Communication-Avoiding Gang Scheduling of Resources in Tessellation OS

Juan A. Colmenares and John D. Kubiatowicz
Par Lab, CS Division, UC Berkeley

Parallel Computing Laboratory

1. Motivation

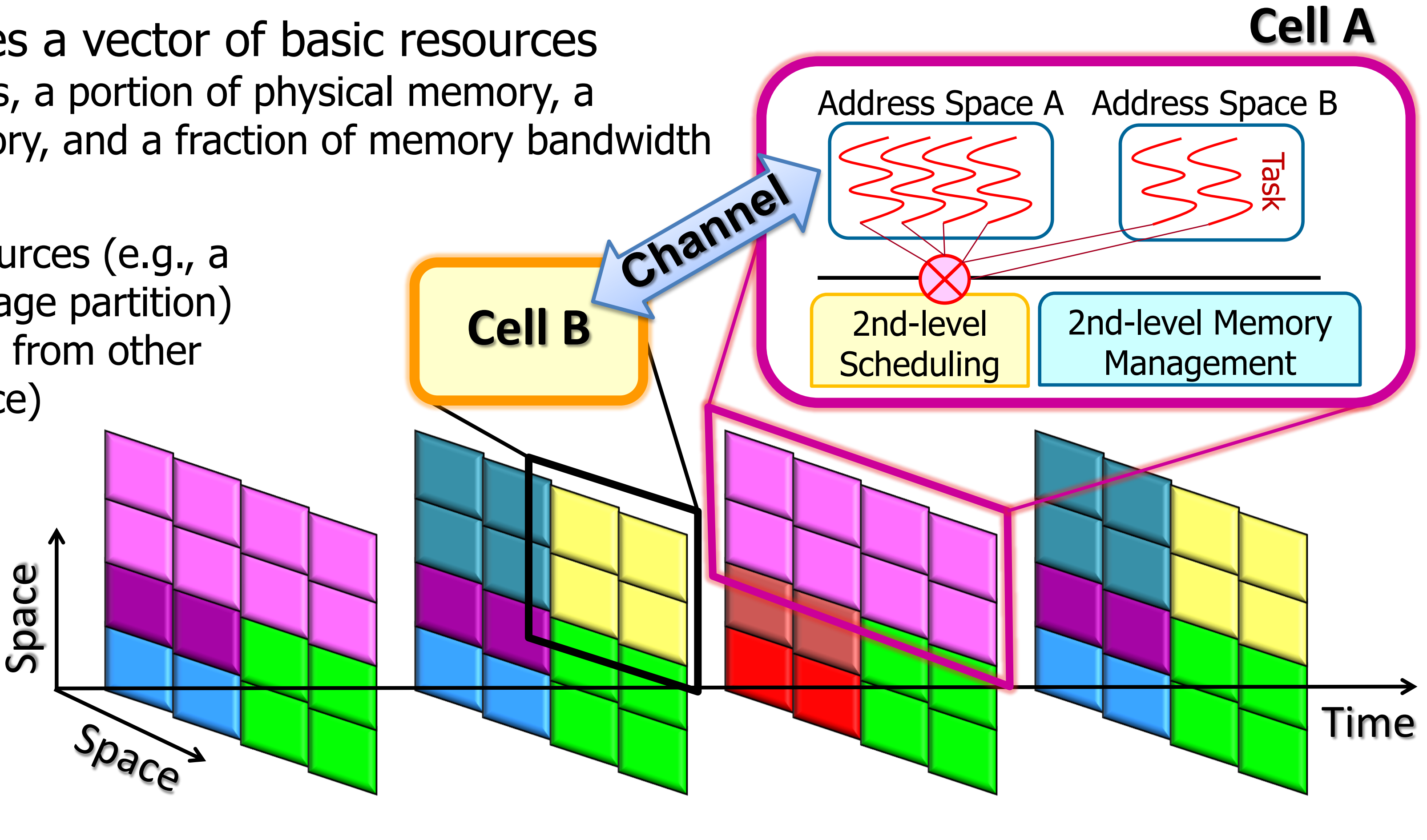
- Performance gap between computation and communication
- Communication cost increases with core count
- We want to determine the actual benefits of trading communication for computation in resource gang-scheduling, which is key to Tessellation OS?

2. Basic Goals in Tessellation OS

- Support a simultaneous mix of high-throughput parallel, interactive, and real-time applications
- Allow applications to consistently deliver performance

3. Space-time Partitioning and Two-level Scheduling

- A **Spatial Partition** receives a vector of basic resources
  - A number of hardware threads, a portion of physical memory, a portion of shared cache memory, and a fraction of memory bandwidth
- A Partition may also receive
  - Exclusive access to other resources (e.g., a hardware device and raw storage partition)
  - Guaranteed fractional services from other partitions (e.g., network service)
- Spatial partitioning may vary over time
  - Partitions can be **time multiplexed**; resources are **gang-scheduled**
  - Partitioning adapts to needs of the system



- **The Cell**: Our partitioning abstraction
  - User-level software container with guaranteed access to resources
- Basic properties of a cell
  - Full control over resources it owns when mapped to hardware
  - One or more address spaces
  - Communication channels

- **Scheduling at Level 1**: Coarse-grained resource allocation and distribution at the cell level
- **Scheduling at Level 2**: Fine-grained application-specific scheduling *within* a cell

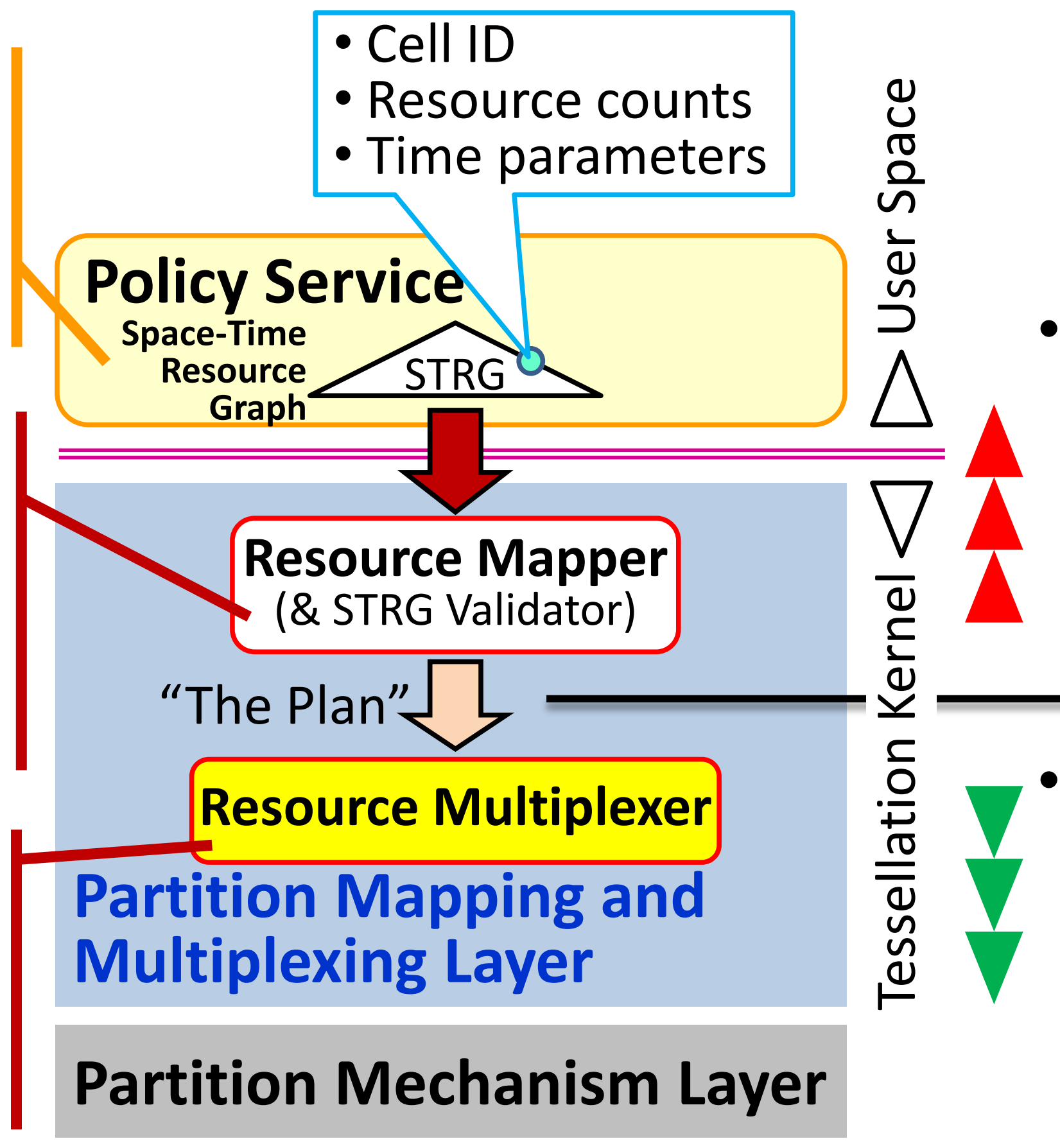
4. Resource Allocation Architecture

A Partial and Simplified View

- Distributes resources among cells
- Establishes how cells should be time multiplexed

- Assigns specific resources to cells
- Produces only feasible mappings
  - Rejects invalid and infeasible STRGs

- Determines when cells should be activated and suspended
- Actually activates and suspends cells



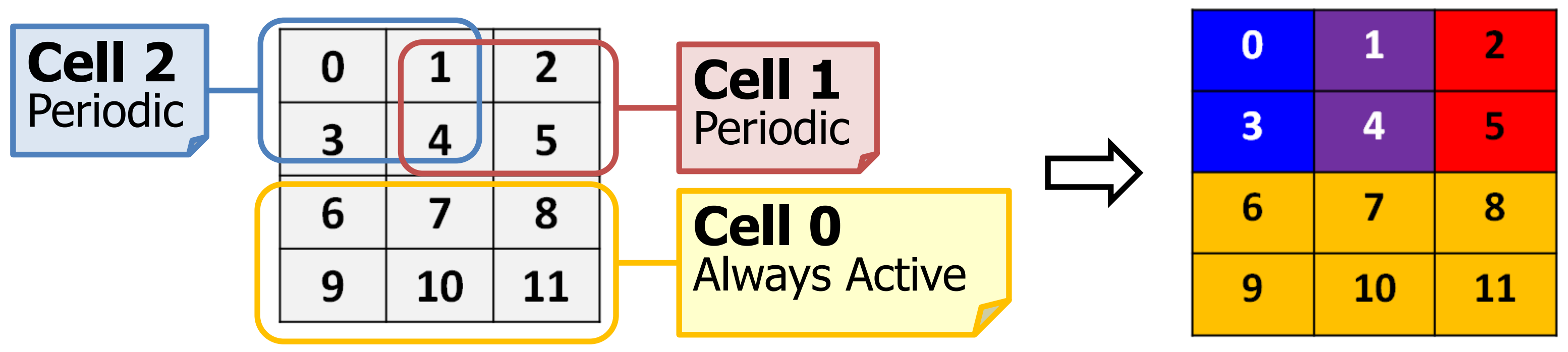
Separation between Mapper and Multiplexer

- Decision Making Process
  - Mapping of multiple resources
  - Centralized because it requires global knowledge
  - Often expensive
- Execution
  - Relatively simple and fast
    - If the set of cells does not change
  - Allows us to explore decentralized approaches

5. Communication-Free and Centralized Multiplexers

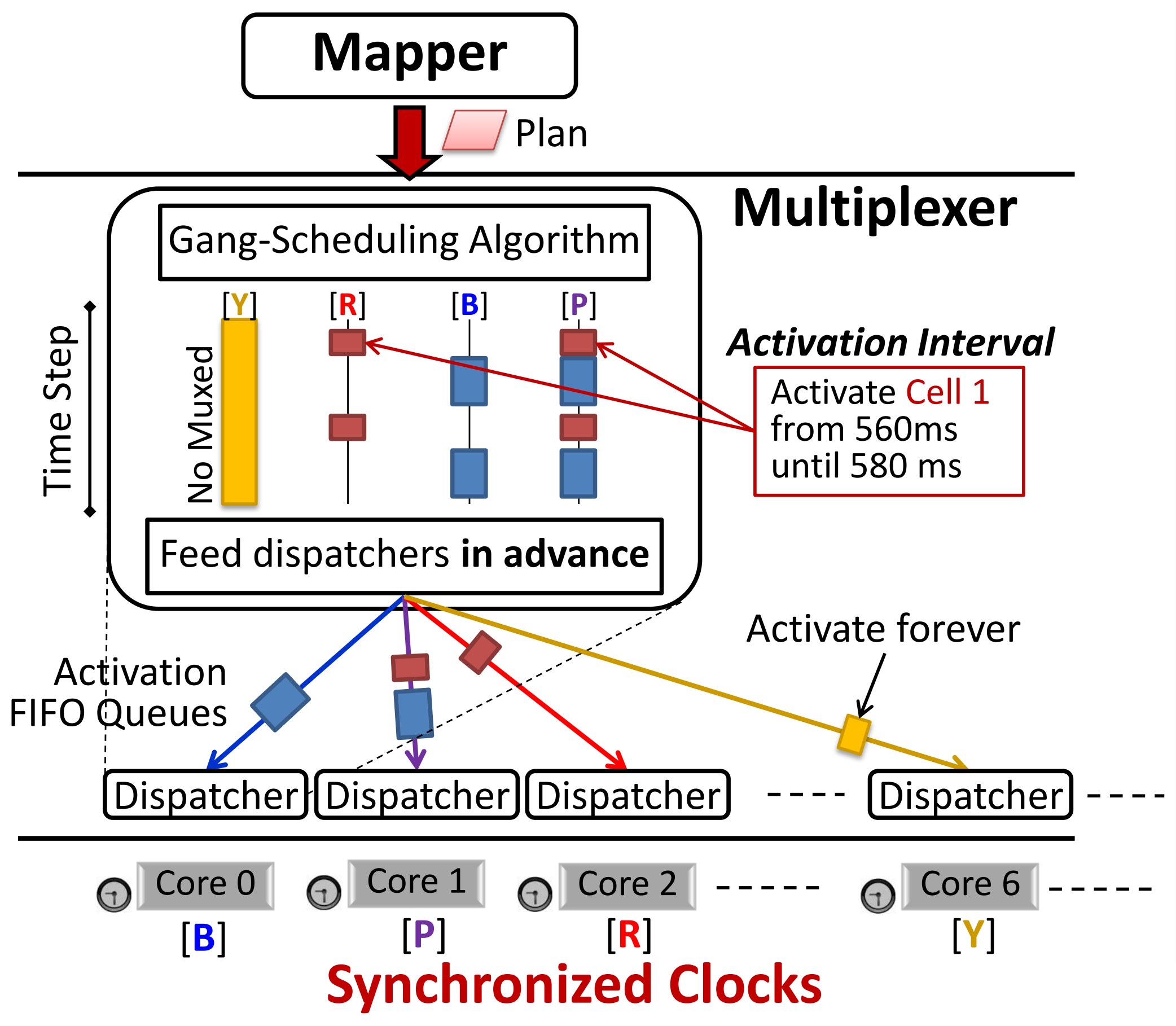
Sample Cell-Core Mapping

Produced by the **Mapper** after checking feasibility

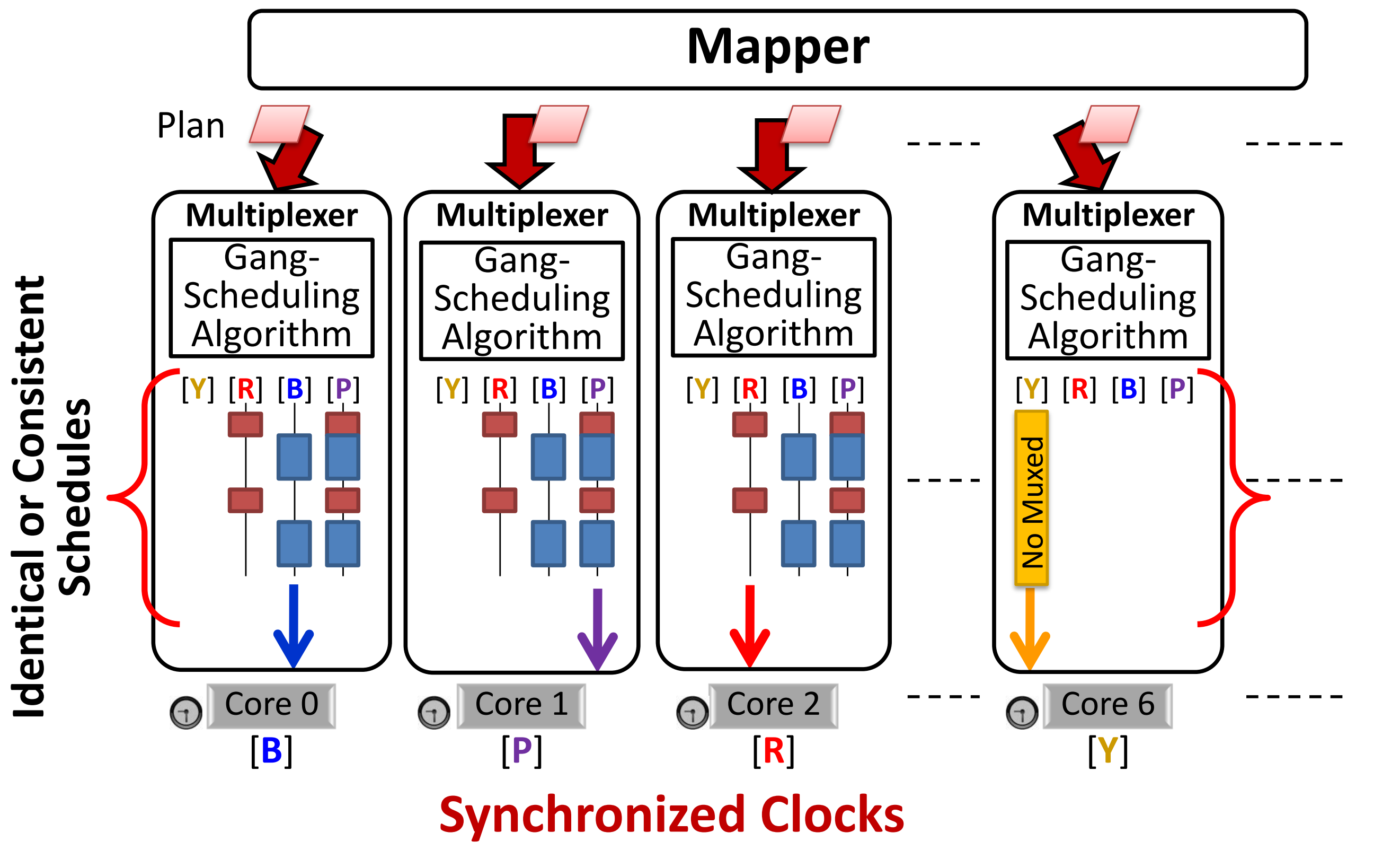


- Cores with **same color**
  - Have the **same cells** assigned to them
  - Run the **same schedule**

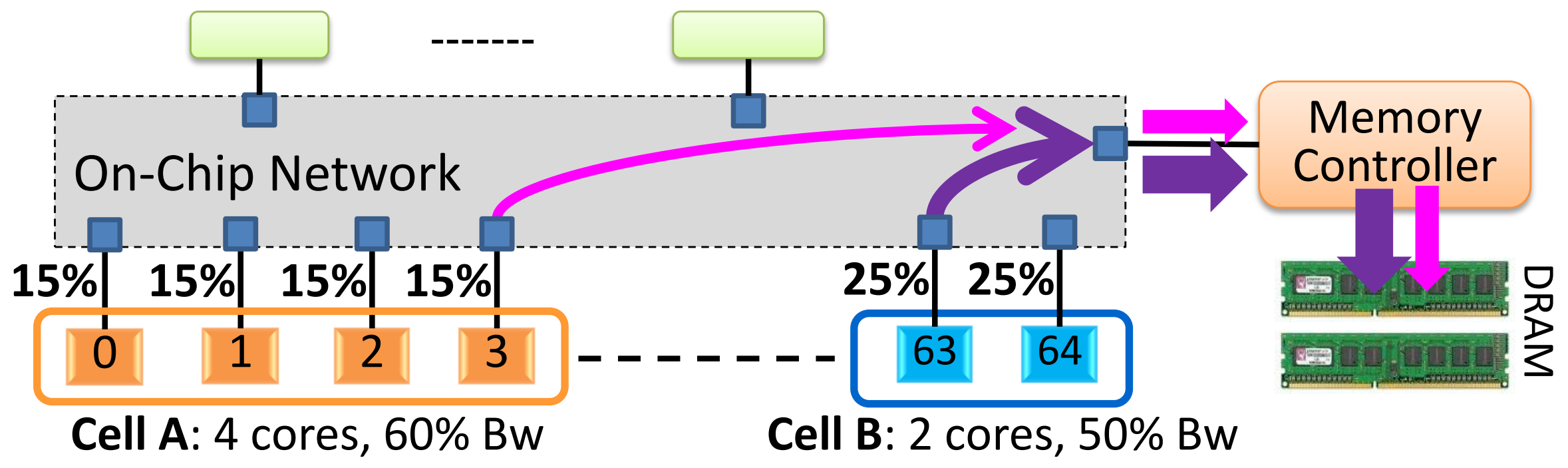
A Centralized Version



Communication-avoiding Version

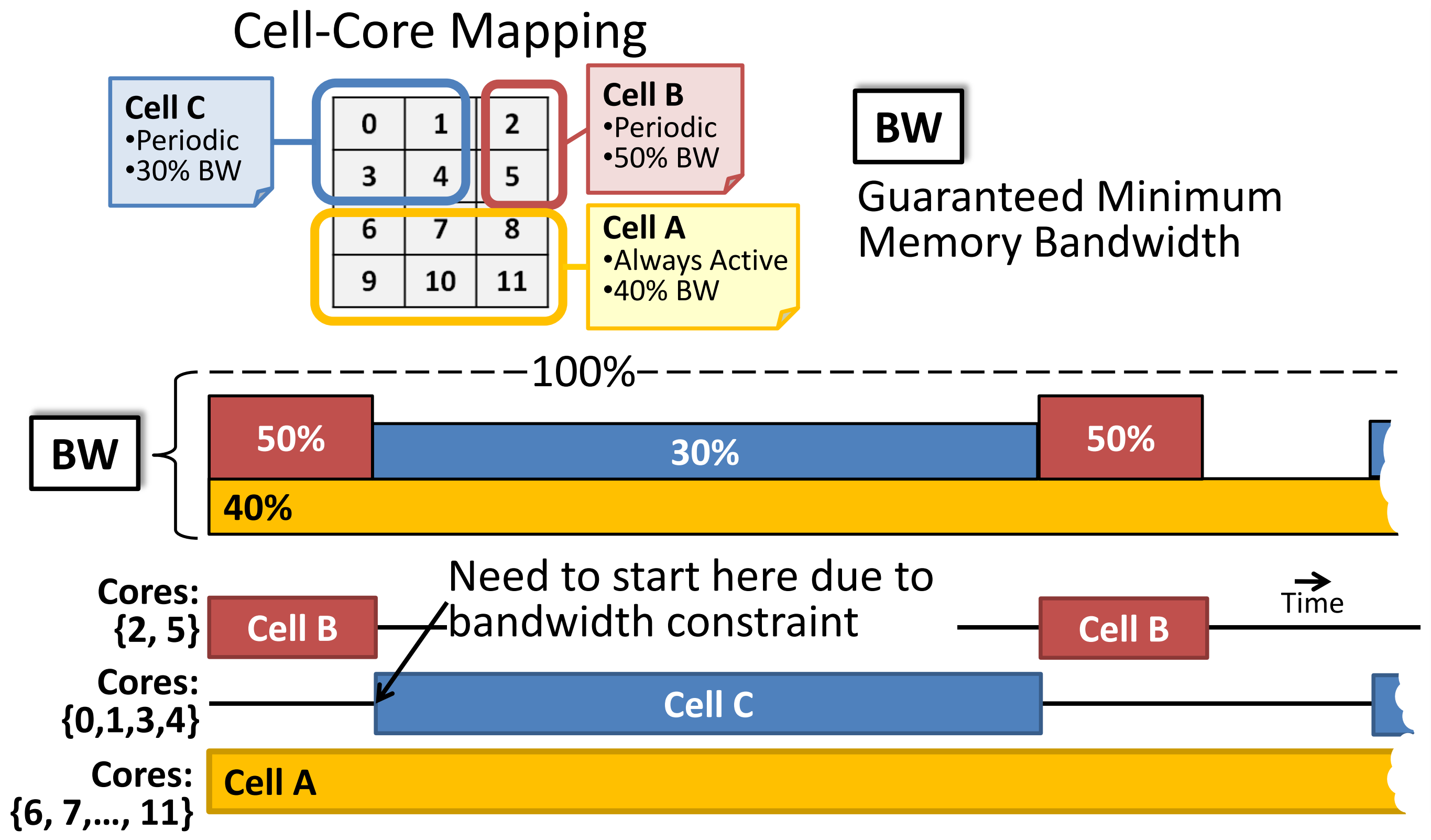


6. Gang-Scheduling of Cores and Memory Bandwidth

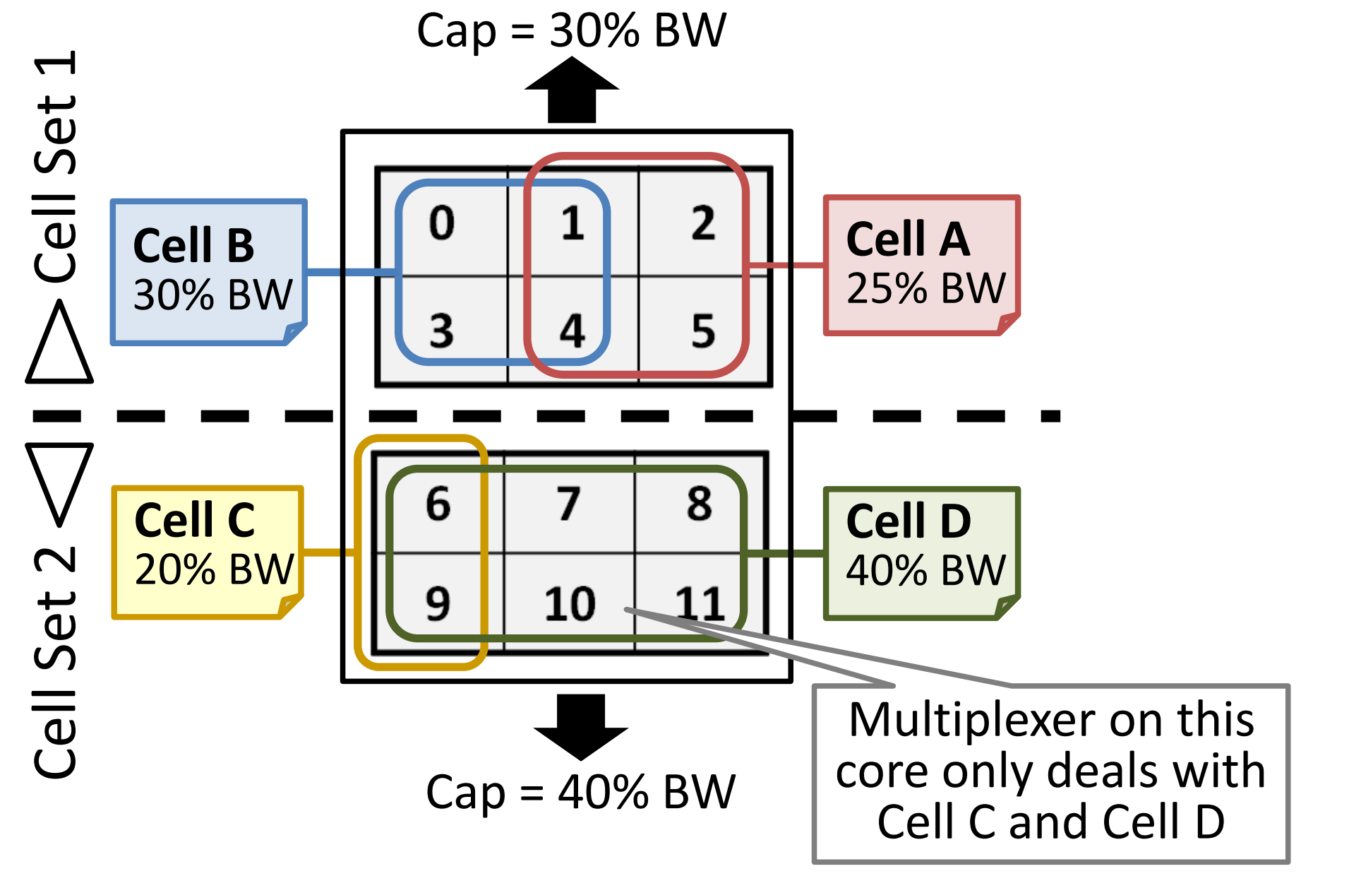


Memory bandwidth is a global **shared** resource

Assume that each core can request its own guaranteed minimum fraction of memory bandwidth



- Independent sets of cells with caps on guaranteed bandwidth
  - Less computation cost for each multiplexer



7. Status

- Initial versions of the gang-scheduling algorithm, centralized multiplexer, and communication-free multiplexer exist and they are being tested
- Implementation of the Mapper is underway

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG0710227)
We like to thank other members of the Par Lab OS Group, especially Sarah Bird, Gage Eads, Steven Hofmeyr, and Krste Asanovic.