Efficient MPI/OpenMP parallelization of the Hartree-Fock method on Intel® Xeon Phi processors

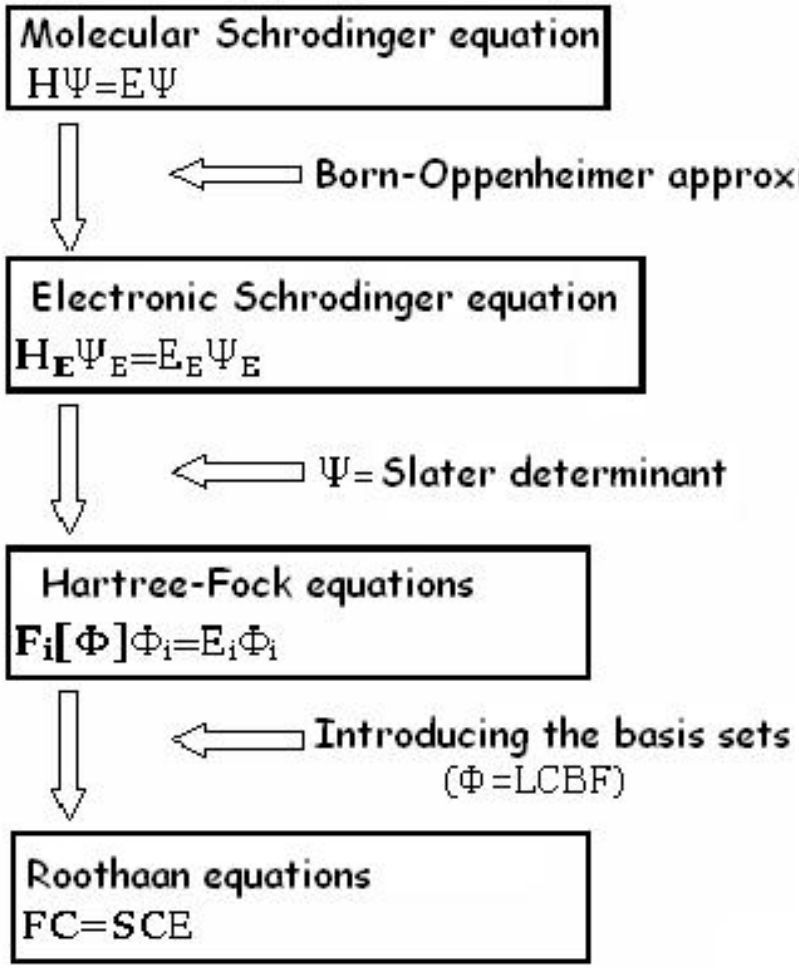
Yuri Alexeev[1],

Vladimir Mironov[2], Alexandr Moskovsky[2]

[1]Argonne National Laboratory, Argonne, IL, USA

[2]Lomonosov Moscow State University, Russia

Hartree-Fock Introduction



Algorithm 1 Hartree-Fock algorithm

1: *Calculate 1e contribution to the Fock matrix*
2: *Calculate exchange integrals for ERI screening*
3: *Guess initial density matrix* $\mathbf{D}$
4: **repeat**
5: *Calculate 2e contribution to the Fock matrix*
6: *Add 1e contribution to Fock matrix*
7: *Solve Fock equation,*
update energy and density matrix $(\epsilon, \mathbf{D_{new}})$
8: **until** $|\mathbf{D_{new}} - \mathbf{D}| < conv$
9: *Calculate properties, print results*

Hartree-Fock Introduction

2-e Electron Repulsion Integrals (ERIs) scale as N^4:

$$(i,j|k,l) = \iint dr_1 dr_2 \chi_i(r_1)\chi_j(r_1)\frac{1}{|r_1 - r_2|}\chi_k(r_2)\chi_l(r_2)$$

Fock update ($D_{ij} = \sum_k C_{ik}C_{jk}$):

$$F_{ij} \leftarrow (i,j|k,l) \cdot D_{kl}$$
$$F_{kl} \leftarrow (i,j|k,l) \cdot D_{ij}$$
$$F_{ik} \leftarrow (i,j|k,l) \cdot D_{jl}$$
$$F_{jl} \leftarrow (i,j|k,l) \cdot D_{ik}$$
$$F_{il} \leftarrow (i,j|k,l) \cdot D_{jk}$$
$$F_{jk} \leftarrow (i,j|k,l) \cdot D_{il}$$

Challenges and Solutions

- Challenges
  - Monolithic code (~2 million lines) written decades ago in Fortran 77
  - Optimized for old CPU architectures (cache unfriendly)
  - Global data structures (COMMON blocks) are used throughout the code
  - Use of obsolete constructs (IF computed GOTO, EQUIVALENCE etc.)
  - Custom dynamic array allocator (hard to debug)
  - High memory demand of quantum chemistry algorithms
- Why we care
  - Over 200,000 users world wide (100+ active code developers, GAMESS-UK, Firefly (former PC-GAMESS))
  - One of the most popular quantum chemistry packages
  - INCITEs, ESPs, ECP, 2 IPCCs
- Solutions
  - Rewrite critical subroutines in Fortran 95
  - Remove global data structures
  - OpenMP parallelization and vectorization

Hartree-Fock original MPI code

Algorithm 2 Fock matrix algorithm in original GAMESS MPI code

```
1: for I = 1, NSH do
2:     for J = 1, I do
3:         J-index MPI dynamic load balance
4:         for K = 1, I do
5:             (K == I)?L_max = K : L_max = J
6:             for L = 1, L_max do
7:                 Schwartz screening
8:                 if not screened then
9:                     Calculate (I,J—K,L)
10:                    Update local 2e Fock matrix
11:                end if
12:            end for
13:        end for
14:    end for
15: end for
16: 2e Fock matrix reduction
```

Algorithm 3 Fock matrix algorithm in new GAMESS OpenMP code

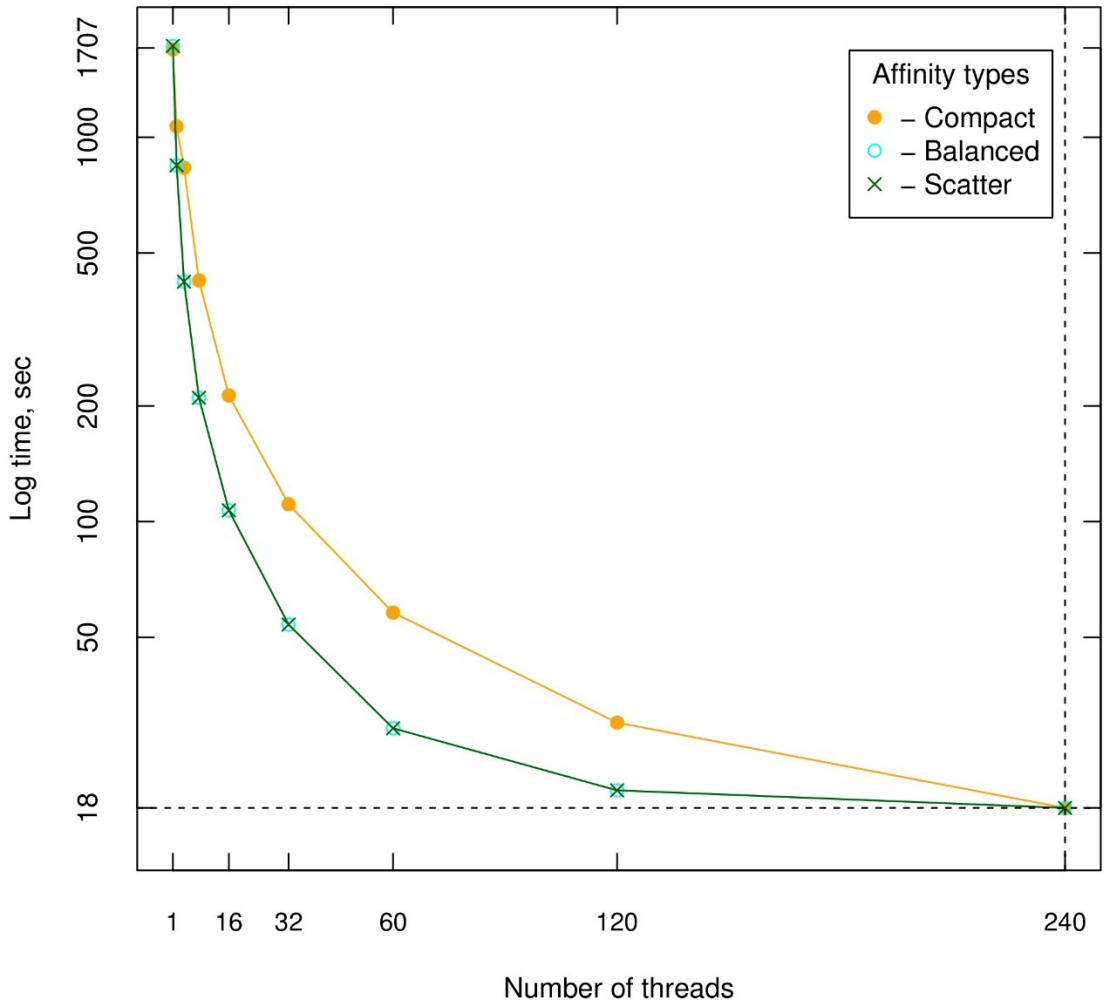
```
1: if not sorted then
2:     Shell sorting
3: end if
4: Sort density matrix
5: !$omp parallel private(J, K, L, L_max) shared(I)
       reduction(+ : Fock)
6: loop
7:     !$omp master
8:     Get new I index over MPI
9:     !$omp end master
10:    !$omp barrier
11:    !$omp do collapse(2) schedule(dynamic,1)
12:    for J = 1, I do
13:        for K = 1, I do
14:            (K == I)?L_max = K : L_max = J
15:            for L_1 = 1, L_max, Nstride do
16:                L_2 = min(L_1 + Nstride − 1, L_max)
17:                for L = L_1, L_2 do
18:                    Schwartz screening
19:                    if not screened then
20:                        Calculate (I, J|K, L)
21:                    end if
22:                end for
23:                Update local 2e Fock matrix
24:            end for
25:        end for
26:    end for
27:    !$omp end do
28: end loop
29: !$omp end parallel
30: Revert 2e Fock matrix to original order
31: Pack 2e Fock matrix to triangular form
32: 2e Fock matrix reduction
```

OpenMP scaling on single Xeon Phi

Thread affinity impact:

Graphene bilayer benchmark
60 carbon atoms, 6-31G basis set

- OpenMP code scaling up to 240 threads
- Thread affinity does not matter at thread limit
- It is better to distribute threads over cores otherwise
- 1x Xeon Phi 5120D

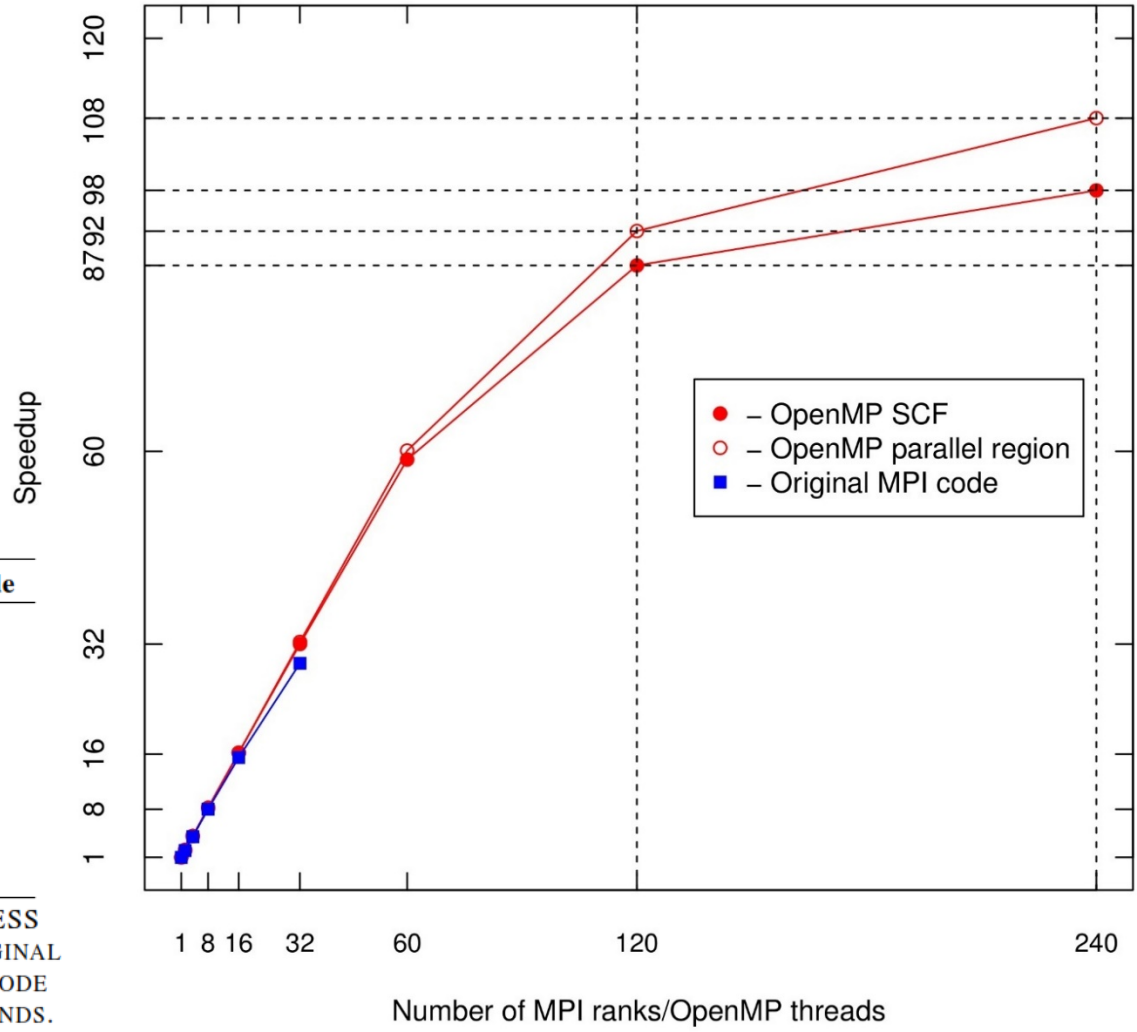


OpenMP and MPI scaling on single Xeon Phi

Scalability of original GAMESS code and OpenMP version on a single Intel Xeon Phi card using C60 benchmark

# OpenMP threads/MPI ranks	OpenMP code	MPI code
1	1728	1723
2	845	852
4	421	425
8	210	215
16	107	111
32	54	59
60	29	–
120	20	–
240	18	–

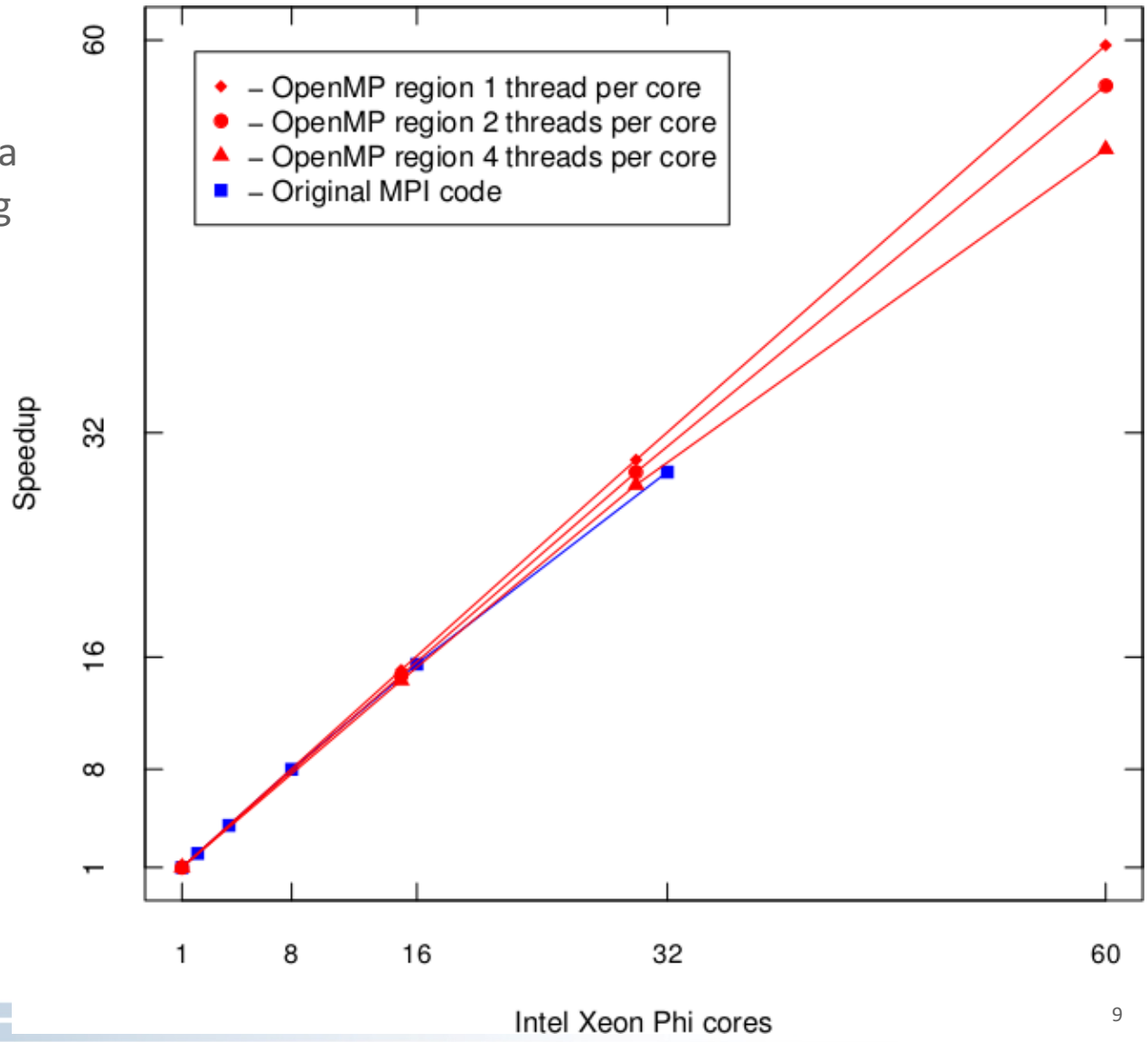
TABLE 1. COMPARISON OF TIMINGS FOR THE ORIGINAL GAMESS MPI VS NEW HYBRID GAMESS MPI/OPENMP CODES. THE ORIGINAL MPI CODE RUNS ONLY OVER MPI RANKS, WHILE THE HYBRID CODE UTILIZES ONLY A SINGLE MPI RANK. ALL TIMINGS ARE IN SECONDS.



OpenMP and MPI scaling on single Xeon Phi

Scalability of original GAMESS code and OpenMP version on a single Intel Xeon Phi card using C60 benchmark

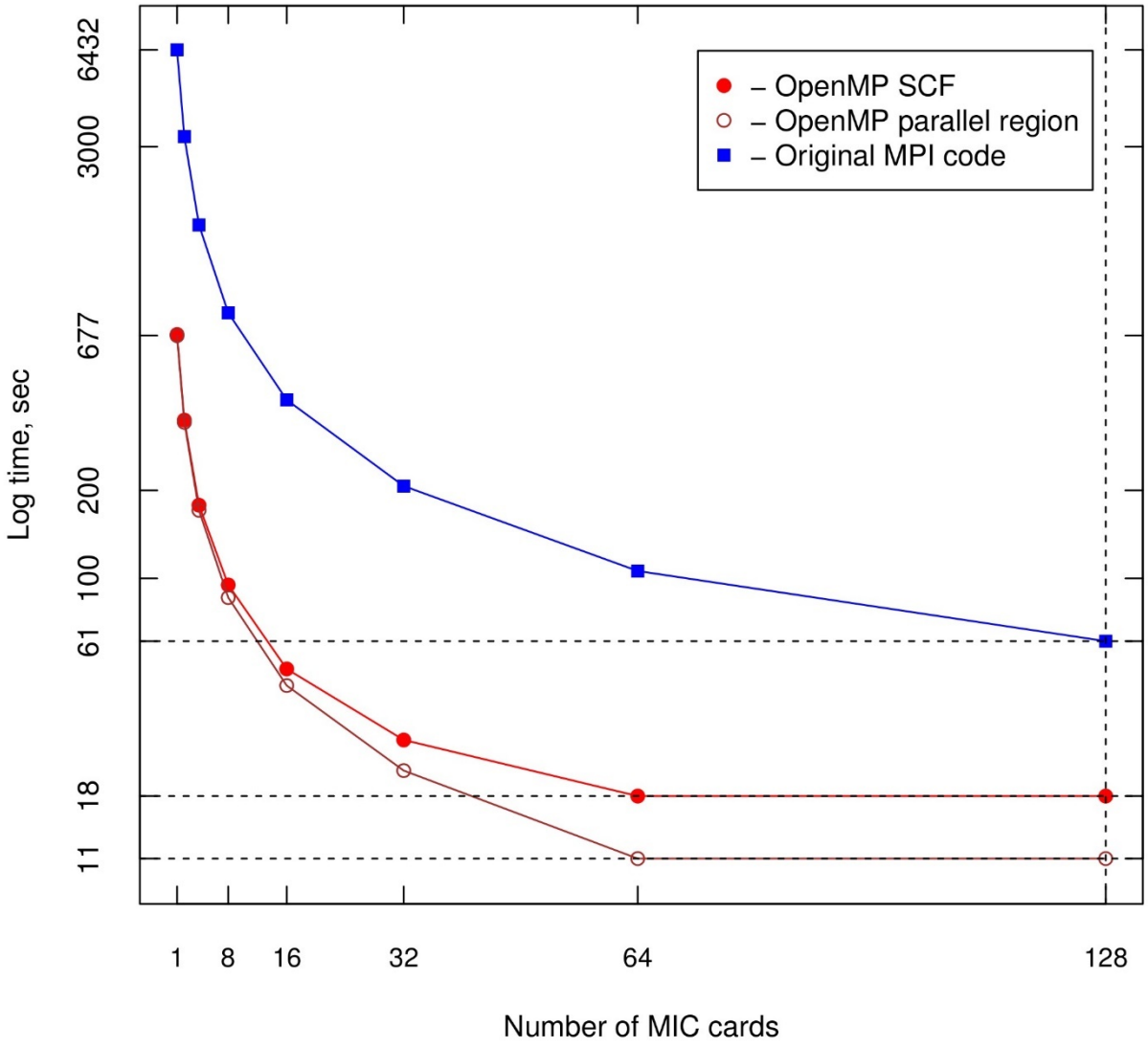
The performance of OpenMP code is very similar with 1/2/4 threads per core



OpenMP and MPI scaling on many Xeon Phi

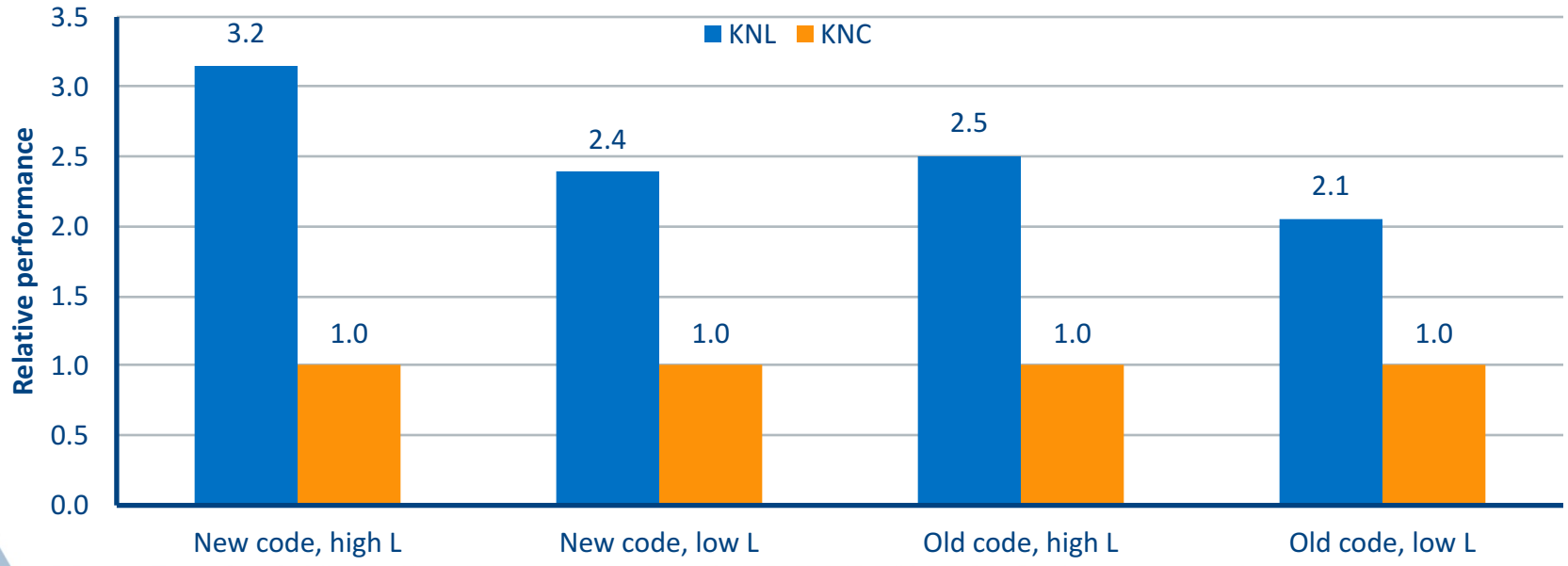
Graphene bilayer benchmark
240 carbon atoms, 6-31G basis set

- Both original and hybrid codes scale well
- Original code requires DAPL UD feature to work (but not OpenMP code)
- Memory footprint of original MPI code is too high for KNC
- RSC PetaStream (based on Xeon Phi 5120D)



Preliminary KNL results

- Early KNL results for new ERI code:
  - Intel® Xeon™ 7250 (68 cores, All-to-All mode)
  - No KNL-specific code optimization
  - KNL vs KNC (Xeon Phi 7120D):
    2-3x performance improvement even for all-to-all cluster mode
  - MCDRAM vs DDR4:
    2-5x performance difference
  - On small working set cache memory mode works as well as flat
- Vladimir is working on data vector length increase to get more performance



Scaling on Theta

Chemical system: mesoporous silicate nanoparticle (MSN) - 1770 atoms, 8946 basis functions

Non-threaded Hartree-Fock calculations with STO-3G basis set

64 MPI ranks per node, cache/quad mode

These preliminary scaling benchmarks suggest that we should be able to scale calculations on all Theta nodes

Nodes	Cores	Wall Time (s)	Speed up
128	8192	942.5	1.00
256	16384	485.3	1.94
512	32768	202.2	4.67

Results

- ERI code is now thread-safe (Rotated axis and Rys ERI)
- Implemented OpenMP parallelization in SCF driver
- Significantly reduced memory footprint (only Fock matrix is replicated)
- Threaded energy code for RHF, UHF, ROHF, and HF-exchange part in hybrid DFT
- Threaded gradient code for RHF, UHF, ROHF, and HF-exchange part in hybrid DFT

Acknowledgments

- This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357
- Vladimir Mironov thanks Intel® Parallel Compute Center program for funding
- Authors thank the Supercomputing Center of Peter the Great St. Petersburg Polytechnic University
- Authors thank Kristopher Keipert for providing Theta benchmarking data