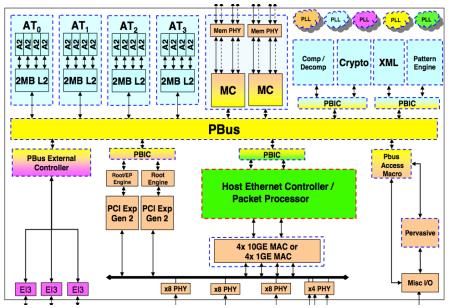
Rinnegan: Efficient Resource Use of Heterogeneous Processors

Sankaralingam Panneerselvam
Michael Swift

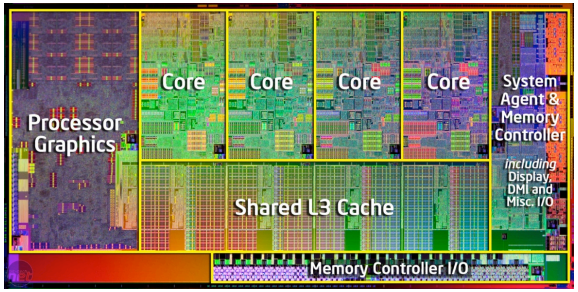
University of Wisconsin - Madison

Heterogeneity in Architectures

Wire speed processor



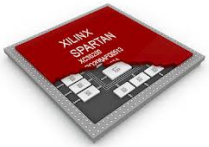
Intel Sandy Bridge



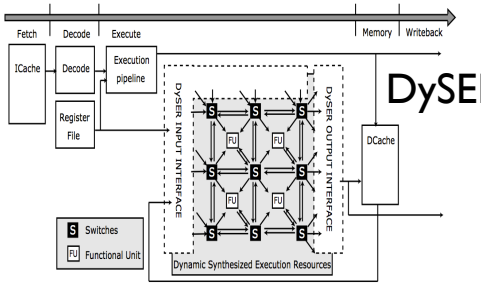
APU

Crypto accelerator

GPU

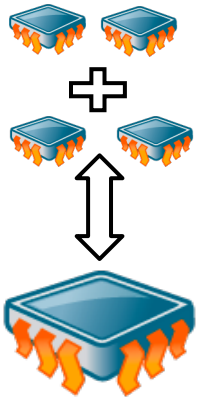


FPGA



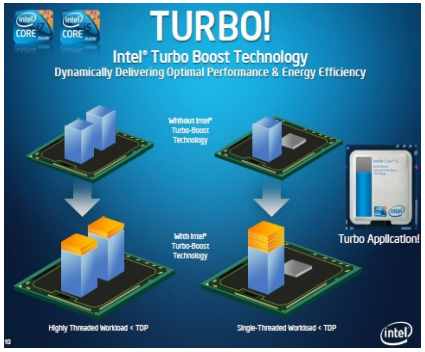
DySER

H.264
MPEG-4/AVC



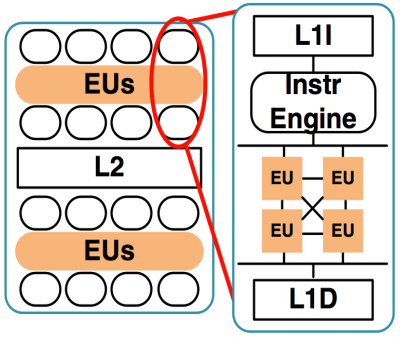
Core Fusion

ARM big.LITTLE



Turbo Boost

WiDGET



Resource Management

- Accelerator Management
  - Accelerators will be easy to use [CUDA, OpenCL, C++ AMP]
  - Treat like any other shared resource [PTask]
- Power Limits
  - Dark Silicon: All units cannot be used at the high performance
  - Hardware does not possess sufficient knowledge for power distribution

Applications meet Heterogeneity

- Task can run on many processing units
  - E.g. Parallel tasks - GPU or multi-core CPUs

    Encryption - crypto accelerator or AES
- Accelerators do not always yield better efficiency
  - **Contention** for accelerator
  - **Data transfer** overhead: Latency to copy 64 bytes of data onto GPU takes 6 μ s
- Power limits could prevent applications from using devices at full performance

Rinnegan

- Goals
  - Accelerators and Power should be treated as primary resource in the system
  - Applications should leverage heterogeneity with less efforts
- Target architecture include
  - Discrete accelerators, on-chip accelerators, single-ISA and multi-ISA heterogeneous processors

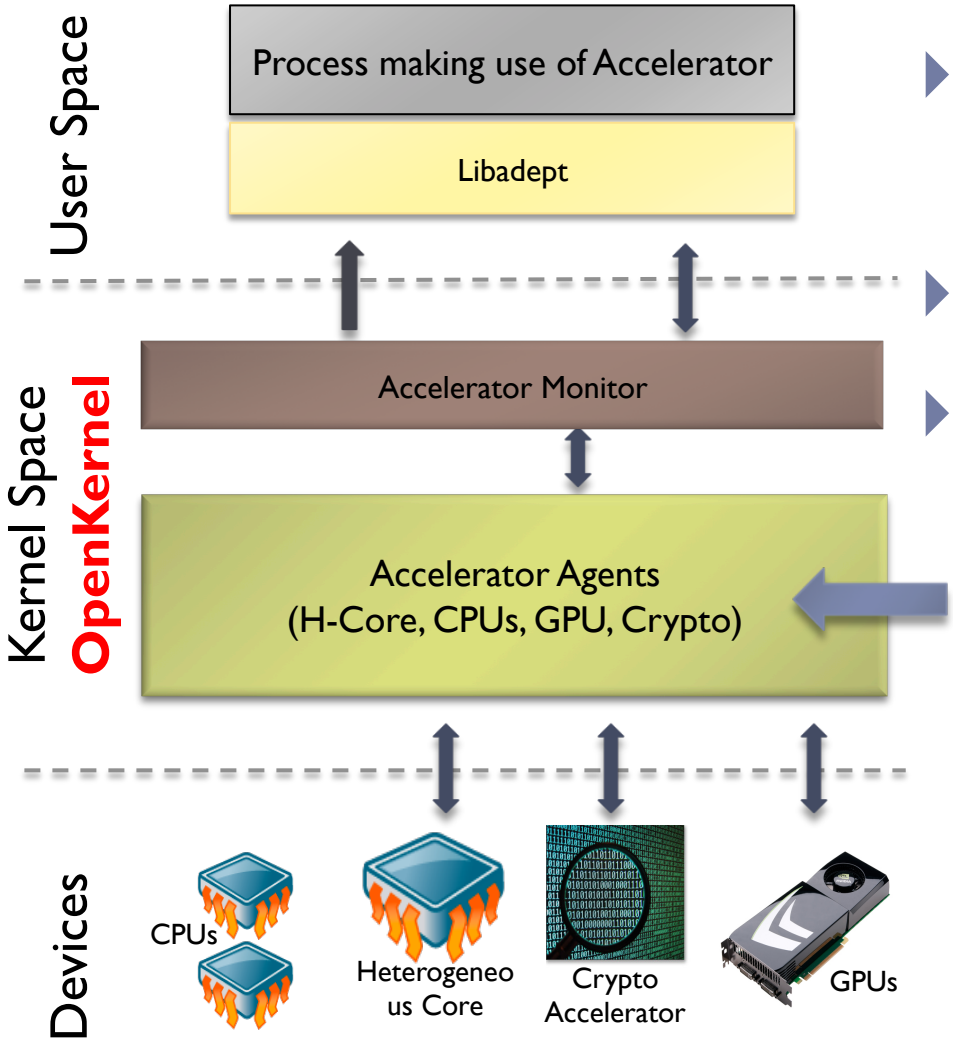
Outline

- Motivation
- Rinnegan
- Evaluation
- Conclusion

Rinnegan

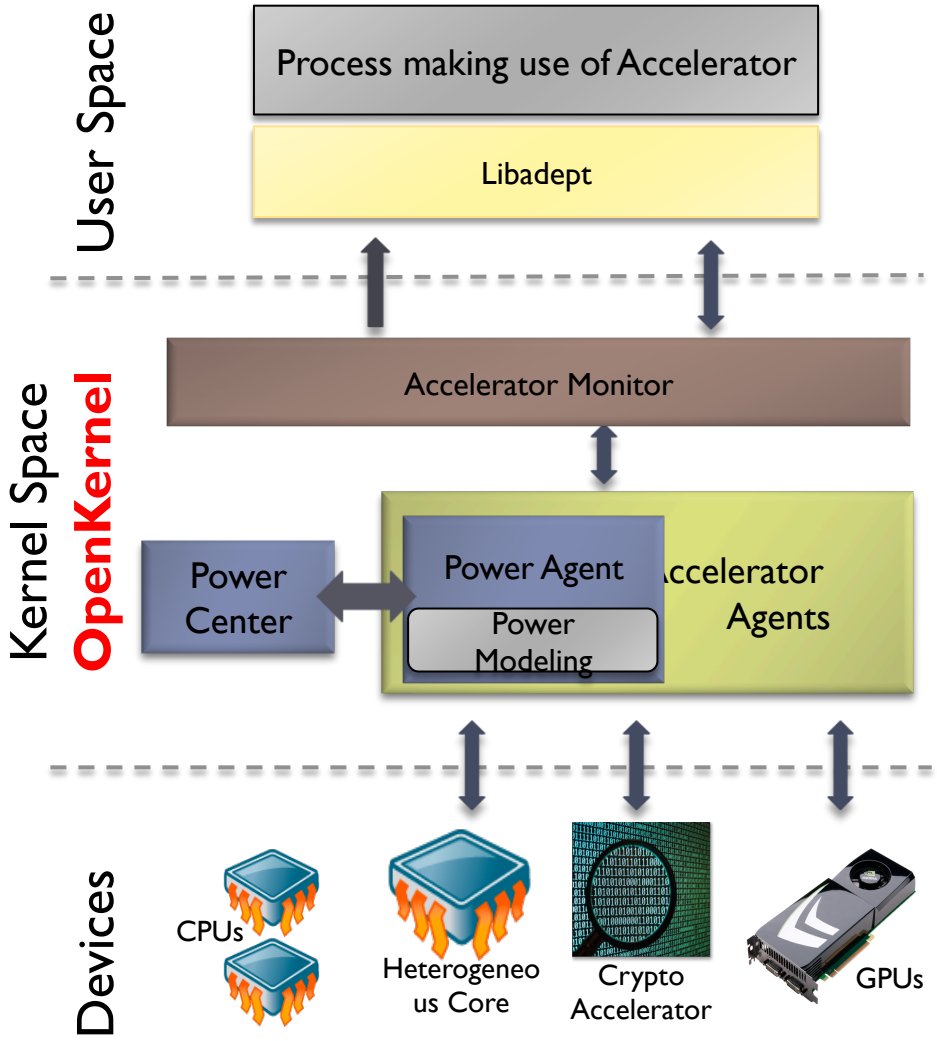
- Layered system with two major components
  - OpenKernel
    - Enforces scheduling policies
    - Exports usage information to application side
    - Enables power awareness in the system
  - Libadept (runtime in user mode)
    - Helps to deal with heterogeneity
    - Handles task placement

[OpenKernel] Accelerator Agent



- Every class of accelerator has its own agent
- Similar to device driver
- Roles of an agent
  1. Expose utilization information
  2. Implement scheduling decisions
  3. Enforces power limits

[OpenKernel] Power Abstraction

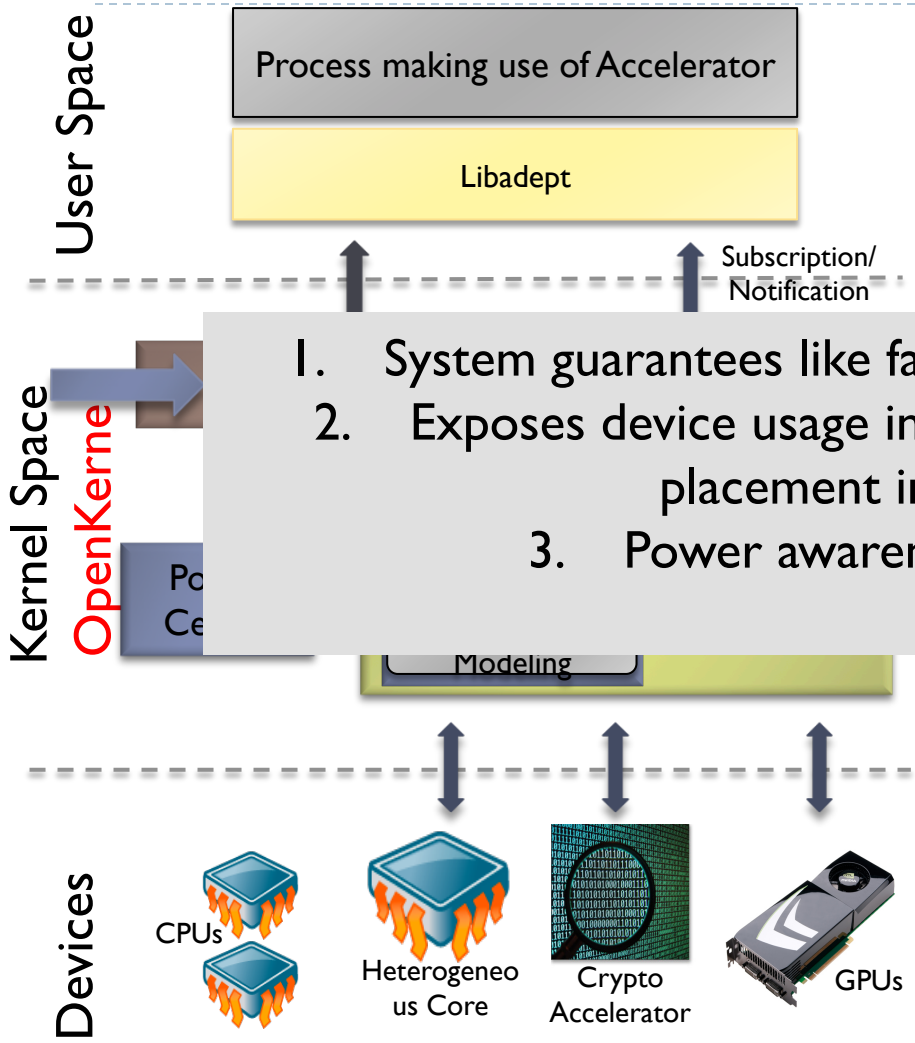


- Power credits
  - Maintained by a central entity called *Power center*
  - Ability to use credits for computation
  - Power limit is expressed in terms of power credits

[OpenKernel] Power Accounting

- Measure the power consumed by each processing unit for every task
- Power Agent
  - Part of the Accelerator Agent
  - Power model (From device or software)
    - Power credits needed for a task to run
    - Mapping between power credits and power-state
  - Gathers sufficient power credits on behalf of the task

[OpenKernel] Accelerator Monitor



1. System guarantees like fairness, isolation are provided
2. Exposes device usage information to help with task placement in the application
3. Power awareness in the system

- Publishes information exposed by agents to applications

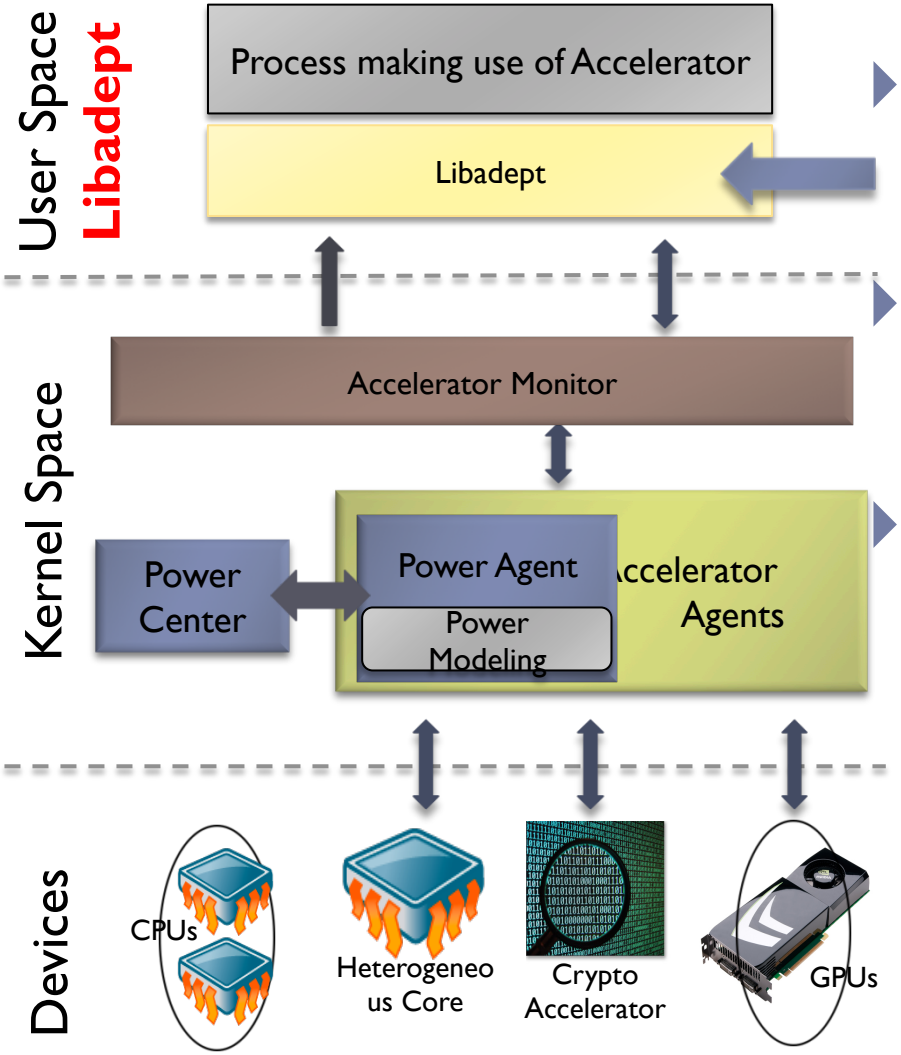
sm to
on
y

events

Rinnegan

- Layered system with two major components
  - OpenKernel
    - Enforces scheduling policies
    - Exports usage information to application side
    - Enables power awareness in the system
  - Libadept (runtime in user mode)
    - Helps to deal with heterogeneity
    - Handles task placement

Libadept Runtime



- Accelerator Stub *abstracts* different processing units
- Includes a *profiler* that predicts task performance
- Selects best processing unit to yield better efficiency
  - Assume multiple implementations are available

Outline

- Motivation
- Rinnegan
- Evaluation
- Conclusion

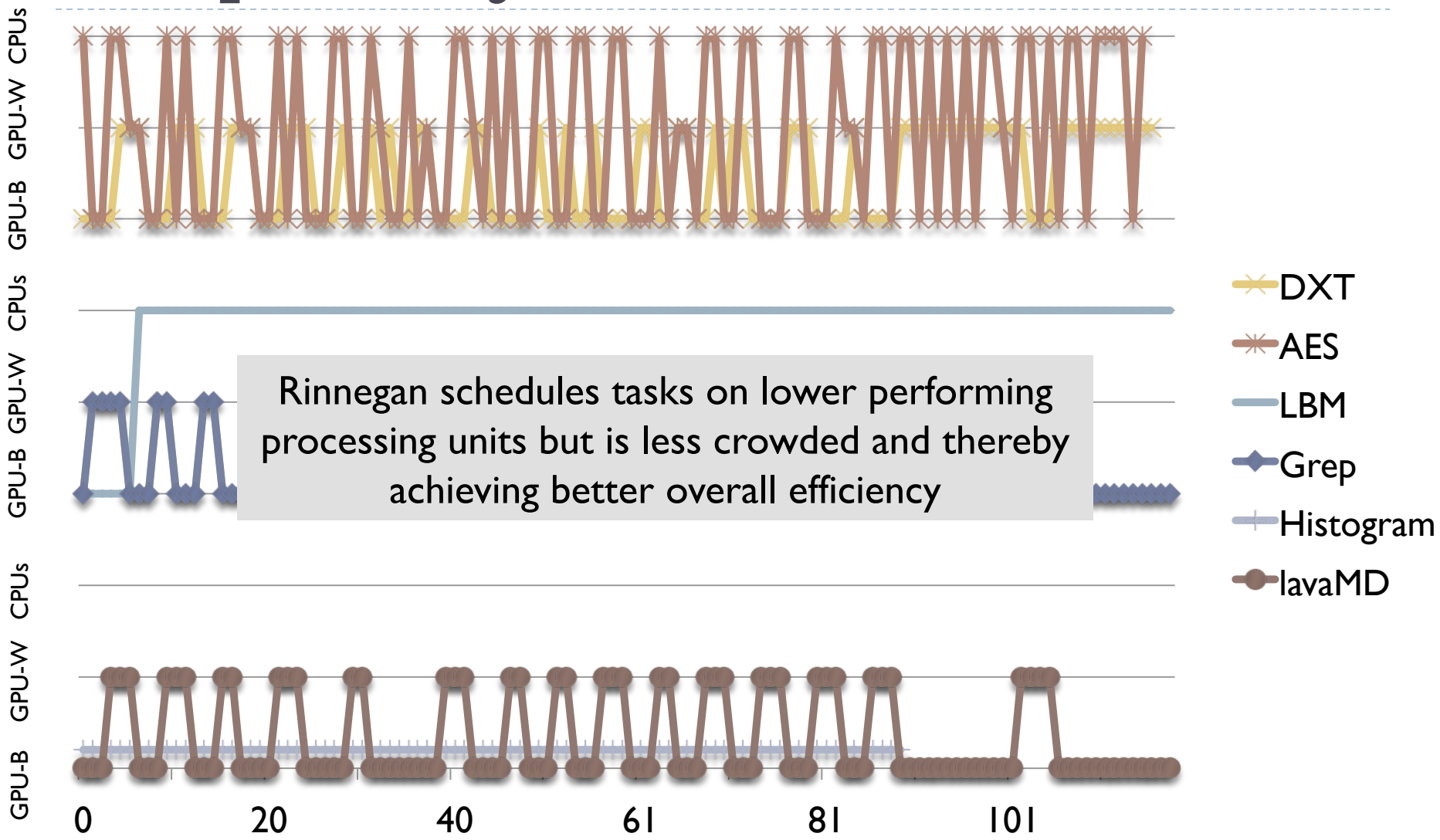
Configuration

Machine	(i) **GPU-B**: Powerful GPU (NVIDIA Geforce 670) (ii) **GPU-W**: Wimpy GPU (NVIDIA Geforce 650) (iii) 12 Cores in 2 intel Xeon 5650	
	Quad core Sandy-Bridge machine for power results	
Workloads	(1) Histogram	Searching for occurrences of dictionary words in a list of files
	(2) Grep	String search
	(3) LBM	Fluid dynamics simulation implemented in OpenCL
	(4) DXT	Image DXT Compression
	(5) AES	AES-128-ECB mode enryption
	(6) lavaMD	Particle simulation

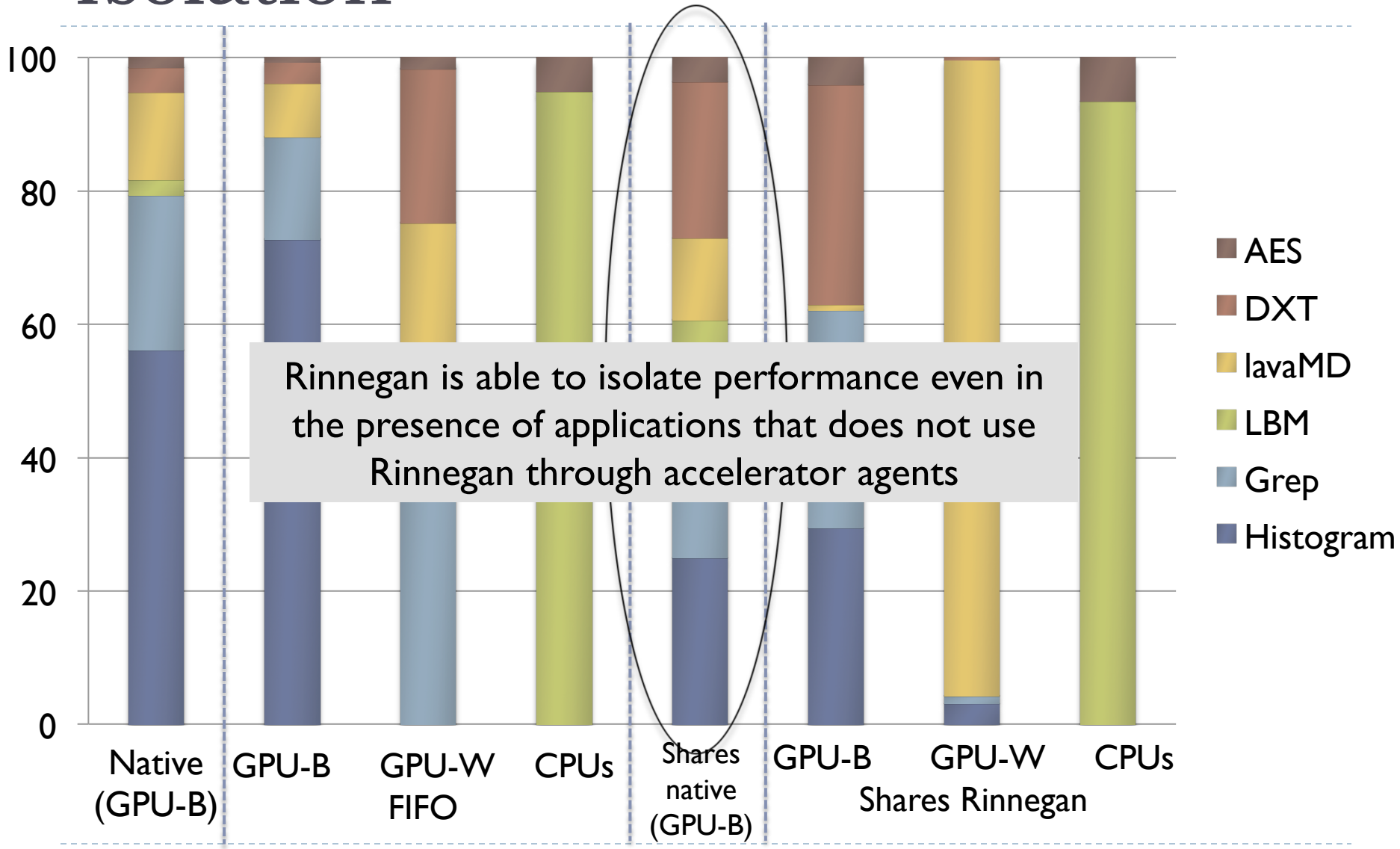
Evaluation

- Adaptability:
  - Can applications leverage heterogeneity through OpenKernel?
- Isolation:
  - Does Rinnegan provide performance isolation in the presence of greedy applications?
- Power distribution:
  - Can Rinnegan leverage heterogeneity in a power-constrained architecture?

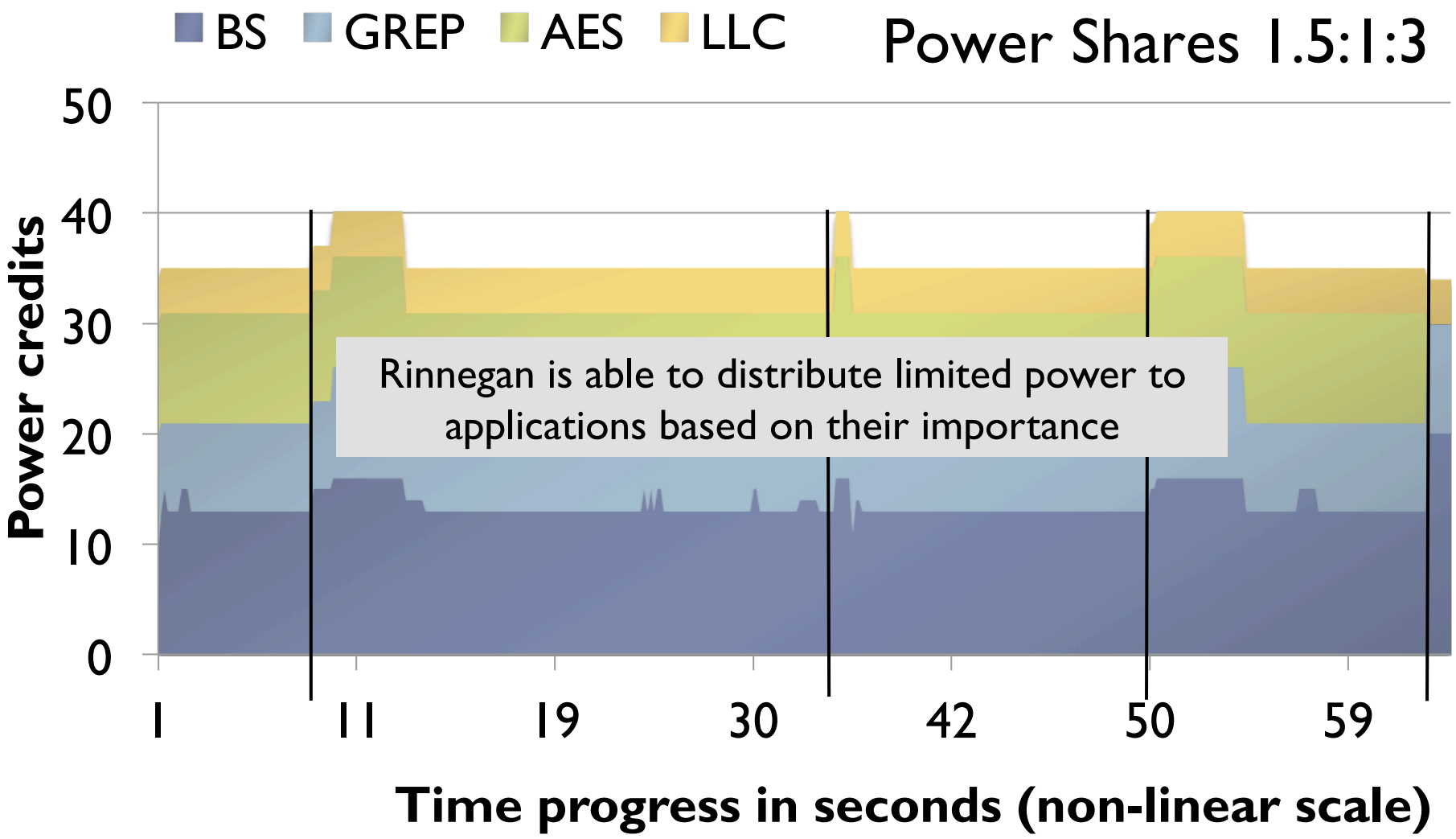
Adaptability



Isolation


100
80
60
40
20
0
Native (GPU-B)
GPU-B
GPU-W FIFO
CPUs
Shares native (GPU-B)
GPU-B
GPU-W
CPUs
Shares Rinnegan
AES
DXT
lavaMD
LBM
Grep
Histogram
Rinnegan is able to isolate performance even in the presence of applications that does not use Rinnegan through accelerator agents

Power Distribution



Conclusion

- Heterogeneous architectures will be common
- Rinnegan leverages heterogeneity through
  - OpenKernel:
    - System guarantees on resources like accelerators and power
    - Helps application by exposing usage information
  - Libadept:
    - Helps in choosing the right configuration

Thank You

For further questions: sankarp@cs.wisc.edu

Task Placement

- Moving placement to applications simplifies kernel which only need to handle scheduling
- Selecting best processing unit requires a performance model of application code
- Application have best knowledge of their scheduling goals
- Some accelerators can be accessed directly from user mode

Sample Program

```
/*** Libadept ***/
void Accelerate(Task *task, bool sync) {
        if(appInfo.optimizer) punit = appInfo.optimizer();
        else punit = defaultOptimizer();
        Schedule(task, punit);
}

/*** Application ***/
void TaskGpu(void *args) {/* Task logic on GPU */}
void TaskCPU(void *args) {/* Task logic on CPU */}

int main() {
        ...
        Task *task = InitializeTask();
        AssociateAcceleratorTask(task, SIMD, TaskGPU, TaskCPU, args);
        Accelerate(task, ASYNC);
        WaitForTask(task);
        ...
}
```