FUJITSU THE POSSIBILITIES ARE INFINITE

# IO Controller

Oct/2009

归剑峰
guijianfeng@cn.fujitsu.com

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposals for IO Bandwidth Controlling

# Background knowledge

FUJITSU

- CGroup(Control Group)
  - A mechanism to put tasks into groups
  - Implemented as a pseudo filesystem
  - Automatically inherited at fork() by any children
  - Resouce control is implemented as “subsystem”

# How to use CGroup

FUJITSU

- Mount a CGroup subsystem

  #mount -t cgroup -o subsystem name /cgroup
- Create a group

  #mkdir /cgroup/group1
- Attach a task

  #echo <PID> > /cgroup/group1/tasks
- Destroy a group

  #rmdir /cgroup/group1

# IO Scheduler

FUJITSU

- The goals of IO Schedler
  - Reduce seek and improve global throughput
- The duty of IO Schedler
  - Merge IO requests
  - Sort IO requests
  - Dispatch requests in different policies

# IO Schedulers

FUJITSU

- CFQ(Complete Fair Queuing)
  - One cfq queue for each task(io context)
  - Supports io scheduling priorities and classes
  - Round-robin scheduling
- Deadline
  - Minimize read starvation
- Anticipatory
  - Anticipating the next read reqeust
- Noop

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposals for IO Bandwidth Controlling

# What is IO Controller

FUJITSU

- **It's a CGroup subsystem just like memcg cpuset etc.**
- **It provides propotional bandwidth control**
- **Control the amount of disk time in terms of group weight**
- **The use cases**
  - Provide io BW isolation among Virtual machines
  - Balance io BW among users
  - Limit io BW for a certain application
  - Reduce latency for concurrently running tasks

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposal for IO Bandwidth Controlling

# How to use IO Controller

FUJITSU

- 1 Apply IO Controller patch and compile kernel
  Enable the block layer
  --> IO Schedulers
- 2 Mount IO Controller on a mountpoint
- 3 Create a CGroup directory
- 4 Setup IO Controller by using CGroup interface

```
[root@localhost cgroup]# ls
blkio.id        io.disk_queue   io.disk_time    io.policy  notify_on_release
io.disk_dequeue  io.disk_sectors  io.ioprio_class  io.weight  tasks
```

- 5 Run tasks in that CGroup

# Simple use

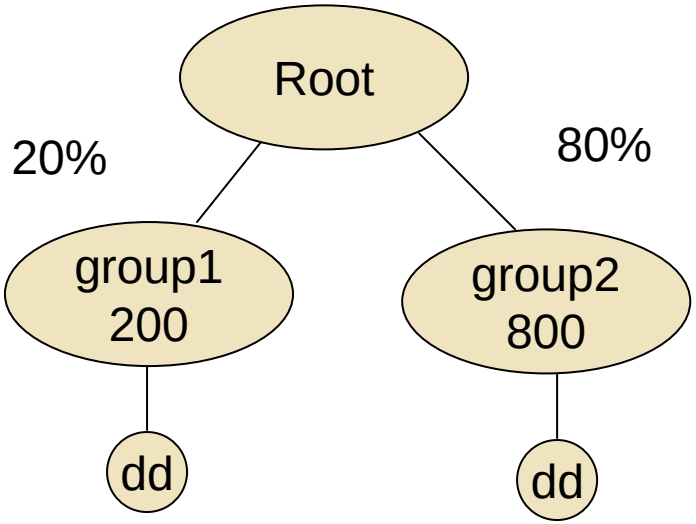
FUJITSU

- mkdir /cgroup
- mount -t cgroup -o io,blkio io  /cgroup

- mkdir /cgroup/group1
- echo 200 > /cgroup/group1/io.weight
- mkdir /cgroup/group2
- echo 800 > /cgroup/group2/io.weight

- dd if=/sdb2/2000M.1 of=/dev/null &
- echo $! > /cgroup/group1/tasks

- dd if=/sdb2/2000M.2 of=/dev/null &
- echo $! > /cgroup/group2/tasks




**group2: 2097152000 bytes (2.1 GB) copied, 36.3494 seconds, 57.7 MB/s**
**group1: 2097152000 bytes (2.1 GB) copied, 57.1376 seconds, 36.7 MB/s**
**sdb disk time usage(io.disk_time) at 10th second:**
**group2: 8:16 14998  group1: 8:16 3825**

# CGroup Interfaces of IO Controller

FUJITSU

- **io.ioprio_class**
  - Denotes class of the cgroup (RT, BE, IDLE). This's the default io class of the group for all device. (1 = RT; 2 = BE, 3 = IDLE)
- **io.weight**
  - Denotes per cgroup weight(1 ~ 1000). This's the default weight of the group for all device.
- **io.policy**
  - One can specify per cgroup per device rules using this interface. Syntax:

    echo "dev_maj:dev_minor  weight  ioprio_class" > /path/to/cgroup/io.policy

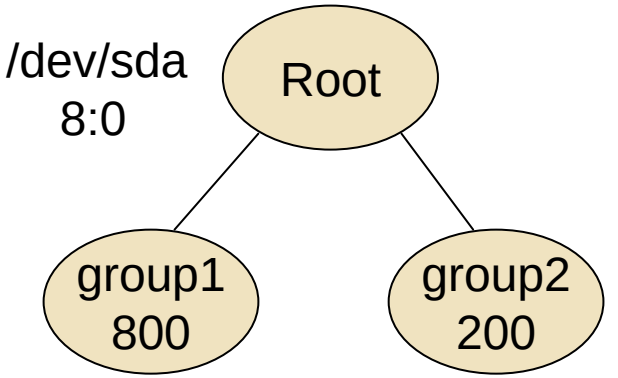
# io.policy example

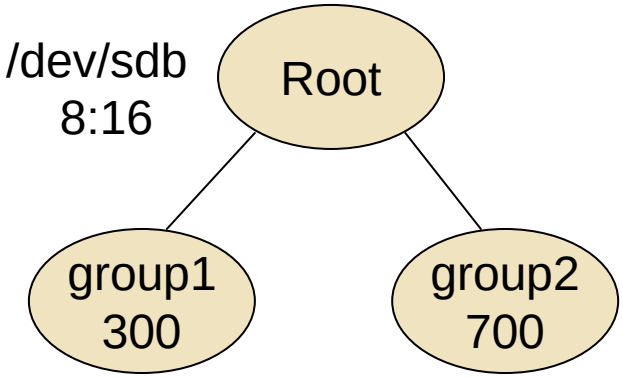
FUJITSU

- **mkdir /cgroup**
- **mount -t cgroup -o io,blkio io  /cgroup**

- **mkdir /cgroup/group1**
- **echo 8:0 800 2 > /cgroup/group1/io.policy**
- **echo 8:16 300 2 > /cgroup/group1/io.policy**

- **mkdir /cgroup/group2**
- **echo 8:0 200 2 > /cgroup/group2/io.policy**
- **echo 8:16 700 2 > /cgroup/group2/io.policy**



# CGroup Interfaces of IO Controller

FUJITSU

- **io.disk_time (read only)**
  - disk time allocated to cgroup per device in milliseconds.

```
[root@localhost ~]# cat /cgroup/group1/io.disk_time
8:16 1930
8:0 38
```

- **io.disk_sectors (read only)**
  - number of sectors transferred to/from disk by the group.

```
[root@localhost tmp2]# cat /cgroup/group1/io.disk_sectors
8:16 442520
8:0 32
```

# Sysfs Interfaces of IO Controller

FUJITSU

- **fairness (0 or 1)**
  - Provide better disk time accounting for async IO
  - Provide isolation between reads and writes
- **nr_group_requests**
  - maximum number of requests per group
- **group_idle**
  - Prevent from losing the group share

# Agenda

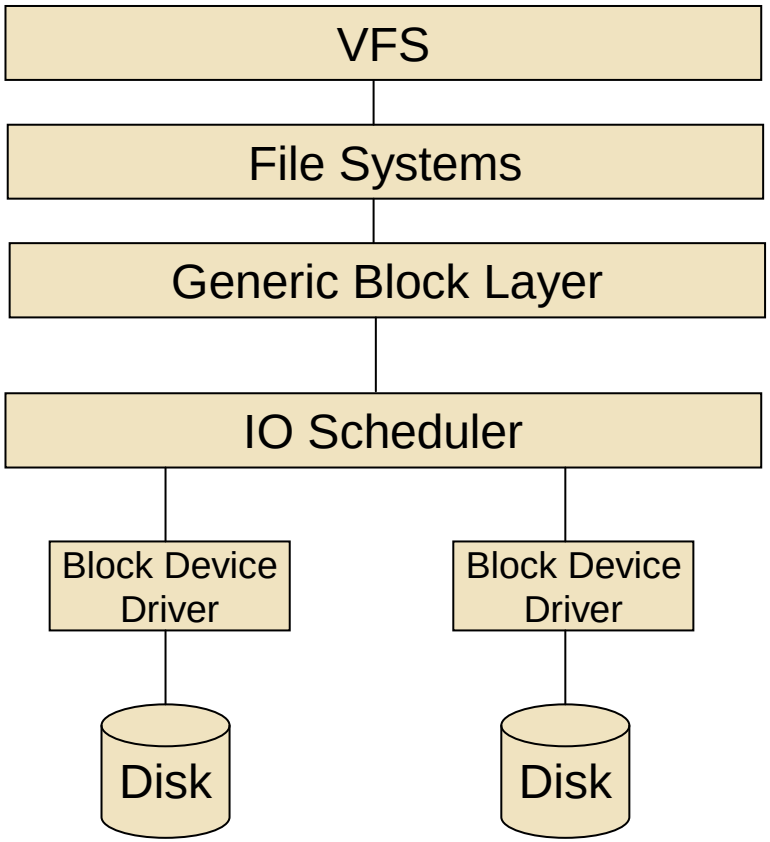
FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposal for IO Bandwidth Controlling

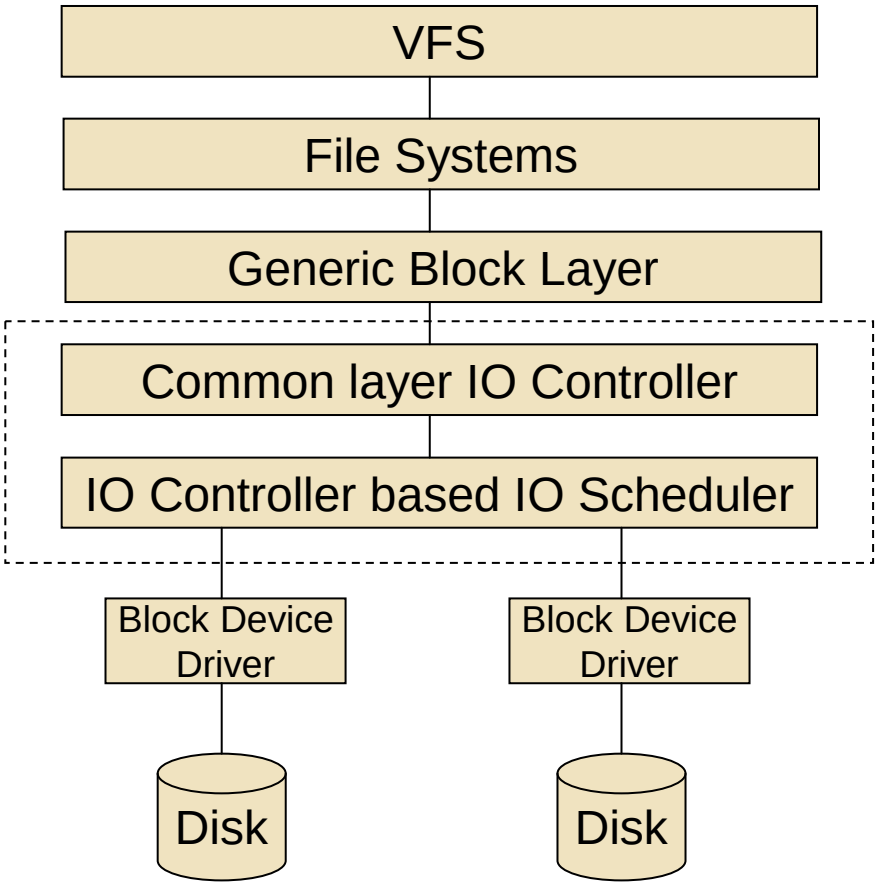
# Where does IO Controller locate

FUJITSU

Vanilla kernel

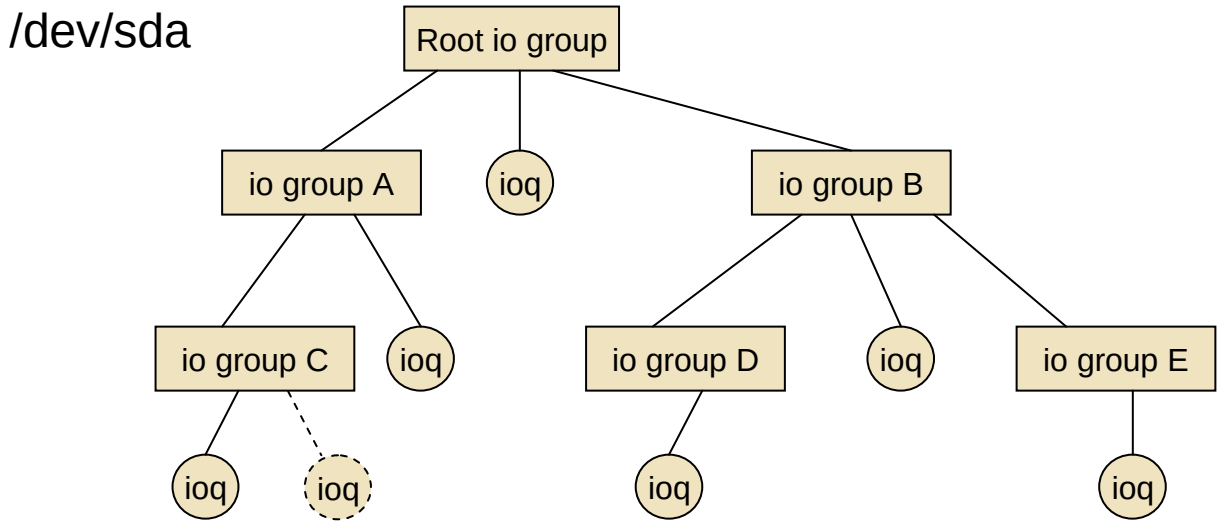



With IO Controller



# Scheduling Overview of IO Controller

FUJITSU




- Imagine each device as a tree where IO requests stay
- Dynamically create and destroy
- Single ioq for anticipatory, deadline and noop scheduler
- Multi ioq for cfq scheduler
- Select one ioq to serve (dispatch requests)

# IO Processing in IO Controller

FUJITSU

- **bio is submitted into block layer (submit_bio)**
- **Create a new request if merging is not allowed**
- **Find out the CGroup that bio belongs to**
  - for sync io case, tracking CGroup by “current” task
  - for async io case, tracking CGroup by blkio_cgroup
- **Find out corresponding io group**
- **Find out the ioq the request should go into**
- **Insert request into ioq**

# IO Controller Internal

FUJITSU

- **A CGroup Based Solution**
- **A Proportional and Weight based controller**
- **IO Scheduler Based Solution**
  - Make each IO Scheduler queue awared
  - Common layer takes care of queue scheduling
- **Per group requests limitation**
  - nr_requests & nr_group_requests
  - Make sure that one group won't allocate too many requests and block other groups.

# IO Controller Internal

FUJITSU

- **A CFS + CFQ Solution**
  - Like CFQ: Give time slice to ioq based on its priority
  - Like CFS: Decide which ioq should dispatch requests based on the Virtual Disk Time
- **Make difference between sync and async IOs**
  - "current" task is used to determine the io group of the sync requests
  - Make use of blkio_cgroup to keep track of async requests
- **in-group preemption and in-group merge**

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposal for IO Bandwidth Controlling

# Problems with IO Controller

FUJITSU

- Works only with physical devices (like sda, sdb)
- Doesn’t work well if several writers are running
- Doesn’t support max bandwidth control
- Which group should be charged if swap writeouts happen
- Extensions of struct bio(bio io_context & bio urgent flag)

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposal for IO Bandwidth Controlling

# Development Status

FUJITSU

- **The Lastest IO Controller Version is V10 (Linux-2.6.31)**
  - http://lkml.org/lkml/2009/9/24/385
- **Who is working on this project**
  - Redhat, Google, Fujitsu
- **The IO Controller mini-summit in Tokyo**
  - Only one IO Controller in kernel
  - Implement for CFQ only
  - Support logical device
  - Memcg will implement Per-CGroup dirty ratio and IO Controller will make use of it

# Agenda

FUJITSU

- Background Knowledge
- What is IO Controller
- How to use IO Controller
- The Implementation of IO Controller
- Problems with IO Controller
- Development Status
- Other Proposals for IO Bandwidth Controlling

# Other Proposals for IO Bandwidth Controlling

FUJITSU

- **dm-ioband (valinux guys)**
  - Implemented as device mapper driver.
  - Implemented in higher layers than IO Schedulers (second level)
  - Complex setup
- **io-throttler (Andrea Righi)**
  - Throttle a process if it exceeds the group limit
  - Implemented in higher layers than block layer (second level)
  - Development has stopped.

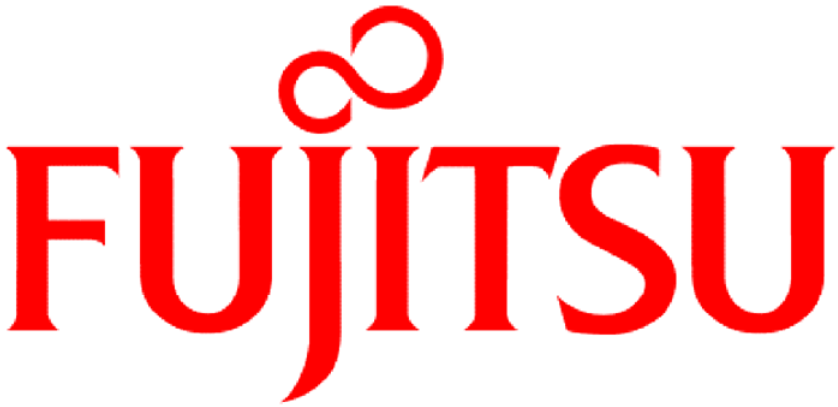
# Disadvantage of second level controller

FUJITSU

- Disk sector vs Disk time accounting
- Bios are buffered in a single queue
  - writes blocks reads issue
  - Doesn't have the notion of io priority
  - More seeks
  - Break the anticipation of underlying IO Scheduler

FUJITSU THE POSSIBILITIES ARE INFINITE

# Thanks!
# Questions?




THE POSSIBILITIES ARE INFINITE