Concrete Architecture

Team Space Jam

Overview

- Architecture of YARN
- Architectural styles and design patterns
- Compare concrete architecture against conceptual architecture via reflexion analysis
- Discrepancies between both architectures
- Diagrams of the concrete architecture
- Noteworthy aspects of the architecture and its subsystems using sequence diagrams (or state diagrams)

Master-Slave Architecture

• An architecture where a single device has unidirectional control over one or several other devices

•The Master-slave pattern is commonly used in instances of:

-process control

-embedded systems

-large-scale parallel computations

-fault-tolerant systems

Components

- The master: Is most often responsible for communication, coordination, computation and most importantly it controls the slave components.
- The slaves: Dedicated specific actions to perform for the master.

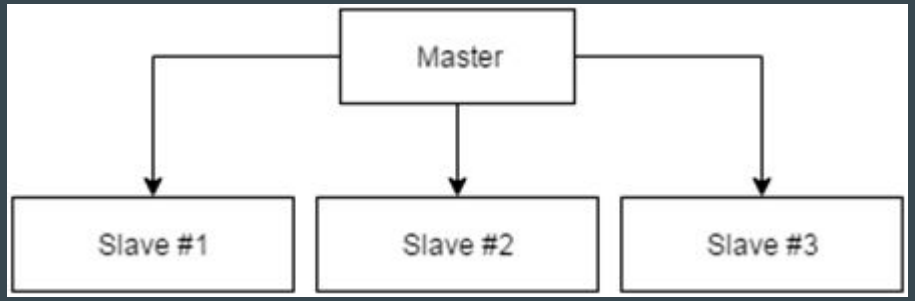


Figure 1.1 Master-Slave diagram

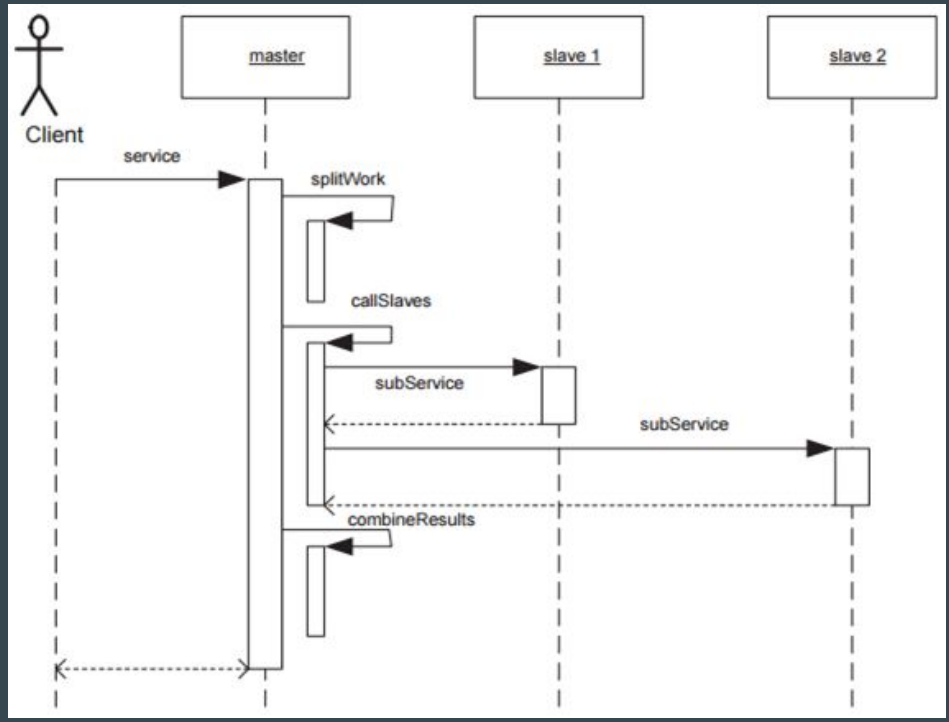


Figure 1.2 Master-Slave sequence diagram

Master-Slave Examples

•Parallel computing: The master component takes a complicated task and divides it into several identical subtasks for the slaves. An example of this would be the computation of a matrix. Each row is computed by a different slave.

Advantages and Disadvantages

- Advantages:
  - Fault tolerance
  - Parallel computation
  - Scaleability
  - Robustness
- Disadvantages:
  - Slaves are isolated
  - Master-Slave latency can be an issue
  - Only applicable to decomposable problems

Master-Slave within YARN

Figure 2.1 The master

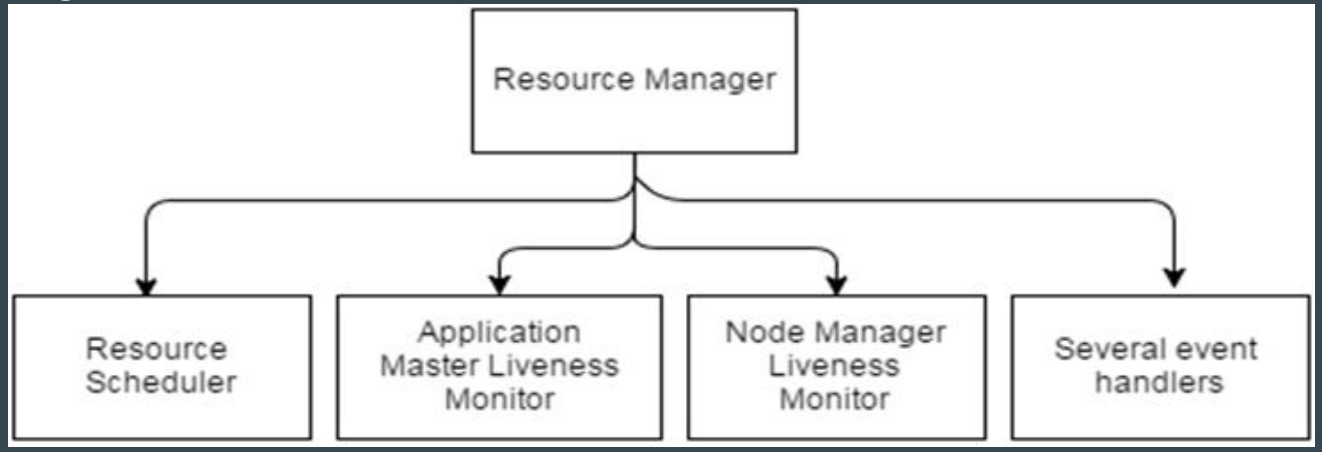
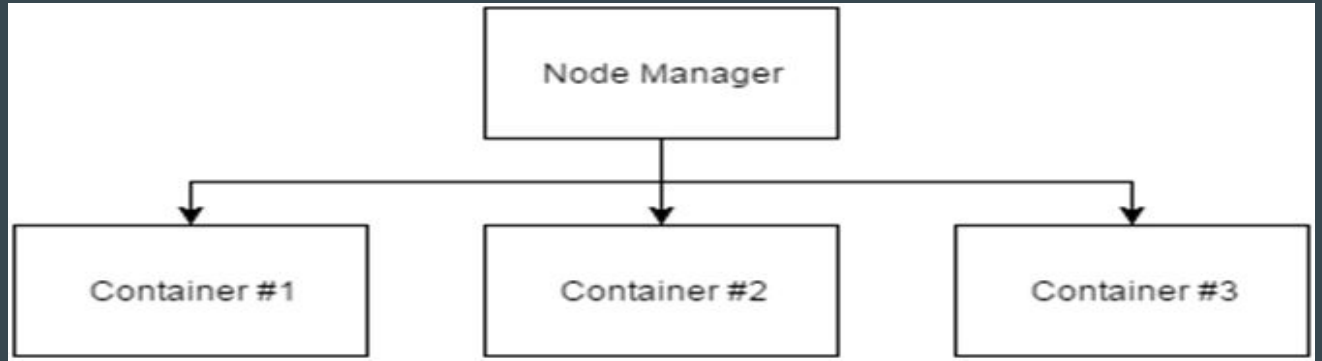


Figure 2.2 The slave



Design Concepts

- Application Submission Client submits an application to the YARN ResourceManager
    - YarnClient Object
- First container of the application contains ApplicationMaster; submits application
- During execution, ApplicationMaster communicates NodeManagers through NMClientASync; events handled by NMClientASync.CallbackHandler
    - Starts, stop, status update
- ApplicationMaster reports execution progress to ResourceManager by handling getProgress() method of AMRM Client ASync.CallbackHandler

Interfaces

- Client ←→ ResourceManager
  - Utilization of YarnClient Objects
- ApplicationMaster ←→ ResourceManager
  - Utilizing AMRMClientASync Objects (AMRMClientASync.CallbackHandler)
- ApplicationMaster ←→ NodeManager
  - Launch containers and communicate with NodeManagers; handling container events

Object Oriented Design Patterns (OODP)

- YARN's source code extensively uses Object Oriented Design Patterns
  - interfaces, information hiding, polymorphism, intermediary objects
- Structural Patterns
  - Decorator Pattern
- Behavioural Patterns
  - Iterator Pattern
  - Observer Pattern
- Coding to Interfaces, not Implementation

Conceptual vs. Concrete

- Conceptual
  - Ideas are thought up of here during documentation
  - Follow as close to the System Requirement Specification as possible
  - “What do we want our system to fulfill?”
- Concrete
  - The physical work of the system in code
  - Users submit feedback, updates made. Cycle
  - “What does the system do for you?”
    - Alternatively “What do we need for our system now that we overlooked?”

Conceptual Model

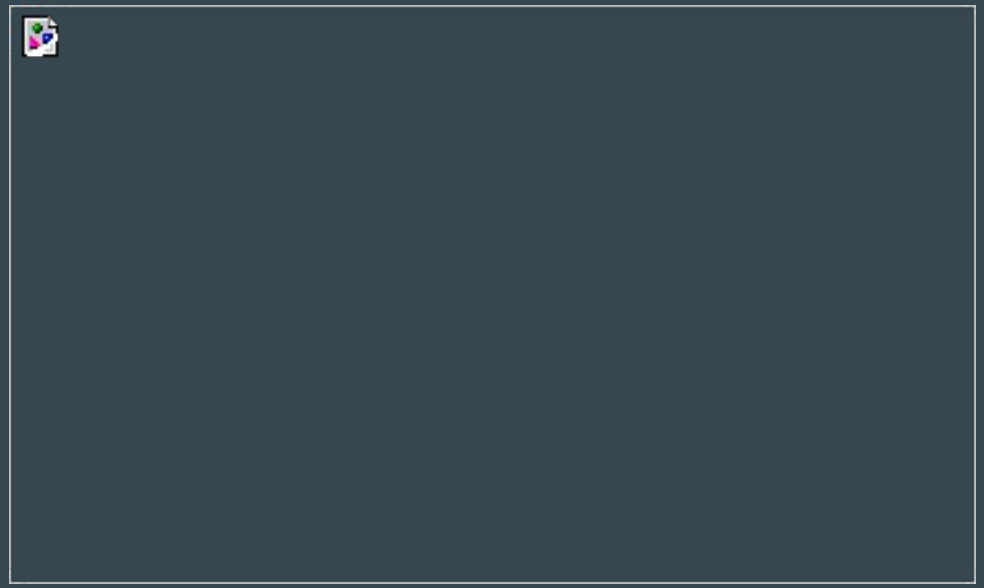


Figure 3.1 - Conceptual Model

Simplified Conceptual Model uncovered from Assignment 1 - Conceptual Architecture

Reflexion Model

Legend

Dashed lines indicate divergence from Conceptual Architecture

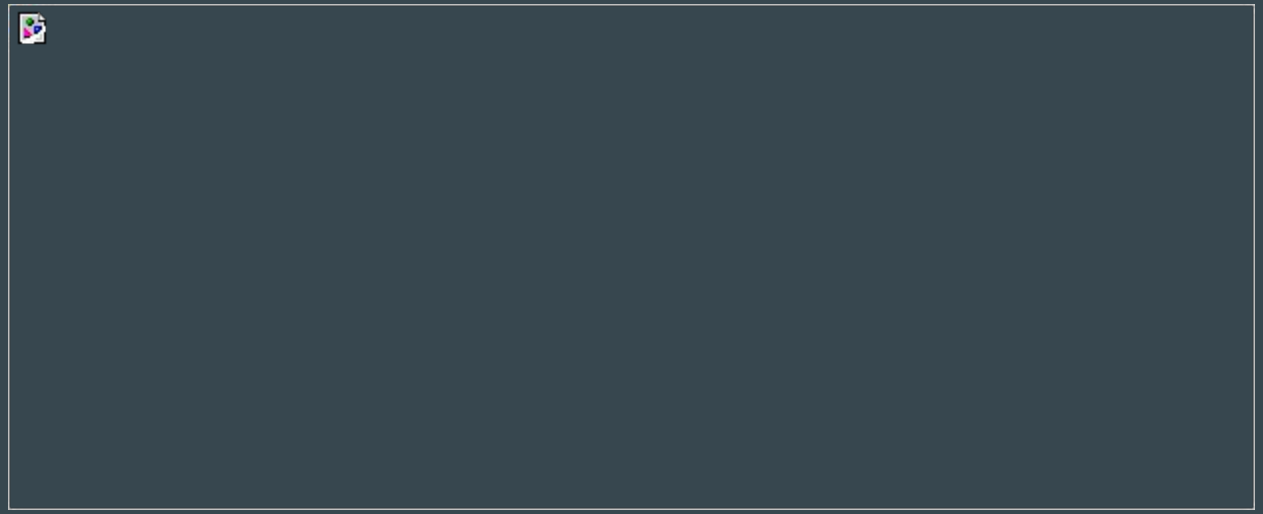


Figure 3.2 - Reflexion Model

Discrepancies between Architectures

- Two-way dependency between MapReduce and YARN (as opposed to the one-way dependency that we believed)
  - MapReduce handed all the data, what does it do after it's done with it?
  - Given back to YARN for use with hadoop-yarn-registry (next slide)
- Client is not actually part of the system
  - No object to directly represent "what a client does"
- Common-project
  - Contains "artifacts" of all classes used in this system
  - Can be thought of as a literal library

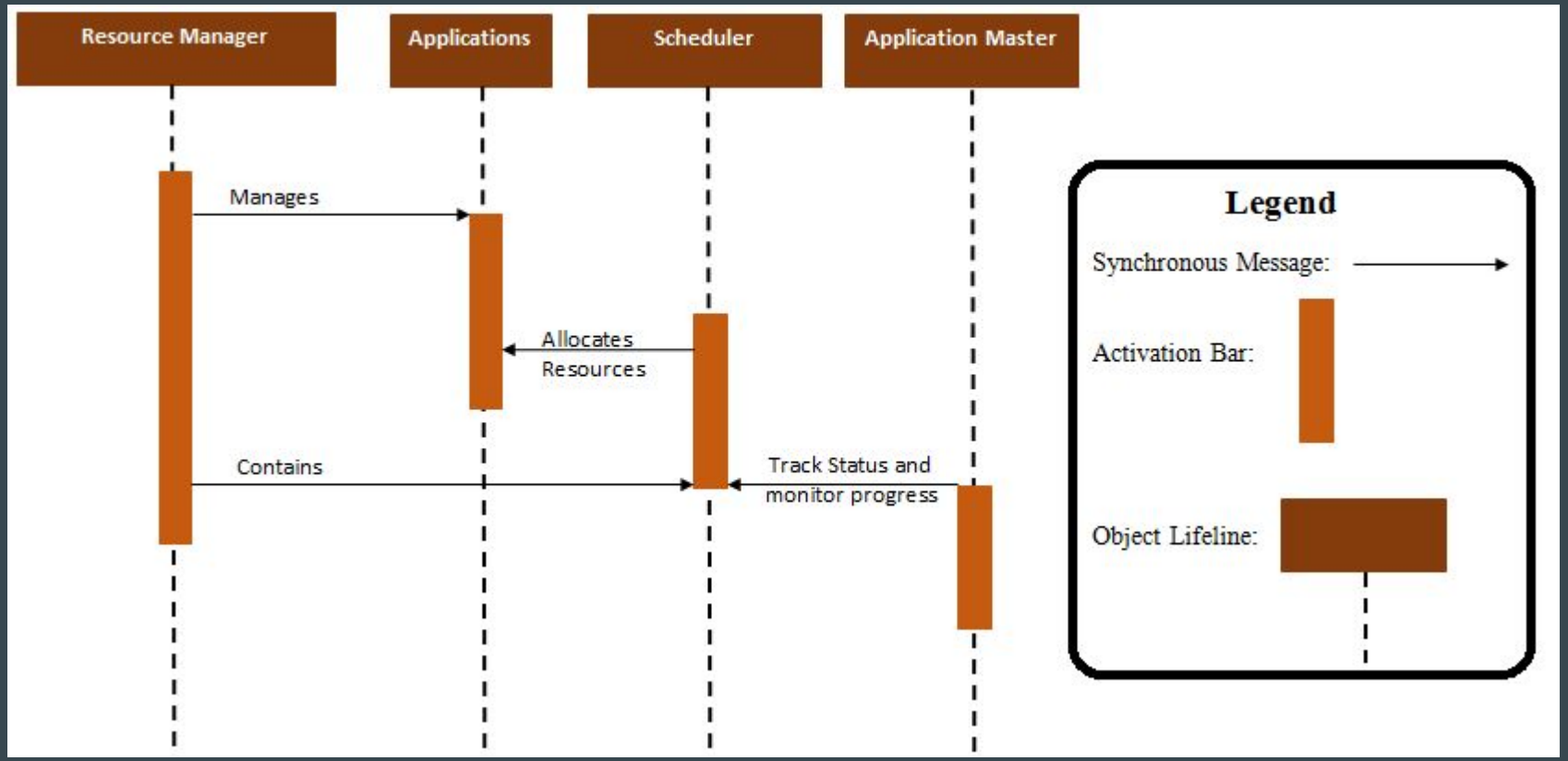
The New Subclasses

- hadoop-yarn-client
  - Generates reports, modifies queues (killing, moving), create reservations
  - User-friendly view of YARN's app processing
- hadoop-yarn-registry
  - Clients can now talk to YARN services
  - Services get registered here, marked with index IDs
  - IDs for use with yarn-client
- hadoop-common-project
  - Core libraries for use by YARN

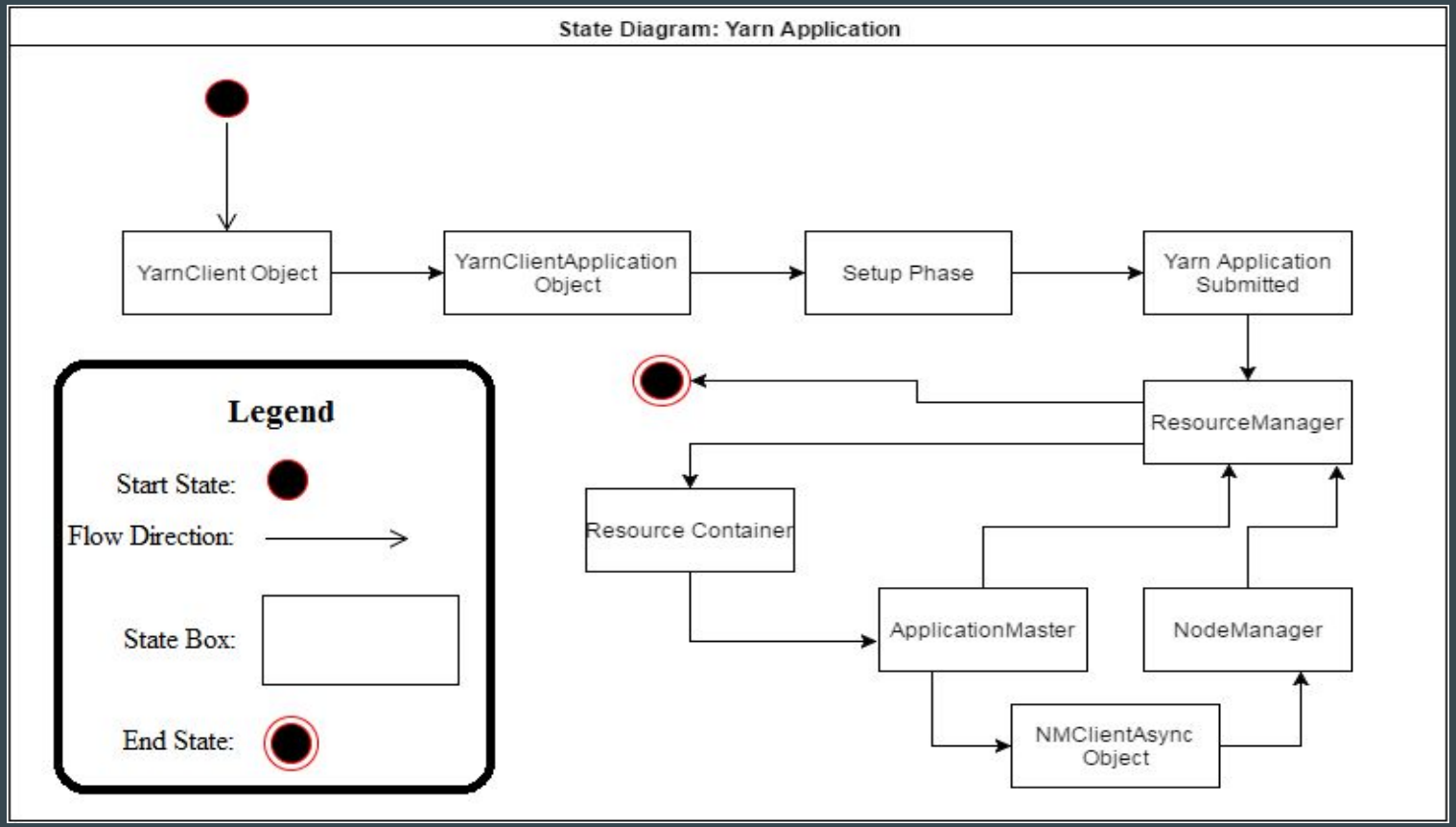
Noteworthy Aspects of Yarn

- Yarn subsystem is broken down into two major responsibilities:
  - JobTracker
  - Job Scheduler/Monitor
- These two sections are then separated into two background processes:
  - ResourceManager
  - ApplicationMaster

Noteworthy Aspects of Yarn



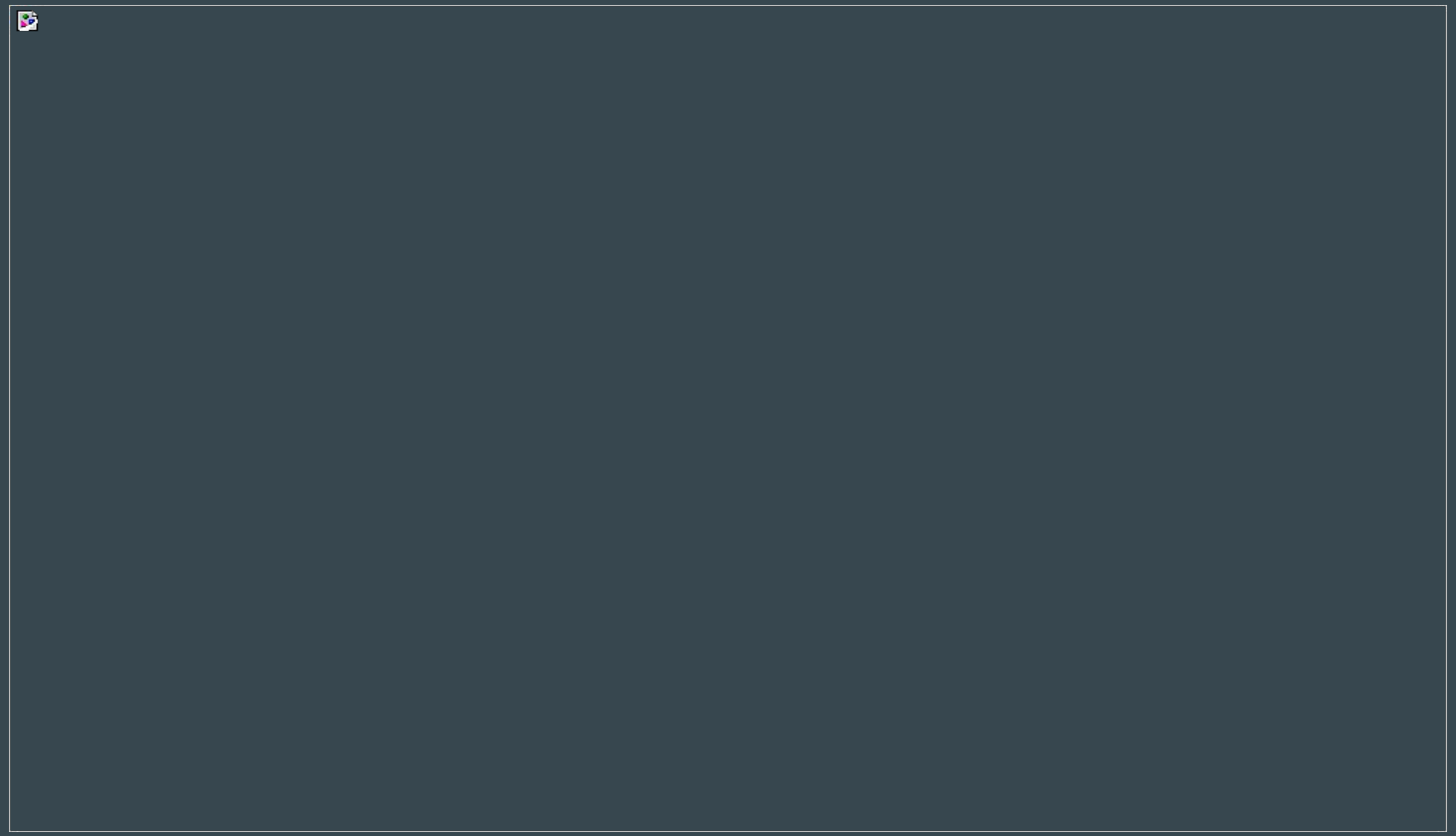
Yarn Application State Diagram



Main Subsystems of YARN

- YARN Server
- YARN Client
- YARN API
- YARN Common
- YARN Registry
- YARN Applications

Yarn High-Level View



Dependencies

- All subsystems are dependent on API
- The server, client, and application subsystems use methods from the YARN Common subsystems core libraries
- The client must submit applications to the server and vice versa
- The application subsystem depends on client to specify application context
- The applications are running on the server

Limitations of Reported Findings

- Unable to to fully represent all important dependencies of the concrete YARN architecture due to the large number of the underlying dependencies between subsystems and classes

Lessons Learned

- Can not present all dependencies
- Concrete architecture has many more dependencies than the conceptual architecture
- Conceptual architecture can not always be directly translated into the Concrete architecture
- With many dependencies: it's hard to pinpoint the most important ones
  - Even a simple call to index another object is enough to create one
  - Documentation is by far the best means of finding these dependencies
- Famous phrase "High cohesion - Low coupling" makes a great point here
  - A system with many dependencies is vulnerable to an error once one of the dependant systems malfunctions.
  - Error handling becomes much simpler without having to trace through repeated calls