AMD Core Math Library for Graphic Processors

Release Notes for Version 1.0

Publication #: 31208	**Revision:** 3.00
Issue Date: March 2009	

Overview

The AMD Core Math Library for Graphic Processors (ACML-GPU) is a mathematic function library designed to allow high-performance computing applications to take advantage of the computing power of AMD FireStream general-purpose graphics processors. ACML-GPU provides GPU-tuned subsets of the routines available in the well-established AMD Core Math Library. The tuned routines help enable rapid migration of certain types of floating point calculations into systems which include a graphics processor.

ACML-GPU includes a complete set of ACML routines. Certain key BLAS routines are optimized to run on the graphics processor. Many of the LAPACK functions also call these BLAS routines, which helps to improve performance when the BLAS function is a significant part of the overall computation. Because the BLAS and LAPACK routines are de facto standards, applications which call these routines need only link in the ACML-GPU library to gain the performance benefits of the graphics processor.

For complete ACML documentation, refer to the online ACML User Guide:

http://developer.amd.com/assets/acml_userguide.pdf.

Features

ACML-GPU includes the following features:

- Automatic selection of GPU or CPU algorithms based on problem size
- Seamless migration of applications from CPU to GPU
- Selection of GPU or CPU algorithms using an environment variable

The first release of ACML-GPU provides GPU enabled versions of the following routines:

- DGEMM
- SGEMM

Supported Graphics Cards

ACML-GPU supports graphics cards supported by the ATI Stream SDK, including AMD FireStream™ 9270 and ATI Radeon™ HD 4870. For more information on supported graphics cards, see the ATI Stream Computing System Requirements page at:

http://developer.amd.com/gpu/ATIStreamSDK/pages/ATIStreamSystemRequirements.aspx

System Requirements

The following software and hardware components are required to use ACML-GPU in the Microsoft® Windows® operating system.

- Windows XP Professional, 64-bit edition or Windows Vista®, 64-bit edition
- Supported AMD graphics card
- 2 GB of system DRAM (4 GB of DRAM is recommended)

- Appropriate 64-bit ATI Catalyst™ drivers for the supported ATI graphics card
- ATI Stream Software Development Kit (SDK), version 1.4 (optional)
- Microsoft Visual Studio 2005 SP1 C compiler (or later version)
- PGI Visual Fortran, version 7.1-6 (required to build the Fortran examples)

The following software and hardware components are required to use ACML-GPU in the Linux operating system.

- SuSE Linux® SLES 10.3 or Red Hat Linux RHEL 5.1, 64-bit installation
- Supported ATI graphics card
- 4 GB of system DRAM
- Appropriate 64-bit ATI Catalyst™ drivers for the supported ATI graphics card
- ATI Stream Software Development Kit (SDK), version 1.4 (optional)
- GCC/GFORTRAN, version 4.1.2

ACML-GPU is available for Microsoft Windows 64-bit Edition and Linux environments. The Windows library is built with the PGI Visual Fortran compiler, and is compatible with the Microsoft Visual Studio development environment. The Linux library is built with GCC/GFORTRAN.

ACML-GPU depends on ATI Stream runtime libraries. The ATI Stream runtime libraries are now included with the ATI Catalyst base display driver installation. Current versions of the drivers require that the graphics card be installed as the primary display device.

ACML-GPU supports systems with multiple graphics cards. The current revision supports multiple cards by dividing the work from DGEMM and SGEMM calls equally among the available GPUs. The library does not provide multithreaded capability and it is not currently thread safe. Only one program may call ACML-GPU at a time, and multiple threads within that program may not concurrently call the SGEMM and DGEMM routines.

The Windows version of ACML-GPU and the ATI Stream interface layer do not operate correctly when used on a Windows Remote Desktop connection.

Installation

Install ACML-GPU as follows.

1. Make sure the system has a PCI Express® slot of the correct size and an adequate power supply (see http://ati.amd.com/products/radeonhd3800/requirements.html or http://ati.amd.com/ products/ radeonhd4800/requirements.html for more information).
2. Install the AMD graphics card by following the manufacturer instructions.
3. Install the ATI Catalyst base drivers.
4. Restart the system.
5. Verify that the base drivers are working correctly.
6. Install the ATI Stream SDK. Run the ATI Stream SDK examples to verify proper operation. This step is optional since the ATI Stream runtimes are now included in the ATI Catalyst base driver installation.
7. Install ACML-GPU as follows. For the Windows operating system, unzip the compressed archive or use Windows Explorer to move the library and example files to a working directory. For the Linux operating system, extract from the .tgz archive to a working directory using `tar -zxf "filename"`. The performance examples run DGEMM problems as large as 8000×8000. Problems of this size can require up to 1.3 GB of the available system memory to run. It may be possible to run the examples in a system with only 2 GB of installed system memory, but 4 GB of installed memory is recommended. For larger problems, more system memory is required.

The Windows version of ACML-GPU includes a dynamic link library that was built with Microsoft Visual C++ 2008. This library requires the corresponding runtime library components to be installed. If these are not already installed on your computer, you can download them directly from Microsoft:

http://www.microsoft.com/downloads/details.aspx?familyid=BD2A6171-E2D6-4230-B809-9A8D7548C1B6&displaylang=en

You can also navigate to www.microsoft.com and search for *Microsoft Visual C++ 2008 Redistributable Package (x64)*.

Building the Examples

The ACML-GPU package includes examples that can be used to benchmark performance. The examples also show how to call the routines and link the library into an application. For the Windows operating system, the example Visual Studio projects demonstrate how to call the project from C and FORTRAN. To use these projects,

1. Double-click the .sln file to open Visual Studio.
2. Type **F7** or select **Build… >Build Solution…** to build the example application.
3. Run the application from within Visual Studio or from a **Command Prompt** window.

To build the examples from the command line, run `acmlexamples.bat`. PGI Visual Fortran must be installed to build the Fortran examples.

For the Linux operating system, a makefile is used to create the C and FORTRAN examples.

1. Open a console window.

2. Change the current directory to the installed example directory.
3. Type `make` to build the example programs.
4. Run individual example programs from the command line.

Environment Variable Requirements

To run ACML-GPU, operating system environmental variables must be modified.

Updating the Microsoft® Windows® PATH variable

There are multiple ways to access and change Windows system environment variables.

To access the variable in Windows XP Professional, either right-click **My Computer** and select **Properties** or go to **Start>Control Panel>System**. After the **System Properties** panel opens, select the **Advanced** tab, click **Environment Variables,** and choose the variable from the list in the **System Variables** pane. Click **Edit** to view or change the variable value.

To access the variable in Windows Vista®, either right-click **Computer** and select **Properties** or choose **Start>Control Panel>System and Maintenance>System>Advanced system settings.** After the **System Properties** panel opens, select the **Advanced** tab, click **Environment Variables,** and choose the variable from the list in the **System Variables** pane. Click **Edit** to view or change the variable value.

Updating PATH for `libacml_dll.dll`

The ACML-GPU examples refer to the `libacml_dll.dll` file. Be sure to add the location of the DLL to the PATH value before running applications. When one of the methods described above is used to access the PATH variable, the value can be changed by pasting or typing into the **Variable Value** field of the **Edit System Variables** window. The value can also be changed from the command line as follows.

Assuming that `libacml_dll.dll` is installed in `c:\acmlg1.0\win64\lib`.

Open a **Command Prompt** window and type

```
PATH="c:\acmlg1.0\win64\lib";%PATH%
```

ACML is also provided as a static (non-DLL) library named `libacml.lib`. The static library is in the same directory as the DLL. When the static library is used, there is no need to change PATH.

Updating PATH for the CAL DLL

The new ATI Catalyst base drivers place the required CAL DLLs into C:\Windows\System32. Therefore, changing the path for the CAL DLLs is not strictly necessary.

Updating the Linux® LD_LIBRARY_PATH Variable

Update the LD_LIBRARY_PATH as follows.

Open a console window and type

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/X11R6/lib64:/opt/acmlg1.0/
gfortran64/lib
```

Optional Environment Variables for Special Situations

ACML-GPU examines the value of the system environment variable **NO_GPU**. When the variable has the value “T”, the library does not use the GPU. This allows comparing performance on the GPU to performance on the CPU without rebuilding an application.

After the PATH and LD_LIBRARY_PATH variables have been set as described above, no additional environment variables should be required. However, the ACML-GPU library will recognize and respond to certain optional environment variables, described in this section, that are intended only for use in special situations. Normally, these environment variables will not be needed.

```
set ACML_GPU_VISTAX2KLUDGE=<anything>
```

Set this environment variable if you are experiencing slow performance from HD4870X2 video cards under the Windows Vista® operating system. There are some potential performance problems with large matrixes in this configuration. The problem does not arise on Windows XP systems. This issue is expected to be addressed by a future release of the ATI Catalyst device drivers. As a temporary work-around, when this environment variable is set, the library will use significantly less of the video RAM available on the video card, circumventing a bottleneck.

This flag is ignored and has no effect under any other operating system.

If this environment variable is detected, ACML-GPU will write an informational acknowledgment message to stderr.

```
set ACML_GPU=0x<mask>
```

When two or more GPUs are installed in a system, ACML-GPU will automatically detect their presence, and attempt to gain control of them as it's compute resources. Large problems are automatically split into multiple work units and given to the multiple GPUs to be processed in parallel. To allocate GPU resources differently, use this environment variable to limit ACML-GPU's use of the available processors. The value string of the variable should be set to a hexadecimal value using C-style 0x<hex-digits> syntax. This value is interpreted as a bit mask to allow (if the corresponding bit is set) or deny (if clear) ACML-GPU access to individual GPUs.

For example, if you have 4 HD4870 video cards installed in your computer, and you want to execute two ACML-GPU application processes concurrently, you might "set ACML_GPU=0x03" in the first process context so that it will use gpu0 and gpu1, and "set ACML_GPU=0x0C" in the second process context, so that it will use gpu2 and gpu3.

If this environment variable is detected, ACML-GPU will write an informational acknowledgment message to stderr, listing the GPU processors detected and which ones it will or will not attempt to use.

```
set ACML_GPU_QUIET=<anything>
```

ACML-GPU may write informational or warning messages to stderr. This would include the "calResCreate2D API" warning described below in the *Other Performance Considerations* section. Also, if either of the optional environment variables mentioned above is set, ACML-GPU will write an acknowledgment to stderr. Setting this variable to any value will suppress these warning and information messages. It will not suppress any severe error messages.

Performance Notes

Graphics processor acceleration is appropriate for problems large enough to justify the transfer of data to the GPU. For small problems, transfer reduces performance compared to computing the problem with the CPU. ACML-GPU uses the CPU for smaller problems and uses the GPU for problems large enough to benefit from acceleration.

The current revision of ACML-GPU is single-threaded. It is not built with OpenMP, and can run calculations on either the CPU or the GPU, but not on both in parallel.

ACML-GPU initializes the GPU interface environment during the first call to a GPU-enabled routine. Initialization takes approximately 1-2 seconds. Initialization time can cause the first call to take significantly longer than expected. Initialization time must be considered when measuring performance.

Users with special needs might prefer to decide for themselves which matrix multiplications are executed on the GPU. For these situations, the header file libcalblas.h is provided that describes certain routines implemented in libCALBLAS.dll (for Windows) or libCALBLAS.so (for Linux). You can also call the SGEMM and DGEMM routines in these shared libraries directly, and the operation will be executed on the GPU regardless of size.

Using this interface directly is not recommended for the following reasons:

- Whereas the SGEMM and DGEMM APIs implemented in libacml_dll are designed to be compatible with other implementations of BLAS, the routines in libCALBLAS do not have the same interfaces and will require programming.
- Checking of parameters being passed to SGEMM and DGEMM is bypassed.
- AMD makes no guarantee or assurance that the APIs exported from libCALBLAS will be implemented in future releases.
- AMD may not provide any technical support for programs using those APIs.

Other Performance Considerations

The current revision of ACML-GPU may print a message to stderr that starts with:

"**WARNING**: calResCreate2D API is not functioning"

This warning indicates that CAL and the ATI Catalyst drivers are not supporting this feature that is normally used by the libCALBLAS interface layer to improve performance. The libCALBLAS interface layer will use a fall back mechanism. Although performance will be slightly reduced, the code will still function properly, and this warning can be ignored.

To suppress this warning message, set an optional environment variable (see above). You can also upgrade to the latest version of the ATI Catalyst drivers.

Troubleshooting

To diagnose installation and build issues, start by verifying proper operation of the graphics drivers. Catalyst Control Center should be operating correctly and the information screens should properly identify the installed graphics card.

Ensure a monitor is plugged into the graphics card. A monitor is required for each card in a system with multiple cards.

Use the fglrxinfo utility to verify that the X windows system is using the ATI driver properly. The fgl_glxgears application can also be used. It should display a rapidly changing cube with spinning gears.

Verify that a *DRI* section is specified in /etc/X11/xorg.conf. Also verify that it is loaded in the *Module* section. The aticonfig -initial command should properly create these settings.

If the graphics drivers are working properly, run the ATI Stream SDK examples in `c:\Program Files(x86)\ATI\ATI CAL 1.4.0\bin` or in `/usr/local/atical/samples`. There are many examples and they can all be run to verify proper CAL operation.

Finally, try building and running the ACML examples. There are C and Fortran examples. Once these are working properly, use the example projects and makefiles to verify the correct build options for your application.

If performance is not as expected, perform the following checks.

Make sure the correct DLL is referenced by the executable program. For instance, when CPU-only libraries are loaded on the system, the system environment variable PATH can point to `libacml_dll.dll` from the CPU library version rather than the GPU version.

Consider the size of the problems. Large problems allow ACML-GPU to provide the best performance. Consider instrumenting calls to DGEMM or SGEMM to determine calling patterns.

For Linux operating systems, make sure the processor is not operating in power-saving mode and is operating at maximum CPU clock speed. Clock speed can be determined by examining the CPU MHz advertised by /proc/cpuinfo. On SuSE systems, use the `powersave -f command` to select maximum clock speed. On Red Hat systems, turn off the cpuspeed daemon to enable maximum speed.

If you are using the Windows Vista® operating system, and have installed video cards with dual processors on the same card (i.e., products with the X2 suffix such as the HD4870X2), and you are experiencing significantly slower performance when processing large matrixes than with small ones, you are probably encountering a known performance issue. This problem does not arise on Windows XP systems. If this is the case, there is an optional environment variable you can set to circumvent it. See above for additional details.