Proposal for a next generation task scheduler in Gaudi

Illya Shapoval
Lawrence Berkeley National Laboratory

Intro: status quo

- The current solution is the Avalanche Scheduler and associated services
  - Relies on Intel TBB runtime system for mapping tasks to kernel threads
  - Other TBB features (full-blown load-balancing, flowgraph, priorities) are not used, for reasons
  - All high-level decision making is made by the Avalanche Scheduler
- Basic and advanced functionality is in place for both offline and online
  - Though my personal wish list for advanced features is somehow never ending
- Advanced functionality for throughput maximization is awaiting real workflows:
  - Topology-aware avalanche task emission modes;
  - Computations offloading mode;
- Optimization is needed in some corners of the scheduling phase space
  - decision making is too aggressive while waiting for resource contention resolution

That said, scalability of this solution over exascale and heterogeneous computer architectures is limited, by design.

Ste||ar HPX

Backup

(won’t teach HPX, will show instead what it may bring)

HPX - High Performance ParalleX

- HPX is a general purpose C++ Standard library for parallelism and concurrency
- Implements the ParalleX execution model (slide 9)
- Portability:
  - Platforms: x86/64, ARM 32/64, Power, Xeon/Phi (who cares?), BlueGene/Q
  - OS: Linux, OS X/macOS, Windows, Android
- License: BSL-1.0
- Developed by the Ste||ar Group:
  - Louisiana State University
  - Friedrich-Alexander-Universität
  - Sandia National Lab
  - New Mexico State University
  - Lawrence Berkeley National Lab
- Funding:
  - National Science Foundation awards (3)
  - Department of Energy awards (2)
  - Defense Technical Information Center contract
  - Bavarian Research Foundation grant
  - EC's Horizon 2020 programme grant

Employ an alternative execution model.
Maximize throughput.

HPX principles (excerpt)

To overcome the **SLOW** barriers on the way to scalability:

- **S**tarvation for available concurrent work
- **L**atencies in accessing local and remote resources
- **O**verhead of concurrency management
- **W**aiting for contention resolution on oversubscribed shared resources

TBB principles (compare)

To overcome the **SLOW** barriers on the way to scalability:

- **S**tarvation for available concurrent work
- ~~**L**atencies in accessing local and remote resources~~
- **O**verhead of concurrency management (partial mitigation)
- ~~**W**aiting for contention resolution on oversubscribed shared resources~~

HPX principles (excerpt)

To overcome the **SLOW** barriers on the way to scalability:

- **S**tarvation for available concurrent work
- **L**atencies in accessing local and remote resources
- **O**verhead of concurrency management
- **W**aiting for contention resolution on oversubscribed shared resources

HPX implements the ParalleX execution model based on the following principles:

- Latency hiding (vs latency avoidance)
- Fine-grained parallelism (vs heavyweight threads)
- Constraint based synchronization (vs global barriers)
- Adaptive locality control (vs static data distribution)
- Work following data (vs data following work)
- Message driven (vs message passing)

HPX principles (excerpt)

To overcome the **SLOW** barriers on the way to scalability:

- **S**tarvation for available concurrent work
- **L**atencies in accessing local and remote resources
- **O**verhead of concurrency management
- **W**aiting for contention resolution on oversubscribed shared resources

HPX implements the ParalleX execution model based on the following principles:

- Latency hiding (vs latency avoidance)
- Fine-grained parallelism (vs heavyweight threads)
- Constraint based synchronization (vs global barriers)
- Adaptive locality control (vs static data distribution)
- Work following data (vs data following work)
- Message driven (vs message passing)

See more details!

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy
- Inherent support for offloading computations

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy
- Inherent support for offloading computations
- Distributed memory computing
  - Based on Active Global Address Space

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy
- Inherent support for offloading computations
- Distributed memory computing
  - Based on Active Global Address Space
- Load-balancing (intra-node and inter-node)

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy
- Inherent support for offloading computations
- Distributed memory computing
  - Based on Active Global Address Space
- Load-balancing (intra-node and inter-node)
- Performance monitoring subsystem

Intel TBB

Intra-node scheduling:

- 1:1 threading model
- Task-based programming model
  - non-blocking tasks only
  - unfair scheduling policy
- Load-balancing

Facilities:

- Generic parallel algorithms
- Synchronization primitives
- Concurrent containers

Ste||ar HPX

Intra-node and inter-node scheduling:

- M:N hybrid threading model
  - lightweight user-level threads (fast context switches)
- Task-based asynchronous programming model
  - blocking and non-blocking tasks
  - cooperative scheduling policy
- Inherent support for offloading computations
- Distributed memory computing
  - Based on Active Global Address Space
- Load-balancing (intra-node and inter-node)
- Performance monitoring subsystem
- Conformance to C++11,14,17 and the ongoing 20.

Facilities:

- Generic parallel algorithms
- Lightweight control objects for fast synchronization

Expected deliverables in Gaudi (and upstream)

- Better abstraction of a scheduling runtime system from the rest of the framework
- New HPX-based scheduler (and other associated services) in Gaudi:
  - Conforming C++11/14/17 and aligned to C++20 (Parallelism TS v2, Concurrency TS)
  - Exascalable:
    - Best in class concurrency management overhead
    - Inherent support for heterogeneous computing (∃ HPX/CUDA and HPX/SYCL extensions)
    - Capacity for distributed memory computing
  - Capable of proactive runtime throughput maximization
    - Based on HPX performance self-monitoring subsystem
- Framework-wide adoption of Asynchronous C++ Standard programming model
  - Resource contention agnostic
  - Disk and network I/O latency agnostic
  - Inherent support for heterogeneous computing model

Outro

Asynchrony. Futurization.
Latency-agnostic ops:
  Heterogeneous computing.
  Distributed memory computing.
Runtime adaptability.
Conformance to C++ Standards.