Application Performance Snapshot (APS) Playbook

==

The Playbook contains command lines starting with $

Please change $PRG, $ARGS into the path,name and parameters of your program!

Version 1.0, 18.02.2019

0. Environment

----

load modules

```
$ module load Intel
$ module load IntelMPI
$ module load VTune
```

For Batch jobs

==============

Please include: --disable-perfparanoid

in command line for sbatch or in scripts with #SBATCH --disable-perfparanoid

check for important executables

```
$ which aps
```

check version

```
$ aps –version
```

1.0 Application Performance Snapshot (APS) usage:

===

Please contain in your SLURM batch jobs:

```
--disable-perfparanoid
```

this will change the perf_event_paranoid to 0. With values > 0 you will get less information (1) and no information for (>2)

Just put aps in front of the executable (non MPI):

```
$ aps $PRG $ARGS
```

For MPI programs:

```
$ srun <srun parameter> aps -r aps_out $PRG $ARGS
```

Poisson Example:

```
srun -n <N> aps -r aps_out ./poisson.x -n 3200
```

Output directory name "-r aps_out" may be omitted or chosen differently

Note: check with $ ls -ltr

for the last created directory.

please choose another directory name if you do another run in thesame directory.

2.0 View Results

================

Summary results can be displayed by

```
$ aps-report aps_out -s
```

result-dir is starting with "aps_" if result dir is not specified.

HTML results

```
$ aps-report -g aps_out
```

generates aps_report_<date>_<time>.html

more MPI statistics (functions) are available.

```
$ aps-report aps_out -f
```

or for the full output:

```
$ aps-report aps_out -a
```

For more detailed MPI output the program has to run under the environment variable:

```
$ export MPS_STAT_LEVEL=4
```

The integer value may be 2 to 4 for even more infos. Higher values will cause more overhead.

Display rank to rank matrix with communication times:

```
$ aps-report -x --format=html aps_out
```

3.0 Code block for analysis may be selected

Insert MPI_Pcontrol(0) right after MPI_Init() to switch off tracing

Insert MPI_Pcontrol(1) before code block   to switch on tracing

Insert MPI_Pcontrol(0) after code block   to switch off tracing

see: https://software.intel.com/en-us/get-started-with-application-performance-snapshot

MPI_Pcontrol will be applied only on the MPI part. For limiting the HW counters use the _itt

library found on web page above

4.0 Jube usage recently untested (please ask instructors)

==

Jube is developed by FZ Juelich (Juelich universal benchmark environment)

http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/JUBE/_node.html

poisson.xml is running poisson.x under Jube. For your own program change poisson.x and parameter.

Additional include files for APS and VTune are available

```
$ cd Poisson_1.3
$ module load JUBE
$ export JUBE_INCLUDE_PATH=$PWD/JUBE_INCLUDE:$JUBE_INCLUDE_PATH
```

4.1 run jube

============

run without tools

```
$ jube run poisson.xml -a -r
```

more output

```
$ jube run poisson.xml -a -r --tag long
```

with aps support

```
$ jube run poisson.xml -a -r --tag aps
```

Alternative Environment (TBD)

Alternative PSXE 2019

```
$ module load Intel
$ source <path to 2019>

$ which aps
```