

# Network anomalies or metabase anomalies cause Project Epoch to change frequently, which may cause the job in pending status for a long time and no longer to be executed

## Root Cause

Project Epoch changes frequently in a short period of time, and the same node quickly regains it after losing Project Epoch, and asynchronously executes the callback methods of the above events to stop/start the build task scheduler; but these callback methods do not guarantee sequential execution, resulting in the build task scheduler being started and then stopped without stopping, and finally the build task cannot be scheduled.

Let's take the project **MOD** as an example and look at the log timeline in the diagnostic package (note that the epoch owner is always `10.203.64.231:7070`):

1. `08:19:55,633` After an epoch renew, the MOD project is lost while the `ProjectEscapedNotifier` event is emitted

```
2023-03-28T08:19:55,633 DEBUG [EpochChecker-p-33-t-2] epoch.EpochManager : after renew new epoch size:7, Project MASTER,BMD2,CMOL,BMD,ARMS,_global,SQUAR owned by 10.203.64.231:7070
2023-03-28T08:19:55,633 DEBUG [EpochChecker-p-33-t-2] epoch.EpochManager : after renew escaped projects: CADILLACAI,IOT_PTENERG,MOD,WOL,TEST,learn_kylin,TSPDOP,SFM,SAAP
2023-03-28T08:19:55,633 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectEscapedNotifier {project=CADILLACAI, subject=null} async
2023-03-28T08:19:55,634 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectEscapedNotifier {project=IOT_PTENERGY, subject=null} async
2023-03-28T08:19:55,634 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectEscapedNotifier {project=MOD, subject=null} async
2023-03-28T08:19:55,636 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectEscapedNotifier {project=WOL, subject=null} async
```

2. `08:19:55,924` After another epoch renew, the MOD project is regained, and the `ProjectControlledNotifier` event is issued

(Note that at the same time the SAAP project, also regained, and emitted the `ProjectControlledNotifier` event)

```
2023-03-28T08:19:55,924 DEBUG [EpochChecker-p-33-t-2] epoch.EpochManager : after renew new epoch size:11, Project MASTER,MOD,TEST,BMD2,CMOL,BMD,ARMS,TSPDOP,SAAP,_global
2023-03-28T08:19:55,924 DEBUG [EpochChecker-p-33-t-2] epoch.EpochManager : after renew controlled projects: MASTER,MOD,TEST,CMOL,ARMS,TSPDOP,SAAP,_global
2023-03-28T08:19:55,924 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=MASTER, subject=null} async
2023-03-28T08:19:55,924 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=MOD, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=TEST, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=CMOL, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=ARMS, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=TSPDOP, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=SAAP, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] scheduler.EventBusFactory : Post event ProjectControlledNotifier {project=_global, subject=null} async
2023-03-28T08:19:55,925 DEBUG [EpochChecker-p-33-t-2] epoch.EpochManager : End renew owned epoch cost:0.000000000s
```

3. `08:19:56,033` After only 100ms, it shows that the SAAP project is already executing the callback method of the `ProjectControlledNotifier` event, and its build task scheduler

is already up; But the MOD project that issued the event earlier and other projects never have the log

```
2023-03-28T08:19:56,026 DEBUG [SchedulerEventBus-p-5-t-10943] transaction.UnitOfWork : UnitOfWork 5067388a-1189-2005-b752-0ebdaa0aa20 takes 19ms to complete
2023-03-28T08:19:56,033 DEBUG [SchedulerEventBus-p-5-t-10942] threadpool.NDefaultScheduler : New NDefaultScheduler created by project 'SAAP': 1582401095
2023-03-28T08:19:56,033 INFO [SchedulerEventBus-p-5-t-10942] initialize.EpochChangedListener : start thread of project: SAAP
2023-03-28T08:19:56,033 DEBUG [SchedulerEventBus-p-5-t-10942] transaction.UnitOfWork : UnitOfWork 21b82257-1185-b336-b756-61fcc940db7 started on project SAAP
```

4. 08:20:16,626 shows that the MOD project is already executing the callback method of the ProjectEscapedNotifier event

```
2023-03-28T08:20:16,626 INFO [SchedulerEventBus-p-5-t-10924] initialize.EpochChangedListener : Shutdown related thread: MOD
```

And stopped building task scheduler

```
2023-03-28T08:20:16,688 DEBUG [SchedulerEventBus-p-5-t-10924] scheduler.EventBusFactory : Post event CliCommandExecutor.JobKilled(jobId=31d69bda-7266-ccaf-aac1-1c292058
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] initialize.ProcessStatusListener : Cannot find pid for job:<31d69bda-7266-ccaf-aac1-1c292058da80-1838edc7-
2023-03-28T08:20:16,688 DEBUG [SchedulerEventBus-p-5-t-10924] scheduler.EventBusFactory : Post event CliCommandExecutor.JobKilled(jobId=731c0392-3fe3-f8c9-4fe2-f5fcc200
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] initialize.ProcessStatusListener : Cannot find pid for job:<731c0392-3fe3-f8c9-4fe2-f5fcc200a97f-c15e0463-
2023-03-28T08:20:16,688 DEBUG [SchedulerEventBus-p-5-t-10924] scheduler.EventBusFactory : Post event CliCommandExecutor.JobKilled(jobId=4aa7e98e-7883-1d24-30b2-9e9167d7
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] initialize.ProcessStatusListener : Cannot find pid for job:<4aa7e98e-7883-1d24-30b2-9e9167d7e042-f02ab858-
2023-03-28T08:20:16,688 DEBUG [SchedulerEventBus-p-5-t-10924] scheduler.EventBusFactory : Post event CliCommandExecutor.JobKilled(jobId=3ce1b637-e07d-3b93-a8f0-0c2baf1e
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] scheduler.EventBusFactory : Cannot find pid for job:<3ce1b637-e07d-3b93-a8f0-0c2baf1e87db-57ce1d70-
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] task.QueryHistoryTaskScheduler : Shutting down QueryHistoryAccelerateScheduler for [MOD] ....
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Shutdown now thread pool [1075325565], drain [2] runnable jobs.
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Cancel future task [1323289614].
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Cancel future task [1256210701].
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Thread poll cancel [2] future jobs.
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] threadpool.NDefaultScheduler : Force to shut down DefaultScheduler for project MOD ....
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Shutdown now thread pool [266520117], drain [3] runnable jobs.
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Cancel future task [245911158].
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Cancel future task [1051151377].
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Cancel future task [1980209889].
2023-03-28T08:20:16,688 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Thread poll cancel [3] future jobs.
2023-03-28T08:20:16,689 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Shutdown now thread pool [1875720570], drain [0] runnable jobs.
2023-03-28T08:20:16,689 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Thread poll cancel [0] future jobs.
2023-03-28T08:20:16,689 INFO [SchedulerEventBus-p-5-t-10924] scheduler.StreamingScheduler : Shutting down DefaultScheduler ....
2023-03-28T08:20:16,689 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Shutdown now thread pool [593101067], drain [0] runnable jobs.
2023-03-28T08:20:16,689 INFO [SchedulerEventBus-p-5-t-10924] util.ExecutorServiceUtil : Thread poll cancel [0] future jobs.
2023-03-28T08:20:16,689 DEBUG [SchedulerEventBus-p-5-t-10924] task.RecommendationTopNUpdateScheduler : cancel MOD future task
```

There are still 2 doubts here, which are answered separately:

- Why is the ProjectEscapedNotifier emitted earlier but executed later than the ProjectControlledNotifier event?

Because the AsyncEventBus of guava is used here for event distribution, the multi-threaded scenario does not guarantee the distribution order;

At the same time, the AsyncEventBus execution callback puts the task into the thread pool, and the execution of the task cannot guarantee the order; (the problem point, KE is configured with 100 threads by default, and tasks submitted after a short period of time may also be executed first)

- Why only logs with SAAP project started?

Related to the code logic, it is speculated that the callbacks of other projects are returned in the red box in the following figure, because the escaped callback has not been executed at this time, so the oldScheduler in the red box is indeed still running

```

72 @Subscribe
73 @ public void onProjectControlled(ProjectControlledNotifier notifier) throws IOException {
74     String project = notifier.getProject();
75     val kylinConfig : KylinConfig = KylinConfig.getInstanceFromEnv();
76     val epochManager : EpochManager = EpochManager.getInstance();
77     if (!GLOBAL.equals(project)) {
78
79         if (!EpochManager.getInstance().checkEpochValid(project)) {
80             log.warn("epoch:{} is invalid in project controlled", project);
81             return;
82         }
83
84         val oldScheduler : NDefaultScheduler = NDefaultScheduler.getInstance(project);
85
86         if (oldScheduler.hasStarted()
87             && epochManager.checkEpochId(oldScheduler.getContext().getEpochId(), project)) {
88             return;
89         }
90
91         // if epoch id check failed, shutdown first
92         if (oldScheduler.hasStarted()) {
93             oldScheduler.forceShutdown();
94         }
95
96         log.info("start thread of project: {}", project);
97         NDefaultScheduler scheduler = NDefaultScheduler.getInstance(project);
98         EnhancedUnitOfWork.doInTransactionWithCheckAndRetry(() -> {
99             scheduler.init(new JobEngineConfig(kylinConfig));
100             if (!scheduler.hasStarted()) {
101                 throw new RuntimeException("Scheduler for " + project + " has not been started");
102             }

```

## Fix Design

In general, if there are no frequent project epoch changes in a short period of time, this problem will not occur because the callback method is equivalent to executing in order (stopping first, then starting).

Therefore, the repair logic is to ensure that the execution order of the callback method is consistent with the order in which the event notification is issued, and the callback method of the same project needs to be executed serially.

Added `ProjectSerialEventBus`, replacing the original `EventBusFactory` in the above need to issue `ProjectControlledNotifier` and `ProjectEscapedNotifier` events

There are 3 places in total, all located in `EpochManager`:

- `ProjectControlledNotifier`



```

Class org.apache.kylin.common.scheduler.ProjectControlledNotifier (org.apache.kylin.common.scheduler) 8 usages
EpochChangedListener.java 27 import org.apache.kylin.common.scheduler.ProjectControlledNotifier;
EpochChangedListener.java 73 public void onProjectControlled(ProjectControlledNotifier notifier) throws IOException {
EpochManager.java 55 import org.apache.kylin.common.scheduler.ProjectControlledNotifier;
EpochManager.java 387 newControlledProjects.forEach(p -> projectSerialEventBus.postAsync(new ProjectControlledNotifier(p)))
EpochManager.java 688 projectSerialEventBus.postAsync(new ProjectControlledNotifier(epochTarget));
SchedulerEventBusTest.java 31 import org.apache.kylin.common.scheduler.ProjectControlledNotifier;
SchedulerEventBusTest.java 236 listener.onProjectControlled(new ProjectControlledNotifier(prj));
SchedulerEventBusTest.java 257 listener.onProjectControlled(new ProjectControlledNotifier(prj));
Press ⌘F7 again to search in 'Project Files'

```

ProjectEscapedNotifier

```

Class org.apache.kylin.common.scheduler.ProjectEscapedNotifier (org.apache.kylin.common.scheduler) 7 usages
EpochChangedListener.java 28 import org.apache.kylin.common.scheduler.ProjectEscapedNotifier;
EpochChangedListener.java 134 public void onProjectEscaped(ProjectEscapedNotifier notifier) {
EpochManager.java 56 import org.apache.kylin.common.scheduler.ProjectEscapedNotifier;
EpochManager.java 366 projectSerialEventBus.postAsync(new ProjectEscapedNotifier(project));
SchedulerEventBusTest.java 32 import org.apache.kylin.common.scheduler.ProjectEscapedNotifier;
SchedulerEventBusTest.java 238 listener.onProjectEscaped(new ProjectEscapedNotifier(prj));
SchedulerEventBusTest.java 259 listener.onProjectEscaped(new ProjectEscapedNotifier(prj));
Press ⌘F7 again to search in 'Project Files'

```

ProjectSerialEventBus has a Queue inside to ensure that only one callback method is running in the same project at the same time;

The thread that finally executes the callback method is still handed over to AsyncEventBus in EventBusFactory for processing

