Optimistic Threading Techniques for MPI+ULT

Shintaro Iwasaki†‡ Abdelhalim Amer†‡ Kenjiro Taura†‡ Pavan Balaji‡‡
† The University of Tokyo, ‡ Argonne National Laboratory

Introduction

MPI+Thread

- Hybrid parallelism is common.
- e.g., MPI+OpenMP.
- OS-level threads (e.g., pthread) are heavy to exploit fine-grained parallelism.

MPI+ULT (User-level thread) [1]

- Threading overheads are small.
- Efficiently overlap communications and computations.
- Avoid internal locks in MPI runtime systems by lightweight context switches.

Problem: Cost of “Suspendability”

- Threads must be created as “suspendable” (Full).
- Unsuspendable threads (RtC) are known to be more lightweight.

Few threads suspend in many cases.
- Not all threads call MPI functions.
- Suspension is unnecessary when locks are successfully acquired.

We need to create Full ULTs if it might call MPI functions, imposing overheads.
- Can reduce overheads when we know most threads never suspend?

Fork-Join Overheads

Not suspend vs suspend

- Implement them in Argobots [2]
- SA is faster than Full when suspension rate < ~60%.
- RtC is fastest when suspension rate = 0, while RtC cannot suspend.

Summary of threading techniques

- Suspendability
- Few runtime systems (Chores[3]) adopted this in the past.

Fork-Join Overheads

Optimistic case

- # of ULTs which over suspend
- # of All ULTs

How do they work? (no-suspension case)

Full:

- Scheduler’s stack
- ULT’s stack
- SA: Suspension

RtC:

- Scheduler’s stack
- ULT’s stack
- SA: Suspension

Optimistic case

- Larger is better

How do they work? (no-suspension case)

Full:

- Scheduler’s stack
- ULT’s stack
- Call

RtC:

- Scheduler’s stack
- ULT’s stack
- SA: Suspension

Future Work

- Alleviate constraints of SA.
- Compare performance with the existing techniques (i.e., OpenMP).

Acknowledgment

This work is supported by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357.

References