Optimal usage of SSDs under Linux:
Optimize your I/O Subsystem

Werner Fischer,
Technology Specialist Thomas-Krenn.AG

LinuxCon Europe 2011,
October 26th - 28th 2011,
Prague, Czech Republic
Introduction

who I am

Werner Fischer
from Australia
working for a Server vendor
freelancer IT journalist
Linux user since 2001
Piano learner

who I am not

Kernel developer
SSD developer
Agenda

1) SSD layout
2) I/O performance metrics
3) Configurations tips

Source: maximumpc.com
1) SSD layout: memory cell

- **memory cells**
  - NAND memory cell = MOS transistor with floating gate
  - permanently store charge
    - programming puts electrons on floating gate
    - erase takes them off
  - one program/erase (p/e) cycle is a round trip by the electrons
  - back-and-forth round trips gradually damage the tunnel oxide
  - endurance is limited, measured in number of p/e cycles:
    - 50nm MLC ~ 10,000 p/e cycles
    - 34nm/25nm/20nm MLC ~ 3,000 – 5,000 p/e cycles

Source: Intel
1) SSD layout: memory cell

- **memory cells**
  - SLC (Single Level Cell) \(\rightarrow\) 1 bit per memory cell
  - MLC (Multi Level Cell) \(\rightarrow\) 2 bits per memory cell
  - TLC (Triple Level Cell) \(\rightarrow\) 3 bits per memory cell
  - 16LC (16 Level Cell) \(\rightarrow\) 4 bits per memory cell

- **multiple memory cells** (e.g. 16.384) build up a “page”
  - page = smallest area, which can be read/written
1) SSD layout: page

- **one line = page within a SSD**
  - 8.192 Bytes (8 kiB)
  - can be read/written individually
  - cannot be changed/erased individually

Note: example sizes of pages and blocks are taken from Intel's Series 320 SSDs (with IMFT's 25nm Flash chips)
1) SSD layout: block

- **one line = page within a SSD**
  - 8.192 Bytes (8 kiB)
  - can be read/written individually
  - cannot be changed/erased individually

- **one blackboard = block within a SSD**
  - consists of 256 lines (pages), 2,097,152 Bytes (2 MiB)
  - smallest area which can be individually erased (we have only watering-cans for that ;-)

Note: example sizes of pages and blocks are taken from Intel's Series 320 SSDs (with IMFT's 25nm Flash chips)
1) SSD layout: change page

- easier way to change lines?
1) SSD layout: change page

- easier way to change lines!
  - (1) mark old line as invalid
1) SSD layout: change page

- easier way to change lines!
  - (1) mark old line as invalid
  - (2) store new content in a free line
1) SSD layout: change page

- easier way to change lines!
  - (1) mark old line as invalid
  - (2) store new content in a free line

- this can be done as long as there are enough free lines left...

we need spare blackboards to have enough free lines!
1) SSD layout: spare area

- **SSDs need spare area**
  - avoids the erasure of a block when a single page is changed
  - after some time spare area will be filled up, too
  - cleaning gets necessary (garbage collection)
1) SSD layout: blocks $\rightarrow$ planes $\rightarrow$ dies $\rightarrow$ TSOPs

- **planes**
  - multiple blocks make up a plane
  - e.g. 1.024 blocks = 1 plane

- **dies**
  - multiple planes make up a die
  - e.g. 4 planes = 1 die

- **TSOP (thin small outline package)**
  - multiple dies (e.g. 1 - 8)

- **SSDs**
  - e.g. 10 TSOPs

Source: http://newsroom.intel.com/community/intel_newsroom/blog/2011/04/14
Source: maximumpc.com
Agenda

1) SSD layout

2) I/O performance metrics

3) Configurations tips

Source: maximumpc.com
2) I/O performance metrics

- **throughput**
  - MByte/s
  - throughput analogy:
    - # of persons/h from Berlin → Prague

- **# of I/O operations per second**
  - IOPS analogy:
    - # of individual trips to Prague (from Berlin, Vienna, Paris, Rome, ...)

- **latency because of queue depth**
  - queue depth analogy:
    - vehicles must use a ferry to reach destination
    - with how many vehicles does the ferry depart?
Agenda

1) SSD layout

2) I/O performance metrics

3) Configurations tips
   - AHCI (NCQ+DIPM)
   - TRIM (discard)
   - noatime
   - tmpfs
   - alignment
   - over-provisioning

Source: maximumpc.com
3) Configuration tips: AHCI (NCQ+DIPM)

- **NCQ (Native Command Queuing)**
  - allows SSD to execute multiple I/O requests in parallel
  - boosts throughput
  - configure queue depth to get your optimal balance between max. # of IOPS and lowest latency

```
root@werner-t410:~# hdparm -I /dev/sdb | grep -i queue
Queue depth: 32

* Native Command Queueing (NCQ)
```

```
root@werner-t410:~# cat /sys/block/sdb/device/queue_depth
31
```

```
root@werner-t410:~# echo 5 > /sys/block/sdb/device/queue_depth
```

```
root@werner-t410:~# cat /sys/block/sdb/device/queue_depth
5
```

```
root@werner-t410:~#
```
3) Configuration tips: AHCI (NCQ+DIPM)

- **DIPM (Device Initiated Interface Power Management)**
  - reduces idle power down to 0.1 Watt

- **Aggressive Link Power Management**
  - min_power
  - medium_power
  - max_performance

3) Configuration tips: TRIM (discard)

- **ATA TRIM**
  - tells SSD which data can be discarded

visible capacity, e.g. 300 GB

spare area, e.g. 44 GB
3) Configuration tips: TRIM (discard)

- **ATA TRIM**
  - tells SSD which data can be discarded

visible capacity, e.g. 300 GB

spare area, e.g. 44 GB
3) Configuration tips: TRIM (discard)

- **ATA TRIM**
  - tells SSD which data can be discarded
  - without TRIM:
    - deleting a big file (e.g. 100 GB) would lead to keep unusable data
    - unusable data will be maintained during garbage collection!
    - more overhead → lower performance & lower endurance!

visible capacity, e.g. 300 GB

spare area, e.g. 44 GB
3) Configuration tips: TRIM (discard)

- **ATA TRIM using discard infrastructure in Linux**
  - online discard
    - Ext4: since Kernel 2.6.33
    - XFS: since Kernel 3.0
    - Btrfs: since Kernel 2.6.32
  - batched discard (using fstrim command)
    - Ext4: since Kernel 2.6.37
    - XFS: since Kernel 2.6.38
    - Btrfs: since Kernel 2.6.39
  - pre-discard on format
    - E2fsprogs >= 1.41.10
    - Xfsprogs >= 3.1.0
3) Configuration tips: TRIM (discard)

- **ATA TRIM using discard infrastructure in Linux**
  - I/O stack discard support (device mapper):
    - since Kernel 2.6.36: DM targets delay, linear, mpath, stripe
    - since Kernel 2.6.38: DM mirror target
  - no I/O stack discard support yet:
    - MD raid

- **alternatives to discard: wiper.sh / raid1ext4trim.sh**
  - use hdparm --trim-sector-ranges-stdin
  - read warnings in the source of those scripts
  - cannot be used with device mapper
3) Configuration tips: TRIM (discard)

- “does TRIM work?” howto

```
root@werner-t410:~# sudo hdparm -I /dev/sda | grep -i trim
* Data Set Management TRIM supported (limit 8 blocks)
* Deterministic read ZEROs after TRIM

root@werner-t410:~# echo "ABCD" > testfile; sync
root@werner-t410:~# hdparm --fitmap testfile

testfile:
    filesystem blocksize 4096, begins at LBA 61052928; assuming 512 byte sectors.
    byte_offset begin_LBA end_LBA  sectors
    0 73923472 73923479     8

root@werner-t410:~# hdparm --read-sector 73923472 /dev/sda | head -n 4

/dev/sda:
    reading sector 73923472: succeeded

4241 4443 0000 0000 0000 0000 0000 0000 0000

root@werner-t410:~# rm -f testfile

root@werner-t410:~# fstrim -v . ; sync
  .: 3496751104 bytes was trimmed

root@werner-t410:~# hdparm --read-sector 73923472 /dev/sda | head -n 4

/dev/sda:
    reading sector 73923472: succeeded

0000 0000 0000 0000 0000 0000 0000 0000 0000

root@werner-t410:~#
```
3) Configuration tips: noatime & tmpfs

- noatime / relatime
  - omits writes of metadata on every read

- tmpfs
  - for temporary data like /tmp/, /var/tmp/, /var/cache/, ...
4) Configuration tips: alignment

- **align partition and file systems**
  - wrong alignment:

  ![Wrong Alignment Diagram]

  - use fdisk parameters: `fdisk -c -u /dev/sda`
  - correct alignment:

  ![Correct Alignment Diagram]
4) Configuration tips: over-provisioning

- do not use full normal visible capacity
  - activate HPA (host protected area)
    - ATA8-ACS SET MAX ADDRESS
    - use hdparm -N
  - or simply do not partition full visible capacity
  - in either case if SSD has been used before:
    - do a secure erase to TRIM all blocks

HPA limited capacity e.g. 200 GB

normal visible capacity, e.g. 300 GB

spare area, e.g. 44 GB
4) Configuration tips: over-provisioning

• over-provisioning is useful when discard cannot be used yet (e.g. MD-RAID, hardware RAID, ...)

• measurements by Intel:

Source: Intel
Review:

1) SSD layout

2) I/O performance metrics

3) Configurations tips
   - AHCI (NCQ+DIPM)
   - TRIM (discard)
   - noatime
   - tmpfs
   - alignment
   - over-provisioning

Source: maximumpc.com
Thanks for your time!

wfischer@thomas-krenn.com