Computational Models for Parallel Accelerators

HW/Theory Reading Group

Grace Dinh
6 April 2022

UC Berkeley
“Fundamental” operations in traditional models of computation: bitwise operations (AND, OR, XOR, NOT, etc.) or arithmetic operations (+, −, ×, /).

Arithmetic circuits, Turing machines...

But in real chips have parallelism, and only in restricted shapes
Motivation

Vector parallelism (image source: Intel AVX-512 docs):
Motivation

NVIDIA TCU:
Motivation

How to **model** these forms of “restricted” parallelism?

Treat parallelism as constant factor: “the ostrich approach” – bury your head in the sand and just ignore hardware factors

Multi-processor models (e.g. NC): don’t account for far more limited forms of parallelism for these architectures, communication costs, etc.

Solution: new computational models with new primitives
Fundamental operation: matrix multiplication

Specifically: a \((m, l)\)-TCU model, a \(\sqrt{m} \times \sqrt{m} \times \sqrt{m}\) matrix multiplication in \(O(m + l)\) time.

Intuition: \(\sqrt{m} \times \sqrt{m}\) systolic array with \(l\) latency cost per instruction.

Asymmetric matmuls \((n \times \sqrt{m} \times \sqrt{m})\) can be performed in \(O(n \sqrt{m} + l)\) time.
Very related to models we’ve looked at!

**Theorem:** Suppose some problem $P$ with two input matrices and one output matrices has a lower bound $F_p$ on communication complexity on a system with memory $M = 3m + O(1)$ with constant block size. Then any algorithm for $P$ in the weak TCU model (i.e. no ability to take advantage of “long” matrices) requires at least $\Omega(F_p)$ time.

**Proof:** each call to tensor unit is equivalent to loading two $\sqrt{m} \times \sqrt{m}$ matrices into memory and computing on the output.
Strassen-like matmul splits each matrix into $n_0$ equally-sized pieces and performs $p_0$ recursive matmuls on the matrices, for a runtime of $O(n^{w_0 := \log_{n_0} p_0})$. Classical $n^3$ matmul is $(n_0, p_0) = (4, 8)$, Strassen is $(4, 7)$.

**Theorem [1]:** exists a $(m, l)$-TCU algorithm performing $\sqrt{n} \times \sqrt{n} \times \sqrt{n}$ matmul (s.t. $m \geq n_0$) in time

$$T(n) = O \left( \left( \frac{n}{m} \right)^{\omega_0} (m + l) \right)$$

**Proof:** Perform recursive matmul until small enough to fit inside $\sqrt{m}$. Theorem comes immediately from solving the recurrence:

$$T_n = \begin{cases} 
O \left( \frac{n^{3/2}}{m^{1/2}} + \frac{n}{m} \right) & \text{if } m \leq n \leq mn_0 \\
 p_0 T(n/n_0) + O(n) & \text{otherwise}
\end{cases}$$
Cooley-Tukey: break an FFT of size \( n = n_1 n_2 \) into (a) \( n_1 \) subproblems of size \( n_2 \), (b) multiplication by roots of unity (\( O(n) \) size operation), and (c) \( n_2 \) operations of size \( n_1 \).

**Theorem:** DFT of a vector with \( n \) entries on \((m, l)\)-TCU costs \( O((n + l) \log_m n) \).

**Proof:** set \( n_1 = \sqrt{m} \), \( n_2 = n / \sqrt{m} \). Cost of (a): \( \sqrt{m} T(n/\sqrt{m}) \). Cost of (b): \( O(n) \). Cost of (c): a single ‘long multiplication’, \( O(n + l) \)

\[
T = \begin{cases} 
\sqrt{m} T(n/\sqrt{m}) + O(n + l) & n > m \\
O(m + l) & n \leq m 
\end{cases}
\]

Solving gives the desired result.
Other Results

Sparse matmul, evenly balanced output sparsity:
\[ O \left( \sqrt{\frac{n}{\text{nnz(out)}}} \left( \frac{\text{nnz(out)}}{m} \right)^{\omega_0} (m + l) + \text{nnz(in)} \right) \]

APSP on n-vertex graph: \( O((n^2/m)^{\omega_0} (m + l) \log n) \)
Thanks for listening! Questions?