# PIHM\_CUDA开发记录

## 1 CMAKE编译SUNDIALS库

CMAKE\_CXX\_STANDARD=11

因此，在CMAKE，增加一个CMAKE\_CXX\_STANDARD的变量，值等于11.

## 2 运行SUNDIALS\_v4.0.0中的CUDA示例代码的问题

因为react\_cuda.cpp代码（github下载的），中UserData分配为统一内存的空间。因此，cvAdvDiff\_kry\_cuda\_managed.cu的例子中，将ud = (UserData) malloc(sizeof \*ud);换成在cudaMallocManaged开辟空间。

发现：在Linux系统下可以运行，得到正确结果，但出现段错误的bug； 而在Windows系统下，完全不能运行！

发邮件给LLNL的人员询问上述问题：

Dear Sir or Ms:

Now I compile and try to run the CUDA version CVode example in examples/cvode/cuda.

I have used CUDA 10.1 to compile the code. In cvAdvDiff\_kry\_cuda.cu, and in the function

UserData SetUserData(int argc, char \*argv[])

I change the CPU allocation:

ud = (UserData) malloc(sizeof \*ud);

into the unified memory allocation:

UserData ud;

cudaMallocManaged(&ud, sizeof(struct \_UserData));

Then, I compile and run. In Linux, it's running normally, but In Windows, it cannot be run normally.

Could you help me to solve it? Thanks in advance.

Your Kindly,

Li Jian

经查询，使用统一内存(Unified Memory)必须满足3个条件：

1、64位的Winodws或Linux系统；

2、显卡架构必须在Kepler以上，计算能力在3.0以上；

3、CUDA版本在6.0以上。

分析Windows系统下不能正确运行的原因，并注意排查：

1、我的笔记的显卡不支持VM? 或者盗版的64位Windows OS导致的？

Ubuntu系统的笔记本显卡是GeForce MX150；

WIndows系统的笔记本显卡是GeForce 940MX

2、CUDA的版本，我的笔记本使用10.1，Ubuntu系统的笔记本使用的是CUDA 7.5.

3、尝试，在cuda示例代码中的UserData加入结构体，结构体中要包含数组。

经过调试CVode中的示例代码cvAdvDiff\_kry\_cuda\_\* ，总结一些经验：

1、在Windows系统中的代码，UserData必须是在CPU中分配空间，即malloc()，不能使用统一内存分配(cudaMallocManaged())

2、PIHM的核心调用顺序总结如下：

SetCVodeParam():

cv\_flag = CVodeInit(cvode\_mem, ODE, 0.0, CV\_Y);

(1) ApplyBC ApplyForce

(2) IntcpSnowEt(): 使用cal结构体中的变量，cuda核函数计算elem->wf和elem->ws的变量；

(3) hydrol()：cuda核函数更新计算elem和river结构体中的变量，即elem->ws.surf[tid] = y[offset1]; （举例）

(4) SolveCVode()：

cv\_flag = CVode(cvode\_mem, tout, CV\_Yout, &solvert, CV\_NORMAL);

int ODE()的代码：

dy[tid] = ?; 用elem和river结构体中的变量计算rhs的dy，出现copy vector from device to host的错误!?

CVode自带的例子代码，以及react\_cuda.cpp的核函数都很简单，仅用y更新计算ydot，没有涉及其他的结构体或其他变量，因此没有问题。

但PIHM的ODE以及核函数很难搞，涉及大量的elem和river结构体参与计算，都出现了copy vector from device to host的错误。

问题提出

我认为：ODE调用的核函数中，出现了CPU内存变量。但PIHM涉及的变量（结构体）很多，无法一一排查。计划，将原始的PIHM的elem和river的所有结构体变量，移到pihm\_struct的外部，并声明为全局变量。

解决方案

通过SUNDIALS 4.0.0的示例代码，测试发现：

int func\_rhs()中调用的核函数中，如果需要使用外部结构体变量或其他数组变量，只能按值传递，绝对不能按地址（指针）传递！！！

另外，外部结构体变量定义为全局变量。

不确定的因素

另外，PIHM的ODE()中，所有的变量和CUDA核函数使用的block和grid数目的类型，都按照SUNDIALS的示例代码中命名，如sunindextype, unsigned

## 3 Linux系统下编译PIHM

(1)安装CVODE

make cvode

(2)make model

model: pihm, pihm-fbr, flux-pihm, or flux-pihm-bgc

(3) OpenMP共享内存并行：

make CVODE\_OMP=on [model]

(4) PIHM自带了V2.7的CVODE软件包，可以使用以下方法编译成静态链接库(.lib, .a)文件，供PIHM调用：

编译OPENMP和MPI版本的CVODE，使用cmake:

cmake -DOPENMP\_ENABLE=on ..

cmake -DMPI\_ENABLE=on .. # PIHM目前还不支持MPI并行

## 4 开发PIHM\_CUDA的过程记录

基于SUNDIALS\_3.2.1或以上版本，开发基于NVector\_CUDA的GPU异构并行版本的PIHM模型，开发记录如下：

4.1 CUDA (nvcc)与C++程序混合编译

首先是，解决 CUDA (nvcc)与C++程序，混合编译的问题，参考GTC的文档。

CUDA编译错误：

1>pihm\_cuda.lib(ode\_cuda.cu.obj) : error LNK2001: 无法解析的外部符号 \_\_cudaRegisterLinkedBinary\_43\_tmpxft\_0000159c\_00000000\_7\_ode\_cuda\_cpp1\_ii\_67a7fccd

1>pihm\_cuda.lib(hydrol\_elem.cu.obj) : error LNK2001: 无法解析的外部符号 \_\_cudaRegisterLinkedBinary\_46\_tmpxft\_00002490\_00000000\_7\_hydrol\_elem\_cpp1\_ii\_bee8d420

1>pihm\_cuda.lib(hydrol\_river.cu.obj) : error LNK2001: 无法解析的外部符号 \_\_cudaRegisterLinkedBinary\_47\_tmpxft\_00000c24\_00000000\_7\_hydrol\_river\_cpp1\_ii\_0b8c0462

解决方法：

Your nvcc command line specifies a compile-only operation (-rdc=true -c).

g++ does not do any device code linking. So in a scenario like this, when doing the final link operation using g++ an extra device code link step is required.

正确设置nvcc编译参数

nvcc -arch=sm\_35 -rdc=true -c file.cu # 注意编译参数： -rdc=true -c或者-compile 独立编译cu，支持 \_\_device\_\_ 函数在另一个.c或.cu文件中调用。

nvcc -arch=sm\_35 -dlink -o file\_link.o file.o -lcudadevrt -lcudart # 注意编译参数： -dlink 意思是：Perform Device Link 连接设备函数

g++ file.o file\_link.o main.cpp -L<path> -lcudart -lcudadevrt # 用g++连接以上的lib文件，生成exe程序

4.2 VS路径设置

在Visual Studio中添加正确的配置路径，对编写makefile文件有帮助。

lib路径：

C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v10.1\lib\x64

D:\GPGPU-dev\SUNDIALS\_GPU\sundials-3.2.1\build\src\nvec\_cuda\Release

D:\GPGPU-dev\SUNDIALS\_GPU\sundials-3.2.1\build\src\cvode\Release

D:\cygwin\home\MM-PIHM-0.10\pihm\_cvode3.2.1\x64\Release\

添加：

cudart.lib

cudadevrt.lib

sundials\_nveccuda.lib

sundials\_cvode.lib

pihm\_vs.lib

# PIHM\_CUDA代码调试记录

src\_局部变量.7z elem\_d river\_d结构体作为局部变量代入核函数

src\_CVode通过 elem\_d river\_d作为全局变量，经过调试，CUDA版本的CVode求解器，安全通过测试。

## 问题1

经过努力，发现：

int ODE()中，

调用计算dy\_d的核函数，elem\_d作为形参代入，

只能按值传递，绝对不能按地址（指针）传递！！！

这样，按值传递方式，当结构体规模很大时，效率将会很低！

2021.03.31早上8点40分，调用CVode()的CUDA版本正常运行！！！

## 问题2

SUNLinSolFree(Linsol); // 在ode\_cuda.cu程序中，如果使用该注销，将导致CVode()调用求解出错!!!

## 问题3

调用

IntcpSnowEt(t, pihm->ctrl.stepsize, &elem\_d);

或者

hydrol(t, CV\_Y, pihm\_d, &elem\_d, &river\_d);

会引起failed to copy vector from device to host的错误?!

为此，设置一个debug\_func():

void debug\_func(int, elem\_struct\_d);

\_\_global\_\_ void func\_kernel(int, elem\_struct\_d);

通过Cvode测试计算。

如果声明为：

void debug\_func(int, elem\_struct\_d \*);

\_\_global\_\_ void func\_kernel(int, elem\_struct\_d \*);

确定引起failed to copy vector from device to host的错误!

看来，不能使用指针形式的形参，在任何核函数的调用参数当中！！！

## 问题4

水文过程计算的影响很大。

在nvcc编译中，加入了hydrol()的核函数计算，就造成failed to copy vector from device to host的错误!

排除hydrol()的核函数计算，就没有问题！原因是：hydrol()中的核函数调用中，结构体(elem\_d, river\_d)作为形参就不行。

ODE()中执行的核函数，只能通过简单的中间变量发生y与ydot之间的传递？

## 问题5

elem\_d和River\_d相关变量，修改为数组，作为形参代入各函数。但是问题出现了：

int ODE()的形参是固定的，而elem\_d和river\_d的数组是局部变量，编译出现局部变量数组没有定义的问题?!

我不想将elem\_d和river\_d的数组定义为全局变量。因为，局部变量的计算效率更高。

## 问题6

所有水文过程的CUDA核函数，与CPU版本的做了对比，确定核函数计算正确。

但是，ODE中计算的CV\_Y和CV\_Ydot的cuda数组，内容不对。而y\_d和dy\_d数组内容正确。与函数的static类型定义有关？

## CVODE代码版本

## Release History ##

Date | SUNDIALS | ARKODE | CVODE | CVODES | IDA | IDAS | KINSOL

---------|-------------|-------------|-------------|-------------|-------------|-------------|-------------

Dec 2018 | 4.0.0 | 3.0.0 | 4.0.0 | 4.0.0 | 4.0.0 | 3.0.0 | 4.0.0

Oct 2018 | 3.2.1 | 2.2.1 | 3.2.1 | 3.2.1 | 3.2.1 | 2.2.1 | 3.2.1

## CVODE API

NV\_Ith\_S(v, i)

This macro gives access to the individual components of the data array of an N\_Vector, using standard 0-based C indexing.

The assignment r = NV\_Ith\_S(v,i) sets r to be the value of the i-th component of v.

The assignment NV\_Ith\_S(v,i) = r sets the value of the i-th component of v to be r.

Here i ranges from 0 to n?1 for a vector of length n.

Implementation:

#define NV\_Ith\_S(v,i) ( NV\_DATA\_S(v)[i] )

NV\_DATA\_S(v)

The assignment v\_data = NV\_DATA\_S(v) sets v\_data to be a pointer to the first component of the data for the N\_Vector v.

Similarly, the assignment NV\_DATA\_S(v) = v\_data sets the component array of v to be v\_data by storing the pointer v\_data.

Implementation:

#define NV\_DATA\_S(v) ( NV\_CONTENT\_S(v)->data )

# CVode\_CUDA调用中ODE()中变量类型对计算的影响分析

SUNDIALS\_CVode的CUDA版本的API调用中，如果int ODE(\*y, \*dy, NVector, UserData \*ud)及核函数中数组变量的命名，不符合API的要求，会导致如下的致命错误：failed to copy vector from device to host

int ODE( )代入了二级指针的形参，也会导致上面的致命错误！

首先可以确定int ODE()函数，只允许这固定的4个形参（CVode API决定的）。也就是说，PIHM模型中的elem和river结构体变量元素，必须包含在UserData \*ud结构体中。

下面总结PIHM模型中的状态变量（数组）以不同形式定义，对CVode CUDA API调用的影响（避免出现致命错误）。

（1）结构体作为形参

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 值传递（数组形参）:ODE(\*, elem\_d) | | 地址传递:ODE(\*, \* elem\_d) | |
| 全局变量 | elem\_d.ws.var[tid]=x; | × | elem\_d->ws.var[tid]=x; | × |
| x=elem\_d.ws.var[tid]; | √ | x=elem\_d->ws.var[tid]; | × |
| 局部变量（ud结构体中的变量） | ud.elem\_d.ws.var[tid]=x; | × | ud->elem\_d.ws.var[tid]=x; | × |
| x=ud.elem\_d.ws.var[tid]; | √ | x=ud->elem\_d.ws.var[tid]; | √ |

（2）数组作为形参

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 值传递（数组形参）ODE(\*, elem\_d\_ws\_var) | | 地址传递ODE(\*, \* elem\_d\_ws\_var) | |
| 全局变量 | elem\_d\_ws\_var[tid]=x; | √ | elem\_d\_ws\_var[tid]=x; | √ |
| x=elem\_d\_ws\_var[tid]; | √ | x=elem\_d\_ws\_var[tid]; | √ |
| 局部变量（ud结构体中的变量） | ud.elem\_d\_ws\_var[tid]=x; | × | ud->elem\_d\_ws\_var[tid]=x; | √ |
| x=ud.elem\_d\_ws\_var[tid]; | √ | x=ud->elem\_d\_ws\_var[tid]; | √ |