

Operating System challenges in the petascale era and beyond

Marc Shapiro
Directeur de recherche
Équipe Regal
INRIA Paris-Rocquencourt & LIP6

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE



centre de recherche **PARIS - ROCQUENCOURT**



Outline

General-purpose computing in the petascale era and beyond

State of the art

Operating system challenges

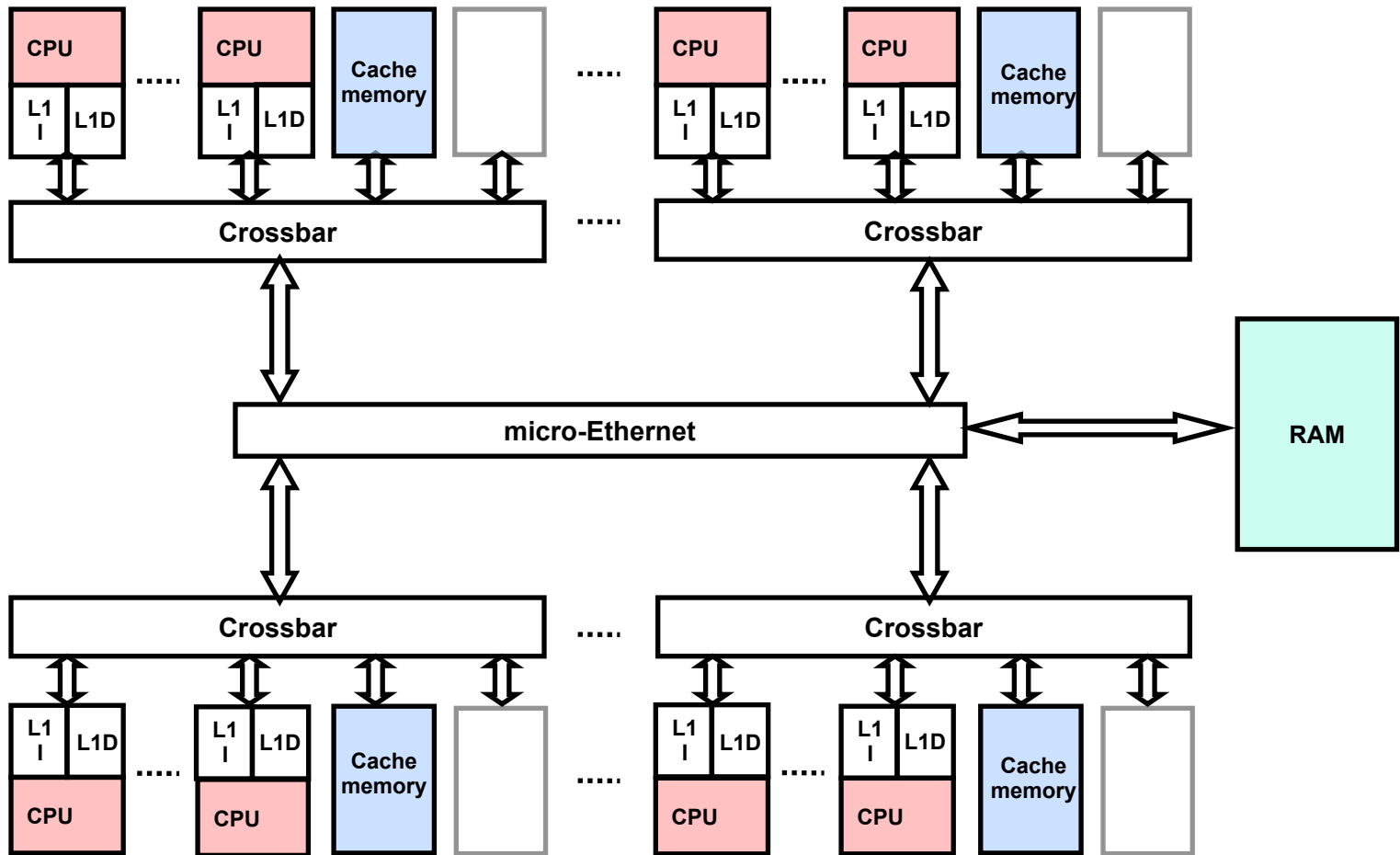
Summary

Sometime in near future...

Tomorrow:

- 10^3 CPUs on desktop $\times$ 10^6 across enterprise
- Heterogeneous: mobile phone to data centre
- Fat pipes ↗, core connectivity ↗
- QoS variability ↗, latency →
- Asynchronous, failure-prone $\Rightarrow$ FLP

Tsar multi-core interconnect



Source: Alain Greiner, LIP6

Users need cycles & storage

New interaction models

- Cameras, microphones
- Surfaces, tablets, sensors

Personal applications

- Personal search
- Constraint solvers
- What-if simulation
- Data mining

Community applications

- Wikis, collaborative editing
- Social networks, games



Grid, Cloud Computing

Data centres, clusters

$< 10^6$ nodes, stable, identified, trusted

Move data to centre

✓ Economies of scale, problems outsourced

✓ Control

- Expensive

- End-to-end latency

- Scalability, fault-tolerance $\Rightarrow$ replicate to edge $\Rightarrow$
consistency, edge issues

Google = 300 TeraFLOPS

Edge, P2P Computing

Large-scale, decentralised, self-organising

> 10^6 , churn, anonymous, untrusted

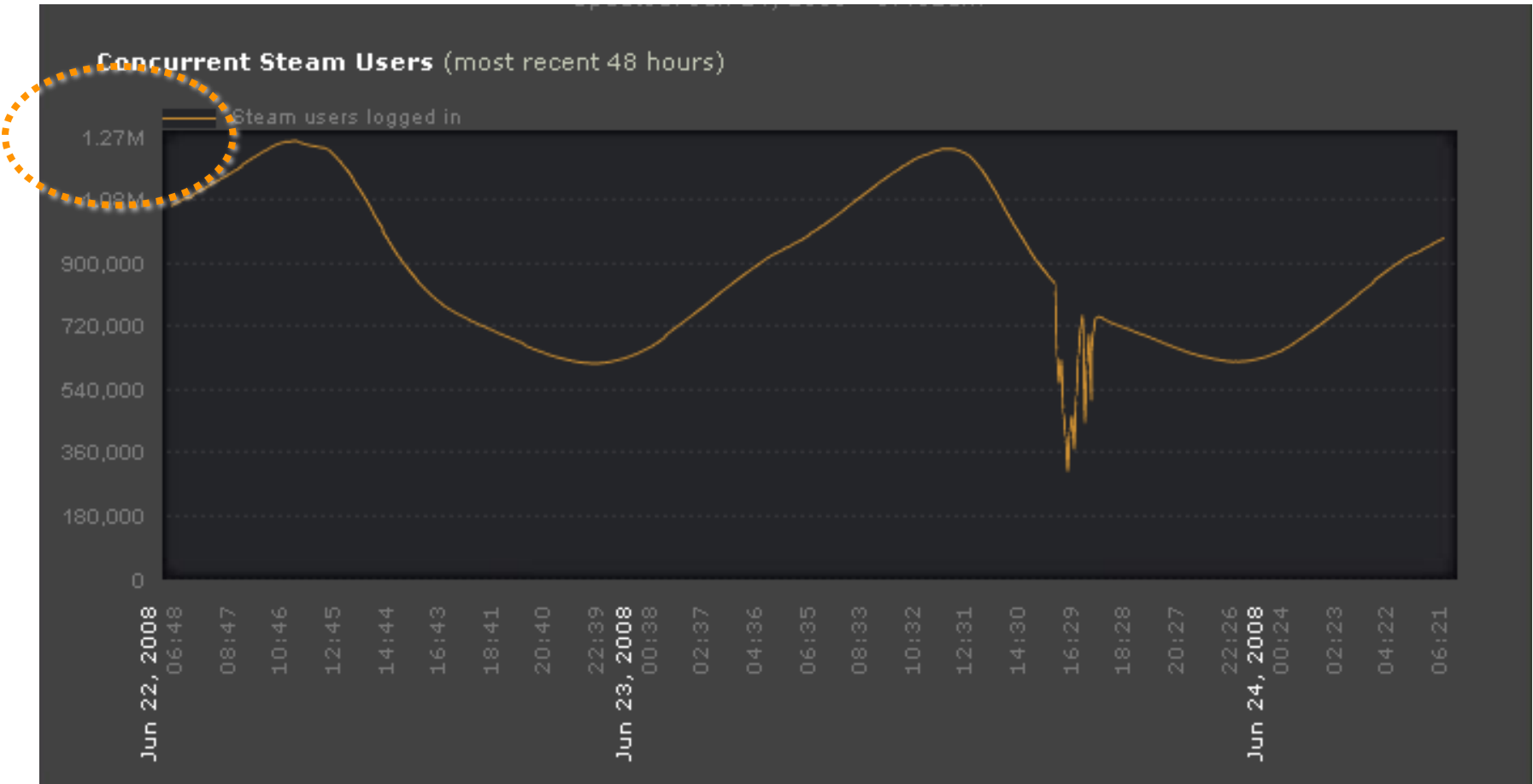
Data, computation near source, use point

- ✓ Low marginal cost, management cost
- ✓ Low latency
- ✓ Fault independence, replication, scalability
- Uncontrollable
- Approximate results
- Scalability, fault-tolerance $\Rightarrow$ replicate $\Rightarrow$ consistency issues

Folding@home = 3.3 PetaFLOPS

Early adopters

First-person shooter



Game architectures

FPS, MMOG-RPG, Virtual Worlds : Multi-server

- Game world partitioned in disjoint zones (kingdom, island) / server
- No interaction between players across servers
- Can change zones: complex communication

Limitations

- Scalability with number of players
- Latency and bandwidth on Internet

Working on server-less, P2P architecture

- Players exchange information directly

Petascale for the masses

Massive parallelism + lots of resources

No more just for the elite

Decentralise to the edge

Issues:

- Support for large-scale personal & community applications
- Mask complexity
 - Transparency, programmability
 - Autonomic control, graceful adaptation
- Control: virtual domains
 - Security, cost boundaries
 - Negotiate cost vs. guarantees

Outline

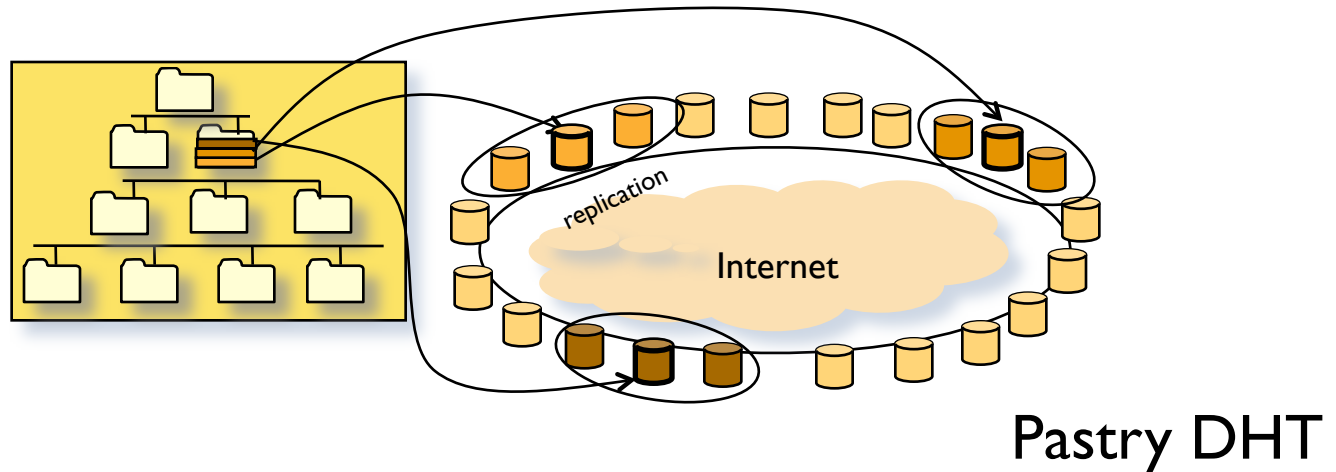
General-purpose computing in the petascale era and beyond

State of the art

Operating system challenges

Summary

Pastis P2P file system

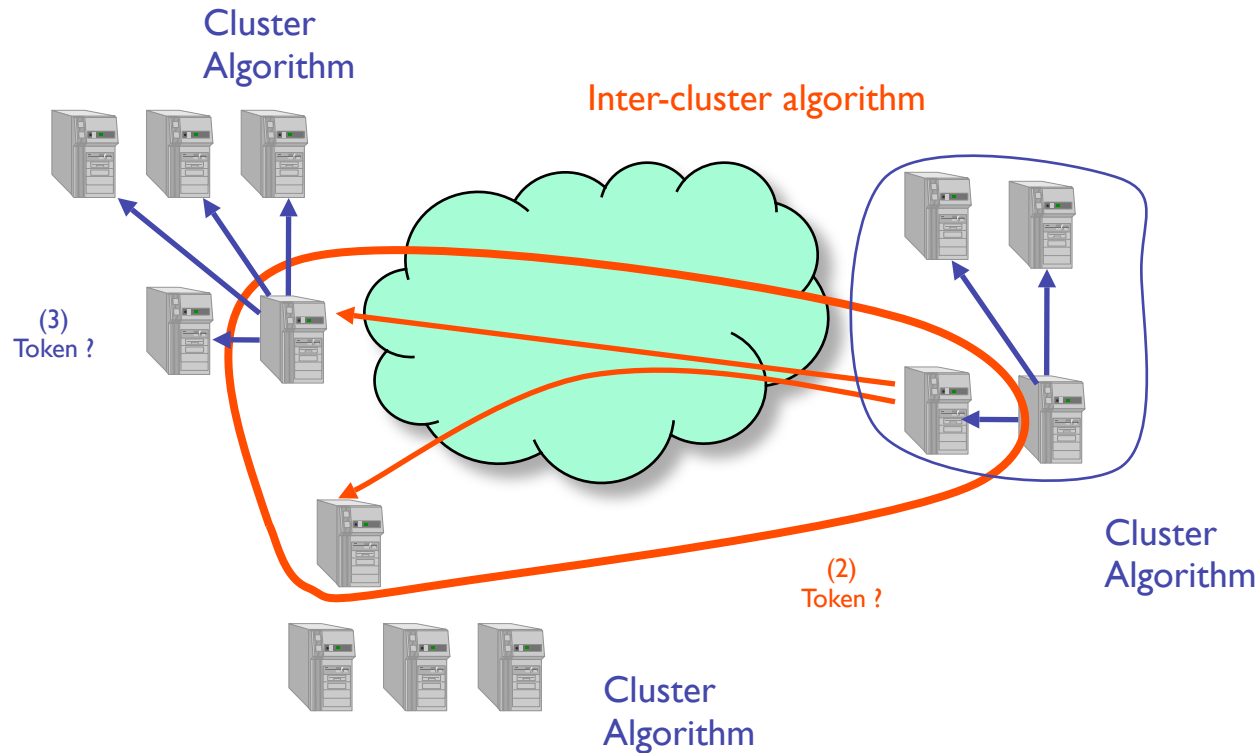


DHT: data storage & replication, fault independence

Store file system blocks in DHT

- Self-organising, adaptive
- High availability, serverless file system
- Used in Internet set-top boxes

Scaling OS primitives to grids



Hierarchical mutual exclusion
Hierarchical failure detector

Grid4All (g4a): democratic grid

Grid for schools, university educators, families,
NGOs.

Usability, decrease complexity

Volatile, harsh environment

- Minimise manual administration costs
- Self-healing, -configuring, -tuning, etc.

Dynamic VOs, available, scalable, decentralised

Collaborative data-centric application scenarios

g4a Virtual Organisations

VO =

- Virtualised set of resources + names
- Virtualised group of users (members)

Management: create, delete VOs, maintain membership, control resources, etc.

VO-aware file system

- Private workspace per VO
- Expose files to workspace: remote link
- Fault tolerant, secure access to other users' files
- P2P, localised, serverless, network transparent
- Read and write: best-effort consistency

School activity: Volcano simulation

Digital simulation of volcano eruption

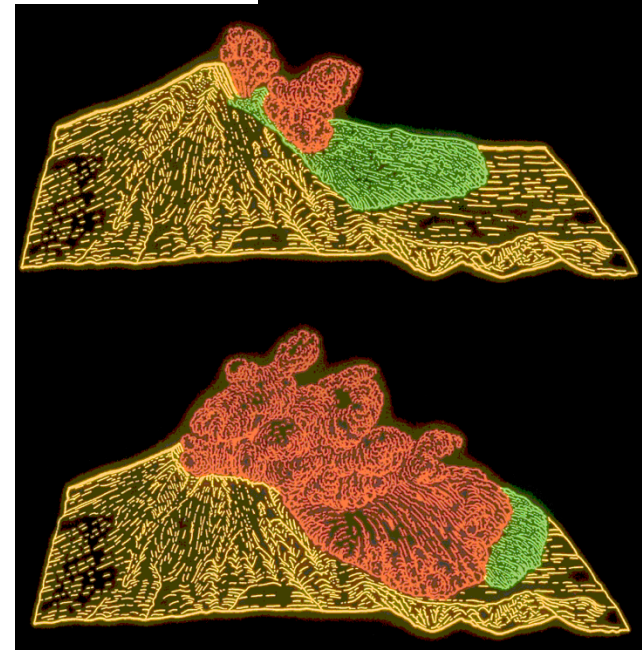
- What-if: fluidity, water table, etc.
- Explain Mt St Helens explosion
- CPU, data intensive
- Many experiments: provenance

Collaborative work

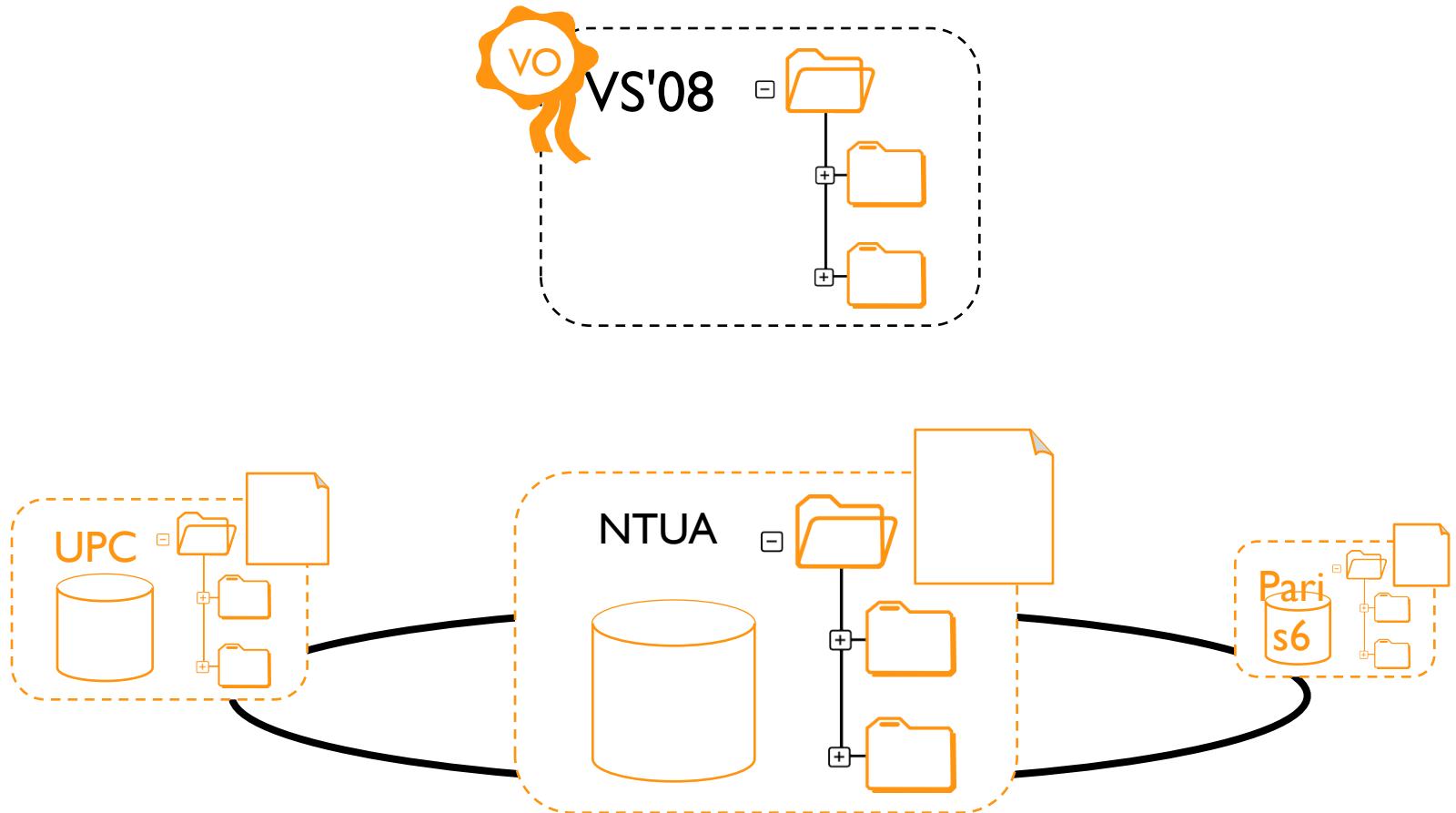
- In class, between schools
- Also: edit report, etc.

Adapt to available resources:

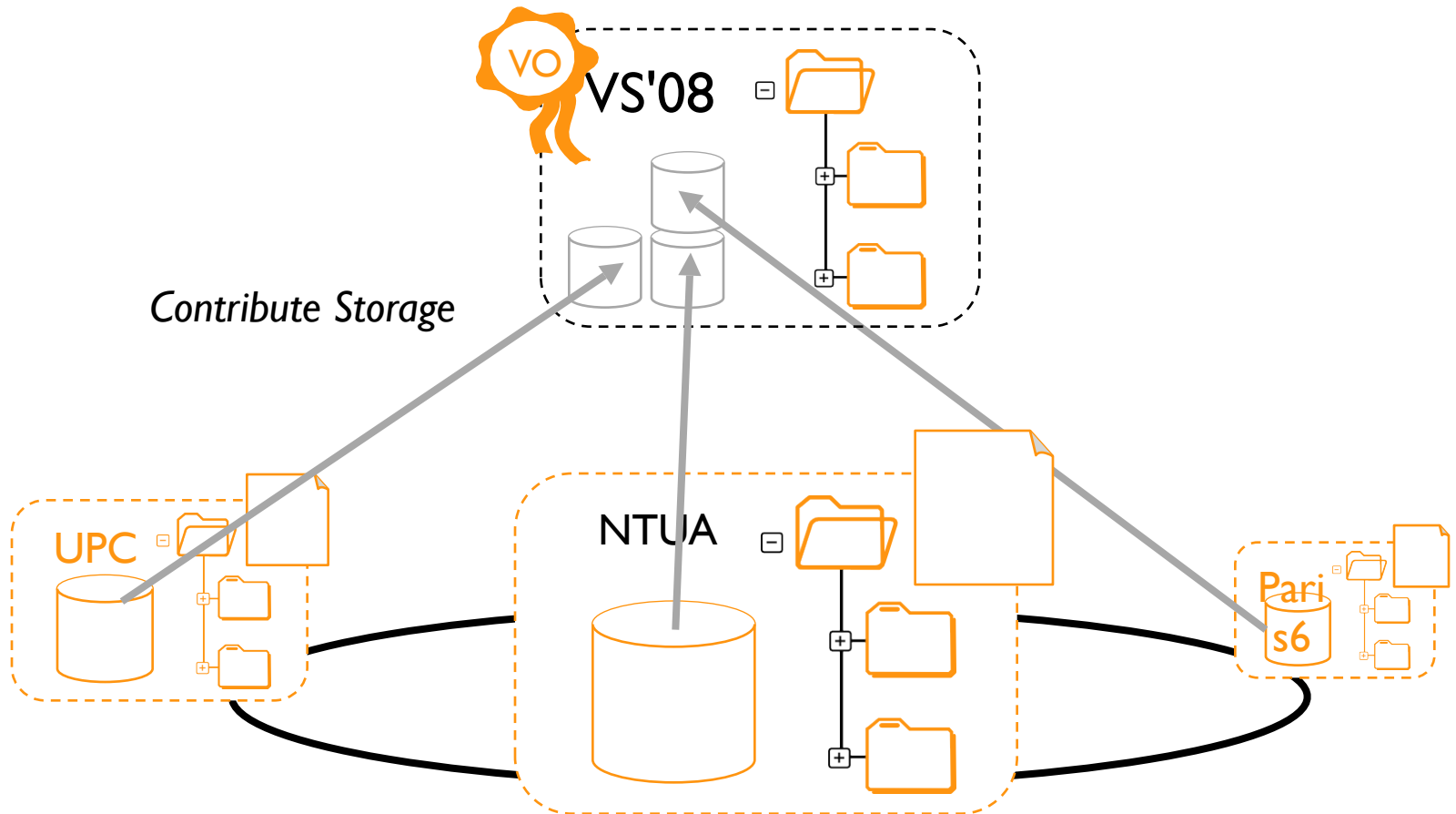
- Owned by schools
- Lent by families
- Bought from Cloud



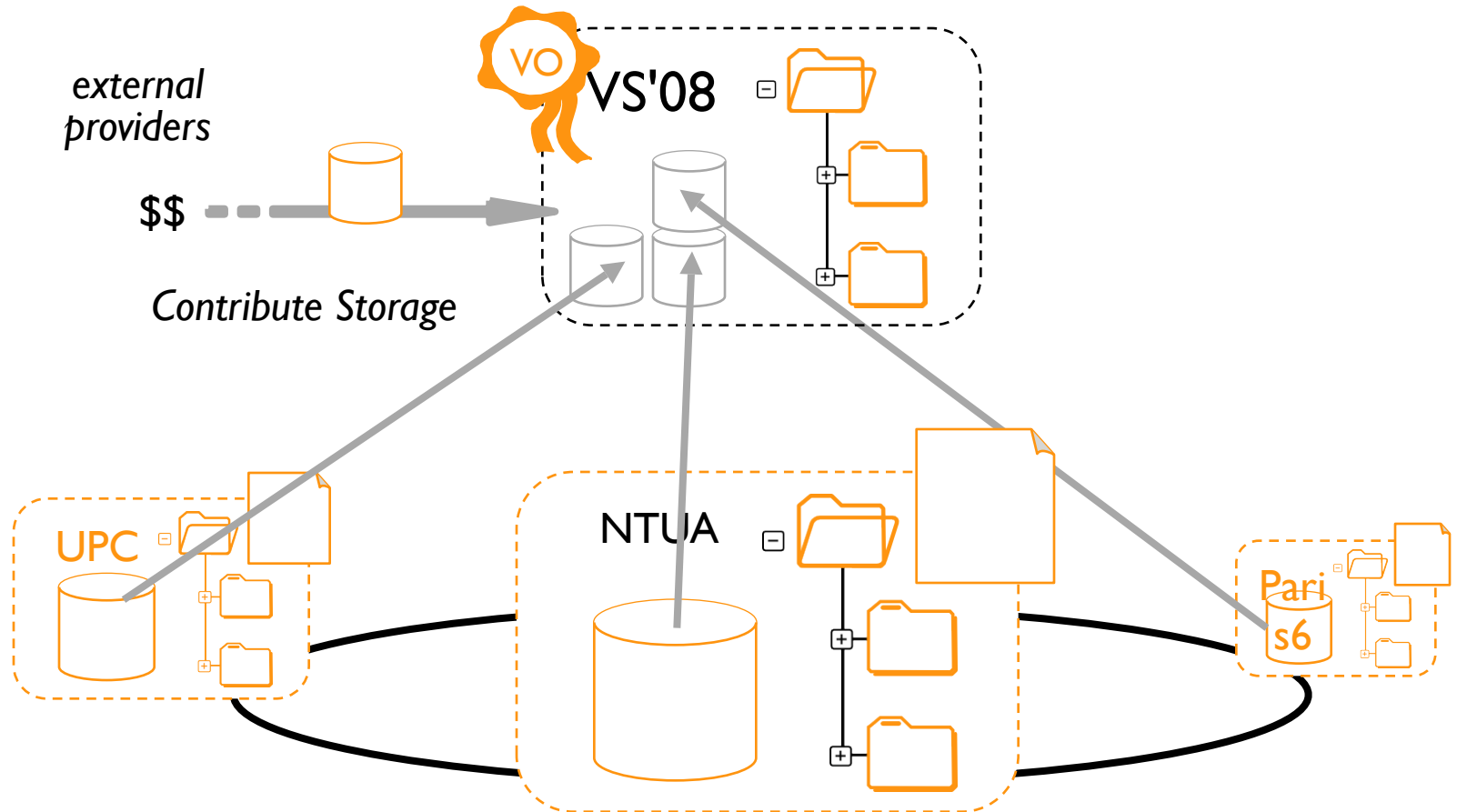
Volcano Simulation workspace



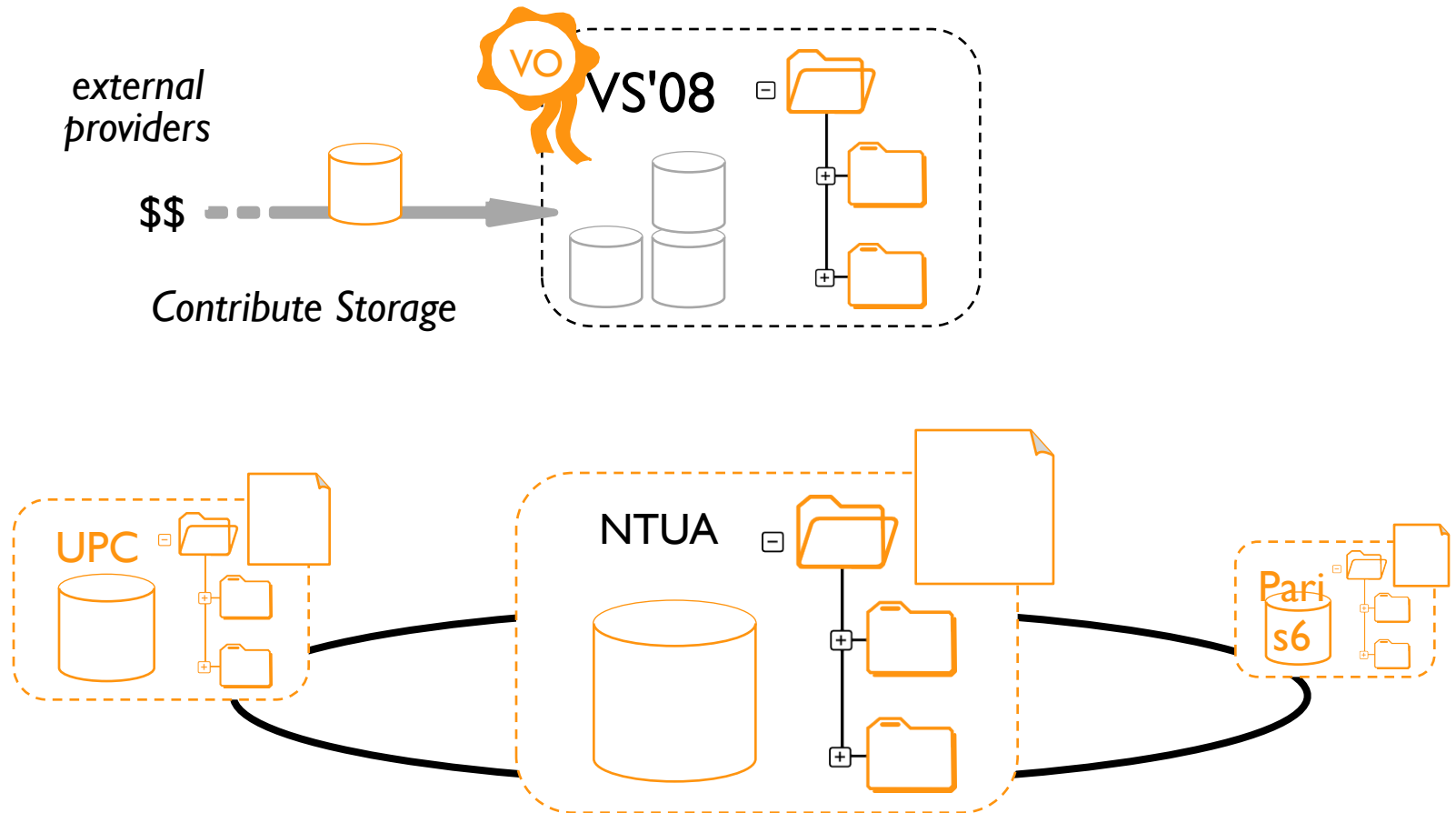
Volcano Simulation workspace



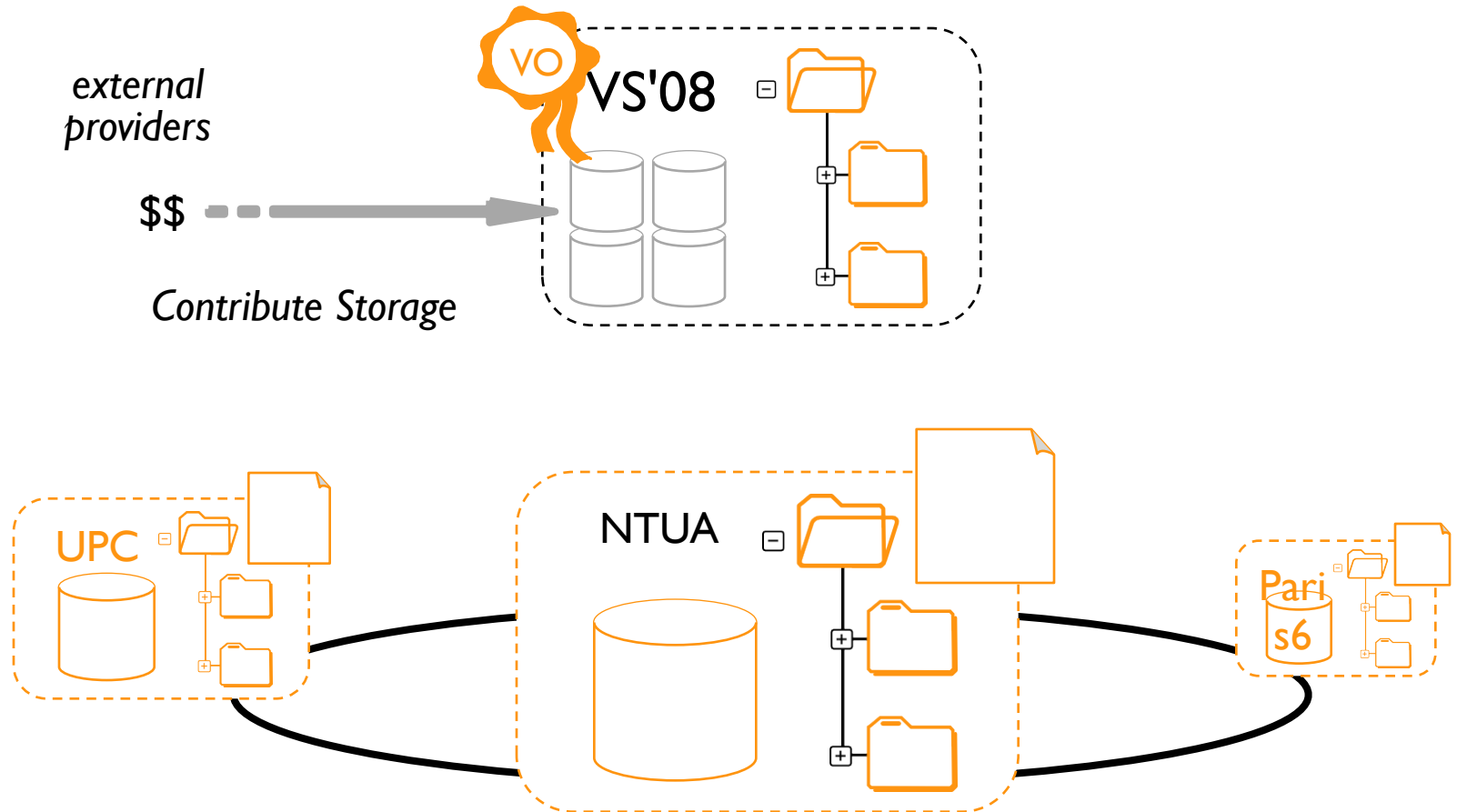
Volcano Simulation workspace



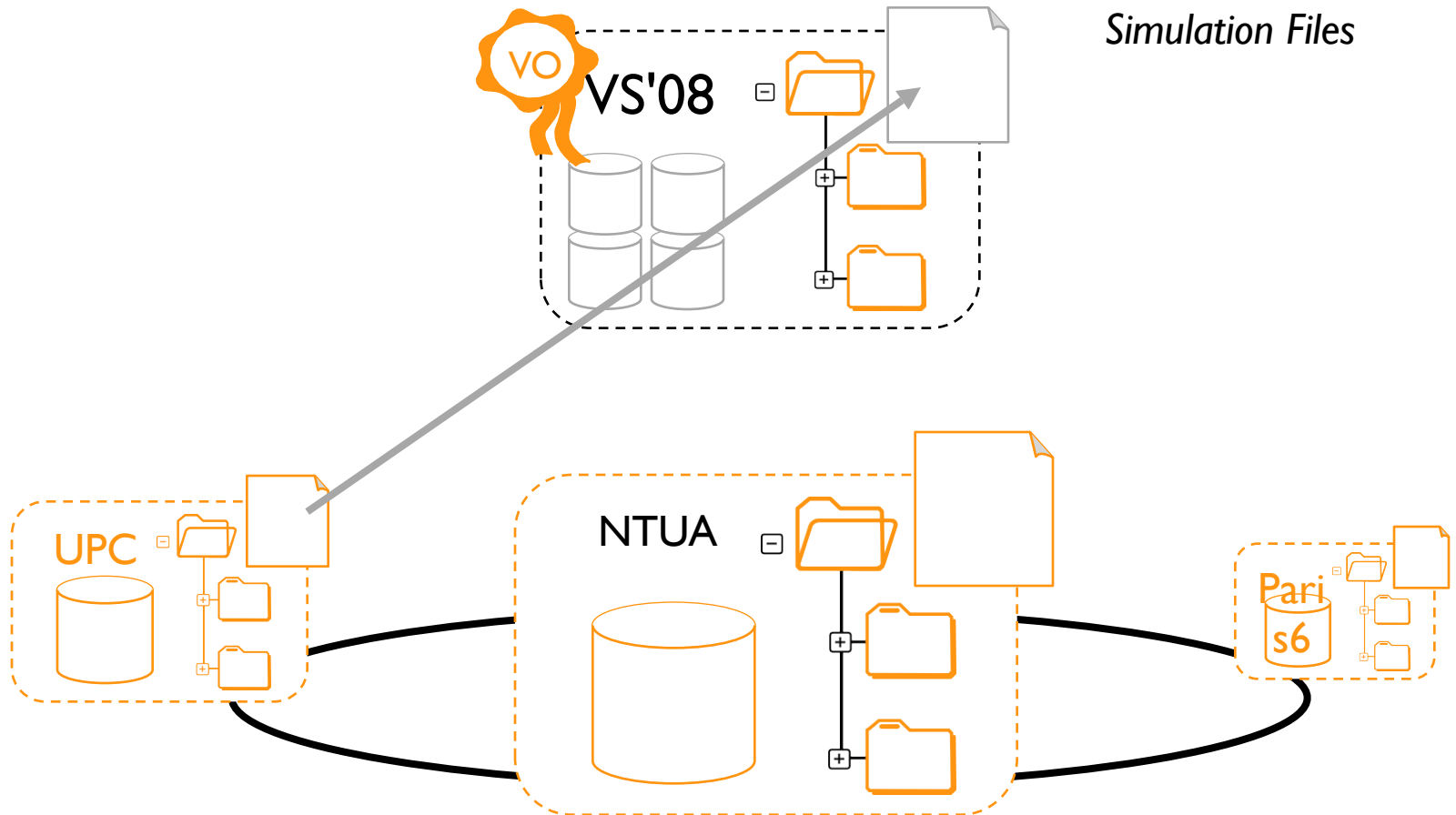
Volcano Simulation workspace



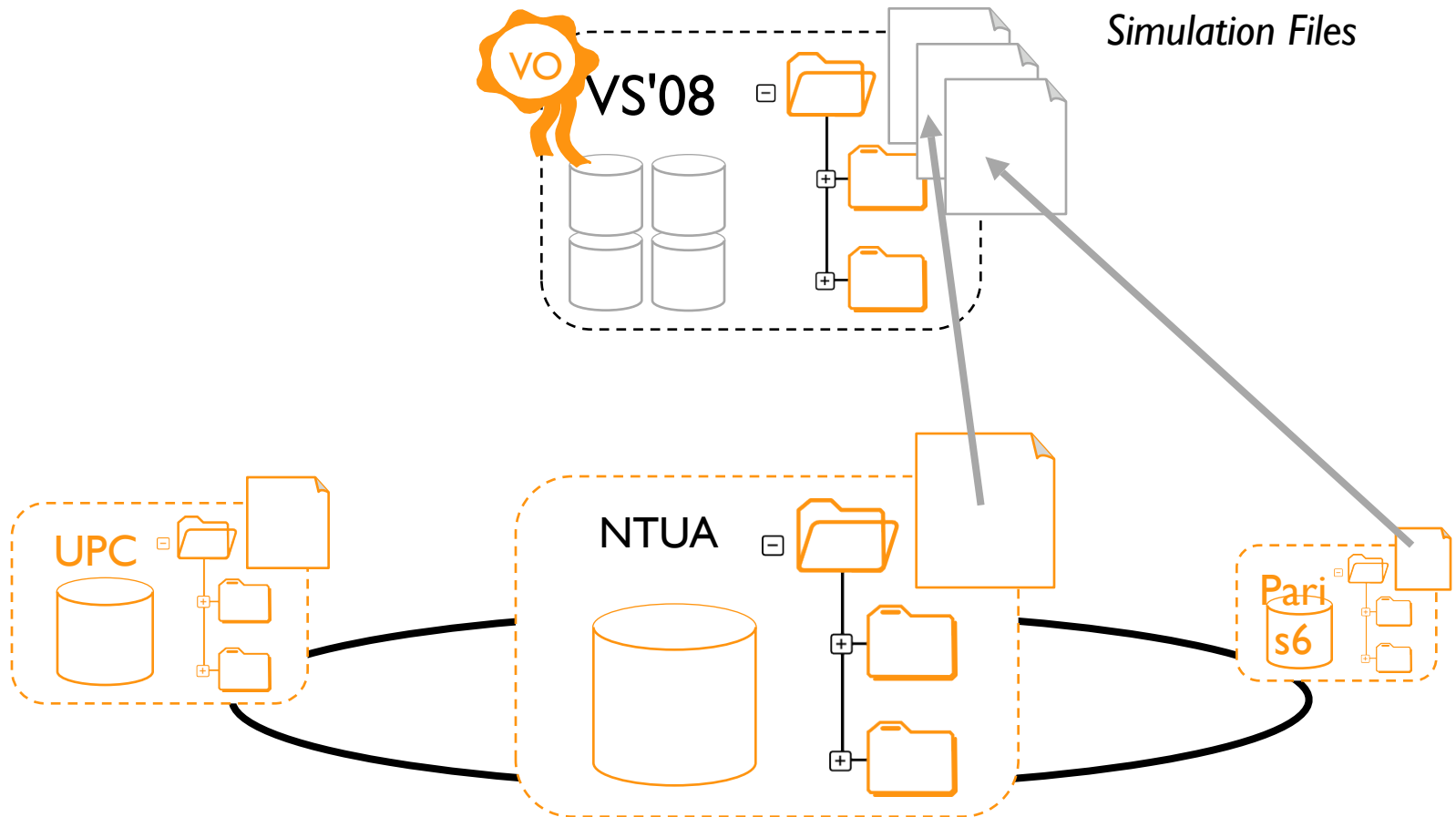
Volcano Simulation workspace



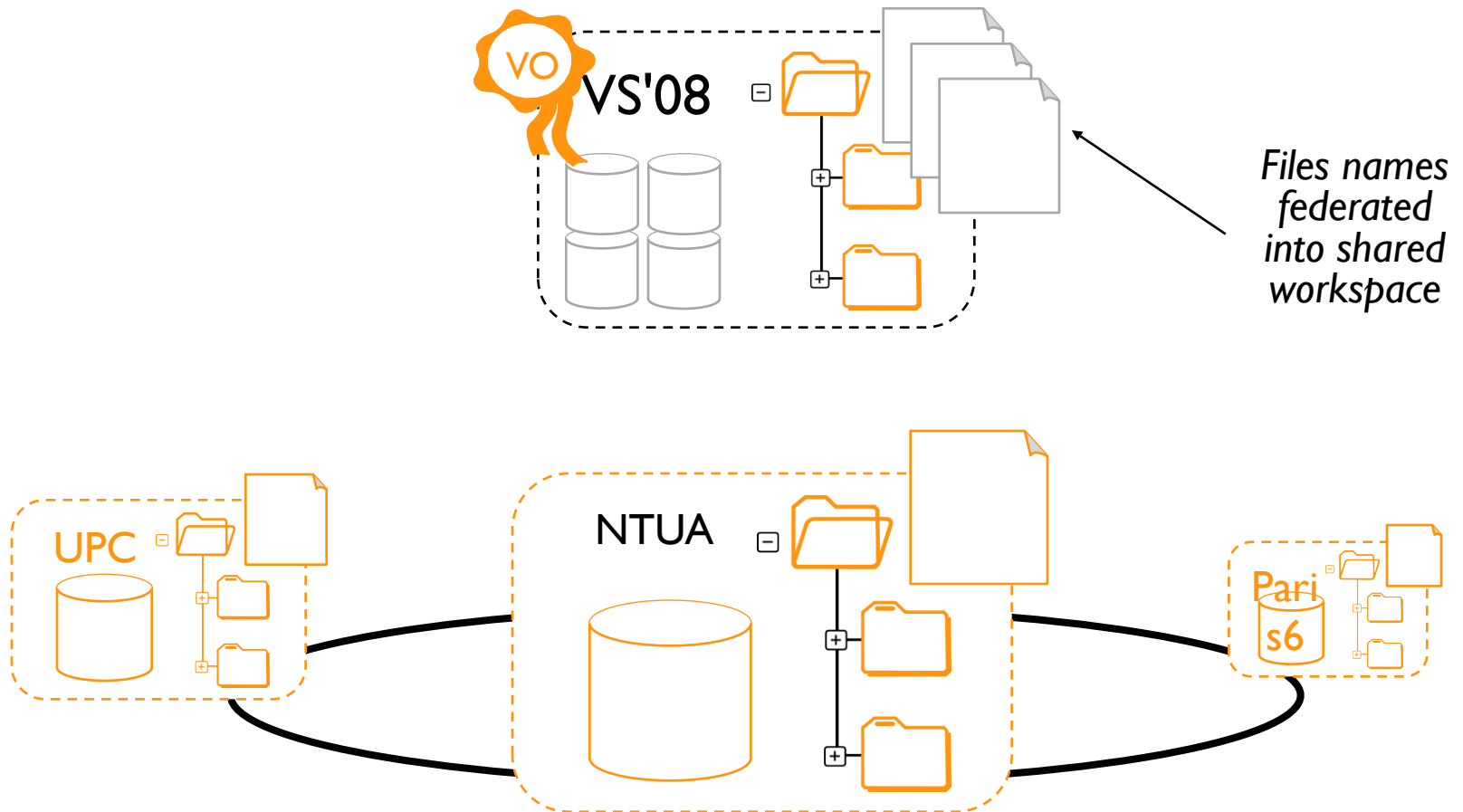
Volcano Simulation workspace



Volcano Simulation workspace



Volcano Simulation workspace



g4a Deployment

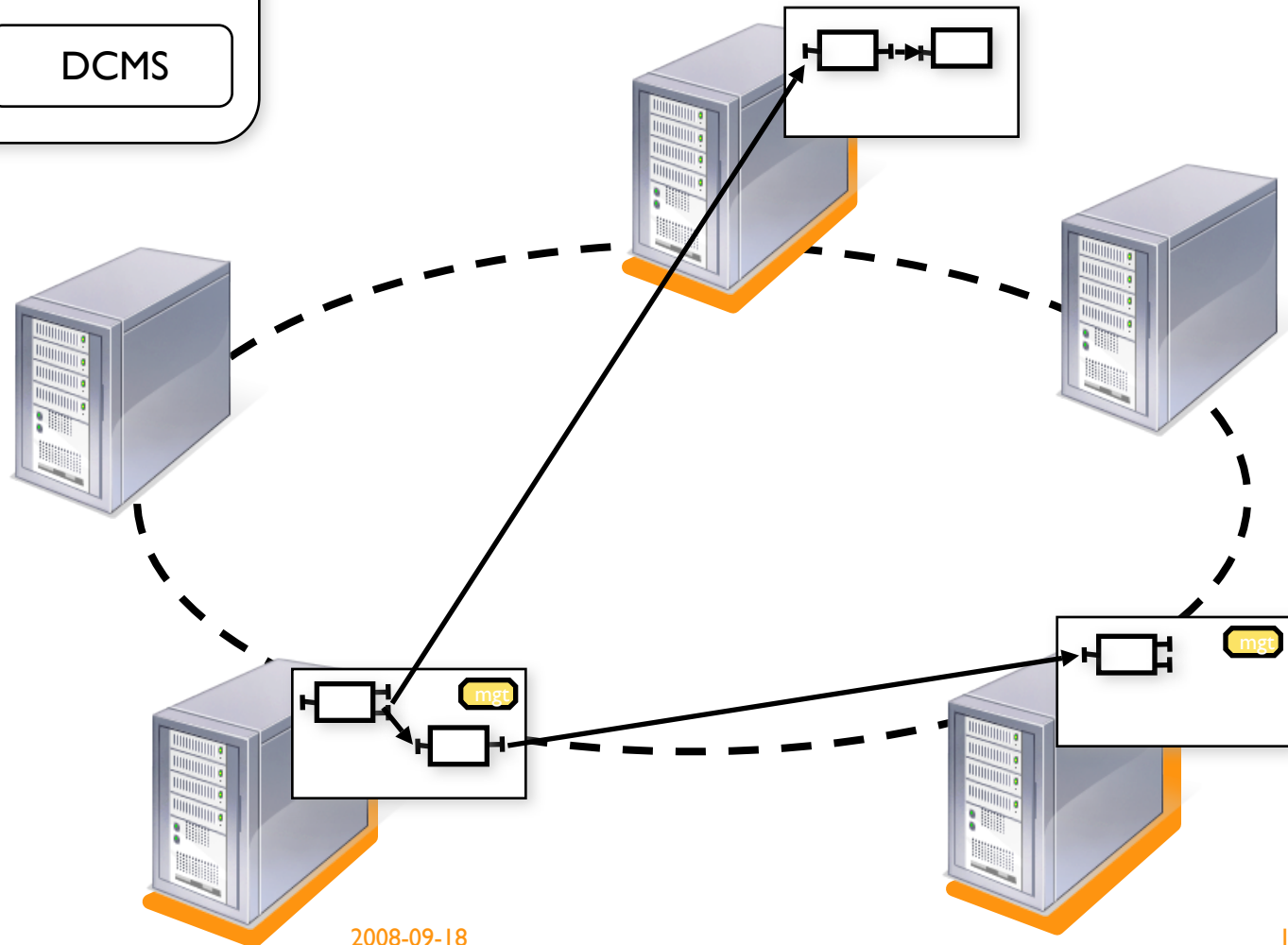


Deploymt

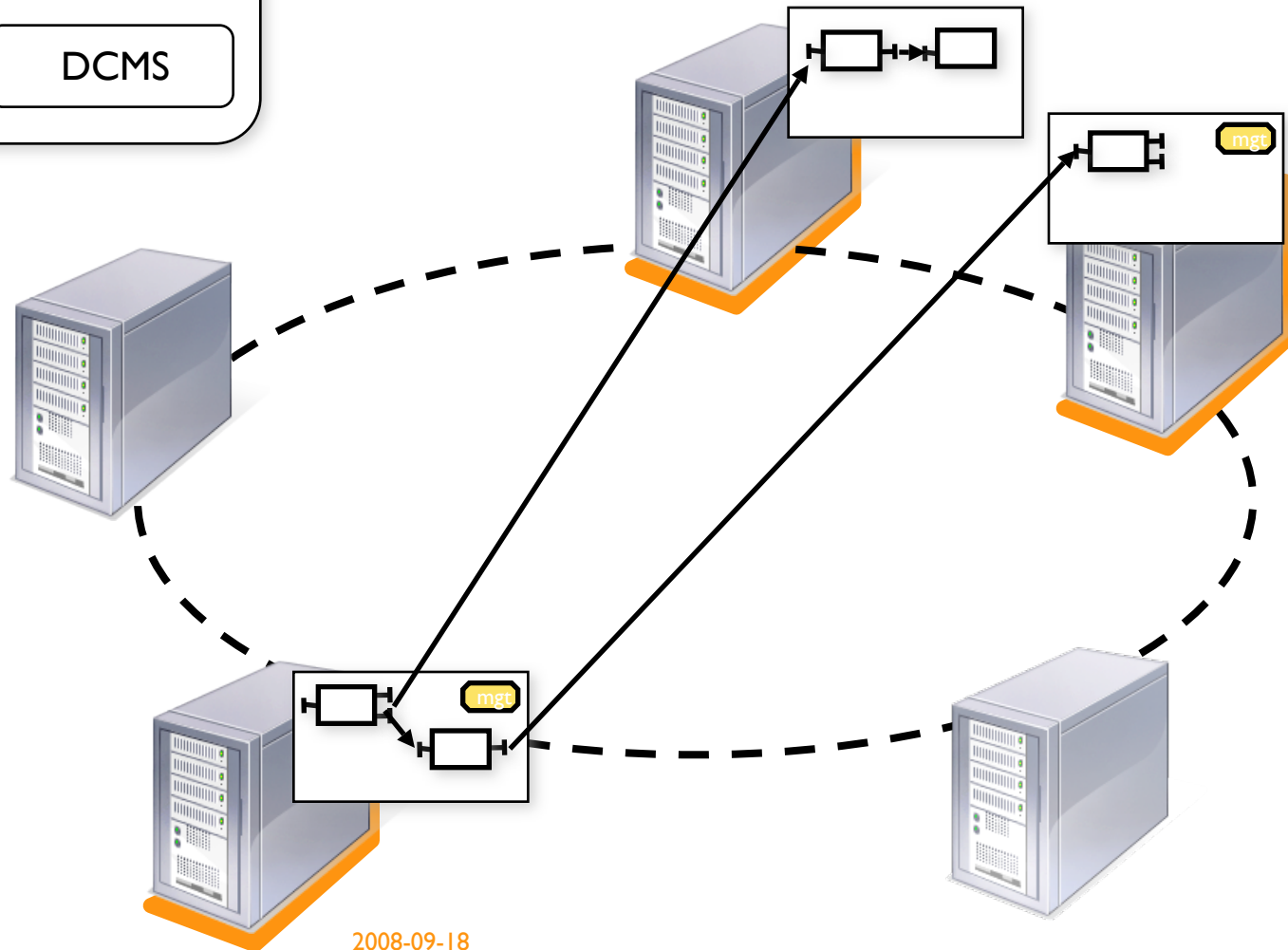
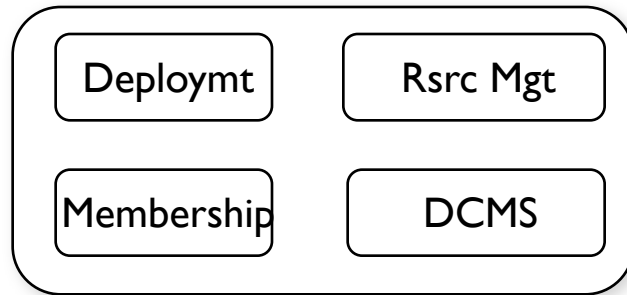
Rsrc Mgt

Membership

DCMS



g4a Autonomic response



g4a Telex Semantic Store

Shared documents updated by multiple users

- Online or offline
- Persistent in VOFS

Detect and resolve conflicts:

- Directed by semantic information from application
- Decentralised; eventual convergence

Separation of concerns

FLP = Fischer, Lynch, Patterson 85

Consensus = processes agree on value of 1 bit

Impossible if:

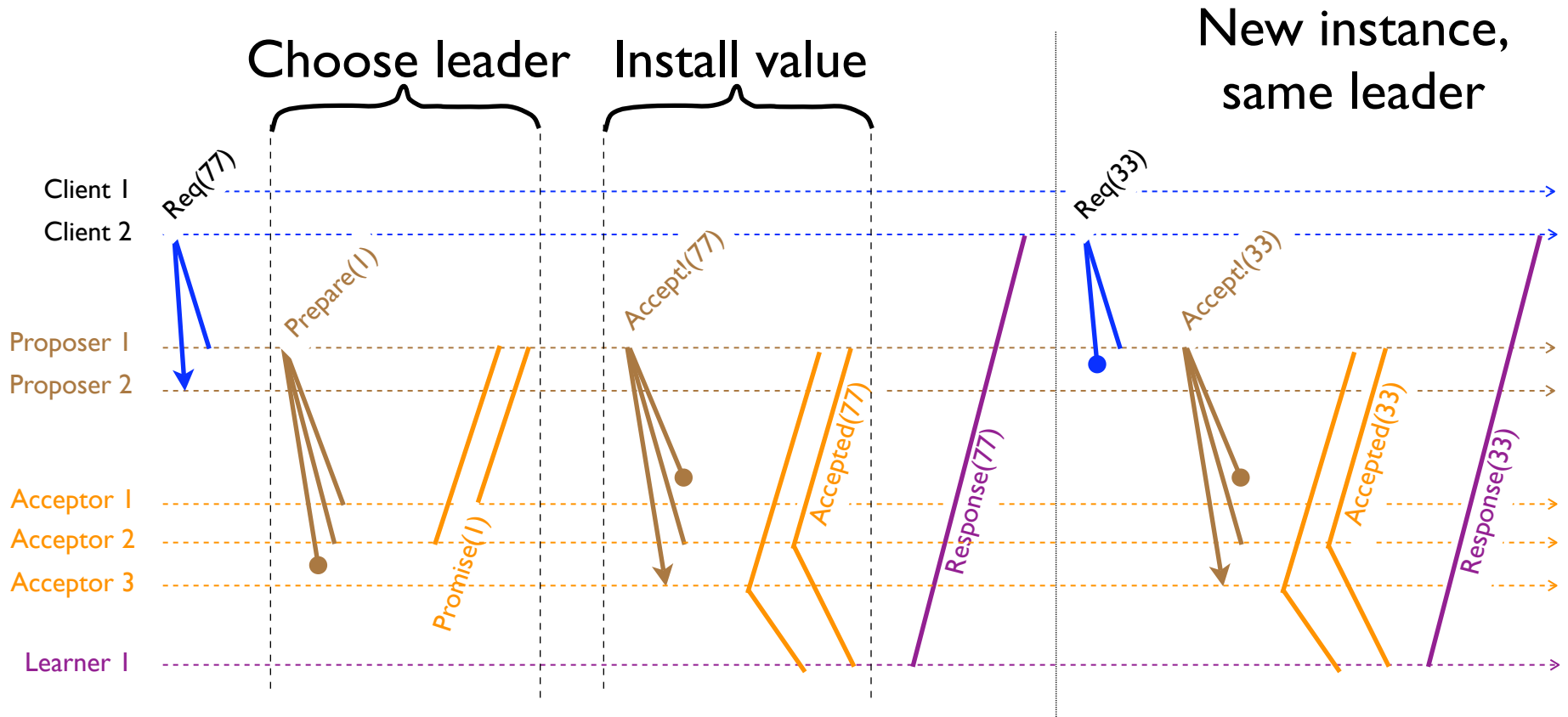
- No upper bound on message delivery
- Failures not detectable
- Deterministic

Sacrifice what?

- Safety? consensus algorithm makes mistakes
- Liveness? algorithm blocks
- Assumptions?

FLP reigns!

Paxos



Outline

General-purpose computing in the petascale era and beyond

State of the art

Operating system challenges

Summary

Operating System challenges

General-purpose: large-scale personal & community applications

Mask complexity

Parallel computing

Fault-tolerant distributed computing

Secure resource sharing

Secure, consistent, replicated mutable shared data

Energy conservation

Transparency

To hide complexity: Design primitives that operate at several scales

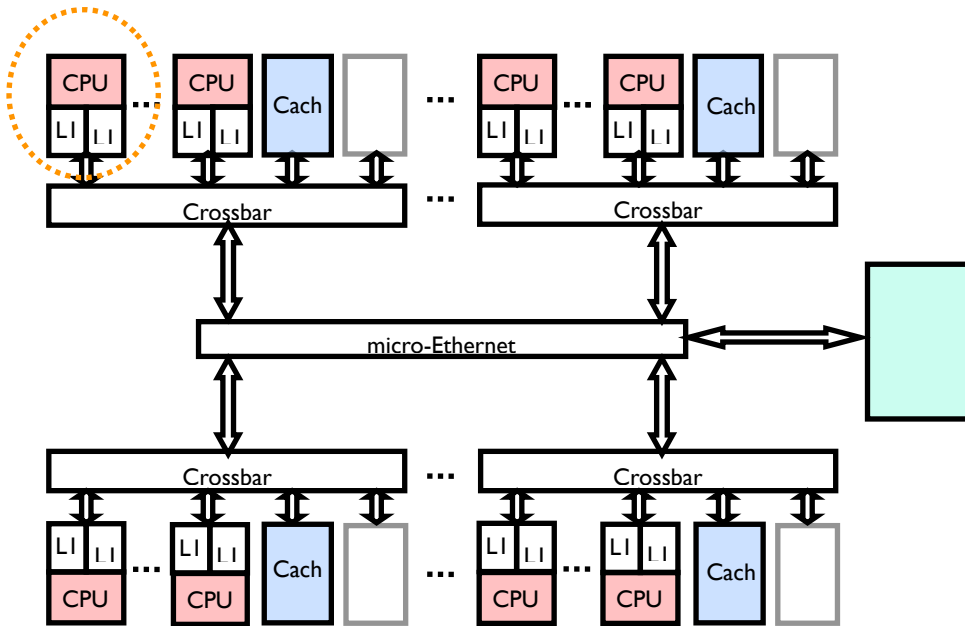
- Message send = file write
- Local or remote
- RPC, distributed shared memory, single-level store, transaction, etc.

“Leaky” abstractions: differing cost, failures, quality of service, guarantees

⇒ Transparency within limited, explicit domain

“Create a domain for atomic multicast with a latency $< 1 \mu\text{s}$; join process A, B”

Domains



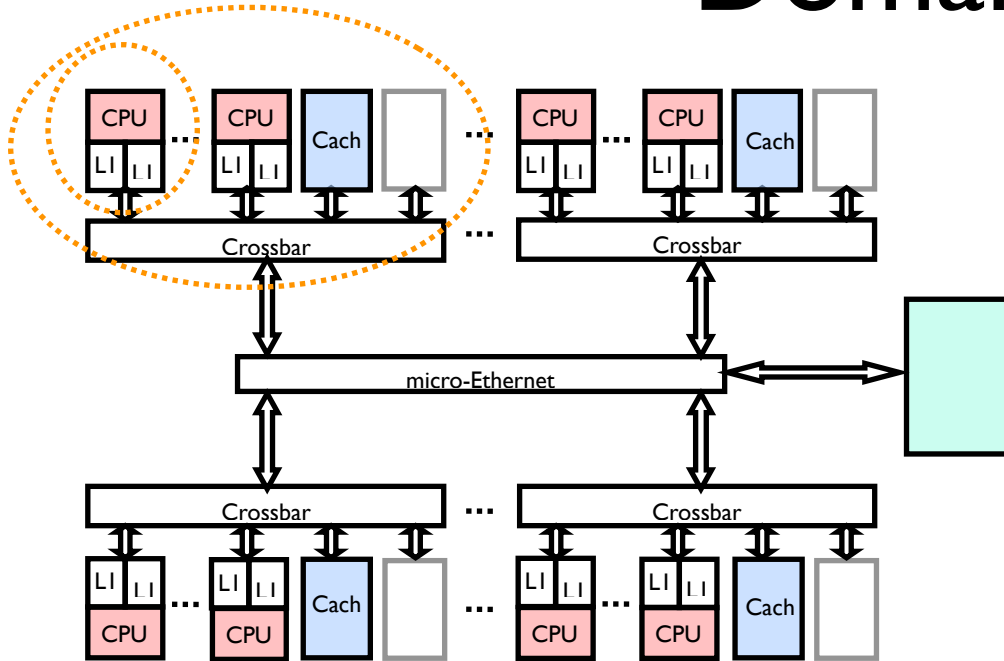
Transparency within domain

Cost, QoS, trust, etc., boundary

Physical or virtual

1st class

Domains



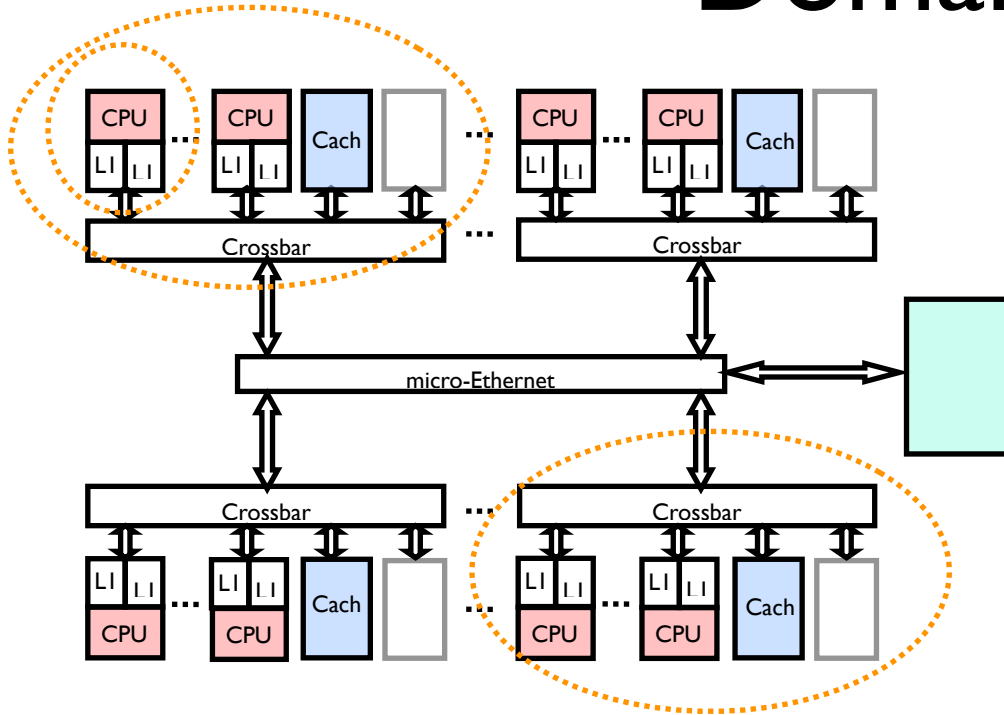
Transparency within domain

Cost, QoS, trust, etc., boundary

Physical or virtual

Ist class

Domains



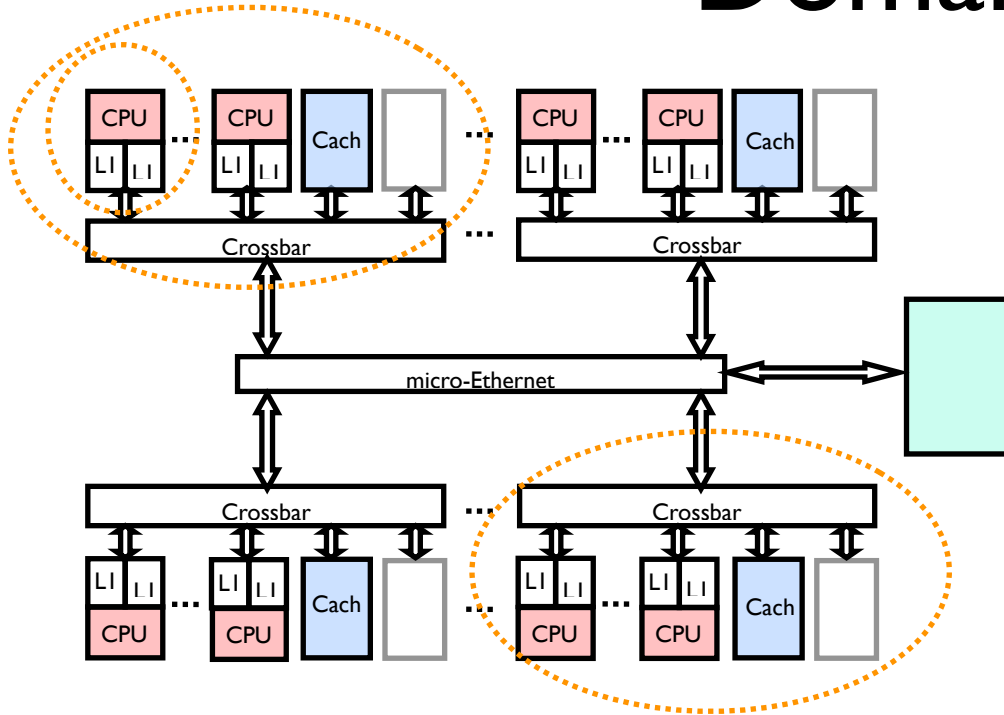
Transparency within domain

Cost, QoS, trust, etc., boundary

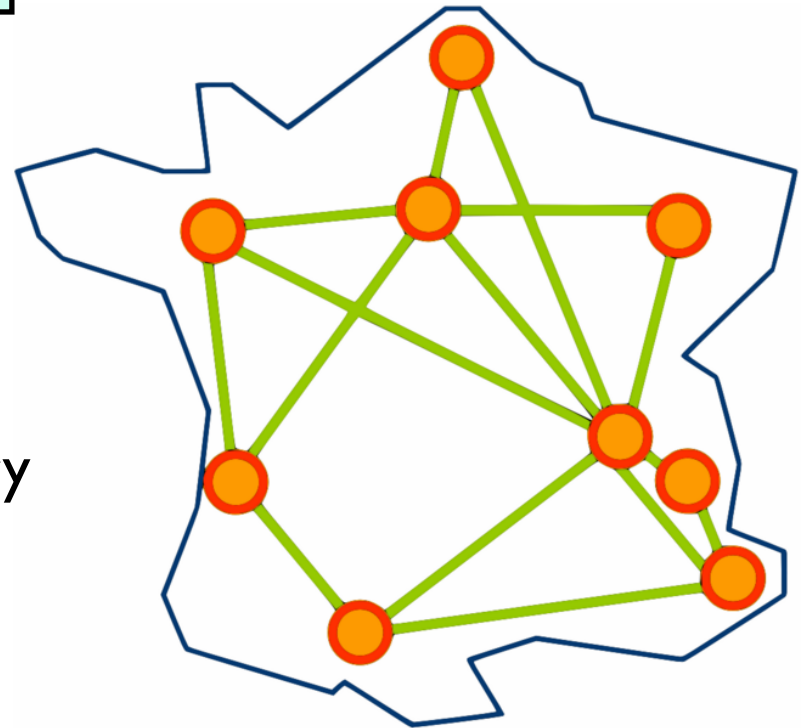
Physical or virtual

Ist class

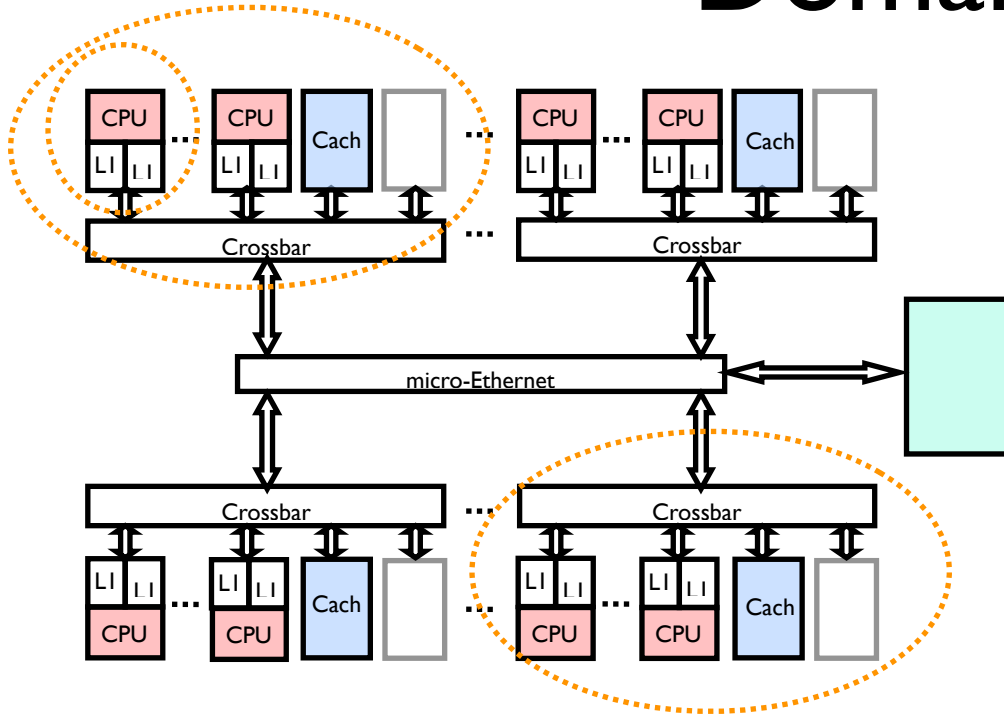
Domains



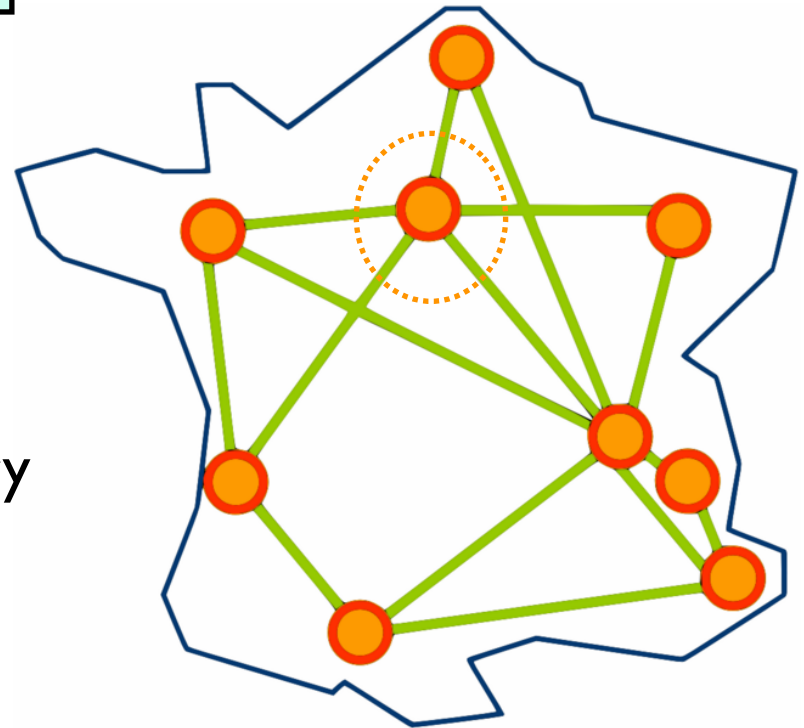
Transparency within domain
 Cost, QoS, trust, etc., boundary
 Physical or virtual
 1st class



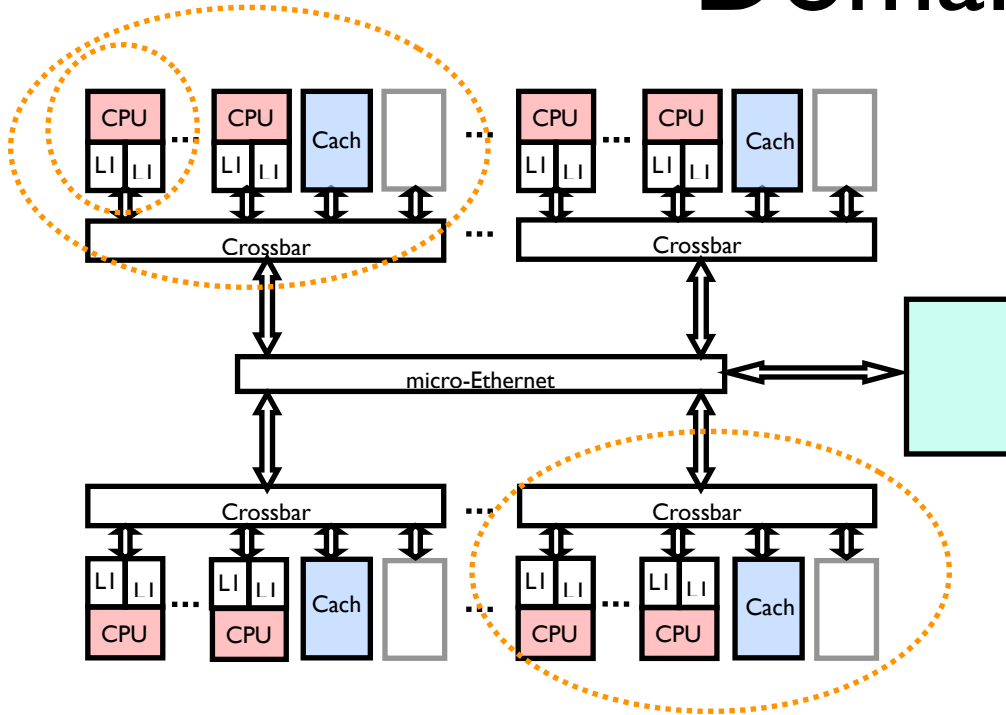
Domains



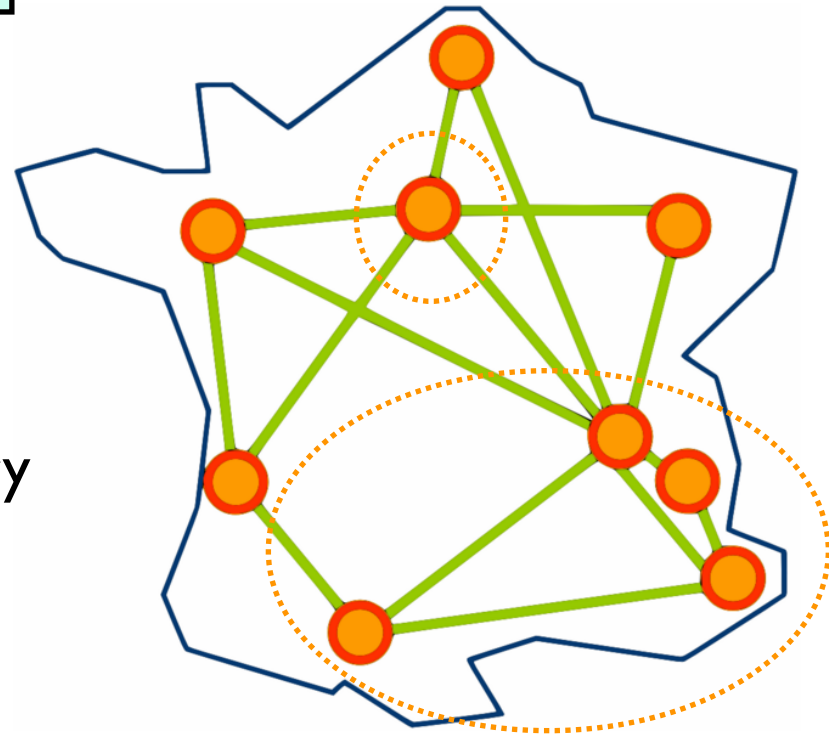
Transparency within domain
 Cost, QoS, trust, etc., boundary
 Physical or virtual
 1st class



Domains



Transparency within domain
 Cost, QoS, trust, etc., boundary
 Physical or virtual
 1st class



Massive multi-core programming

Concurrent programming is for the masses

- Languages, proof tools
- Operating system abstractions
 - Event-driven computation
 - Software transactional memory
 - Support for speculative computing
 - Opportunistic computing (e.g. cache-first)
- Virtual machine
 - Execution environment
 - Garbage collection

Software transactional memory

```
atomic {  
    x := y * 2;  
    y := z + x;  
    if (t < 0) retry;  
}
```

```
atomic {  
    t := x + 1;  
    if (u < 0) abort;  
}
```

High-level concurrent programming primitive:

- Group of arbitrary memory access
- All-or-nothing, isolated, durable (in memory)
- $\approx$ composable
- Optimistic implementation, decent performance

Raises level of abstraction (a bit)

Non-transactional access? Big transactions? I/O?

Universal Virtual Machine

Intermediate language for parallel, distributed programs

- Performance scalability: compile once, execute anywhere
- Hardware isolation: number of cores, cache sizes, instruction sets, CPU/GPU
- Expose useful parallelism
- Encapsulate Transactional Memory
- Affinity allocation & scheduling

LLVM

- Dynamic hardware adaptation
- Run-time code generation for parallel machines

Secure and reliable communication

Multi-scale group management

- Process group
- Dynamic join / leave (P2P: churn)
- Partial connectivity (firewalls, partitions)

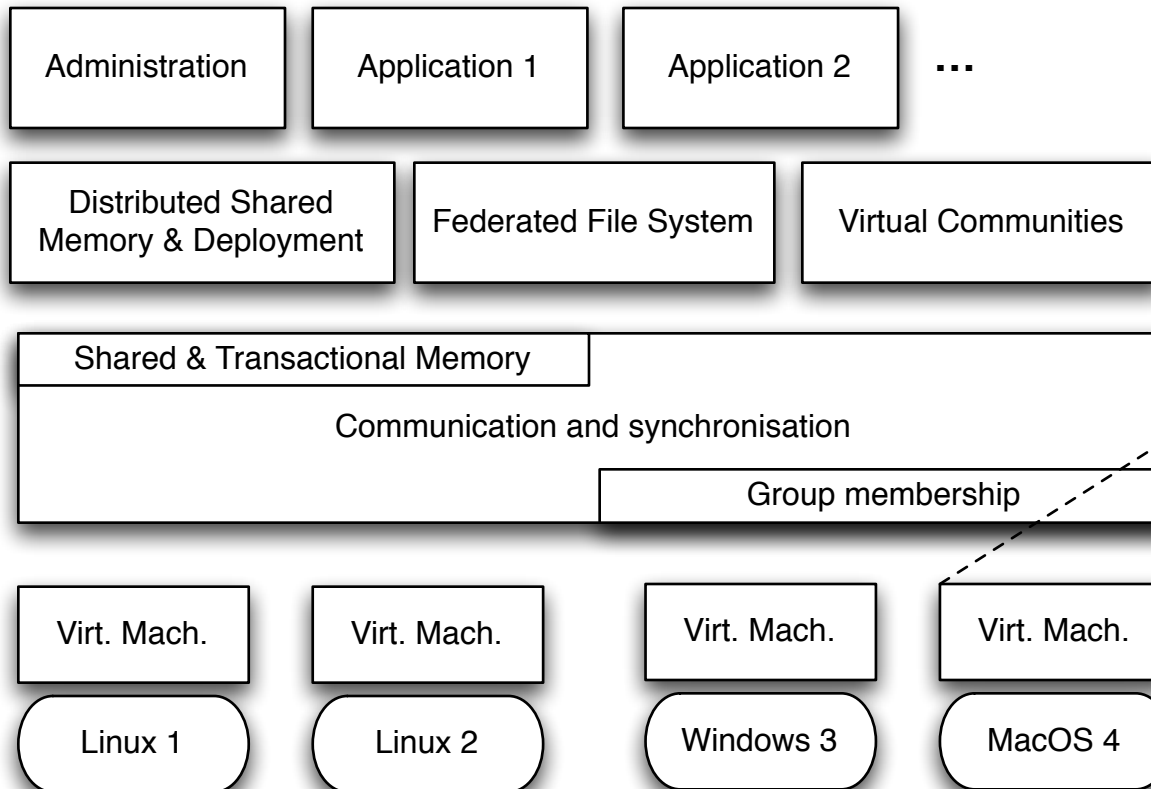
Failure detection

Consensus primitives

- Transactions: memory, file system, communication
- Atomic Broadcast, Group Multicast
- Tolerate free-riders, colluding, etc.
- At different scales / different costs / different guarantees

Replication and consistency

Architecture



Transverse issues:

- Security
- Churn / fault tolerance
- Multi-scale: uniform interfaces, non-uniform costs, locality
- Self-administration

Virtual Machine:

- (Local) Communication & synchronization
- (Local) File System
- (Local) Shared & Transactional Memory
- (Local) Deployment

Outline

General-purpose computing in the petascale era and beyond

State of the art

Operating system challenges

Summary

Summary

Petascale for the masses

- Massive computing, storage is available
- Portal to fluid network
- Very non-uniform
- Cloud and Edge computing converge
- Asynchronous, failure-prone $\Rightarrow$ FLP

Concurrency inherently complex

- Formal verification is essential

Transparency vs. control

Energy = scarce resource, to be managed at all scales