

HPC SIG生态建设展望

方春林 HPC SIG Maintainer



算力爆炸时代：HPC的技术探索与生态构建迫在眉睫



科技强国战略驱动国内HPC建设

- 超算中心承载尖端科研创新，目前已进入高峰期
- 行业科技创新和数字化，加速对高性能计算的建设需求

21年国内HPC市场规模达到40亿\$

- 2021年6月发布的TOP 500排名中，中国上榜的超级计算机数量高达188台，稳居世界第一
- 应用领域以制造、生命科学、EDA和气象为主



HPC应用快速迁移、调优需求难以满足

痛点1：迁移难

不同架构的差别巨大，而且不同的语言不同的软件形式会有不同的迁移方案

痛点2：部署难

集群需要具备X86/ARM等异构算力的统一调度，不同应用的依赖，编译器、通信库都存在差异

痛点3：调优难

HPC应用属于交叉学科领域，除了计算机之外还需要了解不同学科的基础知识，而且集群上面的性能采集和分析对软件和硬件的知识要求很高

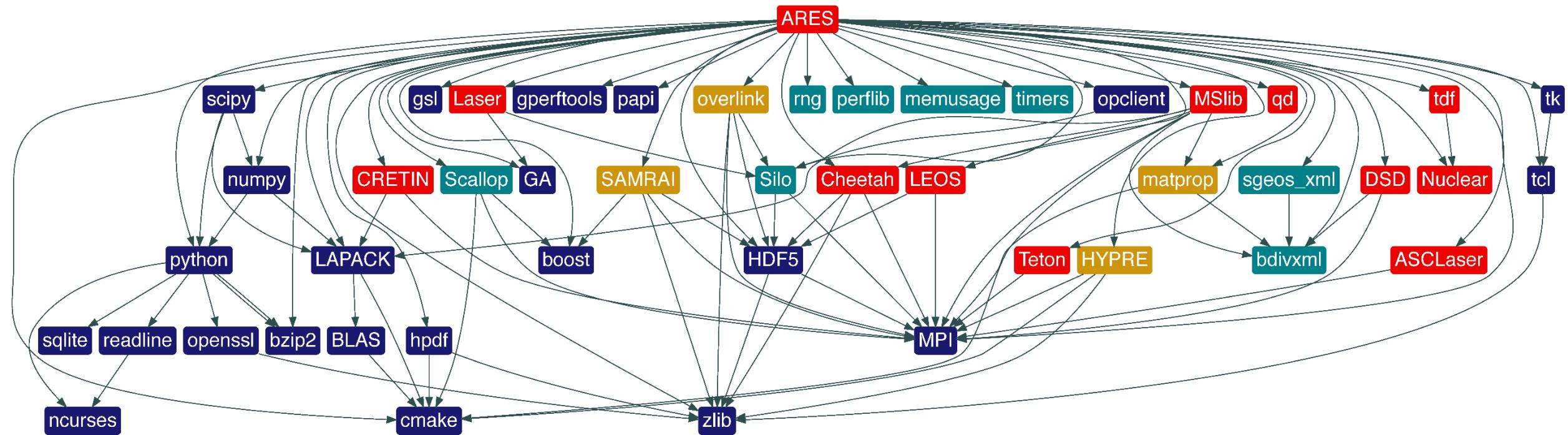
建立气象、分子动力学、制造、基因、生物等领域的高校与企业、研究员与软件开发工程师的交流圈

基于OpenEuler OS，打造HPC开源软件仓和迁移调优平台，降低HPC应用使用者和开发者的门槛

HPC开源软件迁移、HPC算法优化、软硬协同等技术分享和资源整合，加速HPC应用开发效率

HPC应用痛点：海量依赖带来部署的复杂性

高性能计算应用场景众多，并且变得越来越复杂，每种应用需要不同配套软件环境和大量的系统依赖，HPC开发人员花费大量时间部署软件，部署复杂度= **M个编译器*N个编译器版本*O个依赖*P个依赖版本*Q个HPC应用版本**



Intel编译器

ARM编译器

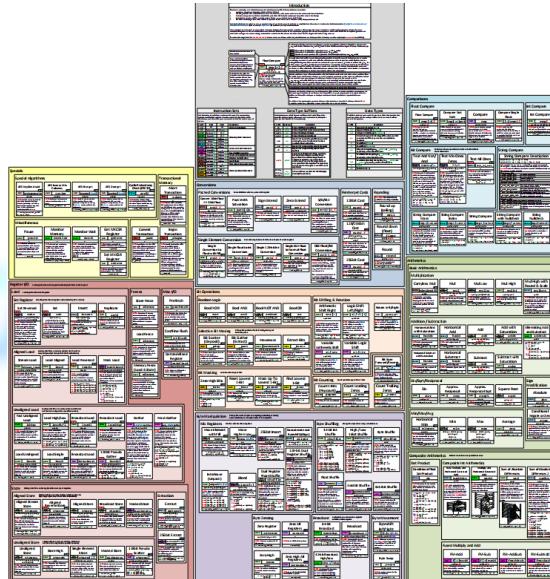
GCC编译器

Nvidia编译器

CLANG编译器

HPC应用痛点：多架构体系带来迁移的复杂性

高性能计算应用普遍使用了大量硬件相关的并行技术和通信技术，导致不同架构之间迁移的复杂性陡增



Intel向量化指令集：
<https://software.intel.com/sites/landingpage/IntrinsicsGuide/>

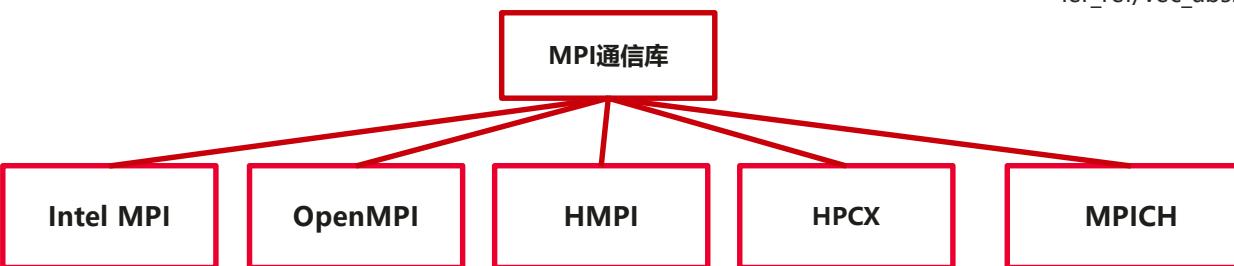
NEON	EXPLANATION	PSEUDOCODE
vdupq_n_f32(a)	New NEON value	a
vsubq_f32(a, b)	Subtract	a - b
vaddq_f32(a, b)	Add	a + b
vmulq_f32(a, b)	Multiply	a * b
vmlaq_f32(a, b, c)	Multiply and add	a + (b * c)
vmlsq_f32(a, b, c)	Multiply and subtract	a - (b * c)
vrsqrteq_f32(a)	Reciprocal square root	1 / sqrt(a)
vcgtq_f32(a, b)	Compare greater than	a > b ? 1 : 0
vcltq_f32(a, b)	Compare less than	a < b ? 1 : 0
vbslq_f32(mask, a, b)	Select by mask	mask != 0 ? a : b
vminq_f32(a, b)	Get minimum	a < b ? a : b
vmaxq_f32(a, b)	Get maximum	a > b ? a : b

ARM向量化指令集：
<https://developer.arm.com/architectures/instruction-sets/simd-isas/neon/intrinsics>

- Vector built-in functions

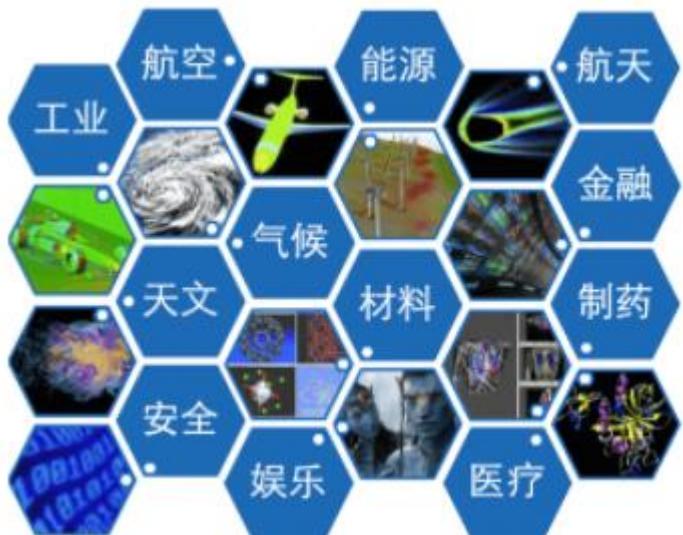
vec_abs

vec_add
vec_all_eq
vec_all_ge
vec_all_gt
vec_all_le
vec_all_lt
vec_all_nan
vec_all_ne
vec_all_nge
vec_all_ngt
vec_all_nle
vec_all_nlt



HPC应用痛点：性能为王

高性能计算应用属于交叉学科，专业性强，算法晦涩难懂，架构设计复杂多样，导致调优难度普遍较高，而业界普遍都在追求非常极致的性能、性价比与能耗比，仅是提高很少量的百分点都能为企业带来巨大的经济效益



HPC应用的专业性

Sparse Linear
Algebra

Spectral
Methods

N-Body
Methods

Monte Carlo → MapReduce

Dense Linear
Algebra



Unstructured
Grids

Structured
Grids

科学计算的7种算法模型

Stevens, R., 2000. Future directions in computer and systems architecture for scientific computing. In Institute for Theoretical Atomic and Molecular Physics Workshop.

HPC SIG生态圈

成员



社区

community / sig / sig-HPC

木得感情的openEuler机器人 update yaml files 478144e 1个月前

update yaml files
update yaml files
update sig/sig-HPC/OWNERS.
update HPC SIG info
update HPC SIG info

README_md

README_cn.md

HPC SIG

背景

openEuler HPC SIG的成员已在HPC气象、制造、分子动力学等领域Top应用进行深入优化，在性能、精度方面取得显著提升，同时开展了HPC众智活动，许多高校师生参与到HPC应用向openEuler的迁移工作中。

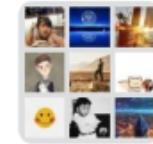
目标

通过建立openEuler HPC SIG小组，汇聚HPC开发者、应用者。让Top应用优化快速应用到科研领域，吸引高校师生参与，共同构建生态，吸收不同领域学者的思路、思想，把HPC SIG建设成HPC领域的根据地，共同繁荣HPC+openEuler生态。

SIG 组职责

- 在 openEuler 社区中添加对 HPC 开源软件的支持
- 负责 HPC 相关软件包的规划、迁移和调优，并输出相应的迁移调优文档
- 及时响应用户反馈，解决相关问题

微信交流群



openEuler sig-HPC



鲲鹏众智计划：覆盖100+HPC应用，优先满足前80%高算力需求应用

通过任务包揭榜方式，邀请高校师生、科研机构、企业伙伴等广大开发者积极贡献智慧，合力打造和提升鲲鹏基础软件能力，实现鲲鹏生态共建共享。

报名进行中

HPC软件AlphaFold2、Manta等5款软件迁移openEuler平台

项目编码 H00122
语言要求 C/C++、Fortran
发布时间 2022/03/04

立即申请 **查看详情 →** 参与 1 团队 **¥ 300,000-¥ 500,000**

报名进行中

HPC软件amber、EIGENSOFT、HMMER等6款软件迁移openEuler平台

项目编码 H00123
语言要求 C/C++、Fortran
发布时间 2022/03/04

立即申请 **查看详情 →** 参与 4 团队 **¥ 450,000-¥ 650,000**

报名进行中

HPC软件bamtools、BUSCO等5款软件迁移openEuler平台

项目编码 H00124
语言要求 C/C++、Fortran
发布时间 2022/03/04

立即申请 **查看详情 →** 参与 3 团队 **¥ 300,000-¥ 500,000**

报名地址：https://www.hikunpeng.com/zh/ecosystem/ecology_remit?id=7
下一期预计4月底揭晓

鲲鹏HPC开源仓：基于鲲鹏软件栈的60+优化应用二进制，方便快速部署

HPC开源软件库

鲲鹏CPU生态 鲲鹏GPU生态

请输入关键词

软件类型 全部 生命科学 制造仿真 气象海洋 基础科研

展示方式 卡片 列表

wrf 中尺度天气预报模型 气象海洋

cesm 地球系统模型 气象海洋

cp2k 是一个量子化学和固态物理软件包，可以对固态，液态，分子，周期性，材料，... 基础科研

gromacs 分子动力学模拟和能量最小化的计算引擎。 基础科研

lammps 分子动力学模拟的开源程序包 基础科研

namd 分子动力学模拟的开源程序包 基础科研

nwchem 可扩展的计算化学工具 基础科研

openfoam 开源的场运算和处理软件 制造仿真

<https://ic-openlabs.huawei.com/client/#/appindex/appcontent>

开源实习计划：培养开源黑土地，激发学生参与HPC的积极性

HPC 生态建设的最大痛点是人才，从高校到产业界，HPC 作为尖端垂直领域，始终缺少大规模、高质量的人才梯队

openEuler开源实习

领取任务



<https://www.openeuler.org/zh/internship/>

- HPC软件Elmer迁移至openEuler平台 intern intern-HPC
#I4ZWTK kydkyky 1
- HPC软件blast迁移至openEuler平台 intern intern-HPC
#I4ZW8D kydkyky 1
- HPC软件QMCPACK迁移至openEuler平台 intern
#I4ZWLP kydkyky 1
- HPC软件Bowtie2迁移至openEuler平台 intern
#I4ZW6M kydkyky 1
- HPC软件mfem迁移至openEuler平台 intern
#I4ZWHZ kydkyky 1
- HPC软件ROMS迁移至openEuler平台 intern intern-HPC
#I4ZW2E kydkyky 1
- HPC软件Trinity迁移至openEuler平台 intern intern-HPC
#I4ZWBC kydkyky 1
- HPC软件OpenLB迁移至openEuler平台 intern intern-HPC
#I4ZVXJ kydkyky 1
- HPC软件scanorama迁移至openEuler平台 intern intern-HPC
#I4ZWAT kydkyky 1
- HPC软件Wannier90迁移至openEuler平台 intern intern-HPC
#I4Z41D kydkyky 1

<https://gitee.com/openeuler/hpcrunner/issues>

业界HPC容器洞察：面向高度定制化HPC应用

Docker

- 非root用户使用授权问题
- 调度器的资源限制无法施加到容器
- HPC应用性能比较差
- 不完全支持IB网络

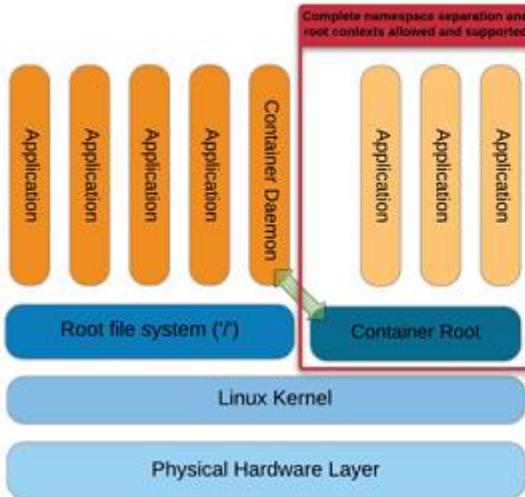
Nvidia-docker

- 可以使用GPU的docker

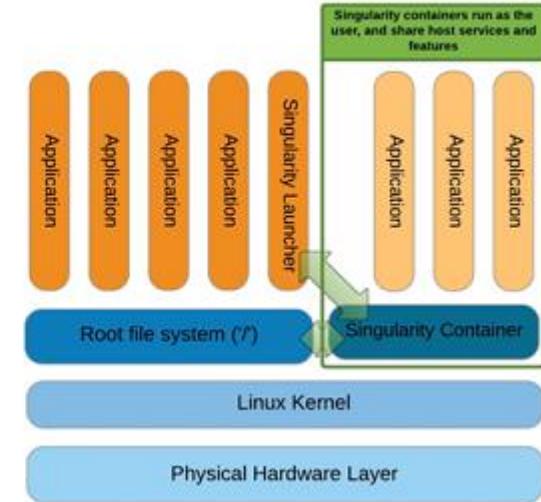
Singularity

- 支持大规模跨节点 HPC集群部署
- 可以由 root 和非 root 用户启动。容器启动前后，用户上下文保持不变
- 性能和宿主机相当，性能损失小于2%

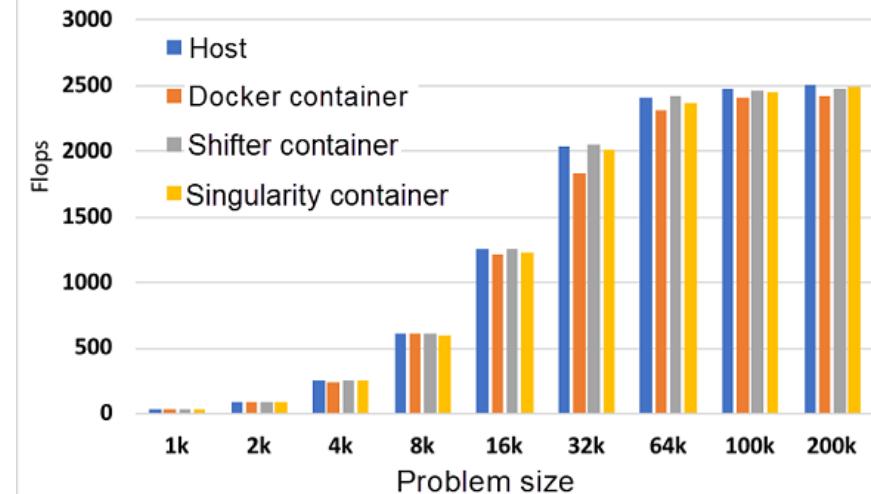
Docker



Singularity



HPL performance test



基于容器的一体化高性能计算解决方案

<https://github.com/sylabs/singularity>

- **平台容器化**

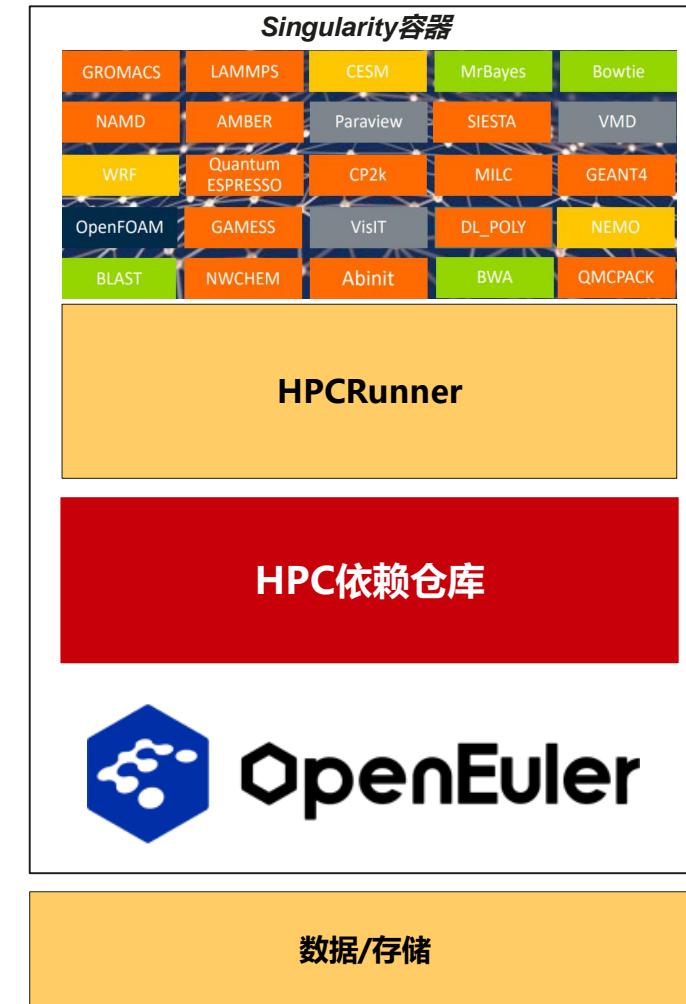
HPC平台采用统一的容器调度系统，openEuler的软件、环境在容器内实现

- **HPC应用镜像化**

整合不同软件栈的HPC应用，打包成不同的镜像模板，通过应用商店形式发布，并可进行一键创建

- **自主可控**

HPC一键部署项目和依赖仓库为自己开发，完全自主可控，可根据HPC应用的需求进行定制



HPC运行助手：一站式服务使能开发者，加速部署和性能调优

HPC运行助手：贾维斯



»»» 一键部署 »»» 一键编译 »»» 一键运行 »»» 一键性能采集 »»» 一键 Benchmark »»»

Hello
HPC in
ARM/X86!

开放部署调优经验积累
部署、调优文档工程化

- 海量依赖安装脚本
- 自动生成ModuleFile
- 海量调优patch

编译运行更“轻松”

- 一张表：HPC应用配置化
- 环境变量：自动生成环境变量，编译运行自动加载
- 自动生成Singularity容器

一键采集性能数据

- 集成跨平台perf
- 内置gemm、OMP、P2p基准性能测试，尽早发现环境问题

目标：

部署成本降低10%
调优效率提升10%

一次部署、处处部署

灵活的运行方式
固化调优成果

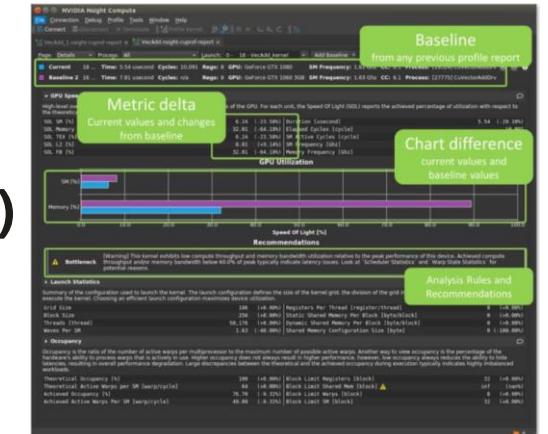
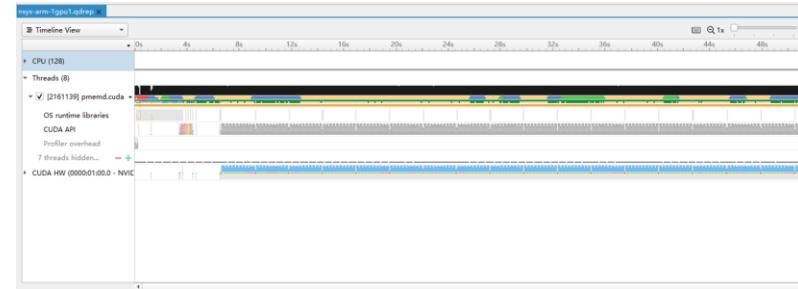
CPU/GPU基础数据收集分析

CPU性能采集流程:

- 加载配置
./jarvis –use XX.config
- 下载依赖
./jarvis –d
- 安装依赖
./jarvis –dp
- 编译并运行
./jarvis –b –r
- 性能采集(生成perf.data)
./jarvis -p

GPU性能采集流程:

- 加载配置
./jarvis –use XX.config
- 下载依赖
./jarvis –d
- 安装依赖
./jarvis –dp
- 编译
./jarvis –b
- GPU性能采集(nsys/ncu)
./jarvis -gp



基于openEuler的首款HPC应用容器

自动生成QE容器流程：

- 加载配置

./jarvis –use XX.config

- 安装singularity

./jarvis -install go/1.18 com

./jarvis -install singularity/3.9.6 gcc

- 生成容器定义文件

./jarvis –container openeuler/openeuler

- 创建容器

singularity build openeuler-kgcc9-openmpi4-qe-6.4.sif

openeuler-kgcc9-openmpi4-qe-6.4.def

```
BootStrap: docker
From: openeuler/openeuler

%environment
    source /etc/profile || true
    cd /hpcrunner
    source env.sh

%post
    # Install the necessary development environment
    yum install -y environment-modules git dmidecode pciutils wget vim
    # Install base gcc
    yum install -y gcc gcc-c++ gcc-gfortran glibc-devel make libgfortran
    # install network package
    yum install -y tcsh tcl lsof tk bc
    source /etc/profile || true
    git config --global http.sslVerify false
    git clone https://gitee.com/openeuler/hpcrunner
    cd hpcrunner
    source ./init.sh
    ./jarvis -i
    cp ./templates/qe/6.4/data.qe.container.config .
    wget --no-check-certificate https://github.com/QEF/q-e/archive/refs/tags/qe-6.4.1.tar.gz
    tar -xzf qe-6.4.1.tar.gz
    # Switch config
    ./jarvis -use data.qe.container.config
    # download dependency
    ./jarvis -d
    # install dependency
    ./jarvis -dp
    # build hpc
    ./jarvis -b
    # run hpc
    ./jarvis -r
    # clean tmp directory
    rm -rf downloads tmp
```

HPC SIG组22年工作规划

1.0: 起步 (2022)

- 支持100+HPC应用在openEuler的迁移构建，覆盖前80%算力需求
- 鲲鹏60+开源二进制仓建设，构筑自研软件栈一键式使用。
- 构建HPC应用自动部署能力，支持100+依赖自动安装，Top20应用优化部署模板(含调优patch)。
- 构建HPC应用快速调优能力，集成主流CPU/GPU调优工具，一键生成性能分析报告。
- 构建HPC应用容器化技术，Top应用容器仓库建设

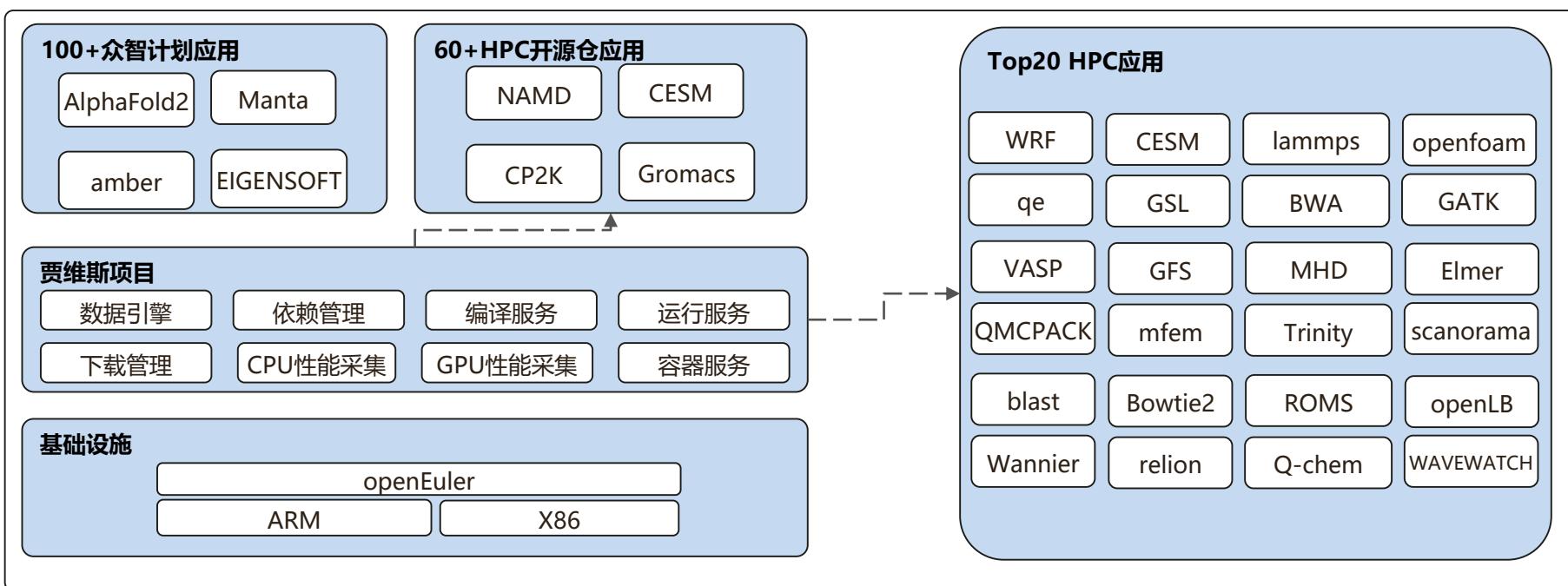
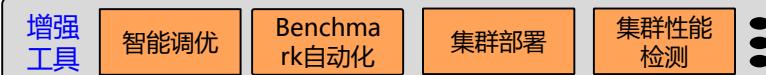


2.0: 增强 (2023~)

- 集成HPC领域常用性能调优手段、核心算法；
- 支持集群调度系统部署和性能分析工具部署。
- 支持自动化智能调优。
- 集成HPC领域常见Benchmark自动化测试
- 集成HPC应用常见workload

规划后续完成

22年内完成



总结：HPC生态建设之路险阻而又漫长，需要大家的共同努力

HPC SIG任务列表：

1.百万奖金众智任务：

https://www.hikunpeng.com/zh/ecosystem/ecology_remit?id=7

2.HPC统一迁移调优平台：

<https://gitee.com/openeuler/hpcrunneropenEuler>

首个HPC容器已通过贾维斯生成！

3.HPC应用调优patch：已发布CESM，WRF和NEMO

4.开源实习计划：<https://gitee.com/openeuler/hpcrunner/issues/>

已发布十款HPC开源实习任务

5.鲲鹏HPC开源仓建设：已上传10+ HPC应用

迸发无尽的能量

Thank you.

发现最好的我们 创造最好的OS

openEuler HPC-sig致力于建立气象、分子动力学、生物和制造等领域的生态交流圈，打造HPC多样性算力迁移调优统一平台，固化优化成果，让看似阳春白雪的HPC应用走向平民化！





助力计算流体力学仿真工具

麒麟软件有限公司

马剑

麒麟+MHT--助力计算流体力学仿真工具开发



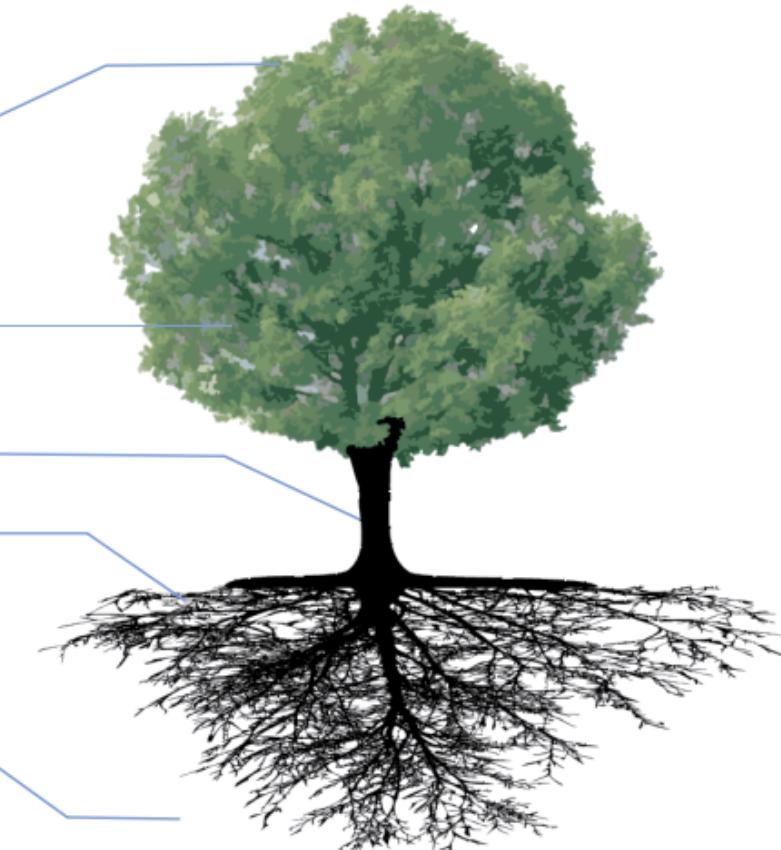
新闻链接:

https://mp.weixin.qq.com/s/fHRYlqCrMq_gUKvb8RnISQ

<https://mp.weixin.qq.com/s/1F7M1ITicTIje9IGcL3AhA>

麒麟--为中国IT系统打造坚实的OS底座

行业应用	党政	金融	电信
	石油	电力	交通
	航空航天	医疗	教育
通用软件	OA系统	邮件系统	ERP系统
	办公软件	即时通讯	工业设计
	中间件	数据库	云平台
操作系统	麒麟软件		
基础硬件	存储	网络	外设
	芯片	整机	



银河麒麟操作系统

- 党政办公桌面场景
- 行业办公桌面场景
- 教育/教学桌面场景
- 开发者桌面场景
- 服务器物理机场景
- 虚拟机GuestOS场景
- 云底座HostOS场景
- 大数据、人工智能场景
- 嵌入式、边缘计算场景
- 多端融合场景
- ...

“加快推进国产自主可控替代计划，构建安全可控的信息技术体系”

“切实提高我国关键核心技术创新能力，把科技发展主动权牢牢掌握在自己手里，为我国发展提供有力科技保障”

麒麟高级服务器V10



组织起来 联合创新

合作厂商

2,800+

硬件认证

65,000+

软件认证

25,000+

数据截至2021年8月底

更多适配信息，
请访问：<http://eco.kylinos.cn>

MHT 框架介绍



在西安交通大学NHT-CFD-EHT研究小组帮助下，西安数峰信息科技有限责任公司研发一套CFD求解平台MHT。

平台具备强大的软件架构、灵活的接口与友好的用户界面，旨在更加高效、方便的纳入最新数值计算方法。

MHT目前已搭载压力基求解器、密度基求解器、湍流模型、两相流模型等模块，已成功应用于实际工程问题。

The screenshot shows a code editor interface with two panes. The left pane is the 'Solution Explorer' showing the project structure:

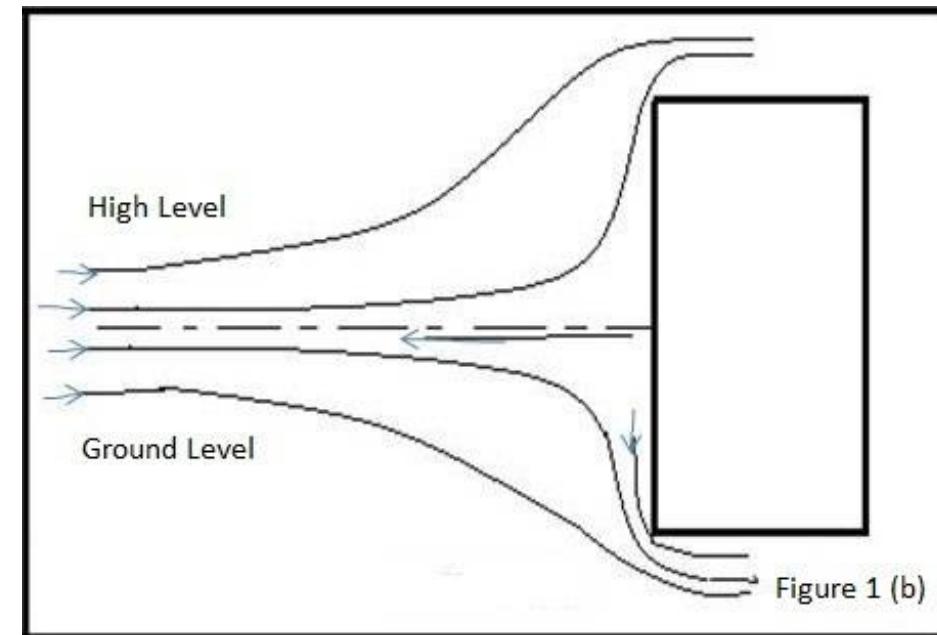
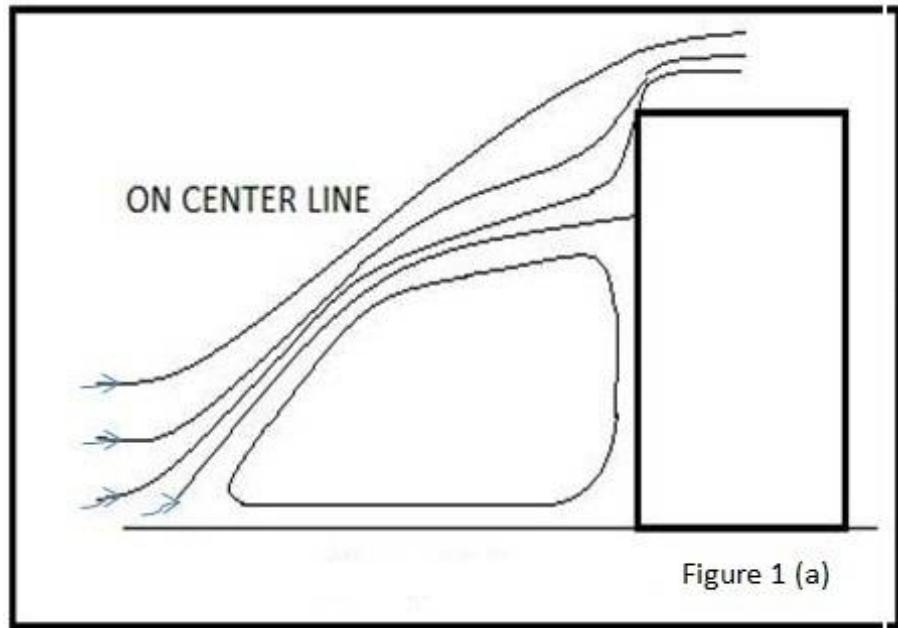
- 解决方案'MHT' (3个项目)
 - CMakePredefinedTargets
 - ALL_BUILD
 - ZERO_CHECK
 - MHT_1.5.0.3.5
 - boundaryConditions
 - cfds
 - common
 - config
 - field
 - initialConditions
 - io
 - main
 - main.cpp
 - materials
 - mesh
 - PVCouple
 - solver
 - turbulentModel
 - wallFunction
 - 外部依赖项
 - CMakeLists.txt

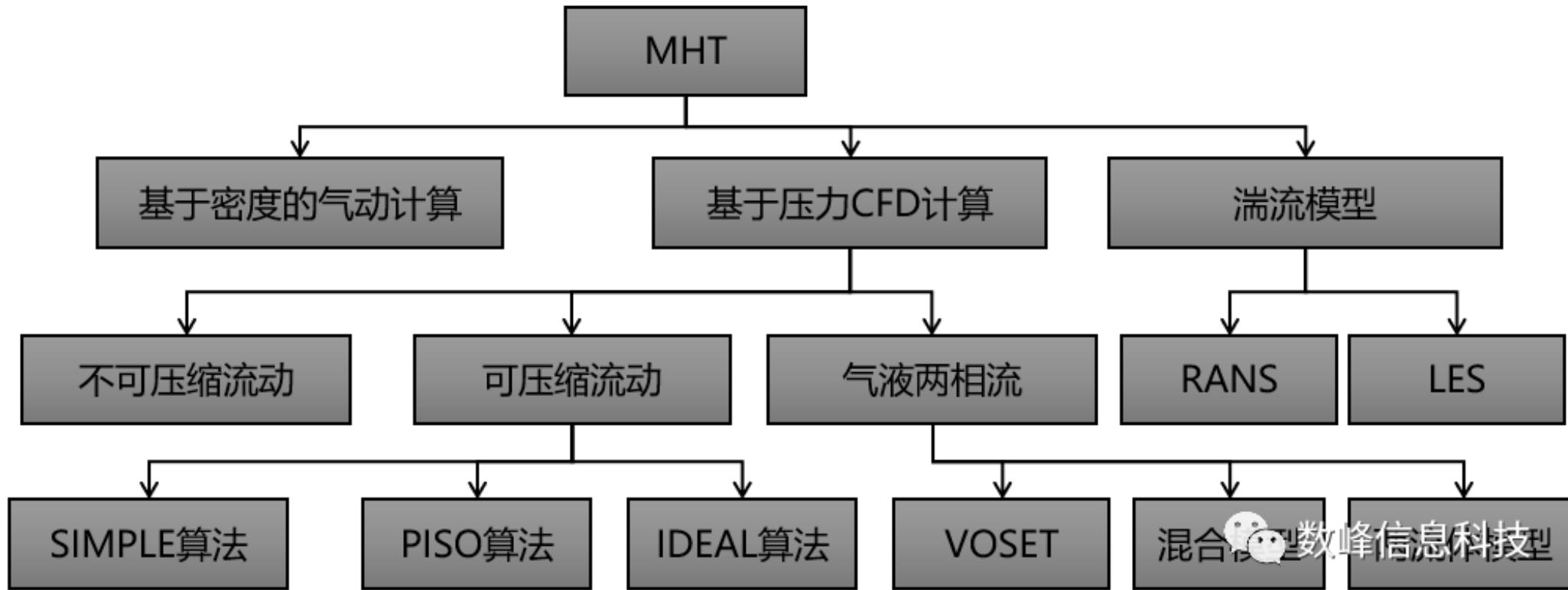
The right pane displays the content of the selected file, `main.cpp`:

```
#include <vector>
#include <iostream>
#include <time.h>
#include <ctime>
#include <string>
#include <iostream>
#include <cmath>
#include <iomanip>
//head files for mesh
#include "mesh/UnGridFactory.h"
#include "mesh/RegionConnection.h"
//head files for field
#include "field/Field.h"
#include "field/FieldManipulation.h"
#include "IO/FieldIO.h"
//head files for solver
#include "CFD/Div.h"
#include "CFD/Laplician.h"
#include "CFD/Transfer.h"
#include "CFD/Source.h"
```

At the bottom right of the editor window, there is a watermark: 数峰信息科技.

计算流体力学（英语：Computational Fluid Dynamics，简称CFD）是21世纪流体力学领域的重要技术之一，使用数值方法在计算机中对流体力学的控制方程进行求解，从而可预测流场的流动。目前有多种商业CFD软件问世，比如FLUENT、CFD-ACE+（CFDRC）、Phoenics、CFX、Star-cd等。





软件的方法层为特定的CFD算法定义标准化的求解流程。目前软件已集成基于密度的求解算法与基于压力的求解算法，同时集成了几种多相流模型与湍流模型。

麒麟--通用HPC底座



应用层

CAE设计仿真
ABAQUS
LS-DYNA

EDA设计仿真
HSPICE
Virtuoso Analog

物理化学
VASP
Gaussian

海洋气象
MM5
WRF

石油勘探
VIIP
Eclipse

生命科学
Blast
Hmmer

AI深度学习
TensorFlow
Caffe2

环境层

并行消息传递库
MPICH/OpenMP/PVM

数学库
Intel MKL/Lapack/Goto/Blas

编译器
GNU/Intel Compiler

容器
Docker/Singularity

系统层

CUI命令行界面/GUI图形界面

集群部署

集群管理

作业调度

集群监控

集群报表

许可证管理

混合云管理

资源层

Kylin V10

X86架构服务器

ARM架构服务器

分布式存储

华为云

其他公有云

适配相关专业软件(前期15款，全部共99款)

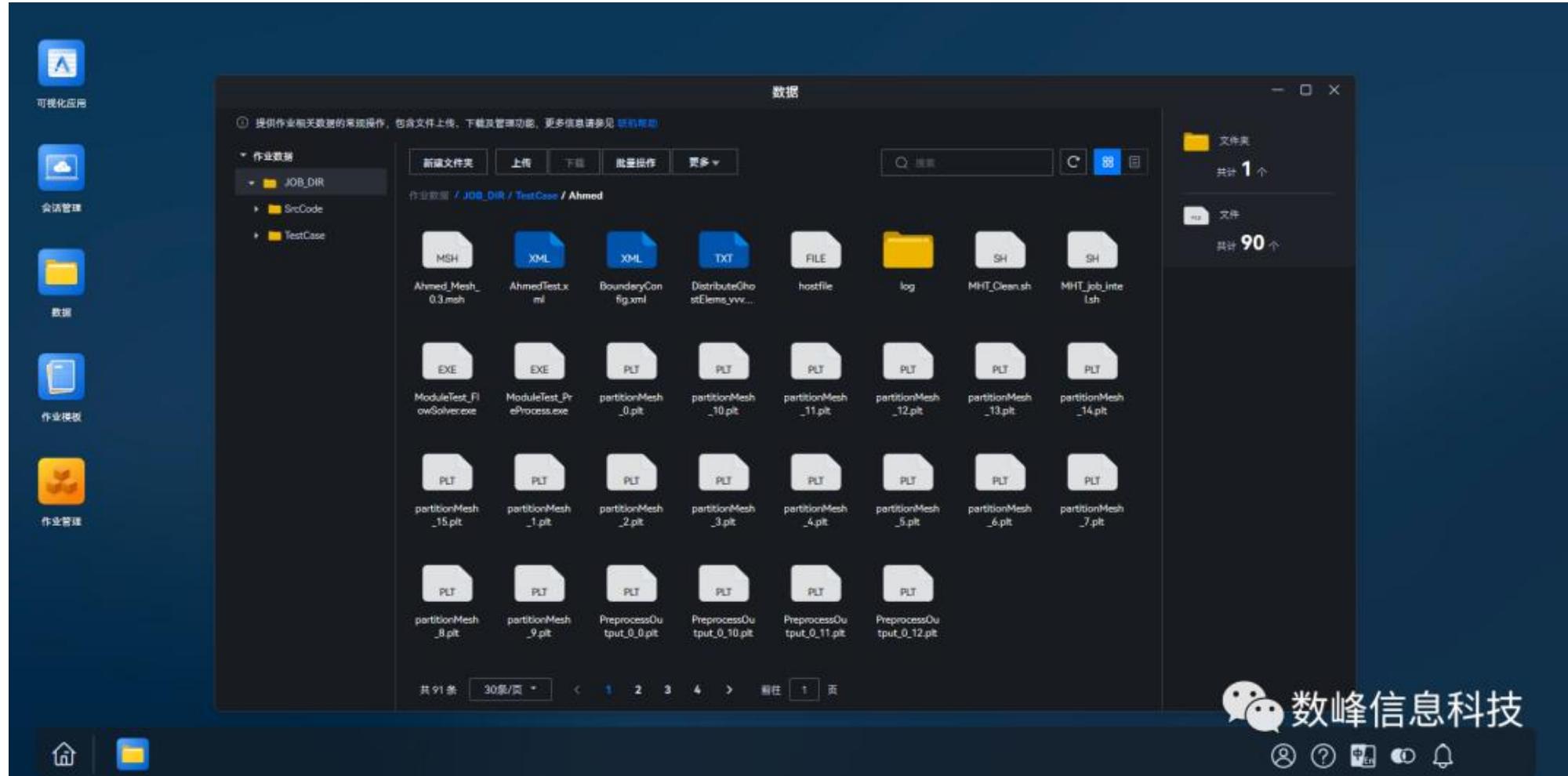
- 气象软件
- 物理化学类软件
- 生命科学类软件
- 数学类/统计类软件

	序号	软件	平台	项目
HPC_CPU场景的TOP应用	1	WRF	x86平台	西安项目
	2	nemo	x86平台	西安项目
	3	CESM	x86平台	西安项目
	4	OpenFoam	x86平台	西安项目
	5	QUANTUM ESPRESSO	x86平台	西安项目
	6	gromacs	x86平台	西安项目
	7	lammps	x86平台	西安项目
	8	CP2K	x86平台	西安项目
	9	HPL	x86平台	西安项目
	10	Chroma	x86平台	西安项目
HPC_CPU场景补充和依赖库	11	CFX fluent	x86平台	西安项目
	12	abaqus	x86平台	西安项目
	13	star ccm	x86平台	西安项目
	14	cfd++	x86平台	西安项目
	15	ansys lsdyna	x86平台	西安项目

- 提供系统调优和内核优化支持
- 提供支持库和软件包共570余个
- etcd, corosync, pacemaker, hwloc, ipmitool, xorg-x11-*, keepalived, expect, postgresql, haveged, ntp, migrationtools, TurboVNC
- OceanStor-Pacific*, MLNX_OFED_LINUX-*, python2*, python3*, qt5*, perl*, texlive*, postfix-*, pygtk2*, kf5-*, fftw-libs-double*

序号	功能类别	HPC需求功能说明	包数量	包别名
1	任务资源管理、资源隔离	计算节点	1	libcgrou
2	高可靠性代理服务器	CCS Master节点	1	haproxy
3	分布式的键值存储系统	CCS Master节点	1	etcd
4	CRM的管理接口工具	CCS Master节点	1	pcs
5	集群消息事务层	CCS Master节点	1	corosync
6	集群资源管理器	CCS Master节点	1	pacemaker
7	linux cpu拓扑查看工具	计算节点	1	hwloc
8	linux 系统下的命令行方式的 ipmi 平台管理工具	CCS Master节点	1	ipmitool
9	X11 服务端	CCPortal	1	xorg-x11-server
10	X11 远程调用能力	CCPortal	1	xorg-x11-xauth
11	完成某些特定的配置工作程序，如字体等	CCPortal	1	xorg-x11-apps
12	负载均衡器	CCPortal	1	keepalived
13	脚本自动化交互功能	全部节点	1	expect
14	数据库	数据库节点	13	postgresql
15	随机数产生工具	全部节点	1	haveged
16	提供时间同步服务	NTP	1	ntp
17	提供用户同步客户端服务所需	LDAP Client	3	nss-pam-ldapd openldap-clients oddjob
18	账号统一管理	LDAP Server	4	openldap openldap-servers openldap-devel migrationtools
19	VNC远程连接	CCPortal	1	TurboVNC
20	Intel编译器, MPI, 数学库	x86平台编译器	1	Intel_oneAPI
21	DPC Client需要的三方依赖	OceanStor-Pacific		OceanStor-Pacific_8.1.RC2_Tools.tar.gz
22	RPC主程序	NFS	1	rpcbind
23	NFS主程序	NFS	1	nfs-utils
24	分布式存储自动挂载	文件系统自动挂载	1	autofs
25	Mellanox网卡驱动 (MLX CX5、MLX CX6)	硬件	1	MLNX_OFED_LINUX-5.4-1.0.3.0-kylin10-x86_64.tgz
26	系统库支持	Python2支持库	10	python2*
27	系统库支持	Python3支持库	41	python3*
28	系统库支持	QT5支持库	5	qt5*
29	Perl系统库	Perl支持库	27	perl*
30	Tex格式系统支持	texlive支持库	165	texlive*
31	邮件	邮件支持	1	postfix-*
32	图形化支持	Python图形化支持	3	pygtk2*
33	图形化支持	KDE框架支持	80	kf5-*
34	系统库支持	fftw系统库支持	1	fftw-libs-double*

根据超算软硬件环境，先在本地配置一个相同的环境，在本地环境上编译。将编译好的库上传超算服务器，即可使用。



作业的提交及运行



通过客户端提交作业

- 命令行脚本提交

```
#!/bin/sh
#DSUB -n MHT_job
#DSUB -A root xiaofxxkjyxzrgs
#DSUB -N 1
#DSUB -R cpu=16
#DSUB --job_type cosched
#DSUB -o /home/xiaofxxkjyxzrgs/shufeng_zs/TestCase/Ahmed/log/output.txt
#DSUB -eo /home/xiaofxxkjyxzrgs/shufeng_zs/TestCase/Ahmed/log/erlog.txt
# 加载运行作业所需要的环境变量
source /share/csuuite/ENV/setenvomp411_intel.sh
MHT_EXE=/home/xiaofxxkjyxzrgs/shufeng_zs/TestCase/ModuleTest_FlowSolver.exe
# 从调度器资源分配列表中获取资源信息，并写入hostfile中
cat $SCCSCHEDULER_ALLOC_FILE awk '{print $1}'>hostfile
ROCE_Parma=-x UCX_TLS=self.mn -x UCX_NET_DEVICES=mrx5_1:0
ENV_params="--prefix /share/csuuite/basic_support/openmp411_intel -x PATH -x LD_LIBRARY_PATH"
# 执行MPI，通过mpm参数指定。此处-up指定数据不能大于从调度器申请的资源总量。
mpirun $ENV_params -mca plm_rsh_agent /opt/batch/agent/tools/start $ROCE_Parma -np 16 -hostfile hostfile $MHT_EXE ./AhmedTest.xml
```

等待资源分配

- 图形界面可以查看资源情况

作业ID	作业名称	状态	任务	用户	调度器	运行时间	创建时间	开始时间	操作
81326	MHT_job	完成	1 ●	shufeng_zs	CCSCHEDU...	00:04:51	2022/3/3 19:26:52	2022/3/3 19:26:53	重启
81324	MHT_job	失败	1 ●	shufeng_zs	CCSCHEDU...	00:00:09	2022/3/3 19:08:57	2022/3/3 19:08:57	重启
77322	MHT_job	失败	4 ●	shufeng_zs	CCSCHEDU...	00:00:06	2022/3/3 18:04:01	2022/3/3 18:04:02	重启
77321	MHT_job	失败	4 ●	shufeng_zs	CCSCHEDU...	00:00:06	2022/3/3 18:02:35	2022/3/3 18:02:36	重启
77320	MHT_job	失败	4 ●	shufeng_zs	CCSCHEDU...	00:00:06	2022/3/3 17:51:38	2022/3/3 17:51:38	重启

使用mpirun进行作业提交

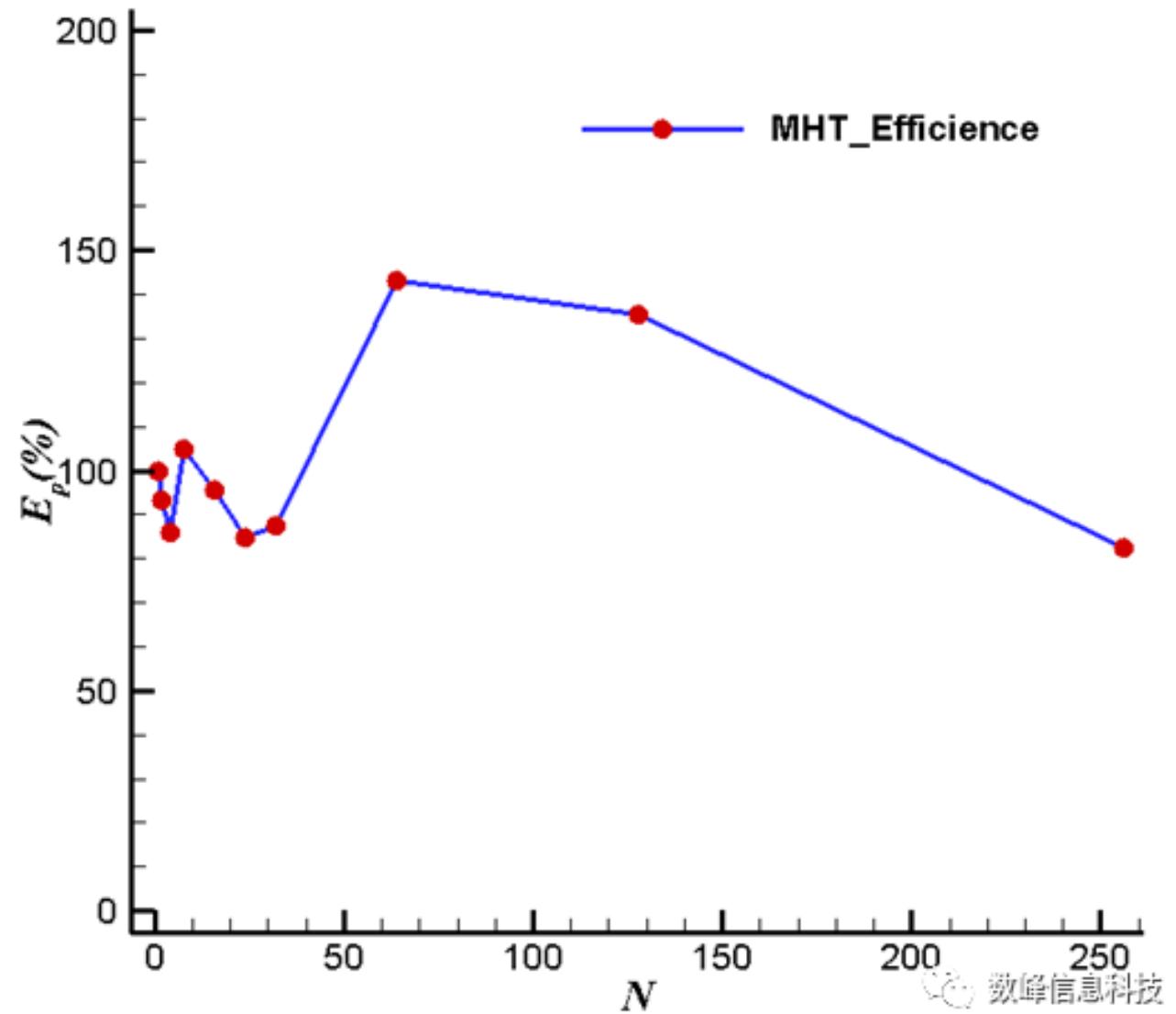
MHT_job_intel.sh

```
1 #!/bin/sh
2 #DSUB -n MHT_job
3 #DSUB -A root.xiasf0dkjyxzrgai
4 #DSUB -N 1
5 #DSUB -R cpu=16
6 #DSUB --job_type cosched
7 #DSUB -oo /home/xiasfxxkjyxxzrgai/shufeng_zs/TestCase/Ahmed/log/output.txt
8 #DSUB -eo /home/xiasfxxkjyxxzrgai/shufeng_zs/TestCase/Ahmed/log/errlog.txt
9 # 加载运行作业所需要的环境变量
10 source /share/ccsuite/ENV/setenvompi411_intel.sh
11 MHT_EXE=/home/xiasf0dkjyxxzrgai/shufeng_zs/TestCase/ModuleTest_FlowSolver.exe
12 #从调度器资源分配列表中获取资源信息，并写入hostfile中
13 cat $SCCSCHEDULER_ALLOC_FILE |awk '{print $1}'|awk NF >hostfile
14 ROCE_Parma="-x UCX_TLS=self,sm -x UCX_NET_DEVICES=mlx5_1:0"
15 ENV_Parma="--prefix /share/ccsuite/basic_suport/openmpi411_intel -x PATH -x LD_LIBRARY_PATH"
16 #执行MPI，通过mpi参数指定。此处-np指定数量不能大于从调度器申请的资源总量。
17 mpirun $ENV_Parma --mca plm_rsh_agent /opt/batch/agent/tools/dstart $ROCE_Parma -np 16 -hostfile hostfile $MHT_EXE ./AhmedTest.xml
```

并行效率



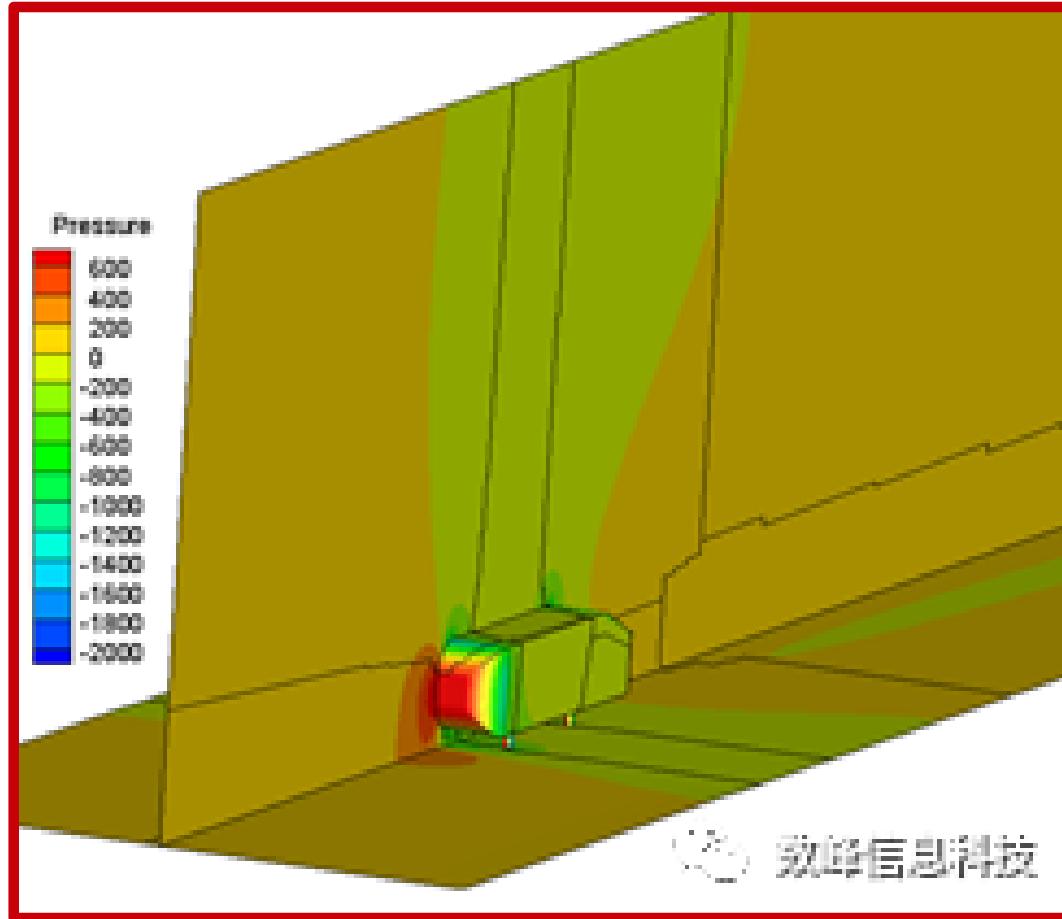
单核网格负载在10万网格时，并行效率较高，MHT在1-256核并行求解速度压力耦合方法时，并行效率均在80%以上。



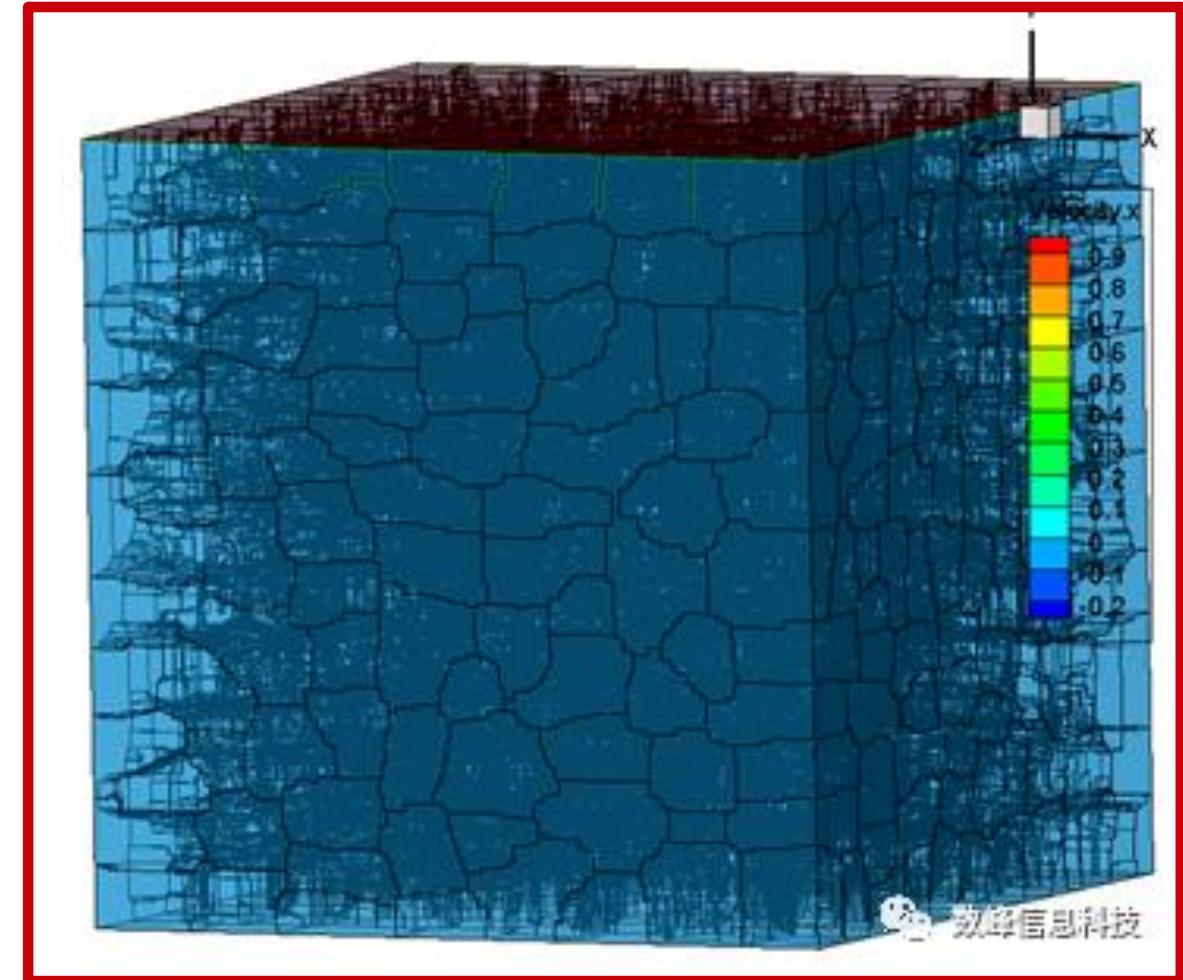
运算结果图



三维云图：物面压力分布。



千核并行网格剖分测试，结果证明了MHT千核并行的正确性。





OpenEuler

谢谢

加速库在HPC领域的应用

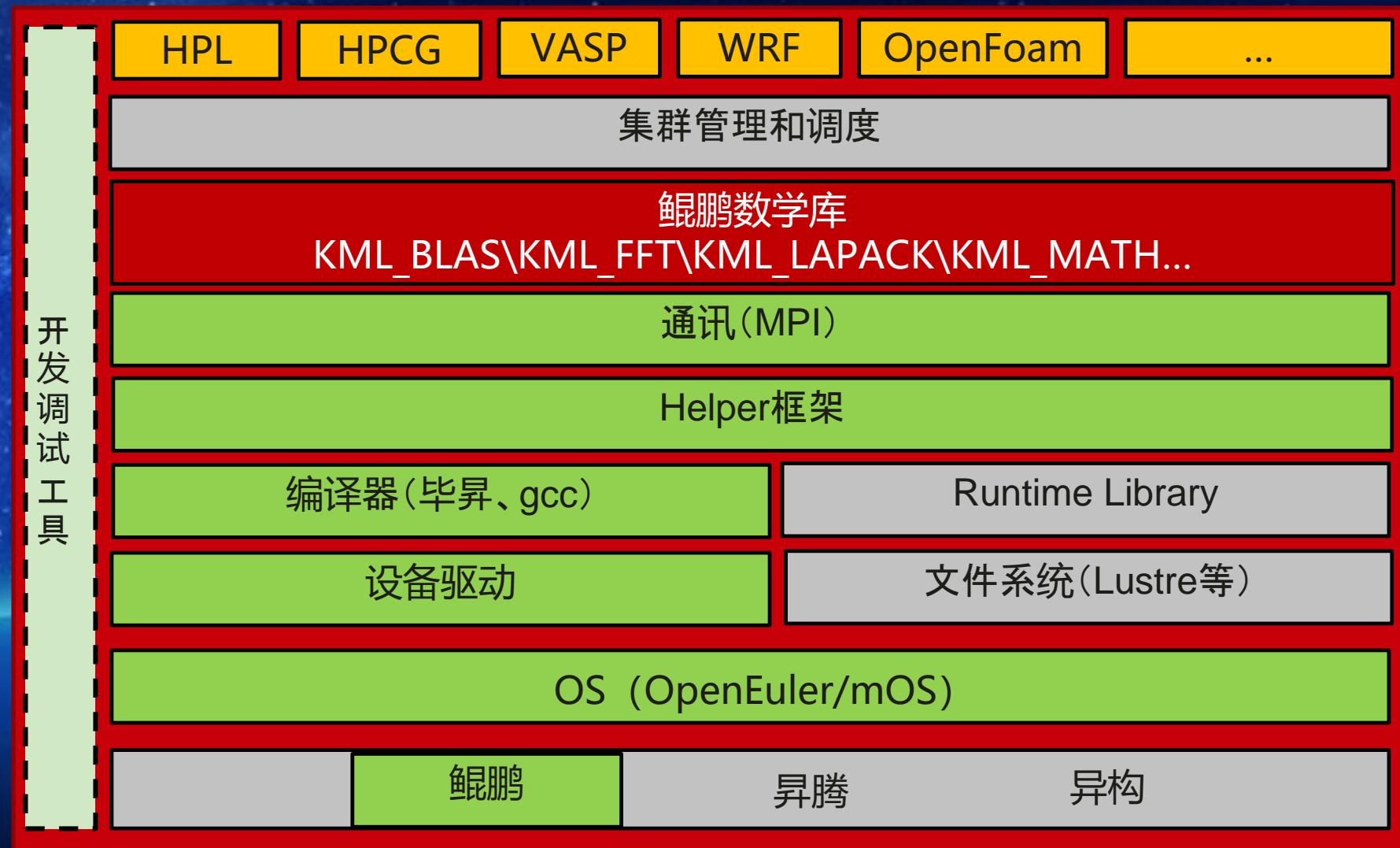
钟可以 ACC SIG Maintainer



目录

1. HPC软件栈全览
2. 使用加速库加速CP2K
3. 加速库如何加速

HPC软件栈全览



CP2K是什么

- CP2K是一个量子化学和固态物理软件包，可以对固态，液态，分子，周期性，材料，晶体和生物系统进行原子模拟。CP2K为不同的建模方法提供了通用框架。支持的理论水平包括DFTB, LDA, GGA, MP2, RPA, 半经验方法 (AM1, PM3, PM6, RM1, MNDO等) 和经典力场 (AMBER, CHARMM等)。CP2K可以使用NEB或二聚体方法进行分子动力学，元动力学，蒙特卡洛，埃伦菲斯特动力学，振动分析，核心能谱，能量最小化和过渡态优化的模拟。
- CP2K做分子模拟其实主要涉及生物和材料两个方面：
 - › 在生物方面，主要研究小分子与蛋白质的作用机制问题，筛选先导化合物，缩短药物研发的周期，目前国内大的药企，基本上都会在前期采用分子模拟方法进行初步研究。
 - › 在材料方面，可以研究不同材料的性质，吸附能力，对于材料的细节模拟，涉及更高性能的材料有着指导意义。



H2O-64测试用例

- H2O-64：由64个水分子组成的系统。
- 测试用例输入：H2O-64.inp
 - › 表示64个水分子中原子（64个O原子，128个H原子）在初始状态时的坐标。
- 测试过程：模拟水分子运动的迭代过程。
- 测试用例输出：系统的能量值ENERGY。
- 测试标准：与官方版本结果误差在 $1 \text{ e-}3$ 以内视为测试通过。

```
***** **** * ***** ** PROGRAM STARTED AT          2022-02-26 14:43:14.362
***** ** ***   *** ** PROGRAM STARTED ON        ecs-kp-test
**   ****   ***** PROGRAM STARTED BY           root
***** **   ** * * ** PROGRAM PROCESS ID       611829
***** **   *****   ** PROGRAM STARTED IN      /home/cp2k-7.1.0/benchmarks/QS
```

耗时27分26秒

```
***** **** * ***** ** PROGRAM ENDED AT          2022-02-26 15:10:40.191
***** ** ***   *** ** PROGRAM RAN ON          ecs-kp-test
**   ****   ***** PROGRAM RAN BY            root
***** **   ** * * ** PROGRAM PROCESS ID       611829
***** **   *****   ** PROGRAM STOPPED IN     /home/cp2k-7.1.0/benchmarks/QS
[root@ecs-kp-test QS]#
```

KML数学库加速CP2K

- 识别数学库并替换

```
LIBS      += $(MATHLIBPATH)/libscalapack.a
LIBS      += $(MATHLIBPATH)/liblapack.a
LIBS      += $(MATHLIBPATH)/librefblas.a
LIBS      += -ldl -lpthread -lstdc++
```

```
LIBS      += $(MATHLIBPATH)/libscalapack.a
LIBS      += $(MATHLIBPATH)/liblapack.a
LIBS      += -L/usr/local/kml/lib/kblas/nolocking/ -lkblas
LIBS      += -ldl -lpthread -lstdc++
```

替换为KML数学库

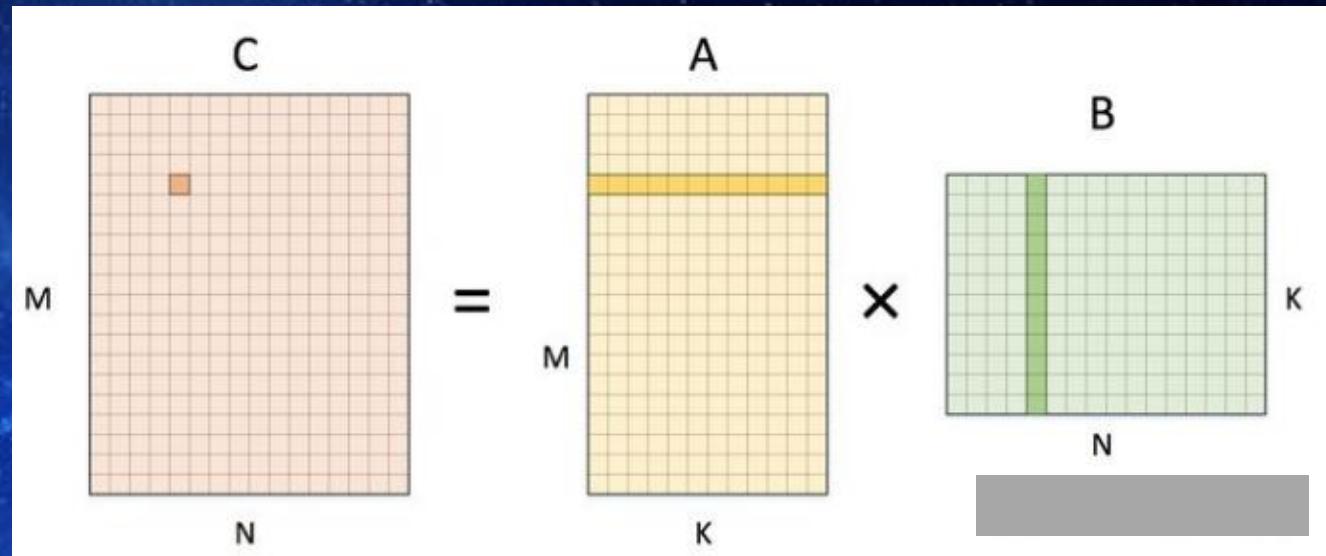
- 测试结果

```
***** **** * PROGRAM STARTED AT 2022-02-26 14:14:52.610
***** *** *** * PROGRAM STARTED ON ecs-kp-test
** **** * PROGRAM STARTED BY root
***** ** ** * * PROGRAM PROCESS ID 609968
***** ** * ***** * PROGRAM STARTED IN /home/cp2k-7.1.0/benchmarks/QS
```

耗时18分6秒

```
***** **** * PROGRAM ENDED AT 2022-02-26 14:32:58.627
***** *** *** * PROGRAM RAN ON ecs-kp-test
** **** * ***** PROGRAM RAN BY root
***** ** ** * * PROGRAM PROCESS ID 609968
***** ** * ***** * PROGRAM STOPPED IN /home/cp2k-7.1.0/benchmarks/QS
[root@ecs-kp-test QS]#
```

Gemm:教科书算法

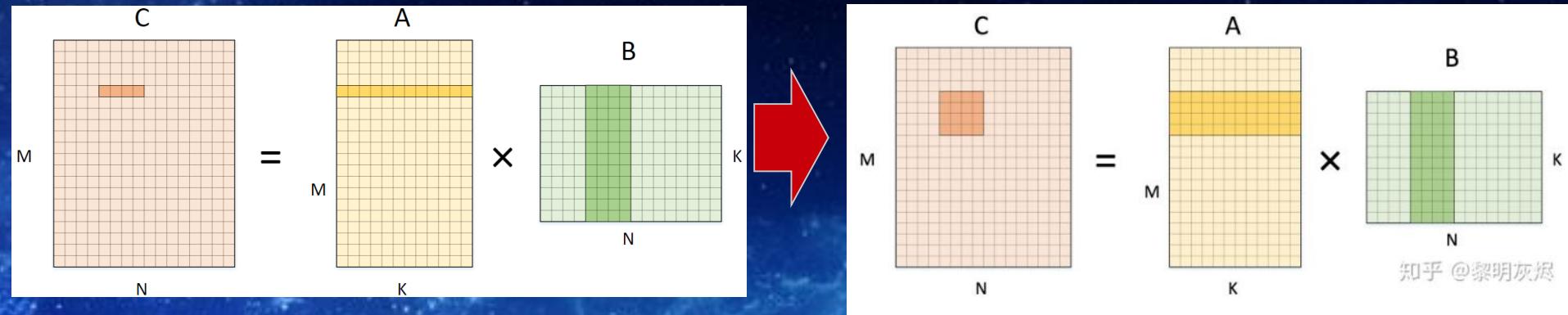


```
for (int m = 0; m < M; m++) {  
    for (int n = 0; n < N; n++) {  
        C[m][n] = 0;  
        for (int k = 0; k < K; k++) {  
            C[m][n] += A[m][k] * B[k][n];  
        }  
    }  
}
```

计算次数: $2^* (M*K*N)$

访存次数: $4^* (M*K*N)$

Gemm优化 (1) : 循环展开, 分块展开



```

for (int m = 0; m < M; m++) {
    for (int n = 0; n < N; n += 4) {
        for (int k = 0; k < K; k++) {
            C[m][n + 0] += A[m][k] * B[k][n + 0];
            C[m][n + 1] += A[m][k] * B[k][n + 1];
            C[m][n + 2] += A[m][k] * B[k][n + 2];
            C[m][n + 3] += A[m][k] * B[k][n + 3];
        }
    }
}

```

1x4 kernel

计算次数: $2 * (M * K * N)$

访存次数: $(2 + 1 + \frac{1}{4} = 13/4) * M * N * K$

优化: 充分利用寄存器, A[m][k]访存次数减少3/4

```

for (int m = 0; m < M; m += 4) {
    for (int n = 0; n < N; n += 4) {
        for (int k = 0; k < K; k++) {
            C[m + 0][n + 0..3] += A[m + 0][k] * B[k][n + 0..3];
            C[m + 1][n + 0..3] += A[m + 1][k] * B[k][n + 0..3];
            C[m + 2][n + 0..3] += A[m + 2][k] * B[k][n + 0..3];
            C[m + 3][n + 0..3] += A[m + 3][k] * B[k][n + 0..3];
        }
    }
}

```

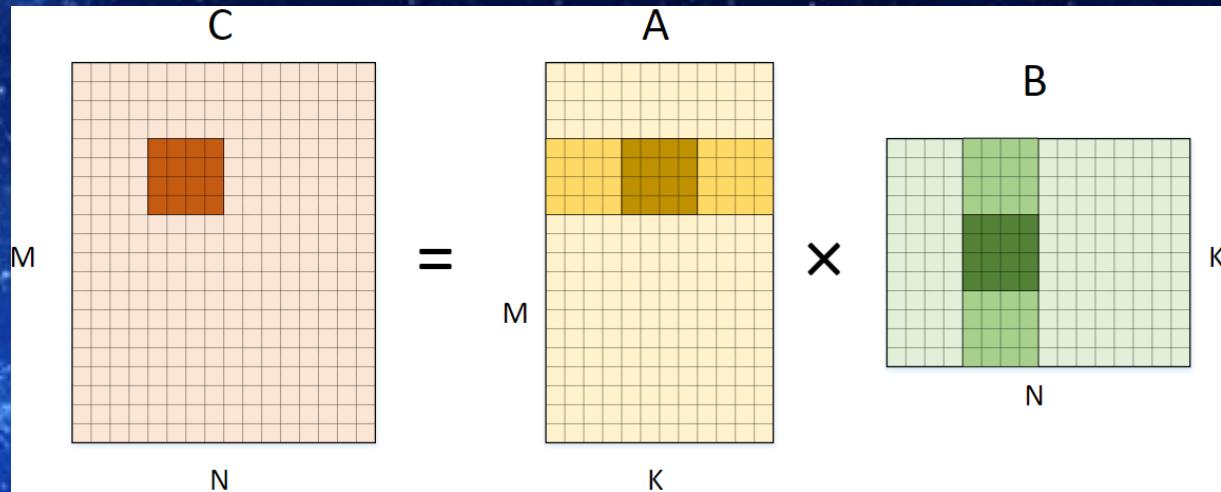
4x4 kernel

计算次数: $2 * (M * K * N)$

访存次数: $(2 + 1/4 + 1/4 = 5/2) * M * N * K$

优化: 充分利用寄存器, A和B访存次数分别减少3/4

Gemm优化 (1) : 循环展开, 分块展开



```

for (int m = 0; m < M; m += 4) {
    for (int n = 0; n < N; n += 4) {
        for (int k = 0; k < K; k += 4) {
            C[m + 0..3][n + 0..3] += A[m + 0..3][k + 0] * B[k + 0][n + 0..3];
            C[m + 0..3][n + 0..3] += A[m + 0..3][k + 1] * B[k + 1][n + 0..3];
            C[m + 0..3][n + 0..3] += A[m + 0..3][k + 2] * B[k + 2][n + 0..3];
            C[m + 0..3][n + 0..3] += A[m + 0..3][k + 3] * B[k + 3][n + 0..3];
        }
    }
}

```

4x4 kernel

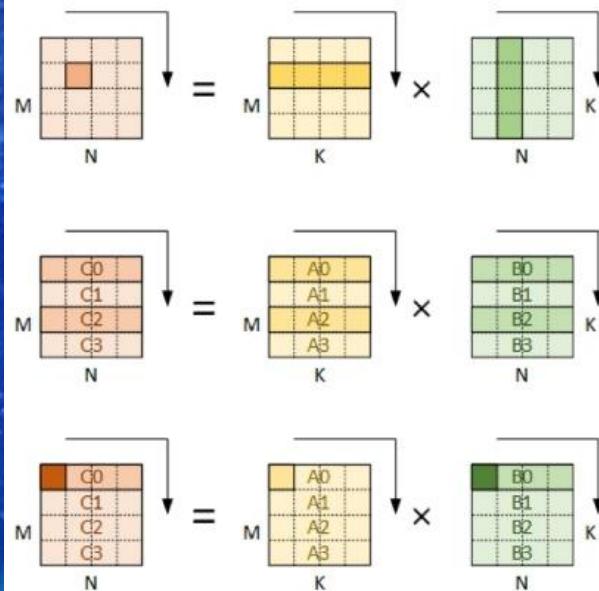
计算次数: $2 * (M * K * N)$

访存次数: $(1/4 + 1/4) * M * N * K + MN$

数据优化: A和B访存次数分别减少
3/4, C矩阵临时数据放到寄存器, 访
存次数为MN, 相比MNK成功降低一
维! ! ! , 相比教科书算法, 性能提
升8倍以上

Gemm优化 (2) : 向量化

(4x4)x(4x4) Vector load/store/arithmetic



```
// C0 in detail
C0[0] += A0[0] * B0[0]
C0[0] += A0[1] * B1[0]
C0[0] += A0[2] * B2[0]
C0[0] += A0[3] * B3[0]

C0[1] += A0[0] * B0[1]
C0[1] += A0[1] * B1[1]
C0[1] += A0[2] * B2[1]
C0[1] += A0[3] * B3[1]

C0[2] += A0[0] * B0[2]
C0[2] += A0[1] * B1[2]
C0[2] += A0[2] * B2[2]
C0[2] += A0[3] * B3[2]

C0[3] += A0[0] * B0[3]
C0[3] += A0[1] * B1[3]
C0[3] += A0[2] * B2[3]
C0[3] += A0[3] * B3[3]
```

```
// C0 scheduled
C0[0] += A0[0] * B0[0]
C0[1] += A0[0] * B0[1]
C0[2] += A0[0] * B0[2]
C0[3] += A0[0] * B0[3]

C0[0] += A0[1] * B1[0]
C0[1] += A0[1] * B1[1]
C0[2] += A0[1] * B1[2]
C0[3] += A0[1] * B1[3]

C0[0] += A0[2] * B2[0]
C0[1] += A0[2] * B2[1]
C0[2] += A0[2] * B2[2]
C0[3] += A0[2] * B2[3]

C0[0] += A0[3] * B3[0]
C0[1] += A0[3] * B3[1]
C0[2] += A0[3] * B3[2]
C0[3] += A0[3] * B3[3]
```

```
// (4x4)*(4x4)
Load C0-C3
Load A0-A3
Load B0
C0 += A0[0] * B0
C1 += A1[0] * B0
C2 += A2[0] * B0
C3 += A3[0] * B0
Load B1
C0 += A0[1] * B1
C1 += A1[1] * B1
C2 += A2[1] * B1
C3 += A3[1] * B1
...
Load B3
C0 += A0[3] * B3
C1 += A1[3] * B3
C2 += A2[3] * B3
C3 += A3[3] * B3
Store C0-C3
```

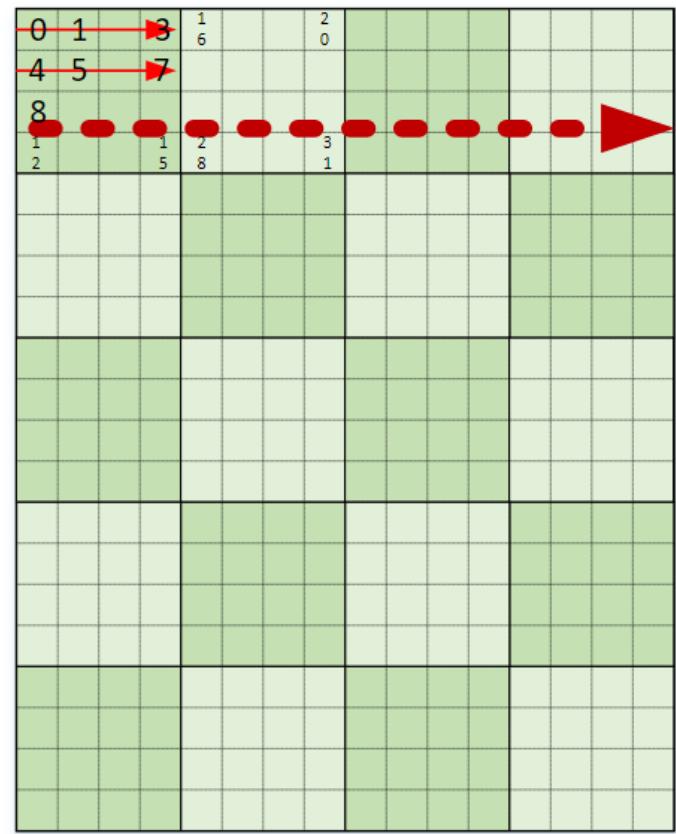
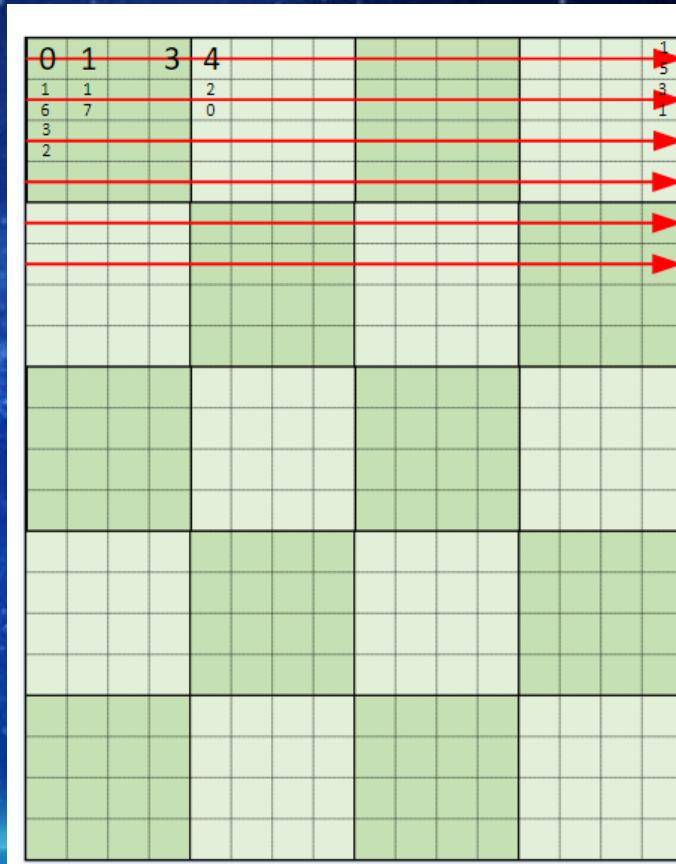
朴素乘法

计算重排

可向量化算法

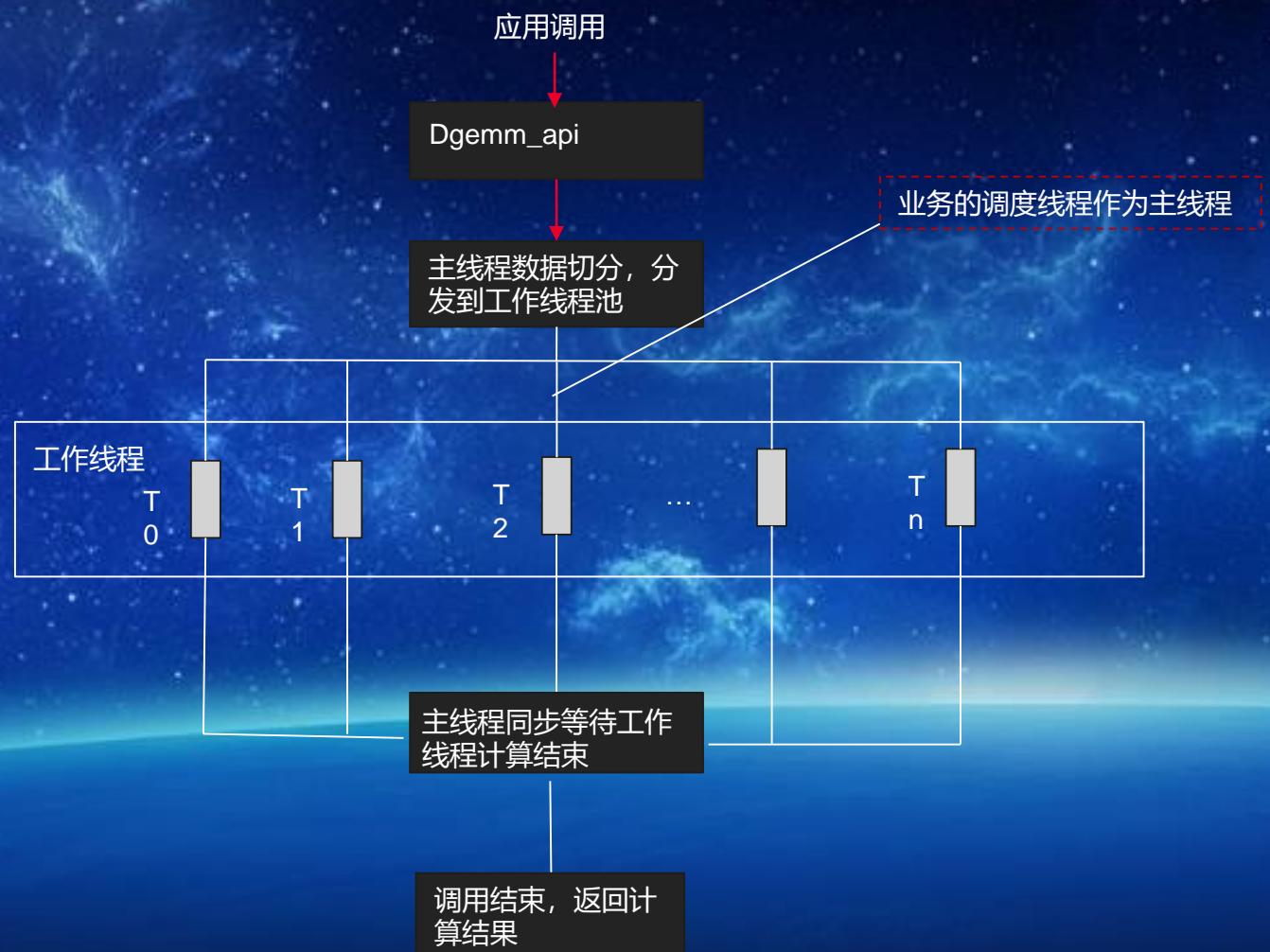
- 1、假定为行存矩阵，朴素算法A矩阵访存连续，B矩阵不连续；优化后A、B访存都连续；
- 2、重排后，通过向量化，计算指令个数从4*4*4减少为4*4；

Gemm优化 (3) : 内存布局



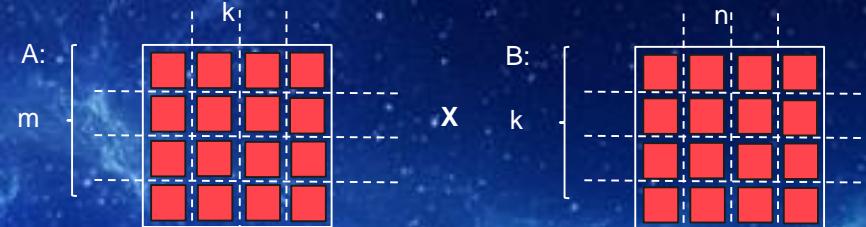
在矩阵计算前，按照计算kernel需要的数据区间，提前完成数据pack，避免运算过程中出现的cache miss和tlb miss等严重影响计算性能的系统操作

Gemm优化 (4) : 多线程并发

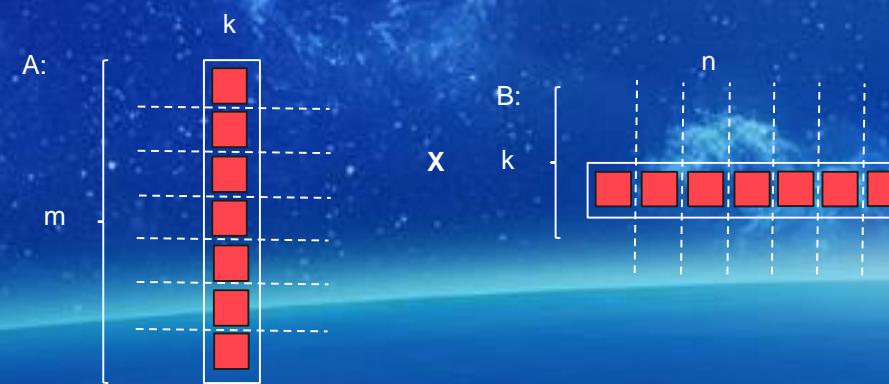


数据多线程切分

普通方形矩阵双向进行切分



长条矩阵单向进行切分



小矩阵不进行切分



通常矩阵计算时，会提前将大矩阵进行packing，

将矩阵根据分配的线程数量切分成多个小矩阵，从

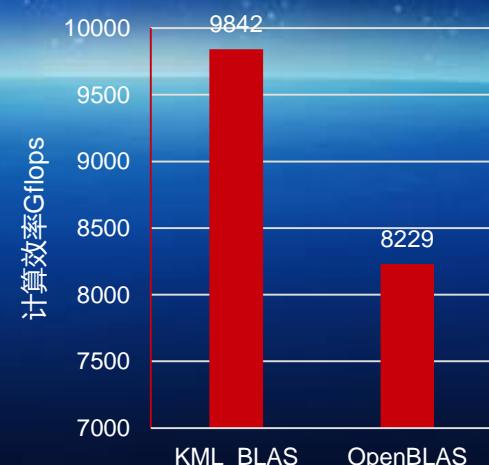
而使单个线程内计算效率达到最高

小矩阵、长条矩阵优化：根据形状，改变packing策略

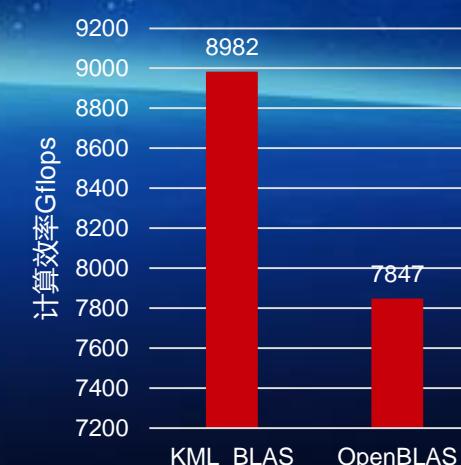
-长条形矩阵：仅对长边进行packing，省略短边的
packing过程

-小矩阵：不进行packing，直接计算

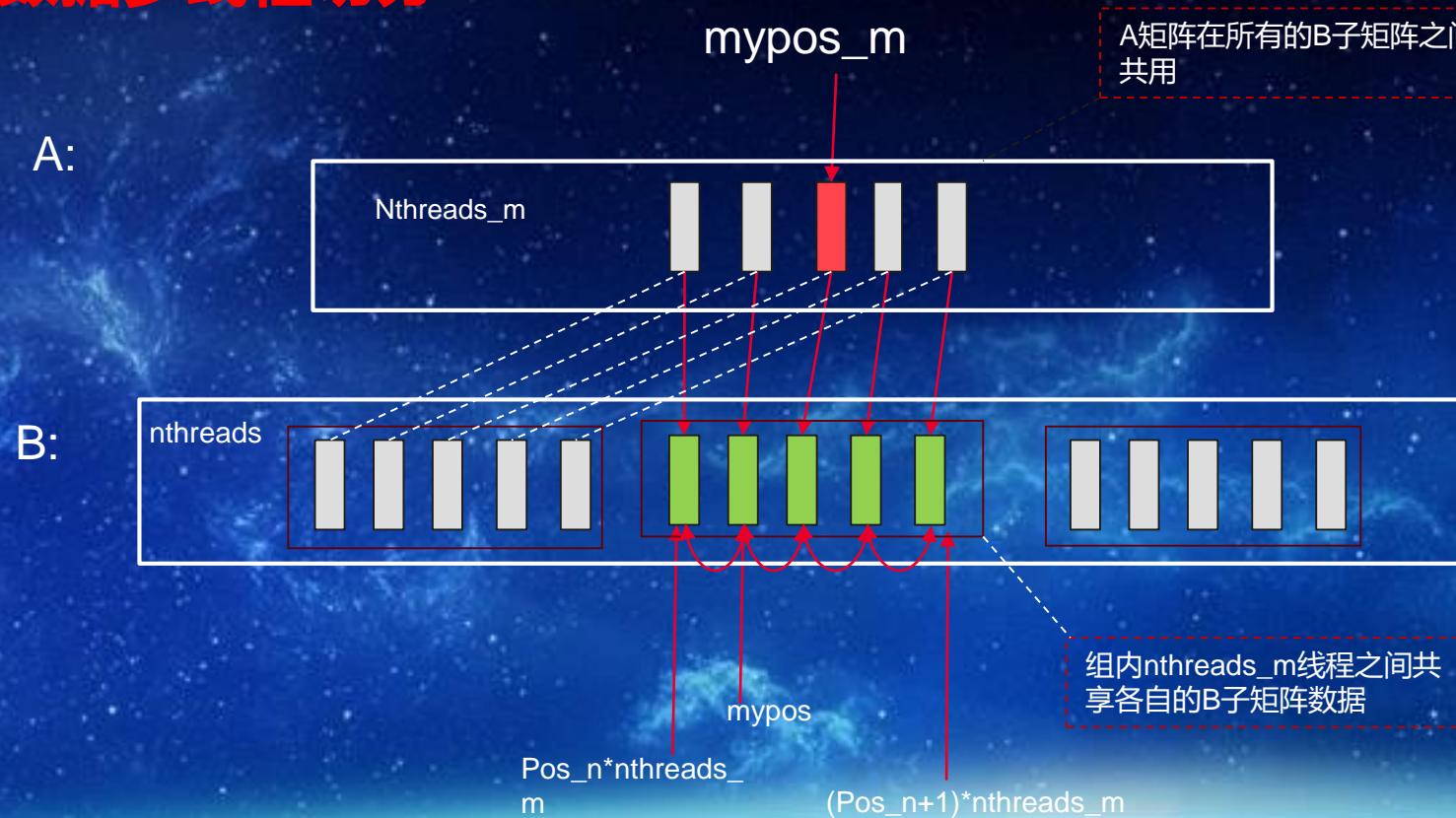
长条形矩阵性能对比



小矩阵性能对比



数据多线程切分



1. m,n 维度不同颗粒度`nthreads_m,nthreads`切分
2. $nthreads = nthreads_m * nthreads_n$
3. $mypos_m = mypos \% nthreads_m$, 其中mypos是切分后的该子任务索引序列`nthreads_m`
4. B切分数据会在`nthreads_m`组线程内共享

Thank you

openEuler HPC-sig致力于建立气象、分子动力学、生物和制造等领域的生态交流圈，打造HPC多样性算力迁移调优统一平台，固化优化成果，让看似阳春白雪的HPC应用走向平民化！