Fortran/C on HCC

This quick start demonstrates how to implement a Fortran/C program on HCC supercomputers. The sample codes and submit scripts can be downloaded from <serial_dir.zip>.

Login to a HCC Cluster (Tusker or Sandhills)

Log in to a HCC cluster through PuTTY (For Windows Users) or Terminal (For Mac/Linux Users) and make a subdirectory called serial_dir under the $WORK directory.

```
$ cd $WORK
$ mkdir serial_dir
```

In the subdirectory serial_dir, save all the relevant Fortran/C codes. Here we include two demo programs, demo_f_serial.f90 and demo_c_serial.c, that compute the sum from 1 to 20.

```
demo_f_serial.f90

Program demo_f_serial
  implicit none
  integer, parameter :: N = 20
  real*8 w
  integer i
  common/sol/ x
  real*8 x
  real*8, dimension(N) :: y

  do i = 1,N
    w = i*1d0
    call proc(w)
    y(i) = x
    write(6,*), 'i,x = ', i, y(i)
  enddo
  write(6,*), 'sum(y) =', sum(y)
stop
end Program

Subroutine proc(w)
  real*8, intent(in) :: w
  common/sol/ x
  real*8 x

  x = w

return
end Subroutine
```

```
demo_c_serial.c

#include <stdio.h>

int main()
{
  int i, N = 20;
  float x = 0.0;
  float w;

  for (i = 1; i <= N; i++)
  {
    w = i * 1.0;
    // call proc(w)
    x = w;
    printf("i, x = %d, %f\n", i, x);
  }
  printf("sum(y) = %f\n", sum(y));
  return 0;
}
```
//demo_c_serial
#include <stdio.h>

double proc(double w){
    double x;
    x = w;
    return x;
}

int main(int argc, char* argv[]){
    int N=20;
    double w;
    int i;
    double x;
    double y[N];
    double sum;
    for (i = 1; i <= N; i++){
        w = i*1e0;
        x = proc(w);
        y[i-1] = x;
        printf("i,x= %d %lf
", i, y[i-1]) ;
    }
    sum = 0e0;
    for (i = 1; i<= N; i++){
        sum = sum + y[i-1];
    }
    printf("sum(y)= %lf
", sum);
    return 0;
}

Compiling the Code

The compiling of a Fortran/C++ code to executable is usually done behind the scene in a Graphical User Interface (GUI) environment, such as Microsoft Visual Studio. In a HCC cluster, the compiling is done explicitly by first loading a choice compiler and then executing the corresponding compiling command. Here we will use the GNU Compiler Collection, gcc, for demonstration. Other available compilers such as intel or pgi can be looked up using the command line module avail. Before compiling the code, make sure there is no dependency on any numerical library in the code. If invoking a numerical library is necessary, contact a HCC specialist (hcc-support@unl.edu) to discuss implementation options.

$ module load compiler/gcc/7.1
$ gfortran demo_f_serial.f90 -o demo_f_serial.x
$ gcc demo_c_serial.c -o demo_c_serial.x

The above commend load the gcc compiler and use the compiling commands gfortran or gcc to compile the codes to .x files (executables).
Creating a Submit Script

Create a submit script to request one core (default) and 1-min run time on the supercomputer. The name of the main program enters at the last line.

<table>
<thead>
<tr>
<th>submit_f.serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>#!/bin/sh</td>
</tr>
<tr>
<td>#SBATCH --mem-per-cpu=1024</td>
</tr>
<tr>
<td>#SBATCH --time=00:01:00</td>
</tr>
<tr>
<td>#SBATCH --job-name=Fortran</td>
</tr>
<tr>
<td>#SBATCH --error=Fortran.%J.err</td>
</tr>
<tr>
<td>#SBATCH --output=Fortran.%J.out</td>
</tr>
<tr>
<td>module load compiler/gcc/4.9</td>
</tr>
<tr>
<td>./demo_f_serial.x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>submit_c.serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>#!/bin/sh</td>
</tr>
<tr>
<td>#SBATCH --mem-per-cpu=1024</td>
</tr>
<tr>
<td>#SBATCH --time=00:01:00</td>
</tr>
<tr>
<td>#SBATCH --job-name=C</td>
</tr>
<tr>
<td>#SBATCH --error=C.%J.err</td>
</tr>
<tr>
<td>#SBATCH --output=C.%J.out</td>
</tr>
<tr>
<td>module load compiler/gcc/4.9</td>
</tr>
<tr>
<td>./demo_c_serial.x</td>
</tr>
</tbody>
</table>

Submit the Job

The job can be submitted through the command `sbatch`. The job status can be monitored by entering `squeue` with the `-u` option.

```
$ sbatch submit_f.serial
$ sbatch submit_c.serial
$ squeue -u <username>
```

Sample Output

The sum from 1 to 20 is computed and printed to the `.out` file (see below).
Fortran.out

i, x = 1 1.0000000000000000
i, x = 2 2.0000000000000000
i, x = 3 3.0000000000000000
i, x = 4 4.0000000000000000
i, x = 5 5.0000000000000000
i, x = 6 6.0000000000000000
i, x = 7 7.0000000000000000
i, x = 8 8.0000000000000000
i, x = 9 9.0000000000000000
i, x = 10 10.0000000000000000
i, x = 11 11.0000000000000000
i, x = 12 12.0000000000000000
i, x = 13 13.0000000000000000
i, x = 14 14.0000000000000000
i, x = 15 15.0000000000000000
i, x = 16 16.0000000000000000
i, x = 17 17.0000000000000000
i, x = 18 18.0000000000000000
i, x = 19 19.0000000000000000
i, x = 20 20.0000000000000000
sum(y) = 210.0000000000000000

C.out

i, x = 1 1.000000
i, x = 2 2.000000
i, x = 3 3.000000
i, x = 4 4.000000
i, x = 5 5.000000
i, x = 6 6.000000
i, x = 7 7.000000
i, x = 8 8.000000
i, x = 9 9.000000
i, x = 10 10.000000
i, x = 11 11.000000
i, x = 12 12.000000
i, x = 13 13.000000
i, x = 14 14.000000
i, x = 15 15.000000
i, x = 16 16.000000
i, x = 17 17.000000
i, x = 18 18.000000
i, x = 19 19.000000
i, x = 20 20.000000
sum(y) = 210.000000