Moulinette as a Queuing System

B. Dudin

On propose, dans cette activité, de s'attarder sur l'infrastructure de correction automatique de l'École, *la moulinette*. Notre objectif est d'en effectuer une analyse sous l'angle des systèmes d'attente. Les éléments qui vous sont présentés ne constituent qu'une simplification de la réalité, mais l'ensemble des questionnements mis à votre disposition relève de considérations ayant été au cœur de son développement et des mises à jour des infrastructures attenantes.

Pour référence, une *moulinette* est une infrastructure logicielle qui permet l'exécution de tests unitaires sur un code fourni par un tiers. Ces tests unitaires décrivent les spécifications fonctionnelles et techniques attendues pour le code fourni, et leur exécution permet de fournir un retour au tiers sur la conformité du code.

Le travail attendu dans le cadre de ce projet doit être effectué en groupe de quatre à cinq étudiants. Les groupes inter-majeures sont bien sûr autorisés. L'évaluation de votre travail se fera lors d'une soutenance de 30 minutes par groupe, évaluée à l'aide d'une grille critériée mise à votre disposition en annexe. Pour toute question, n'hésitez pas à utiliser le forum Moodle destiné aux échanges avec vos enseignants.

1 Livrables

Dans le cadre de ce projet, vous allez vous voir proposer un certain nombre de scénarios relevant des systèmes d'attente. Nous attendons de vous une analyse du comportement de ces systèmes d'attente. Plus précisément, vous devez fournir, pour chaque cas étudié :

1. le code permettant de les simuler ;
2. une analyse du comportement de chaque cas traité, contenant au minimum :
   (a) les paramètres en jeu ;
   (b) le comportement du système d'attente en fonction des paramètres qui le définissent, notamment en ce qui concerne sa stabilité ;
   (c) une évaluation du système au regard de métriques standard : nombre d'agents, temps de séjour, taux de blocage, etc. ;
   (d) une synthèse des résultats et des recommandations pour des plages de paramètres permettant un comportement acceptable et efficace, incluant une analyse des risques côté expérience utilisateur ;
3. les résultats bruts des simulations soutenant vos observations.

Nous souhaitons vous rappeler que les arguments que vous apporterez doivent être étayés par des éléments factuels et un travail de *benchmarking* donnant lieu à des statistiques. Une occurrence unique ou le résultat d'un seul test unitaire ne suffisent pas à appuyer une conclusion.

2 Terminologie

Voici quelques précisions sur la terminologie utilisée dans ce projet.

2.1 Qu'est-ce qu'un utilisateur ?

Dans ce contexte, un utilisateur est une personne ayant accès à l'infrastructure de correction dans le cadre d'une activité pédagogique spécifique. Cette personne peut effectuer deux actions :

- *pousser* son code sur l'infrastructure dans le cadre d'une mécanique standard de versionnage ;
- *pousser un tag* sur un commit pour déclencher l'exécution des *test-suites* associées à l'activité sur le code rendu, afin d'obtenir un retour sur sa conformité aux attentes.

2.2 Qu'est-ce qu'une moulinette ?

Une *moulinette* est constituée formellement de :

- une *test-suite* : un ensemble de tests unitaires, éventuellement stratifiés si l'on souhaite n'exécuter qu'une sous-partie des tests pour un focus particulier ;
- un niveau d'information de retour : quel niveau d'information met-on à disposition des étudiants ? L'erreur précise avec une aide spécifique ? Un message de non-conformité rejetant le code ?
- des ressources : nombre de *push tags* autorisés au total, par heure, ou dans des plages horaires définies.

Le concepteur d'une activité *moulinettée* fournit une suite de tests unitaires, spécifie le type de retour souhaité et détermine le nombre maximal de tags par étudiant, ainsi que l'action du système lorsque ce maximum est atteint.

2.3 Workflow nominal

Un étudiant code les réponses à des exercices (dans le cadre d'un atelier), des attendus (dans le cadre d'un projet) ou d'un TP (dans le cadre des enseignements pratiques) dans un *repository git* dédié. Dans le cadre de son travail, il peut adopter le *workflow git* de son choix et effectuer des *commits* et *push* sur les branches *remotes*, à condition de ne pas utiliser de *tags*.

Lorsqu'un *tag* est utilisé, une vérification est effectuée pour s'assurer qu'il correspond à un *tag* réservé. Si tel est le cas, la test-suite associée est exécutée selon le schéma du système d'attente (exécution immédiate, mise en attente dans une file dédiée, etc.). Le résultat de la test-suite est ensuite affiché à l'étudiant, selon les contraintes et le niveau d'information indiqué par les concepteurs de l'activité.

Les choix mentionnés dans le paragraphe précédent sont au cœur de l'analyse à réaliser ; quelles options de système d'attente pour quelles contraintes imposées.

3 Étude de cas

L'étude d'un modèle de *moulinettage* implique des choix dépendants du contexte dans lequel le système est déployé. Les cas suivants vous sont présentés par ordre croissant de complexité, c'est-à-dire en fonction du nombre de paramètres et de dimensions à prendre en compte. Pour chacun d'eux, une analyse de comportement est attendue.

3.1 Waterfall

Dans ce modèle, tout agent de la population suit le processus suivant :

1. Un *push tag* permet à l'étudiant de placer son code dans une file d'attente FIFO infinie pour l'exécution de la test-suite associée.
   - Un nombre K de serveurs est disponible pour cette tâche.
2. Une fois la test-suite exécutée, le résultat est placé dans une file d'attente FIFO infinie gérée par un serveur unique pour l'envoi vers le front.

1. Proposez un système d'attente modélisant le contexte décrit ci-dessus. Effectuez quelques simulations pour analyser son comportement selon les paramètres en jeu.
2. Avec une augmentation significative du nombre d'étudiants, l'hypothèse de files d'attente infinies devient irréaliste. On note k_s et k_f les tailles respectives des files d'exécution et de renvoi de résultats. Si un *push tag* est refusé, l'étudiant reçoit un message d'erreur. Si le résultat d'un moulinettage est refusé dans la seconde file, l'étudiant reçoit un retour vide.
   — Discutez des proportions de refus selon les paramètres.
3. Pour éviter la perte de données, un back-up des résultats de moulinettage est mis en place en amont de l'envoi vers la seconde file.
   — Quel changement cela opère-t-il sur la proportion de pages blanches ?
   — Quels problèmes peuvent surgir avec cette solution ?
   — Discutez des avantages d'un back-up aléatoire plutôt que systématique.
   — Calculez le temps de séjour moyen et la variance empirique dans ce modèle.

3.2 Channels and dams

On observe dans le modèle précédent que certaines populations d'étudiants ont des temps d'attente plus élevés que d'autres. En particulier, dans l'Atelier C, la population ING a des arrivées fréquentes, alors que la population PREPA, avec des rendus plus rares, occupe plus longtemps la moulinette.

1. Simulez les variations de temps de séjour par population décrites.
2. Pour réguler les moulinettages de la population ING, un blocage de la moulinette est introduit pour un temps t_b, puis ouvert pour $t_b/2$, etc.
   — Comparez ce modèle avec le précédent en termes de temps de séjour.
   — Proposez un autre système d'attente pour minimiser le temps de séjour moyen pour les deux populations.