University of Amsterdam
BSc Informatica

Distributed and Parallel Programming

DAS5 instructions

November 2022

1 Introduction

For the practical assignments you will be working on the DAS5. During the first practical session, you will receive one account per team. The DAS5 consists of several clusters. For example, DAS5/VU and DAS5/UvA. You can use any of the "first" five clusters - fs0 to fs4 - unless otherwise specified. For more details about the accounts, the clusters, and the DAS5 architecture, please check `https://www.cs.vu.nl/das5/accounts.shtml` and `https://www.cs.vu.nl/das5/clusters.shtml`.

2 Accessing and using DAS5

In this section, we discuss the basic commands to allow you to work on DAS5.

2.1 UvA VPN

When working on the assignments from home, you can not directly connect to the DAS5. For installation and usage instructions, please see the Datanose Linux wiki: `https://linux.datanose.nl/linux/vpn/`

2.2 Access and data copying

To access the cluster, you have to use SSH:

```
$ ssh -X USERNAME@fs0.das5.cs.vu.nl
```

USERNAME is the user account you received. The -X flag is optional, but enables X11-forwarding. This way, you can open graphical applications through ssh. To transfer files from your own machine to DAS5, you can use scp.

```
$ scp file USERNAME@fs0.das5.cs.vu.nl:
```

To synchronize files you can use rsync. rsync is a file synchronization program that minimizes network data transfer. To synchronize folder **A** on your machine with folder **A** on DAS you can run:

```
$ rsync -avz -ssh /home/$USER/A USERNAME@fs0.das5.cs.vu.nl:/home/$USER
```

You may be aware of another possibility called sshfs to mount folders from the das on your computer. You are prohibited from using this as it places undue stress on the headnode. We recommend writting a small script that syncs your code for you.

2.3 Running jobs

Logging on one of the DAS5 clusters "lands" you on the headnode of that cluster. It is prohibited to run any jobs on the head node; if you ignore this rule, you risk losing the access to your account.

The headnode is to be used to edit and compile your code, as well as for launching your application on one or more cluster nodes. To edit your code, you can use `vi`, `emacs`, or `nano`. Alternatively, as hinted above, you can develop and test your code locally, and only upload it to the DAS5 for performance measurement.

To execute your code, you need to launch a job on one of the cluster nodes. To do so, you have two options: use the batch queueing system SLURM or use `prun`, which acts as a synchronous, shell-like interface, and was originally developed for DAS. We recommend using `prun`; however, if you prefer to use SLURM, documentation is to be found on the DAS5 website: `https://www.cs.vu.nl/das5/jobs.shtml`.

In a nutshell, using prun secures one or several cluster nodes that match a user's request, and starts the execution of a job on those nodes. To use prun, you must first load the `prun` module[1]. Note that for some of the assignments you will also need additional modules.

To see the available modules you can type:

```
$ module avail
```

To load the prun module type:

```
$ module load prun
```

When you are ready to test your program, you can schedule your job in the queue using prun.

```
$ prun -v -np <N> <program-name>
```

For example you can try:

```
$ prun -v -np 2 date
```

In the `prun` commands above, `-v` stands for *verbose*, while `-np <N>` specifies that we need to reserve N nodes for running the `<program-name>` application. Specifically, in the last example, `-np 2` indicates that we request two cluster nodes[2]. It can happen (especially close to deadlines)

[1] "Modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system. For example, one type of module is the device driver, which allows the kernel to access hardware connected to the system." - from `https://linux.die.net/lkmpg/x40.html`

[2] For the node configuration, please check `https://www.cs.vu.nl/das5/clusters.shtml`.

that all nodes are busy. In that case, your job will have to wait in the queue until resources become available.

You can check the content of the queue using `preserve`:

```
$ preserve -llist
```

If you want to only see the status of your job(s), you can filter the output of the command based on your username, For example, the following command will show all the reservations and jobs currently in the queue for user dpp20999:

```
$ preserve -llist | grep dpp20999
```

2.4 Advanced prun use

Any job launched using `prun` starts by implicitly reserving a number of nodes (as requested using `-np <N>`) for 15:00 minutes. Then, the specified application is executed on the reserved nodes. As soon as the application has finished, the reservation is cancelled.

If you want more control over the behaviour of prun and/or that of the reservation, you can specify additional parameters in your prun command. The most interesting one is the parameter that allows one to tweak the duration of a job.

```
$ prun -v -np <N>  -t <run-time> <program-name>
```

In this case, the job is scheduled to take `<run-time>`. Note that the limit of 15:00 cannot be exceeded during the day; however, choosing for shorter jobs might help reducing the waiting time in the queue.

For example, the following command will run a job for at most 2 minutes.

```
$ prun -v -np 1 -t 02:00 <program-name>
```

One additional challenge when using `prun` is that the job scheduler might provide you with a different node for every run. This can be somewhat challenging when you are comparing different runs of the same assignment, and may lead to non-negligible (yet still small) variability. To avoid this, you can run your final tests using preserved nodes. To preserve a node, you must use the `preserve` command as follows:

```
$ preserve -np <N> -s <start-time> -t <run-time> <program-name>
```

This command requests a reservation for N nodes, starting at `start-time`, and lasting `run-time`.

For example, try this:

```
$ preserve -np 1 -s 21:00 -t 03:00
```

This command requests the reservation of one node, for 3 minutes, starting at 21:00. Note that all the other parameters are identical to those of `prun`.

After posting your reservation, you can check its status using:

```
$ preserve -llist
```

In the list, you will see: the ID of the reservation, the user ID, the start and end data and time (projected for future times), the number of nodes, a status, and the allocated nodes (when available). If the status is `R`, the reservation is active and running. If, instead, the status is `PD`, your reservation is pending: no resources have been allocated. Here is an example:

```
2698015 dpp
20999  10/01    10:15   10/01   10:17  R  1  node053
```

Please note: it is often the case, especially for long, multi-node reservations, that the reservation is *pending* in the queue. To avoid long waiting times, and to limit resource waste, please make sure you do not make unnecessarily long or large reservations.

Once the reservation is running, you can use it to run your jobs. To do so, you need to use the reservation ID, as follows:

```
$ prun -np 1 -reserve <reservation-number> <program-name>
```

This will use 1 node from the reserved ones to run the application. If, for example, the reservation was made for 1 node, all the jobs running within this reservation will be using the same machine.

Concretely, the command can look like this:

```
$ prun -np 1 -reserve 2698015 <program-name>
```

In this case, the code will execute on machine `node053` (as seen in the snipped above, this is where the reservation is running).

In case you do not need a particular reservation, it is **very important** to cancel it. Please do not let reservations run idle to completion when not needed: it is wasteful and not very kind to your fellow DAS5 users. To cancel a reservation, all you need to do to cancle is to use preserve as follows:

```
$ prun -c <reservation-number>
```

2.5 Rules and restrictions

There are a few rules and restrictions to remember when using DAS5:

- Please change your password as soon as you receive it.
- Under no circumstances are you allowed to share the credentials of your account with others; your team-mate, the TAs, and the course teachers are the only ones that may constitute and exception.
- If you have trouble with DAS5, please contact your TAs or the teacher responsible for the assignment. Please avoid contacting the sysadmin unless your TAs/teacher instruct you to do so.
- You are not allowed to run jobs on the head node. All jobs are executed using SLURM or prun.
- You are not allowed to directly connect to specific nodes. All jobs are executed using SLURM or prun.

- Your jobs should never exceed 15:00 minutes from 8:00am to 8:00pm. You can, however, build scripts to run multiple tests within these 15:00 minutes.

- During the night (8:00pm to 8:00am), jobs can be longer. However, you are not to exceed 2h for all your running jobs at night (i.e., one job of 2h or 4 jobs of 30 minutes).

- For your MPI assignment, you are not allowed to run multi-node jobs for a total execution time larger than 4h - that is $Nnodes \times time - per - job < 4h$. Again, exceeding 15 minutes per job per node is not allowed during the day.

- Please do not run more than 4 jobs at a time for a non-MPI assignment, and no more than 1 job at a time for the MPI assignment.

IMPORTANT: Please note that not adhering by these rules can lead to your account being suspended.

2.6 Additional documentation

More comprehensive documentation can be found on the DAS5 website. Please make sure to consult that if you want to learn more about the actual architecture of the DAS5 system and its clusters, more detailed explanations about some of the commands you see in this document, or more insight into how DAS5 is used for research in the Netherlands.

3 Assignment-specific configuration

In this section we give specific instructions for the DPP "technologies" we use on DAS5.

3.1 OpenMP and pthreads

Your first assignment will focus on shared memory programming, using OpenMP and ptheads. There is no additional configuration needed for these models. To compile your code, additional flags might be needed, but they are already listed in the Makefiles. Finally, to execute your code, you will use the following:

```
$ prun -v -np 1 <program-name>
```

It is important to note that the pthreads and OpenMP assignments are to be tested on a single node - hence the use of `-np 1`.

3.2 MPI

Your second assignment will focus on MPI (Message Passing Interface). For this assignment, we will use OpenMPI. To enable OpenMPI, you need to enable an additional module (on top of prun):

```
$ module load openmpi/gcc
```

To check if the module is loaded correctly, thus enabling you to run MPI programs, please run:

```
$ which mpicc
```

If the command executes correctly, the result should be:

```
/cm/shared/apps/openmpi/gcc/64/1.10.3/bin/mpicc
```

If unsuccessful, the result should be something like:

```
/usr/bin/which: no mpicc in ...
```

To compile MPI code use the following:

```
$ mpicc -O2 -o <output> <sourcefile>
```

To run the code, you can use `prun` as follows:

```
$ prun -v -np <N> [<-n>] -script $PRUN_ETC/prun-openmpi <program-name>
```

This command will request N nodes (through `-np <N>`) and requests MPI to use n processes per node (through `-n`). The total number of MPI processes will be $N \times n$. Note that `-n` is optional: if omitted, the default value $n = 1$ is used - i.e., only one process is launched per node. The script `$PRUN_ETC/prun-openmpi`, which you can see with `less` or `cat`, is a generic MPI configuration script. If you need to, you can create your own copy and edit it as you see fit. Note, however, that this is not necessary for your assignments. Finally, note that `-v` enables a verbose output for `prun`. Such an output is useful to see which actual nodes are reserved for your assignment, and how many processes are launched on each node.

Here is an example of a full command:

```
$ prun -v -np 2 -4 -script $PRUN_ETC/prun-openmpi ./mpi-hello
```

Here, `-np 2` means you request 2 nodes, and `-4`, means you request 4 processes per node - thus, in total, $2 \times 4 = 8$ processes.

3.3 GPU programming

Your third assignment will focus on GPU programming. For this assignment, you will program a GPU using CUDA. Specifically, we will use CUDA version 10.1.

To add CUDA 10.1 to your environment, use:

```
$ module load cuda10.1/toolkit
```

If the module has been correctly loaded, you should be able to run the CUDA compiler, like this:

```
$ nvcc --version
```

The output should be:

```
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Sun_Jul_28_19:07:16_PDT_2019
Cuda compilation tools, release 10.1, V10.1.243
```

For running CUDA jobs, you can use:

```
$ prun -v -np 1 -native '-C␣<gpu-name>␣--gres=gpu:1' <program-name>
```

Specifically, you request one single node (which is always the case for your GPU assignment), but you specify that the node must have 1 GPU (via `--gres=gpu:1`) or a certain type (e.g., `<gpu-name>`). For all available GPUs, consult the DAS5 web-page, here: `https://www.cs.vu.nl/das5/special.shtml`.

For example, to execute NVIDIA's system management interface program[3], `nvidia-smi`, you can try:

```
$ prun -v -np 1 -native '-C␣TitanX␣--gres=gpu:1' nvidia-smi
```

4 Summary

DAS5 is a shared machine, used by many researchers in the Netherlands. We are grateful for being allowed to use the machine, as this is a real-world cluster configuration, similar to those available in real data-centers.

In this document, you have a clear overview of all the necessary basic commands to correctly use the DAS machines. More details will be provided as needed for each assignment.

To keep the system functioning smoothly and use it wisely, we ask you kindly to be mindful about your usage, and aim not to waste resources or clutter the system with too many unnecessary jobs. Please remember the maximum job time is 15:00 minutes, which should be sufficient for all your assignments. Also, please remember that your jobs must not be executed on the headnode.

[3] "The NVIDIA System Management Interface (nvidia-smi) is a command line utility, based on top of the NVIDIA Management Library (NVML), intended to aid in the management and monitoring of NVIDIA GPU devices." - from `https://developer.nvidia.com/nvidia-system-management-interface`.