The DIRECT Project

Delaware Interprocedural REgion-based Compiler Toolset

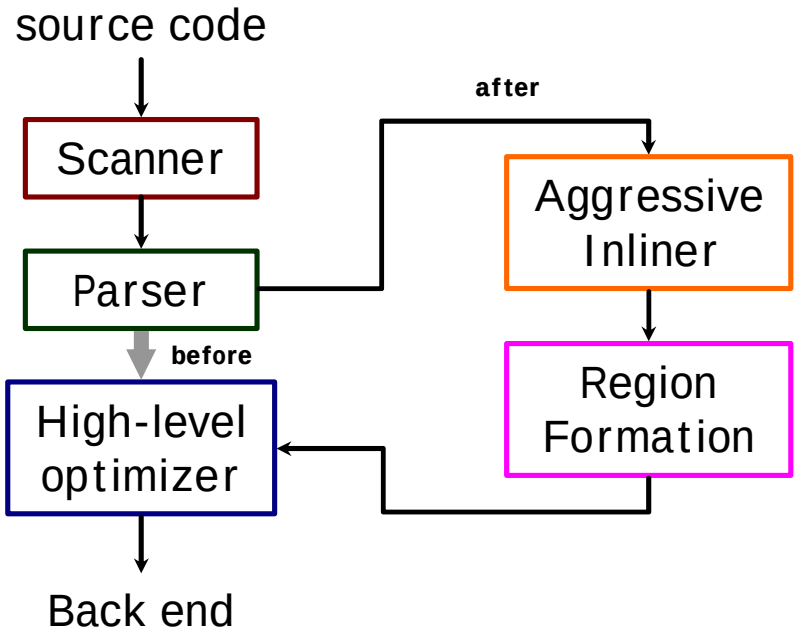
Directing the interaction between inlining and region-formation

Tom Way
Ben Breech
Wei Du
Matt Bridges
Ves Stoyanov
Lori Pollock

Department of Computer & Information Sciences
University of Delaware, Newark, Delaware

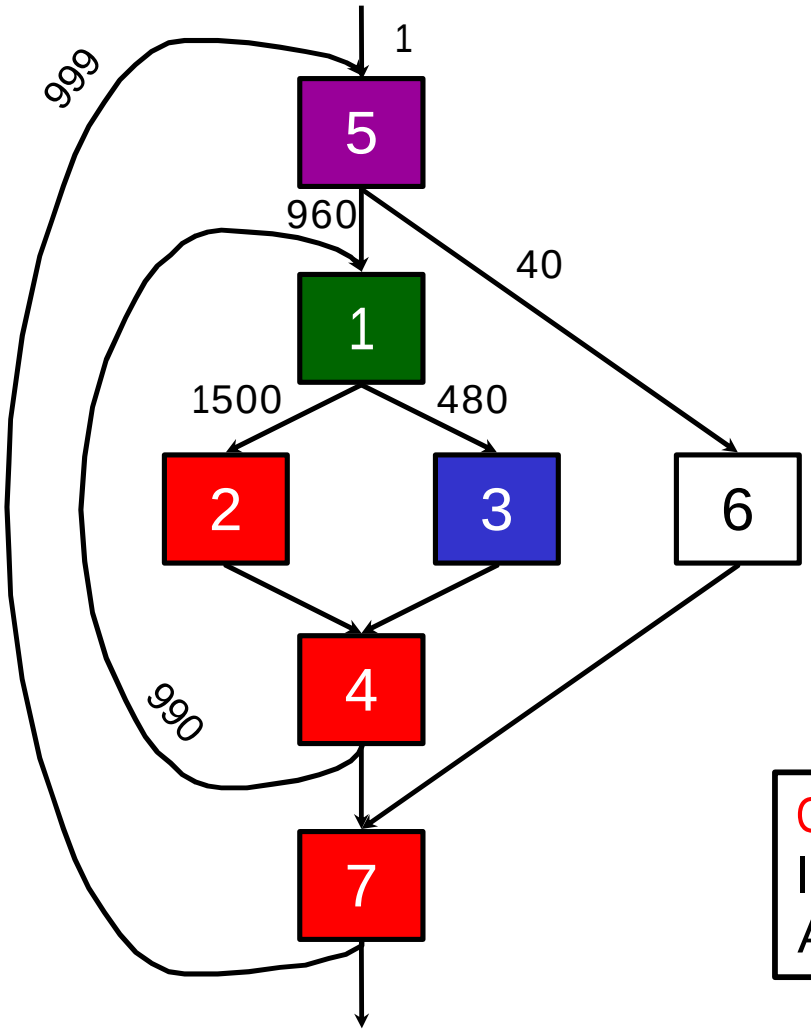
What is Region-based Compilation?

The Compiler:



- Repartitions a program into “regions”, groups of related basic blocks.
- Uses profiling info to select more “important” blocks to include, while weeding out others.
- Wraps up regions to look like functions, so rest of compiler is unchanged.

What happens



The Algorithm:

- Select seed
- Add successors
- Add predecessors
- Add all desirable successors

Criteria: block should be at least 50% as frequent as seed AND current block

So what's wrong with that?

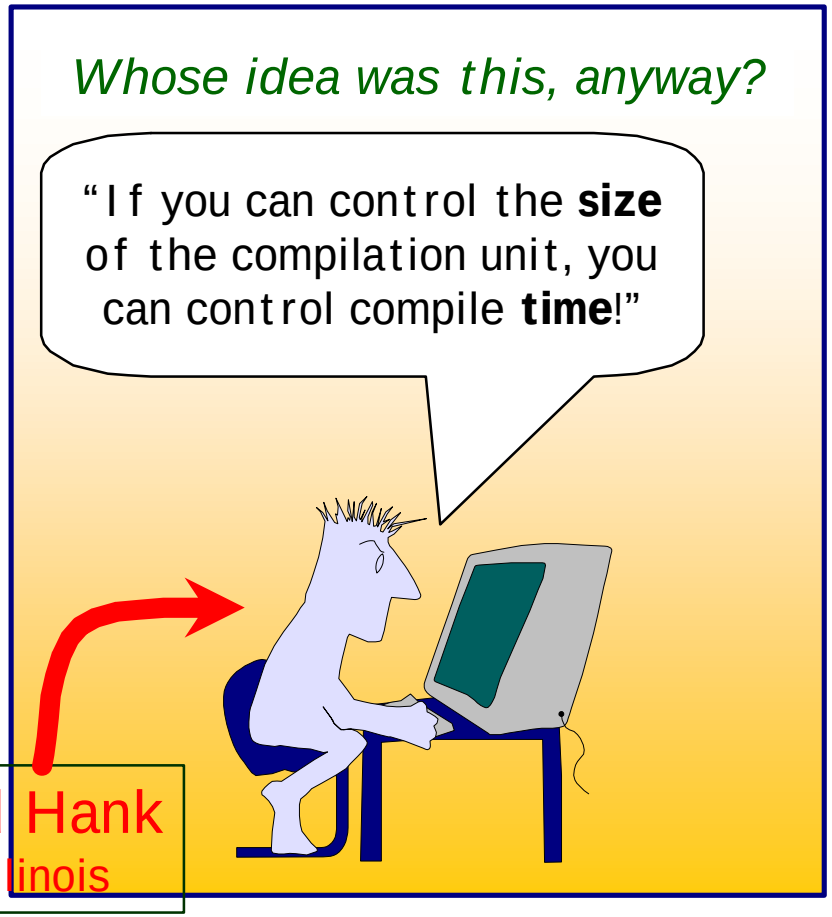
The good news: Controls the size of unit of compilation

The bad news: We inlined the program aggressively…

So the whole program (pretty much) is in memory for the duration of region formation and optimization!

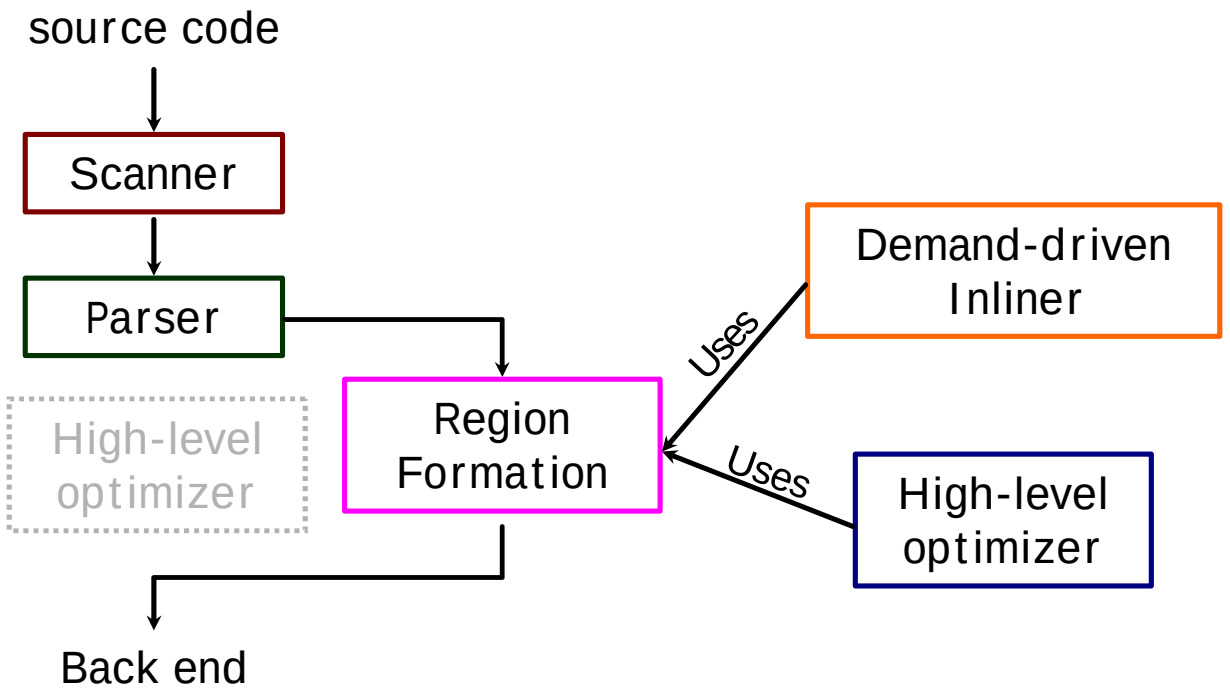
Scalable? Don't think so!

Leads to better scheduling on ILP & VLIW machines



DIRECT Region Formation

Our idea: Integrate inlining and optimization



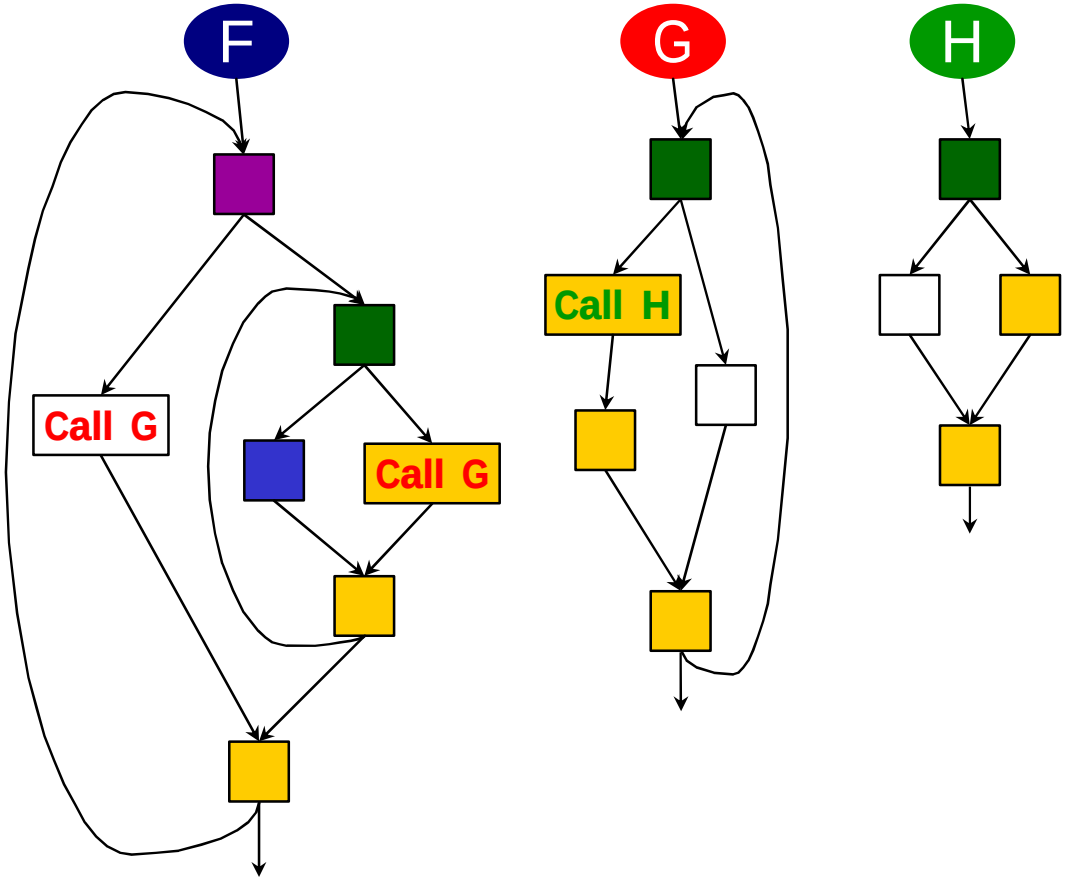
Can you say "interprocedural"?

By doing inlining and optimization as you go, things are much more scalable

Interprocedural Region Formation

Improved Algorithm:

- Select local seed
- Add local successors
- Recurse at callsite
- Finish local successors
- Local predecessors (recurse at call)
- Local desirable successors (recurse)



DIRECT Project: Overview

Trimaran 2.0 Compiler

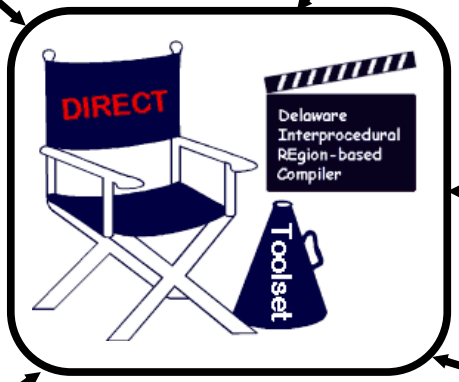
EPIC & ILP research compiler from Univ. of Illinois, NYU & HP

Region formation module

Uses demand-driven inlining to improve scalability and enable interprocedural region-formation

VIPeR region visualizer

Research tool to experiment with interprocedural regions



Demand-driven inliner

Replaces aggressive inliner for improved memory usage

Cloning tool with path-spectra profiling

Provides call-site specific profiling for better decision-making

Custom analysis tools

Suite of Perl scripts to gather variety of statistics from intermediate code and simulator output

Demand-driven Inliner (Wei Du)

The Present

- Replaces aggressive inliner
- Called on-demand at callsite during region formation
- Greatly reduces compile-time memory requirements
- Operates on Trimaran's low-level interm. code (Lcode)

The Future

- Extend in future to demand-driven partial inliner
- Partial inlining is naturally enabled through interprocedural region-based compilation

Cloning Tool (Ves Stoyanov)

The Present

- Instruments Trimaran's high-level interm. Code to output runtime trace
- Collects runtime path-profiles on a per-callsite basis
- Creates path spectra, a per-callsite set of path profiles

The Future

- Use path spectra to guide cloning decisions
- Research using path spectra to help with demand-driven inlining decisions
- Enable call-site specific optimizations

VIPER Tool (Matt Bridges)

Visualizing InterProcEdural Regions

The Present

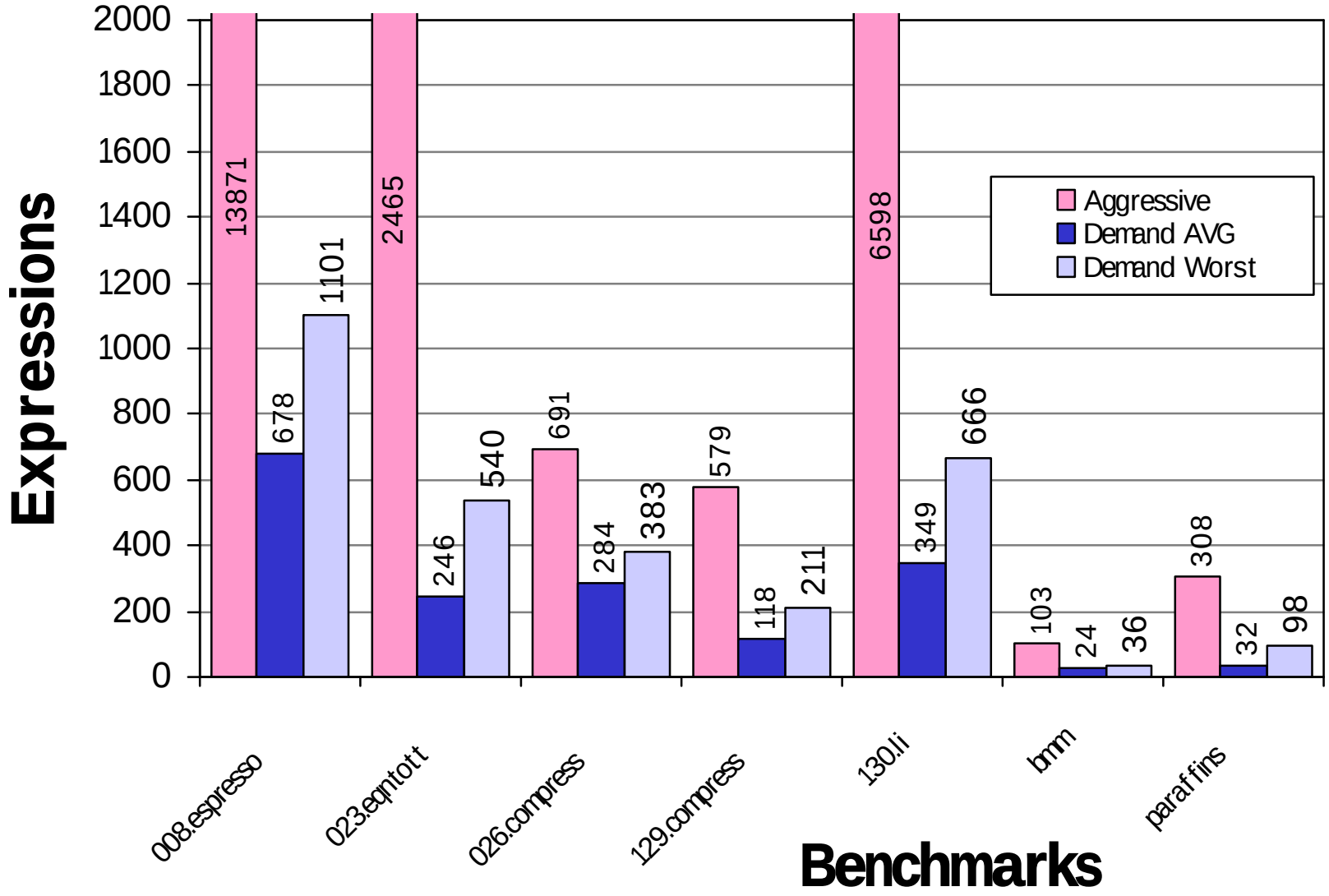
- Valuable research tool for experimenting with region-formation heuristics
- Graphically displays interprocedural regions
- Translates Trimaran's low-level Lcode to *daVinci* graphing code
- Implemented in Perl & Java

The Future

- Generate more region statistics
- Extend annotation abilities
- Improving linkage between source code viewer to graphical viewer

Discover the future now at the VIPER poster!

Sample Results:
Compile-time Memory Usage



DIRECT Project… Wassup?

What we know

- Reduces memory overhead over old method
- Execution times already at least as good as old method
- Improved compile-time anticipated once demand-driven inlining complete
- Developing new region formation heuristics should improve performance

What's next

- Integrate Demand-driven inlining module into compiler
- Integrate Cloning tool into region formation process
- Use VIPER tool to help develop better region formation heuristics
- Goal: more scalable compilation time and memory usage
- Goal: better performance on ILP & VLIW architectures