SLES11-SP1 Driver Performance Evaluation

Christian Ehrhardt
Linux on System z System & Performance Evaluation
IBM Deutschland Research & Development GmbH

Agenda

- **Performance Evaluation Summary**
  - Overview
  - Comparison
  - Recommendation
- **Benchmark Measurement Results**
  - Environment details
  - Important changes you should know

Remember SLES11

SLES11 RC5/GM vs. SLES10 SP2	LPAR 64 throughput	LPAR 64 costs	LPAR 31 (emu) throughput	LPAR 31 (emu) costs	z/VM 64 throughput	z/VM 64 costs	z/VM 31 (emu) throughput	z/VM 31 (emu) costs
Scaling	to -25%	to -34%	to -25%	to -34%				
Mixed I/O ECKD	to -36%	to -10%			to -40%	to -19%		
Mixed I/O SCSI	to -14%	to -18%			to -18%	to -25%		
Kernel	+80 to -66%				+88% to -84%		+84% to -55%	
Compiler INT	+55% to -7%							
Compiler FP	+12 to -18%							
Web serving	0 to -10%	to -18%			+9% to -8%	+5% to -14%		
Seq. I/O ECKD	rd to -17%	rd to -33%			r-14% to +16%	r-13%, w+16%	r-14% to +18%	r-12%, w+14%
Seq. I/O SCSI	r+33% to -38%	w+30%, r-40%			r-40% to +25%	r-88%, w+24%	r-40% to +17%	r-92%, w+17%
Rnd I/O ECKD		to -33%			r-7%, w+10%	r-5%, w+17%	-6% to +8%	r+14%, w+17%
Rnd I/O SCSI	wr +12%	to -15%			r-12% to +33%	r-25%, w+20%	r-14% to +30%	r-29% to +17%
Seq. I/O ECKD DIO		w+33%, r-14%			-2% to +5%	+12% to +46%		
Seq. I/O SCSI DIO		w-12% , r-11%			-5% to +1%	r-10%, w-14%		
Rnd I/O ECKD DIO		to -16%			-2% to +3%	+8% to +46%		
Rnd I/O SCSI DIO		to -14%			-5% to +1%	r-21%, w-16%		
Java	to -4%		to -6%					
GbE 1492/8992	str -7%	to -45%						
10GbE 1492/8992	str -7%	to -65%						
HiperSockets 32K	200x1000 -33%	to -66%						
VSWITCH guest-guest 1492/8992					to -28%	to -44%		
VSWITCH GbE guest-LPAR 1492/8992					0 to -12%	to -54%		
VSWITCH 10GbE guest-LPAR 1492/8992					0 to -10%	to -69%		
attached GbE guest-LPAR 1492/8992					0 to -10%	to -35%		
attached 10GbE guest-LPAR 1492/8992					0 to -9%	to -50%		
HiperSockets 32K guest-LPAR 1492/8992					o -16%	to -28%		

Legend | n/a | better | equal | worse

Overview - SLES11-SP1 vs. SLES10-SP3

SLES11-SP1 vs. SLES10-SP3	LPAR 64 throughput	LPAR 64 costs	LPAR 31 (emu) throughput	LPAR 31 (emu) costs	z/VM 64 throughput	z/VM 64 costs	z/VM 31 (emu) throughput	z/VM 31 (emu) costs
Scaling	+28% to -30%	-35%	+27% to -25%	-35%				
Mixed I/O ECKD	+156% to -29%*	+3% to -30%*			+86% to -15%*	+4% to -25%*		
Mixed I/O SCSI	+37% to -20%*	'+10% to -25%*			'+33% to -10%*	'+12% to -18%*		
Kernel	+45% to -50%		+45% to -50%		+50% to -45%		+50% to -45%	
Compiler INT	+54% to -8%							
Compiler FP	+17 to -18%							
Web serving	+17% to -15%	-15%			+48% to -15%	+7.5% to -11%		
Seq. I/O ECKD	+128%*	+10% to -93%*			+151%*	+36% to -31%*	+127%*	+28% to -33%*
Seq. I/O SCSI	+28% to -5%*	+26% to -35%*			+63% to -5%*	+26% to -33%*	+36% to -6%*	+28% to -30%*
Rnd. I/O ECKD	+89%*	+7% to -9%*			+66%*	+7% to -2%*	+61%*	+5%*
Rnd I/O SCSI	+78% to -16%*	-7%*			+79% to -16%*	-6%*	+68% to -15%*	-10%*
Seq. I/O ECKD DIO	+75%	+37% to -10%			+116%	+25%		
Seq. I/O SCSI DIO	-2%	+11%			-2%	+9%		
Rnd I/O ECKD DIO	+75%	+10%			+115%	+29%		
Rnd I/O SCSI DIO	+41% to +1%	+37% to +1%			+41% to +1%	+39% to +1%		
Java	-2.9%		-0.8%					
GbE 1492/8992	+11% to -17%	+45% to -33%						
10GbE 1492/8992	+35% to -20%	9.8% to -78%						
HiperSockets 32K	+9% to -13%	+21% to -15%						
VSWITCH guest-guest 1492/8992					+68% to -11%	+34% to -13%		
VSWITCH GbE guest-LPAR 1492/8992					+5% to -31%	+47% to -97%		
VSWITCH 10GbE guest-LPAR 1492/8992					+79% to -17%	+20% to -63%		
attached GbE guest-LPAR 1492/8992					+6% to -15%	+63% to -26%		
attached 10GbE guest-LPAR 1492/8992					+29% to -10%	+13% to -80%		
HiperSockets 32K guest-LPAR 1492/8992					+12% to -16%	+19% to -19%		

Legend	n/a	better	equal	worse

*including workarounds for known issues without fixes in code, but e.g. new tunables

Summary I - comparison

- Improvements and degradations summarized
  - 10 vs. 11 +5 / ~31 / **-46**
  - 10 vs. 11SP1 +33 / **~37** / -12
  - 10 vs. 11SP1* +36 / **~46** / -0
  - 11 vs. 11SP1* +53 / ~29 / -0
- Improvements and degradations per area

vs. SLES10SP3

Improvements	Degradations
FICON I/O	CPU costs
Scaling	OSA single C. Latency
Scheduler	
Compiler	
Multiconn. Networking	

vs. SLES11

Improvements	Degradations
Disk I/O	Only corner cases
Scaling	
Scheduler	
Compiler	
Latency	
CPU costs	
Multiconn. Networking	

*including workarounds for known issues without fixes in code, but e.g. new tunables

Summary II - recommendations

- SLES11-SP1 is a very huge improvement compared to SLES11
- With some trade-offs it is roughly as good or better than SLES10-SP3
  - In addition it has a lot new features
  - By that it is the first performance acceptable “modern” Distribution

- An upgrade is generally recommended
  - Especially
    - SLES11 Systems
    - Systems relying heavily on FICON I/O
    - Large Scale Environments, especially network or CPU intensive loads
  - excluding
    - Very CPU cost sensitive systems (e.g. running fully utilized on SLES10)

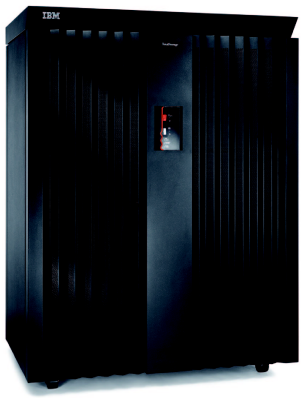
Agenda

- **Performance Evaluation Summary**
  - Overview
  - Comparison
  - Recommendation
- **Benchmark Measurement Results**
  - Environment details
  - Important changes you should know

Our Hardware for Measurements

2097-E26 (z10)
0.23ns (4.4 GHz)
2 Books each with 13 CPUs
192kB L1 Cache (64kB Instr. +128kB Data)
3MB L1.5 Cache (per cpu)
48MB L2 Cache (per book)
320GB RAM (304GB available)
GA3 Driver 79f Bundle 25
FICON Express 4
HiperSockets
OSA Express 2 1GbE + 10GbE

2107-922 (DS8300)
256 GB Cache
1-8 GB NVS
256 * 140 GB disks
15.000 RPM
FCP (4 Gbps)
FICON (4 Gbps)



Detailed Results per Benchmark

- **Compared Drivers**
  - SLES10-SP3 (2.6.16.60-0.54.5-default)
  - SLES11-SP1-GMC (2.6.32.12-0.6-default)

- **Platforms**
  - Linux on LPAR
  - Linux in z/VM 5.4 guest

Overview - SLES11-SP1 vs. SLES10-SP3

SLES11-SP1 vs. SLES10-SP3	LPAR 64 throughput	LPAR 64 costs	LPAR 31 (emu) throughput	LPAR 31 (emu) costs	z/VM 64 throughput	z/VM 64 costs	z/VM 31 (emu) throughput	z/VM 31 (emu) costs
Scaling	-30%	-35%	-25%	-35%				
Mixed I/O ECKD	-29%	-30%			-15%	-25%		
Mixed I/O SCSI	-20%	-25%			-10%	-18%		
Kernel	+45% to -50%		+45% to -50%		+50% to -45%		+50% to -45%	
Compiler INT	+54% to -8%							
Compiler FP	+17 to -18%							
Web serving	+17% to -15%	-15%			+48% to -15%	+7.5% to -11%		
Seq. I/O ECKD	+60% to -35%	-15%			+97% to -43%	+12%	+86% to -35%	+20% to -50%
Seq. I/O SCSI	-5% to -50%	-30%			+27% to -66%	+10% to -10%	+55% to -58%	+38% to -22%
Rnd. I/O ECKD	+50%	-18%			+63%	+11%	+50%	+10%
Rnd I/O SCSI	-15% to +50%	-25%			+63 to -42%	-10%	+62% to -37%	-10%
Seq. I/O ECKD DIO	+75%	+37% to -10%			+116%	+25%		
Seq. I/O SCSI DIO	-2%	+11%			-2%	+9%		
Rnd I/O ECKD DIO	+75%	+10%			+115%	+29%		
Rnd I/O SCSI DIO	+41% to +1%	+37% to +1%			+41% to +1%	+39% to +1%		
Java	-2.9%		-0.8%					
GbE 1492/8992	+11% to 17%	+45% to -33%						
10GbE 1492/8992	+35% to -20%	-10% to -78%						
HiperSockets 32K	+9% to -13%	+21% to -15%						
VSWITCH guest-guest 1492/8992					+68% to -11%	+34% to -13%		
VSWITCH GbE guest-LPAR 1492/8992					+5% to -31%	+47% to -97%		
VSWITCH 10GbE guest-LPAR 1492/8992					+79% to -17%	+20% to -63%		
attached GbE guest-LPAR 1492/8992					+6% to -15%	+63% to -26%		
attached 10GbE guest-LPAR 1492/8992					+29% to -10%	+13% to -80%		
HiperSockets 32K guest-LPAR 1492/8992					+12% to -16%	+19% to -19%		

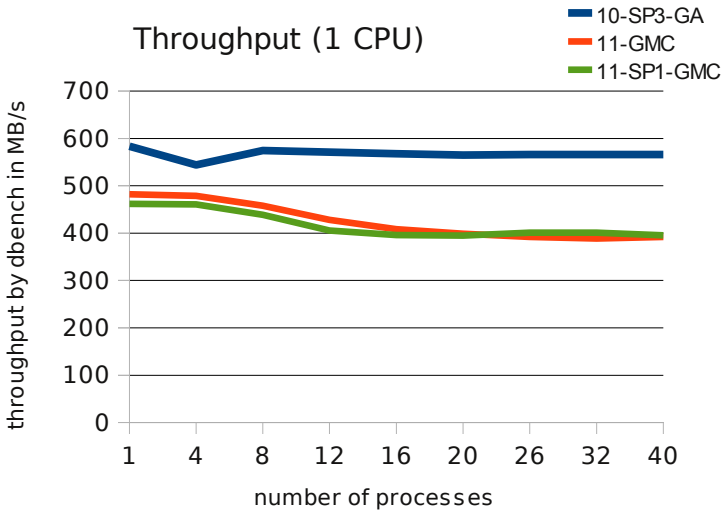
Legend	n/a	better	equal	worse

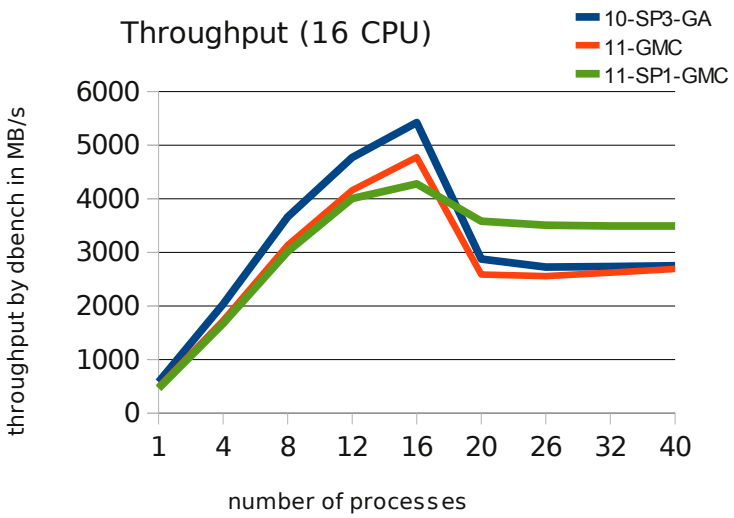
Benchmark descriptions
File system / LVM / Scaling

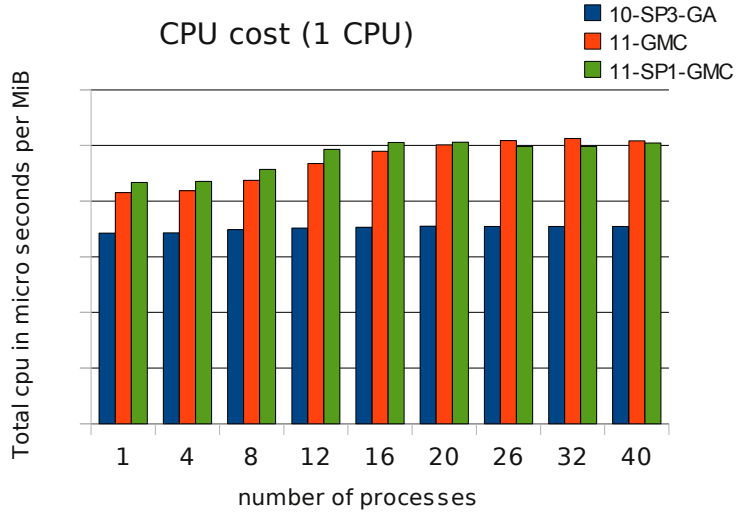
- Filesystem benchmark dbench
  - Emulation of Netbench benchmark
  - generates file system load on the Linux VFS
  - does the same I/O calls like smbd server in Samba (without networking calls)

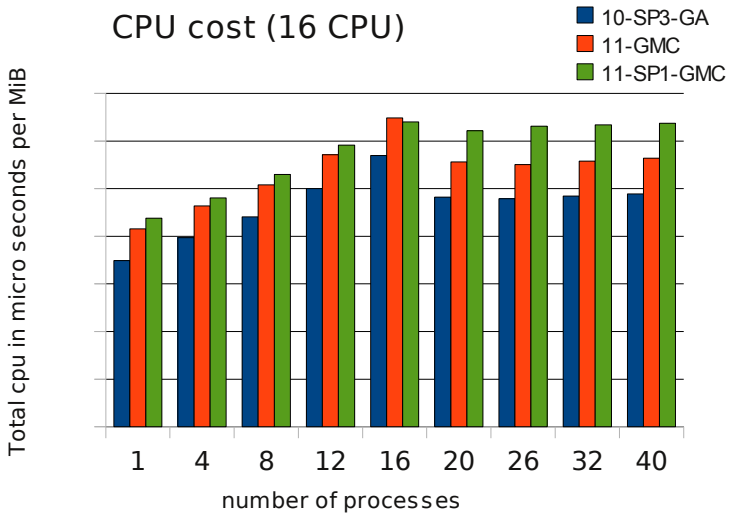
- Simulation
  - Workload simulates client's and server (Emulation of Netbench benchmark)
  - Mainly memory operations for scaling
  - Low Main memory and lvm setup for mixed I/O and lvm performance
  - Mixed file operations workload for each process: create, write, read, append, delete
  - 8 CPUs, 2GB memory and scaling from 4 to 62 processes (clients)
  - Measures throughput of transferred data

File system benchmark - scaling









better

Scaling – Reasons

- Increased page cache aggressiveness
  - Rule of thumb – now about twice as aggressive
  - Higher integrity, but more background I/O operations and amount
  - One might want to tune dirty ratios in

    `/proc/sys/vm/dirty_*`
- CFS scheduler effects
  - It is striving for better interactivity and fairness
  - By that it can cause more cache misses like in this dbench runs
  - Viewed discrete that is more good than bad – workloads migth:
    - benefit from better scaling
    - benefit from better interactivity
    - benefit from better fairness
    - suffer by worse cache usage
  - Probably all of this will happen
  - depends on the workload to what extend which aspects are seen
  - While SLES11 already had CFS, scheduling further improved in SP1

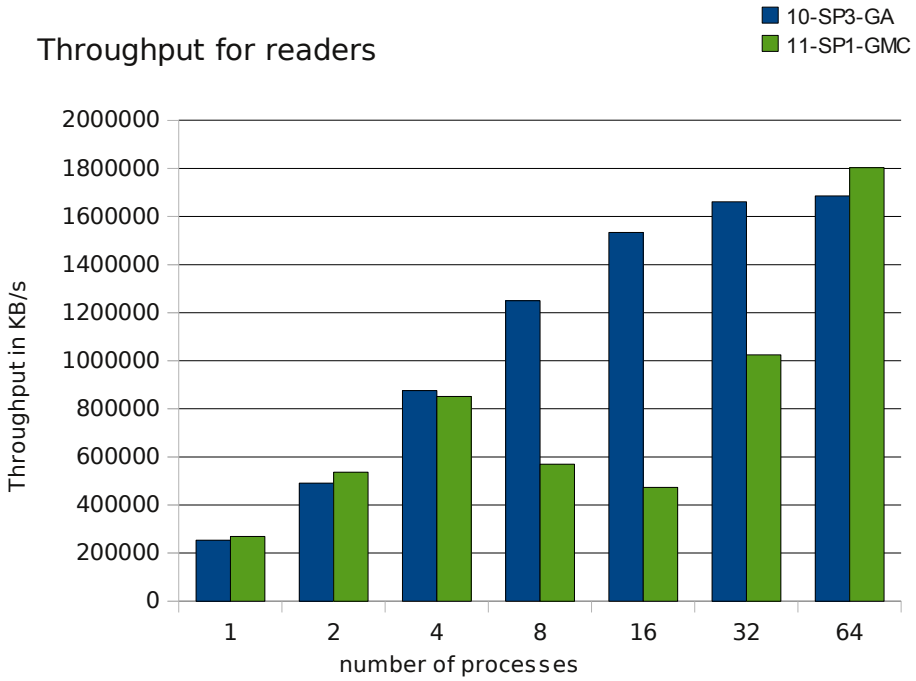
Tuning cpu/memory intensive workload

- To explicitly mark a load as CPU intensive use SCHED_BATCH
  - e.g. runs without a lot of I/O but calculates a lot in memory (e.g. some BI load)
  - Set via sched_setscheduler(pid, ...) from sched.h
    - schedtool not yet available in distributions
  - can be combined lowering the nice value
  - avoid some of the interactivity tunings
  - more deterministic policy
  - usually resulting in a better caching behaviour
- consider tunings of /proc/sys/kernel/sched_* for such loads as well
- Consider Setting /proc/sys/kernel/sched_compat_yield set to 1

Benchmark descriptions
Disk I/O

- **Workload**
  - Threaded I/O benchmark
  - Each process write or read to a single file, volume or disk
  - Benchmark can be configured to run with and without page cache (direct I/O)
  - Operating modes: Sequential write/rewrite/read + Random write/read
- **Setup**
  - Main memory was restricted to 256MB
  - File size (overall): 2GB, Record size: 64kB
  - Scaling over 1, 2, 4, 8, 16, 32, 64 processes
  - Sequential run: write, rewrite, read
  - Random run: write, read (with previous sequential write)
  - Once using page cache and once using Direct I/O (bypass the page cache)
  - Sync and Drop Caches prior to every invocation to reduce noise

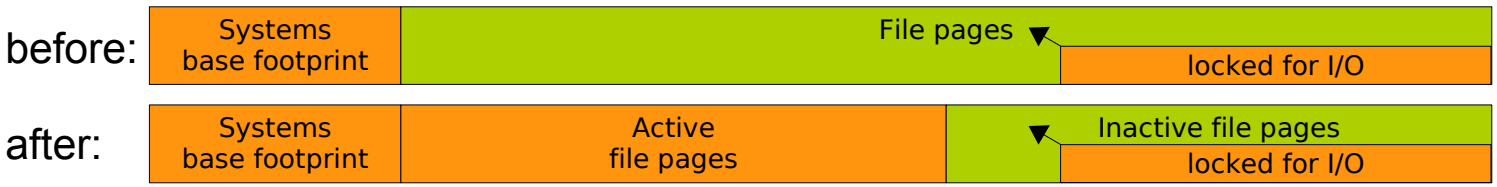
Disk I/O – Sequential Read Issue
(Sequential write/read with page cache)



- This is the mentioned memory management issue hitting streaming reads
- Throughput degrades up to -70%
- SLES11-SP1 includes a symptom reducing fix or it would be even worse

Disk I/O page cache based – Summary

- **Huge impact around 8 to 16 processes**
  - Newer kernels protect 50% of the so called active working set (active file pages)
    - That is actually performance tuning speeding up re-writes/re-reads
  - By that tight memory constraints might get tighter
  - Combined these can cause page cache allocation stalls
    - Almost all the system needs memory for is page cache for read ahead (GFP_COLD)
    - For highly parallel workloads a lot of ram is locked for in-flight I/O
    - Another 50% are protected for the active set
    - Not enough left to reclaim, leading to long in kernel loops



  - Only in very constrained systems, but …
    - Ballooning can let your systems slowly converge to that constraints

Disk I/O page cache based – Summary II

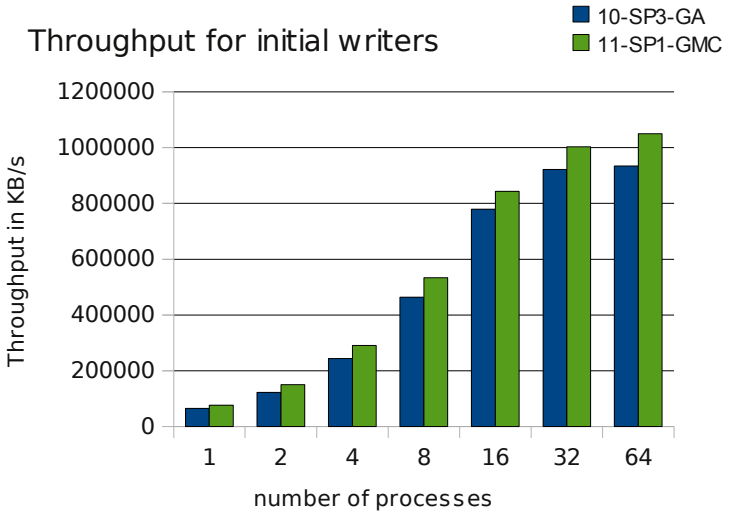
- **Detection**
  - Most workloads won't see the impact or even benefit from these changes
  - Backup jobs are expected that they might see the impact (multi disk seq. read)
  - Our page cache sequential read scenario is the paradigm how to hit the issue
  - Some simple verifications can be done with
    - should actually hurt throughput, huge improvements mean you are probably affected
    - Check e.g. syststat which should reports a huge amount of pgscand/s
    - No free memory and active/inactive ratio is at 1:1 is another requirement

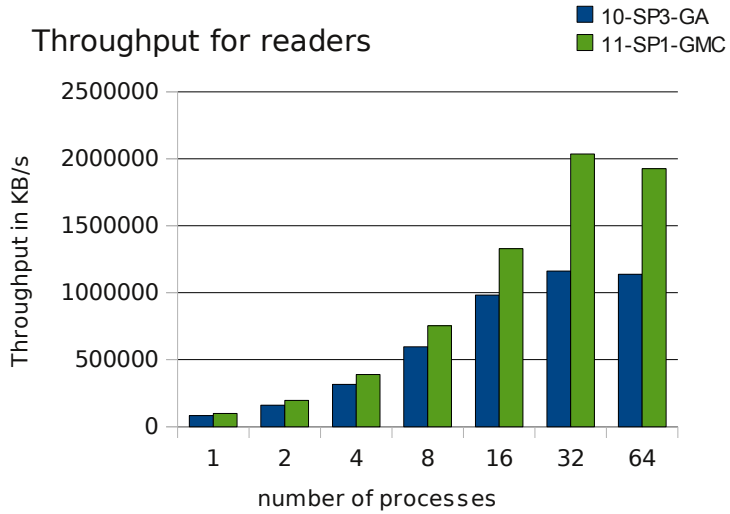
      run "`sync; echo 3 > /proc/sys/vm/drop_caches`"
    - should actually hurt throughput, huge improvements mean you are probably affected
- **Workarounds**
  - Increase available memory if possible
  - Drop caches if there is a single time this happens (e.g. on nightly backup)
  - Depending on the dependency for read-ahead - shrinking or disabling might help
  - Use direct I/O if applicable
  - Patch to tune the protected ratio via sysfs from IBM (not accepted upstream)

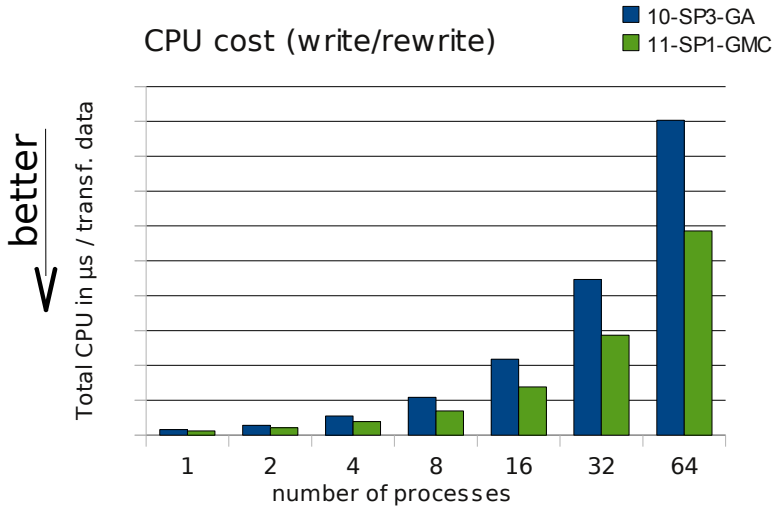
Disk I/O – New FICON features

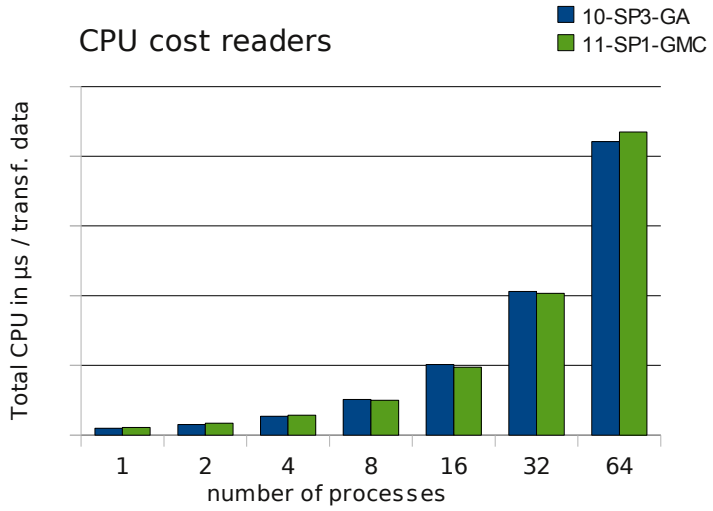
- HyperPAV
  - Avoid subchannel busy
  - automatic management of subchannel assignment/usage
  - No need of multipath daemon
  - Especially useful for concurrent disk accesses
- Read-Write Track Data
  - Allows reading up to a full track in one command word
  - Especially useful for huge requests and streaming sequential loads
- High Performance Ficon
  - New metadata format reduces overhead
  - Especially useful for small requests
- Setup on the following charts
  - HyperPAV uses 1 to 64 processes spread evenly on 4 disks – up to 16 per disk
  - added 3 aliases per Disk for HPAV, so effectively 4 subchannels per Disk
  - HPF/RWTD doesn't need any special setup at all

Disk I/O – Comparison on FICON Disks
(Sequential write/read with direct I/O)

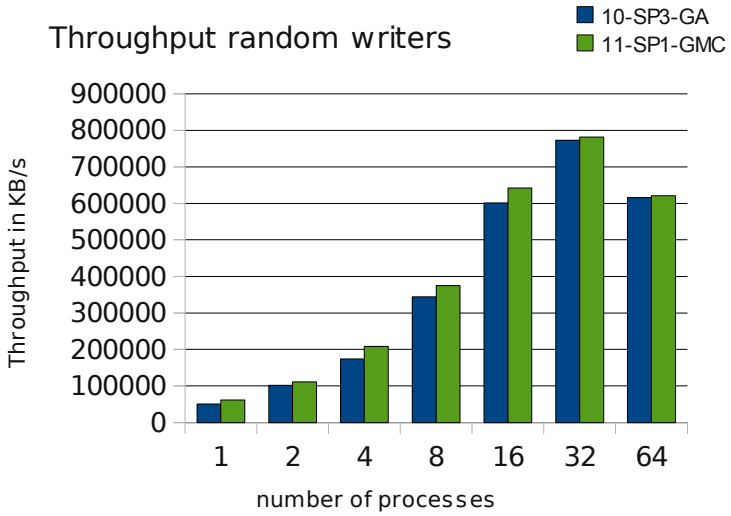


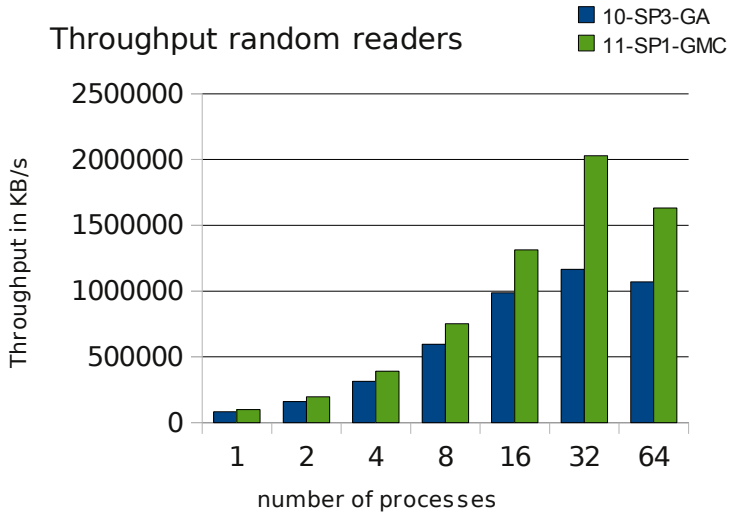


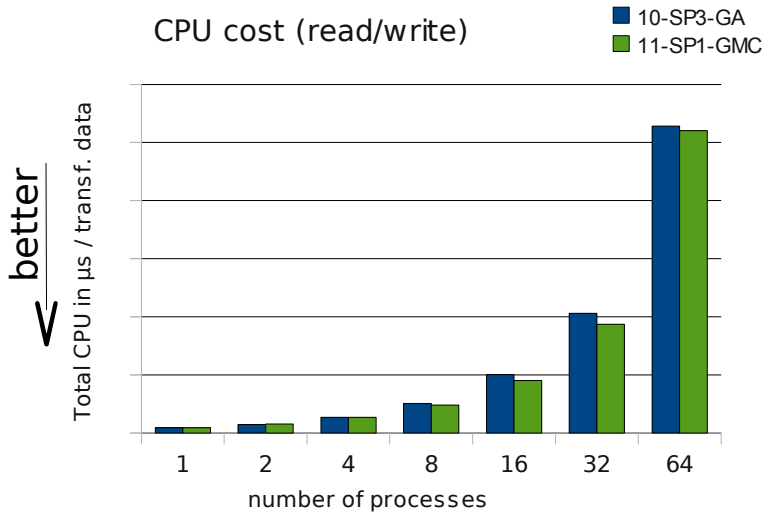




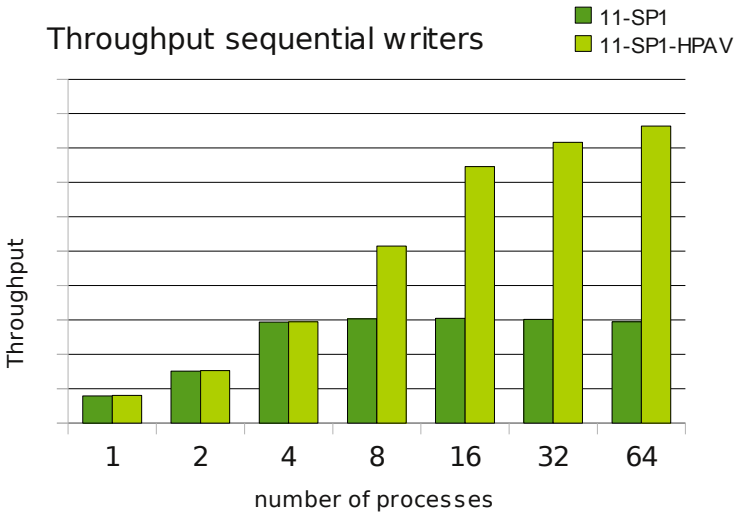
Disk I/O – Comparison on FICON Disks
(Random write/read with direct I/O)

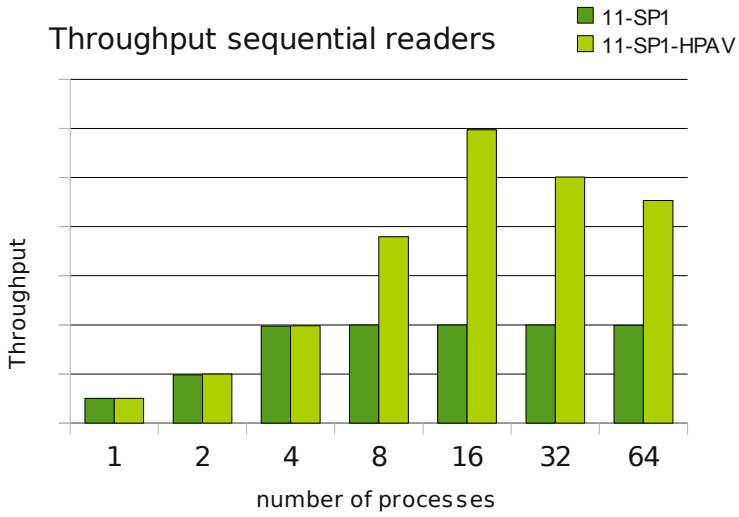


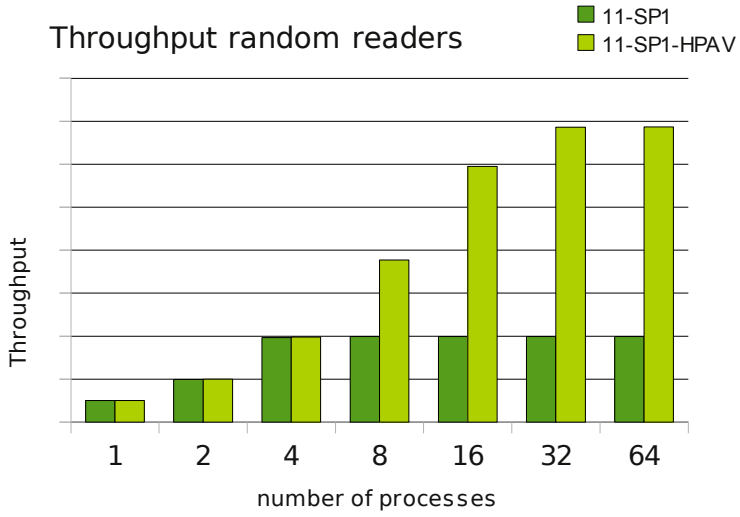




Disk I/O – HyperPAV







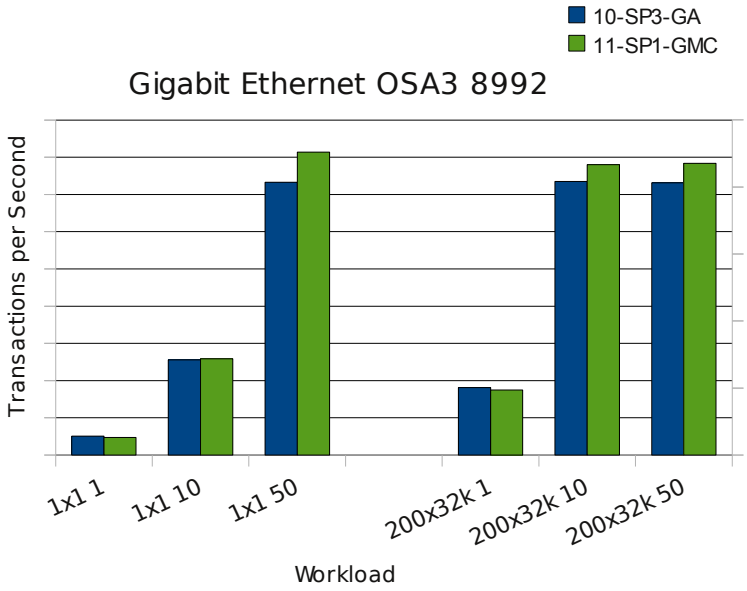
Benchmark descriptions
Network 1/2

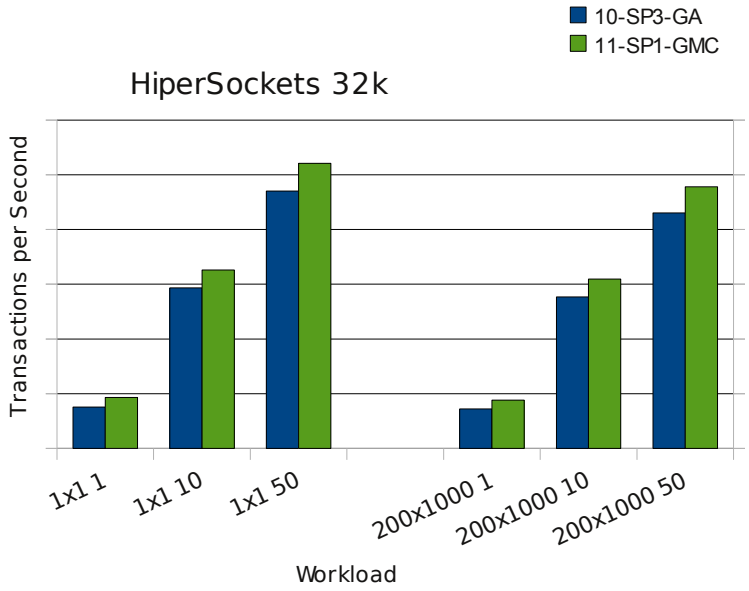
- **Network Benchmark which simulates several workloads**
  - All tests are done with 1, 10 and 50 simultaneous connections
  - Transactional Workloads – 2 types
    - RR – A connection to the server is opened once for a 5 minute timeframe
    - CRR – A connection is opened and closed for every request/response
  - Transactional Workloads – 4 sizes
    - RR 1/1 (send 1 byte from client and server and get 1 byte response
      Simulating low latency keepalives
    - RR 200/1000 (Client sends 200 bytes and get 1000 bytes response)
      Simulating online transactions
    - RR 200/32k (Client sends 200 bytes and get 32kb bytes response)
      Simulating website access
    - CRR 64/8k (Client send 64 bytes and get 8kb bytes response)
      Simulating database query
  - Streaming Workloads – 2 types
    - STRP - "stream put" (Client sends 20 mbytes and get 20 bytes response)
    - STRG - "stream get" (Client sends 20 bytes and get 20 mbytes response)
      Simulating large file transfers

Benchmark descriptions
Network 2/2

- Connection types
  - OSA 1 Gigabit Ethernet MTU sizes 1492 and 8992
  - OSA 10 Gigabit Ethernet MTU sizes 1492 and 8992
  - HiperSockets MTU size 32k
  - VSWITCH z/VM guest to guest MTU sizes 1492 and 8992 (z/VM only)
  - OSA 1 Gigabit Ethernet dedicated z/VM guest to Linux in LPAR MTU sizes 1492 and 8992
  - OSA 10 Gigabit Ethernet dedicated z/VM guest to Linux in LPAR MTU sizes 1492 and 8992
  - OSA 1 Gigabit Ethernet VSWITCH z/VM guest to Linux in LPAR MTU sizes 1492 and 8992
  - OSA 10 Gigabit Ethernet VSWITCH z/VM guest to Linux in LPAR MTU sizes 1492 and 8992

Network Throughput





- Single connection Latency can be an issue, but it is much better than in SLES11
- Scaling is good - parallel connection scenarios improved a lot
- For HiperSockets connections even latency improved

Hints - General

- Cgroup memory support
  - This is a feature coming with newer kernels
  - Recommended by some management tools to enforce very customizable memory constraints
  - Has a rather large footprint by consuming 1% of the memory
  - activated by default
  - In a consolidation environment it is actually 1% multiplied by your virtual/real ratio
  - not pageable by linux, but fortunately by z/VM
  - This can be overridden with a kernel parameter (reboot required):

```
cgroup_disable="memory"
```

Q&A

- SLES11-SP1 is a very huge improvement compared to SLES11
- With some trade-offs it is roughly as good or better than SLES10-SP3
  - In addition it has about two thousand new features compared to SLES10
  - By that it is the first performance acceptable “modern” Distribution

Questions ?

Backup - Major Changes worth to talk about

- I/O related
  - Read-ahead scaling
  - Proper unplugging
  - BLKT settings
- CPU Cost related
  - Generic timer infrastructure
  - I/O cost optimizations
  - Function in-lining
- Already explained in detail in relation to some benchmark results
  - Dirty Pages tracking
  - Active Set protection
  - High Performance Ficon / Read Write Track Data
  - Page cache aggressiveness
  - HyperPAV
  - cgroup

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product and service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates.