WACCPD@SC-19

WACCPD 2019

Sixth Workshop on Accelerator Programming Using Directives

Argonne NATIONAL LABORATORY

PERFORMANCE OF THE RI-MP2 FORTRAN KERNEL OF GAMESS ON GPUS VIA DIRECTIVE-BASED OFFLOADING WITH MATH LIBRARIES

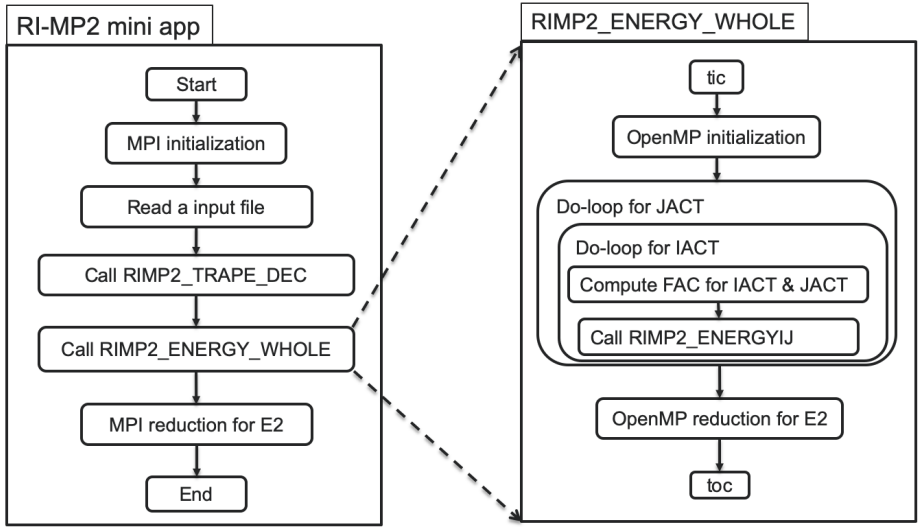
JAEHYUK KWACK[(1)], COLLEEN BERTONI[(1)], BUU PHAM[(2)], AND JEFF LARKIN[(3)]

Argonne National Laboratory [(1)], Iowa State University[(2)], NVIDIA[(3)]

November 18, 2019, Denver, CO

GAMESS AND RI-MP2 KERNEL

- GAMESS is a quantum chemistry software package designed for molecular simulations.
- Written in Fortran and C++, and Fortran portion has limited possible options for GPU computing.
- FMO/RI-MP2 is one of the quantum chemistry algorithms of interest
  – Algorithm of interest in GAMESS ECP problem



RI-MP2 equations:

$$E^{(2)} = \sum_{i \le j}^{occ} (2 - \delta_{ij}) \sum_{ab}^{vir} \frac{(ia|jb)[2(ja|jb) - (ib|ja)]}{\epsilon_i + \epsilon_j - (\epsilon_a + \epsilon_b)}$$

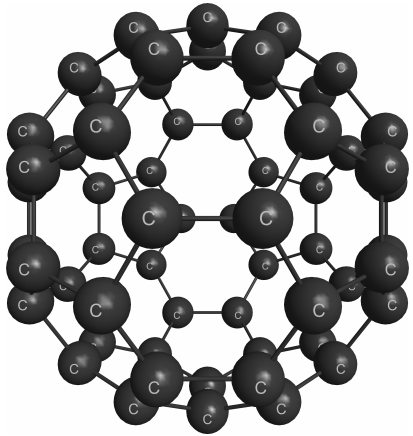
$$(ia|jb) = \sum_{n}^{aux} B_{an}^{i} B_{bn}^{j}$$

Argonne NATIONAL LABORATORY

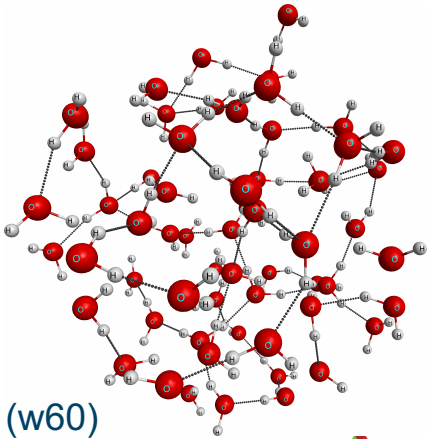
INPUTS FOR RI-MP2 KERNEL FROM GAMESS

- The inputs includes
  - the number of atomic orbital (N) and auxiliary (X) basis functions,
  - the number of correlated occupied (O) and virtual (V) molecular orbitals,
  - the molecular orbital coefficients,
  - the molecular orbital energies,
  - 3-index integral matrix B(X,V,O),
  - the calculated MP2 correlation energy for validation

	N	X	V	O	Total size (GB)
c60	540	3960	360	120	1.37
w30	720	2520	570	120	1.38
w60	1440	5040	1140	240	11.03



Fullerene (c60)

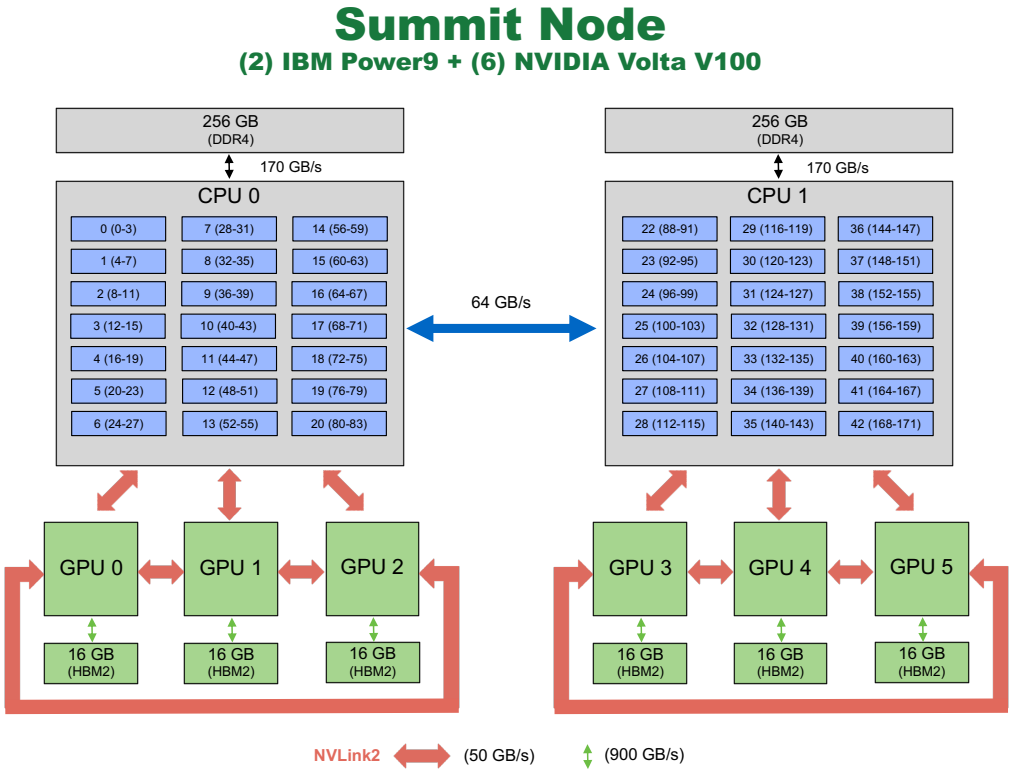


Water cluster (w60)

COMPUTE RESOURCES

- OLCF Summit
  - IBM Power 9 processor
    - ~540 GF/s/socket
    - ~1.08 TF/s/node
  - NVIDIA V100 GPU
    - ~7.8 TF/s/socket
    - ~46.8 TF/s/node

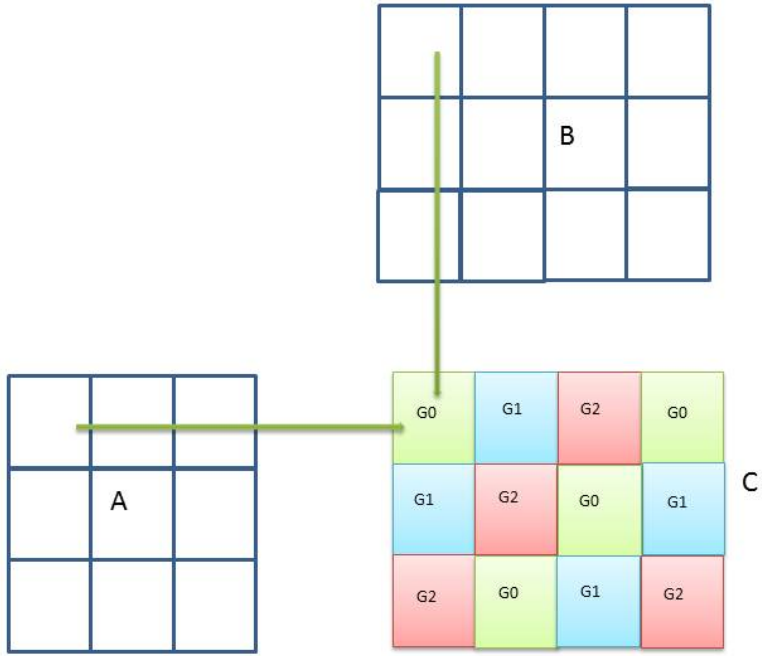
- ALCF JLSE
  - Intel Xeon Platinum 8180M Skylake processor
    - ~ 2.05 TF/s/socket
    - ~ 4.1 TF/s/node



Credit: OLCF

PROGRAMMING ENVIRONMENTS

- Compilers
  - IBM XLF version 16.1.1-3 (OpenMP 4.5)
  - Intel Fortran version 19.0.4.243
  - PGI Fortran 19.4 (OpenACC 2.6)
- Math libraries
  - IBM ESSL version 6.2.0
  - Intel MKL version 19.0.4.243
  - CUDA version 10.1.168
    - NVBLAS
    - CUBLAS
    - CUBLASXT



cublasXTdgemm () tiling (credit: NVIDIA)

OFFLOADING THE RI-MP2 KERNEL

RI-MP2 kernels with OpenMP threading

```
subroutine RIMP2_ENERGY_WHOLE ( ... )
...
   !$omp threadprivate(E2_omp)
   call OMP_SET_DYNAMIC(.FALSE.)
   nthreads=omp_get_max_threads()
...
   !$omp parallel NUM_THREADS(nthreads) default(none) shared(...) private(...)
   !$omp do schedule(DYNAMIC)
   do-loop for JACT ! From 1 to NACT
      do-loop for IACT ! From 1 to JACT
         Set FAC
         call RIMP2_ENERGYIJ (B32(:,:,IACT), B32(:,:,JACT), FAC, E2, ...)
      enddo
   enddo
   !$omp end do
   !$omp atomic
   E2 = E2 + E2_omp
   !$omp end parallel
end !subroutine RIMP2_ENERGY_WHOLE ( ... )

subroutine RIMP2_ENERGYIJ( ... )
...
   call DGEMM for BI(:,:), BJ(:,:), QVV(:,:)
   do-loop for IB ! From 1 to NVIR
      do-loop for IA ! From 1 to NVIR
         compute E2_t with QVV(:,:), eij(:,:), eab(:,:)
      enddo
   enddo
   E2 = E2 + FAC*E2_t
end !subroutine RIMP2_ENERGYIJ( ... )
```

RI-MP2 kernels with OpenMP offloading

```
subroutine RIMP2_ENERGY_WHOLE ( ... )
...
   !$omp target enter data map(alloc: QVV) map(to: eij,eab,B32)
   do-loop for JACT ! From 1 to NACT
      do-loop for IACT ! From 1 to JACT
         Set FAC
         call RIMP2_ENERGYIJ (B32(:,:,IACT), B32(:,:,JACT), FAC, E2, ...)
      enddo
   enddo
   !$omp target exit data map(release: QVV,eij,eab,B32)
   E2 = E2 + E2_omp
end !subroutine RIMP2_ENERGY_WHOLE ( ... )

subroutine RIMP2_ENERGYIJ( ... )
...
   !$omp target data use_device_ptr(BI,BJ,QVV)
   call DGEMM for BI(:,:), BJ(:,:), QVV(:,:)
   !$omp end target data
   !$omp target map(tofrom:E2_t)
   !$omp teams distribute parallel do reduction(+:E2_t) collapse(2)
   do-loop for IB ! From 1 to NVIR
      do-loop for IA ! From 1 to NVIR
         compute E2_t with QVV(:,:), eij(:,:), eab(:,:)
      enddo
   enddo
   !$omp end teams distribute parallel do
   !$omp end target
   E2 = E2 + FAC*E2_t
end !subroutine RIMP2_ENERGYIJ( ... )
```

RI-MP2 kernels with OpenACC offloading

```
subroutine RIMP2_ENERGY_WHOLE ( ... )
...
   !$acc enter data create(QVV) copyin(eij,eab,b32)
   do-loop for JACT ! From 1 to NACT
      do-loop for IACT ! From 1 to JACT
         Set FAC
         call RIMP2_ENERGYIJ (B32(:,:,IACT), B32(:,:,JACT), FAC, E2, ...)
      enddo
   enddo
   !$acc wait
   !$acc exit data delete(QVV,eij,eab,B32)
   E2 = E2 + E2_omp
end !subroutine RIMP2_ENERGY_WHOLE ( ... )

subroutine RIMP2_ENERGYIJ( ... )
...
   !$acc host_data use_device(BI,BJ,QVV)
   call DGEMM for BI(:,:), BJ(:,:), QVV(:,:)
   !$acc end host_data
   !$acc parallel loop collapse(2) reduction(+:E2_t) default(present)
   do-loop for IB ! From 1 to NVIR
      do-loop for IA ! From 1 to NVIR
         compute E2_t with QVV(:,:), eij(:,:), eab(:,:)
      enddo
   enddo
   E2 = E2 + FAC*E2_t
end !subroutine RIMP2_ENERGYIJ( ... )
```

USING NVBLAS, CUBLAS AND CUBLASXT

Initialization for cuBLAS and cuBLASXT

cuBLAS	cublas_return = cublascreate_v2(cublas_handle)
cuBLASXT	cublas_return = cublasXtcreate(cublas_handle) cublasXt_deviceId(1) = 0 cublas_return = cublasXtDeviceSelect(cublas_handle, 1, cublasXt_deviceId) cublas_return = cublasXtSetBlockDim(cublas_handle, 2048)

DGEMM calls for NVBLAS, cuBLAS, & cuBLASXT

NVBLAS	call DGEMM('T', 'N',& NVIR, NVIR, NAUXBASD, 1.0D00, & BI, NAUXBASD, BJ, NAUXBASD, 0.0D00, QVV, NVIR)
cuBLAS	cublas_return = CUBLASDGEMM_v2(cublas_handle, & CUBLAS_OP_T, CUBLAS_OP_N, & NVIR, NVIR, NAUXBASD, 1.0D00, & BI, NAUXBASD, BJ, NAUXBASD, 0.0D00, QVV, NVIR) cublas_return = cudaDeviceSynchronize( )
cuBLASXT	cublas_return = cublasXtDgemm(cublas_handle, & CUBLAS_OP_T, CUBLAS_OP_N, & NVIR, NVIR, NAUXBASD, 1.0D00, & BI, NAUXBASD, BJ, NAUXBASD, 0.0D00, QVV, NVIR) cublas_return = cudaDeviceSynchronize( )

Finalization for cuBLAS and cuBLASXT

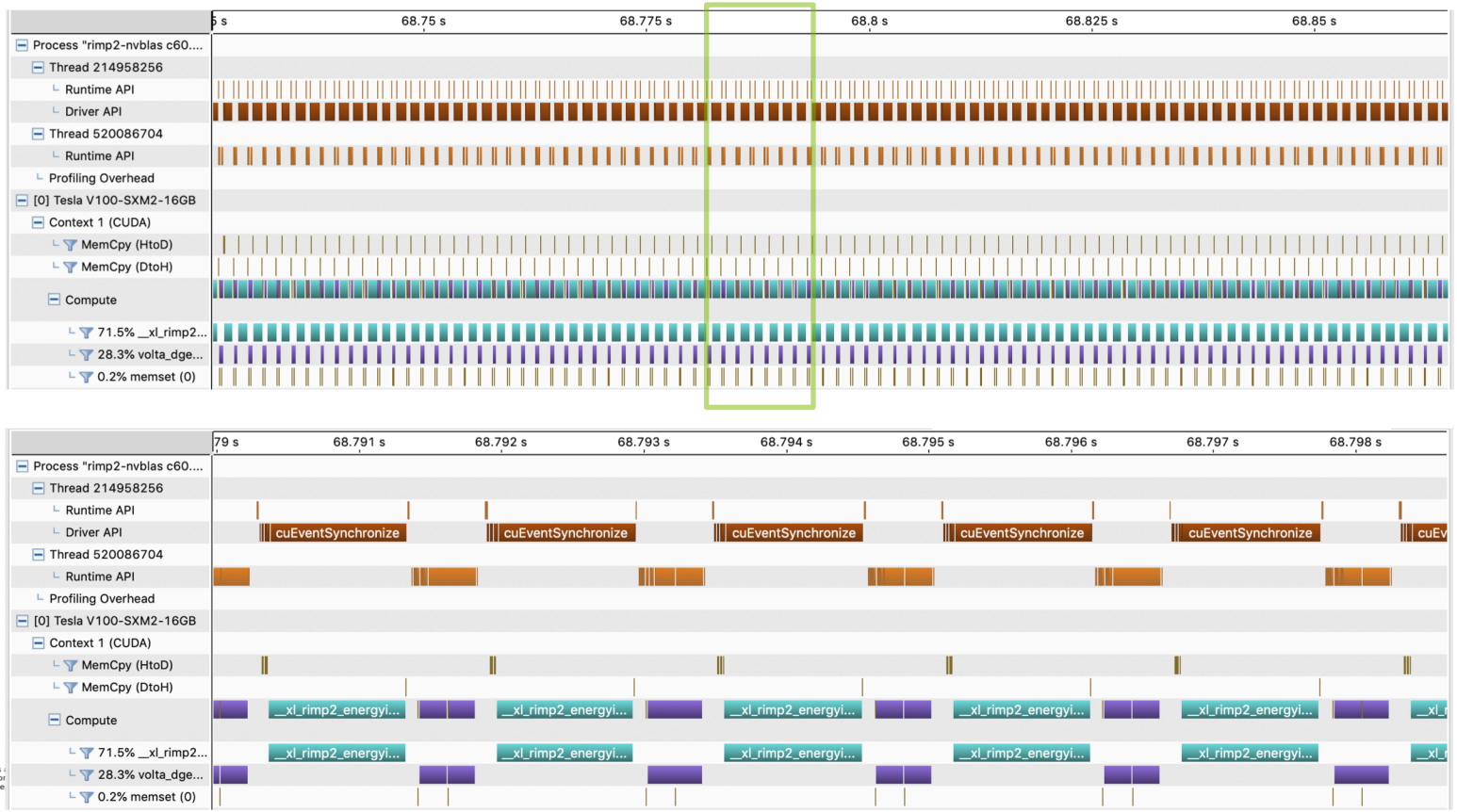
cuBLAS	cublas_return = cublasdestroy_v2(cublas_handle)
cuBLASXT	cublas_return = cublasXtdestroy(cublas_handle)

PERFORMANCE OF RI-MP2 KERNEL WITH OPENMP/OPENACC OFFLOADING

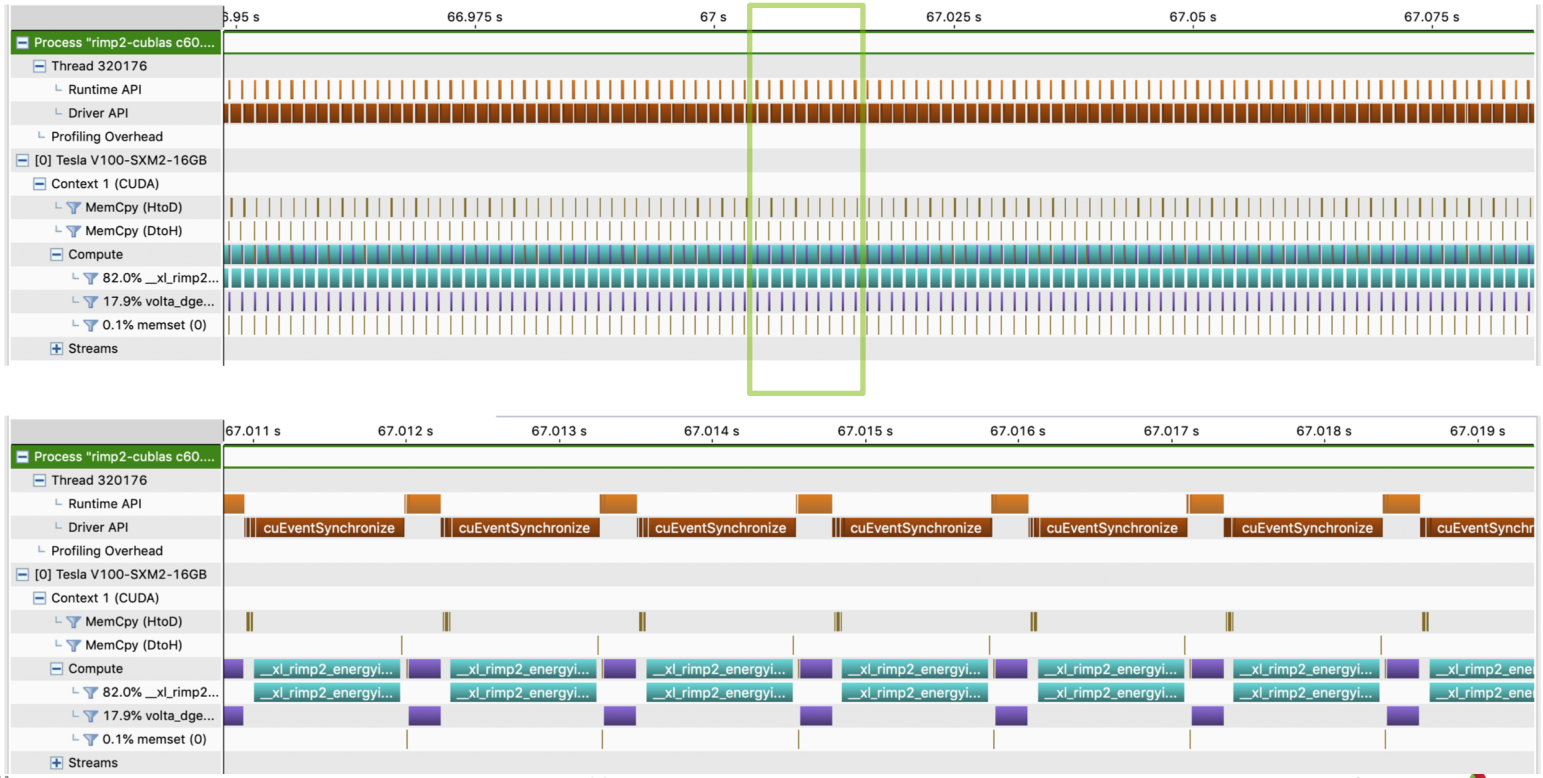
Input: c60.kern

	Wall time (s)	Speedup
Serial w/ 1 core of P9	344.763	0.037 x
OpenMP + ESSL dgemm w/ 42 threads on 2 P9	12.623	1 x
OpenMP + MKL dgemm w/ 112 threads on 2 SKX	4.802	**2.63 x**
OpenMP offloading + nvblas dgemm on 1 V100	11.320	1.12 x
OpenMP offloading + cublas dgemm on 1 V100	9.282	1.36 x
OpenMP offloading + cublasXt dgemm on 1 V100	11.372	1.11 x
OpenACC offloading + cublas dgemm on 1 V100	12.176	1.04 x
OpenACC offloading + cublasXt dgemm on 1 V100	14.548	0.87 x

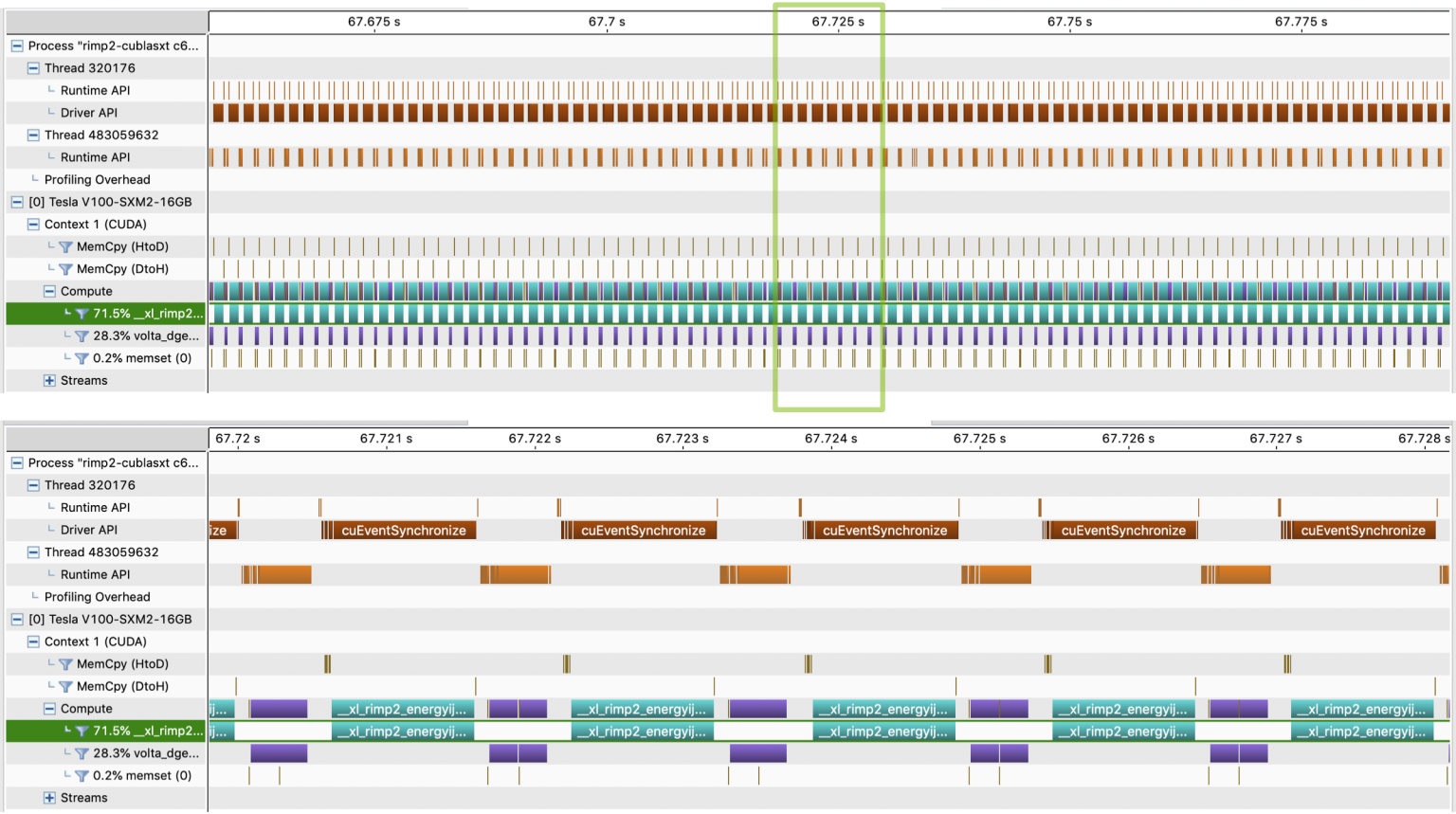
OFFLOADING W/ NVBLAS



OFFLOADING W/ CUBLAS



OFFLOADING W/ CUBLASXT



OFFLOADING

THE RESTRUCTURED RI-MP2 KERNEL

Restructured RI-MP2 kernels for fewer DGEMM calls with larger matrices

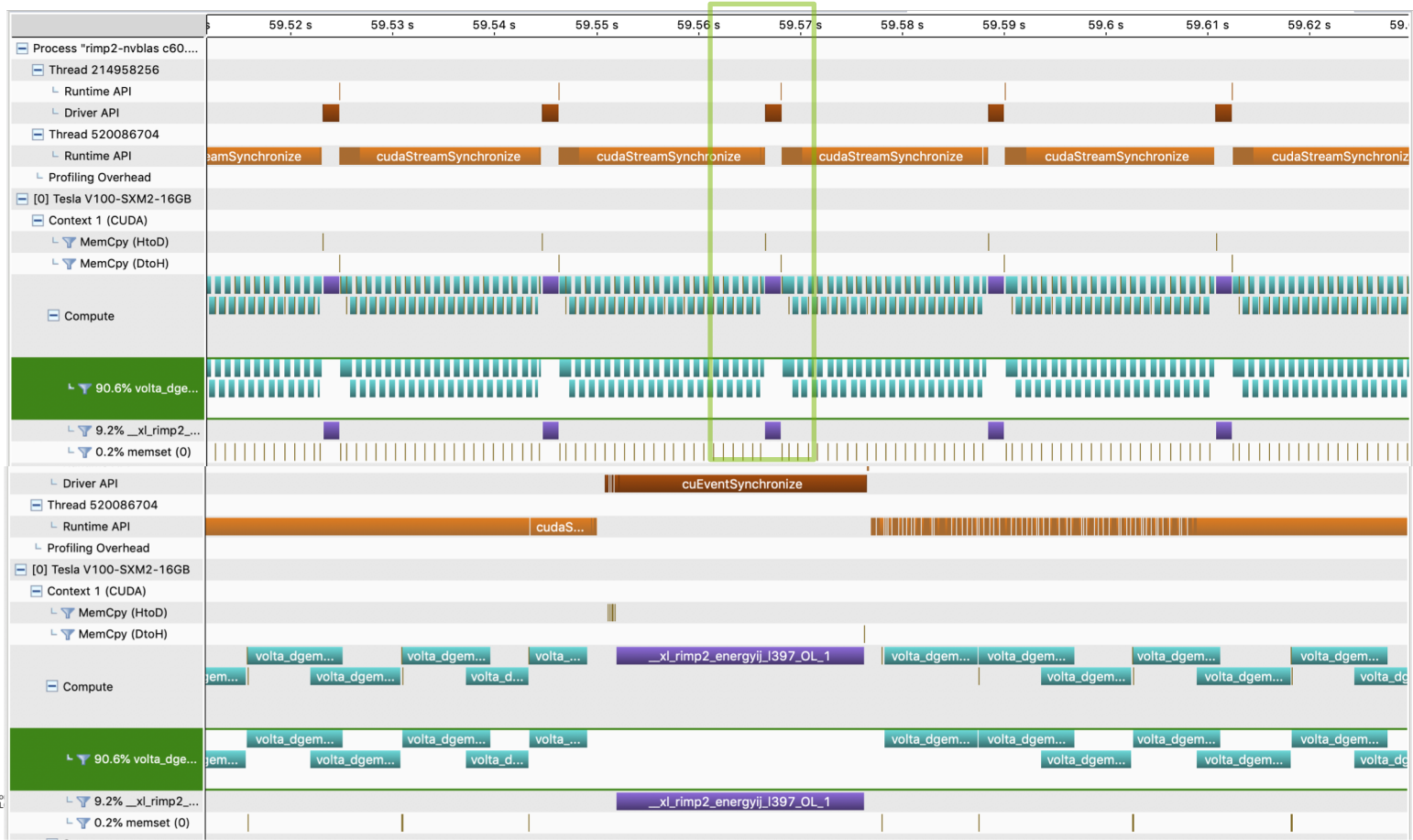
```
subroutine RIMP2_ENERGY_WHOLE ( ... )
...
   do-loop for JACT ! From 1 to NACT
      call RIMP2_ENERGYIJ (B32(:,:,1:JACT), B32(:,:,JACT), E2, ...)
   enddo
...
end !subroutine RIMP2_ENERGY_WHOLE ( ... )

subroutine RIMP2_ENERGYIJ( ... )
...
   call DGEMM for BI(:,:,1:JACT), BJ(:,:), QVV(:,:,1:JACT)
...
   do-loop for IC                    ! From 1 to JACT
      Set FAC
      do-loop for IB                 ! From 1 to NVIR
         do-loop for IA              ! From 1 to NVIR
            compute E2_t with QVV(:,:,IC), eij(:,:), eab(:,:)
         enddo
      enddo
      E2 = E2 + FAC*E2_t
   enddo
...
end !subroutine RIMP2_ENERGYIJ( ... )
```

PERFORMANCE OF THE RESTRUCTURED KERNEL

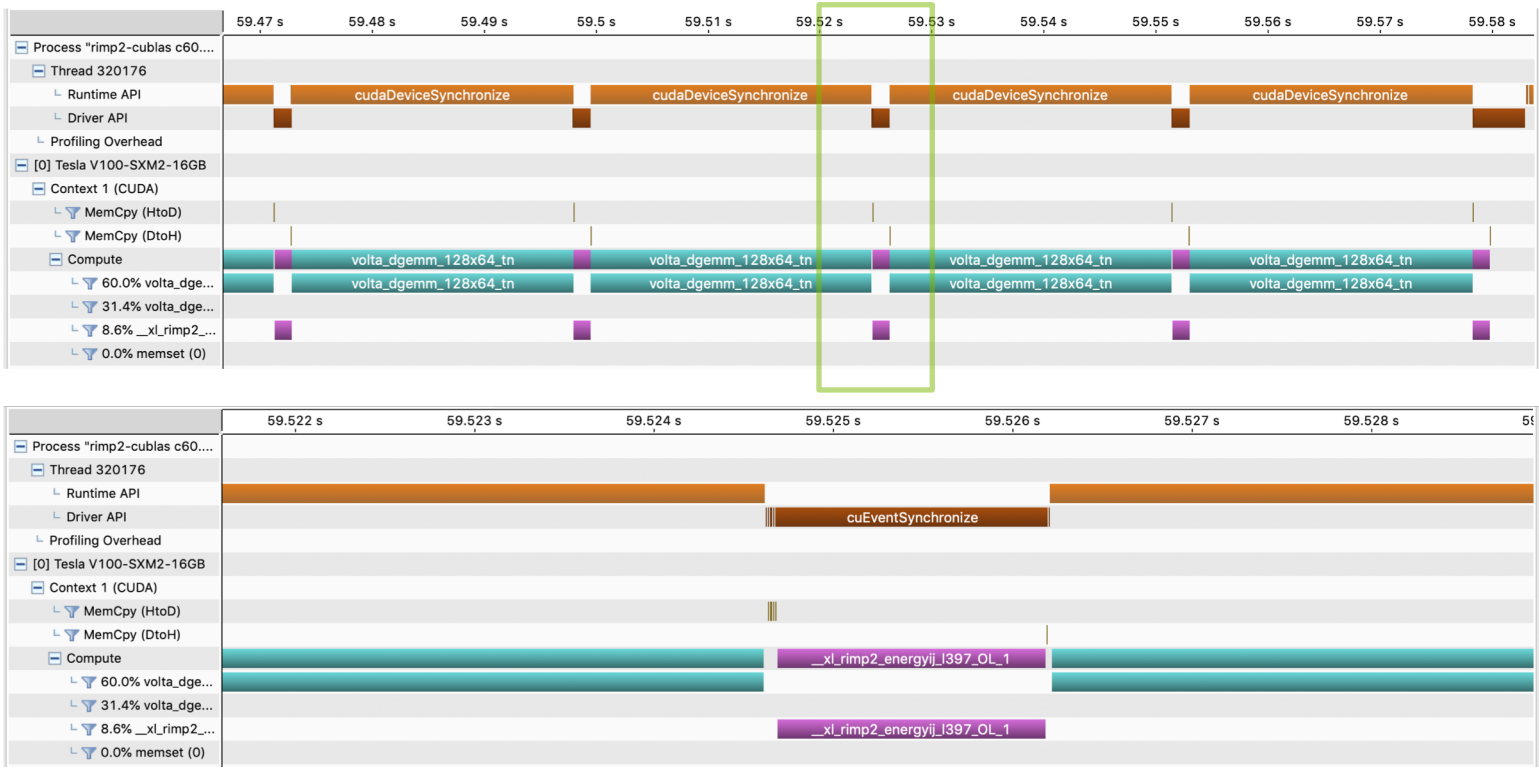
Input: c60.kern

	Wall time (s)	Speedup
Serial w/ 1 core of P9	342.697	0.036 x
OpenMP + ESSL dgemm w/ 42 threads on 2 P9	12.231	1 x
OpenMP + MKL dgemm w/ 112 threads on 2 SKX	4.317	2.83 x
OpenMP offloading + nvblas dgemm on 1 V100	1.734	7.05 x
OpenMP offloading + cublas dgemm on 1 V100	1.983	6.17 x
OpenMP offloading + cublasXt dgemm on 1 V100	1.728	**7.08 x**
OpenACC offloading + cublas dgemm on 1 V100	1.905	6.42 x
OpenACC offloading + cublasXt dgemm on 1 V100	1.692	**7.23 x**

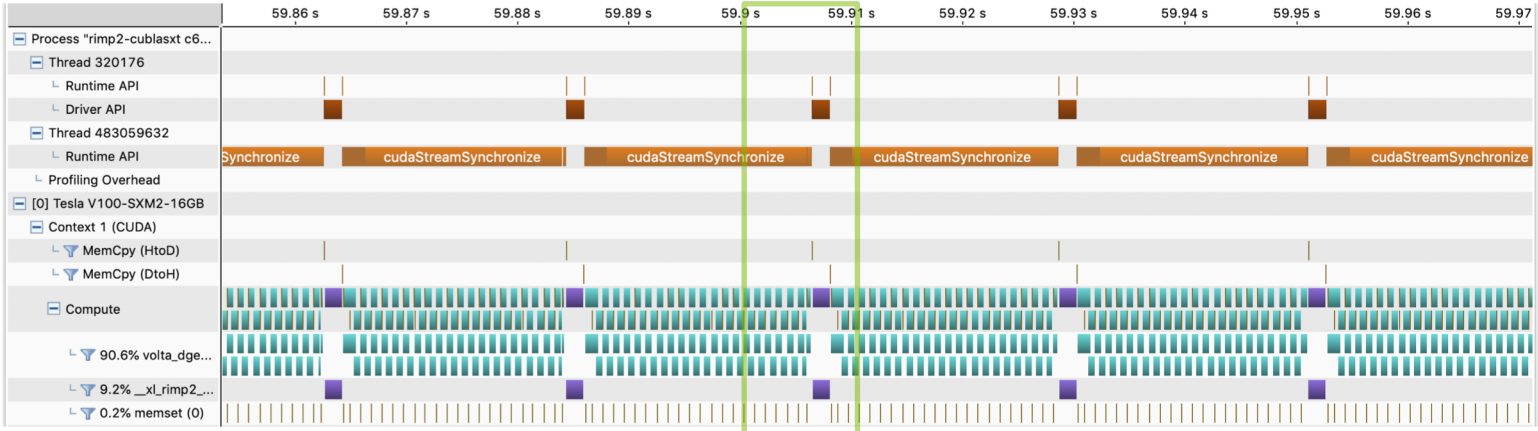
RESTRUCTURED KERNEL W/ NVBLAS

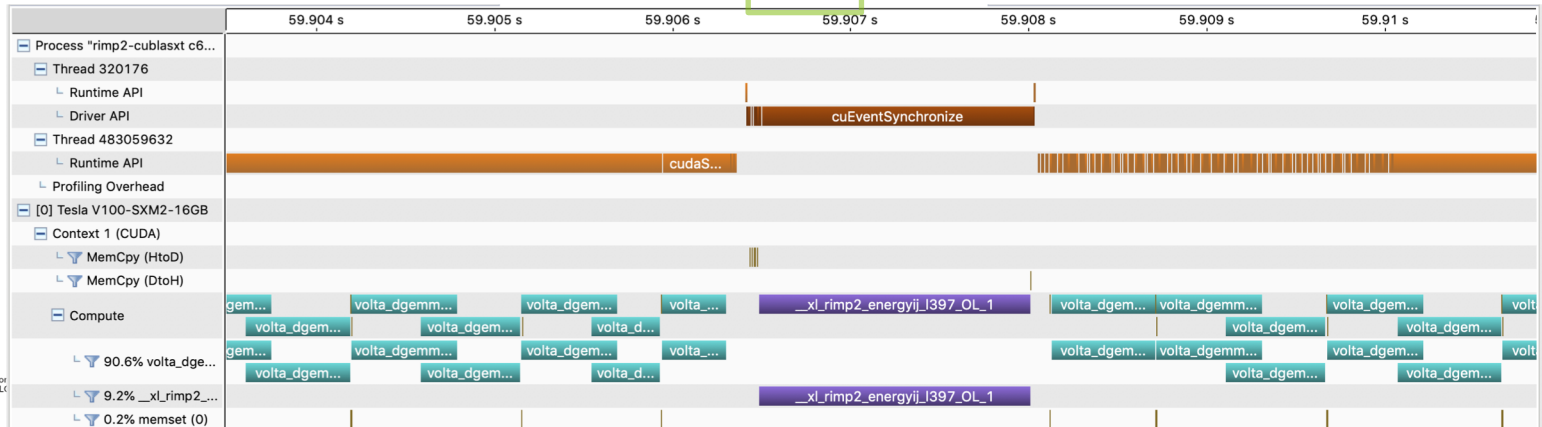


RESTRUCTURED KERNEL W/ CUBLAS



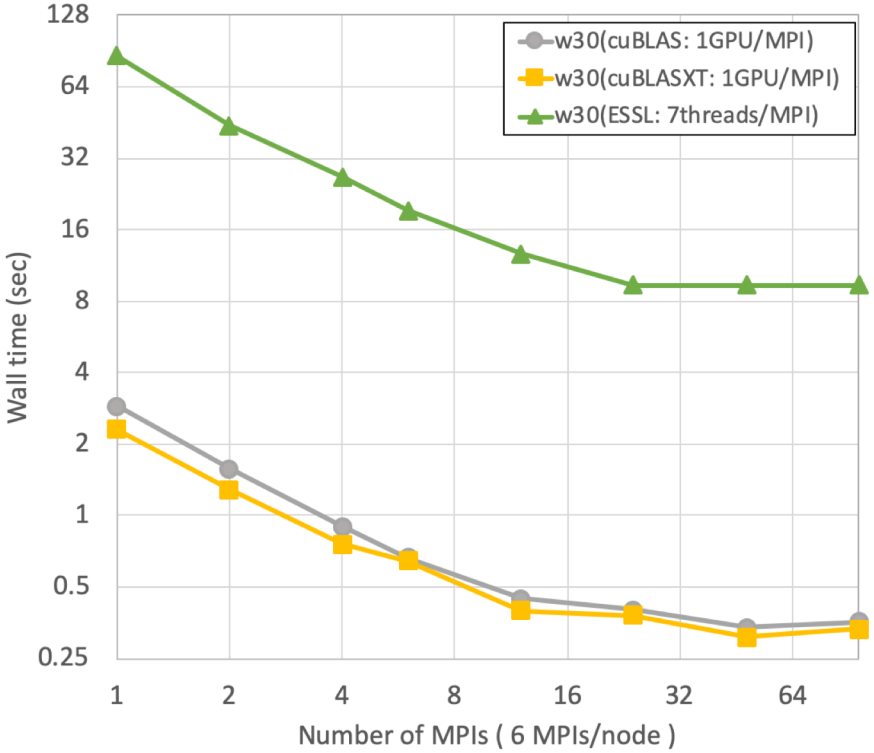
RESTRUCTURED KERNEL W/ CUBLASXT

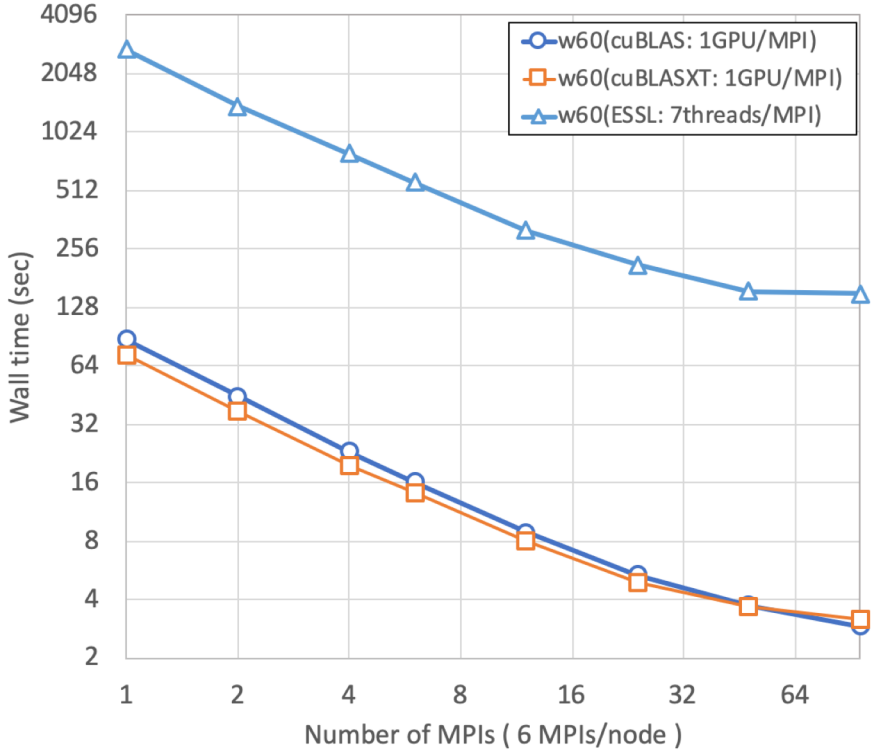




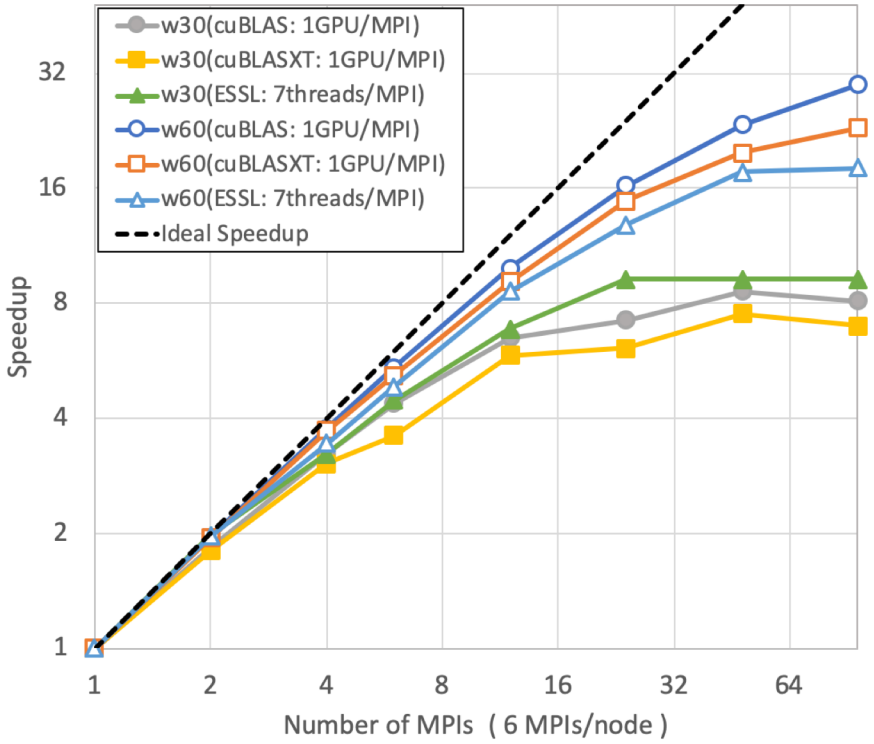
RESTRUCTURED KERNEL ON MULTIPLE GPUS

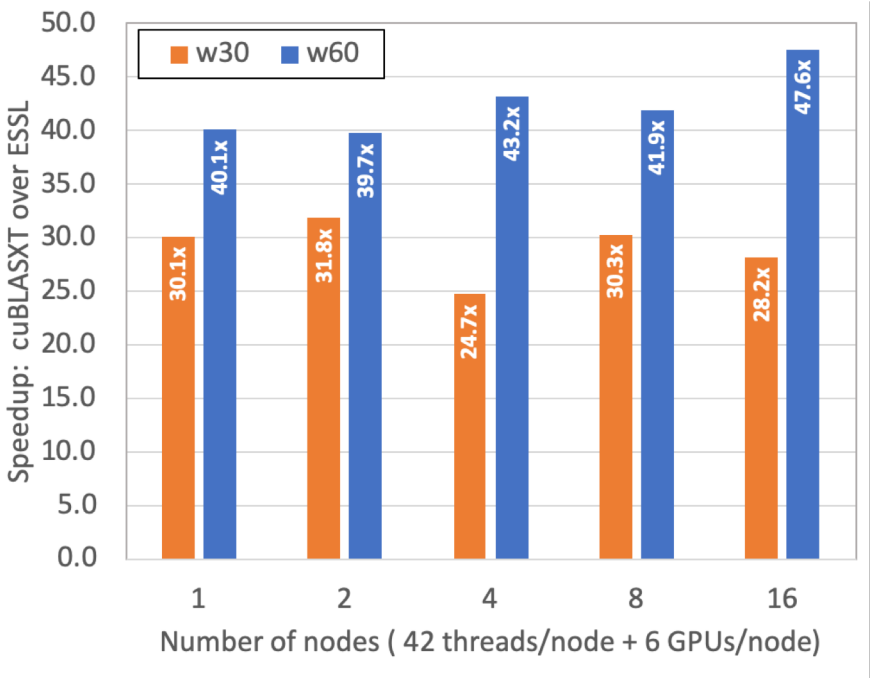
RESTRUCTURED KERNEL ON MULTIPLE GPUS





RESTRUCTURED KERNEL ON MULTIPLE GPUS





CONCLUDING REMARKS

CONCLUDING REMARKS

- RI-MP2 kernel from GAMESS application is re-written via OpenMP and OpenACC offloading implementations with three GPU libraries (i.e., NVBLAS, cuBLAS, and cuBLASXT).

- **Restructuring the original kernels was required to get good performance on GPUs.**

- On a single NVIDIA V100 GPU, the directive-based offloading kernels show
  - more than **7x speedup over 42 threaded code on IBM P9 processors**,
  - around 200 x speedup over the serial run on IBM P9 processors.

- On the same number of Summit nodes, the MPI+OpenMP offloading kernels show
  - More than **40x speedup over the MPI + OpenMP threading kernels.**

CONCLUDING REMARKS

- Observation
  - CUBLAS makes a Fortran code messy, while **NVBLAS provides standard BLAS calls** that are simpler than CUBLAS.
    - However, **NVBLAS and CPU math library** (e.g., ESSL, MKL, and ArmPL) use the **same symbol** (e.g., DGEMM), and it may result in **unexpected errors** or **lower performance on heterogeneous architecture**.

- Suggestion
  - NVBLAS may provide **alternative symbols** in addition to standard BLAS symbols, for users to avoid some conflicts with CPU math library.
  - OpenMP 5 **declare variant** directive may figure out this symbol conflict.

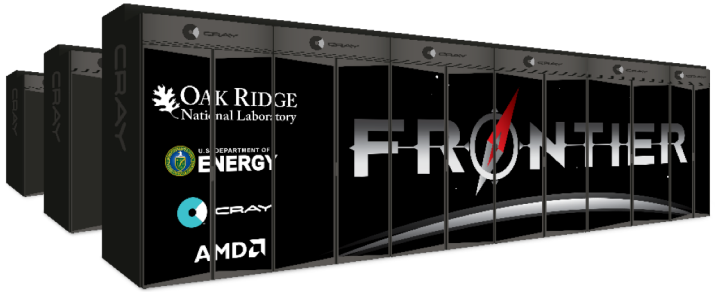
CONCLUDING REMARKS

- A directive-based programming model will be **a portable solution with good performance** on coming pre-exascale/exascale DoE systems.

Perlmutter @NERSC in 2020 (w/ NVIDIA GPUs)

Aurora @ALCF in 2021 (w/ Intel Xᵉ GPUs)

Frontier @OLCF in 2021 (w/ AMD GPUs)



ACKNOWLEDGEMENT

- This work was supported by
  - the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357,
  - and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. DOE Office of Science and the National Nuclear Security Administration,
  - and by a grant from the DOE Exascale Computing Project (ECP), administered by the Ames Laboratory.
- We also gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory, and the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.
- We would like to thank the ECP and OLCF for organizing the 2019 ECP/OLCF OpenMP Hackathon in Knoxville, TN, and give special thanks our mentors, Dmytro Bykov from OLCF and Vivek Kale from BNL for their contributions to this work.

THANK YOU!