PCIe AHCI-IP Demo Instruction

Rev1.0 22-Aug-23

PCIe AHCI-IP Demo Instruction

Rev1.0 22-Aug-23

This document describes the instruction to show PCIeSSD demo by using SATA AHCI-IP, SATA-IP, and PCIeIP connecting with SATA-III/II SSD on Xilinx evaluation kit (VC707/KC705). Through 4-lane PCIe @ Gen2 speed, PC running Fedora21 OS with specific device driver can detect the board to be SCSI device. So, user can write/read data to the board by using general application like typical SATA-III/II device. This demo uses new test application (diskTestApp) to check disk performance.

1 Hardware Requirement

As shown in Figure 2, to run PCIe AHCI-IP demo please prepare

1) Xilinx Evaluation board (VC707/KC705)
2) PC Power adapter included in Xilinx evaluation kit

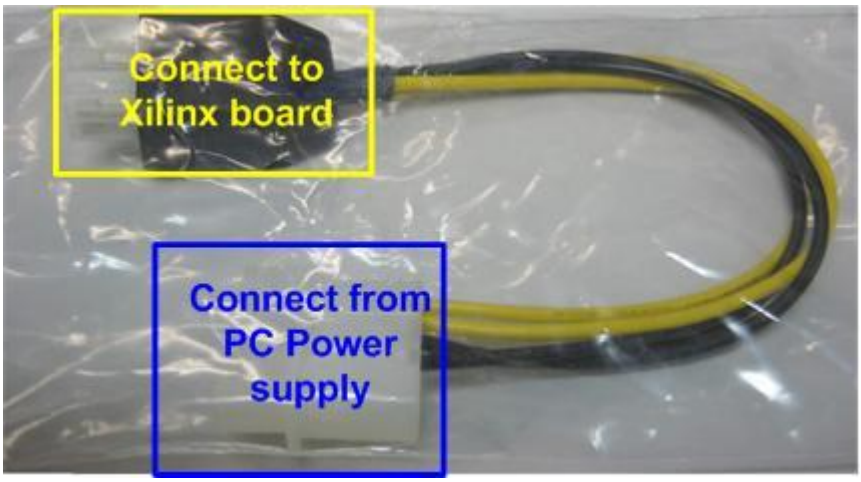

Figure 1 PC Power adapter

3) Micro USB cable for JTAG programming
4) iMPACT ver 14.4 or later to program bit file to the board through JTAG
5) AB09-FMCRAID board, provided by Design Gateway
6) 2.5-inch SATA-III/II Device or other size with adapter cable
7) PC which has available 8-lane PCIe Gen2 and install Fedora21 OS (Linux kernel version 3.18).
   Note: 8-lane PCIe is required to match with PCIe connector size on FPGA board though only 4-lane is used in the design.

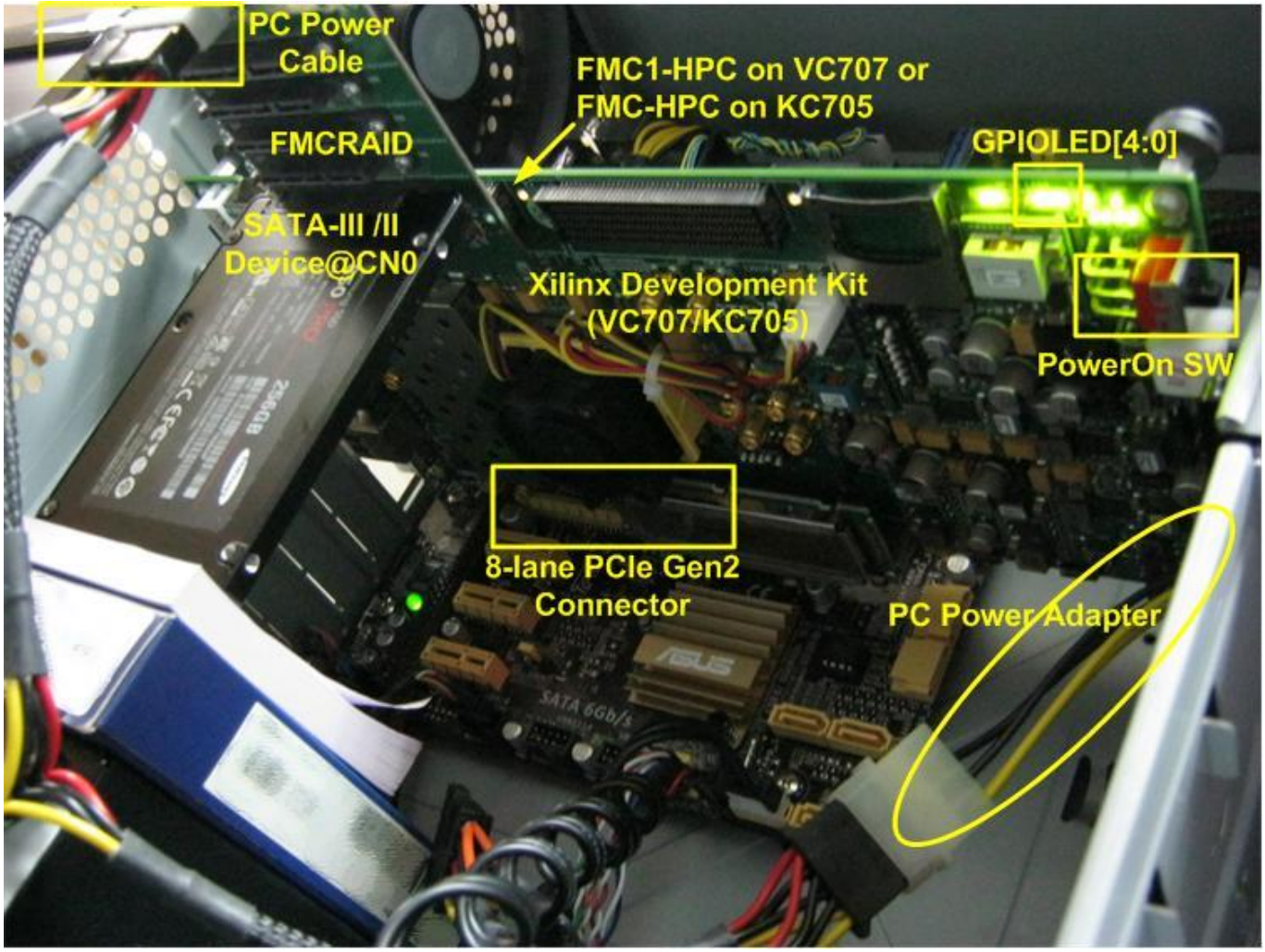

Figure 2 PCIe AHCI-IP Demo Environment Setup on VC707/KC705

2 Hardware setup

- Power off board and PC
- Connect FMC SATA RAID board to FMC1_HPC connector (J35) on VC707 or FMC_HPC connector (J22) on KC705
  Note: FMC SATA RAID board is provided by Design Gateway.
- Connect PC power cable to power connector on AB09-FMCARID board for SATA-III/II device power
- Connect 2.5-inch SATA-III/II Device to CN0 on AB09-FMCRAID board
- Set DIPSW bit [2:1] at SW2 for VC707/SW11 for KC705 to select SATA speed mode. DIPSW Description is shown in Table 1

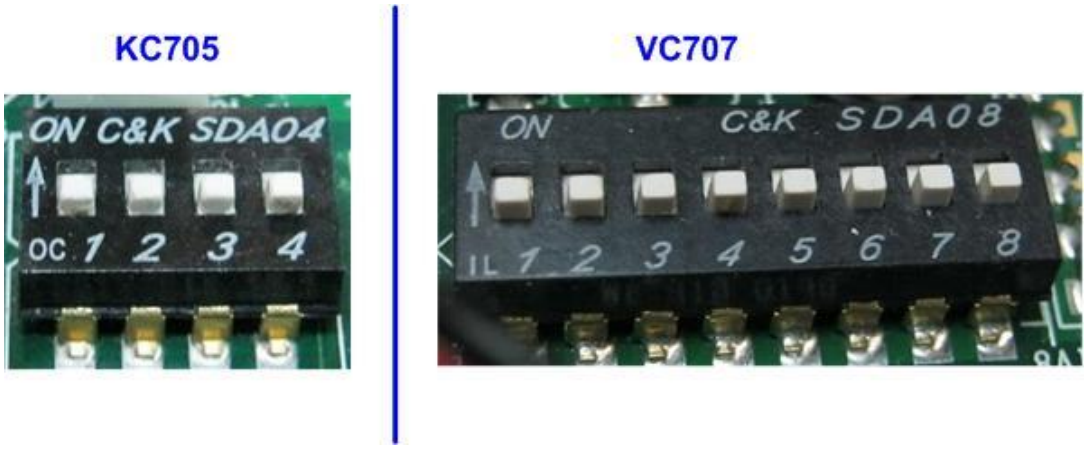

Figure 3 DIPSW to select SATA speed mode

DIPSW[2]	DIPSW[1]	Description
'1'	'1'	Fixed-speed at SATA3 (6.0Gbps)
'1'	'0'	Fixed-speed at SATA2 (3.0Gbps)
'0'	'X'	Auto-speed negotiation mode

Table 1 DIPSW setting description

- Connect USB micro B cable from U26 for VC707/U59 for KC705 to USB Port on PC for JTAG programming
- Connect PC power adapter, provided in Xilinx evaluation kit, between PC power cable and Xilinx evaluation kit
  Warning: Do not use the PCIe connector from the PC power supply to connect to FPGA board

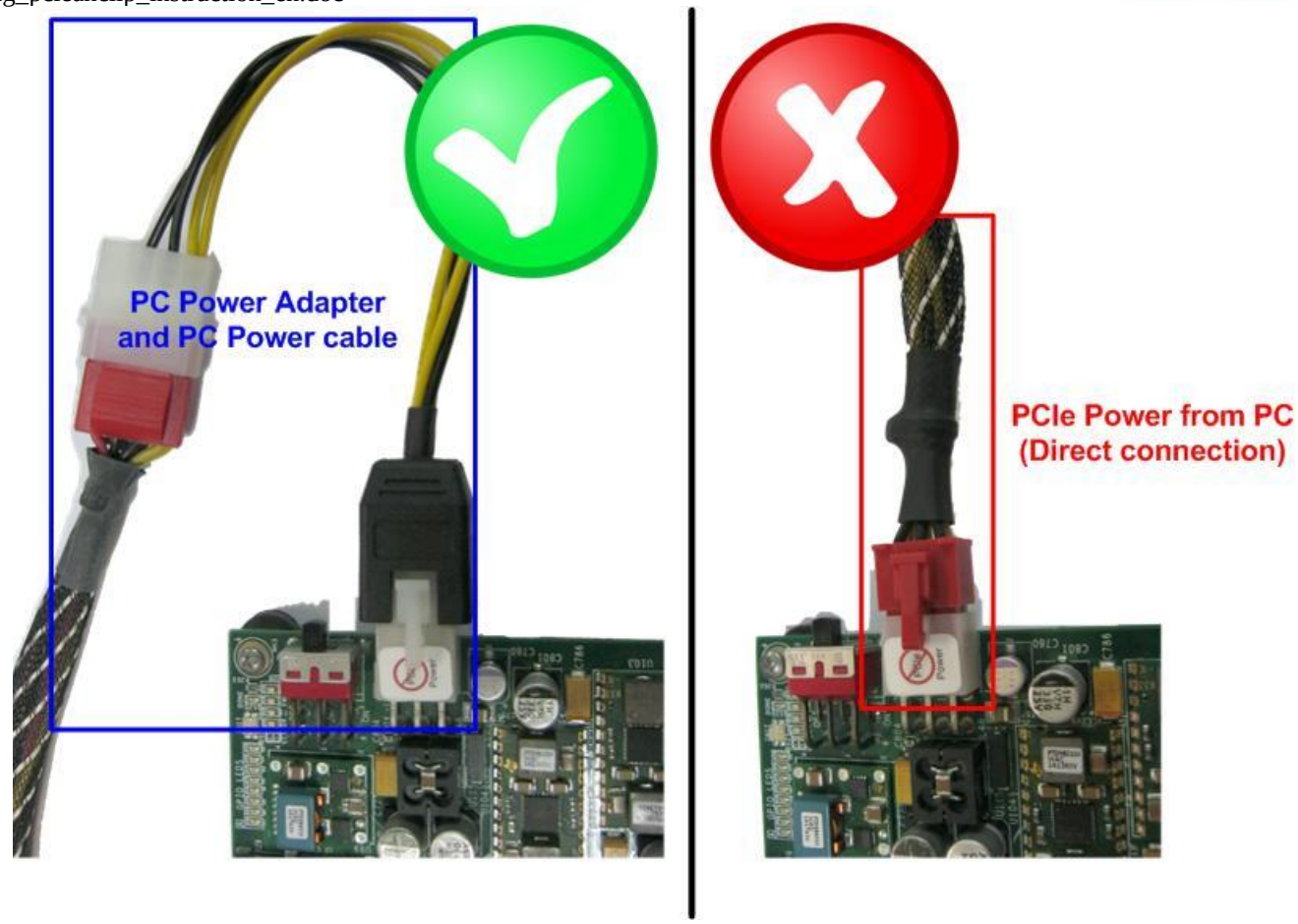

Figure 4 Board power connection through adapter (not direct connection from PC)

- Insert VC707/KC705 board into PC's 8-lane PCIe Gen2 slot
- Turn-on Power switch on FPGA board, and then power up PC
- Open iMPACT and download bit file to VC707/KC705 board.

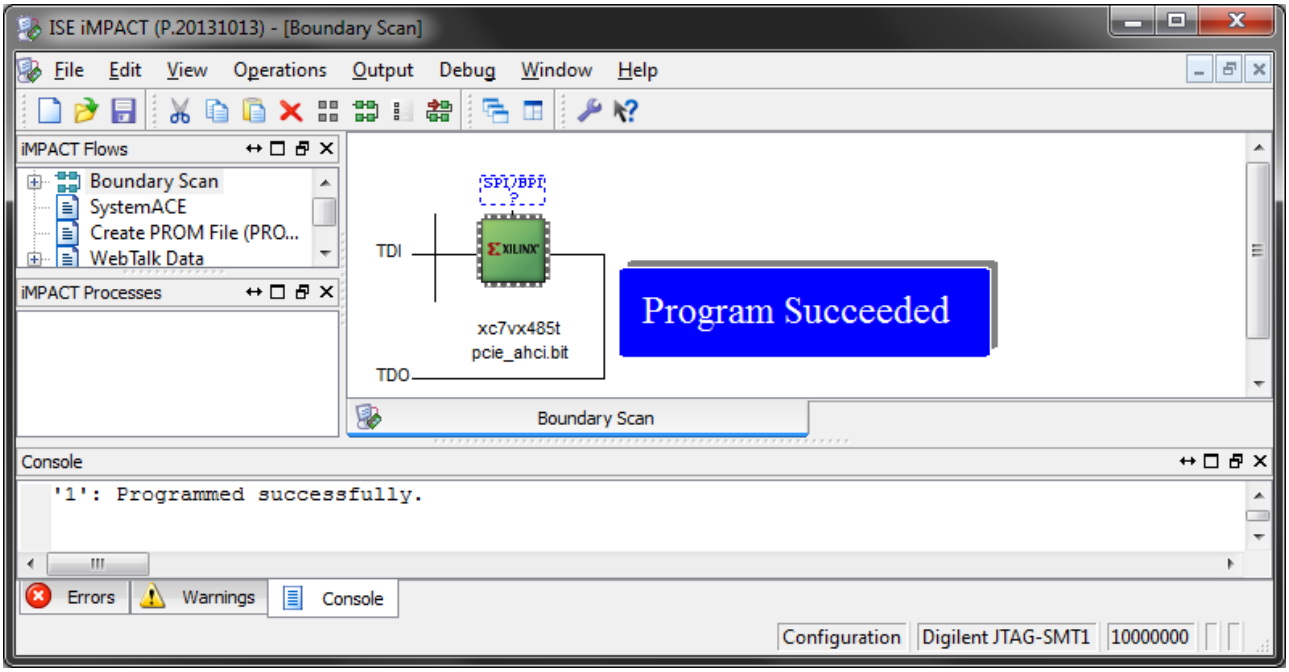

Figure 5 FPGA programming by iMPACT

- Check GPIO LEDs status on VC707/KC705 board at LED0-LED1. Both LEDs must be ON, as shown in Figure 6. LED2 status depends on connecting SATA device speed. Each LED description is described as follows.
  Note: To access hardware register at BAR0 area, the design supports only 1 DW size access. If more than 1 DW is accessed, LED4/LED5 will be ON to show error status.

LED	ON	OFF
LED0	OK	150 MHz of SATA clock on FMC SATA RAID cannot lock. Please check 150 MHz clock source on FMC SATA RAID board.
LED1	OK	SATA-IP cannot detect SATA device. Please check SATA device and the connection.
LED2	Linkup at SATA-III speed	Linkup at SATA-II speed
LED3	SATA in operating	No SATA operating
LED4	Unsupported PCIe write access to BAR0 area	No error
LED5	Unsupported PCIe read access to BAR0 area	No error

Table 2 LED Status of PCIeAHCI reference design

Figure 6 LED status after system set up complete when linkup at SATA-3 speed

- Restart PC to send soft reset and restart PCIe enumeration and configuration. Then, PC can detect the new device.

3 Linux Setup

- Create working directory to store driver and test application file (dg_PCIeAHCI.tar.gz) which can be downloaded from DesignGateway website. In this demo, assumed that working directory is “Home/dg_PCIeAHCI”. Then, extract the file. Three files are provided to run the demo, i.e.
  - dg_libahci.ko : Common AHCI SATA low-level routines
  - dg_PCIeAHCI.ko : AHCI SATA platform driver
  - diskTestApp : Disk Test application to check performance

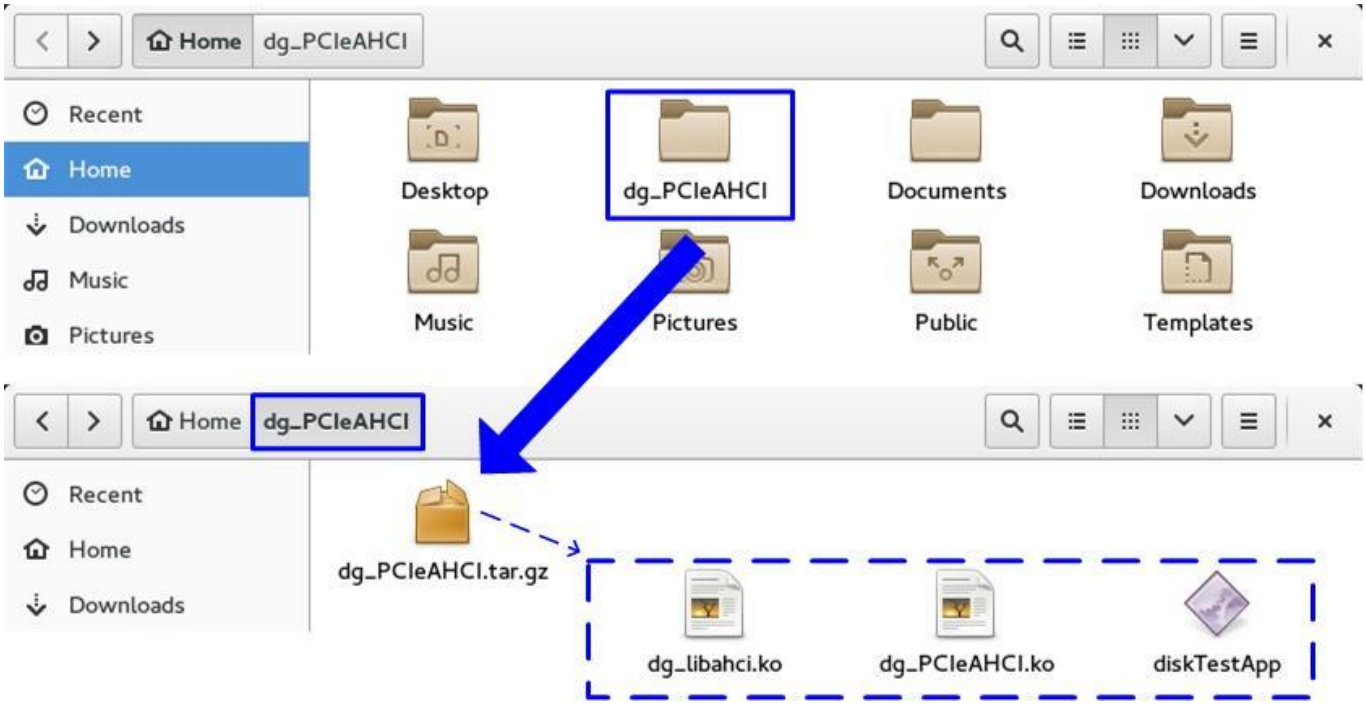

Figure 7 Create working directory on LinuxPC and extract file

- Change current directory to working directory.

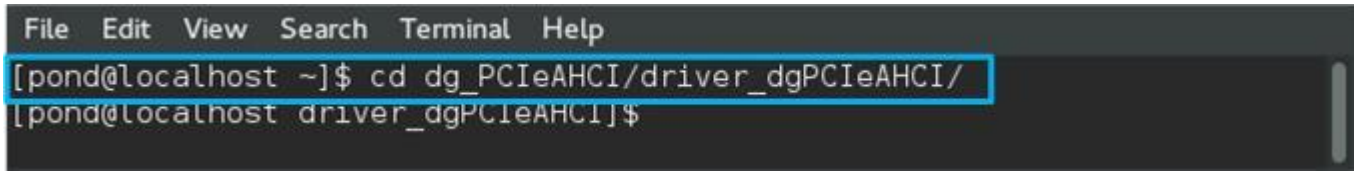

Figure 8 Change current directory

- To insert module, root permission is required. Type “su” to change permission as root.
- Type “insmod dg_libahci.ko” and “insmod dg_PCIeAHCI.ko” sequentially to insert module, as shown in Figure 9.

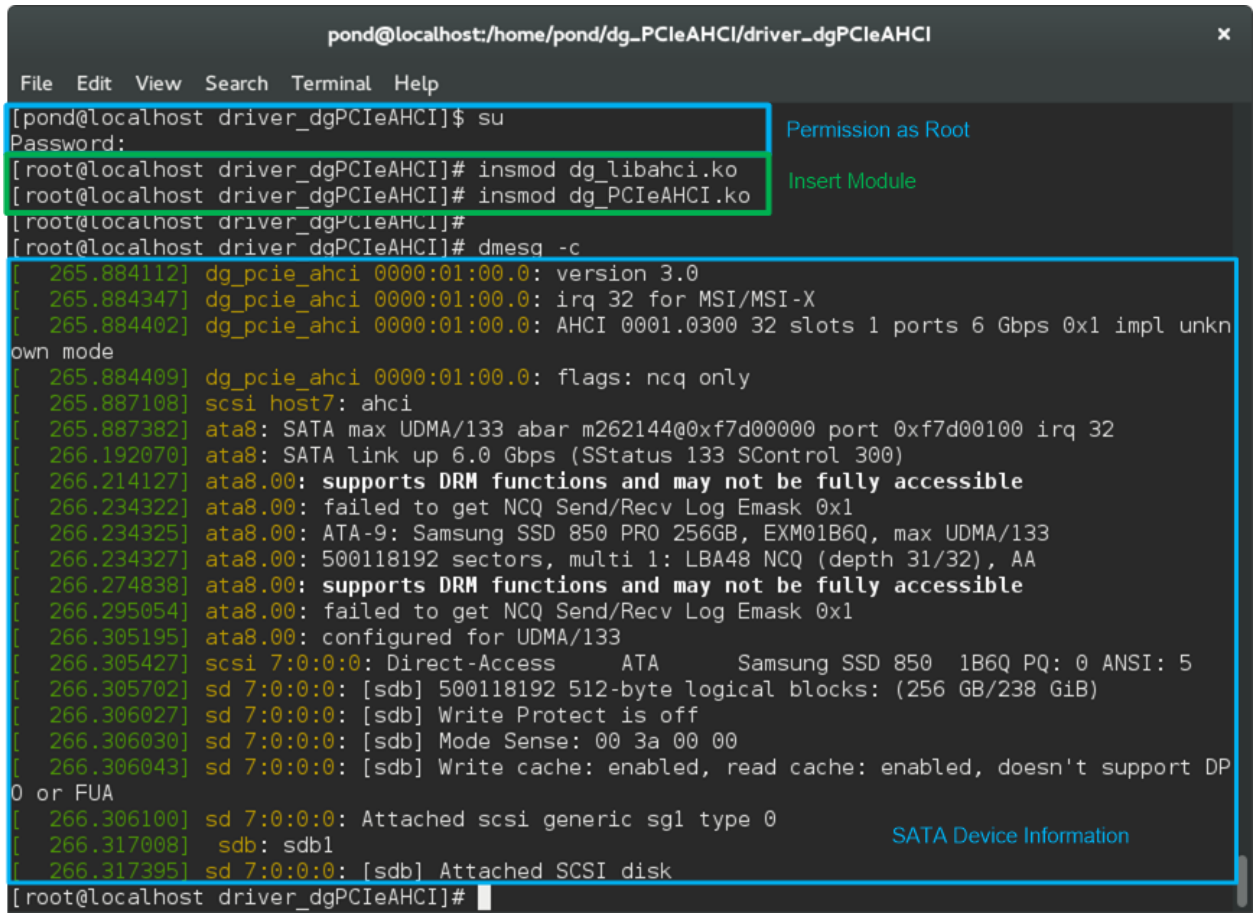

Figure 9 Insert Module

4 Example Disk Command

4.1 Create Disk Partition

As shown in Figure 10, type “fdisk /dev/sdb” to call the tool to start disk management. In the example, new partition (sdb1) is created.

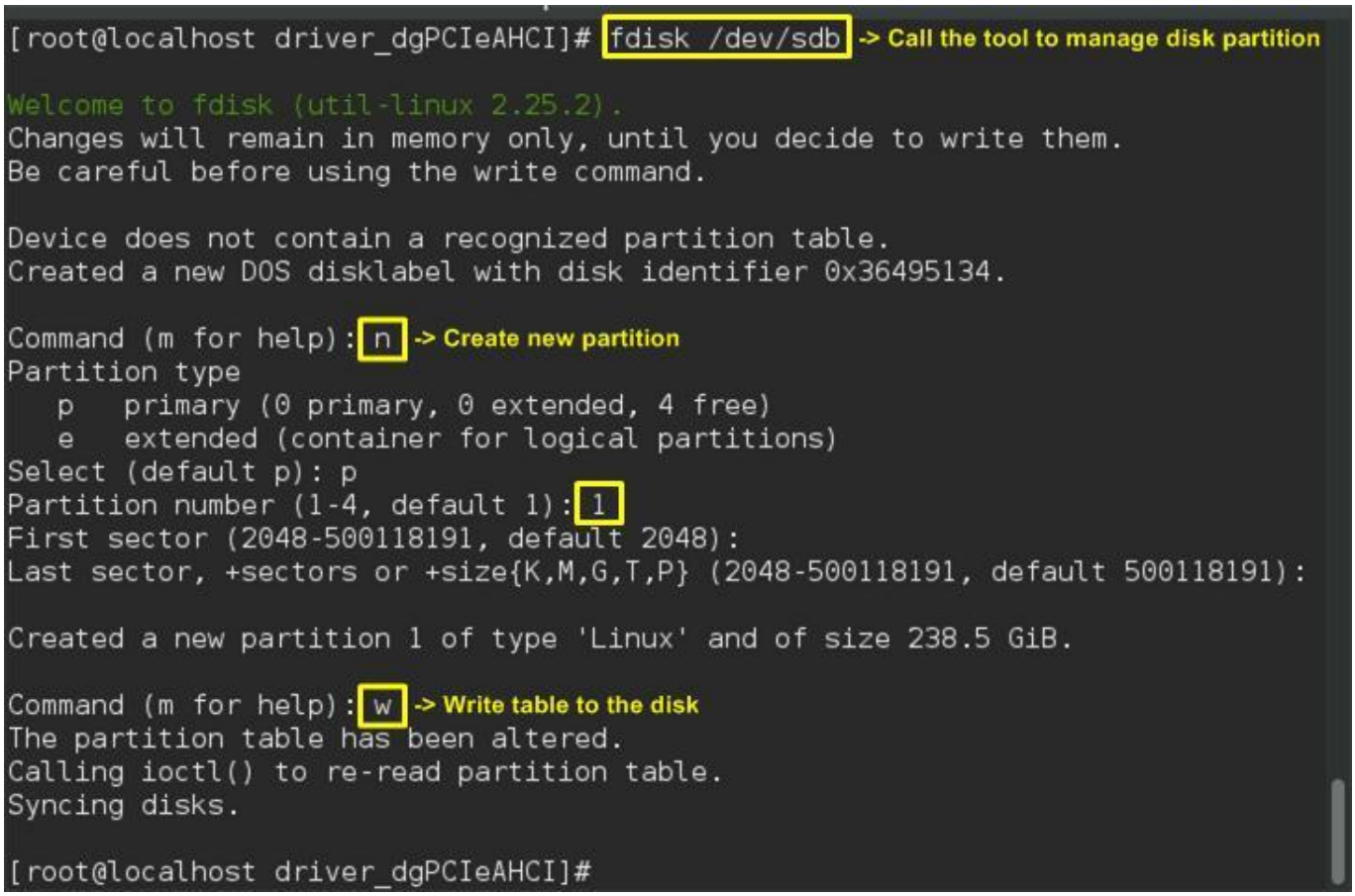

Figure 10 Create Disk Partition

4.2 Format Disk

To format the disk, user needs to select file system type such as FAT, EXT4.
This example shows only the command to format to EXT4 by typing following command.
>> mkfs.ext4 /dev/sdb1

```
[root@localhost driver_dgPCIeAHCI]# mkfs.ext4 /dev/sdb1
mke2fs 1.42.12 (29-Aug-2014)
Discarding device blocks: done
Creating filesystem with 62514518 4k blocks and 15630336 inodes
Filesystem UUID: 9c96da41-7a2b-4ed8-a7d0-4b845c70319a
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,

        4096000, 7962624, 11239424, 20480000, 23887872

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done

[root@localhost driver_dgPCIeAHCI]#
```

Figure 11 Format Disk

4.3 Mount Disk

Before running any application to access the disk such as Test Application or Bonnie++, disk must be mounted firstly by following command.
>> mount /dev/sdb1 /mnt

```
[root@localhost driver_dgPCIeAHCI]# mount /dev/sdb1 /mnt
[root@localhost driver_dgPCIeAHCI]#
```

Figure 12 Mount Disk

5 Performance Test by Test Application

Test application provided by DesignGateway is used in the demo to show disk performance.

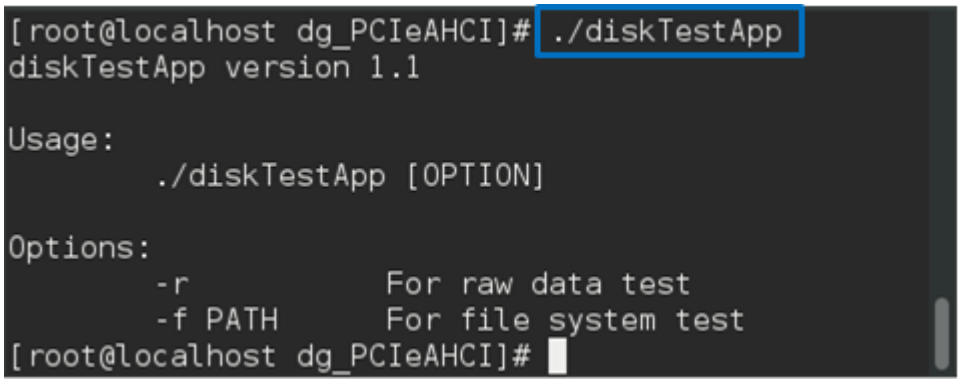

Figure 13 diskTestApp usage

As shown in Figure 13, test application can run in two data formats, i.e. raw data or file system. More details to run test application in each format are described as follows.
***Warning*: If running raw data test, file system in that disk partition will be lost.**

5.1 Run Test Application in Raw Mode

Before run the application, user needs to log in as root. Type "./diskTestApp –r" to run the test application in raw data format. Five input parameters are required in test application, i.e.
1) Disk selection to select the disk to test performance
2) Operation type: '0'-Read disk test, '1'-Write disk test
3) Test pattern:
   - '0': dummy test data for write/no data verification for read
   - '1': 32-bit increment test data for write/verify by 32-bit increment data for read
   - '2': 32-bit decrement test data for write/verify by 32-bit decrement data for read
4) Disk offset: Disk start address in sector unit to run write/read data test. 0x prefix is added to input in hex unit while default value without prefix is decimal unit.
5) Operation length: Transfer length in sector unit to run write/read data test. 0x prefix is added to input in hex unit while default value without prefix is decimal unit.

Figure 13 and Figure 14 show the example of write test while Figure 16 and Figure 17 show the example of read test. Dummy test data will show higher test performance than 32-bit increment pattern for both write and read test because no CPU resource is required to fill or verify test data in dummy mode.

```
[pond@localhost dg_PCIeAHCI]$ su
Password:
[root@localhost dg_PCIeAHCI]# ./diskTestApp -r
// --- --- --- --- --- --- //
// List of disk on system.
0) sdb

Select disk (default 0, hit "Ctrl+c" to exit): 0

Operation type (read(0)/write(1), default 0): 1

Pattern type (none(0)/inc(1)/dec(2), default 0): 0

Disk's offset (0x0-0x00000000_1dcf32af, default MIN): 0x0

Operation length (0x1-0x00000000_1dcf32b0, default MAX): 0x4000000

// *** *** *** *** *** *** //
// Your operation.
Disk: sdb, Opt: Write, Patr: None
Addr: 0x00000000_00000000-0x00000008_00000000

[OK] Writing completed
        @speed = 417.67 MB/s

// --- --- --- --- --- --- //
// List of disk on system.
0) sdb

Select disk (default 0, hit "Ctrl+c" to exit):
```

Write test by 32 GB dummy data

Figure 14 Write Test result by 32 GB dummy data

```
Select disk (default 0, hit "Ctrl+c" to exit): 0

Operation type (read(0)/write(1), default 0): 1

Pattern type (none(0)/inc(1)/dec(2), default 0): 1

Disk's offset (0x0-0x00000000_1dcf32af, default MIN): 0x0

Operation length (0x1-0x00000000_1dcf32b0, default MAX): 0x4000000

// *** *** *** *** *** *** //
// Your operation.
Disk: sdb, Opt: Write, Patr: Increment
Addr: 0x00000000_00000000-0x00000008_00000000

[OK] Writing patterns completed
        @speed = 274.97 MB/s

// --- --- --- --- --- --- //
// List of disk on system.
0) sdb

Select disk (default 0, hit "Ctrl+c" to exit):
```

Write test by 32 GB increment data

Figure 15 Write Test result by 32 GB increment pattern

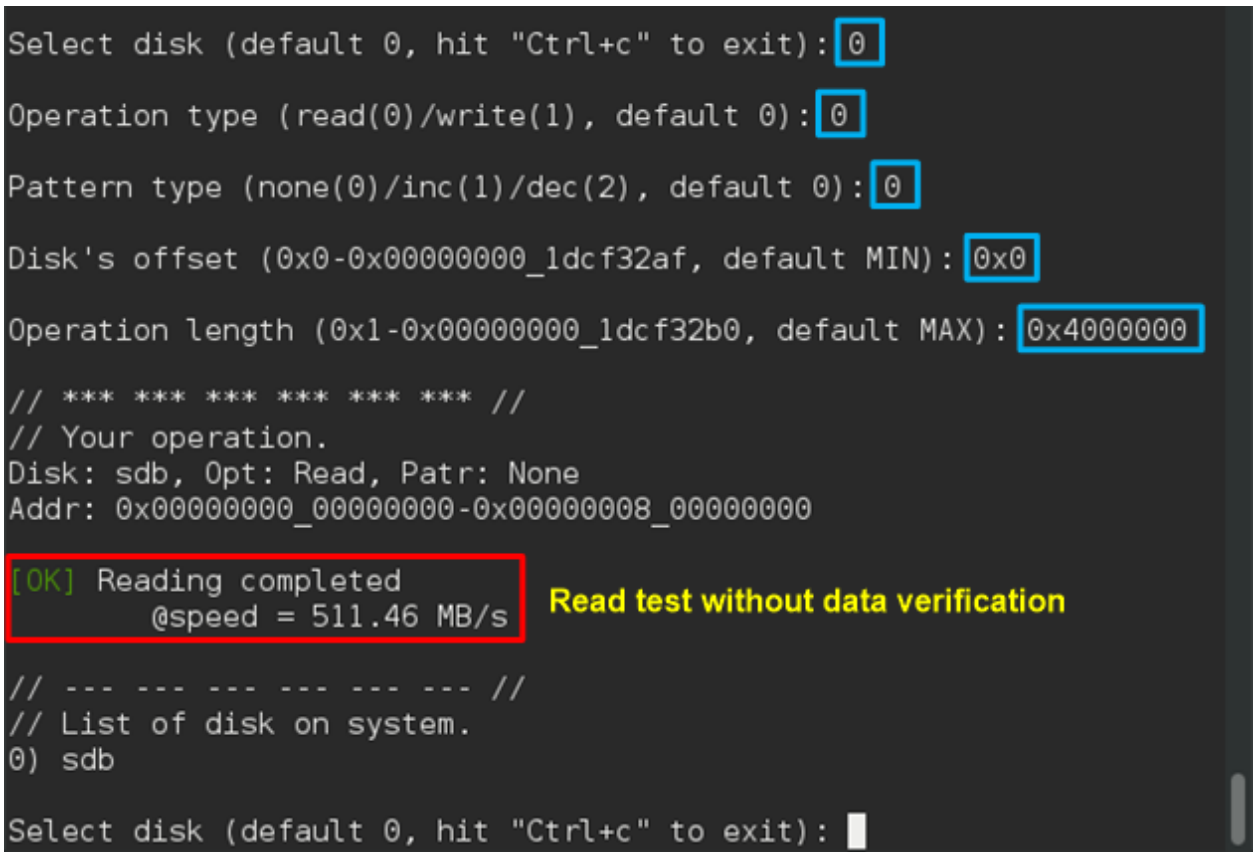

Figure 16 Read Test without data verification by 32 GB size

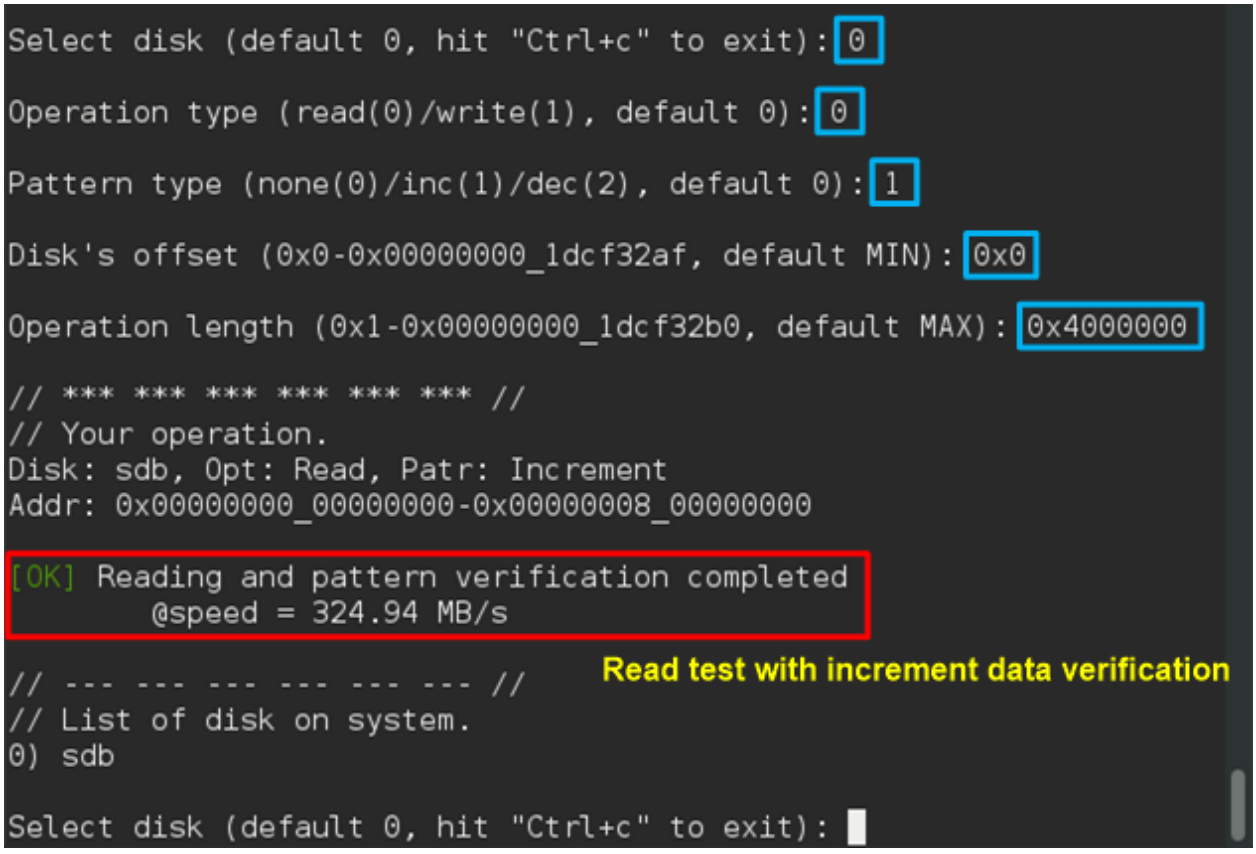

Figure 17 Read Test with data verification by 32 GB size

5.2 Run Test Application in File System Mode

Before run the Application, user needs to log in as root. The disk must have file system to run the test in this mode. Type "./diskTestApp –f <directory>" to run the test in file system format. Five input parameters are required to run test application, i.e.

1) File name input: File name to run the test
2) Operation type: '0'-Read file test, '1'-Write file test
3) Test pattern:
   - '0': dummy test data for write/no data verification for read
   - '1': 32-bit increment test data for write/verify by 32-bit increment data for read
   - '2': 32-bit decrement test data for write/verify by 32-bit decrement data for read
4) File number: Total number of files to run write/read data test
5) File size: Size of one file in sector unit to run write/read data test

Similar to raw data test, dummy mode will show higher performance than 32-bit increment pattern for both write and read test file, as shown in Figure 18 - Figure 21.

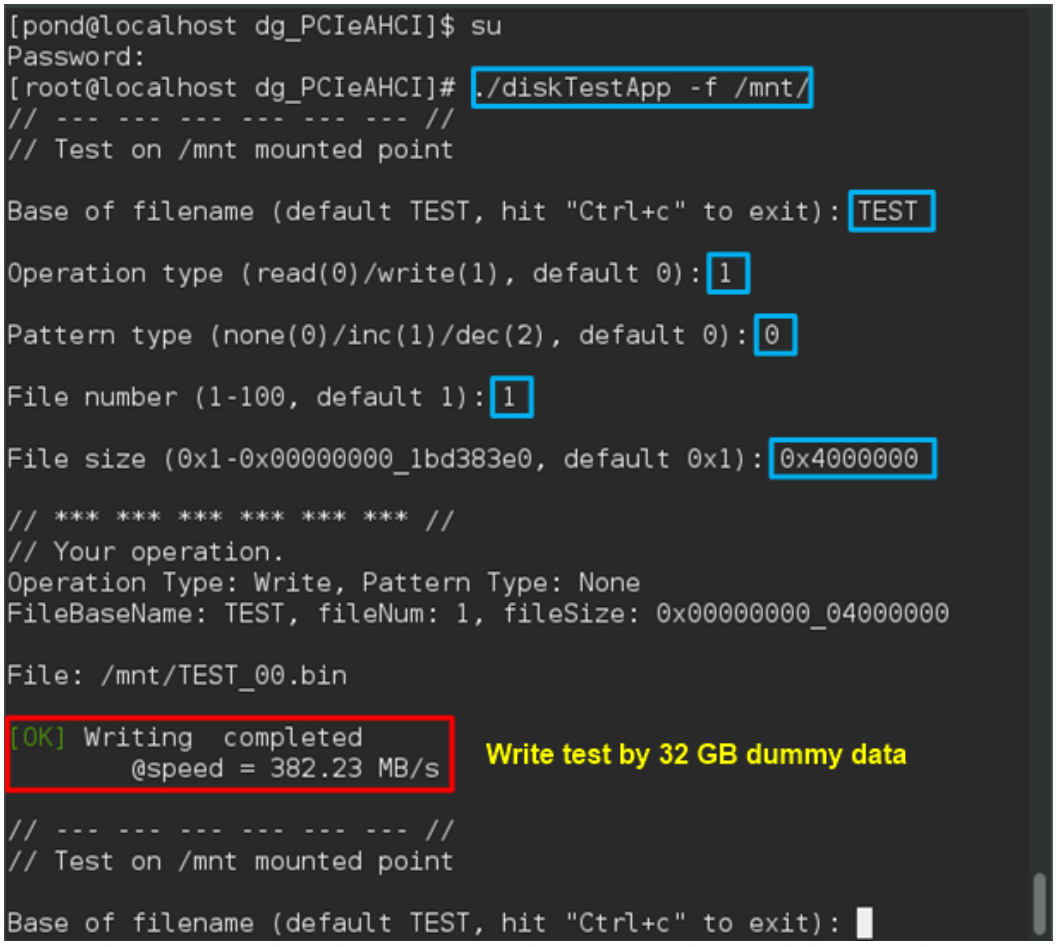

Figure 18 Write File Test result by 32 GB size and dummy data

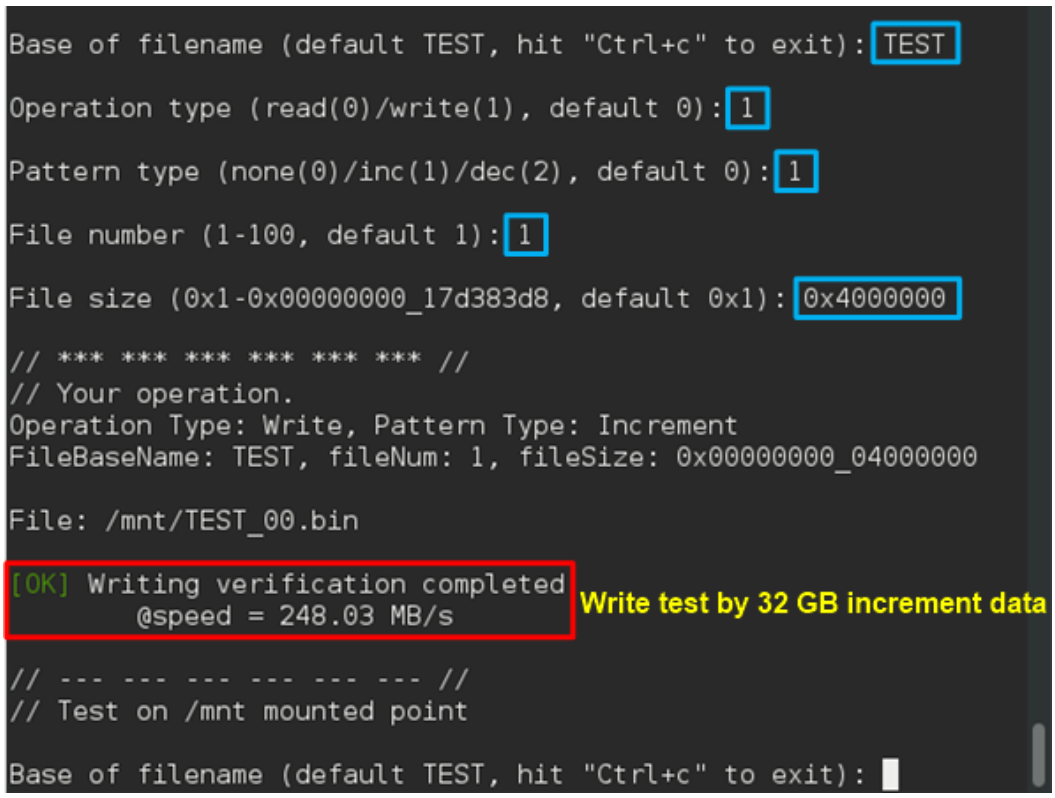

Figure 19 Write File Test result by 32 GB size and increment data

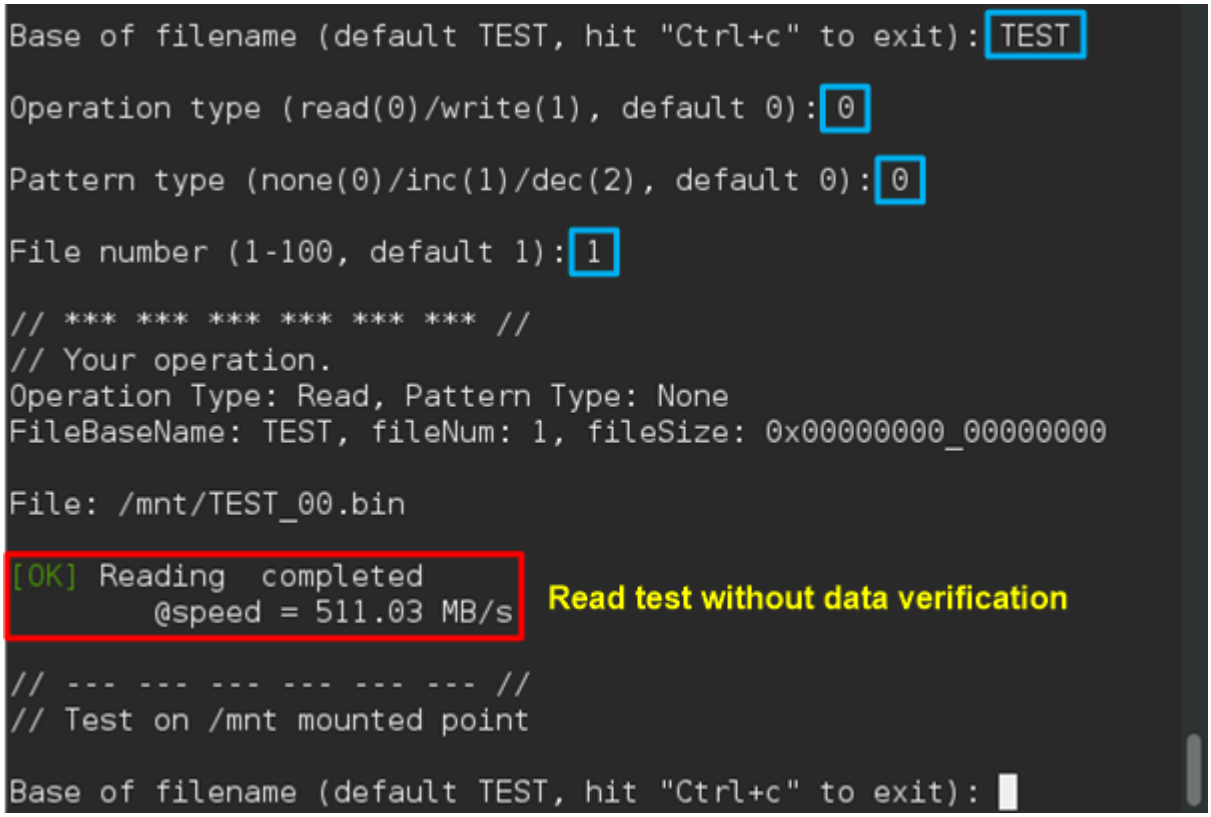

Figure 20 Read File Test without data verification by 32 GB size

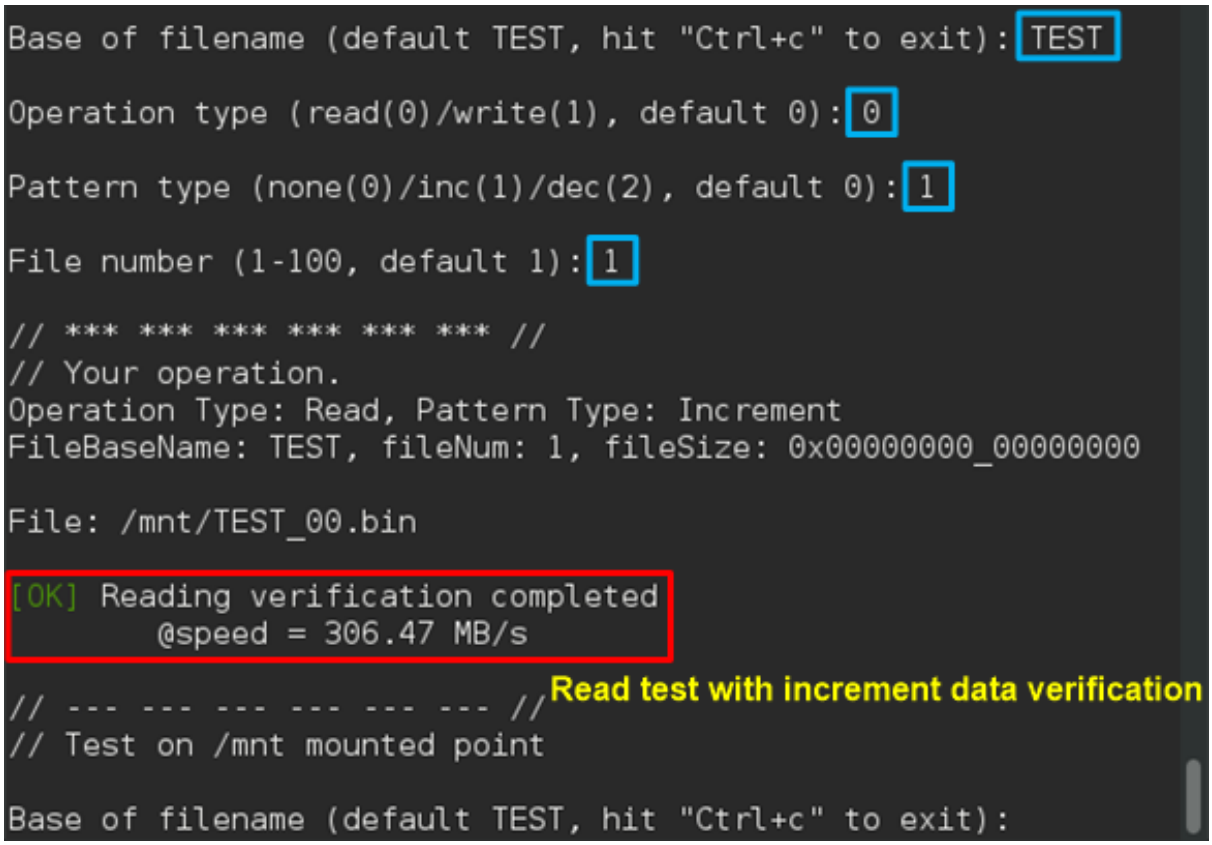

Figure 21 Read File Test with data verification by 32 GB size

5.3 Performance test by Bonnie++ Software

Please see more details about Bonnie++ user manual from http://linux.die.net/man/8/bonnie++.

The example of test result when running by Bonnie++ is shown in Figure 22.

```
[root@localhost dg_PCIeAHCI]# bonnie++ -d /mnt -s 15712M -n 0 -m test -f -u root
Using uid:0, gid:0.
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...done...done...
Version  1.96       ------Sequential Output------ --Sequential Input- --Random-
Concurrency   1     -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
test         15712M           486461  26 121750   9           235140  10  6695 109
Latency                       56360us     275ms              4760us    4085us

1.96,1.96,test,1,1436427961,15712M,,,,486461,26,121750,9,,,235140,10,6695,109,,,,,,
,,,,,,,,,,,,,56360us,275ms,,4760us,4085us,,,,,,
[root@localhost dg_PCIeAHCI]#
```

Figure 22 Test result when running by Bonnie++

6 Revision History

Revision	Date	Description
1.0	10-Jul-15	Initial version release