**log4j.logger.org.ow2.proactive.scheduler.resourcemanager.nodesource**=**DEBUG**

Améliorer les choses également. La notion d’autoscale est très importante surement améliorer les choses. L’objet de cette policy (SchedulerLoadingPolicy) est de piloter le nombre de noeuds à déployer/libérer dans le RM pour faire face à la charge du scheduler. Avec une règle du type “je veux avoir 1 noeud pour chaque groupe de 10 taches en attente"

Et il y a une subtilité sur la notion “tache en attente”.

c’est plus la flexibilité du dialogue entre la NS Policy (qui dit quand on doit faire +1/-1 ou +j/-k) et la NS Infrastructure (qui reçoit et provisionne les +j/-k ProActive Nodes)

Détermination dynamique du nombre de nœuds à ajouter/elever en fonction du nombre de tasks elligibles

1 nœud🡪 N tâche c’est une configuration faite lors de la creation du node source (composé de deux entités qui sont l’infrastructure (comment accéder à ces nodes) et la policy (comment acquérir les nodes ex : la nuit, d’une manière statique….)

But : de permettre d’adapter la taille de la plateforme c'est-à-dire la capacité (nombre de nœuds) au besoin (nombre de taches à scheduler)

Task elligible pas en pending 🡪AJOUTER le test sur ce nombre de tâches pour calibrer la creation/suppression dynamique des nodes

Regarder uniquement les tâches éligibles

SchedulerLoadingPolicy accède déjà au scheduler pour les credentials

ok, tu vas rajouter dans l'interface JobDescriptor un extends Serializable

et tu vas mettre le champ internalJob transient

c'est normal, comme internalTask est transient

la seule piste que je vois serait d'ajouter des méthodes get/setGenericInformations dans les interfaces TaskDescriptor et JobDescriptor et après il faudrait les renseigner correctement

Je précise mon idée, il faut que la ExtendedSchedulingPolicy passe par ces méthodes plutôt que faire appel à internalJob internalTask

C'est dans la SchedulerLoadingPolicy qu'on interroge le scheduler à des moments réguliers

[11:45]

De manière configurable

amine [11:46 AM]

je cherche à avoir ces moments

fabienviale

[11:46 AM]

C'est normal si la policy n'est pas updaté instantanément

amine [11:46 AM]

protected SchedulerEvent[] getEventsList() {

return new SchedulerEvent[] { SchedulerEvent.JOB\_RUNNING\_TO\_FINISHED, SchedulerEvent.JOB\_SUBMITTED,

SchedulerEvent.TASK\_RUNNING\_TO\_FINISHED, SchedulerEvent.JOB\_PENDING\_TO\_FINISHED };

}

[11:46]

il y a cette liste d'event

[11:47]

mais elle est liée au job et task

fabienviale

[11:47 AM]

Oui il n'y a pas d'event Scheduler.SCHEDULING\_JOBS

amine [11:47 AM]

exact !

fabienviale

[11:47 AM]

En même temps quel est le problème, enfin, c'était le cas avant non ?

[11:48]

La policy fait du polling sur le scheduler, y a pas de gestion asyncrhone

amine [11:48 AM]

je ne sais pas qd dois je appeler la méthode

[11:48]

pour récupérer le nombre de tasks schedulée

[11:48]

l'appel doit être lié à un événement

fabienviale

[11:48 AM] dans la policy il y

[11:48]

@Configurable(description = "refresh frequency (ms)")

private int refreshTime = 1000;

amine [11:48 AM]

ahhhh

fabienviale

[11:49 AM]

donc toutes les secondes elle est censé contacter le scheduler

[11:50]

je vois la policy faisait sans doute un truc un peu bizarre

amine [11:51 AM]

moi je veux avoir un événement qui me permet de récupérer le nombre de eligible tasks (non nul)

fabienviale

[11:51 AM]

en fait les méthodes computeRequiredNodesForPendingJob sont appelées en fonction des évênements des jobs

[11:51]

et computeRequiredNodesForRunningJob

[11:51]

il faudrait sans doute changer ça

amine [11:52 AM]

en fait j'utilise plus ces méthodes

fabienviale

[11:52 AM]

Plutôt faire la logique dans la méthode updateNumberOfNodes

amine [11:52 AM]

car c'est lié au job

fabienviale

[11:53 AM]

qui est appelée périodiquement

[11:53]

Donc tous les cycles, on appelle updateNumberOfNodes qui est censé faire le bon boulot

[11:53]

sans doute pas facile à écrire

[11:53]

Car il faut être sûr qu'on ne génère pas de nouveaux noeuds pour les mêmes tâches

amine [11:57 AM]

le problème est que j'arrive pas à savoir où appeler la méthode computeRequiredNodesForEligibleTask(). Est ce dans case TASK\_RUNNING\_TO\_FINISHED ? ou ailleurs

fabienviale

[11:57 AM]

Euh, je ne crois pas du tout dans TASK\_RUNNING\_TO\_FINISHED

amine [11:57 AM]

okey

fabienviale

[11:58 AM]

tu as une map activeTasks qui contient les tasks et les noeuds deployés

amine [11:58 AM]

en fait l'appel à mon sens dois être triguer par le cycle de scheduler

fabienviale

[11:58 AM]

le computeRequiredNodesForEligibleTasksssss() doit être appelé à chaque cycle de refresh de la policy

[11:59]

il y a sans doute des choses à repenser dans la policy actuelle c'est sûr

amine [11:59 AM]

runActivity

[12:00]

dans cette méthode ? public void runActivity(Body body)

[12:00]

car elle fait update Number Of Nodes();

fabienviale

[12:08 PM]

oui c'est ça qui fait un refresh périodique

[12:08]

Donc la méthode updateNumberOfNodes doit être celle qui déclenche l'appel à computeRequiredNodesForEligibleTasks() (edited)

[12:10]

Ceci dit, je suis sûr qu'il y a des logiques à revoir

[12:10]

tu ne peux pas t'en sortir avec les infos enregistrées actuellement

[12:10]

Je pense

[12:11]

typiquement, la structure private Map<JobId, Integer> activeTasks;

[12:11]

n'est pas suffisante

amine [12:11 PM]

il faut rajouter l'Id task ?

fabienviale

[12:11 PM]

je crois oui

amine [12:11 PM]

pour éviter la redondance

fabienviale

[12:11 PM]

c'est ça

[12:12]

Il faut un peu tout revoir, comme avant ça se basait sur les évênements TaskStatus.PENDING

[12:12]

C'était plus facile

amine [12:13 PM]

je dois rajouter un champ qui contient les tasks running ?

[12:14]

indépendamment de leur Job

fabienviale

[12:14 PM]

Il faudrait avoir une map <TaskId,number\_of\_nodes>

amine [12:14 PM]

et de les retirer des elligible tasks

[12:15]

oui oui

[12:15]

c'est vrai

fabienviale

[12:15 PM]

appeler la méthode plutôt updateRequiredNodesForEligibleTasks

[12:15]

qui va renseigner cette map

[12:17]

Rajouter aussi une map <TaskId, boolean node acquisition started>

[12:17]

Et pour chaque tâche qui n'a pas déjà lancé l'aquisition de noeud, lancer l'aquisition

[12:18]

Tu peux si tu veux créer un classe interne qui contient le nombre de noeuds à aquérir et le boolean

[12:18]

Pour éviter d'avoir deux maps à gérer

amine [12:19 PM]

je n'ai pas bien compris l'histoir de l'acquisition

fabienviale

[12:19 PM]

<TaskId, NodeAcquisition>, où NodeAcquisition { number\_of\_nodes, acuisition\_started }

amine [12:19 PM]

elle sert à quoi exactement ?

[12:20]

j'ai compris :smile:

fabienviale

[12:20 PM]

ben dans un premier cycle tu vas trouver des noeuds à déployer pour différentes tâches. Tu vas donc lancer un certain nombre de startNodes

[12:20]

Lors du deuxième cycle, tu ne veut lancer un startNodes que pour le nombre de noeuds pas encore traité

[12:21]

enfin je veux dire acquireNodes()

amine [12:21 PM]

dont NodeAcquisition=false ?

fabienviale

[12:22 PM]

dans updateRequiredNodesForEligibleTasks, pour chaque tâche non encore connue tu rajouter une entrée <TaskId, NodeAcquisition avec acquisition\_started = false

[12:23]

Dans une autre méthode appelée handleNodeAcquisition, tu parcours la map, et pour tous les tâches ont un acquisition\_started = false tu démarre les noeuds avec acquireNodes() et tu mets acquisition\_started = true

amine [12:24 PM]

Je pense que j'ai compris le principe

[12:24]

c'est clair

[12:25]

Merci Fabien !

fabienviale

[12:25 PM]

du coup, il y a des trucs qu'il faudra enlever

amine [12:25 PM]

tout ce qui est Job

[12:25]

garder peut être job submitted ?

fabienviale

[12:25 PM]

Mais il faudra bien gérer quand la tâche est terminée ou d'autres états exotiques !!!!

amine [12:25 PM]

oui oui

fabienviale

[12:26 PM]

Par exemple quand la tâche est redémarrée, skipped, etc

[12:26]

bien voir quoi faire dans ce cas

amine [12:26 PM]

chose pas faite actuellement

fabienviale

[12:26 PM]

exactement

amine [12:27 PM]

je vais voir ça après je pense

fabienviale

[12:27 PM]

Comme la policy n'a jamais trop été utilisée, elle marche pas trop

[12:29]

en terme d'état, il y a TASK\_IN\_ERROR\_TO\_FINISHED qui doit être gérée comme TASK\_RUNNING\_TO\_FINISHED

[12:29]

je pense que TASK\_SKIPPED ne doit pas être traité

[12:30]

(Il faudra faire un test avec une structure de controle IF)

[12:30]

Il reste TASK\_WAITING\_FOR\_RESTART

amine [12:31 PM]

il est censé occuper des nodes ?

[12:31]

ou pas

fabienviale

[12:31 PM]

Que fais-tu actuellement quand tu as un TASK\_RUNNING\_TO\_FINISHED est-ce que tu scale down ?

amine [12:31 PM]

oui

fabienviale

[12:31 PM]

I.e. est-ce que tu supprimes le noeud

amine [12:31 PM]

c'est le policy actuelle

fabienviale

[12:32 PM]

Donc je pense que dans un TASK\_WAITING\_FOR\_RESTART il faut faire aussi un remove node

amine [12:32 PM]

c'est ce qu'elle fait

[12:32]

ahhhh

[12:32]

après allouer de nouveau

fabienviale

[12:32 PM]

Car la tâche doit généralement redémarrer sur un noeud différent

amine [12:33 PM]

c'est noté

fabienviale

[12:33 PM]

Donc pour résumer, on a TASK\_RUNNING\_TO\_FINISHED, TASK\_IN\_ERROR\_TO\_FINISHED et TASK\_WAITING\_FOR\_RESTART qui fait la même chose

amine [12:33 PM]

supprimer les nodes

fabienviale

[12:33 PM]

oui

amine [12:34 PM]

ajouter ces événement ici otected SchedulerEvent[] getEventsList()

[12:34]

@Override

protected SchedulerEvent[] getEventsList() {

return new SchedulerEvent[] { SchedulerEvent.JOB\_RUNNING\_TO\_FINISHED, SchedulerEvent.JOB\_SUBMITTED,

SchedulerEvent.TASK\_RUNNING\_TO\_FINISHED, SchedulerEvent.JOB\_PENDING\_TO\_FINISHED };

}

[12:34]

et les traiter après

fabienviale

[12:34 PM]

oui

amine [12:35 PM]

dac

[12:36]

Merci Fabien

fabienviale

[12:36 PM]

Je sais pas trop ce que ça va donner honnêtement

[12:37]

Le problème que je vois c'est qu'on a peu de contôle sur les noeuds qu'on déploie

[12:37]

On connais juste le nombre en gros

[12:37]

Et pour supprimer pareil, on fait removeNodes(nbNodes)

[12:38]

Or il faudrait pouvoir supprimer des noeuds spécifiques

[12:38]

Je sais pas trop

[12:38]

C'est pas facile

amine [12:38 PM]

je pense que la suppression se fait sur les free nodes

[12:38]

donc ceux libérer par finished task

[12:39]

donc forcement les seuls free nodes

[12:39]

non ?

fabienviale

[12:39 PM]

Oui je pense que c'est ça

[12:39]

Ecoute faudra faire pas mal de tests, pour voir comment ça se comporte dans différents cas

amine [12:40 PM]

ouiii je sais

[12:40]

bcp de job au meme temps

fabienviale

[12:40 PM]

Des tâches en restart, skipped, etc

amine [12:40 PM]

oui :smile:

fabienviale

[12:40 PM]

Pour voir si on n'arrive pas dans un situation de blocage, ou des noeuds qui ne sont jamais libérés

fabienviale

[2:17 PM]

je me demande si ce n'est pas dû aux histoires de timestamp

[2:18]

je comprends pas trop ce qui est fait avec les timers

[2:19]

déjà tu as private int computeRequiredNodesForPendingJob(JobState jobState)

amine [2:19 PM]

ce que j'ai fait a l'air comment ?

fabienviale

[2:19 PM]

qui ne devrait plus être appelé non ?

amine [2:20 PM]

juste à l'intérieur de la classe

[2:20]

c'est une méthode qui existe déjà

fabienviale

[2:20 PM]

ok tu ne l'appelles pas

amine [2:20 PM]

dans la policy

fabienviale

[2:20 PM]

ensuite ce code:

[2:20]

if (!tasksState.containsKey(taskId)) {

tasksState.put(taskId, new NodeAcquisition(etd.getNumberOfNodesNeeded(), false));

nodesForEligibleTasks += etd.getNumberOfNodesNeeded();

} else {

if (tasksState.containsKey(taskId) && !tasksState.get(taskId).isAcquisition\_started()) {

nodesForEligibleTasks += etd.getNumberOfNodesNeeded();

}

}

amine [2:20 PM]

yes c'est moi

fabienviale

[2:20 PM]

ça marche pas

[2:21]

:slightly\_smiling\_face:

amine [2:21 PM]

ah bon

[2:21]

pourquoi

[2:22]

c'est là où je renseigne la map

[2:22]

1er cycle dans le if

fabienviale

[2:22 PM]

oui mais il faut le faire en deux passes

[2:23]

deux méthodes, une update, une handle

amine [2:23 PM]

tu m'as dit cela avant :slightly\_smiling\_face:

[2:23]

mais je n'ai pas bien compris

fabienviale

[2:23 PM]

le update retourne void et ajoute éventuellement des nouvelles entrées dans la map

[2:24]

le handle retourne le nombre de noeuds à créer

[2:25]

le mieux c'est que le hendle appelle directement les acquireNodes

amine [2:26 PM]

je déplace donc le code à l'interieur de updateNumberOfNodes() dans le handle ?

[2:26]

je veux dire ce bloc

[2:26]

if (requiredNodesNumber > nodesNumberInRM && nodesNumberInNodeSource < maxNodes) {

logger.debug("Node deployment request");

timeStamp = System.currentTimeMillis();

acquireNodes(requiredNodesNumber, new HashMap<String, Object>());

return;

}

if (requiredNodesNumber < nodesNumberInRM && nodesNumberInNodeSource > minNodes) {

logger.debug("Node removal request");

timeStamp = -System.currentTimeMillis();

removeNode();

return;

}

fabienviale

[2:26 PM]

non pas tout ce bloc c'est un peu plus compliqué que ça

[2:27]

je peux pas te l'expliquer en deux lignes

[2:27]

Il faut déjà comprendre à quoi sert le timestamp

amine [2:28 PM]

oui

fabienviale

[2:28 PM]

ensuite il faut que le tasksState.get(taskId).isAcquisition\_started() soit mis à true dès qu'on a appele le acquireNodes()

amine [2:29 PM]

l'appel de handle se fait dans updateNumberOfNodes()

[2:30]

pareil pour update ?

fabienviale

[2:31 PM]

enfin il y a aussi le nodesNumberInRM à gérer

[2:31]

comme la policy écoute le RM pour savoir quand de nouveaux noeuds sont créés

[2:32]

Soi dit en passant, il faudrait que la policy filtre uniquement les noeuds créés dans la node source correspondante, pas dans tout le RM

[2:32]

Y a sans doute encore pas mal de code à changer

[2:32]

Donc je peux pas te l'expliquer en deux minutes

amine [2:32 PM]

ok par étape

[2:33]

je commence à changer ça

[2:33]

après on verra

fabienviale

[2:33 PM]

Ok si tu veux

amine [2:33 PM]

juste par rapport à ma dernière question

fabienviale

[2:33 PM]

Le mieux je pense c'est que tu mettes les logs en debug

[2:34]

les logs de la SchedulerLoadingPolicy

[2:34]

Pour bien comprendre comment ça fonctionne

[2:35]

par exemple ce code là:

[2:35]

if (timeStamp > 0) {

logger.debug("Pending node deployment request");

// pending node deployment

if (System.currentTimeMillis() - timeStamp > nodeDeploymentTimeout) {

logger.debug("Node deployment timeout.");

timeStamp = 0;

}

}

if (timeStamp != 0) {

if (timeStamp < 0) {

logger.debug("Pending node removal request");

}

return;

}

[2:35]

Moi je n'y comprends rien

amine [2:37 PM]

ils sont déjà en debug non ?

fabienviale

[2:37 PM]

Il faut activer les logger pour cette classe en debug dans la config log4j

amine [2:37 PM]

okey

fabienviale

[2:39 PM]

en voyant le code, ça ne m'étonne pas que c'est totalement aléatoire

[2:39]

C'est super compliqué

[2:39]

Ce qu'il faudrait c'est clarifier les choses

[2:39]

y a en plus des machins comme ça:

[2:39]

int requiredNodesNumber = activeTask / loadFactor + (activeTask % loadFactor == 0 ? 0 : 1);

[2:40]

youpi

amine [2:40 PM]

hahaha

fabienviale

[2:40 PM]

Bref, il faudrait que tu remettes les choses à plat, formalise sur une feuille de papier ou un document word

amine [2:40 PM]

oui c'est vrai

[2:41]

je trouve pas le log4j.properties dans le projet rm

fabienviale

[2:42 PM]

Tu dois faire intervenir les différents concepts : Tâches éligibles, noeuds présent dans le RM à un instant T, noeuds qui doivent être déployés, noeuds qui ont été déployés, tâches terminées, noeuds à libérer

[2:42]

c'est dans config/log/server.properties

amine [2:43 PM]

config de tout le projet scheduling

fabienviale

[2:43 PM]

l'idée de la policy est d'ajouter suffisament de noeuds au RM pour pouvoir gérer toutes les tâches eligibles modulo un load factor

amine [2:43 PM]

exact$

fabienviale

[2:43 PM]

Si le RM a assez de noeuds, rien à faire

amine [2:44 PM]

oui

fabienviale

[2:44 PM]

L'idée de base je comprens, maintenant l'implémentation je ne comprends pas, et le timestamp encore moins

[2:45]

Donc je peux difficilement t'aider plus ...

[2:48]

Ce qu'il faudra faire c'est créer des structure de données parlantes (Map, List, je ne sais pas), qui aident à clarifier le code et les concepts

amine [2:48 PM]

c'est compliqué.... une dernière question l'appel de update se fait comme handle dans updateNumberOfNodes() ou ailleurs ?

fabienviale

[2:49 PM]

je réfléchis

amine [2:51 PM]

je le fais en partie ici dans taskStateUpdatedEvent() case TASK\_PENDING\_TO\_RUNNING:

fabienviale

[2:51 PM]

peut être que l'utilisation d'une map par task id n'est pas nécessaire, finalement ?

[2:52]

Peut-être qu'on veut juste une méthode computeNumberOfNodesForEligibleTasks() qui retourne un nombre de noeuds désiré

amine [2:52 PM]

oui mais il y la redondance

[2:53]

si les tâches ne sont pas finies

fabienviale

[2:53 PM]

Puis en fonction de ce nombre de noeuds, après application du load factor et des noeuds déjà présents, et des noeuds déjà demandés, on sait combien de nouveaux noeuds on a besoin

[2:54]

Il faut donc savoir :

[2:54]

1) le nombre de noeuds dont on a besoin

[2:54]

2) le nombre de noeuds total dans le RM

[2:54]

3) le nombre de noeuds qu'on a déjà demandé (acquireNodes)

amine [2:56 PM]

donc une seule méthode compute et pas de handle et update

fabienviale

[2:56 PM]

oui, par exemple

amine [2:56 PM]

reste à comprendre le timpstamp

fabienviale

[2:57 PM]

j'en ai oublié un

[2:57]

4) le nombre de noeuds qu'on a demandé de supprimer (removeNodes)

amine [2:58 PM]

oui oui

fabienviale

[2:58 PM]

le 3 et 4 doivent être mis à jour de façon asynchrone quand le noeud est effectivement créé/supprimé

[2:59]

Donc il faut que ce soit des variables de type AtomicInteger

[2:59]

ou volatile int

[2:59]

Mais le mieux c'est AtomicInteger

Il est peut-être malgré tout intéressant de faire les changements dans la LocalInfrastructure

[2:56]

C'est bien pour faire des tests

[2:56]

Il y a des chances que ça fasse ceci dit péter certains tests fonctionnels

[2:57]

mais ça permettrait de rajouter un test fonctionnel pour la SchedulerLoadingPolicy

amine [2:57 PM]

il faut peut être revoir ces tests fonctionnels

[2:58]

et les adapter

fabienviale

[2:58 PM]

on verra

amine [2:58 PM]

apres ton retour certainement

fabienviale

[2:59 PM]

oui certainement

[3:00]

En attendant, tu pourras tester avec différents paramètres (loadFactor, releaseDelay, threshold) pour voir si c'est ok

amine [4:19 PM]

pourrais tu m'envoyer le patch pour faire ces test ?

[4:24]

EC2Policy est inutile ?

[4:25]

Je peux juste supprimer cette ligne dans le fichier config/rm/nodesource/policies

[4:25]

org.ow2.proactive.scheduler.resourcemanager.nodesource.policy.EC2Policy

fabienviale

[4:26 PM]

oui je vais t'envoyer un patch où je supprime cette ligne

[4:26]

il faut juste que je nettoie un peu le code

amine [4:26 PM]

d'accord :slightly\_smiling\_face:

fabienviale

[5:17 PM]

voici le patch

[5:18]

En effet, lorsqu'une tâche est en train d'être répliquée, on se retrouve pendant un temps avec un nombre de tâches vivante = 0, ce qui déclenche la suppression de noeuds

amine [5:18 PM]

cool :))

fabienviale

[5:18 PM]

C'est un peu chiant, mais je sais pas bien comment contourner le problème, à part jouer avec les paramètres de la policy pour ne pas supprimer les noeuds trop vite

[5:20]

pour tester, je te conseille de mettre le paramètre pa.scheduler.task.fork=false dans config/scheduler/setting.ini

[5:20]

pour éviter de créer trop de VM

amine [5:20 PM]

okey

[5:20]

t'as fait les tests en utilisant des worflows ou une classe de test

fabienviale

[5:21 PM]

en utilisant un workflow replicate de 40 tâches

amine [5:21 PM]

okey

[5:21]

dans template il y ce genre de tâche je pense

fabienviale

[5:22 PM]

oui

[5:22]

il faudra rajouter après logger.info("Nodes removal request: " + numberToRemove);

[5:22]

un autre log avec le nombre de noeuds qui vont effectivement être supprimés

[5:23]

il faudra aussi ajouter dans la méthode toString, les autres configurations que j'ai rajoutées

amine [5:23 PM]

après l'appel de la méthode tu veux dire

fabienviale

[5:24 PM]

loadFactor, releaseDelay, threshold

amine [5:24 PM]

oui oui

fabienviale

[5:24 PM]

non pas après l'appel de la méthode à l'intérieur de la méthode, quand on a eu la réponse de chaque removeNode

amine [5:25 PM]

donc il faut rajouter un compteur non ?

[5:25]

pour les compter

fabienviale

[5:26 PM]

oui c'est ça

Fonctionnel Test : TestBrokenNodeSourceRemoval

oui de ce que j'ai compris le test utilises une SchedulerAwarePolicy, qui essaie de se connecter au scheduler. Comme c'est un test du rm et que le scheduler n'est pas démarré, ça ne fonctionne pas

Je ne capte pourquoi le test utilises une SchedulerAwarePolicy (peut-être pour la faire planter exprès) ?

oui en regardant le code du test, je vois que c'est bien ça

// the resource manager should try to recover the node source

// but it will fail because the scheduler does not exist

il est fait pour tester une node source qui pète au redémarrage

il n'y avait que la SchedulerAwarePolicy dans ce cas pète d'une façon bien précise, en n'acceptant pas les paramètres qui lui sont donnés pour la SchedulerAwarePolicy elle n'acceptait plus l'url du scheduler au redémarrage, car il n'existait plus

Mais c'était justement n'importe quoi, et j'ai corrigé le problème en lui permettant d'attendre que le scheduler démarre