PLACING VIRTUAL MACHINES TO OPTIMIZE CLOUD GAMING EXPERIENCE

HUA-JUN, HONG

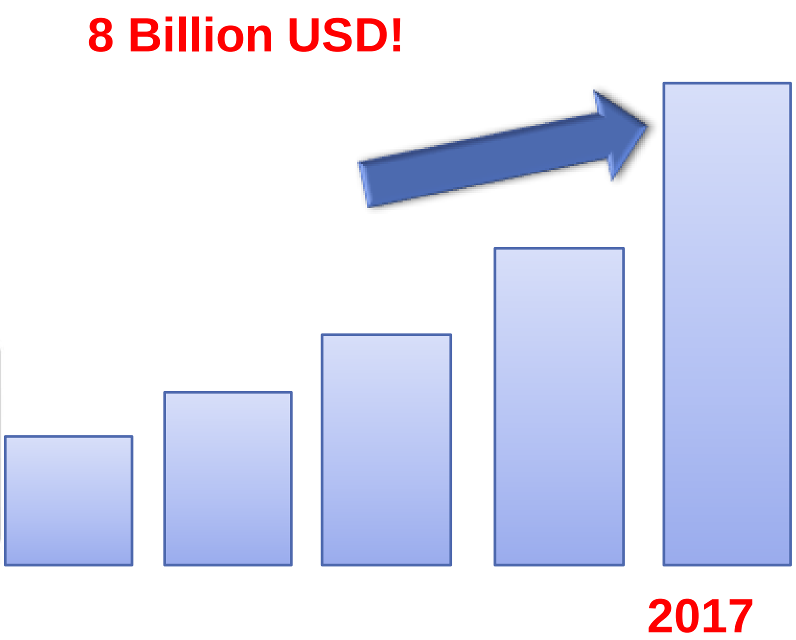
JUNE, 2014

OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

THE NEW APPROACH TO PLAY GAMES

- **Cloud gaming system**
  - Play any game at anytime, anywhere on any device!
  - The increasing market!



FINANCIAL DIFFICULTY OF PROVIDERS

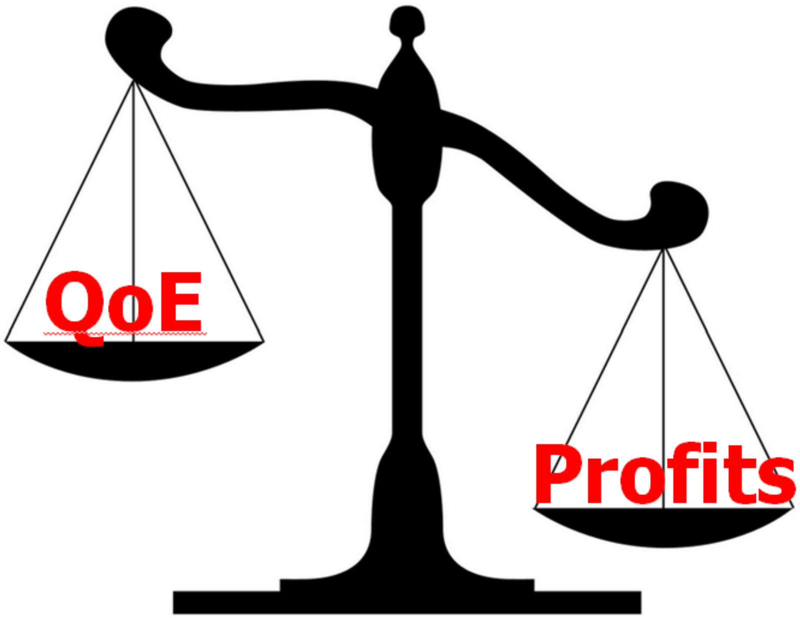
- **Cloud gaming providers**
  - Gaikai
  - Ubitus
  - Onlive – runs into the financial difficulty
  - …
- **The financial problem may cause by…**
  - Network latency
  - Resource allocation of each VM
  - Non-mature GPU virtualization
  - …

PROBLEM STATEMENT

- **Diverse gaming hardware requirements may lead to wasted hardware resources**
- **Consolidating different games results in different profits and gaming quality**
- **Hence, we propose a VM placement policy to maximize the profits while achieve just-good-enough QoE**
- **Also, we conduct a measurement study to make sure that if the modern GPU is powerful enough for the cloud gaming system**

GOALS

- Find the best tradeoff between gaming Quality-of-Experience and profits
- Answer the question that “Are modern GPUs ready for cloud gaming?”



OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

NOTATIONS

- **Network Latency**: $e_{s,p}$
- **Frame Per Second**: $f_p(v)$
- **Processing Delay**: $d_p(v)$
- **CPU Utilization**: $u_s(v)$
- **GPU Utilization**: $z_s(v)$
- **Hourly fee:** g_p
- **Operational Cost:** $w_s(v)$
- **Memory of Server:** G_s
- **Uplink of Datacenter:** B_w

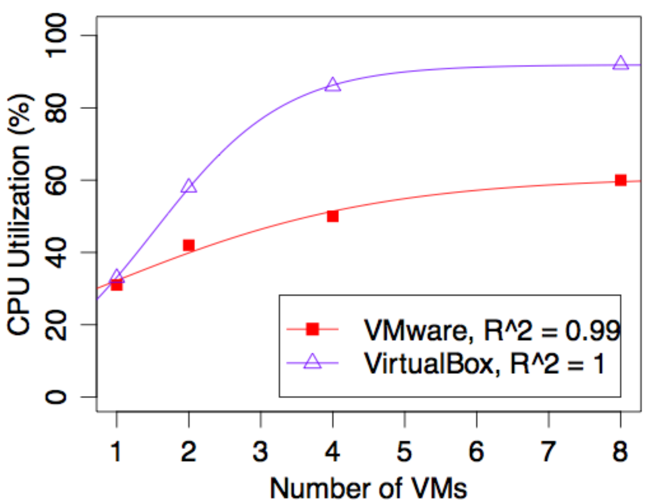
MODELS

- CPU utilization, GPU utilization, frame rate, and processing delay can be modeled as sigmoid functions of the number of VMs on a physical server
- $f_p(v) = \frac{\alpha_{p,1}}{1+e^{-\alpha_{p,2}v+\alpha_{p,3}}}$
- $d_p(v) = \frac{\beta_{p,1}}{1+e^{-\beta_{p,2}v+\beta_{p,3}}}$
-
- $u_s(v) = \frac{\delta_1}{1+e^{-\delta_2 v+\delta_3}}$
- $z_s(v) = \frac{\zeta_1}{1+e^{-\zeta_2 v+\zeta_3}}$

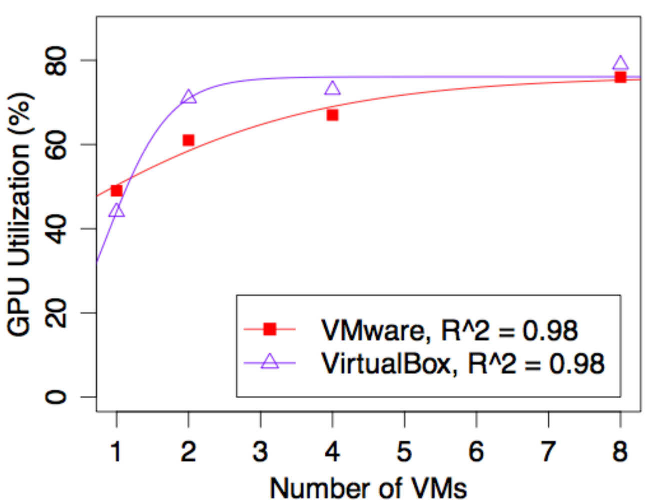
HOW CLOSE ARE THE SIGMOID FUNCTIONS

- The table shows the R-square values of different games/VM
- The figure shows the curve fitting results with different number of VMs

Game	VM	CPU	GPU	FPS	DELAY
Limbo	VMware	0.9910	0.9837	0.9767	0.9955
	VirtualBox	1.0000	0.9877	0.9933	0.9996
Normandy	VMware	0.9999	1.0000	0.9865	0.9995
	VirtualBox	0.9991	0.9986	0.9764	0.9995
PSR	VMware	0.5758	0.9961	0.9917	0.9974
	VirtualBox	0.9898	0.9360	0.9969	0.9943



(a)



(b)

PROBLEM FORMULATION

- **Decision variable:** $x_{s,p} \in \{0,1\},\ \forall 1 \le s \le S, 1 \le p \le P.$

Objective Function: Maximize Profits

$$\max \sum_{p=1}^{P}\sum_{s=1}^{S} x_{s,p} g_p - \sum_{s=1}^{S} c_s\left(\frac{\delta_1}{1+e^{-\delta_2 v_s+\delta_3}} + \frac{\zeta_1}{1+e^{-\zeta_2 v_s+\zeta_3}}\right)$$

Constraint: QoE Degradation

$$Q_p \ge \gamma_{p,1} f_p + \gamma_{p,2} \tilde{d}_p,\quad \forall p$$

Frame Per Second

$$f_p = \alpha_{p,1} / (1 + e^{-\alpha_{p,2}\sum_{s=1}^{S}(x_{s,p} v_s) + \alpha_{p,3}})$$

Delay

$$\tilde{d}_p = \frac{\beta_{p,1}}{1+e^{-\beta_{p,2}\sum_{s=1}^{S}(x_{s,p} v_s)+\beta_{p,3}}} + \sum_{s=1}^{S} e_{s,p} x_{s,p}$$

OTHER CONSTRAINTS

- $$1 = \sum_{s=1}^{S} x_{s,p}, \ \forall p$$
  $$B_w \geq B \sum_{s \in \mathbf{S}_w} \sum_{p=1}^{P} x_{s,p}$$
  $$G_s \geq G \sum_{p=1}^{P} x_{s,p}$$

OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

QUALITY-DRIVEN HEURISTIC (QDH)

- Consolidate more VMs on a server
- Do not exceed the user-specified maximal tolerable QoE degradation
- Pseudocode

```
for each gamer p = 1, 2, ..., P do
    sort servers on network latency to p in asc. order
    for each server s = 1, 2, ..., S do
        if serving p on s satisfies Eqs. (2)–(8) then
            let x_{s,p} = 1
            break
return x
```

Fig. 4: The pseudocode of the QDH algorithm.

QDH' – ALTERNATIVE ALGORITHM

- **Alternative Formulation and Algorithms for Closed Systems**
- **Objective Function:** $\min \left[\sum_{p=1}^{P} \gamma_{p,1} f_p + \sum_{p=1}^{P} \gamma_{p,2} \tilde{d}_p \right]$
- **Pseudocode**

---

for each gamer $p = 1, 2, \ldots, P$ **do**
    sort servers on quality degradation $q_p(\cdot)$ in asc. order
    for each server $s = 1, 2, \ldots, S$ **do**
        if serving p on s satisfies Eqs. (4.2)–(4.5), (4.7)–(4.8) **then**
            let $x_{s,p} = 1$
            break
return $\mathbf{x}$

Figure 5.1: The pseudocode of the QDH$'$ algorithm.

OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

COMPONENTS OF OUR SYSTEM

- **Broker**
  - VMware vCenter 5.1
  - Single-Sign-On: authentication
  - Inventory Service: managing/monitoring the VMs on ESXi servers
- **Physical Servers**
  - VMware ESXi 5.1
- **GA Client/Servers**
  - GA is the first open source cloud gaming system
  - Each VM host one GA server

FLOW OF OUR SYSTEM

① GA client sends the account and password from gamer to broker

② The broker authenticates the gamer

③ GA client sends the user-specified game to the broker

④ The broker determines where to create a new VM and instructs the chosen physical server to launch a VM

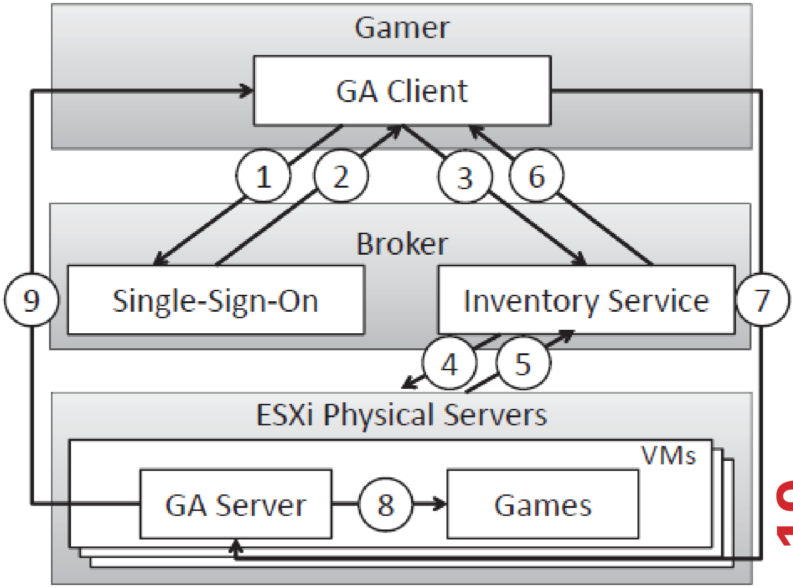
⑤ Physical server sends the VM's IP address to Broker

⑥ Broker Forwards the IP address to GA client

⑦ GA client connects to the GA server

⑧ GA server launches the game

⑨ GA server streams the game to gamer



SET UP – HARDWARE

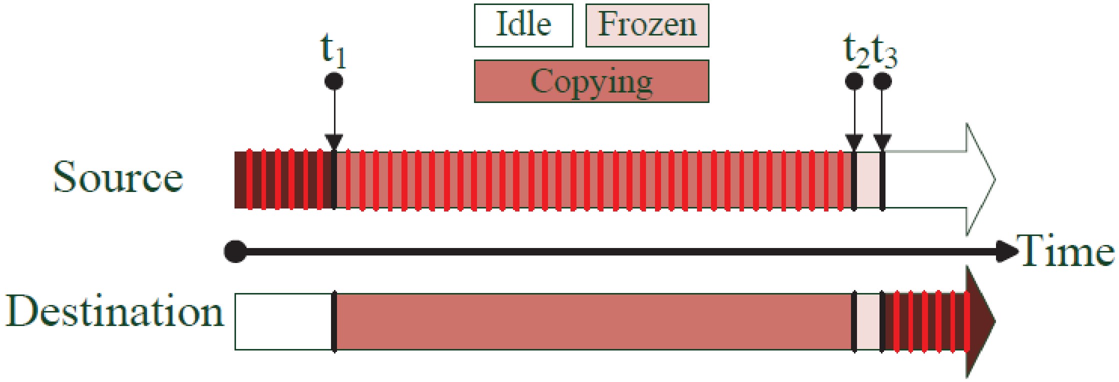
- **Physical Servers**
  - CPU: i5 3.5 GHz
  - GPU: Nvidia Quadro 6000
  - Memory: 16GB
- **Broker**
  - CPU: i7 3.2 GHz
  - Memory: 16GB
- **Clients**
  - CPU: i5
  - Memory: 4GB
- **VMs**
  - Equally allocate the CPU and Memory to VMs

SET UP - SCENARIO

- Join and leaving a game session with D% and (1-D)% probability (D% = 90%) in every minutes
- Game: Limbo, PSR, and Normandy
- Randomly select game
- Up to 2 VMs for each physical server
- Total time: 15 minutes

PRACTICAL CONCERN

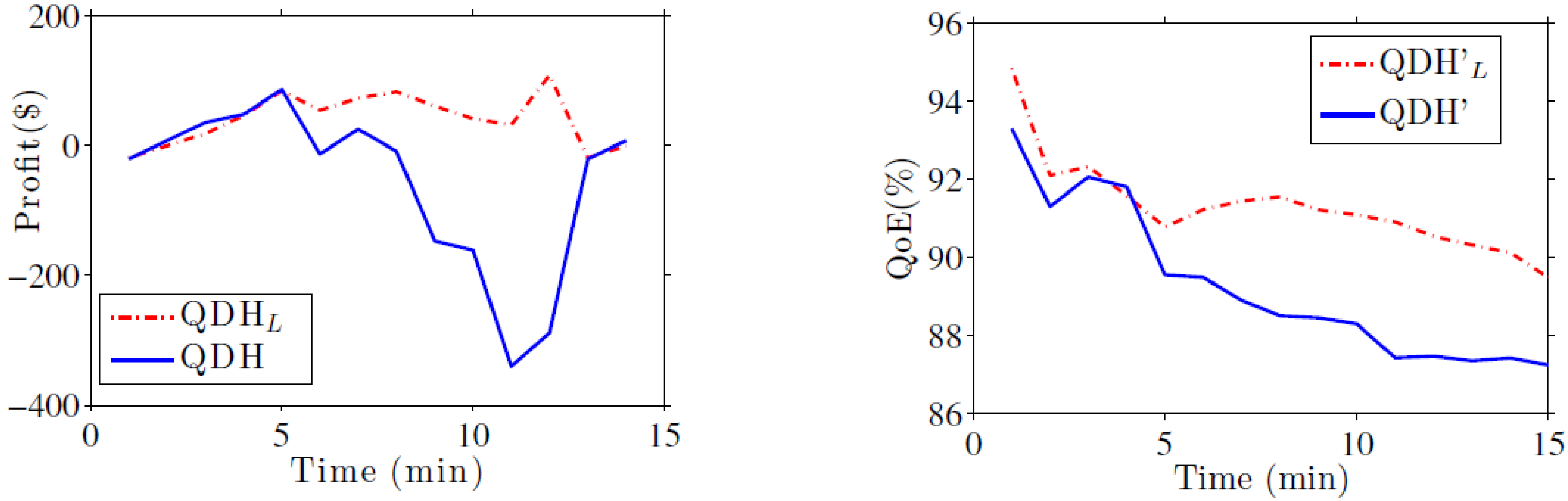
- **Migration time of 20, 30, and 40 GB VM images are about 6, 9, 11 minutes**
- **Double resources will be consumed between t1 to t3 while we do live migration**
  - Decrease the profits
  - Decrease the QoE
- **Hence, we consider an migrationless version of proposed QDH/QDH' algorithms**
  - Only place the VMs of incoming gamers to avoid the degradation caused by migration time



Live Migration

MIGRATIONLESS ALGORITHMS ARE BETTER

- **Outperforms QDH up to 396$ and 4% QoE**

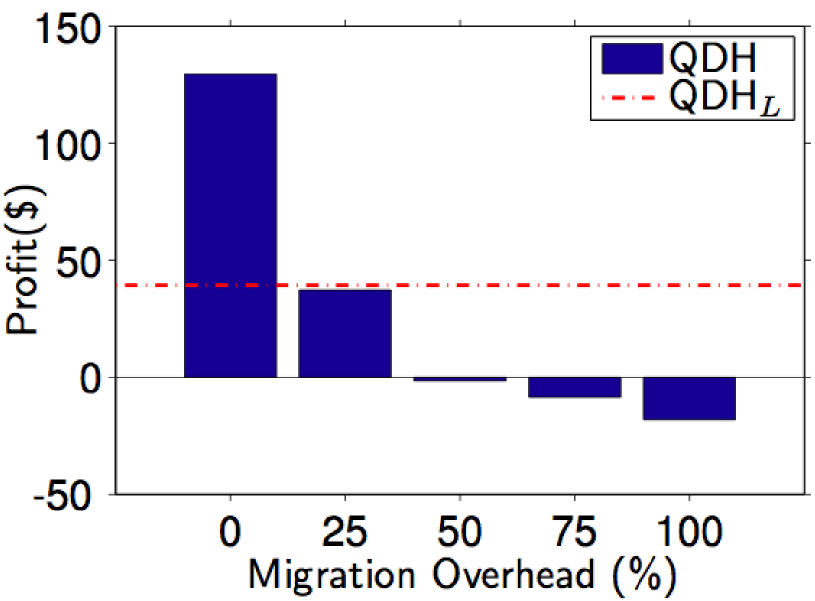


(a)

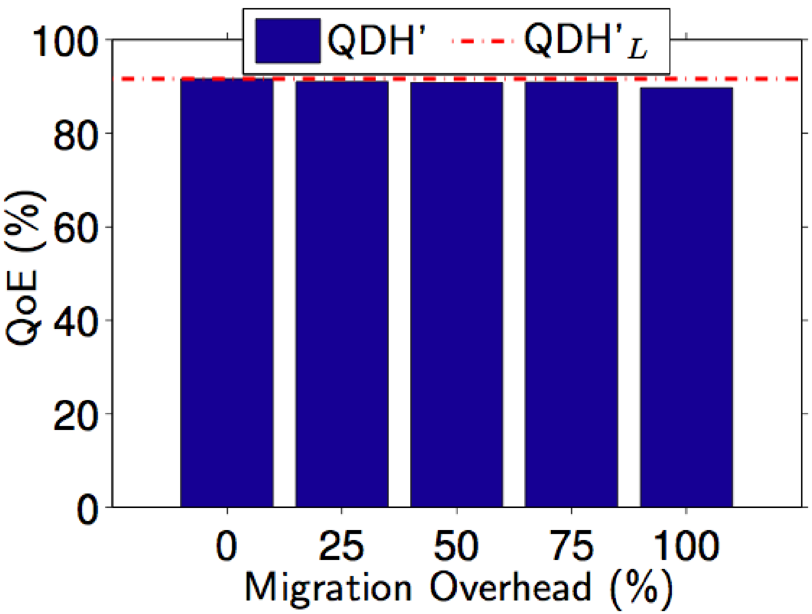
(b)

PERFORMANCE OF MIGRATIONLESS ALGORITHMS WITH DIFFERENT MIGRATION OVERHEAD

- **25% migration overhead will achieve the same profit**
- **Due to the increasingly higher computing power, the migration overhead will be gradually reduced and the performance gains may be diminishing**



(a)



(b)

CONCLUSION OF TESTBED

- **At this time, we do not consider QDH algorithm in the rest of the thesis**
- **QDH algorithm will be useful in the future while the migration time is reduced**

OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

SET UP

- **Network latencies: KING**
  - Server IP: OnLvie data center
  - Client IP: BitTorrent
- **WoW traces**
  - Arrival time and leaving time of gamers
- **Games**
  - Limbo, PSR, and Normandy
- **Computer:**
  - CPU: I7-3770 3.2 GHz
  - Memory: 16GB
- **VMs:**
  - Equally allocate the CPU and Memory to VMs

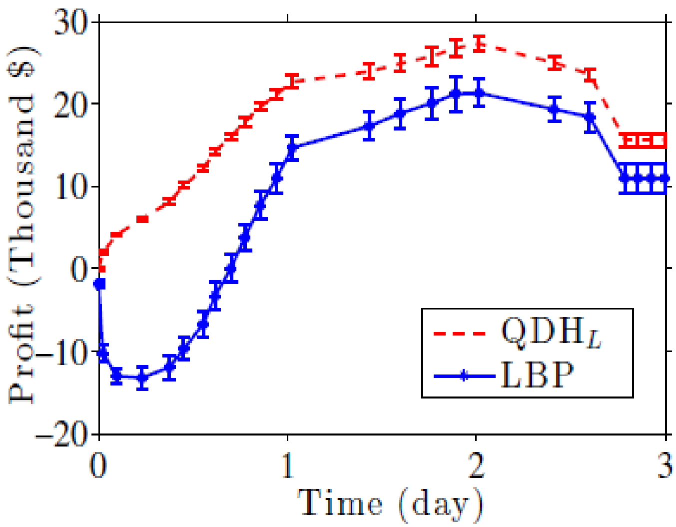
BASELINE ALGORITHM

Location Based Placement (LBP) algorithm:

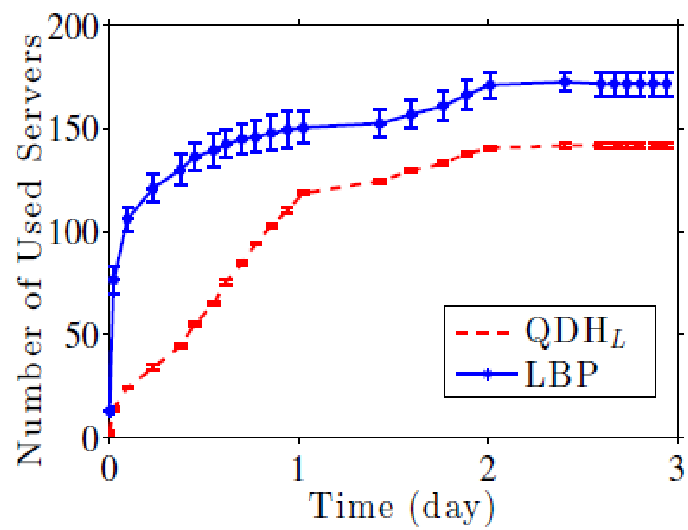
- **LBP places each VM on a random game server that is not fully loaded and the data center geographically closest to the gamer**

RESULTS OF PROVIDER CENTRIC ALGORITHM

- **Earn more money, up to 20+ thousand dollars**
- **Shutdown more servers**



(a)

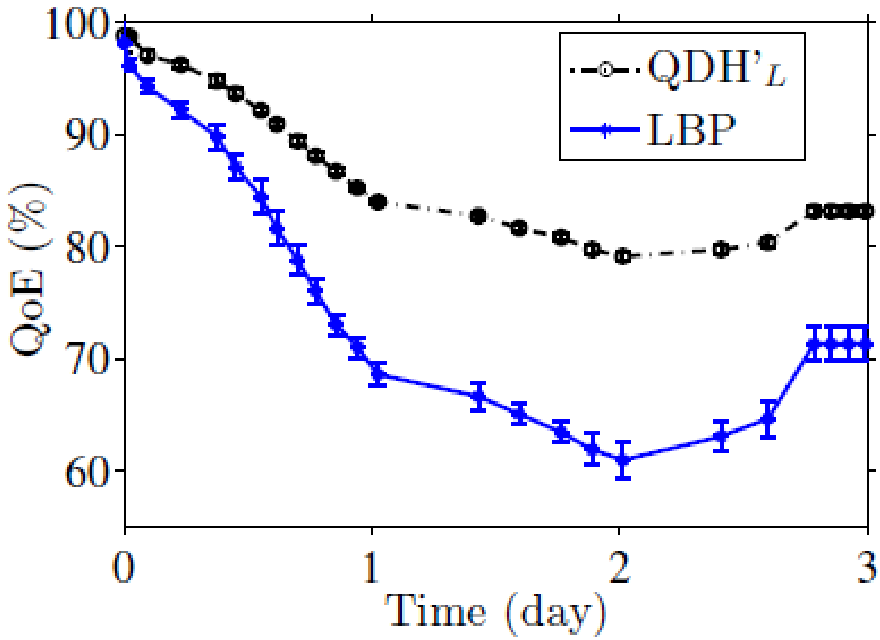


(b)

Simulation results with WoW traces: (a) net profits and (b) used servers

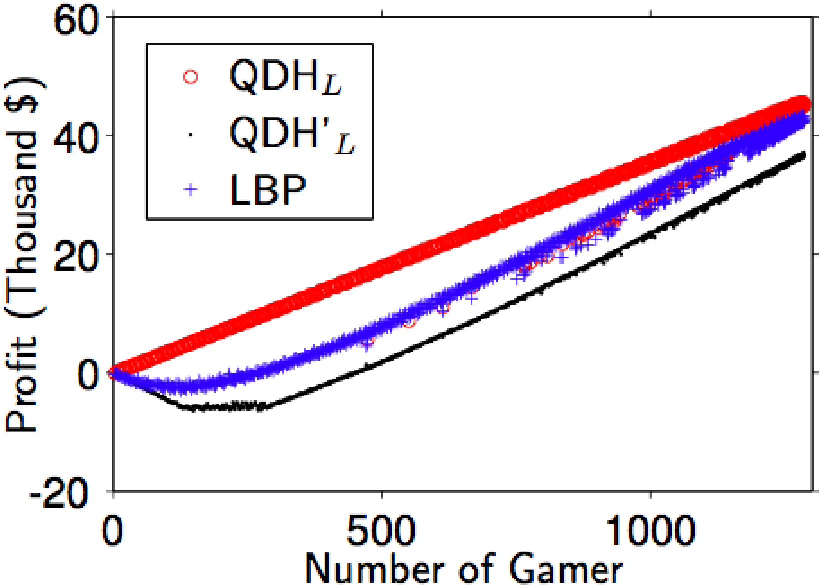
RESULTS OF GAMER CENTRIC ALGORITHM

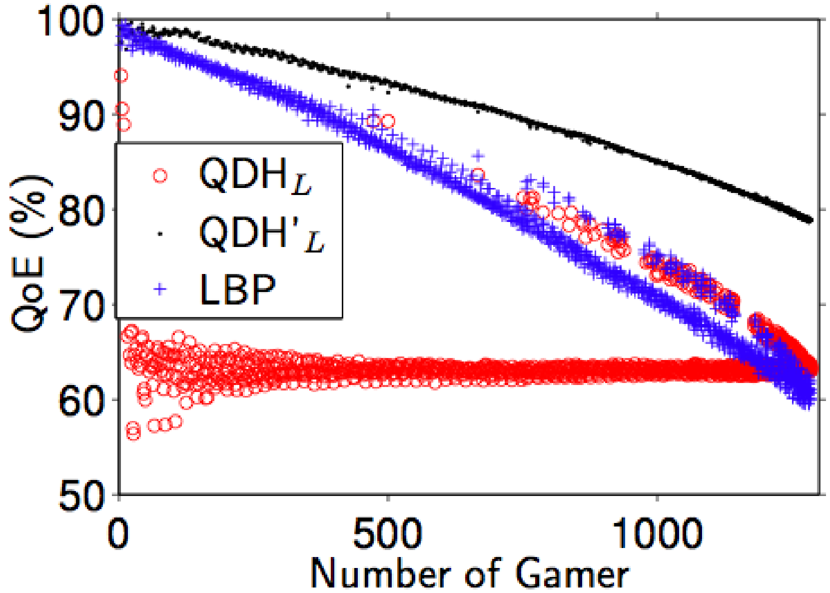
- Outperform LBP algorithm up to 130% QoE



IMPACT OF NUMBER OF GAMERS

- The figure shows that more gamers lead to higher profits and lower QoE levels, and QDH_L/QDH'_L successfully **achieve their design objectives**





RUNNING TIME

- **The efficient algorithms terminate in < 2.5 s on a commodity PC even for large services with 20000 servers and 40000 gamers**

Running Time in Seconds

# of Servers	$\mathbf{QDH_L}$		$\mathbf{QDH'_L}$	
	Mean	Max	Mean	Max
5000	0.215	0.853	0.02	0.05
10000	0.379	0.967	0.05	0.07
15000	0.557	1.9	0.07	0.12
20000	0.819	2.52	0.12	0.23

OUTLINE

- Introduction
- Problem Formulation
- QDH Algorithm
- Testbed
- Trace-Driven Simulations
- Measurement Study of Modern GPU

MODERN GPU

- **Nvidia Quadro 6000**
  - Released in 2010
- **Nvidia K2**
  - Released in 2013
  - Support vGPU
  - Each instance can be configured to:
    1. Pass-through
    2. vGPU with up to 2,4,or 8 VMs

GPU	Year	Core	Memory	No. Inst.	vSGA	vGPU
Quadro 6000	2010	448	6 GB	1	Yes	No
K2	2013	3072	8 GB	2	Yes	Yes

Specifications of Two GPUs

SET UP

- **Cloud gaming server:**
  - OS: XenServer 6.2
  - CPU: Xeon 2.1 GHz
  - Memory: 64 GB
- **VM:**
  - By default, the XenServer allocates 1 CPU core and 2GB memory to Dom0, which is responsible for managing VMs
  - The remaining CPU cores and memory are equally divided among the VMs running Windows 7

WORKLOAD

- **Game:**
  - Limbo: scroll-based puzzle game
  - Fear2: first person shooter game
  - LEGO Batman: action game
- **Benchmark:**
  - Sanctuary: GPU benchmark
  - Cadalyst: 2D versus 3D
- **Tinytask:**
  - A program to record the mouse and keyboard inputs of each game
  - We record 3 minutes for each game and replay the same inputs to ensure fair comparisons

PERFORMANCE METRICS

- **Frame Per Second**
  - The number of rendered frames per second
- **Context Switch**
  - The number of context switches in Dom0
- **CPU Utilization**
  - The CPU load of Dom0 (CPU_{dom0}) and each VM (CPU_{vm}).
- **GPU Utilization**
  - The load of GPUs

MEASUREMENT UTILITIES

- **Fraps:** To measure the FPS of the foreground window
- **Sar:** To measure the number of context switches
- **Xentop:** To measure the CPU utilization of Dom0 and VMs
- **Nvidia-smi:** To measure the GPU utilization under vGPU
- **GPU-Z:** To measure the GPU utilization of pass-through GPUs

PERFORMANCE OF TWO MODERN GPUS

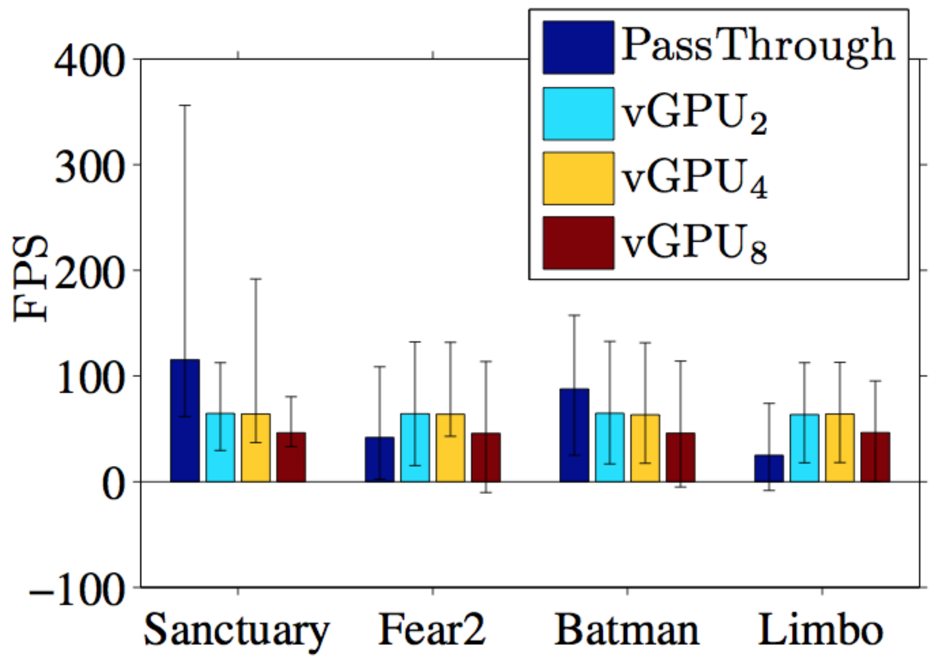
- This table shows that K2 outperforms Quadro 6000 with up to 3.87 times of FPS increases
- Scalability: FPS of K2 does not drop too much even with 8 VMs
- Huge edge of vGPU (mediated pass-through) over vSGA (software-based virtualization)
- we no longer consider Quadro 6000 and vSGA in the rest of this thesis

# of VMs	Quadro 6000	K2	Speed-up (times)
2 VMs	22.3	32.8	1.47
4 VMs	13.1	26.9	2.05
8 VMs	7.0	27.1	3.87

Achieved frame rates on two considered GPUs

SHARED GPUS MAY OUTPERFORM DEDICATED GPUS

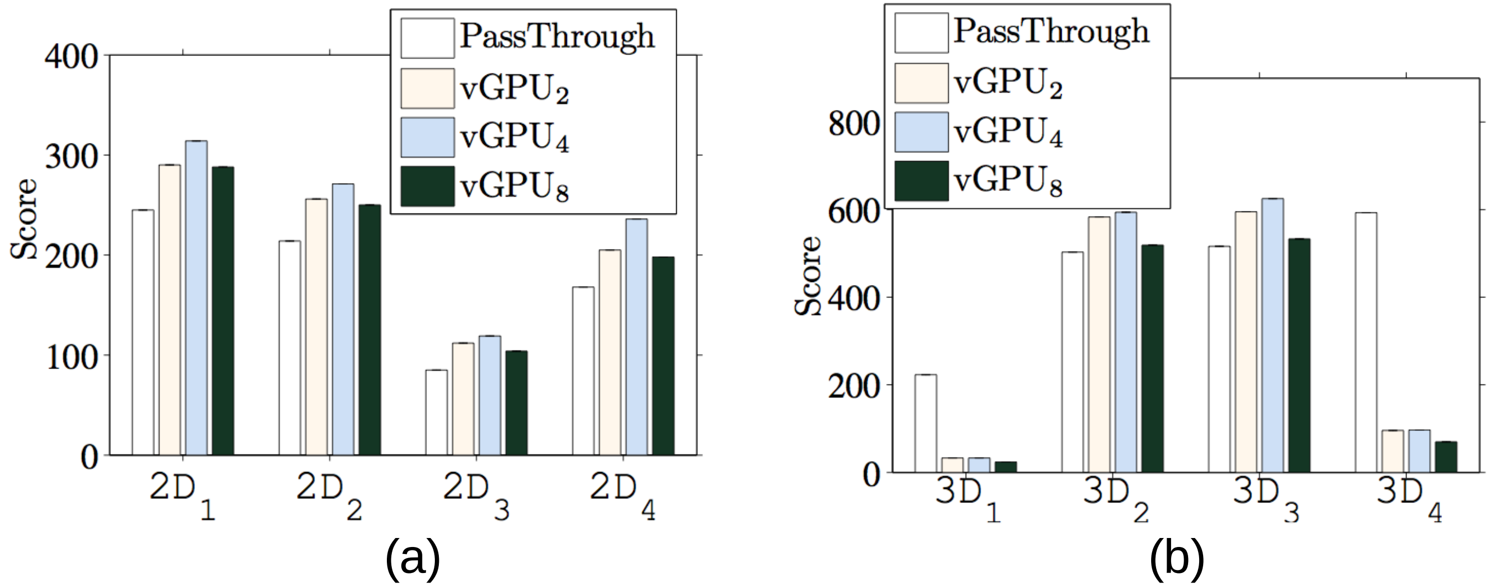
- **vGPU results in higher FPS than pass-through when executing Limbo and Fear2**



Comparing the pass-through and vGPU: resulting FPS

2D/3D PERFORMANCES

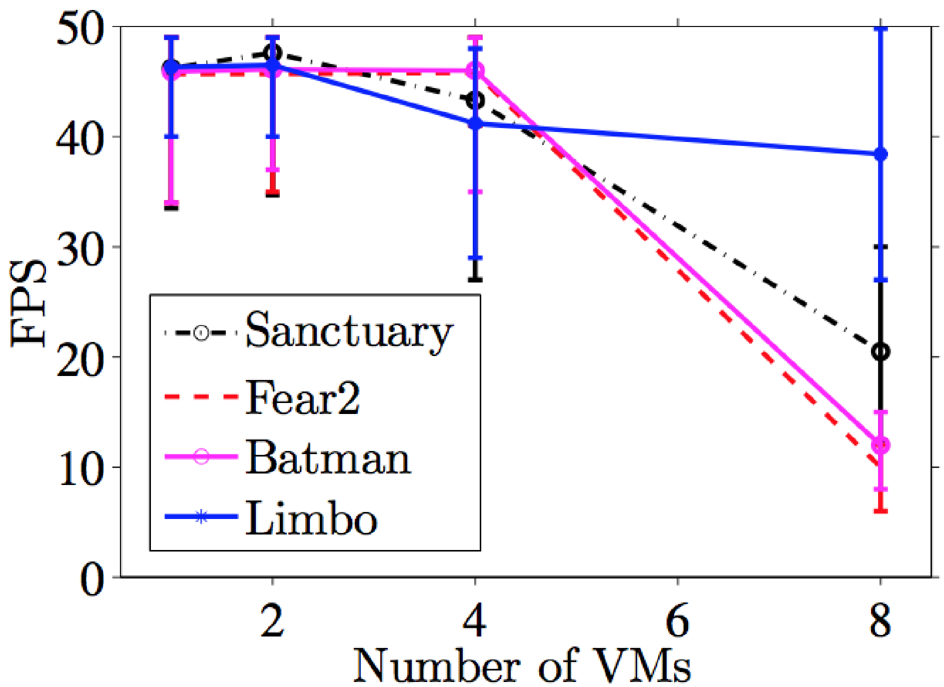
- **vGPU$_2$ outperforms pass-through …**
  - All 2D operations up to 15%
  - Part of 3D operations
- **Similar observations are also true for vGPU$_4$ and vGPU$_8$**



Comparing the pass-through and vGPU: (a) 2D benchmark scores and (b) 3D benchmark scores.

CONSOLIDATION OVERHEAD

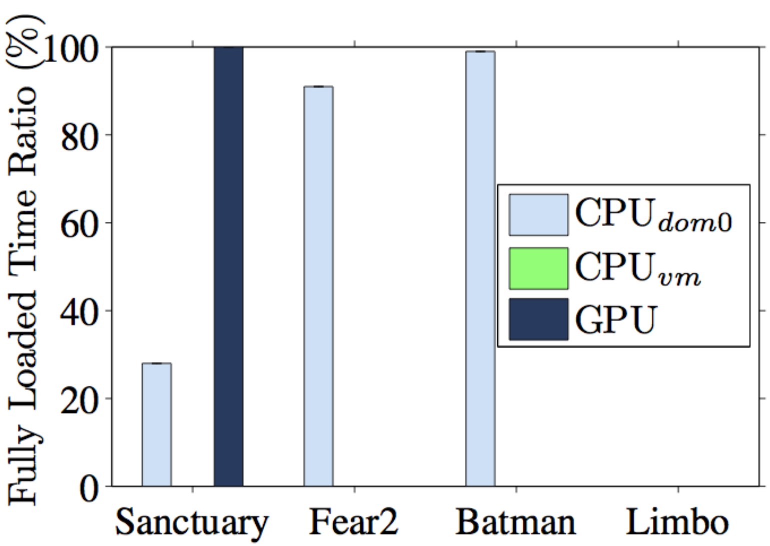
- **Limbo does not suffer from consolidation overhead, while all other games/benchmark do**



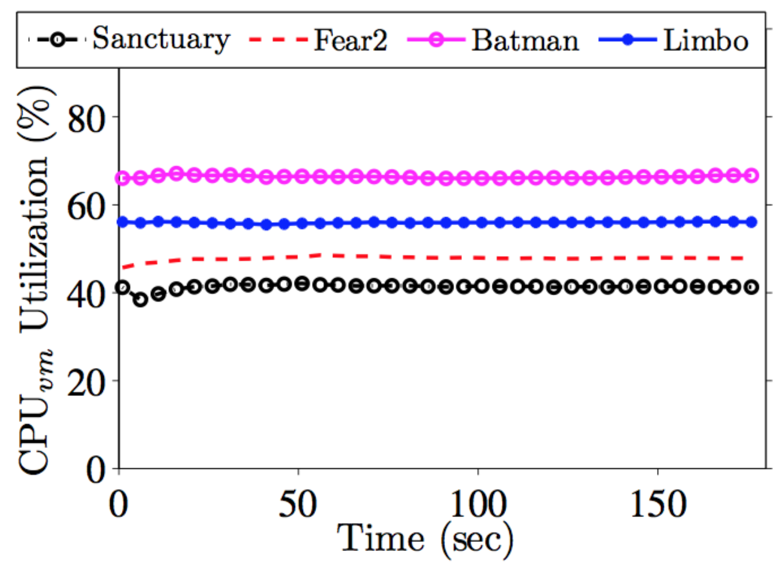
GPU consolidation overhead: resulting FPS

CONSOLIDATION OVERHEAD CAUSED BY…

- **The figure shows that Sanctuary is bounded by GPU, while Fear2 and Batman are bounded by CPU_{dom0}**
- **Allocating more CPU cores to Dom0 to alleviate the high consolidation overhead for more complex games.**



(a)

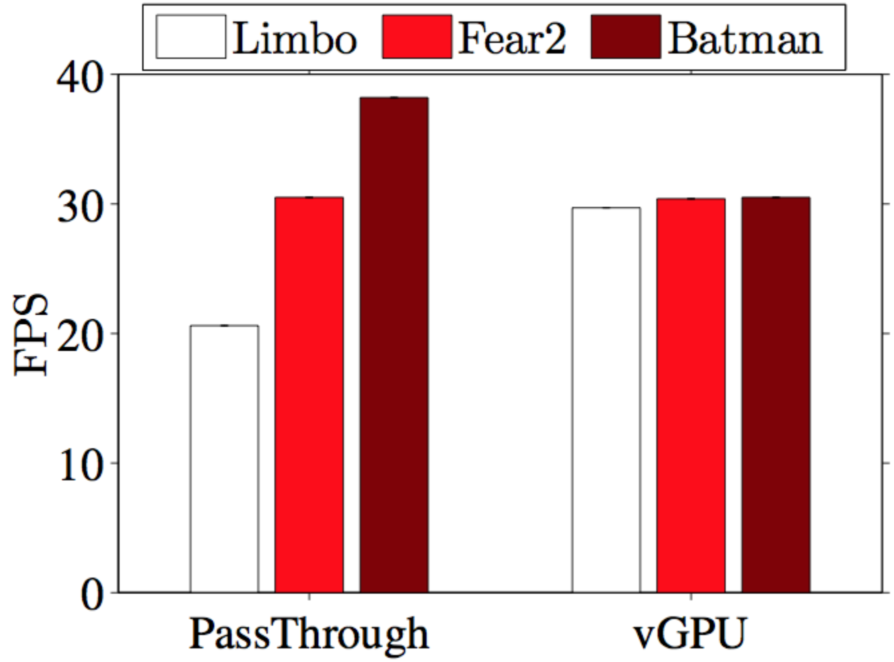


(b)

GPU consolidation overhead: (a) fully loaded time ratio from $vGPU_8$, and (b) CPU_{vm} from $vGPU_8$.

END-TO-END CLOUD GAMING PERFORMANCE

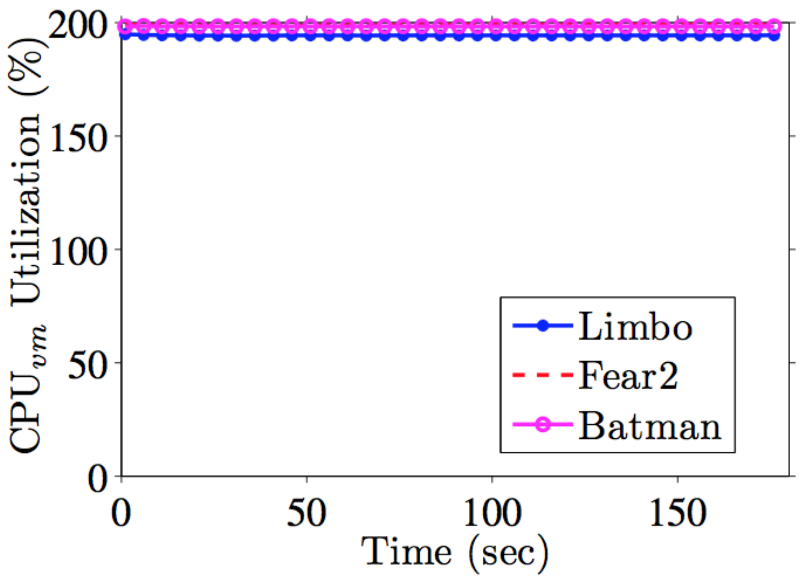
- Only 1 VM
- Open source cloud gaming system: Gaming anywhere
- Not good-enough quality which between 20~42 fps



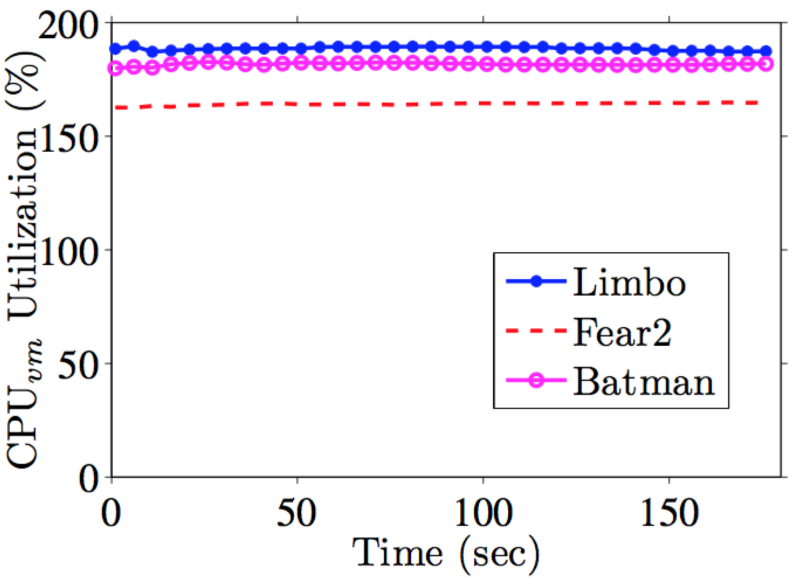
End-to-end performance of a cloud game platform: resulting FPS

REASON OF THE LOW GAMING QUALITY

- **Real-time video encoding relies on computing power of CPU**
- **Leverage the hardware codec on K2 GPU to improve it**



(a)



(b)

End-to-end performance of a cloud game platform: (a) CPU_{vm} utilization with pass-through GPU and (b) CPU_{vm} utilization with $vGPU_2$.

CONCLUSION

- **VM placement algorithms [NetGames'13 short and IEEE TCC'14]:**
  - Migrationless algorithms outperform the state-of-the-art algorithm up to 20+ thousand dollars in net profits and 130% performance in QoE
  - The efficient algorithms terminate in 2.5 s on 20000 servers and 40000 clients
- **GPU measurement [MM'14 short under review]:**
  - Shared GPUs may outperform dedicated GPUs
  - Shared GPUs are rather scalable to the number of VMs
  - Modern GPUs can be shared by VMs running GPU-intensive computer games

FUTURE WORK

- **Hardware codec**
  - Leveraging the hardware H.264 codecs to improve the performance of real-time encoding
- **More comprehensive system models**
  - Other types of resources
  - Heterogeneous server types

Q&A