Argonne NATIONAL LABORATORY

BOLT

OpenMP over Lightweight Threads

http://www.bolt-omp.org

Sangmin Seo

Assistant Computer Scientist
Argonne National Laboratory
sseo@anl.gov

November 15, 2016

U.S. DEPARTMENT OF ENERGY

OpenMP

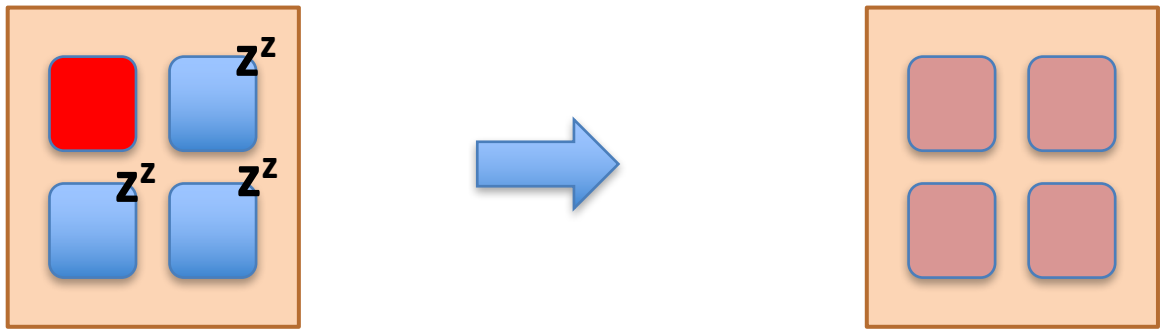
- Directive based programming model
- Commonly used for shared-memory programming in a node
- Many different implementations
  - Typically on top of Pthreads library
  - Intel, GCC, Clang, IBM, etc.

Sequential code

```
for (i = 0; i < N; i++) {
    do_something();
}
```

OpenMP code

```
#pragma omp parallel for
for (i = 0; i < N; i++) {
    do_something();
}
```



Nested Parallel Loop: Microbenchmark

```
int in[1000][1000], out[1000][1000];

#pragma omp parallel for
for (i = 0; i < 1000; i++) {
   lib_compute(i);
}

lib_compute(int x)
{
   #pragma omp parallel for
   for (j = 0; j < 1000; j++)
      out[x][j] = compute(in[x][j]);
}
```

A thread for each CPU is created by default

Each thread executes a portion

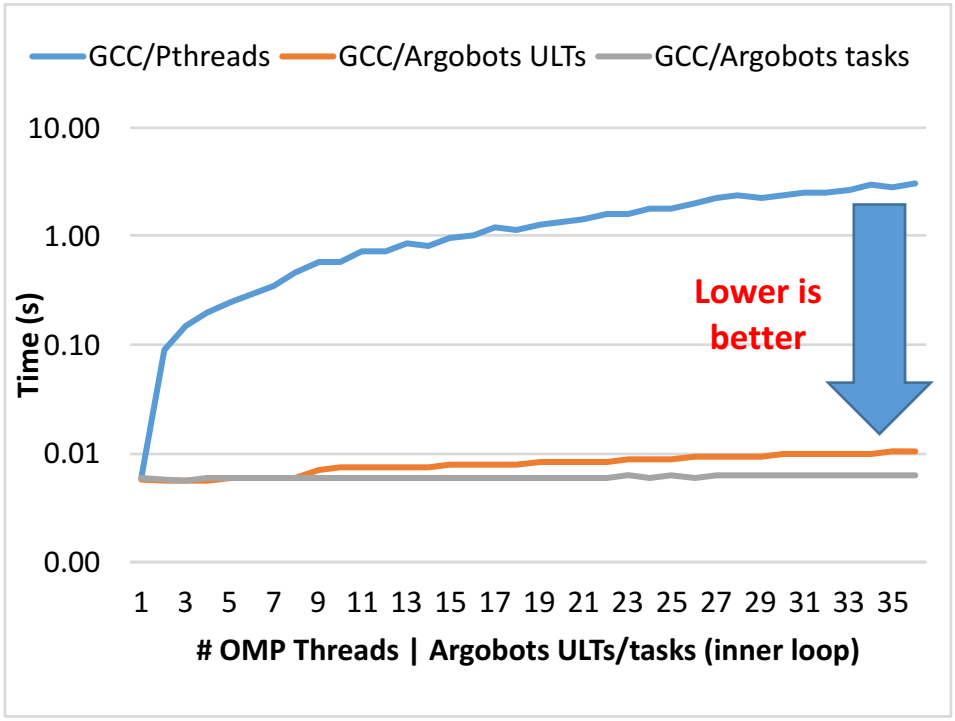
Each thread creates more threads for the second loop

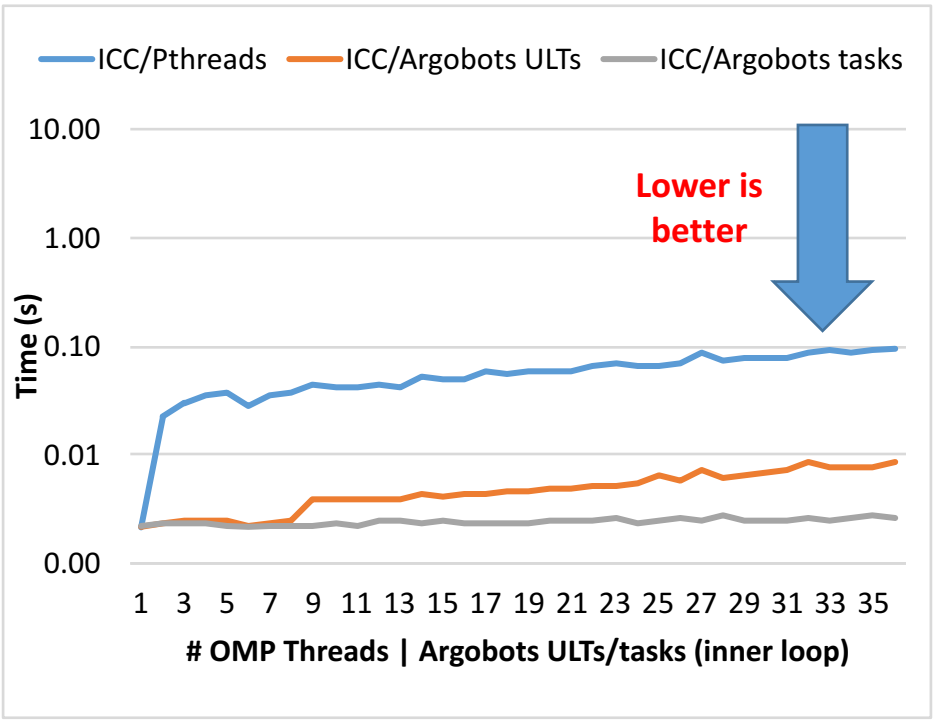
Each inner thread executes a portion

Contribution: Adrian Castello (Universitat Jaume I)

Nested Parallel Loop: Performance

Execution time for 36 threads in the outer loop





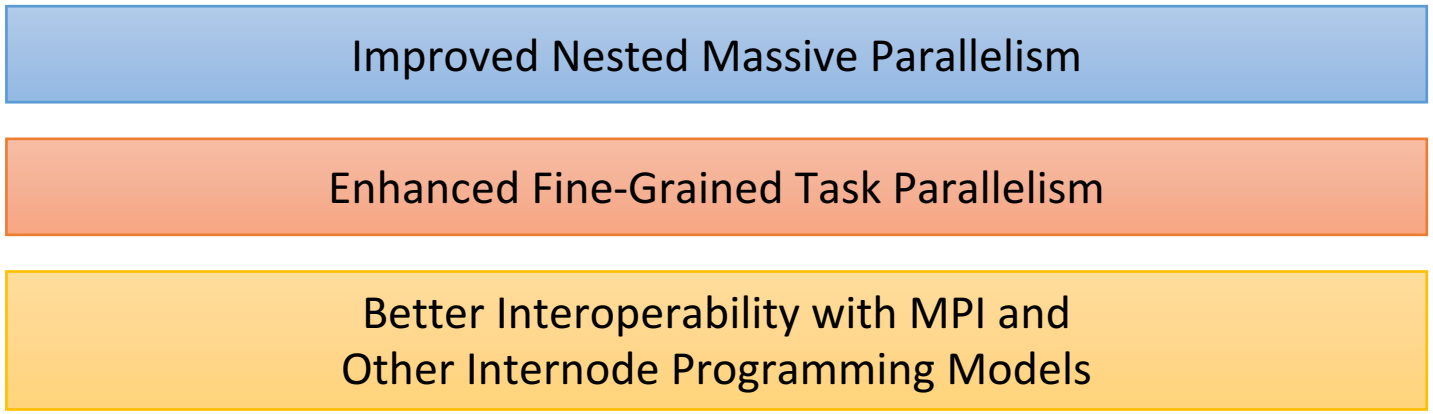
GCC OpenMP implementation does not reuse idle threads in nested parallel regions, all the teams of threads need to be created in each iteration

Some overhead is added by creating ULTs instead of tasks

BOLT: A Lightning-Fast OpenMP Implementation

- About BOLT
  - BOLT is a recursive acronym that stands for "BOLT is OpenMP over Lightweight Threads"
  - http://www.bolt-omp.org

- Objective
  - OpenMP framework that exploits *lightweight threads and tasks*



Approach & Development

- Basic approach
  - Compiler simply generates runtime API calls, while the runtime creates ULTs/tasklets and manages them over a fixed set of computational resources
  - Use ***Argobots*** as the underlying threading and tasking mechanism
  - ABI compatibility with Intel OpenMP compilers, LLVM/Clang, and GCC (i.e., can be used with these compilers)
- Development
  - Runtime
    - Based on Intel OpenMP Runtime API
    - Generates Argobots work units from OpenMP pragmas
    - Can generate ULTs or tasklets depending on code characteristics
  - Compiler (planned)
    - LLVM/Clang
    - Passes characteristics of parallel region or task (e.g., existence of blocking calls) to the runtime
    - Extends pragmas with the option “nonblocking”

Argobots

A lightweight low-level threading and tasking framework

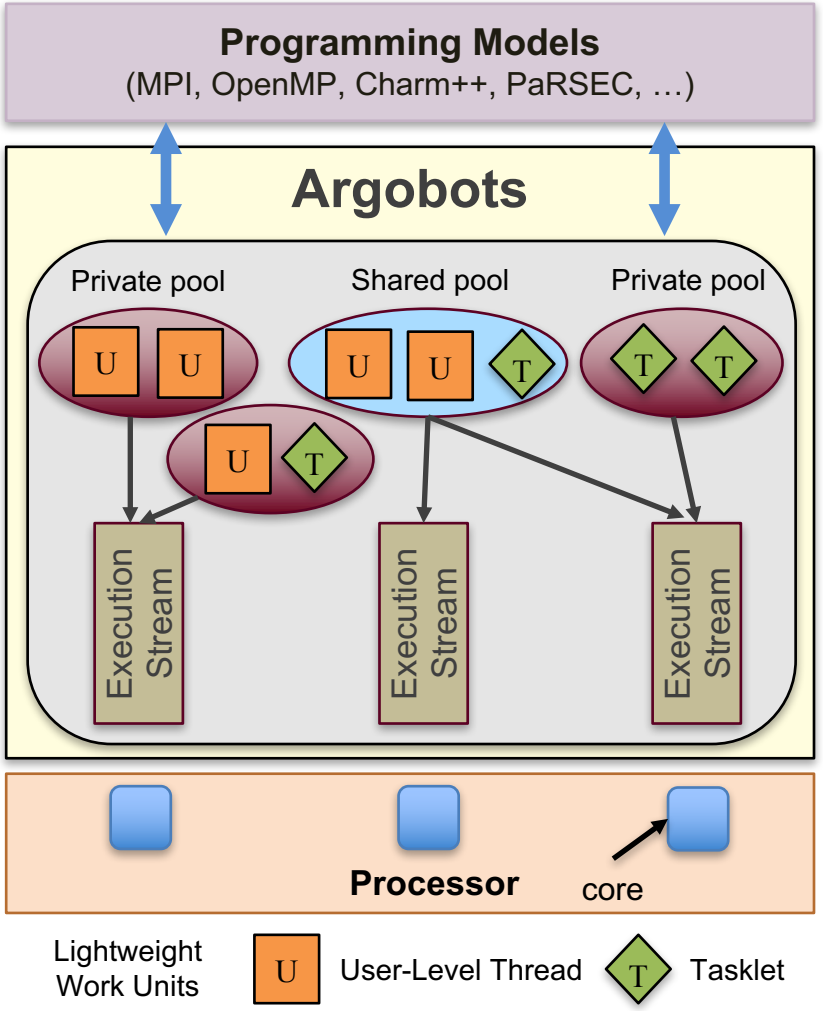
(http://www.argobots.org)

Overview

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

Argobots Innovations

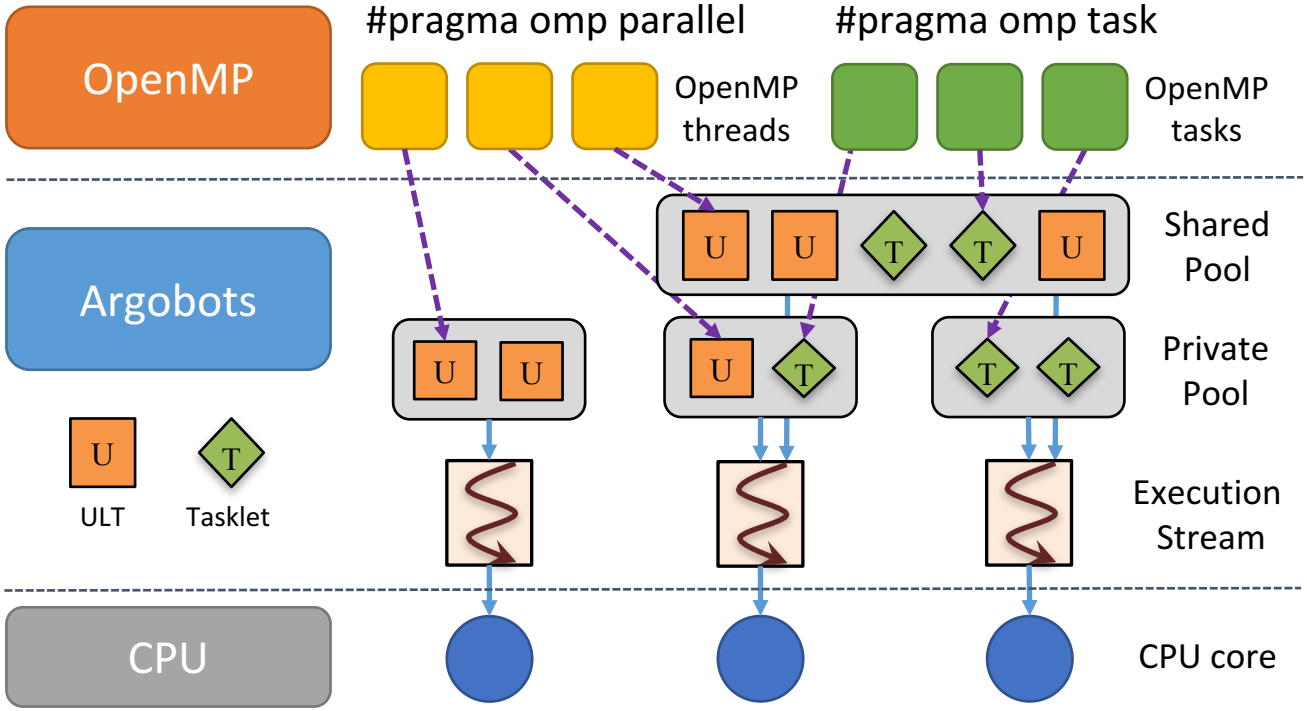
- **Enabling technology, but not a policy maker**
  - High-level languages/libraries such as OpenMP or Charm++ have more information about the user application (data locality, dependencies)
- **Explicit model**:
  - Enables dynamism, but always managed by high-level systems



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

BOLT Execution Model

- OpenMP threads and tasks are translated into Argobots work units (i.e., ULTs and tasklets)
- Shared pools are utilized to handle nested parallelism
- A customized Argobots scheduler manages scheduling of work units across execution streams



OpenMP Pragma Translation

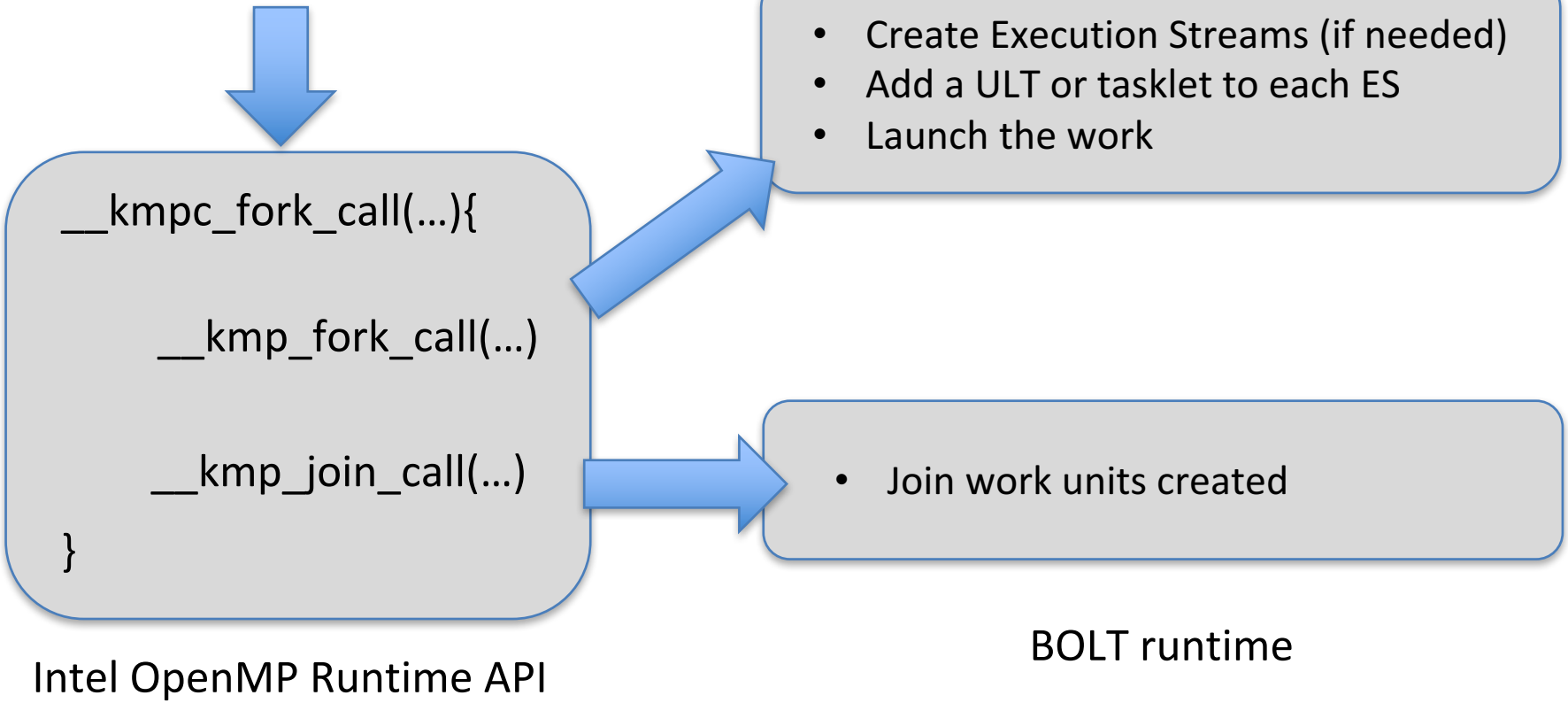
1. A set of *N* threads is created at run time
   - If they have not been created yet
   - Commonly as many as the number of CPU cores
2. The number of iterations is divided between all the threads
3. A synchronization point is added after the for loop
   - Implicit barrier at the end of parallel for

```
#pragma omp parallel for (1,2)
for (i = 0; i < N; i++) {
    do_something();
} (3)
```

OpenMP Compiler & BOLT Runtime

#pragma omp parallel

Clang and Intel compiler



parallel for

```
#pragma omp parallel for
for (i = 0; i < N; i++) {
    do_something();
}
```

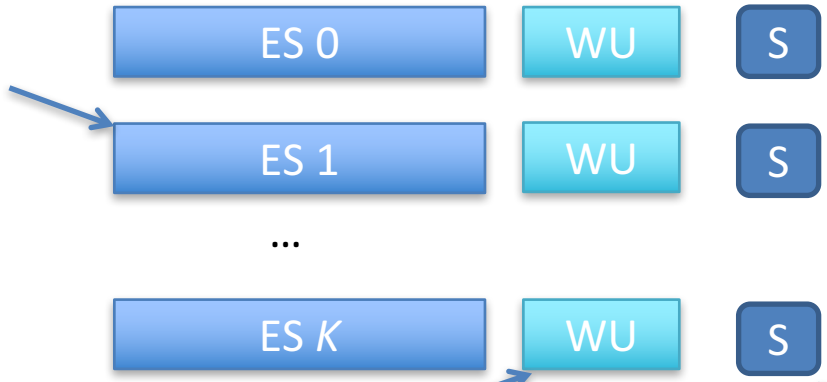
Creates threads

Divides all iterations among threads

Synchronization point

Implementation using Argobots

One Execution Stream
for each CPU core
(or hardware thread)

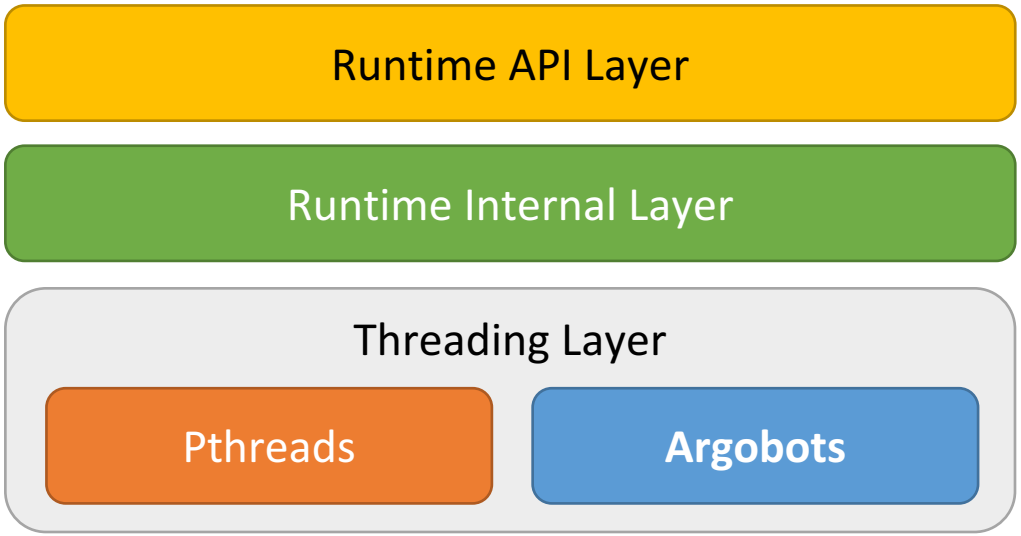


Each work unit executes
a portion of the for loop

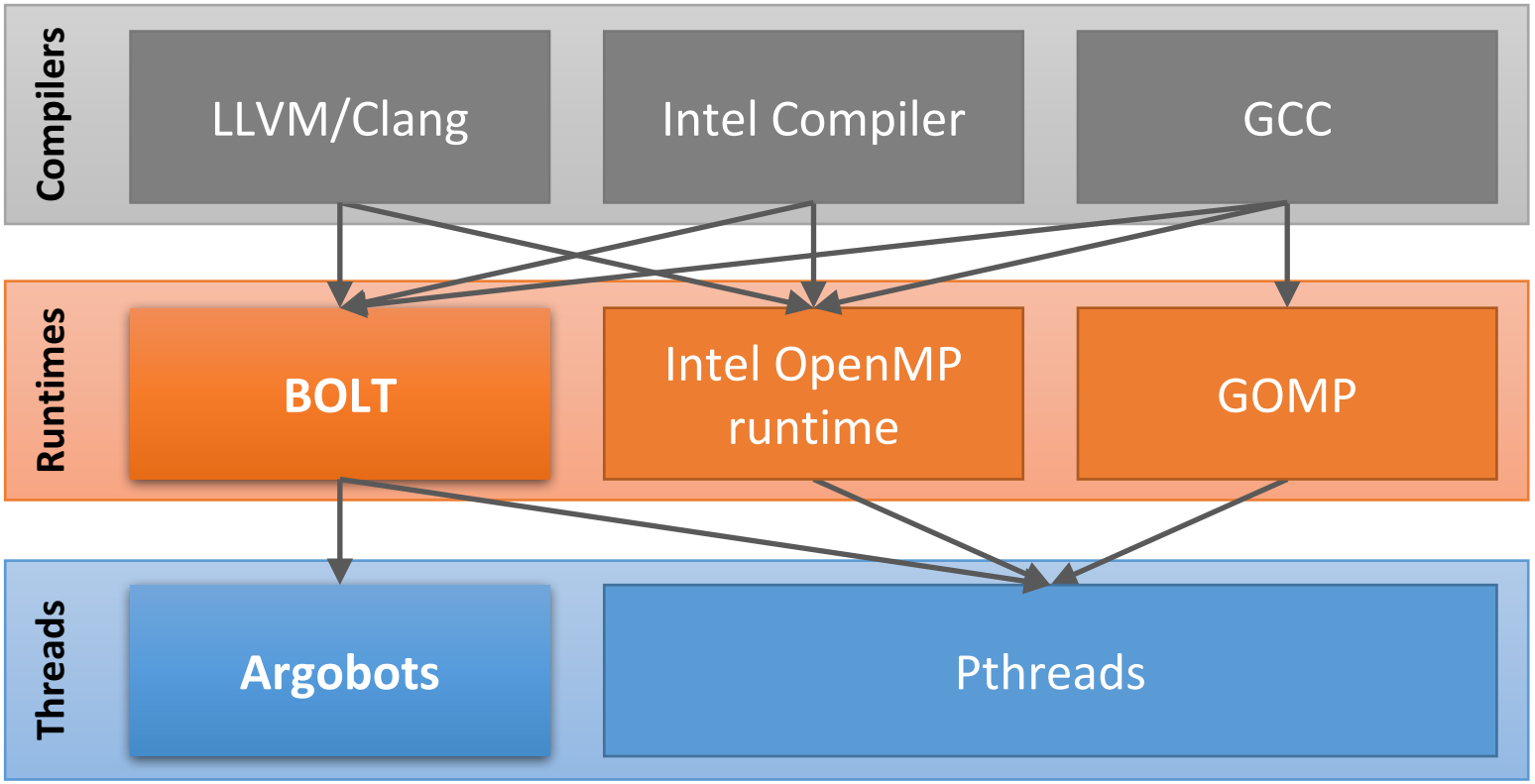
A synchronization point is added

Prototype Implementation of BOLT Runtime

- Based on Intel's open-source OpenMP runtime
  - http://openmp.llvm.org/
- Kept the original runtime API for the ABI compatibility
- Designed and implemented the threading layer using Argobots and modified the runtime internal layer



How to use BOLT?



Two ways to use BOLT

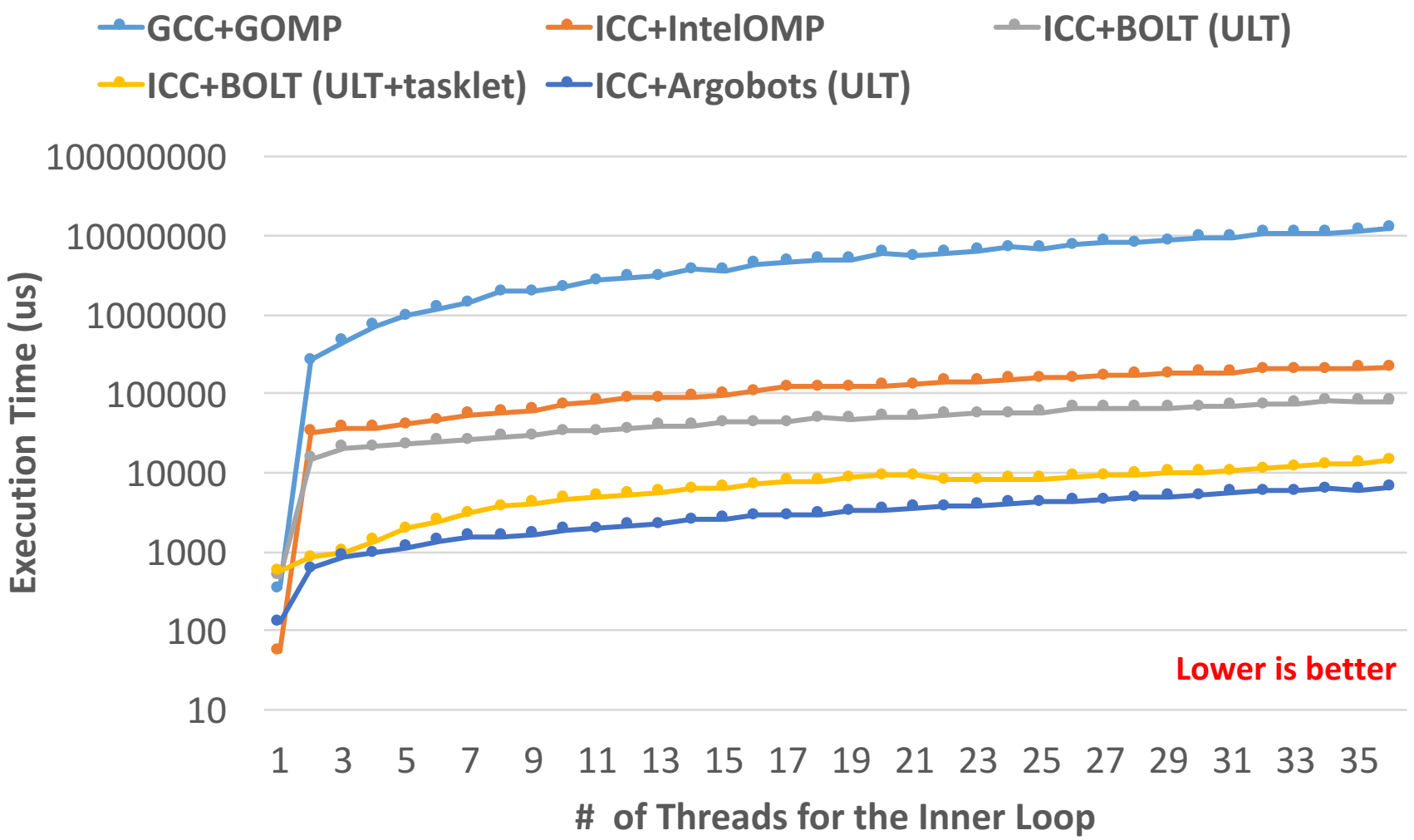
1. Compile your code with LLVM/Clang, Intel compiler, or GCC while linking BOLT
2. LD_PRELOAD=<bolt_installation_path>/lib/libomp.so (no recompilation needed)

OpenUH OpenMP Validation Suite 3.1

	GCC 6.1	ICC 17.0.0 + Intel OpenMP	ICC 17.0.0 + BOLT (Argobots)	LLVM/clang 3.9 + BOLT (Argobots)
# of tested OpenMP constructs	62	62	62	62
# of used tests	123	123	123	123
# of successful tests	**118**	**118**	**122**	**112**
# of failed tests	5	5	1	1
Pass rate (%)	**95.9**	**95.9**	**99.2**	**99.2**

- The BOLT prototype functionally works well!

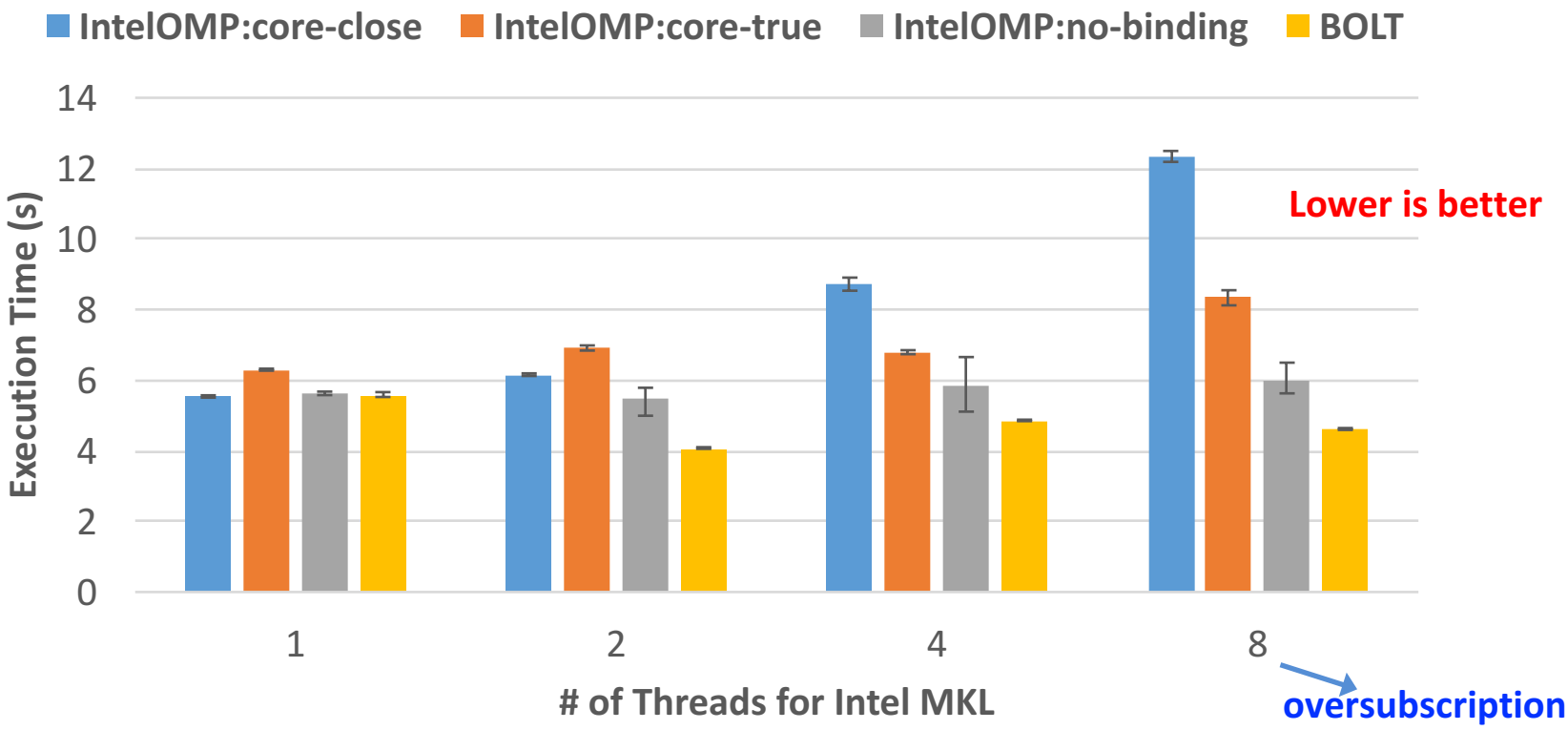
Nested Parallel Loop Microbenchmark



* The number of threads for the outer loop was fixed at 36.

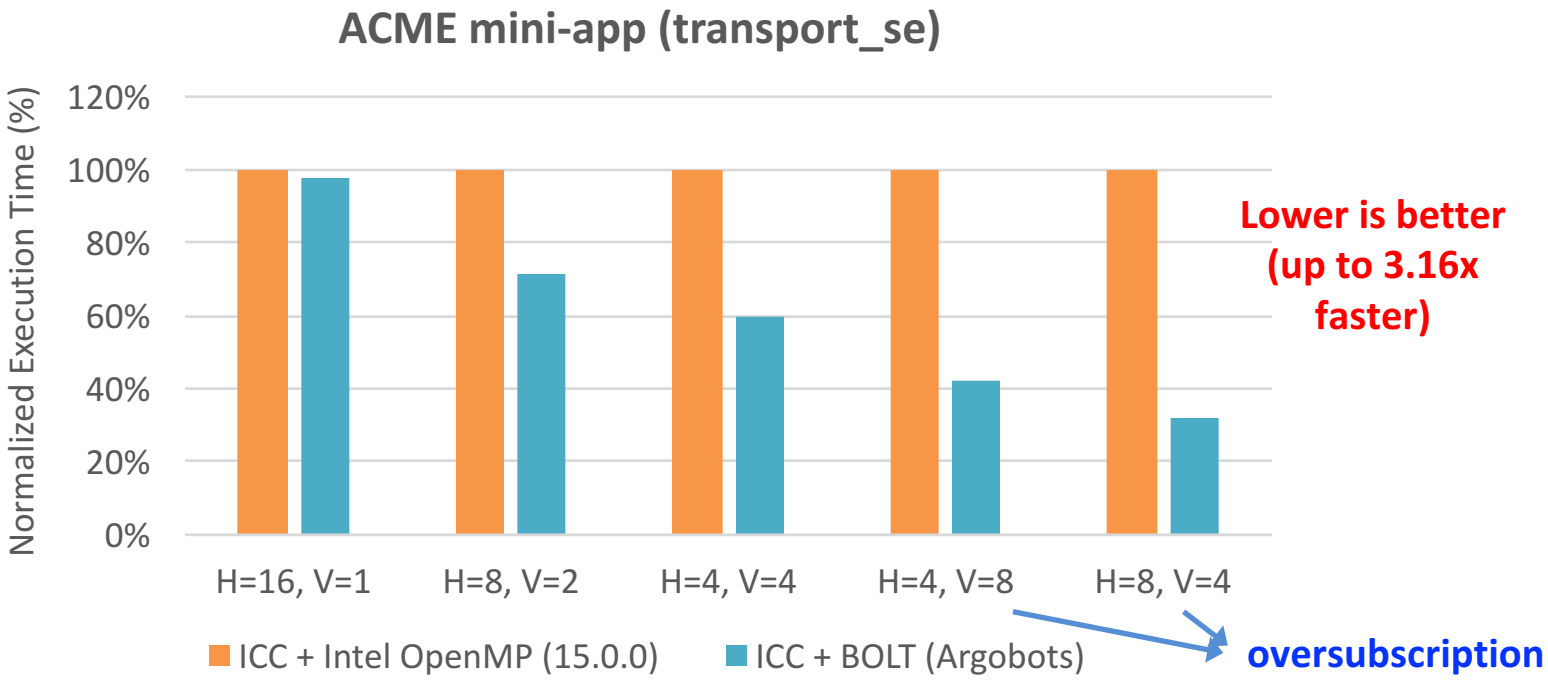
Application Study: KIFMM

- Kernel-Independent Fast Multipole Method (KIFMM)
  - Offload dgemv operations to Intel MKL
- Evaluated the efficiency of the nested parallelism support in Intel OpenMP and BOLT during the Downward stage
  - 9 threads for the application (outer parallel region)



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)



Summary

- BOLT: OpenMP over Lightweight Threads
  - More efficient support of nested parallelism with Argobots ULTs and tasklets
  - Preliminary results show that BOLT is promising
  - BOLT 1.0a1 pre-release is available at http://www.bolt-omp.org

- Argobots
  - A lightweight low-level threading/tasking framework
  - Provides efficient mechanisms, not policies, to users (library developers or compilers)
    - They can build their own solutions

BOLT Team

- **Maintainers**
  - Argonne National Laboratory
    - Sangmin Seo
    - Abdelhalim Amer
    - Pavan Balaji
- **Contributors**
  - Universitat Jaume I de Castelló
    - Adrián Castelló
    - Rafael Mayo
    - Enrique S. Quintana-Ortí
  - Barcelona Supercomputing Center (BSC)
    - Antonio J. Peña
    - Jesus Labarta
  - RIKEN
    - Jinpil Lee
    - Mitsuhisa Sato

Argobots Team

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

Past Team Members:

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

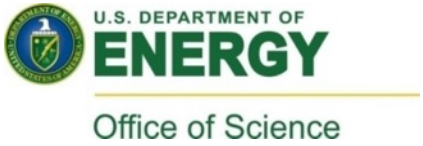
Try BOLT & Argobots

- **BOLT**
  - Pre-release 1.0a1 is available
  - http://www.bolt-omp.org
  - git repository
    - https://github.com/pmodels/bolt

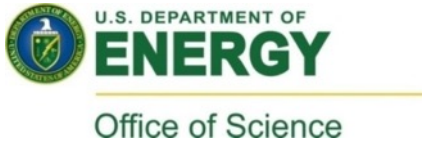
- **Argobots**
  - Pre-release 1.0a1 is available
  - http://www.argobots.org
  - git repository
    - https://github.com/pmodels/argobots

Funding Acknowledgments

Funding Grant Providers



Infrastructure Providers



Programming Models and Runtime Systems Group

Group Lead

- Pavan Balaji (computer scientist and group lead)

Current Staff Members

- Abdelhalim Amer (postdoc)
- Yanfei Guo (postdoc)
- Rob Latham (developer)
- Lena Oden (postdoc)
- Ken Raffenetti (developer)
- Sangmin Seo (assistant computer scientist)
- Min Si (postdoc)

Past Staff Members

- Antonio Pena (postdoc)
- Wesley Bland (postdoc)
- Darius T. Buntinas (developer)
- James S. Dinan (postdoc)
- David J. Goodell (developer)
- Huiwei Lu (postdoc)
- Min Tian (visiting scholar)
- Yanjie Wei (visiting scholar)
- Yuqing Xiong (visiting scholar)
- Jian Yu (visiting scholar)
- Junchao Zhang (postdoc)
- Xiaomin Zhu (visiting scholar)

Current and Recent Students

- Ashwin Aji (Ph.D.)
- Abdelhalim Amer (Ph.D.)
- Md. Humayun Arafat (Ph.D.)
- Alex Brooks (Ph.D.)
- Adrian Castello (Ph.D.)
- Dazhao Cheng (Ph.D.)
- Hoang-Vu Dang (Ph.D.)
- James S. Dinan (Ph.D.)
- Piotr Fidkowski (Ph.D.)
- Priyanka Ghosh (Ph.D.)
- Sayan Ghosh (Ph.D.)
- Ralf Gunter (B.S.)
- Jichi Guo (Ph.D.)
- Yanfei Guo (Ph.D.)
- Marius Horga (M.S.)
- John Jenkins (Ph.D.)
- Feng Ji (Ph.D.)
- Ping Lai (Ph.D.)
- Palden Lama (Ph.D.)
- Yan Li (Ph.D.)
- Huiwei Lu (Ph.D.)
- Jintao Meng (Ph.D.)
- Ganesh Narayanaswamy (M.S.)
- Qingpeng Niu (Ph.D.)
- Ziaul Haque Olive (Ph.D.)
- David Ozog (Ph.D.)
- Renbo Pang (Ph.D.)
- Nikela Papadopoulou (Ph.D)
- Sreeram Potluri (Ph.D.)
- Sarunya Pumma (Ph.D)
- Li Rao (M.S.)
- Gopal Santhanaraman (Ph.D.)
- Thomas Scogland (Ph.D.)
- Min Si (Ph.D.)
- Brian Skjerven (Ph.D.)
- Rajesh Sudarsan (Ph.D.)
- Lukasz Wesolowski (Ph.D.)
- Shucai Xiao (Ph.D.)
- Chaoran Yang (Ph.D.)
- Boyu Zhang (Ph.D.)
- Xiuxia Zhang (Ph.D.)
- Xin Zhao (Ph.D.)

Advisory Board

- Pete Beckman (senior scientist)
- Rusty Lusk (retired, STA)
- Marc Snir (division director)
- Rajeev Thakur (deputy director)

Q&A

- Thank you for your attention!

Questions?