KVM on s390: The good, the bad and the weird

Cornelia Huck, IBM Deutschland Research & Development GmbH
Co-maintainer s390/KVM and s390x/qemu
2016/08/25

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

AIX*	Domino*	GPFS	Informix*	MQ*	SPSS*	XIV*	z Systems
BlueMix	DS8000*	HiperSockets	InfoSphere	POWER*	Storwize*	z/Architecture*	z/VSE*
BigInsights	ECKD	IBM*	LinuxONE	POWER*8*	System z9*	z13	z/VM*
CICS*	FileNet*	Ibm.com	LinuxONE Emperor	PR/SM	Systemz10*	z13s	
Cognos*	FlashSystem	IBM (logo)*	LinuxONE Rockhopper	RACF*	Tivoli*	zEnterprise*	
DB2*	GDPS*	IMS	Maximo*	Spectrum Scale*	WebSphere*	z/OS*	

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

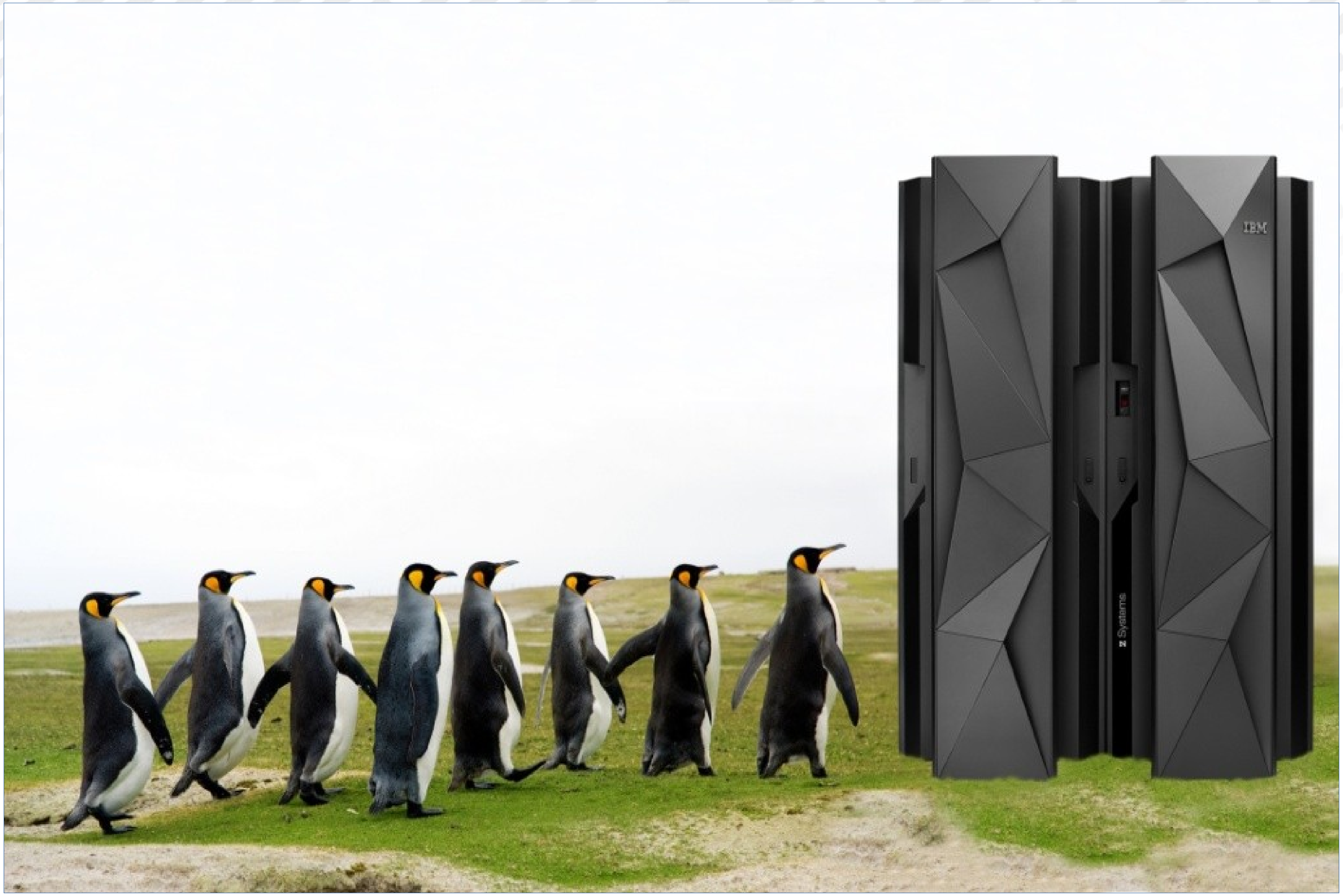
Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.
IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.
Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
Windows Server and the Windows logo are trademarks of the Microsoft group of countries.
ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.
Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

* Other product and service names might be trademarks of IBM or other companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.
All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the “Authorized Use Table for IBM Machines” provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html (“AUT”). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

What's this about?



What's this about?

- s390 (aka z Systems®) was the second architecture to implement KVM
- First with a custom userspace (kuli), then with qemu
- KVM on s390 exploits some neat architecture features...
- ...but also had to deal with some decisions that sounded good at the time...
- ...and some rather odd things that are different from everybody else

Let's get started

SIE – Start Interpretive Execution

Let's get started

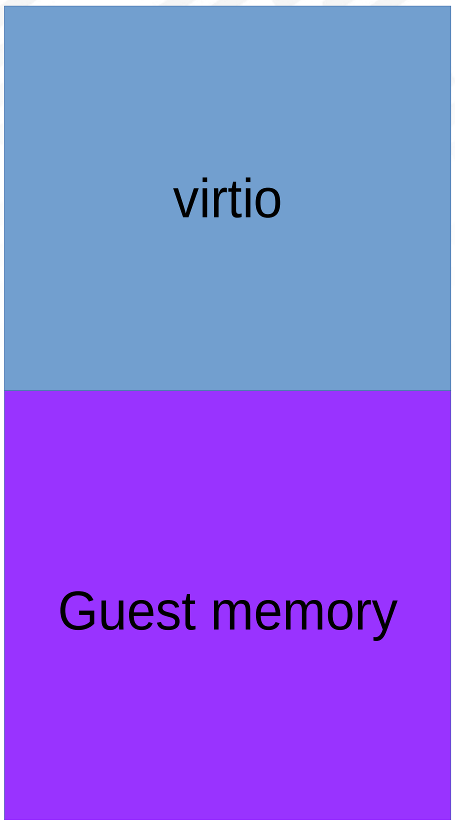
- SIE uses per-vcpu control blocks in host memory
  - ...this is nice for nested virtualization
- Satellite control blocks for some assists
- Intercept controls to enable manual interpretation
- Cool feature: ibc to fence back to a previous machine generation
- Intercept requests to get a vcpu out of the SIE
  - Headscratcher: We can request exit for stop, I/O and external – but not for machine checks
- Various SIE exits: instruction, program interrupt, idle...
- ...but mostly mapped to the same exit code in KVM

Let's get started

- (Nearly) everything used to be mapped to a single SIE exit reason
- Drawbacks: we need to fetch state, as we don't know what we need to handle the intercept
  - Instruction intercepts, wait states or program checks all need different status
- New 'specialist' exit codes (for handling of tsch, stsi, …)
- ...but a far cry from the variety of exit reasons on other architectures

Channels and paths and programs

Channels and paths and programs



diagnose 500

external interrupt

stsch

cc 3

chsc

cc 3

Channels and paths and programs

diagnose 500

I/O interrupt

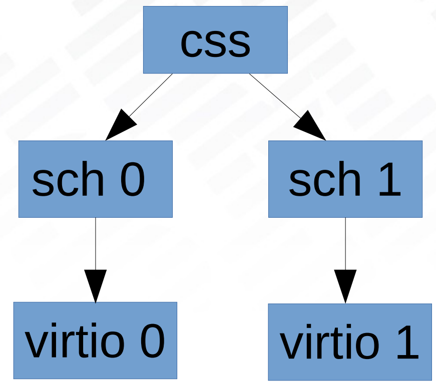
stsch

chsc

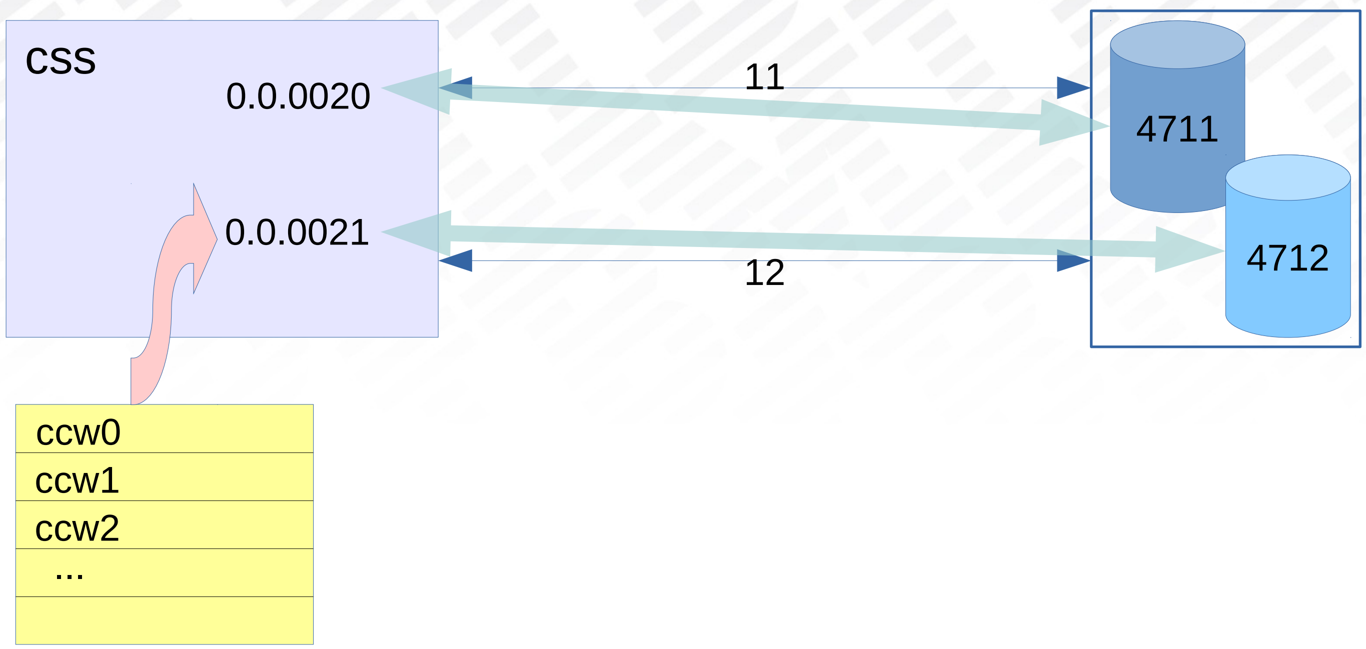
ssch

tsch

...



Channels and paths and programs



Channels and paths and programs

- stsch, msch, tsch – deal with device descriptions
- ssch, rsch, hsch, csch, xsch – deal with channel programs
- chsc – deal with a whole lot of things
- Neat features:
  - All I/O instructions are mandatory intercepts
  - Common set of architectural descriptions for all devices
  - All I/O devices can describe themselves

Channels and paths and programs

- virtio – the easy case
  - Fully virtual channel subsystem
  - Channel paths do nothing
- Passthrough (vfio) and emulation is more complicated
  - Need 'real' channel paths
  - Some refactoring to accommodate non-virtio devices
  - Vfio-ccw would be a talk in itself

It's PCI, but not as you know it

- PCI is a relative newcomer to the s390
- Only certain cards supported (RoCE, Flash, Compression)
- Needed to fit with existing paradigms
- No MMIO!
- Various instructions for reading/writing memory
- Integration into existing I/O infrastructure (adapter interrupts, channel-subsystem machine checks)
- ...and NO topology information!

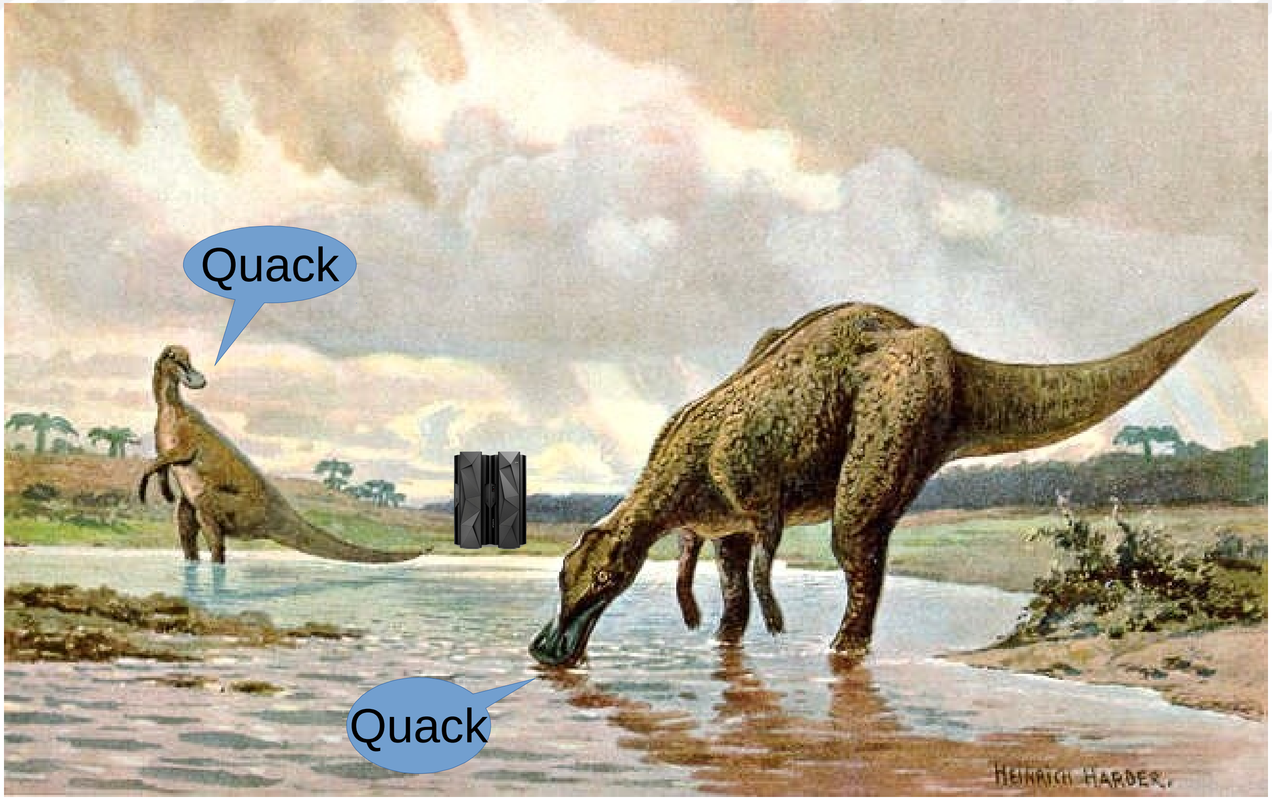
It's PCI, but not as you know it

You want	You use	You get
All PCI functions and their configuration	CLP List PCI Functions and CLP Query PCI Function	List of functions with FH, FID, UID, BARs and DMA values – NO bus/slot/function topology!
Read/write PCI config space	PCI LOAD and PCI STORE	Access to the config space – via privileged instructions!
MSI interrupts	Adapter interrupts and indicators	Message encoded with function index and indicator offset
Hot(un)plug notifications	Machine-check notified events	Information extracted via a channel-subsystem call – but it is still PCI-specific information

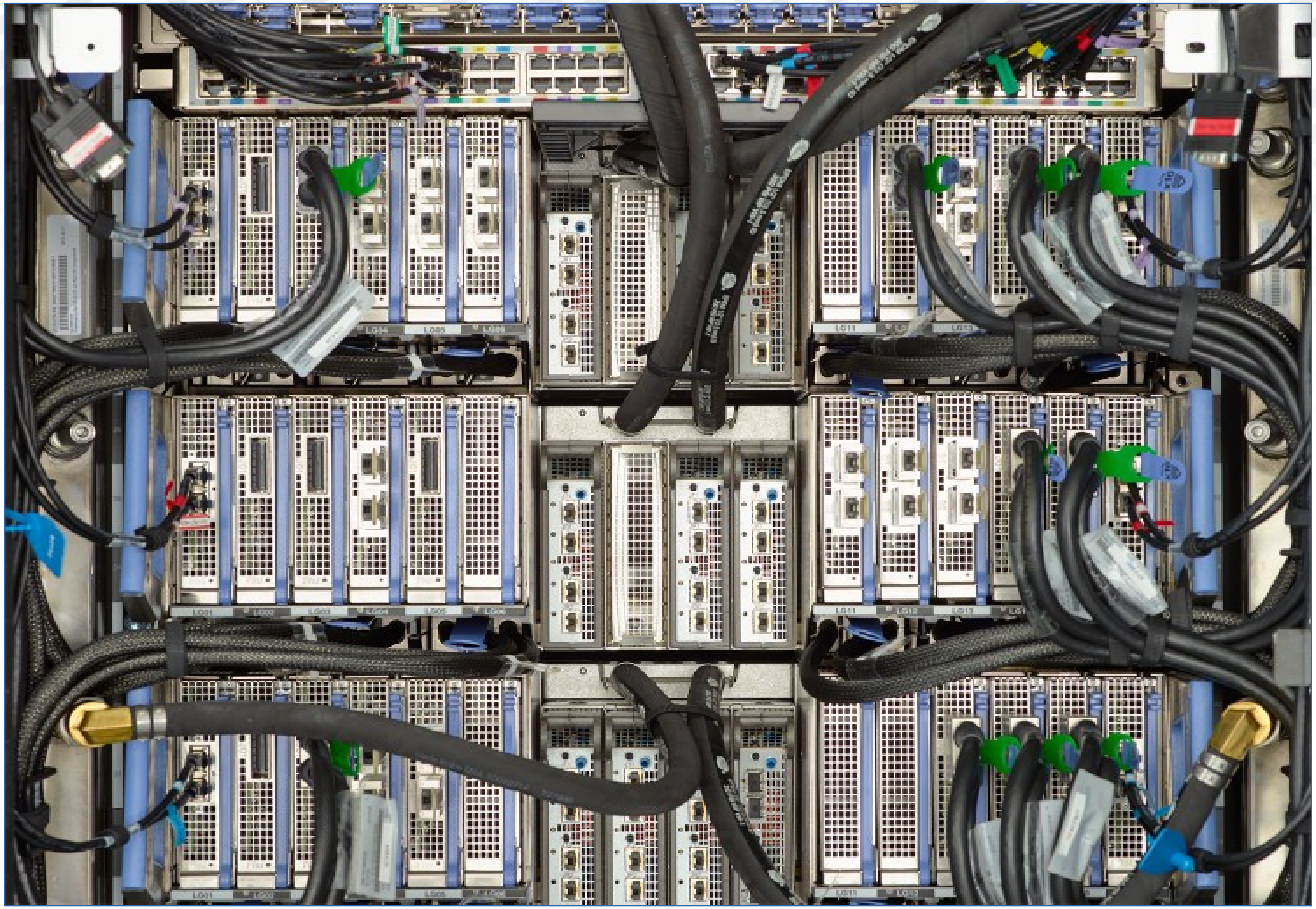
It's PCI, but not as you know it

- Linux guest side integration worked quite well, but...
- ...host side modelling in qemu was not that easy
- Challenge: Reconcile qemu's topology-based modelling with zPCI's information
- Solution: Build a 'fake' topology, add satellite zpci devices to store s390-specific information (fid, uid)

It's PCI, but not as you know it



The changing ways of SIGP



The changing ways of SIGP

- SIGP – Signal Processor
- First implementation: partly in the kernel, partly in userspace
  - This did not play well with keeping cpu state in qemu...
  - ...and was racy between SIGPs
- Moved to userspace, guarded by a capability
  - Privileged program exceptions still handled in the kernel
  - Exception: 'fast' SIGPs which also need access to kernel state
  - But we have to keep the old code around...
- Neat architecture feature: SIGP interpretation
  - For a subset of SIGP calls, we can let the SIE handle it
  - Exitless signalling of other vCPUs → win

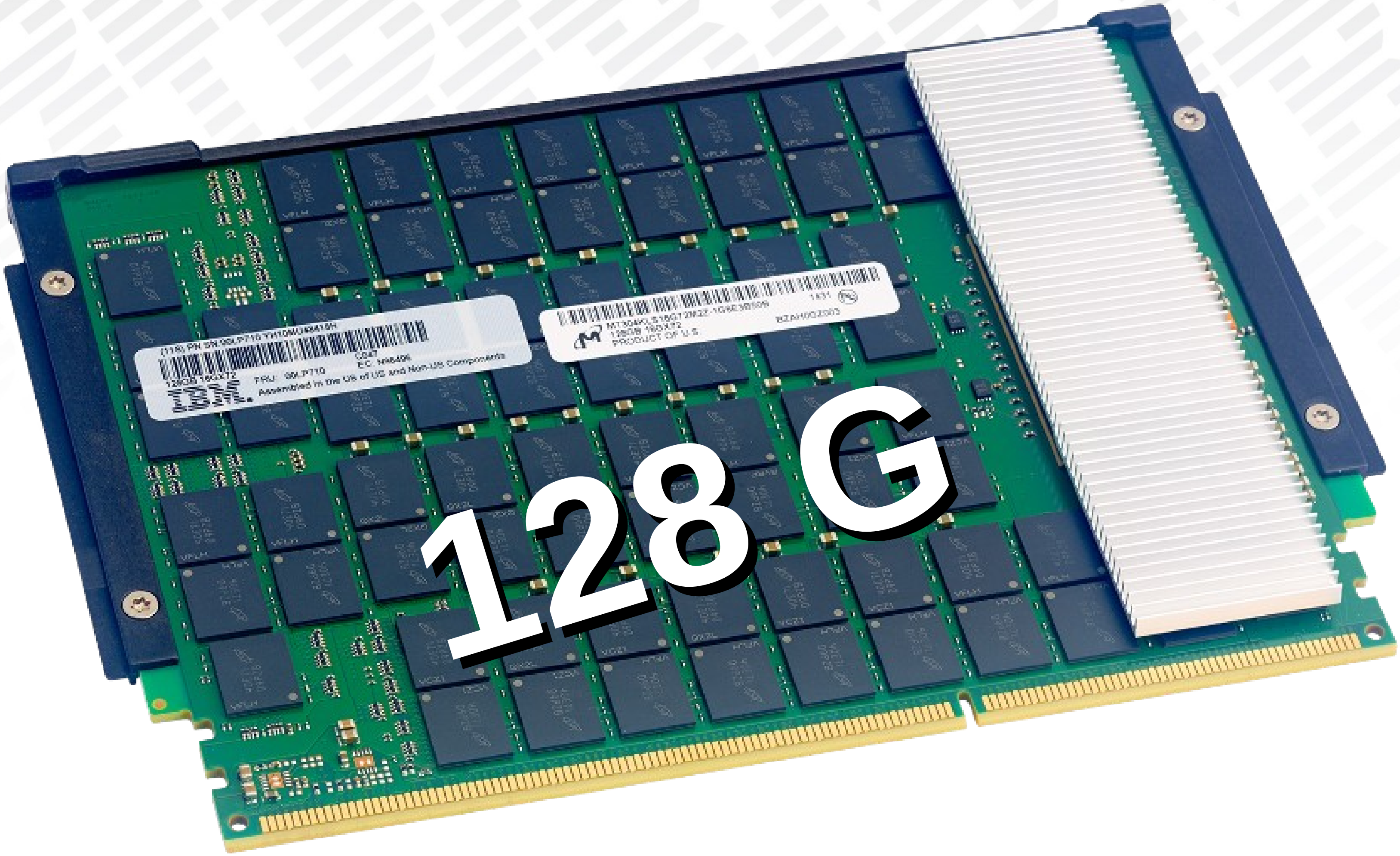
The call that wants to be a processor

- SCLP – service-call logical processor
- Takes on many tasks performed by a PC's BIOS/UEFI
- ...but it is a well-specified interface
- In practice, we emulate it as a simple call
  - Send a control block (SCCB), get an (external) interrupt on completion
- Supported features vary with the machine generation

The call that wants to be a processor

- Provide information about the machine
  - Number of cpus, amount of memory, …
  - ...and a list of facilities that is not completely distinct from cpu facilities
- Allow to dynamically change the machine's current configuration
  - Activate standby cpus, deactivate PCI functions, ...
- Implement a console
  - VT-220 compatible or line mode
- All of this is best implemented in userspace
- qemu models this as a device hierarchy

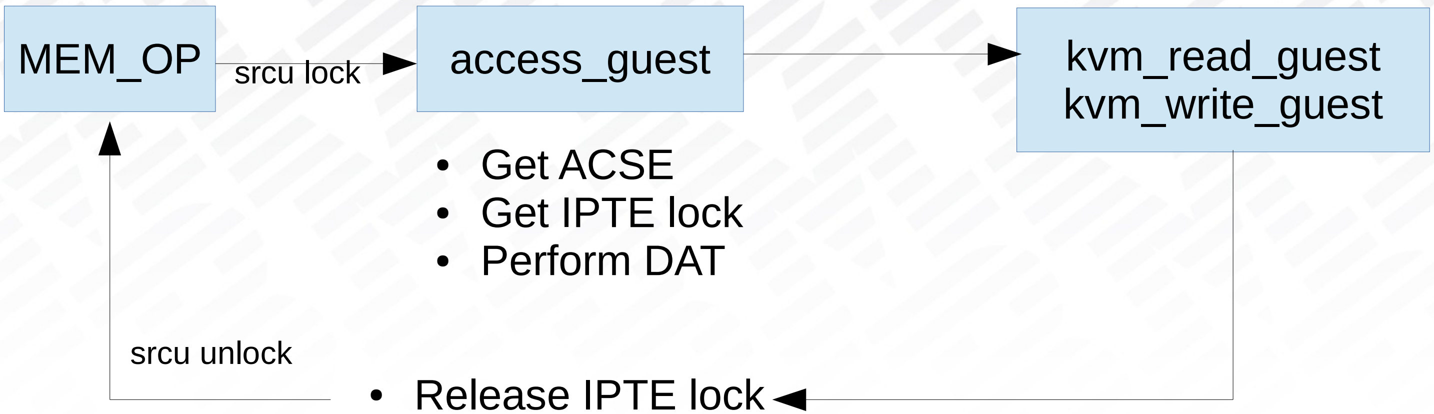
Let's talk about memory



Let's talk about memory

- We need host kernel support to read/write memory in a correct way
- Reasons:
  - IPTE lock (for DAT)
    - To synchronize against page table changes by the SIE
    - Contained in SIE control block
  - Possible storage key operations
- Solution: introduce an IOCTL (KVM_S390_MEM_OP)

Let's talk about memory



Note:

- Program checks may happen
- Key protection currently not implemented

Check arch/s390/kvm/gaccess.c for the gory details

Thank you!