FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Tvorba nástrojů pro simulaci uživatelské aktivity za účelem optimalizace hardwaru mailového kolaboračního řešení
Student:	Martin Zvelebil
Vedoucí:	Ing. Tomáš Vondra, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Manažerská informatika 2021
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

Cílem bakalářské práce je zajistit optimalizaci hardwarové architektury pro firmu zabývající se tvorbou mailové kolaborativní aplikace a následným návrhem vhodného onpremise řešení pro velké zákazníky. Optimalizace by měla být vedena vzhledem k počtu uživatelů, typickému zatížení, ceně a zaručené době odezvy jednotlivých součástí infrastruktury. Této optimalizace bude docíleno tvorbou nástrojů, které budou simulovat zatížení příslušné části infrastruktury – jednak klientské (webový a desktopový klient) a zároveň serverové (SMTP a IMAP servery). Součástí práce také bude analýza typického rozložení velikostí e-mailů ve schránkách a na základě analýzy bude postaven z důvodů ochrany uživatelů program na tvorbu syntetických e-mailových schránek umožňujících testování. Na základě simulace chování uživatelů nad syntetickými daty pak bude možné odhadnout hardwarové nároky na serverovou část a aproximovat počet požadavků, které zvládne server při daném hardware odbavit a tím minimalizovat investiční a provozní náklady.

1. Popište problematiku, identifikujte slabá místa v současných interních procesech a řešeních, které se ve firmě používají.
2. Proveďte průzkum nástrojů, které umožní simulovat chování uživatelů.
3. Navrhněte a implementujte programy na simulaci a měření uživatelské aktivity v daných částech infrastruktury s ohledem na ochranu uživatelských dat.
4. Navrhněte a implementujte postup, jak na základě simulace chování uživatelů zlepšit kvalitu návrhu (ve smyslu optimální architektury vzhledem k počtu uživatelů, očekávané odezvy a ceně).

Elektronicky schválil/a Ing. David Buchtela, Ph.D. dne 24. října 2023 v Praze.

5. Postup ověřte na základě vlastní simulace a porovnání s daty z již realizovaných projektů.
6. Vyhodnoťte přínos Vašeho návrhu z hlediska očekávaných benefitů pro firmu a ušetřených prostředků (TCO) v porovnání s původním řešení

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Tvorba nástrojů pro simulaci uživatelské aktivity za účelem optimalizace hardwaru mailového kolaboračního řešení

Martin Zvelebil

Fakulta informačních technologií ČVUT v Praze
Katedra softwarového inženýrství
Vedoucí práce: Ing. Tomáš Vondra, Ph.D.

12. května 2024

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dne 12. května 2024 ……

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Zvelebil, Martin. *Tvorba nástrojů pro simulaci uživatelské aktivity za účelem optimalizace hardwaru mailového kolaboračního řešení.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2024

Abstrakt

Tato bakalářská práce řeší problém volby vhodného hardwaru využívaného pro běh mailového kolaboračního řešení firmy IceWarp. Součástí bakalářské práce je zhodnocení aktuálního řešení a návrhy na jeho vylepšení díky simulaci chování uživatelů napříč různými prostředími, kde lze aplikaci využívat. Tvorba programů pro simulaci uživatelské aktivity je založena na analýze chování reálných uživatelů, kdy simulované operace jsou vždy vybírány s ohledem na dané prostředí, které využívá specifickou množinu protokolů pro zajištění své funkcionality. Simulaci uživatelských aktivit zajišťuje implementace třívrstvé architektury programů, jejíž přesnost je následně optimalizována vůči produkčnímu prostředí pomocí metod pro porovnávání časových řad. Doprovodným prostředkem pro tuto optimalizaci je monitorování produkčního i simulačního prostředí aplikací Zabbix. Vzhledem k nutnosti ochrany dat reálných uživatelů, tak jsou pro servery simulující uživatelskou aktivitu zajištěna syntetická data, pro jejichž tvorbu je implementován program, který na základě statistických vlastností dat reálných uživatelů z produkčního prostředí vytváří syntetická data se stejnými vlastnostmi. Cílem práce je zajistit úsporu zdrojů při škálování serverů na kterých je aplikace IceWarp nainstalována a umožnit firmě IceWarp lépe poznat chování svých uživatelů. Samotné řešení nabízí možnost simulaci opakovat na jakémkoliv dostupném hardwaru s různou uživatelskou distribucí napříč všemi dostupnými prostředími, ve kterých lze aplikaci IceWarp využívat.

Klíčová slova: hardwarové požadavky aplikace, náročnost aplikace, simulace uživatelské aktivity, tvorba syntetických dat, e-mail, analýza chování uživatelů, Python, Bash, Zabbix

Abstract

This Bachelor's Thesis addresses the problem of choosing the appropriate hardware to run the IceWarp mail collaboration solution. It includes an evaluation of the current solution and suggestions for its improvements by simulating user behaviour across different environments where the application can be used. The programs for simulating user activity are created based on the analysis of real-life user behaviour, where the simulated operations are always selected considering the given environment that uses a specific set of protocols to ensure its functionality. The simulation of user activities is provided by the implementation of a three-layered program architecture, whose accuracy is then optimized against the production environment by using time series similarity methods. Monitoring the production and simulation environments in the Zabbix application is an additional means of this optimization. Due to the need to protect real-life user data, synthetic data is provided to the servers simulating user activity. The implemented program creates this data with the same properties as the statistical properties of real-life user data from the production environment. The Thesis´s goal is to provide a resource-saving solution for scaling the servers on which the IceWarp application is installed and to enable IceWarp to deepen the understanding of its users' behaviour. The solution offers the ability to repeat the simulation for any available hardware with different distributions of users across all available environments in which the application can be used.

Keywords: application hardware requirements, application intensity, user activity simulation, synthetic data generation, e-mail, user behaviour analysis, Python, Bash, Zabbix

Obsah

Seznam grafů

Seznam obrázků

Seznam úryvků kódu

Seznam tabulek

Úvod

Odhadování hardwarových požadavků aplikace bez jakýchkoliv analytických podkladů může vést k nevhodné volbě těchto požadavků a selhání aplikace. Proto se firma IceWarp Technology s.r.o. (dále již IceWarp) v rámci vydání nové verze své aplikace s názvem IceWarp Epos rozhodla pro tvorbu projektu na analýzu chování uživatelů, jejich následnou simulaci a tvorbu doporučení pro optimalizaci hardwaru. Firma IceWarp je tedy zadavatelem této bakalářské práce.

Produkt, který je firmou IceWarp nabízen je komplexní mailové kolaborativní řešení nabízející celou řadu funkcí od e-mailů, přes sdílené kalendáře, složky, dokumenty, týmové konverzace, konference a mnoho dalšího. Všechny z těchto komponent uživatel nalezne uvnitř jednoho okna libovolného internetového prohlížeče. Produkt lze také využívat na telefonních zařízeních přes ActiveSync protokol nebo v desktopových aplikacích jako je například Outlook či eM Client.

S rostoucím počtem zákazníků společnosti IceWarp se začala nabízet otázka, zdali se v rámci hardwaru ve firmě zbytečně neutrácí, nebo naopak, jestli nenastala situace, kdy projekt selhal z důvodů podhodnocení náročnosti produktu a na hardwaru byly prostředky nemístně ušetřeny. Cílem této práce je tuto otázku zodpovědět a dát společnosti IceWarp seznam doporučení na optimalizaci hardwaru, na kterém je aplikace nasazena, a to jak v cloudovém prostředí, tak on-premise prostředí, které je obvyklou volbou pro zákazníky v rámci velkých zakázek, například ve státním sektoru.

Této hardwarové optimalizace bude docíleno pomocí analýzy chování uživatelů, která bude prováděna především z již existujících logů nebo případně z dostupných dat jednotlivých uživatelů/serverů v IceWarp API. Na bázi této analýzy bude možné určit, jak se průměrný uživatel v jednotlivých prostředích chová, a jak je toto chování přenášeno na jednotlivé části aplikace (protokoly). Na těchto protokolech bude poté simulace provedena a vyhodnocena její správnost vůči reálným serverům s obdobným počtem uživatelů. Výstupy z této analýzy jsou dalším cílem práce, jelikož zajistí, že firma bude lépe znát chování svých uživatelů.

S ohledem na ochranu dat reálných uživatelů je nutné zajistit data syntetická, která ale budou vytvořena na základě reálných statistik. Z toho důvodu bude třeba vytvořit nástroj, který zajistí analýzu distribuce velikosti e-mailů ve schránkách u jednotlivých uživatelů, a vhodným způsobem zajistí tvorbu obdobných schránek se syntetickými e-maily, nad kterými bude simulace mailové části prováděna. U ostatních komponent kromě e-mailů nebudou syntetická data potřebná, jelikož data budou vytvářena přímo uvnitř simulačních programů, nebo budou menší vzorky reálných dat importovány. Jedná se tedy o další část, která pomůže firmě IceWarp lépe poznat její uživatele.

Po první fázi hrubé implementace bude provedena tvorba simulačního, monitorovacího a analyzačního prostředí. Pomocí naměřených dat nastane optimalizace implementovaných programů, aby se zajistila vhodná frekvence jednotlivých simulovaných operací a uvnitř simulačního prostředí bude celý systém otestován. Výsledky z těchto testů budou sloužit k výsledné optimalizaci hardwaru

napříč všemi cloudovými datacentry a poté také jako doporučení pro zákazníky on-premise.

Na závěr práce bude zhodnocena celková cenová náročnost řešení, složitost jeho údržby a procesní dopady ve firmě. Součástí této ekonomické kapitoly bude také výpočet TCO před a po optimalizaci hardwaru a shrnutí ostatních ekonomických dopadů, které nelze jednoduše monetárně vyjádřit.

1 Teorie

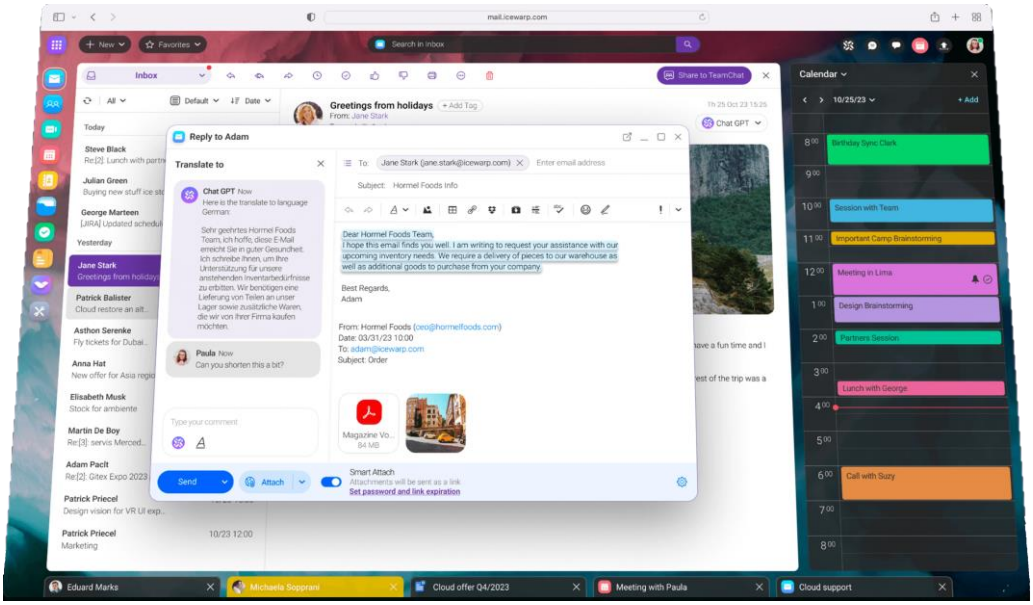
1.1 IceWarp

Firma IceWarp (dříve známa také jako Merak) má své hlavní sídlo v České republice, ale své pobočky má také v USA, Německu a Indii. [1] Jedná se o firmu poskytující e-mailové kolaborativní řešení, které kromě e-mailové elektronické komunikace obsahuje také celou řadu dalších funkcionalit. Produkt je nabízen jak v cloudovém řešení formou pravidelného měsíčního předplatného, tak s možností on-premise instalace. Forma on-premise instalací je obvykle preferovanou volbou v rámci zakázek ve státních sektorech, které lpí na ochranu dat svých uživatelů.

Celkově lze funkcionality produktu rozdělit na 8 hlavních komponent a to e-mail, TeamChat, konference, kalendář, kontakty, dokumenty (cloudové úložiště), úkoly a poznámky. Nad všemi těmito funkcionalitami je od vydání verze Epos v roce 2023 postaven centrální dashboard, který plně umožňuje přizpůsobení aplikace uživatelským představám. [2] Kromě výše zmíněných funkcionalit je IceWarp vybaven také umělou inteligencí a velkým množstvím integrací aplikací třetích stran, aby bylo možné vykonávat denní rutinu veškerých aplikací pouze uvnitř jednoho okna prohlížeče. IceWarp je možné využívat nejen ve webovém prohlížeči, ale také v desktopovém klientu nebo v mobilních zařízeních. Ve všech těchto případech se liší funkcionalita a dostupnost výše zmíněných komponent.

1.1.1. Webový klient (IceWarp WebClient)

Webový klient je hlavním prostředím, které je firmou pro užívání aplikace doporučováno, čímž se také jedná o jednu z nejkritičtějších částí produktu IceWarp z hlediska uživatelského vytížení. V případě e-mailové komponenty se jedná o běžnou e-mailovou schránku, srovnatelnou například se schránkou od Seznamu či Microsoft Outlook. IceWarp v této komponentě navíc nabízí možnost funkce SmartAttach, která umožňuje odeslání e-mailu s velkou přílohou, která je lokálně nahrána do souborů, a místo ní je sdílen pouze link v těle e-mailu. Tato funkcionalita zajišťuje jednoduché sledování a kontrolu nad sdíleným souborem. [3] Pro budoucí testy je důležité zmínit, že limit aplikace na velikost odeslaného e-mailu je 2 GB, ačkoliv většina e-mailových klientů mimo IceWarp by takový e-mail odmítlo, jelikož jejich limity jsou v řádech desítek MB. [4] TeamChat slouží ke správě a komunikaci týmů, obdobně jako tomu je v konkurenční aplikaci Microsoft Teams. Zde je také možné odesílat soubory, zprávy, odkazy a v reálném čase editovat dokumenty. V aplikaci je možné také spravovat své vlastní kontakty a pořádat online konference. Dále aplikace umožňuje sdílené kalendáře, soubory nebo fulltextové vyhledávání. Vše výše zmíněné včetně poznámek a úkolů lze pohodlně kontrolovat z jednoho místa – dashboard.



Obrázek 1: IceWarp WebClient [5]

1.1.2. Desktopový klient

Desktopový klient IceWarp Desktop Client (IWDC) je volně ke stažení na oficiálních stránkách firmy. [6] Jedná se o aplikaci, která obsahuje téměř všechny zásadní funkcionality, jako výše zmíněný webový klient, ale bývá žádanější od zákazníků přecházejících z Microsoft Outlook pro její větší podobnost s prostředím Outlooku.

1.1.3. Mobilní zařízení

Používání IceWarpu přes mobilní zařízení umožňuje nejomezenější počet funkcionalit. V minulosti bylo jediným možným řešením připojení přes ActiveSync neboli EAS vyvinutý společností Microsoft. Mobilní zařízení připojené přes tento protokol podporuje práci s e-maily, kalendáři, kontakty, úkoly a poznámkami uvnitř telefonu, ačkoliv ve velmi omezeném režimu, ve srovnání s funkcionalitami dostupnými z webového nebo desktopového klienta. [7] V posledních měsících se začalo pracovat na vývoji mobilní aplikace, která tento původní způsob užívání mobilního telefonu v souvislosti s aplikací nahradí, ale vzhledem k tomu, že je stále v začátcích svého vývoje, a tudíž obsahuje minimum funkcionalit, tak v rámci simulování uživatelské aktivity bude vynechána. [8]

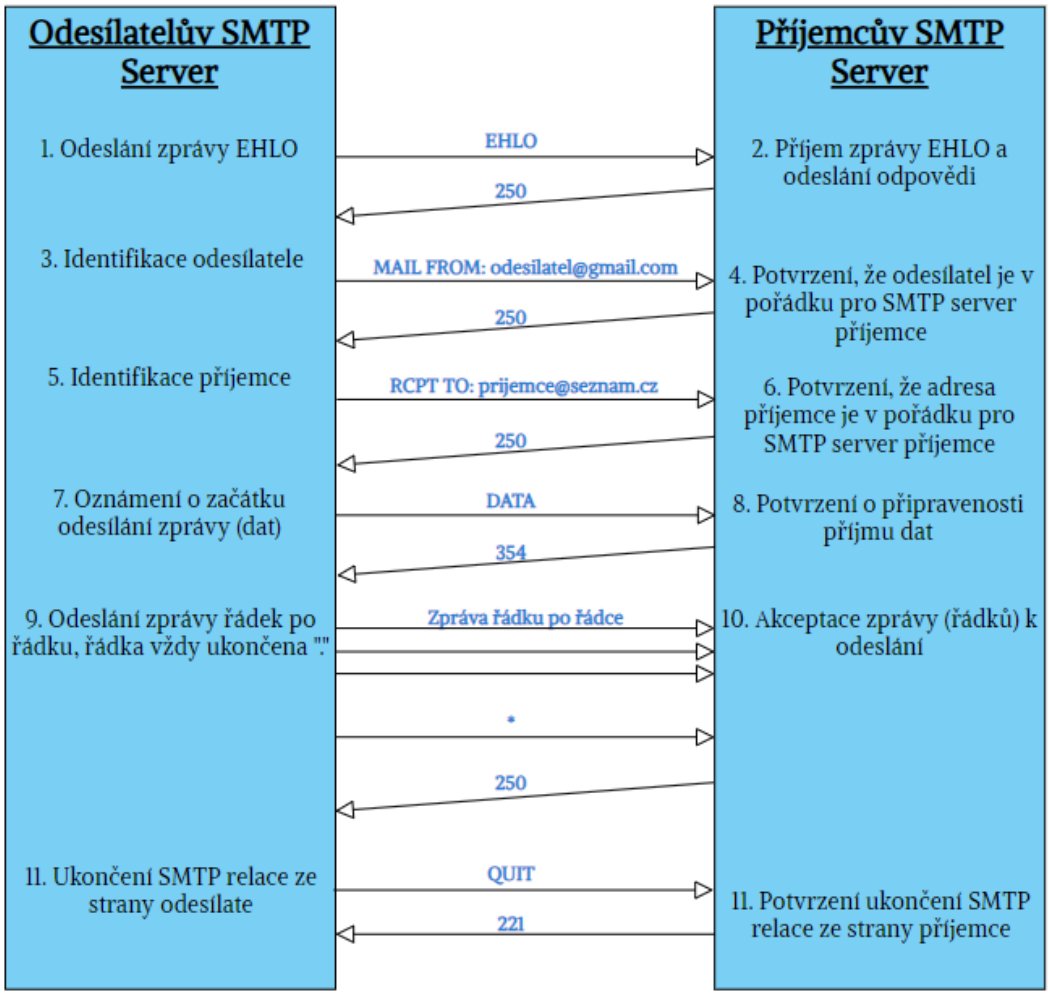
1.2 Protokoly využívané aplikací IceWarp

1.2.1. SMTP

SMTP je protokol z rodiny síťových protokolů TCP/IP komunikačního modelu operující na aplikační vrstvě. Protokol je využíván pro odesílání a přijímání e-mailů napříč SMTP servery. Klient, který má zájem o odeslání pošty otevře TCP spojení směrem k SMTP serveru příjemce. SMTP server obvykle naslouchá na portu 25 (v případě nezabezpečených spojení) nebo v případě zabezpečených SSL spojení na portu 465 či 587. Dané porty se liší dle implementací jednotlivých SMTP serverů, ale port 465 jakožto výchozí pro zabezpečené SSL spojení je již považován za zastaralý a je doporučeno využívat port 587. [9] V rámci SMTP komunikace se vykonávají mezi serverem a klientem sekvence dialogů požadavek-odpověď s pomocí specifických příkazů protokolu SMTP. [10] Během této komunikace jsou využívaní základní 4 agenti pro přenos pošty: [9]

1. **MUA** (Mail User Agent) – Aplikace, která zajišťuje odeslání pošty
2. **MSA** (Mail Submissions Agent) – Přijímá poštu od klienta a zajišťuje verifikaci hlaviček
3. **MTA** (Mail Transfer Agent) – Zajišťuje přenos e-mailu mezi jednotlivými SMTP servery. V případě moderních systémů MTA nahrazuje i funkcionalitu MSA. Z Linuxových operačních systémů je známým MTA například nástroj `sendmail`
4. **MDA** (Mail Delivery Agent) – Poslední agent před kompletním přijetím pošty na server. Zajišťuje doručení pošty do správné složky na serveru příjemce

Pokud komunikace proběhne úspěšně, SMTP server příjemce danou poštu uloží do svého úložiště. SMTP protokol již není zodpovědný za zobrazování pošty příjemcům z SMTP serveru.



Obrázek 2: Schéma interakce uvnitř protokolu SMTP

1.2.2. IMAP

IMAP, stejně jako SMTP, je protokol z rodiny TCP/IP komunikačního modelu operující na aplikační vrstvě. Jeho hlavní funkcionalitou je zajištění zobrazování pošty ze serveru příslušným klientům, kteří se vůči IMAP serveru autentizují. Oproti starším protokolům, jako například POP3, které zajišťovaly stejnou funkcionalitu, má tu výhodu, že umožňuje uživatelům přistupovat k e-mailům z jakéhokoliv zařízení. Pošta totiž není lokálně stahována, nýbrž IMAP zajišťuje přímý přístup k poště na serveru bez nutnosti lokální kopie. A právě z toho důvodu je v případě jeho využití jakákoliv změna v jednom poštovním klientu ihned viditelná i uvnitř jiných připojených klientů. Jedná se tedy o mezivrstvu mezi poštovním klientem a serverem. Jako jeho téměř jedinou nevýhodou je uváděna nutnost internetového připojení pro přístup k e-mailům. [11] IMAP využívá pro svou funkcionalitu port 143 (v případě nezabezpečených spojení) nebo v případě zabezpečených SSL spojení port 993. Stejně jako v případě SMTP je výrazně doporučeno používat port umožňující šifrování pomocí SSL. [12]

Klienty, které s protokolem IMAP pracují, implementují mnoho principů, které jsou před koncovými uživateli skryty. Pro jednotlivé operace s e-maily je nutné znát syntax příkazů pro práci s IMAP serverem. Každý z příkazů odpovídá následující syntaxi, kde tag je identifikátor daného příkazu. Tag je klientem vygenerovaná sekvence písmen a znaků sloužící pro jednoduchou identifikaci daného příkazu:

```
tag command [arguments]
```

Standardní příkazy pro práci s jednotlivými e-maily, které budou níže rozebrány, jsou obvykle dostupné ve své standardní variantě, a pak s klíčovým slovem *UID* před samotným příkazem. Je doporučováno využívat možností *UID*, jelikož se jedná o číselný identifikátor, který se v čase existence e-mailu nezmění. [13]

1.2.2.1. Připojení

Pro připojení k IMAP serveru lze využít nástrojů telnet/netcat nebo případně pro šifrované spojení nástroj OpenSSL uvnitř Linuxové příkazové řádky.

```
tag openssl s_client -crlf -connect imap_server:993
```

Po připojení na server je nutné se autentizovat, pro to slouží příkaz *LOGIN*.

```
tag LOGIN user@email.com password123
```

Po autentizaci je možné vykonávat veškeré IMAP operace nad schránkami daného uživatele. [14]

1.2.2.2. IMAP modseq, získávání obsahu schránek

Po přihlášení uživatele do klienta, a tím i tedy autentizaci na IMAP server, tak v případě většiny klientů je jako první zobrazena doručená pošta. Pro přechod mezi složkami slouží příkaz *SELECT*:

```
tag SELECT „INBOX“
```

Po přechodu do složky je typické, že si klient požádá IMAP server o synchronizaci obsahu aktuální složky. K tomu v případě modernějších klientů slouží hodnoty modseq (mod-sequence). Tento princip je součástí *CONDSTORE* rozšíření protokolu IMAP. Modseq je unikátní, kladná a nutně se zvyšující hodnota, kterou server přiřazuje každému z e-mailů při jejich příjmu a případně změní na novou hodnotu, při jakékoliv změně v metadatech daného e-mailu. [15]

Pokud složka, která byla v předchozím kroku zvolena, tak ještě nebyla uživatelem daného klienta v minulosti navštívena, je příkaz *SEARCH* k zjištění všech klientem neznámých e-mailů odeslán s hodnotou modseq rovno 1. V případě, že již klient složku eviduje, tak využije hodnotu *HIGHESTMODSEQ*, kterou si lokálně udržuje pro každou složku a požádá o všechny e-maily s modseq hodnotou vyšší nežli *HIGHESTMODSEQ*. [15]

```
tag UID SEARCH MODSEQ 1:*
```

Jakmile je klientem přijata odpověď, tak je dle specifik daného klienta zpracována, a po částech, či případně najednou, jsou dané e-maily vyzvednuté pomocí příkazu *FETCH*. Do závorek za samotný příkaz se specifikuje, jaká část e-mailu je požadována. Díky této specifikaci je možné vyzvednout pouze metadata (vlajky, modseq hodnota daného e-mailu apod.) bez nutnosti stahovat celý e-mail nebo naopak v případě, že je cílem získat celý obsah včetně hlaviček, může být využito formátu RFC 822. [16] Každý z klientů k tomuto přistupuje dle své vlastní implementace, a je tedy nutné počítat s tím, že některé klienty první zajišťují hlavičky a poté až tělo nebo dokonce jednotlivé části těla (bodyparts) zvlášť, což může zvyšovat zátěž vytvářenou na cílový server.

```
tag UID FETCH 1,8,10 (RFC822)
```

Po získání všech e-mailů je server klientem dotázán na aktuální *HIGHESTMODSEQ* hodnotu kterou si lokálně aktualizuje.

```
tag STATUS INBOX (HIGHESTMODSEQ)
```

Výše zmíněný scénář je typickou inicializací e-mailového klienta při přihlášení uživatele. Zmíněné příkazy nabízí širokou škálu dalších možností, kdy je možné například vyhledávat e-maily podle specifických příznaků, hlaviček apod. [13] Stejně tak lze přímo bez využití příkazu *SEARCH* požádat o e-maily od stanoveného modseq přímo uvnitř *FETCH*. [17] To vše již závisí na implementaci specifických e-mailových klientů.

1.2.2.3. Přesuny, mazání a kopírování e-mailů

Další ze zásadních funkcionalit e-mailových klientů jsou operace přesunů, kopírování a mazání jednotlivých e-mailů. V případě kopírování existuje v implementaci protokolu IMAP samostatný příkaz *COPY*.

```
tag UID COPY 1:10 destination_mailfolder
```

Pro mazání neexistuje ve standardní implementaci protokolu IMAP samostatný příkaz. Obvyklou klientskou implementací mazání e-mailů je přidání vlajky *\Deleted* s pomocí příkazu *STORE*. V tomto bodě může být implementace klientů různá, kdy v některých případech veškeré e-maily s vlajkou *\Deleted* jsou přesunuty do výchozí složky pro smazané e-maily, nebo jsou tam pouze zobrazovány uvnitř klienta, ale ve skutečnosti jsou stále ve své původní složce. V každém případě pro absolutní smazání e-mailu z IMAP serveru je nutné uvnitř složky, kde je e-mail fyzicky přítomný, odeslat příkaz *EXPUNGE*, který zajistí smazání veškerých e-mailů v dané složce, které mají vlajku *\Deleted*. [18] V případě, že je potřeba příkaz *EXPUNGE* pouze na specifický rozsah e-mailů, nikoliv na celou složku, stačí přidat list jednotlivých *UID* za klíčové slovo *EXPUNGE*.

```
tag UID STORE 1,2,3,4,5,6 +flags.silent \deleted
tag UID EXPUNGE
```

Kopírování e-mailů se liší dle verze IMAP serveru. V případě některých starších IMAP serverů může být nutné provést celou sekvenci příkazů *STORE*, *COPY* a *EXPUNGE*. [19]

```
tag UID COPY 15:30 destination_mailfolder
tag UID STORE 15:30 +FLAGS.SILENT (\Deleted)
tag UID EXPUNGE
```

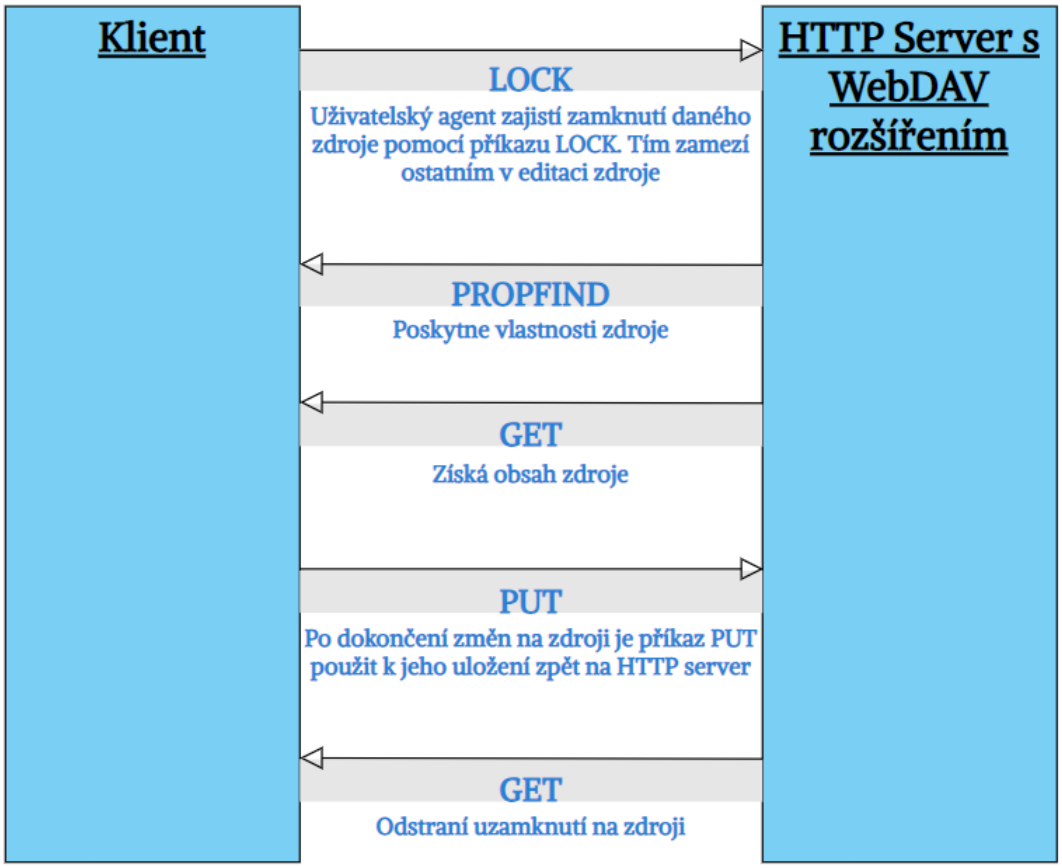
Zlomový bod ve vývoji protokolu IMAP nastal v roce 2013, kdy byl v rámci RFC 6851 představen příkaz *MOVE*, který obstará celý přesun autonomně. [20] Před použitím příkazu *MOVE* je nutné ověřit, že je příkaz podporován, k čemuž nám může pomoci vhodné zpracování odpovědi serveru na příkaz *CAPABILITY*.

```
tag MOVE 15:30 destination_mailfolder
```

1.2.3. WebDAV

WebDAV protokol je webový protokol (rozšíření protokolu HTTP) vzniklý v roce 1999 a aktuálně využívaný ve verzi RFC 4918 z roku 2007. [21] Protokol poskytuje možnost kooperace a správy souborů uložených na webovém serveru. Tato správa zahrnuje jednoduché operace jako mazání, přesuny a editování souborů, ale také komplexnější úkoly jako zamykání souborů, práci s metadaty apod. Tato práce se soubory je vykonávána pomocí specifických příkazů pro WebDAV jako je *LOCK*, *PROPFIND*, *GET*, *PUT*, *MOVE* apod. [22] Požadavky i odpovědi jsou ve formátu XML, kdy kromě příkazu jsou také specifikována příkazová data (tělo požadavku). Cílem této specifikace je získat od serveru pouze takové informace o zdroji, které pro klienta mají význam, a omezit tak zasílání nepotřebných dat.

Z pozice uživatele protokolu data uvnitř WebDAV serveru vypadají jako stromová adresářová struktura, což zjednodušuje práci s protokolem. V případě IceWarp je protokol využíván uvnitř desktopové aplikace pro editaci a správu kontaktů, kalendářů, poznámek a úkolů. Následující obrázek shrnuje typický proces komunikace mezi klientem a WebDAV serverem.



Obrázek 3: Schéma možné interakce uvnitř protokolu WebDAV

Před prací s WebDAV serverem je vhodné ověřit, jaké konkrétní příkazy z celkové sady příkazů podporuje. K tomu slouží příkaz `OPTIONS`. Odpovědí serveru na příkaz `OPTIONS` je seznam dovolených příkazů.

`PROPFIND` neboli „property find“ je příkaz, který slouží k získání vlastností specifikovaného zdroje. Součástí odpovědi na příkaz `PROPFIND` může být kromě jména také druh zdroje, datum vytvoření nebo modifikace. Příkaz `PROPFIND` tedy zajišťuje podrobné informace o specifikovaném adresáři a ostatních dostupných podadresářích. [23]

`GET` slouží pro získání čitelného obsahu entity, nad kterou je vyvolán. Jako příklad si lze představit požadavek na získání obsahu jedné specifické poznámky, jejíž ID je již známo. Odpovědí serveru je v tomto případě událost ve formátu vNote (RFC 5545). Stejným způsobem jsou v aplikaci reprezentovány kalendářové události. Příkazová data jsou v případě `GET` specifikována pouze v URL, a proto je jeho využití vhodné pouze pro jednoduché operace, jako je získání obsahu celého souboru. Není vhodné s příkazem `GET` posílat jakékoliv informace v těle požadavku. [24] Pokud by v záměru klienta bylo také obsah nějak upravit a poté odeslat tuto změnu na server, tak k tomu slouží příkaz `PUT`.

```
GET
https://icewarp.onice.io/webdav/martin@icewarp.onice.io/Notes
/6509dd80b78

BEGIN:VNOTE
VERSION:1.1
PRODID:-//IceWarp//IceWarp Server 14.1.0.0 RHEL7 x64//EN
SUMMARY:New: This is a test note created by JMeter
UID:6509dd80b785
X-SERVER-UID:6509dd80b785
CLASS:PUBLIC
CREATED:20230919T174224Z
LAST-MODIFIED:20230919T174224Z
DTSTAMP:20230919T174224Z
END:VNOTE
```

Úryvek kódu 1: Příklad požadavku GET a následná odpověď serveru

Komplexnější příkazy pro správu zdrojů jsou příkazy *REPORT* a *SEARCH*, které jsou blíže specifikovány v RFC 3253 a RFC 5323. Jejich funkcionalita by teoreticky mohla být nahrazena příkazem *POST*, ale použití *REPORT* a *SEARCH* je idempotentní a bezpečné, což v případě *POST* není zaručeno. [24] Využity jsou v situaci, kdy funkcionalita *GET* není dostatečná – je nutné specifikovat tělo požadavku.

SEARCH hledá uvnitř kolekcí na serveru dle specifikovaných parametrů v těle. Umožňuje tak vyhledávat na základě specifických vlastností a především obsahu. Odpovědí serveru je seznam zdrojů, které splňují kritéria. Tělo požadavku bývá oproti *REPORT* velmi komplexní. [25]

Naproti tomu *REPORT* je méně podrobný a umožňuje klientům požadovat specifické informace, jako jsou vlastnosti zdrojů, informace o serveru nebo vlastní hlášení definované serverem. Nezaměřuje se tedy tak na obsah samotných zdrojů jako na filtrování podle metadat. [26]

```
SEARCH /path/to/collection HTTP/1.1
Host: example.com
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8"?>
<D:searchrequest xmlns:D="DAV:">
    <D:basicsearch>
        <D:select>
            <D:prop>
                <D:getetag />
                <D:displayname />
                <D:resourcetype />
                <D:sync-token />
                <supported-calendar-component-set
xmlns="urn:ietf:params:xml:ns:caldav" />
                <supported-address-data
xmlns="urn:ietf:params:xml:ns:carddav" />
                <xmpp-uri
xmlns="http://calendarserver.org/ns/" />
```

```
                <calendar-color
xmlns="http://apple.com/ns/ical/" />
                <D:supported-report-set />
                <D:current-user-privilege-set />
            </D:prop>
        </D:select>
        <D:from>
            <D:scope>
                <D:href>/path/to/collection</D:href>
                <D:depth>infinity</D:depth>
            </D:scope>
        </D:from>
        <D:where>
            <D:is-collection />
        </D:where>
    </D:basicsearch>
</D:searchrequest>
```

Úryvek kódu 2: SEARCH požadavek

```
REPORT /path/to/resource HTTP/1.1
Host: example.com
Content-Type: application/xml

<?xml version="1.0" encoding="utf-8" ?>
<D:sync-collection xmlns:D="DAV:">
  <D:sync-token />
  <D:prop>
    <D:getcontenttype />
    <D:getcontentlength />
    <D:getlastmodified />
  </D:prop>
</D:sync-collection>
```

Úryvek kódu 3: REPORT požadavek

1.2.4. Exchange ActiveSync (EAS)

Exchange ActiveSync neboli EAS je protokol vytvořený společností Microsoft. Protokol je založený na protokolu HTTP a formátu XML. EAS umožňuje mobilním telefonům přístup k datům organizací na serveru, na kterém běží Microsoft Exchange. Data organizace jsou v tomto kontextu e-maily, kalendáře, kontakty a úkoly. EAS podporuje také SSL, jakožto zabezpečenou formu připojení. Ze serveru, se kterým je mobilní telefon pomocí protokolu spojen, je možné také částečně ovládat telefon. Je proto nutné, aby zabezpečení ze strany serveru bylo maximalizováno, jelikož se zde nabízí možnost i plně resetovat připojené mobilní zařízení. [27]

Během komunikace je využíváno hodnot SyncKey, které je nutné dodržovat, jelikož bez dodržování pravidel SyncKey nebude komunikace považována za validní. Jedná se o kladné, zvyšující se hodnoty, které slouží k synchronizaci a validaci komunikace. [28] Komunikace je zahájena synchronizací složek pomocí příkazu

FolderSync, který je zasílán HTTP metodou *POST*. V případě FolderSync by vždy hodnota SyncKey měla být rovna 0. Hodnota 0 značí serveru, že se jedná o začátek komunikace. Odpovědí na tento požadavek je kompletní adresářová struktura daného uživatele na serveru. Z tohoto výstupu poté je možné dohledat ID jednotlivých mailových, kalendářových a dalších složek.

```
POST https://icewarp.onice.io/Microsoft-Server-
ActiveSync?User=martin@icewarp.onice.io&DeviceId=EASHealthChe
ck&DeviceType=IceWarpTest&Cmd=FolderSync

<FolderSync xmlns="FolderHierarchy:">
<SyncKey>0</SyncKey>
</FolderSync>
```

Úryvek kódu 4: FolderSync požadavek a následná odpověď serveru

Ve chvíli, kdy jsou dostupné veškeré informace o adresářové struktuře uživatele, tak je možné nad jednotlivými složkami vykonávat operace vytváření, mazání a editování. Operace s EAS bývají velmi komplexní, a to jak z pozice klienta, tak z pozice serveru. Důsledkem toho je, že protokol EAS je velmi pomalý v porovnání s jednoduššími protokoly rozebranými v předchozích kapitolách. Příkladem komplexnosti požadavků protokolu je následující požadavek na tvorbu kontaktu. V případě vytváření/mazání poznámek či událostí je požadavek obdobný.

```
POST https://icewarp.onice.io/Microsoft-Server-
ActiveSync?User=martin@icewarp.onice.io&DeviceId=EASHealthChe
ck&DeviceType=IceWarpTest&Cmd=Sync

POST data:
<Sync xmlns="AirSync:">
   <Collections>
     <Collection>
       <SyncKey>0</SyncKey>

<CollectionId>4136a932eb7724a00cb87c3fb9e1ea1d</CollectionId>
       <GetChanges/>
       <WindowSize>25</WindowSize>
       <Options>
         <BodyPreference xmlns="AirSyncBase:">
           <Type>1</Type>
           <TruncationSize>32768</TruncationSize>
         </BodyPreference>
       </Options>
       <Commands>
         <Add>
           <ClientId>134</ClientId>
           <ApplicationData>
             <Body xmlns="AirSyncBase:">
               <Type>1</Type>
               <Data/>
             </Body>
             <Email1Address
xmlns="Contacts:">martin.zvelebil@icewarp.com</Email1Address>
```

```
            <FileAs xmlns="Contacts:">Martin,
ZVELEBIL</FileAs>
            <FirstName xmlns="Contacts:">Martin</FirstName>
            <LastName xmlns="Contacts:">Zvelebil</LastName>
            <MobileTelephoneNumber
xmlns="Contacts:">123456789</MobileTelephoneNumber>
            <Picture xmlns="Contacts:"/>
          </ApplicationData>
        </Add>
      </Commands>
    </Collection>
  </Collections>
 </Sync>
```

Úryvek kódu 5: Požadavek na tvorbu kontaktu

Protokol sám o sobě z velké části umí také nahradit fungování protokolů IMAP a SMTP, kdy může zajistit jak zaslání e-mailu, tak také jeho následné vyzvednutí ze serveru, přesuny a mazání. Veškeré tyto operace i s příklady jsou zdokumentovány na webu Microsoftu. [29] V rámci této práce bude k jednotlivým požadavkům nahlíženo jako na celek bez nutnosti rozboru jednotlivých elementů operací.

2 Průzkum/Analýza

2.1 Stávající řešení používané ve firmě

V praxi je otázka aproximace náročnosti aplikace výzvou pro řadu firem a v případě jejího špatného řešení může vést až k selhání projektů za miliony korun například z důvodu nedodržení slíbeného up-time. Pokud aplikace nemá jedno jediné využití, ale možnosti využití aplikace jsou téměř neomezené, obtížnost tohoto úkolu se ještě zvyšuje. V takovou chvíli je nutné pro vyřešení této otázky začít uvažovat i aspekty, které nemusí být na první pohled vůbec důležité. Bude aplikace nasazena v zemi, kde jsou nějaké tradice? Jaké je v zemi náboženství? Jaký je pracovní trend? Jakým způsobem bývala v předchozích projektech v dané zemi aplikace používána? Veškeré tyto faktory mohou zásadně ovlivnit hardwarové požadavky na danou aplikaci. Cílem řešení je tyto faktory zahrnout jako celek a vytvořit model, který by měl být, co nejvíce univerzální a nabízet variabilitu dle specifikace využití daného serveru.

Aktuální řešení ve firmě vynechává nejen tyto faktory, ale všeobecně je založeno především na ponaučení z předchozích projektů, které nebyly úspěšně zakončeny nebo naopak z projektů úspěšných. Proces odhadu začíná ve chvíli, kdy přijde požadavek na určení hardwarových požadavků pro zákazníka. Tento požadavek je přijat obchodním oddělením a následně ihned předán na technické týmy. Už v tuto chvíli je jasné, že proces není optimalizovaný, jelikož by v ideálním případě tento požadavek mělo být schopné vyřídit obchodní oddělení pouhým zadáním počtu uživatelů do tabulky a výstupem by byl seznam (nejen) hardwarových požadavků na aplikaci a aplikace podpůrné.

Požadavek je dále zpracován technickým oddělením infrastruktury. Ze strany vývoje aplikace neexistují jakékoliv podklady jako jsou testy výkonnosti aplikace nebo propustnosti jednotlivých částí aplikace. Oddělení infrastruktury tedy ručně odhaduje hardwarové nároky nového projektu na základě předchozích projektů, bez ohledu na specifická využití aplikace ze strany uživatelů. V případě podpůrných serverů bývá určení hardwarové náročnosti relativně jednoduché, jelikož se používají například aplikace třetích stran, které mají specifikováno kolik požadavků jsou schopny na určeném hardwaru zpracovat nebo jsou to velmi nenáročné komponenty. V případě hostitelských serverů aplikace IceWarp již není situace tak jednoduchá. Poté, co jsou data z předchozích projektů porovnána, se tým seniorních zaměstnanců na poradě shodne na určitém odhadu funkčního nastavení. Se sepsanými požadavky na CPU, RAM, velikost úložiště apod. předává tým infrastruktury požadavek zpět na obchodní oddělení, které s výsledkem konfrontuje zákazníka.

I přes veškeré zkušenosti týmu, který požadavky odhadl, tak mohou nastat odchylky, které nebyly zohledněny. V takové situaci je nutné škálovat celý projekt podle aktuálních výsledků, což může být záležitostí až několika měsíců. V některých případech takové přeškálování již není možné a chybný odhad hardwarové náročnosti může ve výsledku vést k celkovému ukončení projektu.

Tato práce si dává za cíl optimalizovat výše popsaný proces bez nutnosti využití seniorních zaměstnanců, jejichž čas je nejdražší. Zaměření je spíše na umožnění dedikovat odhad náročnosti na juniorní členy infrastrukturního týmu, kteří budou schopni jednoduše dle reálných dat rozhodnout, jaké požadavky na hardware budou kladeny. V ideálním případě vytvořit specifikaci tak přesnou, že vše bude schopné vyřídit obchodní oddělení s pomocí jednoduché tabulky, a tým seniorních kolegů poté pomocí výstupů z implementace této práce ověří, zda se server chová dle očekávání. S tímto cílem se také pojí zajištění těchto reálných dat, která v tuto chvíli neexistují v jakékoliv podobě. Tato data bude částečně možné získat z logů jednotlivých částí aplikace, či monitorovacích systémů. Z větší části bude nutné založit úplně nové monitorovací a vyhodnocovací systémy, a především programy, které budou simulovat chování uživatelů, aby mohl být vytvořen profil průměrného uživatele aplikace, který bude mít jednoznačně řečeno, kolik požadavků, na jakou část aplikace za den práce zhruba vyvolá.

Za vedlejší výstup lze považovat také doporučení pro již existující servery, kdy bude možné nasimulovat pro kolik dalších uživatelů bude aplikace ještě považována za stabilní na daném hardware. Tím vzniká univerzální, znovupoužitelný nástroj na testování jakékoliv instance aplikace IceWarp. Díky těmto simulacím také bude možné stanovit například limity velikostí odesílaných e-mailů, které nyní neexistují a nastávají poté problémy na úrovni protokolů, které příliš velké e-maily neumí zpracovat. Při nezpracovaných e-mailech nastávají nekonečné smyčky, které mohou vést k absolutnímu vytížení stroje. V aktuální chvíli jsou tyto smyčky řešeny ručně s pomocí techniků a monitorovacích systémů a zároveň předimenzováním hardwaru, na kterém je aplikace nasazena, aby i v případě nějaké smyčky aplikace byla ve stabilním stavu.

2.2 Nástroje na tvorbu syntetických dat

Pro simulaci uživatelské aktivity je potřeba dat, na kterých bude simulace provedena. Uživatelé, kteří aplikaci používají také mají svá vlastní data, se kterými provádí jednotlivé operace. Data těchto uživatelů ale nesmí být během testování, jakkoliv narušena a z toho důvodu je nutností tvorba dat syntetických a pokud možno takovým způsobem, aby byla strukturálně co nejblíže datům reálným.

Nejčastější úkon v aplikaci IceWarp je práce s e-maily. Ať už se jedná o práci uvnitř webového, desktopového nebo mobilního klienta, vždy je základem e-mailová komponenta. E-mailových dat je nejvíce nejen co se týče počtu, ale také co se týče velikosti. Zároveň se jedná o data nejvíce citlivá. Z toho důvodu je nutné zajistit tvorbu syntetických e-mailových schránek.

Tvorba jiných syntetických dat by v určitých případech také mohla být užitečná, ale v rámci úspory času bude tvorba syntetických událostí, poznámek apod. nahrazena funkcionalitou přímo uvnitř simulujících programů, které si data, se kterými budou pracovat, vytvoří a pak také smažou. V případech, kdy je potřeba nějaká data připravit předem, budou vytvořena shodnou strukturou na všech uživatelích, kdy bude zajištěno jejich automatické importování. Tento scénář sice neodpovídá úplně přesně reálnému využití, ale tvorba těchto syntetických dat

vyžaduje úzkou spolupráci s databází a jejich tvorba a nasazení by vyžadovala mnoho dalších prostředků, které aktuálně nejsou dostupné. Z toho důvodu budou vytvořeny pouze syntetické e-mailové schránky.

Se vzrůstající popularitou umělé inteligence začíná být tvorba syntetických e-mailových schránek jednodušší, ale žádné z řešení plně nesplňuje požadavky této práce. Přesněji řečeno, žádné z existujících řešení nezaručuje, že data, která generuje, budou založena na reálných statistikách uživatelů. Žádná z dostupných aplikací nenabízí ani možnost vygenerování falešných e-mailů o specifikované velikosti, nýbrž pouze generátory zátěže (Postman – Bind Email_QA4) [30] případně aplikace zajišťující jednodušší tvorbu e-mailů pro efektivní marketing (Mailchimp, Stripo). Další skupinou dostupných aplikací zajišťující generování syntetických dat jsou aplikace, které zajišťují syntetická data s ohledem na ochranu uživatelů, ale nejsou schopné vygenerovat přímo data využitelná do e-mailů/e-maily samotné. Taková skupina aplikací je vhodná například pro testování datové analýzy, algoritmů nebo strojového učení, jelikož výstupem bývá tabulka dat (databáze dat), se kterými lze dále pracovat. Příkladem této skupiny je mostly.ai. [31]

Z tohoto důvodu bude nutná vlastní implementace programu, který zajistí jak analýzu distribuce velikostí jednotlivých e-mailů uživatelů v jejich schránkách, tak také tvorbu jednotlivých e-mailů a jejich uskupení do schránek na bázi této analýzy. Sekundární výhodou vlastního řešení bude plná kontrola nad obsahem dat a jejich snadná modifikace v případě změny trendů v chování uživatelů.

2.3 Nástroje umožňující simulování chování uživatelů

2.3.1. IceWarp WebClient

Aplikace IceWarp, jak již bylo zmíněno v teoretické části, téměř nikdy v historii nebyla podrobena jakýmkoliv testům, které by si kladly za cíl určit výkonnost a náročnost aplikace. Ačkoliv existuje jedna výjimka, a to IceWarp Annihilator.

Tento nástroj nebyl zaměřen na simulaci chování uživatelů, nýbrž byl koncipován spíše jako test zátěžový. Formou ramp-up testu vytěžoval specifikovaný server a vracel relevantní výsledky. Tento nástroj byl vyhodnocen ve spolupráci s týmem vývojářů jako nevhodný, a to hned z několika důvodů:

- **Stáří** – Nástroj byl vyvinut pro verzi aplikace IceWarp z roku 2013. Od té doby se aplikace transformovala na mnohem komplexnější a proběhl téměř kompletní refactoring.
- **Použité technologie** – Nástroj je napsán v jazyce Pascal, který nesplňuje současné standardy této problematiky.
- **Dostupnost zdrojového kódu** – Zdrojový kód není téměř dostupný, jelikož byl projekt nevhodně verzován a je součástí celého technického základu aplikace z roku 2013 a nelze oddělit.

- **Nejedná se o simulaci** – Nástroj není schopný specifikovat počet uživatelů, nýbrž funguje spíše jako performance test zaměřující se na počet požadavků, které server zvládá odbavit.

Z důvodu nevhodnosti jediného dostupného nástroje bude nutné v případě webového klienta vytvořit nástroj, který simulaci vhodným způsobem zajistí přes IceWarp API. Bude nutné zvolit jazyk, který je jednoduše spustitelný na Linuxových serverech a podporuje komplexní zpracování textových řetězců (odpovědí API). Program, který bude mít tuto simulaci na starost, bude vykonávat jednotlivá volání na API a tím simulovat klikání na jednotlivé komponenty a funkcionality uvnitř komponent. Vzhledem k rozmanitosti možností využívání webového klienta by tento program měl podporovat možnost sestavení vlastních scénářů chování uživatelů a tím pokrýt, pokud možno, co největší množství uživatelských skupin.

2.3.2. IMAP

Pro IMAP již byly shrnuty nejzákladnější operace protokolu v teoretické části. I přes rozsáhlý průzkum nebyl autor schopný nalézt nástroj, který by automatizovaně a náhodně byl schopný s IMAP serverem komunikovat, a tyto operace tedy vykonával. Je tomu tak pravděpodobně z důvodu, že využití IMAP serveru se liší napříč implementacemi jednotlivých klientů a samotná simulace chování není již pouze záležitostí odeslání e-mailu nýbrž sekvencí operací, které mají nutně logickou návaznost.

Nutností tedy bude tvorba vlastního nástroje, který bude schopen operace uvnitř e-mailových složek specifikovaných uživatelů vykonávat, a to i s ohledem na jejich data. Důležité pro řešení tedy bude, aby původní data uživatelů nebyla narušena a frekvence jednotlivých vytipovaných operací z teoretické části odpovídala reálnému vytížení generované jedním uživatelem na jednom e-mailovém klientu. Neméně důležité bude, aby pravděpodobnost jednotlivých náhodných operací byla nastavitelná a operace byly vykonávány jak na jednotlivých e-mailech tak na skupinkách e-mailů. Při přechodu mezi složkami je očekáváno zajištění standardního synchronizačního procesu pomocí `HIGHESTMODSEQ` a `FETCH` příkazů.

2.3.3. SMTP

V případě SMTP je důležitá frekvence, s jakou se e-maily odesílají a jaké jsou jejich velikosti – obdobně jako u tvorby syntetických dat. Díky jednoduchosti a rozšířenosti protokolu SMTP existují veřejně dostupné statistiky, které říkají, že průměrný kancelářský zaměstnanec odešle za den zhruba 40 e-mailů. [32] Tyto statistiky, stejně jako v případě ostatních simulačních modulů (kde tyto statistiky dostupné nejsou), budou ověřeny pomocí porovnání dat ze simulovaných a reálných serverů. Co se týče velikosti těchto jednotlivých e-mailů, tak v tomto případě bude možné plně navázat na tvorbu syntetických schránek, kde bude využito analýzy velikostí e-mailů uvnitř existujících e-mailových schránek. Tím bude zajištěna vhodná distribuce velikostí i při odesílání e-mailů. Jediné nalezené existující řešení,

které je schopné automatizovaně testovat SMTP server a splňuje většinovou část požadavků se nazývá SMTP Bomberman.

- **SMTP Bomberman** [33]
  - Open-source nástroj pro automatizované odesílání e-mailů o specifikované velikosti na cílový server napsaný v jazyce Go. Nástroj umožňuje nastavit velikost odesílaných e-mailů a jejich počet.
  - Jeho zásadní nevýhodou je absence možnosti nastavit frekvenci odesílání e-mailů, a tudíž se jedná spíše o test výkonnosti, ale nejedná se o vhodné řešení na simulaci uživatelské aktivity.
  - Jeho další nevýhodou je nereálná tvorba e-mailů, jelikož odesílané e-maily neobsahují hlavičky vyžadované standardem RFC 822 a zároveň není možné tyto e-maily modifikovat, tudíž neobsahují ani přílohy.
  - V neposlední řadě byl vyhodnocen jako nevhodný z důvodů skromné dokumentace k projektu a nulové údržbě projektu.

Jak již bylo zmíněno, z kompletních řešení se jedná o jediné, které splňovalo požadavky. I přes to, vzhledem ke zmíněným nevýhodám, bylo zamítnuto autorem práce. Konečným řešením tedy i v tomto případě bude vlastní implementace. Jestliže díky tvorbě syntetických dat bude zajištěna složka plná validních e-mailů, tak stačí zajistit vhodného MTA, který předem připravené e-maily odešle v požadovaných rozestupech.

- **Mutt** – e-mailový klient [34]
  - Komplexní e-mailový klient podporující IMAP a POP, příjem a editaci e-mailů uvnitř příkazové řádky apod.
  - Složitější zpracování v příkazové řádce z důvodu grafického rozhraní.
  - Interaktivní, grafický
- **Exim (sendmail)** – MTA Linuxové příkazové řádky [35]
  - Jednoduchý nástroj příkazové řádky zajišťující odesílání e-mailu mezi SMTP servery, které jsou k přenosu nutné.
  - Nemá žádné uživatelské rozhraní.

Vzhledem k tomu, že pro odesílání předem připravených e-mailů je potřeba pouze jednoduché funkcionality MTA agenta, tak jako nejvhodnější nástroj vychází sendmail, jakožto nejméně komplexní nástroj umožňující jednoduché zpracování v příkazové řádce. Exim je novější rozšíření nástroje sendmail, které převzalo e-mailový trh nástroji sendmail. V roce 1996 byl sendmail využíván na 80 % veřejných e-mailových serverů, kdežto v dnešní době to jsou pouze 4 %. [35] Exim nabízí vyšší bezpečnost, komplexnější konfiguraci e-mailů (jako například tvorbu e-mailu uvnitř skriptu) nebo užití proměnných. Jak již bylo zmíněno, tyto výhody nejsou pro tuto práci zásadní, jelikož e-maily budou předem připravené, a proto bude využito staršího a jednoduššího MTA – sendmail.

2.3.4. WebDAV

Díky tomu že je WebDAV součástí rodiny webových protokolů, tak jej lze testovat podobným způsobem jako jakékoliv webové aplikace. Implementace protokolu WebDAV bývá různá napříč aplikacemi, které ho využívají, a tudíž nelze očekávat existenci programu, který by protokol uvnitř aplikace IceWarp vhodně otestoval. Díky tomu, že se jedná o webový protokol, existuje mnoho aplikací, které jeho testování mohou značně ulehčit.

K testování výkonnosti webových aplikací jsou známé automatizační programy jako Selenium nebo Puppeteer, ale ty jsou vhodné spíše pro test funkcionalit nežli zajištění syntetické zátěže. Ve srovnání s nimi jsou programy Apache JMeter nebo Gatling pro syntetickou zátěž optimalizovanější. Oba tyto nástroje jsou napsané v jazyku Java a o Gatlingu se mluví jako o modernější verzí Apache JMeter. Oba tyto nástroje podporují rozhraní pro tvorbu HTTP požadavků, jejich zpracování a editaci. Kromě těchto nástrojů stojí za zmínku také Tsung, Apachebench a Locust. Tyto nástroje pro účel simulace protokolu WebDAV nejsou vhodné ať už z důvodů zdlouhavého zápisu ve formátu XML s absencí doprovodného GUI, nedostatečné výkonnosti nebo nedostatečnému počtu funkcionalit.

S ohledem na autorovy předchozí zkušenosti byl pro účely této práce zvolen program Apache JMeter. Určitou nevýhodou oproti Gatlingu může být fakt, že veškeré funkcionality jsou „modulovány“ a dedikovány na specifický modul, který má svou vlastní syntaxi. V případě Gatlingu je vše řízeno v jazyku Scala, který může být vhodnější v případě vyšších nároků na dynamičnost scénářů. Oba tyto programy potřebují ke svému běhu JVM. Výhodou Apache JMeter je také velká uživatelská základna a velké množství integrací (modulů). Diskutovanou nevýhodou by mohl být fakt, že výsledkem z GUI je soubor ve formátu XML, který je těžko čitelný a bez GUI téměř není možné scénáře upravit. [36]

Tím, že WebDAV je využíván uvnitř IceWarp desktopového klienta, který má oproti webovému klientu omezenější počet funkcionalit a frekvence jeho využívání je v praxi mnohokrát menší, tak není očekávána od testových scénářů taková modularita. Cílem nástroje simulace WebDAV bude tedy vhodně zahrnout nejzásadnější funkcionality protokolu do jednoho opakujícího se scénáře. Důležité, stejně jako v případě všech ostatních simulačních modulů, bude zajistit vhodnou frekvenci odesílaných operací. Z měření z minulých interních projektů firmy zároveň vychází najevo, že jednotlivé různé operace WebDAV protokolu jsou oproti jiným protokolům více záměnné, jelikož jejich náročnost na server bývá ve všech případech podobná. To ještě více přispívá k tomu, že jeden opakující se scénář bude dostatečně dobře reprezentovat uživatele operujícího na protokolu WebDAV.

2.3.5. Exchange ActiveSync (EAS)

Situace je v případě EAS obdobná jako u protokolu WebDAV. Jedná se o webový protokol a k jeho testování, stejně jako v případě WebDAV, bude využito Apache JMeter, jak bylo odůvodněno v předchozí části (2.3.4).

Co se týče specifikace zadání implementace, cílem nástroje bude zahrnout veškeré základní funkcionality popsané v teoretické části do jednoho scénáře, který bude možné periodicky spouštět. Z měření v rámci předchozího interního projektu firmy vyšlo najevo, že protokol EAS je oproti WebDAVu až 10x pomalejší, a i z toho důvodu je v rámci vize firmy nahradit ho vlastní mobilní aplikací. Z toho důvodu se jedná o nástroj, na jehož implementaci je kladen nejmenší nárok a bude považován za dostatečný pouze jeden scénář bez modularity sekvence operací.

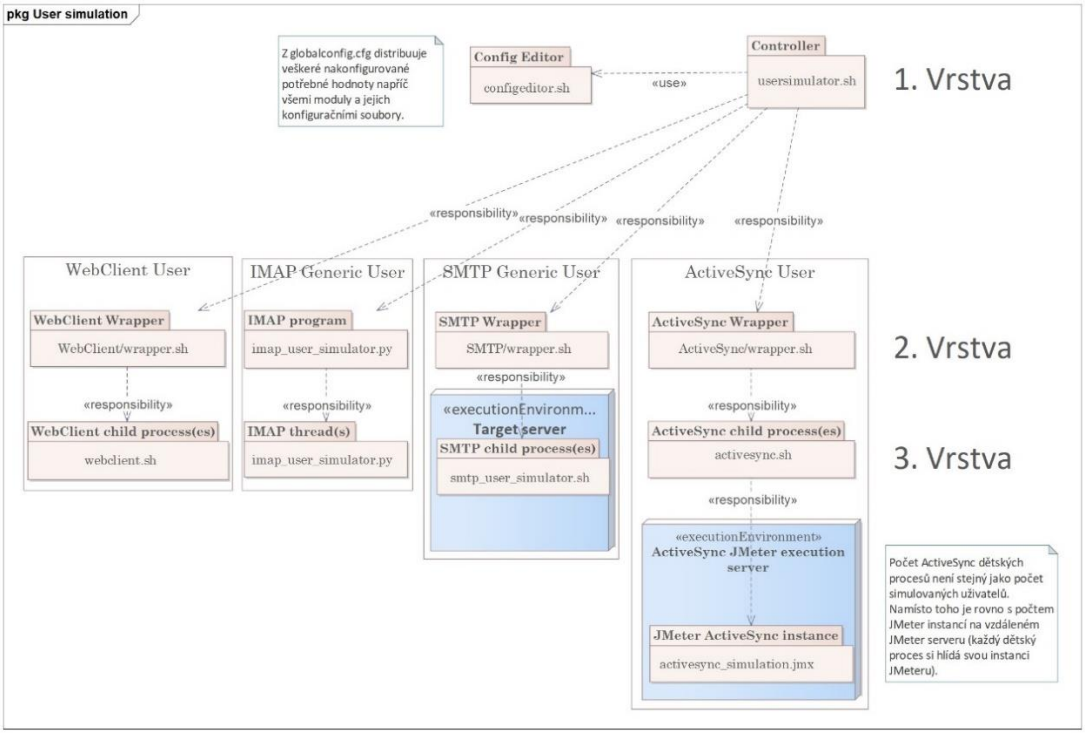
3 Návrh a implementace nástrojů

Celý systém simulace uživatelských aktivit je postaven na simulaci pěti částí (modulů): IceWarp WebClient, SMTP, IMAP, WebDav a EAS. Každý z těchto modulů byl implementován vlastním způsobem ve specifickém jazyku s vlastním konfiguračním souborem. Díky rozdílnosti jazyků jednotlivých implementací bylo nutné zajistit rozhraní, které bude schopné zajistit kontrolu nad všemi moduly. Z toho důvodu je základem celého projektu tříúrovňová architektura.

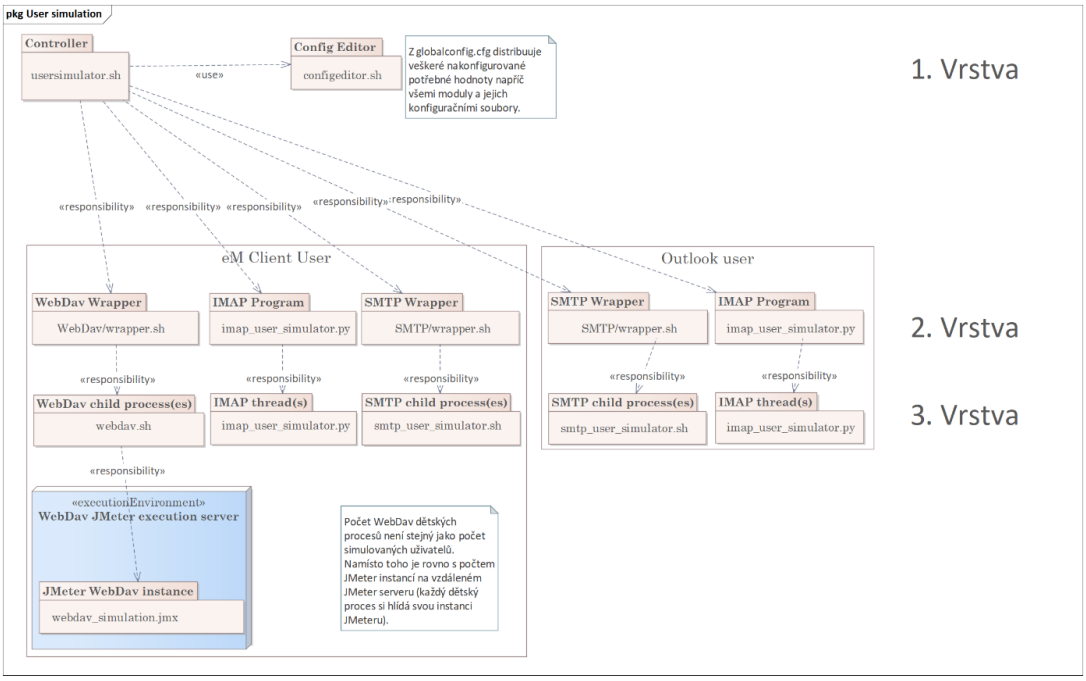
Na nejnižší vrstvě se nachází jednotlivé procesní moduly. V případě Bash skriptů se jedná o jednotlivé procesy, v případě modulů realizovaných v programovacím jazyku Python se jedná o jednotlivá vlákna. Nad nimi jsou ve druhé vrstvě jejich správci – wrappery. Tento wrapper je zpravidla napsán ve skriptovacím jazyku Bash a má na starost kompletní správu procesních modulů. V případě Python modulů tuto funkci zastává program, který vytváří jednotlivá uživatelská vlákna. Na nejvyšší úrovni je jednoduchý Bash skript, který přijímá na vstupu kolik uživatelů je třeba simulovat a zajišťuje předání této informace druhé úrovni, která dle toho spustí potřebný počet procesních vláken. Tento skript nejvyšší úrovně je zároveň zodpovědný za globální editování všech konfiguračních souborů napříč moduly. Před spuštěním je tedy nutné tento globální konfigurační soubor nastavit ve složce `Controller/configedit`. Simulovaní uživatelé jsou rozlišováni následovně:

1. Uživatel IceWarp WebClient (WebClient modul)
2. Generický SMTP uživatel (SMTP modul)
3. Generický IMAP uživatel (IMAP modul)
4. Uživatel mobilního zařízení (EAS modul)
5. Uživatel desktopového klienta (SMTP + IMAP + WebDav modul)
6. Uživatel Outlooku (SMTP + IMAP modul)

Ještě před samotnou implementací jednotlivých modulů je nutné shrnout jakým způsobem funguje tvorba syntetických schránek. Výstup z kapitoly tvorby syntetických schránek je poté využit některými ze simulačních modulů.



Obrázek 4: Schéma projektu (1. část)



Obrázek 5: Schéma projektu (2. část)

3.1 Syntetické e-mailové schránky

V případě syntetických e-mailových schránek je vhodné zmínit adresářovou strukturu dané části projektu. Je tomu tak z důvodu, že tato část není nijak přímo provázána s ostatními moduly simulace a jedná se tedy o separátní program, jehož důležitost ovšem nesmí být opomíjena. Tato struktura si klade za cíl pomoci s orientací v textu.

Program *event_generator.py* je jediný z programů následující adresářové struktury, který nebude blíže popisován, jelikož se k němu neváže žádná analýza a jedná se o prostý generátor událostí do kalendáře. Jak již bylo zmíněno v teoretické části, syntetická data jsou důležitá i v případě jiných komponent aplikace IceWarp, nežli e-mailů, ovšem není dbáno takovým způsobem na jejich statistickou správnost.

```
.
├── calendars
├── Configs
│   ├── from_addresses.txt
│   ├── subjects.txt
│   └── to_addresses.txt
├── Created_Mailboxes
│   └── u<num>
├── Frequence_Data
│   └── u<num>_data.txt
├── Template mailbox folder structure
│   └── <multiple_empty_subfolders>
├── Template_Files
├── Template_Mails_Automatically_Generated
│   └── <id>.imap
├── Template_Mails_By_Hand
│   └── <id>.imap
├── convert_data.sh
├── create_mailbox.sh
├── email_generator.py
├── event_generator.py
├── freq_analysis.sh
└── shuffle_mailbox.py
```

Adresářová struktura syntetických e-mailových schránek

Produkční servery společnosti IceWarp běží na operačním systému Linux (konkrétně CentOS) a pro analytickou část kapitoly syntetických schránek je nutné komplexních textových operací na produkčních serverech. K tomu je vhodný skriptovací jazyk Bash. Oproti alternativímu Pythonu v tomto případě nabízí jednodušší zpracování textových vstupů a bližší kontakt s OS serveru, který poskytne informace o velikosti jednotlivých e-mailů ve schránkách uživatelů.

Ve chvíli, kdy bude existovat statistika distribuce velikostí e-mailů, tak bude využito kombinace Bashe a Pythonu, pro zpracování a optimalizaci těchto dat, a jejich aplikování na tvorbu syntetických schránek. Zde je důležitá součinnost

skriptovacího jazyka Bash a programovacího jazyka Python. Python poskytuje jednoduché rozhraní pro tvorbu syntetických e-mailů o specifikované velikosti díky knihovně email. Oproti tomu Bash je jednoduchý na používání a umožňuje lepší práci s textem. Zároveň je jednoduše spustitelný na každém Linuxovém zařízení.

Pro pochopení toho, jak lze k takové analýze a tvorbě schránek přistupovat, bylo využito publikace „Statistical Methods for Generating Synthetic Email Data Sets“. [37] Ačkoliv samotná implementace není přímo obrazem tohoto vědeckého dokumentu, autorovi pomohla ke komplexnějšímu chápání problematiky.

3.1.1. Frekvenční analýza schránek reálných uživatelů

Jak již bylo zmíněno v úvodu kapitoly o syntetických schránkách, pro frekvenční analýzu bylo využito skriptovacího jazyka Bash, konkrétně se jedná o soubor *freq_analysis.sh*. Pro tuto analýzu bylo zvoleno 10 vzorových serverů z různých částí světa a různých odvětví činnosti, ale v případě přísnějších požadavků ze strany zadavatele na větší statistický vzorek je možné kdykoliv analýzu opakovat a získat tak ještě přesnější výsledky. Celkový uživatelský vzorek činil 12 011 emailových uživatelů společnosti IceWarp. Jednotlivý uživatelé byli rozděleni do 7 skupin dle celkové velikosti jejich e-mailových schránek:

Skupina	Velikost schránek	Zastoupení skupiny v celkovém počtu vzorků
U1	0–40 MB	5226 (43,5 %)
U2	41–220 MB	1417 (11,8 %)
U3	221–650 MB	1053 (8,8 %)
U4	651–2560 MB	1586 (13,2 %)
U5	2561–10240 MB	1286 (10,7 %)
U6	10241–35840 MB	1193 (9,9 %)
U7	35841+ MB	250 (2,1 %)

Tabulka 1: Rozdělení uživatelů dle velikosti schránek

Volba tohoto rozdělení vznikla odhadem na základě zkušeností autora s produktem, kdy cílem bylo vytvořit rovnoměrné rozdělení dat. Po získání počtu uživatelů reprezentující jednotlivé skupiny bylo zjištěno, že se dané skupiny přibližují rovnoměrnému rozdělení s výjimkou U1 a odhad byl tedy správný. Důvodem, proč je počet zástupců U1 ve srovnání s ostatními skupinami vysoký je fakt, že mnoho firem vytvoří e-mailový účet veškerým svým zaměstnancům, ale mnoho z nich ho ve výsledku téměř nevyužívá. Pro účely této frekvenční analýzy nebyly do měření zahrnuty zcela prázdné účty s velikostí schránky 0, protože by došlo ke zkreslení výsledků.

Analýza byla provedena jednoduchým programem, který pro každého uživatele specifikovaného na vstupu spustil příkaz *find* na veškeré soubory s koncovkou .imap v jeho e-mailovém adresáři. Pro každý z těchto e-mailů program zajistil pomocí příkazu *du -bs* jeho velikost v bytech a ta byla vydělena proměnnou *CHUNK_SIZE=1024*, čímž se e-maily shlukovaly do skupin po 1 KiB. Tudíž například

e-mail o velikosti 10300 B a 11000 B byl v rámci měření brán jakožto e-mail o shodné velikosti.

Skupina	Počet analyzovaných e-mailů	Celková velikost e-mailů skupiny
U1	1 362 833	42,87 GB
U2	1 586 119	200,27 GB
U3	1 852 784	502,12 GB
U4	4 798 191	1 881,17 GB
U5	10 183 249	5 580,99 GB
U6	18 723 713	23 760,6 GB
U7	9 312 264	5 248,54 GB
ALL	47 819 153	37 216,56 GB

Tabulka 2: Celkové množství analyzovaných dat dle uživatelských skupin

Výsledkem tohoto měření byl tedy pro každou skupinu soubor, který obsahoval na každém řádku velikost jednoho e-mailu ze vzorku v KiB. Tudíž například výstupem skupiny 4 byl soubor s 4 798 191 řádkami.

3.1.2. Konverze dat

Pracovat s desítkami milionů řádek dat a na jejich základě tvořit syntetická data není praktické. Proto si autor položil několik otázek ohledně vhodné granularity: Není v rámci syntetické tvorby dat téměř ekvivalentní, jestli velikost e-mailu je 105 MB nebo 107 MB? A je ekvivalentní také e-mail, který má 100 MB a 200 MB? Jak zvolit vhodnou granularitu ekvivalence velikostí? Nalézt odpovědi na tyto otázky si autor klade za úkol v této kapitole. Popsané řešení je implementováno v souboru `convert_data.sh`.

Pro ukázku konverze dat bylo využito uživatelské skupiny U5, ale stejný princip byl aplikován na všechny uživatelské skupiny. Z výstupu analýzy skupiny U5 vychází, že celkem uživatelé měli ve schránkách 31 747 různých skupin velikostí e-mailů. Mnoho z nich má ovšem procentuální zastoupení v miliontinách procent, jelikož se jedná o pouhé jednotky zástupců z celkových více než 10 milionů. Takové skupiny s podobnými velikostmi je tedy vhodné sloučit. Slučování dat může mít negativní vliv na jejich vlastnosti a je tedy nutné zvolit vhodnou metodu, která tyto negativní vlastnosti slučování minimalizuje. Tento problém bývá v anglické literatuře nazýván jako „data binning“ nebo „data discretization“. Do porovnání po výzkumu autora vstupují základní tři modely shlukování – exponenciální, lineární a $2^{\lfloor n/\$iterations \rfloor}$.

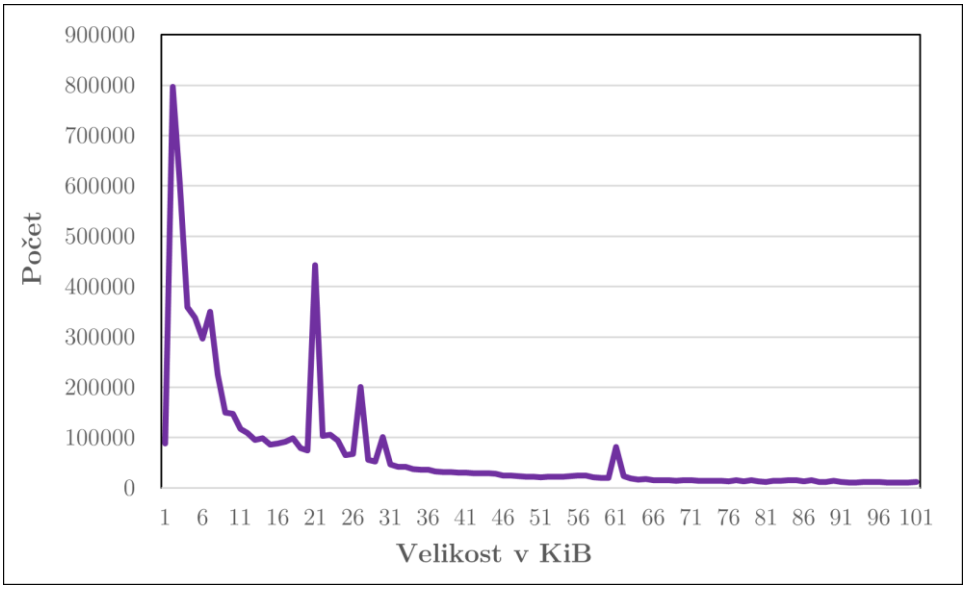
V implementaci lze nalézt čtyři proměnné, které tyto modely formují. Ze začátku je určen počet datových skupin, které je nutné nechat bez konverze, tudíž je nastavena hodnota `start_index` (od kolika KiB začít se slučováním velikostních skupin e-mailů). Proměnná `chunk` říká, kolik skupin je nyní slučováno a `adder` kolik bude přidáno v další iteraci do proměnné `chunk`. Čtvrtou proměnná `iterations` specifikuje po kolika iteracích dojde k dalšímu přidání `adder` do `chunk`. V každé

iteraci je také modifikován samotný `adder`, dle zvoleného modelu. Následující tabulka shrnuje hodnoty proměnných a funkce modifikující `adder` v jednotlivých modelech.

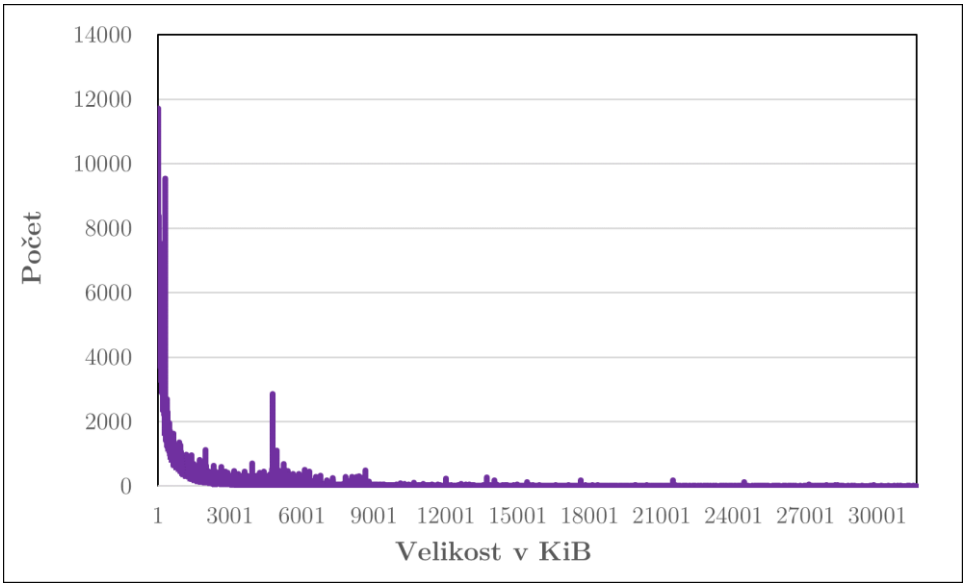
$2^{[n/\$iterations]}$	Lineární	Exponenciální
`start_index` = 30 `iterations` = 10 `chunk` = 2 `adder` = 2 `adder` = `adder` * 2	`start_index` = 30 `iterations` = 5 `chunk` = 2 `adder` = 1 `adder` = `adder` + 2	`start_index` = 30 `iterations` = 5 `chunk` = 2 `adder` = 1 `adder` = `adder` + 2

Tabulka 3: Porovnání hodnot proměnných v případě různých modelů aproximace

V následujících dvou grafech je shrnut výchozí stav distribuce pro skupinu U5. Zároveň si lze povšimnout, že četnost skupiny zahrnující 61 KiB je razantně vyšší oproti okolním skupinám. Jedná se o totiž o velikost uvítacího e-mailu aplikace IceWarp, který mnoho uživatelů nikdy nesmazalo a jeho četnost je vyšší především u nižších skupin U1-U3.

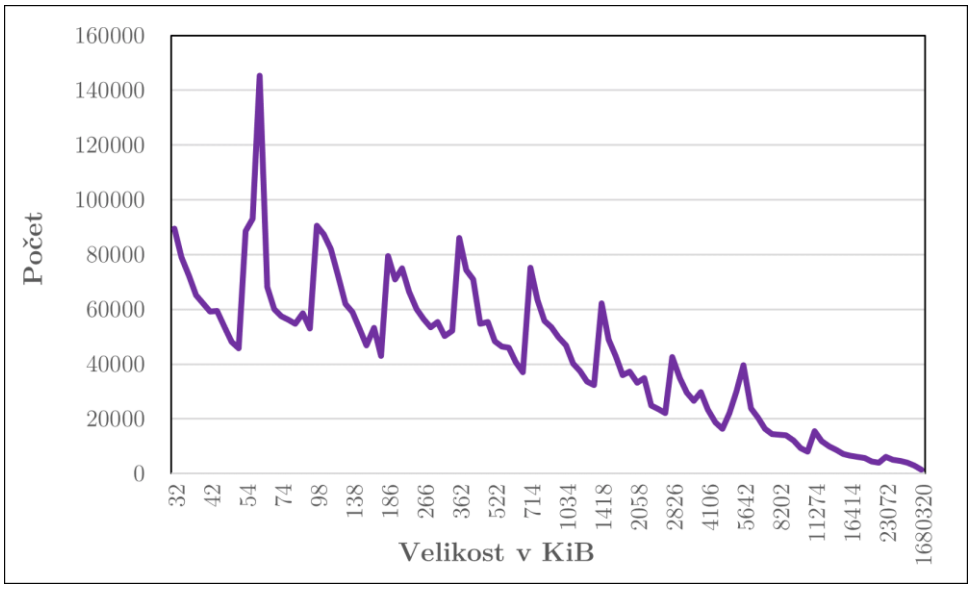


Graf 1: U5 výchozí stav (Frekvence e-mailů pod 100 KiB)

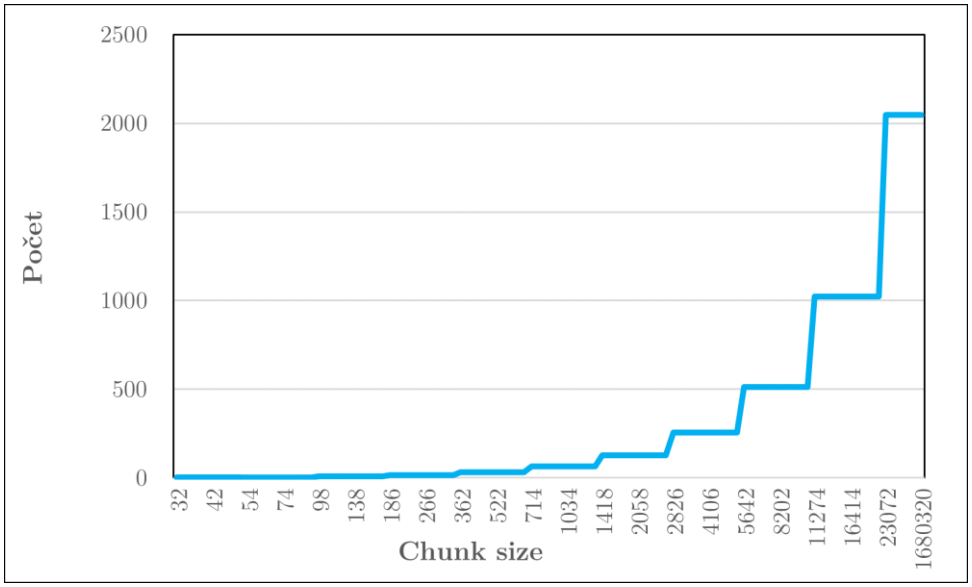


Graf 2: U5 výchozí stav (Frekvence e-mailů nad 100 KiB)

V následujících grafech je shrnuto, jakým způsobem jednotlivé modely modifikovaly data. Tato modifikace byla aplikována až od `start_index=30`, tudíž e-maily pod 30 KiB zůstaly zachovány s výchozí distribucí.

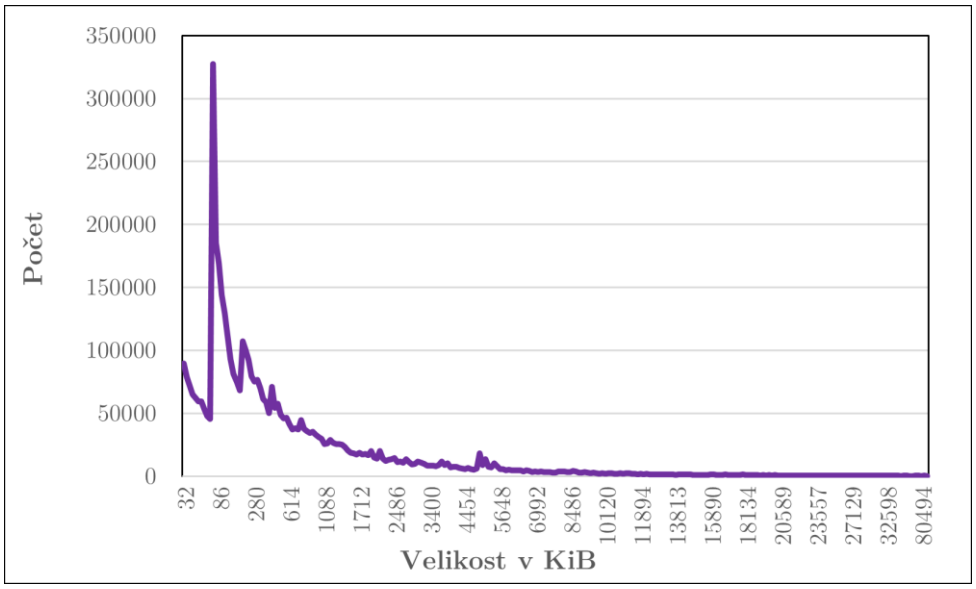


Graf 3: Výsledná data pomocí aproximace $2^{[n/\$iterations]}$

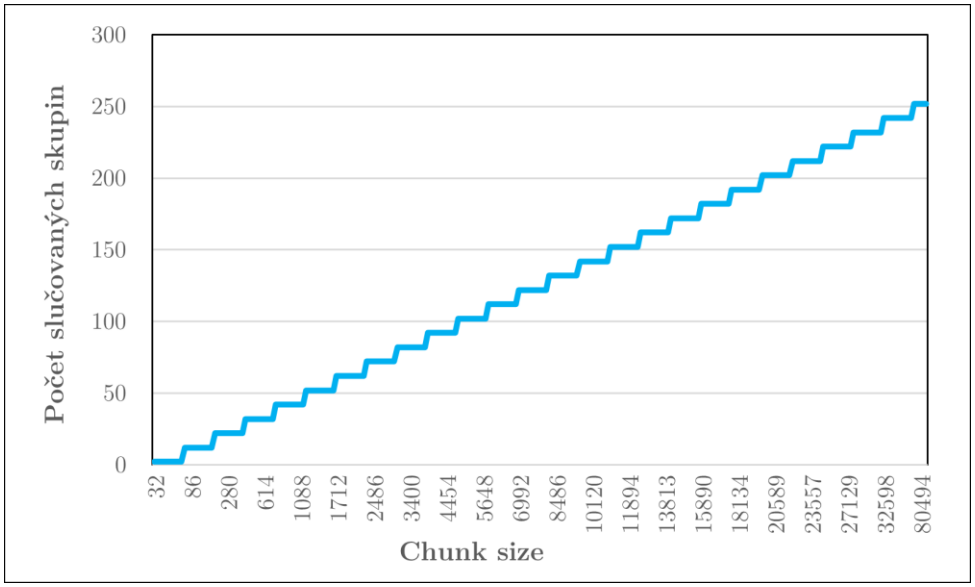


Graf 4: Vývoj proměnné chunk_size v případě $2^{\lfloor n/\$iterations \rfloor}$ aproximace

Metoda $2^{\lfloor n/\$iterations \rfloor}$ zajišťuje nejlepší zploštění dat, ovšem za cenu velkých skoků v datech, které nejsou žádané. Takové skoky v datech značí, že v momentě navýšení počtu sčítaných skupin (`chunk_size`) bylo navýšení příliš vysoké.

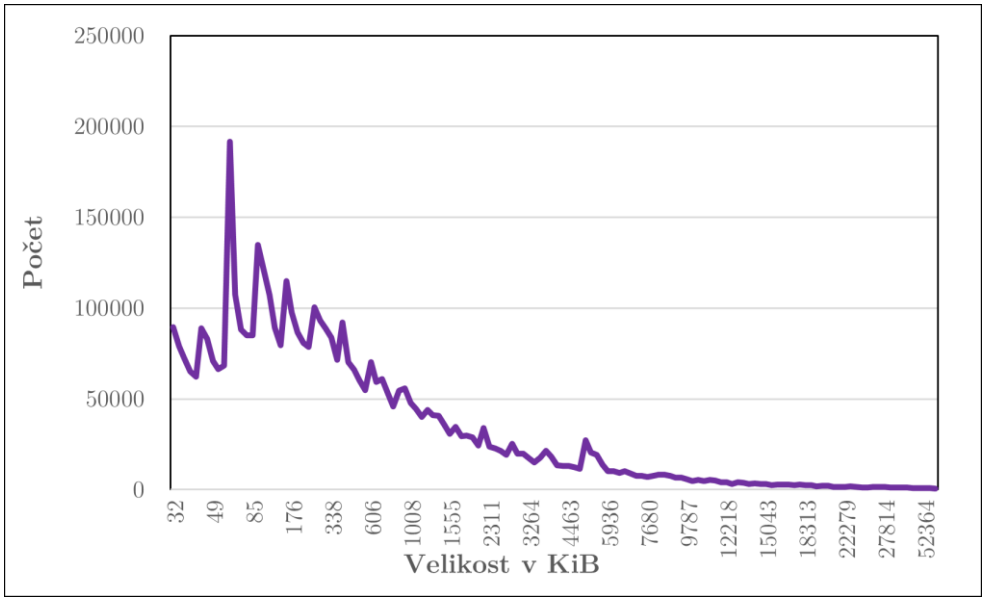


Graf 5: Výsledná data pomocí lineární aproximace

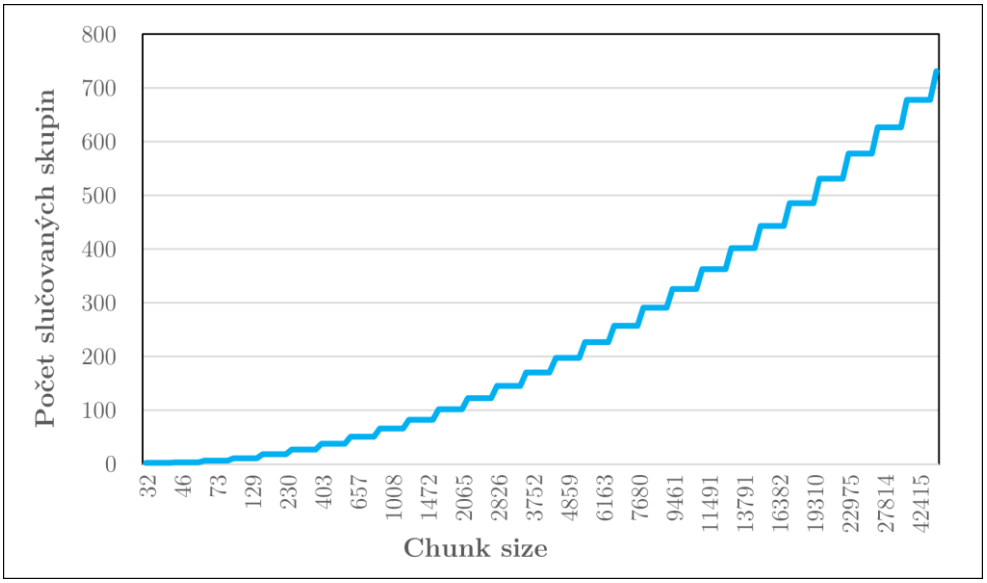


Graf 6: Vývoj proměnné chunk_size v případě lineární aproximace

Metoda lineární naopak nemá téměř žádné skoky v datech, ale naproti tomu data téměř nezplošťuje oproti výchozímu stavu, a tudíž by v případě velkých e-mailů jen těžko nastalo jejich vygenerování, jelikož jejich pravděpodobnost by stále byla velmi malá. Tudíž pro budoucí využití v implementační části je tato metoda také nevhodná.



Graf 7: Výsledná data pomocí exponenciální aproximace



Graf 8: Vývoj proměnné chunk_size v případě exponenciální aproximace

V případě metody exponenciální se jedná o kompromis, kdy skoky v datech nejsou tak znatelné jako v případě metody $2^{\lfloor n/\$iterations \rfloor}$, ale metoda zároveň zajišťuje velmi podobně kvalitní zploštění dat. Toto zploštění je požadováno za žádoucí, aby i v případě e-mailů nejvyšších velikostí nastalo jejich generování. Jako nejlepší se tedy jeví metoda exponenciální, která je obdobně aplikována na všechny uživatelské skupiny. Výstupem pro U5 je tedy namísto původních 31 747 velikostních skupin pouhých 163, kdy největší skupiny zahrnují až 678 skupin z nemodifikovaných dat v jedné skupině. Každá ze 163 skupin je označena maximální velikostí svého zástupce v KiB a počtem e-mailů v dané skupině.

Pro ostatní uživatelské skupiny byl aplikován stejný způsob pomocí exponenciální metody. Výsledky pro všechny uživatelské skupiny jsou dostupné ve složce `Frequence_Data`.

3.1.3. Tvorba syntetických schránek

Po provedení výše popsaných úkonů jsou zajištěna data o distribuci velikostí jednotlivých e-mailů v uživatelských schránkách a jsou optimalizována pro jejich reálné využití. Jediné, co schází pro tvorbu syntetických schránek jsou e-maily z daných velikostních skupin. V ideálním případě by bylo vhodné mít e-maily pro každou ze skupin z reálného provozu – takové, co obsahují všechny specifické hlavičky různých e-mailových klientů apod. To ovšem není možné, jelikož potenciálních velikostí, které by bylo třeba vygenerovat jsou desetitisíce. Z toho důvodu tímto způsobem byly pokryty alespoň skupiny nejčetnější 1-100 KiB. Tyto e-maily byly získány z firemní schránky autora nebo případně ručně vytvořeny odesíláním e-mailů z různých klientů s přílohami a tělem o specifikované velikosti. Pro skupiny, kde takové e-maily dostupné nebyly, tak bylo nutné zajistit jejich generátor. Z dostupných existujících nástrojů žádný nepodporuje tvorbu e-mailů s přílohami (MIME parts), tělem a hlavičkami dle RFC 822 a z toho důvodu bylo

nutné nástroj vytvořit. Konkrétně pro tvorbu syntetických e-mailů je vhodná knihovna email jazyka Python. Tato knihovna podporuje vše z výše zmíněného a díky tomu bude i synteticky vytvořený e-mail svou strukturou nerozeznatelný od toho z reálného provozu.

3.1.3.1. Tvorba syntetických e-mailů

Program *email_generator.py* vytvořený za účelem tvorby syntetických e-mailů přijímá jako vstup velikost generovaného e-mailu, jméno výstupního e-mailu, složku s přílohami, které může využít, a výstupní složku, kam se výsledný e-mail zapíše. V konfiguračních souborech je ještě nutné nastavit množinu předmětů, odesílatelů a příjemců. Ve chvíli, kdy program má veškeré tyto informace, vytvoří z nich hlavičky a zapíše výchozí tělo e-mailu, aby bylo zajištěno, že vygenerovaný e-mail mé vždy nějaké tělo. Poté program projde dostupné přílohy od největších po nejmenší a pokusí se vybrat nejmenší možné množství, aby zaplnil požadovanou velikost e-mailu. Tyto přílohy jsou tzv. „MIME parts“ – součást MIME rozšíření formátu e-mailových zpráv pro podporu jiných znaků mimo ASCII. Pro každou z těchto částí je specifikován jeho typ – „Content-Type“. V případě této implementace je podporováno pouze *image/jpeg*, *image/png* a zbytek je pro usnadnění identifikováno jako *application*. Jednotlivých možností Content-Type existuje mnohem více, ale pro tyto účely této práce není nutné využít všechny. [38] Po odečtení velikosti příloh, výchozího těla e-mailu a hlaviček je tělo e-mailu doplněno dostatečným počtem náhodných znaků, aby e-mail splňoval požadovanou velikost. Tyto 3 části (hlavičky, tělo a MIME parts) jsou poté spojeny za sebe pro zajištění korektnosti e-mailu dle RFC 822 a uloženy pod specifikovaným názvem do složky.

3.1.3.2. Skript na tvorbu schránek

Po vytvoření syntetických e-mailů je třeba veškeré předchozí části spojit dohromady a vytvořit program, který na základě naměřených a optimalizovaných dat určí počet a velikost jednotlivých e-mailů v syntetické schránce. K tomu bylo využito opět skriptovacího jazyka Bash ve skriptu *create_mailbox.sh*. Vstupem programu je statistický soubor ve formátu:

```
<velikost největšího možného emailu v KB - ID skupiny>
<chunk> <% záběr skupiny místem> <% záběr skupiny počtem>
```

Kromě statistického souboru je třeba specifikovat v proměnných na začátku skriptu velikost výsledné schránky pro danou uživatelskou skupinu. Tato velikost by měla být v rozmezí dané uživatelské skupiny, jak bylo nadefinováno v kapitole 3.1.1. Dále je také třeba definovat složku pro automaticky generované e-maily programem v jazyce Python, složku s ručně vygenerovanými e-maily a složku pro soubory výsledné e-mailové schránky.

Program začíná od velikostních skupin s největší velikostí a dle statistického souboru vypočítá, kolik by daná skupina měla zabírat místa. Pokud je výsledek tohoto výpočtu větší nežli minimální možná velikost e-mailu dané skupiny, tak

skript požadované kvantum e-mailů vytvoří a zbytek místa, které měla skupina zabírat, je přičteno do zásobníku. Jestliže je velikost menší, nežli minimální možná velikost e-mailu dané velikostí skupiny, tak velikost skupiny je pouze zapsána do zásobníku. Zásobník je v tomto kontextu pouze proměnná udržující informaci o tom, kolik místa nebylo předchozími skupinami zabráno, ačkoliv dle statistického výpočtu být mělo. V další iteraci se přechází na skupinu nižší akorát se k výpočtu přičítá také zásobník. Každá skupina tedy může pro své vygenerování ze zásobníku čerpat a případné přebytky do něj opět vrací.

Po poslední iteraci s velikostní skupinou 0-1 KiB, tedy skupinou nejmenší, je zahájena druhá fáze, jejímž cílem je využít případný zůstatek v zásobníku. Vstupem této druhé fáze je velikost zásobníku a procentuální pravděpodobnost výskytu e-mailů, které nebyly vygenerovány (tudíž celá paměťová alokace skupiny byla započtena do zásobníku). E-maily jsou vytvořeny dle pravděpodobnosti jejich výskytu, pokud je dostatek místa v zásobníku. Jestliže i přes všechnu optimalizační snahu zbývá v zásobníku místo, je vytvořen jeden e-mail o zbytkové velikosti zásobníku, čímž je zaručena přesná velikost schránky dle nastavených proměnných. Tato zbytková velikost zásobníku je ovšem v případě velkých uživatelských skupin U5-U7 netolerovatelně velká a přesahuje i velikost největší velikostní skupiny.

Řešením tohoto problému by mohla být jemnější aproximace vstupních dat, což by zajistilo více vzorků, z kterých se vybírá a také tedy více možností, kde se zásobník může aplikovat. Další možností, jak tento problém vyřešit je optimalizace volby velikosti vytvářeného e-mailu. Tato varianta je vhodnější, jelikož zajistí nejlepší statistické výsledky s jakýmkoliv vstupním souborem.

3.1.3.3. Optimalizace skriptu

V ideálním případě by skript měl vygenerovat takový počet e-mailů o nějaké velikosti spadající do dané skupiny, aby byla přesně pokryta vypočtená alokace skupiny. Před optimalizací skriptu byla vždy zvolena maximální možná velikost generovaného e-mailu pro danou skupinu, což ale v nějakých případech vedlo ke zbytečně velkému přičtení velikosti do zásobníku. Ovšem v jedné skupině může být více variant možných velikostí vygenerovaných e-mailů, které stále spadají do téže skupiny a nějaké jsou vhodnější než jiné. Tato optimalizace vede k volbě toho nejvhodnějšího řešení. Pro jednodušší pochopení následuje ukázka na příkladu.

Zadaný je výpočet alokace skupiny na 139 128 B a identifikátor skupiny je 37 810 B. Předchozí skupina je identifikována jako 34 120 B. Díky těmto informacím je možné zjistit, že e-maily spadající do této skupiny jsou všechny, které jsou v rozmezí (34 120 B, 37 810 B]. To znamená, že pokud cílem je vygenerovat co nejméně e-mailů pro danou skupinu, bude zvolena velikost 37 810 B, což je ekvivalentní 3.68 e-mailům. V takovém případě by zápis do zásobníku činil 25 698 B. Naproti tomu pro nejmenší možnou velikost 34 121 B to vychází na 4,08 e-mailů se zápisem 2 644 B do zásobníku. Existuje však i optimálnější řešení, které do zásobníku zapíše prakticky 0 B. V tomto případě by se jednalo o tvorbu 4 e-mailů o velikosti 34 782 B. Takové plně optimální řešení je dostupné pouze ve chvíli, kdy je následující vzorec pravdivý:

```
⌊X⌋ - ⌊Y⌋ > 0, kde
X = Počet e-mailů v případě horní mezi velikosti
Y = Počet e-mailů v případě dolní mezi velikosti
```

Na předchozím příkladě je možné pozorovat, že volba největší velikosti není vždy nejvhodnějším řešením. Pokud je zmíněný vzorec nepravdivý, tak je zvolena ta varianta, jejímž výsledkem bude nejmenší zápis do zásobníku aneb ta z variant, jejíž desetinná část počtu e-mailů je nejmenší. Jestliže je vzorec pravdivý a optimálních řešení je více, tedy výsledek výpočtu vzorce je větší než 1, je vybrán medián z optimálních řešení. Medián je vybrán z toho důvodu, že se jedná o řešení, které nejlépe průměruje rozptyl velikostí dané skupiny. Díky této optimalizaci, i v případě vygenerování největších mailových schránek U6 a U7 o velikostech v desítkách GB, není přebytek v zásobníku větší nežli největší generovaná skupina. Pokud je nějaký přebytek v zásobníku i po ukončení optimalizovaného skriptu, je vytvořen jeden e-mail o velikosti zásobníku, který doplní požadovanou velikost schránky.

3.1.3.4. Ověření statistické správnosti vygenerované schránky

Skript kromě samotného generování také vytváří statistický soubor o právě vygenerované schránce, jehož hodnoty jsou porovnatelné se vstupními statistickými hodnotami. Toto porovnání bylo provedeno pro všechny velikostní skupiny e-mailů uživatelské skupiny U3 až U7 pomocí výpočtu:

```
|X / Y|, kde
X = Očekávané zabrané místo e-mailovou skupinou
Y = Synteticky tvořenou skupinou skutečně zabrané místo
```

Tento vzorec byl aplikován na všechny velikostní skupiny. Na tuto množinu výsledných hodnot byl poté aplikován průměr a medián pro každou uživatelskou skupinu. Tento výpočet byl proveden jak pro očekávaný paměťový záběr dané velikostní skupiny, tak pro očekávaný počet e-mailů v dané skupině vůči všem vygenerovaným e-mailům. Tato statistika nebyla vypracována pro uživatelské skupiny U1 a U2, jelikož velikost generované schránky je příliš malá, a tudíž nebyl z mnoha velikostních skupin vygenerován žádný e-mail. Tvorba této statistiky by byla komplikovaná a výsledky by nebyly nijak zásadně přínosné pro účely této práce.

	Průměrná odchylka velikosti	Medián odchylky velikosti	Průměrná odchylka počtu	Medián odchylky počtu
U1	-	-	-	-
U2	-	-	-	-
U3	11,70 %	5,95 %	12,99 %	6,59 %
U4	11,32 %	0,90 %	13,60 %	3,11 %
U5	5,25 %	0,35 %	6,07 %	1,24 %
U6	2,98 %	0,51 %	2,97 %	2,48 %
U7	1,06 %	0,12 %	1,04 %	0,44 %

Tabulka 4: Ověření statistické správnosti syntetických schránek

Z tabulky 4 vyplývá, že čím větší je velikost vytvářené schránky, tím jsou odchylky menší. Zároveň je důležité zmínit, že se jedná o poměr očekávaného zabraného místa/počtu a kolik skutečně daná skupina ve výsledku v syntetické schránce zabrala místa/počtu. Pokud by výpočtem byl absolutní rozdíl mezi očekávaným procentuálním podílem dané skupiny na velikost/počet a výsledným, tak by se výsledné hodnoty pohybovaly v řádech desetitisícin procent. Celkově lze říci, že při tvorbě průměrného serveru s distribucí uživatelů viz. kapitola 3.1.1. nebude odchylka přesnosti schránek oproti realitě převyšovat jednotky procent.

3.1.3.5. Distribuce e-mailů do složek

Ve složce, kam byla schránka vytvořena, jsou veškeré e-maily nejprve pouze v jedné složce, což neodpovídá reálnému scénáři. Z toho důvodu je nutné do dané složky překopírovat připravenou adresářovou strukturu a nad kořenovou složkou obsahující .imap soubory spustit program *shuffle_mailbox.py*. Ten zajistí nelineární distribuci .imap souborů napříč složkami a jejich podsložkami. Tím je schránka hotova.

3.2 Moduly (3. vrstva)

3.2.1. IceWarp WebClient

V případě modulu na simulaci IceWarp webového klienta se jedná o komplexní skript ve skriptovacím jazyce Bash. Jednotlivá volání na API při jednotlivých činnostech uvnitř aplikace nejsou firmou zdokumentované a z toho důvodu bylo nutné si je ručně v rámci webového prohlížeče vždy odchytit, analyzovat a přepsat do vhodné formy uvnitř skriptu.

Konkrétně se jednalo o analýzu síťového provozu a odchytávání požadavků *webmail.php*, které jednotlivá volání obsahovaly. Tímto způsobem byly postupně zachyceny všechny nejdůležitější funkcionality webové aplikace. Celkem je implementováno bezmála 60 kritických funkcionalit webového klienta. Veškeré tyto funkcionality jsou dokumentované v rámci *README.md* tohoto modulu i s příklady užití. Některé z těchto funkcí již obsahují více atomických funkcí spojených v jednu.

Příkladem může být funkce *MailFolderFlow*, která simuluje procházení specifikované složky včetně scrollování a otevírání specifikovaného počtu e-mailů. Zároveň jsou k dispozici i více atomické operace jako je vymazání kontaktu, TeamChat místnosti apod. Neméně důležité je simulování přihlášení, kdy se vykonává mnoho inicializačních operací klienta a autentizace samotná.

Pro simulaci uživatelů bylo potřeba vytvořit scénáře. Scénářem v tomto kontextu je sekvence kritických funkcionalit s optimálními rozestupy mezi jednotlivými operacemi. Celkem jsou připravené 4 scénáře, kdy každý z nich má za úkol simulovat jinak vyspělého uživatele webového klienta. Od uživatelů, pro které se jedná o málo vyspělého e-mailového klienta, až po ty, kteří každou komponentu webového klienta využívají denně. Tvorba scénářů probíhala ve spolupráci s interním týmem firmy IceWarp, aby bylo zamezeno jednostrannému pohledu na užití webového klienta. Původní plán tvorby scénářů na základě dat o využívání jednotlivých funkcí webového klienta musel být nahrazen z důvodů nedostatku podkladů ze strany firmy IceWarp. Tyto scénáře jsou voleny náhodně během spuštění instance skriptu. Obdobně, jako v následující ukázce scénáře kalendářové komponenty je tomu v případě všech komponent:

```
MeasureComponentTime "start" "Calendar"
InitCalendarComponent
CreateAndDeleteEventProcedure
ChangeCalendarView "week"
CalendarFlow "4" "5"
CalendarFlow "-2" "3"
ChangeCalendarView "workweek"
CreateAndDeleteEventProcedure
CalendarFlow "0" "0"
ChangeCalendarView "month"
CalendarFlow "-3" "0"
CalendarFlow "5" "1"
CreateAndDeleteEventProcedure
SearchInEventsProcedure "meeting" "2"
SearchInEventsProcedure "a" "10"
CalendarClickTrash
EmptyCalendarTrash
ChangeCalendarView "day"
CreateAndDeleteEventProcedure
MeasureComponentTime "stop" "Calendar"
```

Úryvek kódu 6: Příklad scénáře pro kalendářovou komponentu WebClient modulu

3.2.2. SMTP

Pro simulování uživatelské aktivity ze strany protokolu SMTP bylo opět využito skriptovacího jazyka Bash. Tento program simuluje odesílání specifikovaného počtu e-mailů za průměrnou pracovní dobu (8 h). Jak bylo zmíněno v průzkumné části k protokolu SMTP, tak tento počet je prozatím nastaven na 40 a bude v následujících kapitolách zpřesněn podrobnou analýzou. K odesílání e-mailů je využíván nástroj sendmail. Pokud jeden uživatel odešle 40 e-mailů za 8 h, tak se jedná o jeden e-mail za 12 minut. Samotné odeslání e-mailu je ovšem záležitost

milisekund, maximálně jednotek sekund a díky tomu je možné jednou instancí skriptu simulovat více uživatelů zároveň. Nejprve je nutné ověřit, že skript takový počet uživatelů je schopen simulovat pomocí příkazu:

```
./smtp_user_simulator.sh --maxval <emaily_za_směnu>
```

Výstupem příkazu je maximální počet uživatelů, které jedna daná instance skriptu zvládne simulovat při daném počtu odeslaných e-mailů za směnu jednoho uživatele.

Před samotným spuštěním skriptu je potřeba ověřit, že e-maily, které budou odesílány z výchozí složky *$email_folder*, mají správné nastavení hlaviček From: a To:. Jestliže nemají, tak je lze nastavit v souboru *$config_file* a spustit příkaz:

```
./smtp_user_simulator.sh --configure
```

Tento příkaz zajistí nahrazení hlaviček From: a To: ve všech e-mailech uvnitř složky *$email_folder*. Jakmile je složka s e-maily připravena, tak lze skript spustit pomocí:

```
./smtp_user_simulator.sh --run <emaily_za_směnu>
<počet_simulovaných_uživatelů>
```

Skript po spuštění vypočítá frekvenci odesílání e-mailů dle specifikovaných parametrů a následně v nekonečné smyčce náhodně vybírá e-maily ze složky *$email_folder* které odešle pomocí nástroje *sendmail*. Pokud by operace odeslání trvala déle, než na ni je vyhrazeno dle předchozího výpočtu, tak je program schopný se adaptovat a v následujících iteracích tento fakt vykompenzovat, aby počet odeslaných e-mailů seděl s předpoklady. Veškeré informace o odesílaných e-mailech, jejich velikostech a případně odezvě či problémech s odesláním lze nalézt v souboru, který je uveden v proměnné *$log_file*.

3.2.3. IMAP

V případě protokolu IMAP by Bash jako jazyk pro řešení nebyl vhodný, vzhledem k tomu, že pro navázání relace v terminálu je nutné využití programu openSSL. Po připojení a autentizaci, která byla popsána v teoretické části je uživatel přítomný v interaktivním openSSL klientu. Což je pro sekvenční zpracování v terminálu nevhodné. Z toho důvodu bylo využito programovacího jazyka Python a knihovny *imaplib*, která je určena k práci s IMAP serverem. Díky volbě programovacího, a nikoliv skriptovacího jazyka, tak se otevírá nová možnost vícevláknové aplikace. Součástí implementace tedy je také podpora více vláken, kdy každé vlákno simuluje jednu aktivní relaci (jednoho aktivního uživatele). Vícevláknová aplikace umožňuje efektivnější tvorbu jednotlivých relací, nežli tomu je v případě skriptovacích jazyků. U skriptovacích jazyků je totiž nutné pro každou relaci vytvořit samostatný proces, nebo v rámci jednoho procesu vytvářet zátěž za více uživatelů, jako tomu je v případě SMTP modulu popsaného výše.

Program spustí požadovaný počet uživatelských vláken, která během svého běhu postupně restartuje, čímž se vlákna musí znovu přihlásit a celý inicializační proces zopakovat. Toto opatření je v programu za účelem přiblížení se k reálnému

scénáři, kdy se uživatelé odpojují a připojují, ačkoliv v aktuální chvíli jsou veškeré programy funkční pouze s konstantě zadaným počtem uživatelů. Každé z těchto vláken je z počátku neaktivní po dobu určenou proměnnou *UMP*, která specifikuje, kolik uživatelů by mělo navázat svou relaci každou minutou. Tím je zajištěno volnější rozložení přihlášení uživatelů, ale zároveň je jistota zajištění prvotního nárůstu přihlášených uživatelů (typické např. v ranních hodinách, kdy zaměstnanci přicházejí do svých zaměstnání, kde je IceWarp využíván).

Následuje navázání IMAP relace. Jelikož se počítá s možným spouštěním tohoto simulátoru na reálných uživatelích, tak si každé vlákno vytvoří své vlastní testovací prostředí ve výchozí složce doručené pošty, kterou musí nutně mít každý validní uživatel – „inbox“. Toto testovací prostředí je tvořeno kořenovou složkou, ve které je *$folder_stack* podsložek. Uvnitř těchto podsložek je celkem *$email_stack* e-mailů, které si program nakopíruje z doručené pošty. Pokud není dostatečný počet e-mailů uvnitř složky doručené pošty, tak prohledává i ostatní složky daného uživatele. Tím je získáno prostředí, které již nezávisí na datech uživatele, ačkoliv s jeho daty pracuje, ovšem bez rizika ztráty těchto dat. V mezičase tvorby těchto složek je také testováno jejich přejmenovávání a přechody mezi jednotlivými složkami.

V testovacím prostředí jsou prováděny náhodné operace přesunů (70 % pravděpodobnost volby operace), kopírování (25 %) a mazání (5 %) e-mailů. Pro každou z těchto operací je vybrán náhodný počet náhodných e-mailů z intervalu *[$min_email_cnt, $max_email_cnt]*. Rozestup mezi jednotlivými operaci je opět určen náhodným čekáním v intervalu *[$MIN_WAIT_TIME, $MAX_WAIT_TIME]*. Každá z IMAP operací je logována s dobou jejího trvání do souboru *$LOG_FILE*. Kromě toho v logovacím souboru je možné vyčíst pro každý záznam uživatele, ID vlákna, čas, typ operace apod. Tento log může sloužit pro budoucí analýzu, stejně jako v případě všech modulů.

Při výběru jakékoliv složky je nutné získat její aktuální podobu. K tomu slouží princip s hodnotami mod-sequence popsaný v teoretické části. Tento princip je také implementován, a tudíž při přechodu mezi složkami po nějaké operaci je vždy složka synchronizována. V současnosti není k programu připojena databáze, která by data o nejvyšší modseq hodnotě pro danou složku spravovala, ale je toto zajištěno za pomocí obyčejného souboru, který je chráněn před zapisováním jiných vláken v daný okamžik pomocí zámků. Tento soubor má na každé řádce trojici „uživatel:jméno_složky:hodnota“. Jestliže přejde program do složky, hledá dvojici „uživatel:jméno_složky“ v souboru, a jestliže je dvojice nalezena, hodnota je extrahována a použita. V případě, že dvojice nalezena není, tak je vrácena výchozí hodnota 1. Ta značí, že složka nikdy v minulosti nebyla na daném IMAP klientu navštívena. Aby v souboru nebyla nikdy zbytečná data, tak po tvorbě relace je vždy pro daného uživatele porovnán aktuální obsah souboru s jeho adresářovou strukturou. Pokud kterékoliv jméno složky související s jeho e-mailovou adresou v souboru není přítomno v aktuální adresářové struktuře, je záznam smazán.

Výsledkem je tedy komplexní nástroj, který plně simuluje základní uživatelské operace na protokolu IMAP. Každý e-mailový klient využívá IMAP

trochu jiným způsobem, ale výše popsaný způsob je častý v aplikacích firmy IceWarp, a proto byl vybrán jako ten nejvhodnější.

3.2.4. WebDav

V případě WebDav, jak již bylo popsáno v části analytické, bylo využito programu Apache JMeter. Jelikož se jedná o specifické prostředí, bylo pro účely této práce stěžejní najít správný způsob práce s programem. Tvorba operací již vycházela pouze z orientace v programu a využití různých modulů pro zpracování výstupů a vstupů.

Základem implementace v programu je takzvaná „thread group“. Tu si lze představit jako obyčejný program, který si spravuje svá jednotlivá vlákna. Podobně jako tomu je v případě IMAP modulu. Takových „thread group“ lze mít v jednom JMeter zdrojovém souboru více, v tomto případě je jen jedna. Na začátku je vhodné vytvořit si proměnné, které budou napříč jednotlivými operacemi využívané, obdobně jako když jsou konfigurovatelné proměnné na úvod zdrojového kódu v programovacích jazycích. Následuje vyčtení přihlašovacích údajů pro jednotlivé uživatele ze souboru a jejich přiřazení danému vláknu. Poté již stačí správně nastavit autorizaci a je možné postupně vytvářet HTTP požadavky. Sekvence a frekvence požadavků je založena na analýze logů z reálného provozu.

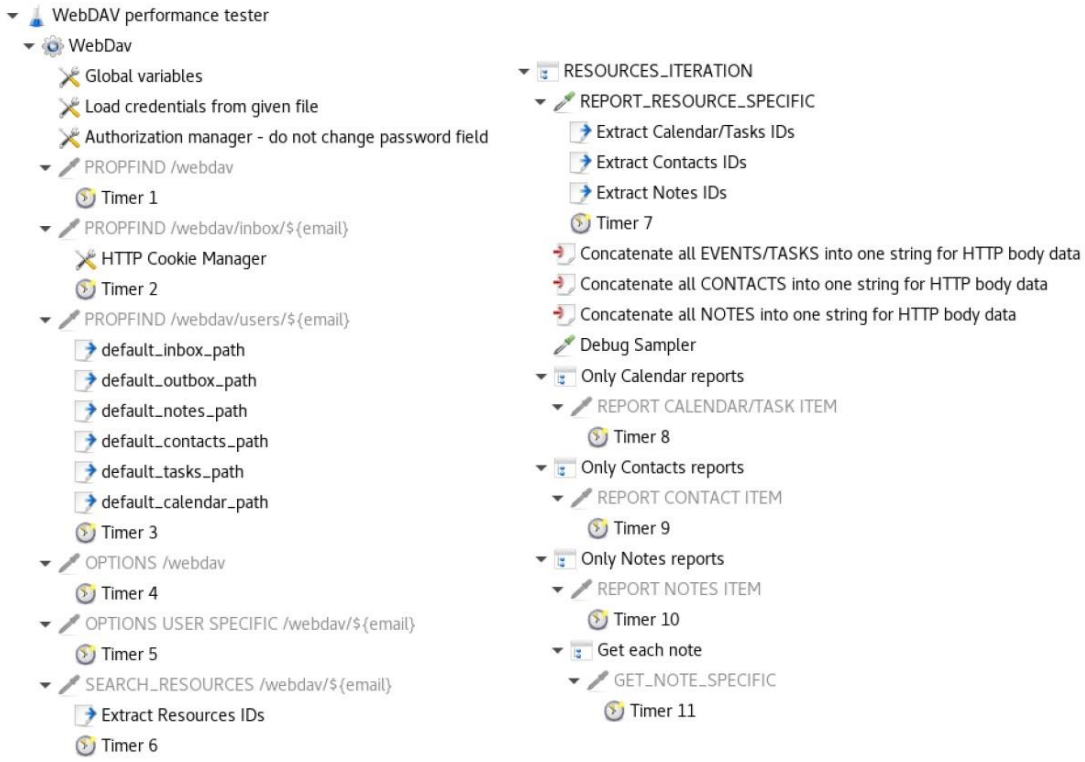
Na úvod programu jsou odeslané `PROPFIND` příkazy napříč adresářovou strukturou výchozí složky WebDav nastavené v globálních proměnných. Z odpovědí serveru jsou vyextrahovány cesty ke složkám kontaktů, kalendářů, e-mailů apod. Extrahování z odpovědí je vždy prováděno modulem XPath2 Extractor. Tato extrakce je prováděna pomocí vyjadřovacího jazyka XPath. Poté následuje `OPTIONS` příkaz, který zjišťuje jaké WebDav metody server podporuje. Vzhledem k tomu, že servery, na kterých bude test prováděn, mají všechny stejnou podporu WebDavu, tak odpověď serveru na tento požadavek dále není nijak zpracována. Je důležité zmínit, že pokud by odpověď měla být zpracována, tak na rozdíl od ostatních metod WebDavu, je zde odpověď serverem odeslána v hlavičkách odpovědi, a nikoliv v těle odpovědi.

Z příkazu `SEARCH RESOURCES` jsou extrahovány identifikátory jednotlivých zdrojů (`resource_id`), nad kterými v následující části programu budou vykonávané operace. Zdrojem v tomto kontextu jsou už konkrétní složky, které obsahují data zpracovávaná protokolem WebDav. O těchto zdrojích zatím není známa žádná informace, kromě jejich identifikátoru. K tomu, aby bylo možné identifikovat druh zdroje, slouží WebDav metoda `REPORT` odeslaná na URL obsahující `resource_id`. Tato metoda totiž specifikuje obsah zdroje, a poté pomocí vhodné XPath formule je možné vyextrahovat jednotlivé položky zdroje a určit, jakého jsou typu. Tím je tedy možné všechny položky roztřídit podle toho, jestli se jedná o poznámky, kontakty či kalendářové události/úkoly. Úkoly a kalendářové události jsou v rámci programu vnímány ekvivalentně, jelikož jejich formát je téměř stejný. Jediné, v čem se v implementaci liší mezi úkoly a kalendářovými událostmi, je prohozený čas začátku a konce v případě úkolů. Všechny položky dané složky, které spadají do nějakého typu, jsou poté pomocí BeanShell PreProcessoru spojeny

v jeden string. Tento string je následně vstupem do WebDav metody *REPORT*, která zajistí od serveru obsah všech specifikovaných položek uvnitř stringu. Tím jsou v jednom požadavku zajištěny veškeré informace o všech položkách daného typu ve specifikované složce. Tento *REPORT* je prováděn zvlášť pro každý druh zdroje. V případě poznámek je navíc nutné pomocí základní metody *GET* získat individuální poznámky.

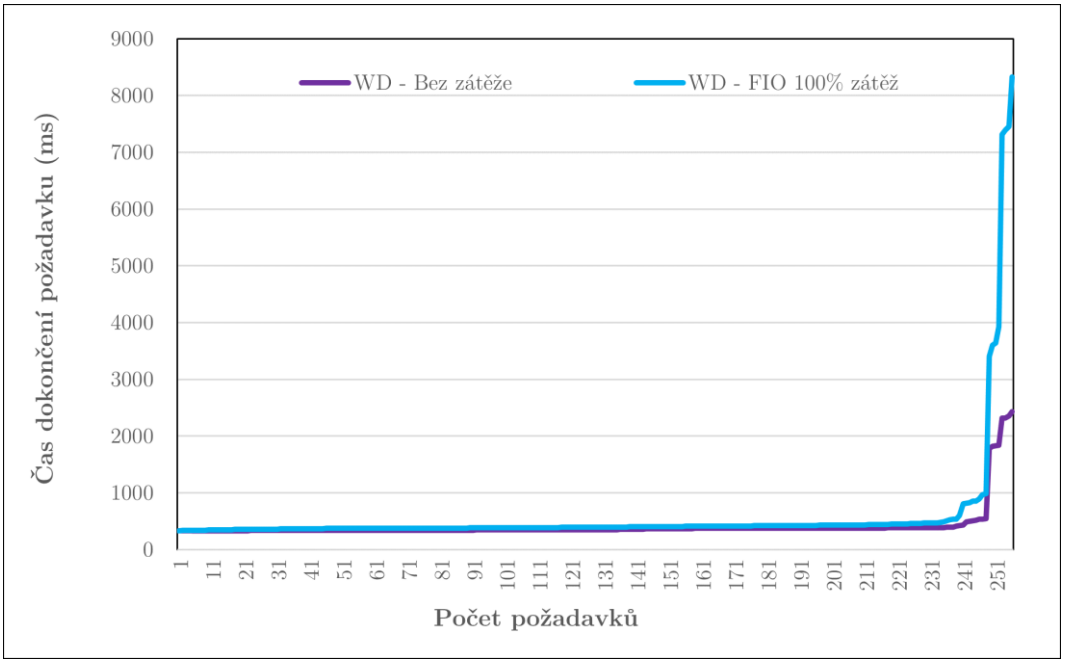
V případě programů vytvořených v Apache JMeter není součástí žádná náhodnost scénářů. Scénář je tvořen sekvenčně, jak již bylo popsáno v analytické části. Výsledkem je tedy sekvence operací, která odpovídá jednomu specifickému využívání protokolu WebDav. V případě, že by se sekvenčnost scénáře v čase ukázala jako nevhodná, tak by bylo nutné zajistit více různých scénářů, simulujících různě vyspělé uživatelské požadavky, stejně jako je tomu v případě WebClient modulu.

Vše výše zmíněné je dobře editovatelné v grafickém rozhraní Apache JMeter, ovšem toto rozhraní není vhodné a samotnými vývojáři není doporučované na zátěžové/simulační testy. Z toho důvodu bylo nutné zajistit program, který bude klíčové části zdrojového kódu ve formátu XML pro Apache JMeter konfigurovat. K tomu slouží soubor *webdav.sh*, který umožňuje tvorbu zdrojového souboru pro Apache JMeter dle konfiguračního souboru ve složce *configs*. Podle nastavených hodnot přepíše kód uvnitř XML souboru a vygeneruje tak nový vstupní zdrojový kód pro JMeter. Tento proces je dostupný pod přepínačem *--prepare*. Pokud tento zdrojový kód ve formátu XML je již připravený, tak je možné pomocí *--run* odeslat tento soubor na JMeter server a nastartovat běh. Tento program zároveň dohlíží na korektní ukončení programu na externím JMeter serveru a následné smazání zbytečných dat po jejich stažení na lokální server.



Obrázek 6: Struktura modulu WebDav v programu JMeter (obsahující optimalizační časovače)

Součástí zadání od zadavatele bylo také zjistit, jak moc má na protokol WebDav vliv aktuální vytížení serveru. K vygenerování zátěže na pozadí bylo využito Linuxového nástroje Flexible IO Tester (FIO). Výsledkem byl v případě většiny požadavků nárůst doby vyřízení o 9-15 %. V případě náročnějších požadavků jako je například `PROPFIND /webdav` se jednalo ovšem o 224 % nárůst z průměrných 2354 ms na průměrných 7617 ms. Následující graf toto srovnání zobrazuje. Jedná se o všechny odeslané požadavky z výše popsaného scénáře seřazené dle doby vyhotovení.



Graf 9: Porovnání protokolu WebDav bez zátěže/se zátěží (Požadavky seřazeny vzestupně dle doby vyhotovení)

3.2.5. Exchange ActiveSync (EAS)

Stejně jako v případě WebDav, tak i zde byl vybrán jako vhodný program Apache JMeter. O jeho specifikách viz. předchozí kapitola.

V případě ActiveSync jsou na úvod pomocí FolderSync získány informace o složkách a pomocí XPath2 Extractoru vyextrahovány identifikátory kalendáře, kontaktů, poznámek, rozepsaných e-mailů, příchozí pošty, koše a odeslané pošty. Ve chvíli, kdy jsou dostupné ID jednotlivých složek, je možné uvnitř nich vytvářet, mazat a manipulovat s daty. Program tedy ve třech separátních cyklech vytvoří, a po náhodném čase opět smaže, událost, kontakt a poznámku. Jedná se o reálné položky s fiktivními údaji. Ve čtvrtém cyklu, který je nejkomplexnější, probíhá kompletní správa e-mailů, což zahrnuje jejich synchronizaci, vyhledávání v e-mailech, odesílání e-mailů, získávání obsahu e-mailů (`FETCH`), jejich čtení a následné smazání. Každý z těchto čtyř cyklů je separátně ovládatelný a lze nastavit jeho frekvenci.

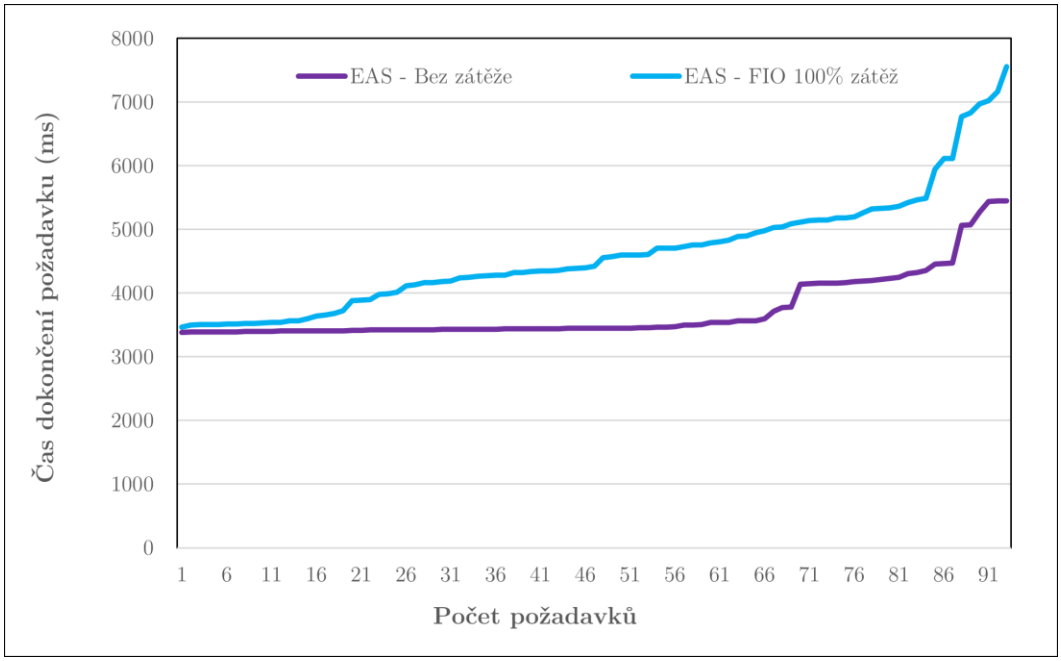
Výsledkem je aplikace, která pokrývá veškeré nejdůležitější funkcionality protokolu ActiveSync v sekvenčním scénáři. Rozestupy mezi jednotlivými operacemi jsou plně konfigurovatelné a jednotlivé části scénáře lze vynechávat. Stejně jako v případě WebDav modulu, pokud by se sekvenčnost scénáře ukázala jako nevhodná, bylo by nutné zajistit více různých scénářů simulující více druhů uživatelské používání, obdobně jako tomu je v případě WebClient modulu.

K přípravě zdrojového kódu z terminálu pro Apache JMeter slouží program `activesync.sh`. Jeho funkcionalita je téměř shodná jako v případě modulu WebDav. Jediným zásadním rozdílem je nutnost vstupního souboru s názvy zařízení, která se budou registrovat na cílovém serveru. Tyto názvy musí být unikátní

z důvodu sekvence SyncKeys, se kterou tvoří pár. Jak bylo popsáno v teoretické části, pokud by pod identifikátorem jednoho zařízení sekvence SyncKeys nešla ve správném pořadí a byla narušena, tak je komunikace považována za neplatnou. K čemuž by došlo v situaci, kdyby na dvě vlákna byl aplikován stejný název zařízení.

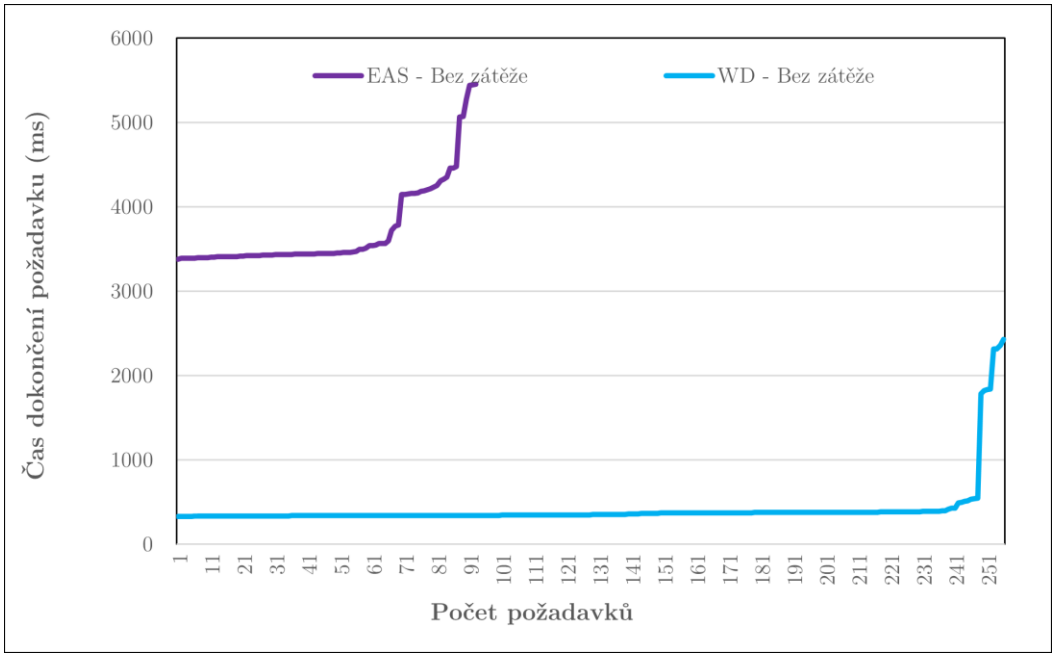
Obrázek 7: Struktura modulu EAS v program JMeter (obsahující optimalizační časovače)

Tak jako v případě WebDav protokolu, i zde bylo součástí přání zadavatele porovnání rychlosti protokolu EAS bez zátěže a při maximálním vytížení serveru. Zde byl nárůst doby vyhotovení v případě většiny požadavků mezi 15-40 %. Nenastává zde žádný zásadní výkyv, jako tomu bylo v případě WebDav. Následující graf toto srovnání zobrazuje. Jedná se o všechny odeslané požadavky z výše popsaného scénáře seřazené dle doby vyhotovení.



Graf 10: Porovnání protokolu EAS bez zátěže/se zátěží (Požadavky seřazeny vzestupně dle doby vyhotovení)

Posledním grafem k této kapitole je porovnání protokolu WebDav a EAS. Ačkoliv se může zdát, že protokoly jsou značně odlišné, tak jejich funkcionalita je v určitých případech užití velmi podobná, pouze se liší způsob, jakým dojdou k cíli. Téměř vše, co umí protokol WebDav, je obsaženo uvnitř protokolu EAS, který ovšem nabízí mnoho dalších funkcionalit. Již v minulosti se uvažovalo o nahrazení protokolu EAS a podle výsledků porovnání je zřejmé proč. Toto porovnání na datech nevytížených serverů ilustruje, jak moc je protokol EAS pomalý oproti WebDavu. Konkrétně průměrná doba vyřízení požadavku je v případě WebDav protokolu 417 ms. Naproti tomu v případě EAS se jedná o 3770 ms, což z něho dělá více než 9x pomalejší protokol. V obou případech byl dostupný rozdílný počet vyřízených požadavků k analýze a z toho důvodu nejsou grafy stejné v rámci osy X. To ovšem nemá vliv na ilustraci časové neefektivity protokolu EAS, kde je možné pozorovat, že ani nejpomalejší požadavek protokolu WebDav není pomalejší nežli nejrychlejší požadavek protokolu EAS.



Graf 11: Porovnání protokolu EAS vs. WebDav (Požadavky seřazeny vzestupně dle doby vyhotovení)

3.3 Wrappery (2. vrstva)

Každý z výše zmíněných modulů, který je implementován v Bash nebo v Apache JMeter je nutné nějakým způsobem ovládat, spravovat jeho jednotlivé procesy nebo připravit jeho parametry před spuštěním. A přesně k tomu slouží wrapper. Takové wrappery tedy jsou čtyři – SMTP, WebClient, WebDav a EAS. IMAP jako jediný svůj wrapper nemá, jelikož správa vláken je díky komplexnosti jazyka Python implementována přímo uvnitř programu. Každý z wrapperů je implementován pro specifické potřeby daného modulu ve skriptovacím jazyce Bash. Samotné moduly jsou 3. vrstvou – tou nejnižší. Wrappery jsou prostředník mezi kontrolním skriptem (1. vrstvou) a jednotlivými moduly.

Nejjednodušší mezi wrappery je WebMail wrapper, který má na starost správu jednotlivých procesů, a to především jejich korektní spuštění a případné ukončení. Kromě toho z konfiguračního souboru vyčítá e-mailové adresy, na které má cílit a předává je 3. vrstvě.

O něco komplexnější je wrapper pro SMTP, který kromě výše zmíněného pro WebMail má navíc na starost zajištění korektního vstupu pro SMTP modul a tvorbu pouze tolika procesů, kolik je nutné. Toto je prováděno z důvodu, že jeden SMTP proces simuluje desítky až stovky uživatelů, jak bylo popsáno v implementaci SMTP. Každý proces má ale maximální počet uživatelů, které může při daných parametrech simulovat.

Odlišný od předchozích je WebDav wrapper. Tento wrapper navíc musí zajistit, aby před samotným spuštěním skriptů byly připraveny také zdrojové soubory ve formátu XML pro JMeter. Dalším rozdílem je zpracování logů. Jakmile je obdržen signál ukončení, tak tento signál je distribuován z wrapperu do 3. vrstvy

(*webdav.sh*), která korektně ukončí veškeré procesy na vzdáleném JMeter serveru a zajistí stažení informací o běhu (logů). Jakmile wrapper zjistí, že veškeré *webdav.sh* procesy skončily, tak veškeré logy, ze všech procesů a vláken spojí dohromady, chronologicky seřadí jednotlivé záznamy a přidá je do globálního logovacího souboru modulu. V případě WebDav není možné nechávat program zapisovat do lokálních souborů rovnou, jelikož se program nespouští lokálně, nýbrž na vzdáleném serveru. Zároveň je doporučené, aby každý Apache JMeter proces měl za běhu svůj vlastní log z důvodu případných časově závislých chyb, které program umí řešit pouze v rámci vláken, nikoliv v rámci procesů.

Poslední EAS wrapper je rozšířením předchozího WebDav wrapperu. Podporuje veškeré funkcionality jako WebDav wrapper, ale s tím rozdílem, že navíc zajišťuje tvorbu souborů s názvy zařízení pro každý spouštěný proces. Zároveň je dobré zmínit, že oba tyto wrappery vytváří dva téměř totožné zdrojové kódy, které se poté odesílají na spuštění na server. Jedná se o zdrojový kód s maximálním počtem vláken, které jsou jedním procesem na JMeter serveru podporovány a pak případným dopočteným počtem vláken do specifikovaného počtu uživatelů. Pokud je tedy maximální počet vláken pro jeden proces 30 a wrapper obdržel informaci o tom, že je nutné simulovat 100 uživatelů, tak spustí 3x Apache JMeter proces, jehož zdrojový kód je nakonfigurovaný na 30 uživatelů/vláken a 1x Apache JMeter proces, který je nakonfigurovaný na 10 uživatelů.

Důležitost 2. vrstvy tedy vyplývá z toho, že se jedná o komplexní komunikátor mezi kontrolní vrstvou, která má na starost pouze odeslání informací o počtu simulovaných uživatelů a nejnižší vrstvou, která vykonává veškerou simulační práci. Kromě komunikace zajišťuje také datové vstupy.

3.4 Kontrolér (1. vrstva)

Nejvyšší vrstvu lze nazvat jako kontrolér, jelikož určuje práci všech nižších vrstev a jedná se o prostředníka mezi wrappery a uživatelem. Jeho práce spočívá v editaci konfigurovatelných proměnných napříč všemi programy na hierarchicky nižších vrstvách a předání informací wrapperům o počtu simulovaných uživatelů na jednotlivých protokolech.

Pro editaci souborů slouží program *configedit.sh*. Tento program prečte obsah souboru *globalconfig.cfg*, který obsahuje všechny konfigurovatelné proměnné napříč moduly. Tyto proměnné přepíše ve všech zdrojových kódech, kde je potřeba, a zajistí tak správné spuštění všech modulů. Je tedy před spuštěním důležité zkontrolovat konfigurační soubor a případně hodnoty změnit. Zásadní je neměnit formát konfiguračního souboru, ale pouze hodnoty jednotlivých proměnných.

Jakmile je editace zdrojových a konfiguračních souborů hotová, program dle specifikovaných parametrů pro počet uživatelů na WebClient, SMTP, IMAP, ActiveSync, eMClient a Outlook spustí wrappery jednotlivých modulů. Jakmile kontrolér obdrží signál ukončení, tak tento signál předá všem wrapperům, které spustil, a ti již dále distribuují tento signál nejnižší vrstvě a zpracují případné výstupy, jak bylo popsáno výše.

4 Postup optimalizace nástrojů

4.1 Popis problému optimalizace frekvence operací

Po dokončení kroků popsaných v implementační části, umí každý z modulů vytvářet požadavky na danou část aplikace, zaznamenat odezvu a případně nějakým způsobem rozhodnout o dalším požadavku, který odešle. Jako zásadní problém se ovšem v rámci testování jednotlivých simulačních modulů ukázalo, že testy zásadně ovlivňuje frekvence operací, které každý uživatel odešle. Z toho důvodu je většina z modulů v tomto stavu zatím jako simulační program nepoužitelná a spíše se jedná o stresové testy nežli o testy simulační.

Je vhodné klást si otázku, jestli je nutné testy optimalizovat natolik, aby správně reprezentovaly jednotlivé uživatele. Odpovědí je ano. Pokud by bylo možné z existujících statistik určit kolik operací na daném protokolu průměrný uživatel provede, tak by celá tato část byla nepotřebná, ale jak bylo zmíněno v úvodních kapitolách práce, takové potřebné podklady pro vypracování neexistují a získat je v aktuálních firemních podmínkách není možné. V ideální situaci, kdy by tato data byla dostupná, by stačilo postupným zvyšováním počtu odeslaných požadavků (ramp-up test) počkat na situaci, kdy bude server vytížen takovým způsobem, že bude považován za nespolehlivý. V takové situaci by byl zaznamenán počet operací za časový úsek, které ještě byl server schopen odbavit, než byl prohlášen za nespolehlivý a tento počet by se převedl na počet uživatelů pouhým vydělením.

Ovšem tento ideální scénář převodu počtu operací na počet uživatelů není realistický. Z toho důvodu bylo nutné pečlivým porovnáváním reálných serverů se servery simulačními optimalizovat frekvenci odesílaných operací z jednotlivých instancí simulačních modulů. To znamená, že v rámci produkčního cloudového prostředí byly vytipovány vzorové servery, jejichž hardware bude reprodukován do simulačního prostředí. Poté, co budou zajištěny dva identické servery, tak na simulačním serveru bude spuštěna simulace o stejném počtu uživatelů jako se bude v danou chvíli vyskytovat na produkčním serveru s kterým je porovnání tvořeno. Po určeném časovém úseku bude simulace ukončena a bude vyhodnoceno procesorové a paměťové vytížení na jednotlivých protokolech v případě simulačního i produkčního prostředí. Po vyhodnocení těchto dvou aspektů bude nutné upravit čekací dobu mezi jednotlivými požadavky. Toto bude nutné opakovat až do chvíle, kdy metriky vyhodnocování podobnosti časových řad budou pod stanovenou úrovní. Vedle těchto metrik bude také nutné data opticky kontrolovat.

Z výše zmíněného tedy vychází, že je nutné zajistit prostředí, kde bude možné vytvářet jednotlivé simulační servery. K tomu bude sloužit firemní interní testovací systém založený na Linuxovém nástroji `virsh` – uživatelské rozhraní pro správu virtuálních serverů. Tyto servery je nutné také monitorovat, aby bylo možné získat statistiky o jejich vytížení a zajistit nad nimi vysokou kontrolu. Pro tento účel bude založen monitorovací server Zabbix 6.0. Na závěr bude nutné veškeré výstupy

dát dohromady a vhodným způsobem je analyzovat v Pythonu s knihovnami *mathplotlib* a *numpy*.

4.2 Tvorba testovacího prostředí

Pro tvorbu testovacího prostředí již existuje firemní server nakonfigurovaný pro jednoduchou tvorbu a správu virtuálních serverů pomocí Linuxového nástroje *virsh*. Pro konfiguraci simulačních serverů bylo potřeba z produkčních serverů zjistit: počet jader, velikost RAM, typ databáze, typ souborového systému a jeho velikost. Pro získání těchto informací o vytipovaných produkčních serverech bylo využito Linuxových příkazů *lscpu*, *lsmem*, *df* a *lshw*. [39] U všech testovacích serverů je databáze vedena na separátním serveru a stejně tak uživatelská data jsou na připojeném NFS souborovém systému ve složce */mnt*. Velikost připojeného NFS systému se liší dle jednotlivých simulačních serverů a je založena na velikosti produkčních serverů. Specifikace NFS je shodná s produkčními servery zákazníků. Základní místo pro systém na každém vytvořeném simulačním serveru bylo stanoveno na 50 GB. V případě připojeného NFS systému se jedná až o jednotky TB.

Kromě cílových serverů, které zastupují funkci produkčních zákaznických serverů, bylo nutné zajistit jednotné spouštěcí prostředí pro eliminaci možných odchylek z důvodů vytížení serveru jiným procesem. Pro tento účel byly vytvořeny separátní JMeter servery a jeden hlavní exekuční server, který bude mít na starost celou testovací strukturu jednotlivých modulů. V případě testování EAS a WebDAV využije exekuční stanice výpočetní síly dvou JMeter serverů.

Z důvodu nutnosti odštěpit část z již existujícího firemního projektu, kterého není autor této práce součástí, byla tato kapitola po analytické práci autora předána jiné části týmu ve firmě IceWarp, která zajistila tvorbu vyspecifikovaných serverů pomocí Ansible software. Výsledkem jsou již zmíněné JMeter servery a exekuční stanice, ale také 2 simulační servery o malé velikosti (typicky desítky uživatelů) – 4 jádra CPU, 8/12 GB RAM, 2 simulační servery pro středně velké zákazníky (stovky až jednotky tisíc uživatelů) – 8 jader CPU, 12/16 GB RAM a 1 simulační server pro extra velké zákazníky – 12 CPU, 24 GB RAM. Co se týče typu procesoru, tak se konkrétně jedná o Intel Xeon 4210R Silver, stejně jako v případě produkčních serverů.

Autor dále sám zajistil tvorbu uživatelů a vhodné distribuce e-mailových schránek a veškerých dat. Na každém serveru bylo nutné na základě dat z analytické části vytvořit domény a uživatele a těm nakopírovat vzorové e-mailové schránky ve specifikované frekvenci. Pro každou uživatelskou skupinu byly vytvořeny tři různé verze e-mailových schránek, které byly náhodně voleny. Všechny tyto schránky vycházely z originálních statistických dat, každá ovšem měla jinou velikost v rámci povoleného rozmezí. Poté bylo třeba každý server nakonfigurovat dle jeho velikosti a naplnit ho kromě e-mailových dat také daty GroupWare. Mezi GroupWare data patří kalendáře, poznámky, úkoly, dokumenty a kontakty. Pro tento účel stačilo obohatit WebClient modul o nové funkcionality importu dat přes IceWarp API. Zajištění dat k importu proběhlo ve všech případech ruční tvorbou

v IceWarp WebClientu a následným exportem ve vhodném formátu s výjimkou kalendářů na které byl využit modifikovaný, volně dostupný program *ical-fake-meetings-generator*. [40] Na vlastnosti dat typu GroupWare nebyla provedena žádná analýza, jelikož jim není přisuzována taková důležitost jako e-mailovým schránkám a z toho důvodu na všech uživatelích bude pouze jedna stejná vzorová struktura sestávající se z jednotek dokumentů, úkolů, poznámek apod.

Na závěr pro zajištění funkčního testovacího prostředí bylo potřeba nakonfigurovat lokální DNS server, jelikož je celé prostředí nasazené uvnitř uzavřené sítě. Zároveň by nebylo vhodné pouze za účelem simulace pořizovat domény u poskytovatele DNS a z toho důvodu veškerá komunikace probíhá výhradně na doménách lokálních. Tento DNS server zajišťuje bezchybnou komunikaci mezi jednotlivými servery a je na serverech nastaven jako primární.

	U1	U2	U3	U4	U5	U6	U7
CPU 4, RAM 8	26,6 %	26,6 %	20,0 %	20,0 %	6,7 %	-	-
CPU 4, RAM 12	41,6 %	15,5 %	7,1 %	25,0 %	9,6 %	1,2 %	-
CPU 8, RAM 12	52,0 %	31,0 %	6,6 %	5,2 %	4,0 %	0,8 %	0,4 %
CPU 8, RAM 16	49,5 %	11,2 %	8,0 %	11,0 %	8,2 %	10,4 %	1,7 %
CPU 12, RAM 24	-	-	-	-	-	-	-

Tabulka 5: Rozložení uživatelských skupin na testovacích serverech

Vzhledem k nedostatku prostředků na serverech firmy v době tvorby testovacího prostředí, bylo nutné vyřadit z testování největší server. Zároveň v době tvorby uživatelů nebylo možné vyhodnotit data o distribuci různých uživatelských skupin na dostatečně velkém statistickém vzorku, což může vytvářet lehké zkreslení dat o distribucích jednotlivých uživatelských skupin. Vzhledem k formám testů by tento rozdíl neměl být zásadní.

4.3 Tvorba monitorovacího prostředí

Monitorování serverů a vyhodnocování ukládaných metrik je základním úkonem každé firmy, která si sama zajišťuje svou infrastrukturu pro hostování aplikace. Pro monitorování infrastruktury existuje mnoho aplikací, některé fungující přes agenty, některé bez nich. Mezi nejznámější příklady takových aplikací patří Zabbix, Prometheus, Nagios nebo Sematext Monitoring. [41] Společnost IceWarp si zajišťuje monitoring své interní, ale také produkční infrastruktury právě pomocí aplikace Zabbix 6.0. V rámci této práce bude prostředí Zabbixu zachováno a vytvořen zcela nový Zabbix server, který bude mít na starost monitoring vytipovaných produkčních, exekučních a simulačních serverů.

Ačkoliv by se zde mohla nabízet možnost využít již existujícího monitoringu produkčních serverů a pouze vhodně nakonfigurovat nově vytvořené simulační klienty, tak to nelze považovat za vhodné. Je nutné pamatovat na to, že data, která poté budou analyzována, tak budou muset být vytažena přímo z databáze, a to v případě velkých produkčních monitorovacích systémů může zbytečně zasahovat do výkonu a stability monitorovacího serveru. Z toho důvodu bude striktně odděleno

produkční monitorovací prostředí a testovací monitorovací prostředí pro účely tohoto projektu, ačkoliv data z vytipovaných produkčních serverů tím pádem budou muset být odesílána na dva monitorovací systémy zároveň.

Instalace čistého Zabbixu byla dedikována na ostatní členy týmu, obdobně jako v případě tvorby prázdných simulačních serverů. Výchozím bodem je tedy prázdná instalace Zabbixu 6.0. Pro zajištění monitorování jednotlivých serverů bude nutné nastavit tzv. „autodiscover" a data, která je žádáno sbírat na straně Zabbixu. Zároveň bude potřeba správně nakonfigurovat jednotlivé servery (klienty), které mají být monitorovány. V tomto modelu je nazýván monitorovací server jakožto Zabbix-Server a monitorované servery Zabbix-Agent. [42]

4.3.1. Nastavení Zabbix serveru

Na úvod, po čisté instalaci Zabbixu, je vhodné si nastavit „templates" a „items", jako specifikaci, co se má na daných serverech monitorovat. Velká část z těchto templates byla dostupná z produkčních serverů. Tudíž pomocí funkce export/import z produkčního Zabbixu bylo možné naklonovat určité templates. Aby tato funkce fungovala bezchybně, tak je nutná shoda verzí aplikace Zabbix mezi servery. Tato funkcionalita zajistí reprezentaci monitorovacích předpisů ve formátu XML, které lze poté importovat na nový Zabbix server pomocí jednoho tlačítka. [43] Po importování z produkčního monitorovacího prostředí byly tyto templates pročištěny o položky zbytečné pro tuto práci. Jako zásadní položky pro monitoring každého serveru byly autorem ve spolupráci s vedoucími projektu vyhodnoceny:

- Celkové vytížení serverového CPU, vytížení CPU jednotlivými službami aplikace IceWarp (IM, Control, IMAP/POP3, GW, SMTP)
- Celkové využití paměti RAM, využití RAM jednotlivými službami aplikace IceWarp
- Počet aktivních spojení na službu
- Celkový počet registrovaných uživatelů na daném serveru
- Odhad procentuálního počtu aktivních uživatelů ze všech registrovaných na daném serveru

Je nutné zmínit, že toto nejsou jediné položky, které jsou v rámci těchto serverů monitorovány. Takových položek je zhruba 100. Výše vyjmenované jsou položky, které jsou považovány za ty naprosto zásadní pro budoucí vyhodnocování přesnosti simulačních modulů. V případě některých z nich bylo nutné je také doimplementovat.

Za demonstraci takové dodatečné implementace monitorované položky může být například celkový počet registrovaných uživatelů na daném serveru. Po rozkliknutí template a záložky items je v pravém horním rohu možnost „add item". Po nastavení typu položky, názvu, frekvence sbírání této informace a doba jejího uchování je možné ji uložit. [44] Každá z těchto položek je určená především polem „Type" a „Key". Type specifikuje, jakým způsobem se data na server dostanou. Celkem je těchto specifikací ve verzi 6.0. dostupných 17. Typicky se může jednat o

situaci, kdy agent zasílá data na server nebo naopak Zabbix server je zodpovědný za jejich získání apod. Klíč je příkaz, kterým se daná data získávají. Zabbix sám o sobě podporuje celou řadu příkazů, které zjednodušují proces konfigurace a získávání dat. [45] V tomto případě, kdy cílem je získat celkový počet registrovaných uživatelů na daném serveru, tak z hlediska klienta musí být zajištěno, že daná sbíraná hodnota bude uvnitř nějakého souboru na monitorovaném serveru. Ve chvíli, kdy takový soubor je na monitorovaném serveru dostupný, tak je možné klíč nastavit na hodnotu *vfs.file.contents[/path/to/file]*. Tento klíč zajistí přečtení obsahu souboru.

Způsobů, jakými lze jednotlivé monitorované položky nastavovat, je nespočet a pro jejich kompletní popis je doporučeno projít si celou obsáhlou dokumentaci na oficiálních stránkách aplikace Zabbix. [46] Kromě těchto položek bylo také nutné zajistit obrazovky, které budou všechna data vhodně graficky vizualizovat, takzvané „dashboards“ a případné „triggers“ pokud nějaká data překročí stanovenou mez. Ačkoliv jejich konfigurace proběhla, detailní popis těchto kroků je v této práci vynechán s odkazem na oficiální dokumentaci. [46]

Poté, co jsou připravena pravidla pro monitorování, tak je vhodné pro ulehčení práce s jednotlivými Zabbix agenty zavést na serveru „autoregistration rule“. Toto pravidlo slouží k tomu, že pokud kterýkoliv server, pokud odešle svá data směrem na Zabbix server, tak bude zaregistrován a bude mu přidělený nadefinovaný „template“, skupiny a jiné specifikované nastavení. Jedná se tedy o dobrý nástroj pro automatickou registraci jednotlivých agentů, kteří budou monitorováni, bez nutnosti zásahu administrátora. [47]

4.3.2. Nastavení Zabbix klientů

Nastavení klientů je závislé na nastavení monitorovacích serverů. Podle specifikace sbírání jednotlivých položek je nutné nastavit také klienta, který musí být nastaven pro případné odesílání dat na server, povolení určitých příkazů ze strany serveru apod. Instalace probíhá ze standardních Linux repozitářů. Po stažení je nutné upravit konfigurační soubor *zabbix_agent.conf*, který je v adresáři */etc*, případně */etc/zabbix*. Pokud je nutné monitorování na dvou separátních Zabbix serverech, tak soubor může vypadat například následovně:

```
PidFile=/run/zabbix/zabbix_agentd.pid
LogFile=/var/log/zabbix/zabbix_agentd.log
LogFileSize=0
Server=<PROD-PROXY>,<MY-ZABBIX-SERVER-IP>
ServerActive=<PROD-PROXY>:10071,<MY-ZABBIX-SERVER-IP>:10051
ListenIP=<THE-CLIENT-SERVER-IP>
ListenPort=10050
Include=/etc/zabbix/zabbix_agentd.d/
Timeout=30
AllowKey=system.run[chronyc tracking*]
AllowKey=system.run[ss*]
AllowKey=system.run[cat /proc/net/sockstat*]
```

Úryvek kódu 7: Podoba konfiguračního souboru zabbix_agent.conf

V přiloženém příkladu si lze povšimnou tří řádek ve formátu *AllowKey=system.run[command].* Tato řádka slouží ke specifikaci, co za Linuxový příkaz má Zabbix server povolení na daném klientu spustit. Jedná se tedy o příklad toho, kdy je důležité vědět, co je ze strany Zabbix serveru monitorováno, a jakým způsobem, abychom to případně mohli ze strany klienta povolit. Zároveň je důležité zmínit, že používání *system.run* by mělo být minimalizováno a pokud je přece jen nutné, tak je doporučeno povolit pouze specifické příkazy. V případě povolení všech příkazů pomocí *AllowKey=system.run[*]* by útočník s přístupem k monitorovacímu serveru mohl mít téměř plnou kontrolu nad monitorovaným serverem. V aktuálních podporovaných verzích Zabbixu je toto zakázané, a tudíž to není ani možné nastavit, každopádně i v takovém případě je dobré mít na mysli, že je vhodnější příkaz plně specifikovat a vyhnout se obecnosti. Pro kompletní list možností konfiguračního souboru je dostupný v dokumentaci. [48]

Po nastavení konfiguračního souboru klienta stačí zajistit restartování služby *zabbix-agent* a povolení komunikace se specifikovaným serverem na firewallu a monitoring bude funkční. V případě, že je firewallem *iptables*, tak takové pravidlo může vypadat například takto:

```
-A INPUT -s <zabbix_server_ip> -p tcp -m tcp --dport 10050 -m
state -state NEW,ESTABLISHED -j ACCEPT
```

Úryvek kódu 8: IPtables pravidlo pro povolení komunikace mezi serverem a klientem

4.4 Tvorba analyzačního prostředí

V tuto chvíli zbývá jen data vhodným způsobem reprezentovat, vyhodnocovat a optimalizovat jednotlivé moduly. Díky nově vytvořenému Zabbix serveru je možné bez starostí o kritickou infrastrukturu vytvářet SQL dotazy přímo nad databází, která statistiky o monitorovaných serverech zaznamenává. K tomu slouží následující SQL dotaz:

```
SELECT clock - ${start_time}, value FROM ${tablename}
 WHERE clock > ${start_time}
   AND clock < ${end_time}
     AND itemid =
     (
         SELECT itemid FROM items WHERE name = ${stat}
             AND hostid =
             (
                 SELECT hostid FROM hosts WHERE host =
${server}
             )
     );
```

Úryvek kódu 9: SQL příkaz pro získání monitorovaných dat ze Zabbix databáze

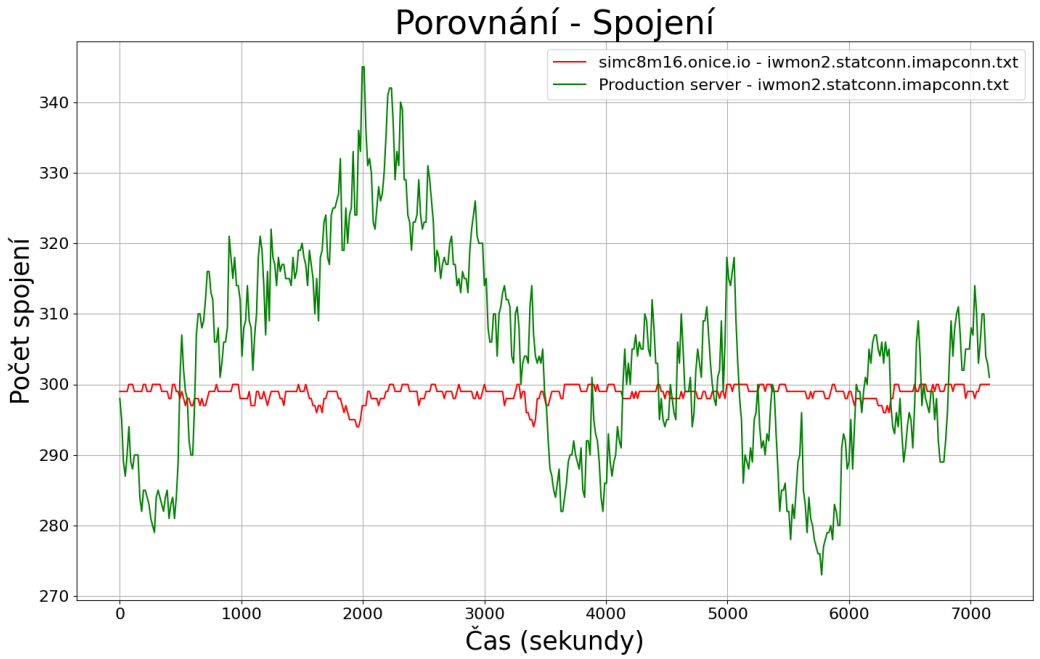
Tento dotaz na základě názvu monitorované položky, času začátku testu a jména serveru vrátí ze Zabbix databáze dva sloupce (čas od začátku testu a hodnotu specifikované položky v daný čas). Kromě výše zmíněného je nutné vědět, jaký je

nastavený u dané položky „Type of information“, jelikož podle toho Zabbix databáze ukládá jednotlivé položky do různých tabulek své databáze.

Pro tato data je vytvořen v jazyce Python program, který zajistí výpočet mediánu a průměru z dat a vykreslí porovnání grafů sledované položky do grafu. Kromě průměru a mediánu jsou také sledované hodnoty, které se používají k podrobnější analýze podobnosti časových řad. Konkrétně se jedná o hodnoty MAPE, MAE a DTW. Ačkoliv metrik pro porovnávání podobnosti časových řad je mnoho, tyto byly zvolené jako ty nejvhodnější dle dostupných zdrojů. [49] Nejlépe pro tyto účely vychází dle dostupných vědeckých prací právě DTW. [50] Jako hlavní ukazatel pro vyhodnocení podobnosti časových řad je stále využíván průměr a medián vykreslovaných dat, ovšem podstatu zmíněných ukazatelů pro vyhodnocování podobnosti nelze opomínat, jelikož na rozdíl od průměru a mediánu umí metriky mnohem lépe zohlednit tvary grafů a podobnost jejich vlastností. [49]

Pro každou ze sledovaných metrik zajistí program také vykreslení grafu počtu spojení ve stejném časovém úseku. Cílem je vypočtené hodnoty MAPE, MAE, DTW minimalizovat tím, že frekvence odesílaných operací je optimalizována co nejblíže realitě. Zároveň je také cílem, aby se průměr a medián ze simulačních dat a produkčních dat, pokud možno rovnaly. Výstupy z tohoto Python programu jsou hlavním vodítkem, jak frekvenci jednotlivých operací v případě všech modulů upravovat.

Graf 12: Příklad grafu – porovnání produkčního a simulačního prostředí zvolených metrik (Využití CPU v tomto případě)



Graf 13: Příklad grafu – počet spojení časově korespondující s grafem CPU výše

Program umí pracovat ve dvou režimech – jeden je vhodný pro optimalizaci podle počtu spojení, kdy jsou pouze vykreslována reálná data metrik a spojení ze simulačního a produkčního prostředí (`MODE=1`). Ten druhý požaduje z produkce na vstupu graf průměrného vytížení na jednoho aktivního/licencovaného uživatele (`MODE=0`). Tento graf metrik je průběžně násoben počtem spojení v simulačním prostředí a porovnáván s grafem metrik simulačního prostředí. Což je vhodné pro optimalizaci podle počtu licencovaných/odhadovaných aktivních uživatelů.

4.5 Aplikace optimalizačních a analyzačních technik

Na optimalizaci modulů lze pohlížet ze dvou pohledů. Jedním pohledem je situace, kdy by cílem jedné instance modulu bylo simulovat jedno spojení. Toto je vhodnější způsob, který ovšem nelze vždy praktikovat. Z toho důvodu se nabízí druhý pohled na optimalizaci, který ji vede vůči počtu licencovaných uživatelů, ze kterých se koeficientem určí počet uživatelů, kteří se podíleli na vrcholné zátěži serveru.

4.5.1. Optimalizace pomocí počtu spojení

Tento způsob optimalizace využívá programu z kapitoly 4.4. v (`MODE=1`). Při reálných testech se ukázalo, že tato optimalizace je vhodná pouze pro moduly, které si i při své neaktivitě udržují aktivní relaci. Takovým protokolem je v případě aplikace IceWarp pouze IMAP, kdy klient, pokud zrovna od serveru nic nevyžaduje, tak posílá příkaz `NOOP`/`IDLE`, který udržuje relaci aktivní.

Opačným příkladem může být protokol SMTP, který již z podstaty jeho fungování žádný takový příkaz nenabízí. Obdobně tomu je i u jiných protokolů, které jsou v rámci simulací využívány. Ačkoliv například v případě připojení pomocí EAS je možné sledovat, kdy naposledy byla provedena nějaká operace, jedná se o využívání dat uložených přímo v databázi a aplikace IceWarp toto v sobě nemá integrováno. Tudíž v případě všech jiných protokolů bude nutné optimalizaci provádět vzhledem k počtu licencovaných uživatelů.

V případě IMAP bylo provedeno zhruba 10 testů v délce trvání 1 až 3 hodiny, které si dávaly za cíl měnit krajní hodnoty intervalu *[$MIN_WAIT_TIME, $MAX_WAIT_TIME]*. Tyto testy byl vykonávány pro 40, 100 a 180 simulovaných uživatelů. Nakonec byly tyto hodnoty stanovené na *[35, 215]*. Tato optimalizace v tuto chvíli není považována za finální, jelikož se jednalo o porovnání s jednotkami serverů jejichž spojení nebyla zprůměrována. Vzhledem k tomu, že chování napříč různými produkčními servery je velmi rozdílné, tak i v případě IMAPu bude provedena sekundární optimalizace pomocí počtu licencovaných uživatelů.

4.5.2. Optimalizace pomocí počtu licencovaných uživatelů

Tento způsob optimalizace využívá programu z kapitoly 4.4. v (*MODE=0*). Aby bylo možné tento mód použít, tak je třeba získat průměrná data pro jednotlivé služby aplikace IceWarp (Control, IMAP/POP3, GW, SMTP). Toho bylo docíleno pomocí programu *analyse_production_data.py*, který je součástí analyzačního prostředí. Optimalizace samotná se po otestování na menších simulačních serverech s 4 jádry a 8/12 GB RAM přesunula na největší server s 8 jádry a 16 GB RAM. Pro všechny produkční servery se stejným hardware bylo tedy zajištěno získání týdenních dat z databáze Zabbixu ohledně vytěžování jednotlivých služeb aplikace IceWarp. Tyto soubory s týdenním vytížením jsou vstupem programu. Dalším důležitým vstupem je koeficient aktivních uživatelů, který říká, kolik procent z licencovaných uživatelů se pravděpodobně podílelo na největším vytížení serveru. Pro tento koeficient v aktuální chvíli ze strany firmy neexistují podklady, ovšem po experimentálním přezkoumání dat v monitorovacím systému byla hodnota nastavena na 0.35 (35 % aktivních uživatelů) pro všechny zkoumané servery.

Program poté hledá v těchto datech číselnou řadu s největším součtem o požadované délce. Tím zajistí nalezení časového okna, kdy byl server nejvíce vytížen. Toto je provedeno pro každý server a data jsou vzhledem k počtu uživatelů na serverech zprůměrována dvěma způsoby:

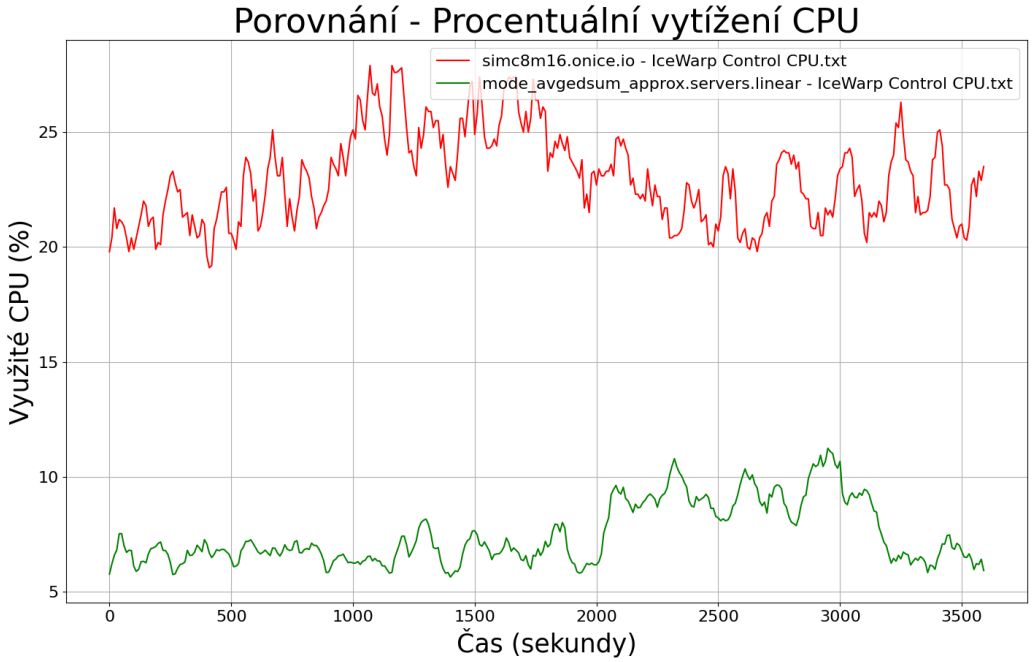
1. Hodnota v časovém bodě je vydělena počtem uživatelů daného serveru. Výsledky této operace jsou napříč servery pro bod ve stejném čase sečteny a vyděleny počtem serverů. Tento způsob se ukázal jako nevhodný pro data o procesoru, ale byl využit při optimalizaci SMTP protokolu, kdy se jednalo o rostoucí graf. V případě této optimalizace bylo také nutné upravit hledání číselné řady na tu s největším rozdílem prvního a posledního prvku, nikoliv největší součet.

2. Hodnoty v časovém bodě jsou napříč servery sečteny a vyděleny součtem uživatelů na všech serverech. Tento výsledek je poté ještě vydělen počtem serverů. Tento způsob se ukázal jako skvělá aproximace pro průměrné zatížení vytvářené jedním uživatelem.

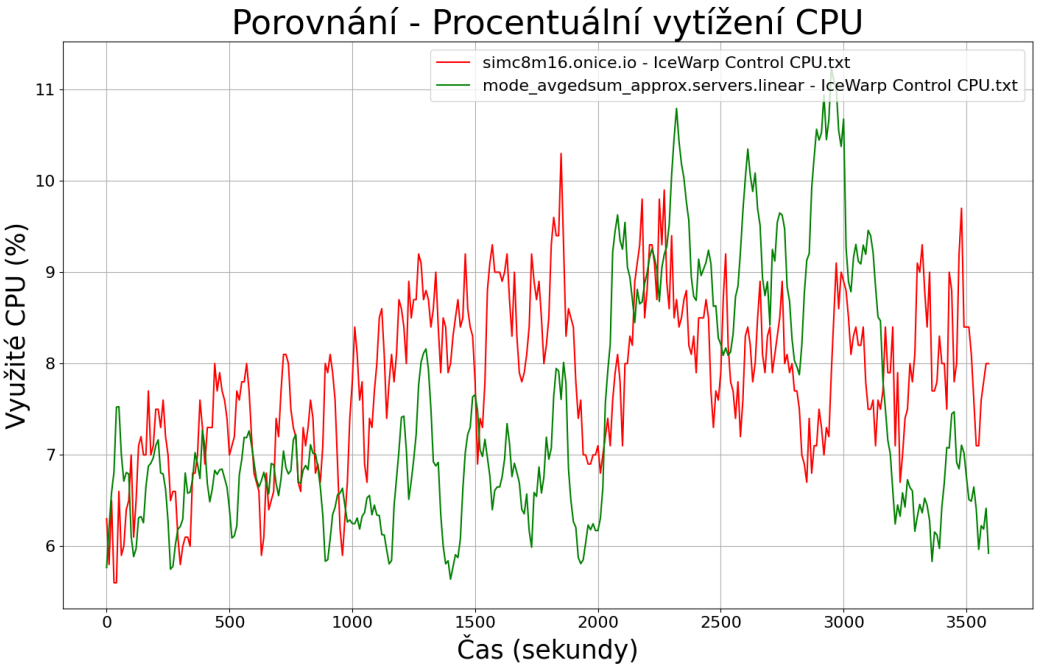
Výsledkem je tedy soubor obsahující řadu dat o specifikované délce. Data této řady reprezentují průměrné procentuální zatížení CPU jedním uživatelem na dané službě. Při porovnávání těchto dat se simulací je tedy třeba celou řadu vždy vynásobit počtem uživatelů na simulačním serveru. Program kromě toho umí také dohledat, jaký byl v daných časových rámcích na službě počet spojení, ale jak již bylo zmíněno, toho v aktuální chvíli vzhledem ke specifikám určitých protokolů nebude využito.

V tuto chvíli je vše připravené pro samotnou optimalizaci v (`MODE=0`). Za příklad této optimalizace bude uveden modul WebClient, který vytěžuje především služby Control a GW. Stejným způsobem byly optimalizovány veškeré ostatní moduly celého projektu, pouze s tím rozdílem, že se optimalizovaly jiné hodnoty rozestupu mezi operacemi a jiné spektrum operací. V případě nějakých modulů se jednalo o změnu pouze dvou hodnot, u komplikovanějších modulů, jako například WebClient se jednalo o změnu více úrovní nečinností (10x více hodnot). Sledovány vždy byly všechny služby, které jsou daným modulem vytěžovány.

WebClient modul vyžadoval tedy přidání příkazů pro nečinnost mezi funkce připravených scénářů. Pro tento účel je funkce SleepForRandom, která má 10 úrovní nečinnosti, kde 1 je nejnižší a 10 je nejvyšší. Tato funkce s parametrem 1-10 byla proložena do připravených scénářů dle odhadů reálného uživatelského čekání a poté byla optimalizována pouze samotná funkce a krajní hodnoty intervalu, ze kterého se doba nečinnosti náhodně vybírá. Například pro úroveň nečinnosti 1 se interval vyšplhal z původního odhadu `[2, 7]` na `[28, 37]`. V případě úrovně 10 se jednalo o skok z `[100, 150]` na `[310, 550]`. Jednalo se o mnoho několikahodinových testů, které byly poté vyhodnoceny výše zmíněnými programy. Hodnoty byly následně upravovány až dokud vytížení nebylo na simulačním a aproximovaném produkčním serveru pro daný počet aktivních uživatelů téměř shodné. Testy byly vykonávány pro 50, 150 a 300 uživatelů. Na následujících grafech jsou simulační data znázorněna červeně a produkční zeleně.



Graf 14: První test WebClient modulu pro 300 uživatelů (Z pohledu služby Control)



Graf 15: Poslední test WebClient modulu pro 300 uživatelů – optimalizovaný (Z pohledu služby Control)

Jak lze pozorovat výše, prvotní odhad byl oproti produkčním serverům nepřesný a optimalizace velmi pomohla k přesnosti simulace. Pro službu GW se jednalo o obdobnou situaci, s tím rozdílem že při prvních testech byl rozdíl ještě znatelnější. V případě že funkce na dobu nečinnosti ve scénářích vůbec obsažena

nebyla, server kompletně přestal reagovat po spuštění již 20–30 simulovaných uživatelů (využití CPU služby Control bylo v řádu stovek procent).

	1. Test	2. Test	3. Test	4. Test	5. Test	6. Test
DTW	298.18	105.49	51.08	21.16	15.65	11.39
MAE	15.48	6.31	4.22	2.47	1.43	1.19
MAPE	2.17	0.89	0.60	0.36	0.21	0.16
AVG Prod	7.48	7.48	7.48	7.48	7.48	7.48
AVG Sim	22.96	13.79	11.7	9.89	8.43	7.85

Tabulka 6: Porovnání klíčových ukazatelů v řadě testů s 300 uživateli modulu WebClient

V případě protokolu IMAP, který byl jako jediný optimalizován také počtem spojení se došlo k závěru, že bude lepší hodnoty intervalu ještě snížit na výsledných `[25, 170]`.

5 Aplikace nástrojů – návrh postupu optimalizace hardwaru

Nyní lze veškeré moduly považovat za optimalizované a schopné spolehlivě simulovat požadovaný počet uživatelů. Díky tomuto faktu je nyní možné navrhnout postup, jak na základě simulace chování uživatelů zlepšit kvalitu návrhu a jaké je praktické využití tohoto nástroje.

Z hlediska postupu aplikace nástrojů je nutné testovací a monitorovací prostředí, které již bylo vytvořené v kapitole 4. Je důležité zmínit, že tento krok je také součástí postupu pro optimalizaci hardwaru (dále HW), pouze bylo nutné prostředí zajistit i pro samotnou optimalizaci simulačních modulů, tudíž tato část je v této kapitole vynechána.

Jakmile je prostředí připravené, je možné začít veškeré optimalizované moduly společně aplikovat. Otázky, na které firma IceWarp neznala odpověď, lze nyní pomalu začít zodpovídat:

- Je možné produkčnímu serveru X snížit alokaci jader procesoru na polovinu?
- Je možné udělat obdobnou operaci i s pamětí RAM?
- A jak je to z hlediska připojeného NFS, které je využíváno jako úložiště aplikace?
- Jak se chová aplikace IceWarp na 2, 4 nebo 8 jádrech při stejném počtu aktivních uživatelů?
- Měli bychom pro 850 licencovaných uživatelů zvolit spíše 2, 4, 6 nebo 8 jader procesoru? A kolik GB paměti RAM?
- Velmi vytížený produkční server vyžaduje další alokaci prostředků, bude stačit zvýšit počet jader procesoru z 6 na 8 nebo bude potřeba 12 jader?
- Jaká je korelace mezi CPU utilizací hostujícího serveru a odezvou aplikace IceWarp?
- Jaké jsou v aktuální chvíli odezvy na obvyklé operace uživatelů při simulaci jednotek uživatelů na produkčním serveru X?

Postup optimalizace tedy začíná volbou jedné z těchto nezodpovězených otázek, jelikož aplikace nástroje se liší dle zodpovídané otázky. Pro účely této práce bude zodpovězena otázka, jak se chová aplikace IceWarp na 2, 4 nebo 8 jádrech při stejném počtu aktivních uživatelů. Vedlejším produktem zodpovězení této otázky bude také odpověď na otázku, kolik jader procesoru je třeba pro 850 licencovaných uživatelů, jelikož to bude počet uživatelů, které budeme v rámci testování využívat. Respektive počet uživatelů bude roven počtu licencovaných uživatelů vynásobených koeficientem aktivních uživatelů, který byl stanoven na 0.35, což znamená, že testy budou prováděny s 298 aktivními uživateli.

Je důležité zmínit, že bohužel v aktuálních podmínkách není možné zjistit z produkčních serverů, kolik uživatelů se vyskytuje v rámci jednotlivých klientů

(Outlook, Webový, Desktopový apod.). Z tohoto důvodu bude rozložení těchto 298 aktivnímu uživateli napříč jednotlivými klienty odhadnuto následovně:

- 150 IceWarp WebClient
- 13 SMTP generických
- 25 IMAP generických
- 30 ActiveSync
- 50 IWDC
- 30 Outlook

Po spuštění prvních testů vyšlo najevo, že aktuální způsob monitoringu jednotlivých služeb aplikace IceWarp je nevhodně navržený. Na základě těchto testů došlo tedy ke změně, kdy při monitorování jednotlivých služeb bylo potřeba upravit data sbíraná Zabbixem pomocí *proc.cpu.util* tak, aby odpovídaly sbírané statistiky průměrnému zatížení napříč všemi jádry. Aktuální statistiky totiž toto zohledňovaly pouze u celkového vytížení serveru, ovšem v případě monitorování jednotlivých služeb aplikace se jednalo o součet vytížení všech jader procesu. Zároveň díky této změně bylo možné určit kolik procent z celkového vytížení serveru tvoří právě aplikace samotná nebo její jednotlivé části.

5.1 Popis postupu testu

Test začíná stažením repozitáře projektu a editací veškerých hodnot konfiguračního souboru aplikace ve složce Controller. Poté je možné spustit samotný *usersimulator.sh*, který zajišťuje nejvyšší kontrolní vrstvu, jak bylo popsáno v části implementační. Dle požadovaného testu jsou zvoleny parametry skriptu, v případě výše zmíněného testu by tedy spuštění vypadalo následovně:

```
./usersimulator.sh "150w" "13s" "25i" "30a" "50e" "30o"
```

Skript v tuto chvíli začne informovat o postupném spouštění jednotlivých wrapperů a jednotlivých modulů. Jakmile je spuštění ukončeno, je stav simulátoru považován za stabilní.

Ve chvíli, kdy je simulátor stabilní, je nutné průběžně sledovat statistiky o všech sledovaných atributech v monitorovacím systému Zabbix. Tyto statistiky je také možné díky programům ve složce *Analyse_correlation* lokálně stáhnout přímo z databáze Zabbix serveru a poté porovnávat s produkčními servery, jako bylo prováděno ve fázi optimalizace. Mezi hlavní sledované statistiky by mělo patřit:

- Průměrné vytížení jednoho jádra CPU pro jednotlivé služby
- Průměrné vytížení jednoho jádra CPU pro všechny služby aplikace IceWarp v součtu
- Celková utilizace/doba nečinnosti procesoru
- Využívaná paměť RAM jednotlivými službami
- Statistiky o systému NFS (volitelné)

Po ukončení testu je možné provést analýzu odezvy jednotlivých modulů na individuální operace vykonávané simulátory. Tato analýza musí být provedená z logů, kde moduly každou provedenou operaci zaznamenávají i s odezvou serveru. Tato analýza bude pro účely práce vynechána. Server je považován za stabilní v případě, že hodnota CPU utilizace nepřesáhne 80-85 %. Dle dostupných dat je optimální CPU utilizace v datacentrech mezi 40 a 80 % s ohledem na výkon a životnost HW. Hodnota 80-85 % by měla být cílová v nejvíce vytížených hodinách a překračována jen pro krátké časové úseky. [51; 52]

Využití dat ze Zabbix databáze se liší dle zodpovídané otázky. Pokud otázkou bylo chování aplikace na různém počtu jader, je třeba testy opakovat na různém HW a vhodně porovnat výsledky sledovaných parametrů přímo v GUI Zabbix Serveru nebo lokálně po jejich stažení.

V případě otázky, jakou specifikaci HW zvolit pro 850 uživatelů, je třeba sledovat právě CPU utilizaci společně s průměrným vytížením jednoho jádra CPU pro všechny služby aplikace a volitelně i ostatní sledované metriky. Poté je třeba určit hodnotu, na které již server nebude považován za stabilní nebo provést výše zmíněnou analýzu logů a sledovat nárůst odezvy během nárůstu vytížení CPU aplikací. Tím by se sledovaná hodnota utilizace CPU orientovala také podle odezvy samotného serveru na různé operace.

Pokud by údaje o rozložení uživatelů napříč klienty byly dostupné, výsledná přesnost simulace a její předchozí optimalizace vůči produkci by byla mnohem vyšší. Firma IceWarp do budoucna přislíbila zajištění těchto statistik, což umožní efektivnější využití tohoto postupu. Díky těmto statistikám bude také možné určovat servery, které mají nestandardní výchylky v aktivitě vůči počtu licencovaných uživatelů, a tomu přizpůsobit případný test zvýšením počtu simulovaných uživatelů. Aktuálně toto zvýšení je možné aplikovat, ovšem je k tomu nutné rozložení uživatelů odhadnout a sledovat, zda vytížení procesu na produkčním a simulovaném serveru jsou napříč službami aplikace podobné. Tím se simulátor přizpůsobí specifikám, které jsou s daným serverem spojeny.

Toto zvýšení počtu simulovaných uživatelů je třeba aplikovat například ve chvíli, kdy je zodpovídanou otázkou kolik jader procesoru přidat pro velmi vytížený specifikovaný produkční server.

Díky zodpovězení těchto a dalších otázek je možné na hostovaném HW ušetřit významné finanční a lidské prostředky nebo případně předejít nespokojenosti zákazníků s nespolehlivými odezvami serveru.

6 Porovnání výsledků z již realizovaných projektů

K dispozici je mnoho produkčních serverů, u kterých je podezření, že jsou zbytečně předimenzované nebo naopak se jedná o servery, kde přidělený HW je nedostatečný. Tato kapitola nabízí porovnání simulovaných serverů a těch produkčních a návrh optimalizace některých z nich.

6.1 Aplikace postupu na 2/4/6/8 jádrech CPU s 298 aktivními uživateli

Při spuštění simulátoru s 298 aktivními uživateli nejevil systém s 16 GB RAM paměti žádné zásadní potíže ani s nejnižším počtem 2 jader. Jednalo se o spuštění simulátoru následovně:

```
./usersimulator.sh "150w" "13s" "25i" "30a" "50e" "30o"
```

Po ukončení hodinových testů na všech čtyřech hardwarových specifikacích vyšlo najevo, že pokud by se jednalo o průměrný produkční server s průměrným chováním uživatelů, tak by optimální specifikací byla 2 jádra CPU, která dosáhla průměrné CPU utilizace 42 %. Co se týče samotné paměti RAM, tak po více následujících testech vyšlo jako dostatečných 4-6 GB. Ačkoliv se dle testů může zdát 4 GB dostatečných, aplikace občas trpí na neoptimalizované čištění paměti RAM službou IMAP a při několikadenním běhu simulátoru by se ukázalo, že se paměť postupně plní (při přihlašování a odhlašování uživatelů). Zároveň i po doběhnutí testů, tak se paměť RAM nevyčistí, ale zůstane stejně využita i po ukončení testů. Z toho důvodu vychází lépe zvýšit paměť RAM na 6 GB, ačkoliv 4 GB by v krajních případech mohly být dostatečné.

Je důležité zmínit, že se jedná o testy průměrných uživatelů. Jsou produkční servery, které jsou svým využitím velmi specifické, a počet jader i paměti RAM by pro ně nebyl dostatečný. Z toho důvodu je vždy důležité přizpůsobit testy k daným serverům. Následující tabulka shrnuje průměrné hodnoty sledovaných parametrů napříč testy.

	C8M16	C6M16	C4M16	C2M16	C2M6	C2M4
CPU utilization	11.3 %	14.9 %	21.3 %	34.9 %	37.9 %	41.9 %
RAM utilization	10.9 %	10.8 %	10.7 %	11.2 %	34.1 %	49.2 %
IW CPU	5.5 %	7.4 %	10.1 %	17.9 %	19.1 %	21.2 %
CPU IO wait	0.6 %	0.9 %	1.1 %	1.8 %	1.9 %	1.9 %

Tabulka 7: Průměrné sledované hodnoty pro jednotlivé HW specifikace během hodinového testu (po inicializační fázi)

Z tabulky výše je zřejmé, že se zdvojnásobením počtu jader nepřichází přímo dvakrát takový výkon, ale koeficient zvýšení výkonu je nižší (1.6–1.9x). Zároveň lze

pozorovat, že při čtyřnásobném snížení paměti RAM měla tato změna vliv i na výkon CPU.

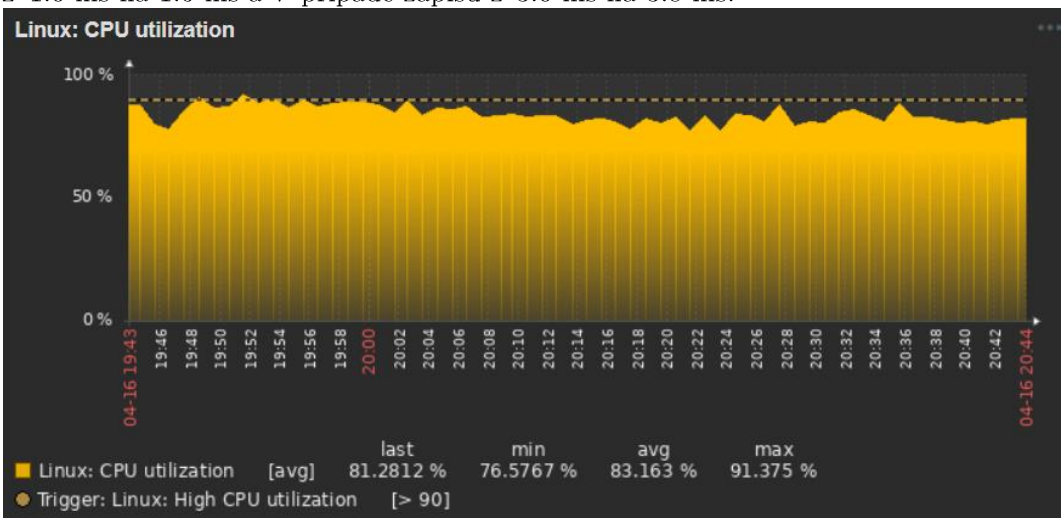
6.1.1. Ověření maximálního počtu uživatelů pro výslednou konfiguraci

Když pro server s 850 licencovanými uživateli je možné na základě předchozí kapitoly tvrdit, že pro průměrné uživatelské chování je vhodná konfigurace 2 jader CPU a 6 GB RAM, je vhodné také ověřit horní hranici počtu aktivních uživatelů, kterou server pojme. Cílem je tedy vytvořit takový test, který bude obdobnou distribucí napříč klienty vytěžovat CPU během hodinového testu v průměru mezi 80–85 % a výše se dostane pouze na krátké časové úseky.

Po sérii testů vyšlo najevo, že i při vytížení serveru všemi 850 licencovanými uživateli, by server byl stále považován za spolehlivý, jelikož průměrná hodnota CPU utilizace se pohybovala kolem 83 %. Tudíž se potvrdilo, že zvolená konfigurace s 2 jádry je ideální v případě průměrného chování uživatelů. Výsledný test, který byl stále server schopen ustát spolehlivě byl následovný:

```
./usersimulator.sh "300w" "50s" "90i" "100a" "170e" "140o"
```

Hodnoty utilizace paměti RAM se také pohybovaly v akceptovatelném rozmezí 50–55 %. Z hlediska CPU IO wait se jednalo o zvýšení z 1.8 % na 3.0 %. Ke statistikám o vhodném vytížení NFS serveru nebyl proveden žádný průzkum, ovšem i přes to není nárůst odezvy považován za markantní. V případě hlavní sledované statistiky o době kompletace požadavku se doba zvýšila v případě čtení z 1.0 ms na 1.6 ms a v případě zápisu z 3.0 ms na 3.8 ms.



Graf 16: Zabbix monitoring – hodnota CPU utilizace pro test s koeficientem aktivních uživatelů roven 1

	298 Active – C2M6	850 Active – C2M6
WC všechny požadavky	0.2302 s	0.2653 s
SMTP send e-mail	0.6313 s	0.7032 s
IMAP Fetch	0.1201 s	0.1797 s
WD REPORT resource	0.1597 s	0.1842 s
EAS Delete/Create item	3.3396 s	3.7140 s

Tabulka 8: Analýza průměrné odezvy vybraných operací jednotlivých modulů při rozdílném počtu aktivních uživatelů v 1 h testu (desítky tisíc požadavků)

6.2 Ověření výkonnosti pro 1 jádro CPU

Cílem této kapitoly je zjistit, zda je možné provozovat IceWarp pro jednotky uživatelů také na jednom jádře při 4 GB RAM. V tuto chvíli nově vytvářené servery s jednotkami uživatelů, začínají na 2-4 jádrech CPU. Testy budou prováděny následovné:

- T1 – běh aplikace IceWarp bez zátěže uživatelů
- T2 – 10 WebClient uživatelů (se zvýšenou aktivitou)
- T3 – 10 WebClient + 5 IWDC uživatelů + 3 Outlook

	T1	T2	T3
CPU utilization	26.4 %	33.1 %	37.5 %
RAM utilization	34.3 %	45.2 %	45.2 %
IW CPU	4.0 %	6.1 %	7.5 %
CPU IO wait	0.05 %	0.13 %	0.45 %

Tabulka 9: Průměrné sledované hodnoty pro jednotlivé automatické testy jednoho jádra

Při testech s malým počtem uživatelů hraje mnohem větší roli specifičnost chování uživatelů, kterou bez telemetrie nelze určit. Z toho důvodu byly pro tyto testy zvýšené aktivity uživatelů zhruba o 100–170 %. Výhodou těchto testů pro minimum uživatelů je, že v případě pochybností, je možné jejich výsledky i ručně ověřit a přizpůsobit maximálnímu možnému vytížení uživatelského rozhraní klientů.

V případě testů na jednom jádře je nutné považovat výsledky pouze za orientační a výsledné hodnoty nebrat jako jediný zdroj pravdy. Cílem bylo především zjistit, zda aplikace je schopna chovat se spolehlivě na jednom jádře a při tom odbavit malý počet uživatelů. Vzhledem k výkyvům chování uživatelů, které jsou v případě jednoho jádra mnohem více znatelné, byl po sadě ručních testů v kombinaci s těmi automatizovanými stanoven maximální počet licencovaných uživatelů pro jedno jádro na 10–15. V budoucnu se po týdnech monitorování nabízí prostor na zpřesnění a potenciální zvýšení tohoto limitu.

6.3 Vytipování problémových produkčních serverů

V tuto chvíli je již známé, jak se změna HW projeví na výkonu serveru. Je tedy třeba zjistit, jaké servery se jeví jako neoptimalizované. K tomu byla využita specificky modifikovaná verze programu popsaného v sekci 4.5.2.

Program zajistí z databáze Zabbix serveru pro každý produkční server šest monitorovaných položek a všechny jejich hodnoty za poslední týden:

- Počet licencovaných uživatelů
- Celková paměť RAM
- RAM utilizace
- CPU počet jader
- CPU utilizace
- IceWarp utilizace CPU

Nalezne nejvytíženější sekvenci o specifikované délce CPU utilizace a následně vyhledá v ostatních pěti položkových souborech, jaké byly jejich statistiky ve stejném časovém intervalu. Výsledkem programu je .csv soubor, který shrnuje minimální/průměrné/maximální vytížení CPU utilizace v této nejvytíženější sekvenci, celkové statistiky během jednoho týdne (RAM i CPU), chování RAM paměti během této nejvytíženější sekvence, počet uživatelů, jader, paměti apod.

Hlavní sledovanou hodnotou je utilizace CPU. Pokud server v posledním týdnu jevil známky přetížení, kdy průměrná hodnota CPU utilizace v nejvytíženější moment překračovala 85 %, tak je třeba se zamyslet nad přidáním jader CPU. Pokud se naopak pro nějaký server ani v nejvytíženějších hodinách CPU utilizace nepřibližovala hranici 85 %, ale pohybovala se v řádu nižších desítek, tak je na zvážení, jestli serveru jádra CPU neubrat. Obdobný způsob lze aplikovat na paměť RAM. Ostatní statistiky v souboru slouží jako pomocné.

Velice podobným způsobem, byly také zajištěny statistiky pro výpočet průměrného využití úložiště jedním uživatelem napříč všemi cloudovými datacentry. Tato hodnota bude využita při zhodnocení ekonomických dopadů této práce.

6.4 Aplikace optimalizace na produkční server

Simulace již byly provedeny a problémové servery vytipovány. Pro ukázku aplikace optimalizace byly vybrány tři produkční servery. Dva ze serverů mají přidělených příliš mnoho zdrojů a třetí naopak vyžaduje více zdrojů, než má alokováno. Do budoucna bude optimalizace nasazena postupně napříč cloudovými datacentry. Níže je tabulka, která shrnuje výsledky z předchozí kapitoly, ze kterých vyplývá, jaké změny bude třeba v rámci cloudu vytvářet. V tuto chvíli pouze s ohledem na CPU, ačkoliv do budoucna bude vhodná optimalizace i z hlediska

RAM. To ovšem může nastat až po optimalizaci CPU, jelikož je mezi změnou CPU a RAM korelace, která byla nastíněna v předchozích kapitolách.

	Příliš málo zdrojů	Optimálně zdrojů	Příliš mnoho zdrojů
Zastoupení	0.6 %	3.0 %	96.4 %

Tabulka 10: Procentuální zastoupení serverů s nedostatečnou/optimální/přebytečnou konfigurací (Vyhodnoceno dle CPU)

U zákazníků on-premise bude optimalizace nasazena obdobným způsobem, kdy po ověření výsledků optimalizace na cloudových datacentrech bude pro nové zákazníky doporučována nová HW specifikace. Níže jsou dostupné tabulky porovnání klíčových ukazatelů tří optimalizovaných serverů před a po optimalizaci. Délka dat využitých pro evaluaci je jeden týden před optimalizací a jeden týden po optimalizaci. V ideálním případě by bylo vhodné mít tato data spíše v rozmezí celého měsíce, ovšem z důvodů nedostatečné kapacity Zabbix serveru, z jehož databáze jsou data extrahována, je délka doby ukládaných dat nastavena na 7 dní.

	PS 1	PS 2	PS 3
Výchozí počet jader	4	4	2
Počet uživatelů	62	8	161
Průměrná CPU utilizace	6.94 %	2.92 %	40.64 %
Průměrná CPU utilizace v peaku	16.37 %	4.15 %	99.80 %
IW CPU utilizace v peaku	6.00 %	0.52 %	86.11 %
RAM utilizace v peaku	85.28 %	76.24 %	32.66 %

Tabulka 11: Sledované hodnoty před optimalizací (1 týden)

	PS 1	PS 2	PS 3
Výsledný počet jader	2	1	4
Počet uživatelů	62	8	161
Průměrná CPU utilizace	22.39 %	12.06 %	17.01 %
Průměrná CPU utilizace v peaku	65.67 %	14.84 %	45.87 %
IW CPU utilizace v peaku	9.46 %	2.2 %	32.3 %
RAM utilizace v peaku	67.80 %	66.46 %	55.85 %

Tabulka 12: Sledované hodnoty po optimalizaci (1 týden)

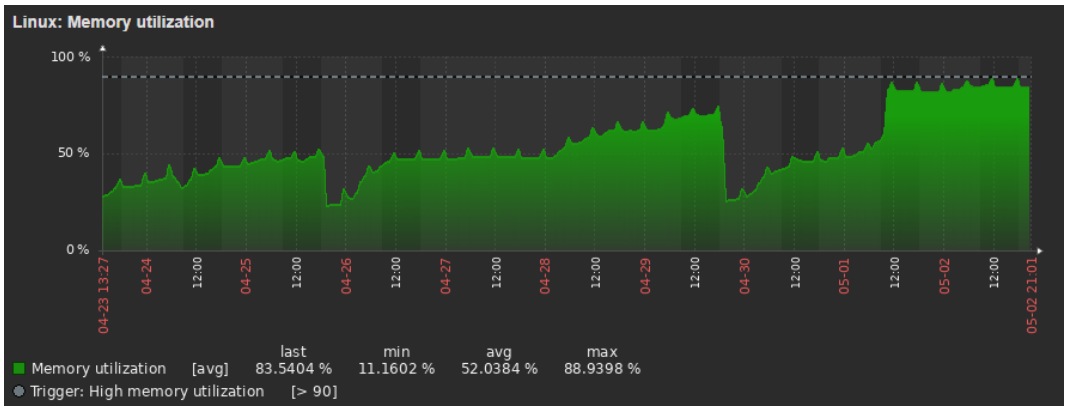
Z tabulky výše plyne, že v případě produkčního serveru 1 (PS1) optimalizace měla za důsledek velký nárůst CPU utilizace, který byl o dost větší, než by dle předchozích testů bylo očekávané. Zda k tomuto nárůstu došlo zvýšenou aktivitou uživatelů nelze bez telemetrických dat určit. Důležité je, že hodnota průměrné CPU utilizace ve sledovaném dvouhodinovém peaku nepřesáhla hodnotu 90 %, ale zároveň se drží nad 50 %. Optimalizace tedy v tomto případě byla úspěšná a snížení na pouhé jedno jádro by vedlo k tomu, že by průměrná utilizace v peaku teoreticky přesáhla 100 % a server by tedy nebyl nadále responzivní.

V případě produkčního serveru 2 (PS2) by se mohlo zdát, že jsou zdroje stále nevyužité. Je tomu doopravdy tak, ovšem méně než jedno jádro již serverům přidělit nelze. Zde je názorná ukázka, že v rámci optimalizace je cílem se zaměřit

hlavně na menší servery, které mají mnohokrát více nebo případně stejně zdrojů jako servery s desetinásobným počtem uživatelů. Právě u těchto nejmenších serverů, kterých je nejvíce, tak bude úspora největší.

Produkční server 3 (PS3) vyžadoval navýšení počtu jader, ačkoliv dle předchozích testů by měl být výchozí počet jader dostatečný. Záměrně byl vybrán server, který z průměrného využití serveru uživateli nejvíce vybočuje a na poměr počtu uživatelů vytváří větší nežli očekávanou zátěž. Zde optimalizace přispěla k tomu, aby se ve chvíli největšího náporu aplikace nestala neresponzivní, a tedy nepřesahovalo vytížení CPU hranici 90 %, ale zároveň se nepohybovala tato hodnota v nižších desítkách procent. Na datech v tabulce lze opět pozorovat, že sledovaný týdenní úsek není dostatečný a v týdnu, kdy optimalizace nastala, lze z grafů počtu spojení na různých službách aplikace sledovat pokles oproti týdnu před optimalizací. Tyto grafy nejsou v práci přítomny, ovšem jsou také součástí monitorovacího systému, kde byl tento výkyv zaznamenán. Proto by se mohlo jevit, že zvýšení jader na dvojnásobek vyvolalo více než dvojnásobné zvýšení výkonu, což není možné. Optimalizace v každém případě přispěla zlepšení výkonu serveru a zlepšení odezvy v době největšího vytížení.

V případě všech optimalizovaných serverů zůstala nastavená hodnota paměti RAM stejná. Zde je možné pozorovat fakt, že aplikace po sobě správně neuvolňuje paměť a proto zde není možné hledat jakoukoliv korelaci mezi počtem uživatelů a využitím paměti RAM. Toto lze také pozorovat na grafu utilizace paměti RAM pro produkční server 3 níže. Graf utilizace paměti je téměř vždy rostoucí, dokud nedojde k prudkému poklesu, který je obvykle způsoben výpadkem služby nebo případným cíleným restartem. Z toho důvodu je také paměť RAM z optimalizace vynechána a její budoucí optimalizace je dále rozebrána v následující kapitole.



Graf 17: Utilizace paměti RAM a její nevhodné uvolňování

7 Zhodnocení ekonomických a procesních dopadů navrhovaných změn

Cílem této kapitoly je čtenáři přiblížit ekonomické dopady práce a specifikovat nově vzniklé, případně optimalizované procesy ve firmě IceWarp. Mezi ty nejlépe vyčíslitelné dopady může patřit tzv. TCO (Total Cost of Ownership), který určuje celkové náklady spojené s hostováním IceWarp aplikace a samotným provozem. Výsledky této práce také napomáhají tomu, aby žádné projekty v budoucnu neskončily selháním z důvodu podhodnocení HW specifikace projektu. Špatně navržený HW obvykle vede k pomalejší odezvě serverů, s čímž se pojí nespokojenost zákazníků a potenciální ztráta příjmů v řádech milionů korun. Tyto dopady jednoduše vyčíslit nelze, ale jedná se o prevenci před ztrátou příjmů firmy a potenciální ztráty důvěry od zákazníků.

7.1 Časová náročnost řešení / Cena řešení

Časová náročnost navrhovaného řešení pro simulaci uživatelské aktivity za účelem optimalizace hardwaru mailového kolaboračního řešení je vyčíslena počtem hodin potřebných pro tvorbu řešení v ideálním případě, kdy by firma IceWarp dodala podklady, které byly přislíbeny. V případě této práce se vzhledem k nedostatku podkladů časová náročnost řešení zvýšila oproti ideálnímu případu zhruba o 20 %. Výpočet ceny řešení také předpokládá, že zhotovitel má hlubokou znalost využívaných protokolů a jeho povinností je pouze naučit se specifika využití protokolů v aplikaci IceWarp a práci s jednotlivými klienty aplikace. Dalším předpokladem je existence jednoho jednoduchého testovacího serveru již před začátkem projektu (nikoliv celého simulačního prostředí).

	Počet hodin
U1	50–80
U2	30–50
U3	80–130
U4	90–110
U5	350–550
U6	24–40
U7	60–100
U8	40–60
U9	24–60
Součet	748–1180 hodin

Tabulka 13: Časová náročnost řešení v ideálním případě

Popis jednotlivých činností, jejichž časová náročnost je popsána v tabulce výše:

- **U1**
  - Analýza aplikace IceWarp včetně seznámení se s jednotlivými klienty
- **U2**
  - Analýza stávajícího řešení, průzkum existujících nástrojů na tvorbu syntetických dat, výkonnostního benchmarkingu a simulace uživatelů
- **U3**
  - Analýza dat reálných uživatelů a tvorba syntetických dat se stejnými vlastnostmi
- **U4**
  - Sběr požadavků pro simulátory uživatelských aktivit, analýza logů jednotlivých protokolů pro zjištění specifik jejich využití v aplikaci IceWarp, volba programovacích jazyků, návrh architektury
- **U5**
  - Tvorba implementace pro simulátory uživatelských aktivit, realizace návrhu (optimalizace modulů, tak jak byla vykonána v této práci, není součástí U5, jelikož je předpokladem, že frekvence jednotlivých operací byla dodána firmou a řešitel pouze ověřuje, že frekvence v jeho simulátoru je ekvivalentní – U6)
- **U6**
  - Ověření implementace s telemetrickými daty, která zajišťují informace o frekvenci jednotlivých operací, vytížení napříč jednotlivými klienty apod.
- **U7**
  - Tvorba simulačního prostředí (IceWarp servery, JMeter servery, Zabbix servery, tvorba uživatelů, návrh a realizace importování dat)
- **U8**
  - Aplikace výsledného nástroje v simulačním prostředí a zodpovězení kladených otázek, porovnání s produkčními servery, doporučení na optimalizaci
- **U9**
  - Tvorba dokumentace

Pro vše z výše zmíněného platí, že je nutné, aby implementace byla jednoduše opakovatelná a co nejvíce automatizovaná. To například znamená, že součástí U7 je také tvorba nástrojů, které automatizovaně uživatele o požadované distribuci založí, vytvoří schránky o požadované velikosti apod.

Výsledná časová náročnost projektu se tedy pohybuje někde mezi 748–1180 hodinami v ideálním případě. Pokud by se vzala v potaz průměrná hodinová mzda dle portálu prumerneplaty.cz na pozici IT Specialista, tedy 320 Kč, poté by se dal projekt nacenit na 240–378 tisíc Kč. [53]

7.2 Vliv optimalizace na TCO

TCO neboli celkové náklady spojené s vlastnictvím je finanční metrika, která měří celkové náklady na pořízení, provoz a údržbu aktiva po dobu jeho životnosti. V tomto případě je aktivem aplikace IceWarp a vše potřebné pro její provoz. Zahrnuje přímé i nepřímé náklady. [54] Důležité parametry vstupující do výpočtu v případě jakékoliv aplikace hostované v cloudovém datacentru jsou:

1. Poplatky účtované poskytovatelem cloudu
2. Náklady na aktualizaci OS
3. Náklady na aktualizaci aplikace
4. Náklady na podporu uživatelů systému
5. Náklady na vývoj aplikace
6. Cena za umístění serveru do datacentra (pouze za místo)
7. Cena za 1 kWh elektřiny
8. Cena za 1 jádro CPU
9. Cena za 1 GB paměti RAM
10. Cena za 1 GB SSD
11. Cena za 1 GB HDD
12. Životnost jednotlivých HW komponent
13. Spotřeba jednotlivých HW komponent

Pro účely výpočtu TCO v této práci bude většina z bodů zanedbána. Cílem této kapitoly je totiž především ukázat množství ušetřených prostředků díky samotné optimalizaci, a nikoliv vypočítat přesné náklady, které firma musí při provozování aplikace uvažovat.

Bod 1. je v případě firmy IceWarp zanedbatelný, jelikož si své datacentra spravuje firma sama a nemá tedy žádného poskytovatele jako je například AWS. Body 2.–6. jsou, co se týče vlivu optimalizace na výpočet TCO neměnné. Hodnoty, které se díky výsledkům této práce mění, jsou body 7. – 13. Respektive tato práce zajišťuje snížení potřebného hardwaru průměrně pro jednoho licencovaného uživatele, což zajišťuje například menší množství spotřebované elektřiny.

Položka	Cena
Umístění 1 serverové skříně	Od 8550 Kč / měsíc [55]
1 kWh elektřiny	7.82 Kč [55]
1 jádro CPU	974 Kč
1 GB paměti RAM	137 Kč
1 GB SSD	9.49 Kč
1 GB HDD	1.02 Kč
Průměrná doba životnosti HW	4–6 let
Průměrná maximální spotřeba 1 jádra CPU	10 W

Tabulka 14: Přehled cen jednotlivých parametrů vstupujících do výpočtu TCO [55]

Výše zmíněné hodnoty vychází z reálných hardwarových specifikací serverů firmy v datacentrech při průzkumu aktuálních cen na trhu od společnosti ANAFRA s.r.o. [56] Firma IceWarp si nepřála zveřejňovat konkrétní HW specifikace serverů ani způsob virtualizace. Z toho důvodu bude ve výpočtu uvažován monolitický hardwarový přístup, kdy každý ze serverů má pevně dedikovaný počet jader, a tedy nelze v mezičase nevyužívání jader CPU tyto prostředky přidělit jiným serverům. Tato situace je v případě cloudu nereálná a v dnešní době se již nepoužívá, ovšem v případě mnoha on-premise zákazníků je stále aktuální.

V tuto chvíli je přiděleno v součtu na cloudu 2572 jader veškerým produkčním serverům. Jak vychází ze statistik v kapitole 6.4., v případě 0.6 % serverů, které mají nedostatek CPU se jedná o navýšení celkem 14 jader. V případě 96.4 % serverů, které mají příliš mnoho zdrojů, tak se jedná v průměru o snížení počtu jader o polovinu, tedy 50 % úspora počtu jader. Tento průměrný případ vychází z toho, že téměř žádný z těchto serverů i v nejvytíženějších hodinách nepřesahoval hodnotu 50 % CPU utilizace. Součet počtu jader přidělených těmto serverům je rovných 2500 jader. Pokud by se tedy z těchto 2500 jader CPU povedlo ušetřit polovinu, jedná se o úsporu 1250 jader. Od toho je třeba odečíst jádra, která bylo třeba v rámci optimalizace přidat, tedy výsledná úspora je rovna 1236 jádrům CPU. Díky této optimalizaci se jedná pouze z hlediska CPU o úsporu 1 204 000 Kč. Reálná úspora díky virtualizaci bude několikrát menší, ovšem konkrétní úspora velmi závisí na způsobu virtualizace, která není známa. Co se týče spotřeby energie, tak během jednoho měsíce by v případě 1250 ušetřených jader proběhla úspora energie 9300 kWh. Taková úspora je ekvivalentní 72 726 Kč. Pokud vynásobíme tuto částku střední hodnotou životnosti komponent, tedy 5 let, jedná se o 363 630 Kč.

Během spouštění testů simulovaných uživatelů se také podařilo zjistit, že aplikace IceWarp, především služba IMAP, během ukončení relace uživatele nevyprázdní data z paměti RAM, a paměť tedy zůstává nadále zaplněna nepotřebnými daty z neexistující relace uživatele. Na základě této skutečnosti se firma rozhodla přejít na systém kernelového profilování a dalších změn v implementaci. Tyto změny jsou již v kompetenci seniorních zaměstnanců firmy. Tato změna podložená daty ze simulací by do budoucna měla zajistit úsporu paměti RAM o 30–40 % dle prvotního testování. Aktuálně veškeré cloudové servery mají v součtu přidělených 5940.6 GB paměti RAM. V případě dokončení této optimalizace se tedy bude jednat o úsporu 1782–2376 GB RAM pouze z této prvotní optimalizace. To je ekvivalentní úspoře 244–326 tisíc Kč. Jakmile bude dosaženo tohoto stavu, je možné znovu využít vytvořené nástroje se zaměřením na optimalizaci RAM a zajistit tak ještě větší úsporu peněžních prostředků.

Posledním parametrem, jehož hodnotu pomohla tato práce zjistit, je průměrné místo využité jedním licencovaným uživatelem na cloudu. To vychází v průměru na 4343 MB pro jednoho uživatele. Tedy cena místa za jednoho licencovaného uživatele je v průměru 4.42/41.21 Kč dle použitého typu úložiště. Tuto hodnotu nelze příliš optimalizovat, jelikož plánování místa pro jednotlivé servery je tvořeno pomocí NFS kvótování, které pracuje s velikostí jako se sdíleným fondem zdrojů. Tato metrika bude pro firmu prospěšná pro kompletní výpočet TCO,

ovšem v rámci výpočtu úspory díky tomuto řešení není podstatná. Do budoucna by měla posloužit také pro změnu marketingového plánu firmy, který v aktuální chvíli slibuje pro každého uživatele až stovky GB dat, které by samy o sobě přesahovaly cenu pořízené licence.

Pokud se tedy sečte celková úspora v případě monolitického návrhu hardwaru napříč všemi datacentry, jedná se o úsporu 1 852 500 Kč ve středním případě, kdy by úspora kernelovým profilováním byla rovna 35 %. Tato úspora je vypočtena na jeden životní cyklus komponent, který ve středním případě je stanoven na 5 let. Průměrná měsíční úspora se tedy rovná 30 877 Kč, což znamená, že se investice firmy do tohoto řešení může vrátit již v rozmezí 8 až 13 měsíců.

Tato celková úspora nezapočítává blíže neurčenou optimalizaci RAM, která nastane pomocí vytvořených nástrojů po implementaci profilování. Co se týče zlepšení životnosti HW, tam žádná zásadní změna nenastane, jelikož problémy s HW byly spíše v jeho nedostatečném využití než v přílišném zatěžování. To by mohlo vést k otázce, zda nenastane vlivem navrhovaných změn snížení životnosti komponent, ale vzhledem k tomu, že optimalizace dbala i na nepřetěžování zdrojů, tak by tato situace nastat neměla.

7.3 Ostatní ekonomické dopady

Dalšími ekonomickými dopady jsou takové, které nelze jednoduše vyjádřit částkou, ovšem jsou přímo ovlivněné výsledky této práce. Tyto dopady souvisí především s prevencí selhání případných budoucích velkých projektů poskytování řešení firmy IceWarp zákazníkům. V případě, že je smluvně garantovaný čas bez výpadku přes 99 % je prostor pro chyby skutečně malý a je nutné mít servery dobře navržené. Procentuální zastoupení doby, kdy aplikace má být stabilní, tak obzvlášť v případě velkých zakázek ve veřejném sektoru, je pevně stanovena ve smlouvách, které poskytovatel služby musí dodržovat. V případě jejich nedodržování mohou být na poskytovatele uplatněny pokuty nebo ukončení spolupráce ze strany zákazníka. V rámci analýzy předchozích projektových řešení autor narazil na nevhodně navržené servery, které by v tuto chvíli při aplikaci znalostí výsledků této práce nemohly být schváleny. V té době nebylo možné v konkrétních případech zákazníka, který se snažil ušetřit na hardwaru, přesvědčit, že daná specifikace je doopravdy nutná, jelikož k tomu neexistovaly jakékoliv podklady. Toto vedlo k poddimenzování HW a nedodržení smluvních podmínek

Dalším ekonomickým dopadem, který aktuálně nelze vyčíslit a úzce souvisí s prevencí před selháváním velkých projektů jsou zákazníci on-premise. Těchto zákazníků je aktuálně ve firmě majoritní část a jedná se jak o malé firmy, tak o firmy s desítkami tisíc uživatelů. Ve všech případech, jakmile zákazník žádá o licenci produktu, tak konzultuje také HW požadavky na aplikaci IceWarp. V tuto chvíli již odpověď na tuto otázku může být v případě jakéhokoliv množství uživatelů jednoduše zodpovězena.

7.4 Procesní dopady

V prvotní fázi si práce kladla za cíl vytvořit takový nástroj, který by poskytl natolik přesná data, aby na základě matematického vzorce bylo možné hardware pro hostující server určit i bez jakékoliv technické znalosti. S pozdějším pochopením komplexnosti problému se ukázalo, že takové řešení není v aktuální chvíli bez telemetrických dat možné, a i v případě, že by taková data byla dostupná, tak by se jednalo o velmi komplikovaný úkol.

I přes tyto skutečnosti, tak autor vytvořil pro počet jader CPU univerzální vzorec, který předpovídá na dané specifikaci budoucí utilizaci CPU, při určitém počtu uživatelů. Cílem tedy je pro daný vzorec najít takové parametry, aby se utilizace CPU (výsledek výpočtu) pohybovala v rozmezí požadovaných hodnot. Tento vzorec není v práci zveřejněn, jelikož je potřeba jeho důkladné ověření i pro testy s větším množstvím jader a uživatelů, čehož nebylo možné s aktuálními prostředky v testovacím prostředí docílit.

Tento vzorec pomůže k optimalizaci procesu volby počtu jader pro hostující servery aplikace IceWarp. Poté, co se statisticky ověří správnost tohoto vzorce, budou moci určovat počet jader i zaměstnanci mimo technická oddělení. Nebude tedy potřeba pro zajištění hardwarové specifikace vytěžovat kapacitu seniorních zaměstnanců firmy, jejichž čas je nejdražší.

7.4.1. Vznik nového procesu

Výsledky práce také přispěly k tvorbě nového procesu. Konkrétně v tuto chvíli je třeba změny v hardwarové specifikaci provést napříč všemi cloudovými datacentry. Jakmile dojde ke změně, je třeba zajistit, aby každý zaměstnanec technického oddělení měl možnost reagovat na případné výkyvy od očekávaného stavu serveru.

Z toho důvodu v aplikaci Zabbix vznikly nové spouštěče, které při třicetiminutovém kontinuálním vytížením CPU serveru nad 90 % vytvoří oznámení. Obdobným způsobem pro paměť RAM. Pokud se za jeden týden vygenerují více než čtyři tato oznámení pro daný server, pak je třeba ze strany týmu podpory o situaci informovat tým infrastruktury, který provede případné přidání počtu jader. Tento proces je naprosto zásadní pro správné fungování optimalizace, jelikož servery, kde jsou odchylky od očekávaného chování uživatelů největší, tak mohou vykazovat známky nedostatečné odezvy serveru. Do budoucna by měl být monitorovací systém ve spolupráci s virtualizačními systémy schopen tyto problémy řešit sám bez přičinění zaměstnanců zajišťující monitoring aplikace.

7.5 Budoucí rozšiřitelnost řešení

Aktuální řešení nabízí také do budoucna rozšiřitelnost o další funkcionality či případné vylepšení stávajícího systému. Prvním zásadním bodem pro zlepšení funkcionality by bylo vhodné nahradit v případě většiny systémů skriptovací jazyk Bash za programovací jazyk Python. Během fáze spouštění testů se ukázalo, že Bash

není vhodný pro aplikace, kde je třeba paralelně vytvářet až tisíce vláken. Je tomu tak z toho důvodu, že Bash neumí pracovat s vlákny, ale je vždy třeba spouštět nový proces, který případně vytváří další podprocesy. Autor o tomto problému věděl již během implementace, ovšem z počátku byl Bash ze strany firmy preferován. Postupně projekt ovšem nabyl větších rozměrů, než bylo očekáváno, a Bash se projevil jako nevhodný. Dále by bylo vhodné vyřadit funkcionalitu JMeter serverů, které by neměl být problém nahradit Pythonem se správnými knihovnami. JMeter se sice ukázal jako spolehlivý, ovšem tvorba změn v samotných scénářích je velmi složitá a dlouhodobě neudržitelná pro účely tohoto projektu.

Základem rozšiřitelnosti aktuálního řešení by mělo být, zapracování telemetrických dat do jednotlivých modulů a ověření statistické správnosti simulace oproti reálným serverům pomocí logové analýzy. Tuto logovou analýzu by do budoucna také bylo prospěšné automatizovat. Z jednotlivých modulů by tato automatizace měla být schopná vytvořit souhrn odezvy protokolu v případě jednotlivých druhů volání. V návaznosti na zapracování telemetrických dat by bylo vhodné zlepšit komunikaci mezi jednotlivými vrstvami implementace, která v případě fatálních chyb běhu testu vykazuje občasné nedokonalosti.

Kontrolní skript by také mohl přijímat parametr, který bude schopný přizpůsobit frekvenci operací ve všech jednotlivých modulech. Díky tomuto parametru, který by musel vycházet právě z telemetrických dat, by bylo možné vždy provést simulaci naprosto ekvivalentní právě sledovanému produkčnímu serveru pro jeden časový úsek. V tomto případě by parametry byly staticky změněny na začátku testu a při běhu již neměnné. Posledním a nejvíce technologicky náročným rozšířením by mohla být funkcionalita, která by pouze na základě předložených několikahodinových dat produkčního serveru, v kombinaci s telemetrickými daty, upravila v reálném čase frekvenci operací jednotlivých modulů, a tedy vytvořila na simulačním serveru stejnou křivku vytížení CPU, RAM a NFS.

Závěr

Hlavním cílem této práce byla optimalizace hardwaru mailového kolaboračního řešení za pomocí analýzy chování uživatelů a následné tvorby simulátoru jejich chování.

Úvodem práce bylo shrnutí nejdůležitějších teoretických pojmů z oblasti protokolů aplikace IceWarp a klientů, které tyto protokoly využívají. Následovala analýza současného řešení a průzkum nástrojů, které mohou pomoci s optimalizací aktuálního stavu. Nástroje, které by pomohly k optimalizaci aktuálního stavu téměř neexistují, a tudíž bylo nutné provést vlastní návrh třívrstvé architektury. Tato architektura umožňuje opakované spouštění testů, nezávislost mezi jednotlivými moduly, které simulují specifický protokol aplikace, a dědění zodpovědnosti mezi jednotlivými vrstvami.

Po implementaci bylo nutné moduly optimalizovat, což vzhledem k absenci telemetrických dat muselo být provedeno matematicky, na základě dat z reálných serverů, přizpůsobení parametrů testů těmto datům a vyhodnocení klíčových ukazatelů. Ve chvíli, kdy byla optimalizace dokončena a jednotlivé moduly tvořily zátěž na cílový server ekvivalentní k reálným uživatelům, bylo možné se přesunout do fáze testování.

V této fázi vzniklo testovací, monitorovací a analyzační prostředí. Během desítek testů byly eliminovány poslední chyby. Výsledky z těchto testů, ať už lokální záznamy z logů nebo data z monitorovacího systému Zabbix, byly využity k samotné optimalizaci hardwaru v produkčním prostředí.

Tato optimalizace byla v rámci práce aplikována pouze na jednotky serverů ze stovek vytipovaných, čímž bylo zajištěno ověření správnosti. Lze přepokládat, že v budoucnosti firma IceWarp optimalizaci postupně nasadí napříč všemi cloudovými datacentry a s nově vzniklými procesy bude možné rychleji reagovat na případné odchylky specifického využití serveru skupinou uživatelů.

Autorovo řešení problému zajistilo firmě lepší porozumění chování uživatelů, nástroj na tvorbu syntetických e-mailových schránek, které odpovídají reálné statistické distribuci, a moduly, které shrnují chování uživatelů v jednotlivých protokolech. V neposlední řadě řešení zajistilo také pozitivní vliv na hodnotu TCO, kterou snížilo až o 31 000 Kč měsíčně v nákladech na provozování cloudových datacenter. Využití tohoto projektu není omezeno pouze na optimalizaci hardwaru, ale do budoucna může sloužit také pro stanovení uživatelských limitů pro velikost odesílaných e-mailů nebo v upravené verzi jako testovací nástroje funkčnosti aplikace. V neposlední řadě se také jedná o nástroj, který může určit, které protokoly je vhodné nahradit za jiné nebo případně jaké operace vykonávané uživateli jsou pro aplikaci ty nejnáročnější.

Výsledný třívrstvý model umožňuje svým během zodpovědět celou řadu otázek, dle přání uživatele, pokud vhodně přizpůsobí parametry testů své otázce. Tato vlastnost je velmi důležitá pro rozporování nepodložených rozhodnutí vrcholného managementu a jeho přesvědčení o nevhodném rozhodnutí na základě dat z testů.

Tato práce přinesla během téměř 1000 hodin, které celé řešení zabralo, nespočet osobních přínosů i pro mě jako autora, a jsem velmi rád, že jsem tento projekt mohl vypracovat sám od začátku až do konce. Ačkoliv se povedlo během více než roku trvání projektu splnit veškeré cíle práce, tak některé volby během tvorby řešení bych zpětně považoval za nevhodné. Jako primární chybu bych zhodnotil upřednostňování skriptovacího jazyka Bash ve spolupráci s nástrojem Apache JMeter před programovacím jazykem Python. Toto rozhodnutí bylo z velké části ovlivněno ze strany zadavatele a já jakožto řešitel neměl dostatečný rozhled pro rozporování tohoto návrhu. Sekundárním aspektem, který negativně ovlivnil výsledek, je absence jakýchkoliv telemetrických dat, které byly z počátku přislíbeny. Absence těchto dat vedla k významnému navýšení náročnosti projektu a zvýšení nepřesnosti výsledků. Do budoucna by firma IceWarp měla tato data začít sbírat, protože díky tomu bude možné postupně řešení dostat do stavu naprosté dokonalosti.

Literatura

[1] ICEWARP TECHNOLOGY S.R.O. *IceWarp, o společnosti* [online]. 2023 [cit. 2023-11-30]. Dostupné z: https://www.icewarp.cz/company/

[2] ICEWARP TECHNOLOGY S.R.O. *This is IceWarp Epos* [online]. 2023 [cit. 2023-11-30]. Dostupné z: https://blog.icewarp.com/this-is-icewarp-epos/

[3] ICEWARP TECHNOLOGY S.R.O. *IceWarp, SmartAttach* [online]. 2023 [cit. 2023-11-30]. Dostupné z: https://www.icewarp.cz/smartattach/

[4] ZAPISOTSKYI, Andriy. *Recommended Maximum Email Size and Proven Ways to Optimize It* [online]. 2019 [cit. 2023-11-30]. Dostupné z: https://mailtrap.io/blog/email-size/

[5] Kompletní kancelář v prohlížeči (obrázek). In: ICEWARP TECHNOLOGY S.R.O. *IceWarp* [online]. 2023 [cit. 2023-11-30]. Dostupné z: https://www.icewarp.cz

[6] ICEWARP TECHNOLOGY S.R.O. *IceWarp, downloads* [online]. 2023 [cit. 2023-11-30]. Dostupné z: https://www.icewarp.cz/downloads/apps/

[7] MICROSOFT. *[MS-ASHTTP]: Exchange ActiveSync: HTTP Protocol* [online]. 2022, 2022-04-29 [cit. 2023-11-30]. Dostupné z: https://learn.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ashttp/fee47e08-b416-46b0-9350-ca9eb5a587da

[8] ICEWARP TECHNOLOGY S.R.O. *IceWarp Mobile* [online]. 2023 [cit. 2023-12-01]. Dostupné z: https://www.icewarp.com/apps/icewarp-mobile/

[9] LEPILKINA, Diana. *Understanding SMTP – The Protocol Behind Email Delivery* [online]. 2024, 2024-01-25 [cit. 2024-03-09]. Dostupné z: https://mailtrap.io/blog/smtp/

[10] IBM. *Simple Mail Transfer Protocol* [online]. 2021, 2021-03-08 [cit. 2024-03-09]. Dostupné z: https://www.ibm.com/docs/en/i/7.1?topic=information-smtp

[11] ACHARYA, Durga Prasad. *What Is IMAP and How Does It Work?* [online]. 2023, 2023-08-04 [cit. 2023-12-01]. Dostupné z: https://geekflare.com/imap-internet-message-access-protoco/

[12] MARKOVA, Vasilena. *What is IMAP?* [online]. 2023, 2023-10-12 [cit. 2023-12-01]. Dostupné z: https://www.cloudns.net/blog/what-is-imap/

[13] ATMAIL. *IMAP Commands* [online]. 2020 [cit. 2023-12-01]. Dostupné z: https://www.atmail.com/blog/imap-commands/

[14] ATMAIL. *IMAP 101: Manual IMAP Sessions* [online]. 2018 [cit. 2023-12-01]. Dostupné z: https://www.atmail.com/blog/imap-101-manual-imap-sessions/

[15] MELNIKOV & CRIDLAND. *RFC 7162, CONDSTORE Extension* [online]. 2014 [cit. 2023-12-02]. Dostupné z: https://www.rfc-editor.org/rfc/rfc7162#section-3.1

[16] MICROSOFT. *RFC 822 Message Format* [online]. 2013 [cit. 2023-12-02].

Dostupné z: https://learn.microsoft.com/en-us/previous-versions/office/developer/exchange-server-2010/aa493918(v=exchg.140)

[17] MELNIKOV & CRIDLAND. *RFC 7162, MODSEQ Message Data Item in FETCH Command* [online]. 2014 [cit. 2023-12-02]. Dostupné z: https://www.rfc-editor.org/rfc/rfc7162#section-3.1.4.2

[18] ICEWARP. *How do I delete old messages in a mailbox using IMAP?* [online]. 2017 [cit. 2023-12-02]. Dostupné z: https://support.icewarp.com/hc/en-us/articles/115003357567-How-do-I-delete-old-messages-in-a-mailbox-using-IMAP-

[19] CRISPIN, Mark. *Internet Message Access Protocol (IMAP) - UIDPLUS extension* [online]. 2005 [cit. 2023-12-02]. Dostupné z: https://www.rfc-editor.org/rfc/rfc4315

[20] GULBRANDSEN, A. *Internet Message Access Protocol (IMAP) - MOVE Extension* [online]. 2013 [cit. 2023-12-02]. Dostupné z: https://www.rfc-editor.org/rfc/rfc6851

[21] EASY-SOFTWARE. *WebDAV protocol* [online]. 2023 [cit. 2024-01-05]. Dostupné z: https://easy-software.com/en/glossary/webdav-protocol/

[22] IBM. *WebDAV for HTTP Server* [online]. 2021 [cit. 2024-01-05]. Dostupné z: https://www.ibm.com/docs/en/i/7.1?topic=concepts-webdav

[23] DUSSEAULT, L. COMMERCENET. *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)* [online]. 2007 [cit. 2024-01-05]. Dostupné z: https://www.ietf.org/rfc/rfc4918.txt

[24] POT, Evert. *WebDAV features that might be useful for HTTP services* [online]. 2018 [cit. 2024-01-05]. Dostupné z: https://evertpot.com/webdav-features-for-http/

[25] RESCHKE, Julian. *RFC 5323, The SEARCH Method* [online]. 2008 [cit. 2024-01-05]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc5323#section-2

[26] CLEMM, Geoffrey M. *RFC 3253, REPORT Method* [online]. 2002 [cit. 2024-01-05]. Dostupné z: https://datatracker.ietf.org/doc/html/rfc3253#section-3.6

[27] MICROSOFT. *Exchange ActiveSync in Exchange Server* [online]. 2023 [cit. 2024-01-13]. Dostupné z: https://learn.microsoft.com/en-us/exchange/clients/exchange-activesync/exchange-activesync?view=exchserver-2019

[28] MICROSOFT. *2.2.3.181.4 SyncKey (Sync)* [online]. 2022 [cit. 2024-01-13]. Dostupné z: https://learn.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/bb649593-c793-415e-912e-e7da9b7319bf

[29] MICROSOFT. *2.2.3 Elements* [online]. 2019 [cit. 2024-01-13]. Dostupné z: https://learn.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-ascmd/f2c33009-dd06-4af4-a6a4-3969114c9a57

[30] POSTMAN. *Bind Email_QA4* [online]. [2024] [cit. 2024-03-09]. Dostupné z: https://www.postman.com/material-architect-

7281562/workspace/api/documentation/18940220-7dcf295c-5be5-4cd6-a16c-211eb8aea83e

[31] MOSTLY.AI. *Synthetic Data* [online]. 2024 [cit. 2024-03-09]. Dostupné z: https://mostly.ai

[32] NORQUAY, James. *How Many Emails Are Sent Per Day In 2024?* [online]. 2023 [cit. 2024-01-20]. Dostupné z: https://prosperitymedia.com.au/how-many-emails-are-sent-per-day-in-2023/

[33] YILMAZ, Oğuzhan. *SMTP Bomberman* [online]. 2020 [cit. 2024-01-20]. Dostupné z: https://github.com/c1982/bomberman

[34] SHRIVASTAVA, Tarunika. *Mutt – A Command Line Email Client to Send Mails from Terminal* [online]. 2023 [cit. 2024-01-20]. Dostupné z: https://www.tecmint.com/send-mail-from-command-line-using-mutt-command/

[35] PLESK. *Postfix vs Sendmail vs Exim* [online]. 2021 [cit. 2024-01-20]. Dostupné z: https://www.plesk.com/blog/various/postfix-vs-sendmail-vs-exim/

[36] LOAD IMPACT. *Open Source Load Testing Tool Review* [online]. 2017 [cit. 2024-03-09]. Dostupné z: https://medium.com/@loadimpact/open-source-load-testing-tool-review-9b3e622984c1

[37] K. O. BABALOLA, O. B. JENNINGS, E. URDIALES AND J. A. DEBARDELABEN. *Statistical Methods for Generating Synthetic Email Data Sets* [online]. Seattle, WA, USA, 2018 [cit. 2024-03-21]. ISBN 978-1-5386-5035-6. Dostupné z: https://ieeexplore.ieee.org/document/8622601. Science Paper. IEEE.

[38] MOZILLA. *MIME types (IANA media types)* [online]. c1998–2024 [cit. 2024-03-09]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types

[39] FOSDICK, Howard. *Linux commands to display your hardware information* [online]. 2019 [cit. 2024-03-09]. Dostupné z: https://opensource.com/article/19/9/linux-commands-hardware-information

[40] GITHUB. *ICal Fake Meetings Generator* [online]. 2024 [cit. 2024-03-09]. Dostupné z: https://github.com/zylai/ical-fake-meetings-generator

[41] QADAH, Ehab. *15 Best IT Infrastructure Monitoring Tools & Software [2023 Comparison]* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://sematext.com/blog/infrastructure-monitoring-tools/

[42] ZABBIX. *Zabbix processes* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual/concepts

[43] ZABBIX. *Configuration export/import* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual/xml_export_import

[44] ZABBIX. *Creating an item* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual/config/items/item

[45] ZABBIX. *Item types* [online]. 2023 [cit. 2024-03-09]. Dostupné z:

https://www.zabbix.com/documentation/current/en/manual/config/items/itemtypes

[46] ZABBIX. *Zabbix Manual* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual

[47] ZABBIX. *Active agent autoregistration* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual/discovery/auto_registration

[48] ZABBIX. *Zabbix agent (UNIX)* [online]. 2023 [cit. 2024-03-09]. Dostupné z: https://www.zabbix.com/documentation/current/en/manual/appendix/config/zabbix_agentd

[49] BADER, Alexander. GORILLA. *How can we quantify similarity between time series?* [online]. 2021, 2021-05-27 [cit. 2024-03-22]. Dostupné z: https://tech.gorilla.co/how-can-we-quantify-similarity-between-time-series-ed1d0b633ca0

[50] SERRÀ, Joan a Josep Ll. ARCOS. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*. 2014, **2014**(67), 305-314. ISSN 0950-7051.

[51] ORACLE. *Performance Tuning Roadmap* [online]. c2024 [cit. 2024-04-09]. Dostupné z: https://docs.oracle.com/cd/E13222_01/wls/docs91/perform/basics.html

[52] HOSTWINDS. *What Is The Maximum CPU Load For Your Cloud/Dedicated Servers?* [online]. 2021 [cit. 2024-04-09]. Dostupné z: https://www.hostwinds.com/tutorials/maximum-cpu-load-vpsdedicated-servers

[53] PERSONÁLKA. *Průměrný plat na pozici Specialista IT* [online]. 2024, 2024-04-20 [cit. 2024-04-20]. Dostupné z: https://prumerneplaty.cz/pozice/specialista-it

[54] *How do you compare the total cost of ownership (TCO) of cloud vs on-premise servers?* [online]. c2024 [cit. 2024-04-25]. Dostupné z: https://www.linkedin.com/advice/1/how-do-you-compare-total-cost-ownership

[55] COOLHOUSING. *PROFESIONÁLNÍ RACKHOUSING* [online]. c2009-2024 [cit. 2024-04-25]. Dostupné z: https://www.coolhousing.net/cz/rack-server-housing

[56] ANAFRA S.R.O. *Prodejce HW* [online]. c2024 [cit. 2024-05-01]. Dostupné z: https://smicro.cz

A Seznam použitých zkratek

API	Application Programming Interface
AWS	Amazon Web Services
CPU	Centrální procesorová jednotka / Computer Processing Unit
DNS	Domain Name System
DTW	Dynamic Time Warping
EAS	Exchange ActiveSync
FIO	Flexible I/O tester
GUI	Graphical User Interface
GW	GroupWare
HTTP	Hypertext Transfer Protocol
HW	Hardware
ID	Identifier
IW	IceWarp
IMAP	Internet Message Access Protocol
IWDC	IceWarp Desktop Client
JVM	Java Virtual Machine
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MDA	Mail Delivery Agent
MIME	Multipurpose Internet Mail Extensions
MSA	Mail Submissions Agent
MTA	Mail Transfer Agent
MUA	Mail User Agent
NFS	Network File System
OS	Operační Systém / Operational System
POP3	Post Office Protocol
RAM	Random-Access Memory
RFC	Request For Comments
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language

SSL	Secure Sockets Layer
TCO	Total Cost of Ownership
UID	User Identifier
URL	Uniform Resource Locator
USA	United States of America
WD	WebDav
WebDAV	Web-based Distributed Authoring and Versioning
XML	Extensible Markup Language

B Obsah přiloženého CD

├── README.md	Stručný popis spuštění simulace
├── ActiveSync	Adresář modulu EAS
├── Analyse_corelation	Adresář analyzačního prostředí
├── Controller	Adresář kontroléru (1. Vrstvy)
├── IMAP	Adresář modulu IMAP
├── Prepare_users	Adresář pro tvorbu uživatelů IceWarpu
├── SMTP	Adresář modulu SMTP
├── Syntetic_data_analysator-creator	Adresář tvorby syntetických mail. schránek
├── WebDav	Adresář modulu WebDav
└── WebMail	Adresář modulu WebMail
└── Text	Adresář s textem práce
└── thesis.pdf	Text práce ve formátu PDF