ՀՏԴ 004.383:004.451.5

Գ.Հ. ՍԱՐԳՍՅԱՆ

ՍԵՐՎԵՐՈՒՄ ՄԵԾ ՏՎՅԱԼՆԵՐԻ ՊԱՀՊԱՆՄԱՆ ԿԱՌԱՎԱՐՈՒՄԸ ԳԵՐԲԵՌՆՄԱՆ ՊԱՅՄԱՆՆԵՐՈՒՄ

Աշխատանքի նպատակն է բացահայտել տվյալների գերբեռնվածության հիմքերը, կառավարել և վերահսկել սերվերի մուտքի մեխանիզմը՝ գերծանրաբեռնվածության պայմաններում տվյալների պաշտպանության համար: Կատարվել է տվյալների գերբեռնվածության հիմքերի բացահայտում, սերվերի մուտքի տվյալների ամբողջականության մեխանիզմի կառավարում, մոնիտորինգ, և մշակվել է վերահսկման ռազմավարություն:

Տվյալների գերբեռնվածությունը կանխելու կամ մեղմելու համար հավասարակշռման և սերվերի վերահսկողության մեխանիզմները տվյալների անկորուստ մշակման գրավականն են: Հետազոտման հիման վրա մշակվել է մեխանիզմ ադմինիստրատորի համար՝ հետևելու, հսկելու և կառավարելու գերբեռնված սերվերի բեռի բաշխումը իրատեսական ժամանակում: Մշակվել է տվյալների կառավարման և գերբեռնվածություններից պաշտպանության միջոց Ապաչ (Apache) սերվերի համար:

Տվյալների բեռի կառավարման համակարգի մշակումը հերթերի կառավարման համակարգերի տեսական հասկացությունների ինտեգրումն է և գործնական ծրագրային հմտությունների կիրառումը սերվերի կառավարման մեջ: Նման համակարգը կարող է օգտակար լինել տարբեր ոլորտներում, ինչպիսիք են հեռահաղորդակցությունը, համակարգչային և խելացի ցանցերը: Հետազոտվել են Apache-Cassandra սերվերում մեծածավալ տվյալների պահպանման և կառավարման խնդիրները գերբեռնման պայմաններում:

Առանցքային բառեր. Ապաչի-Կասանդրա սերվեր, ինտերակտիվ համակարգ, առցանց տեխնոլոգիաներ, կառավարում, գերբեռնվածություն:

Ներածություն: Զանգվածային սպասարկման տեսության վրա հիմնված ինտերակտիվ ցանցային բեռնվածության կառավարման համակարգի մշակումը ներառում է տեսական հասկացությունների ինտեգրում հերթերի տեսության համակարգերից և գործնական կիրառություն ցանցի կառավարման մեջ: Մեծ տվյալների մշակման համար սահմանվել են գերբեռնվածության կառավարման կոնկրետ խնդիրները: Նկարագրվել է տվյալների բեռի վարքը սերվերում՝ կիրառելով հերթերի տեսության մոդելներն ու ալգորիթմները: Ցանցային բեռի կառավարման համար կիրառվել են մաթեմատիկական մոդելները, կատարվել սերվերի ընտրություն, ստեղծվել արհեստական գերբեռնում, առաջադրվել են բեռի բաշխման լուծումներ, մշակվել է նոր ցանցային ճարտարապետություն:

Աշխատանքի նպատակն է բացահայտել տվյալների գերբեռնվածության հիմքերը, կառավարել և վերահսկել սերվերի մուտքի մեխանիզմը՝ գերծանրաբեռնվածության պայմաններում տվյալների պաշտպանության համար: Մշակվել

են տվյալների գերբեռնման պայմաններում տվյալների անկորուստ պահպանման մեթոդներ, և իրականացվել է մոնիտորինգ:

Հետազոտության նյութը: Աշխատանքի կատարման ընթացում դիտարկվել են երկու զուգահեռ M/M/N/N հերթեր: Յուրաքանչյուր հերթում առկա են N սերվեր և սպասման հերթեր: Ցանցը սնվել է մեկ Պուասոնի պրոցեսում: Սա մոդել է մի շարք դիսկրետ իրադարձությունների համար, որտեղ իրադարձությունների միջև միջին ժամանակը հայտնի է, բայց իրադարձությունների ճշգրիտ ժամանակը պատահական է [1]: Իրադարձության առաջացումն անկախ է նախկինում տեղի ունեցած իրադարձությունից (ժամանման հոսքի λ արագությամբ), իսկ 2N սերվերները նույնական էքսպոնենցիալ սերվերներ են, որոնք աշխատում են μ արագությամբ: Նոր, մեծ տվյալները տեղ հասնելիս ուղղորդվում են ավելի փոքր քանակությամբ զբաղեցրած սերվերներին: Եթե երկու սերվերներն էլ ունեն նույն զբաղվածությունը, ապա ժամանումն ուղղորդվում է պատահականորեն, ընդ որում, հերթին միանալու հավանականությունը 1/2 է: Այս մոդելը կարող է դիտվել որպես Էրլանգի[1] կորստի դասական մոդելի ամենակարճ հերթի տարբերակը: Եթե բոլոր 2N սերվերները զբաղված են, հետագա ժամանումները հետ են մղվում և կորչում: Դիցուք ρ = λ/μ և a = N/ρ = Nμ/λ: Դիտարկվում է մոդելը և՛ թվային, և՛ ասիմպտոտիկ, որի համար գերբեռնված համակարգերը (ρ) ներկայացվել են մեծ թվով սերվերներով (N a = O(1)-ով):

Խնդրի լուծման համար կիրառվել են H. Yao և Ch. Knessl-ի վերը նշված աշխատանքում առկա մաթմոդելները. համաձայն որոնց առաջարկվել են լուծումներ[3]: «Ասիմպտոտիկ մոտարկումները բերեցին առաջին հերթում զբաղեցրած m սերվերների հայտնաբերման միասնական կայուն վիճակի բաշխմանը, իսկ երկրորդում՝ n: Հաշվի առնվեց երկրորդ հերթում զբաղեցրած սերվերների քանակի մարգինալ բաշխումը, ինչպես նաև՝ որոշ պայմանական բաշխումներ: Ցույց տրվեց, որ լուծման կողմերը շատ տարբեր են՝ ըստ a > 1/2, a1/2, 1/4 < a < 1/2, a1/4 կամ 0 < a < 1/4: Սերվերում գեռբեռնման բաշխման ասիմպտոտիկ մոտարկումները թվային առումով բավականին ճշգրիտ են»[2]:

Քննարկելով մեծ տվյալների հասանելիության խնդիրները և կորստի վտանգները՝ պետք է կիրառել այնպիսի սերվերներ, որոնք կունենան տվյալների բաշխման ուղղորդման հնարավորություն: Հետազոտման արդյունքում մեծ տվյալներով աշխատող բազմաթիվ սերվերներից ընտրվել է Apache Cassandra[3]-ն,

[1] Yao H. and Knessl C. On the Shortest Queue Version of the Erlang Loss Model https://doi.org/10.1111/j.1467-9590.2007.00399

[2] YAO H. KNESSL C, On the infinite server shortest queue problem: Non-symmetric case, Queueing Systems-2006, 52 p.157–177

[3] https://cassandra.apache.org/ /index.html

որը բաց կոդով, բաշխված NoSQL տվյալների բազայի կառավարման համակարգ է՝ նախատեսված հսկայական քանակությամբ (սկսած 1ՊԲ) տվյալների մշակման համար՝ բարձր հասանելիությամբ, մասշտաբայնությամբ և սխալների հանդուրժողականությամբ: Cassandra-ն լայնորեն օգտագործվում է կազմակերպությունների կողմից տվյալների ինտենսիվ աճի, իրական ժամանակում տվյալների հասանելիության և վերլուծության համար: Apache Cassandra-ն ունի բաշխված ճարտարապետություն. այն կարող է ընդգրկել բազմաթիվ հանգույցներ և տվյալների կենտրոններ: Տվյալները բաշխվում են հանգույցների վրա՝ ապահովելով բարձր հասանելիություն և անվտանգություն: Cassandra-ն մասշտաբավորվող է, հանդուրժող սխալների նկատմամբ. հանգույցներից մեկի ձախողման դեպքում տվյալներն առբերվում են այլ հանգույցների կրկնօրինակներից: Տվյալները կազմակերպված են աղյուսակներով, և յուրաքանչյուր տող կարող է ունենալ տարբեր սյունակներ, որը թույլ է տալիս արագաշարժ տվյալների մոդելավորում: Cassandra-ն սովորաբար օգտագործվում է ֆինանսների, էլեկտրոնային առևտրի, մեդիայի, IoT և այլ ոլորտներում:

Աշխատանքի կատարման ընթացքում հետազոտվել են գերբեռնվածության կանխարգելման ալգորիթմները, որոնք օգտագործվում են տարբեր համակարգերում և ցանցերում՝ կառավարելու և մեղմելու համակարգի ծանրաբեռնվածության կամ չափազանց մեծ ռեսուրսների օգտագործման ռիսկը: Այս ալգորիթմները շատ կարևոր են համակարգի կայունության, կատարողականի և հուսալիության պահպանման տեսանկյունից:

Ներկայացնենք գերբեռնվածություն դասական կանոնը: «Երբ ցանցում միաժամանակ փոխանցվող փաթեթների քանակը գերազանցում է շեմային մակարդակը, ցանցի արտադրողականությունը սկսում է նվազել: Այսպիսի իրավիճակը կոչվում է գերբեռնվածություն» [4]:

Ներկայացված ներքոհիշյալ գերբեռնվածության կանխարգելման ալգորիթմներից կատարվել է ընտրություն [5].

- Երթևեկության ձևավորման ալգորիթմները, որոնք վերահսկում են տվյալների ուղարկման կամ ստացման արագությունը՝ կանխելու երթևեկության հանկարծակի աճերը, որոնք կարող են ծանրաբեռնել ցանցը կամ սերվերը:
- Հարցումների կամ գործարքների քանակի սահմանափակումը, որը կարող է կարգավորել համակարգը որոշակի ժամկետում: Այն կանխում է համակարգի ռեսուրսների ավելորդ պահանջարկը:
- Ծանրաբեռնվածության հավասարակշռման ալգորիթմները բաշխում են մուտքային երթևեկությունը մի քանի սերվերների կամ ռեսուրսների վրա՝ կանխելու առանձին սերվերի գերբեռնումը: Բեռի հավասարակշռման ընդհանուր

տեխնիկան ներառում է Round Robin, Least Connections և Weighted Round Robin մեխանիզմները:

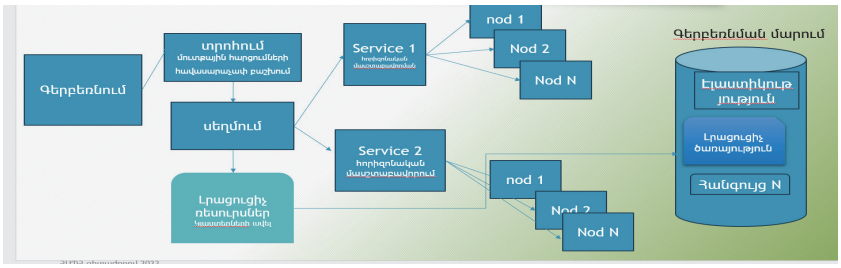
▪ Խցանումների գերբեռնվածության վերահսկման ալգորիթմները կարգավորում են տվյալների ուղարկման արագությունը՝ ցանցի գերբեռնումից խուսափելու համար: TCP-ի գերբեռնվածության վերահսկման մեխանիզմները, ինչպիսիք են TCP Reno-ն և TCP Vegas-ը:

▪ Ռեսուրսների ամրագրման ալգորիթմները նախապես հատկացնում և պահում են ռեսուրսներ կարևոր առաջադրանքների կամ ծրագրերի համար: Սա ապահովում է հավելվածներին անհրաժեշտ ռեսուրսները՝ կանխելով ծանրաբեռնվածությունը:

▪ Հերթերի կառավարման ալգորիթմներն առաջնահերթություն են տալիս և վերահսկում մուտքային հարցումների կամ հաղորդագրությունների արագությունը՝ կանխելով հերթերի գերբեռնումը և ռեսուրսների հագեցվածությունը:

▪ Անոմալիաների հայտնաբերման ալգորիթմները վերահսկում են համակարգի չափումները և օգտատերերի վարքագիծը՝ հայտնաբերելով անսովոր օրինաչափությունները:

Աշխատանքի կատարման ընթացակարգը: Աշխատանքի համար ստեղծվել է հիբրիդային համակարգ (նկ.1.) բաղկացած սերվերից, համակարգչից և 3 ամպային պլատֆորմների վրա տեղակայված շտեմարանային հարթակներից (PaaS)՝ կազմելով 5 հանգույց (5 nod): Համակարգի վրա տեղակայվել է Apache Cassandra:



Նկ. 1. Հիբրիդային կառուցվածքը

Ներմուծվել են մեծ տվյալներ, և տվյալների անվտանգ պահպանման գործընթացում Cassandra-ով աշխատելու համար կիրառվել են Java 11, Oracle Java Standard Edition 11. OpenJDK 11 ծրագրային միջոցները, իսկ cqlsh-ի օգտագործման համար՝ Python 3.6+: Իրականացվել է գերբեռնում:

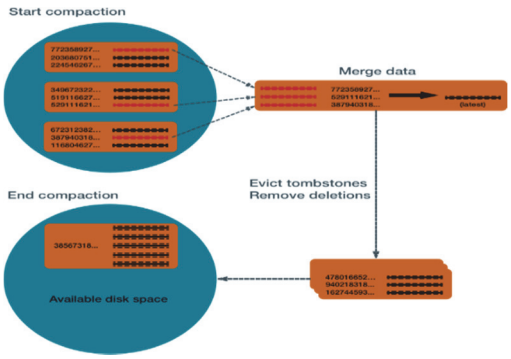
Հիբրիդային համակարգում իրականացվել է մեծ տվյալների հավաքագրման, մշակման և պահպանման գործընթաց, առաջացվել է պարբերական և խառը

գերբեռնում, որի արդյունքները ներկայացված են ստորև (մտահսկման արդյունք, նկ. 2): Գերբեռնման հայտնաբերման համար կիրառվել է Datadog Apache Cassandra Monitor-ը:

Նկ.2. Տվյալների վիճակն ըստ Datadog Apache Cassandra Monitor

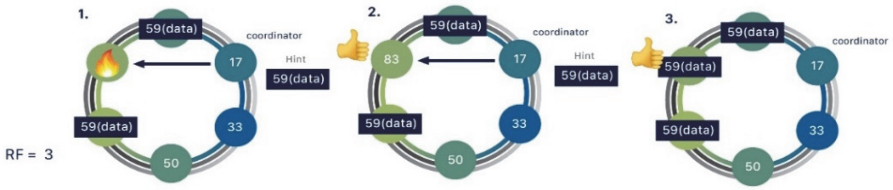
Գերբեռնման ցուցիչները ցույց են տվել, որ բեռնումը ժամանակավորապես գերազանցել է համակարգի տվյալ հատվածի ռեսուրսների հնարավորությունները: Ըստ գոյություն ունեցող լուծումների՝ կամ պետք է ավելացնել համակարգի ռեսուրսները, կամ հավասարակշռել և նվազեցնել բեռնվածությունը: Փորձարկման առաջին փուլում բեռի նվազեցման համար կիրառվել են սեղմման սխեմաները՝ LZ4Compressor, SnappyCompressor և DeflateCompressor ծրագրային միջոցներով (նկ. 3), որոնց մեջ մեծ տվյալների սեղմման սխեման կախված է տարածության խնայողության պահանջից: Փորձարկվող LZ4-ն սխեման ամենաարագն էր:

Սեղմման արդյունավետությունը հակադարձ փոխկապակցված է ընդլայնման արագության հետ: Հետազոտման ընթացքում Deflate-ը կամ Snappy-ը գերբեռնվածության դեպքում չդրսևորեցին մեծ արագագործություն, սակայն լավ աշխատեցին արխիվային տվյալների հետ: Հայտնի է, որ հերթերի կառավարման ալգորիթմների և մուտքային հարցումների վերահսկման կիրառմամբ, հերթերի գերբեռնումից խուսափելու և ռեսուրսների հագեցվածությունը ապահովելու համար, Cassandra-ն կարող է ավելացնել հզորությունը՝ առցանց նոր հանգույցներ ավելացնելով [6]:



Նկ. 3. Սեղմման կառուցվածքը

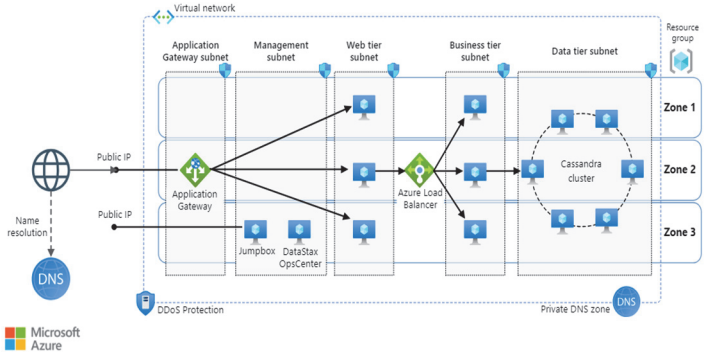
Երկրորդ փուլում ավելացվեցին համակարգի ռեսուրսները, բեռի հավասարակշռման և նվազեցման համար: Բեռի հավասարակշռման որոշումները կարող են հիմնված լինել իրական ժամանակի չափումների վրա, օրինակ՝ հանգույցի բեռնվածությունը, հիշողության օգտագործումը և արձագանքման ժամանակները: Cassandra-ի բաշխված բնույթը նպաստում է համակարգի ճկունությանը և կատարողականությանը: Նույն տվյալների մի քանի կրկնօրինակումը երաշխավորում է տվյալների անվտանգությունը՝ կրկնօրինակներից մեկի անհասանելիության դեպքում, մյուս հանգույցները հասանելի են՝ հարցումը կատարելու համար (նկ. 4):



Նկ.4. Cassandra-ում կրկնօրինակման իրականացումը

Բազմաթիվ կրկնօրինակների կիրառումն ունի առավելություն, սակայն հանգեցնում է պահվող տվյալների ծավալի բեռի ավելացման:

Cassandra-ն ավտոմատ կերպով կարող է կրկնօրինակել այդ տվյալները աշխարհի ծագերում գտնվող տարբեր տվյալների հանգույցների վրա: Մեր մեծ տվյալները գրվել են Cassandra-ի ամպային տարբեր հարթակների հանգույցներում, և այդ տվյալները ավտոմատ հասանելի են եղել մեր ստացիոնար համակարգչի և սերվերի տվյալների հենքում: Գերբեռնման լուծման ուշագրավ մեխանիզմի լավ օրինակ կարող է լինել Cassandra-ի դասական սխեմաներց մեկը, որը կիրառում է MS Azure [7]-ը (նկ. 5):



Նկ.5. Azure սխեման [7]

Չունենալով գերբեռնման համակարգի բեռի հավասարակշռման Ազուրի հնարավորությունները, որն ունի առանձին բեռի բաշխիչ, մեր կողմից ստեղծվել է նոր եռաստիճան մեխանիզմ՝ Cassandra-սերվերի համար: Ինչպես նշվեց վերը, 2 փուլով գերբեռնման փորձարկումները պահանջեցին լրացուցիչ մեծ քանակությամբ ռեսուրսներ: Առաջարկվող նոր եռաստիճան մեխանիզմն ավելի քիչ ռեսուրսներ է պահանջել՝ կիրառելով տրոհման գաղափարը, իրականացրել մուտքային հարցումների հավասարաչափ բաշխում, ապա կատարել սեղմում և հետո միայն առաջադրել ռեսուրսների ավելացման պահանջ:

Cassandra-ում իրականացվեց մեծ տվյալների բեռնվածության դինամիկ հավասարակշռում և տվյալների տեղաբաշխում հանգույցների միջև՝ Network Topology Strategy ռազմավարության շնորհիվ, որում հանգույցները կարող են ավելացվել կամ հեռացվել կլաստերից: Cassandra-ն ավտոմատ կերպով վերաբալանսավորեց տվյալները՝ հավասարաչափ բաշխումը պահպանելու համար: Մեծ տվյալների պահպանման կենտրոններում սովորաբար օգտագործվում է արտաքին բեռնաչափման ծրագրակազմ (Load Balancer Software): Հանրաճանաչ բեռի հավասարակշռիչները, ինչպիսիք են HAProxy-ը կամ NGINX-ը, կարող են կազմաձևել հարցումները՝ Cassandra հանգույցներին բաշխելու համար: Մեր մեխանիզմում արտաքին բեռնաչափման ծրագրակազմ չի կիրառվում: Cassandra-ն ունի բեռի հավասարակշռման քաղաքականություն, որը որոշում է հանգույցների քանակը: Կիրառել ենք այն հետևյալ կարգավորմամբ.

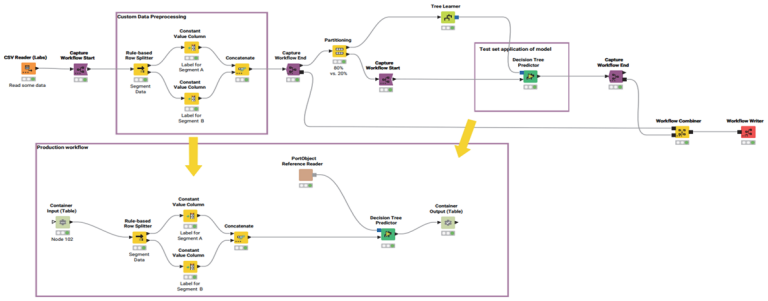
```
datastax-java-driver.basic.load-balancing-policy
 {
 class = DefaultLoadBalancingPolicy
}
```

Յուրաքանչյուր հանգույցի համար հաշվվել է ստեղծված կապերի հեռավորությունը, կատարվել հարցումը, հաշվվել է հանգույցների ցանկը:

Բեռը հավասարակշռելու համար ընտրվել են այն հանգույցներ, որոնք կարող էին հարցումները մշակել: Հիմնավորվեց տիպային հարցումների մեկ կամ մի քանի կիրառական օրինակների տեղակայումը Cassandra-ի տվյալների հանգույցներում.

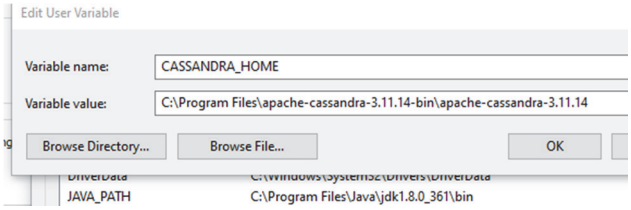
```
datastax-java-driver.basic.load-balancing-policy {
 local-dataserver1 = datacenter1
}
CqlSession session = CqlSession.builder()
 .withLocalDatacenter("datacenter1")
 .build();
```

Ըստ Yao H. և Knessl C. աշխատանքում առկա մաթմոդելների բեռի բաշխման համար լռելյայնորեն հարցումները առաջնահերթորեն ուղղորդվել են այն կրկնօրինակներին, որոնց պատկանում են հարցվող տվյալները: Տվյալներն ուղղորդվել են այն հանգույցին, որը պատասխանատու է տվյալների համար՝ հիմնված բաժանման բանալիի վրա՝ նվազագույնի հասցնելով հոփերի քանակը և նվազեցնելով հետաձգումը: Cassandran աջակցում է բալանսավորմանը, հաշվի առնելով յուրաքանչյուր հանգույցի հետաձգումը: Հարցումներն ուղարկվել են ամենացածր ուշացումով հանգույցին, որը կարող է բարելավել իրականացումը՝ կատարելով տվյալների տրոհումը (նկ. 6): Տվյալների բեռի տրոհման սխեման կատարվում է ըստ տվյալների առաջնայնության, պատկանելության և իրականացման:



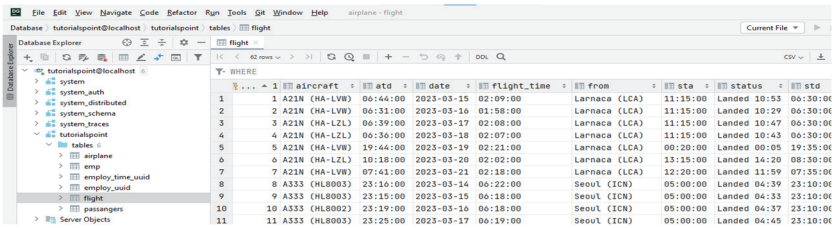
Նկ.6. Տրոհման սխեման

Առաջարկվող մեխանիզմը փորձարկվել է սերվերային հիբրիդային համակարգի և բազմաթիվ օգտատերերի ստեղծմամբ: Ստեղծվել և փորձարկվել է գերբեռնման համակարգի նոր սերվերային մոդելը: Ստեղծված սերվերային համակարգն (ըստ նկ. 1) իրականացրել է աշխատանքի կատարումը և հսկումը իր հարակից գործիքակազմերով և կարգաբերումներով Edit environment կառավարման վահանակի system properties (նկ. 7) բաժնում:



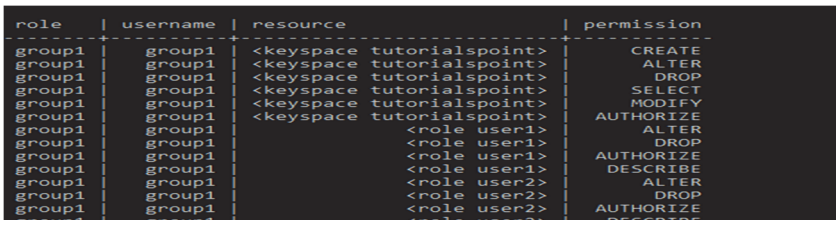
Նկ. 7. Edit user variable վահանակ

Տվյալների անվտանգ փոխանակման, սեղմման, տրոհման և հետագայում ռեսուրսների ավելացման համար իրականացվել են ցանցային կարգավորումներ և անհրաժեշտ բնիկների բացում բոլոր 5 հանգույցներում և սերվերի վրա՝ Cassandra-ի աշխատանքի հավասարաչափ և անվտանգ իրականացման համար: Տվյալների անվտանգության համար իրականացվել են conf թղթապանակի աու-տենտիֆիկացիայի կարգաբերումներ Cassandra.yaml ֆայլ-ում, որտեղ կատար-վել է autenticatore-ի (իսկորոշիչի) փոփոխում PasswordAuthenticator: Ստեղծվել է XX օգտատիրոջ ադմինիստրատիվ դեր և տրվել հնարավորություններ ու մուտքի իրավասություն Cassandra-ի աշխատանքային միջավայրի կառավարման համար: Գրաֆիկական մասում մուտքի թույլատվությունը տրվել է իսկորոշիչի միջոցով DataGrip-ի միջավայրում (նկ. 8) [8]:



Նկ. 8. DataGrip-ի միջավայրում սրեղծված աղյուսակները

Ստեղծվել են կիրառող օգտատերեր, Cassandra-ում ստեղծված նոր դերերից յուրաքանչյուրին տրվել է համապատասխան լիազորություններ՝ կատարելով փոփոխությունները Cassandra.yaml ֆայլում՝ ըստ CassandraAutorizer-ի, և ըստ ստեղծված նոր դերերի կատարվել է խմբավորում (նկ. 9):



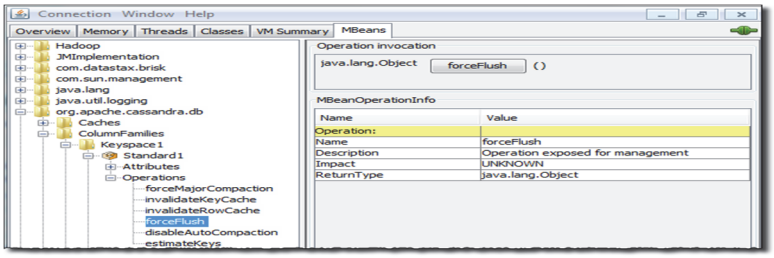
role	username	resource	permission
group1	group1	<keyspace tutorialspoint>	CREATE
group1	group1	<keyspace tutorialspoint>	ALTER
group1	group1	<keyspace tutorialspoint>	DROP
group1	group1	<keyspace tutorialspoint>	SELECT
group1	group1	<keyspace tutorialspoint>	MODIFY
group1	group1	<keyspace tutorialspoint>	AUTHORIZE
group1	group1	<role user1>	ALTER
group1	group1	<role user1>	DROP
group1	group1	<role user1>	AUTHORIZE
group1	group1	<role user1>	DESCRIBE
group1	group1	<role user2>	ALTER
group1	group1	<role user2>	DROP
group1	group1	<role user2>	AUTHORIZE

Նկ. 9. Դերերի ցուցակը և դերերին տրված լիազորությունները

Տվյալների խառը ծանրաբեռնվածության թեստեր մշակելու համար կիրառվել է YAML-ը, որն աջակցում է օգտատիրոջը: Ստեղծվել և փորձարկվել է գերբեռնման համակարգի նոր մոդել՝ կիրառելով YAML-ը քանի որ այն տարբեր սեղմման ձևերով և քեշի կարգավորումներով աջակցում է Cassandra-stress-ին: Գերբեռնվածության փորձարկման համար կիրառվել է cassandra-stressJava-ի վրա հիմնված սթրես-թեստ ծրագիրը, որը կիրառվել է Cassandra կլաստերի հիմնական չափորոշիչի հսկման համար: Տվյալների գերբեռնման փորձարկումների ընթացքում խնդիրների հայտնաբերման համար կիրառված cassandra-stress գործիքը (կլաստերի արդյունավետ համալրման) որոշել է, թե ինչպես են գործում առաջարկվող եռաստիճան մեխանիզմը և տվյալների հենքերի մասշտաբները:

Բաժանվել է բեռը մի քանի cassandra-stress թեստերի տարբեր հանգույցների:

Բաշխումը հսկելու համար մոնիտորինգն իրականացվել է JConsole-ի միջոցով, որը JMX-ին համապատասխանող գործիք է՝ ներառված Sun JDK 5.0-ում: JConsole-ն օգտագործվել է Cassandra-ի ներկայացված JMX չափորոշիչների, գործառնությունների ցուցադրման համար: Մոնիտորինգի ենթարկված յուրաքանչյուր հանգույցի համար JConsole-ը տրամադրել է 6 ներդիր՝ տեղեկություններ Java VM-ի, հիշողության և թելերի օգտագործման, տեղեկատվություն դասի բեռնման և MBeans-ի մասին: Overview և Memory ներդիրները ևս պարունակել են տեղեկատվություն (նկ. 10): Cassandra չափումների և գործառնությունների արդյունքների համար կարևոր JConsole-ի MBeans ներդիրի org.apache.cassandra.auth-ն ներառել է թույլտվությունների քեշը, org.apache.cassandra.db-ը՝ քեշավորումը, աղյուսակի չափումները և սեղմումները, իսկ org.apache.cassandra. internal-ը ներքին սերվերի գործողությունները:



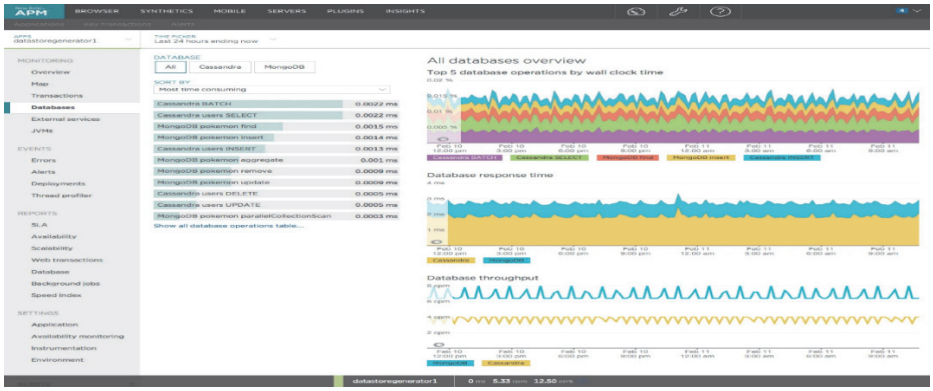
Նկ. 10. Cassandra-ի վերահսկման JConsole

Քանի որ JConsole-ի միջոցով Cassandra-ի վերահսկումը սպառում է համակարգի ռեսուրսների զգալի քանակություն, խնդրի լուծման համար JConsole-ը գործարկվել է հեռահար համակարգչի վրա և ոչ թե Cassandra-ի հանգույցի:

Կիրառվել է JConsole CompactionManagerMBean-ը՝ խտացման և կլաստերի հզորության ավելացման համար: Cassandra-ում գերբեռնման լուծման համար

եռաստիճան մեխանիզմի հետ համադրվել է նաև հորիզոնական մասշտաբավորման գաղափարը, որով իրականացվել են բեռի հավասարակշռումը և մուտքային հարցումների հավասարաչափ բաշխումը:

New Relic Cassandra Monitoring մոնիտորինգի ծրագրի (նկ. 11) միջոցով ստուգվել են կատարված աշխատանքի արդյունքները, JVM, օպերացիոն համակարգի չափումները և կլաստերը: Բեռի բաշխումը և գերբեռնվածության լուծումը առկա են:



Նկ. 11. New Relic Cassandra Monitoring ծրագրի միջերեսը

Եզրակացություն: Բացահայտվել են տվյալների գերբեռնվածության հիմքերը, կառավարվել և վերահսկվել է սերվերի մուտքի մեխանիզմը գերծանրաբեռնվածության պայմաններում՝ տվյալների պաշտպանության համար: Մշակվել են տվյալների գերբեռնման պայմաններում տվյալների անկորուստ պահպանման մեթոդները, և իրականացվել է մոնիտորինգ:

Հետազոտման հիման վրա մշակվել է մեխանիզմ ադմինիստրատորի համար՝ հետևելու, հսկելու և կառավարելու գերբեռնված սերվերի բեռի բաշխումը իրատեսական ժամանակում: Մշակվել է տվյալների կառավարման և գերբեռնվածություններից պաշտպանության միջոց Cassandra սերվերի համար:

ԳՐԱԿԱՆՈՒԹՅԱՆ ՑԱՆԿ

1. https://builtin.com/data-science/poisson-process.
2. **Yao H. and Knessl C.** On the Shortest Queue Version of the Erlang Loss Model 25.- University of Illinois at Chocago, 2007.
3. **YAO H. and Knessl C.** On the infinite server shortest queue problem: Non-symmetric case //Queueing Systems. – 2006.- 52. – P.157–177.
4. **Buttazzo G.C.** Hard real-time computing systems: Predictable Scheduling Algorithms and Applications.-Springer Science & Business Media, 2011. -524p.

5. https://docs.datastax.com/en/developer/java-driver/4.2/manual/core/load_balancing/
6. https://cassandra.apache.org/_/index.html.
7. https://learn.microsoft.com/ru-ru/azure/architecture/reference-architectures/n-tier/n-tier-cassandra.
8. https://docs-datastax-com.translate.goog/en/cassandra-oss/3.x/cassandra/dml/dmlHowDataMaintain.html?_x_tr_sl=auto&_x_tr_tl=hy&_x_tr_hl=ru&_x_tr_pto=wapp.

Г.О. САРГСЯН

УПРАВЛЕНИЕ ХРАНЕНИЕМ БОЛЬШИХ ДАННЫХ НА СЕРВЕРЕ В УСЛОВИЯХ ПЕРЕГРУЗКИ

Цель работы – выявление причин перегрузки данных, управление и контроль механизма доступа к серверу для защиты данных в условиях перегрузки. Разработан механизм выявления причин перегрузки данных, механизм управления целостностью данных доступа к серверу, стратегия мониторинга и контроля.

Механизмы балансировки и управления сервером являются ключом к обработке данных без потерь, чтобы предотвратить или уменьшить перегрузку данных. На основе исследования был разработан механизм, позволяющий администратору отслеживать, контролировать и управлять распределением нагрузки перегруженного сервера в режиме реального времени. Для сервера Apache разработан инструмент управления данными и защиты от перегрузок.

Ключевые слова: сервер Apache-Cassandra, интерактивная система, онлайн-технологии, управление, перегруженность.

G.H. SARGSYAN

MANAGING THE STORAGE OF BIG DATA ON A SERVER UNDER OVERLOAD CONDITIONS

The goal of the work is to identify the basis of data overload, manage and control the server access mechanism for data protection under overload conditions. Identification of data overload mechamism, the server access data integrity mechanism management, monitoring and control strategy are developed.

To prevent or mitigate data overload, balancing and server control mechanisms are the key to lossless data processing. Based on the research, a mechanism is developed for the administrator to track, monitor and manage the load distribution of an overloaded server in real time. A data management and congestion protection tool has been developed for the Apache server.

Keywords: Apache-Cassandra server, interactive system, online technologies, management, congestion.