



HetroOMP: OpenMP for Hybrid Load Balancing Across Heterogeneous Processors

Vivek Kumar¹, Abhiprayah Tiwari¹, Gaurav Mitra²

¹ IIIT Delhi, New Delhi, India

² Texas Instruments, Sugarland, Texas, USA



Outline

- Introduction
- Contributions
- Motivating analysis
- Insights and approach
- Implementation
- Experimental Evaluation
- Summary



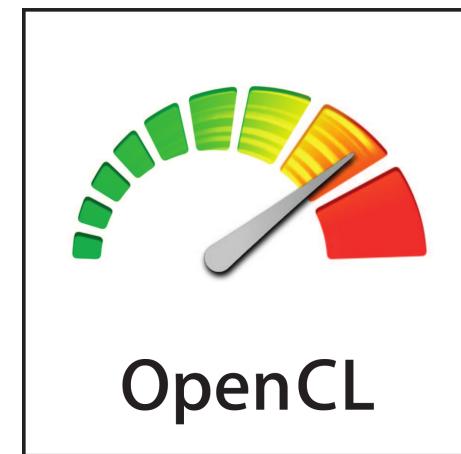
Introduction

Accelerator Programming

Directive-based



Language-based





Accelerator Programming

OpenMP™

Supported on wide range of processors and accelerators

Supports both data and task parallelism

Pragmas for multicore programming

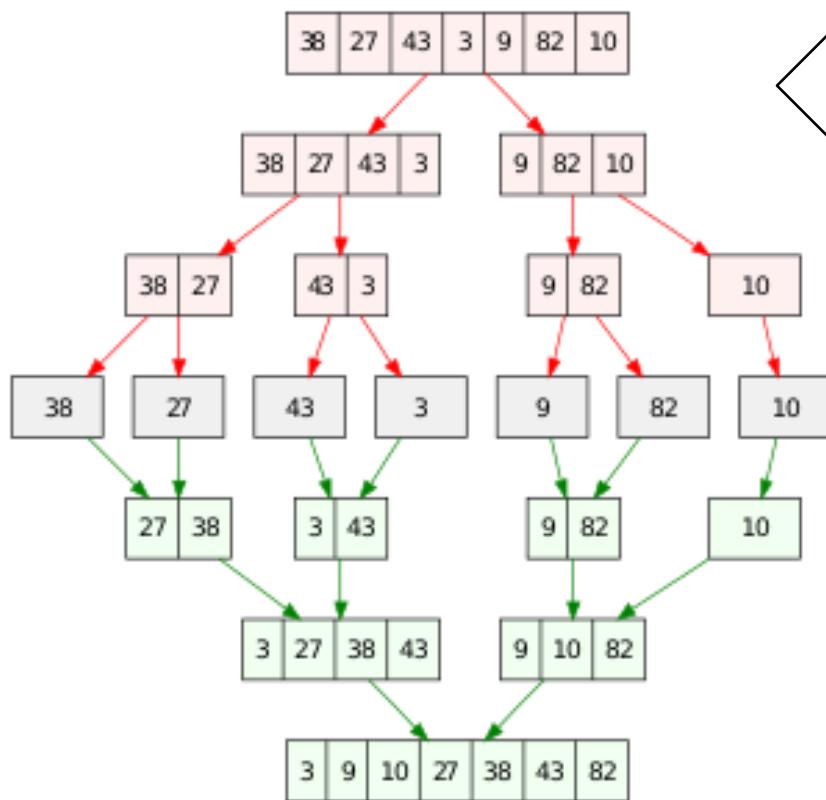
Pragmas for accelerator programming

Can I use both multicore and accelerator together?

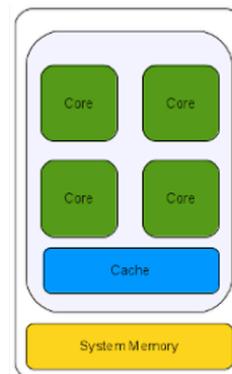


Hybrid Parallelism

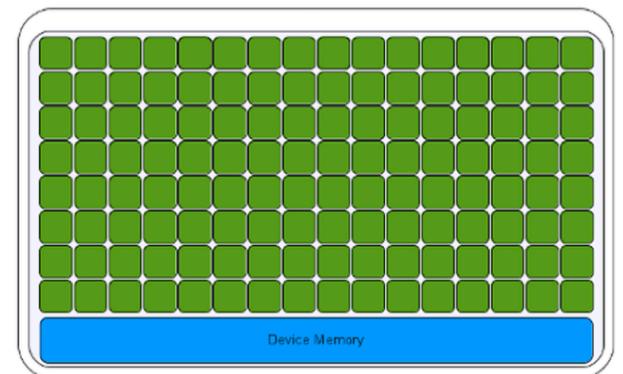
Mergesort: recursive call with divide-and-concur parallelism



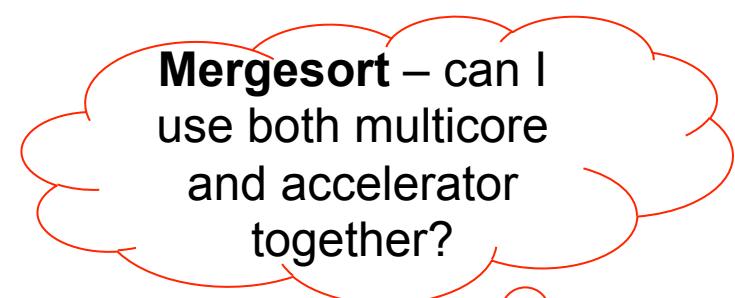
Multicore



Accelerator

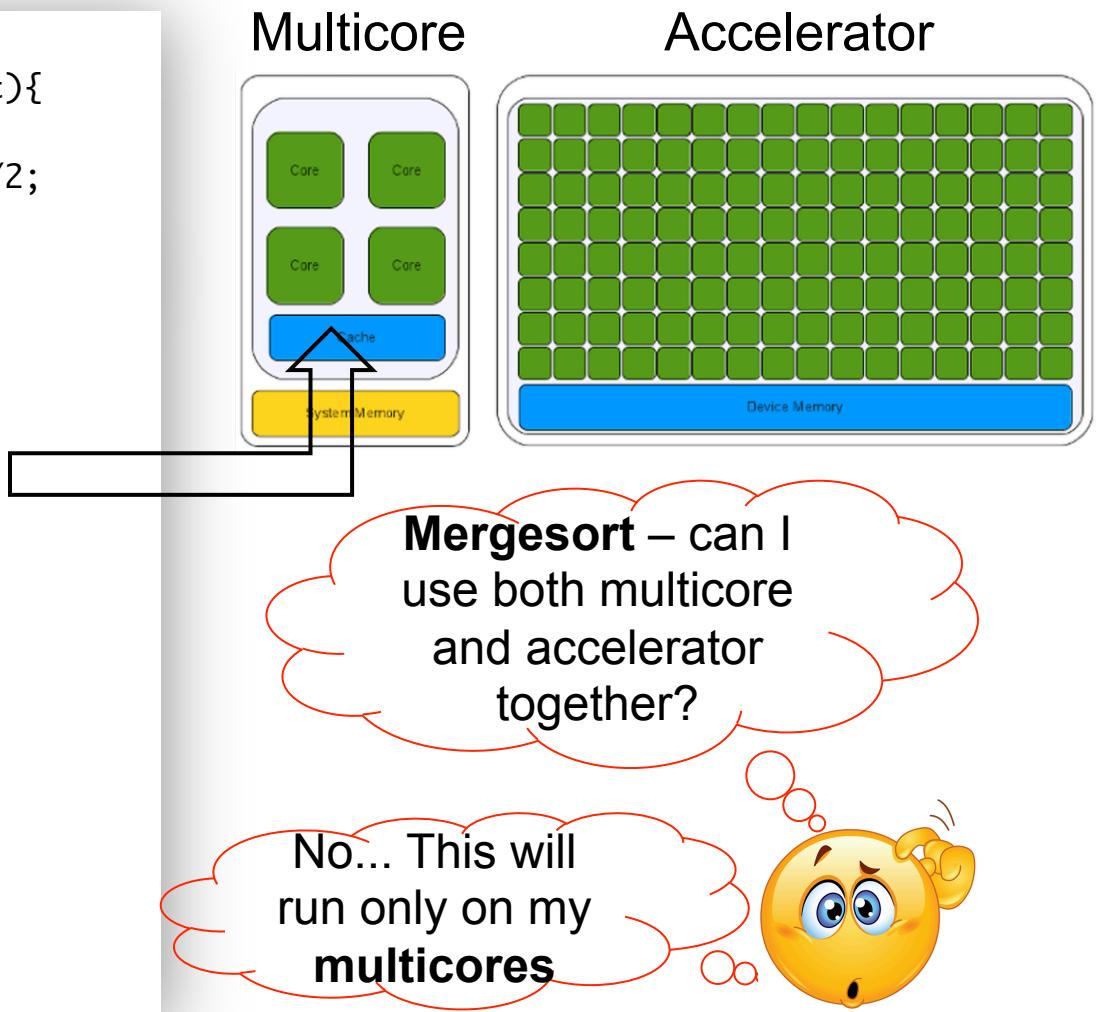


Mergesort – can I use both multicore and accelerator together?



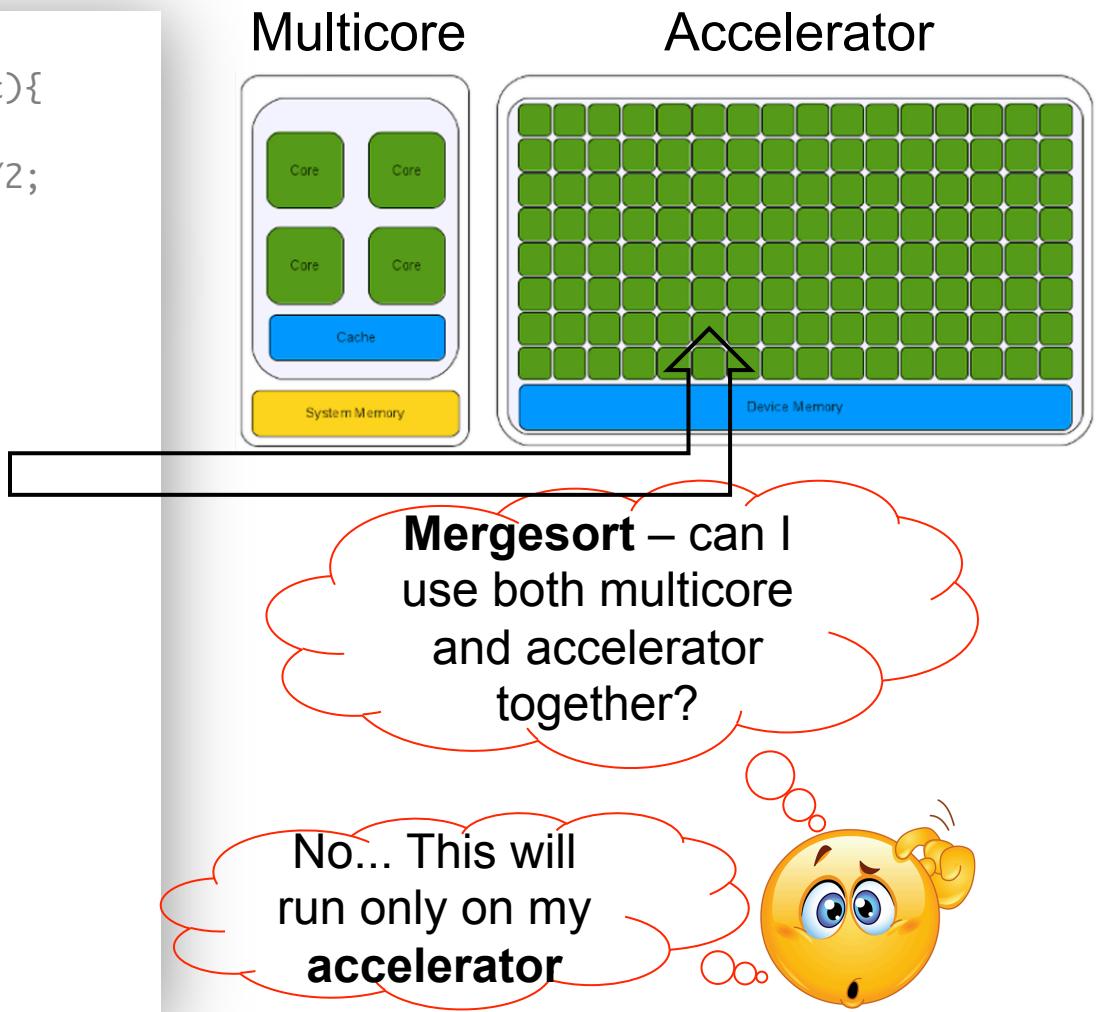
Hybrid Parallelism in OpenMP (Attempt #1)

```
1. int THRESHOLD=/*some value*/;
2. void mergesort(int left, int right){
3.   if(right-left > THRESHOLD) {
4.     int mid = left + (left+right)/2;
5.     #pragma omp task untied \
6.       firstprivate(left,mid)
7.     mergesort(left, mid);
8.     mergesort(mid+1, right);
9.     #pragma omp taskwait
10.    merge(left, mid, right);
11.  } else {
12.    sequentialSort(left, right);
13.  }
14.}
15.void main() {
16.
17.
18. #pragma omp parallel \
19.   firstprivate(A:N)
20. #pragma omp single
21.   mergesort(0, size-1);
22.}
```



Hybrid Parallelism in OpenMP (Attempt #2)

```
1. int THRESHOLD=/*some value*/;
2. void mergesort(int left, int right){
3.   if(right-left > THRESHOLD) {
4.     int mid = left + (left+right)/2;
5.     #pragma omp task untied \
6.       firstprivate(left,mid)
7.       mergesort(left, mid);
8.       mergesort(mid+1, right);
9.     #pragma omp taskwait
10.    merge(left, mid, right);
11.  } else {
12.    sequentialSort(left, right);
13.  }
14.}
15.void main() {
16.  #pragma omp target map(to:N) \
17.    map(tofrom:A[0:N])
18.  #pragma omp parallel \
19.    firstprivate(A:N)
20.  #pragma omp single
21.  mergesort(0, size-1);
22.}
```



Hybrid Parallelism in OpenMP (How?)

- Manually partitioning the workload between multicore and accelerator?
 - Two different kernels, one each for host and device
 - No serial elision – different behavior if directive disabled
 - What should be the optimal partition size?
 - Host and accelerator have different performance
 - Communication latency between host and device
 - There could be several layers of parallelism (NP-hard)

I can only use either host or device.. One of the processing unit will remain idle..





Research Questions

1. Can we extend OpenMP accelerator model to support hybrid parallelism without affecting programmer's productivity?
2. Can we design a high performance hybrid runtime for such an extension?



Contributions

✓ HetroOMP programming model

Extension to OpenMP accelerator model for enabling hybrid parallelism across host and device

✓ Lightweight runtime implementation

That uses hybrid work-stealing runtime for dynamic load balancing over an ARM+DSP based MPSoC

✓ Detailed performance study

Using several data and task parallel benchmarks

✓ Results

That demonstrates HetroOMP achieves significant speedup over OpenMP

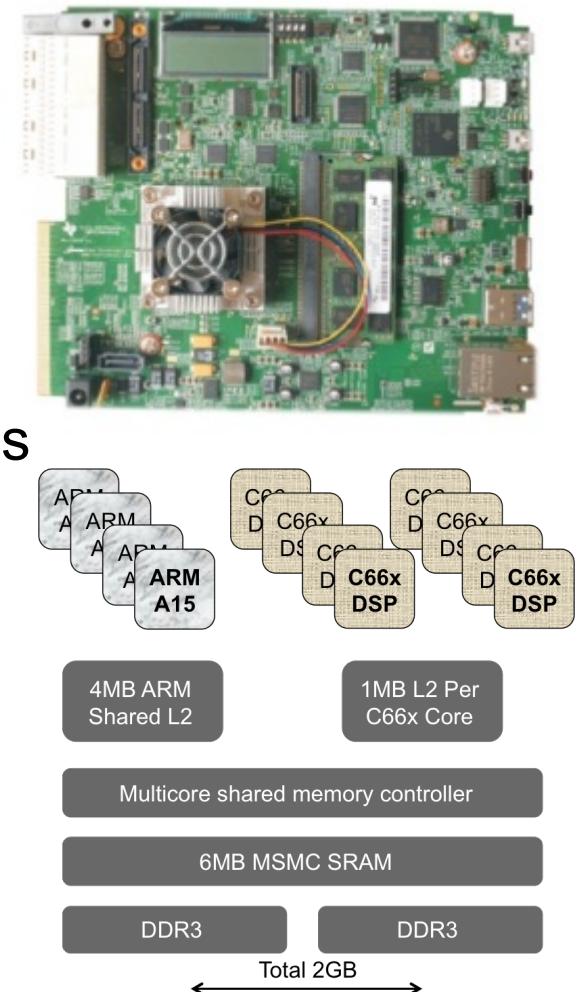


Motivating Analysis

MPSOC used in this Study: TI Keystone II

- **Architecture**

- 4 ARM + 8 DSP cores
- Cache coherency among ARM cores
- No cache coherency among DSPs / between DSP and ARM
- Shared memory w/ different address spaces
 - Pointer conversion needed bw. ARM & DSP
- L1 cache line sizes different at ARM (64 bytes) and DSP (128 bytes)
- C library for DSPs doesn't support concurrency
 - Concurrent **hardware** queues and **hardware** semaphores

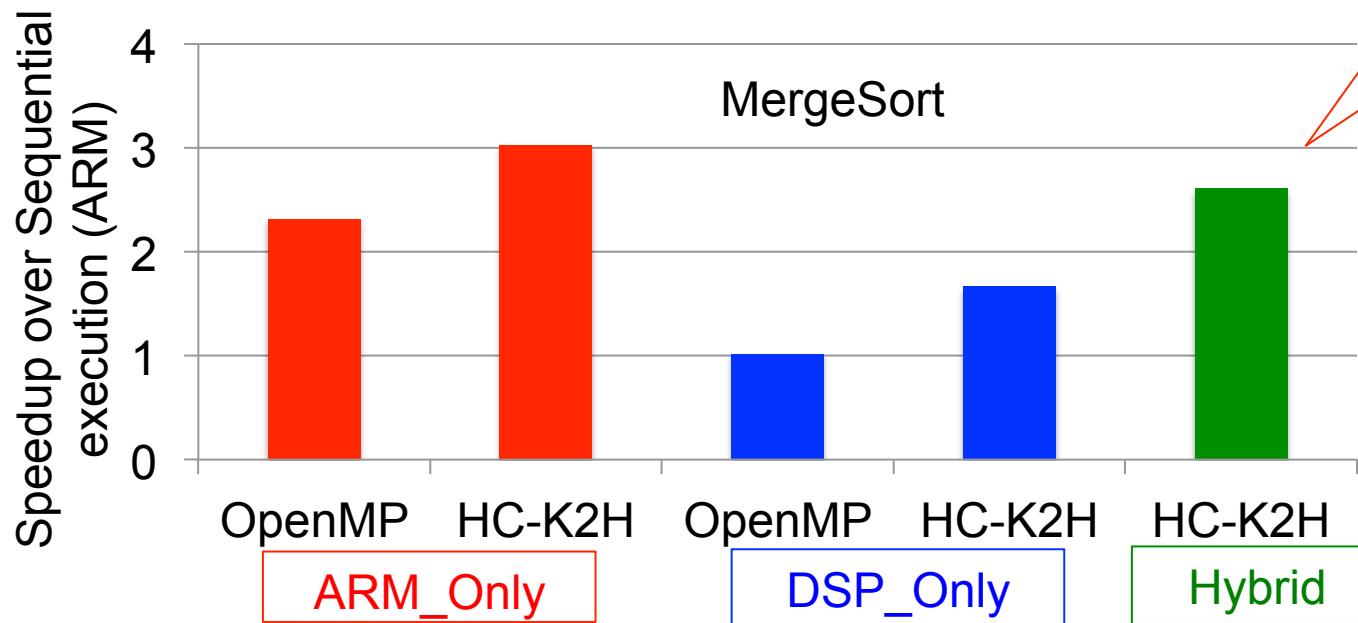


MPSOC used in this Study: TI Keystone II

- **Existing programming models**

- OpenMP accelerator model [1]
- HC-K2H (Habanero C) [2]
 - No serial elision
 - Hybrid ARM/DSP work-stealing scheduler

Hybrid work-stealing performance worse than ARM_Only



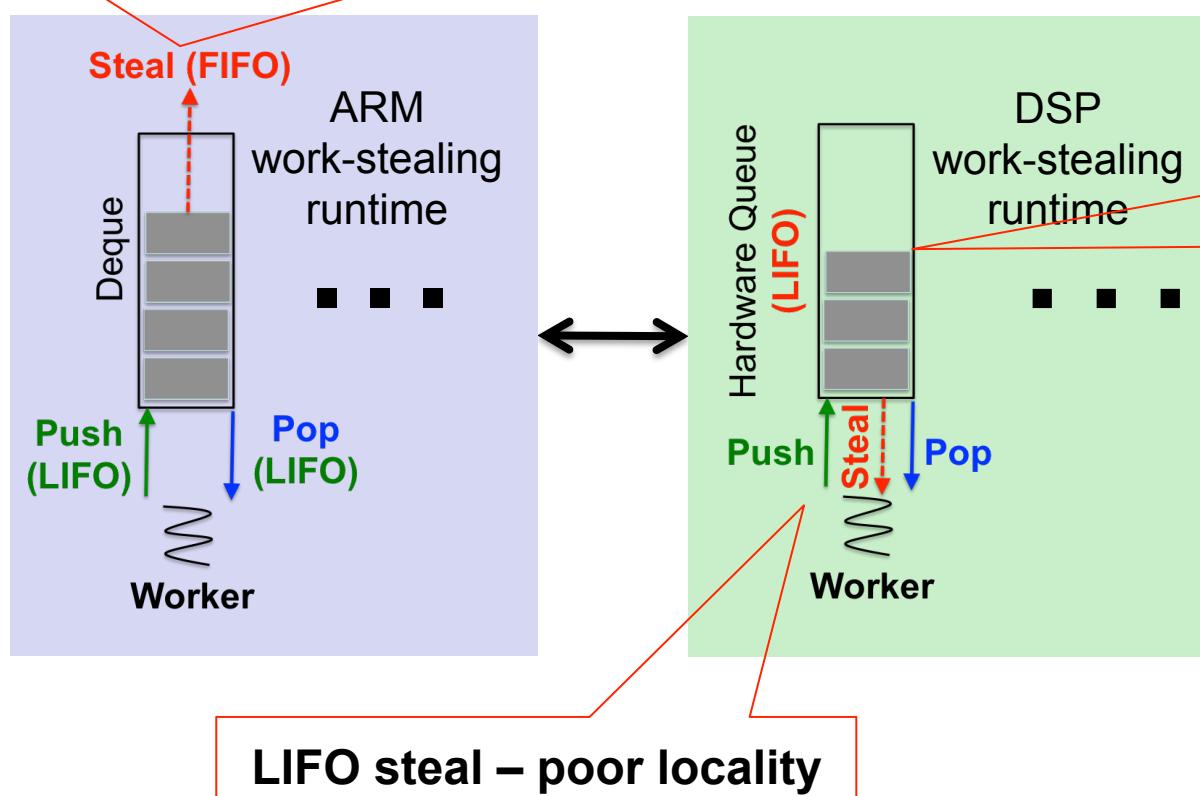
[1] Stotzer et al., OpenMP on the low-power TI Keystone-II ARM/DSP system-on-chip, IWOMP 2013

[2] Kumar et al., Heterogeneous work-stealing across ARM and DSP cores, HPEC 2015

MPSOC used in this Study: TI Keystone II

- Drawbacks in HC-K2H's hybrid work-stealing

FIFO steal by double-ended queue (deque)



Concurrent queues –
Either of the ARM or
DSP cores could
directly steal

Cache incoherent
DSPs require explicit
**cache write backs at
each task
synchronization
points**



Insights

- OpenMP should support hybrid execution across host and accelerator
- Hybrid work-stealing runtime at DSP
 - Should improve locality by supporting FIFO steals
 - Should not perform costly cache writebacks at all task synchronization points



Approach

- Hybrid programming
 - HetroOMP programming model
 - ✓ Simple extension to OpenMP accelerator model
 - ✓ **hetro** clause to define the scope of hybrid execution
- Hybrid execution using work-stealing
 - ✓ Non-concurrent **private deque** [1] on L2 cache of DSP
 - ✓ LIFO push and pop, whereas FIFO steals (improved locality)
 - ✓ Sender initiated steal operations at DSP can keep track if thief is cache coherent
 - ✓ Cache writeback only if a task was sent to a cache incoherent core

[1] Acar et al., *Scheduling Parallel Programs by Work-Stealing with Private Deques*, PPoPP 2013



Implementation



HetroOMP Programming Model

- Usage of the clause “**hetro**”
 1. **#pragma omp parallel hetro**
 - Indicating the scope of hybrid execution
 2. **#pragma omp task hetro(Var1:Count1, ...)**
 - Name and count of all writable type share variables
 - Var should only be a pointer type
 - Count is the number of elements (e.g., array size)
 3. **#pragma omp for schedule(hetro, chunks)**
 - Hybrid execution of loop iterations

HetroOMP Programming Model

- Parallel Mergesort that can perform hybrid execution over both host and device



Task granularity to avoid false sharing between host and device

“hetro” clause listing the shared writable variables

“hetro” clause to define the scope of hybrid execution

```
1. int THRESHOLD=omp_cache_grain()/INT_SIZE;
2. void mergesort(int left, int right){
3.   if(right-left > THRESHOLD) {
4.     int mid = left + (left+right)/2;
5.     #pragma omp task untied \
6.       firstprivate(left,mid) hetro(A:N)
7.       mergesort(left, mid);
8.       mergesort(mid+1, right),
9.       #pragma omp taskwait
10.      merge(left, mid, right);
11.    } else {
12.      sequentialSort(left, right);
13.    }
14. }
15. void main() {
16.   #pragma omp target map(to:N) \
17.     map(tofrom:A[0:N])
18.   #pragma omp parallel \
19.     firstprivate(A:N) hetro
20.   #pragma omp single
21.   mergesort(0, size-1);
22. }
```

HetroOMP Programming Model

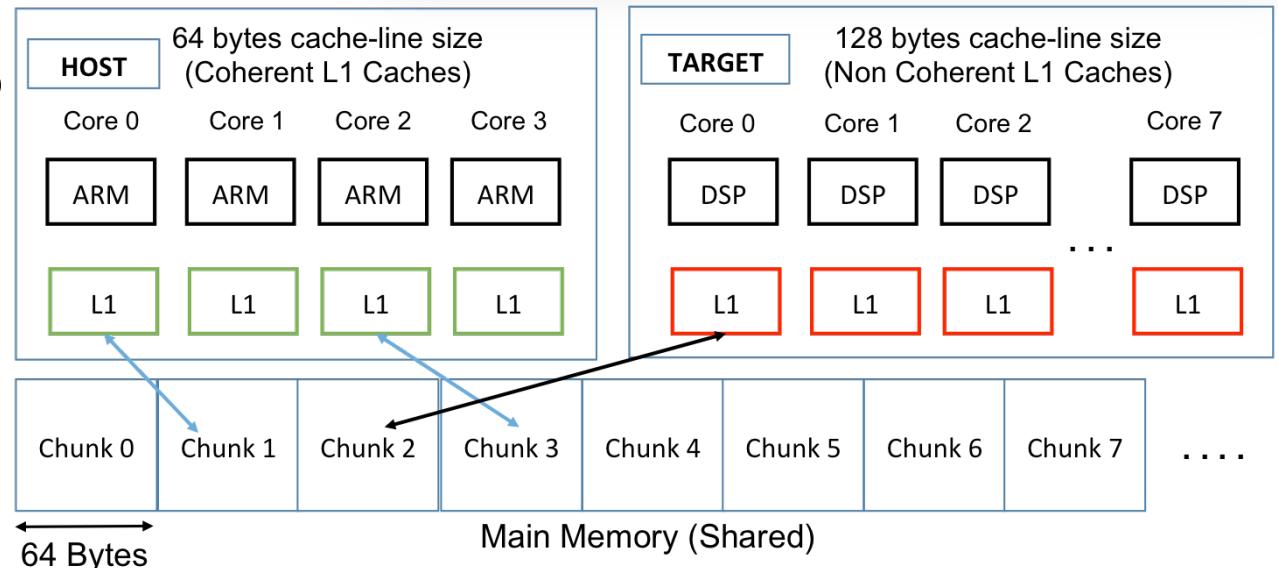
- **Avoiding false sharing between host and device**
 - Granularity
 - `#pragma omp task`
 - Loop tiling
 - `#pragma omp for`
 - Padding
 - Worst case



```

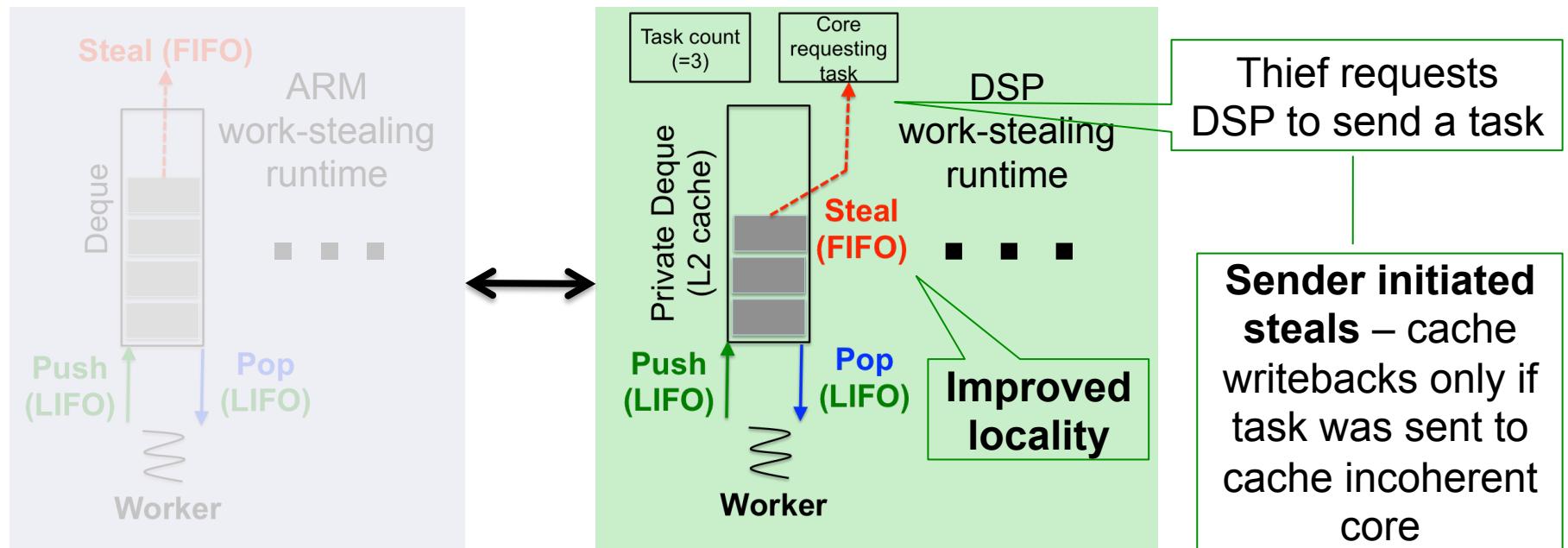
1. int THRESHOLD=omp_cache_grain()/INT_SIZE;
2. void mergesort(int left, int right){
3.   if(right-left > THRESHOLD) {
4.     int mid = left + (left+right)/2;
5.     #pragma omp task untied \
6.       firstprivate(left,mid) hetro(A:N)
7.     mergesort(left, mid);
8.     ....

```



HetroOMP Runtime

- OMP-to-X [1] translator modified to generate runtime code
- Hybrid work-stealing runtime
 - ARM work-stealing runtime same as HC-K2H
 - **Private deque (L2 cache) based DSP work-stealing runtime**



[1] Grossman et al., OpenMP as a High-Level Specification Language for Parallelism, IWOMP 2016



Experimental Evaluation

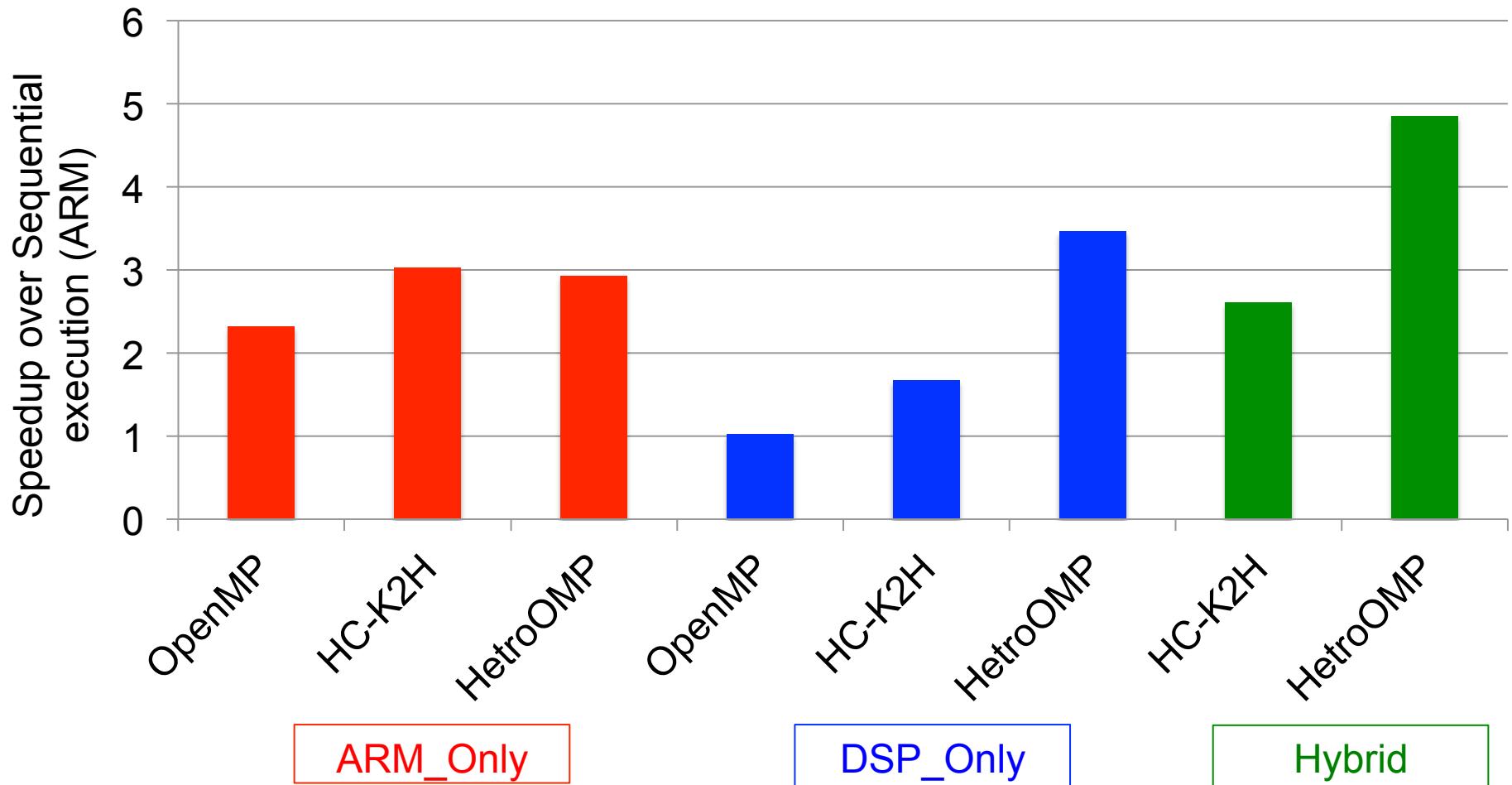
Methodology

- Benchmarks
 - Nested task and taskwait
 - Fibonacci
 - Matmul
 - Knapsack
 - MergeSort
 - Heat
 - Parallel for
 - Rodinia suite
 - BFS
 - Hotspot
 - Srad
 - LUD
 - B+Tree

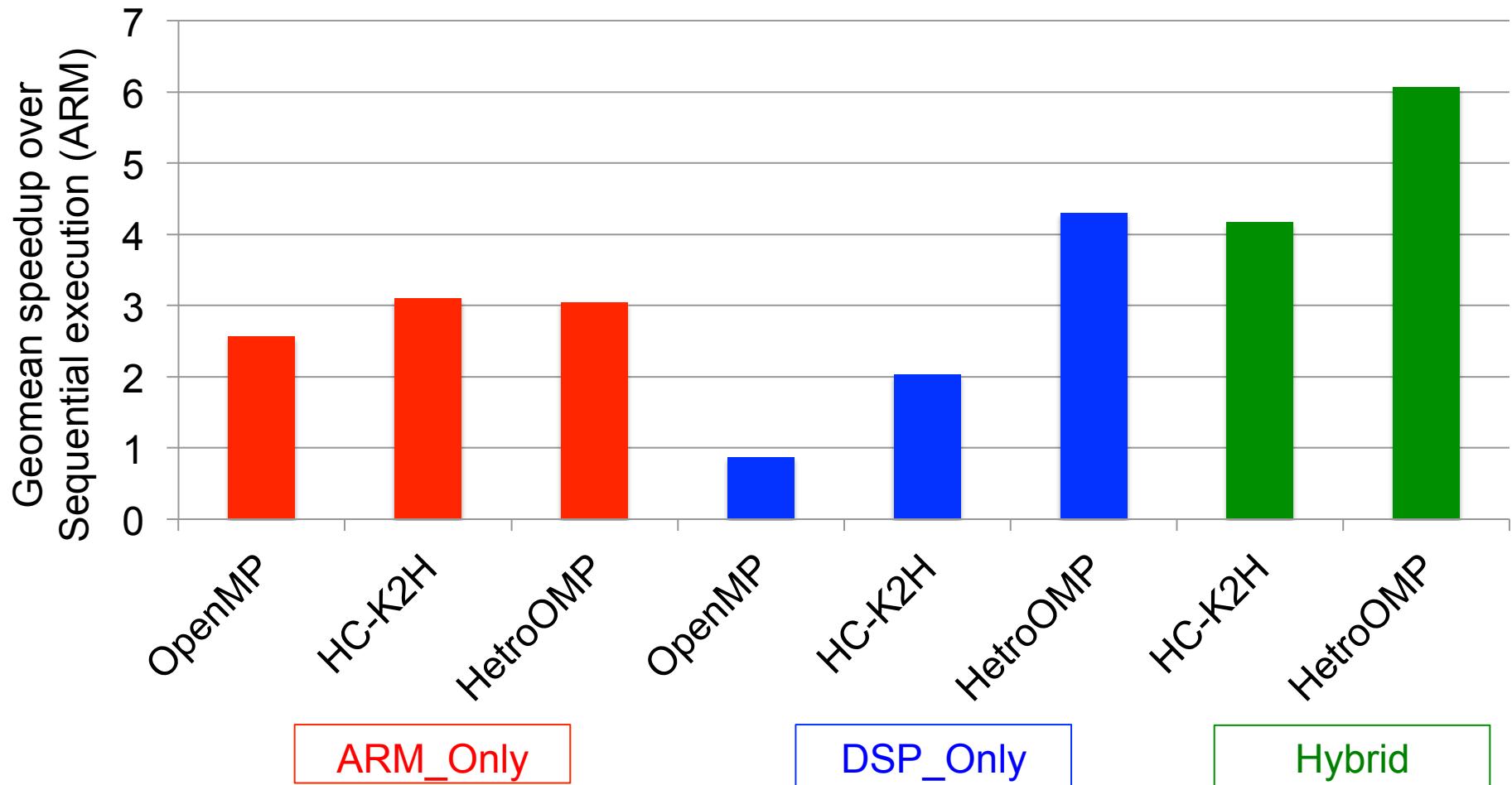
Runtime & Configurations

	ARM_Only (4 cores)	DSP_Only (8 cores)	Hybrid (12 cores)
OpenMP	✓	✓	✗
HC-K2H	✓	✓	✓
HetroOMP	✓	✓	✓

Speedup (MergeSort)

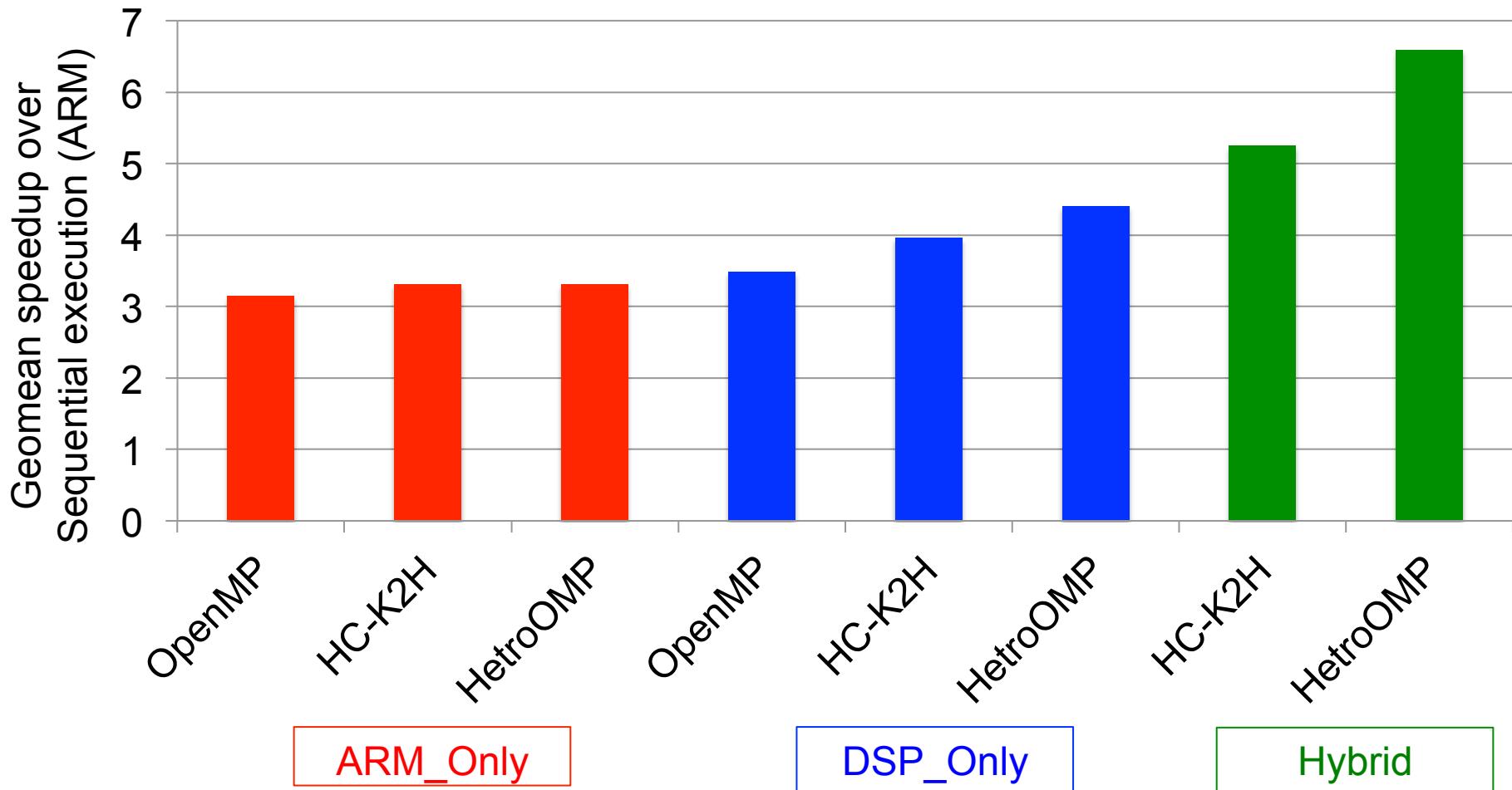


Geomean Speedup (All Tasking Types)



Note: we were unable to run Heat-OpenMP-DSP_Only

Geomean Speedup (All Parallel for)



Note: we were unable to run LUD-OpenMP-DSP_Only



Summary

- OpenMP accelerator model doesn't support hybrid execution across host and device
 - Wastage of CPU resources
- HetroOMP
 - Simple extension to OpenMP accelerator model for supporting hybrid execution
 - Uses hybrid work-stealing runtime
 - ARM work-stealing runtime built on traditional design (Cilk)
 - DSP work-stealing runtime uses private deques allocated on L2 cache instead of inbuilt hardware queues
 - Better locality
 - Fewer cache writebacks for task synchronization
 - Results
 - HetroOMP achieves geometric mean speedup of 3.6x over default OpenMP accelerator model



Backup Slides



HetroOMP Runtime

```
1. int THRESHOLD=omp_cache_grain()/INT_SIZE;
2. void mergesort(int left, int right){
3. .....
4. #pragma omp task untied \
5.         firstprivate(left,mid) hetro(A:N)
6. .....
7. #pragma omp taskwait
8. }
9. void main() {
10. .....
11. #pragma omp parallel \
12.         firstprivate(A:N) hetro
13. .....
14. }
```

```
1. int THRESHOLD= DSP_CACHE_LINE / INT_SIZE;
2. finish = new_scope(A, INT_SIZE * N);
3. finish->incoherent_core_stolen = false;
4. if (ARM_CORE) { /*push task on ARM deque*/ }
5. if (DSP_CORE) {
6.     /*push task on DSP private deque (on L2)*/
7.     if(/*DSP sending task to DSP/ARM){
8.         finish->incoherent_core_stolen = true;
9.     }
10. while /*tasks pending in current finish*/ {
11.     if/* tasks available on my deque */ {
12.         if/*DSP helping DSP/ARM || ARM helping DSP*/{
13.             finish->incoherent_core_stolen = true;
14.         }
15.         /* pop task from my deque and execute*/
16.     } else { /* steal tasks and execute */
17.     }
18. }
19. if(finish->incoherent_core_stolen == true) {
20.     cacheWBInv(finish->writable_sharedVars());
21. }
22. /*initialize A at device and start hybrid run*/
```