Techniques for Enabling Highly Efficient Message Passing on Many-Core Architectures

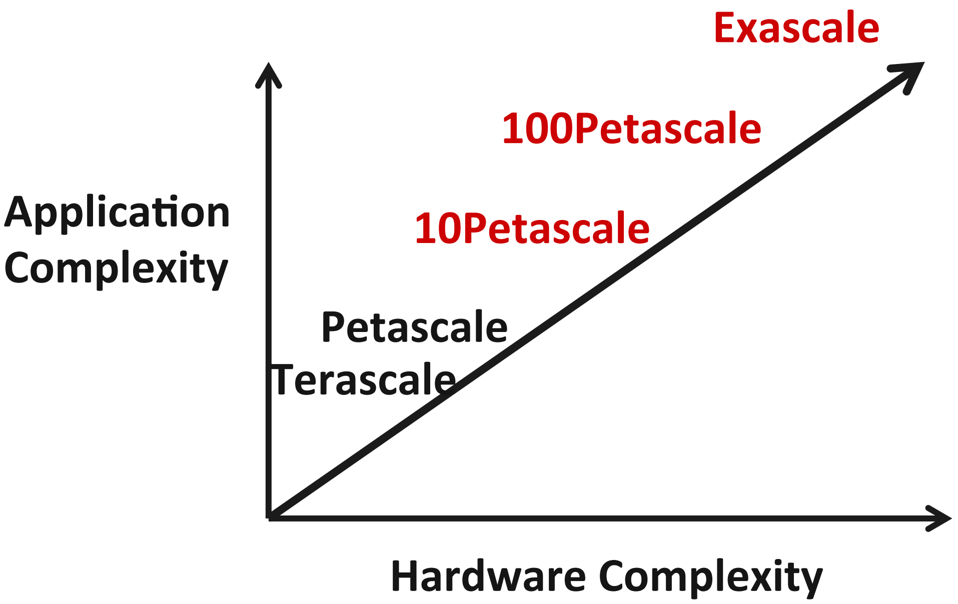
Min Si
PhD student at University of Tokyo, Tokyo, Japan
Advisor : Prof. Yutaka Ishikawa, Prof. Reiji Suda

Guest graduate student at Argonne National Laboratory, IL, USA
Supervisor : Dr. Pavan Balaji

Email: msi@il.is.s.u-tokyo.ac.jp
Homepage: http://sudalab.is.s.u-tokyo.ac.jp/~msi/

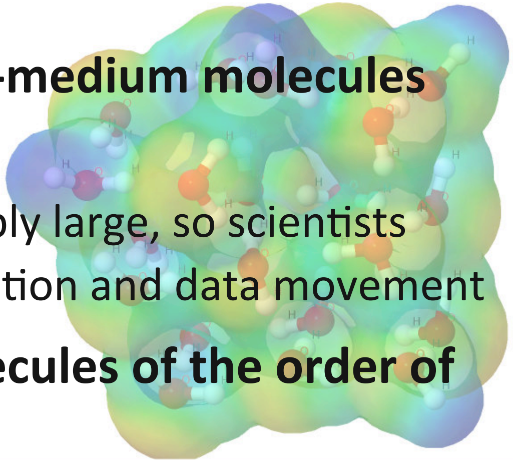
Background

- Complexity in scientific applications
- Trends of hardware change
- Popular programming models and existing challenges

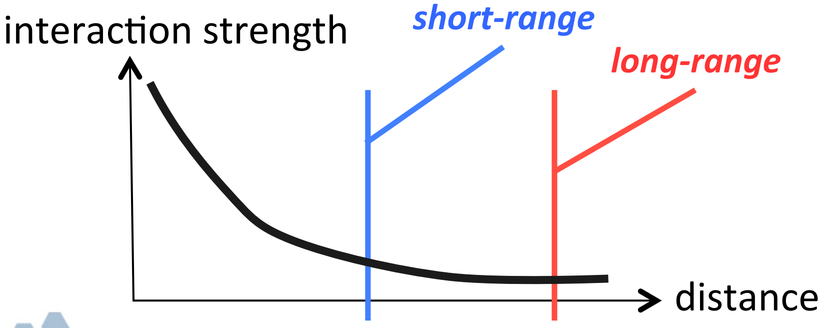


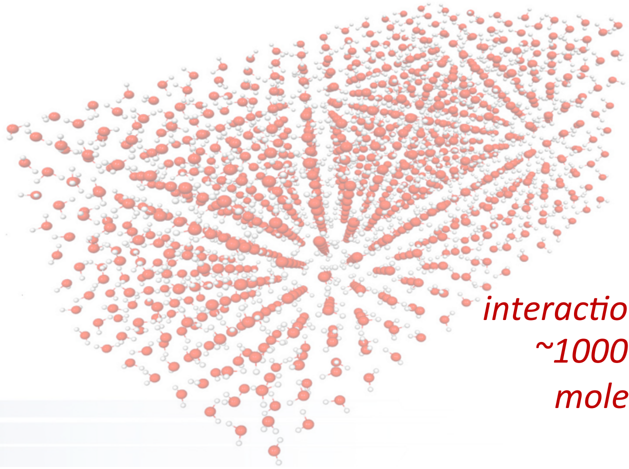
NWChem in Chemistry

interactions among ~21 water molecules



- **Current applications have been looking at small-to-medium molecules consisting of 20-100 atoms**
  - Amount of computation per data element is reasonably large, so scientists have been reasonably successful decoupling computation and data movement
- **For Exascale systems, scientists want to study molecules of the order of a 1000 atoms or larger**
  - Coulomb interactions between the atoms is much stronger in the problems today than what we expect for Exascale-level problems
  - Larger problems will need to support both **short-range** and **long-range** components of the coulomb interactions (possibly using different solvers)



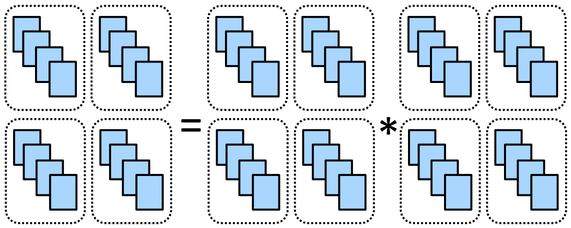


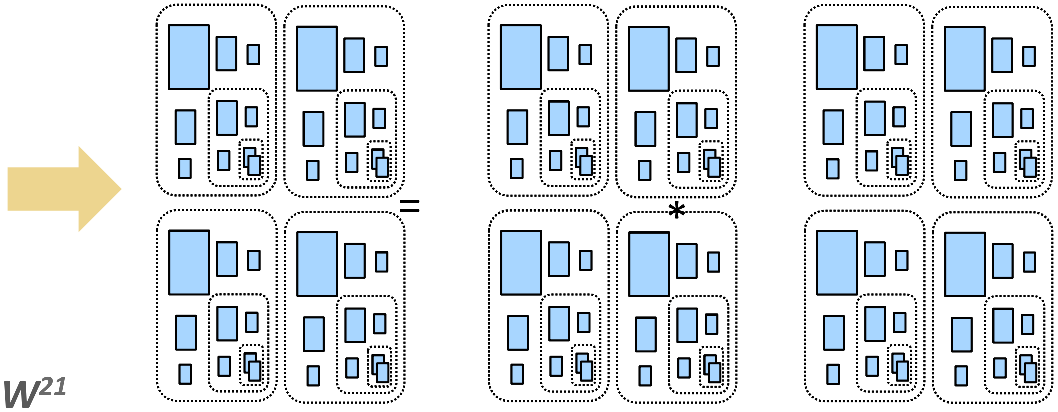
interactions among ~1000 water molecules

Irregular Sparse Computation in NWChem

- Diversity in the amount of computation per data element is going to increase substantially
- Regularity of data and/or computation would be substantially different

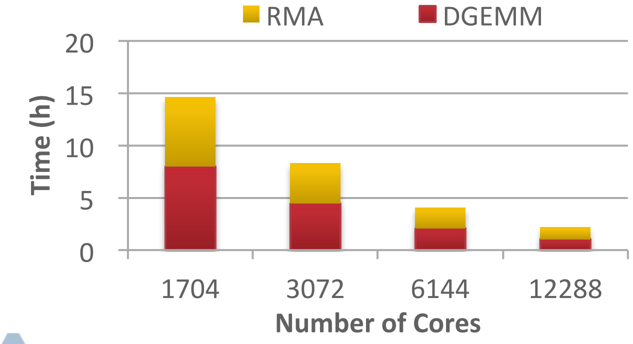
Current computation pattern





Courtesy Pavan Balaji (Argonne)

More than 50% time idling in CCSD(T) for W^{21}

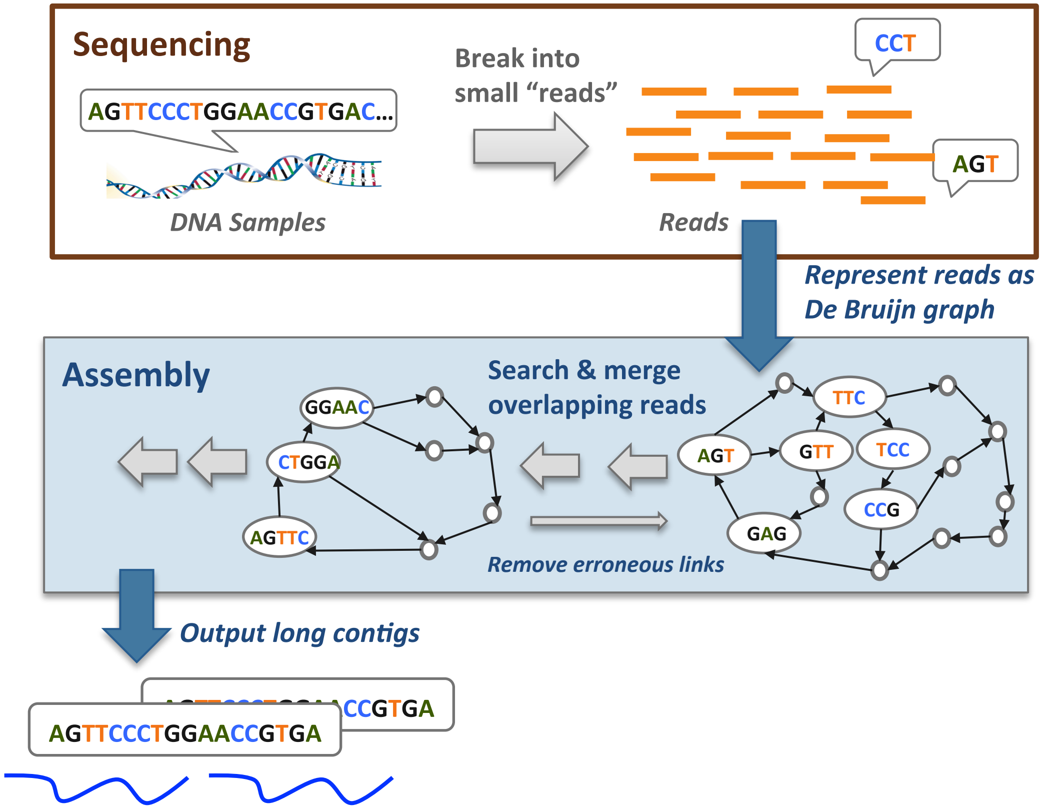


Task load balancing ?
Communication complexity ?

Genome Analysis in Bioinformatics

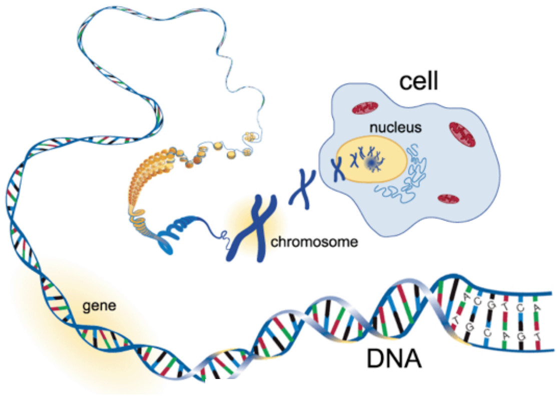
- Sequence alignment
- **Sequence assembly**
  - **Reconstruct** long DNA sequences by merging many small fragments
- Gene mapping

Hard to read whole genomes in current sequencing technology. Instead, read many small fragments , called "**reads**".



Larger raw data & overlapping reads

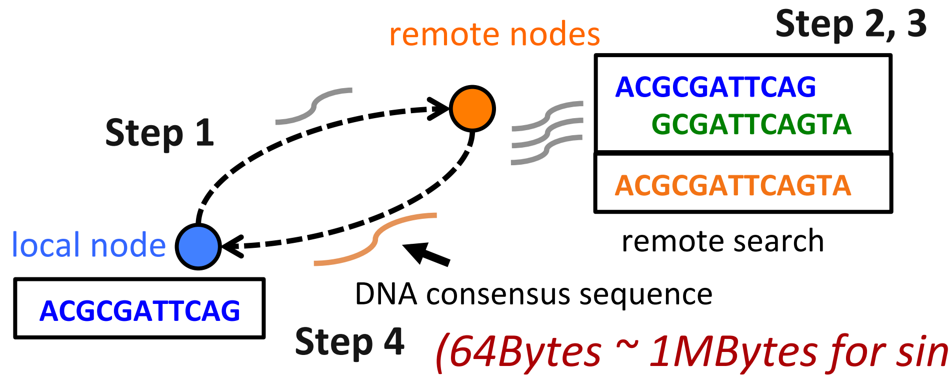
- Human Genome: 2TB ~ 3TB DNA reads
- Metagenome: PB ~ EB+ level DNA reads



[Adapted from National Human Genome Research Institute]

Massive Data Movement in Kiki Genome Assembly

Basic edge merging algorithm



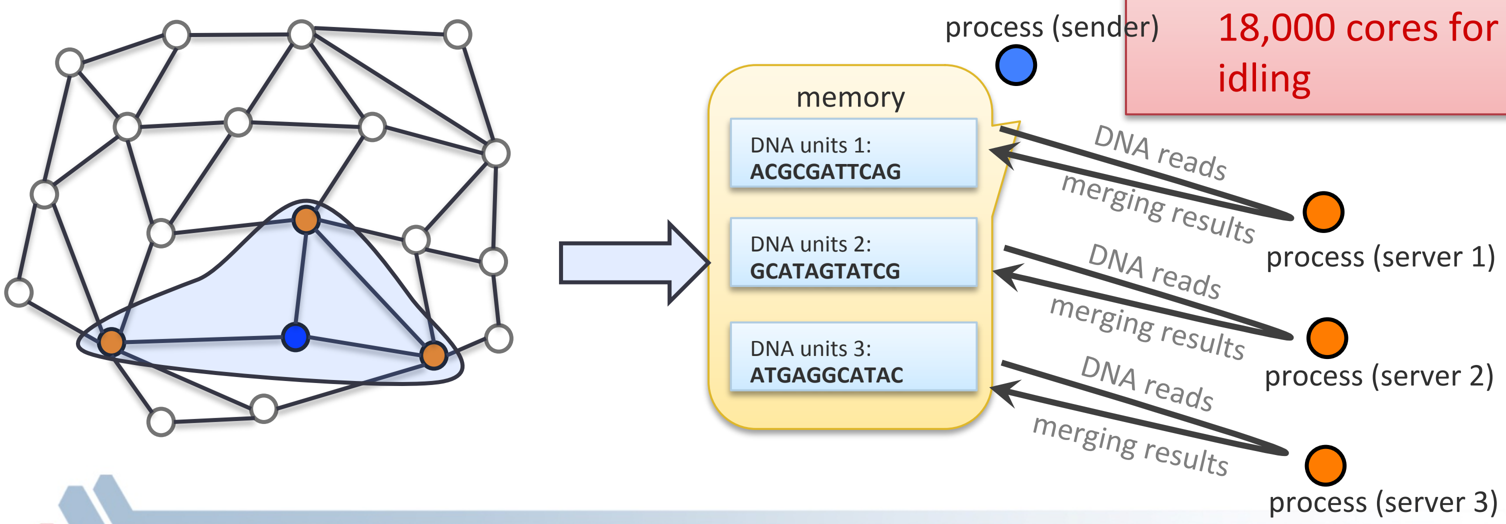
1. **Send local DNA unit to that node;**
2. **Search matching unit on that node;**
3. **Merge two units on that node;**
4. **Return merged unit.**

(64Bytes ~ 1MBytes for single message)

Large amount of outstanding data movement

Hard to balance task load

- 10^6+ outstanding msgs / rank
- 2.3TB sample was assembled on 18,000 cores for 4 days, 90%+ of time idling



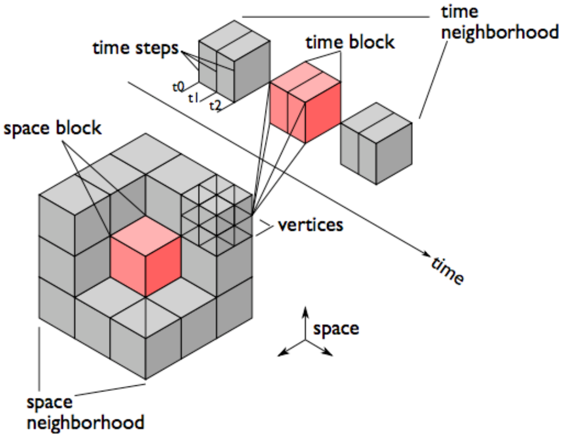
Particle Tracing and Graph in Parallel Visualization

- Particle tracing
  - e.g., For Rayleigh–Taylor instability
    - Interface between a heavy fluid overlying a light fluid

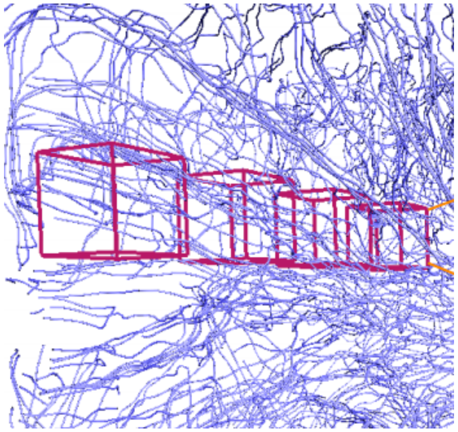
Mushroom cloud:
RTI at the interface between hot less-dense and cold more-dense air

- Irregular graph visualization
  - Completely data-driven
  - Possible optimization is unclear but is interesting to investigate !

Semi-regular Communication in Particle tracing :
Exchange particles in 4D time-space neighborhoods



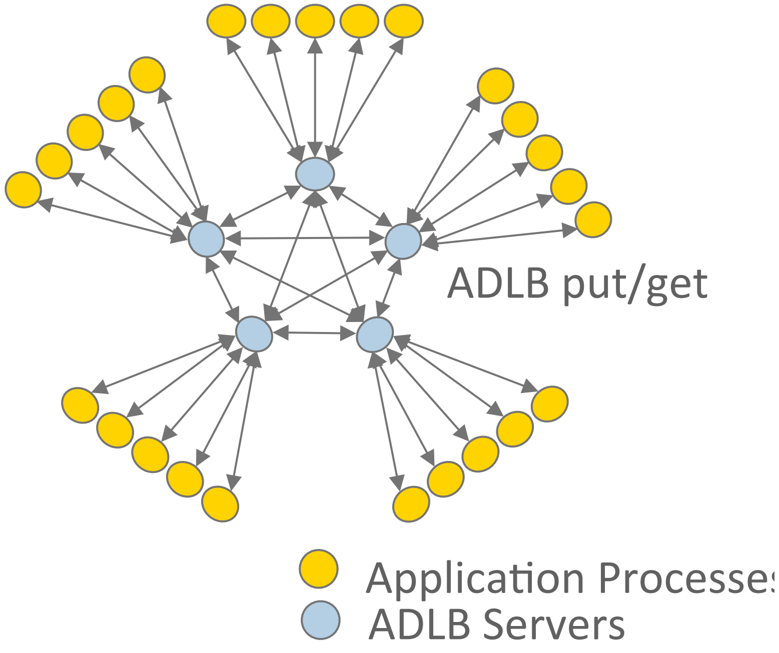
Irregular Task load



Courtesy Tom Peterka (Argonne)

GFMC in Nuclear Physics

- Green’s Function Monte Carlo
  - The “gold standard” for *ab initio* calculations in nuclear physics at Argonne (Steve Pieper, PHY)
- Irregular pattern for load balancing
  - A non-trivial master/slave algorithm, with assorted work types and priorities
  - multiple processes create work dynamically
  - large work units



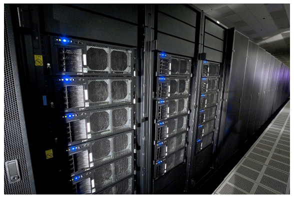
Courtesy Rusty Lusk (Argonne)

Complexity in Hardware Design

1996 ASCI Red

Terascale

2008 IBM Roadrunner



Petascale

Increasing power per processor

Hit the power wall, multi-core started

2012 MIRA

2016 Cori

10Petasale

2017-2018 Summit

2018-2019 Aurora

100Petasale

Complexity of processors and memory design

- Heterogeneous (i.e., CPU+GPU/Manycore)
- Fat node performance (many threads/cores)
- On-package memory
- I/O Burst buffer
- …

Exascale

Many-core Architectures

- Massively parallel environment
- Intel® Xeon Phi co-processor
  - 60 cores inside a single chip, 240 hardware thread
  - SELF-HOSTING in next generation

[Adapted from Intel]

- Blue Gene/Q

Node resources	Mira	Aurora
#Cores/ #Threads	16/64	60+/240+ → ***4X***
Memory	16GB	32GB (High Bandwidth Memory)

→ ***2X***

[Adapted from Wikipedia]

Hardware Characteristics

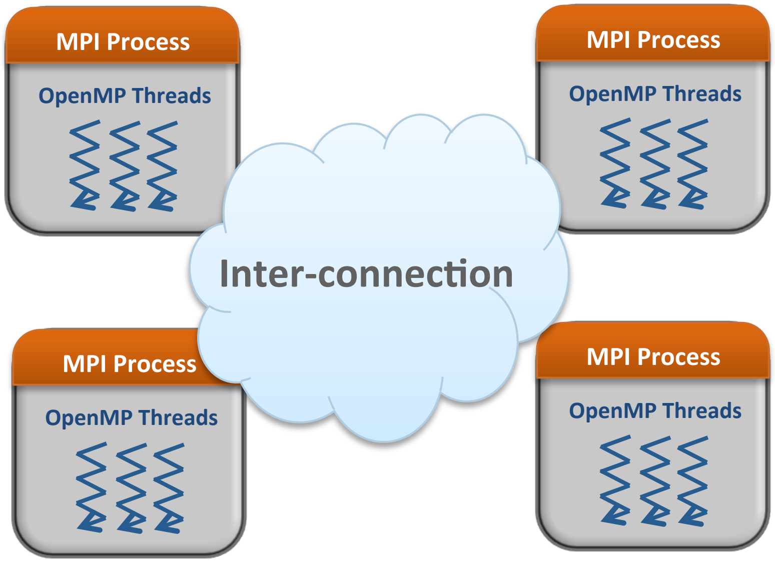
- **Large amount of simple and low frequency cores**
- **Other on-chip resources are growing at a lower rate…**

Scientific Programming models (1)

Hybrid MPI+Threads model

- To fully utilize the hardware resources
  - **Massive parallelism in computation**
  - **On-chip resource sharing**
- To handle complex & irregular computation
  - **Dynamic & fine-grained task scheduling**

- Large amount of low frequency cores
- Limited other on-chip resources (e.g., memory)



Hybrid MPI + threads modes

Traditional Thread Single mode

```
/* user computation */

MPI_Function ( );

/* user computation */
```

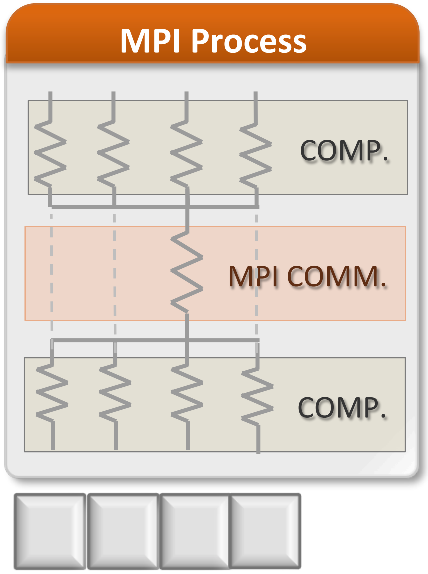
Funneled / Serialized mode (most widely used)

- Multithreaded user computation
- Still single thread issues MPI calls

```
#pragma omp parallel
{ /* user computation */ }

MPI_Function ( );

#pragma omp parallel
{ /* user computation */ }
```

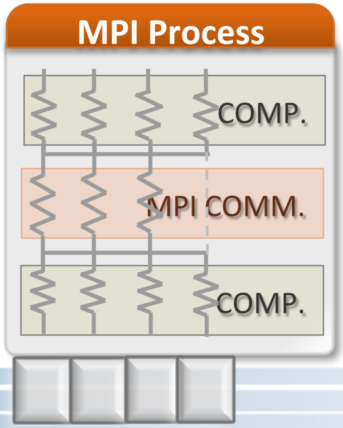


Multithreading mode

```
#pragma omp parallel
{
    /* user computation */

    MPI_Function ( );

    /* user computation */
}
```



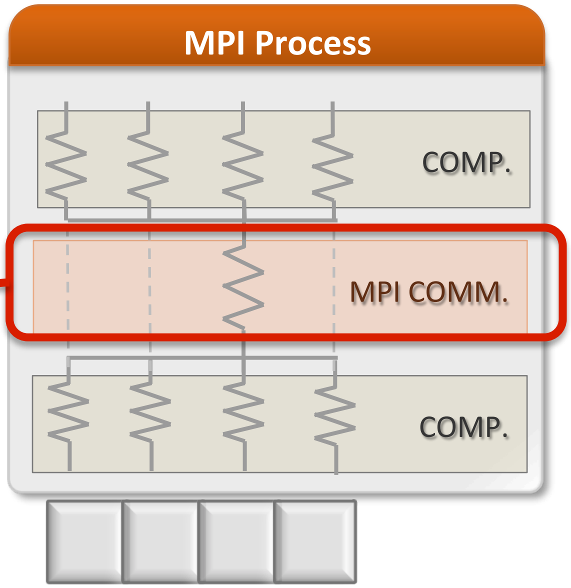
Problem Statement

- Multiple threads are created for user computation
- But only single thread issues MPI

```
#pragma omp parallel
{ /* user computation */ }

MPI_Function ( );

#pragma omp parallel
{ /* user computation */ }
```

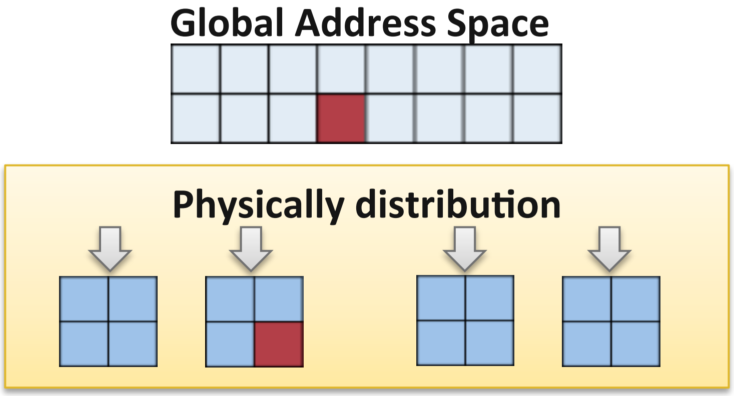


- Large amount of IDLE threads
- Single lightweight core delivers poor performance

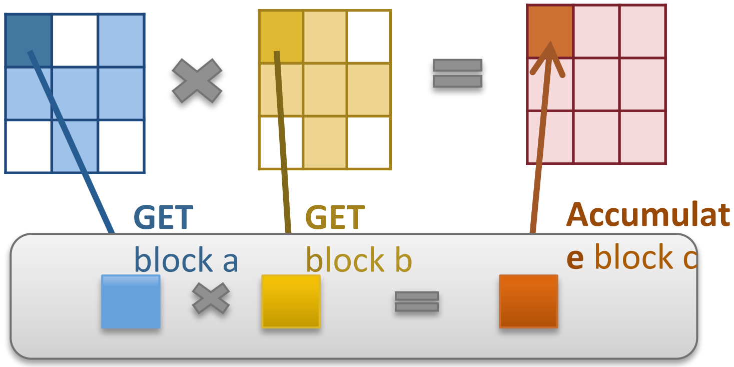
Scientific Programming models (2)

One-sided programming

- PGAS-like applications (e.g., Global Arrays for NWChem)
- CESAR project (Next generation Nuclear Reactor Modeling)



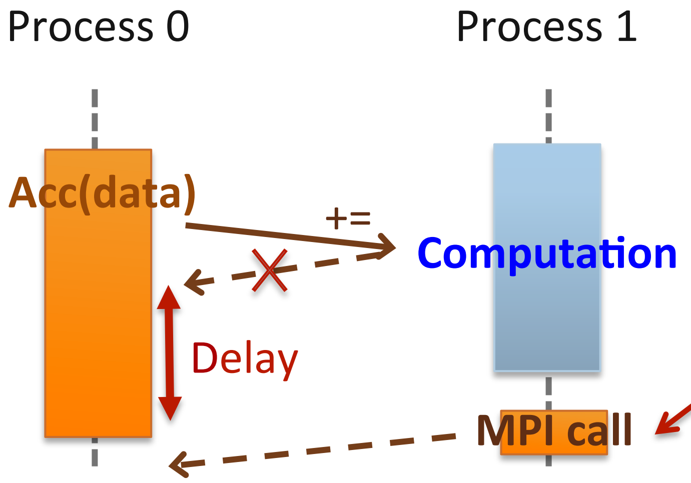
- For better resource sharing
  - **Memory sharing across nodes on distributed memory systems**
- To handle complex & irregular computation
  - **Dynamic, data-driven communication**



Perform DGEMM in local buffer

Problem Statement

- MPI one-sided operations are not truly one-sided !
  - Some operations can be supported by hardware (e.g., PUT/GET on IB, Cray)
  - Other operations still have to be **handled by software** (e.g., 3D accumulates of double precision data)



Software implementation of one-sided operations means that **the target process has to make an MPI call to make progress**.

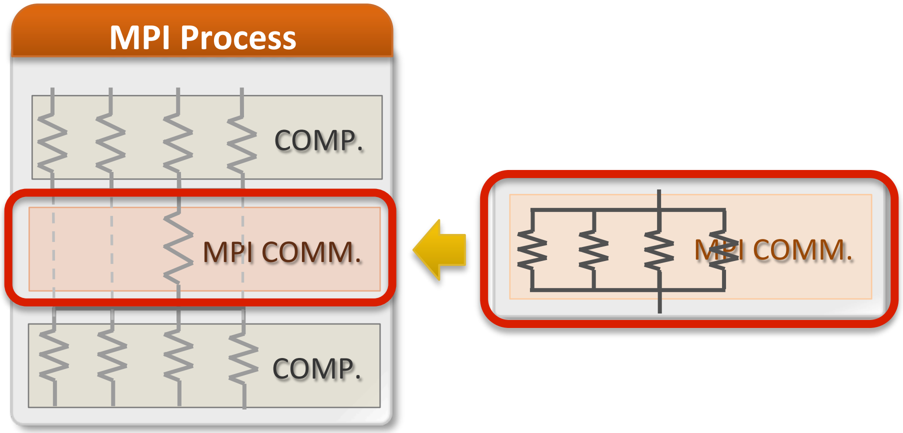
Not TRULY asynchronous !

Research Contribution

- Enable **highly efficient message passing** on many-core architectures for various kinds of scientific applications

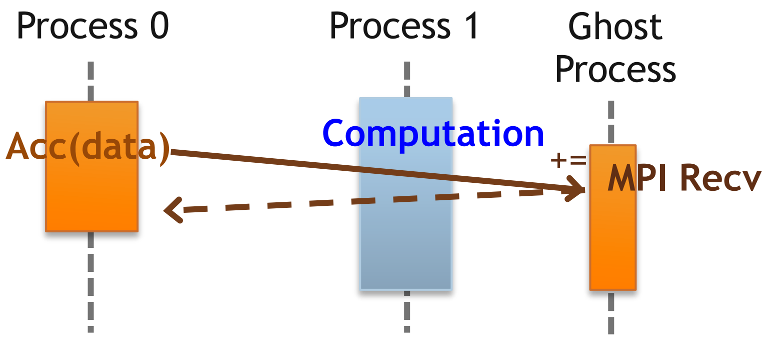
I. Multithreaded MPI for hybrid MPI+ threads model

- **Sharing Idle Threads** with application inside MPI
- Optimizing MPI internal processing by **massive parallelism**



II. Process-based Asynchronous Progress for MPI one-sided programming

- **Flexible & Portable & Low overhead**
- Improve SW-handled RMA operations without affecting HW-handled RMA.



MT-MPI
Multithreaded MPI for Many-Core Environments

Published Paper

1. “MT-MPI: Multithreaded MPI for Many-core Environments.” M. Si, A. Pena, P. Balaji, M. Takagi, and Y. Ishikawa. ICS 2014.

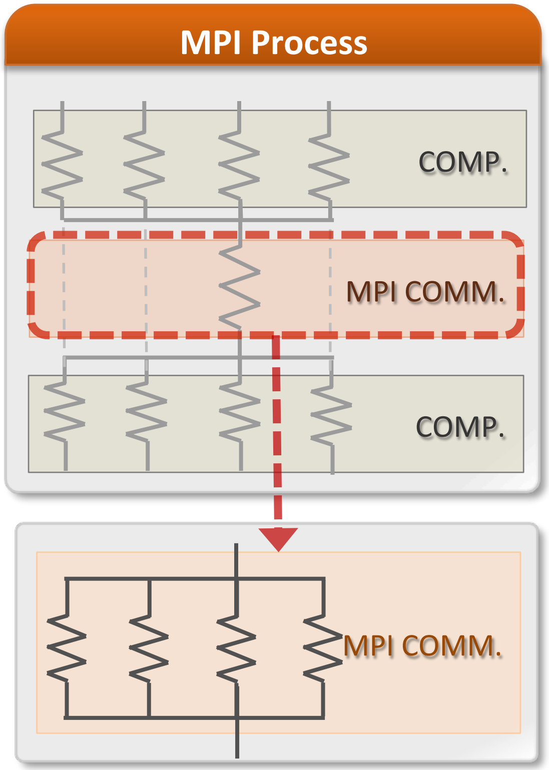
Core Concept of Multithreaded MPI

- Sharing Idle Threads with Application inside MPI
- Parallelizing MPI internal processing

```
#pragma omp parallel
{ /* user computation */ }

MPI_Function ( ){
        #pragma omp parallel
        {
                /* MPI internal task */
        }
}

#pragma omp parallel
{ /* user computation */ }
```

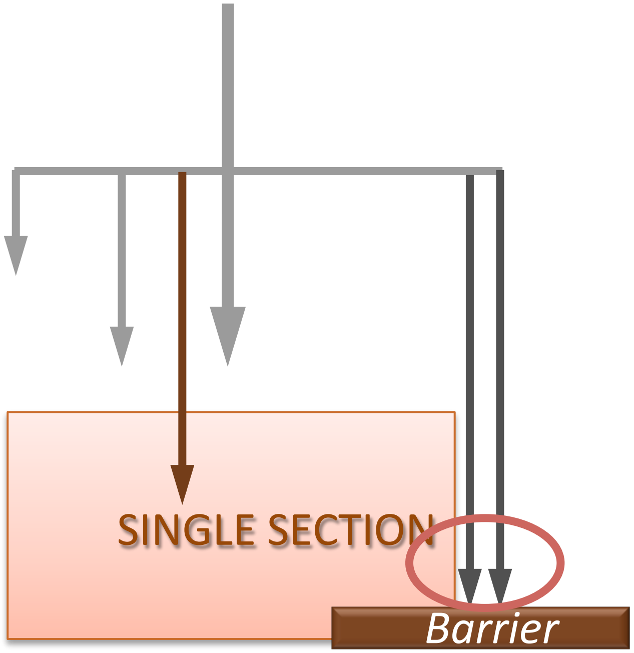


Challenges (1/2)

- Some parallel algorithms are not efficient with insufficient threads, need tradeoff

```
#pragma omp parallel
{
        /* user computation */

        #pragma omp single
        {
                /* MPI_Calls */
        }
}
```



Number of available threads is UNKNOWN !

Challenges (2/2)

- Nested parallelism
  - Simply creates new Pthreads, and offloads thread scheduling to OS

```
#pragma omp parallel
{
      #pragma omp single
      {
            #pragma omp parallel
            { … }
      }
}
```

Creates N Pthreads !

Creates N Pthreads !

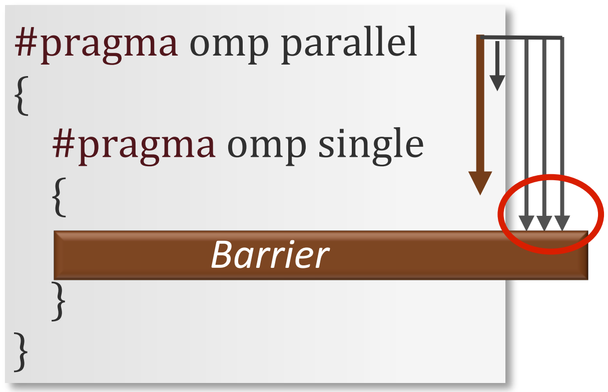
Threads Oversubscription

Should ONLY use IDLE threads.
However, it is UNKNOWN !

Design Overview

- **Modification in OpenMP runtime**
  - Expose number of IDLE threads
    - **Guaranteed Idle Threads**
    - Temporarily Idle Threads
- **Modification in MPI**
  - Parallelize internal tasks
    - Use **num_idle_threads** for **tradeoff** between sequential and parallelism algorithms
    - Use **num_idle_threads** for specifying num_threads in nested parallel region to **avoid threads overrunning issue**

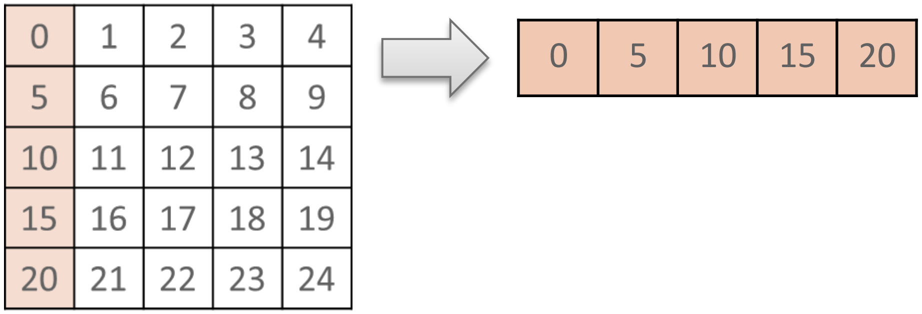
Example of Guaranteed Idle Threads



```
#pragma omp parallel
{
   #pragma omp single
   {
   }
}
```

MPI Internal Parallelism

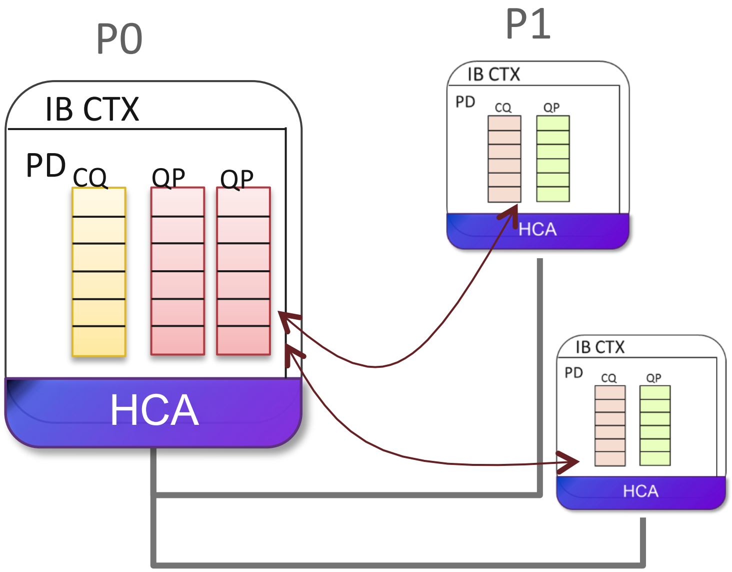
DDT packing/unpacking



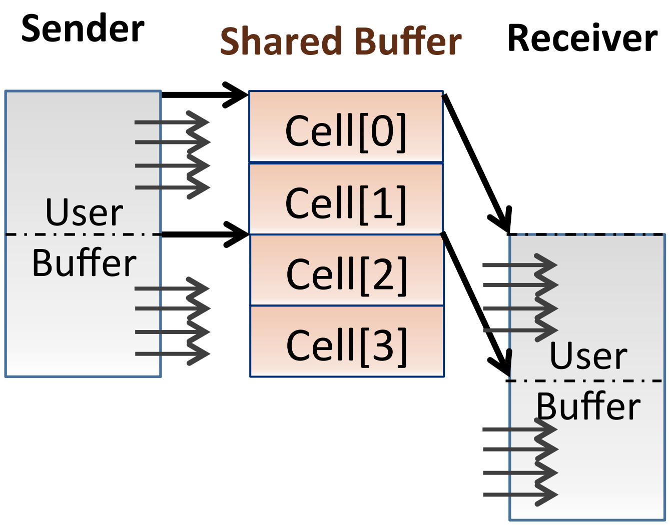
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

0	5	10	15	20

InfiniBand communication

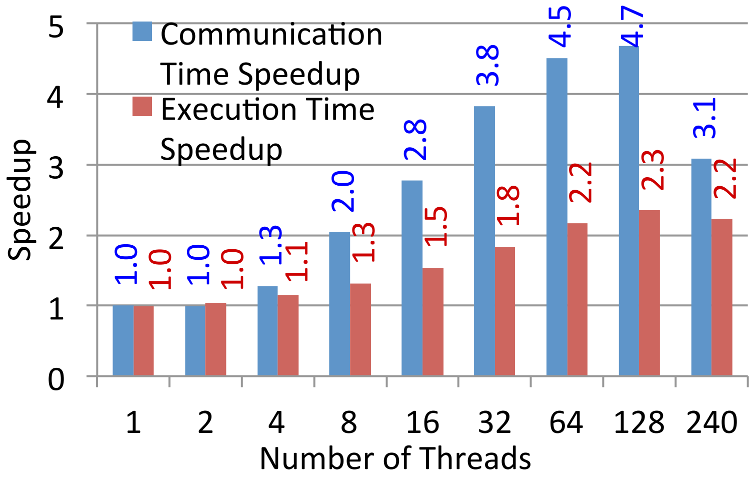


Shared memory communication

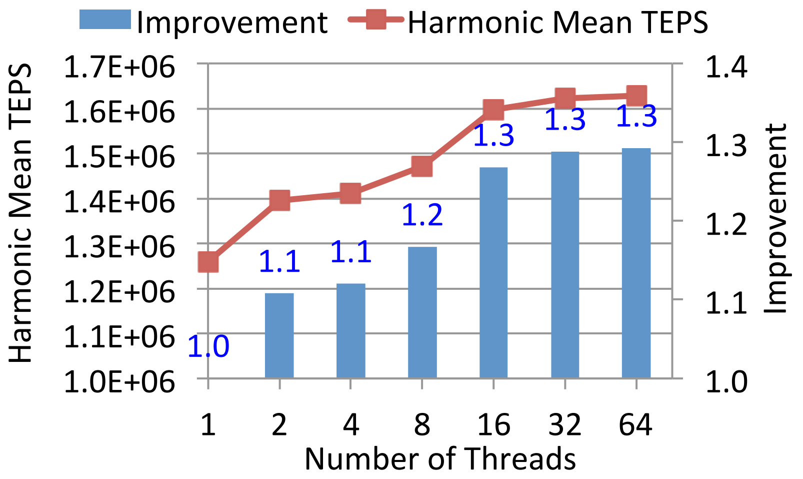


Evaluation on Stampede

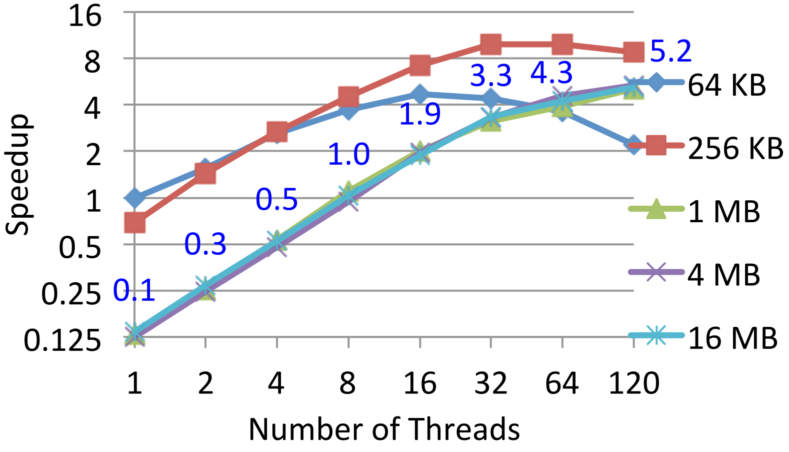
Hybrid MPI+OpenMP NAS MG (Class E, 64 processes) using parallelized DDT packing/unpacking



One-sided Graph500 (Scale $2^{22,}$ 64 processes) using parallelized InfiniBand communication



OSU P2P BW using parallelized shared memory communication



CASPER
Process-based Asynchronous Progress Model for MPI RMA

Papers

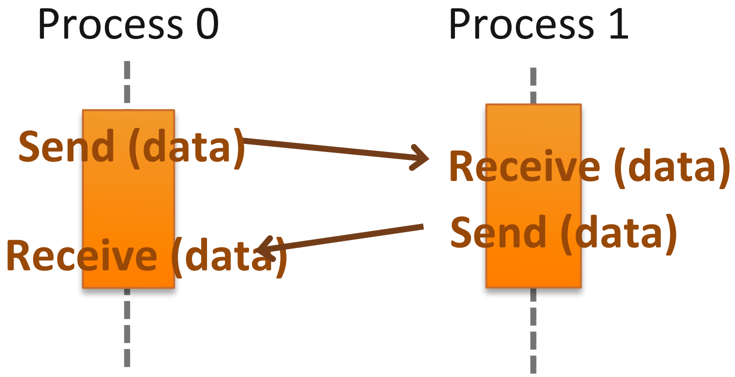
1. “Casper: An Asynchronous Progress Model for MPI RMA on Many-Core Architectures.” M. Si, A, Pena, J. Hammond, P. Balaji, M. Takagi, and Y. Ishikawa. IPDPS 2015.
2. “Scaling NWChem with Efficient and Portable Asynchronous Communication in MPI RMA.” M. Si, A. J Peña, J. Hammond, P. Balaji, and Y. Ishikawa. CCGrid 2015.
3. “A Dynamic Adaptable Process-based Asynchronous Progress“ Journal under preparation.

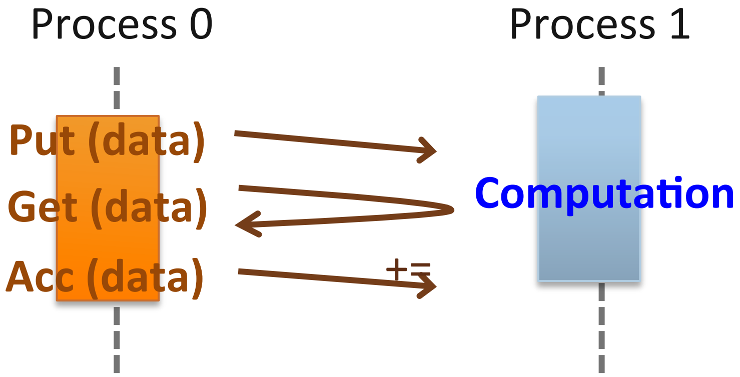
Invited Talk

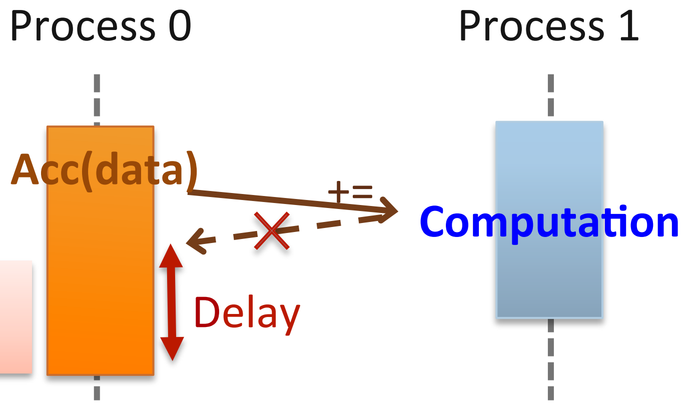
1. “Casper: An Asynchronous Progress Model for MPI RMA on Many-core Architectures.” M. Si. In The 2ed Workshop of INRIA-ILLINOIS-ANL-BSC Joint Laboratory on Extreme Scale Computing, Chicago, USA, 2014

Message Passing Models

- **Regular two-sided communication**
- **Irregular one-sided communication (Remote Memory Access)**





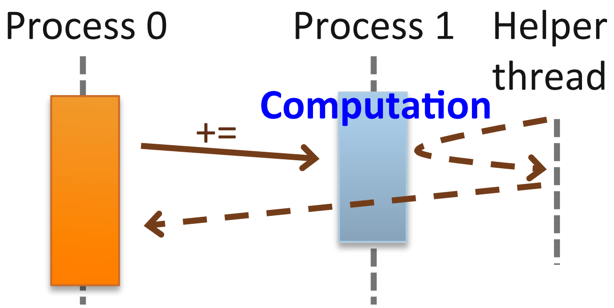


Non-contiguous Accumulate in MPI

Not TRULY asynchronous !

Traditional Approaches of Asynchronous Progress

- **Thread-based approach**
  - Every process has a **communication dedicated background thread**
  - Background thread polls progress



Cons:

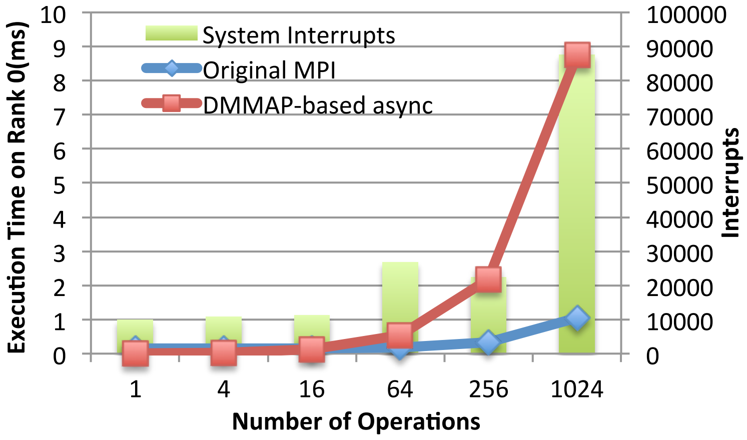
✗ **Waste 50% computing cores** or **oversubscribe** cores

✗ Overhead of **multithreading safety**

- **Interrupt-based approach**
  - Assume **all hardware resources are busy** with user computation on target processes
  - Utilize **hardware interrupts** to awaken a kernel thread

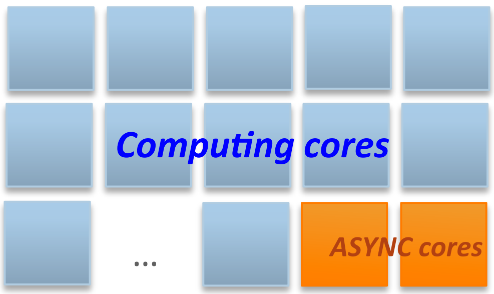
Cons:

✗ Overhead of **frequent interrupts**



DMMAP-based ASYNC overhead on Cray XC30

Casper Process-based ASYNC Progress



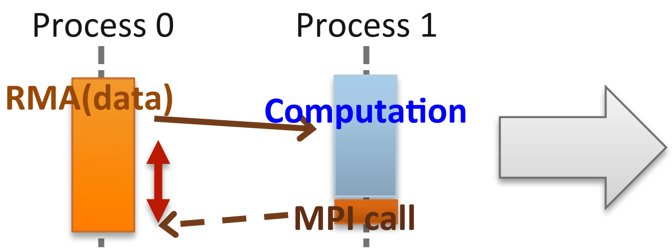
- **Multi- and many-core architectures**
  - “Infinite cores”
  - Not all of the cores are always keeping busy
- **Process-based asynchronous progress**
  - Dedicating **arbitrary number of cores to “ghost processes”**
  - **Ghost process intercepts all RMA operations** to the user processes

Pros:

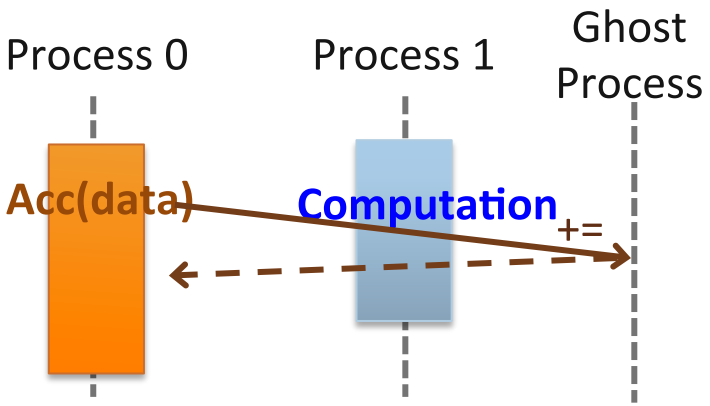
✓ No overhead caused by **multithreading safety** or **frequent interrupts**

✓ **Flexible core deployment**

✓ **Portable PMPI[*] redirection**



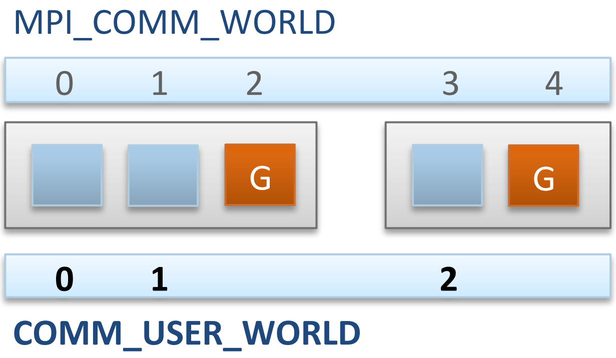
Original communication



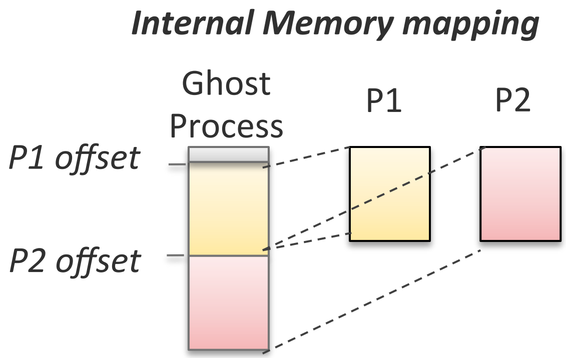
Communication with Casper

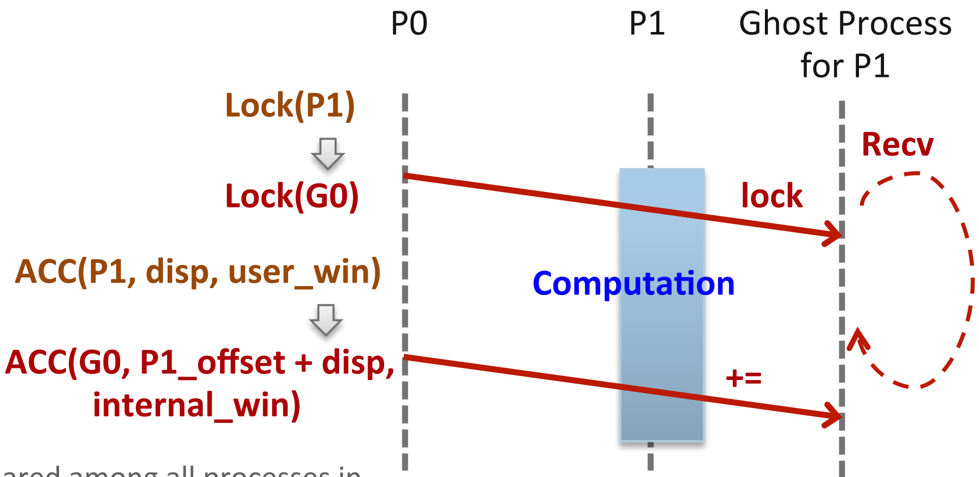
Basic Design of Casper

- **Three primary functionalities**
  1. Transparently replace MPI_COMM_WORLD by **COMM_USER_WORLD**



  2. **Shared memory mapping** between local user and ghost processes by using MPI-3 MPI_Win_allocate_shared*.
  3. **Redirect RMA operations** to ghost processes

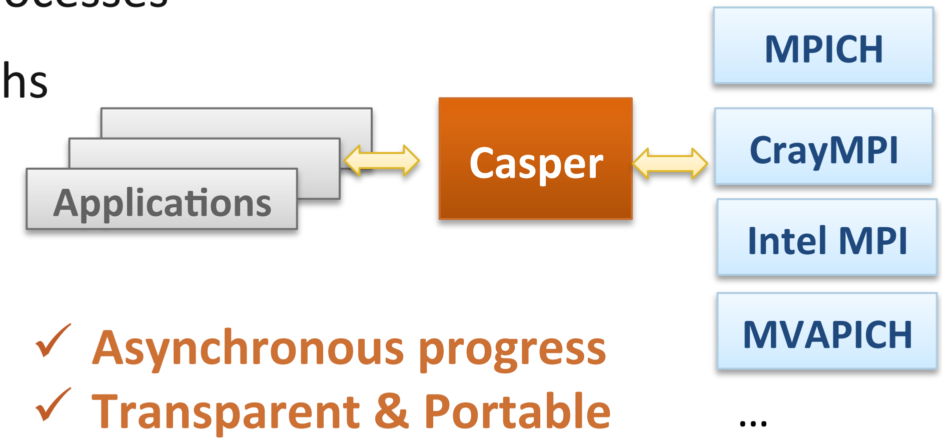




* **MPI_WIN_ALLOCATE_SHARED** : Allocates window that is shared among all processes in the window's group, usually specified with MPI_COMM_TYPE_SHARED communicator.

Challenges

- Ensuring Correctness and Performance
  - Lock Permission Management
  - Self Lock Consistency
  - Managing Multiple Ghost Processes
  - Multiple Simultaneous Epochs



- ✓ Asynchronous progress
- ✓ Transparent & Portable
- ✓ Correctness
- ✓ Performance

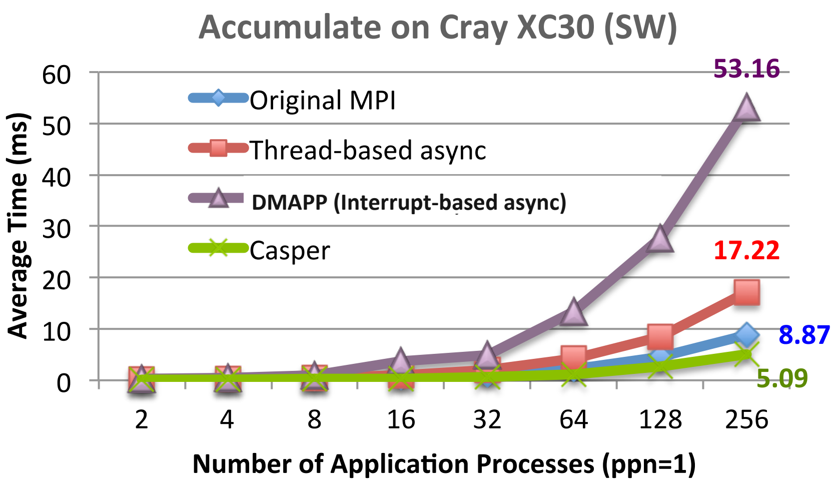
Evaluation on Cray XC30 (1)

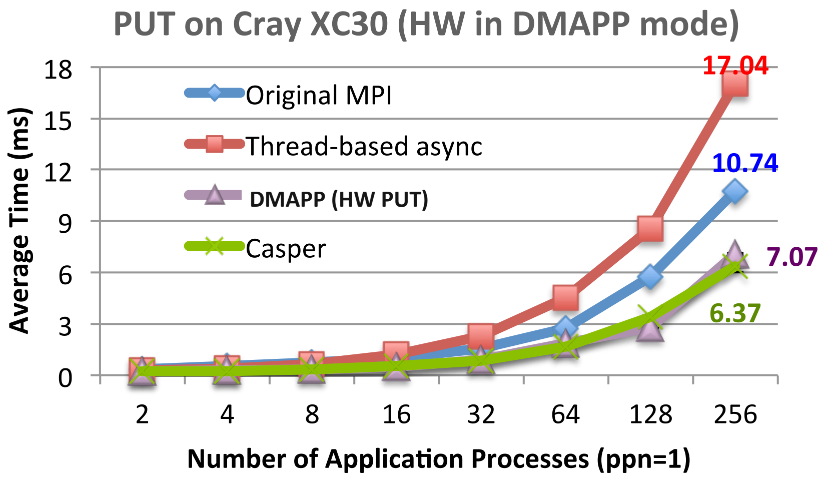
RMA implementation in Cray MPI v6.3.1

	HW-handled OP	ASYNC. mode
Original mode	NONE	**Thread**
DMAPP mode	Contig. PUT/GET	**Interrupt**

Test scenario

```
Lock_all (win);
for (dst=0; dst<nproc; dst++) {
        OP(dst, double, cnt = 1, win);
        Flush(dst, win);
        busy wait 100us; /*computing*/
}
Unlock_all (win)
```



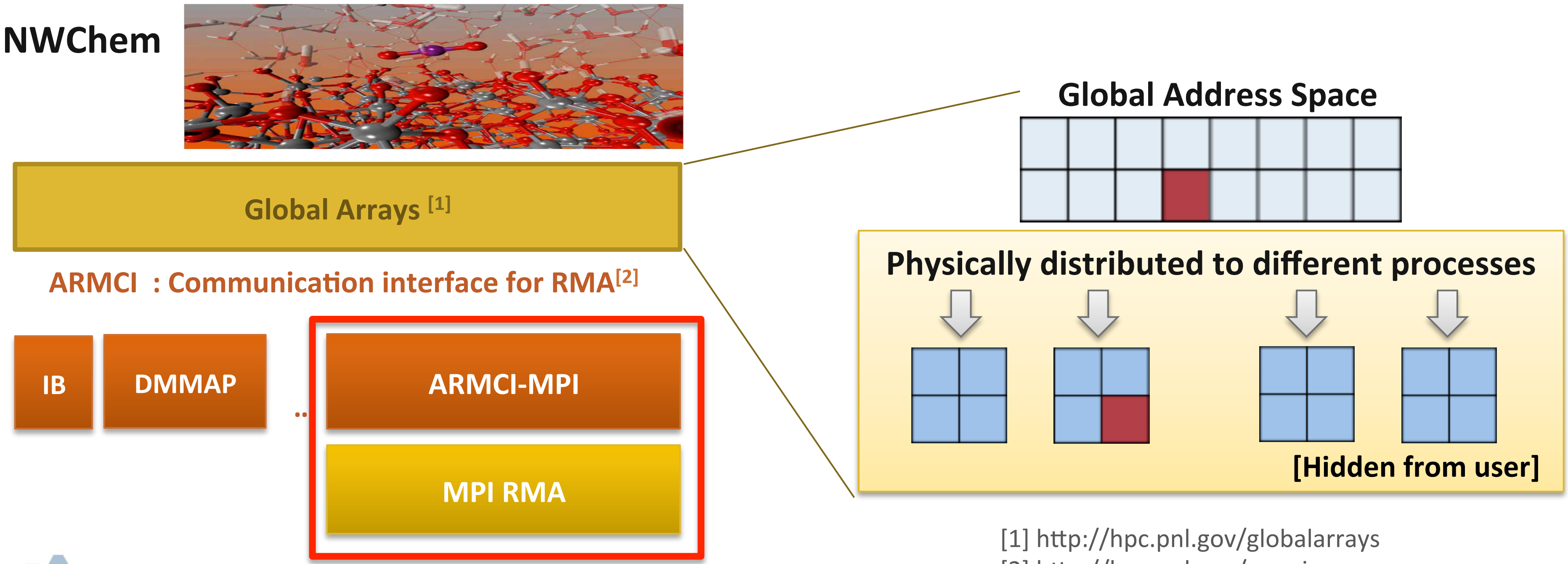


Casper provides asynchronous progress for SW-handled operations.

No impact on HW-handled operations.

Evaluation on Cray XC30 (2)

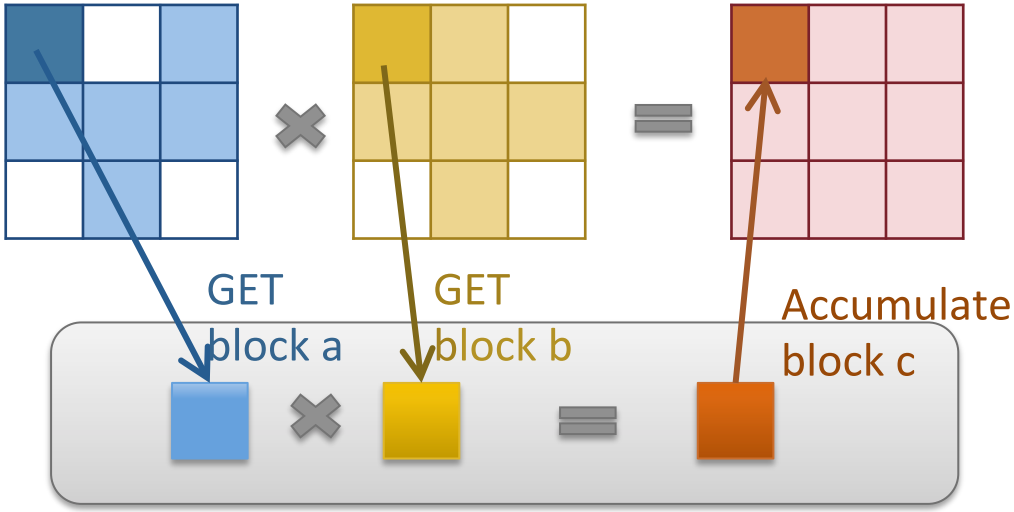
- **NWChem Quantum Chemistry Application**
  - Computational chemistry application suite composed of many types of simulation capabilities.
  - **ARMCI-MPI** (Portable implementation of **Global Arrays over MPI RMA**)


NWChem
Global Arrays [1]
ARMCI : Communication interface for RMA[2]
IB
DMMAP
..
ARMCI-MPI
MPI RMA
Global Address Space
Physically distributed to different processes
[Hidden from user]

[1] http://hpc.pnl.gov/globalarrays
[2] http://hpc.pnl.gov/armci

Evaluation on Cray XC30 (3)

- Typical Get-Compute-Update mode in GA programming



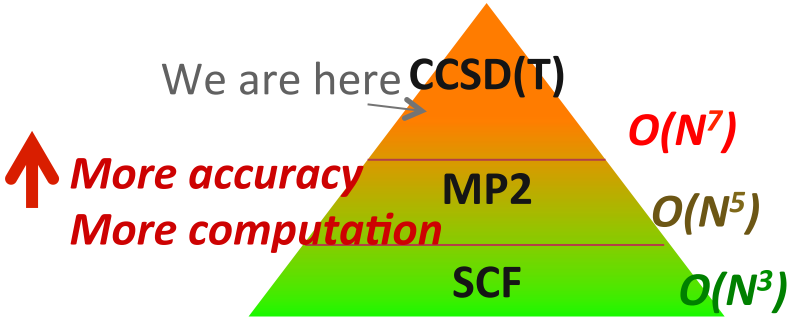
Perform DGEMM in local buffer

Pseudo code

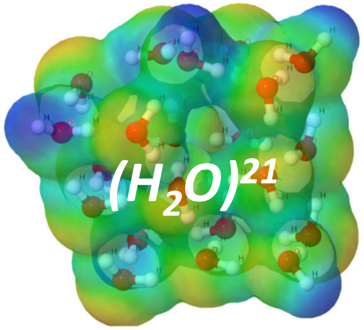
```
for i in I blocks:
  for j in J blocks:
     for k in K blocks:
        GET block a from A
        GET block b from B
        c += a * b /*computing*/
      end do
      ACC block c to C
  end do
end do
```

Evaluation on Cray XC30 (4)

- “Gold standard” CCSD(T)



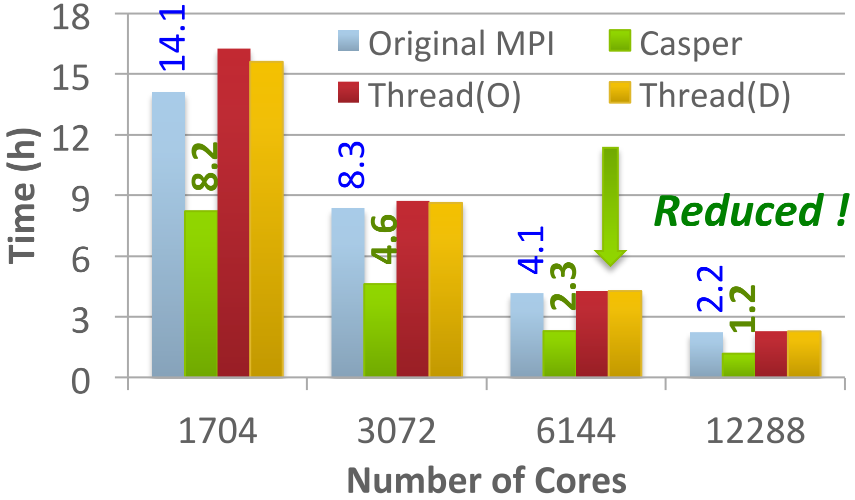
- Water molecular $(H_2O)_{21}$



Core deployment (24 cores per node)

	# COMP.	# ASYNC.
Original MPI	24	0
Casper	20	4
Thread-ASYNC (oversubscribed)	24	24
Thread-ASYNC (dedicated)	12	12

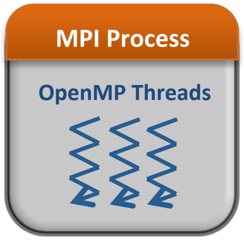
NWChem CCSD(T) for W21=$(H_2O)_{21}$ with pVDZ



Summary

- Applications & hardware architectures are becoming more complex
- Parallelism & Resource sharing & Dynamic computation are important !
- Two most popular programming models used in modern applications

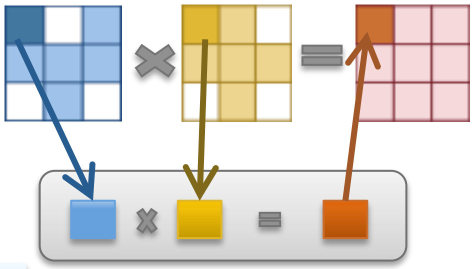
1. Hybrid MPI+Threads model



Problem:

- **Many IDLE threads in COMM.**
- **Single lightweight core performs COMM.**

2. One-sided programing



Problem:

- **Lack asynchronous progress in MPI RMA**

Solutions

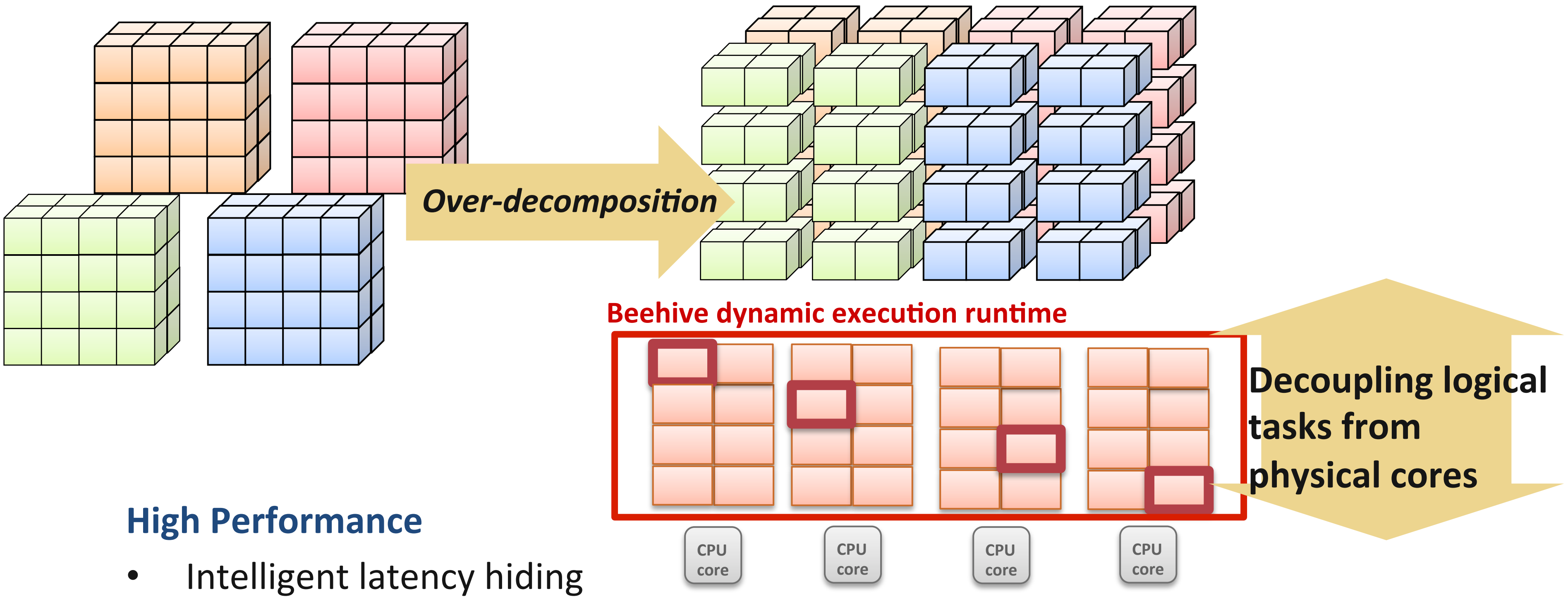
Multithreaded MPI

- Parallelizing MPI communication by utilizing user IDLE threads

Process-based Asynchronous Progress

- Provide Portable & Efficient & Flexible asynchronous progress for MPI RMA

Future Research Plan: BEEHIVE



Over-decomposition

Beehive dynamic execution runtime

Decoupling logical tasks from physical cores

High Performance

- Intelligent latency hiding
- Migration for better load balance

Fault Resilience

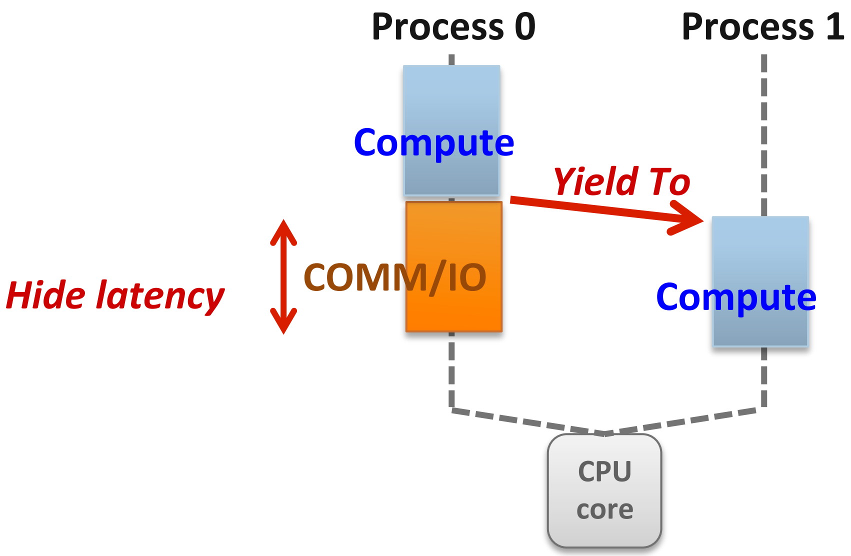
- Lightweight checkpointing
- Dynamic migration

Power Efficiency

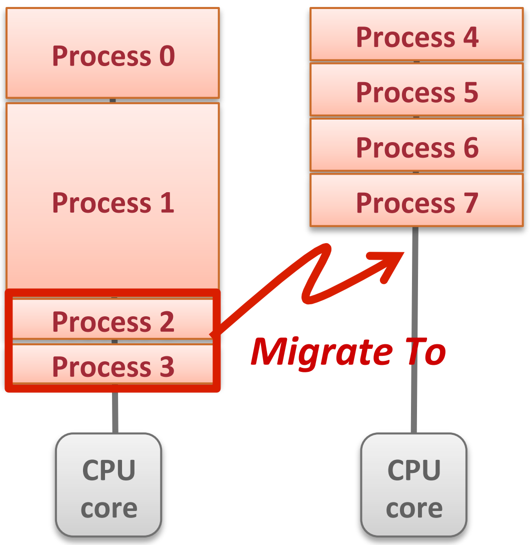
- Computation and data consolidation

Under investigation: Optimization for High Performance

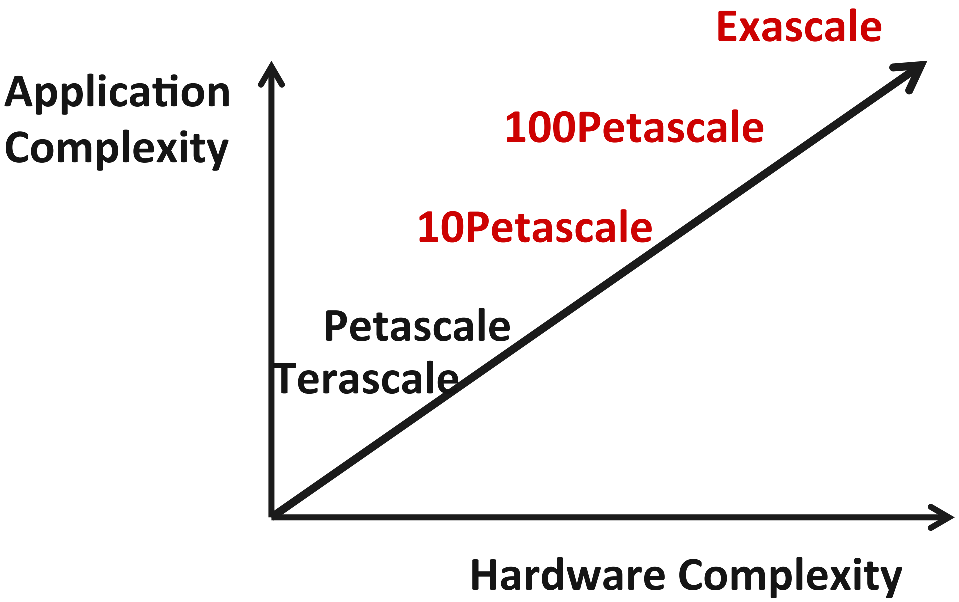
- **Intelligent Latency Hiding**
  - Context switch when blocking in communication / IO.
  - Yield to a "Ready-To-Go" process



- **Load Balancing**
  - Migrate processes from busy core to relatively idle core



Thank you


Exascale
Application Complexity
100Petascale
10Petascale
Petascale
Terascale
Hardware Complexity

Backup Slides

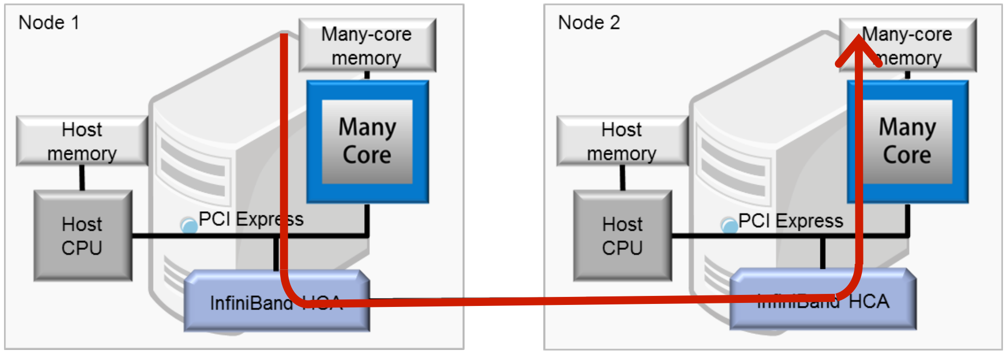
Selected Publications

MPI optimization for many-core architectures (Ph.D. research)

1. "Scaling NWChem with Efficient and Portable Asynchronous Communication in MPI RMA." M. Si, A. J Peña, J. Hammond, P. Balaji, and Y. Ishikawa. CCGrid 2015.
2. "Casper: An Asynchronous Progress Model for MPI RMA on Many-Core Architectures." M. Si, A, Pena, J. Hammond, P. Balaji, M. Takagi, and Y. Ishikawa. IPDPS 2015.
3. "MT-MPI: Multithreaded MPI for Many-core Environments." M. Si, A, Pena, P. Balaji, M. Takagi, and Y. Ishikawa. ICS 2014.

Low level communication facility for many-core architectures (Master research)

5. "Direct MPI Library for Intel Xeon Phi Co-Processors." M. Si, M. Takagi, and Y. Ishikawa. In Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW) 2013.
6. "Design of Direct Communication Facility for Many-Core Based Accelerators." M. Si and Y. Ishikawa. In Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW) 2012.

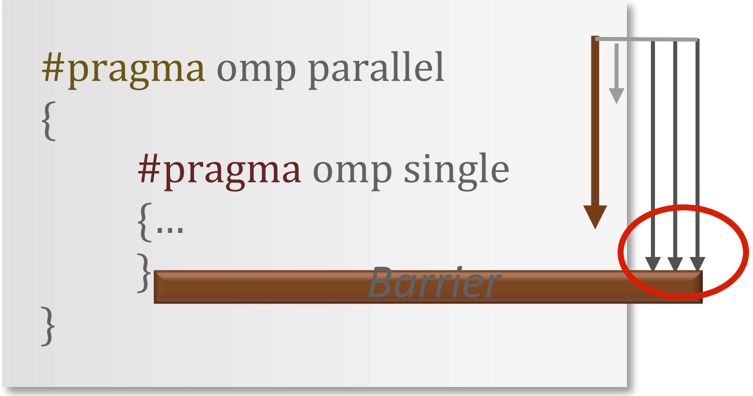


Guaranteed Idle Threads VS Temporarily Idle Threads

- Guaranteed Idle Threads
  - Guaranteed idle until Current thread exits

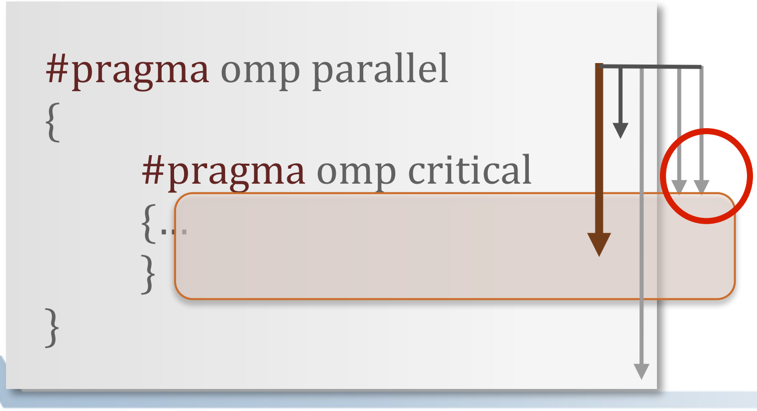
- Temporarily Idle Threads
  - Current thread does not know when it may become active again

Example 1



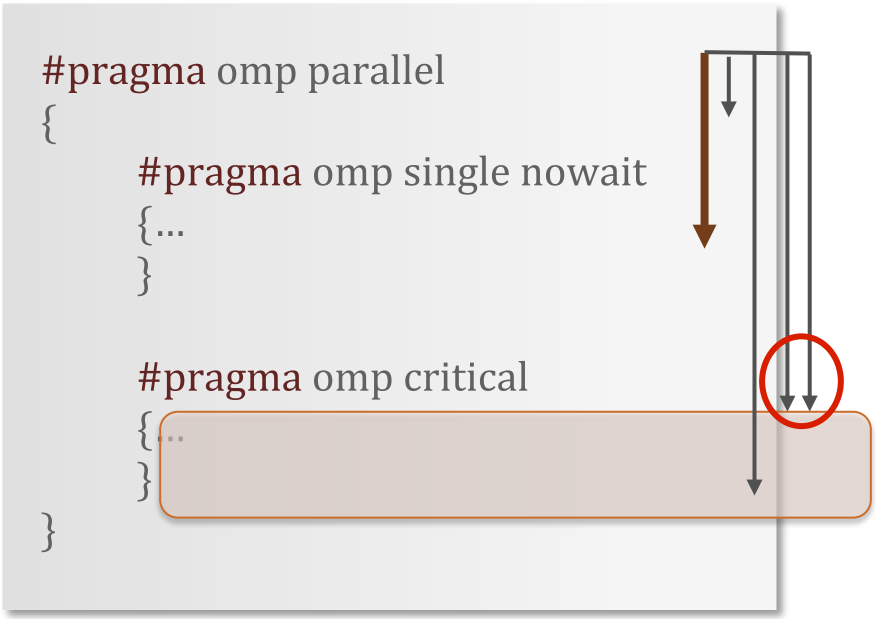
```
#pragma omp parallel
{
        #pragma omp single
        {…
        }
}
```

Example 2



```
#pragma omp parallel
{
        #pragma omp critical
        {…
        }
}
```

Example 3



```
#pragma omp parallel
{
        #pragma omp single nowait
        {…
        }

        #pragma omp critical
        {…
        }
}
```

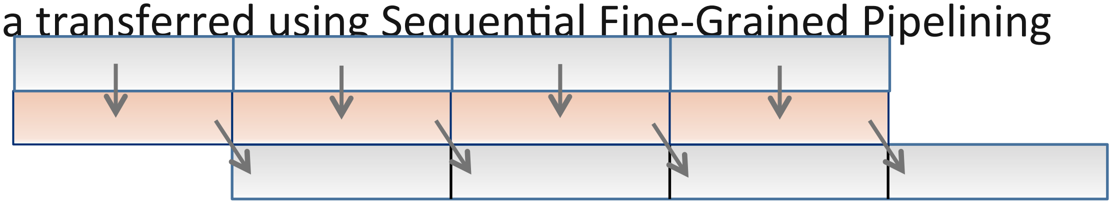
Expose Guaranteed Idle Threads

- MPI uses Guaranteed Idle Threads to schedule its internal parallelism efficiently (i.e. change algorithm, specify number of threads)

```
#pragma omp parallel
#pragma omp single
{
        MPI_Function {
            num_idle_threads = omp_get_num_guaranteed_idle_threads( );
            if ( num_idle_threads < N ) {
             /* Sequential algorithm */
            } else {
                #pragma omp parallel num_threads(num_idle_threads)
                { … }
            }
        }
}
```

Sequential Pipelining VS Parallelism

- Small Data transferring (< 128K)
  - Threads synchronization overhead > parallel improvement
- Large Data transferring
  - Data transferred using Sequential Fine-Grained Pipelining

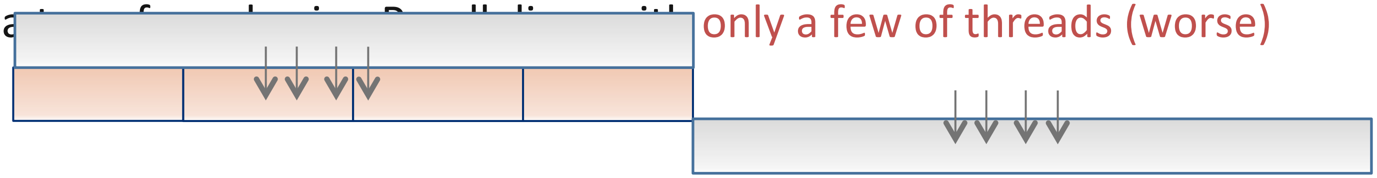


Sender Buffer

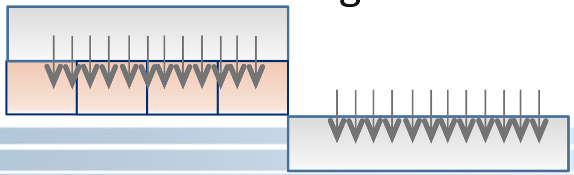
Shared Buffer

Receiver Buffer

  - Data transferred using Parallelism with only a few of threads (worse)



  - Data transferred using Parallelism with many threads (better)

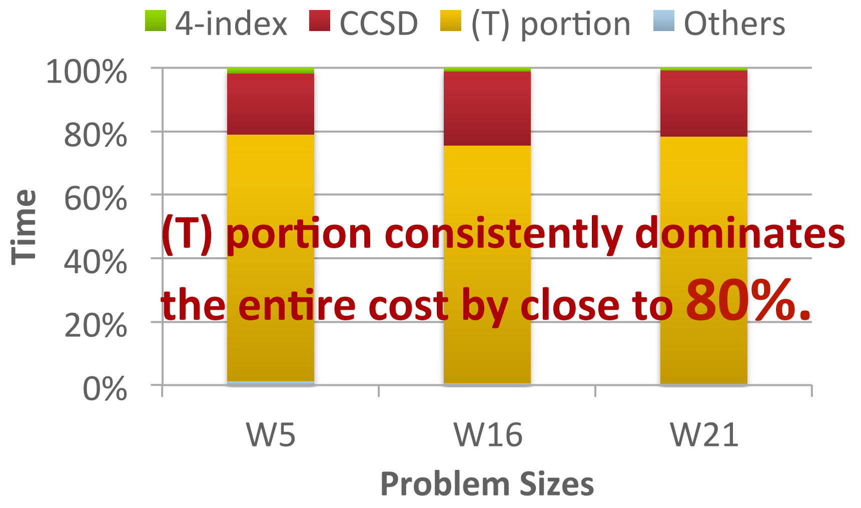


NWChem CCSD(T) Profiling

Internal steps in CCSD(T) task

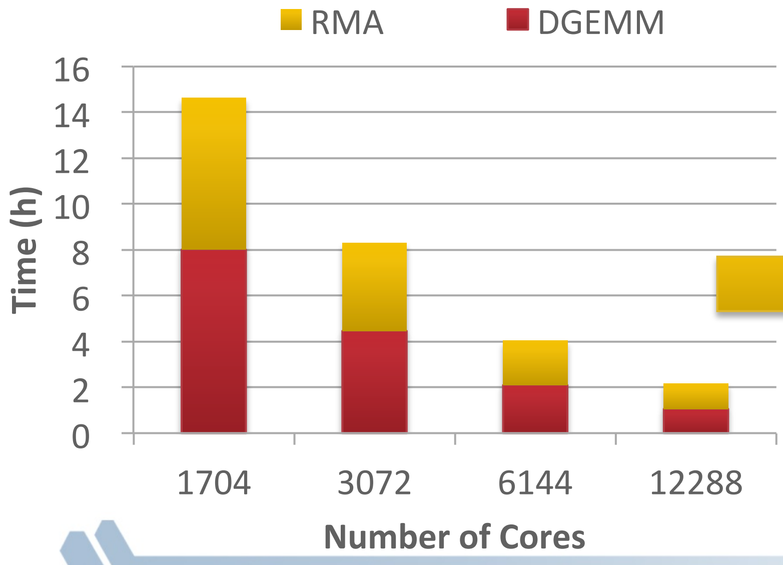
Self-consistent field (SCF)
Four-index transformation (4-index)
CCSD iteration
(T) portion

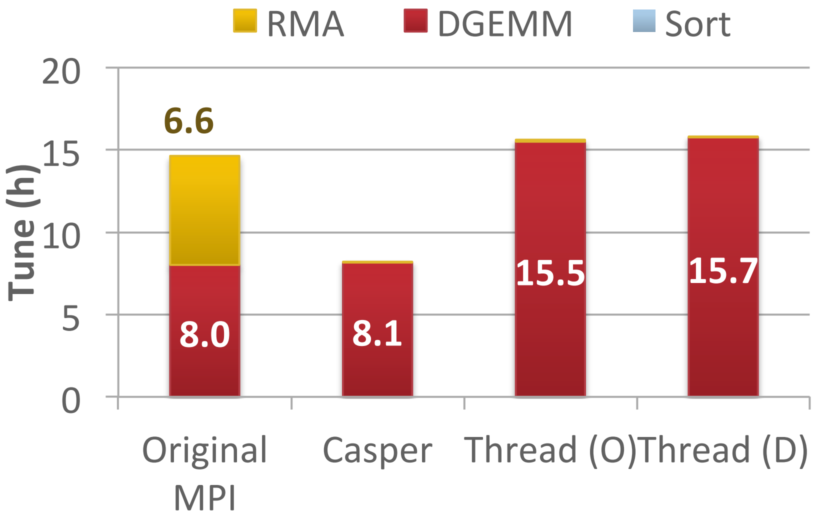
CCSD(T) internal steps in varying water problems



(T) portion consistently dominates the entire cost by close to 80%.

(T) Portion profiling for W_{21} with Original MPI


RMA
DGEMM
Time (h)
Number of Cores


RMA
DGEMM
Sort
Tune (h)
Original MPI
Casper
Thread (O)
Thread (D)