A Details of Scalability Estimator

Spindle characterizes the execution time of MetaOp m over n devices, $T_m(n)$, by a generalized piecewise α-β function:

$$T_m(n) = \alpha_{m,i} + \beta_{m,i} \times c_m + \beta'_{m,i} \times w_m/n, \forall n \in [n_{i-1}, n_i], i = 1, \ldots, k$$

where k is the number of pieces, $\alpha_{m,i}$ represents the coefficient of fixed overheads (e.g., kernel launch costs), $\beta_{m,i}$ and $\beta'_{m,i}$ represent the reciprocal of execution efficiency (e.g., GPU computation speed and network bandwidth), w_m/n denotes the distributed workload of MetaOp m across n devices (e.g., computational workload), and c_m denotes the workload that doesn't scale with n (e.g., communication volume of data parallelism). Such piecewise function indicates that under varying resource scales, due to changes in the per-device workload, coefficients such as α, β and β' might differ, as the invoked kernels may vary across different workloads.

B Details of Bisection Search for Optimum of Continuous Problem

Alg. 2 illustrates our bisection search algorithm to solve the optimum of malleable project scheduling problem, MPSP. The function `Find_Inverse_Value`$(T_m, \widetilde{C} = \frac{\widetilde{C}_{mid}}{L_m})$ finds the value of $T_m^{-1}(\widetilde{C})$. It first finds the closest valid allocations of MetaOp m, denoted as $\underline{n_m}$ and $\overline{n_m}$, such that $\widetilde{C} \in [T_m(\underline{n_m}), T_m(\overline{n_m})]$. It then returns

$$n_m = \frac{(\widetilde{C} - T_m(\underline{n_m})) \cdot \overline{n_m} + (T_m(\overline{n_m}) - \widetilde{C}) \cdot \underline{n_m}}{T_m(\overline{n_m}) - T_m(\underline{n_m})}, \quad (11)$$

which is the linear combination of $\underline{n_m}, \overline{n_m}$ such that $T_m(n_m) = \widetilde{C}$.

Algorithm 2: Bisection Search for MPSP

```
Input: # Devices $N$,
       linear-piecewise execution time functions $\{T_m\}_{m=1}^M$
Output: Optimum $P_{MPSP} = \{m \to \langle n_m^*, 0, L_m \rangle\}_{m=1}^M$
1  $\mathcal{T}_{min}, \mathcal{T}_{max} \leftarrow \{T_m(N) \cdot L_m\}_{m=1}^M, \{T_m(1) \cdot L_m\}$;
2  $\widetilde{C}_{low}, \widetilde{C}_{high} \leftarrow \max \mathcal{T}_{min}, \text{sum } \mathcal{T}_{max}$;
3  while $\widetilde{C}_{high} - \widetilde{C}_{low} > \varepsilon$ do
4      $\widetilde{C}_{mid} \leftarrow (\widetilde{C}_{low} + \widetilde{C}_{high})/2$;
5      $P_{MPSP} \leftarrow \{m \to$ Find_Inverse_Value$(T_m, \frac{\widetilde{C}_{mid}}{L_m}))\}_{m=1}^M$;
6      if sum of allocations in $P_{MPSP} < N$ then
7          $\widetilde{C}_{high} \leftarrow \widetilde{C}_{mid}$;
8      else
9          $\widetilde{C}_{low} \leftarrow \widetilde{C}_{mid}$;
10 return $P_{MPSP}$
```

C Details of Experimental Workloads

Here, we introduce our experimental workloads in detail.
(1) **Multitask-CLIP:** Multitask-CLIP is a generalized version of CLIP [57], which extends CLIP to 6 modalities and multiple contrastive learning tasks of paired data modalities. We utilize the same model structure and configuration of ImageBind [22]. We select 6 modalities as ImageBind, including text, vision, audio, motion, thermal, and depth, and select 10 different multi-modal contrastive learning tasks for evaluation, each with distinct workloads. Each task activates two modality encoders simultaneously without data flow dependency.
(2) **OFASys:** OFASys [9] is a more general MT MM training workload, allowing modalities and tasks to activate the model components flexibly as needed. OFASys supports multiple modalities, such as text, vision, audio, motion, box, structure, and supports various multi-modal tasks, such as text summarization, image captioning, visual grounding, speech recognition, text-to-SQL and etc. OFASys utilizes modality-specific adaptors for different modalities, e.g., ViT for vision data, and adopts a unified encoder-decoder LM with generative loss. We select 7 different multi-modal tasks for evaluation.
(3) **QWen-VAL:** QWen-VAL is a larger-scale MT MM model with up to 9.25 billion parameters, supporting three modalities, including text, vision, and audio. It adopts the same structure and configuration of the popular open-sourced multi-modal LLMs, QWen-VL [8] and QWen-Audio [14]. It has modality encoders for audio and vision, and the extracted modality-specific features are combined with text tokens and together fed into the unified LLM, QWen [7]. We select three tasks for evaluation, i.e., vision-language (VL) task, audio-language (AL) task, and vision-audio-language (VAL) task, representing different combinations of modalities.

D Dynamicity Performance

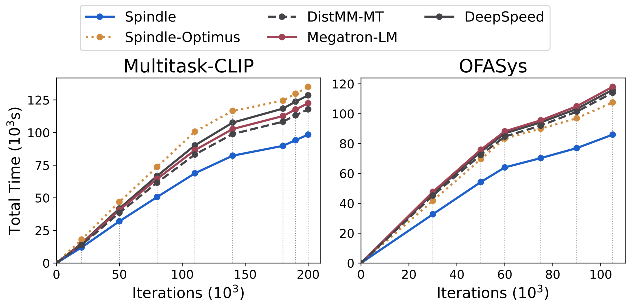

Figure 13. Comparison on dynamic multi-task workloads. Dots on the curve mark the points when the multi-task workload changes.

We evaluate the performance of various systems during dynamic changes of the multi-task workloads, a common occurrence in MT MM training. For instance, tasks with fewer training data may exit early, and new tasks may join partway through training. We simulate these dynamic changes by altering the training task set. When the multi-task workloads

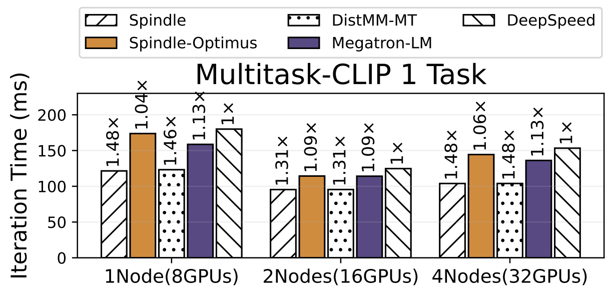

Figure 14. End-to-end performance comparison for Spindle and baseline systems on 1-task Multitask-CLIP workload.

change, the current model is first saved, and the new set of tasks and the saved model is loaded to continue training. Fig. 13 illustrates the performance of each system under such conditions. Spindle consistently achieves optimal training efficiency and the shortest overall training time. This advantage is due to Spindle's adaptability to dynamically changing workloads, enabling it to adopt an appropriate execution plan for the efficient training of MT MM models.

E Larger-scale Simulations

We further conduct evaluation on larger-scale QWen-VAL models (i.e., 30B, 70B). Due to the lack of massive scale resources, we use the simulation-based approach to estimate the system performance on larger cluster with 256 GPUs. The speedup ratio of each system compared to DeepSpeed is summarized in Tab. 2. As can be seen, on larger-scale models and clusters, Spindle consistently achieves a substantial speedup of over 1.3× compared to DeepSpeed, validating its scalability w.r.t model size. In contrast, other competiters achieve speedups of less than 1.1× compared to DeepSpeed. Besides, we also want to clarify that MT MM models within 10B are popular and widely deployed in real-world applications (e.g., BLIP-2 [33], MiniGPT-4 [89], QWen-Audio [14], DeepSeek-VL [40]). Their popularity is not only because many of them are open-source, but also because the required hardware is more accessible in the real world.

Table 2. Simulated iteration time speedup over DeepSpeed on larger-scale QWen-VAL workloads (3 Tasks) and larger-scale clusters (256 GPUs).

Systems	QWen-VAL 30B	QWen-VAL 70B
Spindle	**1.34×**	**1.36×**
Spindle-Optimus	1.05×	1.09×
DistMM-MT	1.04×	1.04×
DeepSpeed	1×	1×

F Comparison on Single-Task Multi-Modal Workload

We also compare Spindle with baseline systems on single-task (ST) multi-modal (MT) scenario, which is a special case of MT MM training, as shown in Fig. 14. We are pleased to observe that even in the single-task scenario, Spindle outperformed SOTA systems by up to 48%. This is attributed to Spindle's fine-grained, operator-level resource allocation and scheduling, which recognize not only the inter-task workload heterogeneity but also intra-task operator workload variations — a capability beyond the reach of task-level strategies as well as SOTA systems. It's worth noting that DistMM-MT has similar performance to Spindle on ST MM scenario, which is reasonable, as DistMM-MT is specifically designed for single-task multi-modal workloads.

G Memory Consumption

We also conduct a comparative analysis of memory consumption between Spindle and the other competitors. Fig. 15 depicts the memory usage for each device in the scenario of Multitask-CLIP (4 tasks, 16 GPUs). Our findings indicate that Spindle generally exhibits lower memory consumption than SOTA systems such as Megatron-LM and DeepSpeed. This efficiency stems from Spindle's operator-level strategy and selective parameter storage feature, where only devices that activate a specific operator need to maintain its corresponding parameters, thereby minimizing redundant storage. Additionally, we've observed that task-level strategy Spindle-Optimus experiences significant memory imbalances. In contrast, Spindle maintains an excellent balance of memory consumption across devices, a success that is attributed to our device placement strategies.

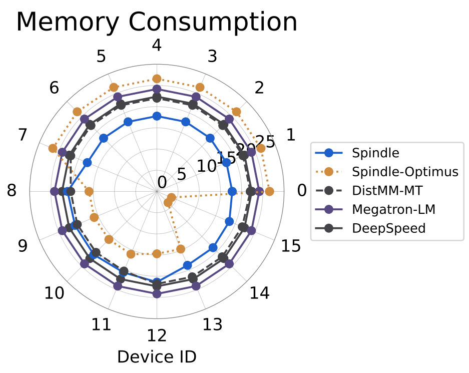

Figure 15. Memory consumption (GB) of each device in Multitask-CLIP (4 tasks, 16 GPUs). Points closer to the inner edge of the spider chart represent lower GPU peak memory usage.

H System Implementation Performance

We validate the system implementation performance of Spindle. Specifically, we implement a simple decoupled baseline on Spindle, Spindle-Seq, which allocates available devices to each task and execute tasks sequentially within each iteration, similar to SOTA systems, Megatron-LM and DeepSpeed. It reflects the implementation performance of our system without specific optimizations for MT MM workloads, i.e., without Spindle's flexible resource allocation and scheduling strategies. As shown in Fig. 16, we find that Spindle-Seq has similar system performance compared to SOTA systems without specific optimization for MT MM workloads.

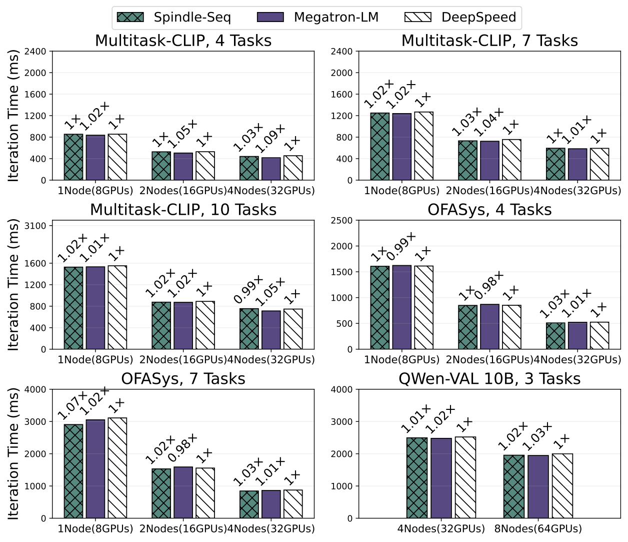

Figure 16. Comparison of Spindle-Seq with Megatron-LM and DeepSpeed.