# Casper: Process-based Asynchronous Progress Model for MPI One-Sided Communication

## Scaling NWChem with Efficient and Portable Asynchronous Communication on NERSC Edison Supercomputer

**Min Si**[1], Antonio J. Peña[2], Jeff Hammond[3], Pavan Balaji[1], Yutaka Ishikawa[4]

[1] Argonne National Laboratory, USA
*{msi, balaji}@anl.gov*

[2] Barcelona Supercomputing Center, Spain
*antonio.pena@bsc.es*

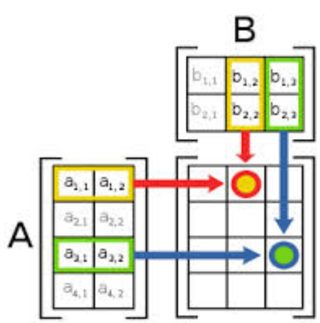
[3] Intel Labs, USA
*jeff.r.hammond@intel.com*

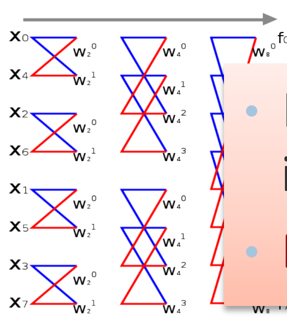
[4] RIKEN AICS, Japan
*yutaka.ishikawa@riken.jp*

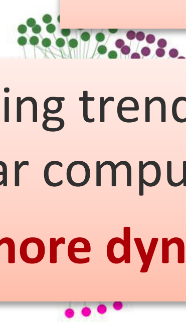
# Irregular Computation in Scientific Applications

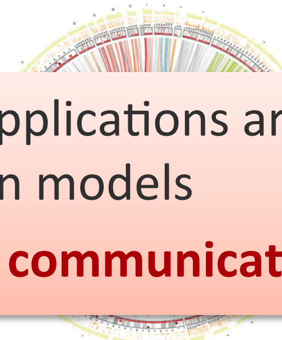
- **Regular computations**
  - Organized around dense vectors or matrices
  - **Regular data movement** pattern, use **MPI SEND/RECV or collectives**
  - More local computation, less data movement
  - Example: stencil computation, matrix multiplication, FFT*

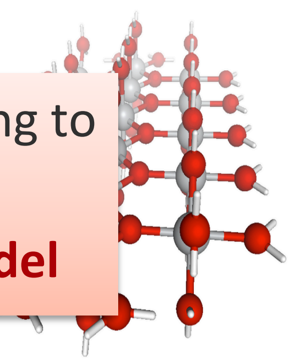
- **Irregular computations**
  - Organized around graphs, sparse vectors, more "data driven" in nature
  - Data movement pattern is **irregular and data-dependent**
  - **Growth rate of data movement is much faster than computation**
  - Example: quantum chemistry, bioinformatics







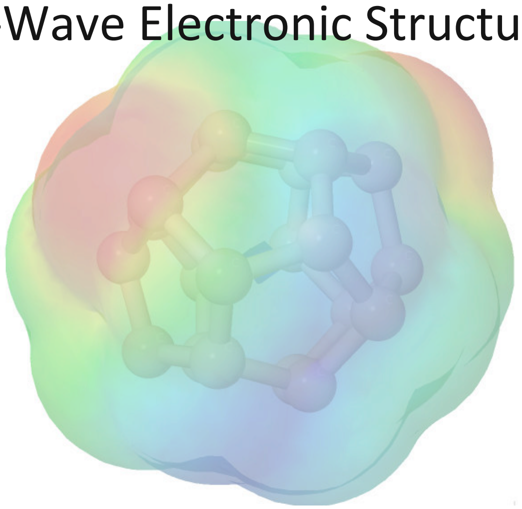


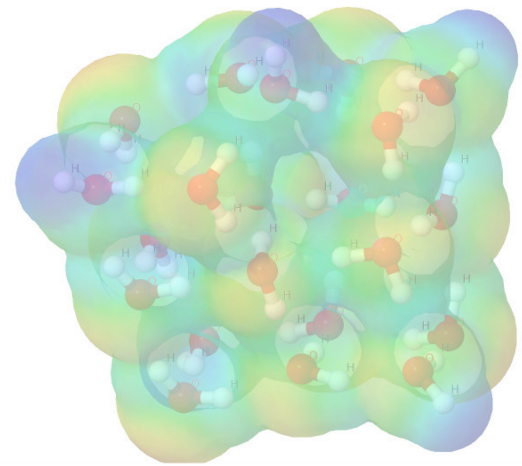
- Increasing trend of applications are moving to irregular computation models
- **Need more dynamic communication model**

* **FFT** : Fast Fourier Transform

# NWChem

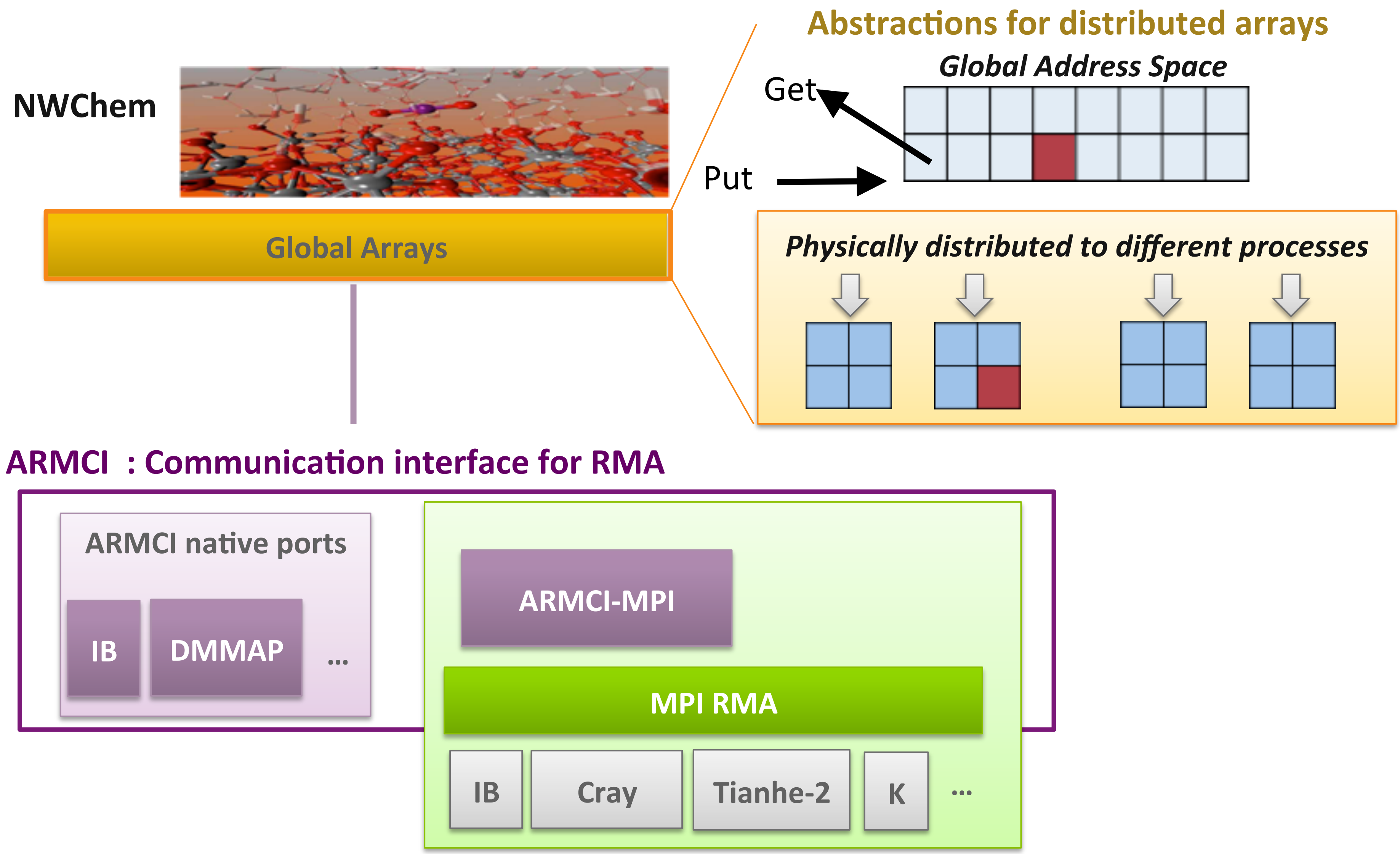
- **High performance computational chemistry application suite**
- **Composed of many types of simulation capabilities**
  - Molecular Electronic Structure
  - Quantum Mechanics/Molecular Mechanics
  - Pseudo potential Plane-Wave Electronic Structure
  - Molecular Dynamics





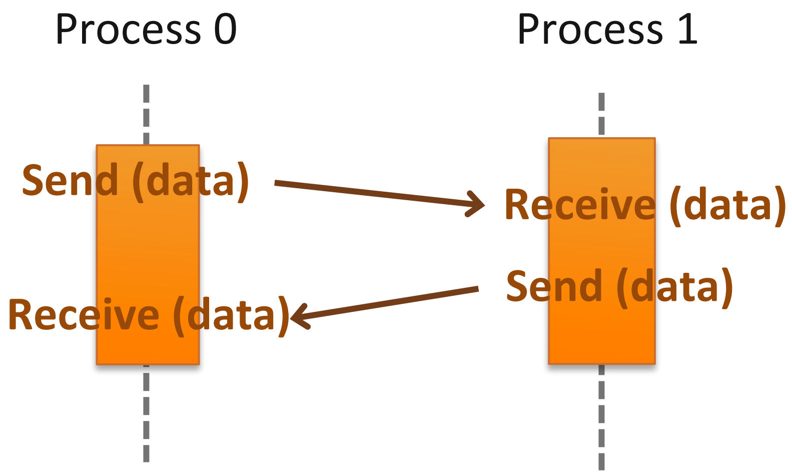
[1] M. Valiev, E.J. Bylaska, N. Govind, K. Kowalski, T.P. Straatsma, H.J.J. van Dam, D. Wang, J. Nieplocha, E. Apra, T.L. Windus, W.A. de Jong, "NWChem: a comprehensive and scalable open-source solution for large scale molecular simulations" Comput. Phys. Commun. 181, 1477 (2010)

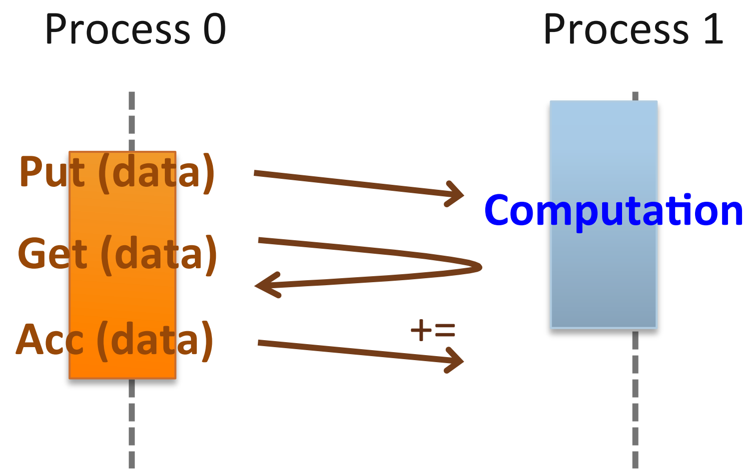
# NWChem Communication Runtime



# MPI RMA Communication

- **Two-sided communication**
- **One-sided communication (Remote Memory Access)**




**Feature:**

- Origin (P0) specifies all communication parameters
- Target (P1) does not explicitly receive or process message

**Is communication always asynchronous ?**

# Outline

- Problem Statement
- Solution
- Evaluation

**Experimental Environment**


- Cray XC30
- 2.57 Petaflops/s peak performance
- 133,824 compute cores…

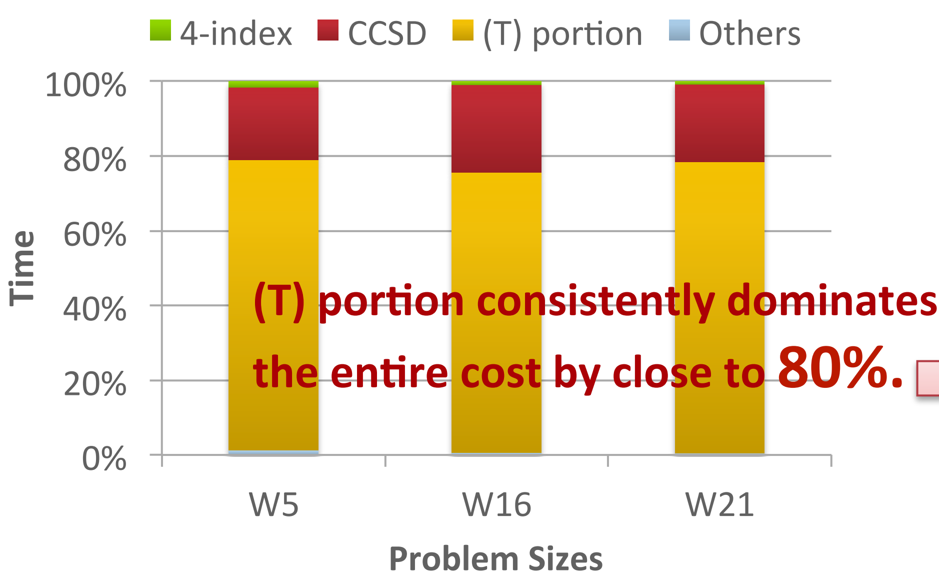
# Inefficient Communication in NWChem

- **"Gold standard" CCSD(T)**
  - Pareto optimal point of high accuracy relative to computational cost

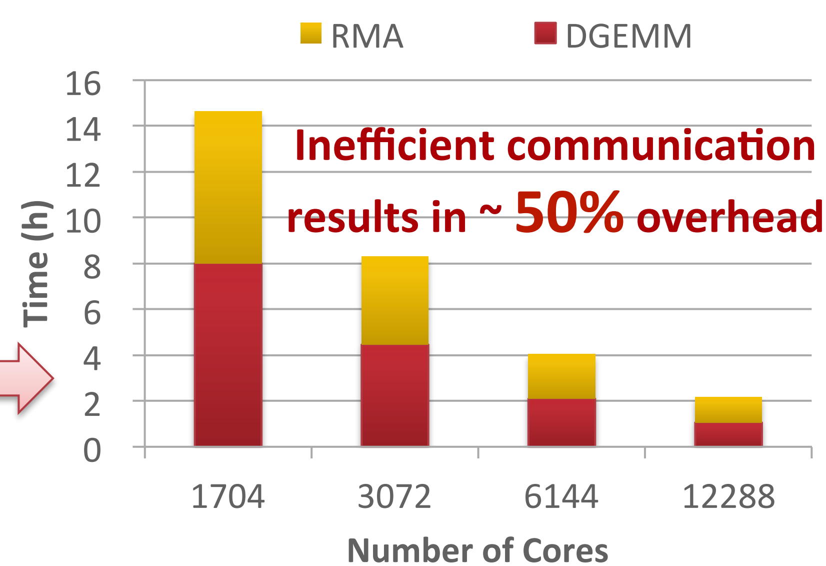
*Internal phases in CCSD(T) task*

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

*CCSD(T) internal phases in varying water problems*

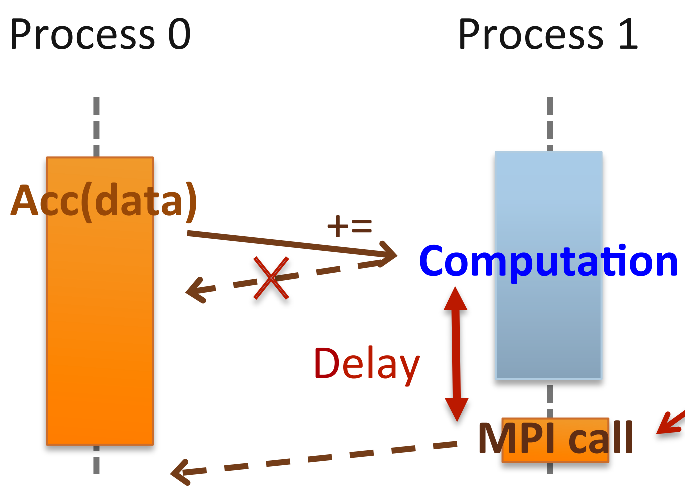



*(T) Portion profiling for $W_{21}$*



# Lack of Asynchronous Progress in MPI RMA

- **MPI one-sided operations are not truly one-sided !**
  - Some operations can be supported by hardware (e.g., PUT/GET on IB, Cray, Tofu)
  - 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 !**

*Non-contiguous Accumulate in MPI*

# Outline

- **Problem Statement**
- **Solution**
  - **Casper: Process-based asynchronous progress for MPI RMA**
- **Evaluation**

Home page: http://www.mcs.anl.gov/project/casper

[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 (SCALE Challenge Final List).

# Traditional Approaches of ASYNC Progress

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

**Cons:**

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

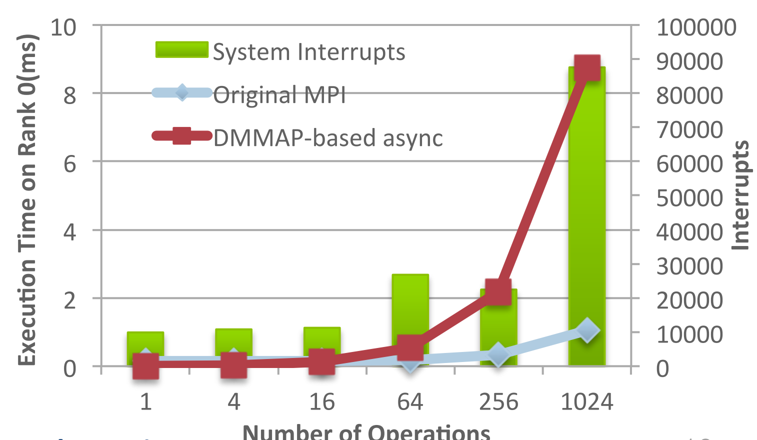
× Overhead of **Multithreading safety** of MPI


- **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**

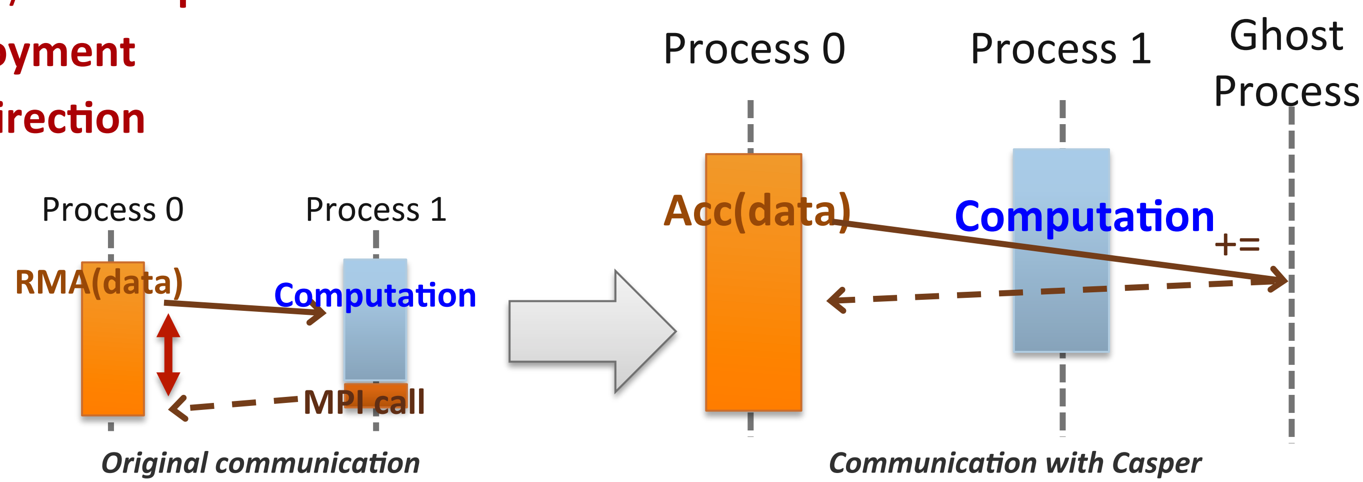
*DMMAP-based ASYNC overhead on Edison*



# [Our Solution]  Casper: Process-based ASYNC Progress

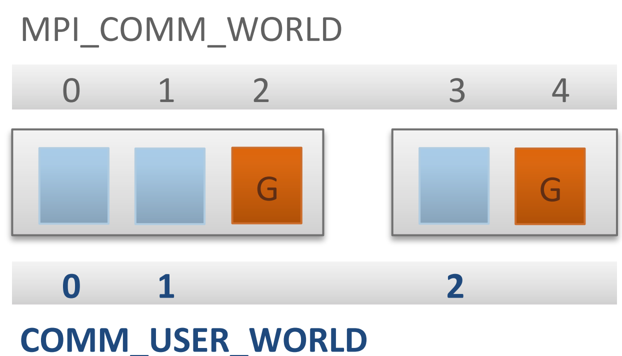

- **Multi- and many-core architectures**
  - Rapidly growing number of cores
  - **Not all of the cores are always keeping busy**
- **Casper**
  - Dedicating **arbitrary number of cores to "ghost processes"**
  - **Ghost process intercepts all RMA operations** to the user processes

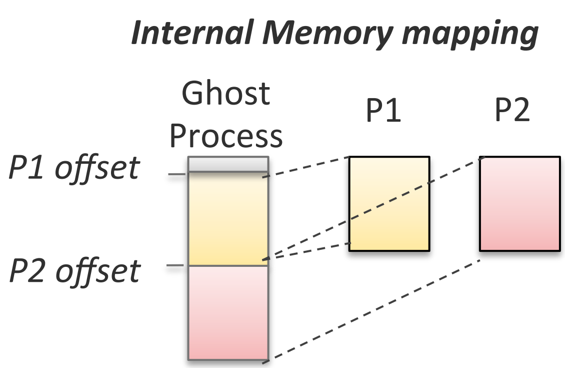
✓ **No multithreading / interrupts overhead**
✓ **Flexible core deployment**
✓ **Portable PMPI redirection**



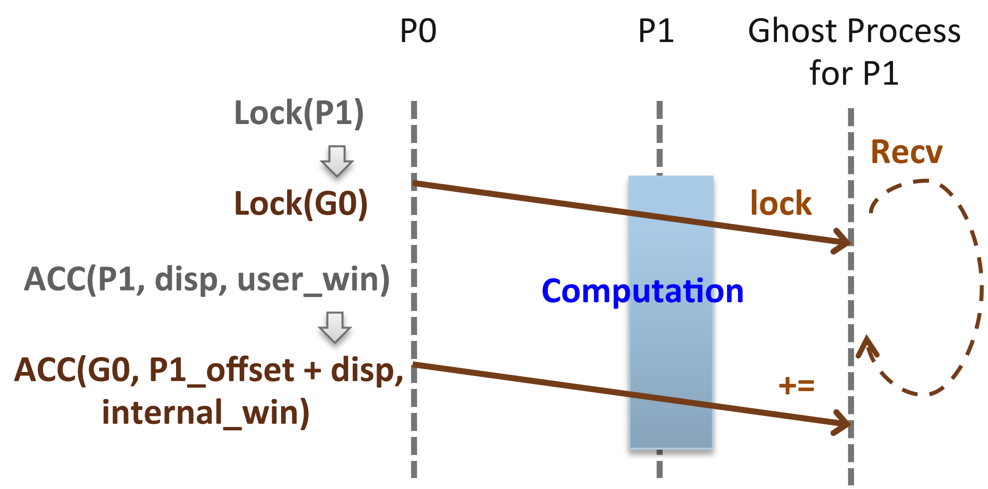
# Basic Design of Casper

- **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*.



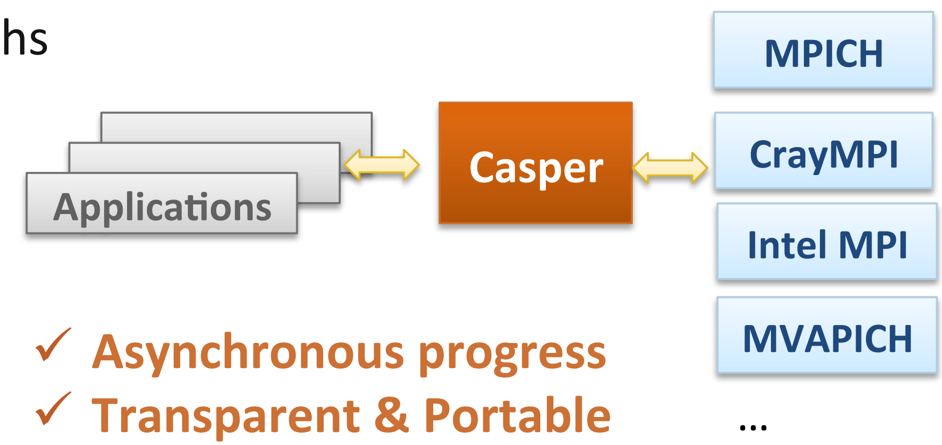

3. **Redirect RMA operations** to ghost processes




* **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 in Casper

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




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

# Outline

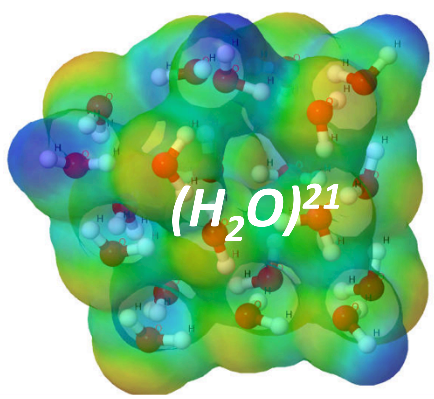
- **Problem Statement**
- **Solution**
- **Evaluation**

**Experimental Environment**


- 12-core Intel Ivy Bridge * 2 (24 cores) per node
- Cray MPI v6.3.1

# Strong Scaling of (T) Portion for W21 Problem

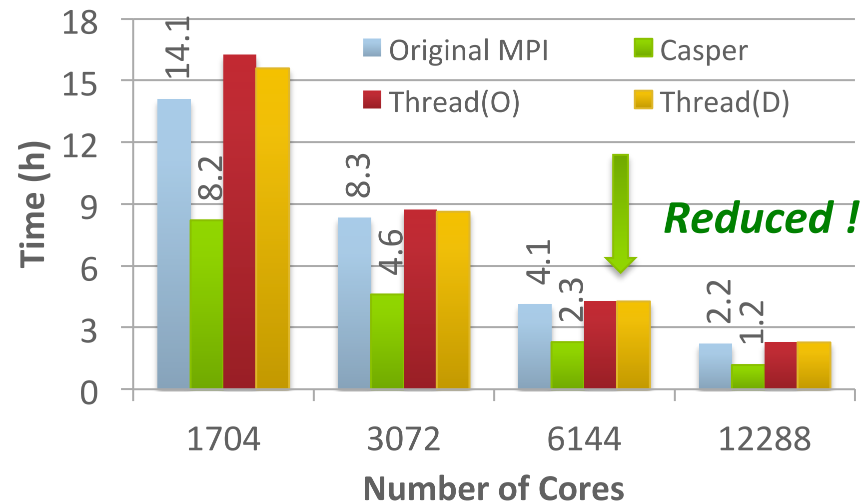
- **“Gold standard” CCSD(T)**
- **Water 21**
  - (T) portion dominates entire cost by 80%
  - Inefficient communication resulted in 50% additional overhead




*Core deployment*

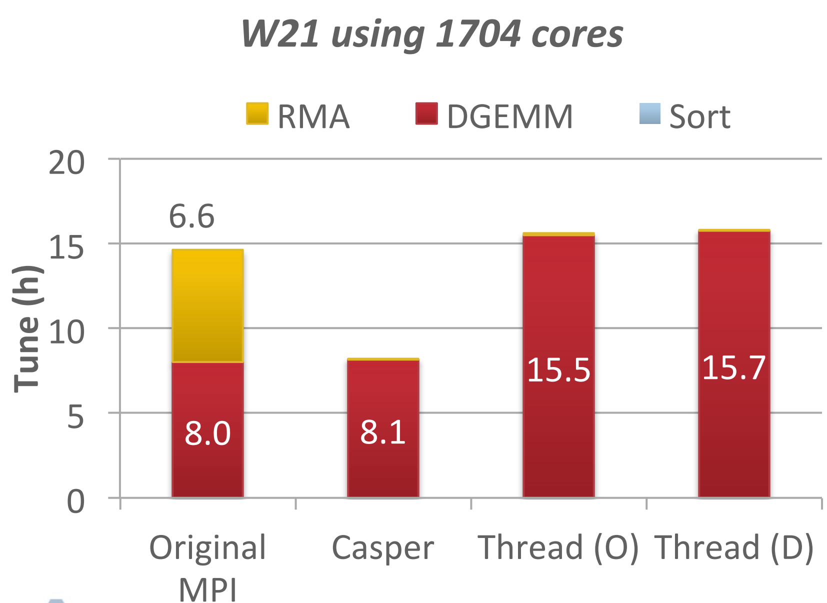
| | # COMP | # ASYNC |
|---|---|---|
| Original MPI | 24 | 0 |
| Casper | 23 | 1 |
| Thread (O) (with oversubscribed cores) | 24 | 24 |
| Thread (D) (with dedicated cores) | 12 | 12 |

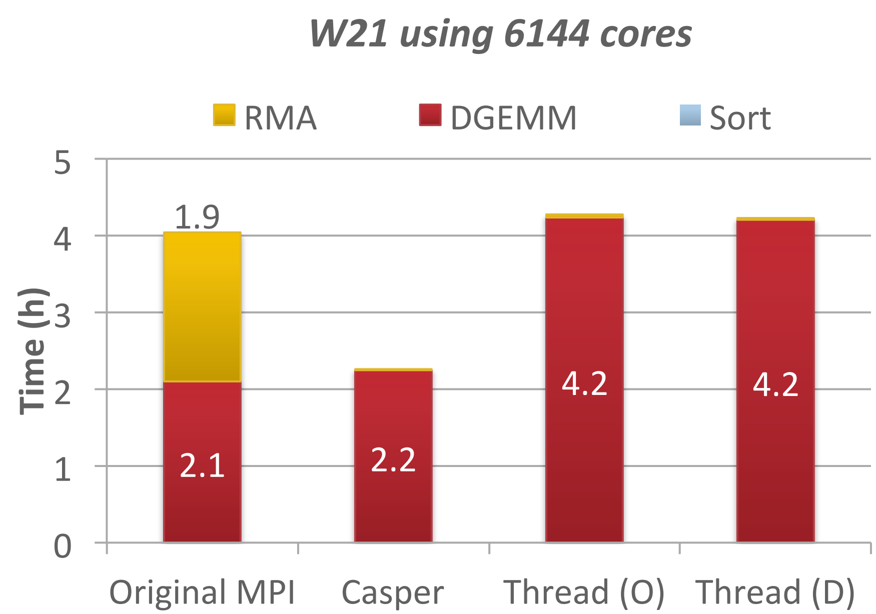
*Execution time*



# WHY Casper Improves the Performance ?

| | # COMP | # ASYNC | |
|---|---|---|---|
| Original MPI | 24 | 0 | |
| Casper | 23 | 1 | Loss only 1 (4%) COMP cores |
| Thread (O) (with oversubscribed cores) | 24 | 24 | Core oversubscription |
| Thread (D) (with dedicated cores) | 12 | 12 | Loss 50% COMP cores |



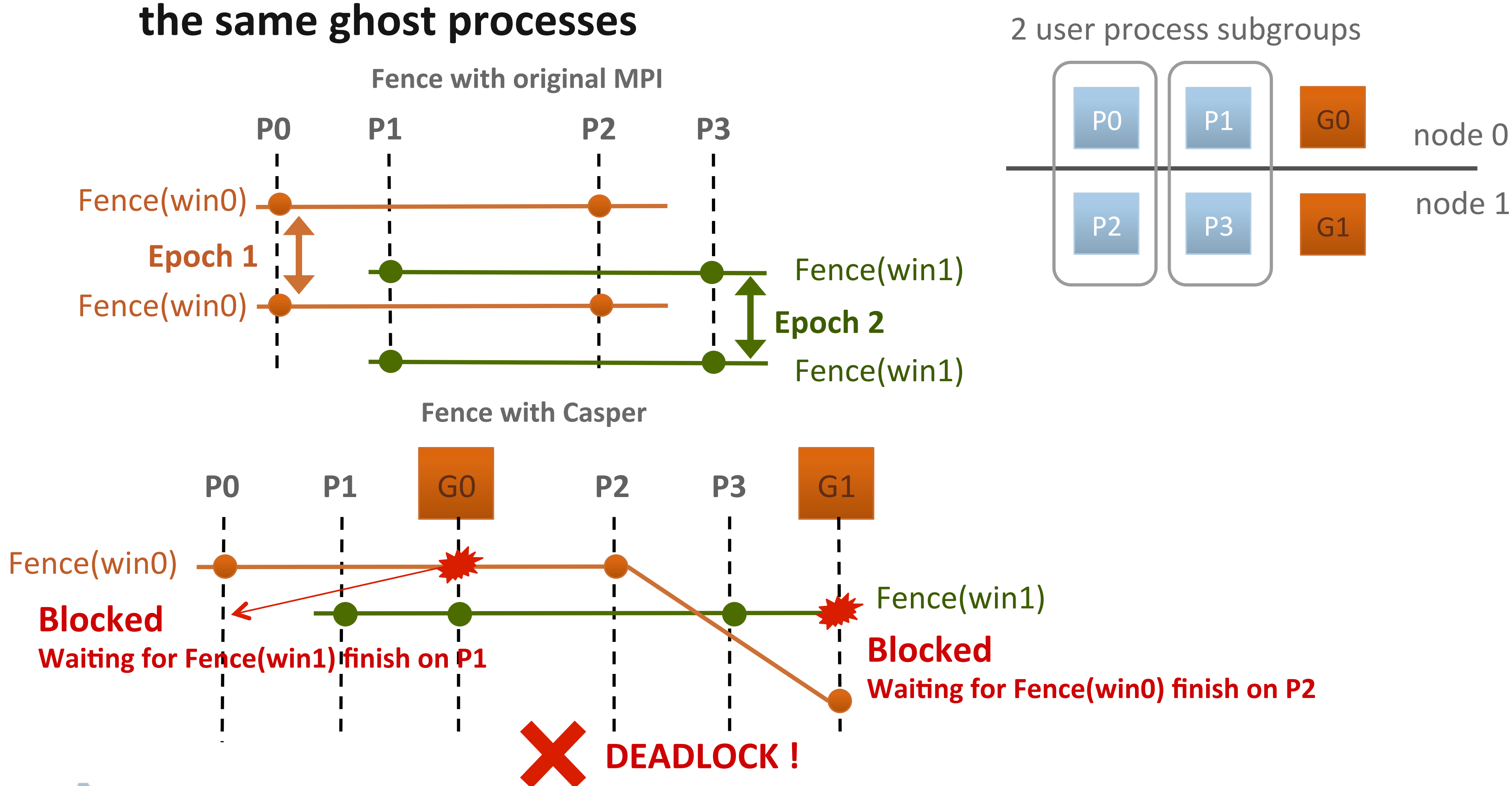
# Summary

- **MPI RMA communication is not truly one-sided**
  - Still need asynchronous progress
- **Multi-/ Many-Core architectures (e.g., NERSC Edison)**
  - Number of cores is growing rapidly, some cores are not always busy
- **Casper: a process-based asynchronous progress model**
  - **Dedicating arbitrary number of cores** to ghost processes
  - **Mapping window regions** from user processes to ghost processes
  - **Redirecting all RMA SYNC. & operations** to ghost processes
  - Linking to various MPI implementation through **PMPI transparent redirection**
- **Improved NWChem performance up to 50% on Edison**

# Backup

# A Challenge : Multiple Simultaneous Epochs (1)

- **Simultaneous fence epochs on disjoint sets of processes sharing the same ghost processes**



# Solution for Multiple Simultaneous Fence Epochs

- **Every user window creates an internal "global window"**
- **Translate to passive-target mode (lockall-flushall-unlockall)**

| Original code | Casper translated code (on user processes) |
|---|---|
| Win_allocate | Win_allocate<br>**Lock_all (global win)** |
| **Fence(win0)** | **Flush_all (global win) + Barrier(COMM_USER_WORLD) + Win_sync** |
| PUT(P) | PUT(G) |
| … | … |
| **Fence(win)** | **Flush_all (global win) + Barrier(COMM_USER_WORLD) + Win_sync** |
| | [Performance issue 1] User hint **MPI_MODE_NOPRECEDE** avoids it<br>[Performance issue 2] User hint **(NOSTORE & NOPUT & NOPRECEDE)** avoids it<br>[Performance issue 3] |
| Win_free | **Unlock_all (global win)**<br>Win_free |