PGI on HPC

What is PGI?

PGI stands for Portland Group, Inc. PGI compilers are a suite of high-performance compilers and development tools developed by Portland Group, Inc. (now part of NVIDIA) for use in high-performance computing (HPC) systems. They are designed to help programmers create efficient and optimized code for a variety of architectures, including x86, ARM, and GPU.

The PGI compilers include:

- The PGFortran compiler, which is used to compile Fortran code.
- The PGCC compiler, which is used to compile C code.
- The PGC++ compiler, which is used to compile C++ code.

All these compilers are designed to work together, allowing developers to create and optimize code using a variety of programming languages. They support the latest standards and include advanced optimization features such as auto-parallelization, vectorization, and OpenMP.

PGI also provides various libraries, such as the PGI Accelerator Math Library (PAMI) for mathematical computations on GPU and the PGI CUDA Fortran for writing CUDA kernels in Fortran.

Links:

Official Website

Versions Available:

The following versions are available on the cluster:

- PGI 18.5-0

How to load PGI?

To load PGI, use the following commands:

```
#Load the PGI module
module load compilers/pgi
```

To verify if the module is loaded correctly, use the following command,

```
# List all the module loaded in the environment
module list
```

In a fresh environment, this should only load PGI since it is an independent compiler collection.

How to use PGI?

Since this is compiling and profiling software from NVIDIA, there are several ways to use it. Some of the useful utility included in the package are,

1. **pgfortran:** The command-line compiler for Fortran programs. It can be used to compile and link Fortran source code into an executable program.
2. **pgcc:** The command-line compiler for C programs. It can be used to compile and link C source code into an executable program.

3. **pgc++**: The command-line compiler for C++ programs. It can be used to compile and link C++ source code into an executable program.
4. **pgdbg**: A source-level debugger for PGI-compiled code. It can be used to debug programs and find and fix errors.
5. **pgprof**: A performance profiler for PGI-compiled code. It can be used to analyze the performance of programs and identify potential bottlenecks.
6. **pgacce**: An accelerator programming tool for PGI-compiled code. It can be used to create GPU-accelerated versions of programs and optimize performance on GPU-enabled systems.
7. **OpenACC:** A directive-based programming model for GPU acceleration. It allows you to add directives to your code to specify which regions should be executed on the GPU.
8. **OpenMP**: A programming model for shared-memory parallelism. It allows you to add directives to your code to specify which regions should be executed in parallel.
9. **PGI Accelerator Math Library (PAMI)**: A library that provides optimized mathematical functions for use with PGI compilers and OpenACC.
10. **PGI CUDA Fortran**: A tool for writing CUDA kernels in Fortran. It allows you to use CUDA programming model for GPU acceleration for Fortran code.

Users may need to enable X11-forwarding to use some of the GUI (Graphical User Interface) tools. Log in using following command to enable X11-forwarding,

```
# Login with display forwarding enabled
ssh -X  bigal@uahpc.ua.edu
```

Use the following fortan script as sample to use pgfortan,

```
program average
implicit none

integer, parameter :: n=5
real :: numbers(n), sum, avg

! Input numbers
do i=1,n
    print *, "Enter a number:"
    read *, numbers(i)
end do
```

```
! Calculate sum
sum = 0.0
do i=1,n
    sum = sum + numbers(i)
end do

! Calculate average
avg = sum / n

! Print result
print *, "The average of the numbers is: ", avg

end program average
```

Use the following to compile the above fortan code using pgi compilers,

```
# Compile and Execute
pgfortran -fast -Minfo=all script.f90  -o myscript
```

These all task can be wrapped up in a slurm script and be submitted to scheduler.

Use the following as a sample slurm script,

```
#!/bin/bash
#SBATCH --job-name=pgi     # Job name
#SBATCH -p main            # Partition (queue)
#SBATCH --qos main
#SBATCH --ntasks=1                    # Number of tasks/cores
#SBATCH --cpus-per-task=5             # Number of cores per task
#SBATCH --mem=1024MB                  # Memory per node
#SBATCH --time=00:05:00               # Time limit hrs:min:sec
#SBATCH --output=parallel_print.out   # Standard output and error log

module load compilers/pgi

# ALL THE STEPS HERE
```

Where to find help?

If you are confused or need help at any point, please contact OIT at the following address.

https://ua-app01.ua.edu/researchComputingPortal/public/oitHelp