System for Artificial Intelligence

Hardware Acceleration

Siyuan Feng
Shanghai Innovation Institute

Recap: Overview of Machine Learning Systems

上海创智学院
Shanghai Innovation Institute

ML Models

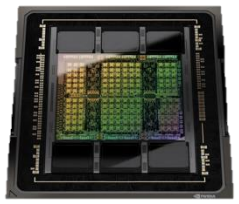
Automatic Differentiation

Graph Optimization

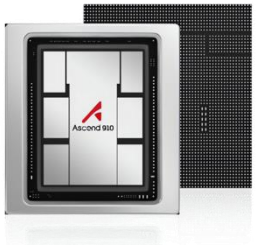
Parallelism / Distributed

Hardware Acceleration

← This Lecture



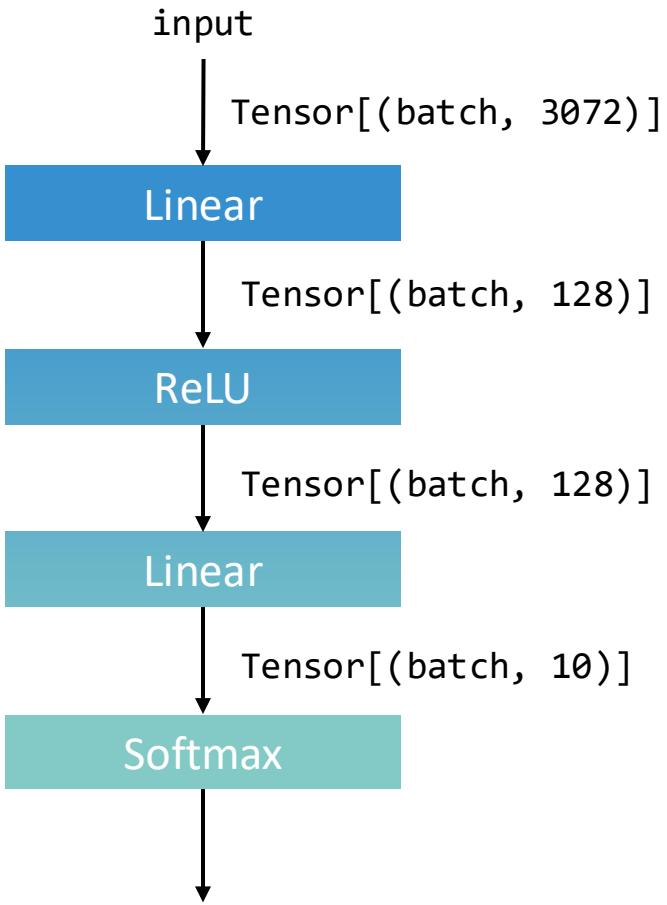
NVIDIA GPU



HUAWEI NPU

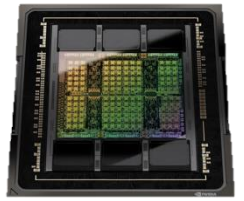
Mobile devices

Discussion: How to run a ML model w/o MLSys



?

CUDA

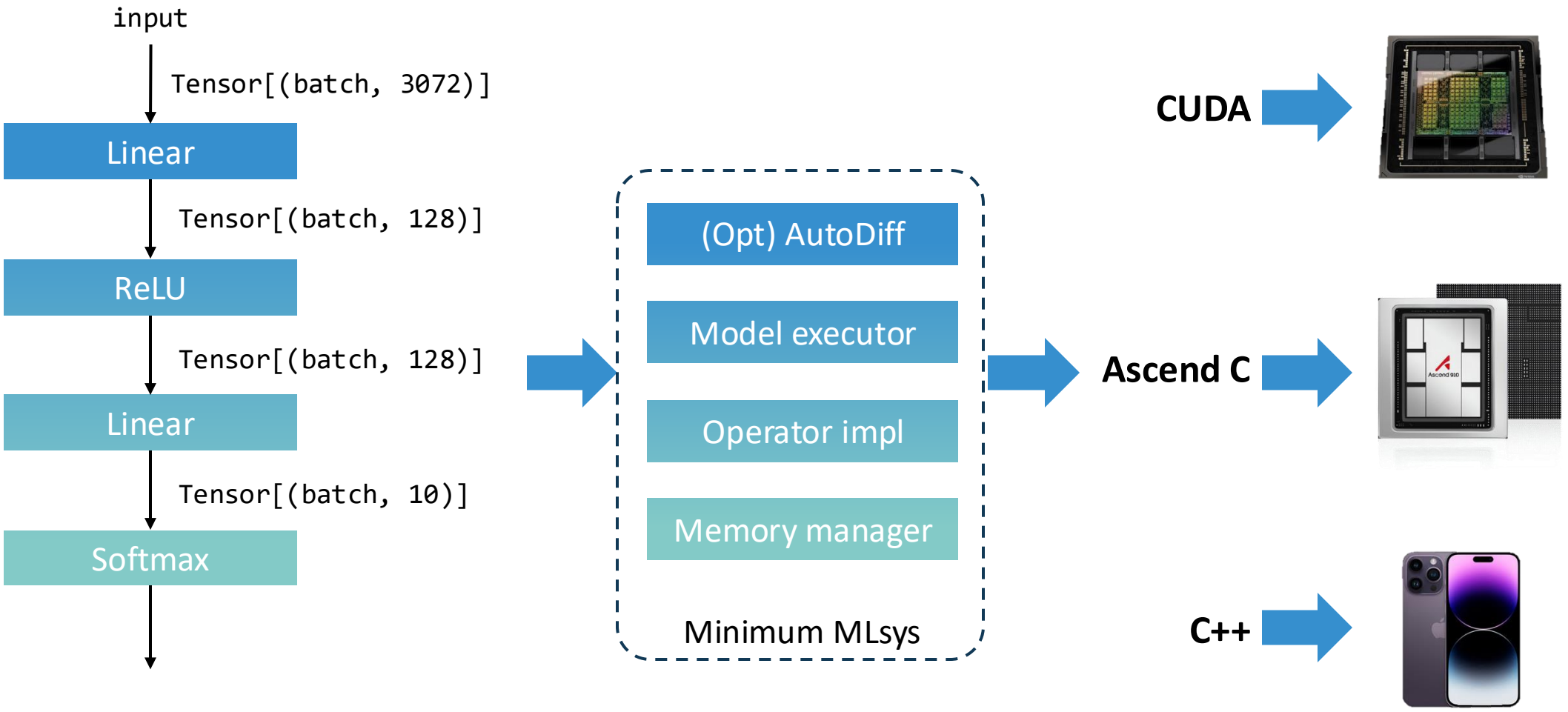


Ascend C

C++

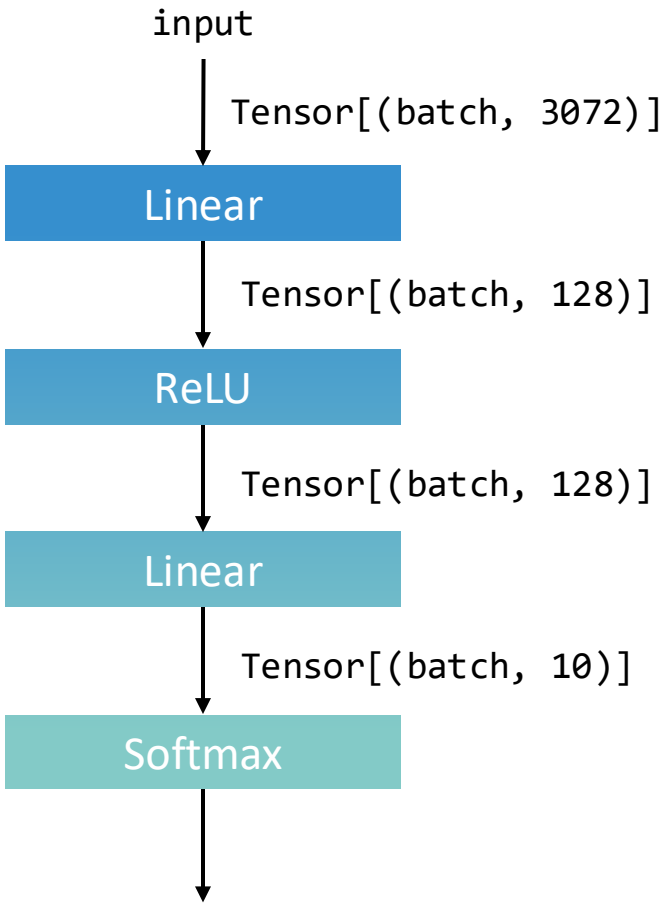
Discussion: How to run a ML model w/o MLSys

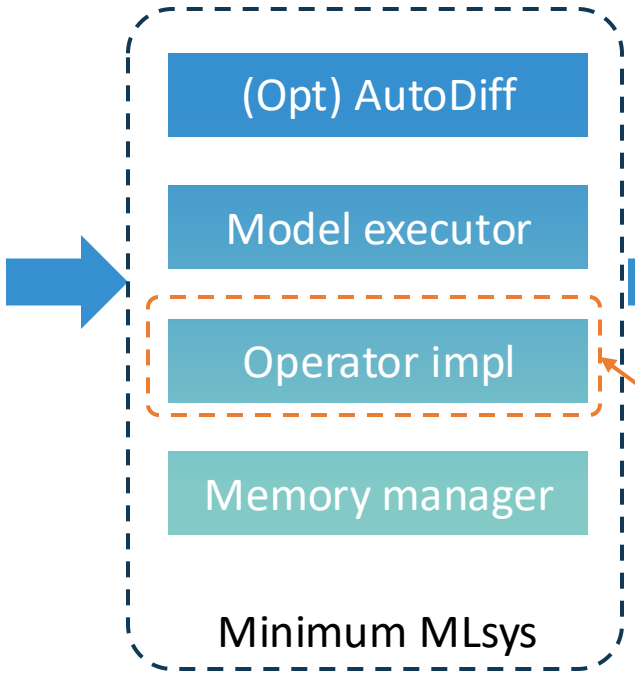
上海创智学院
Shanghai Innovation Institute

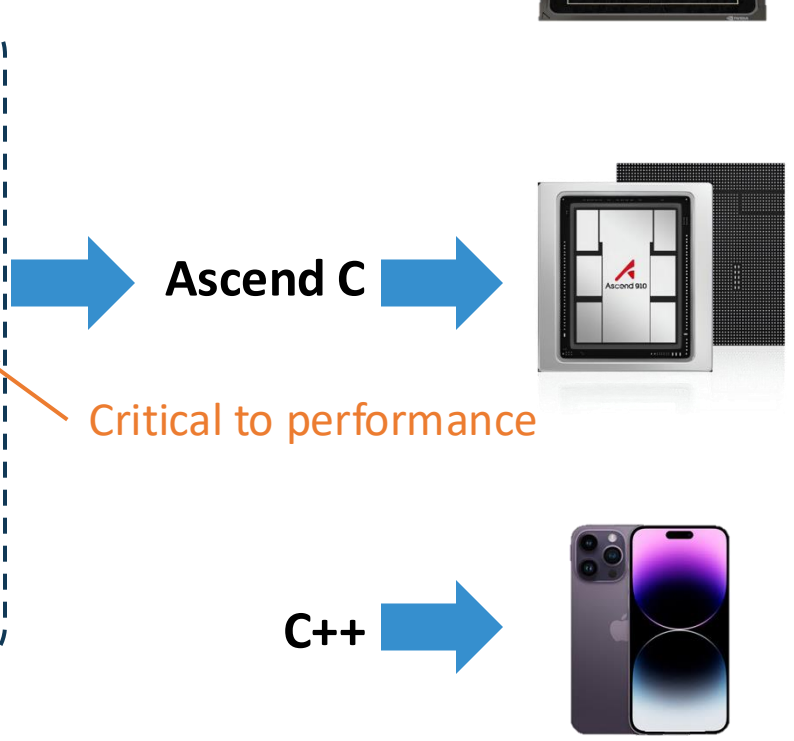


Discussion: How to run a ML model w/o MLSys

上海创智学院 Shanghai Innovation Institute







OUTLINE

01 General acceleration techniques

02 Case study: matrix multiplication

01

General acceleration techniques

Vectorization

上海创智学院
Shanghai Innovation Institute

Adding two float32 arrays of length 256, one float uses 32bit/4Byte memory

```
void vec_add(float* A, float *B, float* C) {
  for (int i = 0; i < 64; ++i) {
    float4 a = load_float4(A + i*4);
    float4 b = load_float4(B + i*4);
    float4 c = add_float4(a, b);
    store_float4(C + i*4, c);
  }
}
```

Additional requirements: memory (A, B, C) needs to be aligned to 128 bits

Parallelization

上海创智学院
Shanghai Innovation Institute

```
void parallel_vec_add(float* A, float *B, float* C) {
  #pragma omp parallel for
  for (int i = 0; i < 64; ++i) {
    float4 a = load_float4(A + i*4);
    float4 b = load_float4(B + i*4);
    float4 c = add_float4(a, b);
    store_float4(C + i*4, c);
  }
}
```

Executes the computation on multiple threads

02

Case study: matrix multiplication

Vanilla Matrix Multiplication

上海创智学院
Shanghai Innovation Institute

Compute `C = dot(A, B.T)`

```
float A[n][n], B[n][n], C[n][n];

for (int i = 0; i < n; ++i) {
  for (int j = 0; j < n; ++j) {
    C[i][j] = 0;
    for (int k = 0; k < n; ++k) {
      C[i][j] += A[i][k] * B[j][k];
    }
  }
}
```

$\Theta(N^3)$

Strassen Algorithm: Reduce Complexity

上海创智学院
Shanghai Innovation Institute

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Thus, we have

$$C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21}$$
$$C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22}$$
$$C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21}$$
$$C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}$$

8 multiplications in normal algorithm

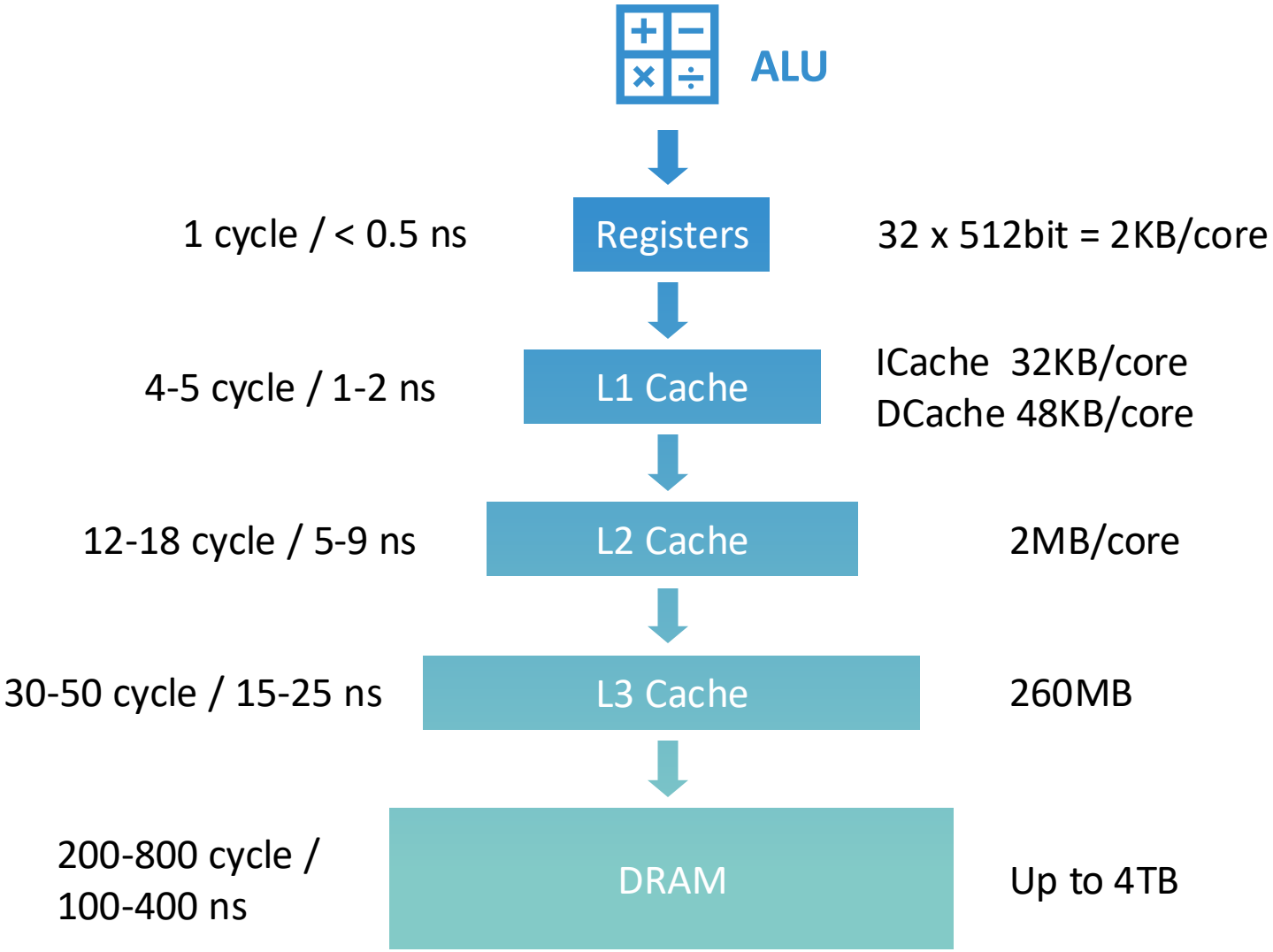
$$M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22}) \cdot B_{11}$$
$$M_3 = A_{11} \cdot (B_{12} - B_{22})$$
$$M_4 = A_{22} \cdot (B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12}) \cdot B_{22}$$
$$M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$$

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 - M_2 + M_3 + M_6 \end{bmatrix}$$

7 multiplications in Strassen algorithm

The general complexity is $\Theta\left(n^{\log_2 7}\right)$

Memory Hierarchy on Modern CPUs



Intel Xeon Platinum 8558 CPU memory hierarchy

Architecture Aware Analysis

上海创智学院
Shanghai Innovation Institute

```
dram float A[n][n], B[n][n], C[n][n];

for (int i = 0; i < n; ++i) {
  for (int j = 0; j < n; ++j) {
    register float c = 0;
    for (int k = 0; k < n; ++k) {
      register float a = A[i][k];
      register float b = B[j][k];
      c += a * b;
    }
    C[i][j] = c;
  }
}
```

A's dram->register time cost: n^3
B's dram->register time cost: n^3

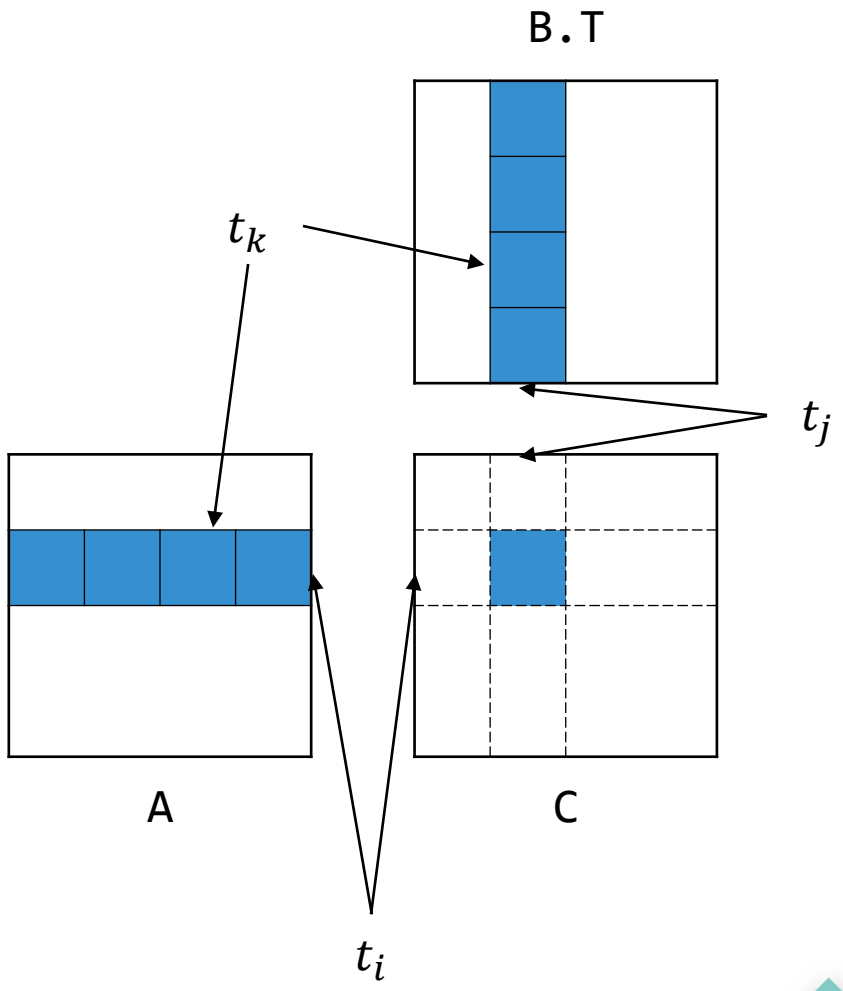
Load cost: $2 * C_{dram \to reg} * n^3$

Register Tiled Matrix Multiplication

上海创智学院
Shanghai Innovation Institute

```
dram float A[n/ti][n/tk][ti][tk];
dram float B[n/tj][n/tk][tj][tk];
dram float C[n/ti][n/tj][ti][tj];

for (int i = 0; i < n/ti; ++i) {
  for (int j = 0; j < n/tj; ++j) {
    register float c[ti][tj] = 0;
    for (int k = 0; k < n/tk; ++k) {
      register float a[ti][tk] = A[i][k];
      register float b[ti][tk] = B[j][k];
      c += dot(a, b.T);
    }
    C[i][j] = c;
  }
}
```



Register Tiled Matrix Multiplication

上海创智学院
Shanghai Innovation Institute

```
dram float A[n/ti][n/tk][ti][tk];
dram float B[n/tj][n/tk][tj][tk];
dram float C[n/ti][n/tj][ti][tj];

for (int i = 0; i < n/ti; ++i) {
  for (int j = 0; j < n/tj; ++j) {
    register float c[ti][tj] = 0;
    for (int k = 0; k < n/tk; ++k) {
      register float a[ti][tk] = A[i][k];
      register float b[tj][tk] = B[j][k];
      c += dot(a, b.T);
    }
    C[i][j] = c;
  }
}
```

A's dram->register time cost: n^3/t_j

B's dram->register time cost: n^3/t_i

Load cost: $C_{dram \to reg}\left(\frac{n^3}{t_i} + \frac{n^3}{t_j}\right)$

Discussion: Why need t_k

Register Tiled Matrix Multiplication

上海创智学院
Shanghai Innovation Institute

```
dram float A[n/ti][n/tk][ti][tk];
dram float B[n/tj][n/tk][tj][tk];
dram float C[n/ti][n/tj][ti][tj];

for (int i = 0; i < n/ti; ++i) {
  for (int j = 0; j < n/tj; ++j) {
    register float c[ti][tj] = 0;
    for (int k = 0; k < n/tk; ++k) {
      register float a[ti][tk] = A[i][k];
      register float b[tj][tk] = B[j][k];
      c += dot(a, b.T);
    }
    C[i][j] = c;
  }
}
```

A's register memory cost: $T_i \times T_k$
B's register memory cost: $T_j \times T_k$
C's register memory cost: $T_i \times T_j$

Load cost: $C_{dram \to reg} \left(\frac{n^3}{T_i} + \frac{n^3}{T_j} \right)$

Register cost: $T_i \times T_k + T_j \times T_k + T_i \times T_j$

Cache Line Aware Tiling

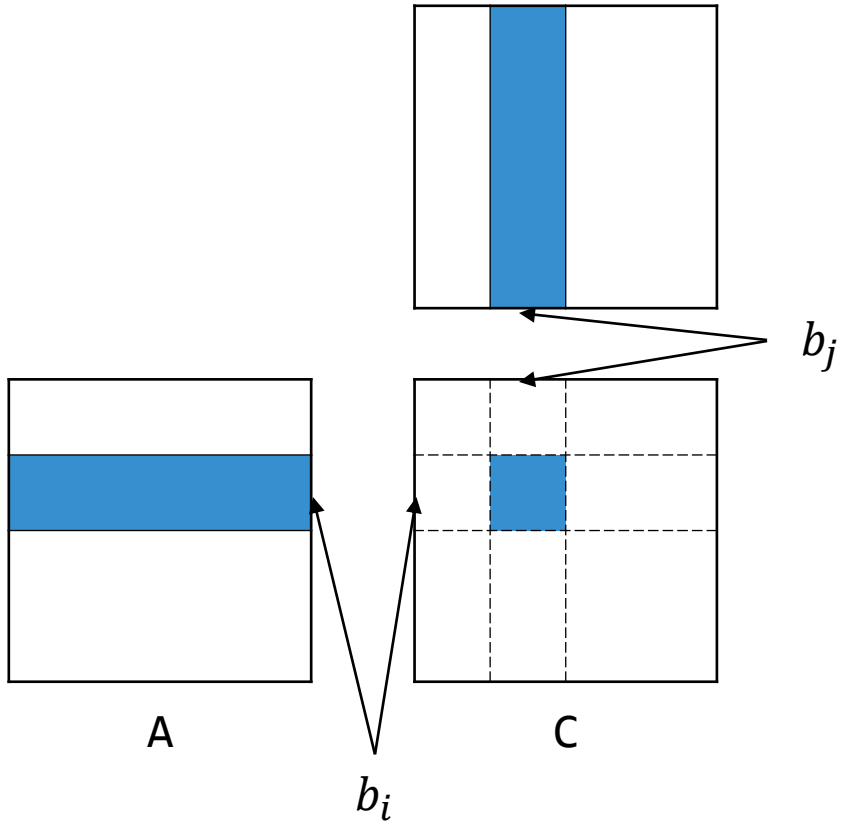
上海创智学院
Shanghai Innovation Institute

```
dram float A[n/bi][bi][n];
dram float B[n/bj][bj][n];
dram float C[n/bi][n/bj][bi][bj];

for (int i = 0; i < n/bi; ++i) {
  l1_cache float a[bi][n] = A[i];
  for (int j = 0; j < n/bj; ++j) {
    l1_cache b[bj][n] = B[j];

    C[i][j] = dot(a, b.T);
  }
}
```

Sub-procedure, can apply register tiling here



Question: Cache used and copy cost (dram->l1_cache)

Cache Line Aware Tiling

上海创智学院
Shanghai Innovation Institute

```
dram float A[n/bi][bi][n];
dram float B[n/bj][bj][n];
dram float C[n/bi][n/bj][bi][bj];

for (int i = 0; i < n/bi; ++i) {
  l1_cache float a[bi][n] = A[i];
  for (int j = 0; j < n/bj; ++j) {
    l1_cache b[bj][n] = B[j];

    C[i][j] = dot(a, b.T);
  }
}
```

A's dram->register time cost: n^2
B's dram->register time cost: n^3/b_i

A's cache memory cost: $b_i \times n$
B's cache memory cost: $b_j \times n$

Putting everything together

上海创智学院
Shanghai Innovation Institute

```
dram float A[n/bi][bi/ti][n][ti];
dram float B[n/bj][bj/tj][n][tj];

for (int i = 0; i < n/bi; ++i) {
  l1_cache float a[bi/ti][n][ti] = A[i];
  for (int j = 0; j < n/bj; ++j) {
    l1_cache b[bj/tj][n][tj] = B[j];
    for (int x = 0; x < bi/ti; ++x)
    for (int y = 0; y < bj/tj; ++y) {
      register float c[ti][tj] = 0;
      for (int k = 0; k < n; ++k) {
        register float ar[ti] = a[x][k][:];
        register float br[tj] = b[y][k][:];
        C += dot(ar, br.T)
      }
    }
  }
}
```

Load cost: $C_{dram\to l1} \times \left(n^2 + \frac{n^3}{b_i}\right) + C_{l1\to reg} \times \left(\frac{n^3}{t_i} + \frac{n^3}{t_j}\right)$

Key insight: Memory Load Reuse

上海创智学院
Shanghai Innovation Institute

```
float A[n][n];
float B[n][n];
float C[n][n];
```

```
C[i][j] = sum(A[i][k] * B[j][k], axis=k)
```

Access of A is independent of j,
tile the j dimension by t enables reuse of A for t times.

Possible Reuse Pattern in Convolution

上海创智学院
Shanghai Innovation Institute

```
float Input[n][ci][h][w];
float Weight[co][ci][K][K];
float Output[n][co][h][w];

Conv[b][co][y][x] =
    sum(Input[b][k][y+ry][x+rx] * Weight[co][k][ry][rx],
        axis=[k, ry, rx])
```

Acknowledgement

The development of this course, including its structure, content, and accompanying presentation slides, has been significantly influenced and inspired by the excellent work of instructors and institutions who have shared their materials openly. We wish to extend our sincere acknowledgement and gratitude to the following courses, which served as invaluable references and a source of pedagogical inspiration:

- Machine Learning Systems[15-442/15-642], by **Tianqi Chen** and **Zhihao Jia** at **CMU**.
- Advanced Topics in Machine Learning (Systems)[CS6216], by **Yao Lu** at **NUS**

While these materials provided a foundational blueprint and a wealth of insightful examples, all content herein has been adapted, modified, and curated to meet the specific learning objectives of our curriculum. Any errors, omissions, or shortcomings found in these course materials are entirely our own responsibility. We are profoundly grateful for the contributions of the educators listed above, whose dedication to teaching and knowledge-sharing has made the creation of this course possible.

System for Artificial Intelligence

Thanks

Siyuan Feng
Shanghai Innovation Institute