Communication Lower Bounds with HBL and Polyhedral Models

HW/Theory Reading Group

Grace Dinh

09 & 16 March 2022

UC Berkeley

Agenda

Model: CPU w/ cache size $M \leftrightarrow$ unlimited slow memory.

Goal: find **lower bound** and **upper bound** (ideally constructive, i.e. actually achievable) on the communication cost.

Interesting open problems will be denoted with 🤔 .

Last time...

Matrix multiplication communication lower bound:

$$\frac{\text{(number of operations)}}{M^{1/2}}$$

General strategy: in the *iteration space* (coordinates corresponding to iteration indices; points corresponding to instructions), bound **volume** of set of points (instructions) subject to constraints on sizes of **projections**.

Use Loomis-Whitney inequality: given some set $S \subset \mathbb{Z}^3$ and projections into 2D S_x, S_y, S_z,

$$|S| \leq |S_x|^{1/2}|S_y|^{1/2}|S_z|^{1/2}$$

Actually finding the tiling

How do we actually get a communication-avoiding algorithm? loop tiling.

Goal: actually find some set S ("tile") such that S actually attains $|S| = |S_x|^{1/2}|S_y|^{1/2}|S_z|^{1/2}$.

Ansatz: S is an $T_x \times T_y \times T_z$ parallelepiped. To find dimensions, maximize $T_xT_yT_z$ subject to $T_xT_y + T_yT_z + T_xT_z \leq M$; either solve numerically ("sigmoidal programs")...

... or relax to maximizing $B_x + B_y + B_z$ s.t. $B_x + B_y \leq 1$, $B_y + B_z \leq 1$, and $B_x + B_z \leq 1$ where $B_* = \log_M T_*$.

Obviously the optimal subtile is a cube, but if, say, T_x is larger than the loop bound in that direction, tile would be illegal, so we add a limits on the tile size. **Theorem:** this is still optimal! (hits some stronger lower bound that only exists for "small" problems).

Can we generalize beyond matmul?

Generalizing...

Given program that accesses multidimensional arrays $A_1, ..., A_n$:

```
for $x_1 \in [L_1], ..., x_d \in [L_d]$ :
    Do computation using $A_1(\phi_1(x_1, ..., x_d)), ...A_n(\phi_n(x_1, ..., x_d))$
```

For example, in a 2D convolution defined by:

```
$for\ \{b, c, k, w, h, r, s\} = 0 : \{B, C, K, W, H, R, S\} - 1$
    $Out(k, h, w, b) += Image(r + \sigma_w w, s + \sigma_h h, c, b) \times Filter(k, r, s, c)$
```

we would have

$$\phi_2(b, c, k, w, h, r, s) = (r + \sigma_w w, s + \sigma_h h, c, b)$$

Limit ourselves to rearranging the program (goal: find optimal blockings).

Communication Lower Bound

Split our program into *tiles* of instructions, each of which consumes exactly M words of memory movement between cache and slow memory.

What's the fewest number of tiles we can do this with?

M words in cache at start of title. Read/write at most M more words. **Number of distinct memory addresses touched per tile $\leq 2M$.**

What's the biggest tile we can make such that the number of memory locations accessed is at most O(M)?

$$\text{communication lower bound} = \Omega\left(\frac{\text{\# of operations}}{\text{max \# of instructions per tile}} M\right)$$

Bounding the Tile Size

Corollary to the discrete Brascamp-Lieb inequality [3] says:

$$\text{\# of operations/tile} \leq \prod_{i\in[n]} M^{p_i}$$

for **any** p_i satisfying:

$$\text{rank}(V) \leq \sum_j p_j \text{rank}(\phi_j(V))$$

for all additive subgroups V of $\mathbb{Z}^d$ (more on this later).

Let $p_{HBL} := \min \sum_i p_i$ under the BL constraints above. Then no tile can be bigger than $M^{p_{HBL}}$, so the communication lower bound is $(\# \, \textit{operations})/M^{p_{HBL}-1}$ words.

How do we enumerate the constraints?

Enumerating Brascamp-Lieb Constraints

$\text{rank}(V) \leq \sum_j p_j \text{rank}(\phi_j(V))$ for all additive subgroups V of $\mathbb{Z}^d$: **polyhedron over p_j**

Theorem [Valdimarsson '10]: can replace "for all subgroups" with "for all element of the lattice of subgroups generated by the kernels of the ϕ_j, where the lattice's operations are intersections and direct sums.

Leads to unbounded-time enumeration algorithm in general case, but tractable for independent or "separable" kernels (as for CNNs).

Theorem [Corollary to Garg-Gurvits-Olivera-Wigderson '16]: can enumerate all the facets in double-exponential time in the number of loops/arrays, or optimize over polyhedron in exponential time - currently best known algorithm. (more on this in extra slides at end)

Duality

How to find actual tile?

HBL LP: lower bound exponent given by min $\sum_i p_i$ s.t.

$$\text{rank}(V) \leq \sum_j p_j \text{rank}(\phi_j(V))$$

Taking the dual gives a maximization problem that gives the optimal tile size! (under certain assumptions, e.g. large loop bounds)

For small loop bounds, add additional constraints to the dual to bound problem size. Provably optimal (hits a stronger lower bound, found by bounding 'slices' of the original problem and summing) for when ϕ_i are all projections.

Optimal Tilings

Generally: Ansatz tile shape (often rectangular), then numerically maximize tile size w.r.t. memory footprint constraints.

Provably optimal (up to asymptotic factors) for certain problems (stride-1 CNNs, or when ϕ_i are projections).

Works well in practice in other cases, or when additional constraints (e.g. minimum tile size to ensure full utilization of a systolic array) or variables need to be imposed for hardware reasons.

Loop Ordering

In tiling, we went from:

```
for $\{x_1, x_2, x_3\} = 0 : \{L_1, L_2, L_3\} - 1$
    $A_1(x_1, x_2) += A_2(x_1, x_3) \times A_3(x_2, x_3)$
```

to

```
for $\{x_1^\dagger, x_2^\dagger, x_3^\dagger\} = 0 : \{L_1/T_1, L_2/T_2, L_3/T_3\} - 1$
    for $\{x_1^{'}, x_2^{'}, x_3^{'}\} = 0 : \{T_1, T_2, T_3\} - 1$
        $x_i = x_i^\dagger T_i + x_i^{'}$
        $A_1(x_1, x_2) += A_2(x_1, x_3) \times A_3(x_2, x_3)$
```

Current loop order: $x_1^\dagger, x_2^\dagger, x_3^\dagger, x_1^{'}, x_2^{'}, x_3^{'}$. Can we get better results by swapping orders?

Loop Ordering Reuse

Which order to place loops?

Loops “inside” the tile irrelevant: it’s all on fast memory, so arrangement doesn’t affect on/off chip communication (i.e. just optimize for architecture).

Loops “outside” the tile need to be ordered correctly. Constant factor, but an important one... for people here familiar with neural nets: corresponds to weight-stationary, output-stationary, input-stationary dataflows.

Intuition: **better bookkeeping to determine how much can be reused between tiles.**

Densities

```
for $x_1 \in [L_1], ..., x_d \in [L_d]$ :
    Do computation using $A_1(\phi_1(x_1, ..., x_d)), ...A_n(\phi_n(x_1, ..., x_d))$
```

Definition: Let the *subdomain* SD_j of a loop dimension x_j be the "slice" formed by fixing all x_i for $i < j$, let its *subdomain data footprint for array* A_i, denoted $SDF_{A_i j}$ be the total memory footprint of the entire nested loop enclosed by dimension x_j. Then the *inter-subdomain reuse* $SDR_{A_i,j}$ is defined as the number of elements from A that may be kept in memory between successive iterations of the x_j loop.

Now define *inverse density*:

$$ID_{A_{i,},j}(x_1, ..., x_j \neq 0) := \frac{SDF_{A_i,j} - SDR_{A_i,j}}{|SD_j|}$$

$$ID_{A_{i,},j}(x_1, ..., x_j = 0) := \frac{SDF_{A_i,j}}{|SD_j|}$$

Finding an optimal loop ordering

Suppose we have a tiled loop nest. Let level l be the outermost dimension whose entire subdomain data footprint fits in fast memory. Then the communication cost for array A is:

$$C = ID_{A,,l}(x_1, \ldots, x_l = 0) \prod_{i \in [1, l-1]} L_i + ID_{A,,l}(x_1, \ldots, x_l \neq 0)(1 - L_l) \prod_{i \in [1, l-1]} L_i$$

Finding optimal loop ordering now just a matter of minimizing the total cost.

[Olivry et al. '21]: let there be *reuse* for some array A, dimension d if, when putting dimension d innermost, the SDF at the second-innermost level is not significantly larger than the SDF at the innermost level.

Then, while there is reuse present for some array and dimension, place that dimension "inside", update the list of dimensions/arrays where there is reuse present, and repeat. Once out of reuse, use arbitrary permutation.

Thanks for listening! Questions?

Extra Slides on BL Inequalities

Brascamp-Lieb Inequalities

Given a tuple $\boldsymbol{B}$ of m functions $B_i : \mathbb{R}^n \to \mathbb{R}^{n_i}$, and a tuple $\boldsymbol{p}$ of nonnegative reals p_i, there exists some constant $C \in (0, \infty]$ such that for **all** integrable nonnegative functions f_i:

$$\int_{x \in \mathbb{R}^n} \prod_{j \in [m]} \left(f_j\left(B_j x\right)\right)^{p_j} d_x \leq C \prod_{j \in [m]} \left(\int_{x_j \in \mathbb{R}^{n_j}} f_j\left(x_j\right) dx_j\right)^{p_j}$$

Generalizes Holder's, Loomis-Whitney, sharp Young's inequalities (among others). Let f_j be indicator functions that are 1 if and only if something is in the shadow...

Questions:

1. Given $(\boldsymbol{B}, \boldsymbol{p})$, find $\mathrm{BL}(\boldsymbol{B}, \boldsymbol{p}) := C$.
2. Given $\boldsymbol{B}$, for which $\boldsymbol{p}$ is C finite?

The Brascamp-Lieb Polytope

1. [8] **Given $(\boldsymbol{B}, \boldsymbol{p})$, find C:** If (B, p) is feasible, the BL constant is:

$$\left[\sup_{X_j \succ 0} \frac{\prod_j (\det X_j)^{p_j}}{\det\left(\sum_j p_j B_j^\dagger X_j B_j\right)}\right]^{1/2}$$

   (Try plugging in Gaussians).

2. [2] **Given $\boldsymbol{B}$, for which $\boldsymbol{p}$ is C finite?** C is finite if and only if the following two constraints hold

    1. $n = \sum_j p_j n_j$
    2. $\dim(V) \leq \sum_j p_j \dim(B_j V) \qquad$ for all subspaces V of $\mathbb{R}^n$

   The set of such p is known as the BL polytope.

The Scaling Approach to BL

Theorem [1]: If a BL datum satisfies these two conditions ("is geometric"), then the corresponding constant is 1.

- **Projection:** $B_j B_j^\dagger = I$ for all j.
- **Isotropy:** $\sum_j p_j B_j^\dagger B_j = I$.

Theorem [2]: Suppose there exist matrices C, C_j such that $B_j' = C_j^{-1} B_j C$. Then

$$\mathrm{BL}(\boldsymbol{B}', \boldsymbol{p}) = \frac{\prod_j \left(\det\left(C_j\right)\right)^{p_j}}{\det(C)} \mathrm{BL}(\boldsymbol{B}, \boldsymbol{p})$$

Intuition: try to scale $\boldsymbol{B}$ to geometric. If we can, then C is finite, and we can find it by keeping track of the scaling factors. **Alternating minimization/operator scaling!**

Scaling Brascamp-Lieb

Theorem [5]: There exists a completely positive operator T such that (a) the BL constant is the **capacity** of T and (b) $\mathrm{BL}(\boldsymbol{B}, \boldsymbol{p})$ is finite if T is scalable to "doubly stochastic", i.e. $\sum_i A_i X A_i^\dagger = I$.

T consists of a set of matrices $A_1, \ldots, A_m$. $T(X) = \sum_i A_i X A_i^\dagger$. The capacity is defined as the minimum determinant of $T(X)$ for all psd X with determinant 1.

How do we find capacity of a completely positive operator?

Scale the A_i to LA_iR so that $\sum_i (LA_iR)(LA_iR)^\dagger = I$ and $\sum_i (LA_iR)^\dagger (LA_iR) = I$.

Theorem [4]: This converges in polynomially many iterations.

Theorem [4]: The operator is dimension non-decreasing, ie. $\dim(\sum_{i\in[m]} A_i(V)) \geq \dim(V)$ for all subspaces V of $\mathbb{C}^n$, if it is (singly) normalized and "close" to doubly stochastic.

Connections to Invariant Theory

Left-right action: determinant-1 matrices L, R acting on a tuple of matrices $(A_1, \dots, A_m)$, output is $(LA_1R, \dots, LA_mR)$.

Hilbert-Mumford for the left-right action: $(A_1, \dots, A_m)$ is not in the **null cone** (all invariant polynomials vanish) iff it is dimension non-decreasing. [6]

Kempf-Ness: $A = (A_1, \dots, A_m)$ is not in the null cone iff A is scalable.

Efficient way to test membership in the BL polytope! [7]: also exists an efficient algorithm to give the witness for a "bad V" if the operator is dimension-decreasing.

Runtime: poly in bit complexity and **the common denominator** of the p_j. Exponential time to optimize, double-exponential to enumerate BL constraints.

🤔 Can we do better?

Embedding Computations in BL Polytopes

Given a ground set of vectors $v_1, \ldots, v_m$ in $\mathbb{R}^n$, the linear matroid is the set of indices $I \subseteq [m]$ such that the vectors $\{v_i : i \in I\}$ are linearly independent.

Suppose we have two matroids $\mathcal{M}_v$ and $\mathcal{M}_w$, over m-sized ground sets $\{v_i\}$ and $\{w_i\}$ respectively. The matroid intersection polytope is the convex hull of the set of dimension-m 0-1 indicator vectors, where the nonzero indices of each vector represent the indices i that give subsets of $\{v_i\}$ and $\{w_i\}$ that are **both** bases for $\mathbb{R}^n$.

Theorem (Folklore): the matroid intersection polytope can be rewritten as:

$$\sum_{j \in [m]} p_j = n \qquad p_i \geq 0 \qquad \forall i$$

$$\sum_{j \in J} p_j \leq \dim(V_J) \quad \text{and} \quad \sum_{j \in J} p_j \leq \dim(W_J) \qquad \forall J \subseteq [m]$$

where V_J is the span of $\{v_j : j \in J\}$, likewise for W_J.

Embedding Computations in BL Polytopes i

Theorem (Folklore): the matroid intersection polytope can be rewritten as:

$$\sum_{j\in[m]} p_j = n \qquad p_i \geq 0 \qquad \forall i$$

$$\sum_{j\in J} p_j \leq \dim(V_j) \quad \text{and} \quad \sum_{j\in J} p_j \leq \dim(W_j) \qquad \forall J \subseteq [m]$$

where V_j is the span of $\{v_j : j \in J\}$, likewise for W_j.

Theorem [5]: This polytope corresponds to the BL polytope for

$$B_i = \begin{pmatrix} 0 & v_i^T \\ w_i^T & 0 \end{pmatrix} .$$

Embedding Computations in BL Polytopes ii

Notice that bipartite matching can be represented as the intersection of two partition matroids, one for each side of the graph, and therefore can be embedded into BL.

🤔 Can general matching be embedded in BL? Does this shed any light on complex polytopes that can be optimized over?

Barthe, Franck (Oct. 1998). "On a reverse form of the Brascamp-Lieb inequality". In: *Inventiones mathematicae* 134.2, pp. 335–361. ISSN: 1432-1297. DOI: `10.1007/s002220050267`. URL: `https://doi.org/10.1007/s002220050267`.

Bennett, J., A. Carbery, M. Christ, and T. Tao (2010). "Finite bounds for Hölder-Brascamp-Lieb multilinear inequalities". In: *Math. Res. Lett.* 17.4, pp. 647–666.

Embedding Computations in BL Polytopes iii

Christ, Michael et al. (May 2013). *Communication Lower Bounds and Optimal Algorithms for Programs That Reference Arrays - Part 1.* Tech. rep. UCB/EECS-2013-61. EECS Department, University of California, Berkeley. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-61.html`.

Garg, Ankit, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson (2015). "A deterministic polynomial time algorithm for non-commutative rational identity testing". In: *CoRR* abs/1511.03730. arXiv: `1511.03730`. URL: `http://arxiv.org/abs/1511.03730`.

— (2016). "Algorithmic aspects of Brascamp-Lieb inequalities". In: *CoRR* abs/1607.06711. arXiv: `1607.06711`. URL: `http://arxiv.org/abs/1607.06711`.

Embedding Computations in BL Polytopes iv

Garg, Ankit and Rafael Mendes de Oliveira (2018). "Recent progress on scaling algorithms and applications". In: *CoRR* abs/1808.09669. arXiv: 1808.09669. URL: http://arxiv.org/abs/1808.09669.

Ivanyos, Gábor, Youming Qiao, and K. V. Subrahmanyam (2015). "Non-commutative Edmonds' problem and matrix semi-invariants". In: *CoRR* abs/1508.00690. arXiv: 1508.00690. URL: http://arxiv.org/abs/1508.00690.

Lieb, Elliott H. (Dec. 1990). "Gaussian kernels have only Gaussian maximizers". In: *Inventiones mathematicae* 102.1, pp. 179–208. ISSN: 1432-1297. DOI: 10.1007/BF01233426. URL: https://doi.org/10.1007/BF01233426.