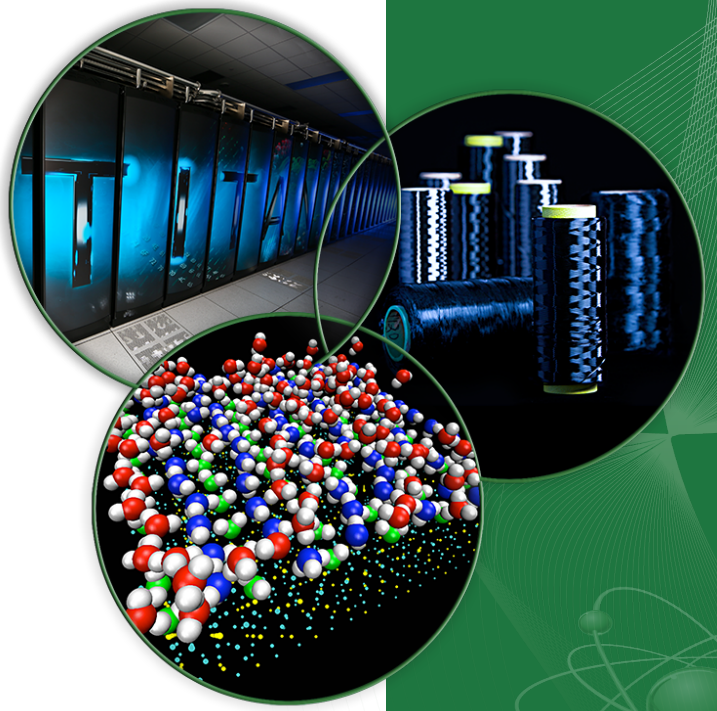
Experiences with using different OpenMP 4.5 programming styles to bring DMRG++ to Exascale



Arghya Chatterjee (ORNL / Georgia Tech.)
Oscar Hernandez (ORNL)
Vivek Sarkar (Georgia Tech.)

Application Support Team
Dr. E. D'Azevedo (CNMS)
Dr. G. Alvarez (CNMS)
Dr. Wael Elwasif (ORNL)

ORNL is managed by UT-Battelle
for the US Department of Energy

OpenMP Con 2017
September 18, 2017

OAK RIDGE National Laboratory

INTRODUCTION

OAK RIDGE National Laboratory

INTRODUCTION

- Rapidly changing microprocessor design and heterogeneous architectures

OAK RIDGE National Laboratory

INTRODUCTION

- Rapidly changing microprocessor design and heterogeneous architectures
- Applications must adapt to exploit parallelism

OAK RIDGE National Laboratory

INTRODUCTION

- Rapidly changing microprocessor design and heterogeneous architectures
- Applications must adapt to exploit parallelism
- Mini-application -- will serve as a foundation for Exascale-ready implementation

INTRODUCTION

- Rapidly changing microprocessor design and heterogeneous architectures
- Applications must adapt to exploit parallelism
- Mini-application -- will serve as a foundation for Exascale-ready implementation
- Co-designing with asynchronous programming models (Habanero C++, Kokkos, MAGMA)

OUTLINE

OUTLINE

DMRG++

Target
Application

OUTLINE

Target Application

Physics / Nano-science & Programming Models

OUTLINE

Target Application

Physics / Nano-science & Programming Models

Design, Implementation & Challenges encountered

OUTLINE

Target Application

Physics / Nano-science & Programming Models

Design, Implementation & Challenges encountered

Programming framework support

OUTLINE

Target Application

Physics / Nano-science & Programming Models

Design, Implementation & Challenges encountered

Programming framework support

Adding new Constructs / functionality

DMRG++

OAK RIDGE National Laboratory

DMRG ++

- DMRG++ → *Density Matrix Renormalization Group*

OAK RIDGE National Laboratory

DMRG ++

- DMRG++ → *Density Matrix Renormalization Group*

- DMRG++ algorithm -- deeper understanding of nanoscale material properties

OAK RIDGE National Laboratory

DMRG ++

- DMRG++ → *Density Matrix Renormalization Group*
- DMRG++ algorithm -- deeper understanding of nanoscale material properties
- *Sparse matrix algebra* computational motif

OAK RIDGE National Laboratory

DMRG ++

- DMRG++ → *Density Matrix Renormalization Group*
- DMRG++ algorithm -- deeper understanding of nanoscale material properties
- *Sparse matrix algebra* computational motif
- Actively developed application by Material Applications group @ORNL

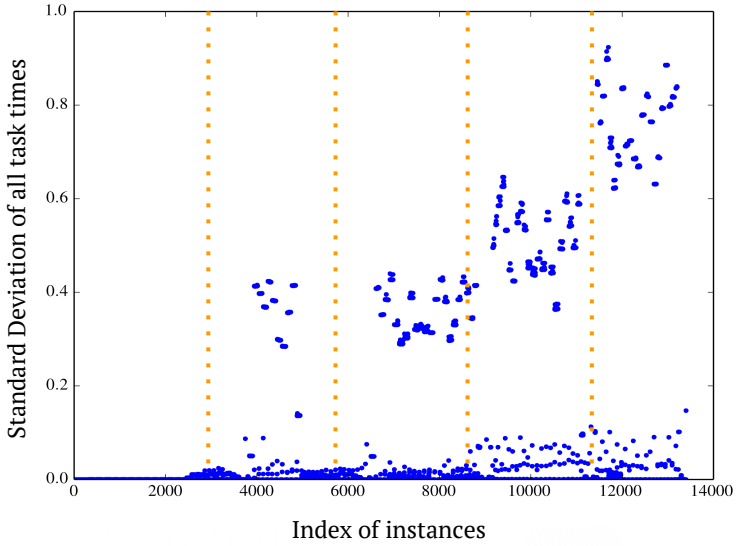
OAK RIDGE National Laboratory

PROFILING : Shared Memory (DMRG++)

OAK RIDGE National Laboratory

PROFILING : Shared Memory (DMRG++)

- Profiled on a Bulldozer AMD Opteron processor (TITAN)
- 80% of execution time → calculating the Hamiltonian

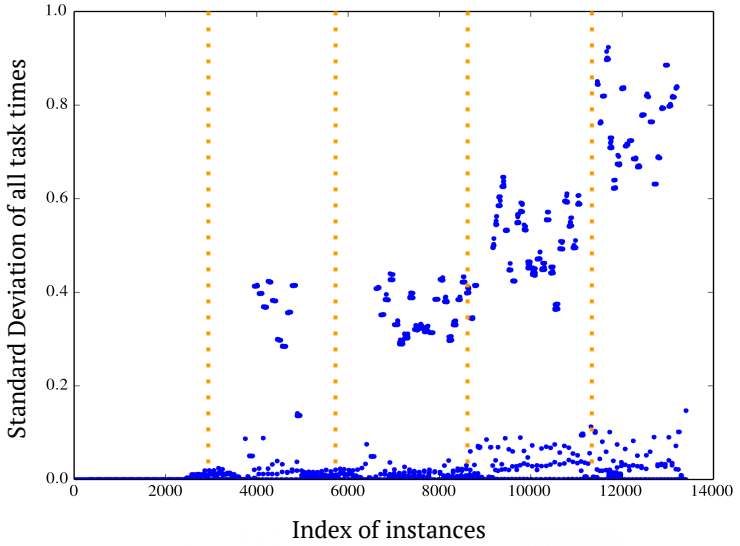


Standard deviation of all execution times per parallel region instance over time. Execution uses 8 threads on a single TITAN node (8 blue dots per instance).

OAK RIDGE National Laboratory

PROFILING : Shared Memory (DMRG++)

- Profiled on a Bulldozer AMD Opteron processor (TITAN)
- 80% of execution time → calculating the Hamiltonian
- Application runs in phases (graph shows 5 phases)

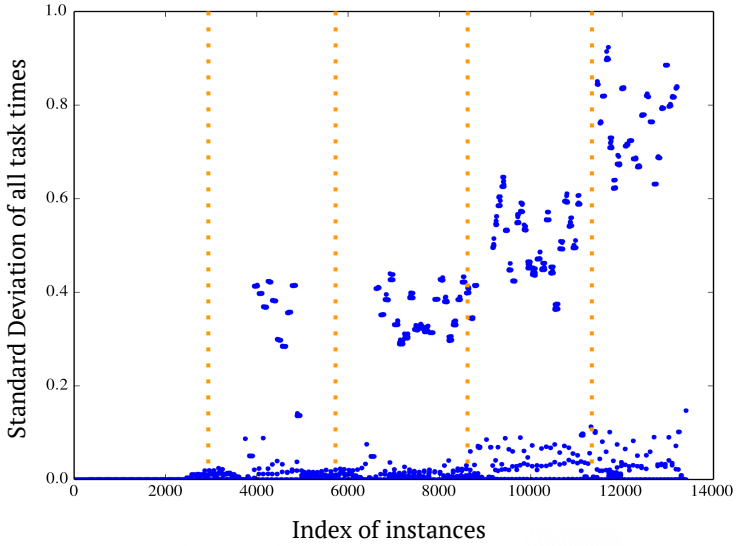


Standard deviation of all execution times per parallel region instance over time. Execution uses 8 threads on a single TITAN node (8 blue dots per instance).

OAK RIDGE National Laboratory

PROFILING : Shared Memory (DMRG++)

- Profiled on a Bulldozer AMD Opteron processor (TITAN)
- 80% of execution time → calculating the Hamiltonian
- Application runs in phases (graph shows 5 phases)
- Shows significant load imbalance

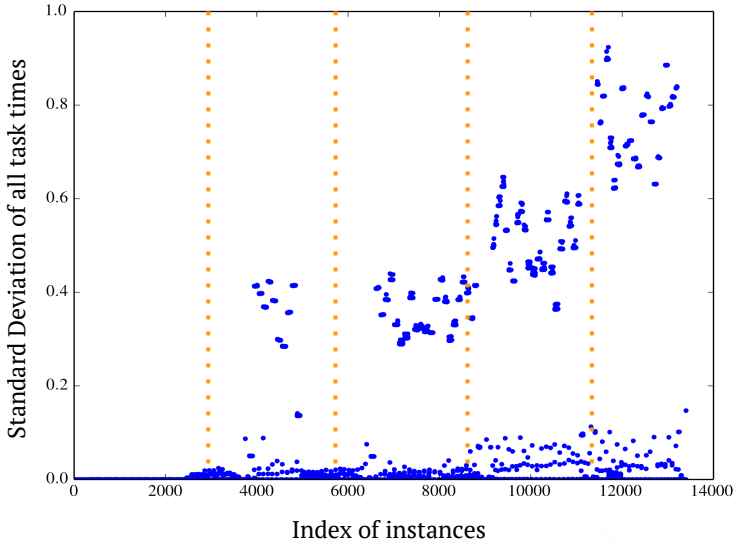


Standard deviation of all execution times per parallel region instance over time. Execution uses 8 threads on a single TITAN node (8 blue dots per instance).

OAK RIDGE National Laboratory

PROFILING : Shared Memory (DMRG++)

- Profiled on a Bulldozer AMD Opteron processor (TITAN)
- 80% of execution time → calculating the Hamiltonian
- Application runs in phases (graph shows 5 phases)
- Shows significant load imbalance
- Dynamic Application → problem size grows → greater load imbalance



Standard deviation of all execution times per parallel region instance over time. Execution uses 8 threads on a single TITAN node (8 blue dots per instance).

OAK RIDGE National Laboratory

MOTIVATION — Physics

OAK RIDGE National Laboratory

MOTIVATION — Physics

- Current DMRG++ — limited to 1-Dimensional Problems

OAK RIDGE National Laboratory

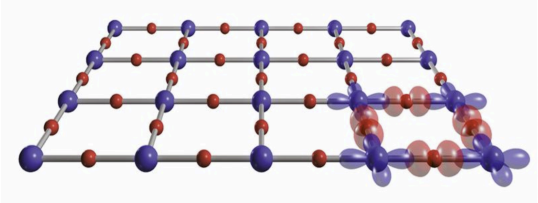
MOTIVATION — Physics

- Current DMRG++ — limited to 1-Dimensional Problems
- Scaling up — enables:

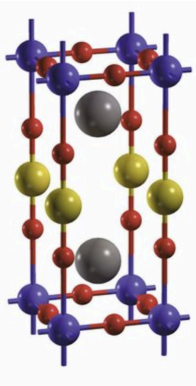
OAK RIDGE National Laboratory

MOTIVATION — Physics

- Current DMRG++ — limited to 1-Dimensional Problems
- Scaling up — enables:
  - ✤ Practical solution for 2-D and 3-D problems



2-D model

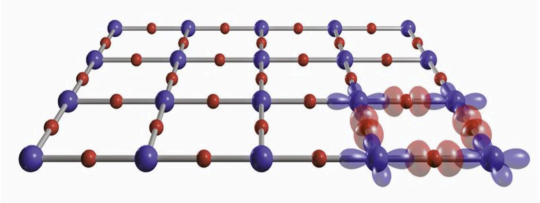


3-D model

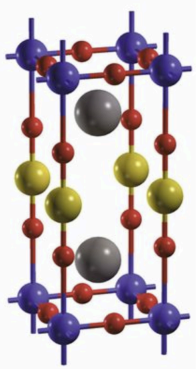
OAK RIDGE National Laboratory

MOTIVATION — Physics

- Current DMRG++ — limited to 1-Dimensional Problems
- Scaling up — enables:
  - ✤ Practical solution for 2-D and 3-D problems
  - ✤ First principle (almost) models without approximating electron-electron interaction



2-D model



3-D model

OAK RIDGE National Laboratory

MOTIVATION — Programming Models

OAK RIDGE National Laboratory

MOTIVATION — Programming Models

- OpenMP
  - ❖ Nested parallelism
  - ❖ OpenMP tasks — address load imbalance

OAK RIDGE National Laboratory

MOTIVATION — Programming Models

- OpenMP
  - ❖ Nested parallelism
  - ❖ OpenMP tasks — address load imbalance

- Exploring asynchronous task based programming models
  - ❖ New ideas could be candidates for future OpenMP extensions

↑ Productivity ↑ Performance

OAK RIDGE National Laboratory

MOTIVATION — Programming Models

- OpenMP
  - ❖ Nested parallelism
  - ❖ OpenMP tasks — address load imbalance
- Exploring asynchronous task based programming models
  - ❖ New ideas could be candidates for future OpenMP extensions

  ↑ Productivity ↑ Performance
- Compiler optimizations

OAK RIDGE National Laboratory

EXECUTION MODEL

OAK RIDGE National Laboratory

EXECUTION MODEL

- DMRG++ → *compute intensive:* calculating the sparse matrix *Hamiltonian*

OAK RIDGE National Laboratory

EXECUTION MODEL

- DMRG++ → *compute intensive:* calculating the sparse matrix *Hamiltonian*
- Two-dimensional dense/sparse matrix multiplication

OAK RIDGE National Laboratory

EXECUTION MODEL

- DMRG++ → *compute intensive*: calculating the sparse matrix *Hamiltonian*
- Two-dimensional dense/sparse matrix multiplication
- Programming styles (using OpenMP 4.5 constructs) :

OAK RIDGE National Laboratory

EXECUTION MODEL

- DMRG++ → *compute intensive*: calculating the sparse matrix *Hamiltonian*
- Two-dimensional dense/sparse matrix multiplication
- Programming styles (using OpenMP 4.5 constructs) :
    - ✤ Nested Parallelism

OAK RIDGE National Laboratory

EXECUTION MODEL

- DMRG++ → *compute intensive*: calculating the sparse matrix *Hamiltonian*
- Two-dimensional dense/sparse matrix multiplication
- Programming styles (using OpenMP 4.5 constructs) :
  - ✤ Nested Parallelism
  - ✤ Multi-level Tasking

OAK RIDGE National Laboratory

EXECUTION MODEL

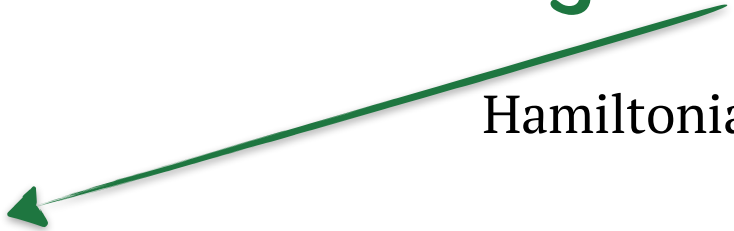
- DMRG++ → *compute intensive*: calculating the sparse matrix *Hamiltonian*
- Two-dimensional dense/sparse matrix multiplication
- Programming styles (using OpenMP 4.5 constructs) :
  - ✤ Nested Parallelism
  - ✤ Multi-level Tasking
  - ✤ Multi-level Tasking with Nested Parallelism

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

OAK RIDGE National Laboratory

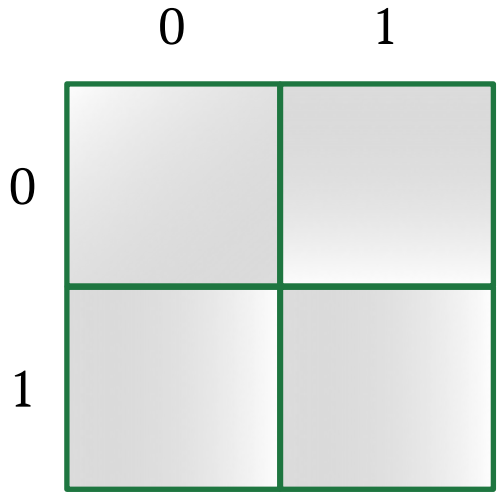
KRONECKER PRODUCT (y = H*x)



OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

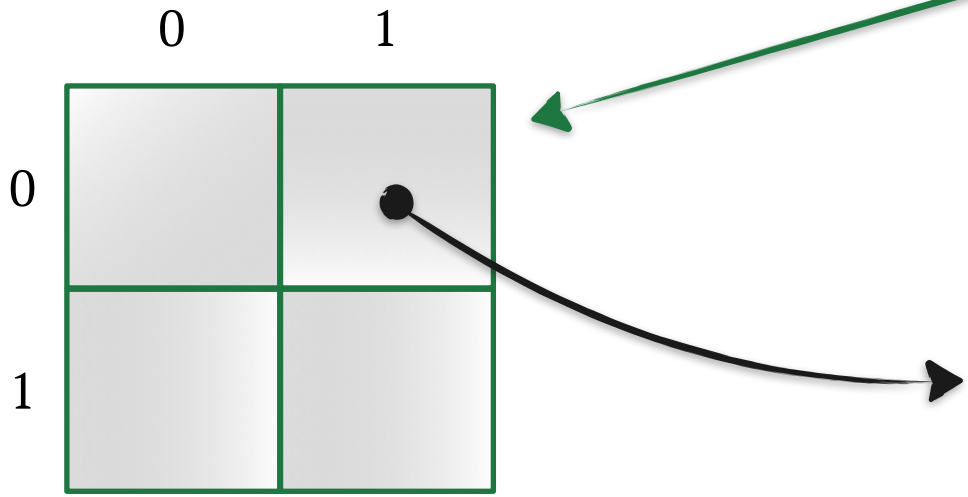
Hamiltonian Matrix



CIJ [npatches : npatches]

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

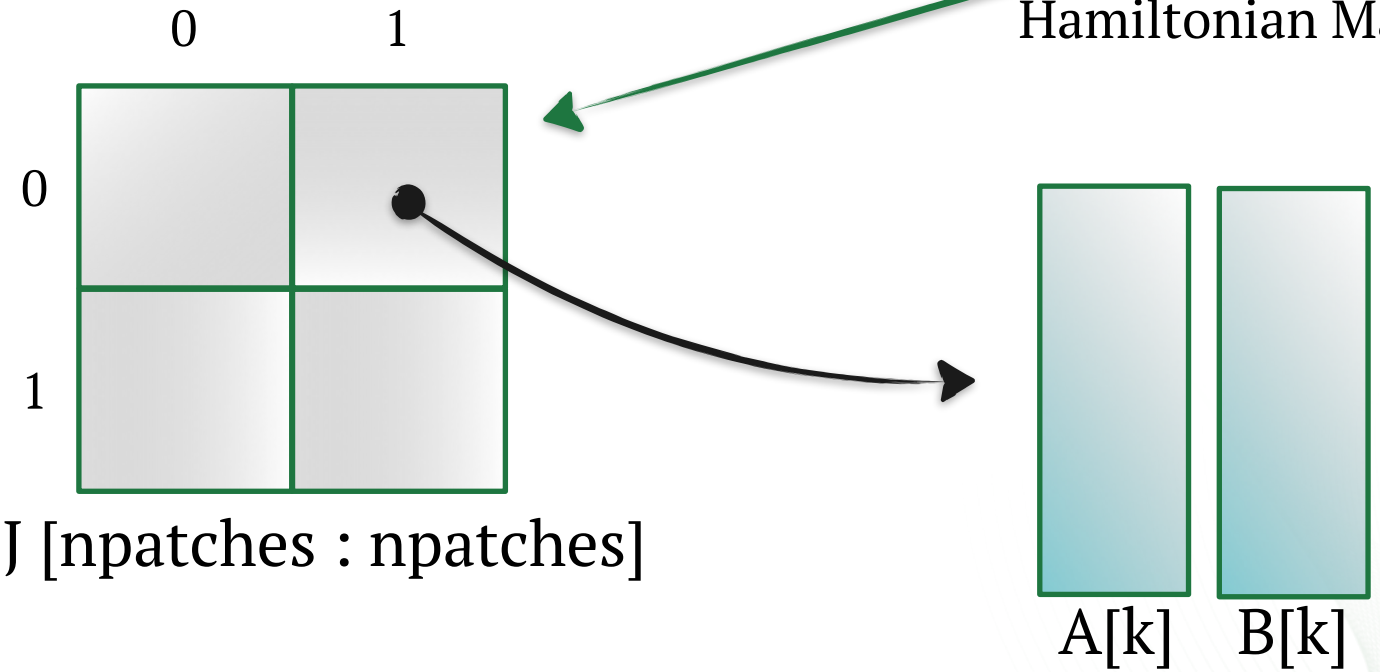


Hamiltonian Matrix

CIJ [npatches : npatches]

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

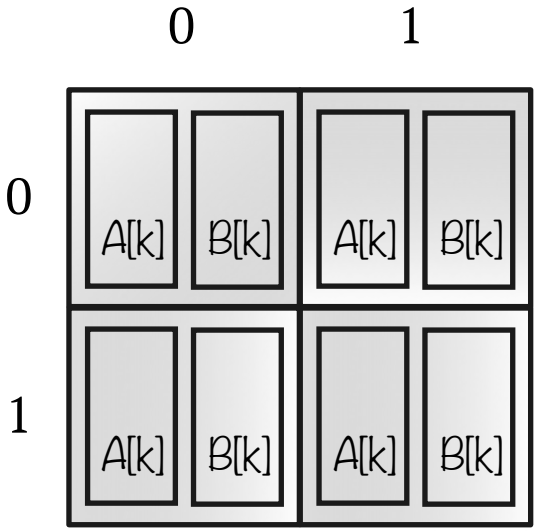


OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)

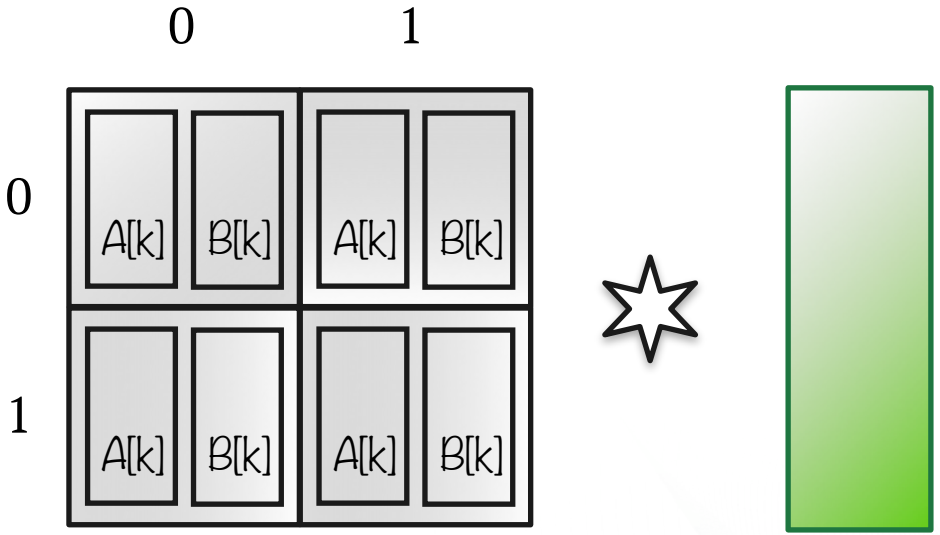


CIJ [npatches : npatches]

Hamiltonian Matrix

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)



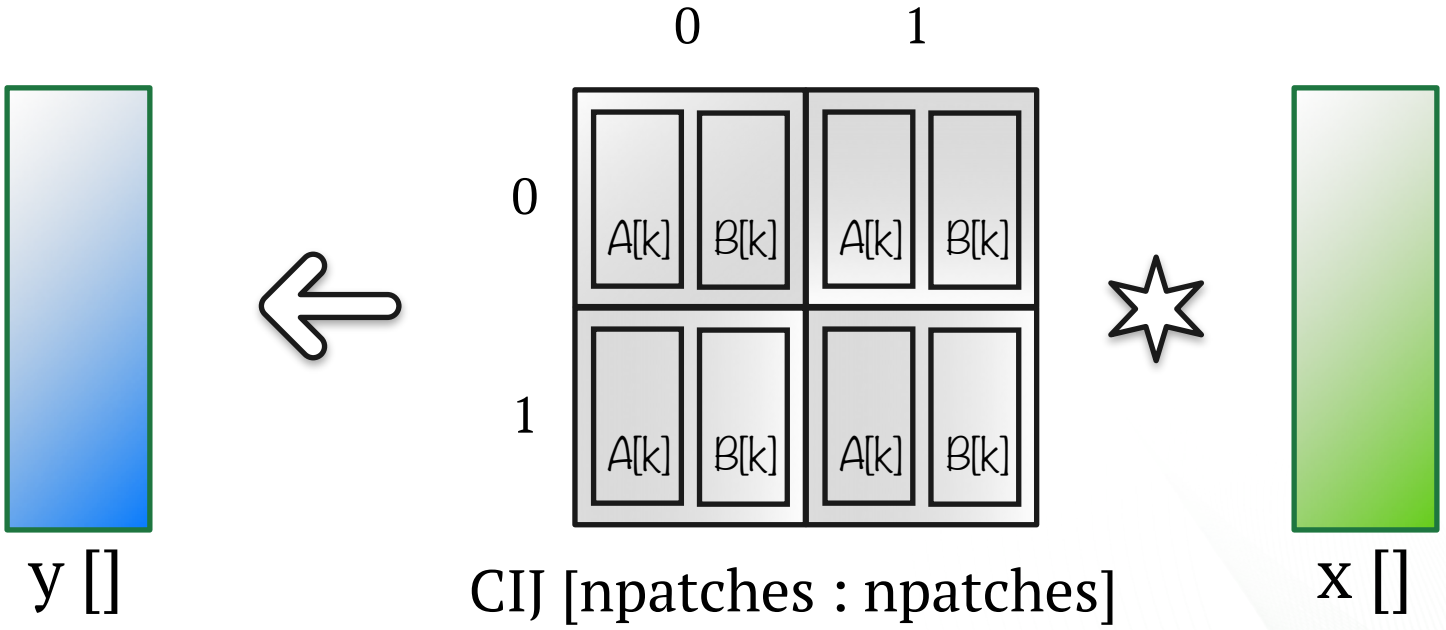
CIJ [npatches : npatches]

x []

Hamiltonian Matrix

OAK RIDGE National Laboratory

KRONECKER PRODUCT (y = H*x)



Hamiltonian Matrix

OAK RIDGE National Laboratory

PSEUDO CODE — (y = H*x) : Sequential

OAK RIDGE National Laboratory

PSEUDO CODE — (y = H*x) : Sequential

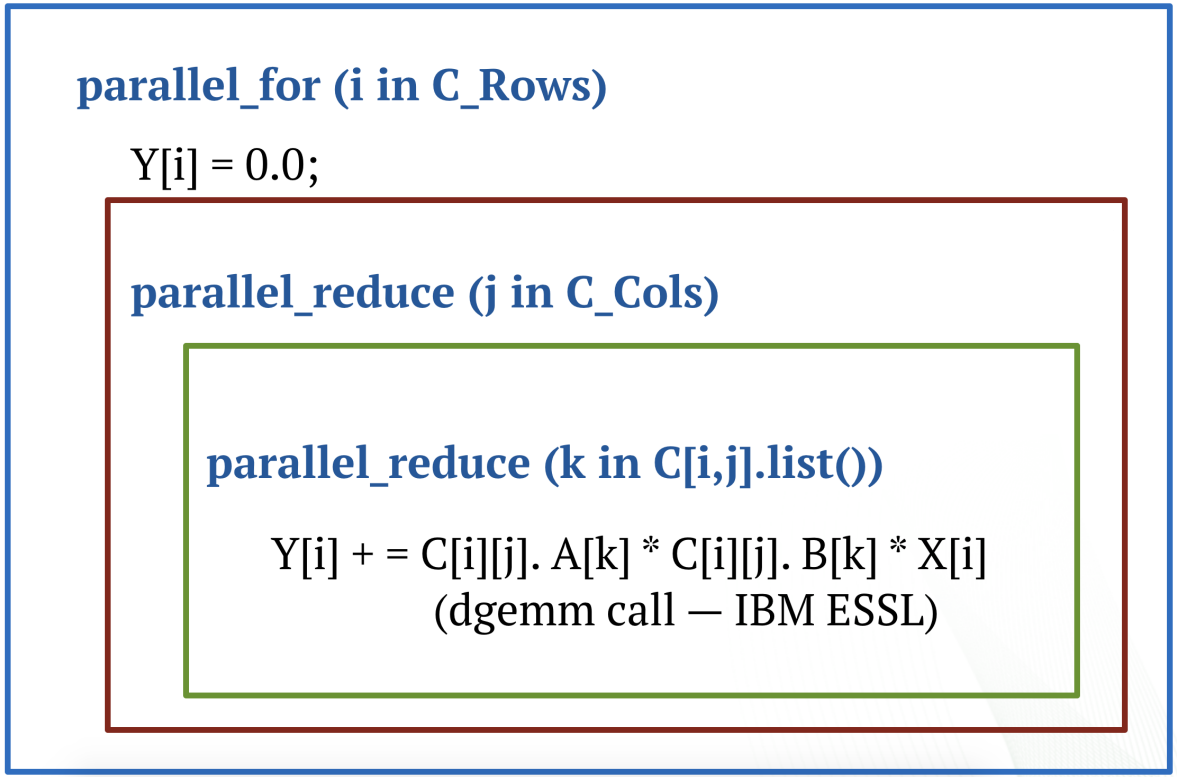
```
for ( i in C_Rows )
    Y[i] = 0.0;
    for ( j in C_Cols )
        for ( k in C [i][j].list() )
            Y[i] + = C[i][j]. A[k] * C[i][j]. B[k] * X[i]
                    (dgemm call — IBM ESSL)
```


OAK RIDGE National Laboratory

PSEUDO CODE — (y = H*x) : Parallel

OAK RIDGE National Laboratory

PSEUDO CODE — (y = H*x) : Parallel



OAK RIDGE National Laboratory

PROGRAMMING STYLES

OAK RIDGE National Laboratory

PROGRAMMING STYLES

- Nested Parallelism

OAK RIDGE National Laboratory

PROGRAMMING STYLES

- Nested Parallelism
- Multi-Level Tasking

OAK RIDGE National Laboratory

PROGRAMMING STYLES

- Nested Parallelism
- Multi-Level Tasking
- Multi-Level Tasking with Nested Parallelism

OAK RIDGE National Laboratory

1. Nested Parallelism

OAK RIDGE National Laboratory

1. Nested Parallelism

1: **INPUT:** C Matrix, X Vector
2: **OUTPUT:** Y Vector
3: Enumerate over the entire C Matrix
4: **omp parallel for:** i in C_Rows
5: Reduction Variable: YI[i]= 0.0
6: **omp parallel for w/ reduction(YI):** j in C_Cols
7: Reduction Variable: YIJ[i]= 0.0
8: **omp parallel for w/ reduction(YIJ):** k in C[i,j].list()
9: **Update YIJ:** += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
10: **Update YI:** YI += YIJ
11: **Update Y:** Y = YI
12: Return vector Y

OAK RIDGE National Laboratory

1. Nested Parallelism

1: **INPUT:** C Matrix, X Vector
2: **OUTPUT:** Y Vector
3: Enumerate over the entire C Matrix
4: **omp parallel for:** i in C_Rows
5: Reduction Variable: YI[i]= 0.0
6: **omp parallel for w/ reduction(YI):** j in C_Cols
7: Reduction Variable: YIJ[i]= 0.0
8: **omp parallel for w/ reduction(YIJ):** k in C[i,j].list()
9: **Update YIJ:** += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
10: **Update YI:** YI += YIJ
11: **Update Y:** Y = YI
12: Return vector Y

OAK RIDGE National Laboratory

1. Nested Parallelism

1: **INPUT:** C Matrix, X Vector
2: **OUTPUT:** Y Vector
3: Enumerate over the entire C Matrix
4: **omp parallel for:** i in C_Rows
5: Reduction Variable: YI[i]= 0.0
6: **omp parallel for w/ reduction(YI):** j in C_Cols
7: Reduction Variable: YIJ[i]= 0.0
8: **omp parallel for w/ reduction(YIJ):** k in C[i,j].list()
9: **Update YIJ:** += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
10: **Update YI:** YI += YIJ
11: **Update Y:** Y = YI
12: Return vector Y

User-defined Reductions

OAK RIDGE National Laboratory

1. Nested Parallelism : Challenges

OAK RIDGE National Laboratory

1. Nested Parallelism : Challenges

- Lack of support for nested work-sharing loops (Spec. Restriction)

OAK RIDGE National Laboratory

1. Nested Parallelism : Challenges

- Lack of support for nested work-sharing loops (Spec. Restriction)
- Creating / destroying parallel regions — affects scalability

OAK RIDGE National Laboratory

1. Nested Parallelism : Challenges

- Lack of support for nested work-sharing loops (Spec. Restriction)
- Creating / destroying parallel regions — affects scalability
- Lack of debugging support within work-sharing loops
  - ✤ omp single / *omp master / omp critical*

OAK RIDGE National Laboratory

1. Nested Parallelism : Challenges

- Lack of support for nested work-sharing loops (Spec. Restriction)
- Creating / destroying parallel regions — affects scalability
- Lack of debugging support within work-sharing loops
  - ✤ omp single / *omp master / omp critical*
- Thread affinity (using OpenMP Places / proc-bind) — depth 3

OAK RIDGE National Laboratory

2. Multi-level Tasking : using taskloop

OAK RIDGE National Laboratory

2. Multi-level Tasking : using taskloop

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  Create OpenMP Parallel region
5:      omp taskloop: i in C_Rows with grain-size
6:          Reduction Variable: YI[i]= 0.0
7:          omp taskloop: j in C_Cols with grain-size
8:              Reduction Variable: YIJ[i]= 0.0
9:              omp taskloop: in C[i,j].list() with grain-size
10:                 Update YIJ: YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
11:             Update YI: YI += YIJ
12:         Update Y: Y = YI
13: Return vector Y
```


OAK RIDGE National Laboratory

2. Multi-level Tasking : using taskloop

1: **INPUT:** C Matrix, X Vector
2: **OUTPUT:** Y Vector
3: Enumerate over the entire C Matrix
4: **Create OpenMP Parallel region**
5: **omp taskloop:** i in C_Rows with grain-size
6: **Reduction Variable:** YI[i]= 0.0
7: **omp taskloop:** j in C_Cols with grain-size
8: **Reduction Variable:** YIJ[i]= 0.0
9: **omp taskloop:** in C[i,j].list() with grain-size
10: **Update YIJ:** YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
11: **Update YI:** YI += YIJ
12: **Update Y:** Y = YI
13: Return vector Y

OAK RIDGE National Laboratory

2. Multi-level Tasking : using taskloop

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  Create OpenMP Parallel region
5:      omp taskloop: i in C_Rows with grain-size
6:          Reduction Variable: YI[i]= 0.0
7:          omp taskloop: j in C_Cols with grain-size
8:              Reduction Variable: YIJ[i]= 0.0
9:              omp taskloop: in C[i,j].list() with grain-size
10:                 Update YIJ: YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
11:             Update YI: YI += YIJ
12:         Update Y: Y = YI
13: Return vector Y
```

User-defined Reductions

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

- Nested task-loops — behavior is implementation specific

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

- Nested task-loops — behavior is implementation specific

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

- Nested task-loops — behavior is implementation specific

- Lack of support for task-level reductions

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

- Nested task-loops — behavior is implementation specific
- Lack of support for task-level reductions
- No support for OpenMP constructs within task-loops
  - ✤ omp single / *omp master* / *omp critical*

OAK RIDGE National Laboratory

2. Multi-level Tasking : Challenges

- Nested task-loops — behavior is implementation specific
- Lack of support for task-level reductions
- No support for OpenMP constructs within task-loops
  - ✤ omp single / *omp master* / *omp critical*
- Task-elasticity — no support for dynamic resource allocation
  - grain-size — thread to task-mapping

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  OpenMP Parallel region
5:      for i in C_Rows
6:          Create OpenMP Tasks: adjust granularity
7:          OpenMP Parallel region w/ Reduction (YI)
8:              Reduction Variable: YI[i]= 0.0
9:              for j in C_Cols
10:                 Create OpenMP Tasks: adjust granularity
11:                 OpenMP Parallel region w/ Reduction (YIJ)
12:                     Reduction Variable: YIJ[i]= 0.0
13:                     for k in C[i,j].list()
14:                         YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
15:                     Reducing YIJ in Parallel region
16:             Reducing YI in parallel region: YI += YIJ
17:         Update Y: Y = YI
18: Return vector Y
```

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  OpenMP Parallel region
5:      for i in C_Rows
6:          Create OpenMP Tasks: adjust granularity
7:          OpenMP Parallel region w/ Reduction (YI)
8:              Reduction Variable: YI[i]= 0.0
9:              for j in C_Cols
10:                 Create OpenMP Tasks: adjust granularity
11:                 OpenMP Parallel region w/ Reduction (YIJ)
12:                     Reduction Variable: YIJ[i]= 0.0
13:                     for k in C[i,j].list()
14:                         YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
15:                     Reducing YIJ in Parallel region
16:             Reducing YI in parallel region: YI += YIJ
17:         Update Y: Y = YI
18: Return vector Y
```

User-defined Reductions

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  OpenMP Parallel region
5:      for i in C_Rows
6:          Create OpenMP Tasks: adjust granularity
7:          OpenMP Parallel region w/ Reduction (YI)
8:              Reduction Variable: YI[i]= 0.0
9:              for j in C_Cols
10:                 Create OpenMP Tasks: adjust granularity
11:                 OpenMP Parallel region w/ Reduction (YIJ)
12:                     Reduction Variable: YIJ[i]= 0.0
13:                     for k in C[i,j].list()
14:                         YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
15:                     Reducing YIJ in Parallel region
16:             Reducing YI in parallel region: YI += YIJ
17:         Update Y: Y = YI
18: Return vector Y
```

User-defined Reductions

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  OpenMP Parallel region
5:      for i in C_Rows
6:          Create OpenMP Tasks: adjust granularity
7:          OpenMP Parallel region w/ Reduction (YI)
8:              Reduction Variable: YI[i]= 0.0
9:              for j in C_Cols
10:                 Create OpenMP Tasks: adjust granularity
11:                 OpenMP Parallel region w/ Reduction (YIJ)
12:                     Reduction Variable: YIJ[i]= 0.0
13:                     for k in C[i,j].list()
14:                         YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
15:                     Reducing YIJ in Parallel region
16:             Reducing YI in parallel region: YI += YIJ
17:         Update Y: Y = YI
18: Return vector Y
```

User-defined Reductions

OAK RIDGE National Laboratory

3. Multi-level Tasking with Nested Parallelism

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Enumerate over the entire C Matrix
4:  OpenMP Parallel region
5:      for i in C_Rows
6:          Create OpenMP Tasks: adjust granularity
7:          OpenMP Parallel region w/ Reduction (YI)
8:              Reduction Variable: YI[i]= 0.0
9:              for j in C_Cols
10:                 Create OpenMP Tasks: adjust granularity
11:                 OpenMP Parallel region w/ Reduction (YIJ)
12:                     Reduction Variable: YIJ[i]= 0.0
13:                     for k in C[i,j].list()
14:                         YIJ += C[i][j].A[k] ⊗ C[i][j].B[k] * X[]
15:                     Reducing YIJ in Parallel region
16:             Reducing YI in parallel region: YI += YIJ
17:         Update Y: Y = YI
18: Return vector Y
```

User-defined Reductions

OAK RIDGE National Laboratory

3. Tasking with Nested Parallelism: Challenges

OAK RIDGE National Laboratory

3. Tasking with Nested Parallelism: Challenges

- Creating / destroying parallel regions — affects scalability

OAK RIDGE National Laboratory

3. Tasking with Nested Parallelism: Challenges

- Creating / destroying parallel regions — affects scalability

- Task-loops — variable are *first private* and **NOT** *thread private*
  - ✤ reduction becomes complicated

OAK RIDGE National Laboratory

3. Tasking with Nested Parallelism: Challenges

- Creating / destroying parallel regions — affects scalability

- Task-loops — variable are *first private* and **NOT** *thread private*
  - ✤ reduction becomes complicated

OAK RIDGE National Laboratory

3. Tasking with Nested Parallelism: Challenges

- Creating / destroying parallel regions — affects scalability

- Task-loops — variable are *first private* and **NOT** *thread private*
  - ✤ reduction becomes complicated

- Task-affinity — within OpenMP nested parallel region
  - ✤ extensions to *tied-tasks* (depth 3)

OAK RIDGE National Laboratory

EXPERIMENTAL SETUP

OAK RIDGE National Laboratory

EXPERIMENTAL SETUP

- Compilers used: **GCC**, XLC++, CLANG++

OAK RIDGE National Laboratory

EXPERIMENTAL SETUP

- Compilers used: **GCC**, XLC++, CLANG++
- Evaluated on Summit-dev (OLCF):
  - **2, 10-core IBM Power8 CPUs (8 h/w threads per core)**
  - 4 NVIDIA Tesla P100 GPUs

OAK RIDGE National Laboratory

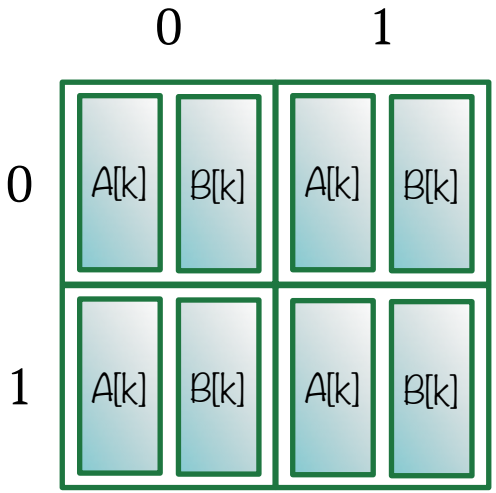
EXPERIMENTAL SETUP

- Compilers used: **GCC**, XLC++, CLANG++
- Evaluated on Summit-dev (OLCF):
  - **2, 10-core IBM Power8 CPUs (8 h/w threads per core)**
  - 4 NVIDIA Tesla P100 GPUs
- DGEMM call: IBM ESSL (and ESSL-smp)

EXPERIMENTAL DATASET

OAK RIDGE National Laboratory

EXPERIMENTAL DATASET

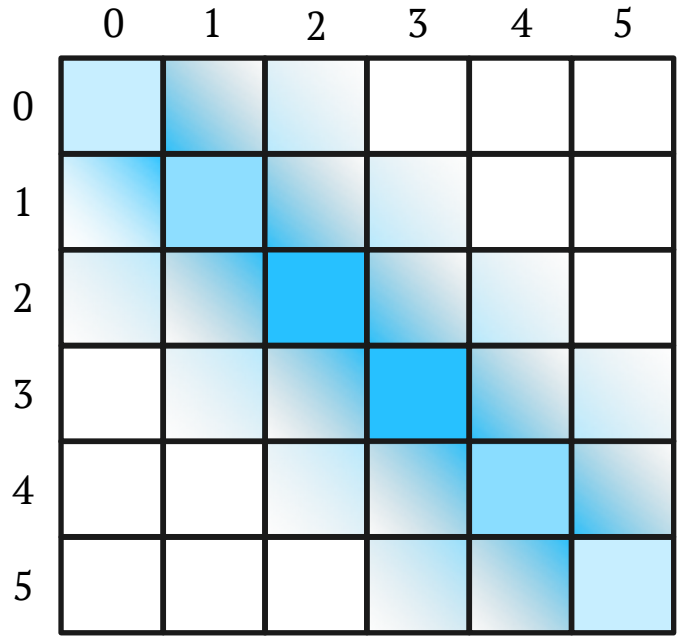


CIJ [npatches : npatches]

Hamiltonian Matrix

OAK RIDGE National Laboratory

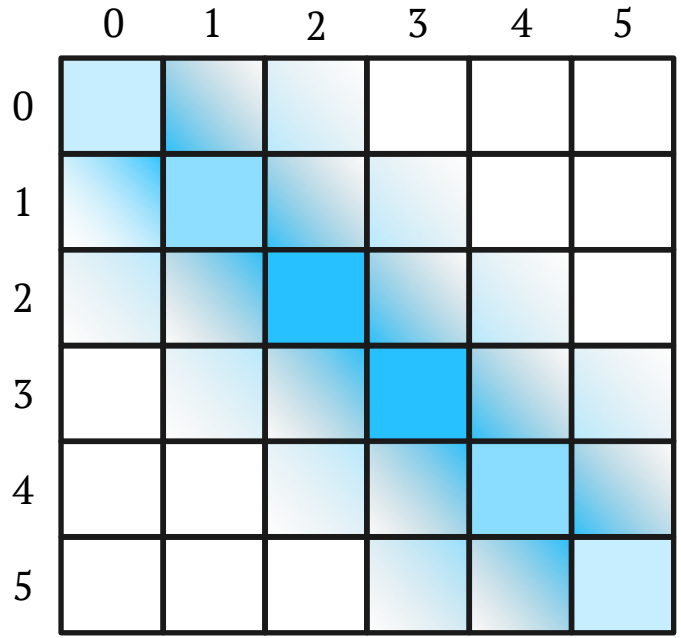
EXPERIMENTAL DATASET



Sparsity of the Hamiltonian Matrix

OAK RIDGE National Laboratory

EXPERIMENTAL DATASET

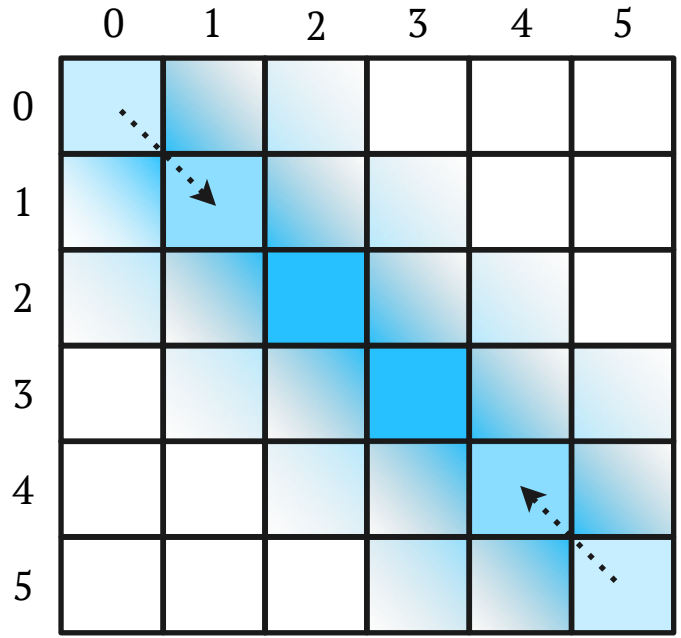


Sparsity of the Hamiltonian Matrix

- Data (A[k]'s and B[k]'s)
  → mostly principal diagonal

OAK RIDGE National Laboratory

EXPERIMENTAL DATASET

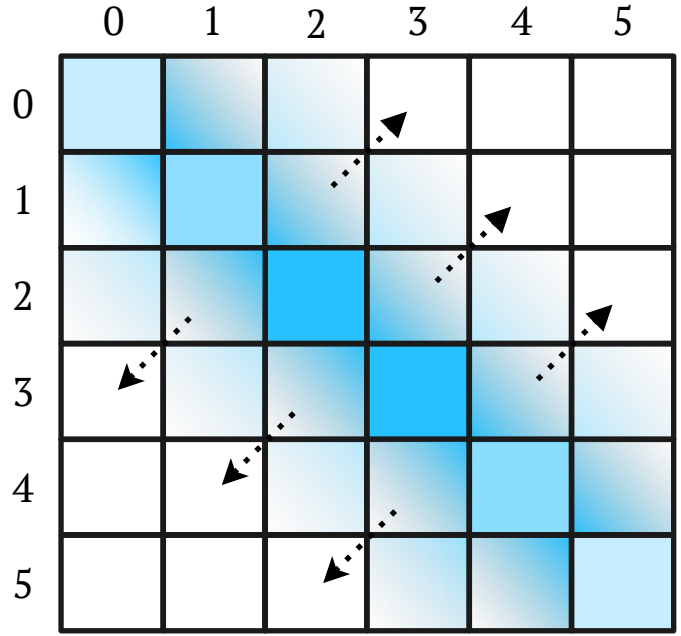


Sparsity of the Hamiltonian Matrix

- Data (A[k]'s and B[k]'s)
  → mostly principal diagonal
- Density increases
  → towards the center

OAK RIDGE National Laboratory

EXPERIMENTAL DATASET

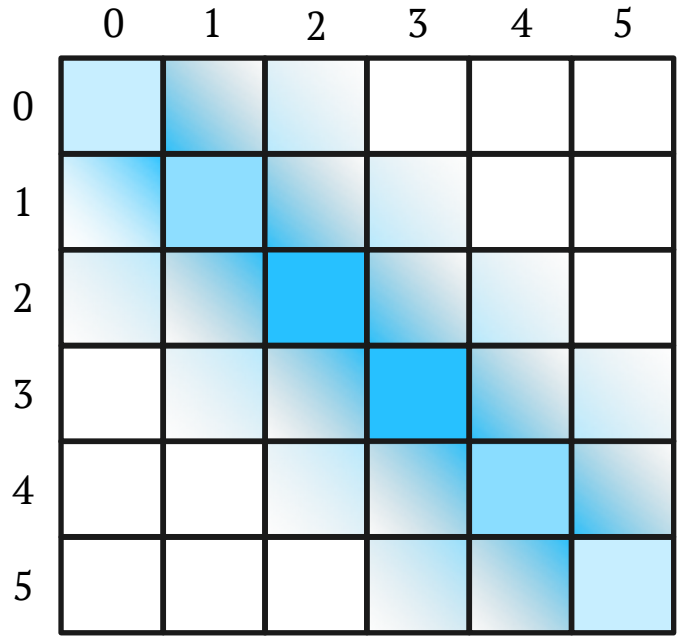


Sparsity of the Hamiltonian Matrix

- Data (A[k]'s and B[k]'s)
  → mostly principal diagonal
- Density increases
  → towards the center
- Sparsity increases
  → away from the principal diagonal

OAK RIDGE National Laboratory

EXPERIMENTAL DATASET



Sparsity of the Hamiltonian Matrix

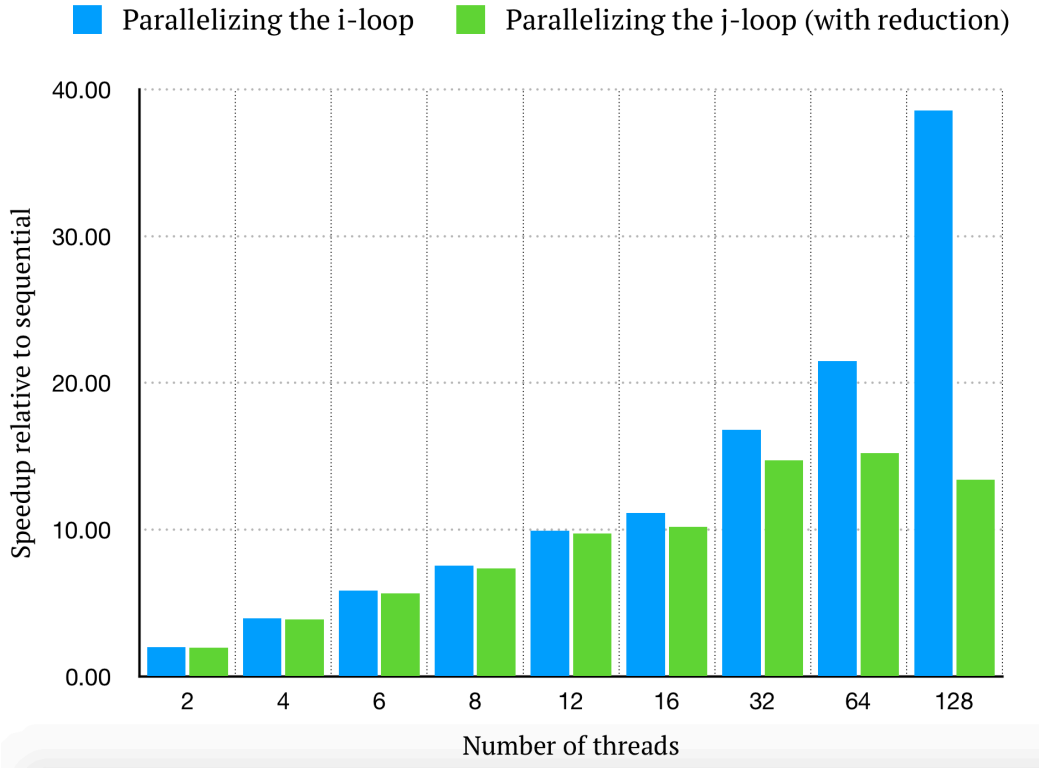
- Data (A[k]'s and B[k]'s)
  → mostly principal diagonal
- Density increases
  → towards the center
- Sparsity increases
  → away from the principal diagonal

OAK RIDGE National Laboratory

EXPERIMENTAL EVALUATION

OAK RIDGE National Laboratory

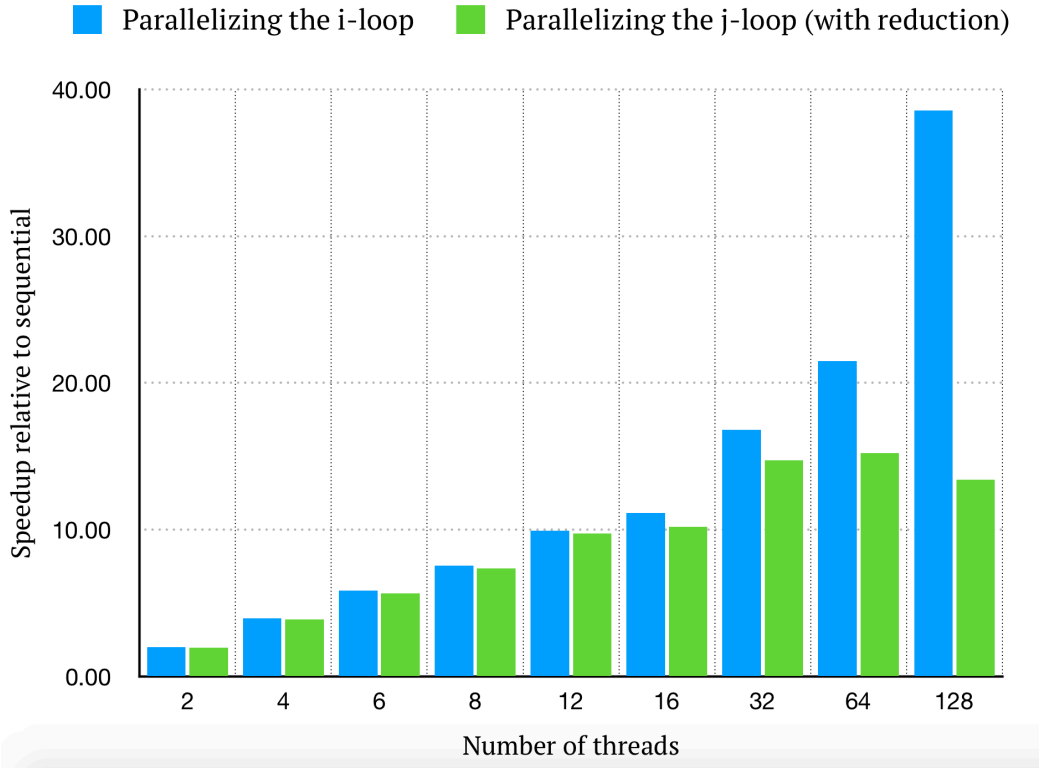
EXPERIMENTAL EVALUATION



- Each loop is parallelized separately

OAK RIDGE National Laboratory

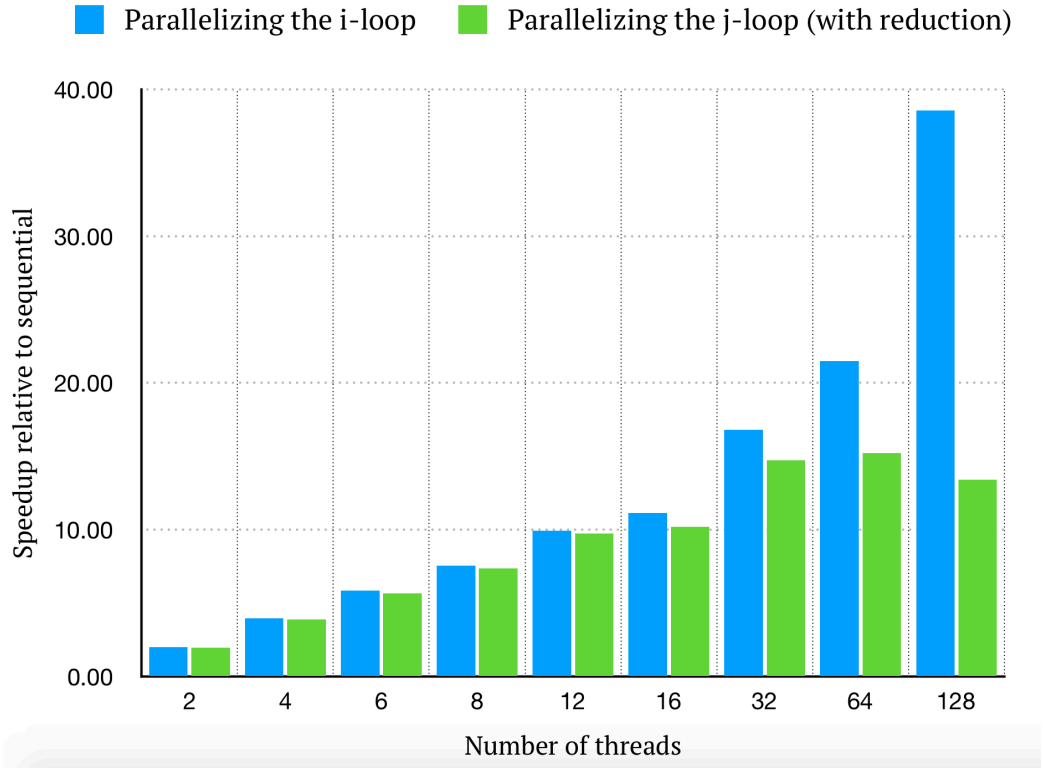
EXPERIMENTAL EVALUATION



- Each loop is parallelized separately
- Inner loop (jPatch) performs *vector reduction*

OAK RIDGE National Laboratory

EXPERIMENTAL EVALUATION



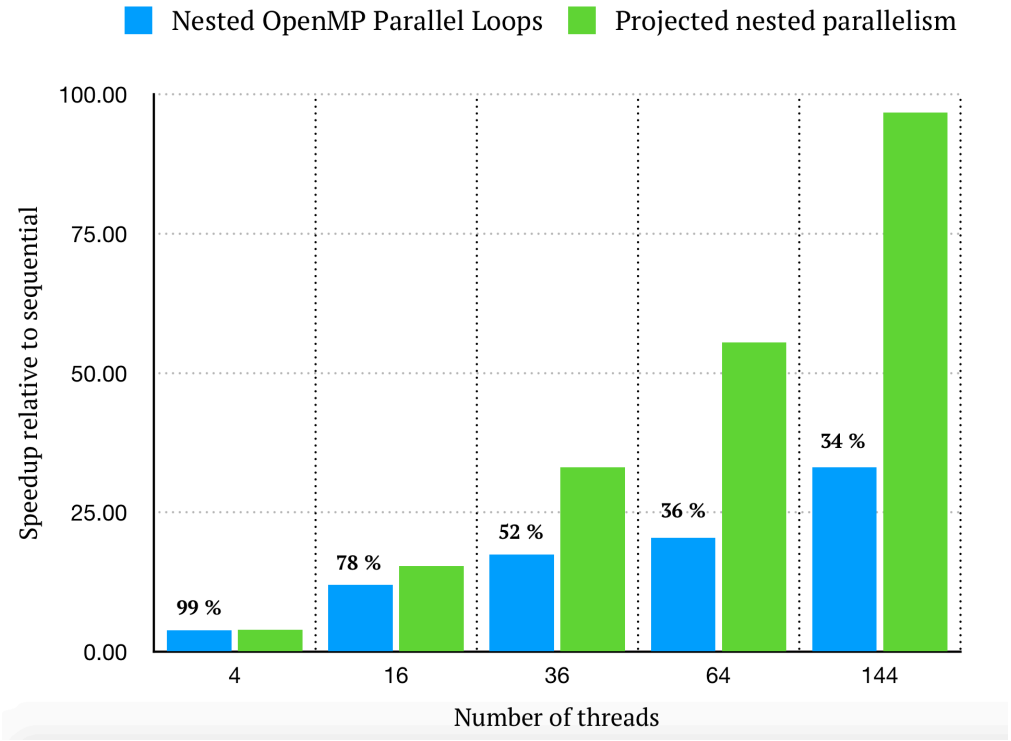
- Each loop is parallelized separately
- Inner loop (jPatch) performs *vector reduction*
- Parallelizing k-loop → *threaded DGEMM (IBM ESSL)*

OAK RIDGE National Laboratory

EXPERIMENTAL EVALUATION: Projection

OAK RIDGE National Laboratory

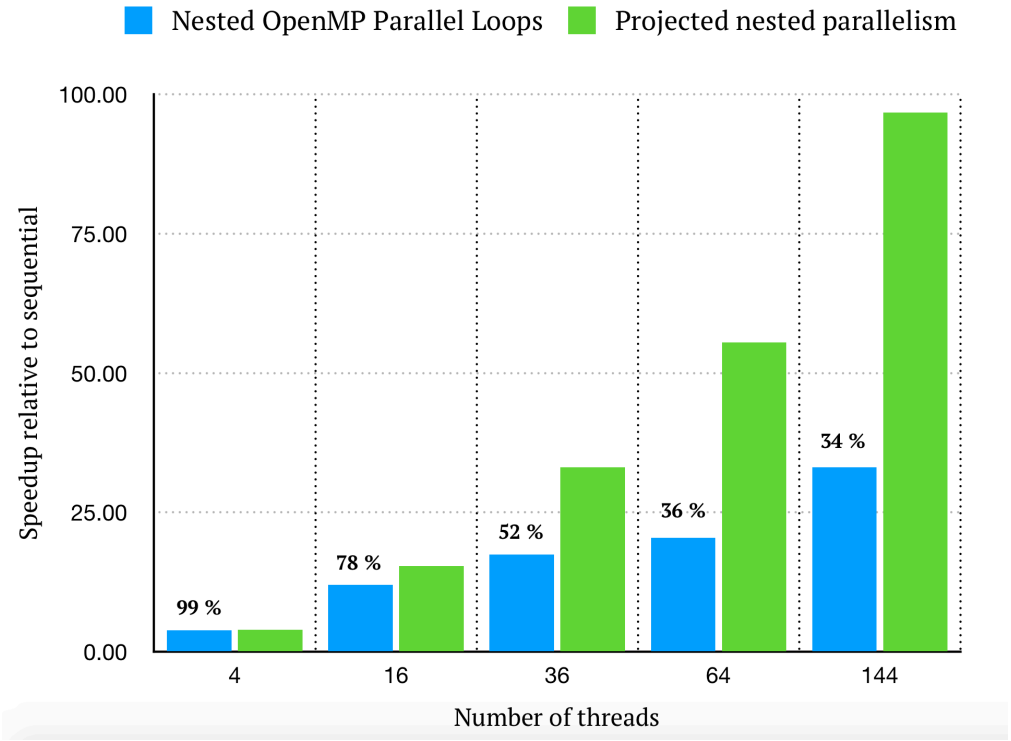
EXPERIMENTAL EVALUATION: Projection



- Nested parallel-for's
  → iPatch and jPatch
- Processor bindings
  - ✤ *Outer*: Spread
  - ✤ *Inner*: Close

OAK RIDGE National Laboratory

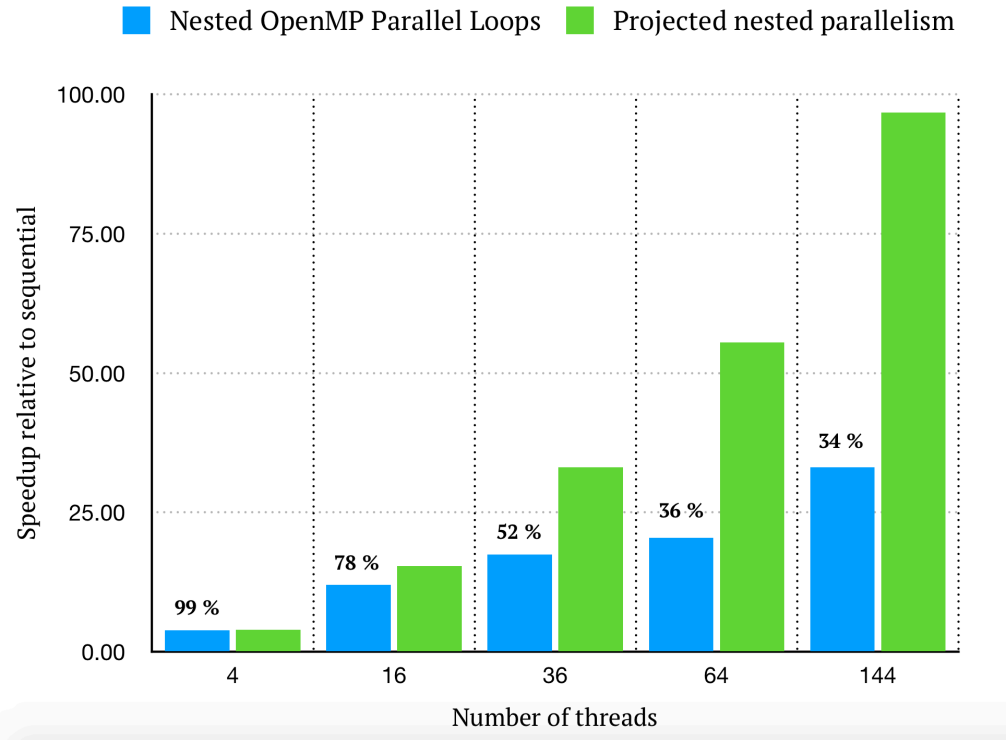
EXPERIMENTAL EVALUATION: Projection



- Nested parallel-for's
  → iPatch and jPatch
- Processor bindings
  - *Outer*: Spread
  - *Inner*: Close
- Dynamic scheduling

OAK RIDGE National Laboratory

EXPERIMENTAL EVALUATION: Projection



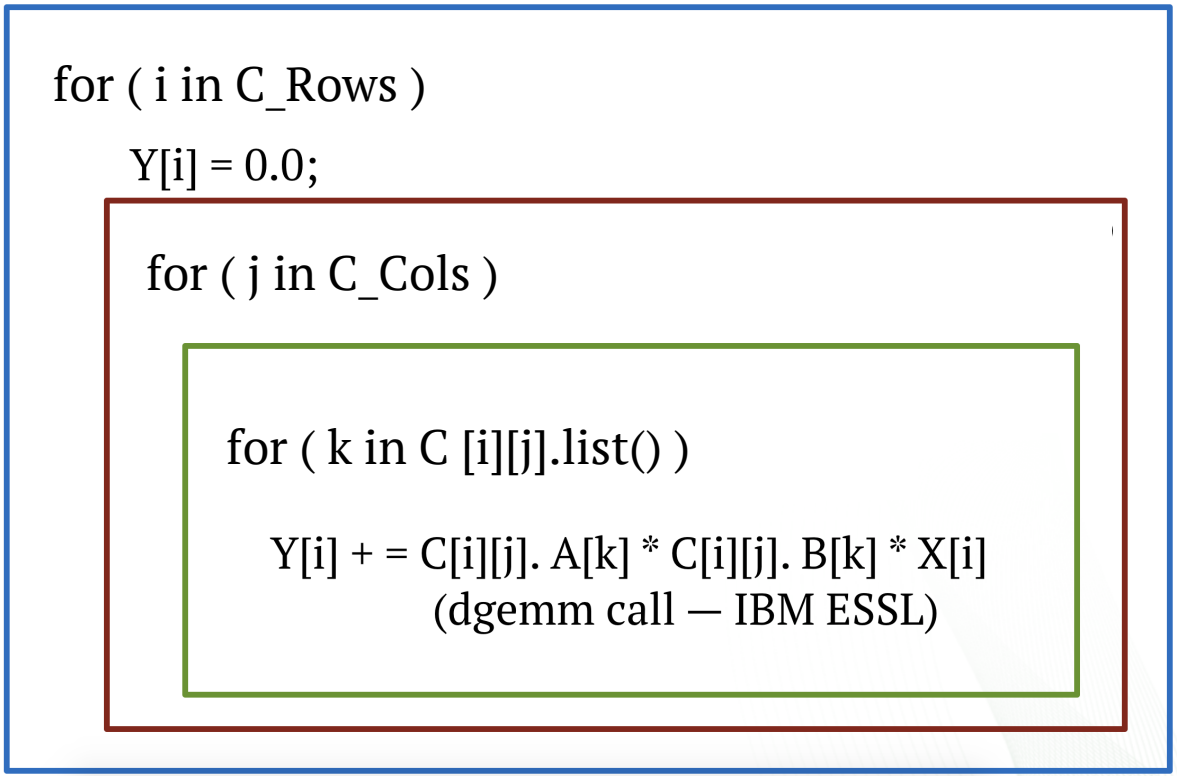
- Nested parallel-for's
  → iPatch and jPatch
- Processor bindings
  - ✤ *Outer*: Spread
  - ✤ *Inner*: Close
- Dynamic scheduling
- Significant overhead
  →creating / destroying || regions

OAK RIDGE National Laboratory

USING OpenMP WITH LIBRARIES

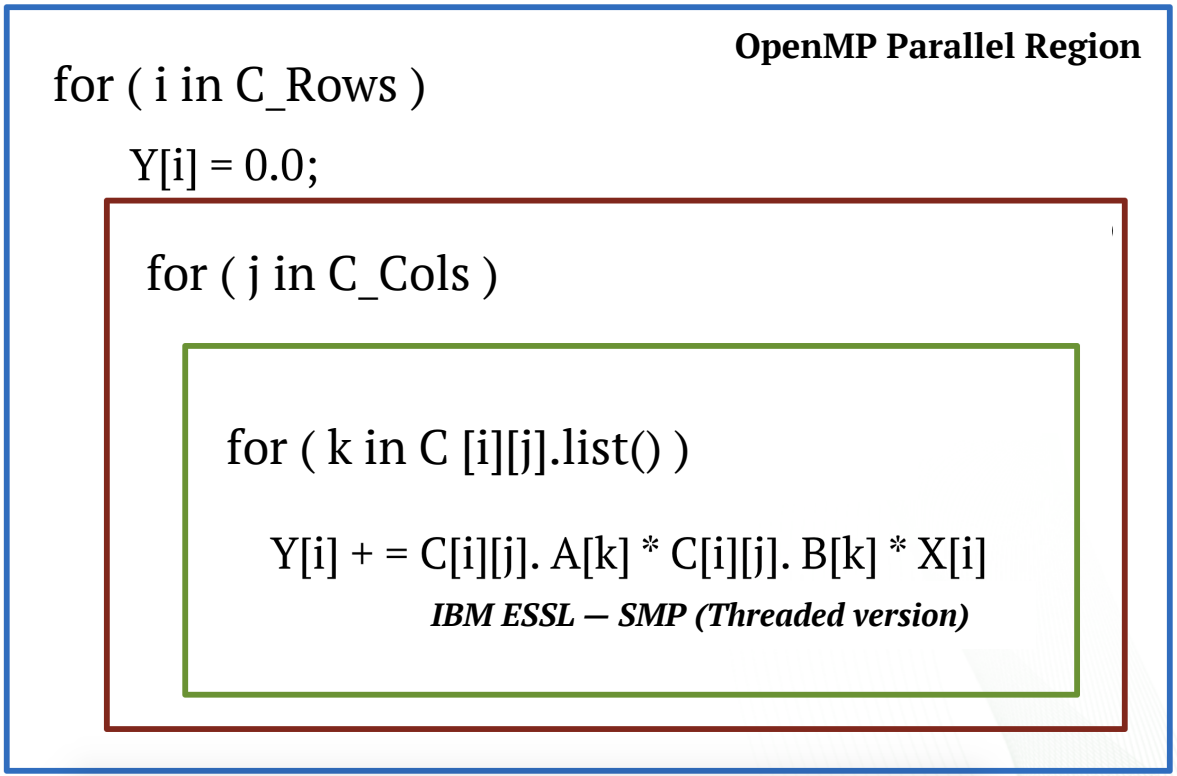
OAK RIDGE National Laboratory

USING OpenMP WITH LIBRARIES



OAK RIDGE National Laboratory

USING OpenMP WITH LIBRARIES



OAK RIDGE National Laboratory

OpenMP WITH LIBRARIES: Challenges

OAK RIDGE National Laboratory

OpenMP WITH LIBRARIES: Challenges

- Current thread support:
  - OMP_SET_NUM_THREADS <num of threads>

OAK RIDGE National Laboratory

OpenMP WITH LIBRARIES: Challenges

- Current thread support:
  - OMP_SET_NUM_THREADS <num of threads>

- Lack of support for dynamic thread assignment

OAK RIDGE National Laboratory

OpenMP WITH LIBRARIES: Challenges

- Current thread support:
  - OMP_SET_NUM_THREADS <num of threads>

- Lack of support for dynamic thread assignment

OAK RIDGE National Laboratory

OpenMP WITH LIBRARIES: Challenges

- Current thread support:
  - OMP_SET_NUM_THREADS <num of threads>
- Lack of support for dynamic thread assignment
- Interoperability with external libraries:
  - Support to extract task-level/thread-level information
  - Within nested parallel region — undefined behavior

HABANERO C/C++ LIBRARY (HCLib)

HabaneroUPC++: a Compiler-free PGAS Library. *V. Kumar, Y. Zheng, V. Cavé, Z. Budimlić, and V. Sarkar, PGAS 2014*

OAK RIDGE National Laboratory

HABANERO C/C++ LIBRARY (HCLib)

- Developed at Rice University as part of the Habanero Extreme Scale Software Research Project

HabaneroUPC++: a Compiler-free PGAS Library. *V. Kumar, Y. Zheng, V. Cavé, Z. Budimlić, and V. Sarkar, PGAS 2014*

HABANERO C/C++ LIBRARY (HCLib)

- Developed at Rice University as part of the Habanero Extreme Scale Software Research Project
- HCLib — Task-based parallel programming model

HabaneroUPC++: a Compiler-free PGAS Library. *V. Kumar, Y. Zheng, V. Cavé, Z. Budimlić, and V. Sarkar, PGAS 2014*

HABANERO C/C++ LIBRARY (HCLib)

- Developed at Rice University as part of the Habanero Extreme Scale Software Research Project
- HCLib — Task-based parallel programming model
- HCLib runtime — light-weight, work-stealing & locality-aware

HabaneroUPC++: a Compiler-free PGAS Library. *V. Kumar, Y. Zheng, V. Cavé, Z. Budimlić, and V. Sarkar, PGAS 2014*

HABANERO C/C++ LIBRARY (HCLib)

- Developed at Rice University as part of the Habanero Extreme Scale Software Research Project
- HCLib — Task-based parallel programming model
- HCLib runtime — light-weight, work-stealing & locality-aware
- HCLib — path to Exascale programming system
  → intra-node: resource management & scheduling
  → inter-node: integration w/ communication models (MPI, UPC++ or OpenSHMEM)

HabaneroUPC++: a Compiler-free PGAS Library. *V. Kumar, Y. Zheng, V. Cavé, Z. Budimlić, and V. Sarkar, PGAS 2014*

FINISH ACCUMULATORS

Finish Accumulators: a Deterministic Reduction Construct for Dynamic Task Parallelism*. J. Shirako, V. Cavé, J. Zhao, & V. Sarkar. WODET 2013*

OAK RIDGE National Laboratory

FINISH ACCUMULATORS

- Construct for pre-defined / user-defined reductions

Finish Accumulators: a Deterministic Reduction Construct for Dynamic Task Parallelism. *J. Shirako, V. Cavé, J. Zhao, & V. Sarkar. WODET 2013*

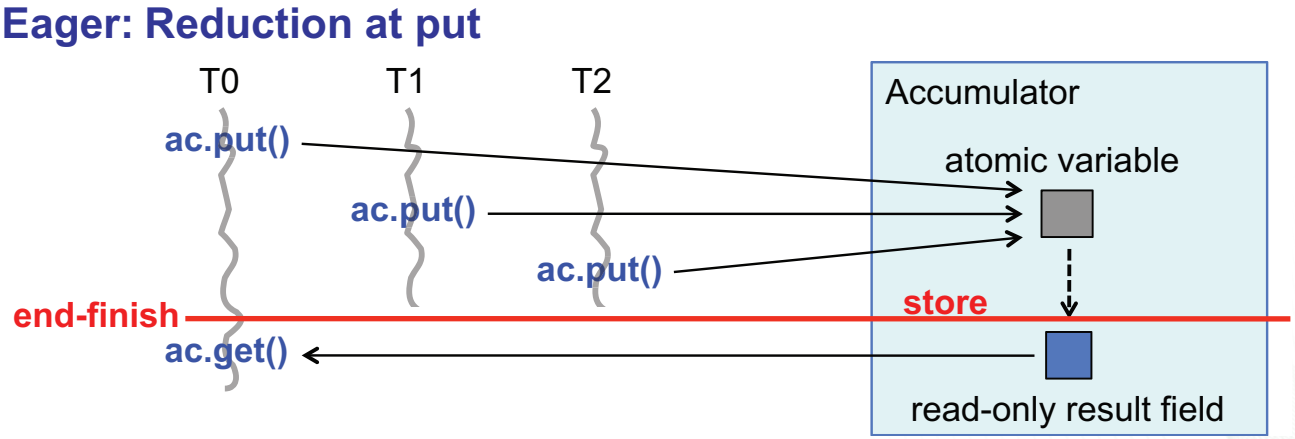
FINISH ACCUMULATORS

- Construct for pre-defined / user-defined reductions
- Large flexibility for implementations:

Finish Accumulators: a Deterministic Reduction Construct for Dynamic Task Parallelism. *J. Shirako, V. Cavé, J. Zhao, & V. Sarkar. WODET 2013*

FINISH ACCUMULATORS

- Construct for pre-defined / user-defined reductions
- Large flexibility for implementations:
  - Eager reduction policy — portable implementation

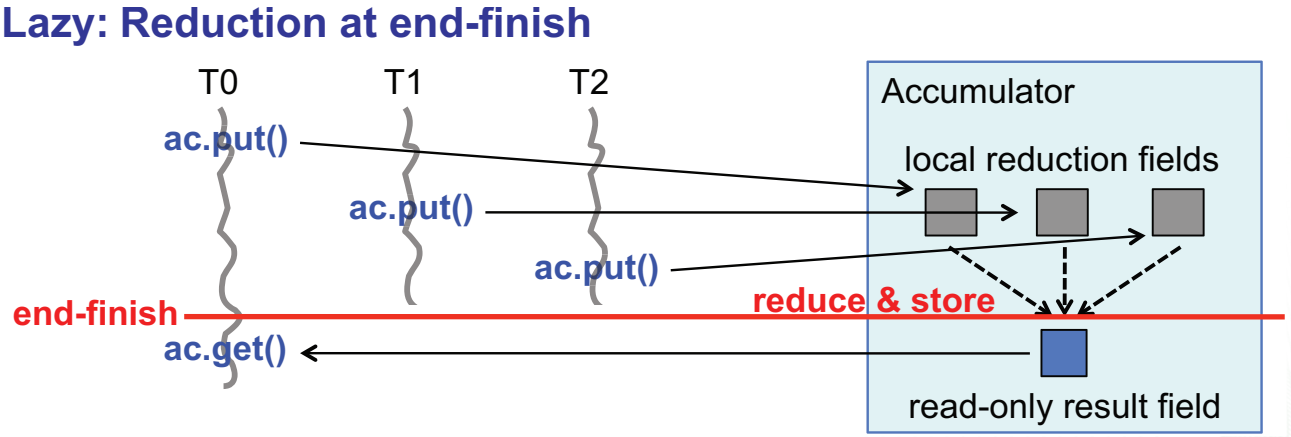


Finish Accumulators: a Deterministic Reduction Construct for Dynamic Task Parallelism. *J. Shirako, V. Cavé, J. Zhao, & V. Sarkar. WODET 2013*

OAK RIDGE National Laboratory

FINISH ACCUMULATORS

- Construct for pre-defined / user-defined reductions
- Large flexibility for implementations:
  - Lazy reduction policy — customized for target runtime task scheduler



Finish Accumulators: a Deterministic Reduction Construct for Dynamic Task Parallelism. *J. Shirako, V. Cavé, J. Zhao, & V. Sarkar. WODET 2013*

OAK RIDGE National Laboratory

PSEUDO CODE: HABANERO C++

OAK RIDGE National Laboratory

PSEUDO CODE: HABANERO C++

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Create array of Accumulators: accs <SUM> (npatches)
4:  Create OpenMP Parallel region
5:  Create OpenMP Single region
6:     for i in C_Rows do
7:        Create OpenMP Tasks
8:        for For: j in C_Cols do
9:           Create OpenMP Tasks
10:          YIJ[i]= 0.0
11:          for k in C[i,j].list() do
12:             Update YIJ: YIJ+ = C[i][j].A[k] ⊗ C[i][j].B[k] ∗ X[]

13:             Update Accumulator: accs[ipatch]->put(YIJ)
14:          end for
15:       end for
16:    end for
17: Retrieve Accumulator: accs[...]->get()
18: Return vector Y
```

PSEUDO CODE: HABANERO C++

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Create array of Accumulators: accs <SUM> (npatches)
4:  Create OpenMP Parallel region
5:  Create OpenMP Single region
6:      for i in C_Rows do
7:          Create OpenMP Tasks
8:          for For: j in C_Cols do
9:              Create OpenMP Tasks
10:             YIJ[i]= 0.0
11:             for k in C[i,j].list() do
12:                 Update YIJ: YIJ+ = C[i][j].A[k] ⊗ C[i][j].B[k] ∗ X[]

13:                 Update Accumulator: accs[ipatch]->put(YIJ)
14:             end for
15:         end for
16:     end for
17: Retrieve Accumulator: accs[...]->get()
18: Return vector Y
```

PSEUDO CODE: HABANERO C++

Parallel Region

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Create array of Accumulators: accs <SUM> (npatches)
4:  Create OpenMP Parallel region
5:  Create OpenMP Single region
6:     for i in C_Rows do
7:        Create OpenMP Tasks
8:        for For: j in C_Cols do
9:           Create OpenMP Tasks
10:          YIJ[i]= 0.0
11:          for k in C[i,j].list() do
12:             Update YIJ: YIJ+ = C[i][j].A[k] ⊗ C[i][j].B[k] ∗ X[]
13:             Update Accumulator: accs[ipatch]->put(YIJ)
14:          end for
15:       end for
16:    end for
17: Retrieve Accumulator: accs[...]->get()
18: Return vector Y
```

PSEUDO CODE: HABANERO C++

```
1:  INPUT: C Matrix, X Vector
2:  OUTPUT: Y Vector
3:  Create array of Accumulators: accs <SUM> (npatches)
4:  Create OpenMP Parallel region
5:  Create OpenMP Single region
6:     for i in C_Rows do
7:        Create OpenMP Tasks
8:        for For: j in C_Cols do
9:           Create OpenMP Tasks
10:          YIJ[i]= 0.0
11:          for k in C[i,j].list() do
12:             Update YIJ: YIJ+ = C[i][j].A[k] ⊗ C[i][j].B[k] * X[]

13:             Update Accumulator: accs[ipatch]->put(YIJ)
14:          end for
15:       end for
16:    end for
17: Retrieve Accumulator: accs[...]->get()
18: Return vector Y
```

Parallel Region

User-defined Reduction

Candidates for Future OpenMP Support

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions
- Task-inflation: dynamic resource allocation to tasks

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions
- Task-inflation: dynamic resource allocation to tasks
- Task-affinity: binding tasks to cores (extending tied-tasks)

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions
- Task-inflation: dynamic resource allocation to tasks
- Task-affinity: binding tasks to cores (extending tied-tasks)
- Thread persistence in nested parallelism (Depth 3)

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions
- Task-inflation: dynamic resource allocation to tasks
- Task-affinity: binding tasks to cores (extending tied-tasks)
- Thread persistence in nested parallelism (Depth 3)
- Dynamic OpenMP Places

OAK RIDGE National Laboratory

Candidates for Future OpenMP Support

- Support for Task-level reductions
- Task-inflation: dynamic resource allocation to tasks
- Task-affinity: binding tasks to cores (extending tied-tasks)
- Thread persistence in nested parallelism (Depth 3)
- Dynamic OpenMP Places
- Addressing debugging Challenges

ONGOING WORK: DMRG++

OAK RIDGE National Laboratory

ONGOING WORK: DMRG++

- OpenMP target directives — GPUs on Summit-dev

OAK RIDGE National Laboratory

ONGOING WORK: DMRG++

- OpenMP target directives — GPUs on Summit-dev
- OpenMP parallel regions — create abstractions on CPUs

OAK RIDGE National Laboratory

ONGOING WORK: DMRG++

- OpenMP target directives — GPUs on Summit-dev
- OpenMP parallel regions — create abstractions on CPUs
- Co-designing:
  - Using Kokkos (Sandia National Lab)
  - Using MAGMA — DGEMM batched kernel (Univ. of Tennessee, Knoxville)
  - Habanero C/C++ Accumulators (Georgia Tech. / Rice University)
    - Original implementation of finish accumulators was done in Habanero-Java

ACKNOWLEDGEMENTS

- We would like to thank our collaborators:
  - ORNL: Dr. Wael Elwasif (CSMD)
    Dr. E. D'Azevedo (CNMS)
    Dr. G. Alvarez (CNMS)
  - Rice University : Dr. Jun Shirako

- *This research used resources of the **O**ak Ridge **L**eadership **C**omputing **F**acility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.*

Bringing the DMRG++ Scientific Application to Exascale

...

Arghya “Ronnie” Chatterjee
Research Collaborator, CSMD, ORNL
Ph.D. Student, Georgia Tech
arghya@gatech.edu
September 18th, 2017