The software platform: a problem of choice

Andrey Kiryanov, NRC KI – PNPI

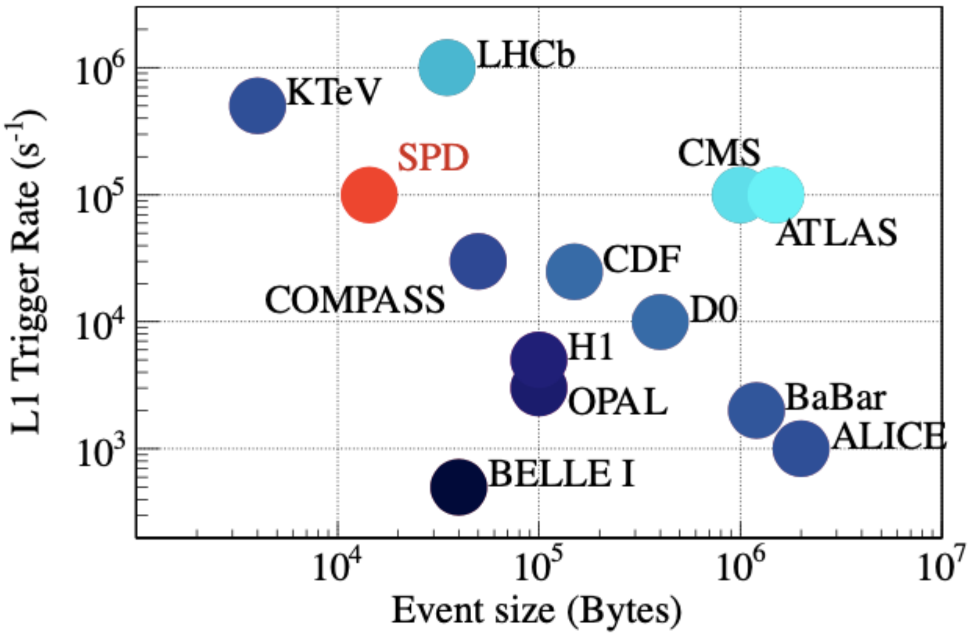
SPD Collaboration Meeting, 3-6 October 2022

Introduction

The expected event rate of the SPD experiment is about 3 MHz (pp collisions at $\sqrt{s} = 27$ GeV and 10^{32} $cm^{-2}s^{-1}$ design luminosity). This is equivalent to a raw data rate of 20 GB/s or 200 PB/year, assuming a detector duty cycle is 0.3, while the signal-to-background ratio is expected to be on the order of 10^{-5}. Taking into account the bunch-crossing rate of 12.5 MHz, one may conclude that pile-up probability cannot be neglected.

SPD TDR

Because of the data volume SPD will require some sort of a distributed infrastructure for offline storage and compute



Distributed computing

- Resources are to be provided by collaboration participants
- Needs a stable set of software solutions, protocols, etc. in the long run
  - AAA – authentication, authorization and accounting (centralized at JINR)
  - Storage – protocols and data layout (may be deployed differently by different participants as long as it's transparent for the users)
  - Compute – all participants must agree on a common platform for offline software, otherwise validation, support and problem solving will become a nightmare

What is a software platform?

- CPU Architecture. Most of the high-performance computing facilities nowadays are 64-bit Intel/AMD with vector extensions (SSE, AVX)
  - GPGPU accelerators are becoming very popular
  - Things are slowly drifting towards ARM64, but are not quite there yet
- Operating system. Both HPC/HTC and university clusters are dominated by Linux kernel-based OSes, but distributions vary
- Software environment (libraries). Basic things like libC, libM, OpenSSL, but also versions of C/C++ compilers (GCC, ICC, LLVM/Clang), Python distribution, etc.
- Deployment methods (packages or CVMFS-like)
- Finally, application software (like Athena or Root)

An example: WLCG

- Started in 32-bit world, transitioned to 64-bit
- Have gone through several versions of LTS RHEL-based distributions starting from Scientific Linux 4, now at CentOS 7 (6+4-year lifetime 2014-2024)
  - Was about to switch to CentOS 8, but long-term support was essentially dropped by RedHat
  - The future is still uncertain, CentOS 8 lifetime is too short
- Initially LHC experiments had very different ideas about software versions and distribution model which caused lots of deployment issues, but finally all of them switched to CVMFS
  - Requires access to a local HTTP proxy from the worker nodes
  - CVMFSexec allows unprivileged use

A brief look at Linux distribution landscape

- There are tons of distros, but there are only two major realms with business-class feature-reach LTS distributions: RedHat and Debian
  - Filesystem layout is alike, but still different, especially in /etc and /var
  - Different approach on system software (initialization, firewall, etc)
  - Different package and update managers (yum+rpm for RedHat and apt+dpkg for Debian)
  - Incompatible package repositories
  - Historically, RedHat-based distros are more popular in scientific world

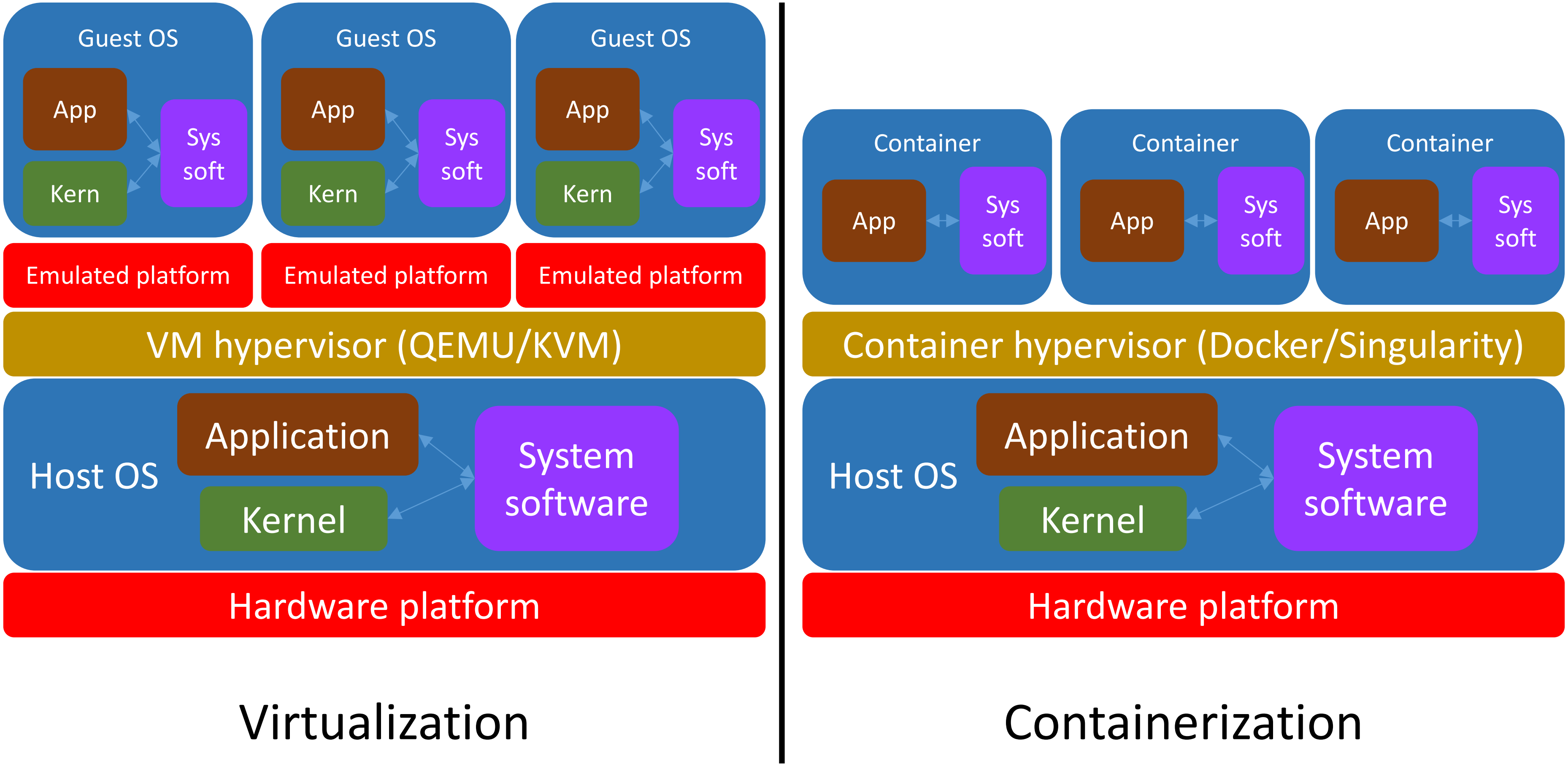
Consider we've chosen a distro…

- Over the years there will be library updates even within a single distro
  - Major versions usually do not change to preserve an ABI
  - This is only true for libraries and tools bundled with the distribution, external community repositories do not impose this limitation, which may cause software malfunction after an update
  - You cannot force everybody to update at the same time
  - This makes physics application software validation really hard. Different sites may have different versions of libraries even within a single distro
  - That's why most LHC experiments bundle EVERYTHING, including compilers, libraries, etc. even if it is already included in the distribution

Stable software platform

- How do we ensure we have fixed versions of everything without micromanaging all collaboration resources?
  - Distribute everything via CVMFS
    - Including all the libraries
    - Hard to manage
    - May cause standard system tools to crash (surprise!)
  - Run pre-made VM images
    - OK for cloud providers, not OK for most university clusters
    - Unnecessary overhead
  - Run application software in containers
    - Docker!

The shiny world of containers



Docker containers

- A very successful commercial multi-platform (yes, it runs on Windows and Mac OS) container platform
- Free for personal/non-commercial use
- Bound to an on-line container repository DockerHub
- Requires administrative privileges
- Containers are single-file images with r/w filesystem

Can we run Docker containers without Docker?

- There's Singularity!
- BSD license (free software)
- Compatible with Docker file format, but mounts filesystem r/o by default
- Can run from an unprivileged user account
- Can run containers from within containers! (hence the name, but I doubt we will use this feature)

Singularity highlights

- Singularity is used successfully by LHC experiments like ATLAS
- Simplifies software deployment
- ~Zero performance overhead
- Trivial configuration from resource providers (one needs to enable user namespaces)
- Explicitly supported by modern schedulers like Slurm
- Containers are the easiest way to contain Python dependency hell and C++ build hell
- Complex software suites like Zabbix and Ceph are actively switching from distribution-dependent repositories to containers

What can we do with containers

- Build everything with modern suitable compilers
  - CentOS 7 is shipped with GCC 4.8 which cannot emit AVX-512 code at all
  - No need to worry about C++ ABI
  - Still requires support from resource providers wrt kernel and singularity patches
- Bundle all the necessary software libraries
  - No dependency on what's available in the distribution or external repositories
- Drop OS dependency completely, let resource providers decide but provide reasonable guidelines
  - Educational institutes may be forced to deploy "Russian flavors" of Linux (ALT Linux)

Hardware requirements

- As long as we ship our software in compiled form we need to ensure that hardware has all the necessary features
  - Basic instruction set
    - Almost any Intel/AMD CPU produced in the last decade will do
  - SSE, AVX, etc.
    - Must be careful, this is not plug-and-play and software will crash on invalid opcode
    - Intel/AMD have some incompatible extensions
    - Performance measurements need to be conducted in order to decide on the least necessary set
  - Available RAM per core
  - Disk scratch space

How do we access computing resources?

- Resource providers will need to have some sort of an endpoint where the Workload Management System (Panda?) could submit jobs
  - ARC CE
  - HTCondor CE
  - Maybe even more lightweight solution since no intelligence is required (Slurm REST interface?)
- We hope that tokens will replace X.509 certificates

What about storage?

- EOS seems to be the most reasonable choice
  - Can handle disks, SSDs and tapes
  - Available for RHEL and Ubuntu
  - Can be containerized as well
- Need to decide on a minimal storage size and network connectivity
  - Allowing 10 TB storages over 1 Gbps link is not practical
- Centralized data management via Rucio/FTS

Security considerations

- We must have our own infrastructure for everything
  - Git (do not rely on GitHub or CERN GitLab)
  - CI (like Jenkins) – software builds and tests must be automated
  - OS repositories (mirrors)
- Access from Russian IP ranges may be banned
  - Already the case for Elasticsearch repositories, Dell, Intel, etc.
- Code-bombs in upstream is the sad reality
  - Do not blindly pull the updates

Thank you!

Bonus: XOP instruction set drama

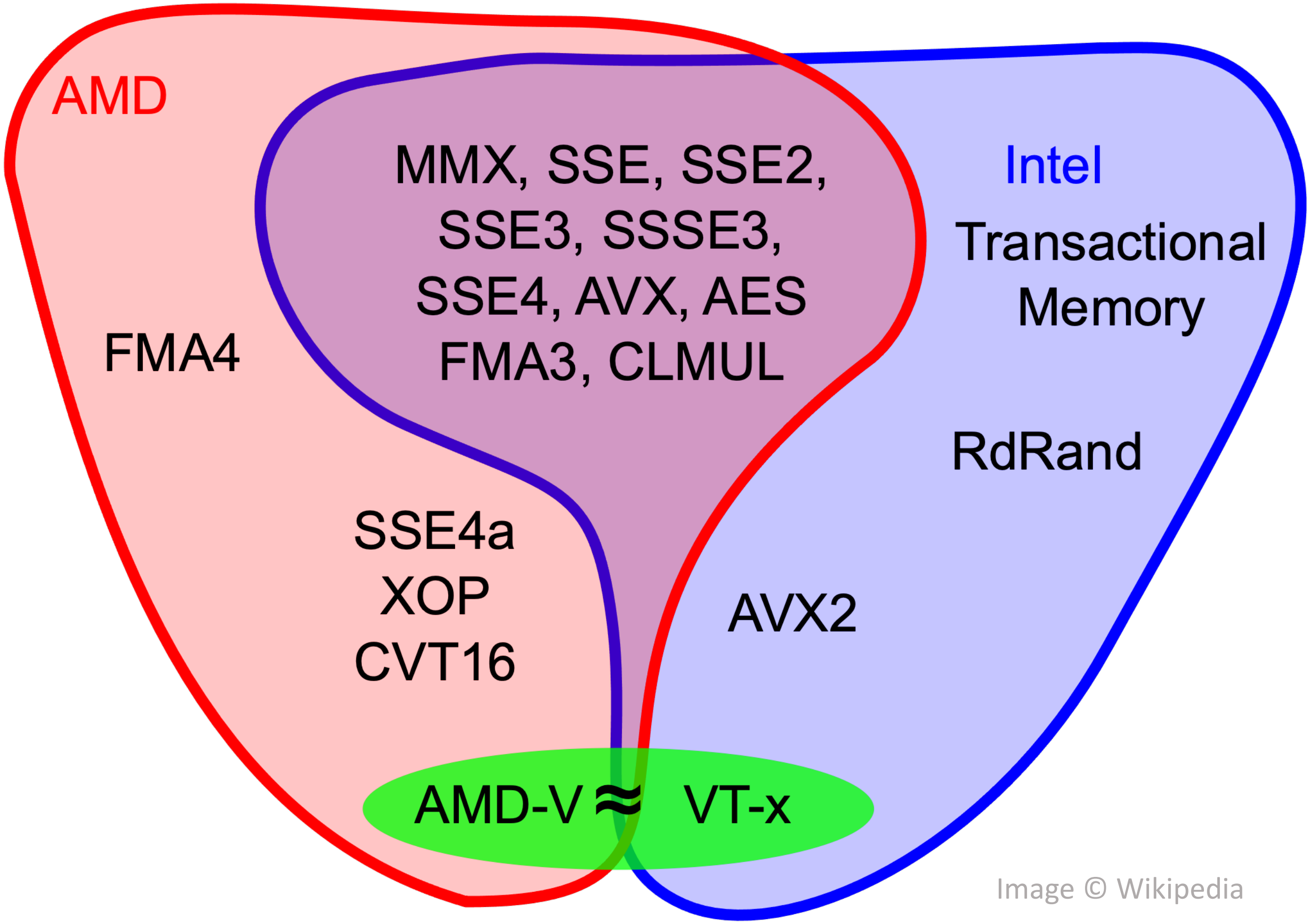


Image © Wikipedia