# Discussion questions for *The Implementation of the Cilk-5 Multithreaded Language*

N. Ramsey and N. Cooper, for COMP 250RTS

October 11, 2017

## Why the system performs

Given sufficiently low overheads, there is an "embarrassment of parallelism," and the number of attempted thefts is small. The system is therefore designed so that expensive operations are limited by the number of thefts.

(1) This question looks at operations and their costs.

   (a) What operations performed by workers are *not* bounded by the number of thefts, and must therefore be cheap? What are their costs?

   (b) What operations performed by workers *are* bounded by the number of thefts, and can therefore be (relatively) expensive?

   (c) What operations are performed by thieves? Are these operations significantly more expensive than the operations in the previous question?

   (d) If the operations in parts (b) and (c) get too expensive, what goes wrong? (*Hint*: Are any of the opening assumptions of section 3 violated?)

## Two call stacks

Cilk uses standard C call stacks, and as described in section 5, the worker uses the ready deque acts as a kind of "shadow stack."

(2) This question explores the call stacks by asking you to fill in information that is not given explicitly in the paper.

   (a) When a Cilk program executes a `spawn` primitive, where is the C call stack on which the spawned code executes?

   (b) Lines 17–18 of Figure 3 trigger an unwinding of the C call stack. What do you imagine is the oldest frame on this stack, and when it detects `FAILURE`, what do you suppose it does?

   (c) What call stacks are there in the system? What are their lifetimes? (That is, when and how are they born? When and how do they die?)

   How does this implementation differ from Cormack's implementation?

   (d) In Figure 3, in the second spawn, which is elided (line 19), what frame do you expect to be pushed on the shadow stack, and why?

## Programming model

Cilk provides superb parallel performance. Cormack provides concurrency without parallelism.

(3) Compare Cilk with Cormack's micro-kernel. Use these dimensions:

   (a) Each system provides mechanisms for ensuring the atomicity of system primitives. Compare the two.

   (b) Each system provides mechanisms that programmers can use to ensure atomicity of their own code fragments. Compare the two.

   (c) Relative to Cormack, is Cilk giving up anything? If so, what and why?

   (d) At a high level, how would you characterize the programming model that each system offers to the programmer?

## Lasting value

(4) In 2008, a selection committee selected this paper as the most influential paper published in PLDI 1998. What do you conjecture were perceived as the contributions of lasting value?