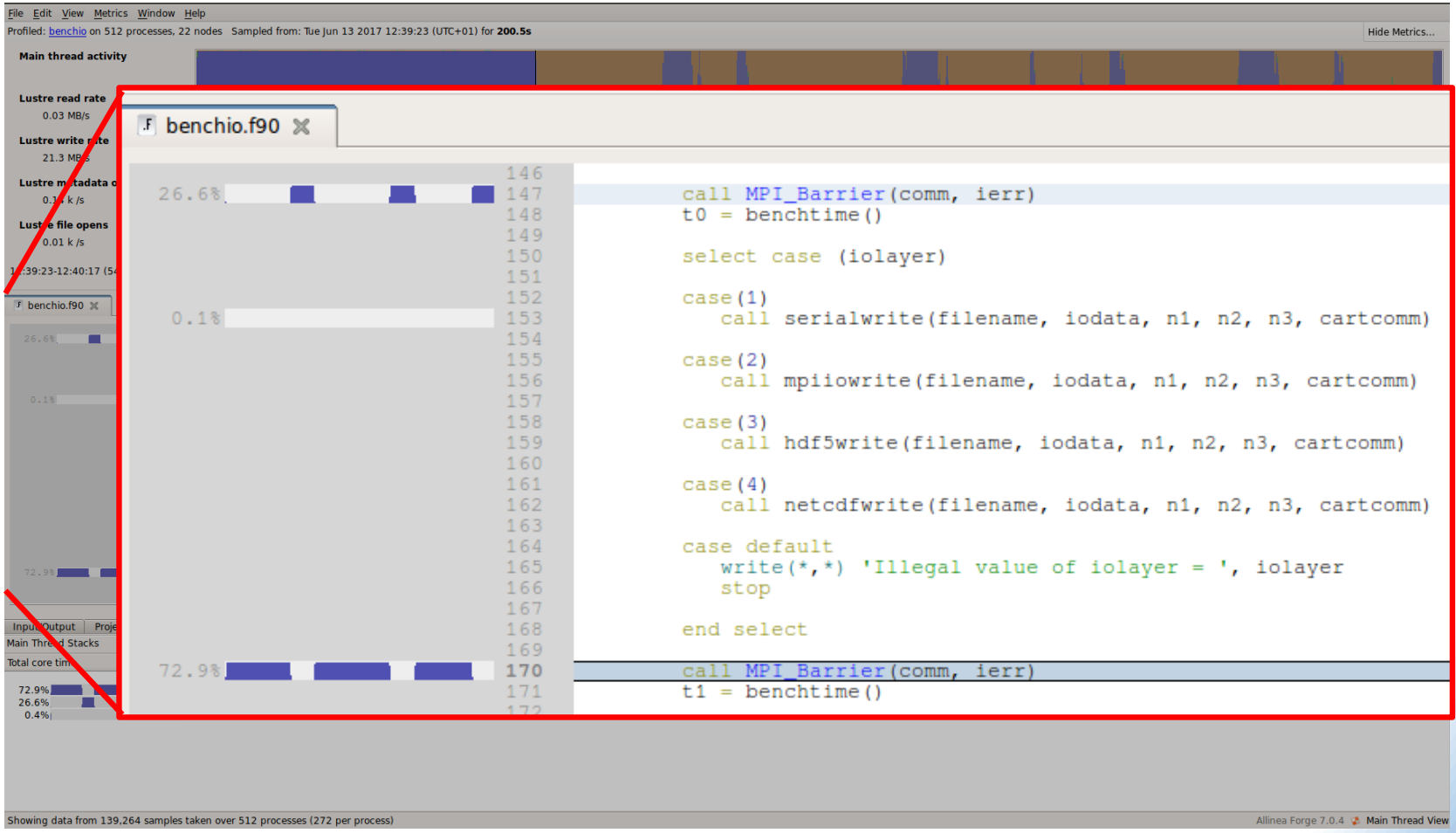
Visualising I/O Performance



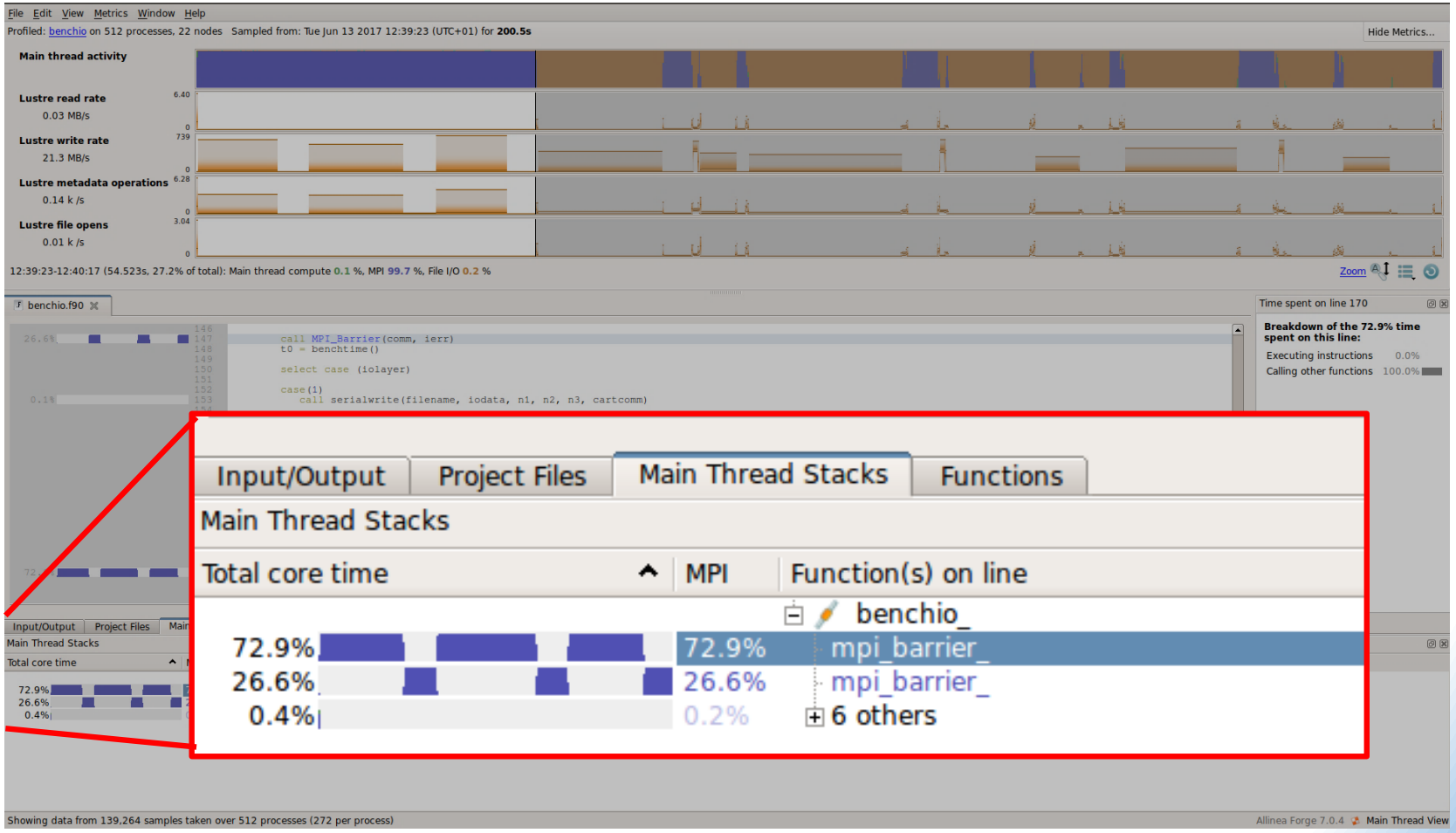
Visualising I/O Performance



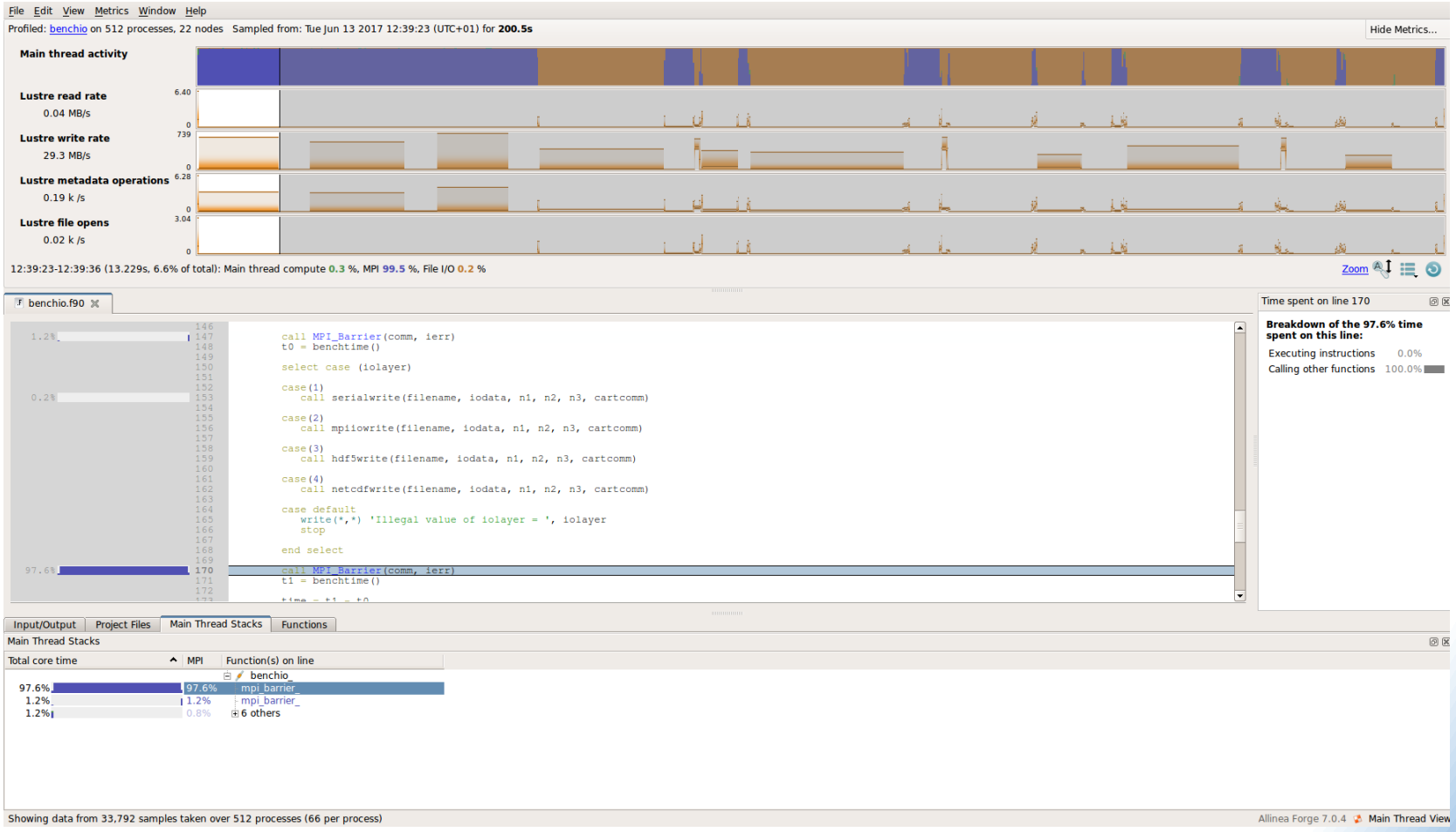
Visualising I/O Performance



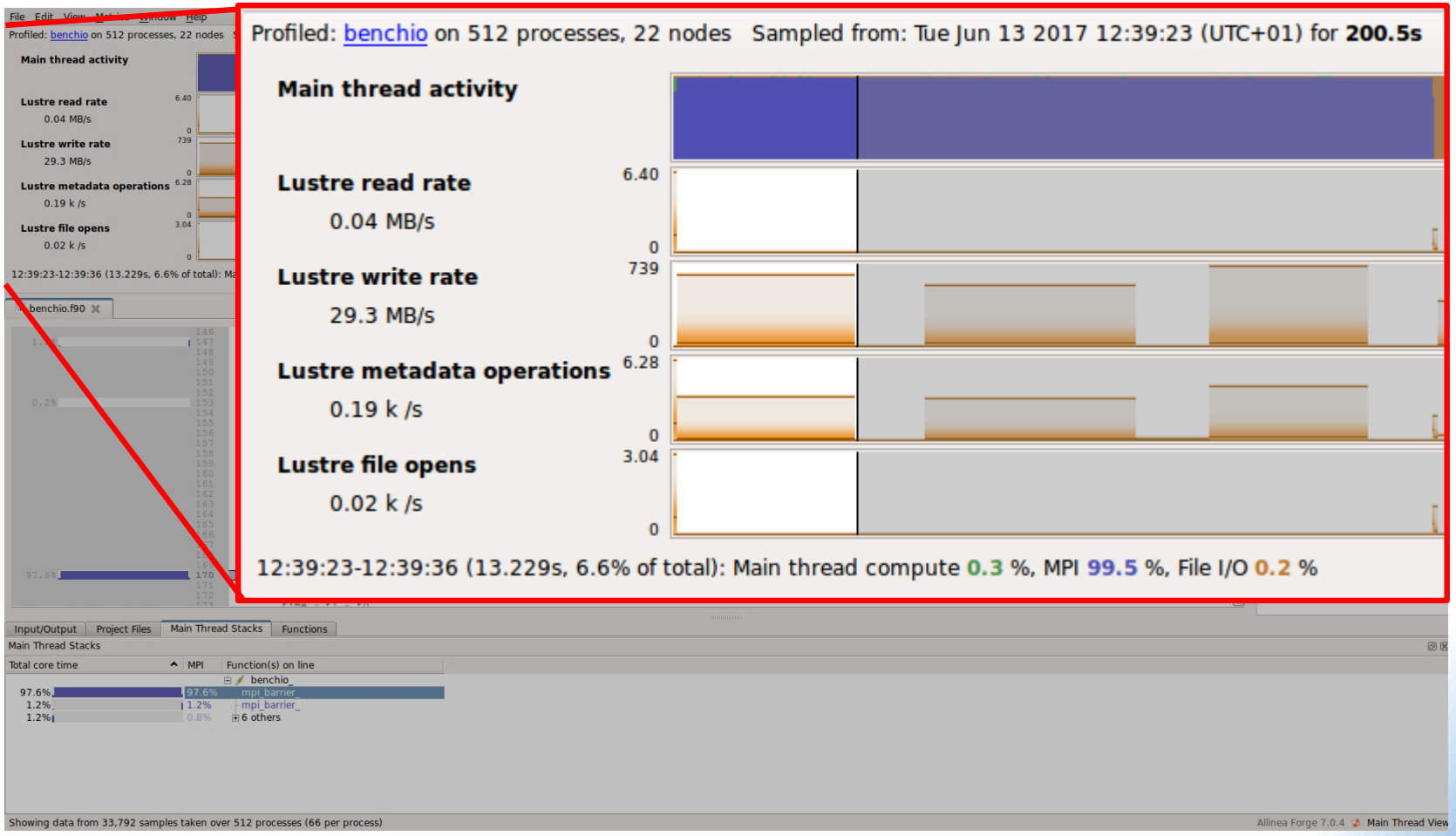
Visualising I/O Performance



Visualising I/O Performance



Visualising I/O Performance



Summary

- Interesting behaviour can be viewed in a single profile

- Viewing data in a timeline can show extra information with respect to the benchmark output

Summary

- Further investigation of the profiles to be done in the hands-on sessions
- Consider not only the bandwidth, but the overhead, which is shown as ‘gaps’ in the write rate in the profiles collected
- To draw conclusions on the performance of different paradigms need a set of runs, at different scales with different volumes of data written (in total and per-process)

Questions?

keeran.brabazon@arm.com