ALGORITMI PARALELI ȘI DISTRIBUIȚI
Tema #3 Protocolul BitTorrent

Responsabili: Radu-Ioan Ciobanu, Cosmin Lovin, Alex Deonise, Dorian Verna, Theodor Oprea, Alex Tudor, Andreea Stan, Robert-Mihai Adam, Andrei Boruga, Mihail Ungureanu, Alex Bala

Termen de predare: 14-01-2024 23:59 (hard)
Ultima modificare: 14-12-2023 20:20

Cuprins

Cerință

Scopul acestei teme este simularea protocolului BitTorrent de partajare peer-to-peer de fișiere, folosind MPI.

Protocolul BitTorrent

BitTorrent este un protocol de comunicație pentru partajarea de fișiere peer-to-peer (P2P), care permite utilizatorilor să distribuie date și fișiere electronice pe Internet într-un mod descentralizat. Pentru a trimite sau primi fișiere, utilizatorii folosesc un client BitTorrent pe un calculator conectat la Internet și interacționează cu un **tracker**. Tracker-ele BitTorrent furnizează o listă de fișiere disponibile pentru transfer și permit clientului să găsească utilizatori de la care pot transfera fișierele. Descărcarea prin BitTorrent este considerată mai rapidă decât HTTP și FTP datorită lipsei unui server central care ar putea limita lățimea de bandă.

În loc să descarce un fișier de la un singur server sursă, protocolul BitTorrent permite utilizatorilor să se alăture unui **swarm** de noduri pentru a încărca și descărca simultan unul de la altul. Fișierul distribuit este împărțit în segmente. Pe măsură ce fiecare nod primește un nou segment al fișierului, devine o sursă (a acelui segment) pentru alți clienți, eliberând sursa originală de la a trimite acel segment către fiecare utilizator care dorește o copie. Cu BitTorrent, sarcina de distribuire a fișierului este împărțită de cei care îl doresc. Astfel, este posibil ca sursa să trimită doar o singură copie a fișierului, dar în cele din urmă să distribuie către un număr nelimitat de clienți. Fiecare segment este protejat de un hash criptografic. Acest lucru asigură că orice modificare a segmentului poate fi detectată ușor, prevenind astfel modificările accidentale și malițioase.

De obicei, segmentele sunt descărcate într-o ordine non-secvențială și sunt rearanjate în ordinea corectă de către clientul BitTorrent, care monitorizează ce segmente îi trebuie, respectiv ce segmente le deține și le poate trimite către alte gazde. Segmentele au aceeași dimensiune pe parcursul unui singur download. Datorită naturii acestui mod de abordare, descărcarea oricărui fișier poate fi întreruptă în orice moment și reluată la o dată ulterioară, fără pierderea informațiilor descărcate anterior, ceea ce face BitTorrent util în transferul de fișiere mari. Acest lucru permite de asemenea clientului să caute segmente disponibile la un moment dat de timp și să le descarce imediat, în loc să oprească descărcarea și să aștepte următorul segment (care ar putea să nu fie disponibil).

Roluri în BitTorrent

În general, în cadrul protocolului BitTorrent, un client poate avea unul din trei roluri:

- **seed** → deține un fișier în totalitate și îl poate partaja cu alți clienți interesați; se ocupă doar cu upload-ul segmentelor unui fișier către alți clienți
- **peer** → deține segmente dintr-un fișier de care este interesat; se ocupă deci atât cu upload-ul, cât și cu download-ul acelui fișier (partajează segmentele pe care le deține, preia de la alți clienți segmentele care îi lipsesc)
- **leecher** → fie nu deține niciun segment dintr-un fișier pe care dorește să îl descarce, fie deține segmente pe care nu le partajează; se ocupă deci doar cu descărcarea de segmente dintr-un fișier de la alți clienți.

Rolurile de mai sus sunt specifice pentru fiecare fișier pe care un client îl partajează sau îl dorește. Astfel, un client poate în același timp să fie seed pentru un fișier și peer pentru altul.

Tracker-ul BitTorrent

Un tracker BitTorrent este un tip special de server care asistă în comunicarea dintre utilizatori folosind protocolul BitTorrent. El ține evidența clienților care dețin segmente dintr-un fișier, precum și a segmentelor pe care le deține fiecare, ajutând astfel la coordonarea transmiterii și reasamblării eficiente a fișierului copiat.

Clienții care au început deja să descarce un fișier comunică periodic cu tracker-ul pentru a negocia o transferare mai rapidă a fișierului cu noi utilizatori și pentru a furniza statistici privind performanța rețelei. Totuși, după ce descărcarea inițială a fișierului este începută, comunicarea peer-to-peer poate continua fără o conexiune la un tracker.

Astfel, tracker-ul are rol de intermediar. El nu deține niciun segment din niciun fișier și nu contribuie la procesul de partajare, ci doar menține liste actualizate cu swarm-urile fișierelor de care este responsabil și cu locațiile segmentelor acestora.

Detalii de implementare

Scopul vostru este de a simula, folosind MPI, logica protocolului BitTorrent. În repository-ul temei, găsiți în directorul **src** scheletul temei, de la care puteți porni.

Rulare

Compilarea temei va rezulta într-un excutabil numit *tema3*, care se va rula în felul următor:

```
$ mpirun -np <N> ./tema3
```

N este numărul de task-uri MPI care se vor porni, și va fi întotdeauna mai mare sau egal cu 3. Task-ul cu rangul 0 va juca rolul de tracker, iar toate celelalte task-uri vor avea rol de clienți.

Fișiere de intrare

Fiecare client va avea câte un fișier de intrare denumit *in<R>.txt* (unde R este rangul task-ului), cu următorul format:

```
numar_fisiere_detinute
nume_fisier_detinut_1 numar_segmente_fisier_detinut_1
segment1_fisier_detinut_1
segment2_fisier_detinut_1
...
nume_fisier_detinut_2 numar_segmente_fisier_detinut_2
segment1_fisier_detinut_2
segment2_fisier_detinut_2
...
numar_fisiere_dorite
nume_fisier_dorit_1
nume_fisier_dorit_2
...
```

Astfel, primul rând conține numărul de fișiere deținute de acest client la începutul rulării programului, urmat de detalii despre fiecare fișier în parte. Pentru un fișier, primul rând de detalii conține numele și numărul de segmente, iar rândurile următoare reprezintă hash-uri (de 32 de caractere) ale segmentelor fișierului, în ordine (astfel, primul rând după numele și dimensiunea fișierul va conține hash-ul primului segment din fișier, și așa mai departe).

După informațiile despre fișierele deținute de un client, urmează un rând cu numărul de fișiere pe care clientul dorește să le descarce folosind BitTorrent, urmat de numele fișierelor dorite (câte unul pe rând).

Un exemplu concret de fișier de intrare (unde clientul deține fișierele *file1* și *file2*, și dorește fișierul *file3*) poate fi observat mai jos:

```
2
file1 3
3fcfb9d1242fdce64aee2bfe35266912
6dd6078d720fc86c885f7f87faf0d32a
b56195fb830b234e8e35acaabc399400
file2 2
0f2ab6f4eab22e6b061f99daa48dd74e
f70fee606c4e57add77b3773fed6622b
1
file3
```

Clienții

Așa cum s-a menționat anterior, clienții sunt toate task-urile MPI cu rangul mai mare decât 0. După cum se poate observa în scheletul din repository-ul temei, un client are două fire de execuție (implementate cu Pthreads), astfel:

- **download_thread** → acest fir de execuție este folosit de client pentru a implementa logica de descărcare de fișiere de la seeds sau peers și pentru a realiza comunicația cu tracker-ul
- **upload_thread** → acest fir de execuție este folosit pentru a răspunde la cereri de segmente pe care acest client le deține, venite din partea altor clienți.

Inițializare. La inițializare, un client realizează următorii pași:

1. citește fișierul de intrare
2. îi spune tracker-ului ce fișiere are (cu alte cuvinte, pentru ce fișiere este seed)
   - pentru fiecare fișier, clientul îi transmite tracker-ului hash-ul fiecărui segment, în ordine
   - astfel, tracker-ul va ști care sunt fișierele din rețea, ce segmente au și în ce ordine sunt, și cine le deține inițial
3. așteaptă un răspuns de la tracker că poate începe căutarea și descărcarea fișierelor de care este interesat.

Descărcare. Odată ce a primit confirmarea de la tracker că poate continua, clientul realizează următorii pași:

1. îi cere tracker-ului lista de seeds/peers pentru fișierele pe care le dorește, precum și segmentele pe care le deține fiecare
2. se uită la segmentele care îi lipsesc din fișierele pe care le dorește, caută în lista de la tracker, și începe să trimită cereri către peers/seeds
3. pentru fiecare segment dintr-un fișier dorit pe care nu îl deține, un client realizează următorii pași:
   (a) alege un seed/peer care deține segmentul căutat
   (b) îi trimite acelui seed/peer o cerere pentru segment
   (c) așteaptă să primească de la seed/peer segmentul cerut
   (d) marchează segmentul ca primit

Atenție! Pentru a primi cele 10 puncte pe eficiență (vedeți secțiunea de notare de mai jos), trebuie ca un client să varieze cât de mult posibil nodurile seed/peer de la care descarcă segmentele căutate. Cu alte cuvinte, nu se dorește ca descărcarea unui fișier să se facă în totalitate de la același seed/peer.

Atenție! Deoarece nu se lucrează cu fișiere propriu-zise (ci doar cu hash-uri ale segmentelor dintr-un fișier), se va simula descărcarea unui segment de fișier de la un seed/peer. Astfel, este suficient ca un client să trimită un mesaj către seed/peer cu hash-ul segmentului dorit, iar seed-ul/peer-ul sa-i răspundă cu un mesaj simplu de genul "ACK" sau "OK" (este la latitudinea voastră exact cum abordați acest lucru).

Actualizare. Pentru a informa restul clienților din rețea despre ce segmente a descărcat, un client trebuie să informeze tracker-ul despre starea sa curentă. Astfel, pentru această temă, un client execută următorii pași după fiecare 10 segmente descărcate de la seeds/peers:

1. informează tracker-ul despre segmentele pe care le deține
2. reia pașii de la secțiunea de **Descărcare**.

Primire de mesaje de la alți clienți. Atunci când primește un mesaj de la un peer, un client îi "trimite" acestuia segmentul cerut (așa cum s-a specificat mai sus, acest lucru se poate simula prin trimiterea un mesaj de tip "ACK" sau "OK").

Finalizare descărcare fișier. Atunci când termină de descărcat toate segmentele unui fișier, un client realizează următorii pași:

1. informează tracker-ul că are tot fișierul
2. salvează fișierul (lista de hash-uri în ordine) într-un fișier de ieșire numit *client*<*R*>_<*NUMEFISIER*> (unde R este rangul clientului, iar $NUMEFISIER$ este numele original al fișierului descărcat); un exemplu de nume pentru fișierul *file1* descărcat de clientul cu rangul 2 este *client2_file1*.

Finalizare descărcare toate fișierele. Atunci când termină de descărcat toate fișierele pe care le dorea, un client realizează următorii pași:

1. îi trimite tracker-ului un mesaj prin care îl informează ca a terminat toate descărcările
2. închide firul de execuție de download (**download_thread**), dar îl lasă deschis pe cel de upload (**upload_thread**), pentru că va putea acționa în continuare ca seed pentru fișierele pe care le avea și pentru cele pe care le-a descărcat.

Finalizare pentru toți clienții. Atunci când un client primește mesaj de la tracker că toți clienții și-au terminat de descărcat fișierele, realizează următorii pași:

1. închide firul de execuție de upload (**upload_thread**)
2. se închide cu totul.

Tracker-ul

Tracker-ul menține o listă de fișiere și swarm-ul asociat fiecăruia dintre ele. Swarm-ul unui fișier conține lista de clienți (seeds sau peers) care au segmente din fișierul respectiv, precum și ce segmente are fiecare client. Un segment este definit de un hash și de poziția sa în fișier.

Inițializare. La inițializare, tracker-ul realizează următorii pași:

1. așteaptă mesajul inițial al fiecărui client, care va conține lista de fișiere deținute
2. pentru fiecare fișier deținut de un client, trece clientul respectiv în swarm-ul fișierului

3. când a primit de la toți clienții, răspunde fiecăruia cu câte un "ACK" (semn că poate începe comunicația între clienți pentru a se descărca fișierele dorite).

Primire de mesaje de la clienți. Atunci când primește un mesaj de la un client, tracker-ul realizează următorii pași:

1. se uită ce fel de mesaj a primit
2. în funcție de tipul mesajului, realizează una din următoarele acțiuni:
   - dacă mesajul primit este o cerere de la un client pentru un fișier, tracker-ul îi dă acestuia lista cu clienții seeds/peers din swarm, precum și ce segmente din fișier are fiecare
   - dacă mesajul primit este o actualizare de la un client, tracker-ul trece clientul respectiv în swarm-ul fișierelor menționate de client (dacă nu era deja) și actualizează lista de segmente deținute de acest client; de asemenea, îi răspunde clientului cu aceleași informații din punctul precedent
   - dacă mesajul primit este de finalizare a unei descărcări de fișier, tracker-ul marchează clientul respectiv ca seed
   - dacă mesajul primit este de finalizare a descărcării tuturor fișierelor de către un client, tracker-ul marchează clientul respectiv ca terminat, dar îl ține în swarm-urile fișierelor pe care le deține (adică cele pe care le avea inițial sau pe care le-a descărcat de la alți clienți)
   - dacă mesajul primit semnifică faptul că toți clienții au terminat de descărcat toate fișierele dorite, tracker-ul trimite câte un mesaj de finalizare către fiecare client, pentru a le spune să se oprească, după care se oprește și el.

Notare

Tema se poate testa local sau pe VMChecker, pe modelul temelor 1 și 2, și după cum se explică mai jos. Tema se va încărca pe Moodle sub forma unei arhive Zip care, pe lângă fișierele sursă, va trebui să conțină următoarele două fișiere **în rădăcina arhivei**:

- *Makefile* - cu directiva *build* care compilează tema voastră și generează un executabil numit *tema3* aflat în rădăcina arhivei
- *README* - fișier text în care să se descrie pe scurt implementarea temei.

Punctajul este divizat după cum urmează:

- **30p** - respectarea pașilor protocolului BitTorrent descriși în enunțul temei
- **40p** - descărcarea corectă la clienți a tuturor fișierelor dorite (acest punctaj este oferit de checker-ul automat și este condiționat de respectarea pașilor protocolului)
- **10p** - eficiența transferului de fișiere
- **20p** - claritatea codului și a explicațiilor din README.

Nerespectarea următoarelor cerințe va duce la depunctări:

- **-100p** - nepăstrarea structurii din schelet pentru clienți (cu alte cuvinte, trebuie să păstrați cele două thread-uri de download și upload; puteți modifica fișierul din schelet, dar logica trebuie să rămână aceeași)
- **-100p** - descărcarea de fișiere direct de la tracker.

Atenție! Checker-ul vă dă doar cele 40 de puncte pentru descărcarea corectă a fișierelor de către clienți. Punctajul pe restul cerințelor (respectarea protocolului, eficiența trasferului, claritatea codului și a explicațiilor) va fi dat la corectarea manuală.

Atenție! Pentru cele 10 puncte alocate eficienței transferului, trebuie să încercați să balansați cât mai bine încărcarea pe clienți. Cu alte cuvinte, dacă există mai mulți clienți care au un segment, se va încerca descărcarea acestuia de la un client mai puțin aglomerat.

Testare

Pentru a vă putea testa tema local, găsiți în repository-ul temei un set de fișiere de intrare de test, precum și un script Bash (numit *local.sh*) pe care îl puteți rula pentru a vă verifica implementarea. Rularea locală în acest mod necesită existența Docker pe sistemul vostru. Script-ul din repository va fi folosit și pentru testarea automată pe VMChecker Next, în fix aceleași condiții[1]. Scriptul de testare locală se rulează astfel:

```
$ ./local.sh checker
```

Atenție! Dacă aveți un calculator Apple cu procesor ARM (ex. M1) și vreți să rulați local folosind Docker, va trebui să activați opțiunea *Use Rosetta for x86/amd64 emulation on Apple Silicon* care se găsește în meniul *Settings* din Docker Desktop. Mai multe detalii puteți găsi aici.

Pentru testarea folosind VMChecker Next, găsiți detalii într-un document intitulat "Tutorial VMChecker Next" aflat pe pagina de Moodle a cursului de APD.

Link-uri utile

1. How Does BitTorrent Work? A Plain English Guide
2. How Does BitTorrent Work?
3. BitTorrent
4. BitTorrent tracker

[1] Nota obținută în urma rulării automate poate fi scăzută pe baza elementelor de depunctare descrise mai sus.