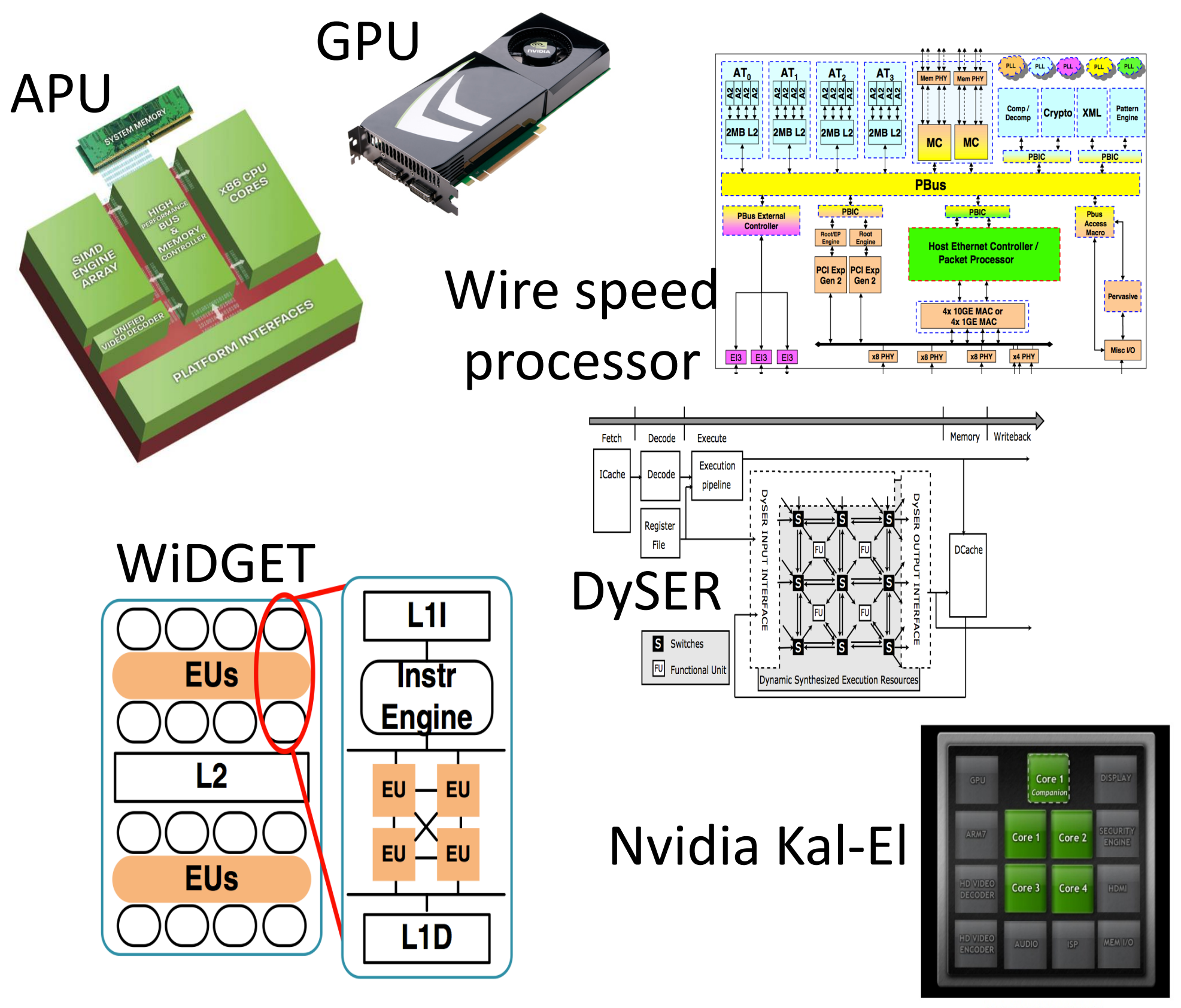
THE UNIVERSITY of WISCONSIN MADISON

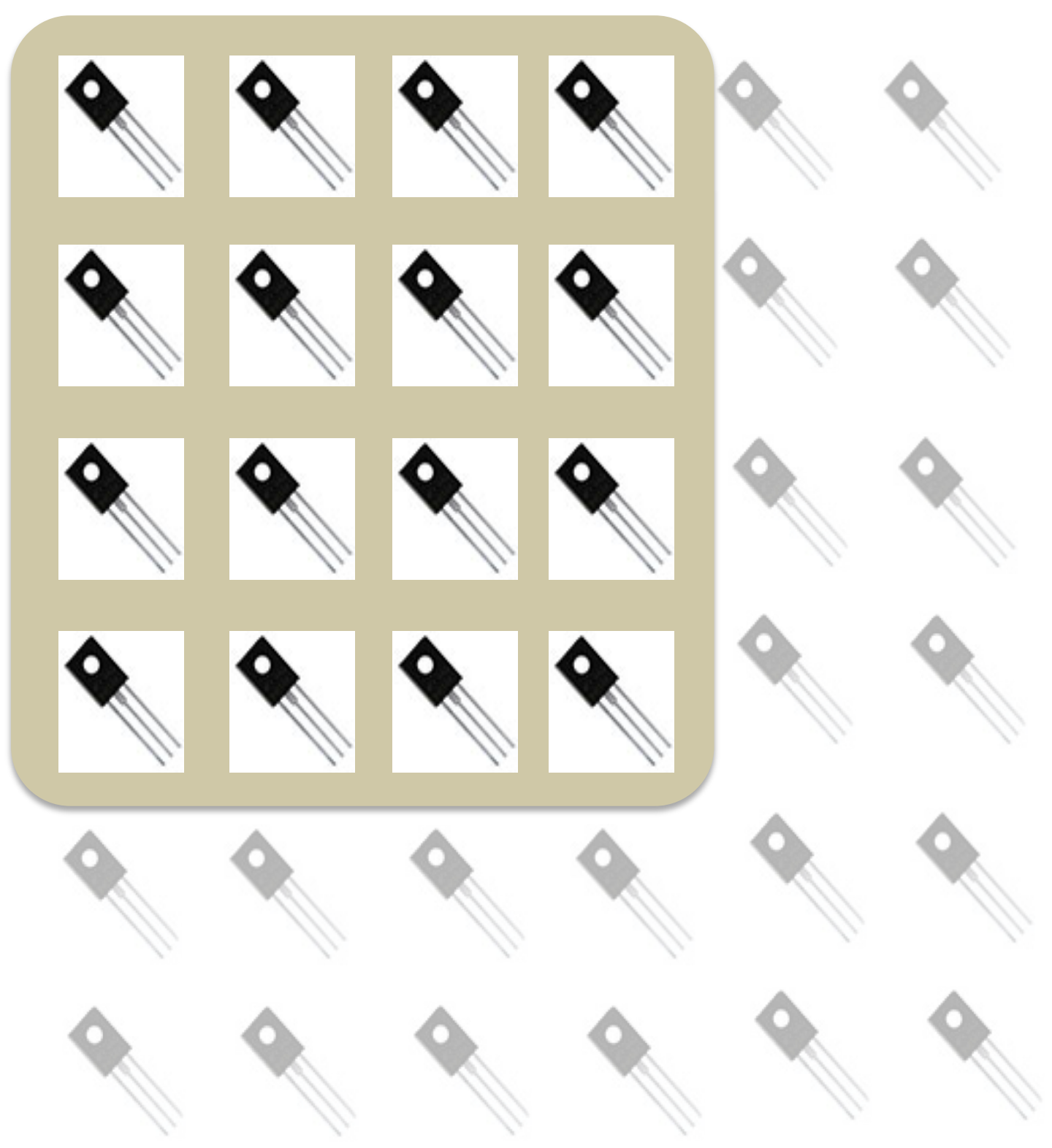
Rinnegan: Efficient Resource Use for Heterogeneous Processors

Sankaralingam Panneerselvam and Michael M. Swift

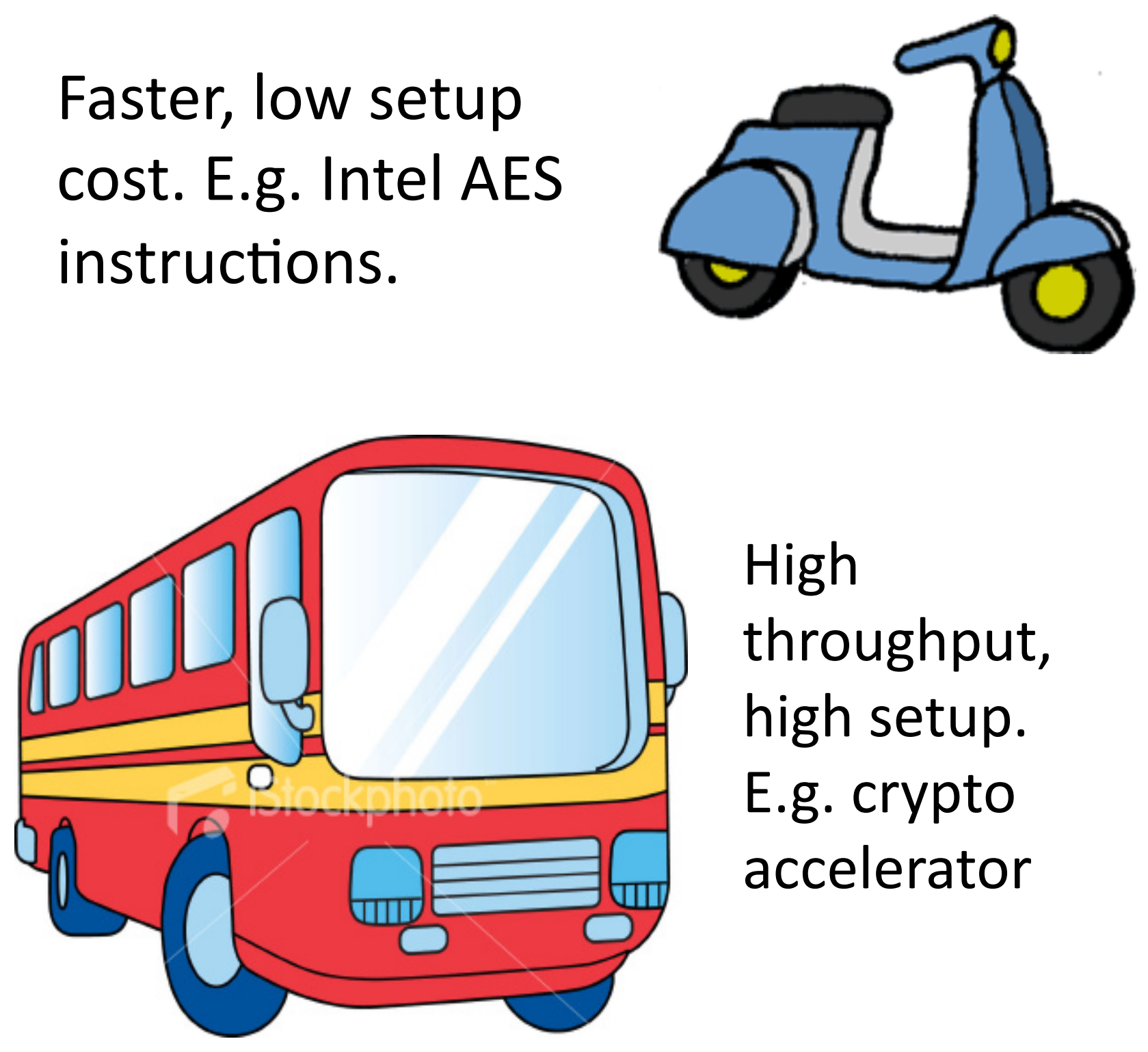
Challenges with Heterogeneous Architectures



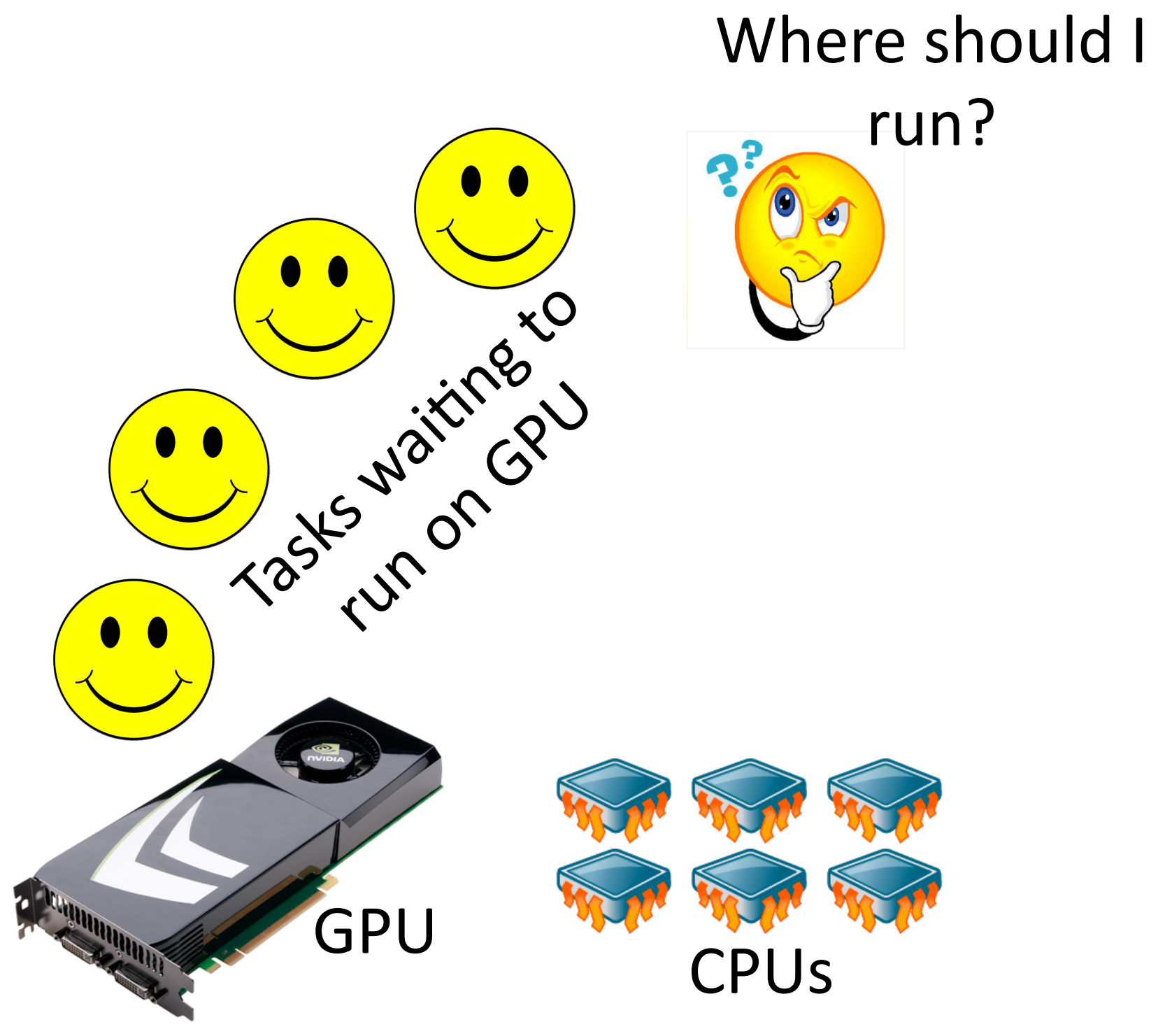
Heterogeneity will be common in future architectures



Dark Silicon demands better power management


Faster, low setup cost. E.g. Intel AES instructions.
High throughput, high setup. E.g. crypto accelerator

Latency-Power Trade-off



Overused accelerators may not always accelerate

Design

Goals

- **Adaptive Application**: Runtime decisions like task placement are made by applications through information exported from OS
- **Power Management:** Promotes power as a first-class resource; allow tasks to run only when power is available
- Optimize for performance or power/energy efficiency

Accelerator Stub

- Abstract the presence of different accelerators
- Runtime *binding to accelerator devices* by using information exposed by the monitor

Task Profiler

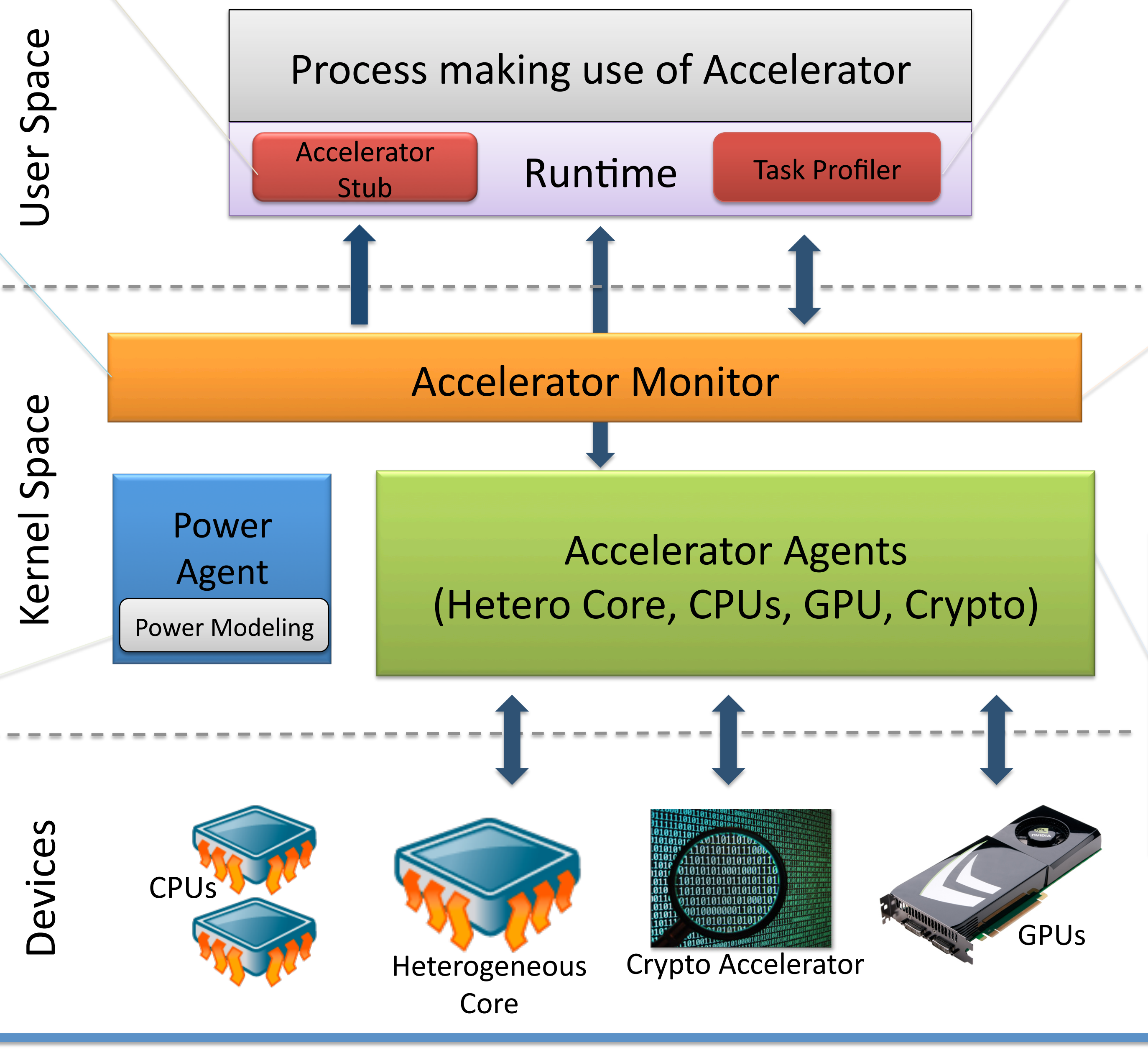
- Helps in predicting task execution time or task speedup on different accelerators
- Combined with information exported through monitor, helps in choosing an accelerator

Power Agent

- Enables to set a power limit for the system
- Controls the distribution of power to each application
- Transfers power from one task to another

Accelerator Monitor

- Centralized kernel service that exports device usage information from agents to user mode
- Supports a subscription mechanism to notify applications on resource allocation events



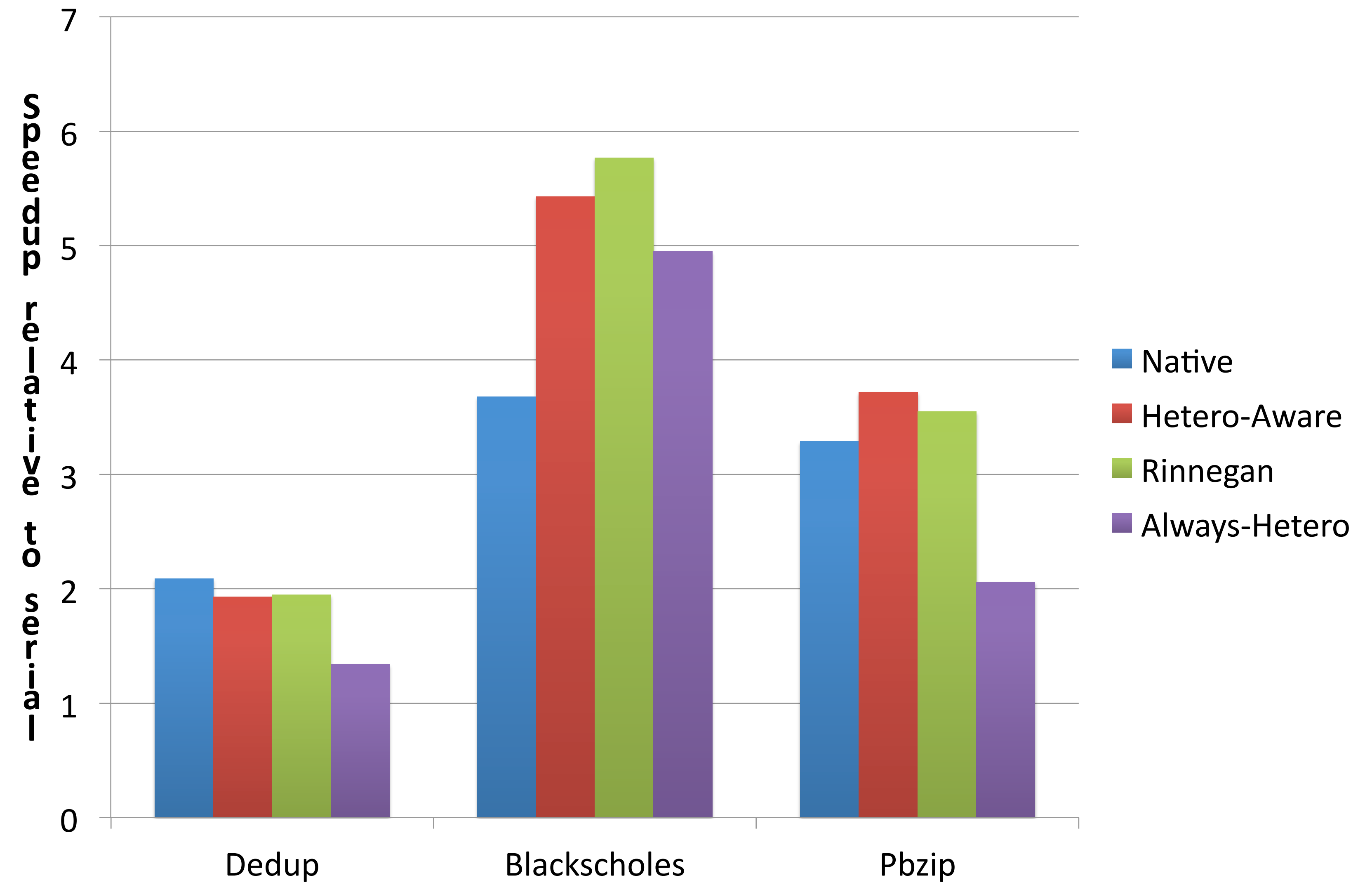
Power Model

- Provides information on power needed by a task to execute on a device
- Maintains information on power states supported by the devices (if any)

Accelerator Agent

- Implements scheduling decisions for the accelerator based on OS policy
- Exposes accelerator usage to the applications through monitor

Evaluation



Experiment:
Heterogeneous configuration - 10 slow cores and 2 fast cores - emulated using Intel's clock-modulation feature. Applications were run together contending for the heterogeneous core (powerful cores). Performance is normalized to serial version of the application running on a powerful core.

Configurations:

(a) *Native:* Unaware of heterogeneous cores and thus applies normal Linux scheduling policy over all the cores

(b) *Always-Hetero:* Runs tasks on the powerful cores

(c) *Hetero-Aware:* Modifies Linux scheduler with a simple heterogeneity-aware policy: tasks execute on normal cores but may be migrated to a powerful core if it becomes idle

(d) *Rinnegan:* Stub decides where to run tasks based on processing unit utilization