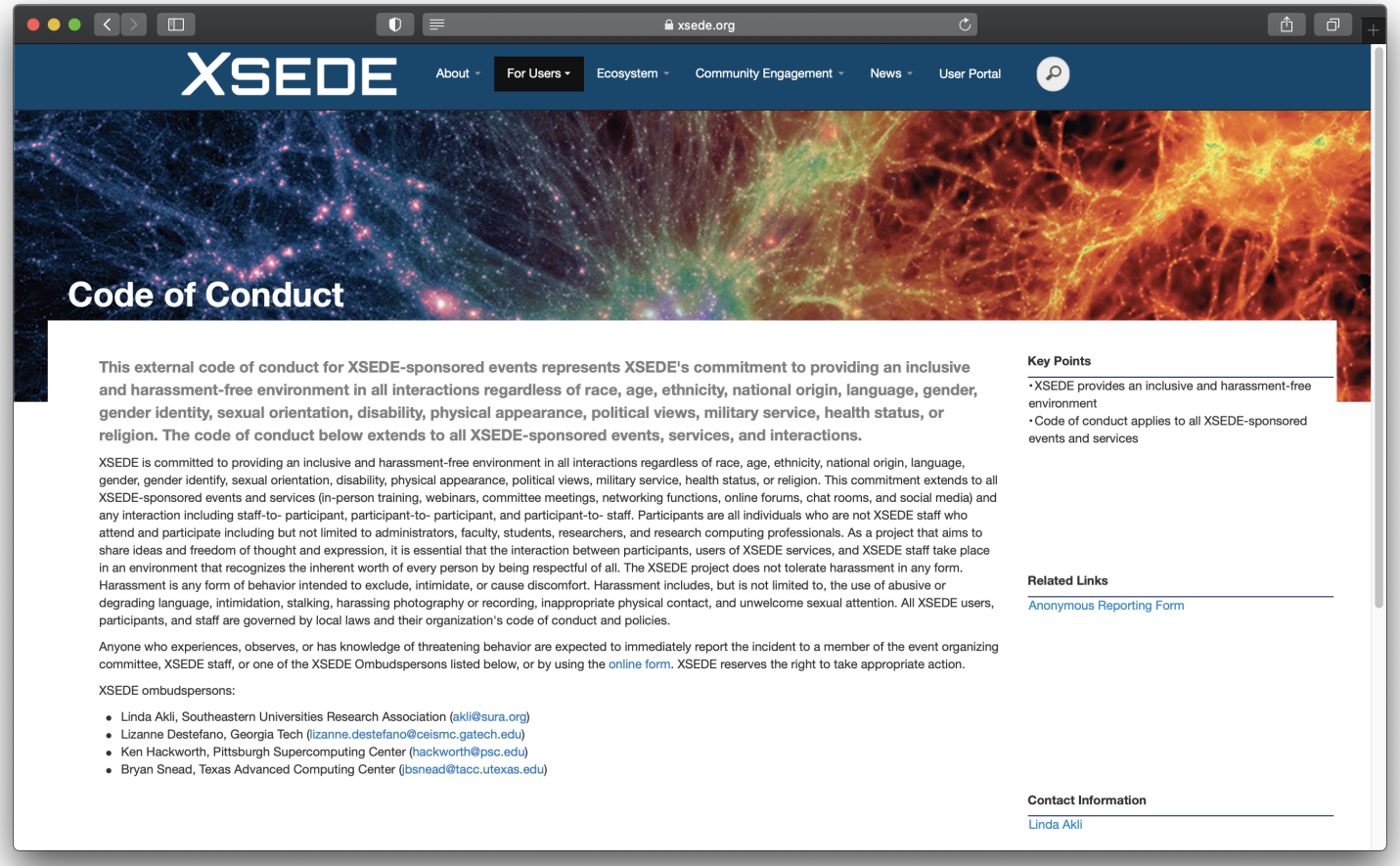
An Introduction to the AMD Optimizing C/C++ Compiler (AOCC) and the AMD Optimizing CPU Libraries (AOCL)

Marty Kandes, Ph.D.

Computational & Data Science Research Specialist
High-Performance Computing User Services Group
San Diego Supercomputer Center
University of California, San Diego

XSEDE Webinar
Wednesday, April 21st, 2021
09:45 AM - 10:35 AM PST

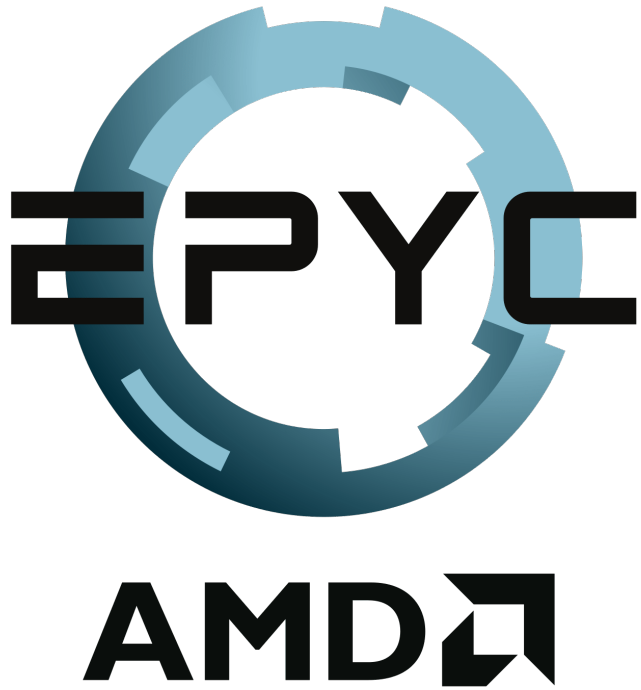
XSEDE Code of Conduct



https://www.xsede.org/codeofconduct

Today

- AMD Optimizing C/C++ Compiler
  - Compiler Options: gpecn1Dtdpbc
- AMD Optimizing CPU Libraries
  - AMD Math Library: savage
  - AMD BLIS: pave; dgemm
- Summary & References
- Q & A



AMD Optimizing C/C++ Compiler (AOCC)

AOCC is a high performance, optimized set of LLVM-based compilers. They provides developers with all of the the essential tools to build and optimize C, C++, and Fortran applications targeting AMD processors. Some features include:

- Enabled, tuned and optimized for AMD EPYC architectures
- Based on LLVM Compiler Infrastructure
- Flang is default Fortran front-end with F2008, Real 128 features
- OpenMP support
- RHEL 8, CentOS 8, SLES 15, and Ubuntu 20.04 LTS
- Spack support

https://developer.amd.com/amd-aocc

LLVM

The LLVM Project is a collection of modular and reusable compiler and toolchain technologies. Its core libraries provide a modern source- and target-independent optimizer, along with code generation support for many popular CPUs architectures. These libraries are built around a well specified code representation known as the LLVM intermediate representation ("LLVM IR").

https://llvm.org

Clang

The Clang project provides a language front-end and tooling infrastructure for languages in the C language family for the LLVM project. It is considered to be a production quality C, Objective-C, C++ and Objective-C++ compiler when targeting X86-32, X86-64, and ARM architectures. Clang supports C++11, C++14 and C++17.

https://clang.llvm.org

Flang

Flang (also known as "Classic Flang") is a Fortran compiler targeting LLVM. It is an open-sourced version of pgfortran, a commercial Fortran compiler from PGI/NVIDIA. It is different from the new Flang (formerly known as "F18"; see https://flang.llvm.org), which has been part of the LLVM project since 2020, although both are developed by the same community. It is also unrelated to other projects of the same name, such as https://github.com/llvm-flang/flang and https://github.com/isanbard/flang.

Classic Flang is used in several downstream commercial projects, and continues to be maintained, but the plan is to replace Classic Flang with the new Flang in the future.

https://github.com/flang-compiler/flang

AOCC Compiler Options

AOCC compiler (with Flang - Fortran Front-End)
Latest release: 2.3, Dec 2020
https://developer.amd.com/amd-aocc/

Architecture	
Generate instructions that run on 2nd Gen EPYC/ RYZEN	-march=znver2
Generate instructions for the local machine	-march=native
Optimization Levels	
Disable all optimizations	-O0
Minimal level speed and code optimization	-O1
Moderate level optimization (default)	-O2/ -O
Aggressive optimizations	-O3
Maximize performance	-Ofast
Enable Link Time Optimization	-flto
Enable unrolling	-funroll-loops
Enable aggressive loop optimizations	-enable-loop-versioning-licm -enable-partial-unswitch -unroll-aggressive
Enable aggressive inline optimizations	-function-specialize -finline-aggressive
Enable aggressive vectorization	-enable-strided-vectorization -enable-epilog-vectorization
Enable memory layout optimizations	-fremap-arrays (use with -flto)
Profile Guided optimizations	-fprofile-instr-generate (1st invocation) -fprofile-instr-use (2nd invocation)
OpenMP®	-fopenmp
For enabling streaming stores, memory bandwidth workloads	-fnt-store
Enable removal of un-used array computation	-reduce-array-computation=3

Other options	
Enable faster, less precise math operations	-ffast-math -freciprocal-math
OpenMP® threads and affinity (N number of cores)	export OMP_NUM_THREADS=N export GOMP_CPU_AFFINITY="0-{N-1)"
Enabling Vector library	-vector-library=LIBMVEC
Link to Vector library	-L/libm-install-dir/lib -lmvec
Link to AMD library	-L/libm-install-dir/lib -lamdlibm

For Fortran workloads	
Compile free form FORTRAN	-ffree-form
Enable precise math operations in FORTRAN	-kieee

AMD Optimized Libraries
Latest release: 2.2, Jun 2020
https://developer.amd.com/amd-aocl/

AMD µProf (Performance & Power Profiler)
Latest release: 3.3, Jul 2020
https://developer.amd.com/amd-uprof/

GNU compiler collection (gcc, g++, gfortran)
Latest release: 10.3, Jul 2020
Recommended version : 9.3
http://gcc.gnu.org

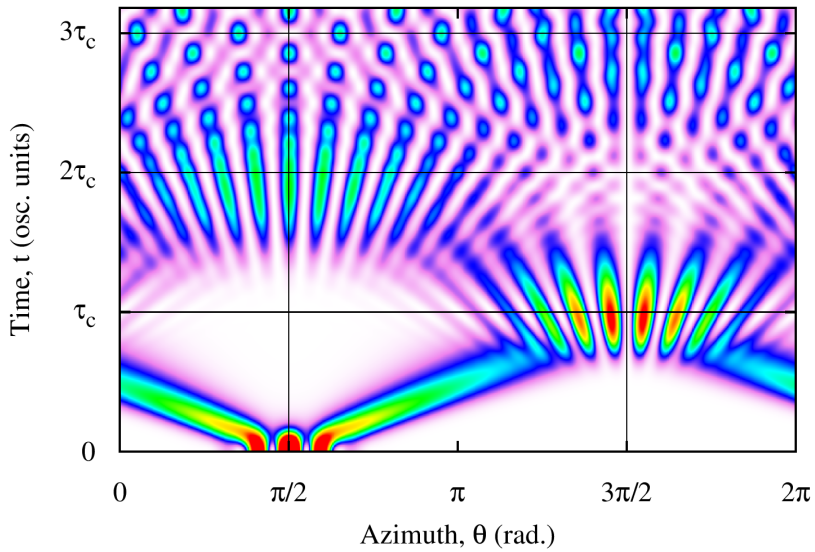
Architecture	
Generate instructions that run on 2nd Gen EPYC/ RYZEN	-march=znver2
Generate instructions for the local machine	-march=native
Optimization Levels	
Disable all optimizations (default)	-O0
Minimal level speed and code optimizations	-O1/ -O
Moderate level optimizations	-O2
Aggressive optimizations	-O3
Maximize performance	-Ofast
Additional Optimizations	
Link time optimization	-flto
Enable unrolling	-funroll-all-loops
Generate memory preload instructions	-fprefetch-loop-arrays --param prefetch-latency=300
Profile-guided optimization	-fprofile-generate (1st invocation) -fprofile-use (2nd invocation)
OpenMP®	-fopenmp
Other options	
Enable generation of code that follows IEEE arithmetic	-mieee-fp
Enable faster, less precise math operations	-ffast-math
Compile free form FORTRAN	-ffree-form
OpenMP® threads and affinity (N number of cores)	export OMP_NUM_THREADS=N export GOMP_CPU_AFFINITY="0-(N-1)"
Link to AMD library	-L/libm-install-dir/lib -lamdlibm

Example: A Real-World Science Code's First Test Drive of AOCC

gpecn1Dtdpbc: Overview

- Gross-Pitaevskii Equation (GPE)
- Crank-Nicolson (CN) scheme; 2nd-order central differences
- One-Dimensional (1D) ring topology
- Time-Dependent (TD)
- Periodic Boundary Conditions (PBC); Sherman-Morrison algorithm

```
78        PROGRAM GPECN1DTDPBC
79        IMPLICIT NONE
80
81  !     Parameter Declarations:
82  !     -----------------------
83        REAL, PARAMETER :: PI = 3.14159265358979323846264338327950288842E0
84
85  !     Input Variable and Array Declarations:
86  !     --------------------------------------
87        INTEGER :: NUMBER_OF_GRID_POINTS
88        INTEGER :: NUMBER_OF_TIME_STEPS
89        INTEGER :: NUMBER_OF_TIME_STEPS_BEFORE_WRITE
90        INTEGER :: NUMBER_OF_ITERATIONS_PER_TIME_STEP
91        INTEGER :: IMAGINARY_TIME_SWITCH
92
93        REAL :: RADIUS
94        REAL :: ROTATION_RATE
95        REAL :: GAMMA_FACTOR
96
97        COMPLEX, ALLOCATABLE, DIMENSION(:) :: WAVEFUNCTION
98        COMPLEX, ALLOCATABLE, DIMENSION(:) :: EXTERNAL_POTENTIAL
99        COMPLEX, ALLOCATABLE, DIMENSION(:) :: NONLINEAR_COUPLING
100
```



https://github.com/mkandes/gpecn1Dtdpbc

$$i\frac{\partial}{\partial t}\psi(\theta,t) = \left[-\frac{1}{2R^2}\frac{\partial^2}{\partial\theta^2} - \Omega L_z + V(\theta,t) + g_s|\psi(\theta,t)|^2\right]\psi(\theta,t)$$

gpecn1Dtdpbc: Computational Features & Operations

- Fortran 90
- Serial, single-core
- Double precision, complex arrays
- Element-wise array operations
- Intrinsics: CMPLX, CONJG, SUM
- LAPACK: ZGTSV

gpecn1Dtdpbc — vi Makefile — 80×45

```
#    Specify general user-defined compilation options.

COMPILER     := gfortran
OPTIMIZATION := ON
OPENMP       := OFF
DEBUG        := OFF
PROFILE      := OFF

#    Set compiler-specific options.

ifeq ($(COMPILER),gfortran)

   STANDARD_OPTIONS     := -fimplicit-none -fmodule-private \
                           -ffree-form -ffree-line-length-none -std=gnu
   INTEGER_OPTIONS      := -fdefault-integer-8
   REAL_OPTIONS         := -fdefault-real-8 -fdefault-double-8
   OPTIMIZATION_OPTIONS := -O3 -ffast-math -mtune=native
   OPENMP_OPTIONS       := -fopenmp
   CHECK_OPTIONS        := -fcheck=all
   DEBUG_OPTIONS        := -ffpe-trap=invalid,overflow -fbacktrace \
                           -fdump-core -finit-real=nan
   WARNING_OPTIONS      := -Wall -fmax-errors=0 -Wno-array-temporaries \
                           -Warray-bounds -Wcharacter-truncation \
                           -Wline-truncation -Wconversion-extra \
                           -Wimplicit-interface -Wimplicit-procedure \
                           -Wunderflow -Wextra -Wuninitialized
   PROFILE_OPTIONS      := -pg

endif

ifeq ($(COMPILER),ifort)

   STANDARD_OPTIONS     := -implicitnone -free -stand none -module build
   INTEGER_OPTIONS      := -integer-size 64
   REAL_OPTIONS         := -real-size 64 -double-size 64
   OPTIMIZATION_OPTIONS := -O3 -fast -mtune=native
   OPENMP_OPTIONS       := -qopenmp
   CHECK_OPTIONS        := -check all
   DEBUG_OPTIONS        := -debug all
   WARNING_OPTIONS      := -warn all
   PROFILE_OPTIONS      := -pg

endif
```

gpecn1dtdpbc: Runtime Performance

ARCH	COMPILER	MIN (s)	MAX (s)	AVG (s)	STDEV (s)
EPYC 7742	gfortran 8.3.1	897.5	1001.3	949.0	52.9
	gfortran 10.2.0	1028.4	1098.0	1080.1	27.4
	ifort 19.1.1.217	1445.8	1669.1	1529.1	76.9
	aocc/flang 2.2.0	1797.7	1980.0	1907.6	85.3
Xeon Gold 6248	ifort 19.0.5.281	795.5	873.8	810.2	22.8

```
gfortran -fimplicit-none -fmodule-private -ffree-form -ffree
    -line-length-none -std=gnu -fdefault-integer-8 -fdefault
    -real-8 -fdefault-double-8 -O3 -ffast-math -mtune=native

ifort -implicitnone -free -stand none -module build -integer
    -size 64 -real-size 64 -double-size 64 -O3 -fast -mtune=
    native

flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-
    real-8 -O3 -ffast-math -mtune=native
```

gpecn1dtdpbc: Shared (L3) Cache Contention

```
[mkandes@exp-2-02 ~]$ top

top - 20:27:21 up 21 days, 11:48,  1 user,  load average: 2.47, 2.09, 3.19
Tasks: 1887 total,   4 running, 1883 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.8 us,  0.2 sy,  0.0 ni, 96.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257517.8 total, 213682.0 free,  15570.0 used,  28265.8 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 238233.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 108113 zhanglab  20   0  985924 256272   1260 R  99.7   0.1   0:51.38 cas_S609086_2A_
 109397 zhanglab  20   0  985924 256272   1260 R  99.7   0.1   0:40.78 cas_S609086_3A_
 130249 umai      20   0   31216  16136   7996 R  99.7   0.0  70:36.43 python
  18444 telegraf  20   0   11.4g 118404  42748 S  10.3   0.0 800:32.14 telegraf
  88089 zhanglab  20   0 1055448 659344    704 S   3.6   0.3   3:02.54 zalign
 111433 zhanglab  20   0   79512  26900   3616 S   2.0   0.0   0:00.40 scontrol
  17801 nslcd     20   0  456856   6616   4168 S   1.7   0.0 669:08.67 nslcd
 113079 mkandes   20   0   67996   6732   3920 R   1.0   0.0   0:00.11 top
  70848 robpearc  20   0   58936  27672   4560 S   0.7   0.0   0:58.93 perl
   4081 root      20   0       0      0      0 I   0.3   0.0   0:00.10 kworker/30:2-mm_
   4196 robpearc  20   0   58936  27672   4560 S   0.3   0.0   0:10.08 perl
   4950 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:28.80 perl
   7076 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:40.41 perl
  10474 root      20   0  210876  33468  31344 S   0.3   0.0 150:13.45 sssd_nss
  13678 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:35.03 perl
  17919 root      20   0       0      0      0 S   0.3   0.0  26:12.17 kiblnd_sd_01_03
  17920 root      20   0       0      0      0 S   0.3   0.0   2:07.97 kiblnd_sd_02_00
  18137 root      20   0       0      0      0 S   0.3   0.0   0:42.71 ptlrpcd_02_05
  18423 root      20   0 3900788 772424 106932 S   0.3   0.3 164:41.60 cmd
  21447 robpearc  20   0   58936  27672   4560 S   0.3   0.0   0:19.58 perl
  22225 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:52.46 perl
  29138 robpearc  20   0   58944  27680   4560 S   0.3   0.0   1:10.16 perl
  42210 robpearc  20   0   58944  27680   4560 S   0.3   0.0   0:13.92 perl
  51239 robpearc  20   0   58940  27676   4560 I   0.3   0.0   0:37.51 perl
  81610 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:03.83 perl
  84081 robpearc  20   0   58944  27680   4560 S   0.3   0.0   0:38.35 perl
  88209 robpearc  20   0   58936  27672   4560 I   0.3   0.0   0:32.31 perl
  96322 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:20.54 perl
 117902 robpearc  20   0   58940  27676   4560 S   0.3   0.0   0:43.26 perl
 129029 root      20   0  279936   7076   5080 S   0.3   0.0   0:01.10 slurmstepd
      1 root      20   0  104340  12880   4436 S   0.0   0.0  14:59.13 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:18.94 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kbl
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root      20   0       0      0      0 S   0.0   0.0   0:49.80 ksoftirqd/0
     11 root      20   0       0      0      0 I   0.0   0.0  46:03.17 rcu_sched
     12 root      rt   0       0      0      0 S   0.0   0.0   0:00.64 migration/0
     13 root      rt   0       0      0      0 S   0.0   0.0   0:00.04 watchdog/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     16 root      rt   0       0      0      0 S   0.0   0.0   0:00.24 watchdog/1
```

1798.7 s

```
[mkandes@exp-2-14 ~]$ top

top - 20:24:40 up 40 days, 22:06,  1 user,  load average: 60.66, 63.40, 66.65
Tasks: 1871 total,  59 running, 1812 sleeping,   0 stopped,   0 zombie
%Cpu(s): 45.3 us,  1.5 sy,  0.0 ni, 52.8 id,  0.0 wa,  0.4 hi,  0.1 si,  0.0 st
MiB Mem : 257517.8 total, 161078.2 free,  55465.6 used,  40974.0 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 194930.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 103137 huantran  20   0 6647996 358348 160876 R 101.7   0.1   3:51.95 vasp_gam
  11796 misbaazn  20   0 1922884   1.1g  32628 R 100.0   0.4 1941:27 vasp_std
  11797 misbaazn  20   0 1922536   1.1g  32808 R 100.0   0.4 1955:58 vasp_std
  11799 misbaazn  20   0 1922588   1.1g  32464 R 100.0   0.4 1945:20 vasp_std
  11800 misbaazn  20   0 1922644   1.1g  32448 R 100.0   0.4 1955:59 vasp_std
  11801 misbaazn  20   0 1922648   1.1g  32492 R 100.0   0.4 1944:12 vasp_std
  11803 misbaazn  20   0 1922624   1.1g  32648 R 100.0   0.4 1937:59 vasp_std
  11805 misbaazn  20   0 1922588   1.1g  32544 R 100.0   0.4 1942:16 vasp_std
  11811 misbaazn  20   0 1922468   1.1g  32924 R 100.0   0.4 1940:21 vasp_std
  11812 misbaazn  20   0 1922488   1.1g  32920 R 100.0   0.4 1939:12 vasp_std
  11815 misbaazn  20   0 1922344   1.1g  32464 R 100.0   0.4 1944:15 vasp_std
  11817 misbaazn  20   0 1922516   1.1g  32328 R 100.0   0.4 1944:23 vasp_std
  11818 misbaazn  20   0 1922972   1.1g  32356 R 100.0   0.4 1943:21 vasp_std
  11819 misbaazn  20   0 1922428   1.1g  32484 R 100.0   0.4 1944:06 vasp_std
  11820 misbaazn  20   0 1922716   1.1g  32324 R 100.0   0.4 1944:02 vasp_std
  11822 misbaazn  20   0 1922444   1.1g  32340 R 100.0   0.4 1945:15 vasp_std
  11824 misbaazn  20   0 1922440   1.1g  32524 R 100.0   0.4 1943:57 vasp_std
  56471 umai      20   0   31220  16144   8012 R 100.0   0.0  67:50.37 python
  56599 umai      20   0   31092  16432   8032 R 100.0   0.0  67:52.37 python
  56649 umai      20   0   31224  16176   8032 R 100.0   0.0  67:51.35 python
  56735 umai      20   0   31200  16168   8032 R 100.0   0.0  67:52.31 python
  56847 umai      20   0   31228  16192   8032 R 100.0   0.0  67:51.53 python
  56869 umai      20   0   31212  16156   8032 R 100.0   0.0  67:52.39 python
 103122 huantran  20   0 6654508 395848 192184 R 100.0   0.2   3:46.66 vasp_gam
 103124 huantran  20   0 6652316 399592 198812 R 100.0   0.2   3:52.32 vasp_gam
 103125 huantran  20   0 6660664 401128 197884 R 100.0   0.2   3:52.35 vasp_gam
 103126 huantran  20   0 6660728 381492 179848 R 100.0   0.1   3:51.27 vasp_gam
 103127 huantran  20   0 6655508 375232 176480 R 100.0   0.1   3:51.41 vasp_gam
 103128 huantran  20   0 6656552 377920 179184 R 100.0   0.1   3:51.38 vasp_gam
 103131 huantran  20   0 6659500 359788 162440 R 100.0   0.1   3:51.32 vasp_gam
 103133 huantran  20   0 6646976 356752 159200 R 100.0   0.1   3:51.34 vasp_gam
 103134 huantran  20   0 6658460 358768 161684 R 100.0   0.1   3:51.99 vasp_gam
 103135 huantran  20   0 6654284 359692 161520 R 100.0   0.1   3:51.96 vasp_gam
 103136 huantran  20   0 6645932 360844 163168 R 100.0   0.1   3:51.97 vasp_gam
 118507 zhanglab  20   0  985924 266552   1260 R 100.0   0.1 161:37.19 cas_S609019_4F_
 128072 zhanglab  20   0  985924 217724   1260 R 100.0   0.1 748:59.35 cas_S608916_5F_
  11792 misbaazn  20   0 1945800   1.1g  31772 R  98.3   0.5 1944:03 vasp_std
  11798 misbaazn  20   0 1922576   1.1g  32732 R  98.3   0.4 1944:05 vasp_std
  11802 misbaazn  20   0 1922756   1.1g  32532 R  98.3   0.4 1944:40 vasp_std
  11804 misbaazn  20   0 1922864   1.1g  32584 R  98.3   0.4 1944:24 vasp_std
  11806 misbaazn  20   0 1922572   1.1g  32520 R  98.3   0.4 1937:40 vasp_std
  11807 misbaazn  20   0 1922820   1.1g  32712 R  98.3   0.4 1944:11 vasp_std
  11808 misbaazn  20   0 1922592   1.1g  32264 R  98.3   0.4 1938:45 vasp_std
```

1980.0 s

gpecn1dtdpbc: Compiler Options & Build Warnings

```
flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-real-8 -O3 -ffast-math
      -mtune=native -c source/gpecn1Dtdpbc.f -o build/gpecn1Dtdpbc.o
F90-W-0155-The type of FLOAT is now double precision with -r8  (source/
     gpecn1Dtdpbc.f: 166)
  0 inform,   1 warnings,   0 severes, 0 fatal for gpecn1dtdpbc
flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-real-8 -O3 -ffast-math
      -mtune=native -c libraries/lapack/zgtsv.f -o build/zgtsv.o
F90-W-0173-PGI Fortran extension: nonstandard use of data type length specifier
     (libraries/lapack/zgtsv.f: 11)
F90-W-0173-PGI Fortran extension: nonstandard use of data type length specifier
     (libraries/lapack/zgtsv.f: 71)
F90-W-0173-PGI Fortran extension: nonstandard use of data type length specifier
     (libraries/lapack/zgtsv.f: 76)
  0 inform,   3 warnings,   0 severes, 0 fatal for zgtsv
flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-real-8 -O3 -ffast-math
      -mtune=native -c libraries/lapack/xerbla.f -o build/xerbla.o
flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-real-8 -O3 -ffast-math
      -mtune=native build/gpecn1Dtdpbc.o build/zgtsv.o build/xerbla.o -o
     gpecn1Dtdpbc.x
clang-10: warning: argument unused during compilation: '-Mfreeform' [-Wunused-
     command-line-argument]
clang-10: warning: argument unused during compilation: '-Mstandard' [-Wunused-
     command-line-argument]
clang-10: warning: argument unused during compilation: '-fdefault-integer-8' [-
     Wunused-command-line-argument]
clang-10: warning: argument unused during compilation: '-fdefault-real-8' [-
     Wunused-command-line-argument]
```

AMD Optimizing CPU Libraries (AOCL)

AOCL is a set of numerical libraries tuned specifically for ADM EPYC processor family. They have a simple interface to take advantage of the latest hardware innovations. The tuned implementations of industry standard math libraries enable fast development of scientific and high performance computing projects.

- AMD Math Library (LibM)
- AMD BLIS, libFLAME, ScaLAPACK, AOCL-Sparse
- AMD FFTW
- AMD Random Number Generator and Secure RNG Library

https://developer.amd.com/amd-aocl

AMD Math Library (LibM)

AMD LibM is a software library containing a collection of basic math functions optimized for x86-64 processor based machines. It provides many routines from the list of standard C99 math functions. AMD LibM is a C library, which users can link in to their applications to replace compiler-provided math functions. Generally, programmers access basic math functions through their compiler, but those who want better accuracy or performance than their compiler's math functions can use this library to help improve their applications.

https://developer.amd.com/amd-aocl/amd-math-library-libm
https://github.com/amd/aocl-libm-ose

AMD BLIS (BLAS) Library

BLIS is a portable software framework for instantiating high-performance BLAS-like dense linear algebra libraries. The framework was designed to isolate essential kernels of computation that, when optimized, enable optimized implementations of most of its commonly used and computationally intensive operations. Select kernels have been optimized for the AMD EPYC processor family. The optimizations are done for single and double precision routines.

https://developer.amd.com/amd-aocl/blas-library
https://github.com/amd/blis

AMD libFLAME

libFLAME is a portable library for dense matrix computations, compatible with Netlib LAPACK specification. It includes a compatibility layer, FLAPACK, which includes complete LAPACK implementation. The library provides scientific and numerical computing communities with a modern, high-performance dense linear algebra library that is extensible, easy to use, and available under an open source license.

libFLAME is a C-only implementation and does not depend on any external FORTRAN libraries including LAPACK. There is an optional backward compatibility layer, lapack2flame that maps LAPACK routine invocations to their corresponding native C implementations in libFLAME. This allows legacy applications to start taking advantage of libFLAME with virtually no changes to their source code.

https://github.com/amd/libflame

AMD ScaLAPACK

ScaLAPACK is a library of high-performance linear algebra routines for parallel distributed memory machines. It depends on external libraries including BLAS and LAPACK for Linear Algebra computations. AMD's optimized version of ScaLAPACK enables using BLIS and libFLAME library that have optimized dense matrix functions and solvers for AMD EPYC processor family CPUs.

https://developer.amd.com/amd-aocl/scalapack
https://github.com/amd/scalapack

AMD AOCL-Sparse

AOCL-Sparse is a library that contains basic linear algebra subroutines for sparse matrices and vectors optimized for AMD EPYC family of processors. It is designed to be used with C and C++. Current functionality of sparse library supports SPMV function with CSR and ELLPACK formats.

https://developer.amd.com/amd-aocl/aocl-sparse
https://github.com/amd/aocl-sparse

AMD FFTW

FFTW is a comprehensive collection of fast C routines for computing the Discrete Fourier Transform (DFT) and various special cases thereof. It is an open-source implementation of the Fast Fourier transform algorithm. It can compute transforms of real and complex-values arrays of arbitrary size and dimension. AMD optimized FFTW includes selective kernels and routines optimized for the AMD EPYC processor family.

https://developer.amd.com/amd-aocl/fftw/
https://github.com/amd/amd-fftw

AMD Random Number Generator Libraries

AMD Random Number Generator (RNG) Library is a pseudorandom number generator library. It provides a comprehensive set of statistical distribution functions which are founded on various underlying uniform distribution generators (base generators) including Wichmann-Hill and Mersenne Twister. The library contains five base generators and twenty-three distribution generators. Users can also supply a custom built generator as the base generator for all of the distribution generators.

AMD Secure RNG is a library that provides APIs to access the cryptographically secure random numbers generated by AMD's hardware-based random number generator implementation.

https://developer.amd.com/amd-aocl/rng-library

Example: Trancendental Functions & AMD Math Library (LibM)

Savage Benchmark

16-Bit Toolbox (Text begins on page 120)

Listing One BASIC and PL-1 versions of Bill Savage's accuracy and speed test for high level languages.

```
100   ' Time and General Accuracy Test Program
110   DEFINT I
120   ILOOP=2500
130   A=1
140   FOR I=1 TO ILOOP-1
150     A = TAN(ATN(EXP(LOG(SQR(A*A))))) + 1
160   NEXT I
170   PRINT USING "A=####.####";A
180   STOP
```

"An interesting program to test the speed and accuracy of various high-level languages has been sent to us by Bill Savage of Microfloat in Houston, Texas ... The final result of each test should be 2500.000; any variations from this figure reflect errors in the language's floating-point algorithms or rounding methods. The arctangent in the central function is designed to accentuate such discrepancies and make them more visible."

- R. Duncan, "16-bit Software Toolbox", Dr. Dobb's Journal, Number 83, September 1983, p. 120

Savage Benchmark: Runtime Performance (circa 1983)

With hardware floating point libraries from MicroFloat:

Language	Version	Result	Time (sec.)
PL/I-86 with 8087	1.01	2477.244	3.7
8080 Assembler with 8232	RMAC	2499.955	10.2
PL/I-80 with 8232	1.40	2499.995	10.4
BASIC-80 with 8232	5.20	2500.000	10.7
Fortran-80 with 8232	3.40	2499.995	12.5

With standard floating point libraries:

Language	Version	Result	Time (sec.)
BASIC-86 Interpreter	5.20	2179.850	92.2
Fortran-80	3.40	2304.863	140.8
BASIC-80 Compiler	5.20	2304.860	140.8
BASIC-80 Interpreter	5.20	2304.860	174.9
PL/I-86	1.01	1641.758	179.6
PL/I-80	1.30	1641.758	254.4

Table 1.

Comparisons of execution speed and accuracy for commonly available high-level languages, with and without hardware floating point support. The 8087 and 8232 support libraries are marketed under the names FLOAT87 and FLOAT80, which are trademarks of MicroFloat.

Savage Benchmark: Runtime Performace (circa 2021)

Measured on Expanse's AMD EPYC 7742 compute nodes when linking against the standard libm distributed with CentOS 8.

			Runtime (s)	
n (M)	a(n)	gcc 10.2.0	icc 19.1.1.217	aocc/clang 2.2.0
1	999999.997712	0.118	0.071	0.116
10	9999816.516784	1.112	0.675	1.161
20	19996559.553870	520.955	1.338	581.503
50	49671498.766991		3.343	
100	94906263.696971		6.66	
1000	94906263.696971		66.61	

```
  gcc -O3 -mtune=native savage.c -o savagec.x -lm
  icc -O3 -mtune=native savage.c -o savagec.x -lm
clang -O3 -mtune=native savage.c -o savagec.x -lm
```

https://github.com/mkandes/savage

Savage Benchmark: Runtime Performace w/AMD LibM

Measured on Expanse's AMD EPYC 7742 compute nodes when linking against the AMD Math Library (LibM).

			Runtime (s)	
n (M)	a(n)	gcc 10.2.0	icc 19.1.1.217	aocc/clang 2.2.0
1	999999.997712	0.105	0.096	0.116
10	9999816.516784	1.024	0.945	1.047
20	19996559.553870	2.045	1.851	1.891
50	49671498.766991	5.102	4.723	4.547
100	94906263.696971	10.221	9.417	10.1
1000	94906263.696971	102.167	94.342	100.068

```
declare -xr AOCC_ROOT_DIR='/cm/shared/apps/spack/cpu/opt/
    spack/linux-centos8-zen/gcc-8.3.1/aocc-2.2.0-
    hikjjf4adelw56s3namnyvplezykf4fi'
  gcc -O3 -mtune=native savage.c -o savagec.x "-I${
    AOCC_ROOT_DIR}/include" "-L${AOCC_ROOT_DIR}/lib" -
    lamdlibm -lm
  icc -O3 -mtune=native savage.c -o savagec.x "-I${
    AOCC_ROOT_DIR}/include" "-L${AOCC_ROOT_DIR}/lib" -
    lamdlibm -lm
clang -O3 -mtune=native savage.c -o savagec.x "-I${
    AOCC_ROOT_DIR}/include" "-L${AOCC_ROOT_DIR}/lib" -
    lamdlibm -lm
```

Savage Benchmark: Other Observations & Assumptions

`-ffast-math` likely simplifies savage expression at compile time

```
gcc -O3 -ffast-math -mtune=native savage.c -o savagec.x -lm
time -p ./savagec.x 1000000000
n = 1000000000
a = 1000000000.465963
real 46.98
user 46.82
sys 0.00
```

fixed point at high iteration count is due to 64-bit limit

```
gfortran -ffree-form -ffree-line-length-none -fimplicit-none
     -O3 -mtune=native -fdefault-integer-8 -fdefault-real-16
     savage.f90 -o savagef.x -lm
time -p ./savagef.x 1000000000
 n = 1000000000
 a =  999999999.99999998554581143508881487 8
real 3318.56
user 3302.81
sys 0.00
```

Example: Matrix Multiplication & AMD BLIS

PAVE Benchmarks

A collection of linear algebra benchmarks using the Hilbert matrix, a square matrix with elements that are unit fractions given by

$$H_{ij} = \frac{1}{i + j - 1}$$

For example, the Hilbert matrix of dimension 4 is

$$\mathrm{H} = \begin{bmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{bmatrix}$$

https://github.com/mkandes/pave

PAVE Benchmark: HxH

Compute the square of the Hilbert matrix of dimension n.

$$H_{ij}^2 = \sum_{k=1}^{n} H_{ik} H_{kj}$$

It can be shown that the first element of resultant matrix in the limit of large n approaches the summation of the reciprocals of the squares of the natural numbers. See Basel problem.

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} \approx 1.644934$$

PAVE Benchmark: HxH - MATMUL

COMPILER	H_{11}^2	AVG (s)	STDEV (s)
gfortran 10.2.0	1.6448730335545843	649.636	26.123
ifort 19.1.1.217	1.64487303355459	4432.478	881.341
aocc/flang 2.2.0	1.644873033554583	1430.356	400.068

```
gfortran -ffree-form -ffree-line-length-none -fimplicit-none
     -O3 -ffast-math -mtune=native -fdefault-integer-8 -
    fdefault-real-8 -fdefault-double-8 hxh-matmul.f90 -o hxh
    -matmul.x

ifort -implicitnone -free -stand none -module build -integer
    -size 64 -real-size 64 -double-size 64 -O3 -fast -mtune=
    native hxh-matmul.f90 -o hxh-matmul.x

flang -Mfreeform -Mstandard -fdefault-integer-8 -fdefault-
    real-8 -O3 -ffast-math -mtune=native hxh-matmul.f90 -o
    hxh-matmul.x
```

PAVE Benchmark: HxH - DGEMM

gfortran 10.2.0

BLAS	H_{11}^2	AVG (s)	STDEV (s)
Netlib 3.9.1	1.6448730335545856	3477.714	2101.324
OpenBLAS 0.3.10	1.6448730335545845	250.870	39.788
Intel MKL 2019.1.144	1.6448730335545829	206.712	32.682
AMD BLIS 2.2	1.6448730335545845	244.464	54.536

```
gfortran -ffree-form -ffree-line-length-none -fimplicit-none
     -O3 -ffast-math -mtune=native -fdefault-integer-8 -
    fdefault-real-8 -fdefault-double-8 hxh-dgemm.f90 -o hxh-
    dgemm.x -lopenblas

gfortran -ffree-form -ffree-line-length-none -fimplicit-none
     -O3 -ffast-math -mtune=native -fdefault-integer-8 -
    fdefault-real-8 -fdefault-double-8 hxh-dgemm.f90 -o hxh-
    dgemm.x -lmkl_avx2 -lmkl_core -lmkl_gf_lp64 -
    lmkl_sequential

gfortran -ffree-form -ffree-line-length-none -fimplicit-none
     -O3 -ffast-math -mtune=native -fdefault-integer-8 -
    fdefault-real-8 -fdefault-double-8 hxh-dgemm.f90 -o hxh-
    dgemm.x "-L${AOCL_ROOT_DIR}/lib" -lblis -lpthread
```

Conclusions and Summary

- AMD is developing its own C/C++ and Fortran compiler infrastrucuture built on the LLVM toolchain.
- A number of optimized math libraries for high-performance computing are already available and under active development.
- Where your code will find performance gains is difficult to say without testing, but you should to familiarize yourself with the AMD compilers and math libraries to make that assesment
- Next steps (for me): Learn how to use AMD LibFLAME, then retry with gpecn1Dtdpbc.

AMD References

- AOCC 2.2 User Guide
- AOCC 3.0 User Guide
- AOCC 2.3 Clang C/C++ Compiler
- AOCC 3.0 Clang C/C++ Compiler
- AOCC 2.3 Flang Fortran Compiler
- AOCC 3.0 Flang Fortran Compiler
- AOCL 2.2 User Guide
- AOCL 3.0 User Guide

AMD References

- AMD EPYC 7xx2-series Processor Compiler Options Quick Reference Guide
- https://developer.amd.com/amd-aocc
- https://developer.amd.com/amd-aocl
- https://developer.amd.com/resources/epyc-resources/epyc-tuning-guides
- https://developer.amd.com/resources/developer-guides-manuals
- https://github.com/amd

Additional References

- Partnership for Advanced Computing in Europe
- PRACE Best Practice Guide - AMD EPYC by Xu Guo

Questions?