Evaluating Shared Memory Heterogeneous Systems Using Traverse-compute Workloads

Yanwen Xu
University of California, Santa Cruz

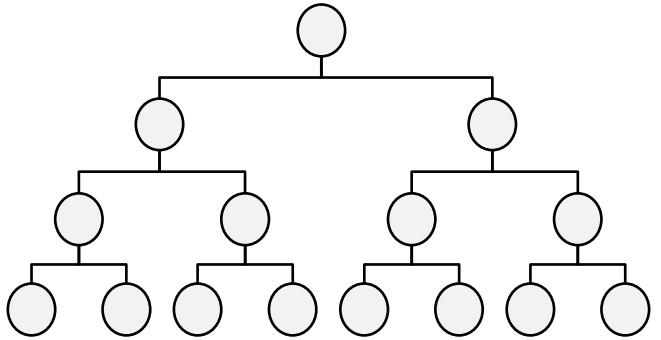
Ang Li
Princeton University

Tyler Sorensen
University of California, Santa Cruz

Second Workshop on Open-Source Computer Architecture Research (OSCAR 2023) Orlando, Florida

Highlights

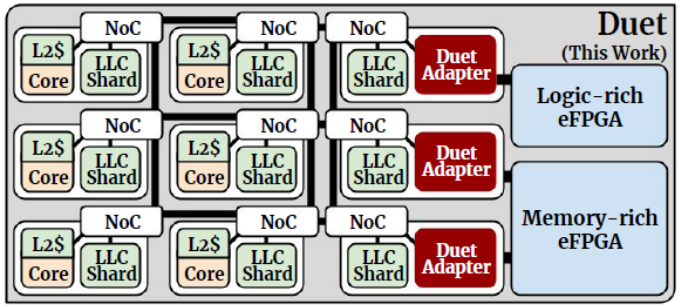
Many applications in edge computing can benefit from utilizing tree data structures to accelerate their workloads



Showed how open-source hardware can be leveraged to accelerate a specific class of tree algorithms, which we call *traverse-compute*

Evaluated open-source heterogeneous architecture called Duet, using a recently published open-source framework and benchmark suite Redwood and Grove

- w/ 9 pragmatic traverse-compute applications



Achieved

- 13.53x highest speedup
- 6.43x geomean speedup

Insight: Traverse-compute workload has natural heterogeneous decompositions on modern shared memory system-on-chips

Motivation: Accelerating Computations at Edge

Edge computing are getting popular …

But they has **constraints**

- *e.g., energy or latency requirement*

Application of edge computing

- *Surveillance cameras*
- *Autonomous vehicles*
- *Mobile gaming*

Motivation: Accelerating Computations at Edge

Edge computing are getting popular …

But they has **constraints**

- *e.g., energy or latency requirement*

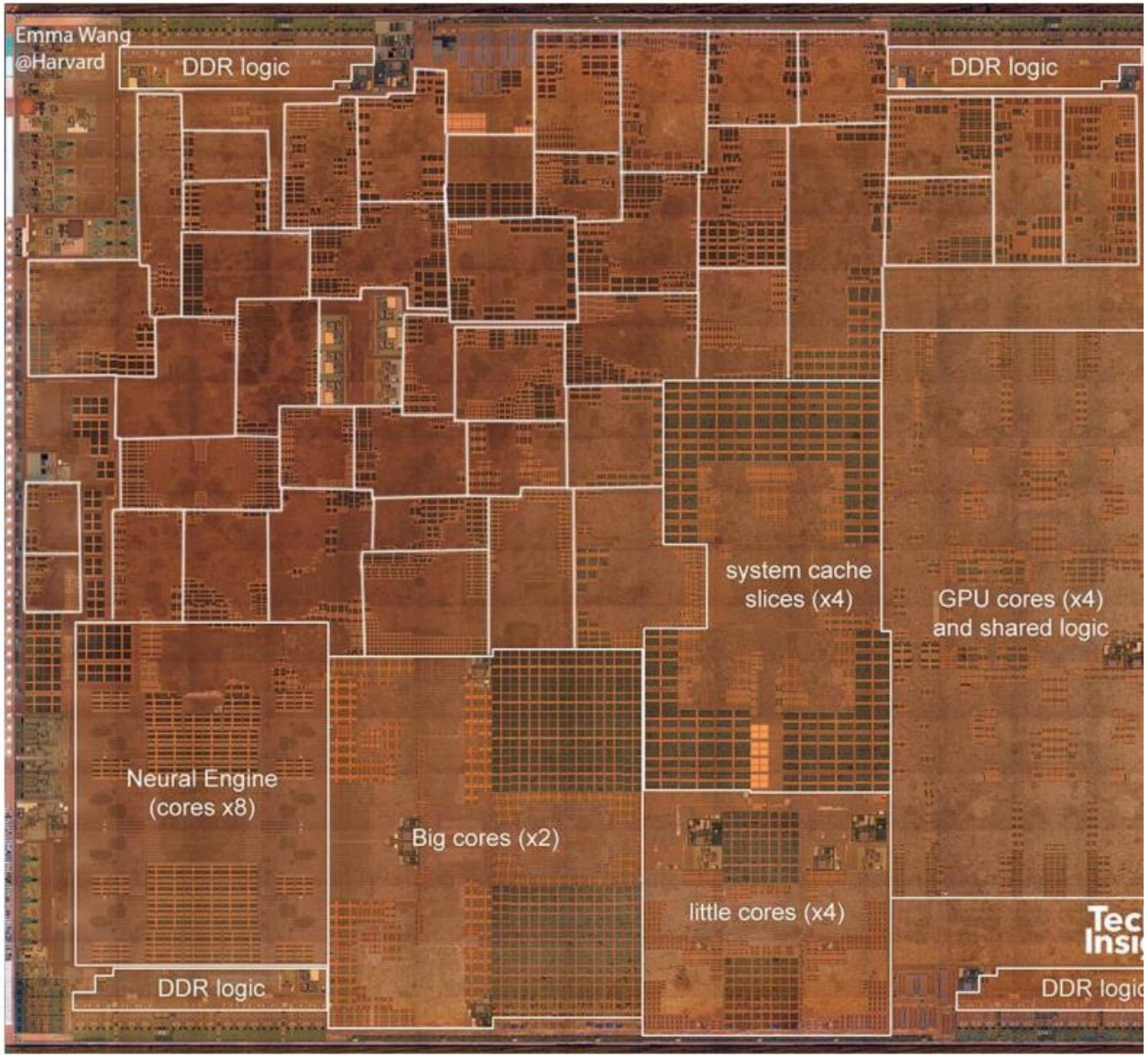
Application of edge computing

- *Surveillance cameras*
- *Autonomous vehicles*
- *Mobile gaming*

Modern edge devices are becoming increasingly heterogeneous

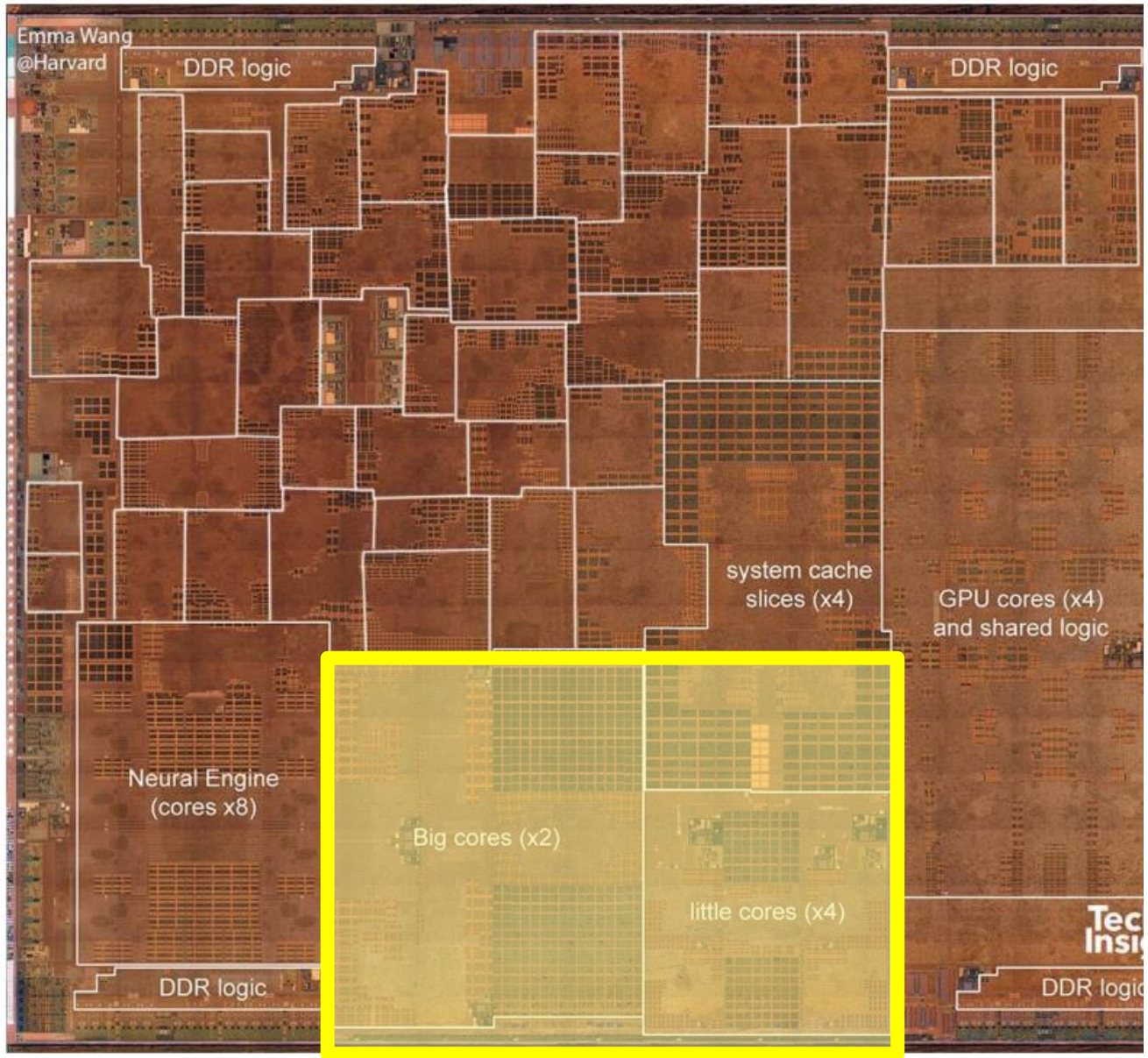
- w/ specialized *Processing Units* (PUs)

We need to efficiently utilize these available system resources



What do we mean by resources?

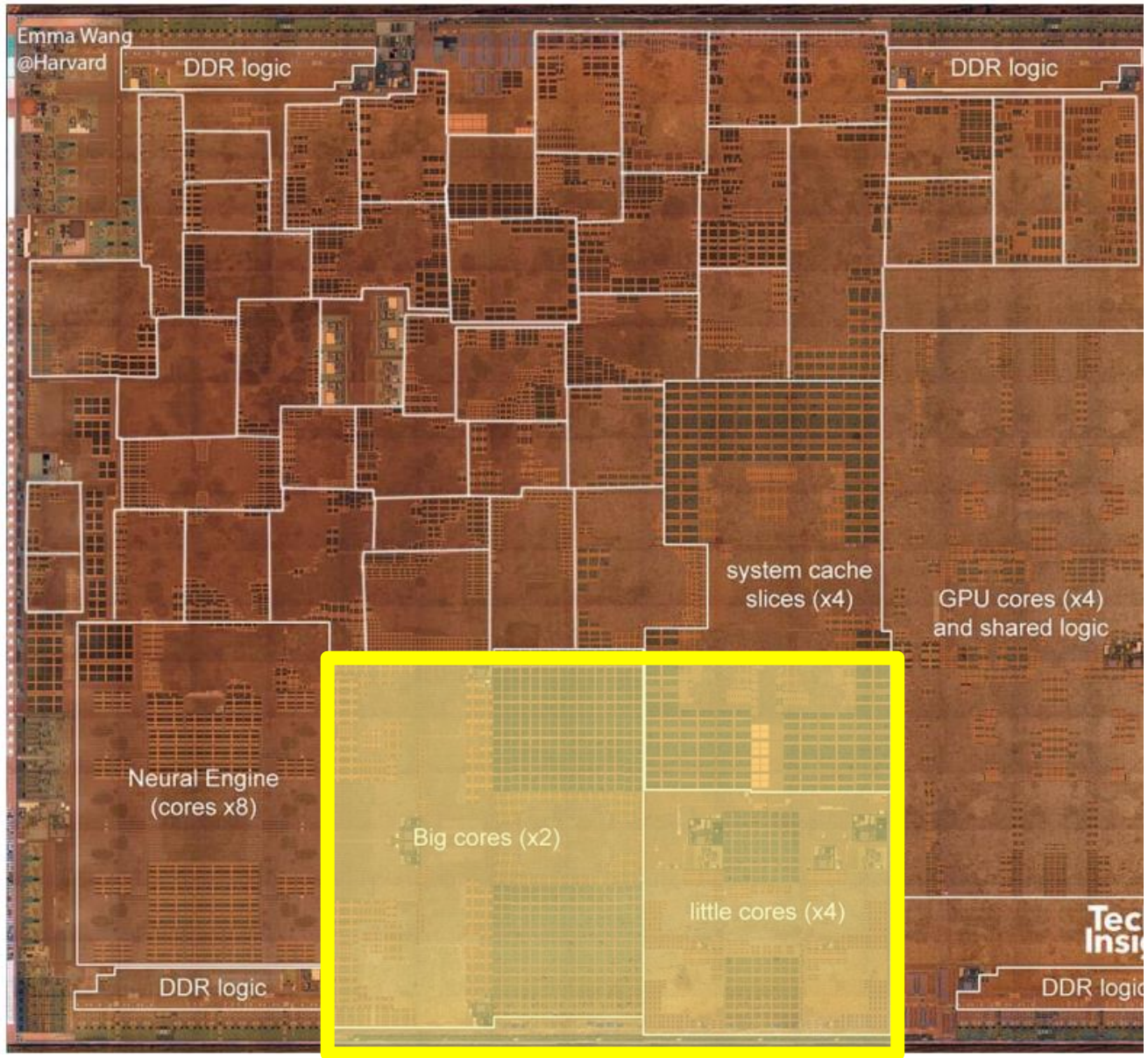
From David Brooks lab at Harvard:
https://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis



From David Brooks lab at Harvard:
https://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis

What do we mean by resources?

- E.g., less than 20% of the die area of an iPhone contains the CPU
- The rest contains specialize *Programmable Accelerating PUs* (PAPU)
  - e.g., integrated GPUs, FPGAs
  - Interconnected to a shared memory hierarchy
    - *Shared Memory Heterogeneous System* (**SMHS**) enables efficient communication between PUs



From David Brooks lab at Harvard:
https://vlsiarch.eecs.harvard.edu/resear

What do we mean by resources?

- E.g., less than 20% of the die area of an iPhone contains the CPU
- The rest contains specialize *Programmable Accelerating PUs* (PAPU)
  - e.g., integrated GPUs, FPGAs
  - Interconnected to a shared memory hierarchy
    - *Shared Memory Heterogeneous System* (**SMHS**) enables efficient communication between

How can workloads efficiently utilize each PU?

Processing Units (PU) Characteristics

CPU

Features: High-performance cores, reorder buffer, load store queue, …

+ **Latency optimized**

- **Limited throughput**

Good for **irregular** programs

Programmable Accelerating PUs (PAPU)

GPU

Features: SIMT (*Single Instruction, Multiple Threads*) execution, coalesced memory access

+ **Throughput optimized**

- **Warp Divergence**

FPGA

Features: Specialized tasks, Pipeline parallelism

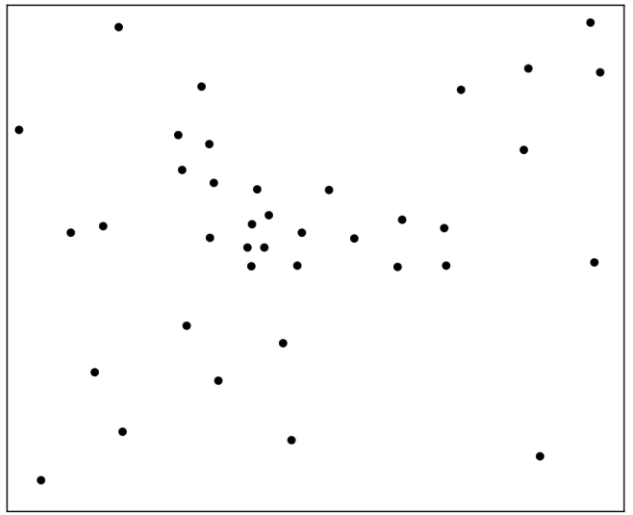
+ **Close to ASIC performance**

- **Orders-of-magnitude harder to program**

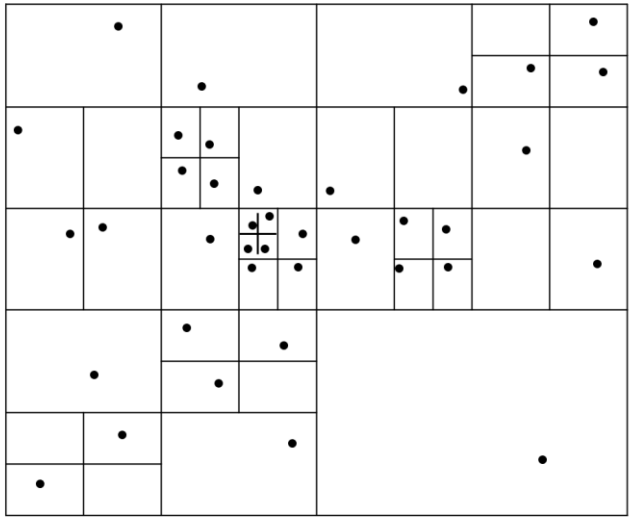
Good for accelerating **compute-intense** programs

Trees on the edge

- Edge applications need to process a large amount of data
- They can utilize tree structures and traversals to perform edge tasks
  - E.g., *octree, k-dimensional tree*



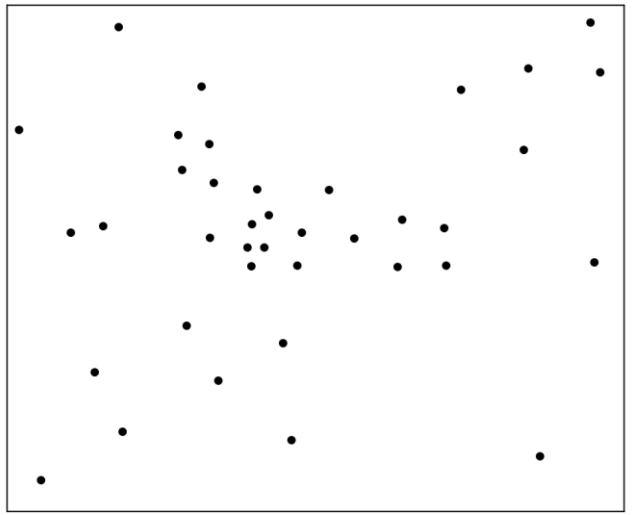
Input data



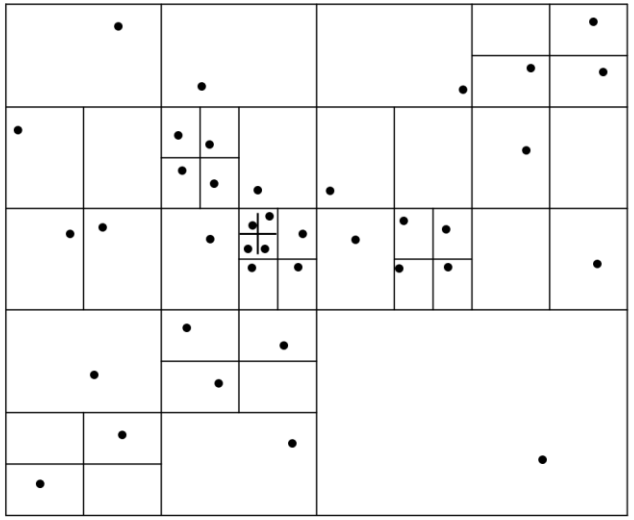
Spatial Partition

Trees on the edge

- Edge applications need to process a large amount of data
- They can utilize tree structures and traversals to perform edge tasks
  - E.g., *octree, k-dimensional tree*
- The dataset are organized into a hierarchical tree structure, allowing data to be **efficiently** searched from $O(n)$ to $O(\log n)$

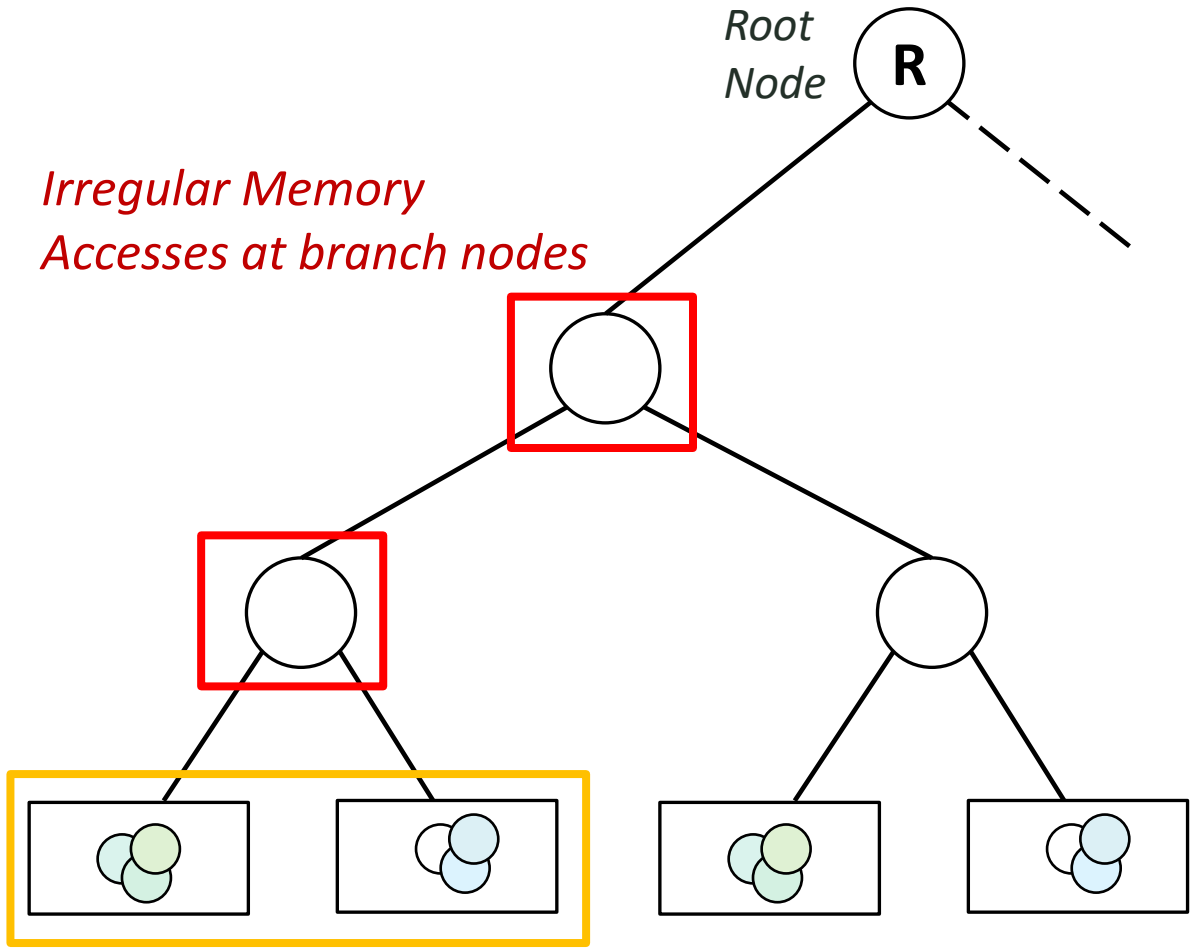


Input data



Spatial Partition

Traverse-Compute Workloads



- Repeatedly traversing a sparse tree structure
- Each traversal consists of
  - Indirect memory loads at branch nodes (Red box)
  - Dense data to be processed at leaf nodes visited (**Orange box**)
    - Computing pairwise interactions (e.g., Euclidean distance)
    - Reductions (e.g., sum, min)
- Example workloads:
  - Barnes-hut Algorithm (octree)
  - Nearest Neighbor (kd tree)
  - Ray Tracing (BVH)

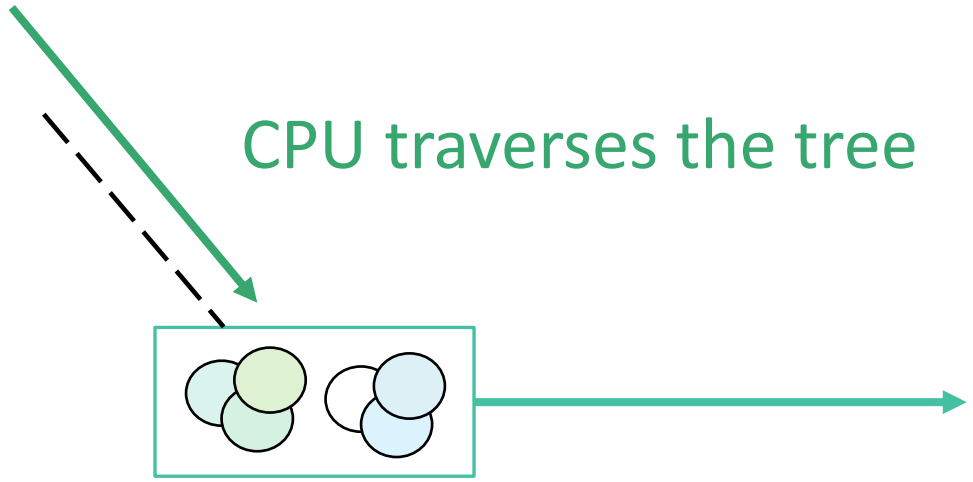
Decomposing Traverse Compute Workloads

Tree applications can benefit from fine-grained acceleration

CPUs are good at handling dynamic control flows and tolerating indirect memory loads

PAPUs are good at accelerating regular, compute-intense operations

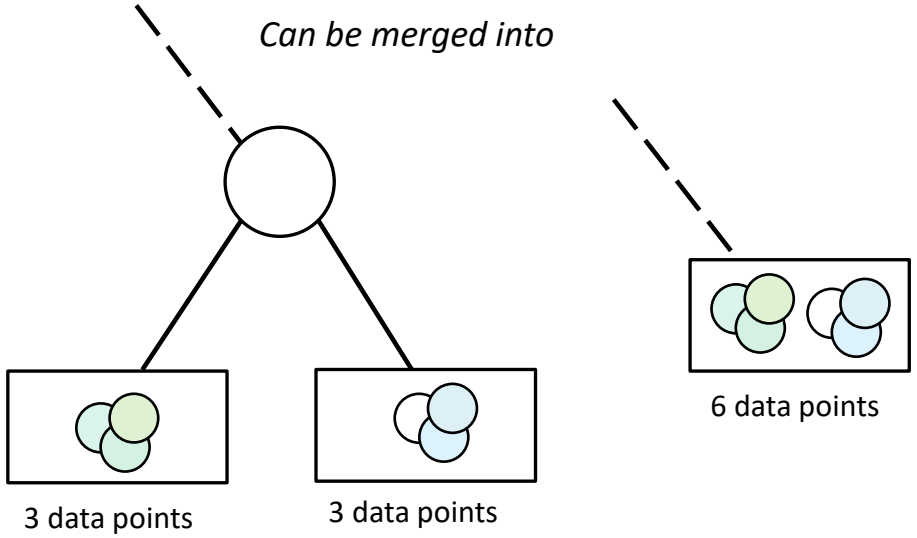
A natural heterogeneous approach is to



Computations at leaf nodes are offloaded to the accelerator

Accelerating Traverse-compute workloads on SMHSs

- The tree can be ***parameterized*** by how many data points exist on the leaf nodes.



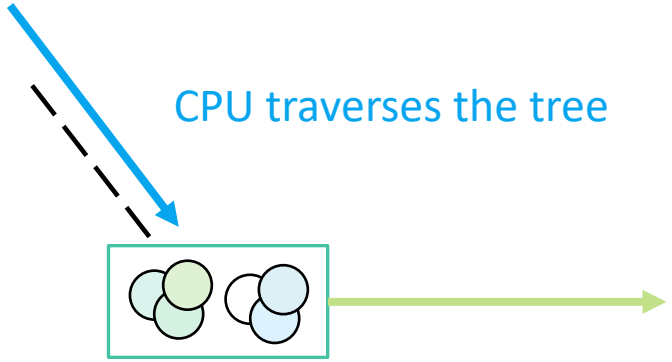
Larger node sizes tradeoffs

+ Larger chunk of contiguous data

+ Less irregular accesses in tree traversals

- Potentially unneeded computation

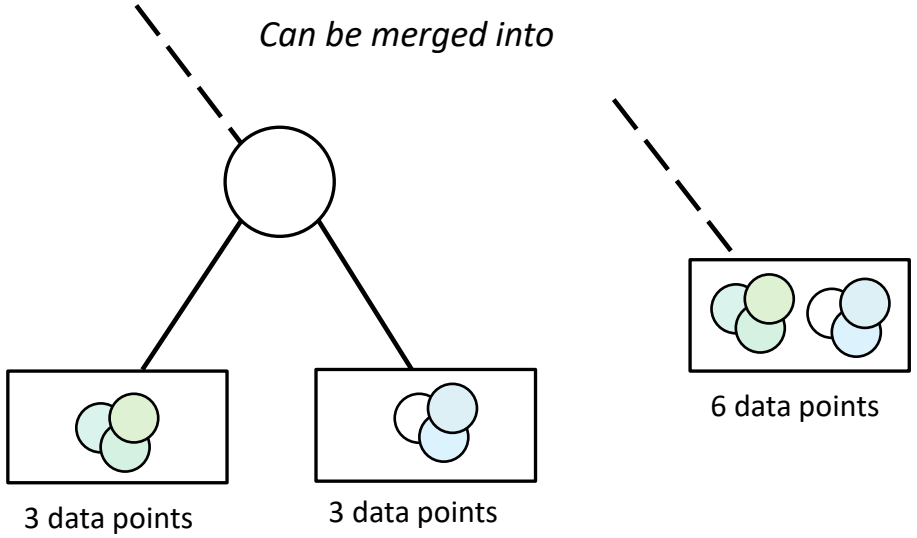
Heterogeneous Approach



Computation is offloaded to accelerator

Accelerating Traverse-compute workloads on SMHSs

- The tree can be ***parameterized*** by how many data points exist on the leaf nodes.



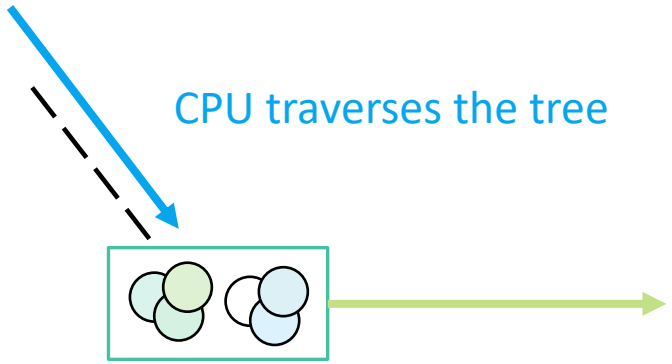
Larger node sizes tradeoffs

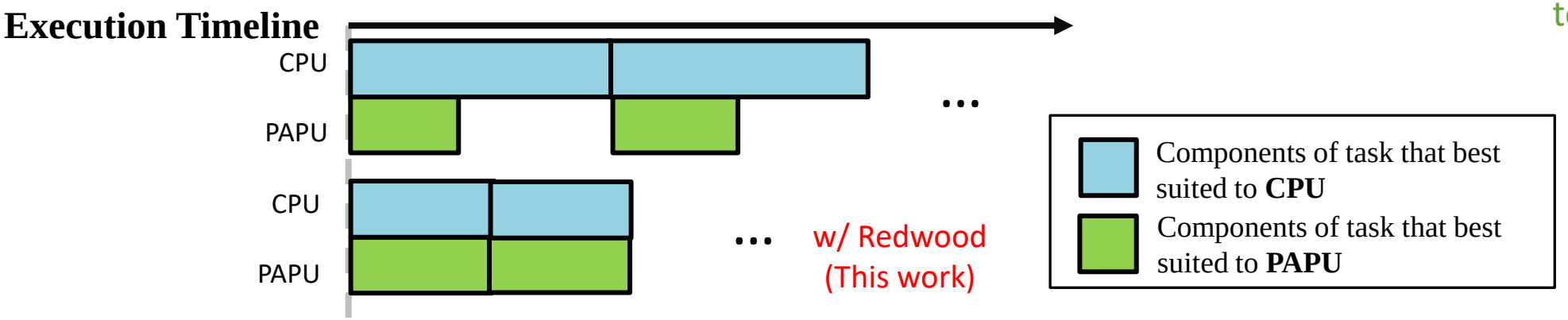
+ Larger chunk of contiguous data

+ Less irregular accesses in tree traversals

- Potentially unneeded computation

Heterogeneous Approach





Flexible & Specialize Heterogeneous Decomposition

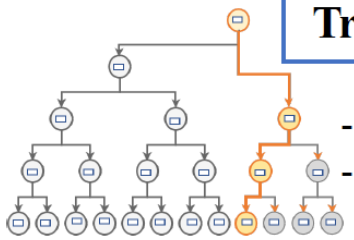
This work: **Redwood** Overview

Redwood APIs

```
void Traverse (Node node, Point q)
{
 if ( node.IsLeaf() )
    result += Compute( node.data, q );
 else
    for( child in node.children )
       Traverse(child, q);
}
```

Traverse-Compute Workloads

Flexible Decomposition



Traverse-heavy config

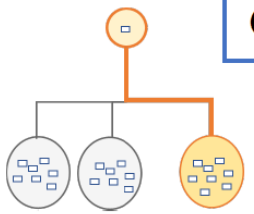
- 4 nodes visited
- 4 computations in nodes

CPU

PAPU

Intel i5-10210U CPU @ 1.60GHz
Intel UHD Graphics 620

Nearest Neighbor 3.36x faster
at optimal leaf node size 256



Compute-heavy config

- 2 nodes visited
- 9 computations in nodes

PAPU

CPU

RISC-V TimingSimple CPU
w/ Duet eFPGA

Nearest Neighbor 13.53x faster
at optimal leaf node size 512

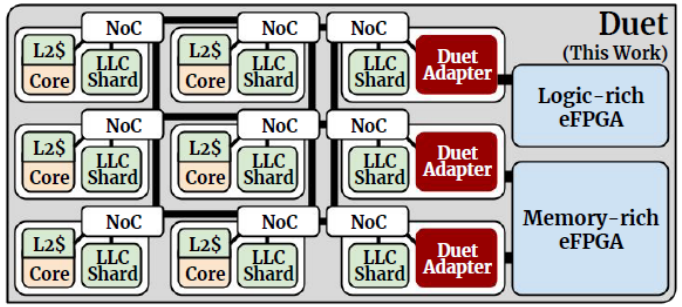
Users implement tree applications using our APIs

Target systems w/ different CPU/PAPU throughputs

KNN based Facial recognition

Intel SoCs

Nvidia SoCs



Duet

Redwood: APIs and Data Structures

CPU Sequential Code (NN)

```
tree = KDTree()
min_dist = 99999.9999
def traverse(node, q):
    if is_leaf(node):
        # Reduce Leaf Node
        for i in range(node.leaf_size):
            kernel_func(q, node.data[i])
    else:
        dist = compute_dist(q, node.data[0])
        min_dist = min(min_dist, dist)
        traverse(node.leaf_child)
        if check_other_side(dist):
            traverse(node.right_child)
```

Implemented using:

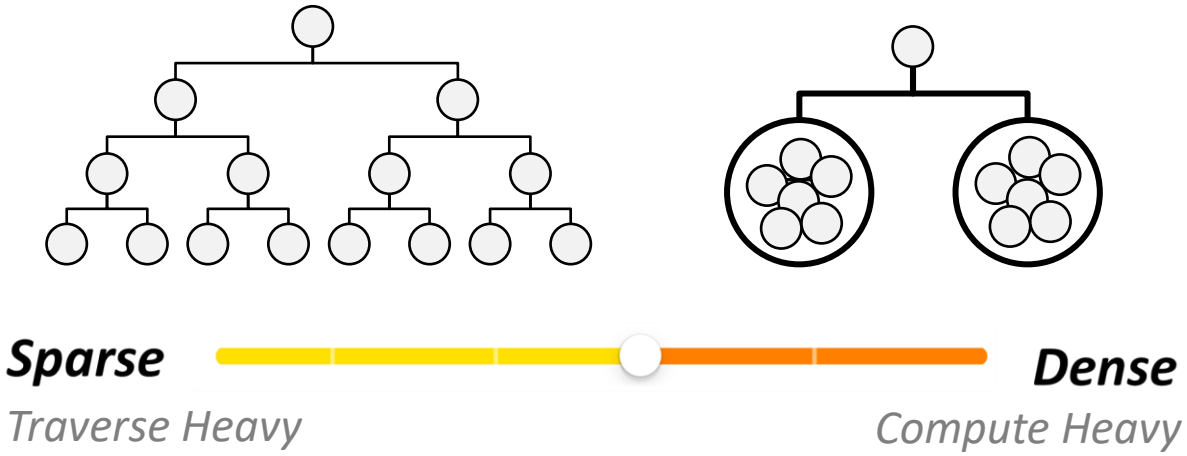
w/ Redwood API

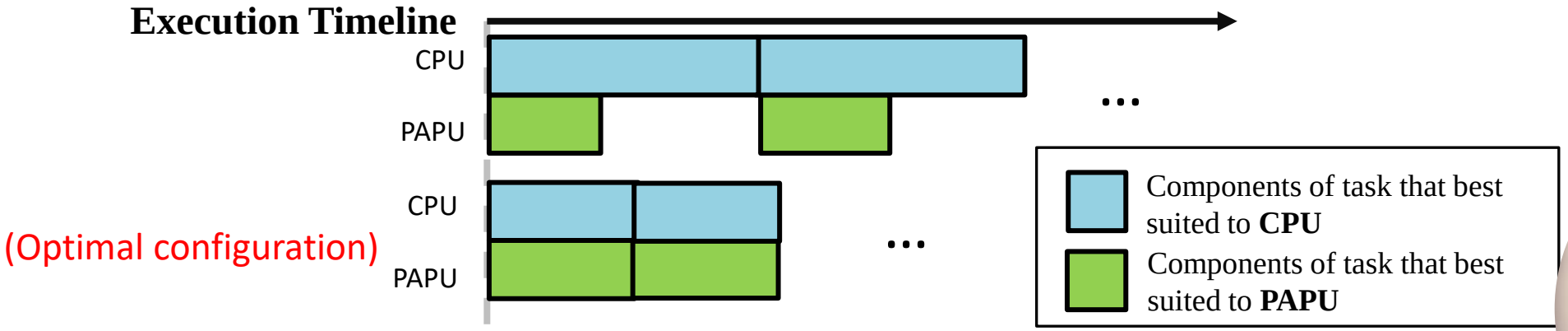
```
tree = KDTree(leaf_size=32)

redwood_set_query(q)

def traverse(node, q):
    if is_leaf(node):
        redwood_compute_leaf(node.data())
    else:
        dist = compute_dist(q, node.data[0])
        min_dist = min(min_dist, dist)
        traverse(node.leaf_child)
        if check_other_side(dist):
            traverse(node.right_child)
```

Redwood Heterogenous Optimizations

Flexible Leaf Size Configuration

- Adapt to various heterogeneous systems with different relative throughput between the CPU and the PAPU

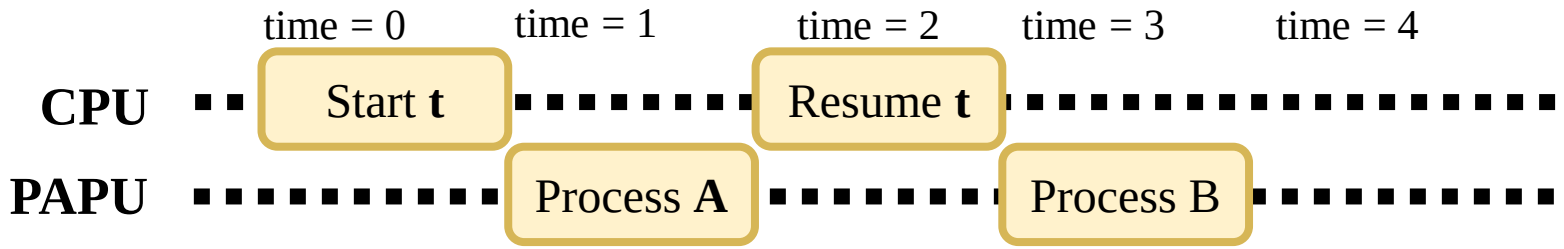


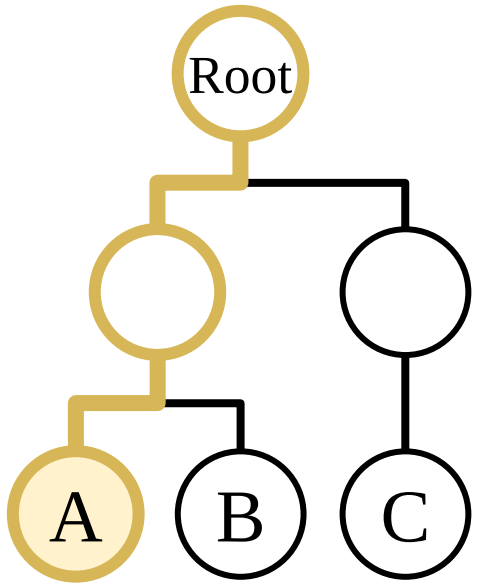


Redwood Heterogenous Optimizations

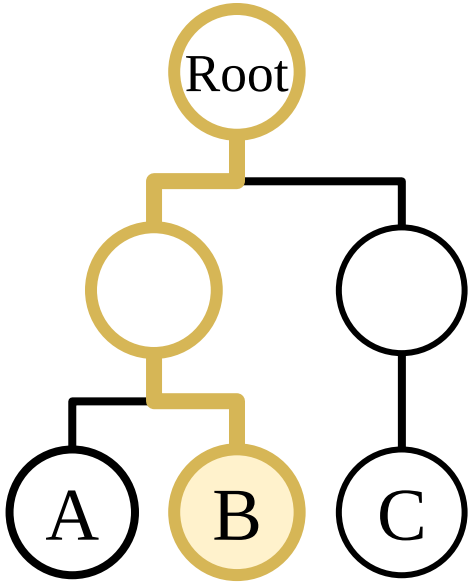
Traverser Runtime

- Allow a single CPU thread to execute many traversals concurrently to avoid stalling when a traversal depends on a PAPU accelerated value





*Start **t***

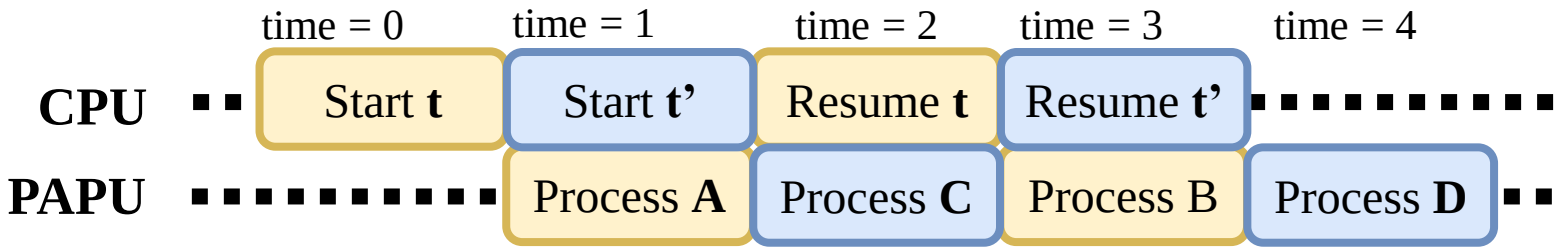


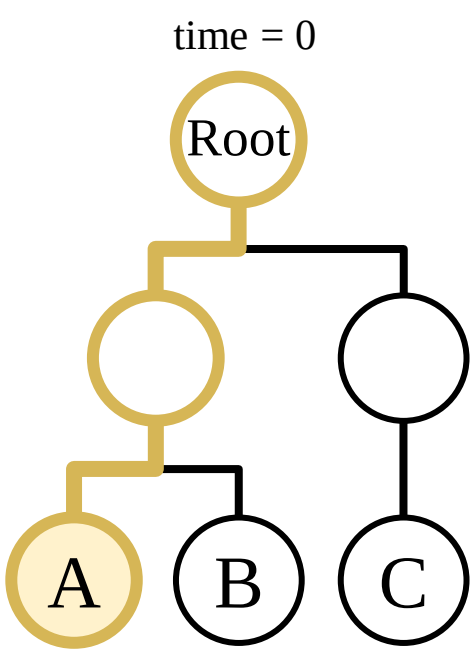
continue when result ready

Redwood Heterogenous Optimizations

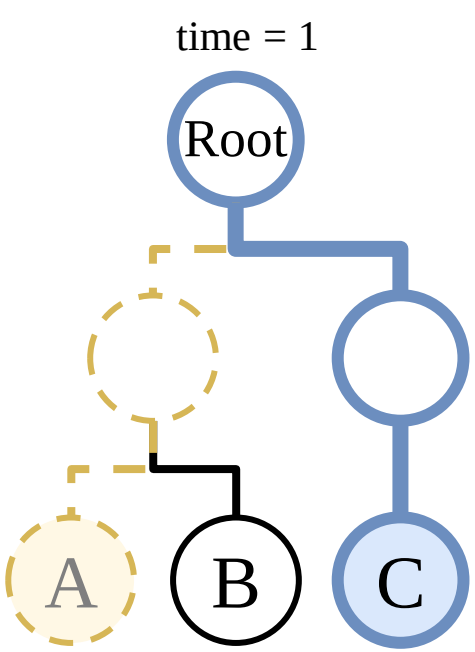
Traverser Runtime

- Allow a single CPU thread to execute many traversals concurrently to avoid stalling when a traversal depends on a PAPU accelerated value
  - *Lightweight Coroutine*
    - ***Suspend***
    - ***Resume***

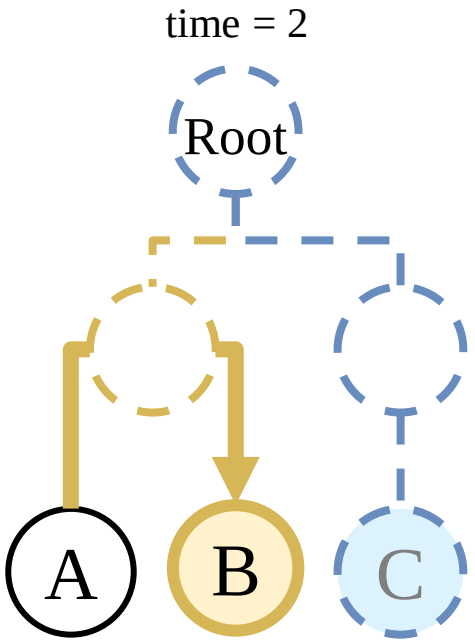




*Suspend **t***



*Start **t'***



*Resume **t***

Grove: Benchmark Suite for SMHS

Grove contains **9** traverse-compute workloads

Can be found in many applications

- *Astrophysics*
- *Facial recognition*
- *Anomaly detection*
- *Outlier detection*
- *Particle simulation*

Tree Structures

- Octree/quadtree
- k-d tree

Three Algorithms

- Barnes Hut
- Nearest Neighbor
- k Nearest Neighbor

Computation Patterns

- Aggregation *(sum)*
- Reduction *(e.g., min)*
- Sorting

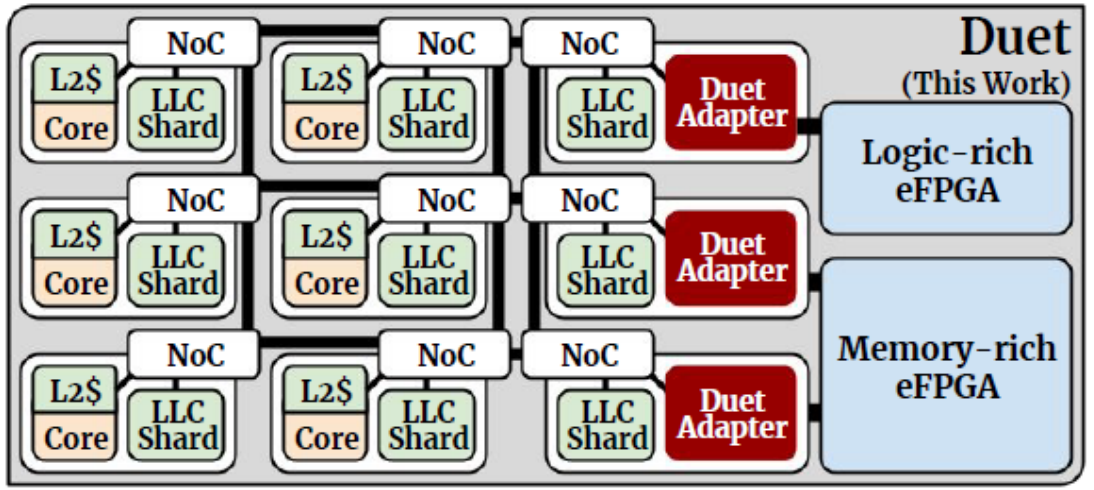
Various Distance Metrics

- Euclidean
- Manhattan
- Chebyshev

Evaluating an Open-Source SMHS: **Duet**

Duet

- A tightly-integrated, cache-coherent CPU-FPGA architecture
- Enables fine-grained transparent data sharing between the processors and the eFPGA-emulated accelerators
- Simulated in using ***Gem5-Duet*** extension



Duet[1]

Platform	Backend	CPU	CPU Frequency	Accelerator	Accelerator frequency
Duet (simulated in **gem5**)	HLS	RISC-V TimingSimple CPU	1.5 GHz	Duet eFPGA	333MHz

Configuration

[1] Li, Ang, August Ning, and David Wentzlaff. HPCA'23

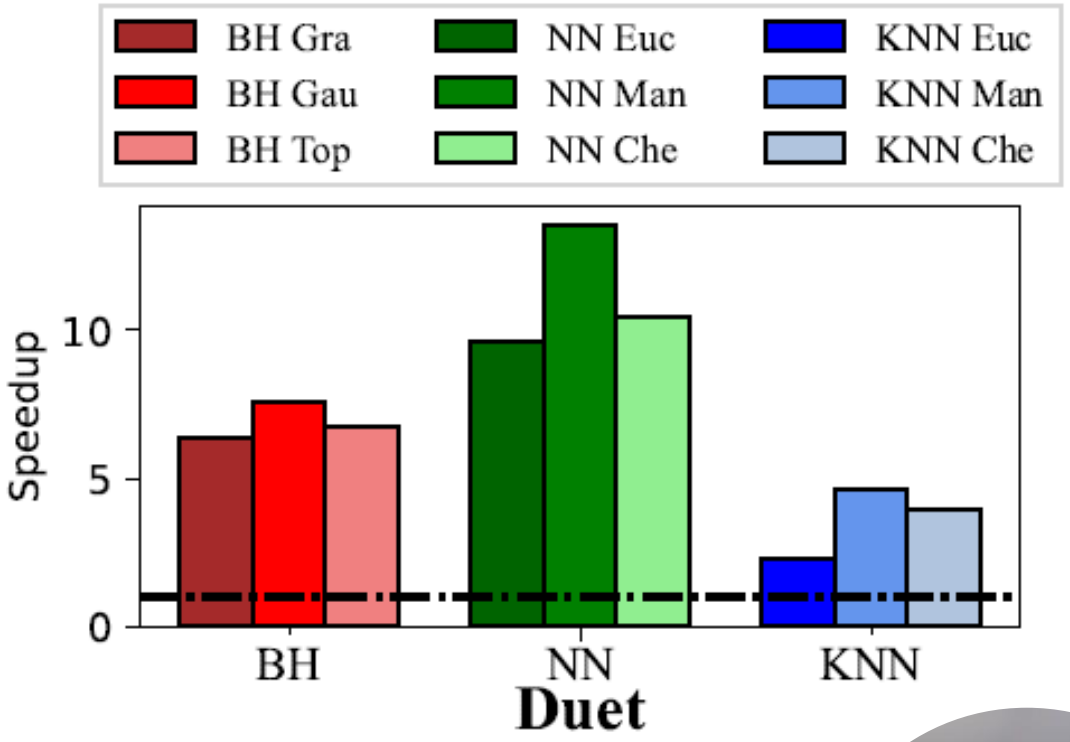
Grove Results Overview

	Leaf Size CPU	Leaf Size w/ Duet	Ratio	Avg Speedup
BH	3.33	512	153.6	6.9x
NN	26.67	426.67	16	11.2x
KNN	26.67	128	4.8	3.64x
Average	19	355	18.8	6.43x

We swept through the leaf node sizes to find the optimal configuration that yield the best performance,

- *Average 18× larger leaf node size than the CPU*

Speedups
highest 13.53x
geomean 6.43x



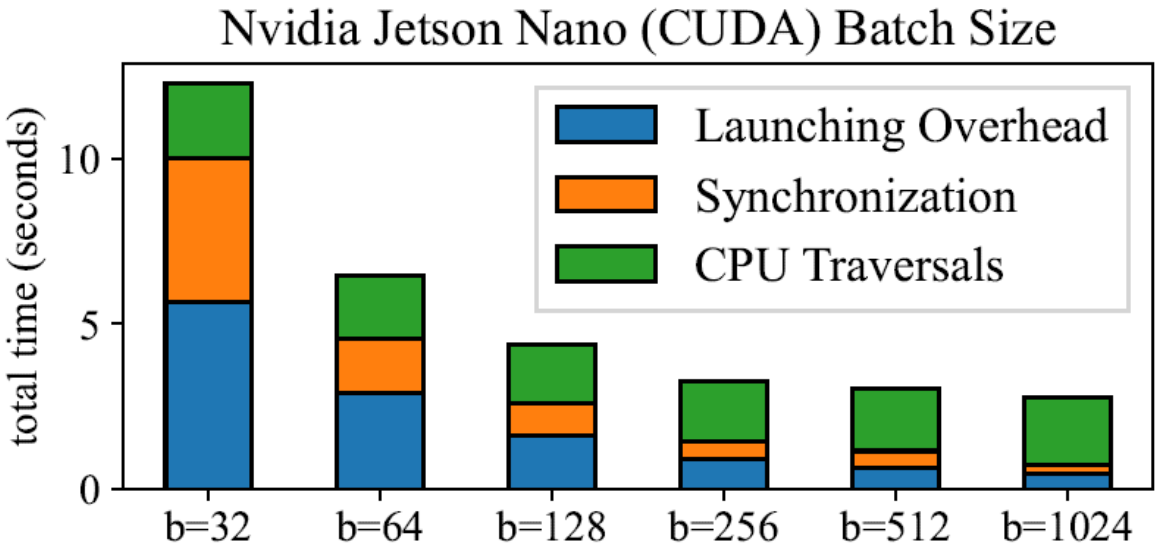
Speedups of the best heterogeneous configuration vs. the best homogeneous configuration of Grove.

We compared Duet to GPU-based SMHSs

Kernel Submission Cost

- Traverse-compute applications **frequently** invoke small kernels
- Useful works are shown in Green
- Orange/Blue are overheads
- Low-cost kernel submission is important for accelerating applications on edge devices

- **Duet has minimal offload overhead**



Batching multiple GPU kernels into a single/larger kernel helps amortizing kernel launching overhead on GPU-based systems

Conclusion

- ✓We present how open-source hardware design can be used to accelerate a pragmatic class of applications
- ✓We show that the Duet system can accelerate a suite of traverse-compute applications by up to 13.5× with a geomean of 6.43×
- ✓We highlight the use of Grove, an open-source benchmark suite of traverse-compute workloads that utilize fine grained synchronization across PUs, and thus can provide a way for architecture researchers to evaluate their heterogeneous designs

UC Santa Cruz Redwood Grove

Team

Yanwen Xu yxu83@ucsc.edu

Ang Li angl@princton.edu

Tyler Sorensen tyler.sorensen@ucsc.edu

Open-Source Repo

Redwood & Grove at

https://github.com/xuyanwen2012/redwood-rt