Serial & Vectorization Optimization Hands-on Lab

Jerome Vienne
viennej@tacc.utexas.edu

TACC Summer Supercomputing Institute
July 8th, 2013

Goals

- Automatic Optimization
  - Compile a code with different optimization levels and study the run times.
  - This example shows how automatic optimization can greatly improve code performance without effort.
  - We will compare performances of Intel compilers and the GNU Compiler Collection
- Vectorization
  - Try to find why a kernel is not being vectorized by the compiler and change the code to fix the problem.
  - This example illustrates the common problem of data dependencies.

Setup

- Login to Stampede:
  - `ssh username@stampede.tacc.utexas.edu`
- Untar the lab files:
  - `cd`
  - `tar xvf ~train00/opt_lab.tar`
- Change directories and ls to see the files:
  - `cd opt_lab`
  - `ls`
- You should see both C and F90 versions of the code

The GNU Compiler Collection

- Use latest gcc version:
  module swap intel gcc/4.7.1
- Compile auto.c or auto.f90 with increasing levels of automatic optimization:

```
gcc -O0 auto.c -o auto_O0 -lm
gcc -O1 auto.c -o auto_O1 -lm
gcc -O2 auto.c -o auto_O2 -lm
gcc -O3 auto.c -o auto_O3 -lm

gfortran -O0 auto.f90 -o fauto_O0
gfortran -O1 auto.f90 -o fauto_O1
gfortran -O2 auto.f90 -o fauto_O2
gfortran -O3 auto.f90 -o fauto_O3
```

The GNU Compiler Collection

- Run each of the executables and make note of the Setup and Kernel execution times.
- Fill a table with the execution times for the different optimization levels.
- Observation ?

Intel Compilers

- Use latest Intel version:

  module swap gcc intel/13.1.1.163

- Compile auto.c or auto.f90 with increasing levels of automatic optimization:

```
icc -O0 auto.c -o iauto_O0 -lm
icc -O1 auto.c -o iauto_O1 -lm
icc -O2 auto.c -o iauto_O2 -lm
icc -O3 auto.c -o iauto_O3 -lm

ifort -O0 auto.f90 -o ifauto_O0
ifort -O1 auto.f90 -o ifauto_O1
ifort -O2 auto.f90 -o ifauto_O2
ifort -O3 auto.f90 -o ifauto_O3
```

Intel Compilers

- Run each of the executables and make note of the Setup and Kernel execution times.
- Fill a table with the execution times for the different optimization levels.
- Observation ?
- Can we do better ?

Vectorization

- Compile the vec.c or vec.f90 source code

  icc vec.c -o vec -xHost -vec-report2

  ifort vec.f90 -o vec -xHost -vec-report2
- Run the executable and take note of the time spent in the kernel.
- Open the source code with your favorite text editor and look at the loop named KERNEL.
- Identify the source of the data dependence and correct it, so that the compiler is able to vectorize both the setup and the kernel loops.
- Recompile and verify both loops are vectorized.
- Submit the job again and compare the timings with the original.

Done !

Conclusion ?

Question ?