**Introduction to PLINQ**

## **What is a Parallel Query?**

Parallel LINQ (PLINQ) is a parallel implementation of the LINQ pattern. PLINQ queries, just like sequential LINQ queries, operate on any in-memory [IEnumerable](https://docs.microsoft.com/en-us/dotnet/api/system.collections.ienumerable) or [IEnumerable<T>](https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.ienumerable-1) data source, and have deferred execution, which means they do not begin executing until the query is enumerated. The primary difference is that PLINQ attempts to make full use of all the processors on the system. It does this by partitioning the data source into segments, and then executing the query on each segment on separate worker threads in parallel on multiple processors. In many cases, parallel execution means that the query runs significantly faster.

PLINQ can achieve significant performance improvements over legacy code for certain kinds of queries, often just by adding the [AsParallel](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable.asparallel) query operation to the data source. However, parallelism can introduce its own complexities, and not all query operations run faster in PLINQ. In fact, parallelization actually slows down certain queries.

## **The ParallelEnumerable Class**

In addition to the standard query operators, the [ParallelEnumerable](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable) class contains a set of methods that enable behaviors specific to parallel execution (AsParallal, AsSequential, AsOrdered).

## **Execution Modes**

At run time, the PLINQ infrastructure analyzes the overall structure of the query. If the query is likely to yield speedups by parallelization, PLINQ partitions the source sequence into tasks that can be run concurrently. If it is not safe to parallelize a query, PLINQ just runs the query sequentially. If PLINQ has a choice between a potentially expensive parallel algorithm or an inexpensive sequential algorithm, it chooses the sequential algorithm by default.

## **Degree of Parallelism**

PLINQ uses all of the processors on the host computer. You can instruct PLINQ to use no more than a specified number of processors by using the [WithDegreeOfParallelism](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable.withdegreeofparallelism) method.

var query = from item in source.AsParallel().WithDegreeOfParallelism(2)

where Compute(item) > 42

select item;

## **Ordered Versus Unordered Parallel Queries**

PLINQ provides the [AsOrdered](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable.asordered) operator. An [AsOrdered](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable.asordered) sequence is still processed in parallel, but its results are buffered and sorted. Because order preservation typically involves extra work, an [AsOrdered](https://docs.microsoft.com/en-us/dotnet/api/system.linq.parallelenumerable.asordered) sequence might be processed more slowly than the default AsUnordered sequence.

var evenNums = from num in numbers.AsParallel().AsOrdered()

where num % 2 == 0

select num;