Experiences Porting Mini-applications to OpenACC and OpenMP on Heterogeneous Systems

Verónica G. Vergara Larrea
Reuben D. Budiardja
Rahul Gayatri
Christopher Daley
Oscar Hernandez
Wayne Joubert

Outline

- Objective
- Application porting experiences
  - Minisweep
  - GenASIS
  - GPP
  - FF
- Conclusions
- Future work

Objectives

- Port four mini-applications to OpenACC and OpenMP 4.5 programming models.
- Evaluate the performance of individual versions on different architectures.
  - Titan, Summitdev, Summit, Cori, and Cori-GPU.
- Document challenges and issues encountered.

Target Systems

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

- **Titan:** Cray XK7
  - 18,688 compute nodes
  - 16-core AMD Opteron + 1 NVIDIA K20X GPU per node
- **Summit:** IBM AC922
  - 4,608 compute nodes
  - Two 22-core POWER9 + 6 NVIDIA V100 GPUs per node
- **Summitdev:** IBM P8+
  - 54 compute nodes
  - Two 10-core POWER8 + 4 NVIDIA P100 GPUs per node

NERSC

- **Cori:** Cray XC40
  - 2,388 multi-core nodes + 9,688 many-core nodes
  - Dual-socket 16-core Intel Haswell per multi-core node
  - One 68-core Intel KNL per many-core node
- **Cori-GPU:** Cray CS-Storm
  - 18 nodes development system
  - Dual-socket 20-core Intel Xeon Gold 'Skylake' + 8 NVIDIA V100 GPUs per node

Minisweep

- Part of the Profugus radiation transport proxy application project.
  - Simulates the sweep pattern used in Denovo S_n radiation transport application.
  - Used for nuclear reactor core analysis, nuclear forensics, radiation shielding, and radiation detection
- Original code developed at ORNL (W. Joubert).
  - Written in C with CUDA and OpenMP 3.1 support.
- OpenACC port available from University of Delaware (R. Searles et al.).

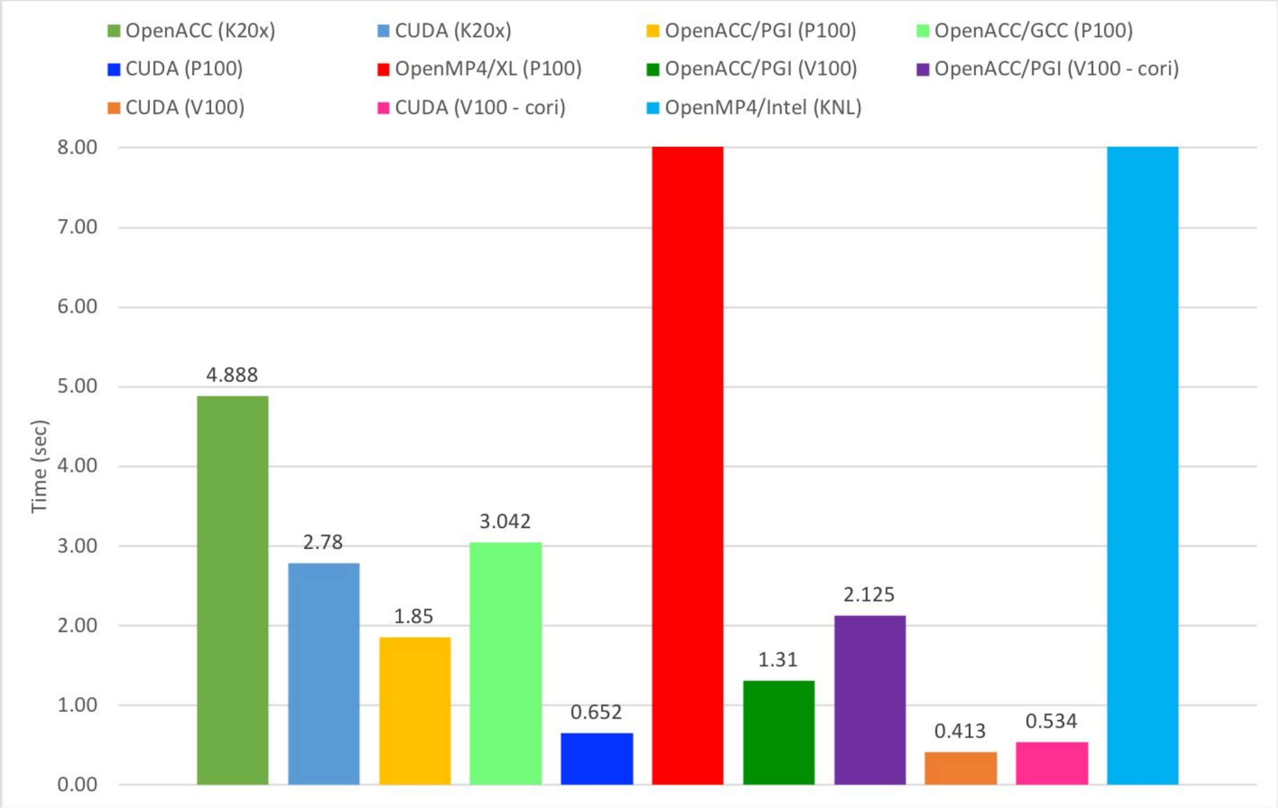
Minisweep compute kernels

- Sweep kernel accounts of >80% of the runtime in Denovo.
- Two main functions that are offloaded to the GPU:
  - `Sweep_sweep()`
  - `Sweep_in_gridcell()`
- Fairly straightforward transformations

OpenACC	**OpenMP 4.5**
`acc loop gang`	`omp teams distribute`
`acc loop vector`	`omp parallel for`
`acc loop seq`	None
`acc parallel`	`omp target`
`acc wait`	`omp taskwait`
`acc data copyout`	`omp data map(from:<var)`
`acc data copyin`	`omp data map(to:<var>)`
`acc data create`	`omp data map(alloc:<var>)`

Minisweep results (Lower is better)

NX x NY x NZ = 32 x 32 x 32; NM =16



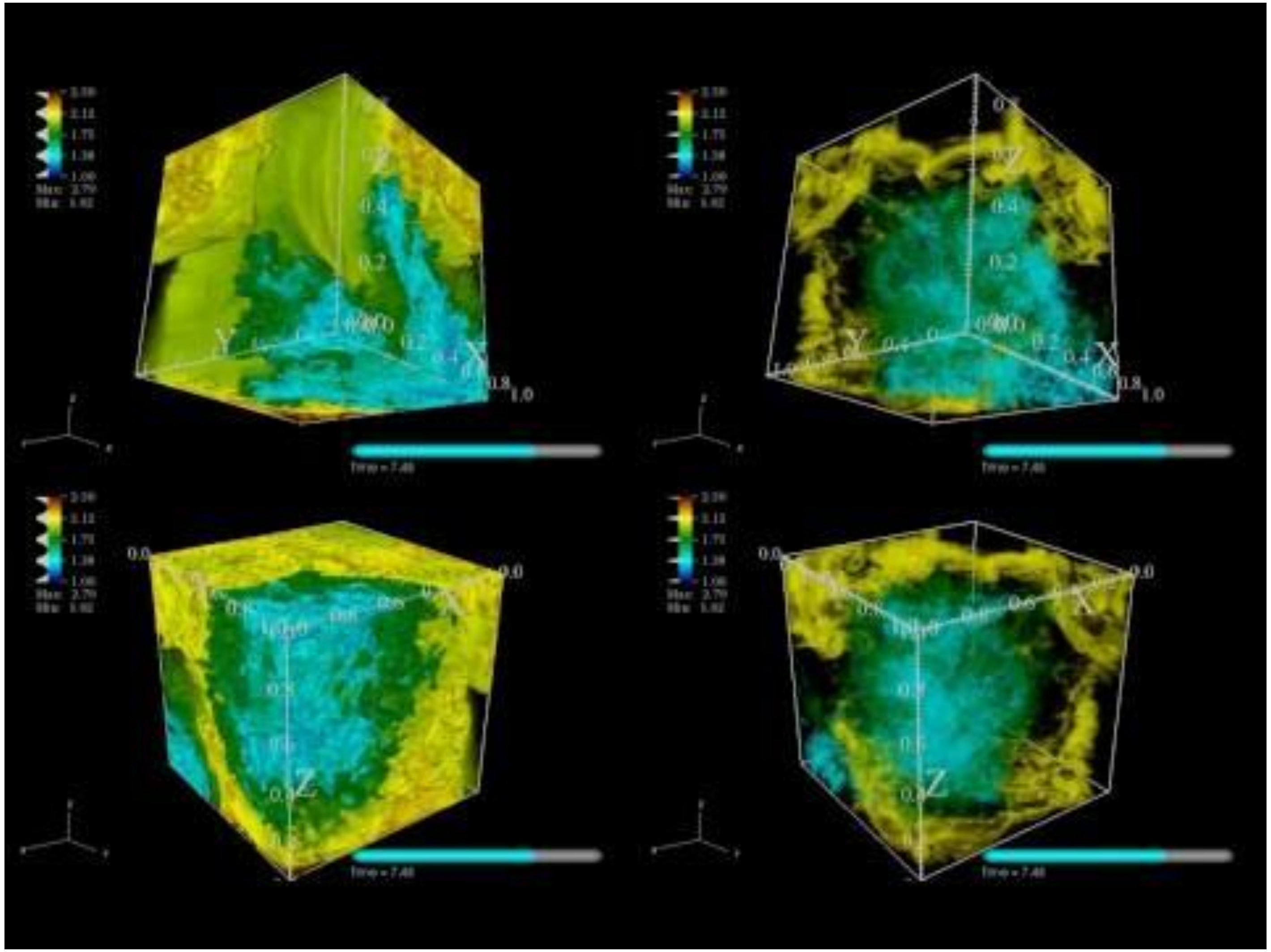
System	Port Version	Compiler	CUDA	Port status	Build status	Run status
Titan	OpenACC	PGI 18.4	9.1	✓	✓	✓
		CCE 8.6.4		✓	✓	❌
	CUDA	GCC 6.3.0		✓	✓	✓
	OpenMP4	CCE 8.6.4		🚧	❌	-
Summitdev	OpenACC	PGI 18.7	9.2	✓	✓	✓
		GCC 7.1.1		✓	✓	⚠️
	CUDA	GCC 6.3.1		✓	✓	✓
	OpenMP4	XL 16.1.1.0		🚧	✓	⚠️
Summit	OpenACC	PGI 18.10		✓	✓	✓
		GCC 8.1.1		✓	✓	⚠️
	CUDA	GCC 6.4.0		✓	✓	✓
	OpenMP4	XL 16.1.1.2		🚧	✓	-
Cori	OpenMP4	Intel 18	-	🚧	✓	⚠️
Cori-GPU	OpenACC	GCC 8.1.1	10.0	✓	❌	-
		PGI 18.10.1		✓	✓	✓
		CCE		✓	❌	-
	CUDA	GCC 7.6.0		✓	✓	✓
	OpenMP4	CCE		🚧	✓	❌
		Clang		🚧	✓	❌

GenASIS

- **Gen**eral **A**strophysical **Si**mulation **S**ystem
- Object-oriented, modular design in modern Fortran (2003, 2008)
- Three major subdivisions → allow for unit testings, mini-apps development
  - **Basics**: Utilitarians functionalities
    - I/O, *StorageForm* class, Devices (OpenMP & CUDA library wrappers), MPI facades, …
    - Solvers & physics are implemented within test program
  - **Mathematics:** Object-oriented Manifolds, Operations, and Solvers
    - Meshing (distributed mesh, manifolds, charts), PDE solvers, Time integration
    - Physics are implemented within test program
  - **Physics:** Stress-Energy, Universes, Spaces
    - Newtonian, GR, Fluids, Radiation
    - Full applications use all three subdivisions

GenASIS: Target Problem

View movie at: https://tinyurl.com/yya2bqwv



Lower-Level GenASiS Functionality

- Fortran wrappers to OpenMP / OpenACC APIs

```
– call AllocateDevice(Value, D_Value)
    → omp_target_alloc(), acc_malloc()

  call AssociateHost(D_Value, Value)
    → omp_target_associate_ptr(), acc_map_data()

  call UpdateDevice(Value, D_Value),
  call UpdateHost(Value, D_Value)
    → omp_target_memcpy(), acc_memcpy_{to,from}_device()
```

Value : Fortran array
D_Value : type(c_ptr), GPU address

- Affirmative control of data movement
- Persistent memory allocation on the device

Higher-level GenASiS Functionality

- StorageForm :
  - a class for data and metadata; the 'heart' of data storage facility in GenASiS
  - metadata includes units, variable names (for I/O, visualization)
  - used to group together a set of related physical variables (e.g. Fluid)
  - render more generic and simplified code for I/O, ghost exchange, prolongation & restriction (AMR mesh)
- Data: `StorageForm % Value ( nCells, nVariables )`
- Methods:
  - `call StorageForm % Initialize ( [`*`shape`*`] )` ← **allocate data on host**
  - `call StorageForm % AllocateDevice ( )` ← **allocate data on GPU**
  - `call StorageForm % Update{Device,Host} ( )` ← **transfer data**

Offloading Computational Kernel

```fortran
subroutine AddKernel ( A, B, D_A, D_B, D_C, C )

  real ( KDR ), dimension ( : ), intent ( in ) :: A, B
  type ( c_ptr ), intent ( in ) :: D_A, D_B, D_C
  real ( KDR ), dimension ( : ), intent ( out ) :: C

  integer ( KDI ) :: i

  call AssociateHost ( D_A, A )
  call AssociateHost ( D_B, B )
  call AssociateHost ( D_C, C )

  !$OMP target teams distribute parallel do schedule ( static, 1 )
  do i = 1, size ( C )
    C ( i ) = A ( i ) + B ( i )
  end do
  !$OMP end target teams distribute parallel do

  call DisassociateHost ( C )
  call DisassociateHost ( B )
  call DisassociateHost ( A )

end subroutine AddKernel
```

```fortran
call F % Initialize &
  ([nCells, nVariables])
call F % AllocateDevice ( )
call F % UpdateDevice ( )
call AddKernel &
  ( F % Value ( :, 1 ),
    F % Value ( :, 2 ), &
    F % D_Value ( 1 ),
    F % D_Value ( 2 ), &
    F % D_Value ( 3 ),
    F % Value ( :, 3 ) )
```

Offloading Computational Kernel

```fortran
subroutine AddKernel ( A, B, D_A, D_B, D_C, C )

  real ( KDR ), dimension ( : ), intent ( in ) :: A, B
  type ( c_ptr ), intent ( in ) :: D_A, D_B, D_C
  real ( KDR ), dimension ( : ), intent ( out ) :: C

  integer ( KDI ) :: i

  call AssociateHost ( D_A, A )
  call AssociateHost ( D_B, B )
  call AssociateHost ( D_C, C )

  !$OMP target teams distribute parallel do schedule ( static, 1 )
  do i = 1, size ( C )
    C ( i ) = A ( i ) + B ( i )
  end do
  !$OMP end target teams distribute parallel do

  call DisassociateHost ( C )
  call DisassociateHost ( B )
  call DisassociateHost ( A )

end subroutine AddKernel
```

Tells OpenMP data location on GPU
→avoid (implicit) allocation & transfer

```fortran
call F % Initialize &
iables])
eDevice ( )
call F % UpdateDevice ( )
call AddKernel &
  ( F % Value ( :, 1 ),
    F % Value ( :, 2 ), &
    F % D_Value ( 1 ),
    F % D_Value ( 2 ), &
    F % D_Value ( 3 ),
    F % Value ( :, 3 ) )
```

Offloading Computational Kernel

```
subroutine AddKernel ( A, B, D_A, D_B, D_C, C )

  real ( KDR ), dimension ( : ), intent ( in ) :: A, B
  type ( c_ptr ), intent ( in ) :: D_A, D_B, D_C
  real ( KDR ), dimension ( : ), intent ( out ) :: C

  in

  ca
  ca
  ca

  !$OMP target teams distribute parallel do schedule ( static, 1 )
  do i = 1, size ( C )
    C ( i ) = A ( i ) + B ( i )
  end do
  !$OMP end target teams distribute parallel do

  call DisassociateHost ( C )
  call DisassociateHost ( B )
  call DisassociateHost ( A )

end subroutine AddKernel
```

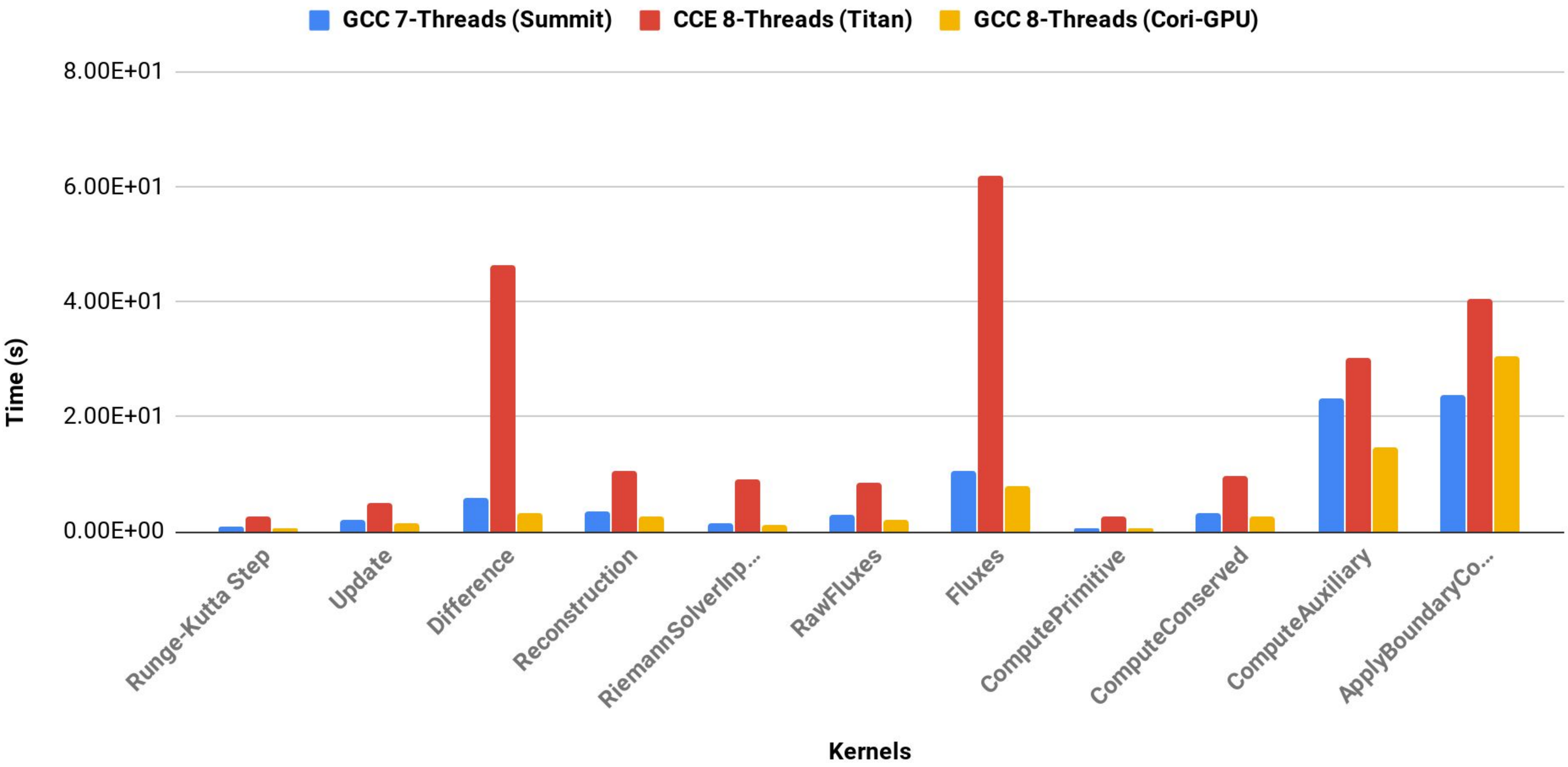
```
!$OMP  target teams distribute parallel do [collapse(n)] &
!$OMP& schedule (static, 1)
!$ACC loop gang vector [collapse(n)]
```

```
&
es])
ice ( )
call F % UpdateDevice ( )
call AddKernel &
  ( F % Value ( :, 1 ),
    F % Value ( :, 2 ), &
    F % D_Value ( 1 ),
    F % D_Value ( 2 ), &
    F % D_Value ( 3 ),
    F % Value ( :, 3 ) )
```

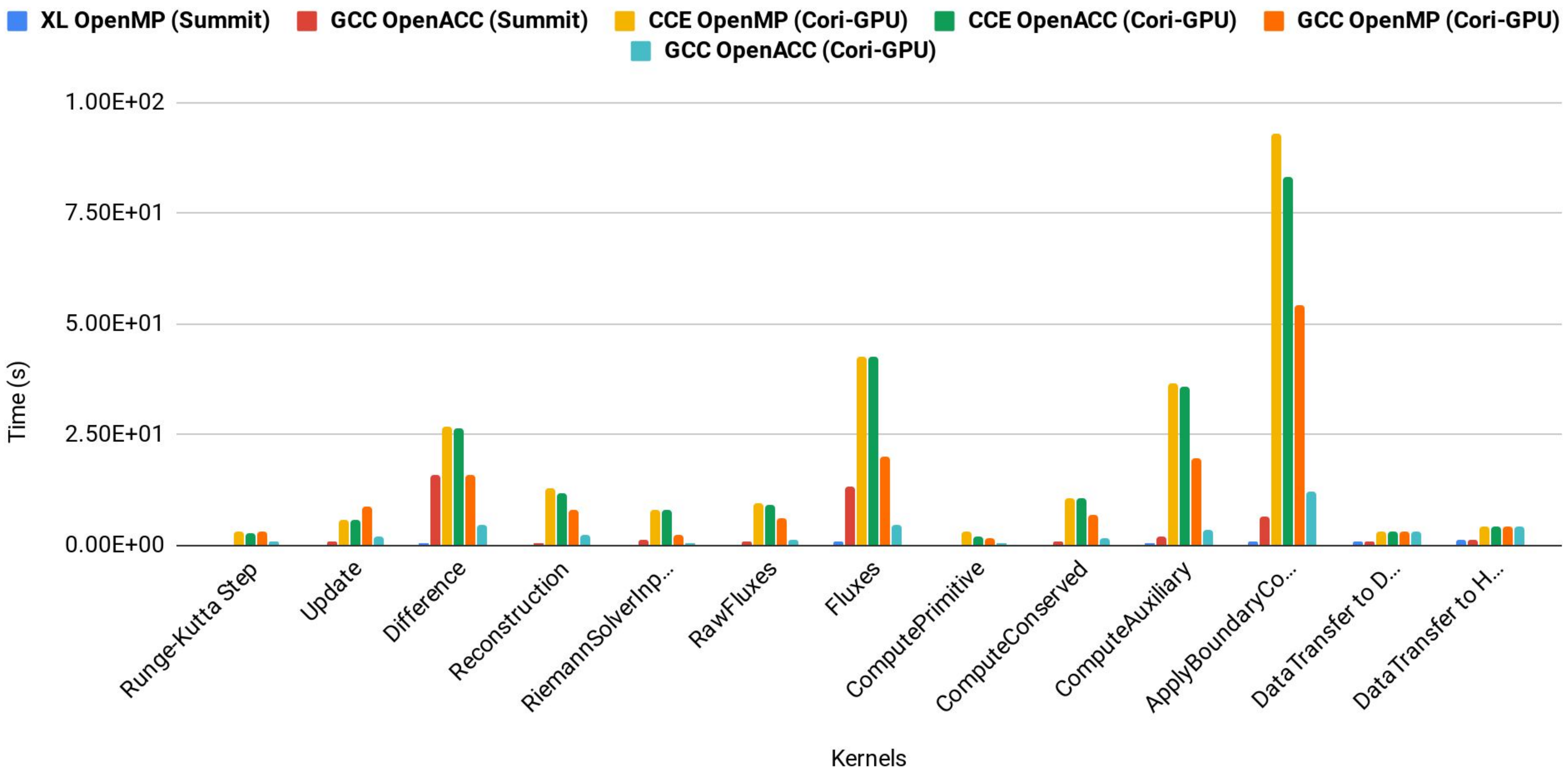
GenASiS: Test Setup

- Compilers:
  - IBM XL (OpenMP)
  - GCC-8 (OpenMP, OpenACC from MentorGraphics)
  - Cray CCE (OpenMP, OpenACC)
- Runtime:
  - 128^3 cells per MPI process, 1 MPI with OpenMP threads per GPU
  - Summit: 7 CPU threads vs. 1 GPU
  - Titan: 8 CPU threads vs 1 GPU
  - Cori-GPU: 8 CPU threads vs 1 GPU

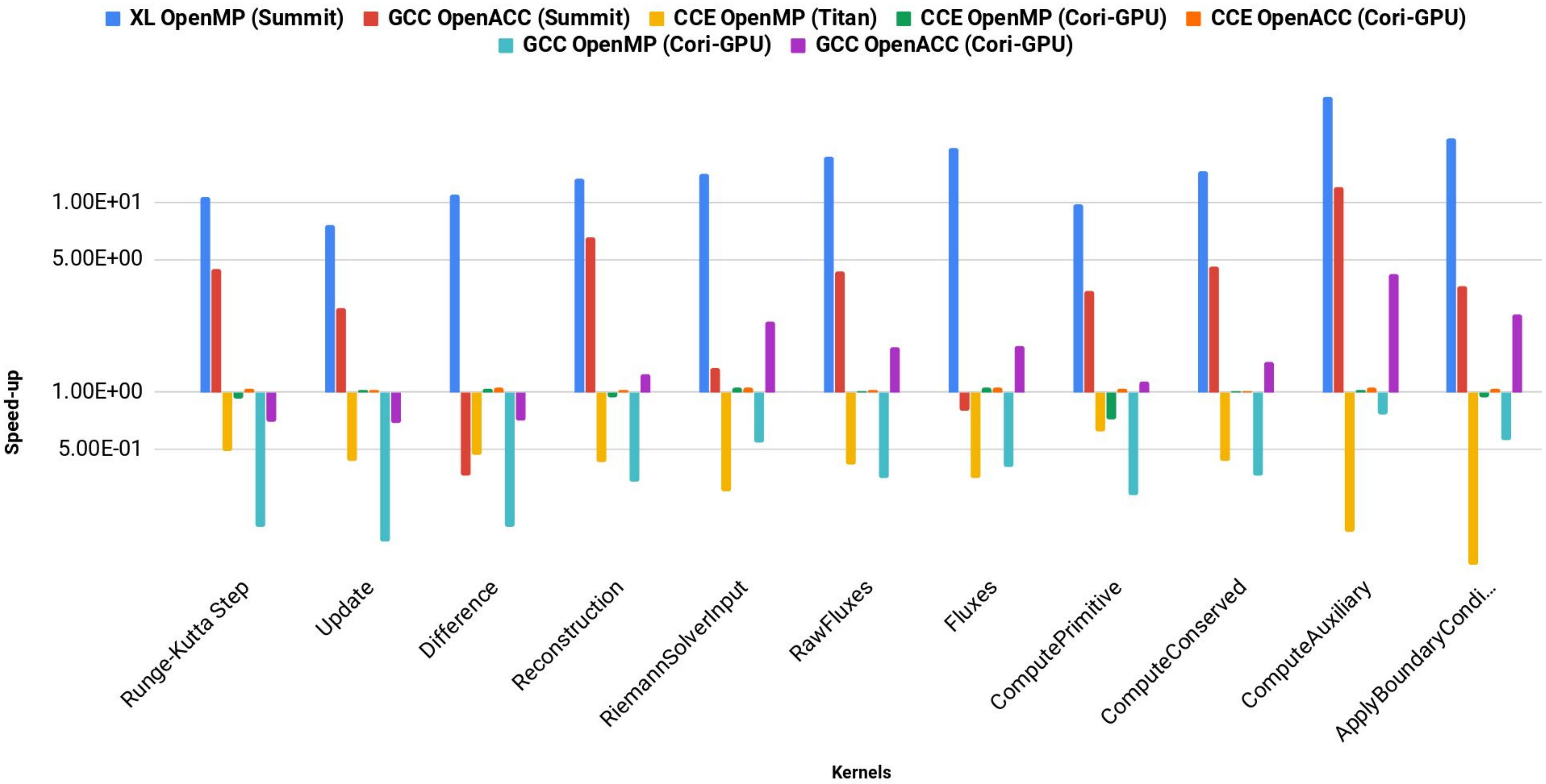
GenASiS: CPU Timings (Lower is Better)

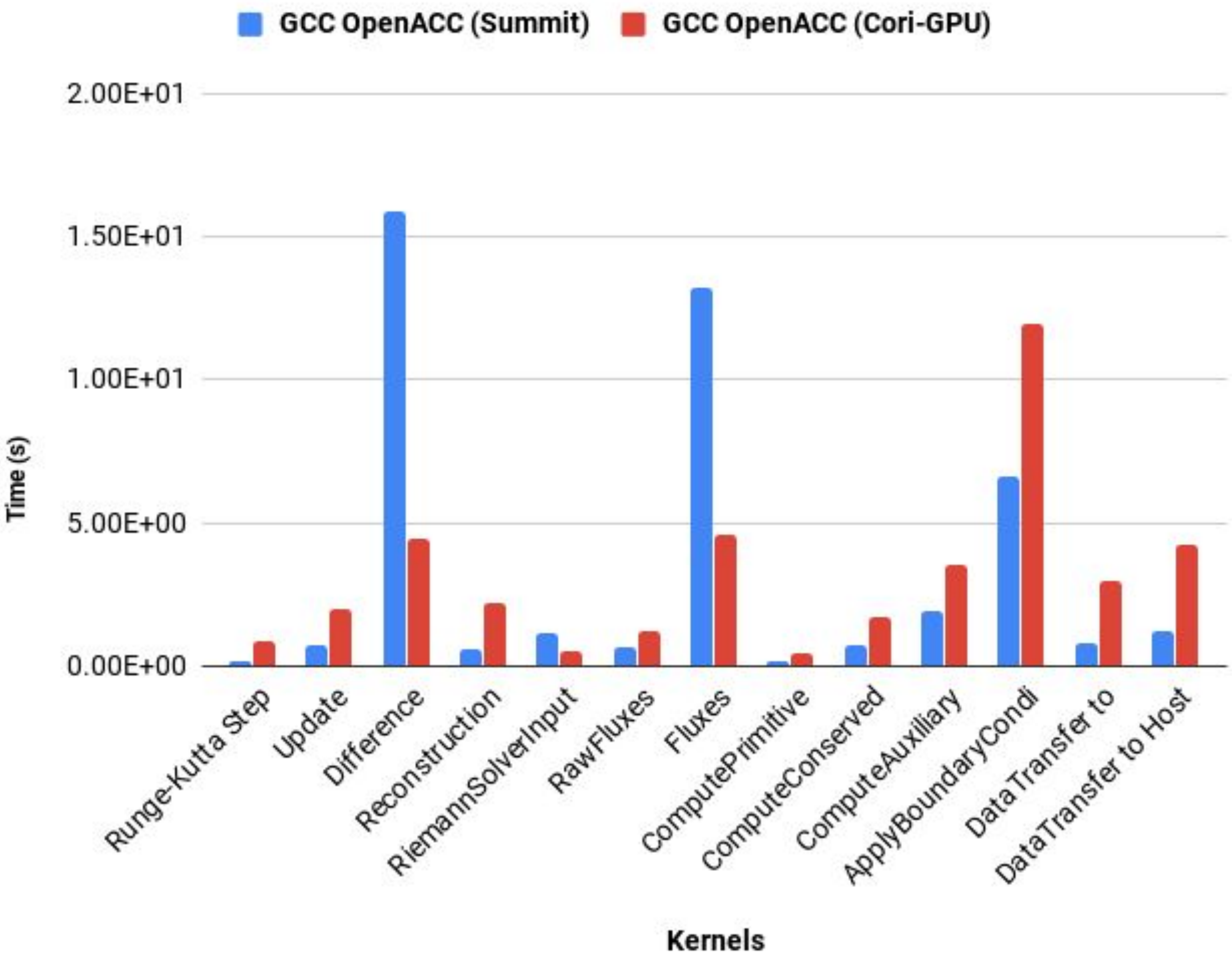


GenASiS: GPU Timings (Lower is Better)



GenASiS: Speed-up / Slow-down Relative to Multithreaded CPU

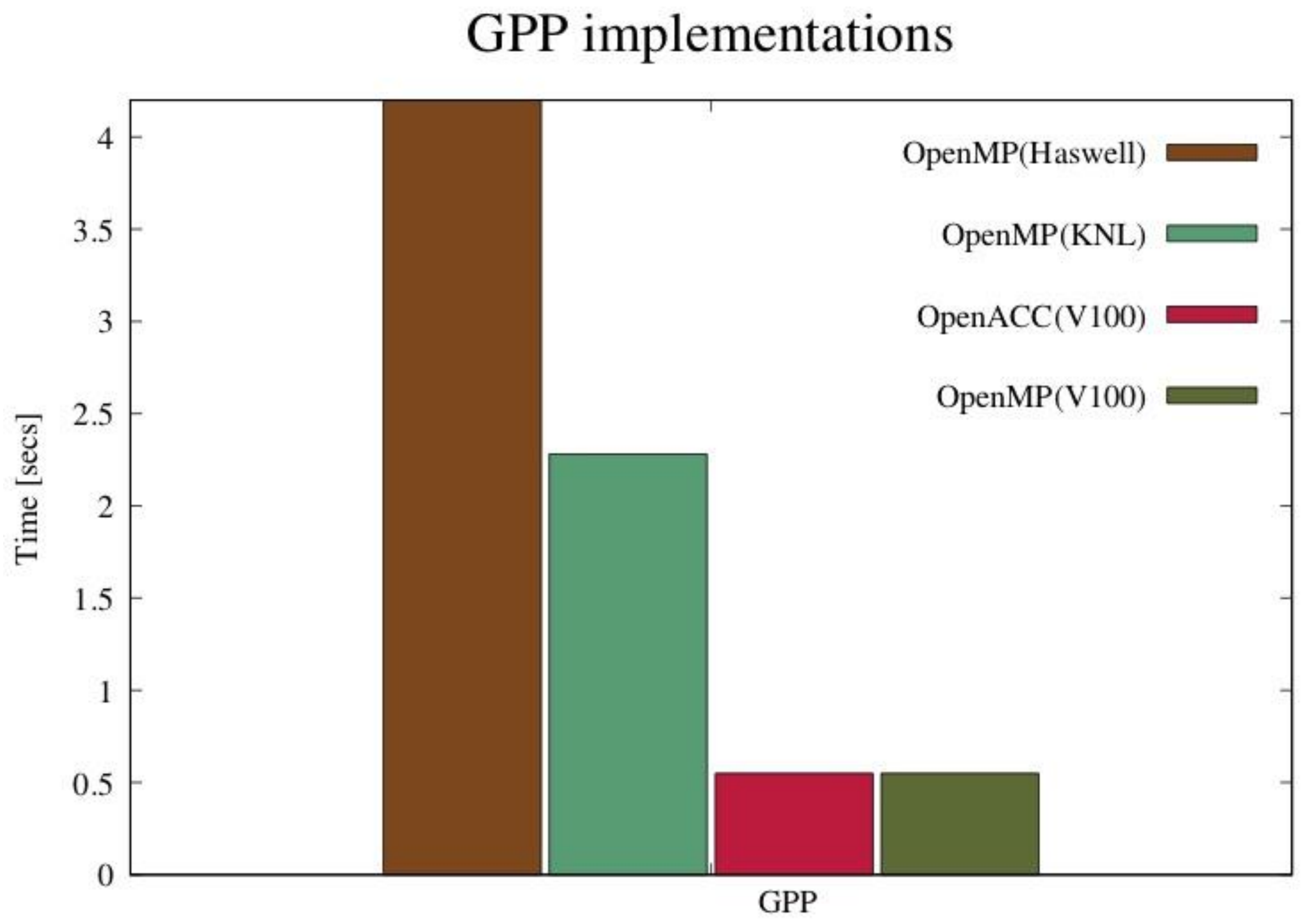



GCC OpenACC (Summit)
GCC OpenACC (Cori-GPU)
2.00E+01
1.50E+01
1.00E+01
5.00E+00
0.00E+00
Time (s)
Runge-Kutta Step
Update
Difference
Reconstruction
RiemannSolverInput
RawFluxes
Fluxes
ComputePrimitive
ComputeConserved
ComputeAuxiliary
ApplyBoundaryCondi
DataTransfer to
DataTransfer to Host
Kernels

GPP

- Mini-application from the BerkeleyGW suite written in C++.
- Computes electron self-energy using the General Plasmon Pole (GPP) approximation.
- Kernel is comprised of 4-nested loops with a reduction in the innermost loop.

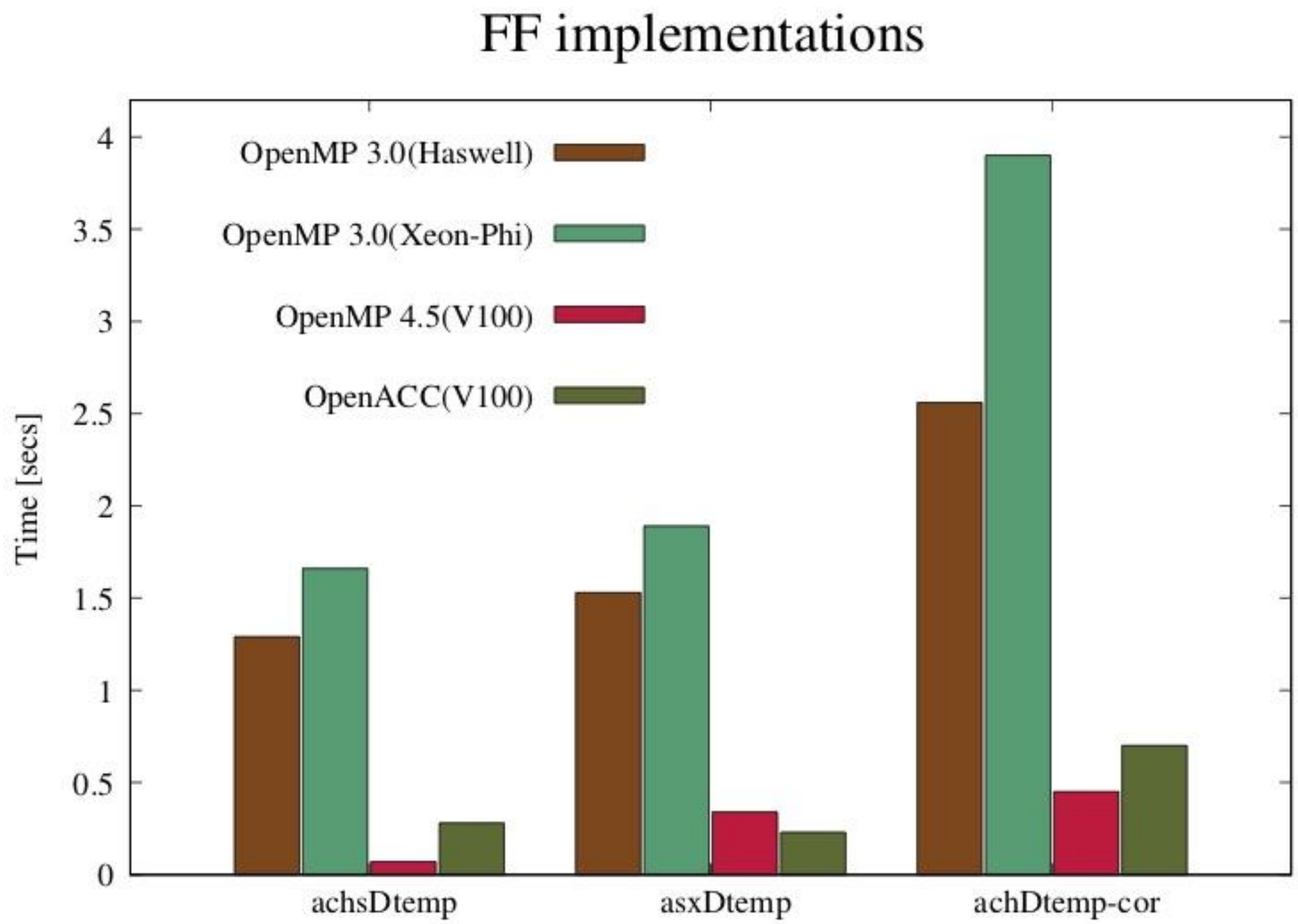
GPP Results


GPP implementations
Time [secs]
4
3.5
3
2.5
2
1.5
1
0.5
0
GPP
OpenMP(Haswell)
OpenMP(KNL)
OpenACC(V100)
OpenMP(V100)

FF

- Also part of the BerkeleyGW suite and written in C++.
- Represents the Full-Frequency (FF) Self-Energy Summations.
- Comprised of 3 kernels with different loop structures.

FF results



FF implementations
OpenMP 3.0(Haswell)
OpenMP 3.0(Xeon-Phi)
OpenMP 4.5(V100)
OpenACC(V100)
Time [secs]
0
0.5
1
1.5
2
2.5
3
3.5
4
achsDtemp
asxDtemp
achDtemp-cor

Summary and Conclusion

- We shared experiences to port (mini)-applications to OpenMP / OpenACC
- Performance portability is not observed
  - Different level of maturity for compilers
  - Some workarounds required
- Code available for further experimentation
  - GenASiS: https://github.com/GenASiS/GenASiS_Basics/releases
  - Minisweep: https://github.com/olcf/minisweep
  - GPP/FF: https://gitlab.com/rgayatri/berkeleygw-kernels

Future work

- Perform in-depth analysis for cases that exhibit performance degradation.
- Identify root cause for remaining issues encountered in each code.
- Report bugs to the different compilers.
- Perform multi-node/large scale experiments.

Acknowledgements

- This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.
- This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231.