IPDPS 2025 • Milano, Italy

SYCL FOR HPC: ADAPTING TO DIVERSE CPU ARCHITECTURE

ASHISH BISHT, ANIKET P. GARADE, H.V. DEEPIKA, P. HARIBABU, S.A. KUMAR, S. D. SUDARSAN

TABLE OF CONTENTS

1. Introduction
   - What is SYCL ?
   - Motivation ?
2. Approach
   - Compilation Model
   - Execution Model
3. Experimentation
   - Architecture used
   - Applications used
   - Performance
   - Scaling
4. Conclusion

INTRODUCTION

- Modern HPC systems comprise of diverse range of hardware processing units
  - CPU
  - GPU
  - AI Accelerators or any other XPU
- Every processing hardware has its own programming language.

Lot of efforts towards porting of applications.

WHAT IS SYCL?

SYCL is a standard by Khronos. It offers following features:

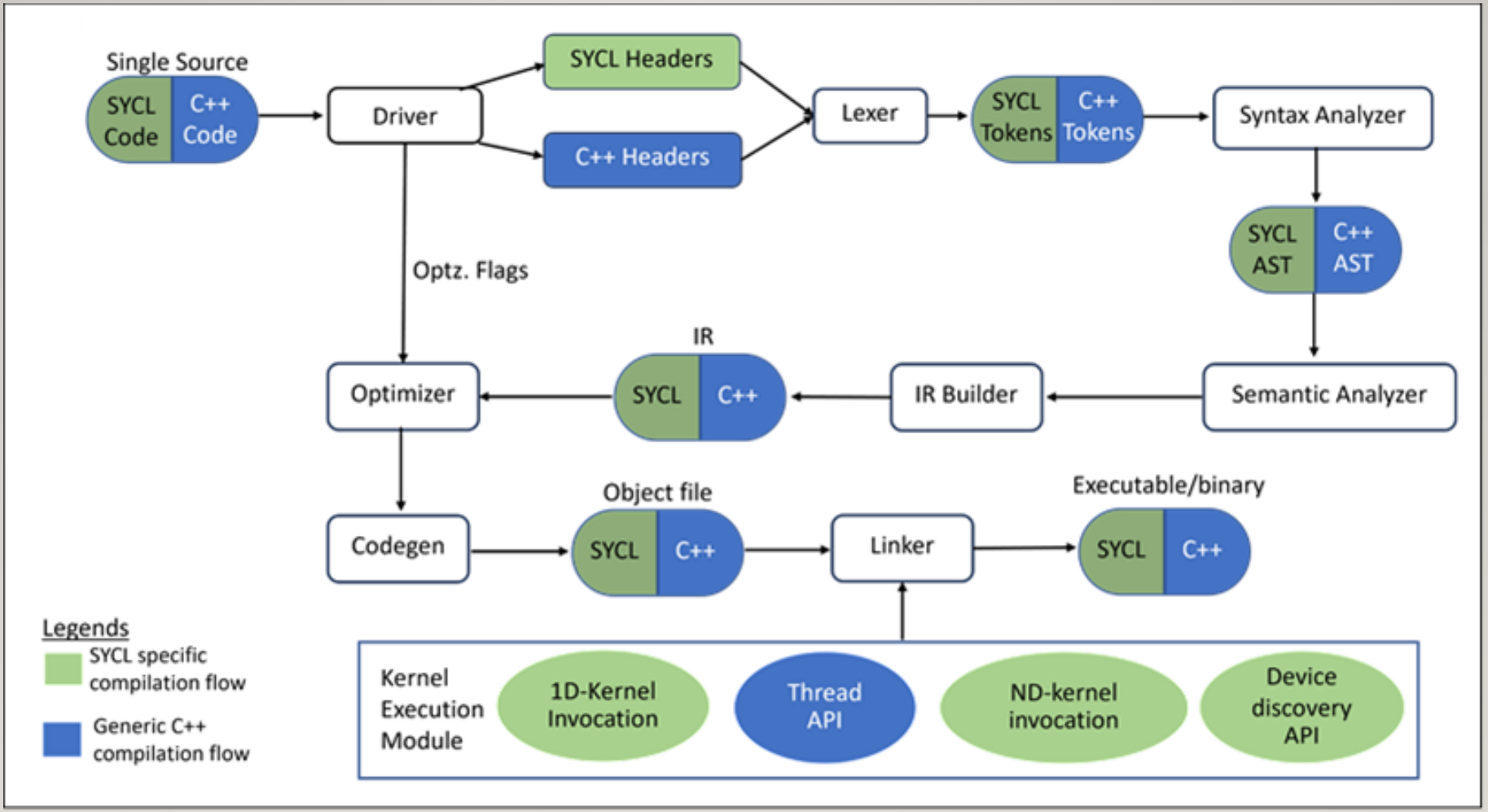
- Programming for Heterogeneous Computing
- Uses modern C++ features
- Single source

MOTIVATION

- Architectures present in CPUs:
  - X86 (Intel & AMD)
  - ARM (NVIDIA's Grace, Fujitsu A64FX)
  - RISC (IBM's POWER-x)
- To develop a scalable and flexible implementation which is not associated to only one specific hardware.

APPROACH

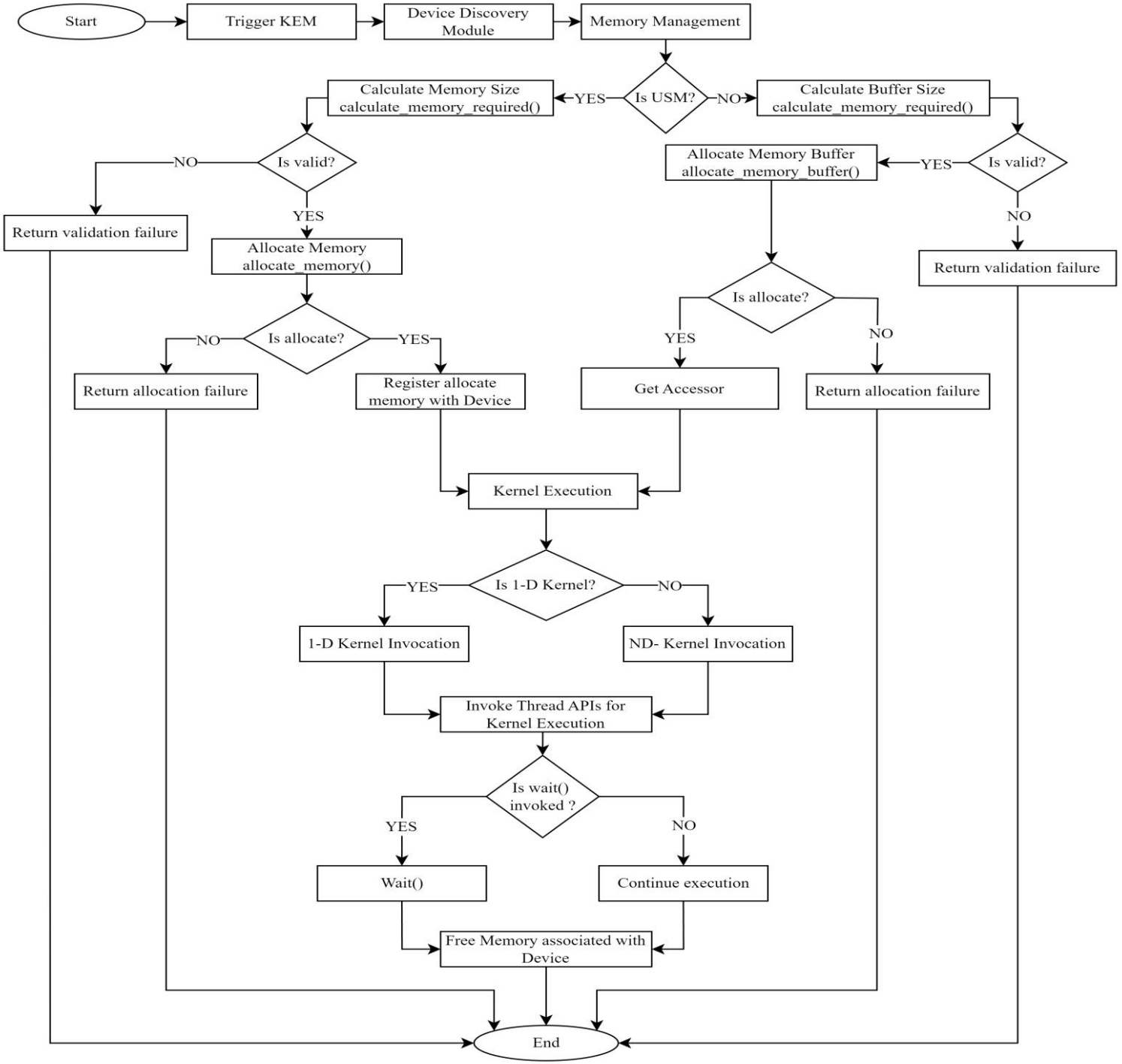
COMPILATION MODEL

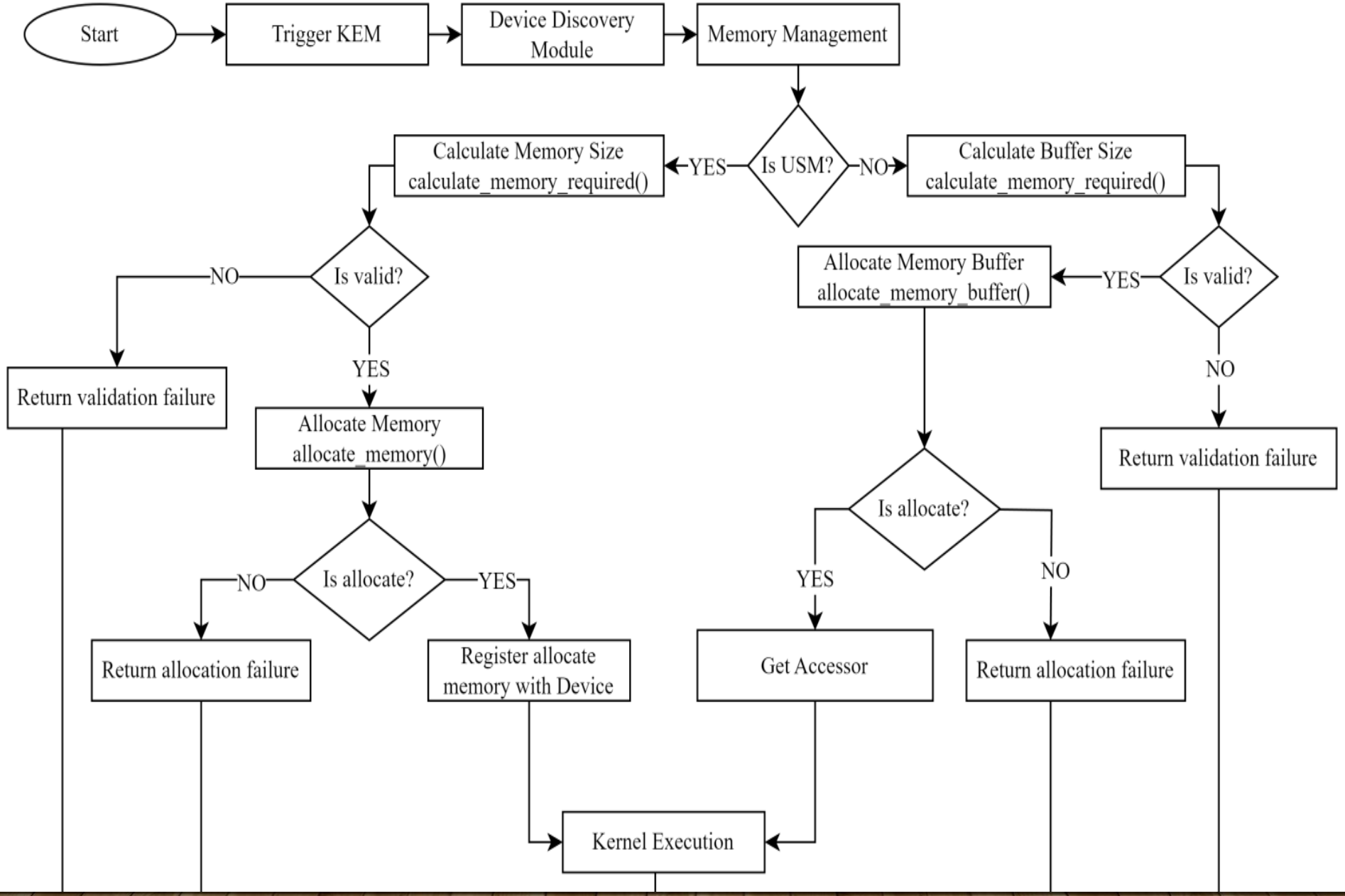


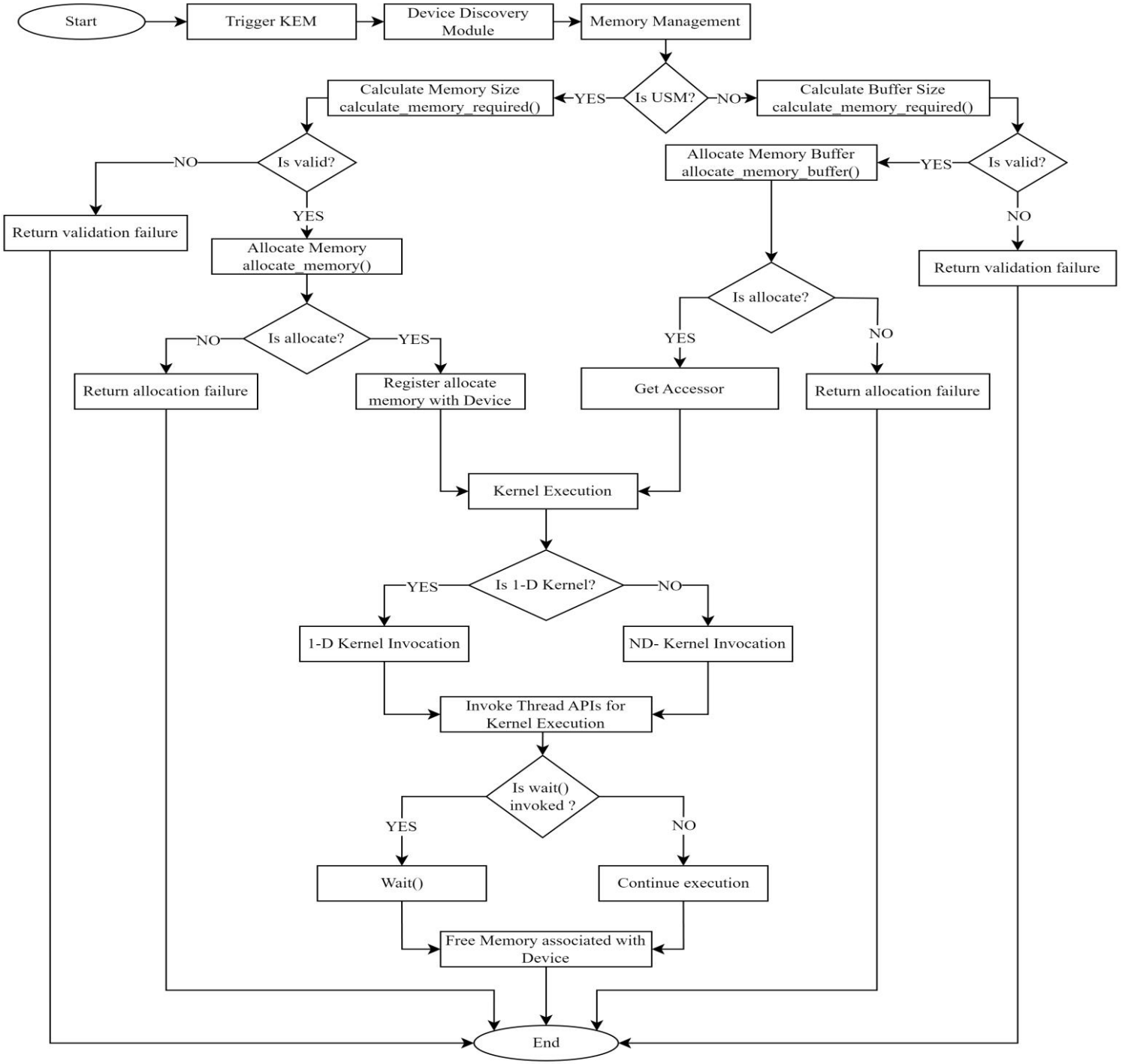
KERNEL EXECUTION MODULE

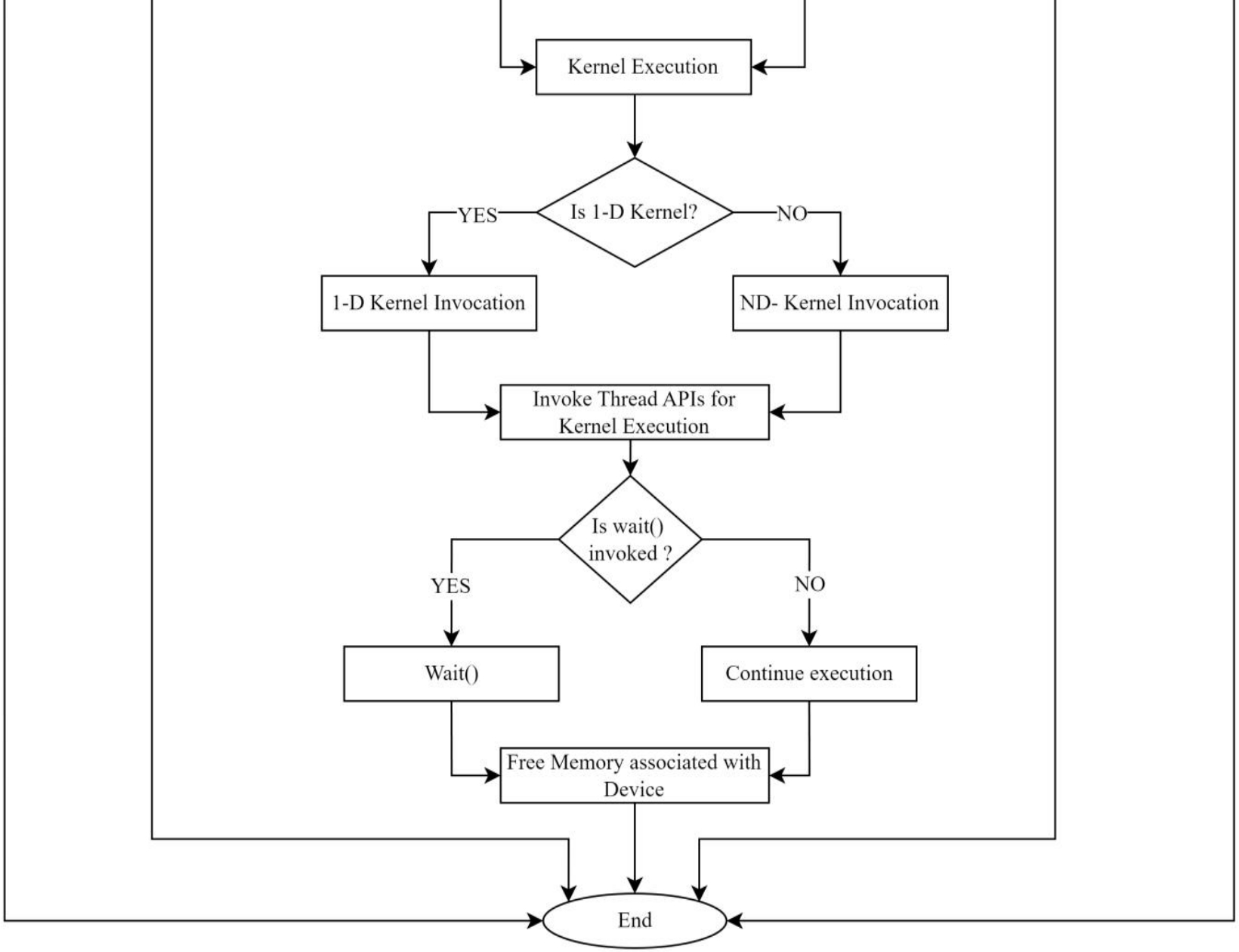
- **Device Discovery API**
  - Identify the underlying CPU architecture
  - Stores the hardware info
- **1-D Kernel invocation**
  - Creates and manages threads based on device info
  - Leverages the native thread APIs
- **N-D kernel invocation**
  - Converts n-dimensional kernels into single dimensional
  - Leverages the native thread APIs
- Thread API

EXECUTION MODEL


Start
Trigger KEM
Device Discovery Module
Memory Management
Is USM?
YES
Calculate Memory Size
calculate_memory_required()
NO
Calculate Buffer Size
calculate_memory_required()
Is valid?
NO
Return validation failure
YES
Allocate Memory
allocate_memory()
Is allocate?
NO
Return allocation failure
YES
Register allocate memory with Device
Allocate Memory Buffer
allocate_memory_buffer()
YES
Is valid?
NO
Return validation failure
Is allocate?
YES
Get Accessor
NO
Return allocation failure
Kernel Execution
Is 1-D Kernel?
YES
1-D Kernel Invocation
NO
ND- Kernel Invocation
Invoke Thread APIs for Kernel Execution
Is wait() invoked ?
YES
Wait()
NO
Continue execution
Free Memory associated with Device
End


Start
Trigger KEM
Device Discovery Module
Memory Management
Is USM?
YES
NO
Calculate Memory Size calculate_memory_required()
Calculate Buffer Size calculate_memory_required()
Is valid?
NO
YES
Return validation failure
Allocate Memory allocate_memory()
Is allocate?
NO
YES
Return allocation failure
Register allocate memory with Device
Allocate Memory Buffer allocate_memory_buffer()
YES
Is valid?
NO
Return validation failure
Is allocate?
YES
NO
Get Accessor
Return allocation failure
Kernel Execution


Start
Trigger KEM
Device Discovery Module
Memory Management
Is USM?
YES
Calculate Memory Size
calculate_memory_required()
NO
Calculate Buffer Size
calculate_memory_required()
Is valid?
NO
Return validation failure
YES
Allocate Memory
allocate_memory()
Is allocate?
NO
Return allocation failure
YES
Register allocate memory with Device
Allocate Memory Buffer
allocate_memory_buffer()
YES
Is valid?
NO
Return validation failure
Is allocate?
YES
Get Accessor
NO
Return allocation failure
Kernel Execution
Is 1-D Kernel?
YES
1-D Kernel Invocation
NO
ND- Kernel Invocation
Invoke Thread APIs for Kernel Execution
Is wait() invoked ?
YES
Wait()
NO
Continue execution
Free Memory associated with Device
End


Kernel Execution
Is 1-D Kernel?
YES
NO
1-D Kernel Invocation
ND- Kernel Invocation
Invoke Thread APIs for Kernel Execution
Is wait() invoked ?
YES
NO
Wait()
Continue execution
Free Memory associated with Device
End

PERFORMANCE

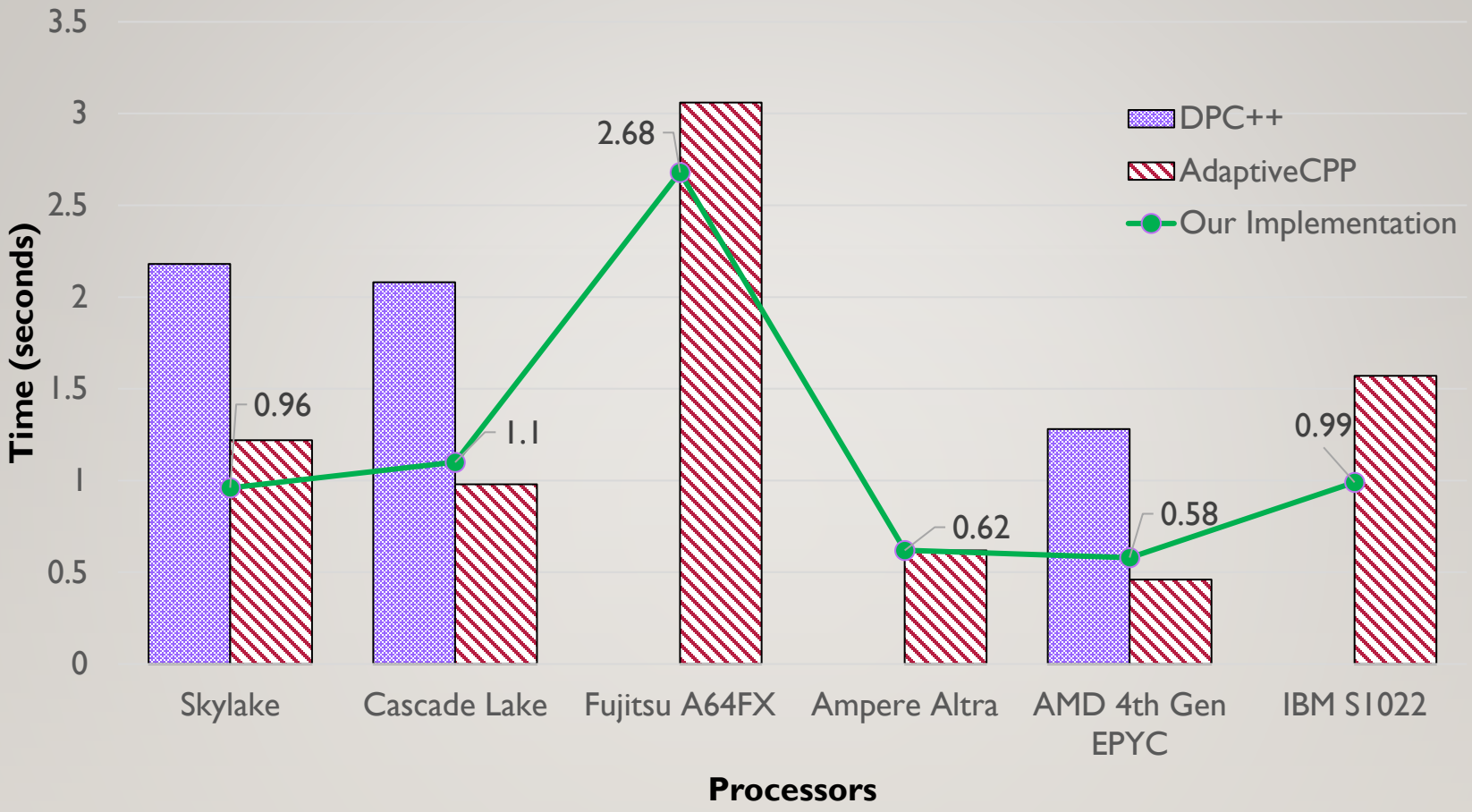
ARCHITECTURES/PROCESSORS USED

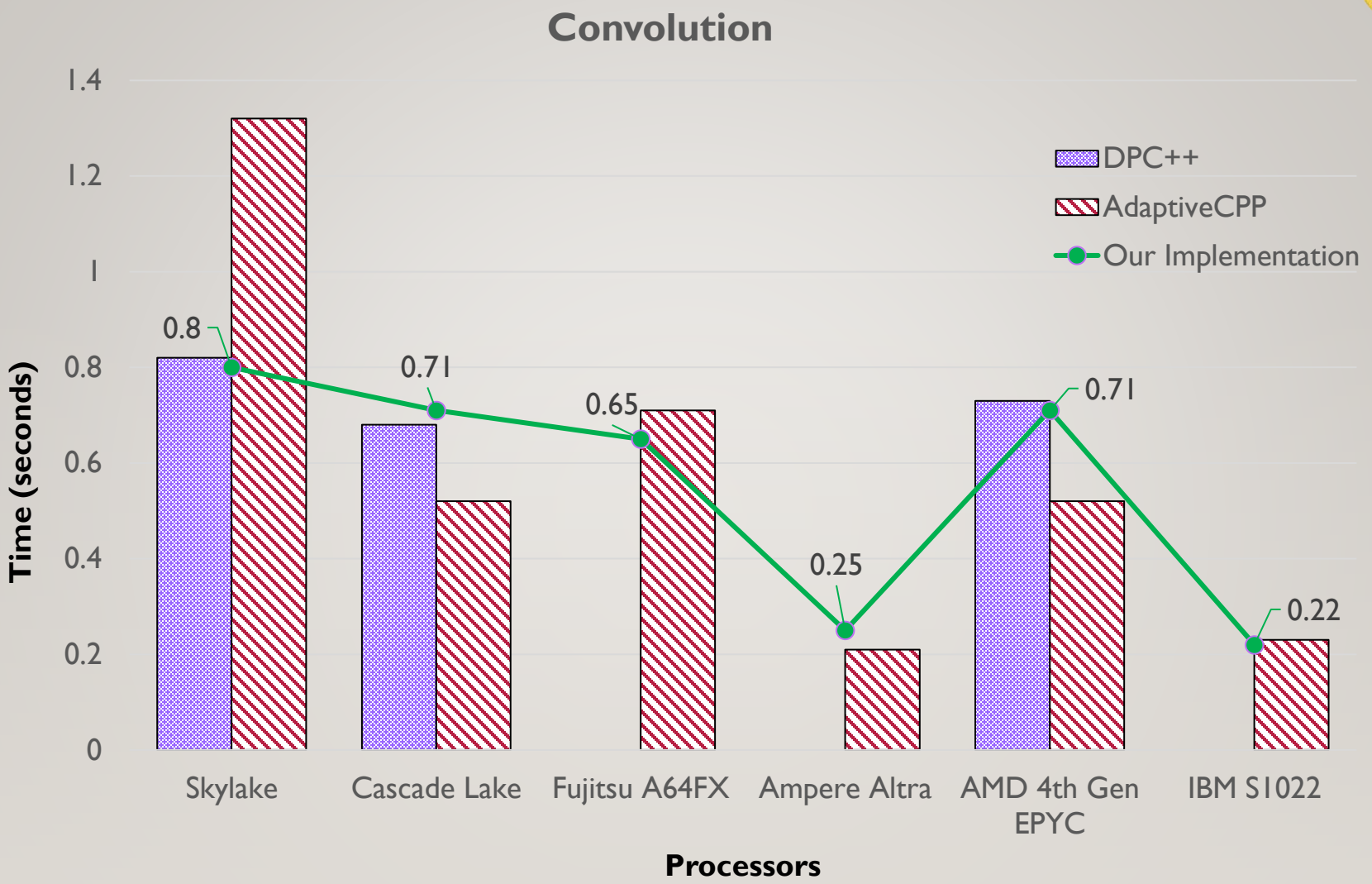
- X86
  - Intel Xeon Gold 6312 (Skylake)
  - Intel Xeon Platinum 8268 (Cascade Lake)
  - AMD Xeon EPYC 4th Gen
- ARM
  - Fujitsu A64FX
  - Ampere Altra
- Power 10
  - IBM S1022

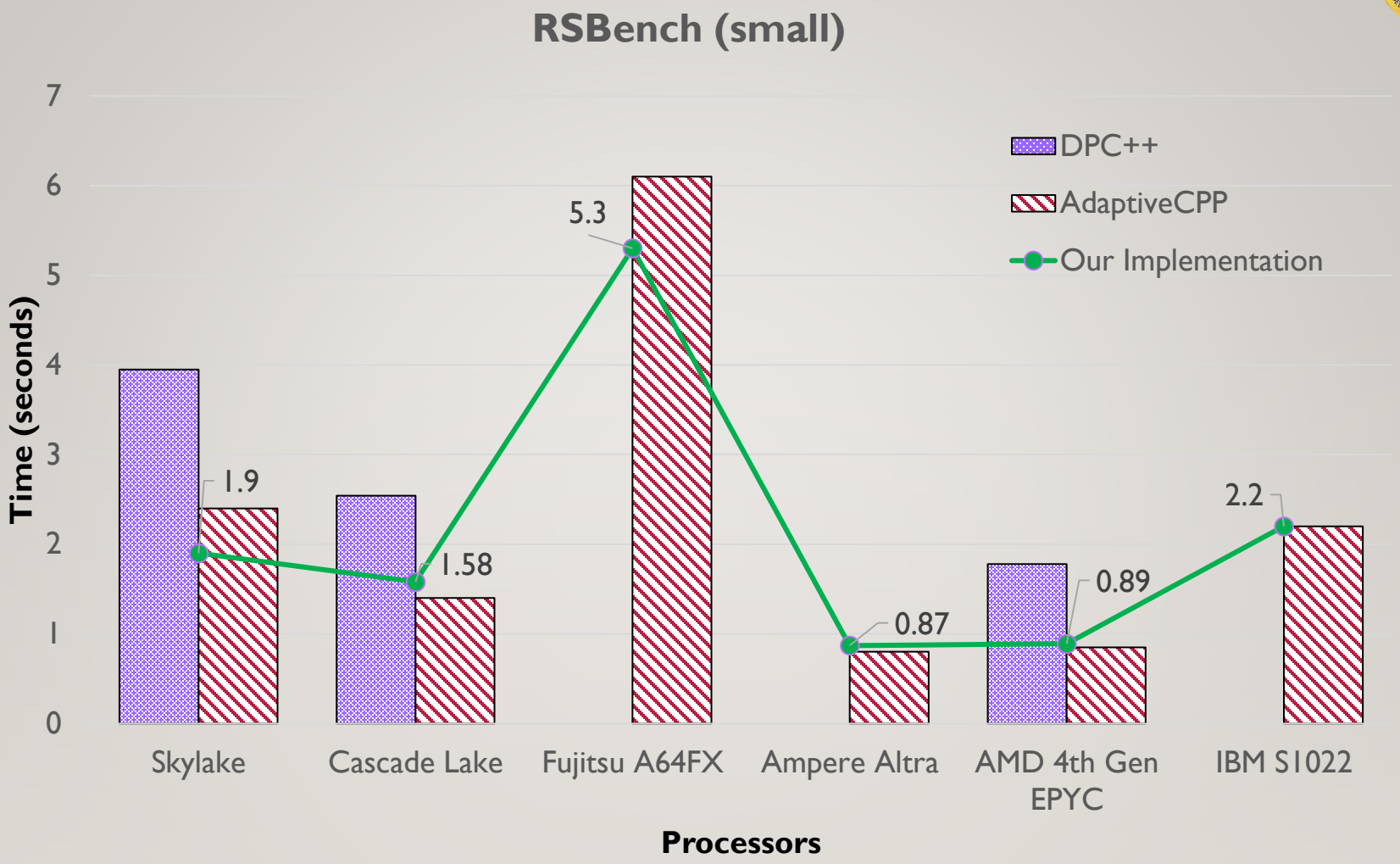
APPLICATIONS

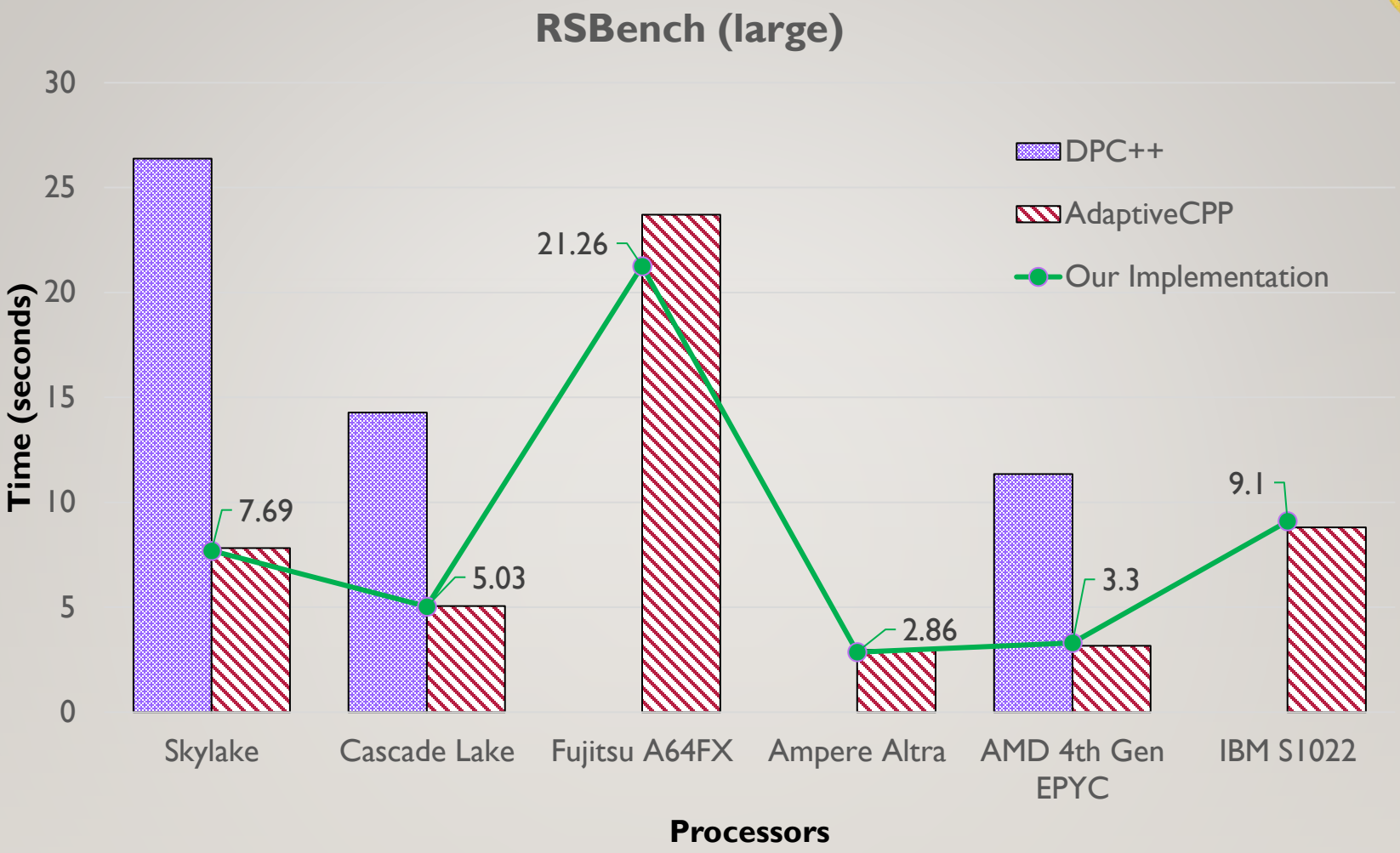
- Heat Equation
  - This application is used to describe the distribution of heat or variation in temperature in a given region over time.
- Convolution
  - Fundamental operation in image processing and computer vision used for tasks like edge detection, image filtering and feature extraction.
- RSBench
  - Represents a key computational kernel of the Monte Carlo neutron transport algorithm.

Heat Equation




Convolution
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Time (seconds)
DPC++
AdaptiveCPP
Our Implementation
0.8
0.71
0.65
0.25
0.71
0.22
Skylake
Cascade Lake
Fujitsu A64FX
Ampere Altra
AMD 4th Gen EPYC
IBM S1022
Processors


RSBench (small)
Time (seconds)
0
1
2
3
4
5
6
7
DPC++
AdaptiveCPP
Our Implementation
1.9
1.58
5.3
0.87
0.89
2.2
Skylake
Cascade Lake
Fujitsu A64FX
Ampere Altra
AMD 4th Gen EPYC
IBM S1022
Processors

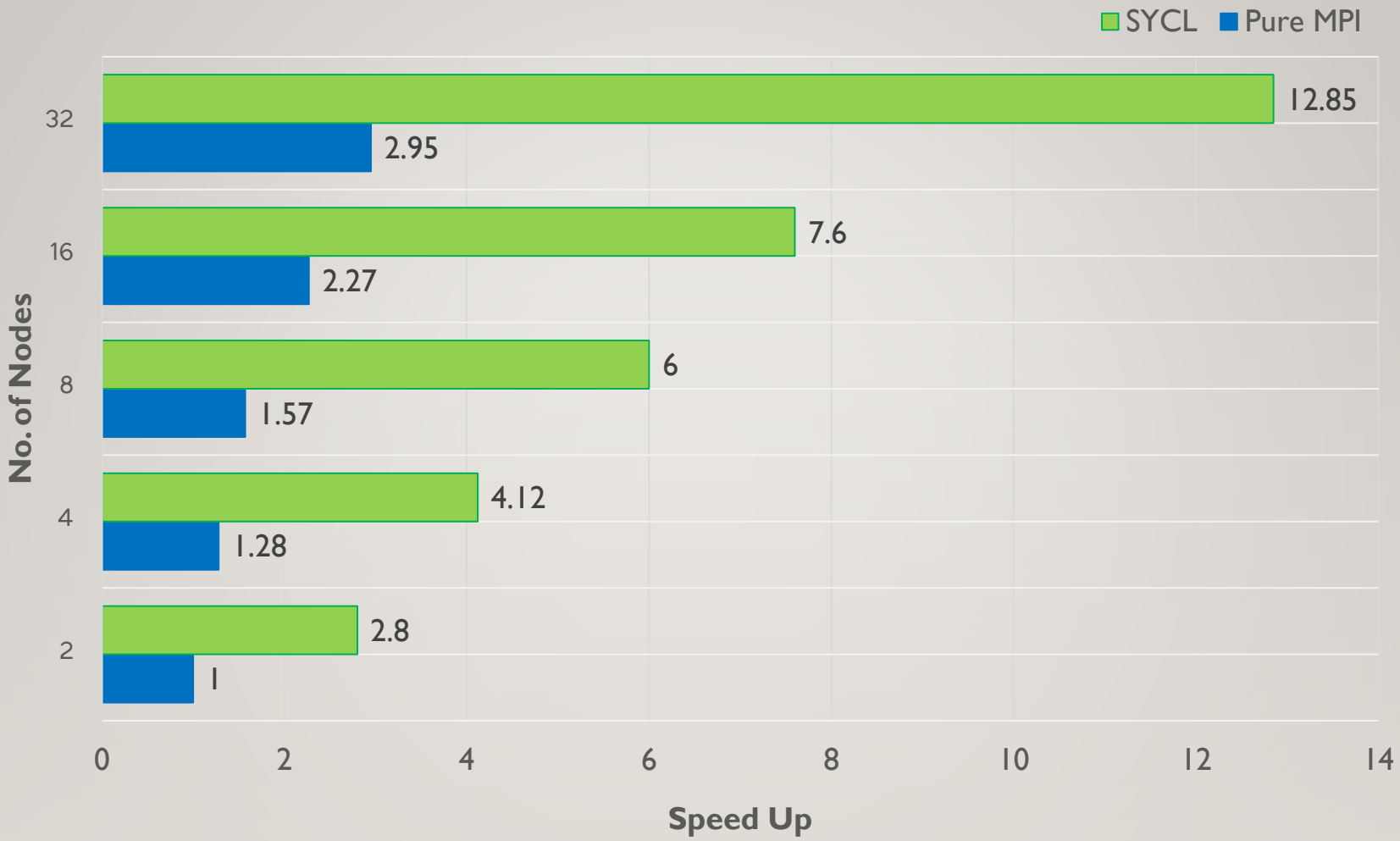

RSBench (large)
30
25
20
15
10
5
0
Time (seconds)
DPC++
AdaptiveCPP
Our Implementation
7.69
5.03
21.26
2.86
3.3
9.1
Skylake
Cascade Lake
Fujitsu A64FX
Ampere Altra
AMD 4th Gen EPYC
IBM S1022
Processors

APPLICATION SCALING

How it performs on a HPC cluster ?

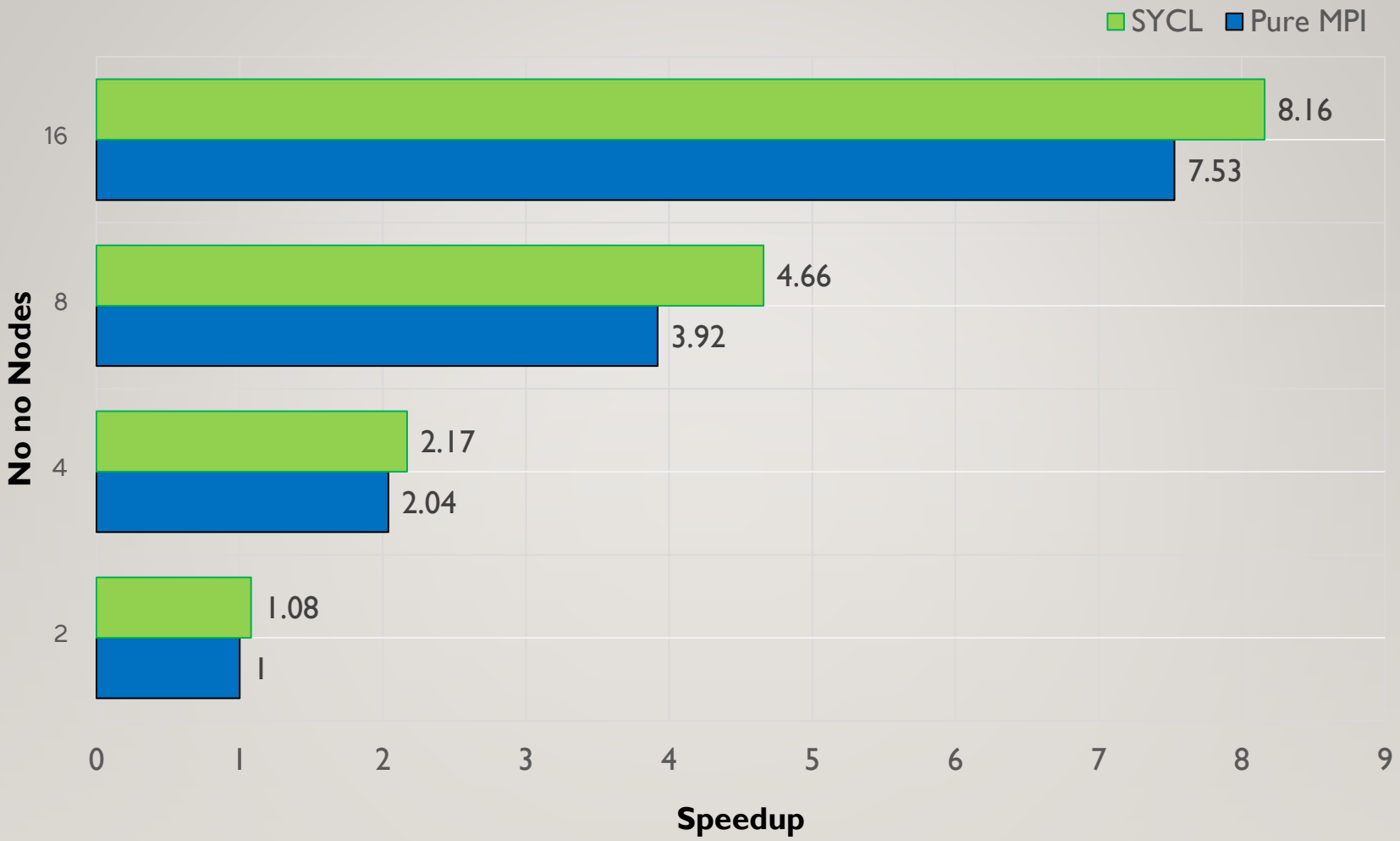
Configuration	Value
Processor	Intel Xeon Platinum 8268
Memory	192 GB
No. of cores/socket	24
No of socket/node	2
Cluster topology	Fat-tree
Peak Bandwidth	100 Gbps
Interconnect	Mellanox Infiniband

MonteCarlo PI (Cluster)



Using Monte-Carlo method for 10 million points.

GEMM (cluster)



Matrix size = 8192 x 8192

CONCLUSION

- We are able to maintain the portability across multiple architectures like X86, ARM and POWER.
- Our implementations performs better by a factor of 1.2x to 1.5x on Fujitsu A64FX and Intel Skylake architectures, while delivering comparable performance to existing SYCL implementations on AMD and Power platforms.
- Our implementation uses native thread APIs making it a versatile SYCL solution.
- Our implementation showcases scaling on HPC cluster environment.

QUERIES ?