Test Mongo DB

Personal Computer

CPU: Intel i7-6500U 2590Mhz

RAM:8Gb

HDD:500gb

Nodo MongoDB su 172.16.34.76

CPU: Intel I7 2400Mhz

RAM: 4Gb

HDD: 30Gb

Fatturapa.tainet

CPU: QEMU 1800Mhz

RAM: 4Gb

HDD: 6,7Gb

Fatturapa2.tainet

CPU: ??? 1800Mhz

RAM: 4Gb

HDD: 5.3Gb

Fatturapa-pg.tainet

CPU: ??? 1800Mhz

RAM: 4Gb

HDD: 5.3Gb

TEST CON NODI 1 MONGO DB

1. Stress test su Nodi 1 con in esecuzione Mongo DB (database vuoto)

Il test è stato lanciato da una macchina fisica con 1.000.000 di entries come obiettivo (Modalità PUT + GET) suddivisi tra 4 threads in parallelo e con visualizzazione grafica dei dati.

Dopo un iniziale picco dovuto alle 4 connessioni aperte in contemporanea, i massimi tempi di put/get si sono stabilizzati intorno ai 105/108 ms sul thread più lento ed i minimi tempi di put/get si sono abbassati a 26/28 ms.

Dopo la prima stabilizzazione è stata collegata un'altra macchina virtuale dopo circa 60 secondi di esecuzione, con 50.000 entries come obiettivo (PUT + GET) suddivisi tra 8 threads in parallelo, senza visualizzazione grafica.

Ciò ha causato un leggero incremento dei tempi di put/get a 162/165 ms sul thread più lento, mentre i tempi minimi sono migliorati a 21/22 ms sul thread più veloce.

E' stato successivamente lanciata un'altra esecuzione dopo circa 100 secondi dall'inizio del test con obiettivo 50.000 entries (Modalità PUT + GET) suddivisi tra 8 threads in parallelo e senza visualizzazione grafica, su un'altra macchina virtuale, per un totale di 20threads in parallelo.

Si è verificato un altro picco nei valori di max put/get fino a 219/221 ms, mentre nessun cambiamento rilevante nei tempi minimi.

Durante il test le macchine virtuali sono state più volte spente e rilanciate con gli stessi obiettivi, e sono state eseguite periodicamente 2 query per un totale di 11 findByCompany() (last execution time 561ms) e 20 CountAll() (last execution time 3ms). Ci sono stati lievi picchi di performance ma non significativi in seguito

a questi eventi. Ci sono stati invece picchi molto più significativi in seguito al maggior numero di dati presenti nel database che hanno portato ad un incremento dei tempi max di get/put ad intervalli lunghi ma apparentemente non regolari (circa ogni 300-1000 secondi si sono verificati cali di performance).

Il test è stato fermato dopo 4969 secondi di esecuzione con un totale di 729896 entries scritte sul database. Non sono state rilevate failures nell'inserimento dei valori (possibile bug nel software, da verificare).

Al termine del test il tempo massimo di put in generale è stato di 583 ms, il tempo massimo di get in generale è stato di 585 ms entrambi sullo stesso thread. Il tempo minimo di put è stato di 21 ms mentre il tempo minimo di get è stato di 22 ms su due processi differenti.

La velocità media di inserimento si è aggirata intorno ai 27,86 ms, il tempo medio d i get invece è stato di 23,5 ms

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Total Entries | Avg Put Time | Avg Get Time | Total Time | Max Put Time | Min Put Time | Max Get Time | Min Get Time | Failure Rate |
| 0 | 137.777 | 22 | 24 | 4969 | 583 | 21 | 585 | 23 | 0 |
| 1 | 137.890 | 21 | 23 | 4969 | 535 | 21 | 538 | 23 | 0 |
| 2 | 137.710 | 22 | 24 | 4969 | 534 | 22 | 536 | 24 | 0 |
| 3 | 137.829 | 22 | 23 | 4969 | 538 | 21 | 540 | 22 | 0 |

2. Lanciati 2 thread con PUT/GET su macchina con GUI, 1.000.000 entries, 8 thread PUT/TGET su macchina virtuale fatturapa da 500.000 entries a t=400s, 8 thread PUT/TGET su macchina virtuale fatturapa2 da 500.000 entries a t=500s

Lanciati 10thread PUT/GET su macchina virtuale fatturapa2 da 150.000 a t=5700s

Lanciati 10thread PUT/GET su macchina virtuale fatturapa da 150.000 a t=5800s

Test interrotto dopo 9437 sec. Tot Entries: 1.967.S284

T=1563 Entries=55491 E/T=35,50entries/sec

T=2353 Entries=80780 E/T=34,33 entries/sec

CountAll() lanciate = 640

findByCompany() lanciate = 640

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Total Entries | Avg Put Time | Avg Get Time | Total Time | Max Put Time | Min Put Time | Max Get Time | Min Get Time | Failure Rate |
| 0 | 97.858 | 17 | 19 | 2837 | 302 | 17 | 342 | 19 | 0 |
| 1 | 97.907 | 17 | 18 | 2837 | 304 | 16 | 342 | 17 | 0 |

3. Test su singolo nodo con 3 macchine virtuali che lanciano 500.000 entries (PUT/GET) suddivise in 8 thread ciascuna + macchina fisica con GUI che lancia 3 thread ed un totale di 1.000.000 di entries + countAll() e findByCompany() ogni 10 secondi. Database contiene già > 1.900.000 di elementi.

Le macchine virtuali verranno lanciate a distanza di 100 secondi l’una dall’altra non appena la precedente sarà a “regime”.

T=310 sec lanciata fatturapa-pg

T=410 sec lanciata fatturapa

T=510 sec lanciata fatturapa2

T=1722 fermata fatturapa-pg

T=1800 sec rilanciata fatturapa-pg con 500.000 entries suddivise in 10 thread

T=2038 fermata fatturapa

T=2083 sec rilanciata fatturapa con 500.000 entries suddivise in 10 thread

T=2450 primo aumento di MaxPutTime e MaxGetTime (N.b. i valori erano già alti ad inizio test per la presenza di dati preesistenti

T=2682 fermata fatturapa2

T=2719 sec rilanciata fatturapa2 con 500.000 entries suddivise in 10 thread

T= 5733 sec test interrotto manualmente

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Total Entries | Avg Put Time | Avg Get Time | Total Time | Max Put Time | Min Put Time | Max Get Time | Min Get Time | Failure Rate |
| 0 | 165869 | 18 | 20 | 5733 | 639 | 18 | 642 | 20 | 0 |
| 1 | 165855 | 19 | 20 | 5733 | 635 | 19 | 638 | 20 | 0 |
| 2 | 166005 | 19 | 20 | 5733 | 641 | 19 | 643 | 20 |  |

4. Test su singolo nodo con 3 macchine virtuali che lanciano 500.000 entries (PUT/GET) suddivise in 8 thread ciascuna + macchina fisica con GUI che lancia 3 thread ed un totale di 1.000.000 di entries + countAll() e findByCompany() ogni 10 secondi. Database vuoto.

Le macchine virtuali verranno lanciate a distanza di 100 secondi l’una dall’altra non appena la precedente sarà a “regime”.

//da rilanciare

5.Test su singolo nodo con una sola macchina con GUI che lancia 200.000 entries (PUT/GET) suddivise in 4 thread + query ogni 10 secondi. Database vuoto.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Total Entries | Avg Put Time | Avg Get Time | Total Time | Max Put Time | Min Put Time | Max Get Time | Min Get Time | Failure Rate |
| 0 | 50000 | 21 | 22 | 1732 | 1349 | 19 | 1352 | 21 | 0 |
| 1 | 50000 | 20 | 22 | 1732 | 1336 | 19 | 1339 | 20 | 0 |
| 2 | 50000 | 20 | 22 | 1733 | 1346 | 20 | 1349 | 22 | 0 |
| 3 | 50000 | 17 | 18 | 1352 | 1352 | 17 | 1355 | 18 | 0 |

Rispetto ai precedenti test abbiamo una performance peggiore in termine di Max Put Time. E’ possibile che i thread lanciati da una stessa macchina siano soggetti a rallentarsi tra loro, mentre per quanto riguarda i thread lanciati successivamente da altre macchine, essi non influiscano con quelli che stanno già inserendo attivamente.

TEST CON NODI 2 MONGO DB

1.Test su 2 nodi mongo db configurati in sharding senza replica-set con un config server.

Vengono utilizzati come client: *fatturapa.tainet, fatturapa2.tainet, fatturapa-pg.tainet* ed il mio personal computer.

Ciascuna macchina virtuale lancerà 10 thread in contemporanea con obiettivo 2.000.0000 di entries in modalità PUT/GET.

Dalla mia macchina lancerò 4 thread monitorati da GUI con 1.000.000 di entries in modalità PUT/GET.

Autocheck attivato da inizio test, aggiorna i dati ogni secondo con richieste GET e lancia ogni 10 secondi una findByCompany(“ww”) ed una CountAll()

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| # | Total Entries | Avg Put Time | Avg Get Time | Total Time | Max Put Time | Min Put Time | Max Get Time | Min Get Time | Failure Rate |
| 0 | 201008 | 16 | 19 | 8591 | 36373 | 16 | 36380 | 19 | 0 |
| 1 | 201024 | 17 | 21 | 8591 | 36346 | 17 | 36363 | 21 | 0 |
| 2 | 201047 | 18 | 20 | 8591 | 36367 | 16 | 36378 | 19 | 0 |
| 3 | 200859 | 19 | 21 | 8591 | 36312 | 19 | 36327 | 21 | 0 |