Universitatea Tehnică "Gheorghe Asachi", Iași

Facultatea de Automatică și Calculatoare

Sistem distribuit de stocare și versionare a fișierelor

- Raport intermediar I pentru lucrarea de diplomă -

Student : Stratulat Ștefănel Constantin (1409A)

Îndrumător : Buțincu Cristian

Anul universitar 2020 – 2021

Cuprins

Scopul temei alese 2
Referințe la teme/subiecte similare 3
Resurse hardware/software utilizate 3
Algoritmi sau metode alese 4
Rezultate așteptate 4

Scopul temei alese

Lucrarea are ca și obiectiv dezvoltarea unui sistem distribuit de stocare și versionare a fișierelor. Mai exact, se are în vedere un sistem în care utilizatorul își poate încărca fișierele astfel încât să le poată accesa ulterior din orice loc în care are acces la aplicație și la o conexiune la internet. Totodată, se asigură versionarea fișierelor astfel încât, dacă utilizatorul hotărăște să adauge o versiune actualizată a fișierului, va avea acces și la vechea versiune. Acesta este, în linii mari, scopul aplicației.

Proiectul adresează, în primul rând, disponibilitatea datelor. În acest sens, în funcție de tipul utilizatorului și de importanța oferită unui fișier încărcat, se va asigura replicarea fișierului pe mai multe noduri ale sistemului și menținerea numărului de replici disponibile pe toată durata de viață a fișierului (dacă un nod al sistemului va pica, se va asigura replicarea fișierului pe alt nod; dacă un nod al sistemului repornește, se va asigura ștergerea fișierului de la unul dintre noduri, cel mai încărcat).

De asemenea, proiectul adresează versionarea datelor. Pentru fiecare fișier existent, în cazul în care se dorește adăugarea unei versiuni actualizate a acestuia, se va asigura păstrarea versiunii anterioare și accesarea acesteia la nevoie.

Aplicația își propune să fie cât mai accesibilă. În acest sens, utilizatorul va putea interacționa cu aplicația atât prin intermediul unei aplicații web, cât și prin intermediul unei aplicații mobile. Totodată, nu se impun restricții în ceea ce privește formatul fișierelor încărcate, însă spațiul de stocare va depinde de tipul utilizatorului.

Referințe la teme/subiecte similare

Principalele două teme similare, care adresează cele două funcționalități (separat) sunt Dropbox și Github.

Dropbox este unul dintre cele mai de succes sisteme de stocare a fișierelor care se impune prin spațiul de stocare (și abonamentele accesibile de extindere), viteza de transfer, disponibilitatea datelor, back-up și posibilitatea de a transfera fișiere între mai multe conturi. Dropbox se adresează publicului larg.

- www.dropbox.com
- "Inside Dropbox : Understanding Personal Cloud Storage Service" – University of Twente (Idilio Drago, Anna Sperotto, Ramin Sadre, Aiko Pras) & Politecnico di Torino (Marco Mellia, Maurizio M. Munafo)

GitHub este cel mai de succes sistem de versionare a fișierelor, care se impune prin multitudinea de operații care se pot executa asupra versiunilor unor documente text. Github se adresează, în principal, dezvoltatorilor de aplicații software, facilitând procesul de lucrarea colaborativă.

- www.github.com
- "The Arhitecture and History of Git : A Distributed Version Control System" – Will Hay Jr. on medium.com, 2018

Resurse hardware/software utilizate

Fiind o aplicație software, nu sunt necesare resurse hardware specializate. Principala resursă hardware necesară este reprezentată de un sistem de calcul care are un sistem de operare care să permită dezvoltarea și rularea de aplicații distribuite, ce necesită comunicare în rețea. În acest sens, putem vorbi despre un computer, un laptop sau un dispozitiv mobil (smartphone).

În ceea ce privește resursele software, sunt necesare unelte care să faciliteze dezvoltarea de aplicații distribuite. Ca și limbaj de programare, se va folosi Java pentru dezvoltarea componetelor de logică de aplicație (componentele de stocare și versionare a datelor) și pentru dezvoltarea serverului Web (Spring Boot). Putem menționa folosirea socket-urilor pentru comunicarea dintre nodurile aplicației (pentru transfer de date și heartbeats - multicast).

Pentru componenta de frontend (interfață cu utilizatorul), se va folosi limbajul JavaScript (framework-ul React).

Pentru persistența datelor, în primul rând, vom folosi sistemul de fișiere al sistemului de operare pentru stocarea fișierelor și un sistem de baze de date relaționale SQL pentru gestiunea utilizatorilor.

Algoritmi sau metode alese

Aplicația are la bază un sistem distribuit. De aceea, avem nevoie de comunicare dintre nodurile sistemului. În acest sens, folosim, mai întâi, un mecanism de service discovery și heartbeats astfel încât să se țină evidența nodurilor noi apărute și a celor deja existente în sistem. Mai mult decât atât, întrucât avem un sistem self-healing (avem un mecanism de replicare a unui fișier, pentru care dorim să menținem numărul de replici), avem nevoie de un manager general al sistemului. Astfel, fiecare nod nou apărut în sistem își va trimite identitatea pe o adresă de multicast astfel încât managerul general să înregistreze noul nod. Mai departe, toate nodurile din sistem trimit la intervale regulate de timp, un mesaj (heartbeat) către nodul general astfel încât acesta să cunoască în permanență statusul nodurilor și dacă este nevoie de replicare.

În ceea ce privește încărcarea efectivă a unui fișier și replicarea acestuia, clientul se va conecta la frontend, atașând fișierul. Mai departe, frontend-ul va trimite o cerere către managerul general care, pe baza numărului necesar de replici și a statusului nodurilor active din sistem, va genera un token care va îngloba detalii despre fișier și nodurile care vor trebui să stocheze acel fișier. În continuare, din frontend se va trimite fișierul către primul nod din acest lanț. Acest nod va extrage adresa lui din token și va trimite mai departe. Această operațiune continuă, atâta timp cât nodurile la care ajung aceste mesaje, există și sunt active.

Principala componentă din acest sistem este managerul general, de care depinde toată activitate, întrucât este responsabil de menținerea statusului stocării fiecărui nod și a tuturor fișierele existente în sistem. Dacă acest nod general ar pica, întregul sistem ar deveni indisponibil. De aceea se asigură un mirror pentru acest nod, mirror care va putea reconstrui foarte ușor starea vechiului nod. Această reconstruire a stării se va realiza foarte ușor cu ajutorul mecanismului de heartbeats, întrucât fiecare nod trimite managerului general informații despre statusul curent (starea stocării și a fișierelor existente), cu o anumită frecvență. Astfel, la fiecare nou beat, managerul general poate reconstrui starea întregului sistem.

Rezultate așteptate

În ceea ce privește rezultatele așteptate, se dorește în primul rând, obținerea unui sistem care să funcționeze încontinuu. Ne referim aici la nodul manager general de care depinde tot sistemul, nod care nu ar trebui să pice niciodată. Astfel, se va garanta funcționarea neîntreruptă prin existența unui mirror pentru acest nod.

În al doilea rând, se dorește ca sistemul să fie capabil să asigure disponibilitatea fișierelor în orice moment. Ne referim aici la mecanismul de replicare controlat de managerul general. Se dorește menținerea numărului de replici ale unui fișier pe întreaga durată de viață a acestuia.

Se dorește totodată menținerea versiunilor anterioare ale unui fișier actualizat și accesarea ușoară a acestora. Ca o privire de ansamblu, se dorește un sistem fiabil, care să funcționeze fără blocaje, să asigure o încărcare/descărcare rapidă a fișierelor cu formate speciale, sau o vizualizare direct din interfață a fișierelor cu format general, se dorește posibilitatea stocării multor fișiere și a multor versiuni ale fișierelor.