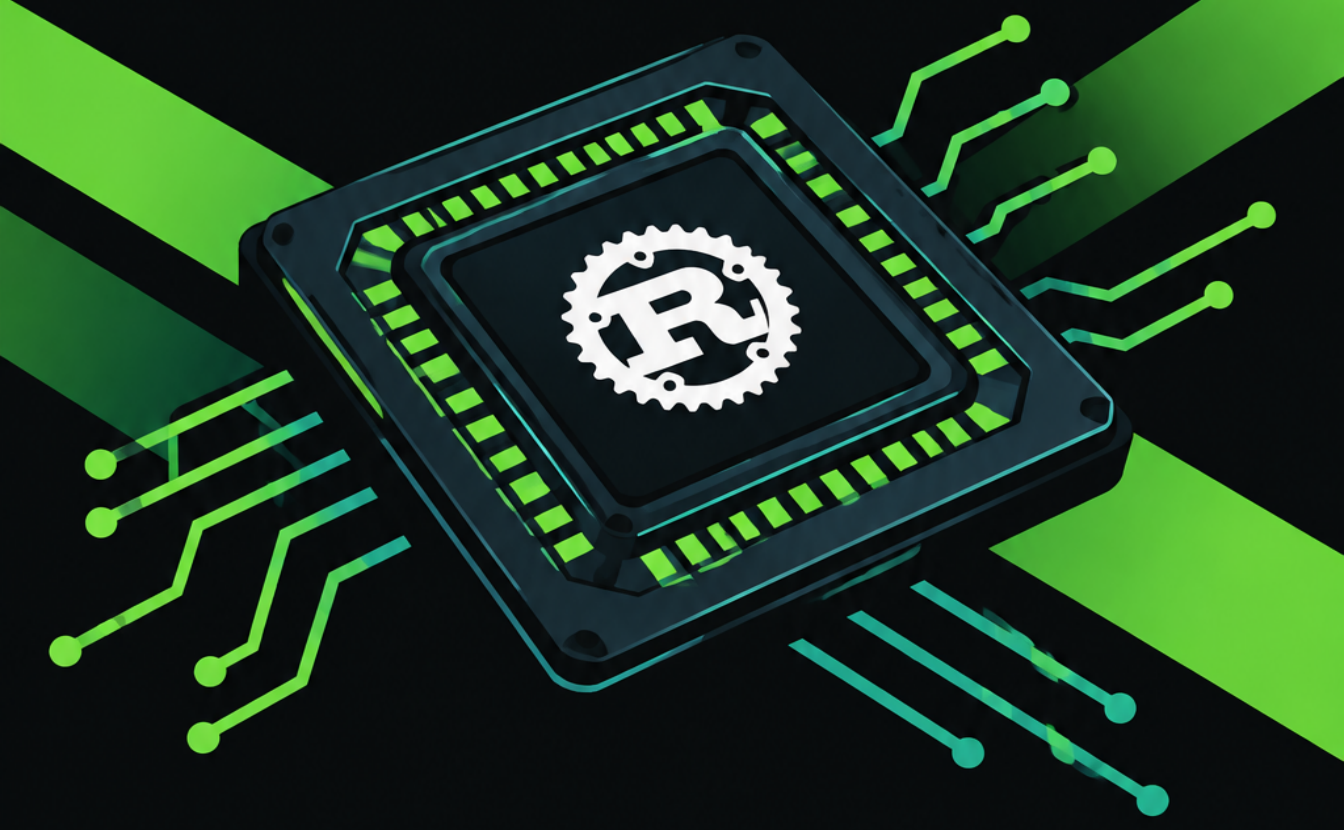
WRITING GPU KERNELS WITH CUDA RUST

A Practical Guide to High-Performance GPU Computing from Rust



MATTHEW STERLING

Writing GPU Kernels with CUDA Rust

A Practical Guide to High-Performance GPU Computing from Rust

Steve Publications

This book is available at
https://leanpub.com/writinggpukernelswithcudarust

This version was published on 2026-09-17

Contents

A Practical Guide to High-Performance GPU Computing from Rust

GPU parallel computing has fundamentally changed how software achieves performance, but writing kernels has historically demanded deep expertise in low-level hardware details and C++ era programming patterns. This book shows Rust developers how to harness NVIDIA GPUs using CUDA Rust, the emerging ecosystem that brings memory safety, expressive type systems, and modern tooling to GPU programming.

You will learn both of NVIDIA's official CUDA Rust tracks: the SIMT-style cuda-oxide compiler backend for fine-grained thread-level control, and the tile-based cutile-rs framework for safe, high-level kernel authoring. Along the way, you will build a solid understanding of GPU architecture, memory hierarchies, synchronization, and performance optimization. Every chapter includes complete, runnable code examples and explains the why behind each concept, not just the how. This is a practical guide for Rust programmers ready to write production-quality GPU kernels without sacrificing the safety and clarity that make Rust compelling.

Introduction

If you write Rust and you care about performance, you have almost certainly encountered a situation where the CPU simply cannot go fast enough. Maybe it is a data science pipeline churning through terabytes of sensor data. Maybe it is an image processing service that must respond in milliseconds. Maybe it is a machine learning model whose training or inference time is your bottleneck. In all of these cases, the GPU is the natural next step. But GPU programming has a reputation for being painful: obscure memory errors, race conditions that appear only at scale, performance cliffs you cannot see until you profile, and a programming model that feels alien if your background is sequential systems code.

CUDA Rust changes that story.

In September 2026, NVIDIA announced a formal commitment to GPU programming in Rust through two distinct tracks: cuda-oxide, which compiles idiomatic Rust directly to CUDA PTX using a custom rustc backend, and cutile-rs, a tile-based DSL that extends Rust ownership semantics to GPU kernels via JIT compilation through CUDA Tile IR. Both tracks share a core goal: bring Rust's fearless concurrency to the GPU. That means compile-time guarantees against data races, aliasing bugs, and use-after-free on device code that historically required meticulous manual discipline.

This book is your guide to the landscape. We start with fundamentals: what a GPU actually is, how it executes code differently from a CPU, and why CUDA's execution model matters even when you write in Rust. We then explore both official tracks in depth, showing how to set up your toolchain, write your first kernels, manage memory across the host-device boundary, and debug when things go wrong. Along the way, you will learn the patterns that separate a working kernel from a fast one: memory coalescing, tiling, occupancy tuning, warp-level operations, and the subtleties of synchronization.

You do not need prior CUDA experience. If you are comfortable with Rust: generics, traits, lifetimes, ownership, and perhaps some unsafe code: this book will take you from GPU programming fundamentals to advanced kernel design. If you have already written CUDA C++, you will find explicit comparisons throughout that map your existing knowledge to the Rust equivalents.

A few upfront notes on the current state of the ecosystem. CUDA Rust is young. Both official tracks are early-stage projects. APIs will change. Tooling gaps exist. This book reflects the state as of September 2026, but you should expect that by the time you read it, some details may have evolved. I flag experimental features explicitly and point you to upstream sources so you can track changes. The good news is that both tracks are already seeing production use. cutile-rs powers inference engines at Hugging Face and in the mistral.rs project. The community ecosystem around cudarc, Rust-GPU/rust-cuda, and related crates is mature enough to build real systems today.

Let us begin.

Chapter 1: Why GPU Computing, Why Rust

The modern software stack sits atop a hardware landscape that has become dramatically heterogeneous. For decades, general-purpose computation lived almost entirely on CPUs, and performance scaling came through clock speed increases, larger caches, and more cores. That scaling trajectory has changed. CPUs still matter enormously, but for certain classes of computation, GPUs now offer orders of magnitude more throughput. Understanding why, and when GPUs are actually the right tool for the job, is the foundation for everything else in this book.

The Parallel Computing Revolution

To understand why GPUs exist, consider how CPU designers hit their limits. For most of the twenty-first century, processor speeds roughly doubled every two years. Around 2005, that pattern broke. Heat dissipation and power consumption made clock rates beyond a few gigahertz impractical. The industry pivoted: instead of faster single cores, we got more cores. A modern desktop CPU might have sixteen to sixty-four physical cores. That is impressive, but it is also nowhere near enough for certain workloads.

A graphics processing unit started life as a specialized chip for rasterizing triangles, fragment shading, and pixel operations. Those tasks are highly parallel: millions of pixels must be shaded each frame, and each pixel's computation is largely independent of its neighbors. GPU architects leaned into that parallelism and built processors with hundreds to thousands of execution units arranged into clusters. NVIDIA calls these clusters Streaming Multiprocessors, or SMs. Each SM contains dozens of arithmetic units, registers, shared memory, and control logic capable of running thousands of threads concurrently.

The key insight that birthed general-purpose GPU computing, or GPGPU, was that those execution units are not specific to graphics. If you can express your problem as many small, independent operations on large datasets, a GPU

can run them all at once. Matrix multiplication, convolution, image filtering, particle simulation, Monte Carlo sampling, neural network inference: all of these are embarrassingly parallel at some level. A GPU can execute millions of these operations in the time it takes a CPU core to execute thousands.

But parallelism on a GPU is not automatic. You cannot simply compile your sequential code and expect speedup. The GPU demands a different programming model that matches its hardware architecture. That model is CUDA, and that is what we are here to master in Rust.

Where GPUs Excel and Where They Do Not

GPUs are not magic accelerators for everything. They shine in specific situations and they can be dramatically worse than CPUs for others. Learning the boundary is as important as learning the CUDA programming model.

GPUs excel at throughput-oriented workloads where:

- The same operation applies to millions of data elements
- Each element can be processed independently
- The per-element computation is non-trivial enough to justify setup overhead
- Data fits in the GPU's memory or can be streamed efficiently

A canonical example is vector addition on two arrays of one million elements each. On a CPU, you loop through the arrays sequentially, perhaps using vectorized instructions. On a GPU, you launch one million threads, each adding one pair of elements. The GPU can execute thousands of those additions simultaneously.

GPUs also excel when data reuse patterns match their memory hierarchy. If threads in a block can cooperate to load data into fast shared memory and reuse it across multiple computations, the bandwidth advantage becomes enormous. That is the basis for optimized matrix multiplication, convolution kernels, and many machine learning primitives.

GPUs perform poorly when:

- Workloads are sequential or have complex dependencies between steps

- Branching is highly divergent: threads in the same group take different code paths
- Data fits easily in CPU caches and the computation per element is tiny
- Latency, not throughput, is the metric that matters

If you need to process a single database query or update one user's account, a GPU will be slower than a CPU because of the overhead of kernel launch, memory transfer, and thread management. GPUs are throughput machines. If you have enough work to keep thousands of threads busy, you can amortize that overhead.

A practical rule: if your bottleneck is a tight loop operating on arrays, matrices, or tensors, a GPU kernel is probably worth exploring. If your bottleneck is I/O, lock contention, or sequential algorithmic complexity, a GPU will not help.

The State of GPU Programming in 2025

For nearly two decades, CUDA C++ has been the lingua franca of GPU programming. It is mature, well-documented, and powers most production HPC and machine learning systems. But it has well-known pain points that frustrate practitioners.

CUDA C++ gives you raw power at the cost of raw responsibility. You manage device memory manually with cudaMalloc and cudaFree. Pointers cross the host-device boundary with no type system distinction. Race conditions in shared memory are a matter of discipline: forget a __syncthreads() call and you get silently wrong results. Off-by-one errors in thread indexing produce corruption that appears intermittently and fails to reproduce on demand. All of this is manageable, but it is also exhausting at scale.

The ecosystem has grown rich workarounds. Libraries like cuBLAS, cuFFT, and cuDNN provide highly optimized implementations of common operations. Higher-level frameworks like PyTorch and TensorFlow hide most of the complexity from their users. But when you need to write custom kernels: for a novel algorithm, a performance-critical operation, or domain-specific computation: you drop down to CUDA C++ and face all of the pitfalls directly.

Rust has been exploring GPU programming for years. Early efforts targeted LLVM's PTX backend, which produced unreliable code for many Rust patterns.

The Rust-GPU community built rustc_codegen_nvvm, a more robust compiler backend. Independent crates like cudarc provided safe wrappers around the CUDA driver API for host-side management. The pieces were there, but integration was fragmented and the toolchain was fragile.

That is what NVIDIA's September 2026 announcement changed. The company recognized that Rust's strengths: ownership, borrow checking, zero-cost abstractions: map naturally to the GPU programming problem. Both data races and memory aliasing are fundamental hazards in CUDA C++. Rust eliminates them at the source. NVIDIA's investment in cuda-oxide and cutile-rs is an explicit bet that Rust will become a first-class language for GPU kernels.

Why Rust for GPU Development

If you already know Rust, you likely have a sense of its advantages: no data races, no null pointer dereferences, no use-after-free, no buffer overruns. For GPU code, those guarantees are not nice to have: they are essential.

Consider a shared memory reduction kernel. In CUDA C++, threads cooperatively write partial results into a shared array. A synchronization barrier must appear between each load and store phase. If you miss it, threads read stale values and the reduction produces garbage. The compiler will not catch that error. It may not appear in testing if your data size is small and memory happens to be in a lucky state.

In Rust, the borrow checker can enforce that mutable access to shared buffers is properly scoped. You cannot accidentally alias two mutable references to the same device memory region. The types themselves encode who owns what and for how long. When you pass a mutable tensor slice to a kernel, the compiler knows that no other code can simultaneously read or write that data.

That is the core promise of CUDA Rust: fearless concurrency on the GPU. The paper "Fearless Concurrency on the GPU" (arXiv:2606.15991) formalizes this claim. Both official tracks enforce memory safety at compile time: cuda-oxide through DisjointSlice and launch contracts, cutile-rs through tensor partitioning and ownership semantics. You cannot write a kernel with an aliasing bug in the first place.

But Rust brings more than safety. Its type system and macro system let you build abstractions that are as clear as high-level libraries and as efficient as

hand-tuned CUDA C++. You can define traits for different reduction strategies, use generics for different tensor shapes, and leverage const generics for compile-time loop unrolling. Zero-cost abstractions are not a marketing slogan in Rust: they are an engineering reality.

Rust's async model also maps naturally to CUDA's stream-based execution. A CUDA stream is a queue of operations that execute on the GPU. Multiple streams can overlap their work. cuda-oxide's DeviceOperation model lets you compose kernels and memory transfers as async futures, scheduling them across streams with round-robin or custom policies. This is idiomatic Rust async: lazy computation graphs that you execute with a runtime like Tokio.

And then there is ecosystem. If you are building a GPU-accelerated Rust application, you are probably integrating with other Rust crates: Serde for configuration, Tracing for logging, Tokio for async I/O, various data formats, networking stacks. CUDA Rust lets you do that integration without crossing language boundaries. Your host-side code remains idiomatic Rust. Your kernels compile to the same PTX that cuBLAS and cuDNN consume. You can call existing CUDA libraries from Rust when you do not need custom kernels.

How This Book Is Organized

This book takes you from GPU fundamentals to production-quality CUDA Rust kernels in a logical progression. You do not need prior CUDA experience, but you should be comfortable with Rust: its type system, ownership model, traits, and the basics of unsafe code.

The book is divided into five parts.

Part I establishes foundations. We cover GPU architecture, the CUDA execution model, and how CUDA C++ works as a reference. You need this background to understand why CUDA Rust is designed the way it is.

Part II introduces the CUDA Rust ecosystem. We explore both official tracks: cuda-oxide and cutile-rs. You will set up your toolchain, understand the API design, and run your first kernels. We also survey the broader ecosystem: Rust-GPU/rust-cuda, cudarc, and related tools.

Part III teaches kernel writing from the ground up. Device code, memory management, kernel launches, shared memory, and global memory optimization are covered with complete examples. Every concept is illustrated with code you can run and modify.

Part IV tackles advanced topics. Synchronization, ownership and safety patterns, error handling, numerical considerations, interoperability with existing CUDA libraries, and integration with larger Rust applications. This is where you learn to write robust kernels for production.

Part V covers the practicalities of GPU development in the real world. Debugging, profiling, optimization, testing, continuous integration, and deployment. You will learn how to find bottlenecks, measure performance correctly, and optimize systematically rather than hoping for the best.

A word on examples. Every code example in this book is designed to be complete and runnable. I include the full project setup: Cargo.toml dependencies, environment requirements, build commands, and expected output. When an API is experimental or likely to change, I flag it explicitly. If a pattern is specific to one track and not the other, I say so.

Let us begin the journey.

Chapter 2: GPU Architecture and Execution Model

Before writing GPU kernels, you must understand what hardware you are targeting. The GPU is not a faster CPU with more cores. It is a fundamentally different machine, designed around a different set of trade-offs. If you treat it like a CPU, your code will be correct but slow. If you understand its architecture, you can write kernels that approach its theoretical limits.

This chapter builds a mental model of GPU hardware and execution. We cover Streaming Multiprocessors, warps, SIMT execution, thread hierarchies, and memory organization. You will see how a kernel launch maps to physical operations on the chip. This is the foundation for everything that follows.

From CPU to GPU: A Hardware Perspective

Let us contrast a CPU core with a GPU execution unit.

A modern CPU core is wide, deep, and complex. It has large L1, L2, and often L3 caches. It uses out-of-order execution, speculative execution, branch prediction, and complex scheduling logic to squeeze instructions from every cycle. It can handle complex control flow with minimal penalty. That complexity consumes silicon area and power. A high-end CPU might have sixteen to sixty-four cores.

A GPU execution unit is narrow, shallow, and numerous. It has smaller caches, no speculative execution, and simpler control logic. In return, a GPU packs hundreds or thousands of such units onto a single chip. On NVIDIA GPUs, these units are organized into Streaming Multiprocessors. Each SM contains dozens of arithmetic logic units, special function units, tensor cores for matrix operations, shared memory, register files, and warp schedulers. A modern GPU might have dozens of SMs, giving it thousands of total execution units.

The design philosophy differs:

- A CPU core is optimized for single-threaded latency. It wants each instruction to finish as quickly as possible so your sequential program runs fast.
- An SM is optimized for multi-threaded throughput. It wants thousands of threads available so that when one thread stalls on memory, another can execute immediately. The latency of any individual thread is less important than the total work completed per second.

This is why GPU code must be massively parallel. The hardware hides memory latency by switching between thousands of threads. If you only launch a few threads, the GPU sits idle waiting for memory, just like a CPU would, but with less sophisticated recovery mechanisms.

Streaming Multiprocessors and Execution Units

An SM is the fundamental scheduling and execution unit on a CUDA GPU. When you launch a kernel, the CUDA runtime distributes blocks of threads across the available SMs. Each SM executes those blocks independently and in parallel with other SMs.

Within an SM, threads are not scheduled individually. They are grouped into warps. A warp is always 32 threads on current NVIDIA hardware. This is a fixed architectural constant, not a configuration parameter. All 32 threads in a warp execute the same instruction at the same time, but on different data. This is the SIMT model: Single Instruction, Multiple Threads.

Each SM contains multiple warp schedulers (typically two or four depending on architecture). Every cycle, each scheduler can dispatch one instruction to a ready warp. A warp is ready when its threads do not need data that is currently in flight: for example, when a memory load is pending, the warp stalls until the data arrives. The SM maintains dozens of resident warps. While one warp waits for memory, other warps execute. This is latency hiding: the hardware switches between warps so fast that you never notice the stalls.

The number of warps an SM can hold depends on available resources:

- Registers: Each thread gets its own private registers. If your kernel uses many registers per thread, fewer warps fit in the SM.

- Shared memory: Each block gets a share of the SM's shared memory. If your kernel uses a lot of shared memory, fewer blocks (and thus fewer warps) can reside on the SM simultaneously.

This trade-off is called occupancy, and we will discuss it in detail in Chapter 20. For now, the key insight is that an SM is most productive when it has enough resident warps to keep its execution units busy despite memory latency.

The Thread Block and Grid Execution Model

When you write a CUDA kernel, you do not specify how many hardware cores to use. Instead, you organize your work into a hierarchy of thread blocks, which are arranged into a grid. This is an execution model, not a physical layout. The runtime maps blocks to SMs dynamically.

A thread block is a group of threads that can cooperate through shared memory and barriers. All threads in a block execute on the same SM. A block is limited in size: up to 1024 threads on current hardware. Within a block, threads are assigned unique local indices.

A grid is a collection of blocks that work together to solve a larger problem. Threads in different blocks cannot share memory or synchronize directly. They communicate through global memory. The grid can span many SMs, and blocks can execute in any order on any SM. The runtime handles placement.

This design gives you scalability. Your kernel code does not depend on how many SMs the GPU has. You launch a grid large enough to cover your data, and the runtime distributes it across whatever hardware is available. A kernel that runs on a GPU with six SMs will also run on one with one hundred SMs, with better performance.

Threads within a block or grid are organized in one, two, or three dimensions. The CUDA runtime provides built-in variables that let each thread compute its position:

- threadIdx: the thread's index within its block
- blockIdx: the block's index within the grid
- blockDim: the number of threads in each dimension of a block
- gridDim: the number of blocks in each dimension of the grid

For a one-dimensional grid of blocks with 256 threads each, processing an array of 1024 elements, each thread computes its global index as:

```
global_id = blockIdx.x * blockDim.x + threadIdx.x
```

This gives values from 0 to 1023, one per element. If the array size is not a multiple of the total thread count, threads that compute an out-of-range index simply do nothing. We will see this pattern repeatedly.

Warp Execution and SIMT

SIMT stands for Single Instruction, Multiple Threads. All 32 threads in a warp share one program counter. When the warp executes an instruction, every thread performs it, each on its own data. This is similar to SIMD (Single Instruction, Multiple Data) on CPUs, but with a crucial difference: each thread in a warp has its own independent program counter and registers.

That independence enables divergent control flow. Consider this kernel code:

```
if (threadIdx.x < 16) {
    a[threadIdx.x] = 1.0f;
} else {
    a[threadIdx.x] = 2.0f;
}
```

The first 16 threads take the if branch. The remaining 16 threads take the else branch. Because the warp must execute one instruction at a time, the hardware serializes these paths: it executes the if body with only the first 16 threads active, then executes the else body with only the remaining 16 threads active. The inactive threads are masked off and do not consume arithmetic cycles, but the cycle still passes. This is warp divergence.

Warp divergence reduces performance because some cycles are partially idle. The worst case is every thread taking a different path, requiring 32 serialized executions. The best case is all threads taking the same path, executing in one cycle. Good kernel design minimizes divergence, especially in tight loops.

On architectures with compute capability 7.0 and later, NVIDIA introduced independent thread scheduling. This allows threads within a warp to maintain separate program counters more efficiently, reducing divergence overhead in some cases. However, SIMT remains the conceptual model: your kernel should be written assuming all threads execute the same instruction sequence, and divergence should be an optimization concern, not a primary design strategy.

Memory Hierarchies on the GPU

The GPU has a rich memory hierarchy with vastly different latency and bandwidth characteristics. Understanding it is essential for writing fast kernels. Here is the hierarchy from fastest to slowest:

Registers are private to each thread. They are the fastest memory available: roughly one cycle latency. The compiler allocates local variables to registers when possible. If a thread needs more registers than are available, excess variables spill into local memory, which is actually in global memory and much slower.

Shared memory is a small, fast scratchpad shared by all threads in a block. It resides on the SM and has roughly five cycle latency: about 100 times faster than global memory. You explicitly declare and manage shared memory in your kernel code. It is the key optimization primitive for tiling: threads cooperate to load data from global memory into shared memory, then reuse it across multiple computations. Shared memory is divided into 32 banks for parallel access. If multiple threads in a warp access the same bank in the same cycle, a bank conflict occurs and accesses serialize. We will cover bank conflicts in detail in Chapter 10.

Global memory is the main device memory: gigabytes of DRAM accessible by all threads in all blocks. It has high latency: roughly 500 cycles or more. It also has high bandwidth: modern GPUs can sustain hundreds of gigabytes per second. Global memory is cached by L1 and L2 caches, which reduce latency for reused data but do not change the fundamental hierarchy. Good kernels minimize global memory traffic through coalesced access patterns and tiling.

Constant memory is a small, read-only region optimized for broadcasting: when all threads in a warp read the same constant value, it is delivered efficiently in a single transaction. It has its own cache and is useful for

configuration parameters, lookup tables, and other data that does not change during kernel execution.

Local memory, despite its name, is thread-private memory that resides in global DRAM. It is used when the compiler cannot fit all of a thread's variables into registers. This is register spilling, and it is a common performance killer. If profiling shows excessive local memory usage, reducing register pressure is a priority optimization.

Texture memory historically provided spatial interpolation and caching for 2D or 3D data. On modern GPUs, it has largely merged with cached global memory access. For general-purpose computing, you can usually ignore texture memory unless you are doing graphics or specialized image processing.

Here is a conceptual comparison:

```
Memory Type       Scope        Lifetime      Latency     Typical Use
-----------------------------------------------------------------
Registers         Thread       Kernel        ~1 cycle    Local vars
Shared            Block        Kernel        ~5 cycles   Tiling, coop
L1/L2 Cache       SM/Device    Dynamic       30-100 cyc  Auto caching
Global            Grid         App           ~500+ cyc   Main storage
Constant          Grid         App           Low*        Config data
```

*Low when accessed uniformly across a warp.

How a Kernel Launch Traverses the Hardware

Let us trace what happens when you launch a kernel, from host code to GPU execution. This mental model will help you understand performance characteristics and debug issues later.

First, the host (CPU) prepares the kernel launch. It allocates device memory, transfers input data from host to device, and constructs a launch configuration specifying the grid and block dimensions.

The host then issues the kernel launch call. This does not block: kernel launches are asynchronous with respect to the host. The call enqueues the kernel on a CUDA stream, which is a queue of GPU operations. The default stream serializes with all other operations. Non-default streams can overlap work.

The CUDA runtime receives the launch request and schedules blocks onto SMs. Blocks are the unit of scheduling: a block is assigned to one SM and stays there until it completes. The runtime may place multiple blocks on the same SM if resources permit, or spread them across many SMs if the grid is large. Block scheduling order is not guaranteed and may vary between runs.

Within each SM, the block's threads are partitioned into warps. The SM's warp schedulers dispatch instructions from ready warps each cycle. Warps that are waiting on memory or synchronization barriers are paused while other warps execute.

Threads execute the kernel code according to the SIMT model. They read from and write to memory according to the access patterns you wrote. They cooperate through shared memory and synchronize through barriers. When all threads in a block have finished execution, the block completes.

When all blocks in the grid have completed, the kernel launch is done. The host can synchronize on the stream to ensure completion, or it can launch subsequent kernels or transfers that implicitly wait for prior work.

Throughout this process, the GPU operates asynchronously from the host. The host can continue executing CPU code while the GPU processes the kernel. This overlap is fundamental to GPU programming. Good designs keep both CPU and GPU busy, using multiple streams to pipeline operations.

This architecture has implications that will recur throughout the book:

- Kernels should have enough threads to keep SMs saturated. A small kernel with 128 threads on a GPU with thousands of execution units wastes most of the hardware.
- Memory latency is hidden by switching between warps. If you do not have enough threads, latency becomes visible and performance suffers.
- Blocks are independent. Design algorithms so that blocks can execute in parallel without synchronization.
- Kernel launch has overhead. For tiny amounts of work, the overhead exceeds the computation, making the CPU a better choice.

In the next chapter, we will see how CUDA C++ expresses this execution model, which sets the stage for understanding CUDA Rust's abstractions.

Chapter 3: CUDA Fundamentals: The C++ Reference Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The CUDA C++ Kernel Syntax

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Host Code Versus Device Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Launch Configuration and Execution Dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Memory Allocation and Data Transfer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

A Complete CUDA C++ Example Walked Through

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

What CUDA Rust Will Build Upon

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 4: Introducing CUDA Rust: History and Vision

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Road to GPU Programming in Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

NVIDIA's CUDA Rust Initiative

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Two Tracks: Native and C Bindings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The SIMT Track: cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Tile Track: cutile-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Choosing Between Tracks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Design Goals and Philosophy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Current Status, Stability, and Versioning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

What CUDA Rust Is Not

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 5: Toolchain Setup and Project Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Prerequisites: Hardware, CUDA Toolkit, and Drivers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

GPU Requirements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA Toolkit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

NVIDIA Drivers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

System Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Installing the CUDA Rust Compiler and Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Installing cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Installing cutile-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Project Layout with cargo-cuda

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Project Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Project Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Build Configuration and Environment Variables

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Your First CUDA Rust Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide: Hello GPU

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs: Hello Tile

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Troubleshooting Common Setup Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“cuda-oxide codegen backend not found”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“CUDA toolkit not found”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“compute-sanitizer or ncu: command not found”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“JIT compilation failed” (cutile-rs)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“Driver version mismatch”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

“Segmentation fault on kernel launch”

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 6: The CUDA-Rust API Surface

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The core_cuda and cuda-rs Crates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Crates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Crates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Pointer Types and Abstractions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

DisjointSlice (cuda-oxide)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Tensor (cutile-rs)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Comparison with CUDA C++

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Kernel Trait and Launch Mechanics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Kernel Definition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Kernel Definition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Launch Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Memory Allocation and Transfer APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Memory Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Memory Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Utility Types: Dimensions, Streams, Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

How the API Maps to CUDA C++ Concepts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 7: Device Code Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Marking Code as Device Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide: #[kernel] and #[cuda_module]

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs: #[cutile::module]

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

What Can Device Code Do?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Computing Thread and Block Indices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Thread Indexing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs: Tile-Based Indexing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Your First Kernel: Vector Addition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Vector Addition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Vector Addition

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Understanding the Launch Dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Block Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Grid Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Execution Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Compilation: How Device Code Becomes PTX and SASS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Compilation Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs JIT Compilation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Performance Implications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Safety Considerations in Device Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Compile-Time Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Runtime Risks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Unsafe in Device Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 8: Memory Management: Allocation and Transfer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Memory Allocation Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Basic Allocation in cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Allocation in cutile-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Memory Capacity and Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Fragmentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Host-to-Device and Device-to-Host Transfers

Synchronous Transfers in cuda-oxide

Streaming Transfers

Transfers in cutile-rs

Pinned Memory and Asynchronous Transfers

Why Pinned Memory?

Using Pinned Memory in cuda-oxide

Using Pinned Memory in cutile-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Cost Model of Data Movement

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Transfer Time Estimates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Strategies to Minimize Transfers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Multi-Operation Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Memory Safety at Host and Device Boundaries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Safety Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs Safety Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Common Boundary Bugs (Prevented)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

What You Are Still Responsible For

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Complete Example: Matrix Initialization and Transfer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 9: Kernel Launch Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Launch Configuration and Grid Size Calculations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Basic Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Two-Dimensional Grids

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Three-Dimensional Grids

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs: Implicit Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Occupancy and Heuristics for Block Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

What Occupancy Is and Is Not

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Factors Limiting Occupancy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Choosing Block Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Grid Size: Big Enough

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Streams and Concurrent Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Why Streams?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Creating and Using Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Ordering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Synchronizing Between Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Synchronous Versus Asynchronous Launches

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Synchronous Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Asynchronous Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda-oxide Async API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Kernel Chaining and Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Simple Chaining on One Stream

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Parallel Chaining with Multiple Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Dependency Graphs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Multi-Kernel Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 10: Shared Memory and Block-Level Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory: Purpose and Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

When to Use Shared Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Declaring and Using Shared Memory in Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Static Shared Memory in cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Dynamic Shared Memory in cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory in cutile-rs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Matrix Transpose Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Naive Transpose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimized Transpose with Shared Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Warp Coalescing and Memory Access Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Coalesced Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stride and Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Structure of Arrays vs Array of Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

A Reduction Kernel with Shared Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Naive Atomic Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Warp Shuffle Reduction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Barriers and Synchronization Within a Block

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

thread::sync_threads()

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Conditional Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Multiple Barriers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 11: Global Memory and Data Access Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Global Memory Architecture and Caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Caching Layers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Write Caching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Coalesced Versus Uncoalesced Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Mechanics of Coalescing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Rules for Coalescing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Diagnosing Uncoalesced Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Bank Conflicts and Shared Memory Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

How Bank Conflicts Happen

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Avoiding Bank Conflicts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory vs L1 Cache Trade-off

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Texture and Constant Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Constant Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Texture Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Designing Kernels for Memory-Bound Workloads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Signs Your Kernel Is Memory-Bound

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimization Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Arithmetic Intensity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Image Filter Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Naive Version

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimized Version with Shared Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 12: Synchronization and Coordination

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Thread-Level Synchronization: __syncthreads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

How It Works

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Usage Rules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Warp-Level Primitives

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shuffle Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Ballot

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Syncwarp

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Cross-Block Synchronization Limitations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Why No Cross-Block Barriers?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Workarounds

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream-Based Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Ordering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Independence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Synchronization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Host-Device Synchronization and Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Synchronize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Synchronize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Non-Blocking Queries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Deadlock and Starvation Pitfalls

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Deadlock Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Starvation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Debugging Synchronization Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 13: Ownership, Lifetimes, and GPU Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Ownership Semantics for Device Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

DeviceBuffer as an Owned Resource

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Transfer of Ownership

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Ownership

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Lifetimes at the Host-Device Boundary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Borrowing for Kernel Launch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Async Launches and Lifetimes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

DisjointSlice: Safe Mutable Output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

RAII for CUDA Handles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CudaContext

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Streams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Aliasing and the Device Pointer Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Rust's Aliasing Rules

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Pointer Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Raw Pointers and Unsafe

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

DisjointSlice and Non-Aliasing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Avoiding Use-After-Free on the GPU

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

How RAII Prevents It

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Async Use-After-Free Prevention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Manual Lifetime Management (Unsafe)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Safe Abstractions Over Unsafe CUDA Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Safe Kernel Launcher

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cutile-rs: Safety by Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 14: Error Handling and Robust GPU Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA Error Types and Categories

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Driver API Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Kernel Launch Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Asynchronous Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Device Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Propagating Errors in Host Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using ? for Propagation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Custom Error Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Detecting Kernel Launch Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Immediate Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Deferred Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Runtime Versus Compile-Time Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Compile-Time Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Runtime Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Defensive Patterns for Production Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Validate Input Before Launch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Check Available Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Boundary Checks in Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

gpu_assert for Device-Side Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

gpu_printf for Tracing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Logging and Diagnostic Infrastructure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Structured Logging with Tracing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Error Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 15: Numerical Computing on the GPU

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Float32, Float64, and Mixed Precision

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Single Precision (FP32)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Double Precision (FP64)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Half Precision (FP16)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

BFloat16 and TF32

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

GPU Floating Point Semantics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

IEEE 754 Compliance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Fast Math Mode

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Portability of Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Denormals, NaN, and Infinity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Denormals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

NaN (Not a Number)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Infinity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Atomic Operations and Race Conditions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Global Memory Atomics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Atomic Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory Atomics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Race Conditions Without Atomics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Numerical Stability in Parallel Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Reduction and Associativity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Parallel Scan (Prefix Sum)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Matrix Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Parallel Statistical Computation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 16: Interoperability: CUDA Rust and the Wider World

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Calling CUDA C++ Libraries from Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using cudarc for Library Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Available Libraries via cudarc

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

FFI Patterns and Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Raw Pointer Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Lifetime Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Interfacing with cuBLAS and Linear Algebra

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

GEMM Example with cudarc

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuFFT and Transform Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

FFT Example with cudarc

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA Graphs and Advanced APIs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Building Hybrid CUDA C++ and CUDA Rust Projects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Strategy: Separate Compilation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example Project Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 17: Integration with Rust Application Ecosystems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Multi-Threaded Hosts and CUDA Contexts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA Context and Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Sharing Contexts Across Threads

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Per-Thread Contexts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA with Rust Async Runtimes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

cuda_async and Tokio

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Composing Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Stream Pool Scheduling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Blocking on Async

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CPU-GPU Hybrid Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Designing Hybrid Algorithms

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: CPU-GPU Image Processing Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Shared Memory Between Host Threads and Device

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Designing a GPU-Accelerated Rust Library

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

API Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Feature Flags

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Error Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Async GPU Processing Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 18: Debugging GPU Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Common GPU Kernel Bugs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Out-of-Bounds Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Race Conditions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Synchronization Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Launch Configuration Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Resource Exhaustion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using cuda-memcheck and Sanitizers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

memcheck: Memory Error Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

racecheck: Data Race Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

initcheck: Uninitialized Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

synccheck: Synchronization Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using Sanitizers in Development

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Debug Builds and GDB for CUDA

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Building for Debug

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Launching cuda-gdb

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Debugging Warps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Print-Based and Trace-Based Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

gpu_printf in cuda-oxide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

gpu_assert for Diagnostics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Debugging Deadlocks and Timing Issues

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Symptoms of Deadlock

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Common Deadlock Causes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Debugging Approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Systematic Approaches to Bug Isolation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Step 1: Verify Input Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Step 2: Reduce Problem Size

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Step 3: Serialize

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Step 4: Compare with CPU Reference

Step 5: Use Sanitizers

Chapter 19: Profiling and Performance Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

NVIDIA Nsight Systems and Compute Profilers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Nsight Systems (nsys)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Nsight Compute (ncu)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Occupancy Metrics and Utilization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Occupancy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Utilization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Memory Bandwidth Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Measuring Memory Bandwidth

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Memory-Bound vs Compute-Bound

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Identifying the Bottleneck

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Identifying Kernel Bottlenecks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Step-by-Step Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Common Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Roofline Model and Performance Reasoning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Understanding Roofline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using Roofline for Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Practical Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Iterative Optimization Case Study

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Baseline: Naive Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimization 1: Shared Memory Tiling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimization 2: Coalesced Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Optimization 3: Multiple Outputs Per Thread

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 20: Performance Optimization Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Register Pressure and Spilling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The Problem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Diagnosing Register Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Reducing Register Pressure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Instruction-Level Parallelism

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Factors Affecting ILP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Improving ILP

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Loop Unrolling and Predicate Elimination

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Benefits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Applying Unrolling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Warp Divergence and Control Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Minimizing Divergence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Acceptable Divergence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Using Specialized Instructions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Warp Shuffle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Tensor Cores

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Atomic Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Example: Optimizing a Convolution Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Version 1: Naive

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Version 2: Shared Memory Tile

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Version 3: Multiple Outputs Per Thread

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Version 4: Separable Convolution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Version 5: IM2COL + GEMM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Chapter 21: Production Deployment and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Testing GPU Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Unit Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Property-Based Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Numerical Tolerance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CPU Reference Comparison

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Continuous Integration for CUDA Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Testing Without GPU

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Docker-Based GPU Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Feature-Gated Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Monitoring in CI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

API Design for Reusable Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Simple, Composable Interfaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Type Safety

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Configuration via Traits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Documentation and Team Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Document Kernels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Document Performance Characteristics

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Team Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Portability and Hardware Targeting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Compute Capability Targeting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

CUDA Version Compatibility

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Multi-Vendor Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

When Not to Use CUDA Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Use CPU Instead When

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Use Existing Libraries Instead When

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Use CUDA C++ Instead When

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Use cutile-rs Over cuda-oxide When

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Use cuda-oxide Over cutile-rs When

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

The CUDA Rust Journey So Far

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Where the Ecosystem Is Heading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

Complementary Technologies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

A Final Word on GPU Computing with Rust

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/writinggpukernelswithcudarust.