HRVATSKI OGRANAK MEĐUNARODNE ELEKTRODISTRIBUCIJSKE KONFERENCIJE - HO CIRED
6. (12.) savjetovanje
Opatija, 13. - 16. svibnja 2018.

SO3 - 27

Branimir Gabrić
HEP ODS
branimir.gabric@hep.hr

Marko Penzar
HEP ODS
parko.penzar@hep.hr

PRIMJENA BAZE ZA POHRANU VREMENSKIH SERIJA U PROCESNIM SUSTAVIMA

SAŽETAK

U procesnim se sustavima svakodnevno prikupljaju velike količine podataka koji se obrađuju i pohranjuju u neku od povijesnih baza podataka. Često se prilikom izrade izvještaja zahtijevaju povijesni podaci unazad nekoliko godina. Dosadašnja rješenja spremanja podataka dosta su skupa u smislu cijena licenci i potrebnog memorijskog prostora. U tu smo svrhu testirali bazu za pohranu vremenskih serija kako bismo procijenili moguću uštedu. Korištena je InfluxDb baza otvorenog koda. Osim iznimne brzine obrade podataka koji u sebi sadrže vremensku oznaku, jedna od odlika je i pristupačnost, mjerne točke se unose i dohvaćaju putem web servisa. U ovom radu ćemo bazu testirati pod većim opterećenjem, pokazat ćemo koliki je omjer zauzeća diskovnog prostora u odnosu na klasičnu relacijsku bazu podataka. Također ćemo se pokušati dotaknuti inženjeringa nad ovakvim podacima korištenjem alata Python i R. U budućnosti bi prilikom zamjene ili nadogradnje pojedinih SCADA sustava svakako trebalo razmotriti jedno od ovakvih rješenja.

Ključne riječi: baza podataka, vremenske serije, Python, R, Datascience

USING TIME SERIES DATABASE IN PROCESS SYSTEMS

SUMMARY

In process systems large amounts of data are collected on a daily basis, which are then processed and stored in relational historical database. Creating reports often requires historical data from a few years ago. Existing solutions for data storage are quite expensive regarding licence cost and required memory space. We have implemented time series database for historical data in our SCADA system to estimate possible cost savings and memory optimisation. For this purpose, we have used InfluxDB database. One of advantages of a time series database is extraordinary fast retrieval of timestamped data. Due to smaller database size, hardware storage requirements are reduced and data retrieval is simplified. Database records can be acessed via web services. We will test time series database with large amount of data, and show ratio of the disk space usage compared to traditional relational database. We will also show how to manage data using Python and R tools. Given these test results, implementation of this solution in SCADA systems should be considered.

Key words: database, time-series, Python, R, Datascience

1. UVOD

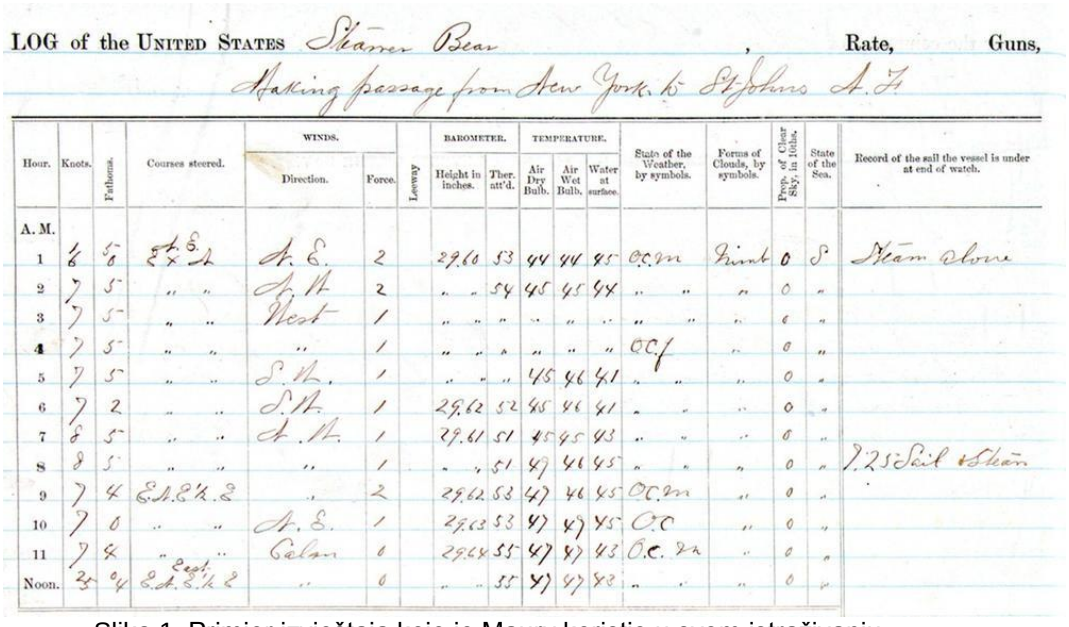
TSDB (engl. *Time Series Database*) je kratica za bazu podataka za pohranu vremenskih serija. Optimizirana je za rad sa serijama podataka koji su indeksirani po vremenu. Relacijske baze podataka nisu modelirane za manipulaciju s vremenskim serijama i to čini zahtjevnijim pretraživanje u svrhu pronalaska odgovarajućeg podatka s vremenskom oznakom. TSDB baze podataka su gotovi modeli koji ovakve probleme rješavaju i pružaju jednostavniji rad s takvim podacima.

TSDB baze podataka omogućuju kreiranje, numeriranje, ažuriranje i brisanje različitih serija podataka, te samu organizaciju tih podataka u ovisnosti o potrebama korisnika. Također su mogući i neki osnovni proračuni nad tim podacima poput množenja, zbrajanja pa i kombiniranja različitih skupova vremenskih podataka u nove skupove, a u svrhu statističkih proračuna.

1.1. Povijest baza podataka za pohranu vremenskih serija

Najpoznatiji primjer iz povijesti gdje je iskorištena prednost zapisa u vremenskim serijama je primjer mornara Matthewa Fontaine Maurya koji je živio polovicom 19. stoljeća. Zbog ozljede noge nije bio u mogućnosti nastaviti ploviti oceanima već je počeo raditi za stolom proučavajući navigaciju, meteorologiju, kretanje vjetrova i morskih struja. Analizirajući tisuće brodskih dnevnika i karti plovidbi te više od milijardu podataka nacrtao je karte vjetrova, morskih struja i vremenskih prilika koje su se počele koristiti diljem svijeta. Kapetani brodova imali su naviku voditi detaljan izvještaj tijekom putovanja: disciplinirani unos podataka: datuma, različitih mjerenja poput brzine broda u čvorovima, proračun geografskih dužina i širina, stanje oceana, vremenske prilike i ostalo, omogućilo je Mauryu da detaljnom analizom podataka dođe do optimalnih brodskih ruta u ovisnosti o vjetru i morskim strujama. Proučavajući njegove karte moreplovci diljem svijeta su naučili koristiti se morskim strujama i vjetrom kako bi umanjili duljinu putovanja (v. Sliku 1.) Prvi je to primjer nečega što bi informatičari danas zvali analizom velikih podataka (engl. *Big Data Analysis*).

S obzirom na nove tehnologije i sve veće uporabe raznih senzora, kojih danas ima više od sveukupne ljudske populacije, a koji kreiraju upravo podatke ovog tipa, bilo je neophodno razviti i alate koji će na jednostavan način pohranjivati te podatke te omogućiti kasniju manipulaciju tih podataka.[1]



LOG of the UNITED STATES Steamer Bear , Rate, Guns,

Making passage from New York to St Johns N.F.

Hour.	Knots.	Fathoms.	Courses steered.	WINDS. Direction.	Force.	Leeway	BAROMETER. Height in inches.	Ther. att'd.	TEMPERATURE. Air Dry Bulb.	Air Wet Bulb.	Water at surface.	State of the Weather, by symbols.	Forms of Clouds, by symbols.	Prop. of Clear Sky, in 10ths.	State of the Sea.	Record of the sail the vessel is under at end of watch.
A. M.																
1	6	5 6	S.E.½S	N. E.	2		29.60	53	44	44	45	O.C.M	Print	0	S	Steam slow
2	7	5	" "	N.N.E	2		" "	54	45	45	44	" "	"	0	"	
3	7	5	" "	West	1		" "	"	"	"	"	" "	"	0	"	
4	7	5	" "	"	1		" "	"	"	"	"	O.C.	"	0	"	
5	7	5	" "	S.W.	1		" "	"	45	46	41	" "	"	0	"	
6	7	2	" "	S.W.	1		29.62	52	48	46	41	" "	"	0	"	
7	8	5	" "	S.W.	1		29.61	51	45	45	43	" "	"	0	"	
8	8	5	" "	"	1		" "	51	47	46	45	" "	"	0	"	7.25 Sail + Steam
9	7	4	E.N.E.½E	"	2		29.62	53	47	46	45	O.C.M	"	0	"	
10	7	6	" "	N. E.	1		29.63	53	47	47	45	O.C	"	0	"	
11	7	4	" 2 cast "	Calm	0		29.64	55	47	47	43	O.C. M	"	0	"	
Noon.	25	04	E.N.E.½E	"	0		" "	55	47	47	43	" "	"	0	"	

Slika 1. Primjer izvještaja koje je Maury koristio u svom istraživanju

1.2. Primjena

Bilježenje točnog vremena kada je kritični parametar izmjeren ili kada se neki događaj dogodio, može imati veliki utjecaj na ozbiljne sigurnosne situacije kao i na predviđanje i smanjenje rizika. Zrakoplovna industrija je jedan od primjera gdje se TSDB naveliko primjenjuje. Ovdje se koriste

takozvane „Crne kutije“ koje se koriste pri rekonstrukciji događaja nakon nekog kvara ili u najgorem slučaju - rušenja zrakoplova. Moderni su avioni opremljeni gomilom senzora koji cijelo vrijeme mjere podatke. Mjeri se primjerice: visina, putanja, nagib, temperatura, snaga motora, brzina, potrošnja goriva i sl. Svako se mjerenje bilježi uz vrijeme, a na temelju tih podataka može se izvršiti točna rekonstrukcija bilo kojeg događaja. Sve je to moguće upravo zato jer je evidencija zapisana u obliku vremenskih serija. Uz samu rekonstrukciju događaja, ovi podaci služe i proizvođačima za optimizaciju i poboljšanje performansi leta, kao i održavanja sigurnosti na najvišem nivou.

1.3. Popis TSDB

U idućoj tablici je napravljen pregled najčešće korištenih baza za pohranu vremenskih serija. [1]

Tablica I. Primjeri TSDB

Tip	Cijena	Alati	Opis
Riak TS	Free & License version	Java, Ruby, Python, Erlang, Node.js	Jedina NoSQL baza podataka optimizirana za IoT i vremenske serije podataka, koja ima osobine neophodne za podršku korisnicima sa jedinstvenim zahtjevima (enterprise-grade)
InfulxDB	Free	JavaScript, Ruby, Python, Node.js, PHP, Java, Clojure, Common Lisp, Go, Scala, R, Erlang, Perl, Haskell, .NET	Baza vremenskih serija, metričkih i analitičkih podataka. Pisana je u GO-u i nema eksternih ovisnosti.
TempoIQ	Subscription	.NET, Java, Node.js, Python, Ruby	Brzo, skalabilno nadgledanje i analiza podataka, dobivenih raznim mjerenjima sa senzora
Axibase TSDB	Free & License version	Java, R Language, PHP, Python, Ruby, JavaScript	Statistička baza podataka. Namijenjena za pristup velikim količinama vremenskih serija
INFINIFLUX	Free & License version	Java, Python, JavaScript, R, PHP	Najbrži Time Series DB za upravljanje, za IoT i BigData
OpenTSDB	Free	Java	Dobro razrađena besplatna baza na koju sigurno treba obratiti pozornost.
PI Server	Commercial		Profesionalna komercijalna baza podataka za pohranu podataka u vremenskim serijama.

2. INFLUXDB

InfluxDB je baza podataka vremenskih serija koja je kreirana s ciljem pohrane velike količine podataka. Kreirana je od strane InfluxData 2013. godine. Pisana je u programskom jeziku GO i optimizirana je za brzu manipulaciju podacima u bazi.

Prednosti InfluxDB-a:

- Prilagođena baza podataka, visoke performanse, pisana prvenstveno za spremanje vremenskih serija,
- zapis u binarne podatke koji ne sadrži dodatne ovisnosti,
- jednostavan, visoke performanse HTTP(S) API-a,
- razni dodaci (*Plugin*-ovi) daju podršku i za druge protokole,

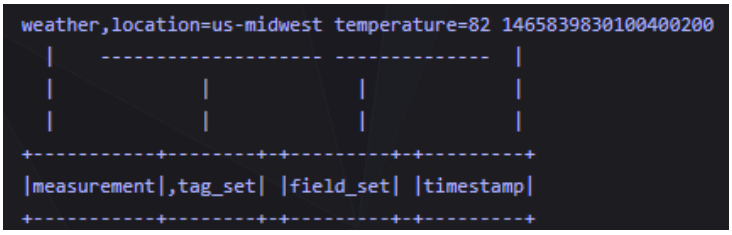
- upiti se pišu u obliku SQL upita koji omogućava jednostavnu manipulaciju podacima,
- Tag-ovi omogućuju indeksiranje podataka, a služe za brže i efikasnije upite te
- automatski upiti za izračun podataka.

Čitavo joj poglavlje posvećujemo upravo zato što smo upravo nju rabili u našem zadatku.

2.1. Struktura InfluxDB podataka

InfluxDB je baza podataka vremenskih serija u kojoj je ključ svega upravo vrijeme. Svi podaci zapisani u InfluxDB bazi imaju kolonu vrijeme (*time*) u kojoj su pohranjene vremenske oznake. Zatim slijede polja (*Fields*) s ključevima i vrijednostima, koja nisu indeksirana pa se pretražuju na način da se slijedno prolazi kroz svaki zapis. Zatim slijede oznake (*Tags*) koje su neobvezne i također se sastoje od ključa i vrijednosti, ali su, za razliku od polja, indeksirane. To znači da se upiti po oznakama brže izvršavaju pa su oznake idealne za pohranu metapodataka koji se najčešće traže. Polja su idealna za pohranu mjerenih veličina (vrijednost struje, vrijednost temperature i sl.)

Podaci se upisuju u bazu koristeći tekstualni format Line Protocol, sintaksu prikazuje Slika 2.



Slika 2. Line Protocol

U nastavku je kratko objašnjenje svake stavke Line Protocola:

- **measurement**: naziv mjerenja u koje se podaci upisuju, npr. mjerUDWstruja.
- **tag_set**: popis oznaka, npr. kratkiNazivMj, dugiNazivMj, SifraPog i sl.
- **field_set**: popis polja, npr. vrijednost i sl.
- **timestamp**: Unix vrijeme s nano preciznošću.

2.2. Osnovni upiti na InfluxDB bazu

InfluxDB koristi jezik nalik SQL-u za manipulaciju nad bazom. Kreiran je tako upravo iz razloga kako bi svi oni koji su upoznati s SQL-om mogli lagano započeti raditi i s InfluxDB bazom.

Ispis svih stavki mjerenja *mjerUDWstruja*:

```
select * from mjerUDWstruja
```

Upit uz vremenski uvjet:

```
select * from mjerUDWstruja where time > '2018-01-15 12:00:01'
```

Primjer brisanja zapisa:

```
DELETE from mjerUDWstruja where kratki='013T710J4STRUJA'
```

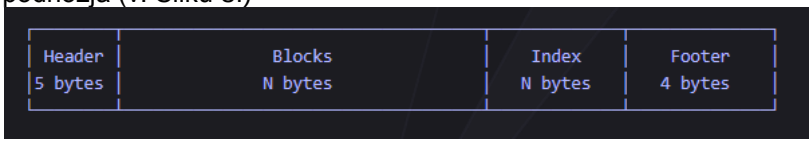
InfluxDB je odlično dokumentiran na internetu, poveznica se nalazi u popisu literature [2].

Za napomenuti je da ćemo u nastavku teksta ponekad dodati pokoji SQL upit s namjerom da čitatelja bolje upoznamo s korištenjem baze.

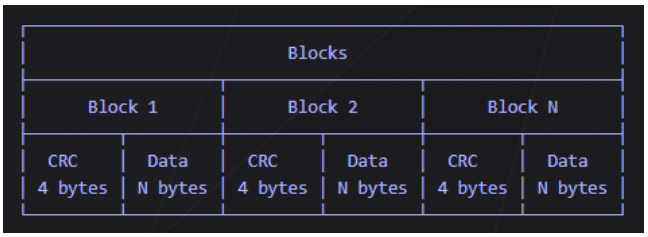
2.3. Pohrana i kompresija podataka

InfluxDB struktura za pohranu podataka nalikuje strukturi LSM stabla (engl. Log-structured merge tree), gdje se sprijeda nalazi zapis (*log*), a zatim kolekcija *read-only* datoteka. Ovdje se radi o vremenski

strukturiranim TSM (Time-Structured Merge Tree) datotekama. TSM datoteke sadrže sortirane komprimirane serije podataka. Kompresija se vrši različitim koderima i dekoderima, ovisno o tipovima podataka. Neki od kodera su statični i uvijek kodiraju podatke na isti način, dok su drugi varijabilni i mijenjaju način kodiranja prema tipu podatka. TSM datoteke su kolekcije read-only datoteka mapiranih u memoriji. Struktura TSM datoteka nalikuje LSM Tree strukturi i sastoji se od četiri dijela: zaglavlja, blokova, indeksa i podnožja (v. Sliku 3.)



Slika 3. Struktura TSM zapisa



Slika 4. Struktura bloka TSM zapisa

Svaki je blok komprimiran kako bi se smanjilo zauzeće prostora za pohranu (na Slici 4.). Blok sadrži vremenske oznake i vrijednosti za određenu seriju i polje. Svaki blok ima zaglavlje (1 Byte), nakon čega slijede komprimirane vremenske oznake i komprimirane vrijednosti. Vremenske se oznake komprimiraju i pohranjuju odvojeno od vrijednosti, tako da za sve vremenske oznake imamo isti koder, pak koderi za same vrijednosti ovise o tipu podatka i mogu varirati. Vremenske se oznake kodiraju kombinacijom: delta kodiranja, skaliranja i kompresije pomoću *simple8b run-length* metode. *Float* zapisi kodiraju se pomoću *Facebook Gorilla* metode gdje se nad uzastopnim bliskim vrijednostima vrši XOR funkcija, a pohranjuju se samo razlike. *Integer* vrijednosti pohranjuju se ovisno o rasponu samih vrijednosti nekomprimiranih podataka. Prvo se koristi *ZigZag* kodiranje kako bismo dobili sve vrijednosti u pozitivnom *Integer* rasponu, nakon čega slijedi jedna od metoda komprimiranja: *simple8b*, *run-length* metoda ili ne komprimiranje. *Boolean* podaci kodiraju se strategijom kodiranja pojedinog bita gdje će svaki boolean zauzeti po jedan bit. *String* zapis kodira se pomoću *Snappy* kompresije gdje se svaki *String* pakira uzastopno, a zatim se svi komprimiraju u jedan veći blok. Autori InfluxDB-a u prijašnjim su verzijama pokušali koristiti različite metode komprimiranja i strukture podataka (*LSM Tree* i *B+ Tree*), te su u konačnoj verziji razvili svoju vlastitu strukturu i kompresiju (*TSM Tree*) i time postigli smanjenje memorijskog zauzeća i do 45 puta.[2]

2.4. Vrijeme zadržavanja podataka

InfluxDB se može nositi sa stotinama tisuća pristiglih podataka u sekundi, međutim zadržavanje tolike količine podataka na dulje vrijeme može zadati probleme. Prirodno je rješenje skratiti vrijeme uzorkovanja, zadržati podatke visoke preciznosti na kraće vrijeme, a pohranjivati podatke niže preciznosti na dulje vrijeme. InfluxDB je u tu svrhu razvio dvije opcije: kontinuirani upiti (engl. *Continuous Queries*, skraćeno CQ) i pravila zadržavanja podataka (engl. *Retention Policies*, skraćeno RP) koji automatiziraju uzorkovanje i čuvanje podataka. Kontinuirani upit (CQ) je InfluxQL upit koji se pokreće automatski i periodički unutar same baze podataka. Pravila zadržavanja (RP) su dio same strukture podataka i opisuju koliko će dugo InfluxDB baza čuvati podatke. InfluxDB uspoređuje lokalno vrijeme na poslužitelju i vremenske oznake podataka te briše podatke starije od definiranog vremenskog intervala unutar RP pravila (*Duration*). Sama baza može imati nekoliko različitih pravila, ovisno kako je definirano.

Primjer jednostavnog kreiranja pravila zadržavanja RP:

```
> CREATE RETENTION POLICY "two_hours" ON "mjerenje" DURATION 2h REPLICATION 1 DEFAULT
```

Jednom linijom naredbe definira se osnovno pravilo kojim se sirovi podaci čuvaju dva sata, a InfluxDB automatski ovdje zapisuje primjerice deset sekundne vrijednosti.

Kreiranje drugog pravila:

```
> CREATE RETENTION POLICY "a_year" ON "mjerenje" DURATION 52w REPLICATION 1
```

Drugim se pravilom npr. definira pohrana od 52 tjedna, gdje će se pohranjivati 30-minutne vrijednosti.

Potrebno je još samo definirati kontinuirani upit koji će automatski periodički pohranjivati 10 sekundne vrijednosti u 30-minutnu rezoluciju i pohranjivati rezultate u drugu tablicu s drugačijim uzorkom čuvanja podataka.

Primjer kreiranja kontinuiranog upita:

```
> CREATE CONTINUOUS QUERY "cq_30m" ON " mjerenje " BEGIN
  SELECT mean("struja") AS "SR_VR_STRUJA",mean("Q") AS "SR_Q"
  INTO "a_year"."padajuci"
  FROM "vrijednost"
  GROUP BY time(30m)
END
```

Kao rezultat dobijemo dvije odvojene tablice: tablicu sirovih vrijednosti koje pohranjuju dva sata 10-sekundnih vrijednosti te godinu dana srednjih 30-minutnih vrijednosti. Kombinacijom odgovarajućih pravila zadržavanja i kontinuiranih upita vrlo se jednostavno izrađuje baza koja podatke visoke preciznosti pohranjuje kraće vrijeme, dok izračunate srednje vrijednosti pohranjuje na dulje staze.[2]

3. TESTIRANJE BAZE

U nekoliko idućih potpoglavlja pokušat ćemo usporediti konvencionalnu relacijsku bazu Oracle 11g s InfluxDb 1.3. Podaci koje ćemo koristiti za potrebe testiranja su povijesni podaci iz zagrebačkog SCADA sustava pohranjeni u Oracle bazi na Linuxu.

Koristit ćemo 10 sekundna mjerenja struje u posljednja 2 dana, koliko se pohranjuju u *online* bazu, potom 18 mjesečni period 15 minutnih vrijednosti usrednjenih struja. Pokušat ćemo pohraniti i događaje koji se također nalaze u Oracle povijesnoj bazi.

Oracle baza se nalazi na moćnom poslužitelju iz 2014. godine (poslužitelj A). Testno Windows računalo je također moćno stolno računalo iz 2014., (Intel Xeon e5-1620 iz 2012. godine s 8 jezgri, 16 GB memorije - računalo B). Testno Linux računalo je slabije računalo iz 2011. godine (Intel Xeon W3520 iz 2009. godine s 4 jezgre, 4GB memorije - računalo C). Za potrebe testiranja InfluxDb je instaliran na Windows (B) i Linux (C) računalo.

3.1. Prijenos 10 sekundnih mjerenja iz Oracle baze u InfluxDB

Kako bi dohvatili 1322 mjerenja struje, 10 sekundnog uzorka u vremenskom rasponu od dva dana, napravljena je SQL skripta koja sve podatke u tablici pohranjuje u CSV datoteku sa sljedećim kolonama: *Vrijeme*, *Vrijednost*, *Kvaliteta*, *Naziv mjerenja*. Skripti je potrebno 18 minuta za izraditi tekstualnu datoteku veličine 1,6 GB (225 MB u ZIP-u).

Tablica u Oracle bazi zauzima 838 MB. U Javi 8 je izrađena aplikacija koja ima mogućnost čitanja ogromnih datoteka u segmentima od, u ovom slučaju, 3000 linija. Učitane linije pretvara u InfluxDB *Line Protokol* te palje u InfluxDB bazu koristeći HTTP protokol, POST metodom. Ovako primjerice izgleda jedna linija tzv. *Line Protocola*:

```
mjerUDWstruja,kratki=014T9110E1STRUJA value=44.0834850311279,qual=1 1466561700000000000
```

Java aplikacija je učitala 29.143.034 mjerenja i pohranila ih u InfluxDB, trajalo je nekih 12 minuta. InfluxDB baza je narasla na 38,4 MB.

U nastavku je testirana brzina odziva na način da su zatražene sve vrijednosti jednog mjerenja. Oracle bazi je potrebno 1,84 sekunda za dohvatiti 22.000 redaka (rač. A). Isto toliko redaka InfluxDB-u na računalu B je potrebno 1,3 sekundi te InfluxDB-u na računalu C oko 600 ms.

3.2. Prijenos 15 minutnih iz Oracle baze u InfluxDB

Za dohvatiti 15 minutne vrijednosti 1322 različitih mjerenja struje u vremenskom rasponu od 18 mjeseci, poslužila je slična SQL skripta. Skripti je u ovom slučaju bilo potrebno oko 36 minuta za izraditi tekstualnu datoteku veličine 3,5 GB (568 MB u ZIP-u).

Tablica u Oracle bazi zauzima otprilike 1964 MB prostora.

Java aplikacija je učitala 70.558.852 mjerenja i pohranila ih u InfluxDB i trajalo je to nekih 36 minuta. InfluxDB baza nakon unosa ima 280 MB.

3.2.1. Dohvat jednog mjerenja

Oracle bazi je potrebno 15,5 sekundi za dohvatiti 54.570 redaka (rač. A). Isto toliko redaka na InfluxDB računalu B je potrebno nešto kraće od 3 sekunde te na InfluxDB računalu C oko 1,3 sekunde kako bi ih pohranio na disk.

3.2.2. Dohvat jednog mjerenja unutar vremenskog raspona

Vremenski raspon od 20 dana (između 100-tog i 80-tog dana) Oracle bazi je potrebno 100 ms za dohvatiti oko 1900 redaka (rač. A). Isto toliko redaka na računalu B je potrebno nešto kraće od 100 ms te na računalu C oko 95 ms.
Korišteni upit u InfluxDB:

```
select * from mjerUDWstruja where kratki='013T710J4STRUJA' and time > (now()-
100d ) and time <( now() - 80d);
```

3.3. Prijenos događaja iz Oracle baze u InfluxDB

U Oracle tablici se nalazi oko 1,2 milijuna događaja koji pokrivaju vremenski raspon od 18 mjeseci koji zauzimaju 331 MB. Kada se tekst događaja i vrijeme pohrane u CSV datoteku, ova zauzima 175 MB (12,8 MB ZIP).

Koristeći Java aplikaciju događaji su uneseni u InfluxDB. Kao rezultat InfluxDB sadrži oko 639 tisuća redaka. Manje, jer mnogo događaja ima istu vremensku oznaku te noviji događaj zamjenjuje prethodni, ali za potrebe testa nije otegotno.

Zauzeće InfluxDB baze s tom količinom događaja iznosi 9,49 MB. Kada se svi događaji iz InfluxDB pohrane kao izlaz u tekstualnu datoteku, ova zauzima oko 100 MB (6,3 MB ZIP).

```
select * from dogaUDW where tekst =~ /.*ZEMLJ.*/
```

Gornji će upit dohvatiti rezultat za otprilike 9 sekundi, dakle sve događaje koji u sebi sadrže znakove „ZEMLJ“. Isto toliko vremena otprilike treba i Oracle-u za filtrirati poruke koje sadrže „%ZEMLJ%“ u većem setu podataka od 1,2 milijuna redaka.

3.4. Zaključak testiranja

Testiranjem su zamijećene neke interesantnosti. Postoji nelinearnost u pohrani 10 sekundnih i 15 minutnih vrijednosti. Primjerice, 29 milijuna 10 sekundnih mjerenja u InfluxDB bazi zauzima oko 38 MB, dok 70 milijuna 15 minutnih vrijednosti u InfluxDB zauzima oko 280 MB. Kako bi se to objasnilo, mora se nešto reći o podacima koji dolaze iz procesa. Često se dogodi da se s mjernog uređaja dulje vremena ne mijenja vrijednost; uređaj je podešen tako da šalje novu vrijednost samo ako se ista promijenila za neki postotak. Iz toga razloga 29 milijuna je vrijednosti, ali ih ima dosta istih. Primjerice za jedno mjerenje koje ima 22 tisuće redaka, tek je 10 različitih vrijednosti. InfluxDB izuzetno efikasno pohranjuje opetovane vrijednosti. 15 minutne vrijednosti nastaju usrednjavanjem baznih vrijednosti i imaju manje ponavljanja. Isto odabrano mjerenje struje koje u 15-minutnoj tablici sadrži oko 56 tisuća redaka ima 11 tisuća različitih vrijednosti. Iz toga razloga je shvatljivije zašto za nešto više od dva puta veći broj redaka povećava bazu za skoro 10 puta. Makar i to povećanje do 10 puta je bitno manje prostora na disku nego što bi to bilo u relacijskoj bazi.

Pristup mjerenjima, vidi se iz priloženoga, bitno je brži za dohvat veće količine podataka, 15 sekundi treba Oracleu za dohvatiti 56 tisuća redaka jednog mjerenja dok na slabijem Linuxu tek 1,3 sekunde. Kad je riječ o dohvatu manje količine podataka, rezultat je otprilike isti.

Pohrana tekstualnih poruka s vremenskom oznakom isto ide u prilog InfluxDB čija baza zauzima oko 9 MB za 600 tisuća događaja. Vrijeme pretraživanja je usporedivo kod obiju baza.

4. DOHVAT PODATAKA KORISTEĆI POPULARNA PROGRAMSKA SUČELJA

U ovom radu opisujemo i prilažemo izvatke koda za zapisivanje i dohvaćanje podataka u InfluxDB koristeći popularna programska sučelja. Programski kod navodimo jer sličnog,

pojednostavljenoga nema lako dostupnog na Internetu. Obradit će se sučelja u C#, Java, Python i R jezicima i to na primjeru unosa i dohvata jednostavnog mjerenja koje će kao oznake (*Tag*) imati *naziv* i *pogon*, a kao polja (*Field*) *vrij* i *kval*; mjerenje (*measurement*) će se zvati *mjertest*, a baza će biti *mojabaza*.

4.1. C#

Sučelje je testirano koristeći Visual Studio 2017, .NET Framework 4.6.1. Korišten je paket s NuGet-a InfluxData.Net. Nakon instalacije NuGet paketa, potrebno je uključiti biblioteke:

```
using InfluxData.Net.InfluxDb;
using InfluxData.Net.Common.Enums;
```

Unos mjerene točke u bazu:

```
private async Task<int> PisanjeTockeAsync()
        {
            var influxDbClient = new InfluxDbClient("http://10.7.130.66:8086/", "", "",
InfluxDbVersion.v_1_3);

            var pointToWrite = new Point()
            {
                Name = "mjertest", // serie/measurement/table to write into
                Tags = new Dictionary<string, object>()
                {
                    { "naziv", "mjerenje1" },
                    { "pogon", "4001" }
                },
                Fields = new Dictionary<string, object>()
                {

                    { "vrij", 22.2 },
                    { "kval", 1}
                },
                Timestamp = DateTime.UtcNow // optional (can be set to any DateTime moment)
            };

            var response = await influxDbClient.Client.WriteAsync(pointToWrite, "mojabaza");

            return 1;
        }
```

A potom:

```
var rezultat = await PisanjeTockeAsync();
```

Dohvat podataka:

```
    private async Task<List<string>> DohvatiPodatkeAsync()
        {
            var    influxDbClient    =    new    InfluxDbClient("http://10.7.130.66:8086/",    "",    "",
InfluxDbVersion.v_1_3);
            var query = "SELECT * FROM mjertest"; // WHERE time > now() - 1h";
            var response = await influxDbClient.Client.QueryAsync(query, "mojabaza");
            List<string> zaVan = new List<string>();
            foreach (var k in response)
            {
                string kolone = "";
                foreach (var kk in k.Columns)
                {
                    kolone += kk+";";
                }
                foreach (var kk in k.Values)
                {
                    string values = "";
                    foreach (var kkk in kk)
                    {
                        values += "(" + kkk.ToString() + "),";
                    }
                    zaVan.Add(k.Name + ";" + kolone + ";" + values);
                }
            }
```

```
    return zaVan;
}
```

Konačno:

```
var rezultat = await DohvatiPodatkeAsync();
            foreach (var k in rezultat)
            {
                listBox1.Items.Add(k);
            }
```

4.2. Java

Korišten je paket InfluxDB-Java. Na Mavenu je potrebno pronaći sljedeće biblioteke: *converter-moshi-2.3.0.jar, influxdb-java-2.8.jar, logging-interceptor-3.9.1.jar, moshi-1.5.0.jar, okhttp-3.9.1.jar, okio-1.13.0.jar, retrofit-2.3.0.jar*.
Ubaciti ih u projekt. Sljedeći kod će napraviti bazu i pohraniti jednu točku:

```
InfluxDB influxDB = InfluxDBFactory.connect("http://10.7.130.66:8086", "root", "root");
String dbName = "mojabaza";
influxDB.createDatabase(dbName);
Point point1 = Point.measurement("mjertest")
                .time(System.currentTimeMillis(), TimeUnit.MILLISECONDS)
                .tag("naziv", "mjerenje1")
                .addField("vrij", 22.2f)
                .addField("kval", 1L)
                .build( );
influxDB.setDatabase(dbName);
influxDB.write(point1);
```

Ovako se podaci mogu dohvatiti:

```
Query query = new Query("SELECT * FROM mjertest", dbName);
 QueryResult rezultat =  influxDB.query(query);
 System.out.println(rezultat.toString());
```

4.3. Python

Sučelje je testirano koristeći Python 3.6.1. Potrebno je instalirati Influxdb, koristeći PIP: pip install influxdb

```
from influxdb import InfluxDBClient
import datetime
from influxdb import InfluxDBClient


sada = datetime.datetime.now()
za_u_bazu = []
data = {}
tags = {}
tags['naziv'] =  'mjerenje1'


fields = {}
fields['vrij'] = 28.1
fields['kval'] = 2



data['measurement'] = "mjertest"
data['tags'] = tags
data['fields'] = fields
data['time'] = sada
za_u_bazu.append(data)
client = InfluxDBClient(host="10.7.130.66", port=8086, database="mojabaza")
client.write_points(za_u_bazu)
```

Za dohvat podataka:

```
result = client.query("select * from mjertest")
```

```
print("Rezultat: {0}".format(result))
```

4.4. R

Potrebno je imati najnoviju inačicu RStudia (1.1.383) i R-a (Microsoft R Open 3.4.2). U konzoli je potrebno najprije instalirati paket: *Install.packages(„influxdbr“).* A potom u R skripti:

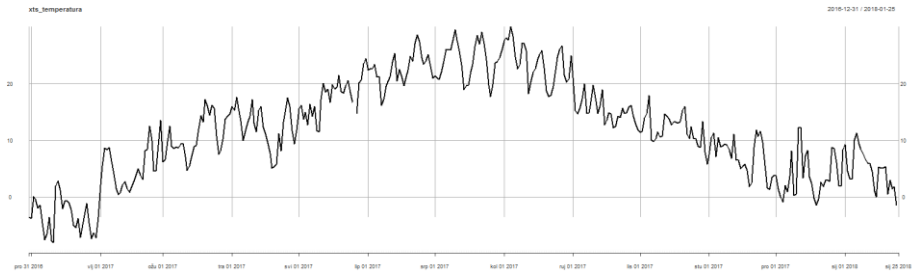
```
library(influxdbr)
library(xts)
pocetnoVrijeme = strptime('2012-09-14', format = '%Y-%m-%d')
vremena <- seq(pocetnoVrijeme, length = 3, by = "hours")
con <- influx_connection("http", "10.7.130.66", 8086 )
show_databases(con)
vrij <- c(22.3,23.1,24.7)
kval <- c(1,2,1)
df_fields = data.frame(vrij, kval)

podaci <- xts(x = df_fields, order.by = vremena, naziv='mjerenje1', pogon='4001')
influx_write(con=con, db="mojabaza", xts=podaci, measurement = "mjertest")
```

Dohvat podataka:

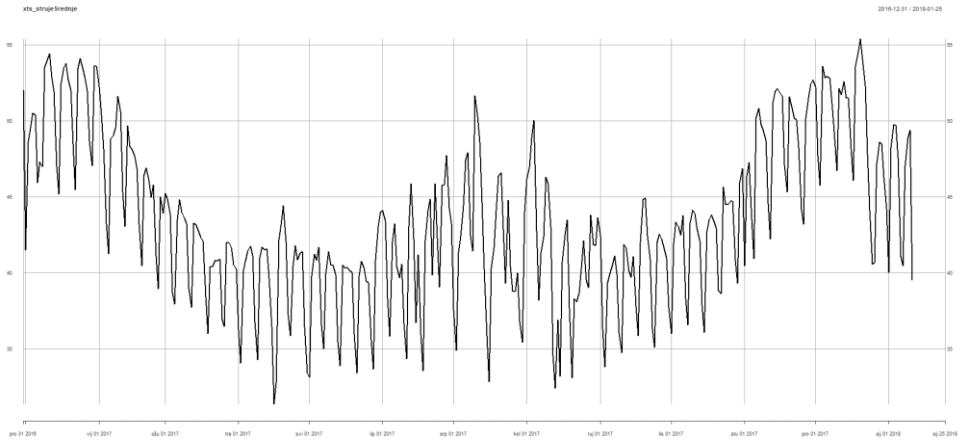
```
rezultat <- influx_query(con=con, db="mojabaza", query="select * from mjertest", return_xts = TRUE)
```

4.5. Primjer za kraj

S prilagođenog DHMZ poslužitelja dohvaćeni su meteorološki podaci DP-ova za 2017. godinu, satne vrijednost (uključen je i kraj 2016. te početak 2018.) Informacije su to o temperaturi, vjetru, vlagi, naoblaci i sl. Testna aplikacija je napravljena u *Pythonu*. Aplikacija analizira ulazni format (XLSX koristeći *openpyxl*) i podatke koristeći *line protocol* HTTP POST-om šalje u Influxdb.
Kao rezultat, u bazi se nalazi oko 121 tisuća zapisa. Podaci se naknadno dohvaćaju i obrađuju koristeći R. Dva su seta podataka. U prvom se nalazi temperatura zagrebačkog područja za vremenski raspon od 390 dana (od sredine prosinca 2016. do sredine siječnja 2018. godine.) U drugom setu podataka se nalaze usrednjene vrijednosti struja (*mean*) na srednjenaponskim vodnim poljima svih zagrebačkih trafostanica, također za 390 dana. Podaci se pohranjuju u XTS objektu (memorijska struktura pogodna za prihvat podataka s vremenskim oznakama) unutar *RStudio* razvojnog sučelja. Rezultat su dva grafa koja su prikazana Slikama 5. i 6.



Slika 5. Prikaz temperatura za područje Grada Zagreba, od kraja 2016. do sredine siječnja 2018.



Slika 6. Usrednjene vrijednosti struja svih SN vodnih polja u Zagrebu

Uspoređujući ova dva grafa, daju se izvući neki zaključci. Dva su profila, za toplije i hladnije razdoblje u godini. U hladnijem razdoblju, kada je temperatura niža, usrednjene struje (opterećenje) imaju veću vrijednost. U toplijem razdoblju, opterećenje raste s temperaturom (klima uređaji). Propadi u grafu usrednjih struja predstavljaju vikende. Također se vidi da je početkom hladnijeg siječnja 2017. godine opterećenje bilo veće od toplijeg početka siječnja 2018. Primjer koda u R:

```
upit = "SELECT mean(value) FROM mjerUDWstruja WHERE kratki='temper'and time > now()-390d group by
time(1d)"
rezultat <- influx_query(con=con, db="Mjerenja", query=upit, return_xts = TRUE)
xts=rezultat[[1]]
xts_temperatura= xts[[1]]
plot(xts_temperatura)
```

Za napomenuti je da se do rezultata dolazi iznimno brzo; ne čeka se dulje od dvije sekunde.

5. ZAKLJUČAK

Prvi put smo za ovakvu bazu čuli na prezentaciji američke firme Osisoft. Naknadno istražujući, naišli smo na modernu *opensource* bazu za pohranu vremenskih serija - InfluxDb. Bazu je vrlo lagano upogoniti na Linux računalu. Može poslužiti kao alternativna baza podataka u koju se mogu slobodno slati podaci koji, kao važan faktor, sadrže vremensku oznaku. Ti podaci čak i ne moraju biti pretjerano strukturirani. Ideja je ostaviti podatke da se slobodno nakupljaju, a potom korisnik s vještinama obrade podataka (svojevrsni *datascience* stručnjak) iz tih podataka naknadno izvlači zaključke. Izazov je zapravo procijeniti koje podatke pohranjivati.

Iz ovoga rada se vidi kako ovakav oblik zapisa zauzima bitno manje mjesta diskovnog prostora, u najgorem slučaju riječ je o 8 puta manjem prostoru u odnosu na Oracle bazu. Koristeći zaključke iz ovoga rada može se reći da će sve 15 minutne vrijednosti, iz svih hrvatskih SCADA sustava (preko 26.000 mjerenja) zauzimati oko 36 GB/10 godina.

Razvojna InfluxDB baza je podignuta na starom, napuštenom serverskom poslužitelju iz 2004. godine pomlađenom besplatnim modernim OS-om Ubuntu Server 16. Računalo ima 300 GB diskovnog prostora u RAID-u. Napravljena je Python skripta koja je svakih 15 minuta preko SCADAinfoMMI servisa (vidjeti rad s ovogodišnjeg CIRED-a na temu „Novosti u aplikacijama SCADAinfo“) ubrizgavala SCADA mjerenja u InfluxDB, 120-tak dana bez prestanka. Ideja je u ovu bazu ubaciti i mjerenja sa sučelja HOPS-ODS, mjerenja iz pametnih brojila, mjerenja temperature i sl.

U budućnosti, usavršavanjem *datascience* vještina, moći će se iz tih podataka, uvjereni smo, izvlačiti zaključci korisni u poslovnom odlučivanju.

6. LITERATURA

[1] A. Lisovac, F. Ademović, A. Tahić, M. Brkić, "Time in Databases: how to manage historical data“, Univerzitet u Sarajevu, Elektrotehnički fakulte, 12. 2016.

[2] InfluxDB 1.4 documentation, https://docs.influxdata.com/influxdb/v1.4/