ioTrace
Another Disk Activity Tracing Tool

Ahao Mu
(ahao.mah@alibaba-inc.com)

June 26, 2018
Background

- Requirement proposed by Alibaba’s business line: Process centralized disk activities.
- Currently implemented tools can’t meet the requirement.
Pain

- The PID/TID are unknown in scenario of disk bandwidth is overhauled.

- It brings difficulties to narrow down the problematic processes/threads.
Disk IO Toolset

- **iotop**
  - Written in Python language, read from `/proc/<pid>/io` and `/proc/diskstats`.
  - Missed DEVICE dimension.

- **iostat**
  - Written in C language, read from `/proc/diskstats`, See `Documentation/iostats.txt`.
  - Regardless of processes.

- **blktrace**
  - Written in C language, massive and bogus output.
  - Tremendous performance overhead.

As above all are not the ideal way in our production environment.
Goal of ioTrace

- Aware of PID/TID and DEVICE dimensions.
- Debugging and monitoring disk’s activities.
- Light, agile and easy for daemonizing in production environment.
IO Stack

Applications (Processes)

VFS

Page Cache

Mapping Layer

Submit I/O

Generic Block Layer (GBL)

I/O Scheduler

Request Queue Processing: Insertion, Merging, Sorting, Staging and dispatch.

request

request queue

dispatch queue

Block Device Driver

Block Devices (HDD, SSD, etc.)

Storage

Userspace

OS Kernel
Techniques of ioTrace

- Work on top of block generic layer.
- Based on kernel blktrace API.
- Built with kernel tracepoints.
The API kernel provided

The statistics that ioTrace collects and manipulates:

```c
struct blk_io_trace {
__u32 magic; /* MAGIC<<8 | version */
__u32 sequence; /* event number */
__u64 time; /* in nanoseconds */
__u64 sector; /* disk offset */
__u32 bytes; /* transfer length */
__u32 action; /* what happened */
__u32 pid; /* who did it */
__u32 device; /* device identifier (dev_t) */
__u32 cpu; /* on what cpu did it happen */
__u16 error; /* completion error */
__u16 pdu_len; /* length of data after this trace */
};
```

The stages of IO requests are represented by:

```c
enum {
BLK_TC_READ = 1 << 0, /* reads */
BLK_TC_WRITE = 1 << 1, /* writes */
BLK_TC_FLUSH = 1 << 2, /* flush */
BLK_TC_SYNC = 1 << 3, /* sync */
BLK_TC_QUEUE = 1 << 4, /* queueing/merging */
BLK_TC_REQUEUE = 1 << 5, /* requeueing */
BLK_TC_ISSUE = 1 << 6, /* issue */
BLK_TC_COMPLETE = 1 << 7, /* completions */
BLK_TC_FS = 1 << 8, /* fs requests */
BLK_TC_PC = 1 << 9, /* pc requests */
BLK_TC_NOTIFY = 1 << 10, /* special message */
BLK_TC_AHEAD = 1 << 11, /* readahead */
BLK_TC_META = 1 << 12, /* metadata */
BLK_TC_DISCARD = 1 << 13, /* discard requests */
BLK_TC_DRV_DATA = 1 << 14, /* binary driver data */
BLK_TC_FUA = 1 << 15, /* fua requests */
BLK_TC_END = 1 << 15, /* we've run out of bits! */
};
```
The design of iotrace

Key objects and components:

1. CPU List
2. Disk group
3. Epoll
4. Collect thread
5. Analyzer thread
6. Hash table record
7. Ranking logic
Functions of ioTrace

- Support TID, PID and DEVICE dimensions.
- Collect read_iops, write_iops, read_bytes, write_bytes, total_counts.
- Support prompt output to console and lagged json output to remote database.
- Support deamonizing and crond’ing mode with systemd.
- Support specifying target DEVICE name for monitoring.
Usage

Support multiple arguments: target device, prompt output mode, daemonization or crond running mode, ranking output.

# iotracer
Usage: iotracer

[ -d <dev> | --dev=<dev> ]
[ -m | --daemon ]
[ -c | --cron ]
[ -n <number> | --top_candidates=<pid top max> ]
[ -f <filename> | --file=<configure file> ]
[ -v <version> | --version ]
[ -l | --live ]
[ -1 <interval> | --interval=<seconds> ]
[ -p <thread> | --thread=<count> ]

-d Used to specify device
-m Used to specify daemonize running or not
-c Used to specify cron running or not
-n Used to specify top candidates, defaults is 3
-l Used to specify show data live or not
-p Used to specify multiple thread max count
-i Used to specify interval(second)
-f Path to iotracer configure file, defaults to /etc/iotracer/iotracer.conf

e.g:

    #/iotrace -d all -l1
    #/iotrace -d /dev/sda,/dev/sdc -l1
    #/iotrace -c
Data Accuracy

**ioTrace**

![ioTrace output]

**iostat**

![iostat output]

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Metric</th>
<th>ioTrace</th>
<th>iostat</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>20180529 13:11:03</td>
<td>r_bytes</td>
<td>2890KB</td>
<td>2737KB</td>
<td>+5.5%</td>
</tr>
<tr>
<td>20180529 13:11:04</td>
<td>r_bytes</td>
<td>13542KB</td>
<td>14052KB</td>
<td>-3.6%</td>
</tr>
</tbody>
</table>
Case

Output from ioTrace:

```
  pid 125864 process: pangu_chun _r_count:53 _w_count:24 _r_bytes:34252488 _w_bytes:15152864 t_count:77
  pid 125865 process: pangu_chun _r_count:56 _w_count:6 _r_bytes:15802672 _w_bytes:6823893 t_count:63
  pid 125865 process: pangu_chun _r_count:48 _w_count:6 _r_bytes:59284688 _w_bytes:5379344 t_count:48
  pid 125864 process: pangu_chun _r_count:53 _w_count:24 _r_bytes:34252488 _w_bytes:15152864 t_count:77
  pid 125865 process: pangu_chun _r_count:56 _w_count:6 _r_bytes:15802672 _w_bytes:6823893 t_count:63
```

Output from SAR: disk util 100%

Consequence: Kworker is the obstacle
Case

Output from ioTrace:

Output from SAR:

Consequence: PID 125872 is suspicious
Thanks & Questions