ScoreP Tutorial

This document contains information about installing ScoreP on HiPerGator and analyzing it with Scalasca.

ScoreP installation –

Here, we install ScoreP v7.1 which has the following pre-requisites:

- Intel/2019.1.144
- Openmpi/4.0.3

Open your HiPerGator terminal and run the following command:

module load intel/2019.1.144 openmpi/4.0.3

Download ScoreP v7.1 tar file from the official website: https://www.vi-hps.org/projects/score-p/ or use this direct download link: https://perftools.pages.jsc.fz-juelich.de/cicd/scorep/tags/scorep-7.1/scorep-7.1.tar.gz

Copy the tar file to a suitable directory on HiPerGator and unzip it using the following command:

tar zxf Scorep-7.1.tar.gz (modify according to the name of your tar file).

Enter into the unzipped folder (**cd Scorep-7.1**) and run the following commands:

- **mkdir _build**
- **cd _build**
- **../configure --prefix=$HOME/install/scorep --enable-static --disable-shared --with-nocross-compiler-suite=intel --with-mpi=openmpi --with-papi-header=$PAPI_BASE/include --with-papi-lib=$PAPI_BASE/lib**

This will run a system check to make sure all the necessary modules are loaded and available (required time ~15 minutes).

- **make && make install** (~15 minutes)

This command will install ScoreP module in the –prefix path provided in the configure command ($HOME/install/scorep).

Scalasca installation –

Scalasca is a tool used to analyze the ScoreP generated output. Here we use Scalasca v2.6 that can be directly downloaded from this link: https://zenodo.org/record/4700519/files/scalasca-2.6.tar.gz?download=1

Copy the tar file to a suitable directory on HiPerGator and unzip it using the following command:

tar zxf scalasca-2.6.tar.gz (modify according to the name of your tar file).

Enter into the unzipped folder (**cd scalasca-2.6**) and run the following commands:

- **mkdir _build**
- **cd _build**

- **../configure --prefix=$HOME/install/scalasca --enable-static --disable-shared --with-nocross-compiler-suite=intel --with-mpi=openmpi --with-papi-header=$PAPI_BASE/include --with-papi-lib=$PAPI_BASE/lib**

This will run a system check to make sure all the necessary modules are loaded and available (required time ~15 minutes).

- **make && make install** (~15 minutes)

This command will install Scalasca module in the –prefix path provided in the configure command ($HOME/install/scalasca).

Adding modules installed to the PATH –

To add the ScoreP and Scalasca module to the PATH, i.e., for the system to recognize the modules in their environment, follow these steps:

- Open the .bashrc file located in the root directory using the command: **nano ~/.bashrc**
- Add this line to the file:
    - **export PATH=$PATH:~/install/scorep/bin:~/install/scalasca/bin**

Note: If you changed the --prefix path in the configure command in the previous sections, update the path in the above line accordingly.

This step ensures that upon startup, ScoreP and Scalasca will be added to the PATH. After this step, the .bashrc file should look like this for example –

```
  GNU nano 2.3.1                                  File: /home/smandavilli/.bashrc

# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions

alias ll='ls -al --color=auto'

export PATH=$PATH:~/install/scorep/bin:~/install/scalasca/bin
```

- Restart the current session (exit and open a new terminal).
- Verify if the modules were installed. I check for installed versions as shown below.

```
[smandavilli@login6 ~]$ scorep --version
Score-P 7.1
[smandavilli@login6 ~]$ scalasca --version
2.6
[smandavilli@login6 ~]$
```

Compiling application with ScoreP –

To compile an application with ScoreP, append "**scorep**" to your command for building the executable file.

For example –

mpicc foo.c -o foo

becomes

scorep mpicc foo.c -o foo

Then append "**scalasca -analyze**" to your command for running the executable.

For example, the modified SLURM script to run an MPI application looks as follows:

```
  GNU nano 2.3.1                                    File: job-hw_mpi_omp.sh

#!/bin/bash
#SBATCH --account=eel6763
#SBATCH --qos=eel6763
#SBATCH --nodes=4
#SBATCH --ntasks=8
#SBATCH --ntasks-per-node=2
#SBATCH --cpus-per-task=2
#SBATCH --mem-per-cpu=500mb
#SBATCH -t 00:10:00
#SBATCH -o outfile
#SBATCH -e errfile
export OMP_NUM_THREADS=4
scalasca -analyze srun --mpi=pmix_v3 ./lulesh2.0 -s 4
```

This will create a scorep_*sum folder in the same path where you run the application. Scalasca also creates a scorep_*sum.log file which can be used for debugging.

After execution finishes, to analyze this profiled data, run the command:

scalasca -examine -s scorep_*sum (replace the name of your generated scorep_*sum folder accordlingly).

This will create a **scorep.score** profile in the scorep_*sum folder. View this file (using cat command or open in any text editor). It should look like this for example –

```
[smandavilli@login6 LULESH_hybrid]$ cat scorep_lulesh2_8x4_sum/scorep.score

Estimated aggregate size of event trace:                   1977MB
Estimated requirements for largest trace buffer (max_buf): 273MB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       281MB
(hint: When tracing set SCOREP_TOTAL_MEMORY=281MB to avoid intermediate flushes
 or reduce requirements using USR regions filters.)

flt     type  max_buf[B]     visits time[s] time[%] time/visit[us]  region
         ALL 286,084,642 73,669,432  172.39   100.0           2.34  ALL
         USR 237,327,948 64,669,246    5.02     2.9           0.08  USR
         OMP  47,741,396  8,644,216   87.34    50.7          10.10  OMP
         COM     945,230    285,950    2.43     1.4           8.50  COM
         MPI     527,405     70,012   77.60    45.0        1108.41  MPI
      SCOREP          41          8    0.00     0.0          45.82  SCOREP

         USR  92,487,850 25,720,713    1.11     0.6           0.04  std::vector<double, std::allocator<double> >::operator[]
         USR  14,640,392  2,096,658    0.16     0.1           0.08  FABS
         USR   9,611,394  2,957,352    0.29     0.2           0.10  Domain::x
         USR   9,611,394  2,957,352    0.28     0.2           0.09  Domain::y
         USR   9,611,394  2,957,352    0.27     0.2           0.09  Domain::z
         USR   8,487,726  2,611,608    0.27     0.2           0.10  Domain::xd
         USR   8,487,726  2,611,608    0.26     0.2           0.10  Domain::yd
         USR   8,487,726  2,611,608    0.25     0.1           0.10  Domain::zd
         OMP   5,956,020    547,680    0.07     0.0           0.13  !$omp parallel @lulesh.cc:2022
         OMP   5,956,020    547,680    0.07     0.0           0.13  !$omp parallel @lulesh.cc:2029
         USR   4,103,944    710,708    0.05     0.0           0.07  SQRT
         USR   3,393,208    994,864    0.04     0.0           0.04  std::vector<int, std::allocator<int> >::operator[]
         USR   3,133,546    506,627    0.05     0.0           0.11  Domain::q
         USR   2,890,316    889,328    0.10     0.1           0.12  Domain::fx
         USR   2,890,316    889,328    0.08     0.0           0.10  Domain::fy
         USR   2,890,316    889,328    0.08     0.0           0.09  Domain::fz
         USR   2,862,314    423,171    0.05     0.0           0.12  Domain::p
         USR   2,591,602    339,735    0.04     0.0           0.12  Domain::e
         USR   2,589,418    339,203    0.04     0.0           0.11  Domain::delv
         USR   2,589,418    339,203    0.04     0.0           0.12  Domain::qq
         USR   2,589,418    339,203    0.04     0.0           0.12  Domain::ql
         USR   2,169,856    667,648    0.03     0.0           0.05  _INTERNAL8615d6bf::VoluDer
```

Note: The scorep*_sum folder also contains a .cubex file which can be used by the Cube tool.

Scorep Guide: https://scorepci.pages.jsc.fz-juelich.de/scorep-pipelines/docs/scorep-4.1/html/quickstart.html