Erik Smistad

RECENT COMMENTS

TIRTHESH SAYS:
correct the spelling of SquareRoot in tests.cpp code.

CAT&ANT SAYS:
Hello Erik, I don't find the gmock after...

CAT&ANT SAYS:
Great. Thanks a lot for this helpful info!!

ERIK SMISTAD SAYS:
The image sampler in OpenCL handles the edges (the variable sampler...

ROYI SAYS:
Hi, Great post. I would like to...

POPULAR POSTS

OPENCL
Getting started with OpenCL and GPU Computing
21 JUN, 2010

Getting started with OpenCL and GPU Computing

BY ERIK SMISTAD · PUBLISHED JUNE 21, 2010 · UPDATED MAY 28, 2016

OpenCL (Open Computing Language) is a new framework for writing programs that execute in parallel on different compute devices (such as CPUs and GPUs) from different vendors (AMD, Intel, ATI, Nvidia etc.). The framework defines a language to write "kernels" in. These kernels are the functions which are to run on the different compute devices. In this post I explain how to get started with OpenCL and how to make a small OpenCL program that will compute the sum of two lists in parallel.

Installing and setting up OpenCL on your computer

First of all you need to download the newest drivers to your graphics card. This is

Erik Smistad Retweeted

NTNUmedisin og helse
@NTNUhelse

Møt blant annet Svein Erik Måsøy fra #CIUS som skal fortelle om det norske ultralydeventyret forskningsdagene.no/artikler/verdi… @Forskningsdg @NTNU

16h

Erik Smistad Retweeted

Dan Hulme
@nasaldemons

Replying to @caffe2ai @bobpoekert

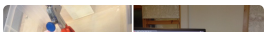
Why are you competing with NNEF rather than supporting it?

Sep 8, 2017

Erik Smistad Retweeted

P Cantillon-Murphy
@cantillonmurphy

Thank you Thomas and team @SINTEF Trondheim for hosting & sharing your work in airway navigation. Looking forward to future collaboration!



GITHUB PROJECTS

- 3D-Gradient-Vector-Flow-for-Matlab
- caffe
- cpd
- CustusX
- Detect-Screen-GPU-With-OpenCL

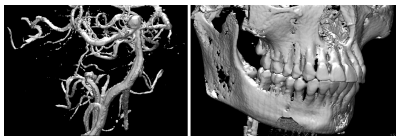


IMAGE PROCESSING & GFX / OPENCL / OPENGL
Marching Cubes implementation using OpenCL and OpenGL
7 OCT, 2011

GENERAL
Getting started with Google Test (GTest) on Ubuntu
5 JUL, 2012

IMAGE PROCESSING & GFX / OPENCL
GPU-based Gradient Vector Flow using OpenCL
23 JUL, 2012

important because OpenCL will not work if you don't have drivers that support OpenCL.

To install OpenCL you need to download an implementation of OpenCL. The major graphic vendors Nvidia and AMD/ATI have both released implementations of OpenCL for their GPUs. These implementation come in a so called software development kits and often include some useful tools such as a visual profiler. The next step is to download and install the SDK for the GPU you have on your computer. Note that not all graphic cards are supported. A list of which graphic cards are supported can be found on the vendors websites.

For AMD/ATI GPUs download the AMD APP SDK (formerly known as AMD Stream SDK)
For Nvidia GPUs download the CUDA Toolkit

The installation steps differ for each SDK and the OS you are running. Follow the installation manual of the SDK carefully. Personally I use Ubuntu Linux and have an AMD 7970 graphics card. Below are some installation steps for this specific setup.

Installing OpenCL on Ubuntu Linux with AMD graphics card

To install the latest AMD drivers on Ubuntu 12.04 open additional drivers and install/active the one called "ATI/AMD proprietary FGLRX graphic driver (post-release updates)".
After that is done, restart and download and extract the AMD APP SDK.

AMD APP SDK 2.8 includes an installer. Run this with the command:

```
sudo sh Install-AMD-APP.sh
```

Next, install the OpenCL headers files

```
sudo apt-get install opencl-headers
```

And your done! Note that the AMD APP SDK and its samples is located at /opt/AMDAPP.

- django-tinymce-gallery
- FAST
- FAST-example-project
- FLTK-OpenGL-OpenCL-Interoperability
- freenect2-test
- GPU-Marching-Cubes
- GPU-Multigrid-Gradient-Vector-Flow
- Gradient-Vector-Flow
- GTest
- keras
- Level-Set-Segmentation
- libfreenect2
- Memory-mapped-file
- midas-journal-111
- Multiple-Windows-Single-OpenGL-Context
- Nerve-Segmentation
- node-webchat
- OpenCL-Gaussian-Blur
- OpenCL-Getting-Started
- OpenCL-GVF
- OpenCL-Level-Set-Segmentation
- OpenCLUtilities
- OpenCLUtilityLibrary
- opengles-book-samples
- Qt-Context-Sharing

Installing OpenCL on Ubuntu Linux with NVIDIA graphics card

Download the CUDA toolkit for Ubuntu from NVIDIAs CUDA site. Open a terminal an run the installation file with the command:

```
sudo sh cudatoolkit_3.1_linux_64_ubuntu
```

Download the Developer Drivers for Linux at the same website and install it by first stopping X, running the file and start X again. To stop X use:

```
sudo /etc/init.d/gdm stop
```

Then get a terminal up by pressing CTRL+ALT+F5, login and navigate to where you downloaded the devdriver then type:

```
sudo sh devdriver_3.1_linux_64_256.40.
```

After the driver has been installed start x again by typing

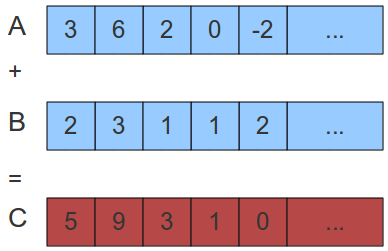
```
startx
```

Before compiling an OpenCL application you need to add the path to the lib folder of CUDA to LD_LIBRARY_PATH like so:

```
export LD_LIBRARY_PATH=/usr/local
/cuda/lib64
```

Your first OpenCL program – Vector addition

To demonstrate OpenCL I explain how to perform the simple task of vector addition. Suppose we have two lists of numbers, A and B, of equal size. The task of vector addition is to add the elements of A with the elements of B and put the result in the element of a new list called C of the same size. The figure below explains the operation.



Two lists A and B and the result list C of vector addition on A and B

The naive way of performing this operation is to simply loop through the list and perform the operation on one element at a time like the C++ code below:

```
for(int i = 0; i < LIST_SIZE; i++) {
    C[i] = A[i] + B[i];
}
```

This algorithm is simple but has a linear time complexity, O(n) where n is the size of the list. But since each iteration of this loop is independent on the other iterations this operation is data parallel, meaning that each iteration can be computed simultaneously. So if we have n cores on a processor this operation can be performed in constant time O(1).

To make OpenCL perform this operation in parallel we need to make the kernel. The kernel is the function which will run on the compute device.

The kernel

The kernel is written in the OpenCL language which is a subset of C and has a lot of math and vector functions included. The kernel to perform the vector addition operation is defined below.

```
__kernel void vector_add(__global const

    // Get the index of the current ele
    int i = get_global_id(0);

    // Do the operation
    C[i] = A[i] + B[i];
}
```

The host program

The host program controls the execution of kernels on the compute devices. The host program is written in C, but bindings for other languages like C++ and Python exists. The OpenCL API is defined in the cl.h (or opencl.h for apple) header file. Below is the code for the host program that executes the kernel above on compute device. I will not go into details on each step as this is supposed to be an introductory article although I can recommend the book "The OpenCL Programming Book" if you want to dive into the details. The main steps of a host program is as follows:

- Get information about the platform and the devices available on the computer (line 42)
- Select devices to use in execution (line 43)
- Create an OpenCL context (line 47)
- Create a command queue (line 50)
- Create memory buffer objects(line 53-58)
- Transfer data (list A and B) to memory buffers on the device (line 61-64)
- Create program object (line 67)
- Load the kernel source code (line 24-35) and compile it (line 71) (online exeuction) or load the precompiled binary OpenCL program (offline execution)
- Create kernel object (line 74)
- Set kernel arguments (line 77-79)
- Execute the kernel (line 84)
- Read memory objects. In this case we read the list C from the compute device (line 88-90)

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3
 4 #ifdef __APPLE__
 5 #include <OpenCL/opencl.h>
 6 #else
 7 #include <CL/cl.h>
 8 #endif
 9
10 #define MAX_SOURCE_SIZE (0x100000)
11
12 int main(void) {
13     // Create the two input vectors
14     int i;
15     const int LIST_SIZE = 1024;
```

```
16      int *A = (int*)malloc(sizeof(in
17      int *B = (int*)malloc(sizeof(in
18      for(i = 0; i < LIST_SIZE; i++)
19          A[i] = i;
20          B[i] = LIST_SIZE - i;
21      }
22
23      // Load the kernel source code
24      FILE *fp;
25      char *source_str;
26      size_t source_size;
27
28      fp = fopen("vector_add_kernel.c
29      if (!fp) {
30          fprintf(stderr, "Failed to
31          exit(1);
32      }
33      source_str = (char*)malloc(MAX_
34      source_size = fread( source_str
35      fclose( fp );
36
37      // Get platform and device info
38      cl_platform_id platform_id = NU
39      cl_device_id device_id = NULL;
40      cl_uint ret_num_devices;
41      cl_uint ret_num_platforms;
42      cl_int ret = clGetPlatformIDs(1
43      ret = clGetDeviceIDs( platform_
44              &device_id, &ret_num_de
45
46      // Create an OpenCL context
47      cl_context context = clCreateCo
48
49      // Create a command queue
50      cl_command_queue command_queue
51
52      // Create memory buffers on the
53      cl_mem a_mem_obj = clCreateBuff
54              LIST_SIZE * sizeof(int)
55      cl_mem b_mem_obj = clCreateBuff
56              LIST_SIZE * sizeof(int)
57      cl_mem c_mem_obj = clCreateBuff
58              LIST_SIZE * sizeof(int)
59
60      // Copy the lists A and B to th
61      ret = clEnqueueWriteBuffer(comm
62              LIST_SIZE * sizeof(int)
63      ret = clEnqueueWriteBuffer(comm
64              LIST_SIZE * sizeof(int)
65
66      // Create a program from the ke
67      cl_program program = clCreatePr
68              (const char **)&source_
69
70      // Build the program
71      ret = clBuildProgram(program, 1
72
73      // Create the OpenCL kernel
74      cl_kernel kernel = clCreateKern
75
76      // Set the arguments of the ker
77      ret = clSetKernelArg(kernel, 0,
78      ret = clSetKernelArg(kernel, 1,
79      ret = clSetKernelArg(kernel, 2,
80
81      // Execute the OpenCL kernel or
82      size_t global_item_size = LIST_
83      size_t local_item_size = 64; //
84      ret = clEnqueueNDRangeKernel(co
85              &global_item_size, &loc
86
87      // Read the memory buffer C on
88      int *C = (int*)malloc(sizeof(in
89      ret = clEnqueueReadBuffer(comma
90              LIST_SIZE * sizeof(int)
91
92      // Display the result to the sc
93      for(i = 0; i < LIST_SIZE; i++)
94          printf("%d + %d = %d\n", A[
95
96      // Clean up
97      ret = clFlush(command_queue);
98      ret = clFinish(command_queue);
99      ret = clReleaseKernel(kernel);
```

```
100     ret = clReleaseProgram(program)
101     ret = clReleaseMemObject(a_mem_
102     ret = clReleaseMemObject(b_mem_
103     ret = clReleaseMemObject(c_mem_
104     ret = clReleaseCommandQueue(com
105     ret = clReleaseContext(context)
106     free(A);
107     free(B);
108     free(C);
109     return 0;
110 }
```

To make OpenCL run the kernel on the GPU you can change the constant CL_DEVICE_TYPE_DEFAULT to CL_DEVICE_TYPE_GPU in line 43. To run on CPU you can set it to CL_DEVICE_TYPE_CPU. This shows how easy OpenCL makes it to run different programs on different compute devices.

The source code for this example can be downloaded here.

Compiling an OpenCL program

If the OpenCL header and library files are located in their proper folders (/usr/include and /usr/lib) the following command will compile the vectorAddition program.

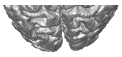
```
gcc main.c -o vectorAddition -l OpenCL
```

How to learn more

To learn more about OpenCL I recommend the book from Fixstars called The OpenCL programming book. Below are some links to useful sites with information on OpenCL:

- Nvidia's page on OpenCL
- AMD/ATI's page on OpenCL
- Official website of OpenCL

Tags: GPGPU GPU computing OpenCL



Level set segmentation on GPUs using OpenCL

JULY 11, 2013

Real-time medical surface extraction with FAST

DECEMBER 5, 2015

OpenCL

OpenCL C++ Utilities

SEPTEMBER 16, 2011

149 RESPONSES

Comments 139 Pingbacks 10

Gabriel · August 20, 2016 at 00:51
Thanks guy xD

Reply

shilpa · March 12, 2015 at 13:08
Hi Erik,

Your program is pretty good to understand . I ran this program on a octacore machine but could not see the expected parallel processing of instructions.
The time taken for this code was more than a sequential code for vector addition. I doubt , this program is not running on multiple cores parallely on my system. Could you plz help ?

Regards,
Shilpa

Reply

Erik Smistad ★
· March 23, 2015 at 17:18
This is probably because the vectors are so small (only 1024 items). Try to increase the size of the vector to lets say 1024*1024*1024, and you will probably see a speedup.

Reply

celio
· August 31, 2016 at 10:01
i changed item_size to 1024*1024*32 but i did not defferene between GPU &CPU

Reply

KSSR · November 28, 2014 at 16:51
hello all,
I need instruction about setting up opencl environemnt for Multicore system i.e on GPU.

Reply

kevinkit · July 21, 2014 at 15:28
Sry I was wrong with the cl_mem stuff but nevertheless the "+1" is missing. (THIS I NOT C++!)

Reply

kevinkit ⊙ July 20, 2014 at 20:01

Doesn't it has to be

int *A = (int*)malloc(sizeof(int)*(LIST_SIZE +1));

and this in every other memory allocation furthermore when you allocate the memory objects it should be

cl_mem a_mem_obj = clCreateBuffer(context, CL_MEM_READ_ONLY, (LIST_SIZE+1) * sizeof(cl_mem), NULL, &ret);

instead of cl_int.

Reply

Fabio ⊙ July 17, 2014 at 11:11

Hi, I have a question.
When you call clEnqueueReadBuffer, you don't have to pass as parameter also the list of events to wait before read the buffer? I mean, you can get the event identifier from clEnqueueNDRangeKernel and pass it to clEnqueueReadBuffer, otherwise the program may read the results before the sum is completed.
If it's not needed, why?

Reply

Erik Smistad ⊙ July 17, 2014 at 13:31

The third argument of clEnqueueReadBuffer with the value CL_TRUE ensures that this call is blocking. That means that the function will not return until it has read the buffer and thus explicit synchronization is not needed. However, if you set this argument to CL_FALSE you have to do explicit synchronization using events as you suggest.

Reply

Fabio ⊙ July 17, 2014 at 14:27

I know that the third argument make the call blocking, but I think that means that (as you say) "the function will not return until it has read the buffer". Howerer the documentation doesn't say anything about the previous enqueued operations for this argument. Maybe is not so clear.
The documentation says also that you must ensure that "All commands that use this buffer object have finished execution before the read command begins execution"

Reply

Erik Smistad
July 17, 2014 at 15:18
Ah yes, that is a good point. The clue here is that your command queue is created with in-order execution (this is default and most devices doesn't support out-of-order execution). In-order execution guarantees that all of the queued commands will be executed in the order in which they were added to the queue. Thus, for clEnqueueReadBuffer to finish, all other queued operations have to finish first.

See http://www.khronos.org/registry/cl/sdk/1.0/docs/man/xhtml/clCreateCommandQueue.html for more info on this

Reply

Fabio
July 17, 2014 at 15:33
Aaahh sorry 🙂
Your queue is not out-of-order.

I'm working with an aout-of-order queue and I have some problems so I'm trying to understand...
Thanks

Reply

Anonymous April 9, 2014 at 10:15
Have you found out how to fix the failure?

Name (required):

I added a printf() to your code after line 43:if (ret != CL_SUCCESS) {printf("Error: Failed to query platforms! (%d)\n", ret);return EXIT_FAILURE;}
after compiling, running it gives me this error: "Failed to query platforms (-1001)"

Reply

meenu February 13, 2014 at 10:03
I am currently using ubuntu13.04 and have a VGA compatible controller: NVIDIA Corporation GK107 [GeForce GT 630 OEM] (rev a1) ... My open CL samples are running fine for CL_DEVICE_TYPE_DEFAULT and CL_DEVICE_TYPE_CPU... But they are not able to find OPENCL devices for CL_DEVICE_TYPE_GPU...

Reply

Erik Smistad
February 13, 2014 at 12:28
The NVIDIA OpenCL platform only support GPUs. So most likely you have more than one platform installed (or the NVIDIA platform is not installed). Try to select the correct platform. You can do this by increasing the number of entries in the clGetPlatformIDs function, see https://www.khronos.org/registry/cl/sdk/1.1/docs/man/xhtml/clGetPlatformIDs.html

Reply

Anonymous
February 18, 2014 at 06:17
Okie i will tell u what exactly my configuration is and what i have observed then probably u might help me out... My CPU is intel core processor -i7 and i have an inbuilt gpu of nvidia. Now i have installed both intel sdk for opencl and nvidia propeitary drivers for ubuntu13.04. How can i create a run time so that it identifies both CPU and GPU. Currently i feel that only intel platform is getting recognised ... hence opencl is working fine for option CPU and not GPU. Is there a way around where both my devices are identified and probably i can transfer data between my CPU and GPU. Also in my vendor directory i can observe both intelocl64.icd and nvidia.icd.

Reply

Erik Smistad
February 18, 2014 at 13:47
An OpenCL context can only be associated with ONE

platform.You have TWO platforms installed and the code above only selects ONE platform, whichever one comes first, in your case the intel platform. To select the NVIDIA platform you need to increase the number of entries:

```
cl_platform_id
platform_ids[2];
clGetPlatformIDs(2,
platform_ids,
&ret_num_platforms);
```

and then select the other platform like this:

```
clGetDeviceIDs(
platform_ids[1],
CL_DEVICE_TYPE_DEFAULT,
1, &device_id,
&ret_num_devices);
```

Reply

meenu
February 19, 2014 at 09:46
Thanks Eric !!! This was the solution... Thanks again...

Reply

Gustavo Rozolin da Silva
October 12, 2013 at 21:25
Excellent post Erik,

Erik what I need to change in this code for to pass _ _local arg to kernel.

Thanks you.

Reply

Erik Smistad
October 14, 2013 at 16:23
Local memory, or shared memory as it is also called, is not accessible from the host. So if you want to use it you have to transfer data from global to local explicitly in a kernel.

Reply

Laxator2 October 12, 2013 at 19:19
Here is how it worked on my machine, using a (rather old) Nvidia card under PCLinuxOS :

gcc main.c -I/usr/local/cuda-5.0/include/ -L/usr/lib/nvidia-current -I OpenCL -o vectorAddition

Great example, short and to the point.

Reply

Kareti July 31, 2013 at 06:37
Hi Erik, The blog was very helpful. Thanks for that.

I have no GPU on my laptop, so is there a way to practise the opencl programs by emulating a GPU! I am using Fedora 18, 64 bit !

Thank you.

Reply

Erik Smistad July 31, 2013 at 11:11
Yes, you can use your CPU instead. That's the nice thing about OpenCL: The code can run on different types of processors. To do so simply install the Intel or AMD OpenCL runtime, depending on which type of processor you have. Afterwards execute the example above and it should run on the CPU.

Reply

Victor July 18, 2013 at 11:58
Hi Erik. Could u tell me what does this mean?
gcc -c -I /usr/include/CL main.c -o main.o
gcc -o local16 main.o -L /usr/lib -l OpenCL
/usr/lib64/gcc/x86_64-suse-linux/4.7/../../../x86_64-suse-linux/bin/ld: skipping incompatible /usr/lib/libOpenCL.so when searching for -lOpenCL
/usr/lib64/gcc/x86_64-suse-linux/4.7/../../../x86_64-suse-linux/bin/ld: skipping incompatible /usr/lib/libc.so when searching for -lc

I dont know if I successfuly link to the lib. What's weird is that the result keeps the same even if I comment the whole kernel code.

Reply

Erik Smistad July 31, 2013 at 11:13
This is a problem with the linking, not the source code. Not sure what the problem is, never seen that error message before

Reply

Xocoatzin June 6, 2013 at 20:42
Wow, it just worked. Awesome!

Reply

Anonymous ⊙ May 12, 2013 at 18:04
Very nice article, thanks!

I'm wondering whether the statements:

ret = clFlush(command_queue);
ret = clFinish(command_queue);

are actually needed. If I'm getting it right, since the command queue is in-order, when clEnqueueReadBuffer (which is blocking thanks to the CL_TRUE parameter) returns, the command queue should be empty.

Another point that would be worth explaining is why there is no clFinish between clEnqueueNDRangeKernel and clEnqueueReadBuffer.

Reply

Erik Smistad ⊙ May 13, 2013 at 10:44
Since the program terminates right after all the clean up statements, none of them are actually needed.

In either case, you are correct, the flush and finish statements are not necessary.

When the command queue is in-order, OpenCL will make sure that all the enqueue commands are performed in the order in which they are called in your C program.

Reply

Rohit Vashistha ⊙ April 25, 2013 at 18:51
Hi
All those getting 'zero' result change the option "GPU" to "CPU" or vise versa
Regards,
Rohit

Reply

Jai ⊙ April 22, 2013 at 15:03
I have a desktop pc with configuration as:
Intel(R) Core(TM) 2 Duo CPU E700@2.93GHz, 2GB RAM 32 bit OS

and I am willing to purchase a graphic card with config as:
Sapphire AMD/ATI RADEON HD 6450 2GB RAM

Can you please tell me is it compatible for my pc...?
Thanks in advance.

Reply

Jack ⊙ April 8, 2013 at 13:39
Can i run openCL on a CPU? (I do not have a GPU but want to experiment with openCL before buying one).
I have an intel i3 processor with a dual boot for Windows 7 and Ubuntu 12.04

Reply

Erik Smistad ⊙ April 8, 2013 at 13:56
Sure. Just install the Intel OpenCL SDK from http://software.intel.com/en-us/vcsource/tools/opencl-sdk

Reply

Anonymous ⊙ February 22, 2013 at 02:41
The sample code all worked fine.
But when I changed
CL_DEVICE_TYPE_DEFAULT into
CL_DEVICE_TYPE_GPU, it runs, but give me:
0 + 1024 = 763461144
1 + 1023 = 32716
2 + 1022 = 763461144
3 + 1021 = 32716
4 + 1020 = 15489024
5 + 1019 = 0
6 + 1018 = 15489024
7 + 1017 = 0
8 + 1016 = 0
9 + 1015 = 0
10 + 1014 = 0
11 + 1013 = 0
12 + 1012 = 0
13 + 1011 = 0
14 + 1010 = 0
15 + 1009 = 0
16 + 1008 = 0
17 + 1007 = 0
18 + 1006 = 0
19 + 1005 = 0
20 + 1004 = 0
21 + 1003 = 0
22 + 1002 = 0
23 + 1001 = 0
24 + 1000 = 0
25 + 999 = 0
26 + 998 = 124817
27 + 997 = 0
28 + 996 = 1801415779
29 + 995 = 1717531240
30 + 994 = 540292720
31 + 993 = 1633643619
32 + 992 = 1717527661
33 + 991 = 540292720
34 + 990 = 1801415779
35 + 989 = 1734308456
36 + 988 = 1633841004
37 + 987 = 1852399468
38 + 986 = 1597125492
39 + 985 = 1702060386
40 + 984 = 1869898079
41 + 983 = 1935894893
42 + 982 = 1600938784
43 + 981 = 1601333355
44 + 980 = 1651469415
45 + 979 = 1767861345
46 + 978 = 842232942
47 + 977 = 1954047327
48 + 976 = 1701080677
49 + 975 = 1952538468

50 + 974 = 1667853679
....
i tried CL_DEVICE_TYPE_CPU, and it worked fine.

why is GPU not working?

Reply

safwan December 29, 2012 at 21:32
Thank you for this briefly example and its work with me but I have a problem when I change the value of LIST_SIZE to another value, the program execute but she doesn't execute the kernel fonction an finaly the result i have for C[i]=0 :
0+1024=0
1+1023=0
...
..
how can I resolve this problem?
Thanks

Reply

Erik Smistad
January 2, 2013 at 14:53
If you change the LIST_SIZE variable to another variable that is not dividable with 64 it will not run because local size is set to 64 (see line 83). This means that 64 work-items are grouped together.

If you only get 0, you probably haven't installed OpenCL correctly.

Reply

Yaknan December 28, 2012 at 06:30
Hi Erik,
I am enjoying your tutorials on OpenCl. Thanks for the good work. Please, am testing a hypothesis here for my thesis work on enhancing graphic rendering of the ray tracing algorithm. Am wondering if it is possible to integrate cilk++ with openCL. The idea is to see if cilk++ will take full and better advantage of CPUs while OpnCL takes advantage of the GPUs more efficiently. Thanks!

Reply

Erik Smistad
January 2, 2013 at 14:50
As far as I know cilk++ can run regular C/C++ code as well. And OpenCL is written in C, so I think it should work..

Reply

prince November 14, 2012 at 12:35
i am using gpu of type nvidia,i am using opencl but when i run the program using ctrl+F5(start without debugging)then i get result in which gpu takes more time than cpu but when i run the program cpu takes more time than gpu and result is also i am giving

start without debugging -> cpu time=6127
ms gpu time= 6240 ms
start with debug-> cpu time= 18354 ms gpu
time= 9125 ms

wt is the reason in this difference.....
visual studio 2010 i am using
the code is here. wt is going wrong.?..thanks

// Hello.cpp : Defines the entry point for the
console application.
//

//#include
#include
#include
#include
#include
#include "CL/cl.h"
#define DATA_SIZE 100000
const char *KernelSource =
"kernel void hello(global float *input , global
float *output)\n"\
"{\n"\
" size_t id =get_global_id(0);\n"\
"output[id] =input[id]*input[id];\n"\
"} "
"\n"\
"\n";
//float start_time,end_time;

int main(void)
{
double start_time,end_time;
start_time=clock();
cl_context context;
cl_context_properties properties[3];
cl_kernel kernel;
cl_command_queue command_queue;
cl_program program;
cl_int err;
cl_uint num_of_platforms=0;
cl_platform_id platform_id;
cl_device_id device_id;
cl_uint num_of_devices=0;
cl_mem input,output;
size_t global;
float inputData[100000];
for(int j=0;j<100000;j++)
{
inputData[j]=(float)j;
}

float results[DATA_SIZE];//={0};

// int i;

//retrieve a list of platform variable
if(clGetPlatformIDs(1,&platform_id,&num_of_platforms)!=CL_SUCCESS)
{
printf("Unable to get platform_id\n");
return 1;
}

//try to get supported GPU DEvice
if(clGetDeviceIDs(platform_id,CL_DEVICE_TYPE_CPU,1,&device_id,

```
&num_of_devices)!=CL_SUCCESS)
{
printf("unable to get device_id\n");
return 1;
}

//context properties list -must be terminated
with 0
properties[0]=CL_CONTEXT_PLATFORM;
properties[1]=(cl_context_properties)
platform_id;
properties[2]=0;

//create a context with the GPU device
context=clCreateContext(properties,1,&
device_id,NULL,NULL,&err);

//create command queue using the context
and device
command_queue=clCreateCommandQueue(context,device_id,0,&
err);

//create a program from the kernel source
code
program=clCreateProgramWithSource(context,1,
(const char**)
&KernelSource,NULL,&err);

//compile the program
err=clBuildProgram(program,0,NULL,NULL,NULL,NULL);
if((err!=CL_SUCCESS))
{
printf("build error \n",err);
size_t len;
char buffer[4096];
//get the build log
clGetProgramBuildInfo(program,device_id,CL_PROGRAM_BUILD_LOG,sizeof(buffer),buffer,&len);
printf("—-build Log—\n%s\n",buffer);
exit(1);

// return 1;
}

//specify which kernel from the program to
execute
kernel=clCreateKernel(program,"hello",&err);

//create buffers for the input and output
input=clCreateBuffer(context,CL_MEM_READ_ONLY,sizeof(float)*DATA_SIZE,NULL,NULL);

output=clCreateBuffer(context,CL_MEM_WRITE_ONLY,sizeof(float)*DATA_SIZE,NULL,NULL);

//load data into the input buffer

clEnqueueWriteBuffer(command_queue,input,CL_TRUE,0,
sizeof(float)*DATA_SIZE,inputData,0,NULL,NULL);

//set the argument list for the kernel
command
clSetKernelArg(kernel,0,sizeof(cl_mem),&input);
clSetKernelArg(kernel,1,sizeof(cl_mem),&output);
global=DATA_SIZE;

//enqueue the kernel command for execution
clEnqueueNDRangeKernel(command_queue,kernel,1,NULL,&global,NULL,0,NULL,NULL);
clFinish(command_queue);

//copy the results from out of the buffer
clEnqueueReadBuffer(command_queue,output,CL_TRUE,0,sizeof(float)*DATA_SIZE,results,0,
NULL,NULL);
```

```
//print the results
printf("output:");
for(int i=0;i<DATA_SIZE;i++)
{
printf("%f\n",results[i]);
//printf("no. of times loop run %d\n",count);
}

//cleanup-release OpenCL resources

clReleaseMemObject(input);
clReleaseMemObject(output);
clReleaseProgram(program);
clReleaseKernel(kernel);
clReleaseCommandQueue(command_queue);
clReleaseContext(context);
end_time=clock();
printf("execution time is%f",end_time-
start_time);
_getch();
return 0;

}
```

Reply

Erik Smistad
November 15, 2012 at 16:05
It is normal for the execution time to increase when debugging an application. It doesn't mean that anything is wrong with the program

Reply

swap August 7, 2012 at 22:26
How will i execute same program on windows 7 with intel HD 4000 GPU.
I have installed Intel opencl SDK

Reply

Erik Smistad
August 10, 2012 at 14:23
Just remember to link to the lib files included in the Intel OpenCL SDK and add the include folder. If you are using Visual Studio you can add these things in the project settings menu.

Reply

swap August 7, 2012 at 22:24
how will i execute same program on windows 7 with intel HD 4000.
I have downloaded Intel Opencl SDK.

Reply

Lennart July 10, 2012 at 01:03
Thanks!

You officially got me started with OpenCL

(Hilsen fra Bergen)

Reply

Ricardas July 4, 2012 at 17:56
Great Tutorial. Thanks ! 🙂

Reply

Shinchan June 3, 2012 at 05:26
Hi,
I followed your steps and was able to get main.o.
But when i did
gcc main.o -o host -L /home/mydir /Downloads/ati-stream-sdk-v2.1-lnx64/lib /x86_64 -l OpenCL

I got
/usr/bin/ld: cannot find -lopenCL
clooect2: id returned 1 eit status

I have no idea what this means. Please help!

Reply

Erik Smistad June 4, 2012 at 11:35
It means that the linker can't find the libOpenCL file which should lie in your /home/mydir/Downloads/ati-stream-sdk-v2.1-lnx64/lib/x86_64 folder. Make sure you are using a large O in "-lOpenCL" and not "-lopenCL" as it says in your error message: "/usr/bin/ld: cannot find -lopenCL"

Reply

spandei April 4, 2012 at 04:05
Thank you!
You write very understandable code:)
To tell the truth your article helped me to understand the OpenCL Mechanism better than the whole AMD SKD. Keep it up!

Reply

Bishoy Mikhael March 7, 2012 at 23:35
i failed to compile the example,can you please review my configuration for the IDE, i've tried MS Visual Studio 2010, NetBeans 7.1 and Eclipse Indigo using both AMD and NVIDIA SDKs on Windows 7 x64 with Nvidia GeForce 330M graphics card.
i've declared the environment variables for CUDA SDK as follows $(CUDA_LIB_PATH), $(CUDA_BIN_PATH) and $(CUDA_INC_PATH) for (CUDA installation path\lib\x64), (CUDA installation path\bin), (CUDA installation path\include) respectively. in NetBeans TOOLS–>Options in C/C++ Code Assistance tab i've added the include directory for CUDA SDK, then in the project properties in "C Compiler" tab i've added the include directory path in "Include Directory", and in the "Linker" tab i've added the library path in "Additional Library Directories" then "opencl.lib" in "Additional Dependencies", i din't know what to add in the "Compilation Line" or if there is another settings i'm missing.

when i build the project i get an error:

"/bin/sh: -c: line 0: syntax error near unexpected token `('

/bin/sh: -c: line 0: `gcc.exe -m64 -c -I(CUDA_INC_PATH\) -MMD -MP -MF build/Release/MinGW-Windows/_ext/141055651/vector_add.o.d -o build/Release/MinGW-Windows/_ext/141055651/vector_add.o /C/Users/Arch/Documents/NetBeansProjects/vector_add/vector_add.c'
make[2]: *** [build/Release/MinGW-Windows/_ext/141055651/vector_add.o] Error 2"

Reply

Erik Smistad
March 12, 2012 at 10:44
Always a nightmare to compile on windows... but it should work on Visual Studio 2010 and setting include and library settings should be enough. Don't know what's wrong in your case

Reply

Bishoy Mikhael
March 13, 2012 at 11:20
i've uninstalled VS 2010, netbeans, MinGW, cygwin and the newly installed Microsoft Visual C++ 2008 Redistributables and .NET frameworks, then installed Code::Blocks then i copied the CL directory from the include directory of NVIDIA Toolkit and set the additional libraries in Code::Blocks to point at the OpenCL.lib, guess what, it worked fine without any errors

Reply

Hao Wang March 1, 2012 at 18:39
Hi,

I'm trying to run OpenCL program on a simulator (gem5).

The simulator supports Full-System mode, that is, first boot Linux from a disk-image and then run the program.

I borrowed the ICD record and libOpenCL.so from AMD SDK, and put them into the proper place in the image file.

But the simulation trace shows that, it fails to find a platform, and then crashes when trying to create a context.

Do you have any suggestions on my situation>
Thank you.

Reply

Erik Smistad March 5, 2012 at 10:14

Hi

Make sure the ICD files can be read by the program and that the AMD APP SDK and display drivers are properly installed.

Reply

rupam ⊙ February 29, 2012 at 08:00
hi, I an pretty new in GPU.Currently I am working on AMD RADEON 6500 series.... I have written some program on MATLAB 2011b...I want to run the codes on GPU...can u plz instruct me how to run .m codes on GPU...thanx in advance...

Reply

vegihat ⊙ February 27, 2012 at 13:15
Hello Erik,

i try to understand what is the physical meaning of below clEnqueueNDRangeKernel's arguments

const size_t *global_work_size
const size_t *local_work_size

you used values
size_t global_item_size = LIST_SIZE;
size_t local_item_size = 64

which means that we have LIST_SIZE/64 work-groups,right?

what's the difference between local_item_size=1 and local_item_size = 64?

i want to understand which is the perfect value of the local_item_size .

Reply

Erik Smistad
⊙ February 27, 2012 at 14:46
The "perfect" value of local_item_size is very system and application dependent. You can omit this parameter and let OpenCL decide by itself. Note that for NVIDIA GPUs the local_item_size should be a multiple of 32 (one warp) or else some work-items will be idle. The same applies to AMD GPUs, but with a multiple of 64 instead (one wavefront as they call it). This is because NVIDIA and AMD schedules 32 and 64 work-items atomically on each compute unit.

Reply

The Ham ⊙ February 25, 2012 at 18:53
Hi, to all people that get wrong output (garbage or all 0)
I used this code under gForce M8400 GS and get garbage and error -52 in clEnqueueNDRangeKernel which is wrong kernel arguments.

that is what i changed in this code:
add 4-th argument for the kernel

```
ret = clSetKernelArg(kernel, 3,
sizeof(int), &LIST_SIZE);
```

remember that in this code LIST_SIZE must be 64 * m (m is integer)

This code works ok on my AMD HD 6670 without any changes,
dont know why (just started with OpenCL)

(cant rly add comment on this site!!)

Reply

The Ham ⊙ February 25, 2012 at 18:52
Hi, to all people that get wrong output (garbage or all 0)
I used this code under gForce M8400 GS and get garbage and error -52 in clEnqueueNDRangeKernel which is wrong kernel arguments.

that is what i changed in this code:
add 4-th argument for the kernel

```
ret = clSetKernelArg(kernel, 3,
sizeof(int), &LIST_SIZE);
```

remember that in this code LIST_SIZE must be 64 * m (m is integer)

This code works ok on my AMD HD 6670 without any changes,
dont know why (just started with OpenCL)

Reply

The Ham ⊙ February 25, 2012 at 18:59
i just can get it right...

another thing in the kernel
change output buffer type
`__global int *C` to `__global float *C`
`or you get all = 0`

Reply

LaKing ⊙ February 22, 2012 at 16:26
Hello. ..

I compiled some sample applications, and get this error when running any OpenCL application. ...

OpenCL SW Info:

Error -1001 in clGetPlatformIDs Call !!!

I was googling for several hours, got some useful info's but could not solve the problem yet. Any ideas? ... Thanks.

Reply

Erik Smistad

February 23, 2012 at 13:46
Hi

I think it means that it can't find any OpenCL platforms. Check to see if the .icd files are properly installed. They should exist under /etc/OpenCL /vendors. If no .icd files exists there you have to find them in the downloaded SDK and extract them manually to this location.

Reply

Otto February 15, 2012 at 19:53
The OpenCL library seems to slow initial execution down a bit. The example above does not really work the GPU. I created the same thing in pure C and it's almost 3 times faster. 😐

time (./cpu >dump)

real 0m0.320s
user 0m0.292s
sys 0m0.024s

time (./gpu >dump)

real 0m0.825s
user 0m0.736s
sys 0m0.100s

Reply

Erik Smistad
February 16, 2012 at 15:48
There is always some overhead with using the GPU such as transferring data back and forth and setup time compared to the CPU. But if you try the same example with a VERY large list/vector you will definitely see a speedup. So if your data parallel problem is very small, using the GPU is usually not faster, but large problems such as volume processing where you have several million elements to be processed you get an enormous speedup.

Reply

Otto February 15, 2012 at 19:25
Create a Makefile
#—cut—
GCC=gcc
CFLAGS= -c -I/opt/AMDAPP/include
LDFLAGS= -L/opt/AMDAPP/lib/x86_64
-L/usr/lib -lOpenCL

all: host

host:
${GCC} ${CFLAGS} main.c -o main.o
${GCC} main.o -o host ${LDFLAGS}

clean:
rm -rf host main.o
#—cut—

I'm using Ubuntu 11.10 64bit + AMD-APP-SDK 2.6 + 11.12 drivers and got it working with the Makefile. In Eriks exmaple above, the linking params are in the wrong order, therefore it fails every time.

Reply

Erik Smistad
February 16, 2012 at 15:45
I can swear that my compilation command worked before, but when I tried it myself just now on the same system it failed. Maybe its a new version of gcc or something... anyway thanks for letting me know, I've updated the post.

Reply

Ajay February 7, 2012 at 10:48
Hey Erik,
Is there a way to write functions inside the opencl kernel?

Reply

Erik Smistad
February 7, 2012 at 13:15
I don't now why you would want to write a function inside a kernel... but you can create a function in OpenCL that is inside the same file as the kernels and then call that function in the kernel (see below). If this is not what you want you may have to use macros or something (like "#define MAX(a,b) a > b ? a : b").

```
int myFunction() {
...
}

__kernel void myKernel() {
myFunction();
```

Reply

José February 4, 2012 at 00:07

José:

I tried to install and run your example but the answer is that it can't find the file cl.h
What should I look for?
I tried to reinstall but it simply don't want to run.
The icd files and the .h are where they are supposed to be.
The instalation of the openCL is on the /opt/AMDAPP/
Also, running on a Phenom II 1090T + HD6850

Forgot this information:

The comand lines that I used:

main.c @ ~/openCL-teste/

comand line used:
gcc -I /opt/AMDAPP/include/CL/ main.c -o main.o

Reply

Erik Smistad
February 7, 2012 at 13:11
It is because the include line is "#include ", but your include path is inside the CL directory. Try this instead: "gcc -I /opt/AMDAPP /include/ main.c -o main.o"

Reply

José February 4, 2012 at 00:03
I tried to install and run your example but the answer is that it can't find the file cl.h

What should I look for?

I tried to reinstall but it simply don't want to run.

The icd files and the .h are where they are supposed to be.

The instalation of the openCL is on the /opt/AMDAPP/

Also, running on a Phenom II 1090T + HD6850

Reply

Jesse January 31, 2012 at 20:25
This is a good guide, so much easier on linux than windows

Reply

Anjil January 16, 2012 at 17:43
I tried with CPP vesrion of your example and it simply displays: clGetPlatformIDs(-1001)

Does this mean the OpenCL is not installed properly? Should I re install the NVIDIA driver?

Reply

Erik Smistad
January 16, 2012 at 21:32
Yes, it looks like OpenCL is not properly installed. Try reinstalling and check that the .icd files are present in the /etc/OpenCL/vendors folder. Download the development driver and CUDA toolkit from here: http://developer.nvidia.com/cuda-toolkit-40 and follow the install instructions for linux

Reply

Anjil January 13, 2012 at 22:00
The program compiles fine but the results are not as expected:

989 + 35 = 0
990 + 34 = 0
991 + 33 = 0
992 + 32 = 0
993 + 31 = 0
994 + 30 = 0
995 + 29 = 0
996 + 28 = 0
997 + 27 = 0
998 + 26 = 0
999 + 25 = 0
1000 + 24 = 0
1001 + 23 = 0
1002 + 22 = 0
1003 + 21 = 0
1004 + 20 = 0
1005 + 19 = 0
1006 + 18 = 0
1007 + 17 = 0
1008 + 16 = 0
1009 + 15 = 0
1010 + 14 = 0
1011 + 13 = 0
1012 + 12 = 0
1013 + 11 = 0
1014 + 10 = 0
1015 + 9 = 0
1016 + 8 = 0
1017 + 7 = 0
1018 + 6 = 0
1019 + 5 = 0
1020 + 4 = 0
1021 + 3 = 0
1022 + 2 = 0
1023 + 1 = 0

Please let me know whats the issue here?

Reply

klm123 June 4, 2012 at 17:58
Why nobody answer? No one can explain it???!!

Reply

klm123
June 4, 2012 at 17:59
I've got same result and it does not depend on the content of vector_add_kernel.cl file.

Reply

Erik Smistad
June 10, 2012 at 09:15
Some of the function calls must be failing. Use the error argument to check which and check which error type it is. Most likely OpenCL is not properly

installed.

Reply

Pratik Anand
July 8, 2016 at 03:37
The platform selection call can be modified as per this URL http://stackoverflow.com/questions/30079550/opencl-clgetdeviceids-returns-1-when-gpu-and-0-when-cpu

In that case, the OpenCL starts detecting the GPU.

Reply

Paraita Wohler
July 11, 2012 at 13:57
You should double check if your NVidia card actually support OpenCL, depending on your card, some operations might not be implemented. Also check what clGetPlatformIDs gives you back (error -1001 ?), I had alot of trouble with this too, at the end it was the card I was using...

Reply

Paraita Wohler
July 11, 2012 at 14:00
in fact, even if your hardware doesn't support OpenCL, you can still run OpenCL code, but it won't work (returning error 1001 which is a "catch-all" error for almost everything related to hardware)

Reply

blogger December 16, 2011 at 05:20
Great tutorial! Thanks a lot, it helped me get started with opencl

Reply

Guilherme August 21, 2011 at 10:55
Great tutorial! Congrats dude.

I'm having just a little problem. I'm trying to use M$ Visual C++ 2010 Express: I created a new empty project, included the path to the necessary files (AMD SDK – wich are located

on C:\Program Files (x86)\AMD APP\include) on the “include settings” of the project, created two files on this project with the names test.cpp and vector_add_kernel.cl with the exact same codes on this tutorial. I get a lot of errors:
(first line in the output)- 1>classe.obj : error LNK2019: unresolved external symbol _clReleaseContext@4 referenced in function _main

Am I doing something wrong? Should I use another IDE? I’m really newbie on OpenCL (and a little bit in C++). And sorry for my english ‘cause I’m brazilian.

Reply

Erik Smistad
August 21, 2011 at 13:23
You have forgot to add additional library paths in Visual Studio. You need to go to project settings, c++ linker, and then add C:\Program Files (x86)\AMD APP\lib/x86 to additional libraries, and on input write OpenCL.lib...

I think I will add some notes on how to set up OpenCL in visual studio in this post in a few days

Reply

Guilherme
August 22, 2011 at 03:02
It worked! Thanks. =]

Reply

Ben August 13, 2011 at 23:19
Yours is probably the clearest and most concise “getting started” guide I’ve seen, so many thanks for this.

Incidentally, both your C and C++ example programs also work perfectly on Windows using AMD’s OpenCL implementation with MinGW-w64.

Reply

rwi July 13, 2011 at 10:55
Hi Erik,
your guide helped me getting started. I’m using the nVIDIA CUDA Toolkit 4 on Ubuntu 10.10. I had trouble compiling with gcc and ended up with:

gcc -lOpenCL -I /usr/lib64/ -I /usr/local /cuda/include main.c -o host

where again lOpenCL starts with a lowercase L and the two locations are prefixed with -(capital i) (just in case). This worked for me.

Thanks a lot!

Reply

Jeffin March 21, 2012 at 15:26

Hi rwi,
Thanks a lot for your post.The command options that you provided was very helpful for my build issues while trying with Netbeans IDE.
Thanks again
Jf

Reply

none June 30, 2011 at 08:03
I feel with you and did not read your desperate endeavours (trying to help the ppl here) till the end. But one thing i had to fight with a year ago was that the nvidia drivers that came with ubuntu (i mean the propietary ones) did not place two links right. One of them was the opencl... (as above). One solution was to install the driver directly from nvidia. The other one is to search the net and the answer will come. It's just two softlinks.

Reply

Erik Smistad July 2, 2011 at 16:00
Thanks for the tip

Reply

Dan June 9, 2011 at 01:07
Can we run OpenCl without installing one of the SDKs? if so, what is the best way to do it?

Reply

Erik Smistad June 9, 2011 at 10:26
You need an OpenCL implementation installed to use it. If you have an intel CPU and want to run OpenCL on it check out their SDK at http://software.intel.com/en-us/articles/opencl-sdk/

Reply

michael May 27, 2011 at 05:21
Here is my result, it's not correct. why?
0 + 1000 = 0
1 + 999 = 0
2 + 998 = 0
3 + 997 = 0
4 + 996 = 0
5 + 995 = 0
6 + 994 = 0
7 + 993 = 0
8 + 992 = 0
9 + 991 = 0
10 + 990 = 0
11 + 989 = 0
12 + 988 = 0

Reply

Anjil January 13, 2012 at 21:58
I am getting the same response, all results as zero's any idea how to fix this?

Reply

Harshit ⊙ May 26, 2011 at 16:33

Hi Erik,
I'm a noob..
I get this error when i Compile the program using
Documents/cuda/Assignment/Ex. 2$ gcc -c -Wall -I /usr/local/cuda/include cpu.c -o main

cpu.c:8: warning: return type defaults to 'int'
cpu.c: In function 'main':
cpu.c:52: warning: control reaches end of non-void function

But I'm getting the right result.. 🙂

Is something wrong??

Reply

Erik Smistad ⊙ May 26, 2011 at 18:03

They are only warnings... and it seems you only need to add a "return 0;" in the end of your main method to get rid of it.

Reply

Ajay ⊙ May 26, 2011 at 10:51

I changed it. But I want to know that whether the program is running on the GPU or not.. Will I get error message if the program is not running on GPU or will it run on CPU if it doesn't find the GPU after changing CL_DEVICE_TYPE_DEFAULT to CL_DEVICE_TYPE_GPU.

I'm asking this coz I wanted to measure performance of an OpenCL program by using clGetEventProfilingInfo. I have done everything right but I'm getting timings only if i set CL_DEVICE_TYPE_CPU. If i set it to CL_DEVICE_TYPE_DEFAULT or CL_DEVICE_TYPE_GPU, then I'm getting answer as 0.00.
Why am I getting timings only on CPU?? Is the program running on CPU even after setting CL_DEVICE_TYPE_GPU?? Thats my doubt..

Reply

Erik Smistad ⊙ May 26, 2011 at 18:02

There is no error checking in the example above, because I wanted the example to be as simple as possible. So it can happen that it doesn't run at all. You can print out ret_num_devices and see if it is larger than 0.

I recommend the free OpenCL/OpenGL profiler tool gDEBugger – http://www.gremedy.com/

With this program you should be able to see exactly where your code is run and how fast.

Reply

Ajay ⊙ May 25, 2011 at 18:30
thanks Erik,
I have another doubt..
How can i find whether the code is actually running on GPU or not??

Reply

Erik Smistad ⊙ May 25, 2011 at 21:40
Change CL_DEVICE_TYPE_DEFAULT to CL_DEVICE_TYPE_GPU on line 43 and it should only run on GPUs.

Reply

Ajay ⊙ May 18, 2011 at 10:11
I am new to OpenCL programming. I successfully compiled the sample program. then I tried './host' to run it. but I'm getting an error..
./host: error while loading shared libraries: libOpenCL.so.1: cannot open shared object file: No such file or directory

Reply

Erik Smistad ⊙ May 18, 2011 at 10:35
That error means that you have forgot to set the LD_LIBRARY_PATH, see the description in the article above.

Reply

Kurt ⊙ May 16, 2011 at 14:40
Hi Erik – Thanks for your blog. Has given me good help.
Now on a slightly different topic: ATI plus NVidia plus CPU.
The reason I want to use OpenCL is so my software can work on any of these parallel platforms – I might even create a driver for high-end DSPs after I learn a bit more.

However, it seems difficult to find platform-neutral info on how to get set up. Is the platform (NV vs ATI) entirely based on what LD_LIBRARY_PATH points to? If so, is it impossible to use –device cpu in a machine equipped with NVidia GPU?

IBM has what I think is intended to be a solution called OpenCLCommonRuntime (aplhaworks), but I can't seem to get it to work with AMD –device cpu. Do you have any experience with this? IBM is, at least, a neutral source.

Reply

Erik Smistad ⊙ May 18, 2011 at 10:34
Hi Kurt

My personal experience with NVidia's and ATI/AMD's OpenCL implementation is that NVidia still only has a buggy version of OpenCL 1.0. Their implementation has some serious issues when using images.

ATI/AMD's implementation on the other hand has worked great and is updated to 1.1.

The thing about OpenCL is that it is to some degree portable, (it is portable if you don't take into account all of NVidia's bugs), but it is not necessary performance portable. Some OpenCL code might be fast on an ATI card while it may not be on an NVidia card. Because of their different architectures. This might and hopefully will be less true when their OpenCL compilers have matured.

It is possible to use the CPU with OpenCL on a machine with a NVidia GPU.

I haven't tried OpenCLCommonRuntime, but it looks quite promising

Reply

Krish ⊙ April 6, 2011 at 08:04
hi erik
it worked .....

thanx

Reply

Krish ⊙ March 30, 2011 at 07:04
Hi erik
(for the post no:28)
even after specifying cuda installation path
[root@ghost OpenCL]# gcc -I /usr/local /cuda/include/CL/ -lOpenCL hellocl.c
In file included from hellocl.c:12:0:
cl.h:32:28: fatal error: CL/cl_platform.h: No such file or directory compilation terminated.

i dont understand what is happening..... how to correct the errror.....

Reply

Erik Smistad
⊙ March 30, 2011 at 23:39
It is supposed to be -I (capitol i) and not -l in front of the include path

Reply

paolo ⊙ March 29, 2011 at 18:43
Hi Erik,
thanks for posting these info

I tried to compile your example on a Mac OSX 10.6 with Envidia GEFORCE 9400M.

I have the drivers installed correctly (I believe), and so it the opencl toolkit.
When I call the linking part of it, I get the error on "host"

it says: "host: no such file or directory"
what is host exactly?
thank you

Reply

Erik Smistad
March 31, 2011 at 14:31
Host is just the name of the executable you are creating when compiling. The name of the executable is not important. There most be something wrong with your compile command. Paste your compile command here

Reply

Krish March 28, 2011 at 12:17
hi
i am just started learning open cl
i have installed nvidia sdk etc... as given on Nvidia website...
i am using Fedora 14 – 64 bit edition
when i am compiling
as
$ gcc hello.c -lopenCL

it gives me
cl.h:32:28: fatal error: CL/cl_platform.h: No such file or directory; compilation terminated.

pls need help from u to rectify it ....

Reply

Erik Smistad
March 28, 2011 at 14:43
Have you remembered to specify the include path -I to the include folder of your CUDA installation?

Reply

Freddan October 23, 2010 at 18:50
main.c:(.text+0x129): undefined reference to `clGetPlatformIDs'
main.c:(.text+0x150): undefined reference to `clGetDeviceIDs'
main.c:(.text+0x17b): undefined reference to `clCreateContext'
main.c:(.text+0x19e): undefined reference to `clCreateCommandQueue'
main.c:(.text+0x1cc): undefined reference to `clCreateBuffer'
main.c:(.text+0x1fa): undefined reference to `clCreateBuffer'
main.c:(.text+0x228): undefined reference to `clCreateBuffer'
main.c:(.text+0x27c): undefined reference to `clEnqueueWriteBuffer'
main.c:(.text+0x2cc): undefined reference to `clEnqueueWriteBuffer'
main.c:(.text+0x2ef): undefined reference to `clCreateProgramWithSource'
main.c:(.text+0x31f): undefined reference to `clBuildProgram'
main.c:(.text+0x33a): undefined reference to `clCreateKernel'
main.c:(.text+0x361): undefined reference to `clSetKernelArg'

main.c:(.text+0x384): undefined reference to `clSetKernelArg'
main.c:(.text+0x3aa): undefined reference to `clSetKernelArg'
main.c:(.text+0x411): undefined reference to `clEnqueueNDRangeKernel'
main.c:(.text+0x47f): undefined reference to `clEnqueueReadBuffer'
main.c:(.text+0x4e7): undefined reference to `clFlush'
main.c:(.text+0x4f6): undefined reference to `clFinish'
main.c:(.text+0x508): undefined reference to `clReleaseKernel'
main.c:(.text+0x51a): undefined reference to `clReleaseProgram'
main.c:(.text+0x529): undefined reference to `clReleaseMemObject'
main.c:(.text+0x538): undefined reference to `clReleaseMemObject'
main.c:(.text+0x54a): undefined reference to `clReleaseMemObject'
main.c:(.text+0x559): undefined reference to `clReleaseCommandQueue'
main.c:(.text+0x568): undefined reference to `clReleaseContext'

Reply

Erik Smistad
October 27, 2010 at 16:57
You need to link the OpenCL library when you compile: (example: gcc -L /path-to-the-lib-folder-with-OpenCL-libfile/ -l OpenCL main.o -o host)

Reply

Rohan
February 23, 2011 at 09:17
I too got the same errors and it actually happens during the first command not the second. I even created the symlinks as stated above by a user, but nothing works 😕

Reply

Erik Smistad
February 23, 2011 at 14:12
What OS and OpenCL implementation are you using? And what is the exact commands you are running? Because this works perfectly on my system...

Reply

snk
February 23, 2011

at 23:26
Hello,
I got these errors, but the solution was simple. Try the -c option for the first command. This option will prevent the gcc from linking, and then the linking can be done with the second command.
It worked for me. I hope it will work for you too.

Reply

Erik Smistad
February 24, 2011 at 17:26
Thanks for clearing that up snk. It seems that the -c flag was missing in the first example.

Reply

Lucas Campos
May 30, 2011 at 14:15
I had these same problems on linking. I'm using nVidia's SDK, on Mint 10 x64 and this solution did not work for me. But, when I compiled using just

gcc -I path-to-opencl-common-inc -lopenCL main.c -o test, worked like a charm. Using SDK 4.0 was

gcc -lOpenCL -I ~/NVIDIA_GPU_Computing_SDK

/OpenCL /common/inc/ main.c -o test

Important to notice that its lOpenCL (lower case L) and -I (upper case -i)

Reply

tom ⊙ September 4, 2010 at 02:25

0 + 1000 = 1561096336
1 + 999 = 32714
2 + 998 = 1561096336
3 + 997 = 32714
4 + 996 = 6311776
5 + 995 = 0
6 + 994 = 6311776
7 + 993 = 0
8 + 992 = 2147483647
9 + 991 = 0
10 + 990 = 12370275
11 + 989 = 0
12 + 988 = 788225073

Reply

tom ⊙ September 4, 2010 at 02:24

once everything compiles, running the program, it actually returns garbage !? the vectors are not added but garbage is displayed.

Reply

Erik Smistad

⊙ September 4, 2010 at 13:06

I have experienced this myself when trying to run this on a NVidia graphics card that doesn't support OpenCL.

You can check to see if your GPU supports OpenCL here: http://www.nvidia.com/object/cuda_gpus.html

Also make sure that you have the proper dev drivers. If you are using linux, check out "Developer Drivers for Linux (256.40)" at http://developer.nvidia.com/object/cuda_3_1_downloads.html

Reply

tom

⊙ September 5, 2010 at 08:55

my nvidia quadro fx3700 is cuda/opencl compatible

Reply

tom

⊙ September 5, 2010 at 08:56

and I installed the newest

drivers, 256.53. this is on opensuse 11.1 (64bit)

Reply

Erik Smistad
September 5, 2010 at 22:58
Hm.. should have worked...

You could try the C++ version at http://www.thebigblob.com/using-the-cpp-bindings-for-opencl/ which use exceptions and see if it catches some error. Maybe that could help you to find out what's failing

Reply

tom September 4, 2010 at 01:25
then I get: "undefined reference to `clGetPlatformIDs'" and similar errors all relating to the functions defined in cl_platform.h (which gcc finds, I checked). any hints appreciated.

Reply

tom September 4, 2010 at 02:23
symlinks need to be created. libOpenCL.so.1.0.0 comes from nvidia. libOpenCL.so and libOpenCL.so.1 are needed symlinks that need to be created (command 'ln')

Reply

Rohan
February 23, 2011 at 08:59
I am getting the same problem. Could please write the exact command that need to be run to create the symlinks?

Reply

tom September 3, 2010 at 23:57
also, the link above to download the source code leads to a 404 file not found.

Reply

Erik Smistad
September 4, 2010 at 12:59
Sorry about that. The link has been fixed now

Reply

tom
September 6, 2010 at 10:11
using your files exactly, still getting garbage output of the

vector addition.

Reply

Erik Smistad
September 6, 2010 at 13:22
Then I'm out of ideas.. it works fine on all of my ATI cards. I will try to run the code on some machines with nvidia cards at the university and see if I get the same problem there.

Reply

Name (required)
September 7, 2010 at 02:29
I added a printf() to your code after line 43:
if (ret != CL_SUCCESS)
{
printf("Error: Failed to query platforms! (%d)\n", ret);
return EXIT_FAILURE;
}

after compiling, running it gives me this error: "Failed to query platforms (-1001)"

Reply

tom
September 10, 2010 at 11:47
have you had a chance to try it out at your university?

it's still not working on my side ☹

Reply

Erik Smistad
⊙
September 10, 2010 at 12:27

Yes, I tested it just now on a geforce 8400 at the university. The installation was no problem.

First I downloaded the CUDA toolkit and installed that. Then I installed the CUDA devdriver, set LD_LIBRARY_PATH to /usr/local /cuda/lib64 and ran the example, no problem... I will add the procedure for nvidia to the post soon.

I don't understand

why it doesn't work for you.. you have a nvidia.icd file in your /etc/OpenCL /vendors folder, right?

Anjil
January 14, 2012 at 20:48

Hi Erik,

Is the garbage value issue posted by few developers above fixed? Is there any solution available. Please let me know, I am facing the same problem, Every time I execute the sample, it simply returns

either zero some garbage value which is not relevant. I am working with Ubuntu 11.04 with NVIDIA graphics driver installed.

Erik Smistad
January 15, 2012 at 15:01
There is no error checking in the example above, because I wanted the example to be as simple as possible. So it can happen that the kernel doesn't run at all, maybe because OpenCL is not properly

installed. Try using the C++ bindings in this post http://www.thebigblob.com/using-the-cpp-bindings-for-opencl/ as this example has error checking. See what error message you get, maybe that can help you find out what the problem is. Post it here afterwards. I got this error myself once and found that it was because OpenCL wasn't properly installed, was lacking icd

files or something like that. Hope that helps!

tom September 3, 2010 at 23:52
gcc question: gcc can't find the cl.h, even though it's in the same directory as the source file, also it can't find it even after explicitly specifying the path!!!???

Reply

tom September 4, 2010 at 01:23
ok, the -I is not a lower case "l" it is an uppercase "i": I

Reply

Achtkraut August 28, 2010 at 09:18
SSE2 support was only added in ATI Stream SDK v2.2

Reply

Jillian Marohnic July 21, 2010 at 19:47
Is it the case that the CPU must be sse3? I followed your blog but the OpenCL samples don't run successfully (have sse2).
Also, when you say to decompress the registration in the root folder, I assumed you meant the sdk root folder … might want to clarify (for dummies like me) that it is the file system root folder.
Thanks for the article … nicely written.

Reply

Erik Smistad July 21, 2010 at 20:24
I don't believe that the CPU must support SSE3. I don't see why it shouldn't support CPUs with SSE2. Can you compile the example and does it run without any errors? What exactly happens when you try to run the example?

I don't mean that you should decompress the SDK to the root folder of your system. What I mean is that you should extract the .icd files of the icd-registration.tgz file to the folder /etc/OpenCL/vendors. This is offcourse only if you are running on linux, I haven't tried setting OpenCL on windows just yet. You have to do this to make it work.

Reply

LEAVE A REPLY

Comment

Name Email

Website

Post Comment

Home
About
Research
FAST
Blog posts