International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 04 | Apr -2017 www.irjet.net p-ISSN: 2395-0072

Implement rCUDA Framework on Embedded System

Mohamed Hussain [1], Viraj Choksi [2], M.B.Potdar [3]

[1] *Research scholar, Bhaskaracharya Institute for Space Applications and Geo-Informatics (BISAG), Gujarat, India*
[2] *Project Scientist, Bhaskaracharya Institute for Space Applications and Geo-Informatics (BISAG), Gujarat, India*
[3] *Project Director, Bhaskaracharya Institute for Space Applications and Geo-Informatics (BISAG), Gujarat, India*

Abstract - *CUDA is programming model developed and introduced by NVIDIA Company in 2006. This programming works on heterogeneous environment where we have two different types of processors CPU and GPU processors.Using CUDA programming allows the programmer to take the advantage of a GPU processor to execute a code which reduces the time of the execution. However using GPU processor in the system comes with some drawbacks such as increase the acquisition cost of the system, increase the power consumption, require more space for the new GPU hardware and in the system it relatively low use a GPU processor.*
To overcome those drawbacks in the system we going to use one of the visualization techniques to enable the application to use the remote GPU to execute the code .There are many visualization techniques which allows the programmer to use the remote GPU such as VGPU, GVirtuS, GridCuda, Shadowfax, GViM ,vCUDA, rCUDA and DS-CUDA.

In this paper we are going to implement rCUDA on one of embedded system platform (Jetson Tk1) to overcome those drawbacks on in the system .rCUDA is platform designed and developed by team of developer in Technical University of Valencia, Spain. The resonances of implementing rCUDA platform in the system in stand of the other platforms because it has more fidelity, showing better performance comparing to another visualization techniques and also it allows sharing the GPU processor between the clients.

Key Words: CUDA, rCUDA, embedded system, and Jetson Tk1.

1. INTRODUCTION

GPU (Graphics Processing Unit) is processor designed initially to handle the graphics operations especially after the graphics application become more complicated and present more overhead on CPU processor. the GPU processor became more popular as the demand for graphic applications increased. It has an ability to process the data in parallel way which boosts the performance and also it able to deal with 2D and 3D data. The success of the GPU cased the software developer start using the GPU for general purpose computation which is called GPGPU.
GPGPU (general purpose computing on graphics processing units) is methodology of using the GPU which is mainly

designed for graphics operation to perform the general purpose computations .GPGPU is always used for an applications which requires high-power CPUs such as, chemical physics, computational fluid dynamics image analysis and many other fields. But using the GPU processor in the system comes with some drawbacks such as increasing in the power consumptions, increase the acquisition costs, it also requires more space to fit the new hardware and the system not require the GPU processing in all operations which makes the GPU processor idle most of the time.

As mentioned previously adding the GPU processor to the system will increase the power consumption around 30% [1], the reasons of incensement in the power consumption is two processors are used in the system instead of one. The second drawback is increasing in an acquisition costs and this because of adding a new hardware to the system which is also required more space to fit in. Adding the GPU processor in the system it doesn't means that the applications will use it all the time, it will only use to execute part of an application code which requires a high number of repetitions, which means the GPU processor will be most the time idle.

CUDA (Compute Unified Device Architecture) is a programming model developed and introduced by NVIDIA in 2006, this programming platform works in heterogeneous platform has two different types of processors CPU and GPU processors, This platform one of the programming modules allows the programmer to use the power of the GPU processors for general purpose computations.it support different programming language such as C,C++,Fortran and Python which make it easier for the programmer to write the code without requirement to learn a new language, however CUDA programming model its only support NVIDIA hardware.

rCUDA (remote CUDA) is a middleware enables the device to use the remote GPU of another device to execute the GPU code. This platform developed by parallel architectures Group from Technical University of Valencia (Spain).This framework gives the applications concurrent access to GPUs installed in other nodes in cluster. In rCUDA if there are more than one application needs to access the GPU, the rCUDA middleware will be responsible to share the GPU between the applications. The source code of the applications does not requires any modification in the code to use the remote GPU, it only present overhead in execution time which is usually less than 4% [7].

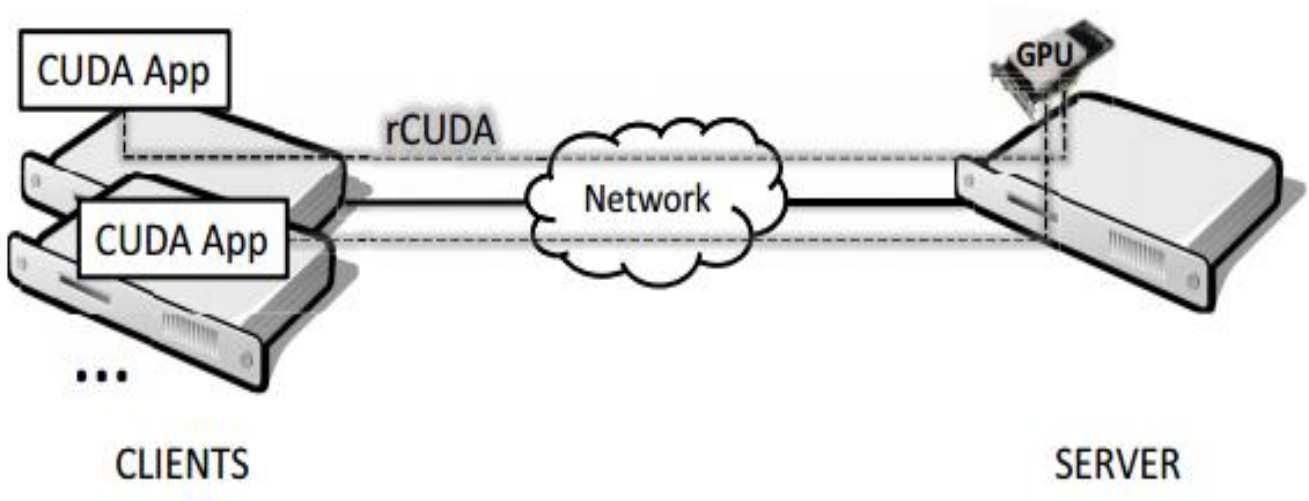

Fig -1: rCUDA Simple scenario [1]

Jetson TK1 (also called Tegra K1) is an embedded board designed and developed by NIVIDA company.it is one of the two solutions released by the NVIDIA for embedded system support GPU processor.
Jetson TK1 has 192 CUDA GPU Cores (Kepler GPU), 4 ARM Cortex-A-15 CPUs and 2 ISP cores. Its also support the CUDA programming which allows the programmer able to take the advantage of the GPU processor to execute the code. Moreover, the Jetson tk1 board has a low cost and low power consumption comparing to a PC with GPU processor which makes it a good choice for researcher and student working on GPU programming. In this paper we are going to implement rCUDA platform on Jetson TK1 board to use its GPU to execute the code of the remote devices.

Fig-2 Jetson TK1 board

2. rCUDA FRAMEWORK ARCHITECTURE

The rCUDA framework client-server distributed architecture consist of two software modules as shown in the figure(3) below:

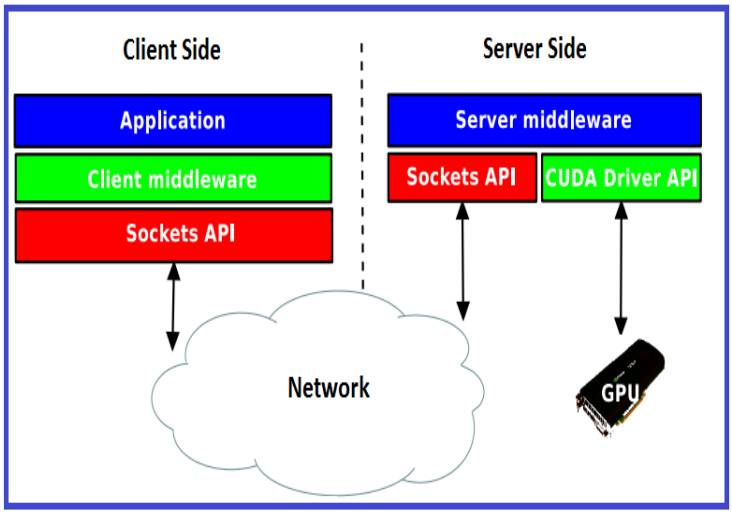

Fig-3 rCUDA Architecture[1]

1. **Client Middleware:** is consists of collection of wrappers responsible for replace runtime library in the client machine with rCUDA libraries. Client middleware also in charge of forwarding the CUDA API calls generated by the applications. At the runtime the client middleware will be responsible to pass the API call from the client machine and it will wait for the result from the server.
2. **Server Middleware:** is installed in the machine which own the GPUs. it will be responsible for receiving , interpreting and execute the API call send by the clients machines. rCUDA platform able to different clients at the same time. to handle more than one client at the same time Server middleware multiplex the access to the GPU processor between the clients, And for effective multiplexing rCUDA integrated with SLURM scheduler which is an open source job scheduler to gives more efficient throughput.

The communication between the client and the server it will be through the network. To establish the communication between the server and the client we can use one of two protocols such as TCP or INFINIBAND. TCP protocol used for Ethernet connection. But, INFINIBAND protocol is used for high speed connection.

3.IMPLEMENTING rCUDA ON JETSON TK1:

To implement rCUDA on the system first you have to download it from www.rcuda.net. For each version of CUDA we have to use specific rCUDA version. For Jetson TK1 there are a special version of rCUDA called 15.07. this version support CUDA version 6.5.

A. Equipment Used In The Experiment

In this experiment we use Jetson TK1 as the server with the previous specifications, and in client side we are going to use laptop with the following specifications:

- CPU Intel Core i5 520M / 2.4 GHz.
  - 4GB DDR3 Memory.
  - 10/100/1000 Gigabit Ethernet network interface adaptor.
  - OS Ubuntu 14.04 32-bit.

B. Experiment Description.

In this demo we are going to use rCUDA v15.07 along with CUDA 6.5,and for commutation between the client and the server TCP protocol are used. The demo can be divided into two parts:

1. Server Side:

After downloading the rCUDA files,first Decompress the rCUDA package using any decompress software. Then copy the rCUDA folder to the machine server. To run rCUDA on the server first the environment should be prepared using export command. Export command will add the variable to environment variables of a shell. This variables are pass to child processes which we can use it to run the programmes. in this scenario LD_LIBRARY_PATH variable will be pointed to the location of the CUDA libraries which is typically located in "/usr/local/cuda/lib64" and if cuDNN libraries are going to used it should also added to LD_LIBRARY_PATH variable. After adding the path of CUDA libraries rCUDA require to mention number of GPUs in the system, and this done by export RCUDA_DEVICE_COUNT=1 command. Finally after preparing the environment and assigning the number of the GPUs in the system we can run the rCUDA binary file by ./rCUDAd command. when the rCUDA run it will show the following output figure(4).

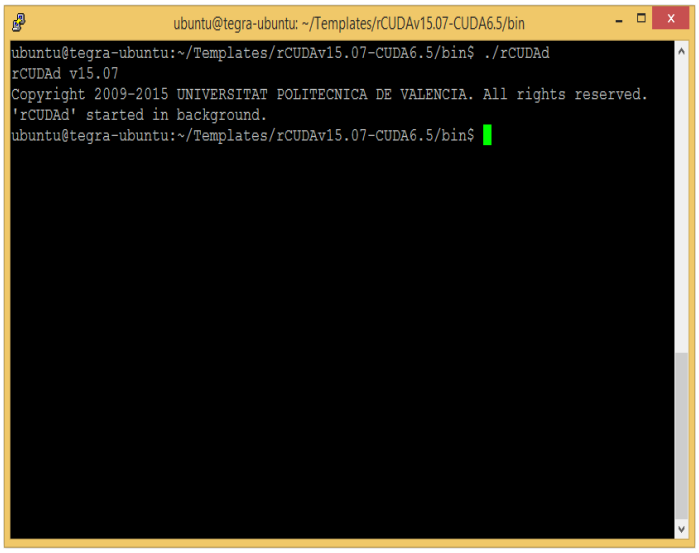

Fig-4 rCUDA server running.

At this point the server will start listening to incoming call from the client, for that the client and server should be in the same network to be able to communicate. For debugging purpose we can run rCUDA in another way where we can see the communication and passing calls between the server and the client.

2. Client Side:

As mentioned previously the rCUDA client is consist of wrappers replace the CUDA runtime libraries, in this way the application which use rCUDA will not be aware about using a remote GPU so that no need any modifications in the codes. rCUDA client 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." [7] those files are copied in the clients machines.

To run the code in remote GPU first CUDA compiler should be installed in the client machine. CUDA compiler will generate the binary file of the code. After generating the binary file from the compiler In order to properly execute the applications using the rCUDA library we have to set the following variable:

- Export LD_LIBRARY_PATH: which point to the location of the rCUDA libraries.
- RCUDA_DEVICE_COUNT: indicate the number of GPUs available in the remote server
- Export RCUDA_DEVICE_0=192.168.x.x: contains the IP address of the server in the network.

When rCUDA works properly it will show the normal output of the application.

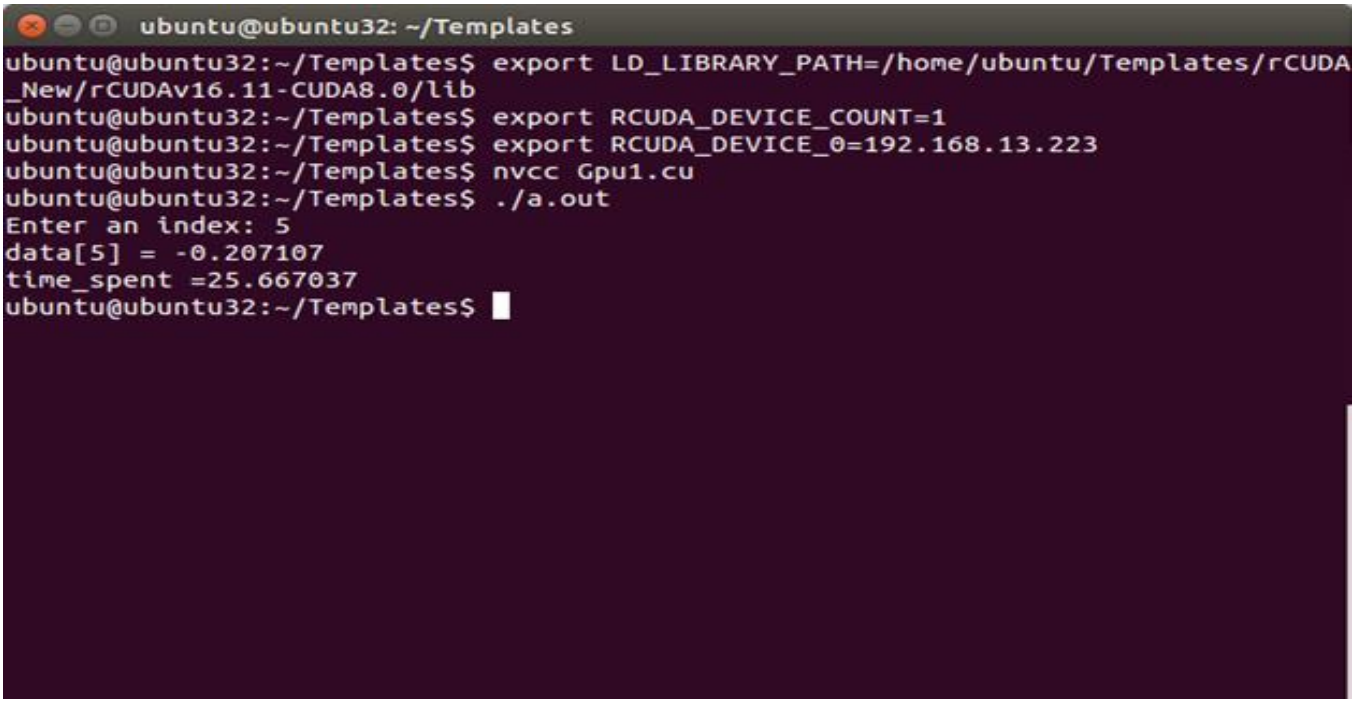

Fig-5 the output of the code.

3. CONCLUSION AND FUTURE WORK

In this paper rCUDA framework has been implemented in Jetson TK1 board which is an embedded board has CPU and GPU processors. After implement the rCUDA framework we become able to access the GPU from the remote PC. the remote application it will able to use the GPU of Jetson TK1 to execute the code.in the future work we going to measure the overhead present on the system when we use rCUDA framework to execute the applications.

6. References

[1]. C. Reaño, F. Pérez and F. Silla, "On the Design of a Demo for Exhibiting rCUDA," 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Shenzhen, 2015

[2]. F. Silla, J. Prades, S. Iserte and C. Reaño, "Remote GPU Virtualization: Is It Useful?," 2016 2nd IEEE International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB), Barcelona, 2016

[3]. M. S. Vinaya, N. Vydyanathan and M. Gajjar, "An evaluation of CUDA-enabled virtualization solutions," 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, Solan, 2012

[4]. J. Duato, A. J. Peña, F. Silla, R. Mayo and E. S. Quintana-Ortí, "rCUDA: Reducing the number of GPU-based accelerators in high performance clusters," 2010 International Conference on High Performance Computing & Simulation, Caen, 2010.

[5]. C. Reaño and F. Silla, "A Performance Comparison of CUDA Remote GPU Virtualization Frameworks," *2015 IEEE International Conference on Cluster Computing*, Chicago, IL, 2015

[6]. NVIDIA, NVIDIA CUDA C Programming Guide 6.5, 2014

[7]. rCUDA v16.11 User's Guide November, 2016