**Parallel Algorithm (GITHUB) (test)**

Parallel algorithm nedir

In computer science, a parallel algorithm, as opposed to a traditional **serial algorithm**, is an algorithm which can do multiple operations in a given time. It has been a tradition of computer science to describe serial algorithms in abstract machine models, often the one known as Random-access machine. Similarly, many computer science researchers have used a so-called parallel random-access machine (PRAM) as a parallel abstract machine (shared-memory).

Many parallel algorithms are executed concurrently – though in general concurrent algorithms are a distinct concept – and thus these concepts are often conflated, with which aspect of an algorithm is parallel and which is concurrent not being clearly distinguished. Further, non-parallel, non-concurrent algorithms are often referred to as "sequential algorithms", by contrast with concurrent algorithms.

**Parallelizability**

Algorithms vary significantly in how parallelizable they are, ranging from easily parallelizable to completely unparallelizable. Further, a given problem may accommodate different algorithms, which may be more or less parallelizable.

Some problems are easy to divide up into pieces in this way – these are called **embarrassingly parallel problems**. Examples include many algorithms to solve Rubik's Cubes and find values which result in a given hash.[citation needed]

Some problems cannot be split up into parallel portions, as they require the results from a preceding step to effectively carry on with the next step – these are called **inherently serial problems**. Examples include iterative numerical methods, such as Newton's method, iterative solutions to the three-body problem, and most of the available algorithms to compute pi (π).[citation needed] Some sequential algorithms can be converted into parallel algorithms using automatic parallelization.

**Motivation**

Parallel algorithms on individual devices have become more common since the early 2000s because of substantial improvements in multiprocessing systems and the rise of multi-core processors. Up until the end of 2004, single-core processor performance rapidly increased via frequency scaling, and thus it was easier to construct a computer with a single fast core than one with many slower cores with the same throughput, so multicore systems were of more limited use. Since 2004 however, frequency scaling hit a wall, and thus multicore systems have become more widespread, making parallel algorithms of more general use.

Issues

Communication

The cost or complexity of serial algorithms is estimated in terms of the space (memory) and time (processor cycles) that they take. Parallel algorithms need to optimize one more resource, the communication between different processors. There are two ways parallel processors communicate, shared memory or message passing.

Shared memory processing needs additional locking for the data, imposes the overhead of additional processor and bus cycles, and also serializes some portion of the algorithm.

Message passing processing uses channels and message boxes but this communication adds transfer overhead on the bus, additional memory need for queues and message boxes and latency in the messages. Designs of parallel processors use special buses like crossbar so that the communication overhead will be small but it is the parallel algorithm that decides the volume of the traffic.

If the communication overhead of additional processors outweighs the benefit of adding another processor, one encounters parallel slowdown.

**Load balancing**

Another problem with parallel algorithms is ensuring that they are suitably load balanced, by ensuring that load (overall work) is balanced, rather than input size being balanced. For example, checking all numbers from one to a hundred thousand for primality is easy to split amongst processors; however, if the numbers are simply divided out evenly (1–1,000, 1,001–2,000, etc.), the amount of work will be unbalanced, as smaller numbers are easier to process by this algorithm (easier to test for primality), and thus some processors will get more work to do than the others, which will sit idle until the loaded processors complete.

**Distributed algorithms**

A subtype of parallel algorithms, distributed algorithms are algorithms designed to work in cluster computing and distributed computing environments, where additional concerns beyond the scope of "classical" parallel algorithms need to be addressed.

-

-

-

-