Lecture: Systolic Arrays I

- Topics: sorting and matrix algorithms
Dense Computation

• Distribute memory across multiple chips; sufficient on-chip wiring to feed computational units

• How do we design the compute units?
  • GPU (too general-purpose)
  • DaDianNao’s NFU (custom SIMD)
  • Eyeriss’ spatial architecture (basic tile, operand network)
  • ISAAC (analog)

• Systolic arrays: dense compute units; data flows through these units with low rd/wr costs; loose connection with the brain; effective for image processing, pattern recog, etc.
Sorting on a Linear Array

- Each processor has bidirectional links to its neighbors
- All processors share a single clock (asynchronous designs will require minor modifications)
- At each clock, processors receive inputs from neighbors, perform computations, generate output for neighbors, and update local storage
Control at Each Processor

• Each processor stores the minimum number it has seen

• Initial value in storage and on network is “∗”, which is bigger than any input and also means “no signal”

• On receiving number Y from left neighbor, the processor keeps the smaller of Y and current storage Z, and passes the larger to the right neighbor
Sorting Example
Result Output

• The output process begins when a processor receives a non-*, followed by a “*”

• Each processor forwards its storage to its left neighbor and subsequent data it receives from right neighbors

• How many steps does it take to sort N numbers?

• What is the speedup and efficiency?
Output Example
The bit model affords a more precise measure of complexity – we will now assume that each processor can only operate on a bit at a time.

To compare $N \times k$-bit words, you may now need an $N \times k$ 2-d array of bit processors.
Pipelined Comparison

Input numbers:     3  4  2
                 0  1  0
                 1  0  1
                 1  0  0
Comparison Strategies

- **Strategy 1**: Bits travel horizontally, keep/swap signals travel vertically; if inputs arrive from the left, the array is sorted in $2N + k$ steps.

- **Strategy 2**: Use a tree to communicate information on which number is greater – can set up a pipeline so the sorting happens in $2N + \log k$ steps.
Lower Bounds

• Input/Output bandwidth: $Nk$ bits are being input/output with $k$ pins – requires $\Omega(N)$ time

• Diameter: the comparison at processor $(1,1)$ influences the value of the bit stored at processor $(N,k)$ – for example, $N-1$ numbers are 011..1 and the last number is either 00…0 or 10…0 – it takes at least $N+k-2$ steps for information to travel across the diameter

• Bisection width: if processors in one half require the results computed by the other half, the bisection bandwidth imposes a minimum completion time
Counter Example

- N 1-bit numbers that need to be sorted with a binary tree
- Since bisection bandwidth is 2 and each number may be in the wrong half, will any algorithm take at least N/2 steps?
Counting Algorithm

- It takes \( O(\log N) \) time for each intermediate node to add the contents in the subtree and forward the result to the parent, one bit at a time.

- After the root has computed the number of 1’s, this number is communicated to the leaves – the leaves accordingly set their output to 0 or 1.

- Each half only needs to know the number of 1’s in the other half (\( \log N - 1 \) bits) – therefore, the algorithm takes \( \Omega(\log N) \) time.

- Careful when estimating lower bounds!
Matrix Algorithms

• Consider matrix-vector multiplication:

\[ y_i = \sum_j a_{ij} x_j \]

• The sequential algorithm takes \(2N^2 - N\) operations

• With an N-cell linear array, can we implement matrix-vector multiplication in \(O(N)\) time?
Matrix Vector Multiplication

Number of steps = 2N – 1
Matrix-Matrix Multiplication

Number of time steps = 3N – 2
Complexity

• The algorithm implementations on the linear arrays have speedups that are linear in the number of processors – an efficiency of $O(1)$

• It is possible to improve these algorithms by a constant factor, for example, by inputting values directly to each processor in the first step and providing wraparound edges ($N$ time steps)
Dataflow for Convolution

For a 3x3 kernel with strides of 1, every input pixel is involved in 9 ops

Pixels 7 4 1

Pixels 8 5 2

Pixels 9 6 3

This will produce partial results that have to be consumed later.
Comparison with Eyeriss Convolution
References

• “Introduction to Parallel Algorithms and Architectures,” Leighton
• Figure credits: Mitsu Ogihara
Title

• Bullet