GCC as a Research Tool

Diego Novillo
dnovillo@google.com

Google™

GROW'10
Pisa, Italy
23 Jan 2010

GCC for research? Really?

Google

- Old
  - Hard to learn
  - Messy internals
  - Fast moving target
- However
  - Industrial strength
  - Widely deployed and used
  - Actively supported

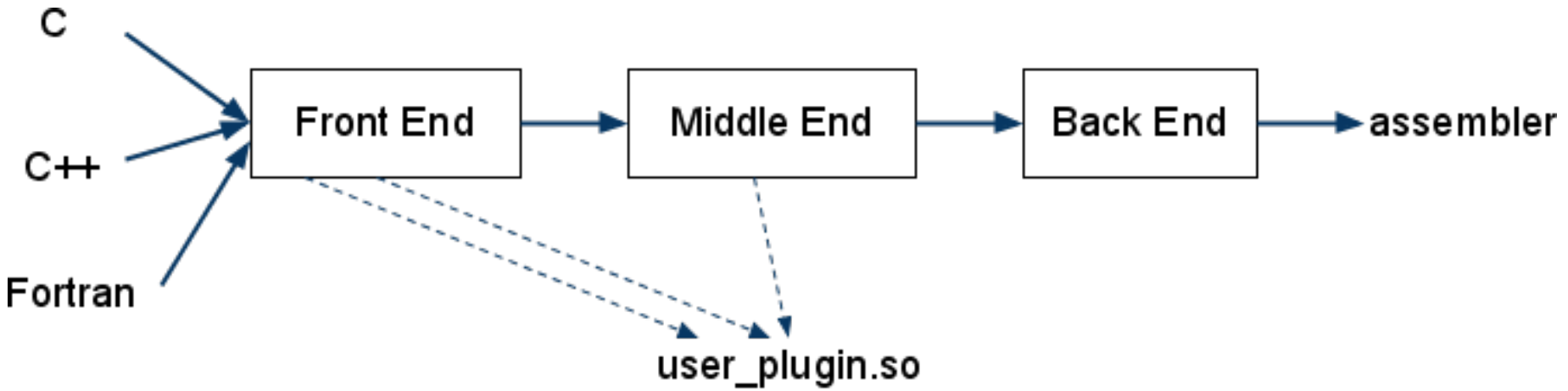
Where is GCC today?

Google

- Feature rich
  - High and low level optimizations
  - Solid support for most popular architectures
- Growing set of features
  - Loadable modules
  - High-level loop optimizations
  - Transactional memory
  - Debugging optimized code
  - Whole program optimization

Plugins

Google



- Interact with the parser
- Add new optimization passes
- Examine intermediate representation
- Implement custom analyses and checks
- Coding guidelines
- Semantic analysis on special code (e.g. locking analysis)

Plug-in Support

Google

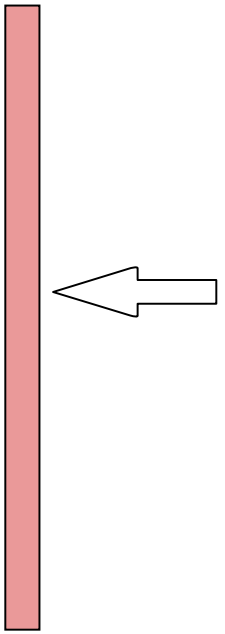
- Extensibility mechanism to allow 3rd party tools
- Wrap some internal APIs for external use
- Allow loading of external shared modules
- Versioning scheme prevents mismatching
- Useful for
  - Static analysis
  - Experimenting with new transformations

Optimizing Very Large Programs

Google

f1.C

```
foo()
{
   for (;;) {
     ...
     x += g (f (i, j), f (j, i));
     ...
   }
}
```



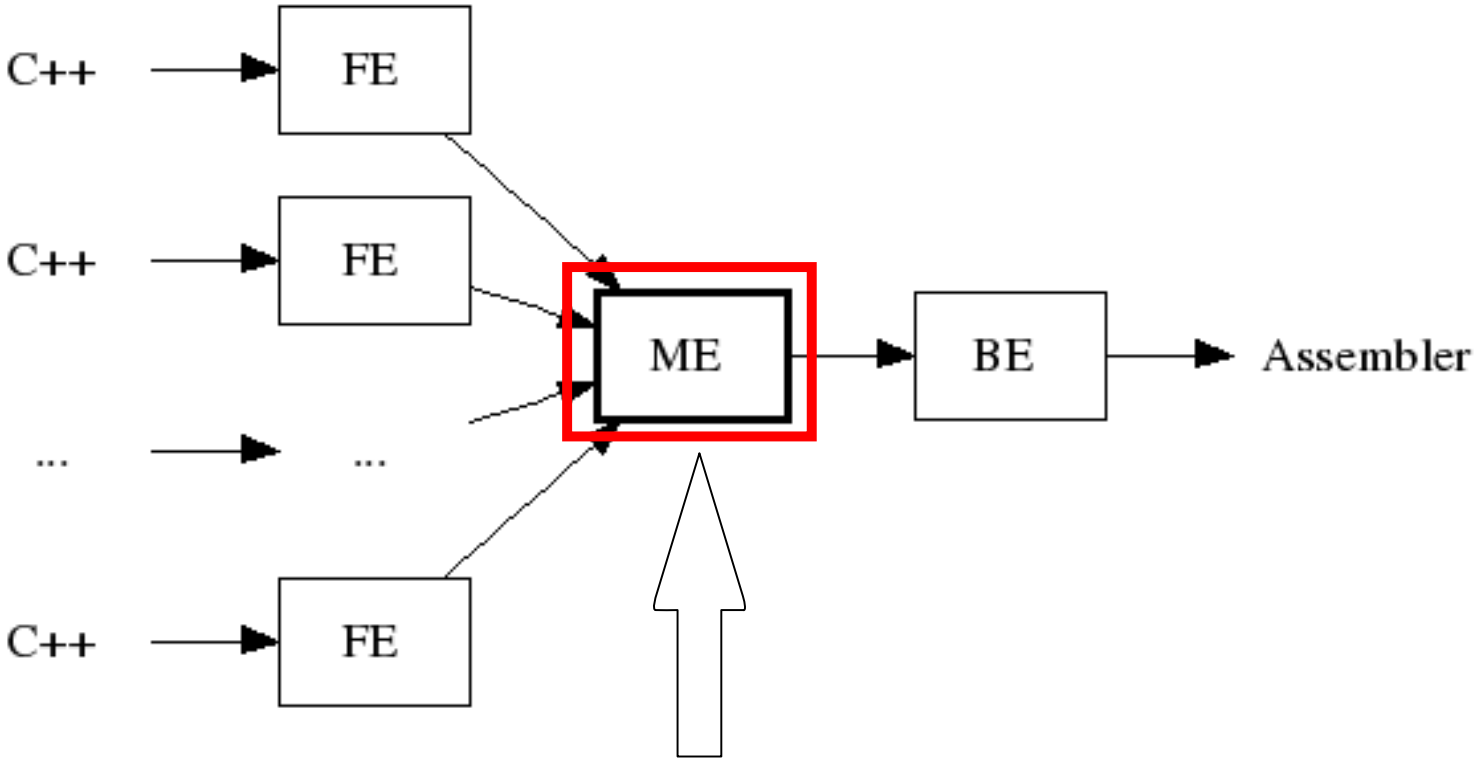
f2.C

```
float f(float i, float j)
{
  return i * (i - j);
}

float g(float x, float y)
{
  return x - y;
}
```

- Optimizations are limited by the amount of code that the compiler can see at once
- Current technology only works across one file at a time
- Compiler must be able to work across file boundaries

Optimizing Very Large Programs



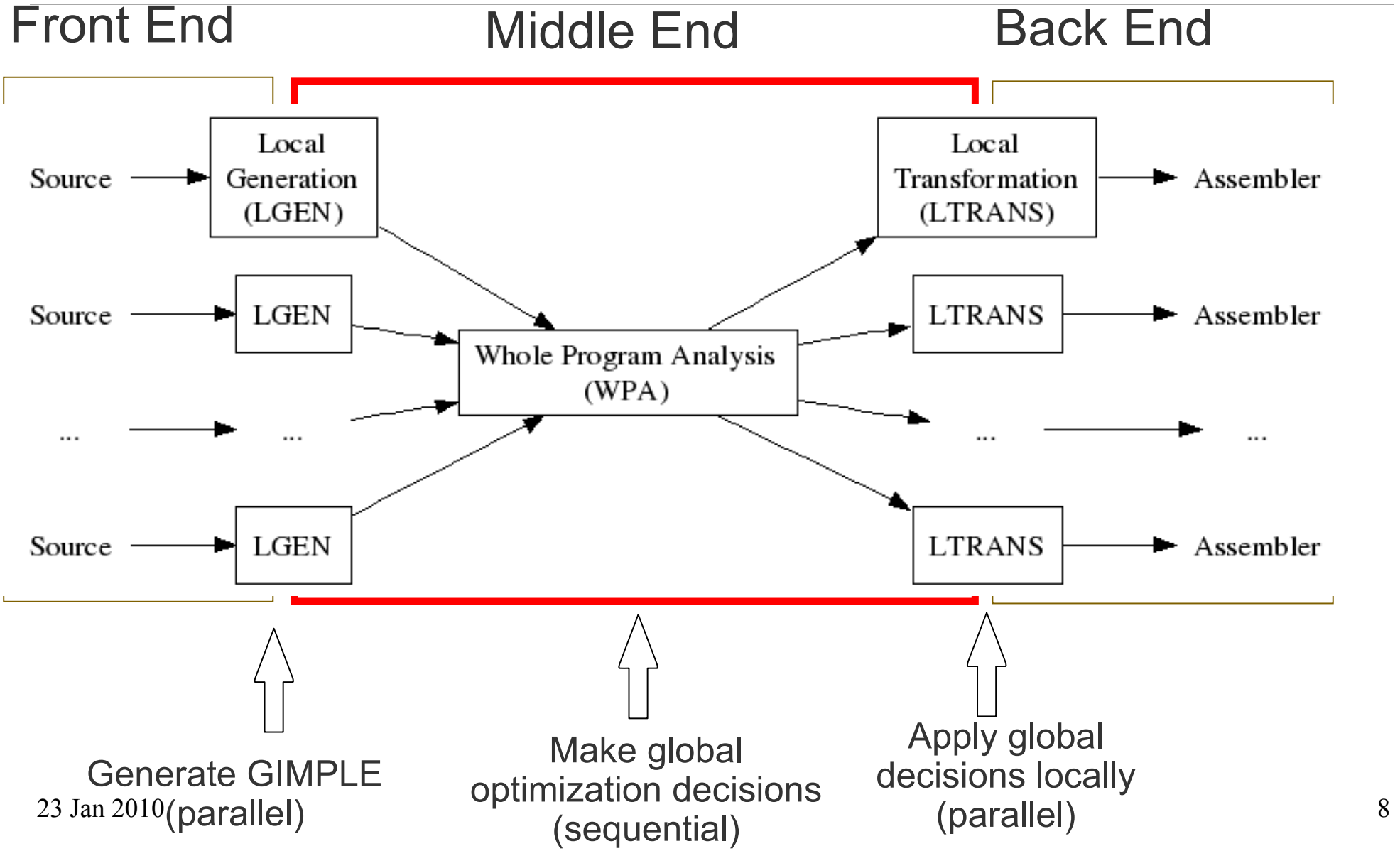
Problem

Thousands of files, millions of functions, tens of gigabytes

Massive memory/computation complexity for a single machine

WHOPR Architecture

Google



Lightweight IPO

Google

LIPO Key Idea:

- Move IPA at end of training phase in FDO, into the binary
- Augment profiles, add IPA analysis results
- During optimization build, use IPA data, read in additional modules, enable inlining, indirect call promotion, etc.

Benefits:

- No more writing of compiler IR to disk, less resources
- Eliminate monolithic link-type IPO
- Reuses existing intra-module IPO
- Minimize code re-generation
- Easier debugging, easier debug info generation

Where should GCC go?

Google

- Infrastructure improvements
  - Increased modularization
  - Attract new developers
  - Improve maintenance
- Stability
  - Many new features
  - Need to smooth out rough edges

Incremental Compilation

Google

- Speed up edit-compile-debug cycle
- Speeds up ordinary compiles by compiling a given header file “once”
- Incremental changes fed to compiler daemon
- Incremental linking as well
- Side effects
  - Refactoring
  - Cross-referencing
  - Compile-while-you-type (e.g., Eclipse)

Dynamic Optimization Pipeline

Google

- Phase ordering not optimal for every case
- Current static ordering difficult to change
- Allow external re-ordering
  - Ultimate control
  - Allow experimenting with different orderings
  - Define `-On` based on common orderings
- Problems
  - Probability of finding bugs increases
  - Enormous search space