Linux Internals

Miguel Filipe, AWS Database Services, 2014-10-15

Intention

Reminding ourselves that mastering our tools is a critical part of good engineering

… and go over specific problems and how insight into aspects of Linux impacted our software stack

Target Audience

Software Engineers

No kernel programming experience

Limited exposure to C programming in Linux

Limited Linux systems administration experience

Topics

Memory Management

File IO nuances and primitives.

Linux *advanced(?)* features

Two Stories

- Why we use Large Pages ?
- How to transfer massive files without affecting steady workload performance ?

Why Large Pages ?

latency sensitive DDB nodes use them

lets refresh memory management

Memory Pages

Typically they are 4KB big on x86 or amd64

They can either be (among other things):

- anonymous (stack, heap..) or *file backed*
- clean or *dirty (write to disk pending)*

Linux buffers writes in memory...

Unified Page Cache

is a single cache for all memory pages

Memory management is tightly coupled with Posix semantics on File IO operations

(read(2), write(2), mmap(2), fsync(2), etc..)

https://www.kernel.org/doc/Documentation/sysctl/vm.txt

Bypass the Page Cache

Page cache can be bypassed using O_DIRECT

> *use O_DIRECT with caution, very nuanced semantics, non-standard, no formal spec...*
> *http://lwn.net/Articles/348739/ (2009)*

"The thing that has always disturbed me about O_DIRECT is that the whole interface is just stupid, and was probably designed by a deranged monkey on some serious mind-controlling substances." -- Linus Torvalds (2002)

http://yarchive.net/comp/linux/o_direct.html

Page creation

Linux uses demand paging

page is created when first access occurs

not at memory allocation

➔ over promising (Overcommitt)

Access to page when no memory available ?

➔ Swap

➔ Out of Memory Killer (free memory)

OOM Killer

Linux will **KILL** a heuristically chosen process when out of memoy (OOM Killer)

We don’t want this, turn it off:

```
sysctl -w vm.overcommit_memory = 2
```

Page cache & Swap

Anonymous memory

- ➔ swapped

File backed memory does NOT

- ➔ clean page : drop
- ➔ dirty page : flush

Linux uses a system-wide modified LRU (~ clock-pro) to decide what to *page out*

Paging and *swapiness*

tune the preference to page out file-backed in detriment of anonymous pages (therefore reducing *swapiness*)

All DDB machines run with:

```
sysctl -w vm.swappiness = 0
```

Large Pages (2MB)

Reduce TLB misses

Reduce page mgmt overhead

Initial TLB support hacky (*hugepages*):

- pre-reserved pages
- cannot be swapped
- must use special syscalls to use them

Using Large Pages

Pre-reserve pages before use (2GB):

```
sysctl -w vm.nr_hugepages = 1024
```

mmap(x,x,x, MAP_HUGETLB | …, x, x)
shmget(x,x, SHM_HUGETLB | …)

http://lwn.net/Articles/375098
https://www.kernel.org/doc/Documentation/vm/hugetlbpage.txt

Transparent Large Pages

Applications will:

- use it without knowing/controlling it

Linux will dinamically:

- merge pages into single large page
- split large page into smaller pages
- page out large pages

  -> transparent fallback is + unpredictable

Transfer Large Files

without affecting latencies
on normal DDB requests

DDB AddReplica

DynamoDB *Replication Groups* store a partition of a table.

Replication Group has *replicas*.

replica: a copy of the partition hosted on a storage node.

AddReplica: group wants to add a new node:

(NodeA, NodeB) -> (NodeA, NodeB, NodeC)

AddReplica *simplified*

1. Change group membership
   a. add new node-id
2. Export replica
   a. prepare files for transfer
3. Transfer replica (~ 20GiB)
4. Import replica
5. … DONE!

Transfer Replica

Destination node:

```
netcat -l tcp-port
    | tar -xf --limit-dirty-pages=8kb target_directory
```

Source node:

```
tar -chf --drop-page-cache source_directory
    | netcat -v nodeC tcp-port
```

Tar modifications

--drop-page-cache -> posix_fadvise(x, DONTNEED)

So that we don't pollute the page cache…

--limit-dirty-pages -> fdatasync()

To moderate disk write load

https://code.amazon.com/packages/Tar/trees/tar-1.17

Problems… ?

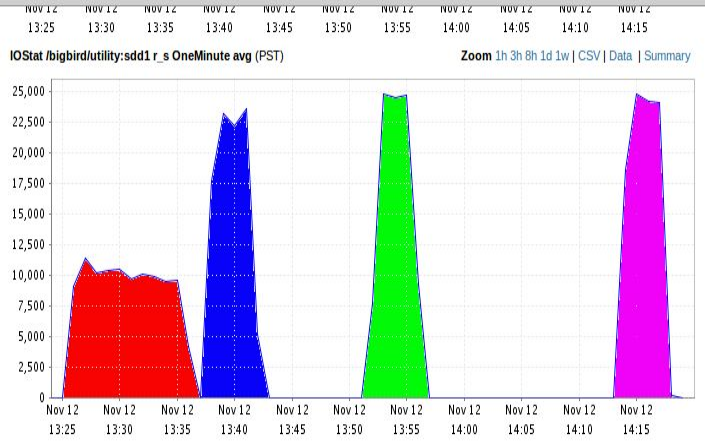
...yup

Host Detail - Mozilla Firefox

Firefox | Host Detail | Host Detail | tcp close_wait - Google Search | linux - How do I remove a CLOS... | windows - What are CLOSE_WAI... | Tiny URL Service -- your URL | Tiny URL Service -- your URL

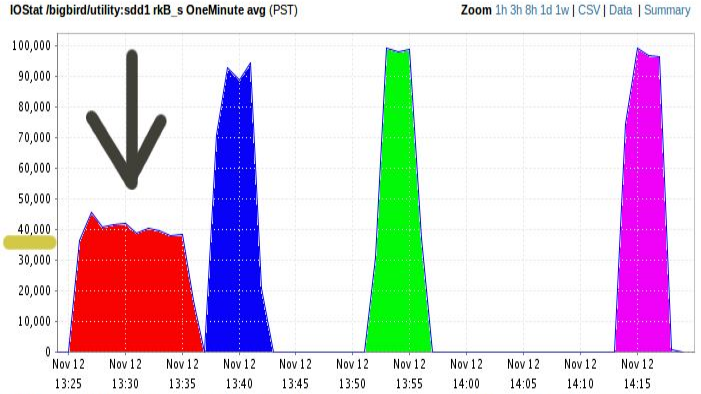
https://monitorportal.amazon.com/hosts/graphs?name=aws-bb-prod-us-east-1-sn-a7218.iad7.amazon.com&is_time_fixed=true&start_time=2013%2F11%2F12+13%3A24&end_time=2013%2F11%2F12

tcp close_wait



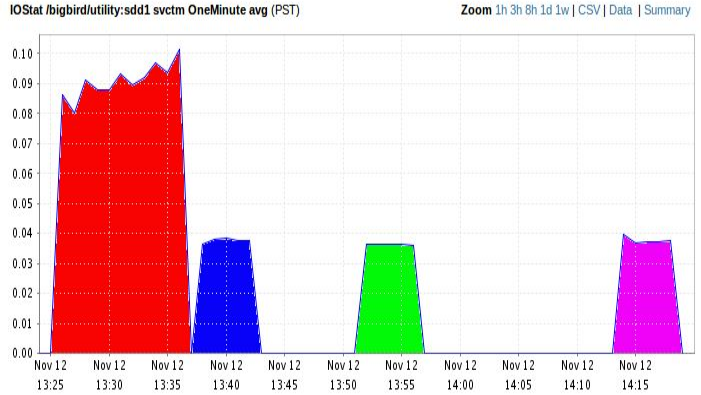
Label	Min	Avg	Max
aws-bb-prod-us-east-1-sn-a7218.iad7.amazon.com	0.0	6,772	24,780

RED - bbtar --drop-page-cache
BLUE - bbtar
GREEN - dd bs=4kb flag=direct
PURPLE - dd bs=4kb



Label	Min	Avg	Max
aws-bb-prod-us-east-1-sn-a7218.iad7.amazon.com	0.0	27,090	99,118

All tests started with a empty page cache



Label	Min	Avg	Max
aws-bb-prod-us-east-1-sn-a7218.iad7.amazon.com	0.0	0.028	0.10

Drop page cache ?

Very useful for testing…

```
sysctl vm.drop_pages = 1
```

Host Detail - Mozilla Firefox

Firefox | Host Detail | Search Results for "aws-bb-prod... | Host Detail

graphs?name=aws-bb-gamma-us-east-1-clu03-sn-7002.iad7.amazon.com&is_time_fixed=true&start_time=2013%2F11%2F12+15%3A43&end_time=2013%2F11%2F1

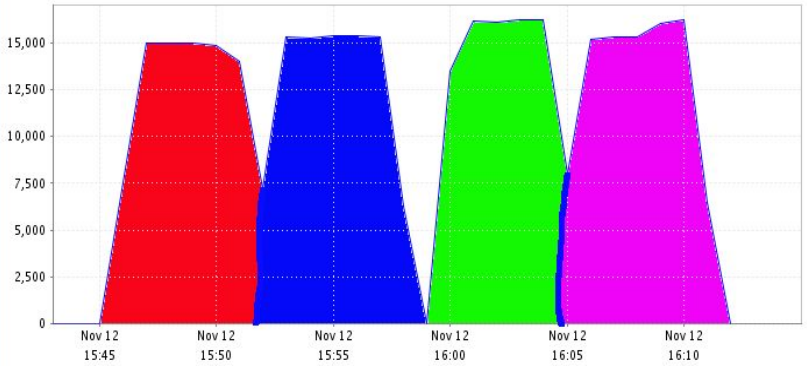
us-east-1-clu03-sn-7002.iad7.amazon.com

Overview | Graphs

SPICY 11.5 utility drive read speed

Graph Controls

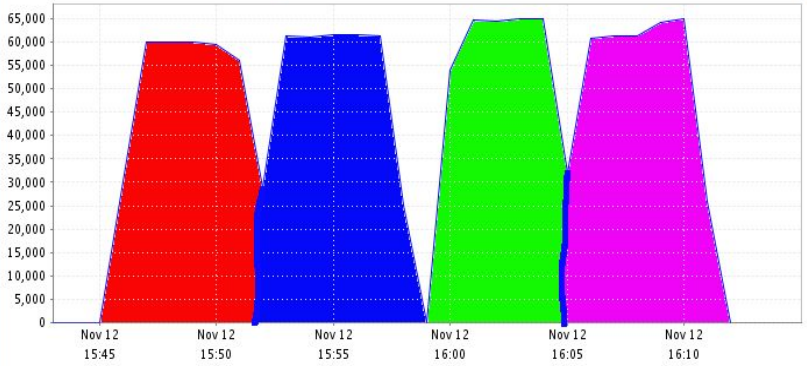
IOStat /bigbird/utility:sde r_s OneMinute avg (PST) — Zoom 1h 3h 8h 1d 1w | CSV | Data | Summary



Label	Min	Avg	Max
aws-bb-gamma-us-east-1-clu03-sn-7002.iad7.amazon.com	0.0	11,402	16,228

RED - bbtar --drop-page-cache
BLUE - bbtar
GREEN - dd bs=4kb
PURPLE - dd bs=4kb flag=direct

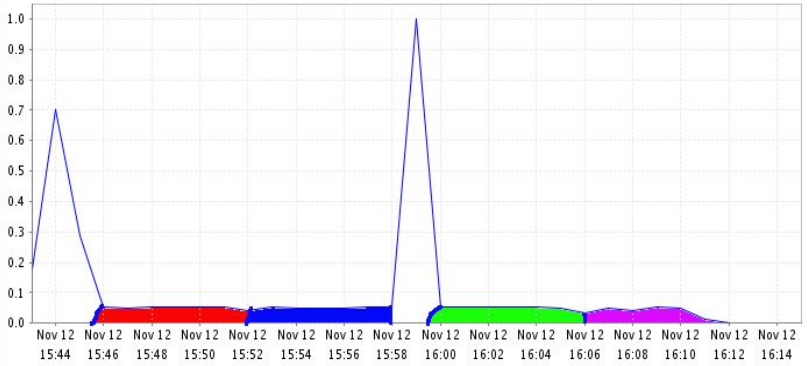
IOStat /bigbird/utility:sde rkB_s OneMinute avg (PST) — Zoom 1h 3h 8h 1d 1w | CSV | Data | Summary



Label	Min	Avg	Max
aws-bb-gamma-us-east-1-clu03-sn-7002.iad7.amazon.com	0.0	45,609	64,913

All tests started with a empty page cache

IOStat /bigbird/utility:sde svctm OneMinute avg (PST) — Zoom 1h 3h 8h 1d 1w | CSV | Data | Summary



Label	Min	Avg	Max
aws-bb-gamma-us-east-1-clu03-sn-7002.iad7.amazon.com	0.0	0.11	1.00

AddReplica Evolves…

1. Change group membership
   a. add new nodeId
2. ~~Export replica~~
3. Transfer replica
   a. straight from StorageEngine (with O_DIRECT)
4. Import replica
5. DONE

Transfer Inefficiencies...

Source node:

```
tar -chf --drop-page-cache source_directory
    | netcat -v NodeC tcp-port
```

Tar:

read() + write()

Netcat:

read() + write()

4 copies of data

transfer @ 100 MB/sec = 400MB/sec of memory copies

Zero Copy: sendfile()

sendfile() copies data between one file descriptor and another.

File system cache -> Network stack buffers

1 single copy …

we might be able to do 10Gb/s (1.2GiB/sec)..

References

http://lwn.net/Kernel/
http://lwn.net/Kernel/Index/
http://kernelnewbies.org/LinuxChanges
http://kernelnewbies.org/LinuxVersions
http://lxr.free-electrons.com/

Books:

Love, Robert. *Linux Kernel Development*

Writting to Disk

POSIX semantics are *performance oriented*.

Data doesn't go to disk on write() or on close().

It goes on fsync() or on fdatasync().

Java close() calls fsync()

Java flush() calls fsync()

Java does NOT have a native fdatasync() support.

Replication log uses JNI to call fdatasync()

Writing to Disk (flushing dirty pages)

Writing dirty pages to disk is done by the writeback logic.

Writeback logic is a topic of continuous research and improvement, (same applies to writeback logic of InnoDB or WiredTiger)

… We can/should avoid letting the OS decide if we have a better understanding of our workload..