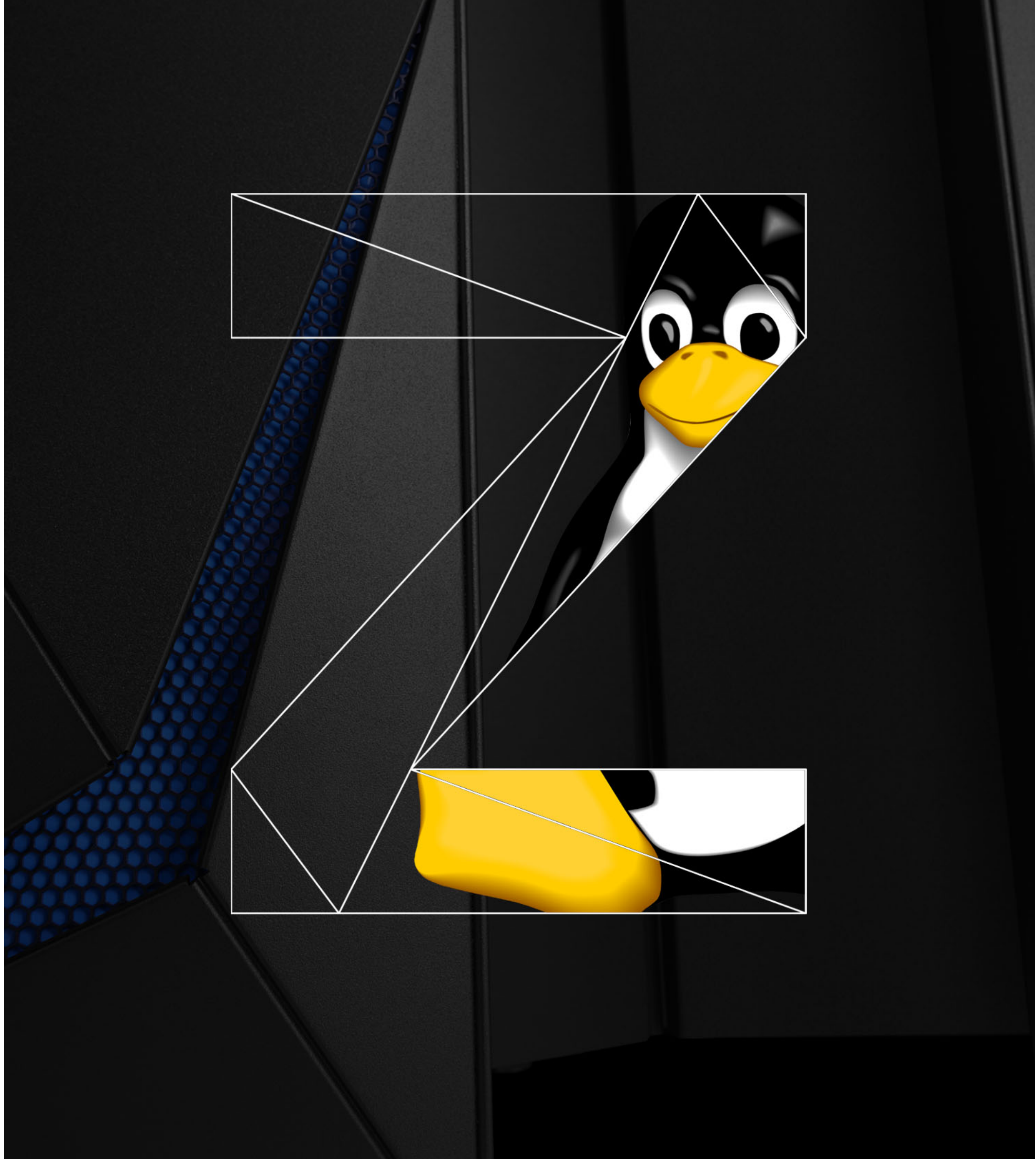
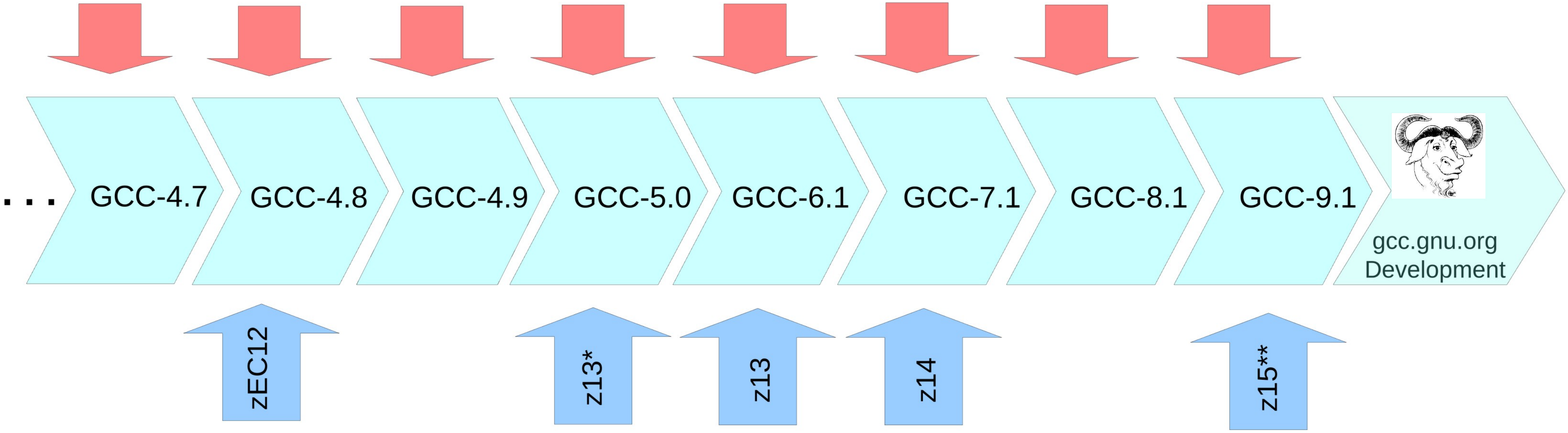
Linux on IBM Z - Compiler GCC

Mario Held
Linux Performance Analyst



IBM GCC compiler development driven by Open Source and new IBM z generations

IBM GCC development provides patches **to exploit new GCC features** also in Linux on IBM Z
(i. e. software DFP, Transactional Execution TX)



IBM development provides patches **to exploit new IBM hardware features** in new GCC versions
(new instructions, hardware DFP, pre-fetching, hardware TX, changed pipeline architecture, SIMD)
IBM development cycle is running in sync with the gcc.gnu.org development cycle

* z13 support with GNU GCC-5.2   **z15 aka arch13

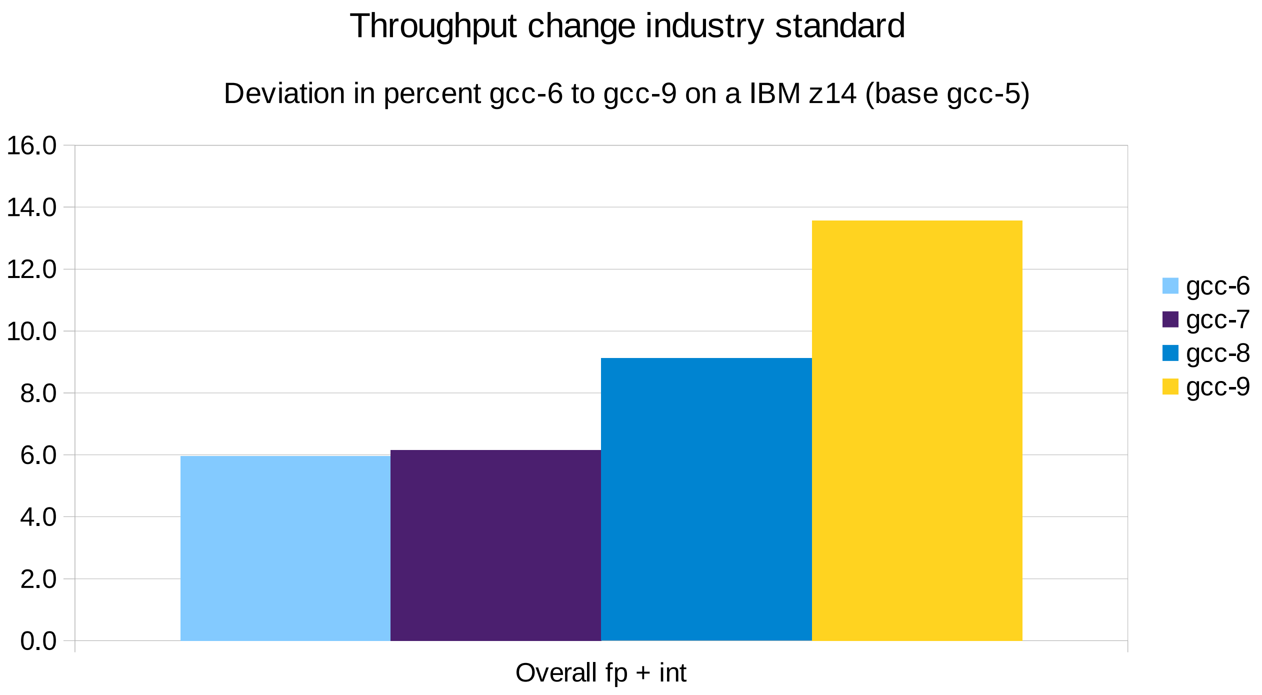
GCC versions in Distributions for IBM Z

GCC stream	First release	Highest -march	SLES	RHEL	Ubuntu
4.1	02/2006	z9-109	SLES10	RHEL5	
4.2	05/2007	z9-109			
4.3	05/2008	z9-ec	SLES11 (backport z10)		
4.4	04/2009	z10		RHEL5.6**/6.1 (backport z196)	
4.5	04/2010	z10	SLES11 SP1		
4.6	03/2011	z196	SLES11 SP2*	RHEL6	
4.7	03/2012	z196	SLES11 SP3 (4.7 opt. zEC12)**		
4.8	03/2013	zEC12	SLES12	RHEL7.3 (4.8 backport z13)	
4.9	04/2014	zEC12			
5	04/2015	zEC12			
5.2	07/2015	z13	SLES12 SP1 (5.2 TCM z13)**		Ubuntu16.04 (5.3)
6.1	04/2016	z13	SLES12 SP2 (6.6 TCM z13)**	RH Developer Toolset 6**	
7.1	05/2017	arch12 (z14)			
7.2	08/2017	z14	SLES12 SP3 (7.2 TCM z14)**	RH Developer Toolset 7**	Ubuntu18.04 (7.3)
8.1	05/2018	z14	SLES12 SP4 (8.8 TCM z14)**	RH Developer Toolset 8**	
9.1	05/2019	arch13 (z15)			
9.2	10/2019	z15			Ubuntu 19.10 (9.2)

*   included in SDK, optional, not supported
** fully supported add-on compiler until next add-on toolchain release

Note: Future Linux distribution contents depend on distributor support and are always subject to change without notice! Also schedules of gcc.gnu.org are always subject to change without notice!

Relative Performance of different GCC versions on identical hardware (IBM z14)



Study on Fedora-29 based Linux with close to GA GCC development versions from gcc.gnu.org

- Throughput change overall (average integer and floating point suite result)
  - Improvement in percent, normalized to base gcc-5

Optimizing C/C++ code

- Options -O3 or -O2 are a good starting point (used in most frequently in our performance measurements)
- Optimize instruction scheduling with performance critical target machine in mind using -mtune parameter
  - -mtune=values <g5*, g6*, z900, z990, z9-109, z10, z196, zEC12 with all supported GCC versions>
  - <z13 with RHEL7.3 gcc-4.8, SUSE TCM gcc-5.2, Ubuntu16.04 gcc-5.3, and GNU gcc-6.1 and higher>
  - <z14 with RHEL Developer Toolset 8, SUSE TCM gcc-8.8, Ubuntu18.04 gcc-7.3, and GNU gcc-8.1 and higher>
  - <z15 with Ubuntu 19.10 gcc-9.2 >
  - Default is the value used for -march if -march is specified as a compile parameter
- Exploit also improved machine instruction set and new hardware capabilities using -march parameter
  - -march=values <g5*, g6*, z900, z990, z9-109, z10, z196, zEC12, z13, z14, z15> available with the same compilers as mentioned above
  - **-march compiled code will only run on the target machine or newer**

* deprecated / will not be supported with gcc-9.1 and higher

Optimizing C/C++ code (cont.)

- Fine Tuning: additional general options on a file by file basis
  - -funroll-loops has often advantages on IBM Z
    - Unrolling is internal delimited to a reasonable value by default
  - Use of inline assembler for performance critical functions may have advantages
  - -ffast-math speeds up calculations (if not exact implementation of IEEE or ISO rules/specifications for math functions is needed)
- For a more comprehensive description of the -m options defined for the architecture see the GCC documentation at gnu.org

  https://gcc.gnu.org/onlinedocs/gcc/S_002f390-and-zSeries-Options.html#S_002f390-and-zSeries-Options

Special Optimization - GCC Feedback Driven Optimization (FDO)

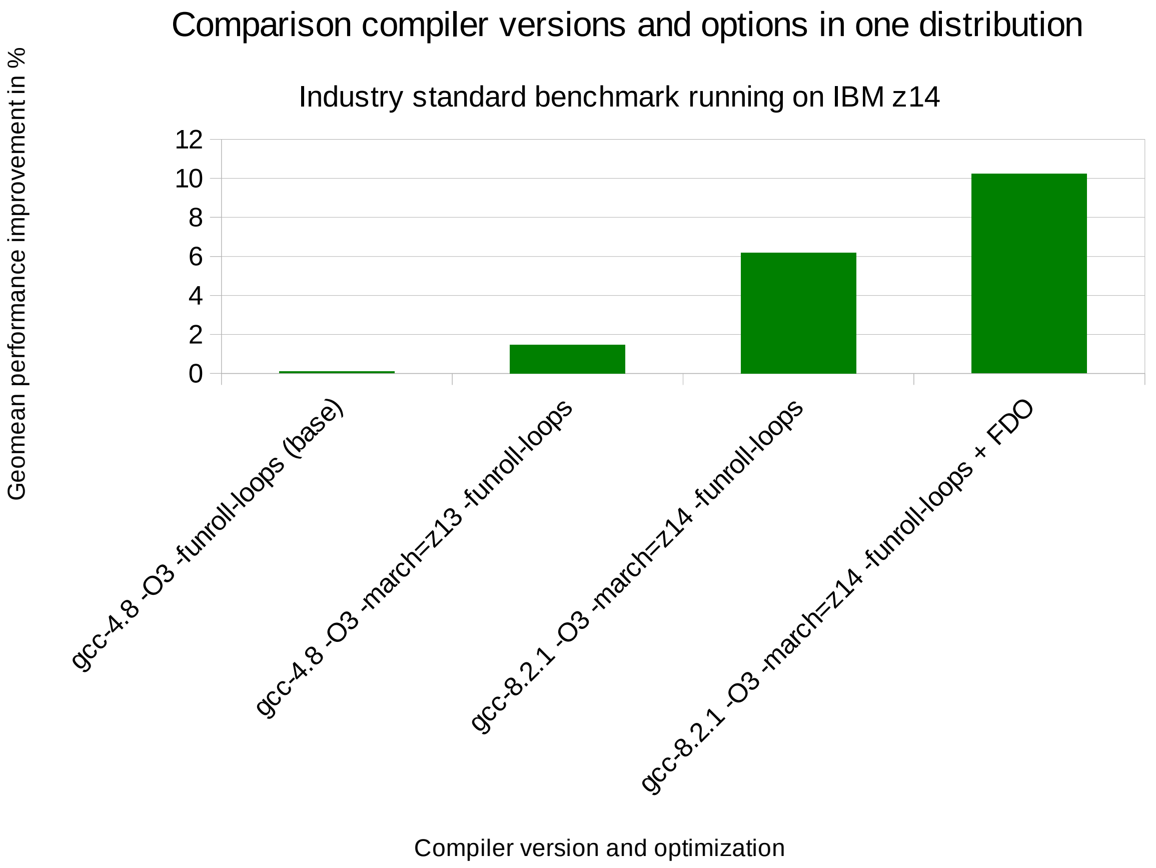
- FDO is also known as Profiled Directed Feedback (PDF)
- With FDO the compile is done in three phases:
  - Profile code generation, instrumentation code gets inserted
  - Training run while statistic information gets collected into a file, especially which code parts are used how often
  - Feedback optimization using the collected data from the previous phase to guide the optimization routines for instance for branch prediction or loop unrolling
- FDO produces in most cases significant better code and improves performance significantly
- FDO requires more compile time because of compiling twice and doing the test run, but it is usually worth the investment
- Best results if the codes' hot paths are not depending on the single input data
  - The advantage of FDO depends on a really representative training workload
  - If the training workload is not good your application could even run more slowly
- Option to add in the first pass: -fprofile-generate
- Option to add in the second pass: -fprofile-use

Special Optimization - GCC Link Time Optimization

- Link Time Optimization (LTO) enables cross-module optimization without changing the build infrastructure.
- Problem: Current build mechanics pass one source code file at a time to the compiler → cross-module optimization not possible.
- Solution: Optimization is postponed until link-step when all the required modules are known
- LTO compilation procedure:
  - First the compilation units are optimized separately
  - GCC internal code representation (GIMPLE) is embedded into the object file
  - During link-step the objects are passed to the compiler again
  - Compiler uses the embedded information to redo the optimization step
- Higher potential together with Feedback Driven Optimization (FDO)
- GCC support introduced with GCC 4.5 and matured since then
- Experiments showed single-digit performance improvement
- Options to add: -flto

GCC Performance influenced by compiler version and compile options

- Advantages of using current compilers and higher optimization are significant in general
  - Improved machine / instruction support come with newer GCC versions
  - In the chart comparing GCC performance in RHEL7.6 using standard and DTS compilers with different options on a IBM z14 using geomean of 22 different real world applications



Options for generating 31-bit code to execute on a 64-bit system

- Some applications have to be compiled for 31-bit mode, because they are currently not prepared for 64-bit mode
- Recommended options and flag settings when compiling for 31-bit mode on a 64-bit system
  - Compiler options for C, C++, and Fortran
    - CFLAGS, CXXFLAGS, FFLAGS: append '-m31'
    - With the option '-m31', the compiler generates code which is compliant to the GNU/Linux for s390 ABI
- When using the '-m31 -mzarch' options the generated code still conforms to the 32-bit ABI but uses the general purpose registers as 64-bit registers internally
  - Requires a Linux kernel saving the whole 64-bit registers when doing a context switch
- **Sometimes in the future distributions will remove the 31-bit support**
  - Be prepared and port the code to 64-bit

Thank You

Mario Held
Performance Analyst

Linux on IBM Z Performance Evaluation

IBM

Research & Development
Schoenaicher Strasse 220
71032 Boeblingen, Germany

mario.held@de.ibm.com

Linux on System z – Tuning hints and tips http://www.ibm.com/developerworks/linux/linux390/perf/index.html

Live Virtual Classes for z/VM and Linux http://www.vm.ibm.com/education/lvc/

Mainframe Linux blog http://linuxmain.blogspot.com

Acknowledgements

- Andreas Krebbel
  GNU Toolchain
  Development

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

- The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.
- Red Hat®, Fedora®, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.
- "SUSE" and the SUSE logo are trademarks of SUSE IP Development Limited or its subsidiaries or affiliates in the United States and other countries.
- Ubuntu and Canonical are registered trademarks of Canonical Ltd in the United States and other countries.
- Other product and service names might be trademarks of IBM or other companies.