Cross-ISA Execution of SIMD Regions for Improved Performance

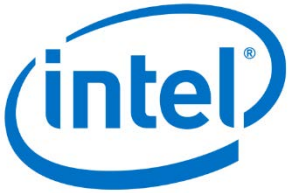
Yihan Pang, Robert Lyerly, Binoy Ravindran
System Software Research Group
Virginia Tech

Embracing ISA Heterogeneity

- Existing commercial products offers **NO** heterogeneity at ISA level

- Prior research shows that heterogeneity at ISA level can provides performance gain over a single-ISA heterogeneity

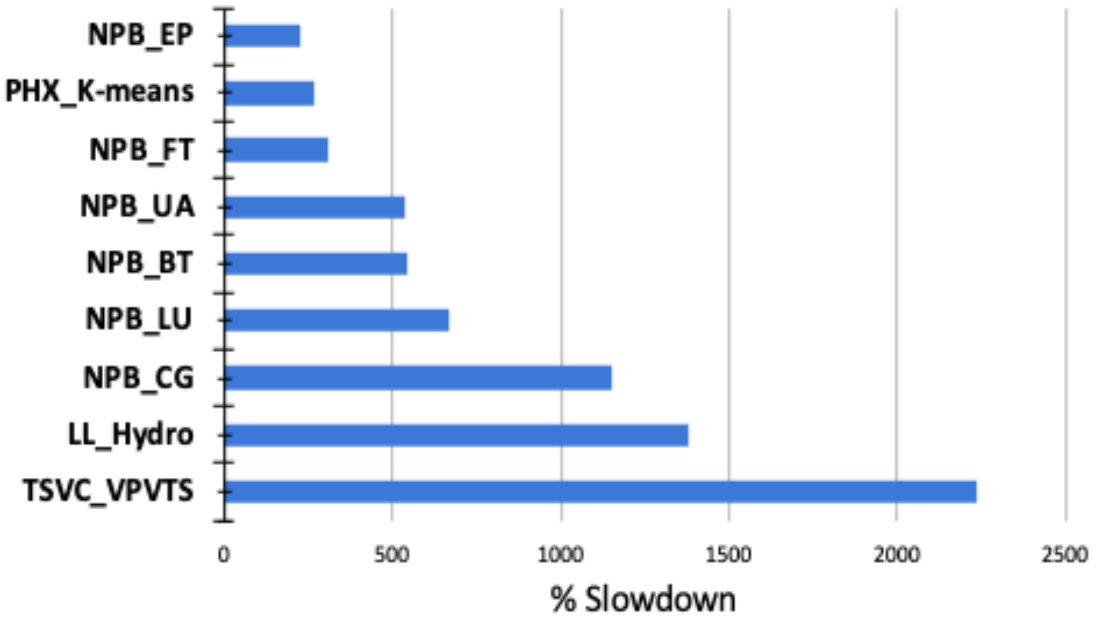
  - ONLY in emulated settings



- Can we embrace ISA heterogeneity using today's servers?

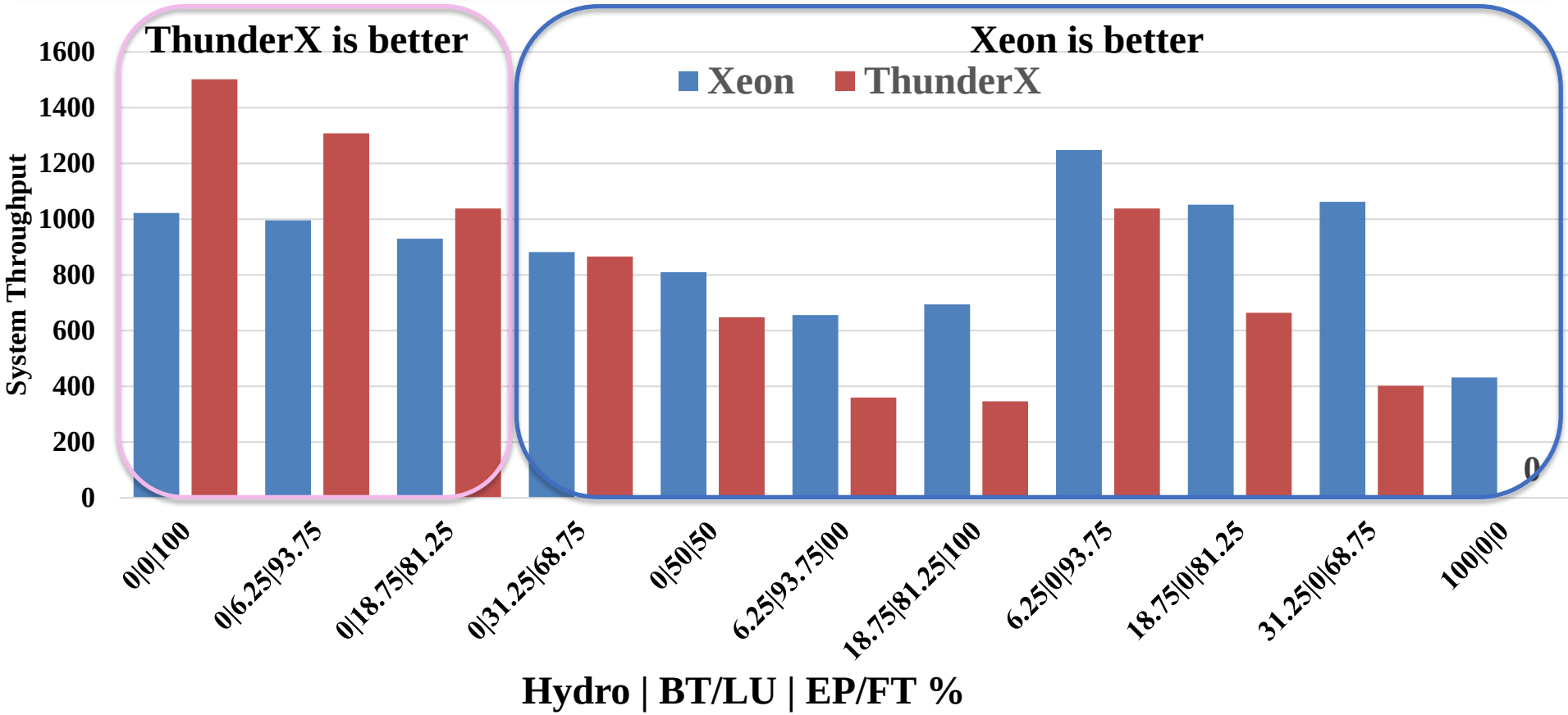
Explore Performance on ISA Different Servers

- Xeon Gold 5118 Server
  - x86-64 ISA (2018)
  - 12 core/24 physical threads
- Cavium ThunderX Server
  - ARMv8 ISA (2014)
  - 96 cores
- Benchmarks selected from 4 different HPC Suites
- Calculate single core, single thread performance



Our ThunderX cores are slower than its Xeon counterparts but differ in degree of slowdown

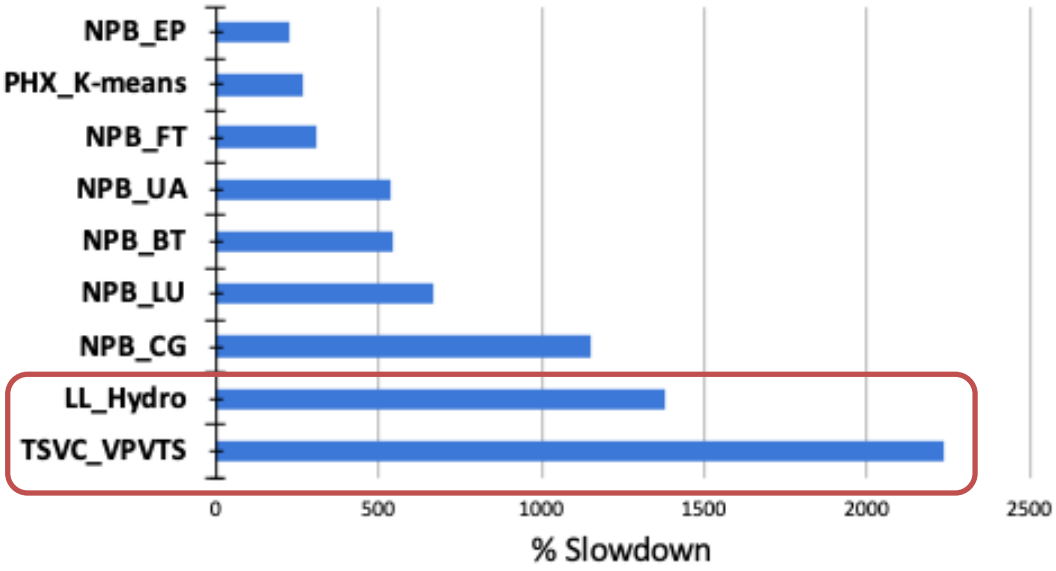
Is There One ISA Good For All?



No ISA is good for all workloads

Re-examine Benchmark Performance

- Hydro and VPVTS:
  - consist mainly of SIMD instruction.
- Latest ARM and x86 servers have very different SIMD support
- Our x86-64 server has SIMD width of 512 (Intel AVX-512)
- Our ARMv8 server only has a SIMD width of 128 (ARM-NEON)
- SIMD instructions gaining more attention in various application domains. (ML, CV, Cypto etc.)



SIMD + Heterogeneous-ISA servers

=

An Interesting platform

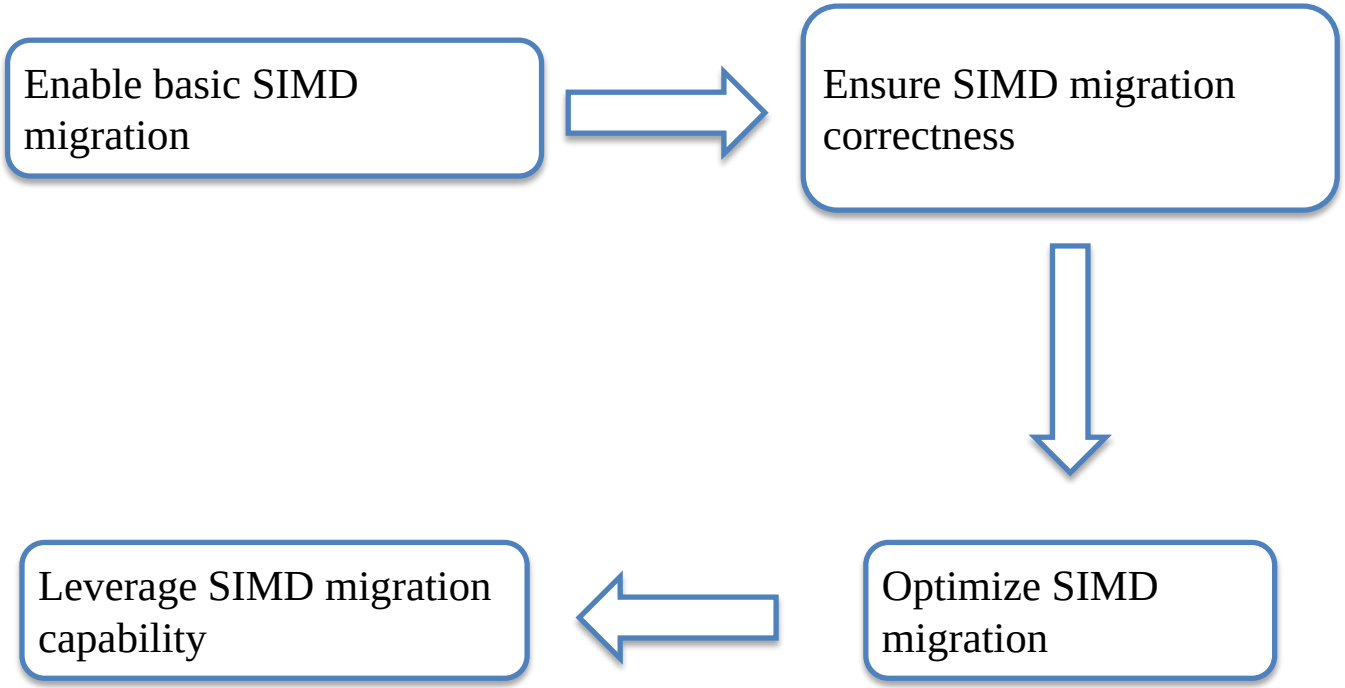
Contribution

- Extended Popcorn Linux compiler/run-time framework* to support a heterogeneous-ISA designs with cross-ISA SIMD migration capability
- Analyzed the effects of co-executing SIMD and non-SIMD workloads on a heterogeneous-ISA system to understand the impact of ISA heterogeneity and SIMD/non-SIMD workload composition on system throughput.
- Developed a migration aware scheduler to improve system throughput.

*Breaking the Boundaries in Heterogeneous-ISA Datacenters, A. Barbalace, R. Lyerly, C. Jelesnianski, A. Carno, H. Chuang, V. Legout, and B. Ravindran, ASPLOS'17

Design Outline

Outline:



Definitions

SIMD Region:

A piece of code within a program that has SIMD instructions

- can vary in execution time
- a program can contain any number of SIMD regions
- nested SIMD regions are considered as a single SIMD region

SIMD Workload:

A set of applications in which each application has at least one SIMD region.

Non-SIMD Workload:

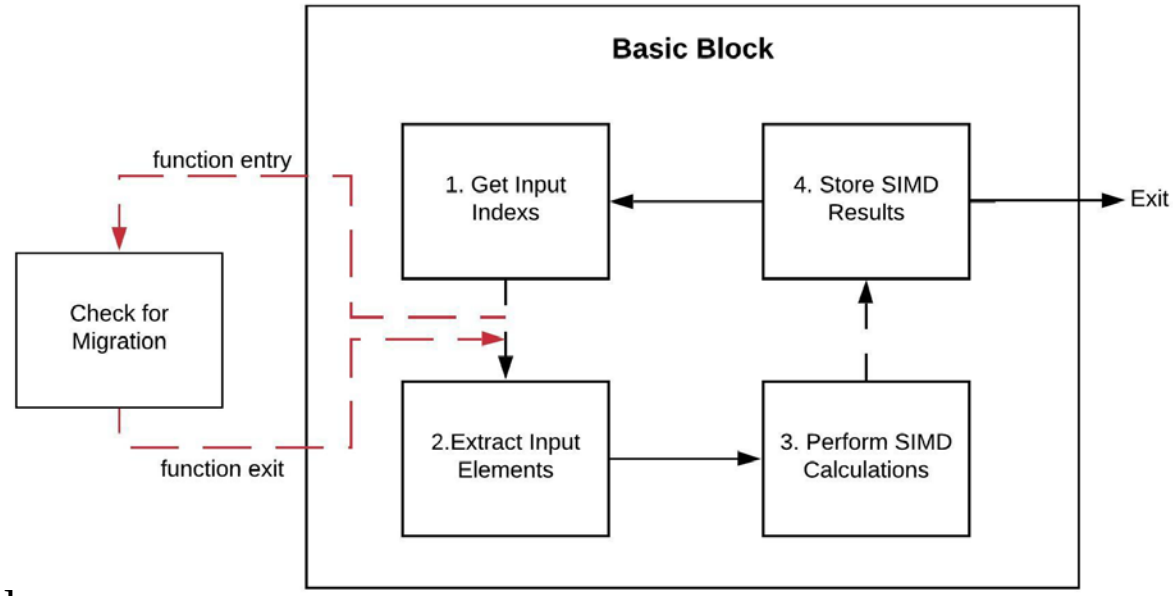
A set of applications that have **NO** SIMD region.

SIMD Intensive Program:

A program that spends more than 50% of its execution time in SIMD regions cumulatively.

Enable Basic SIMD Migration

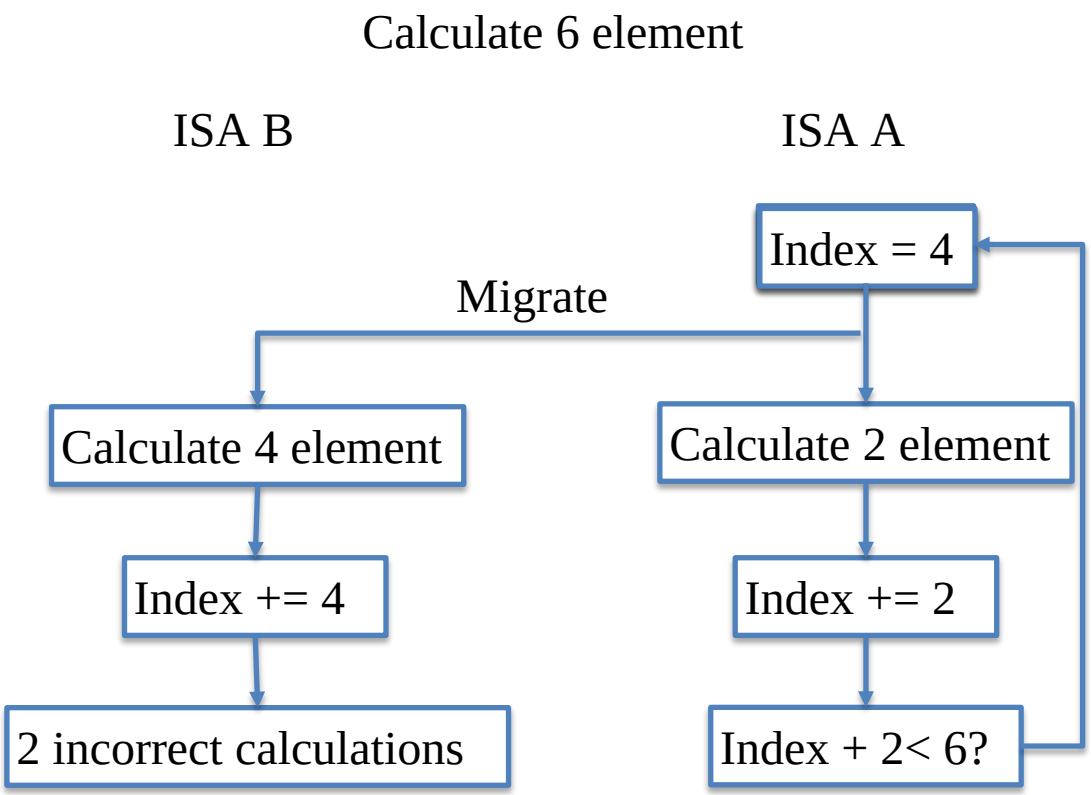
- Goal: Insert at least one migration point inside each SIMD region
- Migration happens at equivalence point*
- Function Boundary ∈ Equivalence Point
- Create migration point through dummy function call
- Insertion location selected based migration costs



*A Unified Model of Pointwise Equivalence of Procedural Computations, David G. von Bank, Charles M. Shub and Robert W. Sebesta ACM Trans. Program. Lang. Syst'1994

Ensure Migration Correctness

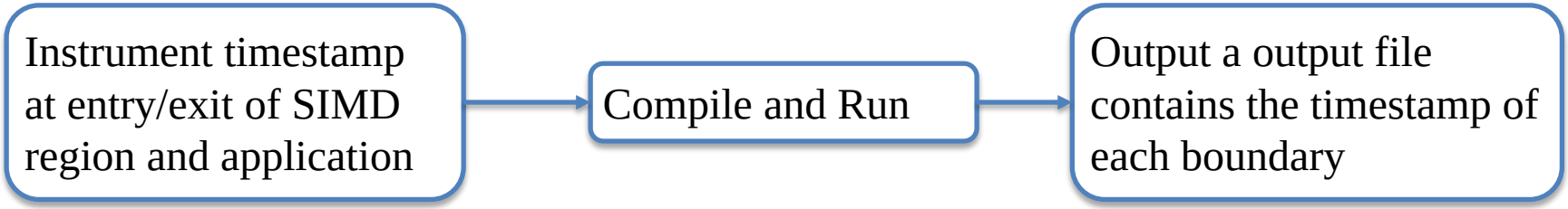
- Each selected ISA can have different SIMD width, optimization configuration
- Number of element being processed in a single SIMD loop can vary
- Correctness is not fully guaranteed after migration if NO modification is done
- Unroll the loop code on both ISAs to the least common multiple of the SIMD element(s) processed in respective loops



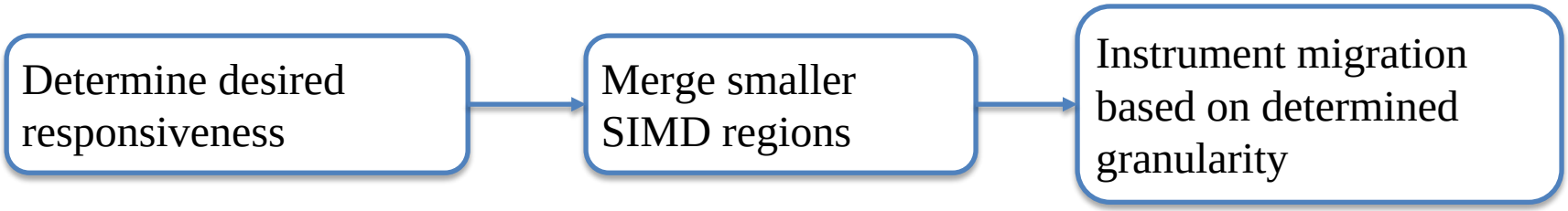
Optimize SIMD Migration Cost

- Checking for migration incurs performance overhead
- We propose a two step profile-guided optimization approach
  - Inspired by LLVM's built-in Profile Guided Optimization(PGO)

Step 1: Profiling



Step 2: Optimization



Leverage SIMD Migration Capability

- How to leverage this new SIMD migration capability?
- We propose a SIMD-aware scheduler
- Central Scheduling Idea:

The speedup gained from executing an application on the optimal ISA core should outweigh the slowdown other applications suffer from not running on that core

$$\frac{speedup\ of\ App\ X}{speedup\ of\ App\ Y} > 1$$

- Further instrumentation to our framework so that each application shall communicate to scheduler at:
  - -program entry/exit
  - -after program migration
- Scheduler uses slowdown information gathered from profiling and server usage to make additional scheduling decisions
- To reduce complexity, we categorize all applications into three slowdown groups: high, medium and low

Experimental Setup & Methodology

- **Server Configurations**:

Het-dynamic (our design w/ instrumentation)
Het-static
x86-static
ARM-static
} (baseline w/o instrumentation)

- **Experiment Length**:

75 min per run

- **Evaluation parameter:**

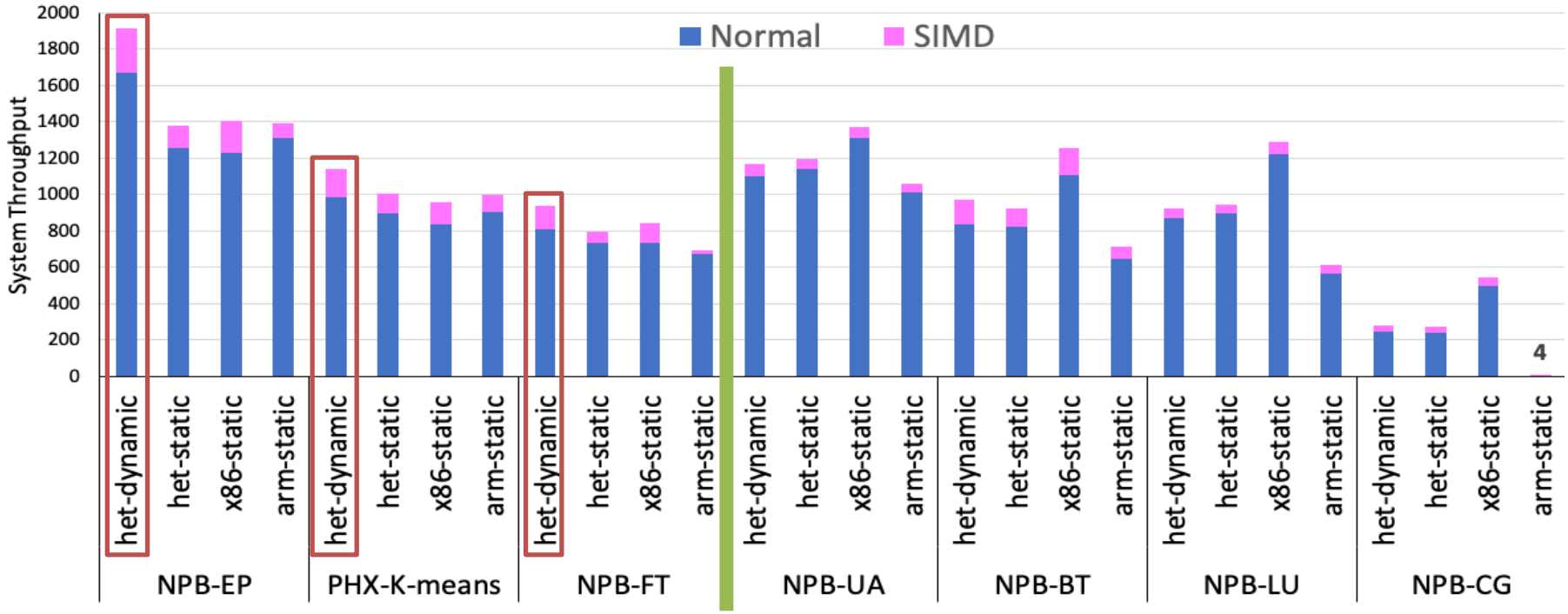
# of benchmark completed

- **Workload Generation**:

Mimic dynamic workload scenario

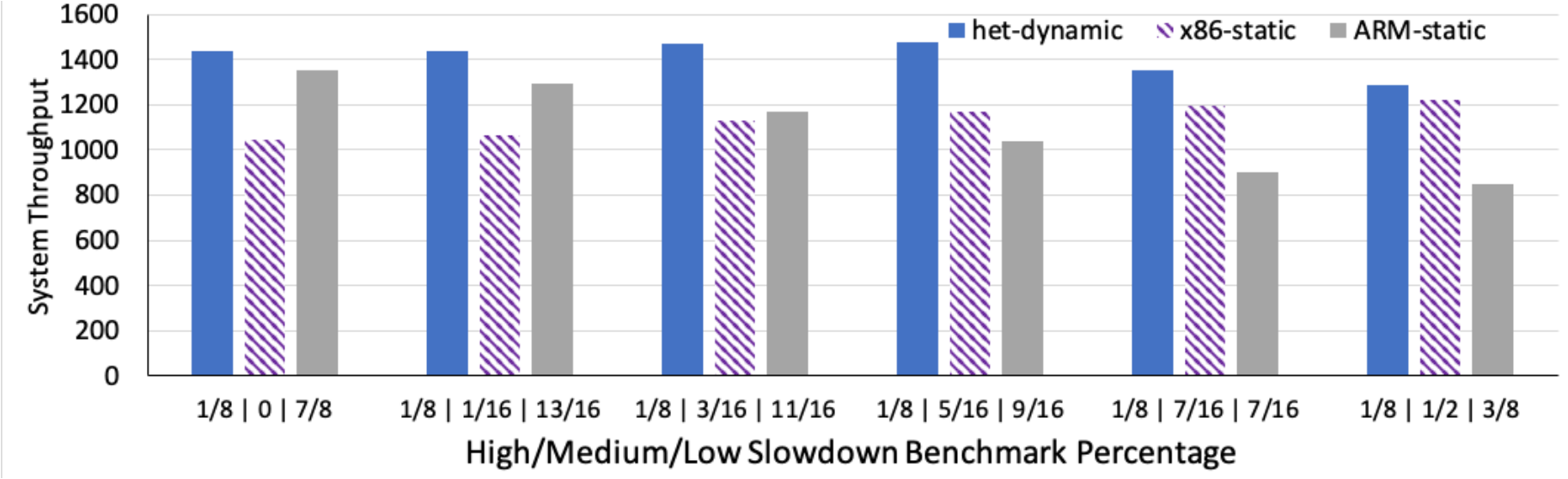
- Predefined SIMD/non-SIMD benchmark composition
- In first iteration, assigns benchmark based on affinity if possible until all core are occupied
- Random assign benchmark from the remaining workload
- Regenerate workload with same ratio if workload is empty

Result: Two Benchmark Scenario (1/8)



- Performance Gain in EP, K-means and FT. Best: ~**36%**
- Average **6.3%** non-SIMD benchmarks migrated to ARM
- Het-dynamic outperforms het-statics

Result: Multi-workload (1/8)



- Average performance gain of **14.6%**. Best: ~**26%**
- Average **23.1%** of total benchmarks migrated to ARM
- Average **24.7%** of total benchmarks migrated back eventually

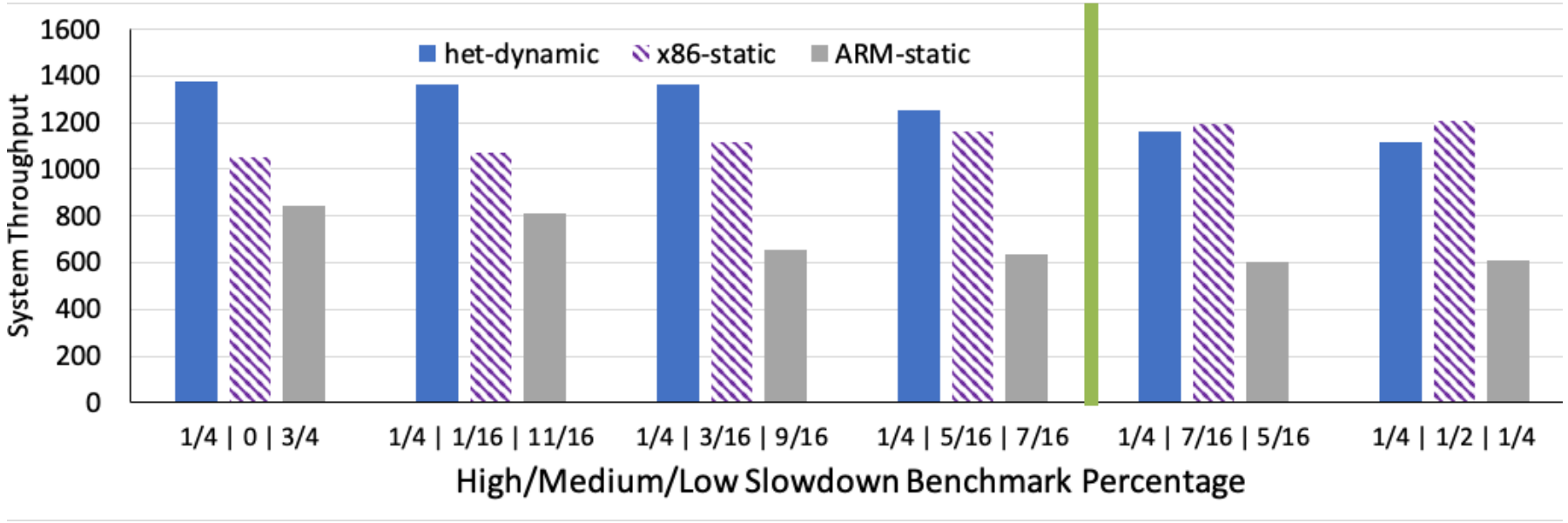
Conclusion

- Efficiently using ISA affinity and dynamically migrating applications to use ISA optimal cores in heterogeneous-ISA systems can result in significant performance gains over homogeneous systems.

"No one ISA/micro-architecture fits all workloads."

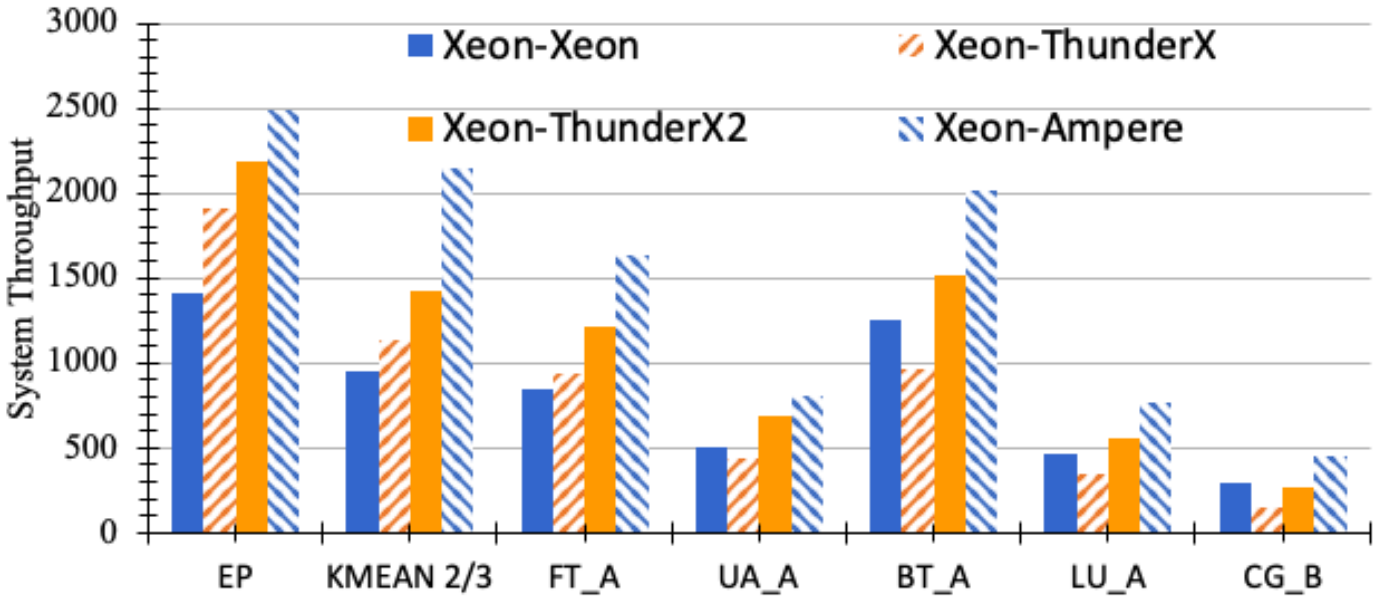
Thank you.
Q & A

Result: Multi-workload (1/4)



- Performance gain in 4/6 test cases, Best: ~31.1%
- Average 28% of total benchmarks migrated to ARM
- Average 24.4 % of total benchmarks migrated back eventually

Discussion & Future Direction



Since

Het-dynamic >= Het-static in Xeon-ThunderX setup

Our design on newer machine (ThunderX2 and Ampere) can potentially yield even more promising results.

Optimize SIMD Migration Cost