CMU 18-746 | **Storage Systems** | **Assigned: 08th Nov. 2006**

Fall 2006 | **Project 3: Implementation of an iSCSI Controller** | **Due: 7th Dec. 2006**

1 Introduction

In this project, you will explore explore issues related to fault tolerance by implementing a disk controller for the iSCSI protocol. Your controller will support both replication and RAID-5 modes and will be capable of talking to an arbitrary number of iSCSI devices.

Before starting this project, you should familiarize yourself with *SAN*s, the *iSCSI* protocol, and *RAID-5*. The following readings might prove useful. Note that some of these readings were assigned in class.

- **RFC 3720: Internet Small Computer Systems Interface (iSCSI)**. J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner. April, 2004.
- **The advent of SANs**. John W. Togio. In *The Holy Grail of Storage Management*. Prentice Hall, PTR. 1999.
- **RAID: High-Performance, Reliable Secondary Storage**. Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz, David A. Patterson. 1994.
- **RAID Levels**. The PC Guide. Charles M. Kozierok. April 17th, 2001. `http://www.pcguide.com/ref/hdd/perf/raid/levels/index.htm`

2 Provided Infrastructure

We have provided you with starter code in the `intel-iscsi-2.0.16` tarball. You will need access to one linux machine, preferably at least two, to run this code. To get started, untar the tarball into a directory of your choosing and `cd` into it. Next type the following to compile the code:

```
./configure --prefix <path_to_the_intel-iscsi-2.0.16_directory>
make
make install
```

There will now be several executables in the `intel-iscsi-2.0.16/bin` directory. You will have to concern yourself with only four of these executables: `controller`, `udisk`, `fitness`, and `utest`. A description of these executables is provided below.

udisk: This executable simulates the functionality of an iSCSI target (an iSCSI target is a device that receives and stores data). Your controller will store data on these virtual iSCSI targets.

fitness: This executable is an iSCSI initiator (it sends iSCSI requests to iSCSI targets). It can be used to test your iSCSI implementation.

utest: This executable is another iSCSI initiator that can be used to test your iSCSI implementation. Note that `utest` explicitly checks data integrity, whereas `fitness` does not.

controller: The controller is both an iSCSI initiator and an iSCSI target. It receives data from test programs, such as `fitness` and `utest` and distributes this data, using either replication or RAID-5, to all available targets. The functionality of provided `controller` is very limited; it is up to you to fill in the necessary RAID-5 and replication functionality.

`Make install` also copies over `ips.conf` and `targets.conf` from `intel-iscsi-2.0.16/src` to `intel-iscsi-2.0.16/e`
So, if you need to modify these files, make sure you modify the version in `src`.

3 Setting up your SAN

For the purposes of this project, you will simulate a storage area network by starting multiple `udisk` targets.

(a) First, start at least one target. At least three targets are required for RAID-5 and at least two are required for replication. To start a target, follow these steps:

  (a) cd to `intel-iscsi-2.0.16/bin`.

  (b) Type `./udisk -d <file> -p <port_num>`.

  `udisk` without any parameters will start a target that stores data in RAM. the "-d <file>" flag specifies the file in which to store data. The "-p <port_num>"flag specifies the port on which to receive iSCSI commands. If you leave out the "-p<portnum>" flag, the default port will be used. However, note that if you start multiple targets on the same machine, you will have to specify different ports for each target. To see other options, type `./udisk -u`; feel free to experiment with any of these other options.

(b) Next start the controller, which is responsible for distributing data to the targets on the SAN, by following these steps:

  (a) cd to `intel-iscsi-2.0.16/bin`.

  (b) Type `./controller -t <targets file> <-r>`.

  The file <targets file> contains the IP addresses (and port numbers) of the targets started in the previous step. See `intel-iscsi-2.0.16/src/targets.conf` for an example. The <-r> flag tells the controller to use RAID-5 instead of replication.

4 Your Assignment: Implement Replication and RAID-5 functionality

Before writing any code, please be sure to familiarize yourself with the following source files (all of which can be found in the `intel-iscsi-2.0.16/src` directory): `utarget.c`, `disk.c`, `controller.c`, `controller_replication_raid.c`, and `initiator.c`. Also, look at the Makefiles to see how the relevant binaries are built.

4.1 Implement Replication Functionality in the Controller

Modify `controller_replication_raid.c` and `controller.c` to implement replication functionality in the controller. Your replication code must be able to handle up to $num\_targets - 1$ failures. When implementing this functionality, you should think about issues such as parallelism and load-balancing.

4.2 Implement RAID-5 Functionality in the Controller

Modify `controller_replication_raid.c` and `controller.c` to implement RAID-5 functionality in the controller. Your RAID-5 code must be able to handle up to one target failure. When implementing this functionality, you should think about issues such as parallelism, stripe size, and the performance of degraded mode reads and writes.

4.3 Test Your Controller

First, use `fitness` and `utest` to test your controller. These programs are setup so as to run on the same machine as the controller. If you wish to run these programs on different machines, you will have to modify `iscsi-v20-3.0/src/ips.conf` to point to the machine on which the controller is running.

Provided below is a an example instantiation of `fitness`.

- `./fitness --config ../etc/ips.conf --cap 50 --wrsz 100 --rdsz 100 --tid 0`

Also, stay tuned for additional ways to test the code!

5 Deliverables

There are two deliverables for this project:

- Your modified source code
- A report detailing your design choices in implementing replication and RAID-5 functionality in the controller. Don't forget to explain how you think your design choices helped or hindered performance.

.

E-mail both deliverables in a single tarball to your friendly project TA by the due date.

6 Addendum: Tips

(a) Make sure to type `make` and `make install` whenever you compile your code. `Make install` copies the binaries from the `intel-iscsi-2.0.16/src` directory to the `intel-iscsi-2.0.16/bin` directory.

(b) Some debugging aids are provided in `debug.h`. You should use them.

(c) The function `initiator_command()` takes in a timeout as one of its parameters. You should use this parameter intelligently when handling target failures.

(d) The controller issues `read_capacity()` calls to determine the block length and number of blocks exported by each target. In the replication case, the capacity, assuming the block length is the same across all targets, is $num\_targets \cdot num\_exported\_blocks$. For RAID-5, the capacity is different – make sure you change the code accordingly.