Chapter 5: Barriers to Performance

Elements of Parallel Computing

Eric Aubanel

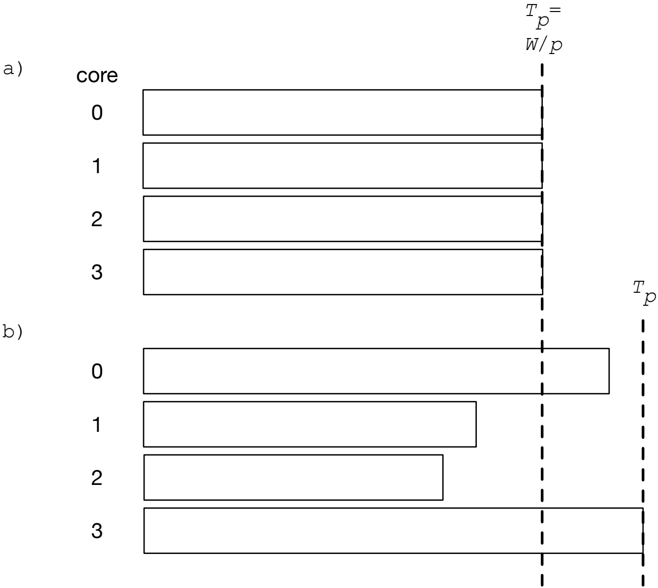
©2017 by Taylor & Francis Group, LLC.

Barriers to Performance

- Load Imbalance
- Communication Overhead
- False Sharing
- Locality: Hierarchical Algorithms
- Inherently Sequential Execution
- Memory
- Other Overhead

©2017 by Taylor & Francis Group, LLC.

Load Imbalance



©2017 by Taylor & Francis Group, LLC.

Load Balancing

- Static load balancing
  - round robin scheduling
- Dynamic load balancing
  - Master-worker
  - work-stealing (fork-join frameworks)

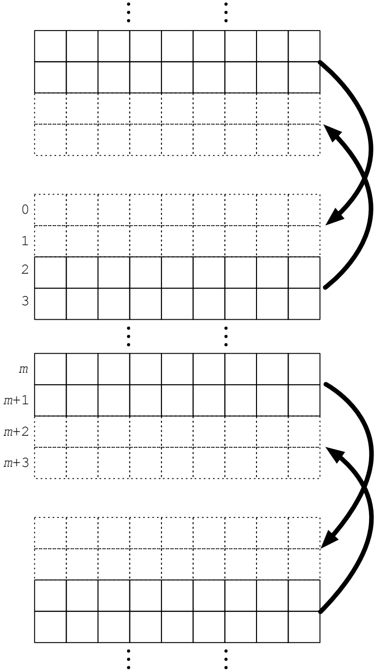
©2017 by Taylor & Francis Group, LLC.

Communication Overhead: Game of Life

Investigate 2 ways to reduce communication overhead:

1. tradeoff redudant communication with reduced communication
2. overlap communication and computation

©2017 by Taylor & Francis Group, LLC.



©2017 by Taylor & Francis Group, LLC.

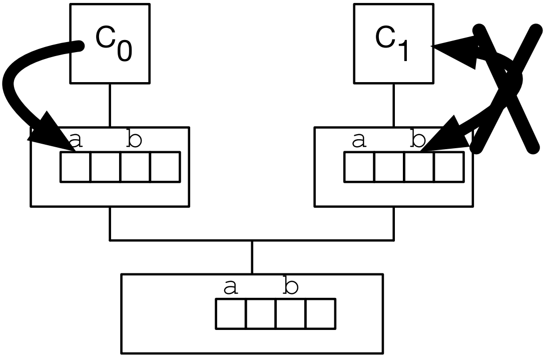
```
// each task has $(n/p+4) \times n$ arrays $grid$ and $newGrid$
$nbDown \leftarrow (id+1) \bmod p$, $nbUp \leftarrow (id-1+p) \bmod p$
$m \leftarrow n/p$ // Assume $n \bmod p = 0$
for $k \leftarrow 0$ to $N-1$ do
    offset $\leftarrow k \bmod 2$
    if offset $= 0$ then
        nonblocking send $grid[m..m+1, 0..n-1]$ to $nbDown$
        nonblocking send $grid[2..3, 0..n-1]$ to $nbUp$
        receive from $nbDown$ into $grid[m+2..m+3, 0..n-1]$
        receive from $nbUp$ into $grid[0..1, 0..n-1]$
    end
    foreach cell at coordinate $(i,j) \in (1+\text{offset}..m+2-\text{offset}, 0..n)$ do
        updateGridCell($grid$, $newGrid$, $i$, $j$)
    end
    swap references to $newGrid$ and $grid$
end
```

```
$nbDown \leftarrow (id + 1) \bmod p$, $nbUp \leftarrow (id - 1 + p) \bmod p$
$m \leftarrow n/p$ // Assume $n \bmod p = 0$
for a number of generations do
    nonblocking send $grid[m, 0..n - 1]$ to nbDown
    nonblocking send $grid[1, 0..n - 1]$ to nbUp

    foreach cell at coordinate $(i, j) \in (2..m - 1, 0..n)$ do
        updateGridCell(grid, newGrid, i, j)
    end
    // receive boundary values from neighbors
       into ghost elements
    receive from nbDown into $grid[m + 1, 0..n - 1]$
    receive from nbUp into $grid[0, 0..n - 1]$
    foreach cell at coordinate
    $(i, j) \in (1, 0..n) \wedge (m, 0..n)$ do
        updateGridCell(grid, newGrid, i, j)
    end
    swap references to newGrid and grid
end
```

©2017 by Taylor & Francis Group, LLC.

False Sharing



- False sharing of *a* and *b*, since intent isn't to share, but sharing is forced because both lie on a cache block.
- Concurrent access to *a* an *b* is not possbile.

©2017 by Taylor & Francis Group, LLC.

Locality: Hierarchical Algorithms

- Challenging to favour local data accesses in hierarchical memory (e.g. GPUs)
- Hierarchical algorithms can enhance locality
    - combine shared memory on node and message passing between nodes

©2017 by Taylor & Francis Group, LLC.

Inherently Sequential Execution

Amdahl's law:

$$\text{speedup} = \frac{t_1}{ft_1 + (1-f)t_1/p}$$

where f is fraction of sequential execution time t_1 for the operations that cannot be done in parallel

- Not significant problem for properly designed parallel programs
- Sequential portion often has lower complexity that parallel portion
- Weak scaling: time of sequential portion decreases as problem size increases

©2017 by Taylor & Francis Group, LLC.

Memory

- Protecting from data races takes overhead
  - use private data structures as much as possible
  - use lock-free algorithms
- Shared memory communication overhead
  - ensure locality

©2017 by Taylor & Francis Group, LLC.

Other Overhead

- Unavoidable
  - packing data before sending message
- Avoidable
  - needlessly allocating memory

©2017 by Taylor & Francis Group, LLC.