Lam Dang

Matrix-Matrix Multiplication Design Sheet

Notice: Please read the README.md file before continue

Serial Program provided by charliep:

- Partitioning:
    1. Command Line Argument Fetching: A lot of if-else statements, require input to be put in the correct order, possibility of error is high, error-handling
    2. Initialize three matrices: Serial when it can be parallelized
    3. Matrix Multiplication: Serial when it can be parallelized

Execution Time:

```
hopper$ ./charlie_mmm 1000 10 1.4 hopper
# itemsPerDimension: 1000, repeats: 10, platform: hopper, coreSpeed: 1.400
# platform, totalBytes, wallTimeForAll, wallTimeForOne, MatrixKBPerSecond
hopper, 4000000, 78.314, 7.831, 498.794
```

Program Profiling:

```
main  /cluster/home/ltdang16/cs360-paral/mmm/charlie_mmm
              mov    -0x60(%rbp),%edx
  0.00        movslq %edx,%rdx
  4.63        mov    (%rax,%rdx,4),%ecx
  0.13        mov    -0x60(%rbp),%eax
              movslq %eax,%rdx
              movslq %r12d,%rax
  4.62        imul   %rdx,%rax
  0.01        shl    $0x2,%rax
              add    -0x30(%rbp),%rax
              mov    -0x64(%rbp),%edx
  4.56        movslq %edx,%rdx
 58.67        mov    (%rax,%rdx,4),%eax
  0.00        imul   %ecx,%eax
  0.00        add    %eax,-0x5c(%rbp)
  4.58        addl   $0x1,-0x60(%rbp)
  8.76  554:  mov    -0x8c(%rbp),%eax
  4.66        cmp    %eax,-0x60(%rbp)
  0.01      ↑ jl     50e
  0.01        mov    -0x68(%rbp),%eax
              movslq %eax,%rdx
              movslq %ebx,%rax
              imul   %rdx,%rax
```

For the serial program, the program spends 60% of the time moving the values of the arrays into register for computation.

```
Performance counter stats for './charlie_mmm 1000 10 1.4 hopper':

     78640.504894      task-clock (msec)         #    0.999 CPUs utilized

  218,755,269,763      cycles                    #    2.782 GHz
     (50.01%)
  240,610,612,358      instructions              #    1.10  insns per cycle
     (75.03%)
   30,333,465,635      cache-references          #  385.723 M/sec
     (74.96%)
       14,444,796      cache-misses              #    0.048 % of all cache refs
     (75.02%)

     78.711539580 seconds time elapsed
```

The cache misses ratio of this program is low (0.048%). Hence, the processors are efficiently executing instructions on the system

Parallel Program by ltdang16: (with 4 threads)

- Foster Methodology
  - Partitioning
    - Command Line Argument Fetching: Change to getopt for a cleaner cmd grab
    - Initialize 3 matrices: Parallelized by assigning the each thread with an amount of rows. Each threads is responsible for initializing the rows that are assigned to them.
    - Matrix Multiplication: Also parallelized by assigning the each thread with an amount of rows. Each threads is responsible for multiplications of the rows that are assigned to them.
  - Communication:
    - The results of each rows are independent from each other => No communication is required from the threads.
    - The results of rows calculated in each threads are written in a shared matrix, but different rows (different addresses),, which will not required locks to protect the whole array.
  - Aggregation:
  - Mapping: Schedule(static) because multiplication process are the same, the input of the dot product are the same, since every value of Matrix A is 333333 and every value of Matrix B is 777777 => The workload done in each thread are the same.

Execution Time:

```
hopper$ ./lam_mmm 1000 10 1.4 hopper
# itemsPerDimension: 1000, repeats: 10, platform: hopper, coreSpeed: 1.400
# platform, totalBytes, wallTimeForAll, wallTimeForOne, MatrixKBPerSecond
hopper, 4000000, 25.848, 2.585, 1511.251
```

Program Profiling:

```
main.omp_fn.1  /cluster/home/ltdang16/cs360-paral/mmm/lam_mmm
  3.51 |        mov    -0x18(%rbp),%eax
  0.00 |        movslq %eax,%rbx
       |        mov    -0x28(%rbp),%rax
  3.58 |        mov    0x24(%rax),%eax
  0.15 |        cltq
  0.00 |        imul   %rbx,%rax
       |        lea    0x0(,%rax,4),%rbx
  3.54 |        mov    -0x28(%rbp),%rax
  0.03 |        mov    0x10(%rax),%rax
       |        lea    (%rax,%rbx,1),%rbx
  3.59 |        mov    -0x1c(%rbp),%eax
 44.35 |        cltq
  0.00 |        mov    (%rbx,%rax,4),%eax
  0.00 |        imul   %ecx,%eax
  3.59 |        add    %eax,-0x14(%rbp)
 11.92 |        addl   $0x1,-0x18(%rbp)
  2.98 |      ↑ jmpq   85
       | 13f:   add    $0x20,%rsp
       |        pop    %rbx
       |        pop    %r12
       |        leaveq
       |      ← retq
```

```
Performance counter stats for './lam_mmm 1000 10 1.4 hopper':

    105235.221780      task-clock (msec)         #    3.950 CPUs utilized

  292,315,169,307      cycles                    #    2.778 GHz
     (50.01%)
  350,893,623,130      instructions              #    1.20  insns per cycle
     (75.01%)
   60,490,307,057      cache-references          #  574.810 M/sec
     (74.98%)
       25,354,026      cache-misses              #    0.042 % of all cache refs
     (75.00%)

     26.641129361 seconds time elapsed
```

The cache misses ratio of this program is low (0.048%). Hence, the processors are efficiently executing instructions on the system