Thievery:
A Cilk Hacker's Guide to the Runtime System

Chris Zakian
Indiana University
czakian@indiana.edu

Abstract

The Cilk runtime system is complex software article. An examination of the runtime system is not a small endeavor and requires the reader to attempt to decipher the myriad lines of uncommented and confusing code contained in the scheduler and its supporting files. This paper is meant to be a guide to the most confusing portion of the Cilk runtime - the work stealing and syncing mechanism. The reader should have a basic knowledge of how to program with Cilk and have read the Cilk 5 Implementation paper [1] which discusses the overview of Cilk in detail. The examination of the Cilk work stealing system will be done by an iterative deepening approach. First it will investigate work stealing on a non-nested program, and then investigate the mechanism deeper as it applies to nested spawns. Finally, the mechanism for sync workers after work has been stolen will be addressed.

1. A Simple Program

Consider the the program in figure 2: it enters at main, grows the stack by three C stack frames before returning to main and exiting.

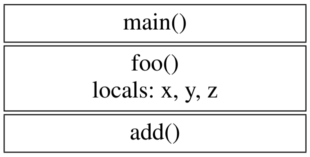

Figure 1. Call Stack of Figure 2.

```
#include <stdio.h>

int add(int x, int y)
{
  return x + y;
}

void foo(void)
{
  int x;
  int y;
  int z;

  x = add(5, 6);
  y = add(7, 8);

  z = add(x, y);

  printf("result:␣%d\n", z);
}

int main()
{
  foo();
  return 0;
}
```

Figure 2. A Simple Serial Program

In contrast, figure 3 shows how the same code as figure 2 is parallelized using Cilk. The

```
#include <stdio.h>

int add(int x, int y)
{
  return x + y;
}

void foo(void)
{
  int x;
  int y;
  int z;

  x = cilk_spawn add(5, 6);
  y = cilk_spawn add(7, 8);
  cilk_sync;

  z = add(x, y);
  printf("result:␣%d\n", z);
}

int main()
{
  foo();
  return 0;
}
```

Figure 3. Parallelized With Cilk

References

[1] M. Frigo, C. E. Leiserson, and K. H. Randall. The implementation of the cilk-5 multithreaded language. *SIGPLAN Not.*, 33(5):212–223, May 1998.