Block device introduction

Block is a random access device with a certain structure. The reading and writing of this device is carried out according to the block. It uses the buffer to store temporary data. When the conditions are ripe, it can write to the device at one time from the buffer or read to the buffer at one time from the device.

Block device is a parallel concept with character device. There are great differences in the driver structure of these two types of devices in Linux. Generally speaking, block device driver is much more complex than character device driver and shows great differences in I / O operation. Buffer, I / O scheduling, request queue and so on are related to block device driver.

In Linux, the input or output (I / O) operation of the driver to the block device will send a request to the block device, which is described by the request structure in the driver. However, for some disk devices, the request speed is very slow. At this time, the kernel provides a queue mechanism to add these I / O requests to the queue (i.e. request queue), and use request in the driver\_ Queue structure description. Before submitting these requests to the block device, the kernel will first perform the pre operation of request merging and sorting to improve the efficiency of access, and then the I / O scheduler subsystem in the kernel is responsible for submitting I / O requests. The scheduler allocates disk resources to all pending block I / O requests in the system. Its job is to manage the request queue of the block device, Decide the order of requests in the queue and when to dispatch requests to the device.

The generic block layer is responsible for maintaining the relationship between the upper file system and the lower physical disk of an I / O request. In the general block layer, a bio structure is usually used to correspond to an I / O request.

Linux provides a gendisk data structure, which is used to represent an independent disk device or partition to access the underlying physical disk. There is a similar character file in gendisk\_ The hardware operation structure pointer of operations is block\_ device\_ Operations structure.

Some units used in writing block device driver:

1. Sectors: the basic unit of data processing by any block device hardware. Typically, the size of a sector is 512 bytes. (for equipment)

2. Blocks: the basic unit for data processing such as kernel or file system formulated by Linux. Typically, a block consists of one or more sectors. (for Linux operating system)

3. Segments: composed of several adjacent blocks. It is a memory page or part of a memory page in Linux memory management mechanism.

The Linux kernel provides the following elevator algorithms to realize IO scheduling:

1. The no op I / O scheduler only realizes the simple FIFO and the simplest combination, which is more suitable for flash based storage

2. The antibiotic I / O scheduler postpones IO requests (about a few microseconds) in order to sort them for greater efficiency.

The deadline I / O scheduler tries to minimize the delay of each request and reorder the bio. It is especially suitable for situations where there are many reads, such as databases

CFQ I / O scheduler allocates uniform IO bandwidth to all tasks in the system and provides a fair working environment. In the multimedia environment, it can ensure that audio and video can read data from disk in time. It is the default scheduler of the current kernel

We can specify the scheduling algorithm to use by passing parameters from the kernel: kernel elevator = deadline

The current request pointer of each block device and the request Necklace table of the device in the request item array form the request queue of the device. A linked list is formed between items. There are only 32 request items, and all block devices share 32 request items.

Block device access scheduling processing flow

Before sending the operation command to the hard disk controller, first sort the order of reading / writing disk sector data (I / O scheduler processing), so that the disk sector data blocks accessed by the request item are operated in order as far as possible, and are not sent to the block device for processing in the order in which the request item is received.

During data writing: the write command sent by CPU is HD\_ Out(), the allowed write signal DRQ is the data request service flag of the controller status register. When the controller writes all the data to the driver (or sends an error), an interrupt signal is generated, and the preset C function (write\_intr()) is called to check whether there is still data to write. If so, the data of a sector is sent to the driver and wait for the interrupt again.

If all data has been written to the drive, the C function performs the processing after the completion of this disk writing: wake up the relevant processes waiting for the data of the request item, wake up the processes waiting for the request item, release the current request item and delete the request item from the linked list, and release the locked relevant buffer. Then call the request item operation function to execute the next read / write disk request item.

Data reading process: the CPU sends parameters to the controller. The controller reads data from the driver to its own buffer as required, generates an interrupt to the CPU, executes the preset C function (read\_intr()) to send the data in the controller to the system buffer, and the CPU continues to wait for the interrupt signal.

The preset C function first puts the data from one sector in the controller buffer to the buffer in the system, adjusts the current write location in the system buffer, and then decreases the number of sectors to be read. If there is still data to read, continue to wait for the controller to send the next interrupt signal.

For virtual devices, since the synchronization operation with external devices is not involved, the current request item reads and writes to the virtual device completely in do\_ rd\_ Implemented in request().

Judge whether the current request item exists.

Check the validity of the device number and the starting sector of the disk.

Calculate the disk track number, head number and cylinder number of the requested data according to the request item.

If the reset flag is set, the hard disk performs the reset operation.

If the recalibration flag is set, send the hard disk recalibration command to the controller, set the C function (recal\_intr()) to be executed in the interrupt caused by the command in advance before sending, and exit.

If the current request item is a write operation, first set the C function called by the hard disk controller to write\_ Intr(), sends the command parameter block of write operation to the controller, and circularly queries the status register of the controller to determine whether the request for service flag (DRQ) is set. If the flag is set, it indicates that the controller has "agreed" to accept the data, and then the data in the buffer referred to in the request item is written to the data buffer of the controller. If the flag is not set after the circular query timeout, the operation fails. So bad is called\_ rw\_ The intr() function determines whether to give up processing the current request item or set the reset flag to continue processing the current request item according to the number of errors occurred in processing the current request item.

If the current request item is a read operation, set the hard disk controller to call the C function read\_ Intr() and send a disk reading operation command to the controller.

。