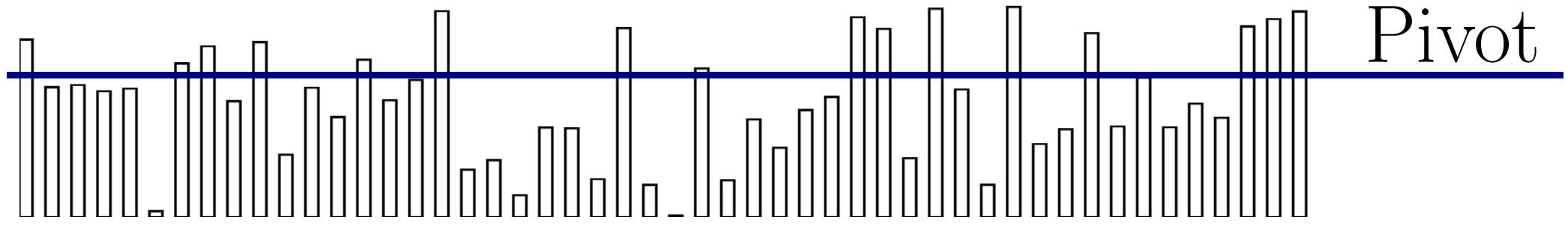


WHAT IS THE PARTITION PROBLEM?

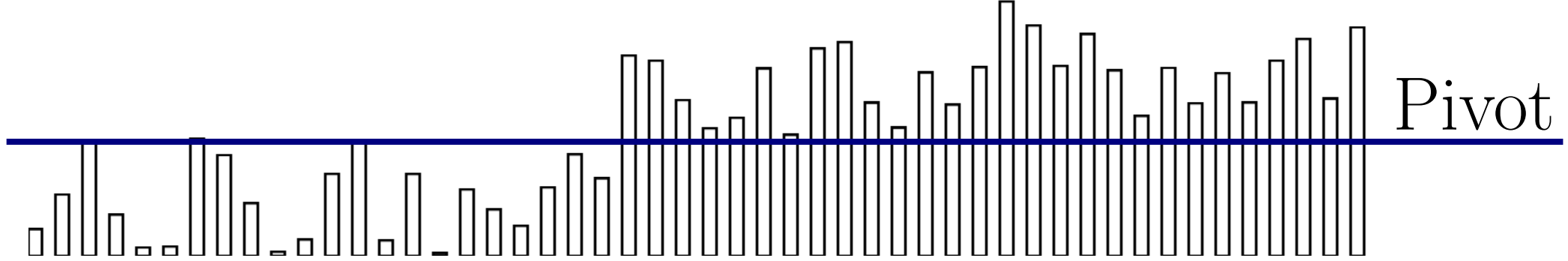
Explanation: The *Partition Problem* is to reorder the elements in a list so that elements in the same group occur in the same part of the list.

Example: A common way of grouping elements is based on whether they exceed or fall short of a certain “pivot” value.

An unpartitioned array:



An array partitioned relative to the pivot value:

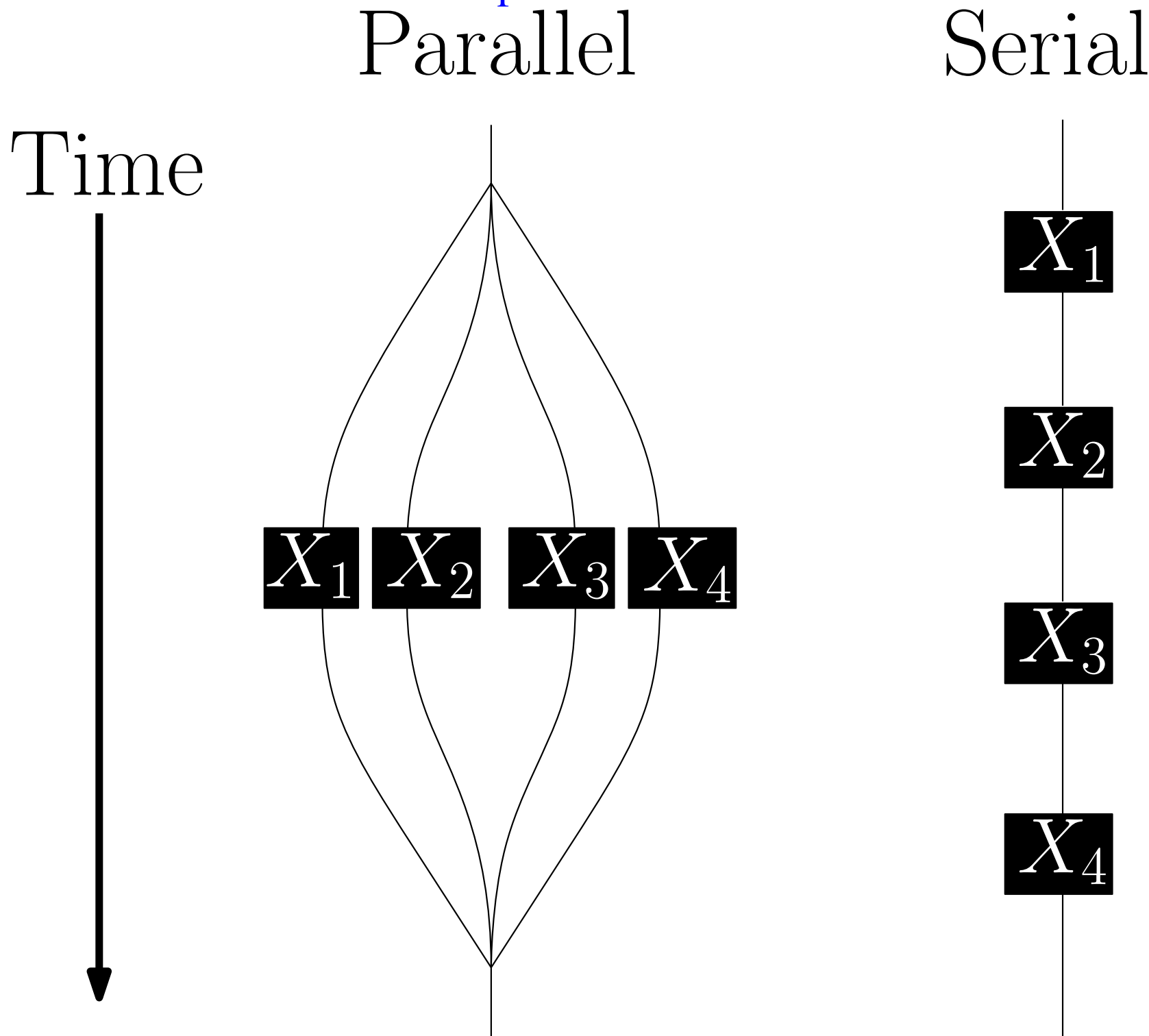


WHAT IS A PARALLEL ALGORITHM?

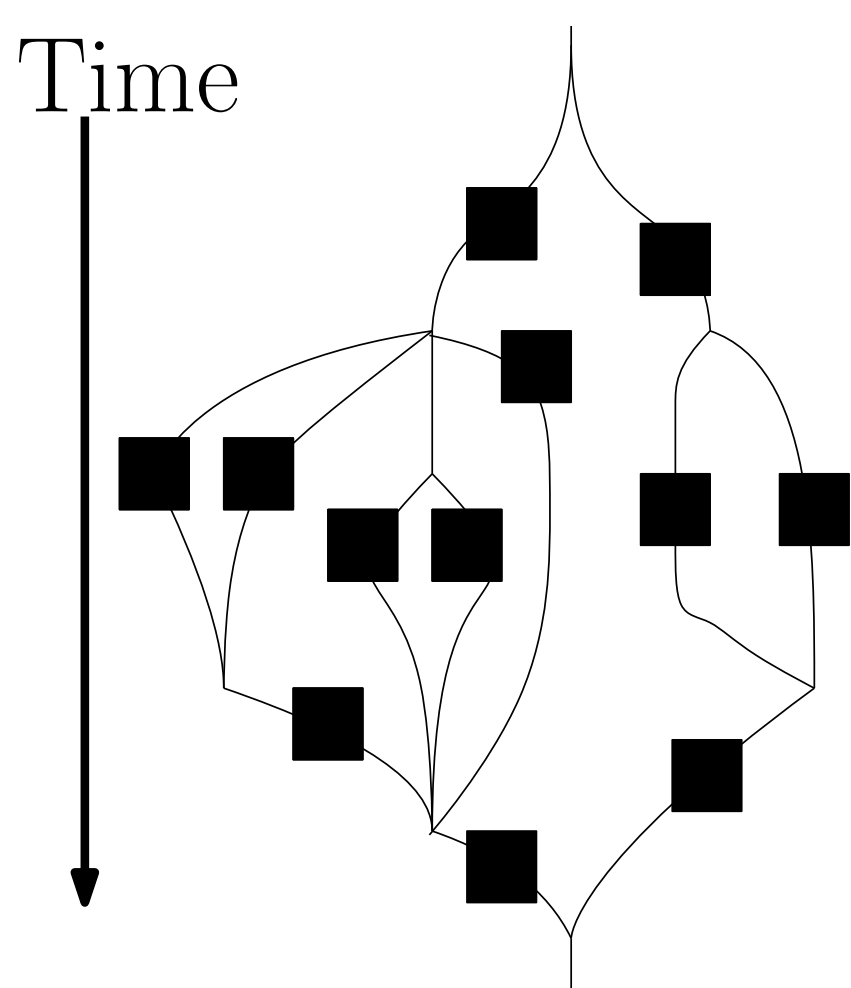
Explanation: Whereas a typical (i.e. serial) algorithm runs on a single processor, a *parallel algorithm* runs on $p \geq 1$ processors. In our model of parallelism, the only concurrency mechanism that we use is parallel-for-loops; in particular we do not use locks or atomic variables. We chose this model of parallelism because it makes our algorithms Exclusive Read Exclusive Write (EREW).

Example: Many tasks have parts that can be performed concurrently; such tasks can be performed faster with parallel computing.

Program execution in serial and in parallel



PERFORMANCE METRICS FOR PARALLEL ALGORITHMS



Important extreme cases:

Work: T_1

- ▶ time to run in serial
- ▶ ”sum of all work”

Span: T_∞

- ▶ time to run on infinitely many processors
- ▶ ”height of the graph”

WHAT IS CACHE EFFICIENCY?

Explanation: *Cache* is a small part of memory that can be accessed much faster than ordinary RAM. When data is already loaded into Cache a program can rapidly access it; this is called a *cache hit*. When data needed by a program isn’t in cache it must be loaded into cache; this is called a *cache miss*, and takes time.

Remark: An algorithm with very few cache misses is *Cache Efficient*; cache efficiency leads to faster performance in practice.

Factors in Cache-Efficiency:

- ▶ Perform low number of passes over the data
- ▶ Don’t use extra memory, i.e. are *In-Place*
- ▶ Deal with elements that are close in memory together

WHY IS THE PARTITION PROBLEM IMPORTANT?

The Partition Problem is a fundamental problem in computer science. Additionally, it is used in many algorithms such as

- ▶ Parallel Quicksort. This is the most well-known application of parallel-partition. Sorting is a very fundamental problem.
- ▶ Filtering operations.

Humans like sorted data.

Computers like sorted data.

PREVIOUS WORK ON THE PARTITION PROBLEM

The “Standard Algorithm” is **theoretically optimal with span $O(\log n)$, but slow in practice due to poor cache behavior.**

The **fastest algorithms in practice lack theoretical guarantees**

- ▶ Lock-based and atomic-variable based algorithms

[Michael Axtmann, Sascha Witt, Daniel Ferizovic, and Peter Sanders, 2017; Philip Heidelberger, Alan Norton, and John T. Robinson, 1990; Philippas Tsigas and Yi Zhang, 2003]

Not Exclusive Read/Write Memory

- ▶ The Strided Algorithm

[Francis and Pannan, 92; Frias and Petit, 08]

No locks or atomic-variables, but no bound on span