# RESOURCE PARTITIONING ON COMMODITY SERVERS

David Daharewa Gureya

SCS, EECS

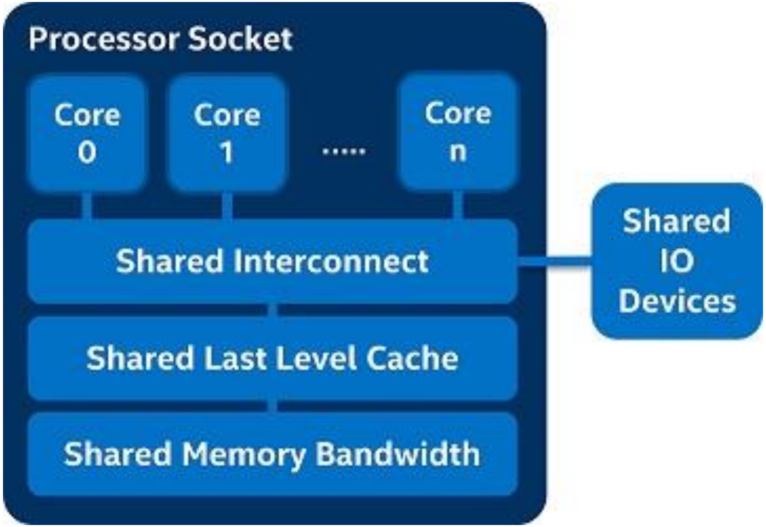
# PAPERS

- CoPart: Coordinated Partitioning of Last-Level Cache and Memory Bandwidth for Fairness-Aware Workload Consolidation on Commodity Servers [EuroSys 2019]
- PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services [ASPLOS 2019]
- SWAP: Effective Fine-Grain Management of Shared Last-Level Caches with Minimum Hardware Support [HPCA 2017]

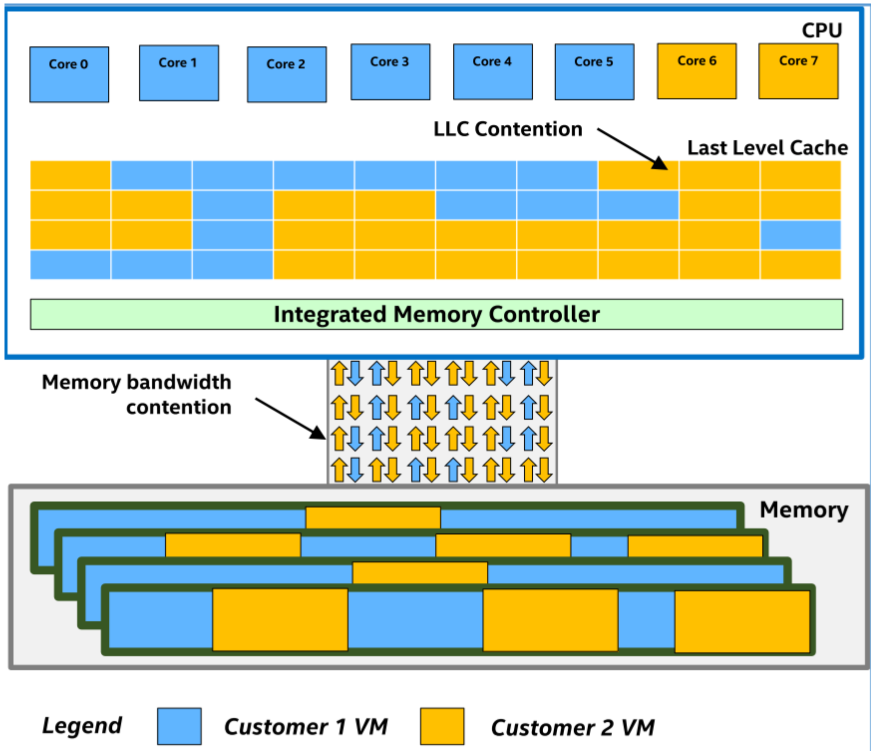
# Colocation of Applications

- Workload consolidation is widely used to improve resource utilization
  - Multiple workloads are consolidated on the same physical servers
  - Cost efficiency
- Challenge: Performance interference among consolidated workloads
  - Mainly caused by the contention over shared hardware resources

# Multi-core Processor



# Interference During Colocation



# Tackling Interference

- Avoid sharing resources with other applications
  - Preserves QoS, lowers resource utilization

- Avoid co-scheduling of apps that may interfere
  - May require offline knowledge
  - Limit colocation options

- Partition shared resources
  - Improves system throughput
  - Guarantee QoS of latency-critical workloads
  - Eliminating timing channels

# CoPart

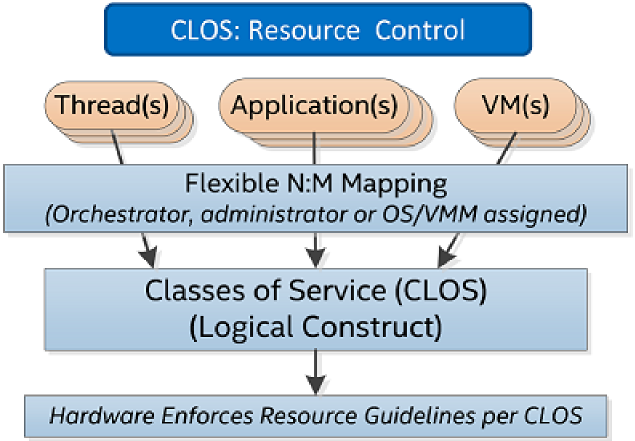
- Recent commodity CPUs support LLC and Memory BW partitioning
  - Which are highly performance-critical shared hardware resources

- Coordinated partitioning of LLC and Memory BW is unexplored

- Coordinated partitioning of LLC and Memory BW for fairness-aware workload consolidation on commodity servers
  - Dynamically analyses the application characteristics
  - Partitions LLC and memory BW in a coordinated manner

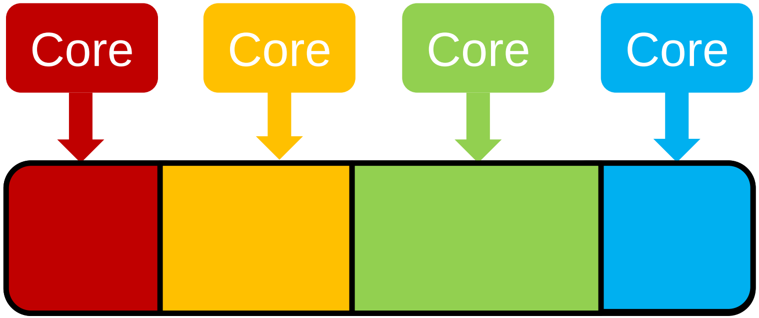
# LLC and Memory BW Partitioning

- X86-64 architecture partitions HW resources across the CLOSes
  - Each CLOS consists of a group of cores or processes



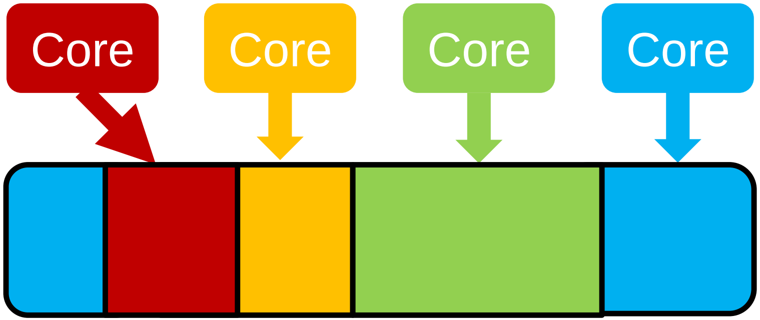
# LLC Partitioning

- Intel Cache Allocation Technology (CAT)
  - Hardware support for LLC partitioning based on way partitioning
    - > *Assign different cache ways to different cores*
    - > *Perfect isolation*
    - > *Low repartition overhead*



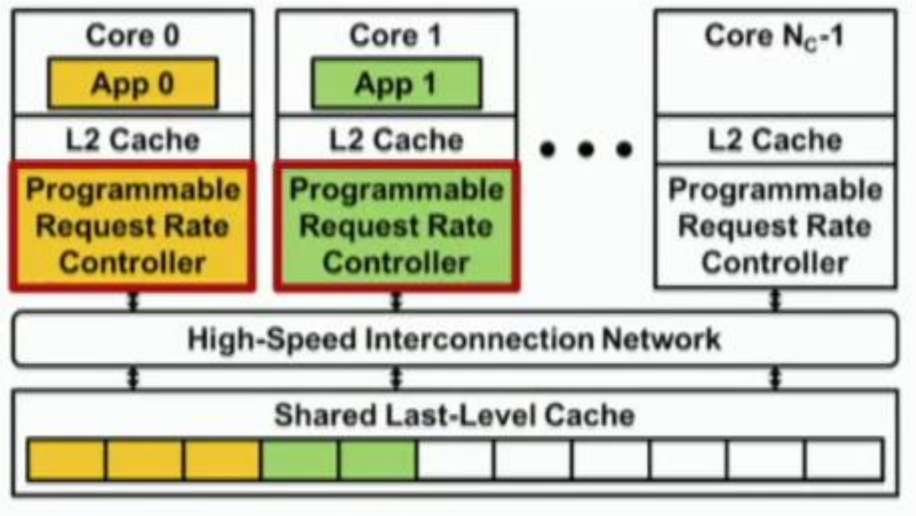
# LLC Partitioning

- Intel Cache Allocation Technology (CAT)
  - Hardware support for LLC partitioning based on way partitioning
    - > *Assign different cache ways to different cores*
    - > *Perfect isolation*
    - > *Low repartition overhead*



# Memory BW Partitioning

- Intel Memory Bandwidth Allocation (MBA)
  - Controls the traffic between the L2 cache and the LLC



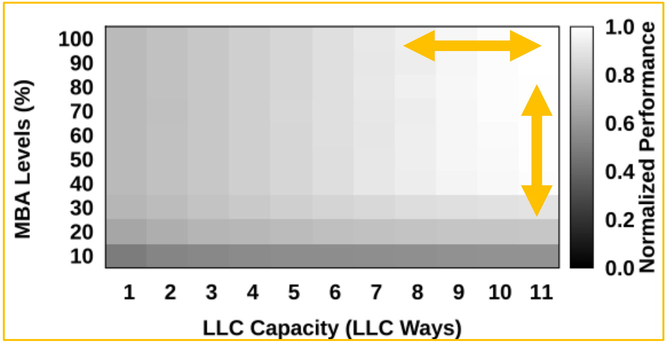
# Terminology and Methodology

- $N_A$ applications are consolidated on the same physical server
- $S_i$ (resource allocation state): ($l_i$, $m_i$) (LLC ways and MBA level)
- $S$ (system state): $\{S_0, S_1, \ldots, S_{N_A\text{-}1}\}$
- Unfairness = σ / μ
  - **σ = the standard deviation of the slowdowns of the applications**
  - **μ = the average slowdown across the consolidated applications**

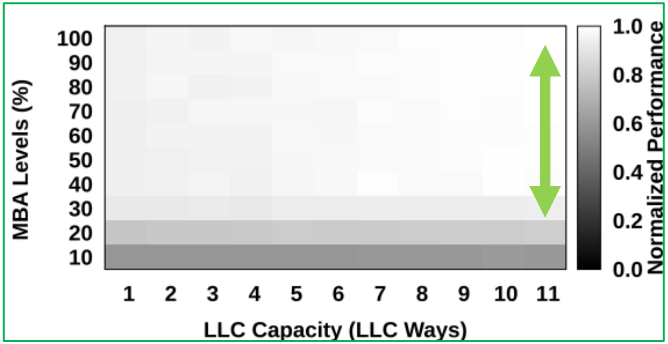
- System configuration
  - Intel Xeon Gold 6130 Processor CPU @ 2.1 GHz (16 cores)
  - Memory: 32GB (2 x 16GB DDR4), total BW of 28GB/s
  - LLC: Shared, 22MB, **11 ways**, dynamically assigned through CAT
  - MBA level can be changed from **100% (no throttling) to 10%**

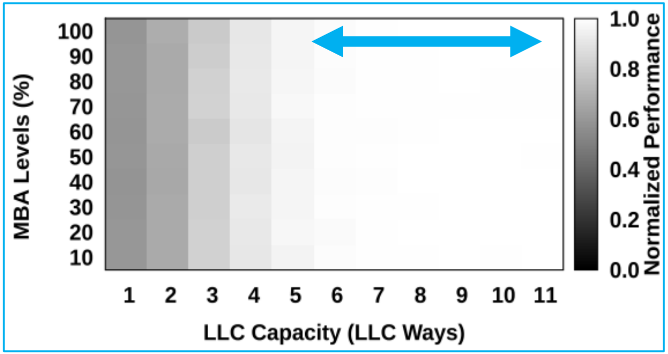
# Performance characterization

(a)




(b)

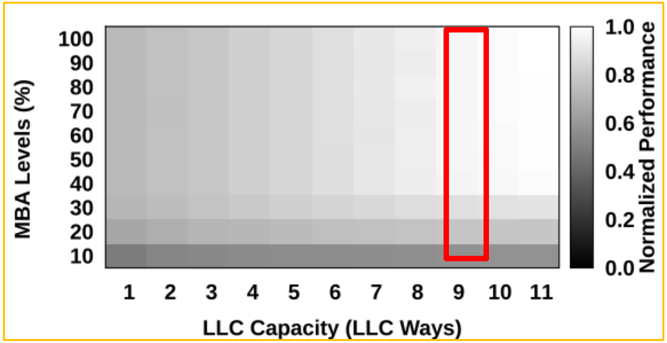


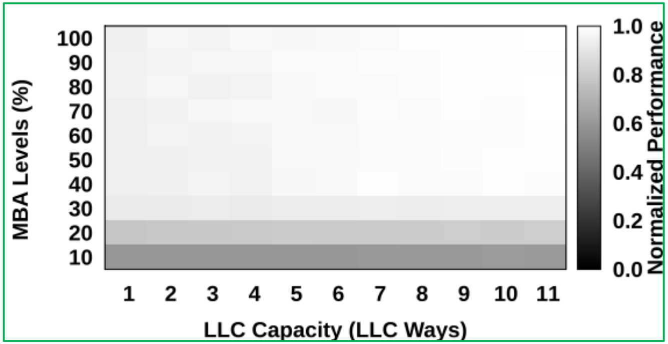

(c)

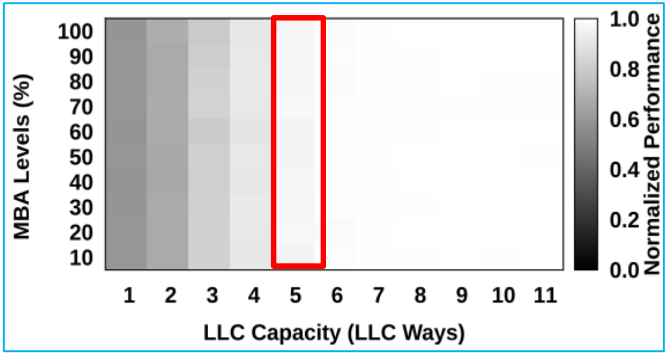
# Performance characterization

(a)




(b)

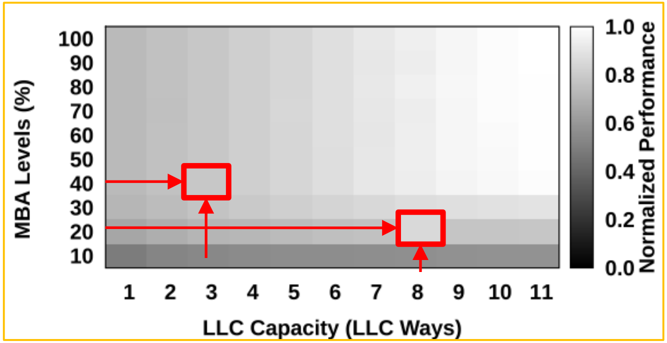


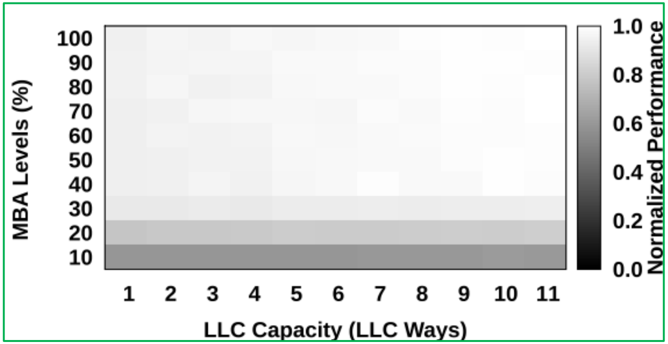

(c)

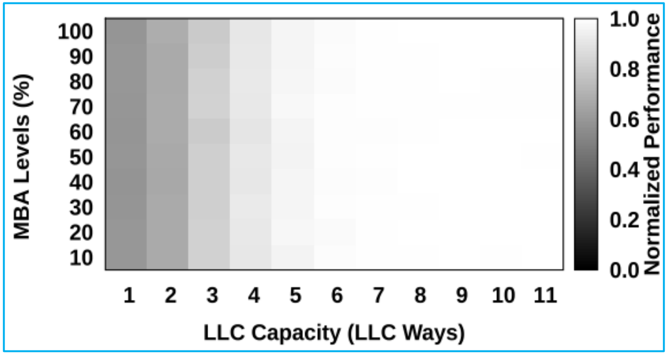
# Performance characterization

(a)




(b)




(c)

# Fairness characterization

- Fairness is dependent on both LLC and memory BW partitioning

- Conclusion:
  - This trends indicate that coordinated partitioning of LLC and memory BW is highly crucial

# Design and Implementation of CoPart

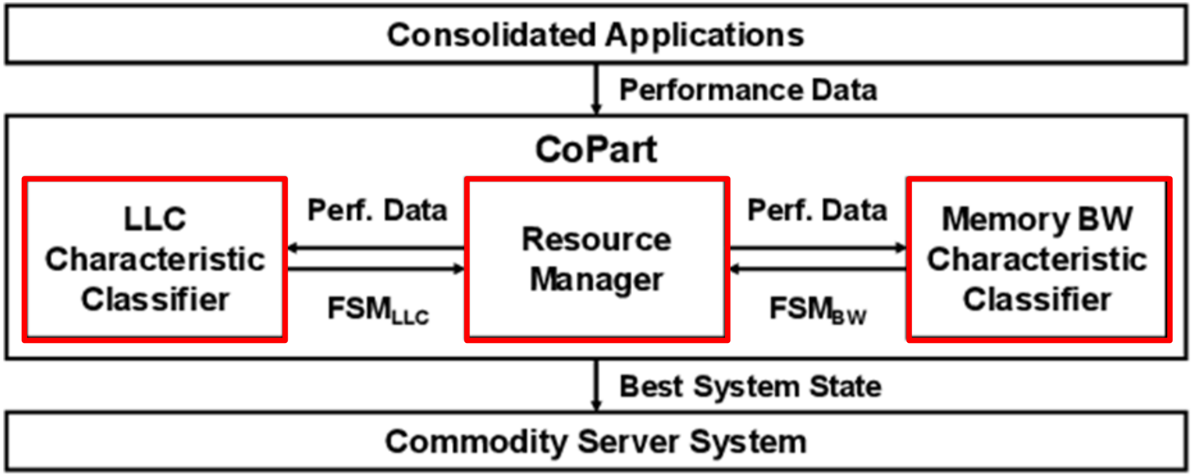
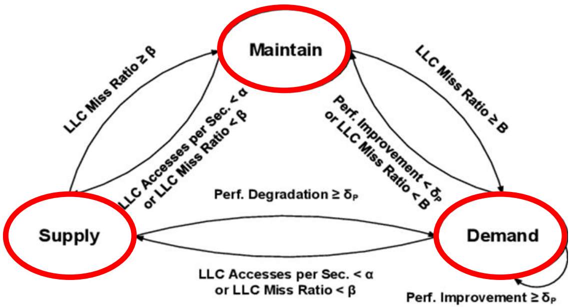



**Figure 7.** Overall architecture of CoPart

# LLC Characteristic Classifier



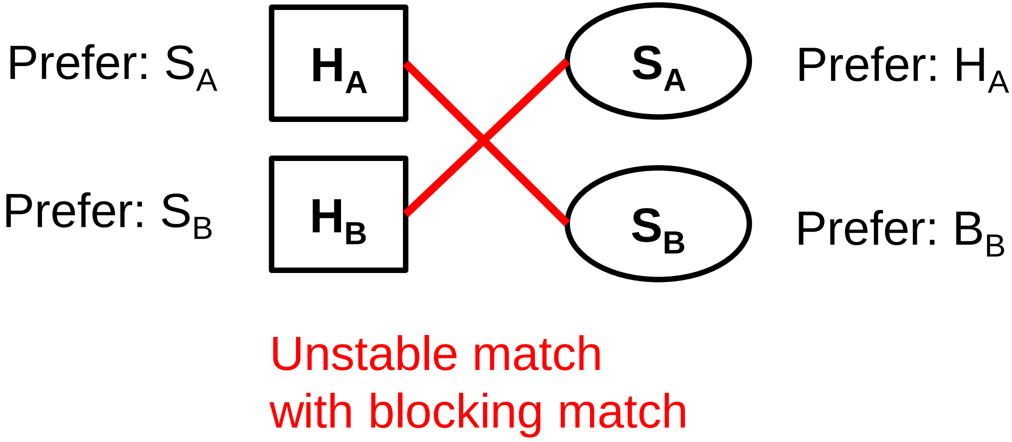

- **Supply: Can supply one of the allocated LLC ways**
  - ✓ Sufficiently low LLC access rate or LLC miss ratio
- **Demand: Demand more LLC ways to improve performance**
  - ✓ Performance can be improved with additional LLC way
- **Maintain: Needs to maintain the currently allocated LLC way**
  - ✓ Allocating an additional LLC way provides marginal performance gains
  - ✓ Reclaiming an LLC way significantly degrades the performance

# Memory BW Characteristic Classifier

- Designed similarly to the LLC characteristic classifier

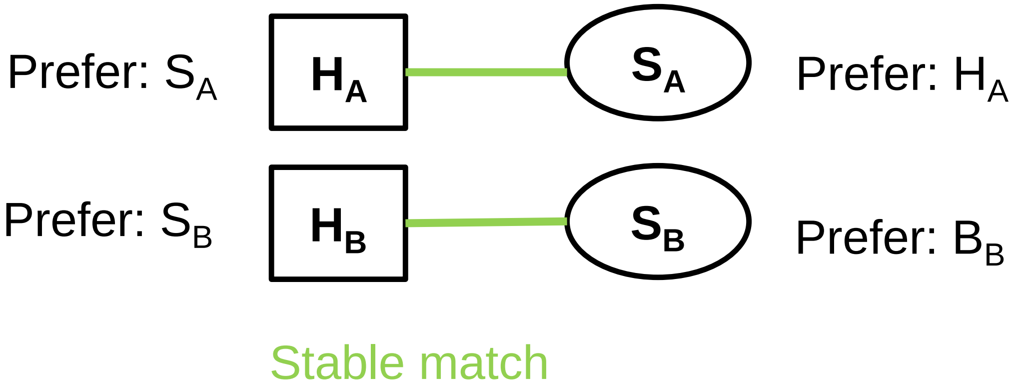
# Resource Manager

- System state space exploration phase
  - Hospitals/Residents (HR) problem
    - > *Extensively-studied and widely-applied problem in economics*
    - > *H hospitals and R medical students with preference lists*
    - > *Finds a stable match that contains no blocking pairs*



# Resource Manager

- System state space exploration phase
  - Hospitals/Residents (HR) problem
    - > *Extensively-studied and widely-applied problem in economics*
    - > *H hospitals and R medical students with preference lists*
    - > *Finds a stable match that contains no blocking pairs*



# Resource Manager

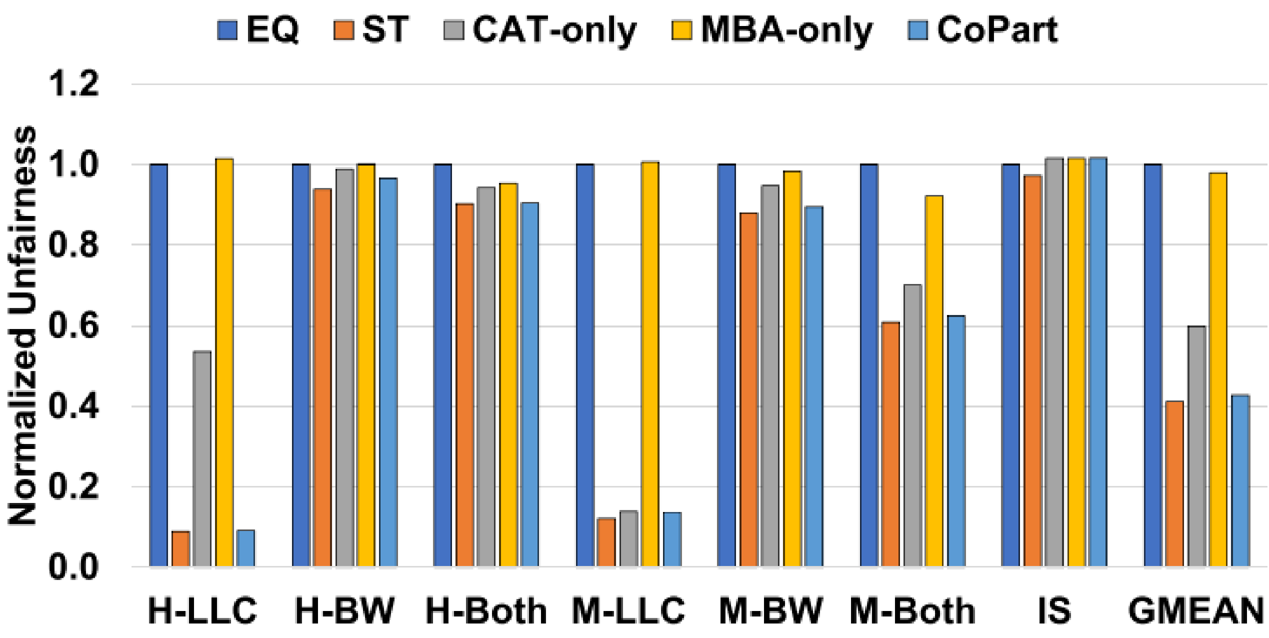
- System state space exploration phase
  - Hospitals/Residents (HR) problem
    - > *Extensively-studied and widely-applied problem in economics*
    - > *H hospitals and R medical students with preference lists*
    - > *Finds a stable match that contains no blocking pairs*

  - Resource allocation as the HR problem
    - > *Resource types (LLC, memory BW, any) -> Hospitals*
    - > *Demanding applications (consumers) -> Medical students*
    - > *Finds a stable match that contains no blocking pairs*

# Resource Manager

- System state space exploration phase
  - Step 1: Periodically collects the runtime data and updates the FSMs
  - Step 2: Determines which consumers can acquire which resources
    - *Demands a single type -> prefers that type than any type*
    - *If oversubscribed, prefers consumers with higher slowdowns*
  - Step 3: Determines which producers should supply which resources
    - *For each resource type, prefers producers with lower slowdowns*
  - Step 4: Transitions to the newly selected system state
    - *Based on the stable match derived from steps 2 and 3*

# Fairness Results



# PARTIES

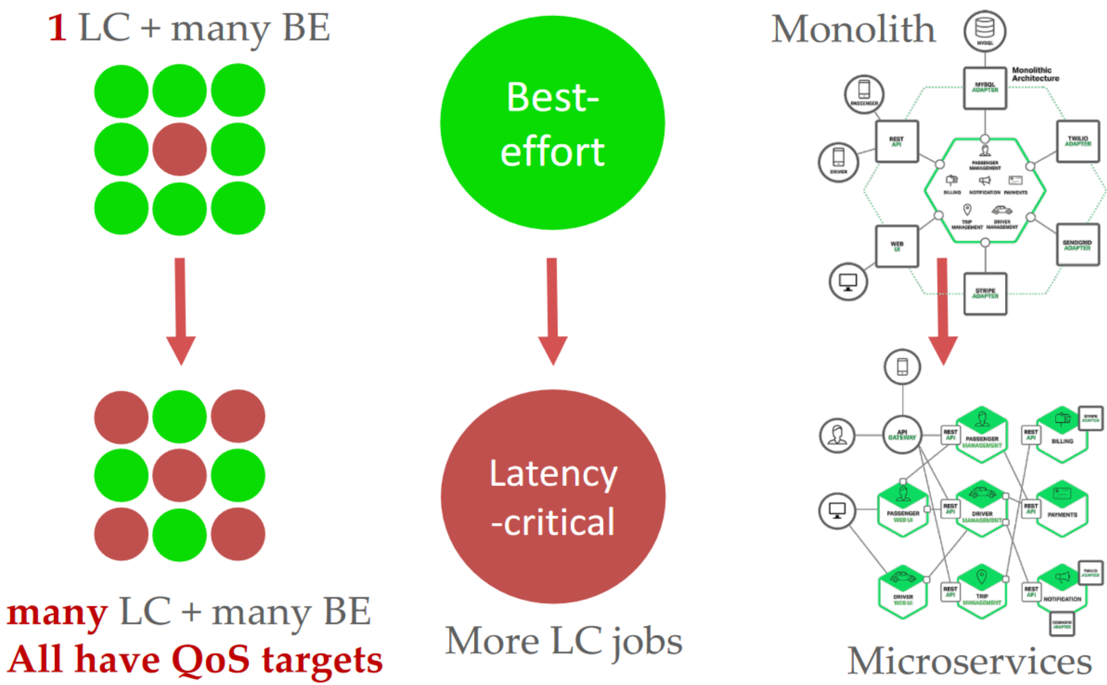


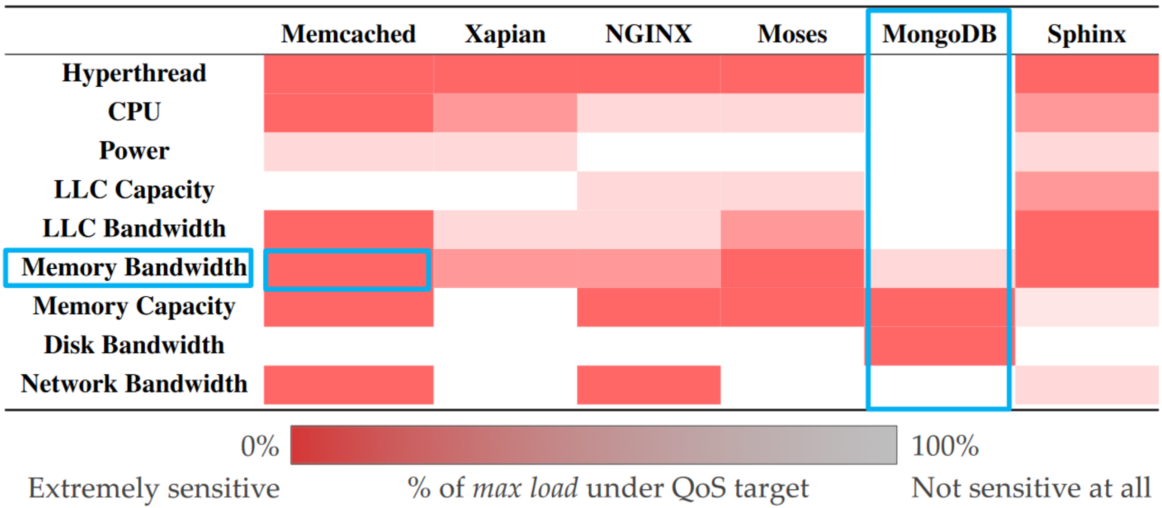

*Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf

# Main Contributions

- Workload Characterization
  - The impact of resource sharing
  - The effectiveness of resource isolation
  - Relationship between different resources

- PARTIES: First QoS-aware resource manager for colocation of many LC services
  - Dynamic partitioning of 9 shared resources
  - No a priori application knowledge
  - 61% higher throughput under QoS constraints
  - Adapts to varying load patterns

# Interference Study




- Applications are sensitive to resources with high usage
- Applications with strict QoS targets are more sensitive

**Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf*

# Isolation mechanisms

- Core mapping
  - Hyperthreads
  - Core counts
- Memory Capacity
- Disk bandwidth
- Core frequency
  - Power
- LLC capacity
  - Cache capacity
  - Cache bandwidth
  - Memory bandwidth
- Network bandwidth

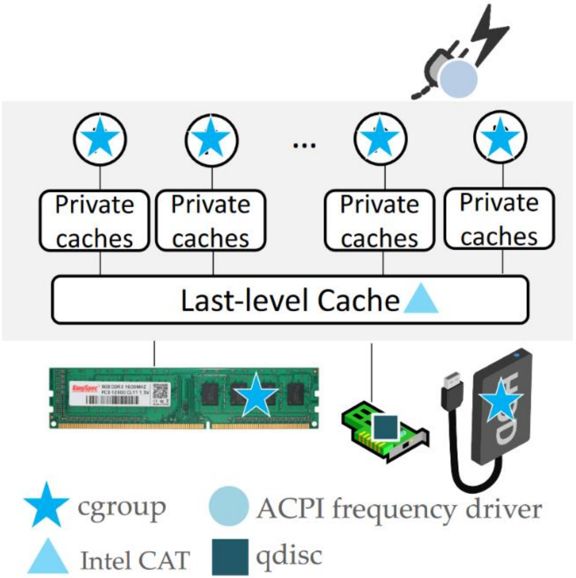
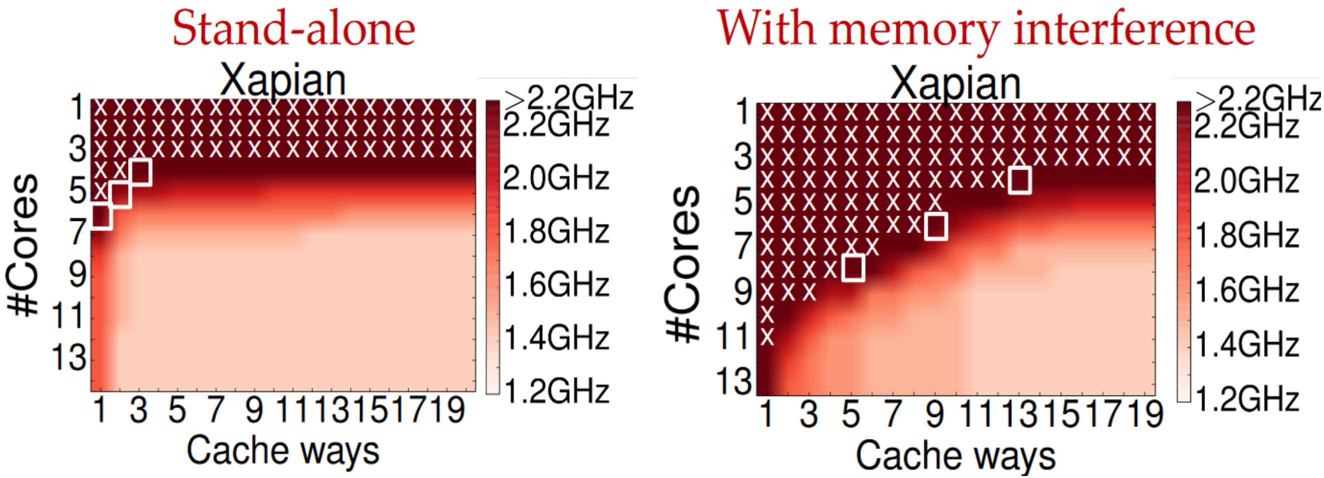



**Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf*

# Resource fungibility




- Resources are ***fungible***
  - More flexibility in resource allocation
  - Simplifies resource manager

**Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf*

# PARTIES

- 5 knobs organized into 2 wheels
- Start from a random resource
- Follow the wheels to visit all resources

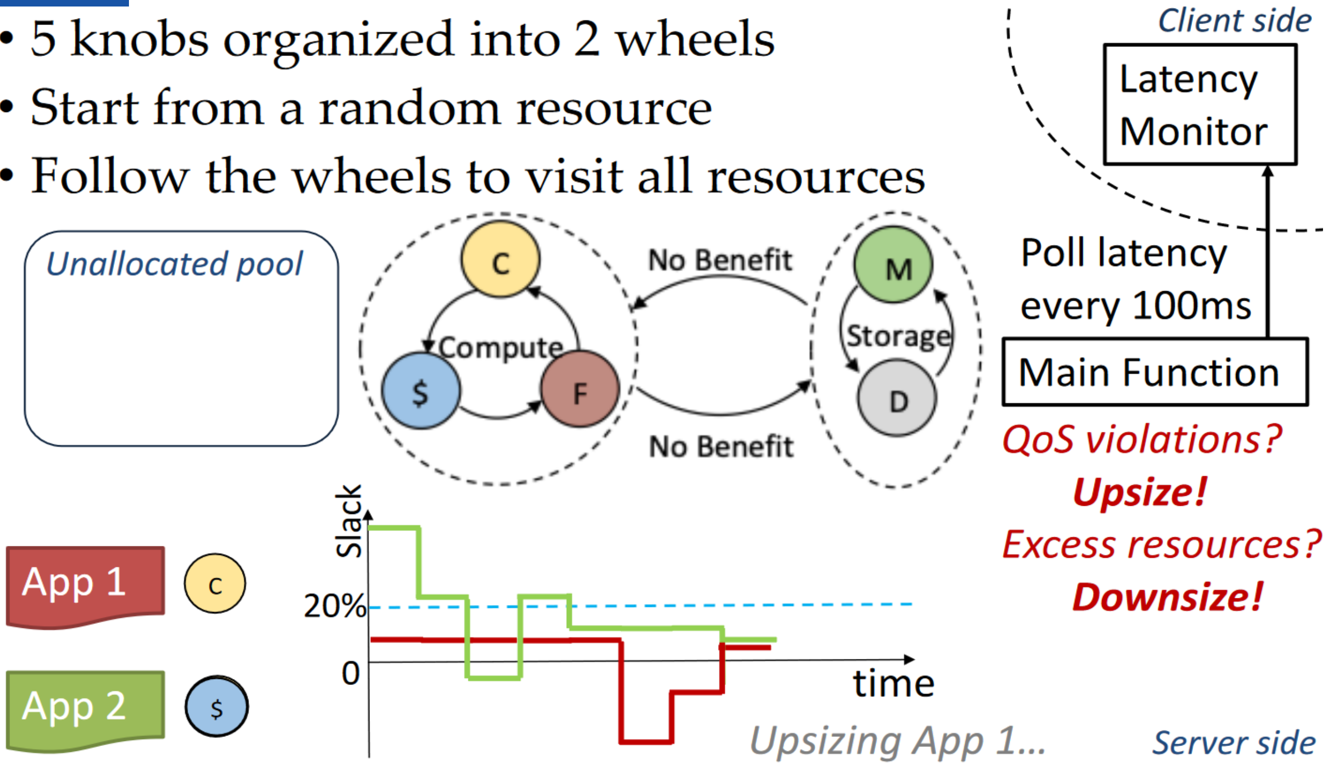



*Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf

# PARTIES RESULTS

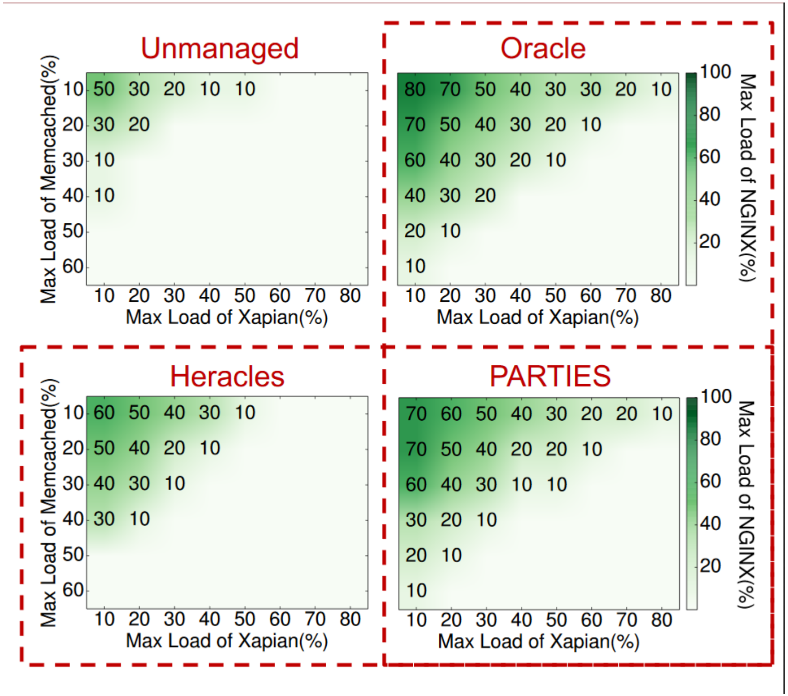
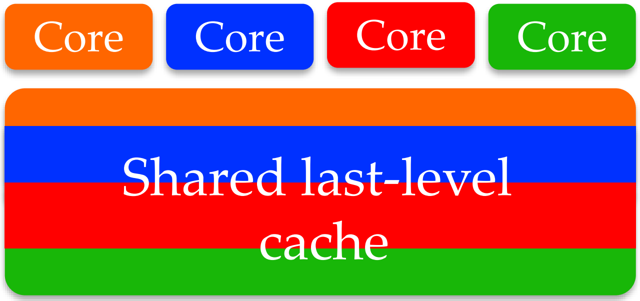



*Image source: https://sc2682cornell.github.io/ppt/PARTIES.pdf

# SWAP - BACKGOUND

- Exclusively focuses on LLC
- Cache Way Partitioning
  - Coarse-grained
    - > *16 cache ways in ThunderX 48-core processor*
- Page coloring
  - Assign different cache sets to different cores
  - Perfect isolation
  - OS-level software technique

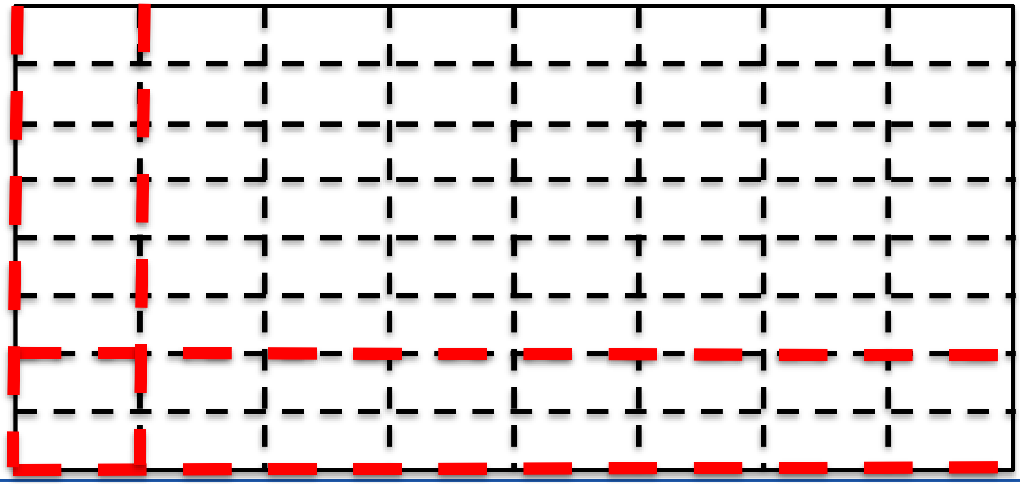


# SWAP - BACKGOUND

- Page coloring
  - High repartition overhead
  - Coarse-grained: the number of page colors is limited
    - > *4 color bits, 16 colors in ThunderX 48-core processor*

# SWAP: Set and WAy Partitioning

- Way partitioning vertically divides the cache
  - 16 cache ways in ThunderX for 48 cores
- Page coloring horizontally divides the cache
  - 16 page colors in ThunderX for 48 cores
- Combine way partitioning and page coloring



# SWAP: Set and WAy Partitioning

- Contribution
  - Combine way partitioning and page coloring that enables fine-grain cache partition in real systems

- Challenges
  - What’s the shape of the partition?
  - How are partitions placed with each other?
  - How to minimize repartition overhead?

# SWAP: Set and WAy Partitioning

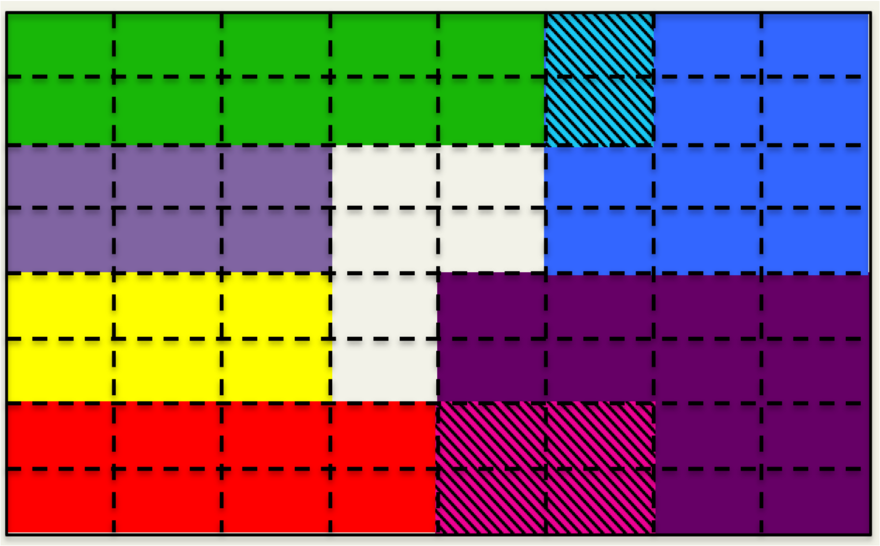
- Partition shape
  - Given the partition size, how many cache ways and pages colors should the partition have?

Partition size = 18

| | | | | |
|---|---|---|---|---|
| # cache way | 3 | 2 | 6 | 9 |
| # page color | 6 | 9 | 3 | 2 |

# SWAP: Set and WAy Partitioning

- Partition Placement
  - Partitions do not overlap (interference-free)
  - No cache space is wasted
  - Partitions cannot simply expand to occupy unused area



# SWAP: Set and WAy Partitioning

- Partition Placement
  - Given the partition size, classify the partitions into different categories
  - Page colors unchanged if the partition size stays within a certain range
  - Partitions aligned with each other

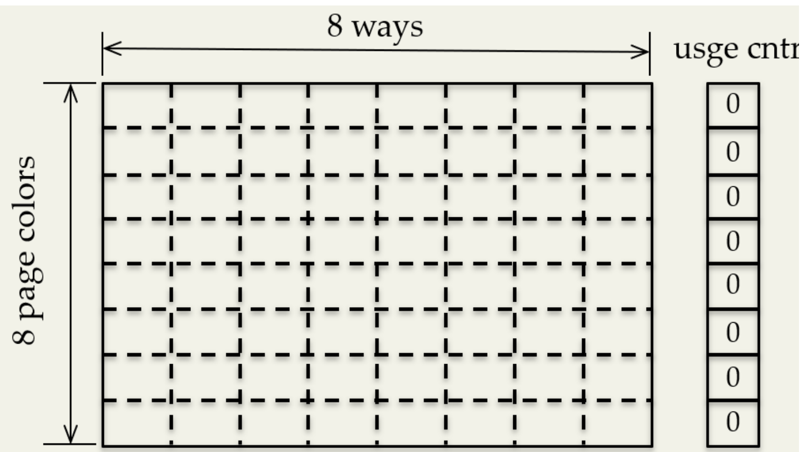
| Category | 1 | 2 | 3 | ⋯ |
|---|---|---|---|---|
| Partition size | $\geq \frac{S}{4}$ | $\frac{S}{8}$ to $\frac{S}{4}$ | $\frac{S}{16}$ to $\frac{S}{8}$ | ⋯ |
| # page color | $K$ | $\frac{K}{2}$ | $\frac{K}{4}$ | ⋯ |

Cache capacity = S, number of page colors = K

# SWAP: Set and WAy Partitioning

- Partition Placement
  - Start with large partitions (with more colors)
  - Assign the partition with page colors that have most cache ways left

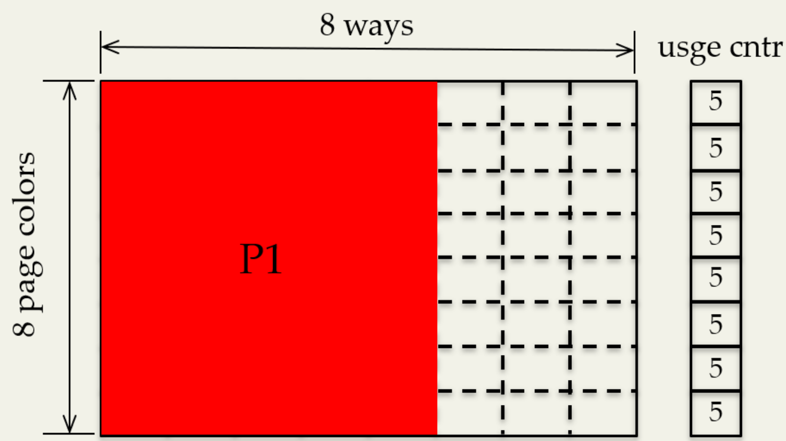
| | Size |
|---|---|
| P1 | 40 |
| P2 | 12 |
| P3 | 12 |



# SWAP: Set and WAy Partitioning

- Partition Placement
  - Start with large partitions (with more colors)
  - Assign the partition with page colors that have most cache ways left

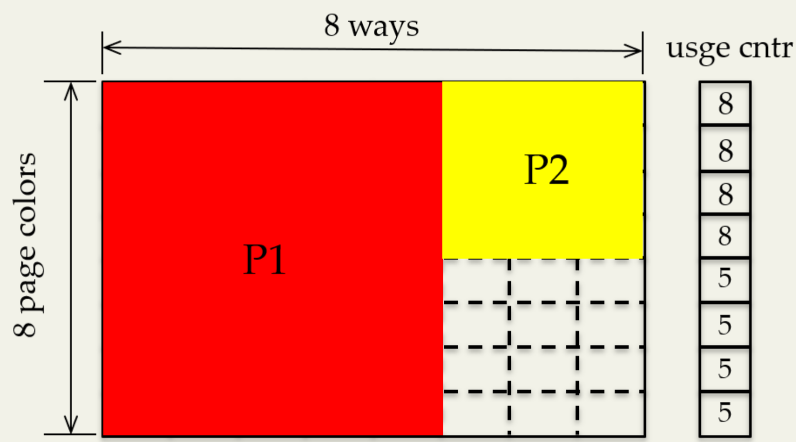
| | Size |
|---|---|
| P1 | 40 |
| P2 | 12 |
| P3 | 12 |



# SWAP: Set and WAy Partitioning

- Partition Placement
  - Start with large partitions (with more colors)
  - Assign the partition with page colors that have most cache ways left

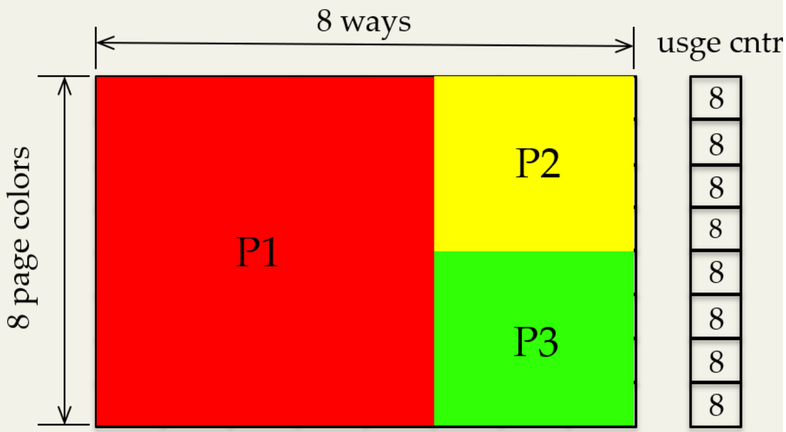
| | Size |
|---|---|
| P1 | 40 |
| P2 | 12 |
| P3 | 12 |



# SWAP: Set and WAy Partitioning

- Partition Placement
  - Start with large partitions (with more colors)
  - Assign the partition with page colors that have most cache ways left

| | Size |
|---|---|
| P1 | 40 |
| P2 | 12 |
| P3 | 12 |



# SWAP - Results