rCUDA v18.10beta
User’s Guide
October, 2018

The rCUDA Team

www.rcuda.net info@rcuda.net
Disclaimer:

This is a beta release and therefore unexpected errors may occur. Additionally, the performance of this release is not optimized yet. Thus, this release is not intended to be used in performance measurements.

Important information:

We will be very happy to receive your feedback and comments. Notice, however, that we are a small team and thus we are not able to provide immediate support for the rCUDA technology. Actually, your feedback and comments will likely be incorporated into the next rCUDA release. Thank you for your patience and understanding.
Changelog:

The most important changes in this version of rCUDA with respect to version v18.0.3beta are:

- Improved support for Deep Learning frameworks using GPUs in different remote servers
- A large amount of bugs have been fixed
- Version 7.0 of the cuDNN library is now supported
- The NCCL library is now supported
- A license server has been introduced into the rCUDA suite. In order to use rCUDA, you need first to set the license server in your cluster. Please refer to the installation chapter later in this guide for more information.

The rCUDA Team hopes that you enjoy this new version of the rCUDA technology!
The most important changes in the previous version of rCUDA (v18.03beta) were:

- The architecture of the rCUDA server has been renewed from scratch in order to provide full support to multi-threaded applications
- The rCUDA daemon now makes a better use of the GPU memory, allowing more memory for applications
- Support for functions in the Driver API has been improved
- Management module is now supported
- Support for Deep Learning frameworks such as TensorFlow (v1.2 and v1.3) or Caffe has been introduced
- Rendering programs such as Blender are now supported
- Support for the cuSOLVER, cuBLAS-XT and NVGRAPH libraries has been introduced
- The NPP library is partially supported
- Version 6.0 of the cuDNN library is now supported
- The rCUDA server uses now a single TCP port to provide service to all clients. The need to shut down the firewall at the server node has therefore been removed
- The RCUDAPROTO environment variable has been replaced by the RCUDA_NETWORK environment variable. Please update the scripts you used with previous versions of rCUDA
Notice:

rCUDA v18.10beta provides support for the following versions of CUDA:

- CUDA 9.0
- CUDA 8.0

Support for other CUDA versions can be provided under request.
The rCUDA Team is committed with the quality of the rCUDA middleware. Because of this, we devote some human resources to testing the rCUDA technology. Unfortunately, our budget is very small and therefore testing activities make progress very slowly. Nevertheless, we have already tried this new version of rCUDA with some applications, which can be found in the table below. We expect this table to be larger in future releases of the rCUDA middleware. Notice, however, that applications not present in this table may still work with rCUDA. In this regard, it is important to remark that this table only reflects the results of our testing activities, which make progress slowly.

<table>
<thead>
<tr>
<th>Application</th>
<th>Tested with CUDA 8.0 and with CUDA 9.0</th>
<th>TCP</th>
<th>InfiniBand</th>
<th>TCP</th>
<th>InfiniBand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>K40</td>
<td>P100</td>
<td>K40</td>
<td>P100</td>
</tr>
<tr>
<td>BarraCUDA</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
</tr>
<tr>
<td>CUDAmerge</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
</tr>
<tr>
<td>CUDAsw</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
<td>ok</td>
</tr>
<tr>
<td>GPUBlast</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td></td>
</tr>
<tr>
<td>GROMACS</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
<td></td>
</tr>
<tr>
<td>LAMMP5</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
<td></td>
</tr>
<tr>
<td>MAGMA</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td></td>
</tr>
<tr>
<td>NAMD</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td></td>
</tr>
<tr>
<td>GPUlibSVM</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td></td>
</tr>
</tbody>
</table>

rCUDA is also being tested with Deep Learning frameworks. In the tables below you can find a summary of our current testing activities.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Tested with CUDA 8.0 &amp; cuDNN 5.1</th>
<th>TCP</th>
<th>InfiniBand</th>
<th>TCP</th>
<th>InfiniBand</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>K40</td>
<td>P100</td>
<td>K40</td>
<td>P100</td>
</tr>
<tr>
<td>cuDNN</td>
<td>MNIST</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>multiGPU</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>AlexNet</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 Multi</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>MNIST</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Benchmark</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Inception</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>ImageNet</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td>TensorFlo</td>
<td>MNIST</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>ImageNet</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td>Tested with CUDA 9.0 &amp; cuDNN 7.0</td>
<td>TCP</td>
<td>InfiniBand</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----</td>
<td>------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>K40</td>
<td>P100</td>
<td>K40</td>
<td>P100</td>
<td></td>
</tr>
<tr>
<td>cuDNN</td>
<td>MNIST</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>multiGPU</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>AlexNet</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10 Multi</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>MNIST</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Benchmark</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Inception</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>ImageNet</td>
<td>ok</td>
<td>in progress</td>
<td>ok</td>
<td>in progress</td>
</tr>
<tr>
<td>TensorFlow</td>
<td>MNIST</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>CIFAR10</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>ok</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
<tr>
<td></td>
<td>ImageNet</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
<td>in progress</td>
</tr>
</tbody>
</table>
Please cite the following papers in any published work if you use the rCUDA software:


rCUDA TERMS OF USE

IMPORTANT - READ BEFORE COPYING, INSTALLING OR USING. Do not copy, install, or use the rCUDA software provided under this license agreement, until you have carefully read the following terms and conditions.

1. LICENSE

1.1 Subject to your compliance with the terms included in this document, you are granted a limited, non-exclusive, non-sublicensable, non-assignable, free of charge license to install the copy of the rCUDA software you received with this license (hereafter ‘rCUDA’ or “Software”) on a personal computer or other device; and personally use the Software. The rCUDA Team reserves all rights not explicitly granted to you under these terms.

1.2 Restrictions. You may not and you agree not to:

(a) sublicense, sell, assign, rent, lease, export, import, distribute or transfer or otherwise grant rights to any third party in the Software;

(b) undertake, cause, permit or authorize the modification, creation of derivative works or improvements, translation, reverse engineering, decompiling, disassembling, decryption, emulation, hacking, discovery or attempted discovery of the source code or protocols of the Software or any part or features thereof (except to the extent permitted by law);

(c) publish or post performance results without explicit authorization from The rCUDA Team;

(d) remove, obscure or alter any copyright notices or other proprietary notices included in the Software;

(e) obtain benefits from or charge additional costs for using the Software or causing the Software (or any part of it) to be used within or to provide commercial products or services to third parties. For such a use, please contact The rCUDA Team.

1.3 Third Party Technology. If you are using Software pre-loaded on, embedded in, combined, distributed or used with or downloaded onto third party products, hardware, software applications, programs or devices (“Third Party Technology”), you agree and acknowledge that: (a) you may be required to enter into a separate license agreement with the relevant third party owner or licensor for the use of such Third Party Technology; (b) some rCUDA functionalities may not be accessible through the Third Party Technology and (c) The rCUDA Team cannot guarantee that the Software shall always be available on or in connection with such Third Party Technology.

2. PROPRIETARY RIGHTS

2.1 The Software, and rCUDA Website contain proprietary and confidential information that is protected by intellectual property laws and treaties.

2.2 The content and compilation of contents included in the rCUDA Website, such as texts, graphics, logos, icons, images and software, are intellectual property of The rCUDA Team. Additionally, Universitat Politecnica de Valencia retains industrial property and exploitation rights of the rCUDA software, and are protected by copyright laws. Such copyright protected content cannot be reproduced without explicit permission from The rCUDA Team. You will not take any action to jeopardize, limit or interfere with intellectual property rights in the Software and/or rCUDA Website.

2.3 In case you, or any of your employees or students, publish any article or other material resulting from the use of the rCUDA software, that publication must cite the following references:


3. EXCLUSION OF WARRANTIES, LIMITATION OF LIABILITY AND INDEMNITY
3.1 No Warranties: TO THE MAXIMUM EXTENT PERMITTED BY LAW: THE SOFTWARE AND rCUDA WEBSITE ARE PROVIDED "AS IS" AND USED AT YOUR SOLE RISK WITH NO WARRANTIES WHATSOEVER; THE rCUDA TEAM DOES NOT MAKE ANY WARRANTIES, CLAIMS OR REPRESENTATIONS AND EXPRESSLY DISCLAIMS ALL SUCH WARRANTIES OF ANY KIND, WHETHER EXPPLICIT, IMPLIED OR STATUTORY, WITH RESPECT TO THE SOFTWARE, AND/OR rCUDA WEBSITE INCLUDING, WITHOUT LIMITATION, WARRANTIES OR CONDITIONS OF QUALITY, PERFORMANCE, NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR USE FOR A PARTICULAR PURPOSE. rCUDA FURTHER DOES NOT REPRESENT OR WARRANT THAT THE SOFTWARE, AND/OR rCUDA WEBSITE WILL ALWAYS BE AVAILABLE, ACCESSIBLE, UNINTERRUPTED, TIMELY, SECURE, ACCURATE, COMPLETE AND ERROR-FREE OR WILL OPERATE WITHOUT PACKET LOSS, NOR DOES rCUDA WARRANT ANY CONNECTION TO OR TRANSMISSION FROM THE INTERNET.

3.2 No Liability: YOU ACKNOWLEDGE AND AGREE THAT rCUDA WILL HAVE NO LIABILITY WHATSOEVER, WHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE) OR ANY OTHER THEORY OF LIABILITY, AND WHETHER OR NOT THE POSSIBILITY OF SUCH DAMAGES OR LOSSES HAS BEEN NOTIFIED TO rCUDA, IN CONNECTION WITH OR ARISING FROM YOUR USE OF rCUDA WEBSITE OR SOFTWARE. YOUR ONLY RIGHT OR REMEDY WITH RESPECT TO ANY PROBLEMS OR DISSATISFACTION WITH SUCH SOFTWARE AND/OR rCUDA WEBSITE IS TO IMMEDIATELY DEINSTALL SUCH SOFTWARE AND CEASE USE OF SUCH SOFTWARE AND/OR rCUDA WEBSITE. Furthermore, rCUDA shall not be liable to you, whether in contract, tort (including negligence) or any other theory of liability, and whether or not the possibility of such damages or losses has been notified to rCUDA, for:

(a) any indirect, special, incidental or consequential damages; or
(b) any loss of income, business, actual or anticipated profits, opportunity, goodwill or reputation (whether direct or indirect); or
(c) any damage to or corruption of data (whether direct or indirect); or
(d) any claim, damage or loss (whether direct or indirect) arising from or relating to:

any product or service provided by a third party under their own terms of service, including without limitation; any Third Party Technology; any third party website.

3.3 If any third party brings a claim against rCUDA in connection with, or arising out of (i) your breach of these Terms; (ii) your breach of any applicable law of regulation; (iii) your infringement or violation of the rights of any third parties (including intellectual property rights), you will indemnify and hold rCUDA harmless from and against all damages, liability, loss, costs and expenses (including reasonable legal fees and costs) related to such claim.

4. TERM OF THE LICENSE AND REGULATION LAWS
4.1. The License shall be valid for period of time equal to the term of the rights to the Software held by Universitat Politècnica de Valencia.

4.2. For matters not expressly provided herein, this agreement shall be subject to the provisions of the Spanish regulations.

4.3. In the event of any conflict, the parties agree to refer to the Courts of the city of Valencia (Spain), waiving their own jurisdiction.
## Contents

1 Introduction  

2 Fixing Most Common Errors with rCUDA  
   2.1 Error 1: "mlock error: Cannot allocate memory"  
   2.2 Error 2: "rCUDA error: function cuGetExportTable not supported"  

3 Installation  
   3.1 Setting up the rCUDA License Server  

4 Components and usage  
   4.1 Client Side  
   4.2 Server Side  
   4.2.1 cuDNN Users  
   4.3 Support for P2P Memory Copies between Remote GPUs  

5 Current limitations  

6 Further Information  

7 Credits  
   7.1 Coordination & Supervision
7.2 Development & Testing ........................................... 27
7.3 rCUDA Team Address ............................................. 27
Chapter 1

Introduction

The rCUDA framework enables the usage of remote CUDA-compatible devices. To enable a remote GPU-based acceleration, this framework creates virtual CUDA-compatible devices on those machines without a local GPU. These virtual devices represent physical GPUs located in a remote host offering GPU services. By leveraging the remote GPU virtualization technique, rCUDA allows to decouple CUDA accelerators from the nodes where they are installed, so that GPUs across the cluster can be seamlessly accessed from any node. Furthermore, nodes in the cluster can concurrently access remote GPUs. Figure 1.1 graphically depicts the additional flexibility provided by rCUDA.

rCUDA can be useful in the following three different environments:

**HPC Clusters and datacenters.** In this context, rCUDA increases the flexibility of using the GPUs present in the cluster. Sharing a given GPU among several applications is made possible. In this way, when rCUDA is used along with the SLURM job scheduler, the time required to complete the execution of a job batch is noticeably reduced. This causes that waiting time for jobs is smaller. Furthermore, GPU utilization is increased at the same time that the overall energy required to execute the job batch is reduced.

**Virtual Machines.** In this scenario, rCUDA enables the shared access from the inside of the virtual machine to the CUDA accelerator(s) installed in the physical machine. In addition to allow accessing the accelerators installed in the real machine that hosts the virtual ones, it is also possible to access remote GPUs from the virtual domain.

**Academia.** When using rCUDA in a teaching lab with a commodity network, our middleware offers concurrent access to a few high performance GPUs from the computers in the lab used by students as well as their laptops, or even virtual machines in the teaching lab. This reduces the acquisition costs of the lab infrastructure.
(a) When rCUDA is not deployed in the cluster, one or more GPUs must be installed in those nodes of the cluster intended for GPU computing. This usually leads to cluster configurations with one or more GPUs attached to all the nodes of the cluster. Nevertheless, GPU utilization may be lower than 100%, thus wasting hardware resources and delaying amortizing initial expenses.

(b) When rCUDA is leveraged in the cluster, only those GPUs actually needed to address overall workload must be installed in the cluster, thus reducing initial acquisition costs and overall energy consumption. rCUDA allows sharing the (reduced amount of) GPUs present in the cluster among all the nodes.

(c) From a logical point of view, GPUs in the cluster can be seen as a pool of GPUs detached from the nodes and accessible through the cluster interconnect, in the same way as networked storage (NAS) is shared among all the cluster nodes and concurrently accessed by them.

Figure 1.1: Different cluster configurations: (a) the traditional CUDA-based cluster deployment; (b) physical view of the cluster when leveraging rCUDA; (c) logical view of the cluster with rCUDA.
The current version of rCUDA (v18.10beta) implements all functions in the CUDA Runtime API and CUDA Driver API version 8.0 and version 9.0 excluding those related with graphics interoperability and Unified Memory Management. It also implements most of the functions in the following libraries of CUDA Toolkit 8.0 and 9.0: cuRAND, cuBLAS, cuSPARSE, cuFFT, cuSOLVER, cuBLAS-XT, NCCL and NVGRAPH. The NPP library is partially supported. Additionally, versions 5.1, 6.0 and 7.0 of the cuDNN library are supported. Other libraries provided by NVIDIA will be supported in future rCUDA releases. rCUDA 18.10beta targets the Linux operating system for 64-bit x86-based configurations. It provides support for the same Linux distributions as CUDA does.
Chapter 2

Fixing Most Common Errors with rCUDA

2.1 Error 1: "mlock error: Cannot allocate memory"

While using rCUDA you may get the error message "mlock error: Cannot allocate memory"

In order to fix this error, you can add the following lines at the end of file "/etc/security/limits.conf" in both client and server nodes:

```
* hard memlock unlimited
* soft memlock unlimited
```

After rebooting the system, you can run the following command to verify that the limits have been changed:

```
$ ulimit -a | grep "max locked memory"
```

You should get an output similar to "max locked memory (kbytes, -l) unlimited".
2.2 Error 2: "rCUDA error: function cuGetExportTable not supported"

This error is caused because the application was compiled to use static libraries (in particular, the CUDA library). Notice that rCUDA needs applications to be compiled to use CUDA as a dynamic library. That is, a compilation using CUDA dynamic libraries is needed to allow the use of the rCUDA software. This step can be done by the user in two different ways:

- If nvcc compiler is used, the flag -cudart=shared is needed.
- If gcc/++ compiler is used, the -lcudart flag is needed.
Chapter 3

Installation

Notice that prior to use the rCUDA middleware, you MUST have a license key and appropriately configure the rCUDA license server in your cluster.

Typically, if you are reading this document, you should already received a license key from the rCUDA Team. If not, please ask the rCUDA Team for a license key at support@rcuda.net

The installation of the rCUDA software is very simple. The binaries of the rCUDA software are distributed within a tarball which has to be decompressed manually by the user. The steps to install the rCUDA binaries are:

1. Decompress the rCUDA package.
2. Install and configure the rCUDA license server as it is explained in Section 3.1 (you should already have a license key).
3. Copy the rCUDA/lib folder to the client(s) node(s) (without GPU) as it is explained in Section 4.1.
4. Copy the rCUDA folder to the server node (with GPU) as it is explained in Section 4.2.

3.1 Setting up the rCUDA License Server

The rCUDA license server is a daemon that must be in execution in one of the nodes of the cluster before the rCUDA server is executed. It is important to know that a license key is required by the rCUDA license server.
In order to get a license key, the user was previously provided by the rCUDA Team with a program that gathers some information of the node that will run the license server. This information comprises, among other parameters, the MAC address of the Ethernet adapter of the node. Other pieces of information are also gathered. Because of this, the license key received from the rCUDA Team can only be used in the same node where the information was initially gathered.

After receiving the license key from the rCUDA Team, the user must store it in the config/rCUDA.conf file. In this way, the user must include a new line in that file, similar to:

```
RCUDA_LICENSE_KEY=3232148765453730306042551...
```

You should include in the config/rCUDA.conf file a line with the IP address of the node where the license server is being run, such as

```
RCUDA_LICENSE_SERVER=192.168.0.34
```

Now the license server can be executed. To that end please launch the rCUDAis program located in the bin folder. It is helpful to execute the license server in interactive and verbose mode by using the -iv options. That is, you should type:

```
./rCUDAis -iv
```

You should get an output similar to:

```
rCUDAis v1.0
Copyright 2009-2018 UNIVERSITAT POLITÈCNICA DE VALENCIA. All rights reserved.
rCUDAis[16812]: License Features:
- Expiration date: 31 Dec 2019
- Max rCUDA server(s): 10
- Max remote GPU(s): 50
- Max virtual GPU(s): 1 per remote GPU
- Allow Migration: no
rCUDAis[16812]: License server daemon successfully started.
```

Once you have checked that the rCUDA license server is properly configured and running, you might decide to run it in non-interactive and non-verbose mode if desired. To that end, just do not use the -iv options.

Finally, notice that the rCUDA license server listens in the TCP port number 8309. Therefore, if a firewall is in execution in the node running the rCUDA license server, then it must be configured to allow incoming connections to TCP port 8309.
Chapter 4

Components and usage

rCUDA is organized following a client-server distributed architecture, as shown in Figure 4.1. The client middleware is contacted by the application demanding GPGPU services, both running in the same cluster node. The rCUDA client presents to the application the very same interface as the regular NVIDIA CUDA Runtime and Driver APIs. Upon reception of a request from the application, the client middleware processes it and forwards the corresponding requests to the rCUDA server middleware. In turn, the server interprets the requests and performs the required processing by accessing the real GPU to execute the corresponding request. Once the GPU has completed the execution of the requested command, results are gathered by the rCUDA server, which sends them back to the client middleware. There, the results are finally forwarded to the demanding application.

In order to optimize client/server data exchange, rCUDA employs a customized application-level communication protocol. Furthermore, rCUDA provides efficient support for several underlying network technologies. To that end, rCUDA supports runtime-loadable specific communication modules that currently target the InfiniBand network (using InfiniBand verbs), the RoCE network (using RDMA) and the general TCP/IP protocol stack (see Figure 4.1). Additional network technologies may be supported in the future.

4.1 Client Side

The client side of the rCUDA middleware is a library of wrappers that replaces the CUDA Toolkit dynamic libraries mentioned at the end of Chapter 1. In this way, CUDA applications that use rCUDA are not aware of being accessing an external device. Also, they do not need any source code modification.
The rCUDA client is distributed in a set of files: “libcuda.so.m.n”, “libcudart.so.x.y”, “libcublas.so.x.y”, “libcufft.so.x.y”, “libcusparse.so.x.y”, “libcurand.so.x.y” and “libcudnn.so.x.y”. These shared libraries should be placed in those machines accessing remote GPGPU services. Set the LD_LIBRARY_PATH environment variable according to the final location of these files (typically “/opt/rCUDA/lib”, “/usr/local/rCUDA/lib”, or “$HOME/rCUDA/lib”, for instance).

In order to properly execute the applications using the rCUDA library, set the following environment variables:

- **RCUDA_DEVICE_COUNT**: indicates the number of GPUs which are accessible from the current node.
  Usage: “RCUDA_DEVICE_COUNT=<number_of_GPUs>”
  For example, if the current node will access two remote GPUs:
  “RCUDADEVICE_COUNT=2”

- **RCUDADEVICE_X**: indicates where GPU X, for the client being configured, is located.
  Usage: “RCUDADEVICE_X=<server[@<port>]:[GPU number]>”
  For example, if GPUs 0 and 1 assigned to the current client are located at server “192.168.0.1” using the default rCUDA port (8308):
  “RCUDADEVICE_0=192.168.0.1”
  “RCUDADEVICE_1=192.168.0.1:1”

1. m.n are based on the exact version of the CUDA driver.
2. x.y refer to the exact version of CUDA supported by the provided rCUDA package.
Furthermore, as the nvcc compiler links with CUDA static libraries by default, a compilation using CUDA dynamic libraries is needed to allow the use of the rCUDA software. This step can be done by the user in two different ways:

- If nvcc compiler is used, the flag -cudart=shared is needed.
- If gcc/++ compiler is used, the -lcudart flag is needed.

In case the user of rCUDA is compiling the NVIDIA CUDA Samples, he/she should notice that NVIDIA CUDA Samples must be compiled after the EXTRA_NVCCFLAGS environment variable has been set to --cudart=shared.

If an InfiniBand network is available and the rCUDA user prefers to use the high performance InfiniBand Verbs APIs instead of the lower performance TCP/IP socket API, then the following environment variables should be considered:

- **RCUDA_NETWORK**: This environment variable must be set to “IB” in order to make use of the InfiniBand Verbs API. If this variable is not set, or if it is set to “TCP”, then the TCP/IP sockets API will be used even if an InfiniBand network is available. For example:
  - “RCUDA_NETWORK=IB” will use the InfiniBand Verbs API
  - “RCUDA_NETWORK=TCP” will use the TCP/IP sockets API even if an InfiniBand network is used

- **RCUDABDEVNO**: (Optional. Default value is 1). In case the computer where rCUDA is being used has two or more InfiniBand network adapters, then the user may instruct rCUDA what adapter to use by appropriately setting this environment variable to the number of the selected InfiniBand adapter. For example:
  - “RCUDABDEVNO=1” will use the first InfiniBand network adapter
  - “RCUDABDEVNO=2” will use the second InfiniBand network card

- **RCUDAIBPOR TNO** and **RCUDAIBPOR TNO2**: (Optional. By default only port 1 is used). In case the computer executing rCUDA makes use of a dual-port InfiniBand card, then the rCUDA user may select to use the first port, the second port, or both. In case the rCUDA user decides to use only one of the two ports, then the port to be used may be selected by setting the environment variable “RCUDAIBPOR TNO” to “1” or to “2”. In case the rCUDA user decides to concurrently use both ports, then the environment variables “RCUDAIBPOR TNO” and “RCUDAIBPOR TNO2” should be set to “1” and “2”, respectively. Notice that using both ports means that every single data transfer between the rCUDA client and server will be split among both ports, which will be concurrently used, thus aggregating their individual performance. For example:
  - “RCUDAIBPOR TNO=1” will use the first port of the InfiniBand network adapter
  - “RCUDAIBPOR TNO=2” will use the second port of the InfiniBand network card
  - “RCUDAIBPOR TNO=1” and “RCUDAIBPOR TNO2=2” will concurrently use both ports of the InfiniBand network adapter
It is important to remark that the `RCUDA_NETWORk` variable must be set both in the client and server sides with the same value. In addition, when using two ports, the `RCUDAIBPORTNO` and `RCUDAIBPORTNO2` variables must be set both in the client and server sides.

### 4.2 Server Side

The rCUDA server is configured as a daemon (rCUDAad) that runs in those nodes offering GPGPU acceleration services. The rCUDA server contacts the rCUDA license server in order to check that its execution is allowed. To that end, the rCUDA server needs to know the IP address of the node executing the rCUDA license server. A line with the license key should also be included. The way to achieve this is to include two new lines in the config/rCUDA.conf file such as

```
RCUDA_LICENSE_KEY=3232145765453730306042551...
RCUDA_LICENSE_SERVER=192.168.0.34
```

The IP address 192.168.0.34 is just an example. The actual IP address of the node running the rCUDA license server in your cluster should be used instead. Notice that every rCUDA server in the cluster must be configured to know the IP address of the rCUDA license server.

Set the `LD_LIBRARY_PATH` environment variable according to the location of the CUDA libraries (typically `"/usr/local/cuda/lib64"`). Notice that this is the path to the original CUDA libraries, not the rCUDA ones. Add also to the `LD_LIBRARY_PATH` environment variable the path to rCUDA cuDNN library (typically `"$HOME/rCUDA/lib/cudnn"`). See Section 4.2.1 for further information on the use of the cuDNN library.

Set the “`RCUDA_NETWORk`” environment variable to IB if an InfiniBand network is available and the InfiniBand Verbs API is to be used. The rest of environment variables related to the use of the InfiniBand Verbs API detailed in the previous section (such as `RCUDAIBDEVNO`, `RCUDAIBPORTNO`, and `RCUDAIBPORTNO2`) should also be considered.

This daemon offers the following command-line options:

- `-i` : Do not daemonize. Instead, run in interactive mode.
- `-l` : Local mode using AF_UNIX sockets (TCP only).
- `-n <number>` : Number of concurrent servers allowed. 0 stands for unlimited (default).
- `-p <port>` : Specify the port to listen to (default: 8308).
-v Verbese mode.
-h Print usage information.

We suggest you to use the -iv options the rst time you execute the rCUDA server in order to check that it properly works. Once you have veri ed that the rCUDA server can connect to the license server, you might stop using the interactive and verbose modes if desired.

4.2.1 cuDNN Users

If you are not going to use the NVIDIA CUDA Deep Neural Network library (cuDNN), you can ignore this section.

If, on the contrary, you plan to use this library, please, notice that in order to use the cuDNN library, the LD_LIBRARY_PATH environment variable in the server node must contain the location of the NVIDIA cuDNN libraries (typically "/usr/local/cudnn/lib"), instead of the rCUDA ones (typically "$HOME/rCUDA/lib/cudnn").

In addition, note that the cuDNN library is distributed separately from the CUDA package and, therefore, must be explicitly downloaded and installed. Furthermore, notice that the cuDNN library is not supported by the rCUDA framework for CUDA versions earlier than 6.5.

4.3 Support for P2P Memory Copies between Remote GPUs

Figure 4.2 presents the possible scenarios when making peer-to-peer (P2P) memory copies with CUDA and rCUDA. As we can see, with CUDA there is only one possible scenario, depicted in Figure 4.2a, where the GPUs are located in the same cluster node and are interconnected by the PCIe link. On the contrary, when using rCUDA there are two possible scenarios for making copies between remote GPUs: (i) the remote GPUs are located in the same remote node and are interconnected by the PCIe link as shown in Figure 4.2b, and (ii) the remote GPUs are located in different remote nodes in the cluster and therefore they are interconnected by the network fabric, as depicted in Figure 4.2c.

By default, rCUDA supports the rst scenario exposed in Figure 4.2b. In this manner, it is possible to perform memory copies between remote GPUs located in the same server node. However, the second scenario presented in Figure 4.2c is only supported for InfiniBand networks. The TCP port for setting up the connection for P2P memory copies between remote GPUs could be indicated in the environment variable "RCUDAP2P_TCP_PORT". If it is not set, port number 18515 is used by default.
Figure 4.2: Possible scenarios for P2P memory copies with CUDA and rCUDA.
Chapter 5

Current limitations

The current implementation of rCUDA features the following limitations:

- Graphics interoperability is not implemented. Missing modules: OpenGL, Direct3D 9, Direct3D 10, Direct3D 11, VDPAU, Graphics

- The Profiler Control module is not supported.

- Unified Memory Management is not supported.

- Thrust library is not supported.

- Timing with the event management functions might be inaccurate, since these timings will discard network delays. Using standard Posix timing procedures such as “clock_gettime” is recommended.

- Please, notice that this is a beta release and therefore unexpected errors may occur. Additionally, the performance of this release is not optimized yet. Thus, this release is not intended to be used in performance measurements.
Chapter 6

Further Information

For further information, please refer to [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. Also, do not hesitate to contact support@rcuda.net for any questions or bug reports.
Chapter 7

Credits

7.1 Coordination & Supervision

Federico Silla
Email: fsilla@disca.upv.es

7.2 Development & Testing

Cristian Peñaranda, Javier Prades, Elías Serrano and Jaime Sierra
Emails: {cripeace, japraga}@gap.upv.es, fraserm3@disca.upv.es, jsierra@gap.upv.es

7.3 rCUDA Team Address

Federico Silla
Department of Computer Engineering
Universitat Politècnica de València
DISCA, Edificio 1G, Camino de Vera, s/n
46022 – Valencia, Spain
Email: support@rcuda.net
Acknowledgements

This work was supported by PROMETEO program from Generalitat Valenciana (GVA) under Grant PROMETEO/2008/000, by PROMETEO program phase II from Generalitat Valenciana (GVA) under Grant PROMETEOII/2013/009 and also by PROMETEO program from Generalitat Valenciana (GVA) under Grant PROMETEO/201/77. Authors are also grateful for the generous support provided by Mellanox Technologies.
Bibliography


Communications and IT Energy-aware Technologies, Chamonix, France, April 2014.


