CS 5220: VMs, containers, and clouds

David Bindel

2017-10-12

Cloud vs HPC

Is the cloud becoming a supercomputer?

- What does this even mean?
- Cloud $\approx$ resources for rent
    - Compute cycles and raw bits, or something higher level?
    - Bare metal or virtual machines?
    - On demand, behind a queue?
- Typically engineered for *different loads*
    - Cloud: high utilization, services
    - Super: a few users, big programs
- But the picture is complicated...

Choosing a platform

Questions to ask

- What type of workload do I have?
    - Big memory but modest core count?
    - Embarassingly parallel?
    - GPU friendly?
- How much data? Data transfer is not always free!
- How will I interact with the system? SSH alone? GUIs? Web?
- What about licensed software?

Standard options beyond the laptop

- Local clusters and servers
- Public cloud VMs (Amazon, Google, Azure)
    - Can pay money or write proposal for credits
- Public cloud bare metal (Nimbix, Sabalcore, PoD)
    - Good if bare-metal parallel performance an issue
    - Might want to compare to CAC offerings
- Supercomputer (XSEDE, DOE)

Topics du jour

- Virtualization: supporting high utilization
- Containers: isolation without performance hits
- XaaS: the prevailing buzzword soup

Virtualization

All problems in computer science can be solved by another level of indirection.

– David Wheeler

From physical to logical

- OS: Share HW resources between processes
  - Provides processes with HW abstraction
- Hypervisor: Share HW resources between virtual machiens
  - Each VM has independent OS, utilities, libraries
  - Sharing HW across VMs improves utilization
  - Separating VM from HW improves portability

Sharing HW across VMs is key to Amazon, Azure, Google clouds.

The Virtual Machine: CPU + memory

- Sharing across processes with same OS is old
    - OS-supported pre-emptive multi-tasking
    - Virtual memory abstractions with HW support
        - Page tables, TLB
- Sharing HW between systems is newer
    - Today: CPU virtualization with near zero overhead
        - Really? Cache effects may be an issue
    - Backed by extended virtual memory support
        - DMA remapping, extended page tables

The Virtual Machine: Storage

- Network attached storage around for a long time
- Modern clouds provide a blizzard of storage options
- SSD-enabled machines increasingly common

The Virtual Machine: Network

- Hard to get full-speed access via VM!
    - Issue: Sharing peripherals with direct memory access?
    - Issue: Force to go through TCP, or go lower?
- HW support is improving (e.g. SR-IOV standards)
- Still a potential pain point

The Virtual Machine: Accelerators?

I don’t understand how these would be virtualized!
But I know people are doing it.

Hypervisor options

- Type 1 (bare metal) vs type 2 (run guest OS atop host OS)
    - Not always a clear distinction (KVM somewhere between?)
    - You may have used Type 2 (Parallels, VirtualBox, etc)
- Common large-scale choices
    - KVM (used by Google cloud)
    - Xen (used by Amazon cloud)
    - HyperV (used by Azure)
    - vmWare (used in many commercial clouds)

Performance implications: the good

VMs perform well for many workloads:

- Hypervisor CPU overheads pretty low (absent sharing)
- May be within a few percent on LINPACK loads
- VMWare agrees with this
- Virtual memory (mature tech) extending appropriately

Performance implications: the bad

Virtualization does have performance impacts:

- Contention between VMs has nontrivial overheads
- Untuned VMs may miss important memory features
- Mismatched scheduling of VMs can slow multi-CPU runs
- I/O virtualization is still costly

Does it make sense to do big PDE solves on VMs yet? Maybe not, but…

Performance implications

VM performance is a *fast* moving target:

- VMs are important for isolation and utilization
    - Important for economics of rented infrastructure
- Economic importance drives a lot
    - Big topic of academic systems research
    - Lots of industry and open source R&D (HW and SW)

Scientific HPC *will* ultimately benefit, even if not the driver.

VM performance punchline

- VM computing in clouds will not give “bare metal” performance
    - If you have 96 vCPUs and 624 GB RAM, maybe you can afford a couple percent hit?
- Try it before you knock it
    - Much depends on workload
    - And remember: performance comparisons are hard!
    - And the picture will change next year anyhow

Containers

Why virtualize?

A not-atypical coding day:

1. Build code (four languages, many libraries)
2. Doesn’t work; install missing library
3. Requires different version of a dependency
4. Install new version, breaking different package
5. Swear, coffee, go to 1

Application isolation

- Desiderata: Codes operate independently on same HW
    - Isolated HW: memory spaces, processes, etc (OS handles)
    - Isolated SW: dependencies, dynamic libs, etc (OS shrugs)
- Many tools for isolation
    - VM: strong isolation, heavy weight
    - Python virtualenv: language level, partial isolation
    - Conda env, modules: still imperfect isolation

Application portability

- Desiderata: develop on my laptop, run elsewhere
    - Even if “elsewhere” refers to a different Linux distro!
- What about autoconf, CMake, etc?
    - Great at finding *some* library that satisfies deps
    - Maintenance woes: bug on a system I can’t reproduce
- Solution: Package code and deps in VM?
    - But what about performance, image size?

Containers

- Instead of virtualizing HW, virtualize OS
- Container image includes library deps, config files, etc
- Running container has own
    - Root filesystem (no sharing libs across containers)
    - Process space, IPC, TPC sockets
- Can run on VM or on bare metal

Container landscape

- Docker dominates
- rkt is an up-and-coming alternative
- Several others (see this comparison)
- Multiple efforts on containers for HPC
    - Shifter: Docker-like user-defined images for HPC systems
    - Singularity: Competing system

Containers vs VMs?

- VMs: Different OS on same HW
    - What if I want Windows + Linux on one machine?
    - Good reason for running VMs locally, too!
- VMs: Strong isolation between jobs sharing HW (security)
    - OS is supposed to isolate jobs
    - What about shared OS, one malicious user with root kit?
    - Hypervisor has smaller attack surface
- Containers: one OS, weaker isolation, but lower overhead

XaaS and the cloud

IaaS: Infrastructure

- Low-level compute for rent
    - Computers (VMs or bare metal)
    - Network (you pay for BW)
    - Storage (virtual disks, storage buckets, DBs)
- Focus of the discussion so far

PaaS: Platform

- Programmable environments above raw machines
- Example: Wakari and other Python NB hosts

SaaS: Software

- Relatively fixed SW package
- Example: GMail

The big three

- Amazon Web Services (AWS): first mover
- Google Cloud Platform: better prices?
- Microsoft Azure: only one with Infiniband

The many others: HPC IaaS

- RedCloud: Cornell local
- Nimbix
- Sabalcore
- Penguin-on-Demand

The many others: HPC PaaS/SaaS

- Rescale: Turn-key HPC and simulations
- Penguin On Demand: Bare-metal IaaS or PaaS
- MATLAB Cloud: One-stop shopping for parallel MATLAB cores
- Cycle computing: PaaS on clouds (e.g. Google, Amazon, Azure)
- SimScale: Simulation from your browser
- TotalCAE: Turn-key private or public cloud FEA/CFD
- CPU 24/7: CAE as a Service