格点 QCD 的计算方法与软件研发

宫明

中国科学院高能物理研究所
CLQCD 合作组

中国科学技术大学

2024.07.16

格点量子色动力学 (Lattice QCD)

格点 QCD 是从第一性原理出发非微扰地系统研究强子和强相互作用的高能物理前沿领域

格点量子色动力学 (Lattice QCD)

格点 QCD 是从第一性原理出发非微扰地系统研究强子和强相互作用的高能物理前沿领域

格点量子色动力学 (Lattice QCD)

- **第一性原理**
- **非微扰**
- **系统研究**
- **强相互作用**
- **高能物理前沿**
- **领域**

格点量子色动力学 (Lattice QCD)

第一性原理

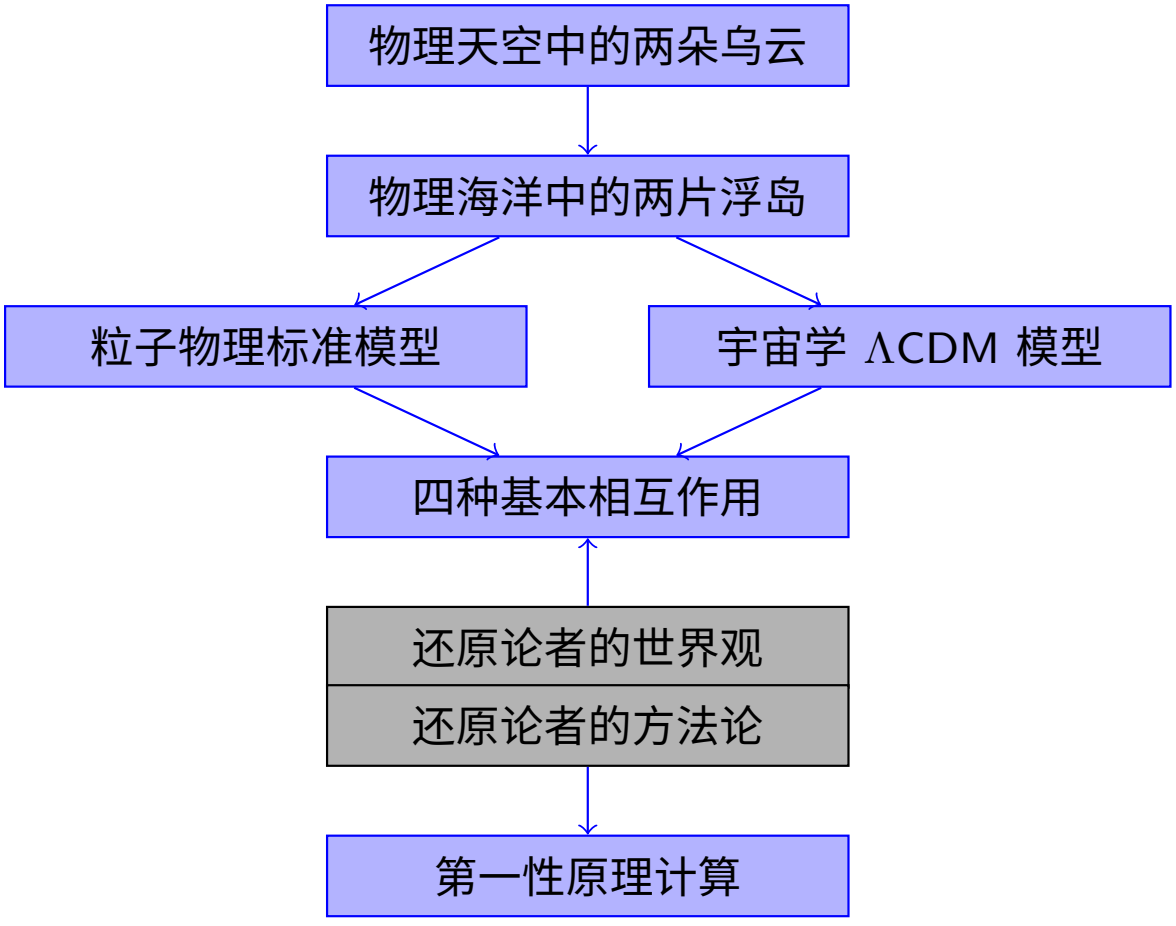
非微扰

系统研究

强相互作用

高能物理前沿

领域



格点量子色动力学 (Lattice QCD)

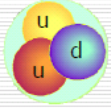
- **第一性原理**
- **非微扰**
- **系统研究**
- **强相互作用**
- **高能物理前沿**
- **领域**

粒子物理标准模型

The Standard Model

- Matter particles
  - 6 quarks $\begin{pmatrix} u \\ d \end{pmatrix}$ $\begin{pmatrix} s \\ c \end{pmatrix}$ $\begin{pmatrix} t \\ b \end{pmatrix}$
  - 6 leptons $\begin{pmatrix} e \\ \nu_e \end{pmatrix}$ $\begin{pmatrix} \mu \\ \nu_\mu \end{pmatrix}$ $\begin{pmatrix} \tau \\ \nu_\tau \end{pmatrix}$

Hadons(proton, neutron,pion etc) are cpmposites of 3 or 2 quarks



proton=uud

- Particles mediating interactions
  - photon γ EM
  - Weak bosons $W,\ Z$ Weak interactions

Weinberg-Salam theory

  - gluons g Strong interaction

Quantum Chromodynamics (QCD)

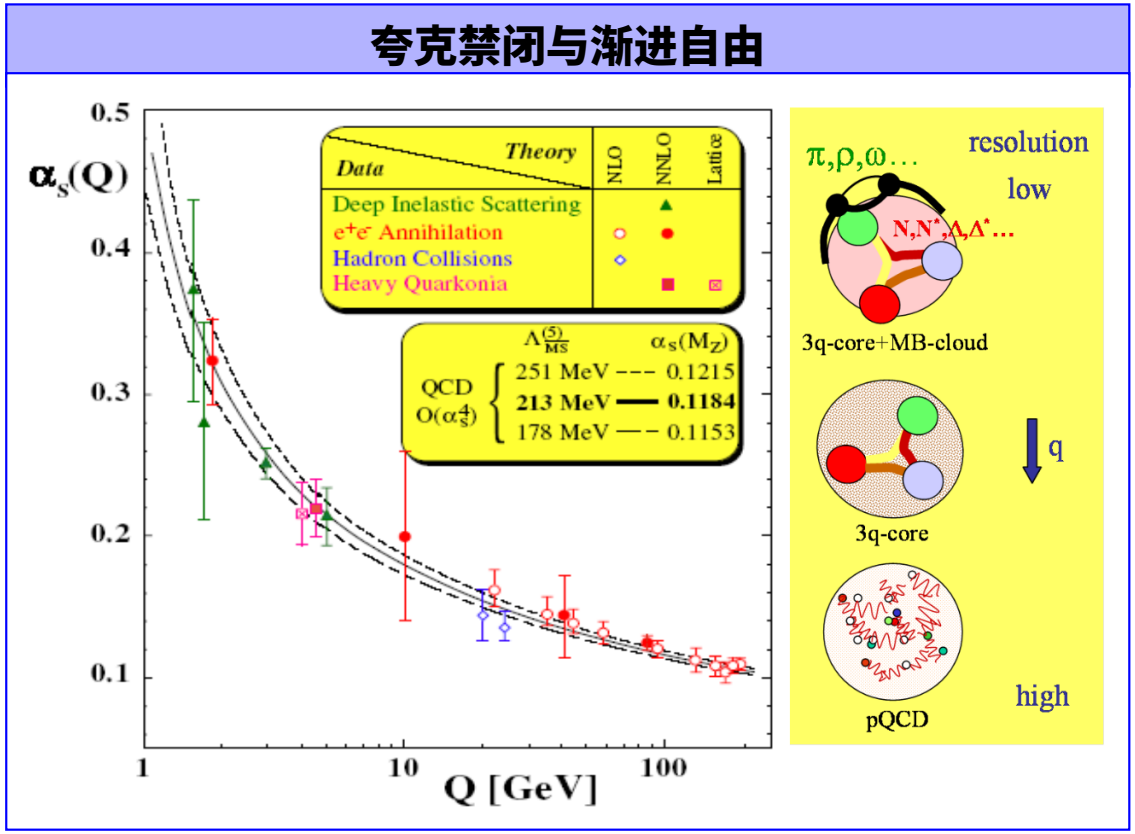
- Gauge field theory based on

$$\underbrace{SU(3)}_{QCD} \otimes \underbrace{SU(2) \otimes U(1)}_{EM+Weak}$$

$$L_{QCD} = -\frac{1}{4} Tr(F^{\mu\nu} F_{\mu\nu}) + \sum_f \bar{q}_f \left[i\gamma^\mu . \left(\partial_\mu - igA_\mu \right) - m_f \right] q_f$$

格点量子色动力学 (Lattice QCD)

- 第一性原理
- 非微扰
- 系统研究
- 强相互作用
- 高能物理前沿
- 领域



格点量子色动力学 (Lattice QCD)

- 第一性原理
- 非微扰
- 系统研究
- 强相互作用
- 高能物理前沿
- 领域

Wick 转动

$$x_0 \equiv t \rightarrow -ix_4 \equiv -i\tau$$

$$p_0 \equiv E \rightarrow ip_4$$

$$e^{iS_M} \equiv e^{i\int d^4 x_M \, L(x_M)} = e^{\int d^4 x_E \, L(x_E)} \equiv e^{-S_E}$$

欧氏时空的场论语言和统计语言

The equivalences between a Euclidean field theory and Classical Statistical Mechanics.

Euclidean Field Theory	Classical Statistical Mechanics
Action	Hamiltonian
unit of action h	units of energy $\beta = 1/kT$
Feynman weight for amplitudes $e^{-S/h} = e^{-\int \mathcal{L}dt/h}$	Boltzmann factor $e^{-\beta H}$
Vacuum to vacuum amplitude $\int \mathcal{D}\phi e^{-S/h}$	Partition function $\sum_{conf.} e^{-\beta H}$
Vacuum energy	Free Energy
Vacuum expectation value $\langle 0 \vert \mathcal{O} \vert 0 \rangle$	Canonical ensemble average $\langle \mathcal{O} \rangle$
Time ordered products	Ordinary products
Green's functions $\langle 0 \vert T[\mathcal{O}_1 \ldots \mathcal{O}_n] \vert 0 \rangle$	Correlation functions $\langle \mathcal{O}_1 \ldots \mathcal{O}_n \rangle$
Mass M	correlation length $\xi = 1/M$
Mass-gap	exponential decrease of correlation functions
Mass-less excitations	spin waves
Regularization: cutoff Λ	lattice spacing a
Renormalization: $\Lambda \rightarrow \infty$	continuum limit $a \rightarrow 0$
Changes in the vacuum	phase transitions

格点量子色动力学 (Lattice QCD)

- 第一性原理
- 非微扰
- 系统研究
- 强相互作用
- 高能物理前沿
- 领域

配分函数与作用量

$$Z = \int DA_\mu \; D\psi \; D\bar{\psi} \; e^{-S}$$

$$S = \int d^4x \left(\tfrac{1}{4} F_{\mu\nu} F^{\mu\nu} - \bar{\psi} M \psi \right)$$

把费米场积分掉

$$Z = \int DA_\mu \, \det M \, e^{-\frac{1}{4} F_{\mu\nu} F^{\mu\nu}}$$

物理观测量

$$\langle O \rangle = \tfrac{1}{Z} \int DA_\mu \, O \, e^{-S}$$

问题的转化

在以规范场为自由度的统计系统里，求解观测量的系综平均值

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

科学的研究范式

1、实验观测

2、理论推演

3、数值模拟：格点 QCD 面向理论，但“以上帝的视角做实验”

4、数据推动?

格点量子色动力学 (Lattice QCD)

- 第一性原理
- 非微扰
- 系统研究
- 强相互作用
- 高能物理前沿
- 领域

科学的研究范式

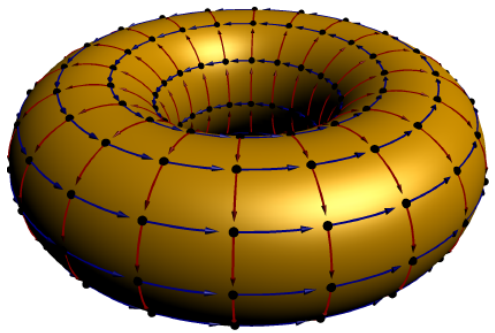
1、实验观测

2、理论推演

3、数值模拟：格点 QCD 面向理论，但“以上帝的视角做实验”

4、数据推动?

离散化：四维环面



在计算机上的表示

- 规范场用群表示：$U_\mu(x) = e^{iagA_\mu(x)}$，是 3×3 的复矩阵
- 费米场只需夸克传播子：$G_q(U) = M^{-1}(U)$，由规范场决定

格点量子色动力学 (Lattice QCD)

- 第一性原理
- 非微扰
- 系统研究
- 强相互作用
- 高能物理前沿
- 领域

作用量的设计

- 规范场能构成的规范不变的量，只有圈，最简单的示例：

$$S_g = \frac{6}{g^2} \sum_x \sum_{\mu<\nu} \mathrm{ReTr}\, \frac{1}{3} \left(1 - W_{\mu\nu}^{1\times 1}\right)$$

- 为了减小离散误差，其他定义按特定比例加入了较大的圈
- 费米场作用量示例：Wilson 费米子矩阵：

$$M^W_{x,y}[U]a = \delta_{x,y} - \kappa \sum_\mu \left[\left(r - \gamma_\mu\right) U_{x,\mu}\delta_{x,y-\mu} + \left(r + \gamma_\mu\right) U^\dagger_{x-\mu,\mu}\delta_{x,y+\mu}\right]$$

- 其他费米子作用量定义有：Staggered 费米子、Domain-wall 费米子、Overlap 费米子等

格点 QCD 计算的基本框架：蒙特卡洛法

1、随机生成大量规范场组态 $\{U_\mu(x)\}$，使得出现概率：

$$P(U) \propto e^{-S_g(U)} \det M(U)$$

2、计算可观测量：

$$\langle O\rangle = \frac{1}{N} \sum_{\{U\}} O(U, G_q(U))$$

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

误差预算

- 统计误差：随组态个数的平方根压低
  - 可通过增加统计量而稳步降低
- 系统误差：离散误差、有限体积效应
  - 可对格距和体积进行外推，得到连续的无限体积时空的结果
- 系统误差：参数偏差
  - 示例：夸克质量参数设置不精确，或故意设置较重的质量
  - 可通过多参数内插或外推消除偏差
- 系统误差：一些计算细节导致的其他误差或偏差
  - 示例：对关联函数进行拟合时选取拟合范围的模糊因素
  - 可通过改进算法消除或压低

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

误差预算

- 统计误差：随组态个数的平方根压低
  - 可通过增加统计量而稳步降低
- 系统误差：离散误差、有限体积效应
  - 可对格距和体积进行外推，得到连续的无限体积时空的结果
- 系统误差：参数偏差
  - 示例：夸克质量参数设置不精确，或故意设置较重的质量
  - 可通过多参数内插或外推消除偏差
- 系统误差：一些计算细节导致的其他误差或偏差
  - 示例：对关联函数进行拟合时选取拟合范围的模糊因素
  - 可通过改进算法消除或压低

所有误差，归根结底，都交由摩尔定律解决！

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

格点 QCD 的研究方向

- 强子谱学
  - 传统强子的高精度计算、奇特强子态、用散射方法研究共振态、……
- 强子结构
  - 各种形状因子、部分子分布函数、……
- 极端条件下的 QCD
  - 高温/高密/强磁场、QCD 临界点、手征相变、强磁场下的相结构、……
- 味物理和超出标准模型的新物理
  - CKM 矩阵元、中微子相关的强子矩阵元、……
- ……

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

高能物理的三个传统前沿

- 高能量前沿
  - LHC(ATLAS、CMS、LHCb)、CEPC 等
- 高亮度前沿
  - BEPCII(BESIII)、Belle II 等
  - 中微子实验 (大亚湾、江门、TUNE 等)
- 宇宙学前沿
  - 宇宙线观测 (LHAASO 等)

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

高能物理的三个传统前沿

- 高能量前沿
  - LHC(ATLAS、CMS、LHCb)、CEPC 等
- 高亮度前沿
  - BEPCII(BESIII)、Belle II 等
  - 中微子实验 (大亚湾、江门、TUNE 等)
- 宇宙学前沿
  - 宇宙线观测 (LHAASO 等)

第四个前沿

- 高精度前沿
  - 经过四十年的发展，从解释现象到预言现象
  - 很多探索新物理的瓶颈在强相互作用的高精度计算上

格点量子色动力学 (Lattice QCD)

第一性原理

非微扰

系统研究

强相互作用

高能物理前沿

领域

世界观

对量子场论的非微扰定义

方法论

可操作的计算模型

领域交叉

- 不同的研究方法：
  - 有效场论、粒子物理唯象学、少体核物理、……
- 拓展研究方法：
  - 量子计算、……
- 研究方法本身：计算机科学
  - 高性能计算、数值计算方法
  - 软件设计、计算机语言、计算机体系结构

国内外格点 QCD 计算软件现状

USQCD SciDAC 软件集（Chroma、QUDA、QDP++、……）

+ 国际普遍使用
+ 模块化，支持 intel、nvidia 等架构

- 原版不支持国产超算
- 单任务，中小规模运行

Bridge++、Grid、tmLQCD、openQCD、……

+ 代码量较小，易于扩展和修改
+ 适应各自特定需求

- 功能单薄
- 跨平台性较差
- 不支持国产超算

各研究组的其他软件等

+ 实现通用软件不支持的功能

- 规模小、功能单一、大多不跨平台
- 大多不开源

国产新一代软硬件环境的挑战

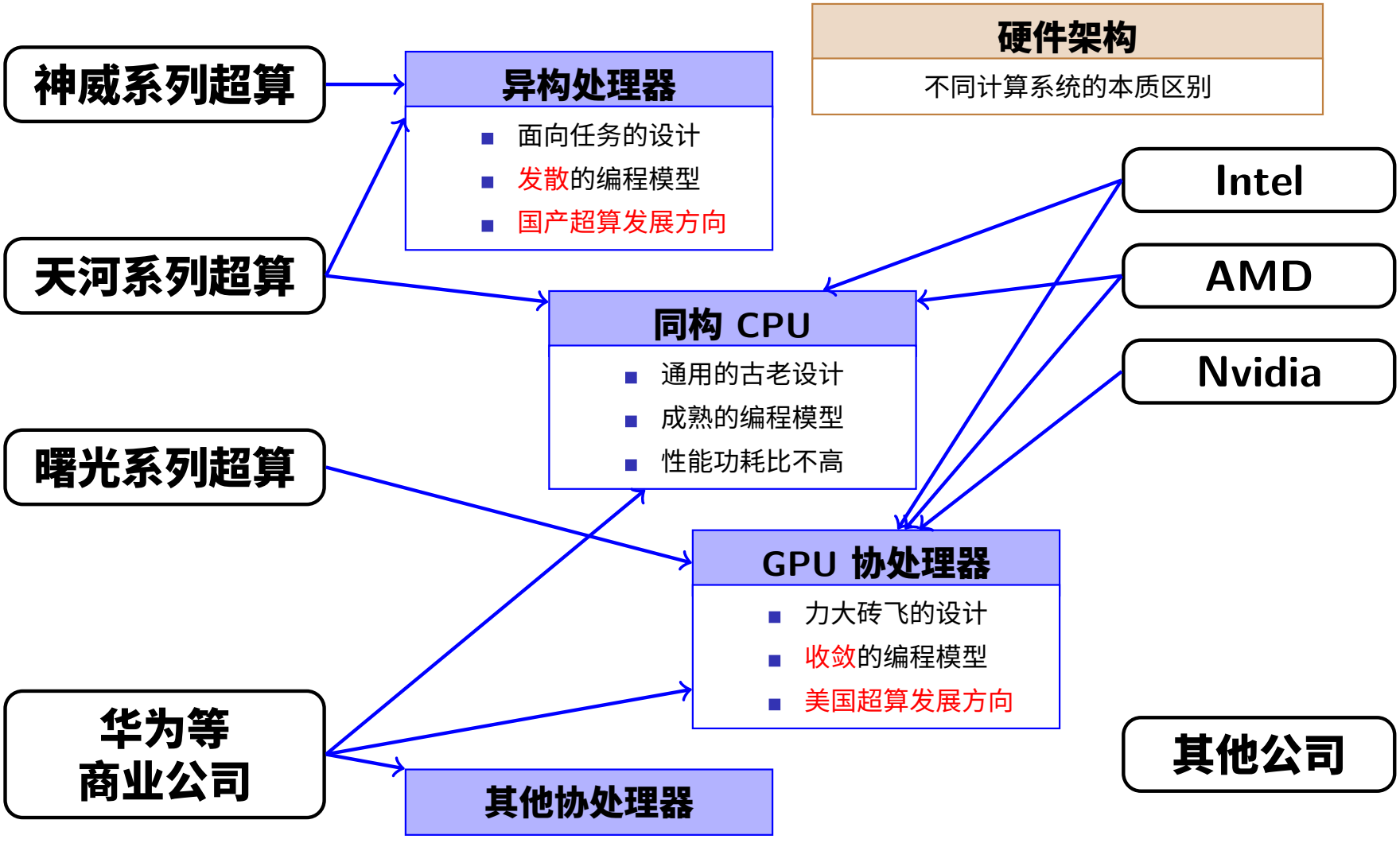
硬件的多样性

- 神威、天河、曙光三个架构都非常不同
- 同系列下一代超算架构变化也非常大
- 未来超算仍处于快速演化过程中
- \+ 所以软件设计必须挖掘共性特点、面向未来趋势。

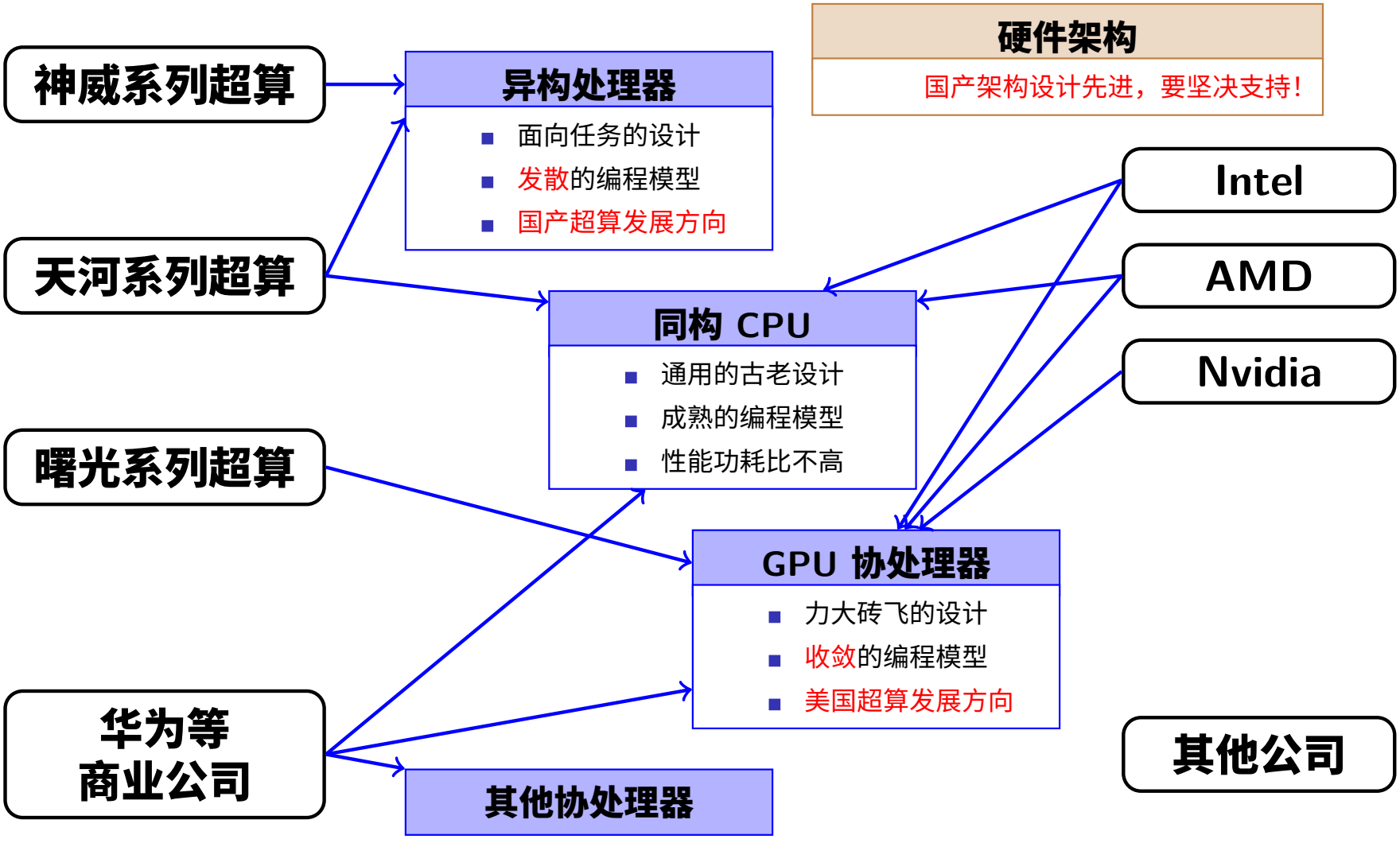
基础软件刚刚起步

- 新天河系统、神威系统仍不完善，急需大量基础软件支持。
- 曙光系统有开源社区支持，但也缺少很多基础软件。
- \+ 所以软件研发需要从基础软件做起，但同时也抛弃了历史包袱：一张白纸绘新图。

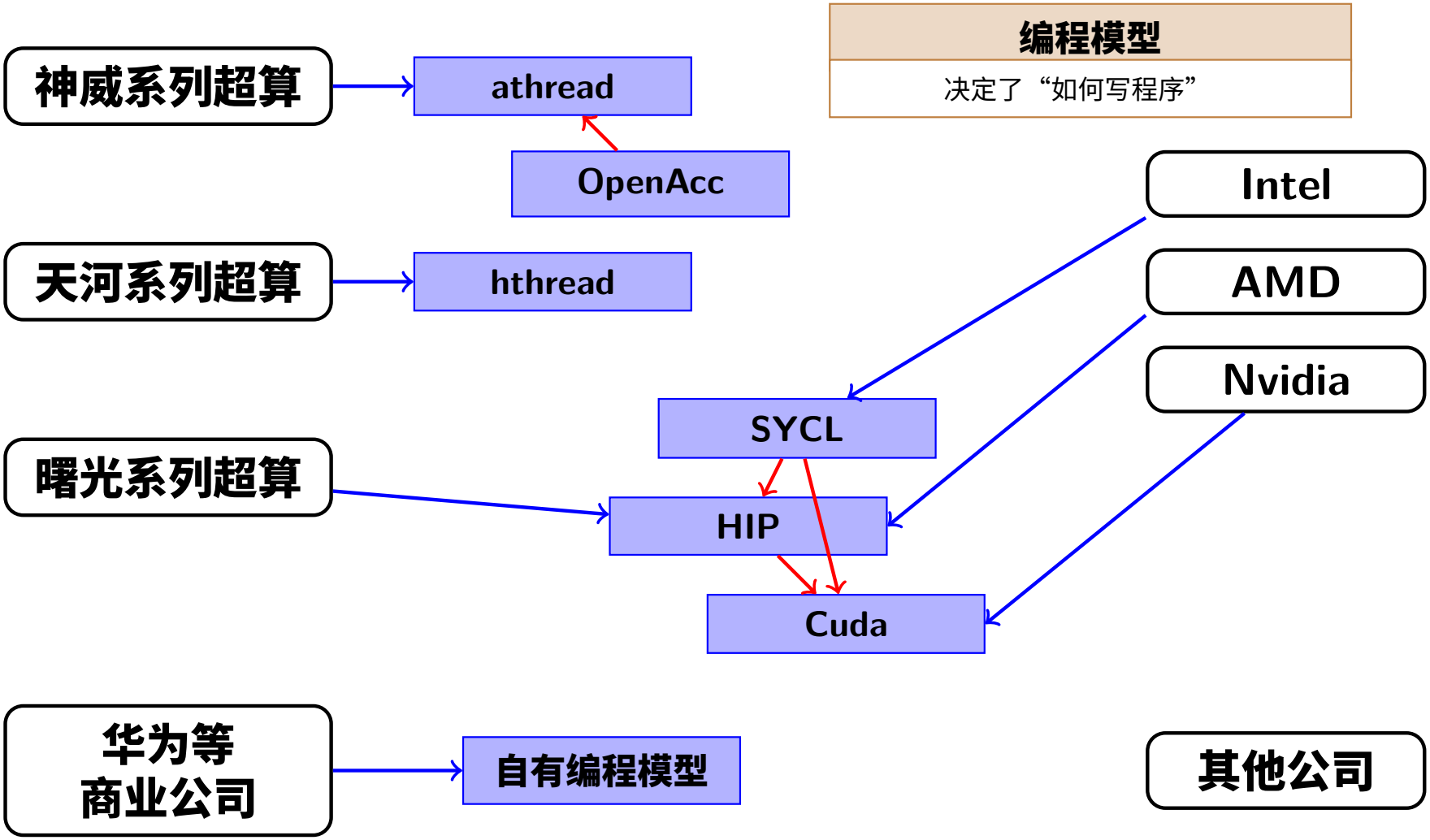
超级计算机的主要处理器平台



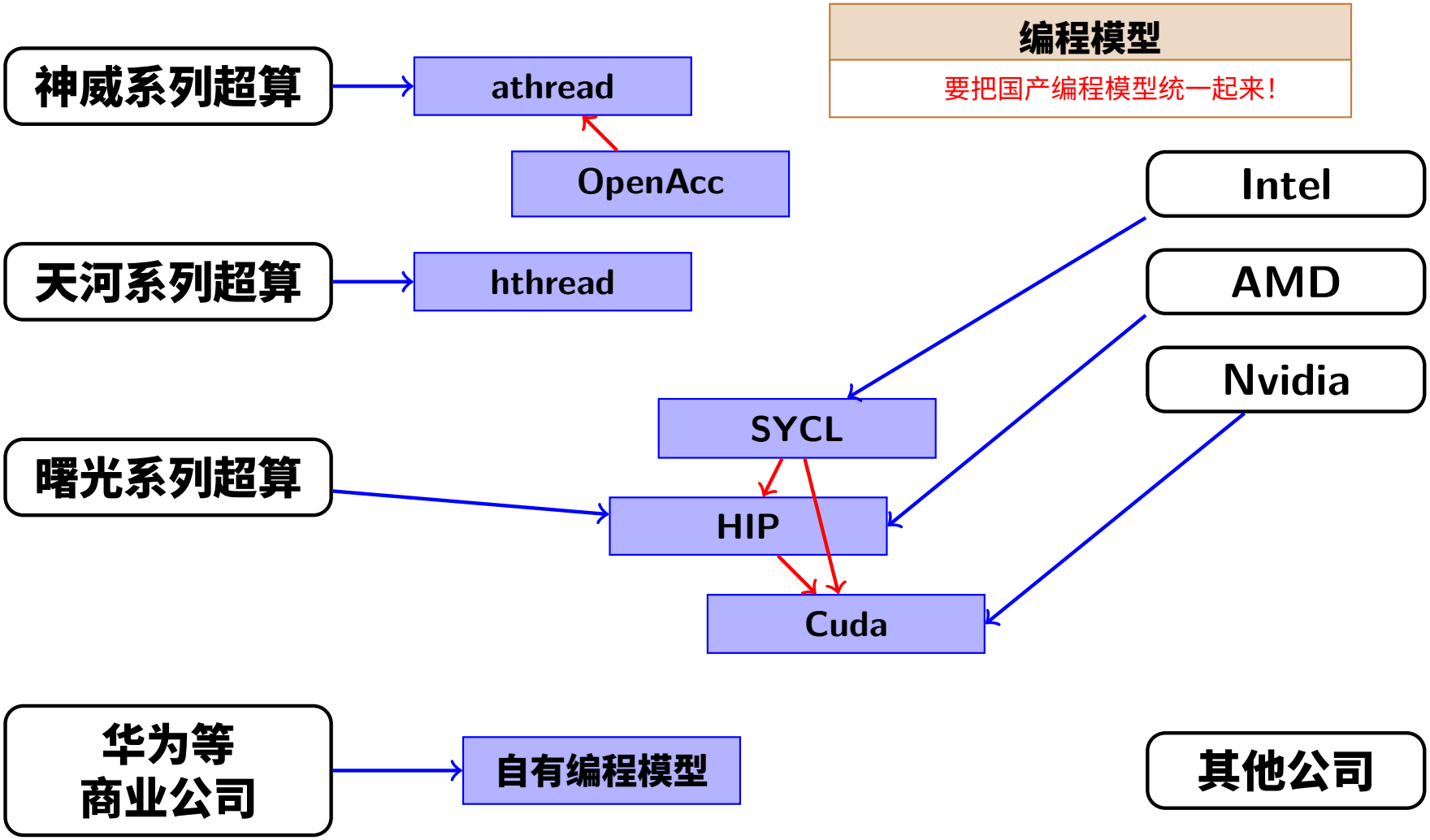
超级计算机的主要处理器平台



超级计算机的主要处理器平台



超级计算机的主要处理器平台



并行编程模型回顾

常用并行编程模型

- 格点 QCD 定制并行模型：QDP、Grid
- 进程并行：MPI
- 核心函数线程并行：CUDA、HIP、athread
- 匿名函数线程并行：SYCL、KOKKOS、RAJA
- 代码段线程并行：OpenMP、OpenAcc
- 指令并行：SIMD

- 以上编程模型在结构上分为两大类：对称、主从。
- 格点 QCD 软件通常混合采用对称和主从模型实现二级并行，代码结构很复杂。
- 以上模型中，除了 KOKKOS 外，大多数模型没有系统地处理多层次异构内存。
- 需要根据彻底的异构前提，设计新的编程模型。

未来科学计算技术的四个发展方向

自动代码生成

- 这是当前热点：元编程技术
- 路径：公式 $\rightarrow$ 中间表示 $\rightarrow$ 代码
- 是其他三个发展方向的基础设施
  - 参数化生成、面向平台生成、低代码界面

跨平台部署和调度

- 大数据框架是关键基础设施
  - Spark、Map-Reduce、Pregel、……
- 传统科学计算软件在技术上已落后
- 急需“高性能 + 可移植”的两全设计

自动代码优化

- 人工智能相关技术正在跃进
  - 科学计算软件要准备好借东风
- 优化理论和智能算法有发展潜力
- 需要完整的支撑平台来测试和部署

适合科研用户的人机界面

- “物理的归物理，计算的归计算”
  - 这是目前科学计算的最大痛点
- 面向物理公式的编程模型创新
- 面向科研工作的敏捷开发平台

格点 QCD 在四个方向上的需求和解决方案

MetaTensor：自动代码生成

- 物理计算基于复杂的公式
- 所有公式都可以转写为广义的“张量表达式”

张量表达式计算

DDQ：高效跨平台调度器

- 格点 QCD 的算法结构主要是循环迭代
- 需要大规模并行、需要高效数据交换

循环迭代

拆解算法流程

格点 QCD 计算软件

拆解计算程序

纯计算部分

- 与物理结果无关，即为优化空间
- 参数化此空间，开展自动优化研究

Trene：通用自动优化示例

纯物理部分

- 隔离计算细节，面向物理公式
- 适合做科研用户界面的编程模型

x 语言：图形化编程语言

研发贡献者

- 中国科学院高能物理研究所

  宫明、陈莹、刘朝峰、毕玉江、孙玮、施春江、蒋翔宇

- 北京航空航天大学

  栾钟治、韩斌、王御臣、肖敏毅、房歆哲、龚煜涵、马世清、刘星宇、李根、王逸杰、李亦白

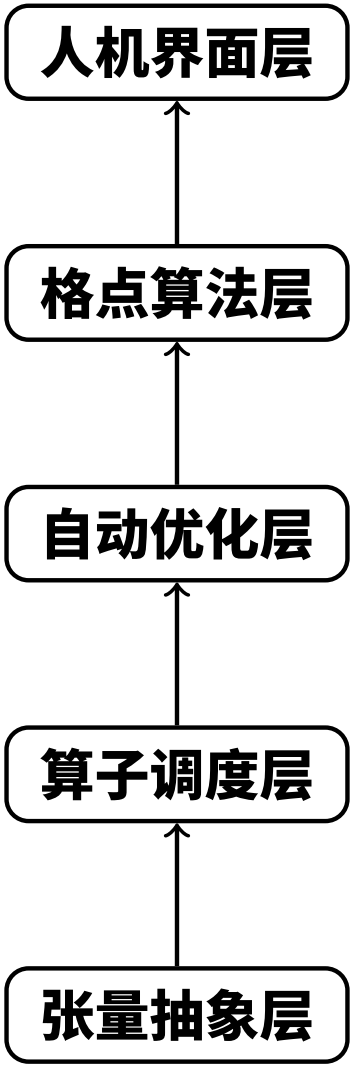
- 中国科学院计算机网络信息中心

  徐顺、张克龙、韩秉豫、张术飞

- 中国科学院理论物理研究所

  王建成

格点 QCD 新软件框架



格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

面向张量的代码生成器：MetaTensor

- 对所有的数据有统一的表示：数据描述 (datadesc)
- 对程序语法树有统一的表示：x 语言块 (block)
- 自身功能组织为统一的插口：插件系统 (plugin)
- 对软硬件环境有统一的管理：编译目标 (target)

格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

面向张量的代码生成器：MetaTensor

- 对所有的数据有统一的表示：数据描述 (datadesc)
- 对程序语法树有统一的表示：x 语言块 (block)
- 自身功能组织为统一的插口：插件系统 (plugin)
- 对软硬件环境有统一的管理：编译目标 (target)

MetaTensor 的数据描述

- 数据有“维度”和“指标”的结构，像多维数组
- 数据可以是稀疏的
- 编译期和运行时的数据是统一描述的

格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

面向张量的代码生成器：MetaTensor

- 对所有的数据有统一的表示：数据描述 (datadesc)
- 对程序语法树有统一的表示：x 语言块 (block)
- 自身功能组织为统一的插口：插件系统 (plugin)
- 对软硬件环境有统一的管理：编译目标 (target)

MetaTensor 的中间表示

- MetaTensor 的中间表示 (IR) 是 x 语言的子集
- x 语言是自主设计的图形化的编程语言
- MetaTensor 用 x 语言的“块”结构来描述张量表达式

格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

面向张量的代码生成器：MetaTensor

- 对所有的数据有统一的表示：数据描述 (datadesc)
- 对程序语法树有统一的表示：x 语言块 (block)
- 自身功能组织为统一的插口：插件系统 (plugin)
- 对软硬件环境有统一的管理：编译目标 (target)

MetaTensor 的插件系统

- 包含且仅包含四个定义良好的插口：
  - 复制 (datacopy) 插口
  - 求值 (block2data) 插口
  - 优化 (optimize) 插口
  - 编译 (compile) 插口
- 强大而灵活，足够描述所有需要的功能！

格点 QCD 新软件框架

编译期的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 表达式的分析越早越好：元编程

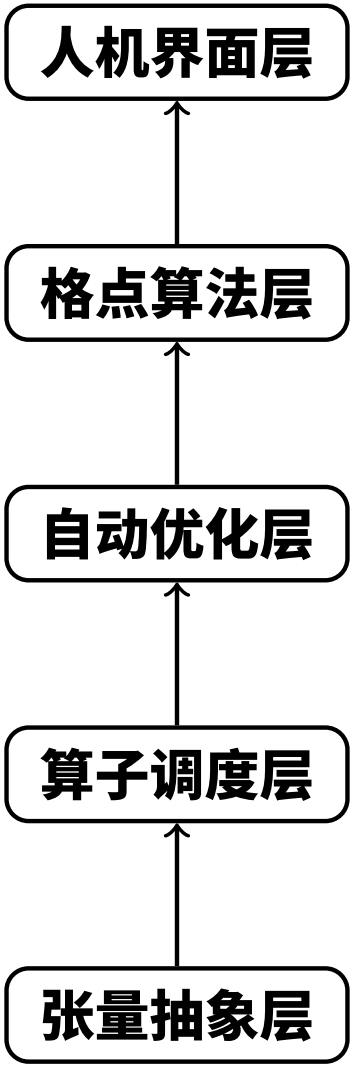
面向张量的代码生成器：MetaTensor

- 对所有的数据有统一的表示：数据描述 (datadesc)
- 对程序语法树有统一的表示：x 语言块 (block)
- 自身功能组织为统一的插口：插件系统 (plugin)
- 对软硬件环境有统一的管理：编译目标 (target)

MetaTensor 的编译目标

- 编译目标是对插件集合的选取方案
- 每个插口上安排一条插件链
- 为不同的软硬件环境设置不同的编译目标
- 目前的代码编译到 C 语言，保留其他后端的兼容性
- 插件和编译目标的维护工作是脱耦的

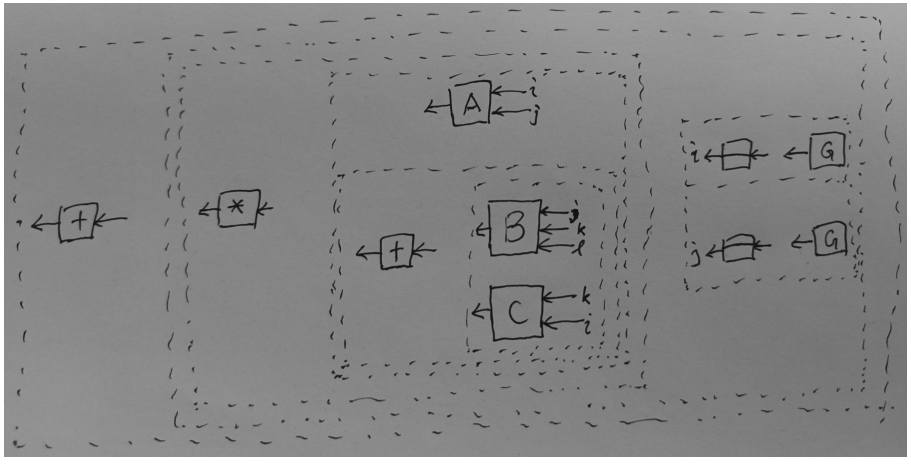
格点 QCD 新软件框架



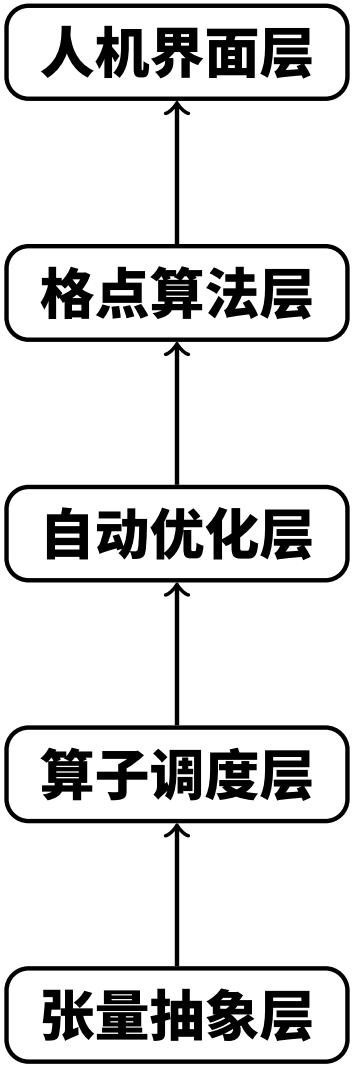
需要计算的张量表达式：

$$result_{k,l} = \sum_{i,j} A_{i,j} * (B_{j,k,l} + C_{k,i})$$

用 x 语言转写为：



格点 QCD 新软件框架



MetaTensor 把它编译到 C 语言：

```c
#include <stdint.h>
#include <stdlib.h>
void calc(void *res, void *a , void *b, void *c){
        double*var_5_=malloc(2048);
        {
                double*var_2_=(double*)((char *)(b)+0);
                double*var_3_=(double*)((char *)(c)+0);
                double*var_4_=(double*)((char *)(var_5_)+0);
                int32_t var_7_=0;
                int32_t var_8_=0;
                int32_t var_9_=0;
                const int32_t var_10_[4]={ 0,16,32,48};
                const int32_t var_11_[4]={ 0,64,128,192};
                for(int32_t var_12_=0; var_12_<4; var_12_++){
                        int32_t var_13_=var_7_+var_10_[var_12_];
                        int32_t var_14_=var_9_+var_11_[var_12_];
                        const int32_t var_15_[4]={ 0,4,8,12};
                        const int32_t var_16_[4]={ 0,4,8,12};
                        const int32_t var_17_[4]={ 0,16,32,48};
                        for(int32_t var_18_=0; var_18_<4; var_18_++){
                                int32_t var_19_=var_13_+var_15_[var_18_];
                                int32_t var_20_=var_8_+var_16_[var_18_];
                                int32_t var_21_=var_14_+var_17_[var_18_];
                                const int32_t var_22_[4]={ 0,1,2,3};
                                const int32_t var_23_[4]={ 0,4,8,12};
                                for(int32_t var_24_=0; var_24_<4; var_24_++){
                                        int32_t var_25_=var_19_+var_22_[var_24_];
                                        int32_t var_26_=var_21_+var_23_[var_24_];
                                        const int32_t var_27_[4]={ 0,1,2,3};
                                        const int32_t var_28_[4]={ 0,1,2,3};
                                        for(int32_t var_29_=0; var_29_<4; var_29_++){
                                                int32_t var_30_=var_20_+var_27_[var_29_];
                                                int32_t var_31_=var_26_+var_28_[var_29_];
                                                var_4_[var_31_]=0+var_2_[var_25_]+var_3_[var_30_];
                                        }}}}}
        double*var_37_=malloc(2048);
        {
                double*var_34_=(double*)((char *)(a)+0);
                double*var_35_=(double*)((char *)(var_5_)+0);
                double*var_36_=(double*)((char *)(var_37_)+0);
                int32_t var_39_=0;
                int32_t var_40_=0;
                int32_t var_41_=0;
                const int32_t var_42_[4]={ 0,4,8,12};
                const int32_t var_43_[4]={ 0,1,2,3};
                const int32_t var_44_[4]={ 0,64,128,192};
                for(int32_t var_45_=0; var_45_<4; var_45_++){
                        int32_t var_46_=var_39_+var_42_[var_45_];
                        int32_t var_47_=var_40_+var_43_[var_45_];
```

格点 QCD 新软件框架

运行时的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 格点 QCD 计算特点：大规模并行、数据带宽要求高
- 国产超算架构多样，生态环境建设落后

格点 QCD 新软件框架

运行时的设计思考

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
- 格点 QCD 计算特点：大规模并行、数据带宽要求高
- 国产超算架构多样，生态环境建设落后

跨平台运行时调度器：DDQ

- 跨平台！跨平台！跨平台！
  - 用协程模型写任意软硬件平台的驱动程序
- 通用的算子模型和内存对象模型，包含自动内存管理
- 工作流用有环图描述，可以运行时动态生成，以此实现复杂的动态控制流
- 协程并发机制，自动实现双缓冲、后台 IO 等优化技术
- 计算任务与调度逻辑完全脱耦，非常适合作为通用的计算服务器

格点 QCD 新软件框架

运行时的设计思考

- ……

跨平台运行时调度器：DDQ

- ……驱动程序……
- ……算子……

驱动程序

- DDQ 通过驱动程序在软硬件平台上运行算子
- 只需要几百行规模的代码即可实现一个驱动程序
- 目前实现的驱动程序：
  - 任意处理器的“本地”运行
  - x86/arm 的 fork 进程
  - x86/arm 的 pthread 线程
  - 包含MPI/QMP 的 pthread 驱动
  - 天河三 DSP 设备上的线程
  - CUDA 设备 (初步)
- 考虑扩展到：
  - CUDA/HIP/SYCL 等常用异构平台、申威主/从核上的线程、通过网络的 RPC 调用

格点 QCD 新软件框架

运行时的设计思考

- ……

跨平台运行时调度器：DDQ

- ……驱动程序……
- ……算子……

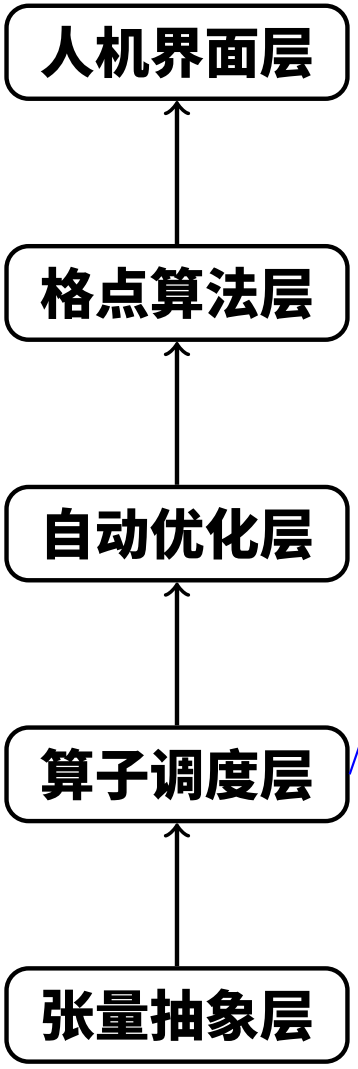
算子

- DDQ 的算子声明：

```
typedef task_ret    task_f(void **inputs, void **outputs);
```

  - 通用性非常强，可以描述几乎所有功能
- 目前实现的算子：
  - 文件 IO 算子、命令行调用算子
  - 主内存和 CUDA 显存的 memcpy 算子
  - zmq 网络接口算子、Redis 数据库访问算子
  - 天河三 DSP 设备上的 Gromacs 热点函数算子
  - 天河三 DSP 设备上的软 cache 算子
  - 格点 QCD 的 Chroma 包装算子
  - 脚本编译算子

格点 QCD 新软件框架



运行时的设计思考

- ……

跨平台运行时调度器：DDQ

- ……驱动程序……
- ……算子……

脚本与复合算子

- DDQ 提供一个定制的脚本环境
- 脚本负责描述算子构成的有向图，实现程序控制流
- 可以用脚本把多个算子组合成为复合算子，从而层迭构建各种粒度的模块
- 脚本的语法是图灵完全的
- 所有不同功能的基本算子和复合算子，整理为算子库（oplib），以树形结构规划，适合分布式开发和维护

格点 QCD 新软件框架

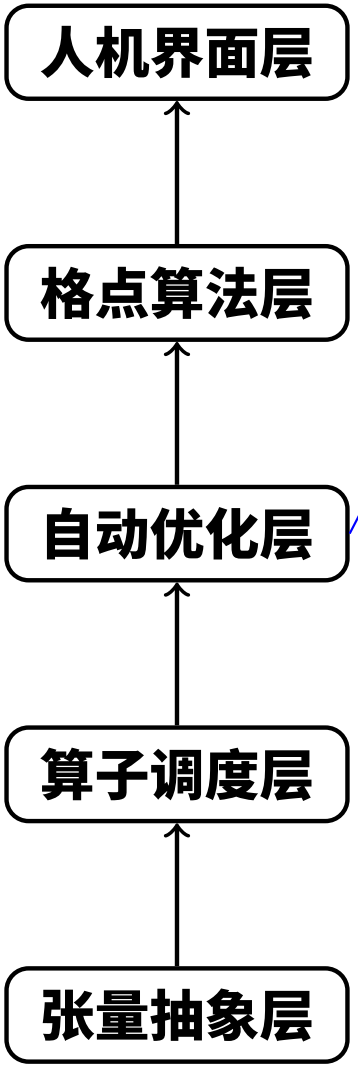
举例：计算圆周率 π

$$\frac{1}{4}(\pi-3)=\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{2\,k\,(2\,k+1)\,(2\,k+2)}=\frac{1}{2\cdot3\cdot4}-\frac{1}{4\cdot5\cdot6}+\frac{1}{6\cdot7\cdot8}-\dots$$

用 DDQ 脚本转写为：

```
--------------------------------------------------------------------------------
| tests/pi.ddq :        |                                                      |
|-----------------------                                                       |
| print := load_op(std/real/print)                                             |
| add := load_op(std/real/add)                                                 |
| mul4 := load_op(std/real/mul4)                                               |
| mul := load_op(std/real/mul)                                                 |
| inv := load_op(std/real/inv)                                                 |
| init := load_op(std/real/init)                                               |
| sleep := load_op(std/time/sleep)                                             |
|                                                                              |
| print < [pi] < add < [pi term]                                               |
| [term] < mul4 < [s ia ib ic]          ---------------------------------------|
|                                       | std/real/mul4 :        |             |
| [s] < mul < [s -1.0]                  |-----------------------              |
| [ia] < inv < [a] < add < [a 2.0]      | [s] < newop < [a b c d]              |
| [ib] < inv < [b] < add < [b 2.0]      | [s] < load_op(std/real/mul) < [abc d]|
| [ic] < inv < [c] < add < [c 2.0]      | [abc] < load_op(std/real/mul) < [ab c]|
|                                       | [ab] < load_op(std/real/mul) < [a b] |
| [pi] < init < [3.0]                   |--------------------------------------|
| [s] < init < [4.0]                    | std/real/inv :         |             |
| [a] < init < [2.0]                    |-----------------------              |
| [b] < init < [3.0]                    | [out] < newop < [in]                 |
| [c] < init < [4.0]                    | [out] < load_op(std/real/div) < [1.0 in]|
|------------------------------------------------------------------------------|
| std/real/mul.ddq :    |                                                      |
|-----------------------                                                       |
| newop = load_so(std/real/mul)                                                |
| newop.processor = "direct"                                                   |
| [load_type(std/real)] < newop < [load_type(std/real) load_type(std/real)]    |
--------------------------------------------------------------------------------
```

格点 QCD 新软件框架



自动优化的设计思考

- 自动优化的关键：优化空间
- 好的编程模型和软件框架，应该把计算过程的“有关部分”和“无关部分”充分分开，后者就是优化空间
- 应当为优化空间构建适当的表示：
  - 参数表、变换图、大模型参数……

格点 QCD 新软件框架

自动优化的设计思考

- 自动优化的关键：优化空间
- 好的编程模型和软件框架，应该把计算过程的“有关部分”和“无关部分”充分分开，后者就是优化空间
- 应当为优化空间构建适当的表示：
  - 参数表、变换图、大模型参数……

优化空间的图表示

- MetaTensor 的数据描述和 IR、DDQ 的算子图，都是图的顶点；对它们的等价变换，构成图的边
- 自动优化问题转变为图搜索问题
  - 深度/广度优先搜索、A* 搜索、遗传算法、……

格点 QCD 新软件框架

自动优化的设计思考

- 自动优化的关键：优化空间
- 好的编程模型和软件框架，应该把计算过程的“有关部分”和“无关部分”充分分开，后者就是优化空间
- 应当为优化空间构建适当的表示：
  - 参数表、变换图、大模型参数……

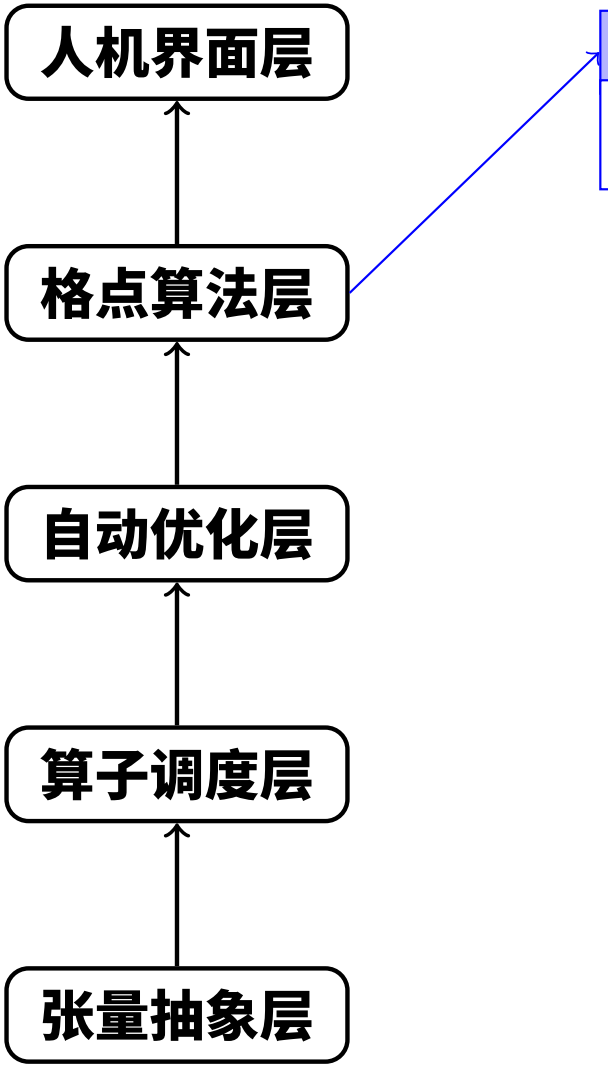
优化空间的图表示

- MetaTensor 的数据描述和 IR、DDQ 的算子图，都是图的顶点；对它们的等价变换，构成图的边
- 自动优化问题转变为图搜索问题
  - 深度/广度优先搜索、A* 搜索、遗传算法、……

针对树结构的通用优化器：Trene

- 采用遗传算法，变异是无损的
- 可用于代数表达式或程序 AST
  - 正在测试在量子计算中的应用

格点 QCD 新软件框架



应用软件的设计思考

- 从更抽象的层面上审视应用需求，构建可重用、可移植、高性能的代码

格点 QCD 新软件框架

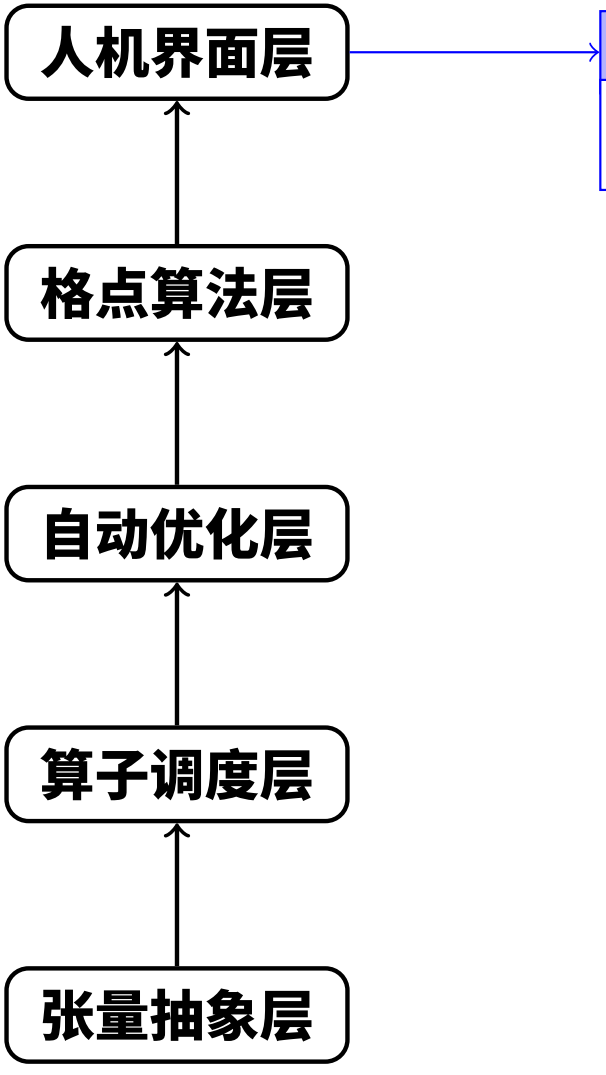
应用软件的设计思考

- 从更抽象的层面上审视应用需求，构建可重用、可移植、高性能的代码

格点 QCD 应用软件开发

- 格点 QCD 算法归结为：张量表达式计算、循环迭代
  - 所有张量计算核心都用 MetaTensor 自动产生代码，变成 DDQ 算子
  - 把算法中的循环迭代的流程用 DDQ 脚本构建为图，组合为更大的算子
  - 层迭构建不同尺度的算子，直到描述完整的计算任务
- 用自动优化工具透明地优化计算任务
- 把计算任务提交给部署在各平台上的 DDQ 运行

格点 QCD 新软件框架



人机界面的设计思考

- 好的编程模型，应该把计算过程的“有关部分”和“无关部分”充分分开，人机界面只操作前者

格点 QCD 新软件框架

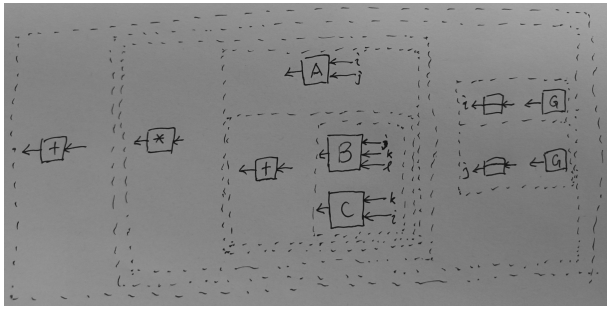
人机界面的设计思考

- 好的编程模型，应该把计算过程的“有关部分”和“无关部分”充分分开，人机界面只操作前者

x 语言

- x 语言是一个自主设计的新编程语言
- 极简原则、纯函数式、静态强类型
- 只有一种数据/程序类型：块
- 适合图形界面操作
- 正在开发基于 HTML5 的人机界面

x 语言示例：



格点 QCD 新软件框架

人机界面的设计思考

- 好的编程模型，应该把计算过程的“有关部分”和“无关部分”充分分开，人机界面只操作前者

DDQ 脚本环境

- DDQ 脚本环境用于编辑 DDQ 计算任务
- 脚本是强类型的，可以保证 DDQ 运行时的类型安全
- 可以通过脚本，将算子组合为更大的算子，并储存在算子库 (oplib) 里
- 在 x 语言成熟后，DDQ 脚本可以从 x 语言生成

DDQ 脚本示例：

```
| std/real/mul.ddq :    |
|-----------------------
| newop = load_so(std/real/mul)
| newop.processor = "direct"
| [load_type(std/real)] < newop < [load_type(std/real) load_type(std/real)]
```

总结

- 格点 QCD 是从第一性原理出发非微扰地系统研究强子和强相互作用的高能物理前沿领域
- 我们正在为格点 QCD 编写新的软件框架：
  - 它的研发目标是通用于各类科学计算的
  - 它的技术设计是面向未来发展的

谢谢！