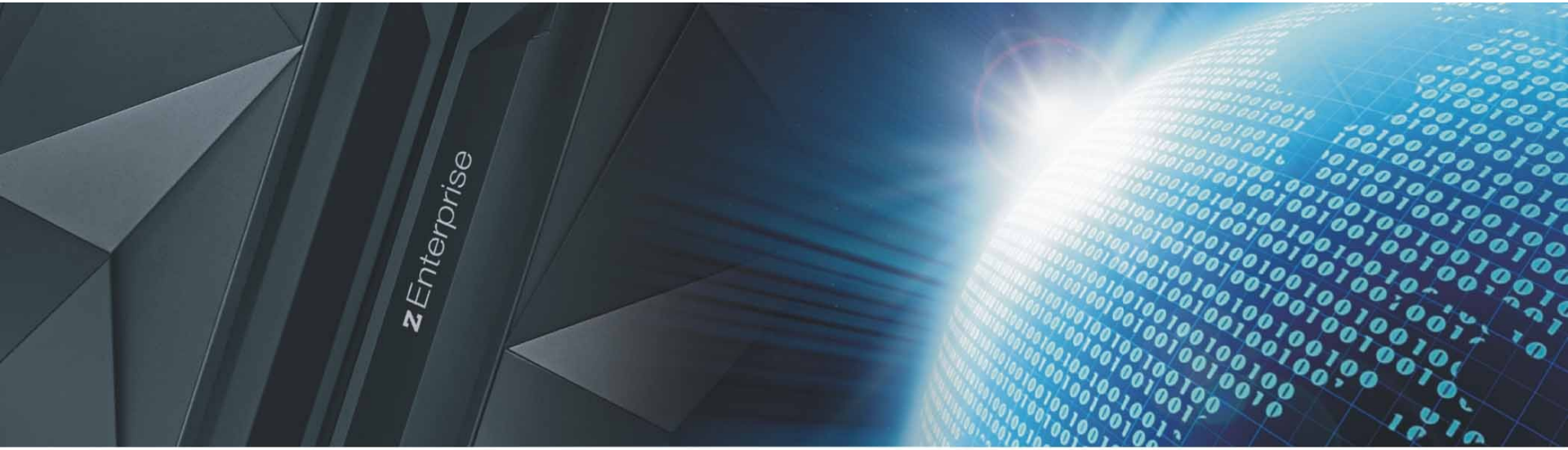
GCC – IBM z Systems Compile Options

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



Special GCC compile options for IBM z Systems

- With SLES9 gcc-3.3 and RHEL4 gcc-3.4 compile options '-march' and '-mtune' became available to Linux users on the mainframe for the first time
- The gcc-4.1 introduces the additional argument **'z9-109'** for parameters '-mtune' and '-march'
- The gcc-4.4 provides IBM System z10 support introducing the new argument **'z10'** for '-march' and '-mtune' parameters
  - Backported into SUSE SLES11 gcc-4.3
- The gcc-4.6 provides IBM System zEnterprise 196 support introducing the new argument **'z196'** for '-march' and '-mtune' parameters
  - Backported into Red Hat RHEL6 U1 gcc-4.4
  - Backported into SUSE SLES11 SP1 gcc-4.5 optional compiler (in SDK, not supported)
- The gcc-4.8 provides IBM System zEnterprise EC12 support introducing the new argument **'zEC12'** for '-march' and '-mtune' parameters
  - SLES11 SP3 add-on gcc-4.7
- The gcc-5.3 provides IBM System zEnterprise 13 support introducing the new argument **'z13'** for '-march' and '-mtune' parameters
- The gcc-6.1 provides IBM System zEnterprise 13 support introducing the new argument **'z13'** for '-march' and '-mtune' parameters
  - Backported into SLES12 SP1 add-on gcc-5.2
- Performance improvement varies and depends upon processor utilization and optimization opportunities in the program code

Special GCC compile options for IBM z Systems (cont.)

- **'-mtune=z900 | z990 | z9-109 | z9-ec | z10 | z196 | zEC12 | z13'** generates code optimized for the particular processor and the set of available instructions.
  - The compiler's instruction scheduling is influenced but the instruction set keeps to the used GCC compilers default value

The generated code targeting one processor type will run on prior mainframe generations but may cause a performance degradation there

  - For BC machines use their EC equivalent parameter

- **'-march=z900 | z990 | z9-109 | z9-ec | z10 | z196 | zEC12 | z13'** generates code optimized for the particular processor, using the instruction set of the processor.
  - The generated code will run on the target processor type and any newer mainframe processor type
    - So code compiled with 'march=z196' will run on a IBM zEC12 but it is not guaranteed that code compiled with 'march=zEC12' will run on an IBM z196.

- Check GCC defaults by using command 'gcc -v' and search in the output for '... - -with-arch=< > - -with-tune=< > … '

Special GCC compile options for IBM z Systems (cont.)

- Our experiments show remarkable overall performance improvements.
  - Use '-march' if the generated optimized code is to be executed on only that single target machine type
    - If more than one target machine is identified use the argument for the oldest model ('-march' parameter is upward compatible)
  - Use '-mtune' parameter if the generated optimized code is intended to run optimal on a specific target machine type but should be runnable on older machines too
  - Defaults coming with the compiler
    - For **RHEL5 GCC-4.1** '-mtune=z9-109' and '-march=z900'
    - For **RHEL6 GCC-4.4** '-mtune=z10' and '-march=z9-109'
    - For **RHEL7 GCC-4.8** '-mtune=zEC12' and '-march=z196'
    - For **SLES10 GCC-4.1** '-mtune=z9-109' and '-march=z900'
    - For **SLES11 GCC-4.3** '-mtune=z9-109' and '-march=z900'
    - For **SLES12 GCC-4.7** '-mtune=zEC12' and '-march=z196'
    - For **Ubuntu 16.04 GCC-5.3** '-march=zEC12'

- General GCC compile options improving performance
  - Utilize the highest optimization level using parameter '-O3'.
  - The option '-funroll-loops' might help to speed up your application in most cases.
  - For a more comprehensive description of the -m options defined for the (31-bit) and (64-bit) architectures see the GCC page 'S/390 and zSeries Options at

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

Options for generating 31-bit code 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

Trademarks

