BACKSLASHX90

Project Proposal

By Nicolas Broeking and Joshua Rahm

PROJECT PROPOSAL

Group

The members of backslashx90 include Nicolas Broeking and Joshua Rahm. Both Broeking and Rahm are students in the MS/BS program.

Problem

A major problem in software engineering is how can we computer parallel problems as efficiently as possible. Many problems in computer science can be parallelized but a lot of the time the overhead for creating multiple threads isn't worth the time for the programmer. Another problem with creating parallel programs is the architecture that they run on. In recent years there have been multiple platforms for programs to run on. In recent years with the technologies like OpenCL, OpenGl and other such multicore architectures the over to create parallel applications has increased. This is where Implicit parallelism comes in to play. A lot of research has been done lately in the field of implicit parallelism. Implicit parallelism is when a compiler can analyze the code and find sections that can be paralyzed and take the overhead away from the programmer.

Proposal

We propose to extend the p4 language to include the codeword for. During the compilation of programs we will analyze the body of the for statement attempt to determine if the block can be paralyzed. It it is then the assembly that we produce will use threads to complete those lines of code. The code below shows an example of this.

```
x = [1,2,3,4,5,6,7]
for y in x:
        y += digitInPi(5000)

print x
```

This simple program takes the list x and adds each of the numbers against the 5000 digit of pi. Computationally this can take a lot of time. By introducing implicit parallelism we can then paralyze the y += digitInPi(5000) line agains all the members in x. This drastically increases the run time of the program.

MORE EXAMPLE PROGRAMS

Approach

In order to solve this problem we will add an optimization phase to our chain. This phase will analyze the Intermediate asm language and determine if the code can be optimized. The new asm will then contain instructions to paralyze the blocks of code.
Our project is based off a paper published in ACM Computing Surveys called, Partitioned Global Address Space Languages. The paper focus on creating Partitioned Global Address Space Languages in order to apply optimizations to the code. A Partioned Global Address Space Language is a parallel programming model that "aims to improve programmer productive while aiming for high performance." We intend to implement their programming model to achieve implicit parallelism within our compiler.

Evaluation

To evaluate our program we will consider twenty test programs. Each of these programs will be catered to implicit parallelism. We will compare the execution time of our new compiler against the hw6 course compiler and if we are able to measure a speedup in execution time then we will have succeeded in optimizing our compiler. The Partitioned Global Address Space Languages paper provides what they believe to be good examples of where implicit parallelism can be used. Our tests will implement their examples and show the performance gain of implicit parallelism.

Sources

Mattias De Wael, Stefan Marr, Bruno De Fraine, Tom Van Cutsem, Wolfgang De Meuter. Partitioned Global Address Space Languages. ACM Computing Surveys, Association for Com- puting Machinery (ACM), 2016, pp. 29. <hal-01109405>

BUDGET

Ut vehicula nunc mattis pede

Curabitur labore. Ac augue donec, sed a dolor luctus, congue arcu id diam praesent, pretium ac, ullamcorper non hac in quisque hac. Magna amet libero maecenas justo.

Description	Quantity	Unit Price	Cost
Item 1	55	$ 100	$ 5,500
Item 2	13	$ 90	$ 1,170
Item 3	25	$ 50	$ 1,250
Total			**$ 7,920**