请不要外传 :)

感谢各位同学 - ”教学相长“嘛

目录

- 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

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

> Twins.

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

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

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

where *u* is the quantity that we want to know, *t* is for temporal variable, *x* and *y* are for spatial variables, and *α* 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)

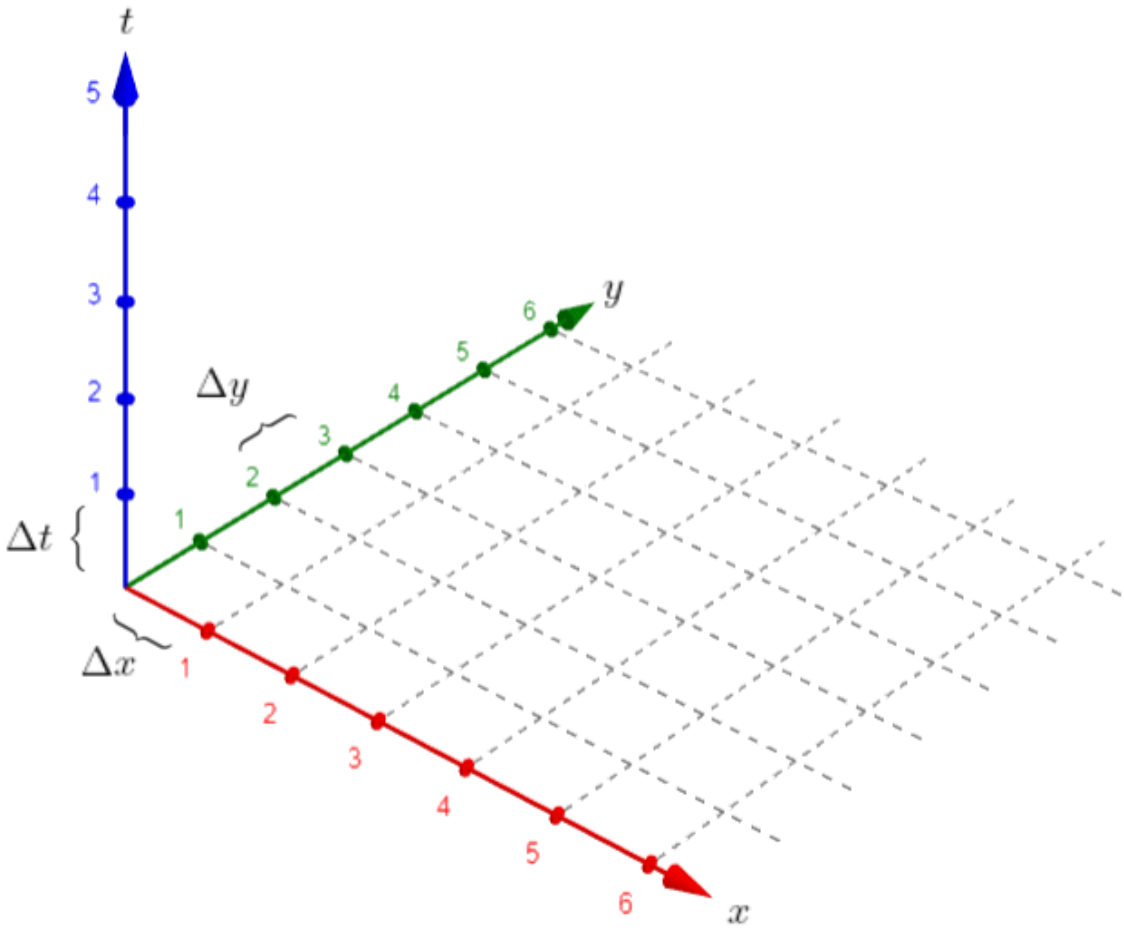
Now let's see the finite-difference method (FDM) in a nutshell. Finite-difference method is a numerical method for solving differential equations by approximating derivative with finite differences. Remember that the definition of derivative is

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

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 Δ*x* = Δ*y*, we get this final equation

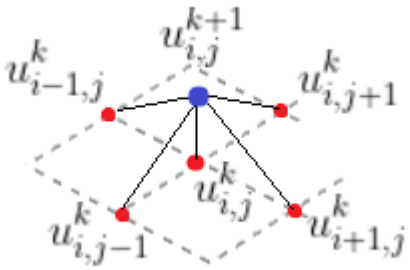
$$u_{i,j}^{k+1} = \gamma(u_{i+1,j}^k + u_{i-1,j}^k + u_{i,j+1}^k + u_{i,j-1}^k - 4u_{i,j}^k) + u_{i,j}^k$$

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)

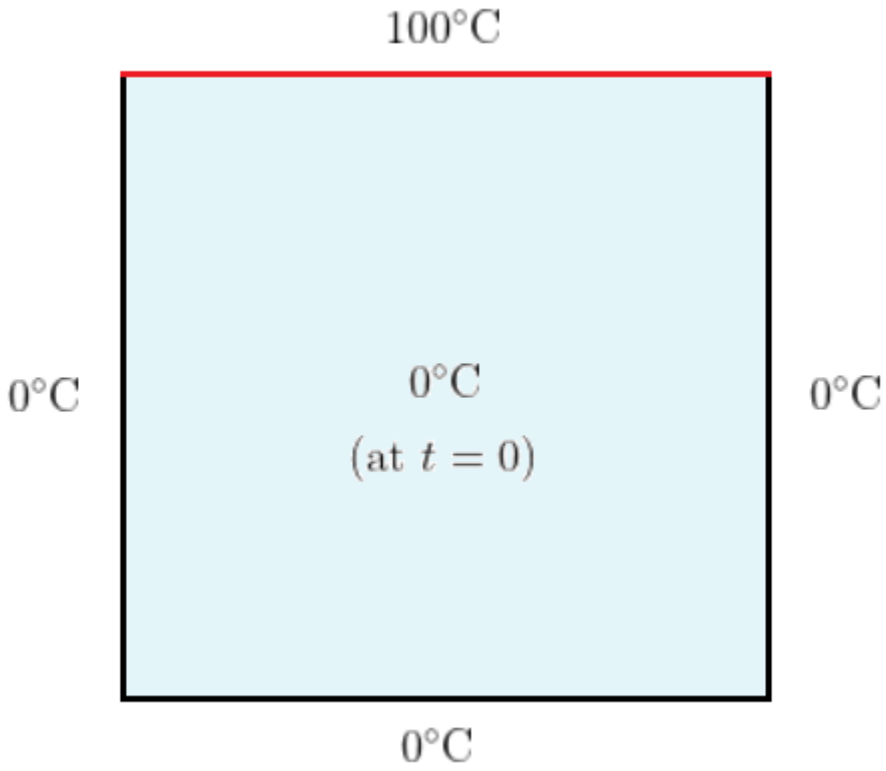


Explicit method stencil (image by author)

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

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

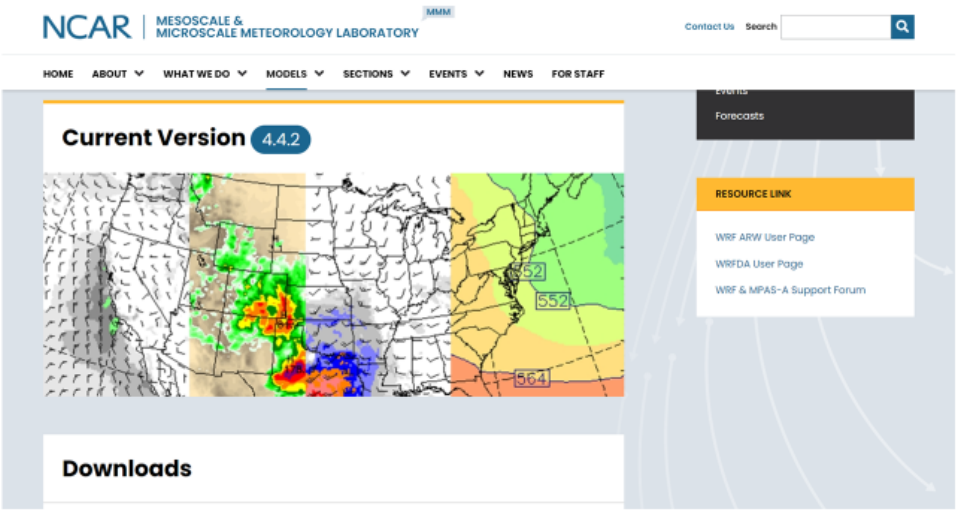
Everything is ready. Now we can solve the original heat equation approximated by algebraic equation above, which is computer-friendly. For an exercise problem, let's suppose a thin square plate with the side of 50 unit length. The temperature everywhere inside the plate is originally 0 degree (at $t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

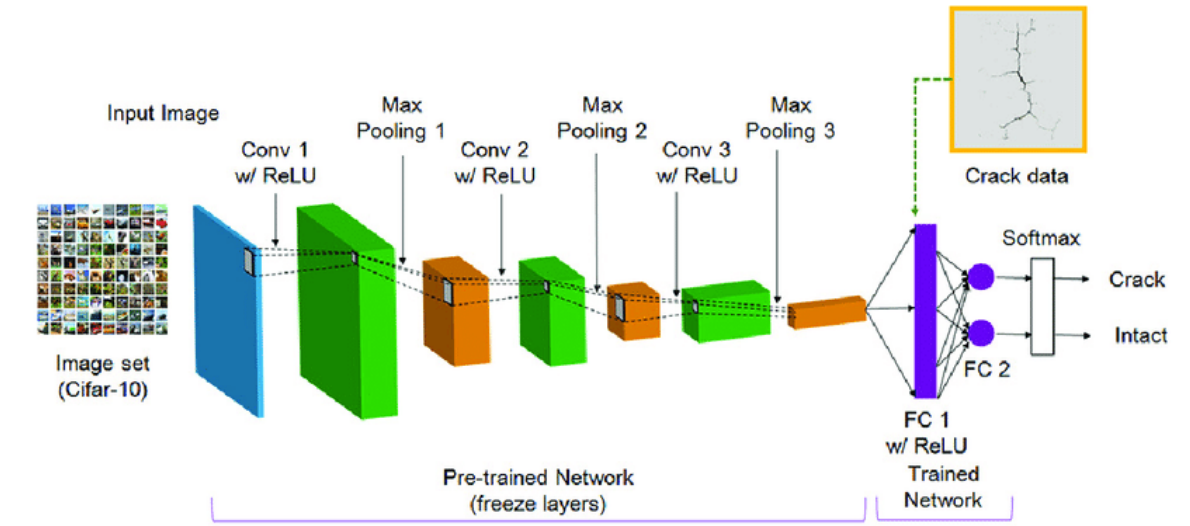


Global Grid (0.25*0.25 → 1440X720 dots = 62208000)

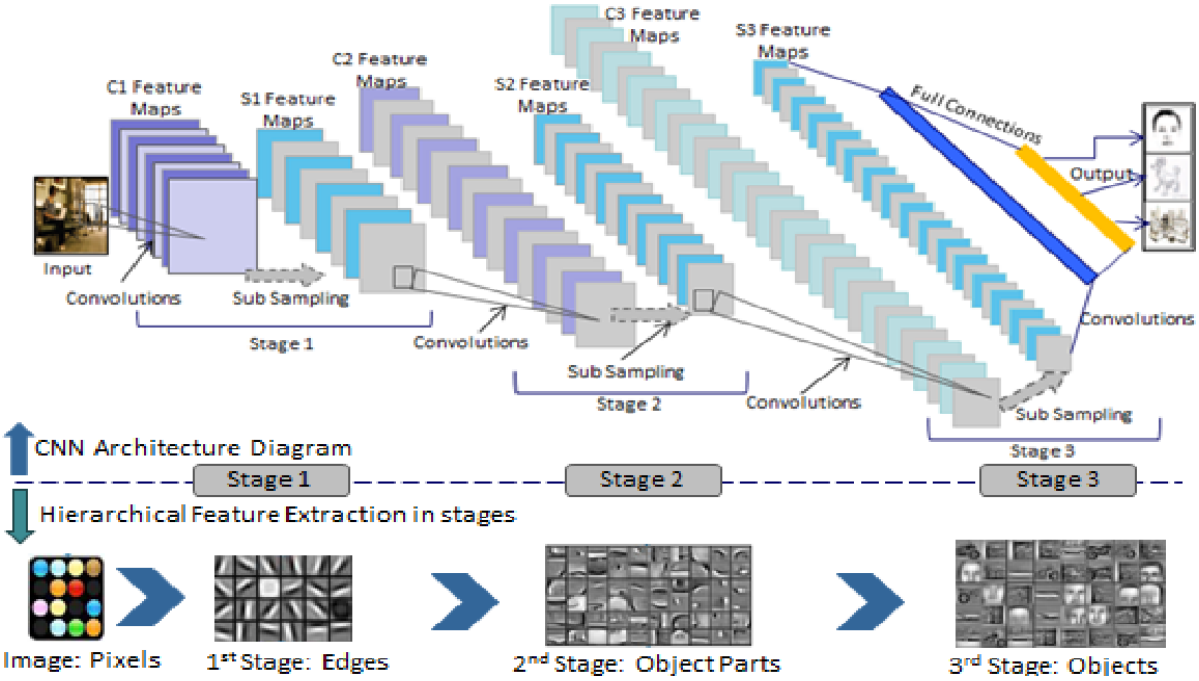
- If we take resolution as 0.25*0.25, and 60 vertical layers, the matrix size will be 1440*720*60 = $6.22*10^7$.
- If 6 variables, the number is 6*$6.22*10^7$.
- If we want to predict the weather in 24 hours, and 5 minutes as span, we should compute values of
  - 288*6*$6.22*10^7$.≈$1.08*10^{11}$.
 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

https://www.esrl.noaa.gov/psd/

深度学习 - CNN、ChatGPT

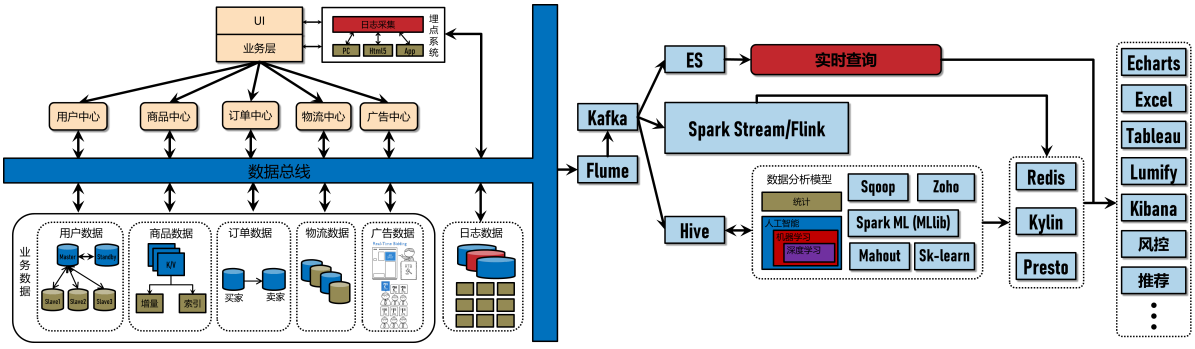


https://www.researchgate.net/figure/Schematic-of-the-deep-learning-architecture_fig4_325663483



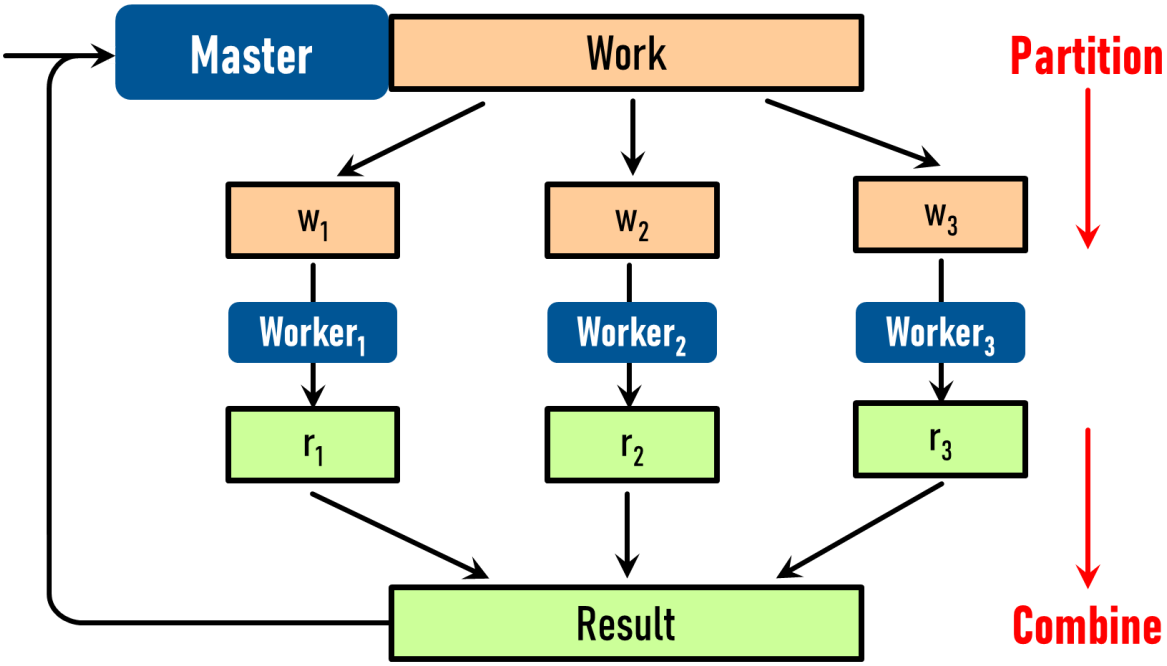
https://d3i71xaburhd42.cloudfront.net/f78e280123b1c0c68f84da3cc6c66615f6e7cebd/3-Figure1-1.png

商务系统 - 阿里淘宝，Google, 滴滴， 高德地图等

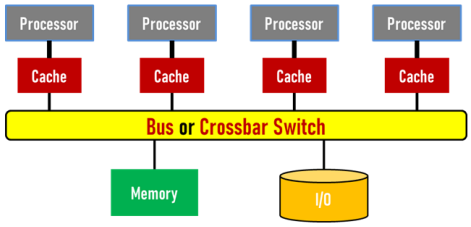


编写 LSC程序的套路

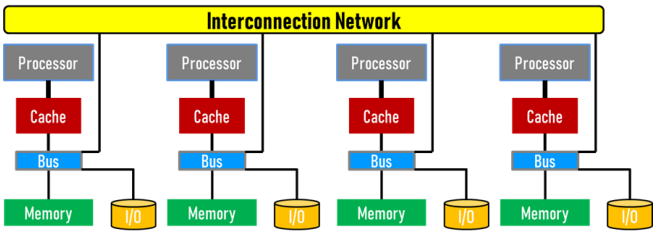
问题的分解：Divide & Conquer



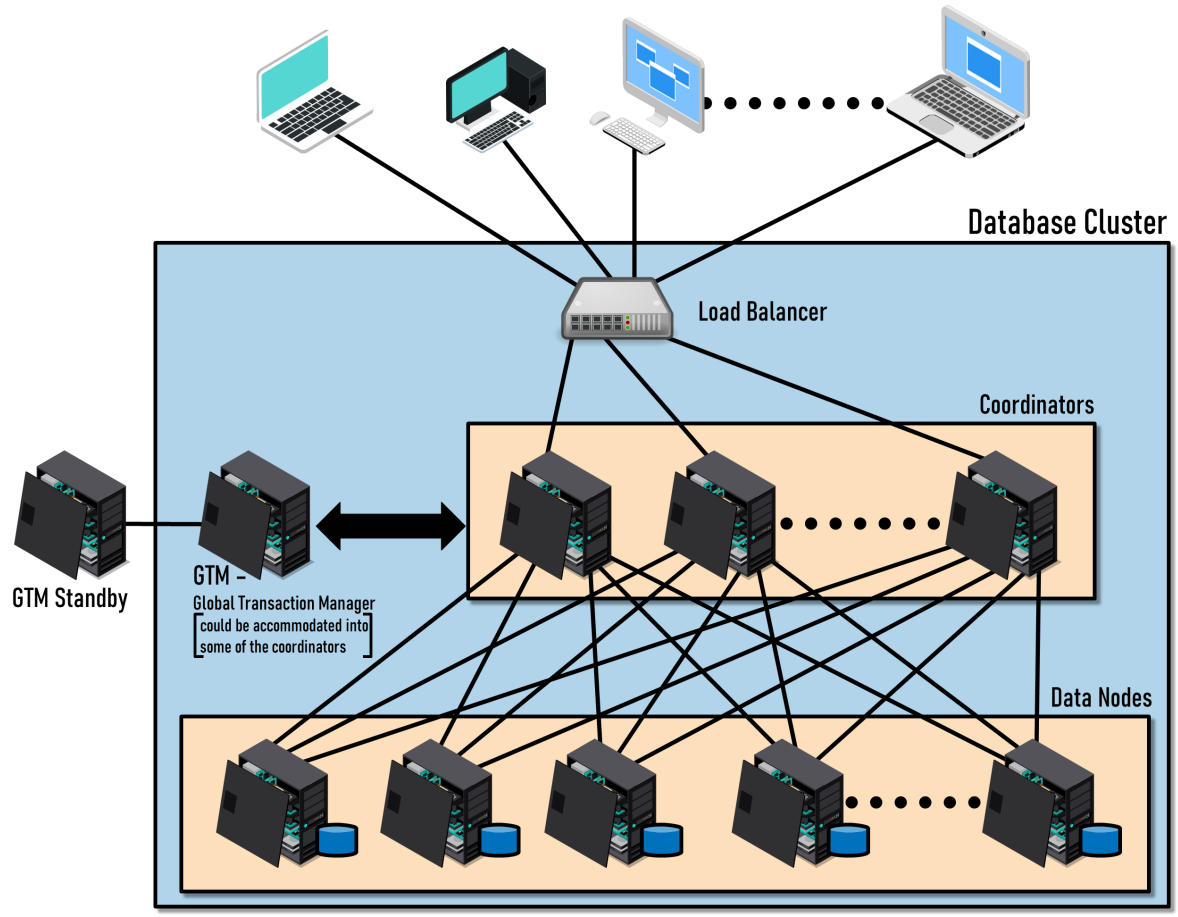
运行的环境：多 - 核，节点



1961 SMP → 1974 Vector Machine →
2005 Multi-Core CPU (AMD, Intel) → 2006 GPGPU (CUDA)



1979 MMP → 1994 Cluster

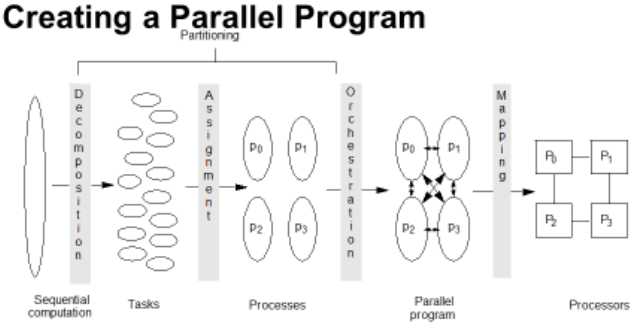


DAOM/PCAM

DAOM (1991)

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

PCAM设计方法学

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**)

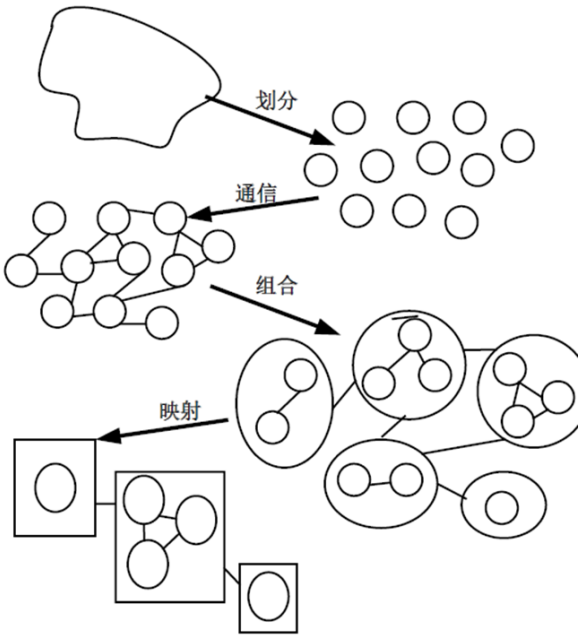
- 确定诸任务执行中所需要交换的数据和协调诸任务的执行，由此检测上述划分的合理性

任务组合(**Agglomeration**)

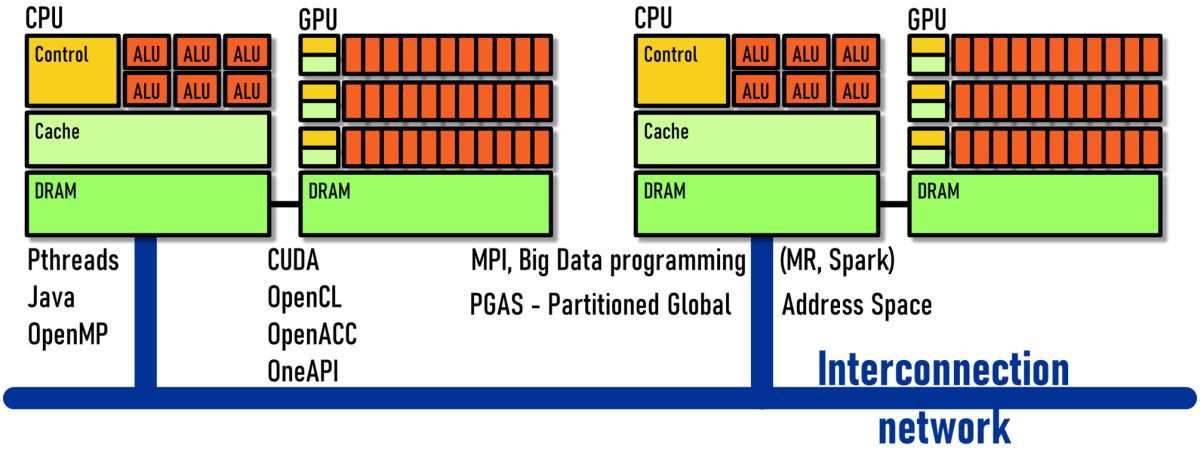
- 按性能要求和实现的代价来考察前两阶段的结果，必要时可将一些小任务组合成更大的任务以提高性能和减少通信开销

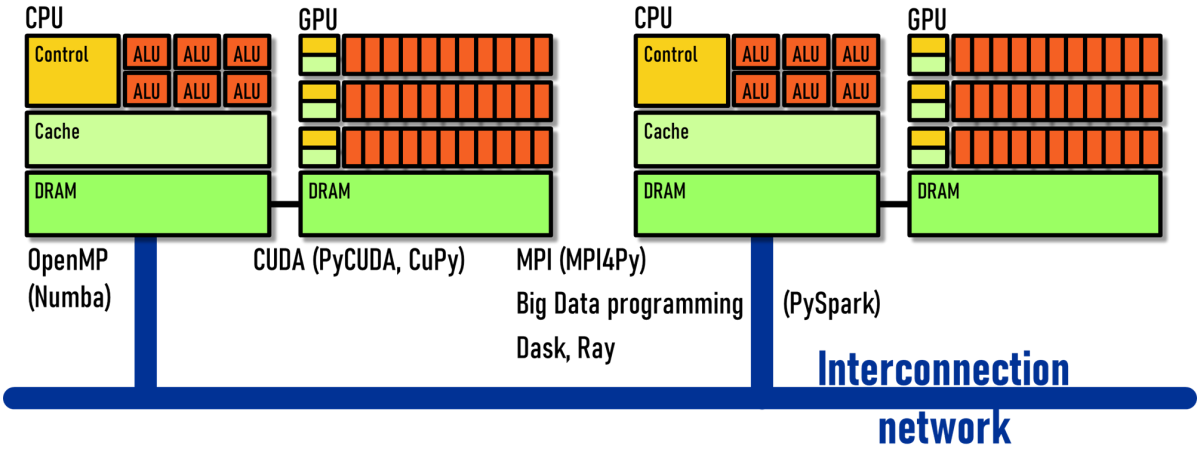
处理器映射(**Mapping**)

- 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

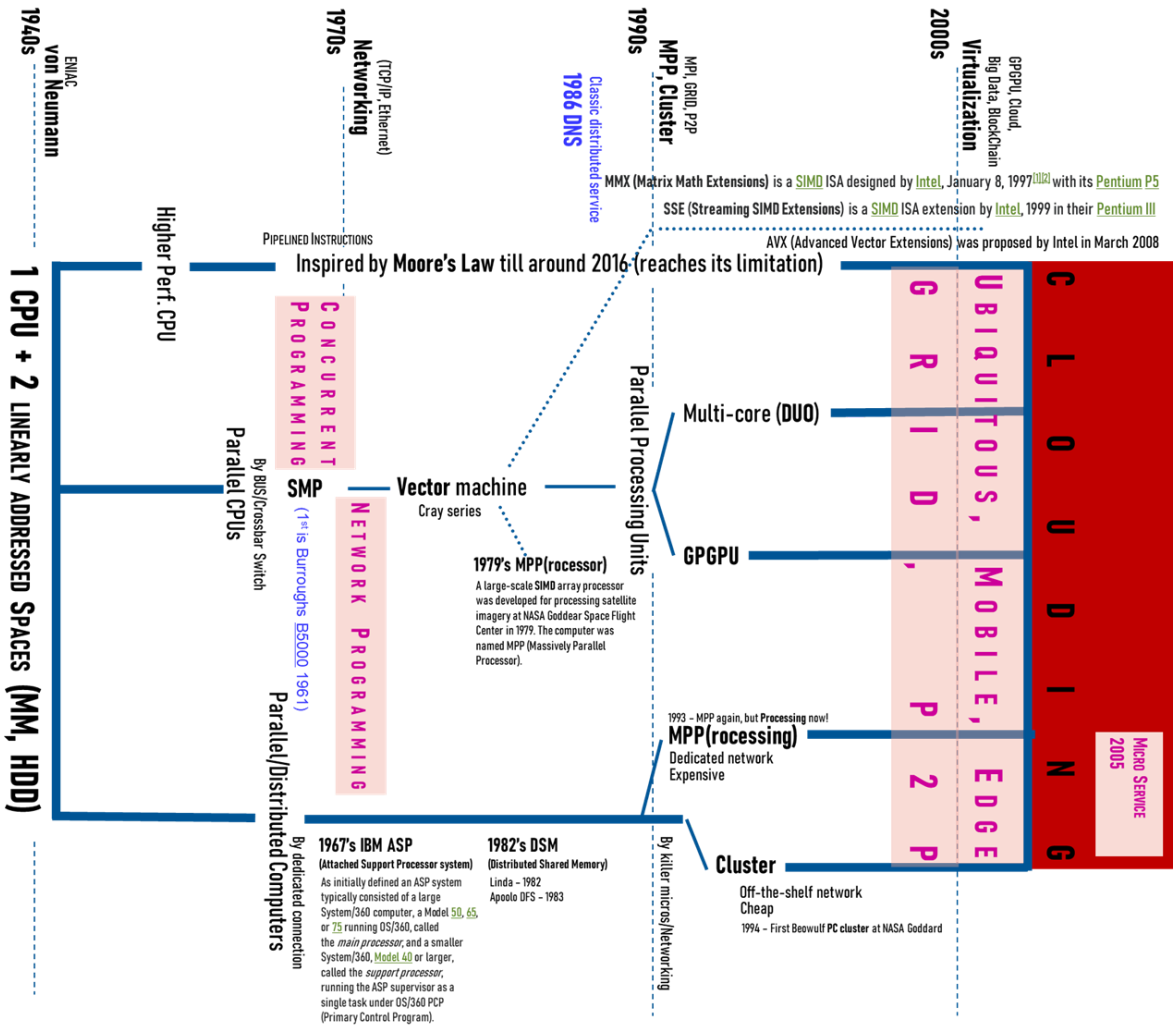


编程框架

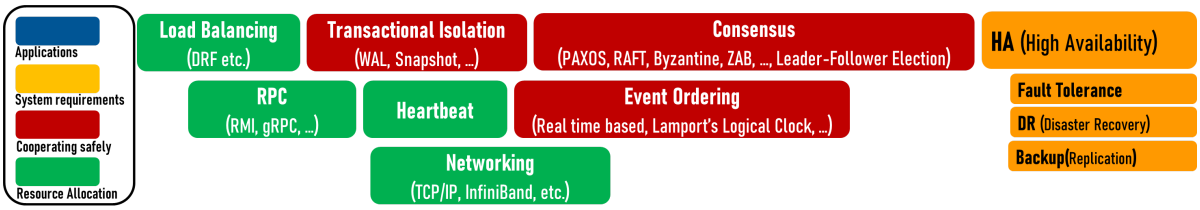




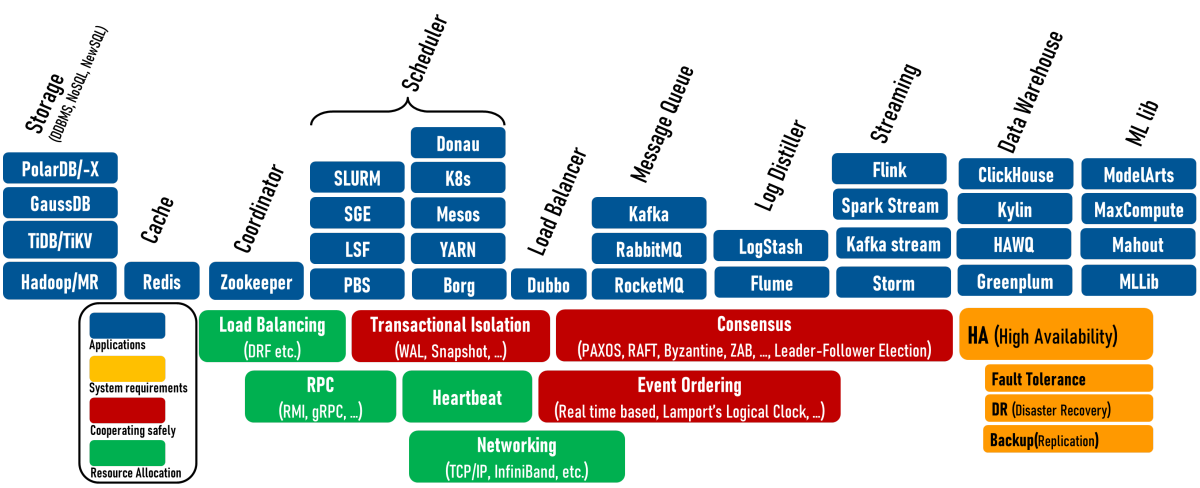
硬件环境

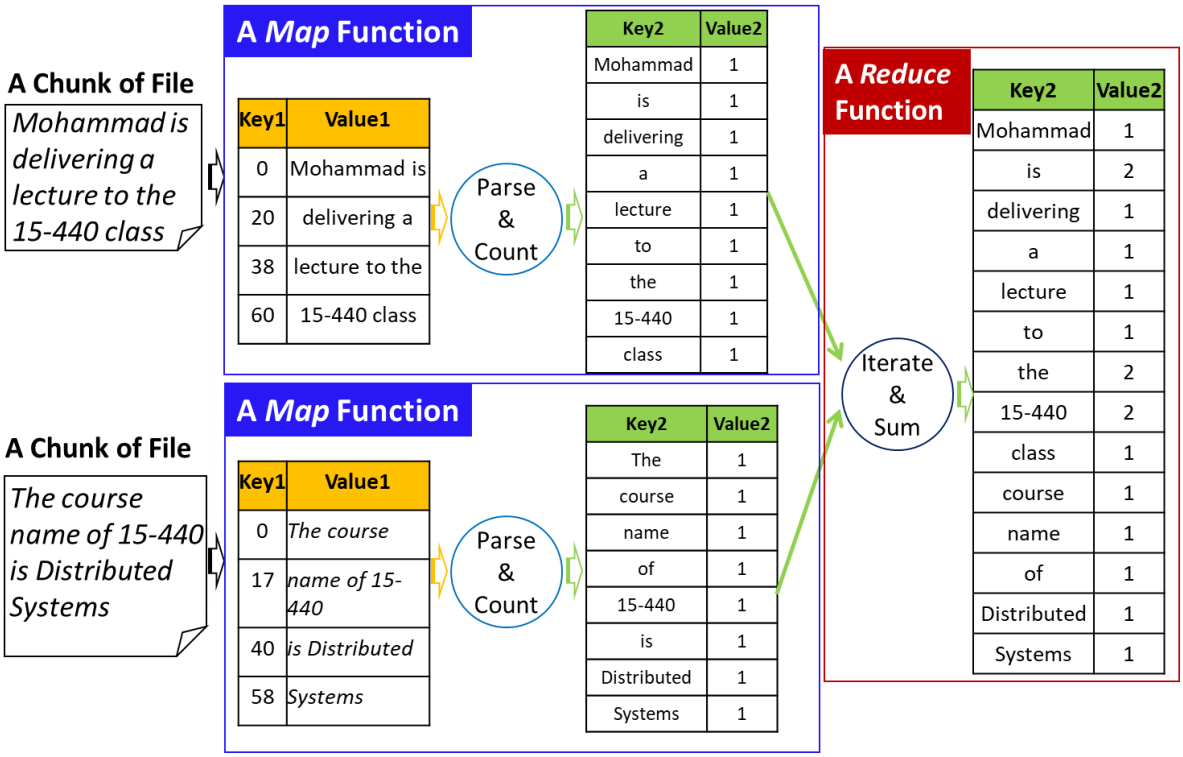


软件环境 - 协议栈



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

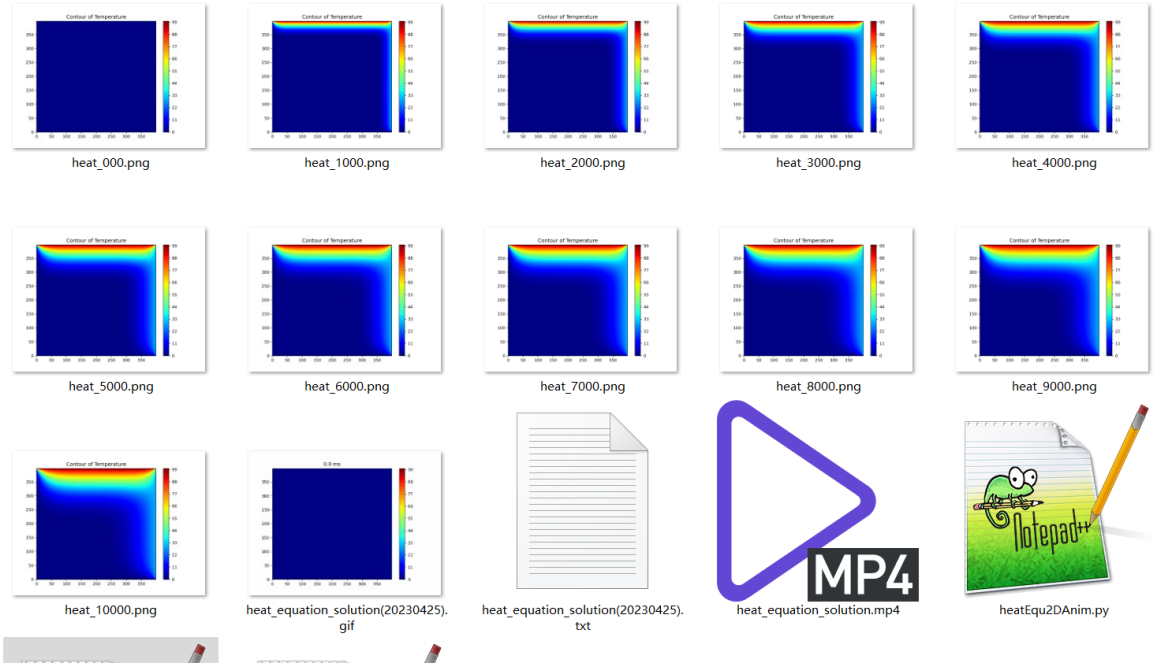
[Images] heatEqu2DSeqImgs.py

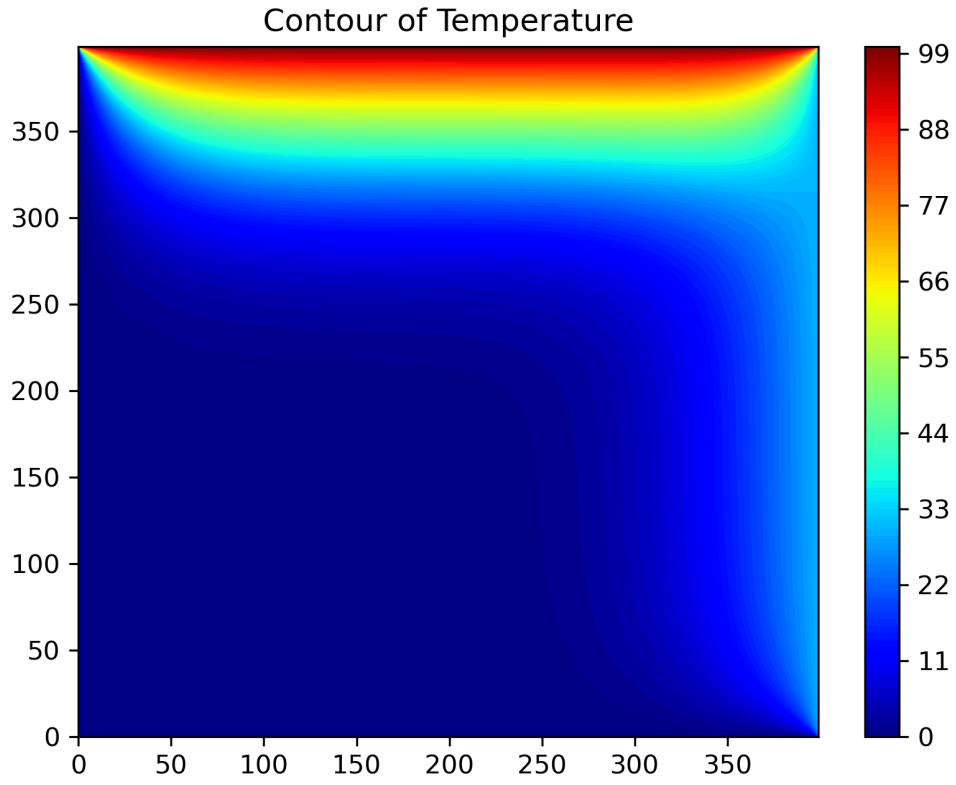
"D:\myCodes\HPCbook\00heatEqu\heatEqu2DSeqImgs.py"

```
'''
https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2p.py

"D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-heatequ2Dk.py"
'''

from __future__ import print_function
import numpy as np
import timeit


import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt

# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
```

```
plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
plt.cm.coolwarm
plt.figure(dpi=300)

# Set colour interpolation and colour map
colorinterpolation = 100
colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm

# Set Dimension
lenX = lenY = 400 #we set it rectangular

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))


# Basic parameters
a = 0.1  # Diffusion constant
timesteps = 10000  # Number of time-steps to evolve system
image_interval = 1000  # Write frequency for png files

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))


# Boundary condition
Ttop = 100
Tbottom = 0
Tleft = 0
Tright = 30

# Initial guess of interior grid
Tguess = 0

def init_fields():
    # Set array size and set the interior value with Tguess
    field = np.empty((lenX, lenY))
    field.fill(Tguess)

    # Set Boundary condition
    field[(lenY-1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX-1):] = Tright
    field[:, :1] = Tleft

    print("size is ",field.size)
    print(field,"\n")

    field0 = field.copy()  # Array for field of previous time step
    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      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
```





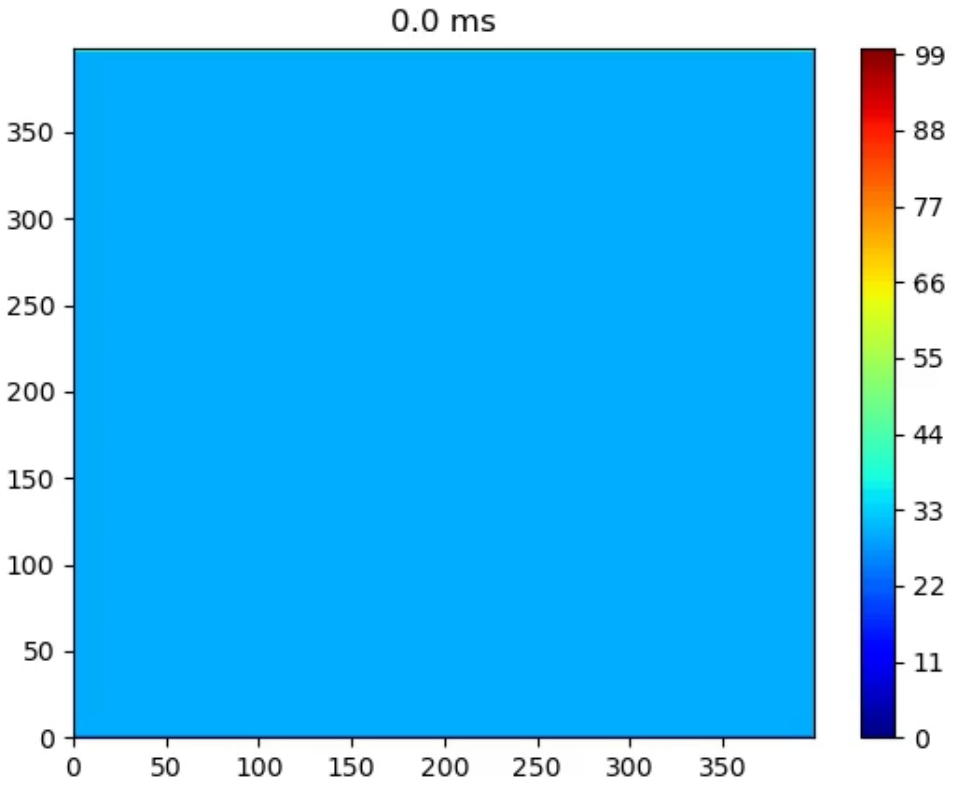
[Animation]

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

```
"D:\myCodes\HPCprojects\SourceCodes\Learning Scientific Programming with Python(Christian Hill)\Animation of the diffusion equation.md"
"""

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from PIL import Image

SAVE_ANIMATION = True # False True

# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap = plt.cm.coolwarm
# plt.figure(dpi=300)

# Set colour interpolation and colour map
colorinterpolation = 100
colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm


# Basic parameters
a = 0.1  # Diffusion constant  #例如 Thermal diffusivity of steel, mm2.s-1
timesteps = 100000  # Number of time-steps to evolve system
image_interval = 1000  # Write frequency for png files

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))

# Set Dimension
lenX = lenY = 4000 #we set it rectangular

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))


# Boundary condition
Ttop = 100
Tbottom = 0
Tleft = 0
Tright = 30

# Initial guess of interior grid
Tguess = 30

def init_fields():
    # Set array size and set the interior value with Tguess
    field = np.empty((lenX, lenY))
```

```
    field.fill(Tguess)

    # Set Boundary condition
    field[(lenY-1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX-1):] = Tright
    field[:, :1] = Tleft

    print("size is ",field.size)
    print(field,"\n")

    field0 = field.copy()  # Array for field of previous time step
    return field, field0

u, u0 = init_fields()

def evolve(u, u0, a, dt, dx2, dy2):
    # Propagate with forward-difference in time, central-difference in
space
    u[1:-1, 1:-1] = u0[1:-1, 1:-1] + a * dt * (
            (u0[2:, 1:-1] - 2 * u0[1:-1, 1:-1] +
             u0[:-2, 1:-1]) / dx2 +
            (u0[1:-1, 2:] - 2 * u0[1:-1, 1:-1] +
             u0[1:-1, :-2]) / dy2)
    u0[:] = u[:]

    # u0 = u.copy()
    return u, u0

fig, ax = plt.subplots()
# Configure the contour
plt.title("Contour of Temperature")
plt.contourf(X, Y, u0, colorinterpolation, cmap=colourMap)
# Set Colorbar
plt.colorbar()
plt.axis('on')

def animate(i):
    """Set the data for the ith iteration of the animation."""

    global u0, u, a, dt, dx2, dy2

    # plt.gca().clear()
    plt.cla()
    plt.clf()
    plt.title('{:.1f} ms'.format(i*dt*1000))
    plt.contourf(X, Y, u0, colorinterpolation, cmap=colourMap)
    # Set Colorbar
    plt.colorbar()
    plt.axis('on')

    u, u0 = evolve(u, u0, a, dt, dx2, dy2)

    return u0, u

if SAVE_ANIMATION:
```

```
    anim = animation.FuncAnimation(fig, animate, frames=timesteps,
repeat=False, interval = 50)
    # anim.save('us.gif', writer='imagemagick', fps=5)
    # anim.save("heat_equation_solution.gif", writer='imagemagick', fps=10)
    anim.save("heat_equation_solution.mp4", writer='ffmpeg', fps=10)
else:
    anim = animation.FuncAnimation(fig, animate, frames=timesteps)
    plt.show()


```



[Show]

```
'''
https://github.com/csc-training/hpc-python/blob/master/mpi/heat-
equation/solution/heat-p2p.py

"D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-
heatequ2Dk.py"
'''

from __future__ import print_function
import numpy as np
import timeit

```

```
import matplotlib
import matplotlib.pyplot as plt

# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
plt.cm.coolwarm
# plt.figure(dpi=300)

# Set colour interpolation and colour map
colorinterpolation = 100
colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm

# Basic parameters
a = 0.1  # Diffusion constant  #例如 Thermal diffusivity of steel, mm2.s-1
timesteps = 10000  # Number of time-steps to evolve system

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))

# Set Dimension
lenX = lenY = 400 #we set it rectangular

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))

# Boundary condition
Ttop = 100
Tbottom = 0
Tleft = 0
Tright = 30

# Initial guess of interior grid
Tguess = 0

def init_fields():
    # Set array size and set the interior value with Tguess
    field = np.empty((lenX, lenY))
    field.fill(Tguess)

    # Set Boundary condition
    field[(lenY-1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX-1):] = Tright
    field[:, :1] = Tleft

    print("size is ",field.size)
    print(field,"\n")

```

```
    field0 = field.copy()  # Array for field of previous time step
    return field, field0

def evolve(u, u_previous, a, dt, dx2, dy2):
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""

    u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
            (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
             u_previous[:-2, 1:-1]) / dx2 +
            (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
             u_previous[1:-1, :-2]) / dy2)
    u_previous[:] = u[:]


def main():
    field, field0 = init_fields()

    starting_time = timeit.default_timer()
    for m in range(1, timesteps + 1):
        evolve(field, field0, a, dt, dx2, dy2)

    # print("Iteration finished")
    print("Iteration finished. {} Seconds for Time
difference:".format(timeit.default_timer() - starting_time))

    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    # Set Colorbar
    plt.colorbar()
    plt.axis('on')
    # Show the result in the plot window
    plt.show()

if __name__ == '__main__':
    main()

```

算法级编程

MPI4PY

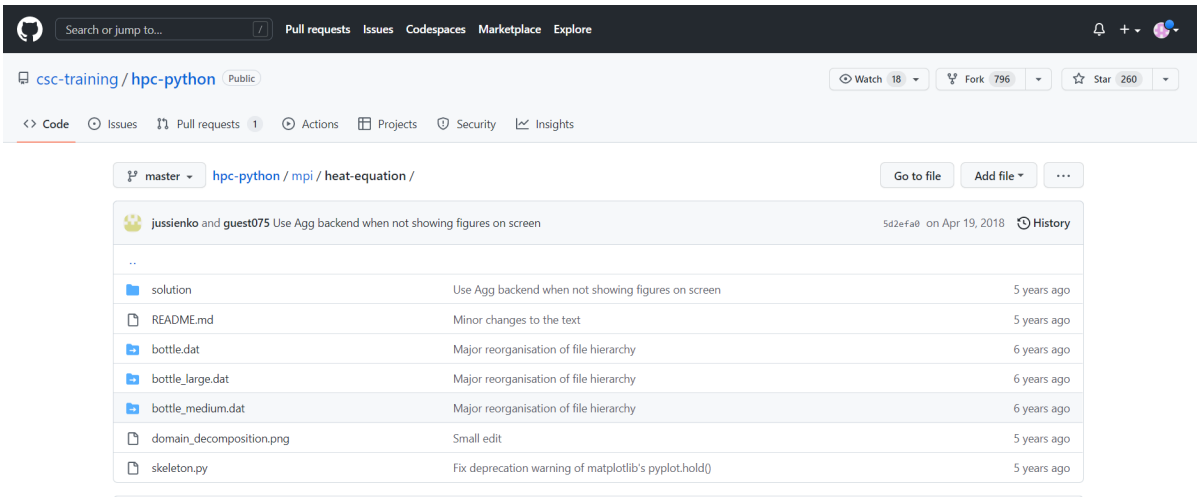
MPI - 概念

MPI4PY 编程基础

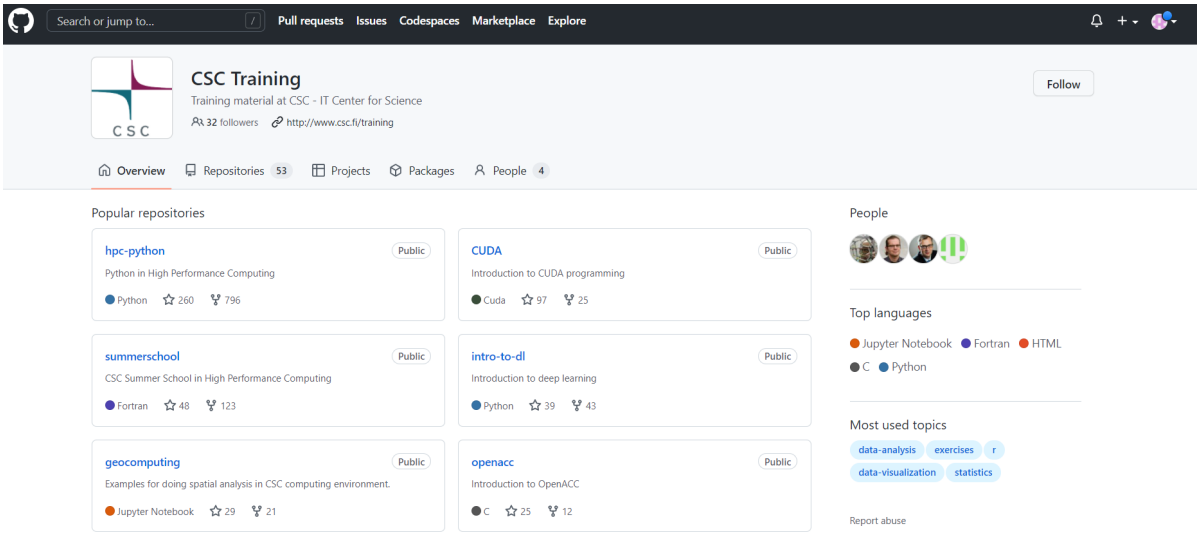
[Images] Stripe

线索

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




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



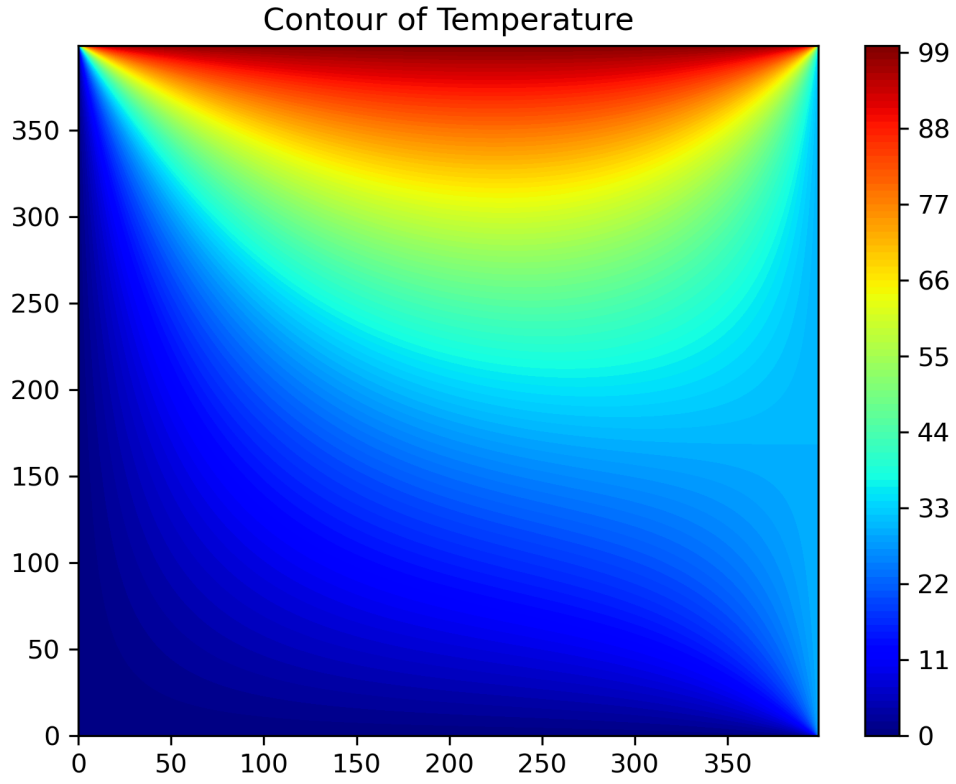

```
'''
https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2p.py
```

```
'''

from __future__ import print_function
import numpy as np
import time
from mpi4py import MPI

import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt

# Set the colormap
plt.rcParams['image.cmap'] = 'BrBG'

# Basic parameters
a = 0.5  # Diffusion constant
timesteps = 600  # Number of time-steps to evolve system
image_interval = 40000  # Write frequency for png files

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))

# MPI globals
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# Up/down neighbouring MPI ranks
up = rank - 1
if up < 0:
    up = MPI.PROC_NULL
down = rank + 1
if down > size - 1:
    down = MPI.PROC_NULL

def evolve(u, u_previous, a, dt, dx2, dy2):
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""

    u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
            (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
             u_previous[:-2, 1:-1]) / dx2 +
            (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
    number ' \
113                          + '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
    step
122
123     # Fix outer boundary ghost layers to account for aperiodicity?
124     if True:
125         if rank == 0:
126             local_field[0, :] = local_field[1, :]
127         if rank == size - 1:
128             local_field[-1, :] = local_field[-2, :]
129     local_field0[:] = local_field[:]
130
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]

```
'''
https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2p.py

"D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-heatequ2Dk.py"
'''
```

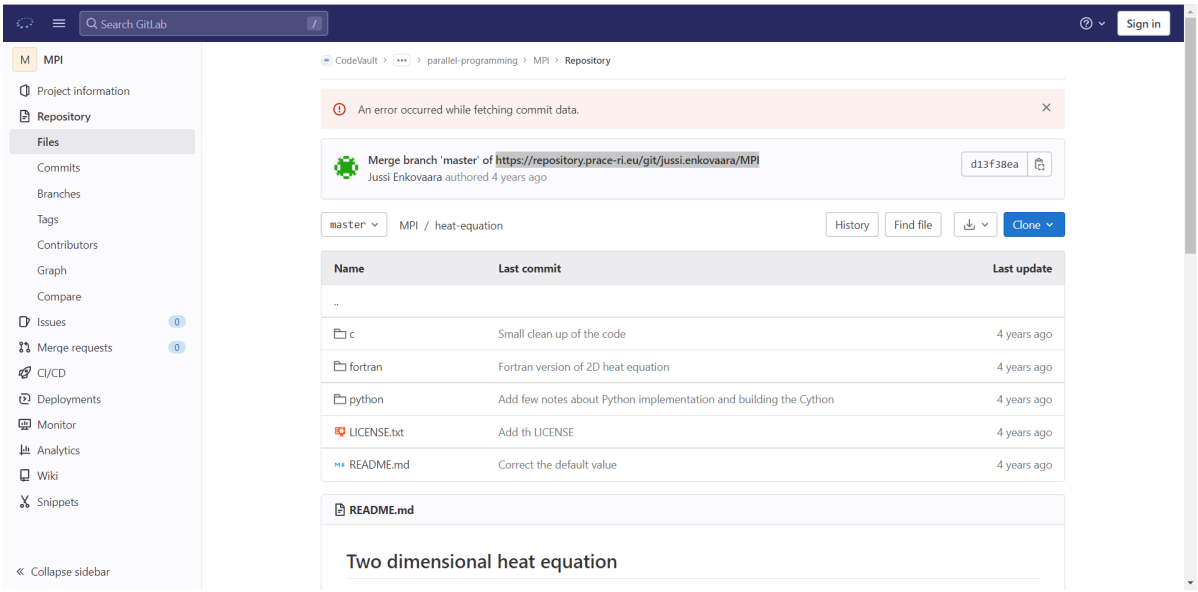
```
6
7  from __future__ import print_function
8  import numpy as np
9  import timeit
10
11
12 import matplotlib
13
14 matplotlib.use('Agg')
15 import matplotlib.pyplot as plt
16
17 # Set the colormap
18 # plt.rcParams['image.cmap'] = 'BrBG'
19 plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap =
   plt.cm.coolwarm
20 plt.figure(dpi=300)
21
22 # Set colour interpolation and colour map
23 colorinterpolation = 100
24 colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
25
26 # Set Dimension
27 lenX = lenY = 400 #we set it rectangular
28
29 # Set meshgrid
30 X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
31
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))
```

```
    field.fill(Tguess)

    # Set Boundary condition
    field[(lenY-1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX-1):] = Tright
    field[:, :1] = Tleft

    print("size is ",field.size)
    print(field,"\n")

    field0 = field.copy()  # Array for field of previous time step
    return field, field0

def evolve(u, u_previous, a, dt, dx2, dy2):
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""

    u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
            (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
             u_previous[:-2, 1:-1]) / dx2 +
            (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
             u_previous[1:-1, :-2]) / dy2)
    u_previous[:] = u[:]

def write_field(field, step):
    # plt.gca().clear()
    plt.cla()
    plt.clf()

    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    # Set Colorbar
    plt.colorbar()
    plt.axis('on')
    plt.savefig('heat_Seq_{0:03d}.png'.format(step))

def main():
    field, field0 = init_fields()
    write_field(field, 0)

    for m in range(1, timesteps + 1):
        evolve(field, field0, a, dt, dx2, dy2)
        if m % image_interval == 0:
            write_field(field, m)

if __name__ == '__main__':
    main()

```

[Images] Stencil

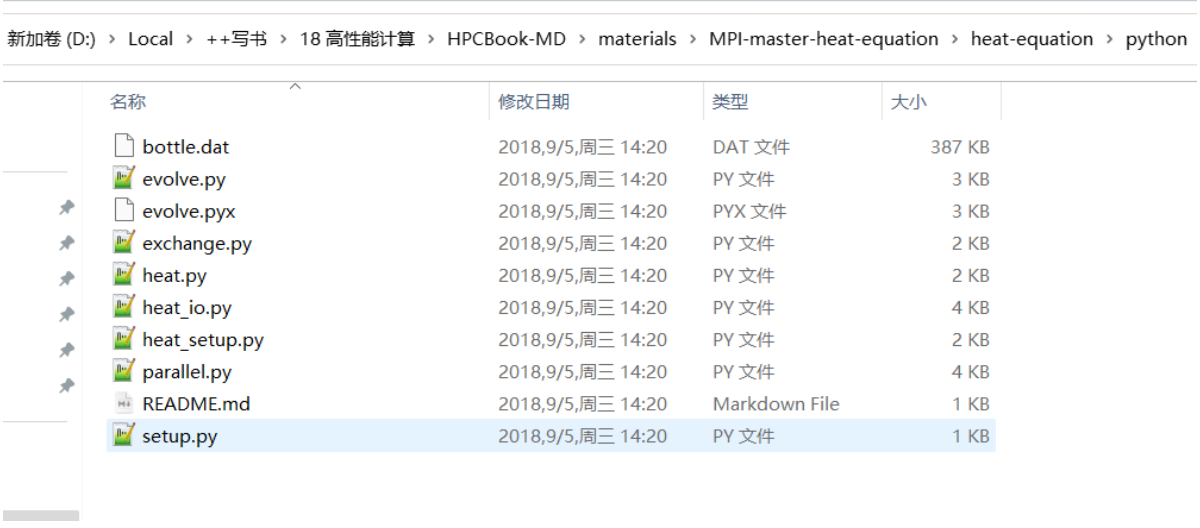
线索

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



本地：MPI-master.zip

MPI-master-heat-equation.zip



[我的代码]

```
"""
D:\myCodes\HPCbook\02mpi4py\ndArraySplitTestMPICart8.py
2023年6月5日21:07:20 终于解决了！！！
艰苦路程，参看 - "D:\Local\++写书\18 高性能计算\HPCBook-MD\05-MPI4py-B-stencil.md"

"""
from __future__ import division
from __future__ import print_function
import numpy as np
import time
import math
from mpi4py import MPI

import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt

# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap = plt.cm.coolwarm

# ++++++++++++++++ MPI ++++++++++++++++++++
def init_fields():

    # Set array size and set the interior value with Tguess
    field = np.zeros((lenX, lenY), dtype = np.float32)
    # field = np.linspace(1,lenX*lenY,lenX*lenY).reshape((lenX,lenY))
    # field = np.linspace(1,lenX*lenY,lenX*lenY).reshape((lenX,lenY))
    field.fill(Tguess)

    # Set Boundary condition
    field[(lenY-1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX-1):] = Tright
    field[:, :1] = Tleft

    print("size is ",field.size)
    print(field,"\n")

    return field

"""
借助 输入的 neighbour，可以确定
"""
def exchange(localField, neighbour):
    validDirections = np.where(np.array(neighbour) !=-1)
    # print("\nvalidDirections[0]\n", validDirections[0])
    for direction in validDirections[0]:
```

```
        rbuf = np.zeros(localField.shape[1], dtype = localField.dtype)
        if direction == 0: # Down
            sbuf = localField[-2, :]
            rbuf = localField[-1, :]
            comm.Sendrecv(np.array(sbuf), dest=neighbour[direction], recvbuf=rbuf, source=neighbour[direction])

        elif direction == 1: # Right
            sbuf = localField[:, -2]
            # rbuf = localField[:, -1]
            comm.Sendrecv(np.array(sbuf), dest=neighbour[direction], recvbuf=rbuf, source=neighbour[direction])
            localField[:, -1] = np.array(rbuf).T
        elif direction == 2: # Up
            sbuf = localField[1, :]
            rbuf = localField[0, :]
            comm.Sendrecv(np.array(sbuf), dest=neighbour[direction], recvbuf=rbuf, source=neighbour[direction])

        else: # Left
            sbuf = localField[:, 1]
            # rbuf = localField[:, 0]
            comm.Sendrecv(np.array(sbuf), dest=neighbour[direction], recvbuf=rbuf, source=neighbour[direction])
            localField[:, 0] = np.array(rbuf).T

def evolve(u, u_previous, a, dt, dx2, dy2, neighbour):
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""
    # neigh = [Down, Right, Up, Left]
    up = 1
    down = u.shape[0]-1
    right = u.shape[1]-1
    left = 1

    validDirections = np.where(np.array(neighbour) ==-1)
    for direction in validDirections[0]:
        if direction == 0: # Down
            down = u.shape[0]-2
        elif direction == 1: # Right
            right = u.shape[1]-2
        elif direction == 2: # Up
            up = 2
        else: # Left
            left = 2

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

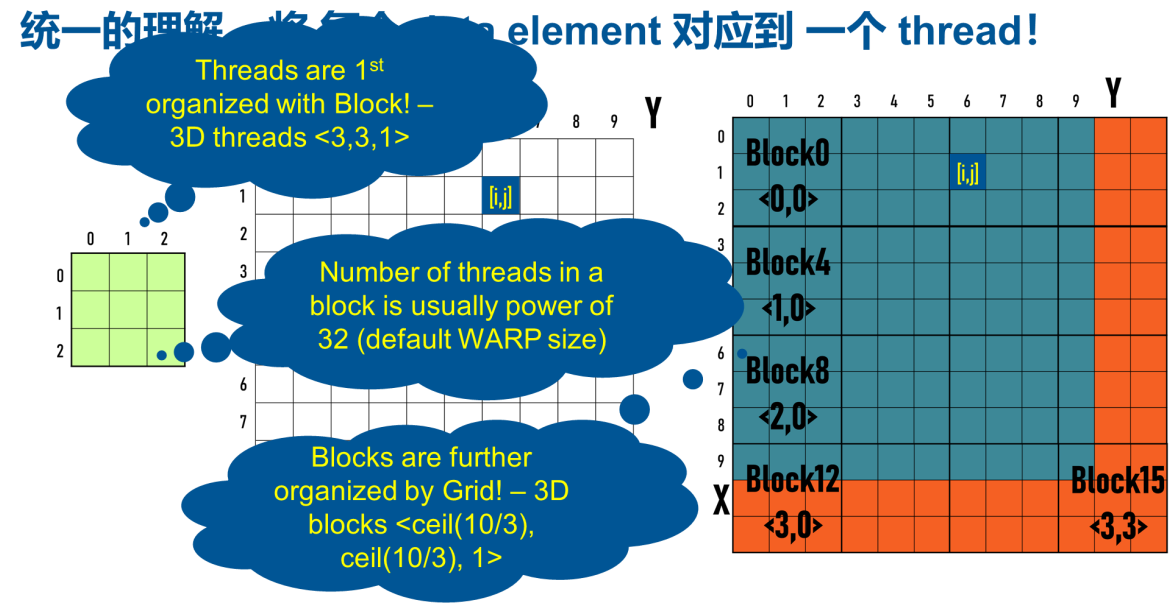
```
# 400x400
# 16 = 4x4，那么，每个local 就是 100x100
mpi_rows = mpi_cols = 4
# 也就意味着 mpiexec -np number 中的number 一定要等于mpi_rows*mpi_cols = 9
hvGrid = (mpi_rows, mpi_cols)

# Basic parameters
a = 0.1  # Diffusion constant  #例如 Thermal diffusivity of steel, mm2.s-1
timesteps = 10000  # Number of time-steps to evolve system
image_interval = 1000  # Write frequency for png files

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))

# Set Dimension and delta
delta = 1.

# Boundary condition
Ttop = 100.
Tbottom = 0.
Tleft = 0.
Tright = 30.

# Initial guess of interior grid
Tguess = 0.

# MPI globals
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# print("Creating a {} x {} processor grid...".format(mpi_rows, mpi_cols) )
ccomm = comm.Create_cart((mpi_rows, mpi_cols), periods=(False, False),
reorder = False)

if rank == 0:
    oringin = init_fields()
    # field = np.linspace(1,lenX*LenY,lenX*LenY).reshape((lenX,LenY))
    # print("size is ",oringin.size)
    # print(oringin,"\n")


    full_split = hvSplit(oringin, hvGrid)

    # print('\nb_fullSplit: ')
    # print(full_split)

    bbb = hvFlatten(full_split, hvGrid)
    # print('\nbbb: ', bbb)
```

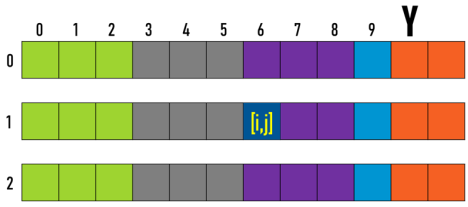
```
201
202     field = np.array(bbb).reshape(dataShape)
203     # print('\nfield for scatter(): \n', field, "\n", field.dtype)
204
205     # Set colour interpolation and colour map
206     colorinterpolation = 100
207     colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
208
209     # Set meshgrid
210     X, Y = np.meshgrid(np.arange(0, field.shape[0]), np.arange(0,
    field.shape[1]))
211
212     # hvField = reverseHVFlatten(field, dataShape, hvGrid)
213     # write_field(hvField, 0)
214     write_field(oringin, 0)
215
216     shape = field.shape
217     dtype = field.dtype
218     localShape = (dataShape[0]//hvGrid[0],dataShape[1]//hvGrid[1])
219     # print(localShape)
220     # print('\ndtype = {}'.format(dtype))
221
222     comm.bcast(shape, root = 0)  # broadcast dimensions
223     comm.bcast(dtype, root = 0)  # broadcast data type
224     comm.bcast(localShape, root = 0)
225 else:
226     field =  None
227     full_split = None
228     shape = comm.bcast(None, 0)
229     dtype = comm.bcast(None, 0)
230     localShape = comm.bcast(None, 0)
231
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("\nits neighbour is {}\n".format(neigh))
252
253 local_field = np.zeros((localShape[0]+2, localShape[1]+2), dtype)  # need 4
    ghost rows!
```

```
# print("\n local_field:\n", local_field)
local_field[1:-1, 1:-1] = buff[:]  # copy data to non-ghost rows
# print("\n local_field after buff:\n", local_field)
local_field0 = np.zeros_like(local_field)  # array for previous time step

# print("\nlocal_field.shape", local_field.shape)
# print("\n", local_field)
# print("\n", local_field[0][0])

# exchange(local_field,neigh)
local_field0[:] = local_field[:]
# print("\nafter local_field0[:] = local_field[:]\n", local_field0)

# Iterate
t0 = time.time()
for m in range(1, timesteps + 1):
    exchange(local_field, neigh)
    evolve(local_field, local_field0, a, dt, dx2, dy2, neigh)

    if m % image_interval == 0:
        # exchange(local_field, neigh)
        # evolve(local_field, local_field0, a, dt, dx2, dy2, neigh)
        # print("\nlocal_field after evolve():\n", local_field)
        # print("\nBefore comm.Gather()")
        comm.Gather(np.array(local_field[1:-1, 1:-1],dtype), field, root =
0)

        if rank == 0:
            # comm.Igather(local_fieldGather, field, root = 0)
            # print("\nGathered field: \n", field,"\n")
            hvField = reverseHVFlatten(field, dataShape, hvGrid)
            write_field(hvField, m)
            # write_field(field, m)
            # if m == timesteps:
                # print("\nGathered field: \n", hvField,"\n")
t1 = time.time()

# Plot/save final field
# comm.Gather(local_field[1:-1, 1:-1], field, root=0)
# comm.Gather(local_field, field, root=0)

if rank == 0:
    # print('\nfield shape:')
    # print(field.shape)
    # write_field(field, 10)
    print("Running time: {0}".format(t1 - t0))
```

PyCUDA

CUDA 编程基本概念



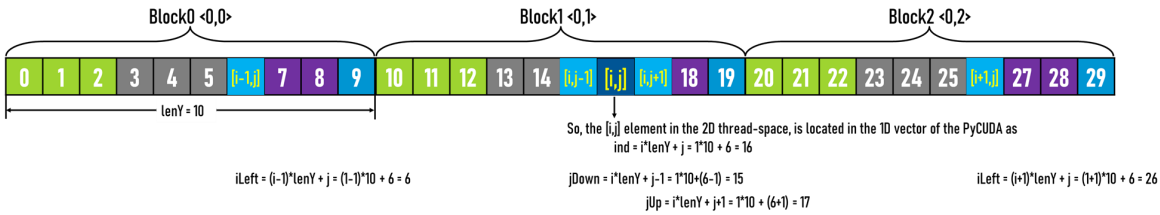



```
int i = blockIdx.x * blockDim.x + threadIdx.x;
int j = blockIdx.y * blockDim.y + threadIdx.y;
```

```
i = 0*3 + 1 = 1;
j = 2*3 + 0 = 6;
```

计算时，自然要将 thread[i,j] 只对应实际的数据！

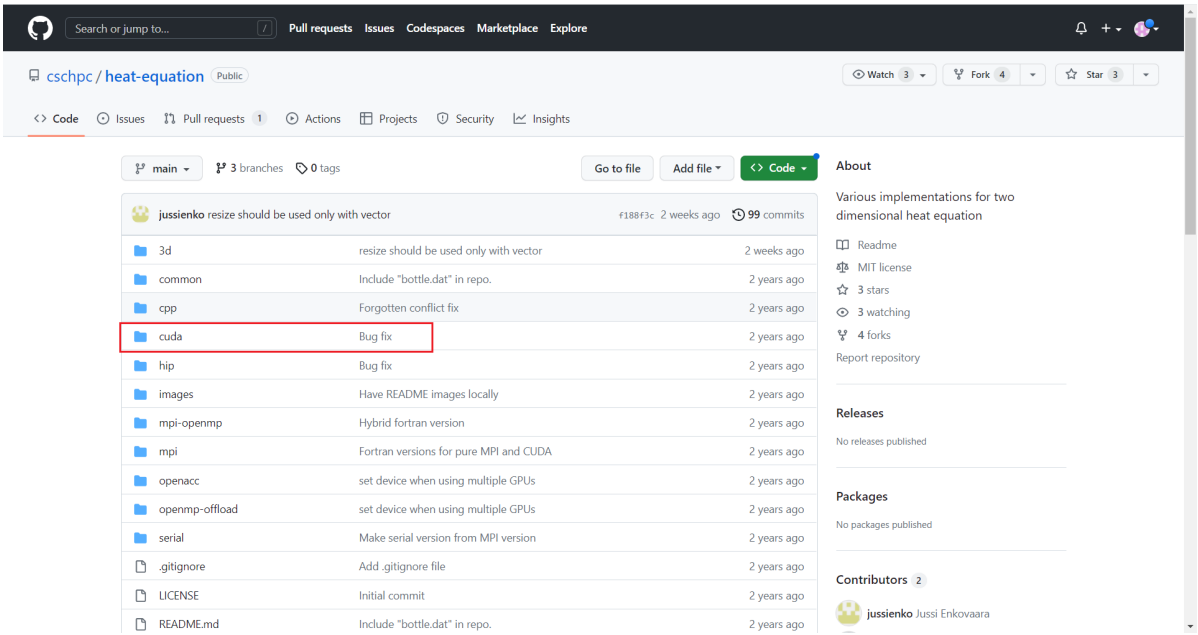
```
For i>=0 & j>=0 & i<=10-1 & j<=10-1
        thread[i,j] + = ...
```

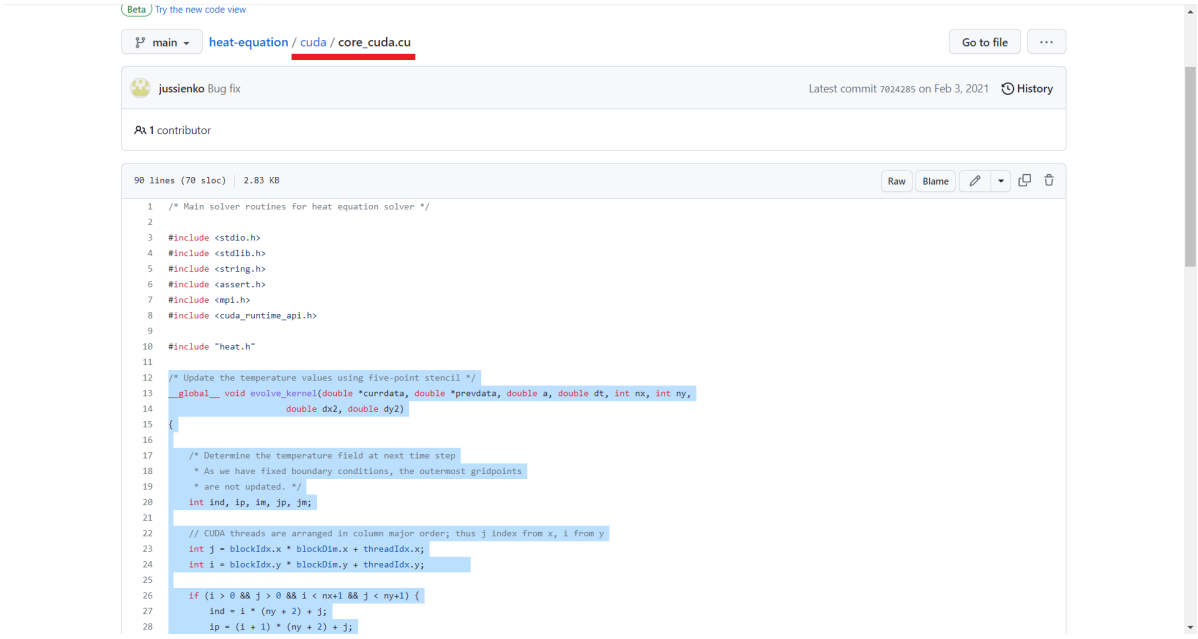


[Images]PyCUDA

线索

https://github.com/cschpc/heat-equation





我的实践

- 人家的CUDA kernel 代码可以直接拷贝至 PyCUDA中

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <mpi.h>
#include <cuda_runtime_api.h>

#include "heat.h"

/* Update the temperature values using five-point stencil */
__global__ void evolve_kernel(double *currdata, double *prevdata, double a, double dt, int nx, int ny,
                              double dx2, double dy2)
{

    /* Determine the temperature field at next time step
     * As we have fixed boundary conditions, the outermost gridpoints
     * are not updated. */
    int ind, ip, im, jp, jm;

    // CUDA threads are arranged in column major order; thus j index from x, i from y
    int j = blockIdx.x * blockDim.x + threadIdx.x;
    int i = blockIdx.y * blockDim.y + threadIdx.y;

    if (i > 0 && j > 0 && i < nx+1 && j < ny+1) {
        ind = i * (ny + 2) + j;
        ip = (i + 1) * (ny + 2) + j;
        im = (i - 1) * (ny + 2) + j;
        jp = i * (ny + 2) + j + 1;
        jm = i * (ny + 2) + j - 1;
        currdata[ind] = prevdata[ind] + a * dt *
              ((prevdata[ip] -2.0 * prevdata[ind] + prevdata[im]) / dx2 +
              (prevdata[jp] - 2.0 * prevdata[ind] + prevdata[jm]) / dy2);

    }

}

void evolve(field *curr, field *prev, double a, double dt)
{
    int nx, ny;
    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！

```python
dt = np.float32(dt)
lenX = np.int32(lenX)
field = field.astype(np.float32)

```

[我的代码]

```python
'''
https://github.com/csc-training/hpc-python/blob/master/mpi/heat-equation/solution/heat-p2p.py

鍩轰簬 "D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\mpi4py-heatequ2Dk.py" 杩涜淇敼銆?

2023骞?鏈?1鏃?1:29:07 鎴愬姛鐨勮繍琛屼簡
鍘熷鍦板潃锛?"D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\examplesHeatMPI\simpleFDM-PyCUDA.py"
鏂囨。璺緞 "D:\Local\++鍛戒护\18 楂樻€ц兘璁＄畻\HPCBook-MD\05-GPU-A-PyCUDA.md"

'''

from __future__ import print_function
import numpy as np
import time
import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt

```

```
# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
plt.rcParams['image.cmap'] = 'jet'  # you can try: colourMap =
plt.cm.coolwarm
# ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Set colour interpolation and colour map
colorinterpolation = 100
colourMap = plt.cm.jet  # you can try: colourMap = plt.cm.coolwarm

import pycuda.autoinit
import pycuda.driver as cuda
from pycuda import gpuarray
from pycuda.compiler import SourceModule

# Basic parameters
a = 0.1  # Diffusion constant


timesteps = 10000  # Number of time-steps to evolve system
image_interval = 1000  # Write frequency for png files

# Grid spacings
dx = 0.01
dy = 0.01
dx2 = dx ** 2
dy2 = dy ** 2

# For stability, this is the largest interval possible
# for the size of the time-step:
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))

# Set Dimension and delta
lenX = lenY = 400  # we set it rectangular
delta = 1

# Boundary condition
Ttop = 100
Tbottom = 0
Tleft = 0
Tright = 30

# Initial guess of interior grid
Tguess = 0

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))

def write_field(field, step):
    plt.gca().clear()
    # plt.clf()

    plt.figure(dpi=300)
    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    # Set Colorbar
```

```
    plt.colorbar()
    plt.axis('on')
    plt.savefig('heat_PyCUDA_{0:03d}.png'.format(step))


a = np.float32(a)
lenX = np.int32(lenX)
lenY = np.int32(lenY)
dx = np.float32(dx)
dy = np.float32(dy)
dx2 = np.float32(dx2)
dy2 = np.float32(dy2)
dt = np.float32(dt)
timesteps = np.int32(timesteps)
image_interval = np.int32(image_interval)

#
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++
# evolve_kernel 鏃ヨ嚍 D:\myCodes\HPCprojects\heat-equation-
main\cuda\core_cuda.cu
ker = SourceModule('''
/* Update the temperature values using five-point stencil */
__global__ void evolve_kernelCUDA(float *currdata, float *prevdata, float
a, float dt, int nx, int ny,
                        float dx2, float dy2)
{

    /* Determine the temperature field at next time step
     * As we have fixed boundary conditions, the outermost gridpoints
     * are not updated. */
    int ind, iRight, iLeft, jUp, jDown;

    // CUDA threads are arranged in column major order; thus j index from
x, i from y
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;

    if (i > 0 && j > 0 && i < nx-1 && j < ny-1) {
        ind = i * ny + j;
        iRight = (i+1)  * (ny) + j;
        iLeft = (i - 1) * (ny ) + j;
        jUp = i * (ny) + j + 1;
        jDown = i * (ny) + j - 1;
        currdata[ind] = prevdata[ind] + a * dt *
          ((prevdata[iRight] -2.0 * prevdata[ind] + prevdata[iLeft]) / dx2
+
          (prevdata[jUp] - 2.0 * prevdata[ind] + prevdata[jDown]) / dy2);
    }
}
''')

evolve_kernel = ker.get_function('evolve_kernelCUDA')


# 浠ｇ爜鏃ヨ嚍 D:\myCodes\HPCprojects\heat-equation-main\cuda\core_cuda.cu
```

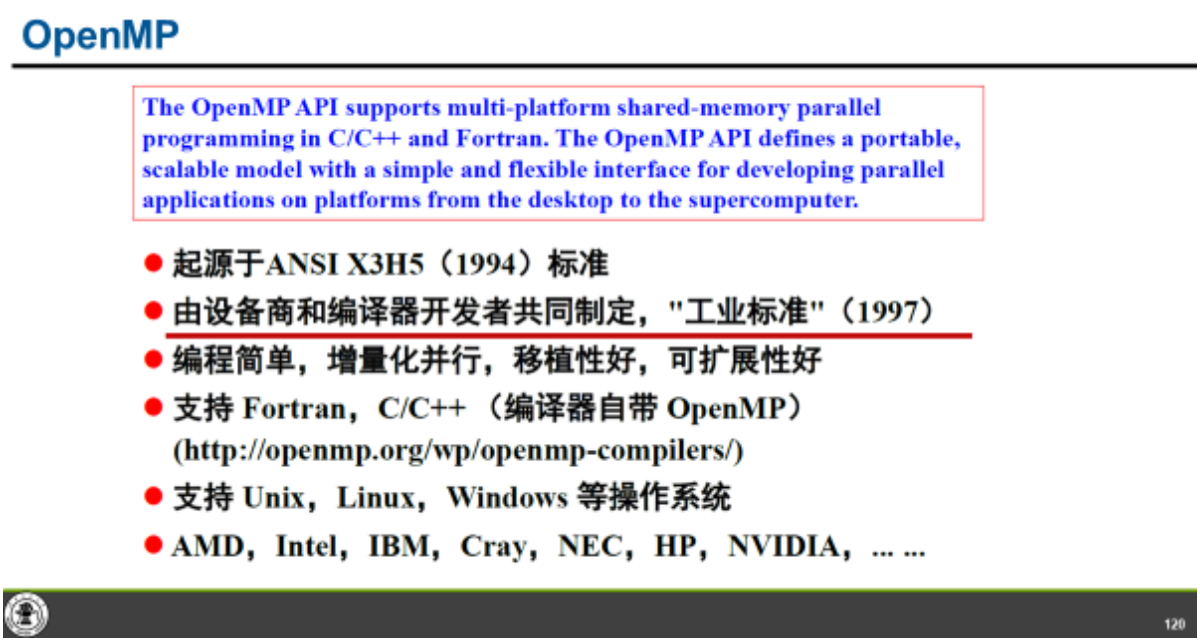
```
# void evolve(field *curr, field *prev, double a, double dt)
# {
# int nx, ny;
# double dx2, dy2;
# nx = prev->nx;
# ny = prev->ny;
# dx2 = prev->dx * prev->dx;
# dy2 = prev->dy * prev->dy;

# /* CUDA thread settings */
# const int blocksize = 16;  //!< CUDA thread block dimension
# dim3 dimBlock(blocksize, blocksize);
# // CUDA threads are arranged in column major order; thus make ny x nx grid
# dim3 dimGrid((ny + 2 + blocksize - 1) / blocksize,
# (nx + 2 + blocksize - 1) / blocksize);

# evolve_kernel<<<dimGrid, dimBlock>>>(curr->devdata, prev->devdata, a, dt, nx, ny, dx2, dy2);
# cudaDeviceSynchronize();
# }


#
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def main():

    # Set array size and set the interior value with Tguess
    field = np.empty((lenX, lenY))
    field.fill(Tguess)


    # Set Boundary condition
    field[(lenY - 1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX - 1):] = Tright
    field[:, :1] = Tleft

    # gpuarray.to_gpu(field, allocator=None)
    # field = gpuarray.to_gpu(np.random.randn(lenX,lenX).astype(np.float32))
    # field_gpuArr = gpuarray.to_gpu(field.astype(np.float32))

    print("field's size is ", field.size)
    print(field, "\n")

    field0 = field.copy()  # Array for field of previous time step
    print("field0's size is ", field0.size)
    print(field0, "\n")


    write_field(field, 0)

    field = field.astype(np.float32)
    field0 = field0.astype(np.float32)
```

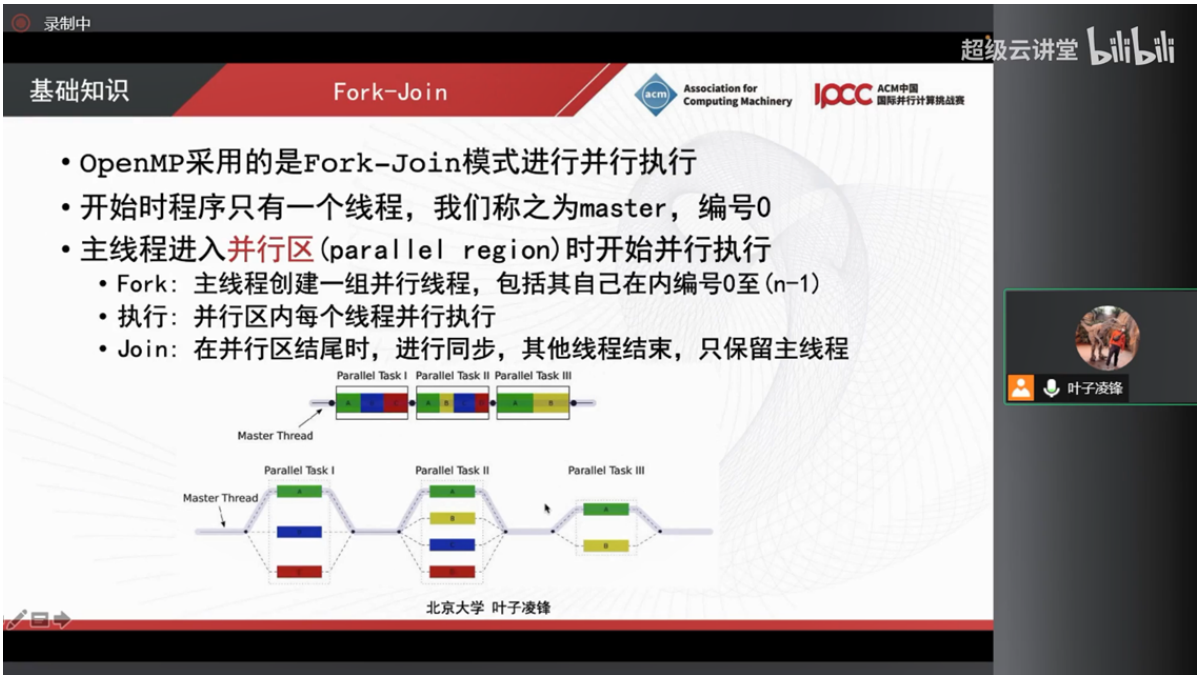
```
177
178     blocksize = 32  # !< CUDA thread block dimension
179     dimBlock = (blocksize, blocksize, 1)
180     # CUDA threads are arranged in column major order; thus make ny x nx
    grid
181     # dimGrid = ((lenY + 2 + blocksize - 1) // blocksize,
182               # (lenX + 2 + blocksize - 1) // blocksize,
183               # 1)
184
185
186     dimGrid = (int(lenX/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,
    block=dimBlock, grid=dimGrid)
205         # cuda.cudaDeviceSynchronize()
206         # cuda.Context.synchronize()
207
208         if (m % image_interval == 0):
209             # Copy back the result
210             cuda.memcpy_dtoh(field, field_gpu)
211             write_field(field, m);
212             # print(field, "\n")
213
214         cuda.memcpy_dtoh(field, field_gpu)
215         cuda.memcpy_dtoh(field0, field0_gpu)
216
217         # Swap current field so that it will be used as previous for next
    iteration step
218         tmp = field
219         field = field0
220         field0 = tmp
221
222     t1 = time.time()
223     print("Running time: {0}".format(t1 - t0))
224
225     field_gpu.free()
226     field0_gpu.free()
227
228 if __name__ == '__main__':
229     main()
```

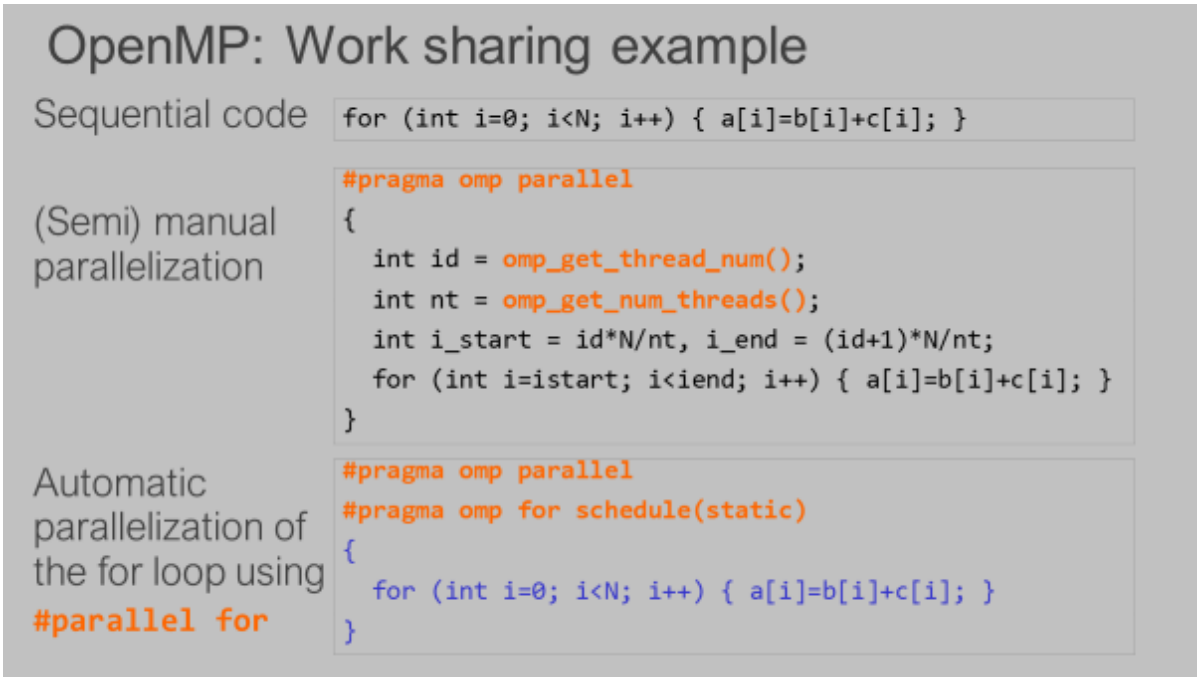
```
230
231
```

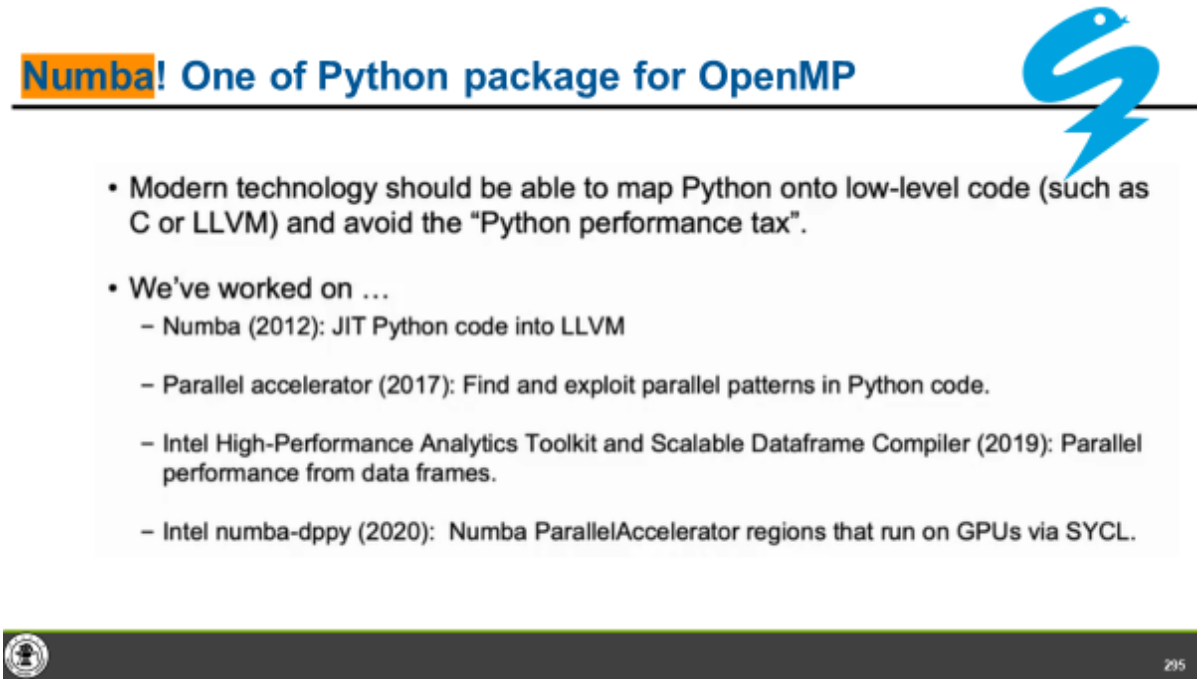
Numba

从OpenMP 到 Python 的 Numba







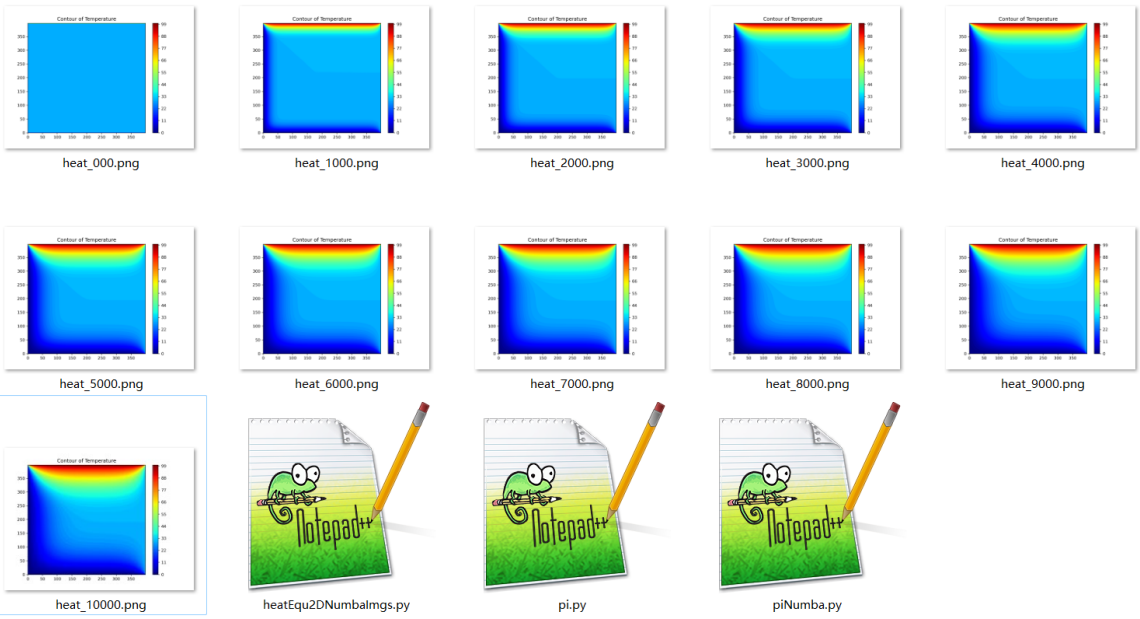


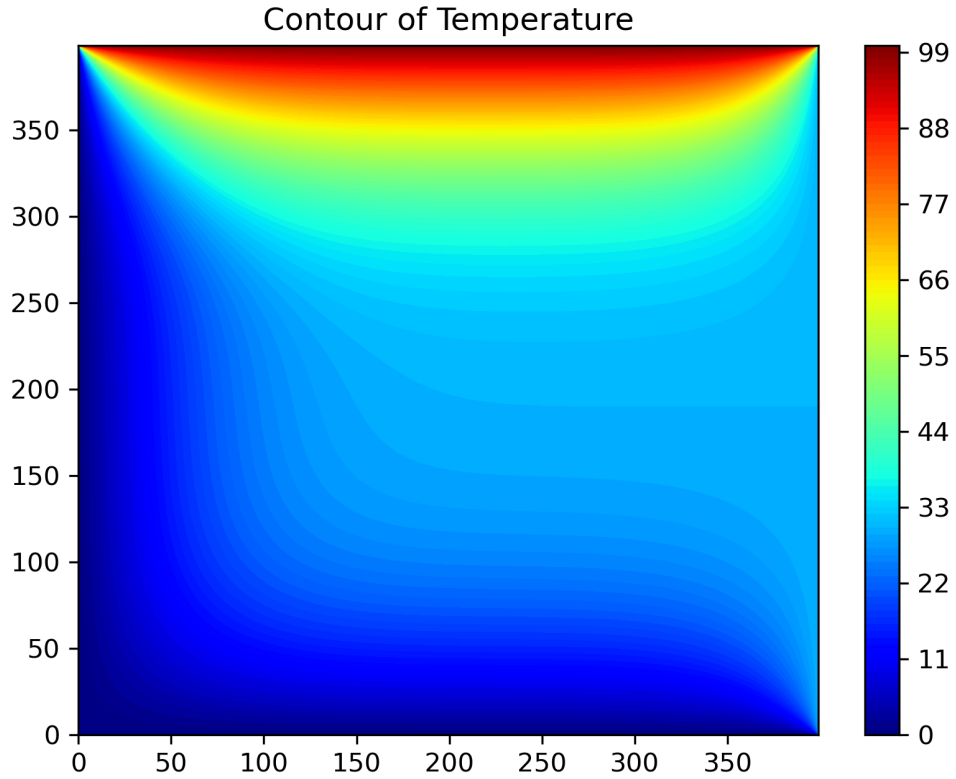
[Images]Numba

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`

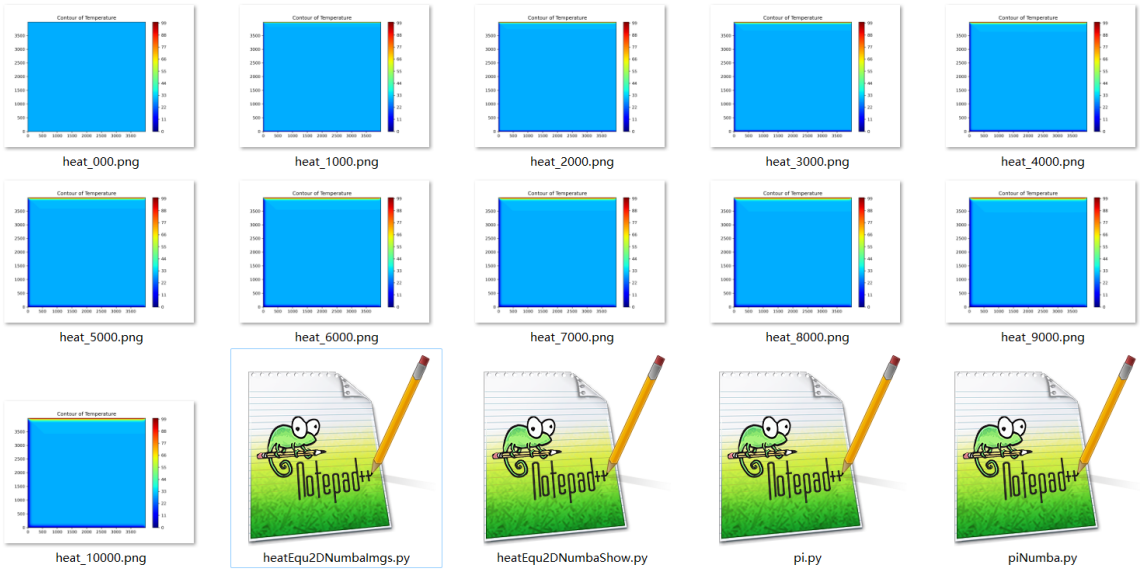
我的代码的效果





但是，当数据是 4000*4000 时，就出问题了

```python
# Set Dimension and delta
lenX = lenY = 4000 #we set it rectangular

@njit(fastmath=True)
def evolve(u, u_previous, lenX, lenY, a, dt, dx2, dy2):
```



为什么?

[我的代码]

```
""" https://www.codeproject.com/Articles/1087025/Using-Python-to-Solve-Computational-Physics-Proble
# D:\My7\MyProjects\Memoirs\2018-12-18-Heat Transfer with Python.docx
# 2019骞?鏈?鏃?3:03:49

2023骞?鏈?7鏃?5:56:13
D:\myCodes\HPCprojects\SourceCodes\parallel_python-master\examplesHeatMPI\simpleFDM-Numba1.py
"""
# Simple Numerical Laplace Equation Solution using Finite Difference Method
import numpy as np
from numba import jit, njit, prange
import timeit


import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt

# Set the colormap
# plt.rcParams['image.cmap'] = 'BrBG'
plt.rcParams['image.cmap'] = 'jet' #you can try: colourMap = plt.cm.coolwarm
plt.figure(dpi=300)

# Set colour interpolation and colour map
colorinterpolation = 100
colourMap = plt.cm.jet #you can try: colourMap = plt.cm.coolwarm
```

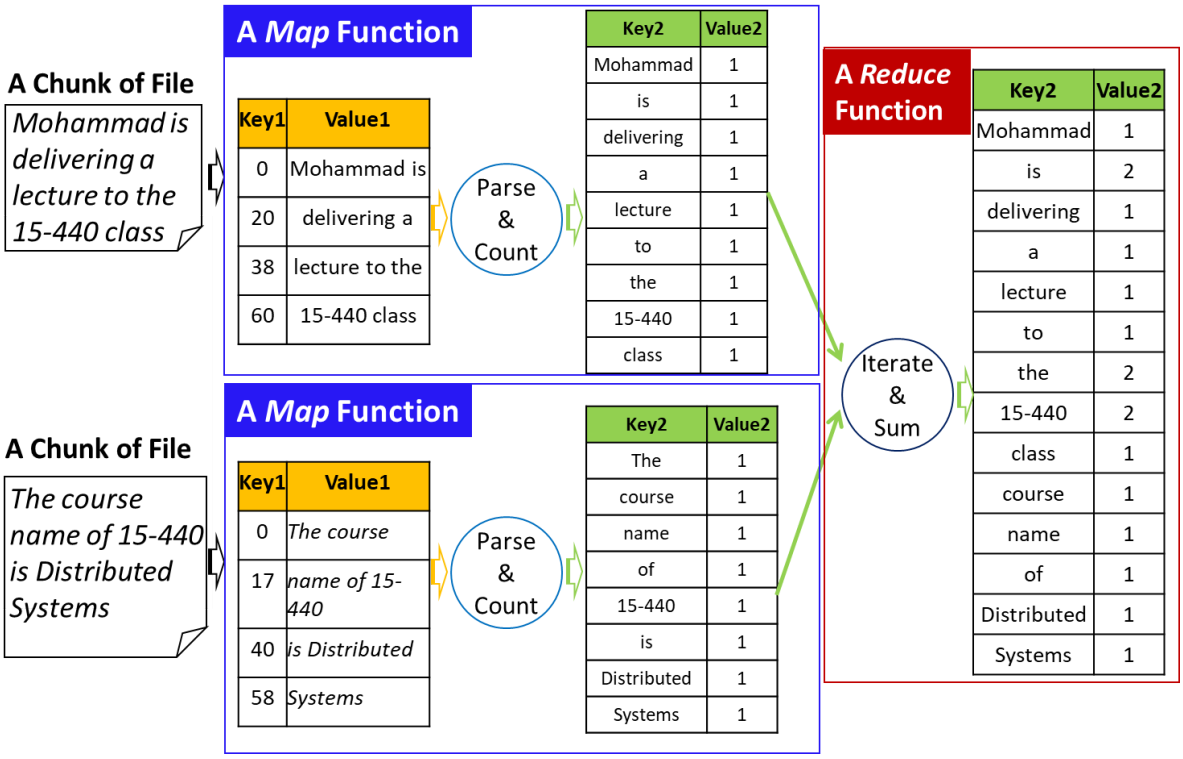
```
27
28
29  # Basic parameters
30  a = 0.1  # Diffusion constant
31  timesteps = 10000  # Number of time-steps to evolve system
32  image_interval = 1000  # Write frequency for png files
33
34  # Grid spacings
35  dx = 0.01
36  dy = 0.01
37  dx2 = dx ** 2
38  dy2 = dy ** 2
39
40  # For stability, this 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
```

```
# @njit(fastmath=True)
@njit(fastmath=True)
def evolve(u, u_previous, lenX, lenY, a, dt, dx2, dy2):
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""
    for i in range(1, lenX-1, delta):
        for j in range(1, lenY-1, delta):
            u[i, j] = u_previous[i, j] + a * dt * (
                (u_previous[i+1, j] - 2 * u_previous[i, j] +
                 u_previous[i-1, j]) / dx2 +
                (u_previous[i, j+1] - 2 * u_previous[i, j] +
                 u_previous[i, j-1]) / dy2)
    u_previous[:] = u[:]


def write_field(field, step):
    # plt.gca().clear()
    plt.cla()
    plt.clf()

    plt.figure(dpi=300)
    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    # Set Colorbar
    plt.colorbar()
    plt.axis('on')
    plt.savefig('heat_Numba_{0:03d}.png'.format(step))


def main():
    field, field0 = init_fields()
    write_field(field, 0)

    print("size is ",field.size)
    print(field,"\n")

    # Iteration (We assume that the iteration is convergence in maxIter =
500)
    print("Please wait for a moment")

    starting_time = timeit.default_timer()
    for iteration in range(0, timesteps+1):
        evolve(field, field0, lenX, lenY, a, dt, dx2, dy2)
        if iteration % image_interval == 0:
            write_field(field, iteration)

    print("Iteration finished. {} Seconds for Time
difference:".format(timeit.default_timer() - starting_time))

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]

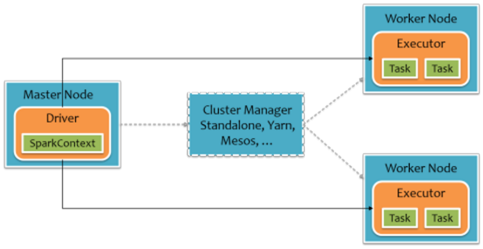


PySpark 编程基础

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())
```

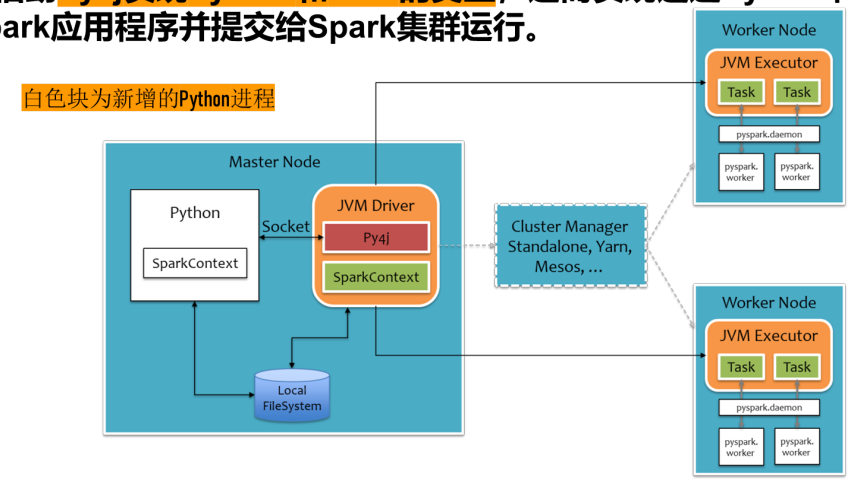


- Driver负责总体的调度，Executor负责具体Task的运行，它们都是运行在JVM进程之中的，而这些JVM进程则是可以部署在多种的资源管理系统中的，比如Yarn、Mesos或者是K8s等；
- 用户提交的Spark程序交给Driver进行管理，Driver将程序分解为一个个的Task交给Executor执行。

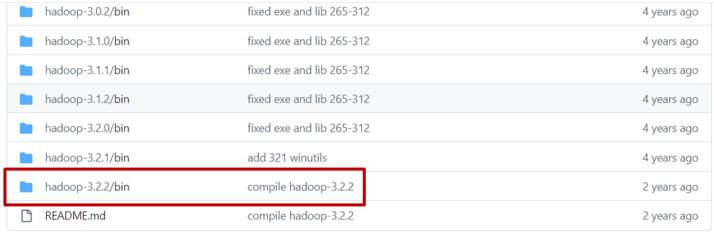
https://www.jianshu.com/p/eaab74e34dae

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



README.md

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:

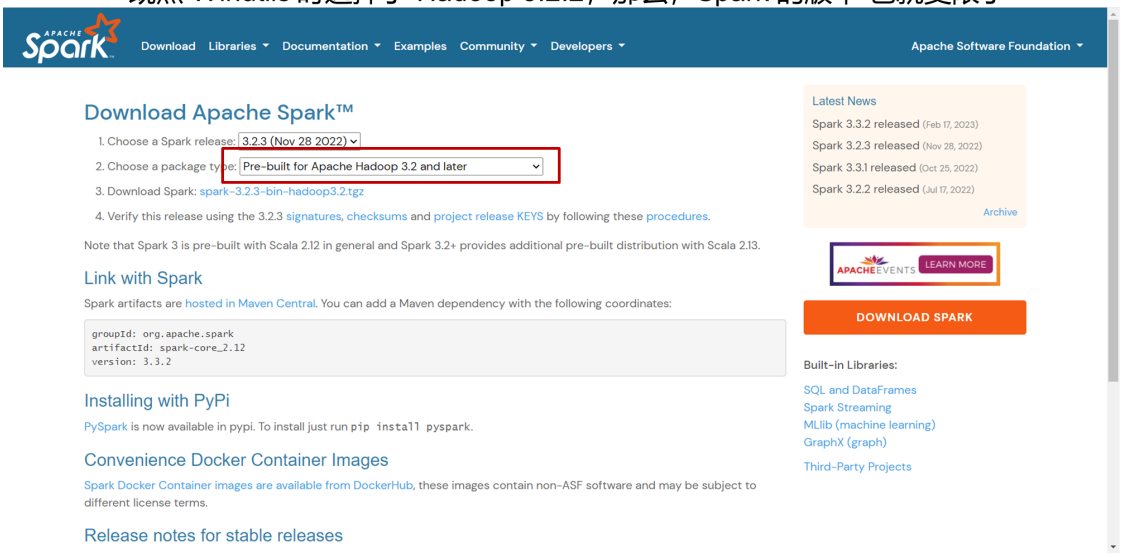
```
HADOOP_HOME=<your local hadoop-ver folder>
PATH=%PATH%;%HADOOP_HOME%\bin
```

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

Spark?

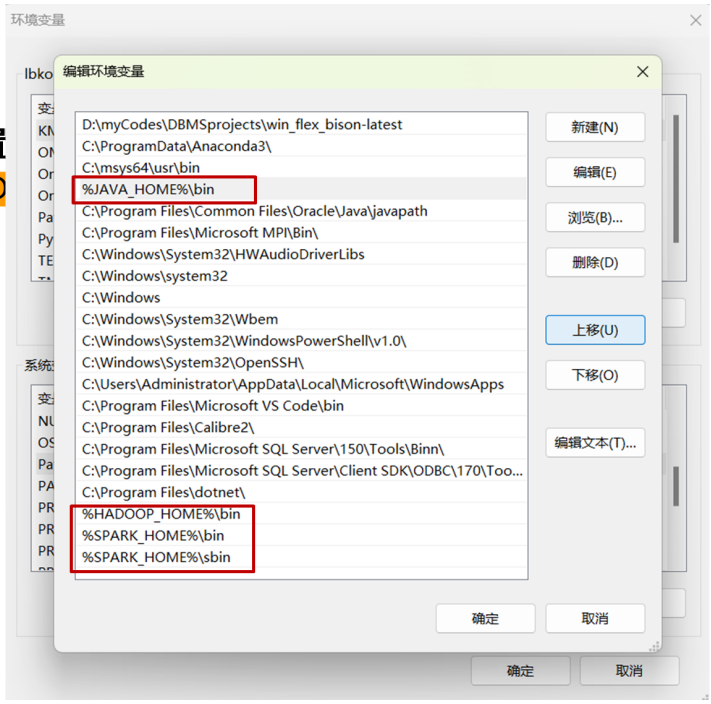
https://spark.apache.org/downloads.html

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



Windows 的 环境配置

- JAVA_HOME, HADO
- 别忘了 更新 PATH




```python
"""
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()

```

```python
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()

```

```python
"""
https://blog.csdn.net/qq_42582489/article/details/106579484
"""
from pyspark import SparkConf, SparkContext

# 创建 SparkConf 和 SparkContext
conf = SparkConf().setMaster("local[4]").setApp
sc = SparkContext(conf=conf).getOrCreate()

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

def g(x):
    print(x)

# 将 Collection 的 data 转化为 spark 中的 rdd 并
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']

helloe 0:>                                                          (0 + 4) / 4]
count
hello
world
hello
word
count

('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。
```

```python
"""
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)
rdd.foreach(g)
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']

counte 0:>                                                          (0 + 4) / 4]
hello
hello
world
hello
word
count

('count', 1)
('hello', 1)
('hello', 1)                                                        (0 + 4) / 4]
('word', 1)
('count', 1)
('world', 1)
('hello', 1)

('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。

D:\myCodes\HPCprojects\SourceCodes\pyspark>
```

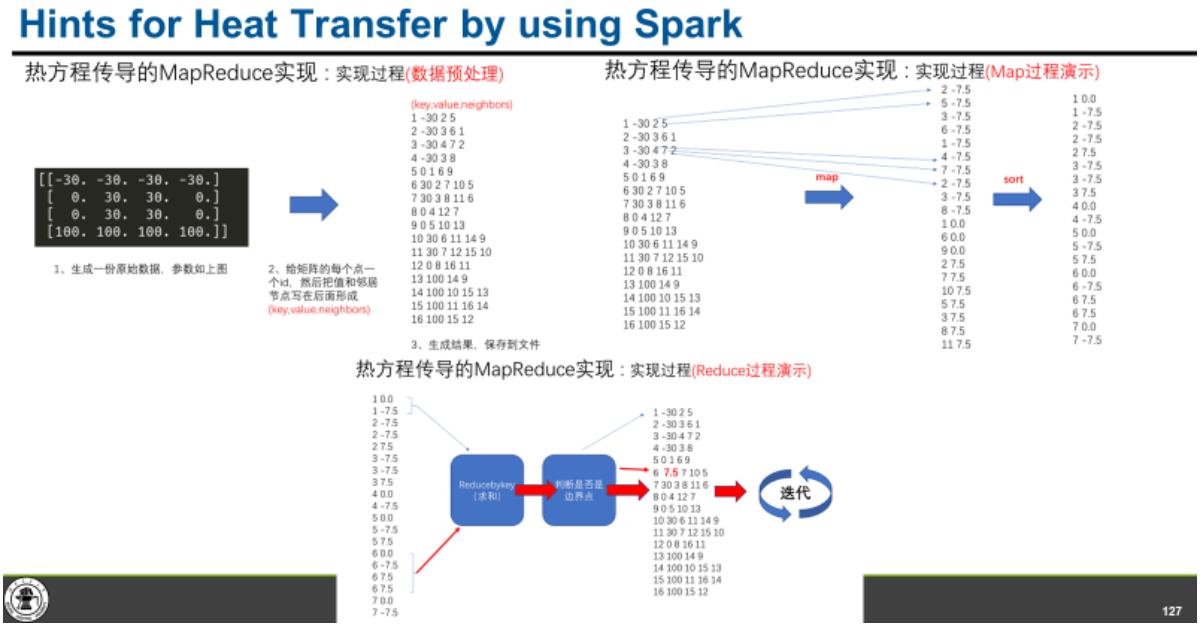
[Images] PySpark

线索

仔细体会 PageRank在Spark上的实现！

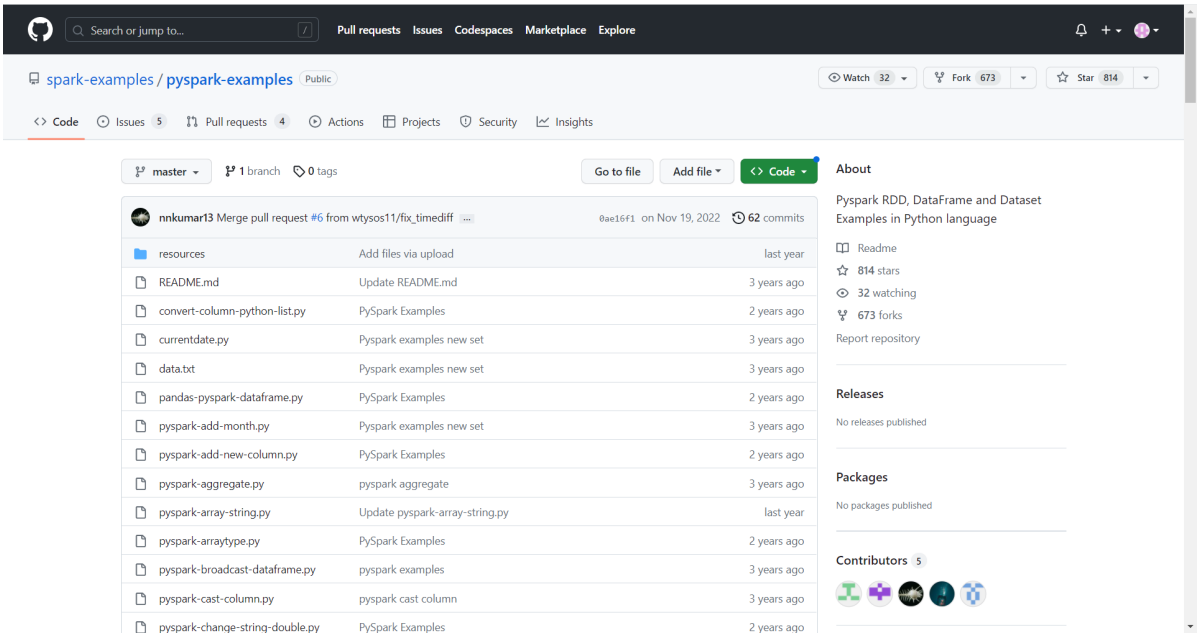
- Spark 自带的 Example

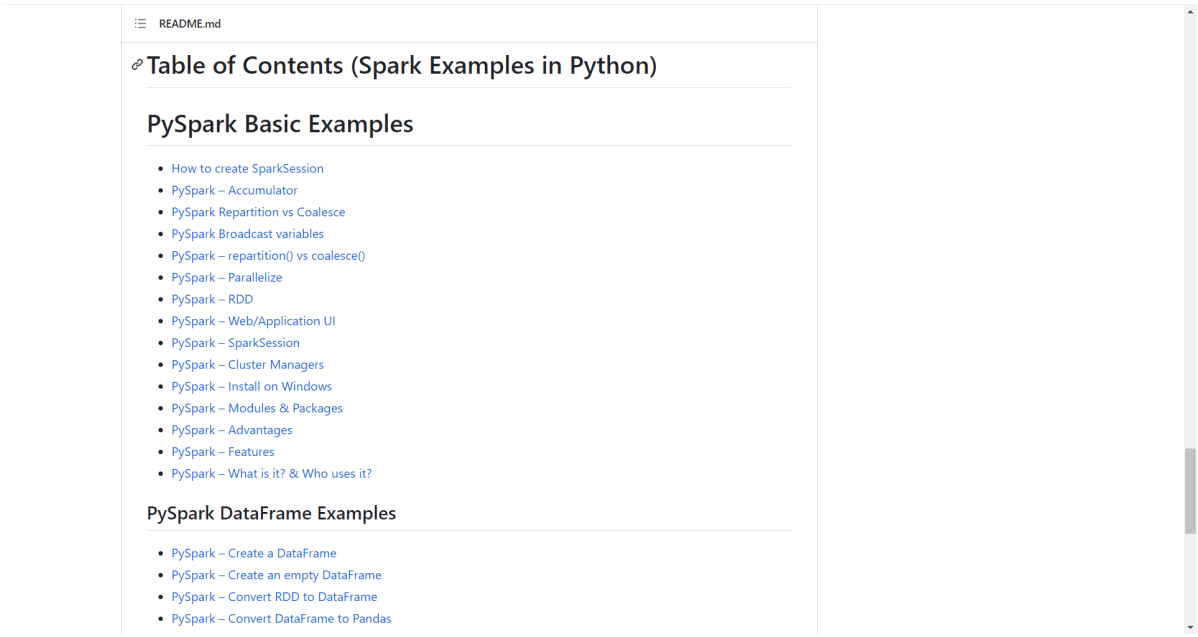
```
python pagerank.py "ml\data\mllib\pagerank_data.txt" 10
WARN: This is a naive implementation of PageRank and is given as an example!
Please refer to PageRank implementation provided by graphx
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use 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>
```



[学习 PySpark 的代码]

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






```
"""
https://stackoverflow.com/questions/36198264/convert-numpy-matrix-into-pyspark-rdd

mat = np.arange(100).reshape(10, -1)
rdd = sc.parallelize(mat)

np.all(rdd.first() == mat[0])
## True

https://zhuanlan.zhihu.com/p/619901768

"""

from __future__ import print_function
import numpy as np
import pyspark

sc = pyspark.SparkContext.getOrCreate()
broadcastVar = (0, 9, 0.5, 0.1)
```

```python

def g(x):
    print(x)

def func(tup: tuple) -> list:
    """主要作用是把每个值打上行列index
    """
    Array, row_index = tup
    return [[row_index, col_index, value] for col_index, value in enumerate(Array)]

def diffuse(lst: list) -> list:
    global broadcastVar
    res = []

    if lst[0] > broadcastVar[0] and lst[0] < broadcastVar[1] and lst[1] > broadcastVar[0] and lst[1] < broadcastVar[1]: # 鍗抽冶鏄□唬閾丨倍
        res.append([lst[0],lst[1], lst[2]*broadcastVar[3]])
        if lst[0]-1 == broadcastVar[0]:
            res.append([lst[0]+1,lst[1], lst[2]*broadcastVar[2]])
        elif lst[0]+1 == broadcastVar[1]:
            res.append([lst[0]-1,lst[1], lst[2]*broadcastVar[2]])
        elif lst[1]-1 == broadcastVar[0]:
            res.append([lst[0],lst[1]+1, lst[2]*broadcastVar[2]])
        elif lst[1]+1 == broadcastVar[1]:
            res.append([lst[0],lst[1]-1, lst[2]*broadcastVar[2]])
        else:
            res.append([lst[0]+1,lst[1], lst[2]*broadcastVar[2]])
            res.append([lst[0]-1,lst[1], lst[2]*broadcastVar[2]])
            res.append([lst[0],lst[1]+1, lst[2]*broadcastVar[2]])
            res.append([lst[0],lst[1]-1, lst[2]*broadcastVar[2]])
    else:
        res.append([lst[0],lst[1], lst[2]])
        if (lst[0],lst[1]) not in [(broadcastVar[0], broadcastVar[0]),(broadcastVar[0], broadcastVar[1]),(broadcastVar[1],broadcastVar[0]),(broadcastVar[1],broadcastVar[1])]:
            if lst[0] == broadcastVar[0]:
                res.append([lst[0]+1,lst[1], lst[2]*broadcastVar[2]])
            elif lst[0] == broadcastVar[1]:
                res.append([lst[0]-1,lst[1], lst[2]*broadcastVar[2]])
            elif lst[1] == broadcastVar[0]:
                res.append([lst[0],lst[1]+1, lst[2]*broadcastVar[2]])
            elif lst[1] == broadcastVar[1]:
                res.append([lst[0],lst[1]-1, lst[2]*broadcastVar[2]])
    return res

def merge_by_index(tup: tuple):
    """按row_index 把对应数据放入对应位置
    """
    row_index, data_list = tup
    res = [0]*len(data_list)
    for col_index, value in data_list:
        res[col_index] = value
    return (row_index, np.array(res, dtype = float))

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 = np.arange(100).reshape(10, -1)
    rdd = sc.parallelize(mat)

    rdd.foreach(g)
    print()

    for i in range(5):
        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())
        rdd = sc.parallelize(rdd7)
        rdd.foreach(g)
        print()

    sc.stop()

if __name__ == "__main__":
    main()

```

[我的代码]

```
"""
基于两个 准备的代码

"D:\myCodes\HPCbook\05pyspark\sparkTest1NumpyMattoRDD9.py"
"D:\myCodes\HPCbook\05pyspark\sparkTest1NumpyMattoRDD10.py"


"""


from __future__ import print_function
import numpy as np
import timeit

# =============matplotlib 配置 ==========================
import matplotlib

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

```
    return field

def write_field(field, step):
    # plt.gca().clear()
    plt.cla()
    plt.clf()


    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    # Set Colorbar
    plt.colorbar()
    plt.axis('on')
    plt.savefig('heat_Spark_{0:03d}.png'.format(step))

# ============================================================
import pyspark
sc = pyspark.SparkContext.getOrCreate()

ps = (lenX, lenY, a, dt, dx, dy)

def g(x):
    print(x)

def func(tup: tuple) -> list:
    """主要作用是把每个值打上行列index
    """
    Array, row_index = tup
    return [[row_index, col_index, value] for col_index, value in
enumerate(Array)]

def merge_by_index(tup: tuple):
    """按row_index 把对应数据放入对应位置
    """
    row_index, data_list = tup
    res = [0]*len(data_list)
    for col_index, value in data_list:
        res[col_index] = value
    return (row_index, np.array(res, dtype = float))

internalSelf = 1.0 - 2.0*a*dt/dx - 2.0*a*dt/dy
diff4DirectionsDx = a*dt/dx
diff4DirectionsDy = a*dt/dy

def diffuse(lst: list) -> list:
    global ps
    res = []

    if lst[0] > ps[0] and lst[0] < ps[1] and lst[1] > ps[0] and lst[1] <
ps[1]: # 即都是内部点
        res.append([lst[0],lst[1], lst[2]*internalSelf])
        if lst[0]-1 == ps[0]:
            res.append([lst[0]+1,lst[1], lst[2]*diff4DirectionsDx])
        elif lst[0]+1 == ps[1]:
```

```
            res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDx])
        elif lst[1]-1 == ps[0]:
            res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDy])
        elif lst[1]+1 == ps[1]:
            res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDy])
        else:
            res.append([lst[0]+1,lst[1], lst[2]*diff4DirectionsDx])
            res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDx])
            res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDy])
            res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDy])
    else:
        res.append([lst[0],lst[1], lst[2]])
        if (lst[0],lst[1]) not in [(0, 0),(0, ps[1]-1),(ps[0]-1,0),
(ps[0]-1,ps[1]-1)]:
            if lst[0] == 0:
                res.append([lst[0]+1,lst[1], lst[2]*diff4DirectionsDx])
            elif lst[0] == ps[0]-1:
                res.append([lst[0]-1,lst[1], lst[2]*diff4DirectionsDx])
            elif lst[1] == 0:
                res.append([lst[0],lst[1]+1, lst[2]*diff4DirectionsDy])
            elif lst[1] == ps[1]-1:
                res.append([lst[0],lst[1]-1, lst[2]*diff4DirectionsDy])
    return res

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)
```

```
        # print()

    # print("Iteration finished")
    print("Iteration finished. {} Seconds for Time difference:".format(timeit.default_timer() - starting_time))

    sc.stop()

if __name__ == "__main__":
    main()


```

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 heatEqu2DPySparkImgs.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]

```
"""
2023年5月3日09:50:52

贺思超同学完成了 MPI + PyCUDA 的代码示意

https://github.com/Routhleck/heat_conduction

"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.
```

```
# You can try set it to 10, or 100 to see the difference
# You can also try: colourMap = plt.cm.coolwarm
colorinterpolation = 50
colourMap = plt.cm.jet

# Set meshgrid
X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))

# device
device = 'gpu'

# benchmark
isBenchmark = True

# MPI globals
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# Up/down neighbouring MPI ranks
up = rank - 1
if up < 0:
    up = MPI.PROC_NULL
down = rank + 1
if down > size - 1:
    down = MPI.PROC_NULL

# Define the kernel function
mod = SourceModule("""
    __global__ void evolve_kernel(double* u, double* u_previous,
                                   double a, double dt, double dx2, double dy2,
                                   int nx, int ny) {
        int i = blockIdx.x * blockDim.x + threadIdx.x + 1;
        int j = blockIdx.y * blockDim.y + threadIdx.y + 1;
        if (i >= nx - 1 || j >= ny - 1) return;
        u[i * ny + j] = u_previous[i * ny + j] + a * dt * (
                        (u_previous[(i + 1) * ny + j] - 2 * u_previous[i * ny + j] +
                         u_previous[(i - 1) * ny + j]) / dx2 +
                        (u_previous[i * ny + j + 1] - 2 * u_previous[i * ny + j] +
                         u_previous[i * ny + j - 1]) / dy2);
        u_previous[i * ny + j] = u[i * ny + j];
    }
""")

# Get the kernel function
def get_cuda_function():
    return mod.get_function("evolve_kernel")

# get CUDA kernel
evolve_kernel = get_cuda_function()

# main calculate function
def evolve(u, u_previous, a, dt, dx2, dy2, device):
```

```
    """Explicit time evolution.
       u:            new temperature field
       u_previous:   previous field
       a:            diffusion constant
       dt:           time step
       dx2:          grid spacing squared, i.e. dx^2
       dy2:            -- "" --          , i.e. dy^2"""
    if device == 'cpu':
        u[1:-1, 1:-1] = u_previous[1:-1, 1:-1] + a * dt * (
                (u_previous[2:, 1:-1] - 2 * u_previous[1:-1, 1:-1] +
                 u_previous[:-2, 1:-1]) / dx2 +
                (u_previous[1:-1, 2:] - 2 * u_previous[1:-1, 1:-1] +
                 u_previous[1:-1, :-2]) / dy2)
        u_previous[:] = u[:]
    elif device == 'gpu':
        block_size = (16, 16, 1)
        grid_size = (int(np.ceil(u.shape[0] / block_size[0])),
                     int(np.ceil(u.shape[1] / block_size[1])),
                     1)

        # Call the evolve_kernel with the prepared grid and block sizes
        # import pandas as pd
        # df = pd.DataFrame(u)
        # df.to_csv('u.csv')
        evolve_kernel(driver.InOut(u), driver.InOut(u_previous),
np.float64(a), np.float64(dt),
                                      np.float64(dx2), np.float64(dy2),
np.int32(u.shape[0]),
                                      np.int32(u.shape[1]), block=block_size,
grid=grid_size)
        u_previous[:] = u[:]
    else:
        raise ValueError('device should be cpu or gpu')


# init numpy matrix fields
def init_fields():
    # init
    field = np.empty((lenX, lenY), dtype=np.float64)
    field.fill(Tguess)
    field[(lenY - 1):, :] = Ttop
    field[:1, :] = Tbottom
    field[:, (lenX - 1):] = Tright
    field[:, :1] = Tleft
    # field = np.loadtxt(filename)
    field0 = field.copy()  # Array for field of previous time step
    return field, field0


# save image
def write_field(field, step):
    plt.gca().clear()
    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    if step == 0:
```

```
        plt.colorbar()
    plt.savefig(
 'imgMPIPyCUDA/heat_{thread}_{device}_{shape}_{timesteps}_{step}.png'.format(thread=size, device=device, shape=lenX,
timesteps=timesteps, step=step))


# 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
```

```
    local_field = np.zeros((n + 2, m), dtype)  # need two ghost rows!
    local_field[1:-1, :] = buff  # copy data to non-ghost rows
    local_field0 = np.zeros_like(local_field)  # array for previous time
step

    # Fix outer boundary ghost layers to account for aperiodicity?
    if True:
        if rank == 0:
            local_field[0, :] = local_field[1, :]
        if rank == size - 1:
            local_field[-1, :] = local_field[-2, :]
    local_field0[:] = local_field[:]

    # Plot/save initial field
    if rank == 0:
        write_field(field, 0)

    # Iterate
    t0 = time.time()
    iterate(field, local_field, local_field0, timesteps, image_interval)
    t1 = time.time()

    # Plot/save final field
    comm.Gather(local_field[1:-1, :], field, root=0)
    if rank == 0:
        write_field(field, timesteps)
        if (isBenchmark):
            import pandas as pd
            import os
            # 若不存在csv文件，则创建，若存在，则追加
            if not os.path.exists('data.csv'):
                df = pd.DataFrame(columns=['MPI_thread', 'device', 'len',
'timesteps', 'time'])
                df.to_csv('data.csv', index=False)
            df = pd.read_csv('data.csv')
            df = df.append(
                {'MPI_thread': size, 'device': device, 'shape': lenX,
'timesteps': timesteps, 'time': t1 - t0},
                ignore_index=True)
            df.to_csv('data.csv', index=False)
        print("Running time: {0}".format(t1 - t0))


if __name__ == '__main__':
    opts, _ = getopt.getopt(sys.argv[1:], 'd:t:i:l:', ['device=',
'timesteps=', 'image_interval=', 'len='])
    for opt, arg in opts:
        if opt in ('-d', '--device'):
            device = arg
        elif opt in ('-t', '--timesteps'):
            timesteps = int(arg)
        elif opt in ('-i', '--image_interval'):
            image_interval = int(arg)
        elif opt in ('-l', '--len'):
            lenX = int(arg)
            lenY = int(arg)
```

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

MPI+Numba.CUDA

[我的代码]

```
"""
2023年5月3日09:50:52

贺思超同学完成了 MPI + PyCUDA 的代码示意

https://github.com/Routhleck/heat_conduction

"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"

"""

from __future__ import print_function

import getopt
import sys

import numpy as np
import time
from mpi4py import MPI
from numba import cuda # 从numba调用cuda

import matplotlib

matplotlib.use('Agg')
import matplotlib.pyplot as plt
from tqdm import tqdm



# 基本参数设置

# Basic parameters
a = 0.5  # Diffusion constant
timesteps = 10000  # Number of time-steps to evolve system
image_interval = 1000  # Write frequency for png files

# Set Dimension and delta
lenX = lenY = 400  # we set it rectangular
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
```

```
    field[:1, :] = Tbottom
    field[:, (lenX - 1):] = Tright
    field[:, :1] = Tleft
    # field = np.loadtxt(filename)

    print("size is ",field.size)
    print(field,"\n")

    field0 = field.copy()  # Array for field of previous time step
    return field, field0


# save image
def write_field(field, step):
    plt.gca().clear()
    # Configure the contour
    plt.title("Contour of Temperature")
    plt.contourf(X, Y, field, colorinterpolation, cmap=colourMap)
    if step == 0:
        plt.colorbar()
    plt.savefig(

 'imgMPINumbaCUDA/heat_{thread}_{device}_{shape}_{timesteps}_{step}.png'.fo
rmat(thread=size, device=device, shape=lenX,

timesteps=timesteps, step=step))

# MPI globals
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

# Up/down neighbouring MPI ranks
up = rank - 1
if up < 0:
    up = MPI.PROC_NULL
down = rank + 1
if down > size - 1:
    down = MPI.PROC_NULL

# 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)

@cuda.jit
def evolve_kernel(u, u_previous, a, dt, dx2, dy2, xlen, ylen):
    """Explicit time evolution.
       u:            new temperature field
       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] +
159             (u_previous[i][j+1] - 2.0 * u_previous[i][j] + u_previous[i][j-
    1])/dy2[0])
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,
    dt, dx2, dy2, device)
209         if m % image_interval == 0:
210             comm.Gather(local_field[1:-1, :], field, root=0)
211             comm.Barrier()
212             if rank == 0:
213                 write_field(field, m)
214
215 def main():
216     # Read and scatter the initial temperature field
217     if rank == 0:
218         field, field0 = init_fields()
219         shape = field.shape
220         dtype = field.dtype
221         comm.bcast(shape, 0)  # broadcast dimensions
222         comm.bcast(dtype, 0)  # broadcast data type
223     else:
224         field = None
225         shape = comm.bcast(None, 0)
226         dtype = comm.bcast(None, 0)
227     if shape[0] % size:
228         raise ValueError('Number of rows in the temperature field (' \
229                          + str(shape[0]) + ') needs to be divisible by the
    number ' \
230                          + 'of MPI tasks (' + str(size) + ').')
231
232     n = int(shape[0] / size)  # number of rows for each MPI task
233     m = shape[1]  # number of columns in the field
234     buff = np.zeros((n, m), dtype)
235     comm.Scatter(field, buff, 0)  # scatter the data
236     local_field = np.zeros((n + 2, m), dtype)  # need two ghost rows!
237     local_field[1:-1, :] = buff  # copy data to non-ghost rows
238     local_field0 = np.zeros_like(local_field)  # array for previous time
    step
239
240     # Fix outer boundary ghost layers to account for aperiodicity?
241     if True:
242         if rank == 0:
243             local_field[0, :] = local_field[1, :]
244         if rank == size - 1:
245             local_field[-1, :] = local_field[-2, :]
246     local_field0[:] = local_field[:]
247
248     # Plot/save initial field
249     if rank == 0:
250         write_field(field, 0)
251
252     # Iterate
253     t0 = time.time()
254     iterate(field, local_field, local_field0, timesteps, image_interval)
```

```
    t1 = time.time()

    # Plot/save final field
    comm.Gather(local_field[1:-1, :], field, root=0)
    if rank == 0:
        # write_field(field, timesteps)
        if (isBenchmark):
            import pandas as pd
            import os
            # 若不存在csv文件，则创建，若存在，则追加
            if not os.path.exists('data.csv'):
                df = pd.DataFrame(columns=['MPI_thread', 'device', 'len',
'timesteps', 'time'])
                df.to_csv('data.csv', index=False)
            df = pd.read_csv('data.csv')
            df = df.append(
                {'MPI_thread': size, 'device': device, 'shape': lenX,
'timesteps': timesteps, 'time': t1 - t0},
                ignore_index=True)
            df.to_csv('data.csv', index=False)
        print("Running time: {0}".format(t1 - t0))


if __name__ == '__main__':
    opts, _ = getopt.getopt(sys.argv[1:], 'd:t:i:l:', ['device=',
'timesteps=', 'image_interval=', 'len='])
    for opt, arg in opts:
        if opt in ('-d', '--device'):
            device = arg
        elif opt in ('-t', '--timesteps'):
            timesteps = int(arg)
        elif opt in ('-i', '--image_interval'):
            image_interval = int(arg)
        elif opt in ('-l', '--len'):
            lenX = int(arg)
            lenY = int(arg)
            X, Y = np.meshgrid(np.arange(0, lenX), np.arange(0, lenY))
    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"

概述

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

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

其中`hello.pyx`的代码为

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

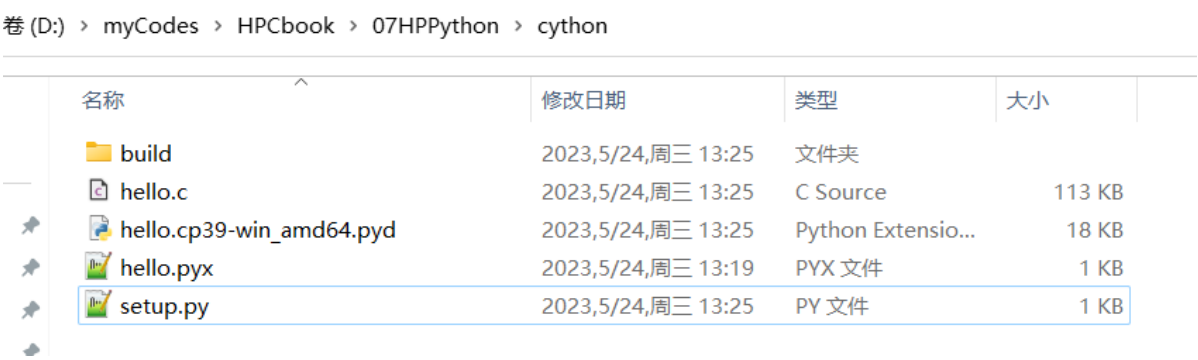
编辑保存之后直接运行`python setup.py build_ext --inplace`即可进行编译。

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

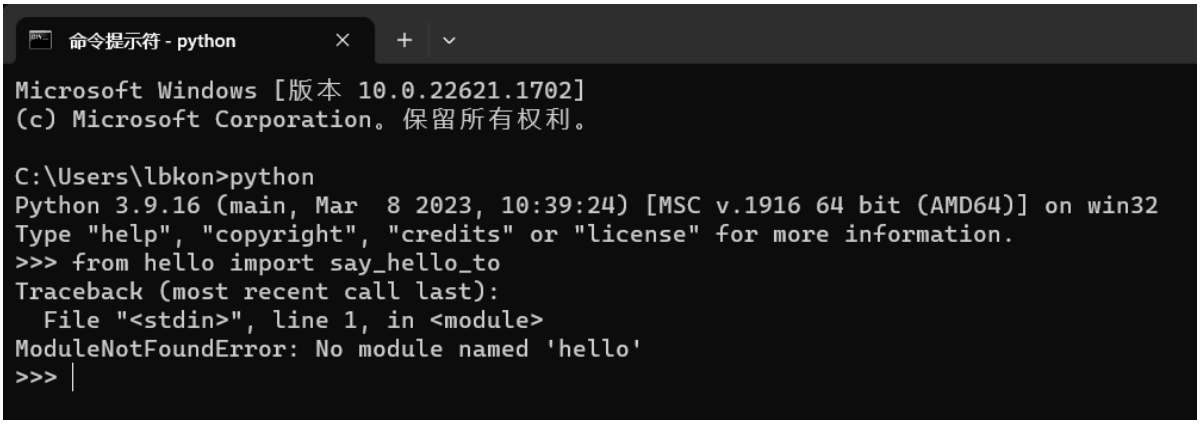
- `error: command 'cl.exe' failed: No such file or directory`:
    - 这说明在环境中没有找到C编译器。由于我电脑中安装了Visual Studio，我的解决方法是根据平台使用VS的**x86 Native Command Prompt**或**x64 Native Command Prompt**来运行编译命令。
    - 此外还可以选择通过`python setup.py build_ext --inplace --compiler=mingw32`使用Anaconda内置的MinGW32编译器，不过这种情况下还可能会遇到内置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\Rel
ease\hello.cp39-win_amd64.exp
正在生成代码
已完成代码的生成

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

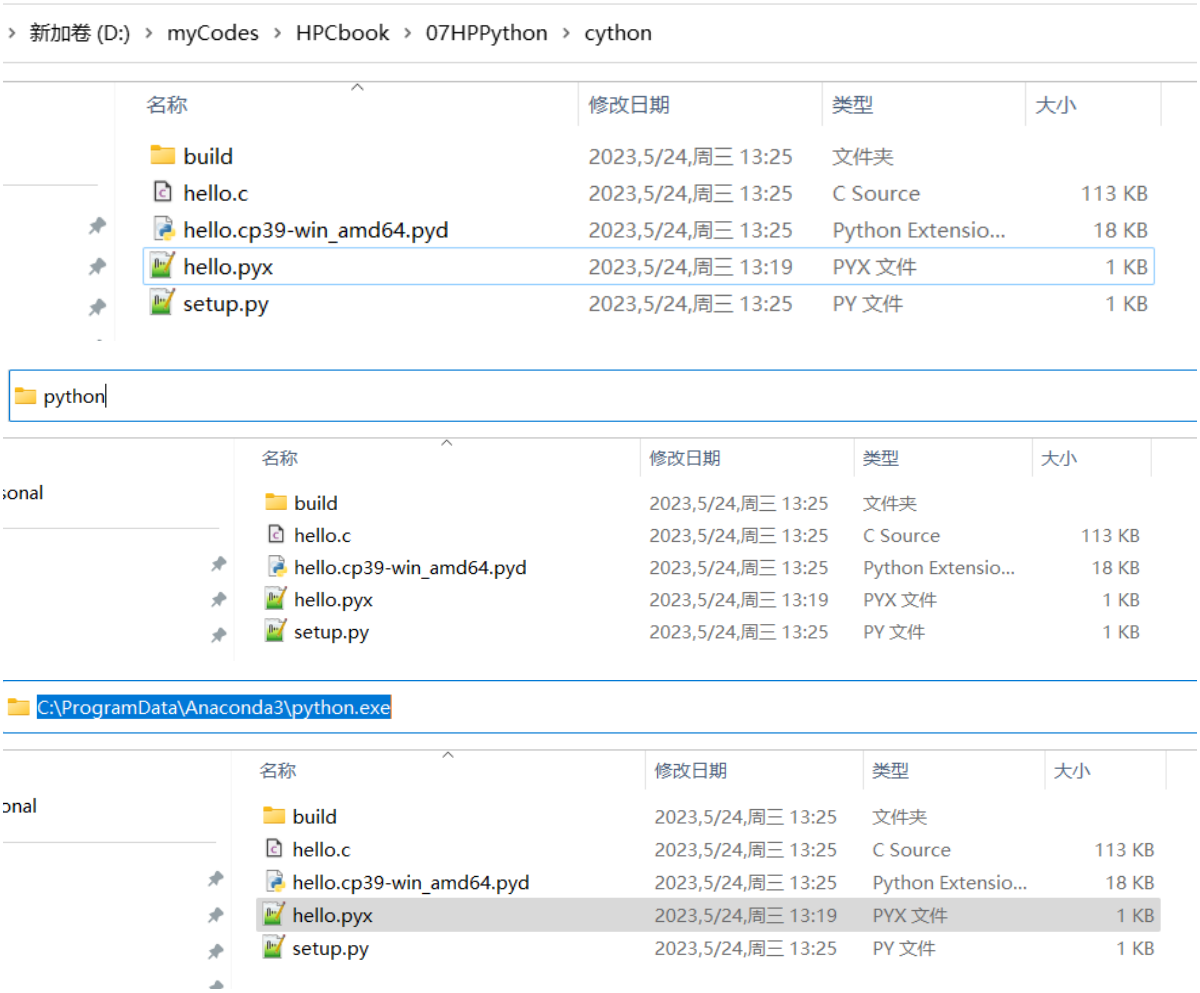


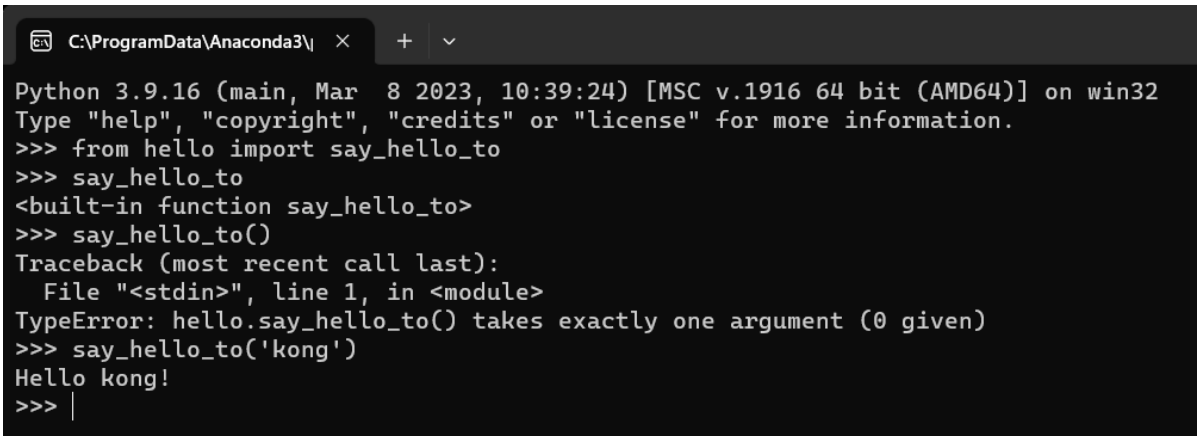
编译之后直接通过 `import hello` 或 `from hello import say_hello_to` 即可调用这个编译好的库。

- 还是要在所在目录 运行 Python 才行!



- 在Explorer 在的所在目录里 `python` 即可





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：

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Build import cythonize
extensions = [Extension("Module Name", ["file1.pyx", "file2.cpp", ...],

                  include_dirs = [...],
                  libraries = [...],
                  library_dirs = [...]),
              Extension("Module 2", ...) ]
setup(    name = "XXX",    ext_modules = cythonize(extensions), )
```

最后的`setup`部分还有另一种写法：

```
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

例子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`

```cpp
#include <pybind11/pybind11.h>
#include <pybind11/stl.h>
namespace py = pybind11; // 引入 pybind11/stl.h 后 std::vector会自动与python中list类型绑定
int sum(const std::vector<int>& buffer) {
    int ret = 0;
    for(auto item: buffer)
        ret += item;
    return ret;
}
// pybind11_sum 这里约定要与文件名相同
PYBIND11_MODULE(pybind11_sum, m) {
    m.doc() = "pybind11 sum plugin"; // optional module docstring
    m.def("sum", &sum, "A function which calulate sum in buffer", py::arg("buffer"));
}
```

然后运行shell命令

```
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`

```
import pybind11_sum print(pybind11_sum)
print(dir(pybind11_sum))
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

```
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          13 KB  defaults
    clang-14.0.6               |       haa95532_1        88.2 MB  defaults
    clang-14-14.0.6            | default_h120b48b_1        30.5 MB  defaults
    clangxx-14.0.6             | default_h120b48b_1        29.5 MB  defaults
    gast-0.5.3                 |     pyhd3eb1b0_0          21 KB  defaults
    libclang-cpp-14.0.6        | default_h120b48b_1         137 KB  defaults
    pythran-0.12.1             |   py39hd4e2768_0         2.0 MB  defaults
    xsimd-10.0.0               |       h59b6b97_0         114 KB  defaults
    ------------------------------------------------------------
                                           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`

```
"""
(C) rgc
All rights reserved
create time '2021/1/26 10:50'
Usage:
"""

```

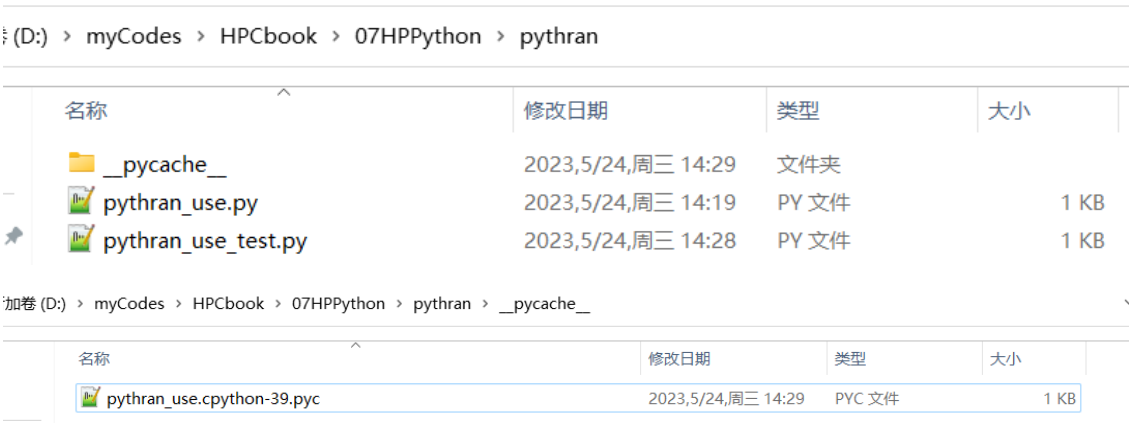
```python
def a(b):
    print('df', b)

def dprod(l0, l1):
    a(2)
    return sum(x*y for x, y in zip(l0, l1))

# pythran export drop1(int[], int list,str:float dict)
def drop1(l0, l1, _dict):
    a(2)
    l1.append(l0[-1])
    print(type(l0))
    for k, v in _dict.items():
        print(k, v)
    return l1
```

命令行中运行如下命令

```
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 Pyth
on 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`



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

```python
"""
(C) rgc
All rights reserved
create time '2021/1/26 11:04'
Usage:
"""

import numpy as np

```

```python
from pythran_use import drop1

if __name__ == '__main__':
    l0 = np.array([1, 2, 3])
    print(type(l0))
    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 Pyth
on 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

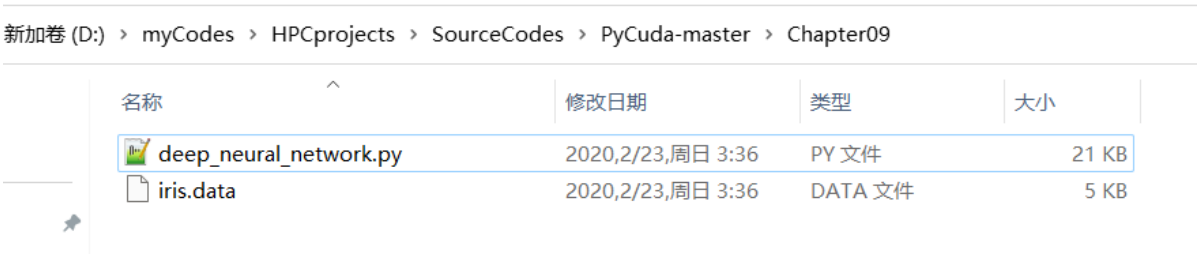
- 每次代码改变时 需要从新编译一次
- Pythran的预编译 和 Numba的运行时编译正好相反;
    - Numba第一次运行时进行编译,所以会较慢;
    - **而Pythran将编译放在了 代码运行前**; - Windows 是似乎不是这样的!
- Numba 在numpy的大数据量的for循环时 比较特长;
- NumExpr 在涉及到 一堆的数学计算时比较特长,但是支持的numpy函数较少;
- Pythran 则适用于 大众情况,且对numpy的函数支持比NumExpr多一些,更适用于 list,dict 等的数据处理 及 numpy操作;
- 具体 使用 Pythran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



[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

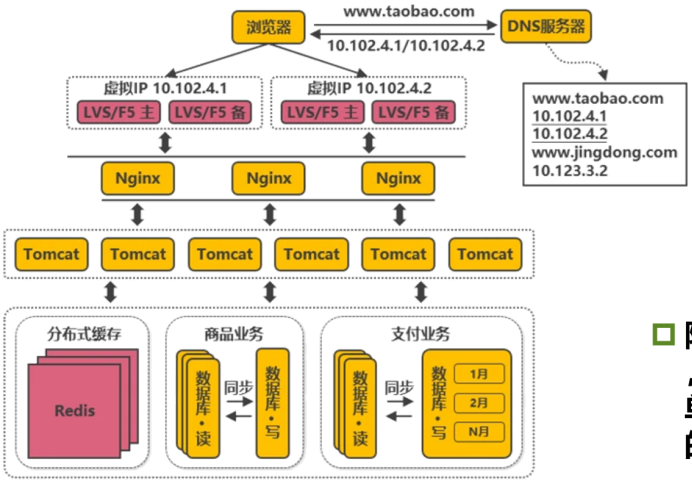
关键点

- **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`

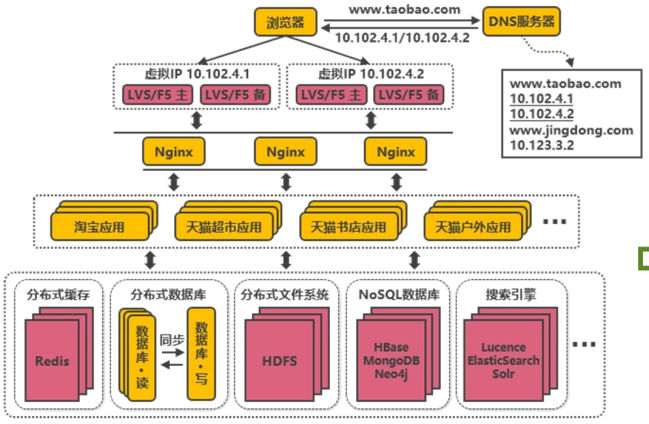
9 第八次演进：通过DNS轮询实现机房间的负载均衡



- 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。此方式能实现机房间的负载均衡，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**

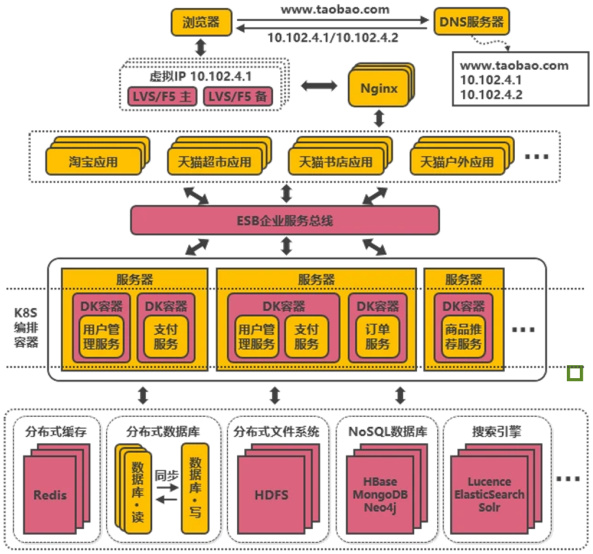
11 第十次演进：大应用拆分为小应用



- 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**

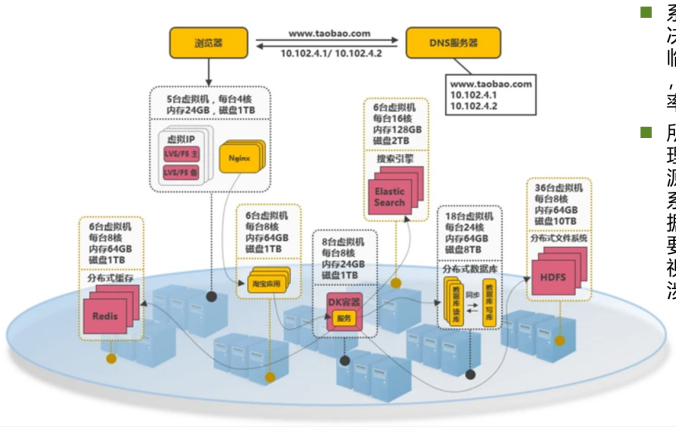
14 第十三次演进：引入容器化技术实现运行环境隔离与动态服务管理



- 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起来，使服务的部署和运维变得简单。
- 在大促的之前，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）。

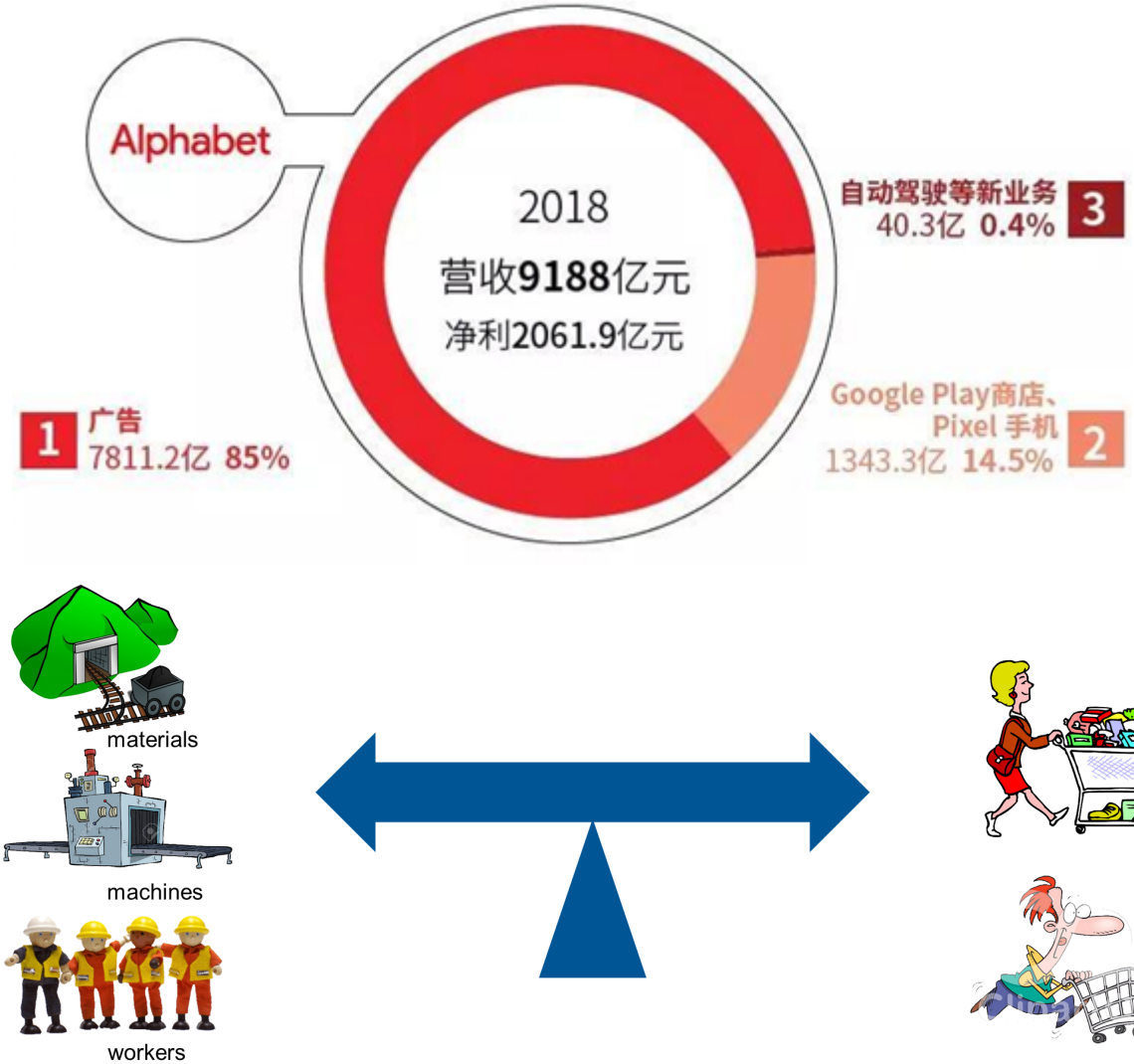
- **使用容器化技术后服务动态扩缩容问题得以解决，但是机器还是需要公司自身来管理，在非大促的时候，还是需要闲置着大量的机器资源来应对大促，机器自身成本和运维成本都极高，资源利用率低**

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。

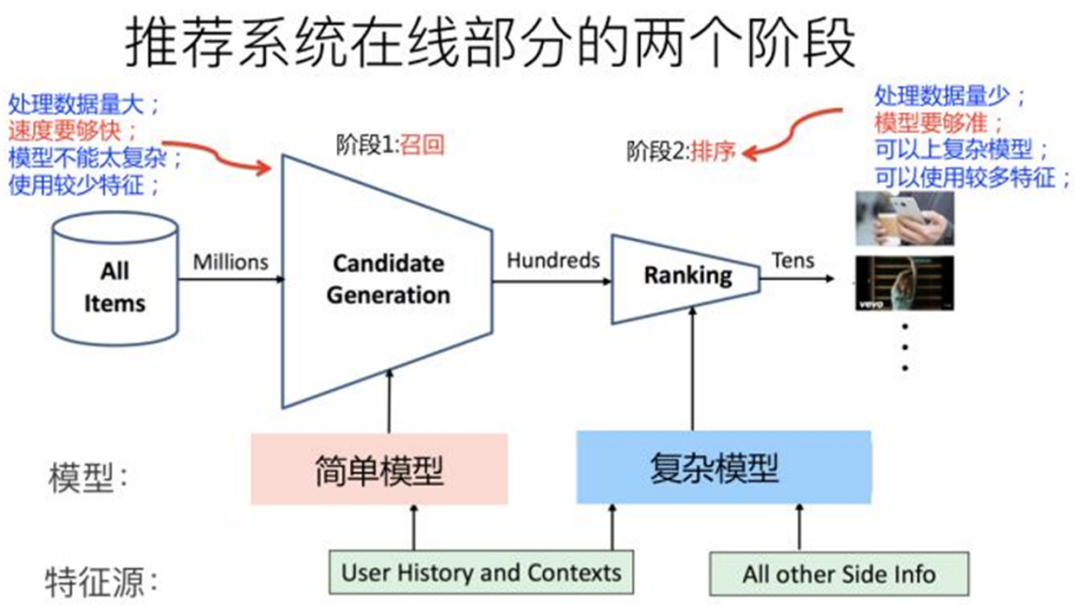
计算广告



Platform 挣广告的钱?

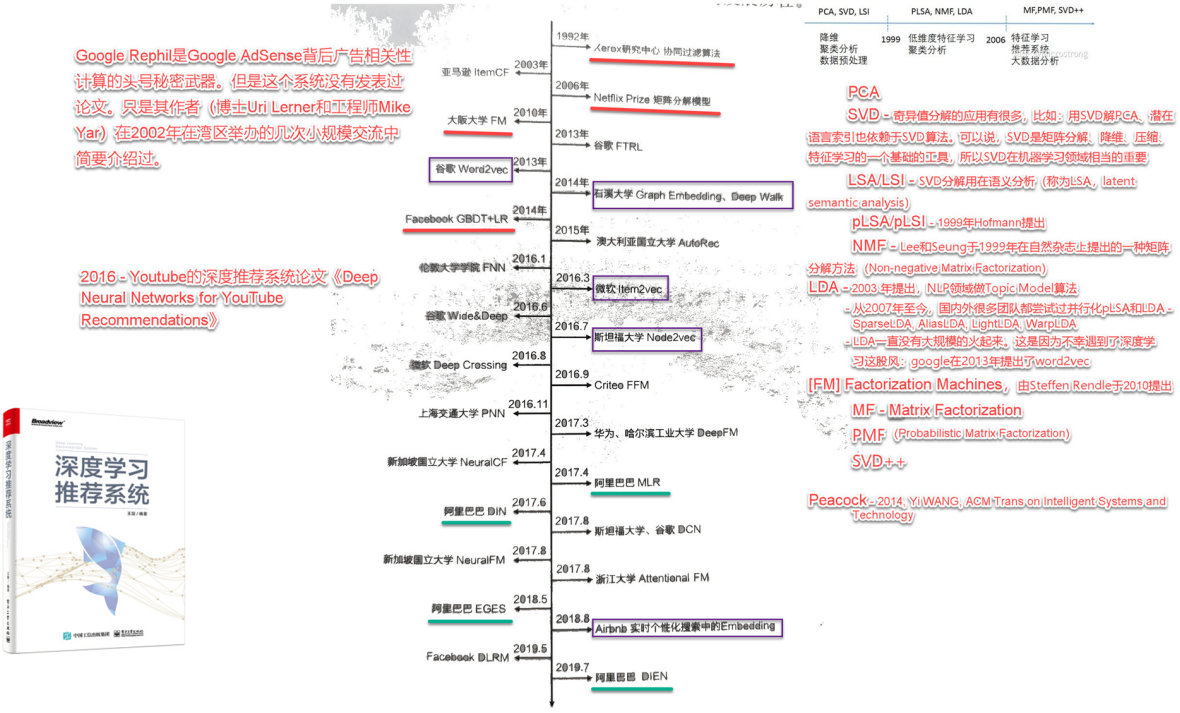
- 巨量的黏着客户 - 是产品/服务提供方的潜在客户
    - 客户买提供方的产品/服务，提供方才能有利润
    - 自然，买的客户越多越好
- 平台因为有巨量的客户，可以做 `掮客` 的角色
    - 因为可以帮助促成交易，自然对提供方有价值
    - 似乎，从提供方收取一定的费用，也就 `理所当然` 了
- 所以，简单的理解
    - 假如 一次交易 提供方 可以有(净)利润10块钱，似乎，给平台1块钱，也是可以接受的 - 也算是抽成吧
- 付费方式?
    - 平台 和 提供方 之间相互选择
    - CPM, ...
-

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



微博 weibo.com

- 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 兴趣的大小 - 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

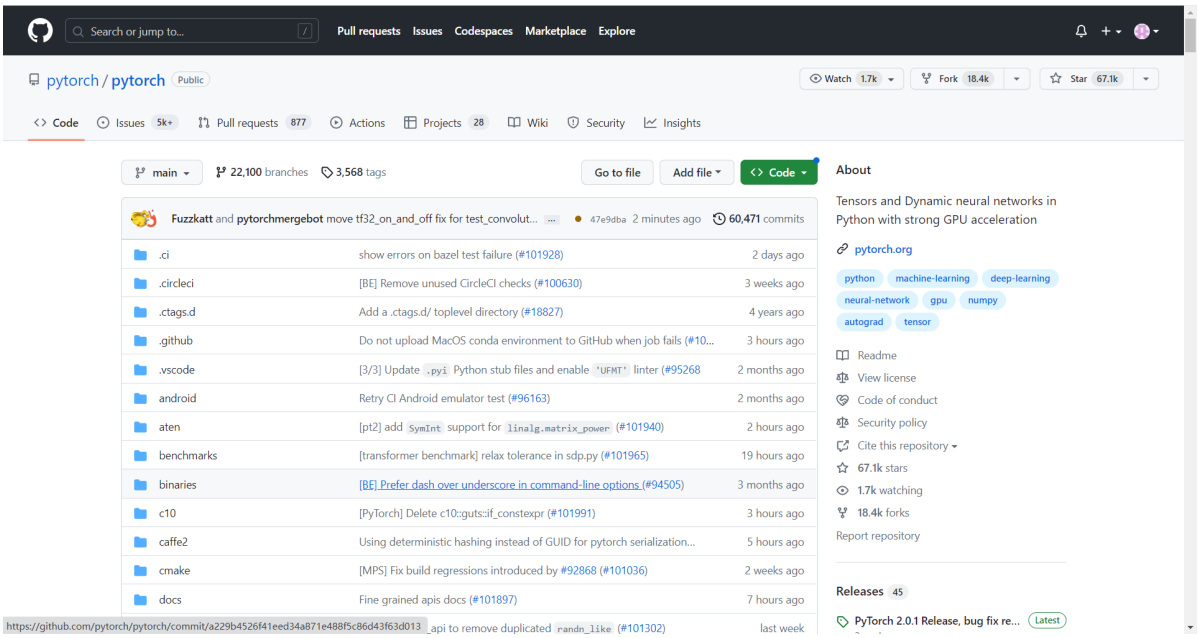
[代码]

例子

分布式机器学习

PyTorch 的设计与实现

https://github.com/pytorch/pytorch



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`.

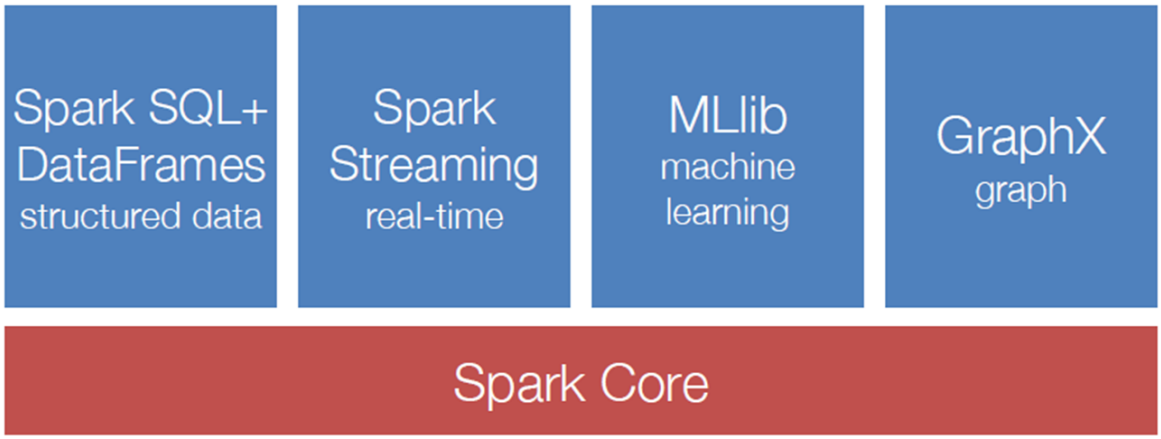
If you are building for NVIDIA's Jetson platforms (Jetson Nano, TX1, TX2, AGX Xavier), Instructions to install PyTorch for Jetson Nano are available here

If you want to compile with ROCm support, install

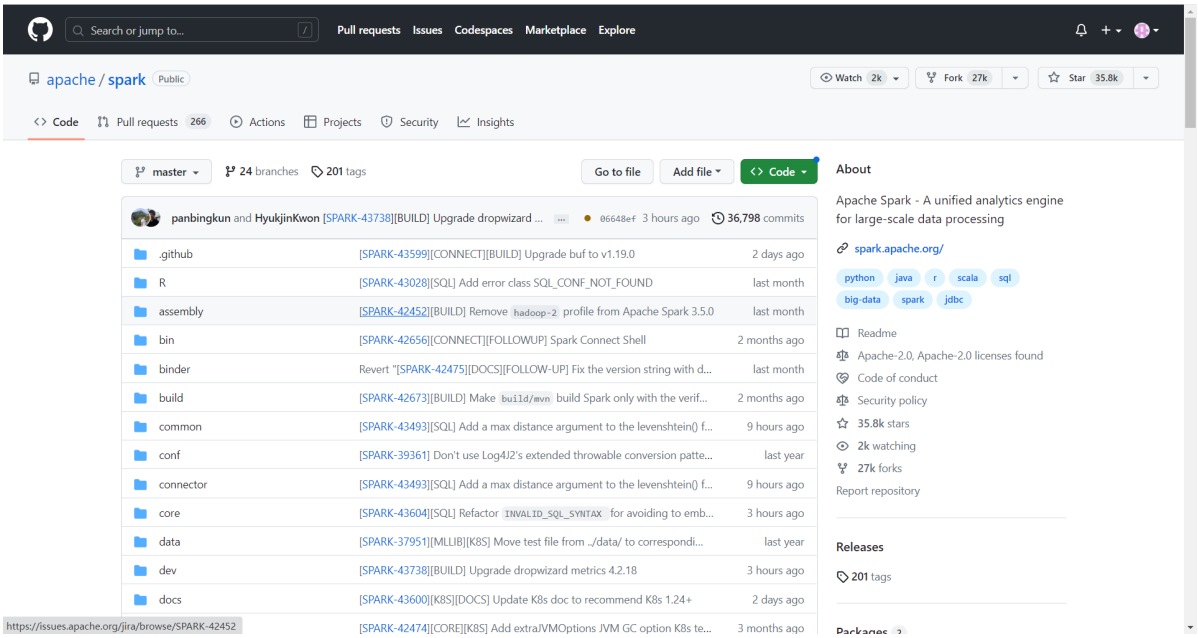
- AMD ROCm 4.0 and above installation
- ROCm is currently supported only for Linux systems.

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`.

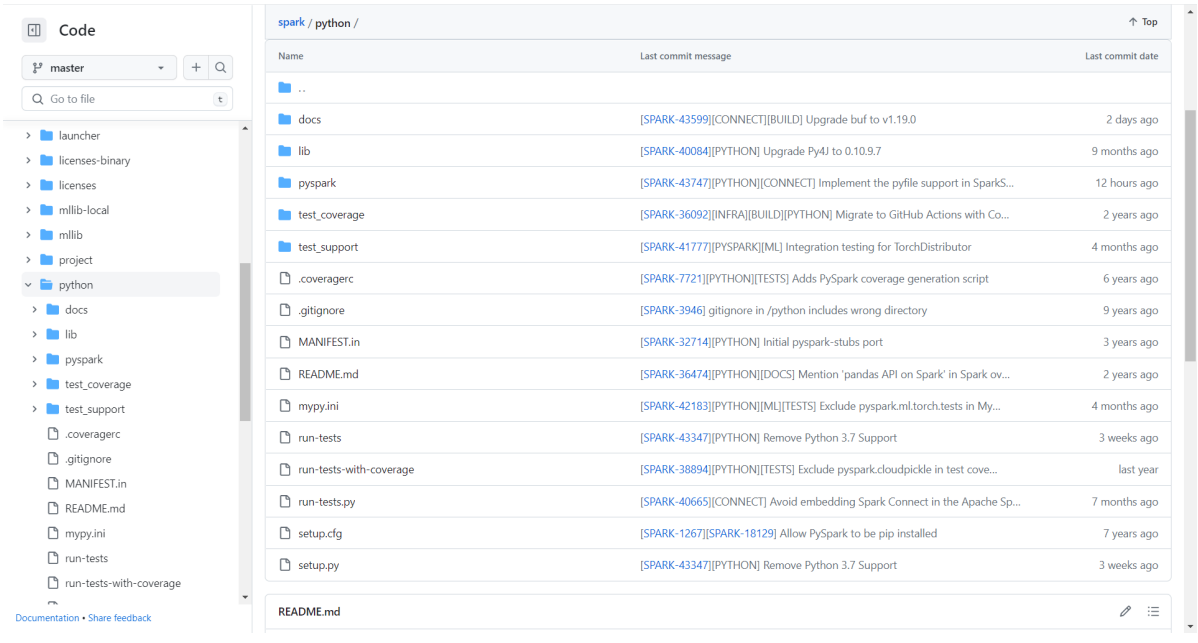
Spark MLib



https://github.com/apache/spark

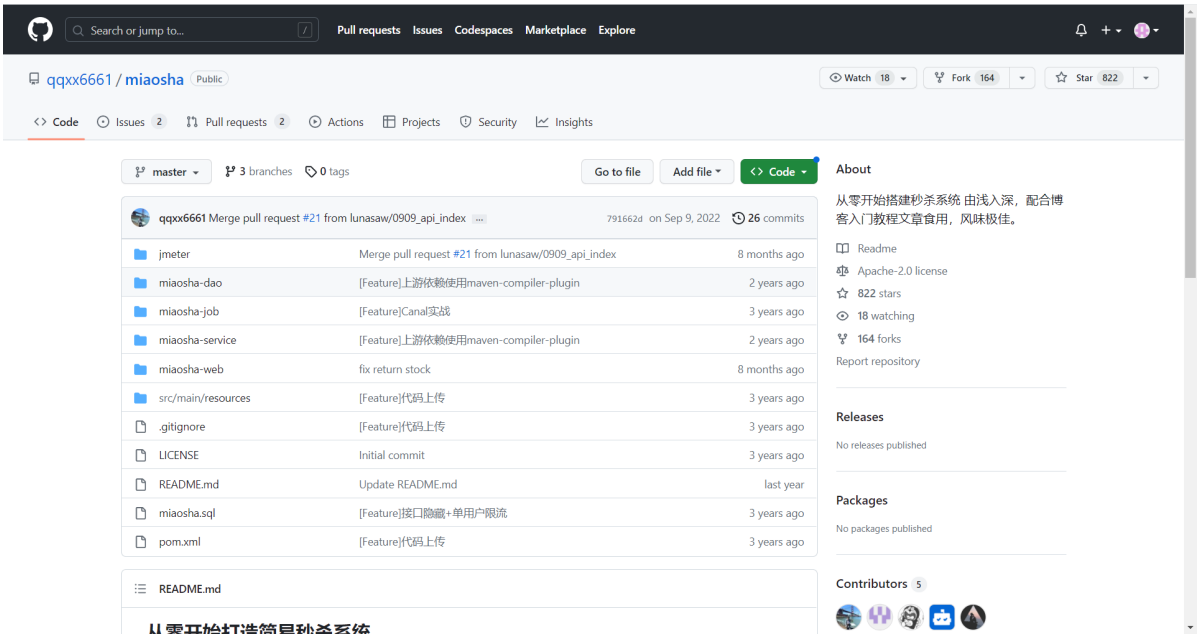


https://github.com/apache/spark/tree/master/python

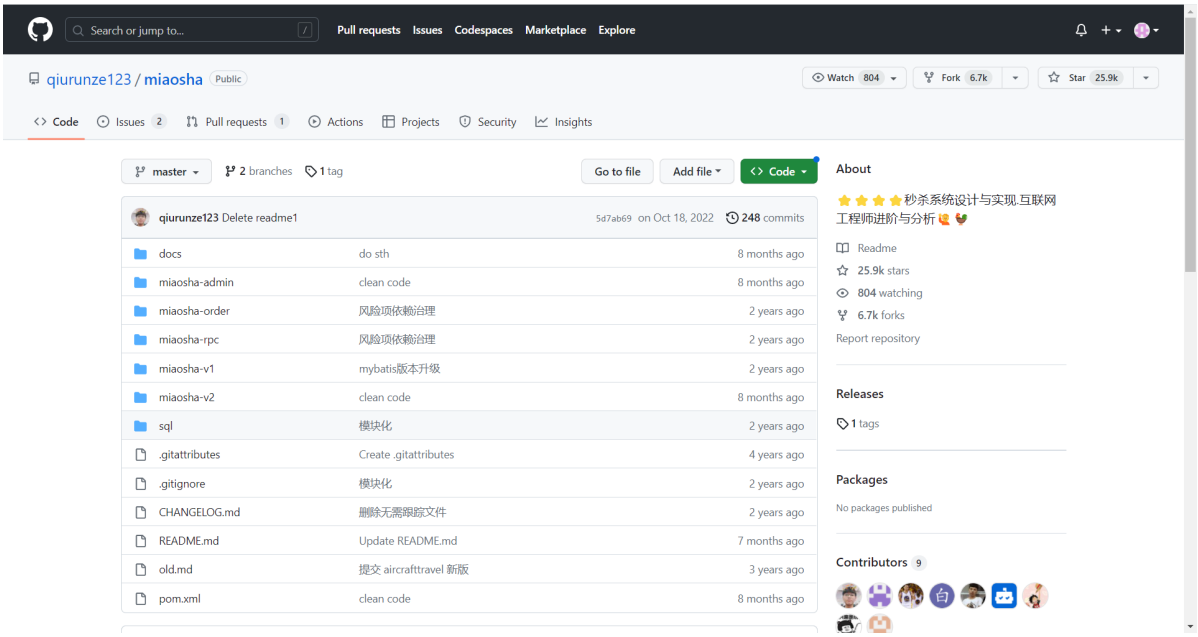


秒杀

https://github.com/qqxx6661/miaosha



https://github.com/qiurunze123/miaosha

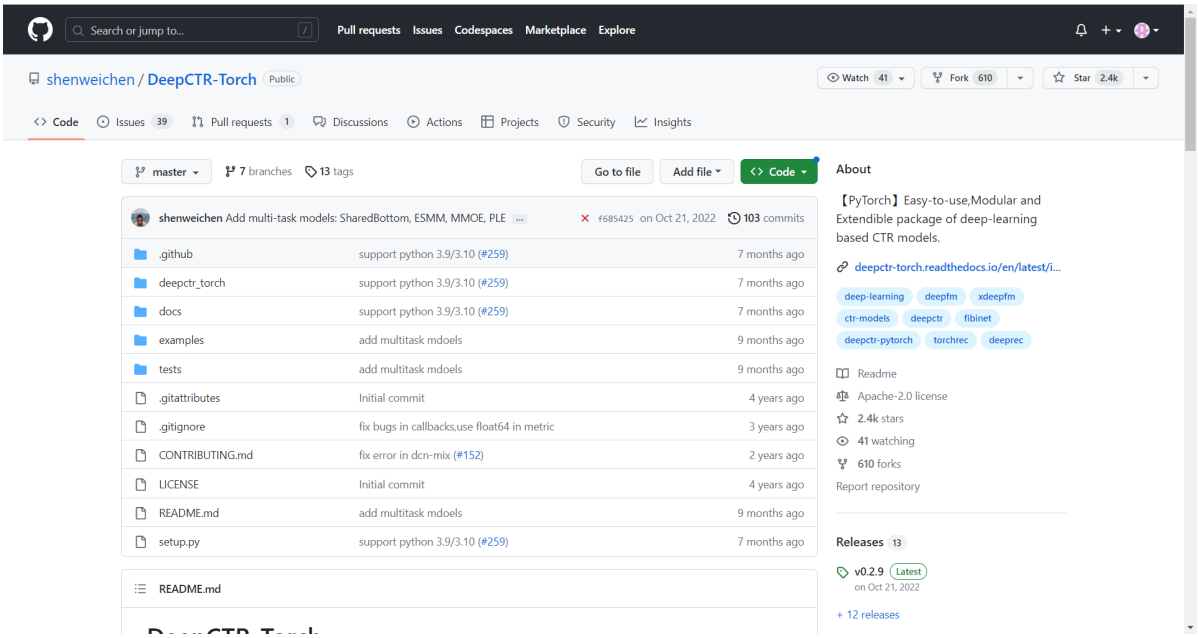


计算广告

CTR

DeepCTR-Torch

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



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

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

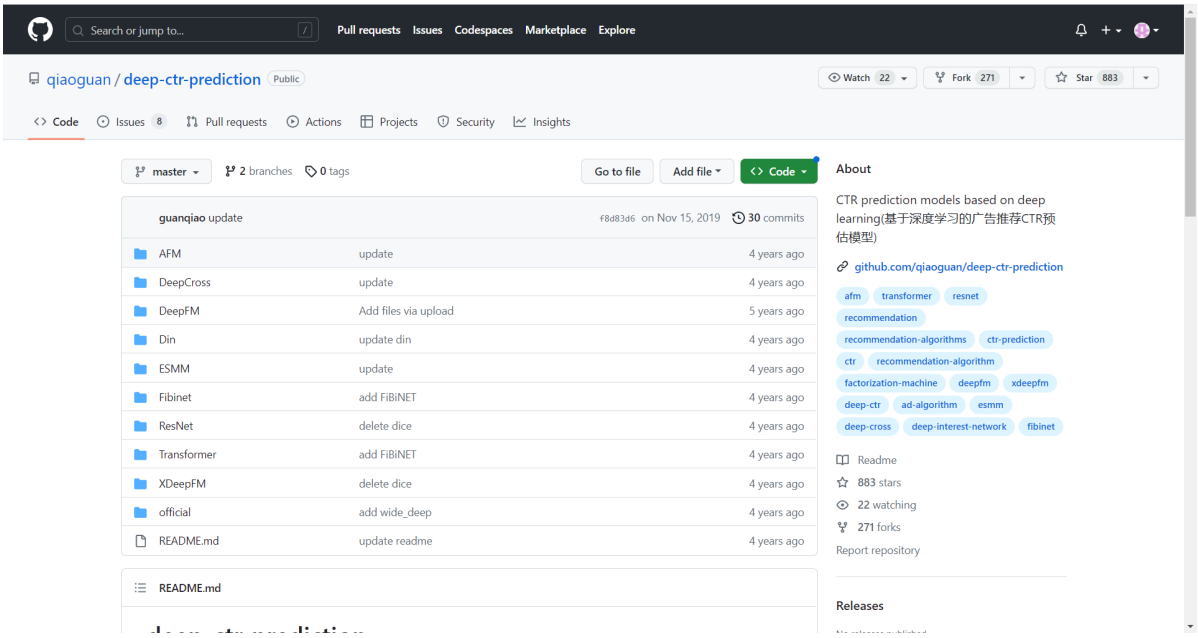
Model		Paper
DCN V2	2020	[arxiv 2020][DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems](https://arxiv.org/abs/2008.13535)
DIFM	2020	[IJCAI 2020][A Dual Input-aware Factorization Machine for CTR Prediction](https://www.ijcai.org/Proceedings/2020/0434.pdf)
AFN	2020	[AAAI 2020][Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions](https://arxiv.org/pdf/1909.03276)
PLE	2020	[RecSys 2020][Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations](https://dl.acm.org/doi/10.1145/3383313.3412236)

DisscussionGroup & Related Projects

- Github Discussions
- Related Projects
-
    - AlgoNotes
    - DeepCTR
    - DeepMatch
    - GraphEmbedding

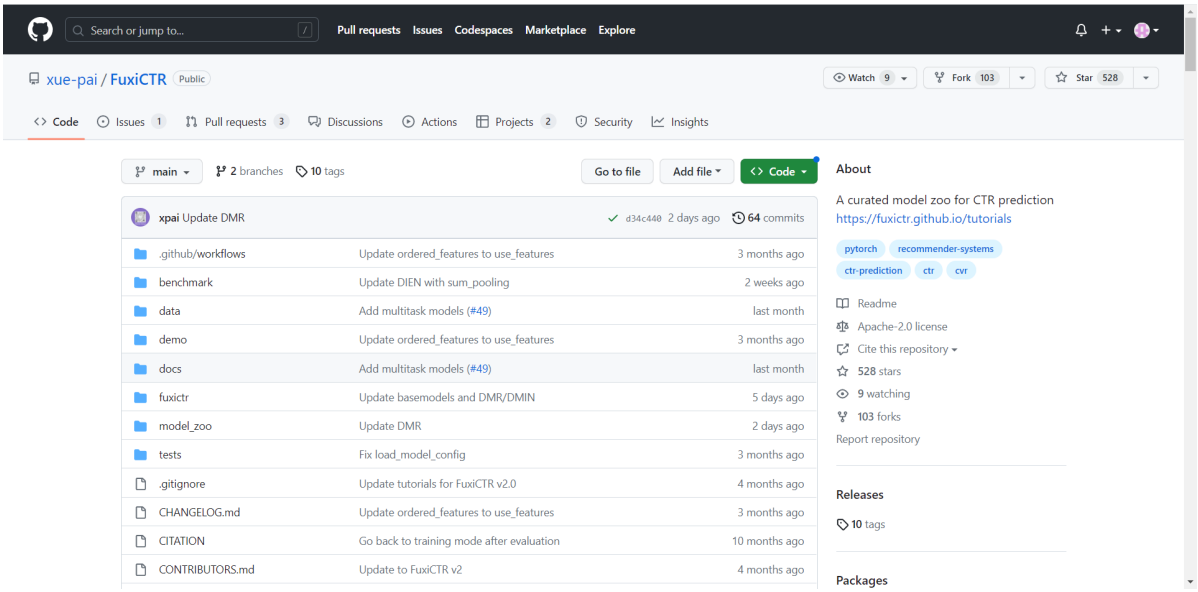
deep-ctr-prediction [TF]

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



FuxiCTR

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



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
			👇 **Feature Interaction Models**		
1	WWW'07	LR	Predicting Clicks: Estimating the Click-Through Rate for New Ads 🚩 **Microsoft**	↗	`torch`
2	ICDM'10	FM	Factorization Machines	↗	`torch`
3	CIKM'13	DSSM	Learning Deep Structured Semantic Models for Web Search using Clickthrough Data 🚩 **Microsoft**	↗	`torch`
4	CIKM'15	CCPM	A Convolutional Click Prediction Model	↗	`torch`

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

No	Publication	Model	Paper	Benchmark	Version
17	CIKM'19	FiGNN	FiGNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction	↗	`torch`
18	CIKM'19	AutoInt/AutoInt+	AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks	↗	`torch`
19	RecSys'19	FiBiNET	FiBiNET: Combining Feature Importance and Bilinear feature Interaction for Click-Through Rate Prediction 🚩 **Sina Weibo**	↗	`torch`
20	WWW'19	FGCNN	Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction 🚩 **Huawei**	↗	`torch`
21	AAAI'19	HFM/HFM+	Holographic Factorization Machines for Recommendation	↗	`torch`
22	Arxiv'19	DLRM	Deep Learning Recommendation Model for Personalization and Recommendation Systems 🚩 **Facebook**	↗	`torch`
23	NeuralNetworks'20	ONN	Operation-aware Neural Networks for User Response Prediction	↗	`torch`
24	AAAI'20	AFN/AFN+	Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions	↗	`torch`
25	AAAI'20	LorentzFM	Learning Feature Interactions with Lorentzian Factorization 🚩 **eBay**	↗	`torch`
26	WSDM'20	InterHAt	Interpretable Click-through Rate Prediction through Hierarchical Attention 🚩 **NEC Labs, Google**	↗	`torch`
27	DLP-KDD'20	FLEN	FLEN: Leveraging Field for Scalable CTR Prediction 🚩 **Tencent**	↗	`torch`

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

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

- 👉 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.

   ```
   cd demo
   python example1_build_dataset_to_h5.py
   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.

   ```
   cd model_zoo/DCN/DCN_torch
   python run_expid.py --expid DCN_test --gpu 0

   # Change `MODEL` according to the target model name
   cd model_zoo/MODEL_PATH
   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.

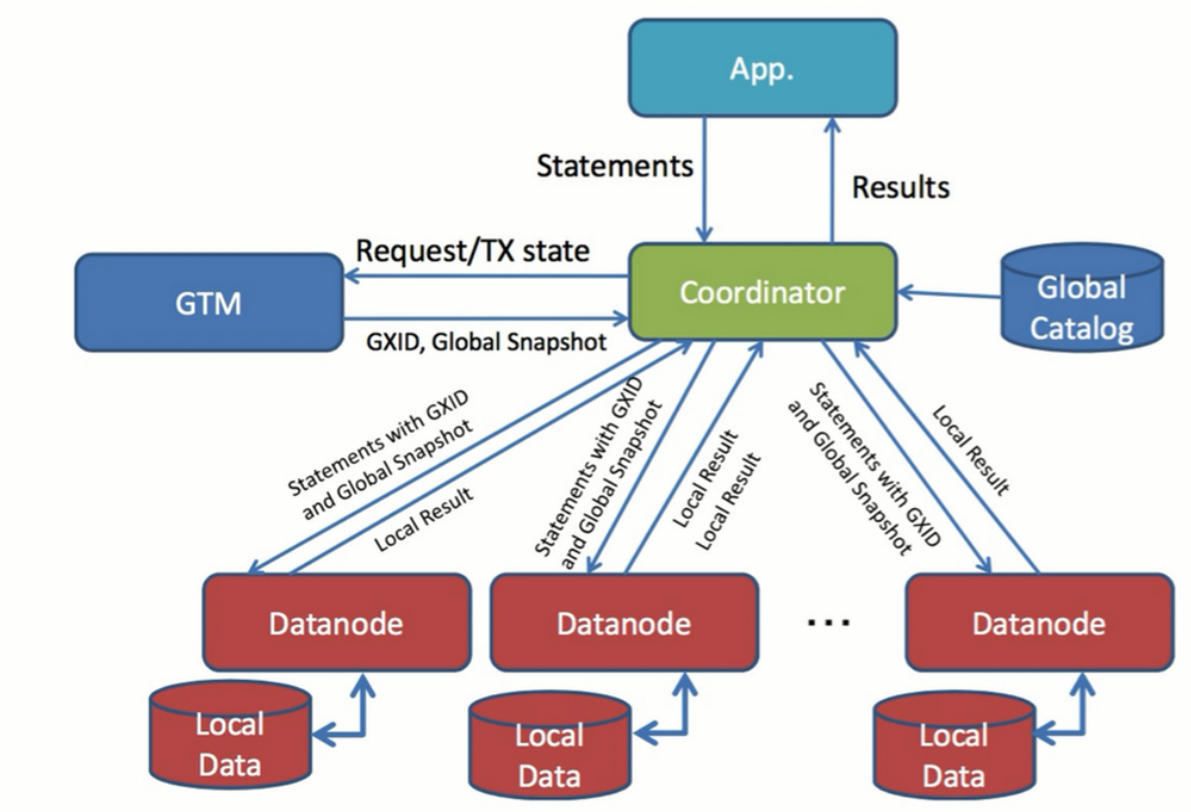
Citation

🔔 *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]

> Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, Rui Zhang. BARS: Towards Open Benchmarking for Recommender Systems. *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2022. [Bibtex]

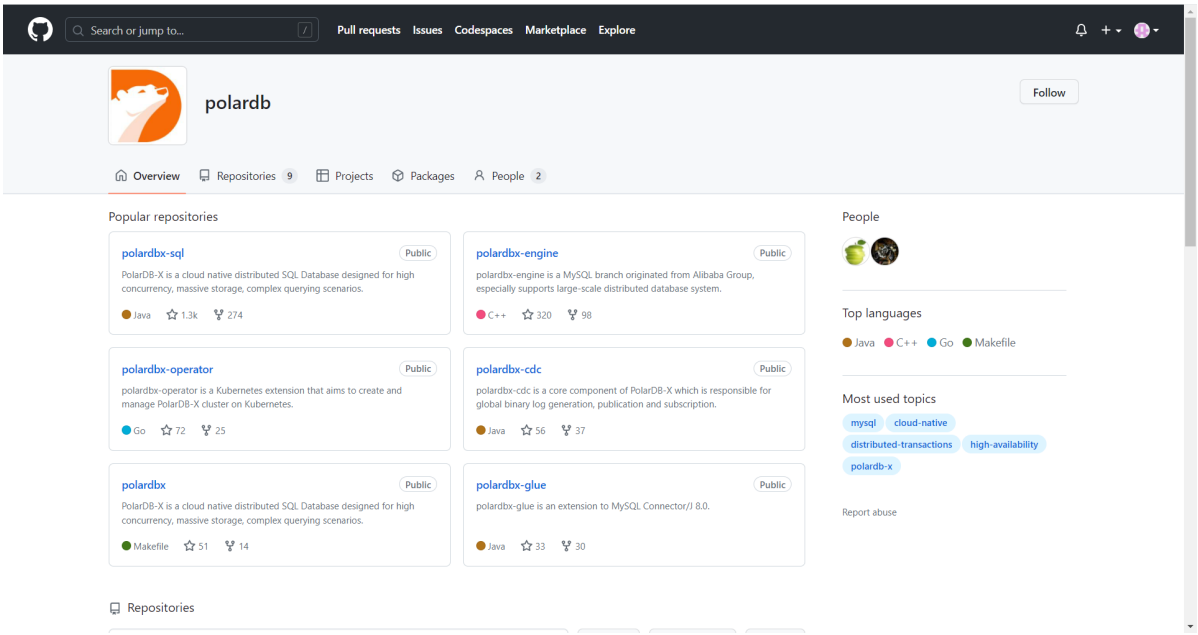
数据库



分布式数据库？云原生数据库？

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

https://github.com/polardb



https://github.com/polardb/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/polardb/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为继。
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。

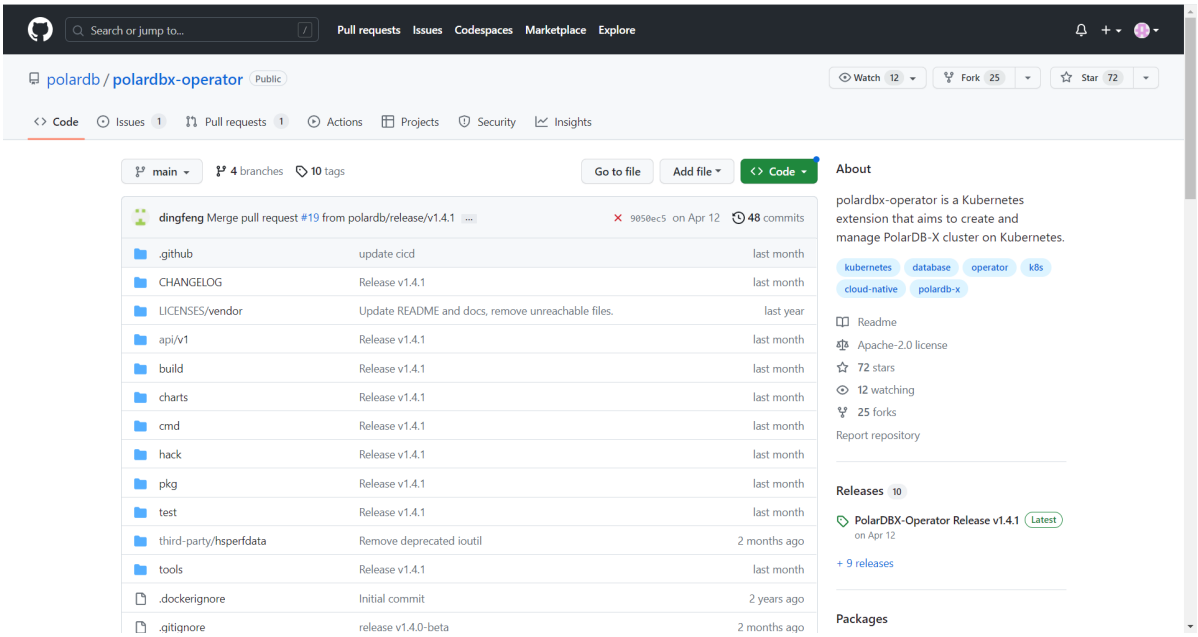
我们探索的方向包括新型存储器件/新型定制计算硬件/AI for DB等领域。团队已发表的论文如下：

- 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

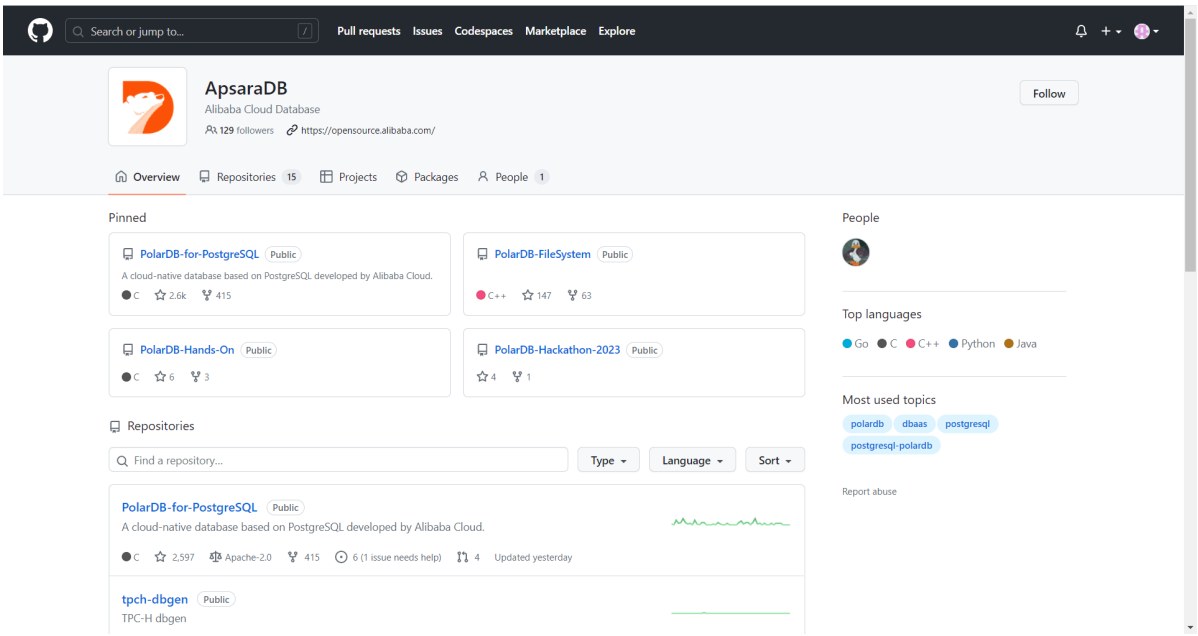
PolarDB-X Operator is a **Kubernetes extension** that aims to create and manage PolarDB-X cluster on Kubernetes. It follows the operator pattern and automates the management tasks.

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

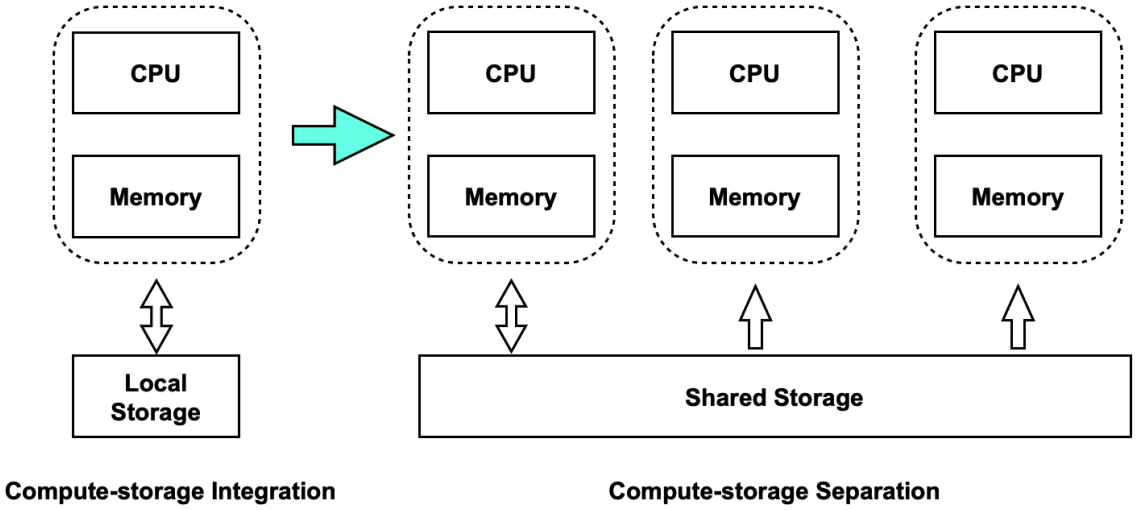


[ApsaraDB] - PolarDB for PostgreSQL (hereafter simplified as PolarDB)

https://github.com/ApsaraDB



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.

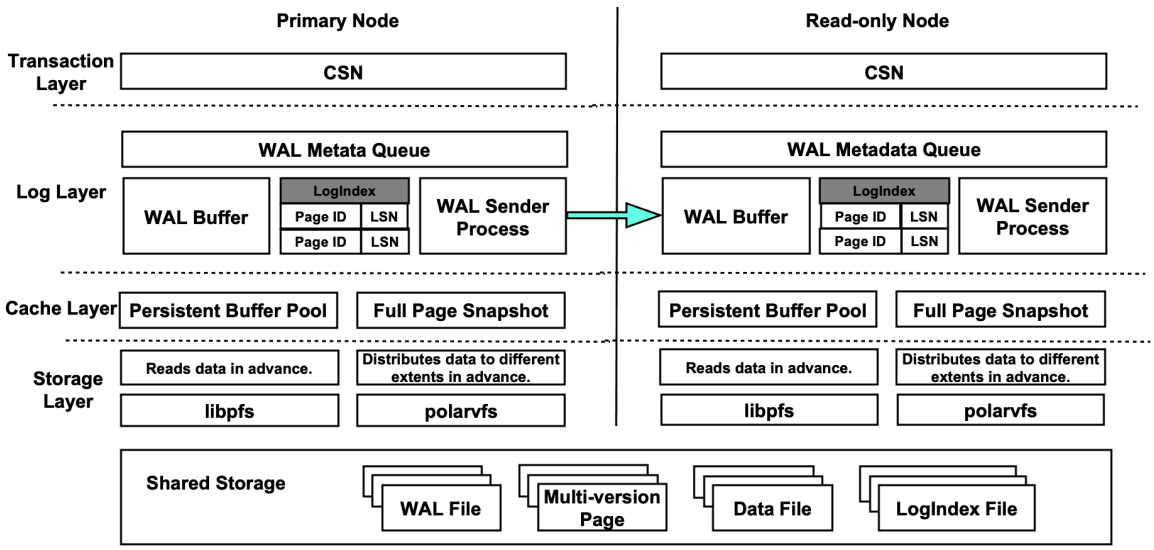


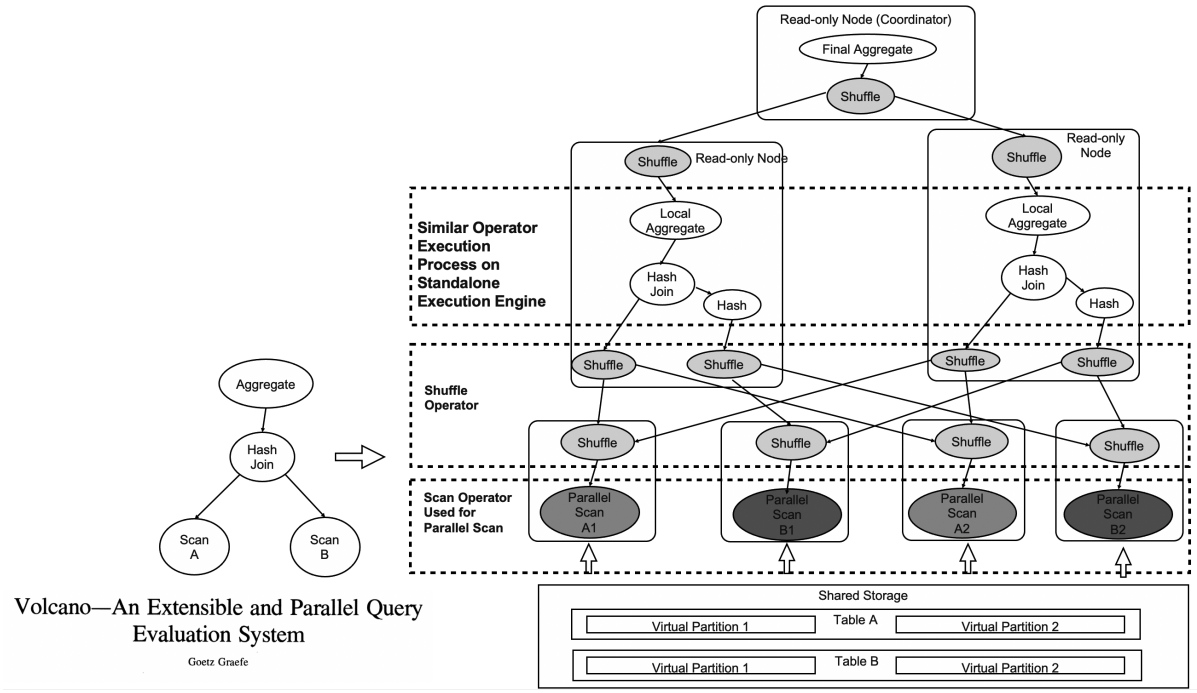
https://apsaradb.github.io/PolarDB-for-PostgreSQL/theory/arch-overview.html#polardb-htap

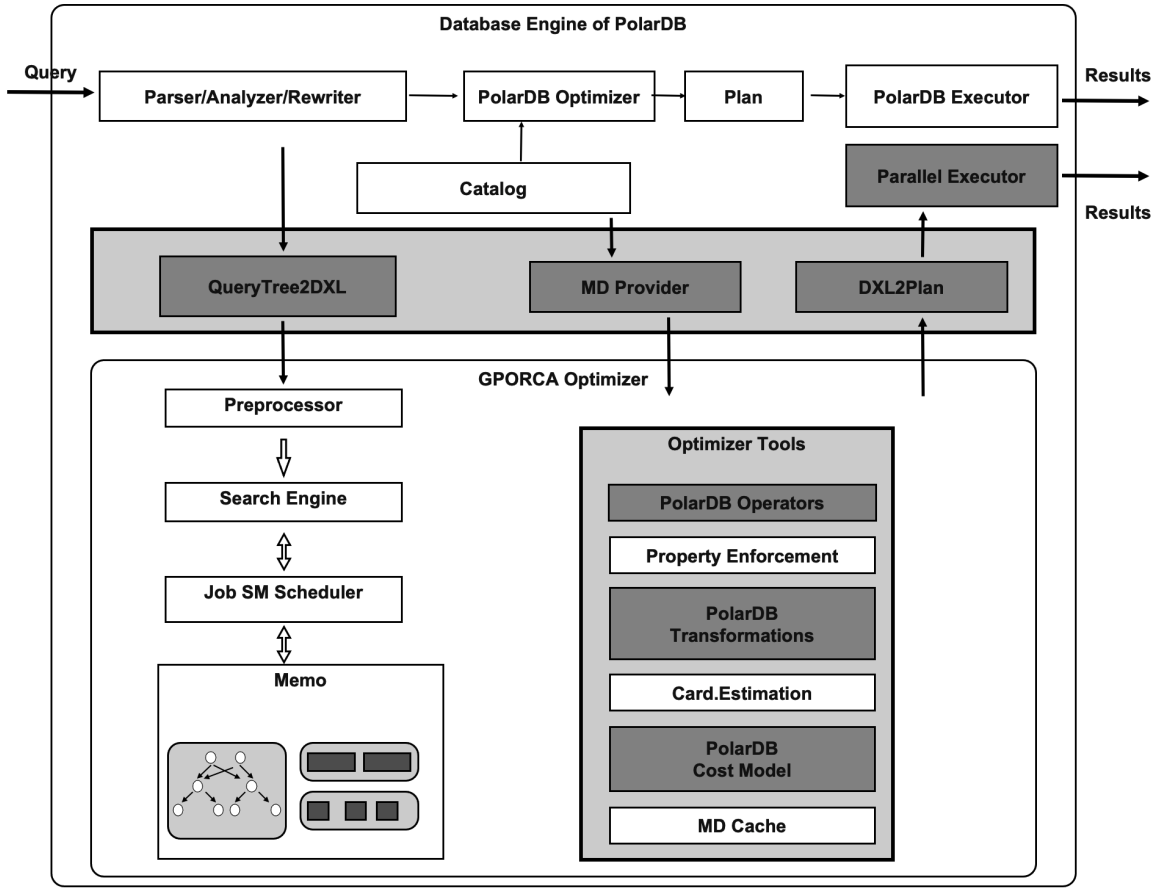
PolarDB uses a shared-storage-based architecture in which computing is decoupled from storage. The conventional shared-nothing architecture is changed to the shared-storage architecture. N copies of data in the compute cluster and N copies of data in the storage cluster are changed to N copies of data 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.

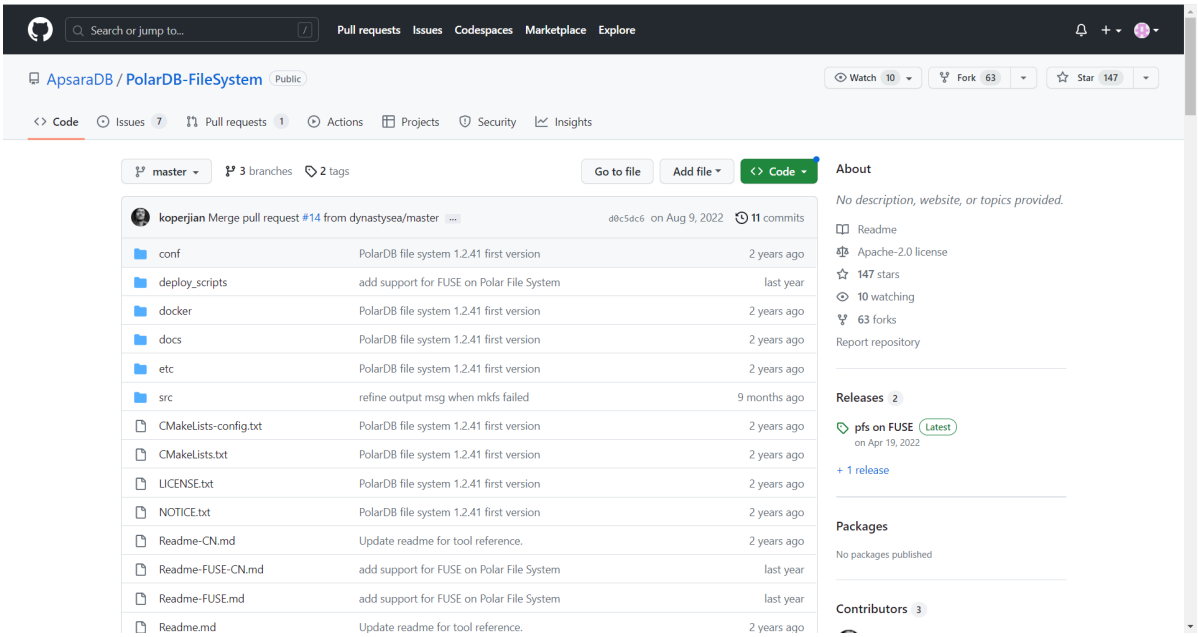
For more information, see Architecture.







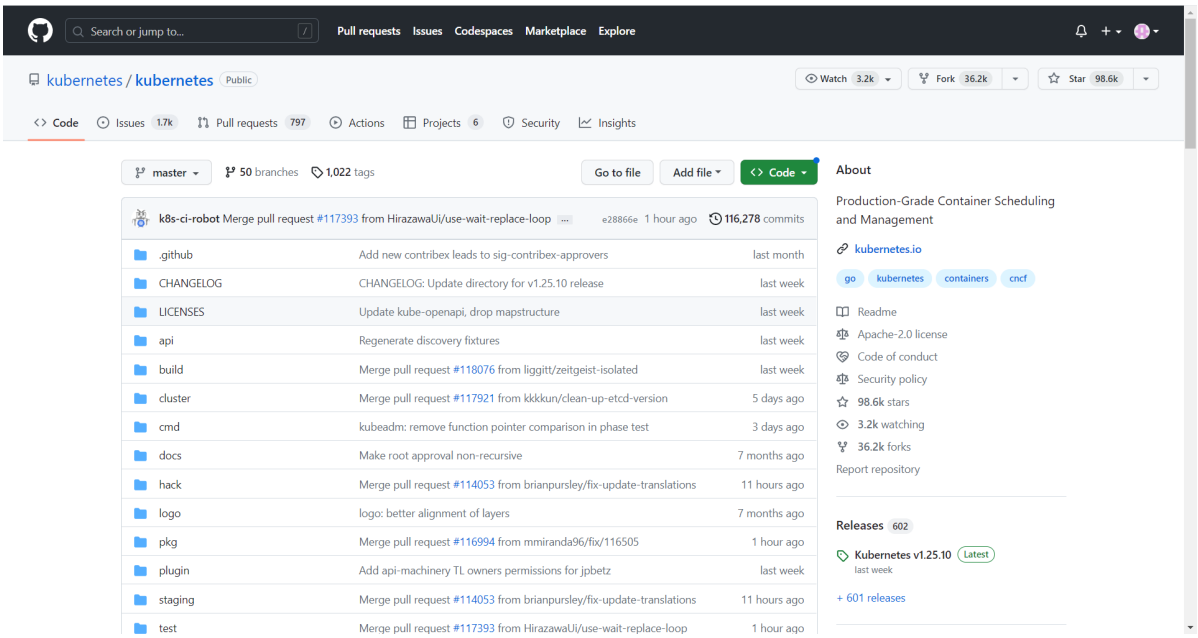
https://github.com/ApsaraDB/PolarDB-FileSystem

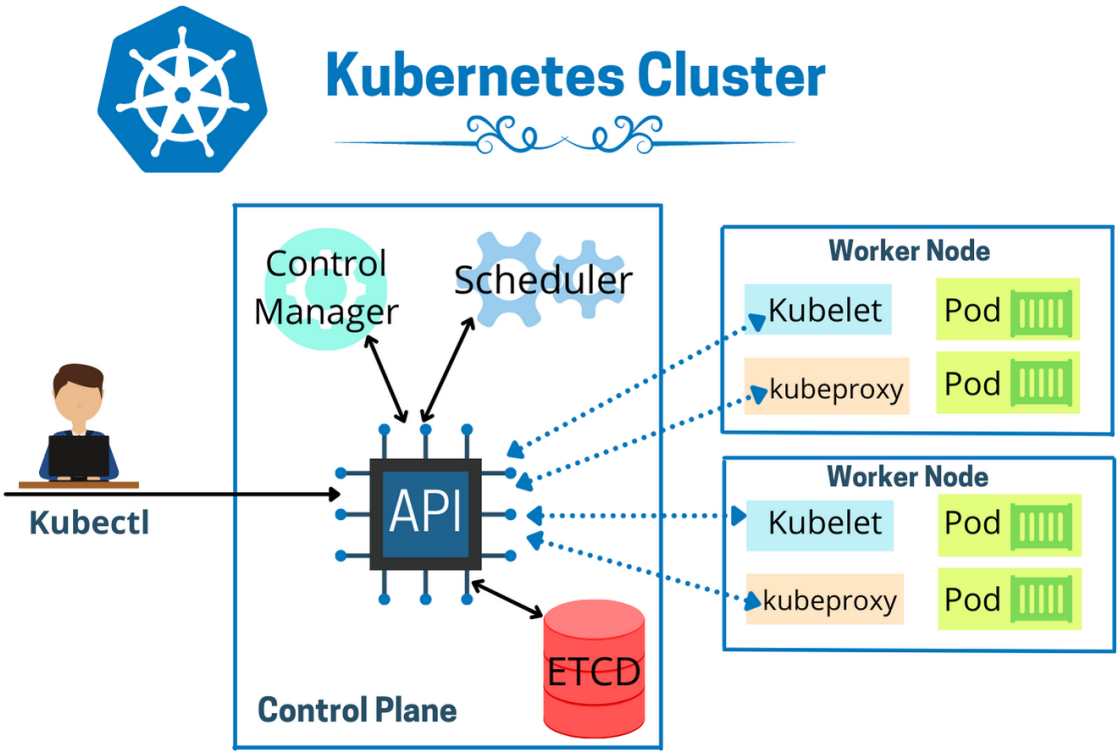


大数据

K8S

https://github.com/kubernetes/kubernetes



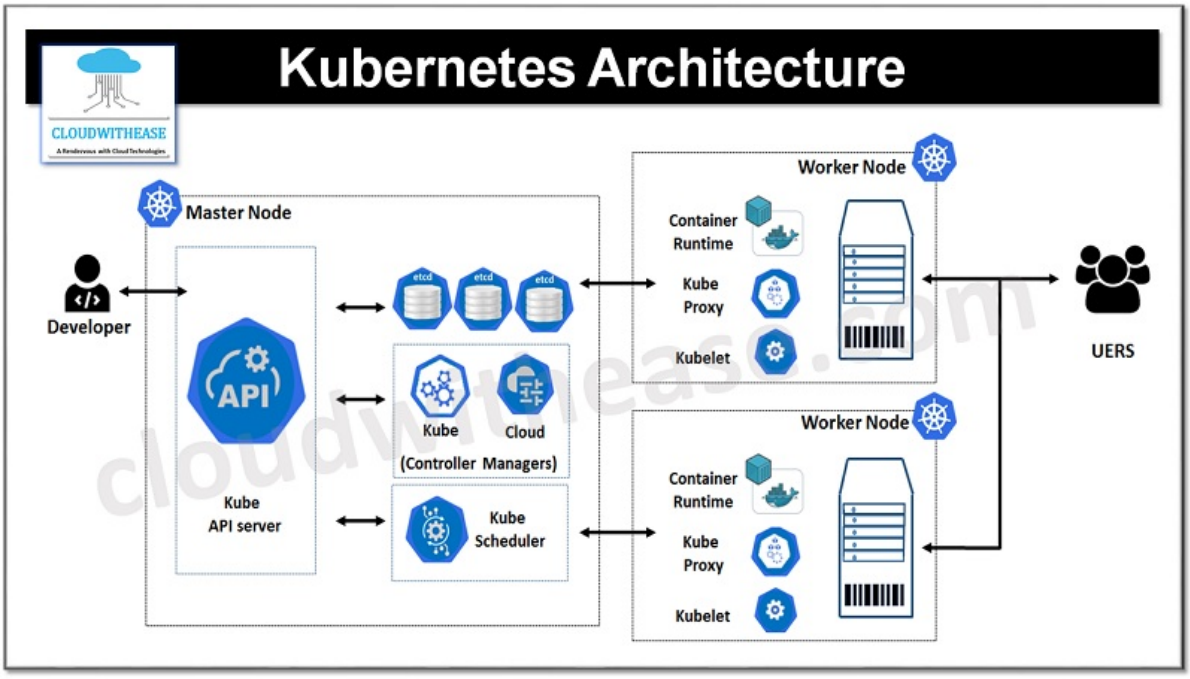


https://justdeployit.hashnode.dev/deploy-an-application-on-kubernetes

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/

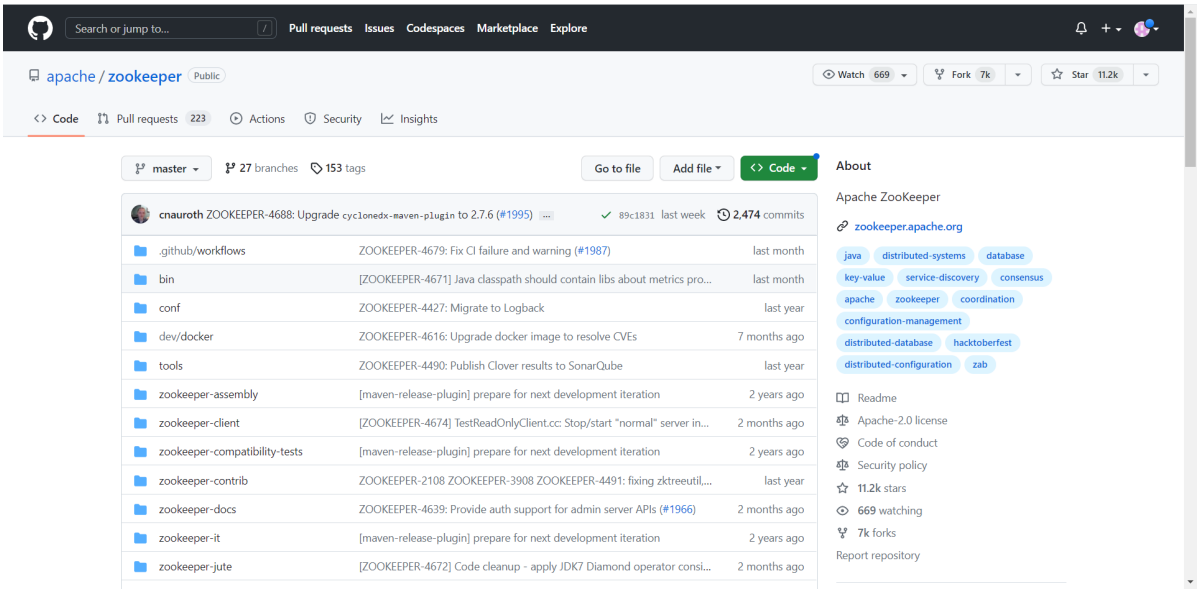
Kube-scheduler is responsible for scheduling Pods to worker nodes. Implementation of sophisticated scheduling algorithms takes a lot of information into account such as availability of resources on each node, constraints mentioned by user, type of available nodes, resource limits and quotas, and other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

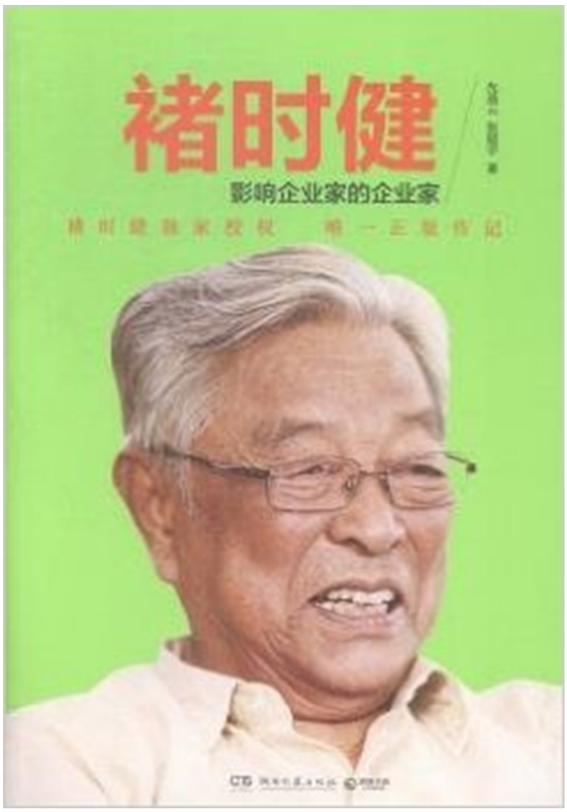
前面留了很多"惜乎"，其实也多少准备了的，终究因为时间限制，没有展开，很是遗憾 - 不能给各位同学更广阔的了解

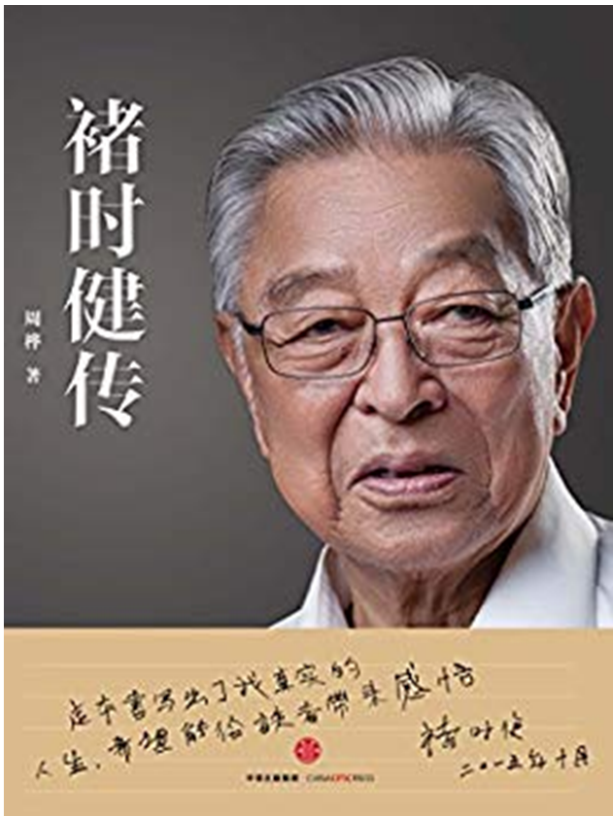
- 学应广博，做要深刻！

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！

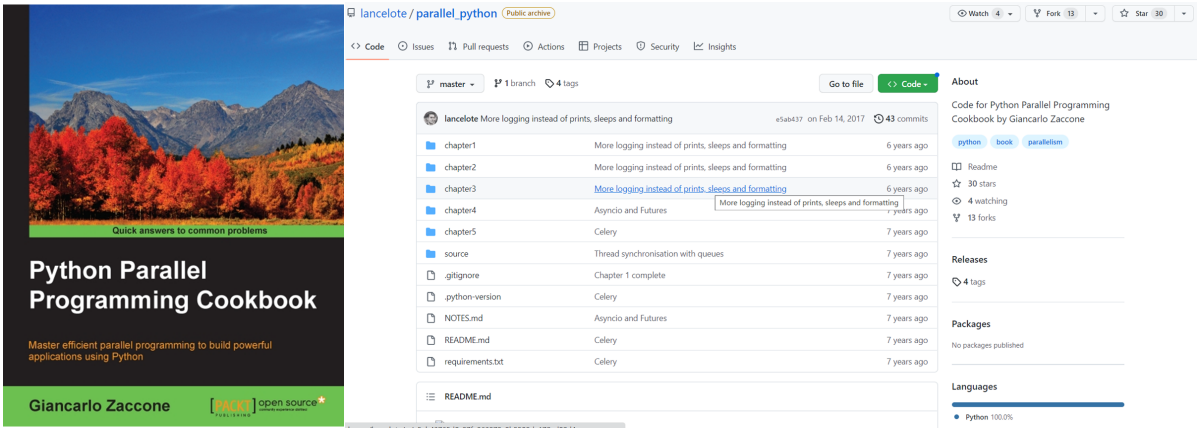
几本书

褚时健传 - 了解下这位老人，体会下"商务思维" - 推荐

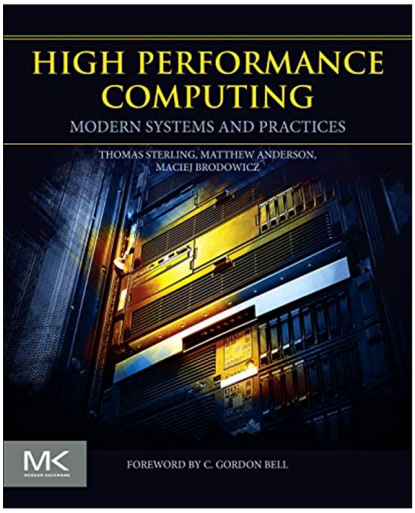




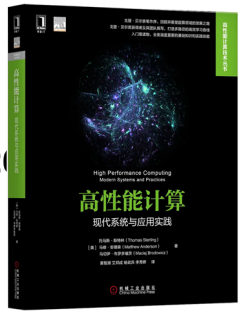
Python Parallel Programming Cookbook - 推荐



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: 2(
- pages 页数: (718)



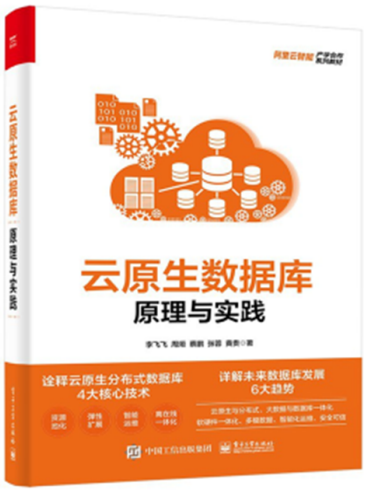
高性能科学与工程计算 [德] Georg Hager, [德] Gerhard Wellein

- 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。
  - 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. 李飞飞 (男)



- 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

几个视频

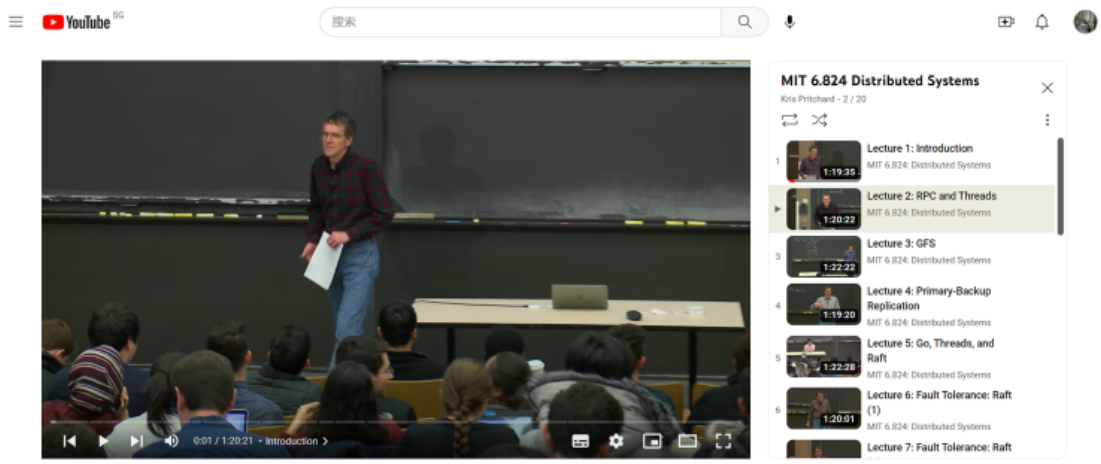
强烈推荐！

https://www.bilibili.com/video/BV1ct411k7jn/?spm_id_from=333.788.recommend_more_video.0

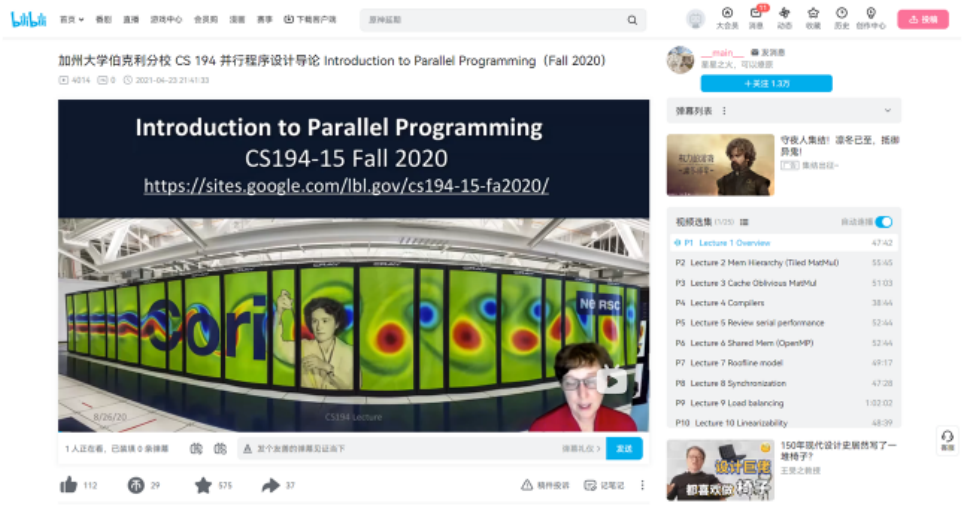
https://www.zhihu.com/people/WangTaoSequoiaDB

MIT 6.824

https://www.youtube.com/watch?v=gA4YXUJX7t8&list=PL4VtY94TBbCx7rQ9r5yc_Tx60RT-hWc45&index=2

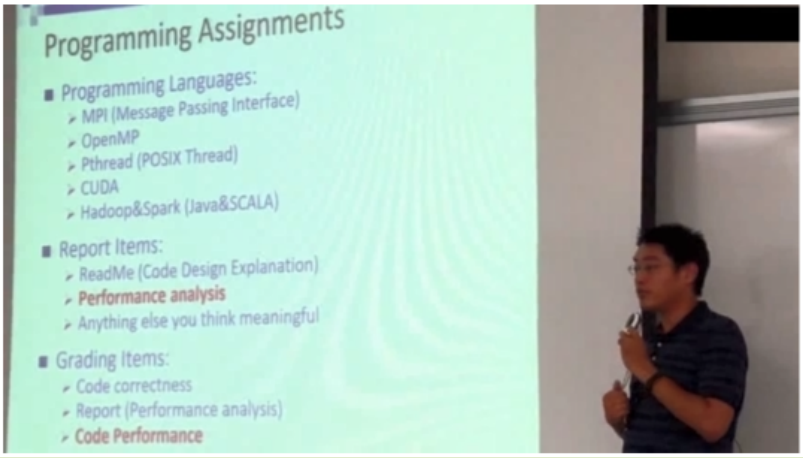


加州大学伯克利分校
CS 194 Introduction to Parallel Programming（Fall 2020）



https://www.bilibili.com/video/BV1QQ4y1o7rn/?spm_id_from=333.337.search-card.all.click&vd_source=e54e9f9a24decfb0d0ebbbf026a36ea3

- MPI, OpenMP, Pthread, CUDA, Hadoop/Spark



预祝大家鹏程万里！