A CODELESS INTRODUCTION TO PARALLELLIZATION

Will Landau, Matt Simpson, Prof. Jarad Niemi

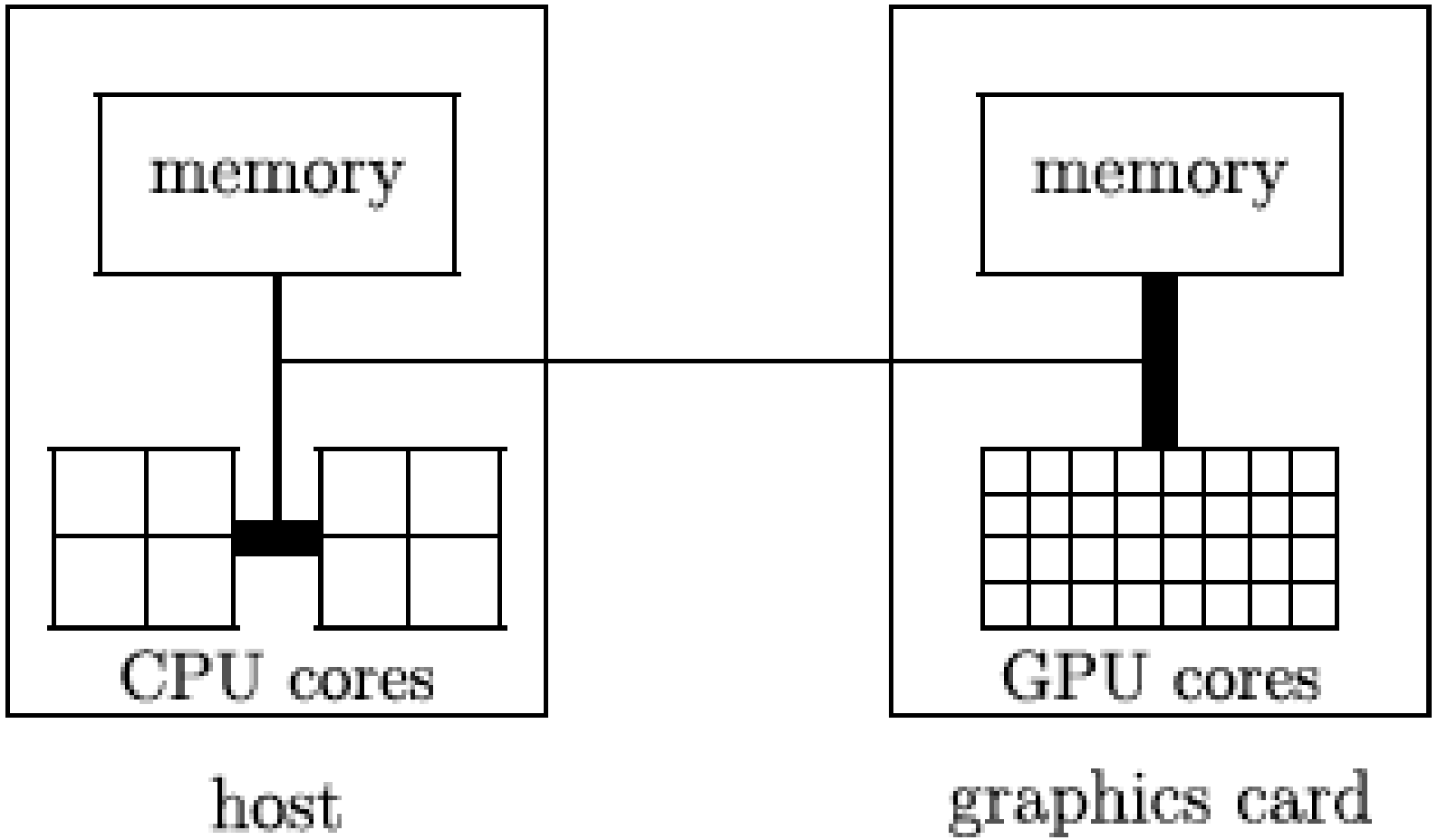
HOW THE CPU AND GPU WORK TOGETHER

A GPU can't run a whole computer on its own because it can't do control flow and it doesn't have access to all the system hardware.

In a GPU-capable computer, the CPU is the main processor, and the GPU is an optional hardware add-on.

The CPU is the "master" of the computer, and it can delegate its highest-throughput parallelizable arithmetic load to the GPU "minion".

Another analogy: the CPU uses the GPU in the same way that a human uses a hand-held calculator.


memory
memory
CPU cores
GPU cores
host
graphics card

GPUS AND PARALLELIZATION

Parallelization: Running different calculations simultaneously. It speeds up calculations dramatically, and GPUS are much better at it than CPUs.

Kernel: An instruction set executed on the GPU. (All others are executed on the CPU.)

In CUDA C, a kernel is any function prefixed with the keyword, `__global__`. (More on that in a later talk.)

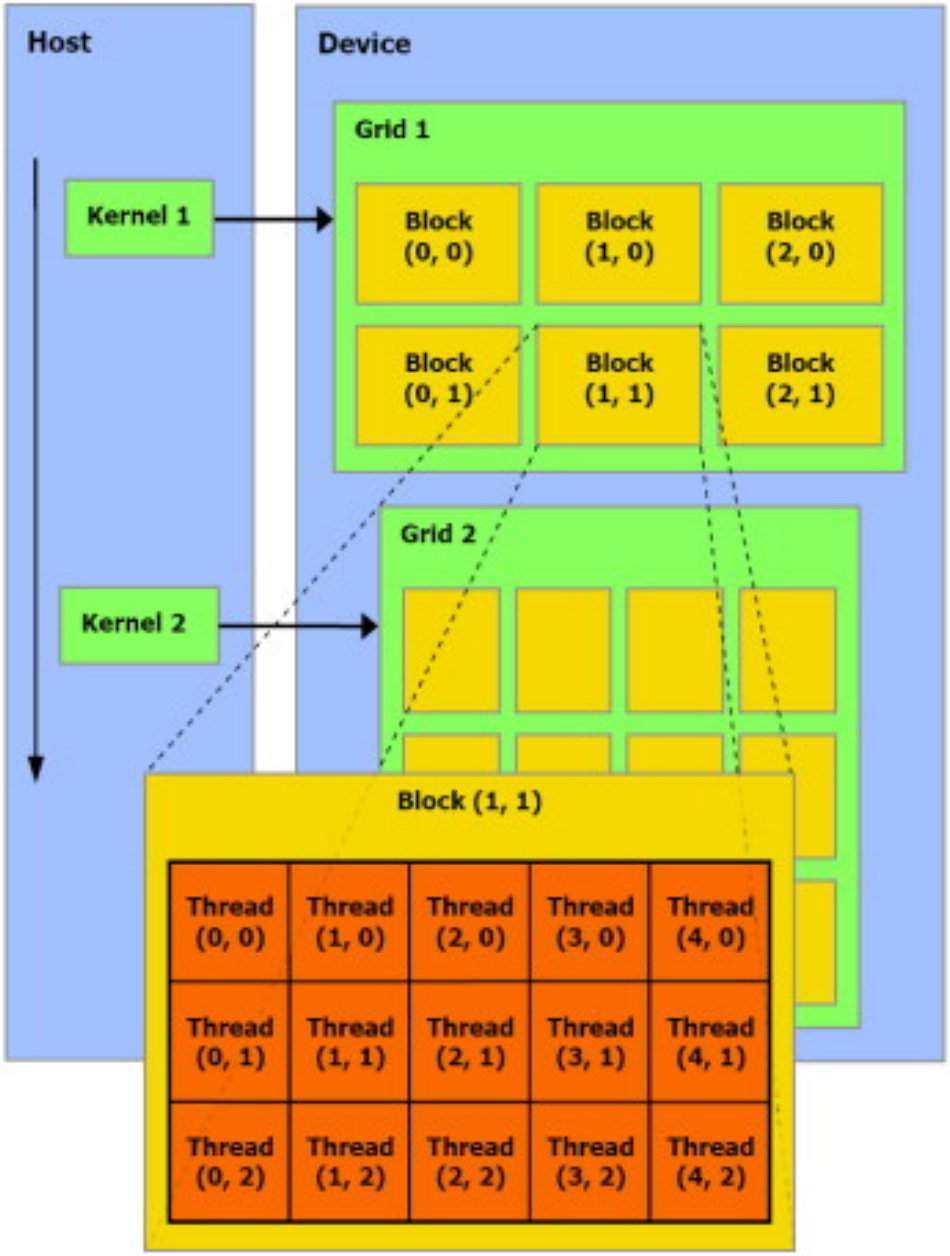
IMPLEMENTING PARALLELIZATION ON THE GPU IN PRACTICE

1. The CPU sends a kernel (instruction set) to the GPU.
2. For every time the CPU sends a kernel, the GPU executes the kernel multiple times simultaneously (in **PARALLEL**). Each such execution of the kernel is called a **thread**.

ORGANIZATION OF THREADS

Grid; The collection of all the threads that are spawned when the CPU sends a kernel to the GPU.

Block: A collection of threads within a grid that share memory quickly and easily.


Host
Device
Grid 1
Kernel 1
Block (0, 0)
Block (1, 0)
Block (2, 0)
Block (0, 1)
Block (1, 1)
Block (2, 1)
Grid 2
Kernel 2
Block (1, 1)
Thread (0, 0)
Thread (1, 0)
Thread (2, 0)
Thread (3, 0)
Thread (4, 0)
Thread (0, 1)
Thread (1, 1)
Thread (2, 1)
Thread (3, 1)
Thread (4, 1)
Thread (0, 2)
Thread (1, 2)
Thread (2, 2)
Thread (3, 2)
Thread (4, 2)

IMPORTANT REMARKS:

- With one grid per kernel, GRIDS are executed SEQUENTIALLY.
- Blocks within the same grid are executed SIMULTANEOUSLY.
- Threads within the same grid are executed SIMULTANEOUSLY, whether they share a block or not.

NOTE: PARALLELIZATION HAS TWO EQUIVALENT DEFINITIONS

1. Running different calculations simultaneously.
2. Breaking up a calculation into grids, then into blocks, and then into threads.

When I say "parallelization" in practice, I will most likely be referring to definition 2.

WHEN TO PARALLELIZE

Calculations you want to parallelize:

- Repeated floating point arithmetic procedures that can all be done simultaneously.
- Anything that can be broken down into or framed as such.

Calculations you don't want to parallelize:

- Inherently sequential calculations, such as recursions.
- Control flow: if-then statements, etc.
- CPU system routines, such as printing to the console.

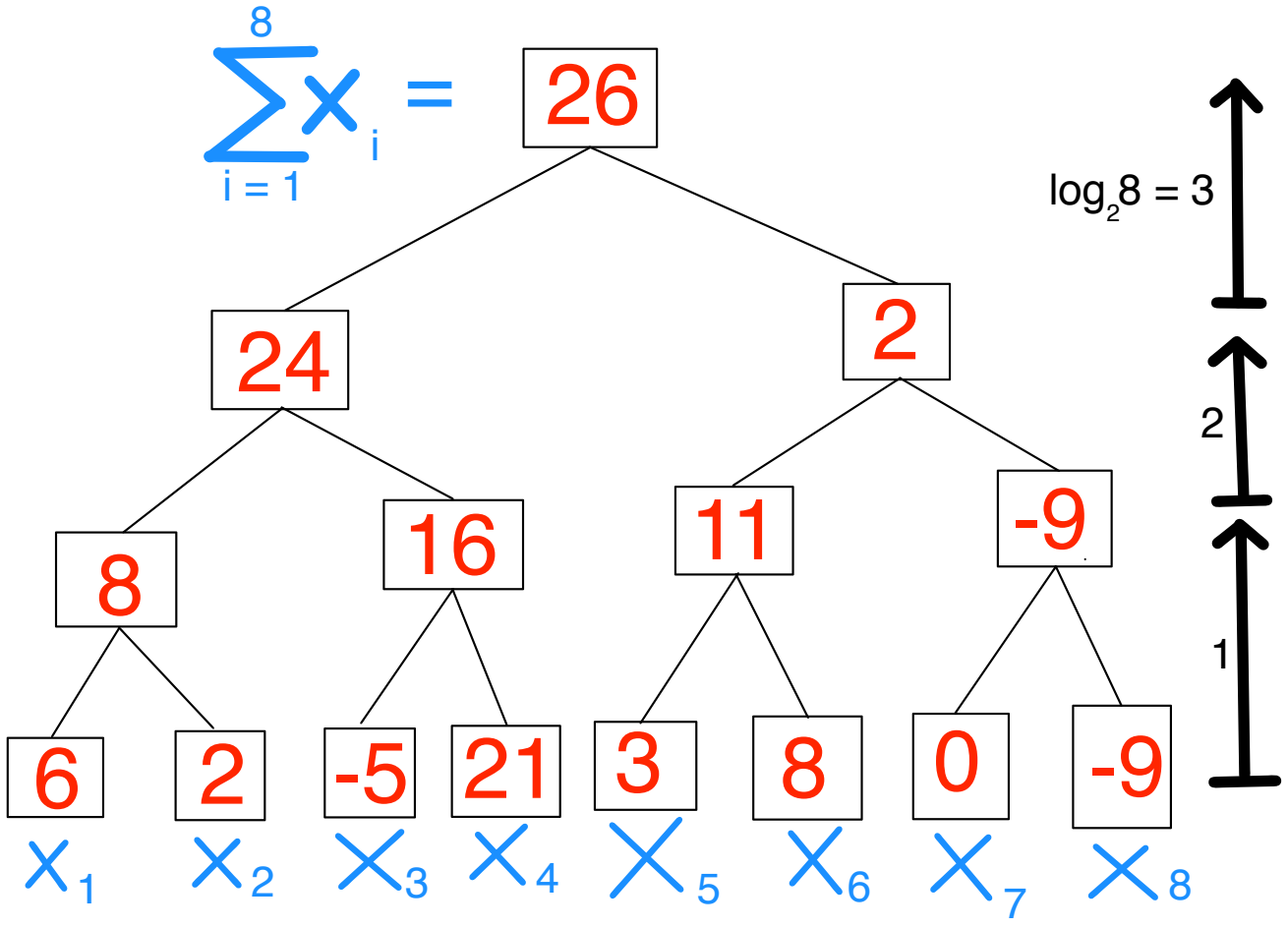
EXAMPLES OF EASILY PARALLELIZABLE ALGORITHMS

Linear algebraic algorithms are particularly amenable to GPU computing because they involve a high volume of simple arithmetic.

I will showcase:

1. the pairwise c(ascading) sum
2. matrix multiplication
3. the QR factorization

1. THE PAIRWISE (CASCADING) SUM



A RIGOROUS DESCRIPTION

Suppose you have a vector $X_0 = (x_{(0,1)}, x_{(0,2)}, \ldots, x_{(0,n)})$, where $n = 2^m$ for some $m > 0$.

Compute $\sum_{i=1}^{n} x_{(0,i)}$ in the following way:

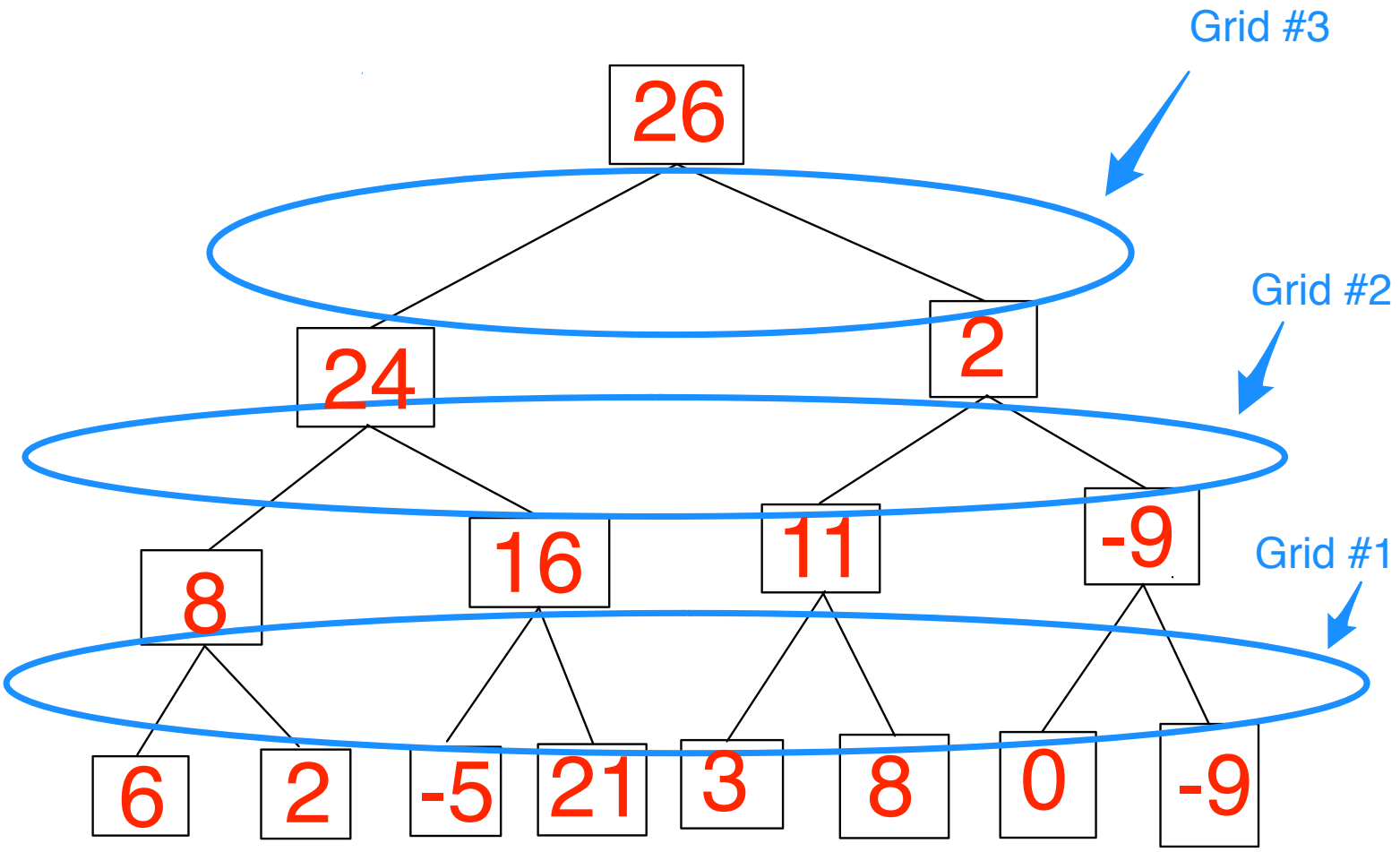
1. Create a new vector:
   $X_1 = (\underbrace{x_{(0,1)} + x_{(0,2)}}_{x_{(1,1)}}, \underbrace{x_{(0,3)} + x_{(0,4)}}_{x_{(1,2)}}, \ldots, \underbrace{x_{(0,n-1)} + x_{(0,n)}}_{x_{(1,n/2)}})$
2. Create another new vector:
   $X_2 = (\underbrace{x_{(1,1)} + x_{(1,2)}}_{x_{(2,1)}}, \underbrace{x_{(1,3)} + x_{(1,4)}}_{x_{(2,2)}}, \ldots, \underbrace{x_{(1,n/2-1)} + x_{(1,n/2)}}_{x_{(2,n/4)}})$
3. Continue this process until you get a singleton vector:
   $X_{\log_2(n)} = (\underbrace{x_{(\log_2(n)-1,1)}, x_{(\log_2(n)-1,2)}}_{x_{(\log_2(n),1)}})$

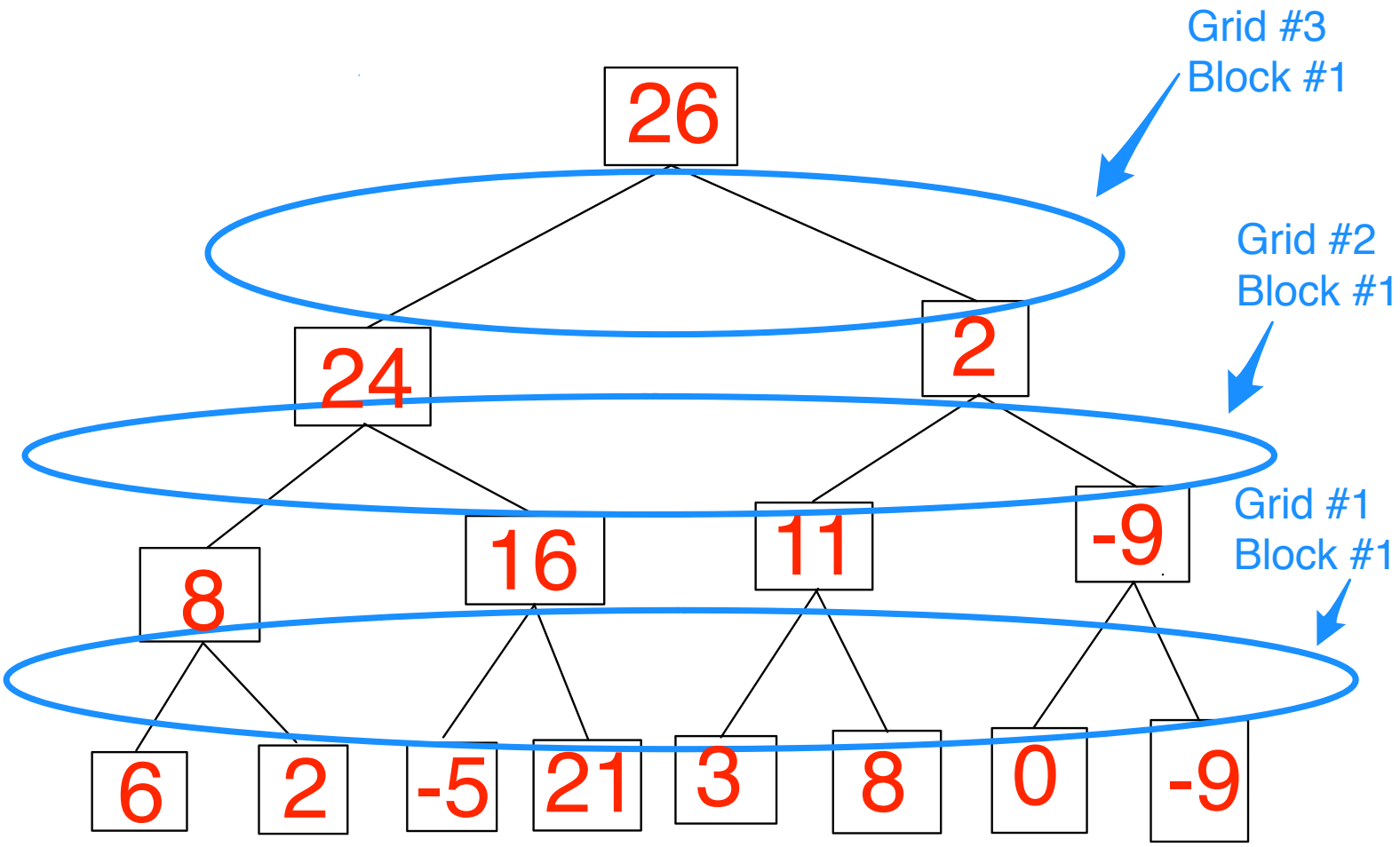
Notice: $\sum_{i=1}^{n} x_{(0,i)} = x_{(\log_2(n),1)}$

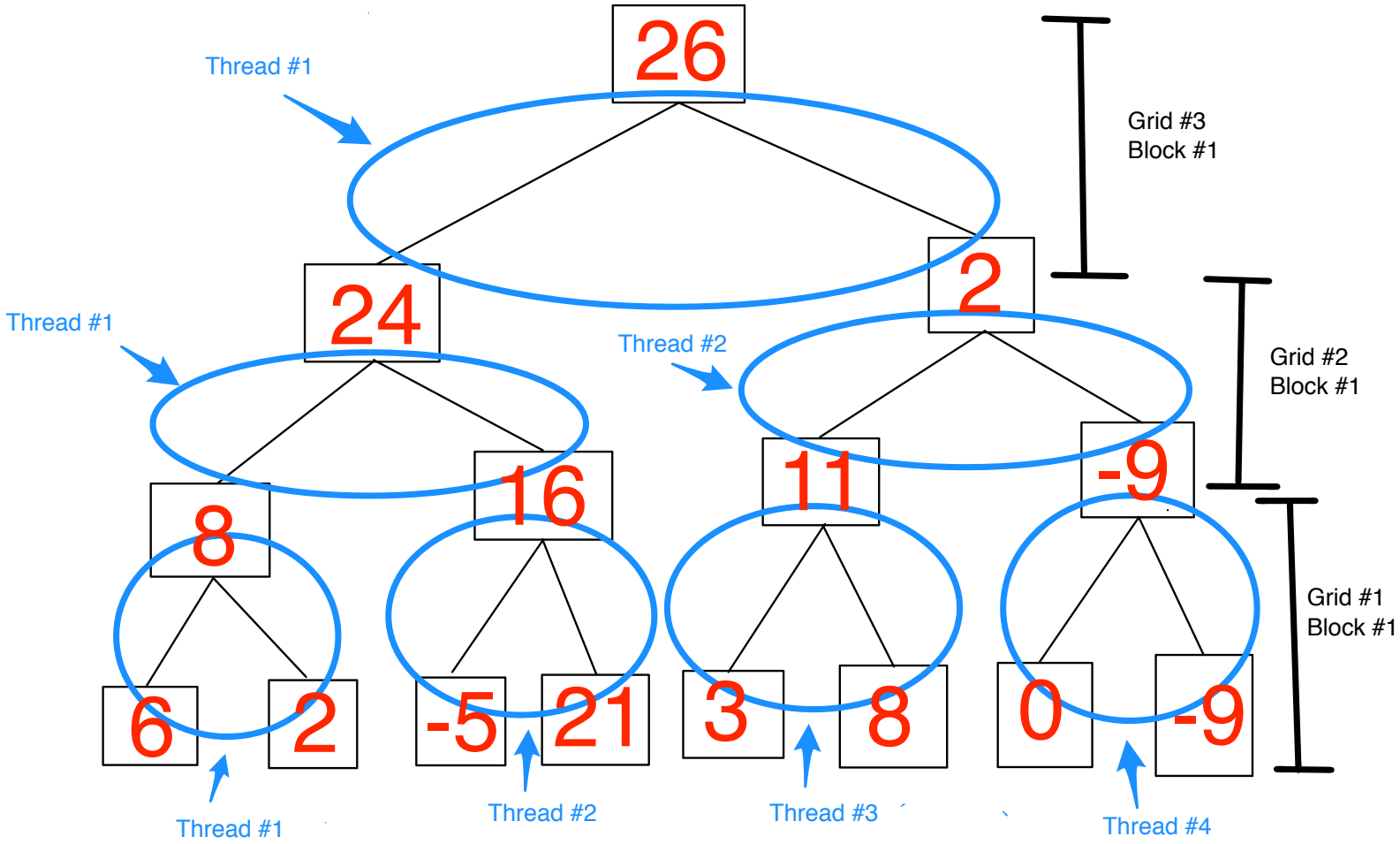
PARALLELIZING THE PAIRWISE SUM

Create $\log_2(n)$ grids, each to compute $X_1, X_2, \ldots, X_{\log_2(n)}$, respectively, in sequence. For the i'th grid:

1. Spawn one block of $n/2^i$ threads.
2. Let the j'th thread compute the j'th element of X_i by pairwise summing the appropriate two elements of X_{i-1}.


Grid #3
26
Grid #2
24
2
Grid #1
8
16
11
-9
6
2
-5
21
3
8
0
-9


Grid #3
Block #1
Grid #2
Block #1
Grid #1
Block #1
26
24
2
8
16
11
-9
6
2
-5
21
3
8
0
-9


26
Thread #1
Grid #3
Block #1
24
2
Thread #1
Thread #2
Grid #2
Block #1
8
16
11
-9
Grid #1
Block #1
6
2
-5
21
3
8
0
-9
Thread #1
Thread #2
Thread #3
Thread #4

2. MATRIX MULTIPLICATION

Consider an $m \times n$ matrices, $A = (a_{ij})$, and an $n \times p$ matrix, $B = (b_{ij})$. Compute $A \cdot B$:

1. Break apart A into its rows: $A = \begin{bmatrix} a_{1\#} \\ a_{2\#} \\ \vdots \\ a_{m\#} \end{bmatrix}$, where each $a_{i\#} = \begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \end{bmatrix}$

2. Break apart B into its columns: $B = \begin{bmatrix} b_{\#1} & b_{\#2} & \cdots & b_{\#p} \end{bmatrix}$, where each $b_{\#j} = \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix}$

3. Compute $A \cdot B$ elementwise, using the usual matrix multiplication rules to find each $a_i \cdot b_j$:

$$A \cdot B = \begin{bmatrix} (a_{1\#} \cdot b_{\#1}) & (a_{1\#} \cdot b_{\#2}) & \cdots & (a_{1\#} \cdot b_{\#p}) \\ (a_{2\#} \cdot b_{\#1}) & (a_{2\#} \cdot b_{\#2}) & & (a_{2\#} \cdot b_{\#p}) \\ \vdots & & \ddots & \vdots \\ (a_{m\#} \cdot b_{\#1}) & (a_{m\#} \cdot b_{\#2}) & \cdots & (a_{m\#} \cdot b_{\#p}) \end{bmatrix}$$

PARALLELIZING MATRIX MULTIPLICATION

One approach is to use two sequential grids:

1. Grid 1: spawn $m \cdot p$ blocks. The (i, j)'th block does the following:
   a. Spawn n threads.
   b. Tell the k'th thread to compute $c_{ink} = a_{ik}b_{kj}$.
2. Grid 2: spawn $m \cdot p$ blocks. The (i, j)'th block does the following:
   a. Compute $(A \cdot B)_{(i,j)} = \sum_{k=1}^{n} c_{ijk}$ as a pairwise sum.

EXAMPLE

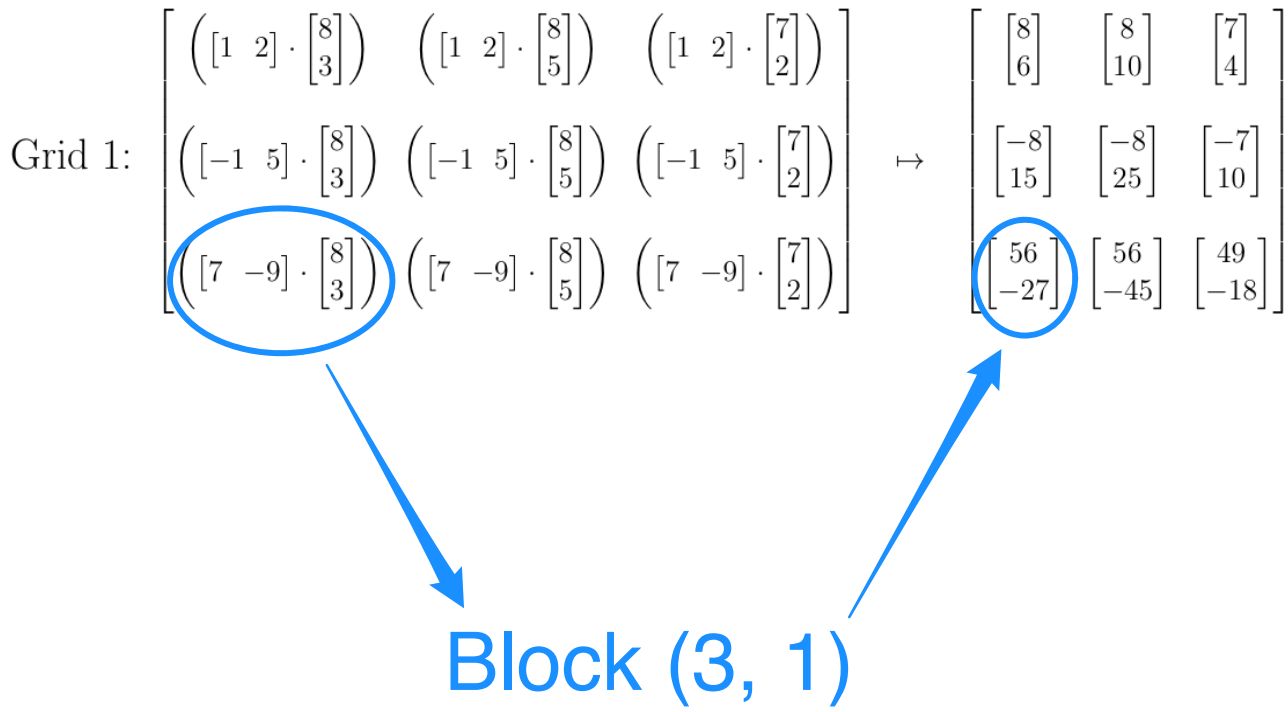
Say I want to compute $A \cdot B$, where:

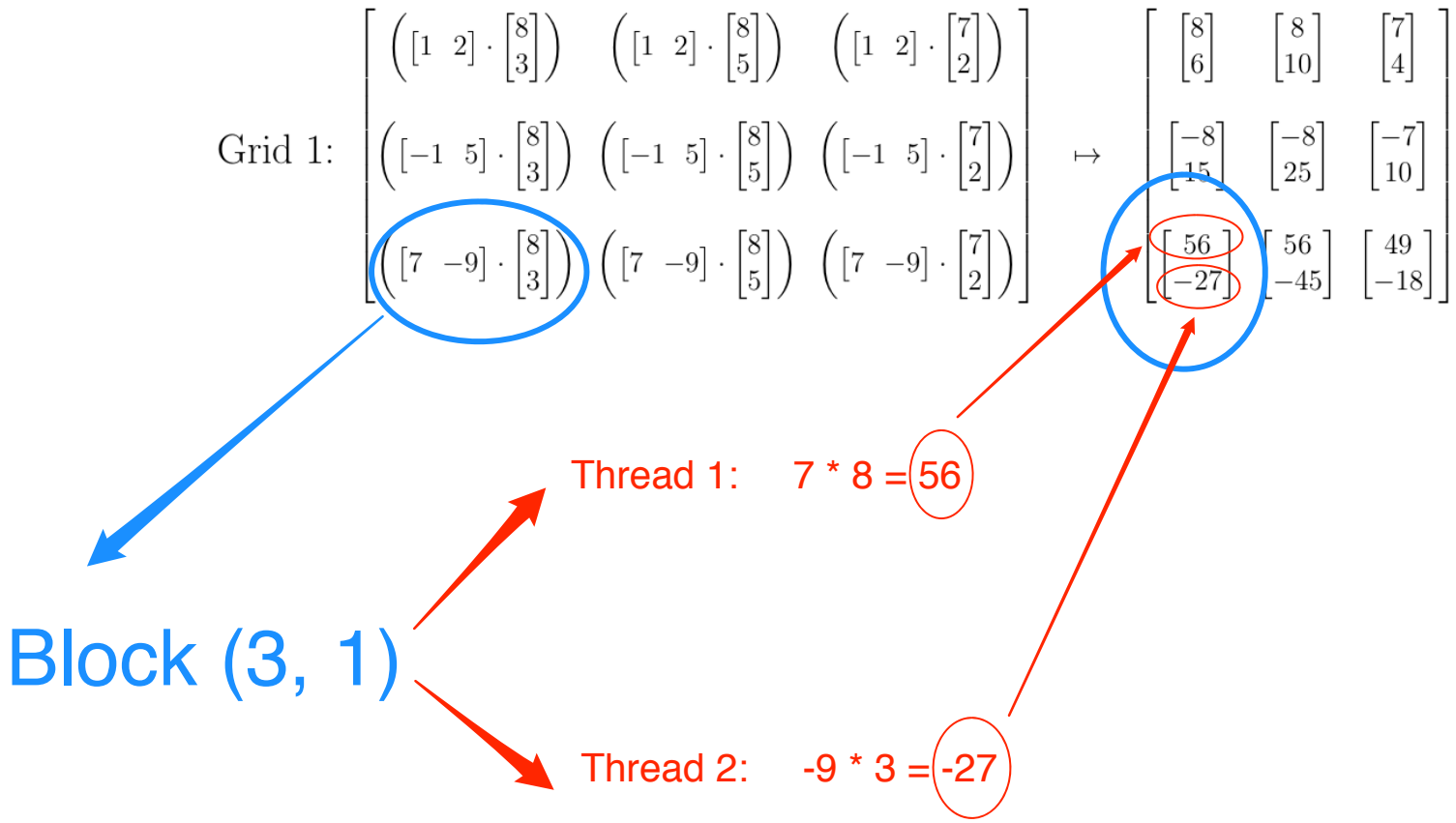
$$A = \begin{bmatrix} 1 & 2 \\ -1 & 5 \\ 7 & -9 \end{bmatrix} \quad B = \begin{bmatrix} 8 & 8 & 7 \\ 3 & 5 & 2 \end{bmatrix}$$

which I'm setting up as:

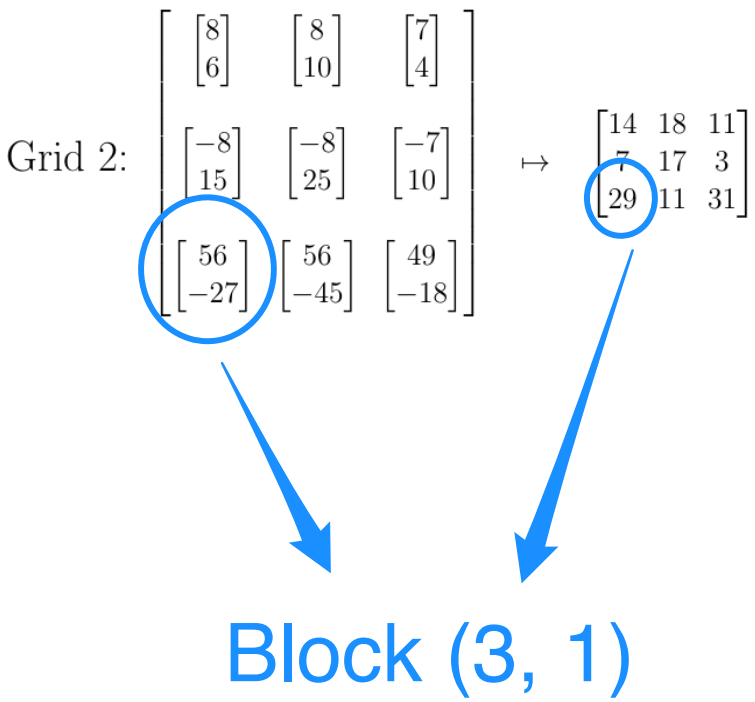
$$A \cdot B = \begin{bmatrix} \left([1 \ 2] \cdot \begin{bmatrix} 8 \\ 3 \end{bmatrix} \right) & \left([1 \ 2] \cdot \begin{bmatrix} 8 \\ 5 \end{bmatrix} \right) & \left([1 \ 2] \cdot \begin{bmatrix} 7 \\ 2 \end{bmatrix} \right) \\ \left([-1 \ 5] \cdot \begin{bmatrix} 8 \\ 3 \end{bmatrix} \right) & \left([-1 \ 5] \cdot \begin{bmatrix} 8 \\ 5 \end{bmatrix} \right) & \left([-1 \ 5] \cdot \begin{bmatrix} 7 \\ 2 \end{bmatrix} \right) \\ \left([7 \ -9] \cdot \begin{bmatrix} 8 \\ 3 \end{bmatrix} \right) & \left([7 \ -9] \cdot \begin{bmatrix} 8 \\ 5 \end{bmatrix} \right) & \left([7 \ -9] \cdot \begin{bmatrix} 7 \\ 2 \end{bmatrix} \right) \end{bmatrix}$$

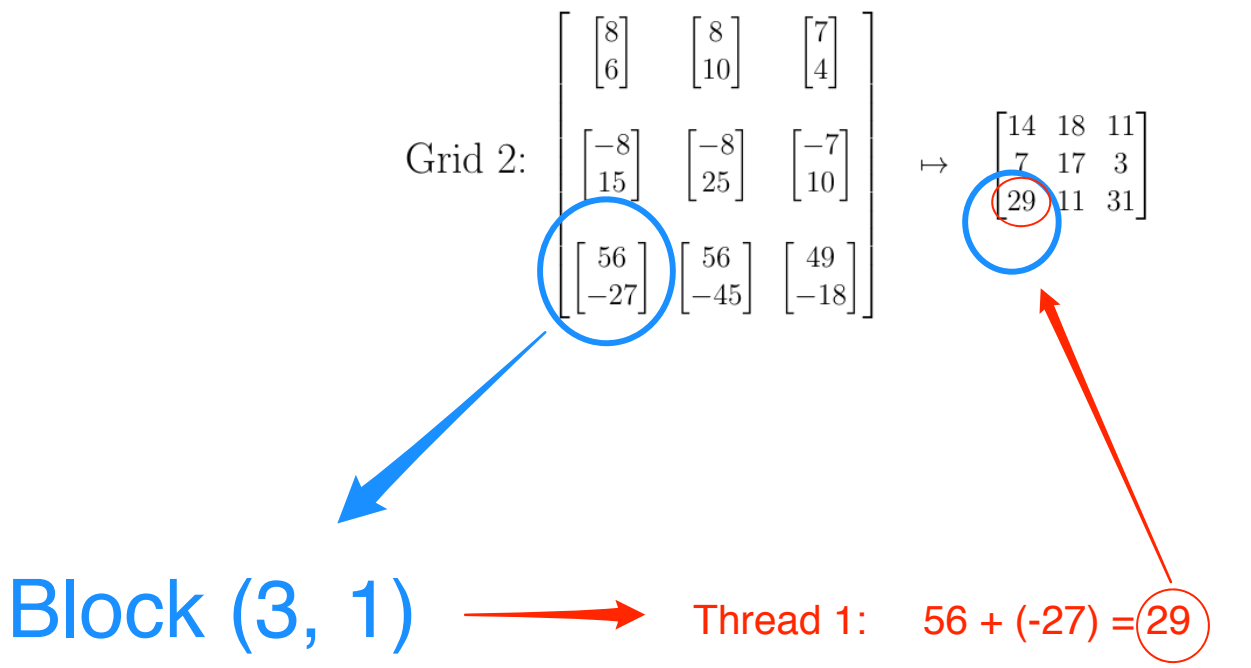
$$\text{Grid 1: } \begin{bmatrix} \left(\begin{bmatrix}1 & 2\end{bmatrix} \cdot \begin{bmatrix}8\\3\end{bmatrix}\right) & \left(\begin{bmatrix}1 & 2\end{bmatrix} \cdot \begin{bmatrix}8\\5\end{bmatrix}\right) & \left(\begin{bmatrix}1 & 2\end{bmatrix} \cdot \begin{bmatrix}7\\2\end{bmatrix}\right) \\ \left(\begin{bmatrix}-1 & 5\end{bmatrix} \cdot \begin{bmatrix}8\\3\end{bmatrix}\right) & \left(\begin{bmatrix}-1 & 5\end{bmatrix} \cdot \begin{bmatrix}8\\5\end{bmatrix}\right) & \left(\begin{bmatrix}-1 & 5\end{bmatrix} \cdot \begin{bmatrix}7\\2\end{bmatrix}\right) \\ \left(\begin{bmatrix}7 & -9\end{bmatrix} \cdot \begin{bmatrix}8\\3\end{bmatrix}\right) & \left(\begin{bmatrix}7 & -9\end{bmatrix} \cdot \begin{bmatrix}8\\5\end{bmatrix}\right) & \left(\begin{bmatrix}7 & -9\end{bmatrix} \cdot \begin{bmatrix}7\\2\end{bmatrix}\right) \end{bmatrix} \quad \mapsto \quad \begin{bmatrix} \begin{bmatrix}8\\6\end{bmatrix} & \begin{bmatrix}8\\10\end{bmatrix} & \begin{bmatrix}7\\4\end{bmatrix} \\ \begin{bmatrix}-8\\15\end{bmatrix} & \begin{bmatrix}-8\\25\end{bmatrix} & \begin{bmatrix}-7\\10\end{bmatrix} \\ \begin{bmatrix}56\\-27\end{bmatrix} & \begin{bmatrix}56\\-45\end{bmatrix} & \begin{bmatrix}49\\-18\end{bmatrix} \end{bmatrix}$$


Grid 1:
Block (3, 1)


Grid 1:
Block (3, 1)
Thread 1: 7 * 8 = 56
Thread 2: -9 * 3 = -27

$$\text{Grid 2:} \quad \begin{bmatrix} \begin{bmatrix} 8 \\ 6 \end{bmatrix} & \begin{bmatrix} 8 \\ 10 \end{bmatrix} & \begin{bmatrix} 7 \\ 4 \end{bmatrix} \\ \begin{bmatrix} -8 \\ 15 \end{bmatrix} & \begin{bmatrix} -8 \\ 25 \end{bmatrix} & \begin{bmatrix} -7 \\ 10 \end{bmatrix} \\ \begin{bmatrix} 56 \\ -27 \end{bmatrix} & \begin{bmatrix} 56 \\ -45 \end{bmatrix} & \begin{bmatrix} 49 \\ -18 \end{bmatrix} \end{bmatrix} \quad \mapsto \quad \begin{bmatrix} 14 & 18 & 11 \\ 7 & 17 & 3 \\ 29 & 11 & 31 \end{bmatrix}$$


Grid 2:
$\begin{bmatrix} \begin{bmatrix}8\\6\end{bmatrix} & \begin{bmatrix}8\\10\end{bmatrix} & \begin{bmatrix}7\\4\end{bmatrix} \\ \begin{bmatrix}-8\\15\end{bmatrix} & \begin{bmatrix}-8\\25\end{bmatrix} & \begin{bmatrix}-7\\10\end{bmatrix} \\ \begin{bmatrix}56\\-27\end{bmatrix} & \begin{bmatrix}56\\-45\end{bmatrix} & \begin{bmatrix}49\\-18\end{bmatrix} \end{bmatrix}$
$\mapsto$
$\begin{bmatrix}14 & 18 & 11\\ 7 & 17 & 3\\ 29 & 11 & 31\end{bmatrix}$
Block (3, 1)


Grid 2:
$\begin{bmatrix} \begin{bmatrix}8\\6\end{bmatrix} & \begin{bmatrix}8\\10\end{bmatrix} & \begin{bmatrix}7\\4\end{bmatrix} \\ \begin{bmatrix}-8\\15\end{bmatrix} & \begin{bmatrix}-8\\25\end{bmatrix} & \begin{bmatrix}-7\\10\end{bmatrix} \\ \begin{bmatrix}56\\-27\end{bmatrix} & \begin{bmatrix}56\\-45\end{bmatrix} & \begin{bmatrix}49\\-18\end{bmatrix} \end{bmatrix}$
$\mapsto$
$\begin{bmatrix}14 & 18 & 11\\ 7 & 17 & 3\\ 29 & 11 & 31\end{bmatrix}$
Block (3, 1)
Thread 1: 56 + (-27) = 29

3. THE QR FACTORIZATION

Theorem: Let A be an $m \times n$ matrix with linearly independent columns. Then:

$$A = QR$$

where:

- Q is an $m \times n$ matrix whose columns form an orthonormal basis for the column space of A.
- R is an $n \times n$ upper triangular (and therefore invertible) matrix with all positive entries on the diagonal.

EXAMPLE

$$A = \begin{bmatrix} 5 & 9 \\ 1 & 7 \\ -3 & -5 \\ 1 & 5 \end{bmatrix}$$

The columns of A are linearly independent. Therefore, A has a QR factorization:

$$A = QR$$

The following choices for Q and R work:

$$Q = \frac{1}{6} \cdot \begin{bmatrix} 5 & -1 \\ 1 & 5 \\ -3 & 1 \\ 1 & 3 \end{bmatrix} \qquad R = \begin{bmatrix} 6 & 12 \\ 0 & 6 \end{bmatrix}$$

FINDING THE QR FACTORIZATION USING ORTHOGONALITY AND THE GRAHAM SCHMIDT PROCESS

Orthogonal: two vectors u and v are orthogonal if $u \bullet v = 0$.

Note: If u and v are two component vectors in $\mathbb{R}^n$, $u \bullet v = 0$ iff they form a right angle. In Euclidian space, orthogonality is the same as perpendicularity.

Here "$\bullet$" denotes the dot product (or inner product) of two component vectors:

If $a = (a_1, a_2, \ldots, a_n)$ and $b = (b_1, b_2, \ldots, b_n)$, then:

$$a \bullet b = (a_1 \cdot b_1,\ a_2 \cdot b_2,\ \ldots,\ a_n \cdot b_n)$$

THE GRAHAM SCHMIDT PROCESS

To find the QR factorization of $A = [a_1, \ldots, a_n]$ (where $a_1, \ldots, a_n$ are linearly independent), you first want to find an orthogonal basis for the column space of A.

That's where the Graham Schmidt process comes in. The Graham Schmidt process is the construction one such basis, $\{v_1, v_2, \ldots, v_n\}$, by:

$$v_1 = a_1$$

$$v_2 = a_2 - \frac{a_2 \bullet v_1}{v_1 \bullet v_1} \cdot v_1$$

$$v_3 = a_3 - \frac{a_3 \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_3 \bullet v_2}{v_2 \bullet v_2} \cdot v_2$$

$$\vdots$$

$$v_n = a_n - \frac{a_n \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_n \bullet v_2}{v_2 \bullet v_2} \cdot v_2 - \cdots - -\frac{a_n \bullet v_{n-1}}{v_{n-1} \bullet v_{n-1}} \cdot v_{n-1}$$

Even better, you can get an ortho**normal** basis, $\{u_1,\ u_2, \ldots,\ u_n\}$, by:

$$\{u_1 = \frac{v_1}{v_1 \bullet v_1},\ u_2 = \frac{v_2}{v_2 \bullet v_2},\ \ldots,\ u_n = \frac{v_n}{v_n \bullet v_n}\}$$

REARRANGING THAT LONG LIST OF EQUATIONS...

$$a_1 = (a_1 \bullet u_1) \cdot u_1$$

$$a_2 = (a_2 \bullet u_1) \cdot u_1 + (a_2 \bullet u_2) \cdot u_2$$

$$a_3 = (a_3 \bullet u_1) \cdot u_1 + (a_3 \bullet u_2) \cdot u_2 + (a_3 \bullet u_3) \cdot u_3$$

$$\vdots$$

$$a_n = (a_n \bullet u_1) \cdot u_1 + (a_n \bullet u_2) \cdot u_2 + (a_3 \bullet u_3) \cdot u_3 + \cdots + (a_n \bullet u_n) \cdot u_n$$

We can put it all in matrix form:

$$A = [a_1, \ldots, a_n] \qquad Q = [u_1, \ldots, u_n] \qquad R = \begin{bmatrix} (a_1 \bullet u_1) & (a_2 \bullet u_1) & (a_3 \bullet u_1) & \cdots \\ 0 & (a_2 \bullet u_2) & (a_3 \bullet u_2) & \cdots \\ 0 & 0 & (a_3 \bullet u_3) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

And we're all done.

A SUMMARY OF THE WORKFLOW

a. Start with $A = [a_1, \ldots, a_n]$

b. Compute:

$$v_1 = a_1$$

$$v_2 = a_2 - \frac{a_2 \bullet v_1}{v_1 \bullet v_1} \cdot v_1$$

$$v_3 = a_3 - \frac{a_3 \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_3 \bullet v_2}{v_2 \bullet v_2} \cdot v_2$$

$$\vdots$$

$$v_n = a_n - \frac{a_n \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_n \bullet v_2}{v_2 \bullet v_2} \cdot v_2 - \cdots - -\frac{a_n \bullet v_{n-1}}{v_{n-1} \bullet v_{n-1}} \cdot v_{n-1}$$

c. Compute: $\{u_1 = \frac{v_1}{v_1 \bullet v_1},\ u_2 = \frac{v_2}{v_2 \bullet v_2},\ \ldots,\ u_n = \frac{v_n}{v_n \bullet v_n}\}$

d. Compute: $R = \begin{bmatrix} (a_1 \bullet u_1) & (a_2 \bullet u_1) & (a_3 \bullet u_1) & \cdots \\ 0 & (a_2 \bullet u_2) & (a_3 \bullet u_2) & \cdots \\ 0 & 0 & (a_3 \bullet u_3) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$

ONE WAY TO PARALLELIZE b

$$v_1 = a_1$$

$$v_2 = a_2 - \frac{a_2 \bullet v_1}{v_1 \bullet v_1} \cdot v_1$$

$$v_3 = a_3 - \frac{a_3 \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_3 \bullet v_2}{v_2 \bullet v_2} \cdot v_2$$

$$\vdots$$

$$v_n = a_n - \frac{a_n \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_n \bullet v_2}{v_2 \bullet v_2} \cdot v_2 - \cdots - \frac{a_n \bullet v_{n-1}}{v_{n-1} \bullet v_{n-1}} \cdot v_{n-1}$$

Give each $v_i = a_i - \frac{a_i \bullet v_1}{v_1 \bullet v_1} \cdot v_1 - \frac{a_i \bullet v_2}{v_2 \bullet v_2} \cdot v_2 - \cdots - \frac{a_i \bullet v_{i-1}}{v_{i-1} \bullet v_{i-1}} \cdot v_{i-1}$ three SEQUENTIAL grids:

Gr1d 1: Create a block of threads to calculate each of $v_{i-1} \bullet v_{i-i}$ and $a_i \bullet v_j$ for all $j < i$. The other dot products have already been calculated from the step that calculates v_{i-1}.

Gr2d 2: Create one block of threads, where the i'th thread computes $-\frac{a_i \bullet v_{i-1}}{v_{i-1} \bullet v_{i-1}} \cdot v_{i-1}$.

Gr3d 3: Create one final block of threads that performs a cascading sum on the terms, a_i, $-\frac{a_i \bullet v_1}{v_1 \bullet v_1} \cdot v_1$, $-\frac{a_i \bullet v_2}{v_2 \bullet v_2} \cdot v_2$, $-\cdots$, $-\frac{a_i \bullet v_{i-1}}{v_{i-1} \bullet v_{i-1}} \cdot v_{i-1}$

ONE WAY TO PARALLELIZE c

$$\{u_1 = \frac{v_1}{v_1 \bullet v_1},\ u_2 = \frac{v_2}{v_2 \bullet v_2},\ \ldots,\ u_n = \frac{v_n}{v_n \bullet v_n}\}$$

Spawn a grid containing one block of threads, where the i'th thread computes $\frac{v_i}{v_i \bullet v_i}$. All the dot products, $v_i \bullet v_i$, should be available from step b.

ONE WAY TO PARALLELIZE d

$$R = \begin{bmatrix} (a_1 \bullet u_1) & (a_2 \bullet u_1) & (a_3 \bullet u_1) & \cdots \\ 0 & (a_2 \bullet u_2) & (a_3 \bullet u_2) & \cdots \\ 0 & 0 & (a_3 \bullet u_3) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Spawn a block of threads, where the (i, j)'th thread computes $(a_1 \bullet u_1) = \frac{1}{v_1 \bullet v_1}(a_1 \bullet v_1)$ for each $j < i < n$. Each $v_i \bullet v_i$ should be available from step b.

REFERENCES

Lay, David C. *L*inear Algebra and Its Applications. 3rd Ed. Addison Wesley, 2006.

J. Sanders and E. Kandrot. *CUDA by Example*. Addison-Wesley, 2010.