Python计算生态构建

嵩 天

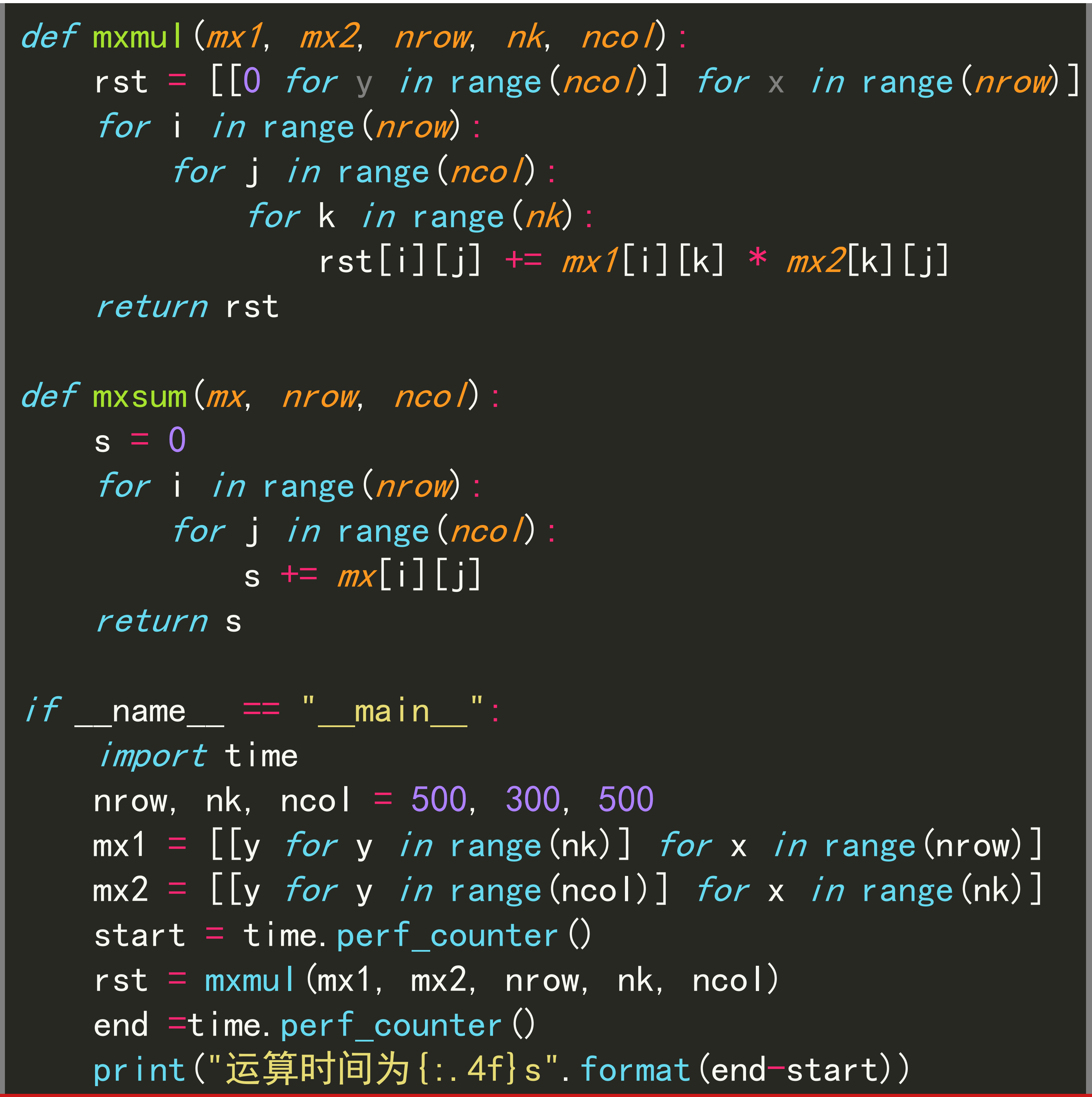
实例2: 矩阵乘法的C语言加速

嵩 天

Python计算生态构建

"矩阵乘法的C语言加速"需求分析

代码纵览



```python
def mxmul(mx1, mx2, nrow, nk, ncol):
    rst = [[0 for y in range(ncol)] for x in range(nrow)]
    for i in range(nrow):
        for j in range(ncol):
            for k in range(nk):
                rst[i][j] += mx1[i][k] * mx2[k][j]
    return rst

def mxsum(mx, nrow, ncol):
    s = 0
    for i in range(nrow):
        for j in range(ncol):
            s += mx[i][j]
    return s

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 500, 300, 500
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = mxmul(mx1, mx2, nrow, nk, ncol)
    end =time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

mxmul.py

模块的编写

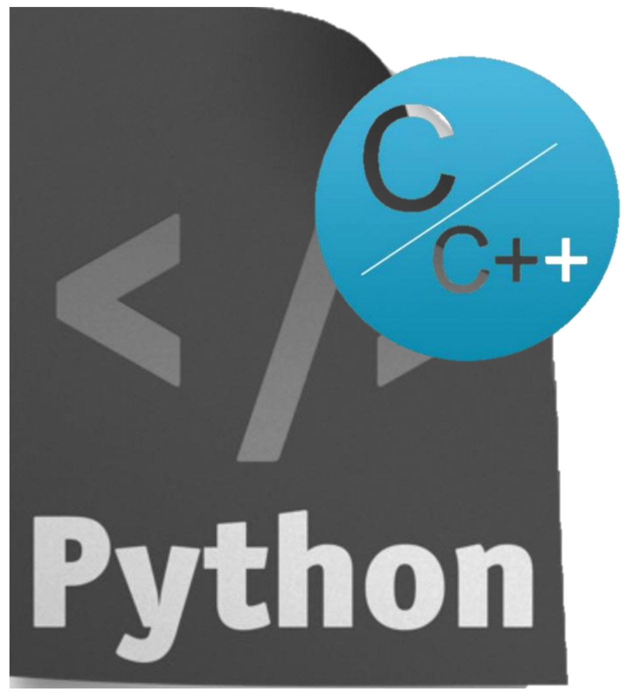
运行结果

nrow, nk, ncol = 50, 30, 50	运算时间为0. 0190s
nrow, nk, ncol = 100, 60, 100	运算时间为0. 1820s
nrow, nk, ncol = 500, 300, 500	运算时间为19. 2137s

需求分析

矩阵乘法的C语言加速

- Python扩展的实例
- 关键计算部分采用C语言实现
- 采用Python语言进行调用和集成



需求分析

C语言比Python语言性能快多少倍?

Python计算生态构建

"矩阵乘法的C语言加速"实例详解

Python▸123

代码纵览

文件结构

- 可编译为.dll的C语言代码：mxmul.h和mxmul.c
- 用来封装.dll的Python模块：cmxmul.py
- 用来测试效果的Python程序：test.py

代码纵览

mxmul.h

```
#ifndef _DLL_H_
#define _DLL_H_

#if BUILDING_DLL
#define DLLIMPORT __declspec(dllexport)
#else
#define DLLIMPORT __declspec(dllimport)
#endif

DLLIMPORT int *mxmul(int nrow, int nk, int ncol, int mx1[][nk], int mx2[][ncol]);

#endif
```

代码详解

```
#ifndef _DLL_H_
#define _DLL_H_

#if BUILDING_DLL
#define DLLIMPORT __declspec(dllexport)
#else
#define DLLIMPORT __declspec(dllimport)
#endif

DLLIMPORT int *mxmul(int nrow, int nk, int ncol, int mx1[][nk], int mx2[][ncol]);

#endif
```

mxmul.h

定义dll的头文件

代码纵览

```c
#include "mxmul.h"
#include <windows.h>
#include <stdlib.h>

/* Parameters: nrow, nk, ncol, mx1, mx2  */
DLLIMPORT int *mxmul(int nrow, int nk, int ncol, int mx1[][nk], int mx2[][ncol])
{
    int x, i, j;
    int *rst;
    rst = malloc(sizeof(int) * nrow * ncol);

    for (i = 0; i < nrow; i++) {
        for (j = 0; j < ncol; j++) {
            rst[i*ncol + j] = 0;
            for (x = 0; x < nk; x++) {
                rst[i*ncol + j] += *(*(mx1 + i) + x) *
                                   *(*(mx2 + x) + j);
            }
        }
    }
   return rst;
}
```

矩阵乘的主体内容

mxmul.c （上）

代码纵览

```
BOOL WINAPI DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
{
    switch(fdwReason)
    {
        case DLL_PROCESS_ATTACH:
        {
            break;
        }
        case DLL_PROCESS_DETACH:
        {
            break;
        }
        case DLL_THREAD_ATTACH:
        {
            break;
        }
        case DLL_THREAD_DETACH:
        {
            break;
        }
    }
```

mxmul.c （下）

代码结果

$$\begin{tikzcd}
\text{mxmul.c} \arrow[dr] & \\
 & \text{mxmul.dll} \\
 & \text{mxmul32.dll} \\
\text{mxmul.h} \arrow[ur] &
\end{tikzcd}$$

代码纵览

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

cmxmul.py

代码详解

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

引入array标准库

引入cffi

调用了dll库

代码详解

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

64位计算机加载

mxmul.dll

32位计算机加载

mxmul32.dll

代码详解

```
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

基本类型参数关联

代码详解

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

列表类型参数关联

代码详解

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

数据的内存拷贝

代码详解

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

声明并调用函数

Python▸123 python

代码纵览：回头看

```python
import array
from cffi import FFI

def cmxmul(nrow, nk, ncol, mx1, mx2):
    ffi = FFI()

    c_nrow = ffi.cast('int', nrow)
    c_nk = ffi.cast('int', nk)
    c_ncol = ffi.cast('int', ncol)

    _mx1 = array.array('l')
    _mx2 = array.array('l')
    [_mx1.fromlist(x) for x in mx1]
    [_mx2.fromlist(x) for x in mx2]

    c_mx1 = ffi.new('int[]', len(_mx1))
    c_mx2 = ffi.new('int[]', len(_mx2))
    ffi.memmove(c_mx1, _mx1, ffi.sizeof(c_mx1))
    ffi.memmove(c_mx2, _mx2, ffi.sizeof(c_mx2))

    ffi.cdef('''
            int *mxmul(int nrow, int nk, int ncol, int *mx1, int *mx2);
            ''')
    try:
        C = ffi.dlopen('mxmul.dll')
    except:
        C = ffi.dlopen('mxmul32.dll')
    c_res = C.mxmul(c_nrow, c_nk, c_ncol, c_mx1, c_mx2)

    return ffi.unpack(c_res, nrow * ncol)
```

cmxmul.py

代码纵览

```python
from cmxmul import cmxmul

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 100, 60, 100
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = cmxmul(nrow, nk, ncol, mx1, mx2)
    end = time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

test.py

测试效果的Python代码

代码详解

```
→ from cmxmul import cmxmul

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 100, 60, 100
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
→   rst = cmxmul(nrow, nk, ncol, mx1, mx2)
    end = time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

导入cmxmul

并调用函数

代码详解

```
from cmxmul import cmxmul

if __name__ == "__main__":
    import time
    nrow, nk, ncol = 100, 60, 100
    mx1 = [[y for y in range(nk)] for x in range(nrow)]
    mx2 = [[y for y in range(ncol)] for x in range(nk)]
    start = time.perf_counter()
    rst = cmxmul(nrow, nk, ncol, mx1, mx2)
    end = time.perf_counter()
    print("运算时间为{:.4f}s".format(end-start))
```

赋初值并计时

运行结果

	Python版本	C版本
nrow, nk, ncol = 50, 30, 50	运算时间为0. 0190s	运算时间为0. 0334s
nrow, nk, ncol = 100, 60, 100	运算时间为0. 1820s	运算时间为0. 1800s
nrow, nk, ncol = 500, 300, 500	运算时间为19. 2137s	运算时间为0. 3135s
nrow, nk, ncol = 1000, 600,1000	运算时间为163. 6118s	运算时间为2. 5936s

Python▸123

Thank you