格点计算新软件框架构想

宫明

中科院高能所
CLQCD 合作组

2022.10.10

提纲

提纲

中国格点计算的软件危机

硬件架构的发展

向量机 -> 定制芯片 -> 集群 -> 异构

中国超算的特点

- 独立发展
- 产品线丰富
- 未来潜力大

中国格点计算的软件危机

硬件架构的发展

向量机 -> 定制芯片 -> 集群 -> 异构

中国超算的特点和困难

- 独立发展 ——与美国逐渐脱钩
- 产品线丰富 ——互不兼容
- 未来潜力大 ——版本快速变化

中国格点计算的软件危机

硬件架构的发展

向量机 -> 定制芯片 -> 集群 -> 异构

面向中国超算的特点和困难的对策

- 独立发展 ——与美国逐渐脱钩 ——不能跟随 USQCD/QUDA
- 产品线丰富 ——互不兼容 ——可移植性是痛点
- 未来潜力大 ——版本快速变化 ——要设计前瞻性的框架

中国格点计算的软件危机

硬件架构的发展

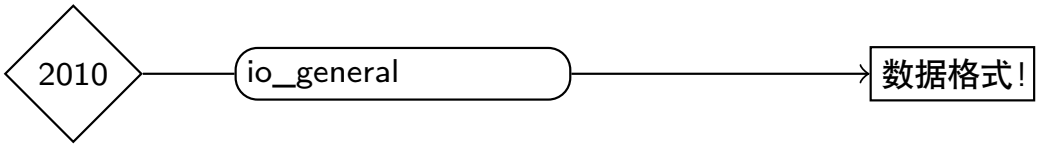
向量机 -> 定制芯片 -> 集群 -> 异构

面向中国超算的特点和困难的对策

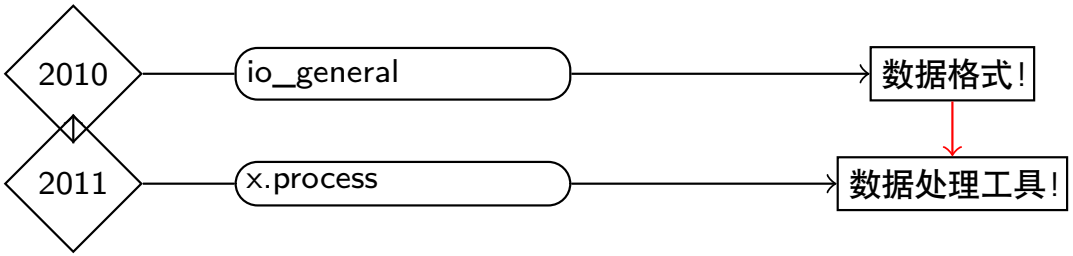
- 独立发展 ——与美国逐渐脱钩 ——不能跟随 USQCD/QUDA
- 产品线丰富 ——互不兼容 ——可移植性是痛点
- 未来潜力大 ——版本快速变化 ——要设计前瞻性的框架

重起炉灶，研发一个自己的程序框架，一个面向多样化大规模异构环境的、高效的、方便扩展的、有前瞻性的框架！

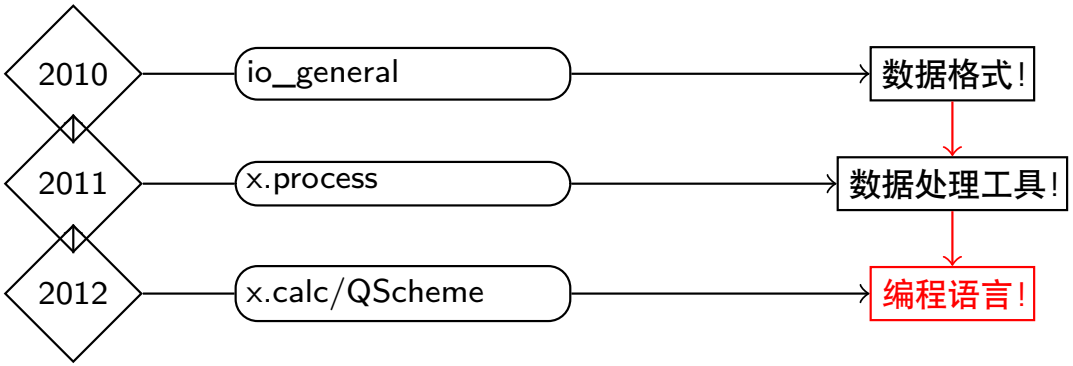
思考的脉络



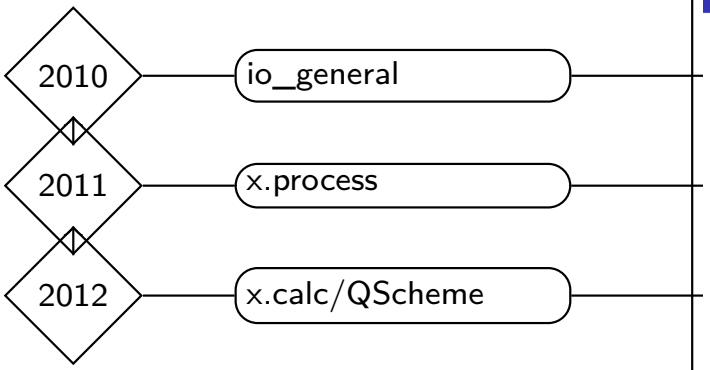
思考的脉络



思考的脉络


2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!

思考的脉络



栗子：MPI 并行化

I have a function

```
(define (my_func par1 par2 par3)
        ...
)
```

I have an MPI wrapper

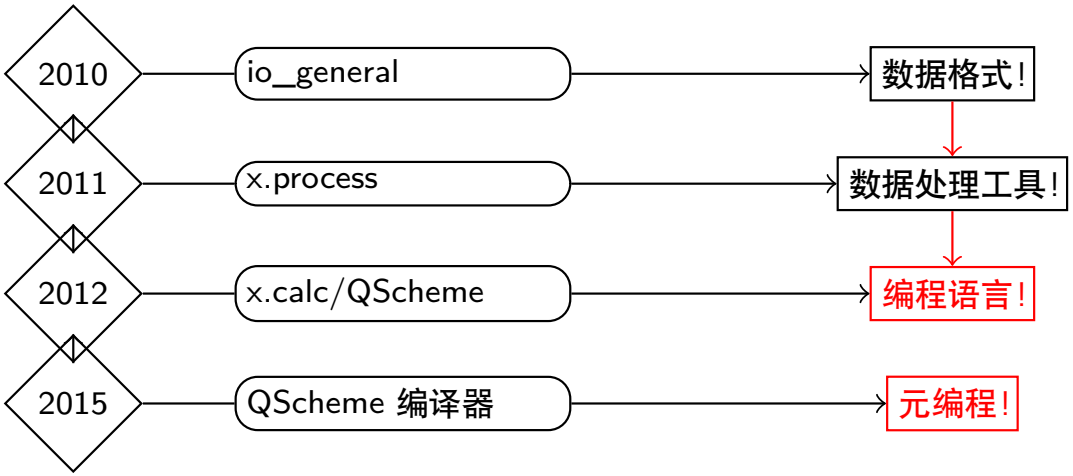
```
(define (x.grind/mpi f . dims)
        ...
)
```

MPI function!

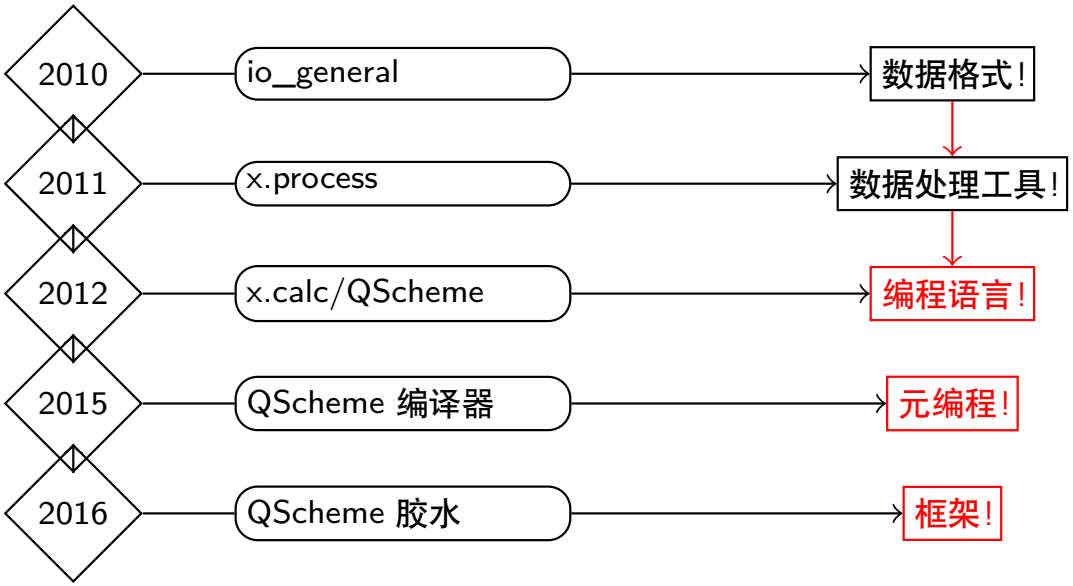
```
(define my_mpi_func
        (x.grind/mpi my_func 'mass 't))
```

宫明 CLQCD

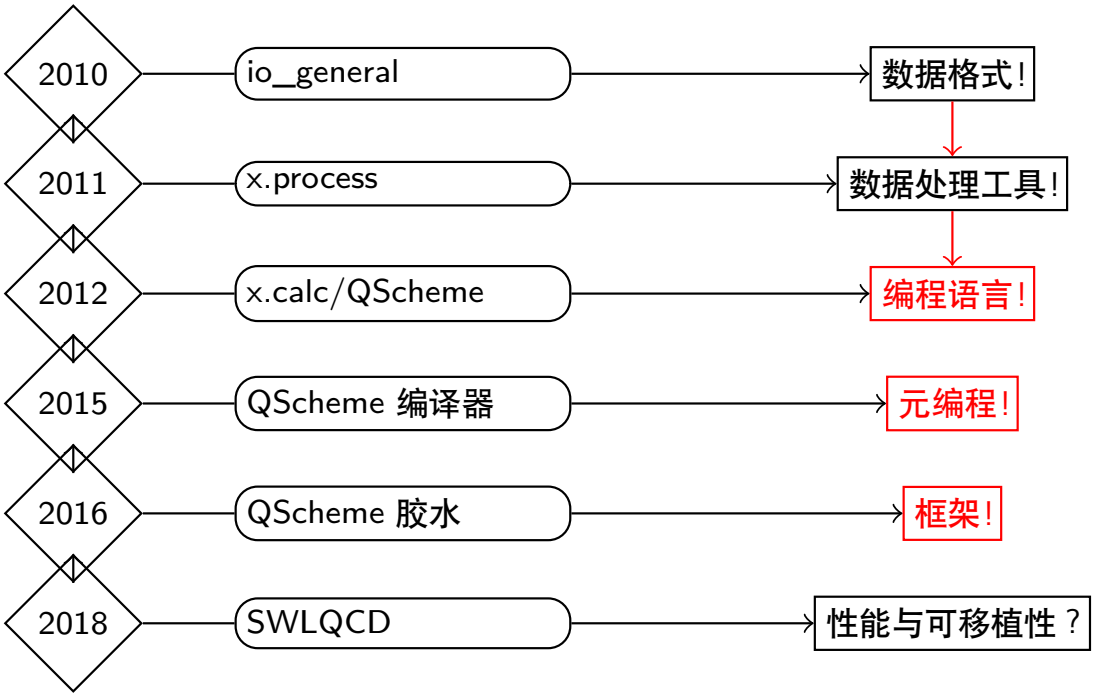
思考的脉络


2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!

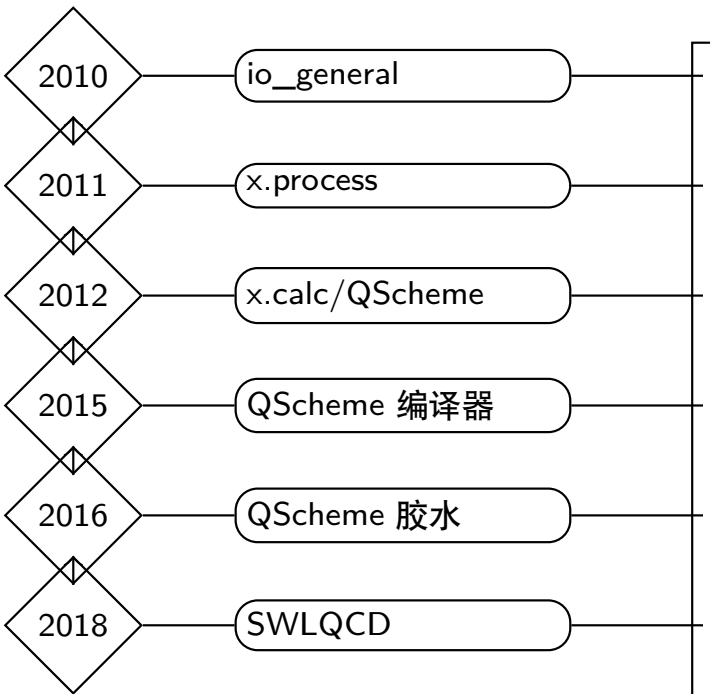
思考的脉络

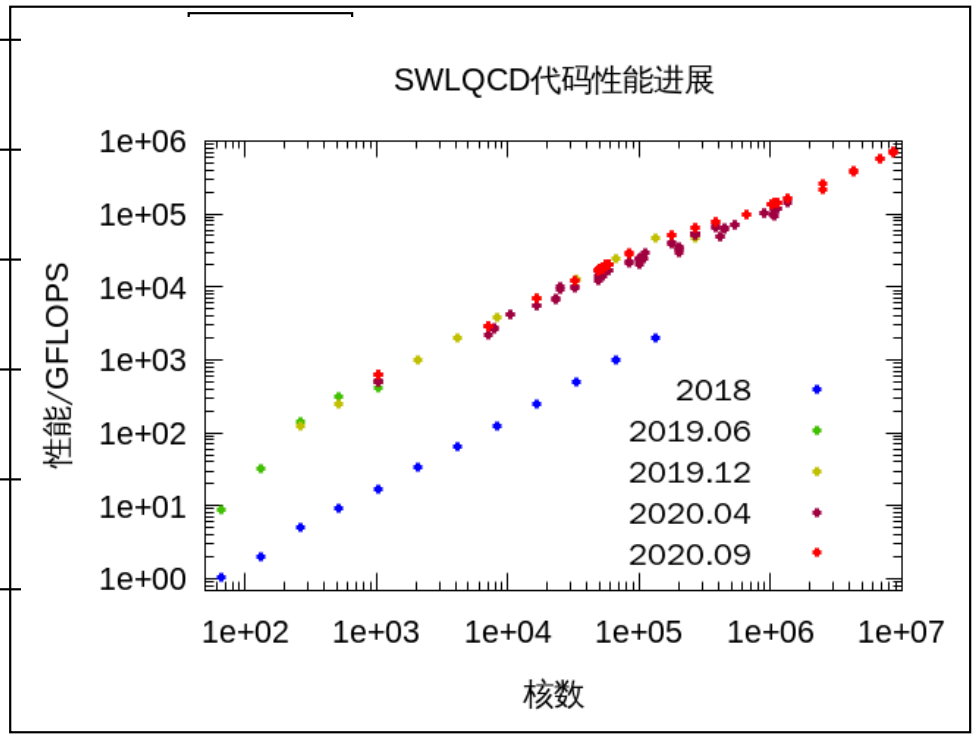

2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!
2016
QScheme 胶水
框架!

思考的脉络

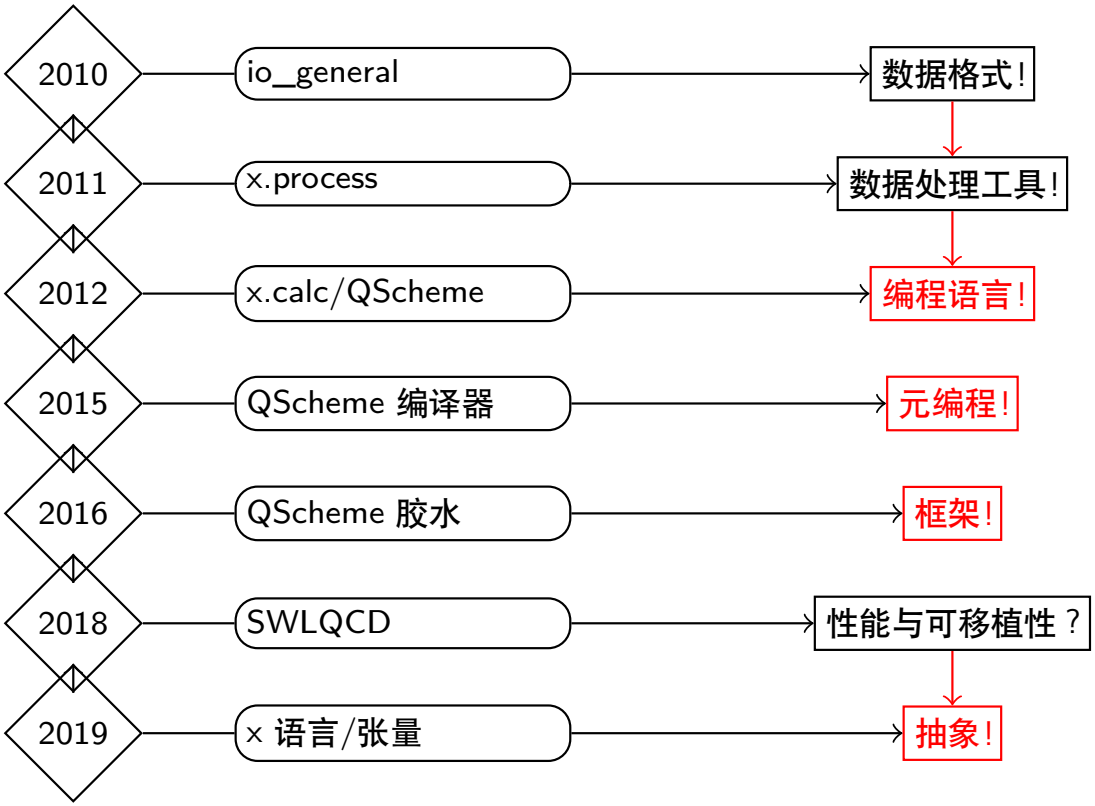

2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!
2016
QScheme 胶水
框架!
2018
SWLQCD
性能与可移植性？

思考的脉络

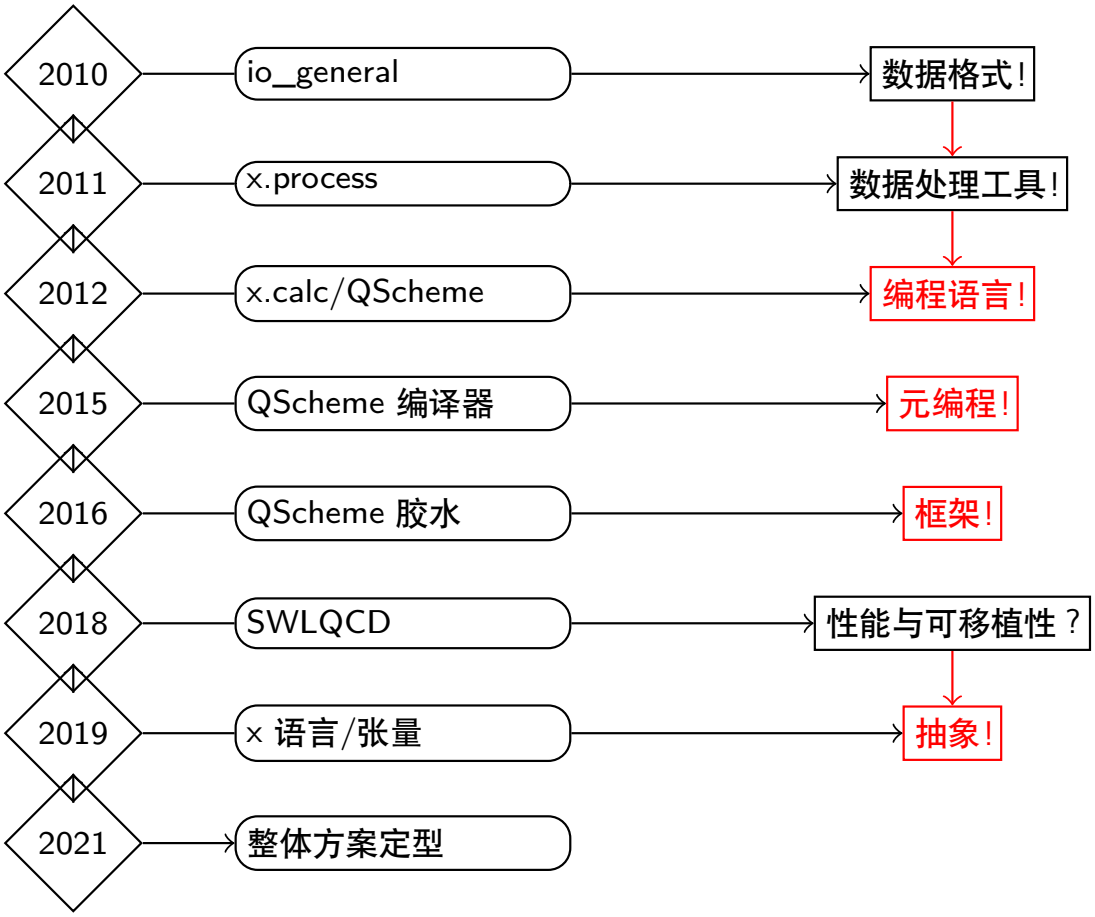

2010
io_general
2011
x.process
2012
x.calc/QScheme
2015
QScheme 编译器
2016
QScheme 胶水
2018
SWLQCD


SWLQCD代码性能进展
性能/GFLOPS
1e+06
1e+05
1e+04
1e+03
1e+02
1e+01
1e+00
1e+02
1e+03
1e+04
1e+05
1e+06
1e+07
核数
2018
2019.06
2019.12
2020.04
2020.09

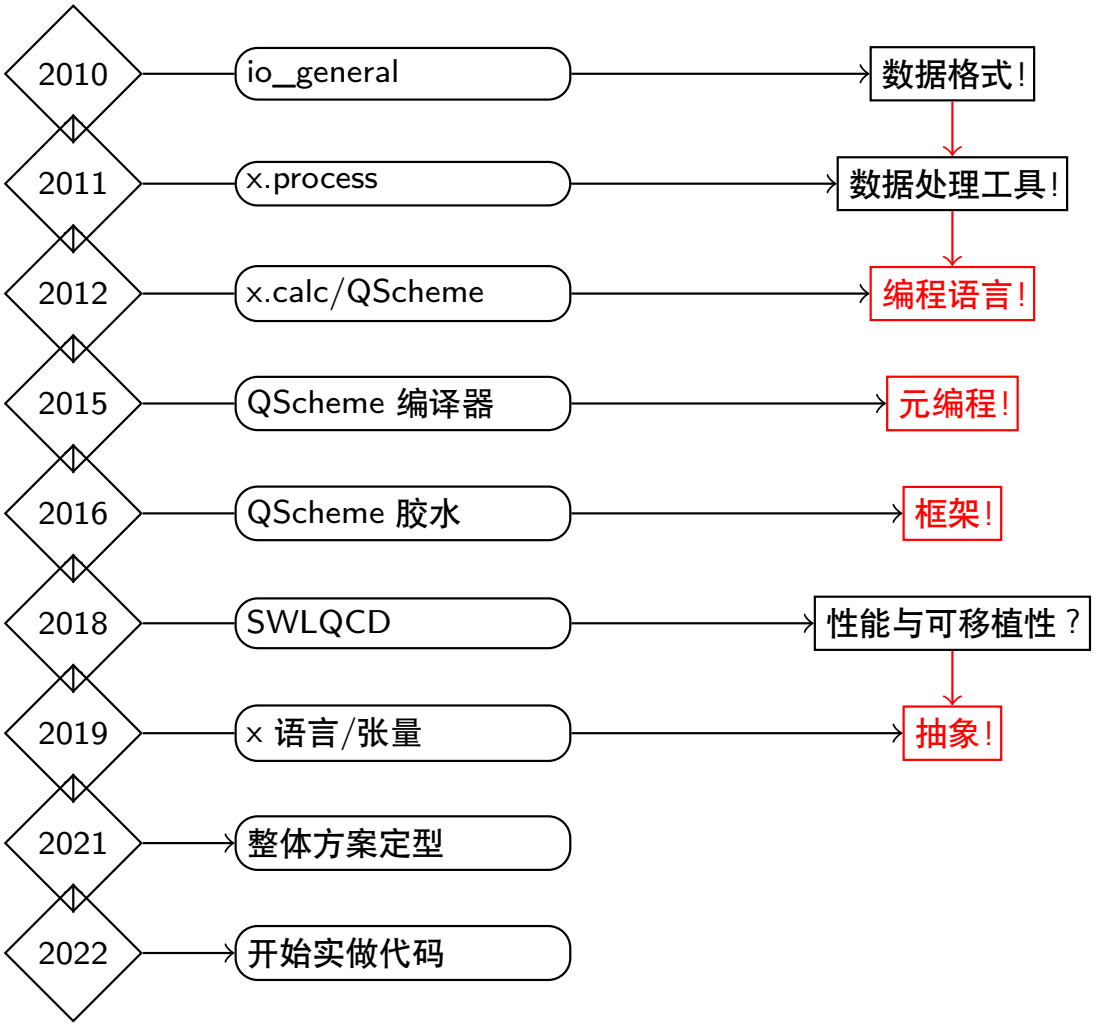
思考的脉络


2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!
2016
QScheme 胶水
框架!
2018
SWLQCD
性能与可移植性？
2019
x 语言/张量
抽象!

思考的脉络

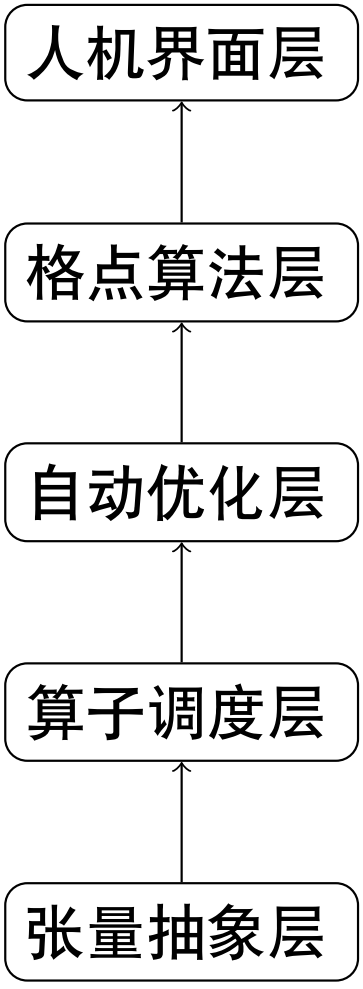

2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!
2016
QScheme 胶水
框架!
2018
SWLQCD
性能与可移植性？
2019
x 语言/张量
抽象!
2021
整体方案定型

思考的脉络

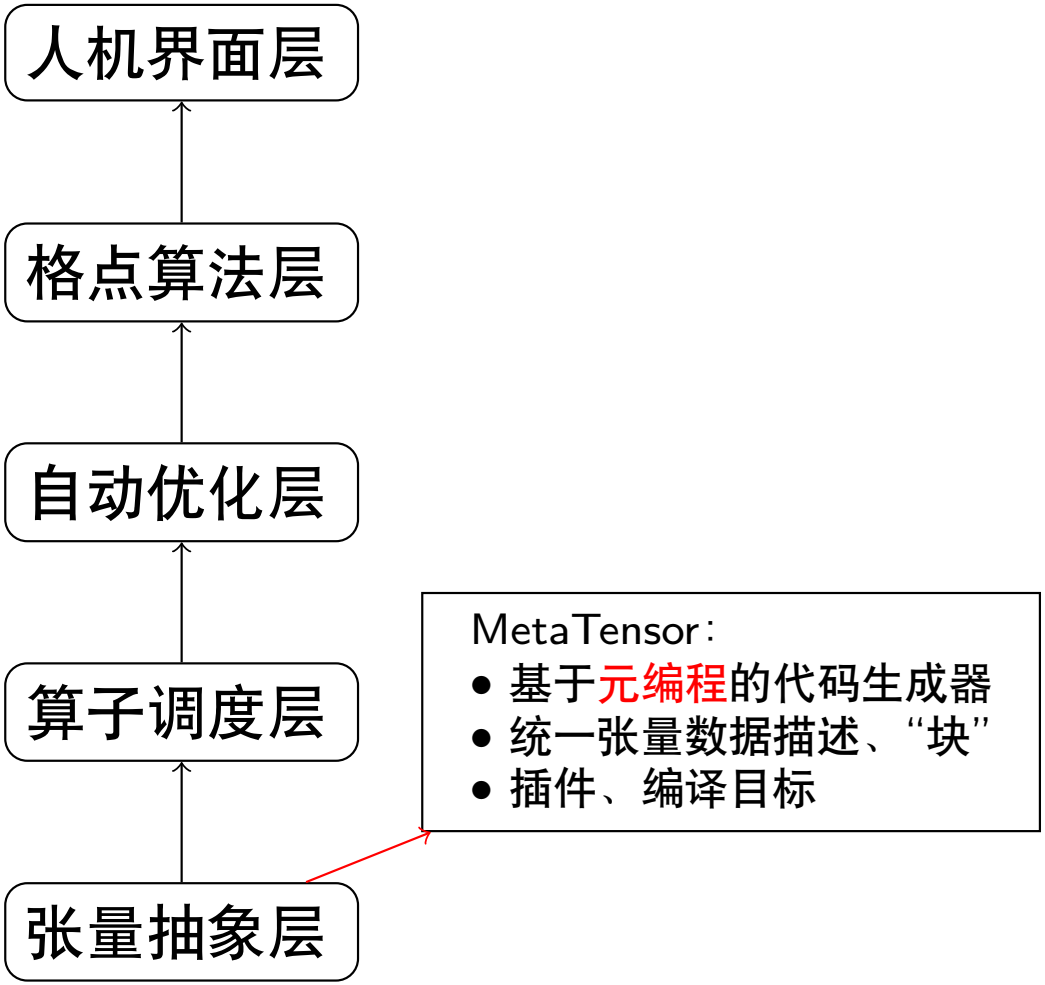

2010
io_general
数据格式!
2011
x.process
数据处理工具!
2012
x.calc/QScheme
编程语言!
2015
QScheme 编译器
元编程!
2016
QScheme 胶水
框架!
2018
SWLQCD
性能与可移植性？
2019
x 语言/张量
抽象!
2021
整体方案定型
2022
开始实做代码

提纲

新软件框架的结构



新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
MetaTensor:
• 基于元编程的代码生成器
• 统一张量数据描述、“块”
• 插件、编译目标

新软件框架的结构

人机界面层

格点算法

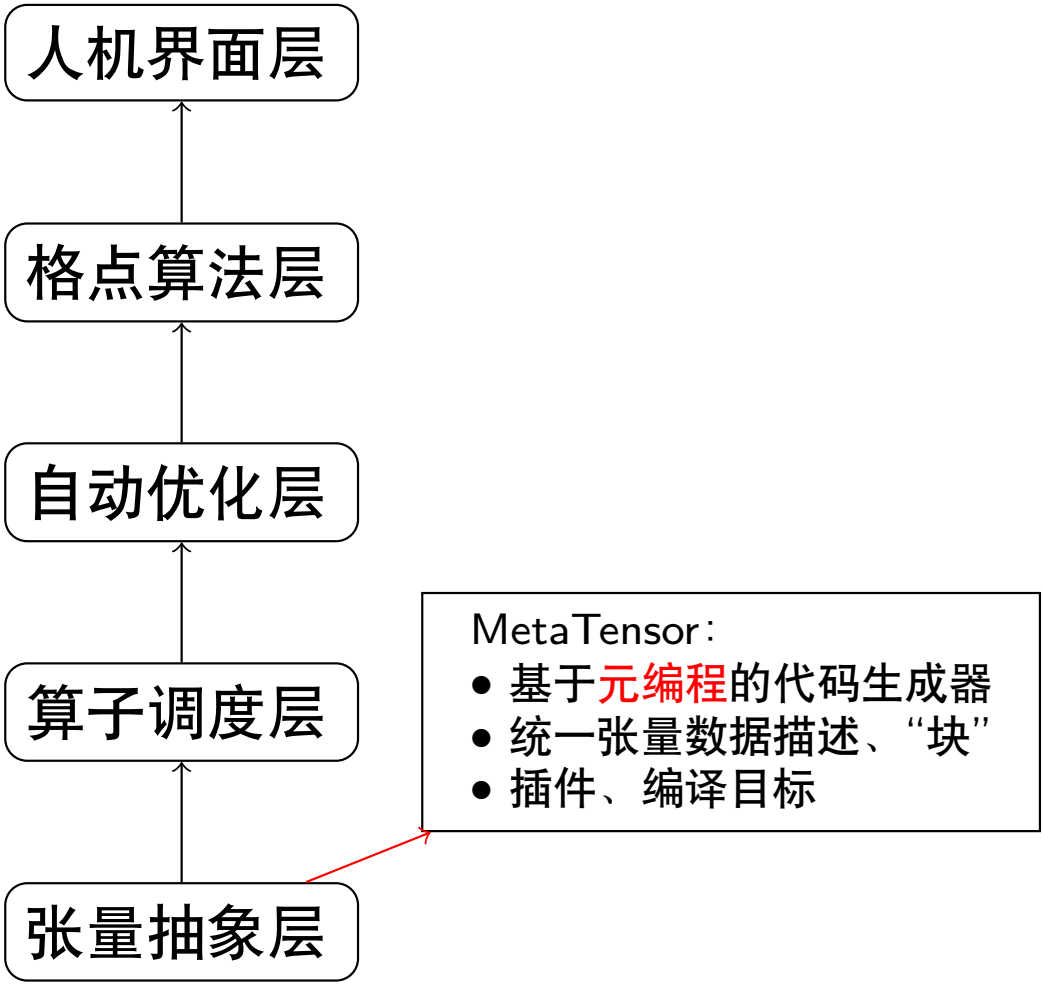
自动优化

算子调度

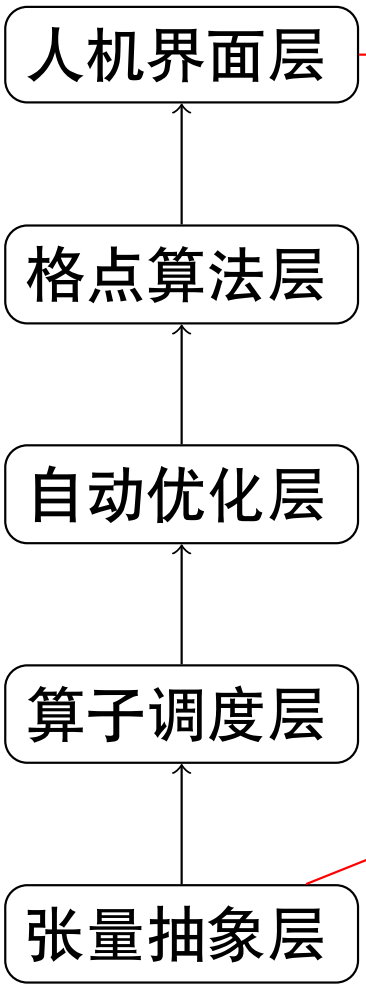
张量抽象

```
char *  t_dst_offset_p = token_new();   // 每个维度的各指标的偏移量列表
char *  t_src_offset_p = token_new();

emit_push(t, int_type, " ", t_dst_offset_p, "[", itoa(d_dst->n_entry), "]={", itoa(d_dst->offsets[0]));
for (i=1; i<d_dst->n_entry; i++)
        emit_push(t, ",", itoa(d_dst->offsets[i]));
emit_push(t, "};");

for (i_src=0; i_src<d->p[1]->n_dim && d_dst->dims[0]!=d->p[1]->dims[i_src]->dims[0]; i_src++);
if (i_src<d->p[1]->n_dim)
{
        dimdesc d_src = d->p[1]->dims[i_src];

        emit_assert(t, d_src->n_tuple==1);
        emit_assert(t, d_dst->n_entry==d_src->n_entry);
        emit_push(t, int_type, " ", t_src_offset_p, "[", itoa(d_src->n_entry), "]={");
        {
                for (j=0; j<d_src->n_entry && d_dst->indices[0]!=d_src->indices[j*d_src->n_tuple]; j++);
                emit_assert(t, j<d_src->n_entry);
                emit_push(t, itoa(d_src->offsets[j]));
        }
        for (i=1; i<d_dst->n_entry; i++)
        {
                for (j=0; j<d_src->n_entry && d_dst->offsets[i*d_dst->n_tuple]!=d_src->offsets[j*d_src->n_tuple]; j++);
                emit_assert(t, j<d_src->n_entry);
                emit_push(t, ",", itoa(d_src->offsets[j]));
        }
        emit_push(t, "};");
}

char *  t_offset_i = token_new();       // 每个维度对指标的循环变量

emit_push(t, int_type, " ", t_offset_i, ";");
emit_push(t, "for(", t_offset_i, "=0;", t_offset_i, "<", itoa(d_dst->n_entry), ";", t_offset_i, "++){");

char *  t_tmp = token_new();
emit_push(t, int_type, " ", t_tmp, "=", t_dst_i, "+", t_dst_offset_p, "[", t_offset_i, "];");
t_dst_i = t_tmp;
```

新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
MetaTensor:
• 基于元编程的代码生成器
• 统一张量数据描述、“块”
• 插件、编译目标

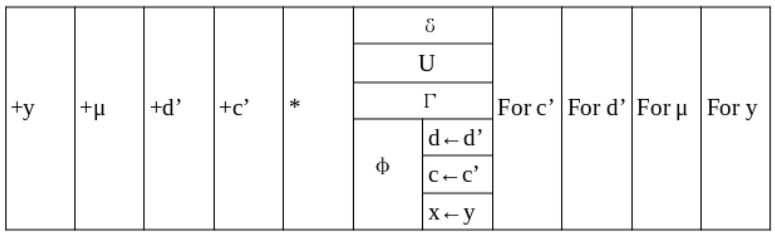
新软件框架的结构



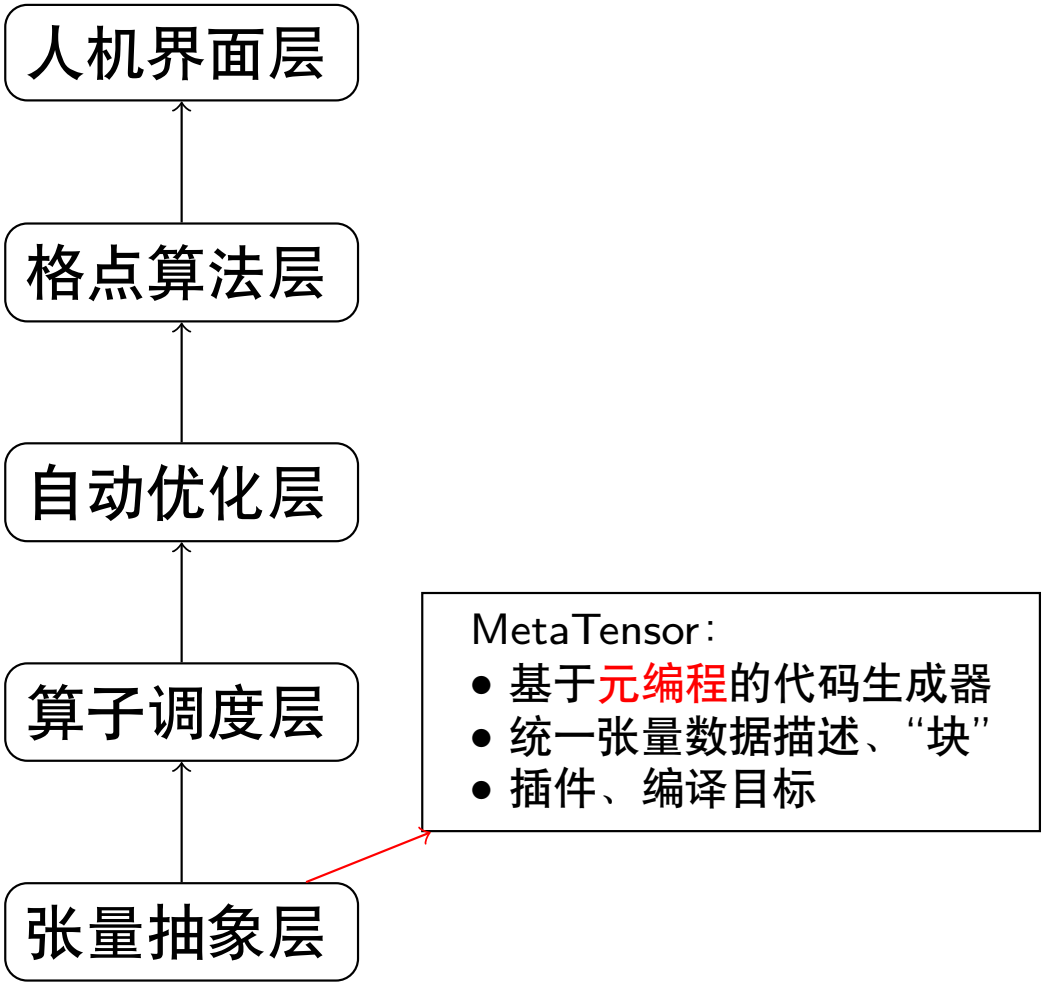
$$\not{D} = \sum_\mu \left[(1^{(4)} - \gamma_\mu) U_\mu(\vec{x}) \delta_{\vec{x}+\vec{\mu},\vec{y}} + (1^{(4)} + \gamma_\mu) U^\dagger_\mu(\vec{x}-\vec{\mu}) \delta_{\vec{x}-\vec{\mu},\vec{y}} \right]$$

$$\sum_{\vec{y}} \sum_\mu \sum_{d'} \sum_{c'} \Gamma^{d,d'}_\mu \, U^{c,c'}_{\mu,\vec{x}} \delta_{\vec{x}+\vec{\mu},\vec{y}} \phi^{d',c'}_{\vec{y}}$$

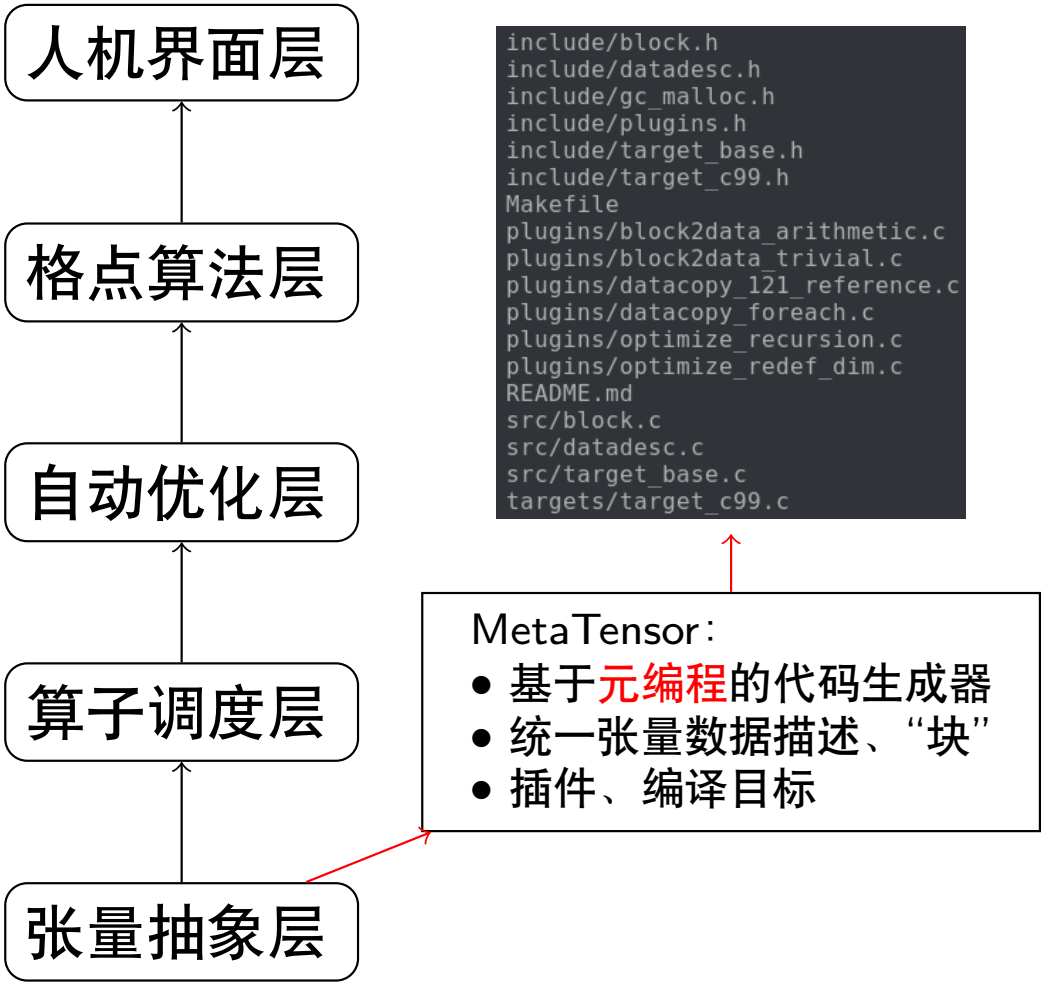
MetaTensor：
- 基于元编程的代码生成器
- 统一张量数据描述、“块”
- 插件、编译目标



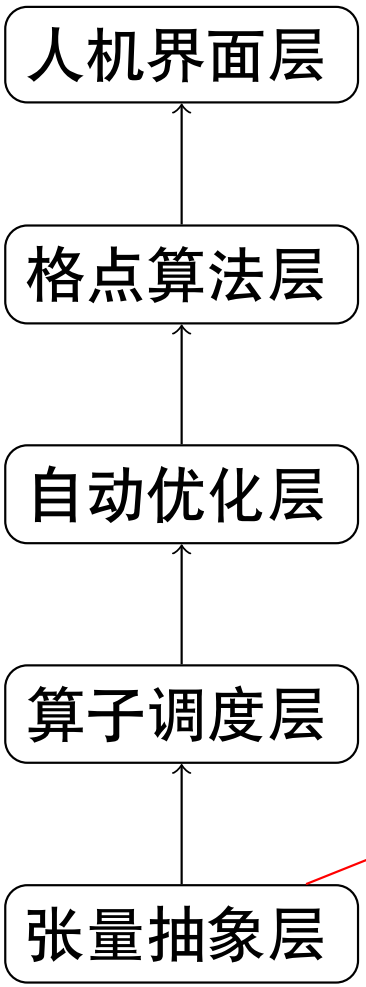
新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
MetaTensor:
• 基于元编程的代码生成器
• 统一张量数据描述、“块”
• 插件、编译目标

新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
include/block.h
include/datadesc.h
include/gc_malloc.h
include/plugins.h
include/target_base.h
include/target_c99.h
Makefile
plugins/block2data_arithmetic.c
plugins/block2data_trivial.c
plugins/datacopy_121_reference.c
plugins/datacopy_foreach.c
plugins/optimize_recursion.c
plugins/optimize_redef_dim.c
README.md
src/block.c
src/datadesc.c
src/target_base.c
targets/target_c99.c
MetaTensor:
基于元编程的代码生成器
统一张量数据描述、“块”
插件、编译目标

新软件框架的结构




```
include/block.h
include/datadesc.h
include/gc_malloc.h
include/plugins.h
include/target_base.h
include/target_c99.h
Makefile
plugins/block2data_arithmetic.c
plugins/block2data_trivial.c
plugins/datacopy_121_reference.c
plugins/datacopy_foreach.c
plugins/optimize_recursion.c
plugins/optimize_redef_dim.c
README.md
src/block.c
src/datadesc.c
src/target_base.c
targets/target_c99.c
```

所有插口：

1、datacopy ：

- （本地/远程/集合/同步）数据拷贝
- 数据布局转置/归并
- 数据类型/大小端转换
- 数据归约
- 数据广播

2、block2data：

- 所有张量计算
- 迭代计算

3、optimize：

- 程序等价推演
- 编译期代码优化

4、compile：

- 孤立代码的编译入口

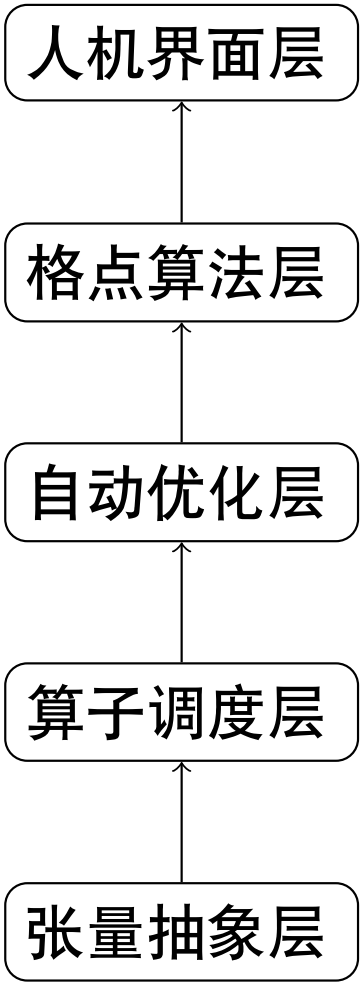
MetaTensor：

- 基于元编程的代码生成器
- 统一张量数据描述、“块”
- 插件、编译目标

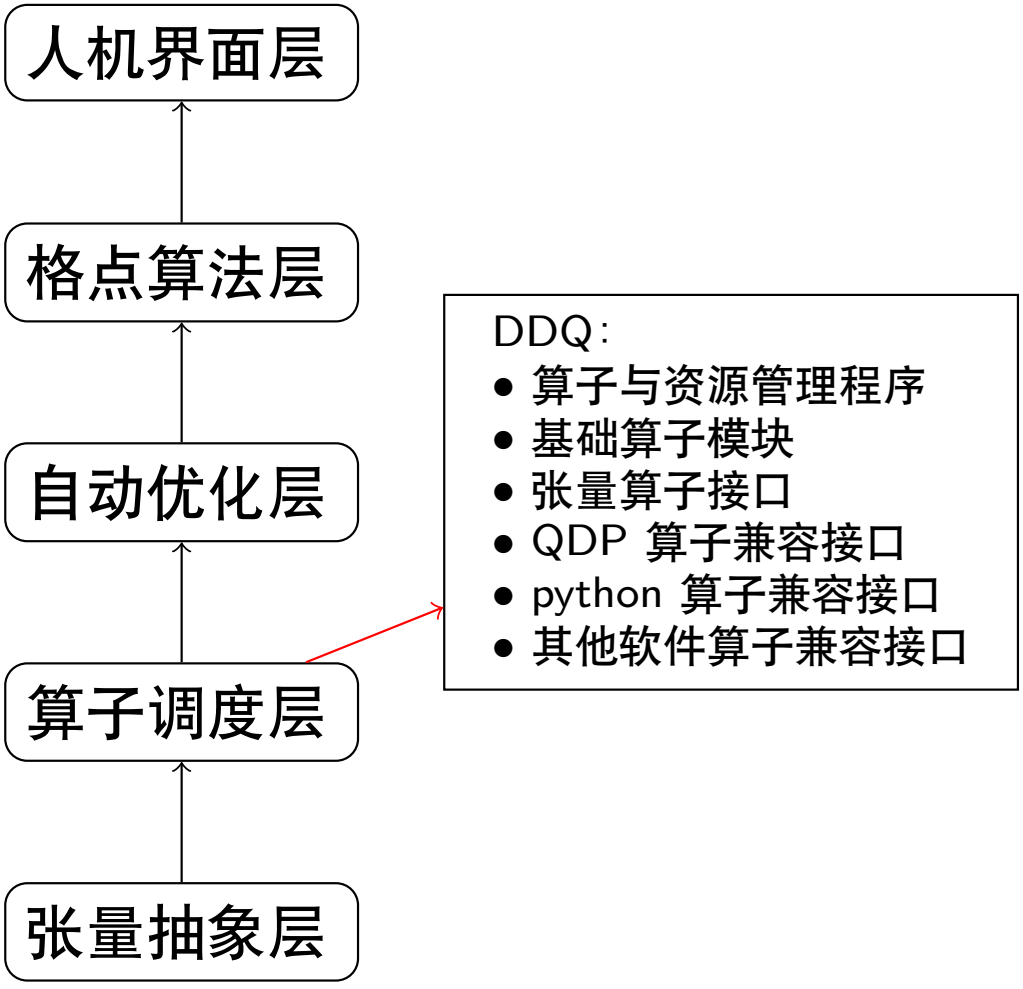
可选的编译目标：

- C99 (IO、mpi、omp、pthread、socket)
- C99 (GPU、天河、申威)
- LLVM
- 运行时翻译执行
- ……

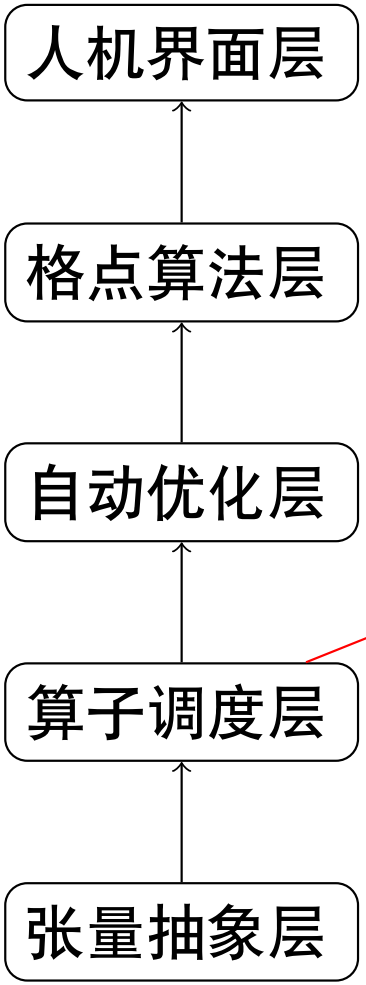
新软件框架的结构



新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
DDQ:
• 算子与资源管理程序
• 基础算子模块
• 张量算子接口
• QDP 算子兼容接口
• python 算子兼容接口
• 其他软件算子兼容接口

新软件框架的结构



DDQ:
- 算子与资源管理程序
- 基础算子模块
- 张量算子接口
- QDP 算子兼容接口
- python 算子兼容接口
- 其他软件算子兼容接口

OSP:
- 用纯 C99 语法写的伪多线程库
- 实现了多任务并发
- 普通 CPU 上测试通过
- 申威从核上测试通过
- 天河三 DSP 上测试通过

新软件框架的结构

人机界面层

格点算

自动化

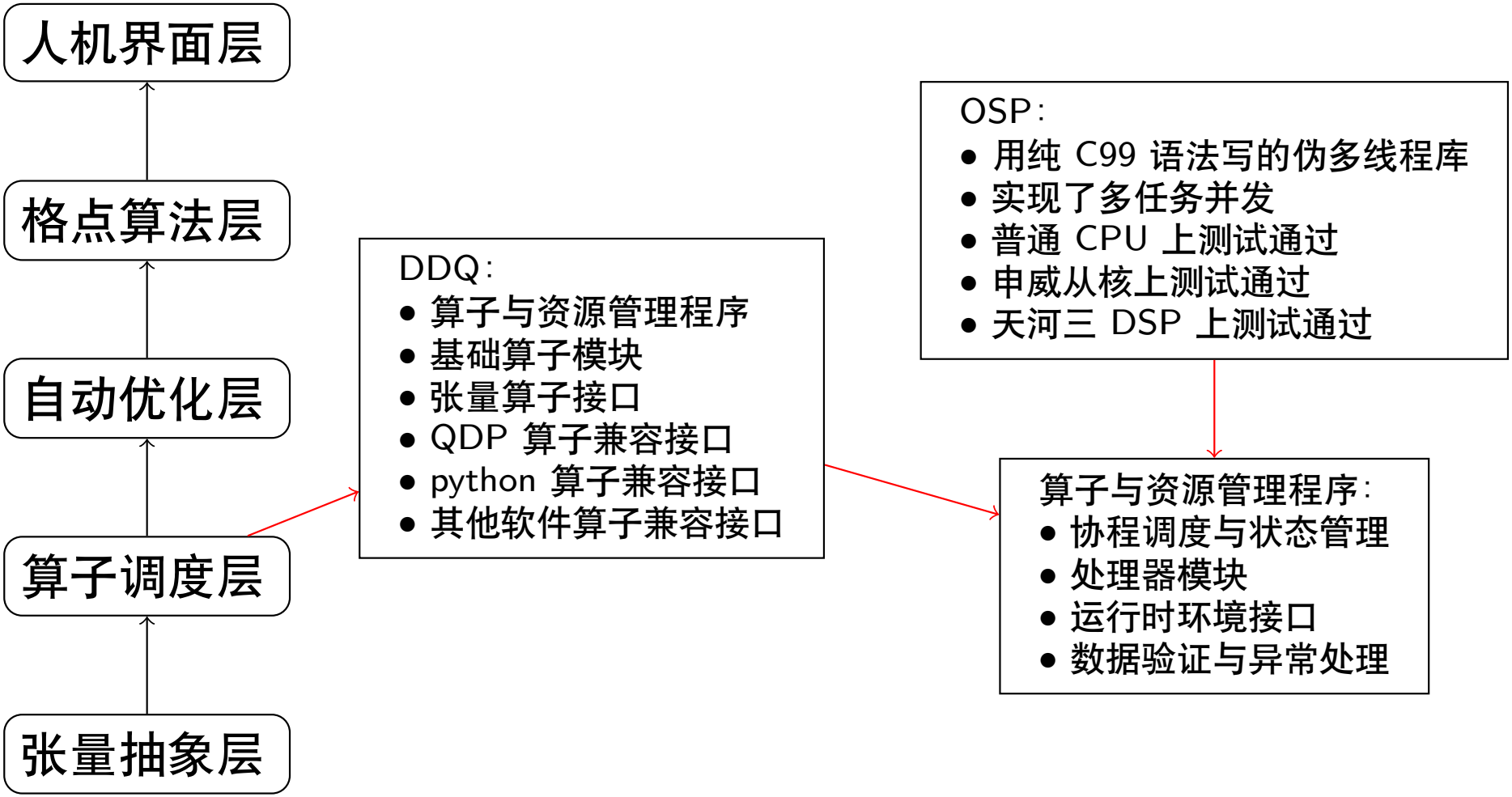
算子计

张量抽象层

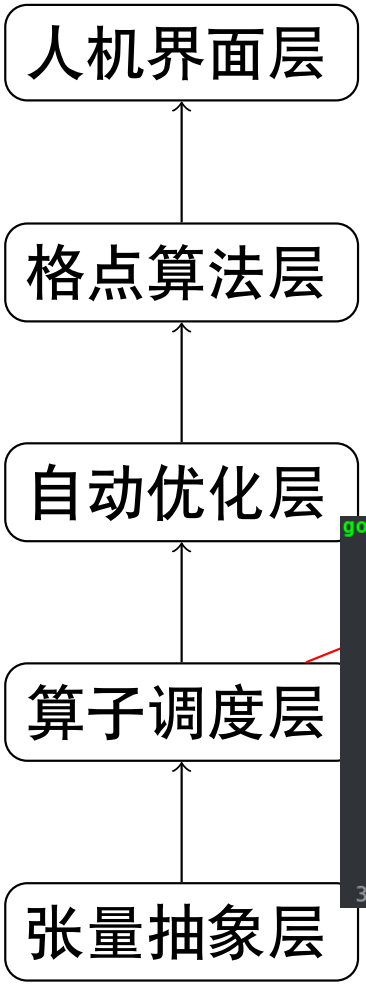
程库

```c
#include        <stdio.h>
#include        <time.h>
#include        <stdlib.h>

#include        "osp.h"


int     main()
{
        time_t  t0 = time(NULL);

        osp_prepare(10, 2);     //开启10个线程，同时最多并发2个

        time_t  osp_private(t);

        osp_start();            //启动线程

        osp_v(t) = rand()%10+time(NULL)-t0;

        printf("Thread #%d started at %ld and should stop at %ld.\n", osp_id, time(NULL)-t0, osp_v(t)+1);

        osp_waitfor(time(NULL)-t0 > osp_v(t));

        printf("Thread #%d have stopped at %ld.\n", osp_id, time(NULL)-t0);

        osp_finish();           //等待所有线程结束
}
```

新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
DDQ:
• 算子与资源管理程序
• 基础算子模块
• 张量算子接口
• QDP 算子兼容接口
• python 算子兼容接口
• 其他软件算子兼容接口
OSP:
• 用纯 C99 语法写的伪多线程库
• 实现了多任务并发
• 普通 CPU 上测试通过
• 申威从核上测试通过
• 天河三 DSP 上测试通过
算子与资源管理程序:
• 协程调度与状态管理
• 处理器模块
• 运行时环境接口
• 数据验证与异常处理

新软件框架的结构



DDQ:

- 算子与资源管理程序
- 基础算子模块
- 张量算子接口

```
gongming@gongmingmobile ~/ddq $ wc processor_*/*
  31   57  741 processor_cuda/processor_cuda.c
  36   49  625 processor_cuda/processor_cuda.h
  20   47  314 processor_cuda/processor_cuda_private.c
  11   12  113 processor_direct/processor_direct.c
  18   22  251 processor_direct/processor_direct.h
   0    0    0 processor_direct/processor_direct_private.c
   8   10  103 processor_fork/Makefile
  81  204 2786 processor_fork/processor_fork.c
  22   33  349 processor_fork/processor_fork.h
   0    0    0 processor_fork/processor_fork_private.c
   8   10  109 processor_pthread/Makefile
  49   94 1606 processor_pthread/processor_pthread.c
  23   32  340 processor_pthread/processor_pthread.h
  13   26  247 processor_pthread/processor_pthread_private.c
 320  596 7584 总用量
```

算子与资源管理程序:

- 协程调度与状态管理
- 处理器模块
- 运行时环境接口
- 数据验证与异常处理

新软件框架的结构

人机界面层

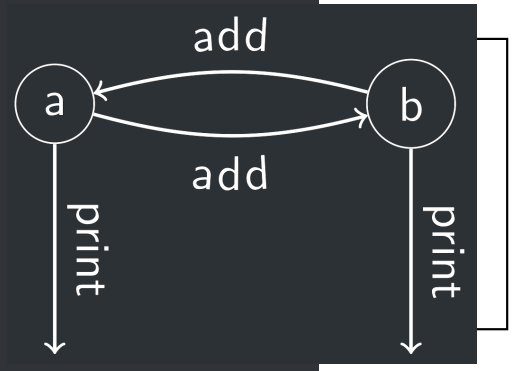
格点算法层

自动优化层

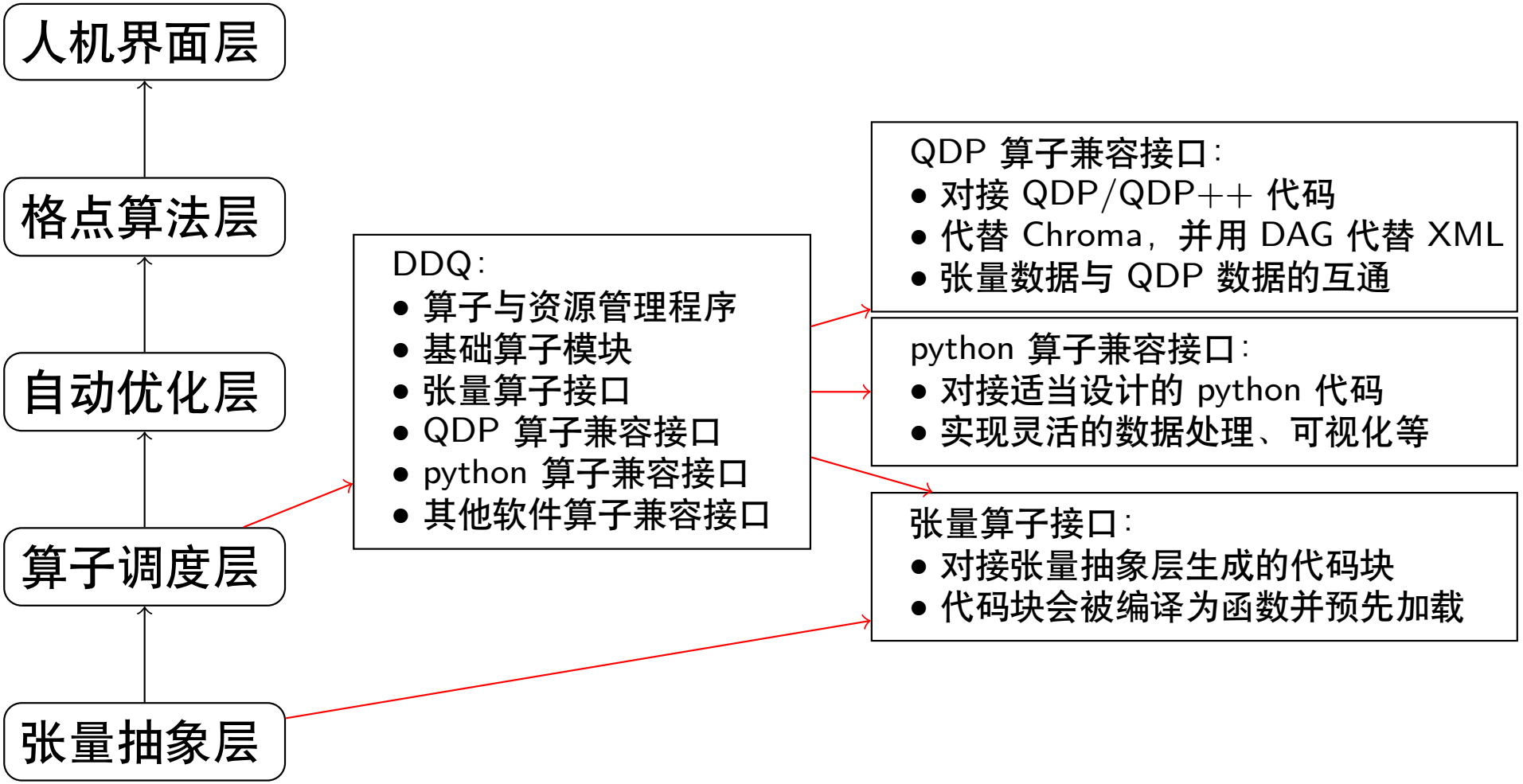
算子调度层

张量抽象层

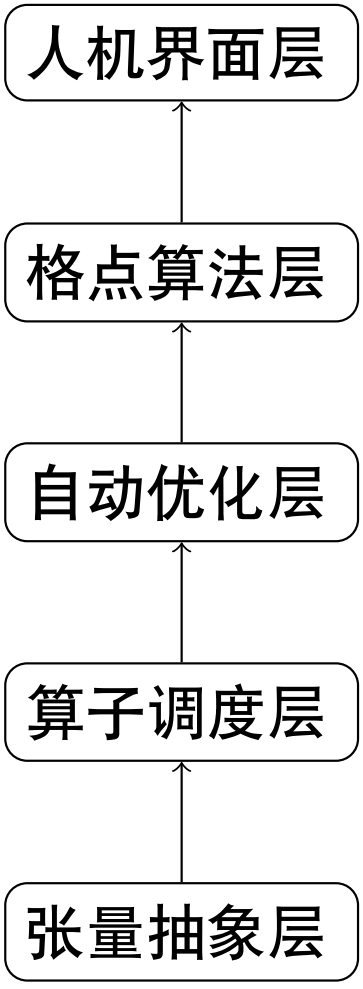
```
task_ret        f_print(void **inputs, void **outputs);
task_ret        f_add(void **inputs, void **outputs);

int     main()
{
        obj     f_a, f_p, a, b;
        ddq_op  ab, ba, pa, pb;

        unsigned long   a0, b0;
        a0 = b0 = 1;

        f_a = obj_import(f_add, NULL, obj_prop_ready);
        f_p = obj_import(f_print, NULL, obj_prop_ready);

        a = obj_import(&a0, NULL, obj_prop_consumable | obj_prop_ready);
        b = obj_import(&b0, NULL, obj_prop_consumable);

        ab = ddq_spawn(processor_pthread, 1, 1);
        ab->f = f_a;
        ab->inputs[0] = a;
        ab->outputs[0] = b;

        ba = ddq_spawn(processor_pthread, 1, 1);
        ba->f = f_a;
        ba->inputs[0] = b;
        ba->outputs[0] = a;

        pa = ddq_spawn(processor_pthread, 1, 0);
        pa->f = f_p;
        pa->inputs[0] = a;

        pb = ddq_spawn(processor_pthread, 1, 0);
        pb->f = f_p;
        pb->inputs[0] = b;

        ddq_loop();

        return 0;
}
```



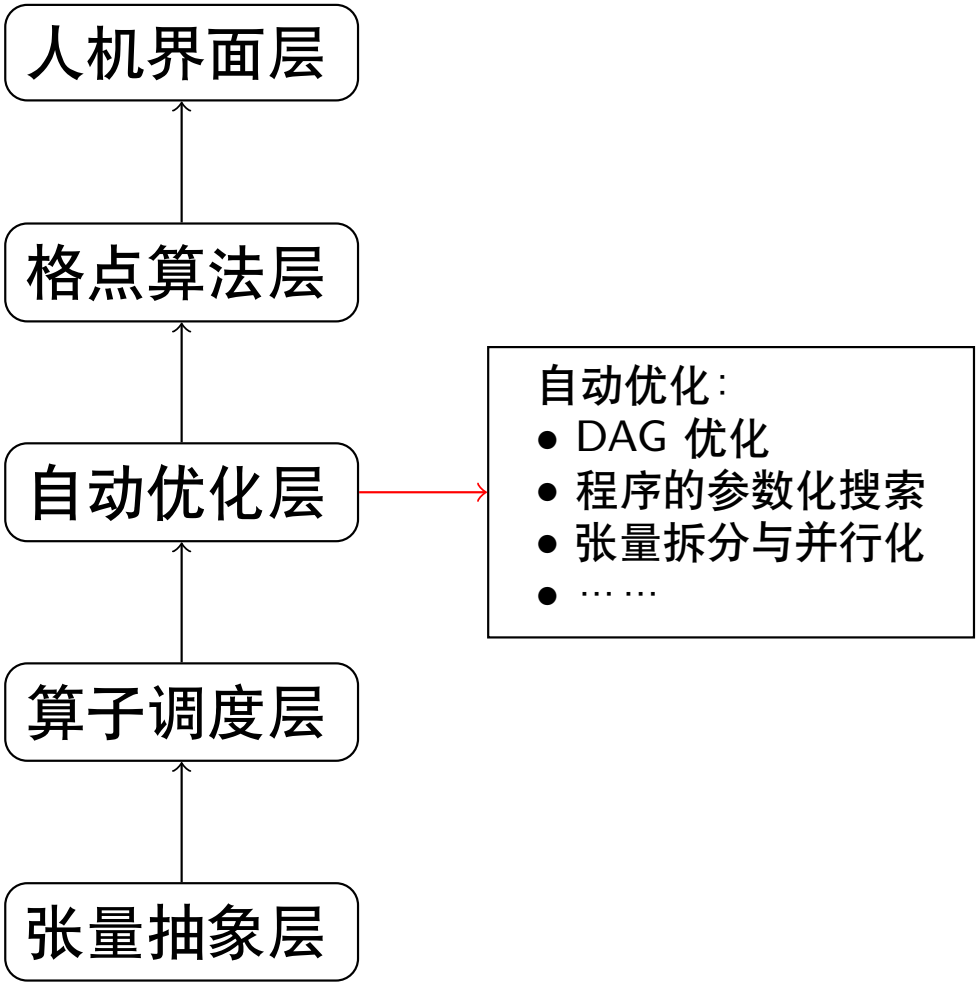
新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
DDQ:
• 算子与资源管理程序
• 基础算子模块
• 张量算子接口
• QDP 算子兼容接口
• python 算子兼容接口
• 其他软件算子兼容接口
QDP 算子兼容接口:
• 对接 QDP/QDP++ 代码
• 代替 Chroma，并用 DAG 代替 XML
• 张量数据与 QDP 数据的互通
python 算子兼容接口:
• 对接适当设计的 python 代码
• 实现灵活的数据处理、可视化等
张量算子接口:
• 对接张量抽象层生成的代码块
• 代码块会被编译为函数并预先加载

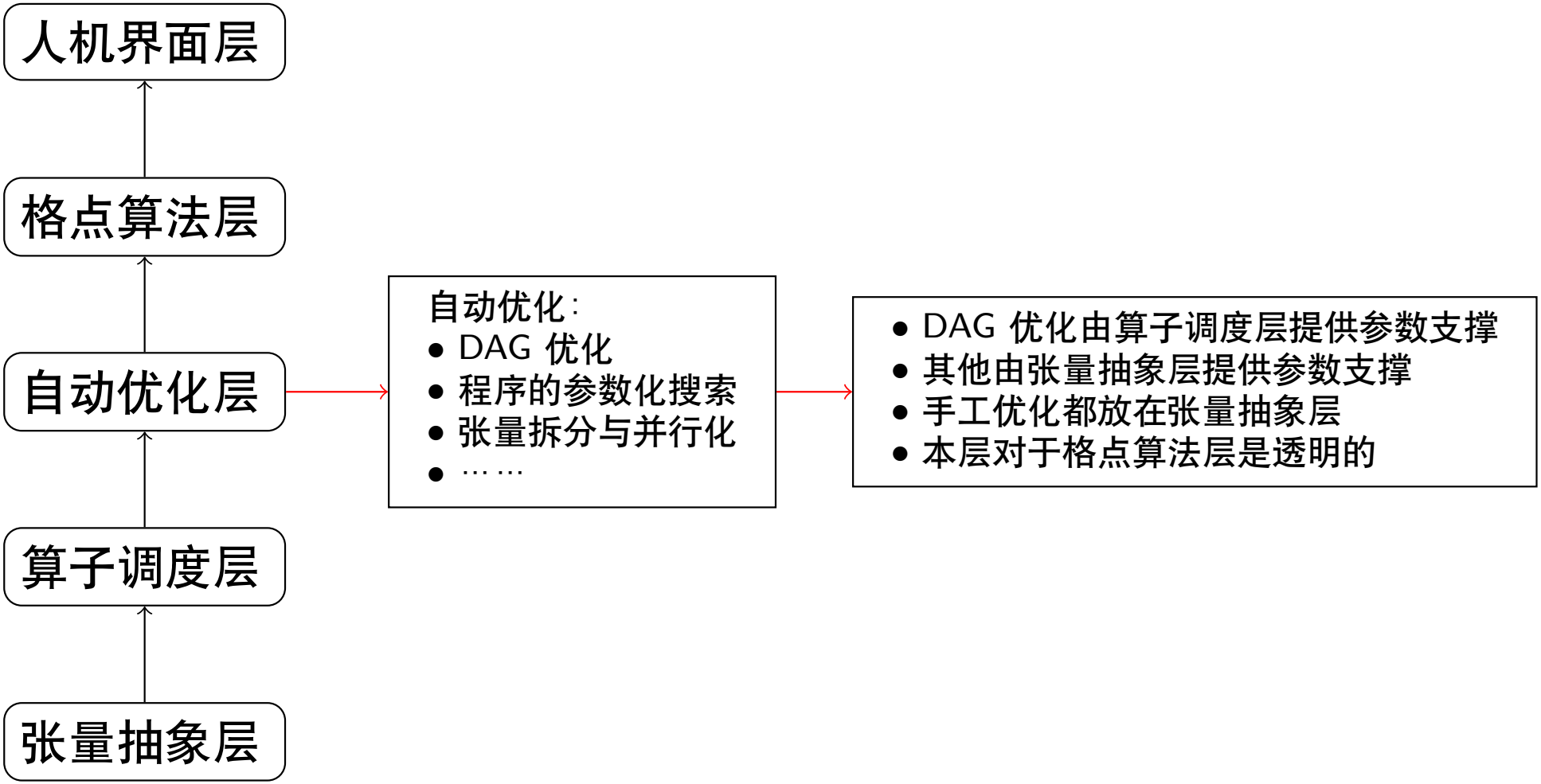
新软件框架的结构



新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
自动优化：
DAG 优化
程序的参数化搜索
张量拆分与并行化
……

新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
自动优化：
• DAG 优化
• 程序的参数化搜索
• 张量拆分与并行化
• ……
• DAG 优化由算子调度层提供参数支撑
• 其他由张量抽象层提供参数支撑
• 手工优化都放在张量抽象层
• 本层对于格点算法层是透明的

新软件框架的结构

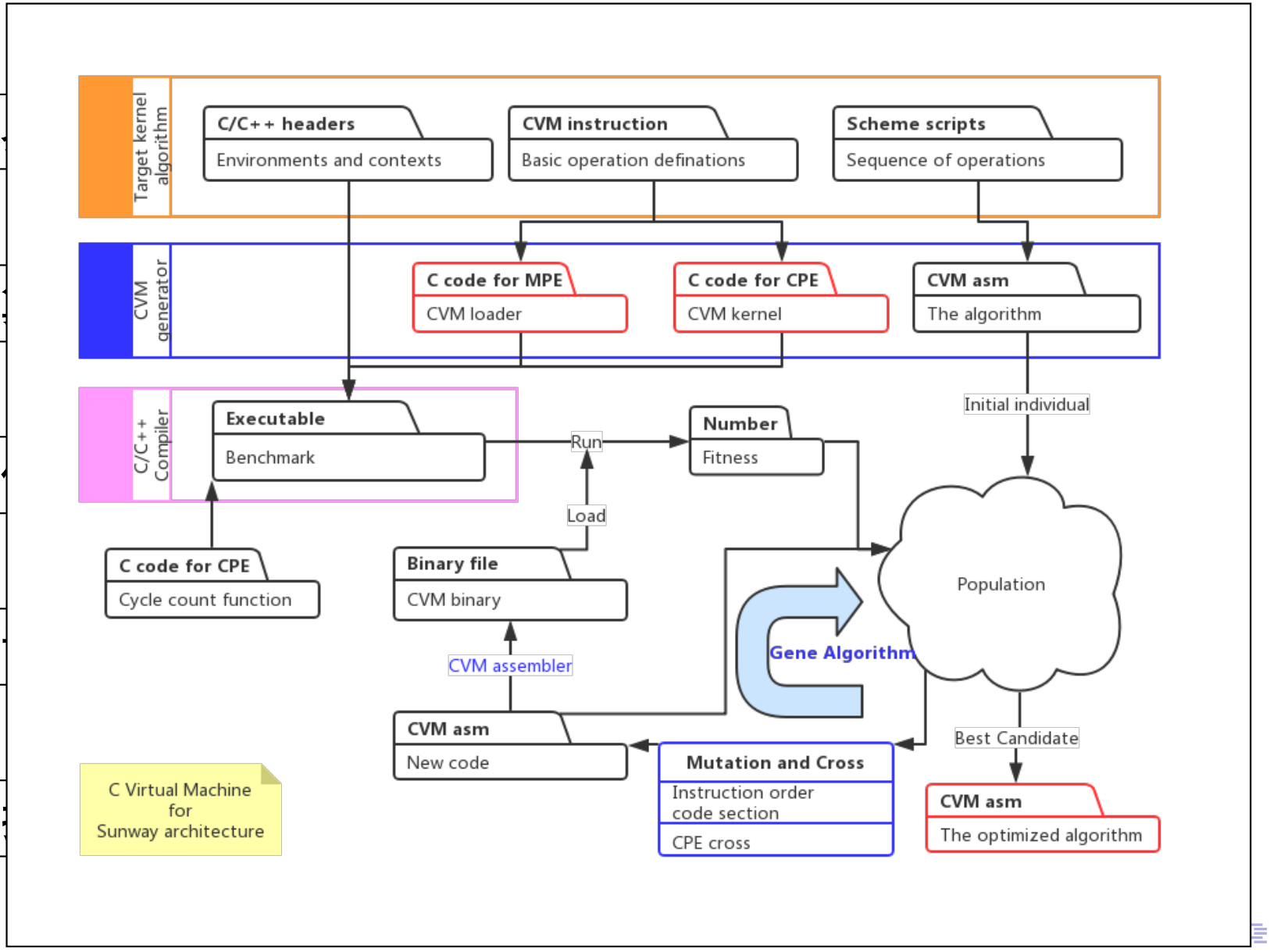
人机

格点

自动

算子

张量



新软件框架的结构

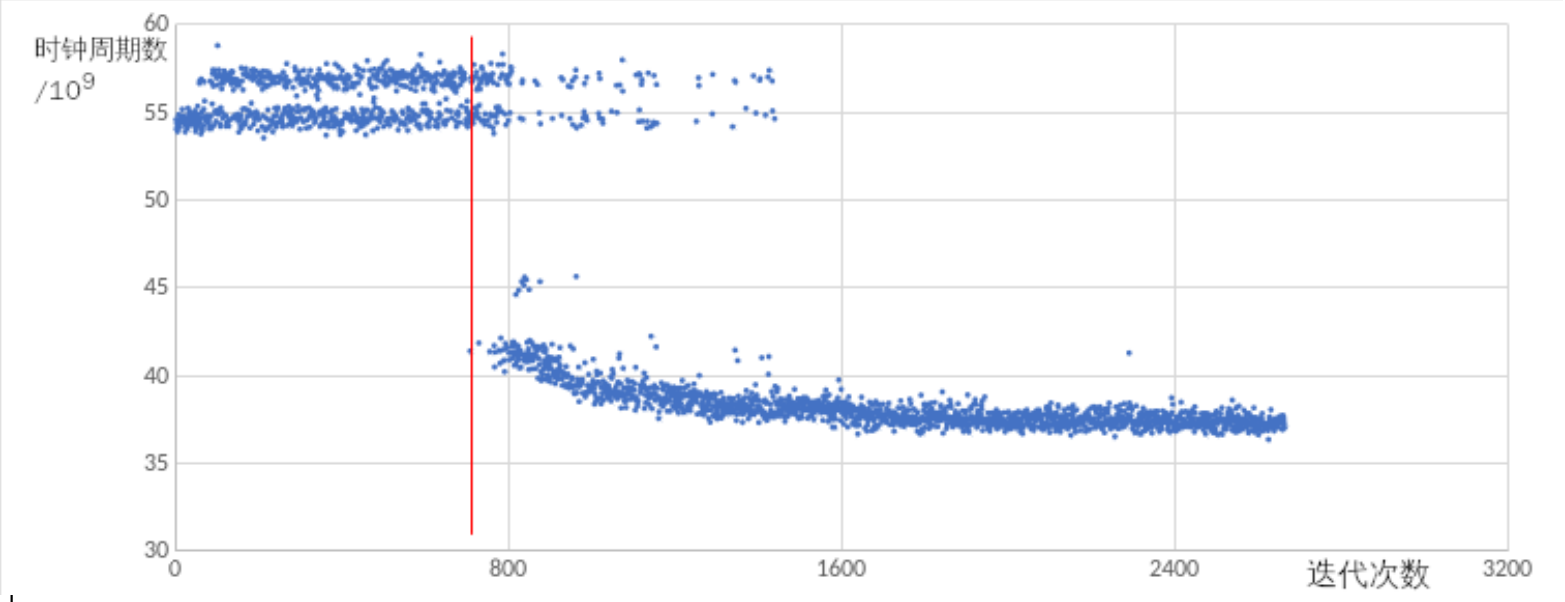
人机界面层

格点算

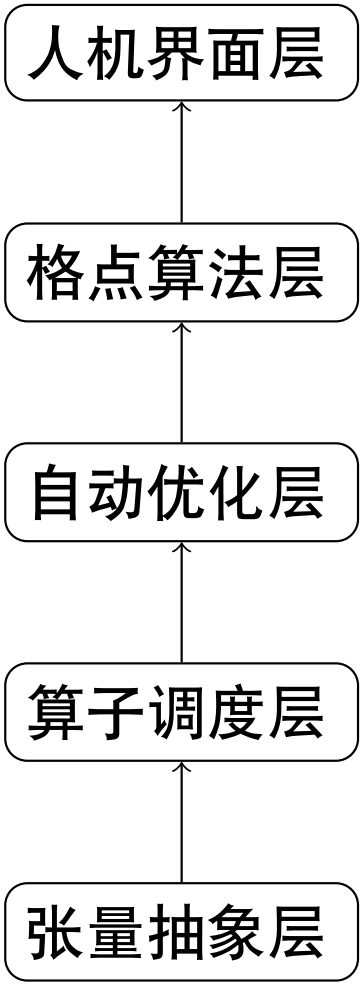
自动化

算子计

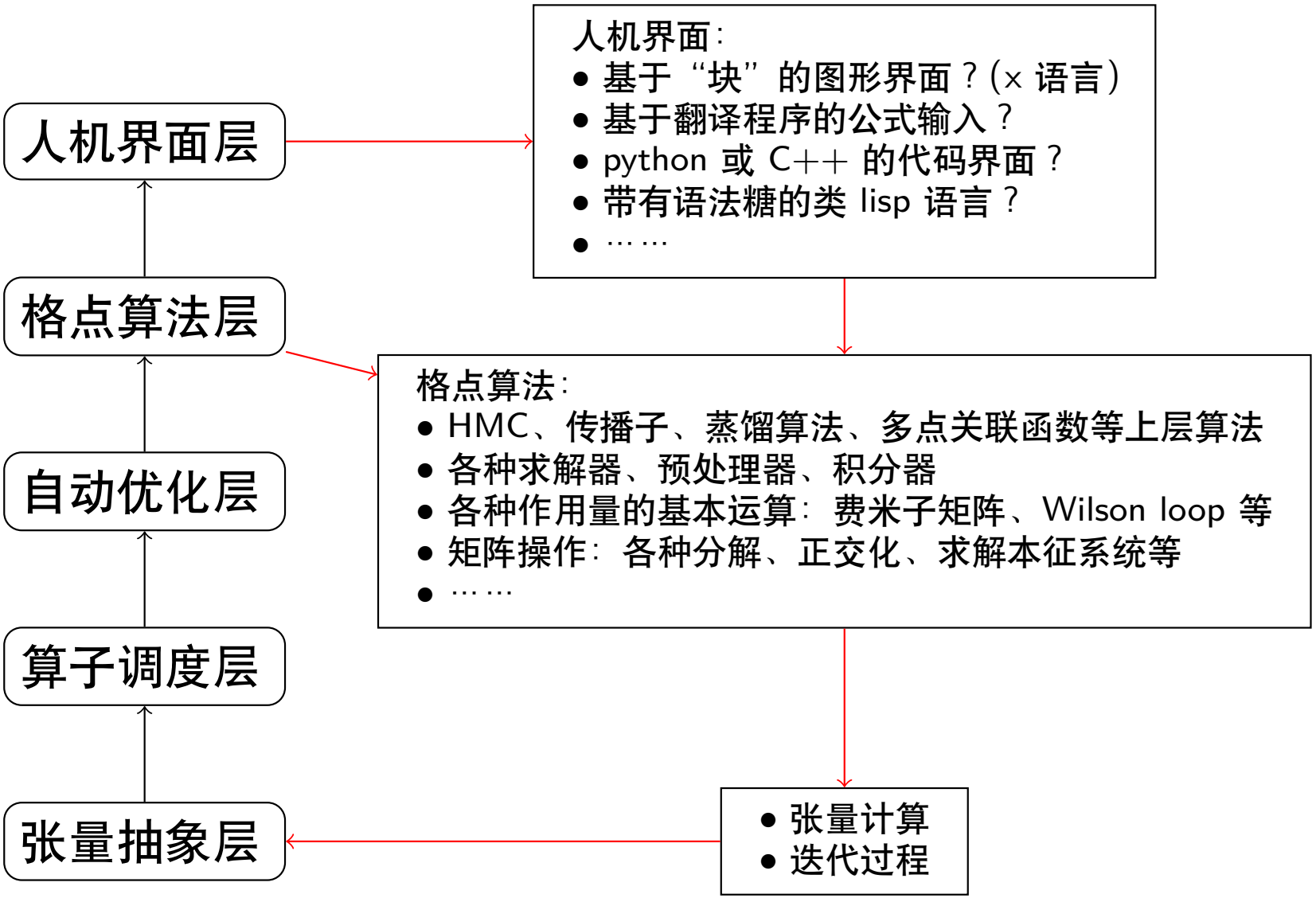
张量抽象层



新软件框架的结构



新软件框架的结构


人机界面层
格点算法层
自动优化层
算子调度层
张量抽象层
人机界面：
• 基于“块”的图形界面？(× 语言)
• 基于翻译程序的公式输入？
• python 或 C++ 的代码界面？
• 带有语法糖的类 lisp 语言？
• ……
格点算法：
• HMC、传播子、蒸馏算法、多点关联函数等上层算法
• 各种求解器、预处理器、积分器
• 各种作用量的基本运算：费米子矩阵、Wilson loop 等
• 矩阵操作：各种分解、正交化、求解本征系统等
• ……
• 张量计算
• 迭代过程

提纲

目前进度与阶段计划

目前进度

- 整体框架的设计有了基本的构想，做了大量技术验证等预研工作
- 张量抽象层设计完毕，进入实做代码阶段，初始版本完成度 10%
- 算子调度层设计完毕，进入实做代码阶段，初始版本完成度 30%
- 自动优化层先留出空间，作为中期目标
- 格点算法层可以先用开源的 QDP 代码顶上，以后慢慢重构
- 人机界面层设计了几种方案，具体用哪个方案取决于用户反馈。

目前进度与阶段计划

第一期计划（1.5 年）

- 完成张量抽象层和算子调度层的基本代码
- 用元编程方案，测试性地实现一些计算程序，比如求解传播子等
- 完成 QDP 兼容接口，把一些开源代码接入框架，完成实际的计算工作
- 人机界面方案定型

第二期计划（+2 年）

- 用元编程方案实现更多格点算法，完成实际的计算工作
- 扩展到多个软硬件平台
- 探索自动优化算法

第三期计划（+2 年）

- 完整实现大部分格点常用算法
- 实现优美的人机界面
- 实现若干有效的自动优化方案
- 扩展到几乎所有可用的软硬件平台

谢谢！

格点计算新软件框架构想

宫明

中科院高能所
CLQCD 合作组

2022.10.10