

Resource Heterogeneity-Aware and Utilization-Enhanced Scheduling for Deep Learning Clusters

Abeda Sultana, Nabin Pakka , *Graduate Student Member, IEEE*, Fei Xu , Xu Yuan ,
Li Chen , *Senior Member, IEEE*, and Nian-Feng Tzeng 

背景：深度学习的应用已经广泛覆盖了各个领域。为了满足深度神经网络（DNN）训练的日益增长的需求，大型企业和云服务提供商构建了专用的深度学习集群。这些集群中的昂贵资源需要在多用户任务之间进行高效调度，以考虑深度学习任务的整体性能、集群资源利用率、公平性等。

论文中的研究现状：尽管 Hadar 能够感知异构性，在多个训练轮次中，每个作业至多只能被调度到一个集群节点（即机器或服务器）上运行，集群资源往往无法得到充分利用。

研究方向和贡献：深度学习集群中的细粒度异构性识别和资源调度优化。

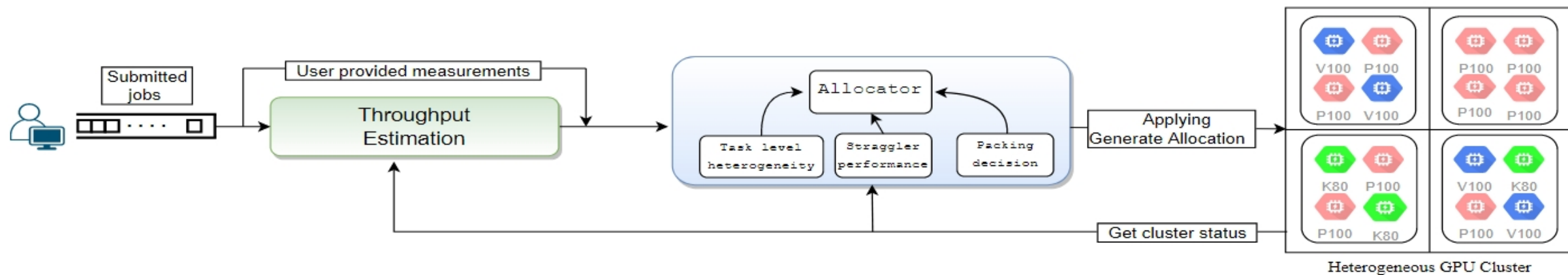


Fig. 2: The overview of *Hadar*, a fine-grained heterogeneity-aware scheduler for a GPU-based deep learning cluster.

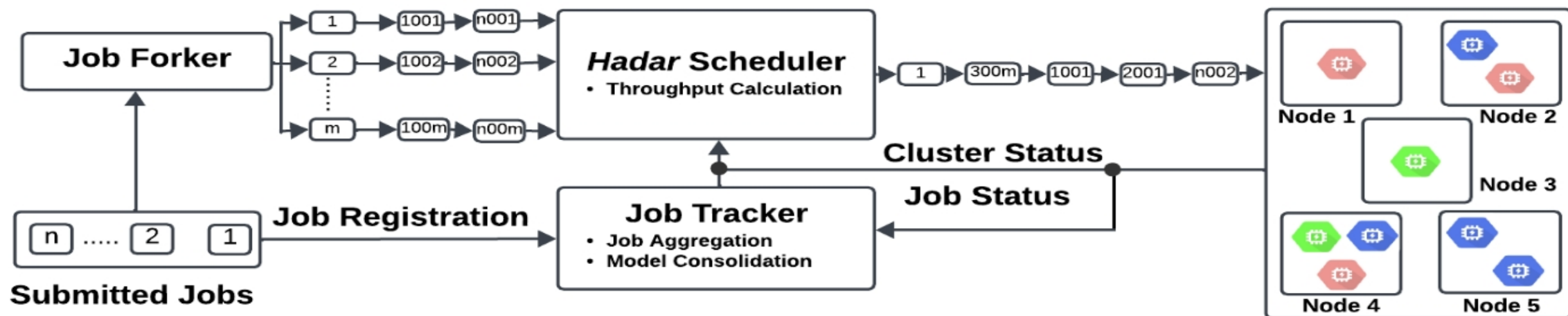


Fig. 7: An overview of *Hadar Enhancement*, *HadarE*.

Scheduling Copies of Training Jobs

Algorithm 1 Online Scheduling in *Hadar*.

Input: $c_h^r, \forall h \in [H], r \in [R]$

- 1: **Initialize:** $w_{jh}^r(t) = 0, \gamma_h^r(t) = 0, k_h^r(t) = k_h^r(0), \forall j \in [J], t \in [T], h \in [H]$
 - 2: **while** true **do**
 - 3: Upon the arrival of each job, admit it to the queue Q
 - 4: In each round t :
 - 5: $\{Q_s, c_h^r, \{w_{jh}^r(t)(t)\}\}$ =
 $DP_allocation(0, Q, c_h^r, null, \gamma_h^r(t), k_h^r(t))$
 - 6: **for** job $j \in [Q_s]$ **do**
 - 7: Run job j until round $t+1$ according to $(\{w_{jh}^r(t)\})$
 - 8: **end for**
 - 9: If j is complete, remove it from Q
 - 10: **end while**
-

Scheduling Copies of Training Jobs

Algorithm 2 $DP_allocation(idx, Q, srvr, \{w_{jh}^r(t)\}, \gamma_h^r(t), k_h^r(t), \forall h, \forall r)$

```

1: if ( $index \geq Q.length()$ ) ||  $is\_Server\_Full(srvr)$  then
2:   return  $Q, \{w_{jh}^r(t)\}, srvr$ 
3: end if
4:  $job = Q[idx]$ 
5:  $\{w_{prev_{jh}}^r\} \leftarrow \{w_{jh}^r(t)\}$ 
6:  $\{w_{job_{jh}}^r\} = \text{FIND\_ALLOC}(job, srvr)$ 
7: if  $\{w_{job_{jh}}^r\} = null$  then
8:   return  $Q, \{w_{jh}^r(t)\}, srvr$ 
9: end if
10:  $\gamma_h^{rc}(t) = \gamma_h^r(t) + w_{job_{jh}}^r, \forall h, \forall r$ 
11:  $k_h^{rc}(t) \leftarrow$  Update  $k_h^r(t)$  according to Eq. (5),  $\forall h, \forall r$ 
12:  $srvr^c \leftarrow$  Update  $srvr$  according to  $\{w_{job_{jh}}^r\}$ 
13:  $\{w_{jh}^r(t)\}.append(w_{job_{jh}}^r), \forall h, \forall r$ 
14:  $(Q, \{w_{jh}^r(t)\}, srvr^c) = DP\_allocation((idx + 1), Q, srvr^c, \{w_{jh}^r(t)\}, \gamma_h^{rc}(t), k_h^{rc}(t)), \forall h, \forall r$ 
15:  $(Q, \{w_{jh}^{cr}(t)\}, srvr) = DP\_allocation((idx + 1), Q, srvr, \{w_{prev_{jh}}^r\}, \gamma_h^r(t), k_h^r(t)), \forall h, \forall r$ 
16:  $cost_Q += \sum_h \sum_r k_h^{rc}(t) w_{jh}^r(t)$ 
17:  $cost_{Q/j} += \sum_h \sum_r k_h^r(t) w_{jh}^{cr}(t)$ 
18: if  $cost_Q < cost_{Q/j}$  then
19:   return  $Q, \{w_{jh}^r(t)\}, srvr^c$ 
20: end if
21: return  $Q, \{w_{jh}^{cr}(t)\}, srvr$ 

```

```

22: procedure FIND_ALLOC( $job, srvr$ ):
23:    $srvr \leftarrow$  sort GPU type (desc. order of  $x_j^r, \forall h \in H$ )
24:    $\{allocs_{packed}\} \leftarrow$  allocs. within a consolidated setting.
25:    $\{allocs_{!packed}\} \leftarrow$  allocs. independent of consolidation.
26:    $\{cost_{packed}\} \leftarrow \sum_h \sum_r k_h^r(t) w_{jh}^r(t), \forall allocs_{packed}$ 
27:    $\{cost_{!packed}\} \leftarrow \sum_h \sum_r k_h^r(t) w_{jh}^r(t) +$ 
    $comm. cost, \forall allocs_{!packed}$ 
28:    $alloc \leftarrow$  alloc. corresponding to
    $\min(\{cost_{packed}\}, \{cost_{!packed}\})$ 
29:    $\mu_j = \mathcal{U}_j(f_{js} - a_j) - \min(\{cost_{packed}\}, \{cost_{!packed}\})$ 
30:   if  $\mu_j > 0$  then
31:     return  $alloc$ 
32:   end if
33:   return  $null$ 

```

Scheduling Copies of Training Jobs

	R1	R2	R3	R4	R5	R6	R7	R8	R9
canpc37	J1	J1							
canpc39	J2			J3	J3				
canpc40		J2	J2						
canpc41	J3	J3	J3	J2					
Dell			J1			J3	J3	J3	J3

(a) Under *Hadar*

	R1	R2	R3	R4	R5	R6	R7	R8	R9
canpc37	J1	J1	J3						
canpc39	J1	J2	J3	J3					
canpc40	J2	J2	J3						
canpc41	J2	J3	J3						
Dell	J3	J3	J3						

(b) Under *HadarE*

Fig. 6: Comparative illustration of scheduling rounds under (a) *Hadar* and (b) *HadarE*.

$$\text{Throughput} = \frac{PMI \times \text{batch_size} \times \text{pcie_scaling}}{\text{model_weight} \times \text{dataset_size}},$$

PMI (性能-内存指数) 表示 GPU 在 Tensor Core 支持下的并行处理能力与 GPU 显存容量之间的关系

Aggregating and Consolidating Training Copies

在一个调度轮次开始时，给定某个作业剩余的训练步数，对于一个包含 n 个节点的集群，HadarE 会根据这些节点各自的吞吐量值，将剩余训练步数划分为 n 份。随后，这 n 份训练任务将被分配给该作业的 n 个分叉副本。

$$\varphi = \frac{\text{训练数据大小}}{\text{mini-batch 大小}} \quad \varphi \times \text{训练 epoch 数}$$

当时间片结束或者节点完成训练的时候会发生聚合，整合。

对于一个作业，由所有参与训练该作业副本的节点所完成的总步数，会由 Job Tracker 进行累加，即完成聚合；同时，该作业模型参数则通过对各个训练副本的参数进行加权平均而完成整合。

EVALUATION

实验环境: AWS 物理集群: 1 个 16 GB 显存的 Tesla V100 GPU; 2 个 12 GB 显存的 Tesla K80 GPU; 以及 2 个 16 GB 显存的 Tesla T4 GPU。

实验室异构测试集群: Nvidia Titan RTX GPU (24 GB 显存)、Tesla T4 GPU (16 GB 显存)、Nvidia T400 GPU (4 GB 显存)、GeForce RTX 3090 GPU (24 GB 显存) 以及 Nvidia RTX A2000 GPU (6 GB 显存)。

以上均是一张卡一个节点。

工作负载:

$$M-3 = \langle LT, 2 \times MM \rangle$$

$$M-4 = \langle IC, LM, LT, MM \rangle$$

$$M-5 = \langle IC, LM, LT, RS, MM \rangle$$

$$M-8 = \langle IC, LM, LT, RS, 4 \times MM \rangle$$

$$M-10 = \langle IC, LM, LT, RS, 6 \times MM \rangle$$

$$M-12 = \langle IC, LM, LT, RS, 8 \times MM \rangle$$

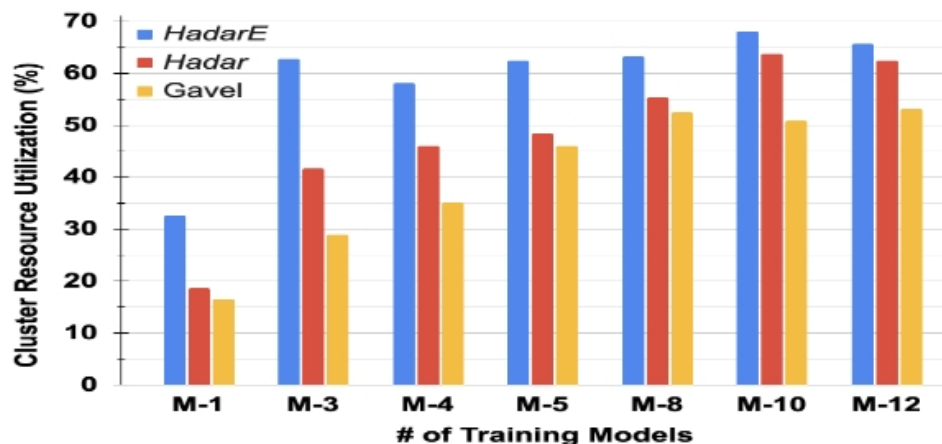
Training Job	Model	Dataset	Size
Image Classification (IC)	ResNet-18 [23]	CIFAR-10 [25]	S
Language Modeling (LM)	LSTM [26]	Wikitext-2 [27]	L
Language Translation (LT)	Transformer [29]	Multi30K [30] (de-en)	L
Recommendation System (RS)	Recorder [34]	ML-20M [35]	XL
MiMa Weather Predictions (MM)	Encoder-Decoder Transformer [32] [33] [36]	Mesonet [37] and WRF-HRRR [38]	M

EVALUATION

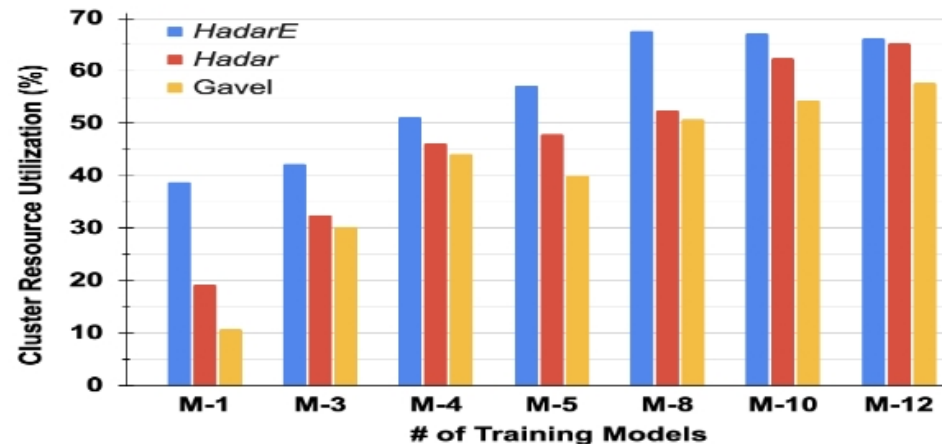
Table IV: Comparative inference quality values for models trained under *HadarE* versus under *Hadar*

Training Job	Forking (under <i>HadarE</i>)	No Forking (under <i>Hadar</i>)	Metric
Language Translation [29]	54.690	52.410	ACC
Image Classification [23]	91.620	87.340	ACC
Recommendation System [34]	38.700	40.300	MSE
Language Modeling [26]	4.310	4.460	MSE
MiMa Weather Predictions [32] [33]	0.025	0.028	MSE

EVALUATION

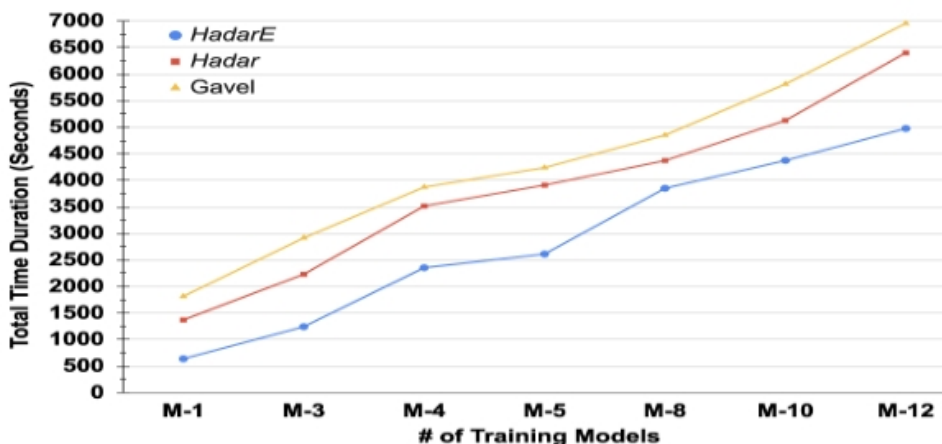


(a) Cluster Resource Utilization (CRU) for AWS Cluster

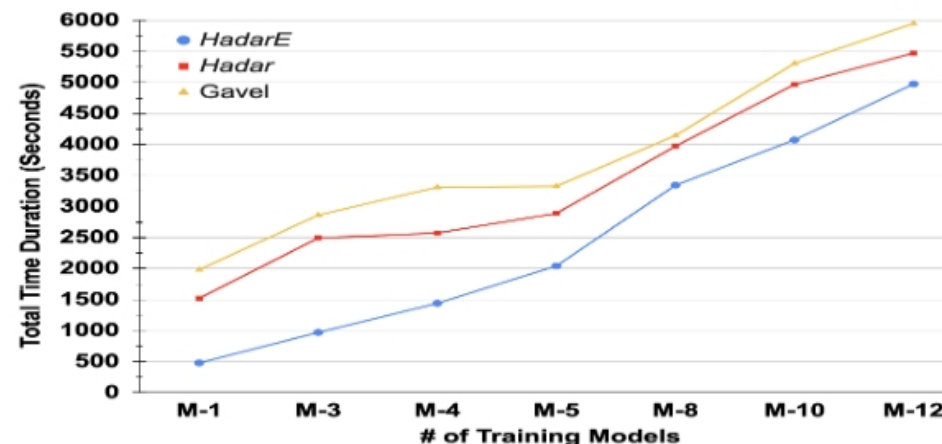


(b) Cluster Resource Utilization (CRU) for Testbed Cluster

Fig. 8: Comparison of cluster resource utilization among Gavel [10], Hadar [1], and HadarE for AWS and testbed clusters.



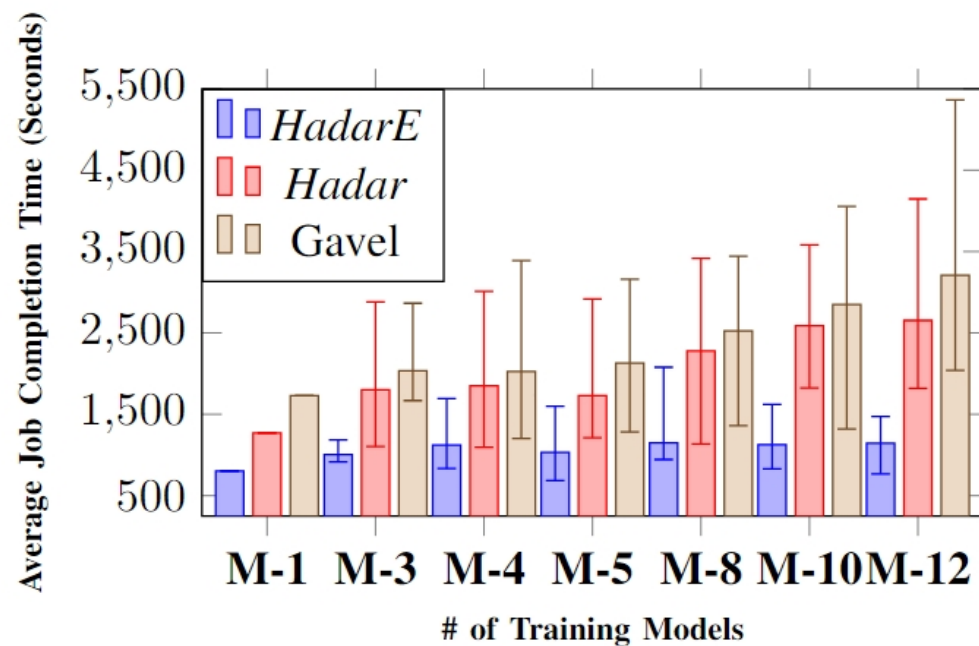
(a) Total Time Duration (TTD) for AWS cluster



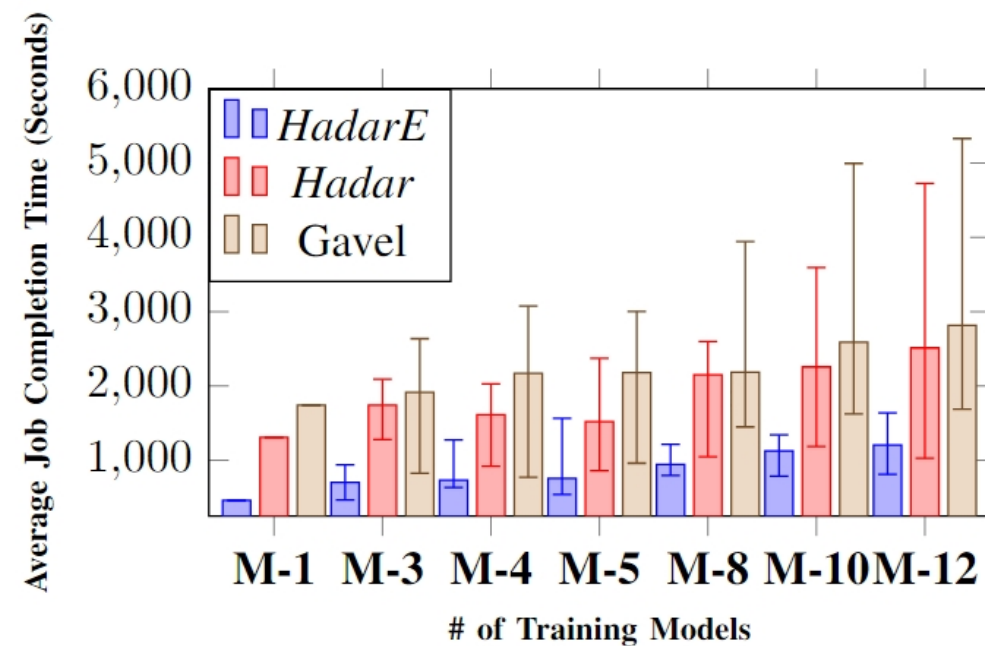
(b) Total Time Duration (TTD) for testbed cluster

Fig. 9: Comparison of TTD among Gavel [10], Hadar [1], and HadarE for AWS and testbed clusters.

EVALUATION



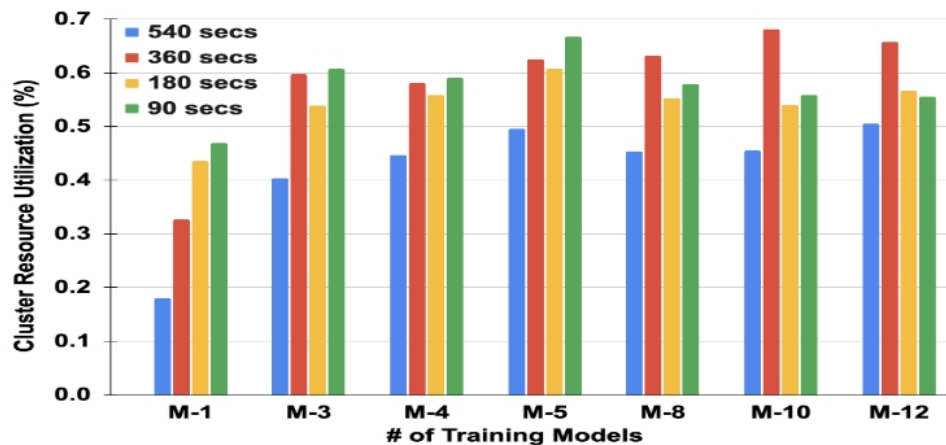
(a) Average Job Completion Time for AWS cluster



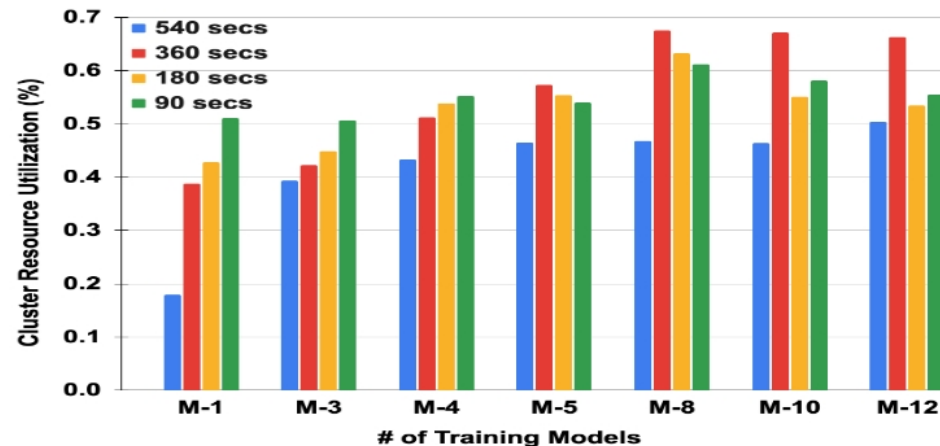
(b) Average Job Completion Time for testbed cluster

Fig. 10: Comparison of JCT among Gavel [10], Hadar [1], and HadarE for AWS and testbed clusters.

EVALUATION

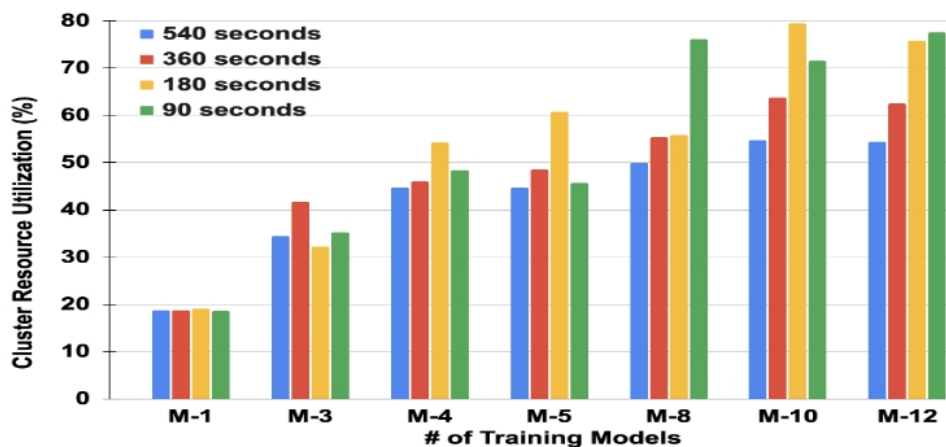


(a) Results on AWS cluster

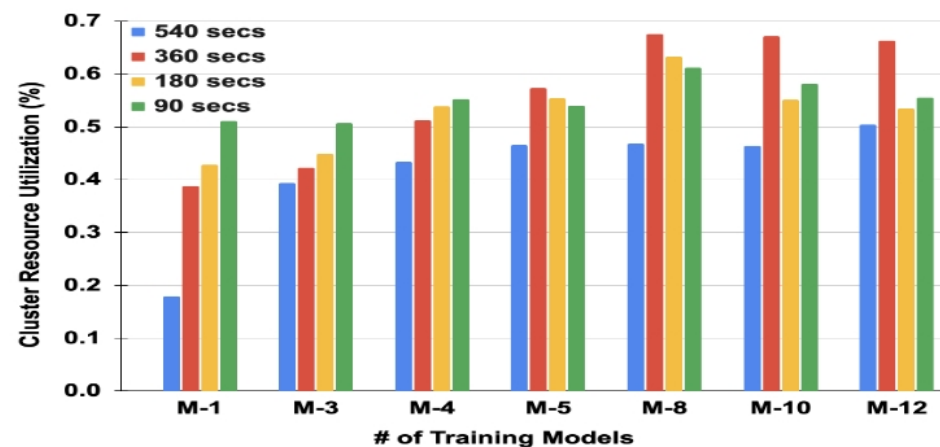


(b) Results on testbed cluster

Fig. 11: CRU results of various slot time spans on AWS and testbed clusters under *HadarE*.



(a) Results on AWS cluster



(b) Results on testbed cluster

Fig. 12: CRU results of various slot time spans on AWS and testbed clusters under *Hadar*.