

# 请不要外传 :)

## 感谢各位同学 - "教学相长"嘛

### 目录

[请不要外传 :\)](#)

[感谢各位同学 - "教学相长"嘛](#)

[目录](#)

[导读：这本书是写啥的？ - 大规模计算](#)

[作者对大规模计算的体会](#)

[英雄所见略同](#)

[According to Università degli studi di Genova:](#)

[作者按：其突出的关键字是 multiple, distributed nodes - Map-Reduce paradigm，从作者的角度而言，是有些狭窄了：基本上就是落脚到大数据\(Big Data\)的领域。](#)

[According to Large-scale computing: the case for greater UK coordination September 2021](#)

[作者按：英雄所见略同。](#)

[本书内容组织](#)

[LSC?](#)

[HPC, DL, Big Data - 三者的硬件、软件基础在趋同](#)

[LSC程序的样子 - 数据巨大，业务复杂](#)

[以天气预报/热传导方程的求解为例](#)

[intro](#)

[finite-difference method \(FDM\)](#)

[Explicit method stencil](#)

[模型 - WRF](#)

[深度学习 - CNN、ChatGPT](#)

[商务系统 - 阿里淘宝, Google, 滴滴, 高德地图等](#)

[编写 LSC程序的套路](#)

[问题的分解：Divide & Conquer](#)

[运行的环境：多 - 核，节点](#)

[DAOM/PCAM](#)

[编程框架](#)

[硬件环境](#)

[软件环境 - 协议栈](#)

[软件环境 - 大数据 - 以 Hadoop 为例](#)

[代码 - 串行](#)

[热传导方程的数值求解 - 概念](#)

[\[Images\] heatEqu2DSeqImg.py](#)

[\[Animation\]](#)

[\[Show\]](#)

[算法级编程](#)

[MPI4PY](#)

[MPI - 概念](#)

[MPI4PY 编程基础](#)

[\[Images\] Stripe](#)

[线索](#)

[我的代码的效果](#)

[\[我的代码\]](#)

[\[Images\] Stencil](#)

[线索](#)

[\[我的代码\]](#)

## PyCUDA

CUDA 编程基本概念

[Images]PyCUDA

线索

我的实践

[我的代码]

## Numba

从OpenMP 到 Python 的 Numba

[Images]Numba

线索

我的代码的效果

但是, 当数据是 4000\*4000 时, 就出问题了

[我的代码]

## PySpark

大数据和 Hadoop 编程 - 体会 基于<key,Value>模式的编程 [Spark处理仍然如此]

PySpark 编程基础

[Images] PySpark

线索

[学习 PySpark 的代码]

[我的代码]

## 混合 - MPI+ [based on Stripe]

MPI+PyCUDA

[我的代码]

MPI+Numba.CUDA

[我的代码]

## HPPython by C/C++ - High Performance Python

Cython

概述

Cython

例子1 - Hello

例子2 - 复杂

PyBind

概述

例子1

Pythran

概述

例子

小结

## 系统级编程

CNN - from scratch with Python

线索

Hands-On GPU Programming with Python and CUDA Explore high-performance parallel computing with CUDA by Dr. Brian Tuomanen (z-lib.org) - PyCuda-master\Chapter09

PyTorch 的设计与实现

[你的理解]

## 互联网商务系统

盈利

在黏着海量用户的基础上, 收益:

关键点

秒杀 的应对

计算广告

Platform 挣广告的钱?

计算广告? - 尽可能精准找到对产品/服务有潜在购买兴趣的客户

算法

Recall 的部分算法

Ranking 的部分算法

[CTR 预估算法](#)

[线索](#)

[\[代码\]](#)

## 例子

[分布式机器学习](#)

[PyTorch 的设计与实现](#)

[Spark MLlib](#)

[秒杀](#)

[计算广告](#)

[CTR](#)

[DeepCTR-Torch](#)

[Models List](#)

[DiscussionGroup & Related Projects](#)

[deep-ctr-prediction \[TF\]](#)

[FuxiCTR](#)

[Key Features](#)

[Model Zoo](#)

[Dependencies](#)

[Quick Start](#)

[Citation](#)

## 数据库

[分布式数据库？云原生数据库？](#)

[\[PolarDB-X\] - a MySQL branch - Cloud-native DB](#)

[PolarDB-X Engine](#)

[X Engine Overview](#)

[PolarDB-X Operator](#)

[\[ApsaraDB\] - PolarDB for PostgreSQL \(hereafter simplified as PolarDB\)](#)

## 大数据

[K8S](#)

[\[你的理解\]](#)

[大数据软件的制作 - Zookeeper 为例\(?\)](#)

[线索](#)

[\[你的理解\]](#)

## 总结

[内容概要](#)

[意犹未尽呀~](#)

[几本书](#)

[褚时健传 - 了解下这位老人，体会下“商务思维” - 推荐](#)

[Python Parallel Programming Cookbook - 推荐](#)

[High Performance Computing: Modern Systems and Practices By 作者: Thomas Sterling,Matthew Anderson,Maciej Brodowicz](#)

[高性能科学与工程计算 \[德\] Georg Hager, \[德\] Gerhard Wellein](#)

[高性能计算应用概览 作者: 厉军](#)

[云原生数据库：原理与实践. 李飞飞 \(男\)](#)

[几个视频](#)

[预祝大家鹏程万里！](#)

# 导读：这本书是写啥的？ - 大规模计算

# 作者对大规模计算的体会

教学和研究过程中，日益体会到：

1. 现实大厂的业务系统，
2. 往往依赖于所谓的数据中心，那可是将大数量的计算机(很多都是上千台)连接在一起供企业使用
3. 科学计算
4. DL的训练
- 5.

实际的教学体系，往往都是分门别类地讲授相关的专题，鲜有专门的课程予以完整地阐述。

## 英雄所见略同

### According to Università degli studi di Genova :

~~Large scale Computing generally refers to the capability of hardware and software systems to dynamically adapt to an increasing load typically employing multiple, distributed nodes to complete a given processing task. Since we are in the Big Data Era, Large Scale Computing models and frameworks are becoming necessary for Data-intensive computations, a class of computing applications which use a data parallel approach to process large volumes of data based on the Map-Reduce paradigm.~~



~~Learning the theoretical, methodological, and technological fundamentals of advanced data processing architectures, large scale distributed environments, and data intensive programming including Docker, HDFS, Hadoop, Spark, and Cloud/IoT platforms.~~

#### SYLLABUS/CONTENT:

- ~~Introduction to Distributed Systems and Cloud Computing~~
- ~~Distributed data systems and shared nothing architectures~~
- ~~Partitioning & Replication~~
- ~~Fault Tolerance~~
- ~~CAP Theorem~~
- ~~Hadoop & MapReduce (incl. HDFS, Hadoop Runtime)~~
- ~~Spark (Internals, RDD Programming, Dataframes, Spark Streaming)~~

~~作者按：其突出的关键字是 multiple, distributed nodes, Map-Reduce paradigm，从作者的角度而言，是有些狭窄了：基本上就是落脚到大数据(Big Data)的领域。~~

## According to Large-scale computing: the case for greater UK coordination September 2021

"D:\Local++写书\18 高性能计算\HPCBook-MD\materials\UK\_Computing\_report-Final\_20.09.21.pdf"

Large-scale computing is an essential tool for solving industrial and scientific problems. It is used to analyse **highly complex or data-intensive problems** involving **simulation, optimisation, data processing and artificial intelligence (AI)**. Large-scale computing supports a range of R&D-intensive sectors and is used as a research tool at many of the frontiers of science. Several large-scale computing systems, such as those used for **weather forecasting**, form part of the UK's critical national infrastructure. Computing needs vary substantially across sectors, with users having different requirements in terms of both capability and support requirements:

- In the public sector, service reliability is a key consideration. For public sector users, cybersecurity and the physical locations of infrastructure are other key considerations.
- Academia requires access to systems of a range of sizes and architectures owing to the diversity of programs run.
- Industrial users of large-scale computing use a wide range of access models. Private sector use of public systems often includes a requirement to publish results and cybersecurity is a key concern.
- Small and medium enterprises (SMEs) often do not have an awareness of the range of business problems that large-scale computing can solve or the technical requirements to use it. In addition, SMEs often require support in adapting their software to run on large systems.

Large-scale computing is a fast-changing field, where systems have a working life of five to eight years before becoming obsolete. We are entering an era of unprecedented hardware diversification, driven by the development of new computing architectures for specific tasks, in particular for AI research. Advances in data science have also driven demand for large-scale computing. As a society, we are generating far more data than ever before, and without the means to analyse it, its value is left unrealised.

Cloud computing has become an increasingly popular access model. Cloud access provides opportunities for new users to reap the benefits of many (but not all) forms of large-scale computing by offsetting upfront costs with cost of use. Users can assess their needs and choose a cost model to suit, whether that be onsite hosting or access via the cloud. Commercial cloud service providers are increasing their range of capabilities. This is still a nascent sector, and the market is currently dominated by a limited number of leading providers based in the US.

Large-scale computing (LSC) has revolutionised our lives in fields of national importance such as weather and climate modelling, and financial services. It is an important enabler of R&D as demonstrated by DeepMind's remarkable recent breakthrough on protein folding. Computing power also underpins key technologies such as Machine Learning and Digital

## 作者按：英雄所见略同。

Large-scale computing is an essential tool for solving **highly complex or data-intensive problems** involving **simulation, optimisation, data processing and artificial intelligence (AI)**.

要么是 **highly complex problems**, 要么是 **data-intensive problems** - 也就是要处理的数据是巨量的，要处理的业务逻辑是巨复杂的。但是，**巨**到什么程度，则没法具体界定。

也间接给出了三大代表领域问题：HPC (Simulation, Optimization), AI (自然是 Deep Learning 为代表的) 和 Business Computing (大数据时代完成复杂商务业务逻辑的系统)。

## 本书内容组织

---

共分为三个部分，十几个章节。

第一部分：LSC程序的样子，及当下的计算环境

### 1. 大规模计算的样子

以天气预报为切入点，并借助热传导方程的数值计算方法，助力读者类比理解天气预报计算的LSC特点，帮助读者对LCS有直观的感受。

在热传导方程的数值计算的串行程序之后，给出转换为并行计算的原理(分而治之)和套路(DAOM 和 PCAM)。

### 2. LCS计算环境 - 硬件

### 3. LSC计算环境 - 协议栈

### 4. LSC计算环境 - 大数据软件生态

第二部分：算法级LSC程序的代表性编程

### 5. MPI - MPI4PY

### 6. GPGPU和CUDA - PyCUDA

### 7. 单机多线程 - Numba

### 8. Spark - PySpark

### 第三部分：系统级LSC程序的代表性编程

9. 深度学习框架的设计与实现

10. 复杂商务系统关键技术的设计与实现

- 盈利
- 秒杀
- 计算广告

11.

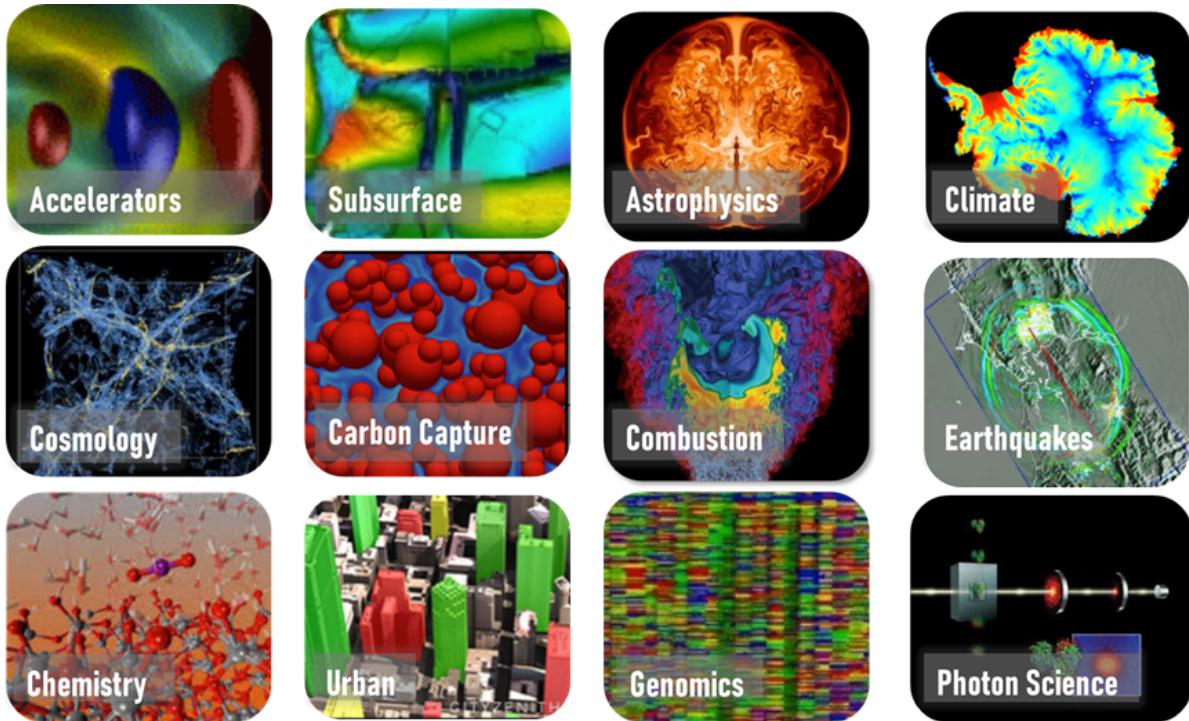
第四部分：典型LSC软件的设计与实现（以后的版本）(甚至是新的书 - 作为配套)

- Storage - HDFS
- Computing - Spark
- Job Scheduler - K8S
- Consistency - Zookeeper
- Distributed/Cloud native DMS - PolarDB/PolarDB-X (可换成不同的)
- 

---

**LSC?**

# HPC, DL, Big Data - 三者的硬件、软件基础在趋同



阿里云 2021 阿里云开发者大会

## 阿里云数据库开源计划：打造云原生分布式数据库生态

MySQL 生态

- ApsaraDB GalaxySQL
- X-Engine
- ApsaraDB GalaxySQL
- X-Engine
- ApsaraDB GalaxySQL
- X-Engine

Distributed File System

基于 X-Paxos 的 PolarDB 分布式

PG 生态

- DataNodes(PG11)
- DataNodes(PG11)

SQL

Cluster Manager

Coordinator Nodes

Coordinator Node

Tx Manager

HLC

高可用 MySQL：使用 Paxos 协议同步日志的高可用数据库  
数据正在加速云化。云原生以及分布式技术正在在底层数据库整个技术栈。  
阿里云在自身互联网业务和云数据库服务有丰富的实践经验：  
在高可用、分布式、云原生、存算分离有独到积累。  
这些技术以组件和系统的方式开发出来，并开源社区一起共建云原生分布式数据库生态。

ApsaraDB GalaxySQL：阿里巴巴 MySQL 分支  
• 借 10 年经验积累，阿里巴巴核心业务场景的考验  
• 阿里云 RDS MySQL 服务的核心基础  
• 极速、并发控制等优化技术支撑事务处理吞吐与热点处理能力  
• 安全、可靠、稳定与易于运维的支持体系

X-Engine：阿里巴巴自主研发的新型存储引擎  
• 借助 MySQL 生态的服务引擎  
• 相比传统冷热数据分离与存算分离，存储成本 2-5 倍  
• 支持共享存储，多租户云原生架构

X-Paxos：确保数据强一致的高可用协议  
• 分布式系统的数据一致性保障  
• 此协议打破与流水线发送机制提升复制数据的吞吐  
• 网络条件探测自适应应发包，跨数据中心长链路优化

PolarDB for PostgreSQL：PolarDB 基于 PostgreSQL 生态系统的云原生分布式数据库

- 基于混合逻辑时钟(HLC)的全局一致性
- 分布式事务(支持 MVCC + 2PC)
- 分布式 SQL 计算(兼容半机 SQL 功能, 分布式 DDL 和分区表支持)
- 使用 X-Paxos 跃步日志的高可用方案
- 模块化分布式 sharding 提升扩展性

LSC程序的样子 - 数据巨大，业务复杂

以天气预报/热传导方程的求解为例

<https://levelup.gitconnected.com/solving-2d-heat-equation-numerically-using-python-3334004aa01a>

## intro

When I was in college studying physics a few years ago, I remember there was a task to solve heat equation analytically for some simple problems. In the next semester we learned about numerical methods to solve some partial differential equations (PDEs) in general. It's really interesting to see how we could solve them numerically and visualize the solutions as a heat map, and it's really cool (pun intended). I also remember, in the previous semester we learned C programming language, so it was natural for us to solve PDEs numerically using C although some students were struggling with C and not with solving the PDE itself. If I had known how to code in Python back then, I would've used it instead of C (I am not saying C is bad though). Here, I am going to show how we can solve 2D heat equation numerically and see how easy it is to "translate" the equations into Python code.

Before we do the Python code, let's talk about the heat equation and finite-difference method. Heat equation is basically a partial differential equation, it is

$$\frac{\partial u}{\partial t} - \alpha \nabla u = 0$$

If we want to solve it in 2D (Cartesian), we can write the heat equation above like this

$$\frac{\partial u}{\partial t} - \alpha \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = 0$$

where  $u$  is the quantity that we want to know,  $t$  is for temporal variable,  $x$  and  $y$  are for spatial variables, and  $\alpha$  is diffusivity constant. So basically we want to find the solution  $u$  everywhere in  $x$  and  $y$ , and over time  $t$ .

## finite-difference method (FDM)

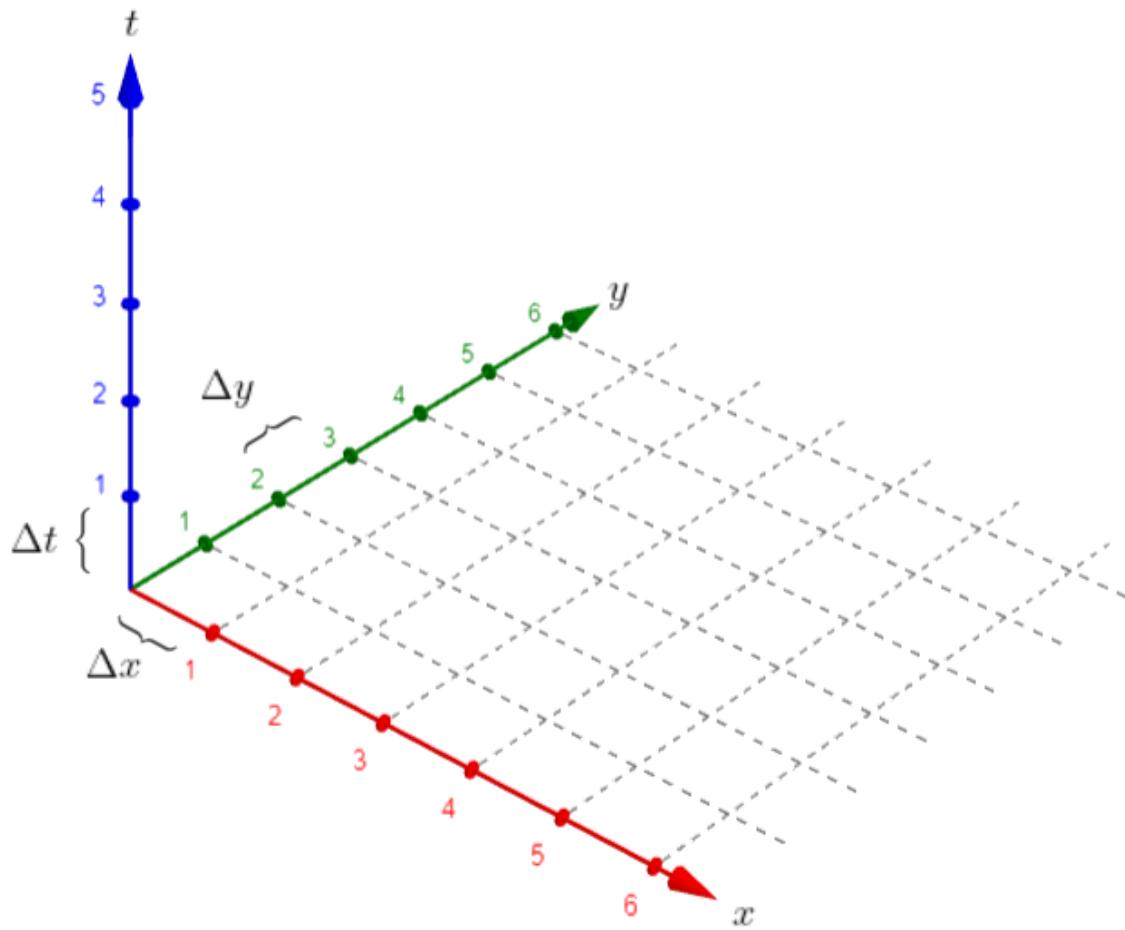
Now let's see the finite-difference method (FDM) in a nutshell. Finite-difference method is a numerical method for solving differential equations by approximating derivative with finite differences. Remember that the definition of derivative is

$$f'(a) = \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$

In finite-difference method, we approximate it and remove the limit. So, instead of using differential and limit symbol, we use delta symbol which is the finite difference. Note that this is oversimplified, because we have to use Taylor series expansion and derive it from there by assuming some terms to be sufficiently small, but we get the rough idea behind this method.

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}$$

In finite-difference method, we are going to "discretize" the spatial domain and the time interval  $x, y$ , and  $t$ . We can write it like this



Cartesian coordinate, where x and y axis are for spatial variables, and t for temporal variable (coordinate axes from [GeoGebra](#), edited by author)

$$x_i = i\Delta x$$

$$y_j = j\Delta y$$

$$t_k = k\Delta t$$

As we can see,  $i, j$ , and  $k$  are the steps for each difference for  $x, y$ , and  $t$  respectively. What we want is the solution  $u$ , which is

$$u(x, y, t) = u_{i,j}^k$$

Note that  $k$  is superscript to denote time step for  $u$ . We can write the heat equation above using finite-difference method like this

$$\frac{u_{i,j}^{k+1} - u_{i,j}^k}{\Delta t} - \alpha \left( \frac{u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k}{\Delta x^2} + \frac{u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k}{\Delta y^2} \right) = 0$$

If we arrange the equation above by taking  $\Delta x = \Delta y$ , we get this final equation

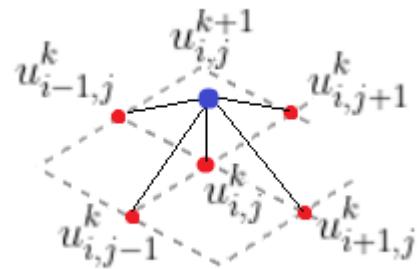
$$u_{i,j}^{k+1} = \gamma(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k - 4u_{i,j}^k) + u_{i,j}^k$$

where

$$\gamma = \alpha \frac{\Delta t}{\Delta x^2}$$

## Explicit method stencil

We can use this stencil to remember the equation above (look at subscripts  $i, j$  for spatial steps and superscript  $k$  for the time step)

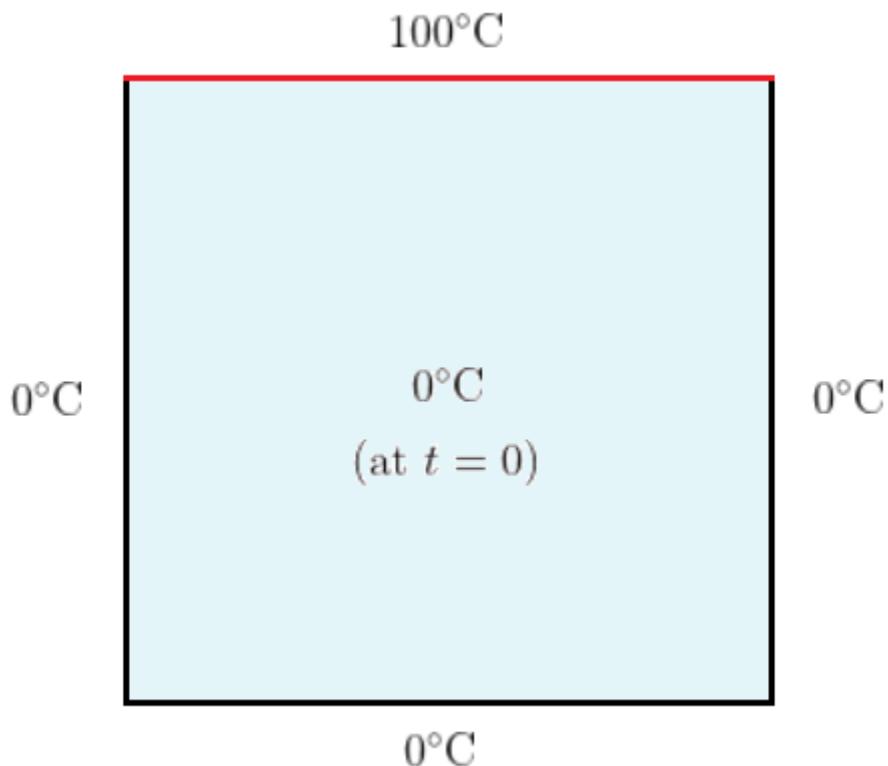


Explicit method stencil (image by author)

We use explicit method to get the solution for the heat equation, so it will be numerically stable whenever

$$\Delta t \leq \frac{\Delta x^2}{4\alpha}$$

Everything is ready. Now we can solve the original heat equation approximated by algebraic equation above, which is computer-friendly. For an exercise problem, let's suppose a thin square plate with the side of 50 unit length. The temperature everywhere inside the plate is originally 0 degree (at  $t = 0$ ), let's see the diagram below (this is not realistic, but it's good for exercise)



Boundary and initial conditions for our exercise (image by author)

<https://www.mmm.ucar.edu/models/wrf>

## Weather Research & Forecasting Model (WRF)

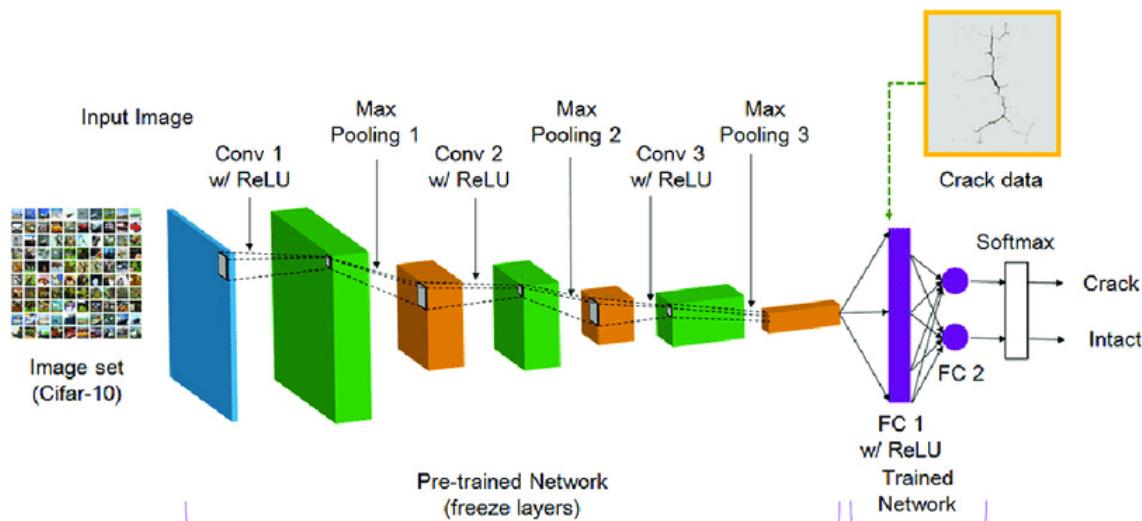
The screenshot shows the homepage of the NCAR WRF website. At the top, the NCAR logo and the text "MESOSCALE & MICROSCALE METEOROLOGY LABORATORY" are visible. The navigation menu includes links for HOME, ABOUT, WHAT WE DO, MODELS (which is underlined), SECTIONS, EVENTS, NEWS, and FOR STAFF. On the right side, there are buttons for "Contact Us" and "Search". A search bar with a magnifying glass icon is also present. The main content area features a map of the US with various weather patterns and data overlays. To the right of the map, there are two dark grey boxes: one labeled "Events" and another labeled "Forecasts". Below the map, there's a yellow box labeled "RESOURCE LINK" containing links to "WRF ARW User Page", "WRFDA User Page", and "WRF & MPAS-A Support Forum". At the bottom left, there's a small circular logo, and at the bottom right, the number "218".

### Global Grid ( $0.25 \times 0.25 \rightarrow 1440 \times 720 \text{ dots} = 62208000$ )

- If we take resolution as  $0.25 \times 0.25$ , and 60 vertical layers, the matrix size will be  $1440 \times 720 \times 60 = 6.22 \times 10^7$ .
- If 6 variables, the number is  $6 \times 6.22 \times 10^7$ .
- If we want to predict the weather in 24 hours, and 5 minutes as span, we should compute values of
  - $288 \times 6 \times 6.22 \times 10^7 \approx 1.08 \times 10^{11}$ .
  - If you want to predict future 3 days, 5 days, 10 days?
- So **HUGE!** We need more powerful computers!
  - Super Computers, HPC etc.

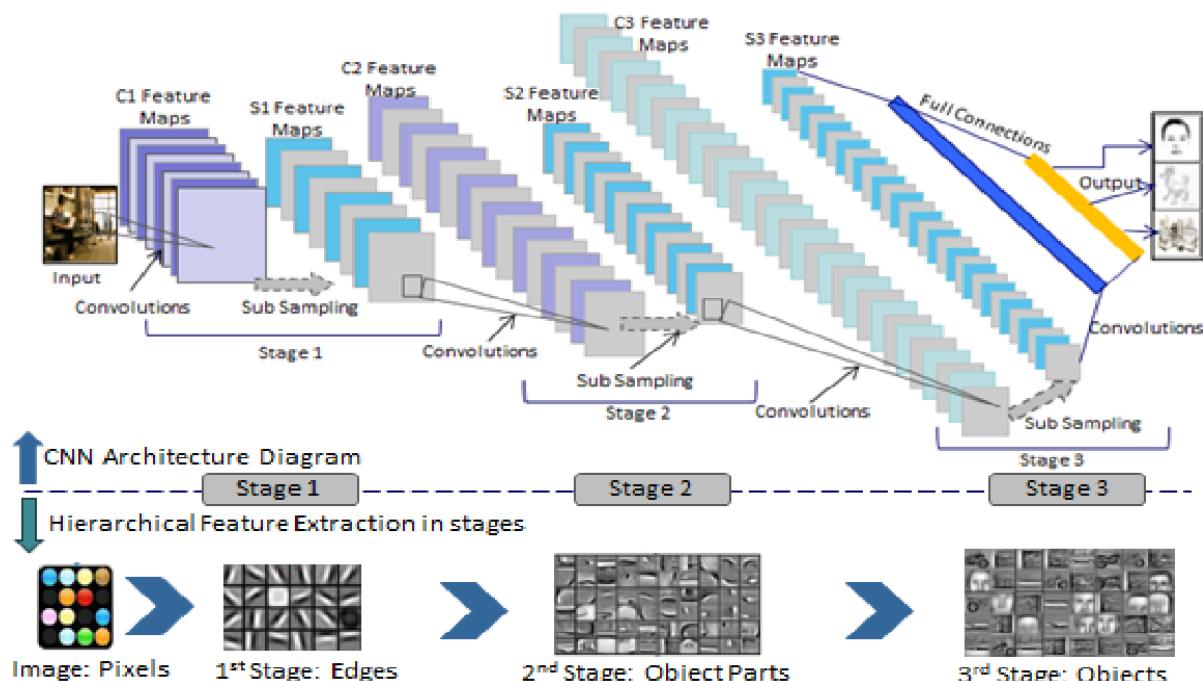
The screenshot shows a browser window with the URL "https://www.esri.noaa.gov/psd/" in the address bar. The main content area of the page is completely blank, showing only a few horizontal lines where text or images would normally appear.

## 深度学习 - CNN、ChatGPT



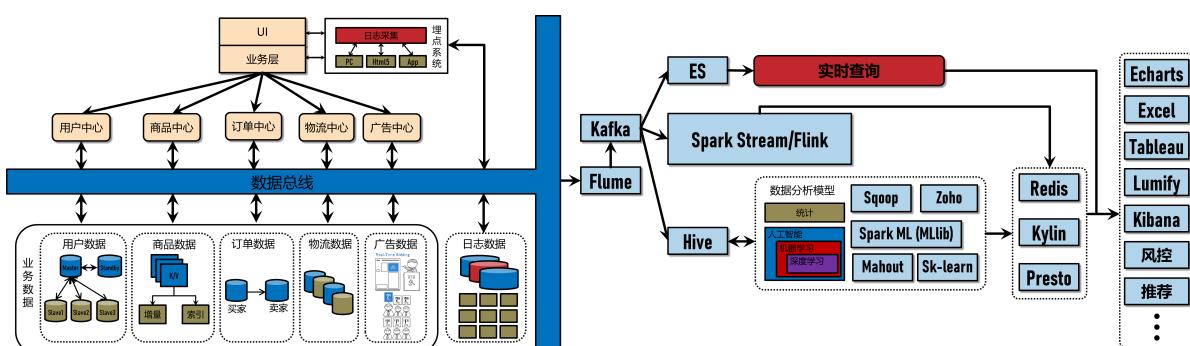
[https://www.researchgate.net/figure/Schematic-of-the-deep-learning-architecture\\_fig4\\_32566](https://www.researchgate.net/figure/Schematic-of-the-deep-learning-architecture_fig4_32566)

3483



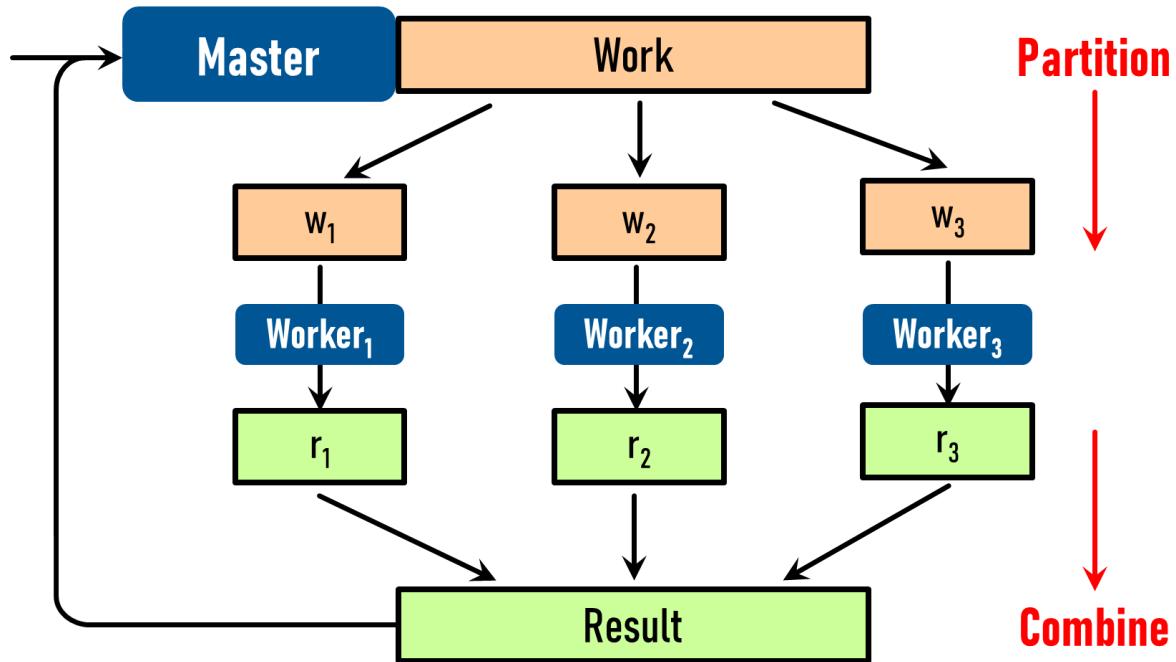
<https://d3i71xaburhd42.cloudfront.net/f78e280123b1c0c68f84da3cc6c66615f6e7cebd/3-Figure1-1.png>

## 商务系统 - 阿里淘宝，Google, 滴滴，高德地图等

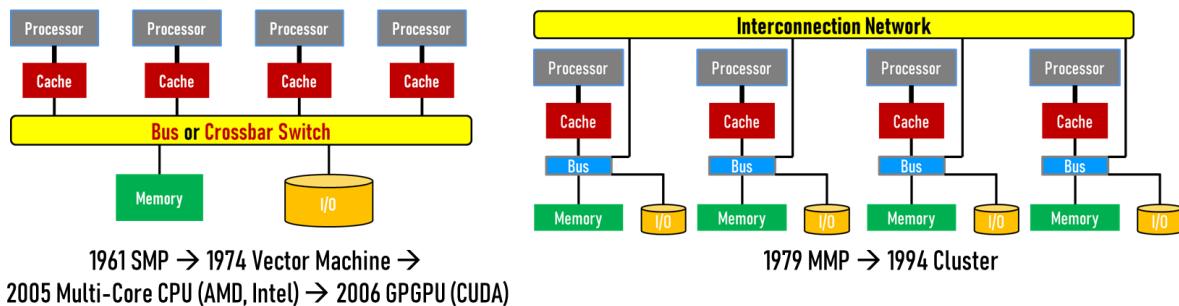


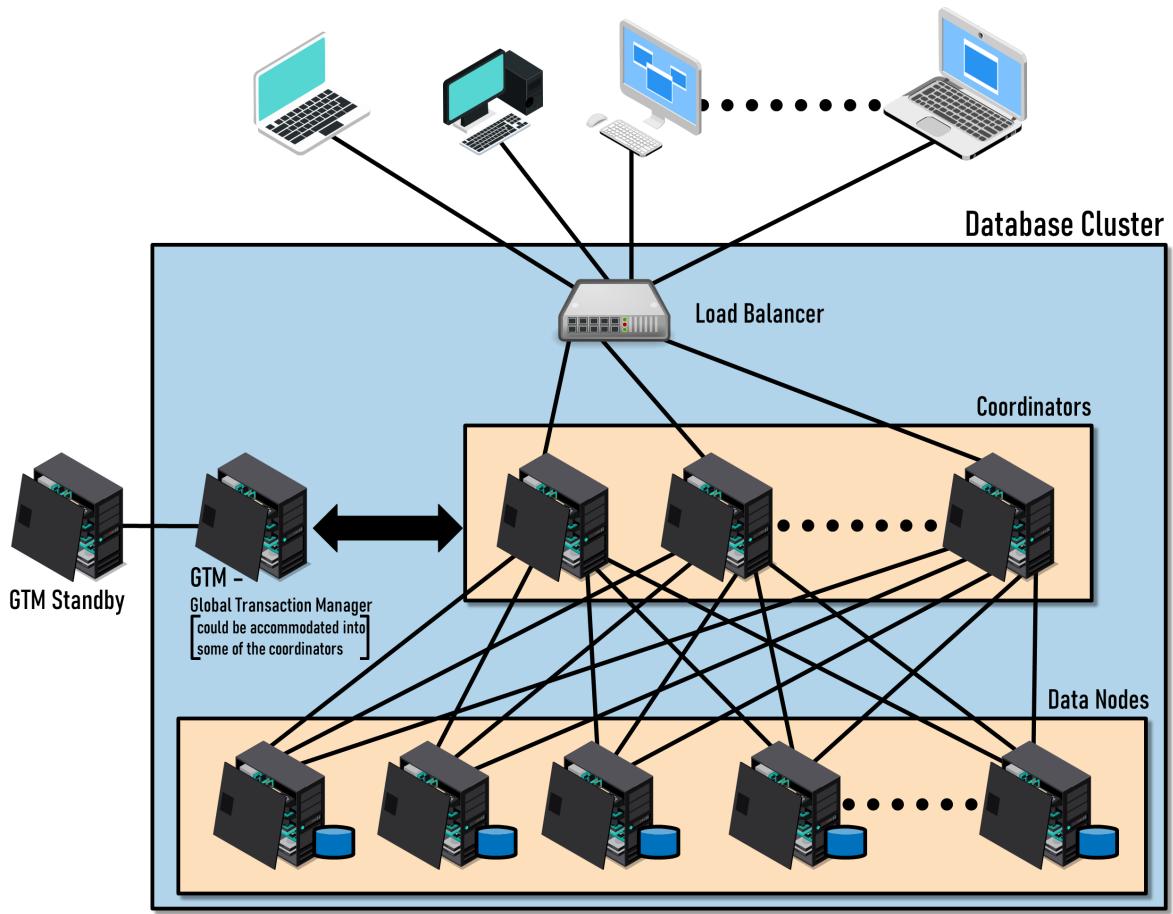
# 编写 LSC 程序的套路

问题的分解: Divide & Conquer



运行的环境: 多 - 核, 节点

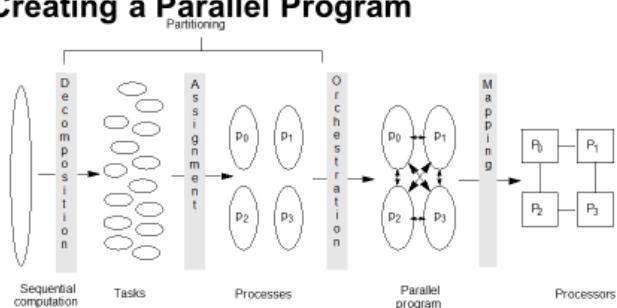




## DAOM/PCAM

K. Mani Chandy, Stephen Taylor. An Introduction to Parallel Programming. Jones and Bartlett. Publishers, Inc., Burlington. **1991**.

### □ 4 Steps in Creating a Parallel Program



- **Decomposition** of computation in tasks
- **Assignment** of tasks to processes
- **Orchestration** of data access, comm, synch.
- **Mapping** processes to processors





## 任务划分(Partitioning)

- 将整个计算分解为一些小任务，其目的是尽量开拓并行执行的机会

## 通信(Communication)

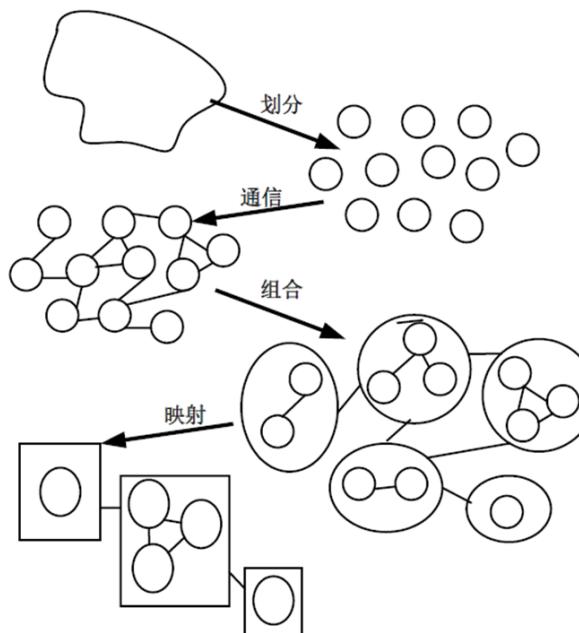
- 确定诸任务执行中所需要交换的数据和协调诸任务的执行，由此检测上述划分的合理性

## 任务组合(Agglomeration)

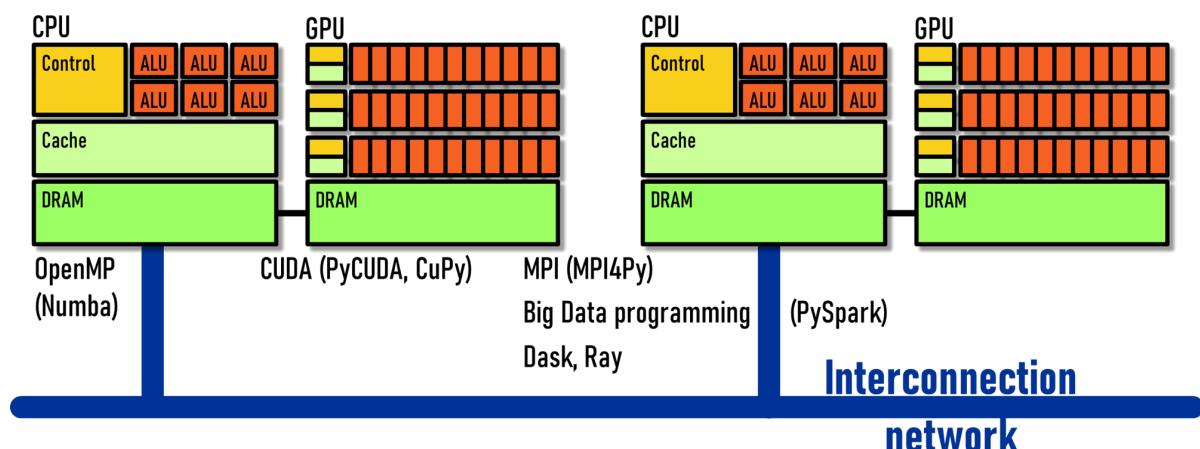
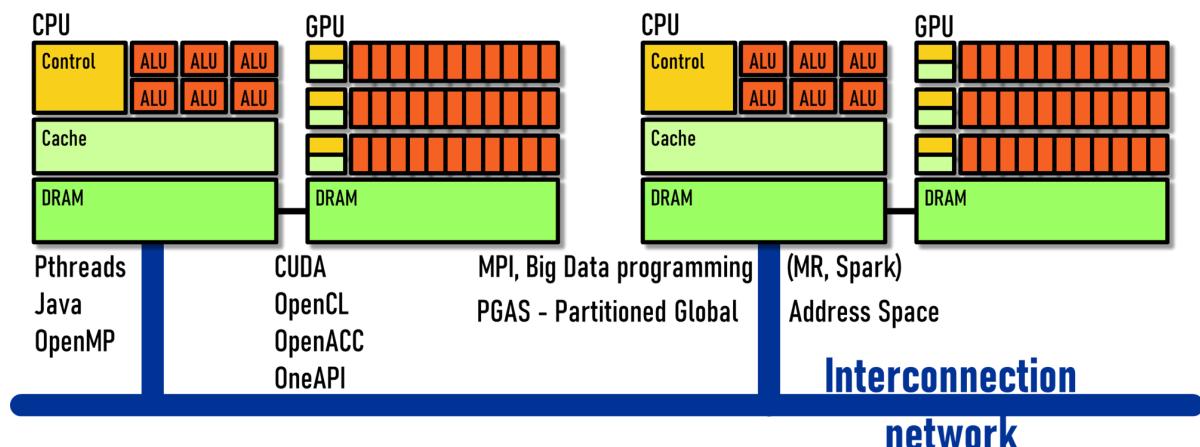
- 按性能要求和实现的代价来考察前两阶段的结果，必要时可将一些小任务组合成更大的任务以提高性能和减少通信开销

## 处理器映射(Mapping)

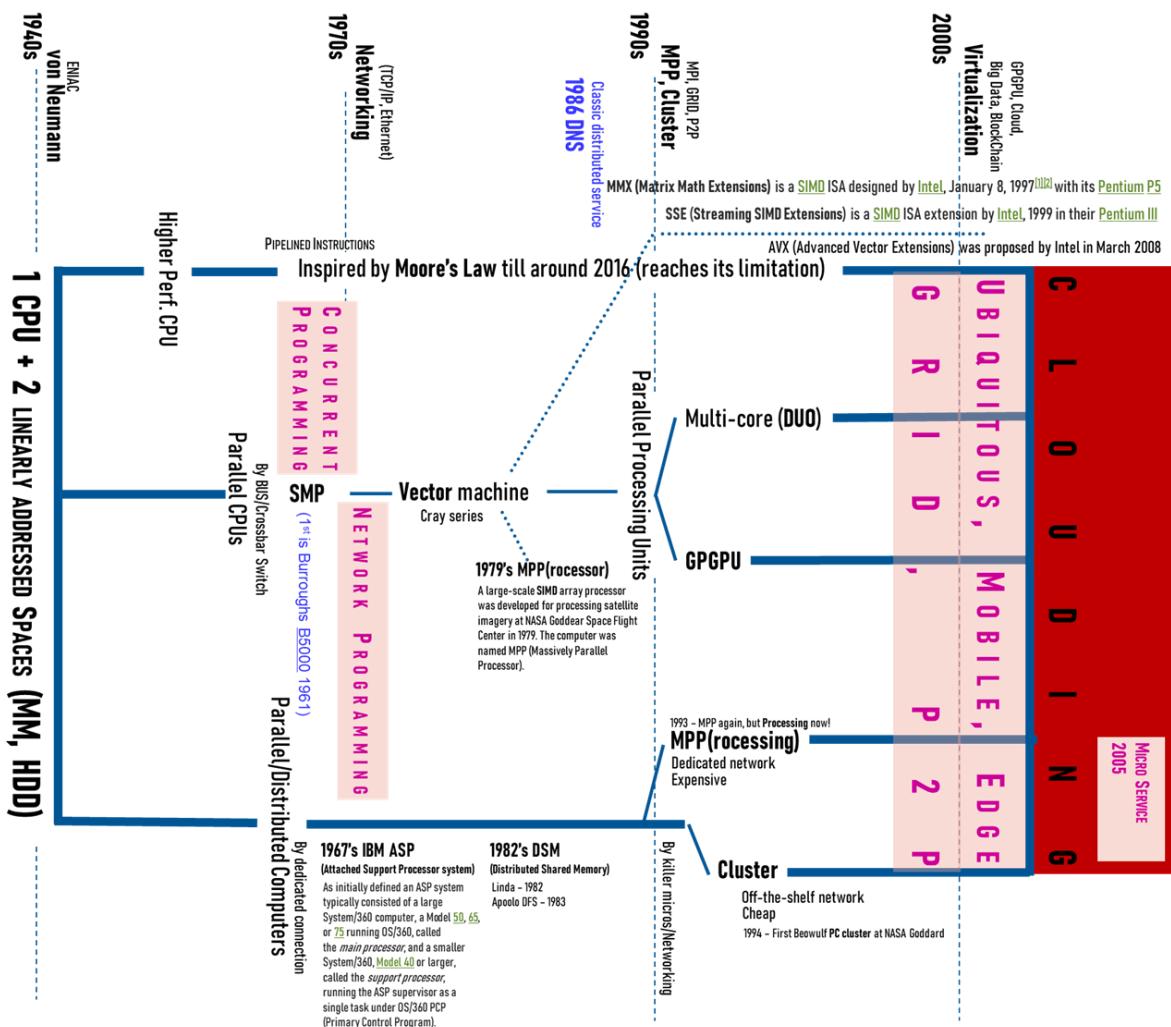
- 将每个任务分配到一个处理器上，其目的是最小化全局执行时间和通信成本以及最大化处理器的利用率



## 编程框架



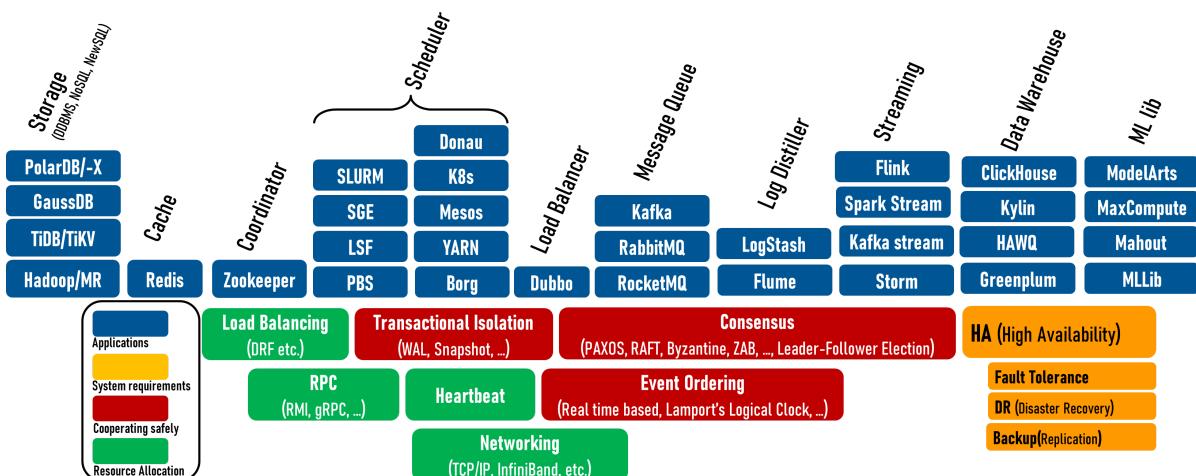
# 硬件环境

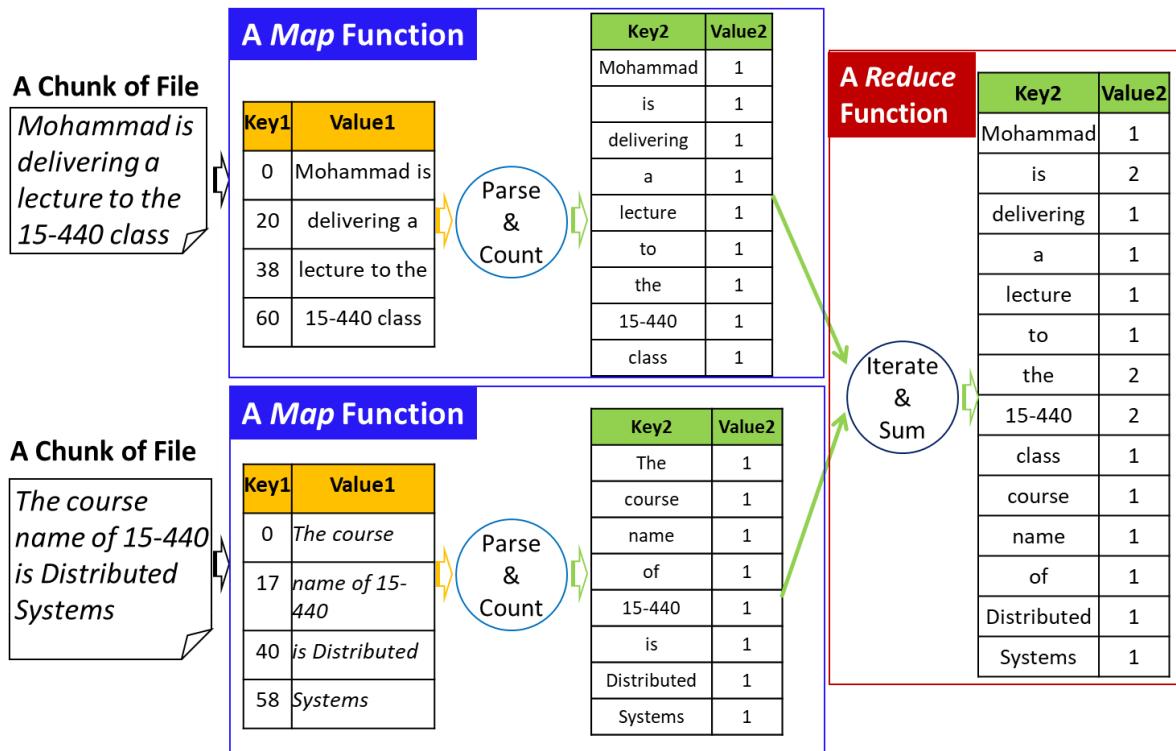


# 软件环境 - 协议栈



# 软件环境 - 大数据 - 以 Hadoop 为例





## 代码 - 串行

### 热传导方程的数值求解 - 概念

#### [Images] heatEqu2DSeqImg.py

"D:\myCodes\HPCbook\00heatEqu\heatEqu2DSeqImg.py"

```

1  """
2   https://github.com/csc-training/hpc-python/blob/master/mpi/heat-
3   equation/solution/heat-p2p.py
4
5   "D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-
6   heatequ2Dk.py"
7   """
8
9
10
11
12 from __future__ import print_function
13 import numpy as np
14 import timeit
15
16
17 import matplotlib
18
19 matplotlib.use('Agg')
20 import matplotlib.pyplot as plt
21
22 # Set the colormap
23 # plt.rcParams['image.cmap'] = 'BrBG'

```

```

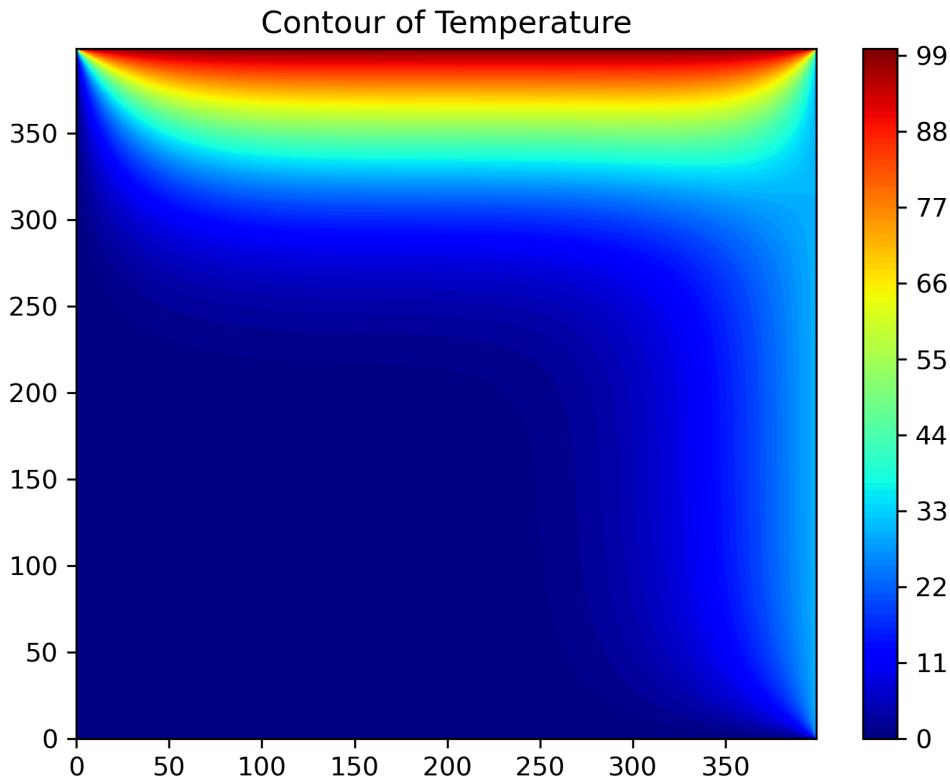
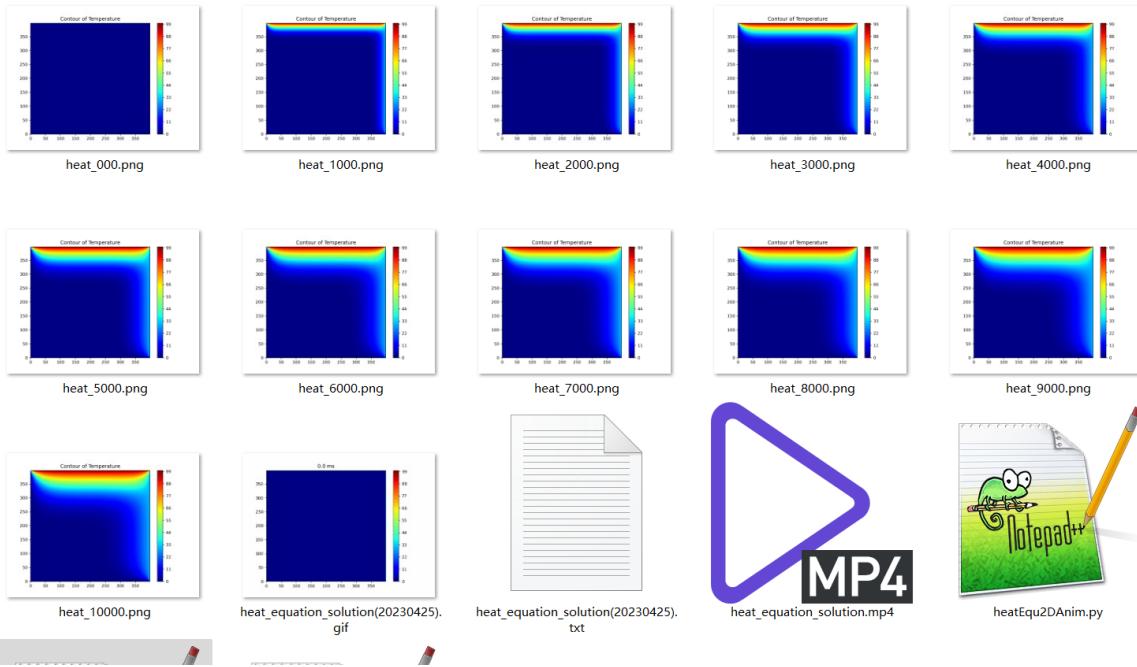
19 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
20 plt.cm.coolwarm
21 plt.figure(dpi=300)
22
23 # Set colour interpolation and colour map
24 colorinterpolation = 100
25 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
26
27 # Set Dimension
28 lenX = lenY = 400 #we set it rectangular
29
30 # Set meshgrid
31 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
32
33 # Basic parameters
34 a = 0.1 # Diffusion constant
35 timesteps = 10000 # Number of time-steps to evolve system
36 image_interval = 1000 # write frequency for png files
37
38 # Grid spacings
39 dx = 0.01
40 dy = 0.01
41 dx2 = dx ** 2
42 dy2 = dy ** 2
43
44 # For stability, this is the largest interval possible
45 # for the size of the time-step:
46 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
47
48 # Boundary condition
49 Ttop = 100
50 Tbottom = 0
51 Tleft = 0
52 Tright = 30
53
54 # Initial guess of interior grid
55 Tguess = 0
56
57 def init_fields():
58     # Set array size and set the interior value with Tguess
59     field = np.empty((lenX, lenY))
60     field.fill(Tguess)
61
62     # Set Boundary condition
63     field[(lenY-1):, :] = Ttop
64     field[:, :] = Tbottom
65     field[:, (lenX-1):] = Tright
66     field[:, :1] = Tleft
67
68     print("size is ",field.size)
69     print(field,"\n")
70
71     field0 = field.copy() # Array for field of previous time step
72     return field, field0
73

```

```

74
75 def evolve(u, u_previous, a, dt, dx2, dy2):
76     """Explicit time evolution.
77         u:           new temperature field
78         u_previous: previous field
79         a:           diffusion constant
80         dt:          time step
81         dx2:         grid spacing squared, i.e. dx^2
82         dy2:         -- "" -- , i.e. dy^2"""
83     u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
84         (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
85         u_previous[:-2, 1:-1]) / dx2 +
86         (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
87         u_previous[1:-1, :-2]) / dy2)
88     u_previous[:] = u[:]
89
90 def write_field(field, step):
91     # plt.gca().clear()
92     plt.cla()
93     plt.clf()
94
95     # Configure the contour
96     plt.title("Contour of Temperature")
97     plt.contourf(x, Y, field, colorinterpolation, cmap=colourMap)
98     # Set Colorbar
99     plt.colorbar()
100    plt.axis('on')
101    plt.savefig('heat_Seq_{0:03d}.png'.format(step))
102
103 def main():
104     field, field0 = init_fields()
105     write_field(field, 0)
106
107     for m in range(1, timesteps + 1):
108         evolve(field, field0, a, dt, dx2, dy2)
109         if m % image_interval == 0:
110             write_field(field, m)
111
112 if __name__ == '__main__':
113     main()
114

```



## [Animation]

```

1  .....
2  https://scipython.com/book2/chapter-7-matplotlib/problems/p77/animation-of-
   the-diffusion-equation/
3  "D:\myCodes\HPCprojects\SourceCodes\Learning Scientific Programming with
   Python(Christian Hill)\Anim2DHeatDiff.py"

```

```

4 "D:\myCodes\HPCprojects\SourceCodes\Learning Scientific Programming with
5 Python(Christian Hill)\Animation of the diffusion equation.md"
6 """
7
8 import numpy as np
9 import matplotlib.pyplot as plt
10 import matplotlib.animation as animation
11 from PIL import Image
12
13 SAVE_ANIMATION = True # False True
14
15 # Set the colormap
16 # plt.rcParams['image.cmap'] = 'BrBG'
17 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
18 plt.cm.coolwarm
19 # plt.figure(dpi=300)
20
21 # set colour interpolation and colour map
22 colorinterpolation = 100
23 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
24
25 # Basic parameters
26 a = 0.1 # Diffusion constant #例如 Thermal diffusivity of steel, mm2.s-1
27 timesteps = 100000 # Number of time-steps to evolve system
28 image_interval = 1000 # Write frequency for png files
29
30 # Grid spacings
31 dx = 0.01
32 dy = 0.01
33 dx2 = dx ** 2
34 dy2 = dy ** 2
35
36 # For stability, this is the largest interval possible
37 # for the size of the time-step:
38 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
39
40 # Set Dimension
41 lenX = lenY = 4000 #we set it rectangular
42
43 # Set meshgrid
44 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
45
46 # Boundary condition
47 Ttop = 100
48 Tbottom = 0
49 Tleft = 0
50 Tright = 30
51
52 # Initial guess of interior grid
53 Tguess = 30
54
55 def init_fields():
56     # Set array size and set the interior value with Tguess
57     field = np.empty((lenX, lenY))

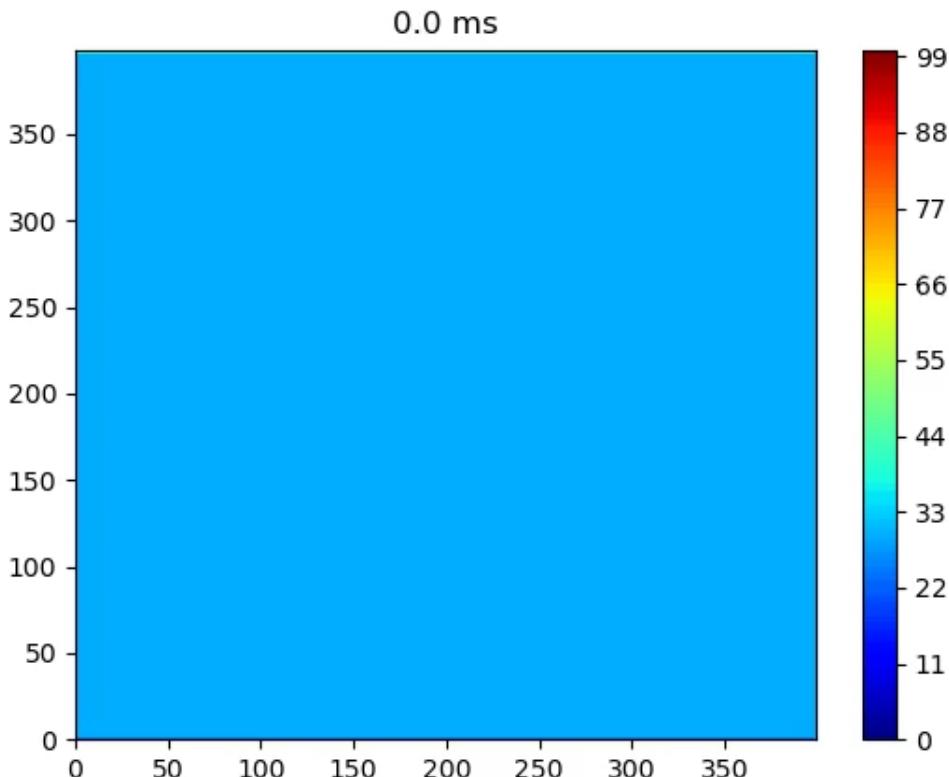
```

```

58     field.fill(Tguess)
59
60     # Set Boundary condition
61     field[(lenY-1) :, :] = Ttop
62     field[:, 1] = Tbottom
63     field[:, (lenX-1) :] = Tright
64     field[:, :1] = Tleft
65
66     print("size is ", field.size)
67     print(field, "\n")
68
69     field0 = field.copy() # Array for field of previous time step
70     return field, field0
71
72 u, u0 = init_fields()
73
74 def evolve(u, u0, a, dt, dx2, dy2):
75     # Propagate with forward-difference in time, central-difference in
76     # space
77     u[1:-1, 1:-1] = u0[1:-1, 1:-1] + a * dt * (
78         (u0[2:, 1:-1] - 2 * u0[1:-1, 1:-1] +
79         u0[:-2, 1:-1]) / dx2 +
80         (u0[1:-1, 2:] - 2 * u0[1:-1, 1:-1] +
81         u0[1:-1, :-2]) / dy2)
82     u0[:] = u[:]
83
84     # u0 = u.copy()
85     return u, u0
86
87 fig, ax = plt.subplots()
88 # Configure the contour
89 plt.title("Contour of Temperature")
90 plt.contourf(x, Y, u0, colorinterpolation, cmap=colourMap)
91 # Set Colorbar
92 plt.colorbar()
93 plt.axis('on')
94
95 def animate(i):
96     """Set the data for the ith iteration of the animation."""
97
98     global u0, u, a, dt, dx2, dy2
99
100    # plt.gca().clear()
101    plt.cla()
102    plt.clf()
103    plt.title('{:.1f} ms'.format(i*dt*1000))
104    plt.contourf(x, Y, u0, colorinterpolation, cmap=colourMap)
105    # Set Colorbar
106    plt.colorbar()
107    plt.axis('on')
108
109    u, u0 = evolve(u, u0, a, dt, dx2, dy2)
110
111    return u0, u
112
113 if SAVE_ANIMATION:

```

```
113     anim = animation.FuncAnimation(fig, animate, frames=timesteps,
114         repeat=False, interval = 50)
115         # anim.save('us.gif', writer='imagemagick', fps=5)
116         # anim.save("heat_equation_solution.gif", writer='imagemagick', fps=10)
117         anim.save("heat_equation_solution.mp4", writer='ffmpeg', fps=10)
118 else:
119     anim = animation.FuncAnimation(fig, animate, frames=timesteps)
120 plt.show()
121
```



## [Show]

```
1 """
2 https://github.com/csc-training/hpc-python/blob/master/mpi/heat-
3 equation/solution/heat-p2p.py
4
5 "D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-
6 heatequ2Dk.py"
7 """
8
9 from __future__ import print_function
10 import numpy as np
11 import timeit
```

```

11 import matplotlib
12 import matplotlib.pyplot as plt
13
14 # Set the colormap
15 # plt.rcParams['image.cmap'] = 'BrBG'
16 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
17 plt.cm.coolwarm
18 # plt.figure(dpi=300)
19
20 # Set colour interpolation and colour map
21 colorinterpolation = 100
22 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
23
24 # Basic parameters
25 a = 0.1 # Diffusion constant #例如 Thermal diffusivity of steel, mm2.s-1
26 timesteps = 10000 # Number of time-steps to evolve system
27
28 # Grid spacings
29 dx = 0.01
30 dy = 0.01
31 dx2 = dx ** 2
32 dy2 = dy ** 2
33
34 # For stability, this is the largest interval possible
35 # for the size of the time-step:
36 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
37
38 # Set Dimension
39 lenX = lenY = 400 #we set it rectangular
40
41 # Set meshgrid
42 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
43
44 # Boundary condition
45 Ttop = 100
46 Tbottom = 0
47 Tleft = 0
48 Tright = 30
49
50 # Initial guess of interior grid
51 Tguess = 0
52
53 def init_fields():
54     # Set array size and set the interior value with Tguess
55     field = np.empty((lenX, lenY))
56     field.fill(Tguess)
57
58     # Set Boundary condition
59     field[(lenY-1):, :] = Ttop
60     field[:, 1] = Tbottom
61     field[:, (lenX-1):] = Tright
62     field[:, :1] = Tleft
63
64     print("size is ", field.size)
65     print(field, "\n")

```

```

66     field0 = field.copy() # Array for field of previous time step
67     return field, field0
68
69 def evolve(u, u_previous, a, dt, dx2, dy2):
70     """Explicit time evolution.
71         u:              new temperature field
72         u_previous:    previous field
73         a:              diffusion constant
74         dt:             time step
75         dx2:            grid spacing squared, i.e. dx^2
76         dy2:            -- " " -- , i.e. dy^2"""
77
78     u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
79         (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
80         u_previous[:-2, 1:-1]) / dx2 +
81         (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
82         u_previous[1:-1, :-2]) / dy2)
83     u_previous[:] = u[:]
84
85
86 def main():
87     field, field0 = init_fields()
88
89     starting_time = timeit.default_timer()
90     for m in range(1, timesteps + 1):
91         evolve(field, field0, a, dt, dx2, dy2)
92
93     # print("Iteration finished")
94     print("Iteration finished. {} Seconds for Time"
95           "difference:".format(timeit.default_timer() - starting_time))
96
97     # Configure the contour
98     plt.title("Contour of Temperature")
99     plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
100    # Set Colorbar
101    plt.colorbar()
102    plt.axis('on')
103    # Show the result in the plot window
104    plt.show()
105
106 if __name__ == '__main__':
107     main()

```

# 算法级编程

## MPI4PY

# MPI - 概念

## MPI4PY 编程基础

### [Images] Stripe

#### 线索

<https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2.py>

The screenshot shows the GitHub repository page for `hpc-training/hpc-python`. The `heat-equation` branch is selected. The commit history for this branch is displayed, showing the following commits:

- jussienko and guest075 Use Agg backend when not showing figures on screen (5 years ago)
- Minor changes to the text (5 years ago)
- Major reorganisation of file hierarchy (6 years ago)
- Major reorganisation of file hierarchy (6 years ago)
- Major reorganisation of file hierarchy (6 years ago)
- Small edit (5 years ago)
- Fix deprecation warning of matplotlib's pyplot.hold() (5 years ago)

1 | "D:\Local\++写书\18 高性能计算\HPCBook-MD\materials\hpc-python-master.zip"

The screenshot shows the GitHub organization page for `CSC Training`. The organization's profile picture is a stylized 'CSC' logo. The page includes the following sections:

- Popular repositories:**
  - `hpc-python`: Python in High Performance Computing (Python, 260 stars, 796 forks)
  - `CUDA`: Introduction to CUDA programming (Cuda, 97 stars, 25 forks)
  - `summerschool`: CSC Summer School in High Performance Computing (Fortran, 48 stars, 123 forks)
  - `intro-to-dl`: Introduction to deep learning (Python, 39 stars, 43 forks)
  - `geocomputing`: Examples for doing spatial analysis in CSC computing environment (Jupyter Notebook, 29 stars, 21 forks)
  - `openacc`: Introduction to OpenACC (C, 25 stars, 12 forks)
- People:** Shows four user profiles.
- Top languages:** Shows the most used languages: Jupyter Notebook, Fortran, HTML, C, and Python.
- Most used topics:** Shows topics: data-analysis, exercises, r, data-visualization, and statistics.

1 |

2 | <https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2p.py>

```

3  """
4
5  from __future__ import print_function
6  import numpy as np
7  import time
8  from mpi4py import MPI
9
10 import matplotlib
11
12 matplotlib.use('Agg')
13 import matplotlib.pyplot as plt
14
15 # Set the colormap
16 plt.rcParams['image.cmap'] = 'BrBG'
17
18 # Basic parameters
19 a = 0.5 # Diffusion constant
20 timesteps = 600 # Number of time-steps to evolve system
21 image_interval = 40000 # Write frequency for png files
22
23 # Grid spacings
24 dx = 0.01
25 dy = 0.01
26 dx2 = dx ** 2
27 dy2 = dy ** 2
28
29 # For stability, this is the largest interval possible
30 # for the size of the time-step:
31 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
32
33 # MPI globals
34 comm = MPI.COMM_WORLD
35 rank = comm.Get_rank()
36 size = comm.Get_size()
37
38 # Up/down neighbouring MPI ranks
39 up = rank - 1
40 if up < 0:
41     up = MPI.PROC_NULL
42 down = rank + 1
43 if down > size - 1:
44     down = MPI.PROC_NULL
45
46 def evolve(u, u_previous, a, dt, dx2, dy2):
47     """Explicit time evolution.
48         u:          new temperature field
49         u_previous: previous field
50         a:          diffusion constant
51         dt:          time step
52         dx2:         grid spacing squared, i.e. dx^2
53         dy2:         -- " " -- , i.e. dy^2"""
54
55     u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
56             (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
57             u_previous[:-2, 1:-1]) / dx2 +
58             (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +

```

```

59             u_previous[1:-1, :-2]) / dy2)
60     u_previous[:] = u[:]
61
62
63 def init_fields(filename):
64     # Read the initial temperature field from file
65     field = np.loadtxt(filename)
66     field0 = field.copy() # Array for field of previous time step
67     return field, field0
68
69
70 def write_field(field, step):
71     plt.gca().clear()
72     plt.imshow(field)
73     plt.axis('off')
74     plt.savefig('heat_{0:03d}.png'.format(step))
75
76
77 def exchange(field):
78     # send down, receive from up
79     sbuf = field[-2, :]
80     rbuf = field[0, :]
81     comm.Sendrecv(sbuf, dest=down, recvbuf=rbuf, source=up)
82     # send up, receive from down
83     sbuf = field[1, :]
84     rbuf = field[-1, :]
85     comm.Sendrecv(sbuf, dest=up, recvbuf=rbuf, source=down)
86
87
88 def iterate(field, local_field, local_field0, timesteps, image_interval):
89     for m in range(1, timesteps + 1):
90         exchange(local_field0)
91         evolve(local_field, local_field0, a, dt, dx2, dy2)
92         if m % image_interval == 0:
93             comm.Gather(local_field[1:-1, :], field, root=0)
94             if rank == 0:
95                 write_field(field, m)
96
97
98 def main():
99     # Read and scatter the initial temperature field
100    if rank == 0:
101        field, field0 = init_fields('bottle_large.dat')
102        shape = field.shape
103        dtype = field.dtype
104        comm.bcast(shape, 0) # broadcast dimensions
105        comm.bcast(dtype, 0) # broadcast data type
106    else:
107        field = None
108        shape = comm.bcast(None, 0)
109        dtype = comm.bcast(None, 0)
110    if shape[0] % size:
111        raise ValueError('Number of rows in the temperature field (' \
112                           + str(shape[0]) + ') needs to be divisible by the \
113                           number ' \
114                           + 'of MPI tasks (' + str(size) + ').')

```

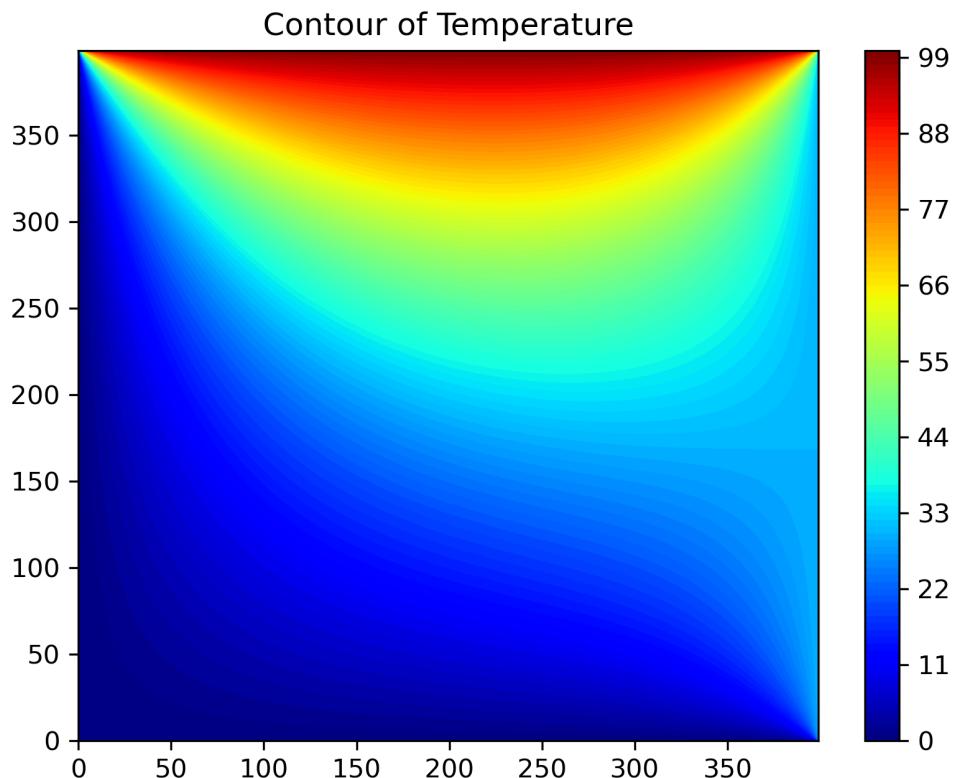
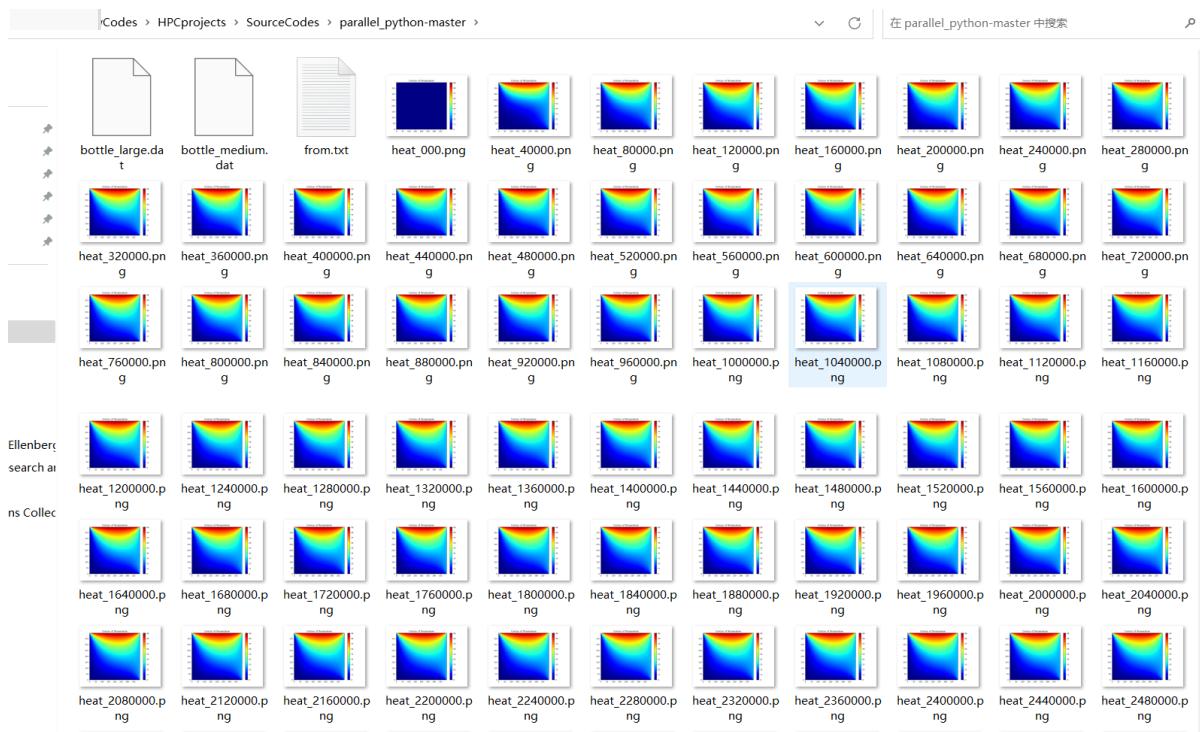
```

114
115     n = int(shape[0] / size) # number of rows for each MPI task
116     m = shape[1] # number of columns in the field
117     buff = np.zeros((n, m), dtype)
118     comm.Scatter(field, buff, 0) # scatter the data
119     local_field = np.zeros((n + 2, m), dtype) # need two ghost rows!
120     local_field[1:-1, :] = buff # copy data to non-ghost rows
121     local_field0 = np.zeros_like(local_field) # array for previous time
122     step
123
124     # Fix outer boundary ghost layers to account for aperiodicity?
125     if True:
126         if rank == 0:
127             local_field[0, :] = local_field[1, :]
128         if rank == size - 1:
129             local_field[-1, :] = local_field[-2, :]
130     local_field0[:] = local_field[:]

131     # Plot/save initial field
132     if rank == 0:
133         write_field(field, 0)
134
135     # Iterate
136     t0 = time.time()
137     iterate(field, local_field, local_field0, timesteps, image_interval)
138     t1 = time.time()
139
140     # Plot/save final field
141     comm.Gather(local_field[1:-1, :], field, root=0)
142     if rank == 0:
143         write_field(field, timesteps)
144         print("Running time: {0}".format(t1 - t0))
145
146
147 if __name__ == '__main__':
148     main()
149

```

## 我的代码的效果



## [我的代码]

```

1 """
2 https://github.com/csc-training/hpc-python/blob/master/mpi/heatequation/solution/heateq-p2p.py
3
4 "D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-heatequ2dk.py"
5 """

```

```
6
7 from __future__ import print_function
8 import numpy as np
9 import timeit
10
11
12 import matplotlib
13
14 matplotlib.use('Agg')
15 import matplotlib.pyplot as plt
16
17 # Set the colormap
18 # plt.rcParams['image.cmap'] = 'BrBG'
19 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
20 plt.cm.coolwarm
21 plt.figure(dpi=300)
22
23 # Set colour interpolation and colour map
24 colorinterpolation = 100
25 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
26
27 # Set Dimension
28 lenX = lenY = 400 #we set it rectangular
29
30 # Set meshgrid
31 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
32
33 # Basic parameters
34 a = 0.1 # Diffusion constant
35 timesteps = 10000 # Number of time-steps to evolve system
36 image_interval = 1000 # write frequency for png files
37
38 # Grid spacings
39 dx = 0.01
40 dy = 0.01
41 dx2 = dx ** 2
42 dy2 = dy ** 2
43
44 # For stability, this is the largest interval possible
45 # for the size of the time-step:
46 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
47
48
49 # Boundary condition
50 Ttop = 100
51 Tbottom = 0
52 Tleft = 0
53 Tright = 30
54
55 # Initial guess of interior grid
56 Tguess = 0
57
58 def init_fields():
59     # Set array size and set the interior value with Tguess
60     field = np.empty((lenX, lenY))
```

```

61     field.fill(Tguess)
62
63     # Set Boundary condition
64     field[(lenY-1):, :] = Ttop
65     field[:, 1] = Tbottom
66     field[:, (lenX-1):] = Tright
67     field[:, :1] = Tleft
68
69     print("size is ", field.size)
70     print(field, "\n")
71
72     field0 = field.copy() # Array for field of previous time step
73     return field, field0
74
75 def evolve(u, u_previous, a, dt, dx2, dy2):
76     """Explicit time evolution.
77         u:           new temperature field
78         u_previous: previous field
79         a:           diffusion constant
80         dt:          time step
81         dx2:         grid spacing squared, i.e. dx^2
82         dy2:         -- " " -- , i.e. dy^2"""
83
84     u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
85         (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
86         u_previous[:-2, 1:-1]) / dx2 +
87         (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
88         u_previous[1:-1, :-2]) / dy2)
89     u_previous[:] = u[:]
90
91 def write_field(field, step):
92     # plt.gca().clear()
93     plt.cla()
94     plt.clf()
95
96     # Configure the contour
97     plt.title("Contour of Temperature")
98     plt.contourf(x, Y, field, colorinterpolation, cmap=colourMap)
99     # Set Colorbar
100    plt.colorbar()
101    plt.axis('on')
102    plt.savefig('heat_seq_{0:03d}.png'.format(step))
103
104 def main():
105     field, field0 = init_fields()
106     write_field(field, 0)
107
108     for m in range(1, timesteps + 1):
109         evolve(field, field0, a, dt, dx2, dy2)
110         if m % image_interval == 0:
111             write_field(field, m)
112
113 if __name__ == '__main__':
114     main()
115

```

## [Images] Stencil

### 线索

<https://repository.prace-ri.eu/git/CodeVault/training-material/parallel-programming/MPI/-tree/master/heat-equation>

Merge branch 'master' of https://repository.prace-ri.eu/git/jussi.enkovaara/MPI  
Jussi Enkovaara authored 4 years ago

master / MPI / heat-equation

Name	Last commit	Last update
..		
c	Small clean up of the code	4 years ago
fortran	Fortran version of 2D heat equation	4 years ago
python	Add few notes about Python implementation and building the Cython	4 years ago
LICENSE.txt	Add th LICENSE	4 years ago
README.md	Correct the default value	4 years ago

README.md

Two dimensional heat equation

本地: [MPI-master.zip](#)

[MPI-master-heat-equation.zip](#)

新加卷 (D:) > Local > ++写书 > 18 高性能计算 > HPCBook-MD > materials > MPI-master-heat-equation > heat-equation > python

名称	修改日期	类型	大小
bottle.dat	2018,9/5,周三 14:20	DAT 文件	387 KB
evolve.py	2018,9/5,周三 14:20	PY 文件	3 KB
evolve.pyx	2018,9/5,周三 14:20	PYX 文件	3 KB
exchange.py	2018,9/5,周三 14:20	PY 文件	2 KB
heat.py	2018,9/5,周三 14:20	PY 文件	2 KB
heat_io.py	2018,9/5,周三 14:20	PY 文件	4 KB
heat_setup.py	2018,9/5,周三 14:20	PY 文件	2 KB
parallel.py	2018,9/5,周三 14:20	PY 文件	4 KB
README.md	2018,9/5,周三 14:20	Markdown File	1 KB
setup.py	2018,9/5,周三 14:20	PY 文件	1 KB

## [我的代码]

```
1 """
2 D:\myCodes\HPCbook\02mpi4py\ndArraysSplitTestMPICart8.py
3 2023年6月5日21:07:20 终于解决了！！！
4 艰苦路程，参看 - "D:\Local\++写书\18 高性能计算\HPCBook-MD\05-MPI4py-B-
5 stencil.md"
6 """
7 from __future__ import division
8 from __future__ import print_function
9 import numpy as np
10 import time
11 import math
12 from mpi4py import MPI
13
14 import matplotlib
15
16 matplotlib.use('Agg')
17 import matplotlib.pyplot as plt
18
19 # Set the colormap
20 # plt.rcParams['image.cmap'] = 'BrBG'
21 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
22 plt.cm.coolwarm
23 # ++++++ MPI ++++++
24 def init_fields():
25
26     # Set array size and set the interior value with Tguess
27     field = np.zeros((lenX, lenY), dtype = np.float32)
28     # field = np.linspace(1, lenX*lenY, lenX*lenY).reshape((lenX, lenY))
29     # field = np.linspace(1, lenX*lenY, lenX*lenY).reshape((lenX, lenY))
30     field.fill(Tguess)
31
32     # Set Boundary condition
33     field[(lenY-1):, :] = Ttop
34     field[:, :1] = Tbottom
35     field[:, (lenX-1):] = Tright
36     field[:, :1] = Tleft
37
38     print("size is ", field.size)
39     print(field, "\n")
40
41     return field
42
43 """
44 借助 输入的 neighbour, 可以确定
45 """
46 def exchange(localField, neighbour):
47     validDirections = np.where(np.array(neighbour) != -1)
48     # print("\nvalidDirections[0]\n", validDirections[0])
49     for direction in validDirections[0]:
```

```

50         rbuf = np.zeros(localField.shape[1], dtype = localField.dtype)
51         if direction == 0: # Down
52             sbuf = localField[-2, :]
53             rbuf = localField[-1, :]
54             comm.Sendrecv(np.array(sbuf), dest=neighbour[direction],
55             recvbuf=rbuf, source=neighbour[direction])
56
57         elif direction == 1: # Right
58             sbuf = localField[:, -2]
59             # rbuf = localField[:, -1]
60             comm.Sendrecv(np.array(sbuf), dest=neighbour[direction],
61             recvbuf=rbuf, source=neighbour[direction])
62             localField[:, -1] = np.array(rbuf).T
63
64         elif direction == 2: # Up
65             sbuf = localField[1, :]
66             rbuf = localField[0, :]
67             comm.Sendrecv(np.array(sbuf), dest=neighbour[direction],
68             recvbuf=rbuf, source=neighbour[direction])
69             localField[:, 0] = np.array(rbuf).T
70
71
72 def evolve(u, u_previous, a, dt, dx2, dy2, neighbour):
73     """Explicit time evolution.
74         u:           new temperature field
75         u_previous: previous field
76         a:           diffusion constant
77         dt:          time step
78         dx2:         grid spacing squared, i.e. dx^2
79         dy2:         -- "" -- , i.e. dy^2"""
80     # neigh = [Down, Right, Up, Left]
81     up = 1
82     down = u.shape[0]-1
83     right = u.shape[1]-1
84     left = 1
85
86     validDirections = np.where(np.array(neighbour) ==-1)
87     for direction in validDirections[0]:
88         if direction == 0: # Down
89             down = u.shape[0]-2
90         elif direction == 1: # Right
91             right = u.shape[1]-2
92         elif direction == 2: # Up
93             up = 2
94         else: # Left
95             left = 2
96
97         # print("u and u_previous in evolve()\n",u,"\\n", u[up:down,
98         left:right], "\\n")
99         u[up:down, left:right] = u_previous[up:down, left:right] + a * dt * (
100             (u_previous[up+1:down+1, left:right] - 2 * u_previous[up:down,
101             left:right] +

```

```

100             u_previous[up-1:down-1, left:right]) / dx2 +
101             (u_previous[up:down, left+1:right+1] - 2 * u_previous[up:down,
102               left:right] +
103               u_previous[up:down, left-1:right-1]) / dy2)
104         u_previous[:, :] = u[:, :]
105         # print("AFTER: u and u_previous in evolve()\n", u, "\n", u[up:down,
106           left:right], "\n")
107
108     def write_field(gatheredField, step):
109         plt.gca().clear()
110         plt.clf()
111
112         plt.figure(dpi=300)
113         # Configure the contour
114         plt.title("Contour of Temperature")
115         plt.contourf(X, Y, gatheredField, colorinterpolation, cmap=colourMap)
116         # Set colorbar
117         plt.colorbar()
118         plt.axis('on')
119         plt.savefig('heat_MPIStencil_{0:03d}.png'.format(step))
120
121     def hvsplit(matrix, hvgrid):
122         matrixHsSplit = np.hsplit(matrix, hvgrid[0])
123         full_split = [np.vsplit(matrixHsSplit[i], hvgrid[1]) for i in
124           range(hvgrid[1])]
125         return full_split
126
127     def hvFlatten(fullsplit, hvgrid):
128         hvFlattenMat = []
129         for i in range(hvgrid[0]):
130             for j in range(hvgrid[1]):
131                 hvFlattenMat = np.concatenate((hvFlattenMat, fullsplit[j]
132                   [i].flatten()))
133         return hvFlattenMat
134
135     def reverseHVFlatten(hvfield, datashape, hvgrid):
136         hvfield2D = hvfield.reshape(datashape)
137         # print("\nhvfield2D:\n", hvfield2D)
138         hvfield2DReverse =
139         np.array_split(hvfield2D.flatten(), hvgrid[0]*hvgrid[1])
140         hvfield2DReverseHV =
141         [hvfield2DReverse[i].reshape((datashape[0]//hvgrid[0],
142           datashape[1]//hvgrid[1])) for i in range(hvgrid[0]*hvgrid[1])]
143         aaa = []
144         for i in range(hvgrid[1]):
145             aaa.append(np.block(hvfield2DReverseHV[i*hvgrid[1]:i*hvgrid[1]+hvgrid[1]:]))
146         return np.block([aaa[i].flatten() for i in
147           range(hvgrid[1])]).reshape(datashape)
148
149     lenX = lenY = 400
150     dataShape = (lenX, lenY)
151
152     # 应严格满足数据能够规整划分！！

```

```
146 # 400x400
147 # 16 = 4x4, 那么, 每个local 就是 100x100
148 mpi_rows = mpi_cols = 4
149 # 也就意味着 mpicexec -np number 中的number 一定要等于mpi_rows*mpi_cols = 9
150 hvGrid = (mpi_rows, mpi_cols)
151
152 # Basic parameters
153 a = 0.1 # Diffusion constant #例如 Thermal diffusivity of steel, mm2.s-1
154 timesteps = 10000 # Number of time-steps to evolve system
155 image_interval = 1000 # Write frequency for png files
156
157 # Grid spacings
158 dx = 0.01
159 dy = 0.01
160 dx2 = dx ** 2
161 dy2 = dy ** 2
162
163 # For stability, this is the largest interval possible
164 # for the size of the time-step:
165 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
166
167 # Set Dimension and delta
168 delta = 1.
169
170 # Boundary condition
171 Ttop = 100.
172 Tbottom = 0.
173 Tleft = 0.
174 Tright = 30.
175
176 # Initial guess of interior grid
177 Tguess = 0.
178
179 # MPI globals
180 comm = MPI.COMM_WORLD
181 rank = comm.Get_rank()
182 size = comm.Get_size()
183
184 # print("Creating a {} x {} processor grid...".format(mpi_rows, mpi_cols) )
185 ccomm = comm.Create_cart((mpi_rows, mpi_cols), periods=(False, False),
186 reorder = False)
187
188 if rank == 0:
189     oringin = init_fields()
190     # field = np.linspace(1, lenX*LenY, lenX*LenY).reshape((lenX, LenY))
191     # print("size is ",oringin.size)
192     # print(oringin,"\n")
193
194     full_split = hvSplit(oringin, hvGrid)
195
196     # print('\nb_fullsplit: ')
197     # print(full_split)
198
199     bbb = hvFlatten(full_split, hvGrid)
200     # print('\nbccc: ', bbb)
```

```

201
202     field = np.array(bbb).reshape(dataShape)
203     # print('\nfield for scatter(): \n', field, "\n", field.dtype)
204
205     # Set colour interpolation and colour map
206     colorInterpolation = 100
207     colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
208
209     # Set meshgrid
210     X, Y = np.meshgrid(np.arange(0, field.shape[0]), np.arange(0,
211                         field.shape[1]))
212
213     # hvField = reverseHVFlatten(field, dataShape, hvGrid)
214     # write_field(hvField, 0)
215     write_field(oringin, 0)
216
217     shape = field.shape
218     dtype = field.dtype
219     localShape = (dataShape[0]//hvGrid[0], dataShape[1]//hvGrid[1])
220     # print(localShape)
221     # print('\ndtype = {}'.format(dtype))
222
223     comm.bcast(shape, root = 0) # broadcast dimensions
224     comm.bcast(dtype, root = 0) # broadcast data type
225     comm.bcast(localShape, root = 0)
226 else:
227     field = None
228     full_split = None
229     shape = comm.bcast(None, 0)
230     dtype = comm.bcast(None, 0)
231     localShape = comm.bcast(None, 0)
232
233     buff = np.zeros(localShape, dtype)
234
235     comm.Scatter(field, buff, root = 0) # scatter the data
236     # print("\nbuff:\n", buff)
237
238     my_mpi_row, my_mpi_col = ccomm.Get_coords(comm.rank)
239     neigh = [0,0,0,0]
240     # neigh = [Down, Right, Up, Left]
241     DOWN = 0
242     RIGHT = 1
243     UP = 2
244     LEFT = 3
245
246     neigh[UP], neigh[DOWN] = ccomm.shift(0, 1)
247     neigh[LEFT], neigh[RIGHT] = ccomm.shift(1, 1)
248
249     # print('\nPID is {} with index as ({}, {})\n'.format(rank,my_mpi_row,my_mpi_col))
250     # print("localshape\n", localShape)
251     # print("\nnits neighbour is {}\n".format(neigh))
252
253     local_field = np.zeros((localShape[0]+2, localShape[1]+2), dtype) # need 4
254     ghost_rows!

```

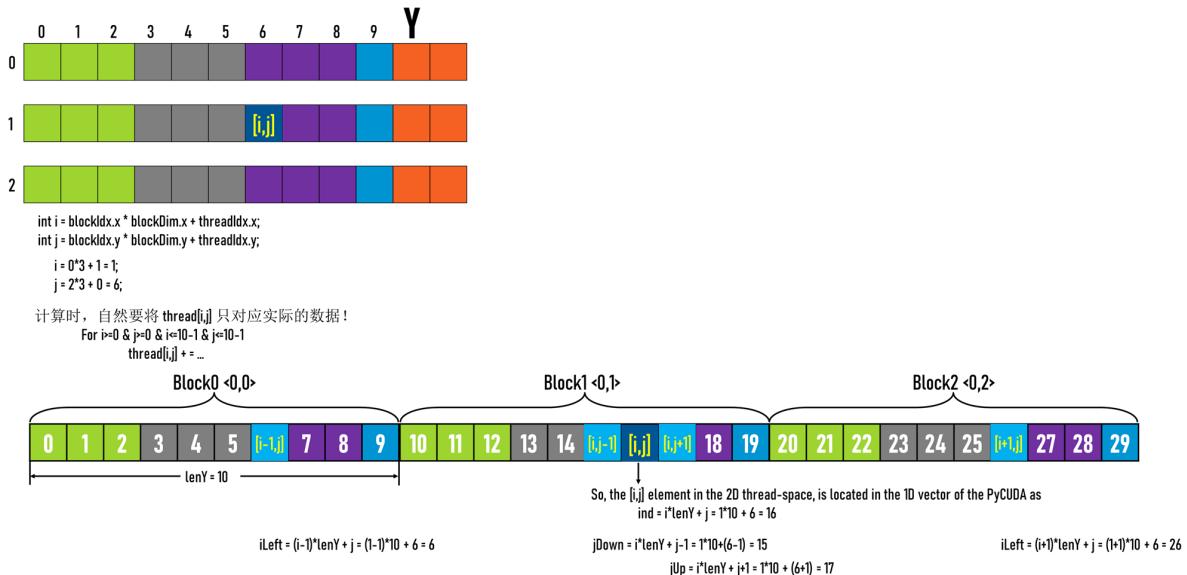
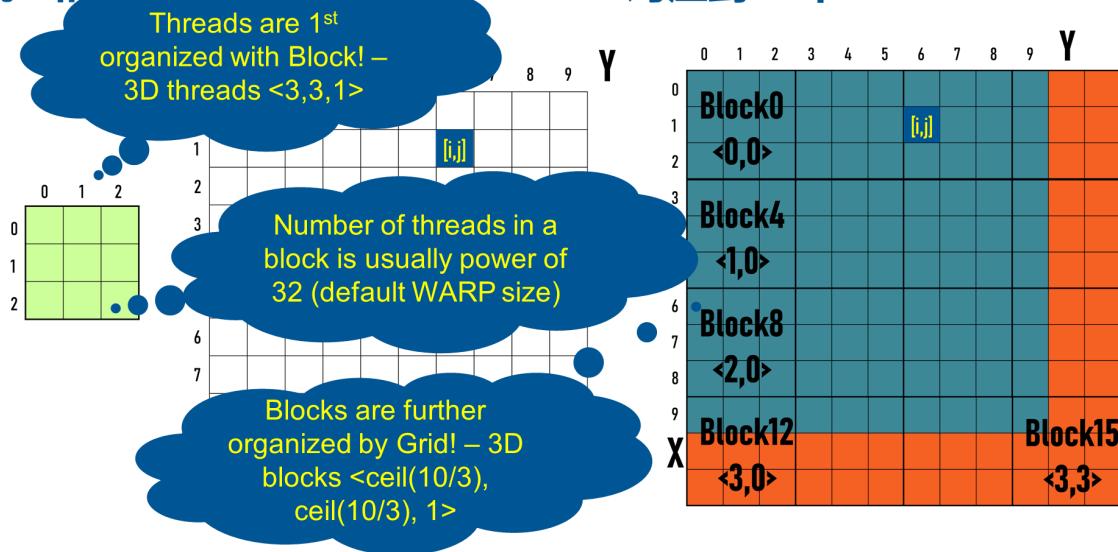
```

254 # print("\n local_field:\n", local_field)
255 local_field[1:-1, 1:-1] = buff[:] # copy data to non-ghost rows
256 # print("\n local_field after buff:\n", local_field)
257 local_field0 = np.zeros_like(local_field) # array for previous time step
258
259 # print("\nlocal_field.shape", local_field.shape)
260 # print("\n", local_field)
261 # print("\n", local_field[0][0])
262
263 # exchange(local_field,neigh)
264 local_field0[:] = local_field[:]
265 # print("\nafter local_field0[:] = local_field[:]\n", local_field0)
266
267 # Iterate
268 t0 = time.time()
269 for m in range(1, timesteps + 1):
270     exchange(local_field, neigh)
271     evolve(local_field, local_field0, a, dt, dx2, dy2, neigh)
272
273     if m % image_interval == 0:
274         # exchange(local_field, neigh)
275         # evolve(local_field, local_field0, a, dt, dx2, dy2, neigh)
276         # print("\nlocal_field after evolve():\n", local_field)
277         # print("\nBefore comm.Gather()")
278         comm.Gather(np.array(local_field[1:-1, 1:-1], dtype), field, root =
0)
279
280         if rank == 0:
281             # comm.Igather(local_fieldGather, field, root = 0)
282             # print("\nGathered field: \n", field, "\n")
283             hvField = reverseHVFflatten(field, dataShape, hvGrid)
284             write_field(hvField, m)
285             # write_field(field, m)
286             # if m == timesteps:
287                 # print("\nGathered field: \n", hvField, "\n")
288 t1 = time.time()
289
290 # Plot/save final field
291 # comm.Gather(local_field[1:-1, 1:-1], field, root=0)
292 # comm.Gather(local_field, field, root=0)
293
294 if rank == 0:
295     # print('\nfield shape:')
296     # print(field.shape)
297     # write_field(field, 10)
298     print("Running time: {0}".format(t1 - t0))

```

# CUDA 编程基本概念

统一的理解：将每一个  $[i,j]$  element 对应到一个 thread！



[Images]PyCUDA

线索

<https://github.com/cschnpc/heat-equation>

The screenshot shows a GitHub repository page for `heat-equation`. The `cuda` branch is selected. A commit titled "Bug fix" is highlighted with a red box. Below the commit list, the `core_cuda.cu` file is displayed. A specific line of code is highlighted with a blue box.

```

1  /* Main solver routines for heat equation solver */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <assert.h>
7  #include <omp.h>
8  #include <cuda_runtime_api.h>
9
10 #include "heat.h"
11
12 /* Update the temperature values using five-point stencil */
13 __global__ void evolve_kernel(double *currdta, double *prevdta, double a, double dt, int nx, int ny,
14                             double dx2, double dy2)
15 {
16
17     /* Determine the temperature field at next time step
18     * As we have fixed boundary conditions, the outermost gridpoints
19     * are not updated. */
20     int ind, ip, im, jp, jm;
21
22     // CUDA threads are arranged in column major order; thus j_index from x, i from y
23     int j = blockIdx.x * blockDim.x + threadIdx.x;
24     int i = blockIdx.y * blockDim.y + threadIdx.y;
25
26     if (i > 0 && j > 0 && i < nx-1 && j < ny-1) {
27         ind = i * (ny+2) + j;
28         ip = (i+1) * (ny+2) + j;

```

## 我的实践

- 人家的CUDA kernel 代码可以直接拷贝至 PyCUDA中

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <assert.h>
5 #include <mpi.h>
6 #include <cuda_runtime_api.h>
7
8 #include "heat.h"
9
10 /* Update the temperature values using five-point stencil */
11 __global__ void evolve_kernel(double *currdat, double *prevdata, double a, double dt, int nx, int ny,
12                             double dx2, double dy2)
13 {
14
15
16     /* Determine the temperature field at next time step
17      * As we have fixed boundary conditions, the outermost gridpoints
18      * are not updated. */
19     int ind, ip, im, jp, jm;
20
21     // CUDA threads are arranged in column major order; thus j index from x, i from y
22     int j = blockIdx.x * blockDim.x + threadIdx.x;
23     int i = blockIdx.y * blockDim.y + threadIdx.y;
24
25
26     if (i > 0 && j > 0 && i < nx-1 && j < ny-1) {
27         ind = i * (ny + 2) + j;
28         ip = (i + 1) * (ny + 2) + j;
29         im = (i - 1) * (ny + 2) + j;
30         jp = i * (ny + 2) + j + 1;
31         jm = i * (ny + 2) + j - 1;
32
33         currdat[ind] = prevdata[ind] + a * dt *
34             ((prevdata[ip] - 2.0 * prevdata[ind] + prevdata[im]) / dx2 +
35              (prevdata[jp] - 2.0 * prevdata[ind] + prevdata[jm]) / dy2);
36     }
37
38 }
39
40 void evolve(field *curr, field *prev, double a, double dt)
41 {
42     int nx, ny;
43     double dx2, dy2;

```

- 一定要注意参数传递问题！

**PyCUDA中的Kernel，本质上是C/C++的，这就意味着参数传递时一定要在Python中就指明数据类型！**

```

1 dt = np.float32(dt)
2 lenX = np.int32(lenX)
3 field = field.astype(np.float32)
4

```

## [我的代码]

```

1 /**
2 https://github.com/csc-training/hpc-python/blob/master/mpi/heat-
3 equation/solution/heat-p2p.py
4 錄轰篠 "D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-
5 heatequ2dk.py" 哆澆口淇口敲鏹?
6 2023賽?鍊?1鍏?1:29:07 鍏懃娴鑽勭增鍊口紵
7 鍮勭增口鐗塙逢鑪?"D:\myCodes\HPCprojects\SourceCodes\parallel_python-
8 master\examplesHeatMPI\simpleFDM-PyCUDA.py"
9 鑷困。璧勬枅 "D:\Local\++鑲欎功\18 楢欒€u祖璁$疇\HPCBook-MD\05-GPU-A-PyCUDA.md"
10
11 /**
12 from __future__ import print_function
13 import numpy as np
14 import time
15 import matplotlib
16
17 matplotlib.use('Agg')
18 import matplotlib.pyplot as plt
19

```

```

20 # Set the colormap
21 # plt.rcParams['image.cmap'] = 'BrBG'
22 plt.rcParams['image.cmap'] = 'jet' # you can try: colourMap =
23 plt.cm.coolwarm
24 # ++++++#
25 # Set colour interpolation and colour map
26 colorinterpolation = 100
27 colourMap = plt.cm.jet # you can try: colourMap = plt.cm.coolwarm
28
29 import pycuda.autoinit
30 import pycuda.driver as cuda
31 from pycuda import gpuarray
32 from pycuda.compiler import SourceModule
33
34 # Basic parameters
35 a = 0.1 # Diffusion constant
36
37 timesteps = 10000 # Number of time-steps to evolve system
38 image_interval = 1000 # Write frequency for png files
39
40 # Grid spacings
41 dx = 0.01
42 dy = 0.01
43 dx2 = dx ** 2
44 dy2 = dy ** 2
45
46 # For stability, this is the largest interval possible
47 # for the size of the time-step:
48 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
49
50 # Set Dimension and delta
51 lenX = lenY = 400 # we set it rectangular
52 delta = 1
53
54 # Boundary condition
55 Ttop = 100
56 Tbottom = 0
57 Tleft = 0
58 Tright = 30
59
60 # Initial guess of interior grid
61 Tguess = 0
62
63 # Set meshgrid
64 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
65
66 def write_field(field, step):
67     plt.gca().clear()
68     # plt.clf()
69
70     plt.figure(dpi=300)
71     # Configure the contour
72     plt.title("Contour of Temperature")
73     plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
74     # Set Colorbar

```

```

75     plt.colorbar()
76     plt.axis('on')
77     plt.savefig('heat_PyCUDA_{0:03d}.png'.format(step))
78
79
80 a = np.float32(a)
81 lenX = np.int32(lenX)
82 lenY = np.int32(lenY)
83 dx = np.float32(dx)
84 dy = np.float32(dy)
85 dx2 = np.float32(dx2)
86 dy2 = np.float32(dy2)
87 dt = np.float32(dt)
88 timesteps = np.int32(timesteps)
89 image_interval = np.int32(image_interval)
90
91 #
#++++#
92 # evolve_kernel 鍋ヨ嚞 D:\myCodes\HPCprojects\heat-equation-
main\cuda\core_cuda.cu
93 ker = SourceModule('''
94 /* Update the temperature values using five-point stencil */
95 __global__ void evolve_kernelCUDA(float *currdata, float *prevdata, float
a, float dt, int nx, int ny,
                           float dx2, float dy2)
96 {
97
98     /* Determine the temperature field at next time step
99      * As we have fixed boundary conditions, the outermost gridpoints
100     * are not updated. */
101     int ind, iRight, iLeft, jUp, jDown;
102
103     // CUDA threads are arranged in column major order; thus j index from
x, i from y
104     int i = blockIdx.x * blockDim.x + threadIdx.x;
105     int j = blockIdx.y * blockDim.y + threadIdx.y;
106
107     if (i > 0 && j > 0 && i < nx-1 && j < ny-1) {
108         ind = i * ny + j;
109         iRight = (i+1) * (ny) + j;
110         iLeft = (i - 1) * (ny ) + j;
111         jUp = i * (ny) + j + 1;
112         jDown = i * (ny) + j - 1;
113         currdata[ind] = prevdata[ind] + a * dt *
114             ((prevdata[iRight] -2.0 * prevdata[ind] + prevdata[iLeft]) / dx2
115 +
116             (prevdata[jUp] - 2.0 * prevdata[ind] + prevdata[jDown]) / dy2);
117     }
118 }
119 ''')
120
121 evolve_kernel = ker.get_function('evolve_kernelCUDA')
122
123
124 # 淩 g 燥鍊ヨ嚞 D:\myCodes\HPCprojects\heat-equation-main\cuda\core_cuda.cu

```

```

125 # void evolve(field *curr, field *prev, double a, double dt)
126 #
127 # int nx, ny;
128 # double dx2, dy2;
129 # nx = prev->nx;
130 # ny = prev->ny;
131 # dx2 = prev->dx * prev->dx;
132 # dy2 = prev->dy * prev->dy;
133
134 # /* CUDA thread settings */
135 # const int blocksize = 16; //!< CUDA thread block dimension
136 # dim3 dimBlock(blocksize, blocksize);
137 # // CUDA threads are arranged in column major order; thus make ny x nx
138 # grid
139 # dim3 dimGrid((ny + 2 + blocksize - 1) / blocksize,
140 # (nx + 2 + blocksize - 1) / blocksize);
141 # evolve_kernel<<<dimGrid, dimBlock>>>(curr->devdata, prev->devdata, a, dt,
142 nx, ny, dx2, dy2);
143 # cudaDeviceSynchronize();
144 #
145
146 #
147 #####
148 def main():
149
150     # Set array size and set the interior value with Tguess
151     field = np.empty((lenx, lenY))
152     field.fill(Tguess)
153
154
155     # Set Boundary condition
156     field[(lenY - 1):, :] = Ttop
157     field[:, :1] = Tbottom
158     field[:, (lenx - 1):] = Tright
159     field[:, :1] = Tleft
160
161     # gpuarray.to_gpu(field, allocator=None)
162     # field =
163     gpuarray.to_gpu(np.random.randn(lenX, lenX).astype(np.float32))
164     # field_gpuArr = gpuarray.to_gpu(field.astype(np.float32))
165
166     print("field's size is ", field.size)
167     print(field, "\n")
168
169     field0 = field.copy() # Array for field of previous time step
170     print("field0's size is ", field0.size)
171     print(field0, "\n")
172
173     write_field(field, 0)
174
175     field = field.astype(np.float32)
176     field0 = field0.astype(np.float32)

```

```

177
178     blocksize = 32 # !< CUDA thread block dimension
179     dimBlock = (blocksize, blocksize, 1)
180     # CUDA threads are arranged in column major order; thus make ny x nx
181     grid
182         # dimGrid = ((lenY + 2 + blocksize - 1) // blocksize,
183             # (lenX + 2 + blocksize - 1) // blocksize,
184             # 1)
185
186     dimGrid = (int(lenY/blocksize+(0 if lenY % blocksize == 0 else 1)),
187                 int(lenY/blocksize+(0 if lenX % blocksize == 0 else 1)),
188                 1)
189     print(dimBlock)
190     print(dimGrid)
191     print()
192
193     # Allocate memory on device
194     field_gpu = cuda.mem_alloc(field.nbytes)
195     field0_gpu = cuda.mem_alloc(field0.nbytes)
196
197     # Iterate
198     t0 = time.time()
199     for m in range(1, timesteps + 1):
200         # Copy matrix to memory
201         cuda.memcpy_htod(field_gpu, field)
202         cuda.memcpy_htod(field0_gpu, field0)
203
204         evolve_kernel(field_gpu, field0_gpu, a, dt, lenX, lenY, dx2, dy2,
205 block=dimBlock, grid=dimGrid)
206         # cuda.cudaDeviceSynchronize()
207         # cuda.Context.synchronize()
208
209         if (m % image_interval == 0):
210             # Copy back the result
211             cuda.memcpy_dtoh(field, field_gpu)
212             write_field(field, m);
213             # print(field, "\n")
214
215             cuda.memcpy_dtoh(field, field_gpu)
216             cuda.memcpy_dtoh(field0, field0_gpu)
217
218             # Swap current field so that it will be used as previous for next
219             iteration step
220             tmp = field
221             field = field0
222             field0 = tmp
223
224             t1 = time.time()
225             print("Running time: {0}".format(t1 - t0))
226
227             field_gpu.free()
228             field0_gpu.free()
229
230     if __name__ == '__main__':
231         main()

```

# Numba

## 从OpenMP 到 Python 的 Numba

### OpenMP

The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. The OpenMP API defines a portable, scalable model with a simple and flexible interface for developing parallel applications on platforms from the desktop to the supercomputer.

- 起源于ANSI X3H5（1994）标准
- 由设备商和编译器开发者共同制定，"工业标准"（1997）
- 编程简单，增量化并行，移植性好，可扩展性好
- 支持 Fortran, C/C++（编译器自带 OpenMP）  
(<http://openmp.org/wp/openmp-compilers/>)
- 支持 Unix, Linux, Windows 等操作系统
- AMD, Intel, IBM, Cray, NEC, HP, NVIDIA, ... ...



录制中

基础知识 Fork-Join

超级云讲堂 bili bili

ACM Association for Computing Machinery IPCC ACM中国 国际并行计算挑战赛

• OpenMP采用的是Fork-Join模式进行并行执行

• 开始时程序只有一个线程，我们称之为master，编号0

• 主线程进入**并行区** (parallel region) 时开始并行执行

- Fork: 主线程创建一组并行线程，包括其自己在内编号0至(n-1)
- 执行: 并行区内每个线程并行执行
- Join: 在并行区结尾时，进行同步，其他线程结束，只保留主线程

Parallel Task I Parallel Task II Parallel Task III

Master Thread

Parallel Task I Parallel Task II Parallel Task III

北京大学 叶子凌峰

叶子凌峰

## OpenMP: Work sharing example

Sequential code

```
for (int i=0; i<N; i++) { a[i]=b[i]+c[i]; }
```

(Semi) manual parallelization

```
#pragma omp parallel
{
    int id = omp_get_thread_num();
    int nt = omp_get_num_threads();
    int i_start = id*N/nt, i_end = (id+1)*N/nt;
    for (int i=i_start; i<i_end; i++) { a[i]=b[i]+c[i]; }
}
```

Automatic parallelization of the for loop using  
**#parallel for**

```
#pragma omp parallel
#pragma omp for schedule(static)
{
    for (int i=0; i<N; i++) { a[i]=b[i]+c[i]; }
}
```

## Numba! One of Python package for OpenMP



- Modern technology should be able to map Python onto low-level code (such as C or LLVM) and avoid the “Python performance tax”.
- We've worked on ...
  - Numba (2012): JIT Python code into LLVM
  - Parallel accelerator (2017): Find and exploit parallel patterns in Python code.
  - Intel High-Performance Analytics Toolkit and Scalable Dataframe Compiler (2019): Parallel performance from data frames.
  - Intel numba-dppy (2020): Numba ParallelAccelerator regions that run on GPUs via SYCL.



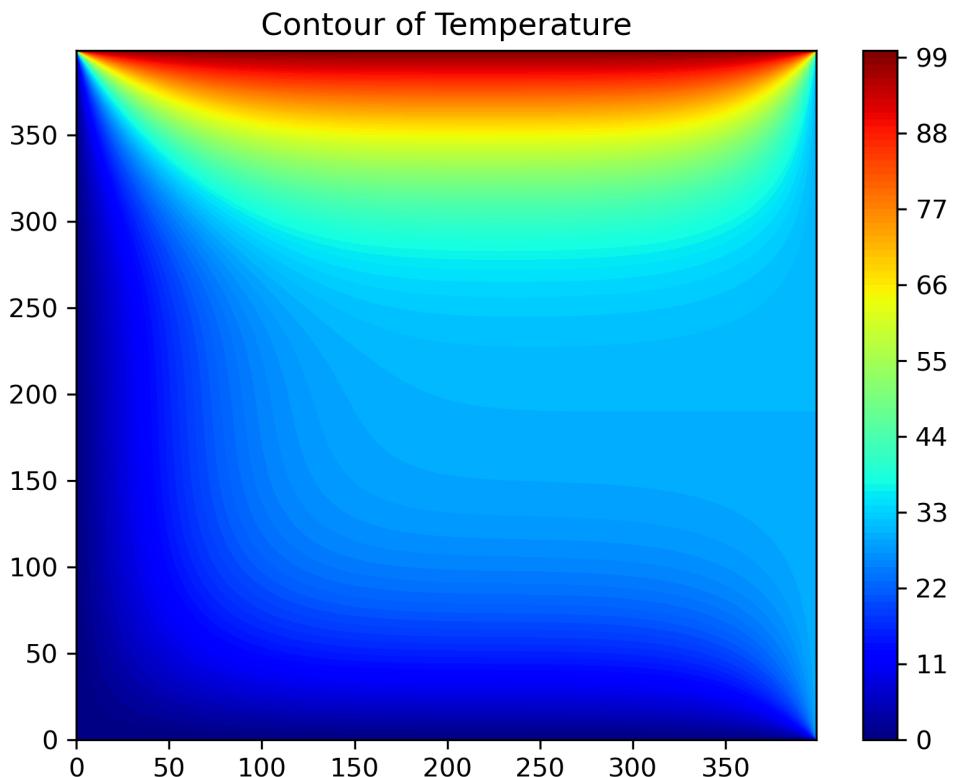
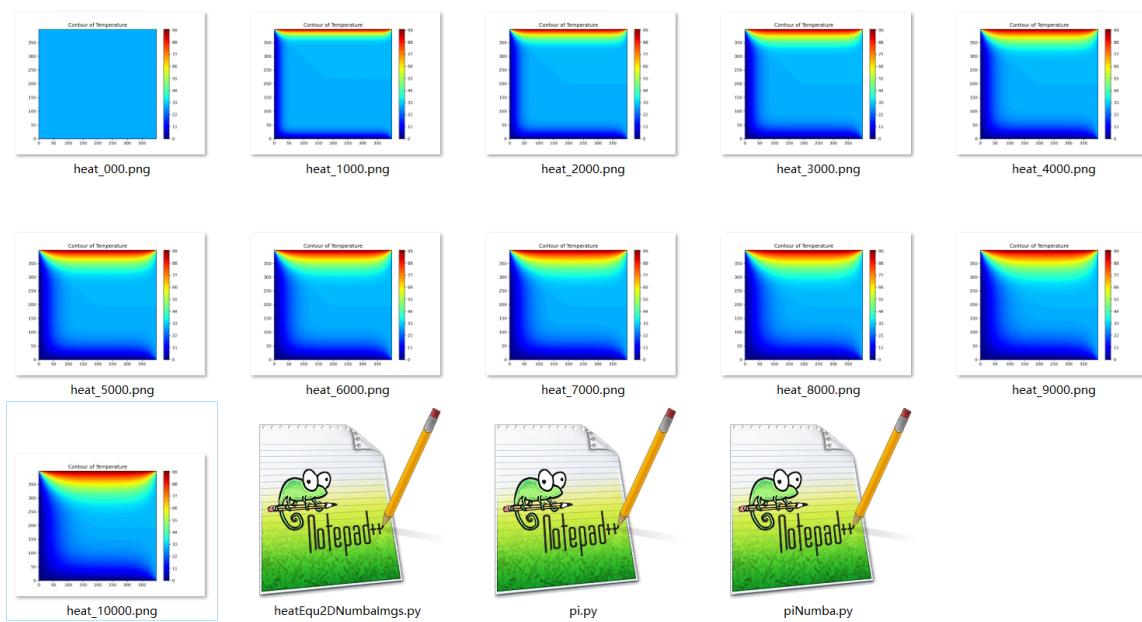
295

## [Images]Numba

### 线索

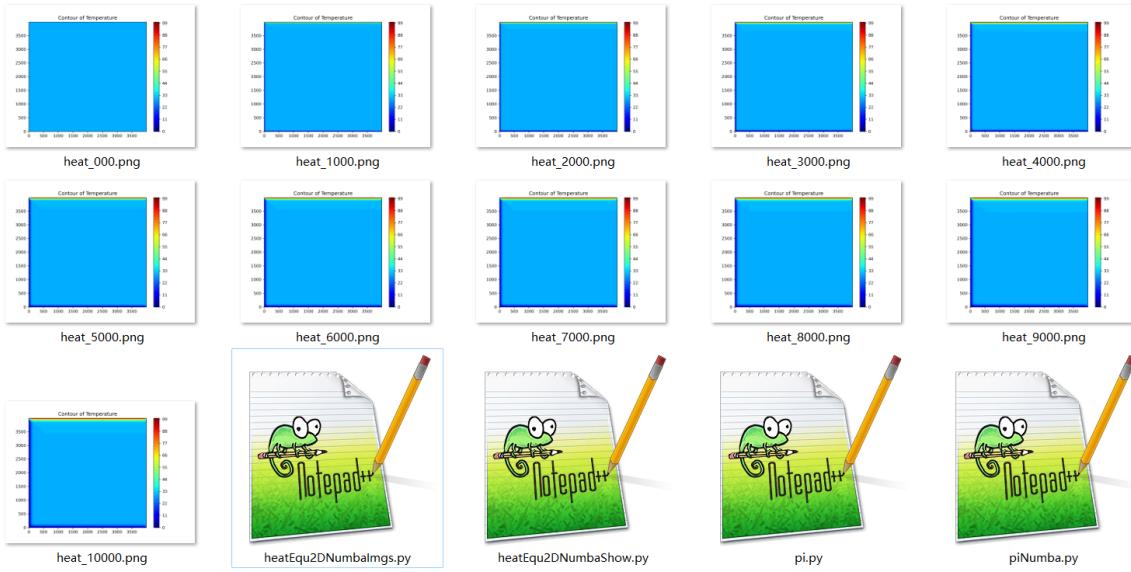
- 将更新单独作为一个函数 - `def evolve()`
- 在 `def evolve()` 前使用 Numba 的 Decorator 函数 - 如 `@numba.jit`

## 我的代码的效果



但是，当数据是 4000\*4000 时，就出问题了

```
1 # Set Dimension and delta
2 lenX = lenY = 4000 #we set it rectangular
3
4 @njit(fastmath=True)
5 def evolve(u, u_previous, lenX, lenY, a, dt, dx2, dy2):
```



为什么?

## [我的代码]

```

1  """ https://www.codeproject.com/Articles/1087025/Using-Python-to-Solve-
   Computational-Physics-Proble
2  # D:\My7\MyProjects\Memoirs\2018-12-18-Heat Transfer with Python.docx
3  # 2019骞?鍾?鏃?3:03:49
4
5  2023骞?鍾?鏃?5:56:13
6  D:\myCodes\HPCprojects\SourceCodes\parallel_python-
   master\examplesHeatMPI\simpleFDM-Numba1.py
7 """
8 # Simple Numerical Laplace Equation Solution using Finite Difference Method
9 import numpy as np
10 from numba import jit, njit, prange
11 import timeit
12
13
14 import matplotlib
15
16 matplotlib.use('Agg')
17 import matplotlib.pyplot as plt
18
19 # Set the colormap
20 # plt.rcParams['image.cmap'] = 'BrBG'
21 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
22 plt.cm.coolwarm
23 plt.figure(dpi=300)
24
25 # Set colour interpolation and colour map
26 colorinterpolation = 100
27 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm

```

```

27
28
29 # Basic parameters
30 a = 0.1 # Diffusion constant
31 timesteps = 10000 # Number of time-steps to evolve system
32 image_interval = 1000 # write frequency for png files
33
34 # Grid spacings
35 dx = 0.01
36 dy = 0.01
37 dx2 = dx ** 2
38 dy2 = dy ** 2
39
40 # For stability, this is the largest interval possible
41 # for the size of the time-step:
42 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
43
44
45 # Set Dimension and delta
46 lenX = lenY = 400 #we set it rectangular
47 delta = 1
48
49 # Boundary condition
50 Ttop = 100
51 Tbottom = 0
52 Tleft = 0
53 Tright = 30
54
55 # Initial guess of interior grid
56 Tguess = 0
57
58 # Set meshgrid
59 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
60
61 def init_fields():
62     # Set array size and set the interior value with Tguess
63     field = np.empty((lenX, lenY))
64     field.fill(Tguess)
65
66     # Set Boundary condition
67     field[(lenY-1):, :] = Ttop
68     field[:, :1] = Tbottom
69     field[:, (lenX-1):] = Tright
70     field[:, :1] = Tleft
71
72     field0 = field.copy() # Array for field of previous time step
73     return field, field0
74
75 # nopython = True选项要求完全编译该函数（以便完全删除Python解释器调用），否则会引发异常。
76 # 这些异常通常表示函数中需要修改的位置，以实现优于Python的性能。强烈建议您始终使用nopython = True。
77 # @jit(nopython=True)
78 # @numba.njit(cache=True, parallel=True)
79 # @jit(nopython = True, parallel = True, nogil = True)
80 # @jit

```

```

81 # @njit(fastmath=True)
82 @njit(fastmath=True)
83 def evolve(u, u_previous, lenX, lenY, a, dt, dx2, dy2):
84     """Explicit time evolution.
85         u:           new temperature field
86         u_previous: previous field
87         a:           diffusion constant
88         dt:          time step
89         dx2:         grid spacing squared, i.e. dx^2
90         dy2:         -- " " -- , i.e. dy^2"""
91     for i in range(1, lenX-1, delta):
92         for j in range(1, lenY-1, delta):
93             u[i, j] = u_previous[i, j] + a * dt * (
94                 (u_previous[i+1, j] - 2 * u_previous[i, j] +
95                  u_previous[i-1, j]) / dx2 +
96                 (u_previous[i, j+1] - 2 * u_previous[i, j] +
97                  u_previous[i, j-1]) / dy2)
98     u_previous[:] = u[:]
99
100
101 def write_field(field, step):
102     # plt.gca().clear()
103     plt.cla()
104     plt.clf()
105
106     plt.figure(dpi=300)
107     # Configure the contour
108     plt.title("Contour of Temperature")
109     plt.contourf(x, Y, field, colorinterpolation, cmap=colourMap)
110     # Set Colorbar
111     plt.colorbar()
112     plt.axis('on')
113     plt.savefig('heat_Numba_{0:03d}.png'.format(step))
114
115
116 def main():
117     field, field0 = init_fields()
118     write_field(field, 0)
119
120     print("size is ", field.size)
121     print(field, "\n")
122
123     # Iteration (We assume that the iteration is convergence in maxIter =
124     500)
125     print("Please wait for a moment")
126
127     starting_time = timeit.default_timer()
128     for iteration in range(0, timesteps+1):
129         evolve(field, field0, lenX, lenY, a, dt, dx2, dy2)
130         if iteration % image_interval == 0:
131             write_field(field, iteration)
132
133     print("Iteration finished. {} Seconds for Time
difference:".format(timeit.default_timer() - starting_time))
134
135 if __name__ == '__main__':

```

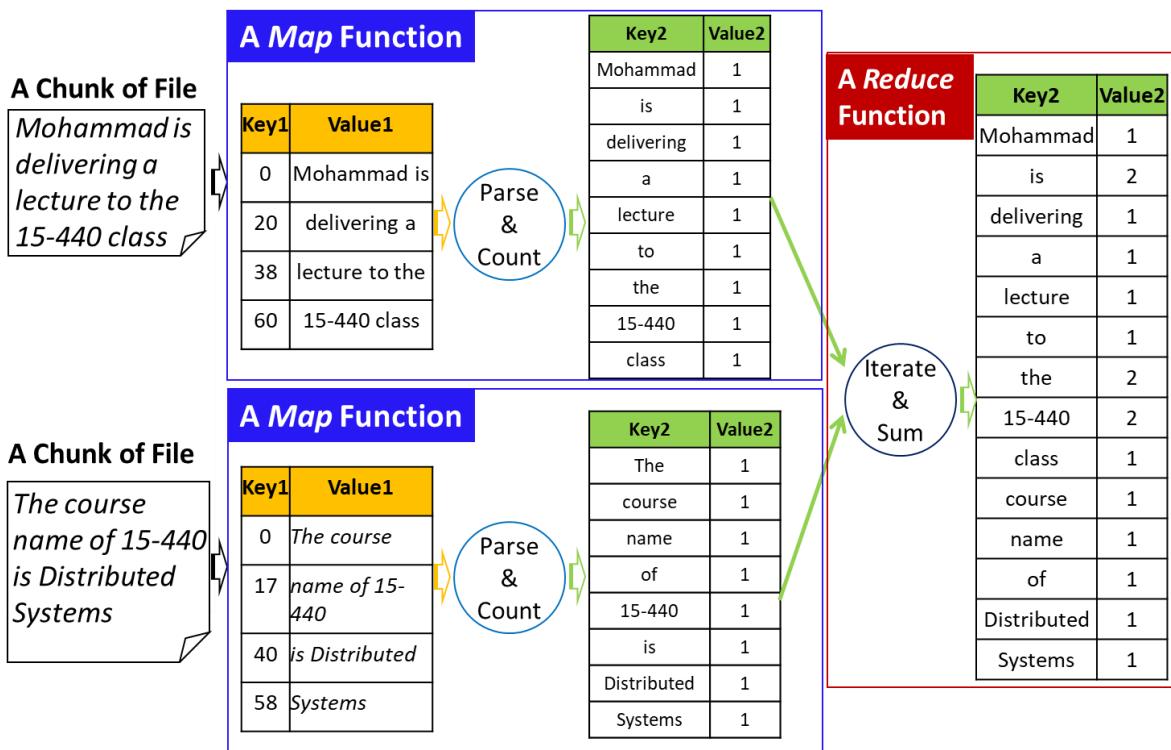
```

135     main()
136
137

```

## PySpark

### 大数据和 Hadoop 编程 - 体会 基于<key,Value>模式的编程 [Spark 处理仍然如此]



## PySpark 编程基础

### Spark?

在没有学习PySpark执行原理之前，很多人可能会认为PySpark编写的程序会被翻译为JVM能识别的字节码，然后提交给Spark运行，其实不是这样的

```

"""
https://www.jianshu.com/p/eaab74e34dae
"""
from pyspark import SparkConf
from pyspark import SparkContext

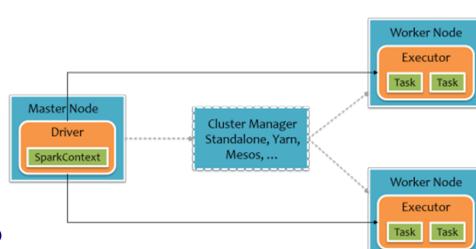
if __name__ == '__main__':
    conf = SparkConf().setAppName("test1").setMaster("local")
    sc = SparkContext(conf = conf)

    rdd = sc.parallelize([1,2,3,4,5,6], 3)

    def add(data):
        return data * 10

    print(rdd.map(add).collect())

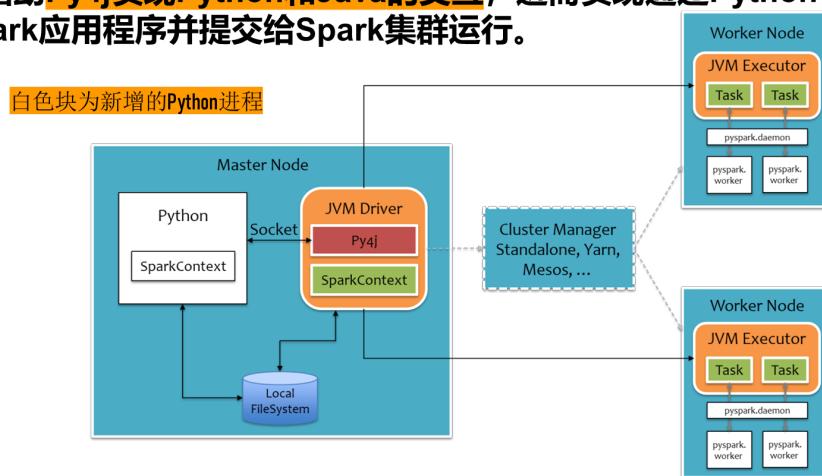
```



- Driver负责总体的调度，Executor负责具体Task的运行，它们都是运行在JVM进程之中的，而这些JVM进程则是可以部署在多种的资源管理系统中的，比如Yarn、Mesos或者是K8s等；
- 用户提交的Spark程序交给Driver进行管理，Driver将程序分解为一个个的Task交给Executor执行。

<https://www.jianshu.com/p/eaab74e34dae>

- 为了不影响现有Spark的工作架构，Spark在外围包装了一层Python的API，借助Py4j实现Python和Java的交互，进而实现通过Python代码来编写Spark应用程序并提交给Spark集群运行。



## 一定要仔细 版本！

- Java, Scala, Hadoop, Spark, Winutils

## 我的笔记本是 Windows 11

- Windows 上安装 Spark，需要 Winutils。所以，要首先注意 Winutils 支持的 Hadoop 版本 <https://github.com/cdarlint/winutils>

<https://github.com/cdarlint/winutils>

hadoop-3.0.2/bin fixed exe and lib 265-312 4 years ago

hadoop-3.1.0/bin fixed exe and lib 265-312 4 years ago

hadoop-3.1.1/bin fixed exe and lib 265-312 4 years ago

hadoop-3.1.2/bin fixed exe and lib 265-312 4 years ago

hadoop-3.2.0/bin fixed exe and lib 265-312 4 years ago

hadoop-3.2.1/bin add 321 winutils 4 years ago

hadoop-3.2.2/bin compile hadoop-3.2.2 2 years ago

README.md compile hadoop-3.2.2 2 years ago

**winutils**

winutils.exe hadoop.dll and hdfs.dll binaries for hadoop on windows

I've been using <https://github.com/steveloughran/winutils> but it stops to update So I tried to compile myself and push binaries here for you all

compile steps (in Chinese)

**how to use**

place a copy of hadoop-ver folder on your local drive set environment vars:

```
HADOOP_HOME=your local hadoop-ver folder  
PATH=%PATH%;%HADOOP_HOME%\bin
```

then you'll pass the "no native library" and "access0" error

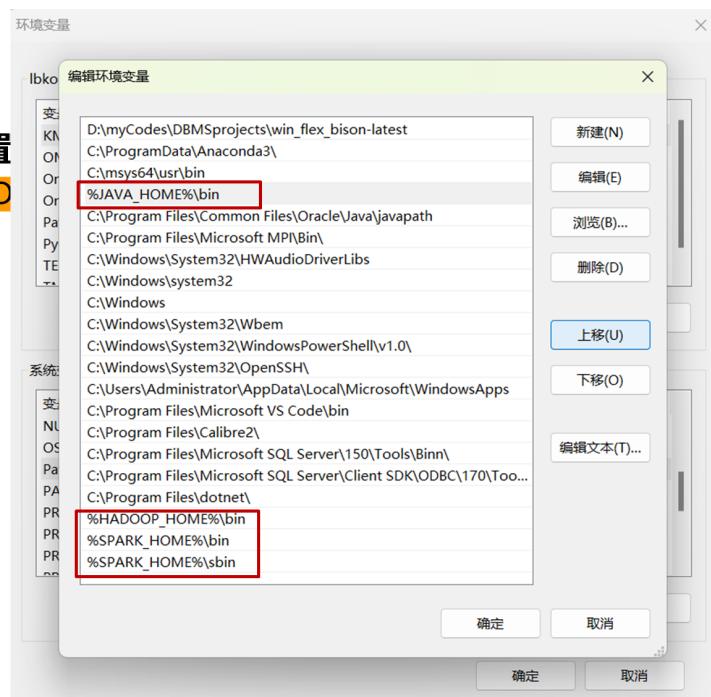
## □ Spark?

- 既然 Winutils 时选择了 Hadoop 3.2.2, 那么, Spark 的版本也就受限了

The screenshot shows the Apache Spark download page. The 'Download Apache Spark™' section has a dropdown menu for 'Choose a package type' set to 'Pre-built for Apache Hadoop 3.2 and later'. Other options like 'Source code' and 'Binary distributions' are also visible. To the right, there's a 'Latest News' sidebar with links to recent releases (Spark 3.3.2, 3.2.3, 3.3.1, 3.2.2) and an 'Archive' link. Below the news is an 'APACHE EVENTS LEARN MORE' button and a large orange 'DOWNLOAD SPARK' button.

## □ Windows 的 环境配置

- JAVA\_HOME, HADOOP\_HOME
- 别忘了 更新 PATH



```
1  ....
2  https://blog.csdn.net/qq_42582489/article/details/106579484
3  ....
4  from pyspark import SparkConf, SparkContext
5
6  # 创建 SparkConf 和 SparkContext
7  conf = SparkConf().setMaster("local[4]").setAppName("lichao-wordcount")
8  sc = SparkContext(conf=conf).getOrCreate()
9
10 # 输入的数据
11 data = ["hello", "world", "hello", "word", "count", "count", "hello"]
12 print(data)
13 print()
```

```

15 def g(x):
16     print(x)
17
18 # 将 Collection 的 data 转化为 spark 中的 rdd 并进行操作
19 rdd = sc.parallelize(data)
20 rdd.foreach(g)
21 print()
22
23 resultRdd = rdd.map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
24
25 # rdd 转为 collecton 并打印
26 resultColl = resultRdd.collect()
27 for line in resultColl:
28     print(line)
29
30 # 结束
31 sc.stop()
32

```

```

"""
https://blog.csdn.net/qq_42582489/article/details/106579484
"""
from pyspark import SparkConf, SparkContext
# 创建 SparkConf 和 SparkContext
conf = SparkConf().setMaster("local[4]").setAppName("PySpark-TestWC")
sc = SparkContext(conf=conf).getOrCreate()

# 输入的数据
data = ["hello", "world", "hello", "word", "count"]
print(data)
print()

def g(x):
    print(x)

# 将 Collection 的 data 转化为 spark 中的 rdd 并进行操作
# rdd = sc.parallelize(data)
# rdd.foreach(g)
# print()

resultRdd = rdd.map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
# rdd 转为 collecton 并打印
resultColl = resultRdd.collect()
for line in resultColl:
    print(line)
# 结束
sc.stop()

```

D:\myCodes\HPCprojects\SourceCodes\pyspark>python PySpark-testWC.py  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
['hello', 'world', 'hello', 'word', 'count', 'count', 'hello']  
(hello 0->  
count  
hello  
world  
hello  
word  
count)  
(world 1->  
('hello', 3)  
('word', 1)  
('world', 1)  
('count', 2))  
(count 2->  
('hello', 3)  
('word', 1)  
('world', 1)  
('count', 2))  
D:\myCodes\HPCprojects\SourceCodes\pyspark>成功: 已终止 PID 29512 (属于 PID 18348 子进程) 的进程。  
成功: 已终止 PID 18348 (属于 PID 11900 子进程) 的进程。  
成功: 已终止 PID 11900 (属于 PID 18184 子进程) 的进程。

```

"""
https://blog.csdn.net/qq_42582489/article/details/106579484
"""
from pyspark import SparkConf, SparkContext
# 创建 SparkConf 和 SparkContext
conf = SparkConf().setMaster("local[4]").setAppName("lichao-wordcount")
sc = SparkContext(conf=conf).getOrCreate()

# 输入的数据
data = ["hello", "world", "hello", "word", "count", "count", "hello"]
print(data)
print()

def g(x):
    print(x)

# 将 Collection 的 data 转化为 spark 中的 rdd 并进行操作
# rdd = sc.parallelize(data)
# addByKey()
# print()

mapRdd = rdd.map(lambda word: (word, 1))
# mapRdd.foreach(g)
# print()

reduceRdd = mapRdd.reduceByKey(lambda a, b: a + b)
# reduceRdd.foreach(g)
# print()

# rdd 转为 collecton 并打印
resultColl = reduceRdd.collect()
for line in resultColl:
    print(line)
# 结束
sc.stop()

```

D:\myCodes\HPCprojects\SourceCodes\pyspark>python PySpark-testWCMR.py  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
['hello', 'world', 'hello', 'word', 'count', 'count', 'hello']  
(count 0->  
hello  
hello  
world  
hello  
word  
count)  
(world 1->  
('count', 1)  
('hello', 1)  
('hello', 1)  
('word', 1)  
('count', 1)  
('world', 1)  
('hello', 1))  
(count 2->  
('hello', 3)  
('word', 1)  
('world', 1)  
('count', 2))  
D:\myCodes\HPCprojects\SourceCodes\pyspark>成功: 已终止 PID 19756 (属于 PID 14864 子进程) 的进程。  
成功: 已终止 PID 14864 (属于 PID 29736 子进程) 的进程。  
成功: 已终止 PID 29736 (属于 PID 36724 子进程) 的进程。

# [Images] PySpark

## 线索

仔细体会 PageRank 在 Spark 上的实现！

### □ Spark 自带的 Example

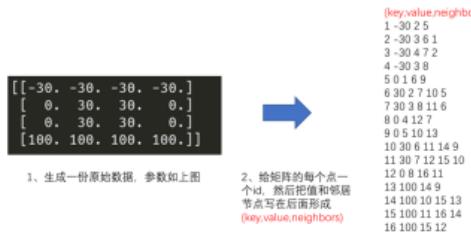
```
python pagerank.py "ml\data\mllib\pagerank_data.txt" 10
WARN: This is a naive implementation of PageRank and is given as an example!
Please refer to PageRank implementation provided by graphx
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
23/04/10 00:33:43 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
23/04/10 00:33:49 WARN BlockManager: Block rdd_13_0 already exists on this machine; not re-adding it
2 has rank: 0.7539975652935547.
3 has rank: 0.7539975652935547.
1 has rank: 1.7380073041193354.
4 has rank: 0.7539975652935547.

D:\spark-3.2.3-bin-hadoop3.2-scala2.13\examples\src\main\python>成功: 已终止 PID 62748 (属于 PID 6820 子进程) 的进程。
成功: 已终止 PID 6820 (属于 PID 27520 子进程) 的进程。
成功: 已终止 PID 27520 (属于 PID 50380 子进程) 的进程。

D:\spark-3.2.3-bin-hadoop3.2-scala2.13\examples\src\main\python>
```

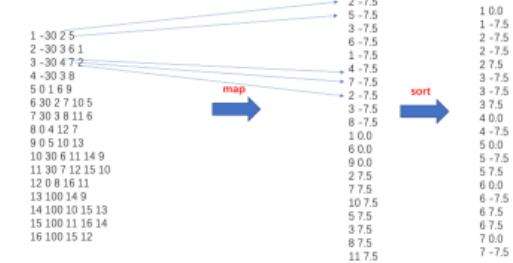
## Hints for Heat Transfer by using Spark

热方程传导的 MapReduce 实现：实现过程(数据预处理)

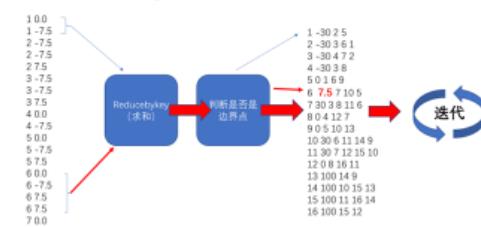


3. 生成结果，保存到文件

热方程传导的 MapReduce 实现：实现过程(Map 过程演示)



热方程传导的 MapReduce 实现：实现过程(Reduce 过程演示)



## [学习 PySpark 的代码]

<https://github.com/spark-examples/pyspark-examples>

Searched: Search or jump to... Pull requests Issues Codespaces Marketplace Explore

spark-examples / pyspark-examples Public

Code Issues 5 Pull requests 4 Actions Projects Security Insights

master 1 branch 0 tags Go to file Add file <> Code

nnkumar13 Merge pull request #6 from wtyos11/fix\_timediff ... 0ae16f1 on Nov 19, 2022 62 commits

resources Add files via upload last year

README.md Update README.md 3 years ago

convert-column-python-list.py PySpark Examples 2 years ago

currentdate.py Pyspark examples new set 3 years ago

data.txt Pyspark examples new set 3 years ago

pandas-pyspark-dataframe.py PySpark Examples 2 years ago

pyspark-add-month.py Pyspark examples new set 3 years ago

pyspark-add-new-column.py PySpark Examples 2 years ago

pyspark-aggregate.py pyspark aggregate 3 years ago

pyspark-array-string.py Update pyspark-array-string.py last year

pyspark-arraytype.py PySpark Examples 2 years ago

pyspark-broadcast-dataframe.py pyspark examples 3 years ago

pyspark-cast-column.py pyspark cast column 3 years ago

pyspark-change-string-double.py PySpark Examples 2 years ago

About Pyspark RDD, DataFrame and Dataset Examples in Python language

Readme 814 stars 32 watching 673 forks Report repository

Releases No releases published

Packages No packages published

Contributors 5

Table of Contents (Spark Examples in Python)

## PySpark Basic Examples

- How to create SparkSession
- PySpark – Accumulator
- PySpark Repartition vs Coalesce
- PySpark Broadcast variables
- PySpark – repartition() vs coalesce()
- PySpark – Parallelize
- PySpark – RDD
- PySpark – Web/Application UI
- PySpark – SparkSession
- PySpark – Cluster Managers
- PySpark – Install on Windows
- PySpark – Modules & Packages
- PySpark – Advantages
- PySpark – Features
- PySpark – What is it? & Who uses it?

## PySpark DataFrame Examples

- PySpark – Create a DataFrame
- PySpark – Create an empty DataFrame
- PySpark – Convert RDD to DataFrame
- PySpark – Convert DataFrame to Pandas

```

1 ****
2 https://stackoverflow.com/questions/36198264/convert-numpy-matrix-into-
3 pyspark-rdd
4
5 mat = np.arange(100).reshape(10, -1)
6 rdd = sc.parallelize(mat)
7
8 np.all(rdd.first() == mat[0])
9 ## True
10
11 ****
12
13
14 from __future__ import print_function
15 import numpy as np
16 import pyspark
17
18 sc = pyspark.SparkContext.getOrCreate()
19 broadcastVar = (0, 9, 0.5, 0.1)

```

```

20
21 def g(x):
22     print(x)
23
24 def func(tup: tuple) -> list:
25     """主要作用是把每个值打上行列index
26     """
27     Array, row_index = tup
28     return [[row_index, col_index, value] for col_index, value in
29     enumerate(Array)]
30
31 def diffuse(lst: list) -> list:
32     global broadcastvar
33     res = []
34
35     if lst[0] > broadcastvar[0] and lst[0] < broadcastvar[1] and lst[1] >
36     broadcastvar[0] and lst[1] < broadcastvar[1]: # 鏡抽複轉±曉闊1倍
37         res.append([lst[0], lst[1], lst[2]*broadcastvar[3]])
38         if lst[0]-1 == broadcastvar[0]:
39             res.append([lst[0]+1, lst[1], lst[2]*broadcastvar[2]])
40         elif lst[0]+1 == broadcastvar[1]:
41             res.append([lst[0]-1, lst[1], lst[2]*broadcastvar[2]])
42         elif lst[1]-1 == broadcastvar[0]:
43             res.append([lst[0], lst[1]+1, lst[2]*broadcastvar[2]])
44         elif lst[1]+1 == broadcastvar[1]:
45             res.append([lst[0], lst[1]-1, lst[2]*broadcastvar[2]])
46         else:
47             res.append([lst[0]+1, lst[1], lst[2]*broadcastvar[2]])
48             res.append([lst[0]-1, lst[1], lst[2]*broadcastvar[2]])
49             res.append([lst[0], lst[1]+1, lst[2]*broadcastvar[2]])
50             res.append([lst[0], lst[1]-1, lst[2]*broadcastvar[2]])
51         else:
52             res.append([lst[0], lst[1], lst[2]])
53             if (lst[0], lst[1]) not in [(broadcastvar[0], broadcastvar[0]),
54             (broadcastvar[0], broadcastvar[1]), (broadcastvar[1], broadcastvar[0]),
55             (broadcastvar[1], broadcastvar[1])]:
56                 if lst[0] == broadcastvar[0]:
57                     res.append([lst[0]+1, lst[1], lst[2]*broadcastvar[2]])
58                 elif lst[0] == broadcastvar[1]:
59                     res.append([lst[0]-1, lst[1], lst[2]*broadcastvar[2]])
60                 elif lst[1] == broadcastvar[0]:
61                     res.append([lst[0], lst[1]+1, lst[2]*broadcastvar[2]])
62                 elif lst[1] == broadcastvar[1]:
63                     res.append([lst[0], lst[1]-1, lst[2]*broadcastvar[2]])
64             return res
65
66 def merge_by_index(tup: tuple):
67     """按row_index 把对应数据放入对应位置
68     """
69     row_index, data_list = tup
70     res = [0]*len(data_list)
71     for col_index, value in data_list:
72         res[col_index] = value
73     return (row_index, np.array(res, dtype = float))
74
75 def evolve(matrixRDD: pyspark.RDD) -> pyspark.RDD:

```

```

72     matrix = matrixRDD.zipWithIndex().flatMap(lambda x:
73         func(x)).flatMap(lambda x:diffuse(x)) \
74             .map(lambda x: ((x[0],x[1]), x[2])).reduceByKey(lambda x,y: x+y)
75     \
76     .map(lambda x: [x[0][0], x[0][1], x[1]]).map(lambda x: (x[0],
77     (x[1],x[2]))).groupByKey().mapValues(list).collect()
78     return matrix
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

```

## [我的代码]

```

1 """
2 基于两个 准备的代码
3
4 "D:\myCodes\HPCbook\05pyspark\sparkTest1NumpyMattoRDD9.py"
5 "D:\myCodes\HPCbook\05pyspark\sparkTest1NumpyMattoRDD10.py"
6
7 """
8
9
10
11 from __future__ import print_function
12 import numpy as np
13 import timeit
14
15 # =====matplotlib 配置 =====
16 import matplotlib
17
18 matplotlib.use('Agg')
19 import matplotlib.pyplot as plt

```

```

20
21 # Set the colormap
22 # plt.rcParams['image.cmap'] = 'BrBG'
23 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
24 plt.cm.coolwarm
25 plt.figure(dpi=300)
26
27 # Set colour interpolation and colour map
28 colorinterpolation = 100
29 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
30
31 # Set Dimension
32 lenX = lenY = 400 #we set it rectangular
33
34 # Set meshgrid
35 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
36
37 # =====参数的配置 =====
38
39 # Basic parameters
40 a = 0.1 # Diffusion constant
41 timesteps = 10000 # Number of time-steps to evolve system
42 image_interval = 1000 # Write frequency for png files
43
44 # Grid spacings
45 dx = 0.01
46 dy = 0.01
47 dx2 = dx ** 2
48 dy2 = dy ** 2
49
50 # For stability, this is the largest interval possible
51 # for the size of the time-step:
52 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
53
54 # Boundary condition
55 Ttop = 100
56 Tbottom = 0
57 Tleft = 0
58 Tright = 30
59
60 # Initial guess of interior grid
61 Tguess = 0
62
63 def init_fields():
64     # Set array size and set the interior value with Tguess
65     field = np.empty((lenX, lenY))
66     field.fill(Tguess)
67
68     # Set Boundary condition
69     field[(lenY-1):, :] = Ttop
70     field[:, :] = Tbottom
71     field[:, (lenX-1):] = Tright
72     field[:, :1] = Tleft
73
74     print("size is ",field.size)
75     print(field,"\\n")

```

```

75     return field
76
77
78 def write_field(field, step):
79     # plt.gca().clear()
80     plt.cla()
81     plt.clf()
82
83
84     # Configure the contour
85     plt.title("Contour of Temperature")
86     plt.contourf(x, Y, field, colorinterpolation, cmap=colourMap)
87     # Set colorbar
88     plt.colorbar()
89     plt.axis('on')
90     plt.savefig('heat_Spark_{0:03d}.png'.format(step))
91
92 # =====
93 import pyspark
94 sc = pyspark.SparkContext.getOrCreate()
95
96 ps = (lenX, lenY, a, dt, dx, dy)
97
98 def g(x):
99     print(x)
100
101 def func(tup: tuple) -> list:
102     """主要作用是把每个值打上行列index
103     """
104     Array, row_index = tup
105     return [[row_index, col_index, value] for col_index, value in
106         enumerate(Array)]
107
108 def merge_by_index(tup: tuple):
109     """按row_index 把对应数据放入对应位置
110     """
111     row_index, data_list = tup
112     res = [0]*len(data_list)
113     for col_index, value in data_list:
114         res[col_index] = value
115     return (row_index, np.array(res, dtype = float))
116
117 internalself = 1.0 - 2.0*a*dt/dx - 2.0*a*dt/dy
118 diff4DirectionsDx = a*dt/dx
119 diff4DirectionsDy = a*dt/dy
120
121 def diffuse(lst: list) -> list:
122     global ps
123     res = []
124
125     if lst[0] > ps[0] and lst[0] < ps[1] and lst[1] > ps[0] and lst[1] <
126     ps[1]: # 即都是内部点
127         res.append([lst[0], lst[1], lst[2]*internalself])
128         if lst[0]-1 == ps[0]:
129             res.append([lst[0]+1, lst[1], lst[2]*diff4DirectionsDx])
130         elif lst[0]+1 == ps[1]:

```

```

129         res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDX])
130     elif lst[1]-1 == ps[0]:
131         res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDY])
132     elif lst[1]+1 == ps[1]:
133         res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDY])
134     else:
135         res.append([lst[0]+1,lst[1], lst[2]*diff4DirectionsDX])
136         res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDX])
137         res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDY])
138         res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDY])
139     else:
140         res.append([lst[0],lst[1], lst[2]])
141         if (lst[0],lst[1]) not in [(0, 0),(0, ps[1]-1),(ps[0]-1,0),
142             (ps[0]-1,ps[1]-1)]:
143             if lst[0] == 0:
144                 res.append([lst[0]+1,lst[1], lst[2]*diff4DirectionsDX])
145             elif lst[0] == ps[0]-1:
146                 res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDX])
147             elif lst[1] == 0:
148                 res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDY])
149             elif lst[1] == ps[1]-1:
150                 res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDY])
151
152     return res
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
def evolve(matrixRDD: pyspark.RDD) -> pyspark.RDD:
    matrix = matrixRDD.zipWithIndex().flatMap(lambda x:
func(x)).flatMap(lambda x: diffuse(x)) \
    .map(lambda x: ((x[0],x[1]), x[2])).reduceByKey(lambda x,y: x+y)
\ \
    .map(lambda x: [x[0][0], x[0][1], x[1]]).map(lambda x: (x[0],
(x[1],x[2]))).groupByKey().mapValues(list).collect()
    return matrix
def main():
    mat = init_fields()
    write_field(mat, 0)

    rdd = sc.parallelize(mat)

    # rdd.foreach(g)
    # print()
    starting_time = timeit.default_timer()
    for m in range(5):
        # for m in range(1, timesteps + 1):
            matrix = evolve(rdd)
            rdd6 = sc.parallelize(matrix)
            rdd7 = np.asarray(rdd6.map(lambda
x:merge_by_index(x)).sortBy(lambda x: x[0], ascending = True) \
                .map(lambda x: x[1]).collect())
            # if m % image_interval == 0:
            if m % 2 == 0:
                write_field(rdd7, m)

            rdd = sc.parallelize(rdd7)
            # rdd.foreach(g)

```

```

180         # print()
181
182     # print("Iteration finished")
183     print("Iteration finished. {} Seconds for Time"
184           difference:.format(timeit.default_timer() - starting_time))
185
186     sc.stop()
187
188 if __name__ == "__main__":
189     main()
190

```

只跑了 5 遍，时间话费就有近乎 290秒！

所以，Spark 还是用在 大数据分析吧！

```

D:\myCodes\HPCbook\05pyspark>python heatEqu2DPySparkImg.py
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
size is 160000
[[ 0.  0.  0. ... 0.  0.  30.]
 [ 0.  0.  0. ... 0.  0.  30.]
 [ 0.  0.  0. ... 0.  0.  30.]
 ...
 [ 0.  0.  0. ... 0.  0.  30.]
 [ 0.  0.  0. ... 0.  0.  30.]
 [ 0. 100. 100. ... 100. 100.  30.]]]

Iteration finished. 286.3016506 Seconds for Time difference:

D:\myCodes\HPCbook\05pyspark>成功：已终止 PID 32076 (属于 PID 25612 子进程)的进程。
成功：已终止 PID 25612 (属于 PID 20016 子进程)的进程。
成功：已终止 PID 20016 (属于 PID 41600 子进程)的进程。

D:\myCodes\HPCbook\05pyspark>

```

## 混合 - MPI+ [based on Stripe]

### MPI+PyCUDA

#### [我的代码]

```

1 """
2 2023年5月3日09:50:52
3
4 贺思超同学完成了 MPI + PyCUDA 的代码示意
5
6 https://github.com/Routhleck/heat_conduction
7
8 "D:\Local\++讲课\HPC.PPTs.2023.wide-Spring\学生的代码\贺思超-MPI+PyUCDA, Python
入门, PyBind\heat_conduction-master.zip"

```

```
9
10
11 """
12
13 from __future__ import print_function
14
15 import getopt
16 import sys
17
18 import numpy as np
19 import time
20 from mpi4py import MPI
21 import pycuda.autoinit
22 import pycuda.driver as drv
23 from pycuda.compiler import SourceModule
24 from pycuda import driver, gpuarray
25
26 import matplotlib
27
28 matplotlib.use('Agg')
29 import matplotlib.pyplot as plt
30 from tqdm import tqdm
31
32
33
34 # 基本参数设置
35
36 # Basic parameters
37 a = 0.5 # Diffusion constant
38 timesteps = 10000 # Number of time-steps to evolve system
39 image_interval = 1000 # write frequency for png files
40
41 # Set Dimension and delta
42 lenX = lenY = 400 # we set it rectangular
43 delta = 1
44
45 # Boundary condition
46 Ttop = 100
47 Tbottom = 0
48 Tleft = 0
49 Tright = 30
50
51 # Initial guess of interior grid
52 Tguess = 0
53
54 # Grid spacings
55 dx = 0.01
56 dy = 0.01
57 dx2 = dx ** 2
58 dy2 = dy ** 2
59
60 # For stability, this is the largest interval possible
61 # for the size of the time-step:
62 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
63
64 # Set colour interpolation and colour map.
```

```

65 # You can try set it to 10, or 100 to see the difference
66 # You can also try: colourMap = plt.cm.coolwarm
67 colorinterpolation = 50
68 colourMap = plt.cm.jet
69
70 # Set meshgrid
71 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
72
73 # device
74 device = 'gpu'
75
76 # benchmark
77 isBenchmark = True
78
79 # MPI globals
80 comm = MPI.COMM_WORLD
81 rank = comm.Get_rank()
82 size = comm.Get_size()
83
84 # Up/down neighbouring MPI ranks
85 up = rank - 1
86 if up < 0:
87     up = MPI.PROC_NULL
88 down = rank + 1
89 if down > size - 1:
90     down = MPI.PROC_NULL
91
92 # Define the kernel function
93 mod = SourceModule("""
94     __global__ void evolve_kernel(double* u, double* u_previous,
95                                     double a, double dt, double dx2, double
96                                     dy2,
97                                     int nx, int ny) {
98         int i = blockIdx.x * blockDim.x + threadIdx.x + 1;
99         int j = blockIdx.y * blockDim.y + threadIdx.y + 1;
100        if (i >= nx - 1 || j >= ny - 1) return;
101        u[i * ny + j] = u_previous[i * ny + j] + a * dt * (
102            u_previous[(i + 1) * ny + j] - 2 * u_previous[i *
103            ny + j] +
104            u_previous[(i - 1) * ny + j]) / dx2 +
105            (u_previous[i * ny + j + 1] - 2 * u_previous[i * ny
106            + j] +
107            u_previous[i * ny + j - 1]) / dy2);
108        u_previous[i * ny + j] = u[i * ny + j];
109    }
110""")
111
112 # Get the kernel function
113 def get_cuda_function():
114     return mod.get_function("evolve_kernel")
115
116 # get CUDA kernel
117 evolve_kernel = get_cuda_function()
118
119 # main calculate function
120 def evolve(u, u_previous, a, dt, dx2, dy2, device):

```

```

118     """Explicit time evolution.
119     u:           new temperature field
120     u_previous: previous field
121     a:           diffusion constant
122     dt:          time step
123     dx2:         grid spacing squared, i.e. dx^2
124     dy2:         -- " " -- , i.e. dy^2"""
125
126     if device == 'cpu':
127         u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
128             (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
129              u_previous[:-2, 1:-1]) / dx2 +
130              (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
131                u_previous[1:-1, :-2]) / dy2)
132         u_previous[:] = u[:]
133     elif device == 'gpu':
134         block_size = (16, 16, 1)
135         grid_size = (int(np.ceil(u.shape[0] / block_size[0])), 
136                       int(np.ceil(u.shape[1] / block_size[1])), 
137                       1)
138
139         # Call the evolve_kernel with the prepared grid and block sizes
140         # import pandas as pd
141         # df = pd.DataFrame(u)
142         # df.to_csv('u.csv')
143         evolve_kernel(driver.InOut(u), driver.InOut(u_previous),
144 np.float64(a), np.float64(dt),
145                                     np.float64(dx2), np.float64(dy2),
146 np.int32(u.shape[0]),
147                                     np.int32(u.shape[1]), block=block_size,
148 grid=grid_size)
149         u_previous[:] = u[:]
150     else:
151         raise ValueError('device should be cpu or gpu')
152
153 # init numpy matrix fields
154 def init_fields():
155     # init
156     field = np.empty((lenX, lenY), dtype=np.float64)
157     field.fill(Tguess)
158     field[(lenY - 1):, :] = Ttop
159     field[:, :1] = Tbottom
160     field[:, (lenX - 1):] = Tright
161     field[:, :1] = Tleft
162     # field = np.loadtxt(filename)
163     field0 = field.copy() # Array for field of previous time step
164     return field, field0
165
166
167 # save image
168 def write_field(field, step):
169     plt.gca().clear()
170     # Configure the contour
171     plt.title("Contour of Temperature")
172     plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
173     if step == 0:

```

```

171     plt.colorbar()
172     plt.savefig(
173
174         'imgMPIPyCUDA/heat_{thread}_{device}_{shape}_{timesteps}_{step}.png'.format(
175             thread=threadsize, device=device, shape=lenx,
176
177             timesteps=timesteps, step=step))
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222

```

# MPI thread communication between up and down

```

def exchange(field):
    # send down, receive from up
    sbuf = field[-2, :]
    rbuf = field[0, :]
    comm.Sendrecv(sbuf, dest=down, recvbuf=rbuf, source=up)
    # send up, receive from down
    sbuf = field[1, :]
    rbuf = field[-1, :]
    comm.Sendrecv(sbuf, dest=up, recvbuf=rbuf, source=down)

# iteration
def iterate(field, local_field, local_field0, timesteps, image_interval):
    for m in tqdm(range(1, timesteps + 1)):
        exchange(local_field0)
        comm.Barrier()
        evolve(local_field, local_field0, a, dt, dx2, dy2, device)
        if m % image_interval == 0:
            comm.Gather(local_field[1:-1, :], field, root=0)
            comm.Barrier()
            if rank == 0:
                write_field(field, m)

def main():
    # Read and scatter the initial temperature field
    if rank == 0:
        field, field0 = init_fields()
        shape = field.shape
        dtype = field.dtype
        comm.bcast(shape, 0) # broadcast dimensions
        comm.bcast(dtype, 0) # broadcast data type
    else:
        field = None
        shape = comm.bcast(None, 0)
        dtype = comm.bcast(None, 0)
    if shape[0] % size:
        raise ValueError('Number of rows in the temperature field (' \
                         + str(shape[0]) + ') needs to be divisible by the \
number ' \
                         + 'of MPI tasks (' + str(size) + ').')
    n = int(shape[0] / size) # number of rows for each MPI task
    m = shape[1] # number of columns in the field
    buff = np.zeros((n, m), dtype)
    comm.Scatter(field, buff, 0) # scatter the data

```

```

223     local_field = np.zeros((n + 2, m), dtype) # need two ghost rows!
224     local_field[1:-1, :] = buff # copy data to non-ghost rows
225     local_field0 = np.zeros_like(local_field) # array for previous time
226     step
227
228     # Fix outer boundary ghost layers to account for aperiodicity?
229     if True:
230         if rank == 0:
231             local_field[0, :] = local_field[1, :]
232         if rank == size - 1:
233             local_field[-1, :] = local_field[-2, :]
234     local_field0[:] = local_field[:]
235
236     # Plot/save initial field
237     if rank == 0:
238         write_field(field, 0)
239
240     # Iterate
241     t0 = time.time()
242     iterate(field, local_field, local_field0, timesteps, image_interval)
243     t1 = time.time()
244
245     # Plot/save final field
246     comm.Gather(local_field[1:-1, :], field, root=0)
247     if rank == 0:
248         write_field(field, timesteps)
249         if (isBenchmark):
250             import pandas as pd
251             import os
252             # 若不存在csv文件，则创建，若存在，则追加
253             if not os.path.exists('data.csv'):
254                 df = pd.DataFrame(columns=['MPI_thread', 'device', 'len',
255 'timesteps', 'time'])
256                 df.to_csv('data.csv', index=False)
257                 df = pd.read_csv('data.csv')
258                 df = df.append(
259                     {'MPI_thread': size, 'device': device, 'shape': lenX,
260 'timesteps': timesteps, 'time': t1 - t0},
261                     ignore_index=True)
262                     df.to_csv('data.csv', index=False)
263                     print("Running time: {0}".format(t1 - t0))
264
265     if __name__ == '__main__':
266         opts, _ = getopt.getopt(sys.argv[1:], 'd:t:i:l:', ['device=',
267 'timesteps=', 'image_interval=', 'len='])
268         for opt, arg in opts:
269             if opt in ('-d', '--device'):
270                 device = arg
271             elif opt in ('-t', '--timesteps'):
272                 timesteps = int(arg)
273             elif opt in ('-i', '--image_interval'):
274                 image_interval = int(arg)
275             elif opt in ('-l', '--len'):
276                 lenX = int(arg)
277                 lenY = int(arg)

```

```
275     X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
276
277     main()
```

## MPI+Numba.CUDA

### [我的代码]

```
1 """
2 2023年5月3日09:50:52
3
4 贺思超同学完成了 MPI + PyCUDA 的代码示意
5
6 https://github.com/Routhlecke/heat\_conduction
7
8 "D:\Local\++讲课\HPC.PPTs.2023.Wide-Spring\学生的代码\贺思超-MPI+PyUCDA, Python
入门, PyBind\heat_conduction-master.zip"
9
10 """
11
12 from __future__ import print_function
13
14 import getopt
15 import sys
16
17 import numpy as np
18 import time
19 from mpi4py import MPI
20 from numba import cuda # 从numba调用cuda
21
22 import matplotlib
23
24 matplotlib.use('Agg')
25 import matplotlib.pyplot as plt
26 from tqdm import tqdm
27
28
29
30 # 基本参数设置
31
32 # Basic parameters
33 a = 0.5 # Diffusion constant
34 timesteps = 10000 # Number of time-steps to evolve system
35 image_interval = 1000 # Write frequency for png files
36
37 # Set Dimension and delta
38 lenX = lenY = 400 # we set it rectangular
39 delta = 1
```

```
40
41 # Boundary condition
42 Ttop = 100
43 Tbottom = 0
44 Tleft = 0
45 Tright = 30
46
47 # Initial guess of interior grid
48 Tguess = 0
49
50 # Grid spacings
51 dx = 0.01
52 dy = 0.01
53 dx2 = dx ** 2
54 dy2 = dy ** 2
55
56 # For stability, this is the largest interval possible
57 # for the size of the time-step:
58 dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
59
60 # timesteps = np.int32(timesteps)
61 # image_interval = np.int32(image_interval)
62
63 # 2023年5月8日10:39:11 网上看到的 生成一个元素的np array
64 # a_arr = np.zeros(1, dtype=np.float32) #常数转矩阵
65 # a_arr[0] = a
66 # 其实 a = np.array([0.01],np.float32) 也就行了
67 a_cuda = np.array([a], dtype = np.float32)
68 dx_cuda = np.array([dx], dtype = np.float32)
69 dy_cuda = np.array([dy], dtype = np.float32)
70 dx2_cuda = np.array([dx2], dtype = np.float32)
71 dy2_cuda = np.array([dy2], dtype = np.float32)
72 dt_cuda = np.array([dt], dtype = np.float32)
73
74
75 # Set colour interpolation and colour map.
76 # You can try set it to 10, or 100 to see the difference
77 # You can also try: colourMap = plt.cm.coolwarm
78 colorinterpolation = 50
79 colourMap = plt.cm.jet
80
81 # Set meshgrid
82 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
83
84 # device
85 device = 'gpu'
86
87 # benchmark
88 isBenchmark = True
89
90 # init numpy matrix fields
91 def init_fields():
92     # init
93     field = np.empty((lenX, lenY), dtype=np.float64)
94     field.fill(Tguess)
95     field[(lenY - 1):, :] = Ttop
```

```

96     field[:, :] = Tbottom
97     field[:, (lenx - 1):] = Tright
98     field[:, :1] = Tleft
99     # field = np.loadtxt(filename)
100
101    print("size is ", field.size)
102    print(field, "\n")
103
104    field0 = field.copy() # Array for field of previous time step
105    return field, field0
106
107
108 # save image
109 def write_field(field, step):
110     plt.gca().clear()
111     # Configure the contour
112     plt.title("Contour of Temperature")
113     plt.contourf(x, Y, field, colorinterpolation, cmap=colourMap)
114     if step == 0:
115         plt.colorbar()
116     plt.savefig(
117
118         'imgMPINumbaCUDA/heat_{thread}_{device}_{shape}_{timesteps}_{step}.png'.fo
119         rmat(thread=size, device=device, shape=lenX,
120
121         timesteps=timesteps, step=step))
122
123 # MPI globals
124 comm = MPI.COMM_WORLD
125 rank = comm.Get_rank()
126 size = comm.Get_size()
127
128 # Up/down neighbouring MPI ranks
129 up = rank - 1
130 if up < 0:
131     up = MPI.PROC_NULL
132 down = rank + 1
133 if down > size - 1:
134     down = MPI.PROC_NULL
135
136 # MPI thread communication between up and down
137 def exchange(field):
138     # send down, receive from up
139     sbuf = field[-2, :]
140     rbuf = field[0, :]
141     comm.Sendrecv(sbuf, dest=down, recvbuf=rbuf, source=up)
142     # send up, receive from down
143     sbuf = field[1, :]
144     rbuf = field[-1, :]
145     comm.Sendrecv(sbuf, dest=up, recvbuf=rbuf, source=down)
146
147 @cuda.jit
148 def evolve_kernel(u, u_previous, a, dt, dx2, dy2, xlen, ylen):
149     """Explicit time evolution.
150         u:           new temperature field
151         u_previous: previous field

```

```

149     a:           diffusion constant
150     dt:          time step
151     dx2:         grid spacing squared, i.e.  $dx^2$ 
152     dy2:         -- "" -- , i.e.  $dy^2$ """
153
154     i = cuda.blockIdx.x * cuda.blockDim.x + cuda.threadIdx.x
155     j = cuda.blockIdx.y * cuda.blockDim.y + cuda.threadIdx.y
156
157     if (i > 0 and j > 0 and i < xlen[0]-1 and j < ylen[0]-1):
158         u[i][j] = u_previous[i][j] + a[0] * dt[0] * ((u_previous[i+1][j] -
2.0 * u_previous[i][j] + u_previous[i-1][j]) / dx2[0] +
(u_previous[i][j+1] - 2.0 * u_previous[i][j] + u_previous[i][j-1]) / dy2[0])
159
160
161 # main calculate function
162 def evolve(u, u_previous, a, dt, dx2, dy2, device):
163     if device == 'cpu':
164         u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
165             (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
166             u_previous[:-2, 1:-1]) / dx2 +
167             (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
168             u_previous[1:-1, :-2]) / dy2)
169         u_previous[:] = u[:]
170     elif device == 'gpu':
171         block_size = (32, 32, 1)
172         grid_size = (int(np.ceil(u.shape[0] / block_size[0])), 
173                     int(np.ceil(u.shape[1] / block_size[1])), 
174                     1)
175
176         u_device = cuda.to_device(u)
177         u_previous_device = cuda.to_device(u_previous)
178         a_device = cuda.to_device(a_cuda)
179         dt_device = cuda.to_device(dt_cuda)
180         dx2_device = cuda.to_device(dx2_cuda)
181         dy2_device = cuda.to_device(dy2_cuda)
182
183         u_lenX_cuda = np.array([u.shape[0]], dtype = np.int32)
184         u_lenY_cuda = np.array([u.shape[1]], dtype = np.int32)
185
186         u_lenX_device = cuda.to_device(u_lenX_cuda)
187         u_lenY_device = cuda.to_device(u_lenY_cuda)
188
189         cuda.synchronize()
190         evolve_kernel[grid_size,block_size](u_device, u_previous_device,
a_device, dt_device,
191                                         dx2_device, dy2_device, u_lenX_device,
192                                         u_lenY_device)
193         cuda.synchronize()
194
195         u_previous = u_device.copy_to_host()
196         u = u_previous_device.copy_to_host()
197         return u, u_previous
198     else:
199         raise ValueError('device should be cpu or gpu')
200
201

```

```

202
203 # iteration
204 def iterate(field, local_field, local_field0, timesteps, image_interval):
205     for m in tqdm(range(1, timesteps + 1)):
206         exchange(local_field0)
207         comm.Barrier()
208         local_field, local_field0 = evolve(local_field, local_field0, a,
209         dt, dx2, dy2, device)
210         if m % image_interval == 0:
211             comm.Gather(local_field[1:-1, :], field, root=0)
212             comm.Barrier()
213             if rank == 0:
214                 write_field(field, m)
215
216 def main():
217     # Read and scatter the initial temperature field
218     if rank == 0:
219         field, field0 = init_fields()
220         shape = field.shape
221         dtype = field.dtype
222         comm.bcast(shape, 0) # broadcast dimensions
223         comm.bcast(dtype, 0) # broadcast data type
224     else:
225         field = None
226         shape = comm.bcast(None, 0)
227         dtype = comm.bcast(None, 0)
228     if shape[0] % size:
229         raise ValueError('Number of rows in the temperature field (' \
230                         + str(shape[0]) + ') needs to be divisible by the \
231                         number ' \
232                         + 'of MPI tasks (' + str(size) + ').')
233
234     n = int(shape[0] / size) # number of rows for each MPI task
235     m = shape[1] # number of columns in the field
236     buff = np.zeros((n, m), dtype)
237     comm.Scatter(field, buff, 0) # scatter the data
238     local_field = np.zeros((n + 2, m), dtype) # need two ghost rows!
239     local_field[1:-1, :] = buff # copy data to non-ghost rows
240     local_field0 = np.zeros_like(local_field) # array for previous time
241     step
242
243     # Fix outer boundary ghost layers to account for aperiodicity?
244     if True:
245         if rank == 0:
246             local_field[0, :] = local_field[1, :]
247             if rank == size - 1:
248                 local_field[-1, :] = local_field[-2, :]
249             local_field0[:] = local_field[:]
250
251     # Plot/save initial field
252     if rank == 0:
253         write_field(field, 0)
254
255     # Iterate
256     t0 = time.time()
257     iterate(field, local_field, local_field0, timesteps, image_interval)

```

```

255     t1 = time.time()
256
257     # Plot/save final field
258     comm.Gather(local_field[1:-1, :], field, root=0)
259     if rank == 0:
260         # write_field(field, timesteps)
261         if (isBenchmark):
262             import pandas as pd
263             import os
264             # 若不存在csv文件，则创建，若存在，则追加
265             if not os.path.exists('data.csv'):
266                 df = pd.DataFrame(columns=['MPI_thread', 'device', 'len',
267 'timesteps', 'time'])
267                 df.to_csv('data.csv', index=False)
268                 df = pd.read_csv('data.csv')
269                 df = df.append(
270                     {'MPI_thread': size, 'device': device, 'shape': lenX,
271 'timesteps': timesteps, 'time': t1 - t0},
272                     ignore_index=True)
273                 df.to_csv('data.csv', index=False)
274                 print("Running time: {0}".format(t1 - t0))
275
276     if __name__ == '__main__':
277         opts, _ = getopt.getopt(sys.argv[1:], 'd:t:i:l:', ['device=',
278 'timesteps=', 'image_interval=', 'len='])
279         for opt, arg in opts:
280             if opt in ('-d', '--device'):
281                 device = arg
282             elif opt in ('-t', '--timesteps'):
283                 timesteps = int(arg)
284             elif opt in ('-i', '--image_interval'):
285                 image_interval = int(arg)
286             elif opt in ('-l', '--len'):
287                 lenX = int(arg)
288                 lenY = int(arg)
289             X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
290         main()

```

## HPPython by C/C++ - High Performance Python

---

### Cython

"D:\Local++写书\18 高性能计算\HPCBook-MD\13-High Performance Python by C-C++.md"

近来用Python用的越来越多，对这种十分灵活的动态语言的哲学也有了较深的理解。虽然Python有不少缺点，如没有强类型，GIL全局锁，没有编译因此效率底下，但正因为其动态性，它非常适合写软件的Prototype。因此一种典型而高效的工作流程是**先用Python调用各种轮子，快速实现软件原型，然后再优化代码，将稳定的部分用C++或其他编译语言进行重写，变成一个供调用的库。**

在这个过程中就难免碰到Python与C++相互调用的问题，尤其体现在：

1. 在原型开发阶段，如何将现有的C++库封装成Python能够方便调用的库，避免重造轮子
2. 当原型开发结束后，如何将Python代码高效地转换成其他语言的代码

能够完成Python与C/C++相互操作的方式有很多，可以参见[知乎的这篇专栏](#)，但是本文介绍的Cython却鲜有详细或通俗的中文资料。Cython是个易用的Python扩展，在Anaconda等发行包里面都自带了，可以用Python的语法写出Python的C语言扩展。

从语言层面来说，Cython是一种拓展的Python，其文件的扩展名为`.pyx`。这种类型的文件通过编译之后可以变成供Python直接调用的动态链接库（Linux/Mac下是`.so`，Windows下是`.pyd`）。

根据官方文档，主要如下几编译方式：

1. **(推荐)** 通过`setup.py`中调用`cython.Build`进行编译
2. 使用`pyximport`调用`.pyx`文件，这种方法`.pyx`文件相当于普通的`.py`文件
3. 在命令行使用`cython`命令从`.pyx`文件生成`.c`文件，再使用外部编译器将`.c`文件编译成Python可用的库
4. 使用`Jupyter Notebook`或者`Sage Notebook`直接运行Cython代码

这上面四种方法里**最简单的是第三种方法**。

运行`cythonize -i <.pyx File>`即可编译`.pyx`成二进制库，并保存在与`.pyx`文件相同的目录下。`cythonize`命令有其他的参数，可以通过命令行查看。这个命令也可以通过`python -m Cython.Build.Cythonize -i <.pyx File>`来完成。

不过推荐使用的是第一种方法，原理也就是通过指定`distutils`或者`setuptools`库中的`ext_modules`参数来编译Cython代码。

## Cython

1) Cython 是一门编程语言，它将 C 和 C++ 的静态类型系统融合在了 Python 身上。Cython 源文件的后缀是`.pyx`，它是 Python 的一个超集，语法是 Python 语法和 C 语法的混血。当然我们说它是 Python 的一个超集，因此你写纯 Python 代码也是可以的。

2) 当我们编写完 Cython 代码时，需要先将 Cython 代码翻译成高效的 C 代码，然后再将 C 代码编译成 Python 的扩展模块。

如上，正因为Cython是一门编程语言，因此不是很推荐学习（笑哭）。

在早期，编写 Python 扩展都是拿 C 去写，但是这对开发者有两个硬性要求：一个是熟悉 C，另一个是要熟悉解释器提供的 C API，这对开发者是一个非常大的挑战。此外，拿 C 编写代码，开发效率也非常低。

而 Cython 的出现则解决了这一点，Cython 和 Python 的语法非常相似，我们只需要编写 Cython 代码，然后再由 Cython 编译器将 Cython 代码翻译成 C 代码即可。所以从这个角度上说，拿 C 写扩展和拿 Cython 写扩展是等价的。

至于如何将 Cython 代码翻译成 C 代码，则依赖于相应的编译器，这个编译器本质上就是 Python 的一个第三方模块。它就相当于是一个翻译官，既然用 C 写扩展是一件痛苦的事情，那就拿 Cython 去写，写完了再帮你翻译成 C。

因此 Cython 的强大之处就在于它将 Python 和 C 结合了起来，可以让你像写 Python 代码一样的同时还可以获得 C 的高效率。所以我们看到 Cython 相当于是高级语言 Python 和低级语言 C 之间的一个融合，因此有人也称 Cython 是 "克里奥尔编程语言" (creole programming language) 。

## 例子1 - Hello

以[官方示例](#)为例，它的 `setup.py` 文件如下

```
1 | from distutils.core import setup
2 | from Cython.Build import cythonize
3 | setup( name = 'Hello world app', ext_modules = cythonize("hello.pyx"), )
```

其中 `hello.pyx` 的代码为

```
1 | def say_hello_to(name):
2 |     print("Hello %s!" % name)
```

编辑保存之后直接运行 `python setup.py build_ext --inplace` 即可进行编译。

其中 `--inplace` 参数可以让对应的链接库生成在源代码所在的目录。

编译中可能遇到的问题有

- `error: command 'cl.exe' failed: No such file or directory:`
  - 这说明在环境中没有找到C编译器。由于我电脑中安装了Visual Studio，我的解决方法是根据平台使用VS的**x86 Native Command Prompt**或**x64 Native Command Prompt**来运行编译命令。
  - 此外还可以选择通过 `python setup.py build_ext --inplace --compiler=mingw32` 使用Anaconda内置的MinGW32编译器，不过这种情况下还可能遇到[内置MinGW32的问题](#)，具体的解决方法在链接里。
- `ModuleNotFoundError: No module named 'cython'`
  - `conda install cython` 即可

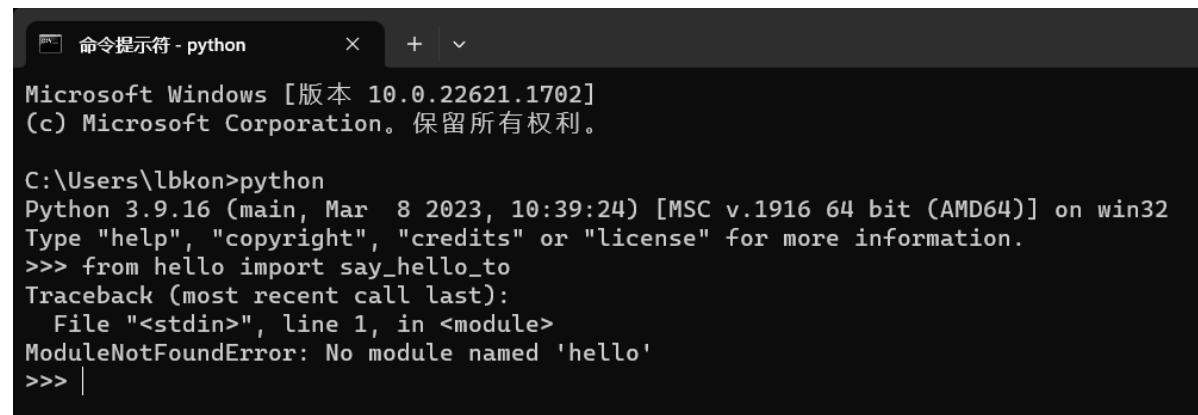
```
D:\myCodes\HPCbook\07HPPython\cython>python setup.py build_ext --inplace
Compiling hello.pyx because it changed.
[1/1] Cythonizing hello.pyx
C:\ProgramData\Anaconda3\lib\site-packages\Cython\Compiler>Main.py:369: FutureWarning: Cython directive 'language_level'
not set, using 2 for now (Py2). This will change in a later release! File: D:\myCodes\HPCbook\07HPPython\cython\hello.p
yx
tree = Parsing.p_module(s, pxd, full_module_name)
hello.c
正在创建库 build\temp.win-amd64-cpython-39\Release\hello.cp39-win_amd64.lib 和对象 build\temp.win-amd64-cpython-39\Re
lease\hello.cp39-win_amd64.exp
正在生成代码
已完成代码的生成

D:\myCodes\HPCbook\07HPPython\cython>
```

名称	修改日期	类型	大小
build	2023,5/24,周三 13:25	文件夹	
hello.c	2023,5/24,周三 13:25	C Source	113 KB
hello.cp39-win_amd64.pyd	2023,5/24,周三 13:25	Python Extensio...	18 KB
hello.pyx	2023,5/24,周三 13:19	PYX 文件	1 KB
setup.py	2023,5/24,周三 13:25	PY 文件	1 KB

编译之后直接通过 `import hello` 或 `from hello import say_hello_to` 即可调用这个编译好的库。

- 还是要在所在目录运行 Python 才行!



```

命令提示符 - python
Microsoft Windows [版本 10.0.22621.1702]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lbkon>python
Python 3.9.16 (main, Mar  8 2023, 10:39:24) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from hello import say_hello_to
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'hello'
>>> |

```

- 在 Explorer 在的所在目录里 `python` 即可

名称	修改日期	类型	大小
build	2023,5/24,周三 13:25	文件夹	
hello.c	2023,5/24,周三 13:25	C Source	113 KB
hello.cp39-win_amd64.pyd	2023,5/24,周三 13:25	Python Extensio...	18 KB
hello.pyx	2023,5/24,周三 13:19	PYX 文件	1 KB
setup.py	2023,5/24,周三 13:25	PY 文件	1 KB

名称	修改日期	类型	大小
build	2023,5/24,周三 13:25	文件夹	
hello.c	2023,5/24,周三 13:25	C Source	113 KB
hello.cp39-win_amd64.pyd	2023,5/24,周三 13:25	Python Extensio...	18 KB
hello.pyx	2023,5/24,周三 13:19	PYX 文件	1 KB
setup.py	2023,5/24,周三 13:25	PY 文件	1 KB

名称	修改日期	类型	大小
build	2023,5/24,周三 13:25	文件夹	
hello.c	2023,5/24,周三 13:25	C Source	113 KB
hello.cp39-win_amd64.pyd	2023,5/24,周三 13:25	Python Extensio...	18 KB
hello.pyx	2023,5/24,周三 13:19	PYX 文件	1 KB
setup.py	2023,5/24,周三 13:25	PY 文件	1 KB

```
C:\ProgramData\Anaconda3\| × + ▾  
Python 3.9.16 (main, Mar 8 2023, 10:39:24) [MSC v.1916 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> from hello import say_hello_to  
>>> say_hello_to  
<built-in function say_hello_to>  
>>> say_hello_to()  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: hello.say_hello_to() takes exactly one argument (0 given)  
>>> say_hello_to('kong')  
Hello kong!  
>>> |
```

## 例子2 - 复杂

如果的Cython工程中有很多 .pyx 文件，甚至有很多 .cpp 文件需要编译，那么这时候最好使用 `setup.py` 进行编译，并使用 `cython.Build.cythonize` 模块。具体编写方式如下：

```
1 | from distutils.core import setup
2 | from distutils.extension import Extension
3 | from Cython.Build import cythonize
4 | extensions = [Extension("Module Name", ["file1.pyx", "file2.cpp", ...],
5 |                         include_dirs = [...],
6 |                         libraries = [...],
7 |                         library_dirs = [...]),
8 |                 Extension("Module 2", ...)]
9 | setup(      name = "XXX",      ext_modules = cythonize(extensions), )
```

最后的 `setup` 部分还有另一种写法：

```
1 | setup(      name = "XXX",      ext_modules = extensions,      cmdclass={'build_ext': Cython.Build.build_ext} )
```

这样的写法可以让安装整个库的时候一起执行掉Cython代码的编译和安装。此外，如果要在编译 .pyx 和 .cpp 时指定语言或者编译参数，在 `Extension` 类的构造函数中添加合适的参数即可。

## PyBind

"D:\Local++\写书\18 高性能计算\HPCBook-MD\13-High Performance Python by C-C++.md"

### [2.2 采用pybind11创建C++代码的Python接口](#)

#### 概述

Pybind11 是一个轻量级的 C++ 库，用于将你的 C++ 代码暴露给 Python 调用（反之也可，但主要还是前者）。

Pybind11 借鉴了 `Boost::Python` 库的设计，但使用了更为简洁的实现方式，使用了大量 C++11 的新特性，更易于使用。

官方文档：<https://pybind11.readthedocs.io/en/stable/index.html>

目前市面上大部分AI计算框架,如TensorFlow、Pytorch、阿里X-Deep Learning、百度PaddlePaddle等,均使用pybind11来提供C++到Python端接口封装。

Pytorch pybind extension文档地址

[https://pytorch.org/tutorials/advanced/cpp\\_extension.html](https://pytorch.org/tutorials/advanced/cpp_extension.html)

## 例子1

```
1 | conda install pybind11
```

```
(base) C:\Windows\System32>conda install pybind11
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- pybind11

The following packages will be downloaded:

package          build
pybind11-2.10.4    py39h59b6b97_0      241 KB defaults
pybind11-global-2.10.4  py39h59b6b97_0     174 KB defaults
                                         Total:        415 KB

The following NEW packages will be INSTALLED:

pybind11      anaconda/pkgs/main/win-64::pybind11-2.10.4-py39h59b6b97_0
pybind11-global  anaconda/pkgs/main/win-64::pybind11-global-2.10.4-py39h59b6b97_0

Proceed ([y]/n)?
```

首先创建c++文件pybind11\_sum.cpp

```
1 | #include <pybind11/pybind11.h>
2 | #include <pybind11/stl.h>
3 | namespace py = pybind11; // 引入 pybind11/stl.h 后 std::vector会自动与python中
4 | list类型绑定
5 | int sum(const std::vector<int>& buffer) {
6 |     int ret = 0;
7 |     for(auto item: buffer)
8 |         ret += item;
9 |     return ret;
10 | }
11 | // pybind11_sum 这里约定要与文件名相同
12 | PYBIND11_MODULE(pybind11_sum, m) {
13 |     m.doc() = "pybind11 sum plugin"; // optional module docstring
14 |     m.def("sum", &sum, "A function which calculate sum in buffer",
15 |           py::arg("buffer"));
16 | }
```

然后运行shell命令

```
1 | g++ -O3 -Wall -shared -std=c++11 -fPIC $(python3 -m pybind11 --includes)
pybind11_sum.cpp -o pybind11_sum$(python3-config --extension-suffix)
```

测试文件test\_pybind.py

```
1 | import pybind11_sum print(pybind11_sum)
2 | print(dir(pybind11_sum))
3 | print(pybind11_sum.sum([5, 6, 7]))
```

## Pythran

[Pythran简单使用](#)

### 概述

Pythran通过预先编译,将python代码转为C++代码的后缀为 .so 的文件,从而在运行时获得更快的运行速度

```
1 | conda install pythran
```

```
(base) C:\Windows\System32>conda install pythran
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- pythran

The following packages will be downloaded:

          package          build
beniget-0.4.1           pyhd3eb1b0_0
clang-14.0.6             haa95532_1
clang-14-14.0.6          default_h120b48b_1
clangxx-14.0.6           default_h120b48b_1
gast-0.5.3                pyhd3eb1b0_0
libclang-cpp-14.0.6       default_h120b48b_1
pythran-0.12.1            py39hd4e2768_0
xsimd-10.0.0              h59b6b97_0

Total:          150.5 MB

The following NEW packages will be INSTALLED:

beniget      anaconda/pkgs/main/noarch::beniget-0.4.1-pyhd3eb1b0_0
clang        anaconda/pkgs/main/win-64::clang-14.0.6-haa95532_1
clang-14     anaconda/pkgs/main/win-64::clang-14-14.0.6-default_h120b48b_1
clangxx      anaconda/pkgs/main/win-64::clangxx-14.0.6-default_h120b48b_1
gast         anaconda/pkgs/main/noarch::gast-0.5.3-pyhd3eb1b0_0
libclang-cpp anaconda/pkgs/main/win-64::libclang-cpp-14.0.6-default_h120b48b_1
pythran      anaconda/pkgs/main/win-64::pythran-0.12.1-py39hd4e2768_0
xsimd        anaconda/pkgs/main/win-64::xsimd-10.0.0-h59b6b97_0

Proceed ([y]/n)? -
```

### 例子

先编写如下代码 `pythran_use.py`

```
1 | ****
2 | (C) rgc
3 | All rights reserved
4 | create time '2021/1/26 10:50'
5 | Usage:
6 | ****
7 |
```

```

8 def a(b):
9     print('df', b)
10
11 def dprod(l0, l1):
12     a(2)
13     return sum(x*y for x, y in zip(l0, l1))
14
15 # pythran export drop1(int[], int list,str:float dict)
16 def drop1(l0, l1, _dict):
17     a(2)
18     l1.append(l0[-1])
19     print(type(l0))
20     for k, v in _dict.items():
21         print(k, v)
22     return l1

```

命令行中运行如下命令

```
1 | pythran pythran_use.py
```

```
D:\myCodes\HPCbook\07HPPython\pythran>pythran pythran_use.py
WARNING: Exporting function 'drop1' that modifies its List argument. Beware that this argument won't be modified at Python call site
WARNING: Failed to find 'pythran-openblas' package. Please install it or change the compiler.blas setting. Defaulting to 'blas'
D:\myCodes\HPCbook\07HPPython\pythran>
```

如果编译通过 会生成一个 pythran\_use.cpython-36m-darwin.so 文件

2023年5月24日14:29:43 因为我的是 Windows 11, `pythran pythran_use.py` 后没有什么 `.so` 文件

而是在 `python pythran_use_test.py` 后, 生成了相关的文件 `pythran_use.cpython-39.pyc`

:(D:) > myCodes > HPCbook > 07HPPython > pythran

名称	修改日期	类型	大小
__pycache__	2023,5/24,周三 14:29	文件夹	
pythran_use.py	2023,5/24,周三 14:19	PY 文件	1 KB
pythran_use_test.py	2023,5/24,周三 14:28	PY 文件	1 KB

名称	修改日期	类型	大小
pythran_use.cpython-39.pyc	2023,5/24,周三 14:29	PYC 文件	1 KB

在另一个py文件中 `pythran_use_test.py` 代码如下,运行即可:

```

1 """
2 (C) rgc
3 All rights reserved
4 create time '2021/1/26 11:04'
5 Usage:
6 """
7
8 import numpy as np
9

```

```

10  from pythran_use import drop1
11
12  if __name__ == '__main__':
13      l0 = np.array([1, 2, 3])
14      print(type(l0))
15      print(drop1(l0, [4, 5, 6], {'a': float(1), 'b': float(2)}))

```

```

D:\myCodes\HPCbook\07HPPython\pythran>pythran pythran_use.py
WARNING: Exporting function 'drop1' that modifies its List argument. Beware that this argument won't be modified at Python call site
WARNING: Failed to find 'pythran-openblas' package. Please install it or change the compiler.blas setting. Defaulting to 'blas'

D:\myCodes\HPCbook\07HPPython\pythran>python pythran_use_test.py
<class 'numpy.ndarray'>
df 2
<class 'numpy.ndarray'>
a 1.0
b 2.0
[4, 5, 6, 3]

D:\myCodes\HPCbook\07HPPython\pythran>

```

## 小结

- 每次代码改变时 需要从新编译一次
- Pythran的预编译 和 Numba的运行时编译正好相反;
  - Numba第一次运行时进行编译,所以会较慢;
  - 而Pythran将编译放在了 代码运行前; - Windows 是似乎不是这样的!
- Numba 在numpy的大数据量的for循环时 比较特长;
- NumExpr 在涉及到一堆的数学计算时比较特长,但是支持的numpy函数较少;
- Pythran 则适用于 大众情况,且对numpy的函数支持比NumExpr多一些,更适用于 list,dict 等的数据处理 及 numpy操作;
- 具体 使用 Pythran预编译后的性能 如何,需要结合实际代码及数据量使用 timeit等工具进行测评;
- 此包使用时的 坑点很多,使用时需要耐心试错,且 国内 相关参考资料较少;

# 系统级编程

---

## CNN - from scratch with Python

---

### 线索

## Hands-On GPU Programming with Python and CUDA Explore high-performance parallel computing with CUDA by Dr. Brian Tuomanen (z-lib.org) - PyCuda-master\Chapter09

新加卷 (D:) > myCodes > HPCprojects > SourceCodes > PyCuda-master > Chapter09

名称	修改日期	类型	大小
deep_neural_network.py	2020,2/23,周日 3:36	PY 文件	21 KB
iris.data	2020,2/23,周日 3:36	DATA 文件	5 KB

## [PyTorch 的设计与实现](#)

### [你的理解]

## 互联网商务系统

### 盈利

在黏着海量用户的基础上，收益：

- 抽成
- 广告
- 云计算
- 理财

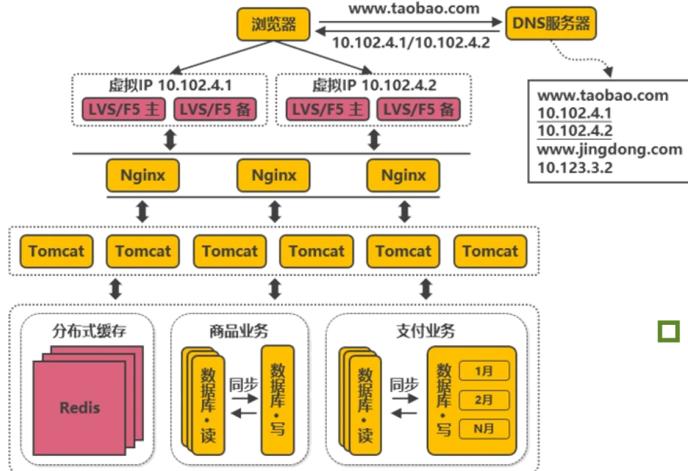
### 关键点

- 黏着用户？ - 褚时健传：利益均沾/大家都能得利才能做大、长久
- 用户多了 - 如何应对秒杀
- 精准营销 - 计算广告

### 秒杀 的应对

仍然是 分而治之 而已 - 只是 现在可以叫 分流

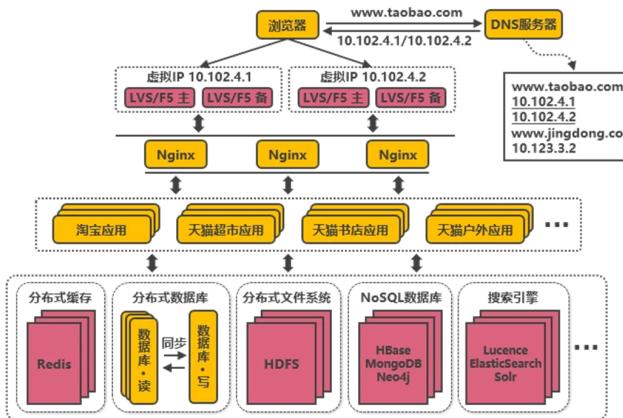
## 9 第八次演进：通过DNS轮询实现机房间的负载均衡



■ 在DNS服务器中可配置一个域名对应多个IP地址，每个IP地址对应到不同的机房里的虚拟IP。当用户访问www.taobao.com时，DNS服务器会使用轮询策略或其他策略，来选择某个IP供用户访问。此方式能实现机房级别的负载均衡，至此，系统可做到机房级别的水平扩展，千万级到亿级的并发量都可通过增加机房来解决，系统入口处的请求并发量不再是问题。

- 随着数据的丰富程度和业务的发展，检索、分析等需求越来越丰富，单单依靠数据库无法解决如此丰富的需求

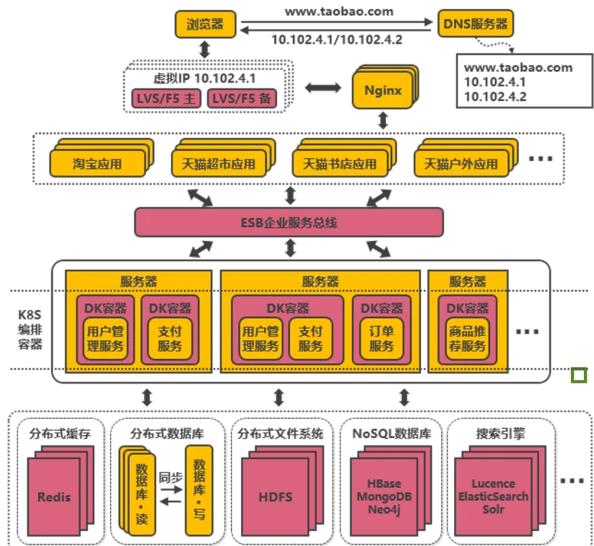
## 11 第十次演进：大应用拆分为小应用



■ 按照业务板块来划分应用代码，使单个应用的职责更清晰，相互之间可以做到独立升级迭代。这时候应用之间可能会涉及到一些公共配置，可以通过分布式配置中心Zookeeper来解决。

- 不同应用之间存在共用的模块，由应用单独管理会导致相同代码存在多份，导致公共功能升级时全部应用代码都要跟着升级

## 14 第十三次演进：引入容器化技术实现运行环境隔离与动态服务管理

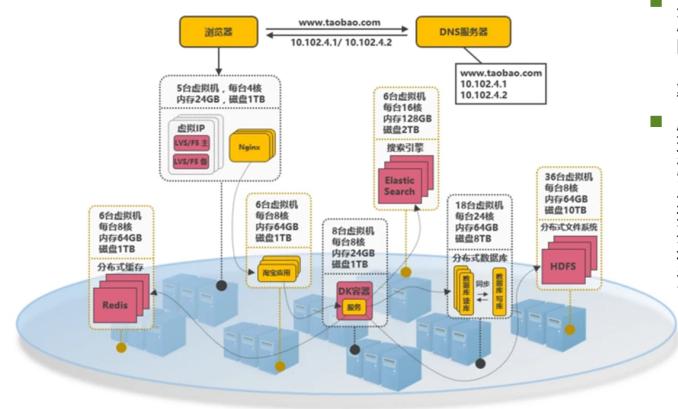


■ 目前最流行的容器化技术是Docker，最流行的容器管理服务是Kubernetes(K8S)，应用/服务可以打包为Docker镜像，通过K8S来动态分发和部署镜像。Docker镜像可理解为一个能运行你的应用/服务的最小的操作系统，里面放着应用/服务的运行代码，运行环境根据实际的需要设置好。把整个“操作系统”打包为一个镜像后，就可以分发到需要部署相关服务的机器上，直接启动Docker镜像就可以把服务起来，使服务的部署和运维变得简单。

■ 在大促之前，可以在现有的机器集群上划分出服务器来启动Docker镜像，增强服务的性能，大促过后就可以关闭镜像，对机器上的其他服务不造成影响（在3.14节之前，服务运行在新增机器上需要修改系统配置来适配服务，这会导致机器上其他服务需要的运行环境被破坏）。

- 使用容器化技术后服务动态扩缩容问题得以解决，但是机器还是需要公司自身来管理，在非大促的时候，还是需要闲置大量的机器资源来应对大促，机器自身成本和运维成本都极高，资源利用率低

## 15 第十四次演进：以云平台承载系统

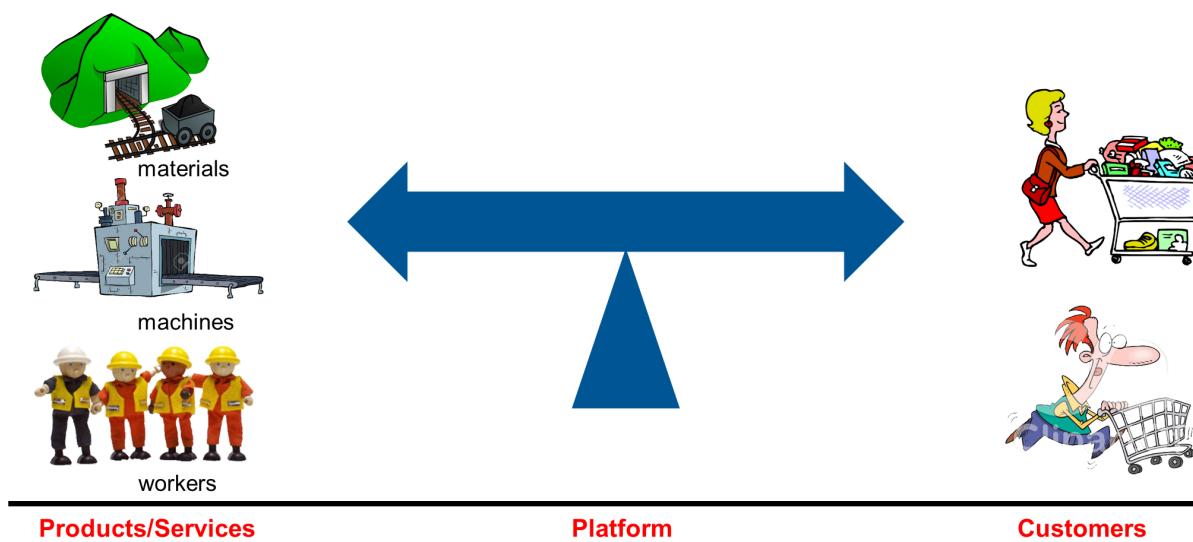


■ 系统可部署到公有云上，利用公有云的海量机器资源，解决动态硬件资源的问题，在大促的时间段里，在云平台中临时申请更多的资源，结合Docker和K8S来快速部署服务，在大促结束后释放资源，真正做到按需付费，资源利用率大大提高，同时大大降低了运维成本。

■ 所谓的云平台，就是把海量机器资源，通过统一的资源管理，抽象为一个资源整体，在之上可按需动态申请硬件资源（如CPU、内存、网络等），并且之上提供通用的操作系统，提供常用的技术组件（如Hadoop技术栈，MPP数据库等）供用户使用，甚至提供开发好的应用，用户不需要关系应用内部使用了什么技术，就能够解决需求（如音视频转码服务、邮件服务、个人博客等）。在云平台中会涉及如下几个概念：

- IaaS：基础设施即服务。对应于上面所说的机器资源统一为资源整体，可动态申请硬件资源的层面；
- PaaS：平台即服务。对应于上面所说的提供常用的技术组件方便系统的开发和维护；
- SaaS：软件即服务。对应于上面所说的提供开发好的应用或服务，按功能或性能要求付费。

## 计算广告

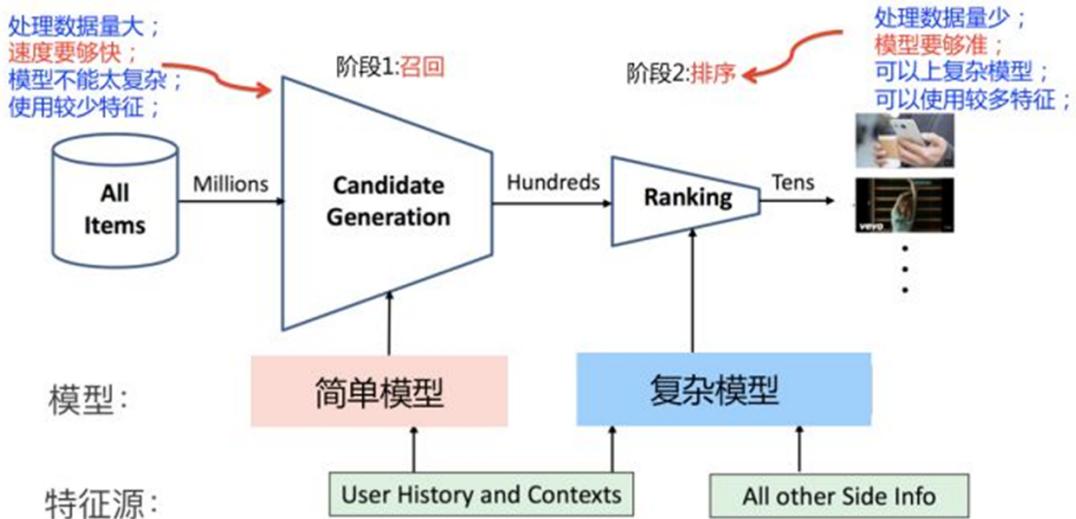


## Platform 挣广告的钱？

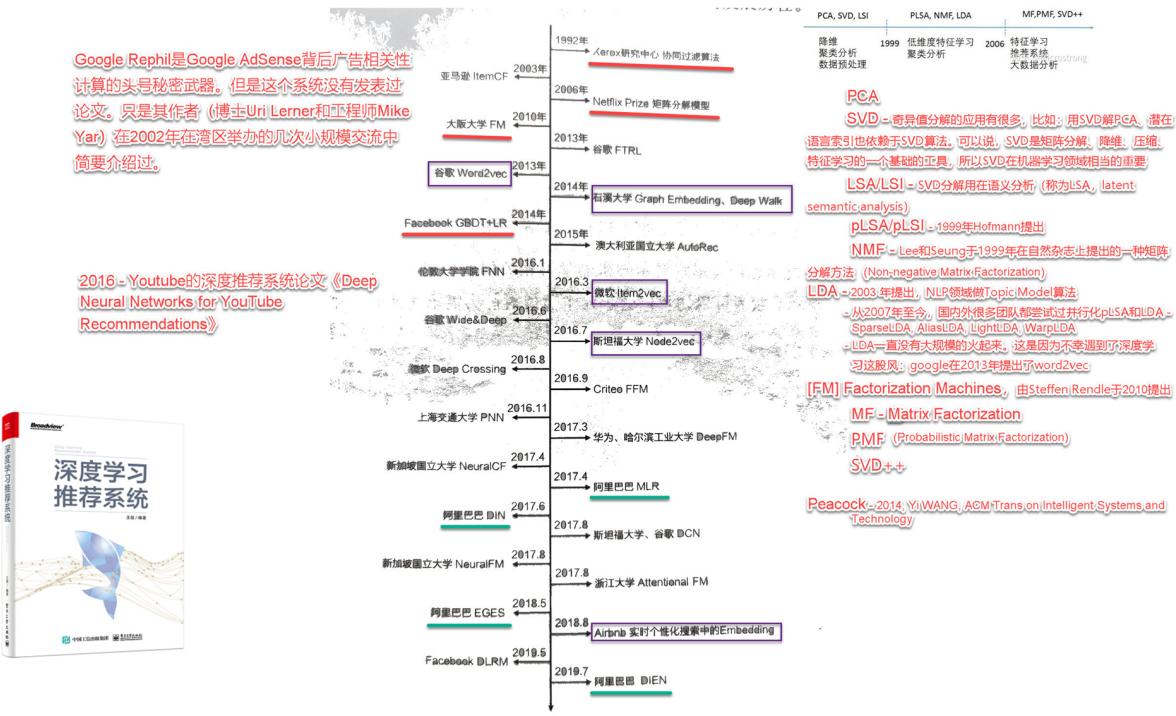
- 巨量的黏着客户 - 是产品/服务提供方的潜在客户
  - 客户买提供方的产品/服务，提供方才能有利润
  - 自然，买的客户越多越好
- 平台因为有巨量的客户，可以做 **掮客** 的角色
  - 因为可以帮助促成交易，自然对提供方有价值
  - 似乎，从提供方收取一定的费用，也就 **理所当然** 了
- 所以，简单的理解
  - 假如一次交易 提供方 可以有(净)利润10块钱，似乎，给平台1块钱，也是可以接受的  
- 也算是抽成吧
- 付费方式?
  - 平台 和 提供方 之间相互选择
  - CPM, ...
- 

**计算广告？ - 尽可能精准找到对产品/服务有潜在购买兴趣的客户**

## 推荐系统在线部分的两个阶段



- 一方面是一堆的产品、服务(的广告)，另一方面是一堆的平台使用者(潜在的客户)
- 当某平台使用者在使用时，将TA**最感兴趣的广告**推送给TA - 最有可能点击
- 处理流程 - Recall + Ranking
  - Recall: 先筛选出 Candidates
  - Ranking: 对每个Candidate 计算该平台使用者(潜在的客户)对该 Candidate 兴趣的大 - 点击的可能性
- Recall + Ranking 都有很多的算法！



## 算法

### Recall 的部分算法

早期的协同过滤, LDA 等

### Ranking 的部分算法

LR, FM, 以及Deep Learning 的许多算法

## CTR 预估算法

### 线索

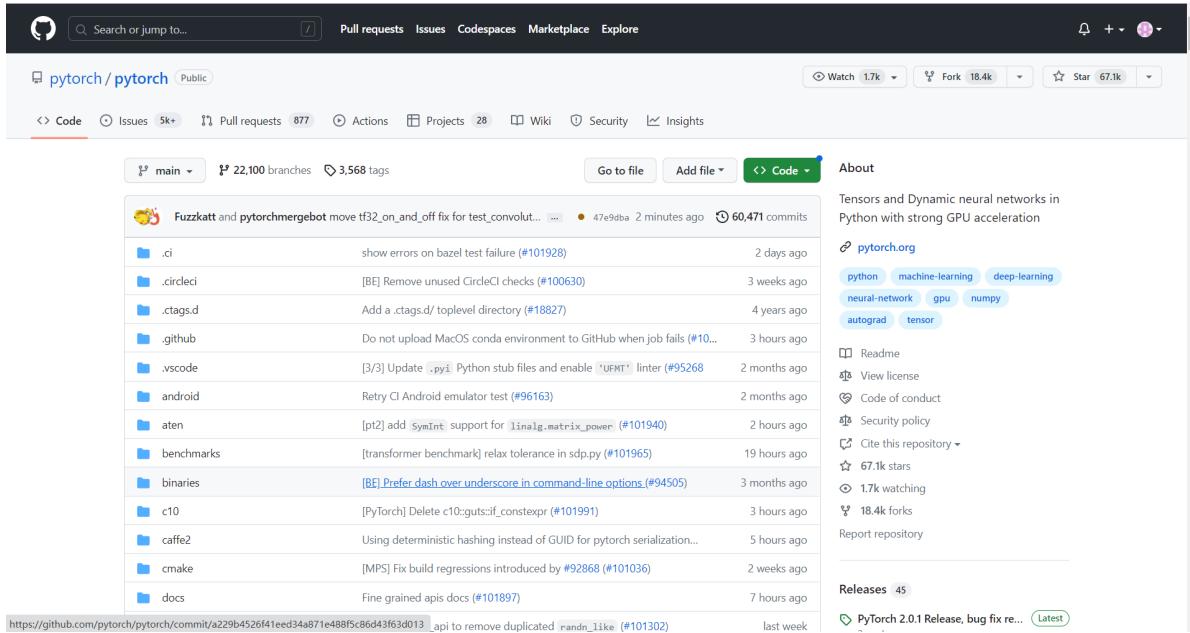
### [代码]

## 例子

## 分布式机器学习

# PyTorch 的设计与实现

<https://github.com/pytorch/pytorch>



The screenshot shows the GitHub repository page for PyTorch. The main branch is 'main' with 22,100 branches and 3,568 tags. The commit history lists several recent changes, including fixes for TensorFlow 2.0 compatibility, CircleCI checks, and various performance and bug fixes across different modules like .ci, .github, and benchmarks. The repository has 60,471 commits and 18.4k forks. The 'About' section highlights PyTorch's focus on Python and GPU acceleration, and lists its primary dependencies: Python, machine-learning, deep-learning, neural-network, gpu, numpy, autograd, and tensor. It also links to the official website, Readme, View license, Code of conduct, Security policy, Cite this repository, and Report repository. The 'Releases' section points to the PyTorch 2.0.1 Release.

PyTorch is not a Python binding into a monolithic C++ framework. It is built to be deeply integrated into Python. You can use it naturally like you would use [NumPy](#) / [SciPy](#) / [scikit-learn](#) etc. **You can write your new neural network layers in Python itself, using your favorite libraries and use packages such as [Cython](#) and [Numba](#).** Our goal is to not reinvent the wheel where appropriate.

We highly recommend installing an [Anaconda](#) environment. You will get a high-quality BLAS library (MKL) and you get controlled dependency versions regardless of your Linux distro.

If you want to compile with CUDA support, install the following (note that CUDA is not supported on macOS)

- [NVIDIA CUDA](#) 11.0 or above
- [NVIDIA cuDNN](#) v7 or above
- [Compiler](#) compatible with CUDA

Note: You could refer to the [cuDNN Support Matrix](#) for cuDNN versions with the various supported CUDA, CUDA driver and NVIDIA hardware

If you want to disable CUDA support, export the environment variable `USE_CUDA=0`. Other potentially useful environment variables may be found in `setup.py`.

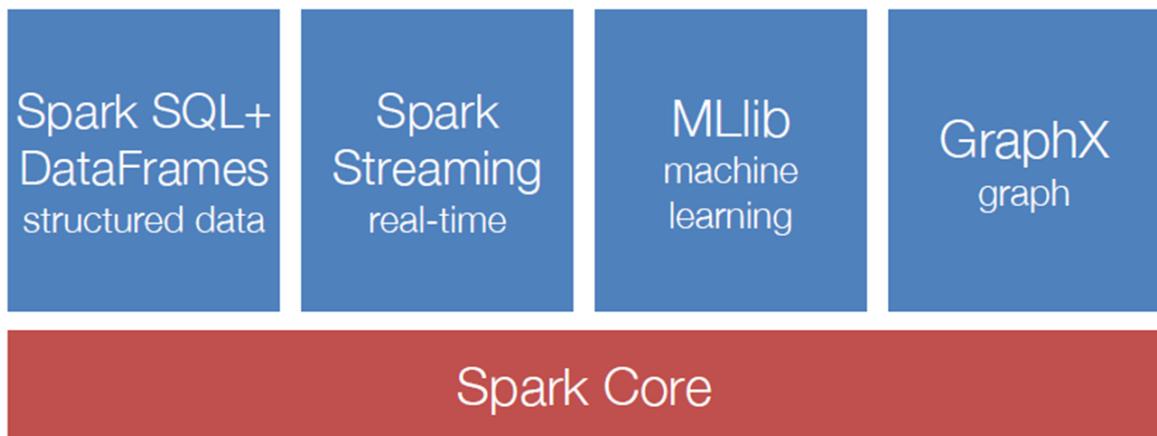
If you are building for NVIDIA's Jetson platforms (Jetson Nano, TX1, TX2, AGX Xavier), Instructions to install PyTorch for Jetson Nano are [available here](#)

If you want to compile with ROCm support, install

- [AMD ROCm](#) 4.0 and above installation
- ROCm is currently supported only for Linux systems.

If you want to disable ROCm support, export the environment variable `USE_ROCM=0`. Other potentially useful environment variables may be found in `setup.py`.

# Spark MLlib



<https://github.com/apache/spark>

This screenshot shows the GitHub repository page for Apache Spark. At the top, there's a navigation bar with links for Search or jump to..., Pull requests, Issues, Codespaces, Marketplace, and Explore. Below the navigation is a header for the repository apache / spark (Public). The main content area shows a list of pull requests from the master branch. Each PR entry includes the author, commit message, date, and a link to the JIRA issue it corresponds to. To the right of the PR list is a sidebar with sections for About, Releases, and Packages.

**About**  
Apache Spark - A unified analytics engine for large-scale data processing  
[spark.apache.org/](http://spark.apache.org/)  
python java r scala sql  
big-data spark jdbc

**Releases**  
201 tags

**Packages** 2

<https://github.com/apache/spark/tree/master/python>

This screenshot shows the GitHub repository page for the Python code of Apache Spark. The left sidebar shows a tree view of the directory structure under the python folder. The right side displays a list of files with their last commit messages and dates. The README.md file is at the bottom of the list.

Name	Last commit message	Last commit date
..		
docs	[SPARK-43599][CONNECT][BUILD] Upgrade buf to v1.19.0	2 days ago
lib	[SPARK-40084][PYTHON] Upgrade Py4J to 0.10.9.7	9 months ago
pyspark	[SPARK-43747][PYTHON][CONNECT] Implement the pyfile support in SparkS...	12 hours ago
test_coverage	[SPARK-36092][INFRA][BUILD][PYTHON] Migrate to GitHub Actions with Co...	2 years ago
test_support	[SPARK-41777][PYSPIKE][ML] Integration testing for TorchDistributor	4 months ago
.coveragerc	[SPARK-7721][PYTHON][TESTS] Adds PySpark coverage generation script	6 years ago
.gitignore	[SPARK-3946] gitignore in /python includes wrong directory	9 years ago
MANIFEST.in	[SPARK-32714][PYTHON] Initial pyspark-stubs port	3 years ago
README.md	[SPARK-36474][PYTHON][DOCS] Mention 'pandas API on Spark' in Spark ov...	2 years ago
mypy.ini	[SPARK-42183][PYTHON][ML][TESTS] Exclude pyspark.ml.torch.tests in My...	4 months ago
run-tests	[SPARK-43347][PYTHON] Remove Python 3.7 Support	3 weeks ago
run-tests-with-coverage	[SPARK-38894][PYTHON][TESTS] Exclude pyspark.cloudpickle in test cove...	last year
run-tests.py	[SPARK-40665][CONNECT] Avoid embedding Spark Connect in the Apache Sp...	7 months ago
setup.cfg	[SPARK-1267][SPARK-18129] Allow PySpark to be pip installed	7 years ago
setup.py	[SPARK-43347][PYTHON] Remove Python 3.7 Support	3 weeks ago
README.md		

# 秒杀

<https://github.com/qqxx6661/miaosha>

qqxx6661 / miaosha Public

Code Issues Pull requests Actions Projects Security Insights

master 3 branches 0 tags Go to file Add file < Code

qqxx6661 Merge pull request #21 from lunasaw/0909\_api\_index ... 791662d on Sep 9, 2022 26 commits

jmeter Merge pull request #21 from lunasaw/0909\_api\_index 8 months ago  
miaosha-dao [Feature]上游依赖使用maven-compiler-plugin 2 years ago  
miaosha-job [Feature]Canal实战 3 years ago  
miaosha-service [Feature]上游依赖使用maven-compiler-plugin 2 years ago  
miaosha-web fix return stock 8 months ago  
src/main/resources [Feature]代码上传 3 years ago  
.gitignore [Feature]代码上传 3 years ago  
LICENSE Initial commit 3 years ago  
README.md Update README.md last year  
miaosha.sql [Feature]接口隐藏+单用户限流 3 years ago  
pom.xml [Feature]代码上传 3 years ago

README.md

从零开始搭建秒杀系统 由浅入深，配合博客入门教程文章食用，风味极佳。

About

从零开始搭建秒杀系统 由浅入深，配合博客入门教程文章食用，风味极佳。

Readme Apache-2.0 license 822 stars 18 watching 164 forks Report repository

Releases No releases published

Packages No packages published

Contributors 5

qqxx6661, 822 stars, 18 watching, 164 forks

<https://github.com/qiurunze123/miaosha>

qiurunze123 / miaosha Public

Code Issues Pull requests 1 Actions Projects Security Insights

master 2 branches 1 tag Go to file Add file < Code

qiurunze123 Delete readme1 5d7ab69 on Oct 18, 2022 248 commits

docs do sth 8 months ago  
miaosha-admin clean code 8 months ago  
miaosha-order 风险项依赖治理 2 years ago  
miaosha-rpc 风险项依赖治理 2 years ago  
miaosha-v1 mybatis版本升级 2 years ago  
miaosha-v2 clean code 8 months ago  
sql 模块化 2 years ago  
.gitattributes Create .gitattributes 4 years ago  
.gitignore 模块化 2 years ago  
CHANGELOG.md 删除无需跟踪文件 2 years ago  
README.md Update README.md 7 months ago  
old.md 提交 aircrafttravel 新版 3 years ago  
pom.xml clean code 8 months ago

About ★★★★☆ 秒杀系统设计与实现\_互联网 工程师进阶与分析 🇨🇳 🇺🇸

Readme 25.9k stars 804 watching 6.7k forks Report repository

Releases 1 tags

Packages No packages published

Contributors 9

qiurunze123, 25.9k stars, 804 watching, 6.7k forks

# 计算广告

# CTR

## DeepCTR-Torch

<https://github.com/shenweichen/DeepCTR-Torch>

The screenshot shows the GitHub repository page for 'shenweichen / DeepCTR-Torch'. The repository is public and has 41 watchers, 610 forks, and 2.4k stars. It contains 7 branches and 13 tags. The 'Code' tab is selected, showing a list of files and their commit history. Key files include .github, deepctr\_torch, docs, examples, tests, .gitattributes, .gitignore, CONTRIBUTING.md, LICENSE, README.md, and setup.py. The repository is described as a PyTorch-based package for CTR models. It includes links to documentation and various GitHub icons.

PyTorch version of [DeepCTR](#).

DeepCTR is a **Easy-to-use**, **Modular** and **Extendible** package of deep-learning based CTR models along with lots of core components layers which can be used to build your own custom model easily. You can use any complex model with `model.fit()` and `model.predict()`. Install through `pip install -U deepctr-torch`.

Let's [Get Started!](#)(Chinese Introduction)

### Models List

Model		Paper
Convolutional Click Prediction Model	2015	[CIKM 2015][A Convolutional Click Prediction Model]( <a href="http://ir.ia.ac.cn/bitstream/173211/12337/1/A_Convolutional_Click_Prediction_Model.pdf">http://ir.ia.ac.cn/bitstream/173211/12337/1/A_Convolutional_Click_Prediction_Model.pdf</a> )
Factorization-supported Neural Network	2016	[ECIR 2016][Deep Learning over Multi-field Categorical Data: A Case Study on User Response Prediction]( <a href="https://arxiv.org/pdf/1601.02376.pdf">https://arxiv.org/pdf/1601.02376.pdf</a> )
Product-based Neural Network	2016	[ICDM 2016][Product-based neural networks for user response prediction]( <a href="https://arxiv.org/pdf/1611.00144.pdf">https://arxiv.org/pdf/1611.00144.pdf</a> )
Wide & Deep	2016	[DLRS 2016][Wide & Deep Learning for Recommender Systems]( <a href="https://arxiv.org/pdf/1606.07792.pdf">https://arxiv.org/pdf/1606.07792.pdf</a> )
DeepFM	2017	[IJCAI 2017][DeepFM: A Factorization-Machine based Neural Network for CTR Prediction]( <a href="http://www.ijcai.org/proceedings/2017/0239.pdf">http://www.ijcai.org/proceedings/2017/0239.pdf</a> )

Model		Paper
Piece-wise Linear Model	2017	[arxiv 2017][Learning Piece-wise Linear Models from Large Scale Data for Ad Click Prediction]( <a href="https://arxiv.org/abs/1704.05194">https://arxiv.org/abs/1704.05194</a> )
Deep & Cross Network	2017	[ADKDD 2017][Deep & Cross Network for Ad Click Predictions]( <a href="https://arxiv.org/abs/1708.05123">https://arxiv.org/abs/1708.05123</a> )
Attentional Factorization Machine	2017	[IJCAI 2017][Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks]( <a href="http://www.ijcai.org/proceedings/2017/435">http://www.ijcai.org/proceedings/2017/435</a> )
Neural Factorization Machine	2017	[SIGIR 2017][Neural Factorization Machines for Sparse Predictive Analytics]( <a href="https://arxiv.org/pdf/1708.05027.pdf">https://arxiv.org/pdf/1708.05027.pdf</a> )
SharedBottom	2017	[arxiv 2017][An Overview of Multi-Task Learning in Deep Neural Networks]( <a href="https://arxiv.org/pdf/1706.05098.pdf">https://arxiv.org/pdf/1706.05098.pdf</a> )
ESMM	2018	[SIGIR 2018][Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate]( <a href="https://dl.acm.org/doi/10.1145/3209978.3210104">https://dl.acm.org/doi/10.1145/3209978.3210104</a> )
MMOE	2018	[KDD 2018][Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts]( <a href="https://dl.acm.org/doi/abs/10.1145/3219819.3220007">https://dl.acm.org/doi/abs/10.1145/3219819.3220007</a> )
xDeepFM	2018	[KDD 2018][xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems]( <a href="https://arxiv.org/pdf/1803.05170.pdf">https://arxiv.org/pdf/1803.05170.pdf</a> )
Deep Interest Network	2018	[KDD 2018][Deep Interest Network for Click-Through Rate Prediction]( <a href="https://arxiv.org/pdf/1706.06978.pdf">https://arxiv.org/pdf/1706.06978.pdf</a> )
Deep Interest Evolution Network	2019	[AAAI 2019][Deep Interest Evolution Network for Click-Through Rate Prediction]( <a href="https://arxiv.org/pdf/1809.03672.pdf">https://arxiv.org/pdf/1809.03672.pdf</a> )
AutoInt	2019	[CIKM 2019][AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks]( <a href="https://arxiv.org/abs/1810.11921">https://arxiv.org/abs/1810.11921</a> )
ONN	2019	[arxiv 2019][Operation-aware Neural Networks for User Response Prediction]( <a href="https://arxiv.org/pdf/1904.12579.pdf">https://arxiv.org/pdf/1904.12579.pdf</a> )
FiBiNET	2019	[RecSys 2019][FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction]( <a href="https://arxiv.org/pdf/1905.09433.pdf">https://arxiv.org/pdf/1905.09433.pdf</a> )
IFM	2019	[IJCAI 2019][An Input-aware Factorization Machine for Sparse Prediction]( <a href="https://www.ijcai.org/Proceedings/2019/0203.pdf">https://www.ijcai.org/Proceedings/2019/0203.pdf</a> )

Model		Paper
DCN V2	2020	[arxiv 2020][DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems]( <a href="https://arxiv.org/abs/2008.13535">https://arxiv.org/abs/2008.13535</a> )
DIFM	2020	[IJCAI 2020][A Dual Input-aware Factorization Machine for CTR Prediction]( <a href="https://www.ijcai.org/Proceedings/2020/04_34.pdf">https://www.ijcai.org/Proceedings/2020/04_34.pdf</a> )
AFN	2020	[AAAI 2020][Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions]( <a href="https://arxiv.org/pdf/1909.03276">https://arxiv.org/pdf/1909.03276</a> )
PLE	2020	[RecSys 2020][Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations]( <a href="https://dl.acm.org/doi/10.1145/338331.3.3412236">https://dl.acm.org/doi/10.1145/338331.3.3412236</a> )

## DisscussionGroup & Related Projects

- [Github Discussions](#)
- Related Projects
  - [AlgoNotes](#)
  - [DeepCTR](#)
  - [DeepMatch](#)
  - [GraphEmbedding](#)

## deep-ctr-prediction [TF]

<https://github.com/qiaoguan/deep-ctr-prediction>

The screenshot shows the GitHub repository page for 'qiaoguan / deep-ctr-prediction'. The repository is public and has 2 branches and 0 tags. The commit history shows several updates from 'guanqiao' on Nov 15, 2019, including updates for AFM, DeepCross, DeepFM, Din, ESMM, Fibinet, ResNet, Transformer, XDeepFM, and official files. The repository has 30 commits in total. On the right side, there is an 'About' section describing it as a CTR prediction models based on deep learning (广告推荐CTR预估模型). It also lists various tags and topics such as afm, transformer, resnet, recommendation, recommendation-algorithms, ctr-prediction, ctr, recommendation-algorithm, factorization-machine, deepfm, xdeepfm, deep-ctr, ad-algorithm, esmm, deep-cross, deep-interest-network, and fibinet. The repository has 883 stars, 22 watching, and 271 forks. There are also sections for 'Readme', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', and 'Insights'.

# FuxiCTR

<https://github.com/xue-pai/FuxiCTR>

The screenshot shows the GitHub repository page for FuxiCTR. At the top, there's a search bar and navigation links for Pull requests, Issues, Codespaces, Marketplace, and Explore. Below that, the repository name 'xue-pai / FuxiCTR' is shown with a public status. On the left, there are tabs for Code, Issues (1), Pull requests (3), Discussions, Actions, Projects (2), Security, and Insights. The main content area shows a list of commits from the 'main' branch, with 64 commits listed. Each commit includes the author, message, date, and time. To the right, there are sections for 'About' (describing it as a curated model zoo for CTR prediction with links to tutorials), 'Releases' (with 10 tags), and 'Packages'. The 'About' section also lists tags: pytorch, recommender-systems, ctr-prediction, ctr, and cvr.

Click-through rate (CTR) prediction is a critical task for many industrial applications such as online advertising, recommender systems, and sponsored search. FuxiCTR provides an open-source library for CTR prediction, with key features in configurability, tunability, and reproducibility. We hope this project could benefit both researchers and practitioners with the goal of open benchmarking for CTR prediction tasks.

## Key Features

- **Configurable:** Both data preprocessing and models are modularized and configurable.
- **Tunable:** Models can be automatically tuned through easy configurations.
- **Reproducible:** All the benchmarks can be easily reproduced.
- **Extensible:** It supports both pytorch and tensorflow models, and can be easily extended to any new models.

## Model Zoo

No	Publication	Model	Paper	Benchmark	Version
			👉 <a href="#">Feature Interaction Models</a>		
1	WWW'07	<a href="#">LR</a>	<a href="#">Predicting Clicks: Estimating the Click-Through Rate for New Ads</a> 🚩 Microsoft	<a href="#">torch</a>	<a href="#">torch</a>
2	ICDM'10	<a href="#">FM</a>	<a href="#">Factorization Machines</a>	<a href="#">torch</a>	<a href="#">torch</a>
3	CIKM'13	<a href="#">DSSM</a>	<a href="#">Learning Deep Structured Semantic Models for Web Search using Clickthrough Data</a> 🚩 Microsoft	<a href="#">torch</a>	<a href="#">torch</a>
4	CIKM'15	<a href="#">CCPM</a>	<a href="#">A Convolutional Click Prediction Model</a>	<a href="#">torch</a>	<a href="#">torch</a>

No	Publication	Model	Paper	Benchmark	Version
5	RecSys'16	<a href="#">FFM</a>	<a href="#">Field-aware Factorization Machines for CTR Prediction</a>  <b>Criteo</b>		<code>torch</code>
6	RecSys'16	<a href="#">DNN</a>	<a href="#">Deep Neural Networks for YouTube Recommendations</a>  <b>Google</b>		<code>torch</code> , <code>tf</code>
7	DLRS'16	<a href="#">Wide&amp;Deep</a>	<a href="#">Wide &amp; Deep Learning for Recommender Systems</a>  <b>Google</b>		<code>torch</code> , <code>tf</code>
8	ICDM'16	<a href="#">IPNN</a>	<a href="#">Product-based Neural Networks for User Response Prediction</a>		<code>torch</code>
9	KDD'16	<a href="#">DeepCrossing</a>	<a href="#">Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features</a>  <b>Microsoft</b>		<code>torch</code>
10	NIPS'16	<a href="#">HOFM</a>	<a href="#">Higher-Order Factorization Machines</a>		<code>torch</code>
11	IJCAI'17	<a href="#">DeepFM</a>	<a href="#">DeepFM: A Factorization-Machine based Neural Network for CTR Prediction</a>  <b>Huawei</b>		<code>torch</code> , <code>tf</code>
12	SIGIR'17	<a href="#">NFM</a>	<a href="#">Neural Factorization Machines for Sparse Predictive Analytics</a>		<code>torch</code>
13	IJCAI'17	<a href="#">AFM</a>	<a href="#">Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks</a>		<code>torch</code>
14	ADKDD'17	<a href="#">DCN</a>	<a href="#">Deep &amp; Cross Network for Ad Click Predictions</a>  <b>Google</b>		<code>torch</code> , <code>tf</code>
15	WWW'18	<a href="#">FwFM</a>	<a href="#">Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising</a>  <b>Oath, TouchPal, LinkedIn, Alibaba</b>		<code>torch</code>
16	KDD'18	<a href="#">xDeepFM</a>	<a href="#">xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems</a>  <b>Microsoft</b>		<code>torch</code>

No	Publication	Model	Paper	Benchmark	Version
17	CIKM'19	<a href="#">FiGNN</a>	<a href="#">FiGNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction</a>		<code>torch</code>
18	CIKM'19	<a href="#">AutoInt/AutoInt+</a>	<a href="#">AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks</a>		<code>torch</code>
19	RecSys'19	<a href="#">FiBiNET</a>	<a href="#">FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction</a> 		<code>torch</code>
20	WWW'19	<a href="#">FGCNN</a>	<a href="#">Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction</a> 		<code>torch</code>
21	AAAI'19	<a href="#">HFM/HFM+</a>	<a href="#">Holographic Factorization Machines for Recommendation</a>		<code>torch</code>
22	Arxiv'19	<a href="#">DLM</a>	<a href="#">Deep Learning Recommendation Model for Personalization and Recommendation Systems</a>		<code>torch</code>
23	NeuralNetworks'20	<a href="#">ONN</a>	<a href="#">Operation-aware Neural Networks for User Response Prediction</a>		<code>torch</code>
24	AAAI'20	<a href="#">AFN/AFN+</a>	<a href="#">Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions</a>		<code>torch</code>
25	AAAI'20	<a href="#">LorentzFM</a>	<a href="#">Learning Feature Interactions with Lorentzian Factorization</a> 		<code>torch</code>
26	WSDM'20	<a href="#">InterHAt</a>	<a href="#">Interpretable Click-through Rate Prediction through Hierarchical Attention</a>		<code>torch</code>
27	DLP-KDD'20	<a href="#">FLEN</a>	<a href="#">FLEN: Leveraging Field for Scalable CTR Prediction</a> 		<code>torch</code>

No	Publication	Model	Paper	Benchmark	Version
28	CIKM'20	<a href="#">DeepIM</a>	<a href="#">Deep Interaction Machine: A Simple but Effective Model for High-order Feature Interactions</a> ➔ <b>Alibaba, RealAI</b>		<code>torch</code>
29	WWW'21	<a href="#">FmFM</a>	<a href="#">FM^2: Field-matrixed Factorization Machines for Recommender Systems</a> ➔ <b>Yahoo</b>		<code>torch</code>
30	WWW'21	<a href="#">DCN-V2</a>	<a href="#">DCN V2: Improved Deep &amp; Cross Network and Practical Lessons for Web-scale Learning to Rank Systems</a> ➔ <b>Google</b>		<code>torch</code>
31	CIKM'21	<a href="#">DESTINE</a>	<a href="#">Disentangled Self-Attentive Neural Networks for Click-Through Rate Prediction</a> ➔ <b>Alibaba</b>		<code>torch</code>
32	CIKM'21	<a href="#">EDCN</a>	<a href="#">Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models</a> ➔ <b>Huawei</b>		<code>torch</code>
33	DLP-KDD'21	<a href="#">MaskNet</a>	<a href="#">MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask</a> ➔ <b>Sina Weibo</b>		<code>torch</code>
34	SIGIR'21	<a href="#">SAM</a>	<a href="#">Looking at CTR Prediction Again: Is Attention All You Need?</a> ➔ <b>BOSS Zhipin</b>		<code>torch</code>
35	KDD'21	<a href="#">AOANet</a>	<a href="#">Architecture and Operation Adaptive Network for Online Recommendations</a> ➔ <b>Didi Chuxing</b>		<code>torch</code>
36	AAAI'23	<a href="#">FinalMLP</a>	<a href="#">FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction</a> ➔ <b>Huawei</b>		<code>torch</code>
37	SIGIR'23	<a href="#">FINAL</a>	<a href="#">FINAL: Factorized Interaction Layer for CTR Prediction</a> ➔ <b>Huawei</b>		<code>torch</code>
			<b>User Behavior Modeling</b>		

No	Publication	Model	Paper	Benchmark	Version
38	KDD'18	DIN	<a href="#">Deep Interest Network for Click-Through Rate Prediction</a>  Alibaba		<code>torch</code>
39	AAAI'19	DIEN	<a href="#">Deep Interest Evolution Network for Click-Through Rate Prediction</a>  Alibaba		<code>torch</code>
40	DLP-KDD'19	BST	<a href="#">Behavior Sequence Transformer for E-commerce Recommendation in Alibaba</a>  Alibaba		<code>torch</code>
41	CIKM'20	DMIN	<a href="#">Deep Multi-Interest Network for Click-through Rate Prediction</a>  Alibaba		<code>torch</code>
42	AAAI'20	DMR	<a href="#">Deep Match to Rank Model for Personalized Click-Through Rate Prediction</a>  Alibaba		<code>torch</code>
43	Arxiv'21	ETA	<a href="#">End-to-End User Behavior Retrieval in Click-Through Rate Prediction Model</a>  Alibaba		<code>torch</code>
44	CIKM'22	SDIM	<a href="#">Sampling Is All You Need on Modeling Long-Term User Behaviors for CTR Prediction</a>  Meituan		<code>torch</code>
			 <b>Multi-Task Models</b>		
45	MachineLearn'97	SharedBottom	<a href="#">Multitask Learning</a>		<code>torch</code>
46	KDD'18	MMoE	<a href="#">Modeling Task Relationships in Multi-task Learning with Multi-Gate Mixture-of-Experts</a>  Google		<code>torch</code>
			 <b>Multi-Domain Models</b>		
47	Arxiv'23	PPNet/PEPNet	<a href="#">PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information</a>  KuaiShou		<code>torch</code>

-  See [reusable dataset splits for CTR prediction](#).
-  See [benchmarking configurations and steps](#).
-  See [the BARS benchmark leaderboard](#).

## Dependencies

FuxiCTR has the following dependency requirements.

- python 3.6+
- pytorch 1.0/1.10+ (required only for torch models)
- tensorflow 2.1+ (required only for tf models)

Other packages can be installed via `pip install -r requirements.txt`.

## Quick Start

### 1. Run the demo examples

Examples are provided in the demo directory to show some basic usage of FuxiCTR.

Users can run the examples for quick start and to understand the workflow.

```
1 | cd demo
2 | python example1_build_dataset_to_h5.py
3 | python example2_DeepFM_with_h5_input.py
```

### 2. Run an existing model

Users can easily run each model in the model zoo following the commands below, which is a demo for running DCN. In addition, users can modify the dataset config and model config files to run on their own datasets or with new hyper-parameters. More details can be found in the [readme file](#).

```
1 | cd model_zoo/DCN/DCN_torch
2 | python run_expid.py --expid DCN_test --gpu 0
3 |
4 | # Change `MODEL` according to the target model name
5 | cd model_zoo/MODEL_PATH
6 | python run_expid.py --expid MODEL_test --gpu 0
```

### 3. Implement a new model

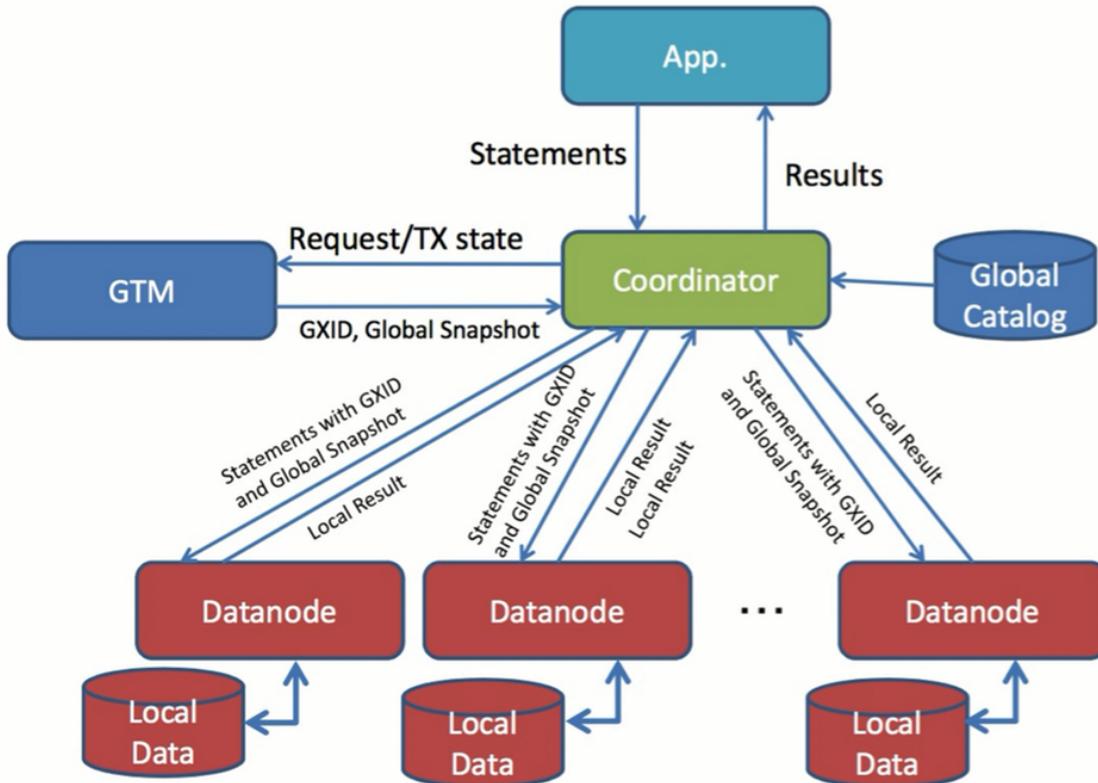
The FuxiCTR code structure is modularized, so that every part can be overwritten by users according to their needs. In many cases, only the model class needs to be implemented for a new customized model. If data preprocessing or data loader is not directly applicable, one can also overwrite a new one through the [core APIs](#). We show a concrete example which implements our new model [FinalMLP](#) that has been recently published in AAAI 2023. More examples can be found in the [model zoo](#).

## Citation

⚠️ If you find our code or benchmarks helpful in your research, please kindly cite the following papers.

Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, Xiuqiang He. [Open Benchmarking for Click-Through Rate Prediction](#). *The 30th ACM International Conference on Information and Knowledge Management (CIKM)*, 2021. [\[Bibtex\]](#)

## 数据库



分布式数据库? 云原生数据库?

[PolarDB-X] - a MySQL branch - Cloud-native DB

<https://github.com/polaradb>

The screenshot shows the GitHub profile for the 'polaradb' organization. The 'polardbx-engine' repository is highlighted. Popular repositories listed include:

- polaradb-sql: Java, 1.3k stars, 274 forks
- polardbx-engine: C++, 320 stars, 98 forks
- polaradb-operator: Go, 72 stars, 25 forks
- polaradb-cdc: Java, 56 stars, 37 forks
- polaradb-glue: Java, 33 stars, 30 forks

On the right, there are sections for 'People' (two profile icons), 'Top languages' (Java, C++, Go, Makefile), 'Most used topics' (mysql, cloud-native, distributed-transactions, high-availability, polaradb-x), and a 'Report abuse' link.

<https://github.com/polaradb/polardbx-engine>

## PolarDB-X Engine

PolarDB-X Engine is a **MySQL** branch originated from Alibaba Group. It is based on the MySQL official release and has many features and performance enhancements, PolarDB-X Engine has proven to be very stable and efficient in production environment. It can be used as a free, fully compatible, enhanced and open source drop-in replacement for MySQL.

PolarDB-X Engine has been an open source project **since October 2021**. It is being actively developed by engineers from Alibaba Group, Everyone is welcomed to get involved.

For further information on PolarDB-X Engine or additional documentation, visit: [PolarDB-X Engine Overview](#)

For further information about x-engine, please ref: [X-Engine Storage Engine](#)

## X Engine Overview

<https://github.com/polaradb/polardbx-engine/wiki/2-X-Engine-Overview>

X-Engine是一个基于**LSM-tree**架构开发的**MySQL事务存储引擎**,由阿里云数据库产品事业部PolarDB新型存储引擎团队开发维护。

目前X-Engine在阿里公共云平台上以两种产品形态对外提供售卖:

- [RDS MySQL X-Engine](#), 该形态以单机MySQL 存储引擎的方式提供服务。
- [PolarDB MySQL X-Engine](#), 该形态以X-Engine一写多读的方式提供服务。

开源版本X-Engine在代码上,与我们RDS MySQL X-Engine产品的代码完全相同。而一写多读版本考虑到线下部署时依赖较多,本次暂未开放源代码。

X-Engine的初衷是想创造一个更好的事务存储引擎,以替代当时在阿里内部业务中被广泛使用的InnoDB引擎。

业务层面,在2016年前后的时间节点,阿里的业务流量在飞速增长,以当时的流量增速趋势,通过在MySQL InnoDB上打patch的小打小闹方式终将难以维继。

技术层面,MySQL官方不可能只为了阿里巴巴的业务需求而定制开发计划,而内核研发团队在InnoDB引擎20W行代码的基础上魔改的代价也过于高昂。

最后的结论是另选一条更合适的技术路线来解决当时的业务问题。最终,我们选择了当时学术界和工业界均有一定研究和积累的的LSM-tree技术架构。

X-Engine的第一个版本的代码源自RocksDB 4.8.1 (2016年7月发布),所以它并不是从第一行代码开始完全重新开发。

选择基于RocksDB的代码作为基础向前演进,符合当时团队的工程能力现状,有一个优秀的蓝本作为参照,我们可以在索引/空间管理/高性能索引/高性能事务处理框架的多个技术方向上,同时开展探索和研究,而不用担心大家各自开发的功能长久不能组装在一起运行。另一方面我们可以复用MyRocks接口将X-Engine适配到MySQL运行,如此团队可以将精力专注在存储引擎核心的能力优化上。X-Engine自2016年之后即开始完全独立演进,截至目前近6年的时间里,绝大部分核心模块均已经被重写,目前除了在存储引擎接口上与RocksDB官方保持一致外,大部分模块的设计理念与RocksDB均都有较大不同。开源之前的代码合规扫描结果显示70%均为自研代码。

X-Engine在阿里内部主要作为MySQL的存储引擎使用,因此我们在MySQL生态下做了大量工作以满足各类型业务的需求。

大部分情况下X-Engine可以作为InnoDB引擎的一个直接替代,并获得3~5倍的存储空间节省,二者的不同可参见[X-Engine与InnoDB的差异](#)。

- X-Engine 支持MySQL 5.7/8.0版本,开源的为8.0版本也是目前内部主要使用的版本,5.7版本已经停止迭代演进。
- 支持[Instant DDL](#)/[Online DDL](#)/[Parallel DDL](#)等DDL优化,提升大表做schema变更时的用户体验。
- 支持完善的统计信息管理,包含表信息(表数据量,表记录数,平均行长),索引信息(索引大小,索引key长度)和索引列分布信息(NDV)等,以辅助优化器选出最优的执行计划。
- 支持通过xtrabackup进行物理备份恢复(此功能此次暂未开源)
- 多种字符集支持,如中文GBK/UTF8MB4等。

PolarDB X-Engine团队除了解决阿里巴巴集团内部和阿里云上出现的技术挑战,也在LSM-tree领域的技术前沿上做了很多的探索和尝试。

我们探索的方向包括新型存储器件/新型定制计算硬件/AI for DB等领域。团队已发表的论文如下:

- SIGMOD'19 [X-Engine: An Optimized Storage Engine for Large-scale E-Commerce Transaction Processing](#)
- VLDB'19 [LB+-Trees: Optimizing Persistent Index Performance on 3DXPoint Memory](#)
- FAST'20 [FPGA-Accelerated Compactions for LSM-based Key-Value Store](#)
- VLDB'20 [Leaper: a learned prefetcher for cache invalidation in LSM-tree based storage engines](#)
- VLDB'21 [Revisiting the Design of LSMtree Based OLTP Storage Engine with Persistent Memory](#)

## PolarDB-X Operator

PolarDB-X Operator is a **Kubernetes extension** that aims to create and manage PolarDB-X cluster on Kubernetes. It follows the [operator pattern](#) and automates the management tasks.

<https://github.com/polardb/polardbx-operator>

**About**  
polaradb-operator is a Kubernetes extension that aims to create and manage PolarDB-X cluster on Kubernetes.

**Releases** 10

**PolarDBX-Operator Release v1.4.1** (Latest) on Apr 12

**Packages**

## [ApsaraDB] - PolarDB for PostgreSQL (hereafter simplified as PolarDB)

<https://github.com/ApsaraDB>

**Pinned**

- PolarDB-for-PostgreSQL (Public)  
A cloud-native database based on PostgreSQL developed by Alibaba Cloud.  
C 2.6k 129 followers
- PolarDB-FileSystem (Public)  
C++ 147 63
- PolarDB-Hands-On (Public)  
C 6 3
- PolarDB-Hackathon-2023 (Public)  
4 1

**Repositories**

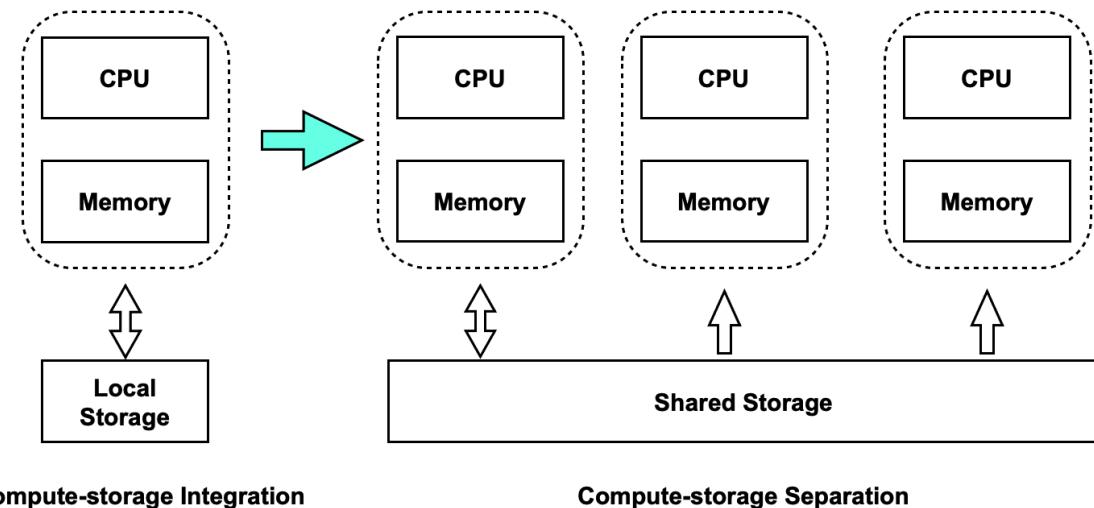
- PolarDB-for-PostgreSQL (Public)  
A cloud-native database based on PostgreSQL developed by Alibaba Cloud.  
C 2,597 Apache-2.0 415 6 (1 issue needs help) 4 Updated yesterday
- tpch-dbgen (Public)  
TPC-H dbgen

**People**

**Top languages**

**Most used topics**

PolarDB for PostgreSQL (hereafter simplified as **PolarDB**) is a cloud native database service independently developed by Alibaba Cloud. This service is 100% compatible with PostgreSQL and uses a shared-storage-based architecture in which computing is decoupled from storage.



Compute-storage Integration

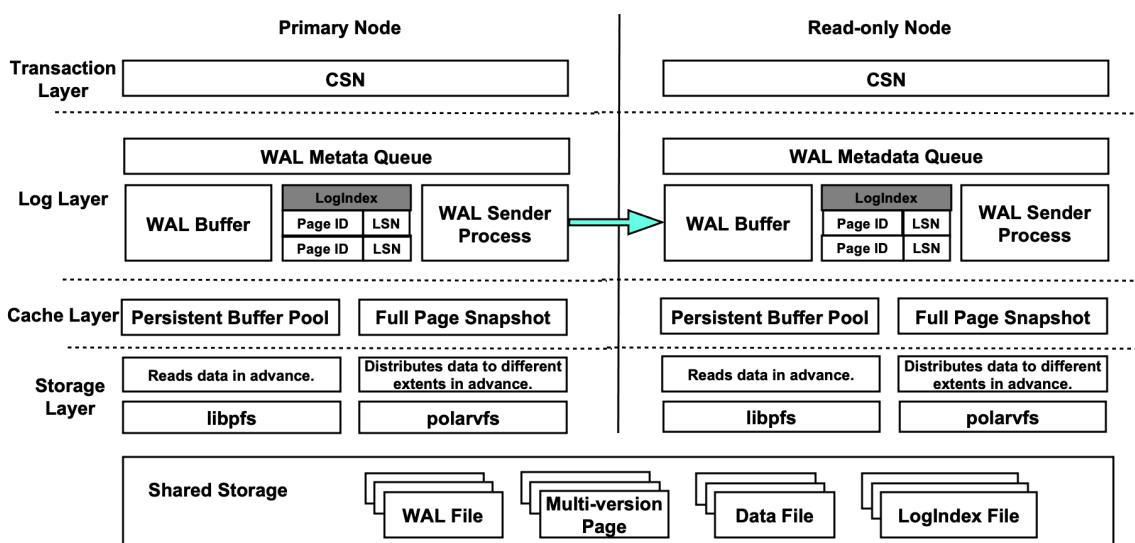
Compute-storage Separation

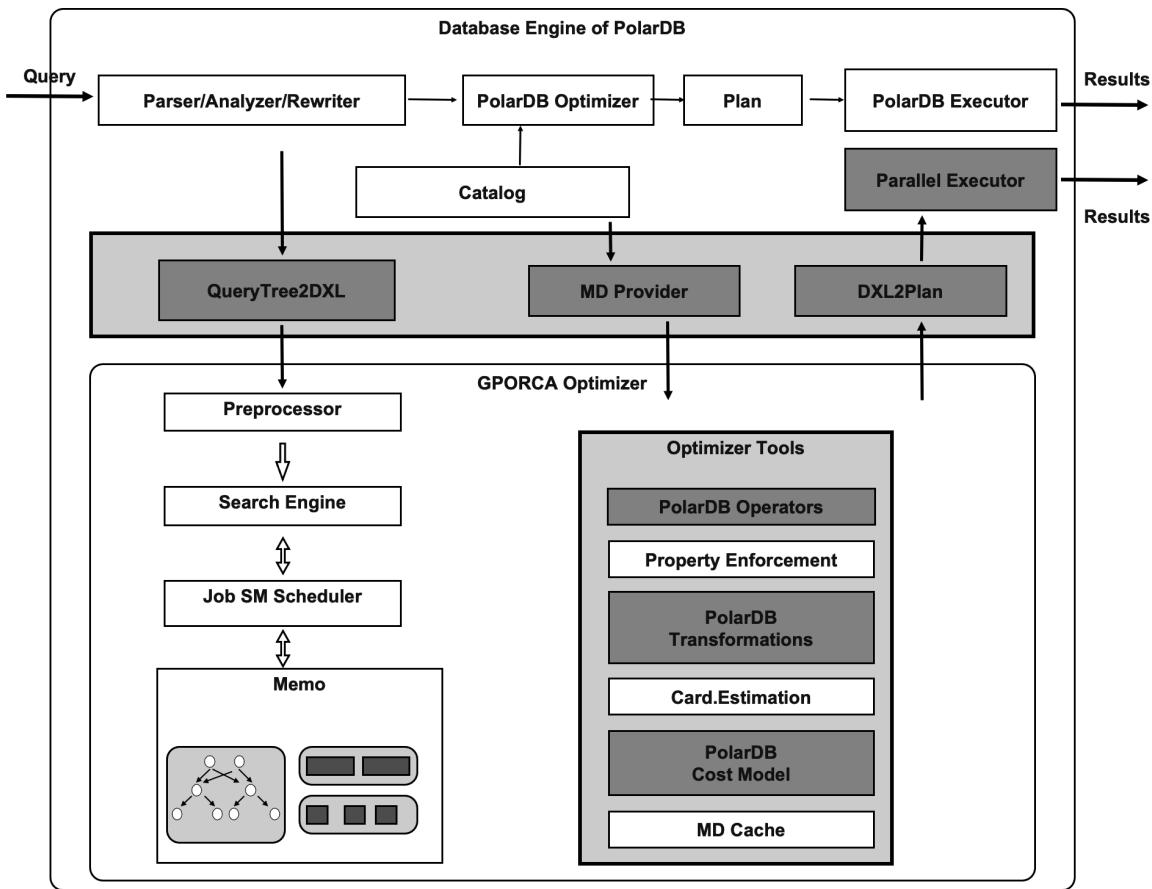
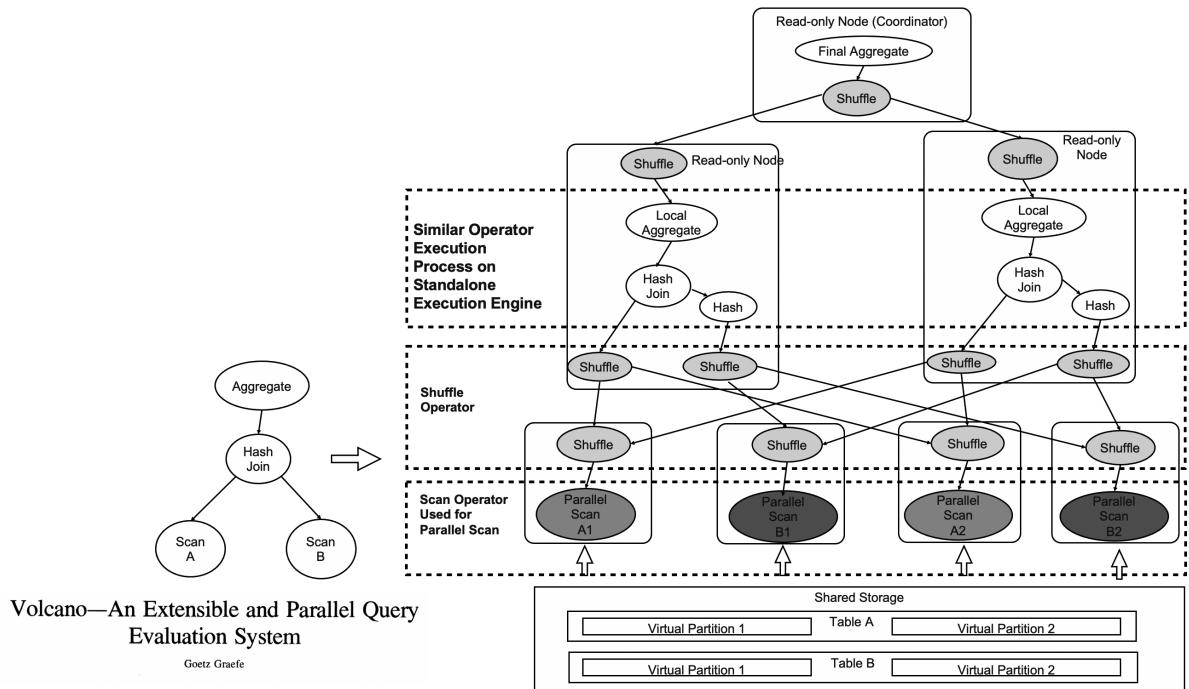
<https://apsaradb.github.io/PolarDB-for-PostgreSQL/theory/arch-overview.html#polardb-htap>

**PolarDB uses a shared-storage-based architecture in which computing is decoupled from storage.** The conventional shared-nothing architecture is changed to the shared-storage architecture. N copies of data in the compute cluster and N copies of data in the storage cluster are changed to N copies of data in the compute cluster and one copy of data in the storage cluster. The shared storage stores one copy of data, but the data states in memory are different. The WAL logs must be synchronized from the primary node to read-only nodes to ensure data consistency. In addition, when the primary node flushes dirty pages, it must be controlled to prevent the read-only nodes from reading future pages. Meanwhile, the read-only nodes must be prevented from reading the outdated pages that are not correctly replayed in memory. To resolve this issue, PolarDB provides the index structure *LogIndex* to maintain the page replay history. *LogIndex* can be used to synchronize data from the primary node to read-only nodes.

After computing is decoupled from storage, the I/O latency and throughput increase. When a single read-only node is used to process analytical queries, the CPUs, memory, and I/O of other read-only nodes and the large storage I/O bandwidth cannot be fully utilized. To resolve this issue, PolarDB provides the shared-storage-based MPP engine. The engine can use CPUs to accelerate analytical queries at SQL level and support a mix of OLAP workloads and OLTP workloads for HTAP.

For more information, see [Architecture](#).





<https://github.com/ApsaraDB/PolarDB-FileSystem>

A screenshot of the GitHub repository page for ApsaraDB / PolarDB-FileSystem. The repository has 3 branches and 2 tags. The master branch is selected. The code tab is active, showing a list of files and their commit history. Key commits include:

- koperjian Merge pull request #14 from dynastyssea/master ... (commit d0c5dc6, Aug 9, 2022, 11 commits)
- conf PolarDB file system 1.2.41 first version (2 years ago)
- deploy\_scripts add support for FUSE on Polar File System (last year)
- docker PolarDB file system 1.2.41 first version (2 years ago)
- docs PolarDB file system 1.2.41 first version (2 years ago)
- etc PolarDB file system 1.2.41 first version (2 years ago)
- src refine output msg when mkfs failed (9 months ago)
- CMakeLists-config.txt PolarDB file system 1.2.41 first version (2 years ago)
- CMakeLists.txt PolarDB file system 1.2.41 first version (2 years ago)
- LICENSE.txt PolarDB file system 1.2.41 first version (2 years ago)
- NOTICE.txt PolarDB file system 1.2.41 first version (2 years ago)
- Readme-CN.md Update readme for tool reference. (2 years ago)
- Readme-FUSE-CN.md add support for FUSE on Polar File System (last year)
- Readme-FUSE.md add support for FUSE on Polar File System (last year)
- Readme.md Update readme for tool reference. (2 years ago)

The repository has 147 stars, 63 forks, and 10 watching. It is a report repository.

## 大数据

## K8S

<https://github.com/kubernetes/kubernetes>

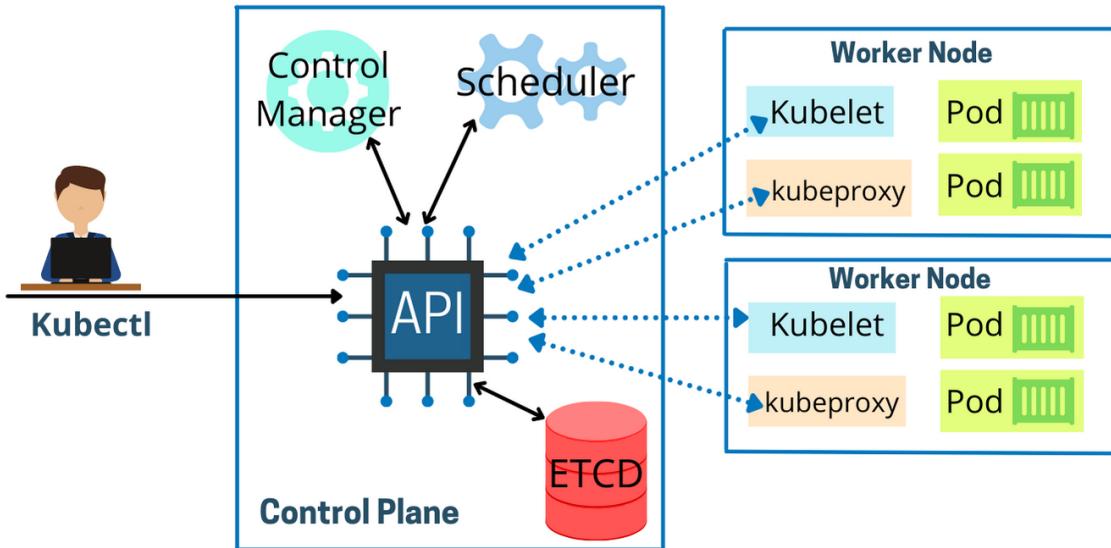
A screenshot of the GitHub repository page for kubernetes / kubernetes. The repository has 50 branches and 1,022 tags. The master branch is selected. The code tab is active, showing a list of files and their commit history. Key commits include:

- k8s-ci-robot Merge pull request #117393 from HirazawaUi/use-wait-replace-loop ... (commit e28866e, 1 hour ago, 116,278 commits)
- .github Add new contribex leads to sig-contribex-approvers (last month)
- CHANGELOG CHANGELOG: Update directory for v1.25.10 release (last week)
- LICENSES Update kube-openapi, drop mapstructure (last week)
- api Regenerate discovery fixtures (last week)
- build Merge pull request #118076 from liggitt/zeitgeist-isolated (last week)
- cluster Merge pull request #117921 from kkkkun/clean-up-etcd-version (5 days ago)
- cmd kubeadm: remove function pointer comparison in phase test (3 days ago)
- docs Make root approval non-recursive (7 months ago)
- hack Merge pull request #114053 from brianpurseley/fix-update-translations (11 hours ago)
- logo logo: better alignment of layers (7 months ago)
- pkg Merge pull request #116994 from mirimanda96/fivx/116505 (1 hour ago)
- plugin Add api-machinery TL owners permissions for jpbetz (last week)
- staging Merge pull request #114053 from brianpurseley/fix-update-translations (11 hours ago)
- test Merge pull request #117393 from HirazawaUi/use-wait-replace-loop (1 hour ago)

The repository has 98.6k stars, 36.2k forks, and 3.2k watching. It is a production-grade container scheduling and management system. It uses kubernetes.io, go, kubernetes, containers, and cncf.



# Kubernetes Cluster

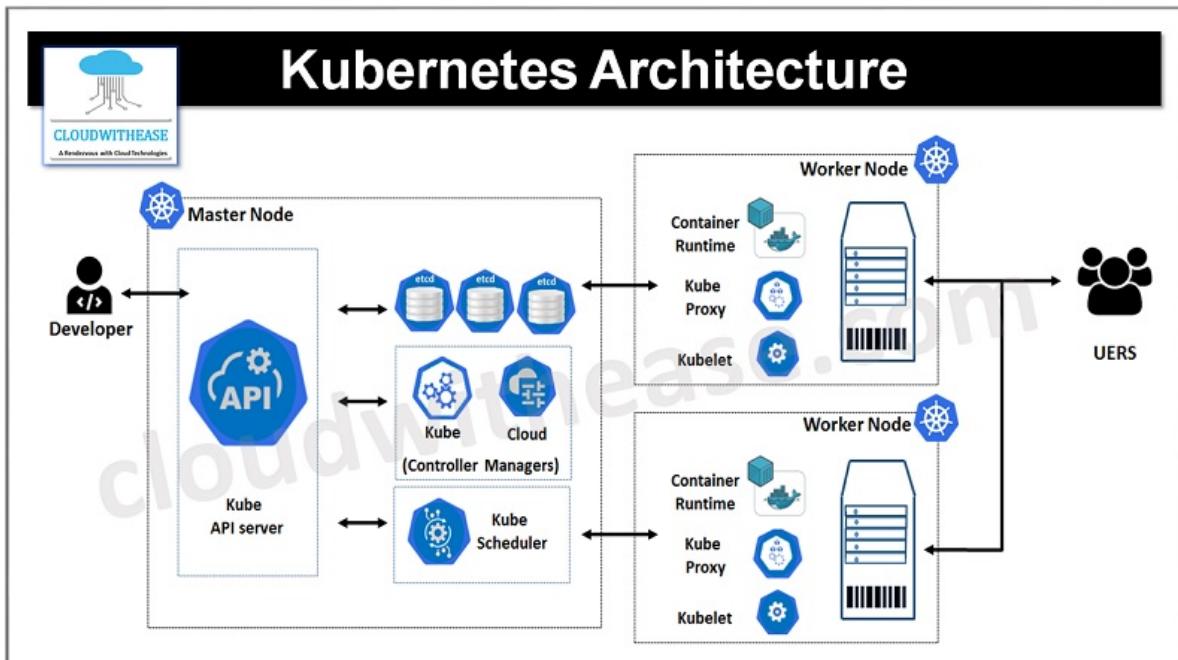


<https://justdeployit.hashnode.dev/deploy-an-application-on-kubernetes>

Kubernetes, also known as K8s, is an open source system for managing [containerized applications](#) across multiple hosts. It provides basic mechanisms for the deployment, maintenance, and scaling of applications.

Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called [Borg](#), combined with best-of-breed ideas and practices from the community.

Kubernetes is hosted by the Cloud Native Computing Foundation ([CNCF](#)). If your company wants to help shape the evolution of technologies that are container-packaged, dynamically scheduled, and microservices-oriented, consider joining the CNCF.



<https://cloudwithease.com/what-is-kubernetes/>

Kube-scheduler is responsible for scheduling Pods to worker nodes. Implementation of sophisticated scheduling algorithms takes a lot of information into account such as availability of resources on each node, constraints mentioned by user, type of available nodes, resource limits and quotas, and other factors such as affinity , anti-affinity, tolerations and taints.

[你的理解]

## 大数据软件的制作 - Zookeeper 为例(?)

线索



<https://github.com/apache/zookeeper>

## [你的理解]

# 总结

## 内容概要

本课程的内容，其实也就体现在了章节目录里：

- LSC程序的样子 - 以理解天气预报的热传导方程(PDE的数值计算)为例
  - 给定了3个Python 的串行代码 - Images, Show, Animation
- LSC程序设计的套路 - PCAM/DAOM等
- 运行环境 - 硬件 - 三个概要的方式
- 运行环境 - 软件 - 那个协议栈
- 算法级编程框架
  - MPI (MPI4PY), Multi-threaded (Numba - OpenMP), GPGPU (PyCUDA) 和 Big Data (PySpark)
  - Dask, Ray 等
    - 惜乎
- 算法级编程 - 进阶
  - Cython, PyBind11, Pythran等
    - 惜乎
  - Deep Learning 的概念，和框架本身的设计与实现 - 如 PyTorch, TensorFlow (现在Google有推出了JAX)，国内的 PaddlePaddle等
    - 除了 JAX 是基于 Google 自己的 XLA，其他的基本都是基于 PyBind11 构建的
    - 惜乎

- 系统级编程 - 那些分布式系统的设计与实现 (包括大数据、云计算)
    - 惜乎
  - 应用案例 - 以商务计算为例
    - 互联网厂商的盈利模式 - 在黏着巨量用户的前提下，抽成、广告、理财、云计算等
    - 支撑技术 - “秒杀”的解决
    - 支撑技术 - 计算广告的概念
- CTR 预估算法 - 从GBDT+LR入手，进一步了解 DL-based CTR算法 (惜乎)

## 意犹未尽呀~

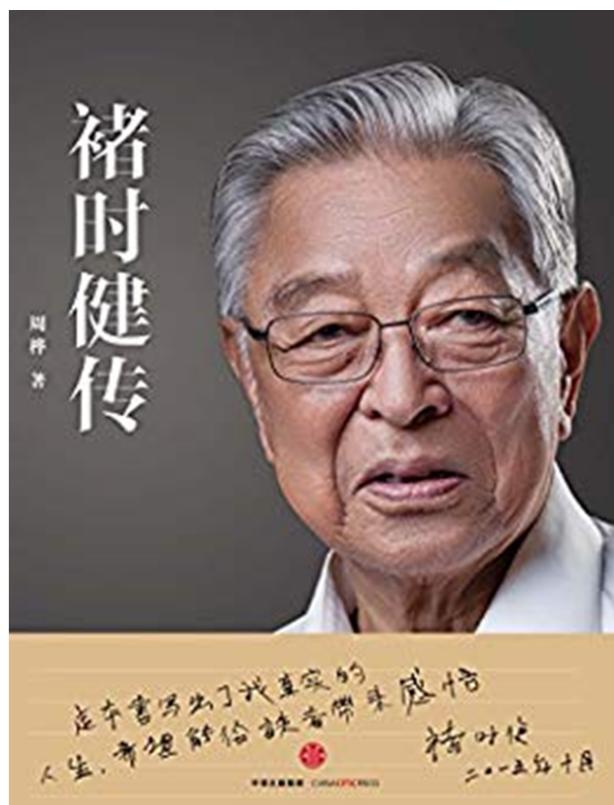
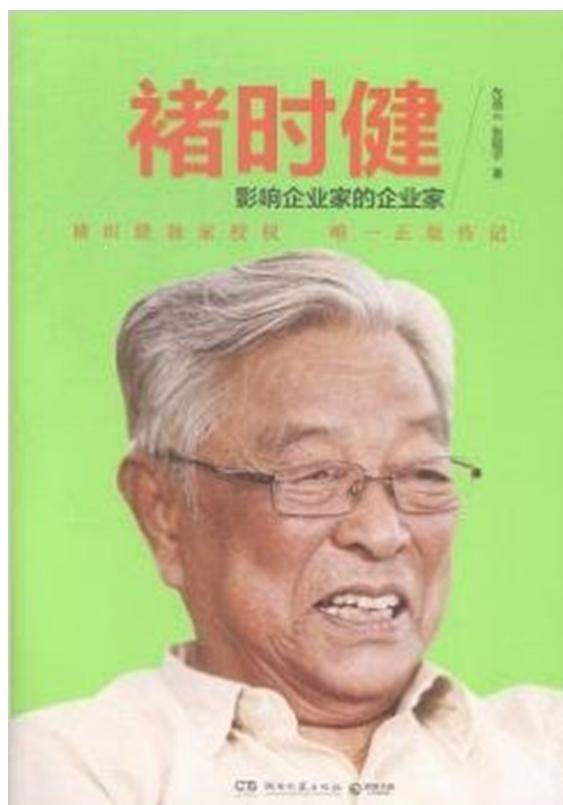
前面留了很多“惜乎”，其实也多少准备了的，终究因为时间限制，没有展开，很是遗憾 - 不能给各位同学更广阔的了解

- 学应广博，做要深刻！

好在有些内容，咱们有学生多少有所涉及，希望其他同学能在前面知识结构的提示下，能有兴趣和勇气进一步去了解！

## 几本书

褚时健传 - 了解下这位老人，体会下“商务思维” - 推荐





## Python Parallel Programming Cookbook - 推荐

**lancelote / parallel\_python** Public archive

Code Issues Pull requests Actions Projects Security Insights

Code Go to file Code

master 1 branch 4 tags

About

Code for Python Parallel Programming Cookbook by Giancarlo Zaccone

python book parallel

Readme 30 stars 4 watching 13 forks

Releases 4 tags

Packages No packages published

Languages Python 100.0%

More logging instead of prints, sleeps and formatting eSah437 on Feb 14, 2017 43 commits

chapter1 More logging instead of prints, sleeps and formatting 6 years ago

chapter2 More logging instead of prints, sleeps and formatting 6 years ago

chapter3 More logging instead of prints, sleeps and formatting 6 years ago

chapter4 Asyncio and Futures More logging instead of prints, sleeps and formatting 7 years ago

chapter5 Celery 7 years ago

source Thread synchronisation with queues 7 years ago

.gitignore Chapter 1 complete 7 years ago

python-version Celery 7 years ago

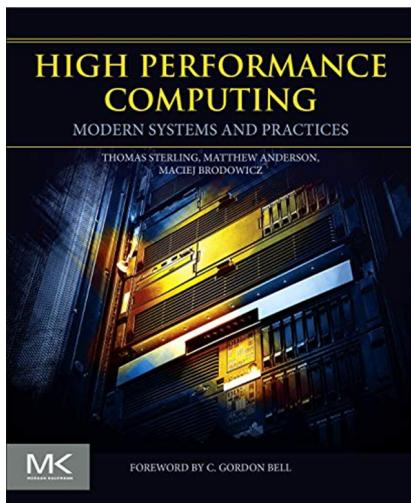
NOTES.md Asyncio and Futures 7 years ago

README.md Celery 7 years ago

requirements.txt Celery 7 years ago

README.md

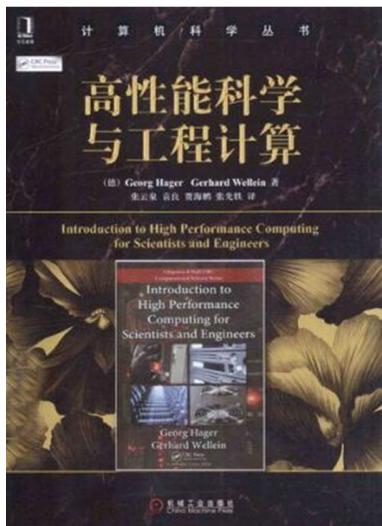
**High Performance Computing: Modern Systems and Practices**  
By 作者: Thomas Sterling, Matthew Anderson, Maciej Brodowicz



- **High Performance Computing: Modern Systems and Practices**
- **By 作者: Thomas Sterling – Matthew Anderson – Maciej Brodowicz**
- **ISBN-10 书号: 012420158X**
- **ISBN-13 书号: 9780124201583**
- **Edition 版本: 1**
- **Release Finelybook 出版日期: 2018-07-01**
- **pages 页数: (718 )**



## 高性能科学与工程计算 [德] Georg Hager, [德] Gerhard Wellein



□ 高性能科学与工程计算

□ [德] Georg Hager, [德] Gerhard Wellein

- 《计算机科学丛书：高性能科学与工程计算》从工程实践的角度介绍了高性能计算的相关知识。主要内容包括现代处理器的体系结构、为读者理解当前体系结构和代码中的性能潜力和局限提供了坚实的理论基础。
- 接下来讨论了高性能计算中的关键问题，包括串行优化、并行、OpenMP、MPI、混合程序设计技术等。
- 作者根据自身的研究也提出了一些前沿问题的解决方案，如编写有效的C++代码、GPU编程等。

## 高性能计算应用概览 作者: 历军



□ “高性能计算应用概览”

□ 作者: 历军

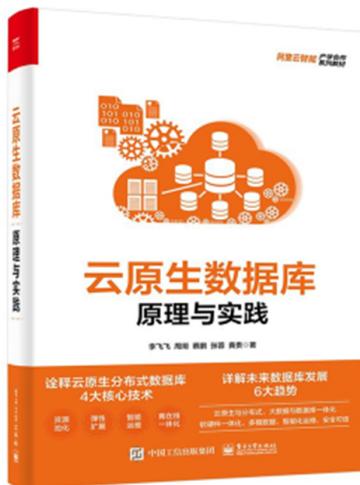
□ 出版社: 清华大学出版社

□ 出版年: 2018-6-1

□ 定价: 69.8

□ 装帧: 平装

□ ISBN: 9787302504726



□ 云原生数据库：原理与实践（全彩）（博文视点出品）  
□ 李飞飞, 周烜, 蔡鹏, 张蓉, 黄贵 ... 著  
□ 2022  
□ 电子工业出版社



347

## 几个视频

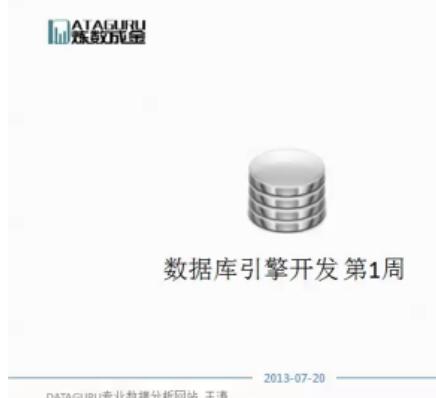
**强烈推荐！**

[https://www.bilibili.com/video/BV1ct411k7jn/?spm\\_id\\_from=333.788.recommend\\_more\\_video\\_0](https://www.bilibili.com/video/BV1ct411k7jn/?spm_id_from=333.788.recommend_more_video_0)



<https://www.zhihu.com/people/WangTaoSequoiaDB>





**动手实现数据库引擎(cpp03)**

6002播放 优优影视 2018-12-29 09:31:05



2013-07-20

DATAGURU专业数据分析网站 王涛



303

**MIT 6.824 Distributed Systems**  
Kris Pritchard - 2 / 20

- Lecture 1: Introduction
- Lecture 2: RPC and Threads
- Lecture 3: GFS
- Lecture 4: Primary-Backup Replication
- Lecture 5: Go, Threads, and Raft
- Lecture 6: Fault Tolerance: Raft (1)
- Lecture 7: Fault Tolerance: Raft



## 加州大学伯克利分校 CS 194 Introduction to Parallel Programming (Fall 2020)

**Introduction to Parallel Programming**  
CS194-15 Fall 2020  
<https://sites.google.com/lbl.gov/cs194-15-fa2020/>

加州大学伯克利分校 CS 194 并行程序设计导论 Introduction to Parallel Programming (Fall 2020)  
4024 0 2021-04-23 21:41:33

main 守夜人集结 漠冬已至，洛杉矶  
+关注 1.3万

视频列表

短刀威武客	守夜人集结 漠冬已至，洛杉矶
—道不平事—	【黑】黑暗当歌~

视频选集 (125)  自动进播

P1 Lecture 1 Overview	47:42
P2 Lecture 2 Mem Hierarchy (Tiled MatMul)	55:45
P3 Lecture 3 Cache Oblivious MatMul	51:03
P4 Lecture 4 Compilers	38:46
P5 Lecture 5 Review serial performance	52:44
P6 Lecture 6 Shared Mem (OpenMP)	52:44
P7 Lecture 7 Runtime model	49:17
P8 Lecture 8 Synchronization	47:28
P9 Lecture 9 Load balancing	1:02:02
P10 Lecture 10 Linearizability	48:39

1人正在看，已获赞 0 条评论 鸟 鸟 挑某个评论的举报记录由下  
112 29 575 37

150年现代设计史居然写了一堆椅子？  
王健之教授

极喜欢做研究

[https://www.bilibili.com/video/BV1QQ4y1o7m/?spm\\_id\\_from=333.337.search-card.all.click&vd\\_source=e54e9f9a24decfb0d0ebbf026a36ea3](https://www.bilibili.com/video/BV1QQ4y1o7m/?spm_id_from=333.337.search-card.all.click&vd_source=e54e9f9a24decfb0d0ebbf026a36ea3)

## □ MPI, OpenMP, Pthread, CUDA, Hadoop/Spark

**Programming Assignments**

- Programming Languages:**
  - MPI (Message Passing Interface)
  - OpenMP
  - Pthread (POSIX Thread)
  - CUDA
  - Hadoop&Spark (Java&SCALA)
- Report Items:**
  - ReadMe (Code Design Explanation)
  - Performance analysis
  - Anything else you think meaningful
- Grading Items:**
  - Code correctness
  - Report (Performance analysis)
  - Code Performance



308

预祝大家鹏程万里！



妙

狂

劍

果

漢卿  
志雅

劍

江





 Artron .Net

11