redhat

GCC – An Architectural Overview

Diego Novillo

dnovillo@redhat.com

Red Hat Canada

OSDL Japan - Linux Symposium

Tokyo, September 2006

Topics

1. Overview
2. Development model
3. Compiler infrastructure
4. Current status and future work

Overview

- Key strengths
  - Widely popular
  - Freely available almost everywhere
  - Open development model
- However
  - Large code base (2.2 MLOC) and aging (~15 years)
  - Difficult to maintain and enhance
  - Technically demanding
- Recent architectural changes bring hope

Development Model

Development Model

- Project organization
  - Steering Committee → Administrative, political
  - Release Manager → Release coordination
  - Maintainers → Design, implementation
- Three main stages (~2 months each)
  - Stage 1 → Big disruptive changes.
  - Stage 2 → Stabilization, minor features.
  - Stage 3 → Bug fixes only (driven by bugzilla, mostly).

Development Model

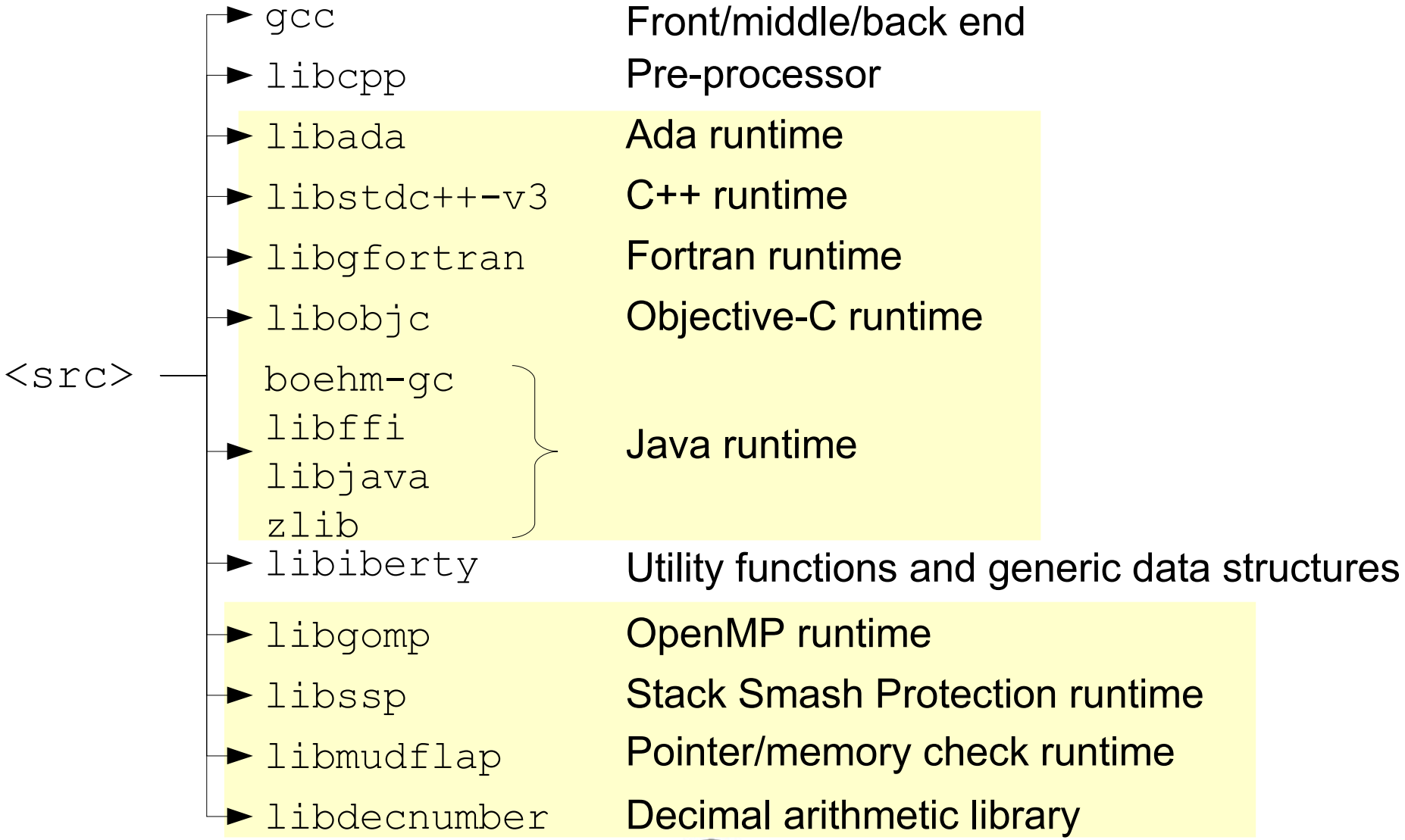
- Major development is done in branches
  - Design/implementation discussion on public lists
  - Frequent merges from mainline
  - Final contribution into mainline only at stage 1 and approved by maintainers
- Anyone with SVN access may create a development branch
- Vendors create own branches from FSF release branches

Development Model

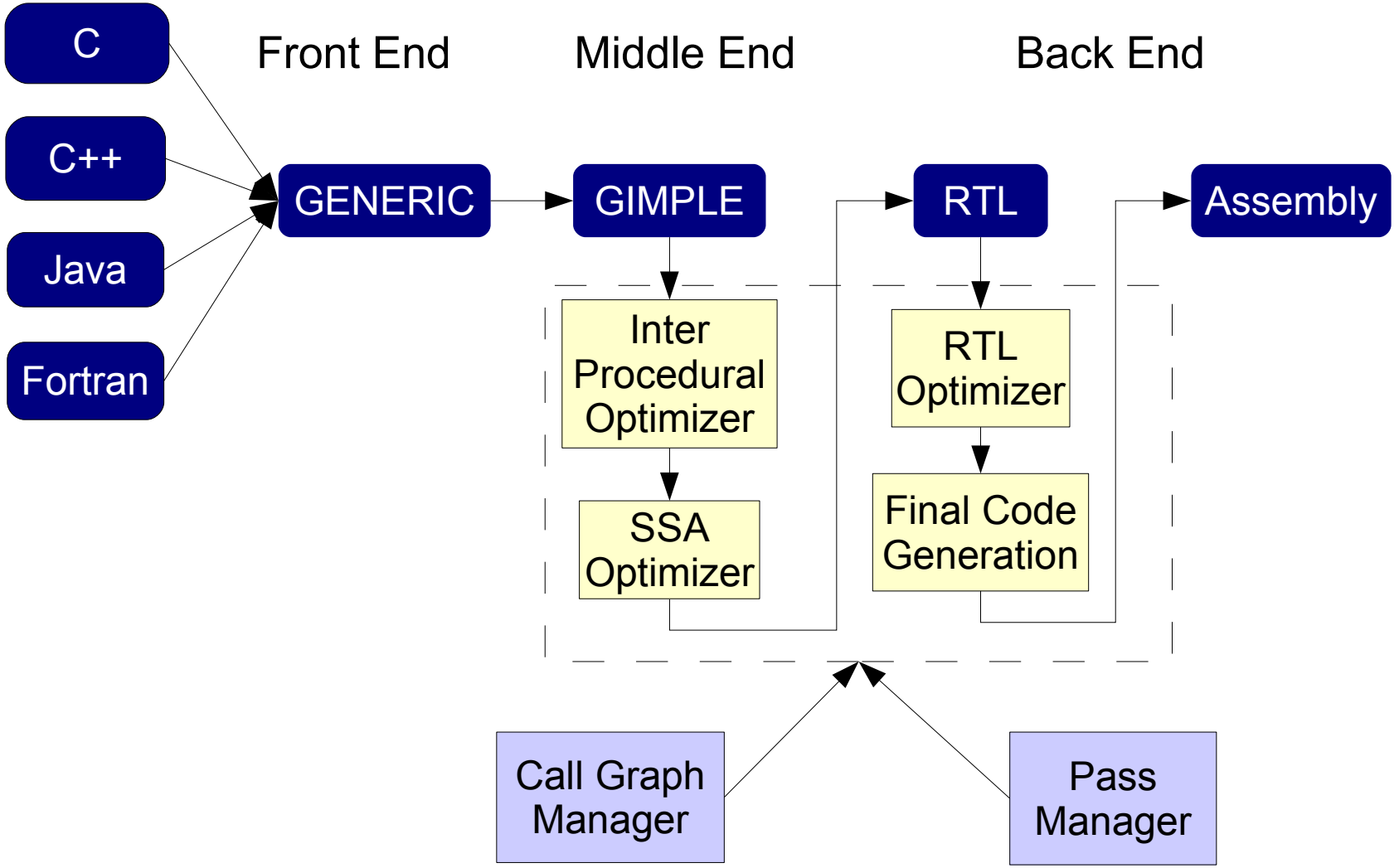
- All contributors must sign FSF copyright release
  - Even if only working on branches
- Three levels of access
  - Snapshots (weekly)
  - Anonymous SVN
  - Read/write SVN

Compiler Infrastructure

Source code


<src>
gcc
Front/middle/back end
libcpp
Pre-processor
libada
Ada runtime
libstdc++-v3
C++ runtime
libgfortran
Fortran runtime
libobjc
Objective-C runtime
boehm-gc
libffi
libjava
zlib
Java runtime
libiberty
Utility functions and generic data structures
libgomp
OpenMP runtime
libssp
Stack Smash Protection runtime
libmudflap
Pointer/memory check runtime
libdecnumber
Decimal arithmetic library

Compiler pipeline



GENERIC and GIMPLE

GENERIC

```
if (foo (a + b,c))
  c = b++ / a
endif
return c
```

High GIMPLE

```
t1 = a + b
t2 = foo (t1, c)
if (t2 != 0)
  t3 = b
  b = b + 1
  c = t3 / a
endif
return c
```

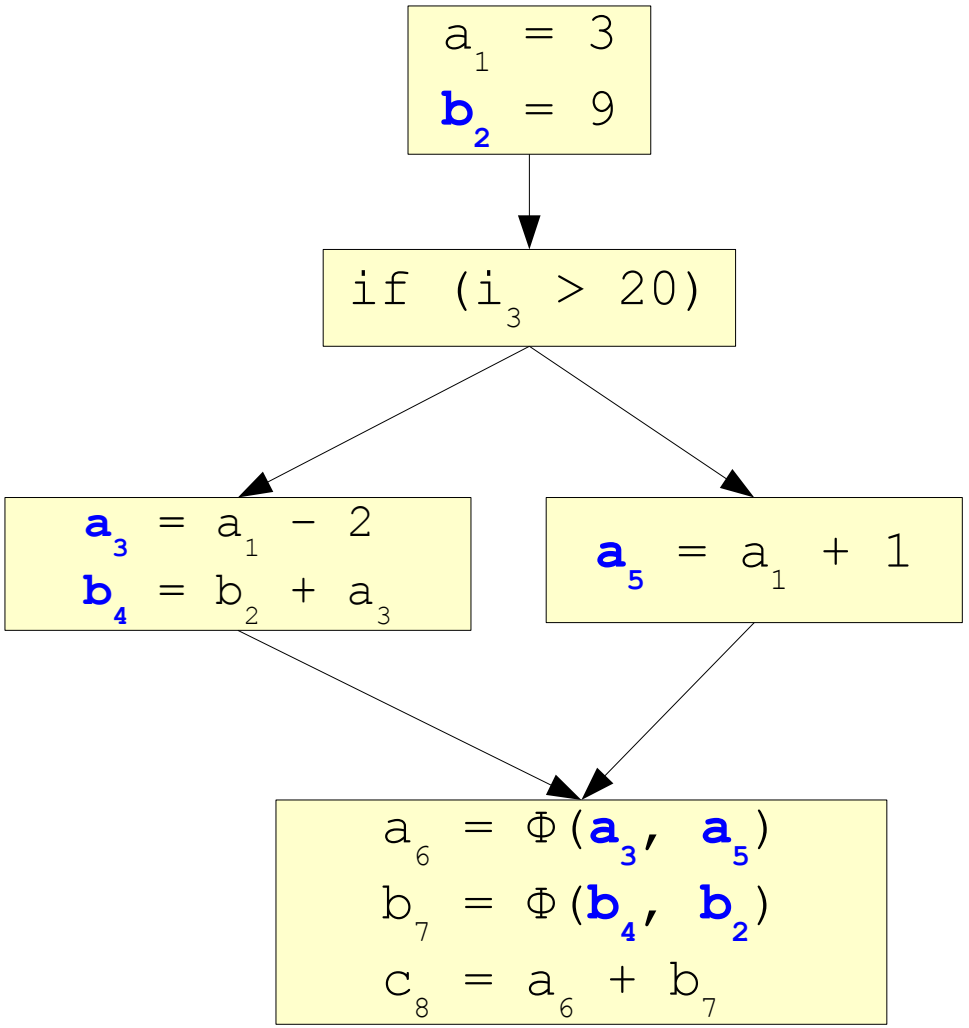
Low GIMPLE

```
t1 = a + b
t2 = foo (t1, c)
if (t2 != 0) <L1,L2>
L1:
t3 = b
b = b + 1
c = t3 / a
goto L3
L2:
L3:
return c
```

SSA Form

Static Single Assignment (SSA)

- Versioning representation to expose data flow explicitly
- Assignments generate new versions of symbols
- Convergence of multiple versions generates new one (Φ functions)



SSA Optimizers

- Operate on GIMPLE IL
- Around 100 passes
  - Vectorization
  - Various loop optimizations
  - Traditional scalar optimizations: CCP, DCE, DSE, FRE, PRE, VRP, SRA, jump threading, forward propagation
  - Field-sensitive, points-to alias analysis
  - Pointer checking instrumentation for C/C++

RTL

- Register Transfer Language
- Assembler for abstract machine with infinite registers

```
b = a - 1
```



```
(set (reg/v:SI 59 [ b ])
     (plus:SI (reg/v:SI 60 [ a ]
                (const_int -1 [0xffffffff]))))
```

RTL

- Abstracts
  - Register classes
  - Memory addressing modes
  - Word sizes and types
  - Compare-and-branch instructions
  - Calling conventions
  - Bitfield operations
  - Type and sign conversions

RTL Optimizers

- Operate closer to the hardware
  - Register allocation
  - Scheduling
  - Software pipelining
  - Common subexpression elimination
  - Instruction recombination
  - Mode switching reduction
  - Peephole optimizations
  - Machine specific reorganization

Current Status and Future Work

Current Status

- New Intermediate Representations decouple Front End and Back End
- Increased internal modularity
- Lots of new features
  - Fortran 95, mudflap, vectorizer, OpenMP, inter/intra procedural optimizers, stack protection, profiling, etc.
- Easier to modify

Future Work

- Static analysis support
  - Extensibility mechanism to allow 3rd party tools
- Link time optimizations
  - Write intermediate representation
  - Read and combine multiple compilation units
- Dynamic compilation
  - Emit bytecodes
  - Implement virtual machine with optimizing JIT

Contacts

- Home page http://gcc.gnu.org/
- Wiki http://gcc.gnu.org/wiki
- Mailing lists
  - gcc@gcc.gnu.org
  - gcc-patches@gcc.gnu.org
  - gcc-help@gcc.gnu.org
- IRC
  - irc.oft.net/#gcc