Gherman Sebastian-Costin
343C2
All assignments done
Grade: 110 points

Questions

How many requests can be handled by a single machine?

- US 0-1 Pot sa duca ~60 RPS pana sa inceapa sa dea failure.
- ASIA 0-1 Pot sa duca ~ 65 RPS pana sa inceapa sa dea failure.
- EMEA 0 Poate sa duca ~ 45 RPS pana sa inceapa sa dea failure.

What is the latency of each region?

- US 0-1 ~ 370 ms.
- ASIA 0-1 ~ 520 ms.
- EMEA 0 ~ 300 ms.

What is the computation time for a work request?

- Work request-ul are in jur de 20 ms.

What is the the response time of a worker when there is no load?

- US 0-1 ~ 370 ms.
- ASIA 0-1 ~ 520 ms.
- EMEA 0 ~ 300 ms.

What is the latency introduced by the ***forwarding unit***?

- US
  - w/o forwarding ~ 340ms
  - with forwarding ~370 ms
  - FWD - w/o FWD = 30 ms

- ASIA
  - w/o forwarding ~ 490 ms
  - with forwarding ~ 520 ms
  - FWD - w/o FWD = 30 ms
- EMEA
  - w/o forwarding ~ 270ms
  - with forwarding ~ 300 ms
  - FWD - w/o FWD = 30 ms

How many requests must be given in order for the ***forwarding unit*** to become the bottleneck of the system? How would you solve this issue?

- Undeva in zona 70-100 RPS am descoperit mai multe fail-uri de tip Device or resource is busy. Pe la 250 RPS am descoperit un failure rate de peste 50% ceea ce reprezinta o problema majora.O posibila solutie ar fi sa se foloseasca un worker specific pe regiune in loc de /work general sau mai multe unitati de forward.

What is your estimation regarding the latency introduced by Heroku?

- Dupa ce am interogat heroku logs -a <app name> -t (pt real time) am observat ca are o latenta de ~20ms in load minim si variaza intre 300 si 600 ms pt un load de la 100 users - 1000 cu un RPS de 50-150.

What downsides do you see in the current architecture design?

- Consider ca exista o problema legata de dimensiunea buffer-ului alocat forward unit-ului care este mult prea mica si aduce multe fail-uri la un numar relativ mic de RPS. Cred ca de asemenea alegerea facuta de /work ar putea sa nu fie randomizata ci sa selecteze un worker in functie de latenta sau de disponibilitate.

Load Balancing

1. Random trimite request-urile catre workeri folosind random.choice() care alege unul din lista. Pentru request-uri putine este destul de prost, in schimb creste performanta la 1000+ request-uri.
2. Round Robin cicleaza prin lista si trimite request-uri la toti in mod egal. Are o performanta buna intre 500 si 1k request-uri.

3. Region Round Robin se foloseste de forwarding unit pentru a face acelasi lucru ca la 2, adica trimite direct la regiuni si nu la workeri. Este un pic mai prost ca Round Robin pentru ca se adauga latenta de la fwd unit.
4. Weighted Round Robin foloseste un vector de greutati care limiteaza numarul de conexiuni active la 3 workeri (nr de regiuni) fara a folosi fwd unit si trimite un chunk format din 3 request-uri catre emea (cel mai rapid), 2 la us si 1 la asia.Performanta foarte buna la un numar mic de request-uri si pentru un numar < 1000 request-uri.
5. leastLatency gaseste latenta cea mai mica disponibila dintre workeri si trimite toate request-urile catre acel worker. Performantele sunt bune < 1000 request-uri.
6. leastConnection este o combinatie intre leastLatency si Weighted Round Robin, adica gaseste care este cel mai rapid si trimite 50 de chunk-uri de catre 3 sesiuni a cate 5,2,1 request-uri si un chunk catre alte 3 workeri. Scopul este sa se ajute putin cel mai rapid worker trimitand din cand in cand si spre restul, cu o pondere mai mare. Nu vad o imbunatatire fata de least latency nici la 1500 request-uri. Posibil la un numar mare de Requests/s sa fie o imbunatatire, in momentul in care se gatuie un worker.

Nu exista o solutie buna pentru orice situatie. Depinde de numarul de request-uri si de incarcarea workerilor. Cea mai buna metoda pentru multe request-uri ar trebui sa fie leastConnection doar ca nu am reusit sa duc un numar foarte mare de conexiuni.

Pentru a afla o metrica de performanta am folosit viteza de terminare a script-ului. Toate scripturile sunt asincrone fara functii blocante. Am facut un script de test care itereaza printr-un vector de useri si de politici de 5 ori si face media trecerilor. Rezultatul il dau din shell catre un fisier text.