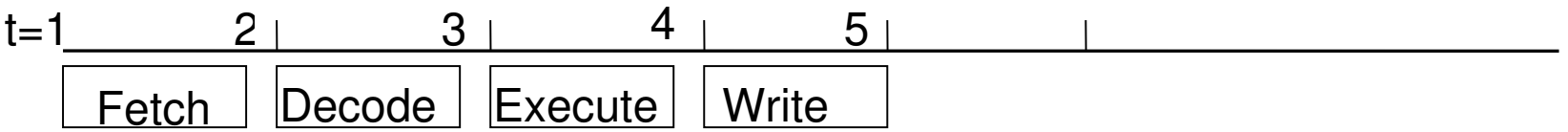
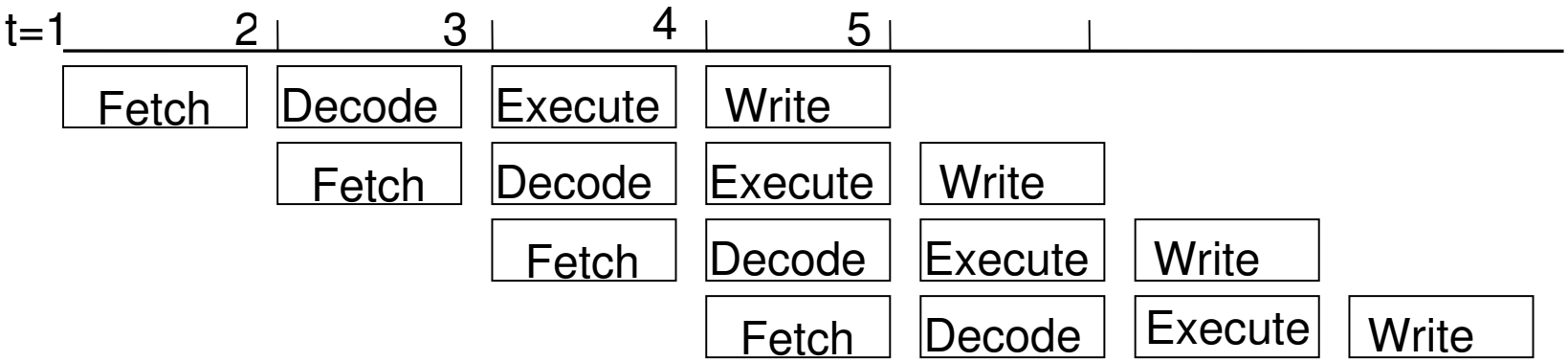
Basic Architecture Concepts



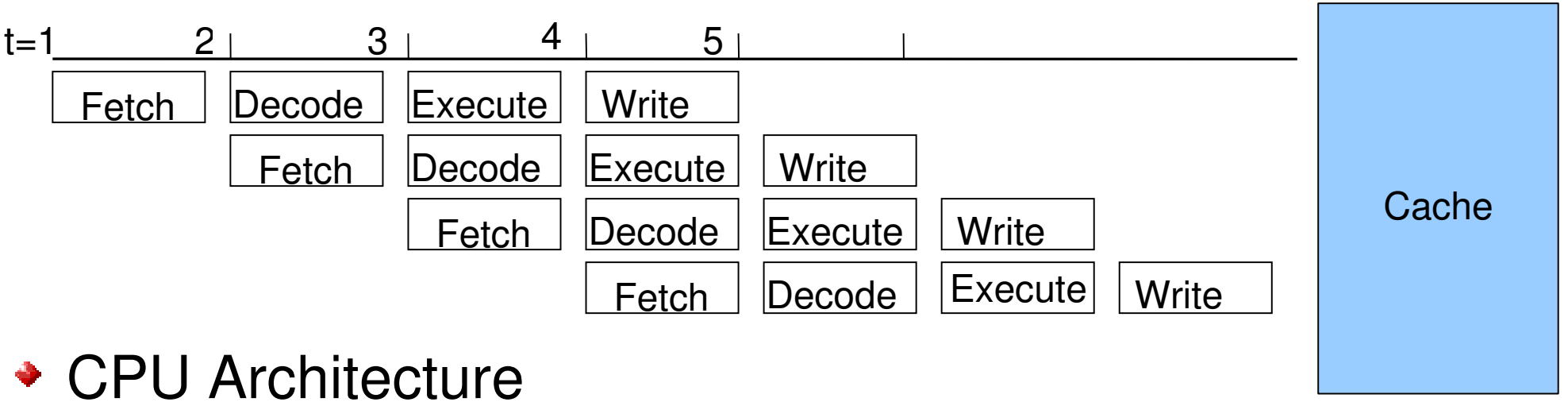
- CPU Architecture
  - 4 stages of instruction execution
    - Too many cycles per instruction (CPI)

Basic Architecture Concepts



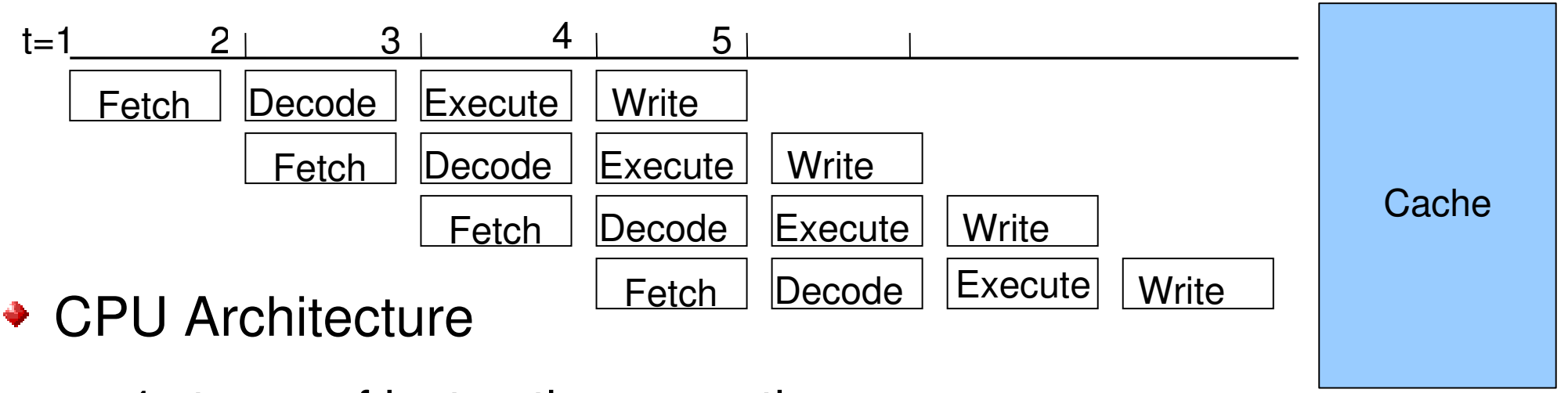
- CPU Architecture
  - 4 stages of instruction execution
    - Too many cycles per instruction (CPI)
  - To reduce the CPI, introduce pipelined execution

Basic Architecture Concepts



- CPU Architecture
  - 4 stages of instruction execution
    - Too many cycles per instruction (CPI)
  - To reduce the CPI, introduce pipelined execution
    - ✂ Needs buffers to store results across stages.
    - A cache to handle slow memory access times

Basic Architecture Concepts



- CPU Architecture
  - 4 stages of instruction execution
    - Too many cycles per instruction (CPI)
  - To reduce the CPI, introduce pipelined execution
    - Needs buffers to store results across stages.
    - A cache to handle slow memory access times
    - Multilevel caches, out-of-order execution, branch prediction, ...

Basic Architecture Concepts

- CPU architecture getting too complex.
- Not translating to equivalent performance benefits
  - Need a rethink on traditional CPU architectures.

Basic Architecture Concepts

- Couple with this the new wisdom in computer architectures.
  - Memory Wall – memory latencies far higher
  - ILP Wall – Reducing benefits from instruction level parallelism
  - Power Wall – Increase in power consumption with increase in clock rates.
- Multi-core is the way forward
  - Ex: GPUs, Cell, Intel Quad core, ...
  - Predicted that 100+ core computers would be a reality soon.

Multicore and Manycore Processors

- IBM Cell
- NVidia GeForce 8800 includes 128 scalar processors and Tesla
- Sun T1 and T2
- Tilera Tile64
- Picochip combines 430 simple RISC cores
- Cisco 188
- TRIPS

The Case for the GPUs

- GPUs are now common. They also have high computing power per dollar, compared to the CPU
- Today's computer system has a CPU and a GPU, with the GPU being used primarily for graphics.
- GPUs are good at some tasks and not so good at others. They are especially good at processing large data such as images.
- Let us use the right processor for the right task.
- Goal: Increase the overall throughput of the computer system on the given task. Use CPU and GPU synergistically.

Evolution of GPUs

- Graphics: a few hundred triangles/vertices map to afewhundred thousand pixels
- Process pixels in parallel. Do the same thing on a large number of different items.
- Data parallel model : parallelism provided by the data
  - Thousands to millions of data elements
  - Same program/instruction on all of them
- Hardware: 8-16 cores to process vertices and 64-128 to process pixels by 2005
  - Less versatile than CPU cores
  - SIMD mode of computations. Less hardware for instruction issue
  - No caching, branch prediction, out-of-order execution, etc.
  - Can pack more cores in same silicon die area

GPUs as a Case Study

- GPGPU – General Purpose Programming on GPUs
- OpenGL extensions
  - Very difficult to program
- Recently manufactures started supporting C-like programming abstraction to program GPUs
  - CUDA from NVidia
- Other benefits of GPGPU
  - Affordable cost, easy availability, computational power

GPUs as a Case Study

- GPUs suited for routines with high arithmetic intensity.
- One feature is high memory latency, depending on the nature of access.
  - Should overlap memory with arithmetic.

CPU Vs GPU

- Few powerful cores Vs. lots of small cores
- GPUs: For good performance, applications need high arithmetic intensity
- GPUs : No system managed cache.

GPGPU as a Case Study

- Regular algorithms
  - Map well to data parallel model of GPUs
  - Each work item operates by itself or with a few neighbors
  - Example settings : image processing.
  - Threads can share data, e.g., apron pixels in an image processing kernel.

GPU as a Case Study

- Irregular algorithms
  - Applications with data accesses that are not regular in nature.
  - Occurs in settings such as graph algorithms, data structures building, etc.
  - Difficult to get high efficiency due to high memory latency of accesses.

GPGPU Tools and APIs

- OpenGL
- CUDA
- OpenCL
- Brook