Containers, HPC, and Cloud: A personal interpretation of "Reinventing HPC" (this year's theme in ISC'24)

- Manolis Marazakis (maraz@ics.forth.gr)

FORTH-ICS Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)

What is this presentation about ?

- A personal interpretation of "Reinventing HPC"
  - This year's theme in ISC'24
- Key question: How to run hybrid ”HPC + Cloud” workloads ?
- Highlights of recent and ongoing work on system software enabling HPC and Cloud to execute on shared infrastructure and in a mix-and-match manner
  - Will start with peer-reviewed material, but will progress to formulate 3 “Position Statements”
  - Extrapolation from prior work + personal views
    - Standard disclaimer applies: YMMV …
- 3Cs framework: Containers, Coordination, Culture

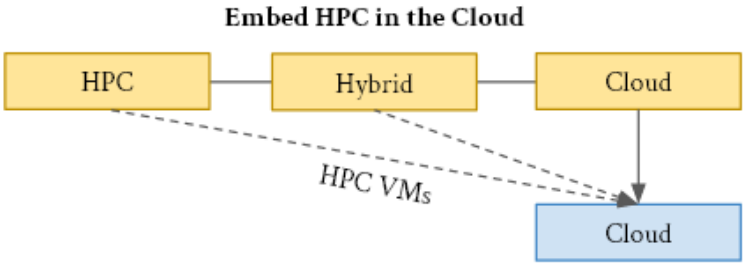
Technical Context & Position

- Convergence of HPC and Data Analytics … and AI/ML
  - At the infrastructure level (HW, SW)
- From the viewpoint of HPC Centres
- Containerized workflows with diverse steps
- Co-existence of Cloud and HPC resource managers (RM)
  - One RM ? Why not two or more ?
    - Starting in Kubernetes run-time, call-out to Slurm-managed HPC cluster
    - Or: Deploy transient Kubernetes run-time via Slurm

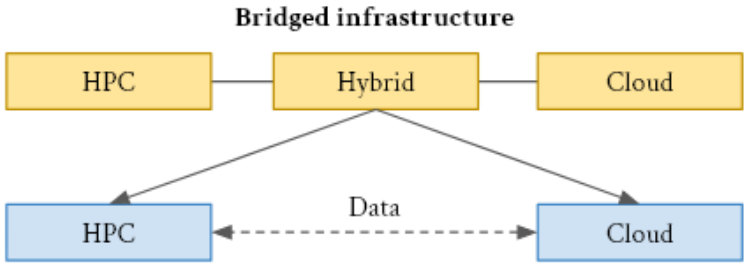
Viewpoint shift: from "Modular Software Stack" to "Factory of Multiple Software Stacks"

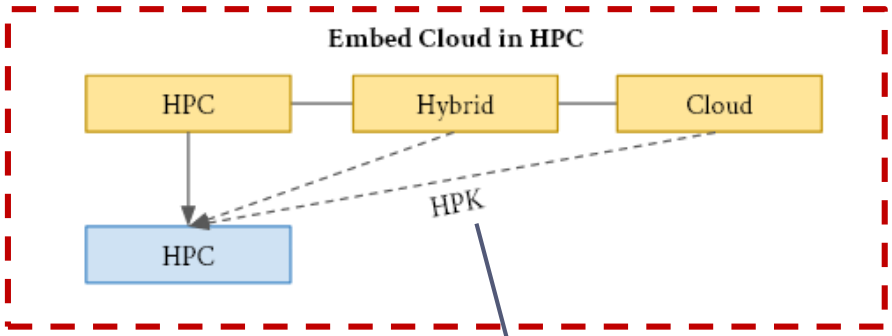
How to run hybrid ”HPC + Cloud” workloads ?

Viewpoint of Cloud Provider



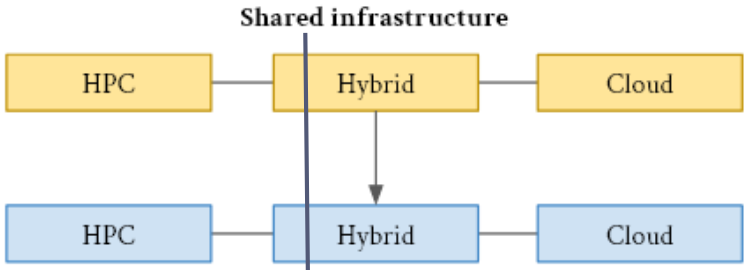
Viewpoint of user of Cloud Services





Viewpoint of HPC Centre

Proof-of-Concept for Factory pattern

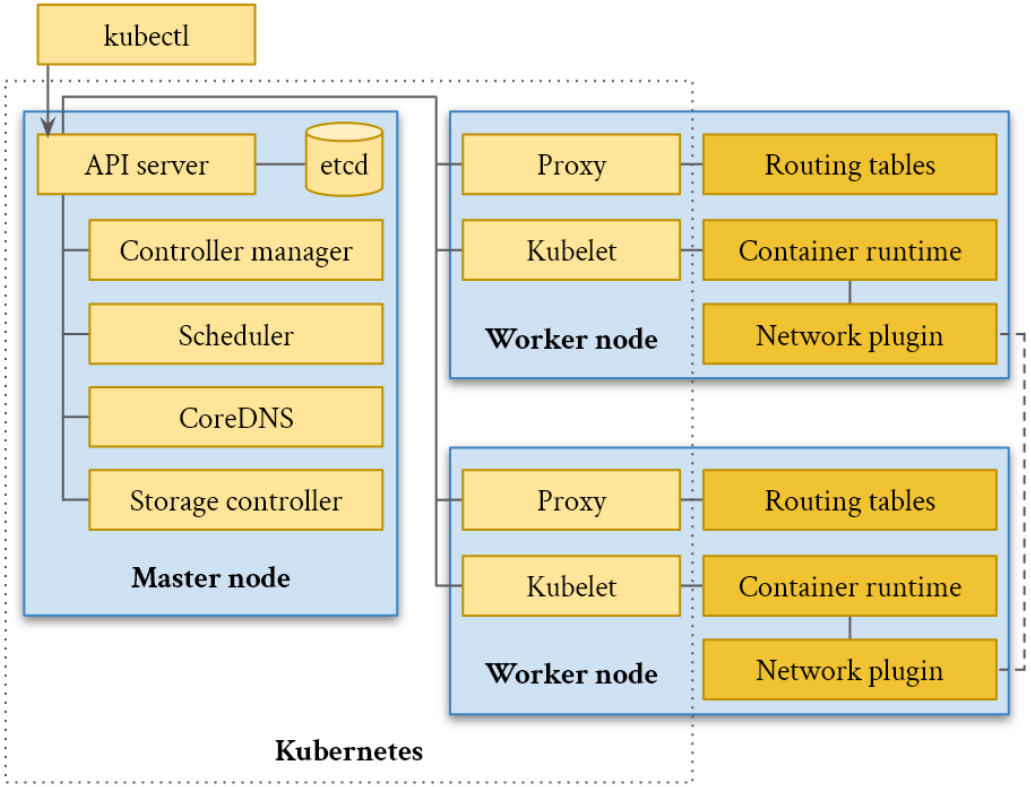


Viewpoint of Federated HPC/Cloud Centres

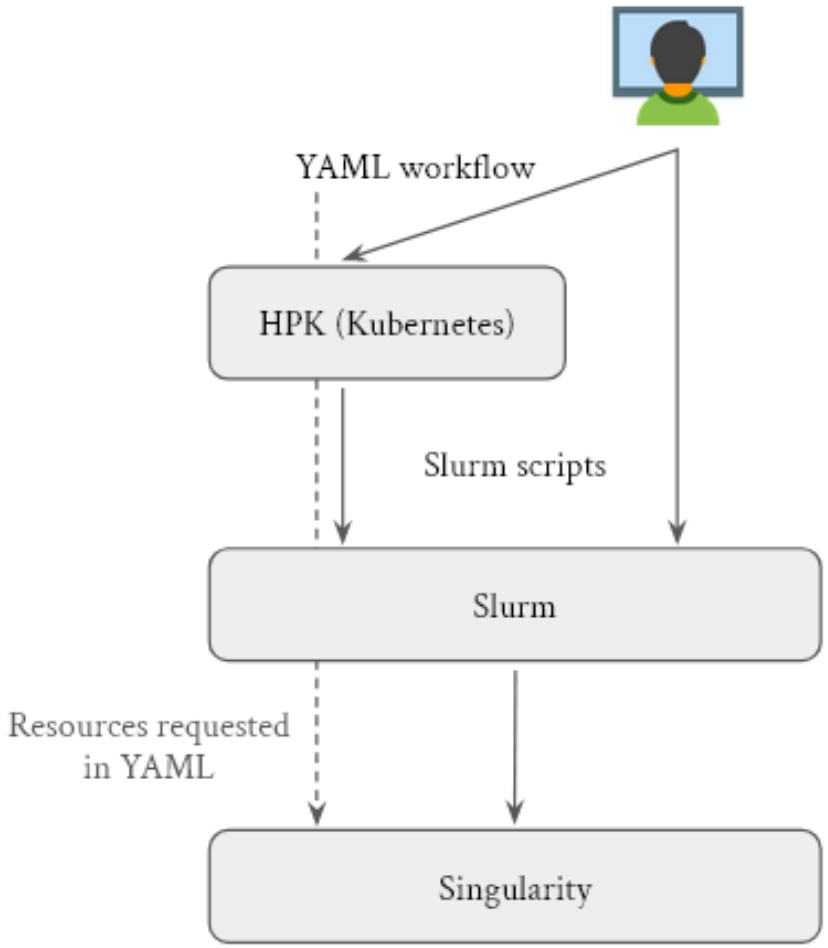
Not there yet: Many challenges (tech + org)

Kubernetes

- Declarative vs imperative
- API endpoint & controllers
- Abstractions
  - Pods → Collection of containers
  - Deployments → Replicated pod groups
  - Services → Naming of Microservices
  - Jobs → Pods that run to completion
  - Volumes → Mountable file collections
  - Labels → Queries on metadata
- DevOps viewpoint
  - Infrastructure as code
  - Version rollouts, CI/CD workflows
- Distributed structure
  - "Control plane" → Scheduling and placement
  - Node agents → Handle execution
  - Monitoring and accounting infrastructure



HPK: “High Performance Kubernetes”



- Custom kubelet for the execution of containers
  - Changes in various other components to enable the integration
  - Runs as user process
    - via Slurm
    - Translates Kubernetes to Slurm scripts
  - Technical overview presented in WOCC’23
  - https://github.com/CARV-ICS-FORTH/HPK

The Factory Design Pattern

- Source: *Design Patterns*: Elements of Reusable Object-Oriented *Software* (Addison-Wesley, 1994)
  - Authors: "Gang of Four" (GoF) - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.
  - How to solve recurring design problems to design flexible and reusable object-oriented software
  - Originally, 23 patterns: Creational, Structural, Behavioral

Factory: "Define an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses." → decoupling creation process from the rest of the code.

Position statement #1: Apply the Factory pattern to instantiate an entire software stack with limited lifetime.
The HPK work shows that it can be done, in existing HPC run-time environments.

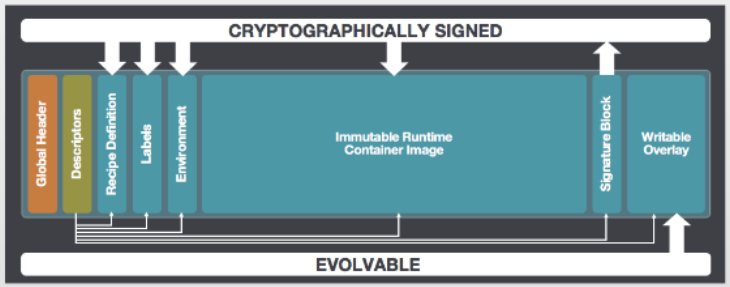
Infrastructure for “Factory of SW stacks” ?

- From the viewpoint of a HPC Centre:
  - Identity & Access Management (IAM) already in place
  - Mostly batch-style time-based usage of (expensive) execution platforms
  - Containers already supported (Singularity, Apptainer)
    - Caveat #1: Not very flexible networking, but we can have some software-defined infrastructure in place (eg. Flannel)
    - Caveat #2: Complications in use of (expensive) hardware accelerators
- … but can we trust containers ?
  - Provenance of containers ? (specifically, binary image files)
    - Complementary concern to data provenance (lineage)

Provenance of Containers

- Verifiable origin
- Integrity of container images
- Trustworthy deployment and execution
- One approach: Trusted container images in HPC, through Singularity and Chain of Trust

- **Immutable Section (for Sharing)**
  - Includes everything the application needs to run: libraries, utilities, settings, ..
  - Provides metadata about the applications.
  - Backward compatible with Docker Images.
- **Mutable Section (for Runtime)**
  - Overlay filesystem for writing temporary data within the container.



Singularity Image Format (SIF)

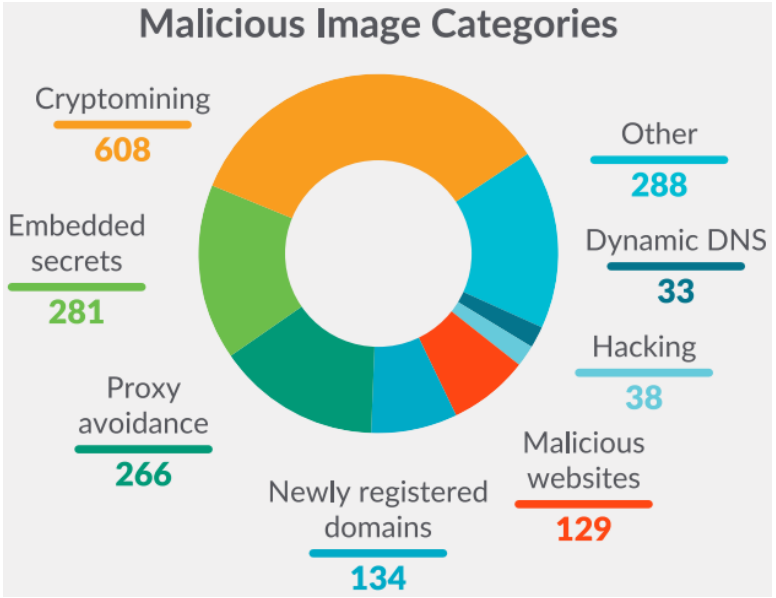
Signature block ensures image integrity.

But what about the source of origin ?

Malicious Container Images

•Most of Singularity images come from **Docker Hub**.

•Docker Hub reportedly contains more than 1600 images that hide malicious behaviors.



•Threat model in **isolated Docker environment** is one thing.
- **Resource Abuse:** cryptocurrency mining, …
- **DOS attacks:** bots for attack on others, attack on the network/storage.

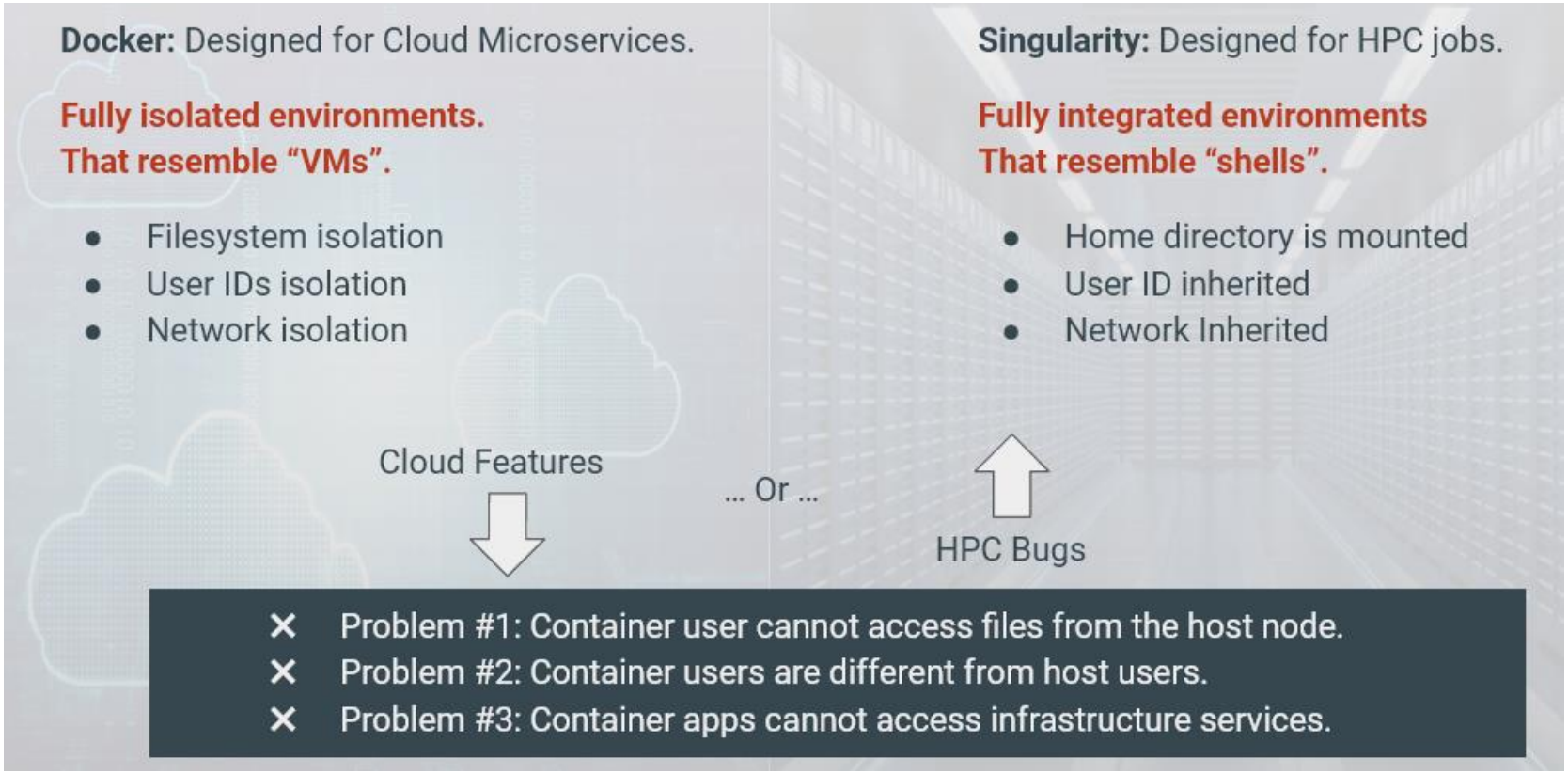
•Threat model on **integrated Singularity environment**, is another.
- All the above, plus
- **Information leaking:** secrets, sensitives files, …
- **Trolling:** removal of persistent files, …
- **Persistent Malware:** e.g., if placed on a shared directory …

A worsening problem

Sysdig says that in 2022, 61% of all images pulled from Docker Hub come from public repositories, a 15% rise from 2021 stats, so the risk for users is on the rise.

Unfortunately, the size of the Docker Hub public library does not allow its operators to scrutinize all uploads daily; hence many malicious images go unreported.

Containers in Cloud vs. Containers in HPC: Different prioritization of features



Trust Options for Container Images ?

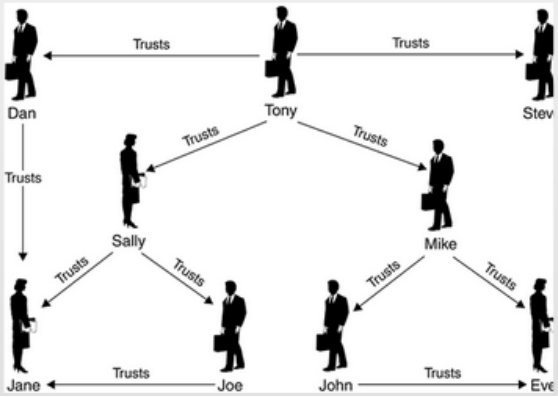
- **Network-circulated Images**
  - Circulate the image between friends, and **hope for the best**.
- **Trusted Private Registry**
  - Image containers are **implicitly trusted by existing on a trusted registry**.
- **Trust-enabling Images**
  - Image containers provide the means to be **explicitly validated, even if circulated via untrusted means** (e.g, Public Registries).

Secularity: Our approach (available in upstream Singularity and Apptainer)

Trust Management Models for Container Images

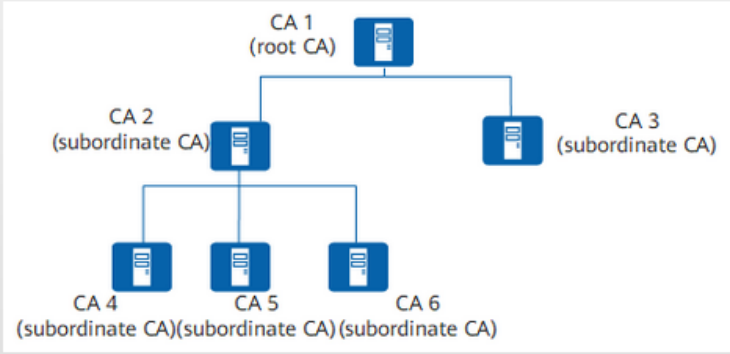
Web of Trust (PGP)

- Decentralized architecture where trust is established by individuals vouching for each other's identity.



Chain of Trust (X509)

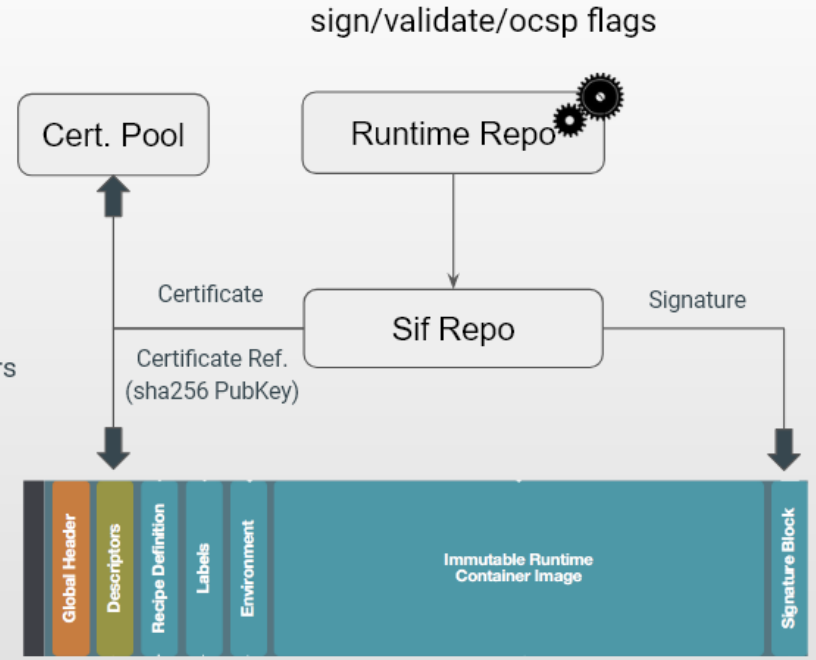
- Centralized architecture where trust is established by chaining a certificate back to a trusted root CA.

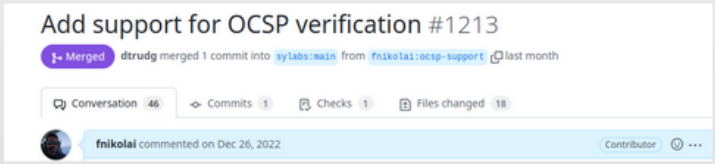


Position statement #2: HPC Centres need to interface with, or even offer, the services of a Certification Authority.

Chain-of-Trust: first merged in Singularity v.3.11

- Digital Signing handled by Singularity maintainers.
  - Key management (RSA, ECDSA, ED25519)
  - Image signing / validation
  - Offline certificate chain validation
- Certificate Revocation handled by FORTH.
  - Integrated client-side OCSP protocol.
  - End-to-end testing
    - Integrated the OCSP responder
    - Tested against external OCSP Responders





https://github.com/sylabs/singularity/releases/tag/v3.11.0-rc.1

(feature was later integrated in Apptainer as well)

Resource management (RM), as per **Flux**

- Evolve scheduler's view beyond the single dimension of "nodes"
  - Framework that enables new resource types, schedulers, and framework services to be deployed
- In Flux, each job is a complete instance of the framework, meaning the individual task can support parallel tools, monitoring, and even launch sub-jobs that are, like fractals, smaller images of the parent job.
  - Because each job is a full Flux instance, users can customize Flux for use within their jobs.
- "Users benefit from pluggable schedulers with deeper knowledge of network, I/O, and power interconnections, and the ability to dynamically shape running work."

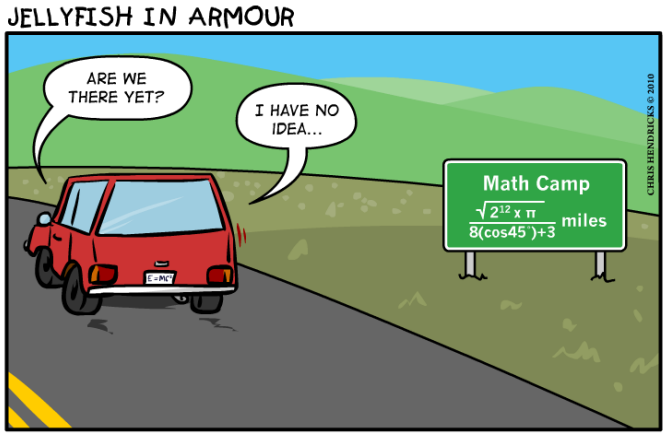
[source: https://computing.llnl.gov/projects/flux-building-framework-resource-management]

Resource management for convergence ?

- As with Flux: multi-level, hierarchical, pluggable
- … but with co-existence of resource managers
  - Eg. Slurm to deploy “transient” Kubernetes environment for each user … which then
  - It can be done … but it is cumbersome: SW components with many assumptions about their role in the actual run-time environment
    - Complexity beyond “simple” version dependencies between modules
    - … these can be handled via EasyBuild, Spack
    - … together with managed runtimes (eg. Conda/Anaconda)
  - Our own proof-of-concept with HPK case studies: Spark TPC-D, PyTorch training flow
- … high-value high-maintenance
  - How to incorporate a “Things will break” clause in a SLA ?

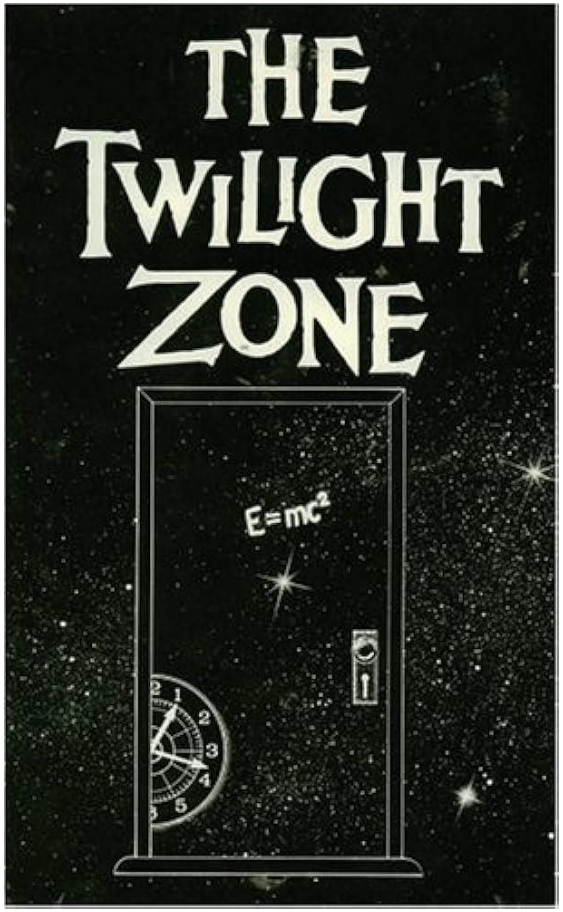
Shift to Coordination, with RM as a component

- Evolution towards coordination-based systems
- Paradigm that encompasses resource allocation & scheduling, communication and cooperation between “processes”
  - … but with a very broad interpretation of “processes”
  - Entire SW stacks, either permanently or transiently deployed
    - Owned/monitored/managed by infrastructure owner(s)
    - … or by users
    - … in single or federated system setting



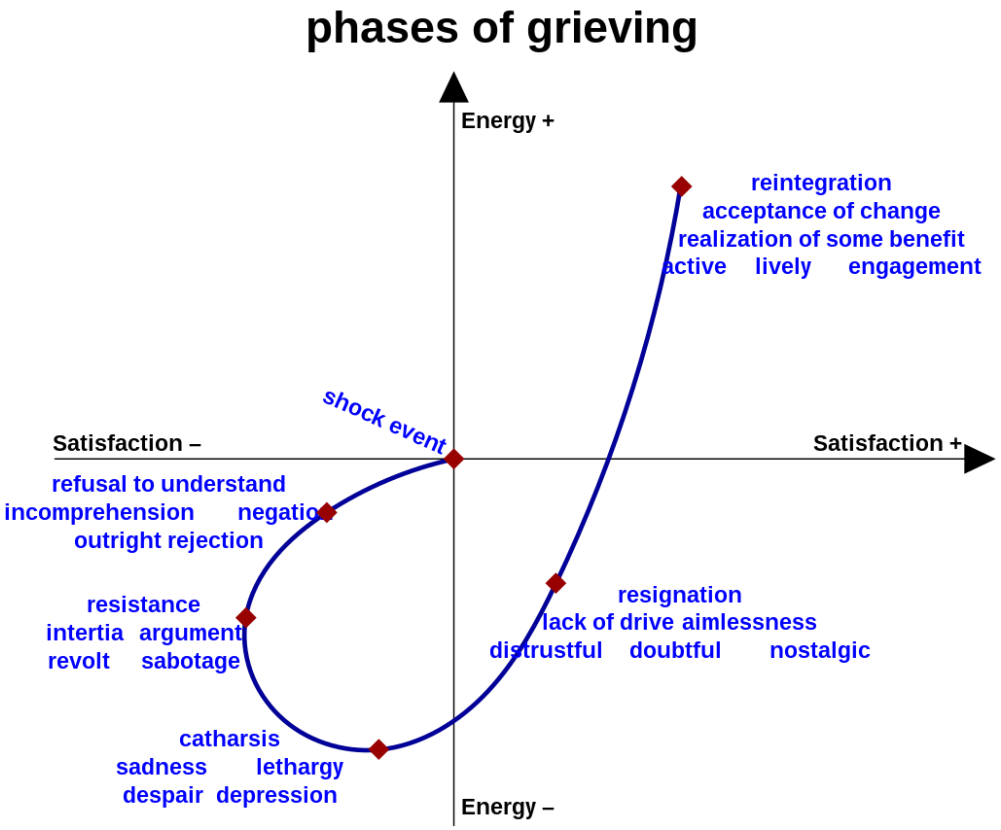
Proposed Framework (actually: frame of mind), following a “3Cs” template

- “You're traveling through another dimension, a dimension not only of sight and sound but of mind, a journey into a wondrous land whose boundaries are that of imagination. That's the signpost up ahead—your next stop, the **Twilight Zone**.”



5 Phases of Grief

Kathy Yelick's keynote @ ISC'24: "Beyond Exascale Computing"
5 stages of AI: denial, anger, bargaining, depression, acceptance
… akin to 5 stages of grief (Kubler-Rossmodel, 1969)



Grief management: **3Cs**
Choose, Connect, Communicate
"It's not about getting over it.
It's about learning to live with it."

Diagram source: https://en.wikipedia.org/wiki/Five_stages_of_grief#/media/File:K%C3%BCbler_Ross_grieving_curve_(edited).svg

3Cs - Containers, Coordination, Culture

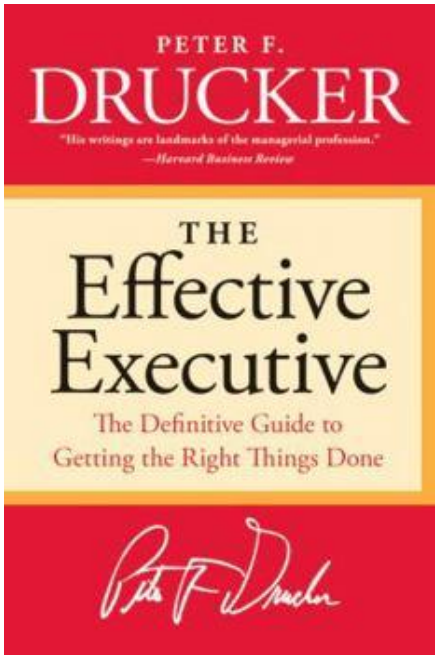
- **Containers**: Make them trustworthy, use them in “Factory of SW stacks” pattern.
- **Coordination**: Resource management in HPC/Cloud needs to evolve beyond the premise of a single owner/controller/tracker of resources.
- **Culture … of brokenness**: No more well-tuned, well-maintained SW environment → Accept the value of short-lived, highly-customized, more interactive, much more fragile SW stacks.

Capability vs. Suitability

- Gary Bernhardt @ Mountain West Rails, 2012
  - https://www.youtube.com/watch?v=NftT6HWFgq0&t=1s
- Capability: Ability to “do new things”
- Suitability: Ability to “do things well”
- “Culture of brokenness”
  - Originally stated in the context of Ruby/Rails SW development
  - Chaotic adoption and abandonment of SW components … because people are excited

Without effectiveness there is no performance

- Peter Drucker, “The Effective Executive” (1966)
  - Efficiency: doing things right
  - Effectiveness: doing the right things
- **Can we have both ?**
  - Well, we should



Position statement #3: Containers, Coordination, Culture are the key elements for HPC+Cloud/AI convergence. Prioritize Capability. Suitability will then follow, driven by ‘peer pressure’ and ‘economic necessity’.

Cyclic process to be embraced and explicitly managed:

Abundant compute/storage/network resources → Expansion of **Capability**

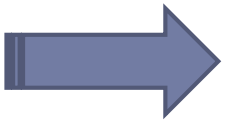
→ **Confusion** → Remedial reaction by **Suitability**-minded contraction.

An interpretation of this year's theme for ISC

- [ISC'24] "Reinventing HPC", with particular emphasis on integrating AI-driven applications and quantum computing
- [My take] Broader scope of high-performance computing
  - Need to evolve the systems software infrastructure

Perspective on engineering HPC systems with more versatility in anticipation of increasingly diverse workloads and complex execution platforms in HPC centers.

installing and maintaining single well-tuned software stack



capacity to build and maintain multiple* software stacks

*: *to serve specialized computational and data processing workloads that encompass classic HPC together with Cloud and AI.*

"3Cs" Framework to think about such matters:
Containers, Coordination, Culture (of brokenness)

Acknowledgements

- Collaborators:
  - Antony Chazapis, Evangelos Maliaroudakis, Nick Kossifidis, Angelos Bilas (FORTH)
  - Fotis Nikolaidis (now at: SuperDuperDB)
  - Ioannis Sfakianakis (now at: ZEDEDA)
- ETP4HPC (Strategic Research Agenda WGs)
  - https://www.etp4hpc.eu/sra.html
- Projects:
  - RISER: RISC-V for Cloud Services
    - HaDEA, Horizon Europe, GA 101092993 / 2023 - <ongoing>
    - https://www.riser-project.eu/
  - DEEP-SEA: Developing a programming environment for European Exascale systems
    - EuroHPC JU + Greek Secretariat for Research and Innovation, GA 955606, 2021-2024 - <completed>
    - https://deep-projects.eu/

Thank you for your time and attention!

Contact:

Manolis Marazakis: maraz@ics.forth.gr

https://www.riser-project.eu/

https://deep-projects.eu/