Sino-Russian Mathematical Challenge Fund Project Application Guide in 2025

1. Overall objective

The Sino-Russian Mathematical Challenge Fund aims to promote innovation and development in the field of mathematics and promote the practical application of mathematical achievements in the industry. Industry and academia collaborate to consolidate and release mathematical problems; At the same time, researchers from cross-border and different fields are encouraged to apply for projects to jointly solve mathematical problems in the industry.

2. Challenge direction

In 2025, the mathematical challenge fund plans to release 17 challenging problems, with funding awarded based on the quality of the proposals submitted. Each project is planned to support RMB 500,000. The project implementation cycle is generally one year.

No.	Challenge Puzzle List
1	High-Precision Vector Retrieval Algorithm based on Binary Quantization
2	Low-bit Attention Acceleration Algorithm
3	Expert Weight Compression Algorithm for MoE Models
4	Reuse of multi-document KV (Key-Value)
5	Out-of-Distribution (OOD) vector retrieval
6	A Memory-Efficient FTL Design for Flash-Based SSDs
7	Generalized Prefetching for High-Latency Scenarios
8	Direct mapping of Massive MIMO algorithms to low-level operator combinations
9	Massive MIMO signal tensor compression and direct calculation
10	Low Bitwidth Calculation for High Condition Number Matrix Decomposition and Inverse
11	Inversion of Structured Matrix
12	Parallel Computing with Low Complexity Based on Function Interpolation
13	Approximate error backpropagation/training algorithm for multilayer models

14	Dynamic Interference Channel Prediction—Parameter Estimation in Non-Stationary Environments
15	Efficient Computing - Low Bitwidth Forward Networks
16	Joint Optimization of Operator scheduling & Memory allocation
17	[Optimization problem] Route planning

Note: The intellectual property rights of the project-related achievements formed by the Funding project during the project research process, including but not limited to papers, works, source code, etc., shall be shared by the Funder, the applicant and the unit to which the applicant belongs to. Detailed terms are specifically agreed by the project funding agreement.

01 High-Accuracy Vector Retrieval Algorithm based on Binary Quantization

Background

Vector retrieval is a classic mathematical problem [1] with nearly 60 years of research history, which is widely applied in scenarios such as search and recommendation systems, retrieval-augmented generation for large language models, vector databases, and image retrieval. Vector retrieval refers to algorithms that find the vectors most similar to a given query vector from a set of vectors, defined as follows:

Definition of Vector Retrieval:

Given a d-dimensional vector $q \in \mathbb{R}^d$, retrieve the vector v most similar to q from a collection of n d-dimensional vectors $B \in \mathbb{R}^{n \times d}$, that is, to solve:

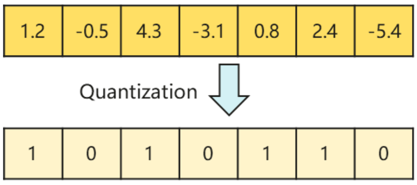
$$v = arg \max_{v \in B} sim(v, q)$$

where q is called the query vector, and B is called the vector base. The similarity between two vectors is usually defined in one of two ways:

1) Euclidean similarity: $sim(v, q) = -\|v - q\|_2$
2) Inner product similarity: $sim(v, q) = \langle v, q \rangle$

Current Situation

Binary quantization algorithms [2][3] are cutting-edge technologies in the field of vector retrieval and a hot research topic in both industry and academia. However, the Top-10 accuracy of existing 1BIT quantization-based vector retrieval algorithms is usually less than 80%, while practical applications typically require Top-10 accuracy of 95%~99%. Therefore, improving the accuracy of vector retrieval after binary quantization remains a pressing technical challenge for the industry.



Definition of Vector Retrieval Accuracy:

Given a d-dimensional vector $q \in \mathbb{R}^d$ and a collection $B \in \mathbb{R}^{n \times d}$ of n d-dimensional vectors, the Top-k accuracy of vector retrieval is defined as:

$$S_k = \frac{|R_k \cap R_k^*|}{k}$$

where $R_k = \{r_1, r_2, \ldots, r_k\}$ is the ids of vectors in B which are most similar to the vector q, retrieved by the brute-force search on binary quantized vectors; and $R_k^* = \{r_1^*, r_2^*, \ldots, r_k^*\}$ is the ids of vectors in B which are most similar to the vector q, retrieved by the brute-force search on original 32bit float-point vectors.

Problem Definition

Input:

1) Vector base: A set $B \in \mathbb{R}^{n \times d}$ of n d-dimensional real-valued vectors.
2) Query vectors: A set $B \in \mathbb{R}^{m \times d}$ of m d-dimensional real-valued vectors.
3) Vector similarity function: $sim(v, q)$, where v and q are vectors, and $sim(v, q)$ is restricted to either Euclidean similarity or inner product similarity, i.e., $sim(v, q) = -\|v - q\|_2$ or $sim(v, q) = \langle v, q \rangle$.

Output:

4) Design a binary quantization function f , whose input is any real-valued vector $x \in B$ from the vector base.
$$f(x) = x' \in \{a, b\}^d$$
where a, b are two fixed constants.
5) Also, design a similarity function g for binary-quantized vectors
$$g(x', q) = g(f(x), q) \in \mathbb{R}$$
such that for all query vectors $q \in Q$, the total error in vector similarity computation with binary quantization is minimized:
$$\min_{f,g} \sum_{q \in Q} \sum_{x \in B} |sim(x, q) - g(f(x), q)|$$

Demand

Design a binary quantization function f and a similarity function g for binary-quantized vectors, such that the Top-1, Top-10, and Top-100 accuracy of the vector retrieval based on binary quantization reaches at least 95%, with a challenge target of 99%.

- References:

[1] Bruch, Sebastian. Foundations of Vector Retrieval. Springer, 2024.

[2] Gao, Jianyang, and Cheng Long. "RaBitQ: quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search." Proceedings of the ACM on Management of Data 2.3 (2024): 1-27.

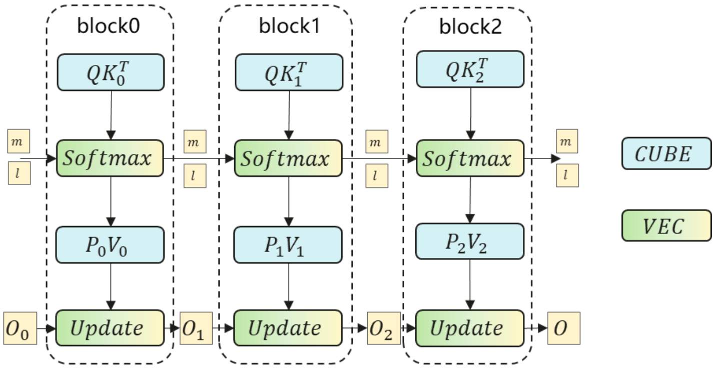
[3] Gao, Jianyang, et al. "Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search." Proceedings of the ACM on Management of Data 3.3 (2025): 1-26.

02 Low-bit Attention Acceleration Algorithm

Background

Attention mechanism is widely used in various machine learning tasks, which not only improves the model capability but also brings a large computational cost. Because the computation complexity is proportional to the square of the input sequence length, as the application develops towards a longer sequence, the impact of Attention computation gradually increases. For example, in a typical multimodal video generation task, Attention computation time accounts for more than 80%.

Attention is computed as a matrix-vector operation, with SoftMax and the additional Update step (in Flash Attention [1]) both being vector operations. Thus, accelerating Attention requires optimizing both matrix and vector operations.



Current Situation

There has been related research in the industry that applies low-bit quantization to QKV in Attention, leveraging low-bit matrix multiplication to accelerate the computation. However, since Flash Attention involves a significant amount of vector operations that cannot be accelerated by low-bit matrix multiplication, low-bit Attention faces a vector computation bottleneck. As a result, on chips with a high cube-to-vector (CV) ratio, it fails to deliver actual speedup.

Industry algorithm	Time	Calculation precision			
		QK^T	$Softmax$	SV	$Update$
FlashAttention2.0[1]	2023.07	FP16	FP32	FP16	FP32
FlashAttention3.0[2]	2025.06	FP16/FP8	FP32	FP16/FP8	FP32
SageAttention1.0[3]	2025.10	INT8	FP32	INT8	FP32
SageAttention2.0[4]	2025.11	INT4/INT8	FP32	INT8	FP32
SageAttention3.0[5]	2025.05	MXFP4	FP32	MXFP4	FP32

Problem Definition

Accelerate Attention computation, methods such as low-bit quantization (e.g., 4/8-bit) and approximate computation (e.g., piecewise or polynomial approximation) can be used, among others. However, it is required to maintain high fidelity for the final Attention output A_{approx}.

$$\cos\theta\left(\mathrm{vec}(\mathrm{Attention}(\mathrm{Q},\mathrm{K},\mathrm{V})),\mathrm{vec}(A_{approx}(Q,K,V))\right) > 0.999$$

Input

The Attention computation takes three tensor Q, K, V as inputs, $Q, K, V \in \mathbb{R}^{N\times d}$, where N is the input sequence length and d is the model's head dimension.

The input Q, K, V matrices are activations in large models and contain a small number of outlier values. Each entry is normally distributed with zero mean and standard deviation 1, but for 0.1% of entries are modeled with variance 10.

$$Q, K, V \sim 0.999 \cdot \mathcal{N}(0,1) + 0.001 \cdot \mathcal{N}(0,100)$$

Output

The Attention computation is defined as follows, primarily consisting of two matrix multiplications and a Softmax operation. The standard Scaled Dot-Product Attention formula is as follows:

$$\mathrm{Attention(Q,K,V)} = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

The Softmax operation is applied to each row of the input matrix, as shown in the formula below.

$$Softmax(x_{ij}) = \frac{exp^{x_{ij}-\max(x_i)}}{\sum_j exp^{x_{ij}-\max(x_i)}}$$

Demand

Accuracy Target：

Under the assumed distribution of *Q*, *K*, and *V* or based on actual model data, the accelerated approach should achieve a cosine similarity of 0.999+ between its output and that of the standard Attention (with FP16 matrix multiplications and FP32 vector computations).

$$\cos\theta\big(\mathrm{vec}(\mathrm{Attention(Q,K,V)}), \mathrm{vec}(A_{approx}(Q,K,V))\big) > 0.999$$

Performance Target：

Compared to standard Attention, the proposed method should theoretically or practically reduce computational complexity by at least 3×. **This reduction must apply to both matrix and vector computations.**

Different types of operations may incur different computational costs during implementation. For example, a typical processor can handle FP16 data at twice the throughput of FP32, while EXP operations, even at the same precision, are significantly more expensive than multiplications. Therefore, when evaluating computational complexity, it is important to distinguish between data types and operation types, as illustrated in the table below.

	Operation	FP32	FP16	FP8	FP4
CUBE	MAC	/	1	0.5	0.25
VECTOR	MUL/ADD/SUB	2	1	0.5	/
	MAX/MIN	2	1	0.5	/
	EXP/Reciprocal	8	4	0.5	/
	DIV	8	4	0.5	/

- References

[1] FlashAttention-2:Faster Attention with Better Parallelism and Work Partitioning

[2] FlashAttention-3:Fast and Accurate Attention with Asynchrony and Low-precision

[3] SageAttention: Accurate 8-Bit Attention for Plug-and-play Inference Acceleration

[4] SageAttention2: Efficient Attention with Thorough Outlier Smoothing and Per-thread INT4 Quantization

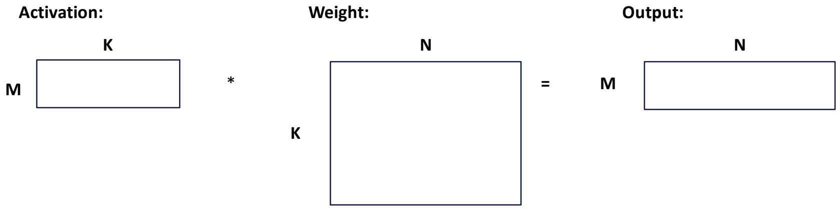
[5] SageAttention3: Microscaling FP4 Attention for Inference and An Exploration of 8-bit Training

03 Expert Weight Compression Algorithm for MoE Models

Background

DeepSeek is a popular application, with its cost-effective and efficient deployment proving indispensable. Currently, inference on the H800 cluster leverages expert parallelism (EP) and large Batch_Size to change the shape (M, N, K) of GeMM operator in MoE layer, thereby achieving full computing power of a single node. This approach has become the mainstream inference solution for central cluster systems.

In the FusionCube scenario, Batch_size is usually less than 10. During the inference phase, the following issues arise: Firstly, the degree of parallelism of MoE operators is insufficient, that is, the GeMM shape (M, N, K) is usually (1, N, K), and the operator performance is limited by the memory access bandwidth. Secondly, increasing Batch_size causes more experts to be activated, therefore the memory access bottleneck becomes more severe. Overall, it is urgent to design an algorithm to compress the activated expert weights and reduce the memory access bandwidth requirements.



Current Situation

Current research on large language model (LLM) weight compression includes the following areas:

1) Intra-matrix compression (single expert weight matrix): Techniques like W4A16 and W4A8 quantization schemes can reduce bandwidth requirements by half compared to original BF8 weights. While W4A16 has been deployed, it hasn't fully solved bandwidth issues, necessitating lower-bit quantization schemes.
2) Intra-layer compression (within the same MoE layer): This involves methods such as group-wise SVD compression [1] and residual-based Delta compression [2].
3) Inter-layer compression (between different MoE layers): Approaches include analyzing inter-layer SVD decomposition [3] and importance-aware pruning [4].
4) Other online pruning strategies.Currently, there is no direct analysis work specifically targeting the DeepSeek-R1/V3 model.

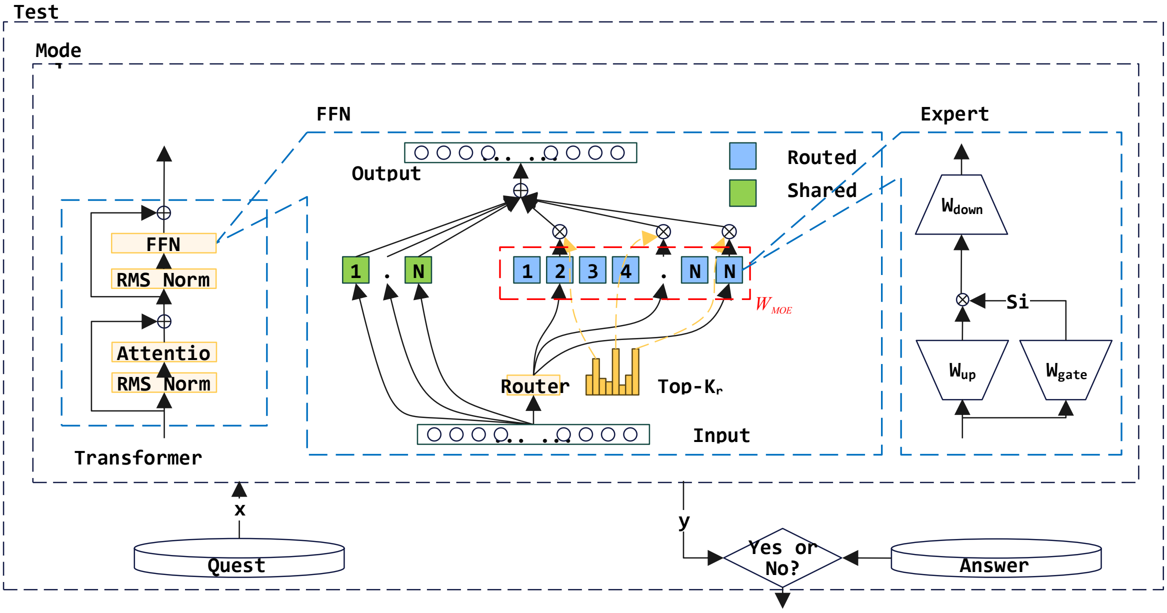
Design Expert Weight compression algorithm needs to pay attention to the following aspects:

1) The distribution of activated experts has hotspots, which are related to the input. Therefore, the compression algorithm should generalize well across diverse inputs.
2) Weight compression algorithm accepts lossy compression, but precision verification requires end-to-end testing. At present, there is no related derivation for precision error propagation

across the entire network.

Problem Definition

MoE module parameterized by $\boldsymbol{W}_{\mathbf{MoE}}$ is the important module of DeepSeek-based network model $\boldsymbol{M}(\boldsymbol{W}_{\mathbf{MoE}}, \boldsymbol{W}_{\mathbf{else}})$. It is composed of a mixture of experts that are activated via routing. Each expert computes its output as: $\boldsymbol{w_{down}}\left(\boldsymbol{silu}\left(\boldsymbol{w_{gate}}(\boldsymbol{x})\right) \cdot \boldsymbol{w_{up}}(\boldsymbol{x})\right)$ where silu(·) denotes the activation function.



Input:

1) DeepSeek Model: 671B parameters $\mathbb{R}^{671B}$ (654B in W_{MoE} and the remainder in W_{else}) with an inference function $M(W_{\text{MoE}}, W_{\text{else}})$.
2) Test queries: A set $\text{Quest} \in \mathbb{R}^{q \times m \times d}$ of q $m \times d$-dimensional real-valued vectors.

Output:

1) Design a MoE weight compression algorithm function f to produce compressed weights $W'_{MoE} = f(W_{MoE})$.
2) Based on the compressed weights W'_{MoE}, design a new inference function $M'(W'_{MoE}, W_{else})$ that minimizes the output error on Quest:

$$\min_{f,M'} \sum_{x \in \text{Quest}} |\mathrm{M}(\mathrm{W}_{\mathrm{MoE}}, \mathrm{W}_{\mathrm{else}})\mathrm{x} - \mathrm{M}'(\mathrm{W}'_{\mathrm{MoE}}, \mathrm{W}_{\mathrm{else}})\mathrm{x}|$$

Demand

Give the quantized DeepSeek V3 model with weight formats W4A16 or W4A8, explore expert similarity across MoE layers without introducing additional quantification solutions. Construct a MoE expert compression algorithm function f and corresponded inference function M' that achieves a notable improvements in computational efficiency while maintaining good precision and strong generalization based on a given dataset.

Precision requirement: After applying the weight compression algorithm, the accuracy on mainstream benchmark datasets remains stable, with less than 1% deviation.

Performance improvement: If Batch_size is set to 10, the input sequence length is 2 KB and the output sequence length is 1 KB, a theoretically analysis of key metrics, such as such as computational cost and memory access of the MoE operator, is required to prove that enabling the weight compression algorithm reduces its execution time by approximately 30%.

- References:

[1] Beyond Standard MoE: Mixture of Latent Experts for Resource-Efficient Language Models

[2] Delta Decompression for MoE-based LLMs Compression

[3] MoE-I^2: Compressing Mixture of Experts Models through Inter-Expert Pruning and Intra-Expert Low-Rank Decomposition

[4] Condense, Don't Just Prune: Enhancing Efficiency and Performance in MoE Layer Pruning

04 Reuse of multi-document KV (Key-Value)

Background

In traditional Retrieval-Augmented Generation (RAG) systems, the workflow consists of two phases:

1. **Retrieval Phase**: Fetch relevant documents $\mathrm{D} = \{d_1, d_2, \ldots, d_n\}$ from a corpus;
2. **Generation Phase**: Concatenate retrieved documents with user query q and system prompt p for LLM inference.

The efficiency bottleneck lies in recomputing attention over retrieved documents for every query, requiring $O(n^2)$ complexity during the Prefill stage. Document KV caching proposes storing attention intermediate results (KV cache) for potential reuse.

Problem Formulation

1. Variable Definitions:

Definition 1 (Input Components).

- *Documents:* $D = \{d_i\}_{i=1}^{n}$ *where each* $d_i = (t_1^i, t_2^i, \ldots, t_m^i)$*;*
- *User query:* $q = (t_1^q, t_2^q, \ldots, t_k^q)$*;*
- *System prompt:* $p = (t_1^p, t_2^p, \ldots, t_l^p)$*;*
- *Model with* ***L*** *layers and* ***H*** *attention heads per layer;*
- *Key/Value dimensions* $\mathrm{d}_k, \mathrm{d}_v$ *per head.*

Definition 2 (KV Cache). *For document* d_i*, its KV cache is*:

$$KV_i = \left\{KV_i^{l,h} | 1 \leq l \leq L, 1 \leq h \leq H\right\},$$

where each $KV_i^{l,h} = (K_i^{l,h}, |V_i^{l,h}) \in \mathbb{R}^{m\times d_k} \times \mathbb{R}^{m\times d_v}$.

2. Attention Computation

Problem 1 (Ideal Attention). For concatenated input $x = [p; q; d_1; \ldots; d_n]$, layer l head h computes:

$$Attention^{l,h}(Q^{l,h}, K^{l,h}, V^{l,h}) = softmax\left(\frac{Q^{l,h}\left(K^{l,h}\right)^T}{\sqrt{d_k}}\right),$$

where $Q^{l,h} = W_Q^{l,h}x, K^{l,h} = W_K^{l,h}x, V^{l,h} = W_V^{l,h}x$.

Problem 2 (KV Reuse Computation). With cached documents, the computation becomes:

$$\widetilde{K}^{l,h} = [k_p^{l,h}; k_q^{l,h}; k_1^{l,h}; \ldots; k_n^{l,h}],$$

$$\tilde{V}^{l,h} = [V_p^{l,h}; V_q^{l,h}; V_1^{l,h}; \ldots; V_n^{l,h}],$$

where subscript p, q denote real-time computed components.

Technical Challenges

Problem 3. The attention weight discrepancy between full recomputation and KV reuse:

$$A_{ij} = \frac{\exp(q_i \cdot k_{j'}/\sqrt{d_k})}{\sum_{j'} \exp(q_i \cdot k_{j'}/\sqrt{d_k})}$$

$$\tilde{A}_{ij} = \frac{\exp(q_i \cdot \tilde{k}_{j'}/\sqrt{d_k})}{\sum_{j'} \exp(q_i \cdot \tilde{k}_{j'}/\sqrt{d_k})}$$

where $\tilde{k}_j$ comes from cached computation, causing $A_{ij} \neq \tilde{A}_{ij}$.

Problem 4. Let *PE(pos)* be the positional encoding function:

$$Full\colon k_j = W_K(x_j + PE(j))$$

$$Cached\colon \tilde{k}_j = W_K(x_j + PE(pos_{cache}))$$

The cached positional encoding pos_{cache} doesn't match the actual position j in new context.

Problem 5. Define layer l output error $\epsilon^l = \|\tilde{h}^l - h^l\|$. For multi-document scenarios:

$$\epsilon^{l+1} \leq \|W_O^l\| \cdot \|\sigma'\| \cdot (\epsilon^l + \delta^l)$$

where δ^l is the current layer's attention error, causing error amplification across layers.

Demand

- **Cross-Document Attention**: Develop mechanism ensuring $|\tilde{A}_{ij} - A_{ij}| < \epsilon$.
- **Position Encoding:** Create adjustment method satisfying:

$$PE_{adjusted}(pos) \approx PE(pos_{actual})$$

- **Error Control:** Establish error bound:

$$\epsilon^{l+1} \leq C \cdot \epsilon^l,\ C < 1$$

- **Efficiency:** Achieve computation time:

$$T_{reuse} \leq \alpha \cdot T_{recompute},\ \alpha \in (0,1)$$

05 Out-of-Distribution (OOD) vector retrieval

Background

Approximate Nearest Neighbor Search (ANNS) is a core technology in databases, information retrieval, and large models. Traditional ANNS algorithms (e.g., HNSW, IVF) rely on the assumption that "neighbors of neighbors are still neighbors" implying consistent spatial locality between queries (Query) and data distributions (Key). However, in practice, queries may come from **Out- of-Distribution (OOD)** scenarios, causing:

- **Reduced efficiency**: OOD queries violate spatial locality, where Key- Key (KK) distances are much smaller than Query-Key (QK) distances.
- **Inadequate adaptability**: Existing methods (e.g., RoarGraph) require known distribution assumptions, harming in-distribution (ID) retrieval performance.

Problem Formulation

1. Notations:

- **Data distribution**: Set $K = \{k_i\}_{i=1}^{N} \subseteq \mathbb{R}^d$ following P_k.
- **Query distribution:** $q \sim P_Q$, potentially with $P_Q \neq P_K$ (OOD scenario).
- **Similarity metric**: Euclidean distance $D(q,k) = \|q-k\|_2$.

2. Input/Output:

- **Input:** Dataset K, graph index $G=(V,E)$, query $q \sim P_Q$.
- **Output:** Top- K approximate neighbors $\widehat{N}_{k(q)}$ with recall ≥ τ.

3. Mathematical Description:

Traditional ANNS assumes $P_Q = P_K$ with locality:

$$\forall q \sim P_Q, if\ k_i \in N_K(q), then\ N_K(k_i) \approx N_K(q)$$

Under OOD ($P_Q \neq P_K$), locality fails:

$$\exists q \sim P_Q, s.t. \min_{k \in K} D(q,k) \gg \min_{k_i,k_j \in K} D(k_i,k_j)$$

Traversal complexity T_{OOD} increases drastically:

$$T_{OOD} \approx \frac{\mathbb{E}_{q\sim P_Q}[D(q,K)]}{\mathbb{E}_{k_i,k_j\sim P_K}[D(k_i,k_j)]} \cdot T_{ID}$$

Demand

Design an OOD-robust vector retrieval method with:

1. **Efficiency**: $T_{OOD} \approx T_{ID}$.
2. **Distribution-agnostic:** No prior knowledge of P_Q.
3. **ID-preserving:** No degradation on P_K retrieval.

06 A Memory-Efficient FTL Design for Flash-Based SSDs

Background

With the rapid growth of SSD capacity (e.g., from TB to PB scales), the de- sign of Flash Translation Layer (FTL) faces significant challenges. Traditional FTL address mapping strategies are categorized into **block-level mapping** and **page-level mapping**:

- **Block-level mapping** manages logical-to-physical address translation at erase block granularity (typically 128KB 2MB). It has low memory over- head but poor write flexibility (overwrite requires erasing entire blocks), leading to severe write amplification and degrading SSD performance/lifetime.
- **Page-level mapping** operates at page granularity (e.g., 4KB), offering high write flexibility. However, its mapping table consumes excessive memory (e.g., 1GB for 1TB SSD), making it impractical for deployment.

Existing hybrid approaches (e.g., cached mapping tables) still suffer from:

1. **Double Read Problem**: If mapping tables reside in flash, each read requires accessing the table (first read) followed by data (second read), significantly increasing latency.
2. **Limitations of Learned Indexes**: While learned indexes (e.g., piece- wise linear models) can compress mapping tables, SSD’s limited computational resources cannot support complex models, and dynamic updates are inefficient for frequent writes.

Thus, a **low-memory**, **high-performance FTL scheme for large-capacity SSDs** is urgently needed.

Problem Formulation

1. Notations:

C: SSD physical capacity (bytes)

L: Logical address space size (bytes)

P: Physical page size (e.g., 4KB)

$B = N \times P$: Physical block size (e.g., B = 256KB when N = 64)

$l \in \{0,1,\dots,[L/P]-1\}$: Logical Page Number (LPN)

$p \in \{0,1,\dots,[C/P]-1\}$: Physical Page Number (PPN)

M: LPN → PPU ∪ {NULL}: Mapping function

2. Objectives:

Design *M* satisfying:

1. **Memory Constraint:**

$$Mem(M) \leq \alpha \cdot [L/P] \cdot \log_2[C/P],\ \ \alpha \ll 1$$

2. **Performance Constraints:**

$$Latency_{read} \leq Latency_{flash} + \delta$$

$$WA(M) \leq \beta$$

3. **Reliability:** Crash consistency for mapping tables.

Mathematical Challenges

- **Mapping Table Compression:** Find encoding function $\mathcal{F}$

$$\mathcal{F}(M) = \langle \theta, \varepsilon \rangle,\ \ |\theta| + |\varepsilon| \ll |M|$$

- **Double Read Avoidance:**

$$Pr[M(l) = \mathcal{F}^{-1}(\theta, l)] \geq 1 - \epsilon, \quad (\epsilon \to 0)$$

Demand

Propose an FTL scheme achieving:

- **Low Memory:** Reduce memory usage to <10% of page-level mapping ($\alpha \leq 0.1$)
- **High Performance:**

  - 99% single-flash-access reads ($\epsilon \leq 0.01$)

  - Write amplification WA ≤ 2
- **Lightweight Computation:** $O(1)$ or $O(\log n)$ complexity

07 Generalized Prefetching for High-Latency Scenarios

Background

Tape media, with its high capacity/cost ratio, has become the mainstream choice for archival and data protection scenarios. However, its minute-level seek latency and high turnaround latency limit its application in warm-cold data scenarios. To overcome this limitation, **Tape-SSD Hybrid Storage** adopts a hybrid architecture of **SSD cache + tape**, combined with data prefetching algorithms, aiming to reduce access latency to the hundred-millisecond level. Traditional prefetching algorithms (e.g., sequential stream, stride prediction) are designed for multi-level storage media like DRAM, SSD, and HDD. The unique characteristics of Tape-SSD Hybrid Storage (e.g., extremely high prefetching overhead, long latency, and limited computational resources) demand novel prefetching algorithms to improve SSD cache hit rates and reduce access la- tency under diverse workloads.

Problem Formulation

1. Variable Definitions:

- Access sequence:$S = \{a_1, a_2, \dots, a_n\}$, where a_i denotes the logical block address (LBA) of the i-th access.
- Prefetch window size: w, representing the maximum number of contiguous blocks to prefetch.
- SSD cache capacity: C, measured in number of data blocks.
- Cache hit function: $H(a_i) \in \{0,1\}$, where $H(a_i) = 1$ if a_i hits in SSD, otherwise 0.
- Prefetch threshold: θ, triggering prefetch when prediction confidence exceeds this value.
- Prefetch algorithm: $f: S \to P,$ where $P = \{p_1, p_2, \dots, p_k\}$ is the predicted sequence of blocks to prefetch.

2. Optimization Objectives:

Maximize SSD cache hit rate while minimizing average access latency T_{avg}:

$$Maximize \frac{1}{n}\sum_{i=1}^{n} H(a_i), \quad Minimize T_{avg} = \frac{1}{n}\sum_{i=1}^{n} T(a_i)$$

where $T(a_i)$ is the latency of accessing a_i: $T(a_i) = T_{SSD}$ if hit, otherwise $T(a_i) = T_{tape}$

3. Constraints:

- **Prediction length constraint**: Due to high tape seek latency, longer sequences must be predicted to hide latency:

$$|P| \geq w, \quad w \gg 1$$

- **Computational resource constraint**: The algorithm's complexity must meet real-time requirements:

$$Time(f(S)) \leq T_{max}$$

- **Cache capacity constraint**: Total prefetched data cannot exceed SSD cache capacity:

$$\sum_{j=1}^{k} \mathbb{I}(p_j \notin SSD) \leq C - CurrentCacheUsage$$

4. Mathematical Formulation of Key Challenges:

Challenge 1 (Generalization): The algorithm must adapt online to access pattern changes:

$$f \text{ must satisfy } \forall S_{new}, \lim_{|S_{new}| \to \infty} \frac{\sum_{a_i \in S_{new}} H(a_i)}{|S_{new}|} \geq \eta$$

where η is the target hit rate threshold.

Challenge 2 (Complex pattern capture): Requires modeling high order Markovity or long-range dependencies:

$$P(a_{t+1} | a_t, a_{t-1}, \dots, a_{t-m}) \approx P(a_{t+1} | State(a_{1:t}))$$

where m is the history window, and State may be encoded by RNN/Transformer.

Challenge 3 (Latency sensitivity): Prefetch timing must avoid:

$$\text{PrefetchAt } t \Leftrightarrow \exists P, s.t. \frac{|P \cap S_{t+1:t+w}|}{|P|} \geq 0$$

Challenge 4 (Resource constraints): Algorithm complexity must be sublinear:

$$\text{Time}(f) = O(\log n) \ or \ O(1)$$

Demand

Design an online adaptive prefetching algorithm for Tape-SSD Hybrid Storage that satisfies:

- **High generalization:** Maintains high hit rates ($\eta \geq 90\%$) under diverse workloads (random/sequential/mixed).
- **Long-sequence prediction:** Supports prediction window $w \geq 10$ to hide tape latency.
- **Low computational overhead:** Single prediction time $\leq$ 1ms.
- **Online learning:** No offline training required, dynamically adapts to access patterns.

Solving this problem will enable broader applications of Tape-SSD Hybrid Stor- age in warm-cold scenarios, significantly reducing storage costs.

08 Optimal Mapping of Cross-Domain Matrix Computations to Hardware Acceleration Operators

Background

Massive MIMO signal processing in wireless communication and Transformer attention computation in AI involve substantial matrix computations. These computations have become a central part of the computational workload for both systems. Wireless signal processing encompasses operations such as matrix inversion, eigenvalue decomposition, and fast Fourier transform (FFT), whereas AI Transformer algorithms rely on large-scale matrix multiplication-addition, normalization, and Softmax. While AI chips offer significant acceleration through dedicated hardware operators (e.g., matrix multiplication-addition units and vector units), there is currently a lack of a unified optimization framework for cross-domain matrix computation tasks. This challenge seeks to establish a unified mathematical model that efficiently maps matrix computations in wireless communication and AI to underlying hardware operators, thereby minimizing computational costs while maintaining required precision levels.

Current Result

1. Wireless signal processing:

   Matrix computation can be optimized through mathematical principles. Reducing computational complexity involves techniques such as direct matrix decomposition or inversion, or the use of iterative methods like Neumann series. Common precision formats include FP32/16, int32/16, and more.

2. AI algorithms:

   Operator fusion: Matrix multiplication-addition and activation functions are combined into a single operator (e.g., GEMM+ReLU).

   Quantization acceleration: The computational workload is reduced by using lower-precision formats (FP16 or INT8). However, the trade-off between precision loss and acceleration must be carefully managed.

   Hardware acceleration: Ascend chips support matrix multiplication-addition (GEMM) for operator acceleration, leading to a speedup of around 100 times with FP32 precision.

   Nonlinear vector operations such as Softmax and ReLU achieve an acceleration of around 10 times, while scalar operations like constant multiplication do not experience any acceleration.

3. Challenges:

(1) Matrix computations are optimized separately for wireless communication and AI. The lack of a unified framework on the same hardware platform leads to redundant operator calls and inefficient use of computing resources.

(2) Accelerating various underlying operators necessitates substantial development efforts and often results in suboptimal operator combinations and constrained computational performance.

Challenge Description

Given massive MIMO signal processing in wireless communication and Transformer attention computation in AI:

1. Massive MIMO signal processing in wireless communication:

$$\text{Uplink MIMO detection: } \boldsymbol{W} = \boldsymbol{H}\,(\boldsymbol{H}\,\boldsymbol{H}^{H} + \boldsymbol{R_{uu}})^{-1}$$

$$\text{Downlink beamforming: } \boldsymbol{W}_n = \boldsymbol{V}_n(\boldsymbol{V}_n^{H}\boldsymbol{V}_n + \delta^2\boldsymbol{I})^{-1}$$

2. Transformer attention computation in AI:

$$\boldsymbol{Q} = \boldsymbol{X} \times \boldsymbol{W_Q}, \boldsymbol{K} = \boldsymbol{X} \times \boldsymbol{W_K}, \boldsymbol{V} = \boldsymbol{X} \times \boldsymbol{W_V}$$

$$Attention(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q} \times \boldsymbol{K^T}}{\sqrt{\boldsymbol{d_k}}}\right) \times \boldsymbol{V}$$

$$\sigma(\boldsymbol{Z})_i = \frac{e^{\boldsymbol{Z}_i}}{\sum_{j=1}^{K} e^{\boldsymbol{Z}_j}}, \boldsymbol{Y} = \frac{\boldsymbol{X} - E[\boldsymbol{X}]}{\sqrt{Var[\boldsymbol{X}] + \varepsilon}} \times \gamma + \beta$$

This computation task can be decomposed into a combination of underlying operators (GEMM, vector operations, and scalar operations). The goal is to determine the optimal operator combination that satisfies the requirements and constraints outlined below.

Mathematical modeling:

Define a computational graph $\boldsymbol{G} = (\boldsymbol{V},\boldsymbol{E})$, where nodes $\boldsymbol{V}$ represent matrix operations and edges $\boldsymbol{E}$ represent data dependencies. Decompose each node $\boldsymbol{v} \in \boldsymbol{V}$ into a combination of hardware operators $\boldsymbol{s_v} = \{\boldsymbol{s_1}, \boldsymbol{s_2}, \dots, \boldsymbol{s_k}\}$, where $\boldsymbol{s_i}$ can be a GEMM, vector operation, or scalar operation.

Objective function:

$$min \sum_{v \in V} \sum_{s \in S_v} T(\boldsymbol{s}) \quad s.t. Error(\boldsymbol{S_v}) \leq \epsilon$$

where $T(\boldsymbol{s})$ represents the computation time of operator $\boldsymbol{s}$, and $Error(\boldsymbol{S_v})$ denotes the precision loss of the overall operator combination.

1. Precision constraint: $\varepsilon \leq 10^{-3}$ (ε indicates the relative error between the output result and the floating-point computation result).

2. Minimal computational cost: Ensure that the total computation time is minimized.

Demand

1. Establish a unified optimization model for cross-domain matrix computations to quantify the relationships between hardware acceleration benefits and precision loss.

2. Design algorithms such as dynamic programming or integer programming to solve for the optimal operator combination.

3. Implement optimization for wireless tasks (MIMO detection and downlink beamforming algorithms) and AI tasks (Transformer attention computation) on the Ascend platform, with experimental verification demonstrating a reduction of ≥ 30% in computation time.

- **References**

[1] W. H. Chen, "Matrix Computation for Wireless Communications," IEEE Trans. Signal Process., 2020.

[2] T. L. Marzetta, "Large-Scale Antenna Systems," Foundations and Trends® in Signal Processing, 2014.

[3] A. Vaswani et al., "Attention Is All You Need," NeurIPS, 2017.

[4] S. Venkataramani et al., "Optimized GEMM for Deep Learning," MLSys, 2021.

[5] Huawei Ascend AI Processor Architecture, Huawei White Paper, 2022.

[6] J. Pool et al., "What's Inside NVIDIA A100 Tensor Cores?" arXiv:2103.05111, 2021.

[7] Y. Yan et al., "Approximate Matrix Multiplication for Wireless Systems," IEEE JSTSP, 2021.

[8] M. Gupta et al., "Deep Learning with Low Precision," ACM Comput. Surv., 2018.

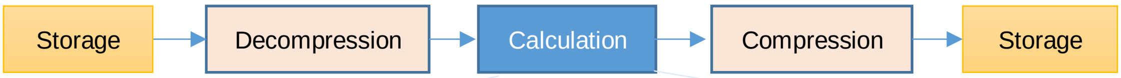
09 Massive MIMO Signal Tensor Compression and Direct Calculation Based on Compressed Data

Background

The significant development of 5.5G communications technologies has shifted our focus toward the application of massive MIMO systems on 6 GHz and higher frequency bands. The number of antennas has been increased exponentially (for example, to hundreds or even thousands of antenna arrays) to enhance spectral efficiency and network capacity. This, in turn, has led to a cubical increase in calculation complexity $O(N^3)$ of matrix operations (such as channel estimation, pre-coding, and detection) in massive MIMO signal processing, as well as a quadratical increase in storage complexity $O(N^2)$. The rising complexity has resulted in rocketing demands for computing resources, which becomes a major bottleneck for system deployment. To reduce the demand for storage space, matrix compression techniques have been employed. However, the complexity issue persists because the existing compression framework needs to restore the compressed data to full-dimensional signals for calculation.

Current Result

Compressed data needs to be restored to full-dimensional matrices for subsequent processing. Consequently, the calculation complexity is not significantly reduced. Additionally, isolated compression and calculation make it hard to leverage the structural characteristics of compressed data to optimize the calculation process.



(1) Receiver equalization coefficient: $n = 1, \dots, 68,\ k = 1, \dots, 1632$

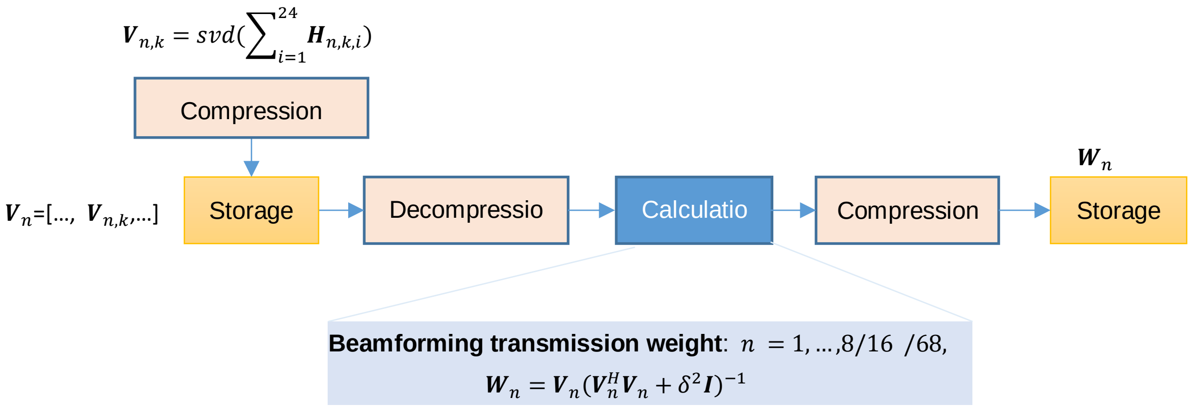
$$\boldsymbol{W}_{n,k} = \boldsymbol{H}_{n,k}\left(\boldsymbol{H}_{n,k}\boldsymbol{H}_{n,k}^{H} + \boldsymbol{R}_{\boldsymbol{uu},n}\right)^{-1}$$

(2) Beamforming transmission weight: $n = 1, \dots, 68,$

$$\boldsymbol{W}_{n} = \boldsymbol{V}_{n}(\boldsymbol{V}_{n}^{H}\boldsymbol{V}_{n} + \delta^{2}\boldsymbol{I})^{-1}$$

Challenge Description

1. Calculation and compression for downlink beamforming


$V_{n,k} = svd(\sum_{i=1}^{24} H_{n,k,i})$
Compression
V_n=[…, $V_{n,k}$,…]
Storage
Decompressio
Calculatio
Compression
W_n
Storage
Beamforming transmission weight: $n = 1,\ldots,8/16\ /68,$
$W_n = V_n(V_n^H V_n + \delta^2 I)^{-1}$

where $V_n \in C^{256\times L}, n = 1,\ldots,8/16/32/64$; $W_n \in C^{256\times L}$, $n = 1,\ldots,8/16/32/64$; L denotes the number of streams scheduled in the downlink ranging from 1 to 100. For this challenge, we have $L = 64$.

Compression target: The storage size of V_n and W_n should be less than 1/10 of the original FP32 data.

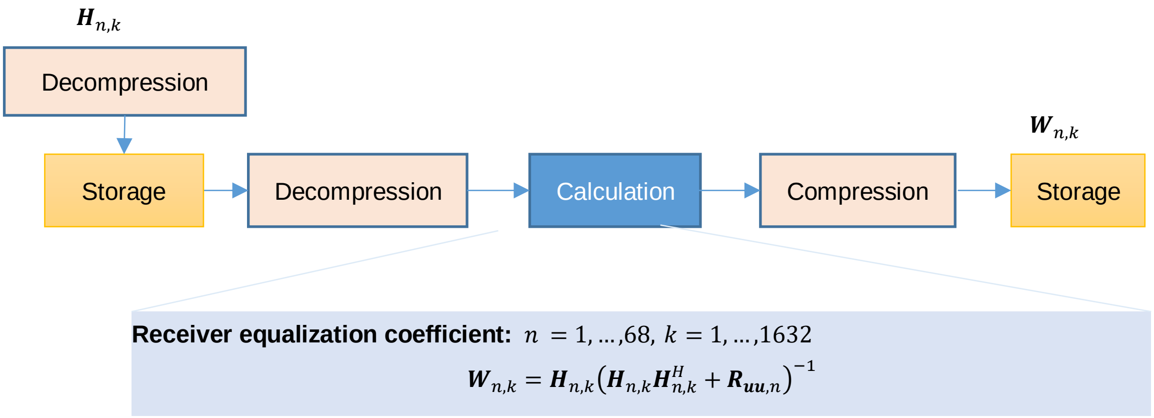
Calculation target: $W_n = V_n(V_n^H V_n + \delta^2 I)^{-1}$, $n = 1,\ldots,8/16/32/64$

Calculation accuracy: $|W'_n - W_n|^2 < \varepsilon$, with the normalized MSE of the calculation result error being less than –40 dB (i.e., the normalized amplitude error < 0.01).

Calculation complexity: The complexity of the compression/decompression and calculation process should be less than 1/3 of that of the uncompressed calculation baseline.

Note: Channel **H** of a specific channel configuration type can be generated using the open-source software Quadriga.

2. Calculation and compression for uplink receiver equalization


$H_{n,k}$
Decompression
Storage
Decompression
Calculation
Compression
$W_{n,k}$
Storage
Receiver equalization coefficient: $n = 1,\ldots,68,\ k = 1,\ldots,1632$
$W_{n,k} = H_{n,k}\left(H_{n,k}H_{n,k}^H + R_{uu,n}\right)^{-1}$

where $H_{n,k} \in C^{256\times L}, n = 1,\ldots,8/16,\ k = 1,\ldots,8/16*12$; $W_{n,k} \in C^{256\times 50}$, $n = 1,\ldots,8/$

$16,\ k = 1, \ldots, 8/16 * 12$; $\boldsymbol{R}_{\boldsymbol{uu},n} \in \mathrm{C}^{256\times 256}, n = 1, \ldots, 68$; L denotes the number of streams scheduled in the uplink. For this challenge, we have $L = 32$.

Compression target: The storage size of $\boldsymbol{H}_{n,k}$ and $\boldsymbol{W}_{n,k}$ should be less than 1/10 of the original FP32 data.

Calculation target: $\boldsymbol{W}_{n,k} = \boldsymbol{H}_{n,k}\left(\boldsymbol{H}_{n,k}\boldsymbol{H}_{n,k}^{H} + \boldsymbol{R}_{uu,n}\right)^{-1}$, $n = 1, \ldots, 8/16/64$

Calculation accuracy: $|\boldsymbol{W'}_n - \boldsymbol{W}_n|^2 < \varepsilon$, with the normalized MSE of the calculation result error being less than –40 dB (i.e., the normalized amplitude error < 0.01).

Calculation complexity: The complexity of the compression/decompression and calculation process should be less than 1/3 of that of the uncompressed calculation baseline.

Note: Channel **H** of a specific channel configuration type can be generated using the open-source software Quadriga.

Compression and decompression can be integrated to satisfy calculation accuracy requirements by reducing the storage space of data volume (for example, through image and video compression). All kinds of compression approaches can be used, including those reducing valid data through low-rank matrix/tensor decomposition or the transform domain and those reducing data volume by adopting lower-bit storage.

Demand

Reduce the normalized MSE of the calculation result error to less than –40 dB (i.e., the normalized amplitude error < 0.01).

1. Reduced amount of data to be stored: The amount of data to be stored in the main storage matrices sent during uplink receiver equalization and downlink beamforming should be less than 1/10 of that of the original FP32 data.

2. Reduced calculation complexity: The complexity of the compression/decompression and calculation process should be less than 1/3 of that of the uncompressed calculation baseline.

10 Low-Bitwidth Computing for Matrix Decomposition and Inversion with a High Number of Conditions

Background

Common matrix operators used in wireless communications include SVD, Cholesky decomposition, QR decomposition, and conjugate matrix inversion. These operators typically involve higher computing overheads in transceivers, as well as larger chip areas and more energy consumption, especially in demanding conditions, such as strong interference and higher-order modulation schemes. This is because larger bitwidths are required for matrices with a higher number of conditions to achieve precise matrix decomposition in such scenarios.

Challenge Description

A bottom-layer computing and bitwidth design solution encompassing matrix condition count identification, matrix decomposition algorithm design, and data representation methods needs to be developed to achieve low-complexity matrix decomposition and inversion. The bottom-layer matrix computing on chips involves SVD, Cholesky decomposition, QR decomposition, and matrix inversion.

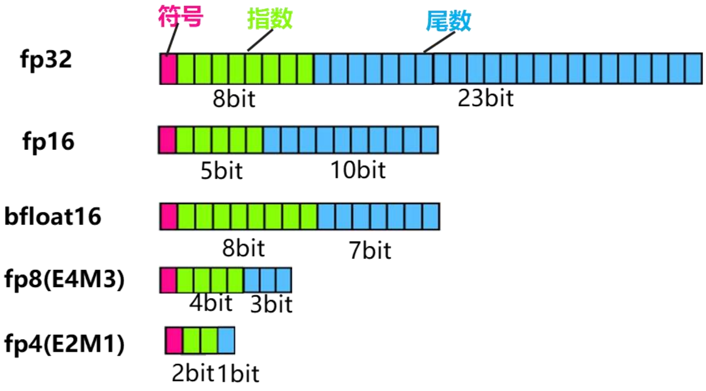
$$U * S * V^H = SVD(A)$$
$$L * L^H = CHOL(A)$$
$$Q * R = QR(A)$$
$$A^{-1} = inv(A)$$

Low-complexity matrix condition count identification is key to dynamic bitwidth design. A system-level design with minimum complexity can be achieved by employing smaller bitwidths for matrices with fewer conditions and larger bitwidths for matrices with more conditions.

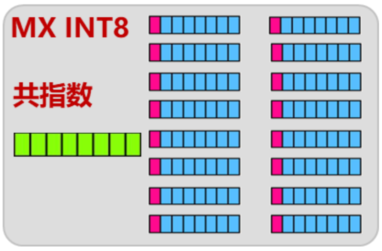
Innovative matrix decomposition algorithms need to be designed based on common ones, including Jacobi, block Cholesky decomposition, and modified Gauss-Seidel. These algorithms must not only feature low multiplication and addition complexity in bottom-layer matrix computing, but also have smaller bitwidths, which affect the overall computing overhead. Additionally, a unified algorithm architecture needs to be implemented for matrices with different numbers of conditions, substantially reducing the hardware overhead.

The representations of bottom-layer data are also key to bitwidth design. Common data representations, including FP, BFP, ANT, and MX, require different bitwidths, which affects the overall computing overhead.

Take FP16 as an example. It has 16 bits, including 1 sign bit, a DAGC factor (5 bits), and 10 data bits. For details, see [1]. The following figure illustrates other data representations and their bitwidths.



Bottom-layer data can also be represented by block quantization. Take MX INT8 as an example. As shown in the following figure, eight complex numbers share one exponent factor, significantly saving the exponent storage space. For details, see [2].



Demand

Develop decomposition/inversion algorithms for matrices of any dimension and verify the performance and complexity gains of these algorithms based on the numbers of matrix dimensions and conditions in the following table.

The algorithms must identify matrices with a high number of conditions for all the four matrix operators and reduce the computing complexity by 75% compared with the baseline, while achieving a decomposition error of no more than 1%.

Note: Bitwidth is considered in the evaluation of computing complexity. For instance, given the same number of complex multiplications, the computing complexity of FP32 is twice that of FP16.

The definition of error is described as follows (with SVD decomposition as an example):

$$\frac{||H-USV||_F}{||H||_F}+||U^HU-I||_F+||V^HV-I||_F$$

It considers both the reconstruction error and the orthogonality of the unitary matrix.

Matrix Decomposition/ Inversion Operator	**Number of Matrix Dimensions**	**Number of Matrix Conditions**	**Baseline Algorithm/Bitwidth**	**Goal**
SVD decomposition	64 x 4, 128 x 4, and 256 x 4	10^4~10^5	Jacobi (FP32)	The decomposition error does not exceed 1%, and the computing complexity is reduced by 75% compared with the baseline.
Cholesky decomposition	Conjugate symmetric matrices: 64, 128, and 256	10^4~10^5	Block Cholesky decomposition [3] (FP32)	
QR decomposition	80 x 16, 160 x 32, and 320 x 64	10^2~10^3	Modified Gauss-Seidel [4] (FP32)	
Conjugate symmetric inversion	Conjugate symmetric matrices: 64, 128, and 256	10^4~10^5	Block Cholesky decomposition [3] + block triangular matrix inversion (FP32)	

- **References**

[1] https://en.wikipedia.org/wiki/Half-precision_floating-point_format

[2] https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf

[3] Chen, Jianping, *et al.* "Block Algorithm and Its Implementation for Cholesky Factorization." (2013).

[4] Barlow, Jesse L. "Block Modified Gram--Schmidt Algorithms and Their Analysis." SIAM Journal on Matrix Analysis and Applications 40.4(2019):1257-1290.

11 Inversion of Structured Matrices

Background

Conventional matrix inversion algorithms, such as block Cholesky decomposition, typically employ serial computation involving the inversion of numerous identical blocks across submatrices. However, the results of these inversions are often dependent on the inversion operations performed on submatrices in both previous and subsequent iterations, preventing their reuse across blocks. Therefore, minimizing redundant computations between identical blocks across matrices is essential for reducing overall computational complexity.

Challenge Description

Input: Assume that $\boldsymbol{R} \in \mathbb{C}^{N*N}$ is a conjugate symmetric matrix and $\boldsymbol{H} \in \mathbb{C}^{N*L}$ is a complex matrix. Solution: Construct a structured matrix $\boldsymbol{P}_g, \boldsymbol{Q}_g$, where $g = 1,2,\dots,G$, so that the column space of $\boldsymbol{W}_0 = \boldsymbol{R}^{-1} * \boldsymbol{H}$ has maximum similarity to that of $\boldsymbol{W}_1$ in the following matrix:

$$\max tr(\boldsymbol{U}_0^H \boldsymbol{U}_1 \boldsymbol{U}_1^H \boldsymbol{U}_0)$$

$$\boldsymbol{W}_1 = [\boldsymbol{P}_1(\boldsymbol{P}_1^H \boldsymbol{R} \boldsymbol{P}_1)^{-1} \boldsymbol{P}_1^H * \boldsymbol{H} \boldsymbol{Q}_1, \dots, \boldsymbol{P}_G(\boldsymbol{P}_G^H \boldsymbol{R} \boldsymbol{P}_G)^{-1} \boldsymbol{P}_G^H * \boldsymbol{H} \boldsymbol{Q}_G] \in \mathbb{C}^{N*L}$$

where, $\boldsymbol{U}_0, \boldsymbol{U}_1$ represent the matrices formed by the orthogonal bases of the column spaces of $\boldsymbol{W}_0, \boldsymbol{W}_1$, respectively. $\boldsymbol{P}_g, \boldsymbol{Q}_g$ is subject to the following constraints:

- $\boldsymbol{P}_g$ is a column extraction matrix with the dimension of $N * M_g$, where $N > M_g$. Each column has only one non-zero element whose value is 1, and the columns are orthogonal to each other, that is, $\boldsymbol{P}_g(n,m) \in \{0,1\}, \boldsymbol{P}_g{}^H * \boldsymbol{P}_g = \boldsymbol{I}$.
- $\boldsymbol{Q}_g$ is a column extraction matrix with the dimension of $L * L_g$, where $L = \sum_g L_g$. Each column has only one non-zero element whose value is 1, and the columns are orthogonal to each other, that is, $\boldsymbol{Q}_g(n,m) \in \{0,1\}, \boldsymbol{Q}_g{}^H * \boldsymbol{Q}_g = \boldsymbol{I}$. Additionally, in the case of $t \neq g$, the column spaces of $\boldsymbol{Q}_g$ and $\boldsymbol{Q}_t$ are orthogonal. Simply put, the matrix $[\boldsymbol{Q}_1, \dots, \boldsymbol{Q}_G]$ can be obtained by rearranging the columns of identity matrices.

Demand

Given typical dimensions $N = 128, \mathrm{L} = 12$ and $N = 256, \mathrm{L} = 24$ and the constraint $M_g \leq \frac{3}{8}N$, achieve the following objectives:

a) Performance objective: $\frac{tr(\boldsymbol{U}_0^H \boldsymbol{U}_1 \boldsymbol{U}_1^H \boldsymbol{U}_0)}{L} > 95\%$

b) Complexity objective: The complexity of solving $\boldsymbol{P}_g, \boldsymbol{Q}_g$ and computing $\boldsymbol{W}_1$ must be reduced by 50% compared to that of block Cholesky decomposition[1] and matrix multiplication used to compute $\boldsymbol{W}_0$.

- **References**

[1] Chen, Jianping, et al. "Block Algorithm and Its Implementation for Cholesky Factorization." (2013).

12 Low-Complexity Parallel Computing Based on Function Interpolation

Background

RF transceivers in wireless communications systems, such as power amplifiers (PAs), are subject to non-linearity, which affects signal quality and spectrum. Mathematical modeling is used to inverse and identify nonlinear systems, and generate nonlinear components opposite to analog circuits. In this way, the nonlinear effect can be canceled, effectively reducing the nonlinear distortion generated by analog circuits. A high-precision, low-complexity mathematical correction model can help PAs improve efficiency by over 5% and reduce the power consumption of digital chips. For 5G massive MIMO, the integration is expected to be improved by 50%.

Current Result

Nonlinear systems are identified as follows:

Assume that $\{x_{2n}\}_{\mathbb{Z}_{\geq 0}}$ is an independently and identically distributed (i.i.d) sample of $X \sim N(0,\ \sigma^2)$. To meet system requirements, $\{x_{2n}\}_{\mathbb{Z}_{\geq 0}}$ performs linear interpolation to obtain $\{x_{2n+1}\}_{\mathbb{Z}_{\geq 0}}$, and both of them form a high-speed signal $\{x_n\}_{\mathbb{Z}_{\geq 0}}$. The equation is as follows:

$$x_{2n+1} = \sum_{l=-L}^{L} s_l x_{2(n-l)}$$

Computing $\{x_n\}_{\mathbb{Z}_{\geq 0}}$ using the nonlinear function f yields $\{y_n\}_{\mathbb{Z}_{\geq 0}}$:

$$y_n = f(x_{n-M}, \cdots, x_{n-1}, x_n, x_{n+1} \cdots, x_{n+M})$$

f is the nonlinear function of the system. Using a memory polynomial model as an example, $a_{q,m}$ is a memory polynomial coefficient, and the equation is expressed as follows:

$$y_n = \sum_{m=-M}^{M} \sum_{q=1}^{Q} a_{q,m} x_{n-m} |x_{n-m}|^{q-1}$$

Typical configurations: $Q = 7, M = 10$. For other patterns of common nonlinear models, see the appendix.

In practice, the system needs to compute and output S instances of y_n within a unit time. However, due to the hardware rate limit, a single function can only generate T instances of y_n ($T < S$ and $T|S$). The common solution converts high-speed serial computing into low-speed parallel computing—"trade space for time." The parallel computing process of $S/T \in \mathbb{Z}^+$ is as follows: $P = S/T$ instances of $\{f_k = f,\ \forall\ k \in \{0, \dots, P-1\}\}$ models are used for parallel computing within a unit time to obtain $y_{Pn+k} = f_k(x_{Pn+k-M}, \cdots, x_{Pn+k-1}, x_{Pn+k}, x_{Pn+k+1} \cdots, x_{Pn+k+M})$.

Then, it is equivalent to computing $T * \frac{S}{T} = S$ instances of y_n within a unit time. However, P times of resources (corresponding to the number of floating-point computations in all models) are required to implement S/T models in parallel.

Challenge Description

Taking 2x parallelism as an example, we plan to use the $\{y_{2n}\}$ information and function interpolation theory to predict (or partially predict) $\{y_{2n+1}\}$ with fewer resources than those required for computing $\{y_{2n+1}\}$ but higher accuracy.

Demand

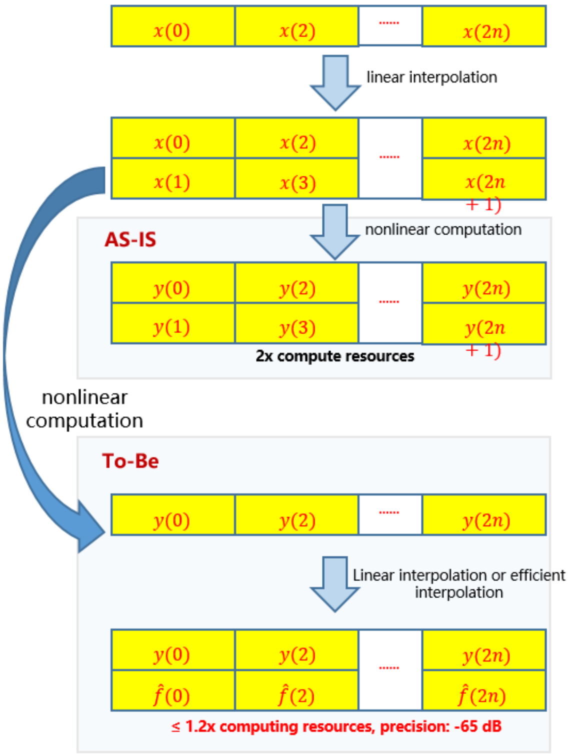
Assume that f is the model, $\{x_n\}$ is the input signal, and $y_n = f(x_n)$, $\mathcal{T}(.) \in \mathbb{R}^+$ is the function or network resource statistics function (usually measured by the number of model multiplication times).

Determine the interpolation function:

$$\hat{f}: \{(y_{2(n-L)}, \ \dots, \ y_{2(n+L)})\}_{\mathbb{Z}_{\geq L}} \to \mathbb{C},\ s.t.\ L \in \mathbb{Z}_{\geq 0},\ 0 < \frac{\mathcal{T}(\hat{f})}{\mathcal{T}(f)} \leq 0.2, \text{and } 10\log_{10}\left(\frac{\|\hat{f} - B\|}{\|U\|}\right)^2 \leq -65$$

$$\text{where, } B(y_{2(n-L)}, \ \dots, \ y_{2(n+L)}) = y_{2n+1}: \{(y_{2(n-L)}, \ \dots, \ y_{2(n+L)})\}_{\mathbb{Z}_{\geq L}} \to \{y_{2n+1}\}_{\mathbb{Z}_{\geq 0}},$$

$$U(2n+1) = y_{2n+1}: 2\mathbb{Z}_{\geq 0} + 1 \to \mathbb{C}, \ \|.\| \text{ represents the 2-norm of the function.}$$



Appendix: Common mathematical expressions of nonlinear mappings f

(1) GMP [1] with typical configurations: $Q = 6$, $M = 10$, $L1 = L2 = 2$

$$f_{GMP}(x_n) = \sum_{m=-M}^{M}\sum_{q=1}^{Q} a_{q,m}x_{n-m}|x_{n-m}|^{q-1}$$

$$+\sum_{m=-M}^{M}\sum_{q=2}^{Q}\sum_{l=1}^{L_1} b_{q,m,l}x_{n-m}|x_{n-m-l}|^{q-1} + \sum_{m=-M}^{M}\sum_{q=2}^{Q}\sum_{l=1}^{L_2} c_{q,m,l}x_{n-m}|x_{n-m+l}|^{q-1}$$

(2) DVR [2] with typical configurations: $K = 8$, $M = 10$

$$f_{DVR}(x_n) = \sum_{m=-M}^{M}\sum_{k=1}^{K} a_{k,m}\big||x_{n-m}| - \beta_k\big|e^{j\theta_{n-m}}$$

$$+\sum_{m=-M}^{M}\sum_{k=1}^{K} b_{k,m}\big||x_{n-m}| - \beta_k\big||x_n|e^{j\theta_{n-m}} \text{ where } e^{j\theta_{n-m}} = \frac{x_{n-m}}{|x_{n-m}|},\ \beta_k = \frac{k}{K}$$

(3) Multistage Cascaded Model [3] with typical configurations: $P = 4$, $K = 4$, $M = 10$, $L1 = L2 = 2$

$$f_{MCM}(x_n) = f_{GMP}(f_{DVR}(x_n))$$

(4) RVTDNN [4] with typical configurations: $M = 10$, $K = 20$

$$u_n^{(k)} = \tanh\left(\sum_{m=-M}^{M} a_{m,k}Real[x_{n-m}] + b_{m,k}Imag[x_{n-m}]\right)$$

$$v_n^{(p)} = \sum_{k=1}^{K} c_{k,p}u_n^{(k)}$$

$$f_{RVTDNN}(x_n) = v_n^{(0)} + iv_n^{(0)}$$

- **References:**

[1] D. R. Morgan, Z. Ma, J. Kim, M. G. Zierdt, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of RF power amplifiers," *IEEE Transactions on Signal Processing*, vol. 54, no. 10, pp. 3852–3860, 2006.

[2] A. Zhu, "Decomposed vector rotation-based behavioral modeling for digital predistortion of RF power amplifiers," *IEEE Trans. Microw. Theory Techn.*, vol. 63, no. 2, pp. 737–744, 2015.

[3] R. Criado, W. Li, W. Thompson, G. Montoro, K. Chuang, and P. L. Gilabert, "Model-Order Reduction of Multistage Cascaded Models for Digital Predistortion," *IEEE Journal of Microwaves*, vol. 5, no. 1, pp. 137–149, Jan. 2025.

[4] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 3, pp. 1025–1033, Mar. 2004.

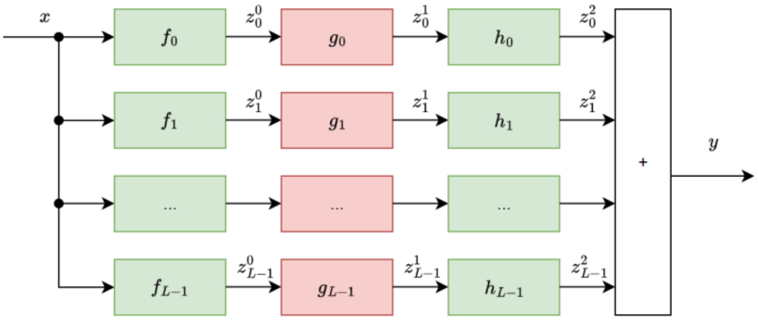
13 Advanced Backpropagation and Training Algorithms for Approximation Errors in Multi-Layer Models

Background

To reduce energy consumption of base stations, the efficiency of RF power amplifiers (PAs), key components in signal transmission, has been steadily improving. However, this improvement also leads to increased nonlinear characteristics, necessitating increasingly complex and deeper digital predistortion models for effective correction. While these models offer enhanced correction capabilities, they also incur significantly higher computational costs in both training and factorization.

Current Result

Let x be the complex signal in the digital domain, y the output complex signal, and $y = \sum_i h_i(g_i(f_i(x)))$ the complex cascading model. The mathematical expressions of different modules in the model for the input x are as follows, where $c_i^k(n)$ denotes the model iteration parameter:



Functions f and h represent linear functions, and g represents a non-linear function. Equations are as follows:

$$f_i: z_i^0(n) \quad = \sum_m c_i^0(m) x_i(n-m)$$

$$h_i: z_i^2(n) \quad = \sum_k c_i^2(k) z_i^1(n-k)$$

$$g_i: z_i^1(n) \quad = \sum_p \sum_j \sum_q c_i^1(n) z_i^0(n-q) |z_i^0(n-j)|^{p-1}$$

The meanings of the parameters are as follows:

- i: ith module at the current layer.

- m, k, q, j: memory depth of the module.
- p: polynomial order.
- n: discrete duration.

Challenge Description

Stochastic gradient descent (SGD) is employed for training, consisting of two steps: error backpropagation and coefficient derivation.

Objective function:

$$argmin_c \ J$$

where,

$$J = \sum_n e_n^* e_n = \sum_n (\, d_n - y_n)^* (d_n - y_n)$$

where, d indicates the desired response, and y the model output.

Error backpropagation of each layer:

$$\begin{aligned} E^2 &= \frac{\partial \sum_n e_n^* e_n}{\partial (z_i^2)^*} & &= \frac{\partial \sum_n e_n^* e_n}{\partial (z_i^2)^*} \\ E^1 &= \frac{\partial \sum_n e_n^* e_n}{\partial (z_i^1)^*} & &= E^2 \cdot \frac{\partial (z_i^2)^*}{\partial (z_i^1)^*} \\ E_0 &= \frac{\partial \sum_n e_n^* e_n}{\partial (z_i^0)^*} & &= E^1 \cdot \frac{\partial (z_i^1)^*}{\partial (z_i^0)^*} \end{aligned}$$

Coefficient derivation of each layer:

$$\begin{aligned} \varDelta_{c_i^2} &= \frac{\partial \sum_n e_n^* e_n}{\partial (c_i^2)^*} = E^2 \cdot \frac{\partial (z_i^2)^*}{\partial (c_i^2)^*} \\ \varDelta_{c_i^1} &= \frac{\partial \sum_n e_n^* e_n}{\partial (c_i^1)^*} = E^1 \cdot \frac{\partial (z_i^1)^*}{\partial (c_i^1)^*} \\ \varDelta_{c_i^0} &= \frac{\partial \sum_n e_n^* e_n}{\partial (c_i^0)^*} = E^0 \cdot \frac{\partial (z_i^0)^*}{\partial (c_i^0)^*} \end{aligned}$$

Coefficient update:

$$\begin{aligned} c_i^2(n) &= c_i^2(n-1) - \mu \varDelta_{c_i^2} \\ c_i^1(n) &= c_i^1(n-1) - \mu \varDelta_{c_i^1} \\ c_i^0(n) &= c_i^0(n-1) - \mu \varDelta_{c_i^0} \end{aligned}$$

Demand

When both the forward models and training process operate online with constraints of less than 0.3 dB performance loss and equivalent convergence speed, the algorithms must achieve over a 30% reduction in backpropagation and training costs on the given models.

- **References**

[1] Jaderberg, Max, et al. "Decoupled neural interfaces using synthetic gradients." International conference on machine learning. PMLR, 2017.

[2] Adamson, Reece. "The Forward-Forward Algorithm: Characterizing Training Behavior." arXiv preprint arXiv:2504.11229 (2025).

[3] Li, Qinyu, Yee Whye Teh, and Razvan Pascanu. "NoProp: Training Neural Networks without Back-propagation or Forward-propagation." arXiv preprint arXiv:2503.24322 (2025).

[4] Hinton, Geoffrey. "The forward-forward algorithm: Some preliminary investigations." arXiv preprint arXiv:2212.13345 2.3 (2022)

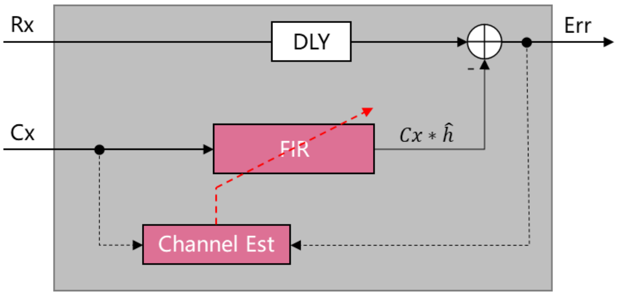
[5] Criado R, Li W, Thompson W, et al. On the parameter identification of cascaded behavioral models for wideband digital predistortion linearization[C]//2025 IEEE/MTT-S International Microwave Symposium-IMS 2025. IEEE, 2025: 657-660.

14 Adaptive Echo Cancellation Under Non-Stationary Conditions

Background

In full-duplex systems operating on the same frequency for transmission (Tx) and reception (Rx), the Rx channel suffers from self-interference of Tx signals, resulting in degraded Rx sensitivity. Traditional approaches employ adaptive filtering architectures to dynamically track external channel variations and suppress Tx-induced inference in the Rx channel.

Figure 1 Interference cancellation using the adaptive filtering architecture



Current Result

Numerous improvements have been proposed for conventional least mean square–based (LMS-based) algorithms. (1) Transforming the time-domain coefficient solution to the frequency domain and assigning different step sizes to each frequency address the issue of large dynamic signal fluctuations in the frequency domain [1]. (2) Introducing subband adaptive filtering reduces the input signal correlation and accelerates convergence [2]. (3) Introducing the proportionate normalized least-mean-squares (PNLMS) algorithm improves the convergence performance in sparse scenarios [3]. However, these techniques still struggle to meet the requirements for fast convergence, exhibit limited capabilities in tracking dynamic channels, and lack stability against interference.

Recursive least square–based (RLS-based) algorithms involve high computational complexity and are difficult to implement on chips.

Challenge Description

Given:

- Tx signal $\boldsymbol{x}_{ref}$, which is a one-dimensional vector and a random signal that obeys the complex Gaussian distribution.
- Rx signal $\boldsymbol{x}_{rx} = \boldsymbol{x}_{si} + \boldsymbol{x}_{ue} + \boldsymbol{n}$, which is a one-dimensional vector, where

  a) $\boldsymbol{x}_{si}$ denotes the self-interference signal with $\boldsymbol{x}_{si} = \boldsymbol{x}_{ref} * \boldsymbol{h}(t)$, and $\boldsymbol{h}(t)$ denotes the actual channel from Tx to Rx, which changes rapidly and non-stationarily due to the environment change (unknown and subject to real-time estimation).

b) $\boldsymbol{x}_{ue}$ denotes the random signal that complies with complex Gaussian distribution.

c) $\boldsymbol{n}$ denotes the complex white Gaussian noise.

Optimization function: Estimate the external channel $\widehat{\boldsymbol{h}}(t)$ based on the Tx signal $\boldsymbol{x}_{ref}$ and Rx signal $\boldsymbol{x}_{rx}$ to eliminate the self-interference signal. The equation is expressed as follows:

$$\min_{\widehat{\boldsymbol{h}}(\boldsymbol{t})}\|\boldsymbol{x}_{err}\|^2 = \left\|\boldsymbol{x}_{rx} - \boldsymbol{x}_{ref} * \widehat{\boldsymbol{h}}(\boldsymbol{t})\right\|^2$$

Demand

1、 Cancel the echo based on the given data, with a cancellation capability greater than 60 dB.

$$10\log_{10}(\frac{\|\boldsymbol{x}_{si}\|^2}{\left\|\boldsymbol{x}_{si} - \boldsymbol{x}_{ref} * \widehat{\boldsymbol{h}}(\boldsymbol{t})\right\|^2}) > 60$$

2、 Maintain the computational complexity at $O(N)$, where N denotes the finite impulse response (FIR) order.

- **References**

[1] Soo J S, Pang K K. Multidelay block frequency domain adaptive filter[J]. IEEE Transactions on Acoustics, Speech, and Signal Processing, 1990, 38(2): 373-376.

[2] Lee K A, Gan W S. Improving convergence of the NLMS algorithm using constrained subband updates[J]. IEEE signal processing letters, 2004, 11(9): 736-739.

[3] Duttweiler D L. Proportionate normalized least-mean-squares adaptation in echo cancelers[J]. IEEE Transactions on speech and audio processing, 2002, 8(5): 508-518.

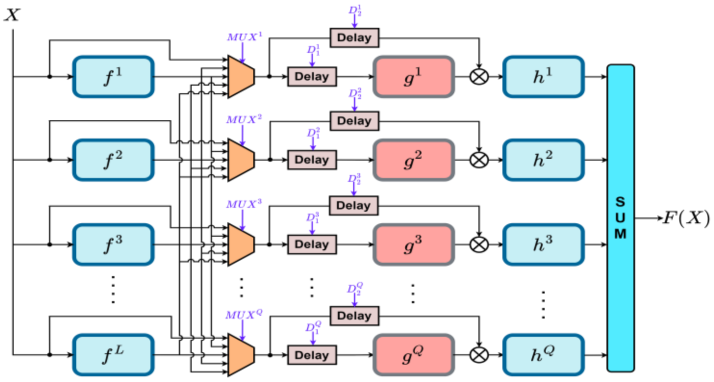
15 Efficient Computing - Low-Power Chip Algorithms: Low-Bit-Width Forward Network

Background

As communication algorithms are required to support increasingly higher bandwidth and data rate, the power consumption and chip area required for algorithm implementation also grow significantly. Bit width optimization is crucial for reducing the hardware overhead. This challenge focuses on typical bit-width design algorithms, aimed to reduce the hardware overhead.

Current Result

The following figure shows a commonly used cascaded structure for modeling complex nonlinearities, which typically offers stronger modeling capabilities compared to non-cascaded structures. The bit width in such models is typically set between 12 bits to 16 bits, which directly impacts the chip resources.



Challenge Description

Optimization objective:

$$\min_{B_X, B_\alpha, B_c, B_\beta} \quad Cost(F, X)$$

$$\text{st. } \|F(X) - F_\infty(X_\infty)\| \le e,$$

where:

$X = [X(1), X(2), \ldots] \in \mathbb{C}^N$ denotes the input signal sequence, N the input signal length, X_∞ the full-precision floating-point version of X, $e \in \mathbb{R}^+$ the upper bound of the expected error, and $Cost(\cdot)$ the cost function.

Forward function $F(\cdot)$ is composed of the following sub-functions according to the topology shown in the figure, and $F_\infty(\cdot)$ indicates the full-precision floating-point version of $F(\cdot)$:

$$f^i : y^i(n) = \sum_{m=0}^{M} \alpha_m^i x^i(n-m)\,, i = 1,2,\ldots,L$$

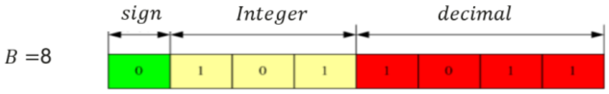
$$D_j^i : y^i(n) = x^i(n - D_j^i),\ i = 1,2,\ldots,Q, j = 1,2$$

$$g^i : y^i(n) = \sum_{p=0}^{P} c_p^i |x^i(n)|^{2p}\,, i = 1,2,\ldots,Q$$

$$h^i : y^i(n) = \textstyle\sum_{m=0}^{M} \beta_m^i x^i(n-m)\,, i = 1,2,\ldots,Q,$$

where $D_j^i, p \in \mathbb{Z}^+$, $\alpha_m^i, c_p^i, \beta_m^i \in \mathbb{C}$, and $B_X, B_\alpha, B_c, B_\beta$ represent the bit widths of the signal X and the sub-function coefficients α_m^i, c_p^i, and β_m^i, respectively.

In the following example,



Demand

1. Develop a bit width search and analysis method for the forward network, achieving a 40% optimization in bit width compared to the baseline.
2. Minimize the signal and coefficient bit width of each node in the above network, and ensure that the error between the output signal $F(X)$ after bit width compression and the output signal $F_\infty(X_\infty)$ of arbitrary-precision floating point is less than the preset value e (e < 0.5 dB).

- **References**

[1] Dinis D C, Cordeiro R F, Barradas F M, et al. Agile single-and dual-band all-digital transmitter based on a precompensated tunable delta–sigma modulator. IEEE Transactions on Microwave Theory and Techniques, 2016, 64(12): 4720-4730.

[2] Gholami A, Kim S, Dong Z, et al. A survey of quantization methods for efficient neural network inference. Low-Power Computer Vision. Chapman and Hall, CRC, 2022: 291-326.

[3] Nagel M, Fournarakis M, Amjad R A, et al. A white paper on neural network quantization. arXiv preprint arXiv:2106.08295, 2021

[4] X. Liu, W. Chen and Z. Feng, "Broadband Digital Predistortion Utilizing Parallel Quasi- Wiener-Hammerstein Model with Extended Dynamic Range," 2021 IEEE MTT-S International Wireless Symposium (IWS), Nanjing, China, 2021, pp. 1-3.

16 Joint Optimization of DAG Scheduling and Memory Allocation

Background

The directed acyclic graph (DAG) for a wireless baseband algorithm comprises hundreds of operators and tensors. To minimize the latency of the entire DAG, each operator must be scheduled using appropriate custom tensor or vector instructions. With each operator mappable to over ten instruction variants—yielding more than 10^100 possible schedules for the entire DAG, it is crucial to efficiently pinpoint the optimal solution. Additionally, since the DAG represents data dependencies between tensors, we must account for tensor reuse to ensure that the DAG's total memory footprint remains within hardware limits. Given that the tensor size is defined by a dynamic shape across various scenarios, it remains challenging in seeking an appropriate memory allocation approach to guarantee that the DAG's total memory footprint never exceeds the hardware limit.

Current Result

1. Operator scheduling:

   While the software tool can automatically choose the scheduling solution for a single operator, there are constraints among operators, making it infeasible to achieve optimal scheduling for every operator on the hardware. Expert experience remains crucial for optimizing operator scheduling at the DAG level.

2. Memory allocation:

   The software tool can automatically produce a memory allocation solution based on data dependencies in the DAG and evaluate whether the memory allocation solution is beyond the hardware limit through symbolic execution. However, the tool cannot directly generate a memory allocation solution within the memory upper bound.

Challenge Description

Given the following inputs:

Conditions:

1. GAG *G=(V, E*), where *V* denotes the set of operator nodes, and *E* denotes the set of tensor edges.
2. Custom tensor or vector instruction set *I*=$\{i_1,\ i_2,...,i_m\}$, and latency cost *L*(v, i) for each instruction.
3. Input parameters $x = (x_1, x_2, \ldots, x_n)$, where the domain of x can be presented as

$D = \{ x \in Z^n \mid l_i \leq x_i \leq u_i, \forall i = 1, \ldots, n \}$, subject to constraints between two parameters x_i, x_j.

4. Dynamic tensor shape $m(e) = f_e(x), e \in E$.

   Constraint:

   Hardware memory limit M_{max}.

   Objective function:

$$\min_{f,r} \sum_{v \in V} L\big(v,\ f(v)\big)\ with \sum_{j=1}^{k} \max_{e \in R_j} f_e(e) \leq M_{max}$$

   Solve the following mapping relationships:

   1) Operator scheduling: f: $V \to I$;

   2) Memory allocation: g: $E \to \{1,\ldots, k\}$,$G_j = \{ e \in E \mid g(e) = j\}$;

Demand

1. Design an evaluation algorithm like dynamic programming or integer programming to evaluate the theoretically optimal upper bound of memory allocation. Ensure that the evaluated value is within 10% of the theoretical value.
2. Based on the evaluation algorithm, generate a memory allocation solution that stays below the hardware limit of less than 150 KB.
3. Based on the given inputs and domains, solve the optimal operator scheduling solution within a set time frame.

- **References**

[1] Zhang, X., Chen, Y., & Wang, M. (2021). IOS: Inter-Operator Scheduler for CNN Acceleration. In Proceedings of the MLSys Conference

[2] Liu, J., Zhao, T., & Li, K. (2020). Operator Fusion and Scheduling for Efficient Deep Neural Network Inference on Specialized Hardware. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 39(7), 1234–1245.

[3] Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Kennedy, K., & Reddy, U. S. P. (1981). Register allocation via graph coloring. Computer Languages, 6(1), 47–57.

[4] Cooper, K., & Torczon, L. (2004). Engineering a register allocator. In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '04) (pp. 113–124)

[5] Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., ... & Chen, Y. (2018). TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In OSDI '18: Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (pp. 578–594).

[6] Chen, H., Li, Y., & Zhang, D. (2021). Efficient Operator Scheduling via Full-Stack Compiler Techniques for DL Accelerators. In Proceedings of PLDI (pp. 234–245).

[7] Wang, M., Li, S., & Zhou, Q. (2022). Domain-Specific Operator Scheduling in Deep Learning: A Survey. IEEE Micro, 42(3), 45–57.

17 [Optimization problem] Route planning

Background

Carrier aggregation (CA) is a vital technology for enhancing user experience. To configure the optimal carrier combination for each user equipment (UE), base stations must optimize carrier combinations for UEs based on site-level settings, cell data, and UE-reported capability profiles. As base station specifications—such as frequencies, cells, neighboring cells, and CA configurations—continue to evolve, the complexity of capability profiles reported by UEs also increases. This has led to rising computational costs for combination optimization. Mathematically, optimizing the carrier combination for a single UE is equivalent to solving a knapsack problem on a conflict hypergraph, which is known to be NP-hard. However, many UEs report similar or identical capability combinations, suggesting a sizable space for optimization. Can we leverage multi-UE capability data to establish a correlation between UE capability profiles and optimal carrier combinations, thereby reducing computational costs in multi-UE CA optimization?

The multi-UE CA optimization challenge can be abstracted as a path planning problem, with only minor simplifications to the original constraints.

Challenge Description

Country A boasts a thriving tourism industry, attracting tens of thousands of tourists annually.

Country A has several provinces, each comprising several cities. Every city is rated publicly based on four categories: history, geography, food, and culture.

Each tourist has multiple travel plans to choose from. The objective is to identify and select the plan with the highest total score for each tourist.

a) **Country A's city information overview**

cityNum: total number of cities in Country A.

Each city is described using the following attributes.

City 1	Example
city name	Nanjing
province	Jiangsu
historical score	90
geographical score	95
food score	100
cultural score	100

The corresponding inputs are described as follows:

city_num: number of cities

city_name[city_num]: name of each city

city_province[city_num]: province of each city

city_history_score[city_num]: historical score of each city

city_geography_score[city_num]: geographical score of each city

city_culture_score[city_num]: cultural score of each city

city_food_score[city_num]: food score of each city

b) Tourist plan information

planNum[i]: number of travel plans for the ith tourist.

For the jth travel plan of the ith tourist, the following information is specified.

Plan A		Example
province number		2
Jiangsu	*historical score set*	{70, 80, 90, 100}
	geographical score set	{85, 95}
	food score set	{95, 100}
	cultural score set	{60, 70}
Anhui	*historical score set*	{70, 80, 90, 100}
	geographical score set	{85, 95}
	food score set	{95, 100}
	cultural score set	{60, 70}

Input for each tourist:

```
plan_num: number of candidate travel plans
For i from 0 to plan_num-1
    plan[i].province_num: number of provinces included in the ith plan
    for j from 0 to province_num - 1
        plan[i].province[j].province: name of the jth province in the ith plan
        plan[i].province[j].historyScoreSet: set of historical scores of the jth province in the ith plan
        plan[i].province[j].geographScoreSet: set of geographical scores of the jth province in the ith plan
        plan[i].province[j].foodScoreSet: set of food scores of the jth province in the ith plan
plan[i].province[j].cultureScoreSet: set of cultural scores of the jth province in the ith plan
```

c) Output

Output for each tourist:

```
target_plan_index: selected plan
for i from 0 to plan[target_plan_index].province_num -1
    plan[target_plan_index].province[i].city_index: selected cities in each province of the selected plan
```

d) Constraints:

For each tourist, the output must meet the following conditions:

- 0 ≤ target_plan_index < plan_num;   # Used to determine the validity of **target_plan_index**
- 0 ≤ city_index < city_num   # Used to determine the validity of **city_index**
- For each prov_index from 0 to plan[target_plan_index].province_num -1
    - city_index = plan[target_plan_index].province[prov_index].city_index
    - city_province[city_index] = plan[target_plan_index].province[prov].province # The target city is within the province.
    - city_history_score[city_index] ∈ plan[target_plan_index].province[prov]. historyScoreSet # The

historical score of the target city is within the range.

- city_geograph_score[city_index] ∈ plan[target_plan_index].province[prov]. geographScoreSet # The geographical score of the target city is within the range.
- city_food_score[city_index] ∈ plan[target_plan_index].province[prov]. foodScoreSet # The food score of the target city is within the range.
- city_culture_score[city_index] ∈ plan[target_plan_index].province[prov]. cultureScoreSet # The cultural score of the target city is within the range.

e) Scoring function

The scoring function for each tourist is defined as follows:

$$planScore(plan\ A) = \sum_{for\ each\ province\ p\ in\ plan\ A} provinceScore(A, p)$$

$provinceScore(plan\ A,\ province\ p) = MAX(score_{city\ 1,A,p}, score_{city\ 2,A,p}, \ldots, score_{city\ k,A,p}),\ k\ is\ number\ of\ cities\ in\ province\ p.$

$$score_{city\ i,A,p} = \begin{cases} SCR_{city\ i,A,p} \begin{cases} if\ city_i's\ province = p \\ if\ city_i's\ historical\ score \in A's\ p's\ historical\ score\ set \\ if\ city_i's\ food\ score \in A's\ p's\ food\ score\ set \\ if\ city_i's\ cultural\ score \in A's\ p's\ cultural\ score\ set \end{cases} \\ 0, \quad otherwise \end{cases}$$

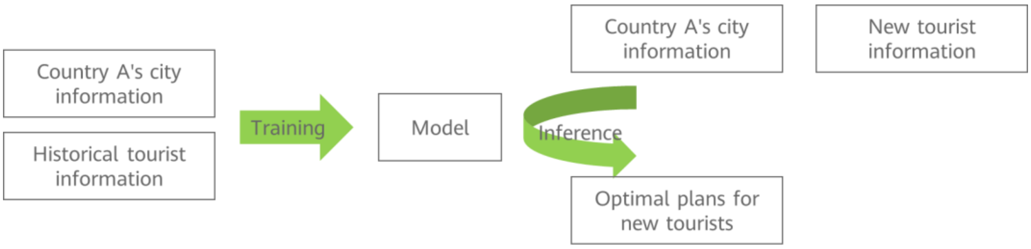
$$SCR_{city\ i,A,p} = city_i's\ historical\ score * city_i's\ geographical\ score * city_i's\ food\ score * city_i's\ cultural\ score$$

Demand

Traditionally, addressing this challenge involves evaluating all possible combinations to identify the best option. However, this method involves a sheer volume of scenarios, leading to high time complexity.

To accelerate the search for the optimal solution, explore advanced optimization approaches—including, but not limited to, AI. One approach is to train an AI model using historical travel plans favored by tourists. Once trained, the model can quickly generate personalized plans for new tourists based on learned preferences.

Compared to conventional algorithms, this new approach reduces computational costs by approximately 75% when identifying optimal travel plans for new tourists.


Country A's city information
Historical tourist information
Training
Model
Country A's city information
New tourist information
Inference
Optimal plans for new tourists

Acknowledgements

Thanks following experts, scholars and staff of Sino-Russian mathematics center who provided guidance and support during the preparation of this guide:

Peking University: Zhang Jiping, Hu Jun, Fan Shaofeng, Xu Ting, Yu Xinhui, Wang Yakun, Zhang Jiaona

Central China Normal University: Liu Hongwei, Luo Jinquan, Wang Guodong

Lanzhou University: Wang Jizeng, Yang Bing

Hunan University: Bai Minru, Zheng Libing

Shandong University: Feng Xinwei

Jilin University: Jia Jiwei

Sichuan University: Song Enbin, Chen Gang

Nanjing University: Liu Keqin

University of Science and Technology of China: Zhang Juyong, Chen Renjie, Xiao Dong

East China Normal University: Ye Qingjie

Great Bay University: Yang Sikun

Shandong National Center for Applied Mathematics: Guo Xin, Yan Xiaodong, Lin Yiwei, Han Haohui

Wuhan University: Xiang Hua, Jiang Wei

University of Toronto: Wang Linbo

Moscow State University: Andrey Shafarevich, Alexander Ivanov, Alexander Zheglov, Vladimir Bogachev, Andrei Krylov;

National Research University Higher School of Economics: Ivan Arzhantsev, Sergei Kuznetsov, Vasilii Gromov,Andrey Delitsyn;

Sobolev Institute of Mathematics: Dmitry Tkachev, Adil Yerzin, Yuri Kochetov;

AI Center: Evgeny Burnaev.