СПб АУ НОЦНТ РАН

Java 02

02.03.2016

Fork-Join

```
@Test
public void testSafePublicationLazy() {
    checkSingleThreadContract(LazyFactory::createLazyLockFre
    checkMultiThreadContract(LazyFactory::createLazyLockFree
}
```

Fork-Join

```
private void checkSingleThreadContract(
  Function<Supplier<?>, Lazy<?>> factory
) {
    checkSingleThreadWithGivenSupplier(
          factory, Object::new);
    checkSingleThreadWithGivenSupplier(
          factory, () -> null);
}
```

Fork-Join

```
private void checkSingleThreadWithGivenSupplier(
  Function<Supplier<?>, Lazy<?>> factory,
  Supplier<?> supplier
) {
    TestSupplier<?> testSupplier =
          new TestSupplier<>(supplier);
    Lazy<?> lazy = factory.apply(testSupplier);

    assertFalse(testSupplier.isCalled());
    assertSame(
          testSupplier.getFirstResult(),
          lazy.get());
    assertTrue(testSupplier.isCalled());
    assertSame(
          testSupplier.getFirstResult(),
          lazy.get());
}
```

Fork-Join

```
private void checkSingleThreadWithGivenSupplier(
  Function<Supplier<?>, Lazy<?>> factory,
  Supplier<?> supplier
) {
    TestSupplier<?> testSupplier =
          new TestSupplier<>(supplier);
    Lazy<?> lazy = factory.apply(testSupplier);

    assertFalse(testSupplier.isCalled());
    assertSame(
          testSupplier.getFirstResult(),
          lazy.get());
    assertTrue(testSupplier.isCalled());
    assertSame(
          testSupplier.getFirstResult(),
          lazy.get());
}
```

Fork-Join

```
class TestSupplier<T> implements Supplier<T> {
    final T firstResult;
    final Supplier<T> supplier;
    boolean isCalled;
    TestSupplier(Supplier<T> supplier) {
        this.firstResult = supplier.get();
        this.supplier = supplier;
    }
    public synchronized T get() {
        if (!isCalled) {
            isCalled = true;
            return firstResult;
        }

        return supplier.get();
    }
}
```

Fork-Join

```
void mergeSort(int begin, int end) {
  int m = (begin + end) / 2;

  // recursive subtasks
  mergeSort(begin, m);
  mergeSort(m, end);

  // join result
  ...
}
```

Fork-Join. Implementation of RecursiveTask<T>.compute

```
Integer compute() {
    if (isLeaf()) return 1;
    List<MyTask> subTasks = new LinkedList<>();
    for (TaskData subTaskData : getSubTasks()) {
        MyTask task = new MyTask(subTaskData);
        task.fork();
        subTasks.add(task);
    }

    int result = 0;
    for(MyTask task : subTasks) {
        result += task.join();
    }
    return result;
}
```

Fork-Join. Run

```
import java.util.concurrent.ForkJoinPool;

public class Main {
    public static void main(String[] args) {
        System.out.println(
            new ForkJoinPool().invoke(new MyTask())
        );
    }
}
```

Задание

- Консольное приложение, вычисляющее check-сумму директории ФС
- $f(file) = MD5(< filecontent >)$
- $f(dir) = MD5(< dirname > + f(file1) + ..)$

1. Однопоточный вариант
2. Вариант с использованием обычных ExecutorService/Future
3. Вариант с Fork-Join

Сравните время работы этих вариантов
NB:

- Используйте DigestInputStream/Paths и внешние библиотеки
- Для этой работы также создаем PR