Effective Computing for AI

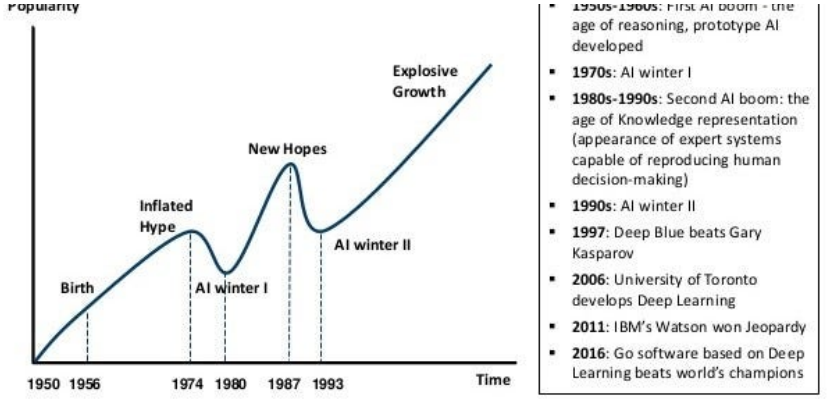
须成忠, 澳门大学

Chengzhong Xu, University of Macau

November 22, 2023

AI & AIGC Background

AI has a long history of being “the next big thing”…



- **1950s-1960s**: First AI boom - the age of reasoning, prototype AI developed
- **1970s**: AI winter I
- **1980s-1990s**: Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
- **1990s**: AI winter II
- **1997**: Deep Blue beats Gary Kasparov
- **2006**: University of Toronto develops Deep Learning
- **2011**: IBM’s Watson won Jeopardy
- **2016**: Go software based on Deep Learning beats world’s champions

(Rise and Fall of AI, :online source)

AI is coming of age …

AI for Entertainment

AlphaGo ALPHAZERO

AI for Science

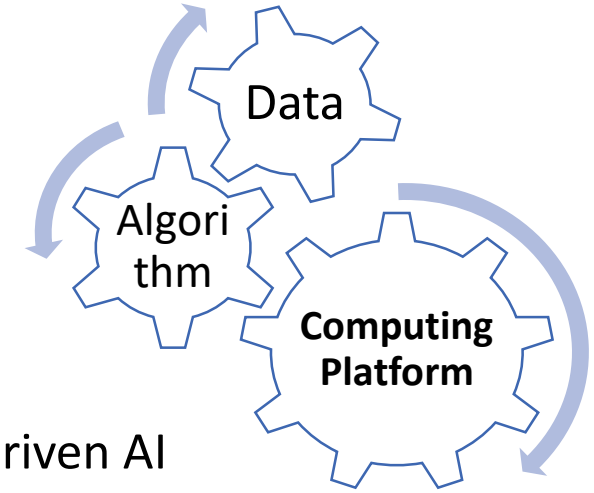
AlphaFold

AI Generated Contents

ChatGPT

Midjourney

Foundation Model

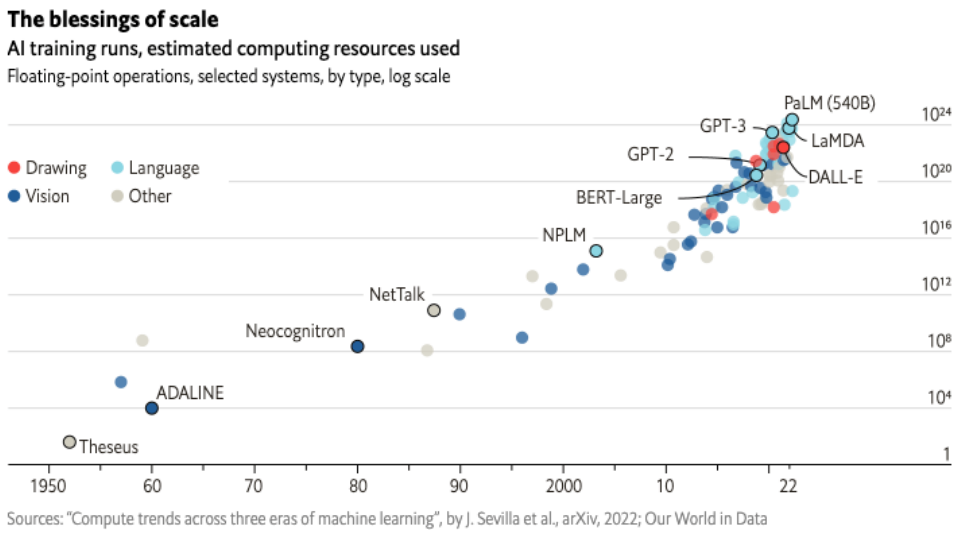


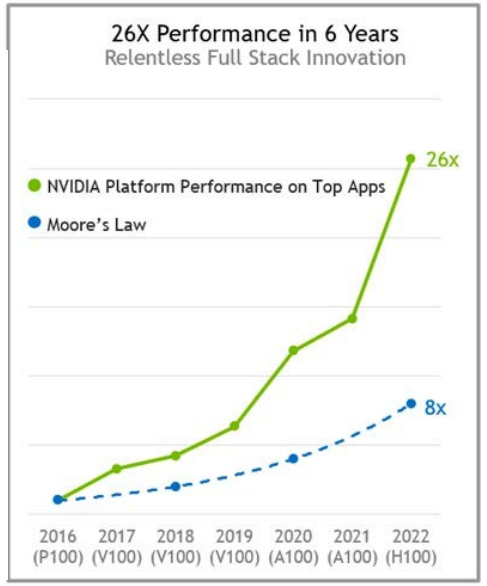
Data-driven AI

AIGC and AI for Science are both selected in Science’s 10 Breakthrough of Year 2022

Needs for Computing Power surpass the Moore's Law

	GPT 3.5	GPT 4
Number of parameters	175 billions	1.8 trillions
Type of inputs	Text	Text Images
tokens	4,096	32,786
#words at once	~3,000	~ 24,000
GPUs	1,000 A100	25,000 A100 GPU
Training time	30 days	90days
Cost	5 mil US $	63 milUS $
Utilization	/	32-36%





From 2016 to 2022, model size is growing 10^5 times larger, while the computing perf is growing only 26X.

Computing power and memory bw required for training increase exponentially in model size. How to design an effective system for AI?

Outline

Part I: Challenges in the Design of Effective Systems

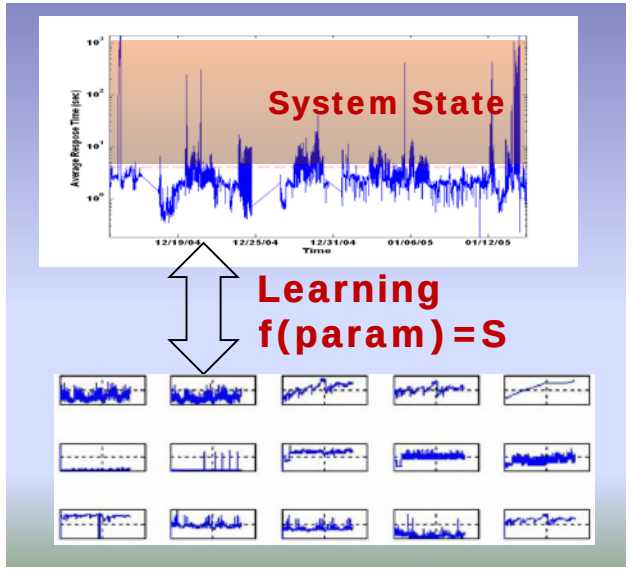
Part II: Recent Experience with Effective Systems for AI

Part III: AIGC and Foundation Model's Impact

Thanks to the teams in University of Macau, and Shenzhen Institute of Advanced Technology, including K. Ye, Y. Wang, X. Gao, C. Gao, J. Zhao, S. Wang, H. Xu, M. Xu, L. Li, S. Luo, X. Li, K. Wang, C. Lu, W. Chen

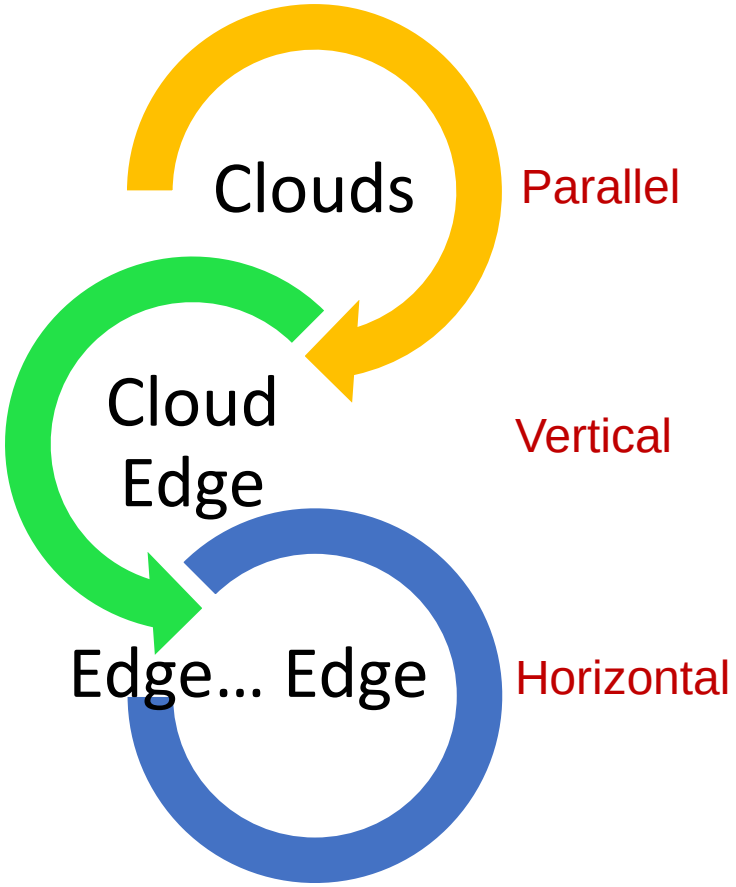
Interplay of Systems and AI

- Systems for AI: using systems (parallel/distributed /cloud/edge/…) to support efficient model training and inference in parallel
  - cluster and cloud, to be shared for resource utilization
  - cloud for model inference, handling millions of queries in real time
- AI for Systems: using AI tools, in contrast to optimization, to config/schedule/ manage/diagnose/… systems
  - Due to uncertainty, sometimes unpredictable relationship between resource and perf



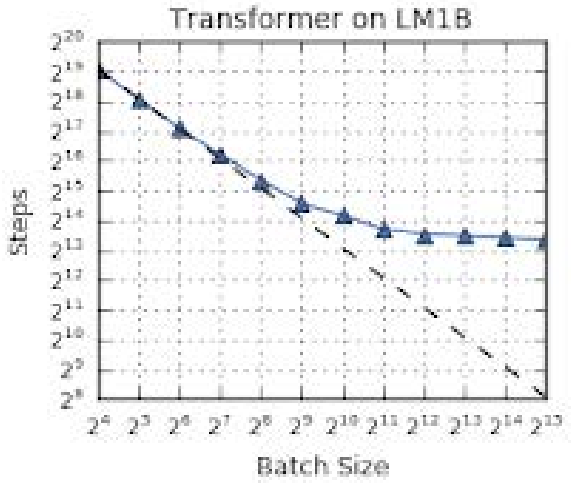
Examples of Effective Systems for AI

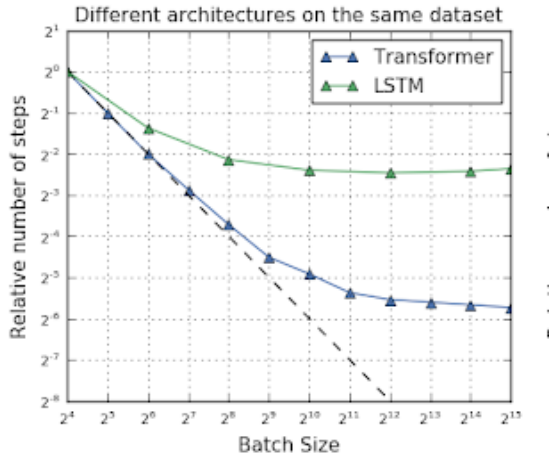
- Parallel Learning in Clouds
- Cloud-Edge Collaboration for Learning (Vertical)
- Cooperative Learning among Peers (Horizontal)



Parallel Learning in Clouds

- Parallel Training
  - Input-data partitioning (data parallelism, SPMD): GPUs run the same code (e.g. SGD optimization) on different parts of dataset
  - Model partitioning (model parallelism, pipelining): Multiple GPU working simultaneously on different parts of a model for an input
- Need to revisit performance issues
  - Load balancing, data locality, overlapping computation with comm, comm efficiency, etc





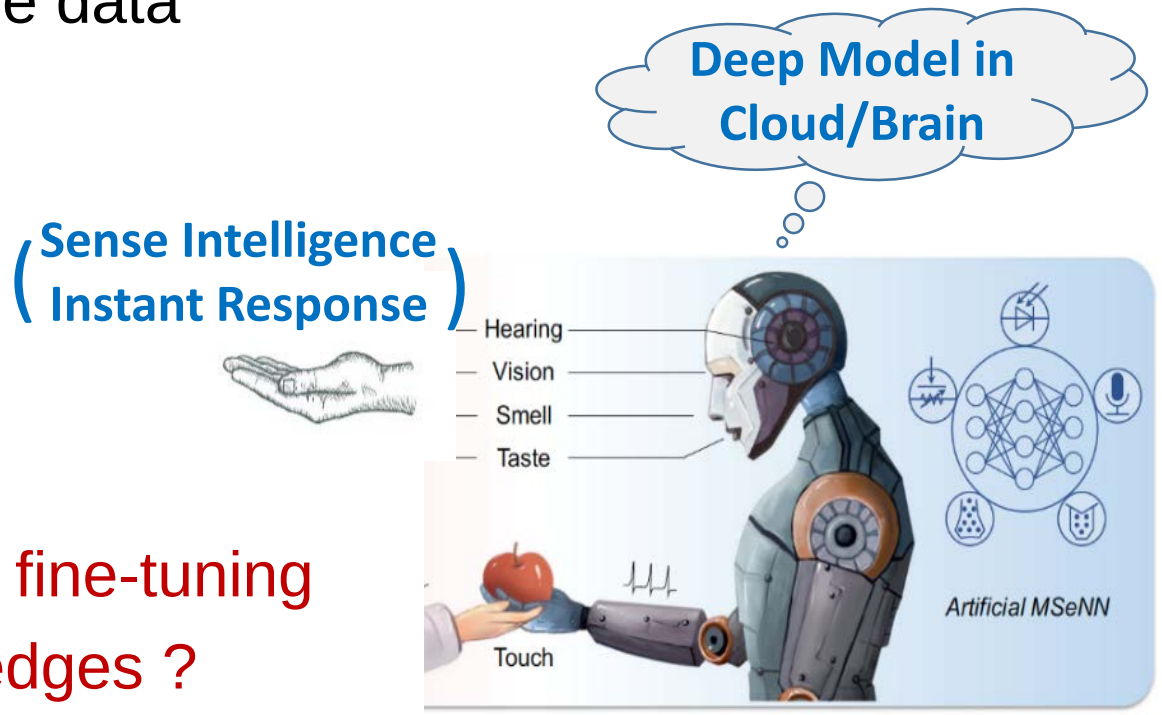
Data Parallelism by Batch Size, saturates quickly. (Google,2019)

Parallel Learning in Clouds (cont')

- Online Model Inference
  - Handle queries from millions of chatGPT users in real-time
- Both training and inference tend to be run in clouds to leverage the cloud resource flexibility (e.g. chatGPT in MS Azure).
- Resource management is a key to success of parallel learning in clouds. How to schedule AI tasks in clouds so as to maximize overall throughput and meanwhile satisfying individual SLO regarding latency?

Cloud-Edge Collaboration for Distributed Learning

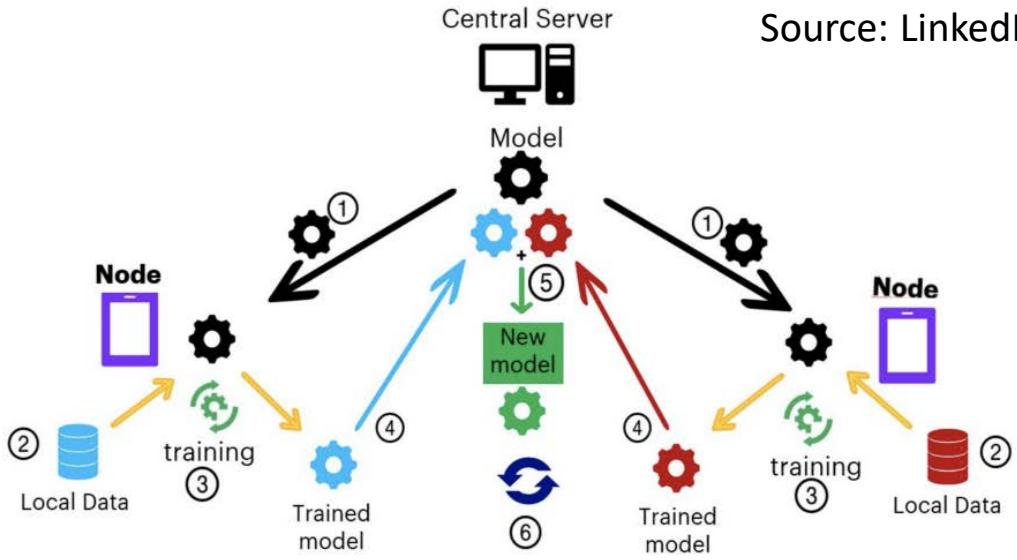
- Edges often located in proximity to users and data, but with limited resources. It provides capability of sensing and interacting the world.
- Cloud-Edge collaboration aims to provide instant response, without need for communication of large volume data
- It bears resemblance to the layered social structure, as well as our body's layered intelligence/response.



- How to split, compress models, or fine-tuning models for resource constrained edges ?

Cooperative Learning among Peers

- Leverage the learning capabilities of peers by distributed optimization or knowledge fusion over different datasets to train a model collectively (horizontally)
- Clouds/clusters are mostly homogeneous, but the peers may not, and resource-constrained. Also, the data is not necessarily balanced.
- How to apply cooperative learning ideas to domains of variance (non-IID data) with heterogeneous systems?



Source: LinkedIn

Outline

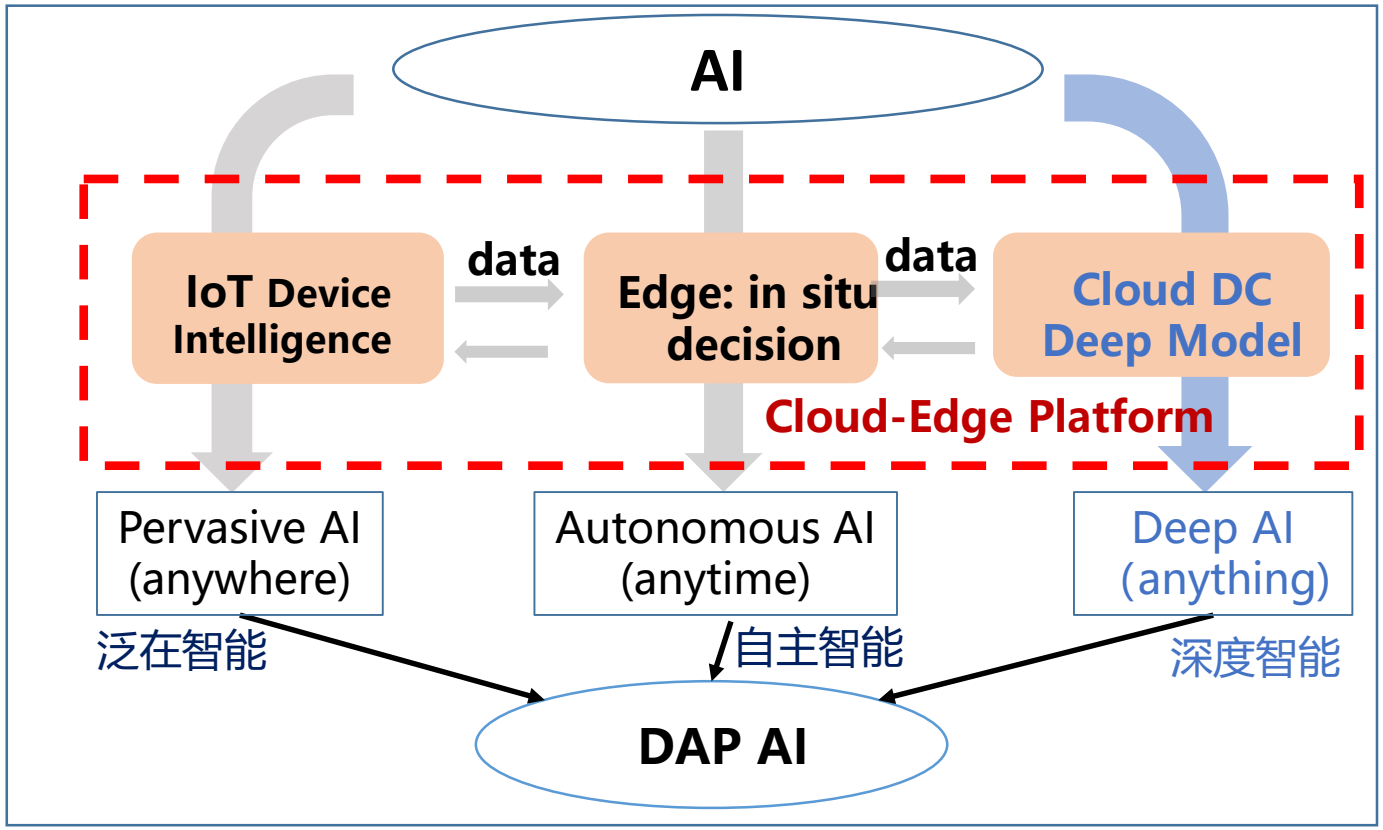
Part I: Challenges in the Design of Effective Systems

Part II: Recent Experience with Effective Systems for AI

Part III: AIGC and Foundation Model's Impact

Early Experience in the Design of Effective Systems

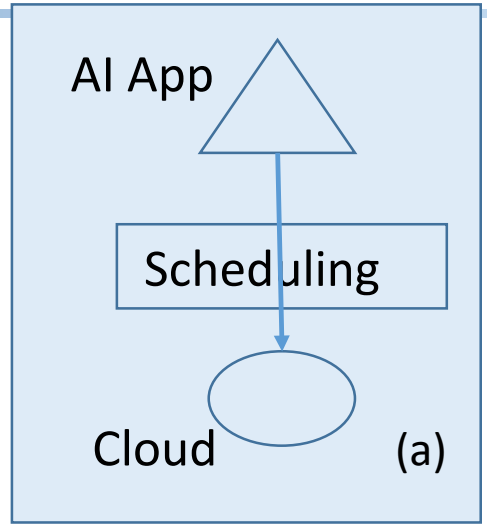
Effective Systems for Deep, Autonomous, Pervasive AI



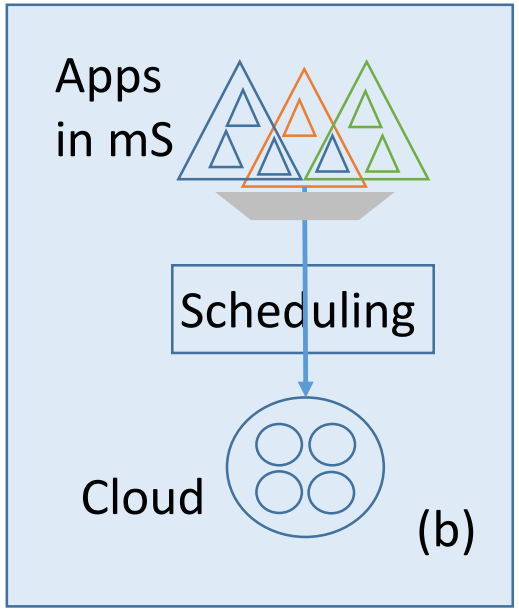
- Parallel Learning in Clouds (ASPLOS'23, EuroSys'2023, SC'2023)
- Cloud-Edge Collaboration for Distributed Learning (ICLR'2019, NeurIPS'2019)
- CooperativeLearning among Peers (CVPR'2023, ICML'2020)

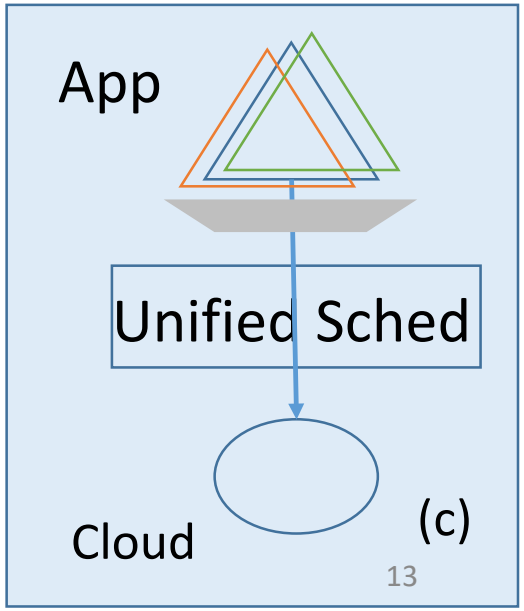
Parallel Learning in Native Clouds

- Appl in the form of microservices, and the cloud in serverless arch so as to leverage the opportunity of fine-grained resource management



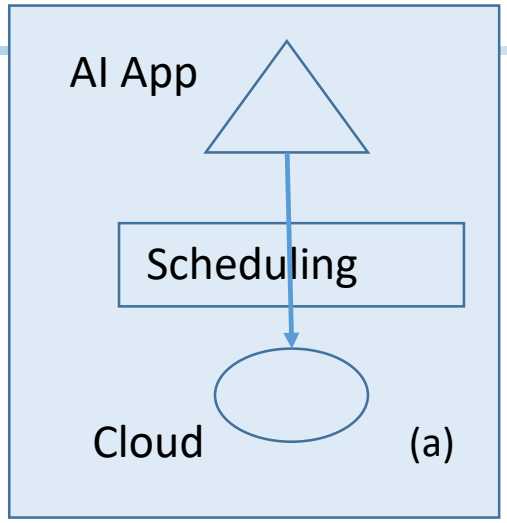
- Current research focus:
  - Scheduling for deep learning (a)
  - Scheduling for microservices (b)
  - Unified scheduling for both batch & online applications (c)

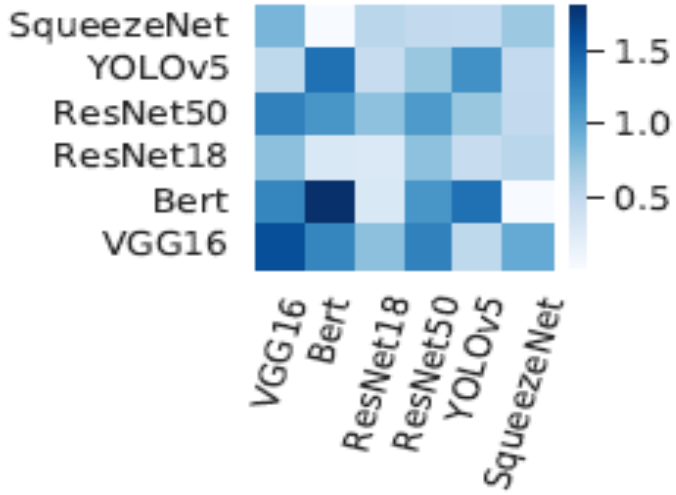




Deep Learning on Dedicated Clusters

- GPU Schedulers for Deep Learning (DL)
  - Packing multiple tasks on the same GPU via time-sharing or spatial-sharing multiplexing to improve the GPU utilization.
- In space sharing (multi-tenant), suffer from underutilization and long queuing delay
  - Up to 60% GPUs are <10% utilization (MS Azure, 2019)
  - Long queuing delay, sometimes intolerable !
- In time sharing, simple multiplexing would cause severe interference among tasks, leading to significant slowdown.

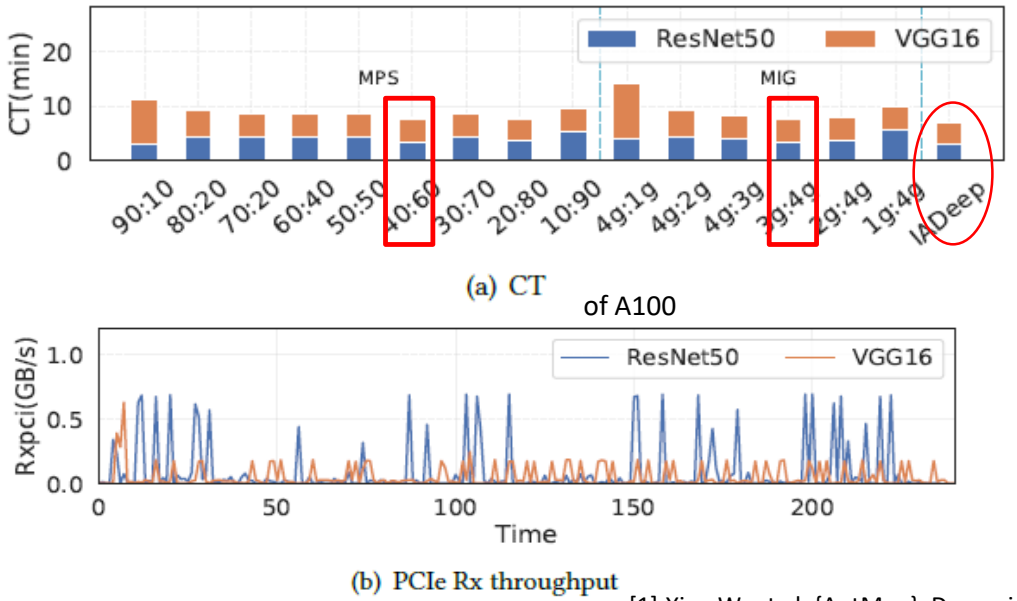




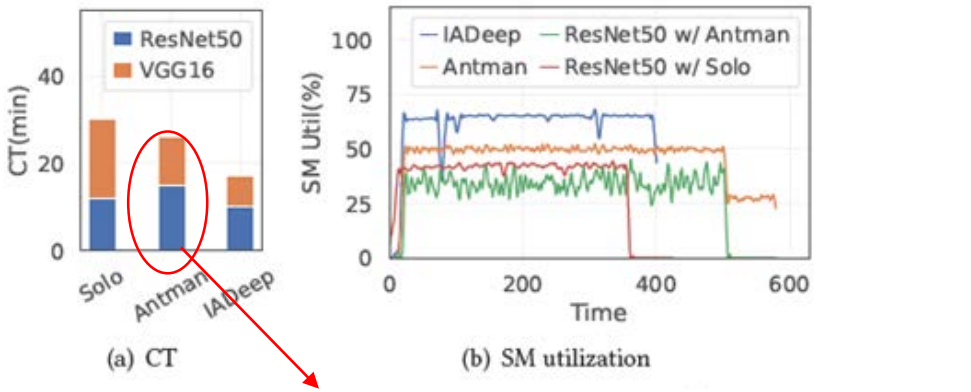
Interference of co-located tasks

Inefficiency due to Low-Level Multiplexing

- Hardware-level solutions
  - **MPS**[2] and **MIG**[3] of NVIDIA
  - Both help reduce interference by explicitly isolating SM and mem resources among tasks.
  - But neither can deal with critical PCIe BW, which prone to be bottleneck in DL



- Kernel-level solutions to fine tuning launching order of co-located tasks
  - **Antman**[1] (Alibaba) with time-sharing
  - OS-level solutions require significant custom modification for various DL
  - Also, tasks cannot perfectly pad each other' idle GPU cycles statically

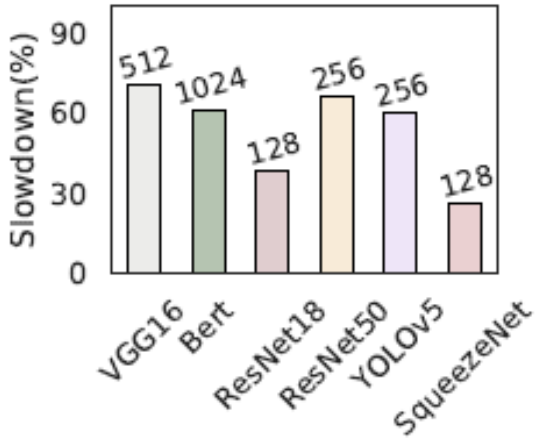


Antman outperforms dedicated config by 15%

[1] Xiao W, et al. {AntMan}: Dynamic Scaling on {GPU} Clusters for Deep Learning[C], OSDI 2020
[2] NVIDIA Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html
[3] NVIDIA Multi-Instance GPU. https://www.nvidia.com/en-us/technologies/multi-instance-gpu

IADeep: Middleware Solution for Online Learning Tasks

- Opportunities
  - Choose different types of learning tasks to multiplex on a GPU can mitigate interference.
  - Co-locate suitable number of tasks to balance the waiting time and training time
  - System wide: GPU mem, PCIe BW



Slowndown of VGG16, running together with diff apps

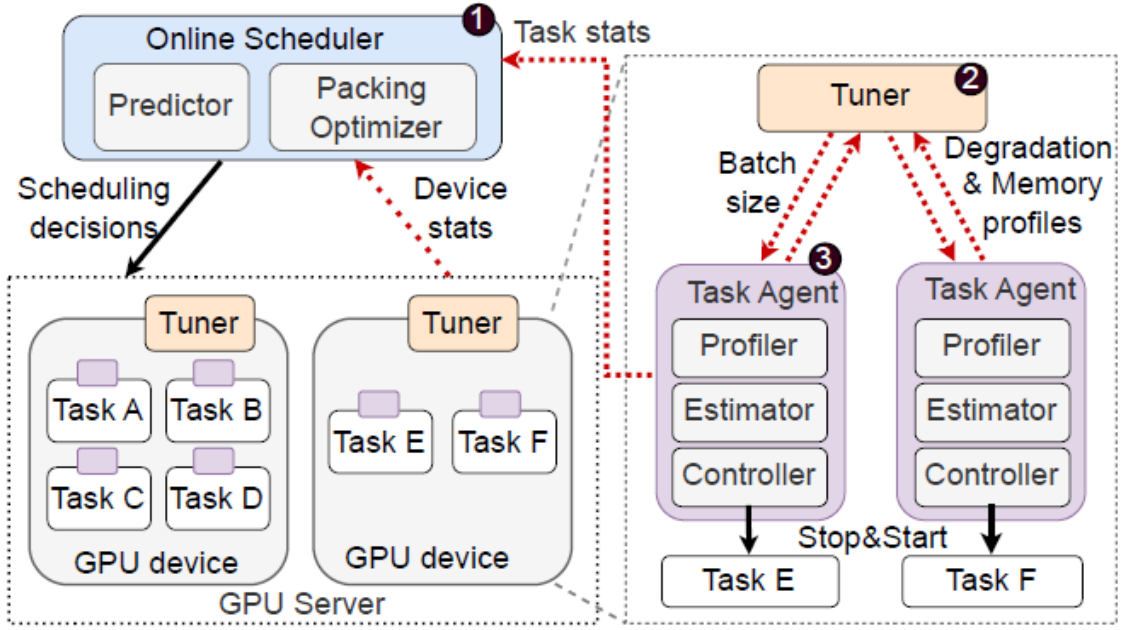
- Challenges
  - Task config heavily influences interference, as well as task training progress
  - Vast search space of task configurations
  - Closely couple between adjusting task config and designing task placement policies
  - Config and placement decisions are made in realtime for newly arrived tasks

IADeep System Design

① **Online Scheduler**: Find the optimal device to place the new arrival task

② **Tuner**: Tune configurations (*batch sizes*) to mitigate the interference.

③ **Task Agent**: Update the configurations for each co-located task.



x_d^k is a binary variable indicating whether task k is placed on device d

$$\min \sum_{d=1}^{D} \sum_{k=1}^{N} x_d^k \cdot \mathrm{PD}_d^k(\star^k)$$

$$\text{s.t.,} \sum_{k=1}^{N} x_d^k \cdot \mathrm{PM}^k(\star^k) \le \mathrm{C}_d, \forall d,$$

$$\sum_{d=1}^{D} x_d^k = 1, \forall k, \text{ and } x_d^k \in \{0,1\}.$$

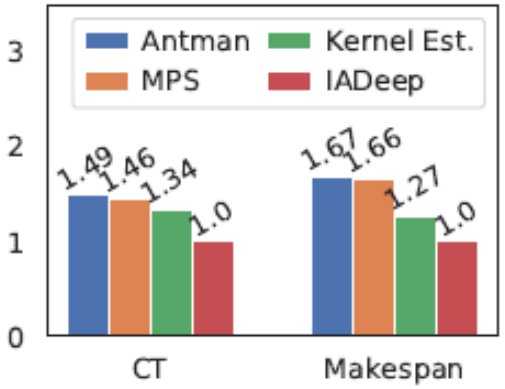
Minimize the overall **performance degradation** (PD) of all co-located tasks on the same device.

$$\mathrm{PD}_d^k(\star^k) = \mathrm{IF}_d^k(\star^k) \times \mathrm{DT}^k(\star^k) \times 1/\mathrm{EFFICIENCY}_t^k(\star^k)$$

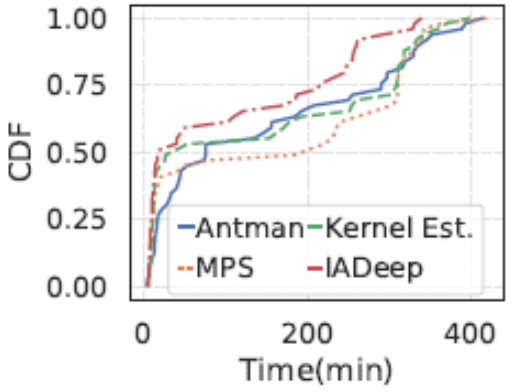
W. Chen, Z. Mo, H. Xu, K. Ye, and C. Xu, IADeep: Interference-aware Multiplexing for Deep Learning, SC'2023

Experimental Results: End-to-End Performance

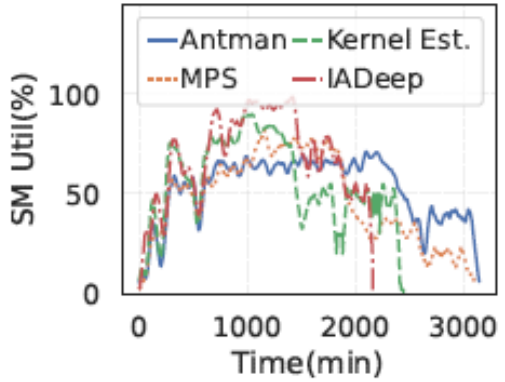
- Full impl on Kubernetes, and evaluation using Microsoft Phillly Traces
- Completion Time (CT), makespan and GPU utilization
  - IADeep reduces the overall CT and makespan by up to 49% and 67% respectively, compared to Antman[1], MPS[2] and Kernel Est[3].
  - IADeep can obtain up to 29% and 31% higher SM and memory utilization.



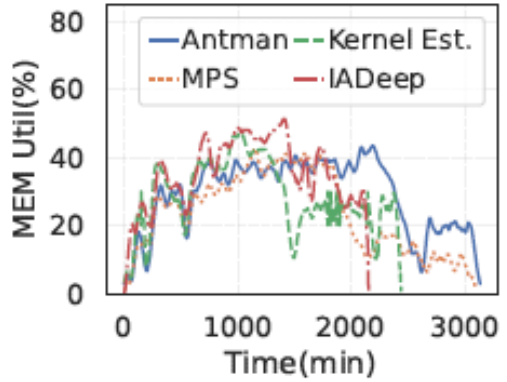
(a) Average CT and makespan



(b) CDF of CT



(a) Avg. SM utilization

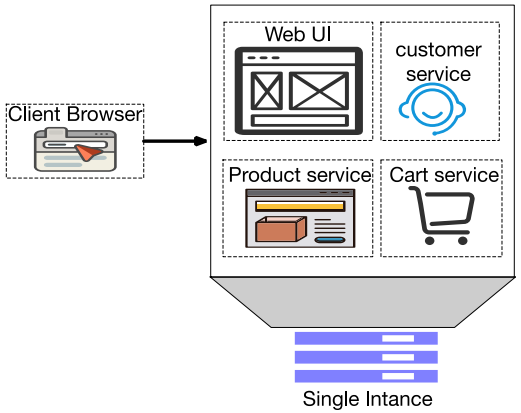


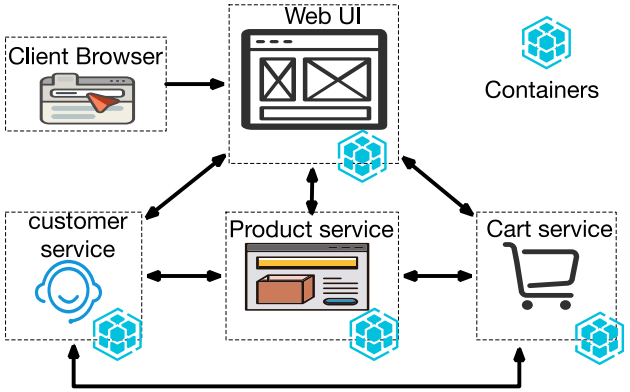
(b) Avg. memory utilization

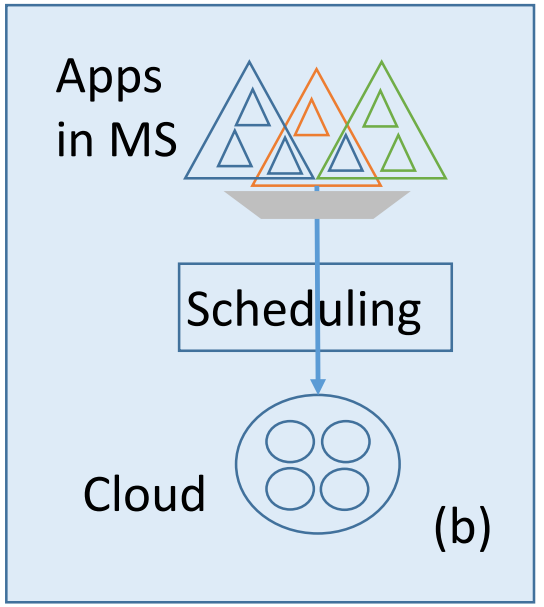
[1] Xiao, Wencong, et al. "AntMan: Dynamic Scaling on GPU Clusters for Deep Learning." In Proceedings of OSDI. 2020.
[2] NVIDIA Multi-Process Service. https://docs.nvidia.com/deploy/mps/index.html
[3] Xu, Xin, et al. "Characterization and Prediction of Performance Interference on Mediated Passthrough GPUs for Interference-aware Scheduler." *In Proceedings of HotCloud*. 2019.

Microservices (MS) in Clouds

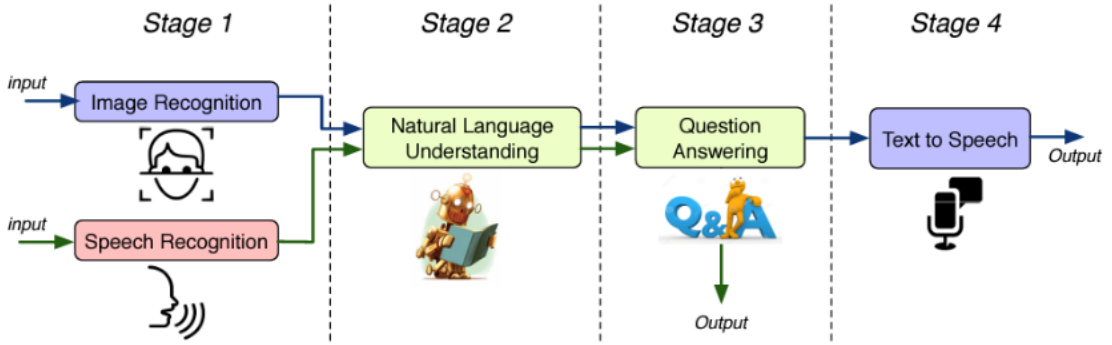
- Monolithic app is divided to multiple MSs







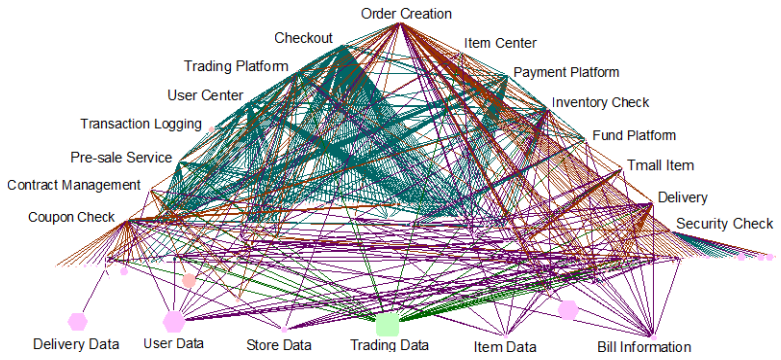
- Model inference in the form of microservices: ease of management, maintenance, and update



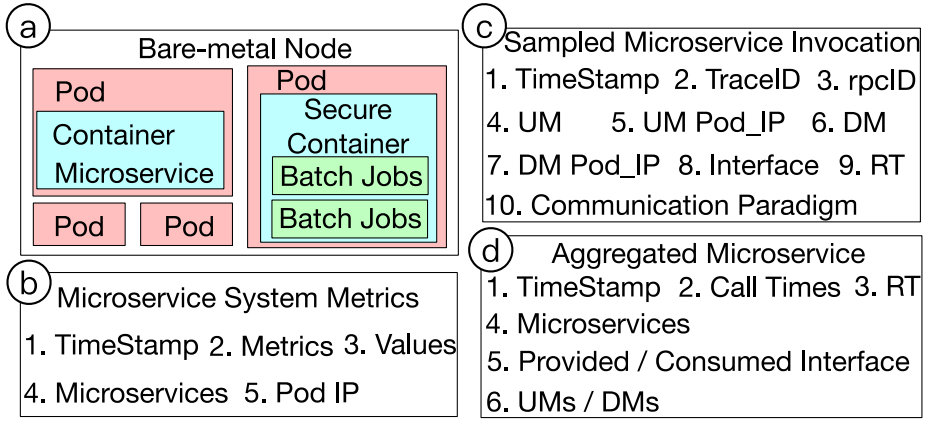
(Stanford, eurosys'2020)

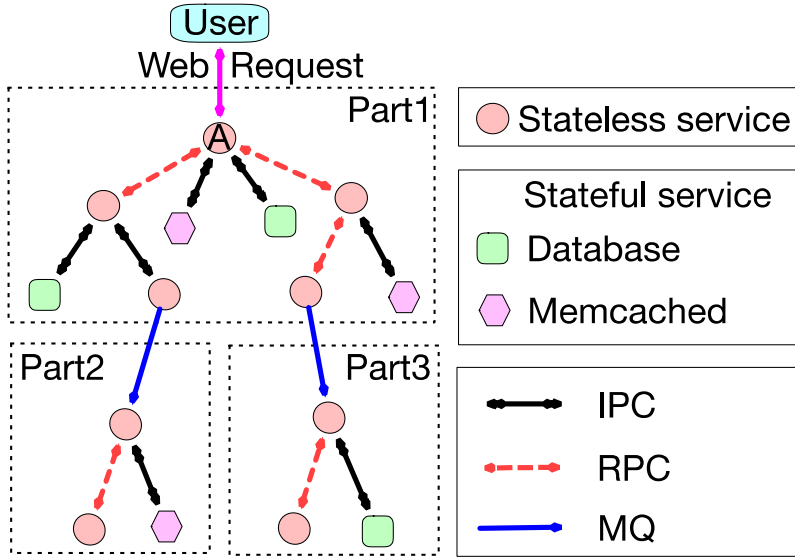
Traces Analysis: Overview of Alibaba traces

- The scale of MS is large in Alibaba.
  - 20000+ MS
  - 10 billion calls between MSs within 7 days
  - Traces contain diverse metrics
- 100+G traces has been released.
  - https://github.com/alibaba/clusterdata



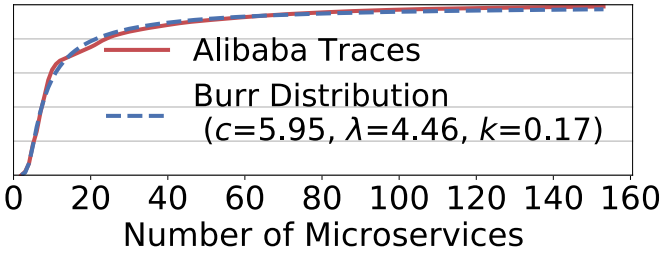
Complicated MSs dependencies graph





Characteristic of Microservices

- Key concept: Dependence Graph (DG)
- Scale of DG, follows a heavy-tail distribution
  - A DG can involve 1000+ MS, and depth of DG can be 100+.
- A DG has a tree-like structure
  - MS DG has more scatter components than gather components.
- A small percentage of MS are hot-spots in DG.
  - 5% of MSs are shared among more than 90% of applications
- MSs form highly dynamic dependencies in runtime.
  - DGs of an online service can be categorized into multiple clusters
  - Complicated DG makes it a challenge to resource management for MS

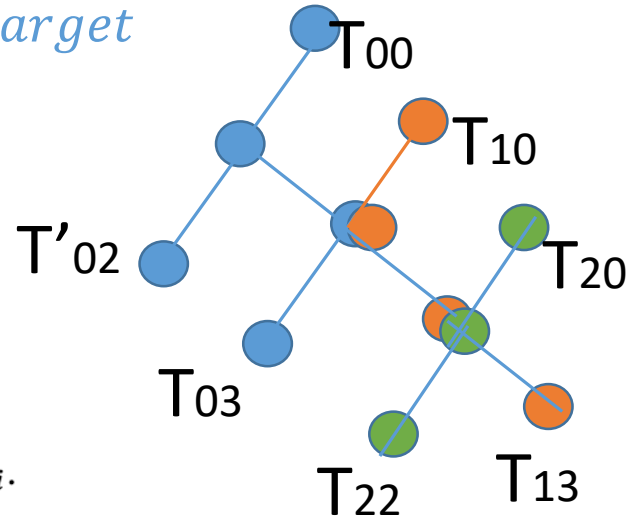


Out-degree	10% MS > 5
In-degree	90 % MS = 1

S. Luo, et al., Characterizing Microservice Dependency and Performance: Alibaba Trace Analysis. SoCC'2021 (best paper award)

Scheduling for Shared Microservices

- Problem Statement
  - Given a set of apps in mServices, each with DG and info about their sojourn time, resource demand and latency, and overall SLA requirements
  - Schedule/deploy mService based on allocated *Optimal Latency Target*
    - Improve resource efficiency, and satisfy SLA requirements



- Optimization problem: $\min_{\vec{n}} \sum_{i=1}^{N} n_i \cdot R_i, \quad \text{s.t.} \quad \text{latency}_k(\vec{n}) \le \text{SLA}_k.$
- Optimal latency target in closed-form:

Latency Target = $\sum \left(a_i \frac{\gamma_i}{n_i^o} + b_i\right)$

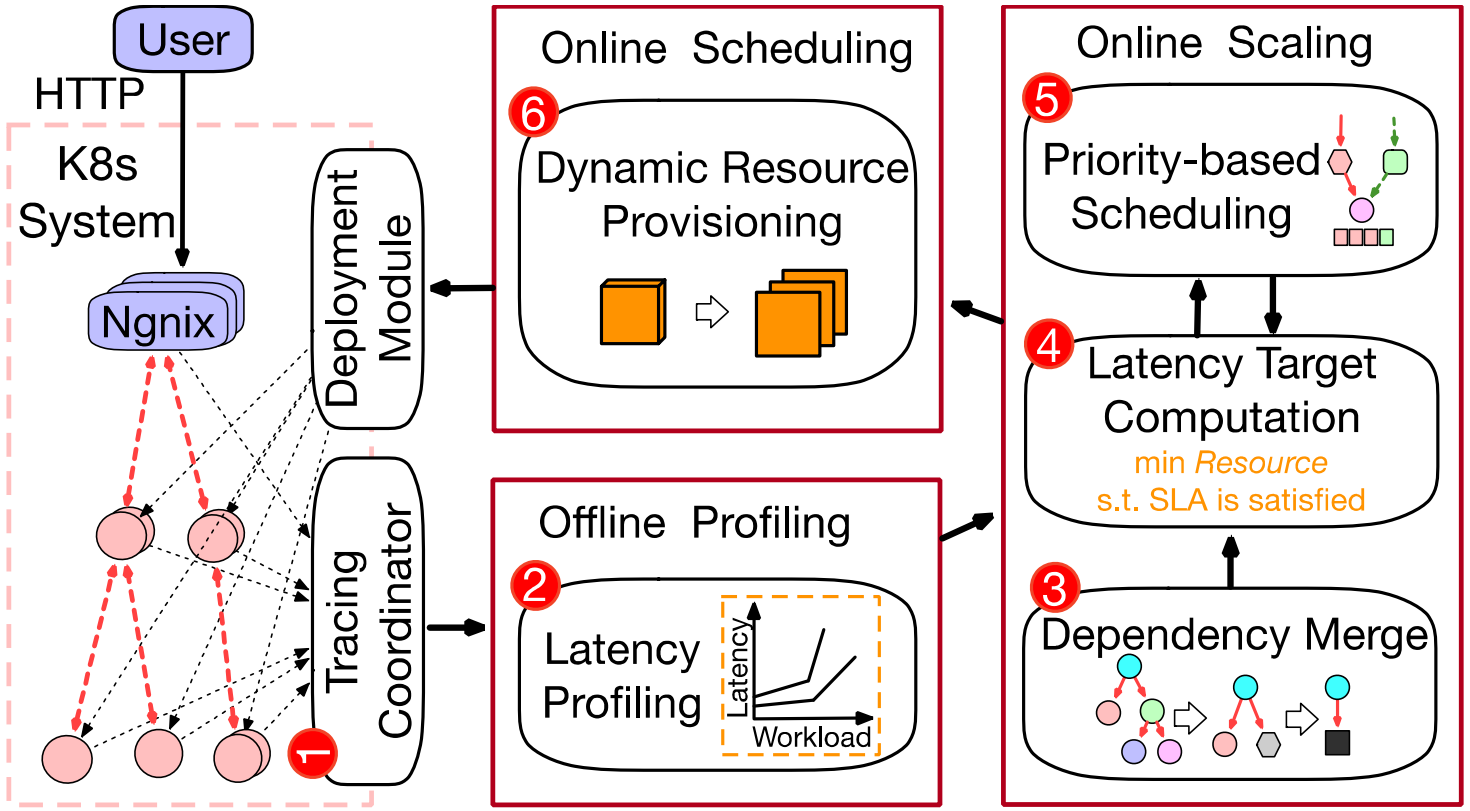
$$a_i \frac{\gamma_i}{n_i^o} + b_i = \frac{\sqrt{a_i \gamma_i R_i}}{\sum_{i=1}^{N} \sqrt{a_i \gamma_i R_i}} \left(\text{SLA} - \sum_{i=1}^{N} b_i\right) + b_i.$$

a_i : Slope of MS i in latency model. γ_i : Workload of (shared) MS i.

- [Theorem] The resource usage yielded by the optimization problem is no larger than that due to FCFS sharing or no-sharing approaches (ASPLOS'2023)

Erms for Shared MS: Design of System

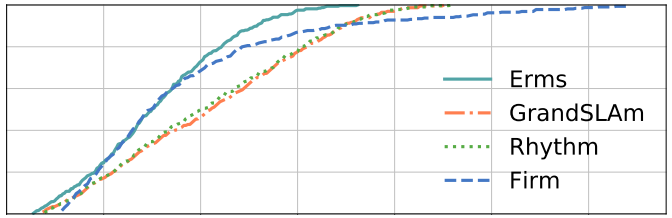
- System arch of Erms (Efficient resource management system)

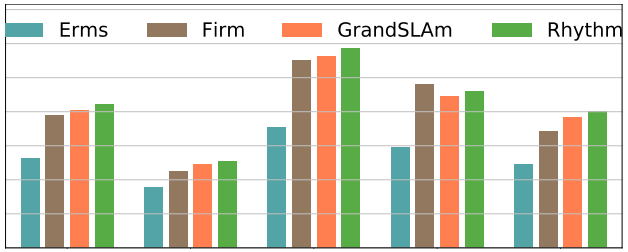


S. Luo, et al., Erms: Efficient Resource Management for Shared Microservices with SLA Guarantees. ASPLOS'2023 (ACM Trans. on Computer Sysgtems, 2023)

RM for Shared MS: Evaluation on Cluster

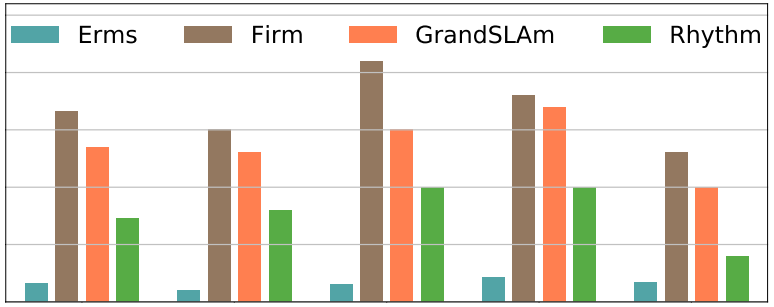
- Replay Alibaba workloads for trace-driven simulation, using 1000 services and 20k+ mservices in 12-h period, on a cluster of 14 nodes, each with 64 CPU cores
- Allocated resource: Erms saves about 60% of containers.

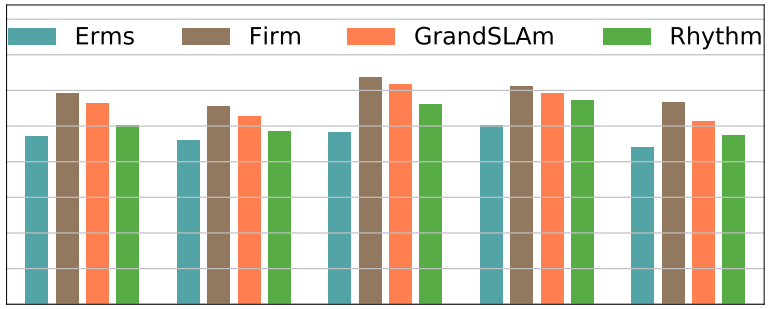




GrandSlam, Eurosys'2019
Rhythm, Eurosys'2020
Firm, OSDI'2020
Erms: our approach

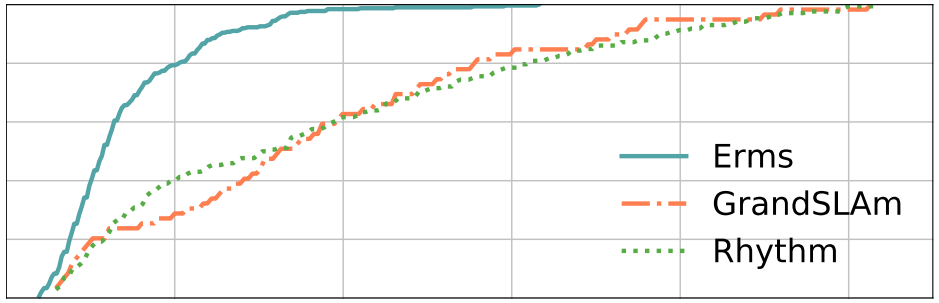
- E-t-E response time: reduce reduces SLA violation probability by 5x and improves performance by up to 10%



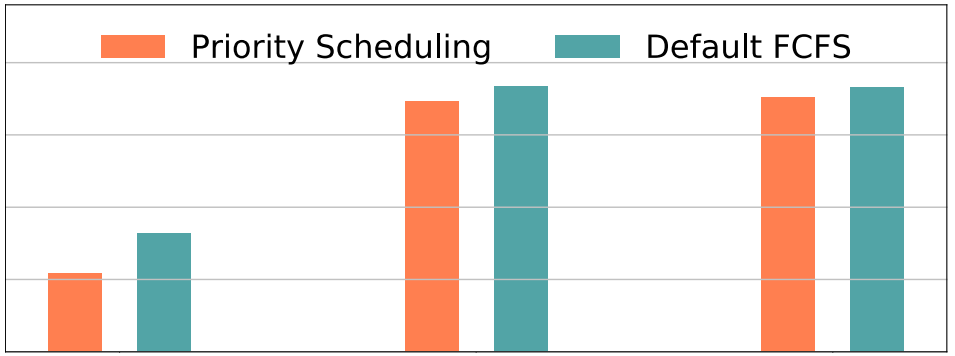


RM for Shared MS: Trace-driven Simulation

- End-to-End performance



Erms reduce # allocated containers by 1.6x.



Latency Target Comp save resource usage by 1.1x.
Priority Scheduling reduce resource usage by 50%.

- Scalability of Erms
  - For largest graph with 1000+ microservices, the computational overhead is 300ms.
  - Overhead of resource provisioning for 1000 containers across 5000 hosts is 200ms.

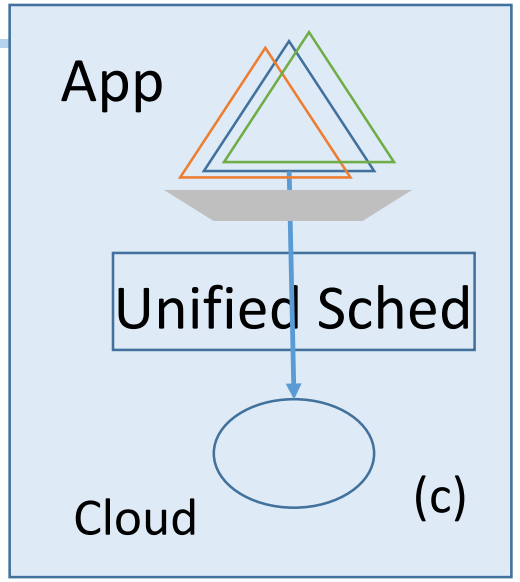
S. Luo, et al., Erms: Efficient Resource Management for Shared Microservices with SLA Guarantees. ASPLOS'2023

Software available online, functional, and results are reproducable

Unified Sched for Batch and Online Jobs

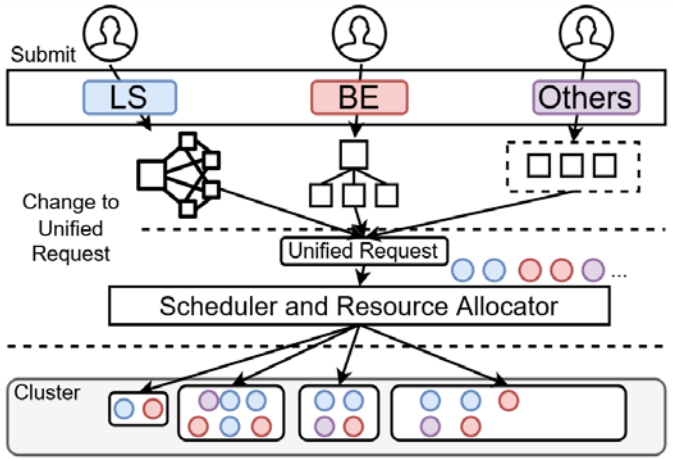
1st gen: hybrid scheduling multi-tenant clouds (--2016)

- Reserve resources for batch and online, and schedule separately
- Sigma of Alibaba for long-running latency-sensitive jobs; Fuxi for batch jobs with many small task with complex dependeces
- ~10-15% utilization, unacceptable!



2nd gen: unified scheduling (2016-)

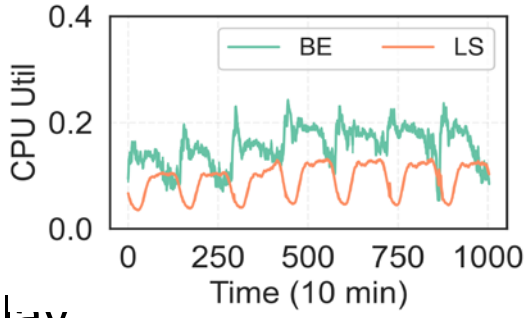
- Unified scheduling for all applications, in the form of unified requests, and manage all resources in a consistent way
- Google Borg (Eurosys 2020), Facebook Twine (OSDI 2020), Alibaba Aliware (2021)
- Overall utilization increases to ~30% on average



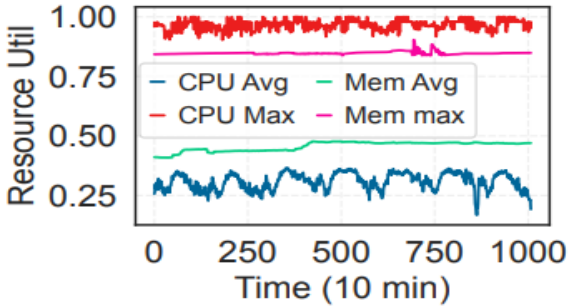
Alibaba unified scheduling framework

Characterization of Unified Scheduling in Alibaba Cloud

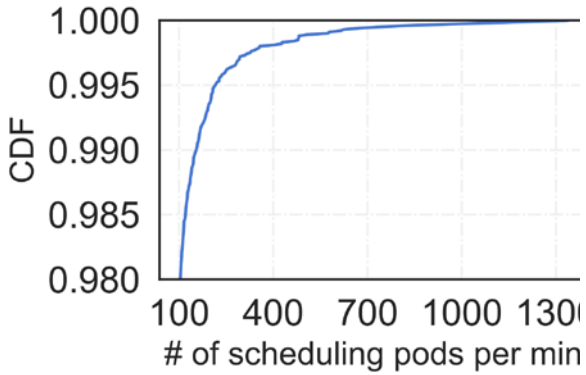
- Over 1 million tasks (pods) deployed in one Alibaba datacenter (2 clusters, each with 6000 physical hosts) for a 8-day period
- Cluster resource utilization
  - Filling valleys and shaving peaks
  - High max (>90%) but low ave (~40%)
- Scheduling efficiency
  - Heavy-tailed distribution in scheduling delay
  - Large number of pods experienced long delay (10% of BE pods had 100 seconds delay)

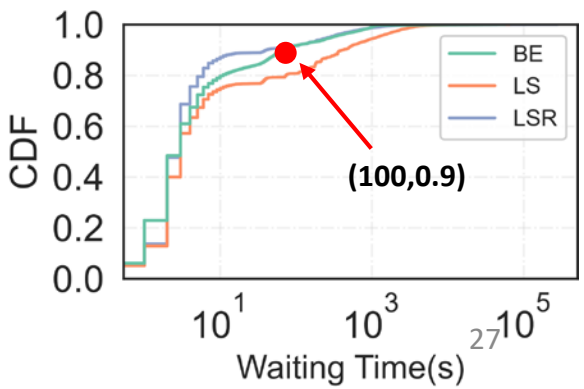


a. Avg Pod CPU util



b. Host resource util



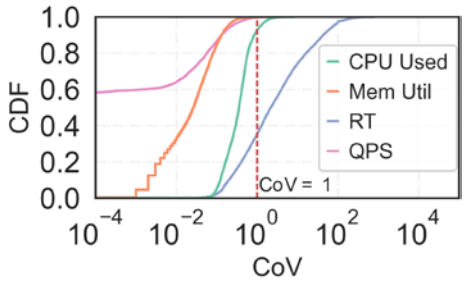


C. Lu, et al, Understanding and optimizing workloads for unified resource management in large cloud platforms, EuroSys'2023

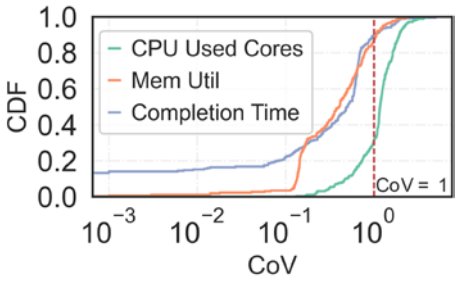
How to Optimize the Unified Scheduler?

- Perf Metrics: key to scheduling
  - CT works well for batch tasks. However, RT NOT a good perf indicator for LS services about resource usage
- PSI (Pressure Install Info.) to measure the ratio of waiting time of LS services (OS-level) to sampling interval (appl-level)
  - PSI shows strong correlation with host CPU utilization and pod CPU utilization

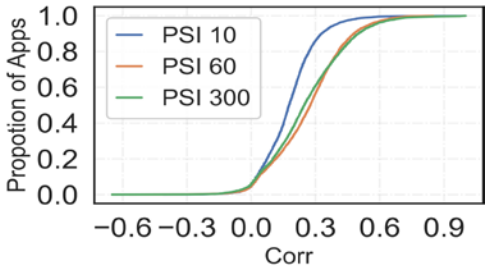
CoV: std deviation/average， pod resource usage vs perf



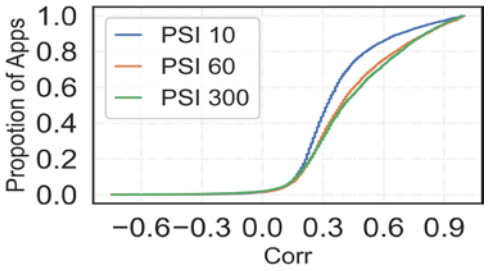
(a) Latency-sensitive (LS) services



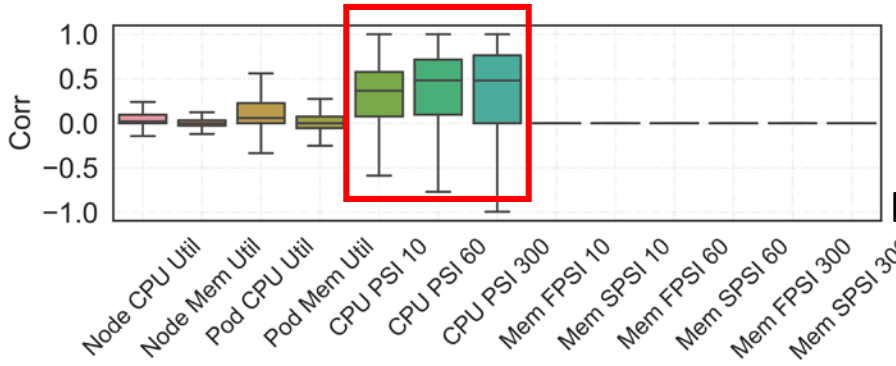
(b) Best-effort (BE) applications



(a) Host CPU utilization



(b) Pod CPU utilization



Distribution of correlation between RT and metrics

How to Optimize the Unified Scheduler

- Resource Usage Predictor
  - Predict the resource usage of the host as

$$Util = \sum_{p,q \in host} \left(\max_{p \in A_p, q \in B_q} \max_{t} \frac{\max\left(RU_p(t) + RU_q(t)\right)}{\max\left(RU_p\right) + \max\left(RU_q\right)} \right) * \left(\max\left(RU_p\right) + \max\left(RU_q\right)\right)$$

- Interference Predictor
  - For LS applications: $\mathrm{PSI}_p^t = f_{S_i}^o\left(\mathrm{C}_p^t, \mathrm{M}_p^t, \mathrm{C}_h^t, \mathrm{M}_h^t, \mathrm{Q}^t\right)$
  - For BE applications: $CT_p = f_{B_j}^b\left(\mathrm{C}_p^m, \mathrm{M}_p^m, \mathrm{C}_h^m, \mathrm{M}_h^m\right)$

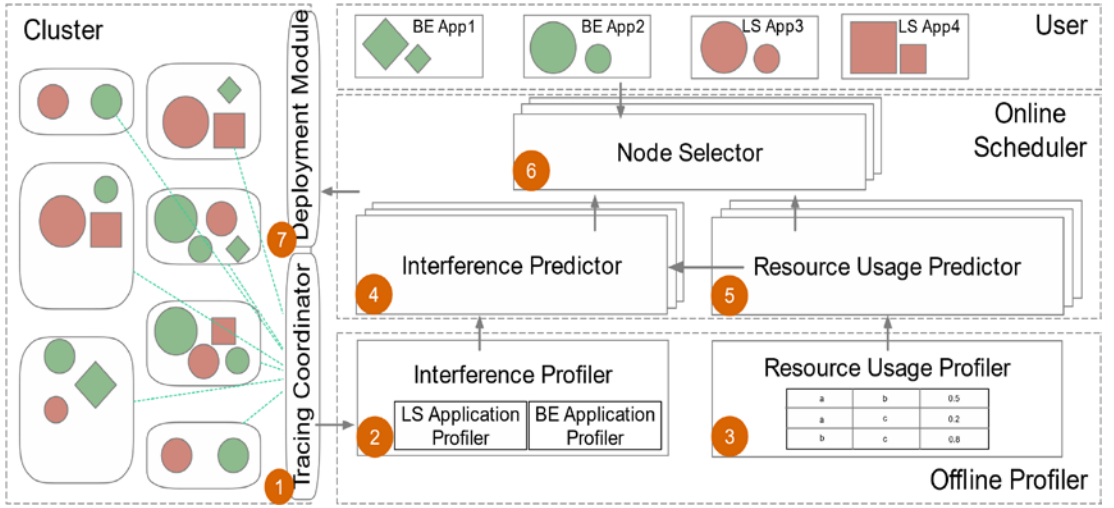
- Optimization framework

$$\max \sum_{h \in H} \mathrm{Uti}_h^C \cdot \mathrm{Uti}_h^M - \omega^o \cdot \sum_{p \in P_i, i \in I} \mathrm{PSI}_i^p - \omega^b \cdot \sum_{q \in P_j, j \in J} CT_j^q$$

$$\text{s.t.}, \mathrm{Uti}_h^C,\ \mathrm{Uti}_h^M \leq 1, \forall h \in H,$$

Resource Usage Predictor

Interference Predictor, based on PSI



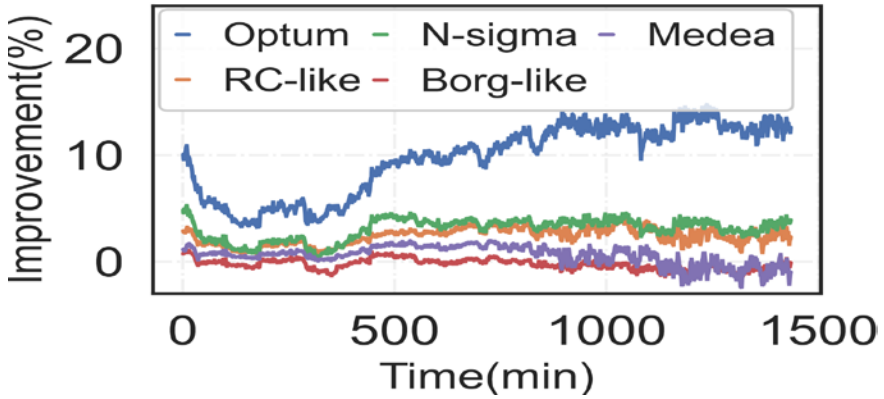
Optimization System Overview

See EuroSys'2023 for Details

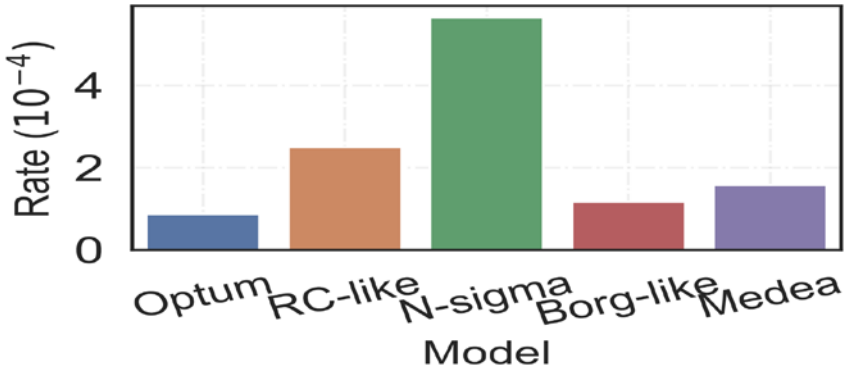
How the optimized unified scheduler works

See EuroSys'2023 for details

- Trace-driven simu: Alibaba load and Optum used pods data of 7 days to build model
- Improved resource utilization by 15% and the violation rate is very low

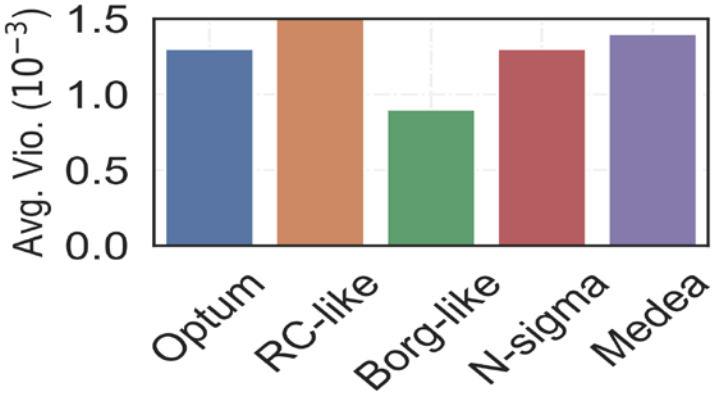


(a) Resource utilization

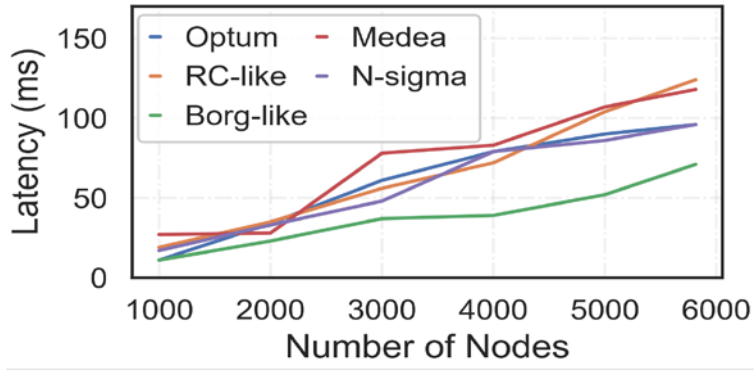


(b) Violation of resource usage

- LS pods have little PSI violation, and few BE pods violate the original completion time

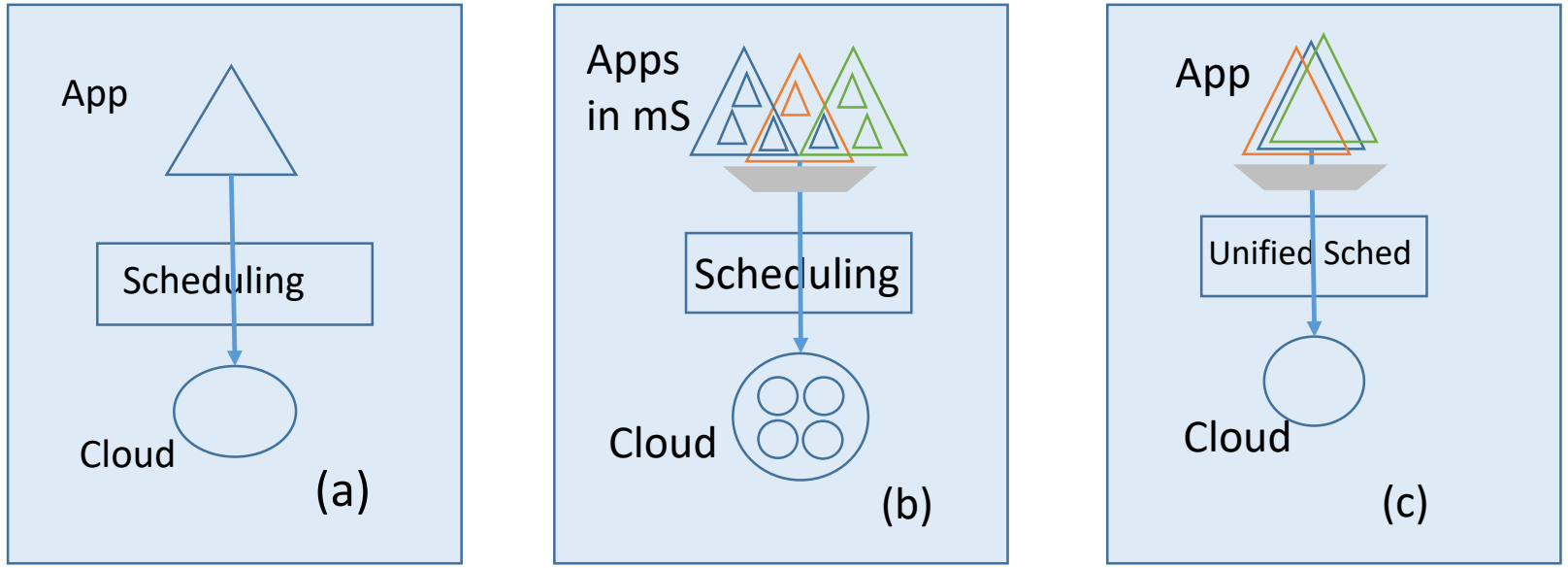


- High efficient: average scheduling latency is 96ms for each pod (python)



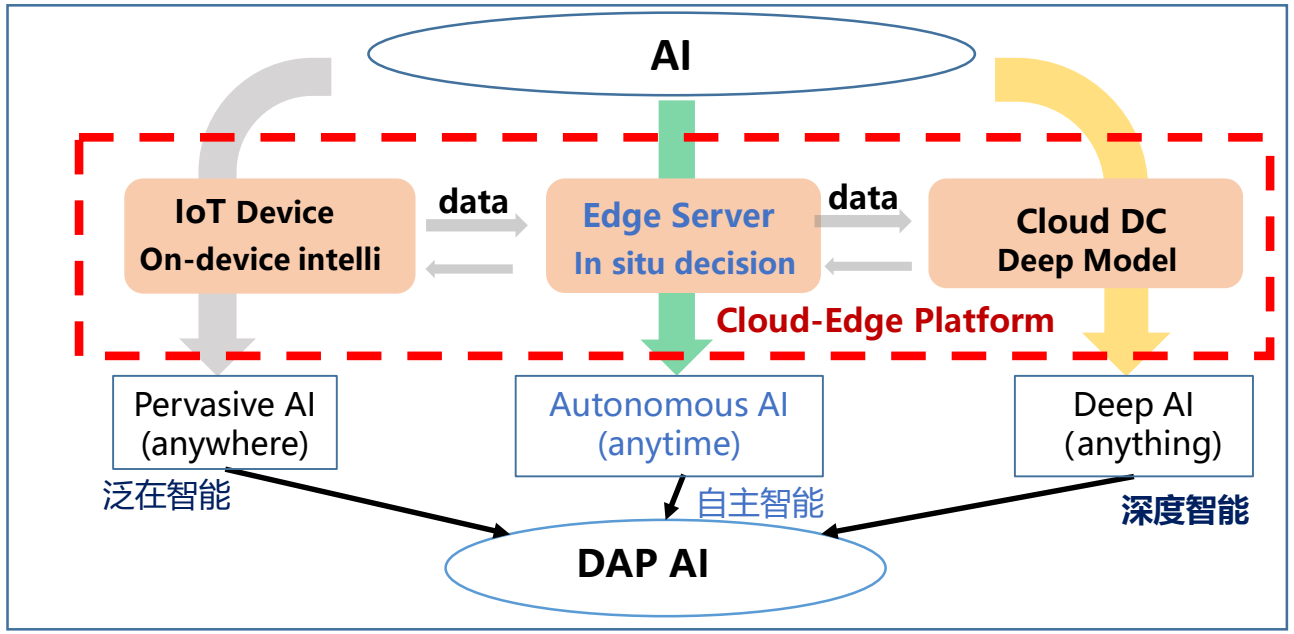
Summary of Parallel Learning in Clouds

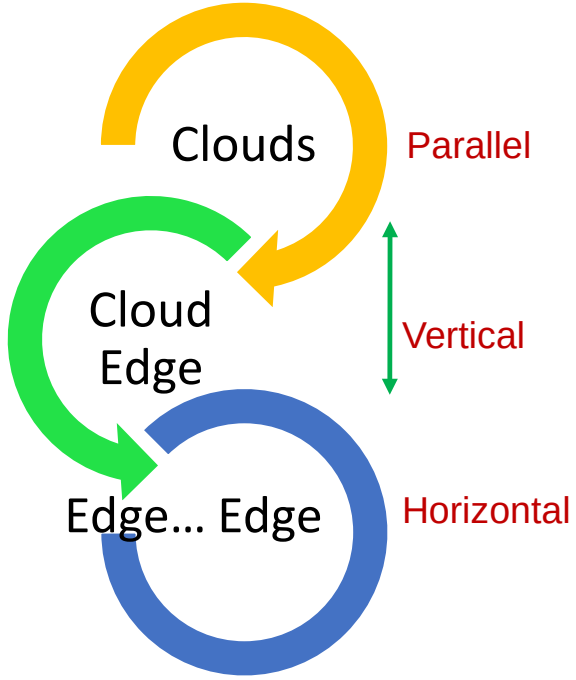
- Early Experience with Scheduling:
  - Scheduling for deep learning (SC'2023) (a)
  - Scheduling for microservices (ASPOLOS'2023) (b)
  - Unified scheduling for both batch and online appl (EuroSys'2023) (c)



Early Experience in Cloud-Edge Intelligence

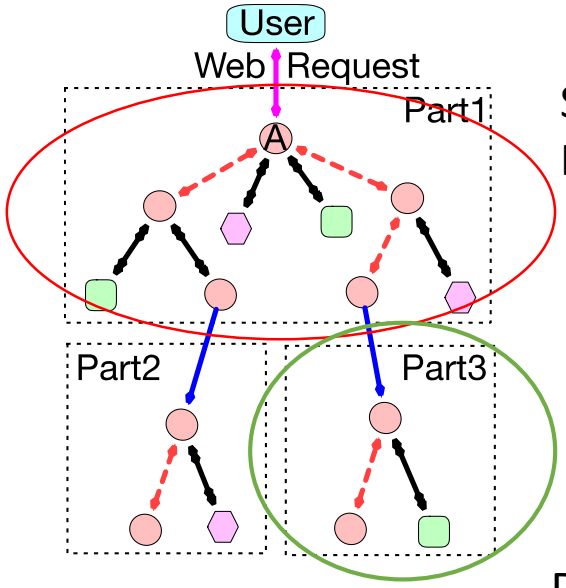
- Parallel Learning in Clouds
- Cloud-Edge Collaboration for (Vertical) Distributed Learning
  - Model Compression (ICLR'2019, NeurIPS'2019)
- Cooperative Learning among Peers (Horizontal)





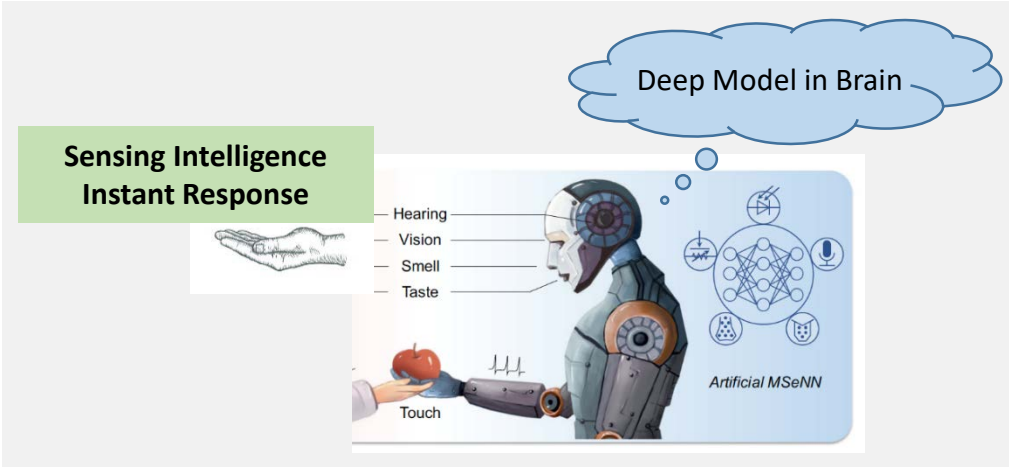
Model Splitting/Compression in Cloud-Edge Collaboration

- Microservice arch meets the needs of model splitting



Sensing/Interactive Intelligence in the front

Deep model in backend

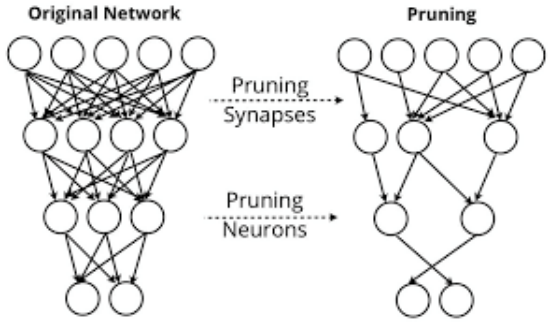


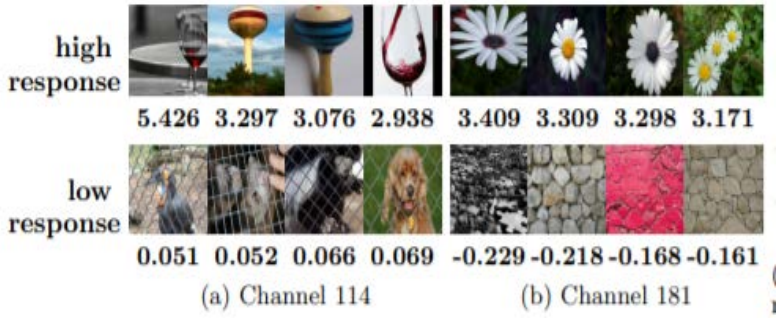
- Model compression is a must for constrained resources

Efficient Model Inference for Constrained Edge

Key Observation/Motivation:

Importance of features produced by deep models is highly input-dependent





Images to excite neurons of ResNet-18 model and outputs high/low weights

New Method: Feature Boosting and Suppression (FBS) to

predictively amplify salient channels and skip unimportant ones at run-time

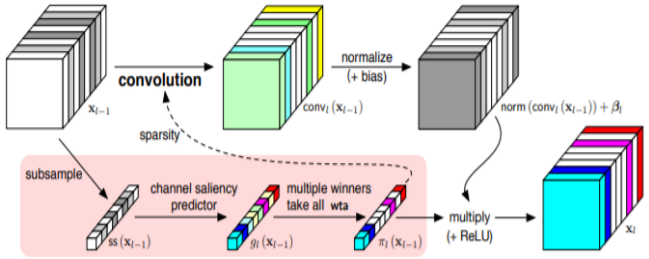
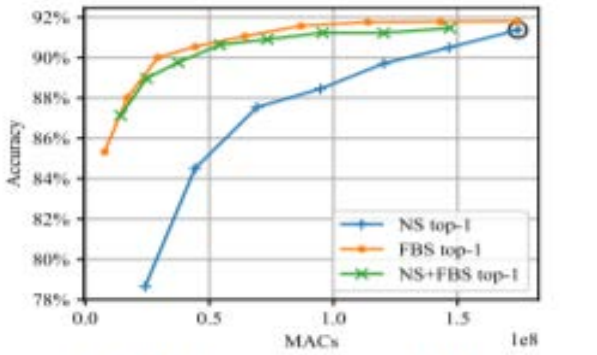
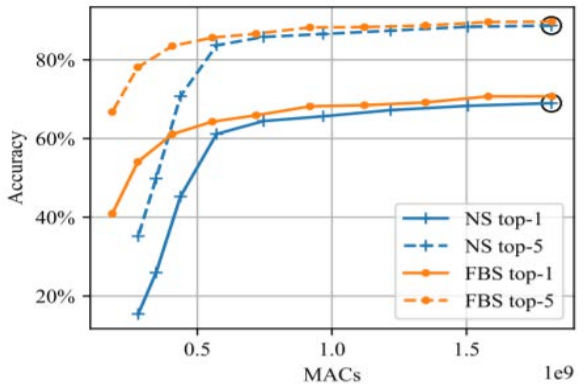


Figure 2: A high level view of a convolutional layer with FBS. By way of illustration, we use the l^{th} layer with 8-channel input and output features, where channels are colored to indicate different saliencies, and the white blocks (□) represent all-zero channels.



(a) M-CifarNet accuracy/MACs tra

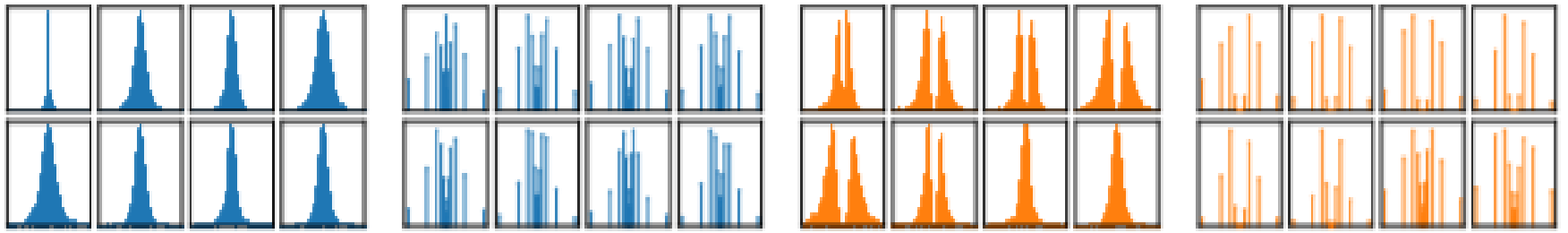


Gao, et al., Dynamic Channel Pruning, ICLR'2019

Efficient Model Inference: Shift Quantization for Sparcity

Key Observations

- Channel pruning introduces various degrees of sparsity to different layers
- But, traditional shift quantization位移量化becomes a poor choice for certain layers in sparse models, as most near-zero quantization levels are under-utilized.

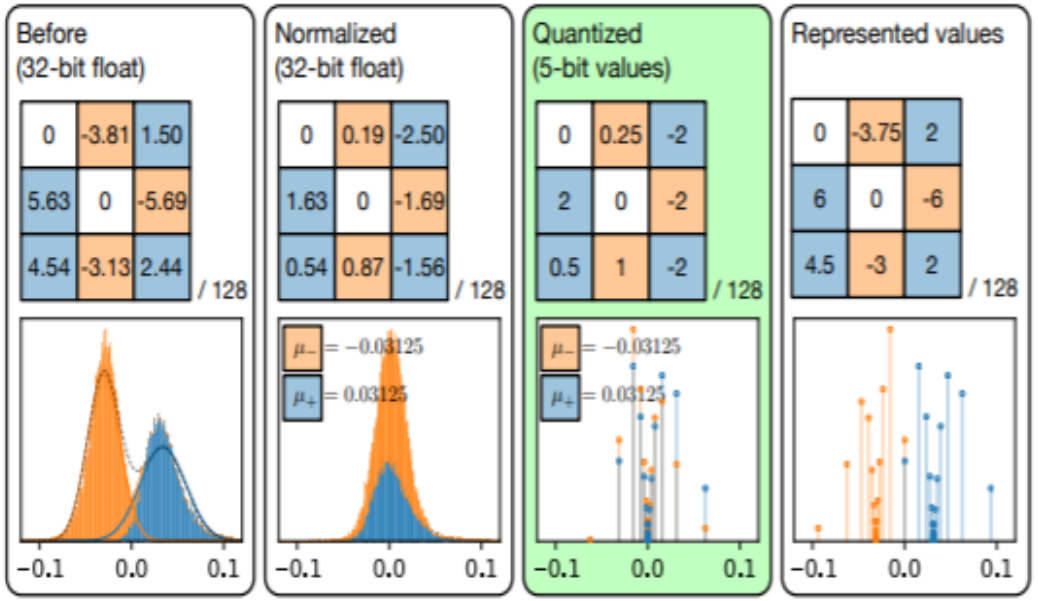


(a) Dense layers (b) After shift quantization (c) Sparse layers (d) After shift quantization

Figure 1: The weight distributions of the first 8 layers of ResNet-18 on ImageNet. (a) shows the weight

Focused Quantization for Efficient Inference

Focused Quantization: exploit the statistical properties of weights in pruned models to quantize them efficiently and effectively



Gao, et al., Focused Quantization for Sparse CNNs, NeurIPS'2019

ResNet-18	Top-1	Top-5	Size (MB)	CR (×)
TTQ [27]	66.00	87.10	2.92*	16.00*
INQ (2 bits) [26]	66.60	87.20	2.92*	16.00*
INQ (3 bits) [26]	68.08	88.36	4.38*	10.67*
ADMM (2 bits) [14]	67.0	87.5	2.92*	16.00*
ADMM (3 bits) [14]	68.0	88.3	4.38*	10.67*
ABC-Net (5 bases, or 5 bits) [15]	67.30	87.90	7.30*	6.4 *
LQ-Net (preact, 2 bits) [23]	68.00	88.00	2.92*	16.00*
D&Q (large) [20]	**73.10**	**91.17**	21.98	2.13*
Coreset [3]	68.00	—	3.11*	15.00
Focused compression (5 bits, sparse)	68.36	88.45	**2.86**	**16.33**
ResNet-50	**Top-1**	**Top-5**	**Size (MB)**	**CR (×)**
INQ (5 bits) [26]	74.81	92.45	14.64*	6.40*
ADMM (3 bits) [14]	74.0	91.6	8.78*	10.67*
ThiNet [17]	72.04	90.67	16.94	5.53*
Clip-Q [22]	73.70	—	6.70	14.00*
Coreset [3]	74.00	—	5.93*	15.80
Focused compression (5 bits, sparse)	**74.86**	**92.59**	**5.19**	**18.08**

HW/SW Co-Design for Model Inference

Key Observations:

- Shift op facilitates HW impl. HW design tends to use flattened streaming arch (vs systolic arrays) for inference acceleration.
- Flatten streaming accelerators isolate layer-wise comp, offering chance to use different arith and precisions for each layer's computation

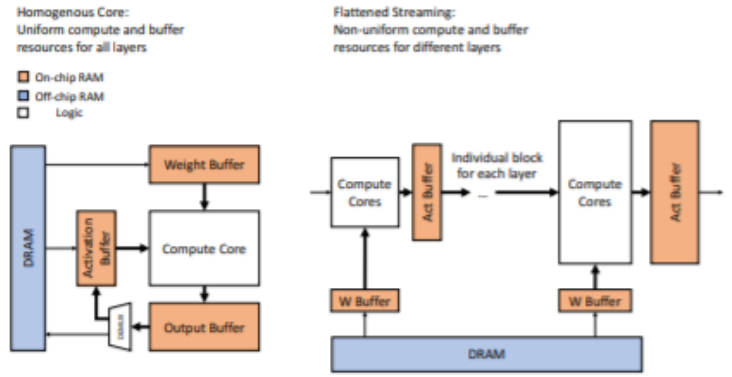


Fig. 1: An illustration of a homogenous core (left) and flattened streaming cores (right).

Proposed Tomato HW/SW Co-Design:

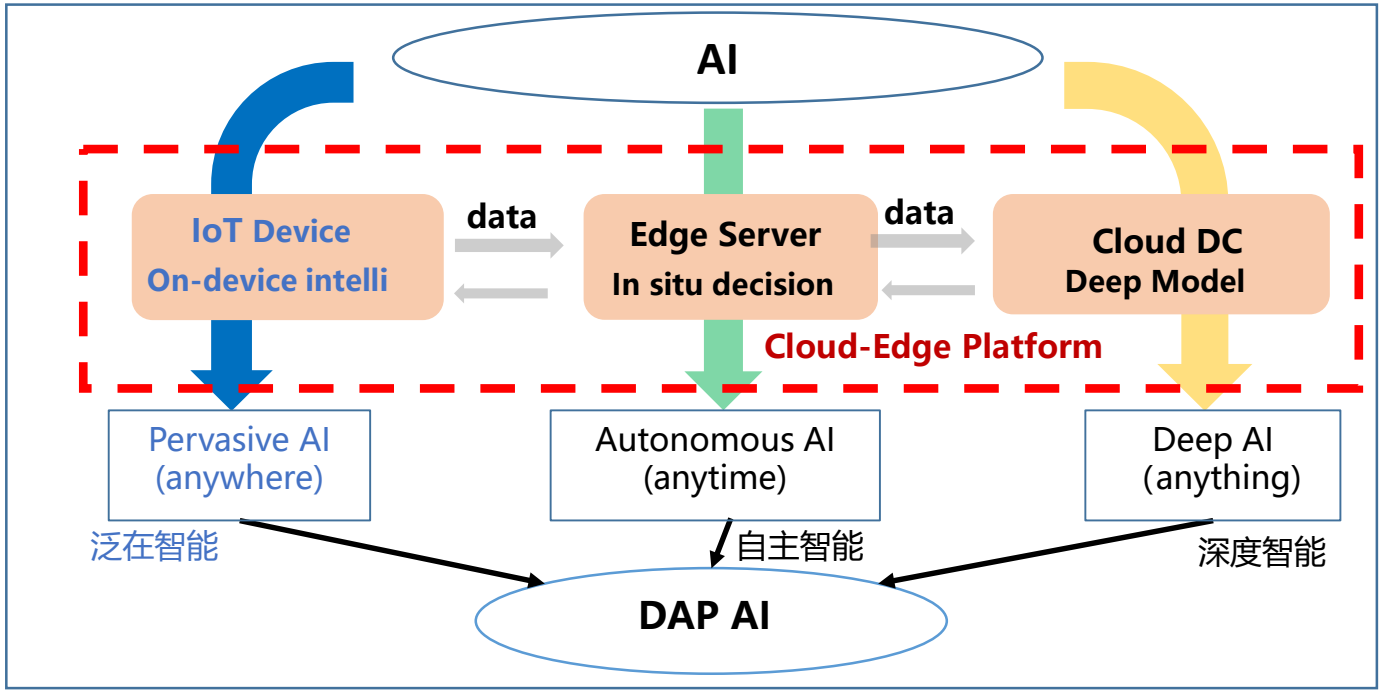
- HW: Multi-Precision Multi-Arith accelerator on Multi-FPGAs
- SW: Hybrid quantization to automate the selection of arith and precisions for different layers of the model, so as to map all the layers onto a single or multiple FPGAs.

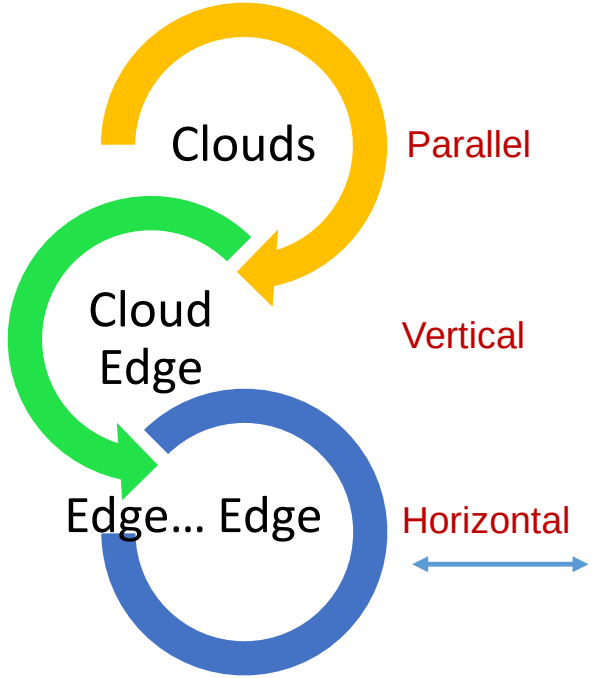
	Implementation	Quantisation(s) Weights	Acts	Platform	Frequency (MHz)	Latency (ms)	Throughput (FPS)	Arithmetic perf. (GOP/s)
VGG16	Throughput-Opt [33]	FXP8	FXP16	Intel Stratix V	120	262.9	3.8*	117.8
	fpgaConvNet [34]	FXP16	FXP16	Xilinx Zynq XC7Z045	125	197*	5.07	156
	Angel-Eye [9]	BFP8	BFP8	Xilinx Zynq XC7Z045	150	163*	6.12*	188
	Going Deeper [25]	FXP16	FXP16	Xilinx Zynq XC7Z045	150	224*	4.45	137
	Shen *et al.* [31]	FXP16	FXP16	Xilinx Virtex US XCVU440	200	49.1	26.7	821
	HARPv2 [23]	BIN	BIN	Intel HARPv2	–	8.77*	114	3500
	GPU [23]	FP32	FP32	Nvidia Titan X	–	–	121	3590
MobileNet	Ours	Mixed	FXP8	Intel Stratix 10	156	0.32	3109	3536
	Ours	Mixed	FXP8	Xilinx Virtex US+ XCVU9P	125	0.40	2491	2833
	Zhao *et al.* [41]	FXP16	FXP16	Intel Stratix V	200	0.88	1131	1287
	Zhao *et al.* [42]	FXP8	FXP8	Intel Stratix V	150	4.33	231	264
	GPU	FP32	FP32	Nvidia GTX 1080Ti	–	279.4	515	586

Gao, et al., FPGA Implementation for CNN acceleration, FPT'2019

Early Experience in Cloud-Edge Intelligence

- Parallel Learning in Clouds
- Cloud-Edge Collaboration for (Vertical) Distributed Learning
- Cooperative Learning among Peers (Horizontal)
  - Dealing with System and Data Heterogeneity (CVPR'2023)

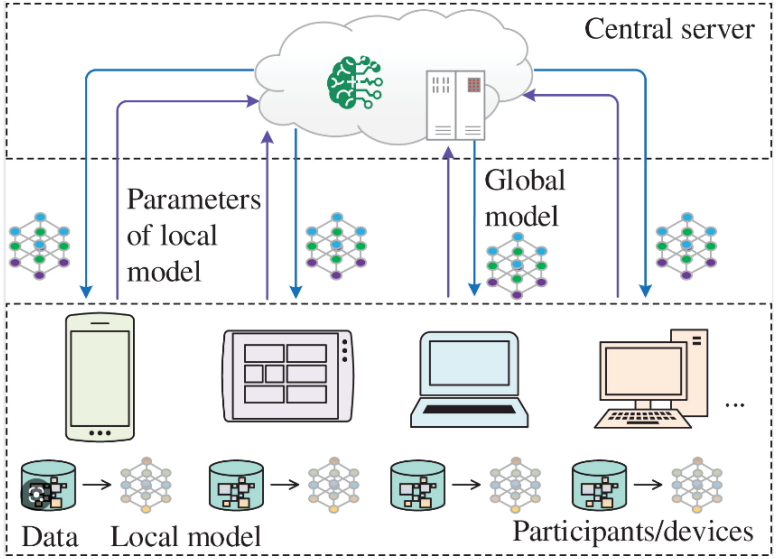




Challenges in Cooperative Learning

- In cooperative learning, peers are not necessarily homogenous
- Data located in peers most likely be biased or unbalanced

- In IoT devices, the savings of comm could be completely dwarfed by the expensive computation cost
  - E.g. an iPhone 12 running fashion-mnist model as a FedAvg of FL client took 6+ hours on training, but only minutes under 5G/WIFI to transmit required 700 MB data



Heterogeneity in computing capabilities clients may result in slow convergence in FL.

Data Heterogeneity in FL

- In MNIST training, clients allocated with the same digits shared similar update patterns, while different client pair's update direction is quite dispersed.

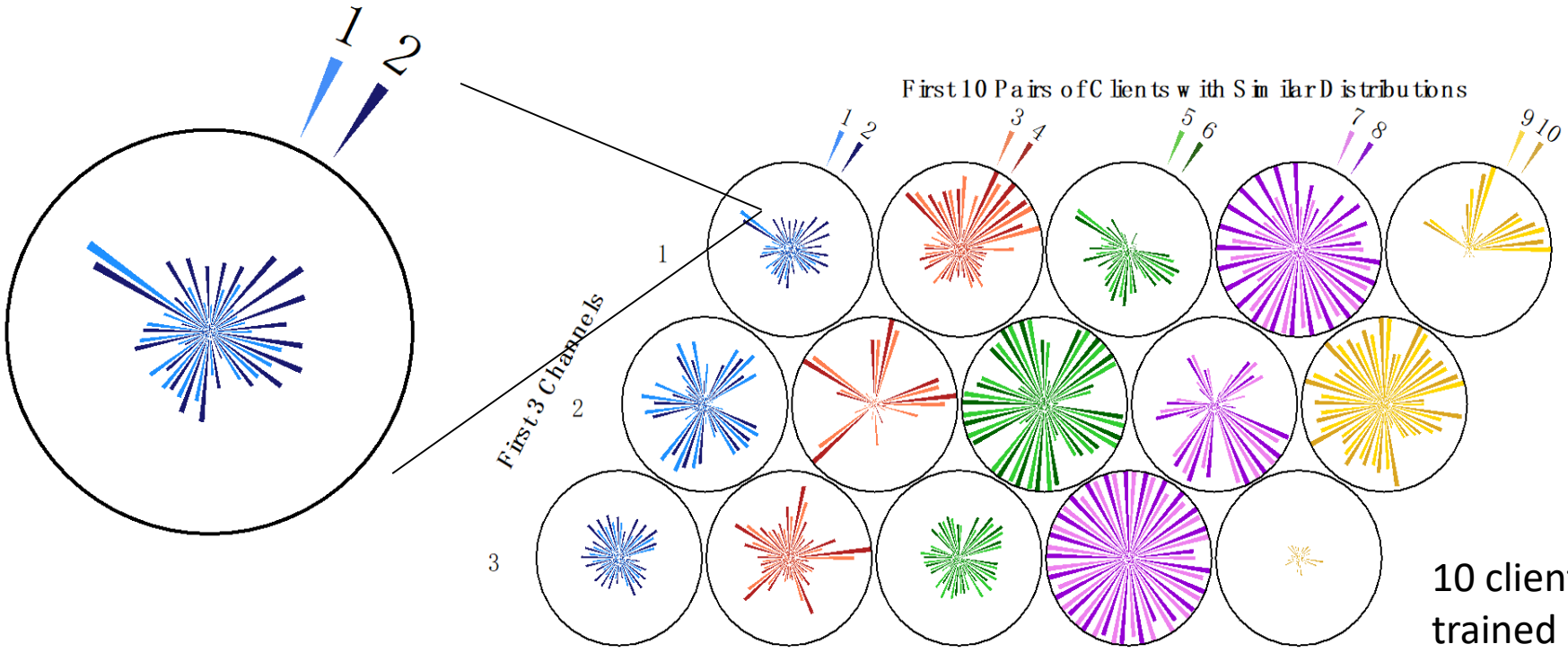
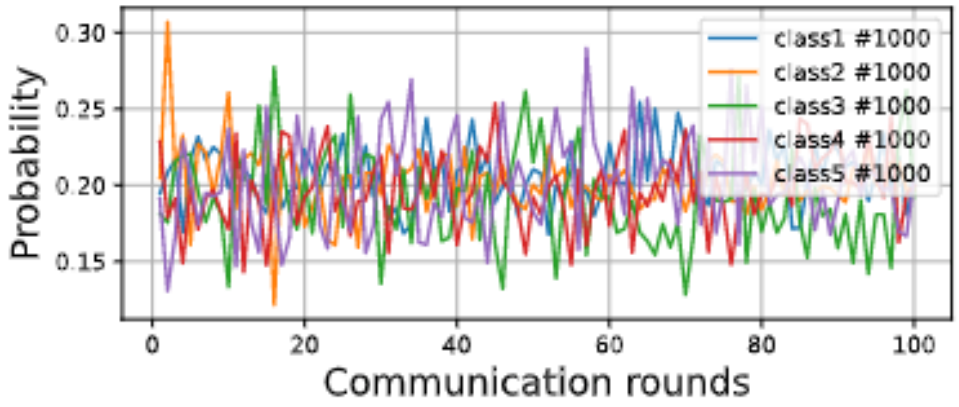


Figure 4. Magnitudes of clients' local gradient update

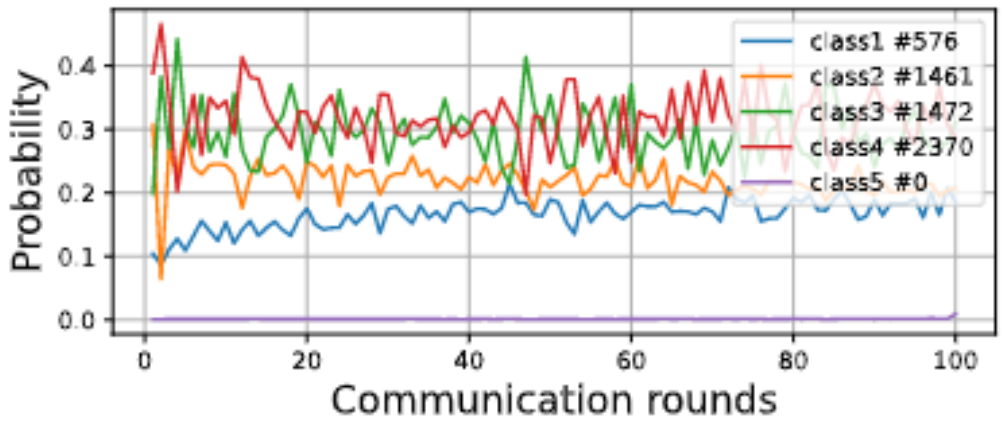
10 clients, each pair be trained with the same class-biased digits

Observation**:** local gradient update is contingent on data distribution

Biased Predictions of Clients due to Data Heterogeneity



(a) Homogeneous data



(b) Heterogeneous data.

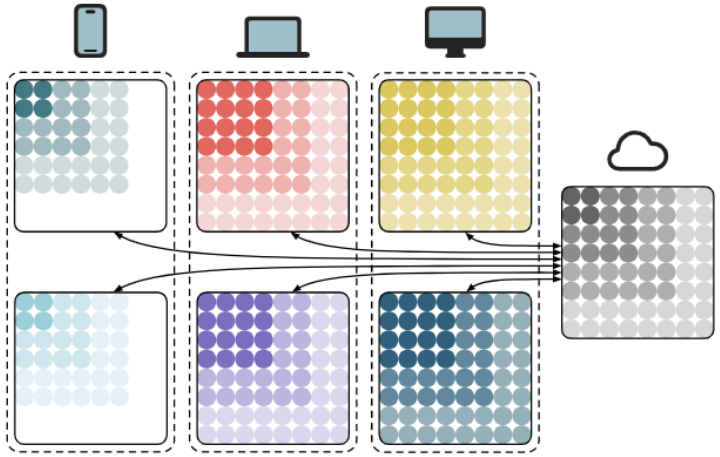
Prediction statistics on homogeneous vs heterogeneous data.

Observations:

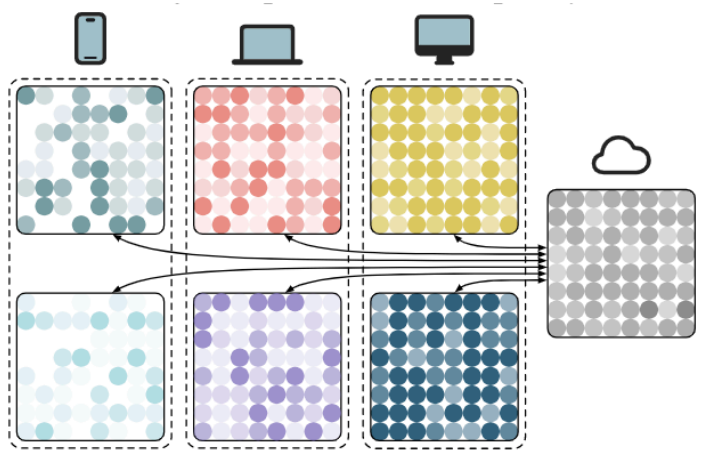
- the output predictions from the locally updated client model on a balanced validation set are *biased towards majority classes.*
- the model prediction bias *consistently exists* throughout the whole training phase.

Adaptive Channel Sparsity for System Heterogeneity

Proposed Method *Flado*: Federated Learning with Adaptive Dropout



(a) FjORD prescribes fixed sparsity.



(b) Adaptive sparsity with Flado.

FjORD trains models with fixed channel sparsities for clients with different capabilities.

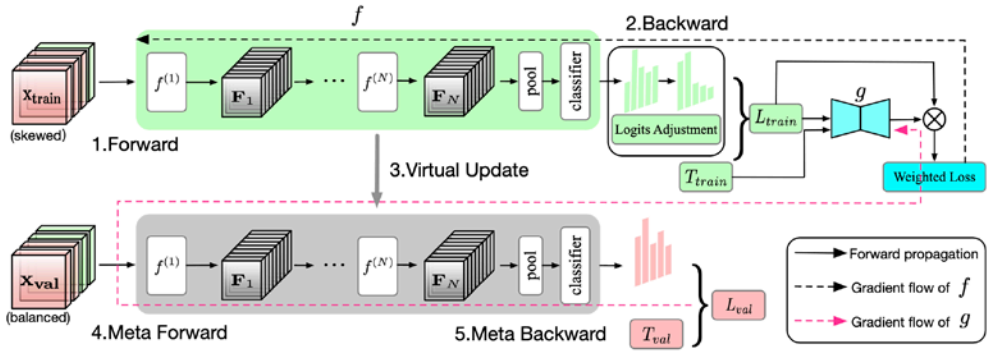
Flado adapts channel sparsities with underlying training trajectories and capabilities for each training round.

Flado tailors an ***adaptive sparsity*** scheme for each client according to their local gradient updates.

Sparsity-driven Trajectory Alignment

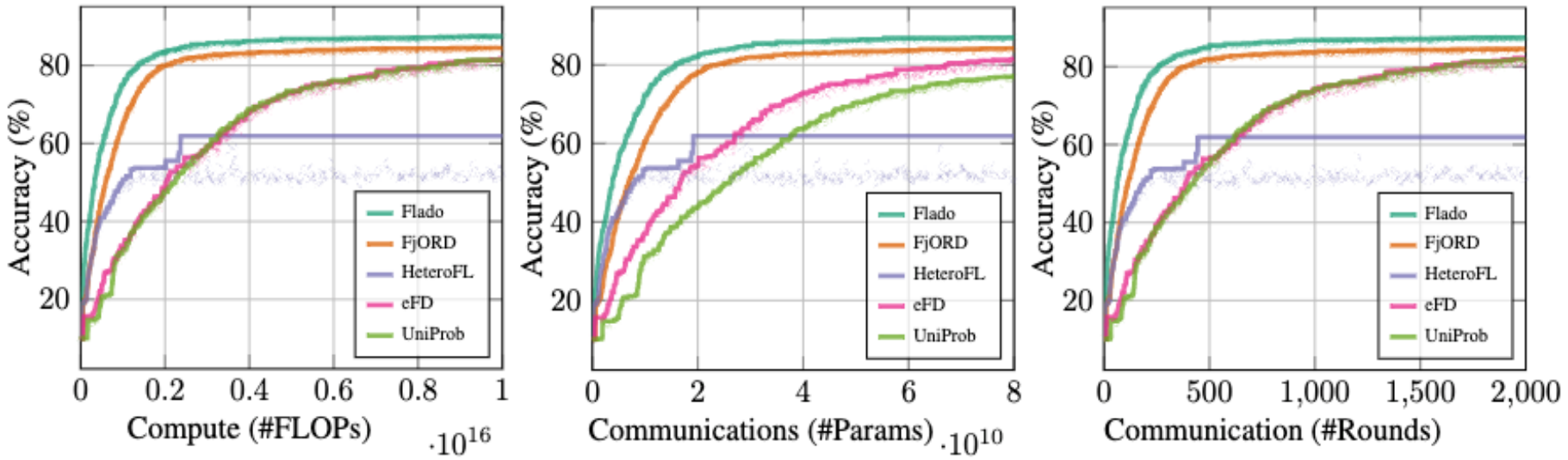
$$\max_{\mathbf{p}_c} \mathbb{E}_{\mathbf{b}_c \sim \mathcal{B}(\mathbf{p}_c)} \operatorname{cossim}\left(J\left(\Delta\boldsymbol{\theta}^{(t)}\right), J\left(\nabla_{\boldsymbol{\theta}^{(t)}} \ell_c\left(\mathbf{b}_c \circ \boldsymbol{\theta}^{(t)}\right)\right)\right), \quad s.t.\ g_c(r_c, \mathbf{p}_c) \geq 0.$$

FLOPs budget constraint



Experimental Results of Adaptive Channel Sparsity

- *Flado* is highly elastic under different system heterogeneity levels.
- *Flado* attains consistently higher converged accuracies on CIFAR-10

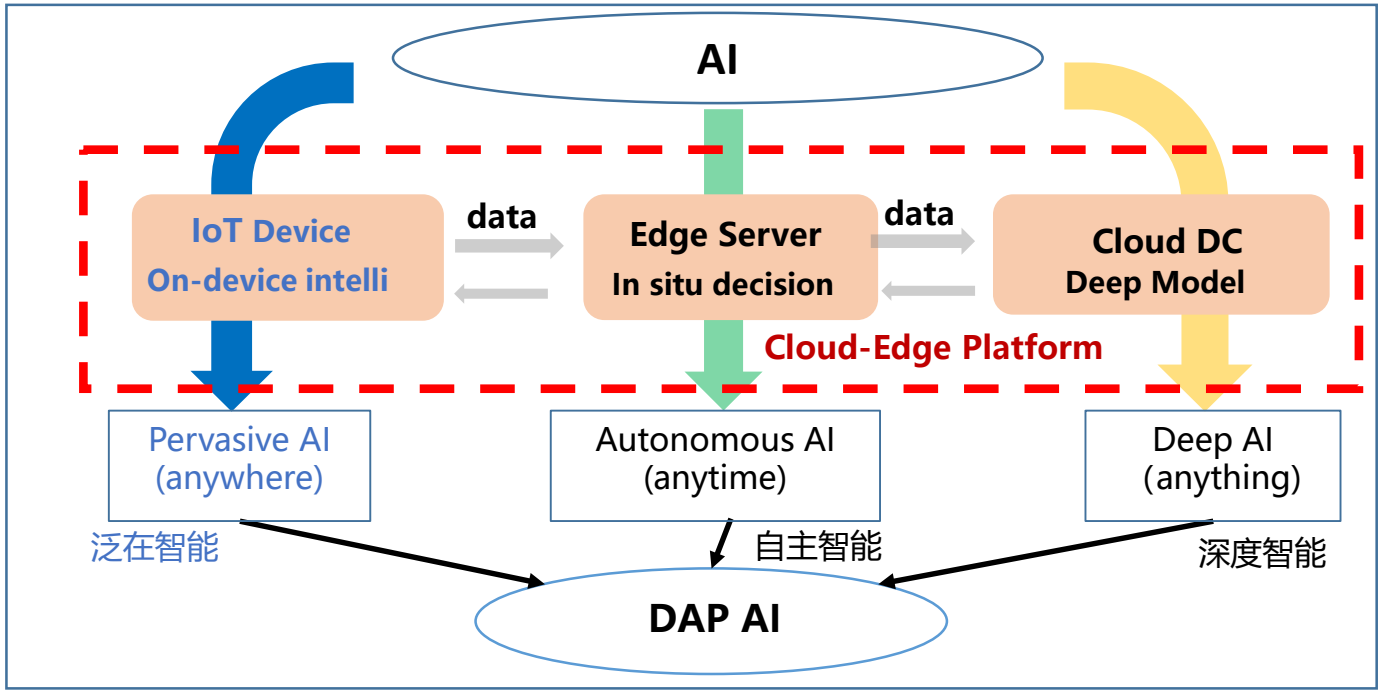


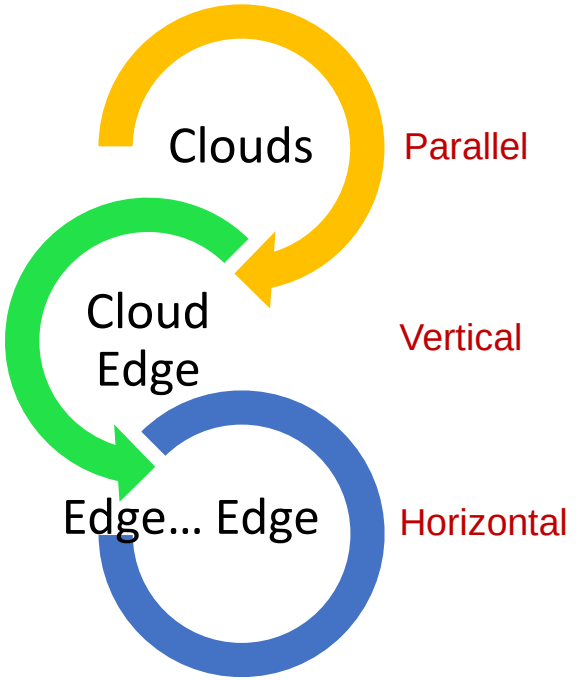
(a) Accuracy *vs*. FLOPs. (b) Accuracy *vs*. Communicated parameters. (c) Accuracy *vs*. Communication rounds.

D. Liao, X. Gao, Y. Zhao, C. Xu, Adaptive Channel Sparsity for Federated Learning under System Heterogeneity, CVPR'2023

In Summary: Effective Systems for AI

- Resource Management for Distributed Learning in Clouds
- Cloud-Edge Collaboration for (Vertical) Distributed Learning
- Cooperative/Federated Learning among Peers (Horizontal)





Outline

Part I: Challenges in the Design of Effective Systems

Part II: Recent Experience wtih Effective Systems for AI

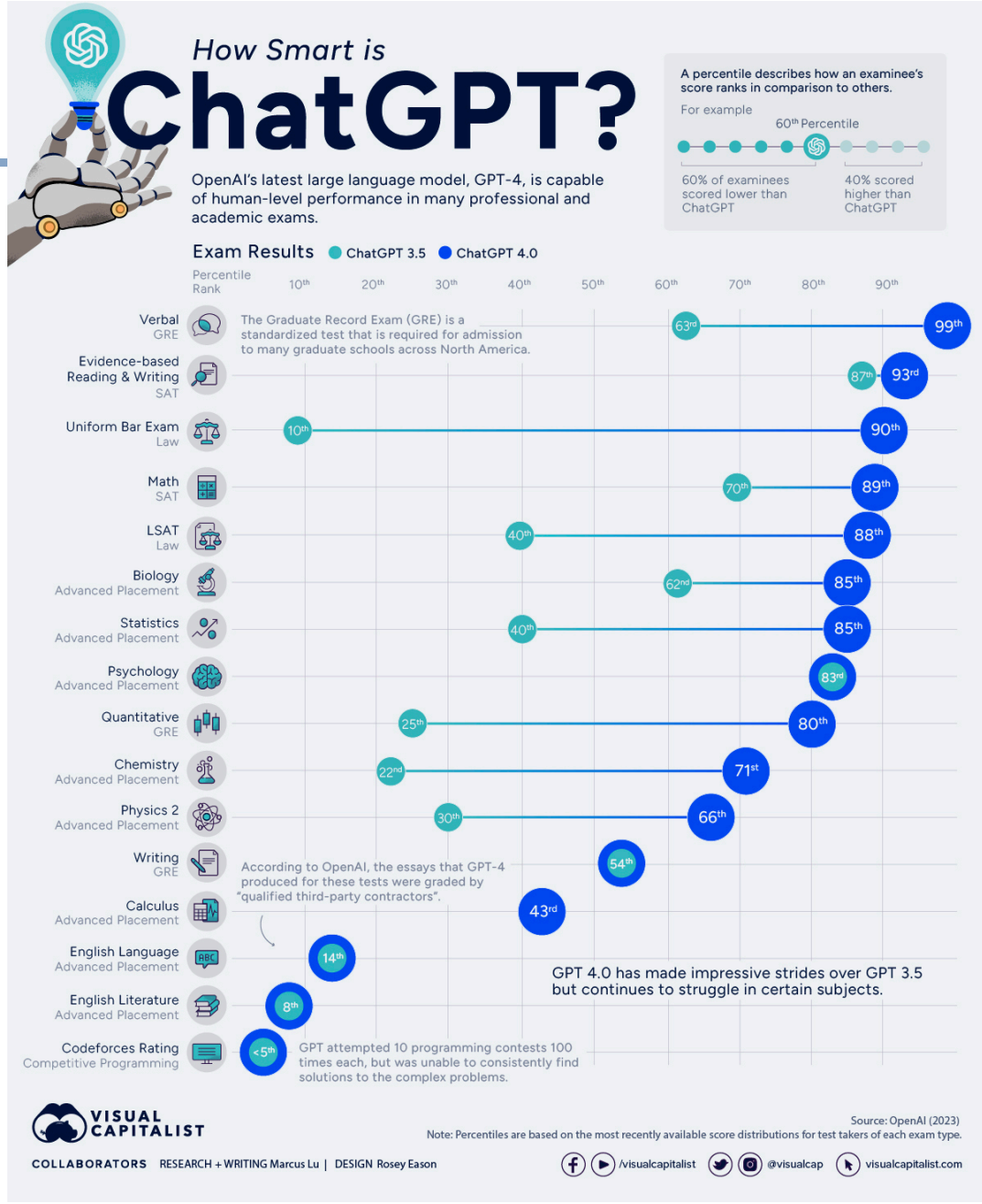
Part III: AIGC and Foundation Model's Impact

Intelligence Emergence

chatGPT/AIGC 智能涌现：4 个能力

- 记忆力： 集体记忆 vs 个性化记忆力（AI agent)
- 创造力：生成语言，文本，图像等。可以吟诗作画，coding）
- 理解力 （对某个事物或事情的认识、认知、抽象的能力）：可以举一反三，可以意译文字语言，个性化翻译
- 推理能力（常识、逻辑）：归纳（induction）演绎（deduction）

https://www.visualcapitalist.com/how-smart-is-chatgpt/



Progress towards AGI

Performance (rows) x **Generality** (columns)	**Narrow** *clearly scoped task or set of tasks*	**General** *wide range of non-physical tasks, including metacognitive abilities like learning new skills*
Level 0: No AI	**Narrow Non-AI** calculator software; compiler	**General Non-AI** human-in-the-loop computing, e.g., Amazon Mechanical Turk
Level 1: Emerging *equal to or somewhat better than an unskilled human*	**Emerging Narrow AI** GOFAI[4]; simple rule-based systems, e.g., SHRDLU (Winograd, 1971)	**Emerging AGI** ChatGPT (OpenAI, 2023), Bard (Anil et al., 2023), Llama 2 (Touvron et al., 2023)
Level 2: Competent *at least 50th percentile of skilled adults*	**Competent Narrow AI** toxicity detectors such as Jigsaw (Das et al., 2022); Smart Speakers such as Siri (Apple), Alexa (Amazon), or Google Assistant (Google); VQA systems such as PaLI (Chen et al., 2023); Watson (IBM); SOTA LLMs for a subset of tasks (e.g., short essay writing, simple coding)	**Competent AGI** not yet achieved
Level 3: Expert *at least 90th percentile of skilled adults*	**Expert Narrow AI** spelling & grammar checkers such as Grammarly (Grammarly, 2023); generative image models such as Imagen (Saharia et al., 2022) or Dall-E 2 (Ramesh et al., 2022)	**Expert AGI** not yet achieved
Level 4: Virtuoso *at least 99th percentile of skilled adults*	**Virtuoso Narrow AI** Deep Blue (Campbell et al., 2002), AlphaGo (Silver et al., 2016, 2017)	**Virtuoso AGI** not yet achieved
Level 5: Superhuman *outperforms 100% of humans*	**Superhuman Narrow AI** AlphaFold (Jumper et al., 2021; Varadi et al., 2021), AlphaZero (Silver et al., 2018), StockFish (Stockfish, 2023)	**Artificial Superintelligence (ASI)** not yet achieved

Google DeepMind

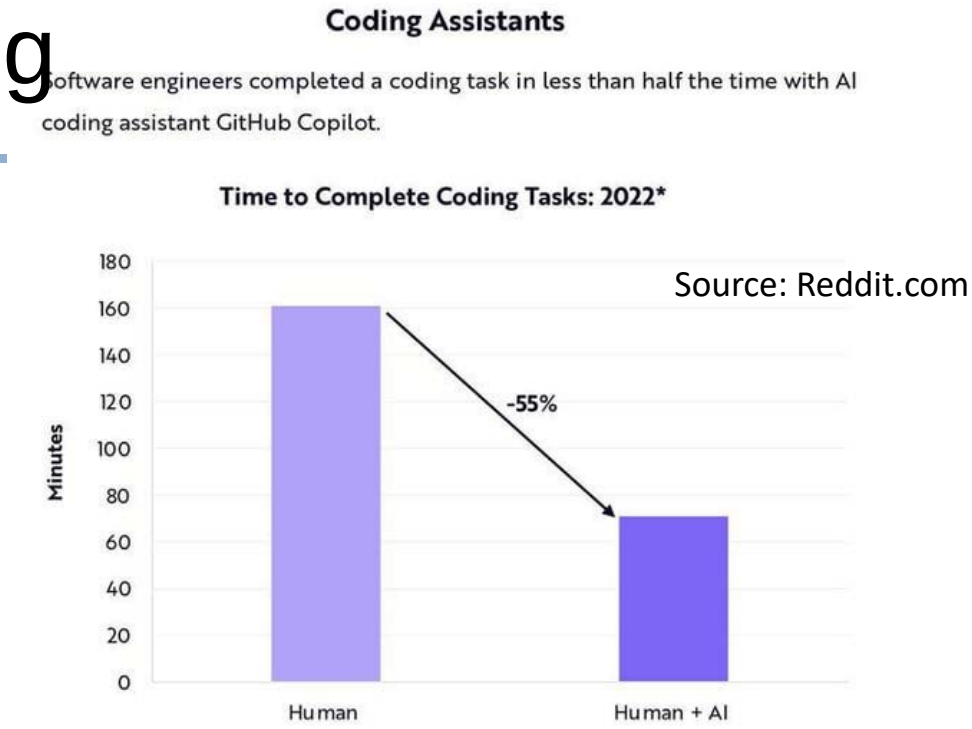
2023-11-04

Levels of AGI: Operationalizing Progress on the Path to AGI

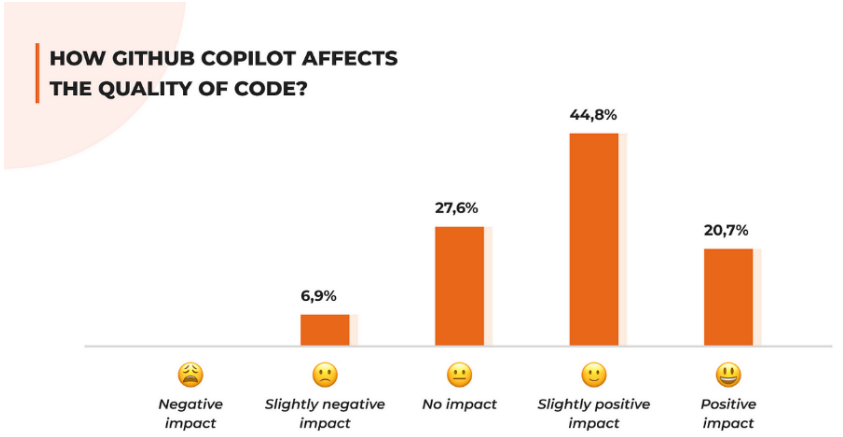
Meredith Ringel Morris[1], Jascha Sohl-dickstein[1], Noah Fiedel[1], Tris Warkentin[1], Allan Dafoe[1], Aleksandra Faust[1], Clement Farabet[1] and Shane Legg[1]
[1]Google DeepMind

All in AI, no exception for coding

- Dev of apps using models, as embodied AI
  - Productivity improvement tool:
    - creative text/image/video,
  - AI Agent: ask questions to AI agent for personalized services: healthcare/education/entertainment/…
  - Challenges: have AI agent to ask questions!
- How to dev apps with assistance of AI
  - AI-assisted coding: GitHub CoPilot, chatGPT



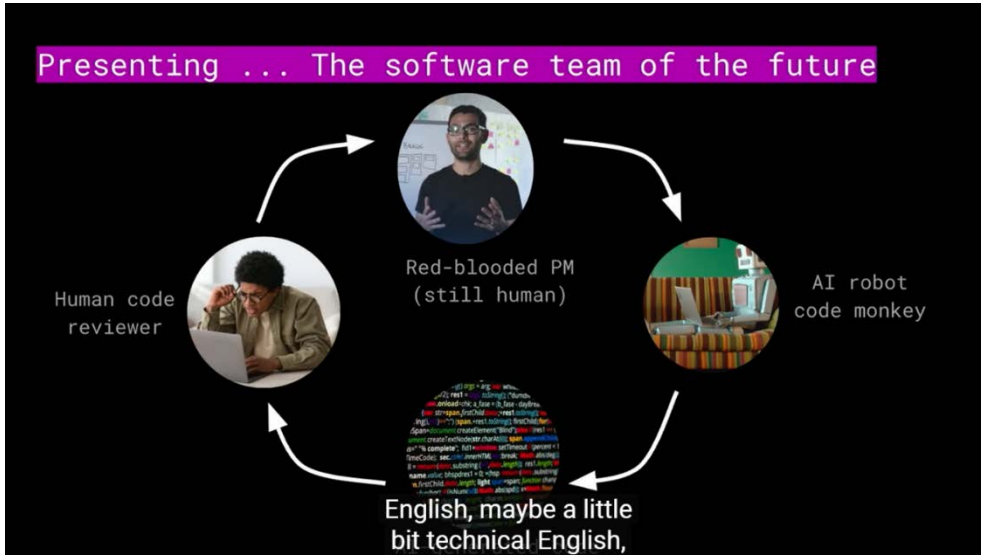
Source: Reddit.com



AKVELON

Future Software Engineering

- Evolution of programming abstraction
  - Low-level: binary/assembly code
  - “high”-level: c/c++/Java/…/Python/
- NL-based coding: coding on how to do could be assisted by AI
  - Abstraction is further lifted
  - Arch/OS/algs need to be revisited
- Need to focus on what to do and learn to ask questions.
  - Creative thinking remains critical in SE.
  - Task-/agent-oriented might be a future

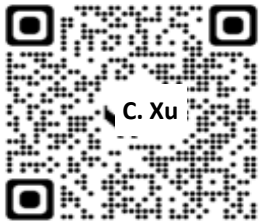


End of programming might be exaggerated at present, but alarming

Conclusion

- Takeaways:
  - Effective systems for AI and Large Models, particularly via cloud-edge collaboration
  - Resource Management is a key to serverless cloud computing, in support of AI
  - Model Compression makes a good tradeoff for edge AI
  - System/Data heterogeneity must be dealt with in collaborative AI among peers
- AI for ALL, including programming/coding. AI for how to do and human for what to do in high abstraction levels.

Thank You !



Chengzhong Xu （须成忠）
University of Macau (澳门大学）