AMD Core Math Library for Graphic Processors

User Guide for Version 1.1

Publication # **31208**	Revision: **3.01**
Issue Date: **July 2010**	

Advanced Micro Devices

Trademarks

Contents

Revision History

Date	Revision	Description
June 2010	3.01	Second Public release. Enables use of generic AMD general-purpose graphic processors. Provides GPU enabled versions of the BLAS general matrix multiplication routines. Support has been added for more GPUs. New features have been added.
March 2009	3.00	Initial Public release.

Chapter 1 Overview

The AMD Core Math Library for Graphic Processors (ACML-GPU) is a mathematical function library designed to allow high-performance computing applications to take advantage of the computing power of AMD 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.

ACML-GPU is available for Microsoft Windows 64-bit Edition and Linux environments.

1.1 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.

Release 1.1 of ACML-GPU provides GPU enabled versions of the BLAS general matrix multiplication routines:

- DGEMM
- SGEMM
- CGEMM
- ZGEMM

1.2 What's New in This Release

Support has been added for more GPUs than were supported by version 1.0, including the ATI Radeon™ HD 5800 series. Also, this version of the library is designed to work with future ATI graphics processing units as soon as they are supported by an ATI Catalyst™ device driver.

The complex arithmetic matrix multiplications, CGEMM and ZGEMM are accelerated in this version.

A number of features have been added detect errors that occur in the GPU, in the display driver, or in the ATI Compute Abstraction Layer (CAL) runtime. When such an error is detected, ACML-GPU will, if possible, recover and compute correct results, either by retrying the GPU computation, or by doing the computation on the CPU.

Chapter 2 System Requirements

ACML-GPU is based upon the ATI Stream SDK. The platforms supported by ACML-GPU are those supported by the ATI Stream SDK, with one major exception: only 64-bit operating systems and applications are supported. For more information or for more current information, see the ATI Stream Software Development Kit System Requirements at:

http://developer.amd.com/gpu/ATIStreamSDK/Pages/default.aspx#two

2.1 Processors

A 64-bit x86 processor with at least 2GB of memory is required. (4GB or more is recommended.)

An AMD-brand processor is recommended, but not required.

A PCI Express® slot of the correct size and an adequate power supply are required. (This depends on the graphics card chosen for use.)

2.2 Operating system

A 64-bit Windows® or Linux® operating system is required. Any 64-bit OS that is supported by the ATI Stream SDK should work, including:

- Windows XP Professional, 64-bit edition
- Windows Vista®, 64-bit editions
- Windows 7 64-bit editions.
- OpenSUSE 11.2 or later (64-bit)
- Ubuntu 9.10 or later (64-bit)
- Red Hat Enterprise Linux 5.4 or later (64-bit).

32-bit operating systems – even those supported by the ATI Stream SDK – are **not** supported by ACML-GPU.

2.3 Supported Graphics Cards

ACML-GPU supports graphics cards supported by the ATI Stream SDK, including ATI FireStream™ and ATI Radeon™ HD products. For a current list of supported graphics cards, see the *ATI Stream SDK System Requirements online*.

ACML-GPU version 1.1 is designed to be forward compatible with future GPUs as soon as they are supported by the latest ATI Catalyst™ drivers.

Current versions of the drivers require that the graphics card be installed as the primary display device, and most cards require that a monitor be attached.

ACML-GPU supports systems with multiple graphics cards. See Section 3.1, on page 13, for more details.

Not all supported GPUs are capable of double-precision arithmetic. If your GPU is only capable of single-precision arithmetic, ACML-GPU will use it for the single-precision BLAS routines, SGEMM and CGEMM. But in that case, the double-precision BLAS routines DGEMM and ZGEMM will be executed by the CPU and not be GPU-accelerated. AMD recommends using the "Info" utility to check whether or not a GPU has double-precision arithmetic. Refer to section 4.2, on page 16, for more information.

2.4 ATI Catalyst™ Device Driver

The appropriate 64-bit ATI Catalyst™ drivers for your ATI graphics card must be installed.

(For Linux®, the ATI proprietary driver is required.) We recommend always installing the latest version of the driver.

For GPGPU applications based on ATI Stream technology, a monitor may need to be attached to each GPU device, and the Windows or Linux desktop must be extended to that device. For more detailed information, see the ATI Stream SDK online documentation at *http://developer.amd.com/gpu/ATIStreamSDK/pages/Documentation.aspx*.

2.5 ATI Stream Software Development Kit (Optional)

ACML-GPU is built upon the ATI Stream SDK. Installing the ATI Stream SDK is not required, but is *highly recommended* for systems intended for software development. If you have any difficulties using ACML-GPU, the first troubleshooting step would be to download the SDK, and use it as a debugging tool. There are no redistributable runtime binaries associated with the ATI Stream SDK; The runtime binaries are installed on target systems by installing the ATI Catalyst device driver.

ACML-GPU 1.1 is based on the ATI Compute Abstraction Layer (CAL), so registration of the OpenCL ICD on Linux systems is not necessary.

2.6 Supported Compilers

For Windows development:

- Microsoft® Visual Studio 2008 SP1 C compiler (or later version)
- PGI Visual Fortran, version 10.5
- Intel Fortran, version 11.1

For Linux development:

- GCC/GFORTRAN, version 4.4.1

A Fortran compiler is required to build the Fortran examples.

C/C++ programmers can dynamically link to the Windows .DLL (dynamic link library) or Linux .so (shared object) file without having to install the Fortran compiler. However, the correct Fortran compiler and it's libraries are needed to statically link the library.

Be sure and download the version of ACML-GPU that works with your compiler.

2.7 Windows® Microsoft® Visual C++ 2008 64-bit Redistributables

The Windows versions of ACML-GPU include 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. We recommend installing both of these packages:

- http://tinyurl.com/29hh4f4 - Microsoft Visual C++ 2008 Redistributable Package (x64)
- http://tinyurl.com/cbyany - Microsoft Visual C++ 2008 SP1 Redistributable Package (x64)

If the URLs above are no longer functioning, go to www.microsoft.com and search for “Visual Studio 2008 redistributables”.

Chapter 3 Using ACML-GPU

In general, no source code changes should be needed to use the GPU-accelerated library. If the ACML, or another BLAS library, is already in use, then only the replacement libraries require installation, and the link for the application may require refreshing.

3.1 Multiple GPUs

ACML-GPU supports systems with multiple graphics cards, and graphics cards containing multiple GPUs. The current revision may divide the work from large GEMM calls equally among the available GPUs. This depends on the sizes of the matrixes and the amount of video memory available on each card. In many cases, the library may choose to execute the GEMM operation on a single GPU, even when multiple GPUs are present, in order to deliver the best performance.

If two GPUs are configured as ATI CrossFireX™, they will be treated as a single GPU by the operating system and the device driver. In this configuration, applications using Microsoft® DirectX® are able to benefit from the improved performance of having rendering tasks shared by the two GPUs. However, in this configuration, applications using ATI Stream such as ACML-GPU are only able to access the resources of one GPU. For GPGPU computing, AMD recommends disabling ATI CrossfireX.

Other special installation and configuration steps may be needed for effective GPGPU use of multiple GPUs, especially on Linux. You should reference the latest ATI Stream SDK installation notes for the most up-to-date instructions.

If you have multiple GPUs and want to have multiple processes using those GPUs, (for example, in an MPI environment), we recommend using the ACML_GPU environment variable to assign GPUs to processes. See section 6.4.1, on page 20, for more information.

3.2 Not Multithreaded

The library does not provide multithreaded capability. Only one program should call ACML-GPU at a time, and multiple threads within that program should not concurrently call the GEMM routines.

On the first call to an accelerated function, the ACML-GPU library will allocate most of the GPU resources available, including local memory on the GPU. If multiple programs attempt to use the library concurrently, only the first task will be able to allocate these resources. The second and subsequent programs will fail to allocate the GPU resources they need. These programs will print warning messages to stderr, and perform all computations on the CPU.

3.3 Remote Access

Special steps are needed to use GPGPU computing when remotely accessing the computer with the GPU. If this ability is needed, please follow the instructions in the Knowledge Base article, which is included separately in the package.

3.4 Calling libCALBLAS Directly

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 call the GEMM 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 GEMM 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 GEMM routines 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.

One possible advantage of using these interfaces is that libCALBLAS can be used side-by-side with the regular, non-GPU-accelerated version of ACML.

3.5 Numerical Accuracy

GEMM products computed with this library should very closely match those computed with any other BLAS library.

Agreement will not be perfect, and small differences between BLAS implementations are caused by:

- Terms being combined in different orders.
- The ATI Radeon™ HD 5800 series of GPUs uses a double-precision fused multiply-add (FMA), where a multiplication, followed by the addition of the product and another value are performed in a single machine instruction, with higher accuracy than can be produced by performing these operations separately. This means that matrix products computed with this library on ATI Radeon HD 5800 processors will be more accurate than most BLAS libraries that perform the computation on a CPU.
- The ATI Radeon HD 4000 series of GPUs does not support denormal (or subnormal) IEEE floating-point values. These bit patterns are interpreted as if they were zeroes.

Chapter 4 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.
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. (Optional, but recommended) Install the ATI Stream SDK. Run one or more ATI Stream SDK examples to verify proper operation.
7. For the Windows operating system, unzip the compressed archive or use Windows Explorer to move the library and example files to a working directory.
8. For the Linux operating system, extract from the .tgz archive to a working directory using tar -zxf "filename".
9. Ensure that the ACML-GPU libraries are on the PATH.
10. For the Windows operating system, include the folder in the PATH environment variable. For the Linux operating systems, include the folder in the LD_LIBRARY_PATH variable. In either case, see Chapter 6, on page 19, for details.
11. Modify the screen-saver settings as described in section 4.1, on page 15.
12. Verify the operation of the library by running the Info utility. See section 4.2, on page 16, for details.

4.1 Disable DirectX® or OpenCL™ Screen-Savers

Screen savers that either use Microsoft DirectX® or OpenCL™ will interfere with GPGPU computation. This may cause undetected errors and can cause ACML-GPU to return incorrect results.

To ensure correct operation of ACML-GPU (or any other application using the ATI Stream SDK), either the screen-saver operation should be disabled, or a screen-saver that does not use OpenCL or DirectX should be selected, such as the blank-screen screen-saver.

On Windows operating systems, the ACML-GPU library will attempt to suppress the screen saver function by periodically calling the Windows SDK SendInput API.

4.2 Use the "Info" Utility to Verify Installation

The ACML-GPU package contains a simple utility named "Info" that writes much useful information to stdout. This includes information about the processor, the ATI driver installation, and the GPUs available for GPU computing. Immediately after installing the package, attempt to execute the "Info" utility program from a command line window. If the generated report shows statistics and capabilities for your installed GPU(s), this will verify that everything is installed correctly. On the other hand, if the report indicates there are no GPUs installed (or fewer GPUs than there should be), this indicates an installation problem. See Chapter 8, Troubleshooting, on page 25, for more information.

Chapter 5 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.

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. ZGEMM samples may require even more memory. 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.

On Windows® Operating System:

PGI Visual Fortran or Intel Fortran must be installed to build the Fortran examples.

To build these projects using Microsoft Visual Studio,

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.

On 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.

Chapter 6 Environment Variables

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

6.1 Updating the Microsoft® Windows® Environment Variables

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.

6.2 Windows® PATH Variable

The ACML-GPU examples use the dynamic link libraries libacml_dll.dll and libCALBLAS.dll. Be sure to add the location of these DLLs to the PATH value before running applications.

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.

6.3 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

6.4 Optional Environment Variables for Special Situations

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.

Note: *The NO_GPU environment variable in ACML-GPU 1.0 is no longer supported. Use ACML_GPU=0x0 instead.*

6.4.1 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 there are four ATI Radeon™ HD 4870 video cards installed in a computer, and two ACML-GPU applications are required to run concurrently, then one possibility is to "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.

This variable may be set from within the program calling ACML-GPU, but if so, it must be set before the first call to an accelerated GEMM routine. For example, in an MPI-based program, we would recommend running 1 copy of the program for each GPU installed on the cluster node. The MPI program could then select it's GPU by setting this environment variable.

6.4.2 set ACML_GPU_VISTAX2KLUDGE=<anything>

Set this environment variable (set ACML_GPU_VISTAX2KLUDGE=<anything>) if slow performance from HD4870X2 video cards under the Windows Vista® or Windows® 7 operating system is experienced. 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 workaround, 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.

6.4.3 set ACML_GPU_QUIET=<anything>

ACML-GPU may write informational or warning messages to stderr. This would include the "calResCreate2D API" warning described in Chapter 7, on page 23. Also, if either of the optional environment variables previously mentioned is set, ACML-GPU will write an acknowledgment to stderr. Setting this variable to any value will suppress these warnings and information messages. It will not suppress any severe error messages.

Chapter 7 Performance Notes

Graphics processor acceleration is appropriate for problems large enough to justify the transfer of data to the GPU. For small problems, the data transfers reduce 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.

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 the warning can be ignored. To suppress the warning message set an optional environment variable. See Chapter 6, on page 19 for environment variables. AMD recommends also upgrading to the latest version of the ATI Catalyst drivers.

Chapter 8 Troubleshooting

To diagnose installation and build issues, start by verifying proper operation of the graphics drivers.

Catalyst™ Control Center software should be operating correctly and the information screens should properly identify the installed graphics card.

Ensure a monitor is attached to the graphics card. A monitor may be required for each card in a system with multiple cards.

On Linux® systems 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.

On Linux systems 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.

Use the “Info” utility; see section 4.2, on page 16, to verify the installation.

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.

If multiple versions of ACML have been downloaded, 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 Windows Vista® or Windows® 7 operating systems are used, and have installed video cards with dual processors on the same card (such as the ATI Radeon™ HD 4870X2), and significantly slower performance is experienced when processing large matrixes than with small ones, then a known performance issue is probably being encountered. This problem does not arise on Windows XP systems. If this is the case, there is an optional environment variable that can be set to circumvent it. See section 6.4.2, on page 21 for additional details.