1.2 Lecture Summary

1 Task-level Parallelism

1.2 Creating Tasks in Java's Fork/Join Framework

Lecture Summary: In this lecture, we learned how to implement the *async* and *finish* functionality using Java's standard Fork/Join (FJ) framework. In this framework, a task can be specified in the `compute()` method of a user-defined class that extends the standard RecursiveAction class in the FJ framework. In our Array Sum example, we created class `ASum` with fields `A` for the input array, `LO` and `HI` for the subrange for which the sum is to be computed, and `SUM` for the result for that subrange. For an instance of this user-defined class (e.g., `L` in the lecture), we learned that the method call, `L.fork()`, creates a new task that executes `L`'s `compute()` method. This implements the functionality of the *async* construct that we learned earlier. The call to `L.join()` then waits until the computation created by `L.fork()` has completed. Note that `join()` is a lower-level primitive than *finish* because `join()` waits for a specific task, whereas *finish* implicitly waits for all tasks created in its scope. To implement the *finish* construct using `join()` operations, you have to be sure to call `join()` on every task created in the finish scope.

A sketch of the Java code for the ASum class is as follows:

```
1  private static class ASum extends RecursiveAction {
2    int[] A; // input array
3    int LO, HI; // subrange
4    int SUM; // return value
5    . . .
6    @Override
7    protected void compute() {
8      SUM = 0;
9      for (int i = LO; i <= HI; i++) SUM += A[i];
10   } // compute()
11 }
```

FJ tasks are executed in a ForkJoinPool, which is a pool of Java threads. This pool supports the invokeAll() method that combines both the `fork` and `join` operations by executing a set of tasks in parallel, and waiting for their completion. For example, `ForkJoinTask.invokeAll(left,right)` implicitly performs `fork()` operations on `left` and `right`, followed by `join()` operations on both objects.

Optional Reading:

1. Tutorial on Java's Fork/Join framework