Κατανεμημένα Συστήματα, NTUA 2017 - 2018

Αναφορά εξαμηνιαίας εργασίας

Ομάδα 34

Γαβαλάς Νίκος	03113121
Καραμέρης Μάρκος	03113148

Περιγραφή

Το ζητούμενο είναι η κατασκευή ενός group messenger, αποτελούμενο από clients (υπεύθυνοι για τη διαχείριση των μηνυμάτων των groups στα οποία ανήκουν) και έναν tracker, που κρατάει μια εικόνα για την κατάσταση των groups και απαντάει σε αιτήματα των clients, όταν αυτοί θέλουν να βρουν/δημιουργήσουν groups και άλλους clients-peers.

Απαιτήσεις

1. Η επικοινωνία με τον tracker πρέπει να είναι reliable. Η επικοινωνία μεταξύ των clients σε κάθε group είναι fault tolerant (δηλ. και να χαθεί κάποιο μήνυμα δεν μας πειράζει τόσο).

2. Ο κάθε client πρέπει να λαμβάνει τα μηνύματα με τη σειρά που στέλνονται προς αυτόν.

3. Πρέπει επίσης να υπάρχει η δυνατότητα ολικής ορθής διάταξης των μηνυμάτων στο group.

4. Ο tracker πρέπει να είναι σε θέση να ανιχνεύει clients που έχουν

αποτύχει και να τους αφαιρεί από τις δομές του.

Υλοποίηση

Σε αντιστοιχία με τα άνωθι:

1. Για reliable επικοινωνία με τον tracker χρησιμοποιούνται TCP sockets. Για τα μηνύματα μεταξύ των clients, UDP sockets.

2. Γίνεται χρήση FIFO ordering, με βάση την οποία ο κάθε client κρατάει έναν buffer για κάθε group στο οποίο ανήκει, και έναν counter για τα μηνύματα του ίδιου group. Παράλληλα, κάθε μήνυμα που κάνει broadcast στο group, το μαρκάρει με τον counter του. Ομοίως και οι υπόλοιποι clients του ίδιου group. Έτσι, για να δεχτούν μήνυμα ελέγχουν αν με βάση τον counter τους (και τον counter του μηνύματος) είναι αυτό που αναμένουν, οπότε και το δέχονται και αυξάνουν τον group counter τους για τον συγκεκριμένο client απ' όπου ήρθε το μήνυμα, ή το βάζουν στον buffer τους και περιμένουν πρώτα τα χαμένα ενδιάμεσα μηνύματα, για να τα παραδώσουν ύστερα με τη σωστή σειρά στην εφαρμογή.

3. Για ολική διάταξη χρησιμοποιείται ένας "ειδικός" client, ένας sequencer, που χρειάζεται να ξέρει μόνο ο tracker. Ο sequencer δέχεται μηνύματα από τους clients, τα διατάσσει κατά FIFO και τα επαναπροωθεί στο group (και ύστερα κάθε client χρησιμοποιεί FIFO για τον εαυτό του).

4. Κάθε client κάνει χρήση ενός background thread που ανά συγκεκριμένο χρονικό διάστημα (δεδομένο στον tracker) στέλνει ένα UDP πακέτο (heartbeat). Αν ο tracker δεν λάβει τέτοιο πακέτο για κάποιον cient για διάστημα μεγαλύτερο του διπλάσιου αυτής της δεδομένης διάρκειας, θεωρεί ότι απέτυχε και τον διαγράφει από τις δομές του.

Ως γλώσσα επιλέχθηκε η Python (v.3.5). Τα μηνύματα στα sockets

ανταλάσσονται με τη μορφή json (θα μπορούσε να χρησιμοποιηθεί pickle αλλά τα jsons είναι πιο απλά και υποστηρίζονται και από άλλες γλώσσες).

Σχεδιαστικές Αποφάσεις

- Για αποδοτική χρήση I/O αντί για πολλαπλά threads στα input streams (socket και stdin) (που εισάγουν overhead και είναι γενικά βαριά) χρησιμοποίησαμε το unix syscall "select", που τίθεται να παρακολουθεί streams και "ειδοποιεί" όταν κάποιο από αυτά έχει δεδομένα και είναι ready.
- Για την αντιμετώπιση της "άγνοιας" της κατάστασης του group σε περίπτωση που κάποιος εισέρχεται (και οι άλλοι πρέπει να τον μάθουν χωρίς να ρωτάνε συνέχεια τον tracker), ή αντίστοιχα εξέρχεται (οπότε και πρέπει να σταματήσουν να τον συμπεριλαμβάνουν στα broadcasts τους), χρησιμοποιήσαμε control messages μεταξύ των clients (πέρα από τα application messages). Συγκεκριμένα, όταν ένας client εισέρχεται σε ένα group, στέλνει σε όλους (που ξέρει ποιοι είναι γιατί του έχει πει ο tracker), ένα ειδικό μήνυμα client hello και αντίστοιχα όταν βγαίνει ένα client bye. Τα μηνύματα αυτά τα χειρίζονται οι clients για να ενημερώσουν τις δομές τους.
- Για την σωστή λειτουργία του FIFO ordering σε κάθε group, απαιτείται σωστή αρχικοποίηση του αντίστοιχου counter. Με αυτό εννοούμε πώς, μπαίνοντας σε ένα group, ένας client, δεν θέλουμε να έχει αρχικοποιημένο τον counter του στο 0, γιατί τότε θα περιμένει μήνυμα με counter 1, το οποίο (αν όσοι είναι group έχουν ήδη ανταλλάξει μερικά μηνύματα) δεν έρθει ποτέ, με αποτέλεσμα ο εν λόγω client να βάζει στον buffer του στο εξής όλα τα μηνύματα που του έρχονται. Επομένως κάναμε την σύμβαση, με το που εισέρχεται στο group, να αρχικοποιεί τον counter του στο πρώτο μήνυμα που λαμβάνει για κάθε client. Με αυτο τον τρόπο "συγχρονίζεται" στο group και μπορεί να επικοινωνήσει κανονικά στο εξής.

Πειράματα

Για τα χρονικά διαστήματα στα παρακάτω πειράματα χρησιμοποιήθηκε το system time. Κανονικά κάτι τέτοιο δεν είναι πολύ ακριβές, αλλά επειδή η υποδομή του εργαστηρίου που μας δόθηκε αποτελείται από 5 hosts που βρίσκονται σε Docker containers, και το Docker χρησιμοποιεί το ρολόι του host συστήματος, αυτό σημαίνει ότι όλοι οι hosts έχουν ακριβώς το ίδιο ρολόι. Οπότε οι μετρήσεις για διαστήματα είναι reliable χωρίς να χρειαστεί να συγχρονίζουμε ρολόγια κλπ.

Deployment

```
$ python3 tests.py -h
usage: tests.py [-h] [-t] [-c] n p test_num

positional arguments:
  n           number of clients
  p           base port
  test_num    1 for first test, 2 for second

optional arguments:
  -h, --help  show this help message and exit
  -t          use total ordering (sequencer)
  -c          close all remote processes
```

Πείραμα 1: Απόδοση του συστήματος

a) FIFO Ordering

```
$python3 tests.py 5 46663 1
```

client0

start	1520864763.220646
end	1520864763.239089
elapsed	0.018443

message	from	at
10e	client4	1520864763.802138
10d	client3	1520864763.317927
10a	client0	1520864763.238586
10c	client2	1520864763.233414
10b	client1	1520864763.195053

client1

start	1520864763.185815
end	1520864763.195127
elapsed	0.009312

message	from	at
10e	client4	1520864763.801671
10d	client3	1520864763.317673
10a	client0	1520864763.237956
10c	client2	1520864763.232598

10b	client1	1520864763.194553

client2

start	1520864763.224812
end	1520864763.232978
elapsed	0.008166

message	**from**	**at**
10e	client4	1520864763.801829
10d	client3	1520864763.317707
10a	client0	1520864763.238106
10c	client2	1520864763.232475
10b	client1	1520864763.194957

client3

start	1520864763.308182
end	1520864763.318171
elapsed	0.009989

message	**from**	**at**
10e	client4	1520864763.801995
10d	client3	1520864763.317729
10a	client0	1520864763.238746
10c	client2	1520864763.232818
10b	client1	1520864763.195013

client4

start	1520864763.787793
end	1520864763.801837
elapsed	0.014044

message	from	at
10e	client4	1520864763.801871
1e	client4	1520864763.791658
10a	client0	1520864763.238704
10c	client2	1520864763.232801
10b	client1	1520864763.195222

b) FIFO & Total Ordering

```
$python3 tests.py -t 5 46663 1
```

client0

start:	1520877999.384225
end:	1520877999.396608
elapsed:	0.012383

message	from	at
10b	client1	1520877999.385221
10a	client0	1520877999.395910

10c	client2	1520877999.455505
10d	client3	1520877999.473458
10e	client4	1520877999.476985

client1

start:	1520877999.366537
end:	1520877999.381957
elapsed:	0.015420

message	**from**	**at**
10b	client1	1520877999.380185
10a	client0	1520877999.390893
10c	client2	1520877999.450411
10d	client3	1520877999.468378
10e	client4	1520877999.471787

client2

start:	1520877999.438028
end:	1520877999.448506
elapsed:	0.010478

message	**from**	**at**
10b	client1	1520877999.382484
10a	client0	1520877999.393192
10c	client2	1520877999.452803

10d	client3	1520877999.470711
10e	client4	1520877999.474160

client3

start:	1520877999.438009
end:	1520877999.448662
elapsed:	0.010653

message	**from**	**at**
10b	client1	1520877999.381281
10a	client0	1520877999.391977
10c	client2	1520877999.451603
10d	client3	1520877999.469470
10e	client4	1520877999.472928

client4

start:	1520877999.441991
end:	1520877999.458041
elapsed:	0.016050

message	**from**	**at**
10b	client1	1520877999.378629
10a	client0	1520877999.389373
10c	client2	1520877999.448982
10d	client3	1520877999.466878

```
10e          client4    1520877999.470432
```

Πείραμα 2: Κλιμακωσιμότητα του συστήματος

a) FIFO Ordering

```
python3 tests.py 2 46663 2
```

```
python3 tests.py 4 46663 2
```

```
python3 tests.py 8 46663 2
```

```
python3 tests.py 16 46663 2
```

b) FIFO & Total Ordering

```
python3 tests.py -t 2 46663 2
```

```
python3 tests.py -t 4 46663 2
```

```
python3 tests.py -t 8 46663 2
```

```
python3 tests.py -t 16 46663 2
```