A Tale of Two IT Universes:
A case study in unifying HPC & enterprise workloads

Mitch Richling

2020-08-18

A Play in 5 Parts

1) The **big** idea
2) TI EDA HPC vs General HPC vs Enterprise IT
3) The IT challenges
4) How this methodology relates to the cloud
5) An even ***BIGGER*** idea

Part 1

The *BIG* idea!

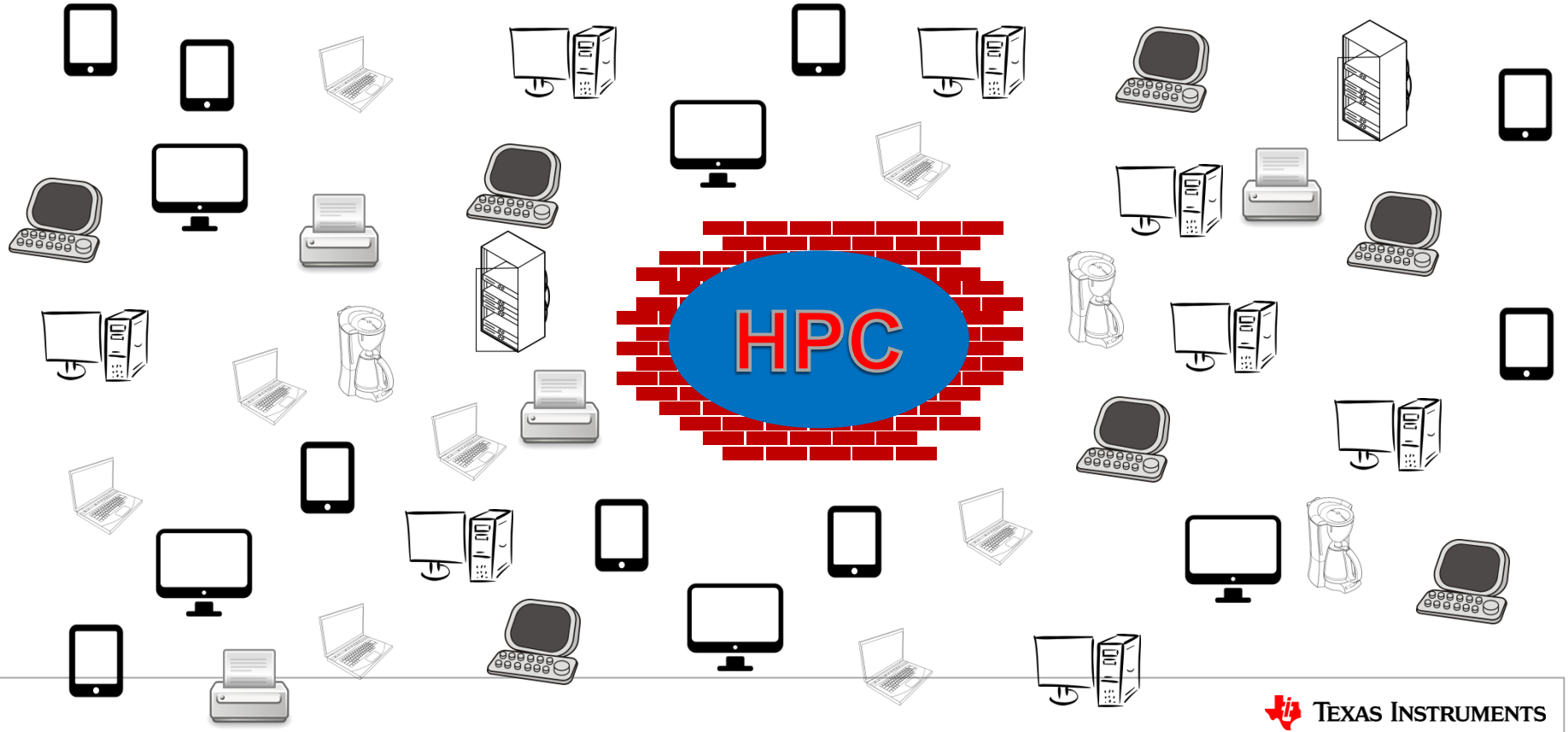
One day my boss said to me…

We always need more HPC compute, but we have thousands and thousands of underutilized CPUs all over the place – factory equipment, VM farms, phones, printers, thermostats, coffee makers!!

I want you to figure out how to run HPC jobs on every CPU we have!

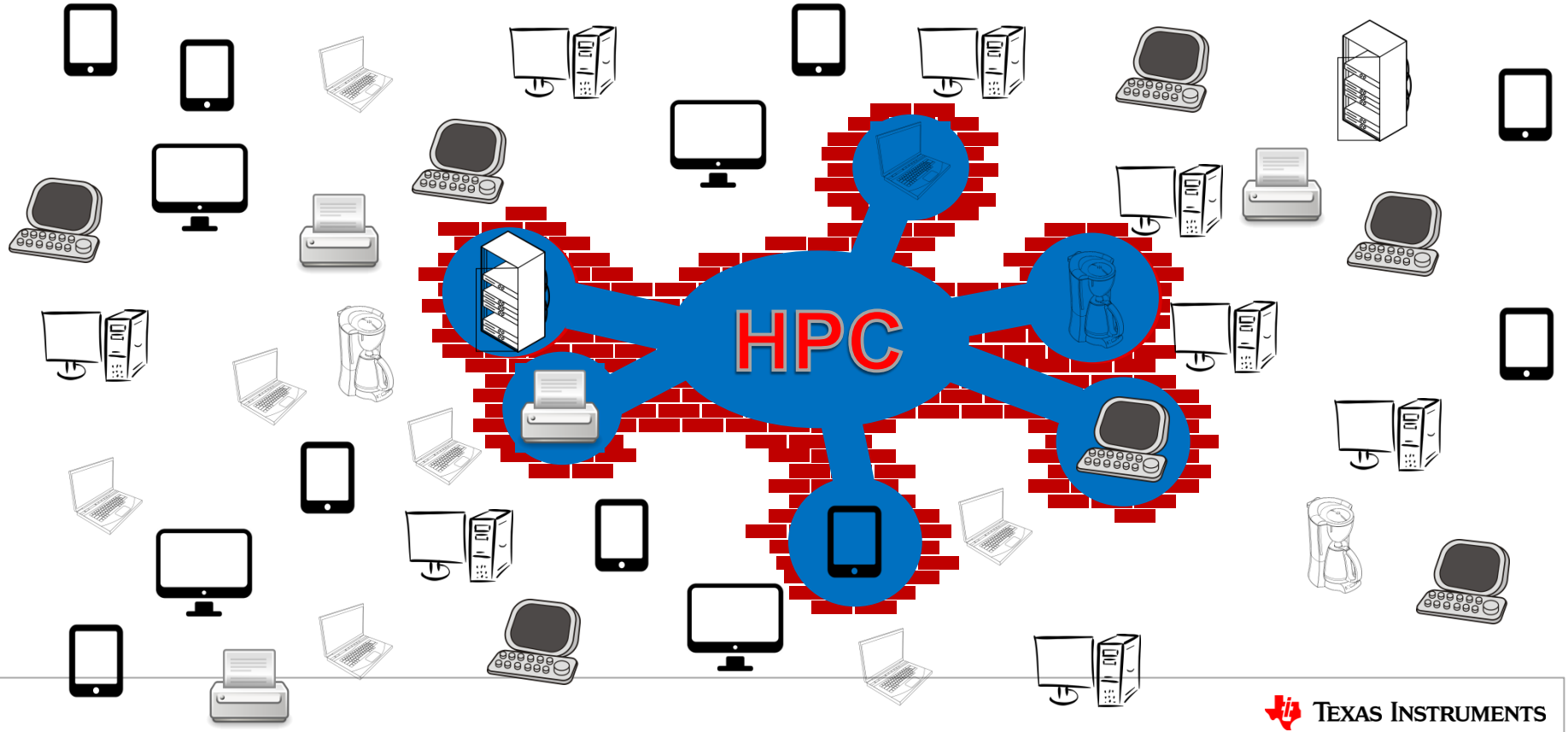
…but don’t break anything!

HPC in a big world



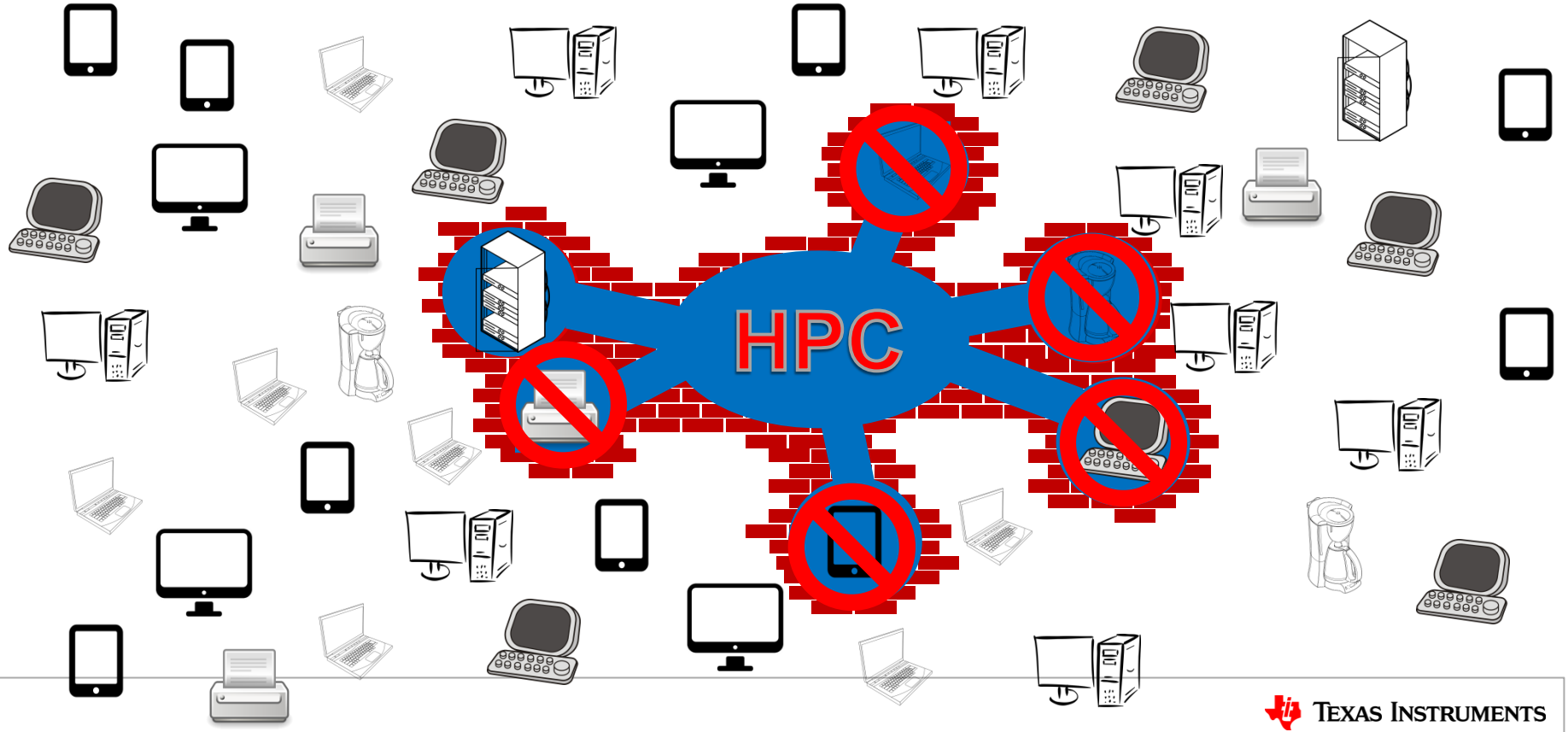
HPC in a big world

How can we consume cycles outside of HPC environments?



HPC in a big world

We can figure out how to run jobs on the coffee machine later!
Today, we can start with “Enterprise” IT resources!



Part 2
TI EDA HPC vs General HPC vs Enterprise IT

EDA HPC vs Typical HPC

Typical EDA HPC

- User environment
  - Workstation environment == Cluster environment
  - Strong “develop on the workstation, and deploy in the grid” methodology
- Data
  - All data is all on global, shared filesystems (NFS & pNFS)
- Jobs
  - Mostly embarrassingly parallel compute
- Network
  - Topology: point-to-point switched
  - Ethernet only
- Hardware
  - CPU: Top or near top bin speed
  - GPU/FPGA/etc..: Super rare
  - Hardware emulation (Palladium, ZeBu, ...): Common
- Software
  - Most software is COTS with DIY integration

Typical HPC environments

- User Environment
  - This is quite uncommon in most HPC environments
- Data
  - Data is partitioned. NFS for slow data. Parallel filesystems for fast data. Sometimes user authenticated.
- Jobs
  - Fine parallelism with complex communication
- Network
  - Topology: complex (toroid, mesh, etc…)
  - Low latency technologies abound (Infinaband, etc…)
- Hardware
  - CPU: Speed is tuned to application type
  - GPU/FPGA/etc: Super common (even dominant)
  - Hardware emulation (Palladium, ZeBu, ...): Rare
- Software
  - Quite a lot of custom developed software

These are generic observations, not universal truths!

EDA HPC vs Typical HPC

In short, the methodologies and solutions presented here will not necessary be applicable to generic HPC environments; however, they may well be applicable to a large subset of the compute run by many EDA companies.

EDA HPC environments are a lot less specialized than many other HPC environments. In fact, they look a lot like enterprise IT environments from a hardware perspective.

TI's HPC & Enterprise Environments

TI HPC Environment

- All compute servers are physical machines (no VM)
- Systems are built and typically stay the same for 3 years
- Capacity planning is about using every CPU cycle -- utilization
- Environment is logically partitioned behind one firewall
- Highly customized but uniform Linux environment
- Centralized network services (DBs, NIS, LSF, NFS, etc…)
- Compute servers have 24-32 cores & 700GB of RAM
  - Run one OS instance
  - Have high CPU utilization (>70%)

TI Enterprise Environment

- Everything is a VM (well… almost…)
- Very dynamic system orchestration
- Capacity planning is about never running out
- Hundreds of environments with various firewalls
- Each system tuned to a workload. Lots of diversity
- Thousands of environments
- Typical physical systems have 32-64 cores & 1TB of RAM
  - They may host hundreds of VMs
  - The CPU utilization is normally quite low

Protecting intellectual property

- EDA HPC clusters house some of the most sensitive corporate data (design IP)
- They frequently use a very “collaborative” IT design
- How we protect data at TI:
  - ACLs & file permissions
  - Strict control over accounts & group memberships
  - Entire network environment is segmented
  - Various other proprietary technologies (for example: data ninjas)

Part 3

The Challenges

The overall problem: Alien abduction

We wish to swoop down in the night, scoop up unsuspecting compute resources, implant them with an HPC brain, transport them to a new universe, put them to work on HPC tasks for a while, and then return them without anyone suspecting that anything ever happened.

First rule of alien abduction

Never leave a trace!

Translated to IT speak:

Never ever negatively impact an enterprise workload or the gig is up!

Second rule of alien abduction

Make it worth it!

Translated to IT speak:

The value of the additional HPC compute we bring to the table should outweigh the cost of implementation.

Some little IT problems

- How do we
  - customize the OS configuration to meet HPC requirements?
  - teleport systems inside the security perimeter?
  - attach systems to network services dynamically?
  - maintain the integrity of the security perimeter?
  - clean systems of IP before they are returned?
  - protect enterprise workloads?
  - find enterprise systems with “extra” capacity?
  - direct jobs that are appropriate to run on these systems?

Problem:

How do we customize the OS configuration to meet HPC requirements?

- We don't!
- TI's enterprise environments are virtualized and employ highly automated orchestration making it trivial to spin up new VMs with a given image.
- Images are configured at startup – hostname, IP address, security keys, etc…
- Significant security protections are placed on all VM clusters from which eHPC systems can be sourced
  - Images stored on external storage can be encrypted and signed
  - Multifactor authentication is required with rotating admin keys
  - Images are essentially immutable – if an OS config needs changed, we deploy a new template.
- OS is customized
  - Both local and remote root login is disabled
  - No root password
  - All remote logins are disabled
  - Several networking changes (see VPN slide)

Problem:

How do we teleport systems inside the security perimeter?

- We wanted to be able to pull systems into the secure perimeter without dependencies on network infrastructure.
- Our solution was to use host based VPN
  - At boot time, after Ethernet comes up but before network clients are started, we fire up a VPN which connects to the secure perimeter firewalls and places the system into a subnet dedicated to “eHPC”
  - We used an open source VPN client that was easy to automate; however, we had to modify the source code to solve some performance issues.
    - The client still introduces 4ms more latency than the vendor supplied client.
    - This latency is hard on NFS
    - We plan on using the vendor client once it has a few automation features added to it
  - We have a network based key and credential repository where the VPN keys are housed.
  - VPN connections are only allowed from a set of heavily instrumented and secured subnets.
  - The only entities allowed to deploy into these subnets are the VM clusters tagged for eHPC use.
  - eHPC server network security
    - Host based firewall blocking everything outside the VPN tunnel
    - All non-VPN interfaces are unplumbed, and devices removed
    - Kernel routing is enabled only for VPN interface
    - Some services are disabled
    - If the VPN is tickled, then the entire VM is destroyed (ex: can’t ping. Ex: VPN looses sync.)

Problem:

How do we attach systems to network services dynamically?

- For most services, this is easy
  - Show up in the correct subnet with the appropriate keys gets a system connected for most things (DNS, NIS, NFS, etc…)
  - For others connections normally happen at job request, and simply depend on keys being provided
- LSF
  - We considered using LSF's cloud capabilities
    - They were pretty difficult to configure given the complexity and diversity already in our enterprise environment
  - We considered LSF dynamic host
    - This was painful because of all the elims we would have to write to get servers integrated into the cluster.
  - We ended up simply configuring a set of generic hosts in the cluster.
    - Super simple
    - LSF recognizes when a host shows up in a few seconds
    - When hosts with the given name are not deployed, LSF considers them "down"

Problem:

How do we maintain the integrity of the security perimeter?

- A combination of what we have already seen mostly covers this
  - Strong eHPC host network security
  - Strong VM infrastructure security
  - Tons of encryption
  - Fast rotation of keys
  - Serious network monitoring
- Penetration testing helps us feel better about the whole thing. ;)

Problem:

How do we clean systems of IP before they are returned?

- eHPC systems life span limits local “IP buildup”
  - Limited life span measured in hours
  - Immediately die if the automation detects any edge conditions.
- VM are reaped completely
  - Storage is wiped & all state is removed

Problem:

How do we protect enterprise workloads?

- **Layer 1:** Measure load and only deploy where we think we have free cycles
- **Layer 2:** LSF measures VM load, and keeps them from getting overloaded
- **Layer 3:** The VM software locks CPU & RAM allocation down
- **Layer 4:** If enterprise workloads receive priority access to CPU & RAM if the physical hardware is under contention. (hypervisor feature)
- **Layer 5:** eHPC machines are part of a LSF cluster managed by the enterprise team
  - If everything else fails, then the enterprise team can manually adjust load

Problem:

How do we find enterprise systems with “extra” capacity?

- The system requires a scheduler running outside of LSF
  - Scans eHPC systems, and reaps them if required
  - Reaps eHPC systems if they are too old
  - Fires up new eHPC systems as necessary
- The obvious thing to do is have this scheduler check VM cluster loads, and add/remove eHPC systems as required.
- We are not doing the obvious yet!
  - We also have some cultural issues to work out with regard to turning on the full automation.
  - That said… We have found so much capacity we can manually identify, we have not felt the need to automate it!
  - Once we pick all the low hanging fruit, and get more comfortable with eHPC, we will turn on the rest of the automation.

Problem:

How do we direct jobs that are appropriate to run on these systems?

- We use multicluster to move jobs between clusters and the eHPC cluster
  - All jobs that land in a “sending” queue will be sent to the eHPC cluster
  - Users don’t generally put jobs in these clusters
- “Smart Queue”
  - Is a tool that scans all queues for jobs that look like they are a good fit, and puts them in the send queue
  - It also can move jobs out of the send queue if they set for too long
  - How do we identify “good jobs”? That’s a secret!

Part 4

What about the public cloud?

Can we extend this methodology to the cloud?

- Sure… but…
  - It can. We have even tested it with AWS
  - Not all components of the solution are required
    - For example: VPC makes the host VPN unnecessary
  - Some components should change
    - For example: Where one stores VPN keys
- In short:
  - This methodology builds a strong foundation for being able to bring in small numbers of compute servers quickly, but should be modified for efficiency.

Part 5

An Even *BIGGER* Idea!

Why can’t we all share?

- What eHPC has taught us is that “HPC” the way we do it in EDA isn’t so exotic.
- Why can’t we have one large, shared VM infrastructure for everyone?
  - Enterprise! HPC! Manufacturing! LAN parties!

What I’m talking about is a single, internal cloud for the entire corporation!

Discussion Time