COMP 322: Fundamentals of Parallel Programming

https://wiki.rice.edu/confluence/display/PARPROG/COMP322

Lecture 15: Point-to-point Synchronization, Pipeline Parallelism, Phasers (contd)

Vivek Sarkar
Department of Computer Science
Rice University
vsarkar@rice.edu
Announcements

• Homework 4 due by 5pm on Wednesday, Feb 16th
  — We will try and return graded homeworks by Feb 23rd

• Guest lecture on Bitonic Sort by John Mellor-Crummey on Friday, Feb 18th

• Feb 23rd lecture will be a Midterm Review

• No lecture on Friday, Feb 25th since midterm is due that day
  — Midterm will be a 2-hour take-home written exam
    - Closed-book, closed-notes, closed-computer
  — Will be given out at lecture on Wed, Feb 23rd
  — Must be handed in by 5pm on Friday, Feb 25th
Acknowledgments for Today’s Lecture


• Handout for Lectures 14 and 15
Point-to-Point Synchronization: Example 1 (Left-Right Neighbor Synchronization)

1. \texttt{finish} \{ // Expanded finish-for-async version of forall
2. \quad \texttt{for} (\texttt{point[i]} : [1:m])
3. \quad \texttt{async} \{
4. \quad \quad \texttt{doPhase1}(i);
5. \quad \quad \quad \texttt{// Iteration i waits for \(i-1\) and \(i+1\) to complete Phase 1}
6. \quad \quad \texttt{doPhase2}(i);
7. \quad \}
8. \}

• Need synchronization where iteration \(i\) only waits for iterations \(i-1\) and \(i+1\) to complete their work in \texttt{doPhase1}() before it starts \texttt{doPhase2}(i)? (Less constrained than a barrier)

\begin{align*}
\texttt{doPhase1}(i) & : & i=1 & i=2 & i=3 & i=4 & i=5 & i=6 & i=7 & i=8 \\
\texttt{doPhase2}(i) & : & & & & & & & & \\
\end{align*}
Summary of Phaser Construct

- **Phaser allocation**
  - `phaser ph = new phaser(mode);`
  - Phaser ph is allocated with registration mode
  - *Phaser lifetime is limited to scope of Immediately Enclosing Finish (IEF)*

- **Registration Modes**
  - `phaserMode.SIG`
  - `phaserMode.WAIT`
  - `phaserMode.SIG_WAIT`
  - `phaserMode.SIG_WAIT_SINGLE`

- **Phaser registration**
  - `async phased (ph1<mode1>, ph2<mode2>, ... ) <stmt>`
  - Spawned task is registered with `ph1` in `mode1`, `ph2` in `mode2`, ...
  - *Child task’s capabilities must be subset of parent’s*
  - `async phased <stmt>` propagates all of parent’s phaser registrations to child

- **Synchronization**
  - `next;`
  - Advance each phaser that current task is registered on to its next phase
  - Semantics depends on registration mode
At any point in time, a task can be registered in one of four modes with respect to a phaser: SIG_WAIT_SINGLE, SIG_WAIT, SIG, or WAIT. The mode defines the set of capabilities — signal, wait, single — that the task has with respect to the phaser. The subset relationship defines a natural hierarchy of the registration modes.
Semantics of next depends on registration mode

SIG_WAIT: next = signal + wait
SIG: next = signal (Don’t wait for any task)
WAIT: next = wait (Don’t disturb any task)

A master task receives all signals and broadcasts a barrier completion
Left-Right Neighbor Synchronization
Example for $m=3$ using Phasers

```java
finish {
    phaser ph1 = new phaser(); // Default mode is SIG_WAIT
    phaser ph2 = new phaser(); // Default mode is SIG_WAIT
    phaser ph3 = new phaser(); // Default mode is SIG_WAIT
    async phased(ph1<SIG>, ph2<WAIT>) { // i = 1
        doPhase1(1);
        next; // Signals ph1, and waits on ph2
        doPhase2(1);
    }
    async phased(ph2<SIG>, ph1<WAIT>, ph3<WAIT>) { // i = 2
        doPhase1(2);
        next; // Signals ph2, and waits on ph1 and ph3
        doPhase2(2);
    }
    async phased(ph3<SIG>, ph2<WAIT>) { // i = 3
        doPhase1(3);
        next; // Signals ph3, and waits on ph2
        doPhase2(3);
    }
}
```

Listing 3: Extension of example in Listing 1 with three phasers for $m = 3$
Whiteboard picture from lecture (Computation Graph for previous slide)
Left-Right Neighbor Synchronization Example for General $m$

```
finish {
    phaser ph = new phaser[m+2];
    forall(point [i]:[0:m+1]) ph[i]=new phaser(); // Default mode is SIG_WAIT
    for (point [i] : [1:m])
        async phased(ph[i]<SIG>, ph[i-1]<WAIT>, ph[i+1]<WAIT>) {
            doPhase1(i);
            next; // Signals ph[i], and waits on ph[i-1] and ph[i+1]
            doPhase2(i);
        }
}
```

Listing 4: Extension of example in Listing 1 with array of $m + 2$ phasers for general $m$