# The OS/App Inversion: escaping POSIX to bring Git to the datacentre

QCon New York Mini talk

Anil Madhavapeddy (speaker)
with Thomas Gazagnaire and Benjamin Farinier
University of Cambridge Computer Laboratory

June 7, 2015

# Common features every distributed system needs

- **Persistence** for fault tolerance and scaling
- **Scheduling** of communication between nodes
- **Tracing** across nodes for debugging and profiling

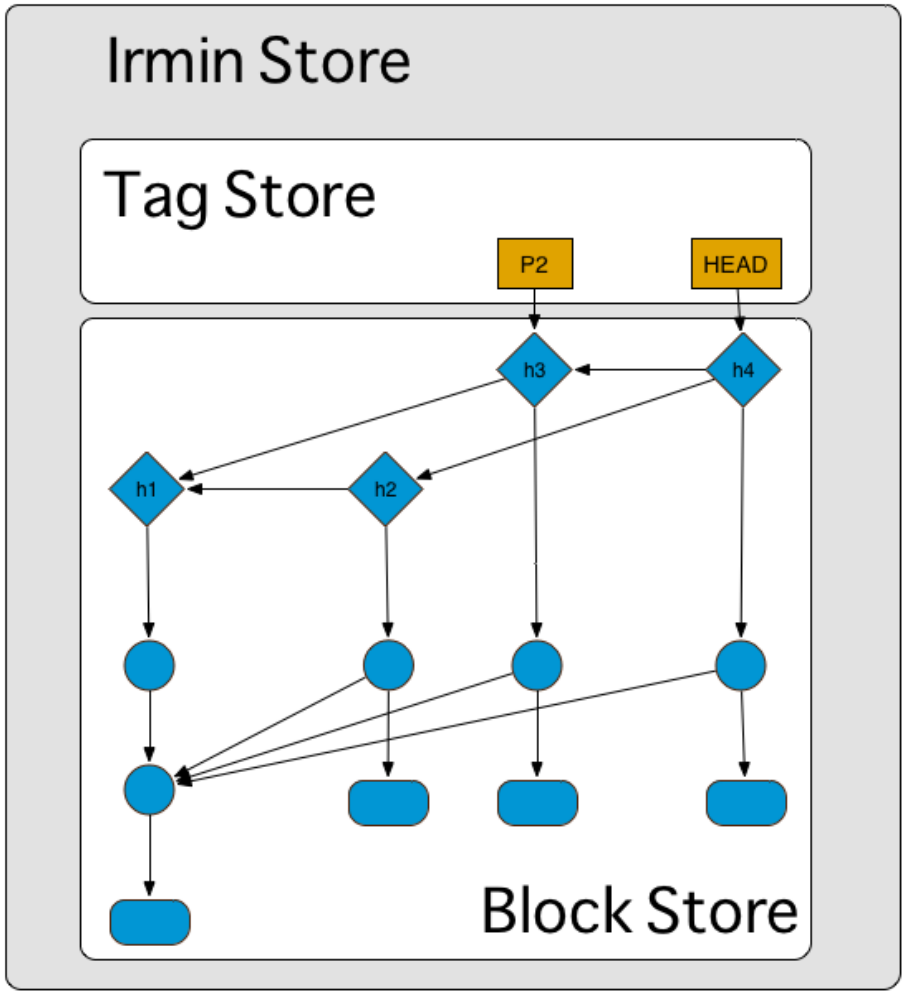
Most distributed systems run over an operating system, and so are stuck with the OS kernel exerting control. We use *unikernels*, which are application VMs that have complete control over their resources.

# What if we just used Git?

- **Persistence**
  - `git clone` of a shared repository across nodes
  - `git commit` of local operations in the node
- **Scheduling**
  - `git pull` to receive events from other nodes
  - `git push` to publish events to other nodes
- **Tracing and Debugging**
  - `git log` to see global operations
  - `git checkout` to roll back time to a snapshot
  - `git bisect` to locate problem messages
- **New Problems**
  - `git rebase` needed for distributed garbage collection
  - Shelling out to `git` is slow and lacks control

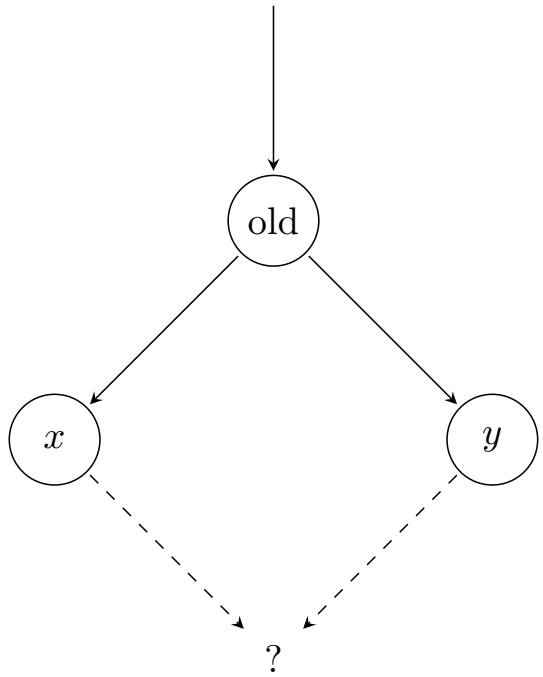
# Irmin, large-scale, immutable, branch-consistent storage

- Irmin is a library to **persist** and **synchronise distributed data structures** both on-disk and in-memory
- It enables a style of programming very similar to the **Git workflow**, where distributed **nodes fork, fetch, merge and push** data between each other
- The general idea is that you want every active node to get a **local** (partial) **copy of a global database** and always be very explicit about how and when data is shared and migrated


Irmin Store
Tag Store
P2
HEAD
h3
h4
h1
h2
Block Store

```
type t = ...
(** User-defined contents. *)
type result = [ `Ok of t |
    `Conflict of string ]

val merge: old:t → t → t →
    result
(** 3-way merge functions. *)
```



# Irmin Features

- Still pre 1.0, but several useful datastructures such as distributed queues and efficient ropes.
- HTTP REST for remote clients, or library via OCaml.
- JavaScript compilation for pure browser operation.
- Bidirectional operation, so `git` commits map to Irmin commits from any direction.
- Open source at `https://irmin.io`
- **Want to know more?** I'm giving a full talk on this on Friday at 1015 titled "Functional Distributed Programming with Irmin"!

# Teaser: Xen Toolstack using Irmin

https://www.youtube.com/watch?v=DSzvFwIVm5s