RINGOT, Patrice

De: RINGOT, Patrice
Envoyé: jeudi 19 février 2015 17:10
À: PERRIN, Stanislas; NIEDERLENDER, Claude
Cc: 'SCHEFFER, Philippe (Philippe.SCHEFFER@inist.fr)'; LUC, Martial; TURRI, Angel (Angel.TURRI@inist.fr); VILLAUME, Michel
Objet: Compte rendu de réunion Gatling/API Istex - 19/2/15
Pièces jointes: searchbasic-1424353270906.zip

Bonjour,

En guise de compte-rendu de réunion, voici le test gatling présenté cet après-midi à Martial Luc et à Philippe Scheffer en salle G.

Les tests de perf Gatling sont des DSL écrits en Scala.

Ces DSL se comprennent assez bien (fluent api).

```
class SearchBasic extends Simulation {

  val httpConf = http // 1
    .baseURL("https://api.istex.fr")
    .proxy(
      Proxy("proxyout.inist.fr", 8080, 8080))
    .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
    .userAgentHeader("Mozilla/5.0 (Windows NT 5.1; rv:31.0) Gecko/20100101 Firefox/31.0")

  val scn = scenario("Istex-Api search") // 2
    .feed(csv("wordnet31.csv").random) // 3
    .exec(
      http("GlobalSearchWord") // 4
        .get("/document").queryParam("q", "${word}") // 5
        .check(status.is(200)))

  setUp(
    scn.inject(rampUsers(100) over (50  seconds))) // 6
    .protocols(httpConf)
}
```

1 // On définit les conditions d'accès du client http
2 // Définition d'un scenario. Plusieurs scenarios peuvent être menés simultanément lors d'une campagne de tir Gatling (ex : un humain qui utilise le service à son rythme, un logiciel qui consomme l'API de manière intensive). Un scénario ne se résume pas à une unique requête http, mais peut être constitué d'un enchainement de requêtes qui tiennent compte du contexte (on parle alors de session côté client, on peut extraire des données des résultats d'une requête, les mettre dans la session et les réutiliser pour la requête qui suit)
3 // wordnet31.csv est un fichier d'environ 70K mots qui commence par une ligne qui contient le mot « word ». Il va servir de gisement de mots aléatoire pour la requête qui va suivre
4 // La requête s'appelle GlobalSearchWord
5// On recherche un mot à partir d'un aléa fournit par wordnet31.csv (le "${word}" correspond au nom de la colonne du fichier – ce n'est pas forcément un fichier d'ailleurs)
6// La campagne de tir est définie comme suit : on prépare une rampe de 100 utilisateurs qui vont se présenter pour jouer le scénario sur une période de 50s, soit 2 utilisateurs par seconde (ce n'est pas violent comme test).

Ci-joint un zip qui contient le résultat de cette « campagne » de test. D'un zip à l'autre, si on prend soin de conserver un environnement de test assez stable, on peut juger des améliorations ou régressions de l'appli en termes de perf.

Ce scénario a été présenté / exécuté sous Scala IDE (une version d'Eclipse pré-packagée pour Scala). Bien pratique pour concevoir des scenarios.

Quand on commencera à faire des choses sérieuses, on utilisera plutôt en ligne de commande les scénarios conçus via l'IDE sur une machine située au cœur de l'infra. Et de plus, on attaquera directement l'API sans passer par le proxy ni par le revprox (http://vp-istex-api:53332).

Lors de la réunion, on a évoqué également le projet de monter une plateforme centrale autour de Graphite de manière à combiner en temps réel lors d'un tir Gatling dans un même graphique des métriques système et réseaux et ce qu'on obtient par Gatling (ex : temps de réponse pour 95% des requêtes). C'est du plus long terme, mais c'est très motivant (Voyages-Sncf.com utilise ces techniques pour leurs tests).

Patrice