Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Spring Term 2013

ADVANCED COMPUTER NETWORKS
Assignment 8: Introduction to RDMA Programming

Assigned on: **2 May 2013**
Due by: **17 May 2013, 23:59**

1 Introduction:

The goal of this assignment is to give an introduction to RDMA programming [1, 2] and related software [3]. This assignment should be done individually, not in groups. This exercise assumes basic knowledge of Unix. Start with downloading the assignment handout tarball we provide from the website and extract it into a folder of your choice.

2 Setting up the framework for development:

In this section we provide the detailed instructions to install and configure the software/packages needed to solve this exercise. For this programming assignment, you will need a 64-bit Linux system (Ubuntu, Debian) with a kernel that is version 2.6.36.2 or newer. One option can be installing an image inside VirtualBox (http://www.virtualbox.org/). We have tested our solution using various configurations: a 64-bit system running Ubuntu (11.04 - the Natty Narwhal) and 64-bit Ubuntu image running inside VirtualBox. Kernel versions varied between 2.6.36 and 2.6.39.

Create your application directory:

As the first task you need to create a folder in which the application you will be working with is going to run. In our case, this folder is going to be named `/home/eucan/local`. Whenever you see an occurance of this directory throughout this document, replace it with your own application folder.

2.1 Installation of RDMA related packages

Here we describe the steps required to get the RDMA framework ready for development. In oder to see all RDMA related packages type: `apt-cache search rdma`

Among these packages the necessary ones to be installed for this assignment are the following:

```
ibverbs-utils, libibverbs-dev, libibverbs1, librdmacm-dev, librdmacm1, rdmacm-utils
```

In addition, make sure that you need the commonly used linux build tools such as:

```
libtool, autoconf, automake
```

You can install these packages using `sudo apt-get install` command.

2.2 Building the SoftiWARP kernel driver and user library

Find the softiwarp.tar file inside the assignment folder and extract it to a folder of your own choice. Build the SoftiWARP kernel driver as follows:

- Go into the `kernel/softiwarp` that exists under the directory you extracted the softiwarp code. Build it using:

  ```
  make clean
  make
  ```

Then go to the `userlib/libsiw-0.1` directory inside your extracted softiwarp directory and build the SoftiWARP user library as follows:

```
./configure --prefix=/home/eucan/local
make clean
make install
```

(Make sure to use your own application directory with the `--prefix` parameter.)

Then, set the LD_LIBRARY_PATH with the command:

```
export LD_LIBRARY_PATH=/home/eucan/local/lib
```

You have to make sure every terminal window you open to run the benchmark application (which will be described later in this document) has to have this variable set. Eventually you might want to put this in your bash profile.

Make sure that you have the `siw.driver` file placed correctly in your system's `/etc/libibverbs.d` directory. If you do not find this file in that directory, you need to run the following commands:

```
sudo mkdir /etc/libibverbs.d
sudo cp /home/eucan/local/etc/libibverbs.d/siw.driver /etc/libibverbs.d/
```

2.3 Installation of the Application

- Find the application tarball inside the assignment folder and extract it.
- Browse to the application directory and execute the following commands one by one:

  ```
  ./autoprepare.sh
  ./configure --prefix=/home/eucan/local
  make clean
  sudo make install
  ```

2.4 Installation of the udev rules file for Ubuntu systems

On Ubuntu systems, you will need to install a new udev rules file in order to be able to use the packages and execute the application properly.

- Find the `90-ib.rules` file inside the assignment handout folder.
- Copy this file to /etc/udev/rules.d directory in your system using the following command:
  `sudo cp 90-ib.rules /etc/udev/rules.d/`
- Reboot your system in order for this modification to take effect.

2.5 Installation of the generated modules

Now you need to install the SoftIWARP modules into the system. For this we provided a script called `addmodules.sh` in your assignment folder.

- Find `addmodules.sh` in your assignment folder and make it executable (if it is not already) using the `chmod +x` command.
- Open this file and edit the path of the siw.ko module to point to your own siw.ko file inside `softiwarp/kernel/softiwarp`.
- Execute the script using the command `sudo ./addmodules.sh`. If you experience any permission problems with the script, try browsing into the `softiwarp/kernel/softiwarp` directory, and install the module directly using `sudo insmod siw.ko`

At this point you should be able to see the siw module installed in your system. You can verify this by doing `lsmod | grep "siw"`.

2.6 In case of problems:

In general, if you are having trouble until this point, make sure that you have the following things done right:

- Installation of all the necessary RDMA packages.
- `LD_LIBRARY_PATH` is set correctly. Verify using `echo $LD_LIBRARY_PATH`
- `siw` related modules are installed. Verify using `lsmod | grep "siw"`
- `strace` command can also be useful for debugging. Do `man strace` to see how to use this command.

If nothing else works, email Patrick or Ercan about the problem.

3 Implement RDMA Client/Server application:

In this section we describe the main task that you need to do in the context of this assignment. We start with giving brief information about the benchmark application that you will be working with.

3.1 Using the applauncher:

Your task is to complete the missing functions of the Applauncher. Applauncher is a simple benchmark program that makes use of the RDMA technology for transferring memory blocks between a client and a server which can potentially live on different computers. It is currently is set up to run on the same machine. Browse to the local bin folder (`/home/eucan/local/bin`). You will find an executable file called `applauncher`. Applauncher can work in two different modes (server and client) and can be provided different parameters at the launch time. For example, the server can be started using the following command line (you need to start the server before you start the client):

```
./applauncher -t cpp-rdma-server -a 127.0.0.1 -s 1000 -k 1 -l info
```

The flags provided to the application are explained below:

- *-t*: the mode that the application should be run in.
- *-a*: address of the server / client.
- *-s*: size of the memory being transferred using RDMA.
- *-k*: number iterations the benchmark should be run.
- *-l*: logger level.

If you have set up your framework properly, then the server should be able to start and then print some messages and exit with a message saying:

```
RDMA server initialization successful. Exiting. Remove this part of
the code if everything until this point has been successful.
```

This is a checkpoint. If you see this print out, it means that your RDMA framework has been installed and configured correctly. When you get to this point, find and remove the exit(0) statement from the server code(RdmaServer.cpp) and continue implementing the missing bits of the code. The client application can be started as follows once you have implemented the missing bits of the benchmark:

```
./applauncher -t cpp-rdma-client -a 127.0.0.1 -s 1000 -k 1 -l info
```

3.2 Implement the missing functions of the application

The missing portions of the code that you need to implement are inside the files *RdmaServer.cpp, RdmaClient.cpp*, and *DataPlaneRdma.cpp*. These portions are marked using `##IMPLEMENT##` in the files. The RdmaServer/RdmaClient can be run in two modes "write" and "read", which can be set using a command line argument.

For a general introduction to Infiniband Architecture (Queue Pairs and CA Operation, etc) you can refer to section 42.3.7 of the *inf_chap42.pdf* included in the assignment tarball.

The benchmark application is part of a larger project but the files of interest in the context of this assignment are the following:

Applauncher.cpp, Applauncher.h: This is the driver of the benchmark application. It parses the command line arguments and launches the program.

RdmaServer.cpp, RdmaServer.h: The server side of the benchmark program.

RdmaClient.cpp, RdmaClient.h: The client side of the benchmark program.

DataPlaneRdma.cpp, DataPlaneRdma.h: These files implement some of the RDMA related calls that are used by the benchmark.

You need to understand how to use ControlUtils, ReadyToReceive, ByteBuffer classes within the implementation of the benchmark.

Some of the RDMA related system calls you will need to use in order to implement the missing portions of the benchmark code:

ibv_reg_mr, ibv_dereg_mr, *ibv_post_send*, *ibv_post_recv*, *ibv_req_notify_cq*, *ibv_poll_cq*

For more information on how these calls work, refer to the man pages (such as `man ibv_reg_mr`).

Once you have correctly implemented the missing parts of the code, the client side should be able to connect to the server and they will run the benchmark and output the computed throughput values.

4 Hand-In Instructions

- You are going to use SVN (`http://subversion.apache.org`) to handin your assignments for this course. Make sure you have it installed on your computer. On Ubuntu, you can install it by typing 'sudo apt-get install subversion'.
- We have set up a directory for each student in the course and the directories are named according to NETHZ User IDs. Clone your SVN folder to your computer using the following command (Replace the NETHZ_USERNAME with your own NETHZ User ID and enter the whole command in a single line).

  ```
  svn co --username NETHZ_USERNAME
  https://svn.inf.ethz.ch/svn/systems/acn13_students/trunk/NETHZ_USERNAME
  ```

- Create a new directory called `assignment8` inside your directory.
- Copy your solution file into the newly created assignment folder and then type:

  ```
  svn add assignment8
  ```

  This will add your assignment folder and its contents into to your working copy and schedule them for addition to the SVN repository. They will be uploaded and added to the repository on your next commit when you type:

  ```
  svn commit -m "checking in assignment 8"
  ```

  Note that the string you provide after the -m switch does not have to be the same as the one provided here. It is intended to be a message for keeping a history on how the document evolves.

- After the handin, if you discover any mistakes in your solution and want to submit a revised copy, place your new solution files into the assignment folder and type:

  ```
  svn commit -m "checking in revised assignment 8"
  ```

- For more information about how to use SVN, start with typing `svn --help`

References

[1] RDMA Consortium. `http://www.rdmaconsortium.org/`.

[2] Work on RDMA host software at IBM Research Zurich. `http://www.zurich.ibm.com/sys/rdma/`.

[3] SoftiWARP: Software iWARP kernel driver and user library for Linux. `http://gitorious.org/softiwarp`.

Appendix

Figure 1 shows the interaction between server and client for the two modes: read mode and write mode.

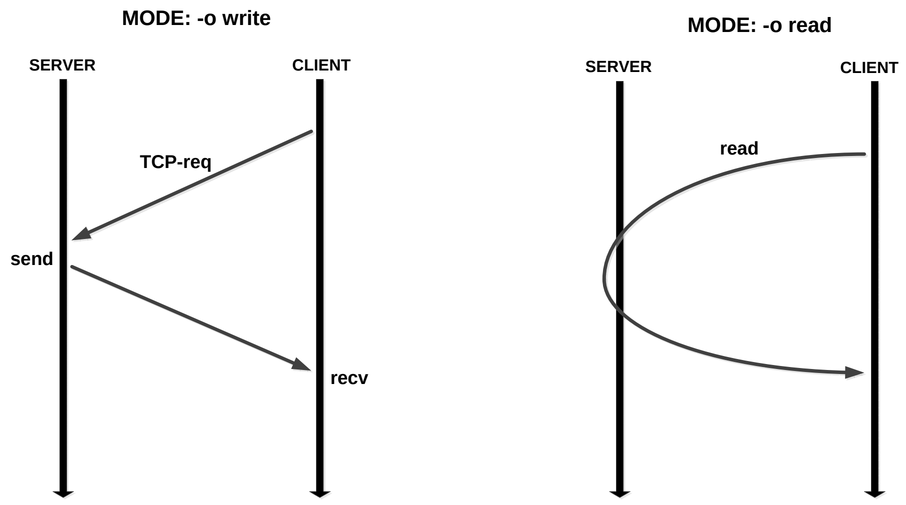

Figure 1: The interaction between server and client in read and write modes.