目前的基于任务的编程模型可分为3类：

(1) library solution, e.g. Intel TBB, Argobots, QThreads, Kokkos, HPX;

(2)语言拓展：Intel Cilk Plus, OpenMP

(3)编程语言：Chapel, Intel ISPC, X10

HPX严格符合C++11和C++17的标准API定义。例如，可使用std::future替换futurization概念的hpx::future，而不会破坏API。

HPX is C++ language extension and runtime system, Charm++ is a stand-alone library.

# N-Body

HPX库用来实施基于AMT的N-Body模拟，并与non-AMT做了对比。

Heller T, Kaiser H, Scher A, Fey D (2013) Using HPX and LibGeoDecomp for scaling HPC applications on heterogeneous supercomputers. In: Proceedings of the workshop on latest advances in scalable algorithms for large-scale systems. ACM, p 1

Khatami Z, Kaiser H, Grubel P, Serio A, Ramanujam J (2016) A massively parallel distributed n-body application implemented with hpx. In: Proceedings of the 2016 7th workshop on latest advances in scalable algorithms for large-scale systems (ScalA). IEEE, pp 57-64.

# Octo-tiger

HPX库在天文模拟的应用，在648,280 Intel Knight's landing cores上并行效率达96.8% (Heller T, 2019)，在2048核的PizDaint超算上并行效率达68.1% (Dai G, 2019)。另外，用libfbric代替MPI通信，可实现2.8x的加速比 (Grun P, et al. 2015)。

Heller T, Lelbach BA, Huck KA, Biddiscombe J, Grubel P, Koniges AE, Kretz M, Marcello D, Pfander D, Serio A, Frank J, Clayton GC, Pflger D, Eder D, Kaiser H. Harnessing billions of tasks for a scalable portable hydrodynamic simulation of the merger of two stars. 2019. Int J High Perform Comput Appl

Dai G, Amini P, Biddiscombe J, Diehl P, Frank J, Huck K, Kaiser H, Marcello D, Pfander D, Pfüger D (2019) From Piz Daint to the stars: simulation of stellar mergers using high-level abstractions. In: Proceedings of the international conference for high performance computing, networking, storage and analysis, pp. 1-37

Grun P, Hefty S, Sur S, Goodell D, Russell RD, Pritchard H, Squyres JM (2015) A brief introduction to the openfabrics interfaces-a new network api for maximizing high performance application efficiency. In: Proceedings of the 2015 IEEE 23rd annual symposium on high-performance interconnects. IEEE, pp 34-39

# NLmech

Patrick Diehl, Prashant K. Jha, Hartmut Kaiser, Robert Lipton, Martin Lévesque. An asynchronous and task‑based implementation of peridynamics utilizing HPX -- the C++ standard library for parallelism and concurrency. SN Applied Sciences (2020) 2: 2144 | https://doi.org/10.1007/s42452-020-03784-x

Patrick Diehl et al., 2020将HPX用于近场动力学(Peridynamics)的断裂力学的应用开发，基于NLmech实现HPX并行化。

https ://github.com/nonlocalmodels/NLMech

隐式和显式离散

加速比和并行效率如下图：



# DGSWEM-HPX

Maximilian Bremer, Kazbek Kazhyken, Hartmut Kaiser, Craig Michoski, Clint Dawson. 2019. Performance Comparison of HPX Versus Traditional Parallelization Strategies for the Discontinuous Galerkin Method. Journal of Scientific Computing, 80: 878-902.

应用HPX并行化DG法的浅水方程求解器DGSWEM模型。HPX的效果超过MPI并行。

Wei Weile, et al. Performance analysis of a Quantum Monte Carlo application on multiple hardware architectures using the hpx runtime. arXiv:2010.07098v3 [cs.DC] 19 Oct 2020

将一个量子Mont Calo应用程序(Dynamical Cluster Approximation (DCA++))迁移到HPX上，并使用HPX-APEX提高抽象层来理解性能问题，识别代码中的任务优化机会，如何将这些与CPU/GPU利用计数器联系起来，设备内存分配和CPU核函数层上下文切换。

# HPXCL

Patrick Diehl的学位论文：Modeling and Simulation of cracks and fractures with peridynamics in brittle materials

主要介绍了HPXCL的开发

HPX compute language (HPXCL)是解决GPU核函数执行和数据拷贝延迟问题的编程环境，异步传输数据和异步执行CUDA核函数，而在主核心上同步执行。

所有的GPU操作都表示为异步任务，就像CPU上的并行任务。因此，这些任务整合到HPX的并行执行图中。

## 实施过程

futurization

该API提供异步返回类型hpx::future<T>

future类型是基于模板的，调用函数的具体返回值隐藏在future中。因此，被调用函数立即返回，甚至返回值尚未计算完成。调用函数的具体返回值可使用future的.get()算子获取。为同步立即返回的futures，使用hpx::future<T>::then与 hpx::future<T>::wait\_all来完成合成，使用hpx::dataflow隐式构建并行执行图(DAG: 有向不循环图)。

图2显示了使用futurization解决简单依赖图（图1）的方法。

 

图1 依赖图示例 图2 使用HPX的合成API模拟依赖图

为整合CUDA函数，诸如数据转移或启用核函数，返回一个hpx::future。因此，核函数的数据依赖与上述方法相同，核函数启动的结果数据可以作为主机上的任务的依赖。注意：所有CUDA函数需返回相同类型的future，可能将他们与在主机上的其他future联合。

CUDA API设计



图4.6 HPXCL-CUDA组件的类，通过主机上的AGAS可以对所有localities做全局寻址

各设备的物理内存组织为类Buffer。设备内存不能使用AGAS全局寻址。因此，实施某种封装(wrapping)。使用该类生成的一个实例，为设备分配需要的空间。注意：

第2个从物理设备衍生的对象是类Program。此处，与CUDA代码的编译过程不同。编写纯CUDA代码意味着核函数和设备函数都使用nvcc编译。HPXCL-CUDA使用nvrtc运行时编译器，在运行时动态编译核函数。



图4.7 核函数执行与使用HPX的composition的同步的顺序示意图

效率评测