# rCUDA v16.11
# User's Guide

November, 2016

The rCUDA Team

**www.rcuda.net** **info@rcuda.net**

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Department of Computer Engineering
Technical University of Valencia
Valencia, Spain

# Notice:

rCUDA v16.11 provides support for the following versions of CUDA:

- CUDA 8.0
- CUDA 7.5
- CUDA 7.0

Support for other CUDA versions can be provided under request.

Please cite the following papers in any published work if you use the rCUDA software:

- C. Reaño, F. Silla, G. Shainer and S. Schultz, **"Local and Remote GPUs Perform Similar with EDR 100G InfiniBand"**, in proceedings of the International Middleware Conference, Vancouver, BC, Canada, December 2015.
- C. Reaño and F. Silla, **"A Performance Comparison of CUDA Remote GPU Virtualization Frameworks"**, in proceedings of the International Conference on Cluster Computing, Chicago, IL, USA, September 2015.

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) sub-license, 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:

C. Reaño, F. Silla, G. Shainer and S. Schultz, "Local and Remote GPUs Perform Similar with EDR 100G InfiniBand", in proceedings of the International Middleware Conference, Vancouver, BC, Canada, December 2015.

C. Reaño, F. Silla, "A Performance Comparison of CUDA Remote GPU Virtualization Frameworks", in proceedings of the International Conference on Cluster Computing, Chicago, IL, USA, September 2015.

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 EXPLICIT, 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);

(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 Politecnica 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

# Chapter 1

# Introduction

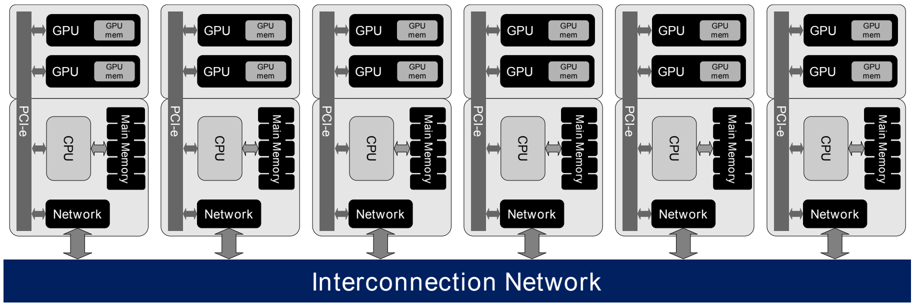
The rCUDA framework enables the usage of remote CUDA-compatible devices. To enable a remote GPU-based acceleration, this framework creates virtual CUDA-compatible devices on those machines without a local GPU. These virtual devices represent physical GPUs located in a remote host offering GPGPU services. 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:

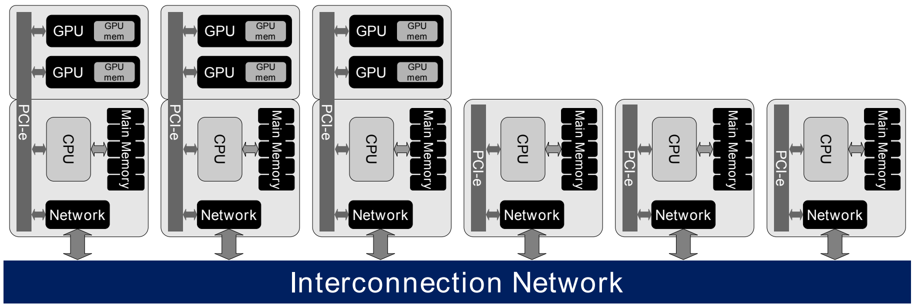
**HPC Clusters and datacenters.** In this context, rCUDA increases the flexibility of using the GPUs present in the cluster. Sharing a given GPU among several applications is made possible. In this way, when rCUDA is used along with the SLURM job scheduler, the time required to complete the execution 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 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.

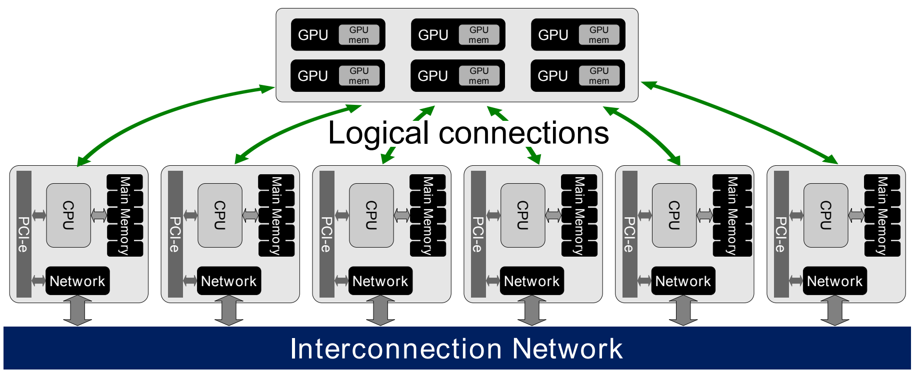
**Academia.** When using rCUDA in a teaching lab with a commodity network, our middleware offers concurrent access to a few high performance GPUs from the computers in the lab used by students as well as their laptops, or even virtual machines in the teaching lab. This reduces the acquisition costs of the lab infrastructure.




(a) When rCUDA is not deployed into 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 (v16.11) implements all functions in the CUDA Runtime API and CUDA Driver API version 8.0, (as well as CUDA 7.5 and CUDA 7.0), excluding those related with graphics interoperability, Unified Memory Management and Module management. It also implements all the functions in the following libraries of CUDA Toolkit 8.0, 7.5, and 7.0: cuRAND, cuBLAS (excluding complex), cuSPARSE, and cuFFT (excluding complex). The cuDNN version 5.1 library is partially supported. Other libraries provided by NVIDIA will be supported in future rCUDA releases. rCUDA 16.11 targets the Linux operating system (for 64-bit x86-based configurations). It provides support for the same Linux distributions as CUDA does.

# Chapter 2

# Installation

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

1. Decompress the rCUDA package.
2. Copy the rCUDA/lib folder to the client(s) node(s) (without GPU) as it is explained in Section 3.1.
3. Copy the rCUDA folder to the server node (with GPU) as it is explained in Section 3.2.

# Chapter 3

# Components and usage

rCUDA is organized following a client-server distributed architecture, as shown in Figure 3.1. 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 do so, rCUDA supports runtime-loadable specific communication modules that currently target the InfiniBand network (using InfiniBand verbs) and the general TCP/IP protocol stack (see Figure 3.1). Additional network technologies may be supported in the future.

## 3.1 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.

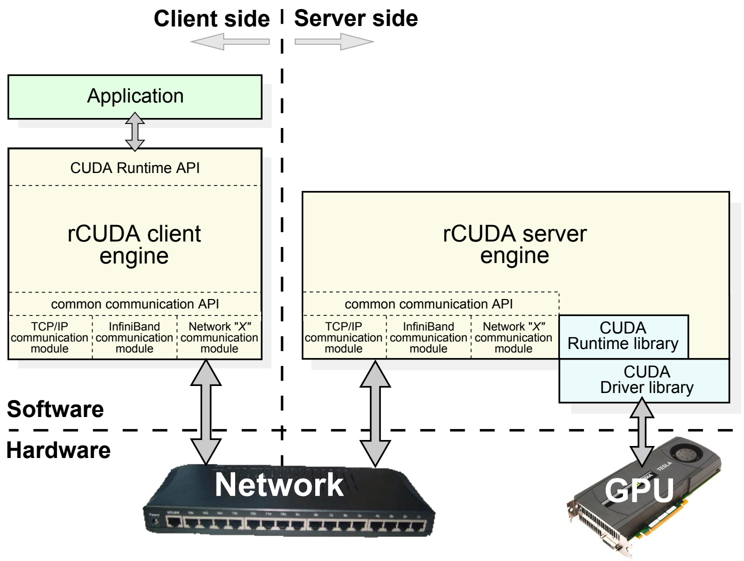



Figure 3.1: rCUDA architecture, showing also the runtime-loadable specific communication modules.

The rCUDA client is distributed in a set of files: "libcuda.so.$m.n$[1], libcudart.so.$x.y$[2], 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:
  "RCUDA_DEVICE_COUNT=2"
- RCUDA_DEVICE_X: indicates where GPU X, for the client being configured, is located.
  Usage: "RCUDA_DEVICE_X=<server[@<port>]>[:GPUnumber]"
  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):
  "RCUDA_DEVICE_0=192.168.0.1"
  "RCUDA_DEVICE_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:

- RCUDAPROTO: 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:
  "RCUDAPROTO=IB" will use the InfiniBand Verbs API
  "RCUDAPROTO=TCP" will use the TCP/IP sockets API even if an InfiniBand network is used
- RCUDAIBDEVNO: (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:
  "RCUDAIBDEVNO=1" will use the first InfiniBand network adapter
  "RCUDAIBDEVNO=2" will use the second InfiniBand network card
- RCUDAIBPORTNO and RCUDAIBPORTNO2: (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 "RCUDAIBPORTNO" to "1" or to "2". In case the rCUDA user decides to concurrently use both ports, then the environment variables "RCUDAIBPORTNO" and "RCUDAIBPORTNO2" 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:
  "RCUDAIBPORTNO=1" will use the first port of the InfiniBand network adapter
  "RCUDAIBPORTNO=2" will use the second port of the InfiniBand network card
  "RCUDAIBPORTNO=1" and "RCUDAIBPORTNO2=2" will concurrently use both ports of the InfiniBand network adapter

It is important to remark that the RCUDAPROTO 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.

## 3.2 Server Side

The rCUDA server is configured as a daemon (rCUDAd) that runs in those nodes offering GPGPU acceleration services.

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 3.2.1 for further information on the use of the cuDNN library.

Set the "RCUDAPROTO" 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** Verbose mode.

**-h** Print usage information.

### 3.2.1 cuDNN Users

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

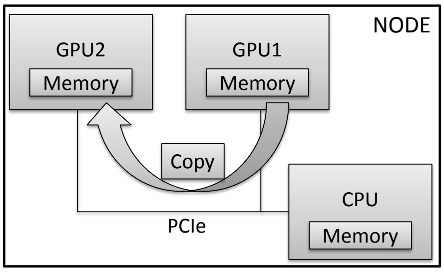
If, on the contrary, you plan to use this library, please, notice that in order to use the cuDNN library, the LD_LIBRARY_PATH environment variable in the server node must contain the location of the NVIDIA cuDNN libraries (typically "`/usr/local/cudnn/lib`"), instead of the rCUDA ones (typically "`$HOME/rCUDA/lib/cudnn`").

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.

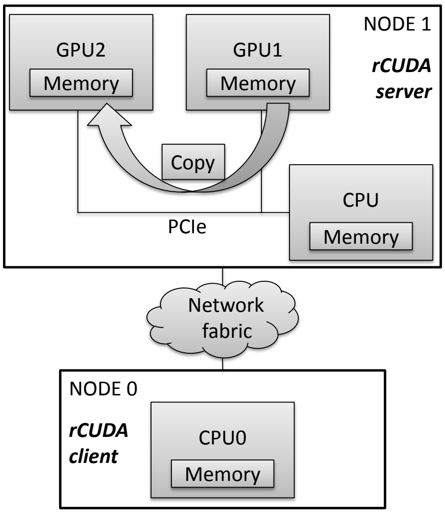
## 3.3 Support for P2P Memory Copies between Remote GPUs

Figure 3.2 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 3.2a, 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 3.2b, 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 3.2c.

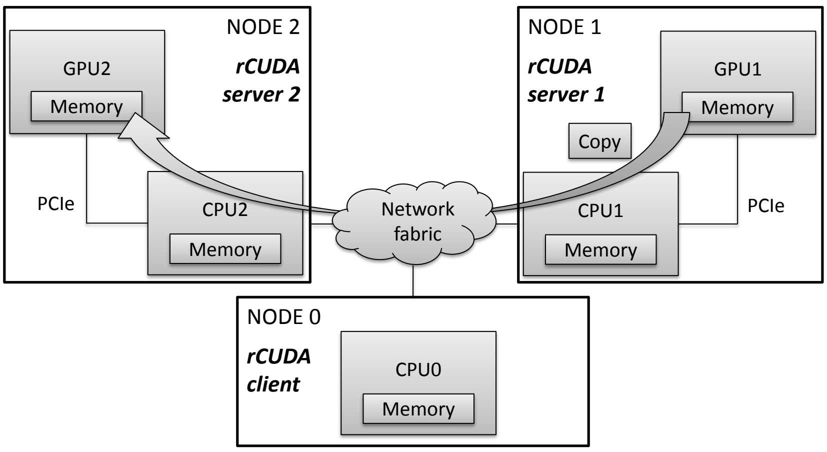
By default, rCUDA supports the first scenario exposed in Figure 3.2b. In this manner, it is possible to perform memory copies between remote GPUs located in the same server node. However, the second scenario presented in Figure 3.2c is only supported for InfiniBand networks. The TCP port for setting 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.




(a) CUDA scenario.




(b) rCUDA scenario 1.




(c) rCUDA scenario 2.

Figure 3.2: Possible scenarios for P2P memory copies with CUDA and rCUDA.

# Chapter 4

# 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.
- Module management 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.

# Chapter 5

# 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]. Also, do not hesitate to contact support@rcuda.net for any questions or bug reports.

# Chapter 6

# Credits

## 6.1 Coordination & Supervision

Federico Silla
Email: fsilla@disca.upv.es

## 6.2 Development & Testing

Javier Prades, Carlos Reaño and Jaime Sierra
Emails: japraga@gap.upv.es, carregon@gap.upv.es and jsierra@gap.upv.es

## 6.3 Advising

Jose Duato
Email: jduato@disca.upv.es

## 6.4 rCUDA Team Address

Federico Silla
Department of Computer Engineering
Technical University of Valencia
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/2008/060 and also by PROMETEO program phase II under Grant PROMETEOII/2013/009. Authors are also grateful for the generous support provided by Mellanox Technologies.

# Bibliography

[1] José Duato, Francisco D. Igual, Rafael Mayo, Antonio J. Peña, Enrique S. Quintana-Ortí, and Federico Silla. An efficient implementation of GPU virtualization in high performance clusters. In *Euro-Par 2009, Parallel Processing – Workshops*, volume 6043 of *Lecture Notes in Computer Science*, pages 385–394. Springer-Verlag, 2010.

[2] José Duato, Antonio J. Peña, Federico Silla, Rafael Mayo, and Enrique S. Quintana-Ortí. rCUDA: reducing the number of GPU-based accelerators in high performance clusters. In *Proceedings of the 2010 International Conference on High Performance Computing and Simulation (HPCS 2010)*, pages 224–231, Caen, France, June 2010.

[3] José Duato, Antonio J. Peña, Federico Silla, Rafael Mayo, and Enrique S. Quintana-Ortí. Performance of cuda virtualized remote gpus in high performance clusters. In *Proceedings of the 2011 International Conference on Parallel Processing (ICPP 2011)*, Taipei, Taiwan, September 2011.

[4] José Duato, Antonio J. Peña, Federico Silla, Juan C. Fernández, Rafael Mayo, and Enrique S. Quintana-Ortí. Enabling CUDA acceleration within virtual machines using rCUDA. In *Proceedings of the 2011 International Conference on High Performance Computing (HiPC 2011)*, Bangalore, India, December 2011.

[5] Carlos Reaño, Antonio J. Peña, Federico Silla, Rafael Mayo, Enrique S. Quintana-Ortí, and José Duato. CU2rCU: towards the complete rCUDA remote GPU virtualization and sharing solution. In *Proceedings of the 2012 International Conference on High Performance Computing (HiPC 2012)*, Pune, India, June 2012.

[6] Carlos Reaño, Antonio J. Peña, Federico Silla, Rafael Mayo, Enrique S. Quintana-Ortí, and José Duato. Influence of infiniband FDR on the performance of remote GPU virtualization. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER 2013)*, Indianapolis, USA, October 2013.

[7] Adrián Castelló, José Duato, Rafael Mayo, Antonio J. Peña, Enrique S. Quintana-Ortí, Vicente Roca, and Federico Silla. On the Use of Remote GPUs and Low-Power Processors for the Acceleration of Scientific Applications. In *The Fourth International Conference on Smart Grids, Green*

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

[8] Carlos Reaño, Federico Silla, Antonio J. Peña, Gilad Shainer, Scot Schultz, Adrián Castelló, Enrique S. Quintana-Ortí, and José Duato. Boosting the Performance of Remote GPU Virtualization Using InfiniBand Connect-IB and PCIe 3.0. In *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER 2014)*, Madrid, Spain, September 2014.

[9] Sergio Iserte, Adrián Castelló, Rafael Mayo, Enrique S. Quintana-Ortí, Carlos Reaño, Javier Prades, Federico Silla, and José Duato. SLURM Support for Remote GPU Virtualization: Implementation and Performance Study. In *Proceedings of the International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2014)*, Paris, France, October 2014.

[10] Antonio J. Peña, Carlos Reaño, Federico Silla, Rafael Mayo, Enrique S. Quintana-Orti, and José Duato. A complete and efficient CUDA-sharing solution for HPC clusters. *Parallel Computing*, 40(10):574 – 588, 2014.

[11] Carlos Reaño and Federico Silla. On the design of a demo for exhibiting rCUDA. In *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Shenzhen, Guangdong, China, May 2015.

[12] Carlos Reaño, Federico Silla, Adrián Castelló Gimeno, Antonio J. Peña, Rafael Mayo, Enrique S. Quintana-Ortí, and José Duato. Improving the user experience of the rCUDA remote GPU virtualization framework. *Concurrency and Computation: Practice and Experience*, 27(14):3746–3770, 2015.

[13] Carlos Reaño, Federico Silla, Gilad Shainer, and Scot Schultz. Local and remote GPUs perform similar with EDR 100G InfiniBand. In *Proceedings of the Industrial Track of the 16th International Middleware Conference, Middleware Industry 2015, Vancouver, BC, Canada, December 7-11, 2015*, pages 4:1–4:7, 2015.

[14] Carlos Reaño and Federico Silla. A live demo on remote GPU accelerated deep learning using the rCUDA middleware. In *Proceedings of the Posters and Demos Session of the 16th International Middleware Conference, Middleware Posters and Demos 2015, Vancouver, BC, Canada, December 7-11, 2015*, pages 3:1–3:2, 2015.

[15] Blesson Varghese, Javier Prades, Carlos Reaño, and Federico Silla. Acceleration-as-a-service: Exploiting virtualised GPUs for a financial application. In *11th IEEE International Conference on e-Science, e-Science 2015, Munich, Germany, August 31 - September 4, 2015*, pages 47–56, 2015.

[16] Carlos Reaño and Federico Silla. InfiniBand verbs optimizations for remote GPU virtualization. In *2015 IEEE International Conference on Cluster Computing, CLUSTER 2015, Chicago, IL, USA, September 8-11, 2015*, pages 825–832, 2015.

[17] Carlos Reaño and Federico Silla. A performance comparison of CUDA remote GPU virtualization frameworks. In *2015 IEEE International Conference on Cluster Computing, CLUSTER 2015, Chicago, IL, USA, September 8-11, 2015*, pages 488–489, 2015.

[18] Javier Prades, Carlos Reaño, and Federico Silla. CUDA acceleration for xen virtual machines in InfiniBand clusters with rCUDA. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP 2016, Barcelona, Spain, March 12-16, 2016*, pages 35:1–35:2, 2016.

[19] Sergio Iserte and Javier Prades and Carlos Reaño and Federico Silla. Increasing the Performance of Data Centers by Combining Remote GPU Virtualization with Slurm. In *Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID 2016)*, Cartagena, Colombia, May 2016.

[20] Javier Prades and Fernando Campos and Carlos Reaño and Federico Silla. GPGPU as a service: Providing GPU-acceleration services to federated cloud systems. In Gabor Kecskemeti, Attila Kertesz, and Zsolt Nemeth, editors, *Developing Interoperable and Federated Cloud Architecture*, chapter 10, pages 281–313. IGI Global, 2016.

[21] Ferran Perez, Carlos Reaño, and Federico Silla. Providing CUDA acceleration to KVM virtual machines in InfiniBand clusters with rcuda. In *Distributed Applications and Interoperable Systems - 16th IFIP WG 6.1 International Conference, DAIS 2016, Held as Part of the 11th International Federated Conference on Distributed Computing Techniques, DisCoTec 2016, Heraklion, Crete, Greece, June 6-9, 2016, Proceedings*, pages 82–95, 2016.

[22] Carlos Reaño and Federico Silla. Reducing the Performance Gap of Remote GPU Virtualization with InfiniBand Connect-IB. In *Proceedings of the 21st IEEE Symposium on Computers and Communications (ISCC 2016)*, Messina, Italy, June 2016.

[23] Carlos Reaño and Federico Silla. Tuning remote GPU virtualization for InfiniBand networks. *Accepted for publication in The Journal of Supercomputing*, 2016.

[24] Javier Prades, Blesson Varghese, Carlos Reaño, and Federico Silla. Multi-tenant virtual GPUs for optimising performance of a financial risk application. *Accepted for publication in Journal of Parallel and Distributed Computing*, 2016.

[25] Carlos Reaño and Federico Silla. Extending rCUDA with support for P2P memory copies between remote GPUs. In *Proceedings of the 18th IEEE International Conference on High Performance Computing and Communications (HPCC 2016)*, Sydney, Australia, December 2016.