Bryan Acosta and Justin Campbell's Hardware HW 1 writeup using LaTeX

Eids: ba25389 and jsc4348

11/23/2021

1 Introduction:

Interactive access to a set of compute nodes allows for users to quickly compile, run and validate MPI and other applications in rapid succession, which is very useful for development. The app, “idev”, stands for “Interactive Development”, and was created by TACC to allow users to acquire compute nodes for interactive access. In this assignment, the “idev” application will be used to access the “Cascade Lake development queue” to edit, compile and run a program, “vector.c” to study code optimization. The icc compiler will be used to explore the effect that different compilation schemes have on the run-time/performance of the program.

2 results:

3) Starting an interactive session

When the command “idev -p development” is run in Frontera, an interactive session is started in the terminal. As shown in the figure below, a command prompt appears in the interactive development session and status messages are presented that report the condition of the job running on a masternode and the condition of the initialization of a development environment.

c) What does the option “-p” do?

- The “-p” flag creates a development queue in the environment

d) What does the command “hostname” do?

- The command “hostname” returns the computer’s DNS (Domain Name System), which is a name given to a computer, and attached to a network to uniquely identify that computer over the network.
Hostname: c205-034[clx]

f) Add the name of the compute node to your report and report multiple hostnames if necessary.
- After running the command inside the author’s interactive session, the names of the two compute nodes the author ran are “c202-001.frontera.tacc.utexas.edu”, and “c203-005.frontera.tacc.utexas.edu”, and “c203-001.frontera.tacc.utexas.edu”, “c203-018.frontera.tacc.utexas.edu”,

3 Instructions: Setup:

a) Create a directory where you will conduct the experiment. The directory where the author conducted the experiment is specified through the following full path:

Full Path: /home1/07674/ba25389/hw1

1. Size of arrays a,b,c

a and c: they are arrays of doubles and each double is 64 bits. the array is 1024 indexes long. A and C are 64*1024 bits, or 8 kilobytes kilobytes. b: is a matrix of 1024*1024. B is made of doubles. So the size of B is 64*1024*1024 or 8 megabytes.

	a	b	c
number of elements	1024	1048576	1024
number of bytes	8.0 kB	8.0 Mb	8.0 kB

2. A: The size of a “Level 3” cache of single socket of a Frontera Cascade Lake node is 38.5 Megabytes which is significantly larger than the size of arrays “a”, and “b”, and approximately four times the size of array “B”. The total size of the three arrays is roughly one-quarter of the socket’s full memory.

3. A: The purpose of the innermost 2 loops in the nested for loop between the start and the end of the timer is to iterate through the entries in the vectors “a”, and “c”, and matrix “B” to update the value of “a” according to the operation “a = B * c”.

4. the outer loop adds up all of the values for A

4 Instructions: Compiling and Executing:

	min run time	avg run time
01	31.614	31.771
02	3.341	3.355
02 -no-vec	6.469	6.470

5. b. The first compilation flag specifies the phase of the optimization; in this case, the report characterizes optimization performed on vectors in the “vector.c” program. The second flag tells the compiler the level of detail with which to express the report in. A level of 3 indicates that information will be presented in a moderate level of detail.

6. a. vector.optrpt

b. 26, 36, 48

d. the compiler reordered the loops at line 36 and 37. instead of 36 then 37, it

was 37 to 36.
e. the compiler changed

```
for (n = 0; n < iter; n++) {
  a[0] = 0.;
  for(j = 0; j < N; j++)
    for(i = 0; i < N; i++)
      a[i] = a[i] + B[i][j]*c[j];
}
```

into

```
for (n = 0; n < iter; n++) {
  a[0] = 0.;
    for(i = 0; i < N; i++)
      for(j = 0; j < N; j++)
      a[i] = a[i] + B[i][j]*c[j];
}
```

f. Reordering the for loops impacts the access pattern to the arrays as it influences the speed with which data can move from the memory to the cache, and ultimately to the CPU. This in turn impacts the amount of time it takes for the CPU to process the arithmetic operation in the final line of the for loop. The ordering of the loops affects the distance that the data of the loop counters “i” and “j” must move to and from the main memory and the CPU, and the greater the distance, the greater the run-time of the loops. When the loops are ordered such that the outermost loop iterates over the loop counter “i” and the innermost loop iterates over the loop counter “j”, the compiler first accesses elements in array “c” and then respectively “B” and “a” as the arrays that operate on the loop counter “j” are accessed first. The converse is true for an ordering where the innermost loop iterates over “i” and the outermost loop over “j”, namely, the compiler accesses elements in array “a”,”B”, and “c” respectively.

g. The access pattern of the reordered code leads to better performance because the data of the loop counter “j” , that are used in the more computationally-intensive, arithmetic operation (B[i][j] * c[j]), travel over a lesser distance to and from the main memory and the CPU than the data of the loop counter, “i” that are used in the less computationally-intensive arithmetic operation (addition and assignment respectively from right to left). Reordering the loop so that the innermost loop counter iterates over the loop counter (j) used in the more computationally-intensive arithmetic operation helps to minimize the distance that the data must travel between cycles/iterations and thus, improves latency and minimizes runtime.

7. -no-vec disables vectorization. The execution with -no-vec is larger because vectorization makes the compiler run a code faster. So if you tell a compiler to not vectorize a code, it will run it the long way.

8. 01 is slower than 02 becuase 02 and higher automatically vectorized the code and thus is faster. So 01 is slower and larger.