

Software Programming for Performance

Name : Abhijnan Vegi

Roll No : 2020101004

Assignment 1

Know your Computer

CPU Specifications

| Specification | Value |
|--------------------------|-----------------------|
| CPU | AMD Ryzen 7 |
| Model | 4800H |
| Frequency Range | 2.9GHz - 4.2GHz |
| Number of physical cores | 8 |
| Hyperthreading | Yes |
| SIMD ISA | AVX2 |
| Cache | 512 KiB, 4 MiB, 8 MiB |
| Memory Bandwidth | 68.27 GB/s |

Theoretical FLOPS

Whetstone benchmark

Running the whetstone benchmark with 10^6 loops gives a total of 6.25 GIPS.

```
Loops: 1000000, Iterations: 1, Duration: 16 sec.  
C Converted Double Precision Whetstones: 6250.0 MIPS
```

Compiling the whetstone benchmark with the flag `-O3` we get the following result

```
Loops: 1000000, Iterations: 1, Duration: 4 sec.  
C Converted Double Precision Whetstones: 25000.0 MIPS
```

Using the `icc` compiler instead of `gcc` gives us 10000 MIPS even with the `-O3` flag

My benchmark

My benchmark has been able to achieve 400 GFLOPS.

Memory Specifications

| Specification | Value |
|--------------------------|------------|
| Memory size | 32 GiB |
| Type | DDR4 |
| Maximum Memory Bandwidth | 34.13 GB/s |

My memory benchmark has been able to achieve a total of 33.5 GB/s.

Storage Specifications

1. SSD

| Specification | Value |
|---------------|-----------|
| Size | 256 GiB |
| Read | 2400 MB/s |
| Write | 950 MB/s |

2. HDD

| Specification | Value |
|---------------|----------|
| Size | 1 TiB |
| Read | 160 MB/s |
| Write | 40 MB/s |

Know your Cluster

1. ADA peak FLOPS : 70.66 TFLOPS
2. Abacus Peak FLOPS : 14 TFLOPS

BLAS Problems

3.1 BLAS Level 1

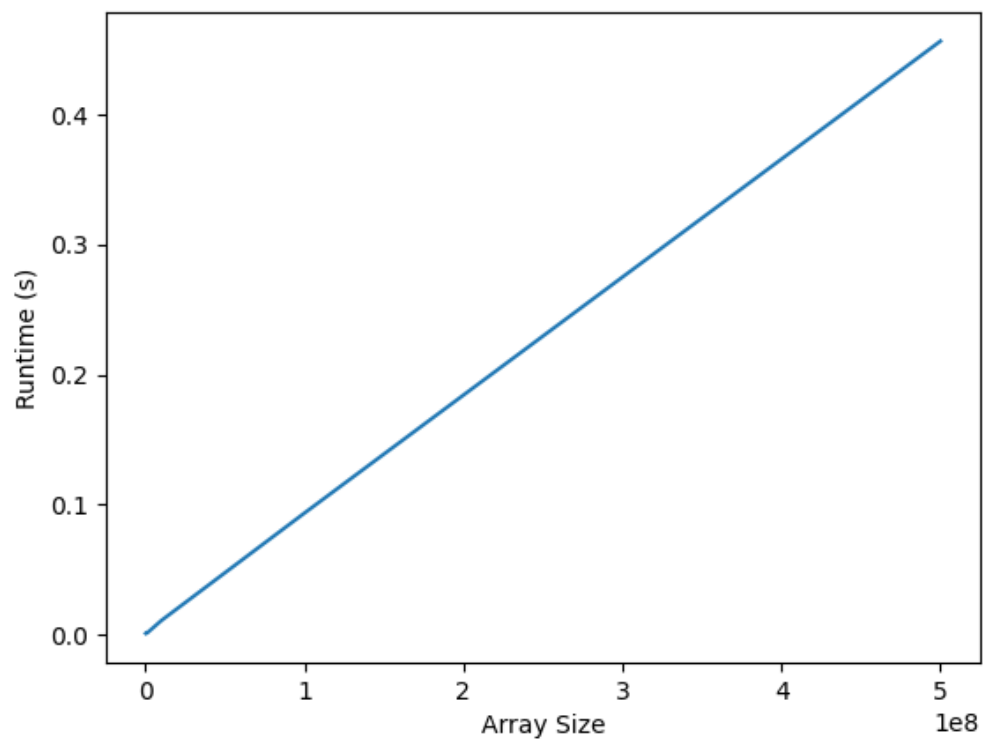
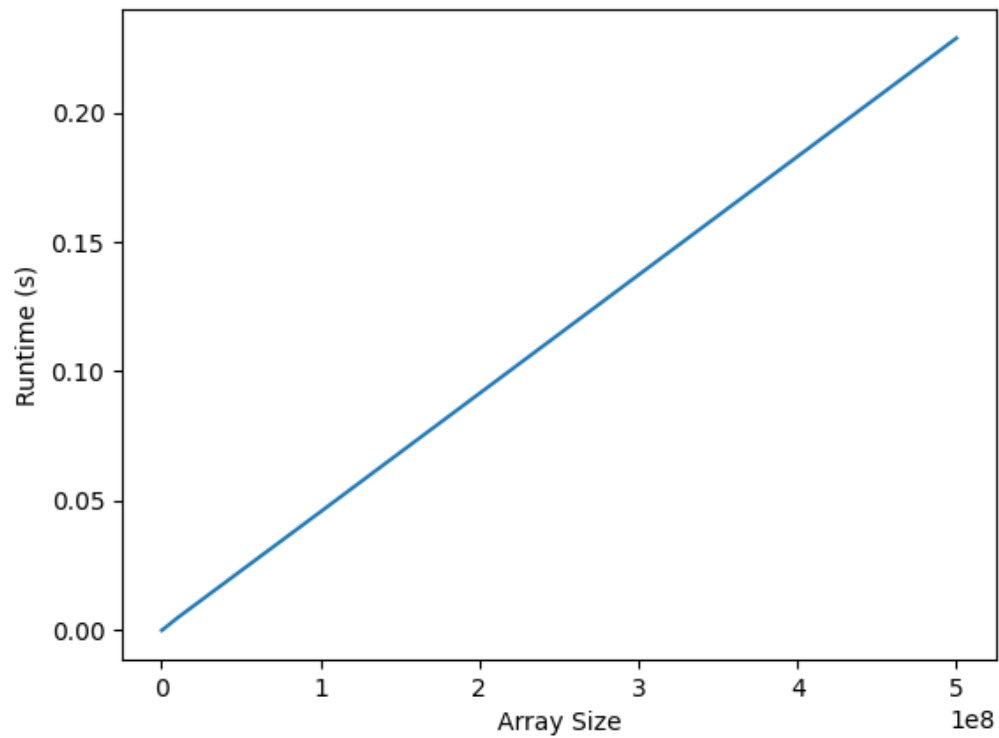
xSCAL

- Operational Intensity : 0.25 FLOP/Byte for `float` , 0.125 FLOP/Byte for `double`
- Execution times (input size : $1e^8$)

```
sscal with gcc : 50ms  
dscal with gcc : 100ms
```

```
sscal with icc : 55ms  
dscal with icc : 110ms
```

- Baseline execution time : 400ms
Best execution time : 50ms
- Speedup : $8\times$
- Baseline GFLOPS : 0.25
Best GFLOPS : 2.0
- Optimization strategies : Vectorization
- Memory bandwidth : 8 GB/s
- The problem is CPU bound!

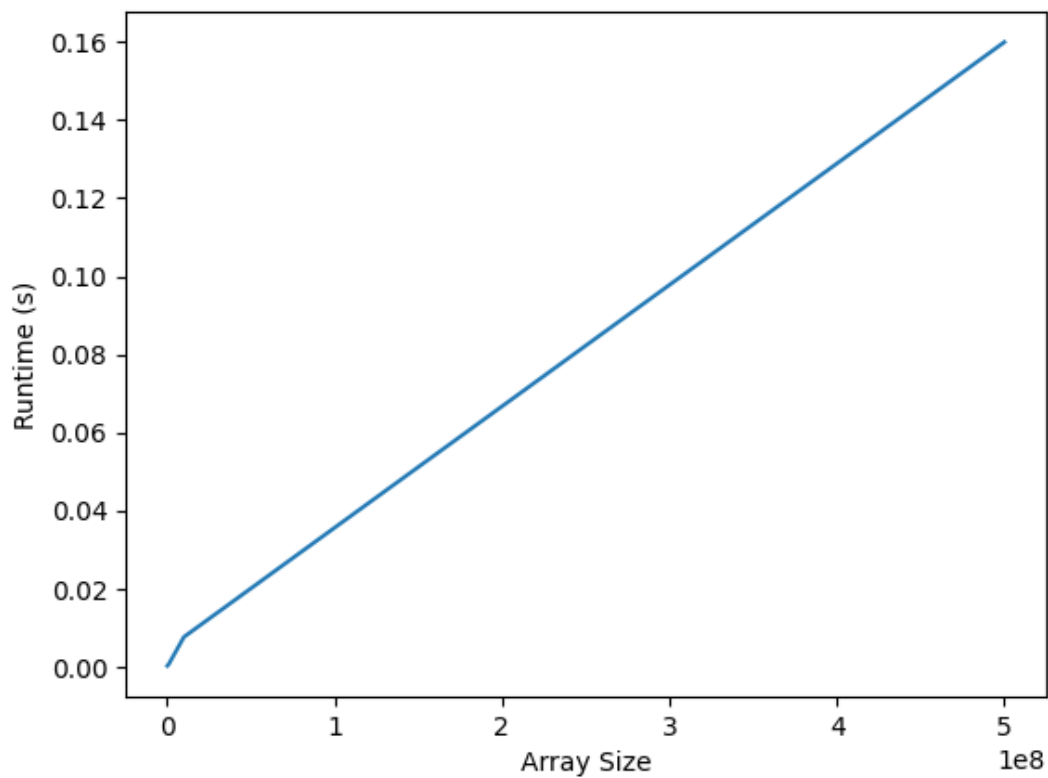


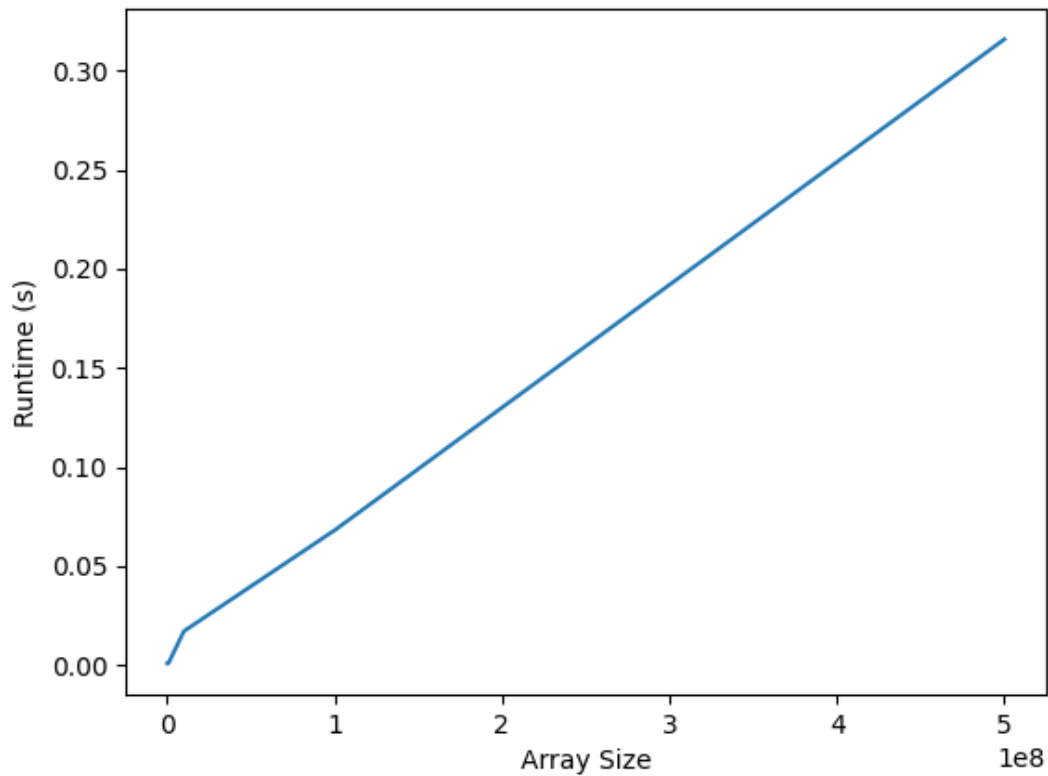
xDOT

- Operational Intensity : 0.25 FLOP/Byte for float and 0.125 FLOP/Byte for double
- Execution times (Input size $1e^8$)

```
sdot with gcc : 40ms  
sdot with icc : 55ms  
  
ddot with gcc : 75ms  
ddot with icc : 80ms
```

- Baseline execution time : 700ms
Best execution time : 40ms
- Speedup : $14\times$
- Baseline GFLOPS : 0.3
Best GFLOPS : 4
- Optimization strategies : O3 and vectorization
- Memory bandwidth : 16 GB/s
- Problem is memory bound



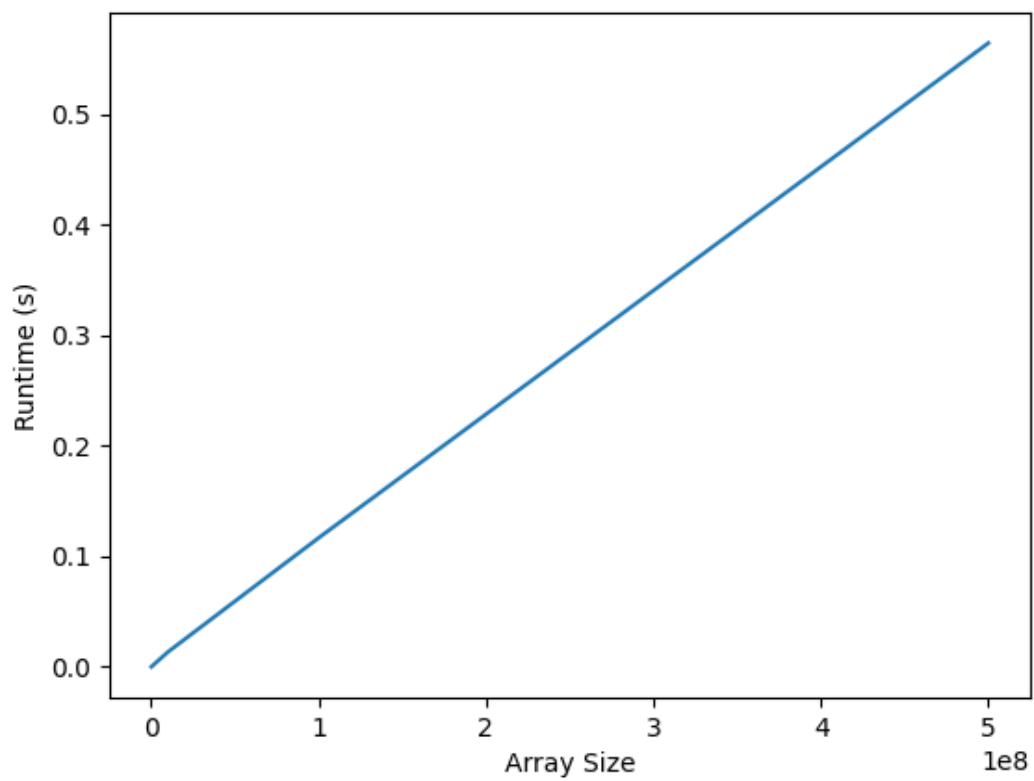
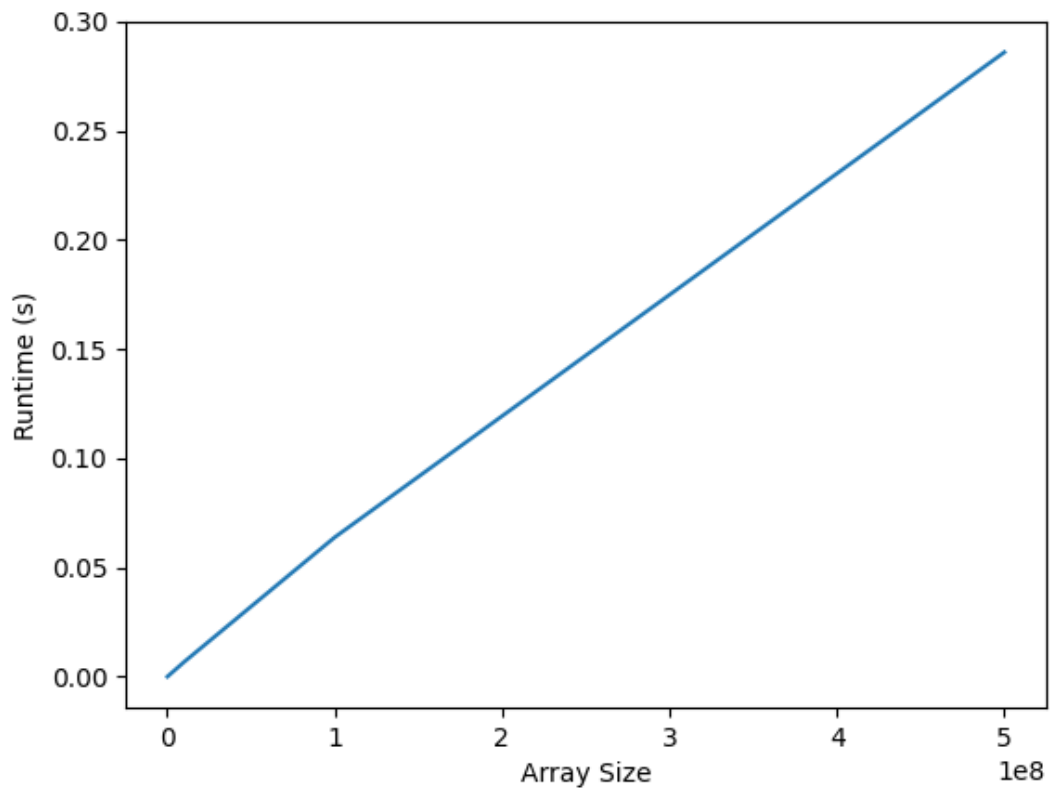


xAXPY

- Operational Intensity
- Execution times (Input size $1e^8$)

```
saxpy with gcc : 60ms  
saxpy with icc : 75ms  
  
daxpy with gcc : 115ms  
daxpy with icc : 130ms
```

- Baseline execution time : 650ms
Best execution time : 60ms
- Speedup : $10\times$
- Baseline GFLOPS : 0.3
Best GFLOPS : 3.5
- Optimization strategies : O3 and Vectorization
- Memory bandwidth : 16 GB/s
- Problem is memory bound



3.2 BLAS Level 2

xGEMV

- Operational Intensity : $\frac{3MN+M}{MN+M+N}$
- Execution times (Input size $1e^8$)

```
sgemv with gcc : 15ms  
sgemv with icc : 19ms  
  
dgemv with gcc : 28ms  
dgemv with icc : 32ms
```

- Baseline execution time : 300ms
Best execution time : 15ms
- Speedup : $20\times$
- Baseline GFLOPS : 0.2
Best GFLOPS : 10
- Optimization strategies : `-O3` , parallelization with OpenMP and vectorization
- Memory bandwidth : 28 GB/s
- Problem is CPU bound

3.3 BLAS Level 3

xGEMM

- Operational Intensity : $\frac{3MNK+MN}{MN+NK+MN}$
- Execution times (Matrix sizes $1e^4$)

```
sgemm with gcc : 38ms  
sgemm with icc : 60ms  
  
dgemm with gcc : 80ms  
dgemm with icc : 95ms
```

- Baseline execution time : 2700ms
Best execution time : 38ms
- Speedup : $71\times$
- Baseline GFLOPS : 1.12
Best GFLOPS : 7
- Optimization strategies : `-O3` , parallelization with OpenMP and vectorization
- Memory bandwidth :
- Problem is CPU bound

Comparision with blis

| Operation | My Implementation | Blis |
|--------------------|-------------------|-------|
| <code>sscal</code> | 50ms | 50ms |
| <code>dscal</code> | 100ms | 100ms |
| <code>sdot</code> | 40ms | 40ms |
| <code>ddot</code> | 75ms | 77ms |
| <code>saxpy</code> | 58ms | 56ms |
| <code>daxpy</code> | 112ms | 113ms |
| <code>sgemv</code> | 15ms | 12ms |
| <code>dgemv</code> | 28ms | 28ms |
| <code>sgemm</code> | 38ms | 25ms |
| <code>dgemm</code> | 80ms | 50ms |

2D Stencil

- Operational Intensity : k^2
- Execution times

```

HD:
gcc : 10ms
icc : 50ms

UHD:
gcc : 30ms
icc : 130ms

```

`icc` looks heavily unoptimized. This might be due to the fact that its being tested on an AMD processor rather than an Intel one.

- Baseline execution time : 130ms
Best execution time : 10ms

Note that the baseline execution time is after compiling with gcc rather than icc

- Speedup : $13\times$
- With $k = 3$ and HD image
Baseline GFLOPS : 0.32
Best GFLOPS : 3.87
- Optimization strategies : `-O3` and parallelization with openmp
- Memory bandwidth : 1.12 GB/s
- The problem is compute bound

On HD image with varying stencil size the runtime varies exponentially as expected.

