Computer Architecture: Parallel Task Assignment

Prof. Onur Mutlu
Carnegie Mellon University

Static versus Dynamic Scheduling

- Static: Done at compile time or parallel task creation time
  - Schedule does not change based on runtime information
- Dynamic: Done at run time (e.g., after tasks are created)
  - Schedule changes based on runtime information
- Example: Instruction scheduling
  - Why would you like to do dynamic scheduling?
  - What pieces of information are not available to the static scheduler?

Parallel Task Assignment: Tradeoffs

- Problem: N tasks, P processors, N>P. Do we assign tasks to processors statically (fixed) or dynamically (adaptive)?

- Static assignment

  + Simpler: No movement of tasks.

  - Inefficient: Underutilizes resources when load is not balanced

    When can load not be balanced?

- Dynamic assignment

  + Efficient: Better utilizes processors when load is not balanced

  - More complex: Need to move tasks to balance processor load

  - Higher overhead: Task movement takes time, can disrupt locality

Parallel Task Assignment: Example

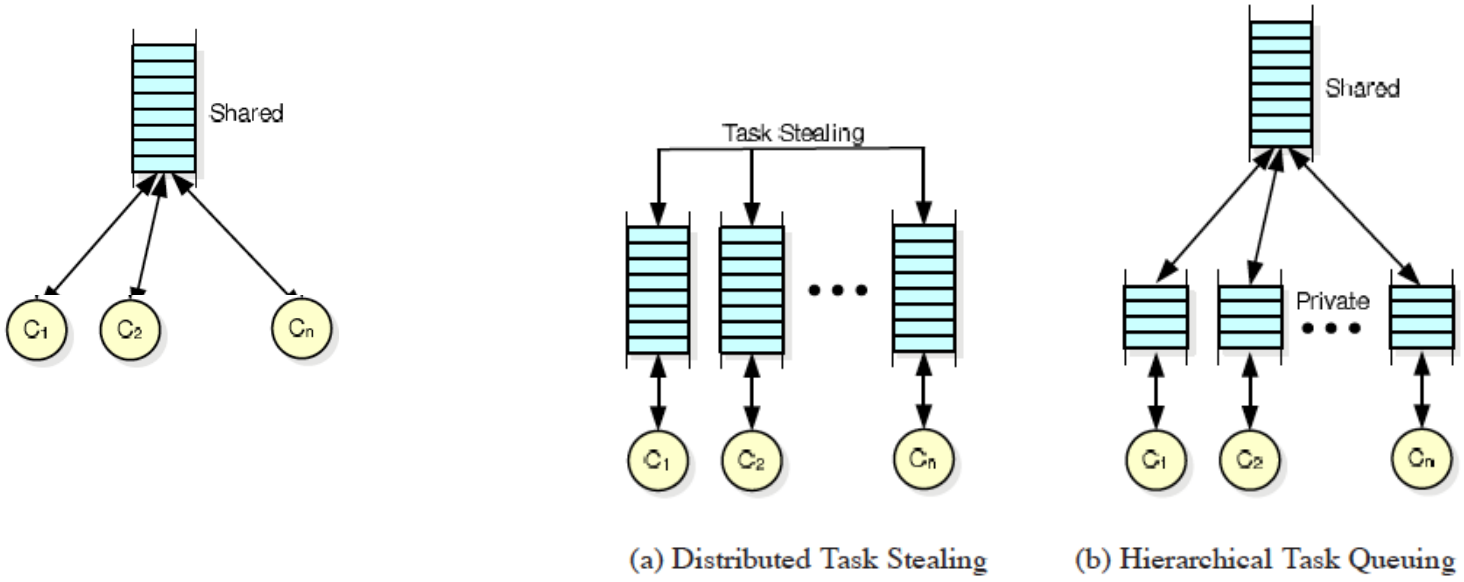
- Compute histogram of a large set of values
- Parallelization:
  - Divide the values across T tasks
  - Each task computes a local histogram for its value set
  - Local histograms merged with global histograms in the end

```
GetPageHistogram(Page *P)

    For each thread: {

        /* Parallel part of the function */
        UpdateLocalHistogram(Fraction of Page)

        /* Serial part of the function */
        Critical Section:
            Add local histogram to global histogram

        Barrier
    }

    Return global histogram
```

Parallel Task Assignment: Example (II)

- How to schedule tasks updating local histograms?
  - Static: Assign equal number of tasks to each processor
  - Dynamic: Assign tasks to a processor that is available
  - When does static work as well as dynamic?

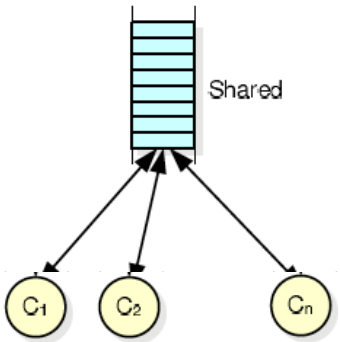
- Implementation of Dynamic Assignment with Task Queues

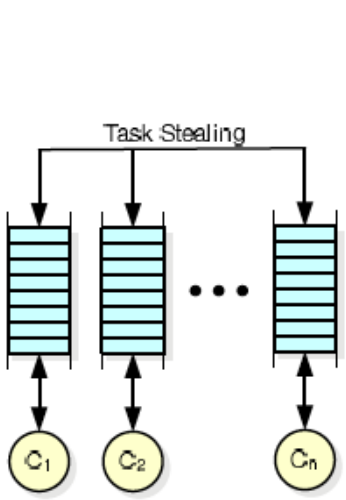


(a) Distributed Task Stealing (b) Hierarchical Task Queuing

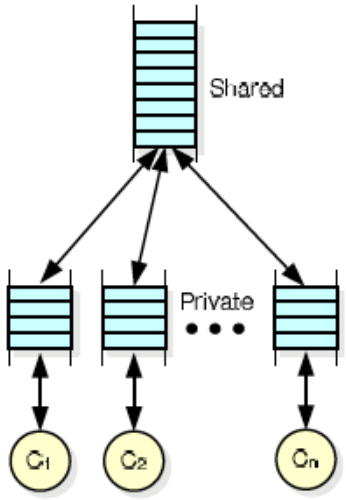
Software Task Queues

- What are the advantages and disadvantages of each?
  - Centralized
  - Distributed
  - Hierarchical





(a) Distributed Task Stealing



(b) Hierarchical Task Queuing

Task Stealing

- Idea: When a processor's task queue is empty it steals a task from another processor's task queue
  - Whom to steal from? (Randomized stealing works well)
  - How many tasks to steal?

+ Dynamic balancing of computation load

- Additional communication/synchronization overhead between processors
- Need to stop stealing if no tasks to steal

Parallel Task Assignment: Tradeoffs

- Who does the assignment? Hardware versus software?

- Software
  - + Better scope
  - - More time overhead
  - - Slow to adapt to dynamic events (e.g., a processor becoming idle)

- Hardware
  - + Low time overhead
  - + Can adjust to dynamic events faster
  - - Requires hardware changes (area and possibly energy overhead)

How Can the Hardware Help?

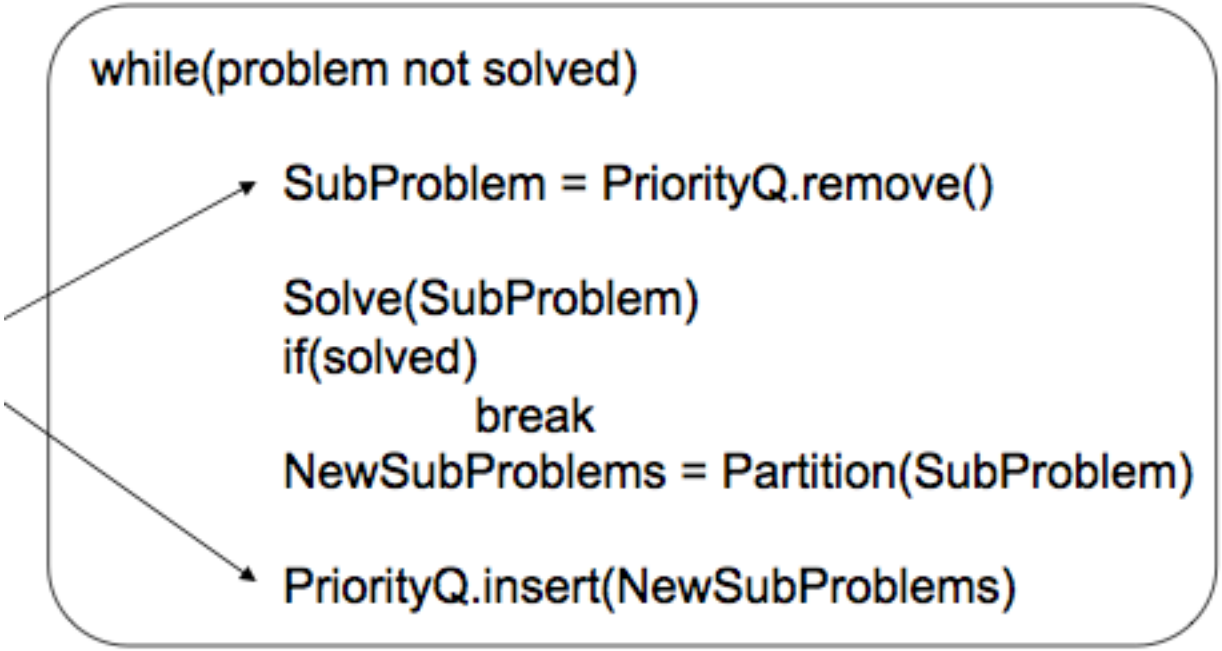
- Managing task queues in software has overhead
  - Especially high when task sizes are small

- An idea: Hardware Task Queues
  - Each processor has a dedicated task queue
  - Software fills the task queues (on demand)
  - Hardware manages movement of tasks from queue to queue
  - There can be a global task queue as well → hierarchical tasking in hardware

  - Kumar et al., "Carbon: Architectural Support for Fine-Grained Parallelism on Chip Multiprocessors," ISCA 2007.
    - Optional reading

Dynamic Task Generation

- Does static task assignment work in this case?
- Problem: Searching the exit of a maze



Computer Architecture: Parallel Task Assignment

Prof. Onur Mutlu
Carnegie Mellon University

Backup slides

Referenced Readings

- Kumar et al., "Carbon: Architectural Support for Fine-Grained Parallelism on Chip Multiprocessors," ISCA 2007.