**Sparse Matrix Multiplication Accelerator Architec**

1. **Fully-Connected (FC) Layer**

Tầng fully-connected trong mạng nơ-ron (mọi nơ-ron đều kết nối với nhau) có thể được biểu diễn dưới dạng phép nhân ma trận. Ban đầu, dữ liệu đầu vào (input) và bộ lọc (filters) có nhiều chiều (ví dụ: kênh, chiều cao, chiều rộng). Nhưng bằng cách "làm phẳng" (flatten) chúng, ta biến nó thành một phép nhân ma trận đơn giản: Z=A×B, trong đó là đầu vào, B là trọng số, và Z là đầu ra.

A diagram of different colored squares

AI-generated content may be incorrect.

1. **Einsum**

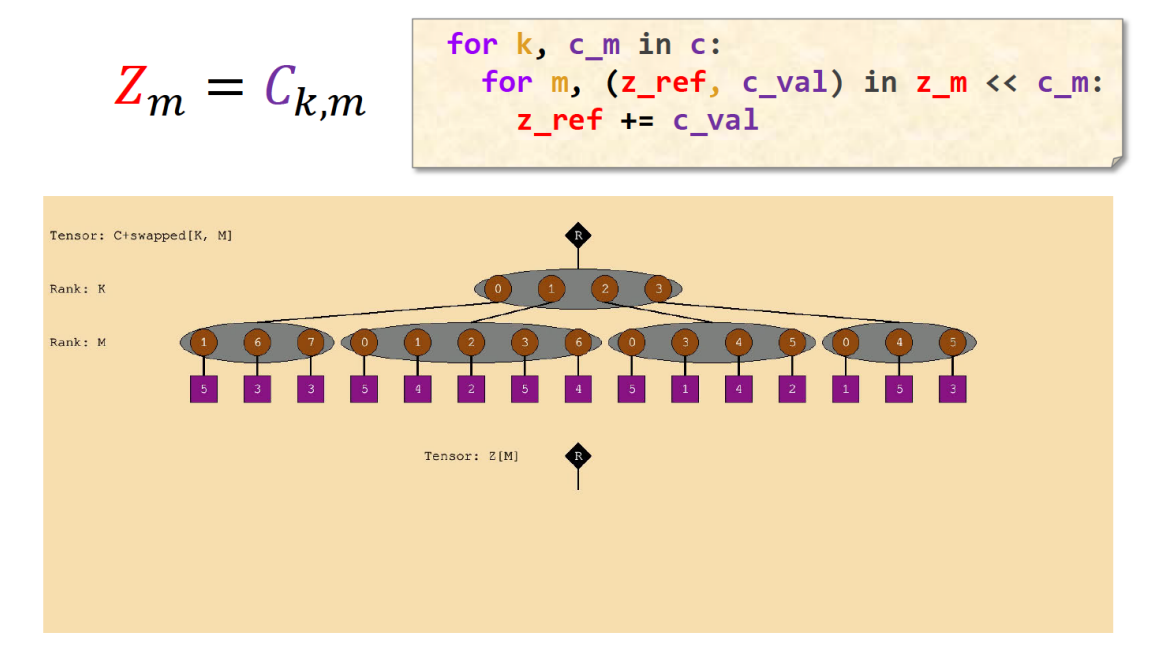
Einsum là một cách viết gọn gàng để mô tả phép nhân ma trận hoặc tensor. Ví dụ: Zm,n=Am,k×Bn,k có nghĩa là nhân hai ma trận A và B để ra Z. Nó giúp ta hiểu rõ cách các chiều (indices) tương tác với nhau.

* **Shared indices -> Intersection (&)**: Các chiều giống nhau giữa A và B (như k) là nơi hai ma trận "gặp nhau". Giao nhau này giống như tìm những phần tử chung để nhân với nhau.

A diagram of a fiber intersection

AI-generated content may be incorrect.

* **Contracted indices -> Reduction (+=)**: Chiều chung k sẽ được "thu gọn" bằng cách cộng tất cả các kết quả nhân lại với nhau. Ví dụ, tổng trên k k để từ Am,k×Bn,k ra Zm,n
* **Uncontracted indices -> Populate output point (<<)**: Các chiều không bị thu gọn (như m và n) quyết định vị trí của kết quả trong ma trận đầu ra Z. Chúng "điền" giá trị vào Zm,n.



* **Index arithmetic -> Projection**: Trong trường hợp phức tạp hơn (như phép cuốn tích), các chỉ số có thể cộng trừ (ví dụ: p+r,q+s) để "chiếu" dữ liệu từ vị trí này sang vị trí khác, giúp ánh xạ đầu vào sang đầu ra.

Nói đơn giản, phần này biến một tầng FC thành phép nhân ma trận, dùng Einsum để mô tả, và phân tích cách các chiều làm việc cùng nhau: giao nhau để nhân, thu gọn để cộng, điền để tạo đầu ra, và chiếu để xử lý vị trí.

A screenshot of a computer

AI-generated content may be incorrect.A black screen with white text

AI-generated content may be incorrect.

1. **ExTensor - Tổng quan**

ExTensor là một kiến trúc phần cứng được thiết kế để tăng tốc xử lý **đại số tensor thưa** (sparse tensor algebra), dựa trên bài báo tại MICRO 2019. Nó tập trung vào việc tận dụng tính thưa (nhiều phần tử bằng 0) để giảm tính toán và tăng hiệu quả trong học sâu.

1. **Phân cấp lát thưa (Hierarchical Sparse Tiling)**:
   * Chia tensor lớn thành các "lát" (tiles) nhỏ hơn để xử lý từng phần, giúp tiết kiệm bộ nhớ và tối ưu truy cập dữ liệu.
2. **Giao nhau phân cấp (Hierarchical Intersection)**:
   * Tìm các phần tử chung (khác 0) giữa các tensor ở nhiều cấp độ khác nhau, thay vì duyệt toàn bộ, để giảm lãng phí tính toán.
3. **Đơn vị giao nhau tối ưu hóa**:
   * Một bộ phận phần cứng đặc biệt được thiết kế để thực hiện phép giao nhau nhanh và hiệu quả.

**Quan sát từ hoạt hình hai cấp**

* **Duyệt dữ liệu**:
  + Tensor A: Chỉ duyệt qua các lát một lần.
  + Tensor B: Duyệt nhiều lần qua các lát.
  + Trong mỗi tensor, xử lý từng lát trước khi chuyển sang lát khác.
* **Tạo đầu ra**: Các lát đầu ra được tạo tuần tự, nhưng một số lát (như (0,0) (0,0) (0,0)) có thể không xuất hiện do tính thưa.

**Ý nghĩa**

ExTensor giúp xử lý tensor thưa nhanh hơn bằng cách chia nhỏ vấn đề, tập trung vào các phần tử quan trọng (khác 0), và dùng phần cứng chuyên dụng để tối ưu giao nhau. Điều này rất hữu ích cho các ứng dụng học sâu cần tính toán hiệu quả.

1. **OuterSPACE - Tổng quan**

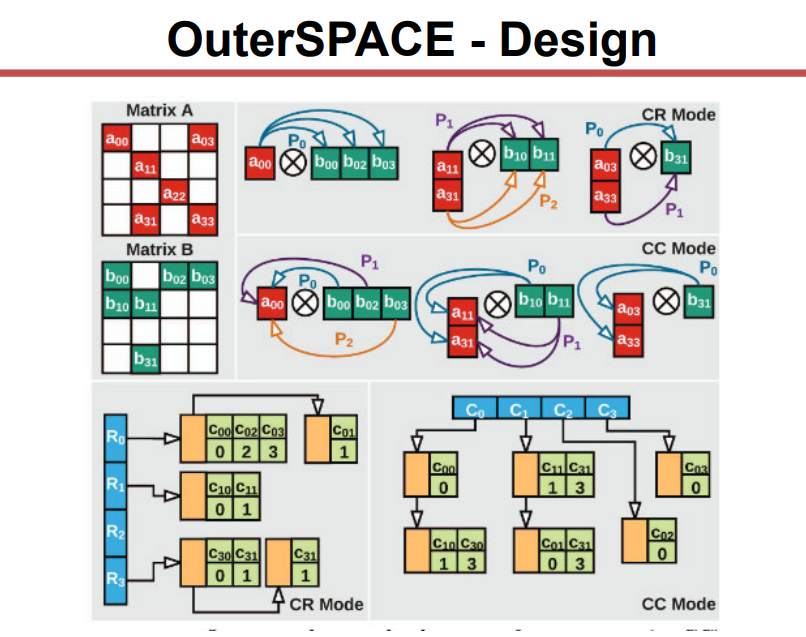
OuterSPACE là một kiến trúc phần cứng được giới thiệu trong bài báo tại HPCA 2018, tập trung vào tăng tốc **phép nhân ma trận thưa** bằng cách sử dụng **tích ngoài** (outer product) thay vì tích trong thông thường. Nó tận dụng tính thưa để giảm tính toán và tối ưu hiệu suất.

1. **Quy trình hai bước**:
   * **Bước 1**: Tạo tensor trung gian Tk,m,n=Ak,m×Bk,n ​ bằng tích ngoài, sinh ra các kết quả từng phần.
   * **Bước 2**: Giảm tensor trung gian Zm,n=Tm,n,k ​ bằng cách cộng dồn trên chiều k để ra kết quả cuối cùng.

A diagram of a function

AI-generated content may be incorrect.

1. **Quan sát** (Observations):
   * **Bước 1**: Duyệt đồng bộ (concordant) A và B, tái sử dụng dữ liệu hiệu quả, tạo từng phần tử của T song song.
   * **Bước 2**: Duyệt đồng bộ T, tạo Z tuần tự, tận dụng truy cập song song.
   * Dùng định dạng đặc biệt để xử lý sự khác biệt về thứ tự duyệt.



1. **Khái niệm chính**:
   * Tạo nhiều lát kết quả từng phần bằng tích ngoài.
   * Giảm các kết quả từng phần để ra ma trận cuối.
   * Tối ưu định dạng dữ liệu cho các thứ tự duyệt khác nhau.

**Ý nghĩa**

OuterSPACE chia nhỏ phép nhân ma trận thưa thành hai giai đoạn, tận dụng tích ngoài để xử lý dữ liệu thưa hiệu quả, giảm chi phí tính toán và tăng khả năng song song. Tuy nhiên, nó cần quản lý cẩn thận thứ tự duyệt để tránh xung đột bộ nhớ.

1. **Traversing Sparse Tensors**
2. **Duyệt Đồng bộ (Concordant Traversal)**:
   * Thứ tự duyệt: K,M (chậm đến nhanh).
   * Dữ liệu được truy cập theo cách tự nhiên, phù hợp với cách lưu trữ (ví dụ: đọc khối bộ nhớ).
   * Hiệu quả cao khi thứ tự duyệt khớp với định dạng tensor.

A screenshot of a computer game

AI-generated content may be incorrect.

1. **Duyệt Không đồng bộ (Discordant Traversal)**:
   * Thứ tự duyệt: K, M.
   * Không khớp với cách lưu trữ, gây khó khăn khi đọc khối bộ nhớ (block memory reads), làm giảm hiệu suất.

A screenshot of a computer game

AI-generated content may be incorrect.

1. **Duyệt trên Fibertree**:
   * Tensor thưa được tổ chức dưới dạng **cây sợi (fibertree)**, giúp duyệt chỉ các phần tử khác 0.
   * Đồng bộ (K,M): Dễ dàng truy cập theo sợi.
   * Không đồng bộ (M,K): Phức tạp hơn, cần điều chỉnh.

A diagram of a diagram

AI-generated content may be incorrect.

1. **Rank Swizzle/Merger**:
   * Khi thứ tự duyệt không khớp, cần "đảo thứ hạng" (swizzle): Lấy phần tử thấp nhất từ sợi đầu vào (ví dụ: sợi M) và đặt vào đầu ra với tọa độ đảo ngược.
   * Giúp điều chỉnh dữ liệu để duy trì hiệu quả.

A diagram of a diagram

AI-generated content may be incorrect.

**Ý nghĩa**

Cách duyệt tensor thưa quyết định hiệu suất xử lý:

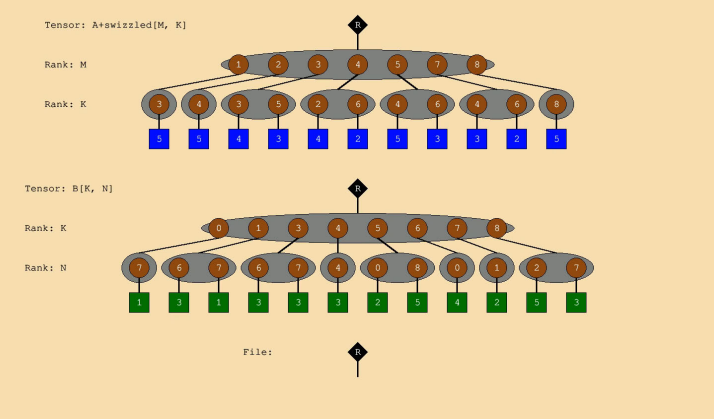
* **Đồng bộ**: Tối ưu, nhanh, tiết kiệm tài nguyên.
* **Không đồng bộ**: Chậm hơn, cần kỹ thuật bổ sung như swizzle để khắc phục.

=> Phần này nhấn mạnh tầm quan trọng của việc chọn thứ tự duyệt phù hợp và dùng cấu trúc như fibertree để xử lý tensor thưa hiệu quả trong các kiến trúc như OuterSPACE.

1. **Gamma:**

**1. Tạo Tensor Trung gian T**

* **Công thức**: Tm,k,n=right(Am,k,Bk,n) (dùng tích ngoài hoặc giao nhau để tạo kết quả từng phần).
* **Thứ tự duyệt**: M,K,N (chậm đến nhanh).
* **Song song hóa**: Xử lý song song trên K (các sợi K), một phần trên M (không mô hình hóa đầy đủ).
* **Quan sát**:
  + **Duyệt đồng bộ A**: Tensor A được duyệt một lần duy nhất, theo thứ tự tự nhiên.
  + **Tái sử dụng B**: Các sợi Bn​ (cột của B) được truy xuất nhiều lần.
  + **Tạo T**: Với mỗi M, xử lý song song trên K tạo ra các sợi Tn​ đồng bộ (concordant).
  + **Rank Swizzle**: T[M,K,N] được tạo sao cho có thể "đảo thứ hạng" thành T′[M,N,K]. Vì xử lý song song trên K và tạo Tn​ đồng bộ, các phần tử đầu tiên cần cho swizzle luôn sẵn sàng.



**2. Rank Swizzled T**

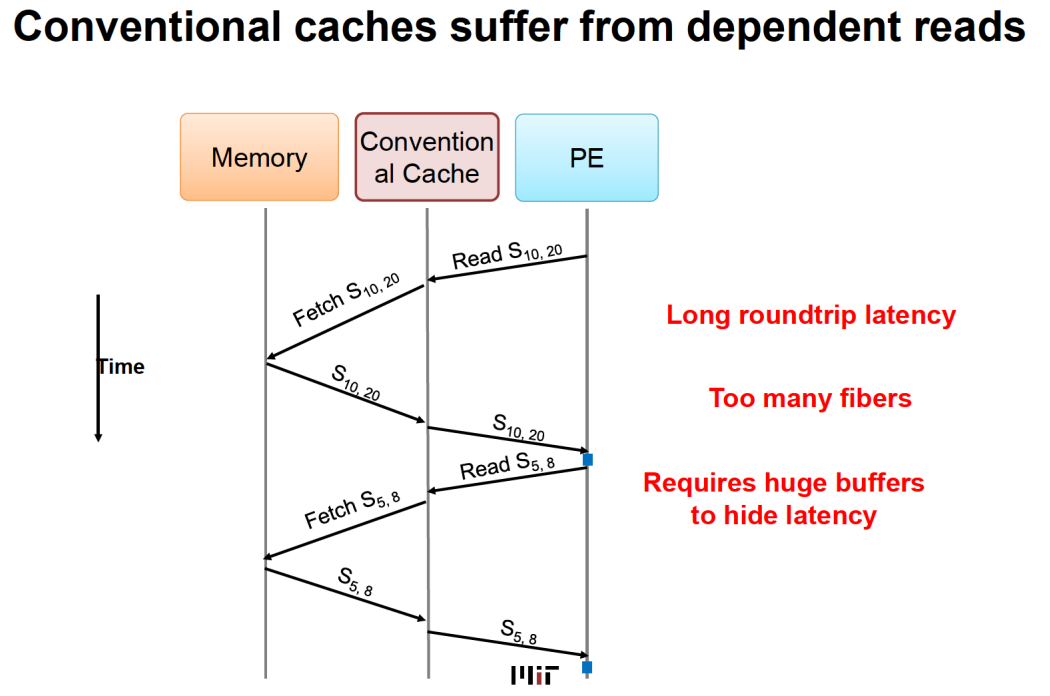
* **Chuyển đổi**: Từ T[M,K,N] sang T′[M,N,K] (thay đổi thứ tự chiều để phù hợp với bước tiếp theo).
* **Lý do**: Đảm bảo duyệt đồng bộ ở bước 2, tránh lãng phí bộ nhớ và tối ưu pipeline.

A diagram of a diagram of a structure

AI-generated content may be incorrect.

**3. FiberCache**

* **Vai trò**: Bộ đệm sợi (FiberCache) tách biệt quá trình đọc dữ liệu và độ trễ bộ nhớ.
* **Hiệu quả**: Giảm thời gian chờ (roundtrip latency) bằng cách dùng bộ đệm nhỏ, đặc biệt khi độ trễ thấp.



**Khái niệm chính của Gamma**

1. **Pipeline với bộ nhớ nhỏ**: Chia nhỏ phép nhân ma trận thành hai bước, xử lý kiểu dòng chảy, giảm nhu cầu lưu trữ trung gian.
2. **Song song hóa và Swizzle**: Dùng song song trên K hoặc M và đơn vị "merge" để đảo thứ hạng, đảm bảo duyệt đồng bộ.
3. **FiberCache tách biệt**: Lưu trữ tạm các sợi B để tái sử dụng, tối ưu truy cập bộ nhớ.
4. **Sắp xếp lại A**: Tái tổ chức A để tăng hiệu quả của FiberCache, giảm truy xuất lặp lại.

**Ý nghĩa**

* Gamma tận dụng thuật toán Gustavson để xử lý ma trận thưa hiệu quả, kết hợp pipeline, song song hóa, và quản lý bộ nhớ thông minh.
* So sánh hiệu suất (trong phần spMspM): Gamma (có tiền xử lý) vượt trội hơn Intel MKL, OuterSPACE, và SpArch trên các ma trận thưa phổ biến.

1. **Mô hình hóa các luồng dữ liệu thưa thớt – TeAAL**

A diagram of a computer

AI-generated content may be incorrect.

**Ý nghĩa**

* TeAAL giống như một "bản đồ" để hiểu và thử nghiệm các cách tổ chức dữ liệu thưa trong phần cứng.
* Nó cho phép người thiết kế "vẽ" trước cách tính toán (Einsum + lịch trình) và suy ra cách thực thi (loop nest), giúp tối ưu hiệu suất và năng lượng.

1. Tóm tắt về các thuộc tính thiết kế quan trọng của các bộ tăng tốc nhân ma trận thưa (**spMspM accelerators**):
2. **Phân lát dữ liệu (Tiling)**: Chia nhỏ dữ liệu để tăng tính cục bộ (locality), giúp truy cập nhanh hơn.
3. **Giao nhau rõ ràng (Explicit Intersection)**: Tính thưa làm phép giao nhau (tìm phần tử khác 0) trở thành bước quan trọng.
4. **Giao nhau phân cấp (Hierarchical Intersection)**: Thực hiện giao nhau ở các mức cao hơn trong cây sợi (fibertree).
5. **Ba luồng dữ liệu chính**: Inner-product (giữ đầu ra cố định), Outer-product (tích ngoài), Gustavson (duyệt tối ưu).
6. **Pipeline nhiều giai đoạn**: Chia nhỏ tính toán thành các bước liên tiếp để xử lý hiệu quả.
7. **Đảo thứ hạng (Rank Swapping)**: Cần thiết để đảm bảo duyệt đồng bộ, dùng đơn vị "merge" để thực hiện.
8. **Tối ưu di chuyển dữ liệu**: Chọn định dạng dữ liệu phù hợp để giảm băng thông.
9. **Bộ đệm tách biệt (Explicit-Decoupled Caching)**: Giảm di chuyển dữ liệu bằng cách lưu trữ tạm thông minh.
10. **Einsum lập lịch**: Hầu hết các yếu tố trên có thể được biểu diễn bằng Einsum + lịch trình.
11. **Suy ra vòng lặp**: Từ Einsum lập lịch, có thể tạo ra cách thực thi bằng vòng lặp.
12. **Đa dạng biến thể**: Có nhiều cách thú vị để tăng tốc spMspM, tùy vào cách tổ chức và xử lý.

Phần này nhấn mạnh rằng việc thiết kế bộ tăng tốc ma trận thưa rất linh hoạt, phụ thuộc vào cách quản lý dữ liệu thưa (tiling, giao nhau, định dạng) và luồng tính toán (pipeline, swizzle). Các công cụ như Einsum giúp đơn giản hóa thiết kế, và có nhiều hướng để tối ưu hiệu suất trong học sâu!