Programming Model
INTERoperability ToWards Exascale

FETHPC 1-2014 – HPC Core Technologies, Programming Environments and Algorithms for Extreme Parallelism and Extreme Data Applications

Nick Brown, EPCC
• Exascale architectures will rely on extreme-scale parallelism exposed in multiple layers of hardware
  - nodes, cores, hardware threads, SIMD lanes, GPUs

• No sign of “silver bullet” programming model (API) to address all these layers within timeframe of first Exascale systems

• Thus applications requiring Exascale performance must use multiple APIs in single program
  - Significant expansions of current hybrid programming (e.g. MPI and OpenMP)
• Problems using different APIs in same ‘layer’
  • E.g. program using OpenMP threads not easily able to exploit library w/ POSIX threads
  • Hybrid codes typically scale modestly due to simplified interaction necessitated by approach to interoperability

=  

• Interoperability between APIs is very important
  • and already becoming a bottleneck

• To advance necessitates changes/ improvements to both
  • API specification
  • Implementation (especially runtimes)
**INTERTWinE: The programming technologies**

- **OmpSs**: Runs sequential applications in data-flow execution model on distributed systems.
- **PaRSEC**: Scheduler for execution of micro-tasks on distributed memory systems.
- **OpenMP**: Widely used API for shared memory parallelism.
- **StarPU**: Task scheduler for distributed, heterogeneous (accelerator) systems.
- **GASPI**: PGAS model focused on asynchronous, one-sided communications.
- **MPI**: Distributed memory model, dominant and ubiquitous.

[http://www.intertwine-project.eu](http://www.intertwine-project.eu)
INTERTWinE Resource manager

Coordinates access to CPU resources between different runtime systems and APIs to avoid both oversubscription and undersubscription situations.

- An offloading API to invoke parallel kernels on a specific set of CPUs from one runtime system to another one.
- A dynamic resource sharing API to transparently lend and borrow resources between parallel runtimes to avoid under-utilization scenarios.
Task-aware communication libraries

• To obtain maximum effectiveness from asynchronous tasks with dependencies, we need to issue communications in tasks
  • Blocking communication calls inside a task wastes resources
  • Out-of-order execution of communications can cause deadlock

• To solve these problems INTERTWinE has created task-aware versions of MPI and GASPI libraries – TAMPI ands TAGASPI
  • Tasks blocked on communications are paused
  • The underlying thread is free to execute other tasks
  • When the data is ready, the communication task is resumed
Distributed tasks

- **Directory/cache**
  - Task-based programming models can work with an abstract view of the distributed memory as a single shared address space
  - Complete independence of runtime from the physical representation of data and from the type of storage
  - Access through the same interface to an extendable list of transport technologies

- **EDAT – Event driven asynchronous tasks**
  - Implementing task models transparently on distributed systems is hard!
  - Instead, keep the distribution explicit and the programmer submits tasks and drives interaction via events. These are fired between processes and contribute to the execution of tasks
GASPI and shared segments

• GASPI is designed to run with one process per node, and MPI is designed to run with one process per core
  • Makes interoperability a bit awkward without using hybrid MPI + OpenMP as well

• INTERTWinE has made enhancements to GASPI to support a one process per core model through shared memory segments
  • Easier migration path from MPI to GASPI
  • Zero-copy data exchanges inside a node
  • Ultra fast implementations of collective and halo-exchange communication patterns
API enhancements/recommendations

**MPI Forum**
- Endpoints / Finepoints – threads as sources and destinations
- Persistent Collectives
- Sessions – context isolation
- MPI_TASK_MULTIPLE construct

**GASPI Forum**
- GASPI Shared Windows
- Fine-grain local completion

**OpenMP ARB:**
- Pause/ Resume API
- Task dependencies on external events
Developer resources

- Developer hub on INTERTWinE web site
  - https://www.intertwine-project.eu/
- Best practice guides
  - MPI + OpenMP
  - MPI + tasks
  - MPI + GASPI
- Using the Resource Manager
- Using GASPI shared notifications
- Example codes to download and run
- Information on the project’s work with standards bodies
- Application workshop white papers
Contact and further information

http://www.intertwine-project.eu/
info@intertwine-project.eu