Desenvolvendo uma aplicação com OpenCL

EP 4 – Entrega 12/12/2012

Computação Paralela e Distribuída – MAC5742/MAC0431
Segundo Semestre de 2012
Prof. Alfredo Goldman (gold at ime.usp.br)

1. Introdução

Essa atividade explora os conceitos e práticas de programação em OpenCL. OpenCL é um padrão para uma plataforma de programação para computação heterogênea, o que possibilita execução de código em GPU (AMD-ATI) e também a execução de código em CPU. No nosso caso essa característica será útil, pois dispensa a necessidade de termos uma GPU, iremos executar nosso código na CPU.

O padrão OpenCL é mantido pela kronos (www.khronos.org). Estão disponíveis neste site documentação sobre o padrão e a API.

Iremos utilizar a AMD-APP-SDK, como o SDK da plataforma dispensa o uso de GPU (não temos uma máquina com GPU) não será disponibilizado um ambiente preparado em uma máquina como na EP3. É necessária a instalação do SDK em seu computador.

2. Instruções de preparação do ambiente

1. Faça download e instale o SDK (AMD-APP-SDK-v2.7-lnx32):
(http://developer.amd.com/tools/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk/downloads/)
Siga as instruções de instalação do *http://developer.amd.com.php53-23.ord1-1.websitetestlink.com/wordpress/media/2012/10/AMD_APP_SDK_Installation_Notes.pdf*

2. Utilize a estrutura disponível na instalação do SDK, no meu ambiente de teste instalei o SDK no diretório /opt, ficando assim /opt/AMDAPP.

3. Acesse o diretório da sua instalação e digite `make` para que os exemplos possam ser compilados. Caso queira testar algum, os binários estarão disponíveis em /opt/AMDAPP/samples/opencl/bin/x86.

4. Copie o exemplo `VectorAdd` para o diretório de fontes das aplicações (/opt/AMDAPP/samples/opencl/cl/app/).
Acesse o diretório /opt/AMDAPP/samples/opencl/cl/app/ e edite o `Makefile` acionando na variável SUBDIRS o nome `VectorAdd` para que a estrutura possa também compilar a aplicação exemplo.

```
DEPTH = ../../../..

include $(DEPTH)/make/openclsdkdefs.mk

SUBDIRS = AESEncryptDecrypt\
        BinarySearch     \
        ...Outros nomes...
```

```
        GaussianNoiseGL \
        DeviceFission11Ext \
        VectorAdd

… Restante do conteúdo omimtido...
```

Não é necessário executar o comando `make` todas as vezes a partir do raiz da instalação, basta executá-lo dentro do diretório da aplicação, o executável será gerado dentro do diretório de binários (/opt/AMDAPP/samples/opencl/bin/x86). Um exemplo de saída é apresentado no Código 1.

Código 1: Exemplo de saída a aplicação VectorAdd

```
rogerio@guarani:/opt/AMDAPP/samples/opencl/cl/app/VectorAdd$ make
mkdir -p depends/x86
perl      ../../../../../make/fastdep.pl      -I.      -I../../../../../include
-I../../../../../samples/opencl/SDKUtil/include     --obj-suffix='.o'     --obj-
prefix='build/debug/x86//' VectorAdd.cpp > depends/x86/VectorAdd.depend
mkdir -p build/debug/x86/
Building build/debug/x86//VectorAdd.o
g++     -m32   -msse2   -Wpointer-arith        -Wfloat-equal   -g3   -ffor-scope
-I  ../../../../../samples/opencl/SDKUtil/include -I  "/opt/AMDAPP/include"  -I
../../../../../include  -o build/debug/x86//VectorAdd.o -c  VectorAdd.cpp
Building build/debug/x86/VectorAdd
g++  -o  build/debug/x86/VectorAdd  build/debug/x86//VectorAdd.o  -m32  -lpthread
-ldl  -L/usr/X11R6/lib      -lSDKUtil    -lOpenCL      -L../../../../../lib/x86
-L../../../../../TempSDKUtil/lib/x86 -L"/opt/AMDAPP/lib/x86"
install                       -D                       build/debug/x86/VectorAdd
../../../../../samples/opencl/bin/x86/VectorAdd
for f in VectorAdd_Kernel.cl; do \
            install -D $f ../../../../../samples/opencl/bin/x86/$f; \
      done
rogerio@guarani:/opt/AMDAPP/samples/opencl/cl/app/VectorAdd$
../../../bin/x86/VectorAdd
Inicializando matriz...
Inicializando matriz...
Imprimindo matriz...
0.840188  0.394383  0.783099  0.798440  0.911647  0.197551  0.335223  0.768230
0.277775  0.553970  0.477397  0.628871  0.364784  0.513401  0.952230  0.916195
0.635712  0.717297  0.141603  0.606969  0.016301  0.242887  0.137232  0.804177
0.156679  0.400944  0.129790  0.108809  0.998924  0.218257  0.512932  0.839112
0.612640  0.296032  0.637552  0.524287  0.493583  0.972775  0.292517  0.771358
0.526745  0.769914  0.400229  0.891529  0.283315  0.352458  0.807725  0.919026
0.069755  0.949327  0.525995  0.086056  0.192214  0.663227  0.890233  0.348893
0.064171 0.020023 0.457702 0.063096 0.238280 0.970634 0.902208 0.850920
Imprimindo matriz...
0.266666  0.539760  0.375207  0.760249  0.512535  0.667724  0.531606  0.039280
0.437638  0.931835  0.930810  0.720952  0.284293  0.738534  0.639979  0.354049
0.687861  0.165974  0.440105  0.880075  0.829201  0.330337  0.228968  0.893372
0.350360  0.686670  0.956468  0.588640  0.657304  0.858676  0.439560  0.923970
0.398437  0.814767  0.684219  0.910972  0.482491  0.215825  0.950252  0.920128
0.147660  0.881062  0.641081  0.431953  0.619596  0.281059  0.786002  0.307458
0.447034  0.226107  0.187533  0.276235  0.556444  0.416501  0.169607  0.906804
0.103171 0.126075 0.495444 0.760475 0.984752 0.935004 0.684445 0.383188
Criando o ambiente OpenCL...
Inicializa o contexto OpenCL.
Inicializa uma fila de comandos.
Alocação das estruturas na memória do dispositivo.
Faz a cópia dos arrays h_A e h_B para d_A e d_B.
 Abre o fonte do kernel.
Cria e compila o fonte do programa.
```

```
Compila o kernel.
Define o grid.
Define os parâmetros do kernel.
Lança a execução do kernel.
Copia o resultado d_C para h_C.
Finaliza a fila de comandos.
Imprimindo matriz...
1.106853  0.934143  1.158306  1.558689  1.424183  0.865275  0.866829  0.807510
0.715412  1.485805  1.408207  1.349823  0.649078  1.251935  1.592209  1.270244
1.323573  0.883271  0.581707  1.487044  0.845502  0.573224  0.366200  1.697549
0.507039  1.087614  1.086259  0.697449  1.656229  1.076933  0.952492  1.763082
1.011076  1.110798  1.321771  1.435259  0.976074  1.188600  1.242769  1.691486
0.674405  1.650976  1.041309  1.323483  0.902911  0.633518  1.593727  1.226484
0.516789  1.175434  0.713529  0.362291  0.748658  1.079728  1.059840  1.255697
0.167343 0.146098 0.953146 0.823571 1.223032 1.905638 1.586653 1.234108
Libera a memória do host.
Libera os recursos do dispositivo.
rogerio@guarani:/opt/AMDAPP/samples/opencl/cl/app/VectorAdd$
```

Para desenvolver uma nova aplicação basta copiar o diretório e modificar os `Makefiles` para que a nova aplicação passe a ser considerada no processo de compilação.

3. Fase única

Esse EP é constituído de apenas uma fase. O desafio é implementar as operações de *shift* das colunas e linhas de uma matriz.

Matriz original:
[10] [13] [27] [85]
[98] [33] [65] [34]
[62] [53] [23] [11]
[89] [18] [44] [78]

Após a execução de row-*shift(1)*

[98] [33] [65] [34]
[62] [53] [23] [11]
[89] [18] [44] [78]
[10] [13] [27] [85]

Após a execução de column-*shift(2)*

[65] [34] [98] [33]
[23] [11] [62] [53]
[44] [78] [89] [18]
[27] [85] [10] [13]

Defina o formato de entrada e de saída. Lembre-se de incluir os detalhes de entrada e de saída no relatório.

O trabalho pode ser individual ou em grupo de duas pessoas. A entrega deve ser feita pelo Paca, por meio de um arquivo ZIP ou TAR.GZ com o nome dos integrantes. Exemplos:

- MarianaAlfredo.zip

- MarianaBravoAlfredoGoldman.tar.gz

Nada de iniciais, por favor! Esse arquivo deve conter o código-fonte do programa desenvolvido. Basta compactar e enviar o diretório da sua aplicação, não precisa enviar o SDK inteiro e lembre-se de executar um `make clean` antes da compactação.
O código será compilado no ambiente que preparei e da forma que especifiquei. Portanto, prepare e use desta maneira. Também é importante que seja enviado um relatório explicando detalhes sobre a utilização e a implementação do algoritmo.

4. Critérios de correção

Deverão ser entregues o código-fonte e o relatório da fase única. O relatório deve conter os detalhes da implementação, os pontos positivos (facilidades) e negativos (dificuldades) encontrados durante o desenvolvimento da aplicação.

5. Recursos

Accelerated Parallel Processing (APP) SDK:

http://developer.amd.com/tools/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk/