Asymmetry Everywhere (with Automatic Resource Management)

Onur Mutlu
onur@cmu.edu

Carnegie Mellon

The Setting

- Hardware resources are shared among many threads/apps in a data center (or peta-scale) system
  - Sockets, cores, caches, interconnects, memory, disks, power, lifetime, …
- Management of these resources is a very difficult task
  - When optimizing parallel/multiprogrammed workloads
  - Threads interact unpredictably/unfairly in shared resources
- Power/energy consumption is arguably the most valuable shared resource
  - Main limiter to efficiency and performance

Shield the Programmer from Shared Resources

- Writing even sequential software is hard enough
  - Optimizing code for a complex shared-resource parallel system will be a nightmare for most programmers

- Programmer should not worry about (hardware) resource management
  - What should be executed where with what resources

- Future computer architectures should be designed to
  - Minimize programmer effort to optimize (parallel) programs
  - Maximize runtime system's effectiveness in automatic shared resource management

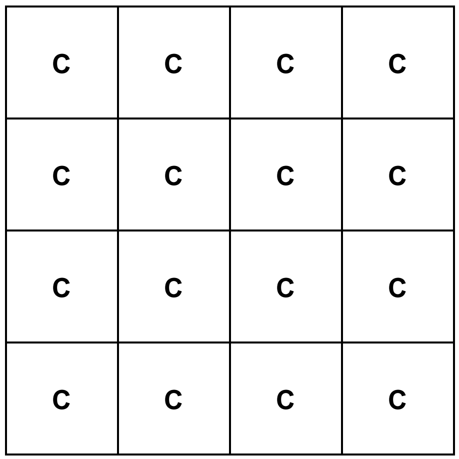
Shared Resource Management: Goals

- Future many-core systems should manage power and performance automatically across threads/apps

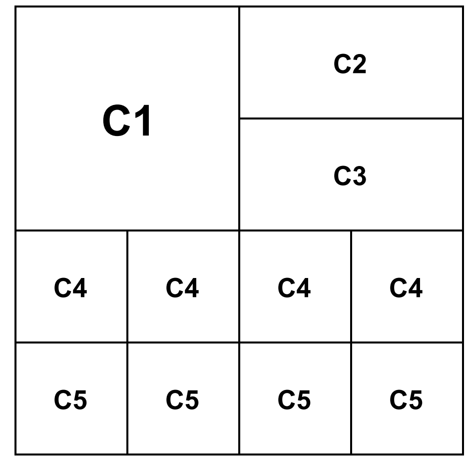
- Minimize energy/power consumption
- While satisfying performance/SLA requirements
  - Provide predictability and Quality of Service
- Minimize programmer effort
  - In creating optimized parallel programs

- Asymmetry and configurability in system resources essential to achieve these goals

Asymmetry Enables Customization



Symmetric

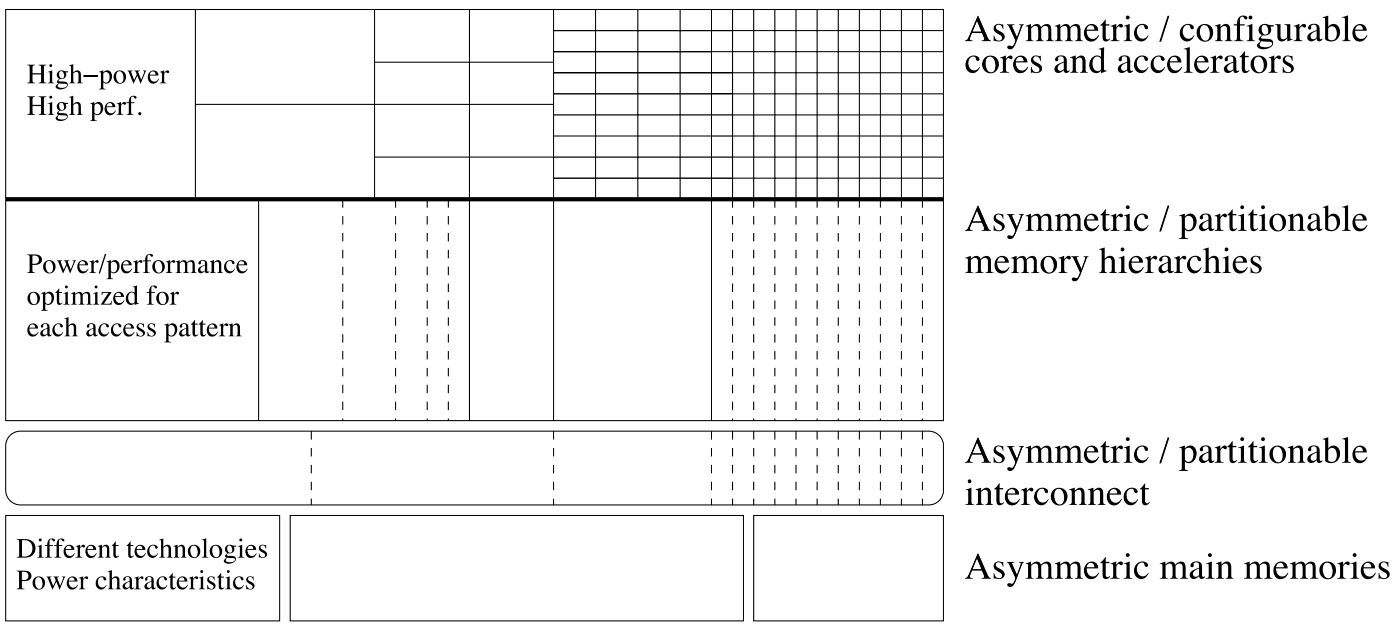


Asymmetric

- Symmetric: One size fits all
  - Energy and performance suboptimal for different phase behaviors
- Asymmetric: Enables tradeoffs and customization
  - Processing requirements vary across applications and phases
  - Execute code on best-fit resources (minimal energy, adequate perf.)

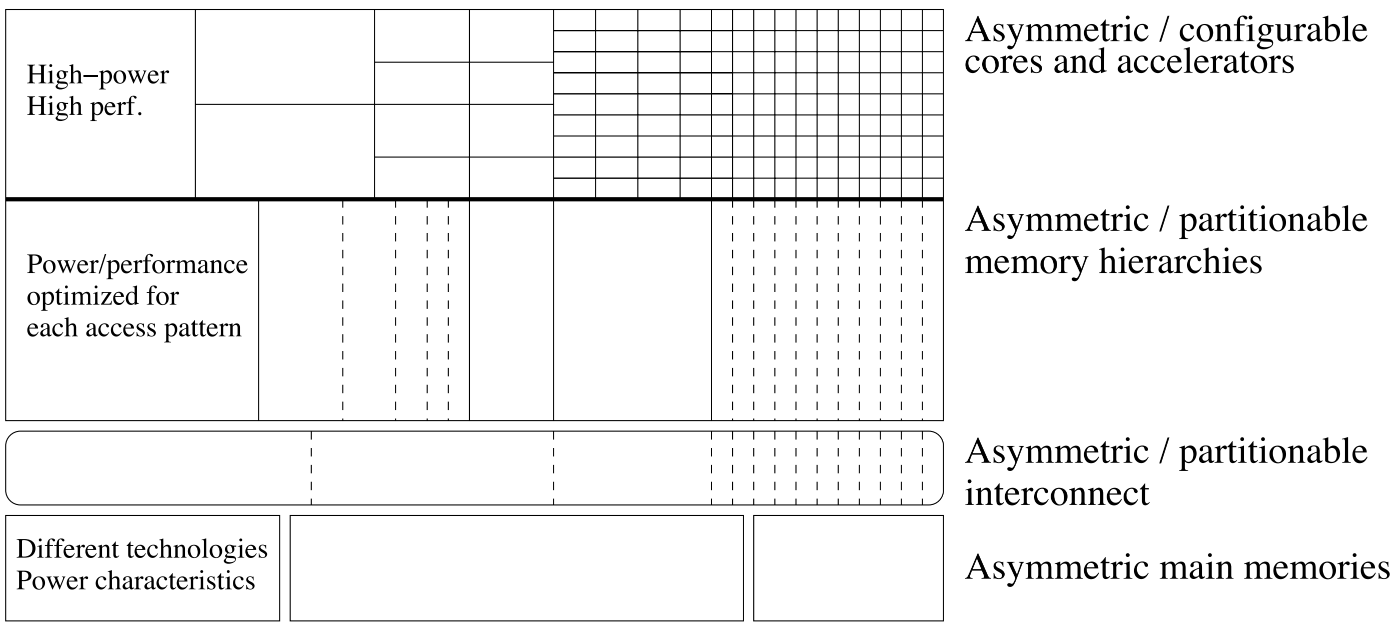
Our Position: Asymmetry Everywhere

- Design each hardware resource with asymmetric, (re-)configurable, partitionable components
  - Different power/performance/reliability characteristics
  - To fit different computation/access/communication patterns



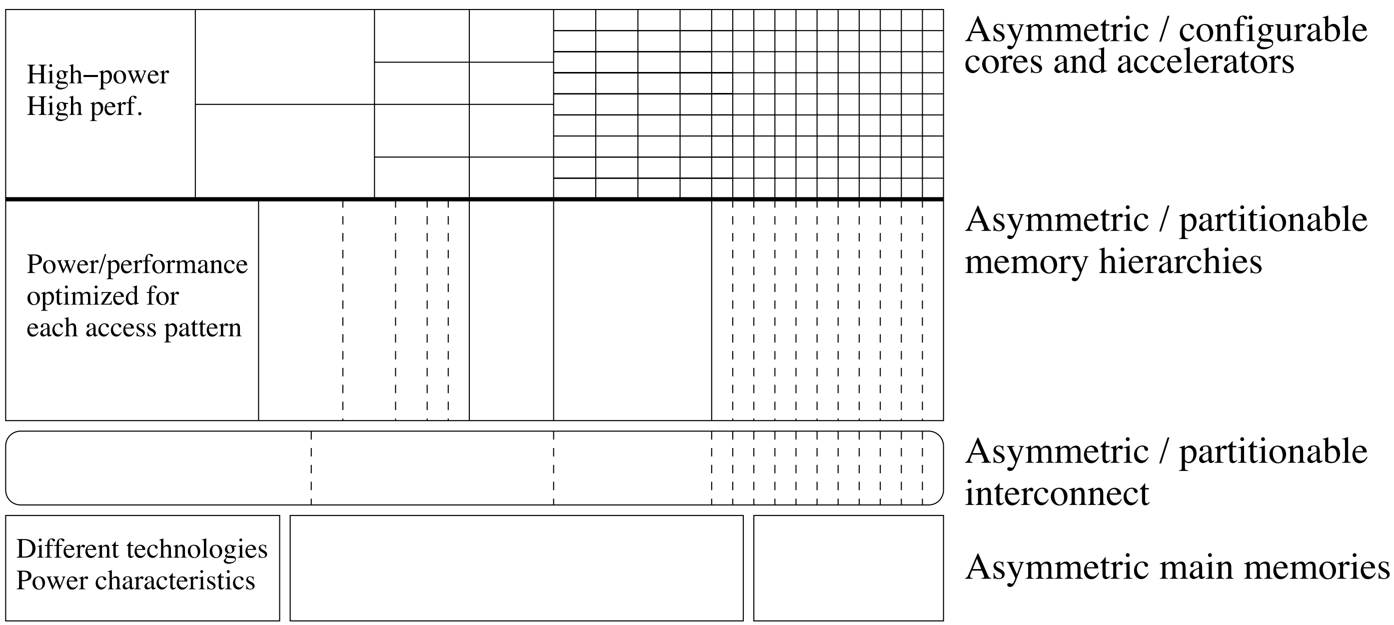
Our Position: Asymmetry Everywhere

- Design the runtime system (HW & SW) to automatically choose the best-fit components for each phase
  - Satisfy performance/SLA with minimal energy
  - Dynamically stitch together the "best-fit" chip for each phase



Our Position: Asymmetry Everywhere

- Morph software components to match asymmetric HW components
  - Multiple versions for different resource characteristics



Our Position: Asymmetry Everywhere

- Design each hardware resource with asymmetric, (re-)configurable, partitionable components

- Design the runtime system (HW & SW) to automatically choose the best-fit components for each phase

- Morph software components to match asymmetric HW components

Many Research Questions

- How to design asymmetric components?
  - Fixed, partitionable, reconfigurable components?
  - What types of asymmetry? Access patterns, technologies?

- What monitoring to perform cooperatively in HW/SW?
  - To characterize a phase and match it to best-fit components
  - Automatically discover phase/task requirements

- How to design feedback/control loop between components and runtime system software?

- How to design the runtime to automatically manage resources?
  - Track task behavior, pick "best-fit" components for the entire workload

Summary

- Need to minimize energy while satisfying performance requirements
  - While also minimizing programmer effort

- Asymmetry key to energy/performance tradeoffs

- Design systems with many asymmetric/partitionable components
  - Many types of cores, memories, interconnects, …
  - Partitionable/configurable components, customized accelerators on chip

- Provide all-automatic resource management
  - Impose structure: HW and SW cooperatively map phases to components
  - Dynamically stitch together the system that best fits the running tasks

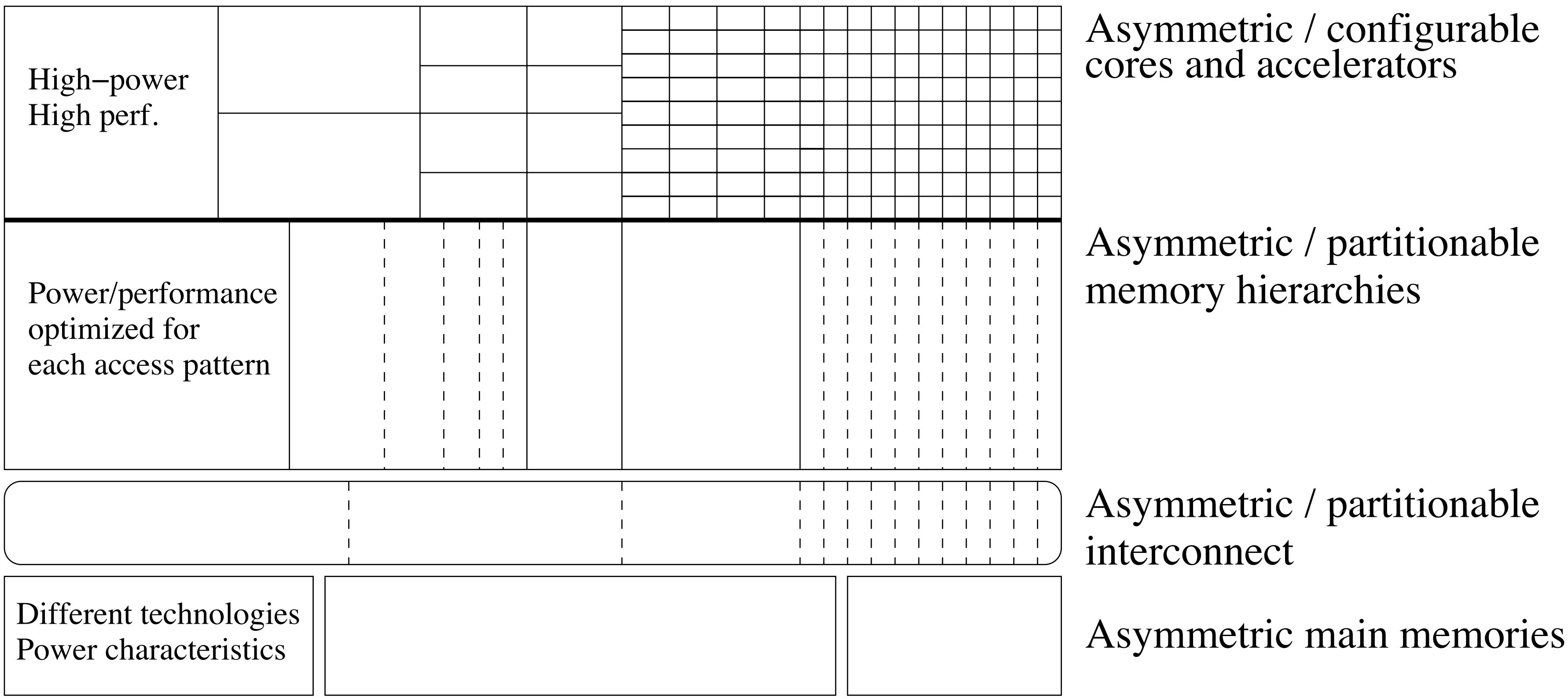
- Programmer does not need to worry about complex resource sharing

Asymmetry Everywhere (with Automatic Resource Management)

Onur Mutlu
onur@cmu.edu

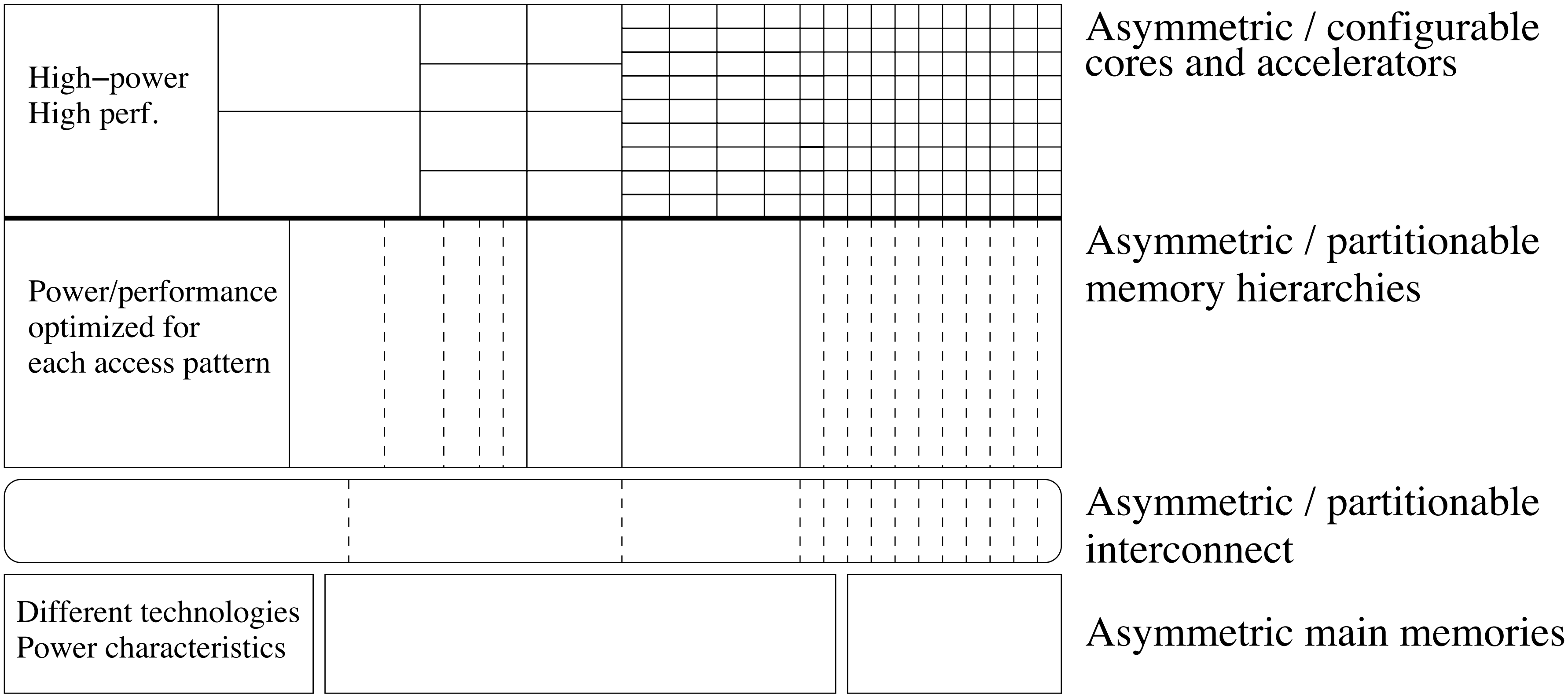
Carnegie Mellon

Exploiting Asymmetry: Simple Examples



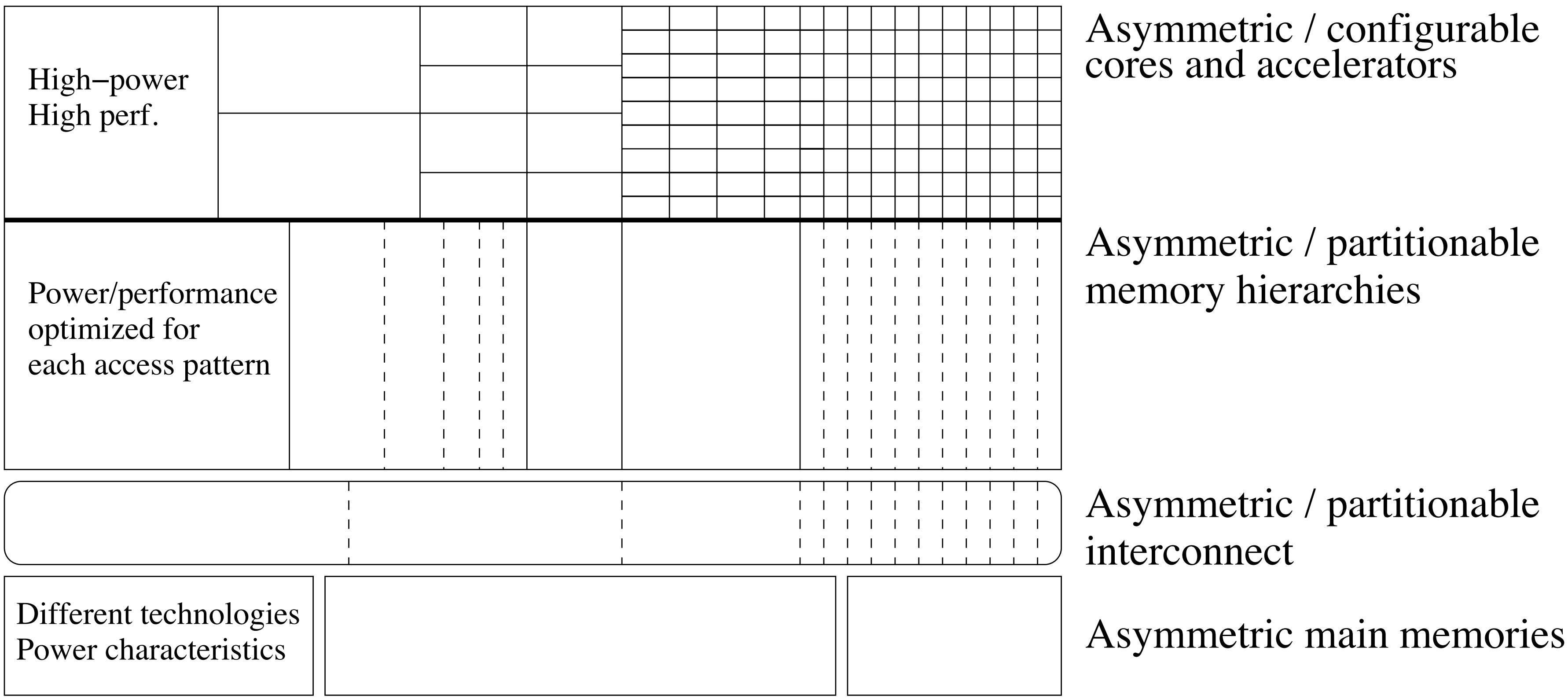
- Execute critical/serial sections on high-power, high-performance cores/resources
  - Programmer can write less optimized, but more likely correct programs

Exploiting Asymmetry: Simple Examples



- Execute streaming "memory phases" on streaming-optimized cores and memory hierarchies
  - More efficient and higher performance than general purpose hierarchy

Exploiting Asymmetry: Simple Examples



- Partition memory controller and on-chip network bandwidth asymmetrically among threads
  - Higher performance and energy-efficiency than symmetric/free-for-all

Possible Promising Directions

- Flexible core and shared resource designs

- Hardware/software cooperation
  - Need to optimize the entire stack (hardware + system software)

- Use of learning
  - Complex tasks, continuous dynamic optimization
  - Collective across time and space (entire system)

- Collaboration
  - Academia/industry
  - Across architecture, distributed systems, networking, theory, ML