Projet Données Réparties

Rapport Final

Paul Anaclet

Guohao Dai

Théo Desprats

2A SN

2022

1 Amélioration de Linda++

1.1 Structures de données

Afin de faciliter la localisation d'un `Tuple` dans l'espace de stockage, nous avons mis en place une `HashTable` de la forme suivante :

```
HashTable<Integer, ArrayList<Tuple>> tuples ;
```

Cette `HashTable` a comme clé un entier correspondant à la taille des `Tuples` (c.à.d le nombre d'éléments dans le `Tuple`, récupérable grâce à `size()`) contenus dans l'`ArrayList` associée. Cela permet une séparation des tuples et un accès plus rapide à ceux-ci.

1.2 Parallélisation

Nous avons transformé les primitives Linda (`take()`, `write()` ...) en tâches qui sont soumises à ce pool par le biais des `ExecutorServices`.

La synchronisation des accès aux `ArrayLists` de `Tuples` et aux `Callbacks` se fait sur le modèle Lecteurs/Rédacteurs équitable (avec un système de tours) : plusieurs Lecteurs sont autorisés à accéder en lecture à la structure de données et un Rédacteur n'est autorisé à écrire que s'il est seul.

1.3 Cache

Pour mettre en place un cache de tuples côté client qui réduira la charge des serveurs, nous comptons modifier la classe `LindaClient` en lui ajoutant une classe LindaClientCache contenant un attribut :

```
HashTable<Integer, ArrayList<Tuple>> tuplesCache ;
```

Cette `HashTable` fonctionnera comme celle décrite en section 1.1. On y ajoute des Tuples quand le client fait des write() ou qu'il reçoit des read() du serveur. En effet, quand le client fait un read(), il recherchera d'abord dans son cache et s'il ne trouve rien, il propagera sa requête au serveur. Pour les autres services de Linda comme take(), les requêtes vont directement sur le serveur.

En cas de suppression d'un `Tuple` dans l'espace, le serveur propage l'information aux clients grâce à une liste de callback contenant une méthode distante invalidate() qui enlèvera le Tuple supprimé du cache des clients.

1.4 Multi-serveurs

La partie multi-serveurs a directement été intégrée dans la classe `LindaServerImpl.java`, parent des diverses implémentations de serveurs Linda. Une liste des voisins est directement gérée dans cette classe et des serveurs voisins peuvent être ajoutés avec la méthode '`addNeighbor(LindaServer)`'.

Comme nous n'avons pas trop eu le temps de nous pencher sur cette partie, seules les requêtes Read et Take sont retransmises aux voisins en cas d'échec sur le serveur auquel le client est initialement connecté.

L'outil `LindaServerStarter` présenté dans la partie suivante, permet de déployer des serveurs interconnectés (ou non en ajoutant l'option `'-n'`) à partir d'un fichier de configuration.

2 Outils développés

Pour développer les outils présentés dans cette partie, nous nous sommes appuyés sur la librairie picocli qui permet de mettre rapidement en place un CLI simple et efficace. Ainsi, pour un bon fonctionnement du projet, le fichier `linda/dependencies/picocli-4.6.3.jar` doit être présent et ajouté aux bibliothèques référencées du classpath.

2.1 Linda Server Starter

`Linda Server Starter` est un programme permettant de démarrer un ou plusieurs serveurs Linda selon si un fichier de configuration est spécifié avec l'option `'-f'` comme visible sur la Figure ci-dessous.

```
Usage: server [-f=<fileName>] <servName> <servNetwork> <servType>
Starts one or more Linda server with specified configuration.
      <servName>          Server name
      <servNetwork>       Server address
      <servType>          Server type
  -f, --file=<fileName>   Starts servers from configuration file
```

Figure: Utilisation de la commande LindaServerStarter

Le programme vérifie les paramètres que l'utilisateur spécifie pour chaque nœud qu'il souhaite créer (nom, adresse et type de noyau du serveur), instancie possiblement un `RMI registry` si le port est libre et lance un `Thread` pour chacun de ces nœuds. Un exemple de fichier de configuration est visible dans la Figure suivante.

```
1   BasicServer1 localhost:4000 Basic
2   BasicServer2 localhost:4002 Basic
3   ParallelServer1 127.0.0.1:4003 Parallel
4   CacheServer1 127.0.0.1:4004 Cache
```

Figure: Exemple de fichier de configuration pour LindaServerStarter

2.2 Linda Performance Analyzer

L'analyseur de performances peut être vu comme une nouvelle application pour `Linda`. En effet, cet outil permet de tester une implémentation `Linda` en faisant varier les types d'accès et le nombre de requêtes et permet aussi d'obtenir une analyse affichant plusieurs mesures de métriques.

Plus précisément, cet objet Java est paramétré par plusieurs valeurs:

- `nbQueries`: nombre total de requêtes à effectuer
- `pTake`: pourcentage de requêtes tryTake à effectuer
- `pRead`: pourcentage de requêtes tryRead à effectuer
- `pWrite`: pourcentage de requêtes Write à effectuer
- `url`: URL du serveur voulant être testé (si null, démarre un `CentralizedLinda`)

Ainsi, 3 `Threads` clients (un par type de requête) sont créés et envoient leurs requêtes au `Linda` spécifié tout en effectuant des mesures avec des timers et des compteurs pour les métriques présentes dans la classe `'Metrics.java'`. Pour l'instant les métriques implémentées sont les suivantes:

- `totalDuration`: durée totale de l'exécution des Threads clients
- `memoryUsage`: mémoire utilisée par le programme
- `succesfullQueries`: nombre de requêtes 'try' retournant un `Tuple`
- `responseTime`: temps entre l'envoi d'une requête et la réception du `Tuple` réponse
- `throughput`: nombre de requêtes émises par seconde

En fin d'exécution et après nettoyage du serveur, on pourra obtenir un affichage similaire à la Figure suivante.

```
Sending Take queries...
Sending Read queries...
Sending Write queries...
Sending Read queries done.
Sending Take queries done.
Sending Write queries done.

--------------------------------------------------------------------------------
Linda Performance Analyzer
Sent 50000 queries in 5555,608ms (12500 Take / 12500 Read / 25000 Write)
--------------------------------------------------------------------------------
Memory usage            4,874% (97/1990 Mb)
Success (Read)          92,456% (11557/12500 q)
Success (Take)          90,304% (11288/12500 q)
Response time (Take)    avg=0,435 ms    min=0,061 ms    max=19,702 ms
Response time (Read)    avg=0,394 ms    min=0,060 ms    max=19,455 ms
Throughput (Take)       avg=2413 q/s    min=1591 q/s    max=6493 q/s
Throughput (Read)       avg=2553 q/s    min=1652 q/s    max=7164 q/s
Throughput (Write)      avg=4500 q/s    min=1451 q/s    max=13934 q/s



Cleaning server...
```

Figure: Exemple d'exécution de LindaPerformanceAnalyzer

En plus d'afficher ces données, `LindaPerformanceAnalyzer` peut générer des fichiers CSV de chacune de ces métriques si un répertoire de sortie lui est communiqué au démarrage de l'analyse. Ces fichiers CSV peuvent être visualisés avec le script python `'PlotOutput.py'` qui est automatiquement exécuté après l'analyse avec pour paramètre le répertoire de sortie.

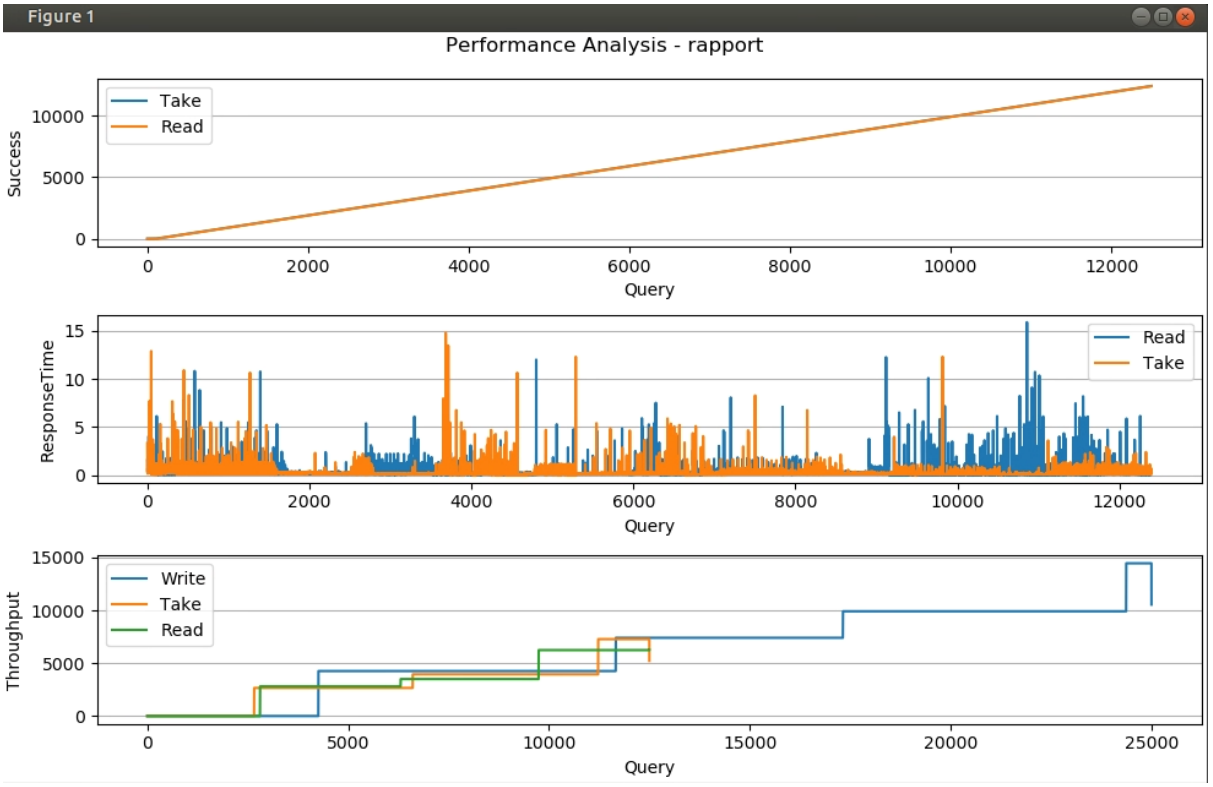

Figure: Exemple d'exécution de PlotOutput

***Note**: pour que l'affichage automatique ait bien lieu après l'exécution de l'analyse, les permissions doivent être accordées au script `'PlotOutput.py'` et les modules python3 `pandas` et `matplotlib` installés.*

2.3 Linda CLI

Dans une volonté d'instrumentation et d'accès aux informations des serveurs nous avons développé un `'Command Line Interface'` pour Linda. Ce CLI, écrit avec picocli, permet de se connecter à un serveur en fonctionnement et propose plusieurs commandes décrites dans la liste suivante :

- `help`: permet d'afficher les commandes disponibles ou l'utilisation d'une commande si elle est spécifiée
- `query`: permet d'envoyer des requêtes au serveur, soit de manière interactive (invite de commande) soit à partir d'un fichier contenant les requêtes souhaitées

```
Usage:  query [-i=<fileName>]
Sends queries interactively or from file to Linda server
  -i, --inputfile=<fileName>
          Run scenario in the specified file
```

Figure: Utilisation de la commande query

Cette commande peut être bloquante, effectue le parsing des Tuples grâce à la méthode statique valueOf de cette même classe et ignore les requêtes dont la syntaxe n'est pas correcte. Un exemple de fichier de requête est présenté dans la figure ci-dessous.

```
1   tryread [ "test" [ 1 2 ] ]
2   write [ "test" ]
3   write [ "test" [ 1 2 ] ]
4   tryread [ "test" [ 1 2 ] ]
5   take [ "test" ]
```

Figure: Exemple de fichier de requêtes pour la commande query

- `status`: permet d'afficher dynamiquement différentes informations du serveur et de son noyau Linda (type de noyau, temps de vie, nombre de tuples, de requêtes, états des Threads…)

```
CentralizedLinda2 (Parallel)
----------------------------
Time alive:  0:00:52
Tuples:      0
Queries:     TAKE     25003
             READ     12500
             WRITE    25000
Callbacks:   TAKE     0
             READ     0
Threads:     pool-1-thread-2      TIMED_WAITING
             pool-1-thread-9      TIMED_WAITING
             pool-1-thread-8      TIMED_WAITING
             pool-1-thread-4      TIMED_WAITING
             pool-1-thread-12     TIMED_WAITING
             pool-1-thread-5      TIMED_WAITING
```

Figure: Exemple d'exécution de la commande status

- `perf`: intégration du `LindaPerformanceAnalyzer` (cf. partie 2.2)

```
Usage:  perf [-o=<outputFile>] <nbQueries> <pTake> <pRead> <pWrite>
Runs a performance analysis of the server.
      <nbQueries>   Number of queries to send
      <pTake>       Percentage of Take queries
      <pRead>       Percentage of Read queries
      <pWrite>      Percentage of Write queries
  -o, --output=<outputFile>
                    Output analysis report as CSV in specified directory
```

Figure: Utilisation de la commande perf

- `exit`: quitte le CLI

2.4 Makefile

Pour faciliter le développement et l'utilisation des différents outils du projet nous avons mis en place un `Makefile` proposant différentes règles dont nous conseillons l'utilisation :

- `make server` (équivalent à `make server ARGS="LindaServer localhost:4000 Basic"`)
- `make cli URL="//localhost:4000/LindaServer"`
- `make clean`
- `make build`

3 Etude expérimentale

3.1 Présentation de l'études

Afin de pouvoir comparer la version de Linda et Linda++ (en centralisé et serveur), nous allons utiliser le Linda performance analyzer décrit plus tôt.

Nous allons tester différentes répartitions de pourcentage de read, take et write pour des nombres de requêtes qui varient (50000,100000,500000, …)

On s'attend à avoir des améliorations, surtout quand la proportion de read sera élevée, car c'est la situation la plus favorable pour voir l'apport de vitesse du cache.

Les mesures sont réalisées sur les machines de l'enseignement en accès à distance. Cela peut avoir des conséquences sur les mesures et la mémoire utilisée.

3.2 Paramètres et critères

Nos mesures des performances s'effectuent selon les critères suivants :

- temps d'exécution : temps écoulé entre l'appel et le retour d'une primitive depuis le client

- utilisation de l'espace mémoire : coût de l'exécution

- fiabilité / résilience : taux d'erreurs, comportement en cas de coupure d'un ou plusieurs serveurs, d'impossibilité de communication serveur/serveur ou serveur/client...

3.2 Résultats et analyse

3.2.1 Noyaux Lindas

Pour 50 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
Linda	7 Mb	0.006ms	0.003ms	18 150 r/s	18 150 r/s	18 150 r/s

Linda ++ (Séquentiel)	12 Mb	0.009ms	0.009ms	16 500 r/s	16 500 r/s	17 000 r/s
Linda ++ (Parallèle)						

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit moyen requête take	débit moyen requête read	débit moyen requête write
Linda	7 Mb	0.006ms	0.002ms	10 000 r/s	30 000 r/s	10 000 r/s
Linda ++	12 Mb	0.017ms	0.005ms	10 000 r/s	30 000 r/s	10 000 r/s

Pour 100 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
Linda	14 Mb	0.005ms	0.003ms	33 000 r/s	33 000 r/s	34 000 r/s
Linda ++	16 Mb	0.011ms	0.011ms	33 000 r/s	33 000 r/s	34 000 r/s

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit moyen requête take	débit moyen requête read	débit moyen requête write
Linda	13 Mb	0.026ms	0.001ms	20 000 r/s	60 000 r/s	20 000 r/s
Linda ++	14 Mb	0.016ms	0.002ms	20 000 r/s	60 000 r/s	20 000 r/s

Pour 500 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
Linda	36 Mb	0.010ms	0.002ms	103 386 r/s	165 000 r/s	170 000 r/s
Linda ++	33 Mb	0.012ms	0.001ms	79 925 r/s	165 000 r/s	157 719 r/s

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit moyen requête take	débit moyen requête read	débit moyen requête write
Linda	39 Mb	0.007ms	0.001ms	100 000 r/s	300 000 r/s	100 000 r/s
Linda ++	40 Mb	0.012ms	0.002ms	100 000 r/s	300 000 r/s	100 000 r/s

3.2.2 Serveurs

Pour 50 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	28 Mb	0.369ms	0.375ms	2 671 r/s	2 628 r/s	3 388 r/s
LindaServer (Linda ++)	9 Mb	0.344ms	0.326ms	2 872 r/s	3 023 r/s	3 281 r/s
LindaCachedServer (Linda++)	52 Mb	0.280ms	0.282ms	3 540 r/s	3 478 r/s	3 765 r/s

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	54 Mb	0.351ms	0.180ms	2 805 r/s	5 455 r/s	3 873 r/s
LindaServer (Linda ++)	14 Mb	0.172ms	0.100ms	5 938 r/s	9 778 r/s	6 625 r/s
LindaCachedServer (Linda++)	16 Mb	0.137ms	0.083ms	7 228 r/s	11 811 r/s	10000 r/s

Pour 100 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	100 Mb	0.251ms	0.246ms	3 943 r/s	4 036 r/s	4 503 r/s
LindaServer (Linda	66 Mb	0.120ms	0.117ms	8 252 r/s	8 490 r/s	10 130 r/s

++)						
LindaCachedServer (Linda++)	116 Mb	0.111ms	0.116ms	8 941 r/s	8 609 r/s	10 543 r/s

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	95 Mb	0.207ms	0.113ms	4 711 r/s	8 649 r/s	7 629 r/s
LindaServer (Linda ++)	62 Mb	0.119ms	0.112ms	8 310 r/s	14 732 r/s	12 169 r/s
LindaCachedServer (Linda++)	243 Mb	0.106ms	0.077ms	9 429 r/s	13 014 r/s	14 034 r/s

Pour 500 000 requêtes:

- Répartition des requêtes: 33% read , 33% take, 34% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	153 Mb	0.132 ms	0.127 ms	7 490 r/s	7 760 r/s	9 986 r/s
LindaServer (Linda ++)	214 Mb	0.126ms	0.117ms	7 883 r/s	8 522 r/s	9 814 r/s
LindaCachedServer (Linda++)	81 Mb	0.117ms	0.108ms	8 482 r/s	9 155 r/s	10 767 r/s

- Répartition des requêtes: 60% read , 20% take, 20% write

	Usage mémoire	avg temps réponse take	avg temps réponse read	débit requête take	débit requête read	débit requête write
LindaServer (Linda)	87 Mb	0.172 ms	0.097 ms	5 748 r/s	10 158 r/s	6 510 r/s
LindaServer (Linda ++)	295 Mb	0.101ms	0.096ms	9 868 r/s	16 203 r/s	12 138 r/s
LindaCachedServer (Linda++)	189 Mb	0.151ms	0.094ms	6 536 r/s	10 562 r/s	7 403 r/s

Nous pouvons voir que pour les versions centralisées, le Linda du S7 est plus performant, ça peut s'expliquer par le fait que l'implémentation du s7 était moins complexe ce qui est un avantage en centralisé.

Pour les versions serveur, nous pouvons voir une amélioration entre le LindaServer du S7 et le LindaServer ++ . Il y a une amélioration encore plus grande pour le LindaCachedServer.

Comme prévu, plus le pourcentage de read est élevé plus les choix d'amélioration pour le Linda ++ ont de l'impact.