Advanced OpenMP Features:

Tasking and Accelerator Support

Piotr Luszczek

OpenMP Tasking Overview

- OpenMP with version 4 started supporting tasking with data dependences
- Tasks were part of OpenMP since version 3
- Version 4 introduced data dependence clause
  - New clause “depend” allows to constraint execution of tasks based on how they pass data between each other
  - The data dependence also allows the tasks to synchronize selectively between each other based on the data they exchange
- OpenMP provides a scheduler that decides the cores to run the tasks
  - The scheduling decisions are made based on
    - Amount of work each core has
    - Availability of data in caches
      - It is beneficial to use the same core for tasks that share data that is already in cache

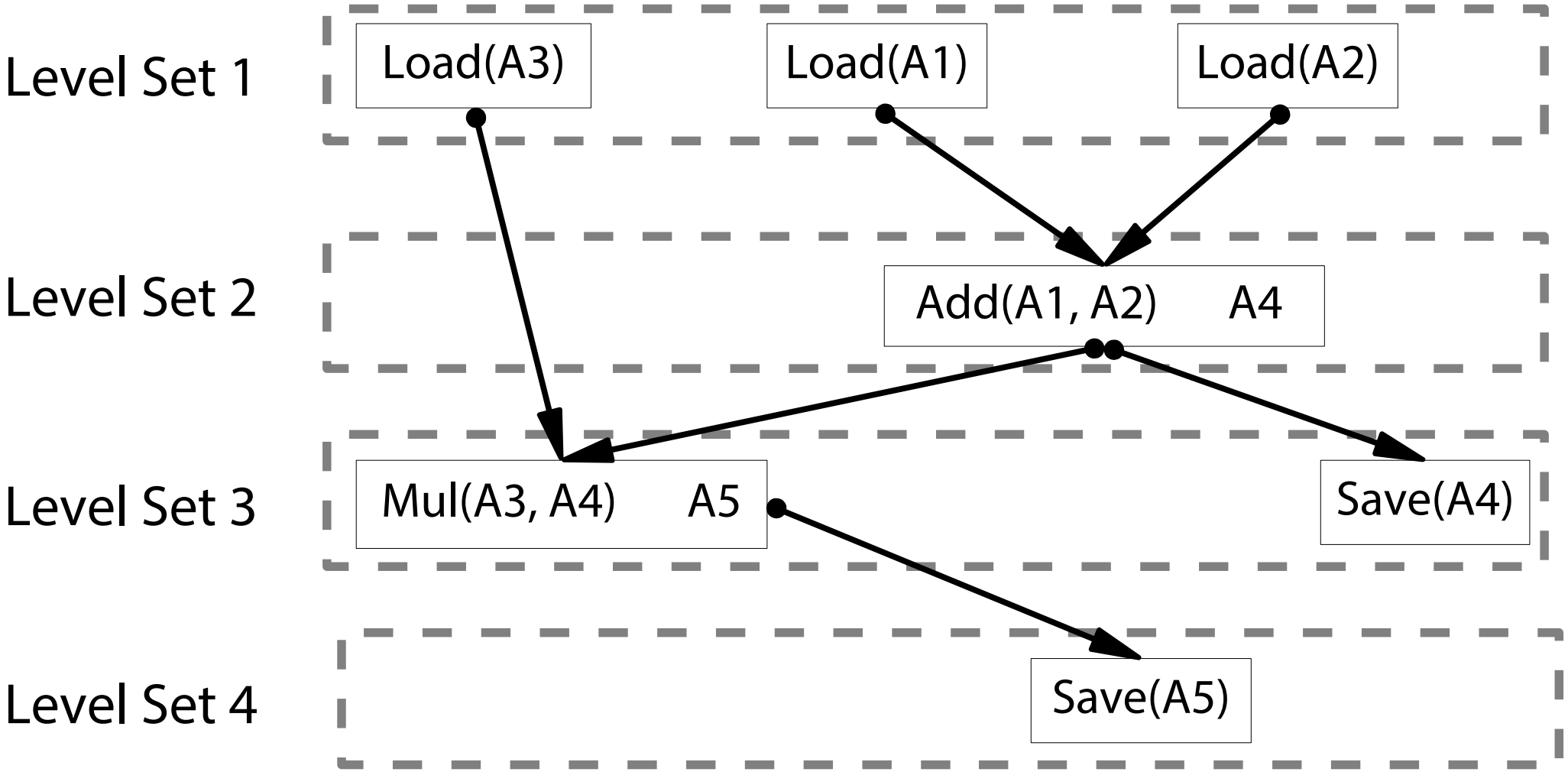
Task Directive Summary

- #pragma omp task
  - if (expression)
  - final
  - untied
  - default
  - mergeable
  - private (list)
  - firstprivate (list)
  - shared (list)
  - depend (dep. type : list)
  - priority (expression)
- By far, task might be the most complicated out of the commonly used ones
- The most important clauses will be discussed next

Depend Clause Summary

- #pragma omp task depend(*dependence type : list*)
- Dependence type can be either
  - in
    - Use it for data that is consumed by the tasks
  - out
    - Use it for data that is produced by the tasks
  - inout
    - Use it for data that is both consumed and produced by the tasks
  - Recall Fortran specification for subroutine function parameters
    - Since Fortran 90: in, out, inout, scratch
- List is a comma-separated enumeration of variables participating in data dependence graph
  - Arrays are specified with ranges:
    - For example: depend(in:A[0:N])

Recall DAG of Tasks and Its Level Sets



Implementation of the Sample Task DAG

```
#pragma omp parallel shared(a1, a2, a3, a4, n)
{
 #pragma omp master
 {
   #pragma omp task depend(out:a1[0:n])
   Load(n, a1);
   #pragma omp task depend(out:a2[0:n])
   Load(n, a2);
   #pragma omp task depend(out:a3[0:n])
   Load(n, a3);
   #pragma omp task depend(in:a1[0:n],a2[0:n]) depend(out:a4[0:n])
   Add(n, a1, a2, a4);
   #pragma omp task depend(in:a3[0:n],a4[0:n]) depend(out:a5[0:n])
   Mul(n, a3, a4, a5);
   #pragma omp task depend(in:a4[0:n])
   Save(n, a4);
   #pragma omp task depend(in:a5[0:n])
   Save(n, a5);
 }
}
```

- The order in which tasks are inserted is important
  - To be safe, exhaust the tasks from the first level set before inserting tasks from the second level

Advanced Use of Tasks

- It is possible to make complicated task graphs
  - `taskwait` directive creates a join point of descendent tasks of the current task
  - `taskgroup` allows creating of and waiting for descendant tasks
  - `taskyield` directive allows to relinquish CPU for other tasks
  - `taskloop` directive allows generation of tasks with loop constructs

OpenMP 4+ is not the Only One for Tasking

- Task-based data-parallel computing has become an important method for parallel computing
  - It has grown tremendously since the multicore era began
- There are many projects that use this paradigm
  - Shared memory: Hstreams, Open Community Runtime, ompSS, QUARK, Thread Building Blocks
  - Distributed memory: Legion, PaRSEC, RAJA, StarPU, Thor
  - There are more projects that provide this functionality

OpenMP and Accelerators

- It is possible to offload OpenMP tasks to accelerators
  - “target data” directive maps variables to a device
    - #pragma omp target data if(*scalar expression*) device(*integer expression*) map(*map type: list*) use_device_ptr(*list*)
  - “target enter data” performs the mapping and “target exit data” unmaps them
  - “target update” assures data consistence between host and device
  - “target” directive can be used to map data and execute code on device
- Support for OpenMP for accelerators is limited due to the market situation
  - GNU Compiler Collection include support for OpenMP offload but it usually is disabled and cannot be used at the same as OpenACC
  - Intel discontinued support for Xeon Phi accelerators beyond Knights Landing and Knights Hill hardware will not be released

OpenMP and OpenACC

- OpenMP started as CPU-only specification that now includes accelerators
  - Continues to have support from major CPU vendors
- OpenACC unified many offloading solutions for accelerators and now includes CPU as a target
  - It may be used with GNU and PGI compilers
  - US' DOE sponsors further development
    - This may introduce LLVM support