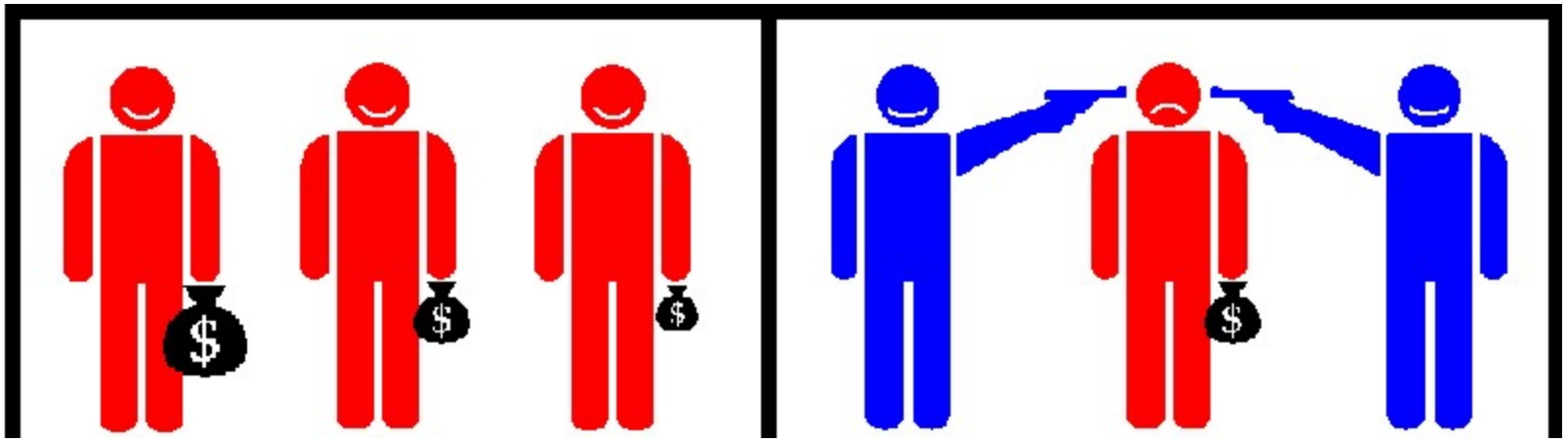
CAGE: A Contention-Aware Game-theoretic Model for Heterogeneous Resource Assignment

Diman Zad Tootaghaj **Farshid Farhat**

The Pennsylvania State University (US)

Why distributed resource management approach?



Distributed

Centralized

Distributed Resource Management

Observations:

Centralized Approaches:

As the **number of applications increase**, centralized approaches are not **scalable**.

The central decision maker does not know about the **applications' need**.

Applications have **different** resource needs.

Applications' needs **changes** during time.

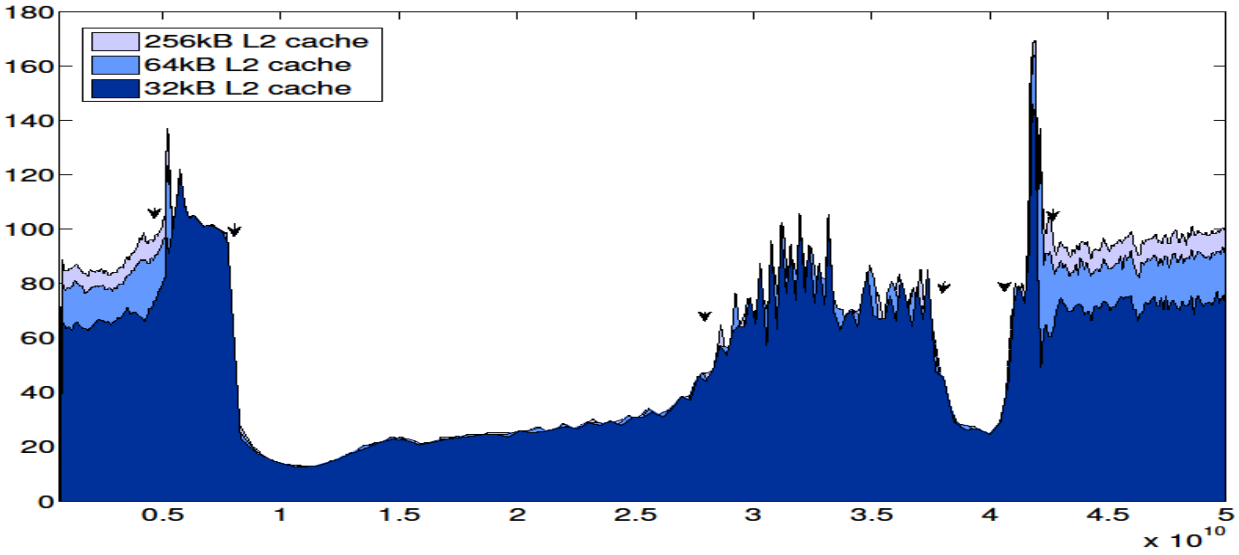
Key Idea: Using a game theoretic resource allocations, where applications compete for the shared resources.

Results:

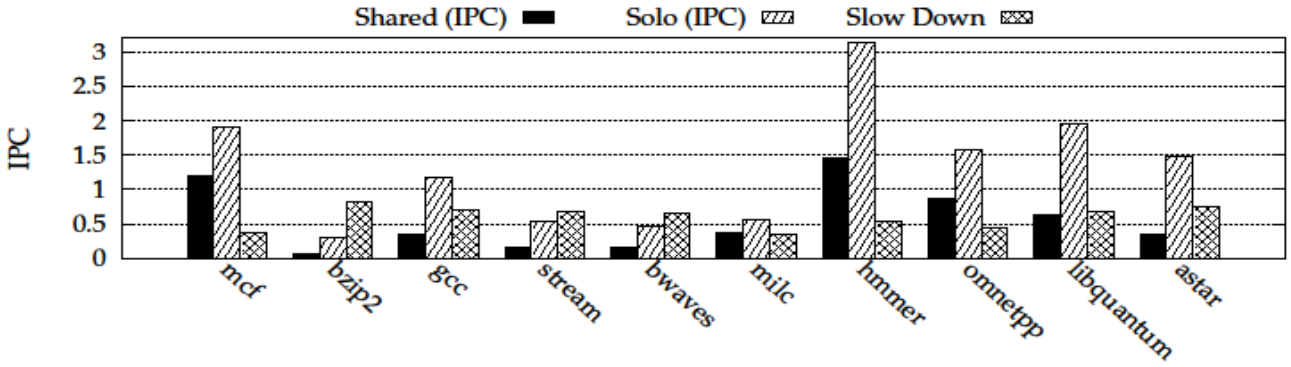
We show that **CAGE** is **strategy proof**:

- No application can get more utilization by bidding more or less than the true value of the resource

Spec 2006 Cache Demand

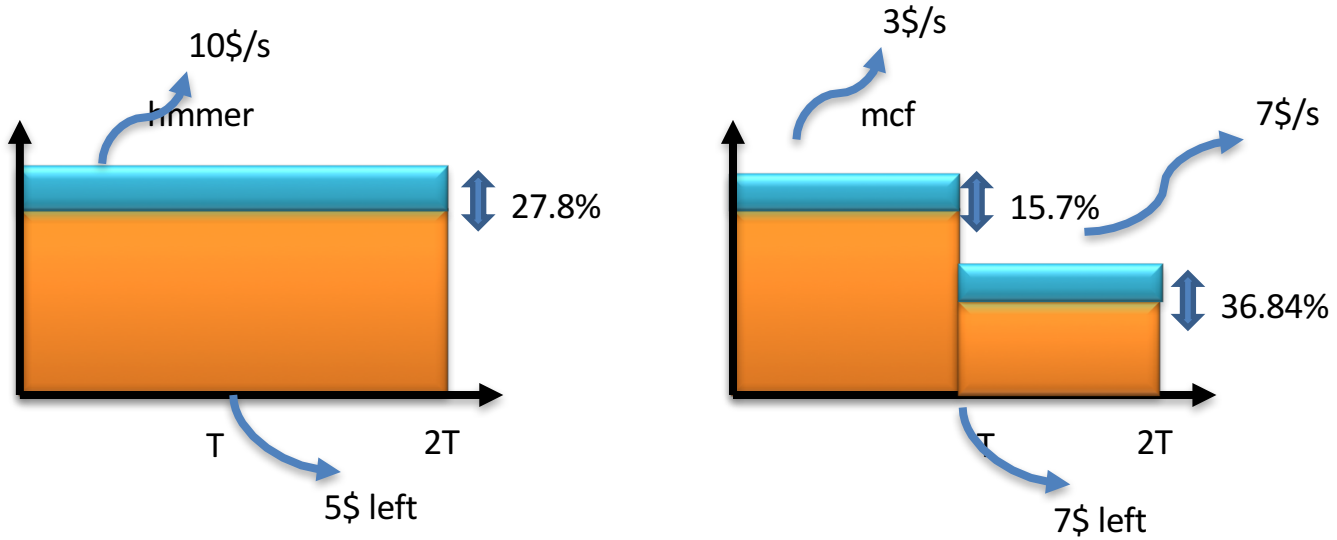


Applications' demand changes over time:



Applications' phases

Consider hmmer and mcf applications:



Utility-based [1] : 27.8% improvement.

CAGE (ours) 27.8/2+36.84/2> 27.8% → **16.2% Improvement**

[1] M. K. Qureshi et al. "Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches." *in MICRO-39.* IEEE, 2006.

Distributed Resource Management

Two Case Studies:

1) A Cache Congestion game

2) Main Processor and Co-processor Congestion game.

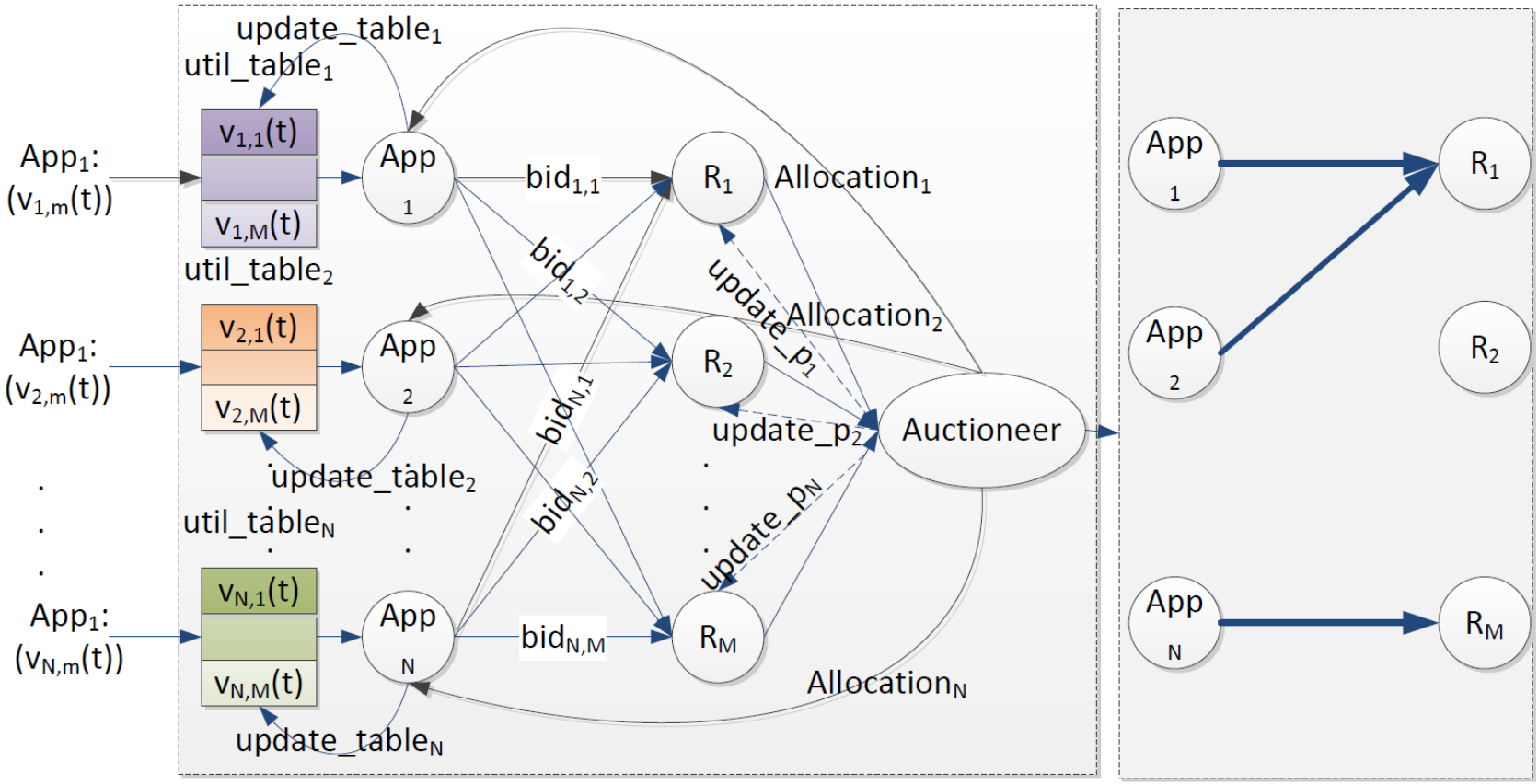
Problem Definition:

Given N **Players** (applications) and a set of M **resources**, we want to find an allocation scheme that maximizes the total utility of all players.

Game theoretic model: Each player i chooses a subset of resources (from a given family of subsets), each resource m has a utility gain u_m which depends on the number of players using this resource.

Each player wants to maximize its own utility.

CAGE Model


update_table1
util_table1
App1:
(v1,m(t))
v1,1(t)
v1,M(t)
App 1
bid1,1
R1
Allocation1
bid1,2
update_p1
util_table2
App1:
(v2,m(t))
v2,1(t)
v2,M(t)
App 2
R2
Allocation2
bidN,1
update_p2
Auctioneer
update_table2
bidN,2
update_pN
utilN
App1:
(vN,m(t))
vN,1(t)
vN,M(t)
App N
bidN,M
RM
AllocationN
updateN
App 1
R1
App 2
R2
App N
RM

Cache Congestion Game

Cache Partitioning:

We have a hierarchical cache where larger chunks are more congested and smaller chunks are more private.
The applications can decide to run their code on the congested part with more cache space or less congested part with less cache capacity.

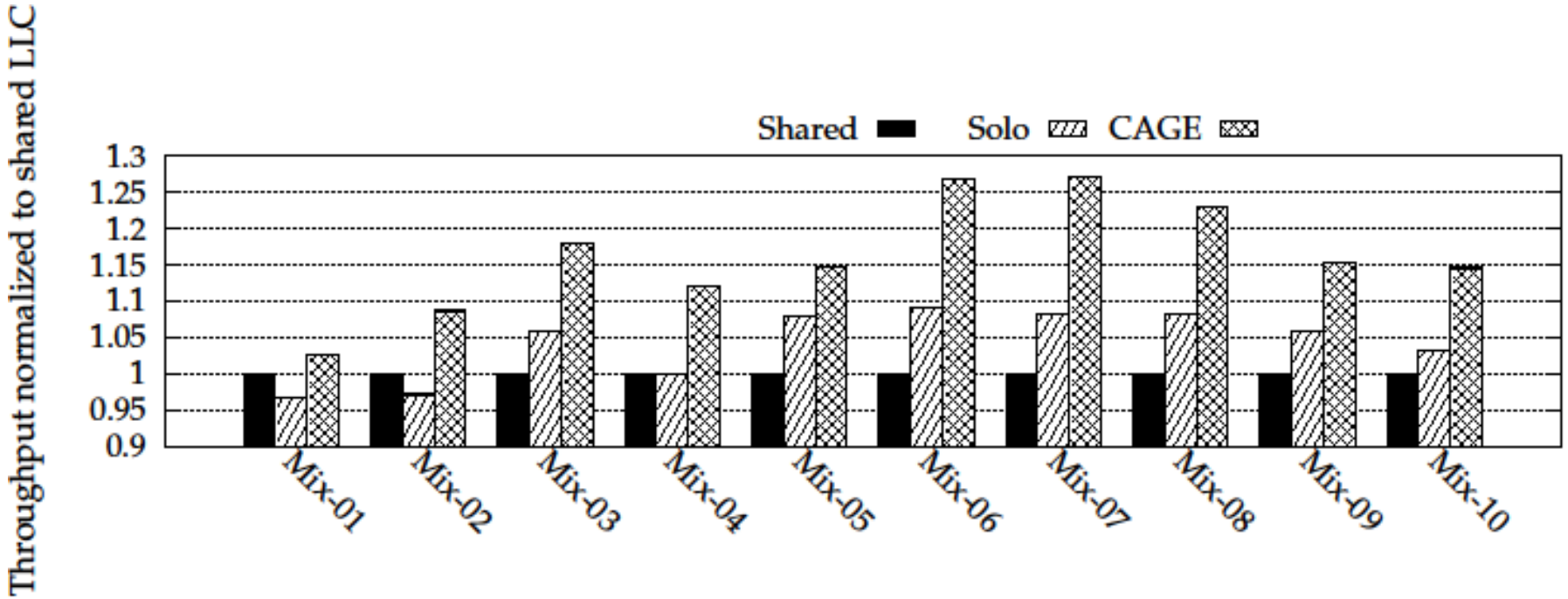
Utility-based cache partitioning:

Shows that LRU-based cache partitioning gives more cache space to applications that have higher demand and lower to those who have lower demand. But higher demand doesn't mean higher performance (streaming applications).

Evaluation

Cache Partitioning:

Comparison of CAGE with solo, shared on different mixes of applications:



Conclusion

Observations:

Centralized Approaches:

As the **number of applications increase**, centralized approaches are not **scalable**.

The central decision maker does not know about the **applications' need**.

Applications have **different** resource needs.

Applications' needs **changes** during time.

Results:

We show that **CAGE** is **strategy proof**:

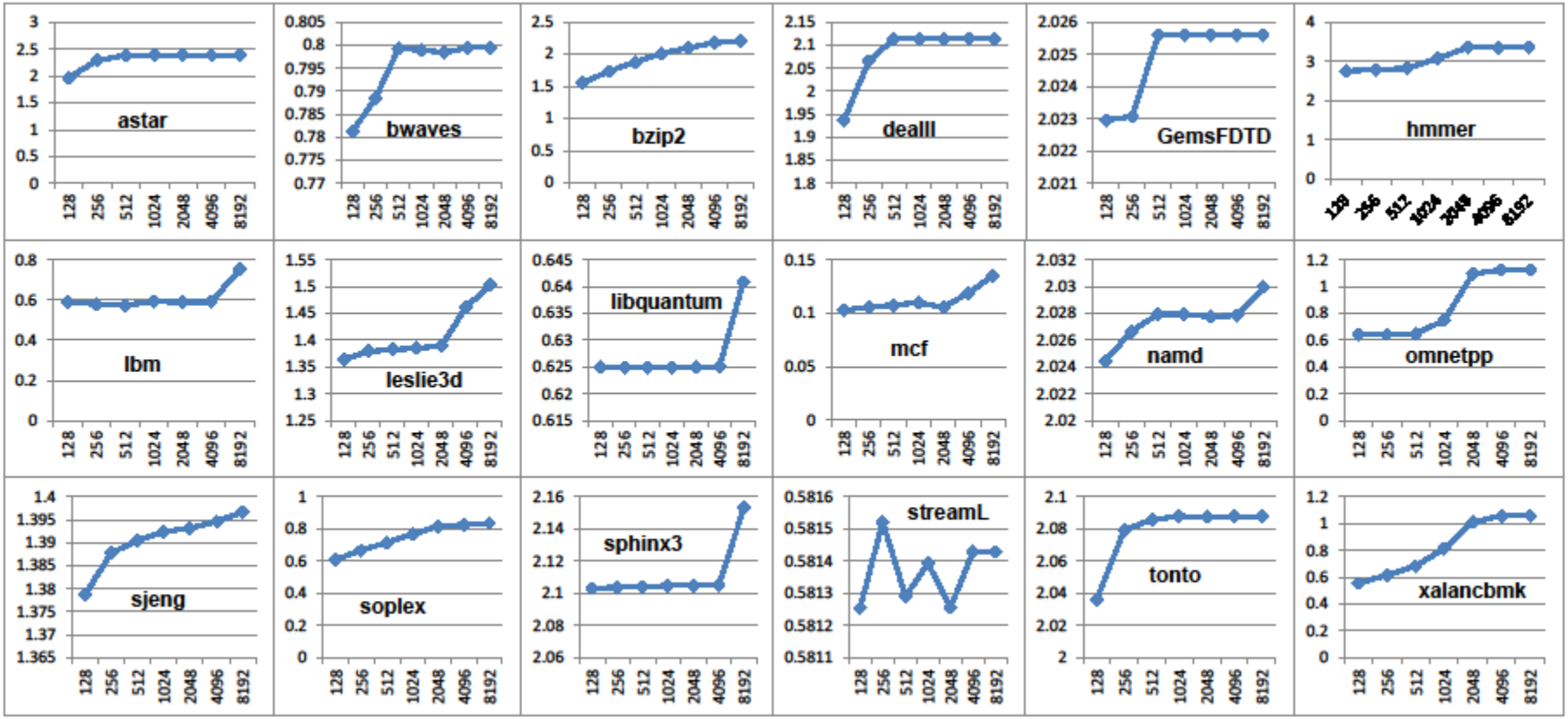
No application can get more utilization by bidding more or less than the true value of the resource

Questions?

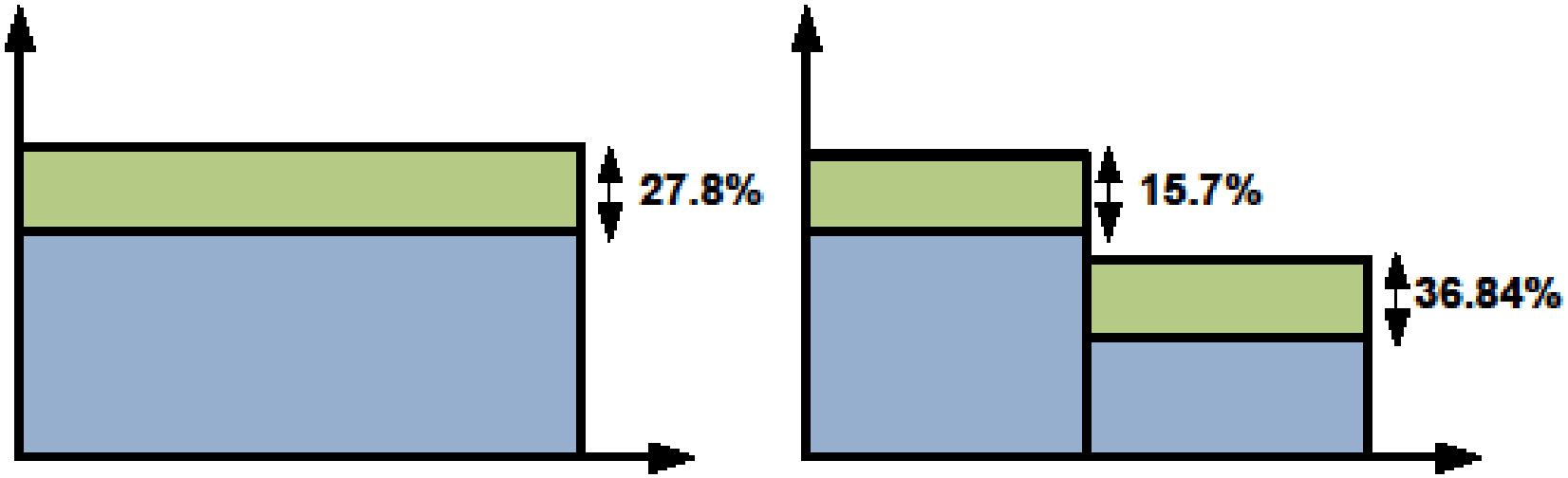
Hidden Slides

Spec 2006 Cache Demand

IPC with respect to the size of LLC:



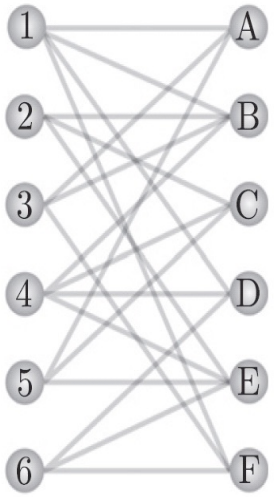
Example (hmmer, mcf)



Static approaches: 27.8% Improvement CAGE: 36.84/2+27.8/2>27.8

Parallel

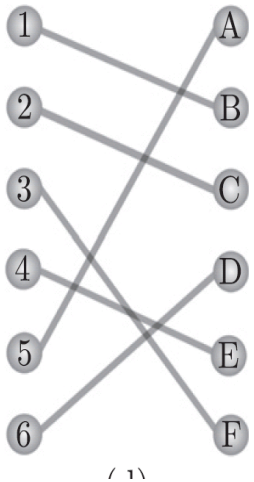
- Consider the following bipartite graph and utility functions for each resource:



$$\begin{bmatrix} 9 & 6 & 0 & 3 & 0 & 2 \\ 0 & 2 & 7 & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & 3 \\ 0 & 6 & 8 & 3 & 4 & 0 \\ 8 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 & 6 & 5 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 9 & \mathbf{6} & 0 & 3 & 0 & 2 \\ 0 & 2 & \mathbf{7} & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & \mathbf{3} \\ 0 & 6 & 8 & 3 & \mathbf{4} & 0 \\ \mathbf{8} & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{7} & 6 & 5 \end{bmatrix}$$



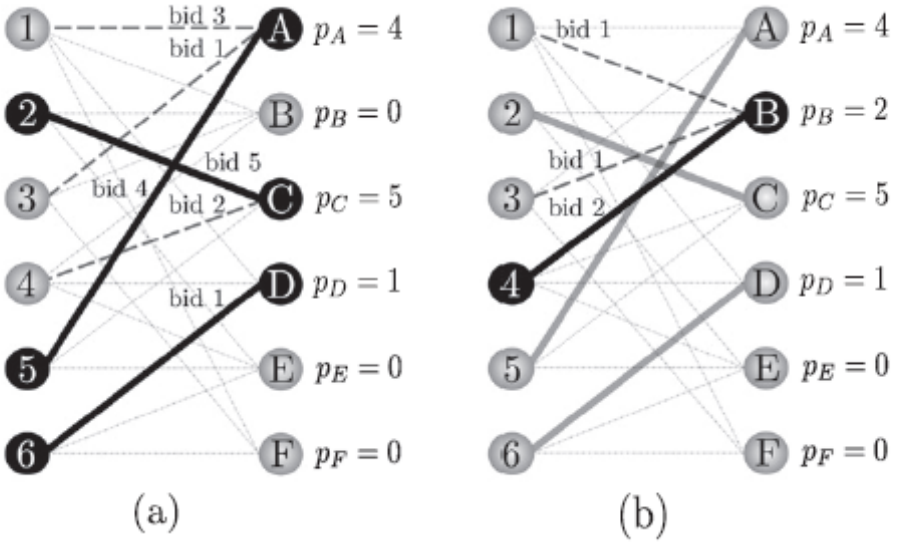
Sequential Auction:

- Parallel actions of each agent: Each user bids for the most valuable resource.

$$\begin{bmatrix} 9 & 6 & 0 & 3 & 0 & 2 \\ 0 & 2 & 7 & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & 3 \\ 0 & 6 & 8 & 3 & 4 & 0 \\ 8 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 & 6 & 5 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 9 & \mathbf{6} & 0 & 3 & 0 & 2 \\ 0 & 2 & \mathbf{7} & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & \mathbf{3} \\ 0 & 6 & 8 & 3 & \mathbf{4} & 0 \\ \mathbf{8} & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{7} & 6 & 5 \end{bmatrix}$$



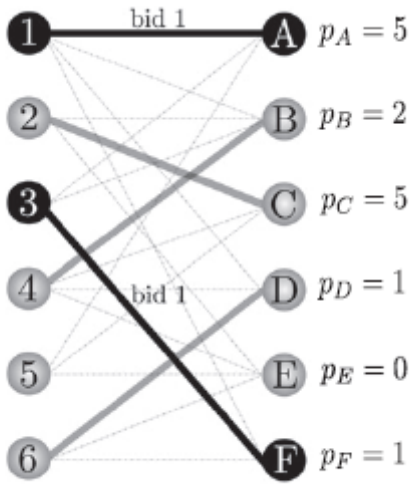
Sequential Auction:

- Parallel actions of each agent:

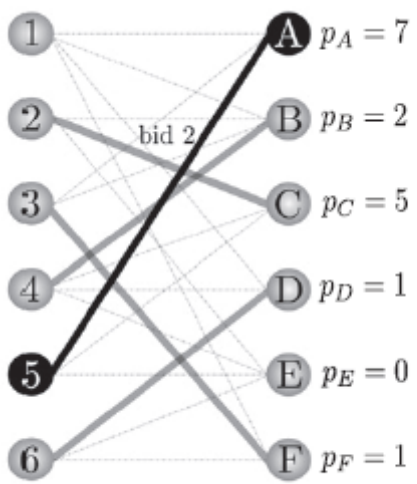
$$\begin{bmatrix} 9 & 6 & 0 & 3 & 0 & 2 \\ 0 & 2 & 7 & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & 3 \\ 0 & 6 & 8 & 3 & 4 & 0 \\ 8 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 & 6 & 5 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 9 & \mathbf{6} & 0 & 3 & 0 & 2 \\ 0 & 2 & \mathbf{7} & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & \mathbf{3} \\ 0 & 6 & 8 & 3 & \mathbf{4} & 0 \\ \mathbf{8} & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{7} & 6 & 5 \end{bmatrix}$$



(c)



(d)

Sequential Auction:

- Parallel actions of each agent:

$$\begin{bmatrix} 9 & 6 & 0 & 3 & 0 & 2 \\ 0 & 2 & 7 & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & 3 \\ 0 & 6 & 8 & 3 & 4 & 0 \\ 8 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & 7 & 6 & 5 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 9 & \mathbf{6} & 0 & 3 & 0 & 2 \\ 0 & 2 & \mathbf{7} & 0 & 1 & 0 \\ 5 & 4 & 0 & 0 & 0 & \mathbf{3} \\ 0 & 6 & 8 & 3 & \mathbf{4} & 0 \\ \mathbf{8} & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & 0 & \mathbf{7} & 6 & 5 \end{bmatrix}$$