COS597C: StreamIt_Cilk_Assignment

Due: December 14, 2010

Submission: please email to august@princeton.edu, dpw@princeton.edu, fengliu@princeton.edu and tpondich@princeton.edu. Put COS597C in the subject line.

1. StreamIt

Write pseudo code for a StreamIt MergeSort algorithm that takes an input N (given at compile time) that indicates how many integers to sort. The operation should be performed by a pipeline that takes the stream of N integers as input and writes a stream of N sorted integers. You may assume that the source and printer filters already exist. If you wish, you may draw the pipeline instead of writing pseudocode. Comment on what types of parallelism (task/data/pipeline) are exposed by the stages in your design. Comment on how the parallelism is more/less clear in this paradigm than in C. You can submit this in an archive with the rest of the assignment.

```
void->void pipeline MergeSort {

   add DataSource(N);

   add Sorter(N);

   add DataPrinter();

}


int->int filter Merger(int N) {

  // pseudocode here

}


int->int pipeline Sorter(int N) {

  // pseudocode here

}
```

BONUS: Implement your algorithm in StreamIt, test its performance, and comment on its scalability. You can get a version of StreamIt that works on hats and c2 from:

http://www.princeton.edu/~tpondich/streamit-2.1.1-modified.tar.gz.

(The version on the StreamIt webpage does not compile on 64 bit machines).

To install run the following:

wget http://www.princeton.edu/~tpondich/streamit-2.1.1-modified.tar.gz

tar xf streamit-2.1.1-modified.tar.gz

cd streamit-2.1.1

source ./setupenv.sh

./configure && make

To build a program:

cd $STREAMIT_HOME

source ./setupenv.sh

cd $PROGRAM_HOME

strc <Program>.str -o <Program> -cluster <Threads>

./<Program> -i <iterations>

You can find documentation here:

http://groups.csail.mit.edu/cag/streamit/shtml/documentation.shtml

2. Cilk

Implement a parallel merge algorithm using Cilk. To help you get started, I'm providing a sequential version of merge sort in the homework package (http://www.princeton.edu/~fengliu/cilk_assignment.tar.bz2), which includes a driver that measures the execution time of your implementation. Make sure your cilk sorted result is correct.

1) Analyze the work and depth of your algorithm. If the merge function were implemented sequentially, how would the work and depth of your algorithm change?
2) For various input sizes, measure the speedup of your implementation over the sequential merge sort (set the processor number as the number of physical cores). Report your measurement as one graph, and note the processor number you used. Does the measured speedup reflect the work/depth analysis of your algorithm, and predicted time according to Brent's theorem (http://qemfd.net/dankwiki/index.php/Brent%27s_Theorem)? Hypothesize why or why not.

Dowload the cilk compiler here:

http://supertech.csail.mit.edu/cilk/cilk-5.4.6.tar.gz

This compiler can get installed by running:
./configure && make

Then you can untar the homework package in the top directory of cilk-5.4.6.

You can find documentation here:

supertech.csail.mit.edu/cilk/manual-5.4.6.pdf