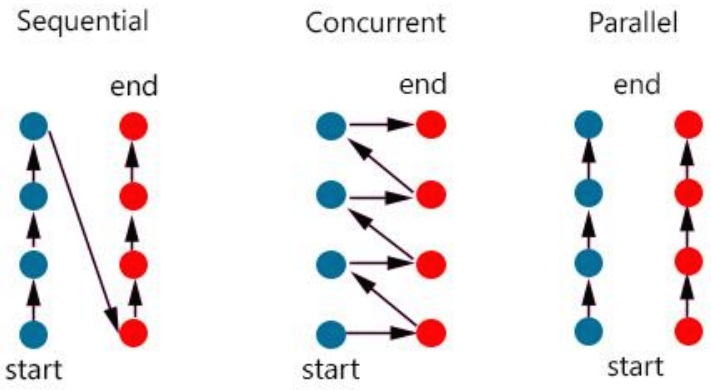
[아이템 48] 스트림 병렬화는 주의해서 사용하라

태그	6주차
장	7장 : 람다와 스트림

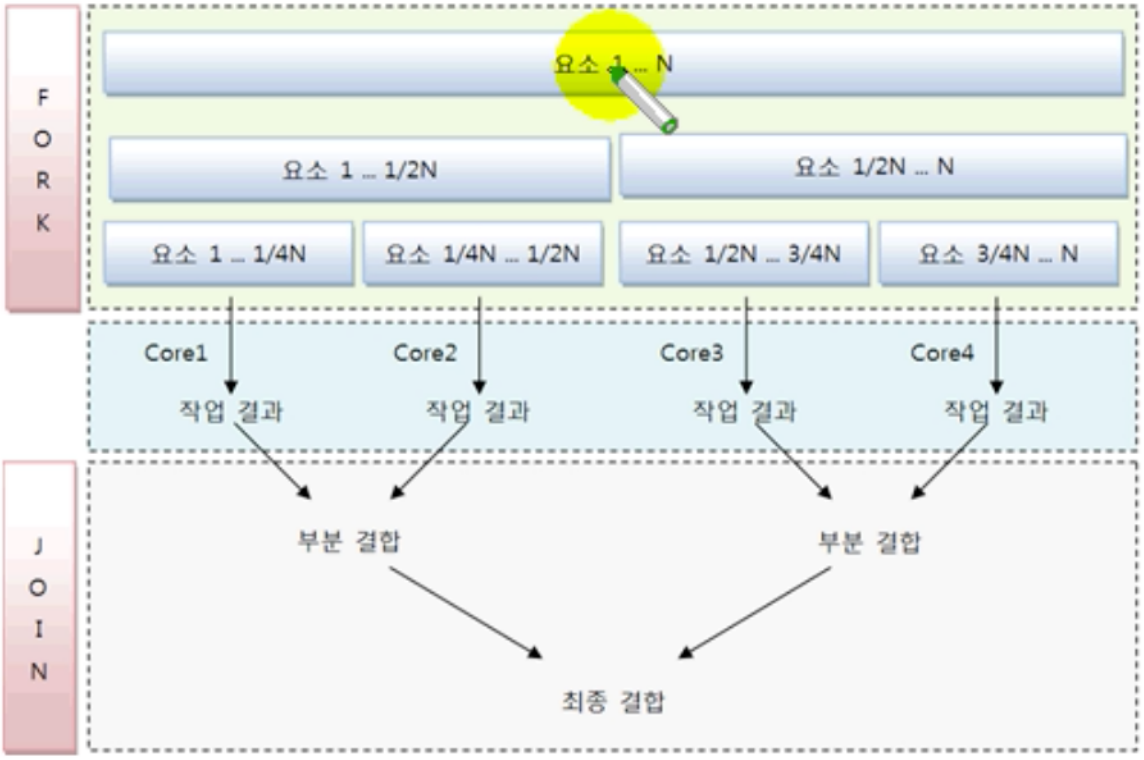
병렬 처리 (Parallel Operation)




```
동시성(Concurrency)과 병렬성(Parallelism)
동시성: 멀티 스레드 환경에서 스레드가 번갈아 가며 실행하는 성질 (싱글 코어 CPU)
병렬성: 멀티 스레드 환경에서 코어들이 스레드를 병렬적으로 실행하는 성질(멀티 코어 CPU)
```

자바는 초기 릴리즈부터 동시성에 대한 여러가지 고려가 이루어졌던 것으로 보입니다. 버전을 거듭하며 지속적으로 개선하고 있습니다.

- release : 스레드, 동기화, wait/notify
- java 5 : 동시성 컬렉션 java.util.concurrent 실행자 프레임워크(Executor)
  - 동시성 컬렉션은 여러 스레드가 안전하게 사용할 수 있는 컬렉션 클래스를 제공합니다. (`ConcurrentHashMap`, `CopyOnWriteArrayList` 등)
  - `Executor` 프레임워크는 스레드 풀을 생성하고 관리하며, 작업을 비동기적으로 실행할 때 유용합니다.
- java 7 : 고성능 병렬 분해 프레임워크 ForkJoin 패키지



    - 포크 단계
        - 데이터를 서브 데이터로 반복적으로 분리한다.
        - 서브 데이터를 멀티 코어에서 병렬로 처리한다.
    - 조인 단계
        - 서브 결과를 결합해서 최종 결과를 만들어 낸다.
        - 실제로 병렬 처리 스트림은 포크 단계에서 내부적으로 서브 요소로 나누는 알고리즘이 있기 때문에 개발자는 신경쓸 필요가 없다.
- java 8 : 병렬 Stream
    - Java의 스트림은 단순히 데이터 소스를 감싸는 래퍼이므로 편리한 방식으로 데이터에 대한 대량 작업을 수행할 수 있습니다.
    - 데이터를 저장하거나 기본 데이터 소스를 변경하지 않습니다. 오히려 데이터 파이프라인에서 기능 스타일 작업에 대한 지원을 추가합니다.
    - 멀티코어의 수만큼 대용량 요소를 서브 요소들로 나누고, 각각의 서브 요소들을 분리된 스레드에서 병렬 처리시킨다.
    - 예를 들어 쿼드 코어(Quad Core) CPU일 경우 4개의 서브 요소들로 나누고, 4개의 스레드가 각각의 서브 요소들을 병렬 처리한다.
    - 병렬 스트림은 내부적으로 포크조인 프레임워크를 이용한다.

병렬 처리 성능병렬 처리는 항상 빠르다?

스트림 병렬 처리가 스트림 순차 처리보다 항상 실행 성능이 좋다고 판단해서는 안됩니다.

병렬 처리에 영향을 미치는 3가지 요인

오버헤드 : 요소의 수와 요소당 처리 시간

컬렉션에 요소의 수가 적고 요소당 처리 시간이 짧으면 순차 처리가 오히려 병렬 처리보다 빠를 수 있습니다.

```
IntStream.rangeClosed(1, 100).reduce(0, Integer::sum);
IntStream.rangeClosed(1, 100).parallel().reduce(0, Integer::sum);
```

간단한 합계 감소에서 순차 스트림을 병렬 스트림으로 변환하면 성능이 저하됩니다.

```
Benchmark                                                  Mode  Cnt        Score        Error  Units
SplittingCosts.sourceSplittingIntStreamParallel            avgt   25      35476,283 ±     204,446  ns/op
SplittingCosts.sourceSplittingIntStreamSequential          avgt   25         68,274 ±       0,963  ns/op
```

병렬 처리는 스레드풀 생성, 스레드 생성이라는 추가적인 비용이 발생하기 때문에 실제 작업을 수행하는 것보다 비용이 많이 들게 되기 때문입니다.

스트림 소스의 종류

데이터 소스를 균등하게 분할하는 것은 병렬 실행을 활성화하는 데 필요한 비용이지만 일부 데이터 소스는 다른 데이터 소스보다 더 잘 분할됩니다.

- ArrayList, 배열은 랜덤 액세스를 지원(인덱스로 접근)하기 때문에 포크 단계에서 쉽게 요소를 분리할 수 있어 병렬 처리 시간이 절약됩니다.
- 반면에 HashSet, TreeSet은 요소를 분리하기가 쉽지 않고, LinkedList는 랜덤 액세스를 지원하지 않아 링크를 따라가야 하므로 역시 요소를 분리하기가 쉽지 않습니다.

```
private static final List<Integer> arrayListOfNumbers = new ArrayList<>();
private static final List<Integer> linkedListOfNumbers = new LinkedList<>();

..

arrayListOfNumbers.stream().reduce(0, Integer::sum)
arrayListOfNumbers.parallelStream().reduce(0, Integer::sum);
linkedListOfNumbers.stream().reduce(0, Integer::sum);
linkedListOfNumbers.parallelStream().reduce(0, Integer::sum);
```

```
Benchmark                                                  Mode  Cnt        Score        Error  Units
DifferentSourceSplitting.differentSourceArrayListParallel     avgt   25    2004849,711 ±    5289,437  ns/op
DifferentSourceSplitting.differentSourceArrayListSequential   avgt   25    5437923,224 ±   37398,940  ns/op
DifferentSourceSplitting.differentSourceLinkedListParallel    avgt   25   13561609,611 ±  275658,633  ns/op
DifferentSourceSplitting.differentSourceLinkedListSequential  avgt   25   10664918,132 ±  254251,184  ns/op
```

또한 BufferedReader.lines()은 전체 요소의 수를 알기 어렵기 때문에 포크 단계에서 부분요소로 나누기 어렵습니다. 따라서 이들 소스들은 ArrayList, 배열 보다는 상대적으로 병렬 처리가 늦습니다.

병렬 처리는 프로세서 코어가 유용한 작업을 계속 바쁘게 할 수 있을 때 성능 이점을 가져옵니다. 캐시 미스를 기다리는 것은 유용한 작업이 아니기 때문에 메모리 대역폭을 제한 요소로 고려해야 합니다.

일반적으로 **데이터 구조에 포인터가 많을수록** 참조 개체를 가져오기 위해 메모리에 더 많은 압력을 가합니다. 여러 코어가 동시에 메모리에서 데이터를 가져오기 때문에 병렬화에 부정적인 영향을 미칠 수 있습니다.

코어(Core)의 수

싱글 코어 CPU일 경우에는 순차 처리가 빠릅니다.

병렬 처리를 할 경우 스레드의 수만 증가하고 번갈아 가면서 스케쥴링을 해야하므로 좋지 못한 결과를 준다고. 코어의 수가 많으면 많을 수록 병렬 작업 처리 속도는 빨라집니다.

병렬 스트림을 사용하는 경우

병렬 스트림을 사용할 때는 매우 신중해야 합니다. 오직 **최적화 수단**임을 기억하고 변경 전후로 반드시 성능을 테스트하여 사용할 가치가 있는지 따져야 한다는 것!

병렬 스트림은 마법 같은 성능 향상 장치로 간주될 수 없습니다. 따라서 **개발 중에는 순차 스트림을 계속 기본으로 사용해야 합니다.**

실제 성능 요구 사항이 있을 때 순차 스트림을 병렬 스트림으로 변환할 수 있습니다 . 이러한 요구 사항이 주어지면 먼저 성능 측정을 실행하고 가능한 최적화 전략으로 병렬 처리를 고려해야 할 것 같습니다.

많은 양의 데이터와 요소당 수행되는 많은 계산은 병렬 처리가 좋은 옵션이 될 수 있습니다.

반면에 적은 양의 데이터, 고르지 않은 소스 분할, 값비싼 병합 작업 및 열악한 메모리 지역성은 병렬 실행에 대한 잠재적인 문제가 있을 수 있습니다.