# Rendering Server Allocation for MMORPG Players in Cloud Gaming

Iryanto Jaya (Nanyang Technological University)

Wentong Cai (Nanyang Technological University)

Yusen Li (Nankai University)

# Agenda

- Background
- Problem Definition
- Proposed Solutions
- Experiments
- Conclusions and Future Works

# Motivations

- Multiplayer cloud gaming
- Reducing the cost for cloud gaming service providers

# Executive Summary

**Problem**
Allocating players to rendering servers (RSes)

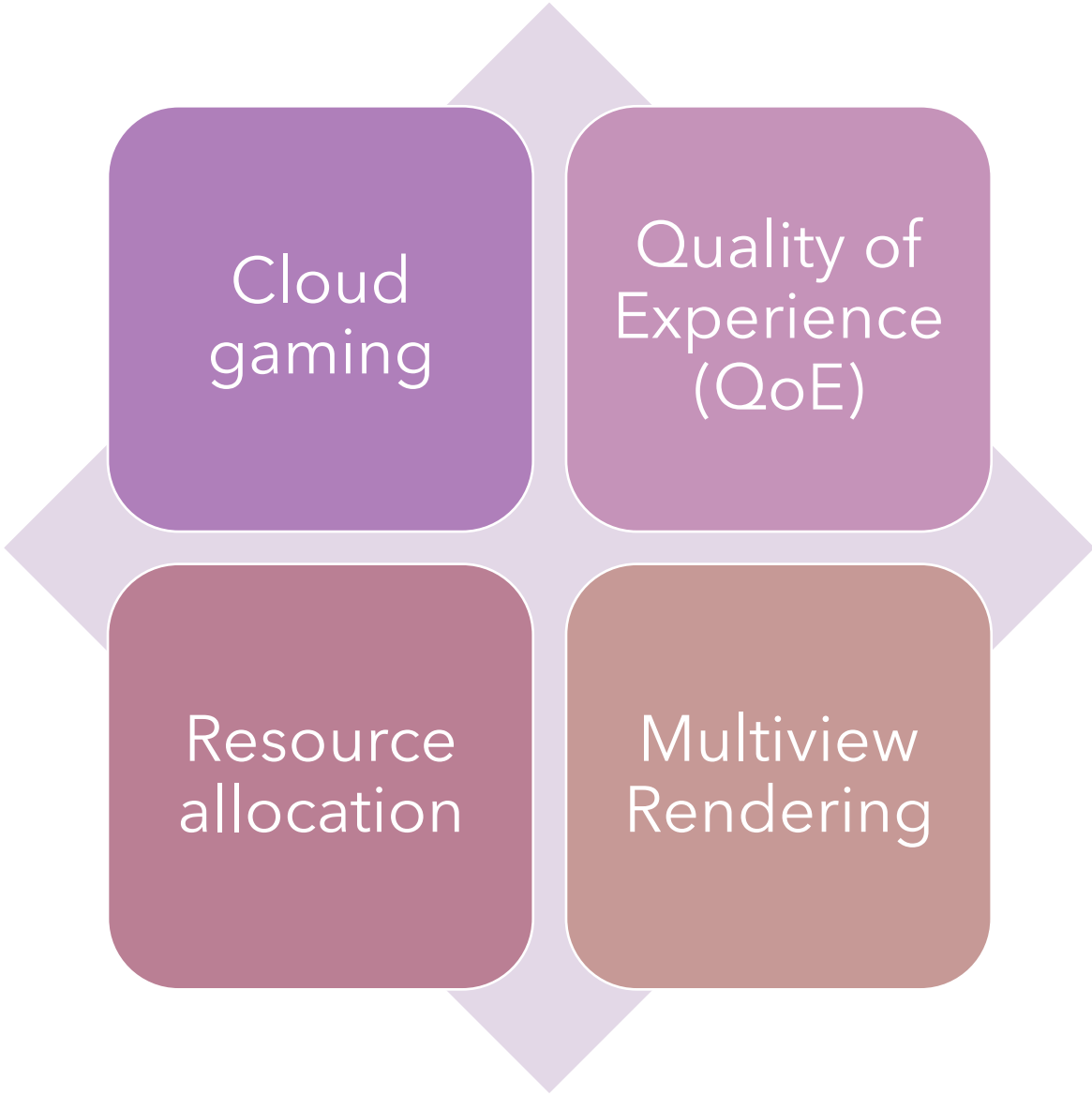
**Our goal**
Minimize the cost of using RSes

**Observation**
The RS resource capacity is the most limiting factor in the allocation

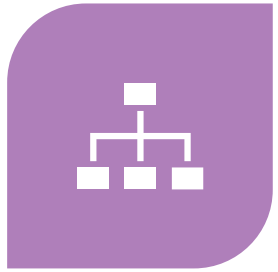
**Key idea**
Use rendering workload sharing to reduce resource usage

**Results**
Workload sharing reduces cost of RSes

# Related Works



# Resource Allocation



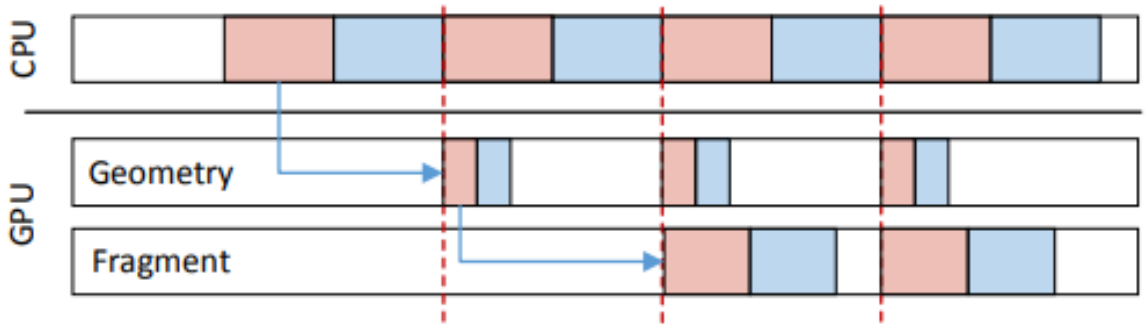
RS AND PLAYER ALLOCATIONS

NP-HARD

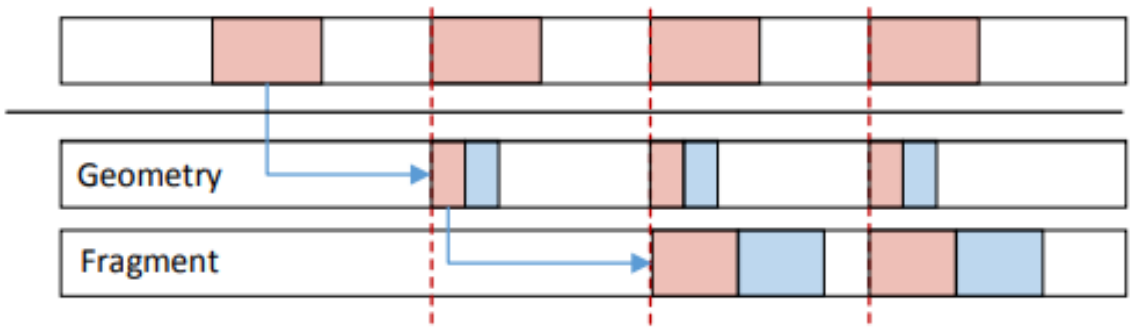
MULTIPLE TIME INSTANCES

OFFLINE VS. ONLINE PROBLEM

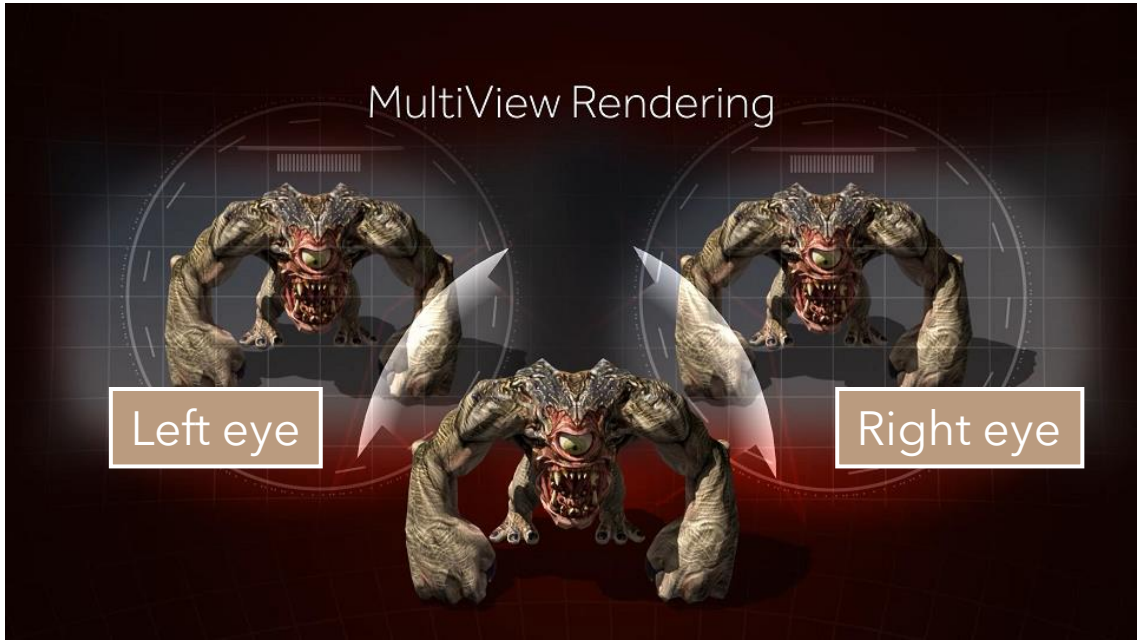
# Multiview Rendering




(a)




(b)



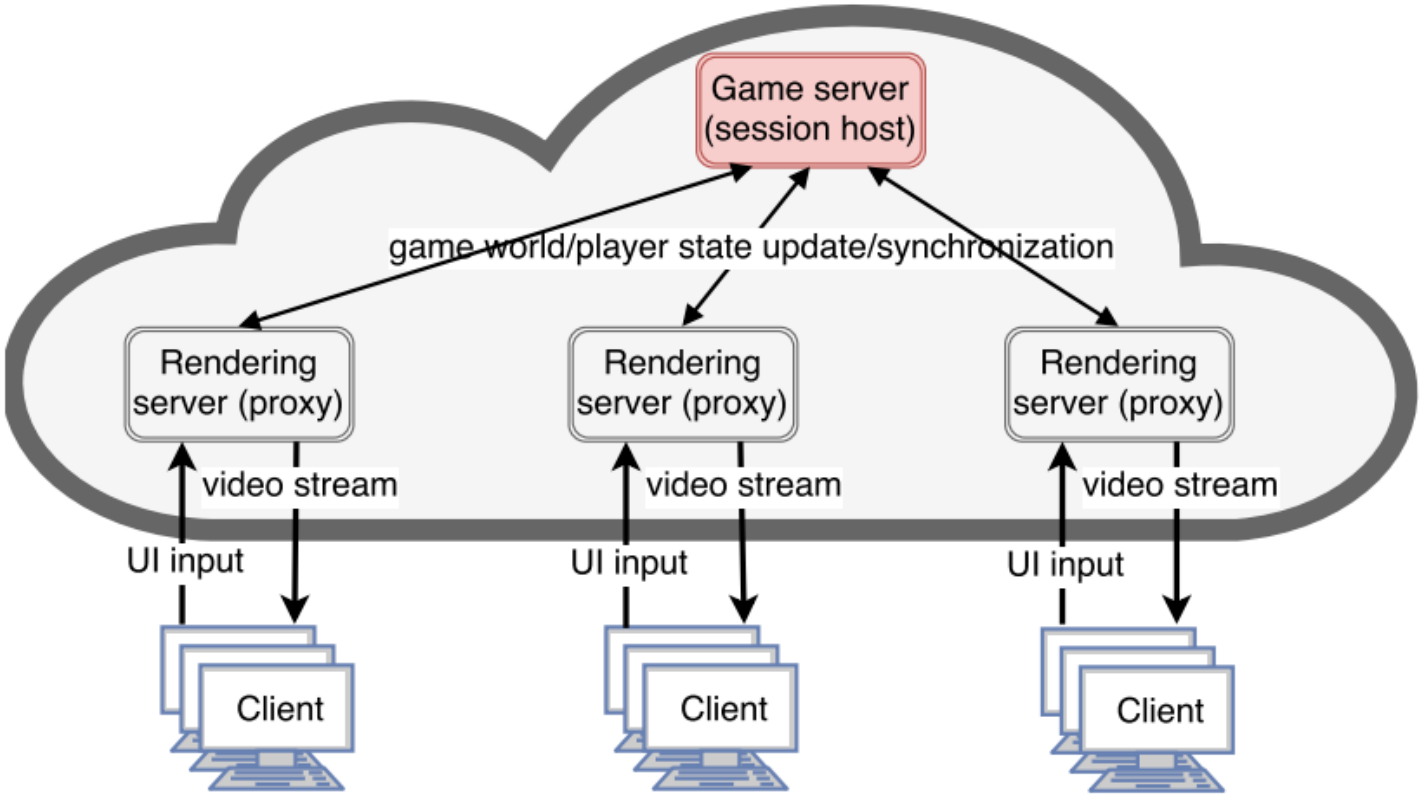
# Challenges & Contributions

Multiview rendering in cloud gaming

Dependency between players allocated to one RS

Optimization over multiple time instances

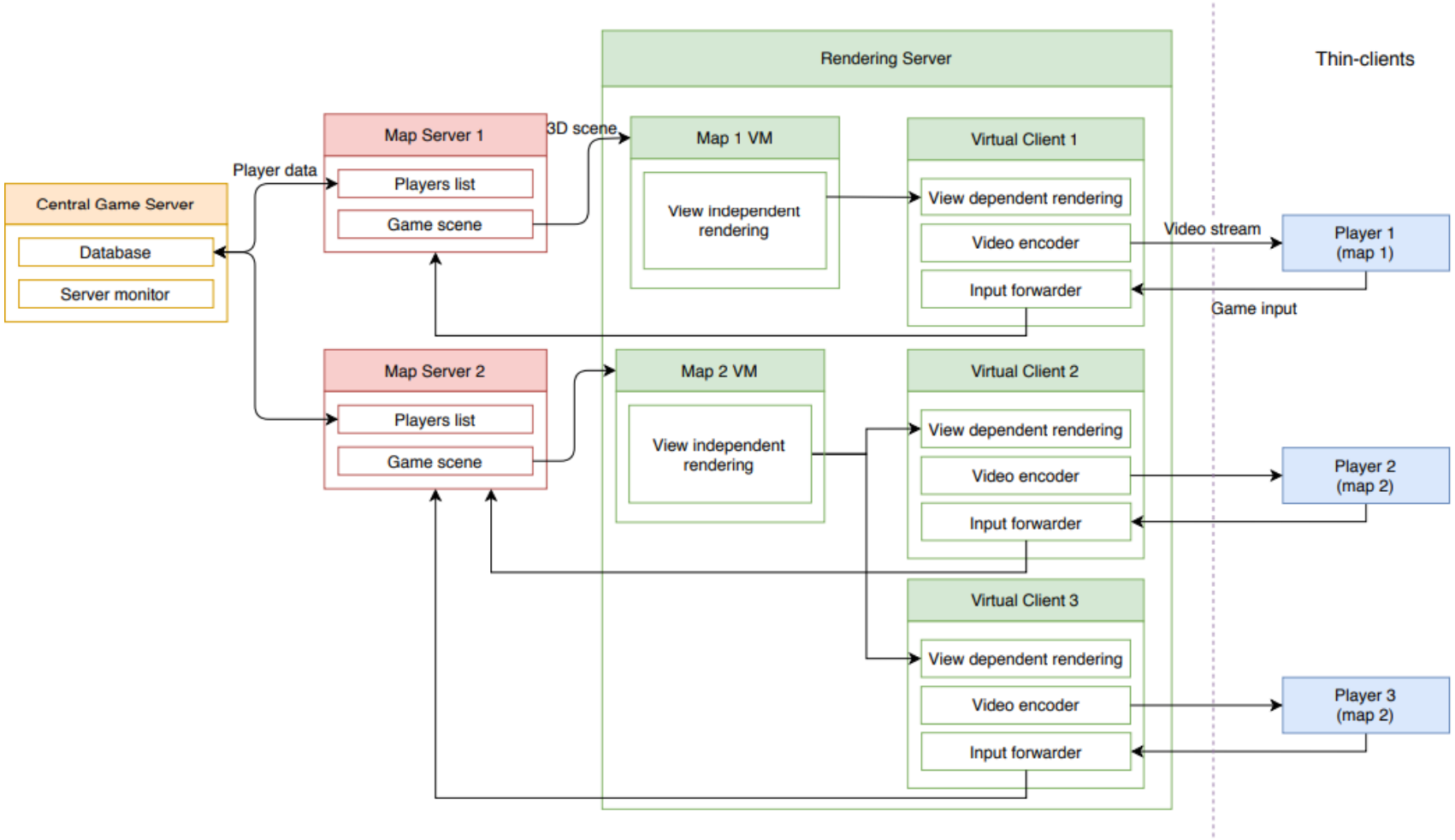
# Conventional Cloud Gaming Architecture



# Key Rationale for Architecture Design

- Make use of common information from players in the same virtual map
- Split the rendering process into two parts: view dependent and view independent
- The game server consists of a central game server to maintain non-visual information (database, login information, etc.) while map servers maintain the game scenes

# Proposed Cloud Gaming Architecture



# Problem Definition

Optimization problem

Objective:

- Minimize server cost

Constraints:

- Server capacity
- Latency

# Detailed Problem Formulation

**Objective** Minimize $\sum_t \sum_r z_r^t \text{Cost}_r$

**Constraints** Subject to:

**Assignment** $\forall t, \forall p \in I_t, \quad \sum_r x_{p,r} = 1$

**CPU capacity** $\forall t, \forall r, \quad \sum_{p \in I_t} x_{p,r} c_{\Gamma_p} + \sum_m y_{r,m}^t c'_m \leq C_r$

**GPU capacity** $\forall t, \forall r, \quad \sum_{p \in I_t} x_{p,r} g_{\Gamma_p} \leq G_r$

**Latency** $\forall t, \forall p \in I_t, \quad \sum_r x_{p,r} \left( l_{p,r} + l_{r,\Gamma_p} \right) \leq L$

| | |
|---|---|
| $P$ | Set of players $\{p_1, p_2, ..., p_n\}$ |
| $M$ | Set of maps $\{m_1, m_2, ..., m_k\}$ |
| $R$ | Set of RS $\{r_1, r_2, ..., r_s\}$ |
| $\Gamma_p$ | Virtual map where player $p$ is located in |
| $\{t_0, t_1, ..., t_T\}$ | Time instances |
| $I_t$ | Set of players in the system at time $t$ |
| $x_{p,r}$ | Binary variable indicating whether player $p$ is assigned to RS $r$ |
| $y_{r,m}^t$ | Binary variable indicating whether there is a player in map $m$ which is assigned to server $r$ at time $t$ |
| $z_r^t$ | Binary variable indicating whether server $r$ is used at time $t$ |

# Challenges

- Trade off between constraints
- Resource allocation is NP-hard
- Cannot derive a simple algorithm from the problem formulation

# Online Heuristics

Obtain the list of eligible RSes from currently active RSes, if there is none, obtain the list from inactive-RSes

- Lowest price (LP)
  Select the lowest priced RS

- Lowest waste resource (LWR)
  Waste resource = Capacity – current workload
  Best fit

- Highest workload share (HWS)
  Prioritize possible workload sharing, then use LP to break ties

- Lowest waste price (LWP)
  Waste price = Waste resource / RS cost

# Offline Algorithms

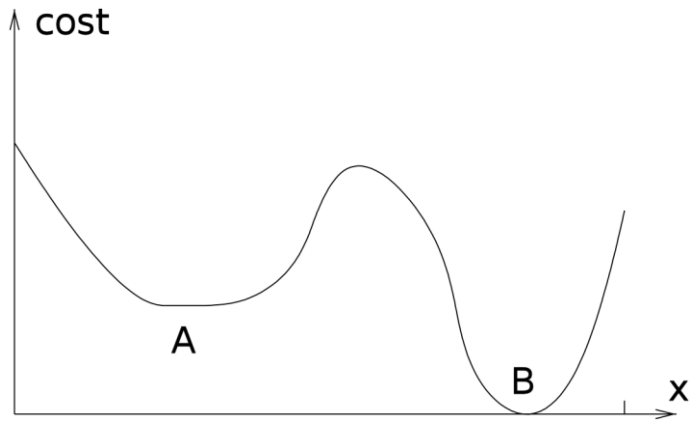
**LOCAL SEARCH (LS)**

GET AN INITIAL SOLUTION, THEN USE LOCAL SEARCH TO OPTIMIZE THE COST

**LOWER BOUND (LB)**

AN OPTIMAL SOLUTION DERIVED USING A MATHEMATICAL SOLVER

# Local Search Algorithm




Aim: to empty RSes with low utilization

1. Gets the first solution
2. Sort the RSes with increasing number of players
3. Move each player from lower index RS to higher index RS if possible
4. Stop when there is no possible move

# Experiments

- 500+ PlanetLab player nodes
- Amazon EC2 & Microsoft Azure to host MSes and RSes
- Poisson distribution player arrival
- Exponential distribution playing duration

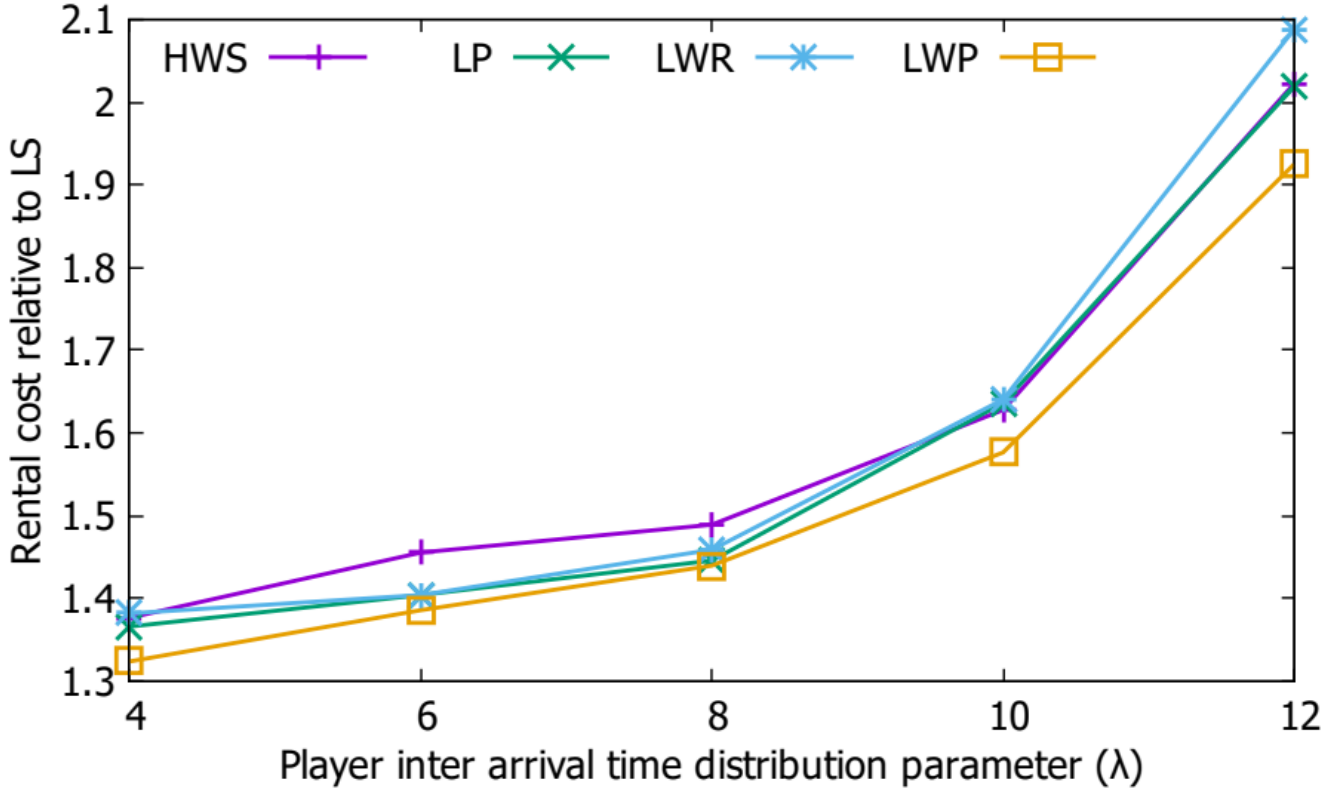
Assumptions:

- The number of servers, maps and players are fixed
- The latency between involved nodes never change
- Each player will be allocated to an RS (no rejection)

# Default Experiment Parameters

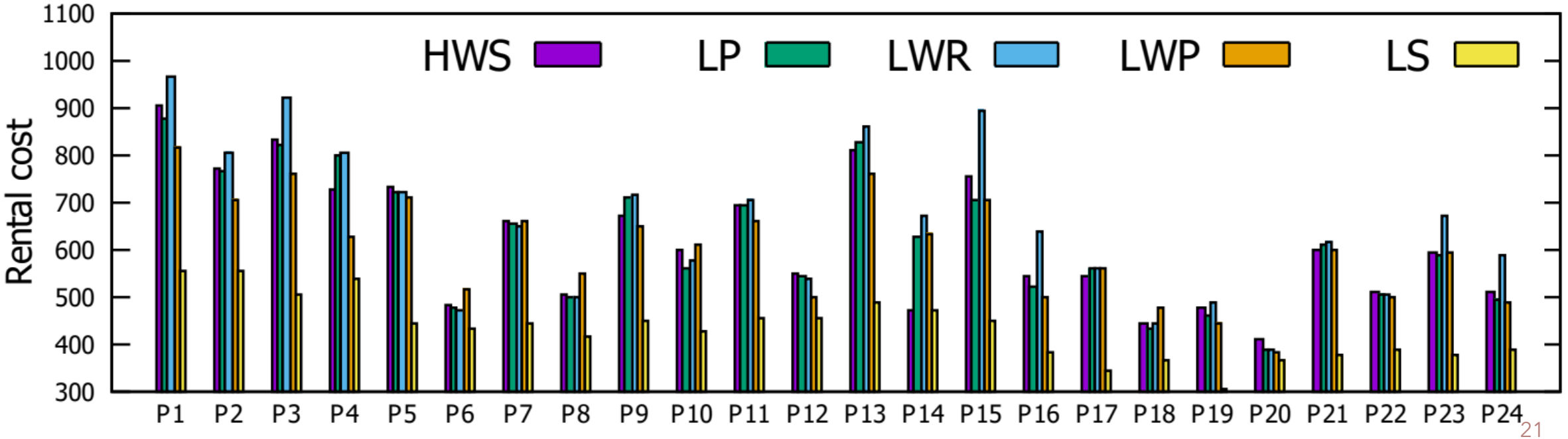
| | |
|---|---|
| Latency bound | 100 ms |
| Player inter arrival time distribution parameter ($\lambda$) | 6 |
| Player maximum inter arrival time | 10 hours |
| Playing duration distribution parameter ($\lambda'$) | 2.5 |
| Maximum playing duration | 5 hours |
| Number of available RSes | 20 |
| RS CPU capacity | 8 to 10 units |
| RS GPU capacity | 10 units |
| CPU view dependent workload | 1 to 3.4 units |
| CPU view independent workload | 1 unit |
| GPU workload | 1 unit |

# Online Heuristics Performance

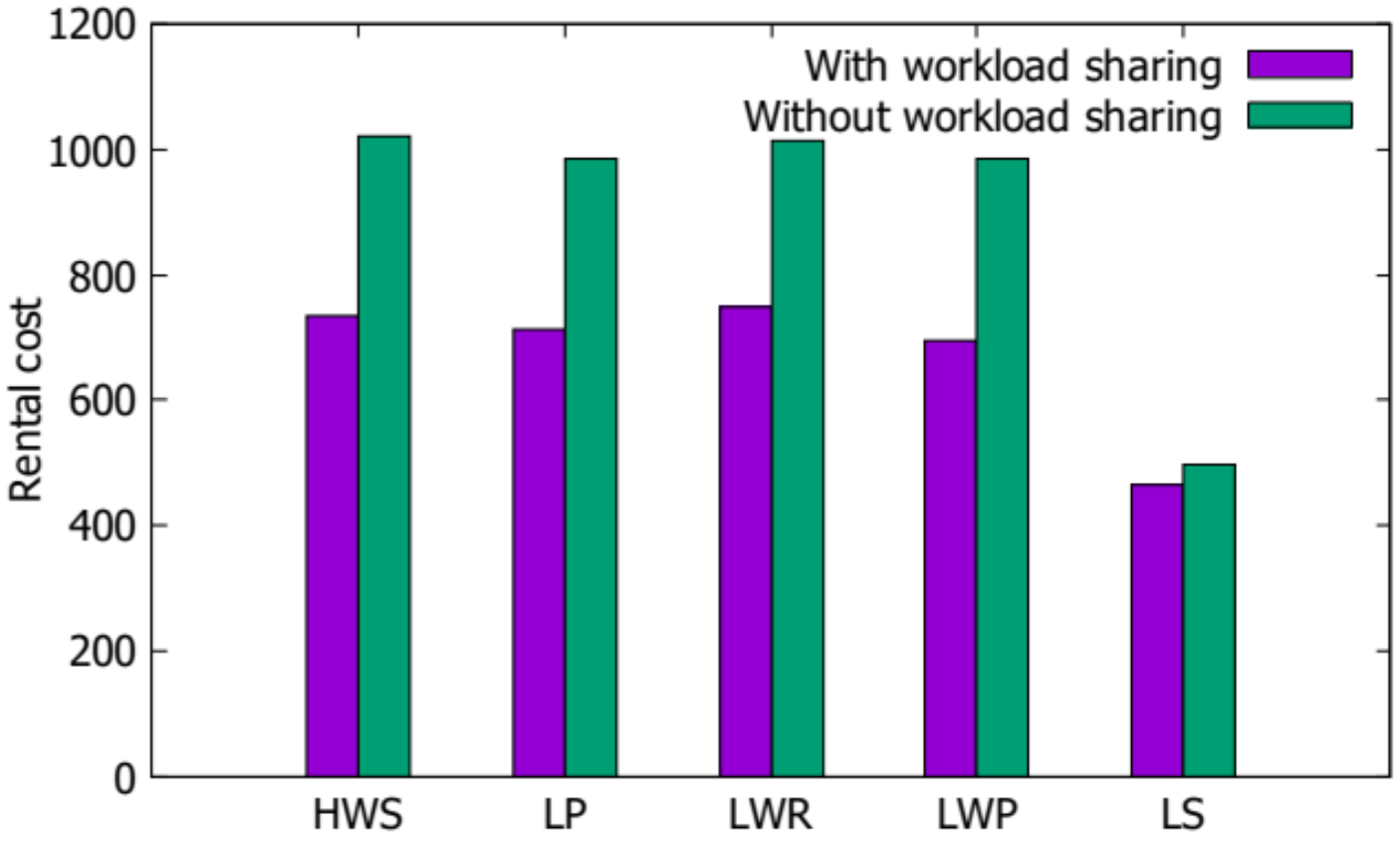


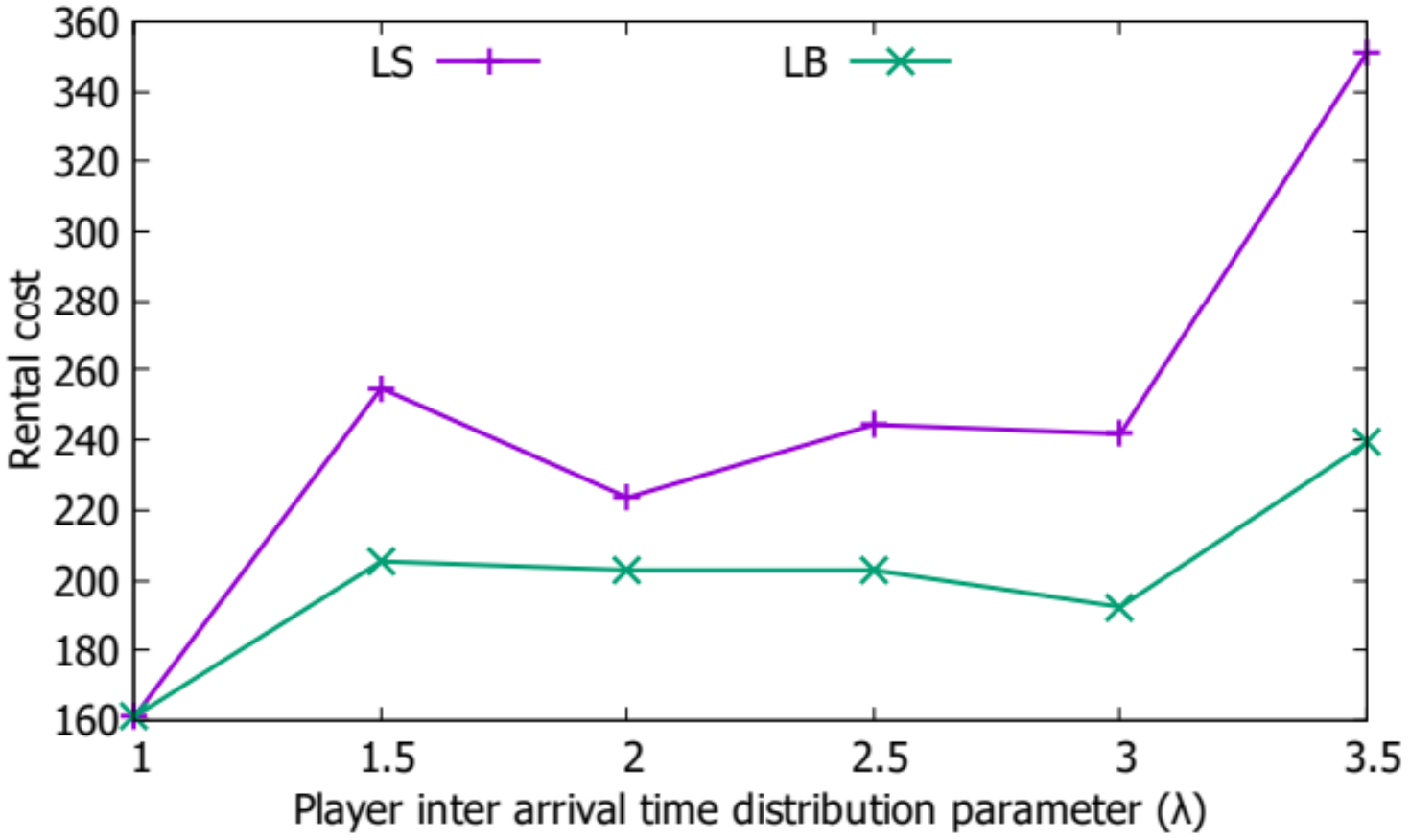
# Online Heuristics Performance

| Parameter | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | P12 | P13 | P14 | P15 | P16 | P17 | P18 | P19 | P20 | P21 | P22 | P23 | P24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RS Pricing | U | U | U | U | U | U | U | U | U | U | U | U | V | V | V | V | V | V | V | V | V | V | V | V |
| Dominant Res. | C | C | C | C | G | G | G | G | H | H | H | H | C | C | C | C | G | G | G | G | H | H | H | H |
| Latency Bound | R | R | S | S | R | R | S | S | R | R | S | S | R | R | S | S | R | R | S | S | R | R | S | S |
| $c'$ Level | L | H | L | H | L | H | L | H | L | H | L | H | L | H | L | H | L | H | L | H | L | H | L | H |



# Comparison with Traditional Cloud Gaming






# Offline Algorithms Comparison

# Conclusions and Future Works

Conclusions:

- MMORPG cloud gaming architecture with multiview rendering
- Rendering workload sharing reduces overall cost
- Increasing player arrival frequency widens the gap between the costs from online and offline approaches

Future works:

- Player rejections
- Edge server involvement
- Future request predictions

# Q&A