# Asynchronous Application Submission

## Current Situation

Currently, in ClientRMService#submitApplication, two operations are executed:

1. Call RMAppManager to create a RMApp and start it;
2. Persist the RMApp instance through RMSateStore.

Since RMStateStore#storeApplication is a synchronized blocking API, and is likely to be slow (e.g., persisting RMApp on FS), the concurrent application submissions are at the risk of being blocked by a slow submission.

## Motivation and Goal

Therefore, to avoid some application submissions' being blocked by a slow one, we need to make ClientRMService#submitApplication and RMStateStore#storeApplication become non-blocking APIs.

## Roadmap of the Changes

1. In ClientRMService#submitApplication, we don't call RMAppManager to handle the submission even directly. Instead, we send the event to AsyncDispatcher of ResourceManager.
2. In ClientRMService#submitApplication, we don't call RMStateStore#storeApplication directly.
3. We change RMStateStore#storeApplication from blocking API to non-blocking API, sending an event to the dispatcher of RMStateStore own.
4. In RMAppState, we add one more state, named "SAVING". An RMApp should transit from NEW to SAVING, then to SUBMITTED. During the SAVING stage. RMApp can schedule the storing application event in the SAVING state. The merit of this state transition flow is that it should be equivalent to the original flow (blocking RMApp instance creation <NEW> and storing application <SAVING> together first). The alternative choice is to delay the SAVING state until ACCEPTED. Then, storing application may not be necessary when the application is killed or rejected in the earlier states. Your opinions are welcome.
5. In RMAppManager#submitApplication, we send the event to move RMApp to SAVING instead of SUBMITTED.
6. In RMAppImpl, we need to modify the state machine, add the state to process the SAVING state.
7. We need to block YarnClientImpl#submitApplication from returning ApplicationId until the RMApp instance is created or enters some state (SUBMITTED or ACCEPTED?). Your idea, please.
8. In RM side (YARNRunner#submitJob), we don't want to getApplicationReport immediately after submitApplication to judge whether the job runs successfully or not.