Memory Usage Tracking on HPC resources.

M. Oliveira[1]

[1]Francis Crick Institute, London

Abstract. Determining the correct amount of resources a given HPC workflow requires is, in most instances, not a trivial process. This is specially acute for the case of memory. In this paper we describe a very simple application, based on a forking model, to automatically monitor memory utilisation on these environments.

Keywords: HPC, MPI, Memory Usage.

1 Introduction

Any HPC workflow invariably starts by writing a job submission scrip that codes the workflow and, most importantly, declares the amount of resources required. These resources vary in all HPC systems but they always include the number of cores to be used, the amount of time needed and the amount of max amount of memory required. Of all these, the amount of memory is usually the most difficult to estimate. And even when an estimation is possible describing is usage throughout the duration of the job, which may have impact in any future debugging, is usually much harder.
In this situations it would desirable if these HPC systems could give the users an automatic tool to monitor memory usage throughout the duration of the job.
This paper details the solution developed by us, as a very simple forking code, for this situation.

2 The forking model for the memory usage code

The model for the code is very simple and was adapted from a code readily available online. The code forks the original code capturing its PID. It subsequently enters a loop that monitors /proc/PID/status for the values of vmsize, vmpeak, vmrss and vmhwm and write it to output memusage file.
An MPI implementation was also produced. The top code is now a simple parallel launcher using the MPI_Comm_spawn directive to independently spawn the memory usage code that in turn forks the original parallel code.

3 Monitoring serial jobs

Using the memory usage tracking code on a serial code is very easy and corresponds to writing a script equivalent to:

```
#!/bin/bash
# SLURM sbatch memusage example - serial
#SBATCH --job-name=serial
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=128
#SBATCH --partition=compute

memusage <app_name> <app_params>
```

4 Monitoring OpenMP jobs

OpenMP threaded codes can also have their memory usage tracked with:

```
#!/bin/bash
# SLURM sbatch memusage example - OpenMP
#SBATCH --job-name=openmp
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=2
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=128
#SBATCH --partition=compute

memusage <app_name> <app_params>
```

5 Monitoring MPI jobs

For MPI parallel codes the parallel version of the code is required:

```
#!/bin/bash
# SLURM sbatch memusage example - MPI
#SBATCH --job-name=parallel
#SBATCH --ntasks=2
#SBATCH --cpus-per-task=1
#SBATCH --time=5:00
#SBATCH --mem-per-cpu=128
#SBATCH --partition=compute

mpimemusage <app_name> <app_params>
```

6 Plotting the results

A very simple Gnuplot code can be used to plot all the information stored on the memusage files by using:

```
module load memuse gnuplot
gnuplot <path/to/plot/utility/>plot
```

Typical outputs for serial runs and parallel runs can be seem on Figures 1 and 2.

7 Conclusions

A very simple memory usage tracking application was presented here to allow users to track and understand its utilisation. On any HPC system this very simple resource allows a vast number of users to best familiarise themselves with the memory footprint of their workflows and have a better understanding of its utilisation throughout its lifetime.

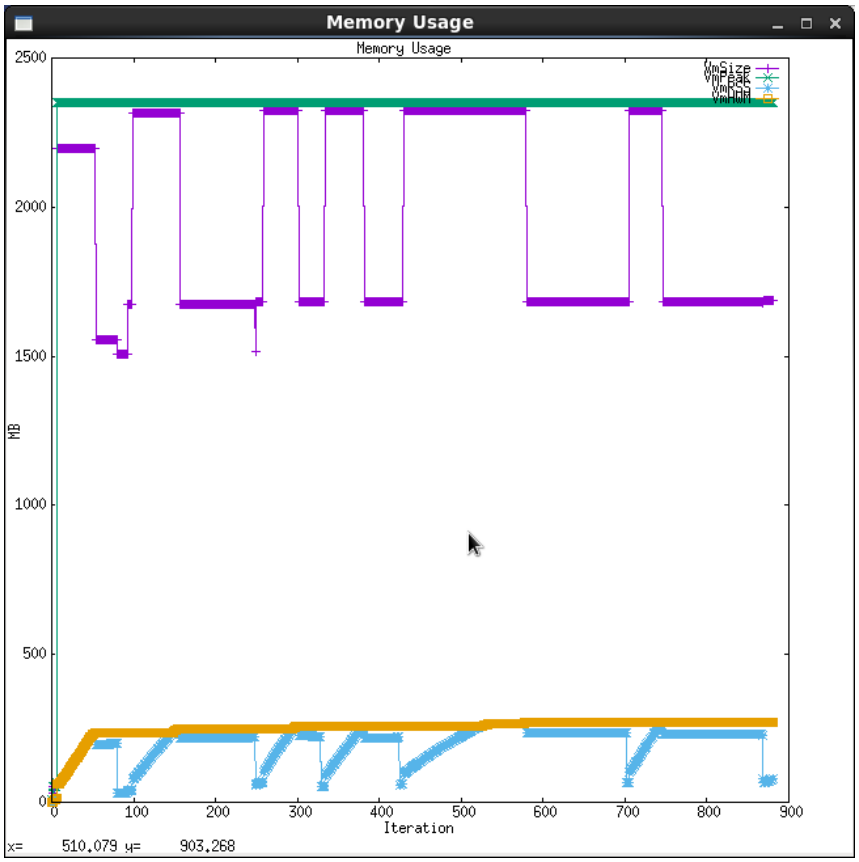

Fig. 1. Example of a memory usage tracking plot for a serial job

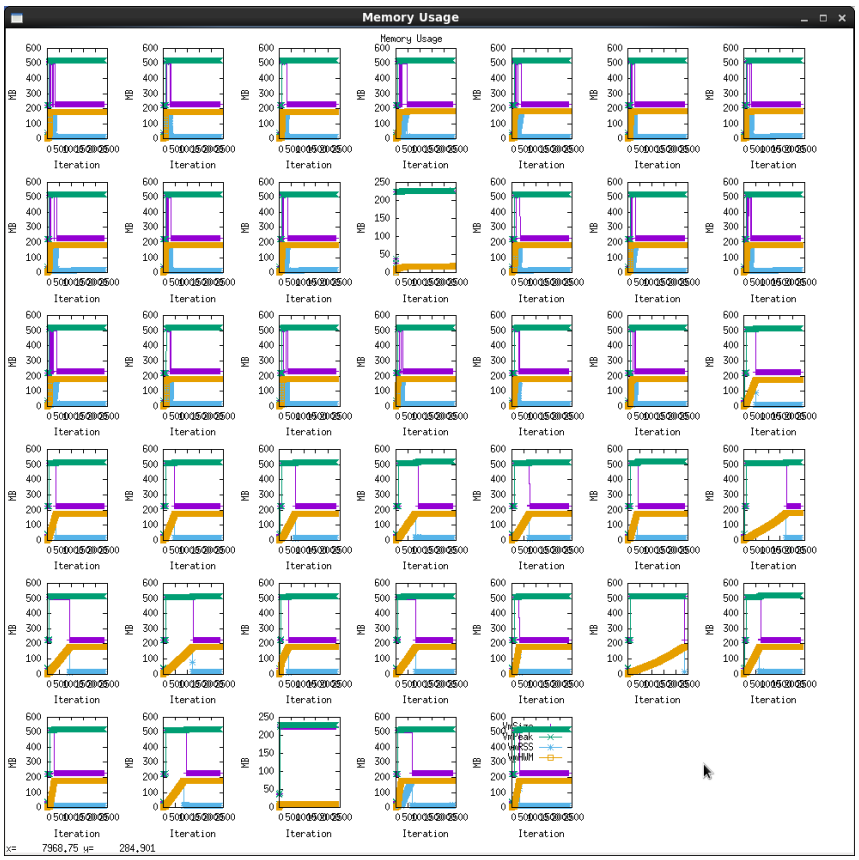

Fig. 2. Example of a memory usage tracking plot for a parallel job