

CSCI-4320/6340 Assignment 3: Parallel Reduction: Blue Gene/L vs. CUDA

Christopher D. Carothers
Department of Computer Science
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180-3590

March 31, 2009

DUE DATE: 12 p.m/Noon, Tuesday, April 7th

1 Description

For this assignment you will be comparing your implementation of a parallel reduction on the Blue Gene/L vs. the existing implementation (already done for you) on CUDA. The Blue Gene details are:

- The reduction operation will be SUM on an array of integer values.
- The range of integer array sizes is: { **1048576, 2097152, 4194304, 8388608, 16777216, 33554432** }.
- All tests will use the SAME 128 processor (64 node) configuration.
- Assume each test array is EVENLY distributed over the 128 processors.
- Initialize each element of the array with it's local index number. (don't include initialization in the timing).
- Use the RTSC timer code for your timing measurement.
- Each processor will sum their local part of the integer array.

- Use the `MPI_Allreduce` to sum the partials and output your results. Please note, you will get a sum overflow most likely but that is fine.

To execute the CUDA reduction algorithm, execute the following command on AREA 51 (IP address is 72.224.56.37):

```
/usr/local/cuda/bin/linux/release/reduction --n=SIZE
```

Where `SIZE` is one of the array sizes above. You can examine the source code in:

```
/usr/local/cuda/projects/reduction
```

NOTE, you will have to execute the following shell command so the CUDA shared lib is loaded when you execute the reduction command above.

```
export LD_LIBRARY_PATH=/usr/local/cuda/lib/:$LD_LIBRARY_PATH
```

2 Timing Code

The following is a routine that you would include as “`rdtsc.h`”. RDTSC stands for the “read time-stamp counter” and it is an x86 or PowerPC assembly language instruction that returns a 64 bit number that is the number of cycles this machine is processed since the last boot-time. The C code for this is:

```
#ifndef __RDTSC_H_DEFINED__
#define __RDTSC_H_DEFINED__

#ifdef __i386__

static __inline__ unsigned long long rdtsc(void)
{
    unsigned long long int x;
    __asm__ volatile (".byte 0x0f, 0x31" : "=A" (x));
    return x;
}

#elif defined(__x86_64__)

// typedef unsigned long long int unsigned long long;

static __inline__ unsigned long long rdtsc(void)
{
    unsigned hi, lo;
```

```

__asm__ __volatile__ ("rdtsc" : "=a"(lo), "=d"(hi));
return ( (unsigned long long)lo)| ( (unsigned long long)hi)<<32 );
}

#elif defined(__powerpc__)

// typedef unsigned long long int unsigned long long;

static __inline__ unsigned long long rdtsc(void)
{
    unsigned long long int result=0;
    unsigned long int upper, lower,tmp;
    __asm__ volatile(
        "0:                                \n"
        "\tmftbu    %0                    \n"
        "\tmftb     %1                    \n"
        "\tmftbu    %2                    \n"
        "\tcmpw     %2,%0                  \n"
        "\tbn     0b                        \n"
        : "=r"(upper), "=r"(lower), "=r"(tmp)
        );
    result = upper;
    result = result<<32;
    result = result|lower;

    return(result);
}

#else

#error "No tick counter is available!"

#endif

#endif

```

Use the above macro `rdtsc`. To do things like:

```

unsigned long long start_time = 0;
unsigned long long finish_time = 0;

```

```
unsigned long long total_time = 0;

rdtsc( start_time );

for( i; i < MAX_WHATEVER; i++ )
{
    DO TEST
}

rdtsc( finish_time );

total_time = finish_time - start_time;
```

Note, I'll place a copy of of this in `rdtsc.h` on the Class website for you to download.

3 HAND-IN INSTRUCTIONS

THE DEADLINE FOR THIS ASSIGNMENT IS TUESDAY, APRIL 7th, 2009.

Write-up a short report that describes your Blue Gene implementation and graphs your Blue Gene and CUDA performance data for the different array sizes. Hand-in a hard copy of your report in class on Tuesday, April 7th. Also, please a copy of your code on AREA 51 in your account under the subdirectory `assignment3`.