Fork/Join Framework

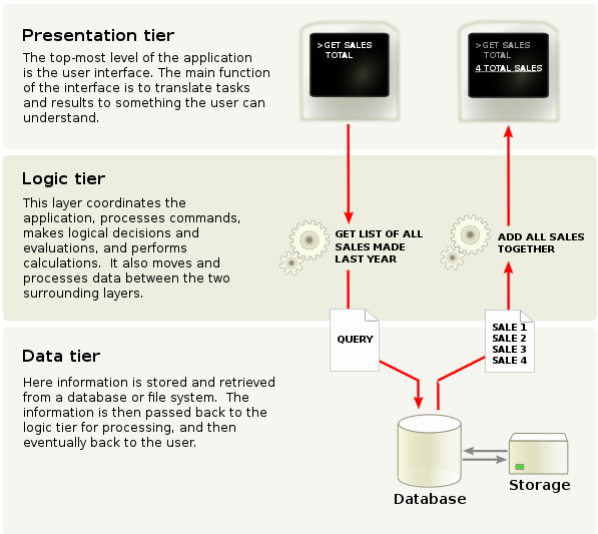
sanghyuck.na@lge.com

CounterCompleter

1

Multithread programming

- Data processing
  - Stream
- Service daemon
  - Fork/join framework



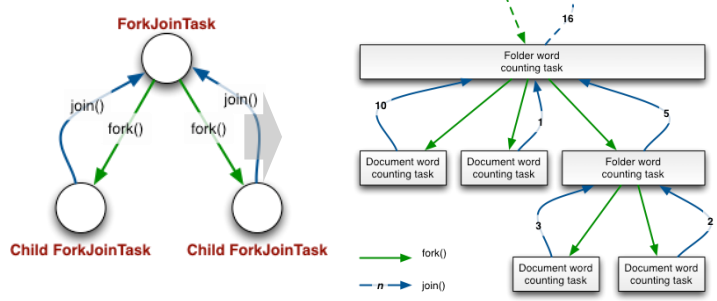
https://goo.gl/GWvgkF Business logic

2

Fork/Join Framework

- ExecutorSevice 구현체 Thread pool JAVA7
  - Multiple processor를 쉽게 사용을 지원하는 도구로써 작업(Work)를 적절한 작은 단위로 분할, 처리하여 완료하는(divide and conquer) Split → fork → join 메커니즘동작
  - ForkJoinPool7은 스레드풀에 작업(Work)을 Worker threads에 배포

```
ForkJoinPool p = ForkJoinPool.commonPool();
ForkJoinPool p = new ForkJoinPool9(2);
```

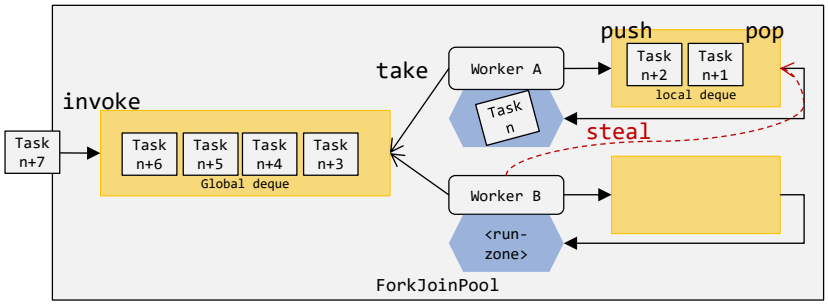


3

Fork/Join Framework

- Worker thread
  - Work thread는 한 시점에 단 1개 task만 실행. 단, Task별 Worker를 만들지 않음.
- Work stealing algorithm
  - 자신 큐가 빈 경우 다른 작업중인 Work thread의 큐에서 task를 가져옵니다(steal). 만약 없다면 global 테일 또는 글로벌 엔트리 큐에서 작업을 수행합니다. 가장 큰 작업이 위치했을 가능성이 있기 때문입니다.

```
ForkJoinPool.commonPool().invoke(atask);
Integer rt = ForkJoinPool.commonPool().invoke(rtask);
```



http://bit.ly/32txVXd Work Stealing

4

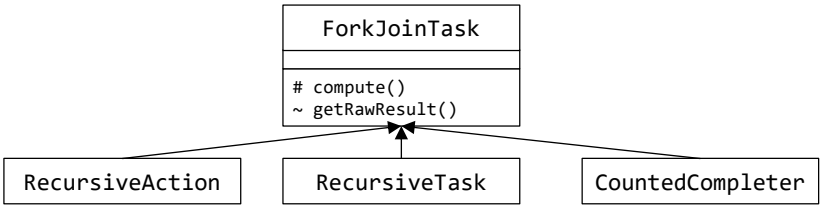
Fork/Join Task

- ForkJoinPool의 Basic task type
  - RecurvsiveAction[7]: 최종결과가 리턴값이 없는 task
  - RecursiveTask[7]: 최종결과가 리턴값이 있는 task
  - CountedCompleter[8]: 실행중인 child task실행개수를 파악할 수 있는 task

```
class CustomRecursiveAction extends RecursiveAction {

class CustomRecursiveTask extends RecursiveTask<Integer> {

class CusttomCountedTask extends CountedCompleter<Integer> {
```



5

CountedCompleter

- compute() 실행 후 후처리 기회제공
  - compute() 메소드 종료직전에 Callback onCompletion()의 실행여부를 Trigger API호출로 제어할 수 있습니다. Trigger API에 따라 onCompletion()는 호출되거나 안됩니다.
  - tryComplete()는 onCallback을 호출되게 합니다. propagateCompletion()는 그렇지 않습니다. 이 두 Target API 호출 시 내부 Pending Count는 감소시킵니다.
- Pending tasks
  - 새로운 sub-task를 실행하면 내부적으로 Pending count를 증가합니다. 이 counter로 실행 종료를 식별합니다. setPendingCount() addPendingCount()가 있습니다.

```
CountedCompleter(CountedCompleter<?> completer)
CountedCompleter(CountedCompleter<?> completer, int initialPendingCount)

void compute()

void tryComplete(), void propagateCompletion()
void setPendingCount(int count), void addToPendingCount(int delta)

void onCompletion(CountedCompleter<?> caller)
int getPendingCount()
```

6

CountedCompleter forEach

- tryComplete()
  - 이 함수 호출후 대기타스크갯수가 0이 아니라면 개수만 차감 되고, 그렇지 않고 0이면 onCompletion(CountedCompleter)이 호출됩니다. 그 이후 "종료"로 표시됩니다.
- propagateCompletion()
  - tryComplete()와 동일하지만, onCompletion(CountedCompleter)은 호출되지 않습니다.
- addToPendingCount(int delta)
  - 주어진 Delta값을 대기타스크갯수에 더합니다
- setPendingCount(int count)
  - 주어진 Count값을 대기타스크갯수에 설정합니다.

```
var cnt = new AtomicInteger(0);
var array = new Random().ints(0, 100)
    .limit(100).boxed().toArray(Integer[]::new);
Consumer<Integer> myop = (i) -> cnt.incrementAndGet();

Main.forEach(array, myop);
System.out.println("testCountedCompleter=" + cnt.get());
```

```
static <E> void forEach(E[] array, Consumer<E> action)
```

7

Stream forEach

```
Stream.of(0).parallel()
    .forEachOrdered(e->Thread.dumpStack());
```

```
static final class ForEachTask<S, T> extends CountedCompleter<Void> {

    public void compute() {
        Spliterator<S> rightSplit = spliterator, leftSplit;
        …
        while (…) {
            if (… ||(leftSplit = rightSplit.trySplit()) == null) {
                …
                break;
            }
            ForEachTask<S, T> leftTask;
            task.addToPendingCount(1);
            taskToFork.fork();
        …
        }

        task.propagateCompletion();
```

	compute()
sizeThreshold	데이터 최소 분할크기
sizeEstimate	현재 분할 크기
rightSplit	Right 분할정보
leftSplit	Left 분할정보
taskToFork	새로운Task
task	새로운Task

9

정리

- Fork And Join framework
- CountedCompleter

10