# Apache Flink

## Discrepancy Analysis

Group Small

# In Today's Presentation

- Concrete VS Conceptual Architecture
- Analysis Process
- Reflexion Analysis
- Absences and Divergences
- Rationale for Differences
- Use Case
- Limitations
- Lessons Learned
- References

# Concrete VS Conceptual Architecture

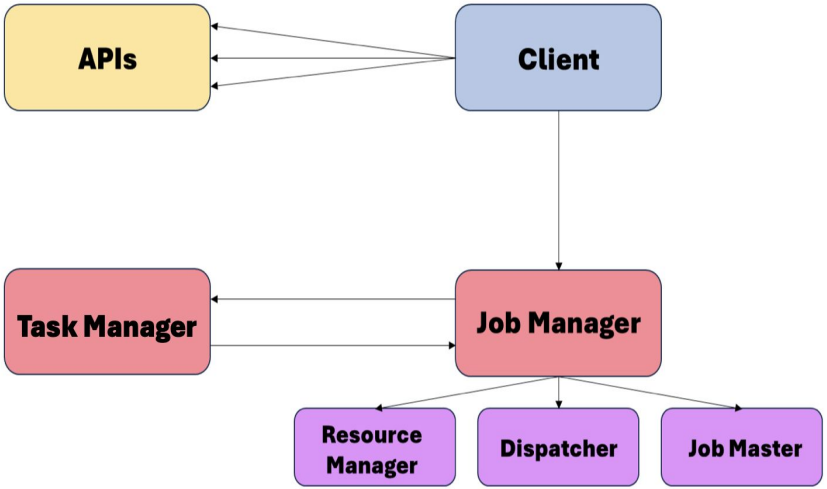


Figure 1: Conceptual architecture of Apache Flink

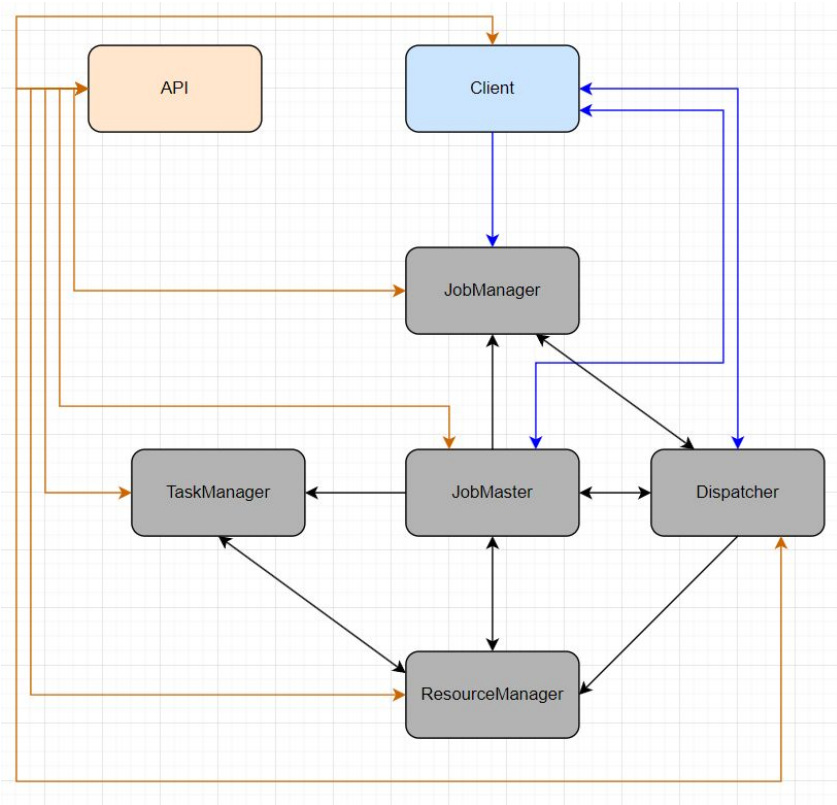


Figure 2: Concrete architecture of Apache Flink

# Analysis Process

- UnderstandDependencyFile
- Find responsible classes
- Comments
- JIRA
- GitHub

Alternative Processes

- Comments only
- JIRA only

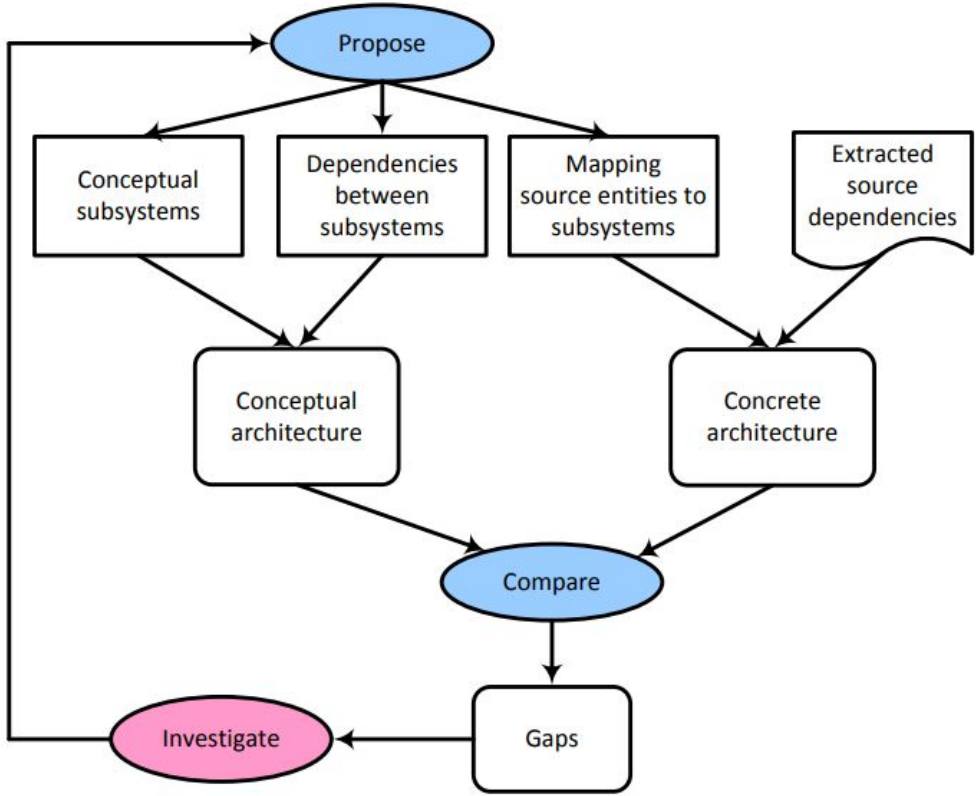


Figure 3: Reflexion Framework

# Reflexion Analysis

Divergence(- - - - -)

- Dependencies present in the concrete architecture, but missing in the conceptual architecture.

Absence(• • • • • • •)

- Dependencies present in the conceptual architecture, and missing in the concrete architecture.

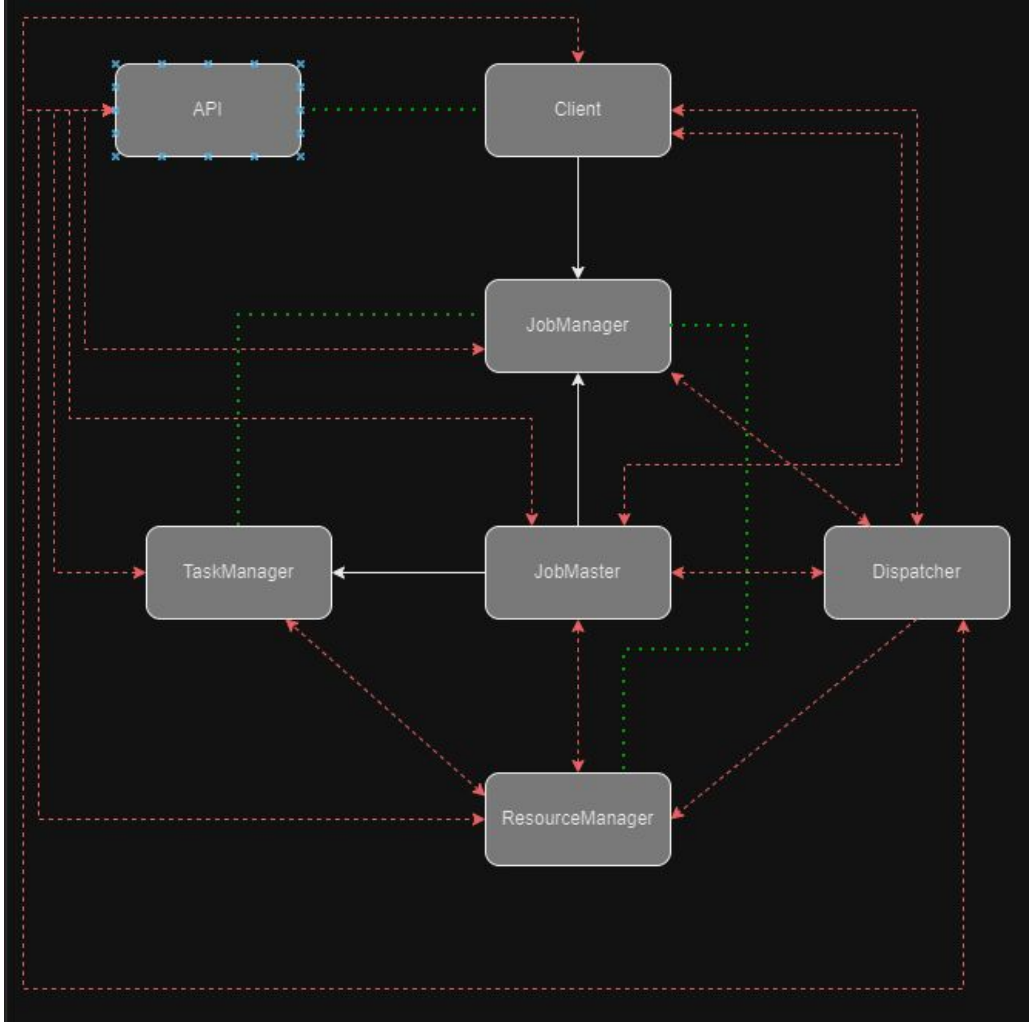


Figure 4 Divergence and Absences Found

# Divergence (ResourceManager ⟷ JobMaster)

| | |
|---|---|
| WHAT | connectToResourceManager (flink-runtime/…/runtime/jobmaster/slotpool/SlotPool.java) depends on ResourceManagerGateway (flink-runtime/…/runtime/resourcemanager/ResourceManagerGateway.java) |
| WHO | Stephen Ewen (Senior Programmer, Co-Creator) |
| WHEN | Created 08/02/2017 Resolved 08/29/2017 |
| WHY | The following are **core aspects of the ResourceManager** design:<br><br>The ResourceManager no longer has a resource pool size, but **receives individual requests for slots**. That way, jobs can request TaskManagers of different resources (Memory/CPU).<br><br>/**<br>* Connects the SlotPool to the given ResourceManager. After this method is called, the SlotPool<br>* will be able to request resources from the given ResourceManager.<br>*<br>* @param resourceManagerGateway The RPC gateway for the resource manager. |

# Divergence (JobMaster ←→ Dispatcher)

| | |
|---|---|
| WHAT | (flink-runtime/…/runtime/jobmaster/) two way dependency on (flink-runtime/…/runtime/dispatcher/) |
| WHO | Till Rohrmann (Flink PMC Member) |
| WHEN | Created 11/Feb/18 00:37 Resolved 23/Feb/18 09:25 |
| WHY | In order to call the JobMaster#rescaleJob via Rest handlers, it has to be exposed via the Dispatcher.<br><br>Extracted from jobResult.java<br>/**<br>* Similar to {@link org.apache.flink.api.common.JobExecutionResult} but with an optional {@link<br>* SerializedThrowable} when the job failed.<br>*<br>* **<p>This is used by the {@link JobMaster} to send the results to the {@link Dispatcher}.**<br>*/ |

# Divergence (Dispatcher → ResourceManager)

| | |
|---|---|
| WHAT | flink-runtime/…/dispatcher/Dispatcher.java to<br>flink-runtime/…/runtime/resourcemanager/ResourceOverview.java |
| WHO | Till Rohrmann (Flink PMC Member) |
| WHEN | Created Oct 10, 2017 Resolved Oct 12, 2017 |
| WHY | `This commit implements the ClusterOverview generation on the Dispatcher. In order to do this, the Dispatcher requests the ResourceOverview from the ResourceManager and the job status from all JobMasters. After receiving all information, it is compiled into the ClusterOverview.` |

# Divergence (ResourceManager ⟷ TaskManager)

| | |
|---|---|
| WHAT | flink-runtime/…/runtime/resourcemanager/slotmanager/SlotManager.java<br>flink-runtime/…/runtime/rest/messages/taskmanager/SlotInfo.java |
| WHO | YangZe Guo (Flink Committer) |
| WHEN | Created 15/Mar/21 Resolved 24/Mar/21 |
| WHY | **It would be helpful to allow retrieving detailed information of slots via rest api.**<br>● **JobID that the slot is assigned to**<br>● **Slot resources (for dynamic slot allocation)**<br>**Such information should be displayed on webui, once fine-grained resource management is enabled in future.** |

# Divergence (Dispatcher ←→ JobManager)

| | |
|---|---|
| WHAT | flink-runtime/…/runtime/Dispatcher/Dispatcher.java two way with<br>flink-runtime/…/runtime/jobmanager/SubmittedJobGraph.java |
| WHO | Till Rohrmann and Chesnay Schepler (Flink PMC Members) |
| WHEN | Created 04/Jul/17 16:04 Resolved 11/Jul/17 17:47 |
| WHY | `The Dispatcher is responsible for receiving job submissions, persisting the JobGraphs, spawning JobManager to execute the jobs and recovering the jobs in case of a master failure. This commit adds the basic skeleton including the RPC call for job submission.`<br>`Add cleanup logic for finished jobs`<br>`Pass BlobService to JobManagerRunner` |

# Divergence (Client ⟷ Dispatcher)

| | |
|---|---|
| WHAT | flink-1.17.1/flink-clients/…/client/deployment/application/ApplicationClusterEntryPoint.java to flink-1.17.1/flink-runtime/…/runtime/dispatcher/ExecutionGraphInfoStore.java |
| WHO | Till Rohrmann (Flink PMC Members) |
| WHEN | Created 03/Jul/17 22:40 Resolved 26/Jul/17 23:25 |
| WHY | Implement a generic entry point for Flink sessions. This ClusterEntryPoint has to start a ResourceManager, the Dispatcher component and the cluster's RESTful endpoint. This class could serve as the basis for a Mesos- and YarnEntryPoint to run Flink sessions.<br><br>Maybe we can use a common base for the session and the per-job mode. The session has to start a dispatcher component and the per-job mode retrieves the JobGraph and directly starts a JobManager with this job. |

# Rationale For Differences

1. A Higher Level of Abstraction
2. Constant Evolution
3. Checkpoint and State Dependencies
4. Concurrency and Team Issues

# Resource Manager

- Allocating and deallocating resources
- Manages Task Slots
- Distributes the slots of TMs
- Slot caching
- Slot sharing

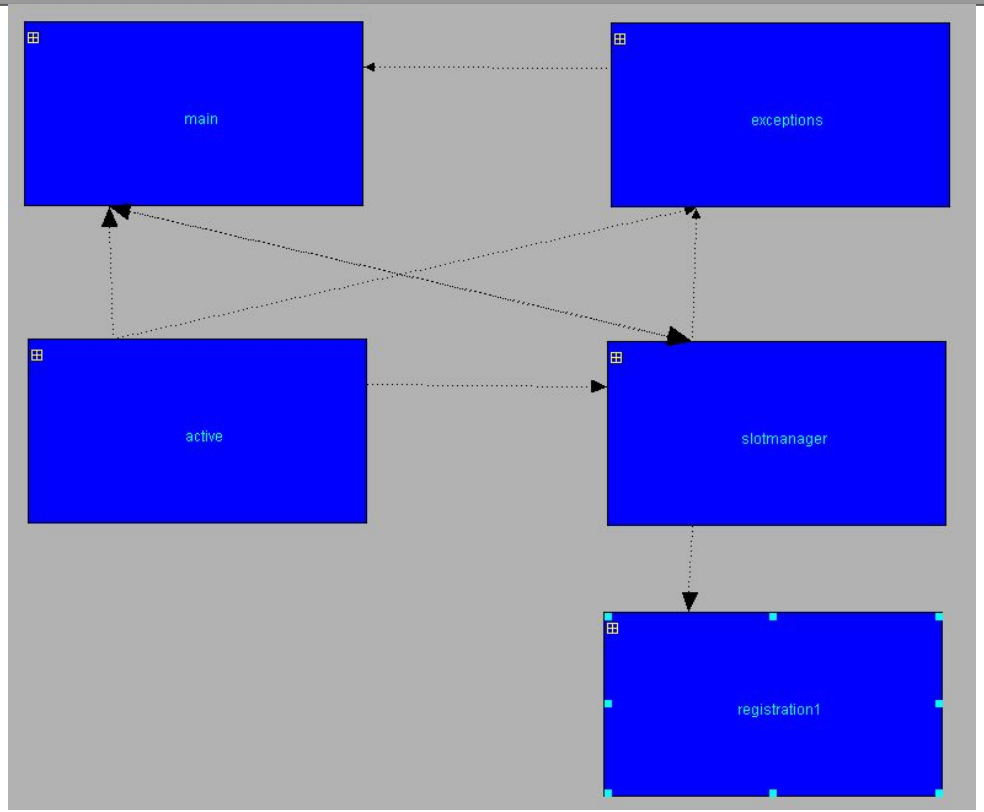


Figure 5 ResourceManager Components and Interactions

# Resource Manager

**ResourceManagerGateway.java**: The ResourceManager's RPC gateway interface.

**ResourceManagerService.java**: Maintains the lifecycle of Resource Manager.

**ResourceManagerServiceImpl.java**: Default implementation of ResourceManagerService

**ResourceManager.java**: The resource manager is responsible for resource deallocation and bookkeeping. It offers various methods as part of its rpc interface to interact with him remotely.

**WorkerResourceSpec.java**: Resource specification of a worker, mainly used by SlotManager requesting from ResourceManager.

# Slot Manager

**ResourceAllocator.java:** *allocates resources to corresponding JobManager*

**ResourceTracker.java:** *tracks for each job how many resources*

*are required/acquired*

**SlotManager.java:** *maintains a view on all registered task manager slots,*

** their allocation and all pending slot requests*

**SlotState.java:** *states whether a slot is free, pending or allocated*

**SlotTracker.java:** *tracks slots and their slot state*

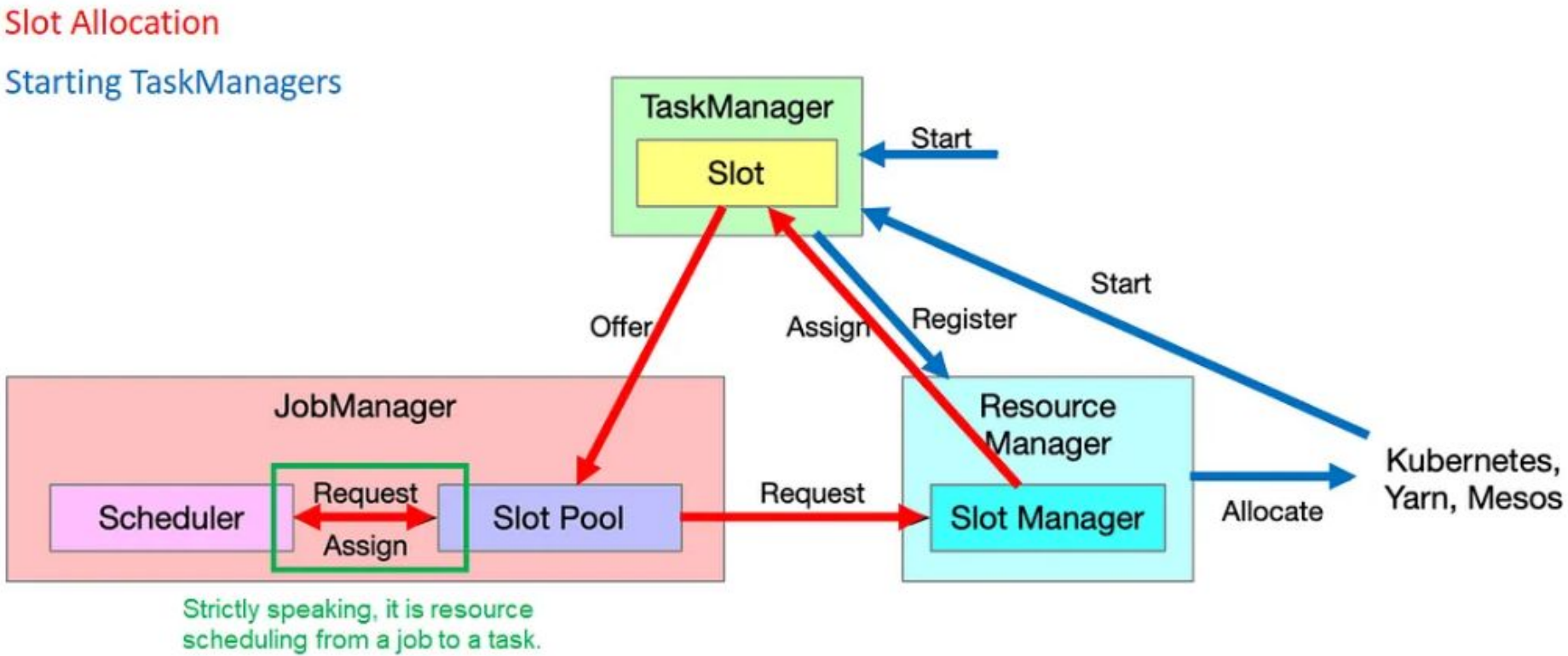


Figure 6

Image taken from: https://alibaba-cloud.medium.com/data-warehouse-in-depth-interpretation-of-flink-resource-management-mechanism-5c13b531abfa

# Previous Sequence Diagram

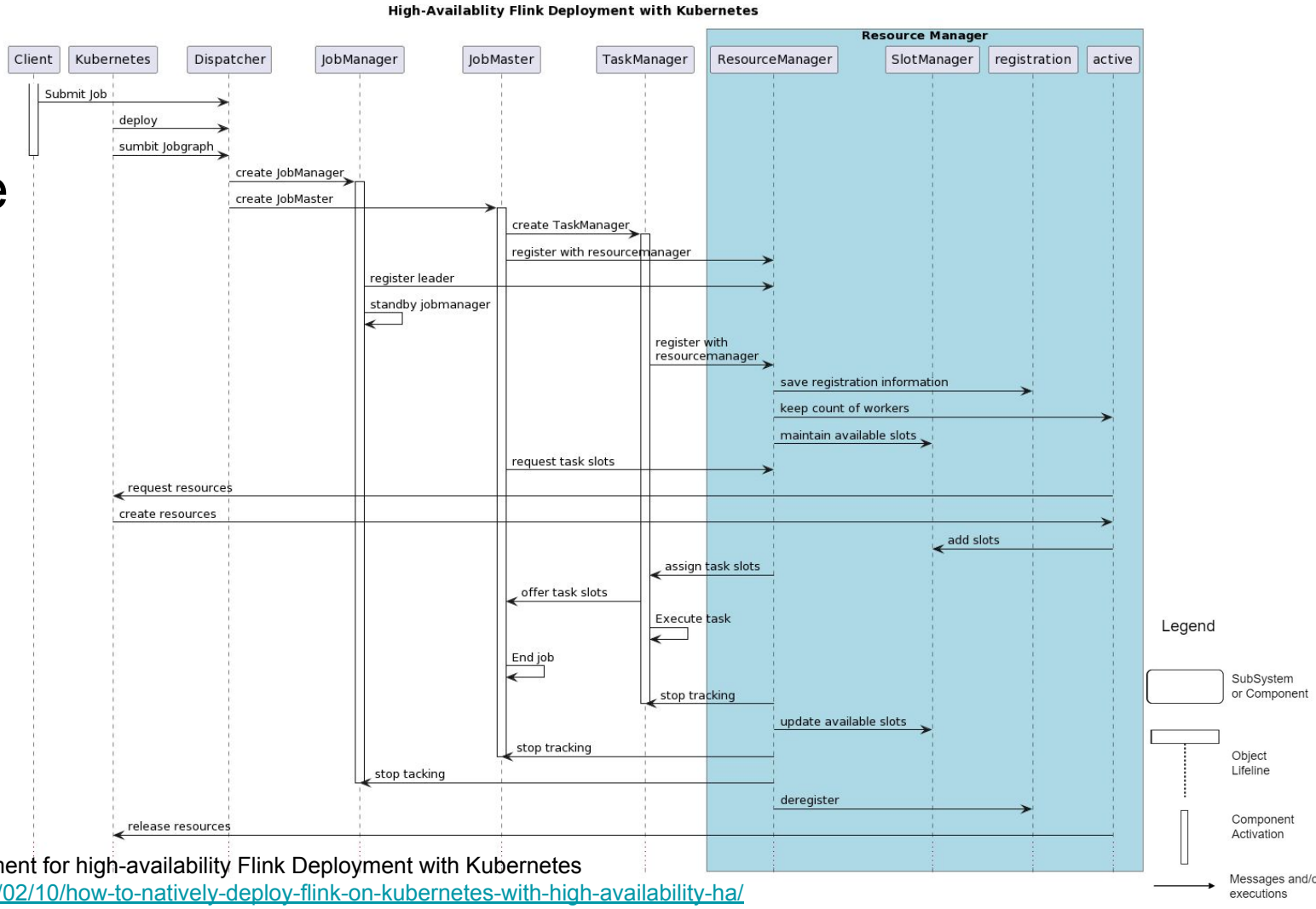


Figure 7 Resource management for high-availability Flink Deployment with Kubernetes
https://flink.apache.org/2021/02/10/how-to-natively-deploy-flink-on-kubernetes-with-high-availability-ha/

# Updated Sequence Diagram

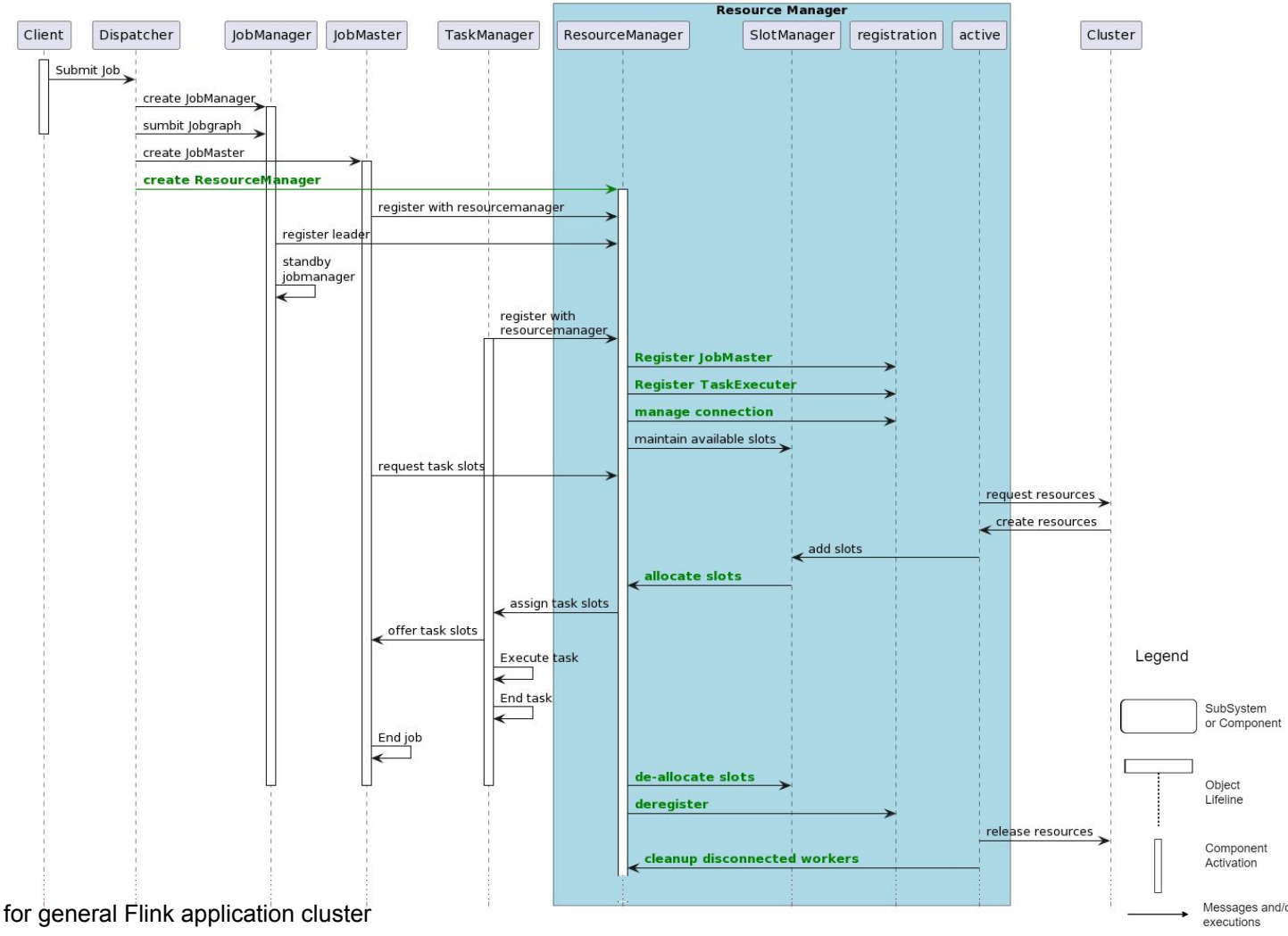


Figure 8 Updated sequence diagram for general Flink application cluster

# Previous Use Case Diagram

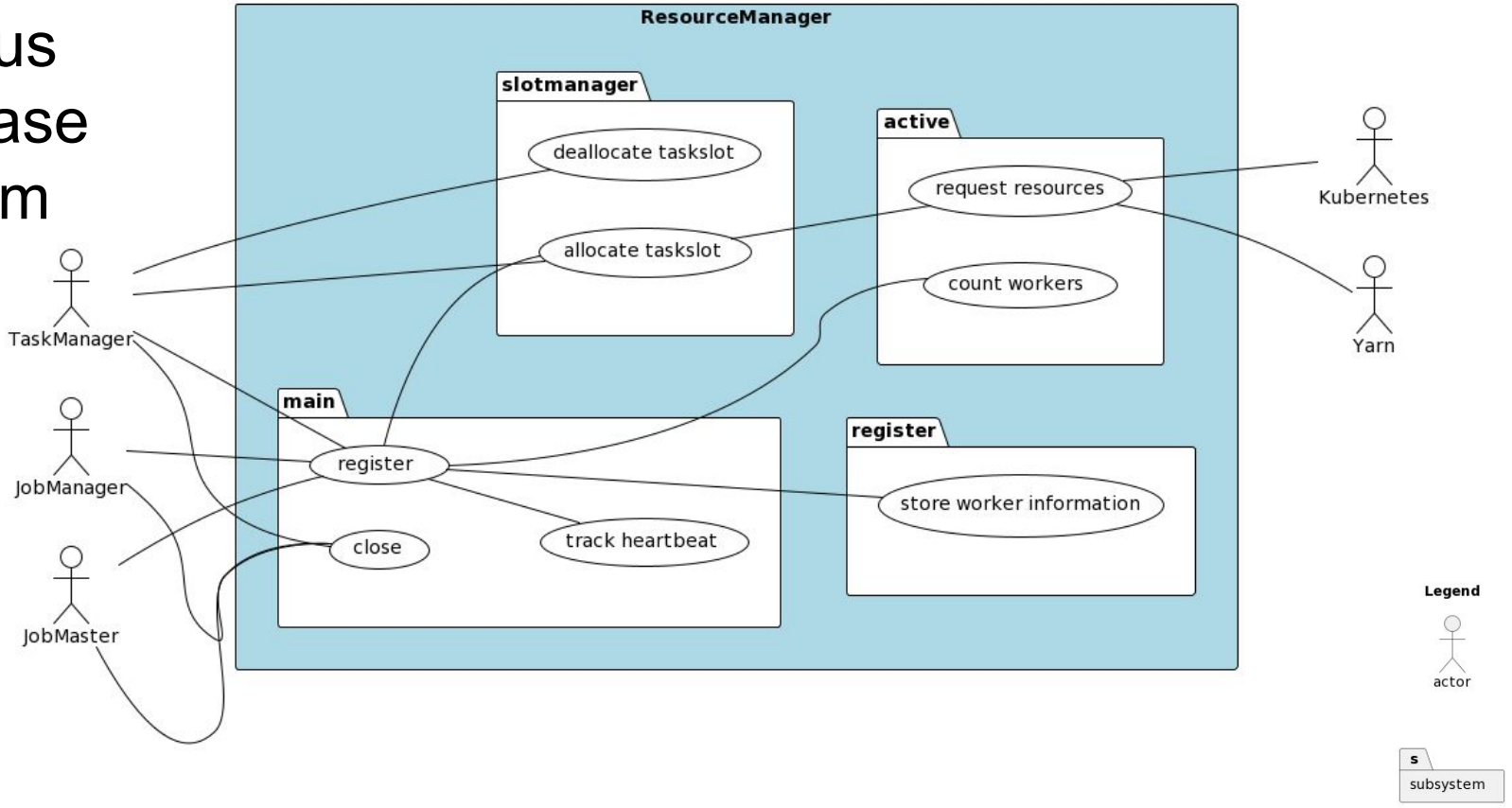


Figure 9 Resource management subsystem use cases
https://flink.apache.org/2021/02/10/how-to-natively-deploy-flink-on-kubernetes-with-high-availability-ha/

# Updated Use Case Diagram

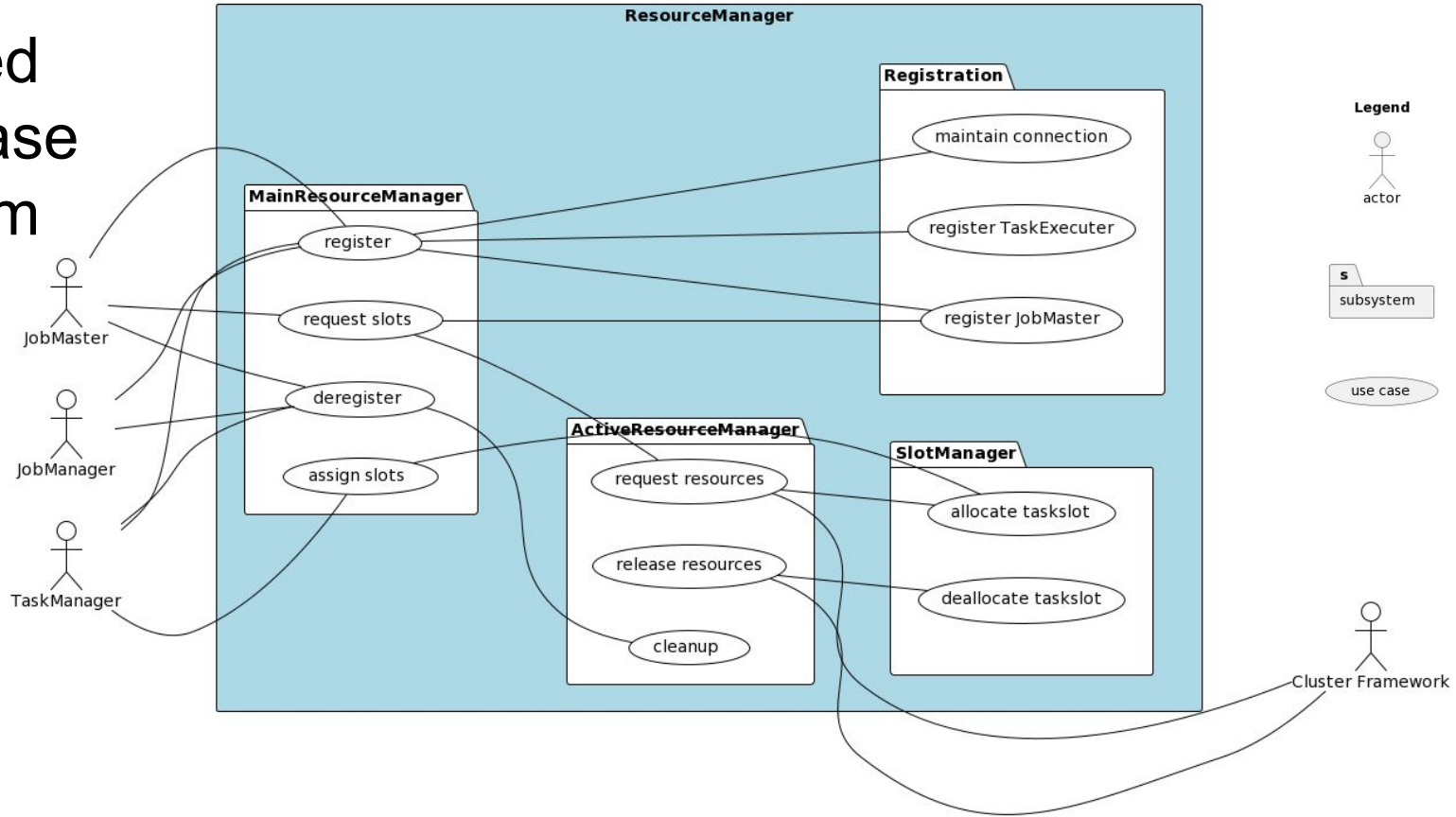


Figure 8 Updated use case diagram of Resource management subsystems

# Limitations

Divergence(ResourceManager←→JobMaster)

- Multiple TaskManagers
- More computations

Divergence(Dispatcher → ResourceManager)

- Unable to see individual JobProgress/ Estimated Completion time

Divergence(ResourceManager←→ TaskManager)

- Cannot list all concurrent Jobs alongside appropriate Slots

# Lessons Learned

- Github version history
  - View when files were added
  - File history did not provide this all the time
- Using scripts
  - Python script to extract dependencies
  - efficiency

# Thank You

# References


https://github.com/apache/flink

https://issues.apache.org/jira/projects/FLINK/summary

Software Reflexion Models: Bridging the Gap between Source and High-Level Models

EECS4314 Lecture Slides