Yanqi SHI, Peng LIANG, Hao ZHENG, et al., 2025. Automatic parallelism strategy generation with minimal memory redundancy. *Frontiers of Information Technology & Electronic Engineering*, 26(1):109-118.
https://doi.org/10.1631/FITEE.2300684

Automatic parallelism strategy generation with minimal memory redundancy

Key words: Deep learning; Automatic parallelism; Minimal memory redundancy

Yanqi SHI; Peng LIANG
E-mail: yqshi@nudt.edu.cn; peng_leung@nudt.edu.cn
ORCID: https://orcid.org/0000-0002-8899-1018
https://orcid.org/0000-0002-5590-5179

Artificial intelligence and memory wall

- In recent years, the rapid advancement of LLM has exacerbated the memory limitations of computing devices, thereby hindering the further development of these models. This challenge is commonly referred to as the "Memory Wall".

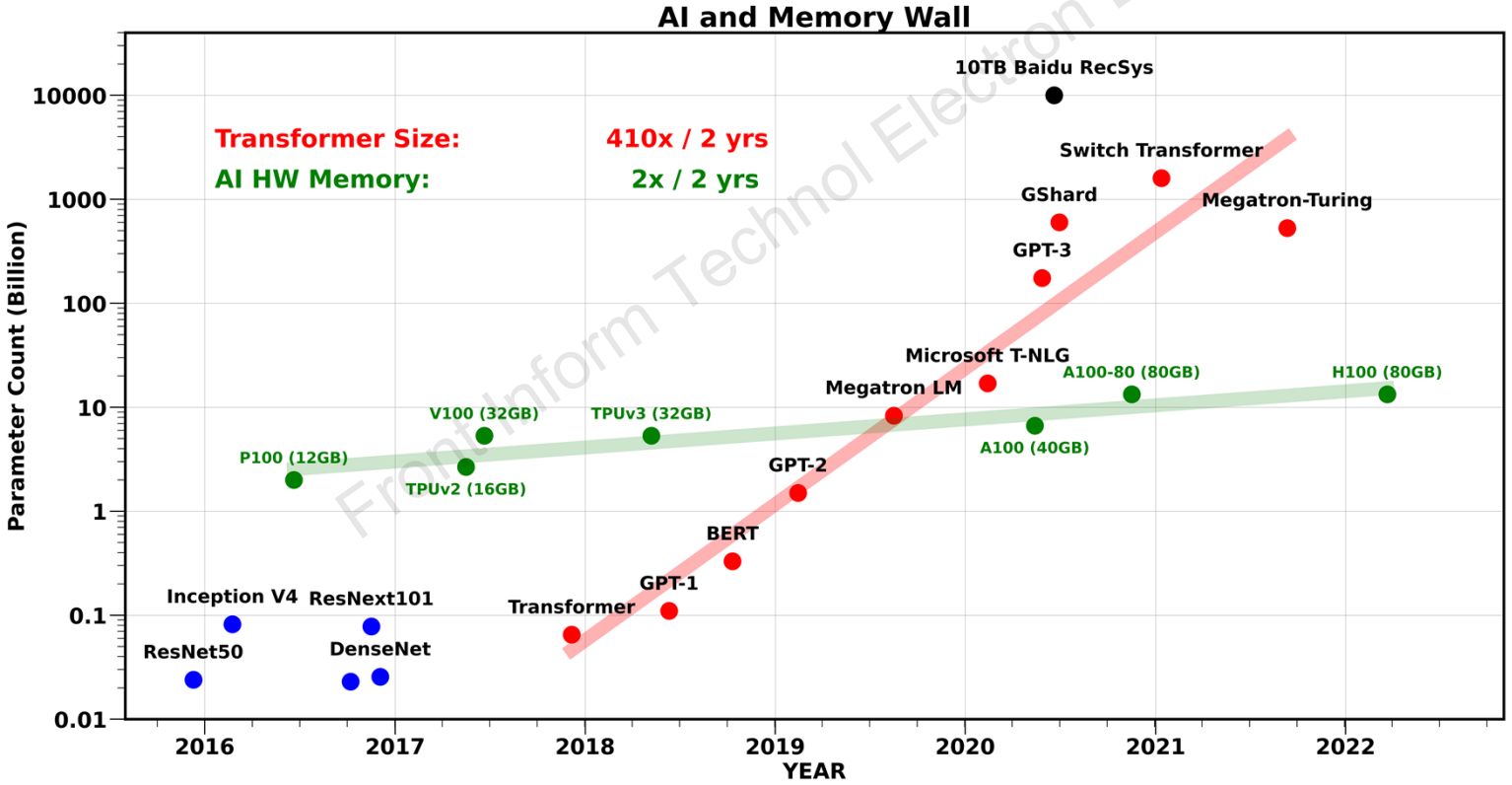

Figure From: Gholami A, Yao Z, Kim S, Mahoney MW, Keutzer K. AI and Memory Wall. RiseLab Medium Blog Post, University of Califonia Berkeley, 2021, March 29.

Main idea

- We propose an algorithm to search for memory-optimized operator-level parallelism strategies by analyzing the memory overhead of neural networks in distributed training. This enables us to further increase the hidden layer or batch size on hardware resources with limited memory.
- Initially, we instantiate a memory analyzer, enabling the computation of memory overhead for any specified strategy. Subsequently, we formalize the problem of strategy search.
- To accomplish this, we leverage an external solver specifically designed for integer linear programming (ILP) and employ it as a basis for arriving at the decisions, pertaining to optimal multi-dimensional intra-operator parallelism strategies that minimize memory usage.

1) Redundant memory cost model

- To accurately evaluate the memory cost arising from the use of different parallelism strategies, we first build a memory cost model to calculate the memory overhead of arbitrary intra-operator strategies.

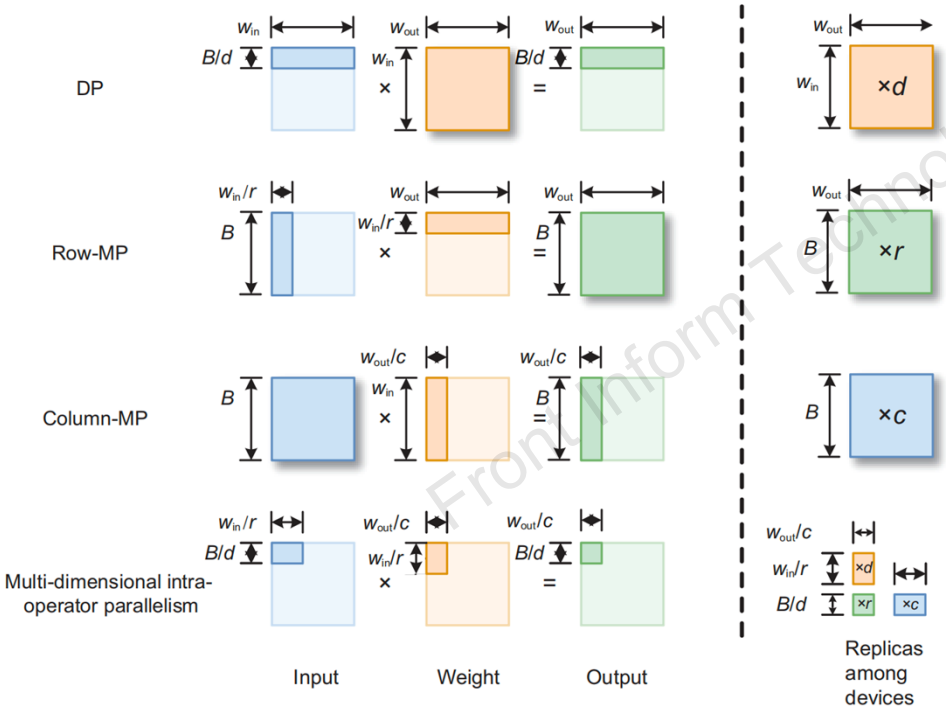

Fig. 1 Illustration of redundancy memory in intra-operator parallelism

$$M = \frac{1}{drc}(d(d-1)\text{size}(W) + r(r-1)\text{size}(Y) + c(c-1)\text{size}(X)). \quad (4)$$

Take the matrix multiplication of a fully connected hierarchy as an example.

2) Path search problem modeling

- We expand the original computation graph to generate a new auxiliary computation graph based on these candidate strategies. Each node in the auxiliary computation graph represents a parallelism strategy adopted for the computational operations corresponding to the original computation graph.

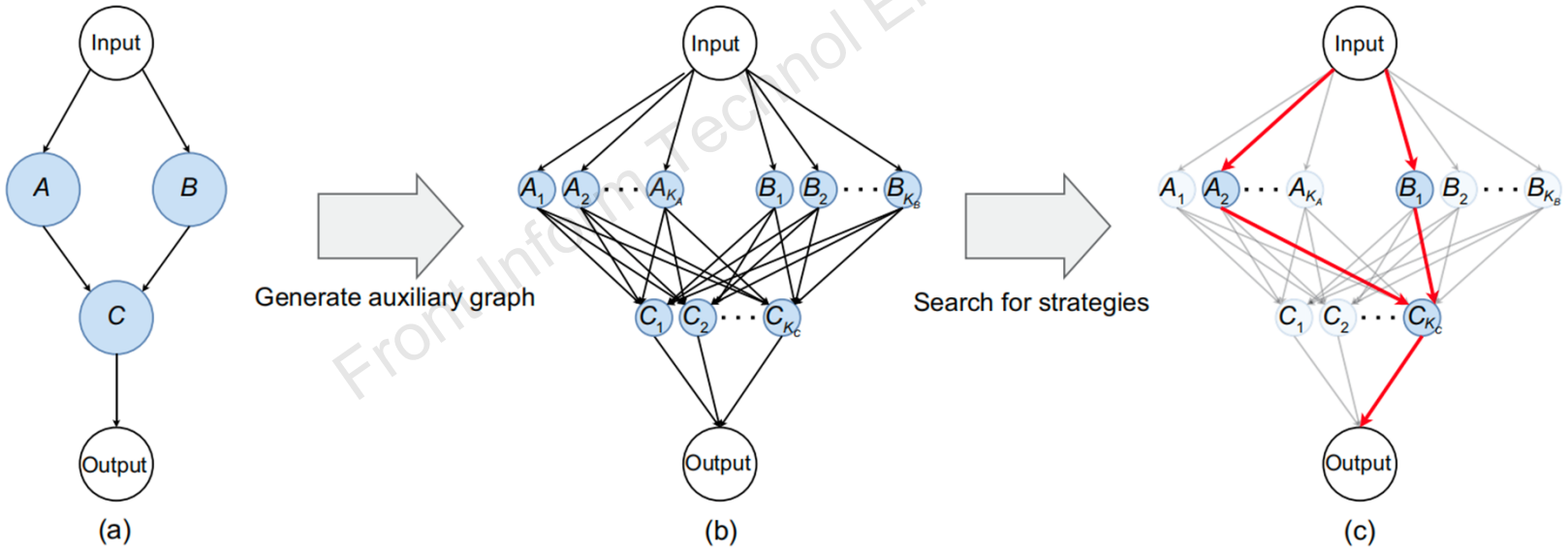

Fig. 2 Illustration of an auxiliary computation graph: (a) original computation graph; (b) auxiliary computation graph; (c) strategy decision example (References to color refer to the online version of this figure)

3) Strategy search algorithm

- To generate parallelism strategies more efficiently, we formalize the path search problem as an ILP problem:

$$\min \quad \sum_{(i,j)\in E_{\mathrm{A}}} B_{ij}M_{ij} + \alpha C(V_{\mathrm{A}}, E_{\mathrm{A}}) \tag{5}$$

$$\text{s.t.} \quad \sum_{v_{\mathrm{A}} \in V_{\mathrm{A}}} X_{v_{\mathrm{A}}} = 1, \tag{6}$$

$$\sum_{(i,v_{\mathrm{A}})\in E_{\mathrm{A}}} B_{iv_{\mathrm{A}}} = X_{v_{\mathrm{A}}} \cdot \mathrm{in\_degree}(v), \tag{7}$$

$$\sum_{(v_{\mathrm{A}},k)\in E_{\mathrm{A}}} B_{v_{\mathrm{A}}k} = X_{v_{\mathrm{A}}} \cdot \mathrm{out\_degree}(v), \tag{8}$$

$$B_{ij}, X_{v_{\mathrm{A}}} \in \{0,1\},\ \forall (i,j) \in E_{\mathrm{A}},\ \forall v_{\mathrm{A}} \in V_{\mathrm{A}},$$

$$C(V_{\mathrm{A}}, E_{\mathrm{A}}) = \sum_{v_{\mathrm{A}}\in V_{\mathrm{A}}} X_{v_{\mathrm{A}}} C_{v_{\mathrm{A}}} + \sum_{(i,j)\in E_{\mathrm{A}}} B_{ij}C_{ij}.$$

Existing high-performance ILP solvers can efficiently accelerate the process of solving integer programming problems.

Major results

Table 2 The memory overhead of our method during runtime compared to those of Megatron-LM, Alpa, and Colossal-Auto

Model	Device number	Memory overhead (MB)				Relative*		
		MLM	Alpa	C-Auto	Ours	MLM	Alpa	C-Auto
BERT-Large	2	216.06	148.81	331.63	185.05	0.86	1.24	0.56
1.7B	2	25 948.15	24 976.17	15 708.42	18 825.93	0.73	0.75	1.20
3.6B	2	39 620.04	37 473.76	16 136.24	27 634.30	0.70	0.74	1.71
7.5B	2	41 473.96	38 345.58	42 265.51	22 369.74	0.54	0.58	0.53
BERT-Large	4	219.22	158.90	331.64	148.70	0.68	0.94	0.45
1.7B	4	31 043.46	27 606.46	15 708.42	10 575.90	0.34	0.38	0.67
3.6B	4	41 419.42	35 070.54	16 136.24	13 568.87	0.33	0.39	0.84
7.5B	4	43 822.80	35 633.63	41 862.86	18 219.26	0.42	0.51	0.44
BERT-Large	8	199.81	177.01	335.73	128.05	0.64	0.72	0.38
1.7B	8	30 067.89	47 090.08	21 731.21	21 450.06	0.71	0.46	0.99
3.6B	8	40 104.50	57 417.39	24 169.72	26 477.73	0.66	0.46	1.10
7.5B	8	42 044.41	54 803.73	41 877.61	26 593.83	0.63	0.49	0.64

* Ratio of memory overhead of our method to that of other methods. MLM: Megatron-LM; C-Auto: Colossal-Auto

Conclusions

The key findings and contributions of this research can be summarized as follows:

- A novel algorithm was proposed that optimizes parallelism strategies, reducing memory redundancy and improving memory efficiency in large scale deep learning models by leveraging the extraction of the model's computational graph and employing auxiliary graphs.
- By conducting experiments on neural networks with various sizes within the Transformer class, the proposed parallel strategy, which aims to minimize memory redundancy as detailed in the paper, achieved up to 67% reduction in memory overhead. In contrast, the gap between the throughput of the proposed strategy and those of state-of-the-art methods is not large.