ŠIAULIŲ UNIVERSITETAS
Technologijos, fizinių ir biomedicinos mokslų fakultetas
Kompiuterių sistemų katedra

Tomas Šeirys

Išankstinio potencialiai vartotojo peržiūrimų internetinių sistemų duomenų talpinimo į podėlį sprendimas

Magistro darbas

Vadovė dr. S. Ramanauskaitė

Šiauliai, 2016

ŠIAULIŲ UNIVERSITETAS

Technologijos, fizinių ir biomedicinos mokslų fakultetas

Kompiuterių sistemų katedra

TVIRTINU

Kompiuterių sistemų katedros vedėjas
dr. E. Paliulis
2016-06-01

Išankstinio potencialiai vartotojo peržiūrimų internetinių sistemų duomenų talpinimo į podėlį sprendimas

Informatikos inžinerijos magistro darbas

Autorius
ITM-14 gr. magistrantas T. Šeirys
2016 m. gegužės 25 d.

Vadovė
Kompiuterių sistemų katedros docentė dr. S. Ramanauskaitė
2016 m. gegužės 25 d.

Recenzentai
Kompiuterių sistemų katedros docentas dr. E. Paliulis
2016 m. birželio __ d.
Kompiuterių sistemų katedros lektorė dr. A. Drukteinienė
2016 m. birželio __ d.

Šiauliai, 2016

SANTRAUKA

Išankstinio potencialiai vartotojo peržiūrimų internetinių sistemų duomenų talpinimo į podėlį sprendimas

Internetinių tinklalapių greičio optimizavimui dažnai naudojamas podėlis, kuriame talpinami pakartotinai panaudojami resursai, kuriuos iš podėlio naudoja didelis kiekis vartotojų. Šiame darbe apžvelgiami egzistuojantys podėlio tipai, jų specifika ir išsikeltas tikslas padidinti internetinės sistemos atsako vartotojui greitį, pritaikant išankstinio individualaus podėlio sudarymo metodą. Siūlomo metodo tikslingumui įvertinti atlikti eksperimentai, kurių metu analizuojama kaip keičiasi vidutinis tinklalapio su kritinėmis sekcijomis laikas kuomet visiškai netaikomas podėlis ir kuomet taikomas pilnas podėlis. Tyrimo metu gauti rezultatai atskleidė podėlio poreikį, o jo adaptavimui realioje situacijoje pasiūlytas naujas dalinio podėlio sudarymo metodas. Šis metodas leidžia nuspėti sekančius vartotojo lankomus puslapius ir į podėlį talpina tik juos. Pasiūlytas metodas ištirtas eksperimentiškai, apibendrinti jo rezultatai ir taip įrodyta, kad vartotojo naršymo laiką tinklalapyje galima išnaudoti sekančio žingsnio metu reikalingų duomenų talpinimui į podėlį ir taip sumažinti tinklalapio atsako laiką.

Raktažodžiai: podėlio sudarymas, apriori, serverio atsako laikas, optimizacija, vartotojų naršymo istorija.

SUMMARY

Prediction Based Solution for Web Data Caching

WebSites speed optimization are often used within cache, which contains the reusable resources that uses a large amount of users. This paper provides an overview of the existing cache types, their specificity and set itself the target of increasing online user system response speed by applying prior individual cache preset method. The proposed method appropriateness of experiments in which the analysis of the changes in the average site with the critical sections when the time is totally exempt from the cache and when there is a full cache. The study results revealed the need for cache, and its adaptation to the real situation, the proposed new partial cache compilation method. This method allows the user to predict the following pages you visit and the cache contains only them. The proposed method is tested experimentally, summarize the results and thus proved that the user browsing time website can be exploited during the next step necessary to store data in the cache and thus reduce site response time.

Keywords: precache, apriori, server response time, optimization, users log analysis.

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS

PHP - Skriptinimo kalba.

SQL - Standartizuota duomenų bazėse naudojama kalba, darbui su resursais.

KN - Kontroliniai duomenys nenaudojant išankstinio podėlio.

KP - Kontroliniai duomenys panaudojant pilnąjį podėlį.

TKN - Vidutinis serverio atsako laikas kontroliniuose duomenyse, be išankstinio podėlio.

TKP - Vidutinis serverio atsako laikas kontroliniuose duomenyse, su pilnuoju podėliu.

Axx/yy/z - Tyrimų pavadinimų formavimas. A – žymi apriori algoritmo panaudojimą; *xx – supp*. *reikšmę*; *yy – conf*. reikšmę; z – vėlinimo laikas *(s)*

Cin - Podėlio įvestis.

Cout - Podėlio išvestis.

Rin - Užklausos įvestis.

Rout - Užklausos išvestis, (užklausos rezultatai).

Min - Matricos įvestis.

Logout - Vartotojo istorijos išvestis.

PreCache - Išankstinis podėlio sudarymo procesas.

Cache - Podėlis.

Kritinių puslapių delsimas - Puslapių kuriems negalima paruošti podėlio remiantis įprastu podėlio sudarymo procesu, nuolat didėjantis atsako laikas.

Vėlinimas - Vartotojo puslapyje praleidžiamo laiko simuliavimas.

PAVEIKSLĖLIŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

TURINYS

ĮVADAS

Nuo 1990 m., kada šiuolaikinis internetas pradėjo skaičiuoti savo pirmąsias dienas, kuriant tinklapius buvo atsižvelgiama į jų užkrovimo greitį. Buvo pasitelkta įvairių metodikų ir technologinių sprendimų tinklalapių užkrovimo greičiui padidinti. Viena svarbiausių ir efektyviausių tinklalapio užkrovimo greičio didinimo technologijų – resursų podėlio sudarymas.

Podėlis gali būti panaudojamas beveik visur, kur vyksta duomenų transakcija ir galima iš anksto paruošti resursus jų vėlesniam panaudojimui. Šiuo metu serverio podėlis naudojamas DNS adresų, duomenų bazės užklausų rezultatams, paveikslėliams, tinklapio šablonams saugoti, tuo tarpu vartotojo, naršyklės podėlyje saugomi vartotojų duomenys, tinklapio failai ir kiti resursai. Nors podėlyje galima saugoti įvairius resursus, tačiau visada podėlio sudarymo procesas turi būti gerai apgalvotas, nes podėlis efektyvus tik tada, kada yra paruoštas ir tinkamai naudojamas. Įprastai podėlis yra paruošiamas tik tam pačiam pakartotinam procesui, kuris kviečiamas daug kartų ir kurio rezultatai nekinta kiekvienos užklausos metu. Tuo tarpu nepastovių resursų išankstinis talpinimas į podėlį dar nėra plačiai naudojamas, nors galėtų pasitarnauti išskirtinių paslaugų ar prioritetinių vartotojų sprendimų realizavimui.

Šio darbo tikslas – padidinti internetinės sistemos atsako vartotojui greitį, pritaikant išankstinio individualaus podėlio sudarymo metodą.

Šiam tikslui pasiekti yra iškelti šie **uždaviniai**:

1. Apžvelgti egzistuojančius podėlio panaudojimo principus ir technologijas.
2. Parinkti sprendimą, vartotojo sekančio žingsnio sistemoje tikimybei įvertinti.
3. Aprašyti siūlomo sprendimo realizavimo galimybes taikymui realioje aplinkoje.
4. Įvertinti siūlomo sprendimo pritaikymo galimybes.

Šio darbo metu gauti rezultatai leis padidinti internetinių sistemų užkrovimo laiką.

1 PODĖLIS IR JO TAIKYMAS INTERNETINĖSE SISTEMOSE

1.1 Podėlio panaudojimo principai internetinėse sistemose

Šiuolaikinėse internetinėse sistemose podėlio sudarymo architektūra dažniausiai susidaro iš šių komponentų (žr. Pav. 1) [5]:

Vartotojas – pagrindinis objektas, kuris reikalauja daugiausiai dėmesio. Jis siunčia užklausas serveriui, reikalaudamas puslapio ar kitokio resurso, ir tikisi jį gauti greitai.

Užklausa – vartotojo siunčiama užklausa, reikalaujamas resursas. Remiantis užklausa servisas nustato kokio resurso reikalauja klientas, ar šį resursą servisas turi podėlyje.

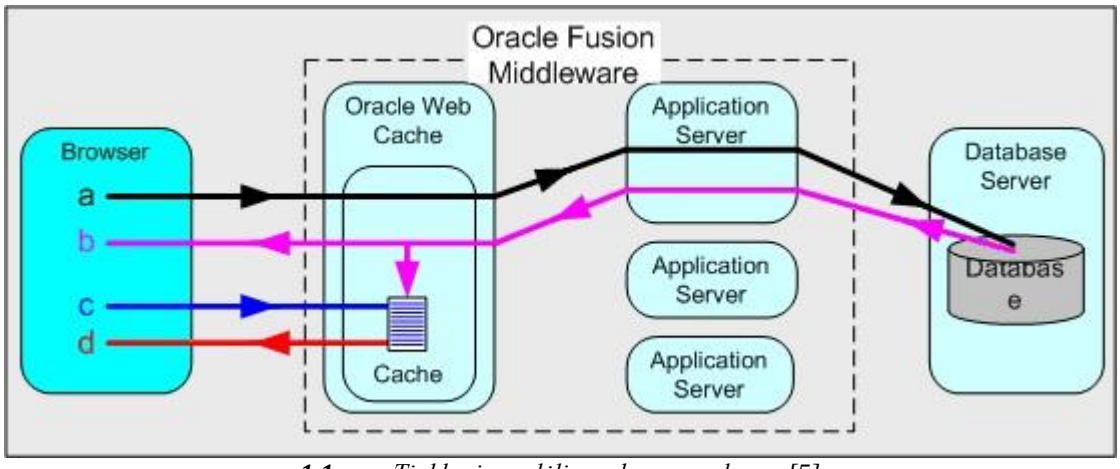
Servisas – komponentas, kuris apdoroja vartotojo užklausas. Jis atlieka daugiausia darbo šioje grandinėje. Jei podėlio apie kliento reikalaujamą resursą nėra, servisas darbo atlieka daugiau.

Resursas – objektas kurio klientas reikalauja siųsdamas užklausą. Gali būti podėlyje saugomas ir podėlyje nesaugomas resursas. Resurso pavyzdžiai internetinėse sistemose yra HTML kodas, paveikslėlis ar kitas dinaminis arba statinis resursas kurio klientas reikalauja.

Saugykla – komponentas, kuris gali atstoti arba yra duomenų bazė. Iš saugyklos servisas ima resursus, jei jų nėra podėlyje.

Podėlis – jei servisas turi duomenis imti iš saugyklos, tam, kad sekantį kartą kai vartotojas pateiks tokią pat užklausą servisui nebereikėtų kreiptis į saugyklą, resursai/duomenys/rezultatai yra talpinami čia [17]. Tai yra pagrindinis šiai grandinei optimizuoti skirtas komponentas.

Rezultatas – resursas grąžintas klientui. Resursas gali būti paimtas iš podėlio, saugyklos arba dabartinio proceso sugeneruotas.



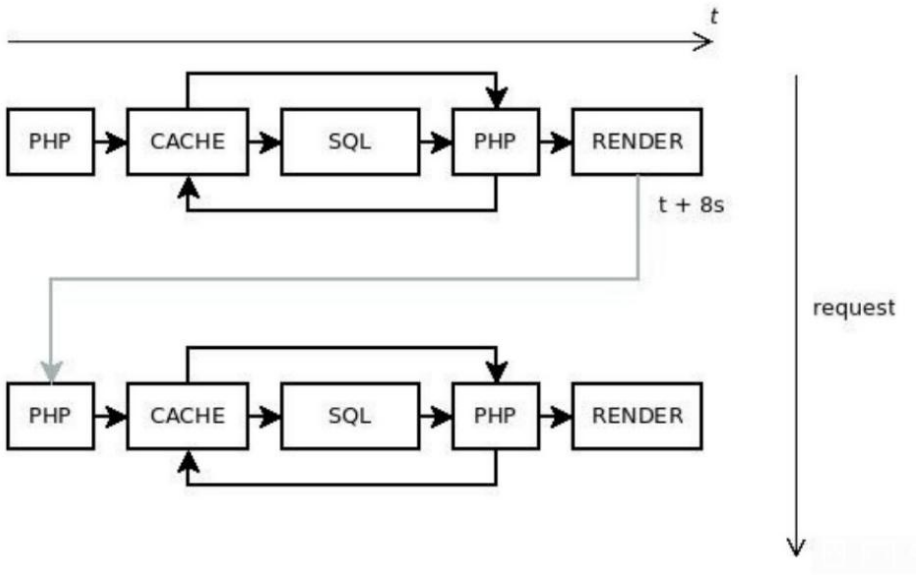
1.1 pav. *Tinklapio podėlio sudarymo schema [5]*

Tarkime turime tinklalapį, kurtą PHP programavimo kalba, kuriame klientui yra atvaizduojami duomenys iš duomenų bazės. Tokiu atveju serveryje naudojami 4 pagrindiniai procesai:

- *PHP* – *PHP* kalba parašytas procesas. Jo metu atliekami puslapyje reikalingi loginiai veiksmai.
- *CACHE* – esamos užklausos podėlis.

- *SQL* – procesas skirtas užklausos duomenų gavybai iš reliacinės duomenų bazės. Šiame procese atliekamos įvairios *SQL* užklausos reikalingos šiai užklausai įvykdyti.
- *RENDER* – puslapio atvaizdavimo klientui skirtas procesas. Šioje vietoje klientui yra grąžinamas *HTML* kodas.

Pagal analizuojamą pavyzdį serverio kviečiami procesai pavaizduoti 1.2 paveiksle. Kai vartotojas siunčia užklausą serveriui, reikalaudamas norimo puslapio ar resurso, serveryje pirmiausia yra atliekamas *PHP* procesas. Atliekami loginiai veiksmai, užklausos eigoje yra reikalingi su duomenų baze susiję veiksmai – duomenų gavimas iš duomenų bazės. Tam, kad nereikėtų su lyg kiekviena užklausa serveriui kreiptis į duomenų bazę, duomenų bazės rezultatai yra saugomi podėlyje [13]. Tad procesas pirmiausia patikrina ar reikalinga informacija jau nėra podėlyje, ir jei nėra tik tada kreipiamasi į duomenų bazę. Po to, kai rezultatai yra grąžinami iš duomenų bazės, jie yra talpinami podėlyje, sekančiam panaudojimui. Šie rezultatai yra grąžinami *PHP* procesui, ir yra naudojami sugeneruojant *HTML* kodą, ar paruošiant kitą resursą, kurio klientas reikalavo.

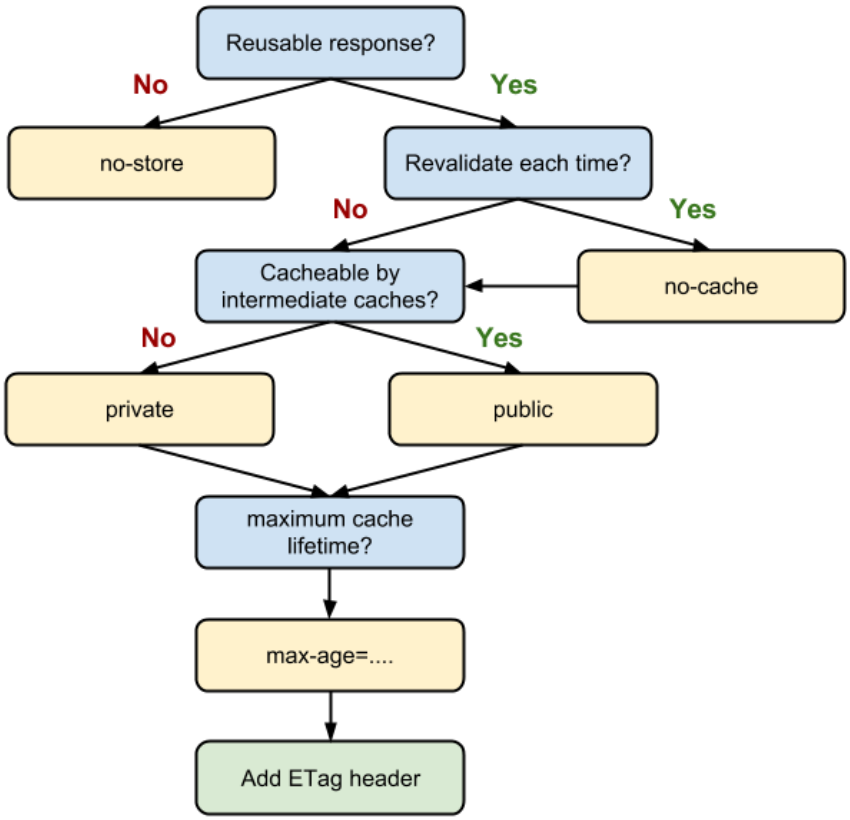


1.2 pav. *Įprastinė internetinio puslapio, naudojančio podėlį, schema*

Kaip pavaizduota schemoje (žr. 1.2 pav.) podėlio tikrinimas atliekamas kiekvienam vartotojo kreipimuisi (X ašis vaizduoja vienos užklausos apdorojimo laiką, o Y ašis atitinka laiko tarpą tarp užklausų, t. y. vartotojo užklausas į sistemą). Nors schemoje naudojamas užrašas *t+8s*, kuris reiškia, kad vartotojas vieno puslapio peržiūrai užtrunka vidutiniškai 8 sekundes, tačiau šis laiko tarpas gali kisti [16], priklausomai nuo sistemos ar jos vartotojo.

Literatūroje egzistuoja ne mažai patarimų, kada verta naudoti podėlį, o kada jo naudojimas yra nerekomenduojamas. Pateikta schema, kurią rekomenduoja Google Developers kūrėjai [6] (žr. 1.3 pav.). Jų teigimu svarbiausia įvertinti ar podėlyje žadamas saugoti resursas yra pakartotinai panaudojamas. Jei jo nebus galima pakartotinai panaudoti, vadinasi jo nereikia saugoti ir podėlyje. Tuo tarpu jei resursas gali būti panaudojamas pakartotinai, tada verta dar įvertinti ar kiekvieną kartą naudojant tą resursą nereikės jo patikrinti (užtikrinti, kad jame saugomi duomenys teisingi ir nepakitę). Jei podėlyje saugomas resursas turės būti tikrinamas, dažniausiai lemia tai, kad jo nerekomenduojama saugoti podėlyje, tačiau tai labai priklauso

nuo to kokio tipo tikrinimas bus vykdomas ir griežtos rekomendavimo nesaugoti podėlyje nėra. Šioje vietoje galutinį pasirinkimą turėtų lemti tai, kokio sudėtingumo yra podėlyje saugomo resurso tikrinimas, ar jam sugaištami resursai nėra artimi visiškai naujų duomenų gavimui, o ne ėmimui iš podėlio. Tuo tarpu jei podėlyje saugomų resursų nereikia papildomai tikrinti arba jei tikrinimas naudoja pakankamai mažai resursų, dar siūloma įvertinti kur bus realizuotas podėlis, t.y. ar jis bus viešas ar privatus. Ir atitinkamai bet kokiu atveju podėliui reikėtų įvertinti jo maksimalų galiojimo laiką.



1.3 pav. *Google Developers rekomenduojama podėlio reikalingumo ir valdymo tvarka[6]*

Nors Google Developers ir pateikia rekomendacijas, kaip nustatyti ar galima naudoti podėlį ir kokio tipo, bet net ir šiose rekomendacijose yra pakankamai daug neapibrėžtumo. Kiekvienu specifiniu atveju podėlio panaudojimas privalo būti gerai apgalvotas ir pasvertas, nes netinkamai jį panaudojus, galimas ne sistemos greičio padidėjimas, o kaip tik priešingas efektas [18].

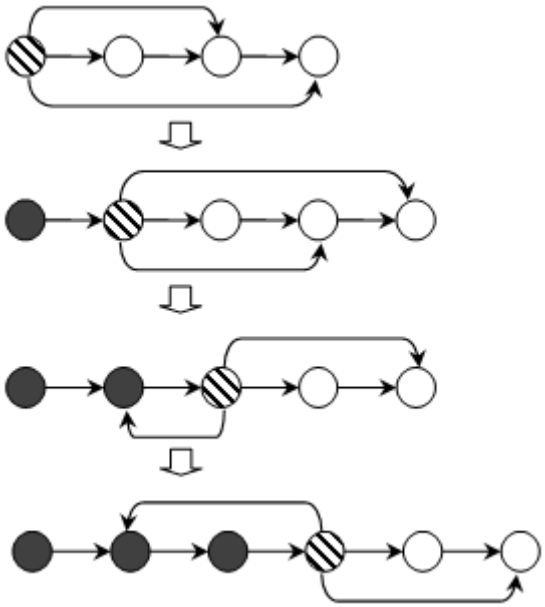
1.2 Išankstinio podėlio gylis

Išankstinis podėlis yra tas, kuris sudaromas remiantis tuo, ką turės matyti pats vartotojas savo sekančiuose žingsniuose, o ne kuo galės pasinaudoti kiti, kurie iškvies tą patį procesą, ar aplankys tą puslapį, kurį lanko dabartinis vartotojas.

1.2.1 0-nio lygio išankstinis podėlis

1.4 paveiksle pateikta iliustracija vaizduoja tradicinį podėlį, kur proceso rezultatai yra talpinami į podėlį tik tada, kada kviečiamas tas konkretus procesas. Brūkšniuotu fonu vaizduojami procesai, kuriuos šiuo metu kviečia vartotojas, baltai – procesai, kurių rezultatai nėra išsaugotas podėlyje, tamsiu fonu – procesai, kurių rezultatai jau išsaugoti podėlyje. Taip iš kairės į dešinę žiūrint matoma vartotojo kviestų ar galimų kviesti procesų seka (galimus perėjimus nuo vieno proceso į kitą vaizduoja rodyklės tarp procesų).

Tuo tarpu žiūrint iš viršaus į apačią galima stebėti kaip keičiasi procesų būsenos, t.y. kaip jos pereina iš vienos į kitą, kada vartotojas iškviečia atitinkamą procesą.



1.4 pav. *0-nio gylio išankstinio podėlio pavyzdys*

Toks išankstinis podėlis dar vadinamas *0*-nio gylio, nes neparuošia sekančiam vartotojo galimam procesui skirto podėlio, o tik saugo šiuo metu proceso gaunamus rezultatus podėlyje pakartotiniam tų pačių procesų keitimui (to pačio, ar kitų vartotojų). Tad 1.4 paveiksle vaizduojama, kad pirmame veiksme vartotojas apsilanko pirmajame puslapyje (procese). Jei šis puslapis podėlio neturėjo, jis yra paruošiamas sekančiam to paties puslapio užkrovimui (sudaromas podėlis tam procesui – apskritimas tampa tamsaus fono). Sekančiu žingsniu vartotojas patenka į antrąjį puslapį ir yra atliekami analogiški veiksmai kokie buvo vykdomi pirmajam puslapiui.

Realus *0*-nio lygio podėlio pavyzdys – sukurtas tinklalapis, kuriame patalpinti įvairūs straipsniai. Klientas apsilanko puslapyje, kuriame yra atvaizduojamas *A* straipsnis. Jei sistemos podėlyje nebuvo straipsnio *A*, jis yra paruošiamas rodymui ir išsaugomas podėlyje. Iš čia, klientas patenka į sekantį puslapį, kuriame yra atvaizduojamas straipsnis *B*. Suprantama, šiame puslapyje nebetiks straipsni *A* podėlis, todėl vykdomas *B* straipsniui sugeneruoti reikalingas procesas, o jo rezultatai saugomi podėlyje vėlesniam naudojimui.

Jei tinklalapis būtų tik dabar paviešinamas, jame apsilankius vartotojui ir kviečiant tinklalapyje vis kitus, vartotojo dar nekviestus procesus, visi jie būtų neparuošti podėlyje [15]. Todėl šis podėlis ir vadinamas *0*-nio gylio.

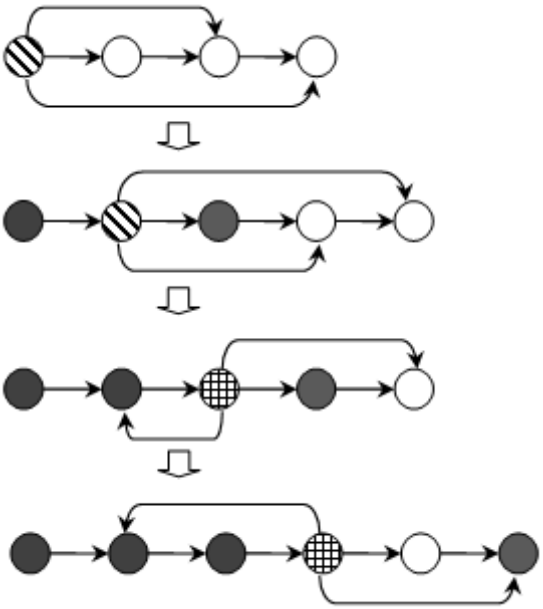
Apibendrinant galima pasakyti, kad *0*-tojo gylio podėlis pasižymi šiomis savybėmis:

- Podėlis gali būti naudojamas tik tam pačiam procesui (puslapiui).
- Podėlis tampa efektyvus tik tada, kai procesas, kuris sugeneravo šį podėlį yra atliekamas pakartotinai (puslapis yra pakartotinai užkraunamas).
- Podėlis yra sugeneruojamas tik tada, kai procesas būna vykdomas.

1.2.2 N-tojo lygio išankstinis podėlis

0-nio lygio podėlio atveju vykdomo proceso rezultatų patalpinimas į podėlį nereikalauja jokios sąsajos su kitais tinklalapio puslapiais ar vykdomais procesais. Tuo tarpu *N*-tojo gylio išankstinis podėlis paveldi *0*-nio gylio išankstinio podėlio galimybes, bet jas papildomai praplečia išsaugodamas ne tik tai kas buvo gauta dabartiniame procese, bet ir tai, ko prireiks kituose, dar vartotojo nekviestuose procesuose.

1.5 paveiksle galima matyti, kad vartotojui iškvietus pirmą procesą yra sudaromas podėlis pakartotiniam to proceso kvietimui ir trečiam procesui, kurį vartotojas tikėtina, kad kažkada ateityje iškvies. Atitinkamai esant antrame žingsnyje paruošto podėlio šiam procesui dar nėra, nes jis nebuvo paruoštas ankstesniuose procesuose, bet atlikus trečią žingsnį vartotojas jau atsiduria procese, kuris iš anksto buvo paruoštas (podėlis jam sukurtas 1 žingsnio metu), tad jis vaizduojamas spalvinant foną langeliais. Taip anksčiau kviesti procesai gali paruošti podėlį, kuris tiks *N* kitų procesų, kurie tikėtina, kad bus kviečiami ateityje.



1.5 pav. *N-tojo gylio išankstinio podėlio pavyzdys*

Realus tokio kliento pavyzdys – paveikslėlių dydžio keitimas ir saugojimas podėlyje. Dažnai vartotojas į serverį įkelia labai didelių matmenų paveikslėlį, nors realiai tinklalapyje vaizduojamos to paveikslėlio miniatiūros. Pagal situaciją gali būti vieno ar kelių skirtingų dydžių miniatiūrų, o jei tas paveikslėlis naudojamas dažnai, tai dalis ar visos to paveikslėlio miniatiūros talpinamos podėlyje. Tad jei sistemoje nėra atitinkamų išmatavimų paveikslėlio miniatiūros ar ji nepatalpinta podėlyje, to proceso metu reikiama paveikslėlio miniatiūra yra sukuriama ir išsaugoma podėlyje. Nors tokių matmenų miniatiūra buvo sukurta tame konkrečiame procese, puslapyje, bet tikėtina, kad tokių išmatavimų paveikslėlis gali būti panaudojamas ir kituose procesuose. Tad automatiškai šio proceso metu bus parengiamas podėlis, tinkamas ir kituose procesuose.

Apibendrinant šį N-tojo gylio išankstinį podėlį, išskiriamos tokios jo savybės:

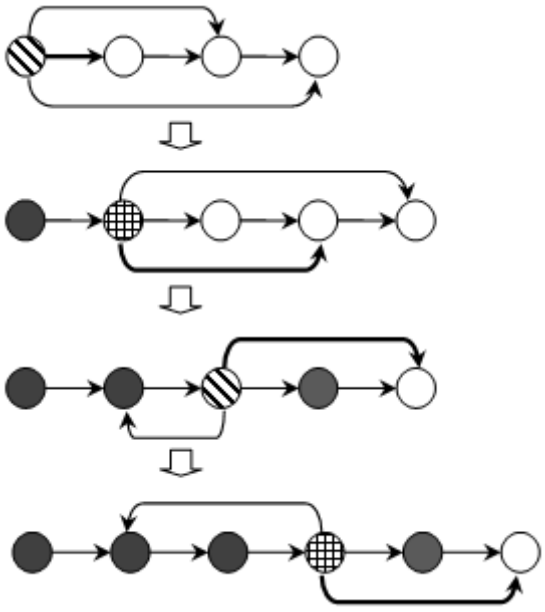
- Podėlis gali būti naudojamas to pačio proceso pakartotiniam iškvietimui bei kituose, dar vartotojo nekviestuose procesuose.

- Podėlis gali būti dalinai ar pilnai naudojamas dar vartotojo nekviestiems procesams.
- Podėlis yra sugeneruojamas tik tada, kai procesas būna vykdomas, bet nesiremia vartotojo veiksmų prognozavimu.
- Toks podėlis nereikalauja papildomos logikos, bet orientuotas į proceso resursų skaidymą, jų panaudojamumui padidinti.

1.2.3 +1 gylio išankstinis podėlis

Nei *0*-nio, nei *N*-tojo gylio išankstinis podėlis nebando nuspėti sekančių vartotojo veiksmų sistemoje. Tuo tarpu *+1* gylio podėlio principas yra nuspėti ko gali prireikti vartotojui sekančiame žingsnyje ir dar prieš vartotojui iškviečiant tą procesą, jį iškviesti iš anksto to proceso podėlio sudarymui.

1.6 paveikslėlyje vaizduojamame +1 gylio podėlio pavyzdyje matyti, kad vartotojui apsilankius pirmame puslapyje (procese) yra paruošiamas antro proceso podėlis (vaizduojama pastorinta linija, kuris kelias nuspėjamas kaip sekantis vartotojo žingsnis), tad jau sekančiame žingsnyje vartotojui yra rodomi podėlio išsaugoti rezultatai. Kadangi dažnai vartotojas gali rinktis kelis tolesnio darbo sistemoje scenarijus, tai netinkamai nuspėjus sekančius vartotojo žingsnius gali būti sudarytas ne to proceso, kurį iš tikro kvietė vartotojas podėlis. Tokiu atveju kai netinkamai nuspėjamas vartotojo sekantis žingsnis, gaunama, jog procesas buvo kviestas, naudosi sistemos resursai to podėlio sudarymui, tačiau rezultatai sekančiame žingsnyje, ar visiškai bus nepanaudoti.



1.6 pav. *+1 gylio išankstinio podėlio pavyzdys*

+1 gylio išankstinis podėlis dažniausiai paveldi ir *0*-nio bei *N*-tojo gylio išankstinio podėlio savybes, o tipinis pavyzdys galėtų būti bet kuris aiškaus nuoseklumo procesas. Pavyzdžiui jei vartotojas tinklalapyje pasirinko funkciją sukurti naują komentarą, tai sekančiame žingsnyje tikėtina, kad bus išsaugomas sukurtas komentaras ir parodomi kiti komentuojamo objekto komentarai. Tad kol vartotojas rašo komentaro tekstą, galima iš karto paruošti podėlį visų kitų komentarų rodymui.

Šio tipo podėlis gali nuspėti ne tik vieną, bet ir daugiau galimų vartotojo sekančių žingsnių, tad tais atvejais, kai esamame procese bandoma nuspėti keletą galimų vartotojo žingsnių ir visus juos parengti paruošiant kiekvienam podėlį, vadinamas *+n* gylio podėliu, kur *n* reiškia kiek alternatyvių kelių yra nuspėjama į priekį.

Apibendrinant galima teigti, kad *+1* ir *+n* gylio podėlis pasižymi šiomis savybėmis:

- Podėlis yra efektyvus vartotojo požiūriu, nes iš anksto yra sudaromas tam tikrų dar nelankytų procesų podėlis.
- Podėlis gali būti sugeneruojamas ir nei karto pačiam vartotojui neiškvietus būtent to proceso.
- Podėlio sudarymui būtina papildoma logika ir resursai, sekančio vartotojo žingsnio nuspėjimui.
- Podėlis gali padidinti sistemos apkrovą ir neefektyviai naudoti jos resursus, jei bus netinkamai nuspėjami sekantys vartotojo veiksmai sistemoje.

1.3 Kritinė sekcija

Dažniausiai podėlio efektyvumas pasiekiamas tuo, kad tai, kas išsaugoma podėlyje, vėliau gali būti naudojama pakartotinai daug kartų ir kiekvieną kartą nebereikia atlikti daug resursų reikalaujančių veiksmų, o tik iš podėlio gauti vartotojui turimą rodyti to proceso rezultatą [14]. Jei tikrinimas, ar reikiamas resursas yra išsaugotas podėlyje ir jo panaudojimas užtrunka ilgiau, nei pačio proceso įvykdymas trunka daugiau ar net panašiai, tai tokia situacija dažniausiai yra netinkama podėlio naudojimui.

Kaip minėta 1.1 skyriuje, podėlio sudarymui netinka ir tokie procesai, kurie yra nuolat kintantys. Pavyzdžiui kiekvieną kartą, kada vartotojas atveria tam tikrą puslapį, jam reikia atvaizduoti kiek vartotojų apsilankė prieš tai buvusiame puslapyje. Tokiu atveju šis skaičius nuolat kis vos tik atvėrus tą puslapį, todėl retai kada juos bus galima parodyti daugiau nei vieną kartą. Tačiau puslapį aplankiusių vartotojų skaičiaus gavimui iš duomenų bazės reikia pakankamai daug resursų, kas lėtina viso tinklalapio užkrovimo greitį. Iš to sektų, jog šių duomenų saugoti podėlyje neverta dėl riboto panaudojimo, bet tinklalapio užkrovimo laiko atžvilgiu tai būtų naudinga. Tad procesai, kurių rezultatai yra kintantys, o vykdymas reiklus resursų naudojimui yra vadinamos kritinėmis sekcijomis.

Kritinės sekcijos pasižymi savybėmis, kurios iš vienos pusės neleidžia naudoti podėlio, o iš kitos pusės, panaudojus podėlį padėtų padidinti tinklalapio užkrovimo greitį. Todėl kiekvieną kartą, kuomet yra sudaroma tinklalapio podėlio valdymo schema, kritinių sekcijų analizei skiriamas ypatingas dėmesys, nes būtina įvertinti ar atitinkamoje situacijoje svarbu sistemos greitis ar sistemos apkrova.

1.4 Podėlio taikymo internetiniuose tinklalapiuose apibendrinimas

Susipažinus su internetinių puslapių podėlio sudarymo koncepcija ir podėlio gylio tipais pastebėta, kad esami sprendimai nepilnai išnaudoja galimybes vartotojams suteikti itin greitai veikiantį tinklalapį, panaudojant išankstinio podėlio sudarymo galimybes. Tokia išvada formuojama remiantis tokiais faktais:

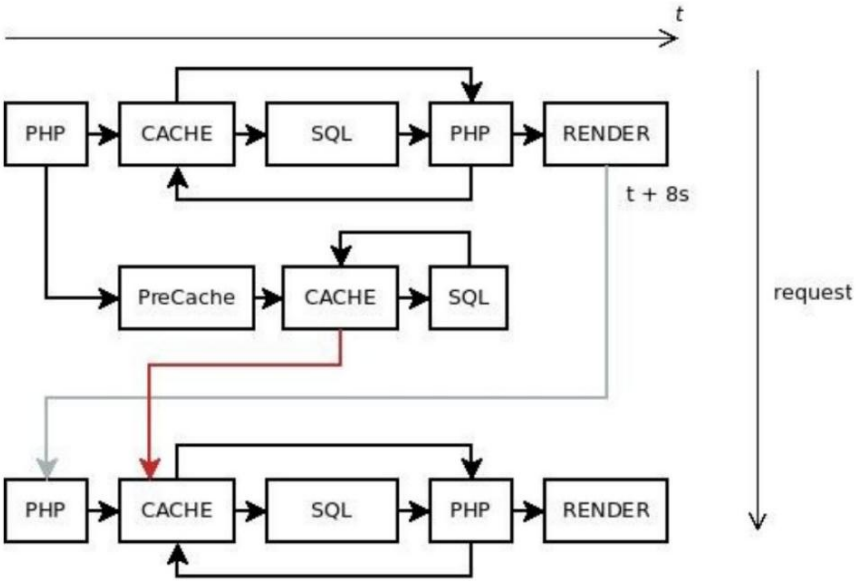
- Standartinio podėlio atveju, podėlis netenka prasmės, jei duomenys yra paruošiami dinamiškai nuo dabar apdorojamų užklausų, t.y. tame procese yra kritinių sekcijų.

- Standartinis podėlis yra tinkamas tik tam pačiam, pakartotinai naudojamam procesui, t. y. podėlis bus efektyvus tik pakartotinai apdorojant tą pačią užklausą, o visais kitas atvejais bus vykdomas procesas, o ne naudojamas jo podėlis.
- Vartotojo tinklapyje praleidžiamas laikas (vidutiniškai 8 sekundes, o dažnai ir daugiau) yra neišnaudojamas. Tuo metu, kada vartotojas žiūri į sugeneruotą tinklalapio vaizdą ar jame atlieka numatytą užduotį (rašo, skaito ir pan.), būtų galima atlikti papildomus veiksmus, tokius kaip sekančio vartotojo žingsnio nuspėjimą ir jo parengimą podėliui. Šį vartotojo tinklapyje praleistą laiką toliau tyrimuose simuliuosime atlikdami **vėlinimą**.

2 IŠANKSTINIS DEDIKUOTASIS PODĖLIS IR JO TIKSLINGUMO ĮVERTINIMAS

2.1 Išankstinio dedikuotojo podėlio koncepcija

Atsižvelgiant į 1 skyriaus išvadose paminėtus faktus, kyla išankstinio dedikuotojo podėlio sudarymo koncepcija. Jos principas – tuo metu kada vartotojas peržiūri dabartinį puslapį, inicijuojamas procesas, kuris lygiagrečiai sukuria podėlį kritinėms sekcijoms. Lyginant su 1.2 paveiksle vaizduojama įprastine internetinio puslapio, naudojančio podėlį, schema, dedikuotojo podėlio sudarymo schema papildomai naudoja *PreCache* procesą, kuris skirtas kritinės sekcijos podėlio sudarymui tuo metu, kol vartotojas peržiūri dabartinį tinklalapio puslapį (žr. Pav. 7). Atitinkamai šis *PreCache* procesas formuoja atitinkamą podėlį, pagal jam turimą vykdyti *SQL* užklausą.



2.1 pav. *Dedikuotojo podėlio veikimo schema*

Tokio dedikuotojo išankstinio podėlio schema leistų išnaudoti tą laiko tarpą, kada vartotojas peržiūri jam rodomus duomenis. Jei sekantis vartotojo lankomas puslapis bus nuspėjamas tiksliai, tai vartotojui užkraunamo puslapio greitis turėtų pagreitėti, tuo tarpu serverio apkrova neturėtų padidėti žymiai. Tačiau ar tikrai šis metodas yra tikslingas aiškių rezultatų nėra, todėl reikalauja papildomo pagrindimo.

2.2 Podėlio panaudojimo kritinėms sekcijoms tyrimas

Šio tyrimo metu siekiama įvertinti ar tikslinga naudoti išankstinį podėlį kritinėms sekcijoms. Tyrimo eigoje sukuriami kontroliniai duomenys, kurie naudojami siūlomam podėlio sudarymo modelio efektyvumui įvertinti. Kontroliniai duomenys sugeneruojami dviem būdais: kai puslapyje yra įjungtas 0-nio gylio podėlio sudarymo procesas (vėliau vadinami KN tyrimu) ir kai puslapyje yra įjungtas pilnas podėlio sudarymas (vėliau vadinami KP tyrimu). Pilno podėlio atveju į podėlį talpinami visi sekančiame žingsnyje potencialiai galimi aplankyti tinklalapiai.

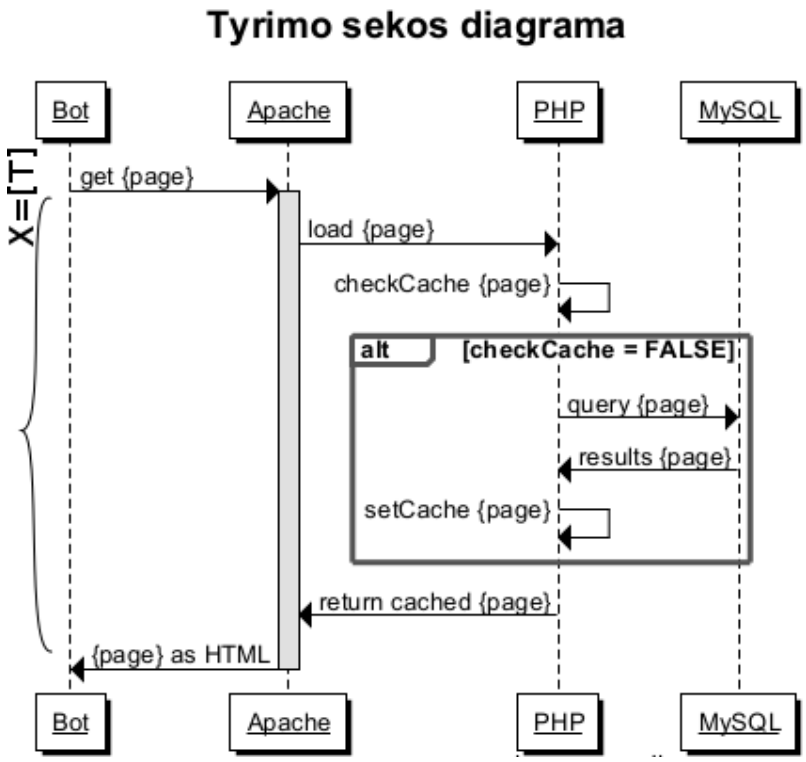
2.2.1 Tyrimo metu stebimos savybės ir jų fiksavimo įrankiai

Kiekvieno tyrimo metu paleidžiamas robotas (automatizuotas programinis kodas), kuris simuliuoja vartotojo naršymą. Robotas gauna informaciją [1] aprašytą 2.2 paveiksle. Tyrimo rezultatams apskaičiuojamas laikas, kuris buvo sugaištas serveryje puslapio padorojimui; laiką *x* nurodytą 2.3 paveiksle,

t. y. matuojamas laikas nuo pirmo bito išsiuntimo iki pirmo bito gavimo pagal 2.4 paveiksle vaizduojamą puslapio apdorojimo laiko nustatymų procesą.

```
1	"time_namelookup":"%{time_namelookup}",¥n
2	"time_connect":"%{time_connect}",¥n
3	"time_appconnect":"%{time_appconnect}",¥n
4	"time_pretransfer":"%{time_pretransfer}",¥n
5	"time_redirect":"%{time_redirect}",¥n
6	"time_starttransfer":"%{time_starttransfer}",¥n
7	"time_total":"%{time_total}"¥n
```

2.2 pav. *Naršymo roboto grąžinami duomenys*



2.3 pav. *Tyrimo sekos diagrama*

```
$process_time = (float)$time_starttransfer - (float)$time_connect;
```

2.4 pav. *Puslapio apdorojimo laiko nustatymas*

Taigi tyrimo eigoje surenkami du parametrai: užklausos numeris ir puslapio apdorojimo laikas. Užklausos numeris yra reikalingas tam, kad tyrime užklausas galėtume tirti nuosekliai. Tyrimo pabaigoje duomenys suvedami į tyrimo rezultatų lentelę (žr. 1 lentelę) kurioje atliekami skaičiavimai ir nurodymai:

- Tyrimas – nurodytas tyrimo žymuo.
- Papildomi parametrai – tyrime nurodyti papildomi parametrai. Apriori algoritme *Supp.*, *Conf.*, ir *len*. reikšmės. Roboto naršymo vėlinimas prieš atliekant sekančią užklausą; – *delay (s).*
- *Min; Mean; Max;* – statistiniai rodmenys apibūdinantys tinklapio krovimo laiką: *min* – mažiausia reikšmė, *mean* – aritmetinis vidurkis, *max* – didžiausia reikšmė. Matuojama *ms*.
- ΔT_{KP}; ΔT_{KN}; – statistinis tyrimo aritmetinio vidurkio skirtumas nuo *KP* ir *KN* tyrimų. Teigiamas skaičius rodo sutrumpėjusį puslapio krovimo laiką, neigiamas – pailgėjusį puslapio krovimo laiką.

1 lentelė. Pavyzdiniai tyrimo duomenys

Tyrimas	***Papildomi parametrai***	***Min;Mean; Max;***	***ΔT_{KP}; ΔT_{KN};***
Tyrimo pavadinimas	Supp. = 0.5	Min.: 0.0221 ms; Mean: 0.1111 ms; Max.: 0.2200 ms;	ΔT_{KP} = -0.0005 ms; ΔT_{KN} = 0.0000 ms;

2.2.2 Tyrime naudojami kontroliniai duomenys ir jų generavimas

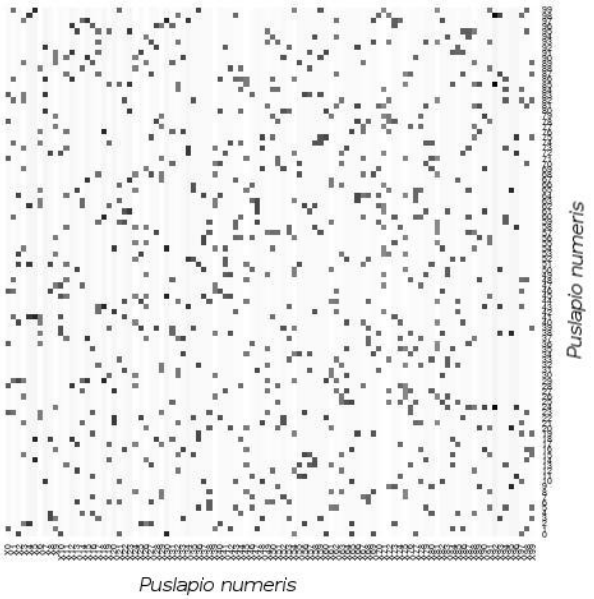
Tyrimui atlikti yra reikalingi kontroliniai duomenys. Kontroliniais duomenimis laikysime duomenis, surinktus naršant puslapį, kuriame yra tik 0-nio gylio podėlis, turintis kritinių vietų. Tyrimui atlikti sukurti du įrankiai – puslapių generatorius ir robotas, kuris naršo po sugeneruotą puslapį ir renka informaciją per kiek laiko buvo pateiktas puslapis.

Puslapių generatorius – tai *PHP* kalba parašytas puslapių generatorius, kuris pagal duotus parametrus sugeneruoja kryptinį grafą – puslapių medį. Generatorius konfigūruojamas trimis parametrais:

- Puslapių skaičius. Kiek tinklapis turės puslapių – kiek grafas turės mazgų.
- Jungčių skaičius $N \in [min;max]$; Atsitiktinis natūralusis skaičius *min, max* rėžyje. Nurodo kiek mazgas turės jungčių su kitais mazgais.
- Kritinių vietų skaičius, nurodoma procentais. Kiek mazgų *0*-nio gylio podėlis bus neveiksmingas.

2.2.3 Kontrolinių duomenų apibendrinimas

Kontroliniams duomenims rinkti buvo sugeneruotas tinklalapis su 100 puslapių, kur kiekvienas puslapis turi nuo 10 iki 20 nuorodų į kitus puslapius (*N(10; 20)*) ir *30*% iš tų puslapių turi kritinių vietų. Šie parametrai yra parinkti atsitiktinai, tačiau tai tyrime matuojamam krovimo laiko įvertinimui įtakos neturės, nes tyrimuose bus naudojamos tos pačios sąlygos. Kiekvieno tyrimo metu robotas atliks *100* skirtingų sesijų po *10* atverčiamų puslapių, tad vieno tyrimo metu bus atliekama po *1000* užklausų.



2.5 pav. *Puslapio jungčių matrica*

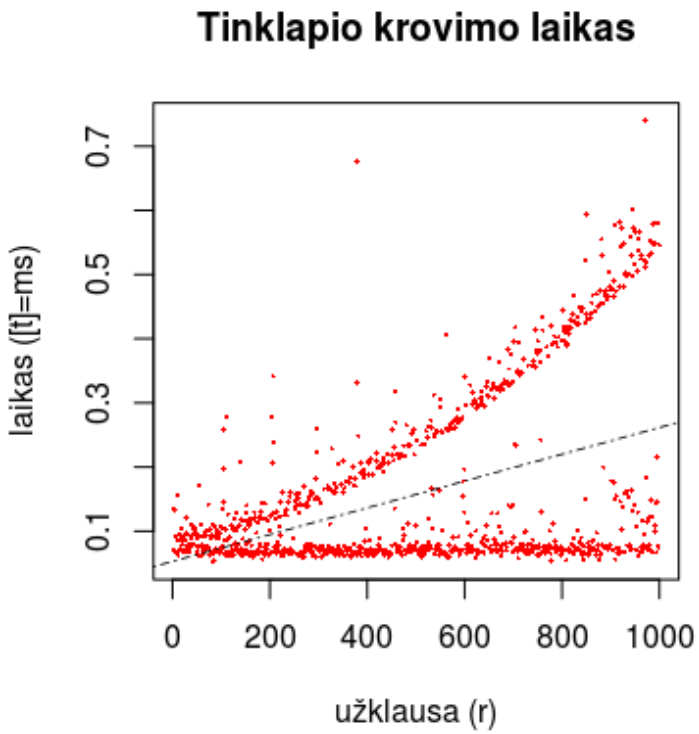
Puslapių jungčių matrica (žr. 2.5 pav.) apibūdina kaip yra sudarytas puslapių grafas: *x* ir *y* ašyse yra nurodomas puslapio numeris, ir jei ašių susikirtimo vietoje yra taškas, vadinasi šie puslapiai yra sujungti tarpusavyje; iš *y* ašyje nurodyto puslapio galima patekti į *x* ašyje nurodyta puslapį. Būtina atkreipti dėmesį, kad puslapių grafas yra vienkryptis, ir nebūtinai puslapiai yra sujungti abipusiškai, t.y. iš jei iš pirmo puslapio galima patekti į trečia puslapį, tai nereiškia, kad iš trečio puslapio bus galima patekti į pirmą puslapį.

2.3 Podėlio panaudojimo kritinėms sekcijoms tyrimo rezultatai

Kaip jau buvo minėta, kontroliniams duomenims sudaryti yra atliekami du tyrimai: *KN* ir *KP*. Apžvelkime juos abu.

2.3.1 KN tyrimo rezultatai

Atlikę tyrimą su *0*-nio gylio podėlio sudarymo metodu, iš grafiko (žr. 2.6 pav.) matome, kad puslapio užkrovimo laiko tendencija nuolat didėja. Tai įvyksta dėl to, jog klientas (robotas) papuolęs į tinklapį kuriame yra kritinė vieta, su lyg kiekviena užklausa pasunkina sekančio puslapio su kritine vieta duomenų apdorojimą, reikalingą puslapio atvaizdavimui. Šią nuolat didėjančio puslapio užkrovimo laiko priežastį tolimesniuose tyrimuose vadinsime ***kritinių vietų delsimu***.



2.6 pav. *Tinklapio krovimo laikas nenaudojant podėlio (KN tyrimas)*

Taip pat 2.6 paveiksle galime aiškiai įžiūrėti dvi užklausų grupes, kurios atitinką realią situaciją, jog vieni puslapiai neturi kritinių vietų, o kiti turi kritinių vietų, kurios didina tinklalapio atsako laiką.

2 lentelėje pateikti pagrindiniai šio eksperimento metu nustatyti *0*-nio gylio podėlio taikymo parametrai. Iš jų matyti, kad kartais pasitaiko pakankamai greitų tinklalapio užkrovimo atvejų, nes minimali puslapio užkrovimo trukmė yra 0.052 ms. Tai siejama su tuo, kad robotas aplankė puslapį, be kritinės sekcijos. Maksimali puslapio užkrovimo reikšmė siekia beveik sekundę, o apibendrinus visų užklausų apdorojimo laiką, vidutinė puslapio užkrovimo reikšmė yra 0.1574 ms.

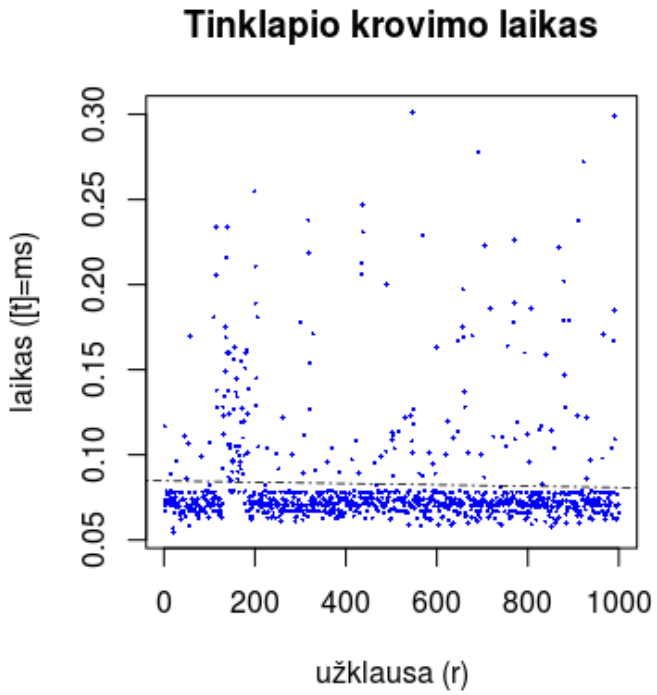
2 lentelė. KN tyrimo rezultatų lentelė

Tyrimas	***Papildomi parametrai***	***Min;Mean; Max;***	***ΔT_{KP}; ΔT_{KN};***
Kontroliniai be podėlio (KN tyrimas)	delay = 0s	Min.: 0.0520 ms; Mean: 0.1574 ms; Max.: 0.7400 ms;	ΔT_{KP} = -0.0746 ms; ΔT_{KN} = 0.0000 ms;

2.3.2 KP tyrimo rezultatai

KP tyrimas yra ypatingas tuo, kad tyrimo rezultatai bus artimi ***optimaliems***; t. y. pasitelkdami kitus algoritmus neturime gauti žymiai geresnio užkrovimo laiko už *KP* vidurkį. Tą teigti galime vadovaudamiesi pilnojo podėlio sudarymo metodu – su lyg kiekviena užklausa yra paruošiamas podėlis visiems puslapiams į kuriuos galime patekti sekančiu žingsniu.

Iš 2.7 paveiksle pateiktų duomenų matome, kad pilno podėlio atveju visų užklausų atsako laikas pasiskirstęs pakankamai tolygiai. Čia nepastebimas kelių užklausų tipų išsiskyrimas, todėl galima teigti, kad išankstinio podėlio taikymas kritinėms sekcijoms padeda išvengti dėl kritinių vietų galimo papildomo vėlinimo atsiradimo.



2.7 pav. *Tinklapio krovimo laikas naudojant išankstinį podėlį (KP tyrimas)*

3 lentelėje pateikti apibendrinti tyrimo duomenys, kurie atskleidžia, kad minimali tinklalapio atsakymo reikšmė yra panaši, kaip ir KN testo atveju, tačiau maksimalus tinklalapio atsako laikas, lyginant su NK tyrimu yra beveik perpus mažesnis, taip pat ir vidutinė puslapių atsako laiko reikšmė.

3 lentelė. KP tyrimo rezultatų lentelė

Tyrimas	***Papildomi parametrai***	***Min;Mean; Max;***	***ΔT_{KP}; ΔT_{KN};***
Kontroliniai (su podėliu) (KP)	delay = 0s	Min.: 0.0550 ms; Mean: 0.0828 ms; Max.: 0.3010 ms;	ΔT_{KP} = 0.0000 ms; ΔT_{KN} = 0.0746 ms;

2.4 Išankstinio podėlio taikymo tikslingumo apibendrinimas

Lygindami *KP* ir *KN* tyrimus galime pastebėti, kad *KP* vidutinis tinklapio užkrovimo laikas palyginus su *KN* vidutiniu tinklapio užkrovimo laiku yra spartesnis 0.0746 ms ($\Delta T_{KN} = 0.0746$ ms).

Lyginant KP ir KN tyrimų rezultatus pastebima, kad išankstinio pilno podėlio taikymas leidžia eliminuoti skirtumą tarp tinklalapių su kritinėmis vietomis ir be jų, nes visų tinklapių atsako laikas nebegali būti akivaizdžiai skirstomas į du klasterius pagal tų atsako laiką.

Taip pat atsižvelgdami į sąlygas nurodančias, kad *KP* rezultatai yra artimi optimaliems, galime teigti, kad yra tikslinga ieškoti tarpinio sprendimo. Taip pat įvedėme tolimesniems tyrimams reikalingas konstantas – kontrolinių duomenų puslapių krovimo laiko aritmetinius vidurkius:

$$T_{KP} = 0.0828; T_{KN} = 0.1574;$$

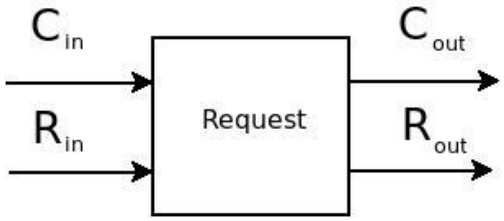
3 IŠANKSTINIO DEDIKUOTOJO PODĖLIO SUDARYMO ARCHITEKTŪRA IR KOMPONENTAI

Išankstinio dedikuotojo podėlio sudarymui vienos vartotojo užklausos metu vykdomi keli priedėlio sudarymo procesai. Visą bendrą išankstinio dedikuotojo podėlio sudarymo architektūrą sudaro keli komponentai:

- *Request* – pagrindinio, 0-nio lygio podėlio sudarymui.
- *Algo* – vartotojo naršymo sekančio žingsnio tikimybiniam įvertinimui.
- P_{ex} – sekančio proceso podėlio sudarymas.

3.1 Komponentas *Request*

Šis komponentas atvaizduoja kliento užklausos procedūrą. Komponentui reikalingi du įvadai, – kliento užklausa (R_{in}) ir šios užklausos podėlis (C_{in}) kuris yra nebūtinas. Procedūra pabaigoje grąžina dvi išvestis: šios užklausos podėlis (C_{out}) ir užklausos rezultatas (R_{out}), kuris yra grąžinamas klientui kaip užklausos atsakymas – *HTML* kodas, paveikslėlis ar panašus resursas, kurį turi grąžinti ši užklausa. Užklausos podėlis (C_{out}) yra resursas, kuris gali būti naudojamas sekančiai tokiai pačiai užklausai (*0*-nio lygio podėlis). Apibendrinus, šis komponentas vaizduojamas 3.1 paveiksle ir atitinka pirmosios užklausos *PHP → RENDER* grandinę ir yra naudojamas didžiąją dalį atvejų.



3.1 pav. *Komponentas Request*

3.2 Komponentas *Algo*

Šis komponentas simbolizuoja vartotojų naršymo istorijos analizės procedūrą. Šiam komponentui reikalingos dvi įvestys – vartotojų naršymo istorija (Log_{in}) ir pasirinktas algoritmas (A_{in}). Procedūra pabaigoje grąžina sudarytą matricą, kurioje nurodyta, kuriai užklausai esant, kokiam puslapiui paruošti podėlį.

Procesas atlieka reikiamą logiką *+1* gylio podėlio sudarymui ir nėra vykdomas kiekvieną kartą vartotojui kreipiantis į sistemą. Komponentas yra įvykdomas vieną kartą norint sudaryti matricą reikalingą komponentui P_{ex}. Sudaryta matrica bus naudojama visam likusiam P_{ex} komponento darbui iki kol bus sugeneruojama nauja matrica (žr. pav. 17).

Šiame procese taikomos asociacijų taisykles – a*priori* algoritmas, kuris padeda nuspėti sekančią vartotojo užklausą. Asociacijų taisyklių pagalba yra surandamos tikėtiniausios vartotojo lankomų puslapių sekos, bet ne tiek smarkiai gilinamasi į pačio tinklalapio puslapių tarpusavio nuorodas ir galimybes vartotojui pereiti iš vieno tinklalapio į kitą [2].

Apriori algoritmas yra konfigūruojamas šiais pagrindiniais parametrais:

- Palaikymas (*supp.*) – nurodoma procentais kiek dažnai pasikartojantys elementai turi būti duomenų masyve. Šiuo atveju šis parametras įtakos vartotojų specifiškumą, t. y. jei šio parametro reikšmė yra artima 100%, vadinasi mes ieškome bendro pobūdžio, neklasifikuotų vartotojų veiksmų ir tiesiog atrenkame pačius populiariausius tinklalapius, kuriuos tikėtina, kad šis bendro pobūdžio vartotojas aplankys. Tuo tarpu jei supp. reikšmė yra maža, mes analizuojame daugiau galimų vartotojo aplankyti kelių ir bandome identifikuoti jam labiausiai specifinį, taip tarsi labiau sukonkretindami vartotojo tipą. Tačiau mažos *supp*. parametro reikšmės naudojimas reikalauja, kad būtų analizuojamas didesnis duomenų kiekis, būtų gilinamasi į ilgesnes vartotojo veiksmų sistemoje grandines. Tai reikalauja daugiau sistemos resursų, o analizei naudojamuose duomenyse vartotojai sesijos metu turi vykdyti taip pat ilgas veiksmų grandines, nes tik kelių perėjimų iš vienos puslapio į kitą duomenų nepakaktų.

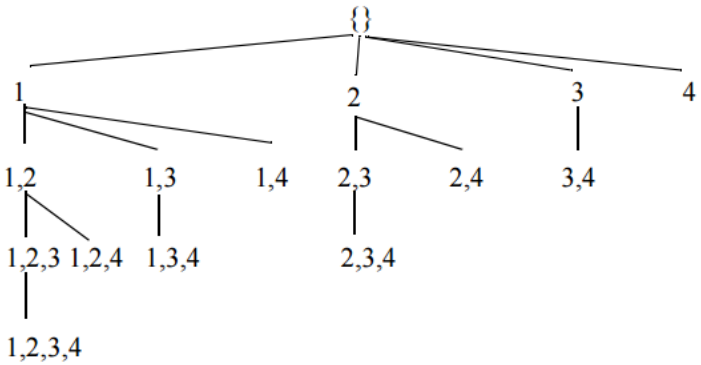
- Pasikliovimas (*conf.*) – nurodoma procentais kaip dažnai šis įrašas yra pasikartojantis rezultatuose. Šiuo atveju šis parametras įtakos kiek tiksliai mes norime nuspėti sekantį vartotojo žingsnį. Analizuojant vartotojų veiksmų sekas, jei šis parametras yra didesnis nei analizuojamos veiksmų sekos koeficientas, ta veiksmų seka yra nebeanalizuojama ir išmetama kaip nepakankamai dažna, nepakankamai tiksli.

- Ilgis (*len.*) – nurodoma kokio ilgio elementų grandinė turi būti tiriama. Šis parametras įvertina kokio ilgio vartotojų veiksmų sekas reikia analizuoti, tad apribojama kiek giliai reikėtų analizuoti vartotojo elgsenos gylį nuo atitinkamos pradžios taško. Pasirinkus tik vieno ar dviejų puslapių gylį algoritmo rezultatai gali būti pakankamai netikslūs, tuo tarpu pasirinkus analizuoti labai ilgas veiksmų sekas, pati sistema bus labiau apkraunama.

Apriori algoritmu remiantis vartotojų elgsenos duomenimis išskiriami visi vartotojų lankyti tinklalapiai. Tada kiekvienam iš tų puslapių yra įvertinama lankymo tikimybė. Ji kiekvienam puslapiui proporcionaliai apskaičiuojama pagal vartotojų vienos sesijos metu lankymosi tame tinklalapyje kiekį, jo santyki su bendru sesijų skaičiumi. Jei gauta tikimybė yra didesnė nei parametras *supp*. Tada ši seka analizuojama toliau analogišku principu, kada yra analizuojami visi kiti galimi sekantys žingsniai ir randama kiek tokių sekų yra analizuojamoje vartotojų naršymo istorijoje. Taip radus tinkamas pagal parametrus sekas paieška vykdoma paieškos į plotį principu tol, kol bus pasiektas naršymo sekos ilgis *len*.

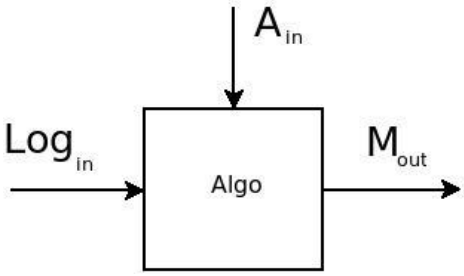
Aiškesniam *Apriori* algoritmo suvokimui pateiktas pavyzdys (žr. 3.2 pav.). Šiame pavyzdyje yra tik keturi puslapiai, kuriuos vartotojai lanko, kurie yra aptinkami vartotojų sesijos metu. Kiekvieną puslapį identifikuojant pagal skaičius 1, 2, 3 ir 4 galime stebėti kaip vyksta *Apriori* algoritmo veikimas: identifikavus visus keturis lankomus tinklalapius, įvertinamas jų pasitaikymo dažnumas ir toliau analizuojami tik tie keliai, kurie viršija nurodytą *supp*. parametro reikšmę. Taip toliau analizei pasirenkami puslapiai 1, 2 ir 3, nes puslapis 4 netenkino šios parametro ribos. Kiekvienam iš likusių trijų puslapių analizuojama ar vartotojai toje pačioje sesijoje lankėsi kitame nagrinėjamame puslapyje. Gaunama, kad iš puslapio 1 analizuojamos galimybės tos pačios sesijos metu naršyti tinklalapius 2, 3 ir 4; iš puslapio 2 – 3 ir 4 (nes 1 jau įtrauktas kaip galimybė ties 1 puslapiu); iš puslapio 3 – į puslapį 4 (nes sekos 1-3 ir 2-3 jau minėtos kituose puslapiuose, o į

jų kryptį nesigilinama). Iš gautų šešių sekų vėl atrenkamos tik tos, kurios viršija supp. parametro reikšmę. Šie veiksmai vykdomi tol, kol pasiekiamas veiksmų istorijos gylis *len*. (šiame pavyzdyje jis būtų lygus 4) arba kol išanalizuojami visi galimi keliai tarp analizuojamų puslapių. Galutinis rezultatas šiame pavyzdyje rodo, kad tikėtina yra tik viena seka 1-2-3-4.



3.2 pav. *Apriori algoritmas*

Kadangi šio komponento tikslas yra pateikti sekančio vartotojo galimo aplankyti puslapio nuspėjimui būtinus duomenis, tai šį komponentą rekomenduojama paleisti kas tam tikrą laiko tarpą ir taip atnaujinti jo sugeneruojamus duomenis. Koks turėtų būti šio komponento vykdymo dažnumas priklauso nuo pačioje sistemoje vykdomų atnaujinimų ir vartotojų elgsenos pokyčių tikimybės. Pavyzdžiui jei tinklalapis yra kasdien atnaujinamas, papildomas naujais puslapiais ar keičiamas esamų puslapių tarpusavio nuorodų ryšys, tada *Algo* komponentas taip pat turėtų būti vykdomas bent kartą per dieną. Kitu atveju nuspėjant vartotojo naršymo sekančius žingsnius bus remiamasi pasenusia tinklalapio puslapių struktūros informacija. Kitas galintis įtakoti faktorių – vartotojų elgsenos pokyčiai. Jei tinklalapis pasižymi tam tikru sezoniškumu, tada komponentą *Algo* būtina vykdyti atitinkamai pritaikant kiekvienam sezonui atskirai. Pavyzdžiui jei elektroninė parduotuvė prekiauja sporto reikmenimis, tai vasarą vyrauja vasaros sporto prekių aktyvesnis naršymas, o žiemą – žiemos ar patalpų sporto prekių aktyvumas. Atitinkamai *Algo* komponentas turėtų būti paleidžiamas kiekvienam sezonui atskirai, kad vasaros sezonu nebūtų tikimasi, kad vartotojas dažniausiai naršys po žiemos sporto inventoriaus kategorijas.



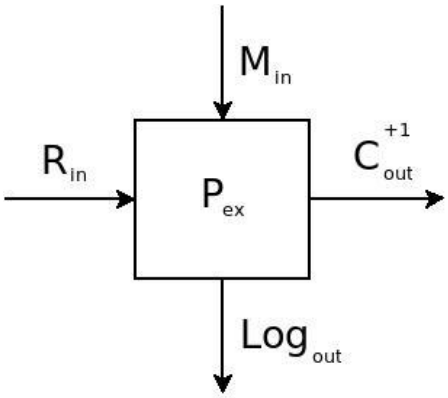
3.3 pav. *Komponentas Algo*

Komponentas *Algo* vykdymo metu naudoja vartotojo naršymo sistemoje duomenis, todėl jį vykdant būtina, kad jau būtų sukauptas atitinkamas vartotojo naršymo sistemoje duomenų kiekis. Tai lemia, kad siūlomas dedikuotojo podėlio sudarymo algoritmas negalės būti naudojamas vos tik sukūrus naują tinklalapį, t. y. sistemos optimizavimas galės būti vykdomas tik po tam tikro laiko, kada jau dalis vartotojų padirbo su naujai sukurta sistema. Šis trūkumas nėra esminis, nes dažnai kritinių sekcijų problema iškyla būtent tada,

kada yra sukaupiamas pakankamai didelis duomenų kiekis. Taip pat norint paspartinti dedikuotojo podėlio sudarymo algoritmo naudojimą realiems klientams, galima naujai sukurtą sistemą pirma duoti naudotis testinėje aplinkoje, testavimo metu surinkti pakankamai duomenų apie tipinę vartotojų elgseną ir tik tada viešai pateikti šią sistemą, su įgalintu dedikuotojo podėlio naudojimu.

3.3 Komponentas P_{ex}

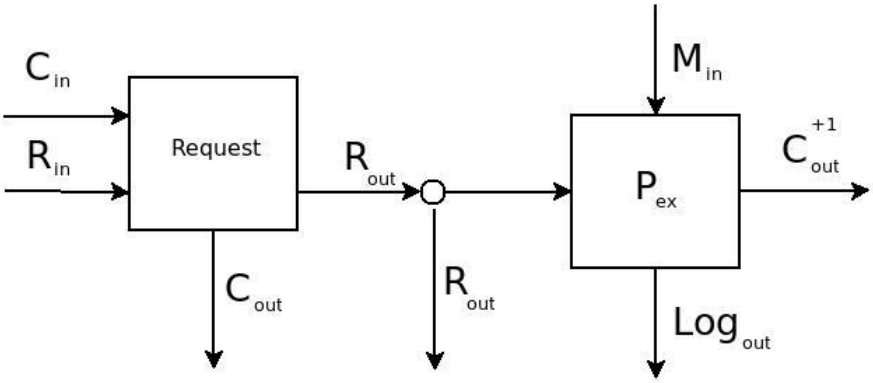
Šis komponentas yra svarbiausias siūlomoje architektūroje ir simbolizuoja sekančios užklausos podėlio paruošimo procedūrą. Šiam komponentui reikalingos dvi įvestys – esama užklausa (R_{in}) ir matrica (M_{in}) kurią sugeneravo komponentas *Algo* (žr. 3.4 pav.). Procedūra grąžina dvi išvestis: tai užklausos istorija (Log_{out}) kuri gali būti vėl naudojama komponento *Algo*, ir sekančios užklausos podėlis (C_{out}^{+1}). Šis komponentas atstoja 3.4 paveiksle vaizduojamą *PreCache* bloką.



3.4 pav. *Komponentas Pex*

3.4 Apibendrinta išankstinio dedikuotojo podėlio sudarymo schema

Tiesiogiai vartotojo (*roboto*) užklausos apdorojime dalyvaus du tarpusavyje sujungti komponentai; – komponentas *Request* ir P_{ex} (žr. 3.5 pav.). Komponentas *Algo* vartotojo užklausos apdorojime tiesiogiai nedalyvauja, todėl nėra pažymėtas grafike. Komponentas *Algo* yra vykdomas tik vieną kartą, todėl komponentui P_{ex} yra paduoda stacionari, nuo vartotojų užklausų nekintanti matrica.



3.5 pav. *Komponentų Request ir Pex sąveika*

4 SIŪLOMO IŠANKSTINIO DEDIKUOTOJO PODĖLIO SUDARYMO METODO EFEKTYVUMO TYRIMAS

Norint įvertinti siūlomo metodo tikslingumą, vykdomas tyrimas, kurio eiga yra beveik analogiška kontrolinių duomenų generavimui, aprašytam 2 skyriuje. Tyrimo eigoje naudojamas Algo komponentas su skirtingais parametrais. Tyrimo rezultatai pateikti analogiškose lentelėse kaip 1 lentelėje.

Apriori algoritmas generuoja kito žingsnio elementų rinkinius kandidatus tik iš rastų dažnų rinkinių prieš tai atliktame žingsnyje. Pagrindinė intuicija yra ta, kad kiekvienas dažnas elementų rinkinio poaibis turi būti dažnas rinkinys. Todėl rinkiniai kandidatai, sudaryti *k* elementų, generuojami sujungiant dažnus elementų rinkinius, turinčius *k-1* elementų, kurie tenkina minimalų pasikartojimų skaičių[3][4].

Norint išgauti tikslesnius rezultatus *apriori* algoritmo tyrime, yra tikslinga skaičiavimus atlikti keletą kartu su skirtingais parametrais. Tyrimo metu naudojami nustatymai pateikti 4 lentelėje.

4 lentelė. Atliekami tyrimai ir jų parametrai

Tyrimas	***Palaikymas (supp.)***	***Pasikliovimas (conf.)***	***Ilgis (len.)***	***Vėlinimas (delay)***
A50/50/0	0.5 (50%)	0.5 (50%)	2	0 s
A50/50/1	0.5 (50%)	0.5 (50%)	2	1 s
A50/30/0	0.5 (50%)	0.3 (30%)	2	0 s
A50/30/1	0.5 (50%)	0.3 (30%)	2	1 s
A50/70/0	0.5 (50%)	0.7 (70%)	2	0 s
A50/70/1	0.5 (50%)	0.7 (70%)	2	1 s
A70/70/0	0.7 (70%)	0.7 (70%)	2	0 s
A70/70/1	0.7 (70%)	0.7 (70%)	2	1 s
A70/50/0	0.7 (70%)	0.5 (50%)	2	0 s
A70/50/1	0.7 (70%)	0.5 (50%)	2	1 s
A70/90/0	0.7 (70%)	0.9 (90%)	2	0 s
A70/90/1	0.7 (70%)	0.9 (90%)	2	1 s
A90/90/0	0.9 (90%)	0.9 (90%)	2	0 s
A90/90/1	0.9 (90%)	0.9 (90%)	2	1 s
A90/70/0	0.9 (90%)	0.7 (90%)	2	0 s
A90/70/1	0.9 (90%)	0.7 (70%)	2	1 s

4.1 Apriori algoritmas Algo komponento rėmuose ir matricos sudarymas

Tam kad *apriori* algoritmo rezultatai būtų galimi panaudoti P_{ex} komponento, turi būti sudaryta tikimybių matrica. (žr. 3.3 ir 3.4 pav.). Pats *apriori* algoritmas savaime negrąžina matricos (žr. 4.1 pav.), todėl reikia atlikti papildomus veiksmus matricos sudarymui.

```
{42} => {91} 0.97     0.9797980   1.010101
{91} => {30} 0.97     1.0000000   1.010101
{30} => {91} 0.97     0.9797980   1.010101
{91} => {24} 0.97     1.0000000   1.010101
{24} => {91} 0.97     0.9797980   1.010101
```

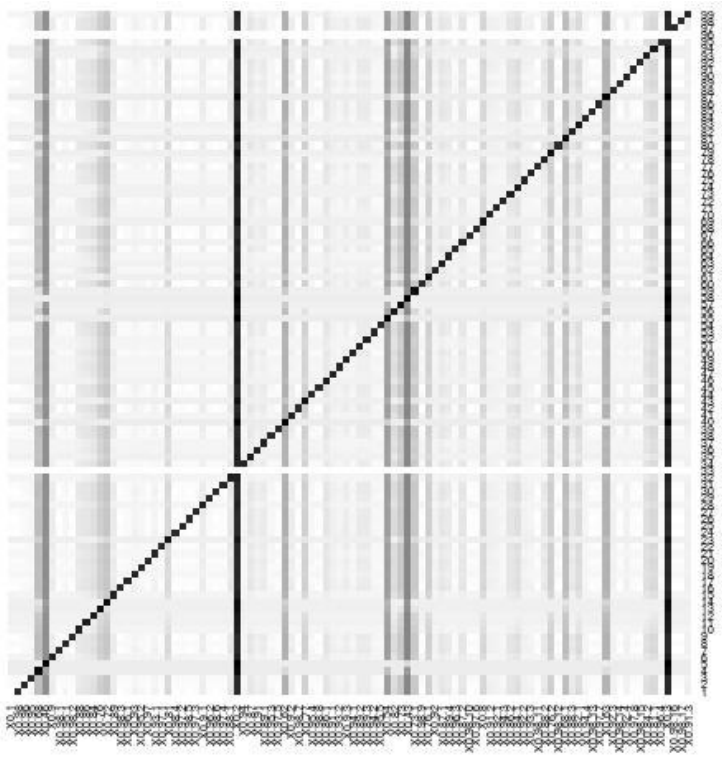
4.1 pav. *Apriori algoritmo grąžinami rezultatai*

Kaip jau buvo minėta P_{ex} komponentui turi būti pateikiama tikimybių matrica pagal kurią komponentas galėtų nuspėti kliento sekantį žingsnį. Ją sudarysime performatuodami apriori algoritmo grąžinamus duomenis. Apriori algoritmo pateikiami duomenys formatu *{A} => {B} support, confidence, lift*. Kur *{A}* yra nurodomas pradinis puslapis, *{B}* nurodomas sekančiu žingsniu siekiamas puslapis. *Support* nurodomas kaip dažnai toks sąryšis kartojasi. *Confidence* nurodo kaip dažnai yra įvykdomas *{B}* kai yra įvykdomas *{A}*. *Lift* nurodo taisykles stiprumą. Pavyzdinė matrica (žr. 5 lentelę) remiantis 4.1 paveiksle vaizduojamais Apriori algoritmo grąžinami rezultatai. Sudarant matricą *{A}* puslapiai yra sudedami *x* ašyje, o *{B}* puslapiai *y* ašyje.

5 lentelė. Pavyzdinė matrica

	42	91	30	24
91	0.98	0.00	0.98	0.98
30	0.00	0.98	00.00	0.00
91	0.00	0.00	00.00	0.00
24	0.00	0.98	00.00	0.00

Tinklalapis ir ryšiai tarp jo puslapių sudaro tarsi grafą. Šis grafas yra sugeneruotas iš 100 mazgų (puslapių). Kadangi tokio dydžio grafui sugeneruotą matricą aprašyti lentele yra sudėtinga, šio testavimo metu duomenys yra pateikti karščio žemėlapiu (žr. 4.2 pav.). Sudarant karščio žemėlapį *{A}* puslapiai yra sudedami *x* ašyje, o *{B}* puslapiai *y* ašyje. Susikirtimo taškuose yra atvaizduojamas *Confidence (conf.)* įvertis. Kuo *Confidence* įvertis yra arčiau 1; – tuo taškas tamsesnis (žr. 4.2 pav.).



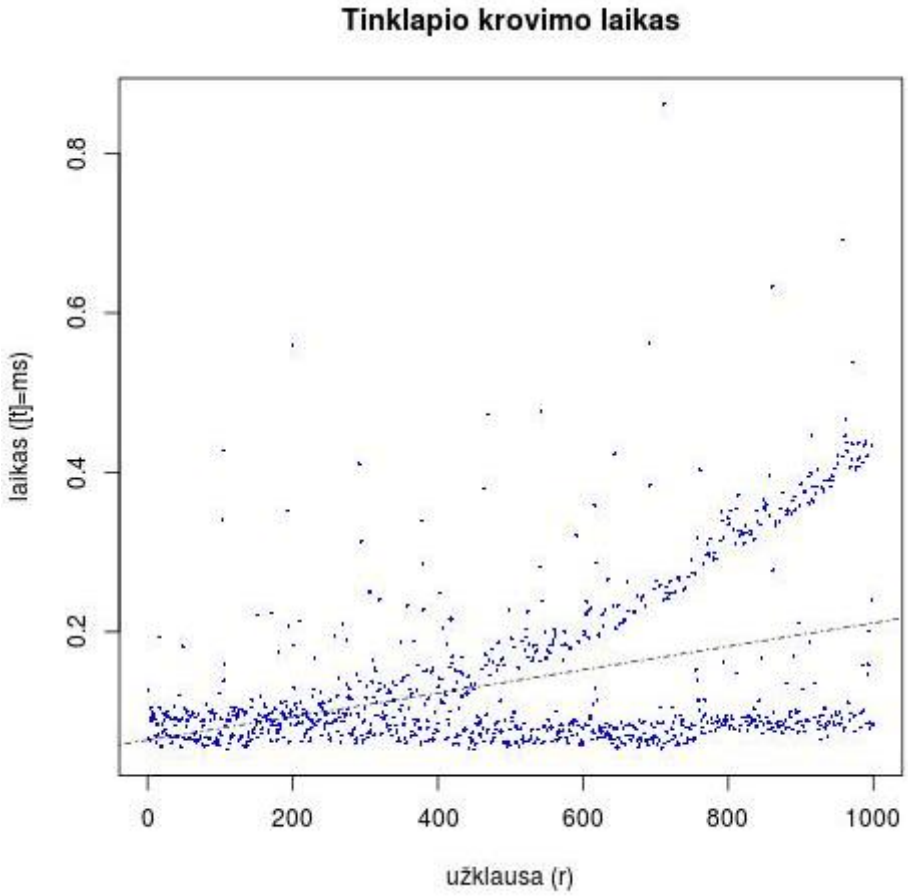
4.2 pav. *Apriori algoritmo sudarytas karščio žemėlapis*

Pasirinkta tirti tik naršymo gylį 2, nes šio algoritmo metu norima nuspėti sekantį puslapį, o ne kelis žingsnius į priekį.

4.2 Tyrimai nenaudojant vėlinimo (delay = 0)

4.2.1 Tyrimas A50/50/0

Tyrimas *A50/50/0*, kuriame palaikymas (*supp.*) yra *0.5 (50%)*, pasikliovimas (*conf.*) yra *0.5 (50%*), ilgis (*len.*) yra *2*. Atlikę tyrimą *A50/50/0*, iš grafiko (žr. 4.3 pav.) matome, kad puslapio užkrovimo laikas kritinėse vietose nuolat didėja dėl *kritinių vietų delsimo*.



4.3 pav. *Tyrimo A50/50/0 puslapio krovimo greitis*

6 lentelė. A50/50/0 tyrimo rezultatų lentelė

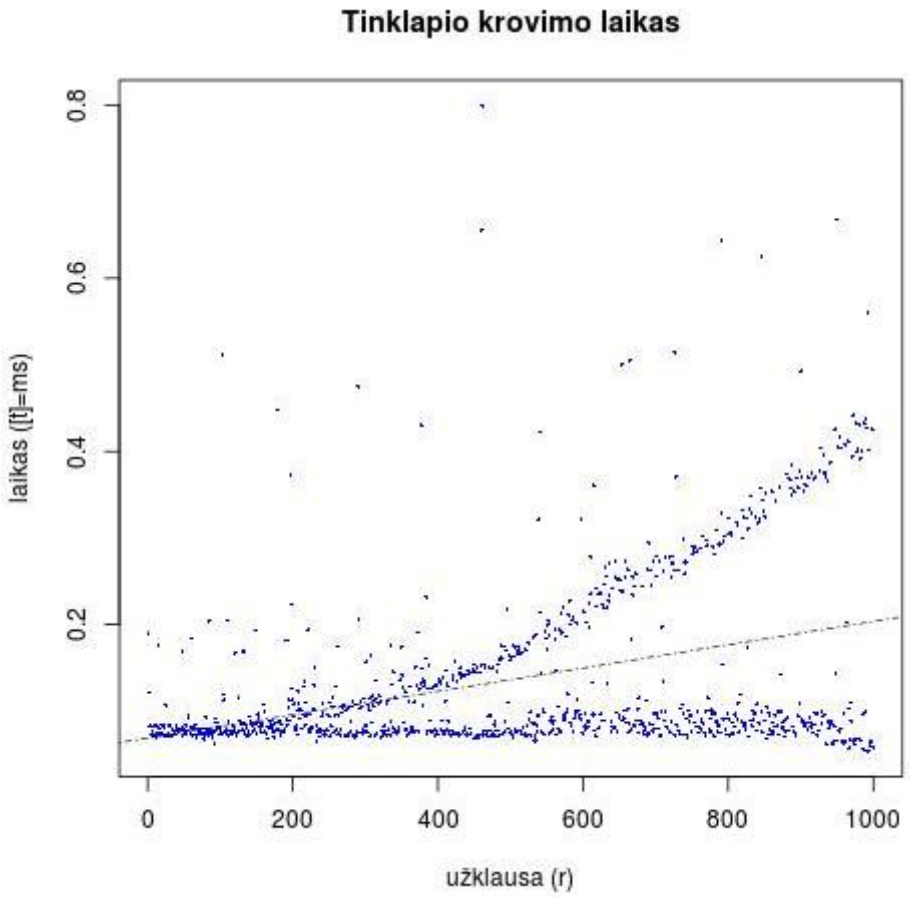
Tyrimas	***Papildomi parametrai***	***Min(ms); Mean(ms); Max(ms);***	***ΔT_{KP}; ΔT_{KN};***
A50/50/0	Supp. = 0.5 Conf. = 0.5 len. = 2 delay = 0s	Min.: 0.0520; Mean: 0.1378; Max.: 0.8620;	ΔT_{KP} = -0.0550 ms; ΔT_{KN} = 0.0196 ms;

Detalesnei analizei pasitelksime tyrimo rezultatų lentelę (žr. 6 lentelę). Lentelėje papildomi parametrai nurodomi kuriais parametrais remiantis buvo sudaryta matrica P_{ex} elementui. Bendrai tyrimo metu didžiausias užkrovimo laikas buvo *0.8620ms*, o mažiausias *0.0520ms*. Didžiausias ir mažiausias užkrovimo laikas nežymiai skiriasi nuo *KN* tyrimo. Tai galime interpretuoti kaip paklaidą dėl gan didelio užkrovimo laiko sklaidos. Atkreipdami dėmesį į užkrovimo laiko vidurkį *0.1378ms* galime pastebėti, kad lyginant su *KN* tyrimu užkrovimo laikas sutrumpėjo. Tai parodo ΔT_{KN} įvertis. Vidurkis pagreitėjo *0.0196ms*.

Taip pat galime pastebėti, kad tendencija užkrovimo laikui yra šiek tiek mažesnė lyginant su *KN* tyrimu $\Delta b_{KN} = 0.0000619ms;$

4.2.2 Tyrimas A70/70/0

Tyrimas *A70/70/0*, kuriame palaikymas (*supp.*) yra *0.7 (70%)*, pasikliovimas (*conf.*) yra *0.7 (70%*), ilgis (*len.*) yra *2*. Atlikę tyrimą *A70/70/0*, iš grafiko (žr. 4.4 pav.) matome, kad puslapio užkrovimo laikas kritinėse vietose nuolat didėja dėl *kritinių vietų delsimo*.



4.4 pav. *Tyrimo A70/70/0 puslapio krovimo greitis*

7 lentelė. A70/70/0 tyrimo rezultatų lentelė

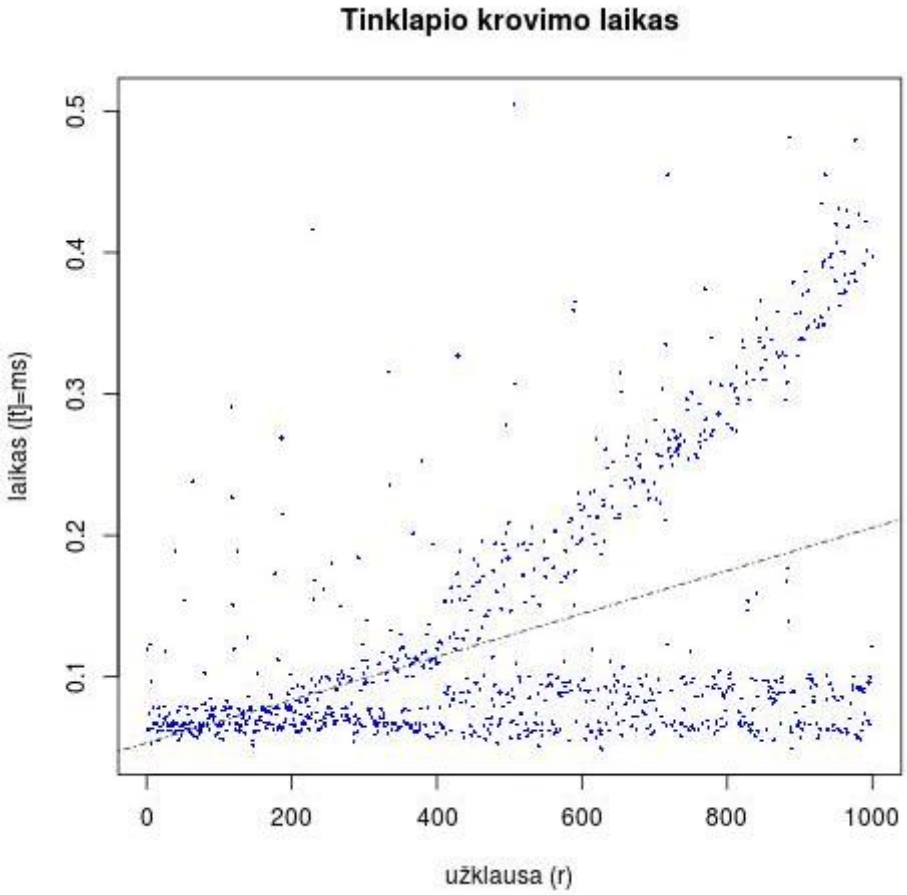
Tyrimas	***Papildomi parametrai***	***Min(ms); Mean(ms); Max(ms);***	***ΔT_{KP}; ΔT_{KN};***
A70/70/0	Supp. = 0.7 Conf. = 0.7 len. = 2 delay = 0s	Min.: 0.0540; Mean: 0.1364; Max.: 0.7990;	ΔT_{KP} = -0.0536 ms; ΔT_{KN} = 0.0210 ms;

Detalesnei analizei pasitelksime tyrimo rezultatų lentele (žr. lentelė 7). Lentelėje Papildomi parametrai nurodomi kuriais parametrais remiantis buvo sudaryta matrica P_{ex} elementui. Bendrai tyrimo metu didžiausias užkrovimo laikas buvo *0.7990ms*, o mažiausias *0.0540ms*. Didžiausias ir mažiausias užkrovimo laikas nežymiai skiriasi nuo *KN* ir *A50/50/0* tyrimų. Tai galime interpretuoti kaip paklaidą dėl gan didelio užkrovimo laiko sklaidos. Atkreipdami dėmesį į užkrovimo laiko vidurkį *0.1364ms* galime pastebėti, kad lyginant su *KN* ir *A50/50/0* tyrimu užkrovimo laikas sutrumpėjo. Tai parodo ΔT_{KN} įvertis. Vidurkis

pagreitėjo *0.0210ms*. Taip pat galime pastebėti, kad tendencija užkrovimo laikui yra šiek tiek mažesnė lyginant su *KN* ir *A50/50/0* tyrimu $\Delta b_{KN} = 0.0000739ms; \Delta b_{A50500} = 0.000012ms;$

4.2.3 Tyrimas A90/90/0

Tyrimas *A90/90/0*, kuriame palaikymas (*supp*.) yra *0.9 (90%)*, pasikliovimas (*conf*.) yra *0.9 (90%*), ilgis (*len*.) yra *2*. Atlikę tyrimą *A90/90/0*, iš grafiko (žr. 4.5 pav.) matome, kad puslapio užkrovimo laikas kritinėse vietose nuolat didėja dėl *kritinių vietų delsimo*.



4.5 pav. *Tyrimo A90/90/0 puslapio krovimo greitis*

8 lentelė. A90/90/0 tyrimo rezultatų lentelė

Tyrimas	***Papildomi parametrai***	***Min(ms); Mean(ms); Max(ms);***	***ΔT_{KP}; ΔT_{KN};***
A90/90/0	Supp. = 0.9 Conf. = 0.9 len. = 2 delay = 0s	Min.: 0.0490; Mean: 0.1295; Max.: 0.5050;	ΔT_{KP} = -0.0467 ms; ΔT_{KN} = 0.0279 ms;

Detalesnei analizei pasitelksime tyrimo rezultatų lentele (žr. 8 lentelę). Lentelėje Papildomi parametrai nurodomi kuriais parametrais remiantis buvo sudaryta matrica P_{ex} elementui. Bendrai tyrimo metu didžiausias užkrovimo laikas buvo *0.5050ms*, o mažiausias *0.0490ms*. Didžiausias užkrovimo laikas žymiai skiriasi nuo *KN* ir *A70/70/0* tyrimų. Atkreipdami dėmesį į užkrovimo laiko vidurkį *0.1295ms* galime pastebėti, kad lyginant su *KN* ir *A70/70/0* tyrimu užkrovimo laikas sutrumpėjo. Tai parodo ΔT_{KN} įvertis. Vidurkis pagreitėjo *0.0279ms*. Taip pat galime pastebėti, kad tendencija užkrovimo laikui yra šiek tiek

mažesnė lyginant su *KN* tyrimu $\Delta b_{KN} = 0.0000565ms$;tačiau šiek tiek didesnė lyginant su *A70/70/0* tyrimu $\Delta b_{A70700} = -0.0000174ms$;

Analogiška tvarka yra atliekami kiti 4 lentelėje aprašyti tyrimai.

4.3 Tyrimų nenaudojant vėlinimo išvados

Apibendrinant galima teigti, kad tyrimuose, kuriuose vėlinimo laikas yra 0s, didesnė *supp*. parametro reikšmė leidžia labiau sumažinti puslapio užkrovimo greitį, bet pasiekti situaciją, analogišką pilno podėlio sudarymo atveju neįmanoma. Taip pat pastebima, kad teoriniu požiūriu optimalesnis yra *A70/70/0* sprendimas; ši konfigūracija lėčiau artėja prie begalybės (žr. 4.4 pav.).

9 lentelė. Apriori algoritmo tyrimo sulyginamieji rezultatai su delay = 0s parametru

Tyrimas	***Papildomi parametrai***	***Min(ms); Mean(ms); Max(ms);***	***ΔT_{KP}; ΔT_{KN};***
Kontroliniai (be podėlio) (KN)	–	Min.: 0.0520; Mean: 0.1574; Max.: 0.7400;	ΔT_{KP} = -0.0746 ms; ΔT_{KN} = 0.0000 ms;
Kontroliniai (su podėliu) (KP)	–	Min.: 0.0550; Mean: 0.0828; Max.: 0.3010;	ΔT_{KP} = 0.0000 ms; ΔT_{KN} = 0.0746 ms;
A50/50/0	Supp. = 0.5 Conf. = 0.5 len. = 2 delay = 0s	Min.: 0.0520; Mean: 0.1378; Max.: 0.8620;	ΔT_{KP} = -0.0550 ms; ΔT_{KN} = 0.0196 ms;
A50/30/0	Supp. = 0.5 Conf. = 0.3 len. = 2 delay = 0s	Min.: 0.0520; Mean: 0.1266; Max.: 0.8840;	ΔT_{KP} = -0.0438 ms; ΔT_{KN} = 0.0308 ms;
A50/70/0	Supp. = 0.5 Conf. = 0.3 len. = 2 delay = 0s	Min.: 0.0500; Mean: 0.1367; Max.: 0.7220;	ΔT_{KP} = -0.0538 ms; ΔT_{KN} = 0.0207 ms;
A70/70/0	Supp. = 0.7 Conf. = 0.7 len. = 2 delay = 0s	Min.: 0.0540; Mean: 0.1364; Max.: 0.7990;	ΔT_{KP} = -0.0536 ms; ΔT_{KN} = 0.0210 ms;
A70/50/0	Supp. = 0.7 Conf. = 0.5 len. = 2 delay = 1s	Min.: 0.0520; Mean: 0.1272; Max.: 0.6840;	ΔT_{KP} = -0.0444 ms; ΔT_{KN} = 0.0302 ms;
A70/90/0	Supp. = 0.7 Conf. = 0.9 len. = 2 delay = 0s	Min.: 0.0480; Mean: 0.1171; Max.: 0.6300;	ΔT_{KP} = -0.0343 ms; ΔT_{KN} = 0.0403 ms;
A90/90/0	Supp. = 0.9 Conf. = 0.9 len. = 2 delay = 0s	Min.: 0.0490; Mean: 0.1295; Max.: 0.5050;	ΔT_{KP} = -0.0467 ms; ΔT_{KN} = 0.0279 ms;
A90/70/0	Supp. = 0.9 Conf. = 0.7 len. = 2 delay = 0s	Min.: 0.0560; Mean: 0.1497; Max.: 0.5310;	ΔT_{KP} = -0.0669 ms; ΔT_{KN} = 0.0077 ms;

4.4 Tyrimai panaudojant vėlinimą (delay = 1)

Svarbu yra atlikti tyrimus simuliuojant vartotojo naršymą ir atliekant vėlinimą. Tyrimai yra atliekami analogiška tvarka, kokia buvo atliekami tyrimai be vėlinimo. Šie tyrimų rezultatai bus reikalingi bendram išankstinio potencialiai vartotojo peržiūrimų internetinių sistemų duomenų talpinimo į podėlį sprendimo efektyvumui įvertinti.

10 lentelė. Tyrimai su Apriori algoritmo tyrimo sulyginamieji rezultatai su delay = 1s parametru

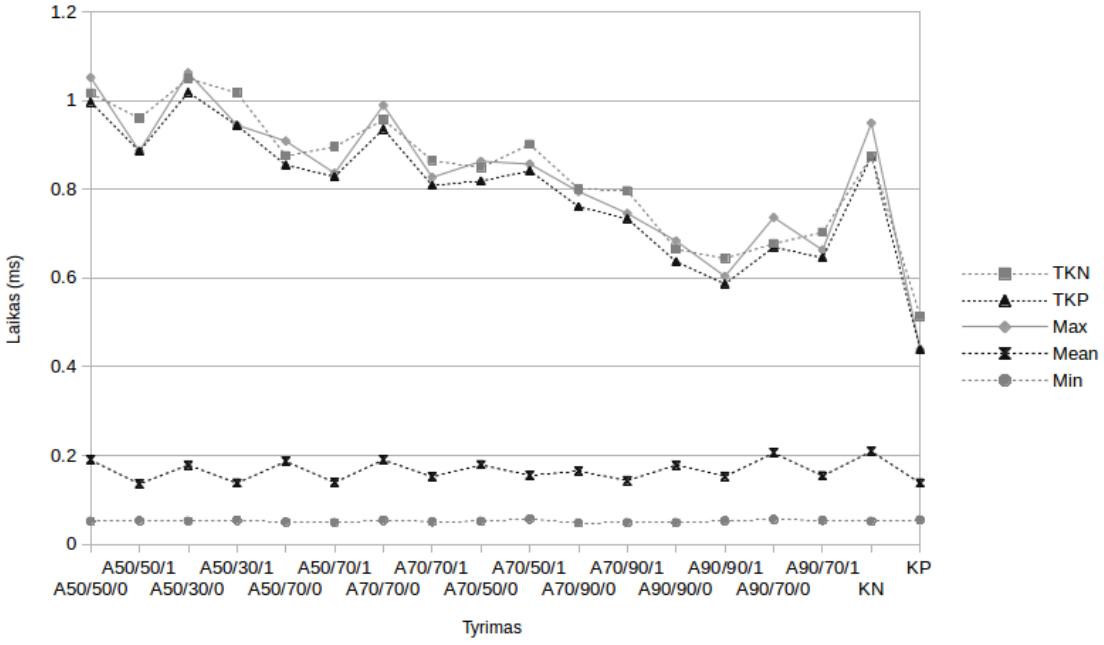
Tyrimas	***Papildomi parametrai***	***Min(ms); Mean(ms); Max(ms);***	***ΔT_{KP}; ΔT_{KN};***
Kontroliniai (be podėlio) (KN)	–	Min.: 0.0520; Mean: 0.1574; Max.: 0.7400;	ΔT_{KP} = -0.0746 ms; ΔT_{KN} = 0.0000 ms;
Kontroliniai (su podėliu) (KP)	–	Min.: 0.0550; Mean: 0.0828; Max.: 0.3010;	ΔT_{KP} = 0.0000 ms; ΔT_{KN} = 0.0746 ms;
A50/50/1	Supp. = 0.5 Conf. = 0.5 len. = 2 delay = 1s	Min.: 0.0530; Mean: 0.0832; Max.: 0.7500;	ΔT_{KP} = -0.0004 ms; ΔT_{KN} = 0.0742 ms;
A50/30/1	Supp. = 0.5 Conf. = 0.3 len. = 2 delay = 1s	Min.: 0.0540; Mean: 0.0838; Max.: 0.8070;	ΔT_{KP} = -0.0001 ms; ΔT_{KN} = 0.0736 ms;
A50/70/1	Supp. = 0.5 Conf. = 0.7 len. = 2 delay = 1s	Min.: 0.0490; Mean: 0.0904; Max.: 0.6970;	ΔT_{KP} = -0.0076 ms; ΔT_{KN} = 0.0670 ms;
A70/70/1	Supp. = 0.7 Conf. = 0.7 len. = 2 delay = 1s	Min.: 0.0510; Mean: 0.1011; Max.: 0.6750;	ΔT_{KP} = -0.0183 ms; ΔT_{KN} = 0.0563 ms;
A70/50/1	Supp. = 0.7 Conf. = 0.5 len. = 2 delay = 1s	Min.: 0.0570; Mean: 0.0980; Max.: 0.7020;	ΔT_{KP} = -0.0152 ms; ΔT_{KN} = 0.0594 ms;
A70/90/1	Supp. = 0.7 Conf. = 0.9 len. = 2 delay = 1s	Min.: 0.0490; Mean: 0.0947; Max.: 0.6020;	ΔT_{KP} = -0.0119 ms; ΔT_{KN} = 0.0627 ms;
A90/90/1	Supp. = 0.9 Conf. = 0.9 len. = 2 delay = 1s	Min.: 0.0530; Mean: 0.1000; Max.: 0.4510;	ΔT_{KP} = -0.0172 ms; ΔT_{KN} = 0.0574 ms;
A90/70/1	Supp. = 0.9 Conf. = 0.7 len. = 2 delay = 1s	Min.: 0.0540; Mean: 0.1002; Max.: 0.5090;	ΔT_{KP} = -0.0174 ms; ΔT_{KN} = 0.0572 ms;

Apžvelgę sulyginamuosius rezultatus su vėlinimu (žr. 10 lentelę) galime pastebėti, kad *A90/70/1* tyrimo metu gauti duomenys buvo patys prasčiausi; – vidutinis atsako laikas *0.1002ms*. Taip pat svarbu atkreipti dėmesį į *A90/90/1* tyrimą: jis nedaug kuo skiriasi nuo *A90/70/1* tyrimo ir yra vienas iš blogiausių

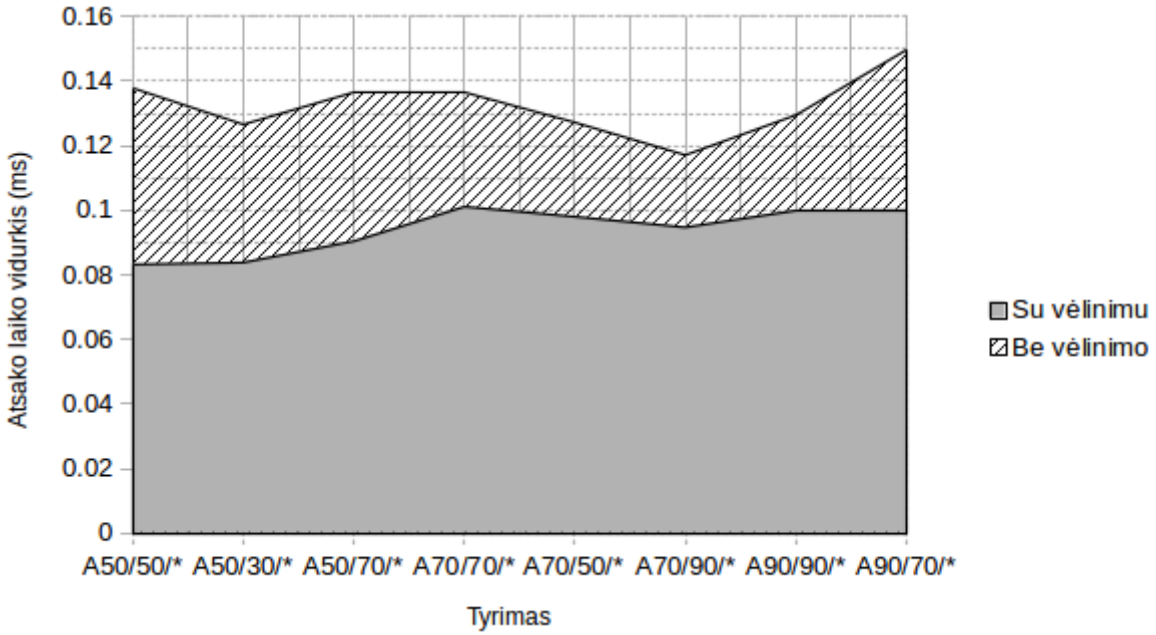
gautų rezultatų naudojant vėlinimą, nors nenaudojant vėlinimo buvo vienas geriausių. Taip pat reiktų atkreipti dėmesį į tyrimą *A50/50/1*, jo $\Delta T_{KP} = -0.0004ms$; yra artimas *KP* tyrimo vidurkiui (artimas geriausiam galimam pasiekti rezultatui).

4.5 Siūlomo sprendimo tyrimų apibendrinimas

Atlikome tyrimus naudojant ir nenaudojant vėlinimo su tokiais pat *apriori* algoritmo parametrais. Apibendrindami, suveskime gautus rezultatus į diagramą (žr. 4.6 pav.). Matome, kad tyrimuose didžiausios reikšmės yra didžiausias atsakymo laikas (*max*), tačiau aiškios *max* reikšmės tendencijos keičiant *apriori* algoritmo parametrus nematome.



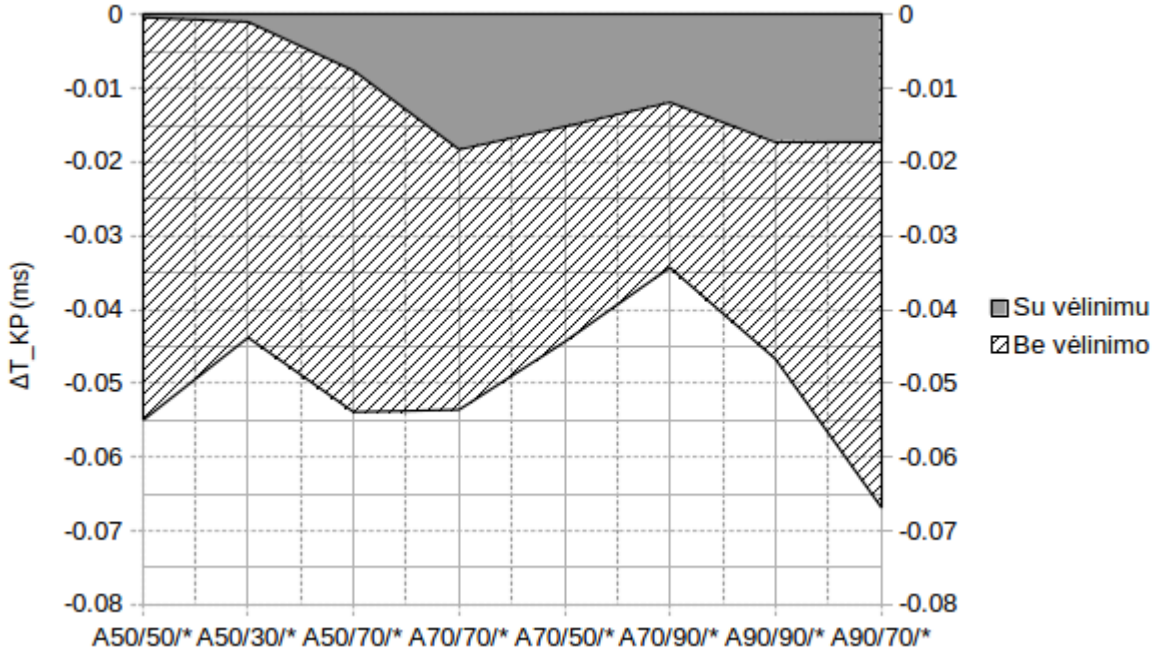
4.6 pav. *Apriori tyrimų sulyginamieji rezultatai*



4.7 pav. *Tyrimų su ir be vėlinimo vidurkio įverčiai. (Kuo arčiau 0, tuo geriau)*

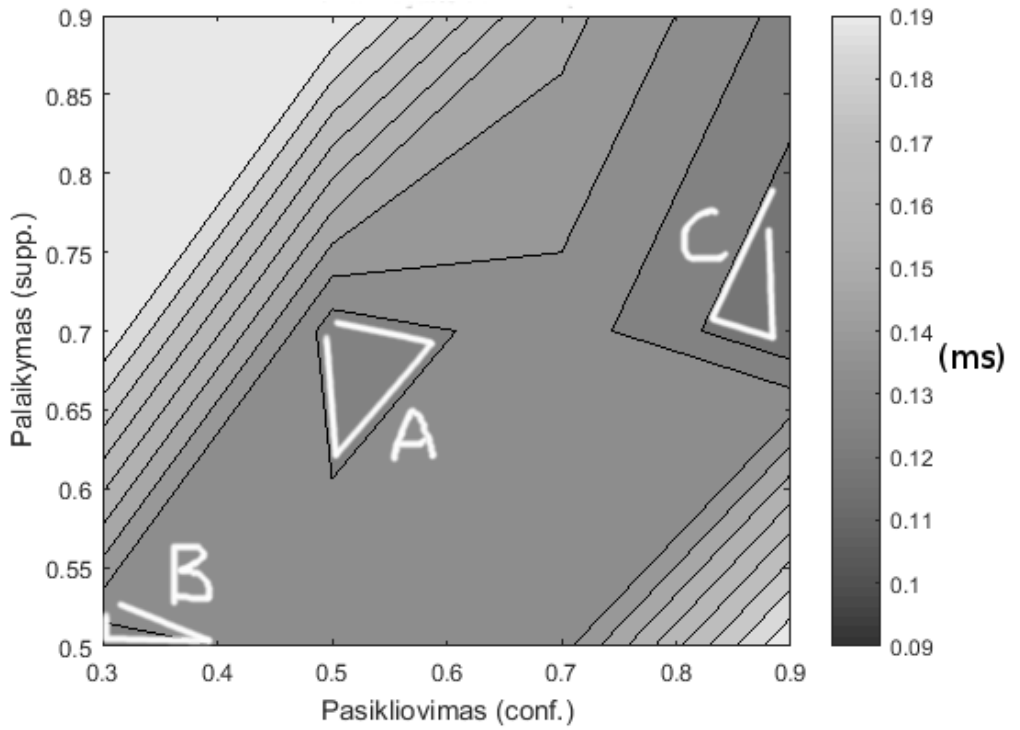
Lygindami tyrimuose gautus puslapių krovimo laiko vidurkius (žr. 4.7 pav.) galime pastebėti, kad didžiausias puslapio užkrovimo vidurkis priklauso tyrimui *A90/70/0*. Galėtume spėti, jog su didesniais *conf*. ir *supp*. parametrais, yra prasčiau nuspėjamas puslapis, kurį klientas atvers sekančiu žingsniu. Tačiau, tai paneigia tyrimas *A50/50/0* turintis mažesnes *supp*. ir *conf*. reikšmes. Mažiausią puslapio krovimo laiko vidurkį turi tyrimas *A50/50/1*. Verta pastebėti, kad nenaudojant vėlinimo šis tyrimas suteikia vienus prasčiausių rezultatų. Taip pat verta pastebėti, kad tyrimuose, kuriuose yra naudojamas vėlinimas, krovimo laiko vidurkis yra visada mažesnis, nei analogiškame tyrime nenaudojant vėlinimo.

Norėdami nustatyti kuris tyrimas suteikė efektyviausius rezultatus, turime palyginti ΔT_{KP} įverčius, tam kad pamatytume, kurio tyrimo puslapio užkrovimo laikas yra arčiausias galimam optimaliam rezultatui (žr. 2.3.2 skyrių). Lygindami atliktų tyrimų ΔT_{KP}įverčius (žr. 4.8 pav.), pastebime, kad mažiausią skirtumą nuo galimo maksimumo, turi tyrimas *A50/50/1*. Tai reiškia, kad šiuo tyrimo metu gauti rezultatai yra efektyviausi. Mažiausiai efektyvūs yra tyrimo *A90/70/0* rezultatai.

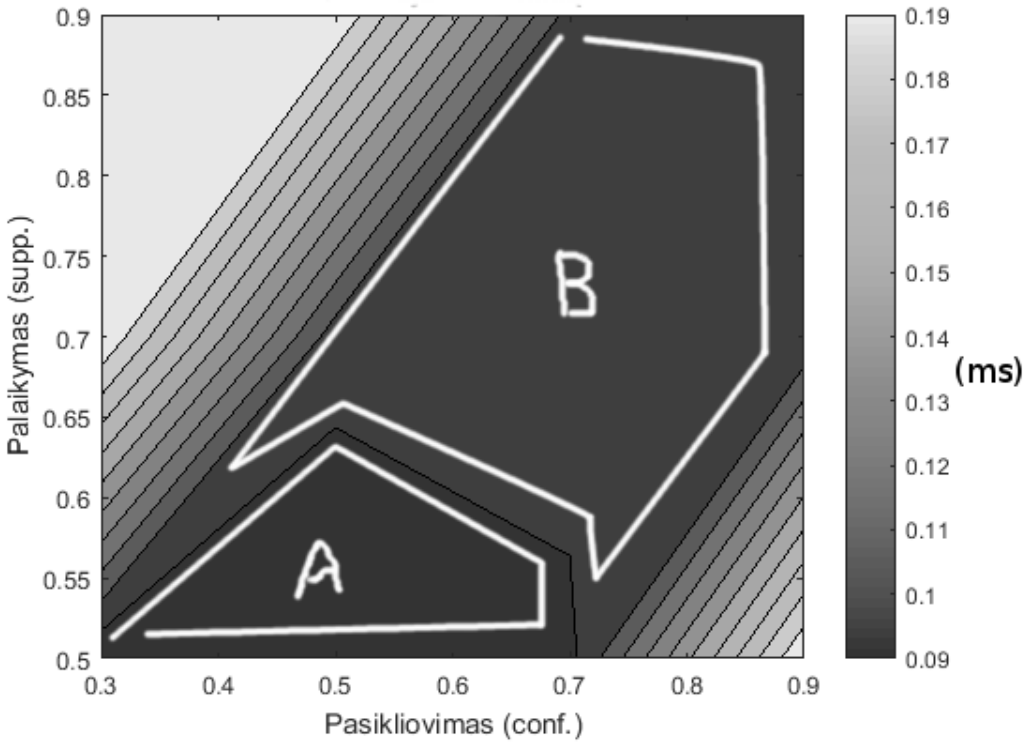


4.8 pav. *Tyrimų su ir be vėlinimo ΔT_{KN} įvertis. (Kuo arčiau 0, tuo geriau)*

Įvertinkime *conf*. ir *supp*. reikšmių įtaką puslapio vidutiniam krovimo greičiui nenaudojant vėlinimo (žr. 4.9 pav.). Galime pastebėti, kad tarp *conf*. ir *supp*. reikšmių pasiskirstymo tiesinės priklausomybės puslapio vidutiniam krovimo greičiui nėra. Tačiau matome, kad yra 3 *supp*. ir *conf*. reikšmių kombinacijos, pažymėtos *A*, *B* ir *C* plotais, kuriomis vidutinis puslapio krovimo greitis yra geresnis. Vertindami *conf*. ir *supp*. reikšmių įtaką puslapio vidutinio krovimo greičiui naudojant vėlinimą (žr. 4.10 pav.), galime pastebėti, jog priešingai nei nenaudojant vėlinimo, *conf*. ir *supp*. reikšmės tiesiškai įtakoja vidutinį puslapio krovimo greitį naudojant vėlinimą. Geriausios kombinacijos pažymėtos *A* ir *B* plotais. Taip pat matome, kad kuo mažesnė *conf*. ir *supp*. reikšmė, tuo vidutinis puslapio krovimo greitis yra didesnis.



4.9 pav. *conf. ir supp. parametrų įtaka tinklapio užkrovimo greičio vidurkiui be vėlinimo.*



4.10 pav. *conf. ir supp. parametrų įtaka tinklapio užkrovimo greičio vidurkiui be vėlinimo.*

Apibendrinant galima teigti, kad siūlomas išankstinio podėlio sprendimas tirtoje situacijoje padeda optimizuoti kritinių vietų vykdymo laiką, o geriausi rezultatai pasiekiami tada, kai palaikymo (*supp.*) ir pasikliovimo (*conf.*) reikšmės yra lygios 50 procentų, o tarp vartotojo užklausų yra vėlinimas. Tačiau vertėtų pastebėti, kad šie nustatymai gali kisti, priklausomai nuo internetinės svetainės struktūros ir vartotojų elgsenos, todėl praktikoje naudojant siūlomą metodą, siūloma kartais atlikti analogiškus bandymus patiems ir įvertinti ar tikrai sistemos naudojami parametrai yra optimalūs būtent toje situacijoje.

DARBO IŠVADOS

1. Atlikus literatūros ir kitų programuotojų apžvalgą pastebėta, kad šiuo metu nepilnai išnaudojamos *+1* podėlio sudarymo galimybės, nes šiuolaikinė techninė įranga leidžia vienu metu apdoroti didesnį užklausų kiekį, o vartotojo naršymo viename puslapyje laiką galima išnaudoti sekančio jo žingsnio nuspėjimui ir individualaus podėlio paruošimui.

2. Siūlomam *+1* gylio podėlio sprendimui pritaikytas a*priori* algoritmas, kuris leidžia talpinti sekančio žingsnio metu galimų aplankyti puslapių duomenis, o tinklalapio atsako laiką leido sumažinti vidutiniškai nuo *0.1574ms* naudojant 0-nio gylio podėlį iki *0.0828ms* naudojant išankstinio podėlio sudarymo metodą.

3. Tyrimu metu nustatyta, kad siūlomas sprendimas neleidžia sumažinti tinklalapio atsako laiko, jei nėra *vėlinimo* tarp tinklalapio užkrovimo, o įvertinus, kad vartotojas natūraliai naršydamas vidutiniškai užtrunka daugiau nei *1s* viename tinklalapyje, galima teigti, kad siūlomas sprendimas tinkamai optimizuos tinklalapių atsako laiką realiems vartotojams, o be perstojo naršantiems internetiniams robotams veiks neoptimaliai ir robotai turės ilgiau laukti tinklalapio užsikrovimo.

4. Siūlomas *+1* gylio podėlis optimaliai butų taikomas sistemose su kritinėmis sekcijomis prioritetiniu vartotojų naršymo metu tinklalapio atsako greičiui sumažinti, nes atlikti tyrimai rodo, jog siūlomas metodas leidžia vidutiniškai ~47% sumažinti tinklalapio atsako laiką.

LITERATŪRA

1. Curl dokumentacija. Iš *curl the man page* [interaktyvus] [žiūrėta 2016 m. Kovo 22 d.]. Prieiga per internetą: <https://curl.haxx.se/docs/manpage.html>.
2. Roberto J. Bayardo Jr., „Efficiently Mining Long Patterns from Databases“. Iš *IBM Almaden Research Center* [interaktyvus]. 1998 m., [žiūrėta 2016 m. Kovo 22 d.] . Prieiga per internetą: <http://www.cs.sfu.ca/CourseCentral/741/jpei/readings/baya98.pdf>.
3. Romanas Tumasonis, „Dažnų Sekų Paieška Dideliuose Duomenų Masyvuose“. Iš *Matematikos Ir Informatikos Institutas, Vytauto Didžiojo Universitetas*, [interaktyvus], 2007 m., [žiūrėta 2016 m. Kovo 22 d.]. Prieiga per internetą: <http://www.mii.lt/files/mii_dis_07_tumasonis.pdf>.
4. Loreta Savulionienė, „Dažnų Posekių Paieškos Algoritmai Ir Jų Rezultatai“. Iš *Matematikos ir informatikos institutas , Vilniaus universitetas,* [interaktyvus], [žiūrėta 2016 m. Kovo 22 d.]. Prieiga per internetą: <http://eif.viko.lt/uploads/file/eif_konf_2011/Visi_sudeti_rev1_2011_final_rev9%28STR17%29.pdf >.
5. *Fusion Middleware Configuration Guide for Oracle Business Intelligence Discoverer* (Oracle dokumentacijų rinkinys) [interaktyvus], [žiūrėta 2016 m. Kovo 22 d.]. Prieiga per internetą: <https://docs.oracle.com/cd/E21764_01/bi.1111/b40107/webcache.htm#BIDCG272>
6. *HTTP caching* (Google programuotojų dokumentacija) [interaktyvus], [žiūrėta 2016 m. Kovo 23 d.]. Prieiga per internetą: <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>.
7. Ernestina Menasalvas, „Subsessions: A granular approach to click path analysis“. Iš *International Journal Of Intelligent Systems* [interaktyvus]. 2015 m., [žiūrėta 2016 m. Kovo 23 d.]. Prieiga per internetą: <http://www.researchgate.net/publication/3949814>.
8. Naga Lakshmi, Raja Sekhara Rao , Sai Satyanarayana Reddy, „An Overview of Preprocessing on Web Log Data for Web Usage Analysis“. Iš *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, ISSN: 2278-3075, Volume-2, Issue-4, March 2013, [interaktyvus], [žiūrėta 2016 m. Kovo 23 d.]. Prieiga per internetą: <http://docplayer.net/6488906-An-overview-of-preprocessing-on-web-log-data-for-web-usage-analysis.html>
9. Robert Cooley*, Bamshad Mobasher, Jaideep Srivastava „Data Preparation for Mining World Wide Web Browsing Patterns“. Iš *Knowledge and Information Systems 1 (1999)*, [interaktyvus], 1999 m., [žiūrėta 2016 m. Kovo 23 d.]. Priega per internetą: <http://facweb.cs.depaul.edu/mobasher/classes/ect584/papers/cms-kais.pdf>
10. Alan L. Montgomery, Shibo Li, Kannan Srinivasan, John C. Liechty „Modeling Online Browsing and Path Analysis Using Clickstream Data“, Trečiasis leidimas, [interaktyvus], 2004 m., [žiūrėta 2016 m. Kovo 23 d.]. Prieiga per internetą: <https://www.andrew.cmu.edu/user/alm3/papers/purchase%20conversion.pdf>

11. Sarina Sulaiman, Siti Mariyam Shamsuddin, Nor Bahiah Ahmad „Meaningless to Meaningful Web Log Data for Generation of Web Pre-caching Decision Rules Using Rough Set“. Iš *Machine Intelligence Research Labs (MIR Labs)*, [interaktyvus], [žiūrėta 2016 m. Kovo 26 d.]. Priega per internetą: <http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6322848>
12. Jaideep Srivastava, Robert Cooley, Mukund Deshpande, Pang-Ning Tan „Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data“. Iš *SIGKDD Explorations* [interaktyvus], 2000 m., [žiūrėta 2016 m. Kovo 26 d.]. Prieiga per internetą: <http://nlp.uned.es/WebMining/Tema5.Uso/srivastava2000.pdf>
13. Ma, Yan, et al. "Cache Management of Big Data in Equipment Condition Assessment." MATEC Web of Conferences. Vol. 55. EDP Sciences, 2016.
14. Taylor, Teryl, et al. "Cache, Trigger, Impersonate: Enabling Context-Sensitive Honeyclient Analysis On-the-Wire." (2016).
15. Wang, Xiao Sophia, Arvind Krishnamurthy, and David Wetherall. "How much can we micro-cache web pages?." Proceedings of the 2014 Conference on Internet Measurement Conference. ACM, 2014.
16. Mahad, Fairuz S., and W. A. N. WAN-KADIR. "IMPROVING WEB SERVER PERFORMANCE USING TWO-TIERED WEB CACHING." Journal of Theoretical & Applied Information Technology 52.3 (2013).
17. Chen, Jay, and Lakshmi Subramanian. "Interactive web caching for slow or intermittent networks." Proceedings of the 4th Annual Symposium on Computing for Development. ACM, 2013.
18. Bhadauriya, Veena Singh, Bhupesh Gour, and Asif Ullah Khan. "Improved Server Response using Web Pre-Fetching: A Review." International Journal of Advances in Engineering & Technology 6.6 (2014): 2508.