E10 – High-Level Architecture

Julian Kunkel on behalf of the E10 contributors

International Supercomputing Conference

25. June 2014

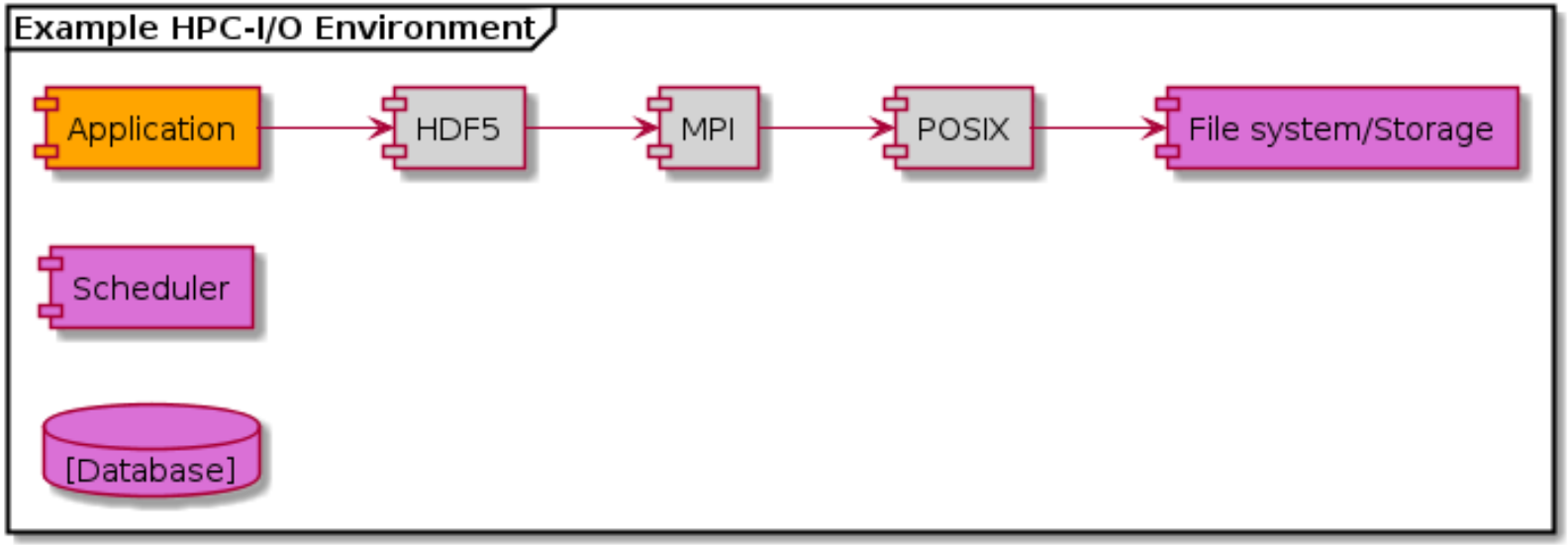
Outline

1 Overview

2 Components

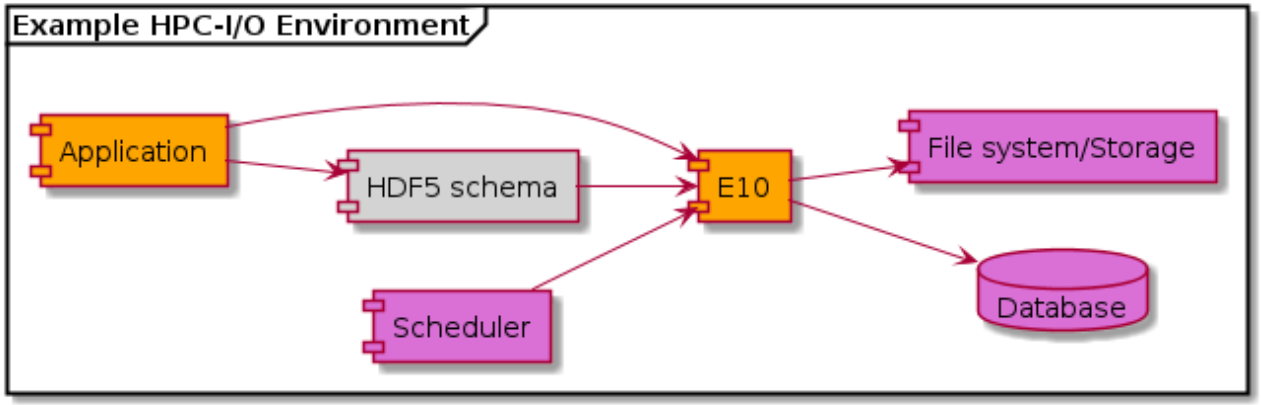
3 Summary

Existing HPC Environment and Stack



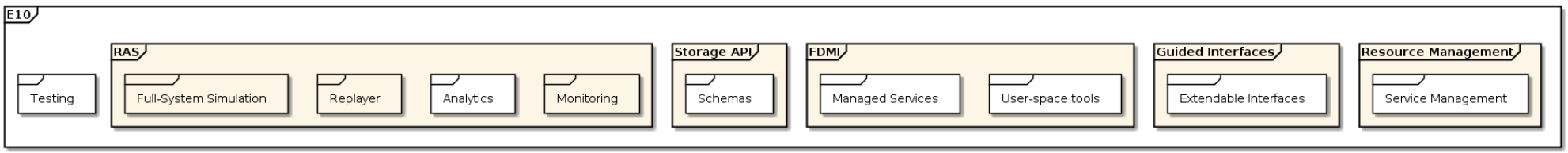
E10 Deployed System

- Middleware: deployed on top of file systems and databases
- Integrates with an existing scheduler
  - Data prefetching, staging of local data
- Application may use advanced features / interfaces directly



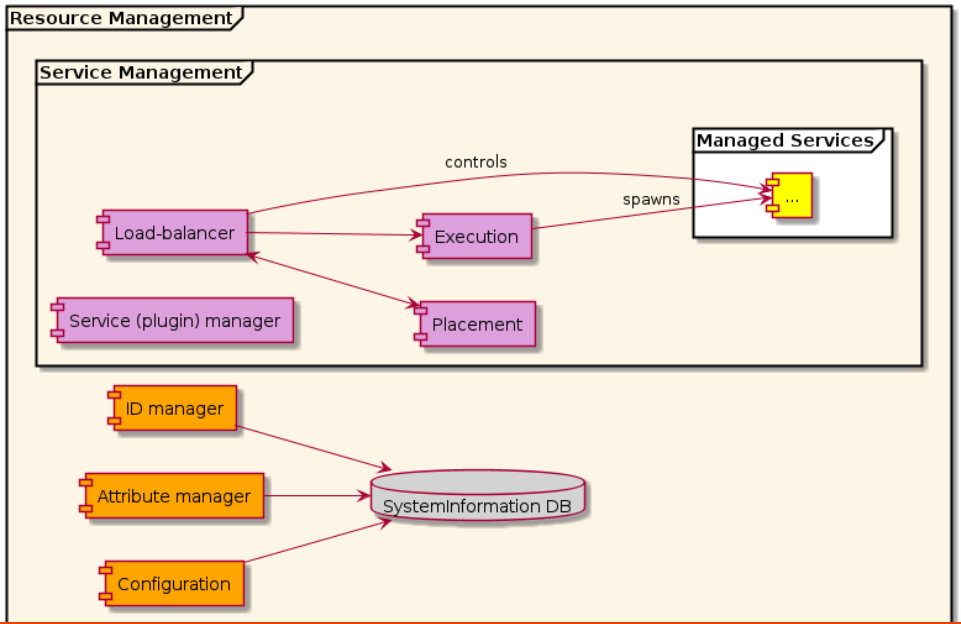
High-Level Components

- Resource mgmt: everything is a resource e.g. variable, hw
- Service mgmt: deploys compute snippets on resources
- FDMI: provide hooks for near-storage computation
- Guided interfaces: expose various ways to direct system
- Storage API: different views / interfaces to data, generic helpers
- Monitoring: observes system (and application) behavior
- Data analytics: analyze behavior and derive optimizations
- Simulation: What-if analysis to predict and extrapolate behavior
- Replayer: Recreate existing behavior (debugging etc).
- Testing: Approach to increase testability and maintainability



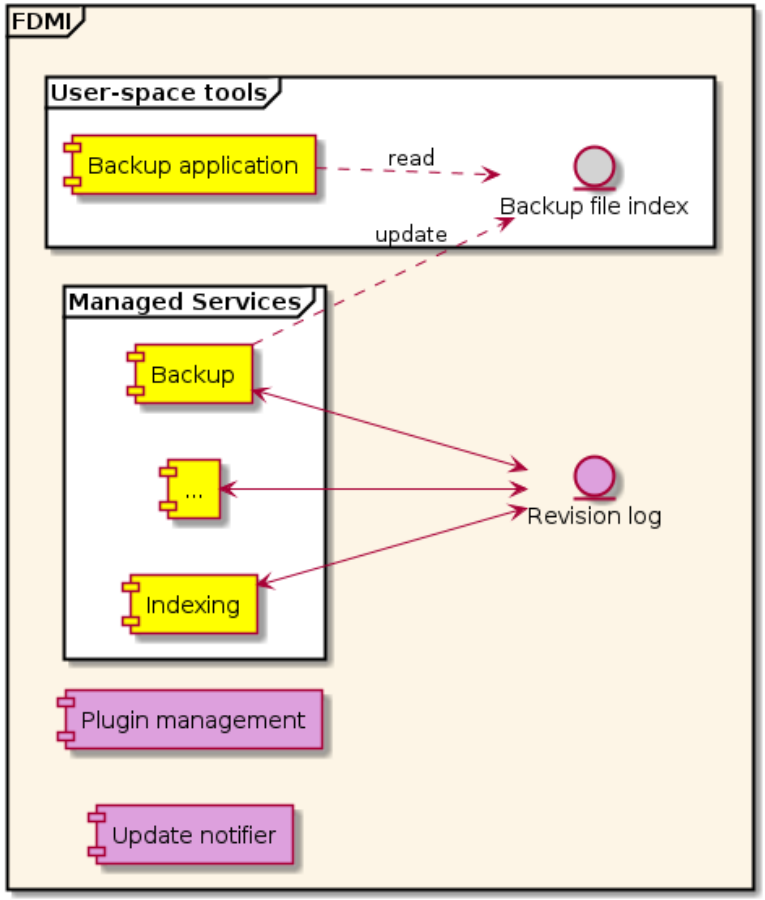
Resource Management

- Everything is a resource: Files, variables, hardware
- IDs are handled, some resources are local (e.g. file handle)
- Attributes can be set to resources
- Configuration of the system is stored
- Users could submit “snippet“ together with their computation
- Service management places these as “managed services“
  - According to preferences close to required source
  - Example: Data compression (in-line) and data post-processing
- Load-balancing required, takes information from monitoring



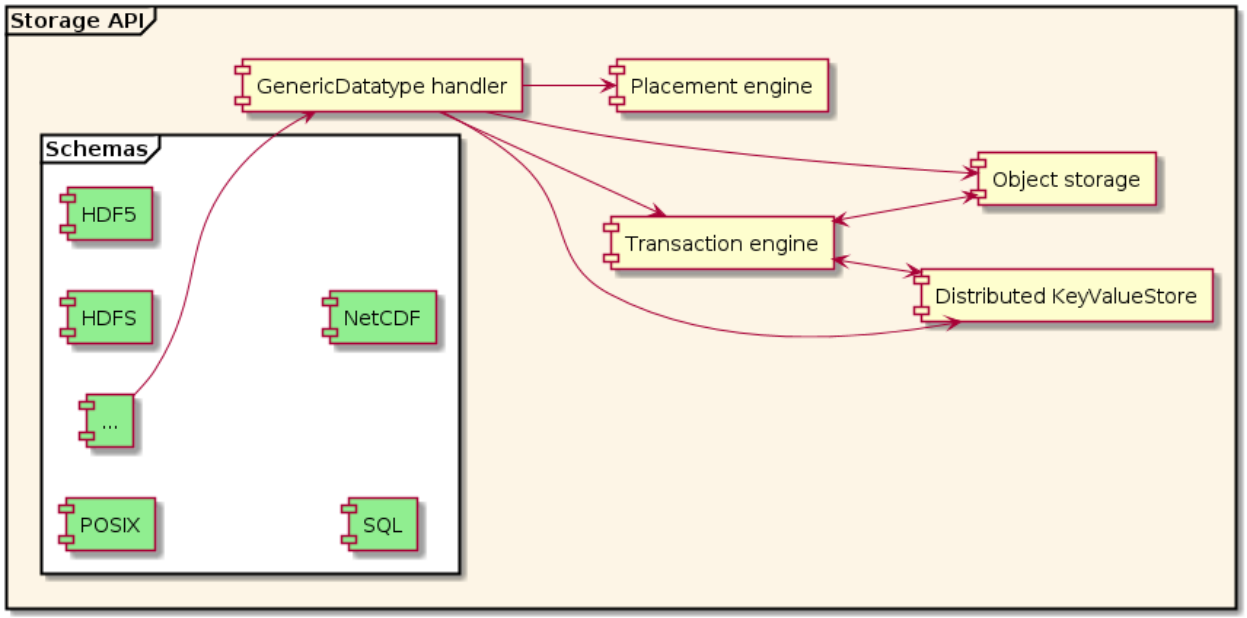
FDMI

- Eases deployment of data services
- Post-process / analyze data
- Backup, Indexing, etc.
- Managed service processes data
  - Creates e.g. an index for backup
- Triggered user-space tool can read these index files
  - Perform required (time-consuming) action
  - Reset index file atomically (if needed)
- Relies on Service management



Storage API

- Schema offers view to data
- Internally generic datatypes are understood
  - Allows to access the same data with different schemas
- Placement engine decides how to place data on available resources based on hints
- Transaction engine bundles access to KV-store and OS
  - Provides hooks for e.g. FDMI, to notify transaction completion
  - Triggers starts of managed services



Guided Interfaces

Guiding vs. automatism vs. technical hints

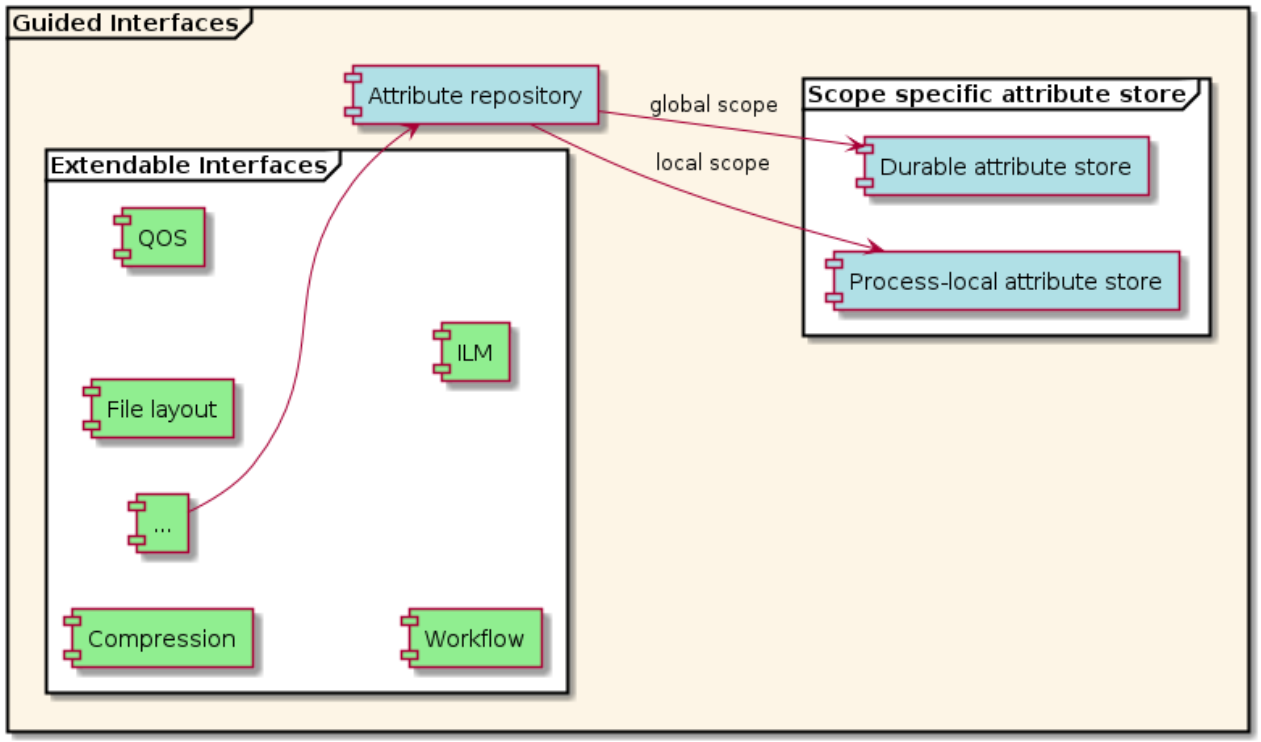
Users provide additional information to guide an intelligent system.
The I/O stack exploits this information.

Information which could be provided by users

- Semantics (consistency, value of data)
- Relations between data
- Lifecycle (especially usage)
- Data compression

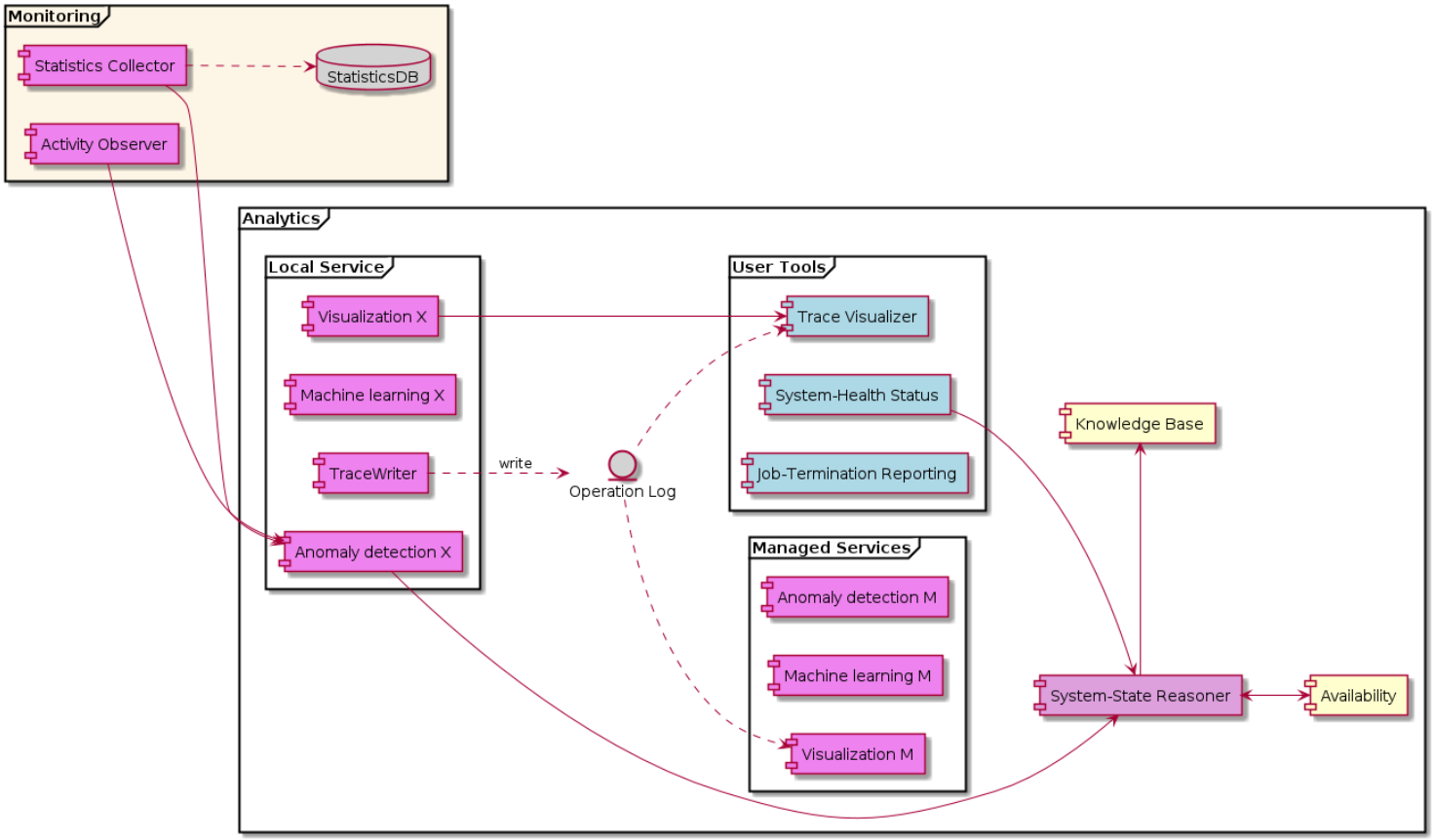
Guided Interfaces

- (Easily) extendable interface offers domain specific hints
  - Accuracy for lossy compression
  - Good file layout (technical hint!)
  - Life cycle incl. pre-fetching and staging
- Attribute repository sets generic attributes
- Any layer can query attributes on resources



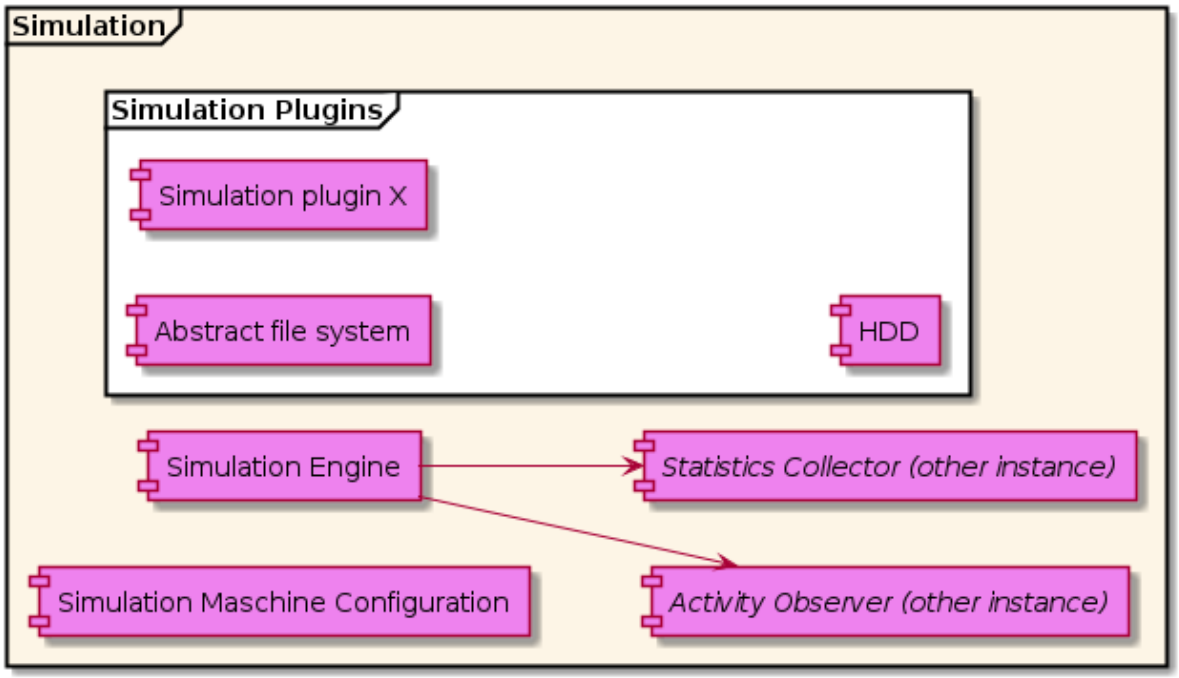
Monitoring and Analytics

- Record system activity and hardware utilization
- Uses managed services to crunch information
- Start intelligent recording of activity if in an abnormal state



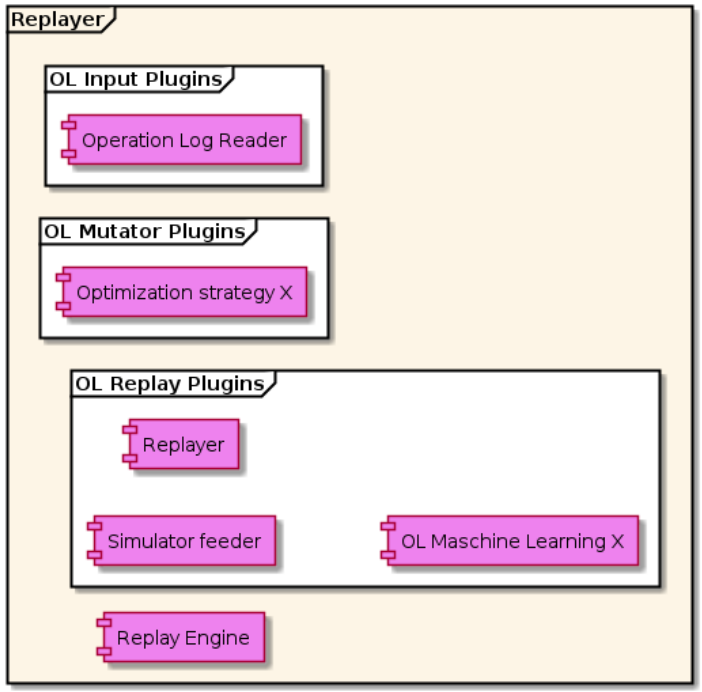
Simulation

- Experiment with arbitrary configurations and systems
- We try to build component such that they can be virtualized
- Alternative – higher-level – component replacements possible
- Report to monitoring (but probably different instance)



Replayer

- Replay occurred events (operation log)
- Eases debugging
- Alter operation log to test different scenarios



Testing

The modular approach simplifies testing:

- Low-overhead dummy for many E10 Components
- Vanilla activity injector
- Layer specific operation-Log replay and simulation
- Layer specific performance monitoring

Summary

- A high-level draft of the E10 architecture is available
- E10 will become a modular allround storage system
- E10 will overcome limitations of the current architecture
  - Data-type aware storage
  - Multiple “views” to the same data (BigData)
  - Guided interfaces instead of technical hints
  - Data “format” handled by storage system
  - Multi-tiering support
  - Intelligent monitoring
  - Feedback to optimization and “what-if” analysis
  - Integrates active storage concept
  - Post-processing handled by file system
- Incremental deployement on top of existing infrastructure
- Huge effort to refine design
- **https://github.com/Exascale10/design**