Programming for Peformance, Assignment

Spring 2022, IIIT-H
Suresh Purini

1 Know your Computer (KYC)

Identify the following information with respect to the computing system you are using.

1. CPU model, generation, frequency range, number of cores, hyper threading availability, SIMD ISA, cache size, main memory bandwidth, etc. Refer `https://www.intel.com/content/www/us/en/support/articles/000032203/processors/intel-core-processors.html`

2. Theoretically estimate peak FLOPS per core and per processor. Compare the peak FLOPS per core with what you get using Whetstone benchmark program. Design and develop a benchmark program(s) using which the peak FLOPS per core and processor are achieved.You have to write a parallel program to achieve peak FLOPS per for processor. Report the theoretical and empirically achieved peak FLOPS.

3. What is the main memory size? Is it DDR3 or DDR4? Maximum main memory bandwidth availability. Check the main memory bandwidth using the Stream benchmark program. Write a benchmark program which achieves the peak main memory bandwidth.

4. What is the secondary storage structure size? SSD or HDD. Read/Write bandwidth of the secondary storage structure. If you have both SSD and HDD on your system, compare their performance.

5. Install gcc, llvm and Intel icc compilers.

2 Know your Cluster

What is the peak FLOPS for Ada and Abacus cluster? Refer `http://hpc.iiit.ac.in/wiki/index.php/Main_Page`

3 BLAS Problems

BLAS stands for Basic Linear Algebra Subprogams. Please read the BLAS wiki page `https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms` and the NetLib

page http://www.netlib.org/blas/. Please check and install Blis which is a specific implementation of BLAS problems https://github.com/flame/blis.

In this assignment we will try to develop high performance code for a subset of BLAS programs. You have to document your journey systematically, like how with each iteration the performance of the program has improved (or decreased!) and what is your approach for design space exploration, and finally submit a detailed final report, along with code. The following are BLAS programs you have to optimize. Consider both floating-point and double precision version of codes.

3.1 BLAS Level 1

BLAS Level 1 consists of Scalar-Vector and Vector-Vector operations.

1. (xSCAL) Scalar multiplication of a vector. x in xSCAL stands for floating or double.

$$X = \alpha X \tag{1}$$

2. (xAXPY) Scalar Multiplication of a vector followed by addition with another vector.

$$Y = \alpha X + Y \tag{2}$$

3.2 BLAS Level 2

BLAS Level 2 consists of matrix-vector operations.

1. (xGEMV) Matrix times a vector.

$$Y = \alpha AX + \beta Y \quad \text{or } Y = \alpha A^T X + \beta Y \tag{3}$$

3.3 BLAS Level 3

BLAS Level 3 consists of matrix-matrix operations.

1. (xGEMM) Matrix-Matrix Multiplication

$$C = \alpha op(A) op(B) + \beta C \quad where \;\; op(A) = A \; or \; A^T. \tag{4}$$

Compare the performance of your best implementation with BLIS implementation. Report the achieved FLOPS and the speedup numbers from the baseline. You can also compare against ATLAS implementation. https://en.wikipedia.org/wiki/Automatically_Tuned_Linear_Algebra_Software

Challenge Problem: Find the optimal performance points through design space search. You can visually represent the design space.

Please refer to the documentation on netlib.org, BLAS quick reference guide and the provided cblas.h file

3.4 Stencil computation

Write a program which takes a HD (1920 x 1080) or an Ultra HD (3840 x 2160) image and applies $k \times k$ stencil. In order obtain an output image of same dimensions apply zero padding to the original image borders. The function prototype may look like the follwing.

```
enum ImageType {HD, UHD};
void stencil(float *X, ImageType typ, int k, float *S, float *Y)
```

X points to the base address of the image, *typ* tells whether the image is HD or UHD, k is the stencil dimension, S is the base adress of the stencil, Y points to the base address of the output image.

4 Report Details

For each programming problem report the following minimum details should be there. Wherever it is applicable, report the following data for varying input sizes in a graph form?

1. What is its operational intensity?
2. What are the execution times using gcc -O3 and icc -O3?
3. What is the baseline and best execution times you obtained?
4. What is the speedup?
5. What are the baseline and optimized GFlops/sec?
6. What optimization strategies used to achieve the same?
7. What is the memory bandwidth achieved?
8. In the baseline and optimized versions, is the problem memory or compute bound?

Please note that the report carries a lot of weight for the grade.