
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

CMP325

Coding for multiple CPU architectures using lessons learned in HPC

Karthik Raman

Principal HPC Specialist SA
AWS

Arthur Petitpierre

Principal Graviton Specialist SA
AWS

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Arthur Petitpierre – Who am I?



2016–2019
Delivering snowballs – Paris



2019–today
Paddling for AWS Graviton – Seattle

But also:
2015–2016: HPC Services CTO
2016–2019: AWS EMEA HPC Specialist SA
2019–today: AWS Graviton Specialist SA

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Karthik Raman – Who am I?



2011–2020 : HPC Performance Engineer (at an x86 company)
2020–2022 : AWS HPC Specialist SA
2022–today : AWS HPC Application engineer

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

x86-64? Arm64? What are you talking about?

Interpreted / bytecode compiled: source → interpreter → machine code

Compiled: source → compiler → machine code

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s see some code

```
int square(int num) {
    return num * num;
}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

And some assembly

x86-64 gcc-12.2.0

```
square(int):
push rbp
mov rbp  rsp
mov DWORD PTR  rbp 4  edi
mov eax  DWORD PTR  rbp 4
imul eax  eax
pop rbp
ret
```

arm64 gcc-12.2.0

```
square(int):
sub sp, sp, #16
str w0, [sp, 12]
ldr w0, [sp, 12]
mul w0, w0, w0
add sp, sp, 16
ret
```

https://godbolt.org/ is your best friend

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

How/when do you write something architecture-specific?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Architecture-specific constructs

Endianness

Crypto

Random numbers

Data types

SIMD

Locks/mutexes

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Graviton specs

Processor	Graviton2	Graviton3
Instances	M6g/M6gd, C6g/C6gd/C6gn, R6g/R6gd, T4g, X2gd, G5g, and Im4gn/Is4gen	C7g
Core	Neoverse-N1	Neoverse-V1
Frequency	2500MHz	2600MHz
Turbo supported	No	No
Instruction latencies	Instruction Latencies	Instruction Latencies
Interconnect	CMN-600	CMN-700
Architecture revision	ARMv8.2-a	ARMv8.4-a
Additional features	fp16, rcpc, dotprod, crypto	sve, rng, bf16, int8, crypto
Recommended -mcpu flag	neoverse-n1	neoverse-512tvb
RNG Instructions	No	Yes
SIMD instructions	2x Neon 128bit vectors	4x Neon 128bit vectors / 2x SVE 256bit
LSE (atomic mem operations)	yes	yes
Pointer Authentication	no	yes
Cores	64	64
L1 cache (per core)	64KB inst / 64KB data	64KB inst / 64KB data
L2 cache (per core)	1MB	1MB
LLC (shared)	32MB	32MB
DRAM	8x DDR4	8x DDR5
DDR Encryption	yes	yes

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Do compilation options matter?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Yes – And not only the options

- Which compiler? (Gcc / LLVM / vendor-specific compiler)
- Compiler versions
- Compiler options (-mcpu / -mtune)
- Cross-compilation (hint: don’t do it)
- Are fat binaries an option?

- An HPC-specific solution – Spack
  https://computing.llnl.gov/projects/spack-hpc-package-manager

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Tale of two (too many) compiler options?

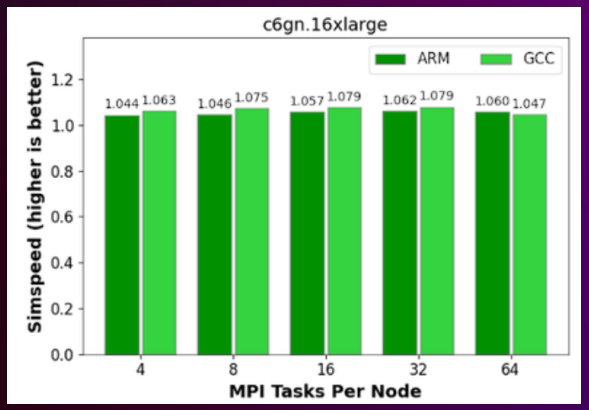
- Latest GCC Compilers provide comparable performance to Arm Compilers

- GCC 10.2 or above recommended for Graviton2 and Graviton3
  - Kernel 5.10+ required for SVE support (Gravtion3)

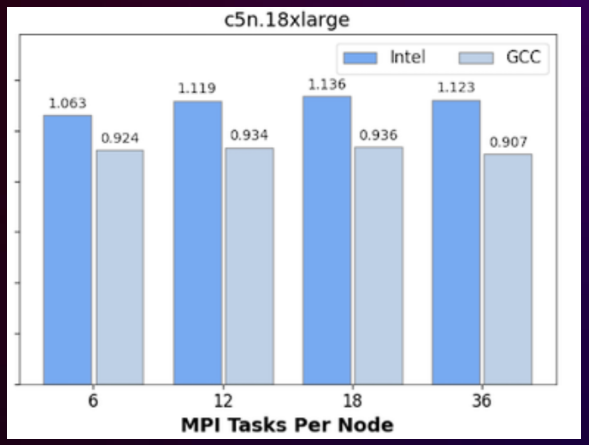
- Compile natively (on Graviton2 or Graviton3) using GCC 10.2+ with “-mcpu=native” option

- Processor-specific options
  - Graviton2: -mcpu=neoverse-n1
  - Graviton3: -mcpu=neoverse-512tvb

TL;DR: Compile natively with GCC 10.2+, kernel 5.10+ using -mcpu=native
Avoid -march and -mtune



Weather Research and Forecasting (WRF) on Gravtion2 (Arm compiler vs. GCC)



Weather Research and Forecasting (WRF) on Intel Skylake (Intel compiler vs. GCC)

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Conditional compilation in C/C++

```
#ifdef __x86_64__
      // x86-64 specific stuff here
#elif __aarch64__
      // aarch64 specific stuff here
#else
      // do something generic here
#endif
```

The same thing can be done for SIMD instructions, RNG, etc.

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Do you Spack?

- Open-source package management framework and tool for installing complex scientific software
- Designed to support multiple versions and configurations of a software on many different platforms and environments
- Automates build and installation of scientific software
- 6501 packages with at least some running Setup
- Abstracted version numbers and dependencies
- Adding packages is straightforward

```
$ git clone https://github.com/spack/spack
```

```
$ spack install hdf5                    unconstrained

$ spack install hdf5@1.10.5   @custom version

$ spack install hdf5@1.10.5 %gcc@11.2.0
% customer compiler

$ spack install hdf5@1.10.5 +mpi
^openmpi@4.1.0 cppflags="-O3 –g3"
```

Spack https://github.com/spack/spack

aws

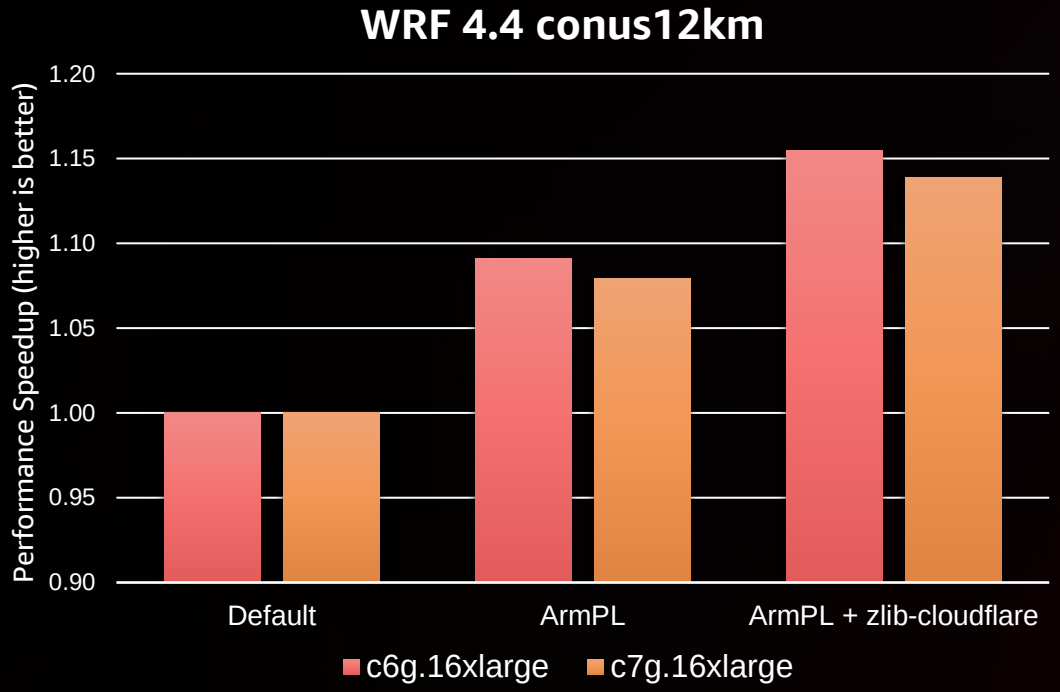
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Libraries and profiling

	X86_64 (CPU)	Arm
Math libraries	OpenBLAS, BLIS, Intel MKL	OpenBLAS, BLIS, ArmPL
Machine learning	Intel oneDNN	Intel oneDNN, ArmCL
MPI	Intel MPI, OpenMPI, MVAPICH, MPICH	OpenMPI, MVAPICH
Debuggers	Arm DDT, TotalView, Intel® Inspector	Arm DDT, TotalView
Profilers	Intel Vtune, AMD uProf, Arm MAP, Linux perf	Arm MAP, Linux perf

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Optimization example



Input/Output | Project Files | OpenMP Stacks | OpenMP Regions | Functions

Functions

Self time	Total	MPI	Child	Overhead	Function	Library
20.7%	20.7%	20.7%			MPI_Bcast	[mpi]
9.2%	9.2%			9.2%	[OpenMP overhead (no region active)]	
8.3%	8.3%	8.3%			MPI_Wait (point-to-point MPI)	[mpi]
6.1%	6.1%		<0.1%		__powf_finite	libm-2.26.so
3.6%	3.6%		<0.1%		__module_advect_em_MOD_advect_scalar_pd	wrf_neoverse-512tvb.exe
2.9%	2.9%				memset (OpenMP)	libc-2.26.so
2.1%	2.1%				__rrtmg_sw_reftra_MOD_reftra_sw	wrf_neoverse-512tvb.exe
2.1%	2.1%		<0.1%		__rrtmg_sw_vrtqdr_MOD_vrtqdr_sw	wrf_neoverse-512tvb.exe
2.0%	6.7%		4.7%		__rrtmg_sw_spcvmc_MOD_spcvmc_sw	wrf_neoverse-512tvb.exe
2.0%	2.2%		0.2%		__rrtmg_lw_rtrnmc_MOD_rtrnmc	wrf_neoverse-512tvb.exe
1.9%	1.9%		<0.1%		__module_advect_em_MOD_advect_scalar	wrf_neoverse-512tvb.exe
1.5%	1.6%		0.1%		__mcica_subcol_gen_lw_MOD_generate_stochastic_clouds	wrf_neoverse-512tvb.exe
1.4%	1.4%				__module_small_step_em_MOD_advance_uv	wrf_neoverse-512tvb.exe
1.4%	8.5%		7.1%		__module_mp_thompson_MOD_mp_thompson	wrf_neoverse-512tvb.exe
1.2%	1.2%				__expf_finite	libm-2.26.so
1.2%	1.4%		0.2%		__module_small_step_em_MOD_advance_w	wrf_neoverse-512tvb.exe
1.2%	1.2%				__memcpy_sve (OpenMP)	libc-2.26.so
1.1%	1.1%				__module_ieva_em_MOD_ww_split	wrf_neoverse-512tvb.exe
1.0%	2.5%		1.5%		deflate_fast	libz.so.1.2.7
1.0%	1.0%		<0.1%		__module_small_step_em_MOD_calc_p_rho	wrf_neoverse-512tvb.exe
0.9%	1.0%		0.1%		__module_small_step_em_MOD_advance_mu_t	wrf_neoverse-512tvb.exe
0.9%	0.9%		<0.1%		__mcica_subcol_gen_sw_MOD_generate_stochastic_clouds_sw	wrf_neoverse-512tvb.exe
0.8%	0.8%		<0.1%		__module_diffusion_em_MOD_cal_deform_and_div	wrf_neoverse-512tvb.exe
0.8%	1.1%		0.3%		__module_em_MOD_rk_update_scalar	wrf_neoverse-512tvb.exe
0.7%	1.2%		0.4%		__module_bl_myjpbl_MOD_myjpbl	wrf_neoverse-512tvb.exe
0.7%	0.7%				longest_match	libz.so.1.2.7
0.7%	6.8%		6.1%		powf	libm-2.26.so
0.7%	0.8%		0.1%		__module_small_step_em_MOD_sumflux	wrf_neoverse-512tvb.exe
0.6%	0.6%				__module_big_step_utilities_em_MOD_horizontal_pressure_gradient	wrf_neoverse-512tvb.exe
0.6%	0.6%				__module_big_step_utilities_em_MOD_curvature	wrf_neoverse-512tvb.exe
0.6%	1.1%		0.5%		__pow_finite	libm-2.26.so
0.5%	0.5%				__aarch64_swp4_rel	libc-2.26.so
0.5%	1.0%		0.5%		__module_small_step_em_MOD_small_step_prep	wrf_neoverse-512tvb.exe
0.5%	0.6%		<0.1%		__module_big_step_utilities_em_MOD_calc_cq	wrf_neoverse-512tvb.exe
0.5%	0.5%				__exp1	libm-2.26.so

https://community.arm.com/arm-community-blogs/b/high-performance-computing-blog/posts/bringing-wrf-up-to-speed-with-arm-neoverse

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ever heard about IEEE754?

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Numerical precision

- 64-bit floating point numbers
- x87 instructions – beware of 80-bits of precision!
- FMA – what is it all about?
- Example: A WRF story…

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Graviton getting started guide on GitHub

- This guide has been assembled by our Graviton team and is designed to help customers transition and optimize their applications https://github.com/aws/aws-graviton-getting-started

- It covers various languages and libraries and includes tips and tricks for each

- In general, using the latest versions of operating systems, compilers, and language runtimes provides access to the latest Arm64 improvements and optimizations

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Karthik Raman
akraman@amazon.com

Arthur Petitpierre
arthurpt@amazon.com
@ArthurPtP



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.