`-O3` should imply `--auto-parallelism`

Paul Bone

January 25, 2012

1 Motivation and Background

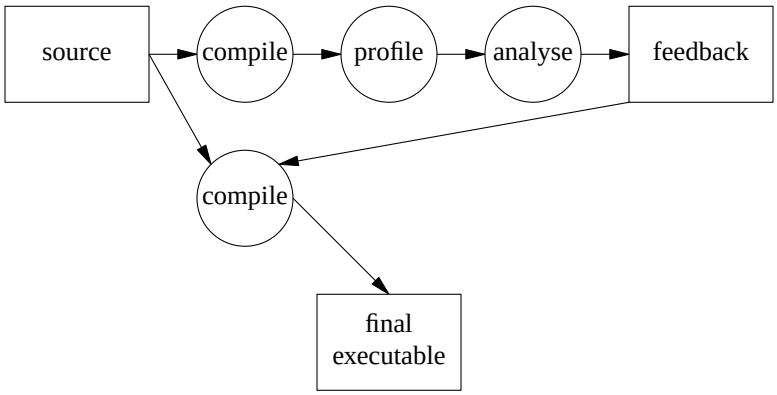
Multicore computing is here to stay, and to take advantage of multiple cores we need to write parallel programs. But doing so in imperative programming languages is painful — much like using Perl :-).

Declarative languages make this much easier and safer, but deciding where to introduce parallelism can still be a significant problem.

We're looking forward to a future where this is handled automatically by the compiler; so that the programmer thinks about it no more than they currently think of register allocation.

```
mmc --make -O3 --auto-parallelize parallel_project
```

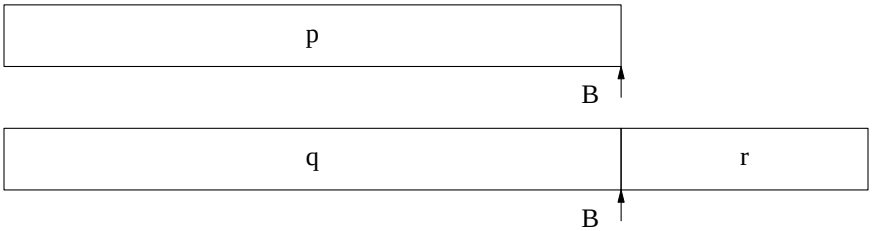
We're using Mercury, a pure declarative language with eager evaluation and a *very* good profiler. Mercury's profiler helps us collect information about the performance of different parts of a user's program. This data is used by an analysis tool to advice the compiler about what should be parallelized.



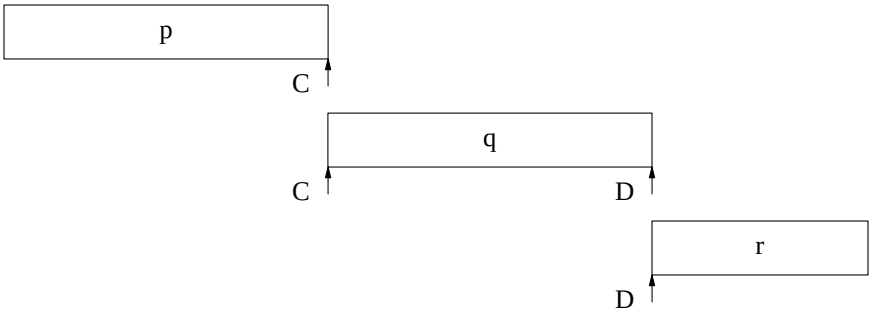
2 Overlap analysis

Because we handle dependent and parallelism we must take into account the dependencies between parallel tasks. Such dependencies affect how much of the parallel computations can 'overlap' their executions. We like to visualize this with diagrams, blocks represent computations and the x axis represents runtime. If a shared variable is produced or consumed in the middle of a block that block is split at that point as necessary. This work was presented as a full paper at ICLP 2011.

Optimal overlap:



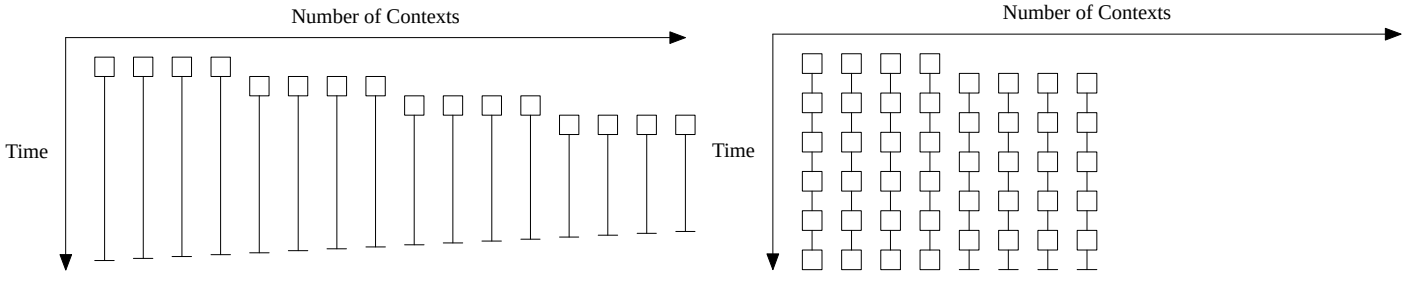
Pessimal overlap:



3 Loop Control

Declarative programmers are trained to write tail recursive code so that it uses a constant about of stack space. Parallelizing a recursive call with some other code prevents tail recursion and, more significantly consumes a lot of memory. This is because of the order we have to handle operations in so that we can provide dependent and parallelism without having to worry about deadlocks.

The digrams show a reprsentation of how stack space both without and with loop control.



4 Vote for me!

Please come and see my Loop Control presentation in Declarative Aspects of Multicore Programming on the 28th and/or read the paper. The tables of results I should in my lightning talk are available in the paper.

Thank you.