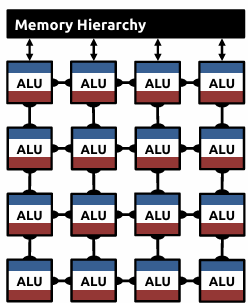
**LECTURE 11: DATAFLOW**

**I. Luồng dữ liệu và phân cấp bộ nhớ**

**1. Mô hình tính toán không gian (Spatial Compute Paradigm)**



**Gồm các thành phần:**

***Phân cấp bộ nhớ (Memory Hierarchy)***

+ Kết nối với tất cả các ALU trong hàng đầu tiên thông qua các mũi tên hai chiều.

+ Cho phép truy cập dữ liệu hai chiều giữa bộ nhớ và các đơn vị xử lý.

***Ma trận ALU (Arthmetic Logic Units):***

+ Được sắp xếp theo lưới 4x4 gồm 16 ALU

+ Mỗi ALU được kết nối với các ALU lân cận theo chiều dọc

+ Thiết kế này cho phép:

- Xử lý song song nhiều phép tính.

- Truyền dữ liệu hiệu quả giữa các ALU

- Tối ưu hóa luồng dữ liệu trong ma trận

***Luồng dữ liệu (Dataflow):***

+ Dữ liệu có thể di chuyền:

- Theo chiều dọc giữa các ALU

- Giữa bộ nhớ và hàng ALU đầu tiên

Kiến trúc này phù hợp cho:

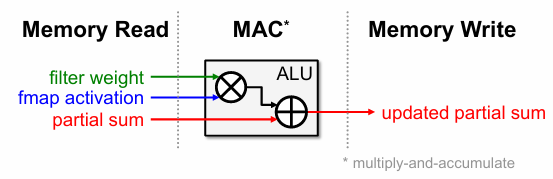
+ Các thuật toán xử lý ma trận

+ Tính toán song song

+ Ứng dụng xử lý dữ liệu quy mô lớn

Kiến trúc này đặc biệt hiệu quả cho các ứng dụng đòi hỏi xử lý song song và tính toán ma trận, như xử lý hình ảnh, học máy, và các phép tính khoa học.

**2. “Truy xuất bộ nhớ là nút thắt cổ chai”**

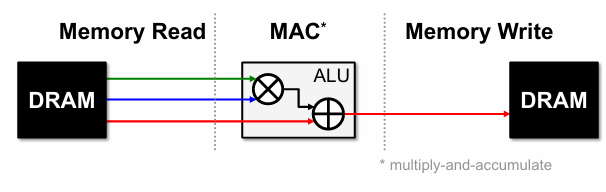


**Cho thấy cơ bản hoạt động của MAC:**

+ Có 3 đầu vào filter weight (xanh lá), fmap activation (xanh dương), và partial sum (đỏ)

+ ALU thực hiện phép nhân (⊗) và cộng (⊕)

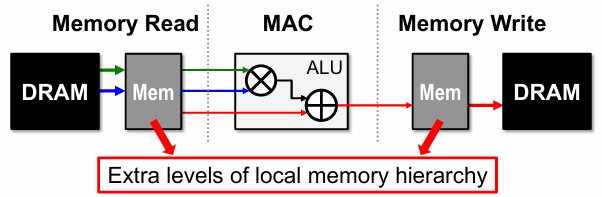
+ Kết quả là updated partial sum (đỏ)



***Minh họa vấn đề “nghẽn cổ chai” khi truy cập DRAM:***

+ Trường hợp xấu nhất là mọi thao tác đọc/ghi đều phải qua DRAM.

+ VD: Với AlexNet có 724M Macs thì cần phải yêu cầu 2896M lần truy cập DRAM



Giải pháp với bộ nhớ phân cấp:

+ Thêm các mức bộ nhớ cục bộ (Mem) giữa DRAM và ALU

+ Giải pháp này hiệu quả khi cường độ tính toán >1

+ Giúp giảm số lần truy cập trực tiếp vào DRAM

Cơ hội tối ưu:

+ Data reuse (tái sử dụng dữ liệu): dữ liệu đã đọc từ DRAM có thể được lưu trong bộ nhớ cục bộ

+ Giảm số lần truy cập DRAM bằng cách tái sử dụng dữ liệu từ bộ nhớ cục bộ

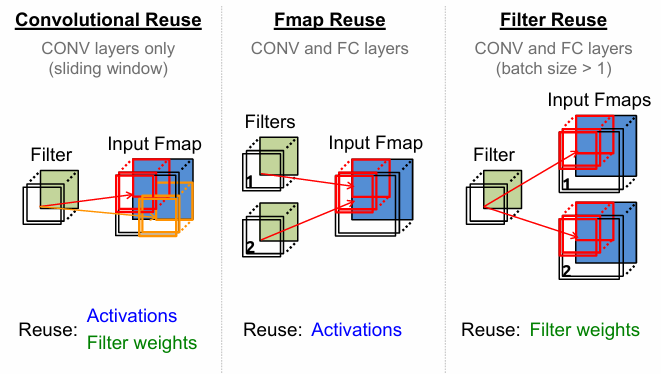
**Vậy vấn đề chính ở đây là:**

+ Truy cập DRAM là “nghẽn cổ chai” chính trong việc hiệu năng hệ thống.

+ Giải pháp là sử dụng bộ nhớ phân cấp và tối ưu việc tái sử dụng dữ liệu.

+ Điều này đặc biệt quan trọng trong các ứng dụng như neural networks nơi có rất nhiều phép MAC

**3. Các loại tái sử dụng dữ liệu trong mạng neural sâu (DNN):**



***a. Convolutional Reuse (Tái sử dụng tích chập):***

+ Chỉ áp dụng cho các lớp CONV

+ Sử dụng kỹ thuật cửa sổ trượt (Sliding Window)

+ Tái sử dụng hai thành phần:

- Thành phần kích hoạt (Activation) – màu xanh dương

- Filter weights (trọng số bộ lọc) – màu xanh lá

+ Một bộ lọc được áp dụng nhiều lần trên các vùng khác nhau của Input Feature Map

***b. Fmap Reuse (Tái sử dụng Feature Map):***

+ Áp dụng cho cả lớp CONV và FC (fully connected)

+ Chỉ tái sử dụng Activations (kích hoạt)

+ Nhiều bộ lọc khác nhau (Filters 1, 2,...) cùng sử dụng một vùng Input Feature Map

+ Giúp tận dụng dữ liệu đầu vào cho nhiều phép tính khác nhau

***c. Filter Reuse (Tái sử dụng Bộ lọc):***

+ Áp dụng cho cả lớp CONV và FC

+ Yêu cầu kích thước batch > 1

+ Chỉ tái sử dụng Filter weights (trọng số bộ lọc)

+ Cùng một bộ lọc được áp dụng cho nhiều Input Feature Maps khác nhau

+ Tối ưu việc sử dụng bộ lọc trên nhiều mẫu dữ liệu

**Lợi ích chính của các phương pháp tái sử dụng:**

+ Giảm số lần truy cập bộ nhớ

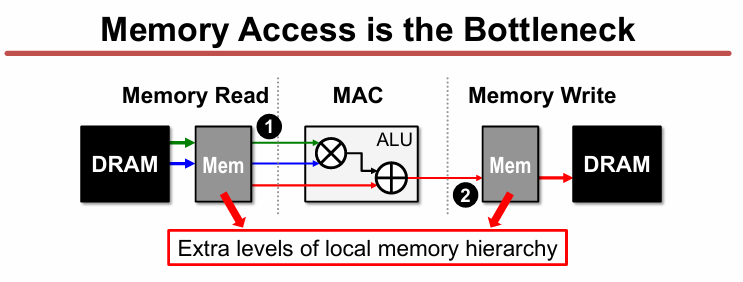
+ Tăng hiệu quả sử dụng dữ liệu

+ Cải thiện hiệu năng tính toán

+ Giảm độ trễ và tiêu thụ năng lượng

Mỗi phương pháp phù hợp với các tình huống khác nhau và có thể được kết hợp để tối ưu hóa hiệu suất của mạng neural.

**Quay lại vấn đề của nghẽn cổ chai trong truy xuất bộ nhớ**



**Về cơ hội/giải pháp đầu tiên: Data reuse (tái sử dụng dữ liệu)**

+ Có thể giảm số lần đọc DRAM của filter/fmap lên đến 500 lần

+ Áp dụng cho các lớp CONV trong mạng AlexNet

+ Sử dụng bộ nhớ cục bộ (Mem) làm bộ đệm giữa DRAM và ALU

**- Về cơ hội/giải pháp thứ hai: Local accumulation (tích lũy cục bộ)**

***Data reuse:***

+ Giảm đọc DRAM cho filter/fmap tới 500 lần

+ Dữ liệu được lưu trong bộ nhớ cục bộ để tái sử dụng

***Local accumulation:***

+ Partial sum (tổng từng phần) được tích lũy trong bộ nhớ cục bộ

+ Không cần truy cập DRAM cho quá trình tích lũy

+ Giảm đáng kể số lần truy cập DRAM

**Kết quả với AlexNet:**

+ Ban đầu: 2896M lần truy cập DRAM

+ Sau tối ưu: chỉ còn 61M lần (trường hợp tốt nhất)

**Cải thiện hiệu suất đáng kể nhờ:**

+ Tái sử dụng dữ liệu hiệu quả

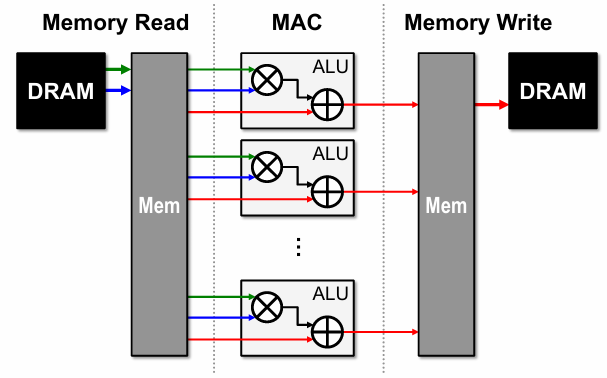
+ Tích lũy cục bộ thông minh

+ Phân cấp bộ nhớ hợp lý

Các phương pháp này giúp giảm đáng kể nút thắt cổ chai trong truy cập bộ nhớ DRAM.

**4. Leverage Parallelism – Tận dụng song song**

***a.Higher performance – Cải thiện hiệu suất***



Minh họa một cách tiếp cận khác để cải thiện hiệu suất, đó là leverage parallelism (tận dụng sự song song).

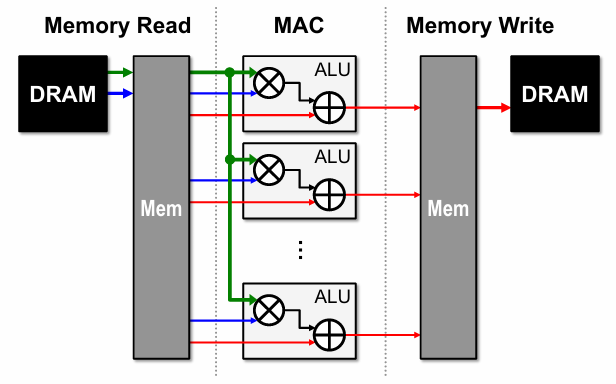
- Ở phần Memory Read, ta thấy có nhiều module Mem (bộ nhớ cục bộ) và ALU (đơn vị tính toán) được sắp xếp ngang hàng.

- Điều này cho phép thực hiện nhiều phép tính song song trên các module ALU khác nhau, tận dụng sức mạnh song song của hệ thống.

- Trong phần Memory Write, các module Mem và DRAM cũng được sắp xếp ngang hàng, cho phép đồng thời ghi dữ liệu vào DRAM.

-> Trọng tâm là tăng hiệu suất tổng thể của hệ thống, tối đa hóa lợi ích của việc tính toán song song, bất kể dữ liệu đầu vào/ đầu ra

***b. Spatial Data Reuse – Tối ưu tái sử dụng dữ liệu không gian***



Minh họa cách tận dụng sự song song để tối ưu việc tái sử dụng dữ liệu không gian (spatial data reuse).

Tương tự như slide 1, ta thấy có nhiều module ALU được sắp xếp ngang hàng, cho phép thực hiện nhiều phép tính song song.

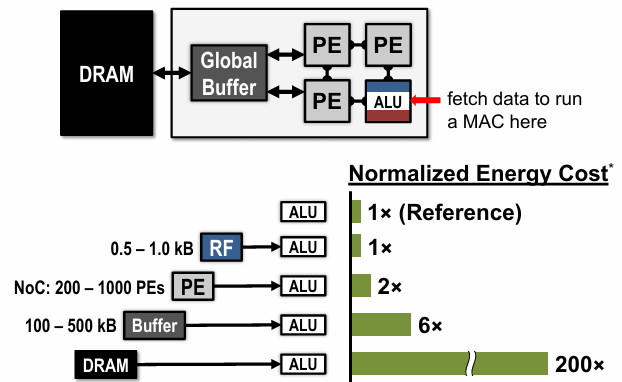
Điều này giúp tận dụng tối đa việc tái sử dụng dữ liệu đầu vào (input) và tích lũy từng phần (partial sum) trên các module ALU.

Kết quả là có thể giảm số lần truy cập DRAM, cải thiện hiệu suất tính toán.

-> Trọng tâm là tối ưu hóa việc tái sử dụng dữ liệu không gian, giảm số lần truy cập DRAM.

Nhìn chung, hai slide này đều minh họa cách tận dụng sức mạnh song song của hệ thống bằng các cấu trúc phân cấp bộ nhớ và tối ưu hóa việc tái sử dụng dữ liệu. Điều này giúp giảm đáng kể số lần truy cập DRAM, là một trong những điểm nghẽn chính về hiệu suất.

**5. Truy cập dữ liệu cục bộ giá thấp**

****

Minh họa một bộ đệm toàn cục (Global Buffer) nằm giữa DRAM và các đơn vị xử lý ALU.

Bộ đệm này cho phép lấy dữ liệu cho các phép MAC (nhân-cộng) từ nguồn gần hơn và tiết kiệm năng lượng hơn so với truy cập trực tiếp DRAM.

Biểu đồ năng lượng cho thấy lợi ích đáng kể từ việc sử dụng bộ đệm cục bộ so với truy cập DRAM.

Câu hỏi đặt ra là: ***Làm thế nào để khai thác việc tái sử dụng dữ liệu và tích lũy cục bộ với lưu trữ cục bộ chi phí thấp hạn chế?***

+ Chìa khóa ở đây là cần có luồng xử lý dữ liệu đặc biệt để tận dụng tối đa lợi ích từ tái sử dụng dữ liệu và tích lũy cục bộ.

+ Biểu đồ năng lượng tiếp tục đưa ra những con số cải thiện ấn tượng, lên đến 6 lần cho bộ đệm cục bộ và 200 lần cho việc tránh truy cập DRAM.

Tóm lại, các kỹ thuật kiến trúc và thuật toán tinh vi có thể giảm đáng kể chi phí năng lượng và hiệu suất của việc truy cập bộ nhớ, một nút thắt cổ chai quan trọng. Tối ưu hóa tái sử dụng dữ liệu và tích lũy cục bộ được nhấn mạnh là những cách tiếp cận có tác động lớn.