Development of Intel MIC Codes in NWChem

Edoardo Aprà
Pacific Northwest National Laboratory

Acknowledgements

- **Karol Kowalski (PNNL)**
- **Michael Klemm (Intel)**
- Kiran Bhaskaran-Nair (LSU)
- Wenjing Ma (Chinese Acad. of Sciences)
- Oreste Villa (Nvidia)
- Sriram Krishnamoorthy, Jeff Daily, Bruce Palmer (PNNL)
- Vinod Tipparaju (AMD)
- Ryan Olson (Cray)
- Jiri Brabec, Jiri Pittner (Czech Ac. Sc.)
- Sotiris Xantheas (PNNL)

Acknowledgements ($$$ and Computer time)

- Dept. of Energy Office of Biological and Environmental Research
- PNNL eXtreme Scale Computing Initiative
- Computing resources
  - EMSL Molecular Science Computing facility
  - PNNL Institutional Computing
  - NERSC
  - National Center for Computational Sciences at ORNL – INCITE, ALCC & DD programs

EMSL is a national scientific user facility

William R. Wiley's Vision:
An innovative multipurpose user facility providing *"synergism between the physical, mathematical, and life sciences."*

Mission

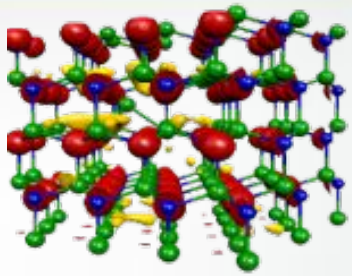
EMSL, a national scientific user facility at Pacific Northwest National Laboratory, provides ***integrated experimental and computational resources*** for ***discovery and technological innovation*** in the environmental molecular sciences to ***support the needs of DOE and the nation***.

- Funded by DOE **Office of Science's Office of Biological and Environmental Research**
- Located in Richland, WA - **Washington state's wine country**

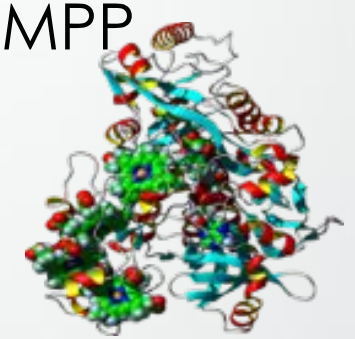
NWChem

NWCHEM

HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE



- Developed at EMSL/PNNL
- Provides major modeling and simulation capability for molecular science
  - Broad range of molecules, including catalysts, biomolecules, and heavy elements
  - Solid state capabilities
- Performance characteristics – designed for MPP
  - Runs on a wide range of computers
- Widely used in the main computing centers
- Open Source → active user community
- Uses Global Arrays/ARMCI for parallelization



www.nwchem-sw.org

NWChem Developers/Collaborators Community

NWCHEM

HIGH-PERFORMANCE COMPUTATIONAL CHEMISTRY SOFTWARE

J. Heyrovský Institute of Physical Chemistry
Academy of Sciences of the Czech Republic

BERKELEY LAB
Lawrence Berkeley National Laboratory

Argonne NATIONAL LABORATORY

NORTHWESTERN UNIVERSITY

Universitat de Girona

UBC THE UNIVERSITY OF BRITISH COLUMBIA

UCSD

PENNSTATE 1855

THE UNIVERSITY of WISCONSIN MADISON

UCLA

UCI University of California, Irvine

Massachusetts Institute of Technology

The Hebrew University of Jerusalem
האוניברסיטה העברית בירושלים

UB University at Buffalo
The State University of New York

UNIVERSITY OF CAMBRIDGE

UNT UNIVERSITY OF NORTH TEXAS
Discover the power of ideas.

IOWA STATE UNIVERSITY

OAK RIDGE NATIONAL LABORATORY
Managed by UT-Battelle for the Department of Energy

UNIVERSITY OF MINNESOTA
Driven to Discover℠

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

Duke UNIVERSITY

TEXAS TECH UNIVERSITY

BROOKHAVEN NATIONAL LABORATORY

Cal

vrije Universiteit amsterdam

UNIVERSITY of WASHINGTON

UCL

PRINCETON UNIVERSITY

UNIVERSITÄT DES SAARLANDES

Science & Technology Facilities Council
Daresbury Laboratory

LSU

Weizmann Institute of Science

FOUNDED 1901 DOMINICAN UNIVERSITY

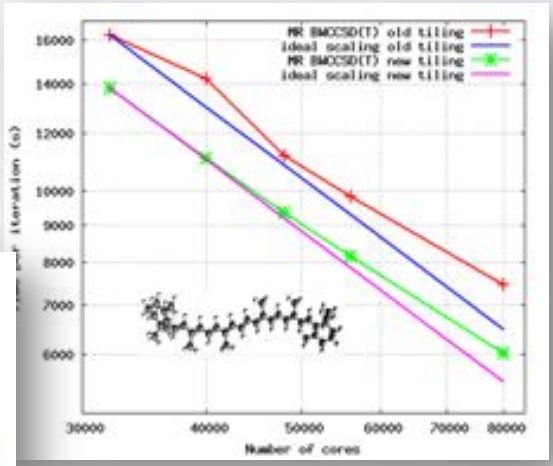
OHSU

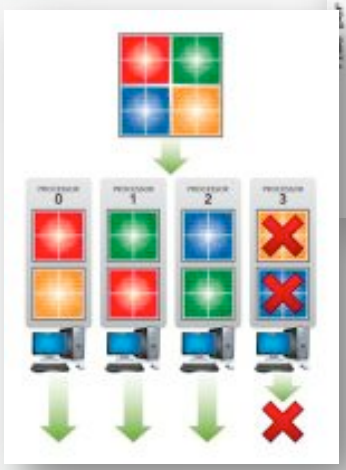
AKADEMIE VĚD ČESKÉ REPUBLIKY

Kitware

Scalability, New Platforms, Novel Algorithms

- Scalable algorithms for CC methods: three levels of parallelism (Kowalski)
- Fault tolerant versions of key NWChem implementations (van Dam)
- Parallel in time algorithms for AIMD & Distributed Computing Across Slow Networks (Bylaska)
- Heterogeneous computer architectures: GPU, Xeon Phi processors (Apra,Kowalski)

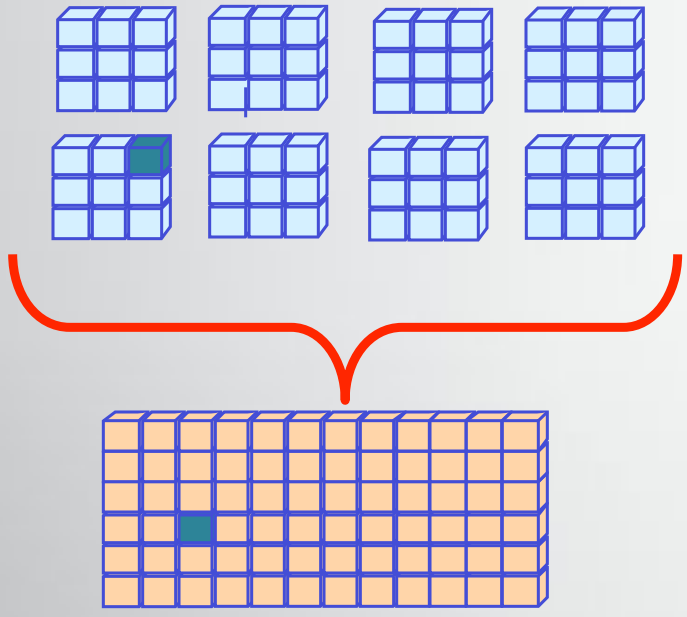




Global Arrays

- **Distributed dense arrays** that can be accessed through a shared memory-like style
- **High level abstraction** layer for the application developer (that's me!)
- **One-sided** model = no need to worry and send/receive

Physically distributed data

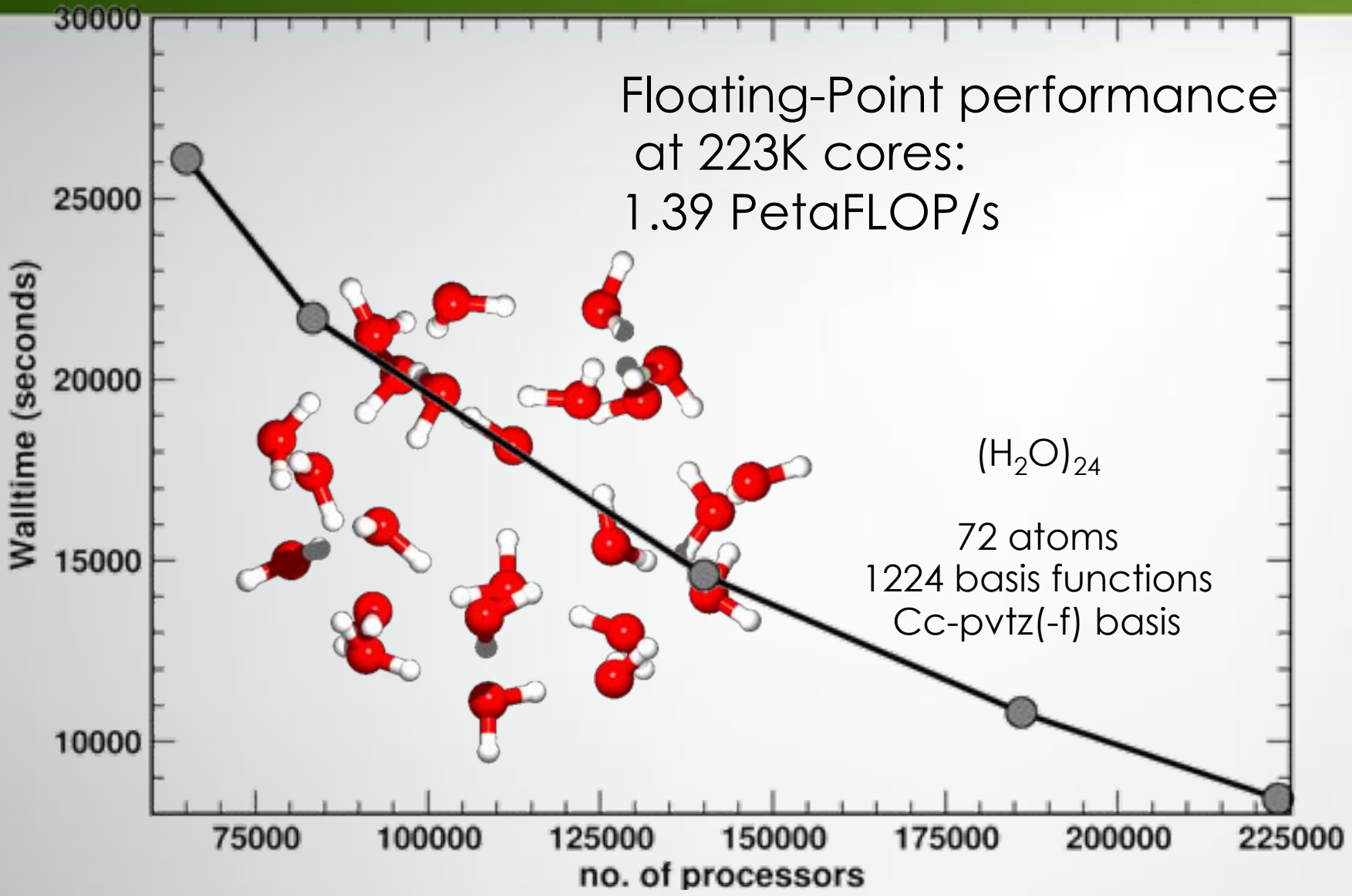


Global Address Space

single, shared data structure/ global indexing

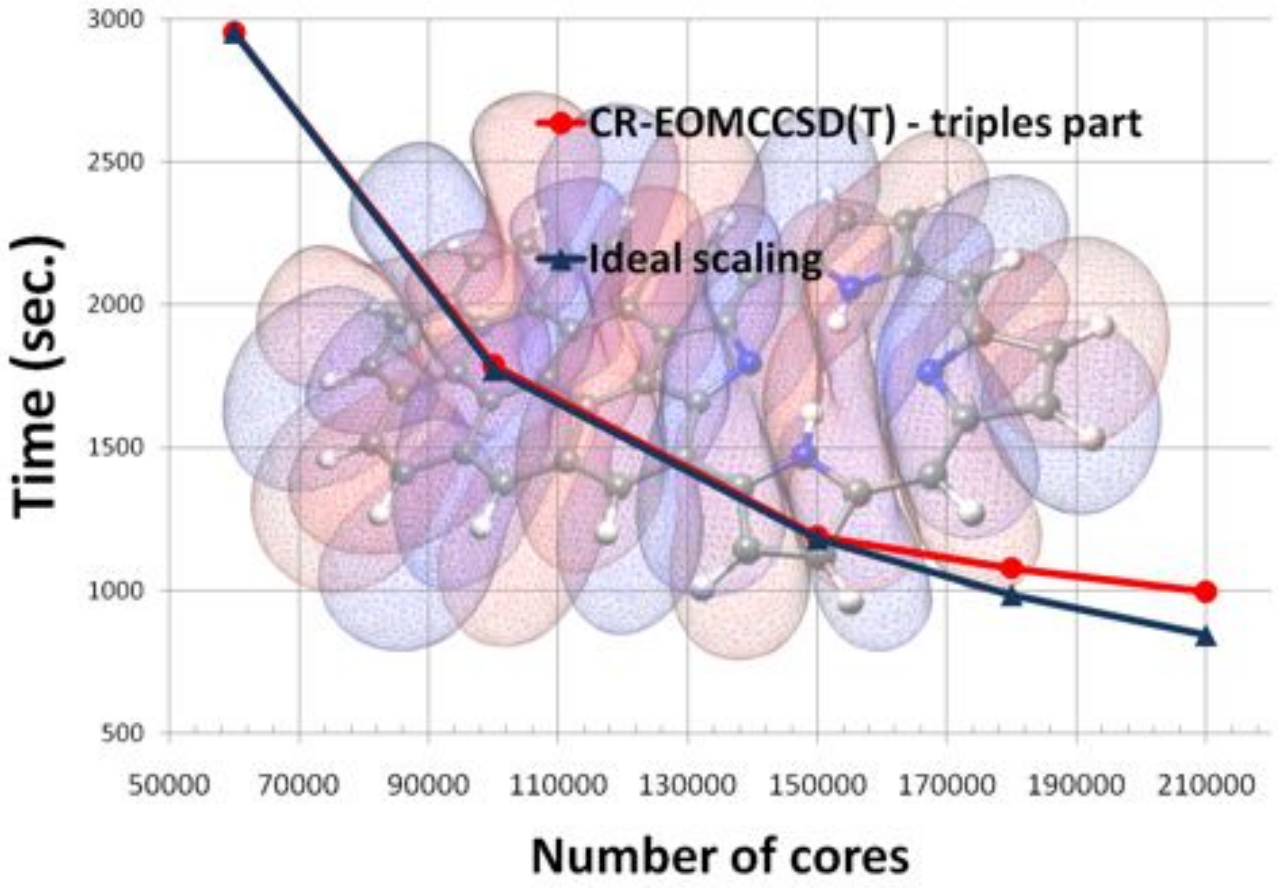
e.g., access A(4,3) rather than buf(7) on task 2

CCSD(T) benchmark on Cray XT5 : 24 water molecules



Scalability of the non-iterative EOMCC code

94 %parallel efficiency using 210,000 cores

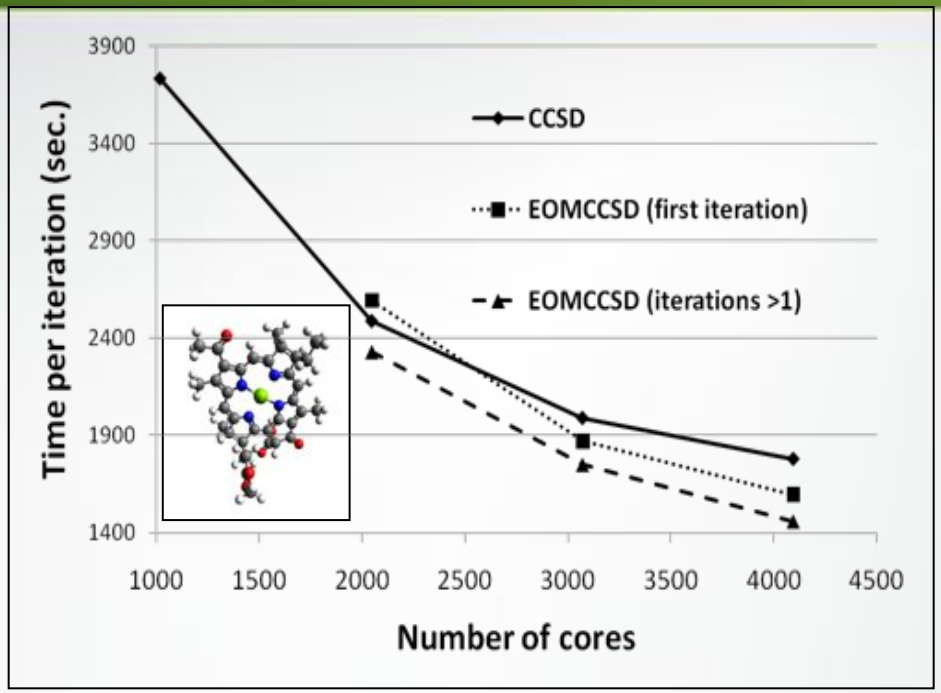


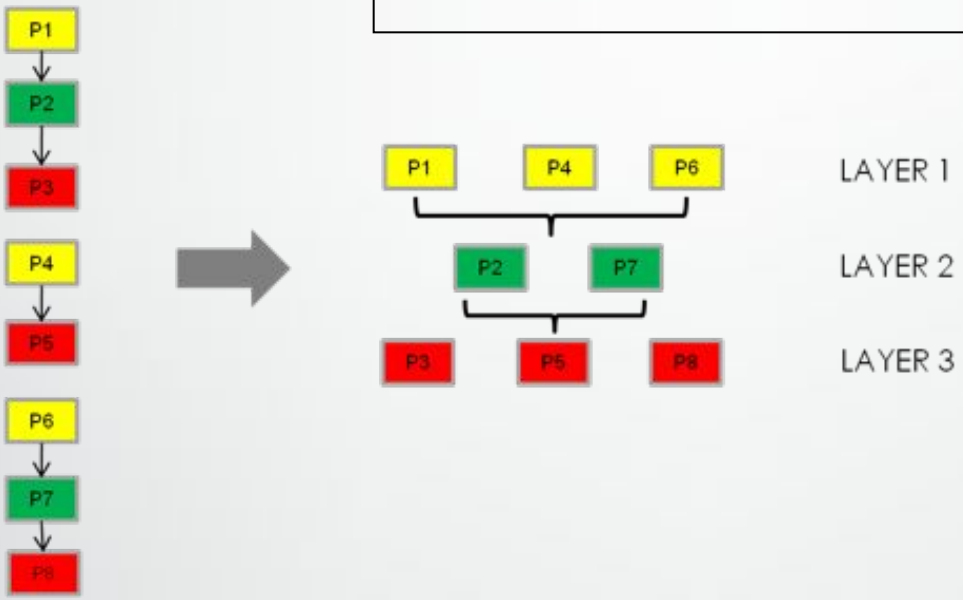
- Scalability of the triples part of the CR-EOMCCSD(T) approach for the FBP-f-coronene system in the AVTZ basis set. Timings were determined from calculations on the Jaguar Cray XT5 computer system at NCCS/ORNL in 2011

Scalability of the iterative EOMCC methods

Alternative task schedulers

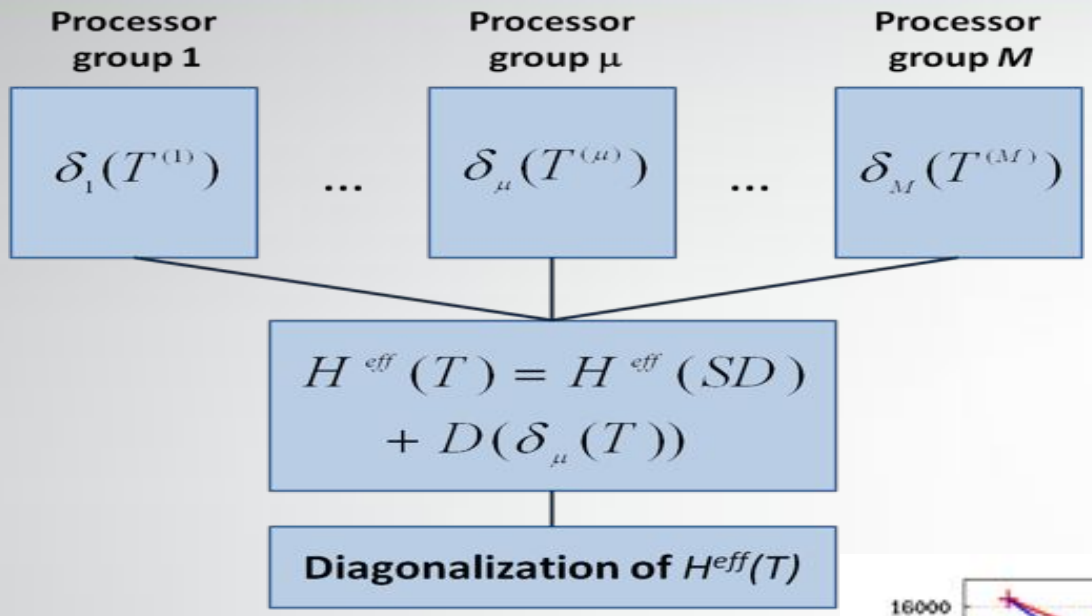
- use "global task pool"
- improve load balancing
- reduce the number of synchronization steps to absolute minimum





MRCCSD(T)

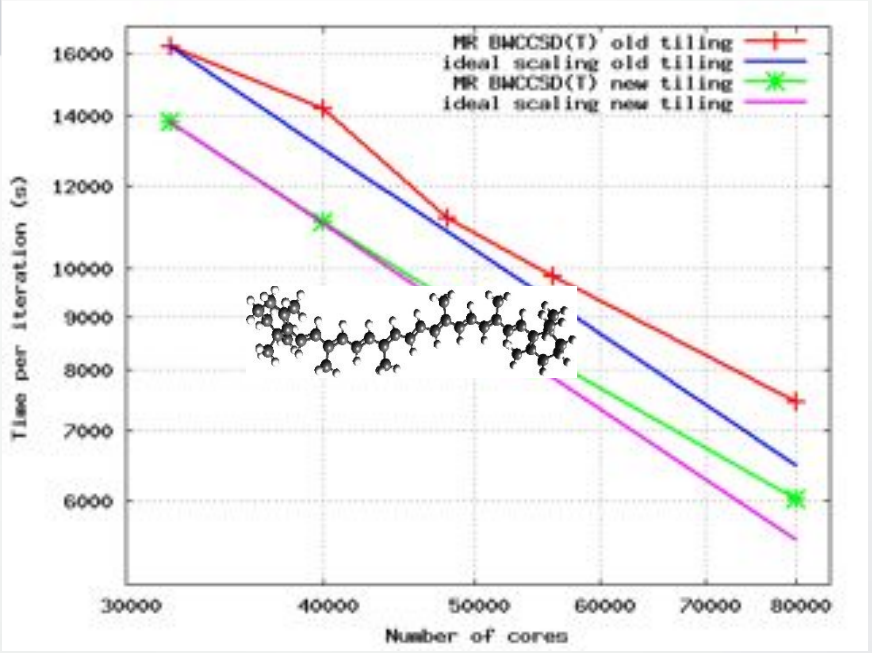
K. Kowalski
K. Bhaskaran-Nair



- Improves the quality of the MRCCSD approaches

Numerical complexity ~ $M \times N^7$

Scalability ~ M x *(scalability of the CCSD(T) approach)*



GPU CCSD(T) Benchmarks

Titan Cray XK7 system at ORNL

CCSD(T)

378 basis set functions, C1 symmetry;

98 nodes: 8 cores per node + 1GPU)

Pentacene $C_{22}H_{14}$

Using 0 GPU device(s) per node wall time / sec 9240.3

Using 1 **GPU** device per node wall time / sec 1630.7

Speed-up GPU vs CPU = 9240.3 /1630.7 ~ 5.6x.

NWChem Xeon Phi port: Design decisions

Identification of suitable kernel: **CCSD(T)**

- Many opportunities for data reuse → minimize data transfer
- Presence of floating-point intensive kernel: $tilesize^7$ flops
- Successfully used for GPU port
- Adopted ***offload*** approach

NWChem Xeon Phi port: Implementation

- Initial port relying on **OpenMP** target offloading
- Moved to **LEO** ← required flexibility for data transfer
- Did not modify existing GA parallelization structure
    - Load balancing determines work distribution: Global memory → Host Local Memory → offload to coprocessor
- Only incremental/localized changes to existing Fortran source
- Cons: adoption of proprietary development tool
- Wish: are OpenMP/OpenAcc/LEO eventually going to merge/converge into a standard for accelerators?)

NWChem Xeon Phi port: CCSD(T)

What is CCSD(T)?

Hierarchy of methods in Quantum Chemistry

Correlation Energy

$$E_{corr} = E_{exact} - E_{HF}$$

CCSD(T)

SD: Singles and Doubles Excitations (iterative)

(T): Triples perturbative excitations (single step)

$$E_{CCSD(T)} = \delta E_{SD} + \delta \mathbf{E_{(T)}}$$

Scheme of the algorithm

```
do [i]=1,noab
do [j]=[i],noab
do [k]=[j],noab
do [a]=1,nvab
do [b]=[a],nvab
do [c]=[b],nvab
```

parallel region

calculate $\left\langle \Phi_{ijk}^{abc} \middle| V_N T_2 \middle| \Phi \right\rangle$ i∈[i],j∈[j],k∈[k] a∈[a],b∈[b],c∈[c]

form contribution $\delta E^{CCSD(T)}$ to the CCSD(T) correction

```
enddo
enddo
enddo
enddo
enddo
enddo
```

$$\langle \Phi_{ijk}^{abc} | V_N T_2 | \Phi \rangle \quad =$$

$$triplesx(a,b,c,i,j,k) \quad -= \quad \sum_{e \in [e]} v_{ab}^{ei} t_{ec}^{jk}$$

$$(i \in [i], j \in [j], k \in [k] \quad , \quad a \in [a], b \in [b], c \in [c])$$

Code Snippet: offloading directives

```
cdir$ offload_transfer target(mic:device)
     N nocopy(triplesx:length(trsx_l) ALLOC)
cdir$ offload_transfer target(mic:device)
     N nocopy(t2sub:length(l_t2sub) ALLOC)
cdir$ offload_transfer target(mic:device)
     N nocopy(v2sub:length(l_v2sub) ALLOC)

cdir$ offload target(mic:device)
     N nocopy(triplesx:length(0) REUSE)
      call zero_triplesx(triplesx)
      do ...
        if (...)
cdir$ offload target(mic:device)
     N in(triplesx:length(0) REUSE)
     I in(h3d,h2d,h1d)
     I in(p6d,p5d,p4d,h7d)
     I in(t2sub:length(l_t2sub) REUSE)
     R in(v2sub:length(l_v2sub) REUSE)
         call sd_t_dX_Y(h3d,h2d,h1d,
     C                  p6d,p5d,p4d,h7,
     C                  triplesx,t2sub,v2sub)
        endif
      enddo

cdir$ offload_transfer target(mic:device)
     R out(triplesx:length(triplesx_l) REUSE)
```

Memory allocation on device

Initialize **6-D** output array triplesx

Do loop

Execute kernel on device
send: t2sub and v2sub
4-D arrays

End do loop

Receive triplesx

Kernel: generic loop

```
!$omp parallel do &
!$omp     private(p4,p5,p6,h2,h1,h3,h7) &
!$omp     collapse(OMPCOLLAPSE)
do p4=1,p4d
do p5=1,p5d
do p6=1,p6d
do h2=1,h2d
do h3=1,h3d
do h7=1,h7d
do h1=1,h1d
  triplesx(h1,h3,h2,p6,p5,p4) =
    triplesx(h1,h3,h2,p6,p5,p4)
      - t2sub(h7,p4,p5,h1)*v2sub(h3,h2,p6,h7)
enddo
enddo
enddo
enddo
enddo
enddo
enddo
!$omp end parallel do
```

OMPCOLLAPSE=3
Multi-threading on 3 outermost loops

✓ Stride 1 loop as innermost

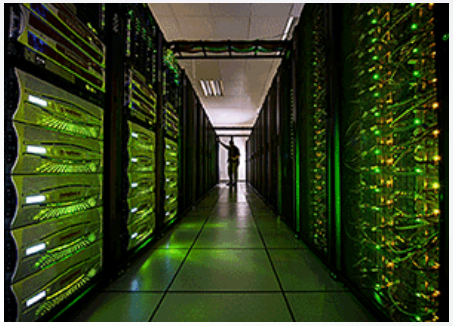
Kernel: collapsed loop …

```
!$omp parallel do &
!$omp     private(p4,p5,p6,h2,h1,h3,h7) &
!$omp     collapse(OMPCOLLAPSE)
do p4=1,p4d
do p5=1,p5d
do p6=1,p6d
do h1=1,h1d
do h7=1,h7d
!dec$ loop count max=1000, min=20
do h3h2=1,h2d*h3d
  triplesx(h3h2,h1,p6,p5,p4) =
    triplesx(h3h2,h1,p6,p5,p4)
      - t2sub(h7,p4,p5,h1)*v2sub(h3h2,p6,h7)
enddo
enddo
enddo
enddo
enddo
enddo
!$omp end parallel do
```

OMPCOLLAPSE=3
Multi-threading on 3 outermost loops

- ✓ 2 innermost loops collapsed
- ✓ Directive to inform compiler of loop count

EMSL cascade Hardware specs



- Theoretical peak: 3.4 Pflops, Linpack run: 2.5 Pflops
- 1440 compute nodes, compute nodes, on each node
  - Two 2.6GHz Intel Xeon E5-2670 8-core processors
  - Two Intel Xeon Phi 5110P
  - 128 GB DDR3 memory
- FDR Infiniband network
- 2.7 petabyte shared parallel filesystem (60 Gb/sec)

NWChem Xeon Phi port: Experimental Setup

- Out of the sixteen cores available on each node:

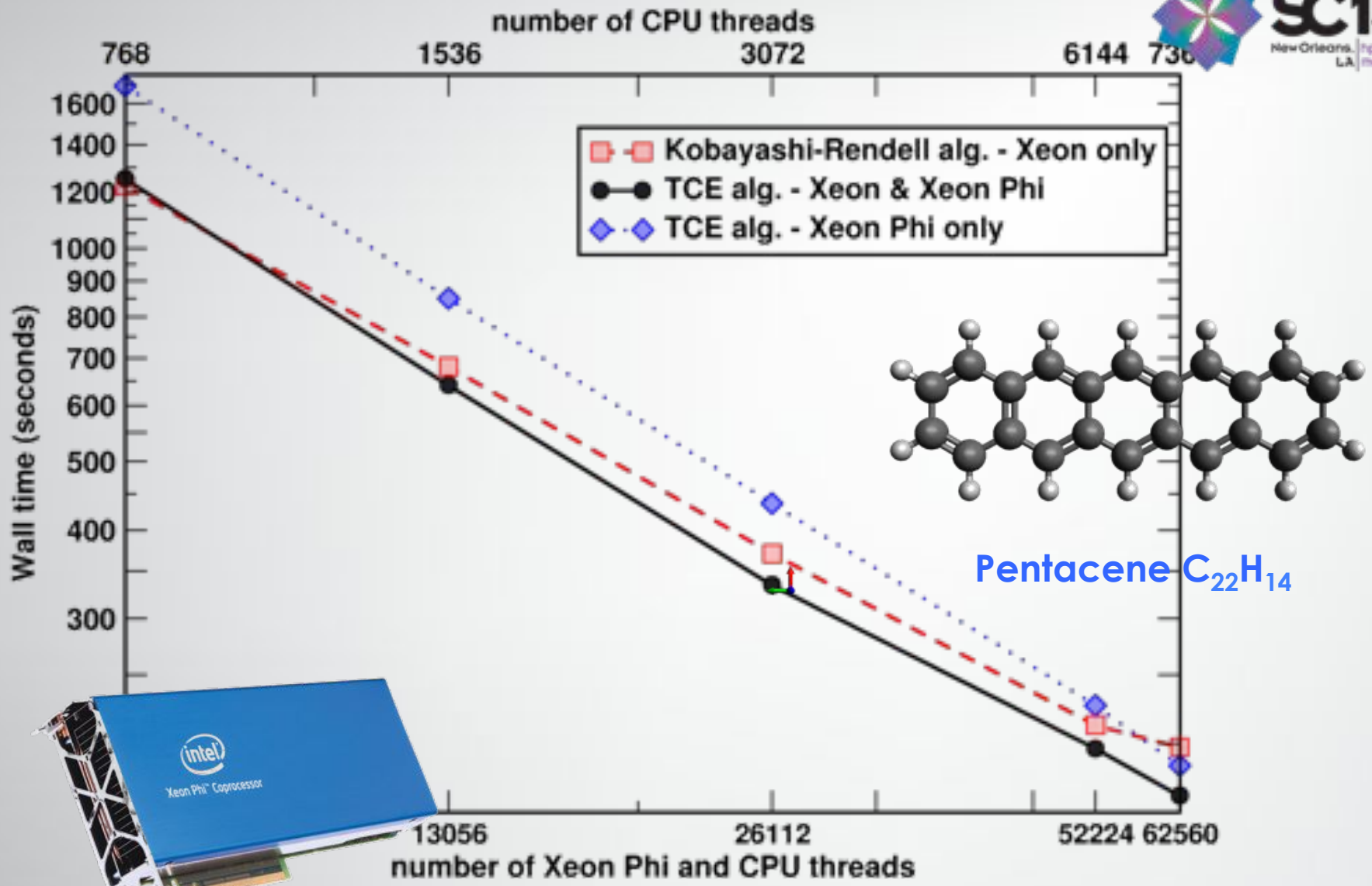
- **Four** ranks dedicate to **offload**
  - two ranks communicate with each Intel Xeon Phi
  - first rank uses MIC cores 0-28
  - second rank uses MIC cores 29-56
  - All four MIC HW threads are used

- **Four** ranks perform traditional **CPU** work (e.g. number crunching)
  - each one of them spawns four OpenMP threads
  - all sixteen cores are utilized

- **`MIC_USE_2MB_BUFFERS=16K`** reduced data transfer time by ~ 30%

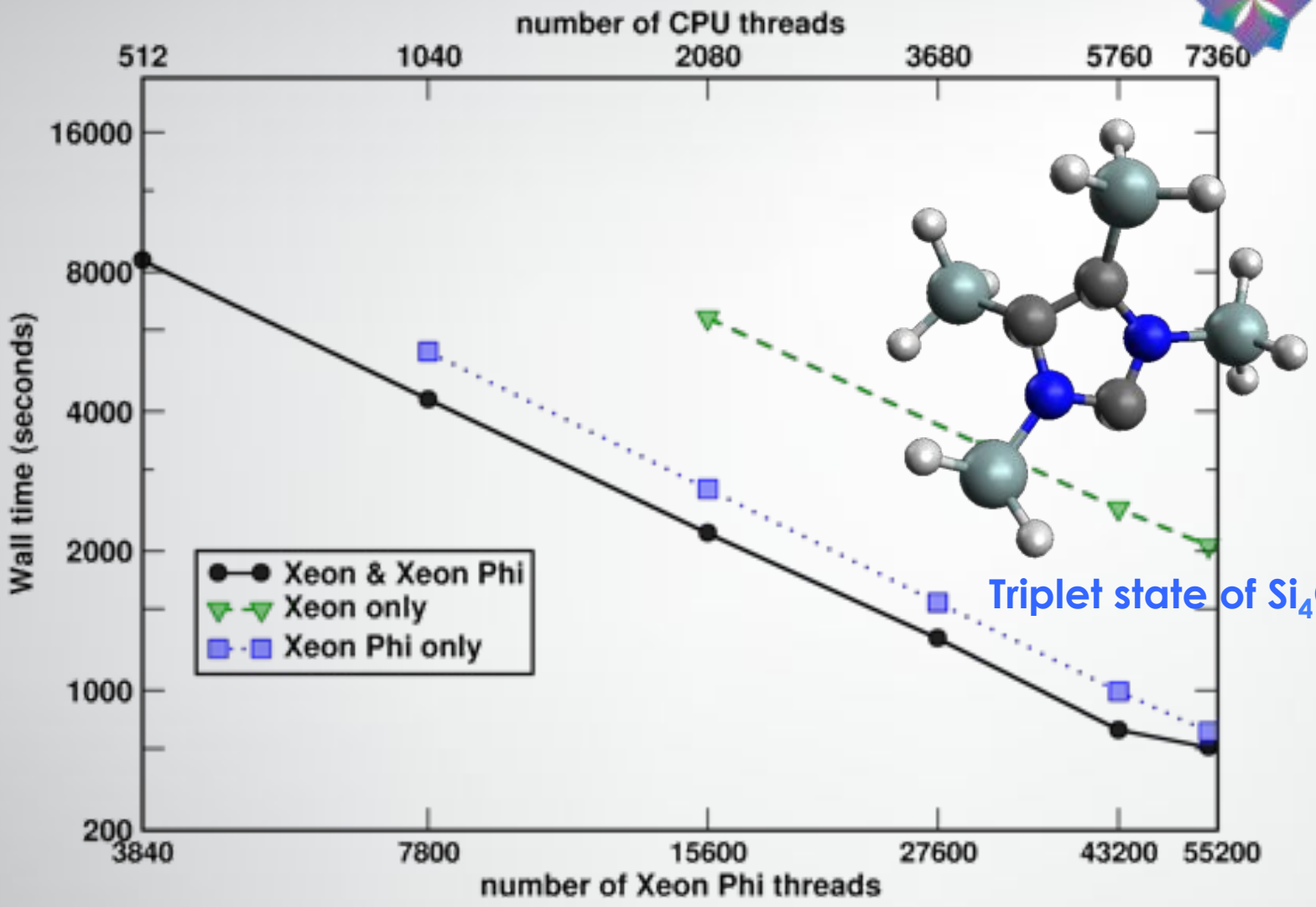
CCSD(T) on Intel Xeon Phi Coprocessor

E. Aprà
M. Klemm
K. Kowalski



CCSD(T) on Intel Xeon Phi Coprocessor

E. Aprà
M. Klemm
K. Kowalski



Triplet state of $Si_4C_3N_2H_{12}$

Comments/Conclusions

Key to Xeon Phi performance:

- multi-threading (OpenMP on outer loops)
- SIMD vectorization (hand tuning, directives, alignment, etc … on innermost loop)
- minimize/hide cost of data transfer

For more information, visit EMSL's website

www.emsl.pnnl.gov