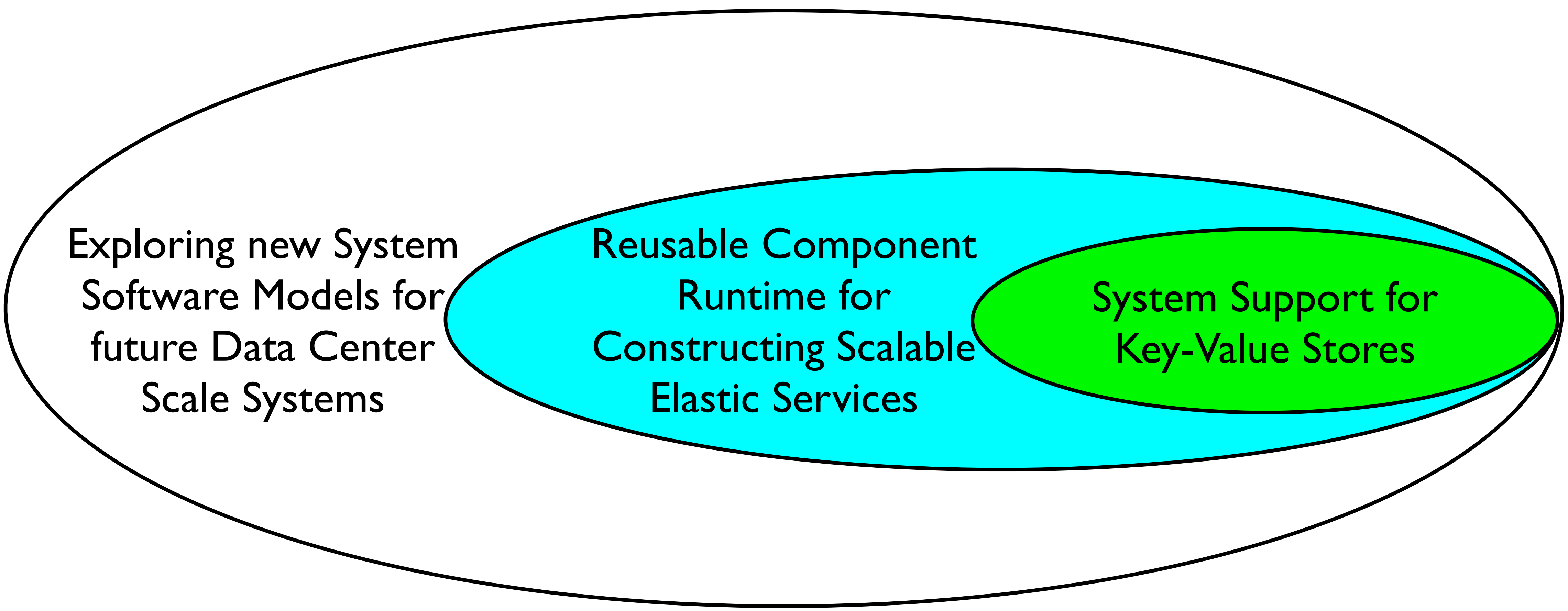
Studying System Support for a Key Value Store

Boston University SESA Group
Dan Schatzberg

Outline

0. Context
1. High-level architecture
2. Software decomposition
3. Key-value store support



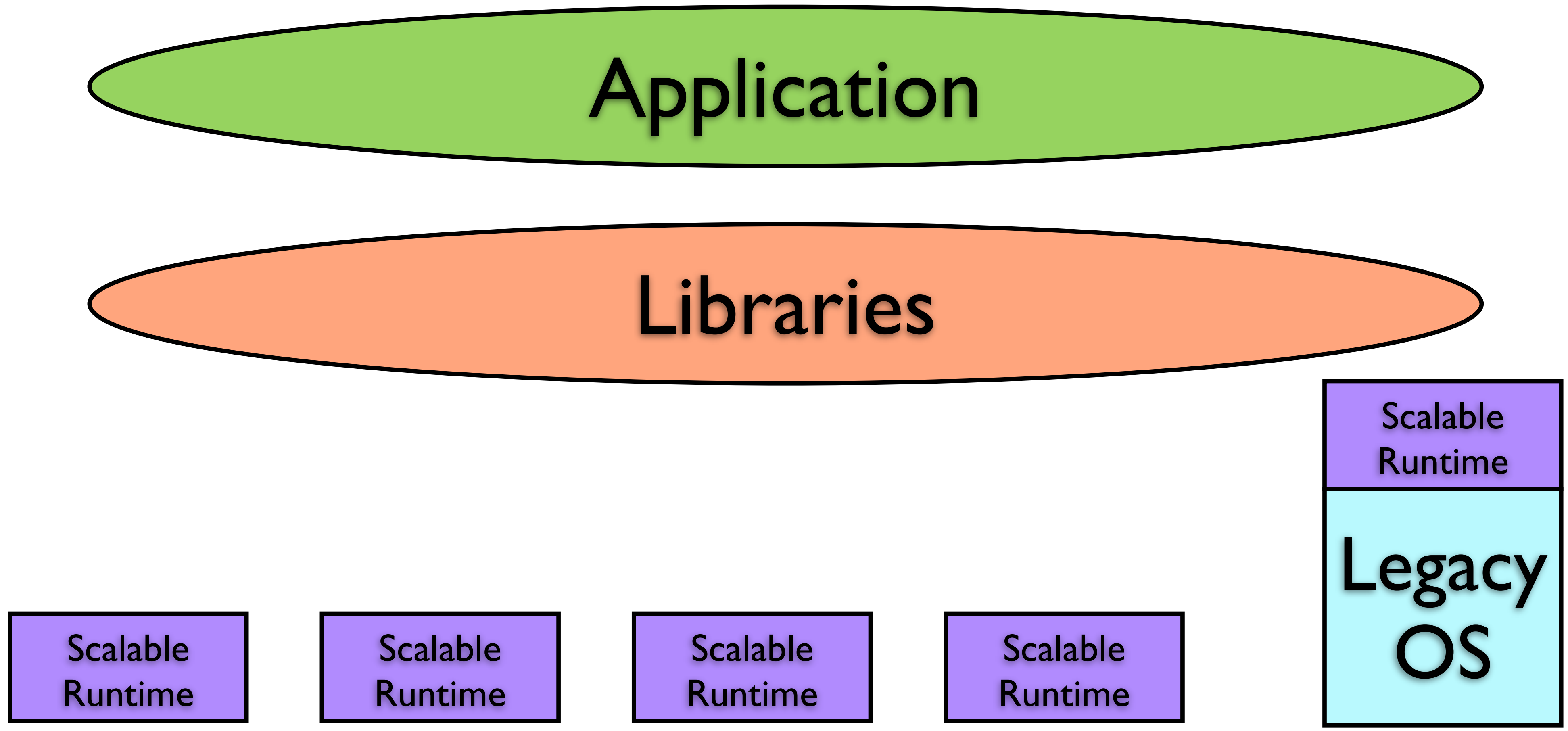
Research Background

- Datacenter scale systems are of increasing importance
  - Scale-out applications not just in HPC but also in cloud environments
- Increasing complexity
  - Heterogeneity
  - Failures/Elasticity

High-level Architecture

- Doing away with the kernel-userspace boundary
  - Software is constructed as libraries on top of a thin scalable runtime
- Low level primitives to aide in the construction of distributed software
- Allow incremental porting of legacy software

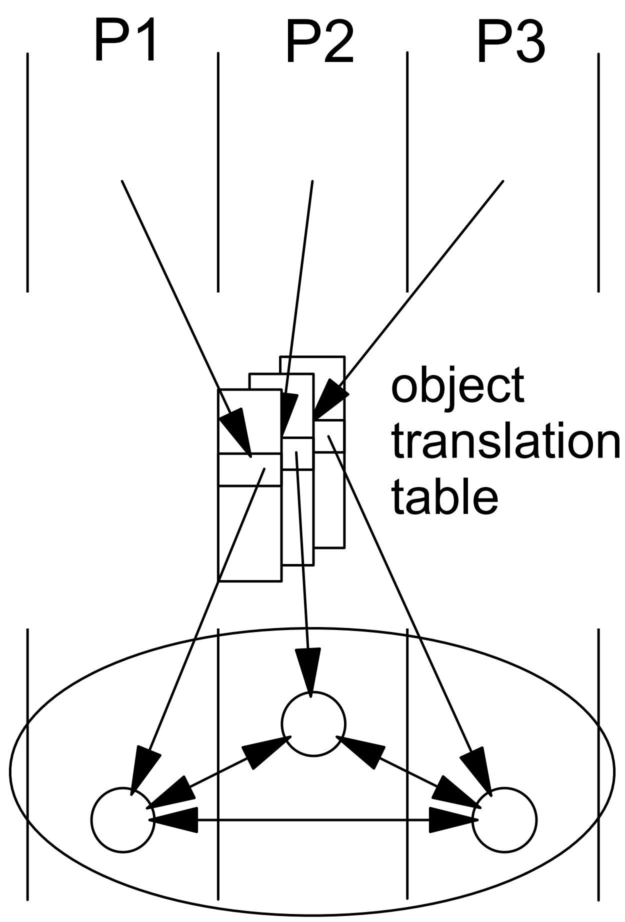
High-level Architecture



Component Level Decomposition

- Managing communication and locality is hard
- Inspired by work on Tornado/K42 - no communication paradigm is best
  - Not only message passing
  - Not only shared memory or RDMA
- Encapsulate communication by decomposing software as components

Elastic Building Blocks

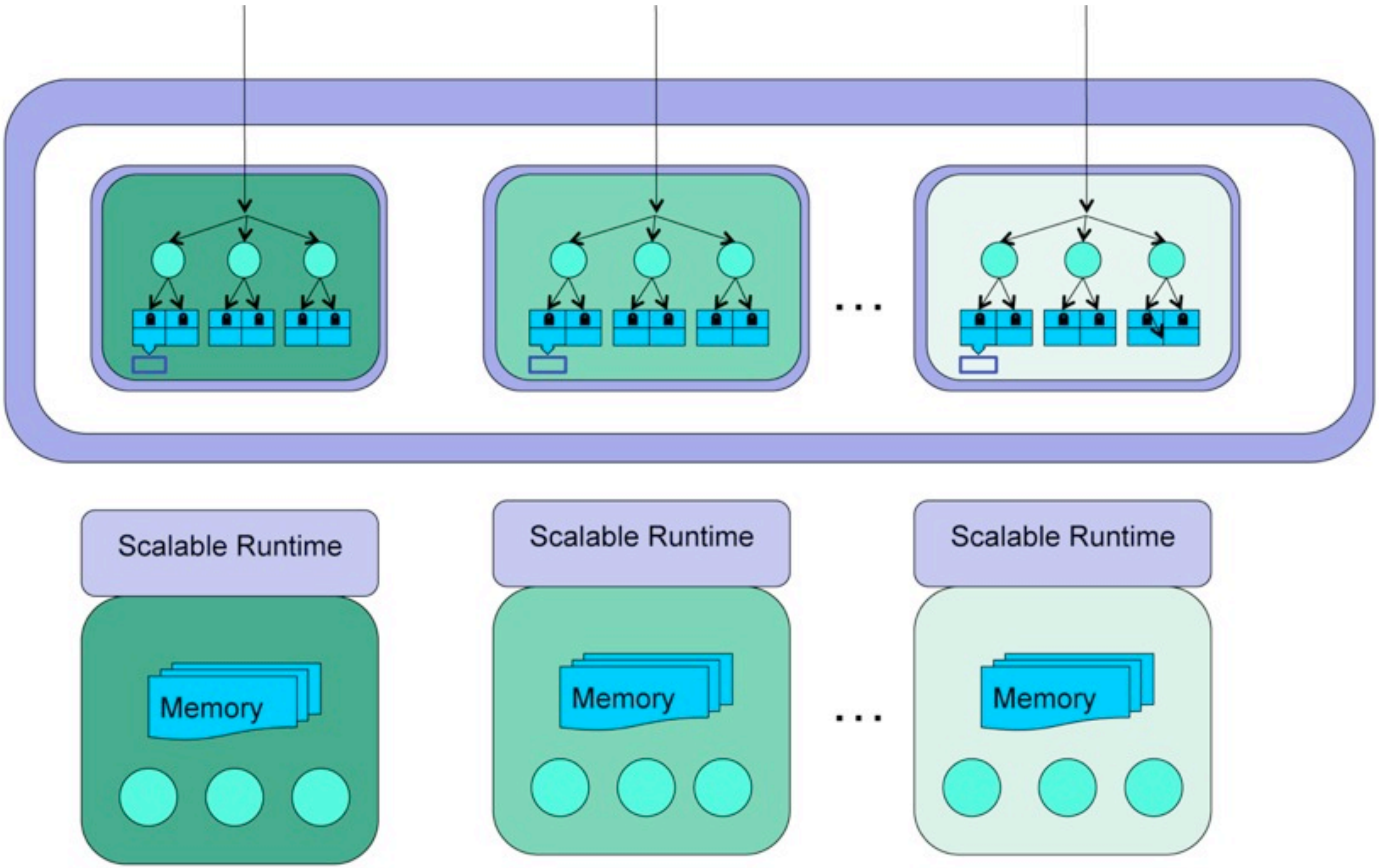


- Invocation of an object goes through a translation table
- Potentially different representatives of an object per processor

Elastic Building Blocks

- Objects respond to events
  - EbbCall
  - First time accessed on a processor
- Also hardware "events"
  - Tree packet arrived
  - Timer interrupt fired

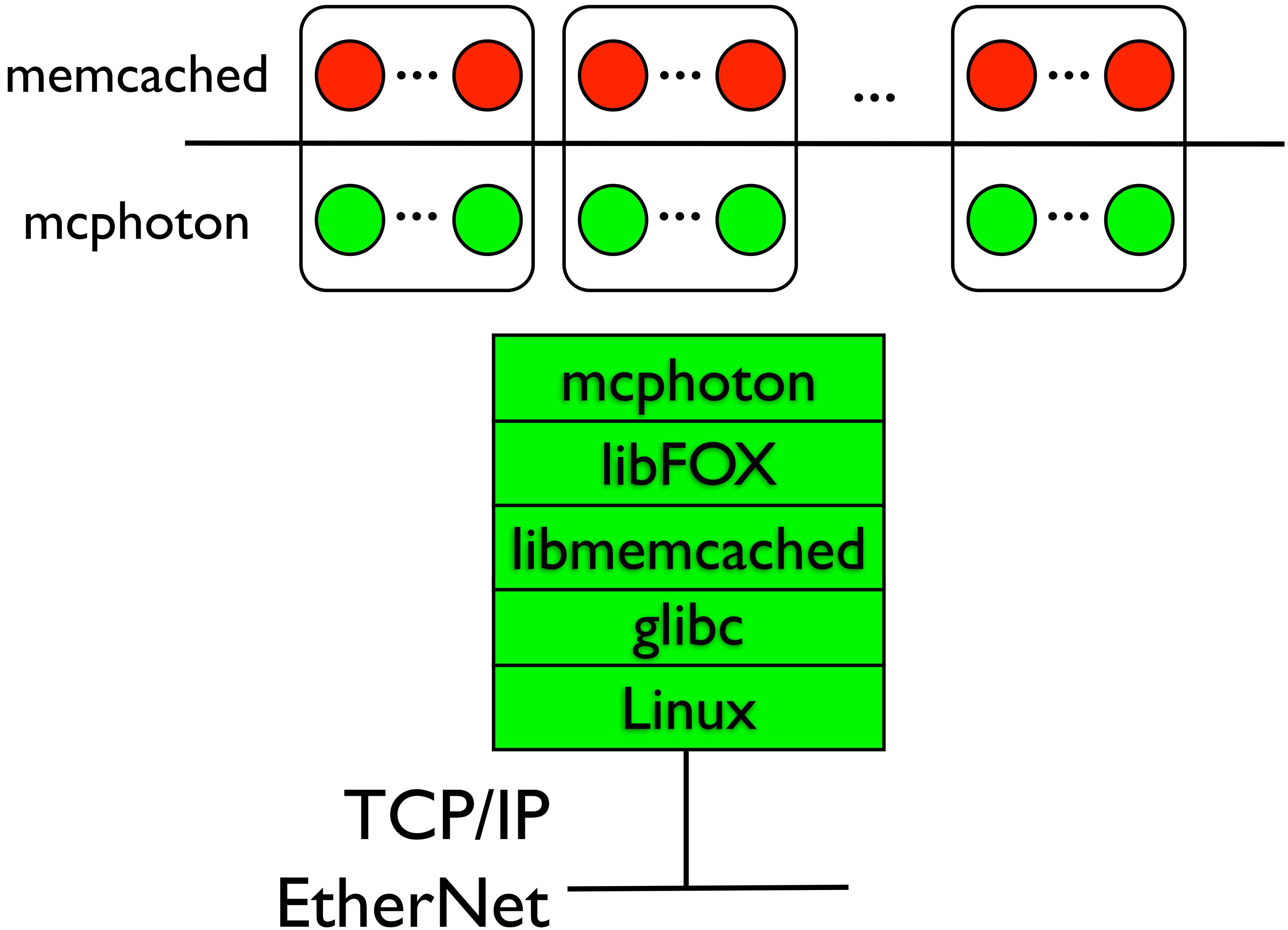
Component Level Decomposition



Our Goals

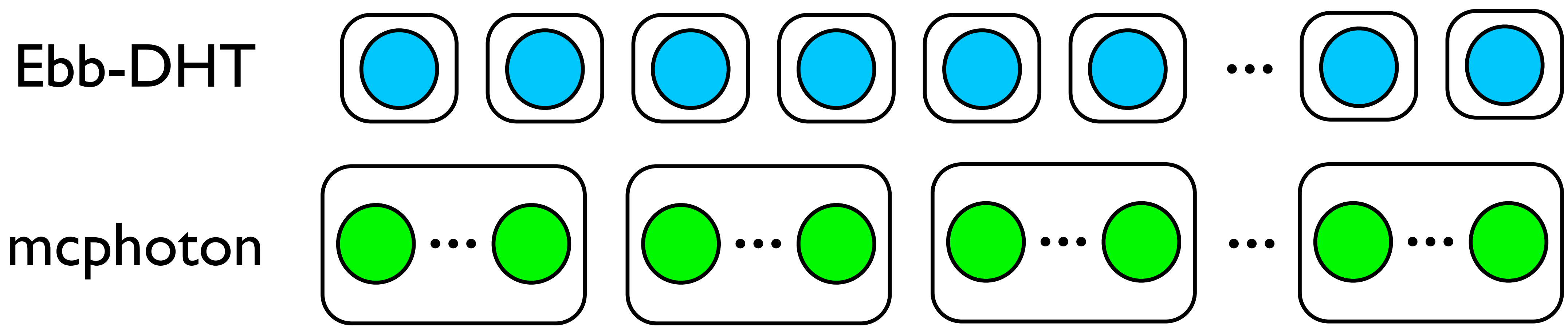
- Explore utility of system level primitives for Key-Value Stores (KVSs):
  - Can they help with faults / elasticity?
  - Can they encapsulate HW level optimization?
  - Can they help achieve HPC scale KVSs?

libFOX

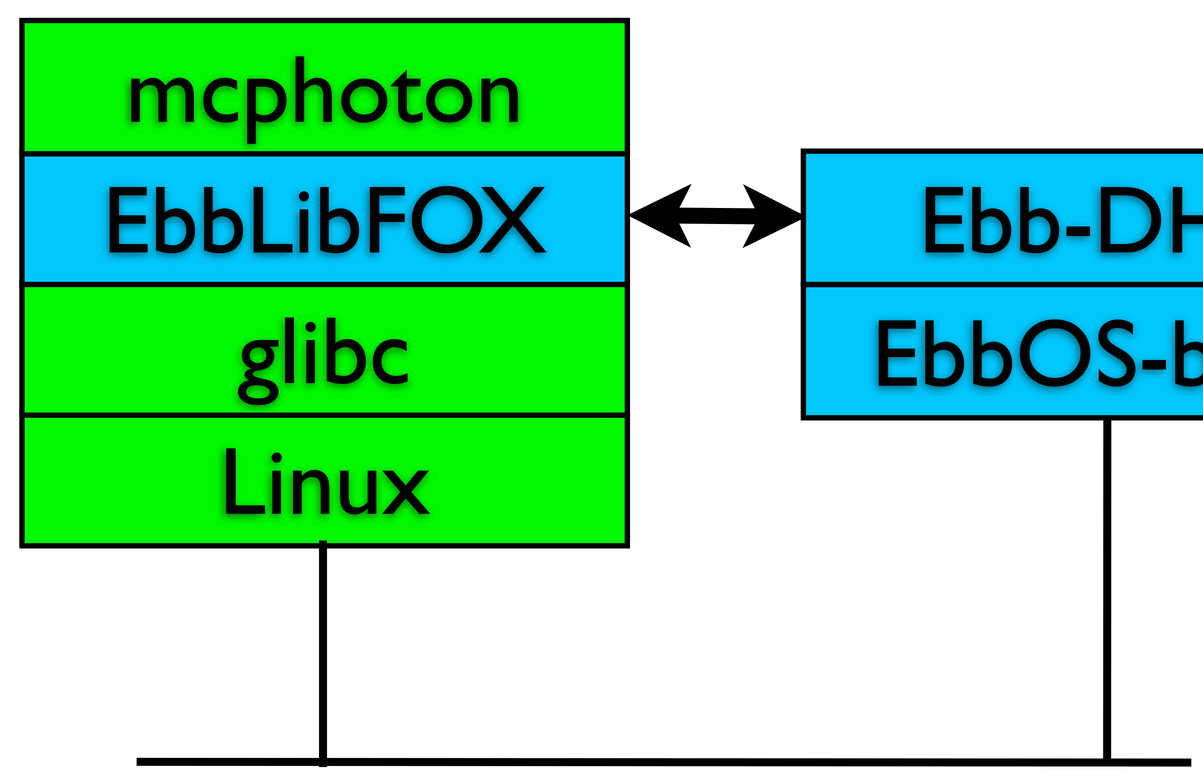


memcached
mcphoton
mcphoton
libFOX
libmemcached
glibc
Linux
TCP/IP
EtherNet

EbbLibFOX



Ebb-DHT
mcphoton
···

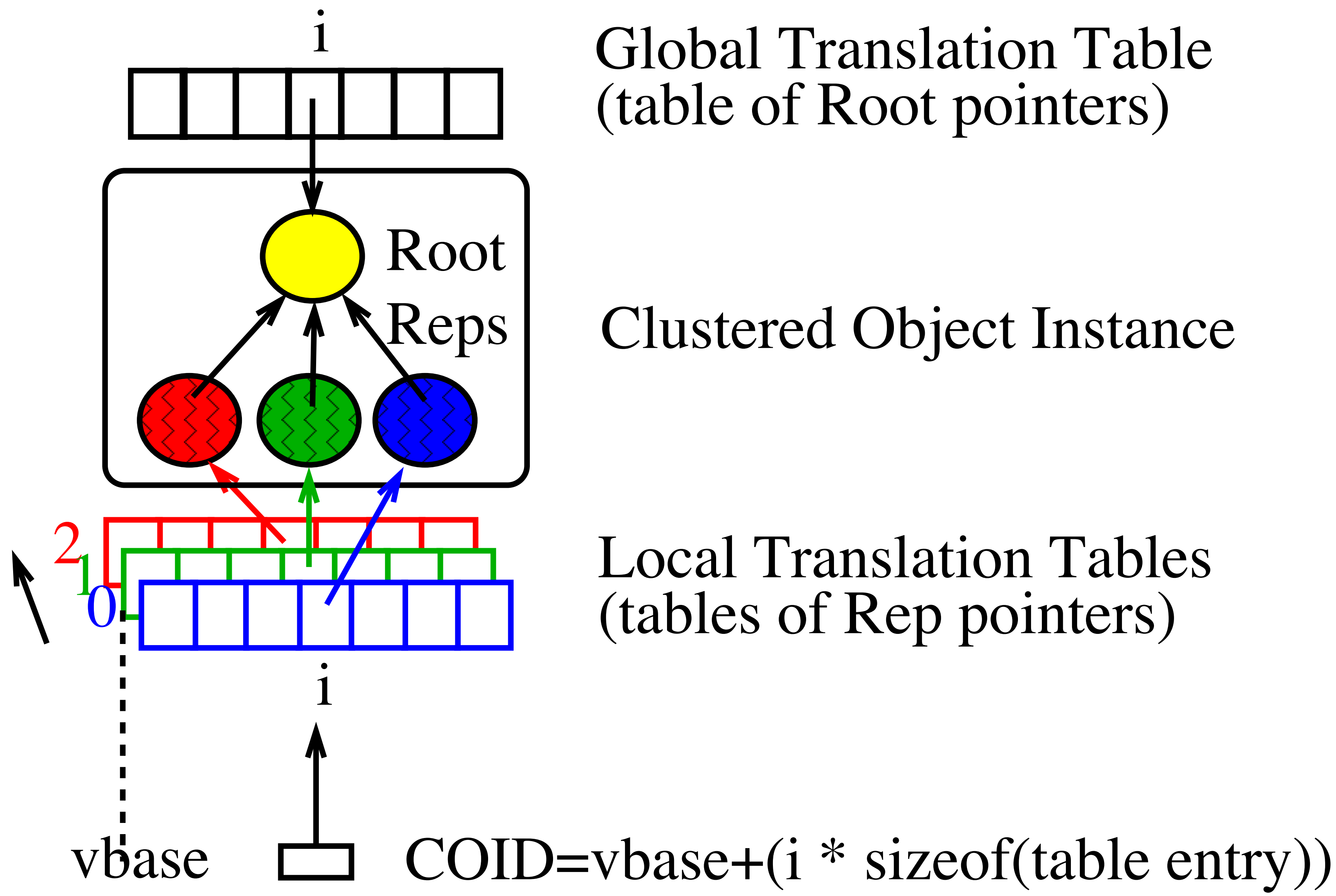


mcphoton
EbbLibFOX
glibc
Linux
Ebb-DHT
EbbOS-bare

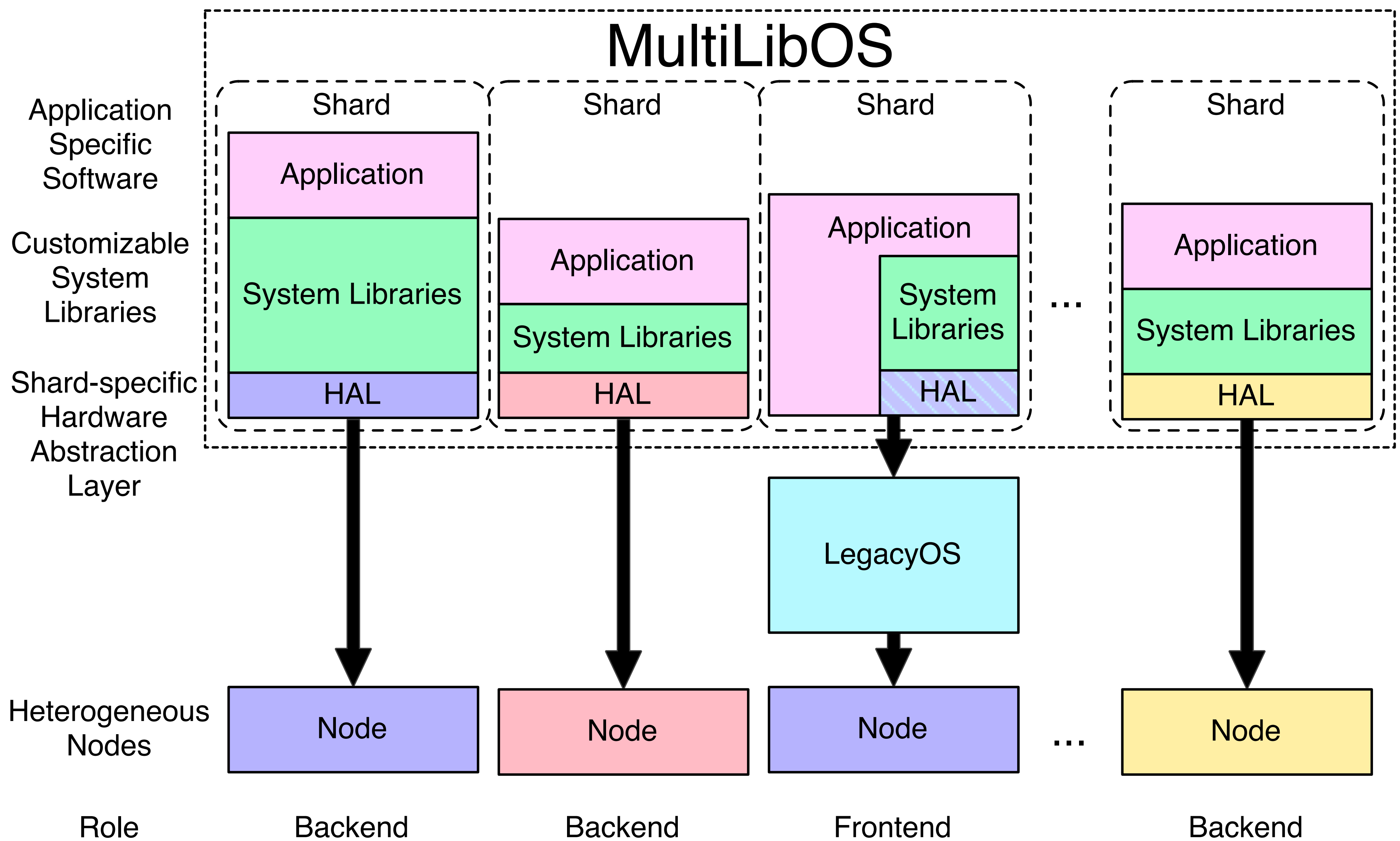
Status

- “Have” EbbOS runtimes for Linux, x86_64, PPC32, PPC64
- Gathering baseline measurements of event dispatch costs for a USENIX poster
- Developing an Ebb that implements a hash table

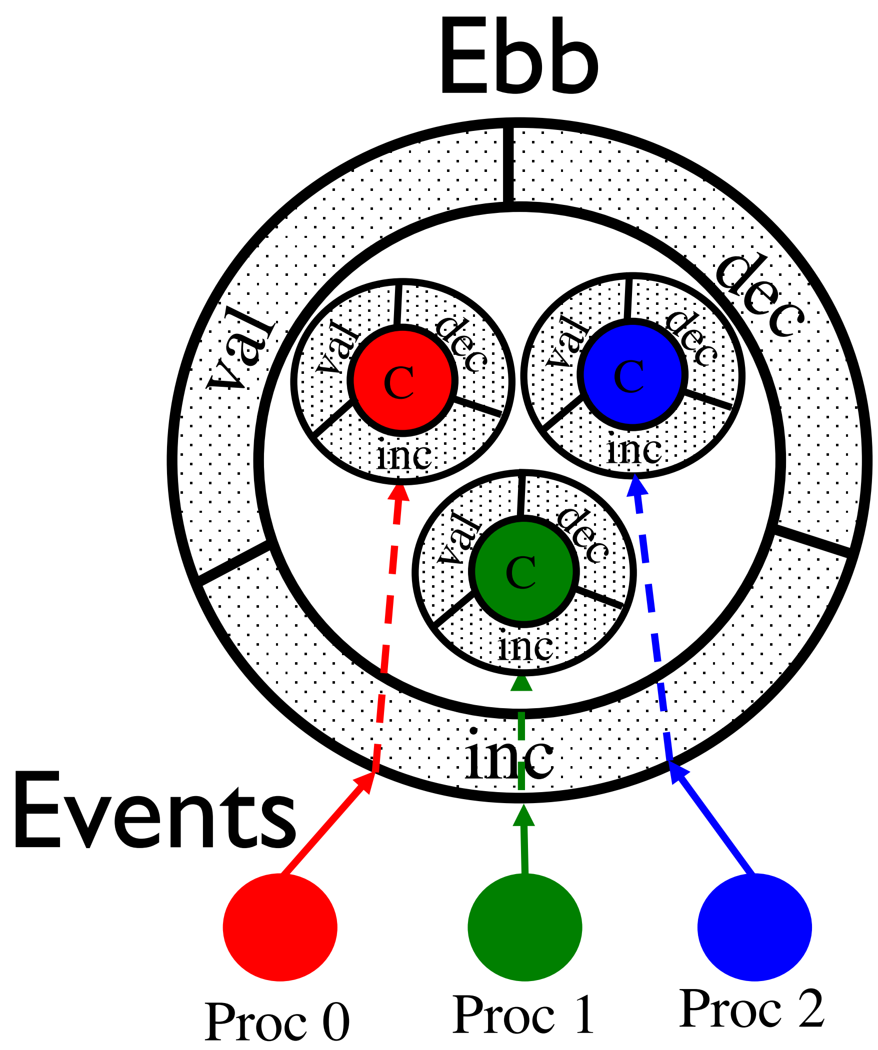
Backup

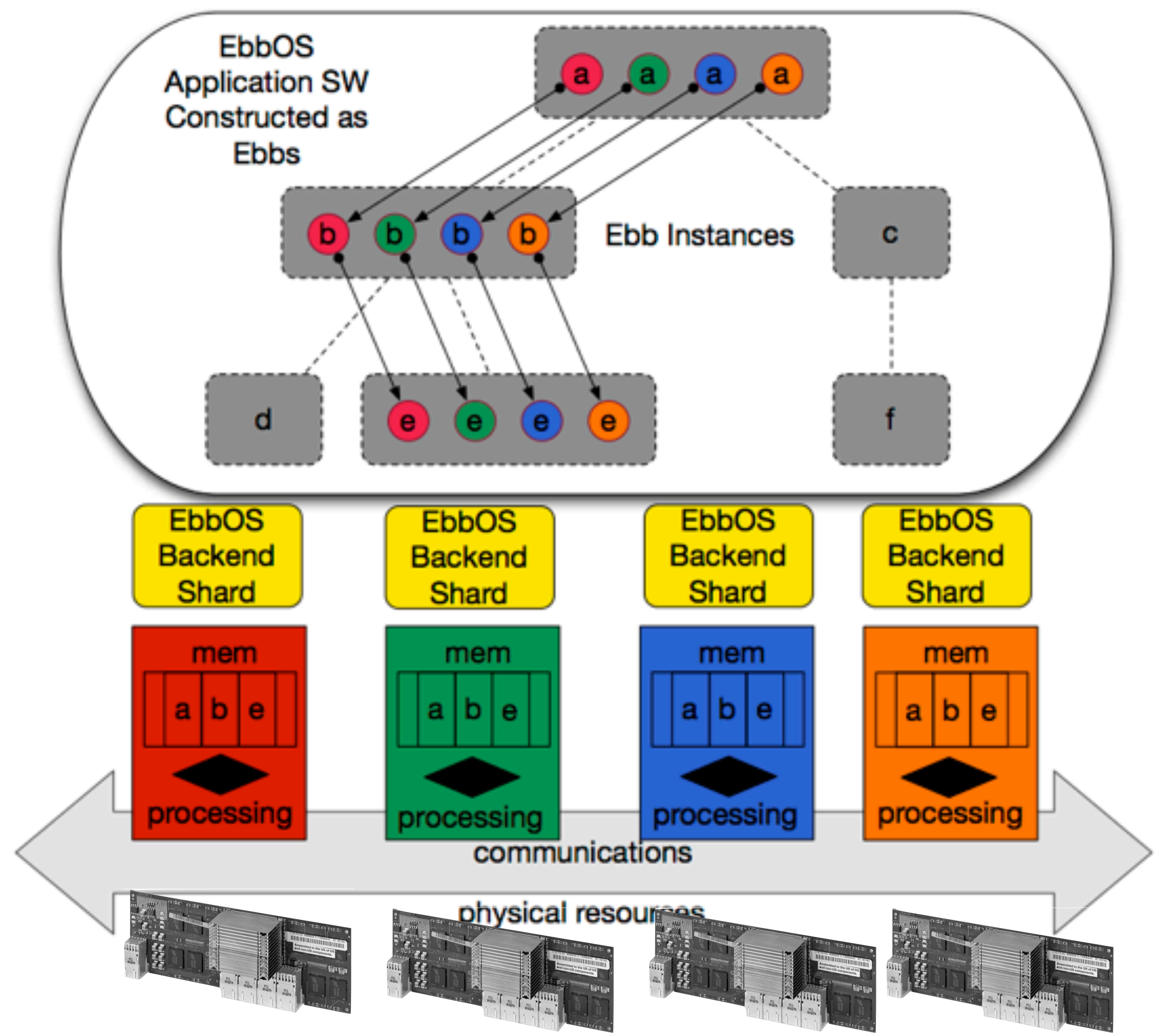

i
Global Translation Table
(table of Root pointers)
Root
Reps
Clustered Object Instance
2
1
0
Local Translation Tables
(tables of Rep pointers)
i
vbase
COID=vbase+(i * sizeof(table entry))

Background


MultiLibOS
Application Specific Software
Customizable System Libraries
Shard-specific Hardware Abstraction Layer
Heterogeneous Nodes
Role
Shard
Application
System Libraries
HAL
Node
Backend
Shard
Application
System Libraries
HAL
Node
Backend
Shard
Application
System Libraries
HAL
LegacyOS
Node
Frontend
...
Shard
Application
System Libraries
HAL
Node
Backend

Component Oriented Runtime EbbOS

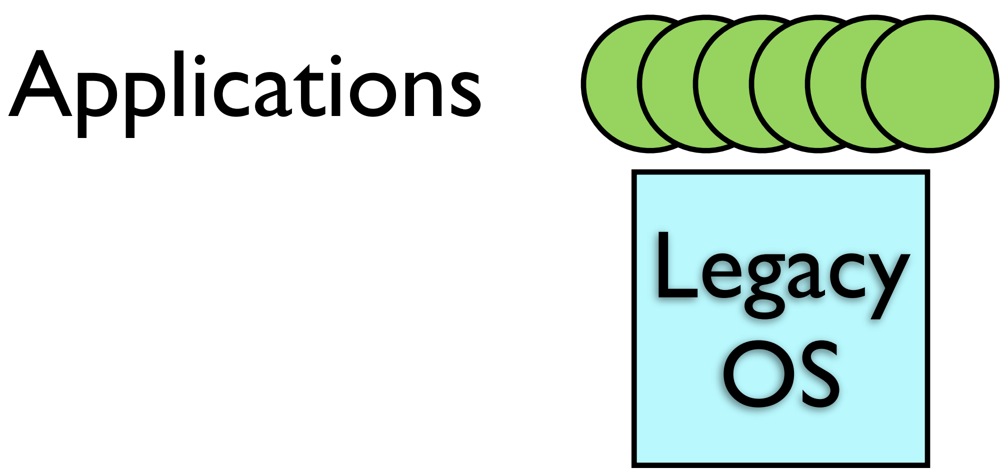




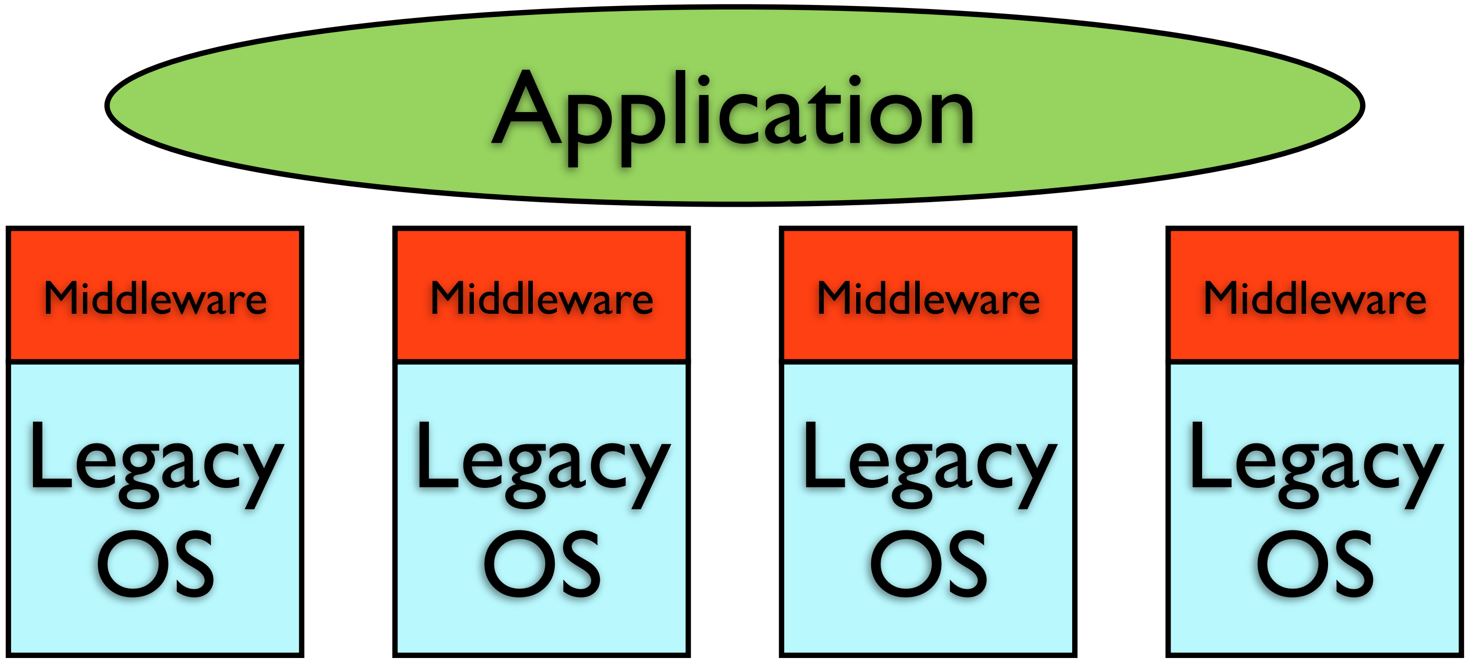
Research Background

- We have been making large scale applications by building collections of single node operating systems stitched together with middleware
- Applications are single user, multi node

High-level Architecture



High-level Architecture



High-level Architecture