LLVM on IBM POWER processors
A progress report

Dr. Ulrich Weigand
Senior Technical Staff Member
GNU/Linux Compilers & Toolchain

Date: Apr 29, 2013

LLVM on IBM POWER processors
A progress report

and System z

Dr. Ulrich Weigand
Senior Technical Staff Member
GNU/Linux Compilers & Toolchain

Date: Apr 29, 2013

Agenda

- LLVM on IBM server processors
- Contributions to PowerPC back end
- New SystemZ back end
- Some observations on LLVM vs. GCC
  from a back-end developer's perspective

My background

- **IBM Linux Technology Center**
  - 2000: Toolchain for IBM mainframe (S/390 / System z)
  - 2005: Toolchain for Cell Broadband Engine
  - 2009: Debugger for IBM OpenCL SDK
  - 2010: Linaro: Toolchain for ARM
  - 2012: LLVM for POWER and System z
- **GNU Compiler & Toolchain**
  - GCC back-end maintainer for s390 and spu
  - GDB global maintainer

LLVM on IBM server processors

IBM's Linux Technology Center

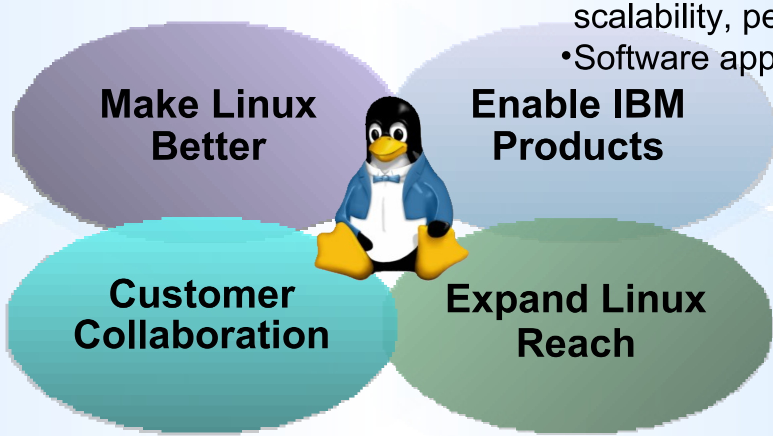
Enhancing Linux capabilities, driving Linux adoption

IBM contributes to the community

- IBM developers contributing to 100+ Linux and Open Source projects
- Develop closely with Red Hat and Novell
- Developers sharing technical knowledge on http://planet-ltc.org

IBM supports Linux as a Tier 1 OS

- All IBM Systems, SW, and Middleware run on and are certified for Linux
- Driving performance toward parity with IBM's own operating systems
- Making contributions in security, RAS, scalability, performance, management
- Software appliances



IBM collaborates with customers

- Specialized and very detailed knowledge of IBM Systems and Software
- The LTC works with customers on unique proof of concept projects
    - Scale Out File Services (SOFS)
    - Real time Linux and Java

IBM enables Linux for new markets

- Working with groups such as the Linux Foundation to address new workloads
- Expanding and providing capabilities for:
    - Blue Cloud Computing
    - SOA / Web 2.0 / SaaS
    - Distributed computing and HPC
    - IBM Smart Analytics System

Linux on IBM Systems:
Leveraging common strengths and differentiated capabilities

Differentiated

System x
- Virtualization and consolidation through KVM and Xen
- Real Time Linux: Latency matters
- Extremely broad range of ISVs
- Innovations such as Power Executive and the rear door heat exchanger

Power Systems
- Advanced RAS features
- Live partition migration
- Performance generally comparable to AIX
- x86 consolidation platform
- Extensive ISV support via the Chiphopper™ program

System z
- Run natively or in an IFL
- Consolidate hundreds or thousands of workloads
- Extensive ISV support via the Chiphopper™ program

Intrinsic

Security
CAPP/EAL 4+ Common Criteria
SELinux, AppArmor
Very rapid time to fix if vulnerabilities are discovered

Efficiency
Dynamic, tickless kernel
Fastest revisions (and newest features) of any mainstream OS kernel

Scalability
Wristwatches to mainframes
Considerable effort in community to support scaling up and out

Linux on IBM server platforms

- **One of the primary tasks of the LTC**
  - First-class support for Enterprise Linux distributions across IBM server platforms:
    - System x, Power Systems, System z
  - Work with/in the community to enable critical software components (e.g. Linux kernel, GNU toolchain, ...)
- **LTC contributions to GNU toolchain**
  - Significant contributions to POWER and System z platform support across the toolchain
  - Contributions to common code, e.g. GCC auto-vectorization, GDB multi-architecture support
- **What about LLVM?**

IBM and LLVM

- **What about LLVM?**
  - Until recently, LLVM was not seen as critical for enterprise Linux platforms
  - LTC did not want to commit the necessary resources to fully support a second toolchain
  - This perception changed due to increased usage of LLVM in both open-source and proprietary apps
- **Current status (as of mid-2012)**
  - Decision to support LLVM across IBM server platforms
    - Fix PowerPC back-end for 64-bit POWER servers
    - Create new SystemZ back-end
- **So what changed?**

Important LLVM use cases

- **Use of LLVM as JIT**
  - 3D graphics: llvmpipe mesa/gallium driver
    - Current GNOME now requires 3D graphics
    - This means llvmpipe will be required for (remote) desktop support in upcoming enterprise distros
  - Certain proprietary database applications
    - LLVM JIT to compile SQL stored procedures
- **Use of LLVM to help software development**
  - Specific requirement by certain (potential) customers
  - Address sanitizer, thread sanitizer, ...
  - Clang error messages
- **Overall: LLVM support seen as critical now**

Contributions to PowerPC back end

Contributions to PowerPC back end

Team:
Bill Schmidt
Will Schmidt
Adhemerval Zanella
Ulrich Weigand

- **Verify & fix correctness issues**
  - Internal regression suite & projects/test-suite
    - Test suite issues
      - Platform assumptions (endian / bitsize / signed-char)
      - Apple GCC assumptions in Altivec tests
      - Math accuracy issues
      - Still issues with matching reference outputs
    - Proper support for PPC64 TOC
    - Exception handling (and DWARF) fixes
    - MachineCSE: insn that uses/defs the same physreg
    - Big-endian codegen bug in ExpandRes_BITCAST
    - Fix post-RA scheduler anti-dependencies breaking
    - Fix invalid pre-inc transformation in the DAG combiner
  - Set up build bots

Contributions to PowerPC back end

- **Verify & fix correctness issues (cont.)**
  - GCC's mixed-compiler ABI compatibility test suite
    - Placement of small struct arguments
    - Proper alignment for certain argument types
    - Support empty aggregate types
    - Implicit sign/zero extension of arguments / return values
    - Save/restore nonvolatile condition code fields
    - Complex argument passing
    - Fix complex float / 128-bit integer return value types
    - Traceback tables
    - Still mismatches for certain special cases
      - e.g. “attribute ((aligned))”

Contributions to PowerPC back end

- **Verify & fix correctness issues (cont.)**
  - Bootstrap compiler
    - Various instances of non-deterministic code generation
      - TOC ordering
      - TLS dynamic models
      - Stack slot ordering
    - No integrated Makefile support for bootstrap?
  - Build tests with integrated assembler forced on
    - Uncovered various wrong instruction encodings
    - Other differences, e.g data & DWARF/EH sections
    - Still some differences in generated object files as compared to GAS output
      - e.g. symbol table ordering
      - Is it feasible to make output fully identical?

Contributions to PowerPC back end

- **New features**
  - Code generation
    - Compile-time PowerPC long double support
    - Fully implement TLS support
    - Implement medium/large code model support
    - Some Altivec enhancements
  - JIT support
    - Implement MCJIT support (64-bit only)
  - Assembler parser support
    - In progress, patches pending review
    - Common code support patches now all accepted
  - Disassembler support
    - t.b.d.

Contributions to PowerPC back end

- **Future work**
  - Improved ISA support
    - Support current processors (power5 ... power7+)
    - In particular: VSX vector instruction support
  - Performance tuning
    - In particular: instruction scheduling
    - Benchmark analysis (LLVM about 7% worse than GCC)
  - Maybe: 32-bit support
    - Verify 32-bit Linux ABI & codegen correctness
    - Implement 32-bit MCJIT support

New SystemZ back end

Contributions to SystemZ back end

Team:
Richard Sandiford
Ulrich Weigand

- **History of LLVM support on SystemZ**
  - Initial support added in 2009 by Anton Korobeynikov
  - Back-end was removed again in 2011
- **New back end to be contributed by IBM**
  - Loosely based on old back end, significant re-implementation
- **Feature set**
  - 64-bit z/Architecture only
  - Support for z10 (and newer) processor only
  - Linux operating system support only
  - Focus on features and correctness, not performance

Contributions to SystemZ back end

- **Current status**
  - Working C/C++ compiler
    - Passes testsuite and projects/test-suite with no failures
    - Passes bootstrap with identical stage2/stage3 results
    - Runs SPECcpu2006 benchmarks successfully
    - Passes the ABI compatibility test suite against GCC 4.8
  - Working integrated assembler
    - Passes testsuites with integrated assembler forced on
  - Working assembler parser
    - Passes testsuites when using clang assembler
  - Working MCJIT (no support for old JIT)
    - Passes JIT testsuite

Contributions to SystemZ back end

- **Next steps**
  - Get back end accepted & integrated
    - Reviews currently in progress
    - Goal: Make LLVM 3.3 release (?)
  - Performance optimization
    - LLVM about 15% worse than GCC
    - Improved condition code handling
    - Exploit more System z instructions (memory-to-memory, string, branch on count, ...)
    - Improved ISA support (z196, zEC12)
    - Instruction scheduling & tuning
  - Maybe: 31-bit support

Working on LLVM vs. GCC

Some observations from a back-end developer's perspective

LLVM vs. GCC back end

- **Many things look similar**
  - Sequence of passes
  - .td files vs .md files
- **Differences**
  - LLVM seems to provide more flexibility in adding target-specific passes / overriding common passes
  - Had some difficulties with .td syntax/semantics
    - Ran into a couple of issues/problems
      - Complex address operands, pre-inc addresses
      - Trying to track down encoding bugs
    - Had to read TableGen source code to understand what's going on ...
    - Reference documentation ?

LLVM vs. GCC – Back-end passes

GCC	LLVM
expand & combine	SelectionDAGISel
early split	EmitInstrWithCustomInserter
Early RTL opt passes	MachineSSAOptimization
sched	EmitSchedule
ira / reload	RegAllocPass
n/a	addPreRegAlloc / addPostRegAlloc
thread_prologue_and_epilogue	PrologEpilogCodeInserter
Late RTL opt passes	MachineLateOptimization
late split	ExpandPostRAPseudos
sched2	PostRAScheduler
reorder_blocks	MachineBlockPlacements
machine_dependent_reorg	addPreEmitPass
final	EmitFile / EmitObjectCode

LLVM vs. GCC – Machine definition

- **GCC .md file**

```
(define_insn "ashrsi3"
    [(set (match_operand:SI 0 "gpc_reg_operand" "=r,r")
          (ashiftrt:SI (match_operand:SI 1 "gpc_reg_operand" "r,r")
                       (match_operand:SI 2 "reg_or_cint_operand" "r,i")))]
    ""
    "@
     sraw %0,%1,%2
     srawi %0,%1,%h2"
    [(set_attr "type" "var_shift_rotate,shift")])
```

- **LLVM .td file**

```
defm SRAW : XForm_6rc<31, 792, (outs gprc:$rA), (ins gprc:$rS, gprc:$rB),
                        "sraw", "$rA, $rS, $rB", IntShift,
                        [(set i32:$rA, (PPCsra i32:$rS, i32:$rB))]>;

defm SRAWI: XForm_10rc<31, 824, (outs gprc:$rA), (ins gprc:$rS, u5imm:$SH),
                          "srawi", "$rA, $rS, $SH", IntShift,
                          [(set i32:$rA, (sra i32:$rS, (i32 imm:$SH)))]>;
```

Summary

- **LLVM usage getting more and more wide spread**
  - Now critical to enterprise Linux applications
- **IBM wants to ensure good LLVM support across our server platforms**
  - Started contributing to PowerPC and SystemZ
  - Ongoing investment going forward
- **LLVM code base**
  - Experienced GCC back-end developers should be able to work on LLVM back end with little difficulties
  - Some more .td documentation could be helpful

Questions

?