MultiCore Architecture and Parallel Programming Final Examination

Name:_____ ID:_____ Class:_____ Date:2014/12

1. List at least four techniques for cache optimization and make a brief explanation of each technique.(10')
   Merging Arrays
   Array transpose Loop Interchange
   Loop Fusion
   Loop blocking for matrix transposition
   Inter array padding
2. John benchmarks one of her parallel programs executing on 40 processors.She discovers that it spends 99 percent of its time inside parallel code.What is the scaled speedup of her program?(Tip: Amdahl's law) (10')
   Scaled speed up is 40*0.99+0.01=39.61
3. Briefly explain OpenMP Execution Model?(10')
   Fork-join model of parallel execution
   Begin execution as a single process (master thread)
   Start of a parallel construct:Master thread creates team of threads (worker threads)
   Completion of a parallel construct:Threads in the team synchronize -- implicit barrier
   Only master thread continues execution
   Implementation optimization: Worker threads spin waiting on next fork
4. Suppose we are going to speed the execution of the data clustering algorithm by using p processors to generate the D-dimensional vectors for each of the N documents.One approach would be to preallocate about N/p documents to each processor.Another approach would be to put the documents on a list and let processors remove documents as fast as they could process them.Discuss advantages and disadvantages of each approach.(10')
   The advantage of preallocation is that it reduces the overhead associated with assigning documents to idle processors at run-time. The advantage of putting documents on a list and letting processors remove documents as fast as they could process them is that it balances the work among the processors. We do not have to worry about one processor being done with its share of documents while another processor (perhaps with longer documents) still has many to process.
5. What is "False Sharing"?(5')
   A cache block contains more than one word,Cache-coherence is done at the block-level and not word-level.
6. Explain Snooping Cache Protocols.(5)
   The cores share a bus .Any signal transmitted on the bus can be "seen" by all cores connected to the bus.When core 0 updates the copy of x stored in its cache it also broadcasts this information across the bus.If core 1 is "snooping" the bus, it will see that x has been updated and it can mark its copy of x as invalid.
7. Briefly explain how a GPU generate pictures from some vertex information,and why we use unifed shader now? (10')
   1. Vertex processing, Clipping and rasterization, Fragment processing, Lighting was computed for each vertex, Frame buffer processing
   2. To elimate to work imbalance between Vertex and Pixel Processing,with unified shaders,we can dynamically adjust their numbers.
8. How many types of memory in CUDA device?Explain their difference and characteristics. (10')

Read/write per-thread
registers

Read/write per-thread
local memory

Read/write per-block
shared memory

Read/write per-grid
global memory

Read/only per-grid
constant memory

9. What is “divergence” in cuda, Use an example to explain how to avoid it? (10’)

- Main performance concern with branching is divergence
  - Threads within a single warp take different paths
  - Different execution paths are serialized in G80
    - The control paths taken by the threads in a warp are traversed one at a time until there is no more.
- A common case: avoid divergence when branch condition is a function of thread ID
  - Example with divergence:
    - `If (threadIdx.x > 2) { }`
    - This creates two different control paths for threads in a block
    - Branch granularity < warp size; threads 0 and 1 follow different path than the rest of the threads in the first warp
  - Example without divergence:
    - `If (threadIdx.x / WARP_SIZE > 2) { }`
    - Also creates two different control paths for threads in a block
    - Branch granularity is a whole multiple of warp size; all threads in any given warp follow the same path

10. Use Semaphore or Mutex to handle Read-Writer Problem with PThread.(One writer,several reader,readers can read simultaneously,readers are not allowed to read when writer is writing,writers are not allowed to write when reader(s) are reading.)(10’)

```
int readcount; // (initial value = 0)
semaphore mutex_rdcnt, r, w; // ( initial value = 1 )
//READER
        wait(r);
          wait(mutex_rdcnt);
           readcount++;
           if (readcount == 1)
             wait(w);
          signal(mutex_rdcnt);
        signal(r);

        // reading is performed

        wait(mutex_rdcnt);
          readcount--;
          if (readcount == 0)
            signal(w);
        signal(mutex_rdcnt);
//WRITER
        wait(r);
        wait(w);
        signal(r);
         // writing is performed
        signal(w);
```

11. Write a program to compute Pi.($\frac{\pi}{4} = \int_0^1 \frac{1}{1+x^2} dx$,you can choose one of these languages:OpenCL,MPI,CUDA)(10')

MPI:

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
double f( double );
double f( double a );
{
      return (4.0 / (1.0 + a*a));
}
int main( int argc, char *argv[])
{    int done = 0, n, myid, numprocs, i;
       double PI25DT = 3.14159265358979323846264 3;
       double mypi, pi, h, sum, x;
       double startwtime = 0.0, endwtime;
       int    namelen;
       char processor_name[MPI_MAX_PROCESSOR_NAME];
       MPI_Init(&argc,&argv);
       MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
       MPI_Comm_rank(MPI_COMM_WORLD,&myid);
       MPI_Get_processor_name(processor_name,&namelen);
       fprintf(stderr,"Process %d on %s\n",    myid, processor_name);
       n = 0;
       while (!done)
```

```
        {
            if (myid == 0)
            {
               if (n==0) n=100; else n=0;
                startwtime = MPI_Wtime();
            }
            MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
            if (n == 0)
                done = 1;
            else    {
                h   = 1.0 / (double) n;
                sum = 0.0;
                for (i = myid + 1; i <= n; i += numprocs)
                {
                    x = h * ((double)i - 0.5);
                    sum += f(x);
                }
                mypi = h * sum;
            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
                if (myid == 0)      {
                    printf("pi is approximately %.16f, Error is %.16f\n", pi, fabs(pi - PI25DT));
                    endwtime = MPI_Wtime();
                    printf("wall clock time = %f\n",                    endwtime-startwtime);
                }
            }
        }
        MPI_Finalize();
        return 0;
}
```

CUDA:

```
    __global__ void reducePI1 (float *d_sum, int num) {
   int id=blockIdx.x*blockDim.x+threadIdx.x;//线程索引
   int gid=id;  float temp;
    extern float __shared__ s_pi[];//动态分配，长度为 block 线程数
   s_pi[threadIdx.x]=0.0f;
    while(gid<num){
     temp=(gid+0.5f)/num;//当前 x 值
     s_pi[threadIdx.x]+=4.0f/(1+temp*temp);
   gid+=blockDim.x*gridDim.x;
   }
    for(int i=(blockDim.x>>1);i>0;i>>=1){
    if(threadIdx.x<i){
      s_pi[threadIdx.x]+=s_pi[threadIdx.x+i];
    }
     __syncthreads();
```

```
        }
        if(threadIdx.x==0)
        d_sum[blockIdx.x]=s_pi[0];
       }
        __global__ void reducePI2(float *d_sum,int num,float *d_pi){
       int id=threadIdx.x;
        extern float __shared__ s_sum[];
       s_sum[id]=d_sum[id];
       __syncthreads();
        for(int i=(blockDim.x>>1);i>0;i>>=1){   if(id<i)
         s_sum[id]+=s_sum[id+i];   __syncthreads();  }
        if(id==0){
        *d_pi=s_sum[0]/num;
       }
}
```