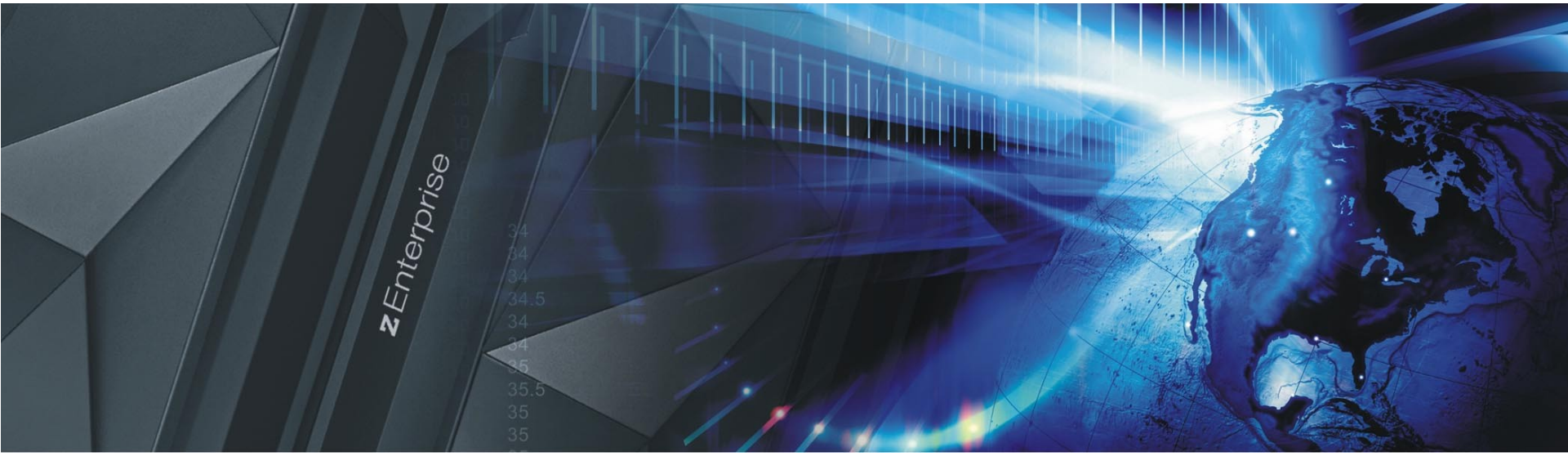
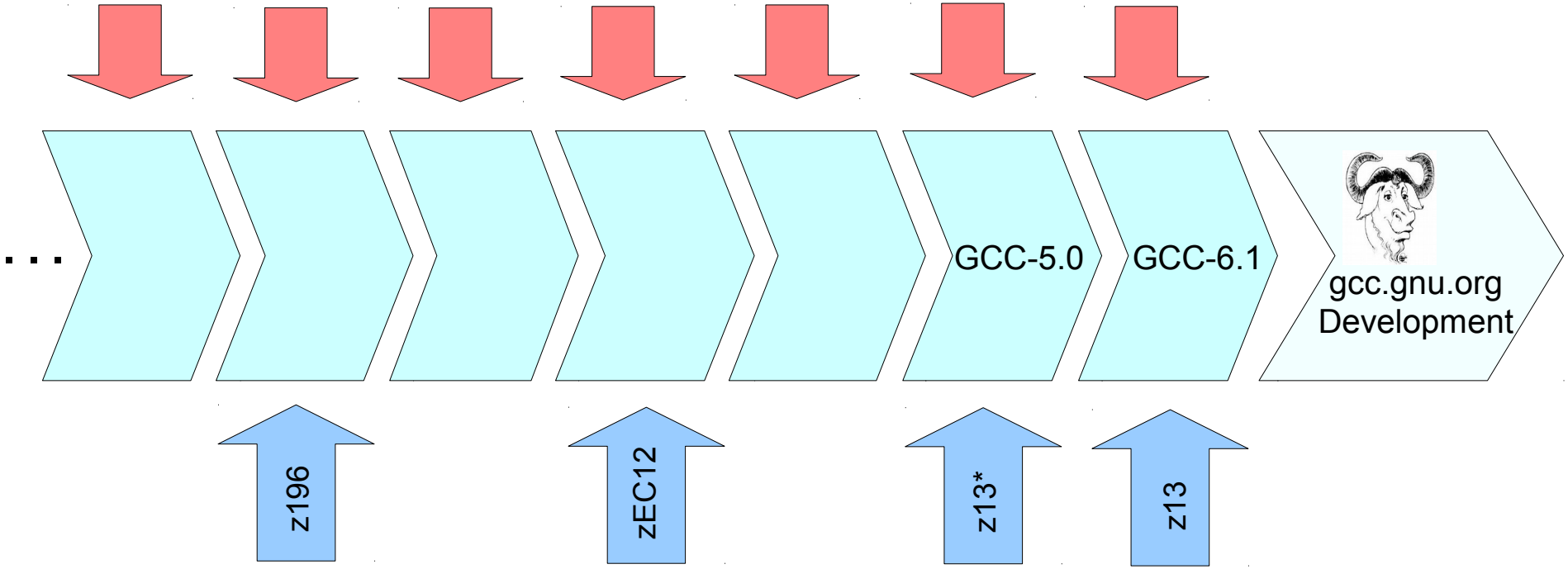
Linux on IBM z Systems – Compiler GCC

Mario Held, IBM Research &Development Germany,
System Performance Analyst



GCC compiler

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



IBM development provides patches to exploit new IBM hardware features in new GCC versions (i. e. 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

GCC versions in Linux z Systems distributions

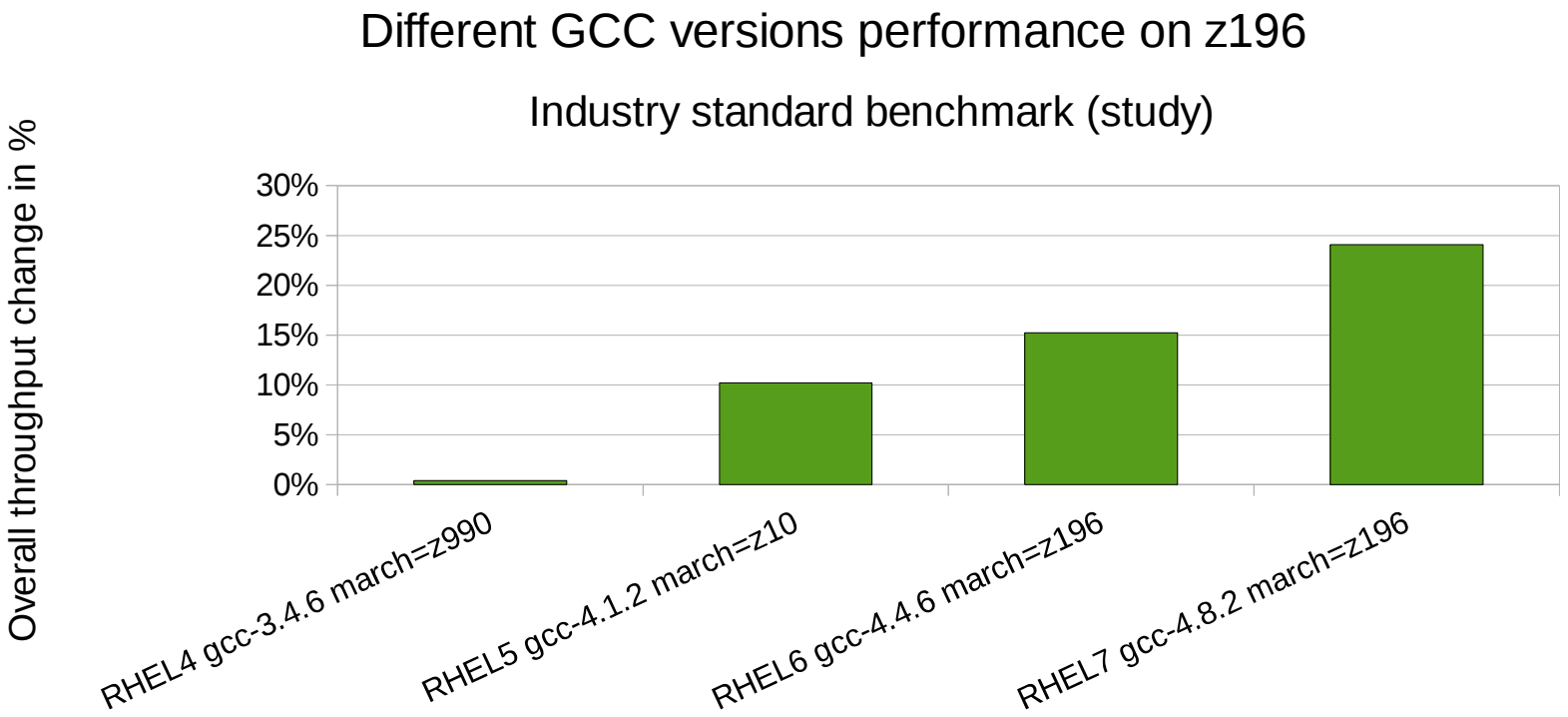
GCC stream	First release	Max -march	Included in SUSE	Included in Red Hat	Included in Ubuntu
4.1	02/2006	z9-109	SLES10	RHEL5	
4.2	05/2007	z9-109			
4.3	05/2008	z9-ec	SLES11 (z10 backport)		
4.4	04/2009	z10		RHEL5.6**/6.1 (z196 backport)	
4.5	04/2010	z10	SLES11 SP1		
4.6	03/2011	z196	SLES11 SP2*		
4.7	03/2012	z196	SLES11 SP3 (zEC12 backport)**		
4.8	03/2013	zEC12	SLES12	RHEL7	
4.9	04/2014	zEC12			
5	04/2015	zEC12	SLES12 SP1 (5.2 with z13)**		16.04 (5.3 with z13)
6.1	04/2016	z13			

* included in SDK, optional, not supported
** fully supported add-on compiler

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!

GCC evolution

- Advantages of using current compilers are significant in general
  - Improved machine / instruction support come with newer GCC versions
  - In the chart comparing GCC performance in RHEL4 – RHEL5 – RHEL6- RHEL7 on a z196 using 14 different applications



Optimizing C/C++ code

- Produce optimized code
  - Options -O3 or -O2 (often found in delivered makefiles) are a good starting point and are used in most frequently in our performance measurements
  - Optimize GCC instruction scheduling with the performance critical target machine in mind using -mtune parameter
    - -mtune=values <z900, z990 with all supported GCC versions>
    - <z9-109 with gcc-4.1>
    - <z10 with SLES11 gcc-4.3 or gcc-4.4>
    - <z196 with RHEL6 gcc-4.4, optional SLES11 SP1 gcc-4.5*, or GNU gcc-4.6>
    - <zEC12 with GNU gcc-4.8>
    - <z13 with Ubuntu16.04 gcc-5.3, or GNU gcc-6.1>
  - Exploit also improved machine instruction set and new hardware capabilities using the -march parameter
    - -march=values <z900, z990, z9-109, z10, z196, zEC12, z13> available with the same compilers as mentioned above
    - Includes implicitly -mtune optimization if not otherwise specified
    - **-march compiled code will only run on the target machine or newer machines**

* not fully supported version

Optimizing C/C++ code (cont.)

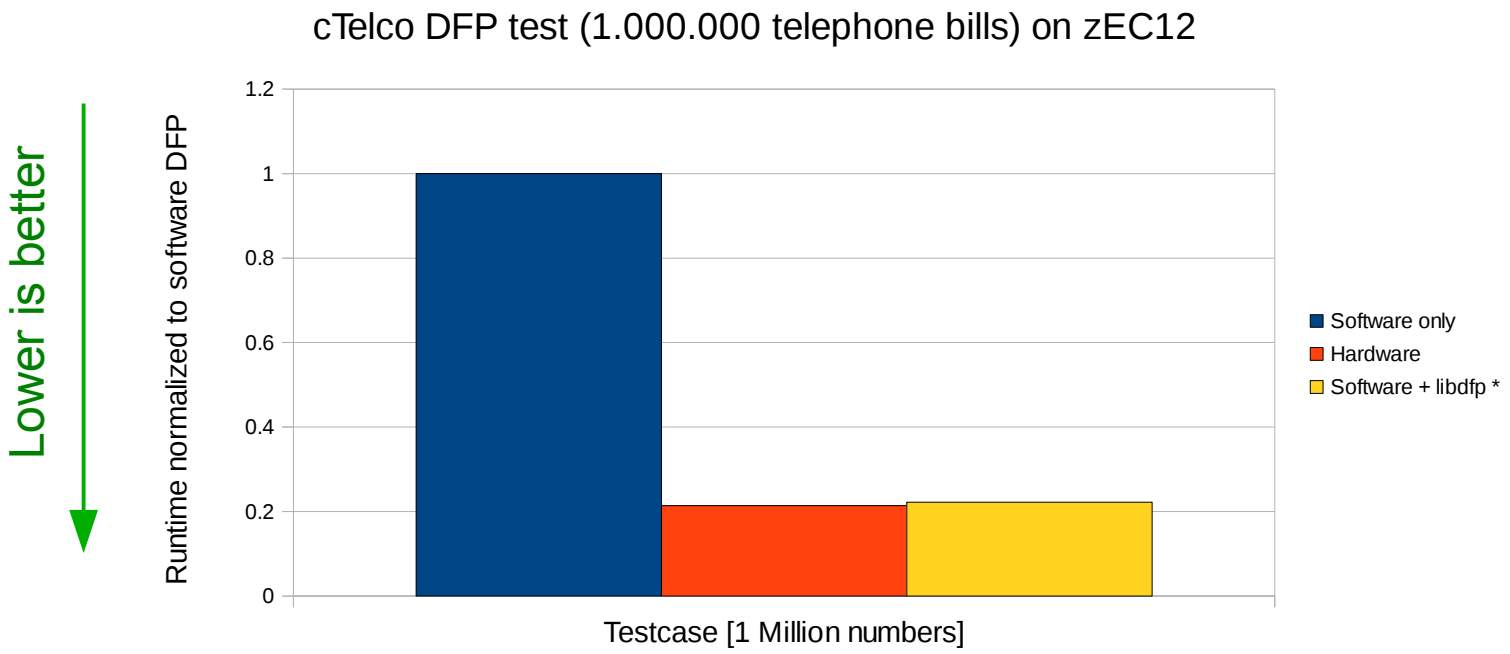
- Fine Tuning: additional general options on a file by file basis
  - -funroll-loops has often advantages on System z
    - Unrolling is internal delimited to a reasonable value by default
  - Use of inline assembler for performance critical functions may have advantages
  - -ffast-math speeds up calculations (if not exact implementation of IEEE or ISO rules/specifications for math functions is needed)

Decimal Floating Point (DFP) - Limitations of binary numbers in business applications

- Mercantile goods and amounts of money cannot be calculated or represented exactly by binary floating point numbers
  - Many numbers cannot be represented properly (1/5, 1/10)
  - People who are used to decimal numbers expect results and calculations to be available with full precision
  - The traditional binary representation is not suitable for usual calculations
    - $ 0.70 x 1.05 = $ 0.734999999999999986677323704498121514916419982910156 25
    - Rounding to two decimal places gives $ 0.73
    - Expected is $ 0.70 x 1.05 = $ 0.735 => rounded to $ 0.74
- If you rely on correctly calculated results without DFP you have to add many lines of code to your program
  - Example: troublesome binary floating point rounding mechanisms
    - Sometimes more than 50 times the number of lines than in an DFP implementation
    - Additional code is error-prone
    - Without DFP more processor cycles are used in calculations

DFP - performance

- Big performance advantage if DFP hardware support is exploited directly or via hardware accelerated library libdfp
- Supported in SUSE SLES11 SP1 and Red Hat RHEL6.0 (and newer)
- Benchmark: Telco testcase models a telephone company's billing system
  - Billing of one million telephone calls including tax using DFP arithmetics



* libdfp provides also hardware support

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.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.Java is a registered trademarks of Oracle and/or its affiliates.

Red Hat, Red Hat Enterprise Linux (RHEL) are registered trademarks of Red Hat, Inc. in the United States and other countries.

SUSE and SLES are registered trademarks of SUSE LLC 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.