ARGOBOTS

A Lightweight Low-level Threading/Tasking Framework

https://collab.cels.anl.gov/display/ARGOBOTS/

Pavan Balaji (ANL) and L.V. (Sanjay) Kale (UIUC)

Argo Concurrency Team

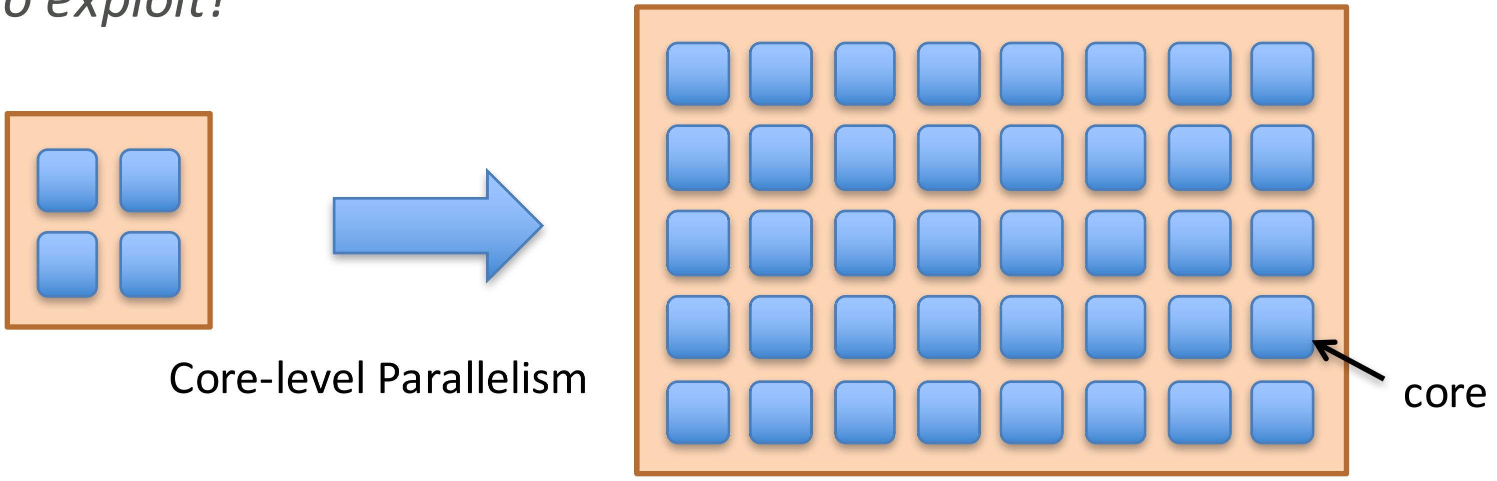
- Argonne National Laboratory (ANL)
  - Pavan Balaji (co-lead)
  - Sangmin Seo
  - Abdelhalim Amer
  - Marc Snir
  - Pete Beckman (PI)
- University of Illinois at Urbana-Champaign (UIUC)
  - Laxmikant Kale (co-lead)
  - Nitin Kundapur Bhat
  - Prateek Jindal
- University of Tennessee, Knoxville (UTK)
  - George Bosilca
  - Thomas Herault
  - Damien Genet
- Pacific Northwest National Laboratory (PNNL)
  - Sriram Krishnamoorthy

Past Team Members:

- Cyril Bordage (UIUC)
- Esteban Meneses (University of Pittsburgh)
- Huiwei Lu (ANL)
- Yanhua Sun (UIUC)
- Jonathan Lifflander (UIUC)

Massive On-node Parallelism

- The number of cores is increasing
- Massive on-node parallelism is inevitable
- Existing solutions do not effectively deal with such parallelism with respect to on-node threading/tasking systems or with respect to off-node communication in the presence of such tasks/threads
- *How to exploit?*



Goal: Provide runtime systems utilizing massive on-node parallelism

Argo Concurrency: Approaches

High-level tasking models (CilkBots, TASCEL, PaRSEC)

- Explore high-level tasking frameworks that will take advantage of the low-level threading and communication frameworks
- CilkBots and TASCEL: PNNL & UIUC
- PaRSEC: UTK

High-level dynamic execution environments (Charm++,..)

- Investigate high-level programming models, e.g., Charm++, that are specialized in dynamic execution environments and can exploit the low-level threading and communication frameworks
- UIUC

Argobots interoperability with a PUT/GET model

- Design new communication libraries that work with such low-level threading models
- Argonne

Argobots: A lightweight low-level threading/tasking framework

- Develop a new low-level threading/tasking model that will allow us to expose the hardware characteristics of exascale computing systems more effectively
- Argonne & UIUC

Argobots

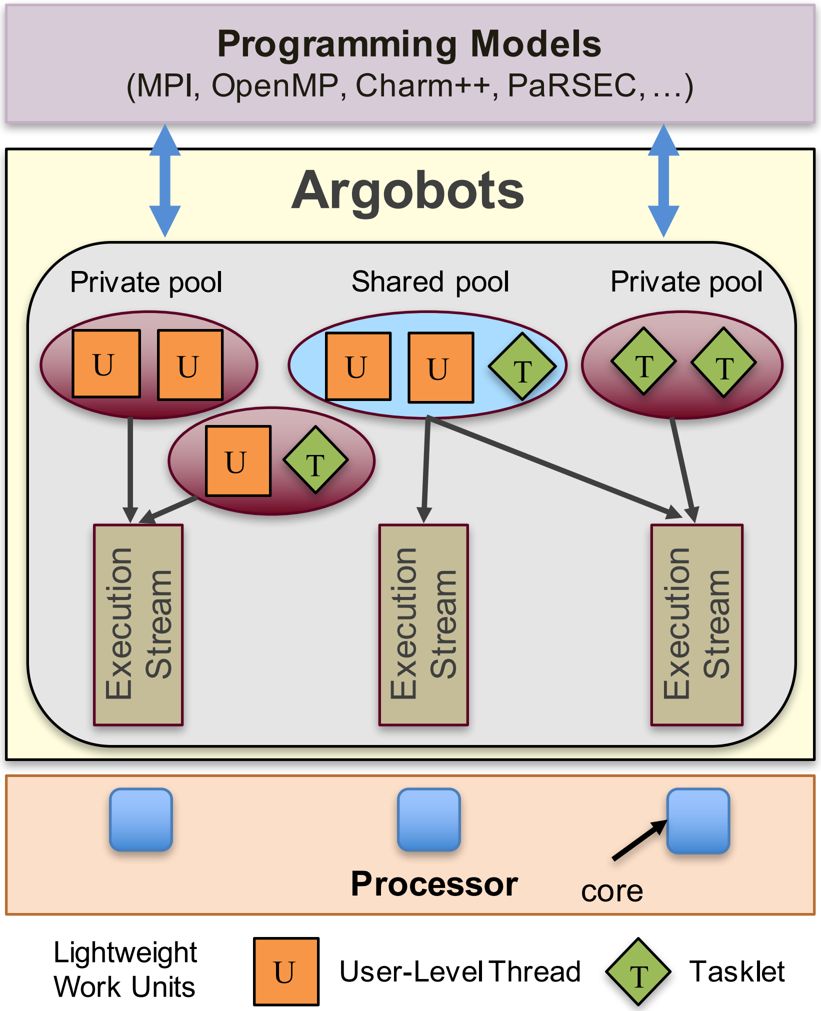
A low-level lightweight threading and tasking framework
(http://collab.cels.anl.gov/display/argobots/)

Overview

- Separation of mechanisms and policies
- Massive parallelism
  - **Exec. Streams** guarantee progress
  - **Work Units** execute to completion
    - User-level threads (ULTs) vs. Tasklet
- Clearly defined memory semantics
  - Consistency domains
    - Provide Eventual Consistency
  - Software can manage consistency

Argobots Innovations

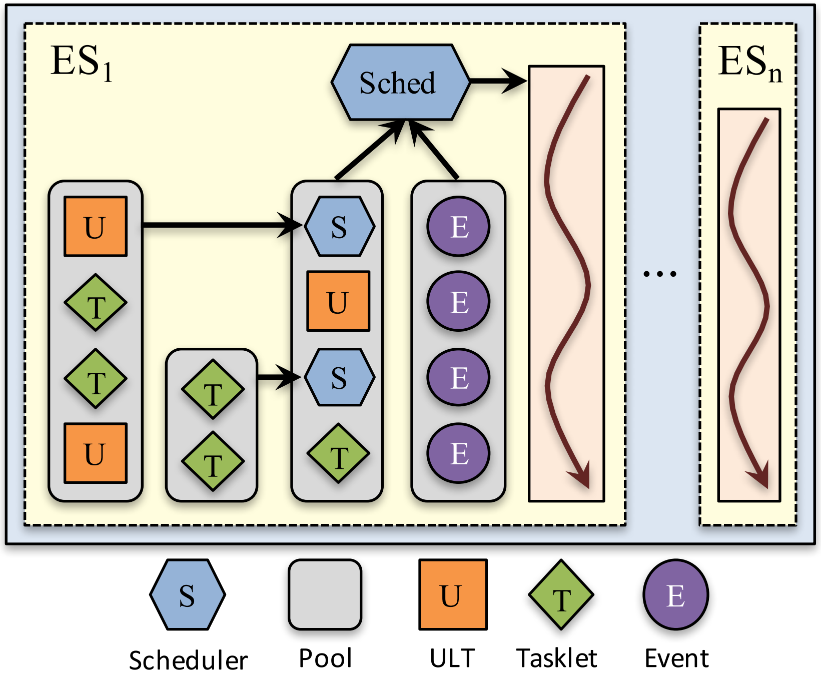
- **Enabling technology, but not a policy maker**
  - High-level languages/libraries such as OpenMP, Charm++ have more information about the user application (data locality, dependencies)
- **Explicit model**:
  - Enables dynamism, but always managed by high-level systems



* Current team members: Pavan Balaji, Sangmin Seo, Halim Amer (ANL), L. Kale, Nitin Bhat, Prateek Jindal (UIUC)

Argobots Execution Model

- **Execution Streams (ES)**
  - Sequential instruction stream
    - Can consist of one or more work units
  - Mapped efficiently to a hardware resource
  - Implicitly managed progress semantics
    - One blocked ES cannot block other ESs
- **User-level Threads (ULTs)**
  - Independent execution units in user space
  - Associated with an ES when running
  - Yieldable and migratable
  - Can make blocking calls
- **Tasklets**
  - Atomic units of work
  - Asynchronous completion via notifications
  - Not yieldable, migratable before execution
  - Cannot make blocking calls

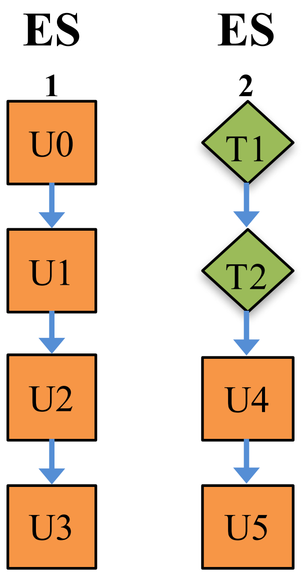


Argobots Execution Model

- **Scheduler**
  - Stackable scheduler with pluggable strategies
- **Synchronization primitives**
  - Mutex, condition variable, barrier, future
- **Events**
  - Communication triggers

Explicit Mapping ULT/Tasklet to ES

- The user needs to map work units to ESs
- No smart scheduling, no work-stealing unless the user wants to use

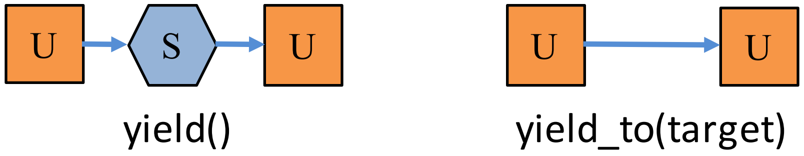


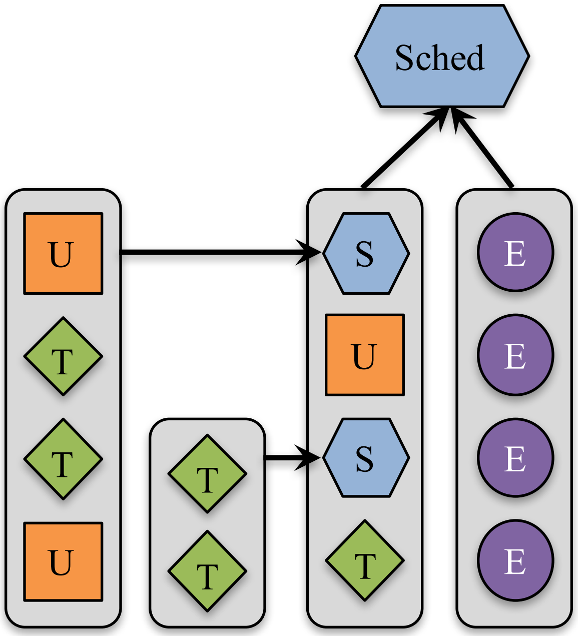
- Benefits
  - Allow locality optimization
    - Execute work units on the same ES
  - No expensive lock is needed between ULTs on the same ES
    - They do not run concurrently
    - A flag is enough

Optional Advanced features: shared and global workpools

Stackable Scheduler with Pluggable Strategies

- Associated with an ES
- Can handle ULTs and tasklets
- *Can handle schedulers*
  - Allows to stack schedulers hierarchically
- Can handle asynchronous events
- *Users can write schedulers*
  - Provides **mechanisms**, not policies
  - Replace the default scheduler
    - E.g., FIFO, LIFO, Priority Queue, etc.
- ULT can explicitly *yield to* another ULT
  - Avoid scheduler overhead

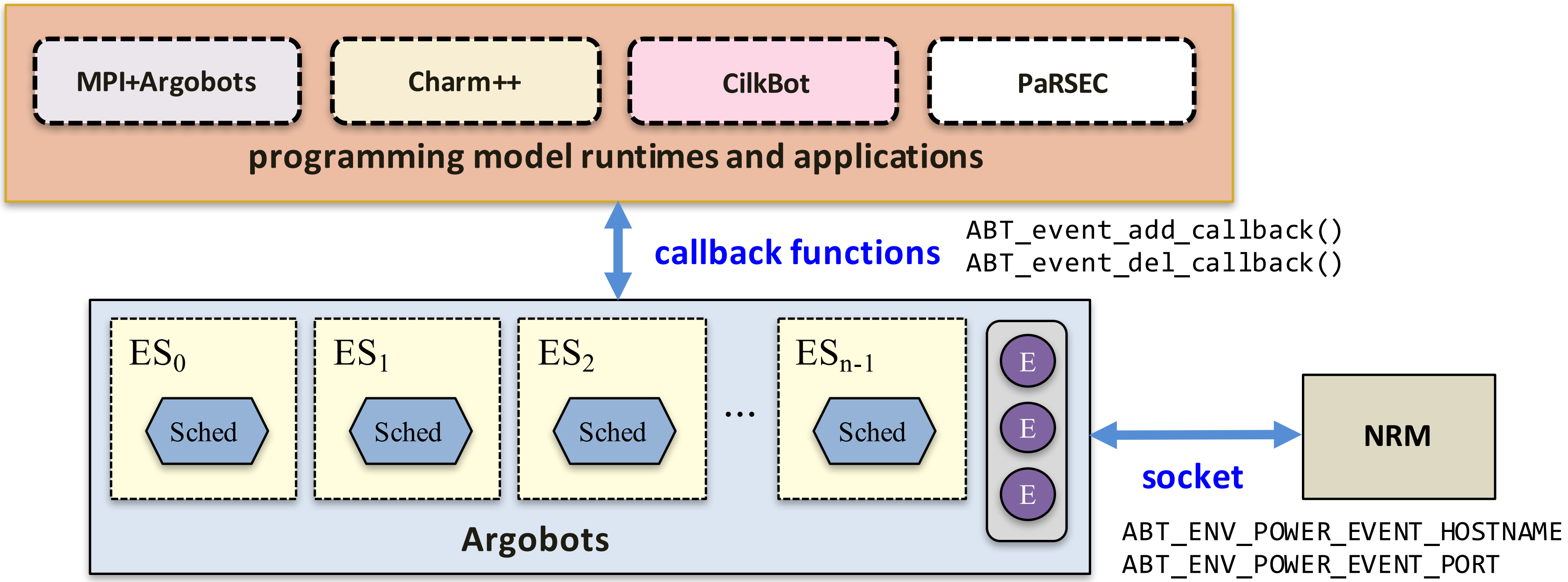




Some Capabilities of Argobots

- Shrinking the set of ESs used
  - E.g. for power/energy optimization
- Expanding the set of ESs used
- Migration of tasks and ULTs
  - Load balancing
  - Locality
  - NUMA issues

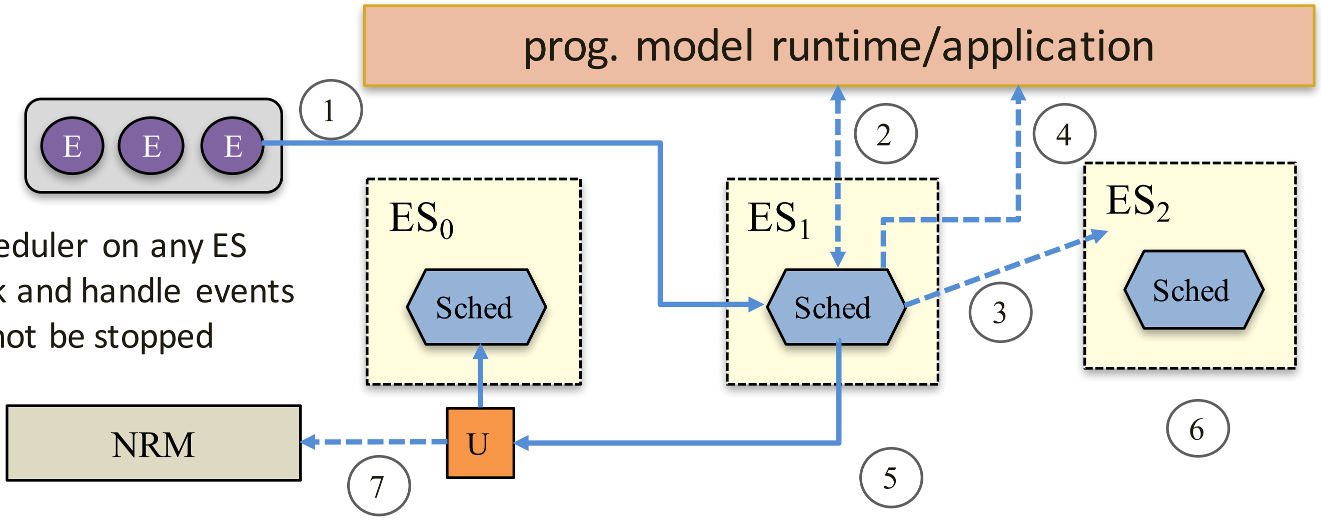
Interfaces for Shrink/Expand Events



1. [Argobots] Connect to NRM using a socket on ABT_init()
2. [Runtimes/applications] Register callback functions for shrink/expand events
3. [Runtimes/applications] Deregister callback functions when they terminate
4. [Argobots] Disconnect from NRM on ABT_finalize()

Shrink/Expand Event Handling

- Shrinking

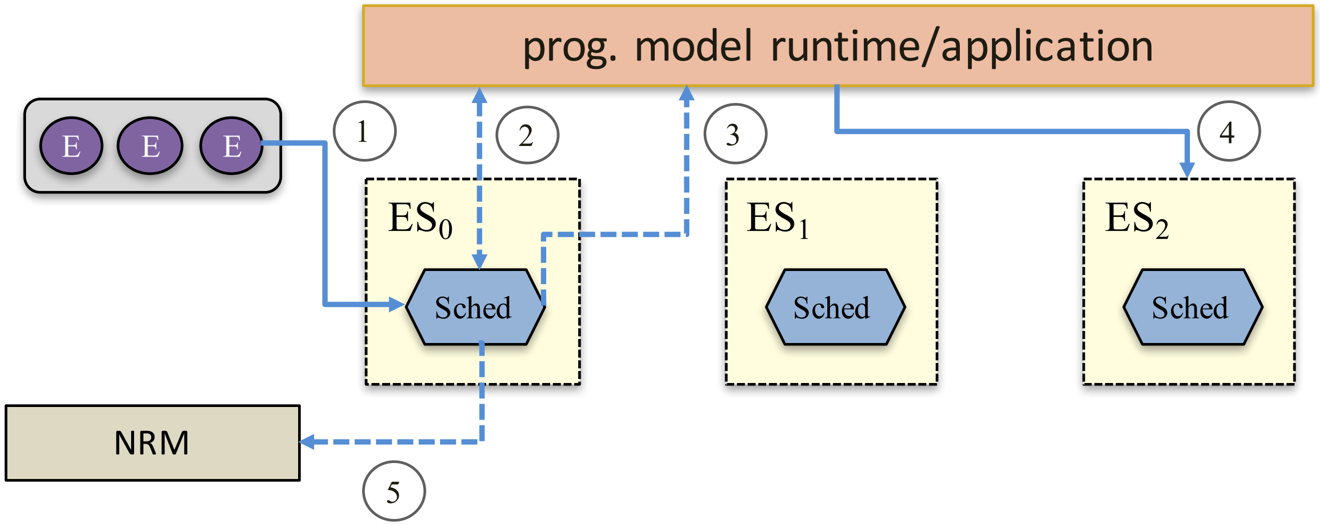


- Any scheduler on any ES can check and handle events
- ES$_0$ cannot be stopped

1. ES$_1$ picks an event, which requests ES$_2$ to be stopped
2. Ask the runtime using callbacks whether ES$_2$ can be stopped
3. If OK, mark ES$_2$ to need to stop so when the scheduler on ES$_2$ checks events, it can be stopped
4. Notify the runtime that ES$_2$ will be stopped
5. Create a ULT on ES$_0$
6. When the scheduler on ES$_2$ stops, ES$_2$ is terminated
7. After ES$_2$ is terminated, the ULT frees ES$_2$ and sends a response to NRM

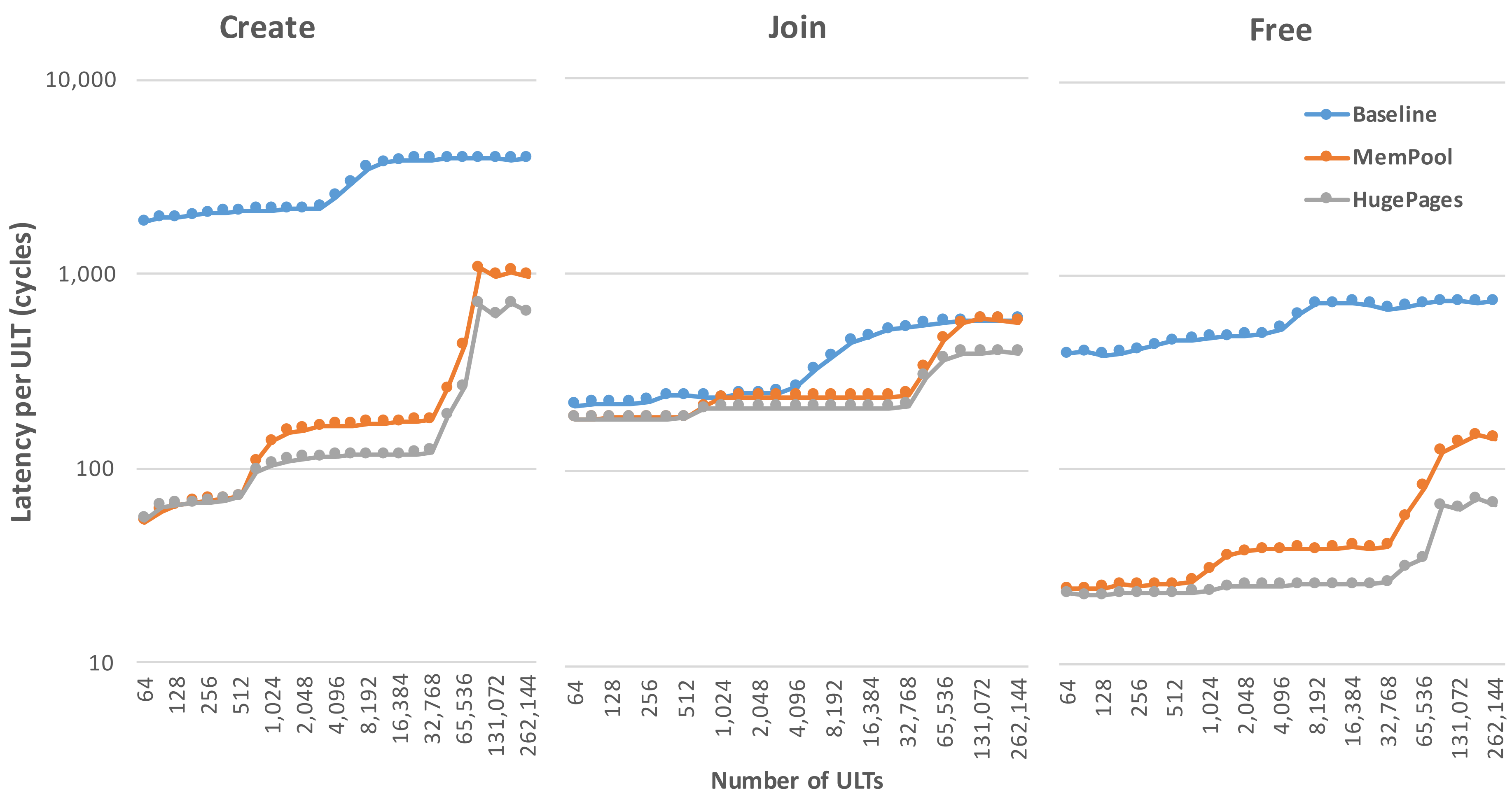
Shrink/Expand Event Handling

- Expanding

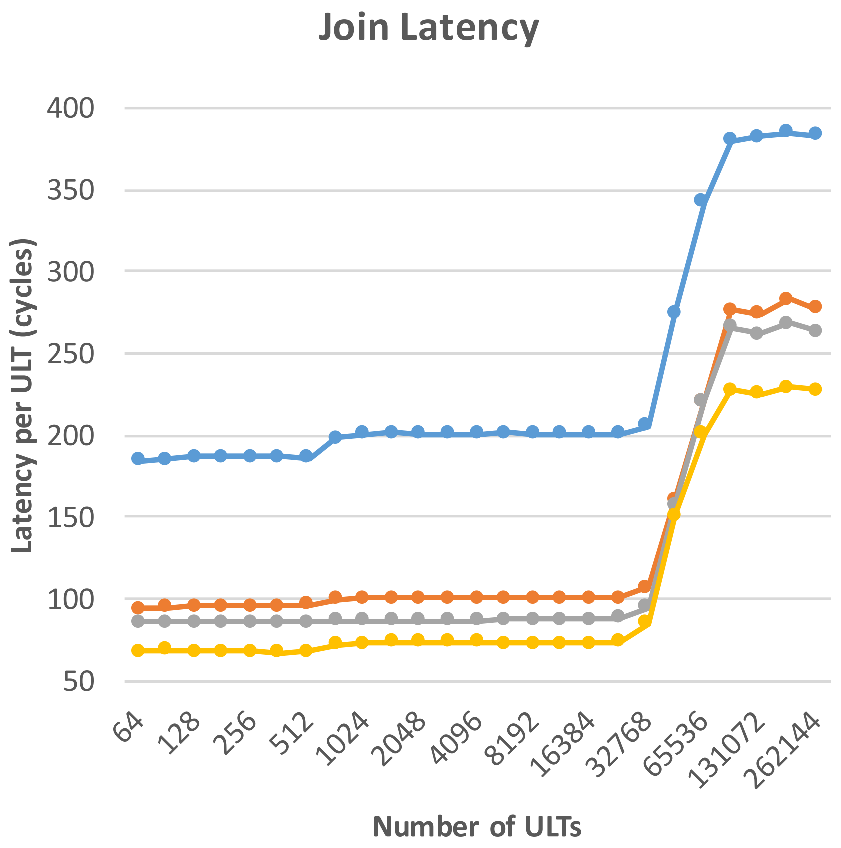


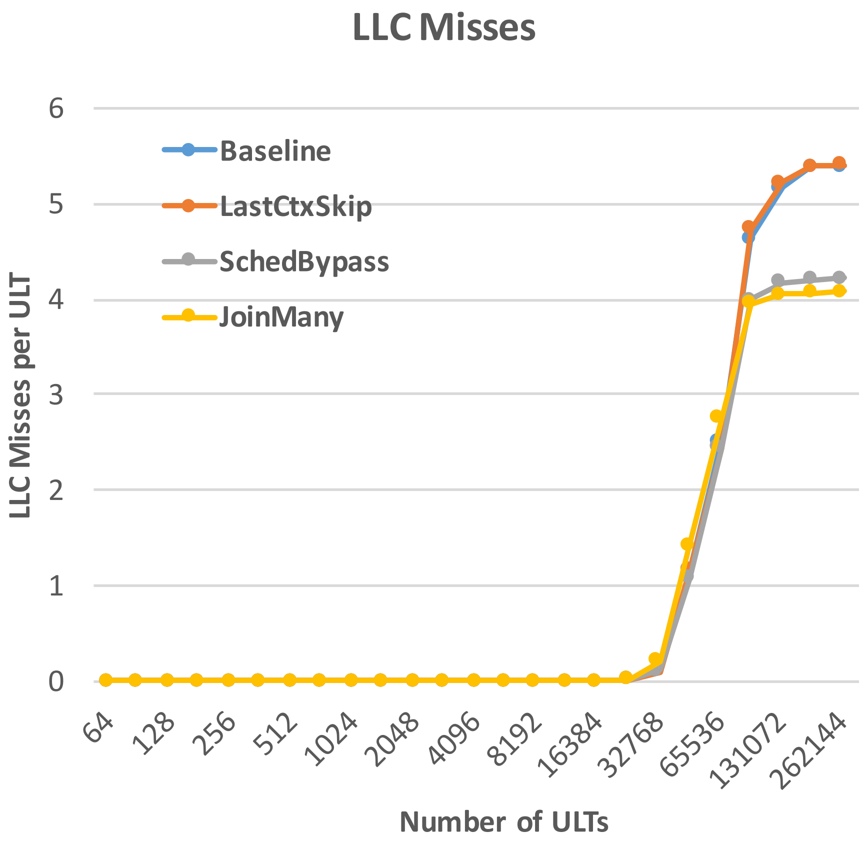
1. ES_0 picks an event, which requests to create an ES_2
2. Ask the runtime using callbacks whether it can create ES_2
3. If OK, invoke a callback function so the runtime creates ES_2
4. Create ES_2
5. Send a response to NRM

Optimization: Memory Pool & Huge Pages

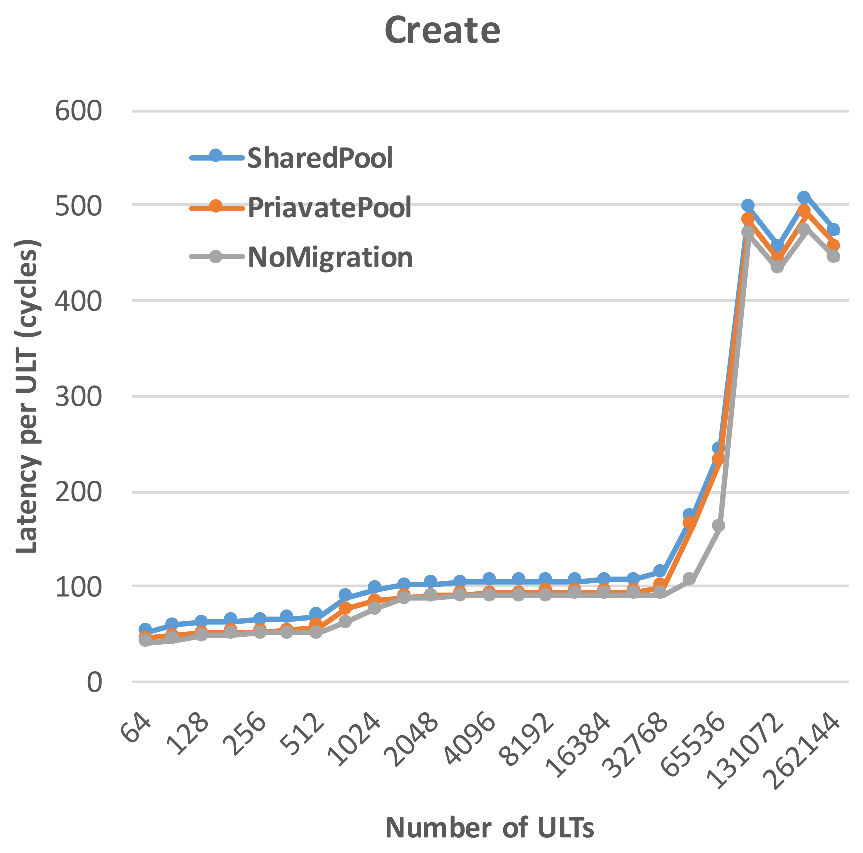


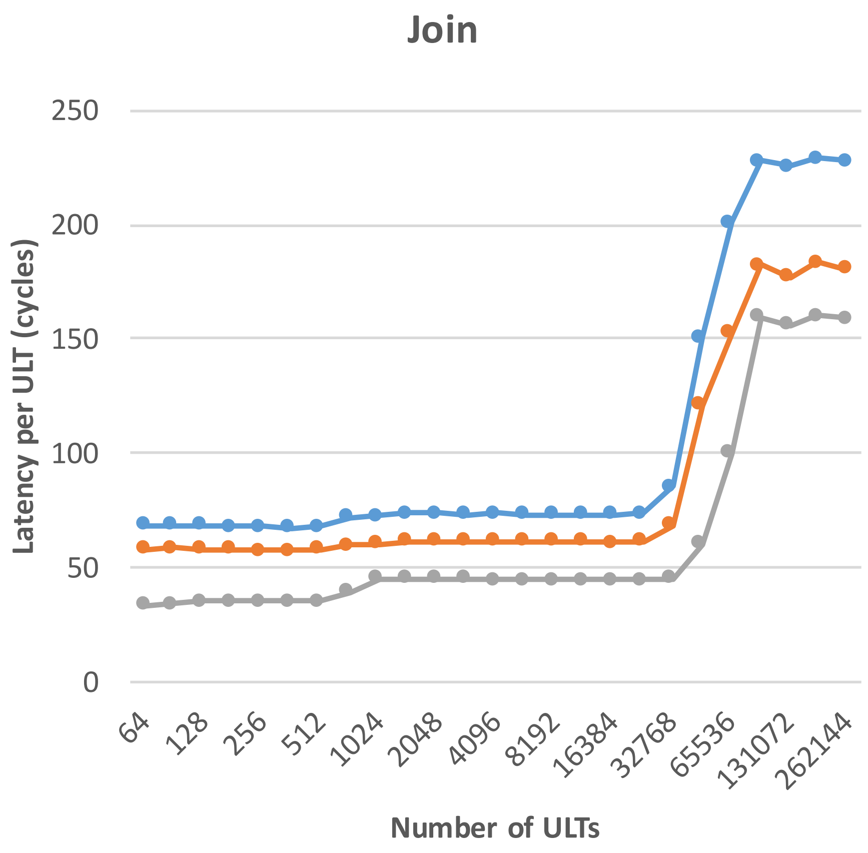
Optimization: Join Operation



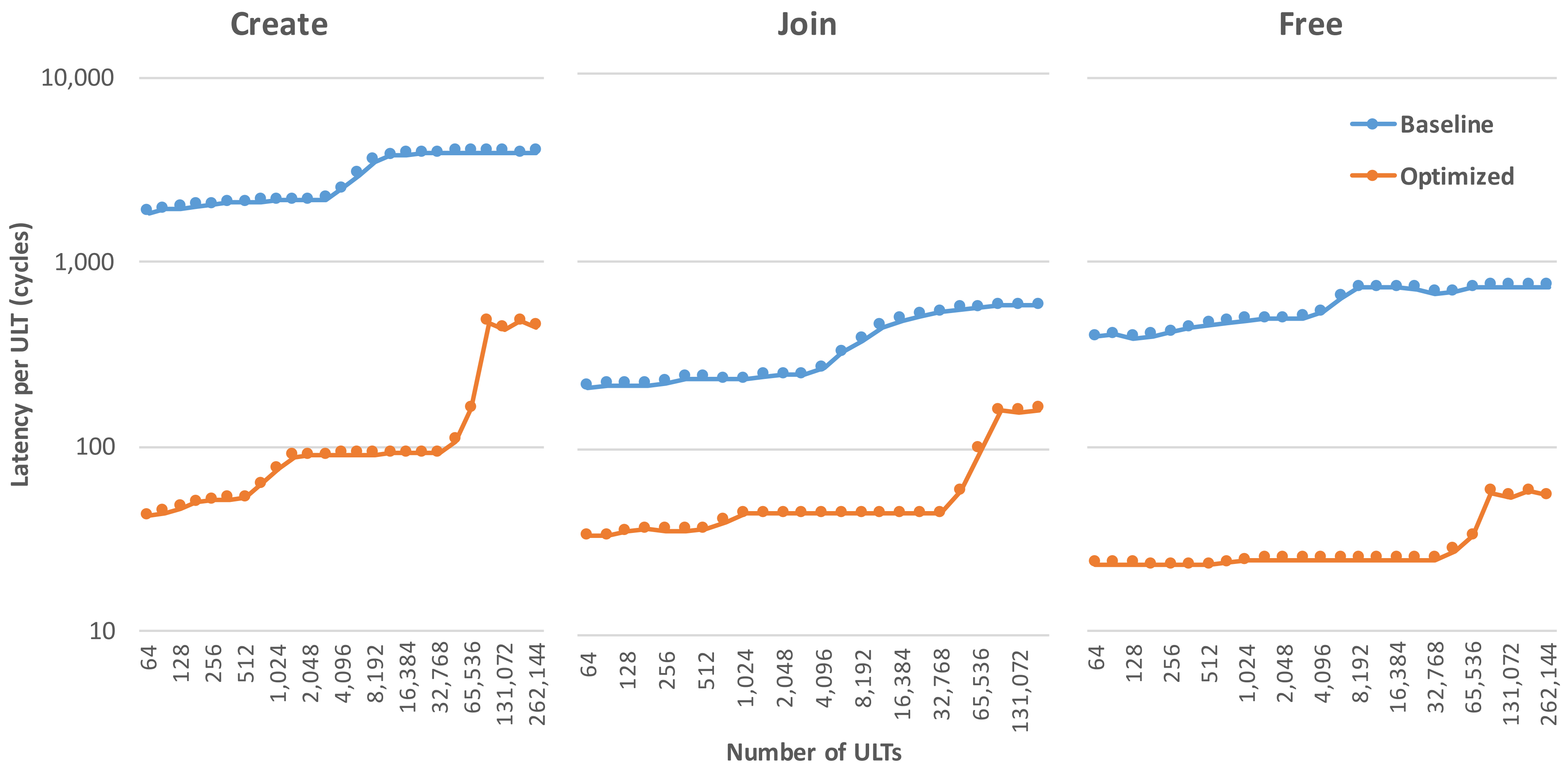


Optimization: Private Pool and Disabling Features





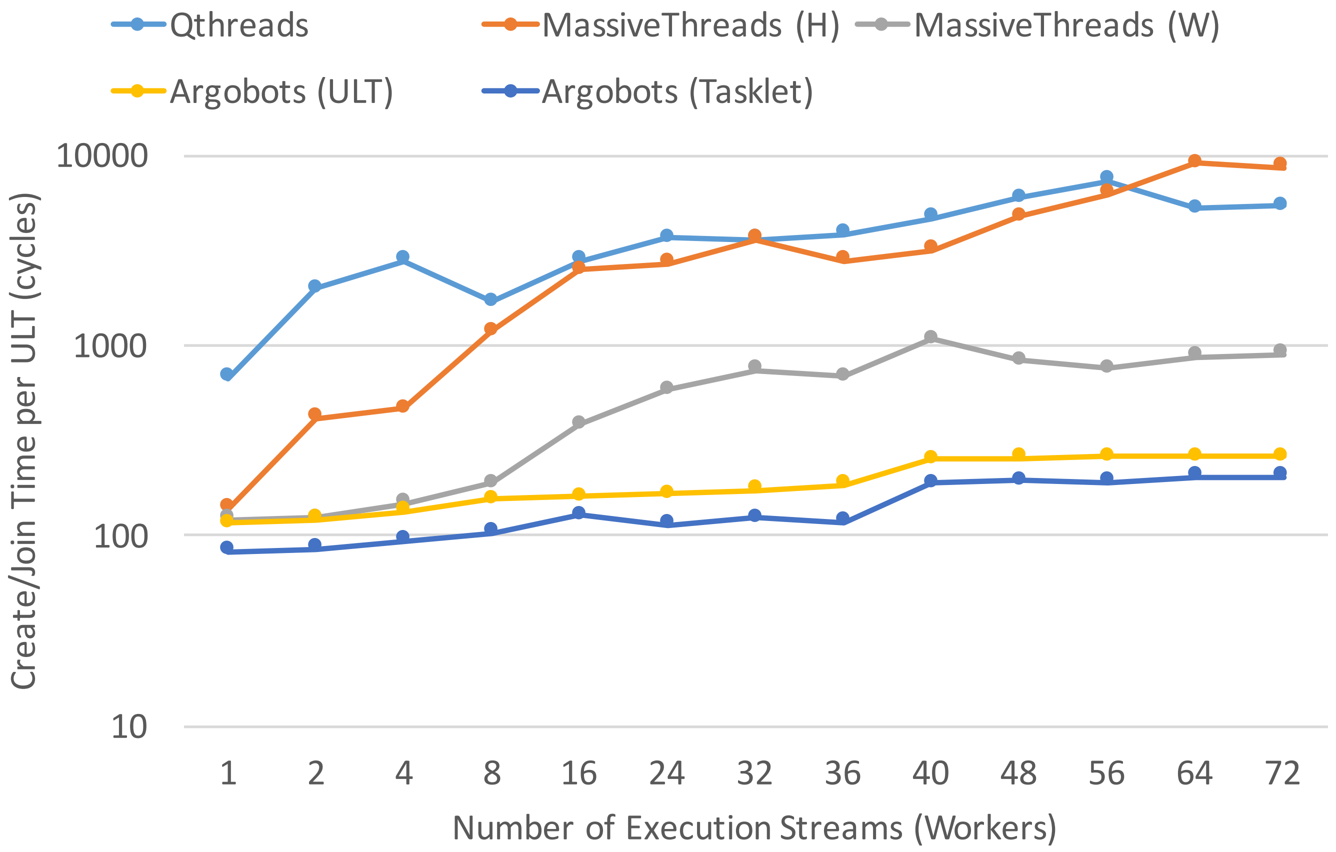
Optimization: Putting It All Together



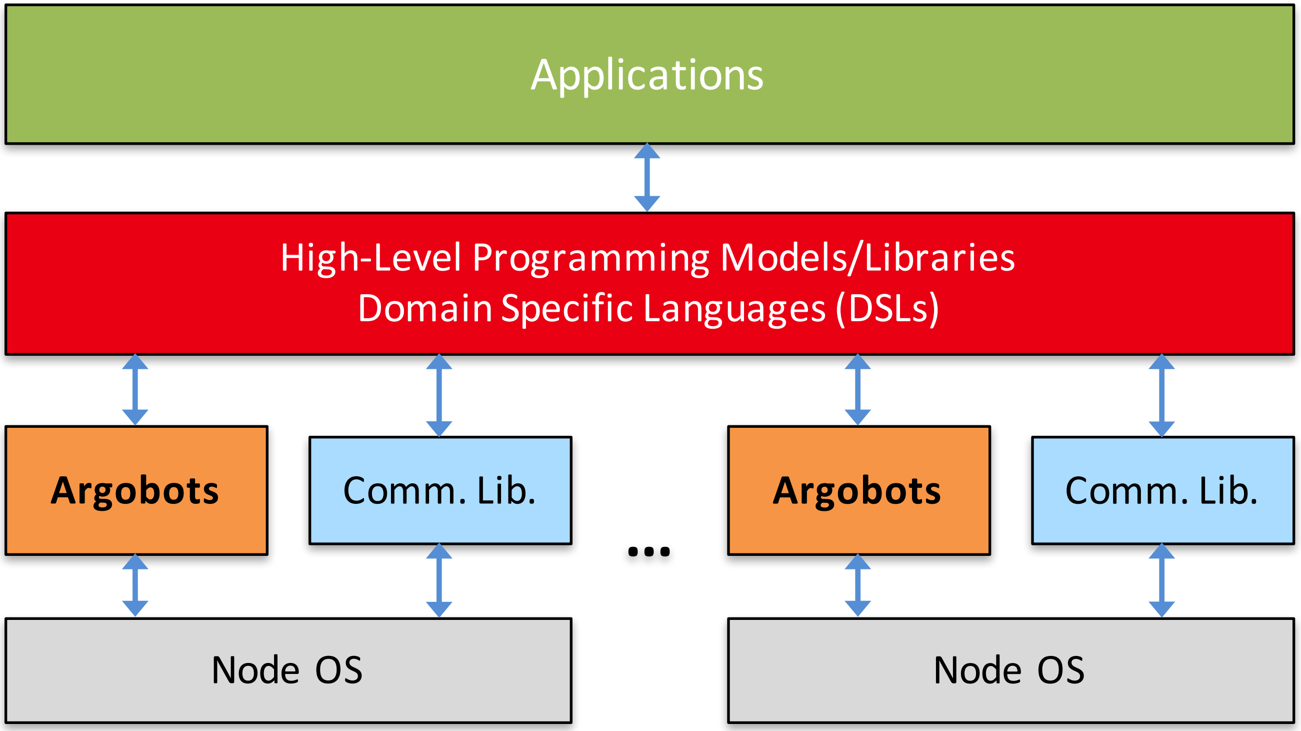
- 64 ULTs : 2,443 cycles (c: 1,837, j: 212, f: 394) -> 99 cycles (c: 42, j: 34, f: 23) **24.7x speedup**
- 262,144 ULTs: 5,212 cycles (c: 3,921, j: 558, f: 733) -> 659 cycles (c: 446, j: 159, f: 54) **7.9x speedup**

Performance: Create/Join Time

- Ideal scalability
  - If the ULT runtime is perfectly scalable, the time should be the same regardless of the number of ESs

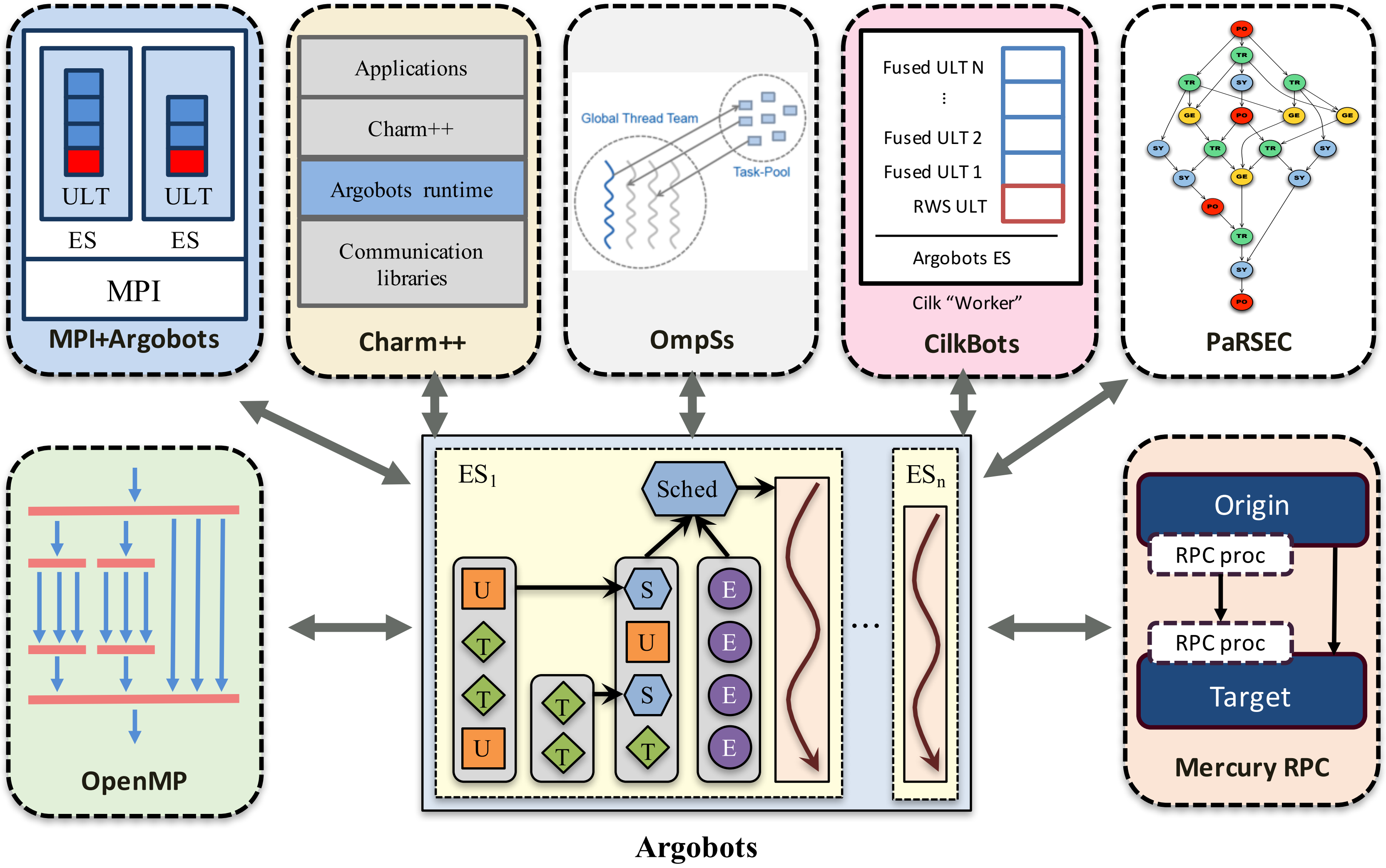


Argobots' Position


Applications
High-Level Programming Models/Libraries
Domain Specific Languages (DSLs)
Argobots
Comm. Lib.
…
Argobots
Comm. Lib.
Node OS
Node OS

Argobots is a low-level threading/tasking runtime!

Argobots Ecosystem



External Connections

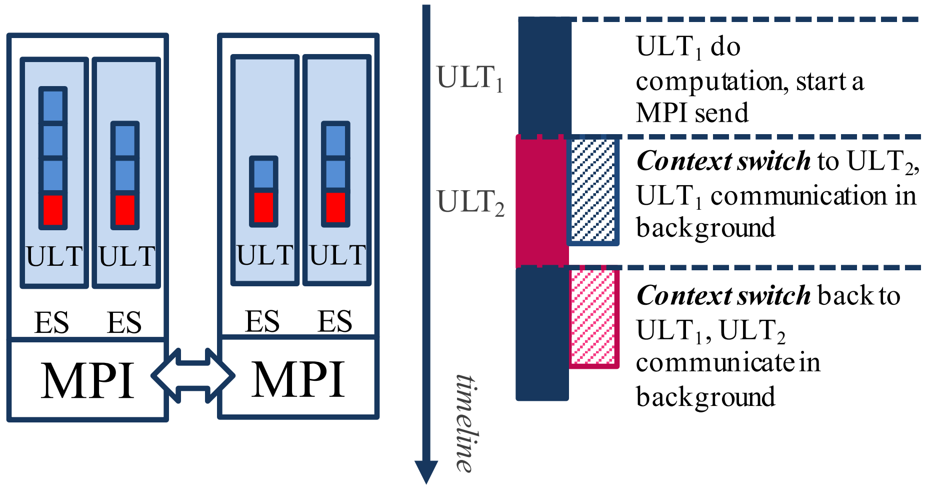
GridFTP, Kokkos, RAJA, ROSE, TASCEL, XMP, etc.

Argobots-Aware MPI Runtime

- Problem
  - Traditional MPI implementations are only aware of kernel threads
  - Thread-synchronization costly to ensure thread-safety and progress requirement from MPI
  - Wasted resources if a kernel thread blocks for MPI communication
- Solution
  - An MPI implementation aware of Argobots threads
  - Lightweight context switching to overlap costly blocking operations (communication, locks, etc.)
  - Reduced thread-synchronization opportunities (guaranteed consistency within an ES without locks or memory barriers)

Contact:
- Pavan Balaji balaji@anl.gov
- Abdelhalim Amer aamer@anl.gov
- Sangmin Seo sseo@anl.gov

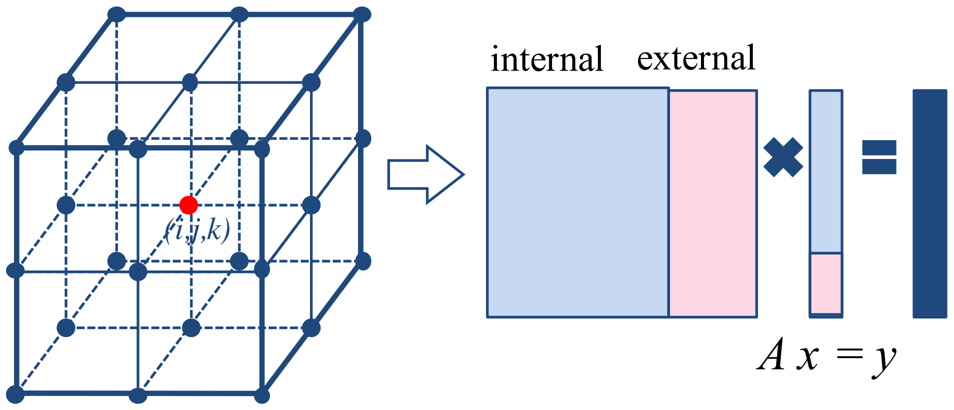
- Recent results
  - Developed an MPICH+Argobots prototype
  - Demonstrated the ability to overlap blocking communication with HPCG, SpMV, etc.
  - Deployed successfully a fully threaded Graph500 benchmark implementation
- Impact/Potential
  - The new MPI+Argobots model has the potential to overcome the long lasting multithreaded MPI communication challenge


ULT
ULT
ULT
ULT
ES
ES
ES
ES
MPI
MPI
ULT_1
ULT_2
timeline
ULT_1 do computation, start a MPI send
Context switch to ULT_2, ULT_1 communication in background
Context switch back to ULT_1, ULT_2 communicate in background

MPI+Argobots Execution Model

Application: HPCG

- High Performance Conjugate Gradient (HPCG)
  - Solves $Ax=b$, large and sparse matrix
- Hiding **Global Collective Communication**
  - at end of each iteration, do DDOT to calculate tolerance (vector multiplication + MPI_Allreduce)
  - overlap communication and computation (overlap DDOT with MG in the next iteration)
  - fork a ULT to do ult_ddot and join in the next iteration

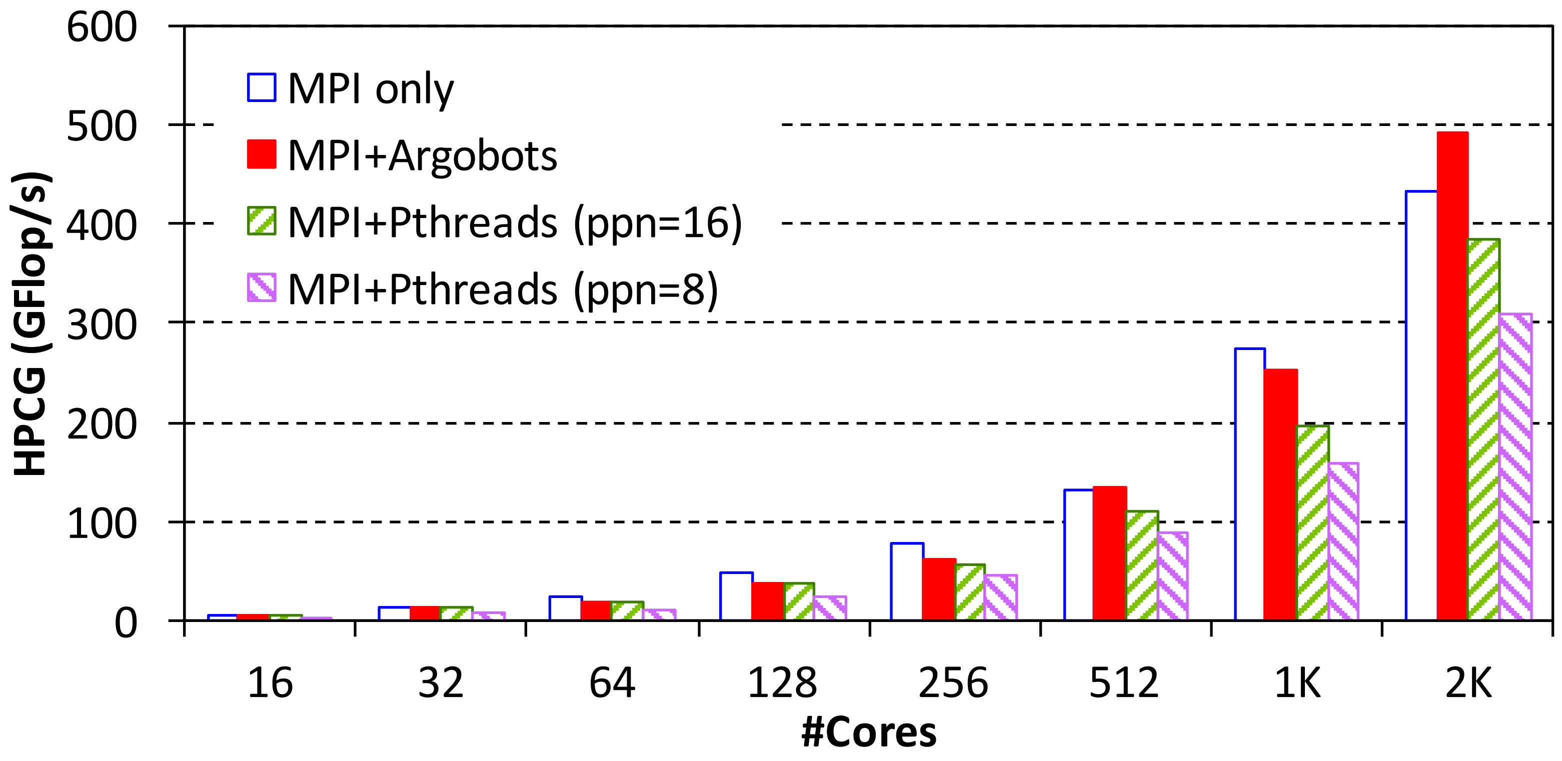


HPCG

```
for k = [1: max_iter]:
    MG(A, r, z);
    if k > 1:
        ult_join (thread);
        if (normr <= tolerance) break;
    ……
    ult_fork(ult_ddot, &param, &thread)
```

MG: preconditioner of CG
DDOT: dot product follow by MPI_Allreduce

Preliminary Results: HPCG



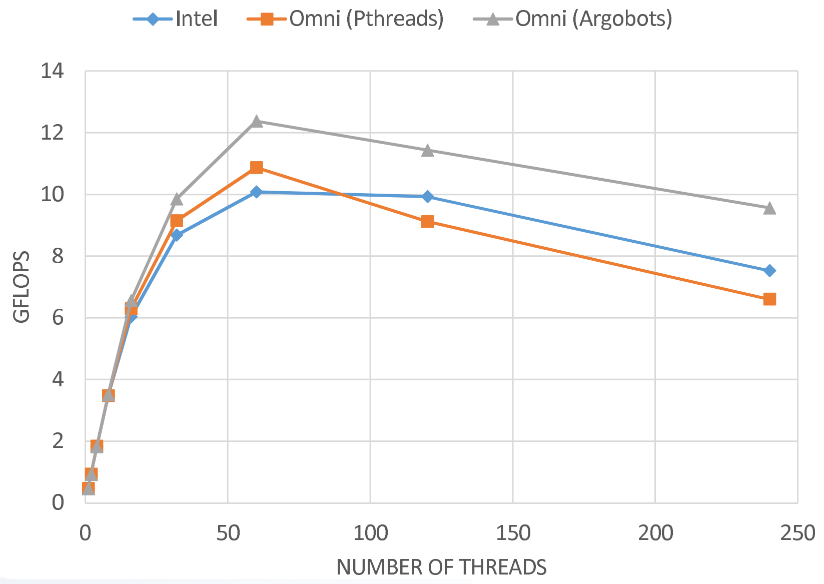
- On 2,048 cores, HPCG using MPI+Argobots shows performance improvement of **13.4%** over MPI-only version, or **27.4%** over MPI+Pthreads version.
  - As core number increases, the benefit of communication hiding begins to reveal. DDOT% increases from 0.62% on 16 cores to 36.8% on 2,048 cores.

OpenMP over Argobots

- Problem
  - OS thread-based OpenMP implementations may not be efficient for massive dynamic parallel applications
  - A blocking communication call makes an entire execution stream (i.e., OS thread) blocked
  - Inefficient support of nested parallelism and tasks
- Solution
  - Develop an OpenMP compiler that generates Argobots ULTs and tasklets depending on the existence of blocking call (e.g. MPI call) in the code
  - Develop an OpenMP runtime that manages created ULTs and tasklets over a fixed set of computational resources

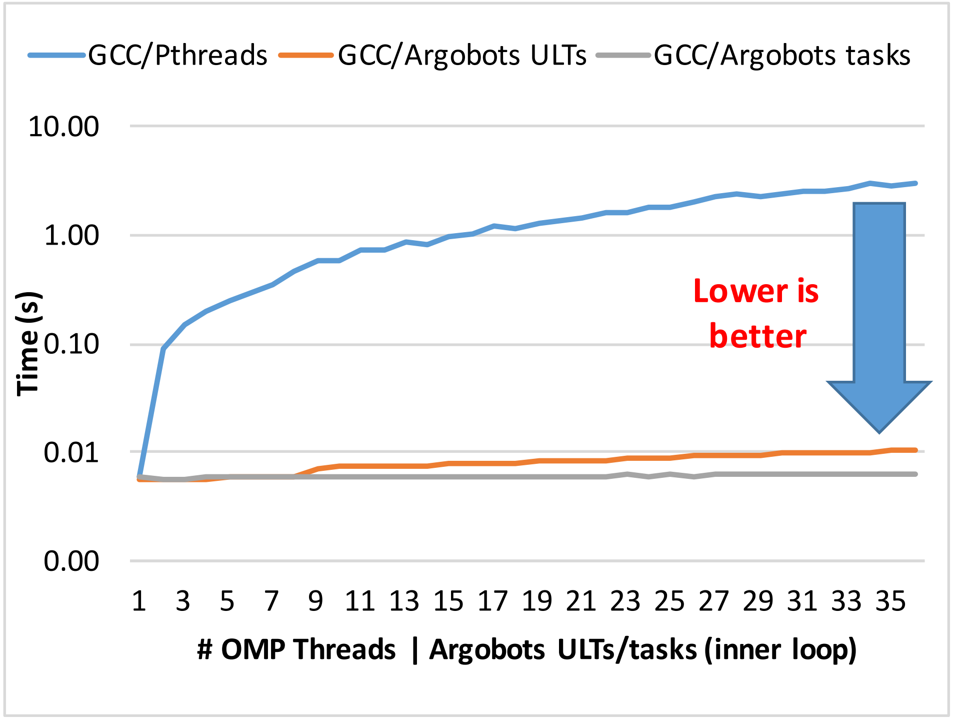
Contact:
- Pavan Balaji balaji@anl.gov
- Sangmin Seo sseo@anl.gov
- Mitsuhisa Sato msato@riken.jp
- Jinpil Lee jinpil.lee@riken.jp

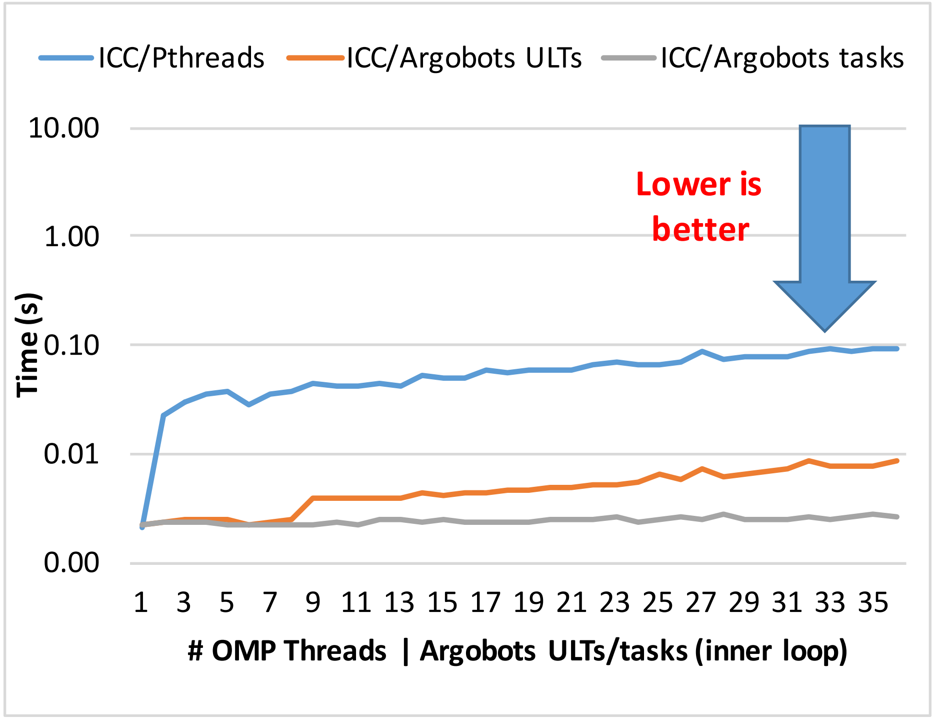
- Recent results
  - Developed an OpenMP compiler/runtime prototype over Argobots based on Clang/LLVM (ANL): BOLT compiler and runtime framework
  - Optimized Omni compiler framework (collaboration with RIKEN)
  - Demonstrated better support of nested parallelism
- Impact/Potential
  - The new OpenMP compiler/runtime developed would improve the performance of applications by leveraging lightweight threads/tasks



Nested Parallel Loop: Performance

Execution time for 36 threads in the outer loop



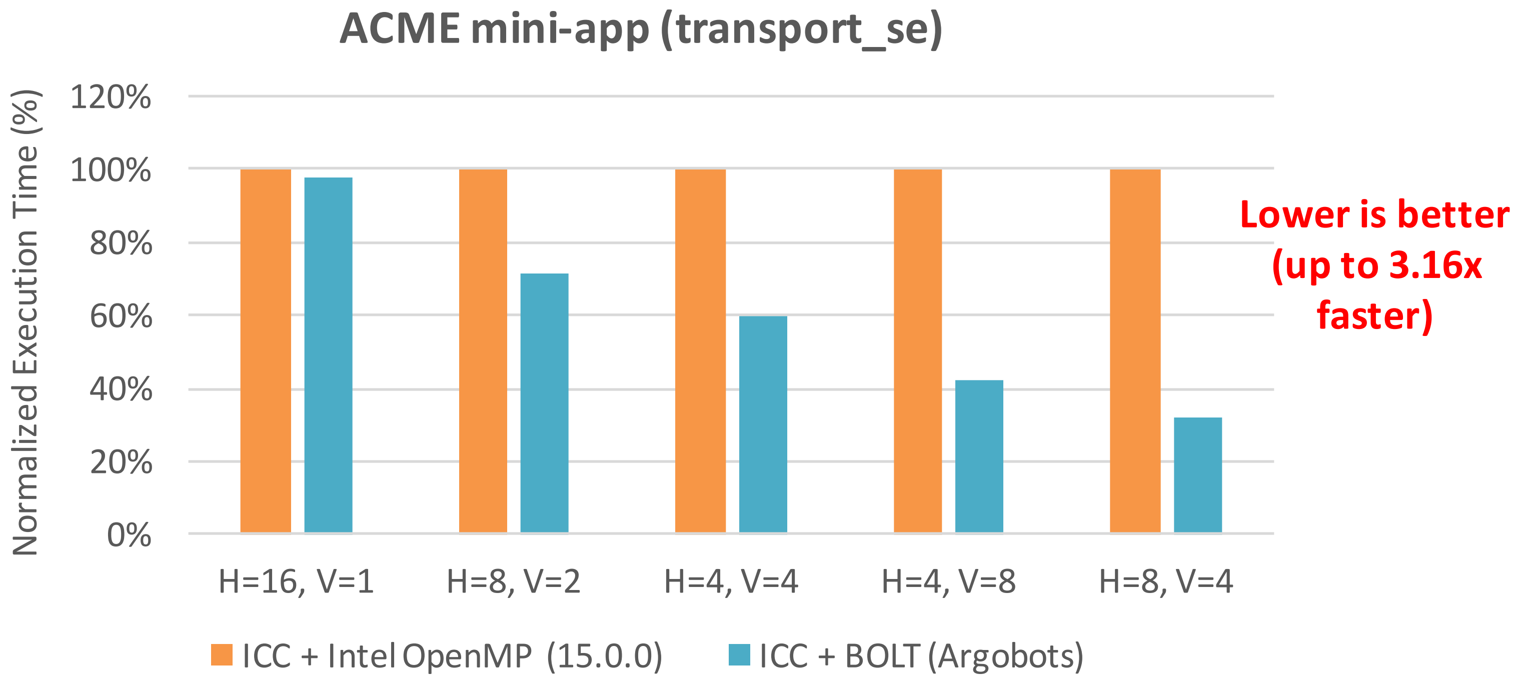


GCC OpenMP implementation does not reuse idle threads in nested parallel regions, all the teams of threads need to be created in each iteration

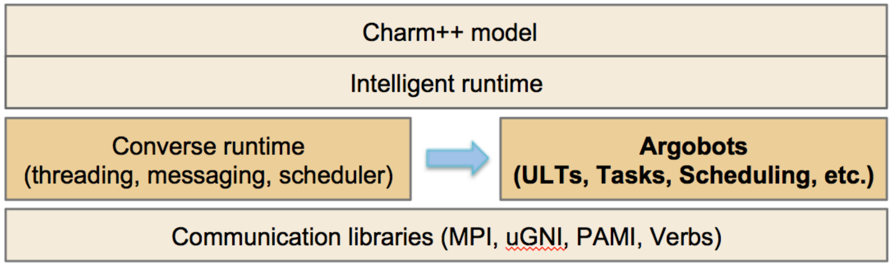
Some overhead is added by creating ULTs instead of tasks

Application Study: ACME mini-app

- ACME (Accelerated Climate Modeling for Energy)
  - Implementing additional levels of parallelism through OpenMP nested parallel loops for upcoming many-core machines
- Preliminary results of testing the `transport_se` mini-app version of HOMME (ACME's CAM-SE dycore)



Charm++ with Argobots

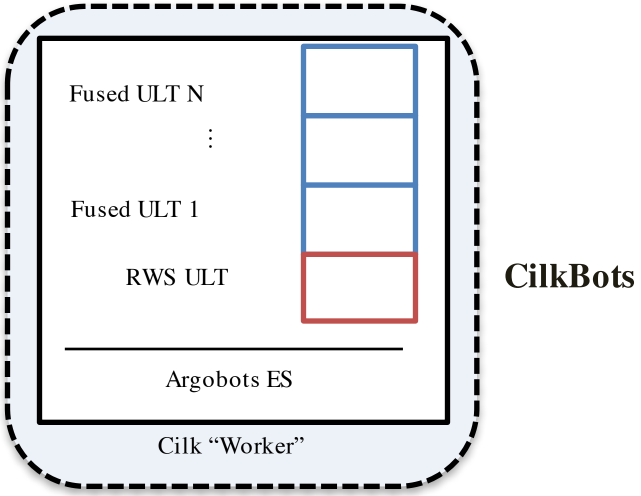

Charm++ model
Intelligent runtime
Converse runtime (threading, messaging, scheduler)
Argobots (ULTs, Tasks, Scheduling, etc.)
Communication libraries (MPI, uGNI, PAMI, Verbs)

- Problem
  - Charm++ is a very rich parallel programming model and several important scientific applications (like NAMD, OpenAtom, ChaNGA etc.) have been written using it.
  - In this project, our focus is to develop an Argobots port for Charm++.
  - We also address the problem of dynamically shrinking/expanding the number of cores for better power management.
- Argobots Port for Charm++
  - Converse is the active messaging layer in Charm++.
  - We prepare the Argobots port for Charm++ by instantiating Charm++ pthreads as Argobots execution streams.
  - Instead of enqueing Charm++ messages in Converse queues, we put them into appropriate Argobots pools as Argobots tasks.
  - Finally, we implement Charm++ threads on top of Argobots threads with condition variables to implement suspend/resume.
- Shrink/Expand
  - Charm++ maintains a set of pools for each scheduler, mapped to an xstream
  - Ranks in Charm++ are virtualized by saving the mapping of pool to a xstream
  - When an xstream is removed, the associated pools are put into a global list
  - To maintain correctness in Charm++, the rank of any tasks/threads in the global list are derived from the pool (ranks are virtualized)
  - Shrink: Other xstreams execute work units from orphaned pools with some added synchronization
  - Expand: A new xstream is created and takes over a set of orphaned pools

Contact:
- Laxmikant Kale kale@illinois.edu
- Jonathan Lifflander jliffl2@Illinois.edu
- Prateek Jindal jindal2@illinois.edu
- Sangmin Seo sseo@anl.gov

CilkBots: Lightweight Threading-Based Cilk

- Problem
  - Massive on-node parallelism and memory domains
  - Existing spawn-sync tasking frameworks are often not inter-operable and do not account for data locality
- Solution
  - Cilk runtime that exploits Argobots low-level threading/tasking model
  - Fused execution of user-level threads in a single Cilk worker. Constrained interleaving to maximize cache locality

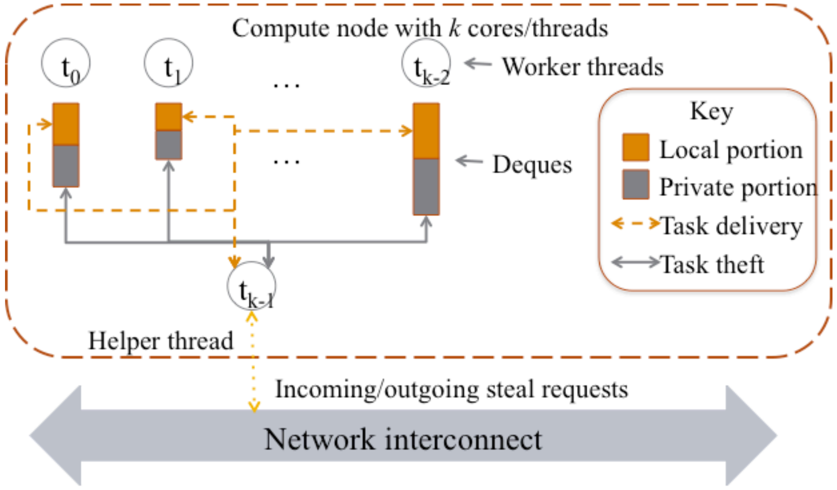


- Recent results
  - CilkBots prototype using Argobots
  - Demonstrated cache locality optimization for CilkBots programs
  - Demonstrated ability to dynamically change the number of ESs to respond to external events (e.g., power management)
    - Dynamically rebalances and executes to completion around changes in available resources
- Impact/Potential
  - CilkBots allows design of spawn-sync programs that are responsive to external events, inter-operate with other models, and enable dynamic locality optimization

Contact:
- Sriram Krishnamoorthy sriram@pnnl.gov
- Jonathan Lifflander jliffl2@illinois.edu
- Laxmikant Kale kale@illinois.edu

TASCEL with Argobots

- Problem
  - To maximize performance, distributed-memory tasking runtimes require significant programmer effort to expose all blocking operations
  - These runtimes also take over program control and do not inter-operate with other runtimes in a program's execution
- Solution
  - Distributed-memory work stealing using lightweight threads (ULTs)
  - Context switching between ULTs for performance and inter-operability

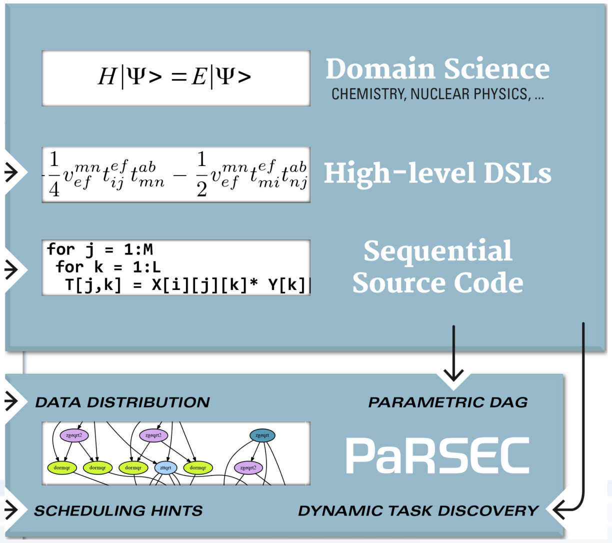


- Recent results
  - TASCEL prototype using Argobots
  - Demonstrated ability to dynamically change the number of ESs to respond to external events (e.g., power management)
- Impact/Potential
  - TASCEL over Argobots enables the design of more performance portable and composable task-parallel program phases

Contact:
- Sriram Krishnamoorthy
sriram@pnnl.gov
- Gokcen Kestor
gokcen.kestor@pnnl.gov

PaRSEC: A distributed runtime for domain specific languages

- Problem
  - Performance portability, a critical component of HPC, is difficult to be guaranteed by programming paradigms on Complex heterogeneous distributed architectures
  - Algorithmic advances and architectural progress are often disjoint
- Solution
  - separation of concerns: compiler optimizes tasks, developer describes dependencies between tasks, runtime orchestrates the dynamic execution
  - Interface with the application developers through specialized domain specific languages
  - Separate algorithms from data distribution



Recent results

- Full integration of PaRSEC with Argobots
- Full support for distributed executions
- Initial support for marshaling resources (e.g., accelerators) from NodeoOS
- Add support for task stealing and migration to adapt to dynamic condition (power management, failure prediction)
- Application support: Support for the ScaLAPACK library, and initial integration with NWCHEM

Impact/Potential

- Remove all control flow from application, resulting on completely data-driven executions
- Ease the user adoption by providing rich and flexible domain specific languages

Contact:
- Damien Genet genet@icl.utk.edu
- Thomas Herault herault@icl.utk.edu
- George Bosilca bosilca@icl.utk.edu

Argo Concurrency: Recent Results & Impact

- Recent results
  - Initial development of the “Argobots” threading/tasking library
  - Initial design of extending inter-node communication in the presence of user-level threads and tasks
  - Prototype development of OpenMP, MPI, Charm++, CilkBots, TASCEL, and PaRSEC over Argobots
- Impact
  - The new models developed would impact both legacy and new programming models that take advantage of massive on-node parallelism