Meeting Notes and Assignment 7/3/19

Paul Scanlon and Matt Murdoch

Colorado State University, Research for Dr. Sanjay Rajopadhye, Summer 2019

Last week, we did zero calibration and found that the significant factor of noise is $1E(-5)$, so any times less than this should not be considered.

We also implemented matrix chain ordering in AlphaZ. A few tests were run and produced correct output. Unfortunately, we did not use the textbook to realize our implementation (Equation 15.7).

Meeting Points

We began with explaining the zero calibration and augmentation of the makefile that was done last week.

A ten micro second zero calibration time yields over 2000 operations lost at 2ghz.

We are trying to deduce equations that are computed by a program, it is the reverse direction as convention. The compiler has this as its job. The point of AlphaZ is that you don't need a sequential language as an input. The compiler generates loops/sequences based on dependencies.

For matrix chain ordering, 'l' is redundant based on the C implementation. Alphaz can generate many different sequential iteration implementations.

Usually, on the department machines, you get a few miliseconds of dedicated scheduling, so you cannot be abusively swapped out. Check to see if this scheduling worked properly.

OSP is a class of problems that does parenthesizing. It can be over any cost. Here we are taking the minimum cost but it could be useful to take the max.

BPMain is similar to BPMax and has been verified by Hamid.

Assignment

New experiment: interrupt yourself during the testing process on the same machine to see if swapped out scheduling is what happened. Try to get two instances of testing matrix chain ordering to compete.

New protocol for data collection on timing - Make sure you will not be interrupted, etc.

Uncomment for step 1, re-comment and uncomment others for step 2.

1) start with the regular compiler script approach to generate .c, wrapper, and makefile.
2) make all: plain check verify verify-rand random. Be sure to implement random alone.
3) Copy .c into verify.c once we see it is correct. Change all "$eval_x$" to "$eval_{xv}erify$" for compatibility. Call it $BPMain_Verify$.
4)Test on a CS machine like eel.cs.colostate.edu or lamborghini@colostate.edu and make Test with the input 2 2 12344321 which gives 123454764321. Then do verify-rand with bigger and bigger inputs.
5) test with testing strategy, 50, 24, 4. (s2)
6) test with more. (bg)
7) uncomment *setSpaceTimeMap* which tells the order that the points need to be evaluated in.

For mcp, X=M

Diag = 0 to n, so we can uniquely identify points in the triangle with D and i where $j - i = D$, and $i1, j1- > j1 - i1$.

The number of loops must be the same as number of dimensions.

Target mapping $(i, j)- > j - i, j$ is a transformation The number of terms on the right must equal the left.

Experiment with different scheduling options because the code generated by *write c* is very inefficient.

Note that the background continues when you log out. The timer script can be run on multiple computers.

Be sure to nice -19 tester1.sh.

See if what they did on the posters is correct/legal for different scheduling.

After OSP, we will see what can be done to improve BPMain and how triangular matrix multiplication is going to relate. Then we can make a library for *max* and *plus* for matrix multiplication.