UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Blaž Hafnar

OBVLADOVANJE PREOBREMENITEV VIROV ERP PRI IZVAJANJU PAKETNIH OBDELAV

MAGISTRSKO DELO

Mentor: prof. dr. Nikolaj Zimic

Ljubljana, 2015

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Večna pot 113
1000 Ljubljana, Slovenija
telefon: 01 47 98 100
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št.: 130-MAG-ISO/2015
Datum: 2. 6. 2015

Blaž HAFNAR, univ. dipl. inž. rač. in inf.

L j u b l j a n a

Fakulteta za računalništvo in informatiko Univerze v Ljubljani izdaja naslednjo magistrsko nalogo

Naslov naloge: **Obvladovanje preobremenitev virov ERP pri izvajanju paketnih obdelav**

Handling batch processing overloads in ERP

Tematika naloge:

Danes se srečujemo z izvajanjem paketnih obdelav v praktično vseh poslovnih računalniških sistemih. Zaradi novih poslovnih potreb po obdelavi podatkov in večanja količine podatkov se z leti potreba po uporabi paketnih obdelav povečuje. Za izvajanje opravil, ki zahtevajo obdelavo večje količine podatkov, uporabljajo paketne obdelave tudi uporabniki sistema ERP Microsoft Dynamics AX. Takšne paketne obdelave navadno povzročajo večje obremenitve virov. Sočasno izvajanje več takšnih paketnih obdelav pa lahko vodi v preobremenitve virov in posledično slabo odzivnost sistema.
V magistrski nalogi izdelajte rešitev za obvladovanje preobremenitev virov omenjenega sistema ERP pri izvajanju paketnih obdelav. V nalogi najprej opredelite vire in breme sistema. Nato preučite možnosti za zbiranje informacij o obremenitvi virov, koncepte obvladovanja preobremenitve virov in možnost razširitve obstoječega algoritma za razporejanje paketnih obdelav. Izdelajte rešitev in jo preizkusite na konkretnih primerih. Ustreznost delovanja rešitve ovrednotite na podlagi rezultatov opravljenih meritev.

M e n t o r :

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

D e k a n :

prof. dr. Nikolaj Zimic

I Z J A V A O A V T O R S T V U

magistrskega dela

Spodaj podpisani **Blaž Hafnar**,

z vpisno številko **63030123**,

sem avtor magistrskega dela z naslovom

Obvladovanje preobremenitev virov ERP pri izvajanju paketnih obdelav.

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod vodstvom mentorja

  prof. dr. Nikolaja Zimica;

- so elektronska oblika magistrskega dela, naslova (slov., angl.), povzetka (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela;

- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 3. 7. 2015 Podpis avtorja:

Zahvala

Na tem mestu bi se rad zahvalil mentorju prof. dr. Nikolaju Zimicu za strokovno pomoč in usmerjanje pri izdelavi magistrske naloge. Hvala tudi podjetju Adacta d.o.o., ki mi je omogočilo izdelavo magistrske naloge. Ob tej priložnosti bi se zahvalil tudi svojim najbližjim, ki so mi bili v podporo, ter vsem ostalim, ki so mi v tem času kakorkoli stali ob strani.

Kazalo vsebine

Kazalo slik

Kazalo tabel

Seznam kratic

AIF	(*Application Integration Framework*) Ogrodje za integracije z aplikacijami.
AOS	(*Application Object Server*) Aplikacijski strežnik, komponenta srednjega nivoja v trinivojski arhitekturi sistema Microsoft Dynamics AX.
AOT	(*Application Object Tree*) Drevo aplikacijskih objektov.
API	(*Application Programming Interface*) Aplikacijski programski vmesnik.
CIL	(*Common Intermediate Language*) Skupni vmesni jezik.
CRM	(*Customer Relationship Management*) Sistem za upravljanje odnosov s strankami.
ERP	(*Enterprise Resource Planning*) Integriran poslovni informacijski sistem.
HTTP	(*Hypertext Transfer Protocol*) Protokol za prenos hiperteksta.
HTTPS	(*Secure Hypertext Transfer Protocol*) Varni HTTP, razširja HTTP z zaščito prenosa podatkov.
IIS	(*Internet Information Services*) Spletni strežnik okolja Windows.
OLAP	(*On-Line Analytical Processing*) Sprotna analitična obdelava podatkov.
RPC	(*Remote Procedure Call*) Princip klica oddaljene procedure.
SQL	(*Structured Query Language*) Standardni programski jezik za delo s podatkovnimi bazami.
SSAS	(*SQL Server Analysis Services*) Storitev strežnika Microsoft SQL Server za analitiko.
SSRS	(*SQL Server Reporting Services*) Storitev strežnika Microsoft SQL Server za poročanje.
SUPB	Sistem za upravljanje podatkovne baze.
WCF	(*Windows Communication Foundation*) Ogrodje za razvoj storitveno orientiranih aplikacij.
WWF	(*Windows Workflow Foundation*) Ogrodje za razvoj delovnih tokov.

Povzetek

Uporabniki sistema ERP Microsoft Dynamics AX za izvajanje opravil, ki zahtevajo obdelavo večje količine podatkov, uporabljajo paketne obdelave. Takšne paketne obdelave lahko povzročajo večje obremenitve virov samega sistema. Sočasno izvajanje več zahtevnih paketnih obdelav pa lahko povzroči preobremenitev enega ali več virov sistema, kar se odrazi kot slaba odzivnost sistema pri interaktivni uporabi.

Magistrska naloga kot rešitev predstavi koncept obvladovanja preobremenitev virov, ki z uporabo kontrole prevzema preprečuje sočasno izvajanje prevelikega števila zahtevnih opravil. Ker se pri tem izvajanje zahtevnih opravil časovno porazdeli, se porazdeli tudi obremenitev virov.

V uvodnem delu je naprej predstavljen sistem Microsoft Dynamics AX in uporaba paketnih obdelav. Temu nato sledi analiza virov sistema in bremena paketnih obdelav na testnem okolju. Pri analizi virov so poleg virov operacijskega sistema predstavljeni tudi podatkovni viri, ki jih opredelimo kot skupine podatkovnih tabel. V nadaljevanju je nato na podlagi rezultatov analize bremena opredeljeno umetno breme, ki je potrebno za preverjanje ustreznosti predlagane rešitve.

V osrednjem delu je predstavljeno delovanje algoritma za razporejanje paketnih obdelav v sistemu AX. Le-ta je nato razširjen s kontrolo prevzema, ki na podlagi podatka o tipu opravila v povezavi s podatki o obremenitvi virov odloči, ali bo opravilo prevzeto v izvajanje ali ne. Ker algoritem za to potrebuje podatke o obremenitvi virov, je zasnovan tudi monitoring, ki za izločanje odvečnih meritev uporablja dinamičen prag.

V zadnjem delu magistrske naloge so najprej predstavljeni rezultati preizkusa rešitve ob uporabi umetnega bremena na testnem okolju. Pri tem je najprej z uporabo bremena, sestavljenega iz generičnih paketnih obdelav, prikazana pravilnost delovanja rešitve na izbranem viru. Temu sledi preverjanje splošne učinkovitosti rešitve na večjem naboru virov z uporabo umetnega bremena, sestavljenega iz realnih paketnih obdelav. Na podlagi pridobljenih rezultatov je na koncu ovrednotena rešitev. Pri tem je kot kriterij učinkovitosti uporabljen čas, v katerem je obremenitev posameznega vira višja od podane referenčne meje.

Ključne besede: obvladovanje preobremenitev, kontrola prevzema, obremenitev vira, paketne obdelave, monitoring, Microsoft Dynamics AX

Abstract

Users of Microsoft Dynamics AX ERP leverage batch jobs for processing large amounts of data. This kind of batch job processing is expected to cause high resource load on the system. Parallel execution of more than one such resource-intensive batch job can lead to overload of system resources. This results in poor interactive user experience due to high response times.

This master's thesis introduces resource overload control solution that reduces number of parallel resource-intensive jobs by performing admission control. Because admission control distributes execution times of resource-intensive jobs over time, resource consumption also gets distributed.

The introduction part of the thesis begins with a brief overview of Microsoft Dynamics AX system and batch job use. That is followed by analysis of system resources and batch job workload on the test environment. In analysis resources are represented as operating system resources and data resources that are defined as groups of database tables. In order to evaluate the proposed solution an artificial workload is needed, which is defined based on the results of workload analysis.

The central part of the thesis presents how the standard algorithm for scheduling of batch jobs in AX works. The scheduling algorithm is then extended with admission control that uses information about tasks in combination with resource consumption in order to decide whether to select task for execution or not. Admission control needs information about state of resources, thus a monitoring mechanism is introduced. This monitoring implements a dynamic threshold algorithm to eliminate insignificant resource state changes.

Final part of the thesis begins with the presentation of results obtained by using the solution on the test environment. First, the results are presented, which were obtained by applying the workload of generic batch jobs that consume a specific resource. This is used to demonstrate that the solution is working according to the design. That is followed by the presentation of the results that were obtained by applying workload of real batch jobs that consume various system resources. With this it is presented how well the solution performs when handling overloads in general. At the end the solution is evaluated by summarizing the results. In the process, the time is used as a criteria of evaluation, in which the resource consumption is above the given reference point.

Keywords: overload control, admission control, resource load, batch job, monitoring, Microsoft Dynamics AX

1 Uvod

Danes se uporablja programska oprema za obdelavo podatkov v praktično vseh poslovnih procesih. Eden od načinov, kako uporabnik izvede obdelavo podatkov, je tudi uporaba paketnih obdelav. Pri tem program izvede enega ali več opravil brez potrebe po uporabniški interakciji. Paketne obdelave se uporabljajo za računalniško obdelavo podatkov že od samih začetkov pojavitve »*mainframe*«[1] sistemov v 50. letih [18, 23]. V tistih časih so bili takšnemu pristopu obdelave podatkov v prid številni razlogi, kot so visoka cena računalniških sistemov, nezmožnost izvajanja več programov hkrati, odsotnost interaktivnih uporabniških vmesnikov, primarno so bili poslovno kritični računovodski problemi itn. Danes pa je uporaba paketnih obdelav prisotna na različnih področjih. Zasledimo jih v operacijskih sistemih, antivirusnih programih, sistemih za elektronsko pošto idr.

Z leti se potreba po uporabi paketne obdelave podatkov ni zmanjšala, temveč se povečuje. Namreč, pojavljajo se tako nove poslovne potrebe (npr. izračuni napovedi), kot se tudi večajo količine podatkov, ki jih je potrebno obdelati. Pri modernih sistemih je izvajanje paketnih obdelav navadno podprto z modulom oz. ogrodjem, ki omogoča robustno odzivanje na napake in skalabilnost sistema za obdelavo velike količine podatkov. Slednje tudi predstavlja zahtevo po uporabi vzporednega procesiranja in večje obremenitve virov sistema, kot so diskovna polja, omrežne povezave, procesorska moč, fizični pomnilnik idr.

Prednosti, ki jih prinaša uporaba paketnih obdelav, so naslednje:

- Preložitev izvajanja opravila na kasneje (npr. 1 uro pred prihodom na delovno mesto).
- Periodično izvajanje opravila (npr.: vsak dan, vsak ponedeljek, vsak prvi dan v mesecu).
- Preložitev izvajanja opravila na čas, ko je sistem manj obremenjen (npr. ponoči). V organizacijah navadno obstajajo časovna okna (angl. *batch window*), ko je sistem manj obremenjen.
- Delegiranje izvajanja časovno zahtevnega opravila na namenski strežnik. Uporabnik prestavi izvajanje opravila iz svoje delovne postaje na strežnik in s tem sprosti svoje delovno okolje za ostala interaktivna opravila.
- Samodejno izvajanje opravila pod določenimi pogoji (npr. upoštevanje precedence) oz. dogodki (npr. prispelo sporočilo v vmesnik za integracijo).
- Vzporedno izvajanje več instanc opravila hkrati (npr. vzporedno knjiženje več računov hkrati).
- Omogoča uporabo različnih prioritet pri razporejanju opravil.
- Z dobro razporeditvijo opravil je možno zagotoviti dobro izkoriščenost sistemskih virov.

Zaradi navedenih prednosti oz. razlogov uporabljajo paketne obdelave tudi uporabniki integriranega informacijskega poslovnega informacijskega sistema (ERP – *Enterprise Resource Planning*) Microsoft Dynamics AX (AX) [25]. Arhitektura le-tega je zasnovana trinivojsko, tako da na prvem nivoju aplikacija klienta skrbi za prezentacijo, na srednjem nivoju aplikacijski strežnik skrbi za izvajanje poslovne logike in na tretjem nivoju podatkovni strežnik skrbi za

[1] Centralni računalnik oz. osrednji računalnik (angl. *mainframe*) je velik, zmogljiv računalnik, ki omogoča sočasno delo več uporabnikov in na katerega se lahko priključi več zunanjih naprav, terminalov, računalnikov.

shranjevanje podatkov [10]. Tako lahko uporabnik sistem uporablja bodisi interaktivno preko funkcij klienta bodisi neinteraktivno z razporeditvijo opravila v obliki paketne obdelave na aplikacijski strežnik.

Kadar se paketne obdelave uporabljajo za obdelavo večje količine podatkov, so viri aplikacijskega in podatkovnega strežnika navadno izpostavljeni večjim obremenitvam. Sočasno izvajanje več takšnih zahtevnih paketnih obdelav pa lahko vodi v preobremenitve virov sistema AX. Posledica le-tega je slaba odzivnost sistema, ki jo občutijo uporabniki pri interaktivni uporabi sistema preko aplikacij klienta.

Sistem je navadno dimenzioniran, tako da ima nekaj proste kapacitete, s katero lahko zadosti občasnim povečanim obremenitvam. Pogosto se za zagotavljanje zadostne kapacitete uporablja tudi podvajanje posameznih komponent sistema v povezavi z možnostjo izenačevanja obremenitev (angl. *load balancing*). Kljub temu se ni mogoče izogniti vsem situacijam preobremenitve sistema. Ker sistem podpira razporeditev novih paketnih obdelav na zahtevo, nikoli ne vemo, kdaj bo prišlo do izvajanja opravila. V različnih situacijah se obremenitve lahko razlikujejo od predvidenih, ker se obdelujejo različne količine podatkov. Prav tako so lahko viri sistema, kot so procesor, pomnilnik, diskovno polje idr., v določenem trenutku obremenjeni zaradi procesiranj, ki niso posledica paketnih obdelav. Le-to je lahko povečana aktivnost klientov sistema AX, izvajanje druge programske opreme na isti strojni opremi, izvajanje vzdrževalnih del idr. Do preobremenitev lahko pride tudi zaradi zmanjšanja kapacitete ob izpadu posamezne komponente, ki je del gruče za izenačevanje obremenitev.

Sistem, ki bi zadostil dovoljšno kapaciteto v vseh najslabših izidih, bi bil predrag in večino časa slabo izkoriščen [12]. Zato je bolj smiselno zasnovati sistem za normalne visoke obremenitve in občasno prerazporediti izvajanje zahtevnih paketnih obdelav, da bi se izognili preobremenitvi. To pomeni, da bi v algoritem za razporejanje paketnih obdelav vpeljali koncept obvladovanja preobremenitev virov. Le-to je tudi edini ustrezni pristop, ker vseh komponent sistema in s tem tudi virov sistema ni mogoče podvajati. Podvajanje je lahko tudi nezaželeno pri postavitvah, ki so namenoma majhne (npr. z enim strežnikom). Iz tega sledi, da pri reševanju problema preobremenitev virov koncepta obvladovanja preobremenitev ni mogoče nadomestiti s konceptom izenačevanjem obremenitev [4].

V nadaljevanju magistrskega dela je najprej predstavljen ERP sistem AX. Pri tem je pozornost posvečena arhitekturi sistema, možnostim za razvoj razširitev in uporabi paketnih obdelav.

V poglavju 3 so predstavljeni rezultati analize virov sistema in bremena paketnih obdelav na testnem okolju. Temu sledi poglavje, ki opredeli umetno breme, ki je namenjeno preverjanju ustreznosti rešitve za obvladovanje preobremenitve virov.

V poglavju 5 je pozornost posvečena izdelavi rešitve. Pri sami izdelavi rešitve je najprej predstavljeno delovanje modula za razporejanje paketnih obdelav sistema AX. S tem opredelimo sam algoritem razporejanja paketnih obdelav in možnosti za razširitev le-tega. Sama zasnova rešitve je razdeljena na tri dele. V prvem delu je vzpostavljeno spremljanje obremenjenosti virov oz. monitoring, katerega izvajanje mora v čim manjši meri dodatno obremenjevati sistem. V drugem delu je predlagan in vzpostavljen koncept obvladovanja preobremenitve virov. V zadnjem delu vgradimo rešitev v obstoječo programsko kodo.

V poglavju 6 je prikazano delovanje rešitve ob uporabi umetnega bremena na testnem okolju. Pri tem je na podlagi rezultatov meritev za posamezne vire sistema izpostavljeno, kako dobro lahko rešitev obvladuje njegove obremenitve. Temu nato sledi poglavje, ki povzame rezultate in ovrednoti učinkovitost rešitve.

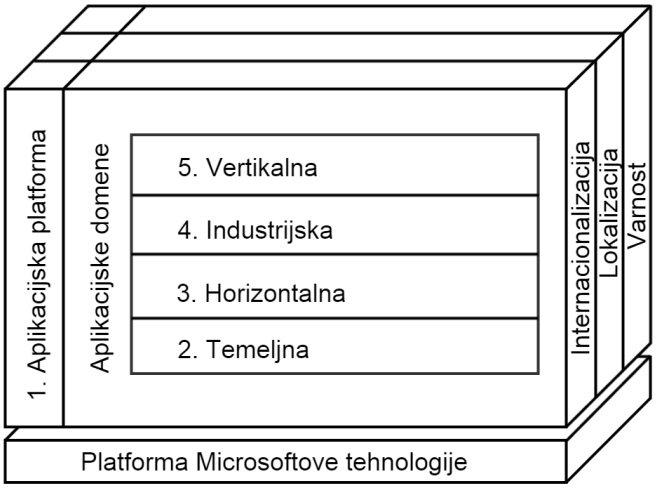
2 Predstavitev sistema Microsoft Dynamics AX

Microsoft Dynamics AX je ERP rešitev, ki s svojimi moduli integrira sklope za upravljanje finančnih, operativnih in človeških virov [19]. Rešitev je prilagojena za obvladovanje multinacionalnih podjetij, skupin podjetij in podjetij, ki pokrivajo različna industrijska področja [10]. Celotno rešitev lahko v principu razdelimo na aplikacijo in aplikacijsko platformo. Namen aplikacijske platforme je zagotoviti ogrodje, ki omogoča razvoj skalabilnih, prilagodljivih in razširljivih ERP aplikacij. Ključni arhitekturni principi, ki to omogočajo, so trinivojska arhitektura, sloji aplikacijske kode in modelno vodeni razvoj.

Logično[2] lahko rešitev razdelimo v pet slojev (slika 1) [10]:

1. Sloj aplikacijske platforme nudi ogrodja in orodja za razvoj, kot so modelno vodeni razvoj MorphX, programski jezik X++, klient, aplikacijski strežnik idr.
2. Sloj temeljne aplikacijske domene nudi referenčne modele (npr.: fiskalni koledar, merske enote, jezikovne kode idr.), modeliranje virov (npr.: model partnerja, model organizacije, produktni model idr.), modeliranje politik, ogrodja za beleženje dogodkov in ogrodja za obdelavo dokumentov.
3. Sloj horizontalne aplikacijske domene nudi podporo procesom nad viri, ki jih podjetje obvladuje (npr.: računovodstvo, knjigovodstvo, kadrovanje, odnosi s strankami idr.).
4. Sloj industrijske aplikacijske domene nudi podporo procesom nad viri, ki so specifični za določena industrijska področja (npr.: proizvodnja, distribucija, javna poraba idr.).
5. Sloj vertikalne aplikacijske domene nudi podporo procesom, ki so specializirani za določeno panogo (npr. proizvodnja avtomobilov, proizvodnja pijač, prodaja električne energije idr.) ali so predmet posebnih lokalnih predpisov.

Komponente vseh slojev ustrezajo standardom, ki zagotavljajo varnost, lokalizacijo in internacionalizacijo rešitve, in so zgrajene z uporabo Microsoftovih tehnologij.

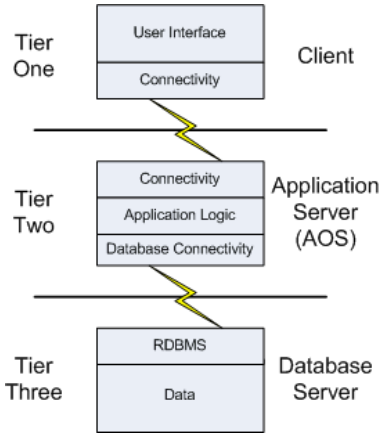


Slika 1: Logična delitev poslovne rešitve AX na pet slojev [10]

[2] Navaja logično delitev celotne rešitve, ki ni neposredno povezana s sloji aplikacijske kode, ki bodo predstavljeni v razdelku 2.2.

2.1 Arhitektura sistema

Arhitektura sistema AX je zasnovana trinivojsko (slika 2). Prvi nivo skrbi za prezentacijo, drugi oz. srednji nivo za izvajanje poslovne logike in tretji nivo za shranjevanje podatkov [10]. Prednost takšnega pristopa je boljša razpoložljivost in skalabilnost (angl. *scale-out*), saj imamo lahko hkrati aktivnih več aplikacijskih in podatkovnih strežnikov. Ker se vsa logika izvaja preko srednjega nivoja, se izboljša varnost sistema in olajša redistribucija sprememb v aplikacijski kodi. Slabost takšnega pristopa je večje število komunikacijskih prehodov, ki lahko pridejo do izraza pri slabo zasnovanih rešitvah.



Slika 2: Trinivojska arhitektura [16]

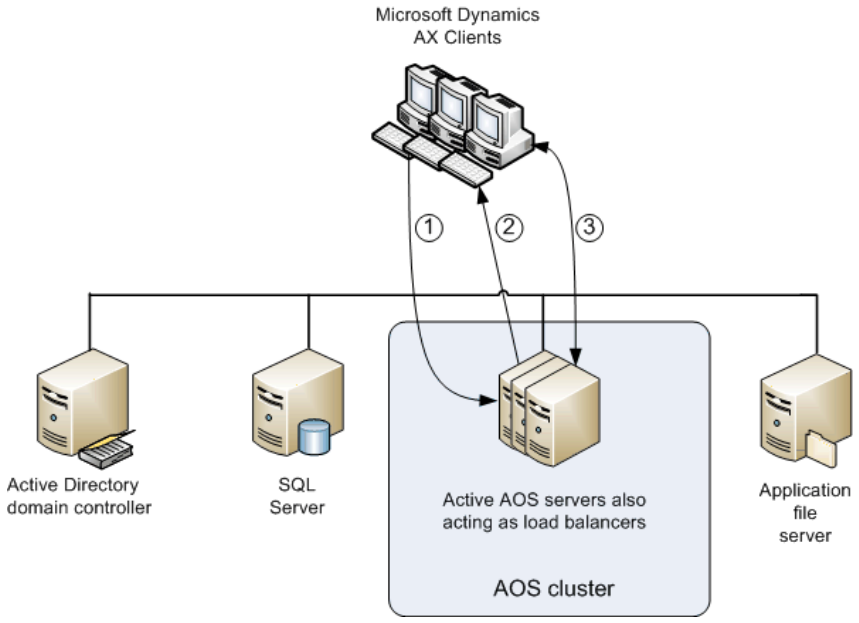
Prvi nivo oz. nivo prezentacije predstavljajo naslednje komponente:

- **Windows klient** je namizna aplikacija za okolje Windows. Aplikacija komunicira z aplikacijskim strežnikom z uporabo tehnologije RPC (*Remote Procedure Call*). Namen klienta je prezentacija in navigacija z uporabo grafičnih gradnikov, ki omogočajo vnos in pridobivanje podatkov. Grafični vmesnik uporablja InteliMorph tehnologijo, ki omogoča kreiranje uporabniškega vmesnika na podlagi meta podatkov in individualno prilagodljivost le-tega posameznemu uporabniku.
- **Enterprise portal** je spletni vmesnik poslovnega portala. Portal za izvajanje uporablja platformo za spletne aplikacije SharePoint [26].
- **Office klienti** predstavljajo uporabo aplikacij Word in Excel iz pisarniškega paketa Microsoft Office. Word in Excel se lahko z uporabo vtičnikov (angl. *plug-in*) povežeta z AX platformo in iz nje črpata podatke.
- **Druge aplikacije** se lahko povežejo z AX platformo z uporabo integracijskih storitev in .NET Business Connector komponente. Le-to je komponenta, ki omogoča eksternim aplikacijam interakcijo z aplikacijskim strežnikom.

Drugi nivo oz. srednji nivo predstavljajo naslednje komponente:

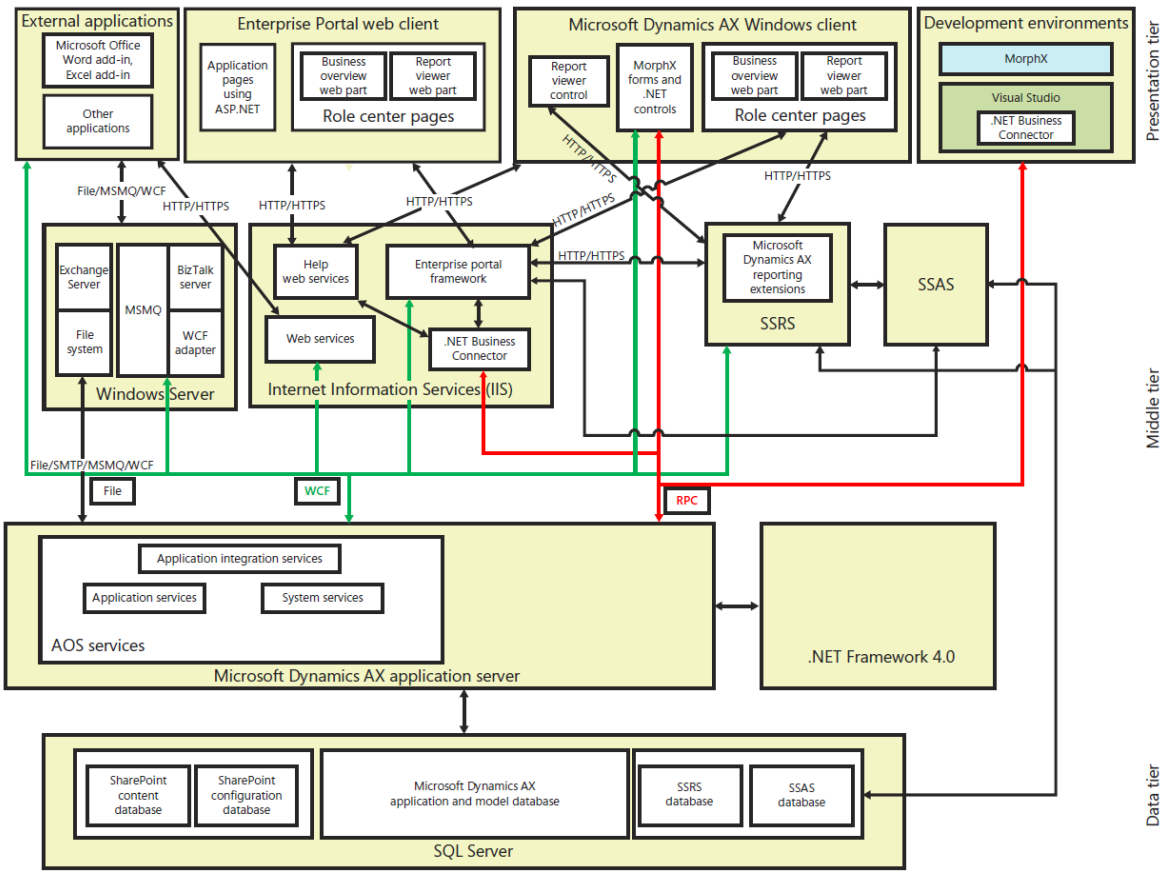
- **AOS (*Application Object Server*)** je aplikacijski strežnik, katerega naloga je izvajanje aplikacijskih storitev, ki se prožijo z uporabo tehnologij RPC in WCF (*Windows Communication Foundation*). AOS je aplikacija, ki teče v obliki storitve na Windows Server okolju. Nameščeno moramo imeti vsaj eno instanco, vendar lahko namestimo

več instanc za potrebe horizontalne skalabilnosti sistema. Instance lahko pri tem nastopajo bodisi v obliki gruč, ki so namenjene streženju hkratnih sej uporabnikov z izenačevanjem obremenitve (slika 3), bodisi v obliki namenskih strežnikov za izvajanje paketnih obdelav. Ker sistem uporablja integrirano Windows avtentikacijo za vse strežnike, je potrebna uporaba Microsoftovega aktivnega imenika (*Active Directory*).



Slika 3: AOS gruča z izenačevanjem obremenitve [28]

- **Ogrodje .NET (*.NET Framework*)** nudi komponente, ki so na voljo pri razvoju aplikacij v osnovnem programskem jeziku X++. Tako AX v ogrodju za delovne tokove uporablja komponento WWF (*Windows Workflow Foundation*) in v ogrodju za integracije uporablja komponento WCF.
- **SQL Server Analysis Services (SSAS)** nudi storitve za procesiranje zahtev pri dostopu do analitičnih podatkov.
- **SQL Server Reporting Service (SSRS)** nudi storitve za poročanje nad transakcijskimi in analitičnimi podatki. SSRS s pomočjo dodatka komunicira z AOS-om preko WCF in dostopa do SSAS z uporabo protokolov HTTP in HTTPS.
- **Ogrodje Enterprise portala** uporablja platformo za spletne aplikacije SharePoint. Pri tem se za prezentacijo uporablja komponente SharePoint in ASP.NET. Komunikacija z aplikacijskim strežnikom pa poteka z uporabo WCF in komponente .NET Business Connector in WCF.
- **Storitev pomoči** nudi uporabnikom dostop do vsebin z uporabniškimi navodili. Storitev je objavljena z uporabo spletnega strežnika IIS (*Internet Information Services*).
- **Spletne storitve dostopne preko strežnika IIS**. Storitve sistema AX lahko objavimo preko strežnika IIS.
- **Ogrodje za aplikacijske storitve (AIF – *Application Integration Framework*)** nudi ogrodje za storitve, ki omogočajo integracijo z drugimi aplikacijami.



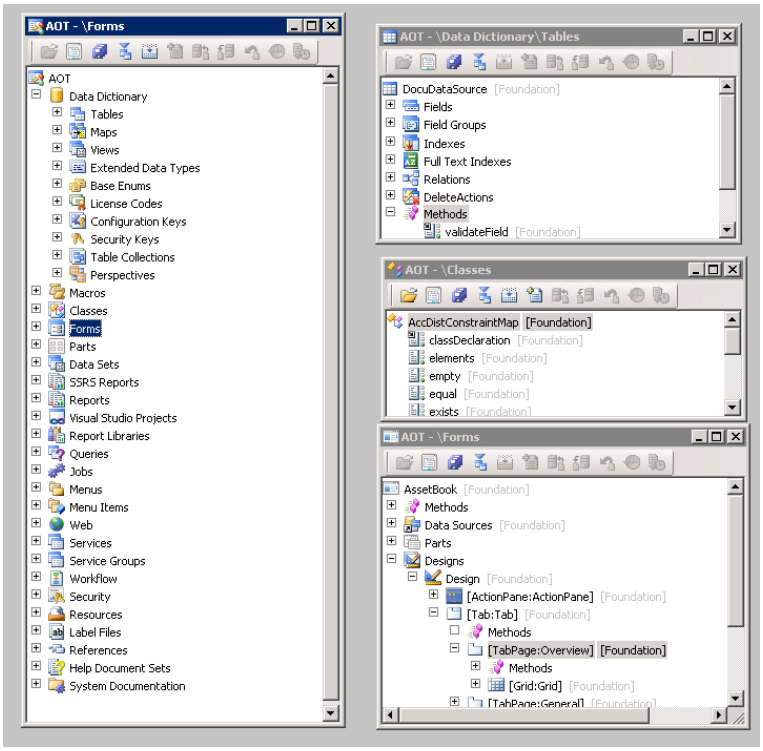
Slika 4: Trinivojska arhitektura sistema AX – podrobno [10]

Tretji nivo oz. podatkovni nivo vsebuje le komponento podatkovnega strežnika Microsoft SQL Server, ki je SUPB (Sistem za upravljanje podatkovne baze) za naslednje podatkovne baze:

- AX podatkovna baza, ki vsebuje definicije aplikacijskih objektov;
- AX podatkovna baza, ki vsebuje poslovne podatke;
- SharePoint podatkovna baza, ki vsebuje vsebino in nastavitve za poslovni portal;
- SSRS podatkovna baza, ki vsebuje vsebino nastavitve za bazo poročil;
- SSAS podatkovna baza, ki vsebuje nastavitve in transformirane poslovne podatke v obliki kock OLAP (*On-Line Analytical Processing*).

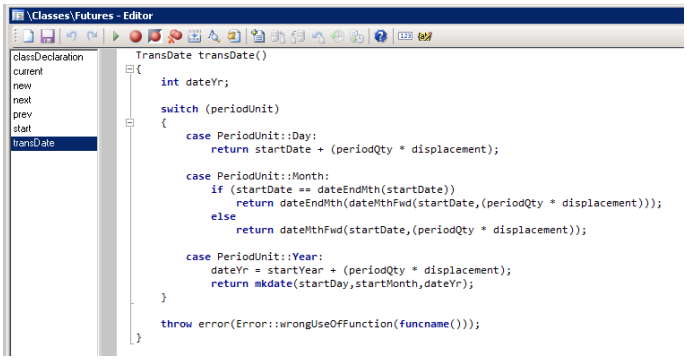
2.2 Razvoj prilagoditev in razširitev

Za razvoj novih in prilagajanje obstoječih funkcionalnosti sta na voljo dve orodji. Prvo je razvojno okolje MorphX, ki je del aplikacije klienta. Le-to nam omogoča razvoj podatkovnega modela in aplikacijske kode z uporabo drevesa aplikacijskih objektov (AOT – *Application Object Tree*) in programskega jezika X++. Drugo razvojno okolje je Microsoft Visual Studio, ki se uporablja za razvoj programskih knjižnic v ogrodju .NET za X++, poslovnega portala in poročil SSRS. Namen tega razdelka ja prikazati zgolj koncepte, ki omogočajo enostavno prilagajanje sistema AX. Sam razvoj je podrobneje predstavljen v [7, 8, 10].



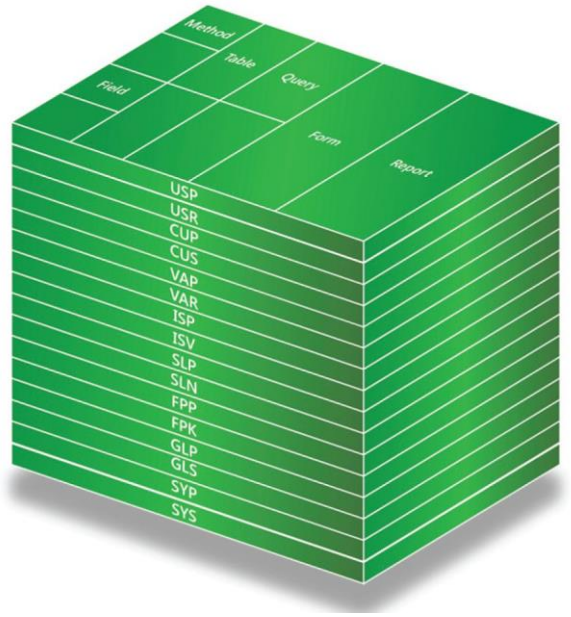
Slika 5: AOT – drevo aplikacijskih objektov (tabela, razred in zaslonska maska)

Program, ki ga okolje AX izvaja, je definiran v obliki elementov, ki so del drevesa AOT (slika 5). Pri tem je definicija posameznega elementa ravno tako drevo. Tako so listi definicije tabele polja, listi definicije razreda metode in listi definicije zaslonske maske (angl. *form*) posamezni grafični gradniki.



Slika 6: Urejevalnik kode X++

Za razliko od ostalih programskih okolji AX lahko hrani po več definicij za vsak element (npr. več implementacij iste metode razreda), pri čemer se vsaka nahaja na svojem sloju (angl. *layer*). Izvajalnik okolja AX pri tem vedno izvede definicijo, ki se nahaja na najvišjem sloju. Slojni pristop omogoča prilagoditev izvornega programa, ki ga razvija Microsoft, Microsoftovi partnerji in neodvisni ponudniki, brez spreminjanja originalne izvorne kode. Tako v primeru, ko želimo dodati dodaten ukaz v določeni standardni metodi, enostavno le prekrijemo definicijo metode z lastno definicijo na višjem sloju. V prvotno stanje element povrnemo z enostavnim izbrisom definicije na višjem sloju. V vsakem trenutku lahko primerjamo definicijo iz trenutnega sloja z definicijami na drugih slojih. V primeru nadgradnje nižjega sloja (npr. ob izidu nove verzije AX-a) je tako naša naloga, da z uporabo orodja za primerjavo definicije objektov propagiramo potrebne spremembe v svoj višji sloj.



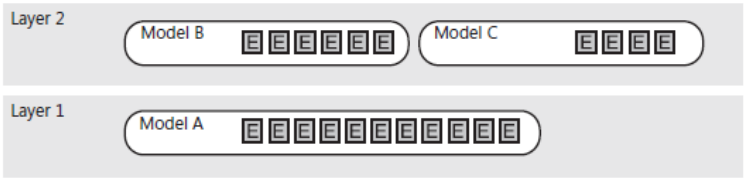
Slika 7: 16 slojev aplikacijske kode [7]

AX vsebuje 16 slojev aplikacijske kode (slika 7), od katerih vsak služi svojemu namenu. Sloji nastopajo v parih, ker vsakemu osnovnemu sloju pripada še t. i. »*patch layer*«, ki je namenjen popravkom. Seznam osnovnih slojev je naslednji:

- **SYS (*System layer*).** Najnižji sistemski sloj, ki vsebuje osnovno aplikacijsko platformo, ki jo razvija Microsoft.
- **GLS (*Global solution layer*)**. Sloj za globalizacijo, ki vsebuje razširitve in prilagoditve za posamezne regije, ki jih razvija Microsoft.
- **FPK (*Feature pack layer*)**. Sloj vsebuje horizontalne rešitve za podporo industriji, ki jih razvija Microsoft.
- **SLN (*Solution layer*)**. Sloj vsebuje vertikalne rešitve, ki jih razvijajo Microsoftovi partnerji.
- **ISV (*Independent Software Vendor layer*)**. Sloj lahko vsebuje rešitve, ki jih razvijajo neodvisni ponudniki.
- **VAR (*Value-added retailer layer*).** Sloj lahko vsebuje prilagoditve in razširitve, ki so s strani ponudnikov namenjene več strankam.

- **CUS (*Customer layer*).** Sloj je namenjen razvoju prilagoditev in razširitev, ki so specifične za posamezno stranko.
- **USR (*User layer*).** Sloj je namenjen manjšim prilagoditvam, ki jih napravijo napredni uporabniki (npr. prilagoditev zaslonske maske ali sprememba varnostne vloge). To je najvišji osnovni sloj (nad njim se nahaja le še sloj za popravke USP).

Z AX 2012 je bil predstavljen tudi koncept modela, ki logično združuje elemente enega sloja. Element na enem sloju lahko pripada le enemu modelu (slika 8). Tako lahko vse elemente, ki tvorijo določeno programsko rešitev, združimo v svoj model. Le-tega lahko nato izvozimo in uporabimo za distribucijo programske rešitve.



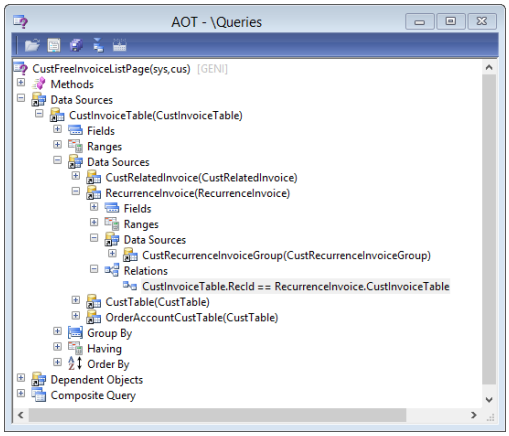
Slika 8: Združevanje elementov po slojih v modele [10]

AX 2012 razširi prilagajanje obstoječe kode X++ s konceptom dogodkov na metodah. Sedaj lahko na vsaki metodi razreda ali tabele registriramo svojo metodo kot dogodek, ki se izvede pred vstopom v metodo (manipulacija parametrov metode) ali po vrnitvi rezultata (manipulacija rezultata metode). Ker pri tem ne prekrivamo spodnjega sloja, se s takšnim načinom prilagoditve izognemo potrebi po razreševanju konfliktov ob nadgradnji spodnjega sloja.

```
select <Find Options>
<TableBuffer or {FieldList from [TableBuffer]}>
<Aggregate (field identifier)>
<Sorting Options field identifier[Direction]>
<Index clause (index)>
where <selection criteria>
<Join Clause join>
```

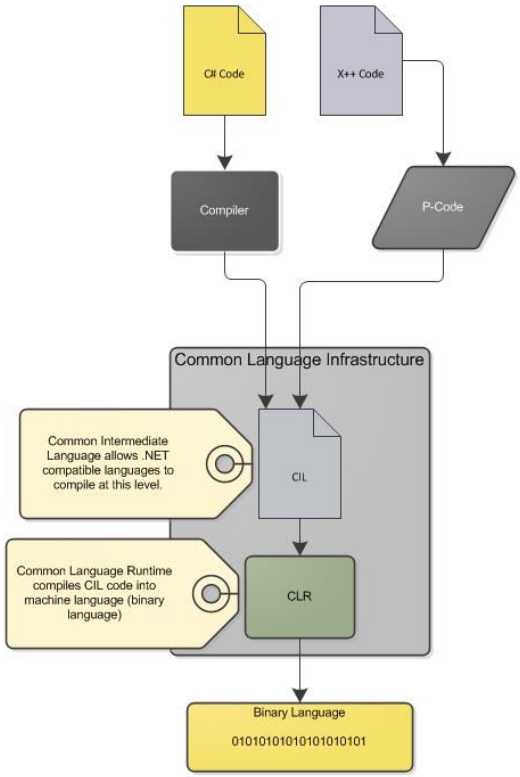
Slika 9: Sintaksa poizvedbe po podatkih z X++ [8]

Programski jezik X++ je objektni jezik, ki je aplikacijsko in podatkovno integriran [10]. Jezik omogoča izbiro, kje se bo koda izvajala glede na nivo aplikacije (klient ali strežnik) in v kontekstu katerega podjetja se bodo manipulirali podatki. Manipulacija podatkov je podprta z uporabo elementov podatkovnih tabel. Pri tem se za dostop do podatkov uporabljajo bodisi SQL-u (*Structured Query Language*) podobni izrazi (slika 9), ki so del jezika X++, bodisi elementi poizvedbe »*Query*« (slika 10).



Slika 10: Element poizvedbe »Query« v AOT

Koda, napisana v programskem jeziku X++, se po prevajanju prevede v vmesno kodo, imenovano »p-code«, ki jo izvaja interpreter. Z AX 2012 je omogočeno izvajanje X++ kode v .NET skupnem vmesnem jeziku CIL (*Common Intermediate Language*) (slika 11). Izvajanje kode v CIL prinaša boljše zmogljivosti in se privzeto uporablja pri izvajanju paketnih obdelav in storitev. Razlogi za boljšo zmogljivost izvajanja kode v CIL napram interpretirani kodi X++ izhajajo predvsem iz bolj učinkovitega sproščanja pomnilnika (angl. *garbage collection*) in manjše potratnosti virov zaradi koncepta uporabe sočasnih sej (angl. *session pooling*) [10]. V kolikor programer izrecno ne uporabi direktive za izvajanje v CIL, se v ostalih primerih koda še vedno izvaja preko interpreterja.



Slika 11: Prevajanje kode X++ v CIL [17]

2.3 Ogrodje za paketne obdelave

Ogrodje za paketne obdelave omogoča uporabnikom asinhrono, zaporedno in paralelno izvajanje opravil na eni ali več instancah strežnikov AOS. AX 2012 izvaja kodo paketnih obdelav zaradi boljših zmogljivosti v .NET CIL-u na nivoju AOS. Za delo s paketnimi obdelavami sta na voljo tako uporabniški vmesnik kot programski vmesnik (API – *Application Programming Interface*).

Pri delu s paketnimi obdelavami v AX-u se srečamo z naslednjimi pojmi [10]:

- **Paketno opravilo (angl. *Batch task*).** Najmanjša enota dela, ki je lahko predmet izvajanja. Realizirano je v obliki paketno-izvršljivega razreda (angl. *batch-executable*), ki implementira poslovno logiko opravila.
- **Paketna obdelava (angl. *Batch job*).** Skupek opravil, ki opravi določeno nalogo (npr.: tiskanje poročila, knjiženje računa, proces zapiranje zaloge idr.). Paketna obdelava se lahko sestoji iz enega ali več paketnih opravil.
- **Paketna skupina (angl. *Batch group*).** Logična kategorizacija paketnih opravil, s katero lahko omejimo, na kateri instanci AOS se opravilo lahko izvaja. Paketna opravila, ki niso dodeljena v nobeno skupino, so razporejena v privzeto »prazno« skupino.
- **Paketni strežnik (angl. *Batch server*).** Instanca AOS, ki je označena, da lahko izvaja paketne obdelave.

Pri delu s paketnimi obdelavami se srečamo z naslednjimi postopki:

1. Kreiranje paketno-izvršljivega razreda (angl. *batch-executable*).
2. Kreiranje paketne obdelave.
3. Konfiguriranje paketnih strežnikov in paketnih skupin.
4. Obvladovanje paketnih obdelav.

2.3.1 Kreiranje paketno-izvršljivega razreda

Razred napisan v X++ mora implementirati vmesnik (angl. *interface*) `Batchable`, v kolikor ga želimo izvajati kot paketno opravilo. V praksi pa se najbolj pogosto uporablja razširitev (angl. *extends*) abstraktnega razreda `RunBaseBatch`, ki zagotavlja potrebno ogrodje (uporabniški vmesnik, validacijo parametrov, priklic zadnjih vrednosti parametrov idr.). Z verzijo AX 2012 je na voljo uporaba novega ogrodja SysOperation, ki ponuja določene prednosti, kot so modelni pristop, vsiljeno izvajanje na nivoju strežnika, vsiljeno izvajanje v CIL, zagotovljena serializacija razreda in različni načini izvajanja (sinhroni, asinhroni, zanesljivo asinhroni in paketna obdelava) [10]. Ker ogrodje SysOperation samo po sebi implementira principe razreda `RunBaseBatch`, bo predstavljena uporaba slednjega.

Pri uporabi razširitve razreda `RunBaseBatch` moramo implementirati naslednje štiri metode:

- `run` – vsebuje kodo, ki realizira logiko paketnega opravila;
- `pack` – vsebuje kodo, ki serializira objekt (stanje objekta se shrani za čas od kreiranja do zagona opravila), tako da vrednosti spremenljivk shrani v seznam;

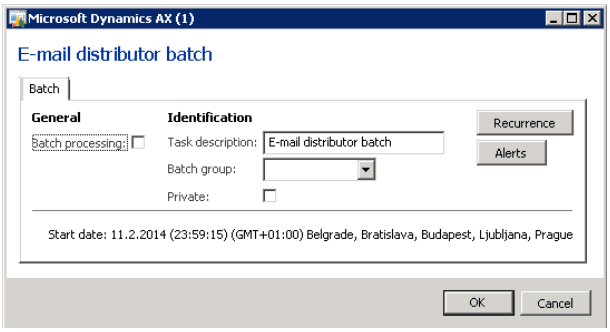
- `unpack` – vsebuje kodo, ki deserializira objekt (stanje objekta se obnovi tik pred začetkom izvajanja), tako da vrednosti iz seznama dodeli spremenljivkam razreda;
- `canGoBatchJournal` – odloči, ali lahko uporabimo razred pri ročnem dodajanju paketnih opravil v paketno obdelavo preko maske za urejanje paketne obdelave.

2.3.2 Kreiranje paketne obdelave

Novo paketno obdelavo lahko kreiramo na tri načine:

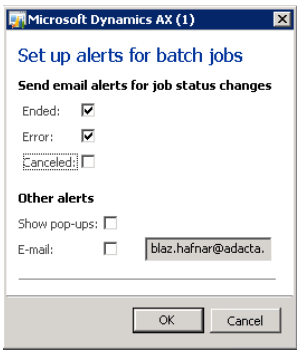
- preko pogovornega okna (angl. *dialog*), ki ga prikaže implementacija paketne obdelave;
- preko zaslonske maske za urejanje paketne obdelave;
- preko uporabe knjižnice API.

Prvi način kreiranja paketne obdelave zahteva klik na gumb, ki aktivira posamezno paketno obdelavo. Prikaže se pogovorno okno (slika 12), ki vsebuje zavihek »*Batch*«, na katerem lahko izberemo možnost »*Batch processing*«. S potrditvijo vnosa kreiramo novo paketno obdelavo, ki se bo izvedla asinhrono na strežniku.

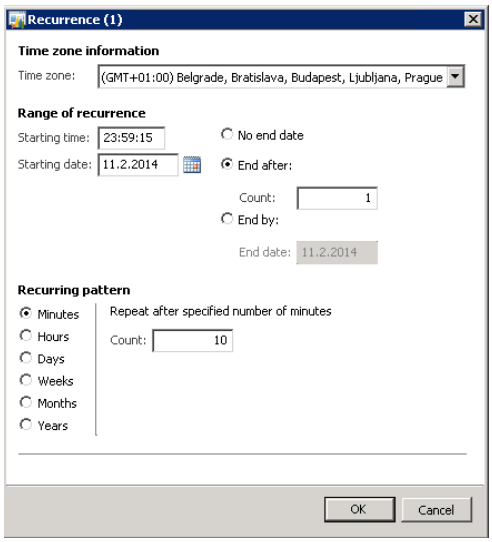


Slika 12: Pogovorno okno paketne obdelave za pošiljanje elektronske pošte

Pred potrditvijo lahko s klikom na gumb »Recurrence« nastavimo ponovljivost obdelave. Pri tem lahko izbiramo med naslednjimi enotami periode: minuta, ura, dan, teden, mesec in leto (slika 14). Pri izbiri leta, meseca in tedna lahko določimo ponovitev do dneva v tednu natančno (npr. februar, drugi teden, sreda). V kolikor želimo biti obveščeni ob določenem tipu zaključka (končano, preklicano in napaka), lahko le-to nastavimo s klikom na gumb »*Alerts*« (slika 13). Z izbiro paketne skupine lahko obdelavo dodelimo v izvajanje na določenem naboru strežnikov.

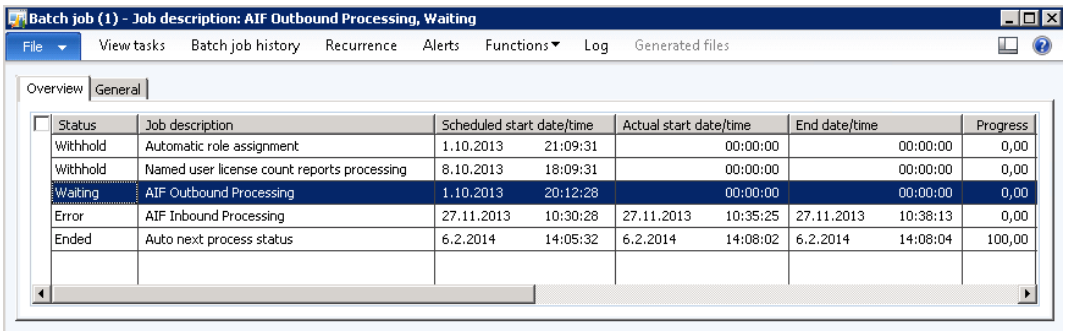


Slika 13: Pogovorno okno za nastavitev obvestil o izvedbi paketne obdelave



Slika 14: Pogovorno okno za nastavitev ponovljivosti paketne obdelave

Drugi način, na katerega lahko kreiramo novo paketno obdelavo, je preko zaslonske maske za pregled paketnih obdelav, ki je prikazana na sliki 15. Na seznamu najprej uporabimo funkcijo za kreiranje novega zapisa, s čimer kreiramo novo paketno obdelavo. Ko imamo zapis obdelave kreiran, lahko s klikom na gumb »*View tasks*« odpremo masko za urejanje paketne obdelave, ki je prikazana na sliki 16.

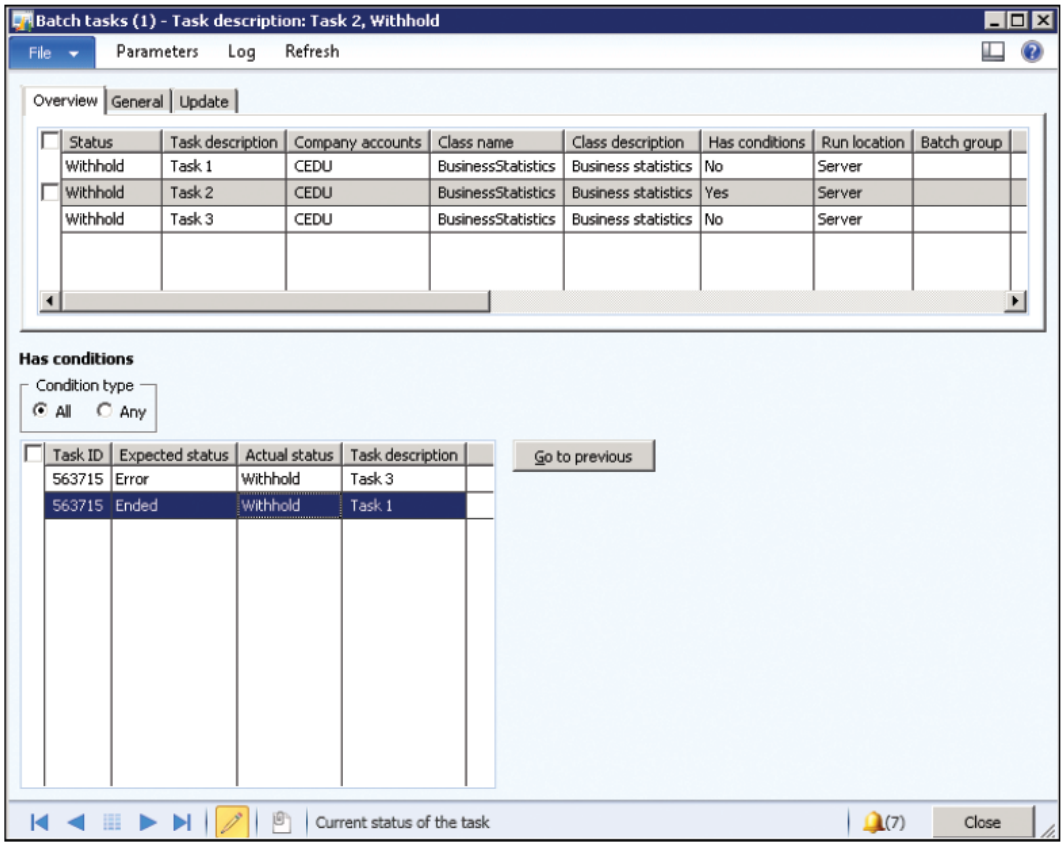


Slika 15: Zaslonska maska za pregled paketnih obdelav

Na maski za urejanje paketne obdelave oz. paketnih opravil lahko dodajamo v nova opravila in nastavljamo odvisnosti med opravili.

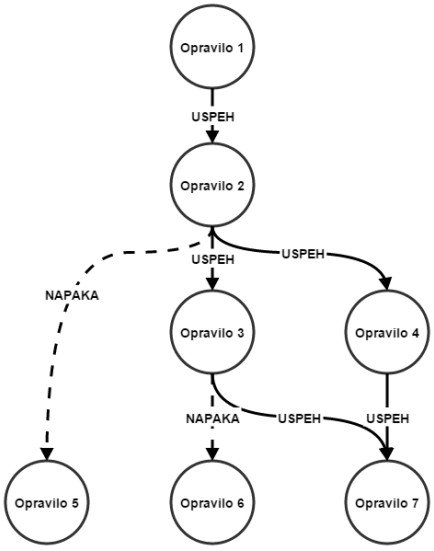
Opravilo dodamo v paketno obdelavo po naslednjem postopku:

1. Ustvarimo nov zapis za paketno opravilo.
2. Vnesemo opis opravila.
3. Izberemo podjetje, v katerega kontekstu bo opravilo teklo.
4. Izberemo razred, katerega kodo bo opravilo izvajalo. Na voljo so samo paketno-izvršljivi razredi, katerih metoda `canGoBatchJournal` vrne vrednost `true`.
5. Nastavimo paketno skupino.
6. Shranimo zapis.
7. Po potrebi s klikom na gumb »*Parameters*« prikličemo pogovorno okno za nastavitev parametrov razreda.
8. Po potrebi nastavimo odvisnosti.



Slika 16: Zaslonska maska za urejanje paketne obdelave

V kolikor imamo v obdelavo vključenih več opravil, lahko za vsako opravilo nastavimo pogoje za začetek na podlagi statusa predhodnega opravila. Pri tem lahko izbiramo med možnostma, da je izpolnjen vsaj en pogoj ali da morajo biti izpolnjeni vsi pogoji. V primeru, ko med dvema opraviloma ni pogojne odvisnosti, se opravili lahko izvajata vzporedno. Na sliki 17 je prikazan primer drevesa odvisnosti opravil, ki ga lahko vzpostavimo.

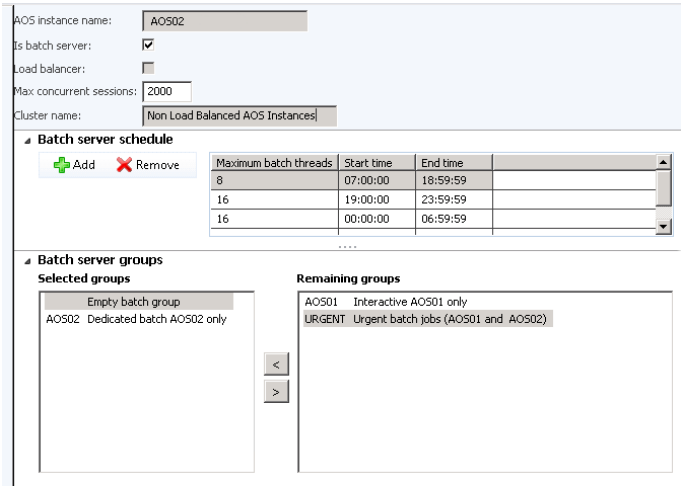


Slika 17: Drevo odvisnosti paketnih opravil

Tretji način kreiranje paketne obdelave je uporaba knjižnice API za paketne obdelave. Knjižnica omogoča uporabo vseh operacij za manipulacijo paketnih obdelav v času izvajanja kode. Navadno se uporablja za dinamično izgradnjo kompleksnih ali velikih paketnih obdelav, kot je na primer postopek nadgradnje podatkovnega modela ob prehodu na novo različico sistema.

2.3.3 Konfiguriranje paketnih strežnikov in paketnih skupin

Predno lahko pričnemo uporabljati paketne obdelave, je potrebno ustrezno konfigurirati vsaj eno instanco AOS. Maska za konfiguracijo instance AOS je prikazana na sliki 18. Z izbiro »*Is batch server*« določimo, da strežnik lahko izvaja paketne obdelave. Koliko niti (angl. *threads*) bo teklo na strežniku za izvajanje obdelav, določimo z urnikom, kjer navedemo število niti v posameznem časovnem intervalu. Na ta način lahko omejimo število hkrati izvajajočih obdelav v dnevnem času. Prav tako lahko nastavimo tudi nabor paketnih skupin. Strežnik bo tako izvajal le paketna opravila iz navedenih paketnih skupin.



Slika 18: Zaslonska maska za konfiguracijo AOS strežnika

2.3.4 Obvladovanje paketnih obdelav

Vse paketne obdelave, ki so v sistemu, lahko vidimo na maski za pregled paketnih obdelav. Pregled predstavlja posnetek trenutnega stanja paketnih obdelav. Tako lahko na podlagi polja status vidimo, katere paketne obdelave so v izvajanju. Paketna obdelava je lahko v enem od naslednjih statusov:

- zadržano (angl. *withhold*): obdelava ustavljena in se ne bo pričela izvajati;
- čakajoč (angl. *waiting*): obdelava čaka, da se bo izvedla po urniku;
- v izvajanju (angl. *executing*): obdelava se izvaja;
- končano (angl. *ended*): obdelava se je izvedla;
- napaka (angl. *error*): pri izvajanju je prišlo do napake;
- v preklicu (angl. *canceling*): izvajanje obdelave je v postopku preklica;
- preklicano (angl. *canceled*): obdelava je bila preklicana.

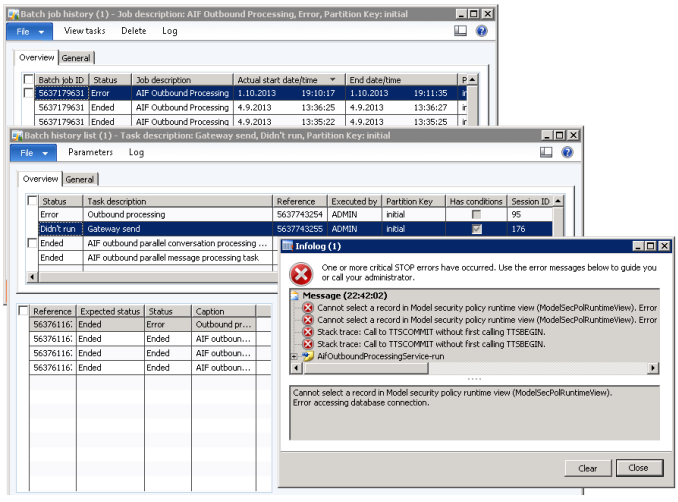
Pri paketnih opravilih je nabor statusov dopolnjen z naslednjima dvema dodatnima statusoma:

- pripravljen (angl. *ready*): opravilo ima izpolnjene vse pogoje za izvajanje;
- neizvedeno (angl. *didn't run*): opravilo se ni izvedlo, ker je prišlo do napake ali preklica.

Status paketne obdelave lahko spreminjamo s funkcijo za spremembo statusa. Pri tem lahko obdelavam v statusu *čakajoč* dodelimo status *zadržano* in s tem preprečimo izvajanje. Obdelavam, ki so v statusu *zaključeno* ali *zadržano*, lahko dodelimo status *čakajoč* in jih vrnemo v ponovno izvajanje. Obdelavam, ki so v izvajanju in jih želimo prekiniti, lahko dodelimo status *v preklicu*. Le-te bodo prešle v status *preklicano*, ko jih sistem uspe prekiniti. Pri tem gredo opravila, ki so del obdelave in so v izvajanju, skozi enak postopek. Opravila, ki niso bila izvedena, pa v status *neizvedeno*. Obdelave, ki imajo nastavljeno ponavljanje, ob zaključku samodejno preidejo nazaj v stanje *čakajoč*.

Za vsako paketno opravilo lahko nastavimo parameter število novih poskusov, ki ima privzeto vrednost 1. V primeru, da pride do izpada ali napake med izvajanjem opravila, se status opravila po vzpostavi normalnega stanja samodejno spremeni nazaj v *pripravljen*. Opravilo preide v napako, v kolikor število novih poskusov preseže nastavljeno vrednost. Na posameznem opravilu lahko tudi nastavimo možnost za ignoriranje napake. Napaka takšnega opravila ne povzroči napake obdelave. Z nastavljanjem prioritete na opravilu določamo, katero opravilo bo imelo prednost pri prevzemu v izvajanje. Vsaka prosta izvajalna nit strežnika vsako minuto skuša prevzeti opravilo v statusu *pripravljen*, ki ima najmanjšo prioriteto in je v paketni skupini strežnika.

Za vsako paketno obdelavo lahko določimo beleženje zgodovine izvajanja. Pri tem lahko izbiramo med možnostmi: vedno, samo napake in nikoli. Do zgodovine lahko dostopamo preko maske za pregled paketnih obdelav, pri čemer se nam prikaže zgolj zgodovina izbrane obdelave, in preko osrednjega menija. Na maski za pregled zgodovine paketnih obdelav (slika 19) lahko za vsako izvajanje vidimo podatke, kot so čas izvajanja, status in dnevnik. Za vsako izvajanje lahko tudi odpremo vpogled v paketna opravila, ki so se izvajala. Za vsako opravilo lahko preverimo parametre, nastavitve, čas izvajanja, status, dnevnik, strežnik izvajanja idr.

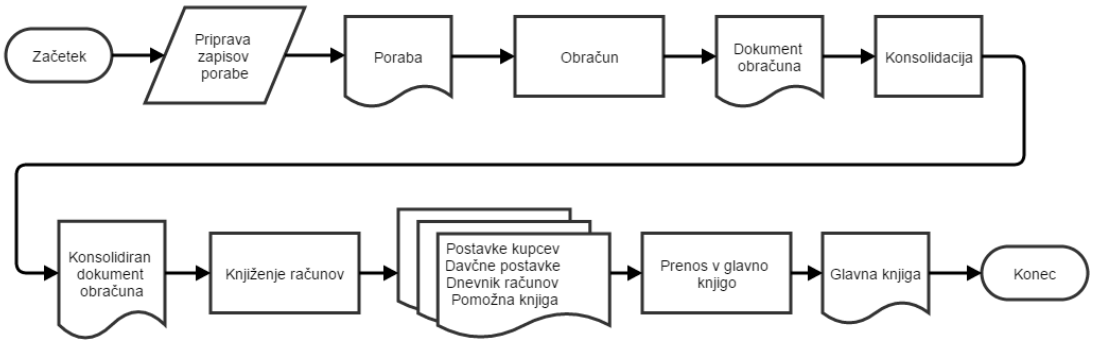


Slika 19: Pregled zgodovine izvajanja paketnih obdelav

2.4 Paketne obdelave izbranega področja

Kot paketne obdelave za obravnavo preobremenitev virov sem izbral nabor paketnih obdelav, s katerimi lahko podpremo poslovno področje dobave zemeljskega plina gospodinjstvom. Procese tega področja lahko v AX-u podpremo z modulom MECOMS, ki ga je razvilo Belgijsko podjetje Ferranti Computer Systems [24]. MECOMS je vertikalna rešitev za področje energetike (zemeljski plin, električna energija, toplovod idr.) razvita na AX platformi, ki podpira področja za upravljanje odnosov s strankami (angl. CRM – *Customer Relationship Management*), obračun (angl. *billing*), upravljanje merskih podatkov, upravljanje storitvenih in s trgom povezanih procesov. Pri tem sem se omejil na procesa priprave računov in knjiženja prejetih plačil. Razlogi za izbiro takšnega pristopa so naslednji:

- možnost uporabe standardnih modulov sistema (standardni AX in MECOMS) in se tako izogniti specifičnim problemom;
- velik del programskih funkcij je realiziranih kot paketna obdelava;
- večina operacij se izvaja nad velikimi količinami podatkov (veliko število kupcev), kar omogoča preučevanje preobremenitev virov sistema;
- pri večini procesov je mogoče potegniti vzporednice s podobnimi procesi (npr. obračun telefonije), kar omogoča pripravo splošno uporabne rešitve.



Slika 20: Proces priprave računov

Pri procesu priprave računov (slika 20) se na podlagi podatkov o porabi pripravijo računi. Pri tem se srečamo z izvajanjem funkcij obračuna, konsolidacije, knjiženja računov in prenosa v glavno knjigo. Vsako od naštetih funkcij lahko izvedemo z uporabo paketne obdelave. Paketne obdelave te vrste se ne izvajajo periodično, ampak se razporejajo na zahtevo uporabnika. Razlog za uporabo izvajanja z uporabo paketne obdelave je v obdelavi velikega številka podatkov. Tako prenesemo izvajanje zahtevne operacije na namenski AOS.

Predmet obdelave za obračun je odjemno mesto. Pri tem se za vsak artikel produkta[3], ki je vezan na trenutno pogodbo, izvede izračun količine na podlagi zapisov porabe (npr. kubični metri porabljenega plina, število dni porabe, število mesecev itn.). Le-ta se uporabi skupaj s

[3] Produkt je sestavljen iz enega ali več artiklov. Primeri artiklov pri prodaji zemeljskega plina so: zemeljski plin, ekološka taksa, trošarina, dodatek k ceni goriva, omrežnina itn. Vsak artikel ima določen svoj cenik in obračunsko funkcijo, ki definira izračun količine artikla.

ceno iz ustreznega cenika za izračun zneska. Rezultat obračuna vsakega artikla je obračunska vrstica, ki nosi informacijo o artiklu, obdobju, količini, ceni in znesku. Za vsak artikel lahko nastane ena ali več obračunskih vrstic (npr. več odčitkov porabe v istem mesecu). V zadnji fazi obračuna pride do kreiranja AX-ovega standardnega izdanega računa (račun v tem trenutku še ni knjižen), ki tvori dokument obračuna. Pri tem za vsako obračunsko vrstico nastane ena vrstica AX-ovega standardnega računa. Paketna obdelava za obračunavanje omogoča vzporedno izvajanje. Pri tem se obdelava izvede v dveh korakih. V prvem koraku osnovna obdelava, ki jo razporedi uporabnik, razdeli nabor merilnih mest v več skupin enakih velikosti. Število skupin je odvisno od nastavitev in maksimalnega števila niti, ki so na AOS dodeljene za izvajanje paketnih obdelav. Po delitvi obdelava programsko kreira novo paketno obdelavo, ki vsebuje za vsako skupino po eno paketno opravilo.

V primeru, ko je stranka lastnica več odjemnih mest, nastane pri postopku obračuna porabe za vsako odjemno mesto svoj dokument obračuna oz. račun. Na željo stranke se takšne dokumente združi v en skupen dokument z uporabo funkcije za konsolidiranje. Pri postopku konsolidacije se ustvari nov dokument, v katerega se prekopirajo vrstice izvornih dokumentov. Izvorni dokumenti se po kopiranju izbrišejo.

Ko so računi pripravljeni za knjiženje, se poknjižijo z uporabo paketne obdelave za knjiženje. Tudi ta paketna obdelava omogoča vzporedno izvajanje, tako da razdeli račune v več skupin. Podobno kot pri obračunu se po delitvi kreira nova paketna obdelava, ki vsebuje po eno paketno opravilo za vsako skupino računov. Rezultat knjiženja računov so zapisi postavk kupcev, dnevnika računov, davčnih postavk in pomožne knjige.

V AX-u se vsak računovodski dokument pri knjiženju najprej zapiše v pomožno knjigo. V glavno knjigo se dokumenti lahko knjižijo ali sinhrono (prenos iz pomožne knjige v glavno se izvede takoj) ali asinhrono (prenos iz pomožne knjige v glavno se izvede kasneje). Asinhroni prenos omogoča tudi sumiran prenos knjižb. Le-to omogoča, da se enake knjižbe (razlika le v znesku) iz pomožne knjige v glavno knjigo zapišejo kot ena sumirana knjižba. Pri masovnih knjiženjih se zaradi zmogljivosti uporablja asinhroni prenos v glavno knjigo, ki se izvaja periodično (npr. vsako uro).

Vsak račun ob knjiženju dobi svojo številko, kar pomeni, da ga lahko natisnemo in shranimo v elektronski obliki. Izdan račun lahko nato pošljemo stranki. Ker je tiskanje računa zaradi specifike enako kot priprava porabe prepuščeno posamezni postavitvi sistema AX, bo uporaba teh dveh funkcij izpuščena.

Slika 21: Proces knjiženja prejetih plačil

Pri procesu knjiženja prejetih plačil (slika 21) je potrebno vsako prejeto plačilo povezati s postavko kupca, ki je nastala v procesu priprave računa, in ga poknjižiti. Slednje omogoča uporaba dokumenta temeljnice. Podlaga za pripravo dokumenta temeljnice je navadno

obdelava bančnega izpiska. Le-to bo zaradi specifik posamezne postavitve sistema izpuščeno. Knjiženje temeljnice lahko izvedemo z uporabo paketne obdelave. Pri tem se obdelavo razporedi na zahtevo uporabnika. Rezultat knjiženja temeljnice so zapisi postavk kupcev, bančnih postavk in glavne knjige. Ob tem pride tudi do zapiranja postavk kupcev, ki pripadajo računom, na katera so bila vezana plačila.

V ERP se poleg paketnih obdelav, ki so neposredno povezane z opisanima procesoma, običajno izvajajo še dodatne paketne obdelave za podporo drugim procesom v podjetju. Paketna obdelava za posodabljanje bruto bilance [4] se običajno izvaja periodično enkrat na uro. Platforma AX implementira .NET komponento WWF za delovne tokove. Za procesiranje prehodov delovnih tokov se uporablja paketna obdelava, ki se izvaja na nekaj minut. Podobno se uporablja tudi paketna obdelava za proženje AIF vrat, ki poskrbijo za integracijo AX-a z ostalimi aplikacijami v podjetju [10]. Enkrat ali večkrat dnevno se v sistemu izvajajo tudi paketne obdelave, ki poskrbijo za osvežitev šifrantov, kot so registri davčnih zavezancev, registri bančnih računov poslovnih subjektov, tečajne liste idr.

2.5 Lastnosti paketnih obdelav

Na podlagi predstavljenih procesov in uporabe paketnih obdelav lahko paketne obdelave sistema AX opredelimo z naslednjimi lastnostmi:

- **Izvedba na zahtevo:** obdelava je razporejena na zahtevo in se izvede enkrat. Pri tem časa izvedbe ni mogoče predvideti.
- **Ponovljivost**: obdelava je razporejena ponavljajoče z določenim intervalom. Ker so ponavljajoče obdelave razporejene vnaprej, zanje lahko predvidevamo, kdaj se bodo izvedle.
- **Precedenčna odvisnost**: pričetek izvajanja opravila obdelave je pogojen z zaključkom drugega opravila. Tako mora opravilo pred pričetkom izvajanja počakati, da se zaključi eno ali več predhodnih opravil z ustreznim statusom.
- **Vzporedno izvajanje**: obdelava razbije celotno opravilo na več manjših opravil, ki jih izvede vzporedno. Ker je čas izvajanja manjšega opravila običajno krajši, je zato tudi čas izvajanja celotne obdelave krajši.
- **Zahtevnost:** obdelava lahko povzroči obremenitev določenega nabora virov. Sočasno izvajanje več takšnih obdelav pa lahko povzroči preobremenitev enega ali več virov sistema.

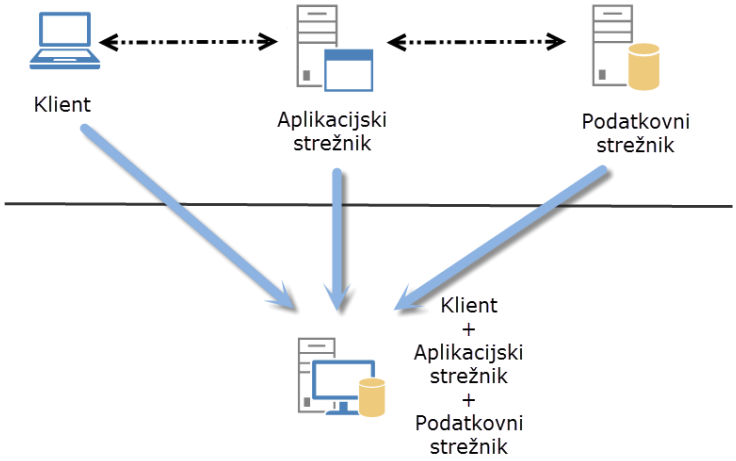
[4] Bruto bilanca (angl. *Trial balance*) je knjigovodski izkaz, ki prikazuje stanje po kontih v obliki salda, vsote knjižb v dobro in vsote knjižb v breme. Namen bruto bilance je prikazati matematično pravilnost v dvostavnem knjigovodstvu.

3 Opredelitev testnega okolja in bremena

Za izvedbo in preučevanje ustreznosti razširitve je bilo uporabljeno testno okolje. V nadaljevanju so opredeljene lastnosti okolja in njegovo breme.

3.1 Omejitve sistema

Pri analizi sistema in izvajanju meritev je bila uporabljena najbolj enostavna topologija sistema, ki se navadno uporablja pri postavitvi razvojnih okolij [28]. Pri tem so bile na isti računalniški sistem nameščene komponente vseh treh nivojev: podatkovni strežnik, aplikacijski strežnik in aplikacija klienta (slika 22). Postavitev bolj kompleksnih topologij, s katerimi omogočamo večjo razpoložljivost in skalabilnost produkcijskih okolij, je bila predstavljena pri opisu arhitekture sistema AX v razdelku 2.1.



Slika 22: Namestitev komponent sistema AX na en računalniški sistem

Takšna postavitev sistema prinaša naslednje omejitve:

- vse tri komponente si morajo deliti isti procesor;
- vse tri komponente si morajo deliti isti pomnilnik;
- vse tri komponente si morajo deliti isto diskovno enoto;
- vpliv zmogljivosti omrežne komunikacije med komponentami je zanemarjen;
- monitoring uporabe določenega sistemskega vira s strani posamezne komponente ni vedno možen.

Navedene omejitve so zahtevale vpeljavo naslednjih predpostavk:

- vpliv procesa aplikacije klienta lahko zanemarimo pri spremljanju izvajanja paketnih obdelav;
- omejitev uporabe istega procesorja lahko obidemo z dodelitvijo posameznih procesorskih jeder procesorja posameznemu procesu;
- omejitev uporabe istega pomnilnika lahko obidemo z rezervacijo pomnilnika za posamezen proces;

- vsa aktivnost na diskovni enoti je posledica procesa podatkovnega strežnika;
- verjetnost preobremenitve omrežja v primerjavi z ostalimi viri je majhna;
- z uporabo sistemskih performančnih števcev lahko dovolj dobro izvajamo monitoring sistemskih virov.

Kot strežnik, na katerega so bile nameščene komponente sistema AX, je bila uporabljena razvojna delovna postaja z naslednjimi proizvajalčevimi specifikacijami:

- **Procesor:** Intel® Core™ i7-3740QM 2,7 GHz, 64 bitni procesor, 4 jedra (8 logičnih jeder z uporabo tehnologije Hyper-Threading[5]).
- **Pomnilnik:** 16 GB, DDR3, dvokanalni pomnilnik.
- **Diskovna enota:** Samsung 840 SSD, kapaciteta 500GB, predpomnilnik 512MB, zaporedno branje do 540 MB/s, zaporedno pisanje do 330 MB/s, naključno branje do 98000 IOPS, naključno pisanje do 70000 IOPS, dostopni čas 0,1 ms.

Procesi vseh treh komponent se izvajajo na istem računalniškem sistemu in uporabljajo iste sistemske vire. Vpliv aplikacije klienta lahko pri spremljanju zanemarimo, ker se izvajanje paketnih obdelav izvaja zgolj na nivoju aplikacijskega strežnika in podatkovnega strežnika.

Z uporabo konfiguracije aplikacijskega in podatkovnega strežnika lahko smiselno razdelimo uporabo procesorskih jeder in pomnilnika. Tako smo vsakemu od strežnikov dodelili po 4 logična procesorska jedra. Pri dodelitvi pomnilnika smo za podatkovni strežnik rezervirali 4GB pomnilnika (podatkovna baza s poslovnimi podatki v času izvajanja meritev ni presegla velikosti 3 GB). Aplikacijski strežnik ni bil omejen pri porabi pomnilnika. Izkustveno je bila pričakovana poraba pomnilnika s strani aplikacijskega v rangu 8 GB. Le-to je bilo zagotovljeno ob upoštevanju 2 GB rezerve za delovanje operacijskega sistema in izvajanje monitoringa.

Obema strežnikoma je bila na voljo ena skupna diskovna enota. Vendar je izvajanje meritev pokazalo, da je praktično vsa aktivnost na diskovni enoti posledica podatkovnega strežnika.

Ker so bile vse komponente nameščene na en računalnik, je vsa komunikacija med komponentami sistema potekala znotraj operacijskega sistema. Iz tega stališča je bilo zanemarjeno opazovanje zmogljivosti omrežne komunikacije. Predpostavljamo, da bi pri obremenjevanju sistema s paketnimi obdelavami prišlo do preobremenitve ostalih sistemskih virov prej, kot bi prišlo do ozkega grla na omrežju. Ker se v praksi pri postavitvah navadno uporabljajo strežniki, ki se nahajajo znotraj iste strežniške omare in so povezani med seboj s hitrim omrežjem, je verjetnost preobremenitve omrežja majhna. Vedno pa lahko pri uporabi ločenih strežnikov uporabimo ustrezni performančni števec za spremljanje prenosa podatkov.

Navedene omejitve in predpostavke se nanašajo na problem omejenosti virov na opazovanem sistemu. Le-to je tudi dobrodošlo s stališča ciljev naloge, ki predstavljajo vpeljavo mehanizmov za izogibanje preobremenitvam posameznih virov sistema. Omenjenim omejitvam se lahko pri postavitvah produkcijskih okolij izognemo s postavitvijo posameznih komponent sistema AX

[5] Tehnologija Intel® Hyper-Threading omogoča hkratno izvajanje več niti na enem procesorskem jedru in s tem večjo izkoriščenost procesorskega vira [21]. Pri tem eno procesorsko jedro operacijski sistem obravnava kot dve logični oz. navidezni jedri.

na ločene strežnike in podvajanjem le-teh. Potrebo po predpostavkah pa bi izničila že osnovna postavitev, kjer bi bila vsaka komponenta nameščena na svoj strežnik.

3.2 Analiza virov sistema

Pri analizi virov sta bili zastavljeni dve vprašanji:

- Kaj so viri sistema AX, ki so ključni pri izvajanju paketnih obdelav?
- Na kakšen način lahko spremljamo njihovo obremenitev?

S preučevanjem testnega okolja sem prišel do ugotovitve, da paketne obdelave za izvajanje potrebujejo:

1. eno ali več izvajalnih niti strežnika AOS;
2. vire operacijskega sistema (v nadaljevanju sistemski viri), kot so procesor, pomnilnik, diskovne enote idr.;
3. vire podatkovnega modela (v nadaljevanju podatkovni viri), kot so zapisi v tabelah podatkovne baze, ki jih posamezna obdelava obdeluje.

Razpoložljivost izvajalne niti strežnika AOS ne vpliva na interaktivno uporabo sistema AX. Prosta izvajalna nit strežnika AOS je zgolj predpogoj, da obdelava lahko preide v stanje izvajanja in prične koristiti sistemske in podatkovne vire. V nadaljevanju bo pozornost posvečena sistemskim in podatkovnim virom, katerih obremenjenost vpliva na interaktivno uporabo sistema AX.

3.2.1 Sistemski viri

Pri izvajanju različni tipi paketnih obdelav različno obremenjujejo posamezne vire operacijskega sistema. Nekatere obdelave so lahko bolj računsko intenzivne in zahtevajo več procesorskega časa. Druge so lahko bolj prostorsko zahtevne in zato zahtevajo porabo pomnilnika.

Spremljanje obremenitve virov operacijskega sistema je mogoče z uporabo strojnih ali programskih komponent, ki se uporabljajo za spremljanje stanja virov računalniškega sistema [30].

Za zbiranje podatkov o obremenitvi sistemskih virov je bilo uporabljeno standardno orodje Windows Performance Monitor, ki je del operacijskega sistema Windows [32]. Orodje omogoča zbiranje in analizo t. i. performančnih števcev (angl. *performance counter*). Prednost uporabe komponente performančnih števcev je, da je dostop do njihovih vrednosti omogočen tudi z uporabo programske knjižnice ogrodja .NET, in sicer z uporabo razreda `System.Diagnostics.PerformanceCounter` [27]. Le-to omogoča uporabo performančnih števcev pri zasnovi programske rešitve v nadaljevanju. Pomanjkljivost določenih tipov performančnih števcev je, da jih ni mogoče zbirati za posamezen proces. Tako na primer števce za diskovne enote lahko zbiramo le za posamezne diskovne enote, medtem ko števce za CPE

lahko spremljamo glede na proces. To je ovira le, ko spremljamo več procesov na istem računalniku in želimo ugotoviti, kateri proces obremenjuje posamezen vir.

Pri analizi sistemskih virov so bili opazovani naslednji performančni števci:

- **Delež procesorskega časa procesa AOS (CPE AOS):** opazovana vrednost je delež uporabljenega procesorskega časa s strani aplikacijskega strežnika. Ker so dodeljena 4 logična procesorska jedra, je vrednost lahko večja od 100 %. Tako ob polni obremenitvi lahko doseže 400 % (100 % za vsako jedro).
- **Pomnilnik AOS (RAM AOS):** opazovana vrednost je zasedenost pomnilnika s strani aplikacijskega strežnika v bajtih.
- **Delež procesorskega časa procesa SQL (CPE SQL):** opazovana vrednost je delež uporabljenega procesorskega časa s strani podatkovnega strežnika. Ker so dodeljena 4 logična procesorska jedra, je vrednost lahko večja od 100 %.
- **Diskovna enota (DISK):** opazovana vrednost je povprečno število zahtev v vrsti in obdelavi.

3.2.2 Podatkovni viri

Ker se obdelave pri izvajanju procesa dopolnjujejo (rezultat prve obdelave je podlaga za izvajanje druge), pride do situacij, ko dve obdelavi uporabljata isto tabelo ali celo isti zapis. Enako lahko skušata posamezni zapis urejati paketna obdelava in uporabnik preko aplikacije klienta. Pri takšnih operacijah nad podatki prihaja na zapisih in tabelah v relacijski podatkovni bazi do zaklepanj, ki so posledica nadzora in upravljanja transakcij nad podatki s strani SUPB [3]. Zaklepanje zapisa tako preprečuje, da bi ga nekdo drug urejal v času, ko je zaklep aktiven. Iz tega sledi, da tabele in zapisi predstavljajo skupni vir, ki ob prevelikem številu sočasnih manipulacij lahko postane ozko grlo.

Za spremljanje obremenitve oz. stanja podatkovnega vira je bila tako definirana mera, ki temelji na številu aktivnih zahtev za zaklepanja (v nadaljevanju zaklepanja). Ker je podatkovni model dokaj obsežen, je bil za lažje spremljanje stanja definiran pojem skupine tabel. Število zaklepanj pa je enako vsoti vseh zaklepanj na tabelah v skupini. Pri tem posamezna skupina tabel predstavlja podatkovne tabele, ki skupaj tvorijo neko zaključeno celoto. Tako na primer skupino dokumenta tvorita tabeli za glavo in vrstico dokumenta.

Za zbiranje podatkov o zaklepanjih na podatkovni bazi je bilo uporabljeno beleženje posnetkov podatkov o zaklepanjih. Le-te lahko pridobimo na podatkovnem strežniku SQL Server 2012 s klicem sistemskega pogleda (angl. *view*) `sys.dm_tran_locks`, ki nam vrne informacije o trenutnih zahtevah upravitelja zaklepanj [29].

Pri podatkovnih virih so bile opazovane naslednje skupine tabel:

- **Poraba:** skupina predstavlja podatkovni model za beleženje porab na odjemnih mestih. Obdelava obračuna med izvajanjem zapise bere in posodablja.

- **Račun:** skupina predstavlja podatkovni model računa, ki se sestoji iz tabel glave računa, vrstice računa, obračunske vrstice in dodatnih tabel, ki nosijo dodatne informacije bodisi vezane na račun kot celoto bodisi vrstico računa. Obdelava obračuna med izvajanjem zapise kreira in posodablja. Obdelava konsolidacije zapise briše in jih na novo kreira v obliki združenih računov. Obdelava knjiženja računov zapise bere in posodablja.
- **Dokument:** skupina predstavlja podatkovni model splošnega računovodskega dokumenta (univerzalna podatkovna struktura za hranjenje podatkov, ki so potrebni za knjiženje), ki omogoča knjiženje preko pomožne knjige in ga uporabljajo različni dokumenti sistema AX. Podatkovni model se v grobem sestoji iz tabel glave dokumenta, vrstice dokumenta, računovodskih porazdelitev, vrstic davka idr. Ker je to podporna podatkovna struktura za račun, je podvržena podobnim manipulacijam s strani obdelav kot skupina tabel računa.
- **Temeljnica:** skupina predstavlja podatkovni model dokumenta temeljnice, ki se uporablja za knjiženje raznovrstnih postavk. V izbranem primeru se dokument uporablja za knjiženje plačil, katerih rezultat so postavke kupcev, bančne postavke in postavke knjižb na konte glavne knjige. Podatkovni model se v grobem sestoji iz tabel glave temeljnice, vrstic temeljnice in specifikacije za zapiranje postavke kupca (zapiranje knjiženega računa). Obdelava knjiženja temeljnice plačil zapise bere in posodablja. Obdelava knjiženja temeljnice zapise specifikacije zapiranj briše.
- **Dnevnik računov:** skupina predstavlja podatkovni model knjiženega računa (dnevnik oz. zgodovina izdanih računov). Podatkovni model se v grobem sestoji iz glave dnevnika računa in vrstice dnevnika računa. Obdelava knjiženja računa zapise kreira.
- **Postavke kupcev:** skupina predstavlja podatkovni model knjižene postavke kupca. Podatkovni model se v grobem sestoji iz tabel postavke, odprtih delov postavke, zaprtih delov postavke idr. Obdelava za knjiženje računov kreira zapise postavk (računov) in odprtih delov postavk. Obdelava za knjiženje temeljnice plačil kreira zapise postavk (plačil) in pretvarja odprte dele postavk v zaprte dele postavk (ob zapiranju postavke se izvede brisanje odprtega dela in kreiranje zaprtega dela).
- **Pomožna knjiga:** skupina predstavlja podatkovni model pomožne knjige. Podatkovni model se v grobem sestoji iz tabel računovodskega dogodka, listine, postavke listine in porazdelitve postavke listine. Obdelava za knjiženje računov zapise kreira. Obdelava za prenos v glavno knjigo zapise bere in posodablja.
- **Glavna knjiga:** skupina predstavlja podatkovni model glavne knjige. Podatkovni model se v grobem sestoji iz tabel listine in postavke listine. Obdelavi za prenos v glavno knjigo in knjiženje temeljnice plačil zapise kreirata.
- **Davčna postavka:** skupina predstavlja knjiženo davčno postavko. Obdelava za knjiženje računa zapise kreira.
- **Bančna postavka:** skupina predstavlja knjiženo postavko bančnega računa. Obdelava za knjiženje temeljnice plačil kreira zapise kot proti knjižbe postavkam plačil kupcev.

3.3 Analiza bremena

Breme je množica vhodnih zahtev, ki obremenjuje strežniške sposobnosti opazovanega računalniškega sistema [15]. Zahteve v opazovanem primeru predstavljajo paketne obdelave, ki jih izvaja aplikacijski strežnik. Pri analizi bremena sem se namenoma osredotočil na nabor paketnih obdelav, ki so po naravi procesno in podatkovno intenzivne. Le-to so obdelave, ki skupaj tvorijo proces obračunavanja energije:

- **Obračun:** obdelava na podlagi zabeleženih porab pripravi račune. Obdelava se izvede v dveh korakih. Prvi korak se izvede kot eno opravilo na eni niti in razbije nabor odjemnih mest v 8 skupin. Drugi korak izvede vzporedno 8 opravil, izmed katerih vsako izvede obračun za svojo skupino.
- **Konsolidacija računov:** obdelava po potrebi združuje po več računov iste stranke v en skupen račun. Obdelava se izvede kot eno opravilo.
- **Knjiženje računov:** obdelava izvede knjiženje pripravljenih računov, ki so rezultat obračuna. Obdelava se izvede v dveh korakih. Prvi korak se izvede kot eno opravilo na eni niti in razbije nabor računov v 8 skupin. Drugi korak izvede vzporedno 8 opravil, izmed katerih vsako izvede knjiženje svoje skupine računov.
- **Prenos v glavno knjigo (GK):** obdelava izvede prenos knjižb računov iz pomožne knjige v glavno knjigo.
- **Knjiženja plačil:** obdelava izvede knjiženje temeljnice plačil. Obdelava se izvede v dveh korakih. Prvi korak pripravi opravilo za knjiženje temeljnice, ki se izvede v drugem koraku.

Izvajanje obdelav je bilo pri analizi bremena spremljano na vzorcu 1000 odjemnih mest. Spodnja tabela prikazuje podatke o izvajanju posamezne paketne obdelave.

Obdelava	Število izvajalnih niti	Čas izvajanja v sekundah
Obračun	8	115
Konsolidacija računov	1	221
Knjiženje računov	8	158
Prenos v glavno knjigo	1	72
Knjiženje plačil	1	58

Tabela 1: Paketne obdelave – izvajalni podatki

Pri izvajanju različni tipi paketnih obdelav različno obremenjujejo posamezne vire sistema, kot so procesor, pomnilnik, diskovne enote idr. V spodnji tabeli so navedene obremenitve posameznega vira s strani posamezne paketne obdelave. Pri tem je vedno navedena povprečna vrednost in poleg nje v oklepaju maksimalna vrednost.

Obdelava\Vir	CPE AOS (%)	RAM AOS (GB)	CPE SQL (%)	DISK (n)
Obračun	341 (400)	5,3 (8,3)	137 (190)	0,80 (16,07)
Konsolidacija	65 (130)	3,3 (6,2)	36 (58)	2,93 (45,27)
Knjiženje računov	230 (348)	7,4 (6,8)	266 (335)	0,42 (21,42)
Prenos v GK	20 (75)	1,5 (3,2)	38 (74)	0,67 (12,45)
Knjiženje plačil	61 (103)	2,3 (3,6)	29 (32)	2,76 (62,49)

Tabela 2: Paketne obdelave – obremenitev sistemskih virov

V spodnji tabeli so navedene obremenitve posamezne skupine tabel s strani posamezne paketne obdelave. Pri tem je navedeno povprečno in maksimalno število zaklepanj na sekundo.

Skupina\Obdelava	Obračun	Konsolidacija	Knjiženje računov	Prenos v GK	Knjiženje plačil
Poraba	5,72 (15)				
Račun	863,25 (1351)	128,68 (346)	91,91 (120)		
Dokument	597,43 (1207)	99,21 (295)	1132,90 (1623)		0,03 (2)
Temeljnica					2895,44 (5672)
Dnevnik računov			866,53 (1082)		
Postavke kupcev			14,03 (107)		50393,09 (101844)
Pomožna knjiga			144,98 (374)	0,39 (3)	
Glavna knjiga				12171,43 (14200)	3322,79 (22595)
Davčna postavka			359,27 (620)		
Bančna postavka					4605,00 (9355)

Tabela 3: Paketne obdelave – obremenitev podatkovnih virov

4 Opredelitev umetnega bremena

Zadani cilji naloge predvidevajo razširitev standardnega modula za izvajanje paketnih obdelav z mehanizmi za detekcijo in preprečevanje preobremenjenosti virov sistema. Zato se je smiselno osredotočiti na testna bremena, ki izpostavljajo sistem nenadnim preobremenitvam in omogočajo opazovanje vpliva vpeljane razširitve. V praksi se takšne situacije lahko pojavijo, ko več uporabnikov istočasno razporedi več paketnih obdelav.

Za ocenitev ustreznosti razširitve je potrebno najprej določiti testno breme, ki omogoča ponovljivost in primerjavo meritev. Zato je bila predvidena uporaba različnih tipov testnih bremen:

- Breme, sestavljeno iz generičnih paketnih obdelav, ki simulirajo obremenitev sistemskega vira, v različnih neugodnih razporeditvah.
- Breme, sestavljeno iz generičnih paketnih obdelav, ki simulirajo obremenitev podatkovnega vira, v različnih neugodnih razporeditvah.
- Breme, sestavljeno iz realnih paketnih obdelav, v različnih neugodnih razporeditvah.

Namen uporabe neugodne razporeditve obdelav je preverjanje ustreznosti delovanja razširitve za preprečevanje preobremenitev. Uporaba vnaprej pripravljenih razporeditev obdelav pa omogoča ponovljivost in primerjavo meritev na testnem sistemu. Vsako meritev lahko tako opravimo na več različnih konfiguracijah sistema ob enakih razporeditvah obdelav.

4.1 Breme generičnih paketnih obdelav

Za preverjanje pravilnosti delovanja razširitve je bila najprej predvidena uporaba umetnega tesnega bremena, s katerim lahko simuliramo obnašanje v najslabših izidih in prikažemo primer uporabe. Ker so bili viri sistema pri analizi virov razdeljeni na sistemske in podatkovne vire, sta bili pripravljeni dve generični paketni obdelavi.

Prva generična paketna obdelava je namenjena simulaciji obremenitve sistemskega vira. Z obdelavo lahko za določen čas v določeni količini obremenimo CPE. Obdelava sprejme naslednja dva parametra:

- delež procesorskega časa procesa, ki ga bo obdelava izrabila;
- čas izvajanja obdelave.

Druga generična paketna obdelava je namenjena simulaciji obremenitve podatkovnega vira. Z obdelavo lahko simuliramo zaklepanja zapisov v dveh podatkovnih tabelah, ki prestavljata testni dokument. Prva tabela predstavlja glavo dokumenta, ki vsebuje znesek vseh vrstic dokumenta. Druga tabela predstavlja vrstico dokumenta, ki vsebuje znesek vrstice. Obdelava ob vsakem zagonu naloži vsak dokument posebej in izvede naslednje operacije:

1. Odpre novo transakcijo na podatkovnem strežniku.
2. Naloži glavo dokumenta.
3. Naloži vsako vrstico dokumenta in vanjo zapiše naključen znesek.

4. Izračuna vsoto zneskov vseh vrstic in jo zapiše v glavo dokumenta.
5. Da proces v spanje (angl. *sleep*) za čas, ki ga določimo s parametrom.
6. Zaključi transakcijo.

Medtem ko je v 5. koraku proces obdelave v spanju in transakcija še ni zaključena, ostajajo aktivna zaklepanja na posodobljenih zapisih. Le-to ob hkratnem izvajanju večjega števila obdelav povečuje število zaklepanj in konflikte ob posodabljanju istih zapisov.

Za preverjanje pravilnosti delovanja rešitve za obvladovanje preobremenitev virov je bila predvidena uporaba vnaprej pripravljenih različnih razporeditev osmih generičnih paketnih obdelav istega tipa. Razporeditve A, B, C, D in E, ki so bile pripravljene za simulacijo preobremenitev sistemskega in podatkovnega vira, prikazuje spodnja tabela. Pri tem je za vsako obdelavo, ki je predstavljena s številko, v tabeli zapisan razporejen čas pričetka v sekundah.

Obdelava\Razporeditev	A	B	C	D	E
1	0	0	0	0	0
2	0	0	0	60	0
3	0	0	0	60	60
4	0	0	60	120	60
5	0	60	60	120	120
6	0	60	120	120	120
7	0	60	180	180	180
8	0	60	180	180	180

Tabela 4: Razporeditve za simulacijo preobremenitev virov z generičnimi obdelavami

Razporeditev A predstavlja situacijo, ko se vseh 8 obdelav prične izvajati istočasno. Ostale razporeditve pa predstavljajo situacije, ko intenziteta postopoma narašča. Časi razporeditve so zaokroženi na 60 sekund, ker je le-to tudi interval preverjanja razpoložljivosti novih paketnih obdelav s strani izvajalnih niti strežnika AOS.

4.2 Breme realnih paketnih obdelav

Primeri realnih paketnih obdelav različno obremenjujejo več virov sistema tako v času kot v intenziteti. Za ocenitev ustreznosti rešitve za obvladovanje preobremenitev virov pri izvajanju realnih paketnih obdelav so bila pripravljena bremena, ki so sestavljena iz naslednjih petih paketnih obdelav:

- obračun,
- konsolidacija računov,
- knjiženje računov,
- prenos v glavno knjigo in
- knjiženje plačil.

Za izvajanje meritev so bile tako pripravljene različne razporeditve, ki vsebujejo vseh pet obdelav. Pri pripravi razporeditev so bili upoštevani rezultati analize bremena. Le-ti so bili podlaga za pripravo neugodnih razporeditev na podlagi informacije o intenziteti obremenitve

posameznega vira s strani posamezne obdelave. Razporeditve A, B, C, D, E, F in G, ki so bile pripravljene, prikazuje spodnja tabela. Pri tem je za vsako obdelavo v tabeli zapisan razporejen čas pričetka v sekundah.

Obdelava\Razporeditev	A	B	C	D	E	F	G
Obračun	0	0	60	0	60	120	0
Konsolidacija	0	60	0	120	240	0	60
Knjiženje računov	0	120	120	240	0	60	180
Prenos v GK	0	180	240	60	120	240	240
Knjiženje plačil	0	240	180	180	180	180	120

Tabela 5: Razporeditve za simulacijo preobremenitev virov z realnimi obdelavami

Razporeditev A predstavlja situacijo, ko ima vseh pet obdelav enak čas pričetka izvajanja. Razporeditev B predstavlja zaporedje, kjer predhodna obdelava ustvarja podatke naslednji obdelavi. Razporeditve C, D, E, F in G predstavljajo različna zaporedja obdelav, ki obremenjujejo podatkovne vire dokumenta, postavk kupcev in glavne knjige.

Časi razporeditve so ravno tako kot pri generičnih obdelavah zaokroženi na 60 sekund, ker je le-to tudi interval preverjanja razpoložljivosti novih paketnih obdelav s strani izvajalnih niti strežnika AOS.

5 Obvladovanje preobremenitve virov

V tem poglavju bo predstavljena rešitev za obvladovanje preobremenitev, ki sem jo razvil kot razširitev standardnega modula za razporejanje paketnih obdelav v sistemu AX. Pri tem bo najprej predstavljeno delovanje standardnega algoritma za razporejanje paketnih obdelav. Le-to bo služilo tudi kot podlaga, na kateri bo v nadaljevanju poglavja zasnovano spremljanje obremenitve virov, kontrola izvajanja paketnih obdelav in sama vgradnja rešitve v sistem.

5.1 Delovanje standardne rešitve

Osnovna naloga modula za paketne obdelave je izvajanje le-teh. Pri tem algoritem za razporejanje opravil oz. razporejevalnik odloča, katera paketna obdelava oz. paketno opravilo se bo izvedlo naslednje.

Problem razporejanja opravil, ki ga rešuje razporejevalnik, lahko opredelimo z definicijo strežnih enot, definicijo opravil in ciljem [9]. Definicija strežnih enot opredeli število, enakost, odvisnost in omejitve strežnih enot idr. Definicija opravil opredeli čase prihoda v sistem, izvajanja, rokov izvedbe, prioritete idr. Cilj problema je pri tem lahko sestavljen iz enega ali več ciljev, kot je na primer minimizacija skupnega časa za izvedbo vseh opravil, maksimalne zamude rokov, utežene vsote časov izvedbe opravil idr. Tako je lahko eden od ciljev tudi zmerna obremenitev virov sistema.

Glede na informacije, ki so dostopne o opravilih, se razporejanje opravil deli na [9, 11]:

- **Sprotno razporejanje (angl. *on-line scheduling*):** število opravil, čas prihoda v sistem in čas izvajanja niso znani, zato mora razporejevalnik sprejeti odločitev šele, ko opravilo prispe v sistem (čas izvajanja postane znan šele po zaključku opravila).
- **Vnaprejšnje razporejanje (angl. *off-line scheduling*):** število opravil, čas prihoda v sistem in čas izvajanja so podani, zato lahko razporejevalnik vnaprej pripravi plan, ki čimbolj zadovolji cilj problema. Pri tem ločimo še deterministično razporejanje, kjer so podani konkretni časi, in stohastično razporejanje, kjer so časi podani statistično.

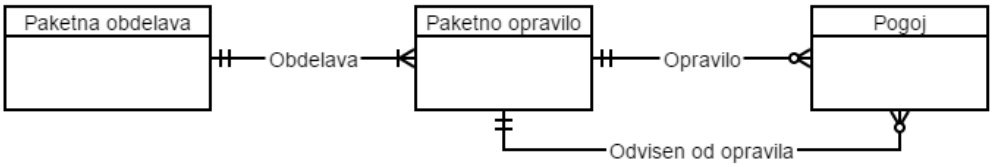
Glede ne predstavitev ogrodja za paketne obdelave iz razdelka 2.3.4 in delovanje algoritma, ki bo predstavljeno v nadaljevanju, lahko izvajanje paketnih obdelav v sistemu AX opredelimo z naslednjimi lastnostmi [9]:

- **Paketno procesiranje:** obdelava je lahko sestavljena iz več opravil (paket), ki se lahko izvajajo sočasno. Obdelava je zaključena šele, ko se zaključi zadnje opravilo.
- **Več različnih procesnih enot:** izvaja se lahko več obdelav hkrati na različno zmogljivih aplikacijskih strežnikih.
- **Precedenčne omejitve:** zaporedje opravil, ki sestavljajo eno paketno obdelavo, je lahko pogojeno.
- **Sprotno razporejanje:** čas razporeditve in čas izvajanja obdelave in njenih opravil nista znana. Čas razporeditve je lahko znan le v primeru, da gre za ponavljajočo obdelavo.

Algoritem razporejanja paketnih obdelav sistema AX ustreza postopku razporejanja na podlagi seznama (angl. *list scheduling*) [9, 11]. Pri tem postopku razporejevalnik vedno vzame prvo opravilo z vrha seznama.

Uporaba paketnih obdelav je bila predstavljena v razdelku 2.3.4. Predno predstavimo sam postopek izbire opravil, si poglejmo še, kako so podatki o opravilih shranjeni v podatkovni bazi. Poenostavljen podatkovni model je prikazan na sliki 23 in je sestavljen iz naslednjih tabel:

- **BatchJob:** predstavlja paketno obdelavo in hrani podatke:
  - *OrigStartDateTime* – čas razporeditve,
  - *recurranceData* – definicija ponovljivosti,
  - *StartDateTime* – čas pričetka izvajanja,
  - *EndDateTime* – čas zaključka izvajanja,
  - *Status* – status obdelave.
- **Batch:** predstavlja paketno opravilo in hrani podatke:
  - *BatchJobId* – identifikator paketne obdelave, kateri opravilo pripada;
  - *ClassNumber* – identifikator razreda X++, ki implementira kodo opravila;
  - *Parameters* – serializirani parametri razreda X++;
  - *StartDateTime* – čas pričetka izvajanja;
  - *EndDateTime* – čas zaključka izvajanja;
  - *Status* – status opravila;
  - *GroupId* – identifikator paketne skupine, ki pogojuje, na katerih strežnikih se lahko opravilo izvaja;
  - *ConstraintType* – način obravnave pogojev odvisnosti, ki ima naslednji dve možnosti:
    - *All* – vsi pogoji odvisnosti morajo biti izpolnjeni;
    - *Any* – vsaj en pogoj odvisnosti mora biti izpolnjen.
- **BatchConstraints:** predstavlja relacijo pogoja odvisnosti med opraviloma in hrani podatke:
  - *BatchId* – identifikator trenutnega opravila, kateremu odvisnost pripada;
  - *DependsOnBatchId* – identifikator predhodnega opravila, od katerega je opravilo odvisno;
  - *ExpectedStatus* – status, ki ga mora predhodno opravilo doseči, da je pogoj izpolnjen.



Slika 23: Podatkovni model paketnih obdelav

Vsaka nova paketna obdelava, ki jo uporabnik razporedi, se tako shrani kot en zapis paketne obdelave v tabeli BatchJob in vsaj kot en zapis paketnega opravila v tabeli Batch.

Sam postopek izbire opravil za izvajanje in vzdrževanje seznama opravil se izvaja na strežniku AOS, ki je dodeljen kot strežnik za izvajanje paketnih obdelav. Celoten postopek izvajajo tri izvajalne niti jedra strežnika AOS, ki periodično kličejo izbran nabor metod razreda `BatchRun` [20]. Le-ta je del aplikacijske kode X++ in ga lahko prilagodimo z razvojnim orodjem MorphX.

Prva nit vsakih 60 sekund izvede klic metode `serverGetTask()`, ki izvede postopek izbire opravila, ki je prikazan na sliki 26. Metoda izvede naslednje korake:

1. Izvede klic metode `serverGetOneTask()`, ki preveri, če v tabeli opravil obstaja kakšno opravilo s statusom *pripravljen* (angl. *Ready*). Poizvedba, s katero metoda pridobi ustrezno opravilo, je prikazana na sliki 24.

```
select firstonly pessimisticlock RecId, CreatedBy, ExecutedBy, StartDateTime, Status,
    SessionIdx,SessionLoginDateTime, Company, ServerId
    ,DataPartition
from batch
where batch.Status == BatchStatus::Ready
&& batch.RunType == BatchRunType::Server
&& (Session::isServer() || batch.CreatedBy == user)
&& batch.DataPartition == partitions.PartitionKey
join Language from userInfo
order by batch.Priority
    where userInfo.Id == batch.CreatedBy
    && userInfo.Enable == true
    && userInfo.Partition == partitions.RecId
exists join batchServerGroup
    where batchServerGroup.ServerId == serverId
        && batch.GroupId == batchServerGroup.GroupId;
```

Slika 24: Poizvedba za izbor opravila

2. V kolikor obstaja pripravljeno opravilo, se ga preda v izvajanje in s tem se postopek izbire zaključi. V nasprotnem primeru se preveri, če v tabeli obdelav obstaja zapis, katerega čas razporeditve je zapadel in ga je potrebno posodobiti v *izvajanje* (angl. *Executing*) (slika 25). S tem postopkom se pripravijo zapisi v tabeli obdelav za izvedbo naslednjega koraka.

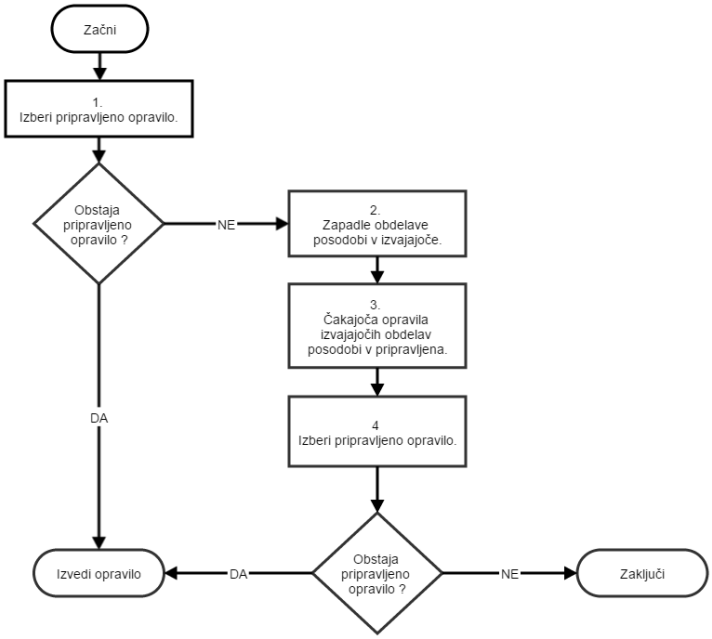
```
ttsbegin;
//Update batchjobs ready to start running. Set their status to executing.
//Change those jobs that have a task with a matching group with the serverId
update_recordset batchJob setting
    Status = BatchStatus::Executing,
    StartDateTime = thisDate
where batchJob.Status == BatchStatus::Waiting
    &&  batchJob.OrigStartDateTime   <= thisDate
exists join batch
    where batch.BatchJobId == batchJob.RecId
exists join batchServerGroup
    where batch.GroupId == batchServerGroup.GroupId
    && batchServerGroup.ServerId == serverId;

ttscommit;
```

Slika 25: Ukaz za posodobitev zapadlih obdelav

3. Izvede se klic metode `serverProcessDependencies()`. Le-ta za vsa opravila, ki so v statusu *čakajoč* (angl. *Waiting*) in katerih obdelave so v statusu izvajanja, izvede posodobitev statusa v *pripravljen*. Ukazi za pripravo paketnih opravil za izvajanje pri tem upoštevajo tudi pogoje odvisnosti med opravili in so prikazani v prilogi 9.1. S tem postopkom se pripravijo zapisi v tabeli opravil za izvedbo naslednjega koraka.

4. Ponovno se izvede klic metode `serverGetOneTask()`, kot v prvem koraku. Opravilo, ki je v prejšnjem koraku prešlo v stanje *pripravljen*, se pri tem izbere in preda v izvajanje. Če še vedno ni nobenega ustreznega opravila, se postopek zaključi.



Slika 26: Postopek izbire opravila

Ob uspešnem izboru opravila jedro strežnika AOS odpre novo sejo, pod katero z uporabo metode `runJobStatic()` izvede kodo razreda X++, katerega identifikator in parametri so shranjeni v izbranem zapisu tabele opravila. Ob zaključku izvajanja se zapisu opravila nastavi ustrezen končen status z uporabo metode `serverFinishTask()`.

Ob zaključku izvajanja kode opravila se posodobi le status opravila. Zato je potrebno še posodobiti status obdelave. Status obdelave se lahko spremeni v končnega šele, ko so zaključena vsa opravila obdelave. Posodobitev statusa obdelave se zato izvede asinhrono s klicem metode `serverProcessFinishedJobs()`, ki ga vsakih 60 sekund izvede dodatna nit. Metoda poleg posodobitve statusa obdelav izvede še logiko, ki poskrbi za beleženje zgodovine izvajanja in za pripravo obvestil za uporabnike.

Tukaj je še tretja nit, ki vsakih 5 minut izvede klic metode `serverCleanUpDeadTasks()` in s tem poskrbi za opravila, ki se zaradi napake niso ustrezno zaključila. Pri tem se opravilom glede na aktivnost seje in nastavitve ponovljivosti nastavi ustrezen status.

Ker je za izvajanje paketnih obdelav pri posamezni postavitvi sistema lahko dodeljenih več instanc aplikacijskih strežnikov, je v izvajanje metod `serverProcessDependencies()`, `serverProcessFinishedJobs()` in `serverCleanUpDeadTasks()` vgrajena logika, ki preprečuje, da bi več procesov oz. niti sočasno in prepogosto izvajalo isto metodo. Vsaka od metod na začetku najprej preveri, če v posebni tabeli BatchGlobal obstaja zapis z identifikatorjem metode in časom zadnjega izvajanja, ki je ustrezno zapadel glede na periodo izvajanja. Če tak zapis obstaja, metoda na njem posodobi čas izvajanja in prične izvajati svojo proceduro. V nasprotnem primeru metoda opusti izvajanje.

5.2 Spremljanje obremenitev

Da bi razporejevalnik paketnih obdelav lahko odreagiral na visoko obremenitev vira, je potrebno najprej zagotoviti podatke o stanju vira. Za zbiranje le-teh se v praksi uporabljajo monitorji, ki omogočajo merjenje obremenitve sestavnih delov sistema [15]. Pri zasnovi programskega monitoringa za distribuiran sistem se srečamo z naslednjimi izzivi [6]:

- *Dodatne obremenitve*, ki jih izvajanje monitoringa predstavlja, bi morale biti čim manjše.
- *Razširljivost* v smislu zbiranja podatkov. Omogočena mora biti enostavna implementacija monitoringa za nove sistemske vire, ki jih na začetku pri zasnovi nismo predvideli.
- *Robustnost* monitoringa v primeru izpada posameznih komponent sistema.
- *Skalabilnost* – zasnova monitoringa mora upoštevati skalabilnost ciljnega sistema.
- *Obvladljivost* – zahtevnost administracije monitoringa ne sme rasti s številom komponent sistema.

Izvajanje programskega monitoringa za sistem predstavlja dodatno obremenitev (angl. *overhead*), kajti monitoring uporablja iste vire kot sistem sam. Tako bi moralo biti monitorsko breme za sistem čim manjše in s tem tudi vpliv na rezultate meritev minimalen [15]. Pri monitoringu se za posamezen vir sistema opravljata dve glavni operaciji. Prva je operacija senzorja, ki zajame podatke o stanju. Druga je operacija beleženje stanja vira, ki zabeleži podatke o stanju v takšni obliki, da je dostopna tudi ostalim delom sistema. Operacija senzorja je navadno nezahtevna in za izvajanje potrebuje zgolj minimalen procesorski čas. Zahtevnost beleženja stanja virov na drugi strani raste s časom izvajanja monitoringa in rastjo števila virov, ki jih spremljamo.

Pri beleženju stanja vira se soočimo z vprašanjem, ali je smiselno zabeležiti vsako meritev in ali vsaka meritev pomeni tudi spremembo stanja. Podatki meritev lahko vsebujejo šum in oscilacije, ki jih ne želimo upoštevati. Glede na to, kateri vir spremljamo in za kakšen namen spremljamo vir, pa je tudi odvisno, kako veliko odstopanje meritve pomeni spremembo stanja vira. Iz tega sledi, da je potrebno najti kompromis med točnostjo podatkov in zahtevnostjo beleženja stanja. Za izločanje odvečnih meritev pri periodičnem zajemanju je v [2, 6] predlagana uporaba pragu (angl. *threshold*).

V rešitvi je bilo uporabljeno izločanje z dinamičnim pragom *d* (1), ki ga algoritem izračuna na podlagi *N* preteklih objavljenih stanj A_i [2]. Pri tem je bila pri izvajanju meritev uporabljena nastavitev 10 preteklih stanj.

$$d = \frac{1}{N} \times \sum_{i=1}^{N} |A_i - A_{i-1}| \qquad (1)$$

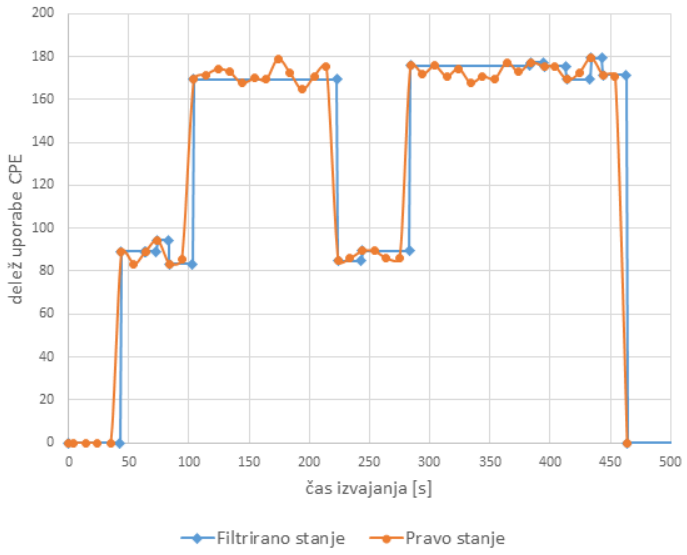
Zasnovan postopek zbiranja podatkov o virih tako vsebuje naslednje ključne lastnosti:

- Izločanje nepomembnih meritev z dinamičnim pragom zmanjšuje količino zabeleženih sprememb stanj. Tako se novo stanje objavi samo v primeru, če je razlika med novim in zadnjim objavljenim stanjem večja od izračunane vrednosti dinamičnega pragu *d*.

- Omejevanje dinamičnega pragu z nastavitvijo zgornje in spodnje meje. Z zgornjo mejo lahko preprečimo, da bi zaradi velikih sprememb prag postal prevelik in pričel izločati pomembne vrednosti. S spodnjo mejo pa lahko vsilimo izločanje nepomembnih sprememb v stanju (npr. sprememba zasedenosti pomnilnika za nekaj bajtov).
- Starost stanja odloča, ali bo meritev upoštevana pri izračunu dinamičnega pragu. Z nastavitvijo (npr. 120 sekund) določimo, kolikšna je največja starost stanja, ki je še lahko upoštevan pri izračunu pragu. S tem se prepreči vpliv prestarih podatkov.
- Predpomnjenje zadnjih *N* stanj v pomnilniku strežnika AOS odstranjuje potrebo po nalaganju stanj iz podatkovne baze ob vsakem izračunu dinamičnega pragu. Vsakič, ko se zabeleži novo stanje, se le-to tudi vrine na prvo mesto seznama v pomnilniku. Ob tem se zavrže zadnje stanje, če je število stanj v seznamu večje od *N*.

Navedene lastnosti omogočajo, da izvajanje monitoringa v čim manjši meri obremenjuje sistem. Z uporabo predpomnilnika zadnjih *N* stanj zmanjšamo število branj iz podatkovne baze. Pripravljen predpomnilnik stanj pri tem ni namenjen le izračunu dinamičnega pragu, ampak je uporabljen tudi v delu rešitve, ki kontrolira izvajanje paketnih obdelav. Uporaba izločanja z dinamičnim pragom zmanjša število pisanj v podatkovno bazo, ker se beležijo le pomembne spremembe v stanju vira. Učinkovitost uporabe izločanja je prikazana na naslednjih dveh primerih (podatki meritev so navedeni v prilogi 9.2):

- Primer A (slika 27): Od 48 meritev je sprejetih 15 meritev, kar je 69 % prihranek.
- Primer B (slika 28): Od 42 meritev je sprejetih 11 meritev, kar je 74 % prihranek.

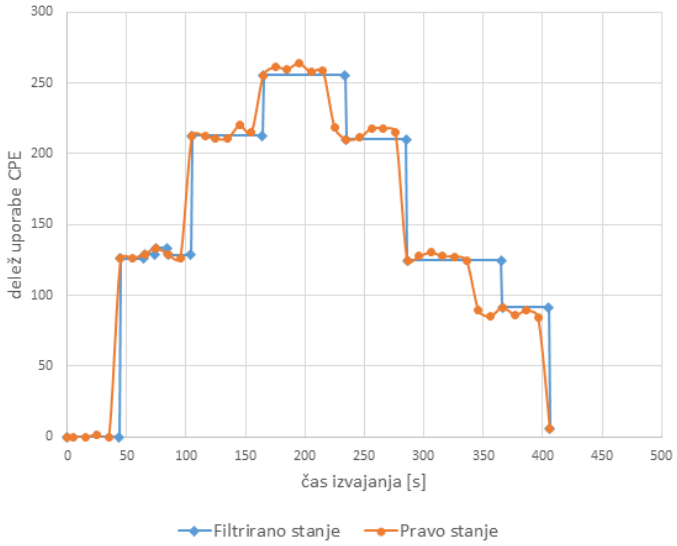


Slika 27: Izločanje meritev z dinamičnim pragom v primeru A

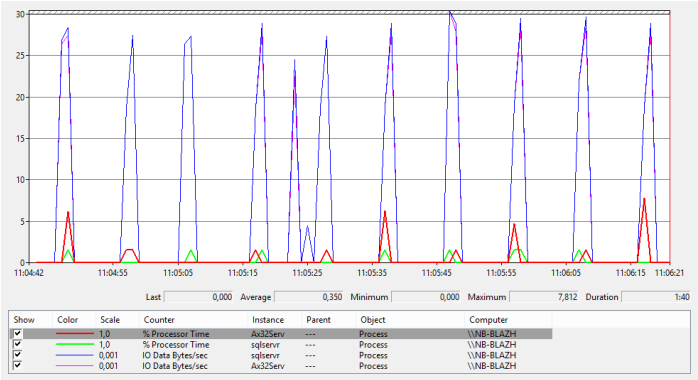
Pri izvajanju monitoringa na testnem okolju s periodo 10 sekund za 25 virov je bila izmerjena obremenitev, ki je navedena v spodnji tabeli (slika 29).

	Povprečna vrednost	Maksimalna vrednost
Delež uporabe CPE AOS	0,35	7,81
Delež uporabe CPE SQL	0,13	1,56
Komunikacija AOS – SQL (KB/s)	5,34	30,98

Tabela 6: Breme monitoringa



Slika 28: Izločanje meritev z dinamičnim pragom v primeru B



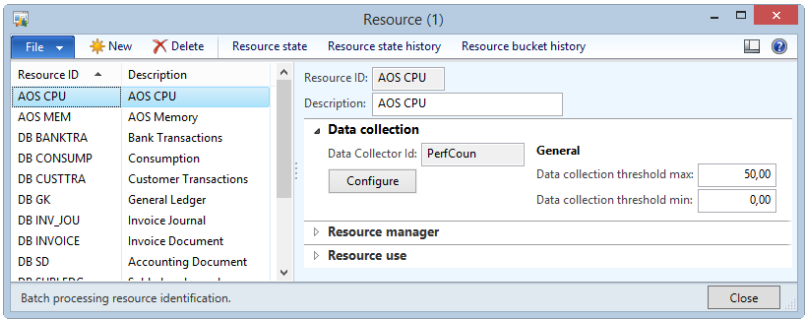
Slika 29: Breme monitoringa

Pri analizi virov sistema na testnem okolju je bilo ugotovljeno, da so za potrebe naloge zanimivi sistemski in podatkovni viri. Ker mora postopek zbiranja podatkov omogočati spremljanje različnih virov, je bil pri zasnovi programske kode uporabljen princip abstraktnega razreda. Implementacija le-tega rešuje problem zbiranja podatkov za posamezno vrsto vira. To omogoča enostavno implementacijo navedenih dveh vrst virov in možnost za kasnejšo razširljivost z novimi vrstami virov. Na zaslonski maski za urejanje vira nastavimo implementacijo zbiranja podatkov z nastavitvijo »*Data Collector Id*« (slika 30).

Abstraktni razred `ResourceDataCollectorBase` predvideva uporabo naslednjih metod:

- `createConfig` – Vsebuje kodo, ki ustvari zapis v tabeli za konfiguracijo zbiranja podatkov za posamezen vir. To je metoda, ki se izvede ob kreiranju zapisa za vir.
- `getConfigurationDisplayMenuItem` – Vsebuje kodo, ki vrne ime objekta za odpiranje zaslonske maske za vnos nastavitev za posamezen vir. To je metoda, ki jo pokliče zaslonska maska za urejanje vira, ko s klikom na gumb »*Configure*« (slika 30) želimo urejati nastavitve za posamezen vir. Metodo je potrebno implementirati.
- `collectData` – Vsebuje kodo, ki realizira zbiranje podatkov. To je metoda, ki jo periodično izvaja ogrodje za zbiranje podatkov. Metodo je potrebno implementirati.

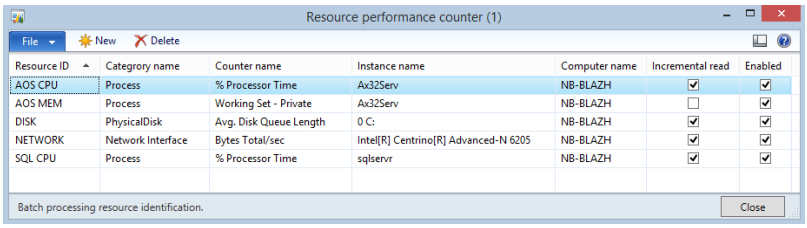
- `publishNewState` – Vsebuje kodo, ki z uporabo dinamičnega pragu odloči, ali naj se prebrano stanje uporabi. Metodo je potrebno uporabiti pri implementaciji metode `collectData`.



Slika 30: Zaslonska maska za urejanje vira – zbiranje podatkov

Kot je bilo povedano pri analizi sistemskih virov, programsko zbiranje podatkov o sistemskih virih omogoča uporaba razreda `PerformanceCounter`, ki je del ogrodja .NET. Uporaba razreda zahteva naslednje parametre [27], ki jih urejamo z uporabo zaslonske maske na sliki 31:

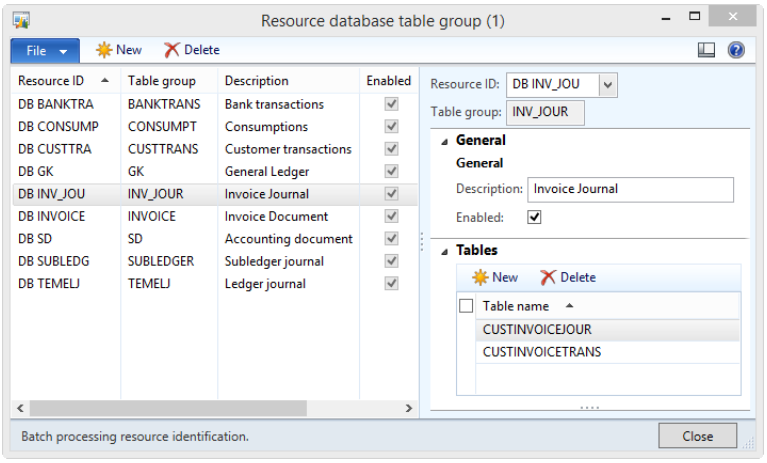
- **Ime kategorije** (angl. *category name*) določa kategorijo števcev, kot je proces, pomnilnik, trdi disk idr.
- **Ime števca** (angl. *counter name*) določa ime števca iz kategorije, kot je delež procesorskega časa, alociran pomnilnik, število zahtev za trdi disk idr.
- **Ime instance** (angl. *instance name*) določa instanco števca, ki meri določen proces, trdi disk, omrežni vmesnik idr.
- **Ime računalnika** (angl. *computer name*) določa omrežno ime računalnika, na katerem se instanca števca nahaja.



Slika 31: Zaslonska maska za nastavitev performančega števca

Posebnost pri uporabi razreda za branje vrednosti števcev je, da je potrebno pri določenih števcih izvesti dve zaporedni branji. Določeni števci namreč potrebujejo referenčno vrednost. Zato pri prvem branju vedno vrnejo vrednost 0 in nato pri drugem pravo vrednost. Pri zajemu vrednosti števcev z nastavitvijo »*Incremental read*« implementirana logika najprej izvede referenčno branje in nato čez 1 sekundo izvede dejansko branje števcev.

Stanje podatkovnih virov je bilo pri analizi definirano kot skupno število zaklepanj na podatkovnih tabelah, ki pripadajo določeni skupini. Zaslonska maska za nastavitev skupine tabel posameznega podatkovnega vira je prikazana na sliki 32.



Slika 32: Zaslonska maska za nastavitev skupine tabel

Do podatkov o zaklepanjih na podatkovnih tabelah podatkovnega strežnika SQL Server 2012 lahko dostopamo s poizvedovanjem sistemskega pogleda `sys.dm_tran_locks` [29]. Poizvedba, s katero je implementiran zajem podatkov o zaklepanjih, je prikazana na sliki 33. Navedena poizvedba vrne za vsako skupino tabel število zaklepanj, ki nakazujejo spremembo podatkov in omejujejo dostop ostalim transakcijam [22].

```
WITH
Locks (TableName)
AS
(
    SELECT
        CASE
            WHEN resource_type = 'OBJECT'
                THEN OBJECT_NAME(dm_tran_locks.resource_associated_entity_id, dm_tran_locks.resource_database_id)
            ELSE OBJECT_NAME(partitions.OBJECT_ID, dm_tran_locks.resource_database_id)
        END AS TableName
    FROM sys.dm_tran_locks
    LEFT JOIN sys.partitions ON partitions.hobt_id = dm_tran_locks.resource_associated_entity_id
    LEFT JOIN sys.indexes ON indexes.OBJECT_ID = partitions.OBJECT_ID AND indexes.index_id = partitions.index_id
    WHERE dm_tran_locks.resource_associated_entity_id > 0
        AND dm_tran_locks.resource_database_id = DB_ID()
        AND dm_tran_locks.request_mode IN ('X', 'IX', 'U', 'IU')
),
GroupLocks (TableGroupId, lockcnt)
AS
(
SELECT TableGroupId, count(*) lockcnt FROM
    AdBpResourceDbTable DbTable INNER JOIN Locks ON DbTable.TableName = Locks.TableName
    GROUP BY DbTable.TableGroupId
)

SELECT DbTableGroup.TableGroupId, ISNULL(GroupLocks.lockcnt, 0) AS lockcnt FROM
    AdBpResourceDbTableGroup DbTableGroup LEFT JOIN GroupLocks ON DbTableGroup.TableGroupId = GroupLocks.TableGroupId
    WHERE DbTableGroup.Enabled=1;
```

Slika 33: Poizvedba SQL za pridobitev podatkov o zaklepanjih

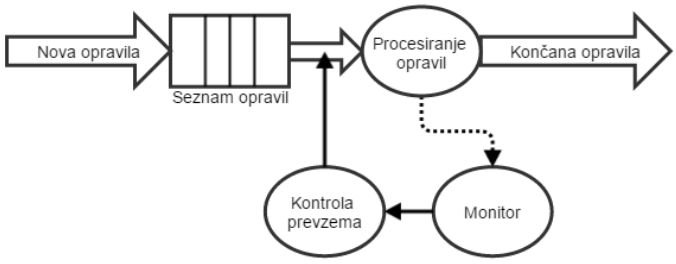
Koda, ki izvaja zbiranje podatkov, se izvaja na strežniku AOS. V primeru izpada le-tega pride do prekinitve zbiranja podatkov. Ker pa je koda realizirana kot paketna obdelava, lahko izvajanje prevzame druga instanca strežnika AOS, ki je tudi dodeljena za izvajanje paketnih obdelav. Tako je monitoring robusten na izpade strežnikov AOS. S stališča skalabilnosti je omogočeno dodajanje novih virov, ki bi opisovali stanje na novo dodanih komponent (npr. dodatna instanca strežnika AOS). V primeru, da bi zaradi prevelikega števila virov čas zajema podatkov presegel periodo zajemanja podatkov, pa bi bilo potrebno paralelizirati postopek zbiranja podatkov. V rešitvi je zbiranje podatkov implementirano kot zaporeden postopek, ki za 25 virov porabi 1056 ms. Pri tem je 1000 ms namenjenih premoru med zaporednimi branji performančnih števcev. Iz tega sledi, da postopek zbiranja podatkov za vsak dodaten vir potrebuje dodatni 2 do 3 ms. S stališča enostavnosti obvladovanja monitoringa je bil v okviru magistrske naloge razvit zgolj uporabniški vmesnik za potrebe konfiguracije testnega okolja.

Smiselno pa bi bilo pripraviti ustrezne skripte oz. predloge, ki bi omogočale tipsko dodajanje virov (npr. viri nove instance strežnika AOS).

5.3 Kontrola izvajanja paketnih obdelav

V standardni rešitvi algoritem za razporejanje opravil paketnih obdelav izbira opravila za izvajanje, tako da vsaka prosta izvajalna nit vzame naslednje opravilo iz seznama. Pri tem se algoritem ne zaveda, da je določen vir sistema pod visoko obremenitvijo in da bo izvajanje dodatnega opravila lahko povzročilo preobremenitev. Namen naloge je zasnovati razširitev, ki bo v algoritem vgradila kontrolo, ki bo upoštevala stanje virov sistema AX.

Za zagotavljanje zadostne razpoložljivosti sistemskih virov pri razporejanju opravil se uporabljajo različni pristopi [1, 4, 5, 12, 13, 14]. Le-ti predvidevajo rezervacijo virov, višanje prioritet opravilom, prerazporejanje opravil, prekinitev opravila, prekinitev opravila z nadaljevanjem, začasni zakup dodatnih zmogljivosti v oblaku, kontrola sprejema opravil idr. Postavitev računalniškega sistema, katerega zmogljivost bi zadoščala vsem obremenitvenim situacijam, je običajno predraga [5, 12]. V skupkih računalnikov se zato uporablja rezervacija virov v kombinaciji s tehniko razporejanja, ki omogoča normalno uporabo rezerviranega vira z možnostjo sprostitve vira za potrebe izvajanja opravila, kateremu je rezervacija namenjena [1, 5]. Takšnemu opravilu se s prerazporeditvijo na prvo mesto ali višanjem prioritete zagotovi najhitrejši možni začetek izvajanja. Od narave samega sistema pa je odvisno, ali je možno manj pomembno opravilo v času izvajanja prekiniti in s tem takoj sprostiti vir. Prekinitev opravila (angl. *kill*) pri tem pomeni zavrženo delo, v kolikor ni omogočeno kasnejše nadaljevanje prekinjenega opravila (angl. *suspend and resume*). Pri določenih sistemih se lahko problem začasnega pomanjkanja razpoložljivosti virov sistema rešuje tudi z začasnim zakupom dodatnih zmogljivosti v oblaku [14]. V tem primeru mora algoritem za razporejanje čim bolje oceniti lastnosti opravil in na podlagi ocen optimizirati izkoriščenost prvotnega sistema ter minimizirati potrebo po uporabi dodatnih zmogljivosti, ki so plačljive. Eden od pristopov je tudi uporaba kontrole sprejema opravil v izvajanje (angl. *admission control*) [4, 12, 13]. Le-ta izkorišča informacije o opravilu in povratne informacije o izrabi virov za izogibanje preobremenitvam. Kontrola tako začasno zadrži opravilo pred predajo v izvajanje, če potrebni viri niso razpoložljivi. Pri tem pristopu ne pride do zavrženega dela, ker se opravilo še ni pričelo izvajati. V izvajanje pa se lahko preda opravilo, za katerega je razpoložljivost virov zadostna.



Slika 34: Arhitektura kontrole prevzema

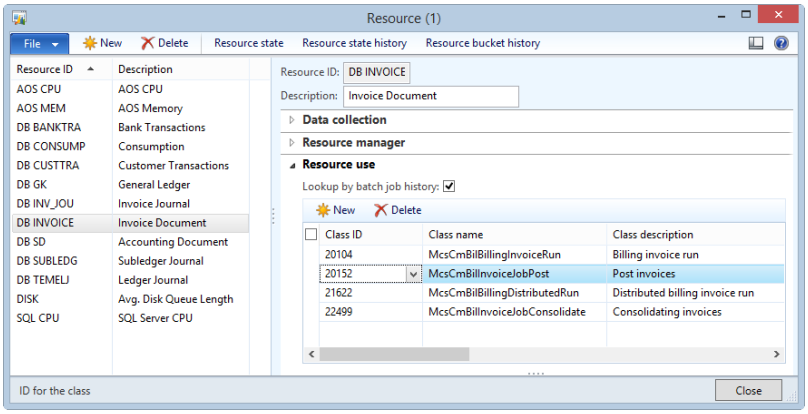
V rešitvi je bil za obvladovanje preobremenitev virov uporabljen pristop kontrole sprejema opravil v izvajanje (slika 34). Razlogi za uporabo tega pristopa so naslednji:

- Na podlagi analize bremena imamo informacijo za posamezno opravilo, katere vire potrebuje za izvajanje.
- Zasnovan monitoring zagotavlja povratno informacijo o izrabi virov.
- Pristop zahteva minimalni poseg v standardni algoritem za razporejanja. Potrebna je le dopolnitev poizvedbe za izbiro opravila.
- Obseg dodatnega procesiranja pri prevzemu opravila je minimalen.
- Ker gre za sprotno razporejanje, ni popolnih informacij o opravilih tako v časovnem smislu kot v smislu izrabe virov. Za določeno opravilo je na voljo le informacija, katere vire potrebuje. V kakšni meri, koliko časa in kdaj bo koristil vire pa je odvisno od samih parametrov in podatkov. Zato vnaprejšnja optimizacija plana ni smiselna.
- Zaradi pomanjkanja informacije, katero opravilo v danem trenutku povzroča preobremenitev določenega vira, uporaba popolne prekinitve opravila ni smiselna. Prav tako je nezaželeno zavreči delo, ki ga je opravilo v izvajanju že opravilo. Prekinjanje opravil s kasnejšim nadaljevanjem pa v sistemu ni podprto.

Pri zasnovi kontrole prevzema je bilo potrebno rešiti tri probleme:

- Kako prepoznati posamezno paketno obdelavo?
- Kako realizirati kontrolo prevzema?
- Kako vgraditi kontrolo prevzema v obstoječi algoritem?

Ker različne paketne obdelave obremenjujejo različne vire sistema AX, jih moramo razlikovati. Namreč če je določen vir preobremenjen, je potrebno preprečiti prevzem le tistim opravilom, ki bi povzročila dodatno obremenitev. Ker je vsaka paketna obdelava in s tem tudi pripadajoče opravilo realizirano kot razred X++, ga lahko prepoznamo po identifikatorju razreda (angl. *Class ID*). Le-ta je tudi shranjen v zapisu paketnega opravila v tabeli Batch. Za vsak vir je tako potrebno navesti seznam razredov, ki implementirajo paketne obdelave, ki ta vir obremenjujejo. Na zaslonski maski za urejanje vira navedemo seznam paketnih obdelav v razdelku »*Resource use*« (slika 35).

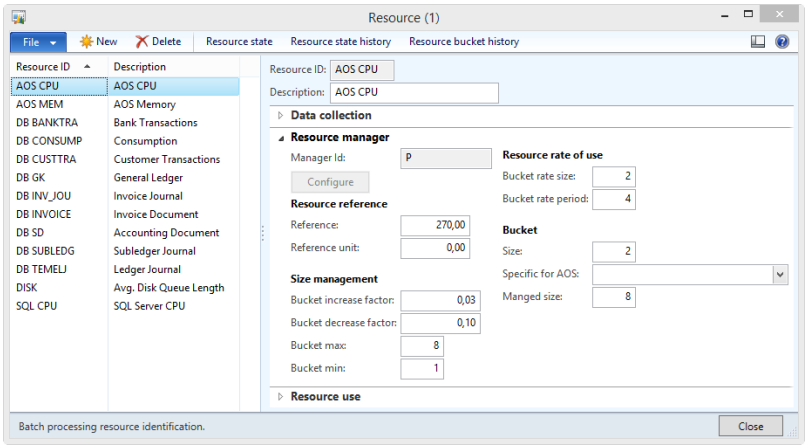


Slika 35: Zaslonska maska za urejanje vira – seznam paketnih obdelav

Pri realizaciji kontrole prevzema je bil uporabljen koncept vedra (angl. *bucket*). Pri tem ima vsak vir svoje vedro, katerega velikost določa, koliko opravil, ki so navedena na seznamu, se lahko istočasno izvaja. Tako se lahko novo opravilo prevzame v izvajanje le, če je velikost vedra večja od števila opravil, ki se v danem trenutku že izvajajo.

Velikost vedra je dinamična in se s časom spreminja z uporabo kontrolne funkcije. Le-to lahko nastavimo na zaslonski maski za urejanje vira v razdelku »*Resource manager*« (slika 36) z naslednjimi parametri:

- Kontrolna funkcija (angl. *Manager Id*). Identifikator implementacije kontrolne funkcije.
- Referenca - *R* (angl. *Reference*). Podaja željeno stanje vira.
- Enota reference – *RU* (angl. *Reference Unit*). V kolikor je nastavljena, se uporabi za normalizacijo vrednosti (npr. vrednost pomnilnika podano v B normaliziramo na vrednost 100 MB).
- Faktor povečanja velikosti – K_{inc} (*angl. Bucket increase factor*).
- Faktor manjšanja velikosti – K_{dec} (*angl. Bucket decrease factor*).
- Maksimalna velikost – S_{max} (*angl. Bucket max*).
- Minimalna velikost – S_{min} (*angl. Bucket min*).
- Stopnja propustnosti – R_s (*angl. Bucket rate size*). Določa, za koliko se lahko poveča velikost glede na trenutno število opravil v izvajanju.
- Perioda propustnosti – R_P (*angl. Bucket rate period*). Določa, koliko ciklov posodabljanja velikosti mora preteči pred ponovnim povečanjem velikosti.
- **Kontrolirana velikost – S_m (*angl. Managed size*).** Predstavlja izračunano maksimalno velikost glede na stanje vira.
- **Velikost – *S* (*angl. Size*)**. Predstavlja velikost, ki se izračuna glede na kontrolirano velikost in stopnjo propustnosti. To je vrednost, ki se uporabi za omejevanje števila opravil.



Slika 36: Zaslonska maska za urejanje vira – kontrola izvajanja opravil

Pri zasnovi postopka izračuna kontrolne funkcije je bil uporabljen princip abstraktnega razreda `ResourceManagerBase`. Zato je prvi od parametrov izbor implementacije le-tega. Abstraktni razred predvideva uporabo naslednjih metod:

- `createConfig` – Vsebuje kodo, ki ustvari zapis v tabeli za konfiguracijo kontrolne funkcije za posamezen vir. To je metoda, ki se izvede ob kreiranju zapisa za vir.
- `getConfigurationDisplayMenuItem` – Vsebuje kodo, ki vrne ime objekta za odpiranje zaslonske maske za vnos nastavitev za posamezen vir. To je metoda, ki jo pokliče zaslonska maska za urejanje vira, ko s klikom na gumb »Configure« (slika 36) želimo urejati nastavitve za posamezen vir. Metodo je potrebno implementirati.
- `manageBucket` – Vsebuje kodo, ki realizira izračun nove velikosti vedra. To je metoda, ki jo periodično izvaja ogrodje za posodabljanje velikosti veder. Metodo je potrebno implementirati.

Za potrebe izvedbe naloge je bila implementirana kontrolna funkcija, ki izračuna novo velikost vedra po postopku, ki je opisan z enačbami (2–9). Pri tem so uporabljene naslednje dodatne spremenljivke:

- *t* – čas,
- *e* – razlika med referenco in trenutnim stanjem vira,
- *g* – razlika med trenutno velikostjo in dovoljeno velikostjo glede na stopnjo propustnosti,
- *p* – število ciklov od zadnje spremembe velikosti glede na stopnjo propustnosti oz. izpolnitve prvega pogoja enačbe (7),
- *Q* – stanje vira,
- *B* – število opravil v izvajanju.

$$e(t) = \frac{R - Q(t)}{RU} \tag{2}$$

$$S''_m(t+1) = \begin{cases} S_m(t) + e(t) \times K_{inc}, & e(t) > 0 \\ S_m(t) + e(t) \times K_{dec}, & e(t) \leq 0 \end{cases} \tag{3}$$

$$S'_m(t+1) = \min(S''_m(t+1), S_{max}) \tag{4}$$

$$\boldsymbol{S_m(t+1) = \max(S'_m(t+1), S_{min})} \tag{5}$$

$$g(t) = \mathrm{B}(t) + R_S - S(t) \tag{6}$$

$$S''(t+1) = \begin{cases} S(t) + g(t), & (g(t) > 0 \wedge p \geq R_P) \vee g(t) < 0 \\ S(t), & (g(t) > 0 \wedge p < R_P) \vee g(t) = 0 \end{cases} \tag{7}$$

$$S'(t+1) = \min(S''(t+1), S_m(t+1)) \tag{8}$$

$$\boldsymbol{S(t+1) = \max(S'(t+1), S_{min})} \tag{9}$$

Uporaba dveh različnih faktorjev v enačbi (3) omogoča, da v primeru prekoračitve dovoljene vrednosti kontrola hitreje zmanjša velikost vedra in s tem prepreči sprejem novih opravil. Uporaba stopnje propustnosti v enačbi (7) omogoča postopno sprejemanje novih opravil in vmesno preverjanje povratne informacije o stanju vira. To je še posebej dobrodošlo, ker informacija o intenziteti obremenitve za posamezno opravilo ni znana vnaprej.

Iz opisa delovanja standardne rešitve je razvidno, da je izbor opravila določen s poizvedbo (slika 24). Zato je bila kontrola prevzema vgrajena kot razširitev pogoja le-te. Pri tem je bil najprej pripravljen poseben pogled, ki za določen razred vrne rezultat oz. vrstico za vsak vir, katerega vedro je polno. Tako pogled ne vrne rezultata, če so vsi viri prosti. Pogled vrne ustrezen rezultat s poizvedovanjem po podatkovnih tabelah na naslednji način:

1) Vsako izvajajoče opravilo ima v tabeli opravil shranjen identifikator razreda in status v izvajanju.
2) Za vsak vir so v tabeli seznama paketnih obdelav shranjeni identifikatorji razredov, ki vir obremenjujejo. To pomeni, da lahko tudi obratno na podlagi istih podatkov pridobimo seznam virov, ki jih določen razred potrebuje oz. obremenjuje.
3) Za vsak vir je v tabeli vedra shranjena trenutna izračunana vrednost velikosti *S*.
4) Če staknemo (angl. *join*) rezultat točke 1 in 2 po identifikatorju razreda in preštejemo vrstice, dobimo podatek *B*, ki pove koliko opravil koristi določen vir.
5) Če staknemo točki 3 in 4 po identifikatorju vira in izračunamo razliko $S - B$, dobimo podatek *P*, ki pove, ali je določen vir prost ($P > 0$) glede na kontrolo prevzema.
6) Če staknemo rezultat točk 2 in 5 po identifikatorju vira in omejimo zapise s pogojem $P \leq 0$, dobimo seznam identifikatorjev razredov, ki jih ne smemo prevzeti v izvajanje.

Obstoječa poizvedba je bila nato razširjena s stikom pripravljenega pogleda z uporabo pogoja ne obstoja (angl. *not exists*). Razširjena poizvedba za izbor opravila je prikazana na sliki 37, kjer je tudi poudarjena vključitev novega pogoja.

```
select firstonly pessimisticlock RecId, CreatedBy, ExecutedBy, StartDateTime, Status,
    SessionIdx,SessionLoginDateTime, Company, ServerId
    ,DataPartition
from batch
where batch.Status == BatchStatus::Ready
&& batch.RunType == BatchRunType::Server
&& (Session::isServer() || batch.CreatedBy == user)
&& batch.DataPartition == partitions.PartitionKey
join Language from userInfo
order by batch.Priority
    where userInfo.Id == batch.CreatedBy
    && userInfo.Enable == true
    && userInfo.Partition == partitions.RecId
exists join batchServerGroup
    where batchServerGroup.ServerId == serverId
        && batch.GroupId == batchServerGroup.GroupId
notExists join resourceBusyView
    where batch.ClassNumber == resourceBusyView.ClassNumber
        && (resourceBusyView.ServerId == '' || resourceBusyView.ServerId == serverId);
```

Slika 37: Poizvedba za izbor opravila s kontrolo prevzema

Iz poizvedbe je razvidno, da pogoj stika poleg polja identifikatorja razreda ClassNumber vključuje tudi polje identifikatorja strežnika ServerId. Razlog zato je možnost upoštevanja določenega vira le, ko prevzema opravilo izvajalna nit določenega strežnika AOS. V primeru

več strežnikov tako izvajalna nit strežnika z neobremenjenim procesorjem lahko prevzame opravilo, medtem ko ga izvajalna nit strežnika z obremenjenim procesorjem ne sme. S tem pa je bil v procesiranje paketnih obdelav vpeljan tudi koncept izenačevanja obremenitve (angl. *load balancing*) med strežniki AOS, ki so dodeljeni za izvajanje paketnih obdelav. Vedro posameznega vira lahko omejimo na posamezen strežnik z nastavitvijo »*Specific for AOS*« (slika 36).

5.4 Postavitev rešitve

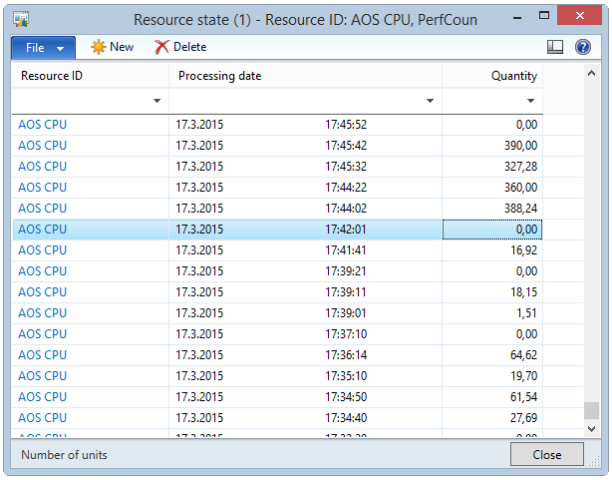
Rešitev za obvladovanje preobremenitve virov sistema AX je bila vgrajena v obstoječi sistem v obliki dveh procesov.

Prvi proces skrbi za posodabljanje virov in je realiziran kot paketna obdelava, ki v zanki zaporedoma izvede najprej posodobitev stanja za vse vire in nato posodobitev velikosti vedra za vse vire. Velikost vedra se izračuna nad svežimi podatki, ker se vedno predhodno izvede posodobitev stanja vira. V izogib prepogostemu posodabljanju virov gre proces po vsaki posodobitvi v spanje za določen čas. Tako lahko nastavimo, da se posodabljanje izvaja z določeno periodo (npr. vsakih 10 sekund). Ker je posodabljanje virov realizirano kot paketna obdelava, za njeno izvajanje potrebujemo dodatno izvajalno nit. V primeru izpada trenutne instance strežnika AOS pa bo izvajanje obdelave samodejno prevzela ena od preostalih aktivnih instanc.

Drugi proces je obstoječi proces razporejanja opravil paketnih obdelav, kjer je bila v razredu `BatchRun` prilagojena metoda `serverGetOneTask()`, ki skrbi za izbor pripravljenih opravil. Pri tem je bila poizvedba za izbor opravila dopolnjena s pogojem kontrole prevzema, ki upošteva stanje virov, ki jih posamezno opravilo potrebuje za izvajanje. V razredu `BatchRun` je bila prilagojena tudi metoda `serverCleanUpDeadTasks()`, ki jo vsakih 5 minut izvede eden od strežnikov, dodeljenih za izvajanje paketnih obdelav. Le-ta je bila uporabljena za izvedbo stražnega mehanizma (angl. *watchdog*), ki preverja, ali se paketna obdelava za posodabljanje virov izvaja. Če se paketna obdelava ne izvaja, mehanizem najprej vzpostavi vse potrebne nastavitve (vzpostavi paketno skupino in jo dodeli strežnikom) in na novo aktivira obdelavo.

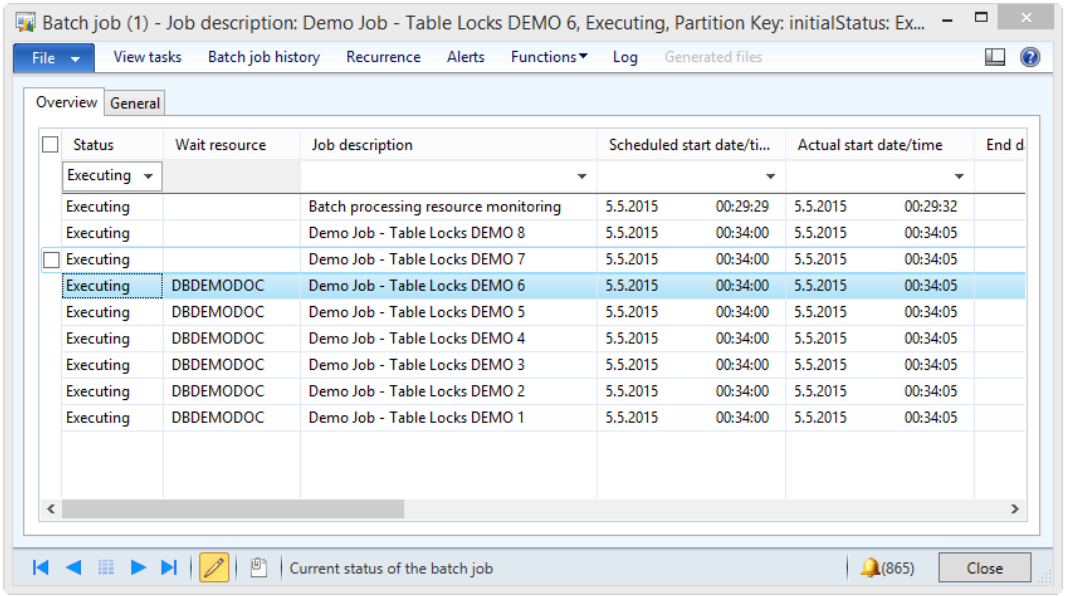
Kot je bilo predstavljeno v prejšnjih dveh razdelkih, je rešitev vzpostavljena kot razširljivo ogrodje, ki omogoča implementacijo novih načinov zbiranja podatkov o stanju vira (npr. obstoj datoteke na omrežni lokacij) in novih kontrolnih funkcij (npr. funkcija, ki bi upoštevala trend iz preteklih stanj).

Pripravljen uporabniški vmesnik za urejanje nastavitev posameznih virov sistema je bil predstavljen v prejšnjih dveh razdelkih. Za spremljanje podatkov o obremenitvi virov je bila za namen naloge izdelana zgolj programska maska, ki tabelira vrednosti (slika 38). Pri tem je vredno omeniti, da ni smiselno razvijati uporabniškega vmesnika za analizo podatkov, ker za ta namen že obstajajo orodja. Uporabnik AX-a se lahko z Excel-om z uporabo vtičnika za AX direktno poveže s strežnikom AOS in od njega zahteva podatke, ki jih nato obdela in vizualizira [31].



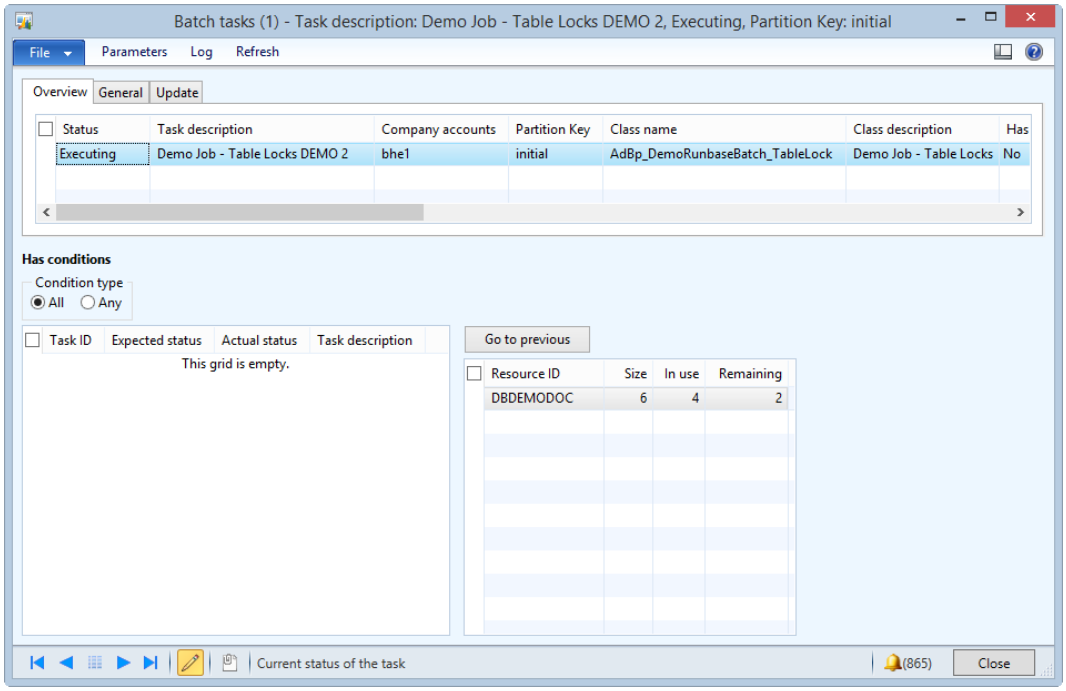
Slika 38: Zaslonska maska za pregled sprememb stanja vira

Od obstoječega uporabniškega vmesnika sta bili prilagojeni dve zaslonski maski. Pri tem je bila zaslonska maska za pregled paketnih obdelav dopolnjena z dodatnim stolpcem »*Wait resource*« (slika 39), ki uporabniku poda informacijo, ali opravila obdelave čakajo, da se kateri od virov sprosti.



Slika 39: Zaslonska maska za pregled paketnih obdelav – čakanje na vir

Zaslonska maska za urejanje paketne obdelave je bila dopolnjena s seznamom virov, ki jih izbrano opravilo potrebuje za izvajanje. Pri tem je za vsak vir navedena velikost vedra, koliko opravil koristi vir in za koliko opravil je vir še prost (slika 40).

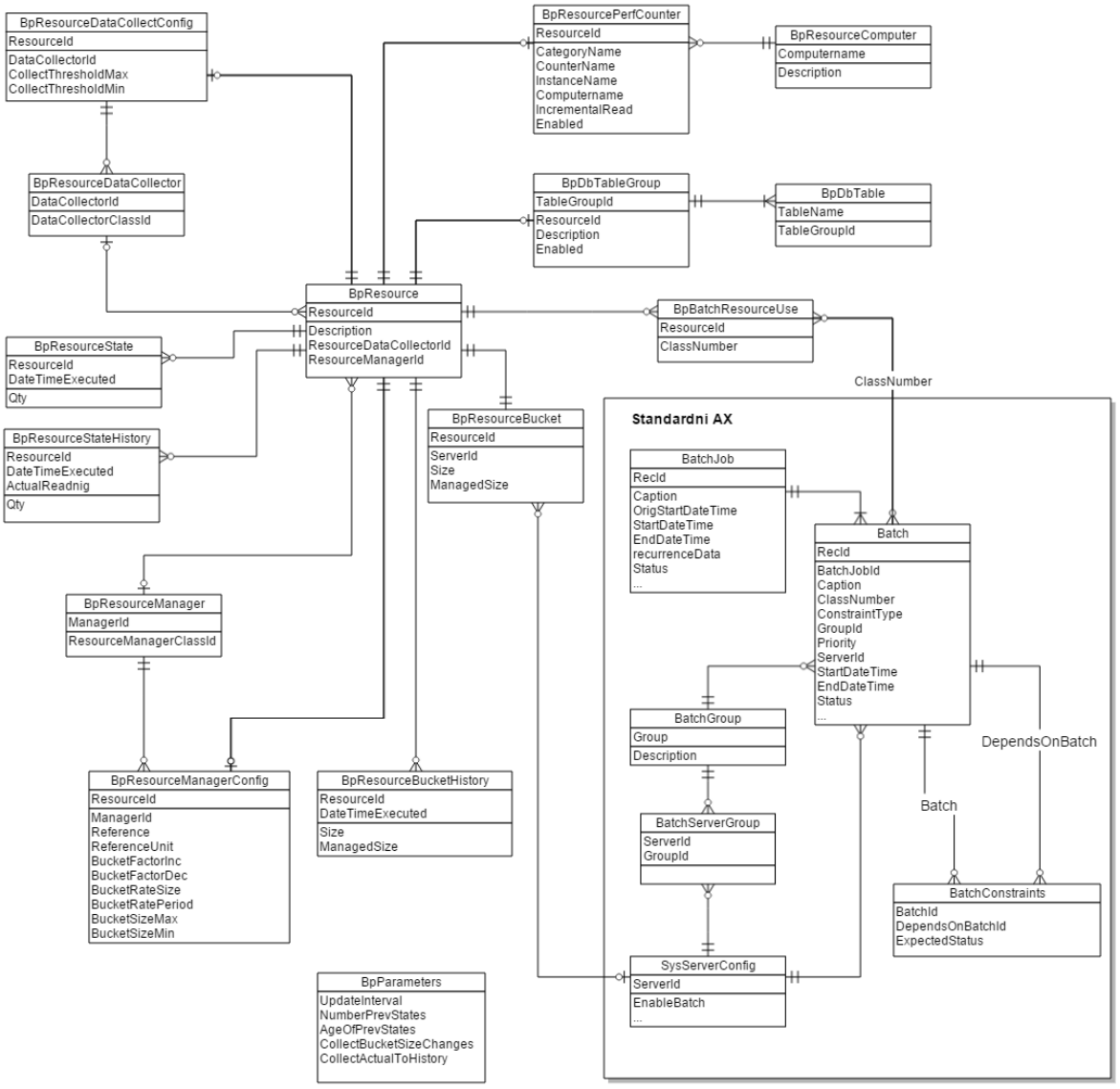


Slika 40: Zaslonska maska za urejanje paketne obdelave – dopolnitev s seznamom virov

Na sliki 41 je prikazan podatkovni model rešitve, ki vsebuje naslednje tabele:

- **BpResource:** vsebuje seznam virov sistema.
- **BpResourceDataCollectConfig:** vsebuje nastavitve za izbrano implementacijo zbiranja podatkov.
- **BpResourceDataManagerConfig:** vsebuje nastavitve za izbrano implementacijo kontrolne funkcije za izračun velikosti vedra.
- **BpResourceBucket:** vsebuje podatke o kontroli sprejema oz. velikosti vedra.
- **BpBatchResourceUse:** vsebuje seznam paketnih obdelav, ki obremenjujejo vir.
- **BpResourcePerfCounter:** vsebuje nastavitve za zbiranje podatkov z uporabo performančnega števca.
- **BpResourceComputer:** vsebuje seznam računalnikov, na katerih lahko spremljamo performančne števce.
- **BpDbTableGroup:** vsebuje seznam skupin tabel.
- **BpDbTable:** vsebuje seznam podatkovnih tabel, ki so v določeni skupini.
- **BpResourceState:** vsebuje seznam zabeleženih stanj vira.
- **BpResourceStateHistory:** vsebuje zgodovino stanj vira. Tabela se uporablja za arhiviranje podatkov. Če v parametrih omogočimo možnost »Collect actual *states*«, se v tabelo beležijo tudi stanja, ki bi bila izločena z uporabo dinamičnega pragu. Nastavitev je bila uporabljena za preverjanje pravilnosti algoritma.
- **BpResourceBucketHistory:** vsebuje zgodovino sprememb kontrole sprejema. Spremembe se beležijo, če v parametrih omogočimo možnost »*Collect bucket size changes*«. Nastavitev je bila uporabljena za preverjanje pravilnosti algoritma.

- **BpResourceDataCollector:** vsebuje seznam implementacij zbiranja podatkov. V tabeli so shranjeni identifikatorji razredov, ki so razširitve abstraktnega razreda `ResourceDataCollectorBase`.
- **BpResourceManager:** vsebuje seznam implementacij kontrolne funkcije. V tabeli so shranjeni identifikatorji razredov, ki so razširitve abstraktnega razreda `ResourceManagerBase`.
- **BpParameters:** vsebuje zapis s parametri, kot so: interval posodabljanja stanj, število prejšnjih stanj za izračun dinamičnega pragu, največja starost prejšnjih stanj za izračun dinamičnega pragu, možnost beleženja vseh sprememb stanj in možnost beleženja sprememb velikosti vedra.



Slika 41: Podatkovni model rešitve za obvladovanje preobremenitev virov

6 Primeri uporabe

V tem poglavju bo prikazano delovanje rešitve ob obremenitvi testnega okolja z umetnim bremenom, ki je bilo opredeljeno v 4. poglavju. Pri tem je glavni namen izvedbe simulacij pokazati, kako dobro lahko rešitev obvladuje obremenitve posameznih virov glede na podano referenčno mejo in uporabo sistema brez obvladovanja preobremenitev. V središču opazovanja bo čas prekoračitve referenčne meje glede na velikostne razrede obremenitve in obnašanje algoritma kontrole sprejema.

V nadaljevanju bo prikazana uporaba rešitve na naslednjih primerih:

- Simulacija obremenitve sistemskega vira CPE z bremenom, sestavljenim iz generičnih paketnih obdelav.
- Simulacija obremenitve podatkovnega vira testnega dokumenta z bremenom, sestavljenim iz generičnih paketnih obdelav.
- Uporaba bremena, sestavljenega iz realnih paketnih obdelav, ki obremenjujejo več različnih virov sistema AX.

6.1 Simulacija obremenitve sistemskega vira

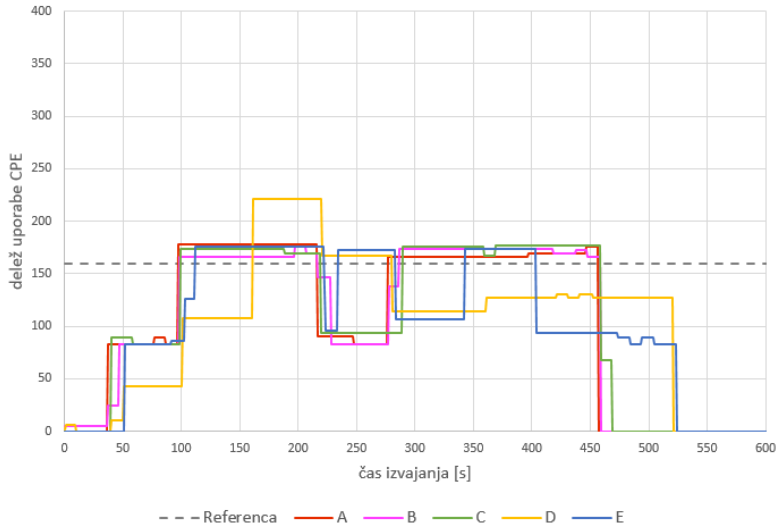
Pri simulaciji obremenitve sistemskega vira je bil uporabljen vir CPE AOS, ki temelji na performančnem števcu, ki spremlja delež uporabljenega procesorskega časa s strani procesa strežnika AOS. Pri tem so bili uporabljeni naslednji parametri:

- referenca 160 %;
- faktor povečanja velikosti 0,05;
- faktor manjšanja velikosti 0,10;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 1 obdelava;
- maksimalni dinamični prag 50 %.

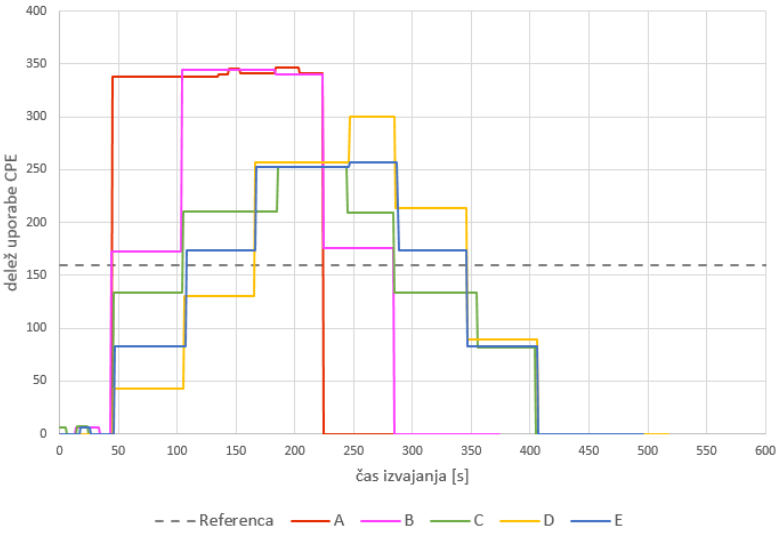
Kot breme so bile uporabljene generične paketne obdelave za obremenjevanje CPE v pripravljenih razporeditvah A, B, C, D in E. Vsaka od osmih paketnih obdelav je bila pri tem parametrizirana tako, da se je izvajala 180 sekund in ob tem porabljala približno 40 % procesorskega časa.

Na slikah 42 in 43 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz tabele 7, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema uspešno vzdržuje nizko prekoračitev. Čas prekoračitve pri tem skoraj v celoti pade v razreda nizke obremenitve, tj. od 160 % do 170 % in od 170 % do 190 %. Izjema je pri razporeditvi D, kjer obremenitev za čas 60 s preseže 190 %. Pri tem je iz maksimalne vrednosti

razvidno, da obremenitev ne preseže 222 %. Iz podatkov je tudi razvidno, da v primeru brez kontrole prevzema obremenitve v večjem delu časa presežejo 190%.



Slika 42: Simulacija obremenitve CPE s kontrolo prevzema



Slika 43: Simulacija obremenitve CPE brez kontrole prevzema

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 160–170 %**		**Čas prekoračitve 170–190 %**		**Čas prekoračitve nad 190 %**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	178,56	346,96	11,95	180,67	170		130			179
B	175,39	344,61	10,47	98,11	140		151	121		120
C	177,26	252,30	14,65	64,04	41		250			180
D	221,53	299,99	34,61	91,68	60				60	181
E	175,38	256,92	14,47	54,60			222	118		121
Povp.	185,62	300,16	17,23	97,82	82,20	0,00	150,60	47,80	12,00	156,20

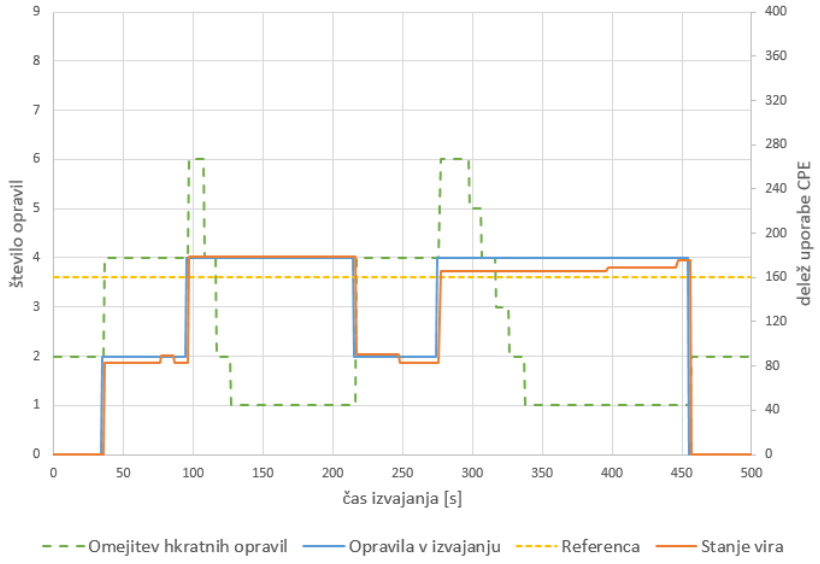
Tabela 7: Podatki simulacije obremenitve CPE – prekoračitve

V tabeli 8 so navedeni časi, ki so bili potrebni za izvedbo vseh opravil. Ker kontrola prevzema zadrži opravila, ko obremenitev preseže referenčno mejo, je potrebno več časa (v povprečju 144 sekund) za izvedbo vseh opravil. Le-to je tudi cena, ki jo je potrebno plačati, da se izognemo previsoki obremenitvi vira.

Razporeditev	A	B	C	D	E	Povprečje
Kontrola DA	420	420	421	480	480	444,20
Kontrola NE	180	241	360	360	360	300,20
Razlika	240	179	61	120	120	144,00

Tabela 8: Podatki simulacije obremenitve CPE – čas izvajanja

Iz primera uporabe razporeditve A (slika 44) je razvidno, da je hitrost zmanjševanja omejitve hkratnih opravil oz. velikosti vedra odvisna od velikosti prekoračitev. Prekoračitev v 95. sekundi je večja kot v 275. sekundi in posledično kontrolna funcija ob vsaki posodobitvi za večji del zmanjša velikost vedra. Reakcija kontrole na prekoračitev je pri 10-sekundnem posodobljanju ob uporabi dovolj velikega faktorja manjšanja velikosti vedra dovolj hitra, ker proste izvajalne niti prevzemajo nova opravila vsakih 60 sekund. Prevzemanje novih opravil vsakih 60 sekund je tudi razlog za 35-sekundno zakasnitev časa začetka opravil glede na čas razporeditve, ki je razvidna iz podatkov o opravilih v tabeli 9.

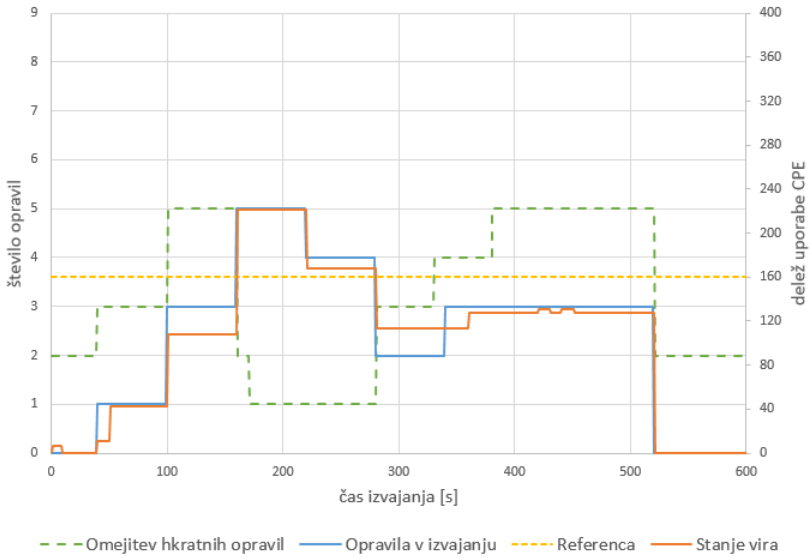


Slika 44: Simulacija obremenitve CPE – razporeditev A

Obdelava	Razporejen čas	Čas začetka	Čas izvajanja	Zakasnitev	Celotni čas
1	0	35	180	35	215
2	0	35	180	35	215
3	0	95	180	95	275
4	0	95	180	95	275
5	0	275	180	275	455
6	0	275	180	275	455
7	0	275	180	275	455
8	0	275	180	275	455

Tabela 9: Opravila razporeditve A v simulaciji obremenitve CPE

V primeru uporabe razporeditve D pride v 160. sekundi do obremenitve vira, ki pade v razred nad 190 %. Razlog je razviden iz slike 45, kjer algoritem kontrole v 100. sekundi zaradi stopnje propustnosti 2 in treh opravil v izvajanju poveča velikost vedra iz 3 na 5 opravil. Le-to omogoči sprejem enega opravila preveč. Z nastavitvijo parametra stopnje propustnosti na 1 bi se temu izognili, vendar bi s tem podaljšali čas, ki je potreben za izvedbo vseh opravil.



Slika 45: Simulacija obremenitve CPE – razporeditev D

Obdelava	Razporejen čas	Čas začetka	Čas izvajanja	Zakasnitev	Celotni čas
1	0	40	180	40	220
2	60	100	180	40	220
3	60	100	180	40	220
4	120	160	180	40	220
5	120	160	180	40	220
6	120	340	180	220	400
7	180	340	180	160	340
8	180	340	180	160	340

Tabela 10: Opravila razporeditve D v simulaciji obremenitve CPE

6.2 Simulacija obremenitve podatkovnega vira

Pri simulaciji obremenitve podatkovnega vira je bil uporabljen vir testnega dokumenta, ki temelji na skupini dveh tabel, tj. tabele glave dokumenta in tabele vrstic dokumenta. Pri tem so bili uporabljeni naslednji parametri:

- referenca 50 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Kot breme so bile uporabljene generične paketne obdelave za obremenjevanje podatkovnega vira testnega dokumenta v pripravljenih razporeditvah A, B, C, D in E. Vsaka od osmih paketnih obdelav je bila pri tem parametrizirana tako, da je obdelala 1000 dokumentov z 10 vrsticami in zadržala odprto transakcijo za 100 milisekund.

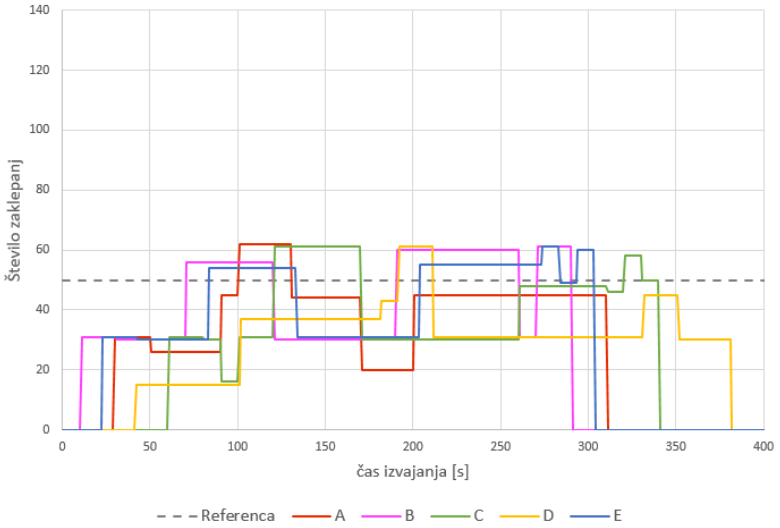
Na slikah 46 in 47 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz tabele 11, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema uspešno vzdržuje nizko prekoračitev. Čas prekoračitve pri tem v celoti pade v razreda nizke obremenitve, tj. od 50 do 55 zaklepanj in od 55 do 65 zaklepanj. Razporeditev E tudi v primeru brez kontrole ne povzroči obremenitve preko 65 zaklepanj. Vendar je v primeru z uporabo kontrole prevzema skoraj v celoti v razredu najnižje prekoračitve. Iz tabele 12, ki vsebuje čase izvajanja, je razvidno v razporeditvah C, D in E, da kljub uporabi kontrole lahko izvedemo vsa opravila v praktično istem času kot brez kontrole. Namreč manjše število zaklepanj pomeni tudi manjše število konfliktov pri posodabljanju podatkov.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 50–55 zaklep.**		**Čas prekoračitve 55–65 zaklep.**		**Čas prekoračitve nad 65 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	62	118	36,37	66,40			30			110
B	61	122	23,01	56,09	120	100	20			50
C	61	76	33,40	35,15	10		50	10		40
D	61	76	28,87	37,10			20	90		10
E	61	62	40,36	38,59	130	60	10	50		
Povp.	61,20	90,80	32,40	46,67	52,00	32,00	26,00	30,00	0,00	42,00

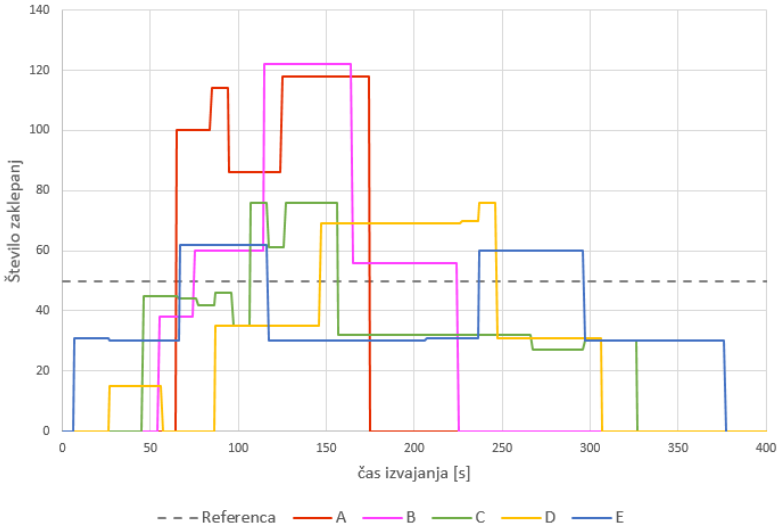
Tabela 11: Podatki simulacije obremenitve podatkovnega vira – prekoračitve

Razporeditev	**A**	**B**	**C**	**D**	**E**	**Povprečje**
Kontrola DA	283	281	281	342	282	293,80
Kontrola NE	112	171	281	281	340	237,00
Razlika	171	110	0	61	-58	56,80

Tabela 12: Podatki simulacije obremenitve podatkovnega vira – čas izvajanja



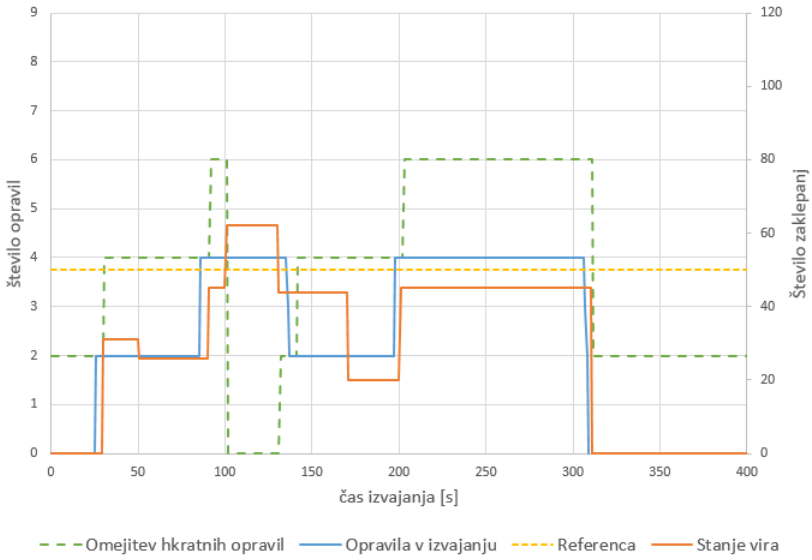
Slika 46: Simulacija obremenitve podatkovnega vira s kontrolo prevzema



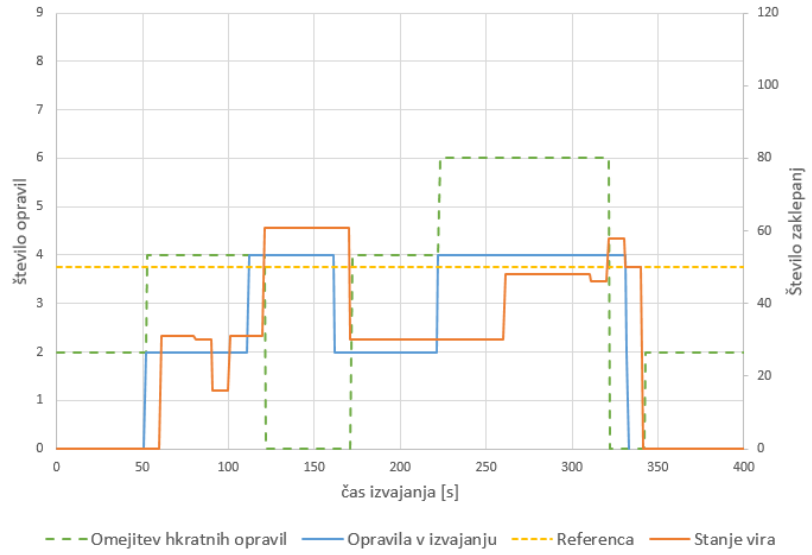
Slika 47: Simulacija obremenitve podatkovnega vira brez kontrole prevzema

Najbolj se učinkovitost kontrole prevzema izkaže pri uporabi razporeditve A (slika 48). V tem primeru kontrola prevzema visoke obremenitve 118 zaklepanj v času 110 sekund omeji na 62 zaklepanj v času 30 sekund. Iz časov začetka posameznih opravil iz tabele 13 je razvidno, kako kontrola prevzema porazdeli sprejem opravil in s tem prerazporedi opravila v razporeditev, ki je podobna razporeditvi E.

Iz grafov razporeditve A v 200. sekundi (slika 48) in razporeditve C v 222. sekundi (slika 49) je razvidno, da bi kontrola, če bi obstajala dodatna opravila, dovolila prevzem 5. in 6. opravila. Le-to bi lahko vodilo v povečano obremenitev. Takšnih dogodkov kontrola prevzema ne zna preprečiti, ker algoritem razpolaga le z informacijo, katere vire posamezno opravilo obremenjuje, in nima informacije, v kakšni meri in koliko časa jih bo opravilo obremenjevalo. Algoritem bo v primeru prekoračitve referenčne meje le preprečil nadaljnji prevzem opravil in s tem poslabšanje situacije.



Slika 48: Simulacija obremenitve podatkovnega vira – razporeditev A



Slika 49: Simulacija obremenitve podatkovnega vira – razporeditev C

Obdelava	Razporejen čas	Čas začetka	Čas izvajanja	Zakasnitev	Celotni čas
1	0	26	110	26	136
2	0	26	111	26	137
3	0	86	111	86	197
4	0	86	111	86	197
5	0	197	110	197	307
6	0	197	111	197	308
7	0	198	111	198	309
8	0	198	111	198	309

Tabela 13: Opravila razporeditve A v simulaciji obremenitve podatkovnega vira

6.3 Uporaba bremena realnih paketnih obdelav

V tem razdelku bo predstavljeno delovanje rešitve ob obremenjevanju testnega okolja z bremenom, sestavljenim iz realnih paketnih obdelav. S stališča obvladovanja preobremenitve bo obravnavanih več različnih virov sistema AX, ki so bili predstavljeni pri analizi virov sistema. Pri tem se bom osredotočil na vire, ki so s stališča naloge zanimivi. Le-to so viri, ki so izpostavljeni preobremenitvam in jih je smiselno obravnavati. Iz obravnave so izpuščeni naslednjih viri:

- *Pomnilnik strežnika AOS*, ker proces vedno zadrži alociran pomnilnik in ga sprošča šele po daljšem času neaktivnosti. Kot sem navedel v razdelku 3.1, je bila zadostnost pomnilnika zagotovljena.
- *Omrežje*, ker gre zaradi namestitve vseh komponent sistema na isti računalnik zgolj za komunikacijo med procesi znotraj operacijskega sistema.
- *Skupina tabel porabe*, ker se je pri analizi izkazalo, da je obremenitev vira zanemarljiva.

Podlaga za nastavitev parametrov posameznega vira, kot je referenčna meja, so bili rezultati analize virov sistema in izkustveno znanje o delovanju kontrole prevzema, pridobljeno z izvedbo simulacij v prejšnjih dveh razdelkih. Določitev velikostnih razredov obremenitev je bila narejena za vsak vir ločeno na podlagi obdelave pridobljenih meritev.

Kot breme so bile uporabljene realne paketne obdelave v pripravljenih razporeditvah A, B, C, D, E, F in G. Vsaka posamezna paketna obdelava je bila pri tem parametrizirana, tako da obdeluje podatke za 1000 odjemnih mest. Tako na primer obdelava obračuna pripravi 1000 računov, obdelava knjiženja računov poknjiži 1000 računov, obdelava knjiženja plačil poknjiži 1000 plačil idr. Da bi bili zagotovljeni čim bolj enaki pogoji za izvedbo meritev ob uporabi posamezne razporeditve, je bila vedno predhodno uporabljena ogrevalna (angl. *warm up*) procedura, ki je z izvedbo primerka vsake paketne obdelave poskrbela za predpomnjenje aplikacijske kode, predpomnjenje podatkov, alokacijo pomnilnika strežnika AOS idr.

Obdelava za konsolidacijo računov se pri uporabi vseh razporeditev brez kontrole prevzema zaključi z napako. Razlog za to je preveliko število konfliktov pri posodabljanju podatkov. Zato je v primerjavo časov v tabeli 14 vključen pribitek 300 sekund za neuspešno izvedeno obdelavo. Pribitek je enak povprečnemu času za izvedbo obdelave konsolidacije računov (240 sekund), povečanim za interval razporeditve nove obdelave (60 sekund).

Razporeditev	**Kontrola DA**		**Kontrola NE**				**Razlika v času**
	Napake	Čas izvajanja	Napake	Čas izvajanja	Pribitek napake	Popravljen čas izvajanja	
A	0	788	1	332	300	632	156
B	0	702	1	307	300	607	95
C	0	816	1	475	300	775	41
D	0	745	1	393	300	693	52
E	0	760	1	550	300	850	-90
F	0	846	1	364	300	664	182
G	0	816	1	403	300	703	113

Tabela 14: Primerjava časov izvajanja razporeditev realnih paketnih obdelav

6.3.1 Obremenitev CPE

Pri spremljanju obremenitve CPE sta bila uporabljena dva vira. Prvi vir je bil CPE AOS, ki temelji na performančnem števcu, ki spremlja delež uporabljenega procesorskega časa s strani procesa strežnika AOS. Drugi je bil CPE SQL, ki temelji na performančnem števcu, ki spremlja delež uporabljenega procesorskega časa s strani procesa podatkovnega strežnika. Za oba vira so bili uporabljeni naslednji parametri:

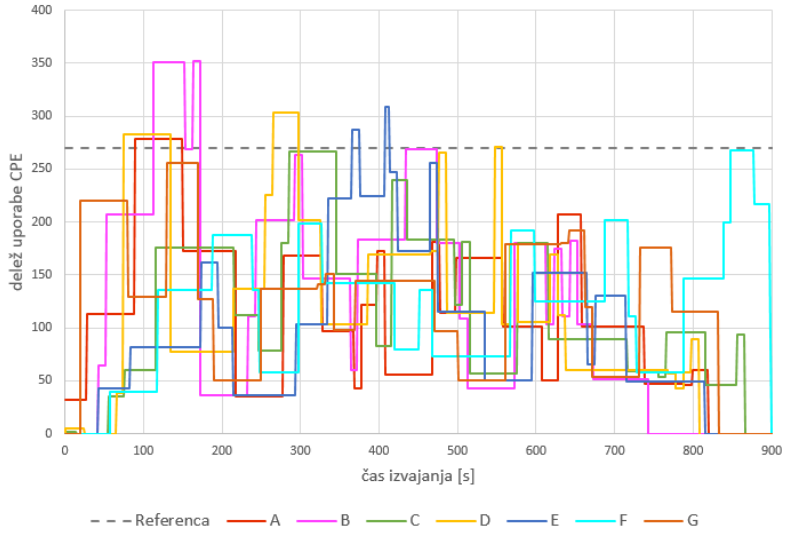
- referenca 270 %;
- faktor povečanja velikosti 0,03;
- faktor manjšanja velikosti 0,10;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 1 obdelava;
- maksimalni dinamični prag 50 %.

Na slikah 50 in 52 so prikazani rezultati meritev vira CPE AOS ob uporabi kontrole prevzema in brez nje. Iz tabele 15, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema uspešno vzdržuje nizko prekoračitev. Čas prekoračitve pri tem skoraj v celoti pade v razreda nizke obremenitve, tj. od 270 % do 290 % in od 290 % do 310 %. Izjema je pri razporeditvi B, kjer obremenitev za čas 50 s preseže 310 %. Pri tem je iz maksimalne vrednosti razvidno, da obremenitev doseže 352 %. Iz podatkov je tudi razvidno, da je brez kontrole prevzema CPE bistveno dalj časa in bistveno bolj obremenjen.

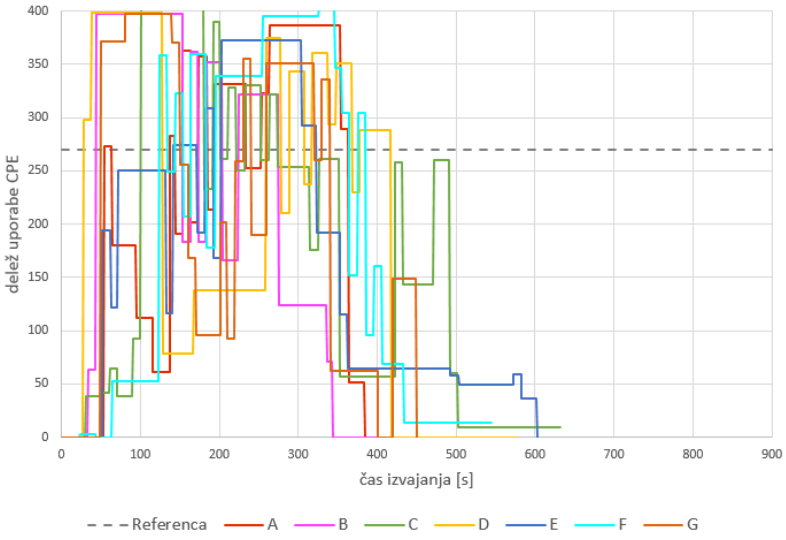
Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 270–290 %**		**Čas prekoračitve 290–310 %**		**Čas prekoračitve nad 310 %**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	278,79	386,77	8,79	83,97	61	26				161
B	352,31	397,30	81,08	100,52					50	191
C	266,18	406,05		110,21						130
D	303,03	398,49	18,43	86,09	70	40	33	20		168
E	309,22	372,73	25,30	70,46	10	30	6	30		101
F	268,18	402,94		92,28				20		201
G	255,39	397,98		100,09						180
Povp.	290,44	394,61	19,09	91,95	20,14	13,71	5,57	10,00	7,14	161,71

Tabela 15: Podatki obremenitve CPE strežnika AOS

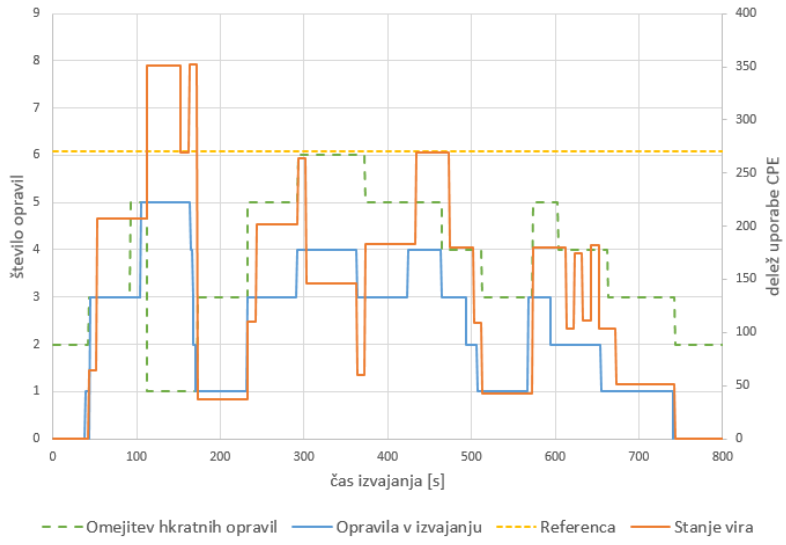
V primeru uporabe razporeditve B v 113. sekundi obremenitev doseže 352 %. Razlog je razviden iz slike 52, kjer algoritem kontrole v 93. sekundi zaradi stopnje propustnosti 2 in treh opravil v izvajanju poveča velikost vedra iz 3 na 5 opravil. Le-to omogoči sprejem dveh novih opravil, ki povzročita visoko obremenitev.



Slika 50: Obremenitev CPE strežnika AOS s kontrolo prevzema

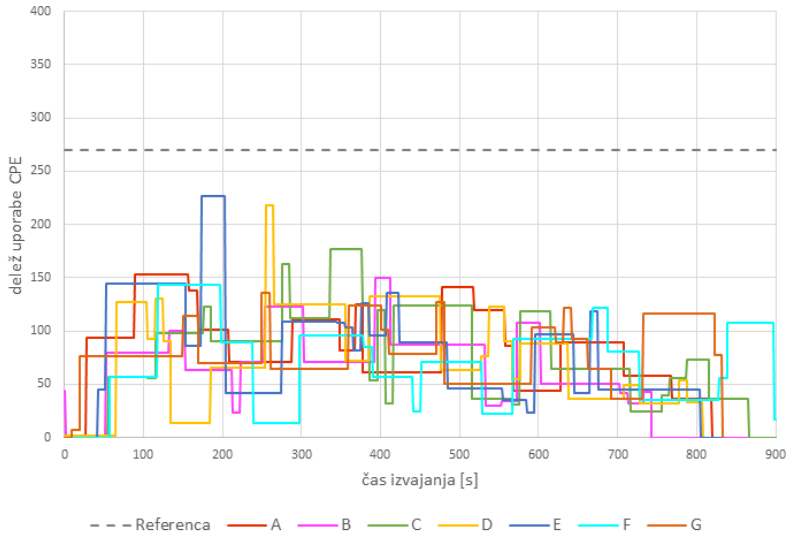


Slika 51: Obremenitev CPE strežnika AOS brez kontrole prevzema

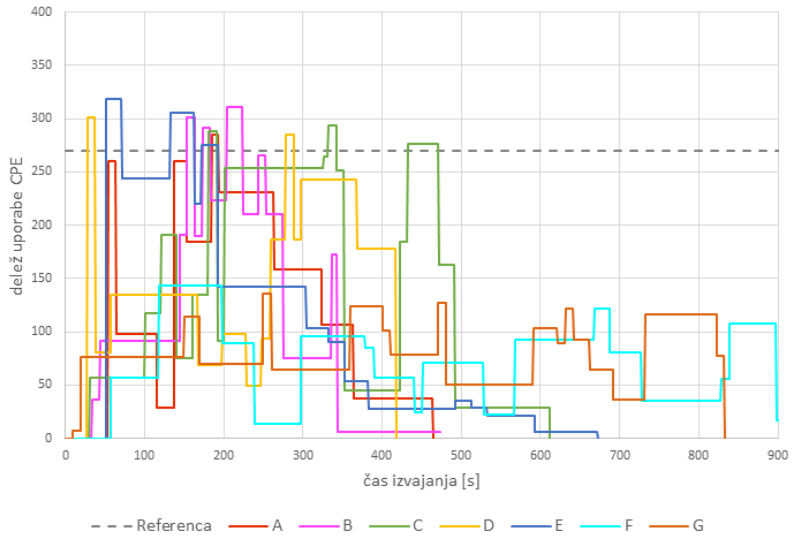


Slika 52: Obremenitev CPE strežnika AOS – razporeditev B

Na slikah 53 in 54 so prikazani rezultati meritev vira CPE SQL ob uporabi kontrole prevzema in brez nje. Obremenjenost vira redko preseže referenčno mejo že ob uporabi brez kontrole prevzema. Z uporabo le-te na preostalih virih sistema pa se posledično zmanjša aktivnost procesa podatkovnega strežnika. Poudariti je potrebno, da med virom CPE SQL in podatkovnimi viri obstaja odvisnost, saj je podatkovni strežnik tisti, ki nadzoruje delo s podatki.



Slika 53: Obremenitev CPE podatkovnega strežnika s kontrolo prevzema



Slika 54: Obremenitev CPE podatkovnega strežnika brez kontrole prevzema

6.3.2 Obremenitev trdega diska

Pri spremljanju obremenitve trdega diska je bil uporabljen vir, ki temelji na performančnem števcu, ki spremlja povprečno število zahtev v vrsti in obdelavi. Za vir so bili uporabljeni naslednji parametri:

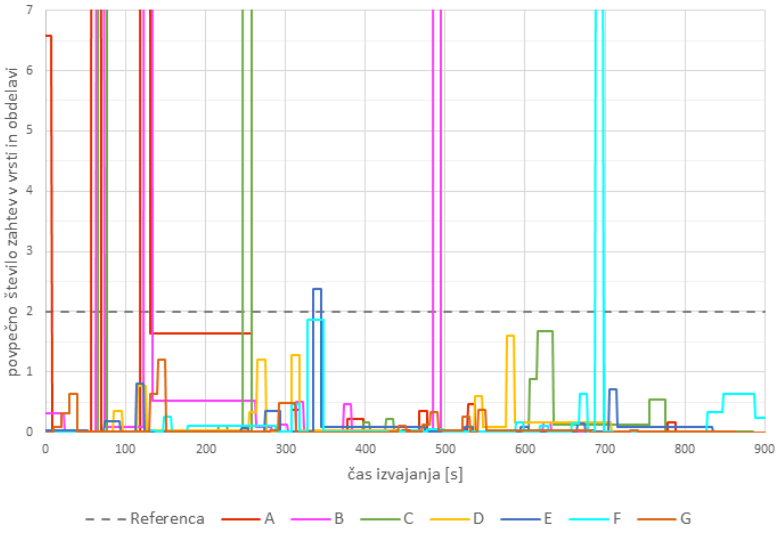
- referenca 2;
- faktor povečanja velikosti 2;
- faktor manjšanja velikosti 4;
- stopnja propustnosti 4 obdelave;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav.

Na slikah 55 in 56 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz prikaza meritev in iz tabele 16, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema ne vpliva na zmanjšanje obremenitev. Monitoring v obeh primerih s kontrolo in brez nje zazna špice, ki trajajo približno 10 sekund. Iz tega razloga je bila kontrola prevzema nastavljena, tako da lahko hitro odreagira na prekoračitev (slika 57).

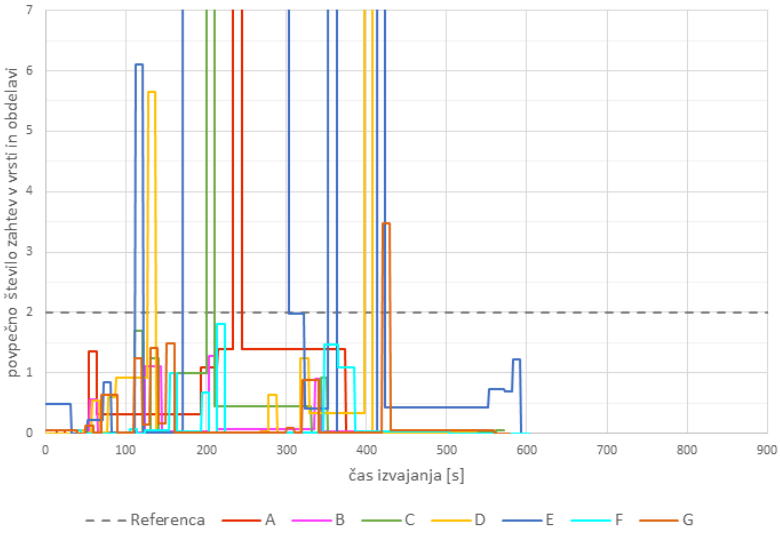
Pri tem je potrebno poudariti, da na samo obremenitev trdega diska lahko vpliva več dejavnikov testnega okolja (npr. lastnosti predpomnilnikov in ostali procesi), ki niti niso v korelaciji z izvajanjem paketnih obdelav. Ker to presega okvir naloge, bo obravnava le-tega izpuščena. Doprinos kontrole prevzema v tem primeru seveda ni čisto zanemarljiv, ker le-ta še vedno ob prekoračitvi referenčne meje zameji prevzem novih opravil in s tem prepreči potencialno poslabšanje situacije.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 2–7**		**Čas prekoračitve 7–28**		**Čas prekoračitve nad 28**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	76,99	53,47	47,18	51,47	8				22	11
B	77,36	1,29	35,94	0,00			10		19	
C	86,31	9,03	84,31	7,03				10	21	
D	1,60	43,84	0,00	22,75		10				10
E	2,37	128,19	0,37	24,66	10	10		122		29
F	8,90	1,80	6,90	0,00			10			
G	1,20	3,47	0,00	1,47		10				
Povp.	36,39	34,44	24,96	15,34	2,57	4,29	2,86	18,86	8,86	7,14

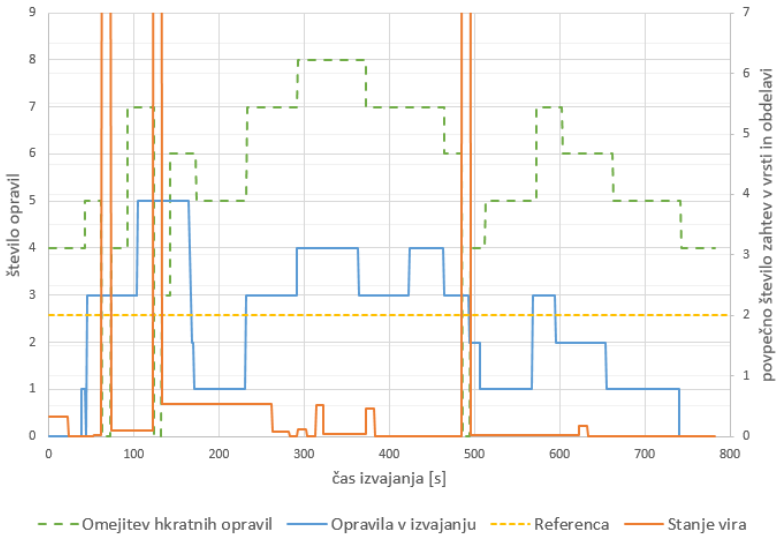
Tabela 16: Podatki obremenitve trdega diska



Slika 55: Obremenitev trdega diska s kontrolo prevzema



Slika 56: Obremenitev trdega diska brez kontrole prevzema



Slika 57: Obremenitev trdega diska – razporeditev B

6.3.3 Obremenitev podatkovnih tabel računa

Pri spremljanju obremenitve skupine podatkovnih tabel računa so bili uporabljeni naslednji parametri:

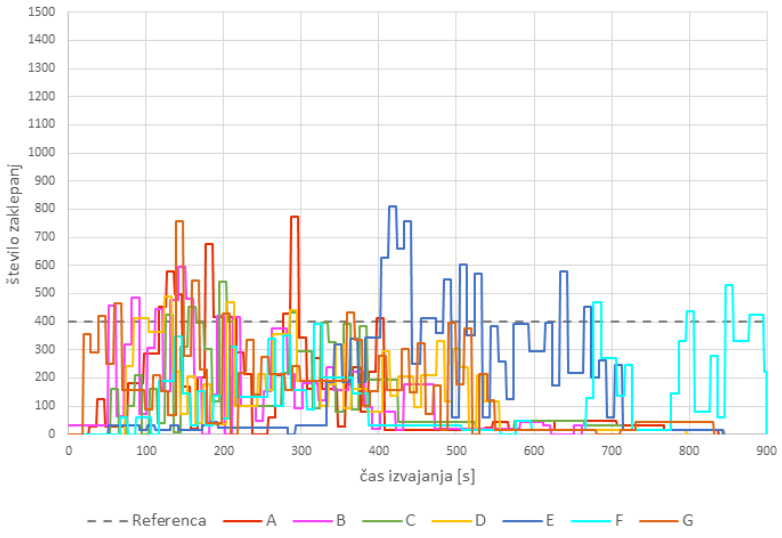
- referenca 400 zaklepanj;
- faktor povečanja velikosti 0,03;
- faktor manjšanja velikosti 0,10;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Na slikah 58 in 59 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz tabele 17, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema uspešno vzdržuje nizko prekoračitev. Čas prekoračitve pri tem v večjem delu pade v razred nizke obremenitve, tj. od 400 do 470 zaklepanj. Prekoračitev 600 zaklepanj je pri razporeditvah A in G kratka. Prav tako so pri tem maksimalne vrednosti v primerjavi z izvedbo brez kontrole občutno nižje. Pri razporeditvi E je obremenitev 600 zaklepanj presežena za čas 50 sekund in doseže 808 zaklepanj. Izločitev časov prekoračitev do 470 zaklepanj pokaže, da je brez kontrole podatkovni vir v povprečju 56 sekund dlje obremenjen.

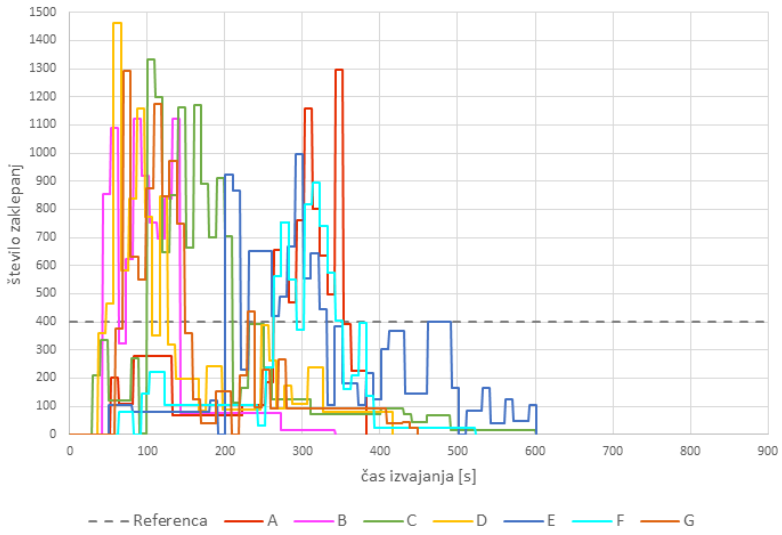
Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 400–470 zaklep.**		**Čas prekoračitve 470–600 zaklep.**		**Čas prekoračitve nad 600 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	774	1295	107,20	381,00	60		20	20	20	70
B	595	1122	72,50	490,89	40		40			90
C	543	1331	51,00	529,55	40		10			110
D	490	1463	45,20	474,86	40	10	10	10		50
E	808	995	184,73	211,28	30	50	30	20	50	81
F	530	896	58,00	262,63	40	10	10	30		40
G	757	1291	108,17	436,67	40	10	10	10	10	70
Povp.	642	1199	89,54	398,12	41,43	11,43	18,57	12,86	11,43	73,00

Tabela 17: Podatki obremenitve tabel računa

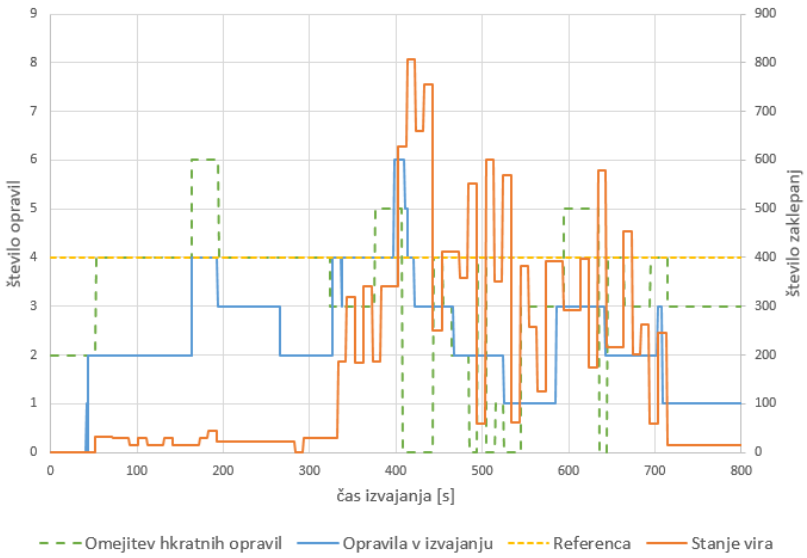
V primeru uporabe razporeditve E v 403. sekundi obremenitev preseže 600 zaklepanj. Razlog je v nastavitvi stopnje propustnosti, kar je razvidno iz slike 60, in podatkov o opravilih v prilogi 9.3.5. V tem primeru večjo obremenitev povzroči kombinacija različnih opravil. Pred prekoračitvijo se izvajajo opravilo predpriprave obračuna, opravilo konsolidacije računov in opravilo obračuna. Prva dva od teh opravil sta manj intenzivna. Kontrola prevzema ob tem poveča število dovoljenih opravil iz 3 na 5 in zaženeta se še dve dodatni opravili obračuna, ki skupaj s prvim povečata obremenitev.



Slika 58: Obremenitev podatkovnih tabel računa s kontrolo prevzema



Slika 59: Obremenitev podatkovnih tabel računa brez kontrole prevzema



Slika 60: Obremenitev tabel računa – razporeditev E

6.3.4 Obremenitev podatkovnih tabel dokumenta

Pri spremljanju obremenitve skupine podatkovnih tabel računovodskega dokumenta so bili uporabljeni naslednji parametri:

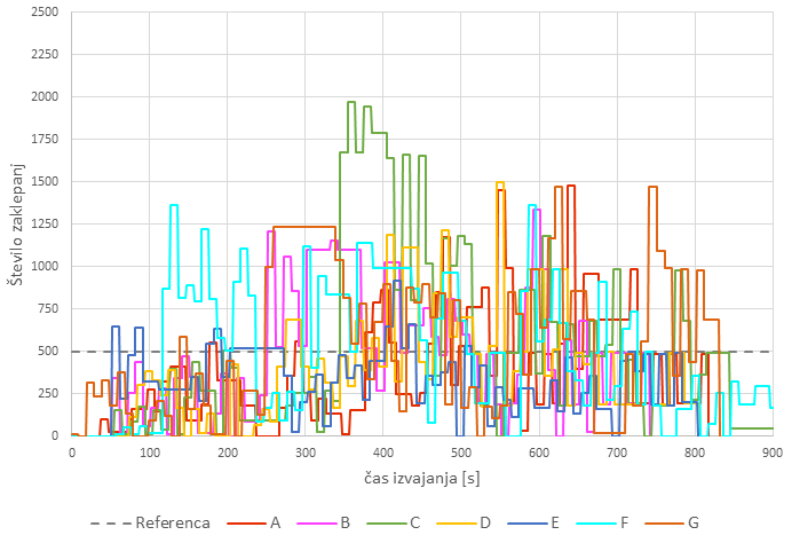
- referenca 500 zaklepanj;
- faktor povečanja velikosti 0,03;
- faktor manjšanja velikosti 0,10;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Na slikah 61 in 62 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz meritev je razvidno, da so kljub uporabi kontrole, prekoračitve referenčne meje visoke. Toda če jih primerjamo z izvedbo brez kontrole, je prispevek kontrole prevzema opazen. Le-ta ob prekoračitvi še vedno zameji prevzem novih obdelav. Pri uporabi brez kontrole prevzema je 75 odstotkov prekoračitev večjih od 980 zaklepanj, medtem ko je pri uporabi kontrole prevzema 75 odstotkov prekoračitev manjših od 1092 zaklepanj. Iz tabele 18, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema večji del prekoračitev zadržuje v razredu obremenitve od 500 do 750 zaklepanj. Večje prekoračitve do 1000 in nad 1000 zaklepanj so posledica opravil paketne obdelave knjiženja računov. Le-to je bilo že ugotovljeno pri analizi bremena. Ker kontrola nima informacije, v kakšni meri bo opravilo obremenjevalo vir, s prevzemom več takšnih opravil pride do večje obremenitve.

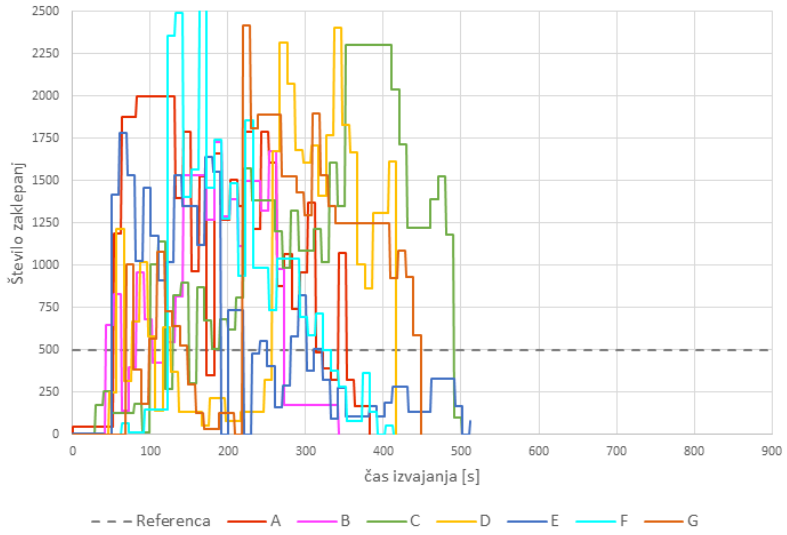
Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 500–750 zaklep.**		**Čas prekoračitve 750–1000 zaklep.**		**Čas prekoračitve nad 1000 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	1477	1995	315	994	150	10	60	50	30	200
B	1336	1727	364	701	130	50	120	20	30	120
C	1970	2305	668	848	70	61	70	70	130	240
D	1501	2402	366	945	101	30	31	30	50	150
E	918	1780	86	636	150	61	10	30		110
F	1364	2527	388	805	170	50	150	60	70	100
G	1471	2419	435	785	171	50	121	50	111	190
Povp.	1434	2165	375	816	134,57	44,57	80,29	44,29	60,14	158,57

Tabela 18: Podatki obremenitve tabel dokumenta

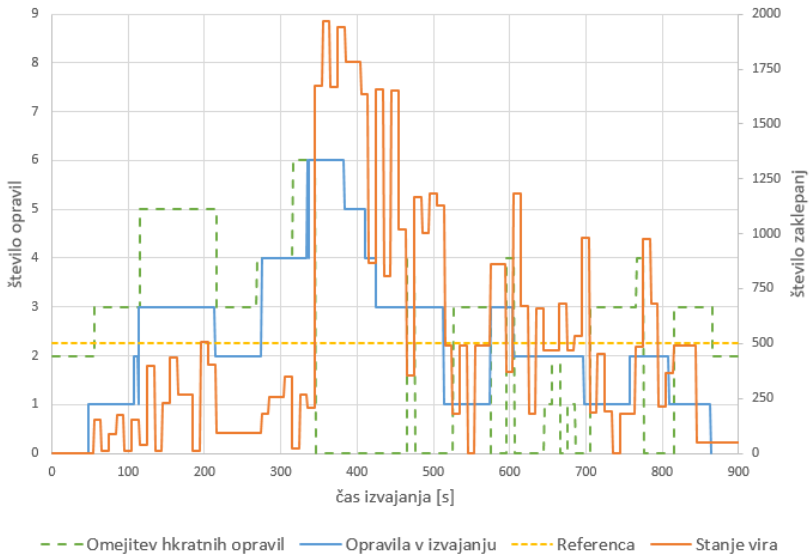
Na sliki 63 je podrobneje prikazan primer uporabe razporeditve C, ki ima v danem primeru najdaljšo prekoračitev nad 1000 zaklepanj. Razlog za prekoračitev je pomanjkljivost v delu algoritma kontrolne zanke, ki s stopnjo propustnosti omejuje prevzem novih opravil. Če v intervalu med posodobitvama velikosti vedra pride do zključka večjega števila opravil, kot je stopnja propustnosti, lahko pride do prevzema novih opravil v istem številu. V danem primeru se v 328. sekundi konča eno opravilo obračuna in v 336. sekundi dve opravili obračuna. Ker se še ni izvedlo zmanjšanje velikosti vedra, se v 337. sekundi lahko prevzamejo tri nova opravila knjiženja računov, ki so bistveno bolj intenzivna od opravil obračuna.



Slika 61: Obremenitev podatkovnih tabel dokumenta s kontrolo prevzema



Slika 62: Obremenitev podatkovnih tabel dokumenta brez kontrole prevzema



Slika 63: Obremenitev tabel dokumenta – razporeditev C

6.3.5 Obremenitev podatkovnih tabel dnevnika računov

Pri spremljanju obremenitve skupine podatkovnih tabel dnevnika računov so bili uporabljeni naslednji parametri:

- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

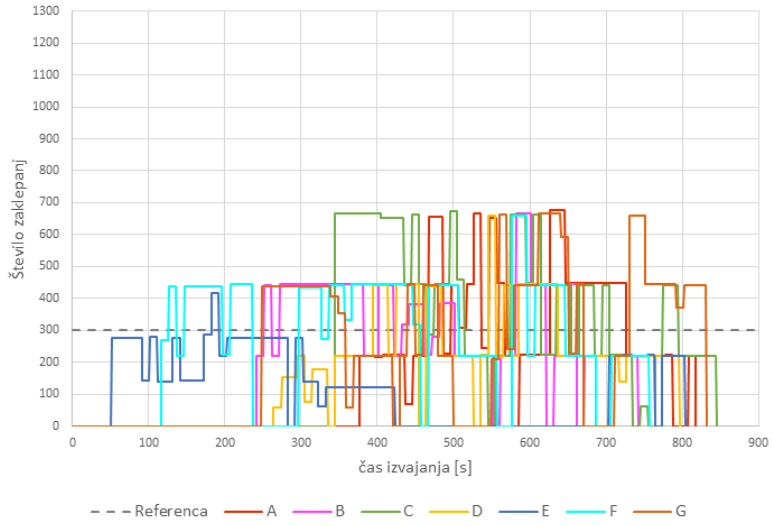
Na slikah 64 in 65 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz tabele 19, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema večino prekoračitev zadrži v razredu od 390 do 460 zaklepanj. Podobno kot pri tabelah dokumenta so tabele dnevnika računa obremenjene z obdelavo knjiženja računov. V tem primeru so prekoračitve referenčne meje manjše. Potrebno pa je poudariti, da skupino tabel dokumenta sestavlja trikrat več tabel (6 tabel) in obremenjujejo tri obdelave (obračun, konsolidacija računov in knjiženje računov), medtem ko skupino dnevnika računov le ena (knjiženje računov). Večje prekoračitve nad 460 zaklepanj so prisotne, toda segajo v povprečju le do 666 zaklepanj. Pri tem po času izstopa uporaba razporeditve C. Kljub prekoračitvam so rezultati bistveno boljši kot v primeru brez kontrole prevzema, kjer je 75 odstotkov prekoračitev večjih od 709 zaklepanj.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–390 zaklep.**		**Čas prekoračitve 390–460 zaklep.**		**Čas prekoračitve nad 460 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	676	1082	211	541	10	10	110		60	200
B	665	1082	147	566	50		170		20	130
C	673	1081	257	517			130		140	270
D	658	1278	158	686			131		10	160
E	418	1125	118	526			10	10		130
F	658	1288	148	601	20		300	30	20	120
G	666	1203	181	474	20	10	241		71	220
Povp.	631	1163	174	559	14,29	2,86	156,00	5,71	45,86	175,71

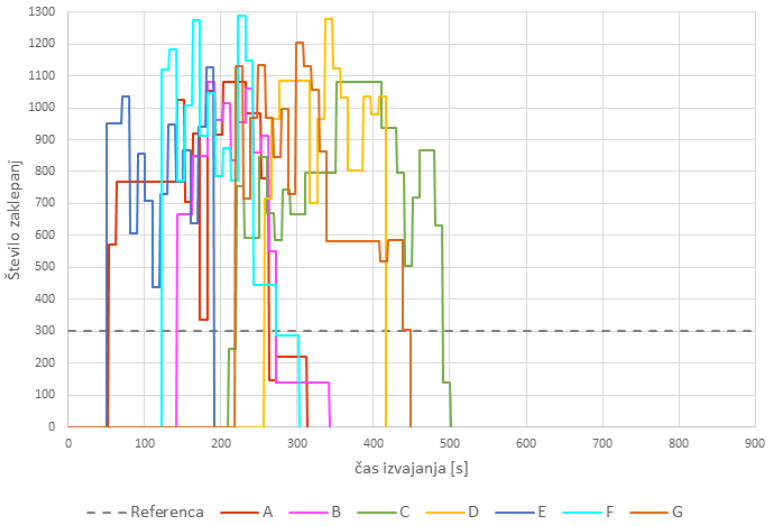
Tabela 19: Podatki obremenitve tabel dnevnika računov

Na sliki 66 je podrobneje prikazan primer uporabe razporeditve C, ki ima v danem primeru najdaljšo prekoračitev nad 460 zaklepanj. Razlog za prekoračitev je v zakasnitvi upoštevanja stopnje propustnosti ob zaključku opravila in je enak razlogu, ki je naveden pri skupini tabel dokumenta. Iz podatkov o opravilih iz priloge 9.3.3 je razvidno, da se od 335. do 337. sekunde izvaja opravilo knjiženja računov, ki kot prvo opravilo knjiženja naredi predpripravo podatkov in je manj intenzivno. To je razlog, da v 336. sekundi kontrola prevzema poveča število dovoljenih opravil iz 2 na 3. Ker pa kontrola ne odreagira na zaključek opravila v 337. sekundi,

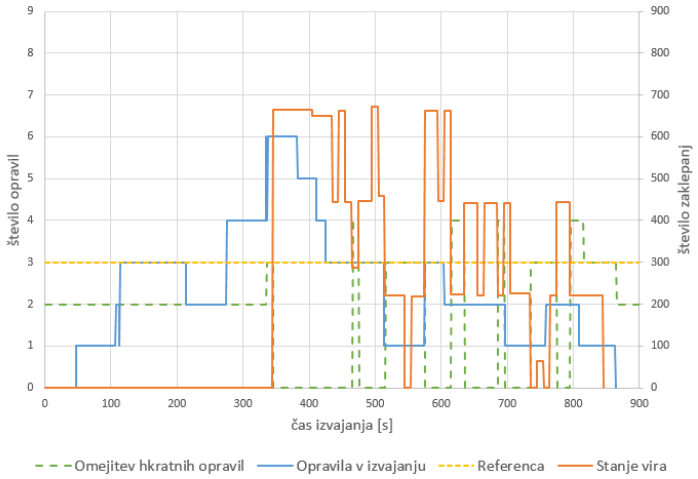
se lahko prevzamejo tri nova opravila knjiženja računov, ki so bistveno bolj intenzivna od osnovnega opravila.



Slika 64: Obremenitev podatkovnih tabel dnevnika računov s kontrolo prevzema



Slika 65: Obremenitev podatkovnih tabel dnevnika računov brez kontrole prevzema



Slika 66: Obremenitev tabel dnevnika računov – razporeditev C

6.3.6 Obremenitev podatkovnih tabel davčnih postavk

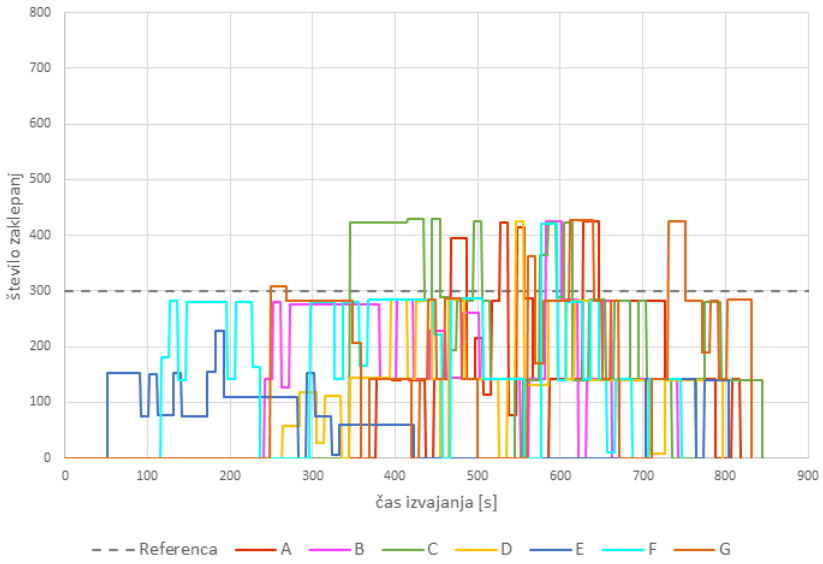
Pri spremljanju obremenitve skupine podatkovnih tabel davčnih postavk so bili uporabljeni naslednji parametri:

- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Kot pri tabelah dnevnika računov so tabele davčnih postavk računa obremenjene le z obdelavo knjiženja računov. Ker gre za ista opravila, ki jih že omeji bolj obremenjen vir, opazimo veliko podobnost, če primerjamo rezultate meritev obeh virov na slikah 67 in 64. Iz tabele 20, ki prikazuje podatke o prekoračitvah referenčne meje, je razvidno, da kontrola prevzema prekoračitve zadrži v razredu od 370 do 430 zaklepanj. Prekoračitev čez 430 zaklepanj ni.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–370 zaklep.**		**Čas prekoračitve 370–430 zaklep.**		**Čas prekoračitve nad 430 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	426	606	113	180		20	60			180
B	425	593	125	164		20	20	10		90
C	429	657	121	186	10	50	130	60		150
D	425	731	125	256		30	10	10		120
E	228	576		144		20		40		60
F	421	739	121	276			20	10		110
G	428	665	90	163	30	100	51			110
Povp.	397	652	99	196	5,71	34,29	41,57	18,57	0,00	117,14

Tabela 20: Podatki obremenitve tabel davčnih postavk



Slika 67: Obremenitev podatkovnih tabel davčnih postavk s kontrolo prevzema

6.3.7 Obremenitev podatkovnih tabel temeljnice

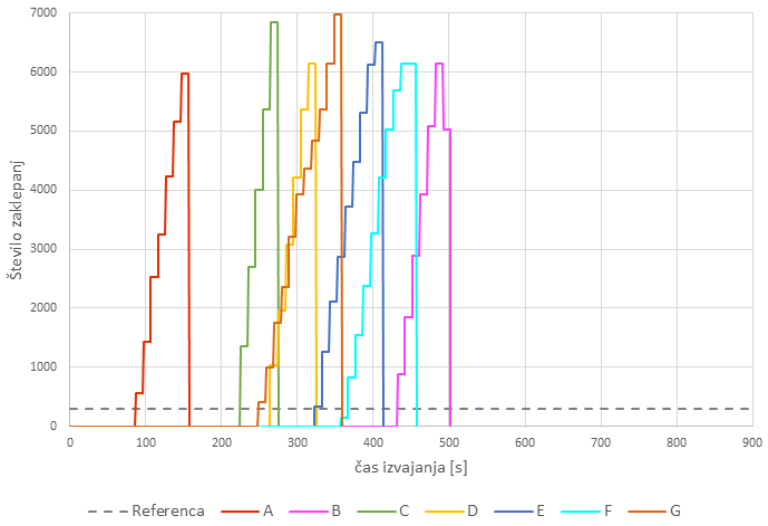
Pri spremljanju obremenitve skupine podatkovnih tabel dokumenta temeljnice so bili uporabljeni naslednji parametri:

- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 1 obdelava;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Skupino tabel obremenjuje le ena obdelava, tj. knjiženje temeljnice plačil. Le-ta izvede vse spremembe podatkov v eni sami transakciji, kar je razlog za veliko prekoračitev. Iz podatkov prekoračitev v tabeli 21 ni razvidne bistvene razlike glede na uporabo s kontrolo prevzema in brez. Edini doprinos kontrole prevzema v takšni situaciji je preprečitev poslabšanja situacije s prevzemom še ene obdelave knjiženja temeljnice. Pri tem je bilo umetno breme sestavljeno le iz ene obdelave knjiženja temeljnice.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–2000 zakl.**		**Čas prekoračitve 2000–5000 zakl.**		**Čas prekoračitve nad 5000 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	5978	6151	3006	3516	20	20	30	30	20	40
B	6149	6151	3388	3375	20	20	20	30	30	20
C	6844	6435	3757	3473	10	31	20	50	20	40
D	6151	5928	3292	3025	21	10	20	30	20	10
E	6495	6151	3337	3440	20	20	40	60	30	30
F	6151	6146	3614	3414	20	30	30	50	40	30
G	6971	6076	3367	3197	30	10	50	30	30	10
Povp.	6391	6148	3394	3349	20,14	20,14	30,00	40,00	27,14	25,71

Tabela 21: Podatki obremenitve tabel temeljnice



Slika 68: Obremenitev podatkovnih tabel temeljnice s kontrolo prevzema

6.3.8 Obremenitev podatkovnih tabel bančnih postavk

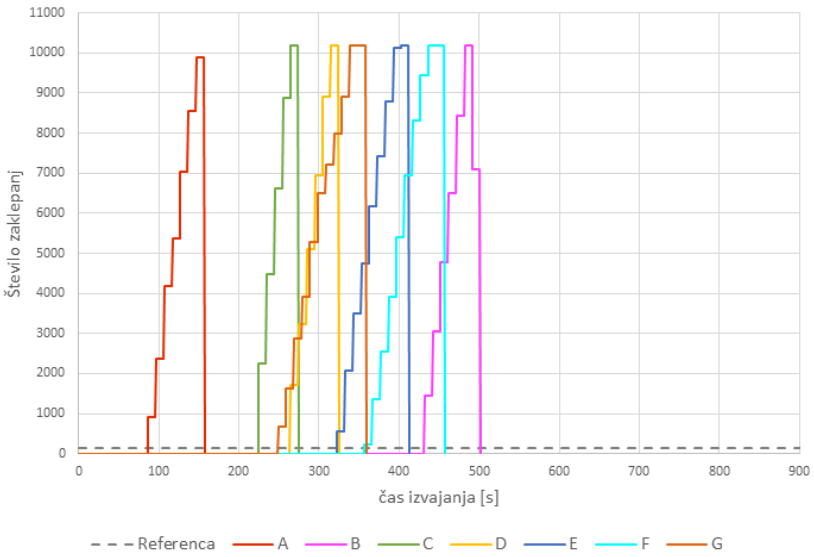
Pri spremljanju obremenitve skupine podatkovnih tabel bančnih postavk so bili uporabljeni naslednji parametri:

- referenca 140 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 1 obdelava;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Enako kot skupino tabel temeljnice obremenjuje skupino tabel bančne postavke paketna obdelava knjiženja temeljnice plačil. Ker gre za isto opravilo, opazimo veliko podobnost, če primerjamo rezultate meritev obeh virov na slikah 68 in 69. Iz podatkov prekoračitev v tabeli 22 ni razvidne bistvene razlike glede na uporabo s kontrolo prevzema in brez, ker eno samo opravilo vedno povzroči visoko obremenitev vira.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 140–3000 zakl.**		**Čas prekoračitve 3000–8900 zakl.**		**Čas prekoračitve nad 8900 zaklep.**	
Kont.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	9876	10177	5328	6173	20	20	40	40	10	30
B	10177	10177	5787	5938	10	20	50	30	10	20
C	10177	10177	6339	6062	10	21	30	70	10	30
D	10177	9795	5803	5352	11	10	30	30	20	10
E	10177	10177	5811	5569	20	30	50	60	20	30
F	10177	10177	5712	6006	30	20	40	60	30	30
G	10177	10055	5800	5646	30	10	60	30	20	10
Povp.	10134	10105	5797	5821	18,71	18,71	42,86	45,71	17,14	22,86

Tabela 22: Podatki obremenitve tabel bančnih postavk



Slika 69: Obremenitev podatkovnih tabel bančnih postavk s kontrolo prevzema

6.3.9 Obremenitev podatkovnih tabel postavk kupcev

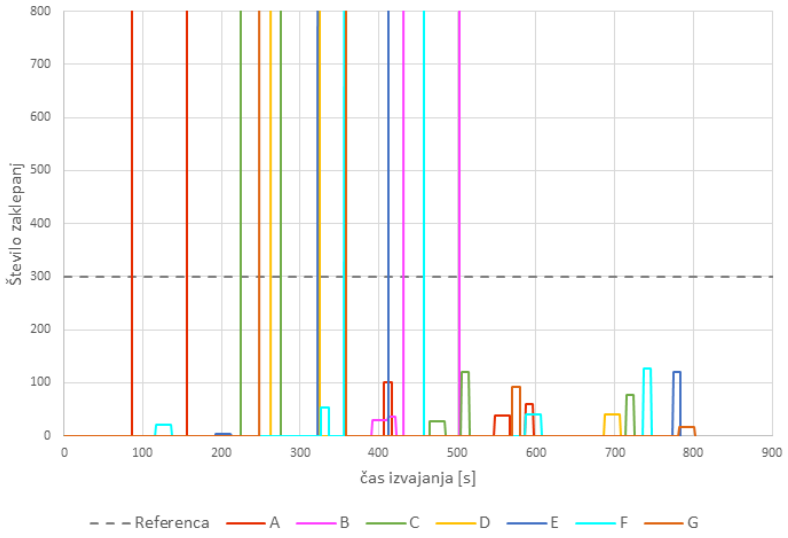
Pri spremljanju obremenitve skupine podatkovnih tabel postavk kupcev so bili uporabljeni naslednji parametri:

- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Skupino tabel obremenjujejo opravila knjiženja računov, ki predstavljajo manjše obremenitve, in opravilo knjiženja temeljnice plačil, ki predstavlja preobremenitve (slika 70). Pri tem prekoračitve sovpadajo s prekoračitvami skupine tabel temeljnice. Iz podatkov prekoračitev v tabeli 23 ni razvidne bistvene razlike glede na uporabo s kontrolo prevzema in brez.

Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–34000 zaklepanj**		**Čas prekoračitve 34000–98000 zaklepanj**		**Čas prekoračitve nad 98000 zaklepanj**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	109653	112808	60462	69900	20	20	40	40	10	30
B	112355	112389	65262	66874	20	20	40	30	10	20
C	112015	114038	71062	69376	10	21	30	60	10	40
D	112101	108337	65324	60544	11	10	30	30	20	10
E	112793	113849	65735	63640	20	30	50	60	20	30
F	112589	113811	64475	68496	30	20	40	60	30	30
G	112821	110933	65638	63644	30	10	50	30	30	10
Povp.	112047	112309	65423	66068	20,14	18,71	40,00	44,29	18,57	24,29

Tabela 23: Podatki obremenitve tabel postavk kupcev



Slika 70: Obremenitev podatkovnih tabel postavk kupcev s kontrolo prevzema

6.3.10 Obremenitev podatkovnih tabel pomožne knjige

Pri spremljanju obremenitve skupine podatkovnih tabel pomožne knjige so bili uporabljeni naslednji parametri:

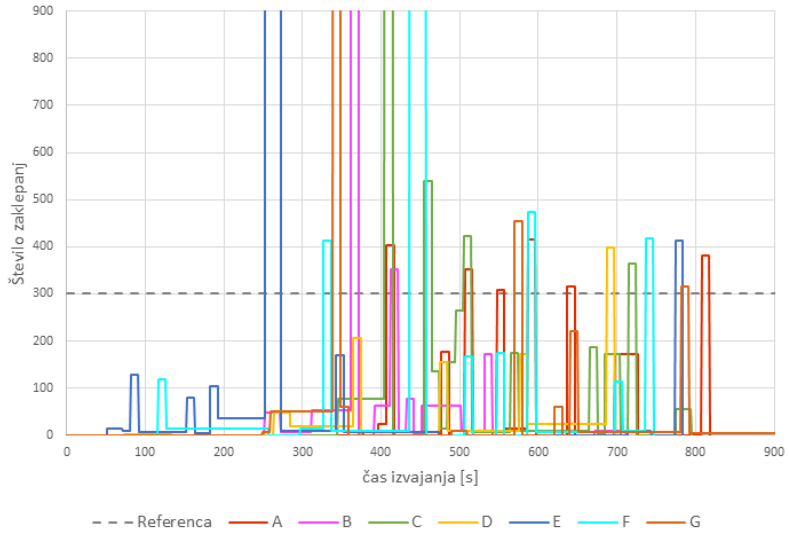
- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 2 obdelavi;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Skupino tabel obremenjujejo opravila knjiženja računov in opravilo prenosa pomožne knjige v glavno knjigo. Na slikah 71 in 72 so prikazani rezultati meritev ob uporabi kontrole prevzema in brez nje. Iz podatkov prekoračitev v tabeli 24 ni razvidne bistvene razlike glede na uporabo s kontrolo prevzema in brez. Visoke maksimalne obremenitve čez 4000 zaklepanj se pojavijo v obeh primerih. Pri uporabi kontrole prevzema je čas prekoračitev v razredu nad 420 zaklepanj malenkost krajši.

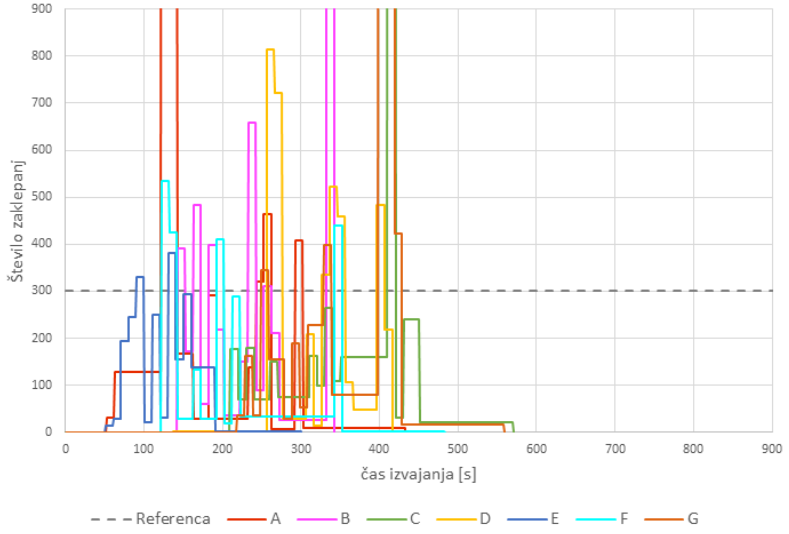
Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–370 zaklep.**		**Čas prekoračitve 370–420 zaklep.**		**Čas prekoračitve nad 420 zaklep.**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	414	4184	63	1073	30	10	30	10		30
B	11367	4084	5559	754	10	10		20	10	30
C	4114	4192	1061	3892	10				30	10
D	398	814	98	256		10	10			50
E	4112	381	2579	56		10	10	10	20	
F	4099	534	1150	152			20	10	30	30
G	4096	11256	1322	3008	10	10		10	20	30
Povp.	4086	3635	1690	1313	8,57	7,14	10,00	8,57	15,71	25,71

Tabela 24: Podatki obremenitve tabel pomožne knjige

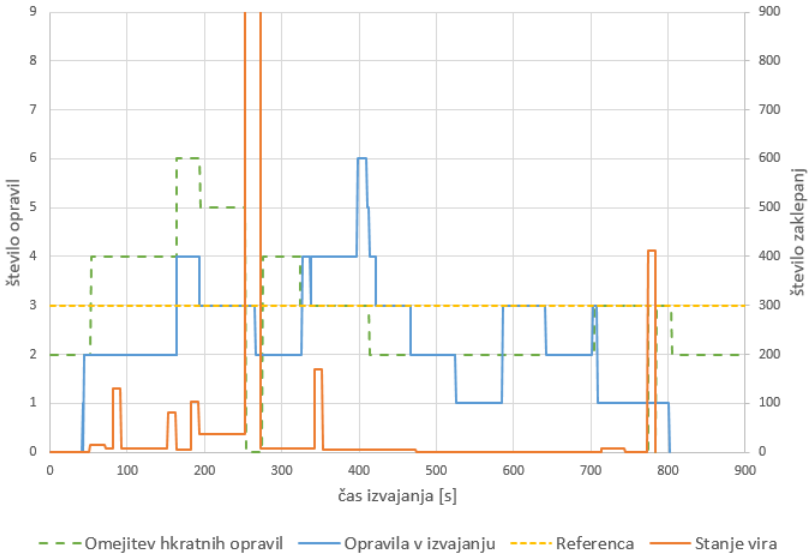
Uporabi razporeditev A in D izstopata z nizko maksimalno obremenitvijo okoli 400 zaklepanj. Le-to je bistveno manj kot 4000 zaklepanj pri uporabi ostalih razporeditev. Po preučitvi podatkov o opravilih je bilo ugotovljeno, da je razlog za špice v obremenitvah kombinacija hkratnega izvajanja opravil knjiženja in opravila prenosa v glavno knjigo. Do nenadne visoke obremenitve pride ravno v času zaključka opravila prenosa v glavno knjigo. Le-to se zgodi v primeru razporeditve E v 266. sekundi, medtem ko sta v izvajanju dve opravili knjiženja računov (slika 73). V primerih A in D do tega pojava ne pride, ker se opravilo prenosa v glavno knjigo izvede pred opravili knjiženja računov. Ker so opravila v vseh primerih že v izvajanju predno pride do prvega minimalnega povečanja obremenitve (npr. 30 zaklepanj), tudi nastavitev nizke referenčne meje ne reši problema.



Slika 71: Obremenitev podatkovnih tabel pomožne knjige s kontrolo prevzema



Slika 72: Obremenitev podatkovnih tabel pomožne knjige brez kontrole prevzema



Slika 73: Obremenitev tabel pomožne knjige – razporeditev E

6.3.11 Obremenitev podatkovnih tabel glavne knjige

Pri spremljanju obremenitve skupine podatkovnih tabel glavne knjige so bili uporabljeni naslednji parametri:

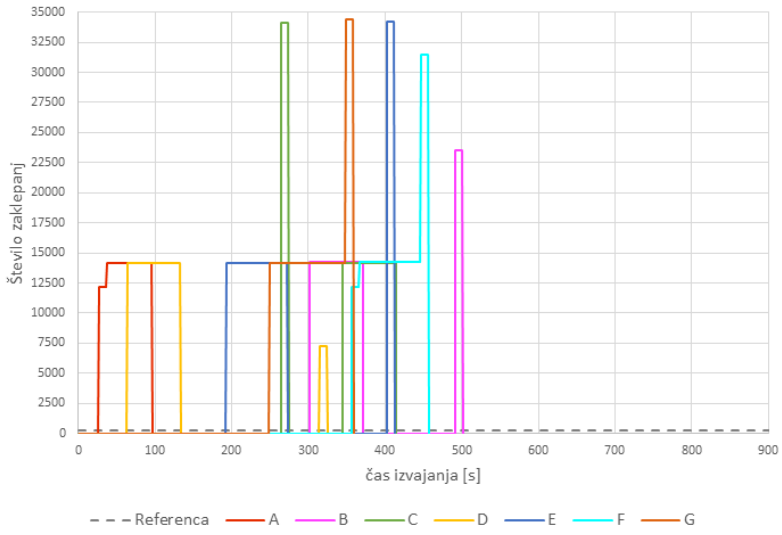
- referenca 300 zaklepanj;
- faktor povečanja velikosti 0,25;
- faktor manjšanja velikosti 1,00;
- stopnja propustnosti 1 obdelava;
- perioda propustnosti 4 cikle (40 sekund ob 10-sekundnem intervalu posodabljanja);
- maksimalna velikost vedra 8 obdelav;
- minimalna velikost vedra 0 obdelav;
- maksimalni dinamični prag 50 zaklepanj.

Skupino tabel obremenjujejo opravilo knjiženja temeljnice plačil in opravilo prenosa pomožne knjige v glavno knjigo. Na slikah 74 in 75 so prikazani rezultati meritev ob uporabi kontrole prevzema opravil in brez nje. Iz podatkov prekoračitev v tabeli 25 ni razvidne bistvene razlike glede na uporabo s kontrolo prevzema in brez nje. Visoke prekoračitve v razredu od 14100 do 14300 zaklepanj se pojavijo v obeh primerih in so posledica prenosa pomožne knjige v glavno knjigo. Razlogi za prekoračitve nad 14300 zaklepanj so posledica zaključka opravila knjiženja temeljnice, ki povzroči špice v obremenitvi. Ker opravilo temeljnice povzroči visoko obremenitev šele, ko je že dalj časa v izvajanju, kontrola prevzema vedno ne prepreči prevzema dodatnega opravila. Pri tem podatkovnem viru bi bilo možno omejiti velikost vedra na 1 in s tem preprečiti sočasno izvajanje opravil, ki ta vir obremenjujejo.

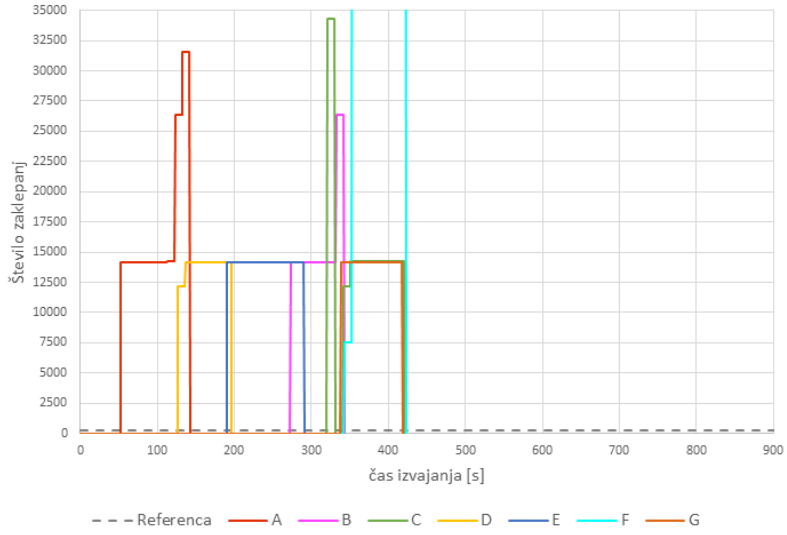
Razp.	**Maksimalna vrednost**		**Povprečna prekoračitev**		**Čas prekoračitve 300–14100 zaklepanj**		**Čas prekoračitve 14100–14300 zaklepanj**		**Čas prekoračitve nad 14300 zaklepanj**	
Kontr.	DA	NE	DA	NE	DA	NE	DA	NE	DA	NE
A	14197	31541	13605	17184	10		60	70		20
B	23485	26361	15077	15642			70	60	10	10
C	34079	34312	16381	15914		10	70	70	10	10
D	14197	14196	13033	13607	10	10	70	60		
E	34214	14206	16121	13906			80	100	10	
F	31455	104886	15445	92414	10	10	80		10	70
G	34400	14203	15732	13903			100	80	10	
Povp.	26575	34244	15057	26081	4,29	4,29	75,71	62,86	7,14	15,71

Tabela 25: Podatki obremenitve tabel glavne knjige

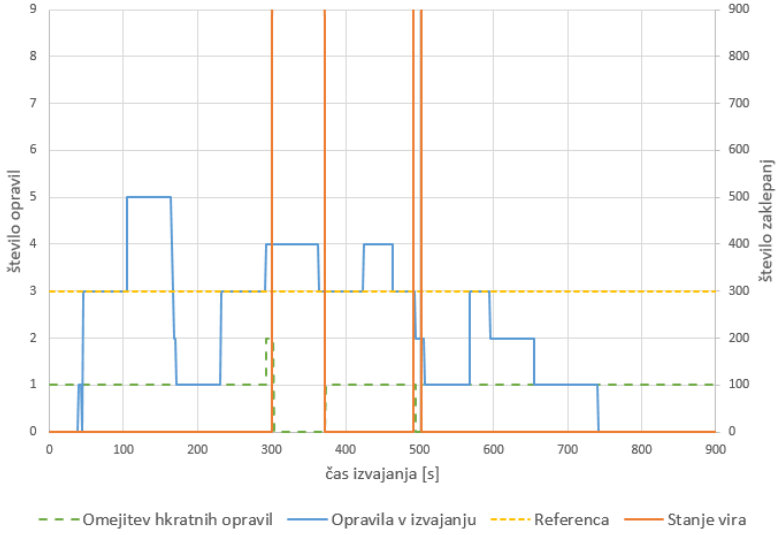
Na sliki 76 je podrobneje prikazano delovanje kontrole prevzema na primeru uporabe razporeditve B. Pri tem pride do omejitve prevzema novih opravil ob prekoračitvi referenčne meje v 302. sekundi zaradi obdelave prenosa pomožne knjige v glavno knjigo in 492. sekundi zaradi obdelave knjiženja temeljnice.



Slika 74: Obremenitev podatkovnih tabel glavne knjige s kontrolo prevzema



Slika 75: Obremenitev podatkovnih tabel glavne knjige brez nadzora



Slika 76: Obremenitev tabel glavne knjige – razporeditev B

7 Vrednotenje rezultatov

Uporaba generičnih paketnih obdelav pri simulaciji obremenitve enega vira je pokazala, da algoritem kontrole prevzema deluje pravilno in učinkovito. Pri tem je algoritem tako pri sistemskem kot pri podatkovnem viru uspel zadržati nizke prekoračitve podane referenčne meje.

Izvedba simulacije je tudi pokazala dve težavni situaciji, ki sta povezani z naravo delovanja kontrole prevzema na podlagi povratne informacije in samim problemom sprotnega razporejanja paketnih opravil. Pri prvi lahko pride do prevelike prekoračitve referenčne meje, ker se prevzame preveč opravil zaradi stopnje propustnosti večje od 1. Pri drugi lahko pride do prevzema novih opravil, ko je trenutna obremenitev vira že tik pod referenčno mejo. Obe situaciji sta neizbežni, saj algoritem razpolaga le z informacijo, katere vire posamezno opravilo obremenjuje, in nima informacije, v kakšni meri in koliko časa jih bo opravilo obremenjevalo. Zmanjšanje stopnje propustnosti po eni strani poveča čas in s tem možnost za zajem povratne informacije o povečani obremenitvi. Po drugi strani pa vsaka odložitev prevzema opravila pomeni čakanje dodatnih 60 sekund in s tem povečanje časa, ki je potreben za izvedbo vseh opravil.

Uporaba realnih paketnih obdelav na večjem naboru virov je pokazala, da je učinkovitost algoritma odvisna od samega vira in paketnih opravil, ki ta vir obremenjujejo. Prvi prispevek algoritma se odraža v tem, da je kontrola prevzema zagotovila dovolj nizke obremenitve virov, da so se lahko uspešno izvedla vsa paketna opravila. V primerih brez kontrole prevzema je namreč opravilo obdelave konsolidacije računov vedno končalo v napaki z delno opravljenim delom. Glede na uspešnost algoritma pri obvladovanju preobremenitev virov lahko obravnavane vire razdelimo v tri skupine:

- vire z nizkimi prekoračitvami referenčne meje;
- vire z visokimi prekoračitvami referenčne meje, ki jih kontrola prevzema obvladuje;
- vire z visokimi prekoračitvami referenčne meje, pri katerih kontrola prevzema ne more omejiti obremenitve in lahko prepreči le prevzem novih opravil, ki bi dodatno povečala obremenitev.

V prvo skupino spadajo viri, kot so CPE, podatkovne tabele računa, podatkovne tabele dnevnika računa in podatkovne tabele davčnih postavk. Pri teh virih se je uporaba kontrole prevzema izkazala za učinkovito, saj uspešno zadržuje čas prekoračitve v razredih, ki predstavljajo nizko prekoračitev referenčne meje. Razlogi za občasne višje prekoračitve izhajajo iz prevzema preveč opravil zaradi stopnje propustnosti in razlik med opravili v smislu intenzivnosti obremenjevanja vira. Le-to so sprejemljive situacije.

V drugo skupino spada vir podatkovnih tabel dokumenta. Pri tem viru kontrola prevzema zameji obremenitev v razredu, ki predstavlja občutno višje obremenitve glede na referenčno mejo. Vendar so obremenitve bistveno nižje kot v primeru brez uporabe kontrole prevzema. Razlogi so v velikem številu opravil, ki zelo različno intenzivno obremenjujejo izbrani vir. Tako se lahko v nekem trenutku izvajajo štiri opravila, ki povzročajo srednjo obremenjenost vira glede na referenčno mejo. Za tem pa sta prevzeti dve opravili, ki povzročita visoko

obremenitev vira. Le-to povzroči nenadno prekoračitev, ki je sicer večja, a jo algoritem vseeno uspe zamejiti. Pri preučevanju visoke obremenitve pri razporeditvi C je bilo odkrito, da se kontrola prevzema lahko prepozno odziva na dogodke zaključka paketnega opravila. Tako lahko pride znotraj intervala med dvema posodobitvama velikosti vedra do situacije, ko se zaključi več opravil hkrati in zatem prevzame več novih opravil. V tem primeru število novih opravil lahko ni skladno s stopnjo propustnosti, ker je velikost vedra ostala enaka tisti pred zaključkom prejšnjih opravil.

V tretjo skupino spadajo viri, kot so trdi disk, podatkovne tabele temeljnice, podatkovne tabele bančnih postavk, podatkovne tabele postavk kupcev, podatkovne tabele pomožne knjige in podatkovne tabele glavne knjige. Pri teh virih kontrola prevzema ne uspe zamejiti obremenitve v nekem sprejemljivem območju bodisi zaradi načina delovanja paketnih opravil bodisi zaradi vira samega. Obremenitev trdega diska lahko povzročijo tudi drugi dejavniki testnega okolja, na katere kontrola prevzema nima vpliva in jih zato ne more omejiti. Pri podatkovnih virih tabel temeljnice, postavk kupcev in glavne knjige je razlog za visoke obremenitve način obdelave podatkov. Opravilo paketne obdelave knjiženja temeljnice obdeluje veliko število zapisov v eni transakciji. Ker je to le eno opravilo, kontrola prevzema nima možnosti omejevanja, kot je to možno pri obdelavah, ki so sestavljene iz več opravil. Pri podatkovnih tabelah pomožne knjige povzroči špice v obremenitvi sočasno izvajanje opravila knjiženja računov in opravila prenosa pomožne knjige v glavno knjigo. Ker pri tem pride do prehoda iz nizke obremenitve v visoko obremenitev šele, ko so opravila že dalj časa v izvajanju, kontrola prevzema situacije ne prepreči. Ta problem bi lahko rešili z implementacijo novega vira, ki bi preprečeval hkratno izvajanje obdelav knjiženja računov in prenosa v glavno knjigo. Pri zasnovi algoritma je bila prekinitev izvajajočega opravila izpostavljena kot nezaželena, ker se pri tem zavrže opravljeno delo. Zato je edina možna reakcija na navedene prekoračitve referenčne meje omejitev prevzema novih opravil, ki ta vir obremenjujejo.

Pri obvladovanju več virov se je pokazalo, da med prekoračitvami referenčne meje določenih virov obstajajo odvisnosti. Tako proces podatkovnega strežnika ne povzroča visokih obremenitev CPE, ker kontrola prevzema zameji obremenitev podatkovnih virov in s tem delo na podatkovnih tabelah. Ko na več virov vpliva isti nabor paketnih opravil, pride do podobnosti v obremenitvah. Tako pri podatkovnih virih postavk temeljnice, postavk kupcev in glavne knjige dobimo daljše visoke obremenitve zaradi obdelave knjiženja temeljnice plačil. Pri podatkovnem viru davčnih postavk pa večina obremenitev ne doseže referenčne meje, ker paketna opravila knjiženja računov omeji kontrola prevzema zaradi bolj obremenjenega podatkovnega vira dnevnika računov.

Na podlagi pridobljenih rezultatov in upoštevanja predpostavke, da algoritem ne razpolaga z informacijo o času in intenziteti obremenitve, lahko potrdim pravilnost delovanja rešitve. Ker se ne more prevzeti več opravil, kot jih je določila zadnja posodobitev velikosti vedra, prepozen odziv na zaključek opravila ne predstavlja bistvene neskladnosti z zasnovo algoritma. Glede učinkovitosti algoritma pri obvladovanju preobremenitve posameznega vira lahko povzamem, da je le-ta odvisna od paketnih opravil, ki vir obremenjujejo. Če je le-teh več in vsak od njih posamezno povzroča nizko obremenitev, je algoritem kontrole prevzema zelo učinkovit. Višja kot je obremenitev, ki jo lahko posamezno opravilo povzroči, višja je lahko

prekoračitev referenčne meje. Namreč kontrola prevzema lahko le vpliva na to, ali bo opravilo prevzeto ali ne. Posledica prevzema enega novega opravila, ki lahko povzroči visoko obremenitev, pa je lahko visoka prekoračitev referenčne meje. Kot so pokazali rezultati, v vseh takšnih primerih kontrola prevzema zaustavi prevzem novih opravil in s tem prepreči poslabšanje situacije. Če upoštevam navedeno, je rešitev v vseh primerih učinkovita pri obvladovanju obremenitev virov sistema AX, ki jih lahko povzroči izvajanje paketnih obdelav.

8 Zaključek

V magistrski nalogi je bil obravnavan problem preobremenitev virov ERP Microsoft Dynamics AX, ki nastanejo kot posledica izvajanja zahtevnih paketnih obdelav. Kot rešitev za navedeni problem je bila predlagana razširitev algoritma za razporejanje paketnih obdelav s konceptom obvladovanja preobremenitev virov. Predlagana rešitev je bila nato tudi zasnovana in preizkušena na testnem okolju.

Pred pričetkom zasnove rešitve je bila najprej izvedena analiza sistema in bremena paketnih obdelav na testnem okolju. S tem sem prišel do ključnih podatkov o virih sistema in paketnih obdelavah. Pridobljene podatke sem nato uporabil pri postavitvi koncepta spremljanja virov in opredelitvi umetnega bremena za preverjanje ustreznosti delovanja rešitve. Pri tem sem vire sistema razdelil na sistemske vire, katerih obremenitve spremljamo z uporabo performančnih števcev, in podatkovne vire, katerih obremenitve spremljamo z uporabo štetja zahtev za zaklepanja na podatkovnih tabelah. Kot umetno breme sem opredelil vnaprej pripravljene razporeditve izbranih paketnih obdelav, ki so s stališča obremenitve virov neugodne.

Rešitev obvladovanja preobremenitev, ki sem jo zasnoval, je bila vgrajena v obstoječi algoritem za razporejanje paketnih obdelav v obliki kontrole prevzema opravil. Pri tem kontrola prevzema prilagaja prevzem novih opravil na podlagi identifikatorja razreda paketnega opravila v povezavi s podatkoma uporabe vira in obremenitve vira. Za periodično zbiranje podatkov o obremenitvi virov je bila pripravljena paketna obdelava, ki teče na enem od aplikacijskih strežnikov. Z vpeljavo koncepta dinamičnega pragu v algoritem zbiranja podatkov pa sem poskrbel za učinkovito izločanje nepomembnih meritev.

Celotna rešitev je zgrajena kot ogrodje okoli standardnega modula za paketne obdelave, ki je razširljivo v smislu zbiranja podatkov o stanju vira in kontrolne funkcije. Tako je mogoče enostavno implementirati nove vrste virov in nove kontrolne funkcije. S tem sem podprl širitev nabora virov in prilagajanje specifikam posameznih postavitev sistema AX.

Izvedba meritev z uporabo umetnega bremena na testnem okolju je pokazala, da algoritem kontrole prevzema deluje v skladu s pričakovanji. Učinkovitost algoritma pri omejevanju obremenitev je odvisna od samega vira in paketnih opravil, ki ta vir obremenjujejo. Kontrola prevzema uspešno vzdržuje nizke prekoračitve referenčne meje, če vir obremenjuje več manjših paketnih opravil, ki kot posamezno opravilo povzročajo razmeroma nizko obremenitev. Ker kontrola prevzema le odloča, ali bo opravilo prevzeto ali ne, pri virih, ki jih obremenjuje le eno zahtevno opravilo, ni mogoče preprečiti visoke prekoračitve. V tem primeru algoritem nima možnosti, da bi visoko obremenitev porazdelil po času. S primeri uporabe pa sem pokazal, da ob večjih prekoračitvah referenčne meje kontrola prevzema uspešno zaustavi prevzem novih opravil in s tem prepreči poslabšanje situacije.

Pomanjkljivosti predlaganega pristopa, ki se odražajo v občasnih višjih obremenitvah, izhajajo predvsem iz predpostavke, da algoritem ne razpolaga z informacijami, koliko časa in kako močno bo posamezno opravilo obremenjevalo vir. Ker algoritem ve le, kateri vir bo opravilo obremenjevalo, lahko zaradi stopnje propustnosti večje od 1 prevzame več zahtevnih opravil.

S tem povzroči večjo prekoračitev referenčne meje, kot bi jo povzročil s prevzemom le enega zahtevnega opravila ali z izbiro alternativnega manj zahtevnega opravila. Navedeno predpostavko sem uporabil pri zasnovi algoritma, ker bi v realnih situacijah težko dobro ocenili navedene podatke. Le-ti so namreč odvisni od parametrov posamezne instance paketnega opravila.

Ena od možnih izboljšav obstoječega algoritma, ki bi reševala omenjeno pomanjkljivost, bi bila vpeljava dinamične stopnje propustnosti, ki bi se spreminjala glede na razmerje med trenutno in referenčno obremenitvijo. Drug možen pristop pa bi bila vpeljava uteži v zapise seznama opravil, ki obremenjujejo posamezen vir. Tako bi lahko s pomočjo uteži razlikovali manj zahtevna opravila od bolj zahtevnih.

Področje, ki bi mu kazalo posvetiti pozornost, je tudi enostavnost uporabe pripravljene rešitve. Uporaba rešitve zahteva inicialno nastavitev virov, katerih obremenitve želimo obvladovati. Le-to bi lahko uporabniku olajšali s predlogami, ki bi poskrbele za nastavitev sistemskih virov aplikacijskega in podatkovnega strežnika. Osnovni viri so pri tem bolj ali manj vedno enaki ne glede na posamezno postavitev sistema AX. Enako bi bile dobrodošle predloge za podatkovne vire standardnih tabel, ki se uporabljajo pri vseh postavitvah (npr. račun, postavke kupcev, glavna knjiga idr.). Prav tako bi bilo zasnovano ogrodje smiselno razširiti s funkcijo opozarjanja. Le-ta bi lahko skrbnika sistema opozarjala na predolge in previsoke obremenitve. Iz raziskovalnega vidika pa bi bil zanimiv tudi razvoj algoritma, ki bi na podlagi analize podatkov zgodovine obremenitev in zgodovine paketnih obdelav predlagal seznam opravil, ki obremenjujejo posamezen vir.

9 Priloge

9.1 Ukazi za pripravo paketnih opravil za izvajanje

Naslednje ukaze algoritem za razporejanje paketnih opravil uporabi za pripravo novih opravil za prevzem v izvajanje, ko ni na voljo nobenega pripravljenega opravila. Ukazi nastavijo status *pripravljen* (angl. *Ready*) pri opravilih, ki so trenutno v statusu *čakajoč* (angl. *Waiting*) in katerih obdelave so v statusu *izvajanja* (angl. *Executing*).

```
ttsbegin;

//There are no more available tasks and the user is asking for any task. Search for more tasks with
//dependencies
update_recordset batch setting Status = BatchStatus::Ready
where batch.Status == BatchStatus::Waiting
    && batch.ConstraintType == BatchConstraintType::Or
exists join batchJob
    where batchJob.Status == BatchStatus::Executing
        && batch.BatchJobId == batchJob.RecId
exists join constraintsOr
    where constraintsOr.BatchId == batch.RecId
exists join batchDependsOr
  where
 (
    ((batchDependsOr.Status == BatchStatus::Finished
        && (constraintsOr.ExpectedStatus == BatchDependencyStatus::Finished
            || constraintsOr.ExpectedStatus == BatchDependencyStatus::FinishedOrError))
    || (batchDependsOr.Status == BatchStatus::Error
        && (constraintsOr.ExpectedStatus == BatchDependencyStatus::Error
            || constraintsOr.ExpectedStatus == BatchDependencyStatus::FinishedOrError)))
    && constraintsOr.DependsOnBatchId == batchDependsOr.RecId
 );


update_recordset batch setting Status = BatchStatus::Ready
where batch.Status == BatchStatus::Waiting
    && batch.ConstraintType == BatchConstraintType::And
exists join batchJob
    where batchJob.Status == BatchStatus::Executing
        && batch.BatchJobId == batchJob.RecId
notexists join constraintsAnd exists join batchDependsAnd
where
 (
    constraintsAnd.DependsOnBatchId == batchDependsAnd.RecId
    && constraintsAnd.BatchId == batch.RecId
    && ((batchDependsAnd.Status != BatchStatus::Finished && batchDependsAnd.Status != BatchStatus::Error)
        || (constraintsAnd.ExpectedStatus == BatchDependencyStatus::Finished
            && batchDependsAnd.Status == BatchStatus::Error)
            || (constraintsAnd.ExpectedStatus == BatchDependencyStatus::Error
            && batchDependsAnd.Status == BatchStatus::Finished))
 );

if(Session::isServer())
{

    //Update the root tasks for all Executing jobs
    //This is important for runtime tasks
    update_recordset batch setting Status = BatchStatus::Ready
    where batch.Status == BatchStatus::Waiting
        exists join batchJob
            where batchJob.Status == BatchStatus::Executing
                && batchJob.RecId == batch.BatchJobId
        notexists join constraintsOr
            where constraintsOr.BatchId == batch.RecId;
}

ttscommit;
```

9.2 Primer izločanja meritev z dinamičnim pragom

Primer A			Primer B		
Čas meritve	**48 meritev**	**15 objav**	**Čas meritve**	**42 meritev**	**11 objav**
0	0,00	0,00	**0**	0,00	0,00
4	0,00		**5**	0,00	
14	0,00		**15**	0,00	
24	0,00		**25**	1,52	
35	0,00		**35**	0,00	
44	89,06	89,06	**45**	126,15	126,15
54	83,07		**55**	126,15	
64	89,23	89,23	**65**	129,23	129,23
74	94,29	94,29	**75**	133,44	133,44
84	83,28	83,28	**85**	128,70	128,70
94	85,44		**95**	126,15	
104	169,22	169,22	**105**	212,61	212,61
114	171,51		**116**	212,31	
124	174,24		**125**	210,77	
134	172,72		**135**	210,77	
144	167,68		**145**	220,07	
155	170,01		**155**	215,39	
164	169,22		**165**	255,38	255,38
174	178,79		**175**	261,19	
184	172,31		**185**	259,60	
194	164,62		**195**	263,99	
204	170,77		**205**	257,57	
214	175,38		**215**	258,33	
224	84,85	84,85	**225**	218,46	
234	86,15		**235**	210,22	210,22
244	89,39		**246**	211,44	
254	89,39	89,39	**256**	217,92	
264	86,15		**266**	217,70	
275	86,02		**276**	215,39	
284	175,75	175,75	**286**	124,61	124,61
294	171,53		**296**	127,69	
304	175,75		**306**	130,17	
314	170,76		**316**	127,69	
324	174,33		**326**	127,27	
334	167,80		**336**	124,24	
344	170,77		**346**	89,39	
354	169,69		**356**	85,34	
364	177,27		**366**	91,32	91,32
374	172,72		**377**	86,15	
384	177,28	177,28	**386**	89,39	
395	175,38	175,38	**396**	84,62	
404	175,38		**406**	6,06	6,06
414	169,23	169,23			
424	172,30				
434	179,32	179,32			
444	171,36	171,36			
454	170,77				
464	0,00	0,00			

9.3 Časi izvajanja opravil pri primerih uporabe

9.3.1 Razporeditev A

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Knjiženje plačil (korak 1)	21	21	0
Knjiženje rač. (korak 1)	21	22	1
Obračun (korak 1)	21	26	5
Konsolidacija	22	403	381
Prenos v glavno knjigo	26	96	70
Obračun (korak 2)	91	156	65
Obračun (korak 2)	91	156	65
Knjiženje plačil (korak 2)	91	167	76
Obračun (korak 2)	96	161	65
Obračun (korak 2)	157	207	50
Obračun (korak 2)	161	211	50
Obračun (korak 2)	272	320	48
Obračun (korak 2)	272	321	49
Obračun (korak 2)	321	367	46
Knjiženje računov (korak 2)	367	452	85
Knjiženje računov (korak 2)	452	551	99
Knjiženje računov (korak 2)	463	564	101
Knjiženje računov (korak 2)	463	564	101
Knjiženje računov (korak 2)	564	653	89
Knjiženje računov (korak 2)	625	720	95
Knjiženje računov (korak 2)	625	723	98
Knjiženje računov (korak 2)	723	809	86

9.3.2 Razporeditev B

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Obračun (korak 1)	39	44	5
Obračun (korak 2)	45	99	54
Obračun (korak 2)	45	100	55
Obračun (korak 2)	45	100	55
Obračun (korak 2)	99	165	66
Obračun (korak 2)	100	167	67
Obračun (korak 2)	100	168	68
Obračun (korak 2)	105	171	66
Obračun (korak 2)	105	171	66
Knjiženje rač. (korak 1)	171	173	2
Konsolidacija	173	464	291
Knjiženje računov (korak 2)	232	396	164
Knjiženje računov (korak 2)	232	398	166
Prenos v glavno knjigo	292	364	72
Knjiženje plačil (korak 1)	396	396	0
Knjiženje računov (korak 2)	396	509	113
Knjiženje računov (korak 2)	398	507	109
Knjiženje plačil (korak 2)	424	494	70
Knjiženje računov (korak 2)	509	595	86
Knjiženje računov (korak 2)	568	655	87
Knjiženje računov (korak 2)	568	656	88
Knjiženje računov (korak 2)	656	741	85

9.3.3 Razporeditev C

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Konsolidacija	48	425	377
Obračun (korak 1)	108	113	5
Obračun (korak 2)	114	164	50
Obračun (korak 2)	114	164	50
Obračun (korak 2)	164	214	50
Obračun (korak 2)	164	215	51
Knjiženje plačil (korak 1)	214	214	0
Knjiženje plačil (korak 2)	215	268	53
Obračun (korak 2)	268	328	60
Obračun (korak 2)	275	336	61
Obračun (korak 2)	275	336	61
Obračun (korak 2)	328	383	55
Knjiženje rač. (korak 1)	335	337	2
Prenos v glavno knjigo	335	411	76
Knjiženje računov (korak 2)	337	514	177
Knjiženje računov (korak 2)	337	514	177
Knjiženje računov (korak 2)	337	516	179
Knjiženje računov (korak 2)	516	605	89
Knjiženje računov (korak 2)	575	697	122
Knjiženje računov (korak 2)	575	698	123
Knjiženje računov (korak 2)	698	809	111
Knjiženje računov (korak 2)	758	864	106

9.3.4 Razporeditev D

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Obračun (korak 1)	60	66	6
Prenos v glavno knjigo	60	134	74
Obračun (korak 2)	67	125	58
Obračun (korak 2)	67	127	60
Obračun (korak 2)	67	127	60
Obračun (korak 2)	127	175	48
Obračun (korak 2)	175	222	47
Obračun (korak 2)	194	239	45
Knjiženje plačil (korak 1)	222	222	0
Obračun (korak 2)	222	273	51
Obračun (korak 2)	239	295	56
Knjiženje rač. (korak 1)	254	257	3
Knjiženje plačil (korak 2)	254	320	66
Konsolidacija	254	547	293
Knjiženje računov (korak 2)	257	364	107
Knjiženje računov (korak 2)	364	463	99
Knjiženje računov (korak 2)	381	484	103
Knjiženje računov (korak 2)	463	555	92
Knjiženje računov (korak 2)	545	636	91
Knjiženje računov (korak 2)	545	636	91
Knjiženje računov (korak 2)	636	721	85
Knjiženje računov (korak 2)	721	805	84

9.3.5 Razporeditev E

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Knjiženje rač. (korak 1)	42	43	1
Knjiženje računov (korak 2)	44	119	75
Knjiženje računov (korak 2)	44	121	77
Knjiženje računov (korak 2)	119	190	71
Knjiženje računov (korak 2)	121	194	73
Knjiženje računov (korak 2)	164	317	153
Knjiženje računov (korak 2)	164	319	155
Prenos v glavno knjigo	190	266	76
Knjiženje plačil (korak 1)	317	317	0
Knjiženje računov (korak 2)	317	414	97
Knjiženje plačil (korak 2)	319	410	91
Obračun (korak 1)	327	337	10
Konsolidacija	327	642	315
Obračun (korak 2)	338	422	84
Obračun (korak 2)	398	467	69
Obračun (korak 2)	398	468	70
Obračun (korak 2)	468	525	57
Obračun (korak 2)	586	650	64
Obračun (korak 2)	586	651	65
Obračun (korak 2)	650	709	59
Obračun (korak 2)	651	709	58
Knjiženje računov (korak 2)	703	802	99

9.3.6 Razporeditev F

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Konsolidacija	52	381	329
Knjiženje rač. (korak 1)	112	113	1
Knjiženje računov (korak 2)	114	230	116
Knjiženje računov (korak 2)	114	232	118
Knjiženje plačil (korak 1)	293	293	0
Knjiženje računov (korak 2)	293	507	214
Knjiženje računov (korak 2)	293	509	216
Knjiženje plačil (korak 2)	340	442	102
Prenos v glavno knjigo	353	448	95
Knjiženje računov (korak 2)	509	596	87
Knjiženje računov (korak 2)	567	653	86
Knjiženje računov (korak 2)	567	657	90
Knjiženje računov (korak 2)	653	752	99
Obračun (korak 1)	657	663	6
Obračun (korak 2)	664	716	52
Obračun (korak 2)	664	717	53
Obračun (korak 2)	752	829	77
Obračun (korak 2)	778	861	83
Obračun (korak 2)	778	861	83
Obračun (korak 2)	829	893	64
Obračun (korak 2)	838	897	59
Obračun (korak 2)	838	898	60

9.3.7 Razporeditev G

Paketno opravilo	Čas začetka	Čas zaključka	Čas izvajanja
Obračun (korak 1)	10	15	5
Obračun (korak 2)	15	66	51
Obračun (korak 2)	15	66	51
Obračun (korak 2)	15	67	52
Obračun (korak 2)	66	112	46
Obračun (korak 2)	67	112	45
Obračun (korak 2)	112	166	54
Obračun (korak 2)	112	166	54
Obračun (korak 2)	127	178	51
Konsolidacija	127	546	419
Knjiženje plačil (korak 1)	239	239	0
Knjiženje rač. (korak 1)	239	241	2
Prenos v glavno knjigo	240	344	104
Knjiženje računov (korak 2)	241	497	256
Knjiženje računov (korak 2)	242	498	256
Knjiženje plačil (korak 2)	245	359	114
Knjiženje računov (korak 2)	546	659	113
Knjiženje računov (korak 2)	559	670	111
Knjiženje računov (korak 2)	559	671	112
Knjiženje računov (korak 2)	671	762	91
Knjiženje računov (korak 2)	731	825	94
Knjiženje računov (korak 2)	731	826	95

10 Viri in literatura

Literatura:

[1] P. Beckman, S. Nadella, N.Trebon, I. Beschastnikh, "SPRUCE: A System for Supporting Urgent High Performance Computing," v zborniku *Grid-Based Problem Solving Environments – IFIP TC2/ WG 2.5 Working Conference on Grid-Based Problem Solving Environments: Implications for Development and Deployment of Numerical Software July 17–21, 2006, Prescott, Arizona, USA,* str. 295–311.

[2] W. Chung, R. Chang, "A new mechanism for resource monitoring in Grid computing," *Future Generation Computer Systems,* št. 1, zv. 25, str. 1–7, 2009.

[3] J. Gray, A. Reuter, *Transaction Processing: Concepts and Techniques*, San Francisco: Morgan Kaufmann Publishers, 1993.

[4] R. Iyer, V. Tewari, K. Kant, "Overload Control Mechanisms for Web Servers," v zborniku *Performance and QoS of Next Generation Networking: Proceedings of the International Conference on the Performance and QoS of Next Generation Networking, P&QNet2000, Nagoya, Japan, November 2000*, str. 225–224.

[5] K. Kurowski, A. Oleksiak, W. Piatek, J. Węglarza, "Impact of Urgent Computing on Resource Management Policies, Schedules and Resources Utilization," *Procedia Computer Science,* zv. 9, str. 1713–1722, 2012.

[6] M.L. Massie, B.N. Chun, D.E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing,* št. 7, zv. 30, str. 817–840, 2004.

[7] Microsoft Corporation, *Development I in Microsoft Dynamics® AX 2012*, Julij 2011.

[8] Microsoft Corporation, *Development II in Microsoft Dynamics® AX 2012*, Avgust 2011.

[9] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems,* New York: Springer, 2012.

[10] M. Sherman, *Inside Microsoft Dynamics AX 2012,* Redmond (Washington): Microsoft Press, 2012.

[11] J. Sgall, "On-line scheduling," *Online Algorithms: The State of the Art, Lecture Notes in Computer Science*, zv. 1442, str. 196–231, 1998.

[12] J.A. Stankovic, C. Lu, S.H. Son, G. Tao, "The Case for Feedback Control Real-Time Scheduling," v zborniku *11th EuroMicro Conference on Real-Time Systems,* str. 11–20, 1999.

[13] T. Voigt, P. Gunningberg, "Handling Multiple Bottlenecks in Web Servers Using Adaptive Inbound Controls," v zborniku *Protocols for High Speed Networks: 7th IFIP/IEEE International Workshop, PfHSN 2002, Berlin, Germany, April 22–24, 2002*, str. 50–68, 2002.

[14] W. Wang, Y. Chang, W. Lo, Y. Lee, "Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments," The Journal of Supercomputing, št.2, zv. 66, str. 783–811, 2013.

[15] N. Zimic, M. Mraz, *Temeljni zmogljivosti računalniških sistemov*, Ljubljana: Fakulteta za računalništvo in informatiko, 2006.

Ostali viri:

[16] AOS Overview [AX 2012], https://msdn.microsoft.com/en-us/library/aa660629.aspx, (9.2.2014).

[17] AX 2012 CIL – How does it work?, http://axwonders.blogspot.com/2013/04/ax-2012-cil-how-does-it-work.html, (10.2.2014).

[18] Batch processing, http://en.wikipedia.org/wiki/ Batch_processing, (10. 4. 2015).

[19] Enterprise resource planning, http://en.wikipedia.org/wiki/Enterprise_resource_planning, (10. 4. 2015).

[20] How batch processing works under the hood AX2009, http://blogs.msdn.com/b/emeadaxsupport/archive/2011/02/22/how-batch-processing-works-under-the-hood-ax2009.aspx, (22. 4. 2015).

[21] Intel® Hyper.Threading Tchnology, http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html, (25. 4. 2014).

[22] Lock Modes, https://technet.microsoft.com/en-us/library/ms175519(v=sql.105).aspx, (25. 4. 2014).

[23] Mainframes working after hours: Batch processing, http://publib.boulder.ibm.com/infocenter/zos/basics/index.jsp?topic=/com.ibm.zos.zmainframe/zconc_batchproc.htm, (30. 1. 2014).

[24] MECOMS™ | Meter Data Management & Customer Information System, http://www.mecoms.com, (17. 4. 2015).

[25] Microsoft Dynamics AX: Comprehensive, Industry-Specific ERP Solution, http://www.microsoft.com/en-us/dynamics/erp-ax-overview.aspx, (17. 4. 2015).

[26] Microsoft SharePoint, http://en.wikipedia.org/wiki/Microsoft_SharePoint, (9. 2. 2014).

[27] PerformanceCounter Class, https://msdn.microsoft.com/en-us/library/system.diagnostics.performancecounter%28v=vs.110%29.aspx, (19. 4. 2015).

[28] Plan system topology [AX 2012], http://technet.microsoft.com/en-us/library/dd309725.aspx, (25. 4. 2014).

[29] sys.dm_tran_locks (Transact-SQL), https://msdn.microsoft.com/en-us/library/ms190345(v=sql.110).aspx, (25. 4. 2014).

[30] System monitor, http://en.wikipedia.org/wiki/System_monitor, (19. 4. 2015).

[31] Using the Microsoft Dynamics AX Add-in for Excel [AX 2012], https://technet.microsoft.com/en-us/library/hh781099.aspx, (3. 5. 2015).

[32] Windows Performance Monitor, http://technet.microsoft.com/en-us/library/cc749249.aspx, (25. 4. 2014).