

---

# The Design of a new Persistent Object Store for PJama

**T. Printezis, L. Daynès, M. Atkinson,  
S. Spence, and P. Bailey**

tony@dcs.gla.ac.uk



University of Glasgow



PJama Team

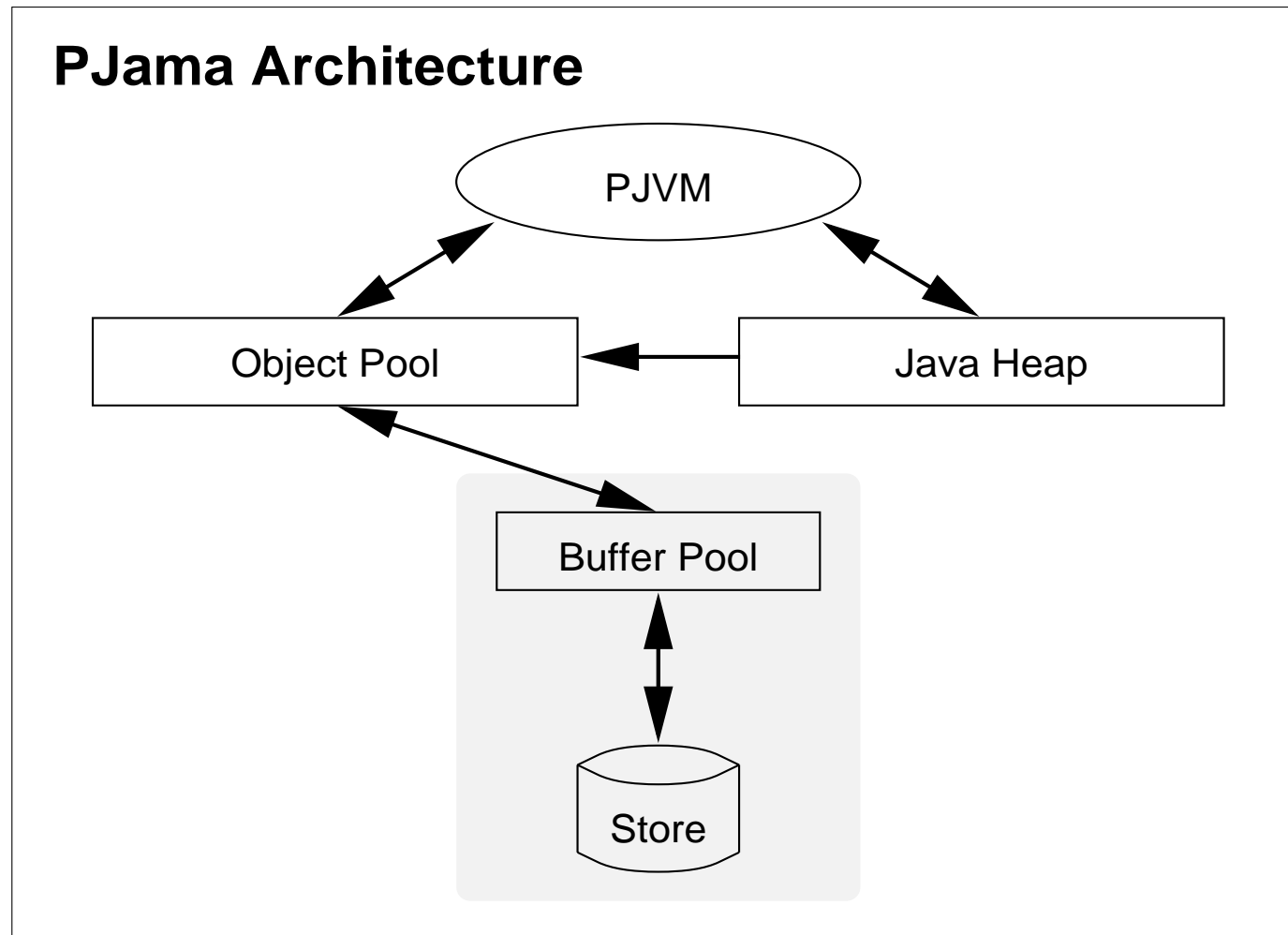
# Overview

---

- Introduction
- High-Level Store Organisation
- Partition Organisation
- Current Status

# Intro - *Current PJama Architecture*

---



# Intro - *What's a Persistent Object Store?*

---

- Similar to a large non-volatile heap
- Stores objects
- Object format similar to memory format
- Retains type-information
- Retains connectivity
- Persistence by reachability
- Fault-tolerance (stays consistent after crashes)

## Intro - *Why?*

---

- Do we need yet another Persistent Object Store?
- Poor maintenance of research-type products
- “Bad” experiences with RVM from CMU
- Nothing commercially available which suits us
- Intellectual property right problems

# Intro - *Design Goals*

---

- Independent of VM / “self-describing” store
- Complete orthogonality
- 10GB size at least...
- Continuous execution
- Distribution
- Parallel transactions

# Store Organisation - *Partitions*

---

- Store too large to GC in one go
- Store split into partitions
- Include enough info to GC them independently
- => Incremental disk GC, one partition at a time

# Store Organisation - *Operations on Partitions*

---

- Several operations are defined on partitions, eg.
  - Allocate new object
  - De-allocate object
  - Garbage collect partition

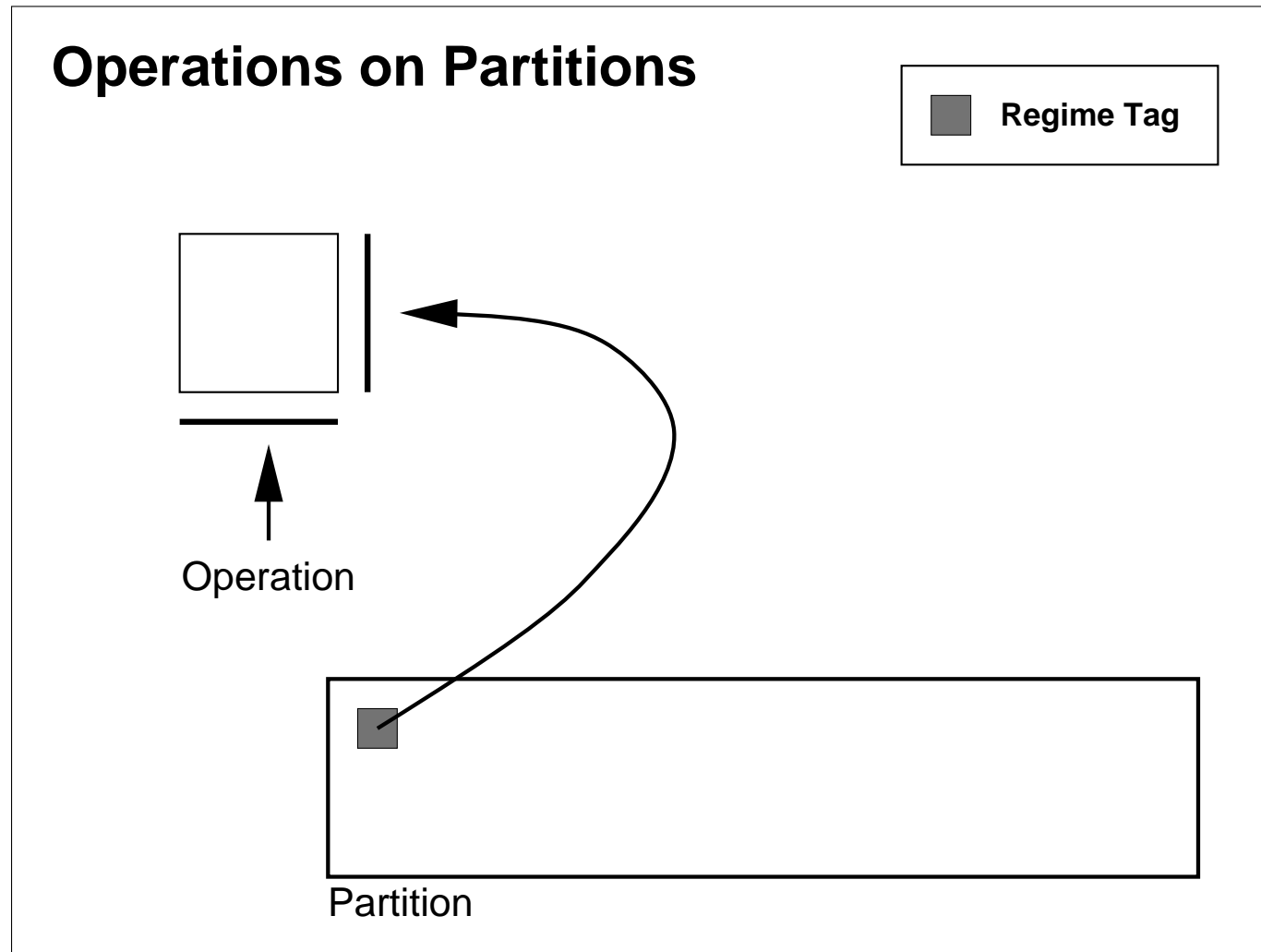
# Store Organisation - *Partition Regimes*

---

- *Partition Regime*: a specific implementation of the partition operations
- Example:
  - Operation: object allocation
  - Implementation: depends on the free-space management: free-list, bitmap, compaction, etc.
- Several regimes might co-exist in the same store, applied to different partitions

# Store Organisation - *Partition Operations*

---

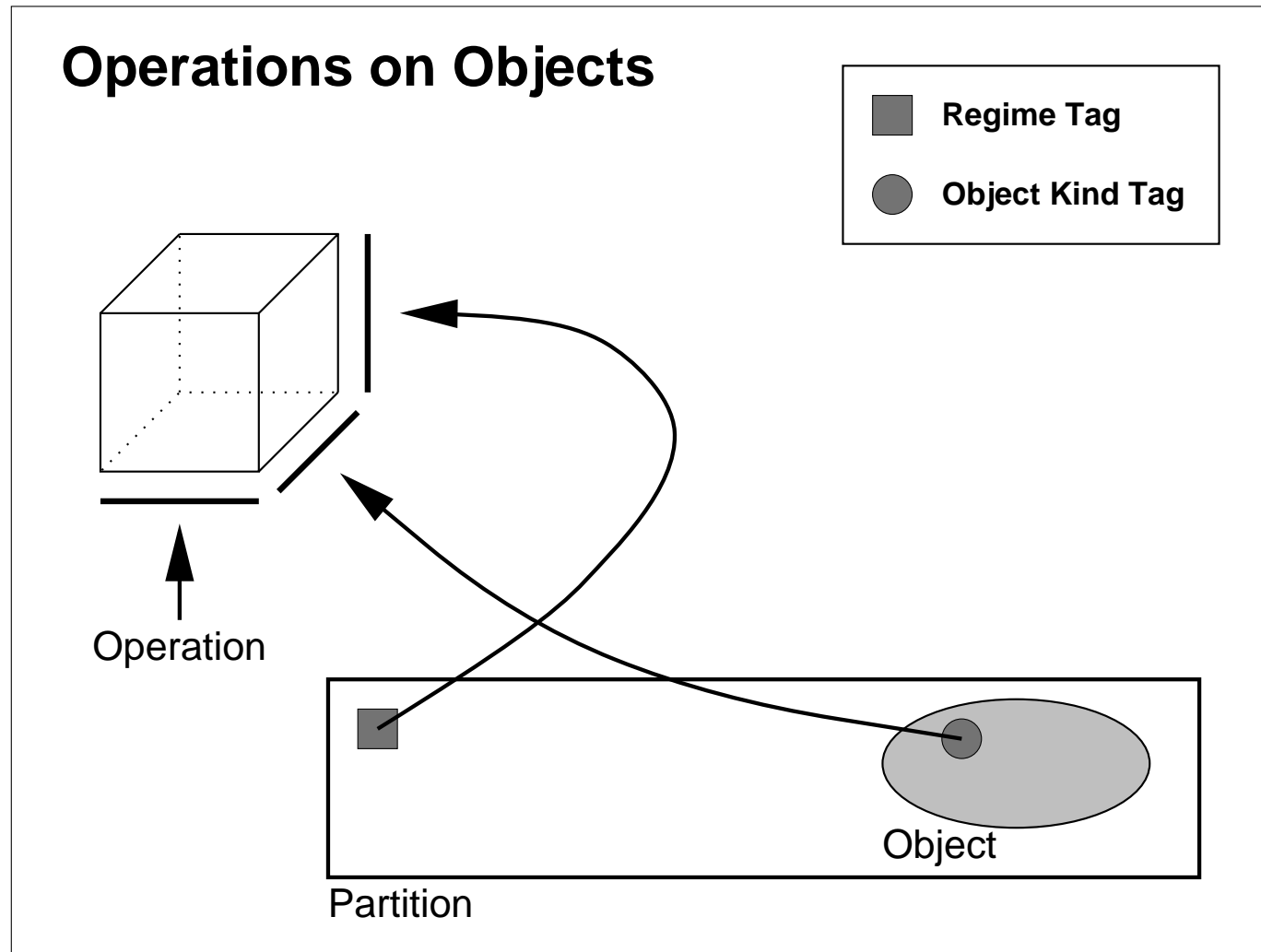


# Store Organisation - *Object Kinds*

---

- PJama needs at least 4 different *Object Kinds*
  - Class Objects
  - Instances
  - Arrays
  - Bytecodes
- Several operations are defined on objects, eg.
  - Identify pointers
  - Swizzle object
  - Update field

# Store Organisation - *Object Operations*

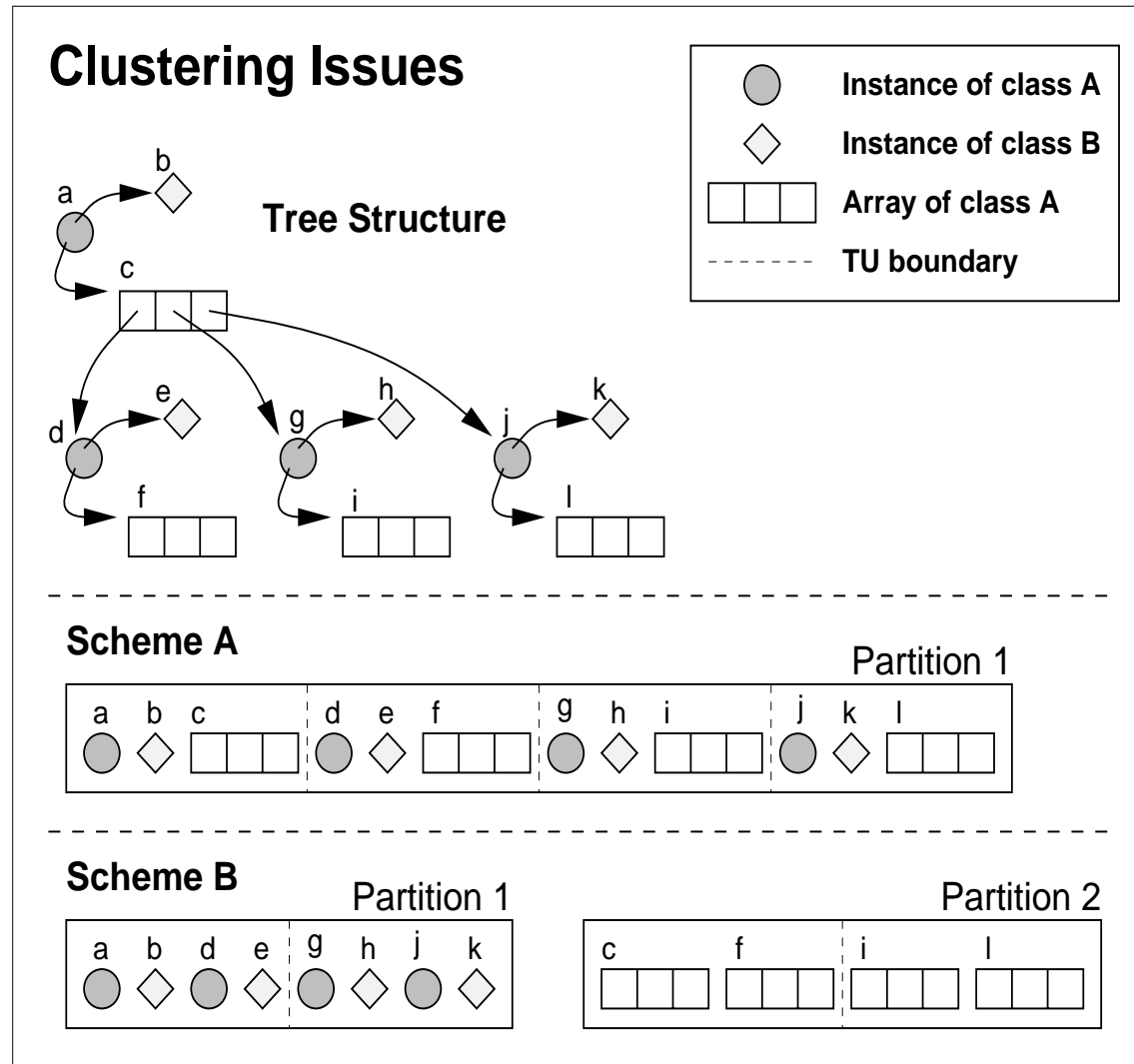


# Store Organisation - *Regimes and Kinds*

---

- Can cluster objects inside partitions according to
  - Locality
  - Object kinds
- We will use the latter
- Locality not totally lost!

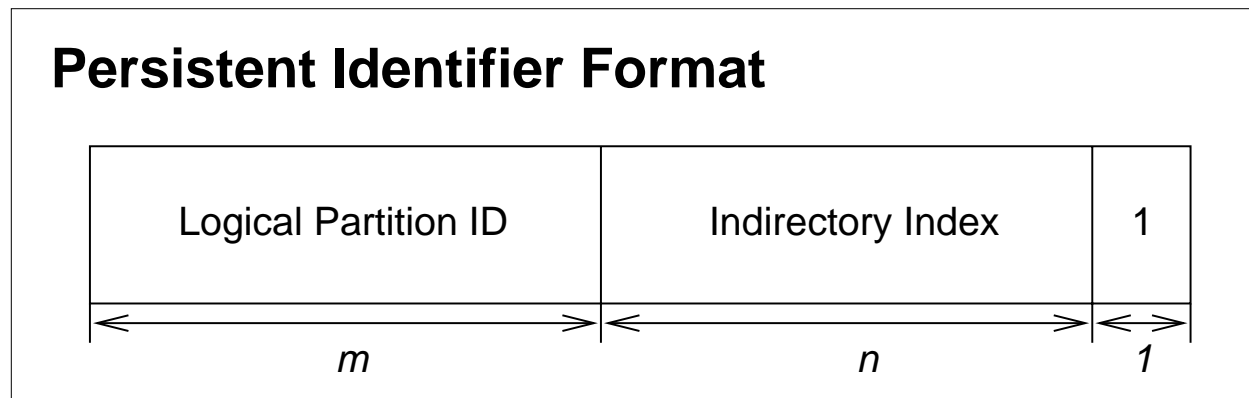
# Store Organisation - *Clustering*



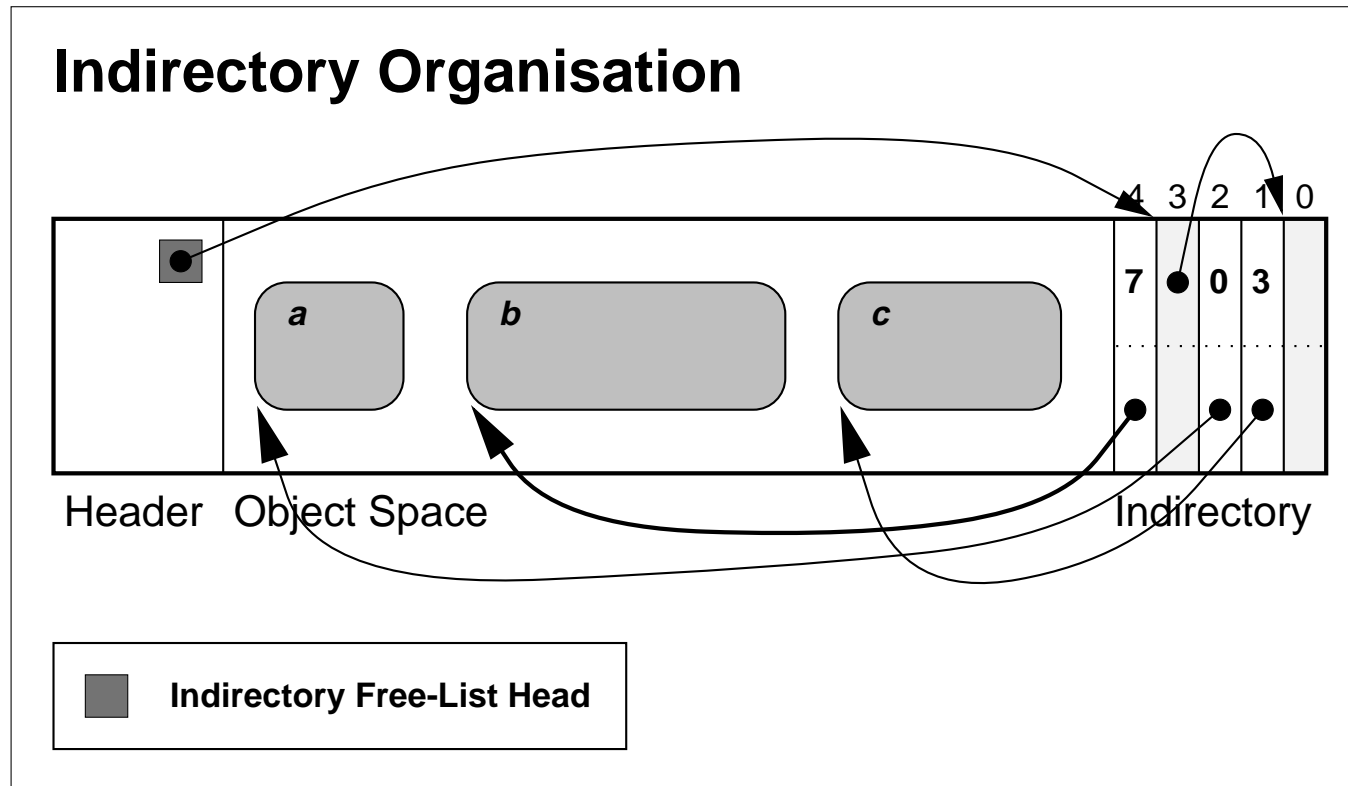
# Partition Organisation - *PIDs*

---

- Persistent Identifiers for Objects
- Can either
  - Have direct pointer to the object (physical addressing)
  - Introduce one level of indirection (logical addressing)
- Objects can be moved more easily in latter case

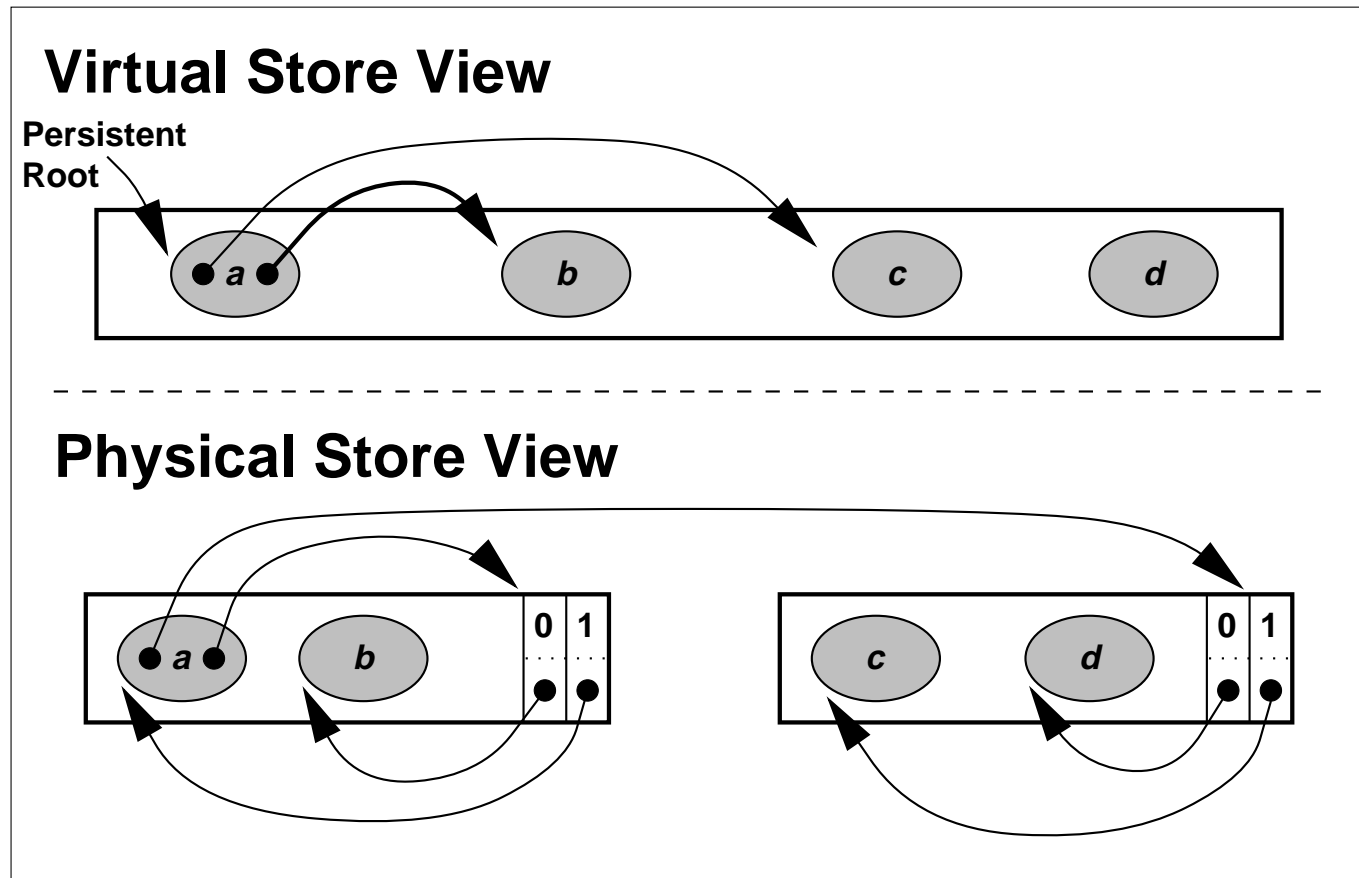


# Partition Organisation - *Indirectory*



# Partition Organisation - *References*

---



# Partition Organisation - *Identifying Pointers*

---

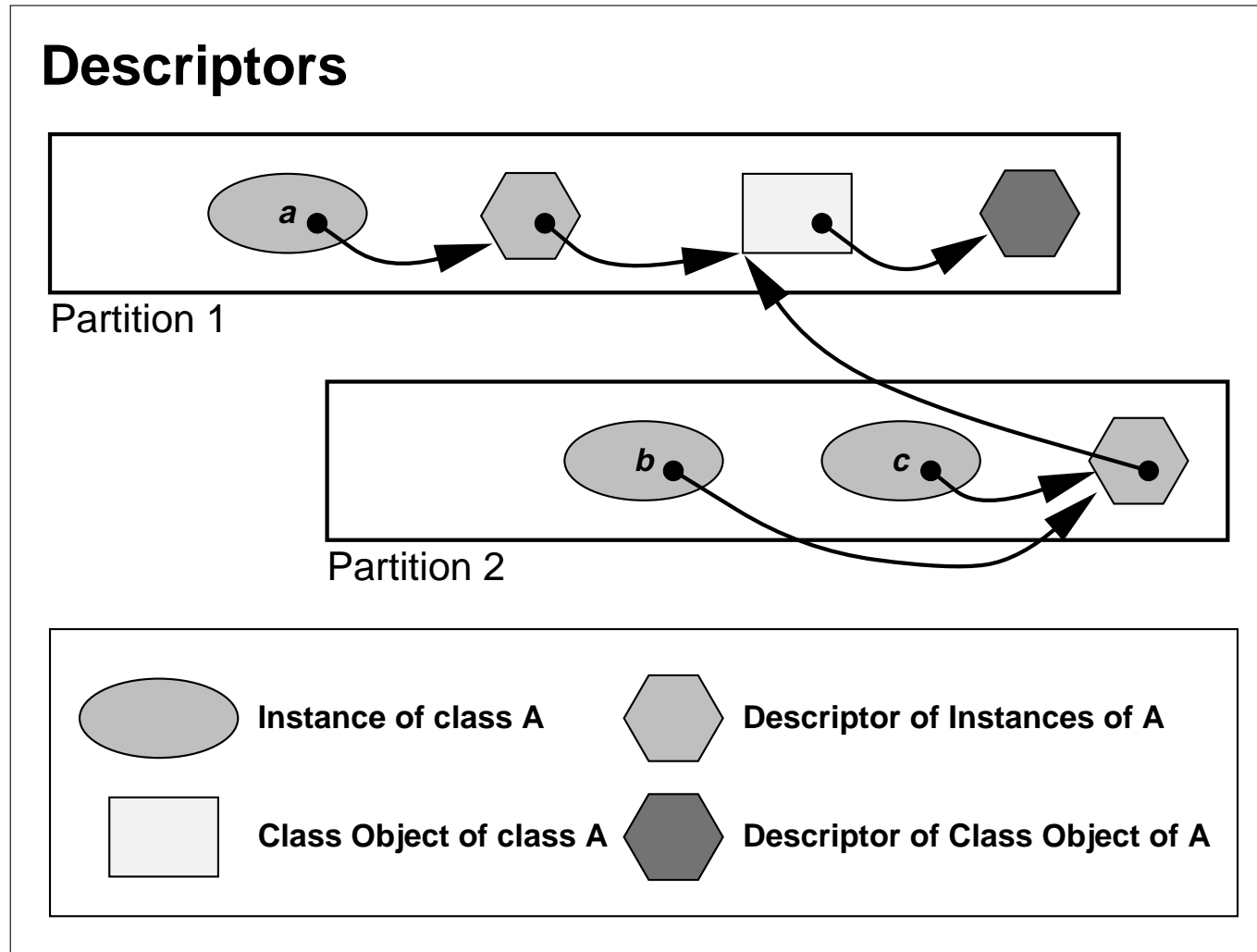
- Should be able to identify pointers inside Objects efficiently, uniformly, and independently of the VM
- For some Object Kinds this is trivial
  - Scalar Arrays
  - Bytecodes
  - Object Arrays
- For others it is not
  - Instances
  - Class Objects

# Partition Organisation - *Descriptors*

---

- Introduce a new Object Kind: *Descriptors*
- A Descriptor contains
  - Information about objects of the same structure
  - At least where the pointers are
- Can also be extended to contain
  - Information on field types (for byte-order transforms)
  - Information on finalisation

# Partition Organisation - *Use of Descriptors*



# Current Status

---

- Design finished
- Implementation started in mid-June
- Basic infrastructure finished
- Single and double dispatch mechanisms implemented

# To Do

---

- Implementation of the dispatch tables
- Recovery
- First “stable” version incorporated in the PJama later on in the year...