Munin Horizontal Scaling Design

xtUML Project Analysis Note

1 Abstract

This note identifies the design for adding in a Job Management domain to aid in the solution for horizontal scaling.

2 Introduction and Background

The current Munin solution allows us to scale and meet the current requirement for scaling but is limited in that it does not allow for dynamic scaling of Munin.

This design note sets out what can be achieved and introduces a new domain Job Management that shall be used to manage the assignment of jobs to PV workers.

3 Design

If we want to increase the number of PV procesess in the current design to manage the allocated lanes, it is necessary to stop the running system and reconfigure to define new instances of PV and their assigned lanes.

This solution makes use of the worker pattern allowing PV instances to be started without the use of fixed lanes. Jobs shall be assigned by the Job Management domain to PV workers using a round robin technique.

Each PV worker will have to maintain a heartbeat with Job Management, and loss of the heartbeat will cause the worker to be retired. Any in-progress jobs assigned to a PV worker who has retired shall become unassigned and added to the queue of jobs that need processing.

The current thinking is that Job Management will be deployed with Reception as a single instance of the Reception domain has been performant in recent bench testing. Therefore a PV worker shall be made up of AEO and SVDC.

3.1 Register Worker

When a PV worker instance is started it shall assign itself a workerId by generating a UUID.

It shall then periodically send out the registerWorker message to Job Management.

When Job Management receives the request to register from the PV instance it shall be added to the pool of employed PV Worker instances that are employed by Job Management.

If Job Management thinks that the PV worker is already registered, then the request shall be acknowledged as if it were not registered. QUESTION: What should it do with any assigned jobs

under this condition?

If Job Management thinks that the PV worker is retired, it shall migrate it back to employed.

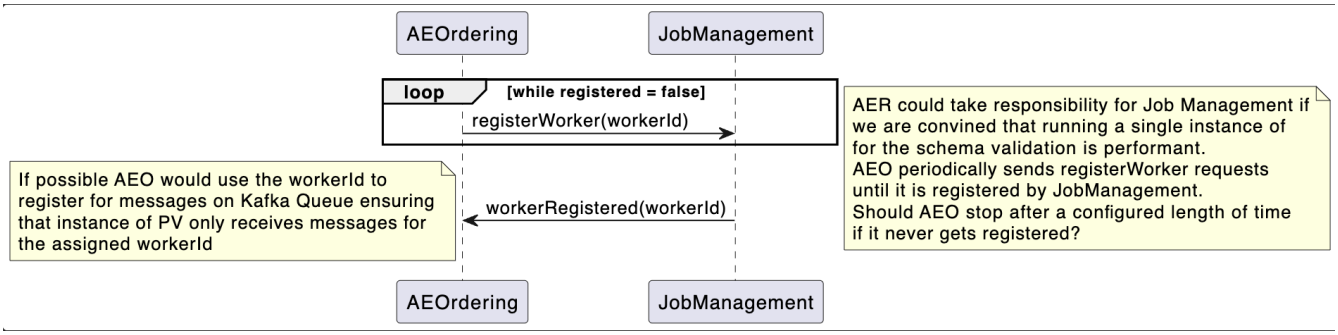

Figure 1. Register Worker

3.2 Worker Heartbeat

The PV worker shall periodically send a heartbeat to Job Management to indicate that it is still operating.

Job Management shall be configured with a timeout that indicates the frequency that this heartbeat should be recieved for each PV worker. When the heartbeat is not received in this period, a counter is incremented and checked against a configurable threshold. If the threshold is breached, then the PV worker is considered to be retired.

If a PV worker is retired while it has in-progress assigned jobs in Job Management, the jobs shall be moved to unassigned and placed back in the queue of jobs to be processed.

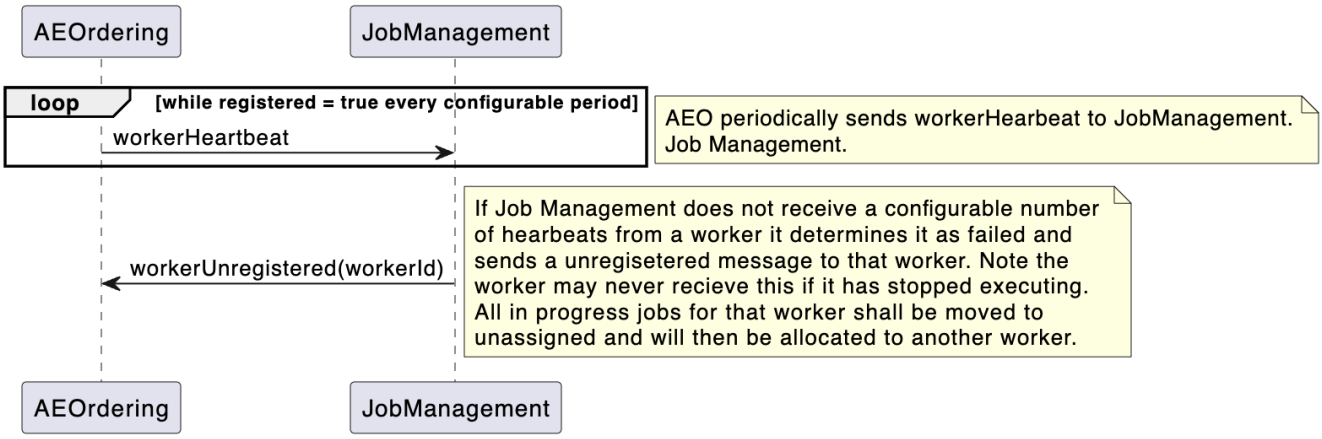

Figure 2. Woker Heartbeat

3.3 Job Assignment

Once Reception has validated the provided event against the schema rather than calling AEOrdering interface acceptEvent, it shall call the Job Management interface acceptEvent.

Job Managment shall check for a job already existing for this event and if not create a new unassigned job and attempt to allocate it to the next available PV worker based on the round robin queue that is being employed. Once assigned the event shall be passed to the PV worker.

Job Managment shall have a maximum number of jobs per PV worker and if they are all maxed out then the job shall remain unassigned.

If an event is received for an existing Job that is already assigned it shall be passed to the PV worker.

If an event is recieved for an unassigned job it shall be stored until that job can be assigned.

Once the PV worker has completed the job it shall report back to job management indicating if it passed or failed. Passed jobs shall be deleted immediately. Failed jobs have a deletion timer that during testing shall allow for further inspection.

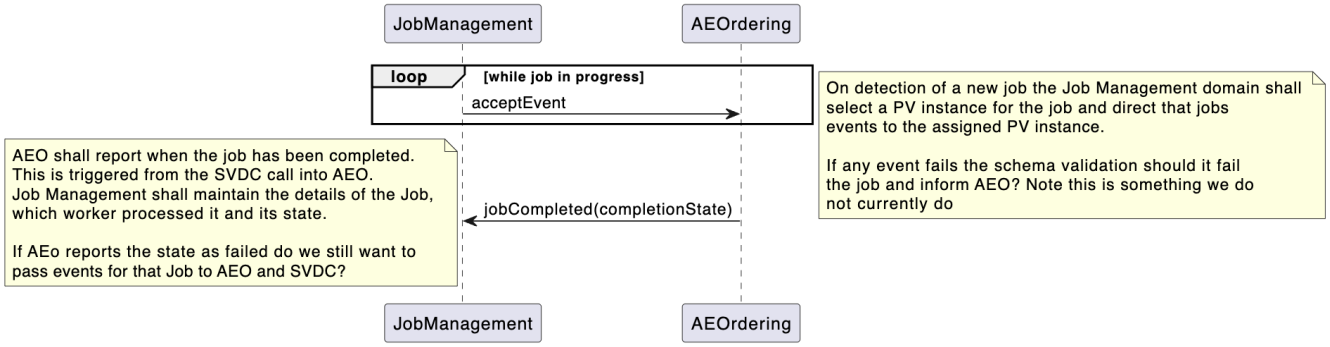

Figure 3. Job Assignment

3.4 Deregister Worker

It is possible for the PV worker to ask to be deregistered. If this happens any assigend jobs that are in progress are moved to unassigned and placed at the back of the queue to be re-assigned. the PV worker is removed to retired and will eventually be deleted from Job Management.

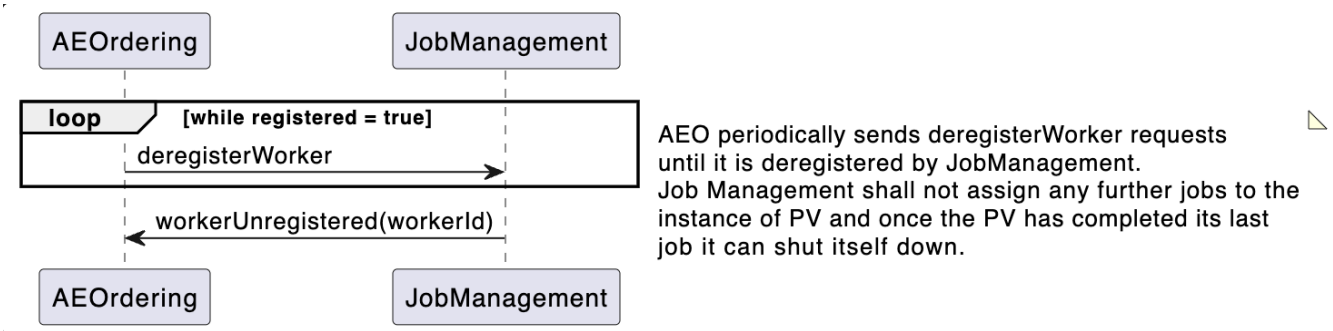

Figure 4. Deregister Worker

3.6 Job Management Domain

As can be seen from the class diagram the only class with any state behaviour is the Employed worker, and that is there to manage the heartbeat from the PV workers.

As all job assignemts are controlled by Job Management, if this process is made persistent, then there is no requirement for a Job ID Store further reducing reliance on file I/O.

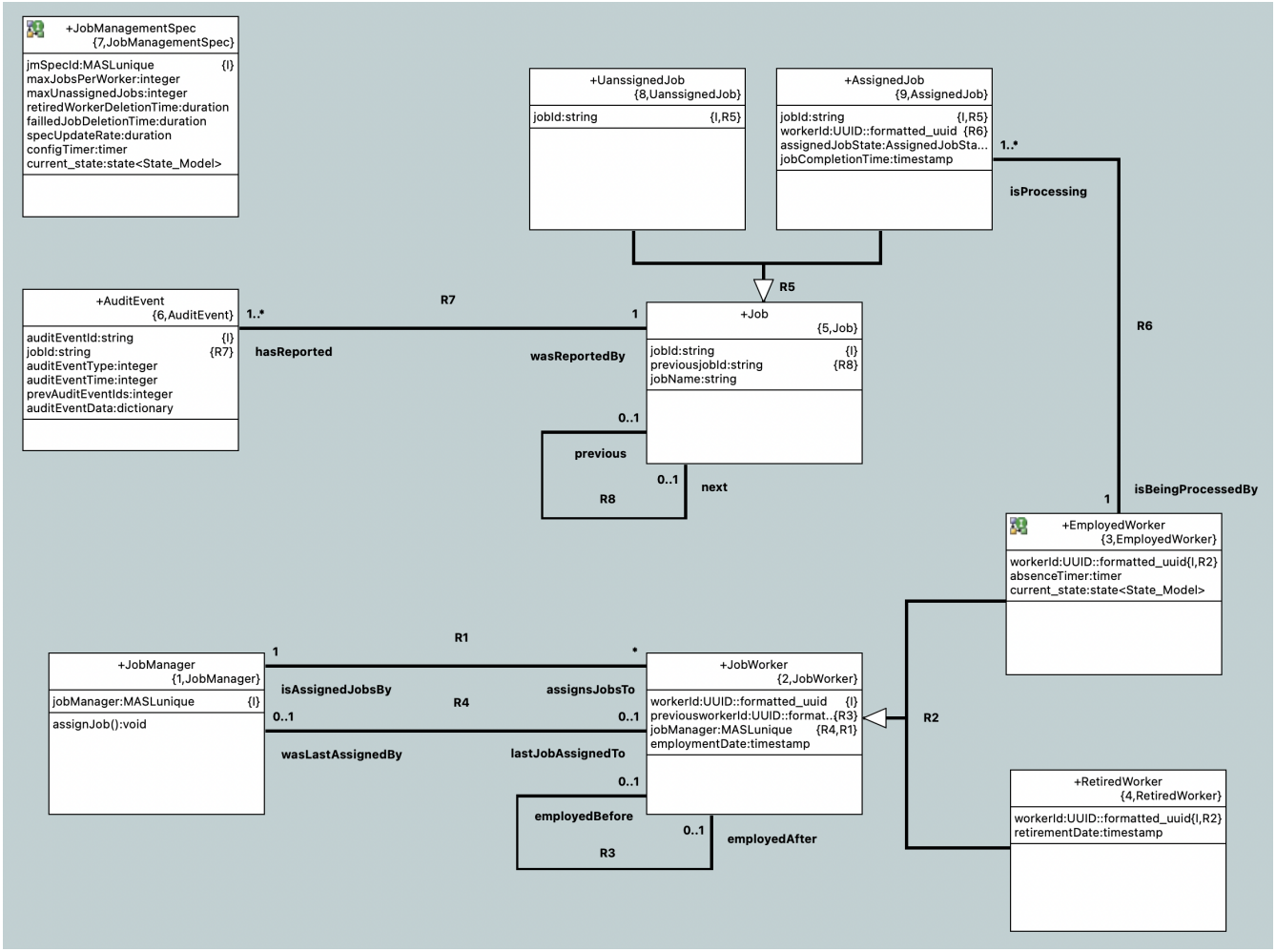

Figure 5. Job Management Class Diagram

4 Work Required

5.1 Job Management

1. Finish implementation of Job Management
2. Create a deployment process containing Job Management and Reception

5.2 AEOrdering

1. Remove "laning" from AEOrdering
2. Add the concept of a worker to AEOrdering to generate the worker id, register with Job Management, deregister with Job Management and maintain the heartbeat with Job Management
3. Update the PV_PROC process to interact with Job Management
4. Remove Job ID Store.

5.3 PV_PROC and Kafka Deployment

1. Update the PV_PROC docker compose to be just one instance and use the docker replica feature.
2. Updte the kafka docker compose for the new messages.

5.4 Kubernetes

To reach full horizontal scaling some research should be carried out into how we can scale the PV_PROC worker instances unsing Kubernetes.