Lucrare de laborator 2 (10 ore)

Tema

Web proxy: realizarea transparenței în distribuire

Scopul lucrării

Studiul protocolului HTTP în contextul distribuirii datelor. Utilizarea metodelor HTTP în implementarea interacțiunii dintre client și un server de aplicații. Studiul metodelor de caching și load-balancing aplicate la crearea unui serviciu proxy.

Obiectivele lucrării

Elaborarea unui server cu procesare concurentă a cererilor HTTP. Realizarea unei aplicații de intermediere a accesului la nodurile warehouse.

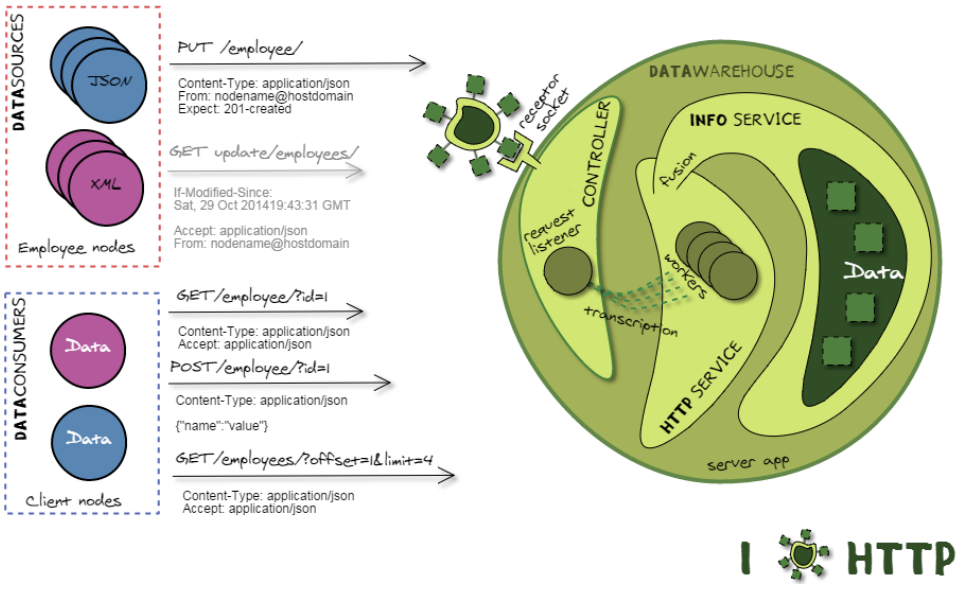
Considerații generale

Etapa 1

În această etapă clientul va obține datele semi-structurate distribuite prin intermediul unui nou nod special creat. Acest nod informativ va juca rolul de **data-warehouse (DW)**.

Serverul informativ va păstra toate datele necesare clientului. Orice interacțiune cu acest server se va face prin intermediul protocolului **HTTP**:

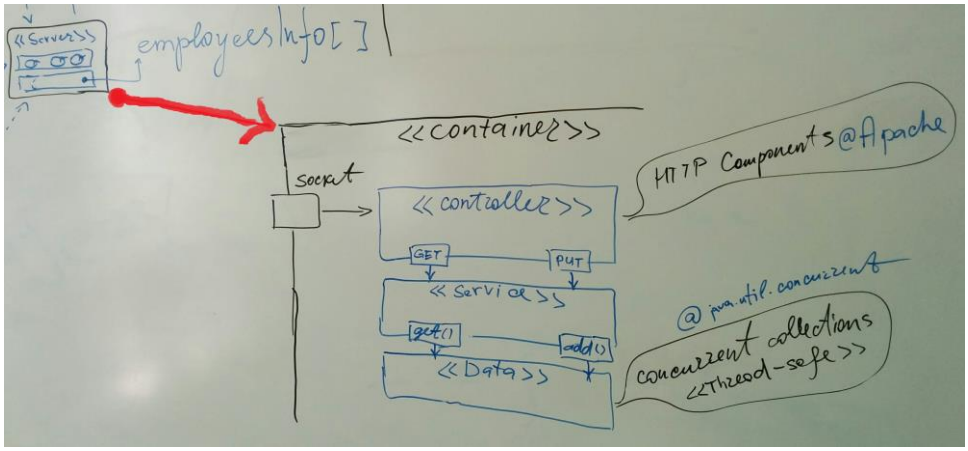
Metoda **GET** se va utiliza de client pentru a cere datele trimise anterior de clienți (e.g. GET /employee/?id=1 sau GET /employees/?offset=1&limit=4). Metoda **PUT** se va utiliza de nodurile de stocare pentru a trimite la nodul informativ datele necesare clientului. Metoda **POST** poate fi utilizata pentru transmiterea datelor de modificat spre DW, iar nodul sursă prin interogări repetate să ceară eventualele modificări (e.g. GET update/employees/ - metoda pull-request). Opțional, se acceptă realizarea metodei **PUSH**, cînd la modificarea unei entități nodul DW va iniția o cerere spre nodul sursă.



În vederea utilizării eficiente a resurselor de calcul se va impune procesarea **concurentă** a cererilor. Modelul multi-threading va corespunde schemei **thread-per-request**.

În condițiile cînd există acces simultan la colecția de date (adăugare/citire) se cere utilizarea unor colecții **thread-safe** sau implementarea unor mecanisme de acces reciproc exclusiv.

„Maparea" cererilor HTTP pe operațiile serviciilor se va face în componente numite generic **Controller**.

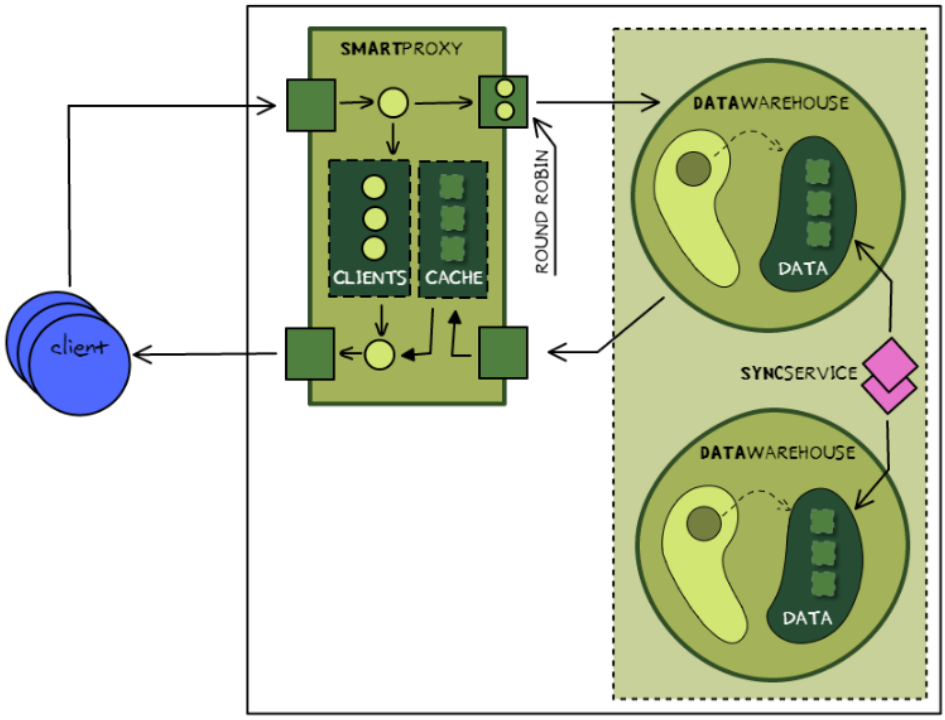


Etapa 2

In această etapă, toate cererile clienților vor fi preluate de un **reverse proxy** (mai departe proxy). Un proxy este un program care acționează ca un intermediar între o aplicație client și un **end-server**. Astfel, în loc de a interacționa direct cu serverul final pentru a obține datele necesare, clientul accesează aplicația proxy care și transmite cererea spre serverul final. Cînd end-server (în contextul DW) trimite răspunsul spre proxy, anume proxy și va trimite răspunsul final spre client. Printre beneficiile utilizării proxy se numără: load balancing și caching.

Caching este memorarea temporară a răspunsurilor trimise de serverele finale la nivelul nodului proxy. Aceasta se face pentru a accelera comunicare între client și server dar și pentru a micșora numărul de cereri către end-servers. Daca nodul proxy detectează o cerere, răspunsul la care a fost memorat, acesta îl va returna instantaneu clientului. Natura temporară a procesului de caching permite ca proxy sa nu transmită răspunsuri învechite care între timp au putut deveni incorecte.

Load balancing este procesul prin care un proxy distribuie sarcina între mai multe servere finale. Aceasta asigură ca nici unul dintre end-servers , la un moment dat, nu este suprasolicitat, iar cererile clienților sunt procesate rapid si într-o manieră eficientă. Unul dintre algoritmii des utilizați pentru load balancing este algoritmul Round-Robin.



Sarcini pentru lucrarea de laborator

Etapa 1

Definiți un nod informativ de tip DW care să comunice concurent cu ai săi clienți prin intermediul protocolului HTTP. Clientul trebuie să aibă posibilitatea să aleagă formatul datelor returnate (XML / JSON).

Etapa 2

Modificați nodul informativ din etapa 1 astfel încît comunicarea acestuia cu orice client să fie prin aplicația proxy. Funcțiile nodului proxy:

- ***Smart-proxy***, pentru păstrarea conexiunilor spre clienți;
- *Caching,* pentru a micșorarea timpul general de răspuns al sistemului;
- *Load-balancing*, în contextul măririi numărului de noduri de servire (DW) nodul proxy va selecta unul din serverele disponibile conform unui algoritm selectat.

Recomandări:

Utilizarea **BD Cassandra** pentru stocarea datelor. Pentru evitarea sincronizării datelor din instanțele de warehouse, se recomandă utilizarea unei instanțe de SGBD Cassandra, care poate imprima scalabilitatea necesară sistemelor informatice distribuite.

Utilizarea **Redis** ca sistem de stocare a conexiunilor. Proxy are nevoie de un sistem rapid de stocare de tip key-value persistent și Redis poate oferi această rapiditate.

Referințe

Scalable Web Architecture and Distributed Systems, http://aosabook.org/en/distsys.html
Forward Proxy vs Reverse Proxy, http://www.jscape.com/blog/bid/87783/Forward-Proxy-vs-Reverse-Proxy

Smart Proxy, *Enterprise Integration Patterns*, http://www.eaipatterns.com/SmartProxy.html
Caching in HTTP, Hypertext Transfer Protocol -- HTTP/1.1, http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html
Elemental Reverse Proxy, *Apache HTTP Components*, http://hc.apache.org/httpcomponents-core-ga/httpcore/examples/org/apache/http/examples/ElementalReverseProxy.java

HTTP

Soluție primară event-based de procesare a cererilor utilizînd Apache HttpComponents. Descarcă model aplicație: InfoNode.zip
Exemplu de aplicație Web utillizând JAX-RS/Jersey. Vezi în Gitlab: https://gitlab.ati.utm.md/diciorba/simple-restful-webservice

Redis

Pagina oficială de download, http://redis.io/download
Încearcă Redis prin interactive tutorial (http://try.redis.io/)!
Lista completă a comenzilor: http://redis.io/commands
Clienți recomandați: http://redis.io/clients
Versiunea pentru Windows, Microsoft Open Tech: https://github.com/MSOpenTech/redis/releases/

Cassandra

Pagina oficială de download: https://cassandra.apache.org/download/
Tutorial Cassandra: https://www.tutorialspoint.com/cassandra/cassandra_quick_guide.htm