1 Task-level Parallelism

1.1 Task Creation and Termination (Async, Finish)

Lecture Summary: In this lecture, we learned the concepts of *task creation* and *task termination* in parallel programs, using array-sum as an illustrative example. We learned the *async* notation for task creation: "*async ⟨stmt1⟩*", causes the parent task (*i.e.*, the task executing the async statement) to create a new child task to execute the body of the *async, ⟨stmt1⟩, asynchronously* (*i.e.*, before, after, or in parallel) with the remainder of the parent task. We also learned the *finish* notation for task termination: "*finish ⟨stmt2⟩*" causes the parent task to execute *⟨stmt2⟩*, and then wait until *⟨stmt2⟩* and all *async* tasks created within *⟨stmt2⟩* have completed. Async and finish constructs may be arbitrarily nested.

The example studied in the lecture can be abstracted by the following pseudocode:

```
1  finish {
2    async S1; // asynchronously compute sum of the lower half of the array
3    S2;       // compute sum of the upper half of the array in parallel with S1
4  }
5  S3; // combine the two partial sums after both S1 and S2 have finished
```

While *async* and *finish* notations are useful algorithmic/pseudocode notations, we also provide you access to a high-level open-source Java-8 library called PCDP (for Parallel, Concurrent, and Distributed Programming), for which the source code is available at https://github.com/habanero-rice/pcdp. PCDP contains APIs (application programming interfaces) that directly support *async* and *finish* constructs so that you can use them in real code as well. In the next lecture, you will learn how to to implement the *async* and *finish* functionality using Java's standard Fork/Join (FJ) framework.

Optional Reading:

1. Wikipedia article on Asynchronous method invocation

Пометить как выполненное