allinea
Now part of ARM

High performance tools to debug, profile, and analyze your applications

Analyzing I/O Profiles

I/O Profiling at Scale

Keeran Brabazon, ARM

DKRZ, UIOP Workshop
23rd March 2017

allinea FORGE allinea DDT allinea MAP allinea PERFORMANCE REPORTS

Acknowledgements

SAGE

KTH:
Stefano Markidis, Sergio Rivas Gomez, Bo Peng

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Allinea – What is it?

- HPC Tools company since 2002

- Help the HPC community develop and design the best applications and make the most use of HPC clusters

- Part of ARM since December 2016
  - Continue to improve tools for new uses in HPC
  - Support for all HPC applications and hardwares

Products

- Allinea Forge
  - Combined debugging and profiling in same interface
  - Designed for application developers

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

Profiling with Allinea MAP

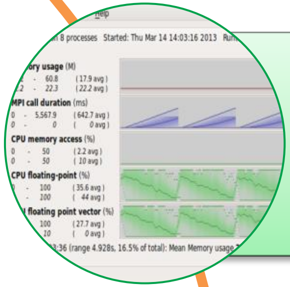
allinea
MAP

Allinea MAP

- Use of performance analysis tools can help to focus attention on the parts of a program with worst performance

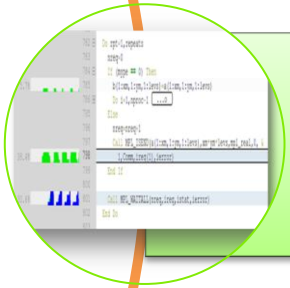
- Allinea MAP can do so for applications running with 100k+ processes

Allinea MAP



Low overhead measurement

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

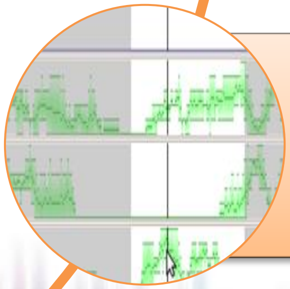


```
Do rpt=1,repeats
 nreq=0
 If (mype == 0) Then
  b(1:nx,1:ny,1:levs)=a(1:nx,1:ny,1:levs)
  Do i=1,nproc-1 ...
 Else
  nreq=nreq+1
  Call MPI_ISEND(a(1:nx,1:ny,1:levs),nx*ny*levs,mpi_real,0, &
   1,Comm,ireq(1),ierror)
 End If

 Call MPI_WAITALL(nreq,ireq,istat,ierror)
End Do
```

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

I/O Profiling – Worked Example

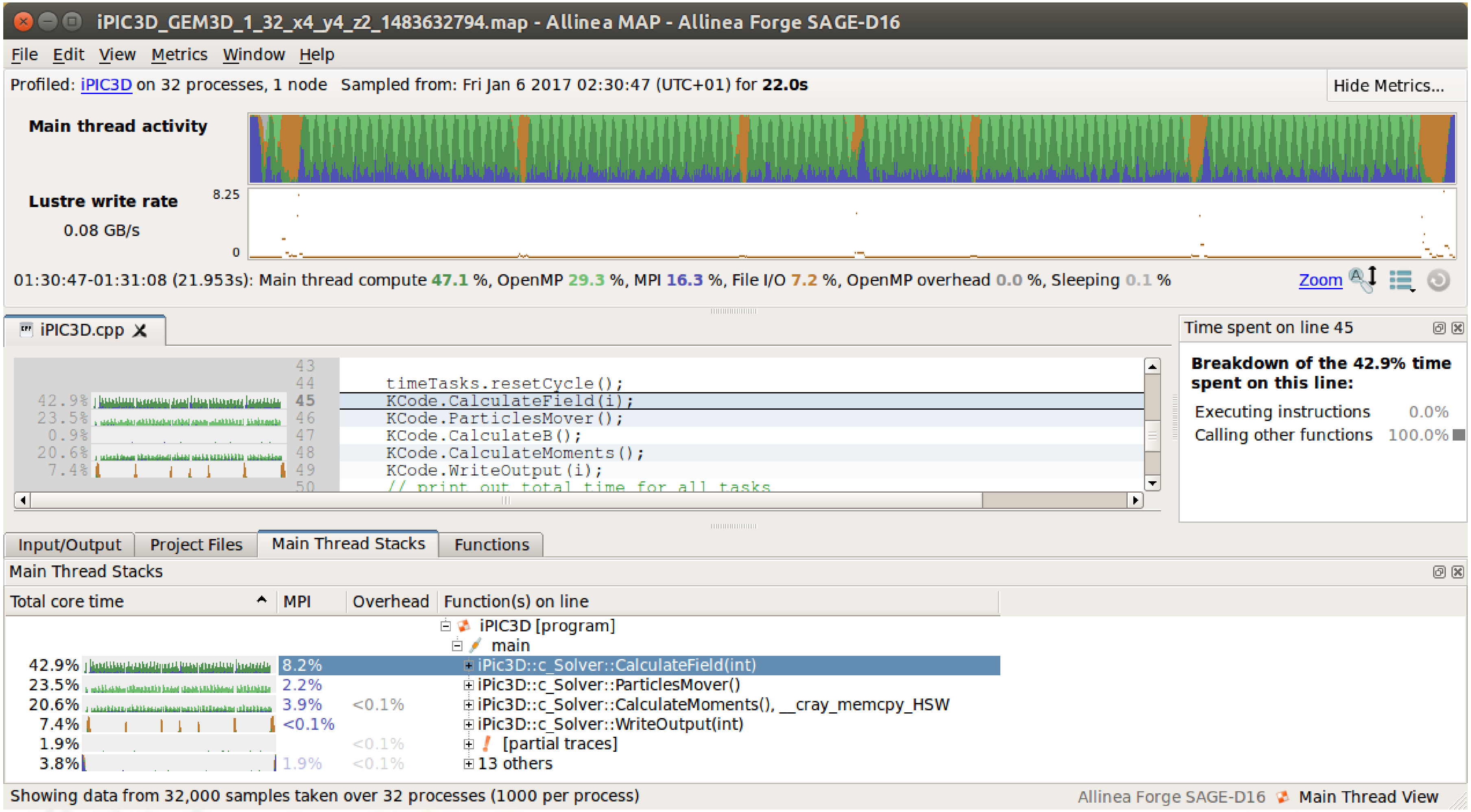
iPIC3D

allinea
Now part of ARM

iPIC3D

- Particle-in-cell code to model interaction between solar wind and Earth’s magnetic field
- Practical problem sizes have billions of particles with velocity, current and charge density
- I/O performed for visualization (every 20 iterations) and checkpointing (every 50 iterations)
- Run on Beskow – Cray XC40, 32 Broadwell per Node

iPIC3D Profiling – 32 Processes (1 node)



Profiled: iPIC3D on 32 processes, 1 node Sampled from: Fri Jan 6 2017 02:30:47 (UTC+01) for **22.0s**

Lustre write rate 0.08 GB/s

01:30:47-01:31:08 (21.953s): Main thread compute 47.1 %, OpenMP 29.3 %, MPI 16.3 %, File I/O 7.2 %, OpenMP overhead 0.0 %, Sleeping 0.1 %

```
        43
        44      timeTasks.resetCycle();
42.9%   45      KCode.CalculateField(i);
23.5%   46      KCode.ParticlesMover();
 0.9%   47      KCode.CalculateB();
20.6%   48      KCode.CalculateMoments();
 7.4%   49      KCode.WriteOutput(i);
        50      // print out total time for all tasks
```

Time spent on line 45

Breakdown of the 42.9% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

Main Thread Stacks

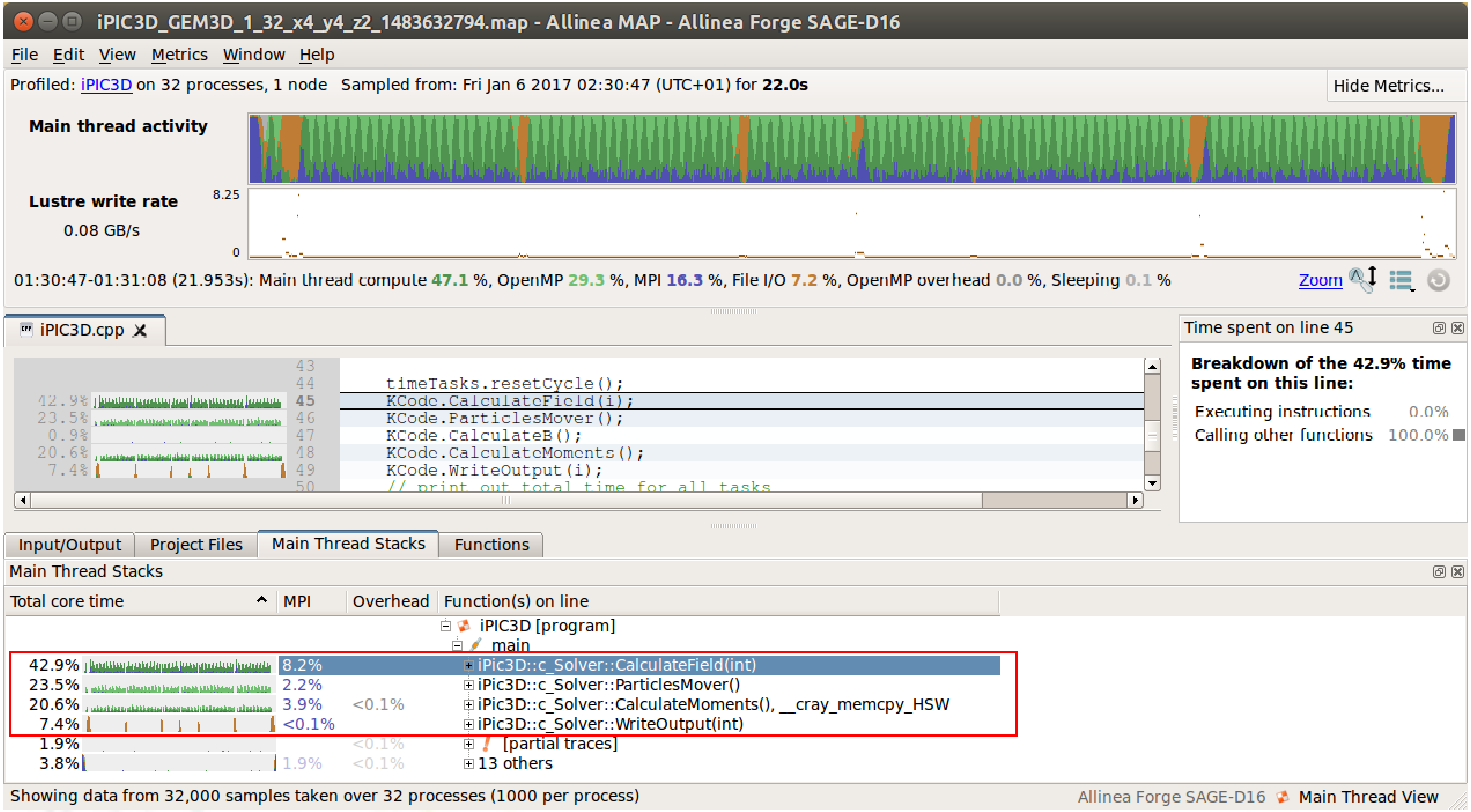
Total core time	MPI	Overhead	Function(s) on line
			iPIC3D [program]
			main
42.9%	8.2%		iPic3D::c_Solver::CalculateField(int)
23.5%	2.2%		iPic3D::c_Solver::ParticlesMover()
20.6%	3.9%	<0.1%	iPic3D::c_Solver::CalculateMoments(), __cray_memcpy_HSW
7.4%	<0.1%		iPic3D::c_Solver::WriteOutput(int)
1.9%		<0.1%	[partial traces]
3.8%	1.9%	<0.1%	13 others

Showing data from 32,000 samples taken over 32 processes (1000 per process)

Allinea Forge SAGE-D16 Main Thread View

allinea
Now part of ARM

iPIC3D Profiling – 32 Processes (1 node)



iPIC3D_GEM3D_1_32_x4_y4_z2_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

Profiled: iPIC3D on 32 processes, 1 node Sampled from: Fri Jan 6 2017 02:30:47 (UTC+01) for **22.0s**

01:30:47-01:31:08 (21.953s): Main thread compute 47.1 %, OpenMP 29.3 %, MPI 16.3 %, File I/O 7.2 %, OpenMP overhead 0.0 %, Sleeping 0.1 %

```
                43
                44      timeTasks.resetCycle();
42.9%           45      KCode.CalculateField(i);
23.5%           46      KCode.ParticlesMover();
 0.9%           47      KCode.CalculateB();
20.6%           48      KCode.CalculateMoments();
 7.4%           49      KCode.WriteOutput(i);
                50      // print out total time for all tasks
```

Time spent on line 45

Breakdown of the 42.9% time spent on this line:

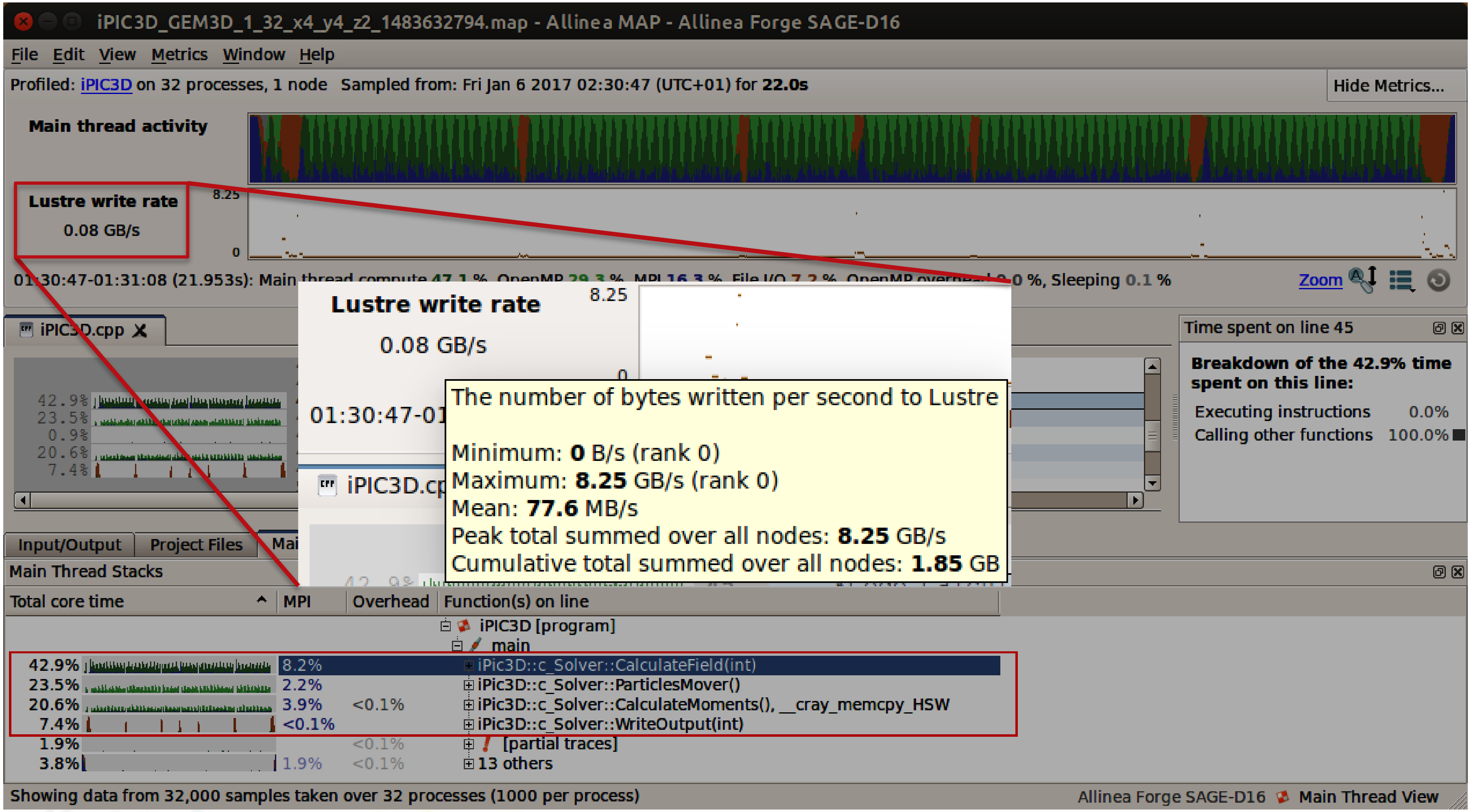
Executing instructions	0.0%
Calling other functions	100.0%

Main Thread Stacks

Total core time	MPI	Overhead	Function(s) on line
			iPIC3D [program]
			main
42.9%	8.2%		iPic3D::c_Solver::CalculateField(int)
23.5%	2.2%		iPic3D::c_Solver::ParticlesMover()
20.6%	3.9%	<0.1%	iPic3D::c_Solver::CalculateMoments(), __cray_memcpy_HSW
7.4%	<0.1%		iPic3D::c_Solver::WriteOutput(int)
1.9%		<0.1%	[partial traces]
3.8%	1.9%	<0.1%	13 others

Showing data from 32,000 samples taken over 32 processes (1000 per process)

iPIC3D Profiling: 32 Processes (1 node)



iPIC3D_GEM3D_1_32_x4_y4_z2_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

Profiled: iPIC3D on 32 processes, 1 node   Sampled from: Fri Jan 6 2017 02:30:47 (UTC+01) for **22.0s**

Lustre write rate 0.08 GB/s

01:30:47-01:31:08 (21.953s): Main thread compute 47.1 %, OpenMP 29.3 %, MPI 16.3 %, File I/O 7.2 %, OpenMP overhead 0.0 %, Sleeping 0.1 %

Lustre write rate

0.08 GB/s

01:30:47-01

The number of bytes written per second to Lustre

Minimum: **0** B/s (rank 0)
Maximum: **8.25** GB/s (rank 0)
Mean: **77.6** MB/s
Peak total summed over all nodes: **8.25** GB/s
Cumulative total summed over all nodes: **1.85** GB

Time spent on line 45

Breakdown of the 42.9% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

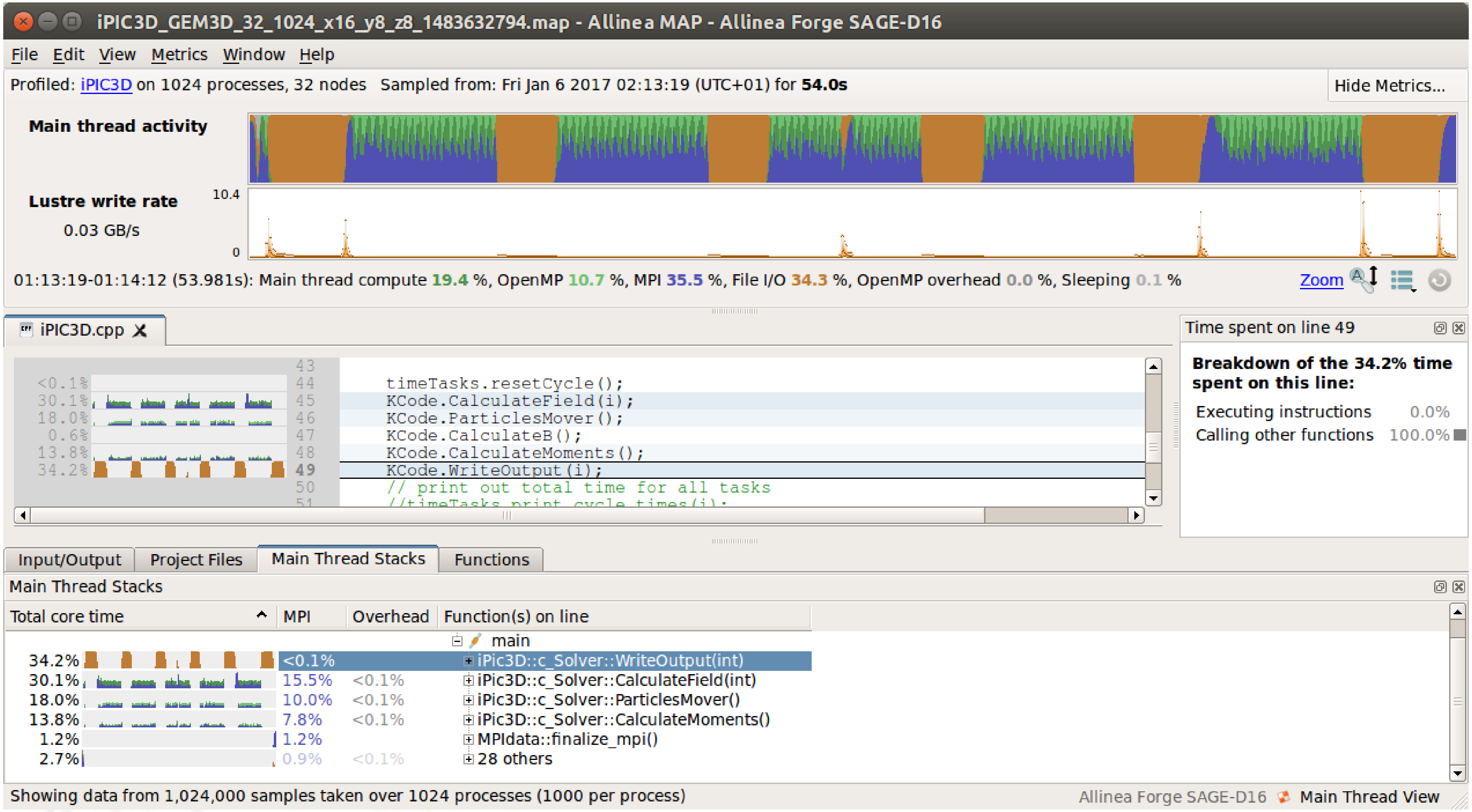
Total core time	MPI	Overhead	Function(s) on line
			iPIC3D [program]
			main
42.9%	8.2%		iPic3D::c_Solver::CalculateField(int)
23.5%	2.2%		iPic3D::c_Solver::ParticlesMover()
20.6%	3.9%	<0.1%	iPic3D::c_Solver::CalculateMoments(), __cray_memcpy_HSW
7.4%	<0.1%		iPic3D::c_Solver::WriteOutput(int)
1.9%		<0.1%	[partial traces]
3.8%	1.9%	<0.1%	13 others

Showing data from 32,000 samples taken over 32 processes (1000 per process)   Allinea Forge SAGE-D16   Main Thread View

iPIC3D Profiling

- I/O does not take up a large amount of run time
- 32 processes is rather small – go to larger core counts with more I/O performed

iPIC3D Profiling: 1024 Processes (32 Nodes)



Profiled: iPIC3D on 1024 processes, 32 nodes Sampled from: Fri Jan 6 2017 02:13:19 (UTC+01) for **54.0s**

01:13:19-01:14:12 (53.981s): Main thread compute 19.4 %, OpenMP 10.7 %, MPI 35.5 %, File I/O 34.3 %, OpenMP overhead 0.0 %, Sleeping 0.1 %

```
         43
 <0.1%   44      timeTasks.resetCycle();
 30.1%   45      KCode.CalculateField(i);
 18.0%   46      KCode.ParticlesMover();
  0.6%   47      KCode.CalculateB();
 13.8%   48      KCode.CalculateMoments();
 34.2%   49      KCode.WriteOutput(i);
         50      // print out total time for all tasks
         51      //timeTasks.print_cycle_times(i);
```

Time spent on line 49

Breakdown of the 34.2% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

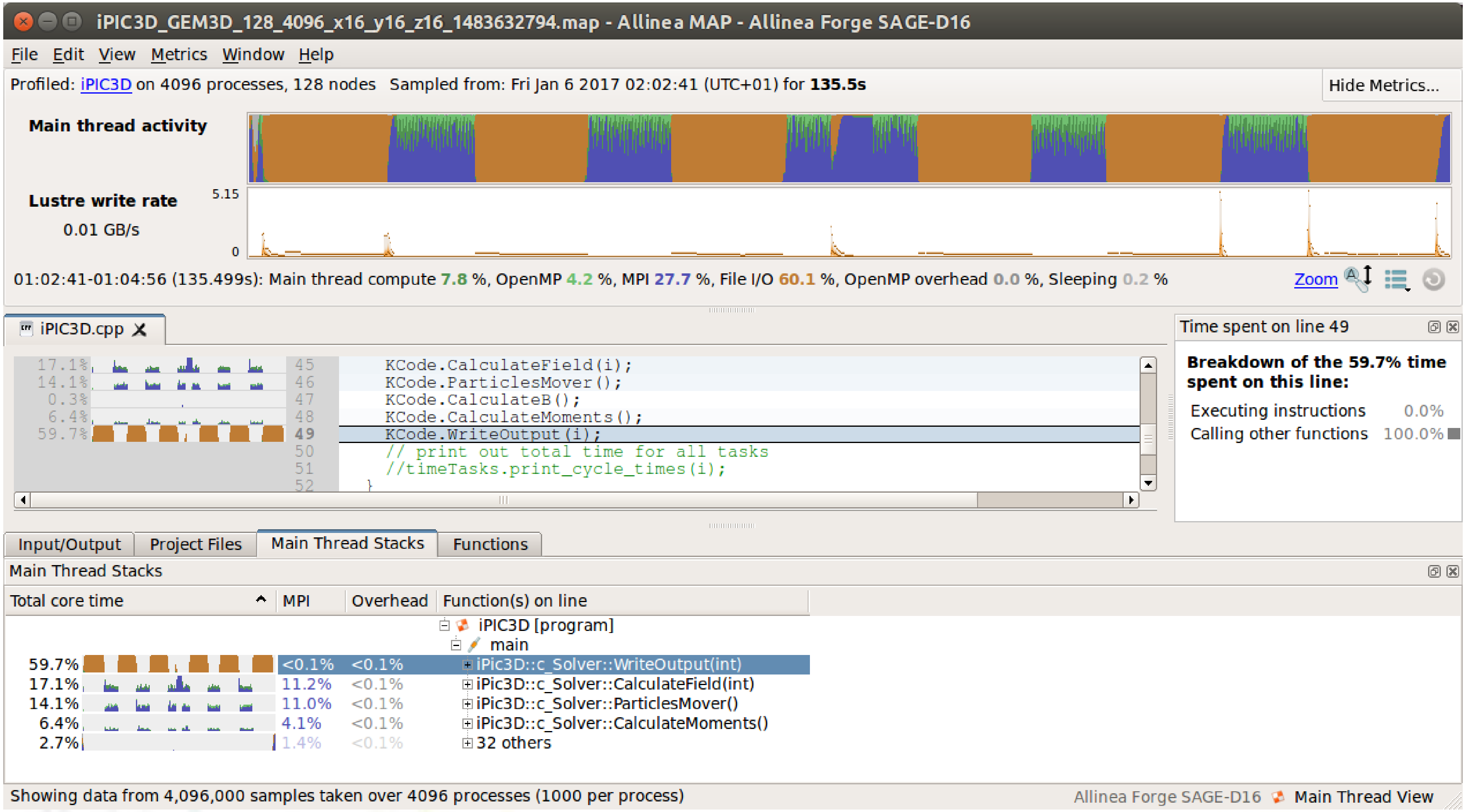
Main Thread Stacks

Total core time	MPI	Overhead	Function(s) on line
			main
34.2%	<0.1%		iPic3D::c_Solver::WriteOutput(int)
30.1%	15.5%	<0.1%	iPic3D::c_Solver::CalculateField(int)
18.0%	10.0%	<0.1%	iPic3D::c_Solver::ParticlesMover()
13.8%	7.8%	<0.1%	iPic3D::c_Solver::CalculateMoments()
1.2%	1.2%		MPIdata::finalize_mpi()
2.7%	0.9%	<0.1%	28 others

Showing data from 1,024,000 samples taken over 1024 processes (1000 per process)

allinea
Now part of ARM

iPIC3D Profiling: 4096 Processes (128 Nodes)



iPIC3D_GEM3D_128_4096_x16_y16_z16_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

01:02:41-01:04:56 (135.499s): Main thread compute 7.8 %, OpenMP 4.2 %, MPI 27.7 %, File I/O 60.1 %, OpenMP overhead 0.0 %, Sleeping 0.2 %

```
17.1%  45      KCode.CalculateField(i);
14.1%  46      KCode.ParticlesMover();
 0.3%  47      KCode.CalculateB();
 6.4%  48      KCode.CalculateMoments();
59.7%  49      KCode.WriteOutput(i);
       50      // print out total time for all tasks
       51      //timeTasks.print_cycle_times(i);
       52    }
```

Time spent on line 49

Breakdown of the 59.7% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

Main Thread Stacks

Total core time	MPI	Overhead	Function(s) on line
			iPIC3D [program]
			main
59.7%	<0.1%	<0.1%	iPic3D::c_Solver::WriteOutput(int)
17.1%	11.2%	<0.1%	iPic3D::c_Solver::CalculateField(int)
14.1%	11.0%	<0.1%	iPic3D::c_Solver::ParticlesMover()
6.4%	4.1%	<0.1%	iPic3D::c_Solver::CalculateMoments()
2.7%	1.4%	<0.1%	32 others

Showing data from 4,096,000 samples taken over 4096 processes (1000 per process)

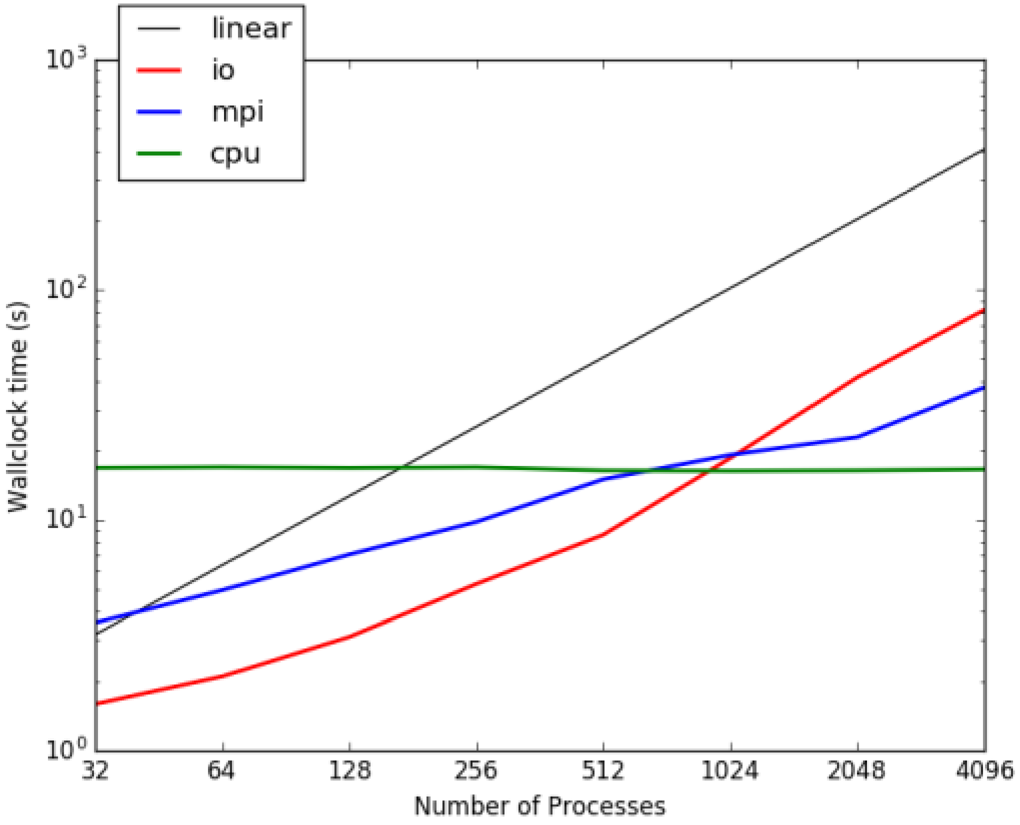
Allinea Forge SAGE-D16 Main Thread View

allinea
Now part of ARM

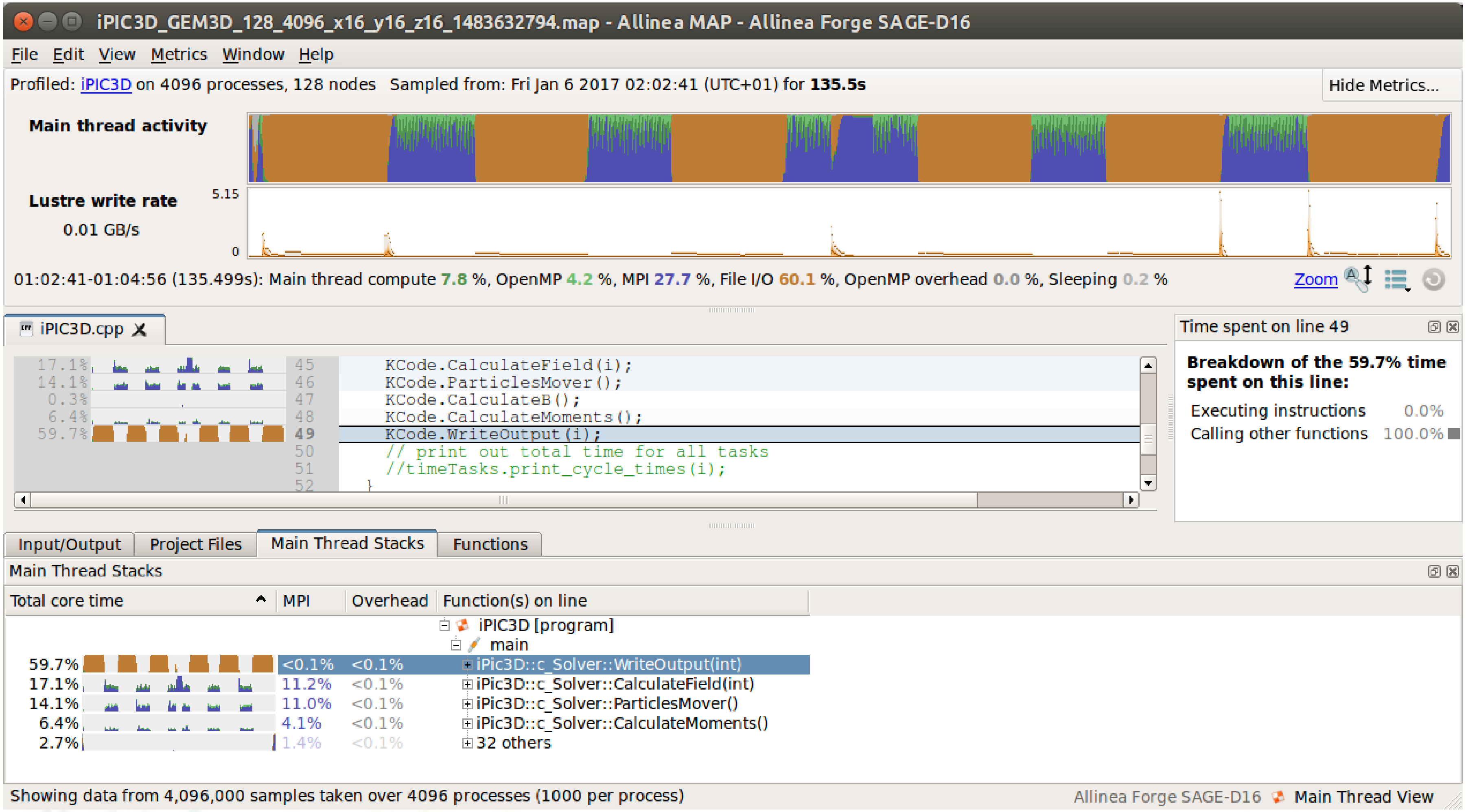
iPIC3D Profiling

- Experiments were run to show weak scaling
- Scaling behaviour cumbersome to view in MAP files
- Export to JSON of profile (new in Forge version 7.0+) allows user to post-process and visualise data

iPIC3D – Weak Scaling



iPIC3D Profiling: 4096 Processes (128 Nodes)



iPIC3D_GEM3D_128_4096_x16_y16_z16_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

File Edit View Metrics Window Help

Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

Hide Metrics...

01:02:41-01:04:56 (135.499s): Main thread compute 7.8 %, OpenMP 4.2 %, MPI 27.7 %, File I/O 60.1 %, OpenMP overhead 0.0 %, Sleeping 0.2 %

iPIC3D.cpp

```
17.1%  45      KCode.CalculateField(i);
14.1%  46      KCode.ParticlesMover();
 0.3%  47      KCode.CalculateB();
 6.4%  48      KCode.CalculateMoments();
59.7%  49      KCode.WriteOutput(i);
       50      // print out total time for all tasks
       51      //timeTasks.print_cycle_times(i);
       52    }
```

Time spent on line 49

Breakdown of the 59.7% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

Input/Output | Project Files | Main Thread Stacks | Functions

Main Thread Stacks

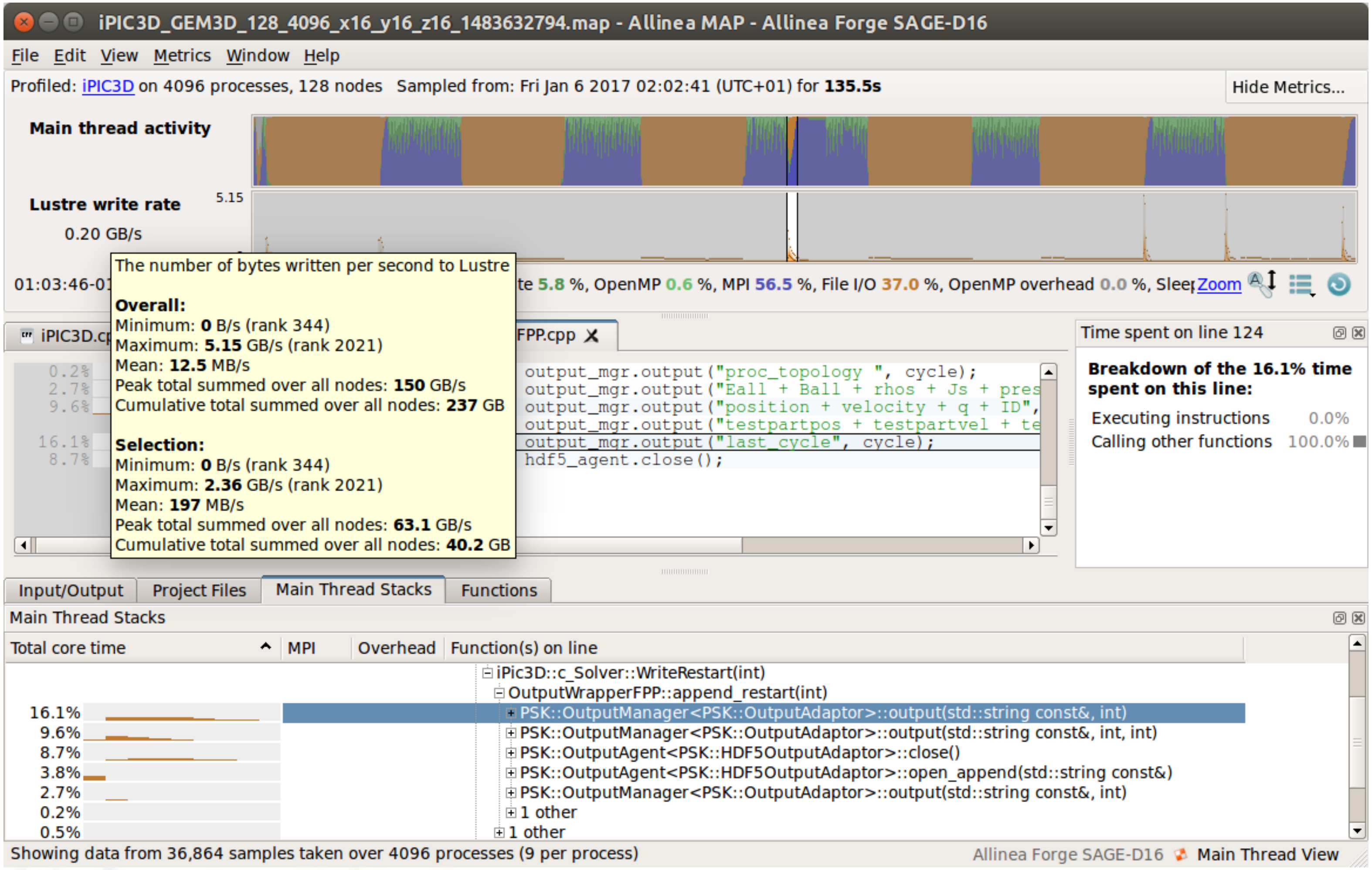
Total core time	MPI	Overhead	Function(s) on line
			iPIC3D [program]
			main
59.7%	<0.1%	<0.1%	iPic3D::c_Solver::WriteOutput(int)
17.1%	11.2%	<0.1%	iPic3D::c_Solver::CalculateField(int)
14.1%	11.0%	<0.1%	iPic3D::c_Solver::ParticlesMover()
6.4%	4.1%	<0.1%	iPic3D::c_Solver::CalculateMoments()
2.7%	1.4%	<0.1%	32 others

Showing data from 4,096,000 samples taken over 4096 processes (1000 per process)

Allinea Forge SAGE-D16 Main Thread View

allinea
Now part of ARM

iPIC3D Profiling: 4096 Processes (128 nodes)



iPIC3D_GEM3D_128_4096_x16_y16_z16_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

01:03:46-01 … te 5.8 %, OpenMP 0.6 %, MPI 56.5 %, File I/O 37.0 %, OpenMP overhead 0.0 %, Sleep Zoom

The number of bytes written per second to Lustre

Overall:
Minimum: **0** B/s (rank 344)
Maximum: **5.15** GB/s (rank 2021)
Mean: **12.5** MB/s
Peak total summed over all nodes: **150** GB/s
Cumulative total summed over all nodes: **237** GB

Selection:
Minimum: **0** B/s (rank 344)
Maximum: **2.36** GB/s (rank 2021)
Mean: **197** MB/s
Peak total summed over all nodes: **63.1** GB/s
Cumulative total summed over all nodes: **40.2** GB

```
 0.2%   output_mgr.output("proc_topology ", cycle);
 2.7%   output_mgr.output("Eall + Ball + rhos + Js + pres
 9.6%   output_mgr.output("position + velocity + q + ID",
        output_mgr.output("testpartpos + testpartvel + te
16.1%   output_mgr.output("last_cycle", cycle);
 8.7%   hdf5_agent.close();
```

Time spent on line 124

Breakdown of the 16.1% time spent on this line:

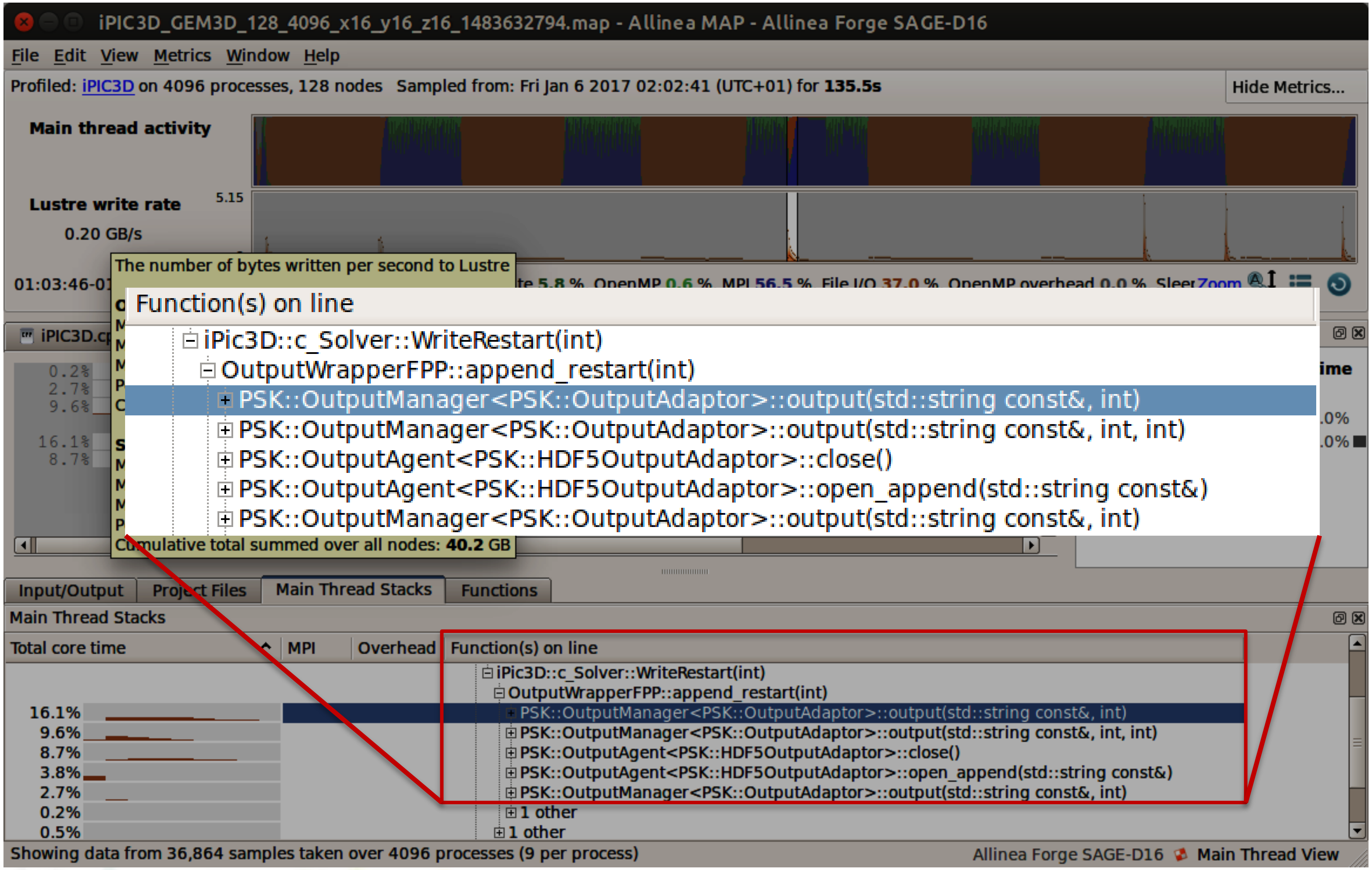
Executing instructions	0.0%
Calling other functions	100.0%

Main Thread Stacks

Total core time	MPI	Overhead	Function(s) on line
			⊟ iPic3D::c_Solver::WriteRestart(int)
			⊟ OutputWrapperFPP::append_restart(int)
16.1%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)
9.6%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int, int)
8.7%			⊞ PSK::OutputAgent<PSK::HDF5OutputAdaptor>::close()
3.8%			⊞ PSK::OutputAgent<PSK::HDF5OutputAdaptor>::open_append(std::string const&)
2.7%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)
0.2%			⊞ 1 other
0.5%			⊞ 1 other

Showing data from 36,864 samples taken over 4096 processes (9 per process) Allinea Forge SAGE-D16 Main Thread View

iPIC3D Profiling: 4096 Processes (128 nodes)



Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

The number of bytes written per second to Lustre

Cumulative total summed over all nodes: **40.2 GB**

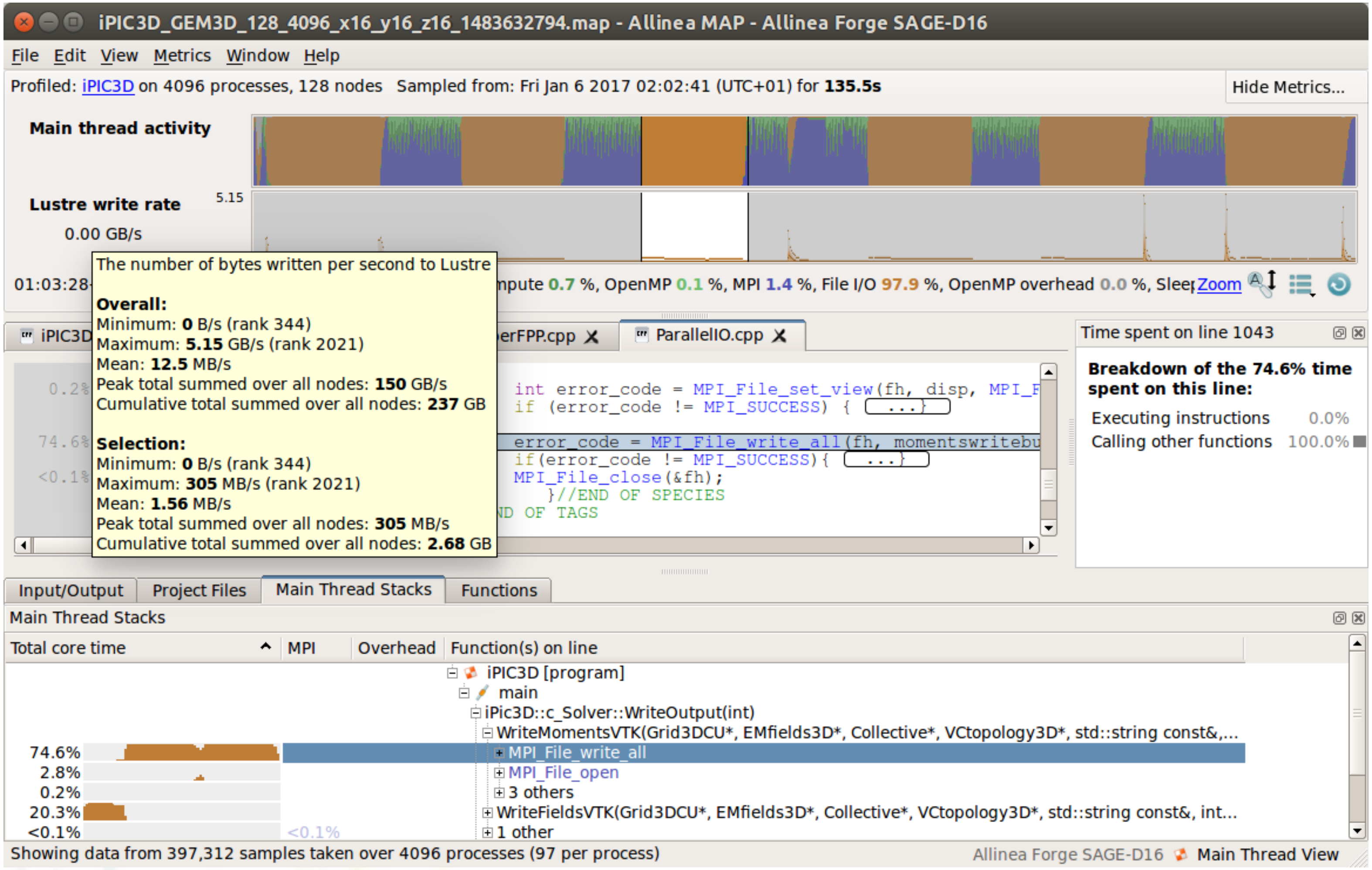
Function(s) on line

- iPic3D::c_Solver::WriteRestart(int)
  - OutputWrapperFPP::append_restart(int)
    - PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)
    - PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int, int)
    - PSK::OutputAgent<PSK::HDF5OutputAdaptor>::close()
    - PSK::OutputAgent<PSK::HDF5OutputAdaptor>::open_append(std::string const&)
    - PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)

Total core time	MPI	Overhead	Function(s) on line
			⊟ iPic3D::c_Solver::WriteRestart(int)
			⊟ OutputWrapperFPP::append_restart(int)
16.1%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)
9.6%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int, int)
8.7%			⊞ PSK::OutputAgent<PSK::HDF5OutputAdaptor>::close()
3.8%			⊞ PSK::OutputAgent<PSK::HDF5OutputAdaptor>::open_append(std::string const&)
2.7%			⊞ PSK::OutputManager<PSK::OutputAdaptor>::output(std::string const&, int)
0.2%			⊞ 1 other
0.5%			⊞ 1 other

Showing data from 36,864 samples taken over 4096 processes (9 per process)

iPIC3D Profiling: 4096 Processes (128 nodes)



iPIC3D_GEM3D_128_4096_x16_y16_z16_1483632794.map - Allinea MAP - Allinea Forge SAGE-D16

Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

01:03:28 ...mpute 0.7 %, OpenMP 0.1 %, MPI 1.4 %, File I/O 97.9 %, OpenMP overhead 0.0 %, Sleep Zoom

The number of bytes written per second to Lustre

Overall:
Minimum: **0** B/s (rank 344)
Maximum: **5.15** GB/s (rank 2021)
Mean: **12.5** MB/s
Peak total summed over all nodes: **150** GB/s
Cumulative total summed over all nodes: **237** GB

Selection:
Minimum: **0** B/s (rank 344)
Maximum: **305** MB/s (rank 2021)
Mean: **1.56** MB/s
Peak total summed over all nodes: **305** MB/s
Cumulative total summed over all nodes: **2.68** GB

```
0.2%   int error_code = MPI_File_set_view(fh, disp, MPI_F
       if (error_code != MPI_SUCCESS) { ...}

74.6%  error_code = MPI_File_write_all(fh, momentswritebu
       if(error_code != MPI_SUCCESS){ ...}
<0.1%  MPI_File_close(&fh);
          }//END OF SPECIES
       ND OF TAGS
```

Time spent on line 1043

Breakdown of the 74.6% time spent on this line:

Executing instructions	0.0%
Calling other functions	100.0%

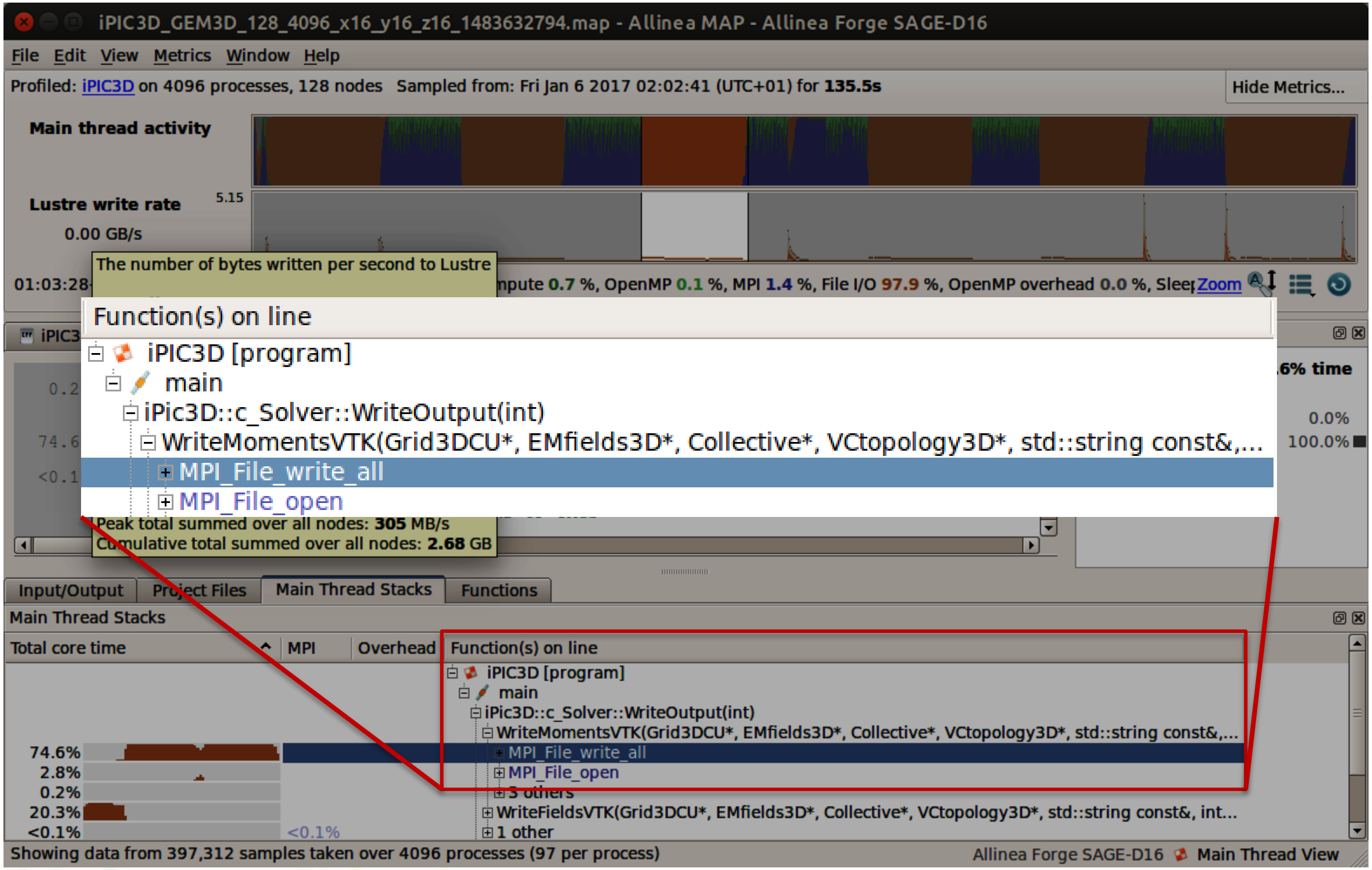
Main Thread Stacks

```
Total core time      MPI     Overhead  Function(s) on line
                                       iPIC3D [program]
                                        main
                                         iPic3D::c_Solver::WriteOutput(int)
                                          WriteMomentsVTK(Grid3DCU*, EMfields3D*, Collective*, VTopology3D*, std::string const&,...
74.6%                                      MPI_File_write_all
 2.8%                                      MPI_File_open
 0.2%                                      3 others
20.3%                                     WriteFieldsVTK(Grid3DCU*, EMfields3D*, Collective*, VTopology3D*, std::string const&, int...
<0.1%                <0.1%                1 other
```

Showing data from 397,312 samples taken over 4096 processes (97 per process)

allinea
Now part of ARM

iPIC3D Profiling: 4096 Processes (128 nodes)



Profiled: iPIC3D on 4096 processes, 128 nodes Sampled from: Fri Jan 6 2017 02:02:41 (UTC+01) for **135.5s**

The number of bytes written per second to Lustre

01:03:28 … mpute 0.7 %, OpenMP 0.1 %, MPI 1.4 %, File I/O 97.9 %, OpenMP overhead 0.0 %, Sleep

Peak total summed over all nodes: **305** MB/s

Cumulative total summed over all nodes: **2.68** GB

```
Function(s) on line
⊟ iPIC3D [program]
  ⊟ main
    ⊟ iPic3D::c_Solver::WriteOutput(int)
      ⊟ WriteMomentsVTK(Grid3DCU*, EMfields3D*, Collective*, VCtopology3D*, std::string const&,...
        ⊞ MPI_File_write_all
        ⊞ MPI_File_open
```

Total core time	MPI	Overhead	Function(s) on line
			⊟ iPIC3D [program]
			⊟ main
			⊟ iPic3D::c_Solver::WriteOutput(int)
			⊟ WriteMomentsVTK(Grid3DCU*, EMfields3D*, Collective*, VCtopology3D*, std::string const&,...
74.6%			⊞ MPI_File_write_all
2.8%			⊞ MPI_File_open
0.2%			⊞ 3 others
20.3%			⊞ WriteFieldsVTK(Grid3DCU*, EMfields3D*, Collective*, VCtopology3D*, std::string const&, int...
<0.1%	<0.1%		⊞ 1 other

Showing data from 397,312 samples taken over 4096 processes (97 per process)

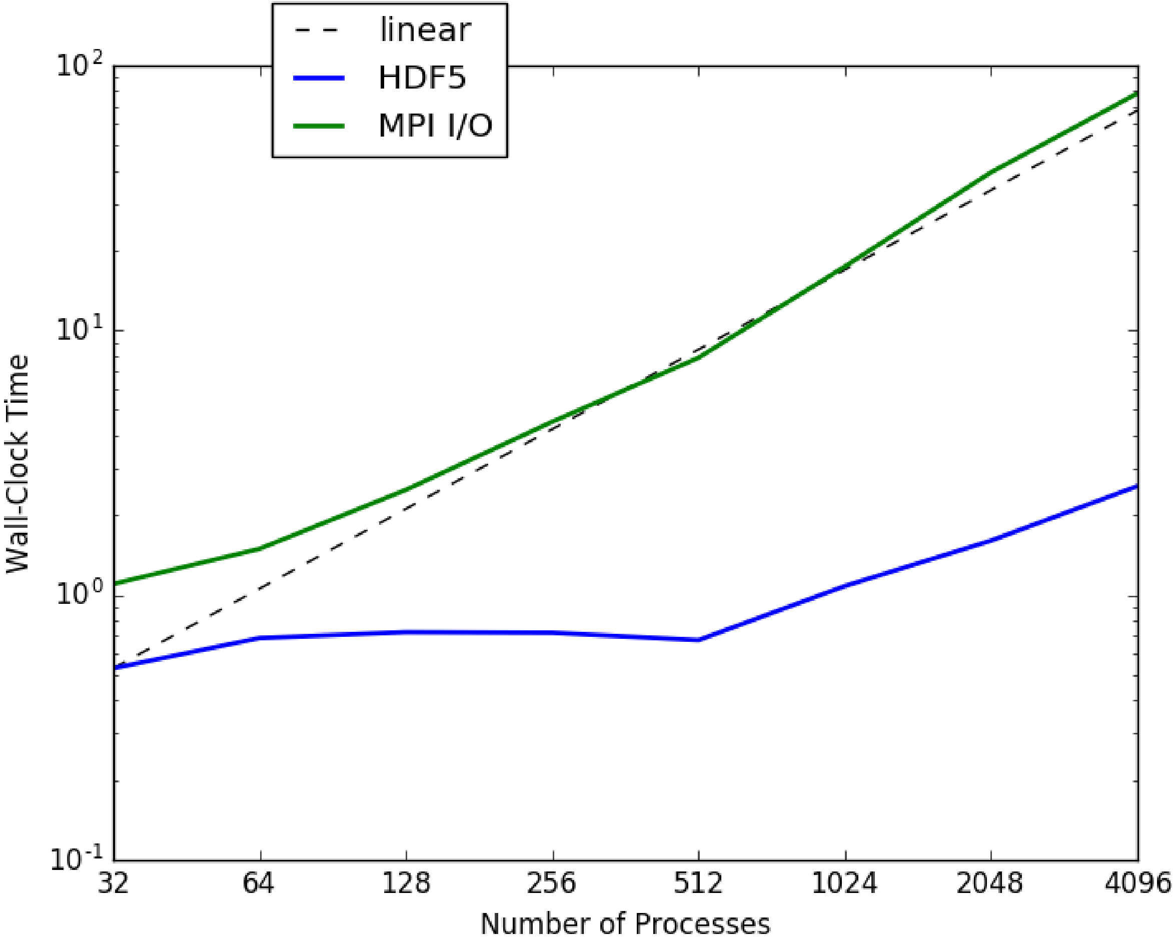
iPIC3D Profiling

- HDF5 and MPI identified as I/O libraries for fast and slow phases of I/O, respectively

- Allinea MAP can show percentage time spent in different functions

- Calculate wallclock time spent in MPI I/O and HDF5 I/O and plot over weak scaling runs

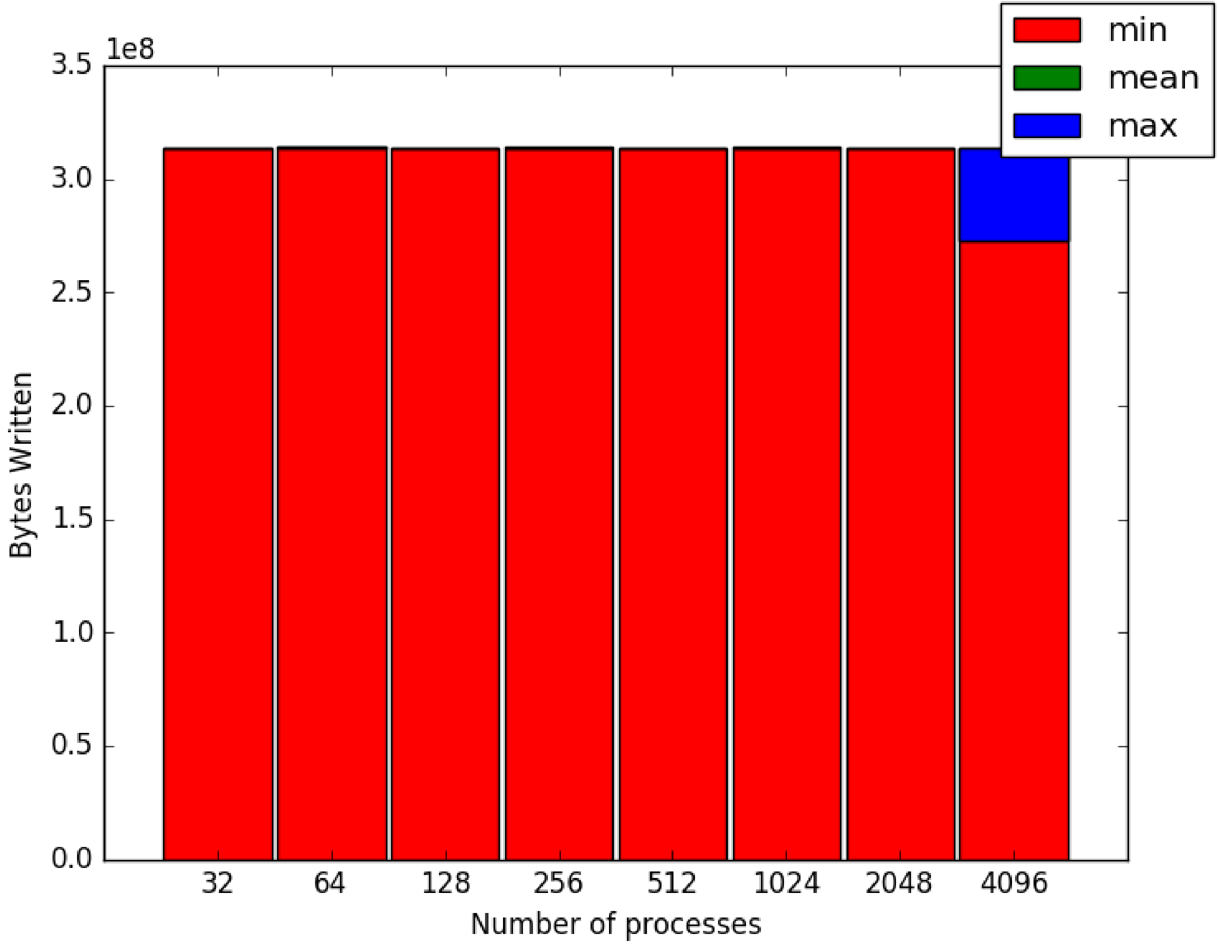
iPIC3D I/O time



iPIC3D Profiling

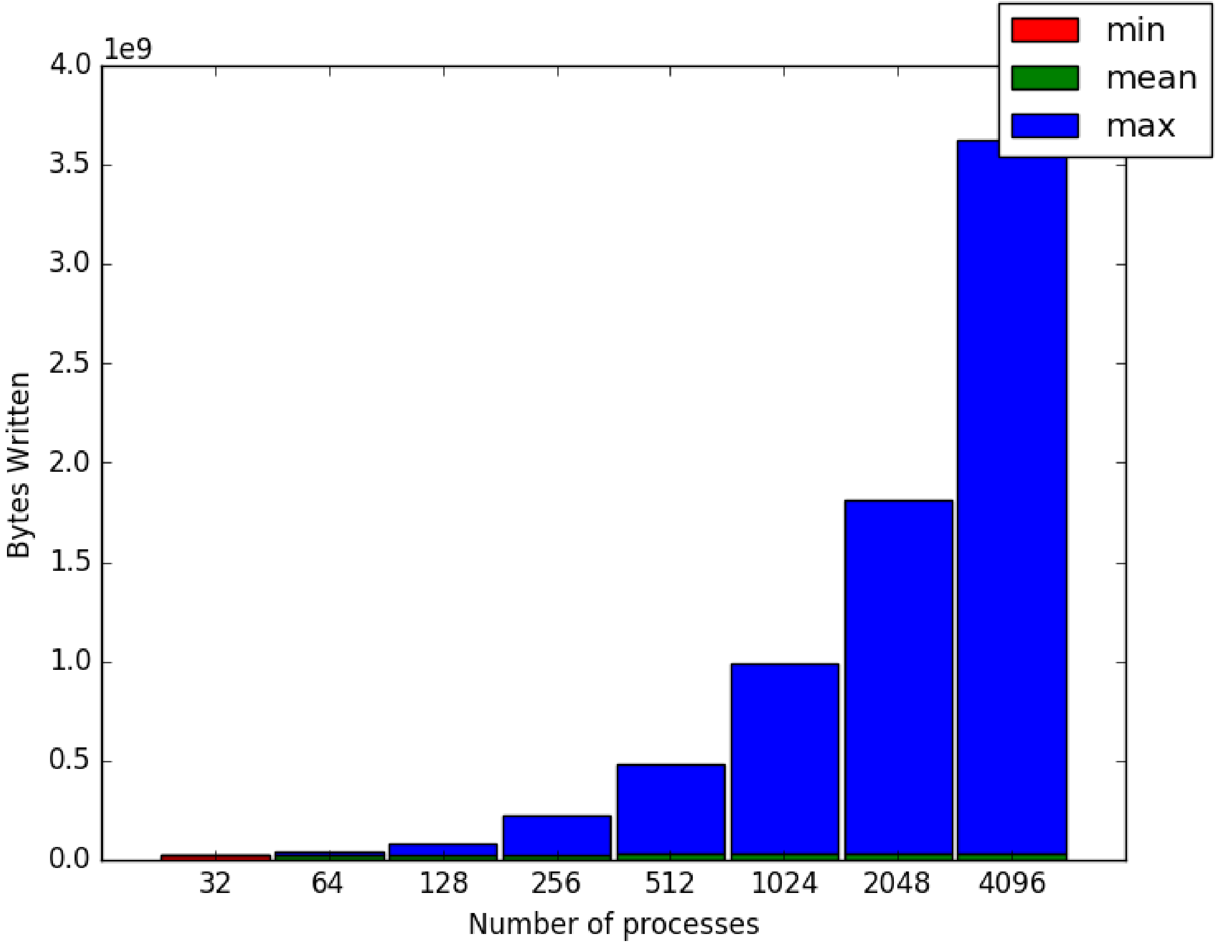
- MPI I/O performs collective write from all processes. 512kB are written per process. Scaling of time is linear with amount of data to write
- HDF5 performs small file write from all processes (10MB per process)
- How much data is written from each node?

iPIC3D I/O Volume – HDF5



allinea
Now part of ARM

iPIC3D I/O Volume – MPI I/O



iPIC3D I/O

- Cray collective MPI I/O accumulates data to a single node

- Possible improvements
  - Dedicated I/O nodes
  - Asynchronous I/O (e.g. burst buffers, function offloading - SAGE)
  - Limiting number of writers per file

I/O Areas of Interest

- Ideal Performance – is it achievable or desirable to achieve for real applications?
  - How sensitive to system performance would an application be which achieves the maximum I/O throughput?
  - Can I/O bandwidth be maximised in an application through better I/O management?

I/O Areas of Interest

- System I/O – how does application I/O tie in with the system?
  - View application I/O alongside system I/O
  - What about I/O in a group of related programs (i.e. workflows)?
  - Should I/O bandwidth be maximised at a system or workflow level rather than at an application level?

Last Words - Profiling

- Profiling at small scales may not show the whole picture – measurement at larger scales show problems related to scale

- MAP provides low overhead measurement with rich information

- Export to JSON of performance data allows for post-processing and analysis across many runs