Parallel Programming Models Overview

John Owens
UC Davis

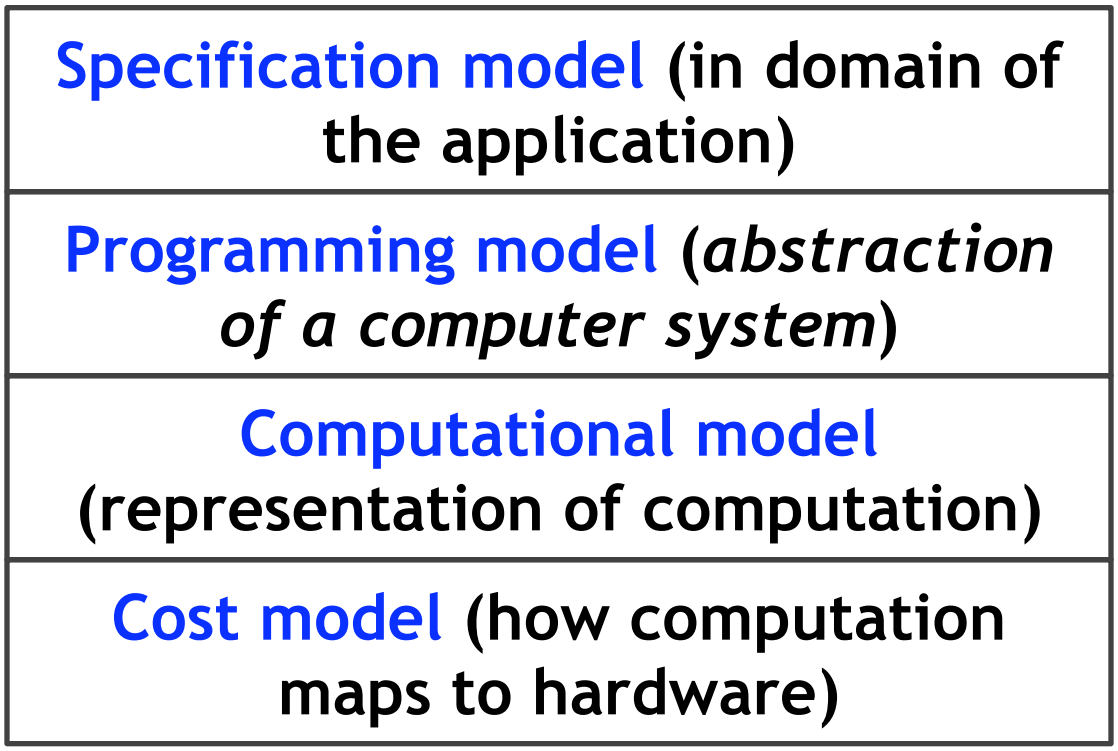
SIGGRAPH2008

Beyond Programmable Shading: Fundamentals

Bridge between HW and SW

- **What does the hardware look like?**
  - Many slimmed-down cores
  - Shared instruction stream across many cores
  - Interleaved execution of elements to hide latency
- **But … many different kinds of hardware share these characteristics**
- **Programming models present an abstraction of the hardware**

What is a programming model?

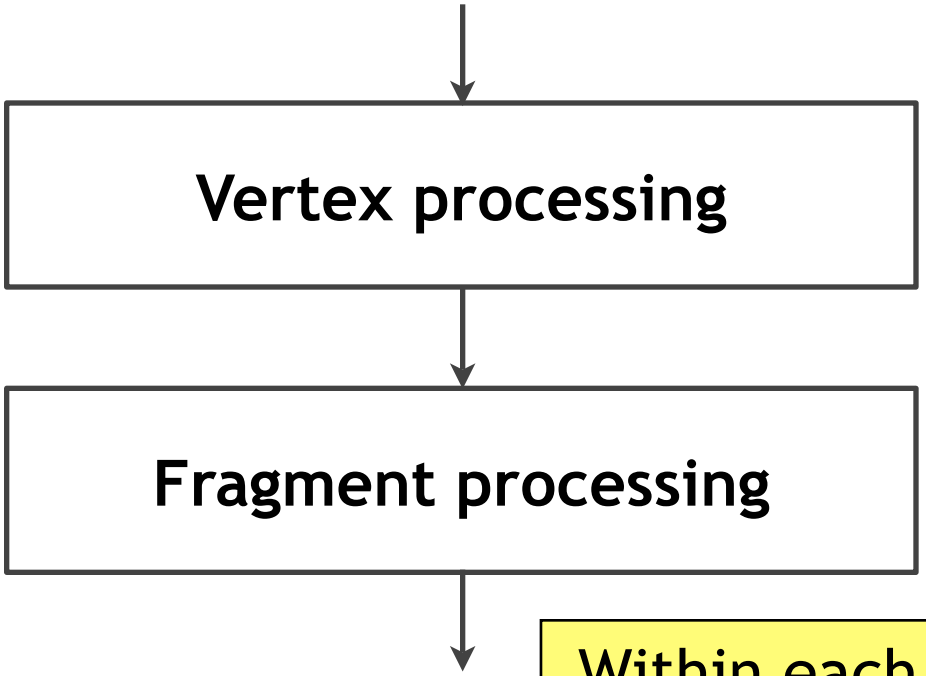


- **Is a programming model a language?**
  - Programming models allow you to express ideas in particular ways
  - Languages allow you to put those ideas into practice

Writing Parallel Programs (Top Down)

- ***Identify*** **concurrency in problem**
  - Do this in your head
- ***Expose*** **the concurrency when writing the code to solve the problem**
  - Choose a programming model that allows you to express this concurrency
- ***Exploit*** **the concurrency in the problem**
  - Choose a language and hardware that together allow you to take advantage of the concurrency

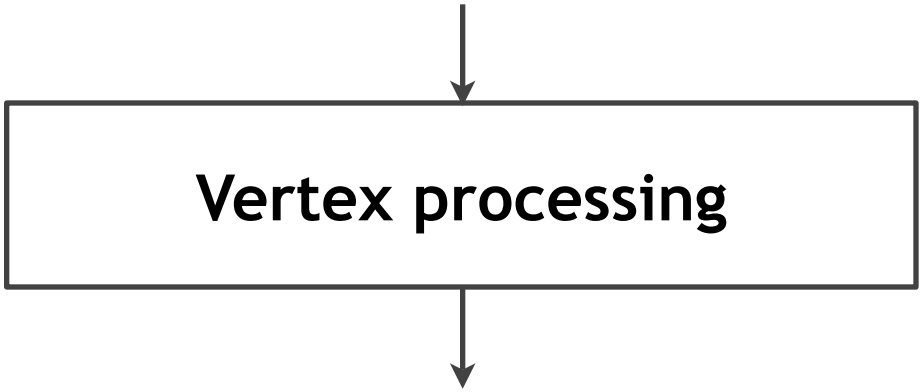
The Graphics Pipeline (simplified)



These two stages can run in parallel.

Within each of these stages, different data elements can run in parallel.

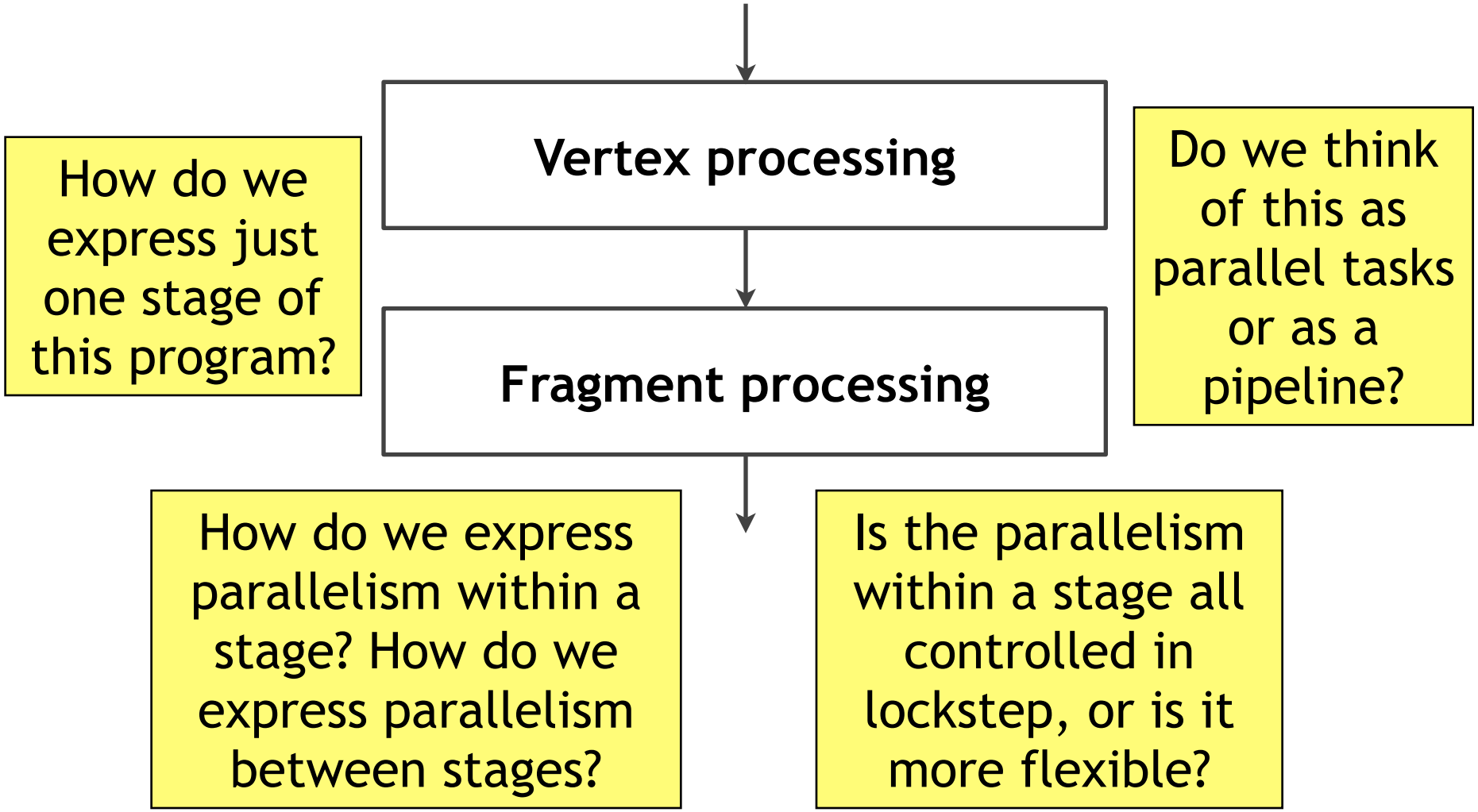
Threads / Units of Execution



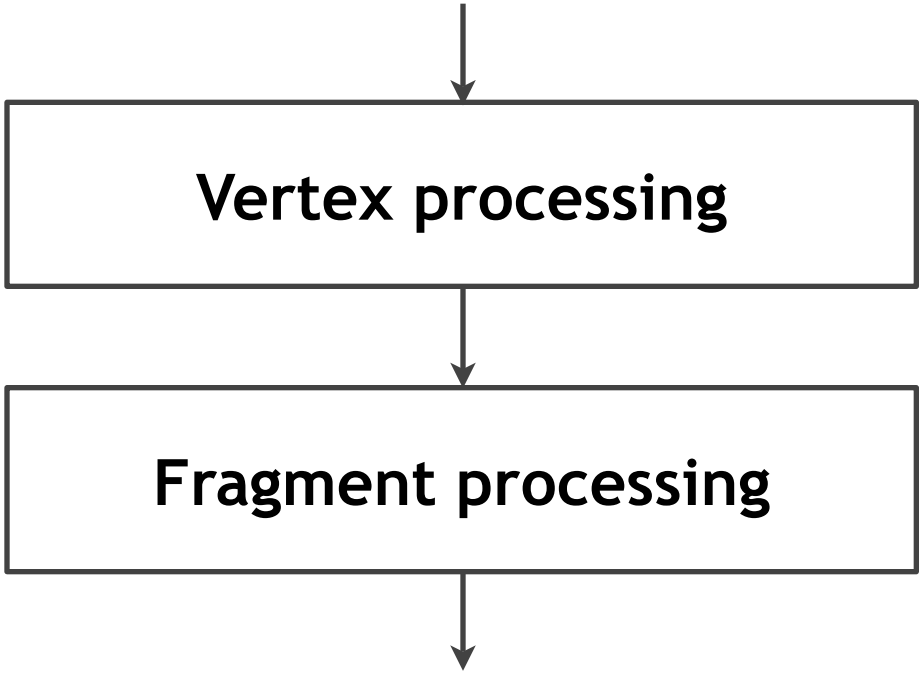
A graphics “vertex processing” stage processes many vertices.

- **Complex applications can be broken down into individual “units of execution” (UEs) or “threads”**
  - “Thread” is a loaded word but we’ll use it today
  - Programming model might restrict dependencies and/or communication between threads

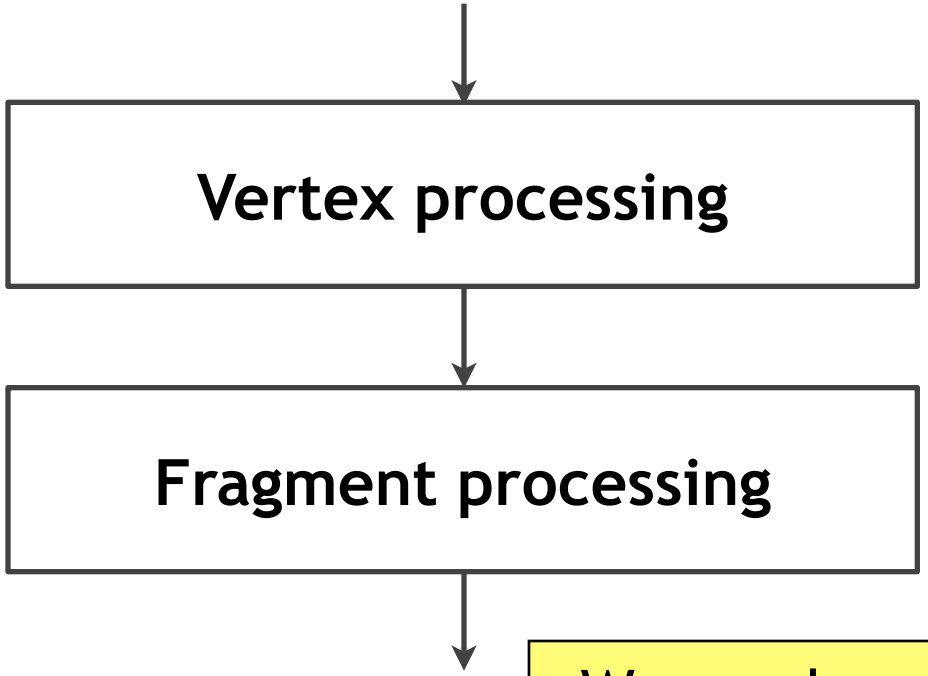
The Graphics Pipeline (simplified)



Kinds of Parallelism



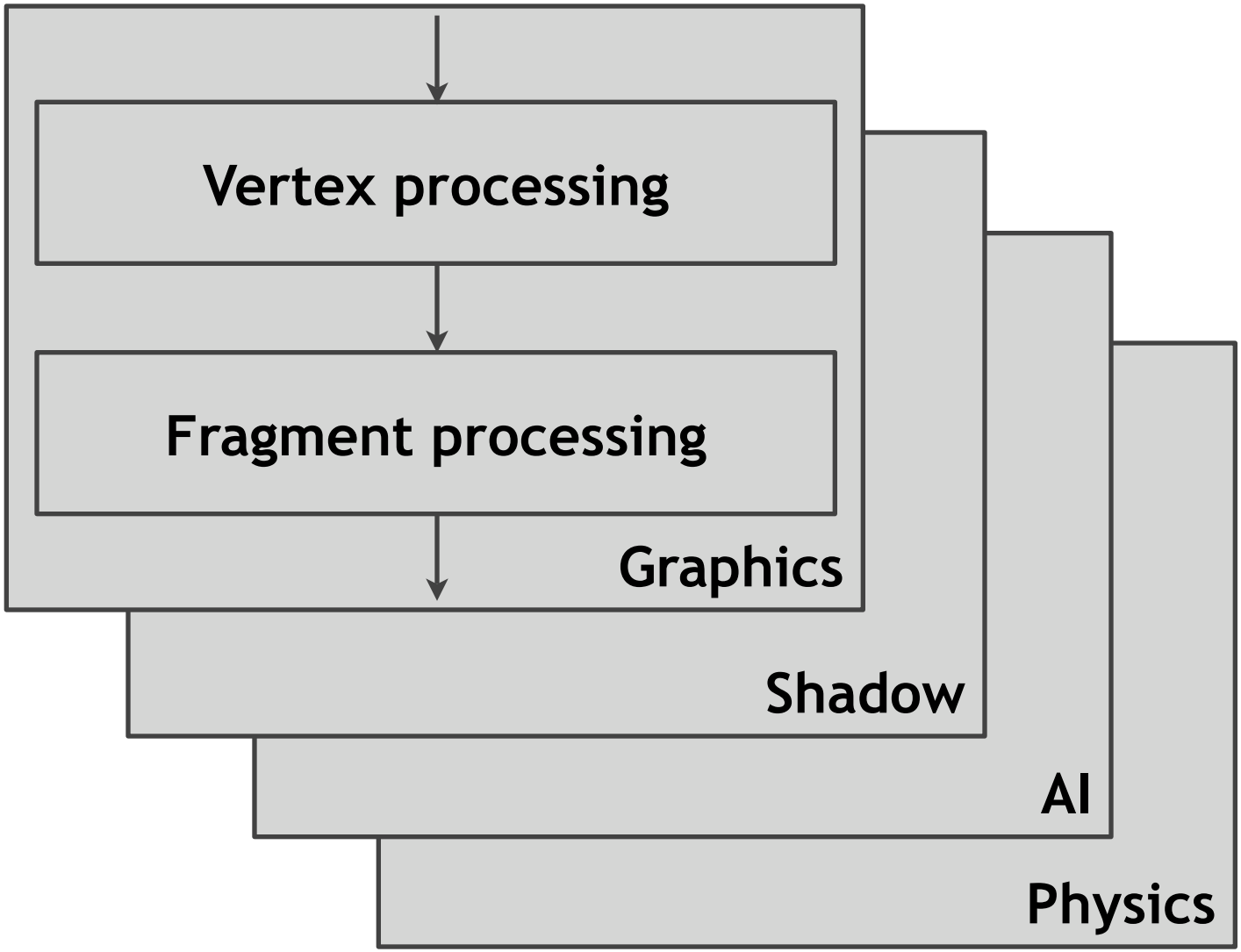
Task Parallelism (1)



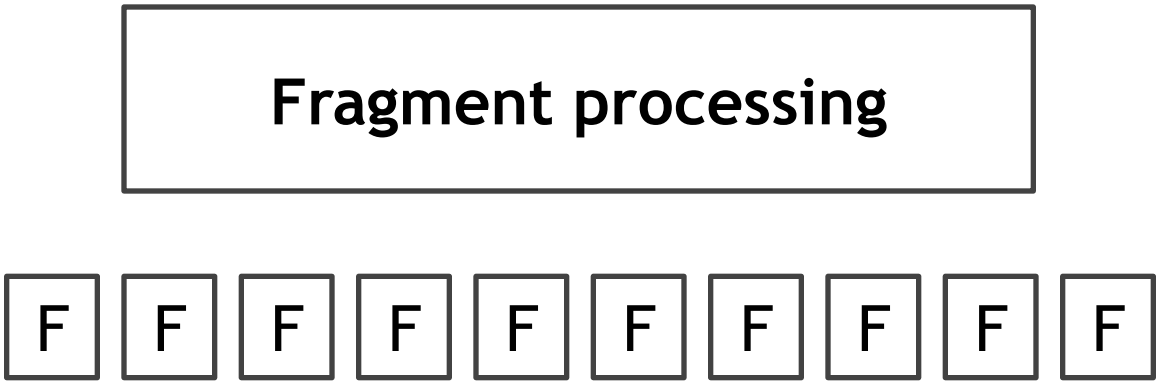
We may be able to process multiple stages at the same time. Here these are structured as a pipeline, but don't have to be.

We can have different hardware or cores work on different pipelines at the same time (task-parallel hardware) or timeslice on one piece of hardware (time-multiplexed).

Task Parallelism (2)



Data Parallelism



We may be able to process multiple data elements at the same time.

Are these elements all running the exact same code? The same program, but taking different directions? Different programs?

Terminology: SIMD, SPMD, MIMD

- **Models for exploiting data parallelism**
  - Many items can be processed in parallel
  - MD = “multiple data”
- **Are multiple threads processed ...**
  - in lockstep? [SIMD: Single Instruction, Multiple Data]
    - GPU model: early fragment programs
  - by the same program, but not in lockstep? [SPMD: Single Program, Multiple Data]
    - GPU example: CUDA
  - by different programs? [MIMD: Multiple Instruction, Multiple Data]
    - GPU example: vertex programs

Terminology: SIMD, SPMD, MIMD

- **Models for exploiting data parallelism**
  - Many items can be processed in parallel
  - MD = “multiple data”
- **Are multiple threads processed ...**
  - in lockstep? [SIMD: Single Instruction, Multiple Data]
    - GPU model: early fragment programs
    - Not SSE
  - by the same program, but not in lockstep? [SPMD: Single Program, Multiple Data]
    - Not MPI
    - GPU example: CUDA
  - by different programs? [MIMD: Multiple Instruction, Multiple Data]
    - GPU example: vertex programs

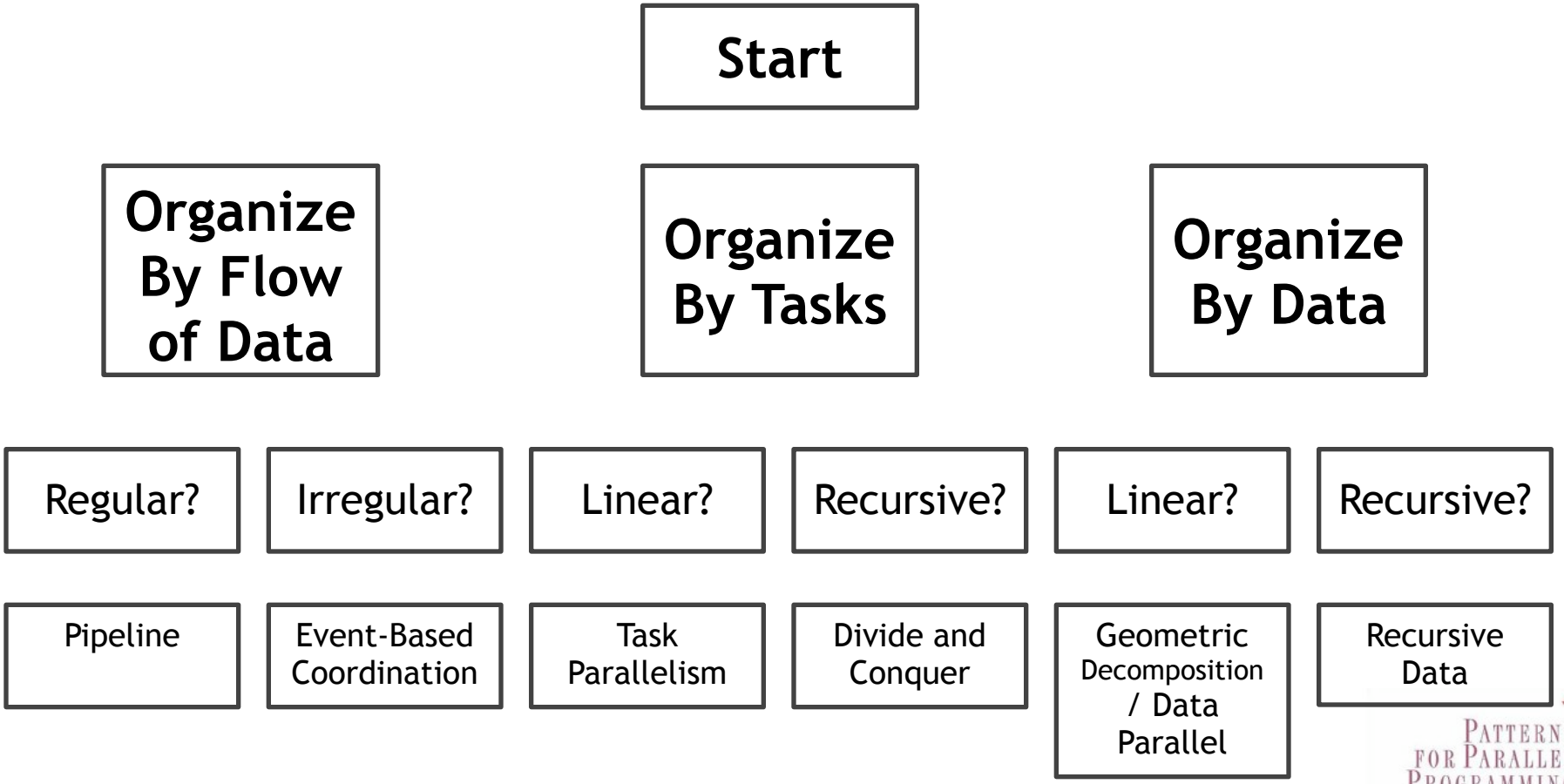
Terminology: Streaming

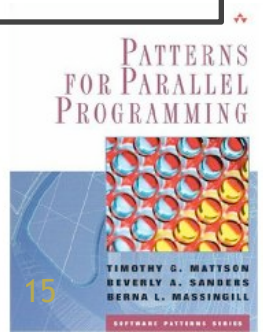
- **In a streaming programming model,**
  - Data structure: streams (list of data items)
  - Algorithm: kernel (operates on streams)
- **Streams have *implicit* parallelism**
  - Why? Programming model asserts *independence*
- **Limited visibility, *O*(1) storage**
- **Explicit data access pattern**
- **GPU example:**
  - Vertex streams
  - Brook (which had other features as well)

Terminology: Traditional Coprocessor

- **Is the parallel processor controlled by another processor ...**
  - ... that is responsible for calling tasks ...
  - ... or allocating/deallocating memory ...
  - This is the traditional coprocessor model
- **... or is the parallel processor responsible for its own global control?**
  - Allocates its own memory, calls its own kernels
  - Can submit work to itself
- **Coprocessors are adding features that are blurring these lines**

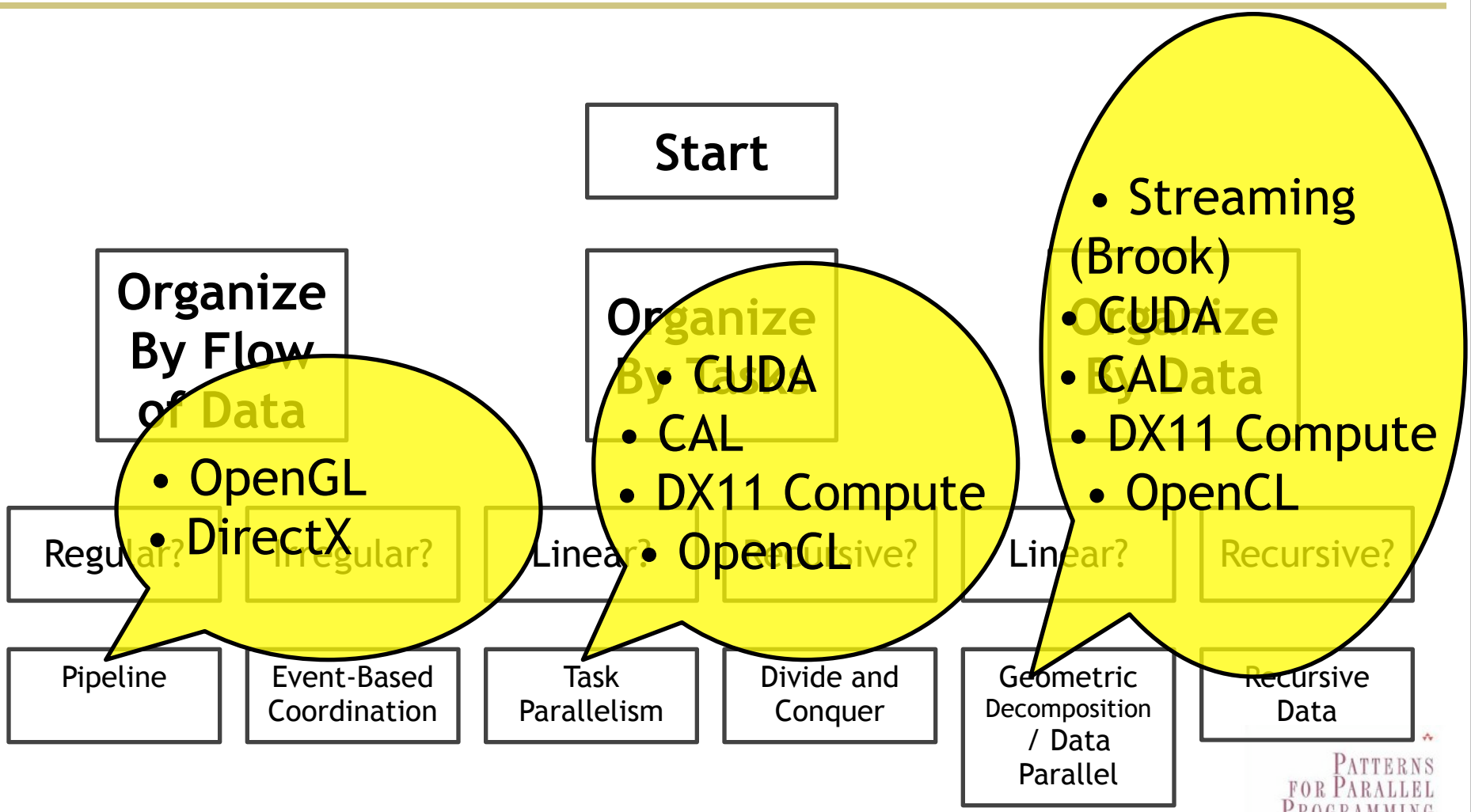
AlgorithmStructure Design Space

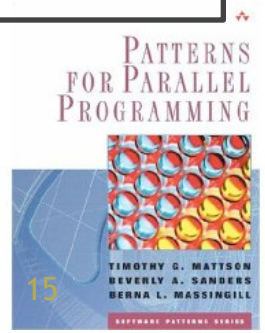




- Courtesy Tim Mattson, Intel

AlgorithmStructure Design Space





- Courtesy Tim Mattson, Intel

Bibliography

- Mattson/Sanders/Massingill, *Patterns for Parallel Programming*, Addison Wesley 2004
  - Thanks to Tim Mattson for his comments on this material!
- Buck et al., "Brook for GPUs: Stream Computing on Graphics Hardware", Siggraph 2004 [*streaming*]
- Lindholm et al., "A User-Programmable Vertex Engine", Siggraph 2001 [*implementation & programming model for programmable vertex processors*]