New release of OmpSs-2 programming model

The Programming Models group at BSC has published the second release (version 18.06) of the OmpSs-2 programming model.

OmpSs-2 extends the tasking model of OmpSs/OpenMP to support both task nesting and fine-grained dependencies across different nesting levels. This enables the effective parallelization of applications using a top-down methodology. In this release, the OmpSs-2 tasking model has been enhanced to better support hybrid (MPI+OmpSs-2) and heterogeneous (OmpSs-2+CUDA C kernels) programming. The following list gives an overview of most prominent features introduced in this release of OmpSs-2:

1. **Task-Aware MPI (TAMPI) library**

This MPI library developed in the context of the [INTERTWinE project](https://github.com/bsc-pm/tampi) augments the interoperability features of MPI to enhance hybrid programming with tasking models such as OmpSs-2. This MPI library has been extended with a new MPI threading level --called MPI_TASK_MULTIPLE-- that enables the safe use of synchronous and asynchronous MPI operations inside a task. This library relies on the Nanos6 pause/resume, external events and polling services APIs to provide this enhanced interoperability between MPI and OmpSs-2.

2. **CUDA Unified Memory**
The OmpSs-2 tasking model has been extended to support tasks that are written in CUDA. The CUDA kernels annotated with the OmpSs-2 `task` construct are invoked and scheduled like regular tasks, simplifying the development of heterogeneous applications. The current implementation relies on the Unified Memory feature provided by the latest NVIDIA cards to automatically move the required data from the host to the device and vice-versa.

3. Task priorities

The new scheduler implementation now supports the OpenMP priority clause to specify task priorities. By default, tasks have priority 0, but that can be changed though an integer value in the priority clause. Greater values indicate higher priority and smaller values indicate lower priority. Priorities can also be negative to indicate less priority than the default.

4. Array reductions (C & C++)

Reductions take advantage of working on multiple copies of a task data to enable parallelism where there wouldn’t be otherwise. While scalar reductions were already part of the model, this release includes the possibility to specify expressions with array type in the reduction clause. In addition, the new weakreduction clause can be used to specify the memory region where reductions are going to be defined, allowing nested reductions as well as memory allocation optimizations.

5. Nanos6 generic polling services

A new API allows to coordinate the execution of tasks with the execution of arbitrary functions. Its purpose is to improve the interoperability with software that requires polling-like behavior, in a way that minimizes the interference on resource usage.

6. External events API

This new API can be used to defer the release of dependencies of a task until the task has been executed and a set of external events are fulfilled (e.g. on completion of an asynchronous MPI operation). This API is used to implement the support of asynchronous MPI operations on the TAMPI library.

Download from Github

Barcelona Supercomputing Center - Centro Nacional de Supercomputación