VELOCITY SOFTWARE

Linux on z/VM Performance

Understanding Disk I/O

Rob van der Heij

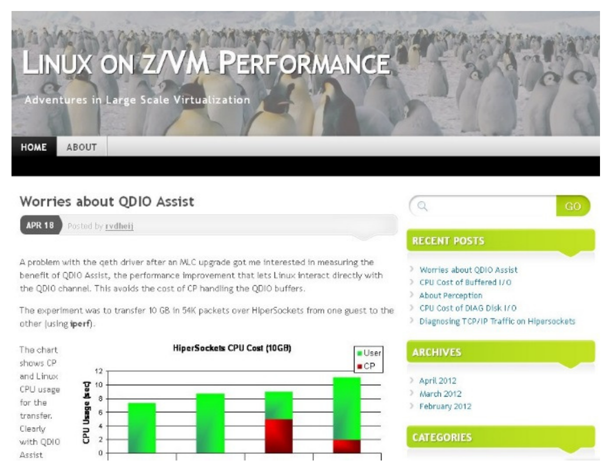
Velocity Software

http://www.velocitysoftware.com/

rvdheij@velocitysoftware.com

Agenda

- I/O Performance Model
- ECKD Architecture
- RAID Disk Subsystems
- Parallel Access Volumes
- Virtual Machine I/O
- Linux Disk I/O



http://zvmperf.wordpress.com/

Linux on z/VM Tuning Objective

Resource Efficiency

- Achieve SLA at minimal cost
  - "As Fast As Possible" is a very expensive SLA target
- Scalability has its limitations
  - The last 10% peak capacity is often the most expensive

Recommendations are not always applicable

- Every customer environment is different
- Very Few Silver Bullets
- Consultant skills and preferences

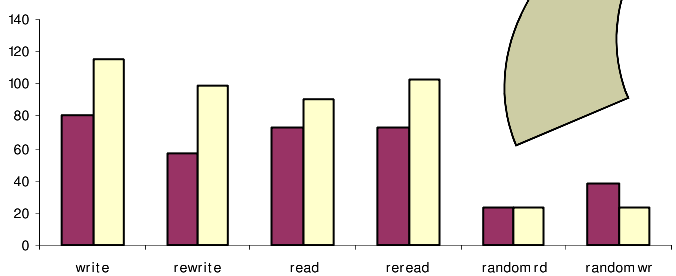
Benchmark Challenges

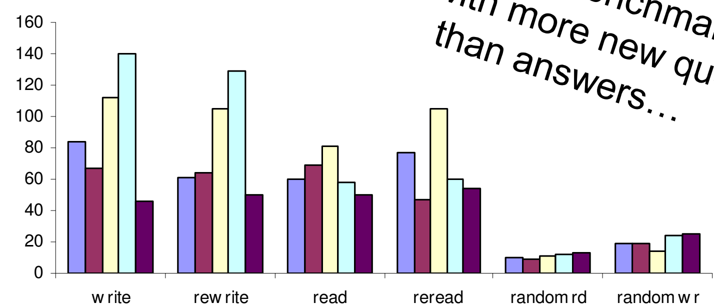
Benchmarks have limited value for real workload

- Every real life workload is different
  - All are different from synthetic benchmarks
  - There are just too many options and variations to try
- Benchmarks can help understand the mechanics
  - Provide evidence for the theoretical model

Use performance data from your real workload

- Focus on the things that really impact service levels





Given enough benchmarks, you end up with more new questions than answers…

Anatomy of Basic Disk I/O

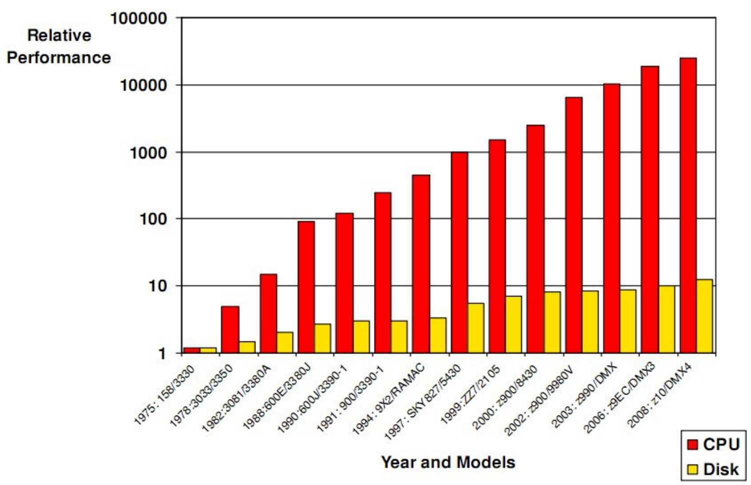
Who Cares About Disk

"Disks are very fast today"

"Our response time is a few ms"

Selection Criteria

- Capacity
- Price



© 2010 Brocade, SHARE in Seattle, "Understanding FICON I/O Performance"

Reality: In comparison, disk I/O today is slow

	IBM 3380-AJ4 (1981)	Seagate Momentus 7200.3 (2011)
Price	$80K	$60
Capacity	2.5 GB	250 GB
Latency	8.3 ms	4.2 ms
Seek Time	12 ms	11 ms
Host Interface	3 MB/s	300 MB/s
Device Interface	2.7 MB/s	150 MB/s

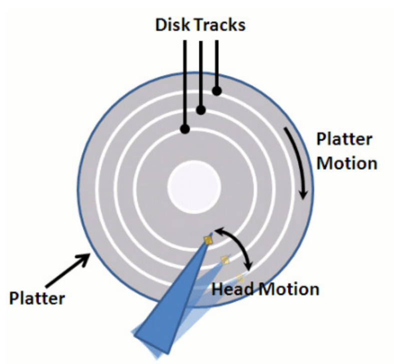
Anatomy of Basic Disk I/O

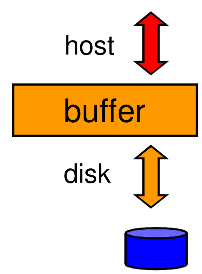
Reading from disk

- Seek – Position the heads over the right track
- Latency – Wait for the right sector
- Read – Copy the data into memory

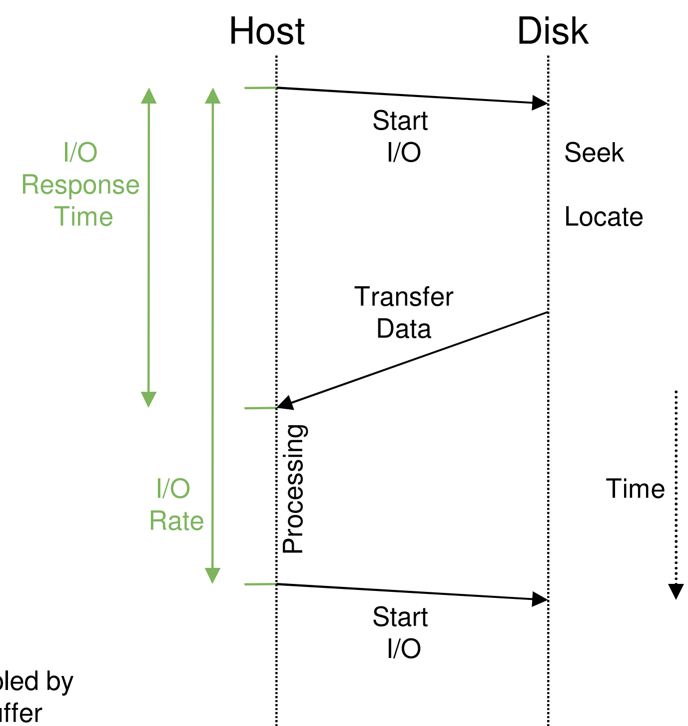
Average I/O Operation

- Seek over 1/3 of the tracks ~ 10 ms
- Wait for 1/2 a rotation ~ 3 ms
- Read the data ~ 1 ms





Host and disk decoupled by speed matching buffer



Classic DASD Configuration

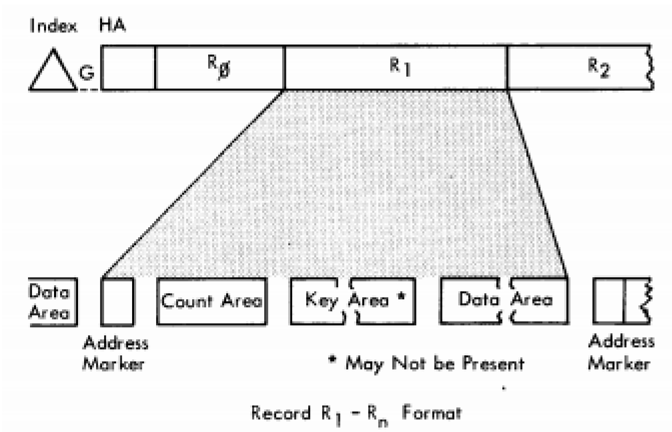
CKD – Count Key Data Architecture

- Large system disk architecture since 60's
- Track based structure
  - Disk record size to mach application block size
- Disk I/O driven by channel programs
  - Autonomous operation of control unit and disk
  - Reduced CPU and memory requirements
- ECKD – Extended Count Key Data
  - Efficient use of cache control units
  - Improved performance with ESCON and FICON channel

Linux disk I/O does not exploit CKD features

FBA – Fixed Block Architecture

- Popular with 9370 systems
- Not supported by z/OS
- Access by block number
- Uniform block size



Classic DASD Configuration

Channel Attached DASD

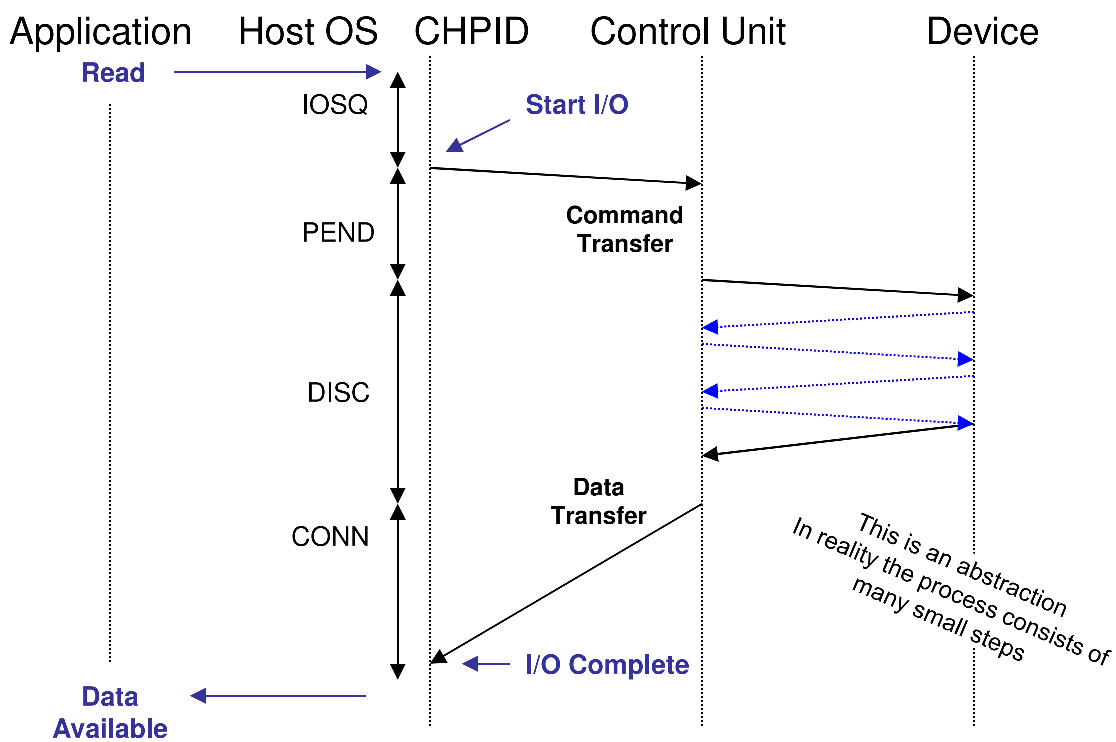
- Devices share a channel
- Disconnect and reconnect
- Track is cached in control unit buffer

IOSQ
- Device Contention
- Interrupt Latency

PEND
- **Channel Busy**
- Path Latency
- Control Unit Busy
- Device Busy

DISC
- **Seek**
- Latency
- Rotational Delay

CONN
- Data Transfer
- Channel Utilization



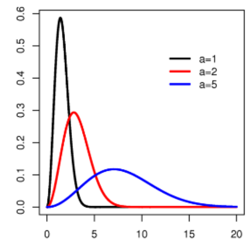
http://zvmperf.wordpress.com/2013/06/07/disk-io-response-time-metrics/

VELOCITY SOFTWARE

Classic DASD Configuration

Instrumentation provided by z/VM Monitor

- Metrics from z/VM and Channel
  - Traditionally used to optimize disk I/O performance
- Response time improvement through seek optimization
  - Relocating data sets to avoid multiple hot spots
  - I/O scheduling – elevator algorithm



DISC = Seek + Rotational Delay

```
Screen: ESADSD2                                 ESAMON 3.807 03/23 16:24-16:33
1 of 3  DASD Performance Analysis - Part 1      DEVICE 3505          2097


          Dev          Device %Dev <SSCH/sec-> <-----Response times (ms)--->
Time      No.  Serial Type   Busy   avg  peak  Resp  Serv  Pend  Disc  Conn
-------- *--- ------ ------ ---- *---- ----- ----- ----- ----- ----- -----
16:25:00 3505 0X3505 3390-? 26.3 728.8 728.8   0.4   0.4   0.2   0.0   0.2
16:26:00 3505 0X3505 3390-? 76.9 977.4 977.4   0.8   0.8   0.3   0.1   0.4
16:27:00 3505 0X3505 3390-? 62.0 480.0 977.4   1.3   1.3   0.5   0.1   0.6
16:28:00 3505 0X3505 3390-? 15.8 198.9 977.4   0.8   0.8   0.1   0.5   0.2
```

VELOCITY SOFTWARE

Contemporary Disk Subsystem

Big Round Brown Disk

- Specialized Mainframe DASD
- One-to-one map of Logical Volume on Physical Volume
- Physical tracks in CKD format
- ECKD Channel Programs to exploit hardware capability

Contemporary Disk Subsystem

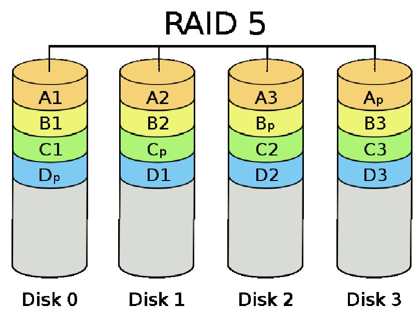
- Multiple banks of commodity disk drives
  - RAID configuration
  - Dual power supply
  - Dual controller
- Microcode to emulate ECKD channel programs
  - Data spread over banks, ranks, array sites
- Lots of memory to cache the data

VELOCITY SOFTWARE

RAID Configuration

RAID: Redundant Array of Independent Disks

- Setup varies among vendors and models
- Error detection through parity data
- Error correction and hot spares
- Spreading the I/O over multiple disks

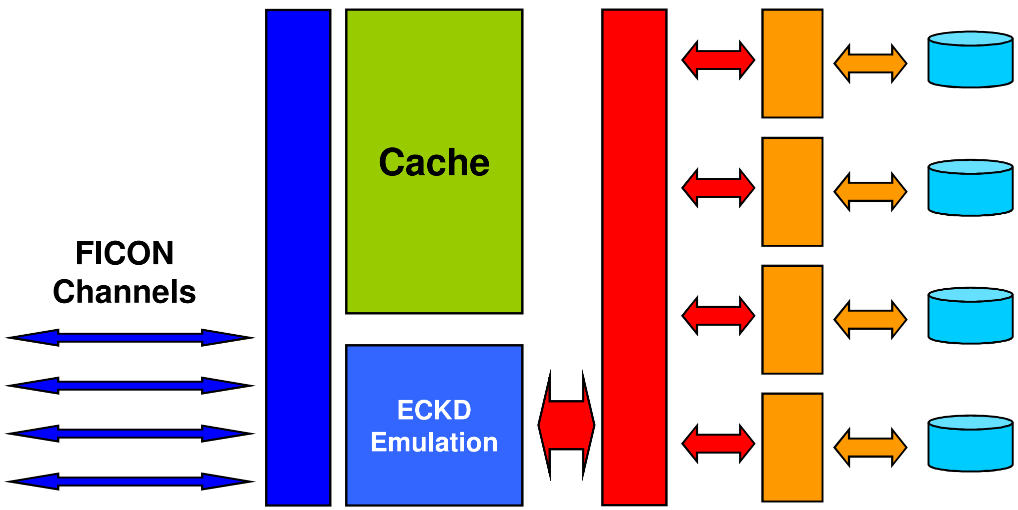


Performance Considerations

- The drives are “just disks”
- RAID does not avoid latency
- Large data cache to avoid I/O
- Cache replacement strategy

Additional Features

- Instant copy
- Autonomous backup
- Data replication

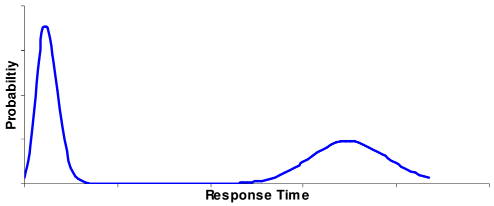


RAID Configuration

Provides Performance Metrics like 3990-3

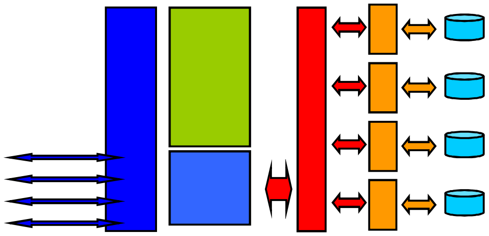
- Model is completely different
- DISC includes all internal operations
  - Reading data into cache
  - Data duplication and synchronization

Bimodal Service Time distribution



- Cache read hit
  - Data available in subsystem cache
  - No DISC time
- Cache read miss
  - Back-end reads to collect data
  - Service time unrelated to logical I/O

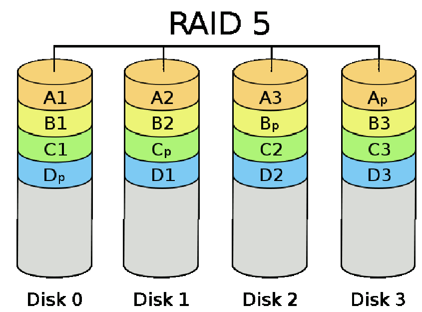
Average response time is misleading



- Cache hit ratio
- Service time for cache read miss

Example:

- Cache Hit Ratio 90%
- Average DISC 0.5 ms
- Service Time Miss 5 ms

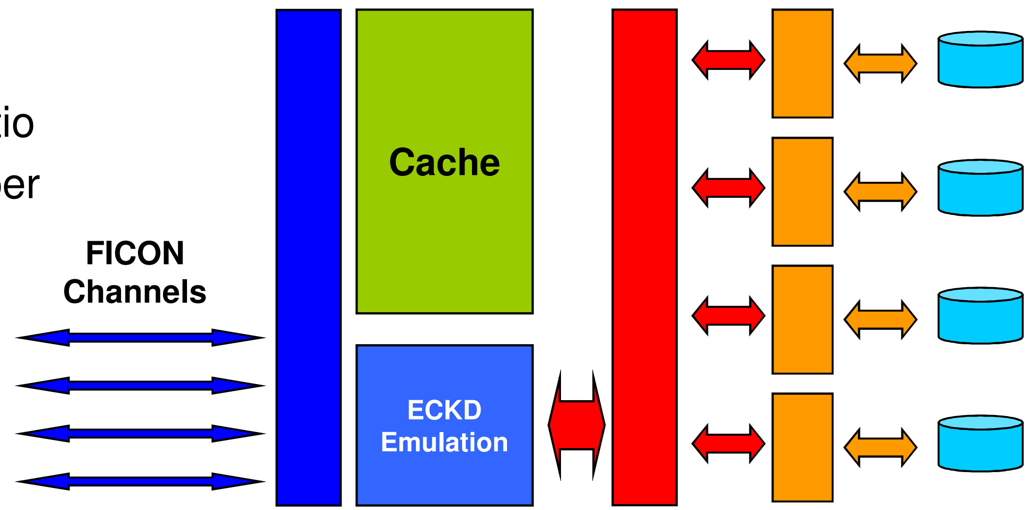


Read Prediction

- Detecting sequential I/O
- ECKD: Define Extent

RAID does not improve hit ratio

- Read-ahead can improve hit ratio
- RAID makes read-ahead cheaper



VELOCITY SOFTWARE

Disk I/O Example

Time	Node	<Processor Pct Util> Total	Syst	User	Nice	Idle Pct	<-Swaps-> In	Out	<-Disk IO-> In	Out	Switch Rate	Intrpt Rate
							<----------Rates (per sec)-------->					
15:12:00	roblnx2	5.9	5.7	0.2	0	60.2	0	0	0	**210K**	**272.1**	0

210K blocks per second = 105 MB/s -> 6.3 GB written

105 MB/s & 272 context switches -> ~ 400 KB I/O's

Time	Dev No.	Serial	Device Type	Total SSCH	ERP SSCH	%Dev Busy	<SSCH/sec-> avg	peak	<-----Response times (ms)---> Resp	Serv	Pend	Disc	Conn
15:12:00	954A	PR954A	3390-9	6350	0	36.8	105.8	105.8	3.5	3.5	0.2	1.2	2.1
15:12:00	95D5	PR954A	3390-9	6677	0	35.9	111.3	111.3	3.2	3.2	0.2	1.1	1.9
15:12:00	95D6	PR954A	3390-9	6532	0	35.7	108.9	108.9	3.3	3.3	0.2	1.2	2.0

PAV Base + 2 Alias
326 I/O per sec

326 writes per second eligible for DFW

Could not keep up with writes
Every 3rd I/O had to wait

DISC time 1.2 ms for 1/3 of I/O's ->
3.9 ms subsystem response for writes

Time	Dev No.	Serial	Pct. Actv Samp	<---- Total I/O ----> <Per Sec> I/O	Hits	Cache Hit%	Read%	<------ Write Activity ------> Total I/O	DFW I/O	DFW Hits	Seq I/O	Hit%	NVS Full
15:12:00	954A	PR954A	100	326	326	100.0	0	**325.7**	326	326	308	100	**123**

Time	Dev No.	Serial	Pct. Actv Samp	<---- Total I/O ----> <Per Sec> I/O	Cach[illegible]	[illegible]	[illegible]	<--Cache---> [illegible]nhib	Bypass	<-Tracks/second-> <-Staged-> Seq	Nseq	De-staged
15:12:00	954A	PR954A	100	326	[illegible]	[illegible]0.0	0	0	0	0	0	**2194**

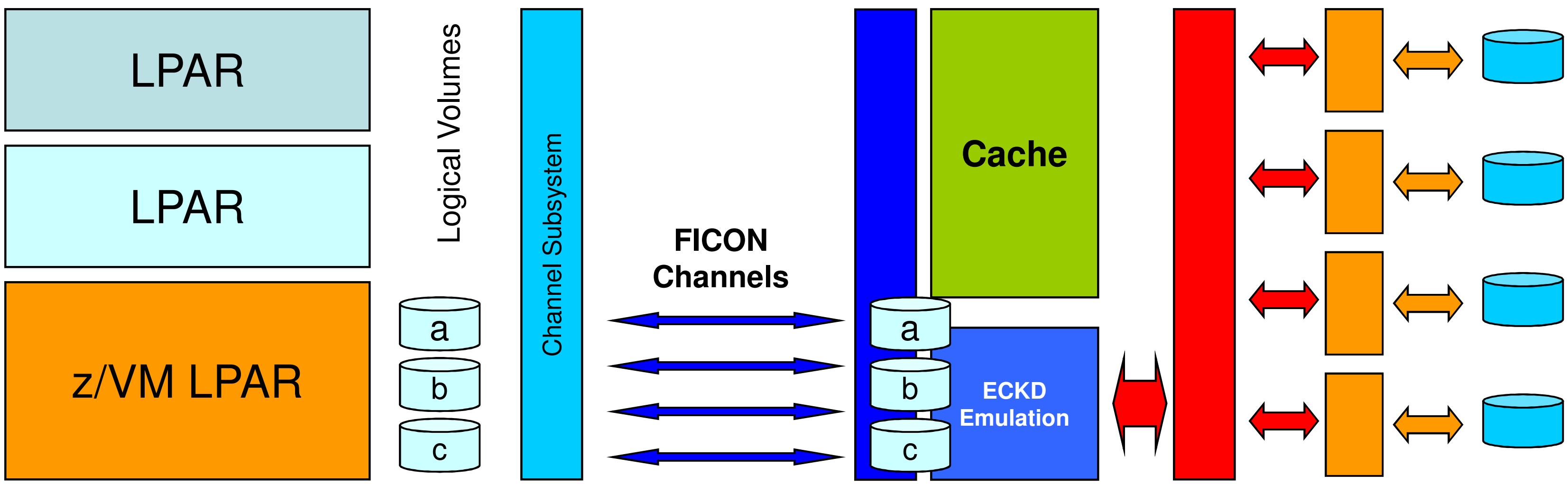
2194 tracks @ 48KB = 105 MB/s

VELOCITY SOFTWARE

Parallel Access Volumes

S/390 I/O Model: Single Active I/O per Logical Volume

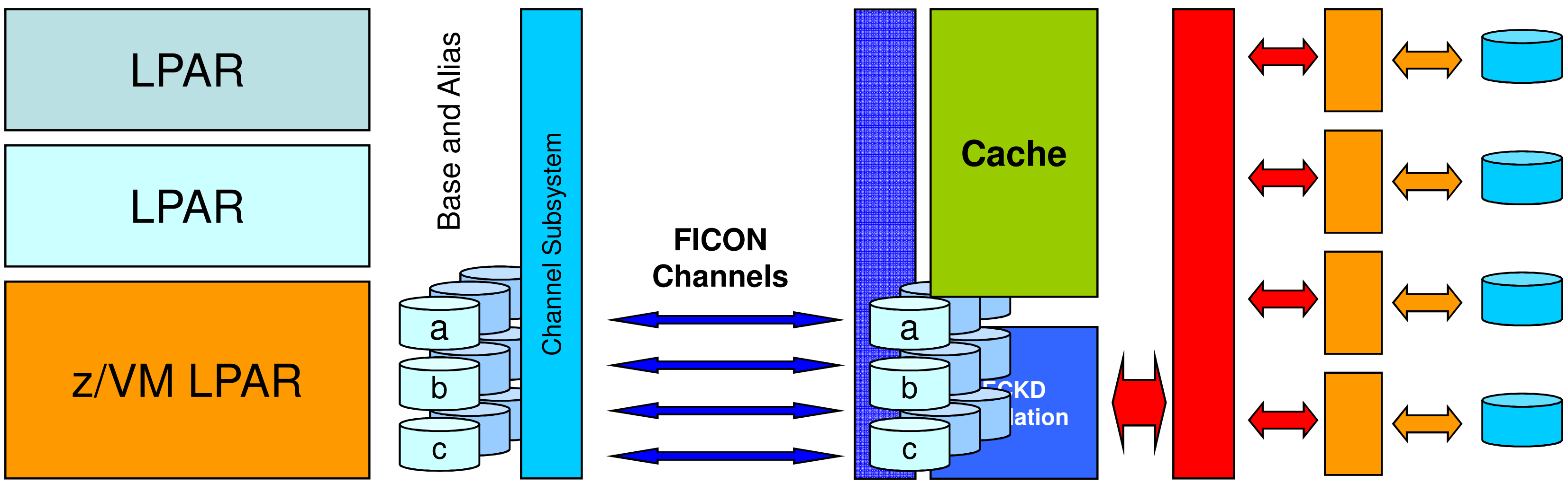
- Made sense with one logical volume per physical volume
- Too restrictive on contemporary DASD subsystems
  - Logical volume can be striped over multiple disks
  - Cached data could be accessed without real disk I/O
  - Even more restrictive with large logical volumes



Parallel Access Volumes

Base and Alias Subchannels

- Alias appear like normal device subchannel
  - Host and DASD subsystem know it maps on the same set of data
  - Simultaneous I/O possible on base and each alias subchannel
- DASD subsystem will run them in parallel when possible
  - Operations may be performed in different order



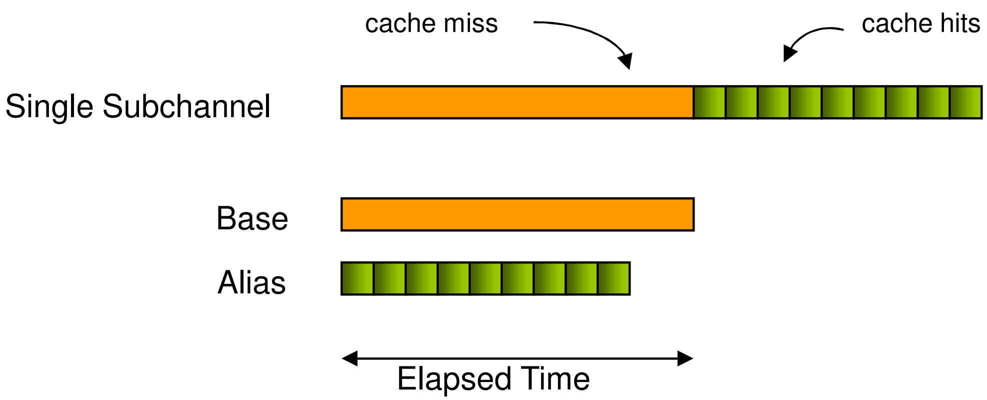
Parallel Access Volumes

Access to cached data while previous I/O is still active

- I/O throughput mainly determined by cache miss operations
  - Assumes moderate hit ratio and an alias subchannel available

Example

- Cache hit ratio of 90%
  - Cache hit response time 0.5 ms
  - Cache miss response 5.5 ms




```
PEND 0.2 ms
DISC 5.0 ms
CONN 0.3 ms
```

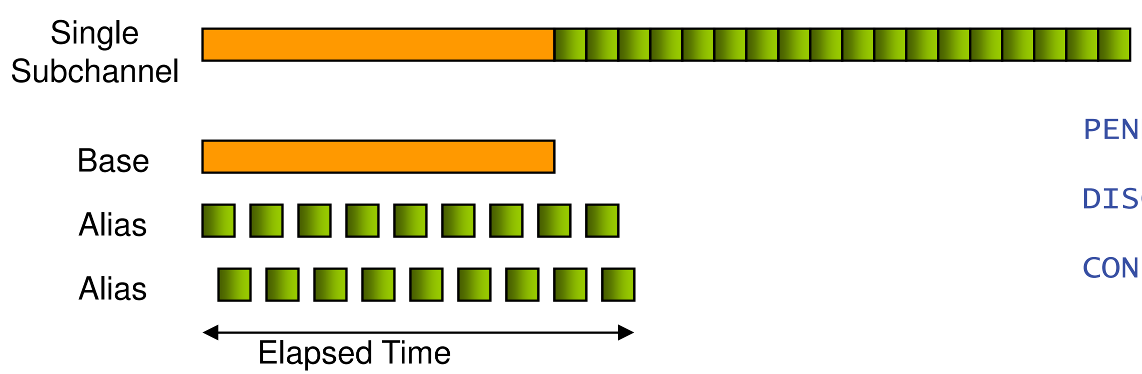
Parallel Access Volumes

Queuing of next I/O closer to the device

- Interesting with high cache hit ratio when PEND is significant
- Avoids delay due to PEND time
  - Service time for cache hit determined only by CONN time
  - Assuming sufficient alias subchannels

Example

- Cache hit ratio of 95%
  - Cache hit response time 0.5 ms
  - Cache miss response 5.5 ms



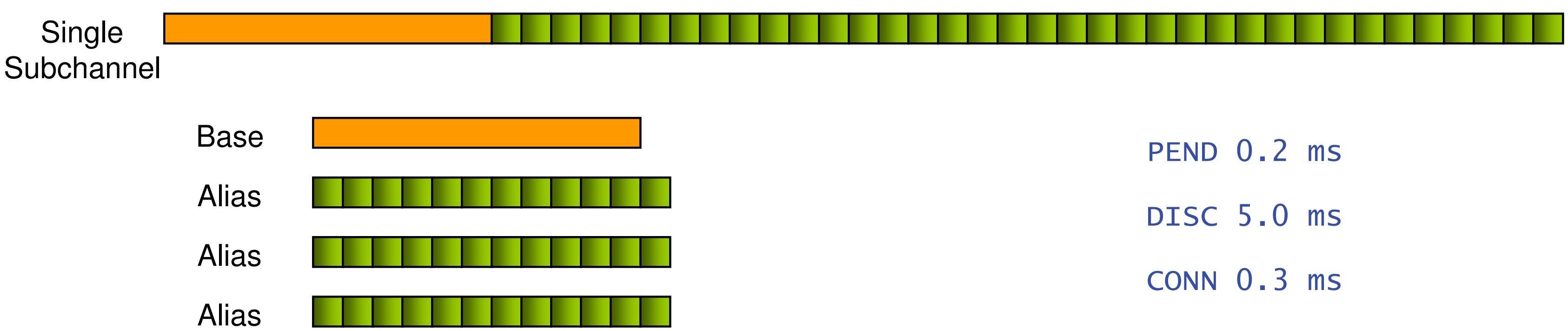

```
PEND 0.2 ms
DISC 5.0 ms
CONN 0.3 ms
```

VELOCITY SOFTWARE

Parallel Access Volumes

Multiple parallel data transfers over different channels

- Parallel operations retrieving from data cache
  - Depends on DASD subsystem architecture and bandwidth
  - Configuration aspects (ranks, banks, etc)
  - Implications on FICON capacity planning
- Cache hit service time improved by the number of channels
  - Combined effect: service time determined by aggregate bandwidth
  - Assumes infinite number of alias subchannels
  - Assumes sufficiently high cache hit ratio



Parallel Access Volumes

Performance Benefits

1. Access to cached data while previous I/O is still active
   - Avoids DISC time for cache miss
2. Queuing the request closer to the device
   - Avoid IOSQ and PEND time
3. Multiple operations in parallel retrieving data from cache
   - Utilize multiple channels for single logical volume

Restrictions

- PAV is chargeable feature on DASD subsystems
  - Infinite number of alias devices is unpractical and expensive
- Workload must issue multiple independent I/O operations
  - Typically demonstrated by I/O queue for the device (IOSQ time)
- Single workload can monopolize your I/O subsystem
  - Requires additional monitoring and tuning

Parallel Access Volumes

Static PAV

- Alias devices assigned in DASD Subsystem configuration
- Association observed by host Operating System

Dynamic PAV

- Assignment can be changed by higher power (z/OS WLM)
- Moving an alias takes coordination between parties
- Linux and z/VM tolerate but not initiate Dynamic PAV

HyperPAV

- Pool of alias devices is associated with set of base devices
- Alias is assigned for the duration of a single I/O
- Closest to “infinite number of alias devices assumed”

Parallel Access Volumes

Virtual machines can exploit PAV

PAV-aware guests (Linux)

- Dedicated Base and Alias devices
- Costly when the guest does not need it all the time

PAV-aware guests with minidisks

- Uses virtual HyperPAV alias devices
- Requires sufficient real HyperPAV alias devices

PAV-unaware guests (CMS or Linux)

- Minidisks on shared logical volumes
- z/VM manages and shares the alias devices
- Break large volumes into smaller minidisks to exploit PAV

VELOCITY SOFTWARE

Linux Disk I/O

Virtual machines are just like real machines

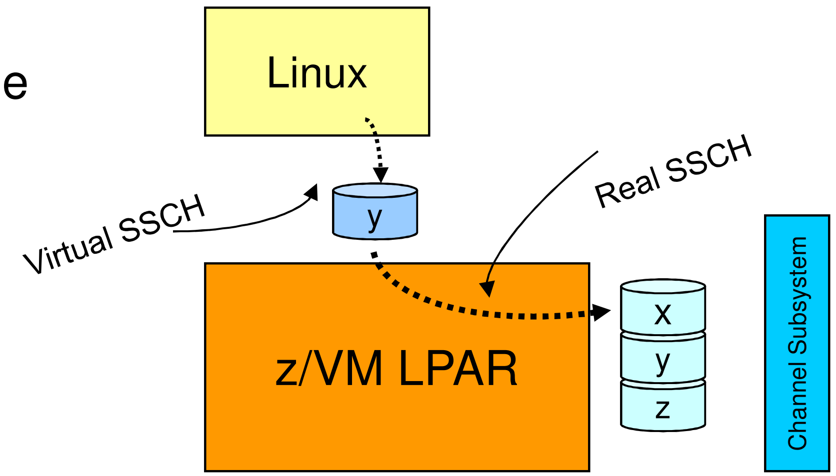
- Prepare a channel program for the I/O
- Issue a SSCH instruction to virtual DASD (minidisk)
- Handle the interrupt that signals completion

z/VM does the smoke and mirrors

- Translate the channel program
  - Virtual address translation, locking user pages
  - Fence minidisk with a Define Extent CCW
- Issue the SSCH to the real DASD
- Reflect interrupt to the virtual machine

Diagnose I/O

- High-level Disk I/O protocol
- Easier to manage
- Synchronous and Asynchronous



VELOCITY SOFTWARE

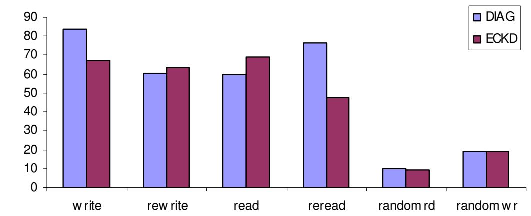
Linux Disk I/O

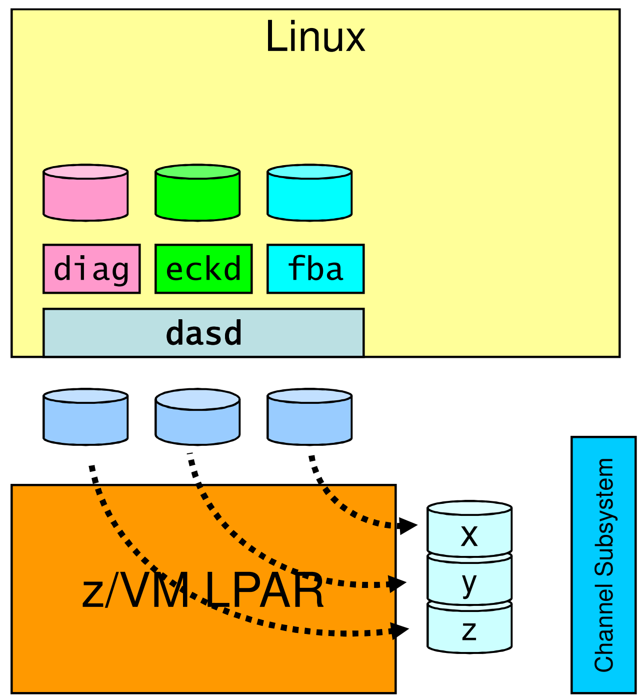
Linux provides different driver modules

- ECKD – Native ECKD DASD
  - Minidisk or dedicated DASD
  - Also for Linux in LPAR
- FBA – Native FBA DASD
  - Does not exist in real life
  - Virtual FBA – z/VM VDISK
  - Disk in CMS format
  - Emulated FBA – EDEVICE
- DIAG – z/VM Diagnose 250
  - Disk in CMS reserved format
  - Device independent

- Real I/O is done by z/VM

- No obvious performance favorite
  - Very workload dependent



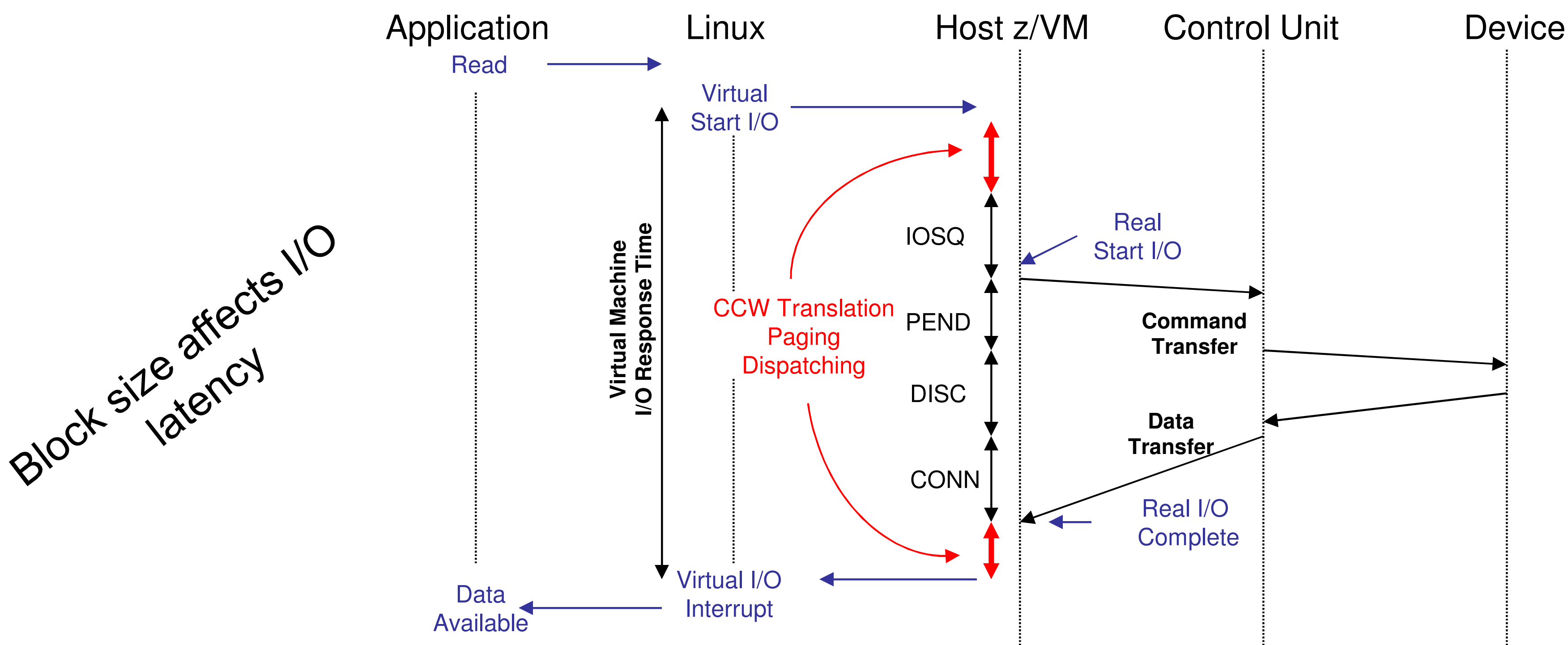


VELOCITY SOFTWARE

Linux Disk I/O

Virtual Machine I/O also uses other resources

- CPU – CCW Translation, dispatching
- Paging – Virtual machine pages for I/O operation



Linux Physical Block Device

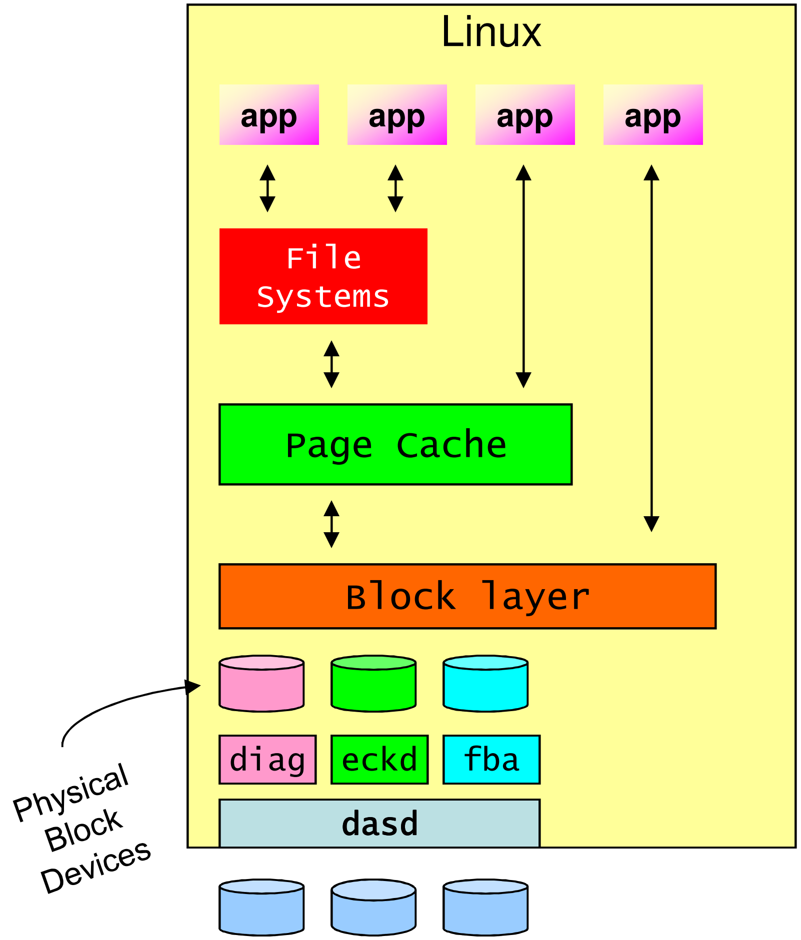
- Abstract model for a disk
  - Divided into partitions
- Data arranged in blocks (512 byte)
- Blocks referenced by number

Linux Block Device Layer

- Data block addressed by
  - Device number (major / minor)
  - Block number
- All devices look similar

Linux Page Cache

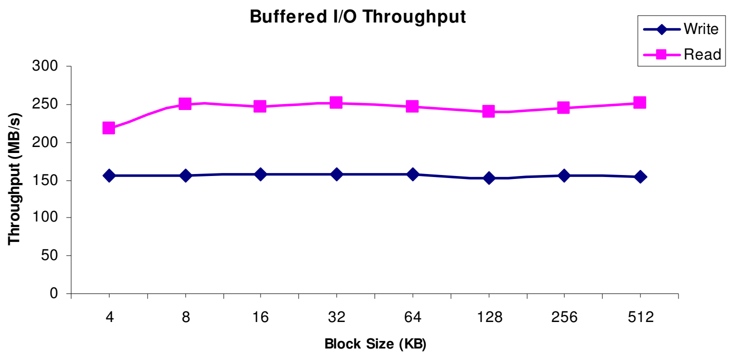
- Keep recently used data
- Buffer data to be written out



Buffered I/O

- By default Linux will buffer application I/O using Page Cache
  - Lazy Write – updates written to disk at “later” point in time
  - Data Cache – keep recently used data “just in case”
  - Read Ahead – avoid I/O for sequential reading

- Performance improvement
  - More efficient disk I/O
  - Overlap of I/O and processing

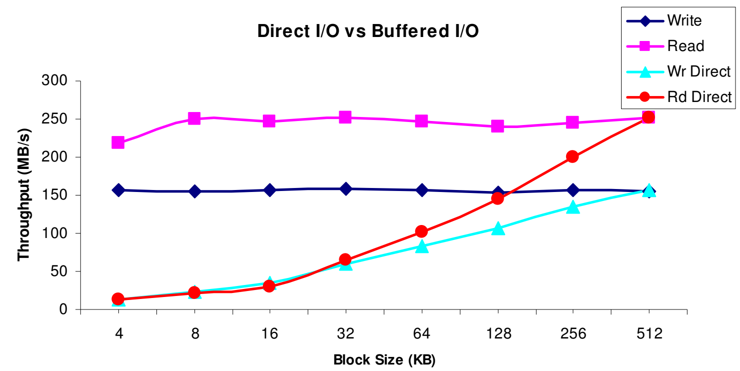


Linux Disk I/O

Buffered I/O

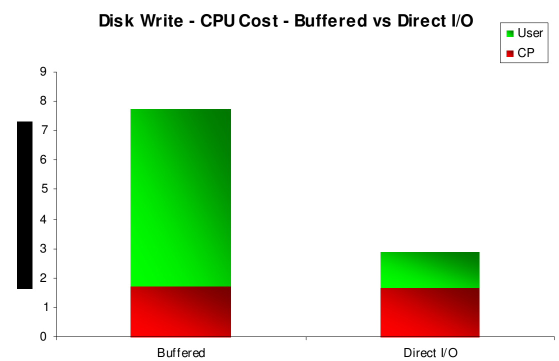
- By default Linux will buffer application I/O using Page Cache
  - Lazy Write – updates written to disk at “later” point in time
  - Data Cache – keep recently used data “just in case”
  - Read Ahead – avoid I/O for sequential reading

- Performance improvement
  - More efficient disk I/O
  - Overlap of I/O and processing



Direct I/O

- Avoids Linux page cache
  - Application decides on buffering
  - No guessing at what is needed next
- Same performance at lower cost
  - Not every application needs it



http://zvmperf.wordpress.com/2012/04/17/cpu-cost-of-buffered-io/

Linux Disk I/O

Myth: Direct I/O not supported for ECKD disks

- Frequently told by DB2 experts

Truth: DB2 does not do 4K aligned database I/O

- The NO FILESYSTEM CACHING option is rejected
- Database I/O is buffered by Linux
  - Uses additional CPU to manage page cache
  - Touches all excess memory to cache data
- FCP disks recommended for databases with business data
  - May not be an option for installations with large FICON investment

Experimental work to provide a bypass for this restriction

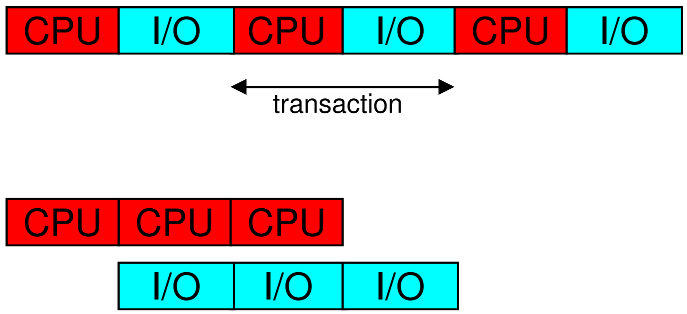
- Interested to work with customers who need this

VELOCITY SOFTWARE

Linux Disk I/O

Synchronous I/O

- Single threaded application model
- Processing and I/O are interleaved



Asynchronous I/O

- Allow for overlap of processing and I/O
- Improves single application throughput
- Assumes a balance between I/O and CPU

Matter of Perspective

- From a high level everything is asynchronous
- Looking closer, everything is serialized again

Linux on z/VM

- Many virtual machines competing for resources
- Processing of one user overlaps I/O of the other
- Unused capacity is not wasted

Myth of Linux I/O Wait Percentage

- Shown in "top" and other Linux tools
- High percentage: good or bad?
- Just shows there was idle CPU and active I/O
  - Less demand for CPU shows high iowait%
  - Adding more virtual CPUs increases iowait%
  - High iowait% does not indicate an "I/O problem"

```
top - 11:49:20 up 38 days, 21:27,  2 users,  load average: 0.57, 0.13, 0.04
Tasks:  55 total,   2 running,  53 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  1.3%sy,  0.0%ni,  0.0%id, 96.7%wa,  0.3%hi,  0.3%si,  1.0%st
```

No I/O problem
Just less CPU usage

High iowait%
I/O problem?

```
top - 11:5   up 38 days, 21:31,  2 users,  load average: 0.73, 0.38, 0.15
Tasks:  55 total,   3 running,  52 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us, 31.1%sy,  0.0%ni,  0.0%id, 62.5%wa,  0.3%hi,  4.3%si,  1.7%st
```

Myth of Linux Steal Time

- Shown in “top” and other Linux tools
  - “We have steal time, can the user run native in LPAR?”
- Represents time waiting for resources
  - CPU contention
  - Paging virtual machine storage
  - CP processing on behalf of the workload
  - Idle Linux guest with application polling
- Linux on z/VM is a shared resource environment
  - Your application does not own the entire machine
  - Your expectations may not match the business priorities
- High steal time may indicate a problem
  - Need other data to analyze and explain

“It was not yours, so nothing was stolen…”

```
top - 11:53:32 up 38 days, 21:31,  2 users,  load average: 0.73, 0.38, 0.15
Tasks:  55 total,   3 running,  52 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us, 31.1%sy,  0.0%ni,  0.0%id, 62.5%wa,  0.3%hi,  4.3%si,  1.7%st
```

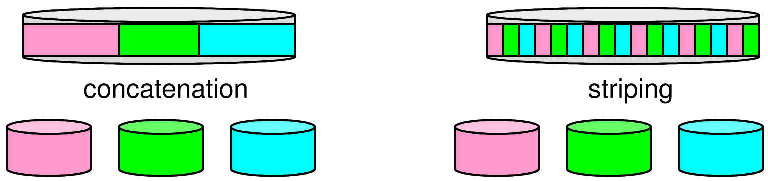
http://zvmperf.wordpress.com/2013/02/28/explaining-linux-steal-percentage/

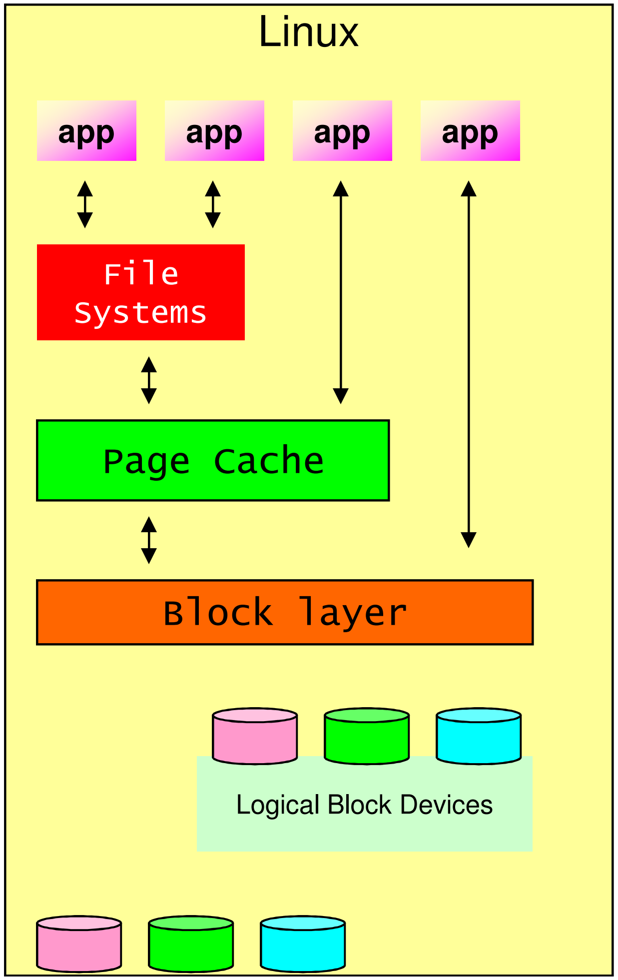
Logical Block Devices
- Device Mapper
- Logical Volume Manager

Creates new block device
- Rearranges physical blocks

Avoid excessive mixing of data

Be aware for more exotic methods
- Mirrors and redundancy
- Anything beyond RAID 0
- Do not mess with I/O scheduler





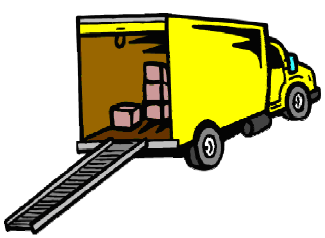
Linux Disk I/O

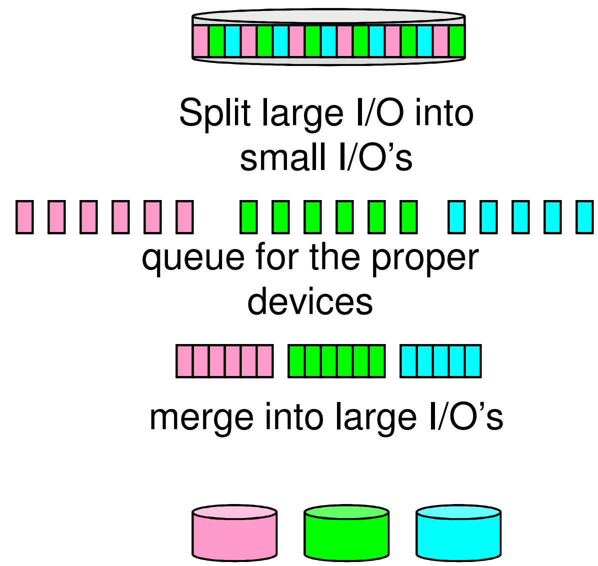
Disk Striping

- Function provided by LVM and mdadm
- Engage multiple disks in parallel for your workload

Like shipping with many small trucks

- Will the small trucks be faster?
  - What if everyone does this?
- What is the cost of reloading the goods?
  - Extra drivers, extra fuel?
- Will there be enough small trucks?
  - Cost of another round trip?





VELOCITY SOFTWARE

Performance Aspects of Striping

- Break up a single large I/O into many small ones
  - Expecting that small ones are quicker than a large ones
  - Expect the small ones to go in parallel
- Engage multiple I/O devices for your workload
  - No benefit if all devices already busy
  - Your disk subsystem may already engage more devices
  - You may end up just waiting on more devices

Finding the Optimal Stripe Size is Hard

- Large stripes may not result in spreading of the I/O
- Small stripes increases cost
  - Cost of split & merge proportional to number of stripes
- Some applications will also stripe the data
- Easy approach: avoid it until performance data shows a problem

The Mystery of Lost Disk Space

Claim: ECKD formatting is less efficient

- “because it requires low-level format” [1]

Is this likely to be true?

- Design is from when space was very expensive
- Fixed Block has low level format too – but hidden from us

ECKD allows for very efficient use of disk space

- Allows application to pick most efficient block size
- Capacity of a 3390 track varies with block size
  - 48 KB with 4K block size
  - 56 KB as single block
- Complicates emulation of 3390 tracks on fixed block device
  - Variable length track size (log-structured architecture)
  - Fixed size a maximum capacity (typically 64 KB for easy math)

[1] *Claim in various IBM presentations*

VELOCITY SOFTWARE

Conclusion

Avoid using synthetic benchmarks for tuning

- Hard to correlate to real life workload

Measure application response

- Identify any workload that does not meet the SLA
- Review performance data to understand the bottleneck
  - Be aware of misleading indicators and instrumentation
  - Some Linux experts fail to understand virtualization
- Address resources that cause the problem
  - Don’t get tricked into various general recommendations

Performance Monitor is a must

- Complete performance data is also good for chargeback
- Monitoring should not cause performance problems
- Consider a performance monitor with performance support

Avoid betting with your Linux admin on synthetic benchmarks

- Drop me a note if you cannot avoid it

VELOCITY SOFTWARE

Linux on z/VM Performance

Understanding Disk I/O

Session 13522

Rob van der Heij

Velocity Software

http://www.velocitysoftware.com/

rvdheij@velocitysoftware.com