Yuzhao WANG, Junqing YU, Zhibin YU, 2023. Resource scheduling techniques in cloud from a view of coordination: a holistic survey.
Frontiers of Information Technology & Electronic Engineering, 24(1):1-40.
https://doi.org/10.1631/FITEE.2100298

Resource scheduling techniques in cloud from a view of coordination: a holistic survey

Key words: Coordination; Co-location; Heterogeneous computing; Microservice; Resource scheduling techniques

Yuzhao WANG
E-mail: yuzhao_w@hust.edu.cn
ORCID: https://orcid.org/0000-0002-9056-4277

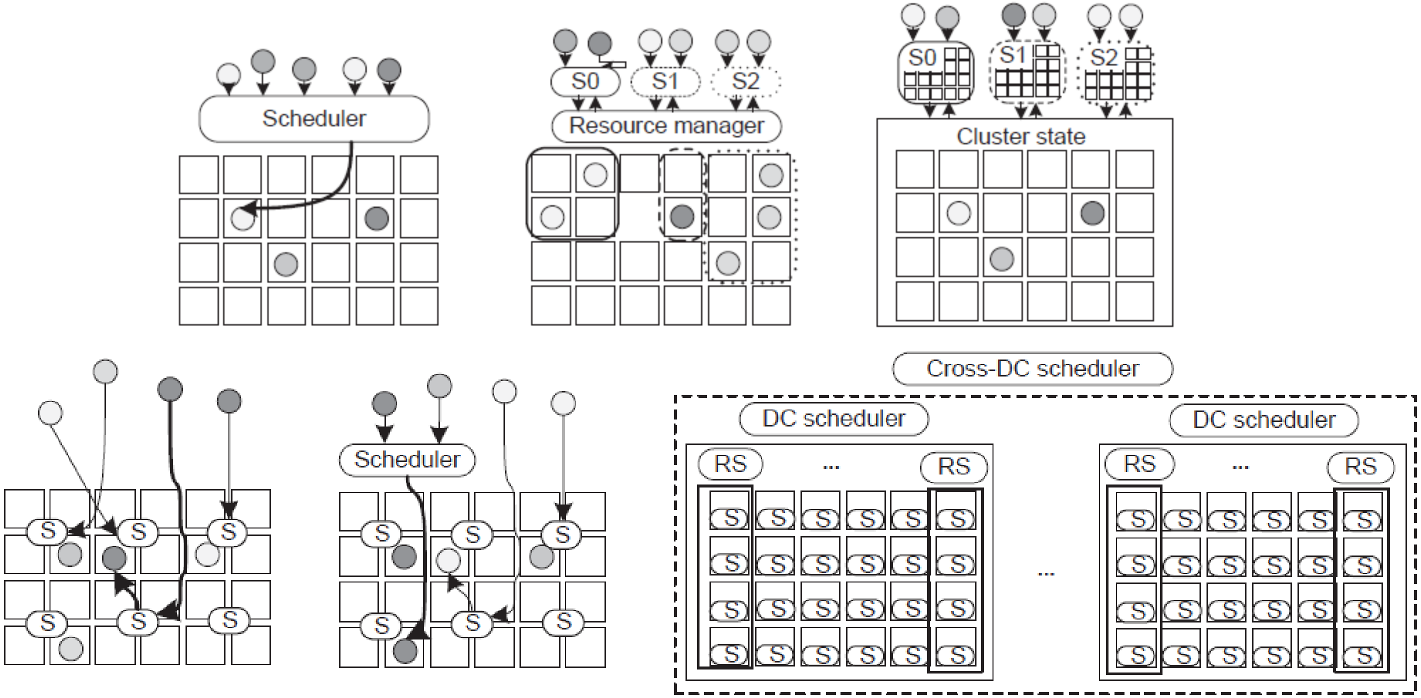
Resource scheduling progress in cloud

- ☐ With the proliferation of new computing paradigm and widespread use of cloud computing, scheduling techniques are evolving.

Date	Event
2010	The delay scheduling in MapReduce is released accounting data locality, give birth to Spark; Breakthrough of traditional computing in HPC
2011–2013	The golden age of resource management system, Mesos and Yarn; Unified architecture for diverse applications with high scalability
2013–2015	The release and development of distributed system, e.g., Sparrow, Tarcil; Research hotspot of decentralized scheduling system
2015–2016	Combination of monolithic and distributed architectures, give birth to hybrid scheduling arch
2017–2020	Hierarchical systems integrating more functions with fine-grain control and application-specific systems e.g., Ray emerge

Tendency and challenges

- **More heterogeneity**: application, hardware, and runtime dynamicity motivate the evolution of scheduling system
- **Challenges**:
  (1) Inefficient isolation support
  (2) Resource utilization v.s. QoS
  (3) Coordination of collocated applications
  (4) Coordinated management of hardware (e.g., CPU & GPU)



Evolution of scheduling architectures □: server; S: scheduler;

Motivation

☐ Highlight the **shortage & dilemma** from a systematic perspective

☐ Summarize the main considerations for resource management system design/optimization and provide a global view of the landscape and technical trends in shared cloud

For example:

Arch.	**Example**	**Scalability**	**Scheduling delay**	**Scheduling quality**
Centralized	TORQUE, Borg [1]	Bad	High	Good
Two-level	Mesos [2], YARN	Medium	Low	Medium
Shared-state	Omega [3], Apollo	Good	Medium	Good
Distributed	Sparrow [4], Tarcil	Best	Low	Bad
Hybrid	Hawk [5], Mercury	Good	Low	Medium
Hierarchical	Fuxi 2.0, Twine	Good	Low–medium	Good

1) Resource isolation summary and analysis

☐ Approaches at micro-architecture: one relies on hardware/kernel support, the other relies on software optimizations.

(1) Hardware/kernel-based: including *Cgroups, Namespace, CAT, DVFS*

Pros: Efficient API (e.g., via system call); effective resource isolation

Cons: Targeting a single resource; lacking interplay characterization or co-optimization knobs regarding different resources; lacking complete support of shared resources

(2) Software-based: including *thread-scheduling, cache partition policy, page-coloring*

Pros: Informed isolation via a better understanding of performance impact of shared resources

Cons: Profile-based and requiring expert knowledge

☐ Approaches at the virtualization level

(1) VM-based: best isolation, long startup delay, large memory footprint

(2) Container-based: short startup, small memory footprint

1) Resource isolation summary and analysis

☐ Isolation approaches for GPU multitasking

(1) **Time multiplexing**

- Preemption: context switch; streaming multi-processor (SM) draining.
- Time-sharing GPU registers with high warp concurrency

(2) **Spatial multiplexing**

- Static cache and memory bandwidth partition among multiple kernels
- Simultaneous multikernel enables co-scheduled kernels
- Dynamic GPU thread number adjustment
- Dynamic page walker partition among the applications
- TLB miss-based policy to regulate the number of warps of contended apps

■ **Problems:**

1. Prior efforts often focused on individual resources, ignoring underlying inter-resource interaction

2. Resource isolation on devcies, e.g., FPGA requires more attention

2) Resource managements analysis at system level

☐ Scheduling against hardware resources: **symbiosis analysis**

(1) Simultaneous multi-threading (SMT): complementary threads are co-located relying performance model; poor scalability

(2) Memory subsystem: characteristic-aware task scheduling minimizing contention on shared resources, e.g., memory bank/bus, cache

(3) Non-uniform memory access (NUMA): ensure thread-data affinity via memory migration

(4) Specialized hardware (e.g., GPU): workload offloading/balancing algorithms to maximize advantage; I/O bottleneck

☐ OS designs for heterogeneity: specialized processor architectures

(1) OS on multi-processor: independent management at processor with inter-coordination

(2) OS on heterogenous memory: application-oriented hotness tracking and type-aware memory allocation

(3) OS over GPU/FPGA: new abstractions to promote to general-purpose, shared resource

3) Scheduling arch. summary at the cluster level

- Scheduling against hardware resources: **symbiosis analysis**

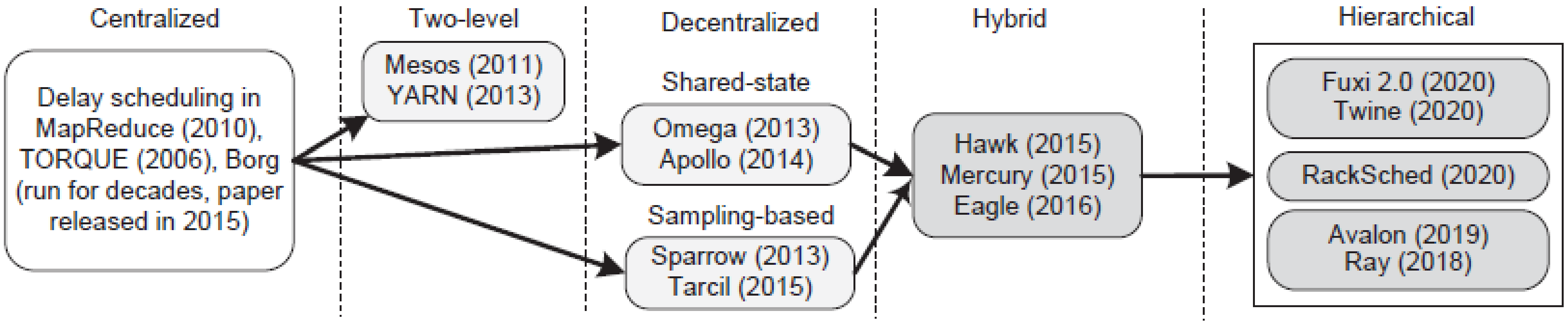
(1) Centralized: a single, integrated scheduling logic for all

(2) Two-level: unified interface for all applications with better scalability

(3) Shared-state: full access to all cluster resources with conflict resolution

(4) Distributed: multiple independent schedulers, each with partial visibility to cluster

(5) Hybrid architecture: application type-aware inter-scheduler coordination



Architectural shift of scheduling systems

4) Task-to-machine mapping algorithm analysis

- Four kinds of algorithm are summarized.

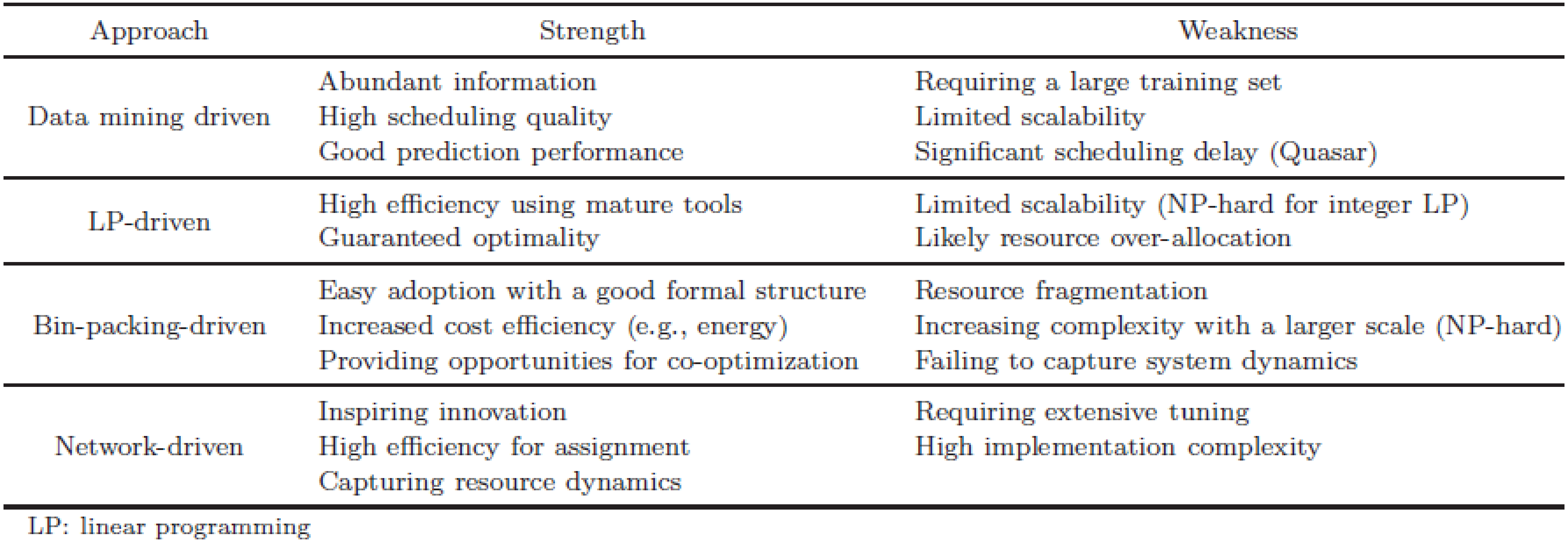
  (1) Data mining driven: relying on knowledge of history data

  (2) Linear-programming: LP formulation of resource allocation

  (3) Bin-packing: individual server as bin, task as item packed

  (4) Network-driven: as a min-cost max-flow problem, neural network-based deep learning.

Summary and analysis of different algorithms



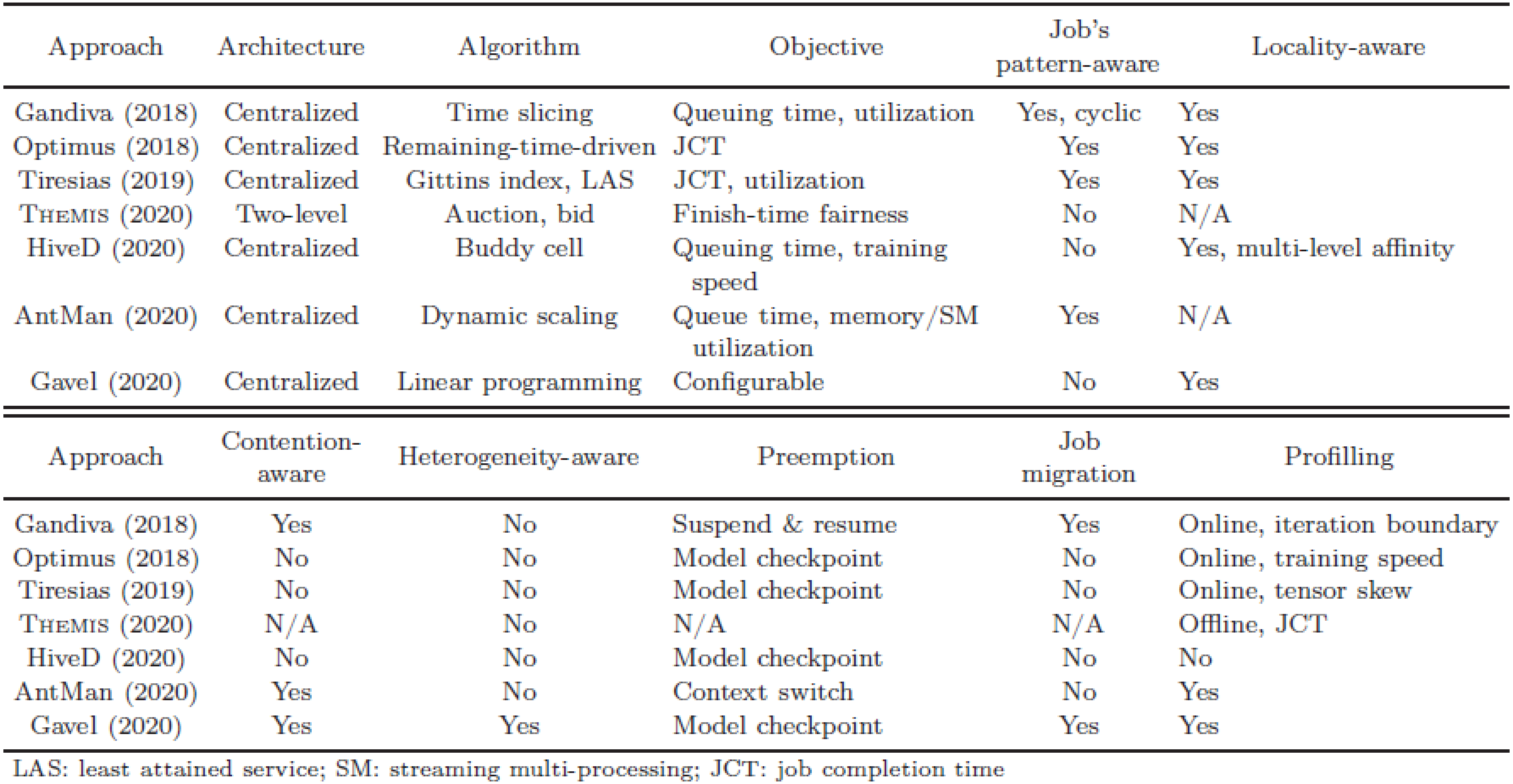
Approach	Strength	Weakness
Data mining driven	Abundant information High scheduling quality Good prediction performance	Requiring a large training set Limited scalability Significant scheduling delay (Quasar)
LP-driven	High efficiency using mature tools Guaranteed optimality	Limited scalability (NP-hard for integer LP) Likely resource over-allocation
Bin-packing-driven	Easy adoption with a good formal structure Increased cost efficiency (e.g., energy) Providing opportunities for co-optimization	Resource fragmentation Increasing complexity with a larger scale (NP-hard) Failing to capture system dynamics
Network-driven	Inspiring innovation High efficiency for assignment Capturing resource dynamics	Requiring extensive tuning High implementation complexity

LP: linear programming

4) Task-to-machine mapping algorithm analysis

☐ DL-oriented GPU scheduling approaches

- Representative efforts are summarized as below

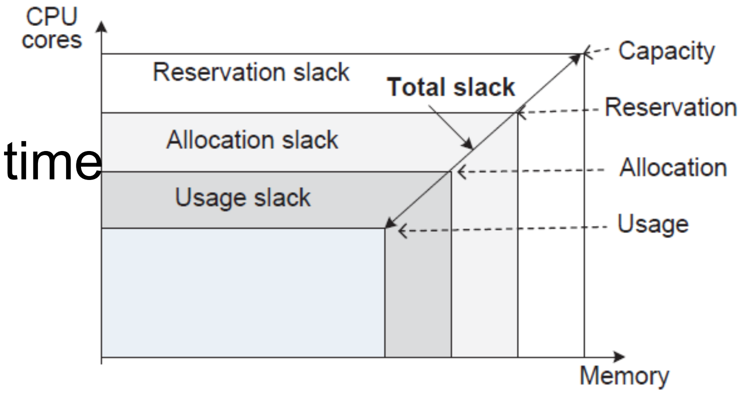


Approach	Architecture	Algorithm	Objective	Job's pattern-aware	Locality-aware
Gandiva (2018)	Centralized	Time slicing	Queuing time, utilization	Yes, cyclic	Yes
Optimus (2018)	Centralized	Remaining-time-driven	JCT	Yes	Yes
Tiresias (2019)	Centralized	Gittins index, LAS	JCT, utilization	Yes	Yes
THEMIS (2020)	Two-level	Auction, bid	Finish-time fairness	No	N/A
HiveD (2020)	Centralized	Buddy cell	Queuing time, training speed	No	Yes, multi-level affinity
AntMan (2020)	Centralized	Dynamic scaling	Queue time, memory/SM utilization	Yes	N/A
Gavel (2020)	Centralized	Linear programming	Configurable	No	Yes

Approach	Contention-aware	Heterogeneity-aware	Preemption	Job migration	Profilling
Gandiva (2018)	Yes	No	Suspend & resume	Yes	Online, iteration boundary
Optimus (2018)	No	No	Model checkpoint	No	Online, training speed
Tiresias (2019)	No	No	Model checkpoint	No	Online, tensor skew
THEMIS (2020)	N/A	No	N/A	N/A	Offline, JCT
HiveD (2020)	No	No	Model checkpoint	No	No
AntMan (2020)	Yes	No	Context switch	No	Yes
Gavel (2020)	Yes	Yes	Model checkpoint	Yes	Yes

LAS: least attained service; SM: streaming multi-processing; JCT: job completion time

5) Adaptive resource management techniques



- Three lines of study are investigated
  - General elasticity for task colocation
    - (1) Classification of resource slacks at runtime
    - (2) Investigation of existing approaches
    - (3) Principles for elastic scheduling
  - Best-effort job-oriented resource throttling
    - (1) Anomaly detection and tracing
    - (2) Resource throttling & culprit migration
  - Resource compensation for latency-sensitive job
    - (1) Tail-tolerant techniques for online services
    - (3) QoS guarantee for microservice
- Lessons from prior work

  …

Future outlook

Category	Outlook
Coordination across system layers	• Decoupled architecture results in more system layers • Hierarchical scheduling will be widespread • **Cooperation of different components** calls for more coordination
Hardware architecture disaggregation	• Traditional functionalities decouple into **loosely coordinated managers** for each disaggregated resource type • **Integrated evaluation** is pressing, including the scheduling policies • Corresponding application paradigm is needed
Accelerator-oriented coordination	• Ubiquitous access to various accelerators • Combinations of different accelerators, computation offloading, and resource multiplexing requires more effort, including **intra/inter-accelerator management**
Edge-cloud coordination	• **Collaborative processing model** is needed to balance response latency and network traffic • Coordinated solutions are essential in the IoT era

References

[1] Verma A, Pedrosa L, Korupolu M, et al., 2015. Large-scale cluster management at Google with Borg. Proc 10th European Conf on Computer Systems, Article 18. https://doi.org/10.1145/2741948.2741964

[2] Hindman B, Konwinski A, Zaharia M, et al., 2011. Mesos: a platform for fine-grained resource sharing in the data center. Proc 8th USENIX Conf on Networked Systems Design and Implementation, p.295-308.

[3] Schwarzkopf M, Konwinski A, Abd-El-Malek M, et al., 2013. Omega: flexible, scalable schedulers for large compute clusters. Proc 8th ACM European Conf on Computer Systems, p.351-364. https://doi.org/10.1145/2465351.2465386

[4] Ousterhout K, Wendell P, Zaharia M, et al., 2013. Sparrow: distributed, low latency scheduling. Proc 24th ACM Symp on Operating Systems Principles, p.69-84. https://doi.org/10.1145/2517349.2522716

[5] Delgado P, Dinu F, Kermarrec AM, et al., 2015. Hawk: hybrid datacenter scheduling. Proc USENIX Annual Technical Conf, p.499-510.

[6] Xiao WC, Bhardwaj R, Ramjee R, et al., 2018. Gandiva: introspective cluster scheduling for deep learning. Proc 13th USENIX Conf on Operating Systems Design and Implementation, p.595-610.

[7] Peng YH, Bao YX, Chen YR, et al., 2018. Optimus: an efficient dynamic resource scheduler for deep learning clusters. Proc 20th EuroSys Conf, Article 3. https://doi.org/10.1145/3190508.3190517

[8] Gu JC, Chowdhury M, Shin KG, et al., 2019. Tiresias: a GPU cluster manager for distributed deep learning. Proc 16th USENIX Symp on Networked Systems Design and Implementation, p.485-500.

[9] Mahajan K, Balasubramanian A, Singhvi A, et al., 2020. Themis: fair and efficient GPU cluster scheduling. Proc 17th USENIX Symp on Networked Systems Design and Implementation, p.289-304.

[10] Zhao HY, Han ZH, Yang Z, et al., 2020. HiveD: sharing a GPU cluster for deep learning with guarantees. Proc 14th USENIX Symp on Operating Systems Design and Implementation, p.515-532.

[11] Xiao WX, Ren SR, Li Y, et al., 2020. AntMan: dynamic scaling on GPU clusters for deep learning. Proc 14th USENIX Symp on Operating Systems Design and Implementation, p.533-548.

Yuzhao WANG, received the BS degree from Henan University of Engineering in 2013, and the MS and PhD degrees from Huazhong University of Science and Technology (HUST), WuHan, China, in 2015 and 2021, respectively. He is current a software engineer at Huawei Technology Co. His research interests include parallel and distributed systems, cloud computing, and machine learning.

Junqing YU, received the PhD degree in computer science from Wuhan University, in 2002. He is currently a professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China. His current research interests include digital video processing and retrieval, multimedia, and parallel algorithm.

Zhibin YU, received the PhD degree in computer science from Huazhong University of Science and Technology (HUST) in 2008. Currently, he is a professor in SIAT. His research interests include microarchitecture simulation, computer architecture, workload characterization and generation, performance evaluation, multicore architecture, GPGPU architecture, virtualization technologies, big data processing, and so forth. He received the outstanding technical talent program of Chinese Academy of Science (CAS) in 2014 and the "peacock talent" program of Shenzhen City in 2013. He serves for ISCA 2013, MICRO 2014, ISCA 2015, and HPCA 2015. He is a member of IEEE and ACM.