

# Contents

<b>List of Figures</b>	xi
<b>List of Tables</b>	xix
<b>Abstract</b>	xxi
<b>Resumen</b>	xxiii
<b>Resum</b>	xxv
<b>1 Introduction</b>	1
1.1 Background	2
1.1.1 GPGPU	2
1.1.2 Main Concerns of Using GPUs	3
1.1.3 Remote GPU Virtualization	3
1.1.4 rCUDA	4
1.2 Objectives of the Thesis	5
1.3 Main Contributions of the Thesis	6
1.4 Thesis Outline	7
References	9
<b>2 On the Effect of using rCUDA to Provide CUDA Acceleration to Xen Virtual Machines</b>	13
2.1 Introduction	14
2.2 Providing CUDA GPUs to Virtual Machines	16
2.3 rCUDA: Remote CUDA	24
2.4 Testbeds Used in The Experiments	25
2.5 Network Performance Observed by Xen VMs	29
2.6 Performance of rCUDA within Xen VMs	31
2.7 Impact of Xen VMs on Real Applications	36
2.7.1 Applications Using One GPU	37
2.7.2 Applications Using Multiple GPUs	42
2.8 Conclusions	46
References	47
<b>3 Made-to-Measure GPUs on Virtual Machines with rCUDA</b>	53
3.1 Introduction	54
3.2 Motivation	56

3.3	Background on GPU Virtualization	60
3.4	Performance Evaluation	62
3.5	Conclusions	67
	References	68
<b>4</b>	<b>Multi-tenant virtual GPUs for optimising performance of a financial risk application</b>	<b>71</b>
4.1	Introduction	72
4.2	Related work	74
4.3	rCUDA	77
4.4	Financial risk application	81
4.4.1	Input and Output Data	82
4.4.2	Algorithm and GPU Implementation	84
4.5	Evaluation	86
4.5.1	Platform	87
4.5.2	Application Scalability	87
4.5.3	Reducing Execution Time Using rCUDA	90
4.5.4	Mitigating the Impact of Data Transfers in rCUDA	92
4.5.4.1	Concurrent vs Sequential Data Transfers	93
4.5.4.2	Multi-tenancy Approach	94
4.5.5	Performance Analysis Using Multi-tenancy	97
4.5.6	Modelling Multi-tenancy for Performance and Energy Estimation	98
4.5.6.1	Performance Model	98
4.5.6.2	Energy Model	102
4.6	Conclusions	105
	References	105
<b>5</b>	<b>Maximizing resource usage in Multi-fold Molecular Dynamics with rCUDA</b>	<b>111</b>
5.1	Introduction	112
5.2	Background	114
5.2.1	MD in Drug Discovery	114
5.2.2	rCUDA (remote CUDA)	115
5.3	System Configurations for Drug Discovery	116
5.4	Flavonoids as a Working Example	119
5.5	System Performance and Throughput	120
5.5.1	Test bed: Hardware and Software Environment	121
5.5.2	Performance Characterization	121
5.5.3	Throughput for Each Case Study	125
5.5.4	Overall System Throughput	131
5.5.5	Analysis of obtained MD results in terms of biological validation	134
5.6	Conclusions and Future Work	134
	References	137
<b>6</b>	<b>Turning GPUs into Floating Devices over The Cluster: The Beauty of GPU Migration</b>	<b>141</b>
6.1	Introduction	142

---

6.2	About Remote GPU Virtualization	145
6.3	Implementing GPU Migration	147
6.4	First results of GPU Migration within rCUDA	149
6.5	Conclusions	158
	References	158
<b>7</b>	<b>GPU-Job Migration: the rCUDA Case</b>	<b>163</b>
7.1	Introduction	164
7.2	About Remote GPU Virtualization	166
7.3	Related Work on GPU Migration	167
7.4	Implementing GPU Migration in rCUDA	169
7.5	Performance Evaluation of GPU Migration with rCUDA	172
7.5.1	Synthetic Application	173
7.5.2	Real Applications	177
7.5.3	Use Cases for GPU-Job Migration with rCUDA	184
7.5.3.1	GPU Server Consolidation	184
7.5.3.2	GPU Load Balancing	187
7.5.3.3	Improved Management of User Priorities	189
7.6	Conclusions	190
	References	191
<b>8</b>	<b>Conclusions</b>	<b>195</b>
8.1	Contributions	196
8.2	Future Work	198
8.2.1	GPU-Job Scheduler	198
8.2.2	Quantification of the Economic Impact of Applying the Mechanisms Developed in this Thesis	199
8.3	Publications	200
8.3.1	Main Publications	200
8.3.2	Main Collaborations	202
	References	206



# List of Figures

1.1 Architecture of the rCUDA middleware. . . . .	4
2.1 Typical architecture used by GPU virtualization solutions. . . . .	17
2.2 Performance comparison among three different GPU virtualization solutions: gVirtuS, DS-CUDA, and rCUDA. The comparison is performed in terms of attained bandwidth. The performance of CUDA is also depicted. Tests have been carried out in native domains with the hardware and software settings described in Section 2.4. . . . .	20
2.3 Typical configuration of a Xen-based system showing how the Ethernet adapter and the GPU available in the host are provided to VMs. The GPU is exclusively assigned to a single VM by making use of the PCI passthrough mechanism. Network connectivity among VMs and between VMs and the external network is provided by means of a software bridge that connects the internal virtual network to the real Ethernet adapter. . . . .	25
2.4 Testbeds used in the experiments presented in this paper, which make use of rCUDA to provide GPU access to VMs. (a) In a single-node testbed, VMs employ the virtual network to access the rCUDA server by means of the TCP/IP protocol stack. (b) When an InfiniBand fabric is available, VMs use such interconnect to access a remote rCUDA server. . . . .	26
2.5 Bandwidth attained by the virtual network among Xen VMs. . . . .	29
2.6 InfiniBand bandwidth tests using ConnectX-3 network cards executed in the different scenarios under study. . . . .	31

2.7	Bandwidth tests for copies between host and device memory, using CUDA and the rCUDA middleware. Tests have been carried out in the different scenarios depicted in Figure 2.4 as well as in native domains. . . . .	32
2.8	Performance of a synthetic application where the percentage of execution time devoted to data transfers to/from the GPU and the percentage of execution time used for computations in the GPU are set by the user. Notice that these percentages are initially established for the executions using CUDA with a local GPU (case “a”) by defining the amount of data to be transferred. For the rest of scenarios, this initial amount of data to be transferred is kept constant, thus producing a deviation of the initial percentages. Furthermore, for each size interval, the exact size of data transfers is randomly set. . . . .	35
2.9	Execution time of several applications when executed in different local and remote scenarios. Execution time is broken down into three components: GPU computation, GPU data transfer, and Other. . . . .	38
2.10	Average overhead with respect to executions with CUDA in a native domain for the four applications depicted in Figure 2.9. . . . .	39
2.11	Histograms showing the percentage of transferred data according to message size. . . . .	41
2.12	Average overhead experienced by applications with respect to executions with CUDA using the PCI passthrough from the inside of a VM. . . . .	42
2.13	Configuration of a Xen-based system showing two GPUs assigned to one of the VMs. The GPU assignment is carried out by making use of the PCI passthrough mechanism. Therefore, both GPUs can only be used by the VM owning them. . . . .	43

2.14 Testbeds used with rCUDA. Two GPUs are provided to VMs. (a) In a single-node scenario, VMs use the virtual network (TCP/IP) to access the rCUDA server running in one of the VMs. (b/c) When an InfiniBand fabric is available in the cluster, VMs use such interconnect in order to access the remote GPUs, which can be located either in the same (b) or in different (c) remote nodes.	44
2.15 Performance of two applications when executed in different local and remote scenarios involving Xen VMs.	45
3.1 Assignment of GPUs to VMs when using the PCI passthrough technique, which causes that GPUs are assigned to VMs in an exclusive way.	55
3.2 Assignment of GPUs to VMs when using the remote GPU virtualization mechanism, which allows GPUs to be concurrently shared among VMs.	56
3.3 Evolution of GPU utilization and memory occupancy during the execution of the four applications considered in this study.	58
3.4 Evolution of GPU utilization and memory occupancy during the execution of a sequence of instances of the four applications considered in this paper for one hour time interval. Only one application is executed at a time. The order of applications is random. Four different intensities for the system load are considered.	60
3.5 Architecture of the rCUDA middleware.	61
3.6 Execution time of the four applications under consideration in this study.	62
3.7 Test beds used in the experiments in this section. Four VMs are used with PCI passthrough (with four GPUs) whereas up to 12 VMs are leveraged with rCUDA (with the same four GPUs).	63
3.8 Average execution time for each of the applications considered in this study when executed in the scenarios depicted in Figure 3.7. 95% confidence intervals are also shown. Four different intensities for the system load are considered.	64

3.9	Total amount of jobs executed (system throughput) for each of the applications considered in this study when executed in the scenarios depicted in Figure 3.7. Four different intensities for the system load are considered.	65
3.10	Average overhead depending on system load for the different test beds depicted in Figure 3.7. Baseline for the overhead calculation is the performance for the PCI passthrough scenario (Figure 3.7(a)).	65
3.11	Average system throughput depending on system load for the different test beds depicted in Figure 3.7. Baseline for the throughput calculation is the performance for the PCI passthrough scenario (Figure 3.7(a)).	66
3.12	Energy consumption for each of the scenarios depicted in Figure 3.7. Labels "A", "B", "C" and "D" refer, respectively, to low, medium, high and maximum intensities of the system load.	66
4.1	Execution time of the financial application on multiple local GPUs	72
4.2	Distributed acceleration architecture facilitated by rCUDA	76
4.3	rCUDA client and server software/hardware stack	77
4.4	Communication sequence between a client and the rCUDA server daemon	79
4.5	Comparison of bandwidth for pinned memory and pageable memory of rCUDA, DS-CUDA and gVirtuS using CUDA as a baseline reference (DS-CUDA does not support pinned memory)	80
4.6	Computation and data transfer times for the financial risk application when executed on single and multiple GPUs with CUDA	88
4.7	Amount of data transferred during the execution of the financial risk application	89
4.8	Attained bandwidth when concurrent data transfers to GPUs are performed. Source data is located in the same memory bank.	90
4.9	Scalability of the financial risk application when executed with rCUDA.	91



---

4.10 Bandwidth attained for multiple data transfers concurrently to different remote GPUs using rCUDA. . . . .	91
4.11 Communication approaches for transferring data to GPUs. . . . .	92
4.12 GPU utilisation, power and energy consumption of concurrent and sequential data transfers to GPUs considered in Figure 4.11 . . . . .	94
4.13 Sequential data copies with several vGPUs per GPU. . . . .	94
4.14 GPU utilisation, power and energy consumption of the multi-tenancy approach considered in Figure 4.13. . . . .	96
4.15 Application performance for different combinations of pGPUs and vGPUs using QDR InfiniBand . . . . .	97
4.16 Application performance for different combinations of pGPUs and vGPUs using FDR InfiniBand . . . . .	98
4.17 Results from performance model for QDR InfiniBand . . . . .	101
4.18 Results from performance model for FDR InfiniBand . . . . .	102
4.19 Results from energy model for QDR InfiniBand . . . . .	103
4.20 Results from energy model for FDR InfiniBand . . . . .	104
4.21 Combined space of energy and execution time using QDR InfiniBand . . . . .	104
4.22 Combined space of energy and execution time using FDR InfiniBand . . . . .	105
5.1 Architecture of the rCUDA middleware . . . . .	115
5.2 Hardware configurations for each of the baseline case studies considered in this paper. . . . .	118
5.3 Performance of the MD simulations when 3, 5, 10 and 20 threads are leveraged. The three basic case studies are considered. . . . .	121
5.4 Energy per simulated ns required by the MD simulations when 3, 5, 10 and 20 threads are leveraged. The three basic case studies are considered. . . . .	123

---

5.5	Average power required by the GPU and by the rest of the system in the CUDA scenario. Peak power required by the entire node also shown. Simulator configurations using either 3, 5, 10 or 20 threads are considered.	124
5.6	Throughput of the CPU-only MD simulations when several instances are concurrently executed in the same node. Simulator configurations using either 3, 5, 10 or 20 threads are considered.	125
5.7	Energy per simulated ns required by GROMACS when several CPU-only instances are concurrently executed in the same node. Simulator configurations using either 3, 5, 10 or 20 threads are considered.	126
5.8	GPU memory and GPU utilization along the execution time of the GROMACS simulator configured to use 10 threads with the molecules under study. Simulation was configured to last 200 ns of simulated time.	127
5.9	Instant power and accumulated energy along the execution time of the GROMACS simulator configured to use 10 threads with the molecules under study. Simulation was configured to last 200 ns of simulated time. Instant power is split into GPU power and system power.	128
5.10	Throughput and GPU utilization when several instance of GROMACS share the GPU in the rCUDA server by leveraging the rCUDA middleware. Simulator configurations using either 20, 10, 5 or 3 threads are considered.	129
5.11	Energy per simulated ns required by GROMACS when several simulator instances share the GPU in the rCUDA server by leveraging the rCUDA middleware. Simulator configurations using either 20, 10, 5 or 3 threads are considered.	130
5.12	Aggregated throughput projection for a hybrid cluster composed of $n$ nodes where half of the nodes own a GPU whereas the other half of the nodes do not leverage any accelerator.	133
5.13	Aggregated throughput projection for a homogeneous cluster composed of $n$ nodes where all the nodes own one GPU.	133

5.14 RMSD over time for the DNA structure. . . . .	135
5.15 Average DNA(center of mass) to DEPHBC distance over time. . . . .	135
5.16 Superposition of first and last frame of the DNA-DEPHBC MD simulation.	135
6.1 Comparison, from a logical point of view, of two cluster configurations: (a) remote GPU virtualization is not leveraged; (b) remote GPU virtual- ization is used. . . . .	143
6.2 Usage of GPU migration in a cluster in order to consolidate GPU jobs and reduce energy. In (a) all the nodes in the cluster are switched on whereas in (b) seven nodes have been switched off thanks to GPU server consolidation after having migrated GPU jobs. . . . .	144
6.3 General organization of remote GPU virtualization frameworks. . . . .	146
6.4 Execution time using CUDA and rCUDA without migration. . . . .	150
6.5 Overhead introduced in executions of Figure 6.4 because of executing the applications with rCUDA using a remote GPU instead of using a local one with CUDA. No migration has been performed. . . . .	152
6.6 Execution time using CUDA and rCUDA. Executions with rCUDA have suffered one live migration process in order to move the GPU-part of the application to another GPU. . . . .	153
6.7 Overhead introduced because of carrying out one live migration while executing the applications with rCUDA using a remote GPU. . . . .	154
6.8 Estimated average time required to perform one live migration while ap- plications are in execution. . . . .	156
6.9 Overhead with respect to CUDA when applications are live migrated up to five times during their execution. . . . .	157
7.1 General organization of remote GPU virtualization frameworks. . . . .	166
7.2 Migration modules inside rCUDA client and server. . . . .	168

7.3 Complete operation of the migration module implemented within the rCUDA middleware. . . . .	170
7.4 Bandwidth attained for several network configurations using different transfer sizes. . . . .	174
7.5 Time required to migrate a job among two P100 GPUs located in different nodes. A synthetic application is leveraged. Several configurations of the FDR and EDR InfiniBand network adapters are used. Performance for the 1 Gbps Ethernet network is also displayed. . . . .	176
7.6 Memory configuration, in terms of total memory allocated and number of memory regions, for each of the applications considered. . . . .	179
7.7 Service downtime for each of the applications considered. Migration was triggered at 25% execution time for each of the applications. . . . .	180
7.8 Evolution of memory occupancy and GPU utilization during execution time of two of the applications considered in this study. Average GPU utilization for each of the applications is also shown. . . . .	182
7.9 Total migration time for the five applications considered in this study. Time is measured since the arrival of the external signal triggering migration until the application resumes execution in the destination GPU. . . . .	183
7.10 GPU migration used to consolidate servers. Two instances of the CUDA-MEME application are being executed in two servers and, at some point in time, the job scheduler decides to migrate one of the jobs to the other server. The emptied server can be later switched off if required. . . . .	186
7.11 Example of applying the GPU-job migration mechanism within rCUDA in order to balance the load among GPUs in the cluster. . . . .	188

# List of Tables

2.1 Data transfers in the applications under analysis . . . . .	40
4.1 Scalability of the financial risk application when executed using CUDA . .	87
4.2 Time in seconds for GPU memory allocation and data transfer tasks of the financial risk application . . . . .	100
5.1 Performance achieved by several GROMACS configurations . . . . .	132
7.1 Amount of seconds required for management tasks in Figure 7.5(b) . . .	177
7.2 Characterization of the real applications used to analyze the migration mechanism. . . . .	178
7.3 Execution time of the CloverLeaf application in different GPUs. . . . .	190