Introduction

This document will guide you through the process of using Open|SpeedShop on HPC Wales.

What is Open|SpeedShop?

Open|SpeedShop (OSS) is a profiling tool built on top of a number of open-source applications (such as PAPI and Vampirtrace) that can be used to gather performance data about serial and parallel codes.

Further details on accessing HPC Wales systems can be found in the User Guide and help can be obtained through the Support Desk:

- E-mail support@hpcwales.co.uk
- Telephone 08452 572 207
- Support Website https://hpcwprod.service-now.com/

Table of Contents

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Preliminaries

OSS collects performance data from your code as it executes and to be able to store this data it needs a location visible to all ranks in the executing job. To do this, you need to create a directory in `/home/user.name` called `oss` and within this directory create the subdirectory `raw`. You will then need to set the appropriate environment variable to tell OSS to use it:

```
[user.name@cf-log-001 ~]$ export OPENSS_RAWDATA_DIR=/home/user.name/oss/raw
```

You might like to put this definition in your `.myenv` file to ensure that it is loaded automatically each time you log on.

Gathering Performance Data With OSS

Before starting OSS you must ensure that you have loaded the necessary modules:

```
[user.name@cf-log-001 ~]$ module load mpi compiler openss/2.0.2
```

You do not need to recompile your code to collect data with OSS.

You can collect a variety of different types of performance data from your code with OSS by using the various experiments that it offers. The following table lists the basic types of experiment on offer; see the OSS User Guide for the full list:

Name	Default Data Collected
`usertime`	CPU time for each function call (including and excluding child functions) based on sampling the call-stack 35 times per second.
`pcsamp`	CPU time for each function call (including and excluding child functions) based on sampling the program counter 100 times per second.
`mpi,mpit, mpiotf`	Wall clock time spent in MPI functions, aggregated across all functions with the same name across all ranks.
`io, iot`	Wall clock time spent in the IO system calls `read`, `write`, `writev`, `open`, `close`, `dup`, `pipe` and `creat`
`fpe`	List of encountered floating point exceptions.

The hardware counter (`hwc`, `hwcsamp` and `hwctime`) experiments are not currently available on HPC Wales.

You invoke OSS from within a job submission script using the syntax:

```
openss -f  <executable> <experiment>
```

where `<executable>` is the command you would normally use to execute your code (including `mpirun` where appropriate) and `<experiment>` is the name of one of the experiments from the table above.

Gathering Performance Data With OSS

For example, to conduct a profiling experiment on an MPI-based executable named source across 24 cores you would use the following job submission script:

```
#!/bin/bash --login
#BSUB -o source.o%J
#BSUB -x
#BSUB -n 24
#BSUB -W 00:30

module purge

module load mpi compiler openss/2.0.2

openss -f "mpirun -n 24 ./source" mpi
```

OSS will display a summary of the collected data at the end of your job's output. For example, for the submission script above we might see:

```
Maximum MPI  Minimum MPI      Average  Number  Function (defining location)
       Call         Call     Time(ms)      of
   Time(ms)     Time(ms)                Calls
1516.185000  1500.532000  1509.050333      12  PMPI_Init (libmonitor.so.0.0.0:
pmpi.c,103)
 214.636000     0.004000     0.032580  188556  MPI_Allreduce (libmpi.so.4.0)
 190.989000     0.000001     0.007885   94296  MPI_Waitall (libmpi.so.4.0)
  68.810000    68.742000    68.776833      12  PMPI_Finalize
(libmonitor.so.0.0.0: pmpi.c,232)
   0.197000     0.000001     0.000439  180987  MPI_Irecv (libmpi.so.4.0)
   0.135000     0.014000     0.111583      24  MPI_Barrier (libmpi.so.4.0)
   0.127000     0.001000     0.017893      84  MPI_Reduce (libmpi.so.4.0)
   0.011000     0.000001     0.000461  180987  MPI_Send (libmpi.so.4.0)
```

Multiplying the average call duration by the number of calls (and ignoring the calls to `PMPI_Init` and `PMPI_Finalize`) we can see that the MPI call that the code spends the longest total time in is `MPI_Allreduce`.

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales:
Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Viewing Collected Data With OSS's GUI

Much more profiling information about your code can be accessed via OSS's GUI once execution is complete. In order to use the GUI you need to ensure that you have X11 forwarding over `ssh` enabled. To do this, create a file named `config` in your `.ssh` directory and put the following text in it (replacing `user.name` with your HPC Wales username):

```
host ??-log-???
 User user.name
 ProxyCommand ssh user.name@login.hpcwales.co.uk nc %h %p 2> /dev/null
```

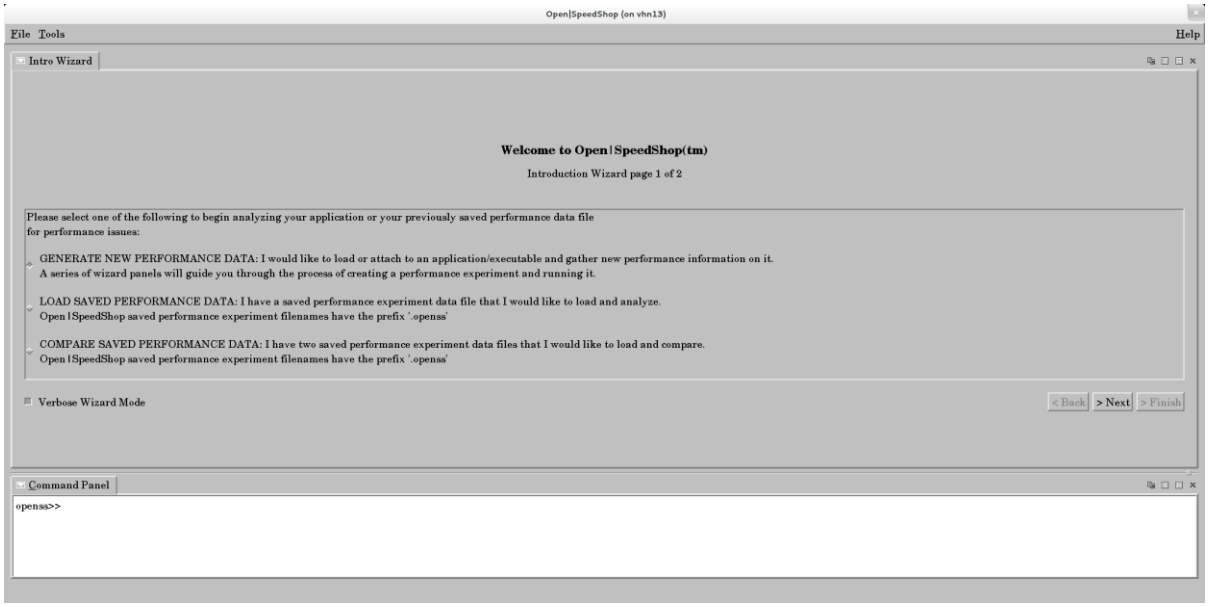
You then log into an HPC Wales login node with:

```
[localhost ~]$ ssh -X cf-log-001
```

Note that you will need to enter your HPC Wales password twice. You can now launch the OSS GUI from the command line:

```
[user.name@cf-log-001 ~]$ openss
```

A start screen will appear that offers you a number of options:



Select "LOAD SAVED PERFORMANCE DATA", click "Next" and then select the output file from the file selection dialogue that appears. If you know which file you want to view, you can skip the process of selecting it from the GUI and invoke OSS directly with:

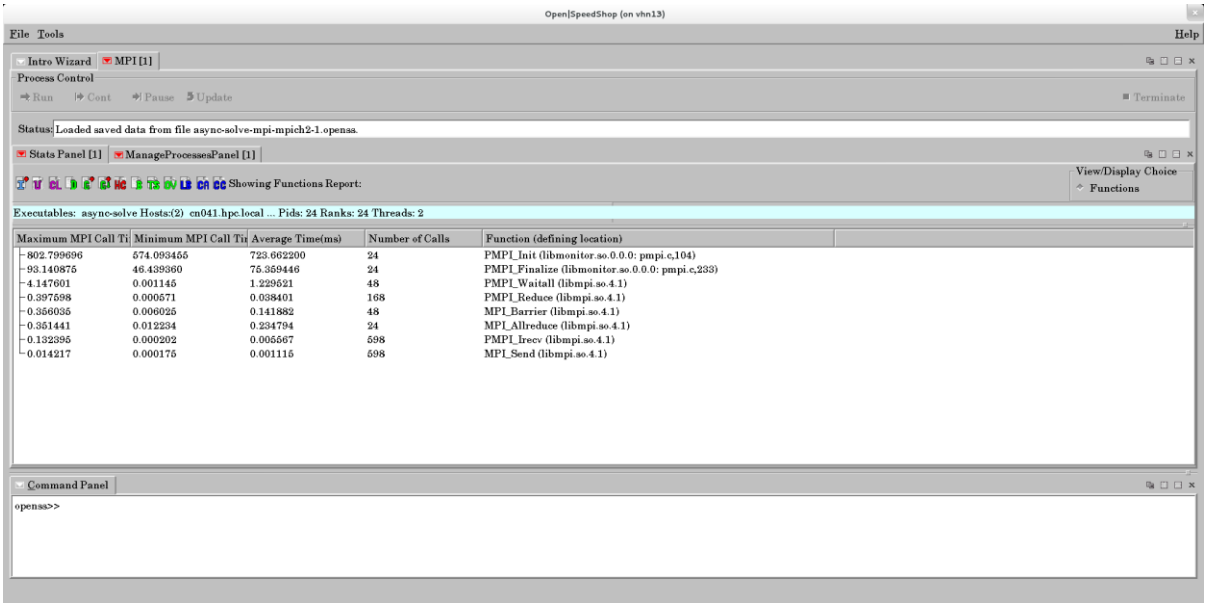
```
[user.name@cf-log-001 ~]$ openss -f <filename>
```

where `<filename>` will have the format `<executable>-<experiment>.openss`.

The GUI will then display the default view for the experiment type that the data was gathered from. For example, opening data from an `mpi` experiment will give:

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales:
Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

Viewing Collected Data With OSS's GUI



The coloured letters above the main data display allow you to access different views of the collected data.

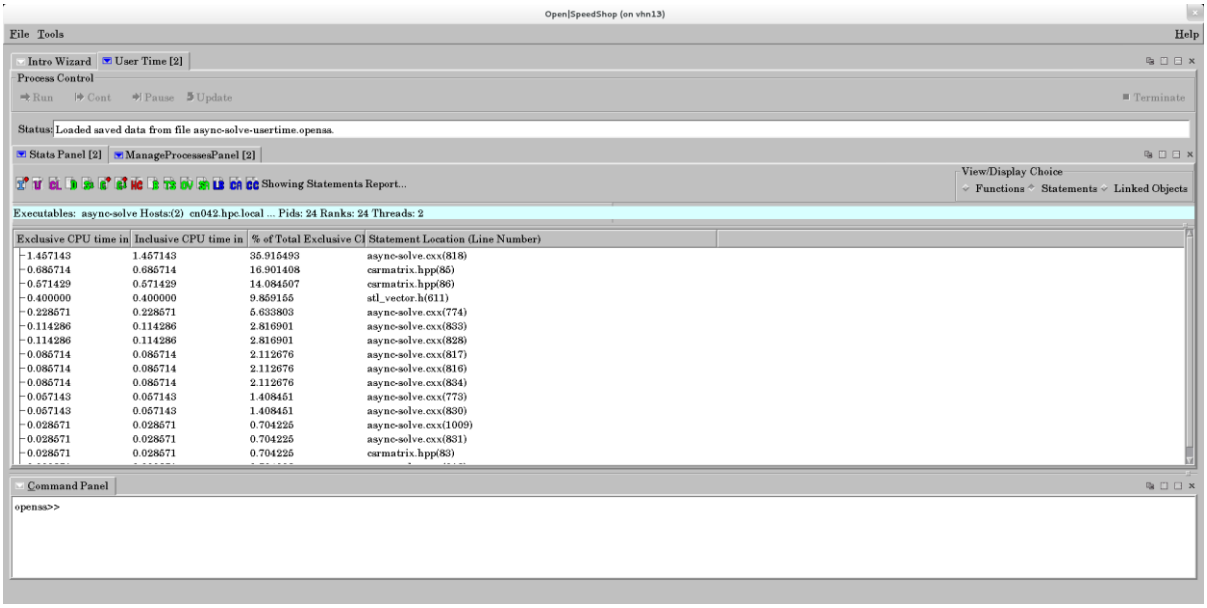
Button	Name	Description
I	Information	Show information about the displayed experiment, including type, executable name and MPI ranks.
U	Update	Update the main display with newly-selected viewing options.
CL	Clear Auxiliary Information	Clear any selected time segments or specific rank/process/thread/function views and restore original display settings.
D	Default	Restore the default display.
C+	Call Path Full Stacks	Show all call paths in the code, including duplicates.
C+↓	Call Path Full Stacks Per Function	Show all call paths in the code for a specific function, including duplicates.
HC	Hot Call Path	Show the 5 call paths in the code that took the longest time.
B	Butterfly View	Show the callers and callees of a specific function
T	Time Segment Selection	Display performance results taken from a subset of the total time of the experiment.
OV	Optional View Selection	Create a new view of the performance data with a selection of optional fields/columns.
LB	Load Balance View	For a multi-process experiment, show which ranks experienced the maximum/minimum values (e.g. of time spent in a function).
CA	Compare and Analyse View	For a multi-process experiment, use cluster analysis to group together similarly-performing ranks.
CC	Custom Comparison View	Compare user-specified performance metrics across threads, ranks or experiments.

Ewrop & Chymru: Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales: Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government

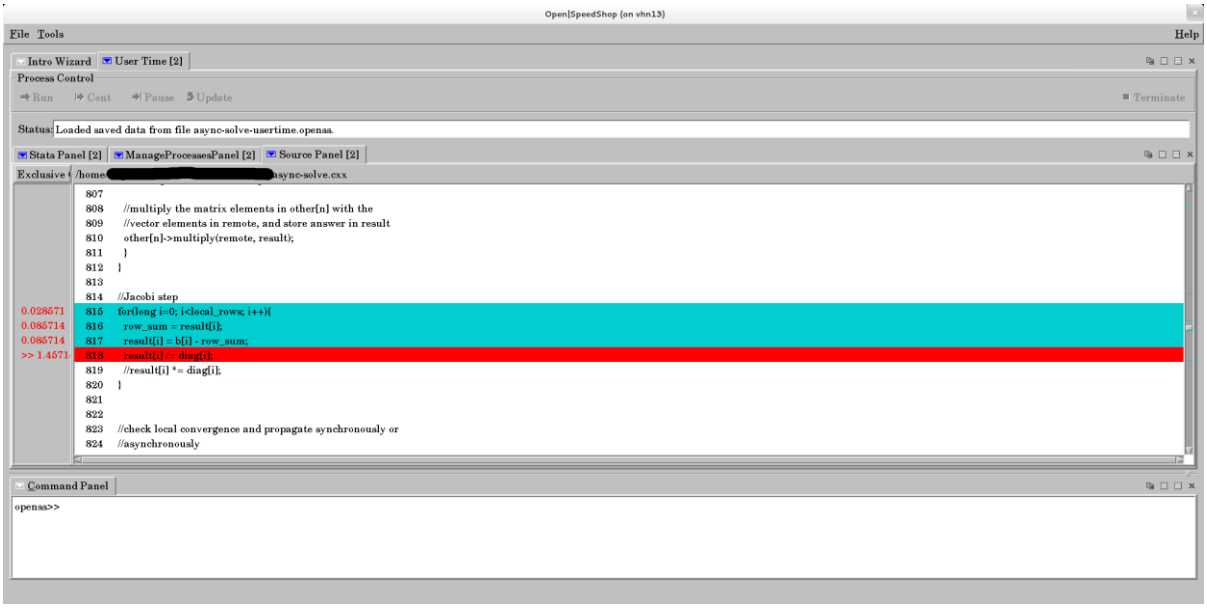
Viewing Collected Data With OSS's GUI

Other buttons may be displayed for different experiment types. If you place your mouse pointer over them, a tool tip will appear that describes what they do.

Other types of experiment allow you to view data at the level of the individual source code lines, rather than the function level displayed above. For example, opening the results of a `usertime` experiment and selecting "Statements" from the "View/Display Choice" options to the top right of the main display area will give this:



You can then double-click on one of the entries in the main display window to be taken to the corresponding line in the source code:



Here, we selected the top line on the default display, and in the resulting source code display this line (818) is highlighted in red as it was the line on which the code spent the highest proportion of time.

Online Documentation

Both the full OSS user guide and a handy quick start guide are available online.

Ewrop & Chymru:
Buddsoddi yn eich dyfodol
Cronfa Datblygu Rhanbarthol Ewrop
Europe & Wales:
Investing in your future
European Regional Development Fund
ERDF
Llywodraeth Cymru
Welsh Government