Dokumentace IMS-2024/25

Simulace Škálování webové služby - Reaktivní vs Prediktivní přístup

Datum:
27.11.2024

Autoři:
Jakub Fukala - xfukal01
Adam Kozubek - xkozub09

1. Úvod

Smyslem této práce je vytvořit simulaci škálování webové služby při proměnných podmínkách. Simulace bude porovnávat dva přístupy škálování (reaktivní a prediktivní). Na základě výsledků simulace vzorových modelů jsme stanovili obecné poznatky a demonstrovali za jakých okolností je lepší využít který přístup. Vytvořený simulační nástroj je ale určen primárně pro firmy na pomoc při rozhodování, kterou podobu škálování a s jakým nastavením je vhodné použít pro konkrétní službu.

Bylo nutné okrajově nastudovat fungování algoritmů Autoscalingu, zjistit reálné režie jednotlivých procesů a především vytvořit co nejpřesnější model předpokládané vytíženosti a na základě studia dalších podkladů, odhadnout jeho odchylku oproti reálné vytíženosti.

1.1 Autoři a významné zdroje

Autory práce jsou Jakub Fukala a Adam Kozubek, kteří ji vytvořili jako projekt do předmětu modelování a simulace na FIT VUT v Brně. Významnými zdroji práce jsou zejména výukové zdroje předmětu IMS [5] [6].

Jako reprezentativní dataset vytížení byl pro naši práci použit reálný záznam denního vytížení z Internet traffic archivu [2]. Ten jsme dále zpracovali pomocí python knihoven.

Pro vytvoření relevantních modelů autoscaleru, loadbalanceru a správné implementace obou přístupů škálování, jsme studovali internetové zdroje [1] [3] [7] [9] a knižní zdroj The Datacenter as a Computer [11]. Všechny použité zdroje jsou uvedeny na konci této dokumentace v sekci Bibliografie.

1.2 Prostředí a validace modelu

Model je vytvořen v jazyce C++ a používá knihovnu SIMLIB/C++ [4]. Ačkoliv je samotná simulace nedeterministická, její běh dá na různých strojích se stejnými vstupy srovnatelné výsledky, resp. nalezená odchylka je totožná s odchylkou mezi více výstupy téže zopakované simulace na témže stroji.

Ačkoliv je model zjednodušený, má 14 nastavitelných parametrů a odpovídá realitě, protože byl vytvořen na základě důkladné analýzy dostupných zdrojů. Také výstup simulace za daných podmínek byl srovnán s reálnými poznatky. Na základě toho model považujeme za validní.

Kapitola 2 Rozbor tématu a použitých metod/technologií

Moderní aplikace jsou stále častěji budovány na principu **mikroservisní architektury** [8], která umožňuje rozdělení aplikace na nezávislé komponenty – **mikroslužby**. Tato architektura umožňuje nejen nezávislý vývoj a nasazování jednotlivých částí systému, ale také jejich nezávislé škálování, čímž je dosaženo efektivního využití zdrojů. Klíčovou výzvou v prostředí mikroslužeb je **efektivní škálování**, které zajišťuje splnění dohodnuté úrovně služeb (**SLA**) Zdroj a zároveň minimalizaci nákladů na provoz infrastruktury.

2.1. Popis použitých postupů pro vytvoření modelu a zdůvodnění

Simulace využívá modelování **škálování aplikací v kontejnerizačním prostředí**, kde je využito jak reaktivního, tak prediktivního přístupu k řízení počtu instancí. Reaktivní škálování reaguje na aktuální celkovou hodnotu zatížení kontejneru, zatímco prediktivní škálování využívá **historická data** a metody k předpovědi budoucí zátěže.

Postup modelování:

- **Generování požadavků:** ????? systémech [11]. Byly použity reálné datové soubory pro distribuci požadavků během 24 hodin.
- **Horizontální škálování:** Model škáluje počet instancí (kontejnerů) na základě průměrné zátěže. Doba spuštění nového kontejneru odpovídá hodnotám měřeným v prostředích Docker a Kubernetes [12] ??????.
- **Latence:** V modelu je zohledněno, že latence při obsluze požadavků roste exponenciálně ????? při překročení kapacity daného kontejneru [13]?????.
- **Porovnání přístupů:** Simulace zahrnuje srovnání reaktivního a prediktivního přístupu. Prediktivní škálování dokáže díky předpovědím snížit riziko přetížení při náhlých nárůstem zátěže, zatímco reaktivní přístup může být efektivnější při minimalizaci nákladů ve stabilním prostředí.

Zdůvodnění postupů: Reaktivní škálování je jednodušší na implementaci a dobře vyhovuje scénářům s předvídatelnou a stabilní zátěží. Naopak prediktivní škálování vyžaduje více výpočetních zdrojů a kvalitní historická data, ale dokáže předcházet přetížení ve špičkách. Oba přístupy jsou v modelu implementovány a simulovány, aby byly vyhodnoceny jejich výhody a nevýhody v různých scénářích.

2.2. Popis původu použitých metod/technologií

Použité metody a technologie byly vybrány na základě ověřených vědeckých zdrojů a zkušeností z provozních systémů:

- **Kontejnerizace:** Simulace je navržena s ohledem na vlastnosti moderních kontejnerizačních nástrojů, jako je Docker a Kubernetes [12].
- **Škálování:** Pravidla škálování (např. prahové hodnoty) byla navržena na základě ??????????.
- **Generování zátěže:** Datová sada obsahující záznamy o počtu požadavků za minutu pochází z reálných provozních měření a byla přizpůsobena pro simulaci pomocí normálního rozptylu s 15% odchylkou.

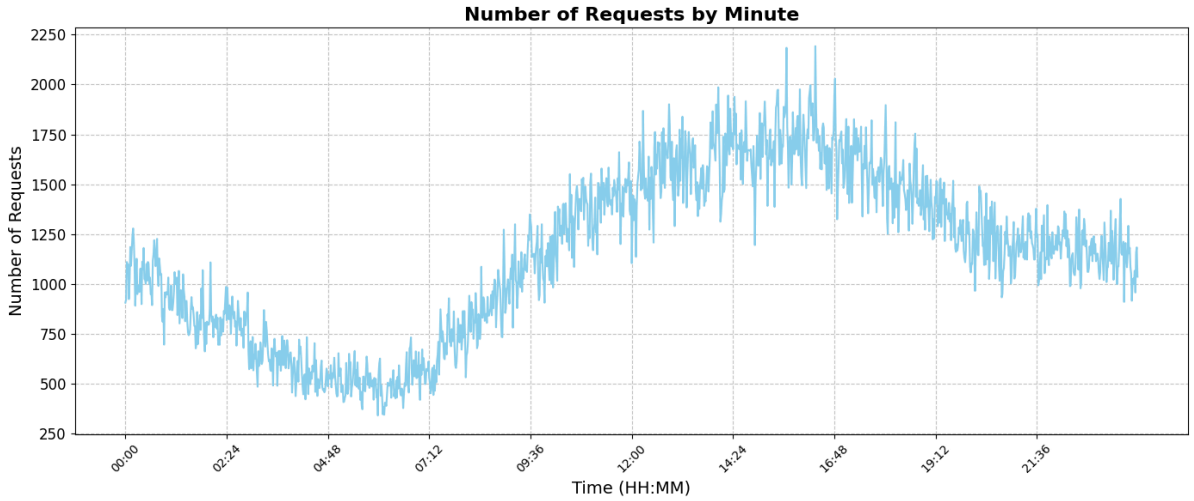
Tento přístup zajišťuje validitu modelu a poskytuje užitečné srovnání škálovacích metod, které mohou být aplikovány v reálných systémech. Na základě těchto simulací lze posoudit efektivitu různých strategií škálování a jejich dopad na náklady i kvalitu služeb.

3. Koncepce - modelářská témata

3.1 Způsob vyjádření konceptuálního modelu

Konceptuální model simuluje dynamické škálování aplikací v prostředí mikroservisní architektury. Tento model abstrahuje klíčové vlastnosti systému, jako je zátěž, odezva aplikací, přidávání a odebírání kontejnerů a jejich čas spuštění, a redukuje je na měřitelné a simulovatelné metriky. Model je vyjádřen kombinací schématického znázornění, stavových diagramů a matematických rovnic:

- **Schéma systému**: Na obrázku XXX je znázorněn obecný tok požadavků od uživatele až po zpracování v rámci kontejnerů. Schéma ilustruje, jak zátěž vstupuje do systému, je rozdělena mezi aktivní kontejnery a jak probíhá reaktivní nebo prediktivní škálování.
- **Matematické rovnice**: ??????????
- **Stavový diagram**: Na obrázku YYY je znázorněn stavový diagram pro jednotlivé kontejnery, které mohou být v jedné z následujících stavů: **Neaktivní**, **Spouštění**, **Připravený** nebo **ZAktivní** Diagram také zahrnuje přechody mezi stavy, například aktivaci při překročení zátěže nebo deaktivaci při nízkém vytížení.
- **Synchronizace procesů**: Procesy generace požadavků, zpracování v kontejnerech a rozhodování škálovacích algoritmů jsou synchronizovány prostřednictvím diskrétní simulace řízené událostmi, kterou zajišťuje knihovna Simlib. Obrázek ZZZ znázorňuje interakci mezi procesy.



7. Bibliografie

[1] CONRAN, Matt; 2015. Load Balancing and Scale-Out Architectures. Dostupné z: https://network-insight.net/2015/02/26/load-balancing-and-scale-out-architectures/. [cit. 2024-11-29].

[2] DANZIG, Peter; et al. 2008. The Internet Traffic Archive. Dostupné z: https://ita.ee.lbl.gov/html/traces.html. [cit. 2024-11-29].

[3] NEWMAN, David; 2024. Load Balancing Fundamentals: How Load Balancers Work. Dostupné z: https://www.linode.com/docs/guides/load-balancing-fundamentals/. [cit. 2024-11-27].

[4] PERINGER, Petr; MARTINEK, David; LEŠKA, David; 2021. SIMLIB/C++ Documentation. Dostupné z: https://www.fit.vut.cz/person/peringer/public/SIMLIB/doc/html/. [cit. 2024-11-26].

[5] PERINGER, Petr; 2022. Modelování a simulace - Studijní opora. Dostupné z: https://moodle.vut.cz/pluginfile.php/900581/mod_resource/content/2/opora-ims.pdf. [cit. 2024-11-28].

[6] PERINGER, Petr; HRUBÝ Martin; 2024. Modelování a simulace - Prezentace. Dostupné z: https://www.fit.vut.cz/person/peringer/public/IMS/prednasky/IMS.pdf. [cit. 2024-11-26].

[7] RABIU, Shamsuddeen; CHAN, Huah Yong; 2022. A Cloud-Based Container Microservices: A Review on Load-Balancing and Auto-Scaling Issues. Dostupné z: https://www.researchgate.net/publication/366562529_A_Cloud-Based_Container_Microservices_A_Review_on_Load-Balancing_and_Auto-Scaling_Issues. [cit. 2024-11-27].

[8] MICROSOFT - AzureDevOps; 2000. Microservices architecture design. Dostupné z: https://learn.microsoft.com/en-us/azure/architecture/microservices/. [cit. 2024-11-29].

[9] SAFA, Bouguezz; 2024. Název: Podnázev. Dostupné z: https://www.baeldung.com/cs/scaling-horizontally-vertically. [cit. yyyy-mm-dd].

[10] BARROSO, Luiz André; HÖLZLE, Urs; RANGANATHAN, Parthasarathy; 2019. The datacenter as a computer. Dostupné z: https://link.springer.com/book/10.1007/978-3-031-01761-2. [cit. 2024-11-29].

VZOROVÁ BIBLIOGRAFIE:

1 autor:

[11] PŘÍJMENÍ Jméno; 2000. Název: Podnázev. Dostupné z: link.example.com. [cit. yyyy-mm-dd].

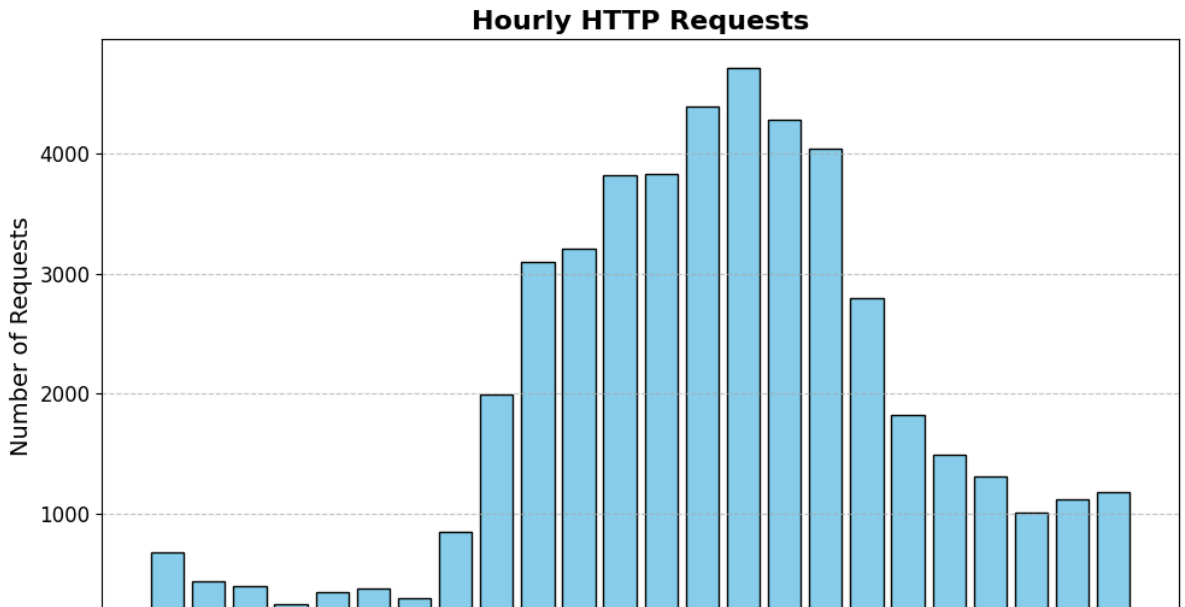
N autorů:

[12] PŘÍJMENÍ Jméno; PŘÍJMENÍ Jméno; PŘÍJMENÍ Jméno; 2000. Název: Podnázev. Dostupné z: link.example.com. [cit. yyyy-mm-dd].

Mnoho autorů:

[13] PŘÍJMENÍ Jméno; et al. 2000. Název: Podnázev. Dostupné z: link.example.com. [cit. yyyy-mm-dd].

- https://www.researchgate.net/publication/366562529_A_Cloud-Based_Container_Microservices_A_Review_on_Load-Balancing_and_Auto-Scaling_Issues
  - Typické webové aplikace mohou mít průměrné zatížení mezi 20 % až 60 % CPU během běžného provozu, s krátkodobými špičkami až 90 % CPU při neočekávaných událostech.
  - Vytvoření a zpřístupnění nového kontejneru na běžné platformě, jako je Kubernetes, trvá mezi 5 až 15 sekundami v ideálním případě. Při zatížení systému se tento čas může zvýšit až na 30 sekund.
- https://www.linode.com/docs/guides/load-balancing-fundamentals/
  - Latence přesměrování: Dynamické algoritmy LoadBalanceru, jako Weighted Round Robin, mají přidanou latenci přesměrování obvykle <10 ms na požadavek. Tato hodnota se může zvýšit při přetížení na 30-50 ms.
  - Směrodatná odchylka u odhadovaného zatížení (např. pro metriky CPU) se typicky pohybuje mezi 5 % až 25 %, v závislosti na kvalitě dat a složitosti modelu.
- https://network-insight.net/2015/02/26/load-balancing-and-scale-out-architectures/
  - Reaktivní autoscalery reagují obvykle v časech 1-5 minut, což zahrnuje čas na detekci potřeby škálování a spuštění nových instancí.
- https://ita.ee.lbl.gov/html/traces.html
  - Dostupná data jsme pomocí python knihovny překreslili do následujícího grafu:



Zadání:

Povinná osnova dokumentace (technické zprávy)

Smysl této povinné osnovy je dát praktický návod pro sepsání technické zprávy k projektu. Číslované kapitoly jsou **naprosto závazné** - jejich případná absence bude hodnocena bodovou srážkou. Berte technickou simulační zprávu jako formu protokolu. Rozhodně se nejedná o beletrii, váš osobní příběh nebo filosofickou úvahu - tyto formy si můžete dovolit až v pokročilejším věku (kdy zřejmě už pochopíte, že beztak v technice jsou preferována relevantní fakta, specificky u platících zákazníků).

Kapitoly první úrovně mají závazný název. Kapitoly druhé úrovně nemají závazný název, upravte si název dle potřeby. V dokumentu jsou povinné tématické bloky kapitol první a druhé úrovně s nenulovým obsahem. Kapitola první úrovně předpokládá určitý text (uvedeno v popisu) rozšířený o témata kapitol druhé úrovně.

Podstatné je, že technická zpráva se neobhajuje a nekomentuje. Všecko relevantní má být obsaženo v ní. Nejsou přípustná žádná dodatečná ústní sdělení. Pokud čtenář z vaší technické zprávy něco chybně pochopí, je to vždy vaše vina.

Je jasné, že různá témata budou klást důraz na jiné partie zprávy:

- implementační - literatura, koncepce, zpracování implementace, ověření na průkazném demopříkladě
- modelační/aplikační - literatura a zjišťování v terénu, zdůvodnění koncepce modelu, simulační studie vedoucí k jasnému závěru

Bez ohledu na charakter tématu lze v technické dokumentaci VŽDY odděleně popsat:

- Úvod a motivaci
- Shrnutí relevantních faktů, zdroje informací
- Koncepci metody, přístupu, modelu
- Implementaci metody, modelu
- Experimentování
- Závěr

1. Úvod

Úvod musí vysvětlit, **proč se celá práce dělá a proč má uživatel výsledků váš dokument číst** (prosím, projekt sice děláte pro získání zápočtu v IMS, ale mohou existovat i jiné důvody). Případně, co se při čtení dozví.

Například:

- v této práci je řešena implementace ..., která bude použita pro sestavení modelu ...
- na základě modelu a simulačních experimentů bude ukázáno chování systému ... v podmínkách ...
- smyslem experimentů je demonstrovat, že pokud by ..., pak by ...
- Poznámka: u vyžádaných zpráv se může uvést, že zpráva vznikla na základě požadavku ... (u školní práce takto zdůvod'novat projekt ovšem nelze, že). Je velmi praktické zdůvodnit, **v čem je práce náročná a proto přínos autora nepopiratelný** (např. pro zpracování modelu bylo nutno nastudovat ..., zpracovat, ... model je ve svém oboru zajímavý/ojedinělý v ...).

Podkapitoly:

- kapitola 1.1: Kdo se na práci podílel jako autor, odborný konzultant, dodavatel odborných faktů, význačné zdroje literatury/fakt, ...
    - je ideální, pokud jste vaši koncepci konzultovali s nějakou autoritou v oboru (v IMS projektu to je hodnoceno, ovšem není vyžadováno)
    - pokud nebudete mít odborného konzultanta, nevadí. Nelze ovšem tvrdit, že jste celé dílo vymysleli s nulovou interakcí s okolím a literaturou.
- kapitola 1.2: V jakém prostředí a za jakých podmínek probíhalo experimentální ověřování validity modelu – pokud čtenář/zadavatel vaší zprávy neuvěří ve validitu vašeho modelu, **obvykle vaši práci odmítne už v tomto okamžiku**.

2. Rozbor tématu a použitých metod/technologií

Shrnutí všech podstatných faktů, které se týkají zkoumaného systému (co možná nejvěcnějším a technickým (ideálně formálně matematickým) přístupem, žádné literární příběhy). **Podstatná fakta o systému musí být zdůvodněna a podepřena důvěryhodným zdrojem** (vědecký článek, kniha, osobní měření a zjišťování). Pokud budete tvrdit, že ovce na pastvě sežere dvě kila trávy za den, musí existovat jiný (důvěryhodný) zdroj, který to potvrdí. Toto shrnutí určuje míru důvěryhodnosti vašeho modelu (nikdo nechce výsledky z nedůvěryhodného modelu). Pokud nebudou uvedeny zdroje faktů o vašem systému, hrozí ztráta bodů.

- kapitola 2.1: Popis **použitých postupů** pro vytvoření modelu a zdůvodnění, **proč jsou pro zadaný problém vhodné**. Zdůvodnění může být podpořeno ukázáním alternativního přístupu a srovnáním s tím vaším. Čtenář musí mít jistotu, že zvolené kapitola 2.2: Popis **původu** použitých metod/technologií (odkud byly získány (odkazy), zda-li jsou vytvořeny autorem) - převzaté části dokumentovat (specificky,

pokud k nim přísluší nějaké autorské oprávnění/licence). Zdůvodnit potřebu vytvoření vlastních metod/nástrojů/algoritmů. Ve většině případů budete přebírat již navržené metody/algoritmy/definice/nástroje a je to pro školní projekt typické. Stejně tak je typické, že **studenti chybně vymýšlí již hotové věci a dojdou k naprostému nesmyslu** - je třeba toto nebezpečí eleminovat v tomto zdůvodnění.

Velmi důležité, až fanaticky povinné, kontrolované a hodnocené: na každém místě v textu, kde se poprvé objeví pojem z předmětu IMS bude v závorce uveden odkaz na předmět a číslo slajdu, na kterém je pojem definován. Pokud bude významný pojem z předmětu IMS takto nedokumentován v textu a zjevně bude používán nevhodným nebo nepřesným způsobem, bude tento fakt hodnocen bodovou ztrátou. Tento požadavek je míněn s naprostou vážností. Cílem je vyhnout se studentské tvůrčí činnosti ve vysvětlování známých pojmů, což mnohdy vede k naprostým bludům, ztrátě bodů a zápočtů. Pokud student pojem cituje korektně a přesto nekorektně používá, bude to hodnoceno dvojnásobnou bodovou ztrátou.

3. Koncepce - modelářská témata

Konceptuální model je abstrakce reality a redukce reality na soubor relevantních faktů pro sestavení simulačního modelu. Předpokládáme, že model bude obsahovat fakta z "Rozboru tématu". Pokud jsou některá vyřazena nebo modifikována, je nuto to zde zdůvodnit (například: zkoumaný subjekt vykazuje v jednom procentu případů toto chování, ovšem pro potřeby modelu je to naprosto marginální a smíme to zanedbat, neboť ...). **Pokud některé partie reality zanedbáváte nebo zjednodušujete, musí to být zdůvodněno a v ideálním případě musí být prokázáno, že to neovlivní validitu modelu.** Cílem konceptuálního (abstraktního) modelu je formalizovat relevantní fakta o modelovaném systému a jejich vazby. Podle koncept. modelu by měl být každý schopen sestavit simulační model.

- kapitola 3.1: Způsob vyjádření konceptuálního modelu musí být zdůvodněn (na obrázku xxx je uvedeno schéma systému, v rovnicích xx-yy jsou popsány vazby mezi ..., způsob synchronizace procesů je na obrázku xxx s Petriho sítí).
- kapitola 3.2: Formy konceptuálního modelu (důraz je kladen na srozumitelnost sdělení). Podle potřeby uveďte konkrétní relevantní:
    - obrázek/náčrt/schéma/mapa (možno čitelně rukou)
    - matematické rovnice - u některých témat (např. se spojitými prvky, optimalizace, ...) naprosto nezbytné. **Dobré je chápat, že veličiny (fyzikální, technické, ekonomické) mají jednotky, bez kterých údaj nedává smysl.**
    - stavový diagram (konečný automat) nebo Petriho síť - spíše na abstraktní úrovni. Petriho síť nemá zobrazovat výpočty a přílišné detaily. Pokud se pohodlně nevejde na obrazovku, je nepoužitelná. Možno rozdělit na bloky se zajímavými detaily a prezentovat odděleně abstraktní celek a podrobně specifikované bloky (hierarchický přístup).

3. Koncepce - implementační témata

Popište abstraktně architekturu vašeho programu, princip jeho činnosti, významné datové struktury a podobně. Smyslem této kapitoly je podat informaci o programu bez použití názvů tříd, funkcí a podobně. Tuto kapitolu by měl pochopit každý technik i bez informatického vzdělání. Vyjadřovacími prostředky jsou vývojové diagramy, schémata, vzorce, algoritmy v pseudokódu a podobně. Musí zde být vysvětlena nosná myšlenka vašeho přístupu.

4. Architektura simulačního modelu/simulátoru

Tato kapitola má různou důležitost pro různé typy zadání. U implementačních témat lze tady očekávat jádro dokumentace. Zde můžete využít zajímavého prvku ve vašem simulačním modelu a tady ho "prodat".

- kapitola 4.1: Minimálně je nutno ukázat mapování abstraktního (koncept.) modelu do simulačního (resp. simulátoru). Např. které třídy odpovídají kterým procesům/veličinám a podobně.

5. Podstata simulačních experimentů a jejich průběh

Nezaměňujte pojmy testování a experimentování (důvod pro bodovou ztrátu)!!! Zopakovat/shrnout **co přesně chcete zjistit experimentováním** a proč k tomu potřebujete model. **Pokud experimentování nemá cíl, je celý projekt špatně.** Je celkem přípustné u experimentu odhalit chybu v modelu, kterou na základě experimentu opravíte. Pokud se v některém experimentu nechová model podle očekávání, je nutné tento experiment důkladně prověřit a chování modelu zdůvodnit (je to součást simulačnické profese). Pokud model pro některé vstupy nemá důvěryhodné výsledky, je nutné to zdokumentovat. Pochopitelně model musí mít důvěryhodné výsledky pro většinu myslitelných vstupů.

- kapitola 5.1: Naznačit postup experimentování – jakým způsobem hodláte prostřednictvím experimentů dojít ke svému cíli (v některých situacích je přípustné "to zkoušet tak dlouho až to vyjde", ale i ty musí mít nějaký organizovaný postup).
- kapitola 5.2: Dokumentace jednotlivých experimentů - souhrn vstupních podmínek a podmínek běhu simulace, komentovaný výpis výsledků, závěr experimentu a plán pro další experiment (např. v experimentu 341. jsem nastavil vstup x na hodnotu X, která je typická pro ... a vstup y na Y, protože chci zjistit chování systému v prostředi ... Po skončení běhu simulace byly získány tyto výsledky ..., kde je nejzajímavější hodnota sledovaných veličin a,b,c které se chovaly podle předpokladu a veličin d,e,f které ne. Lze z toho usoudit, že v modelu není správně implementováno chování v podmínkách ... a proto v následujících experimentech budu vycházet z modifikovaného modelu verze ... Nebo výsledky ukazují, že systém v těchto

podmínkách vykazuje značnou citlivost na parametr x ... a proto bude dobré v dalších experimentech přesně prověřit chování systému na parametr x v intervalu hodnot ... až ...)
- kapitola 5.3: Závěry experimentů – bylo provedeno N experimentů v těchto situacích ... V průběhu experimentování byla odstraněna ... chyba v modelu. Z experimentů lze odvodit chování systémů s dostatečnou věrohodností a experimentální prověřování těchto ... situací již napřinese další výsledky, neboť ...

6. Shrnutí simulačních experimentů a závěr

Závěrem dokumentace se rozumí **zhodnocení simulační studie a zhodnocení experimentů** (např. Z výsledků experimentů vyplývá, že ... při předpokladu, že ... Validita modelu byla ověřena ... V rámci projektu vznikl nástroj ..., který vychází z ... a byl implementován v ...).

- do závěru se nehodí psát poznámky osobního charakteru (např. práce na projektu mě bavila/nebavila, ...). **Technická zpráva není osobní příběh autora.**
- absolutně nikoho nezajímá, kolik úsilí jste projektu věnovali, důležitá je pouze kvalita zpracování simulátoru/modelu a obsažnost simulační studie (rozhodně ne např.: projekt jsem dělal ... hodin, což je víc než zadání předpokládalo. Program má ... řádků kódu). **Pokud zdůrazňujete, že jste práci dělali významně déle než se čekalo, pak tím pouze demonstrujete vlastní neschopnost (to platí zejména v profesním životě).**
- do závěru se velmi nehodí psát "auto-zhodnocení" kvality práce, to je výhradně na recenzentovi/hodnotiteli (např. v projektu jsem zcela splnil zadání a domnívám se, že můj model je bezchybný a výsledky taktéž). Statisticky častý je pravý opak autorova auto-zhodnocení. Pokud přesto chcete vyzdvihnout kvalitu svého díla (což je dobře), tak vaše výroky musí být naprosto nepopiratelně zdůvodněny a prokázány (např. pomocí jiného referenčního přístupu, matematického důkazu, analýzy, ...).

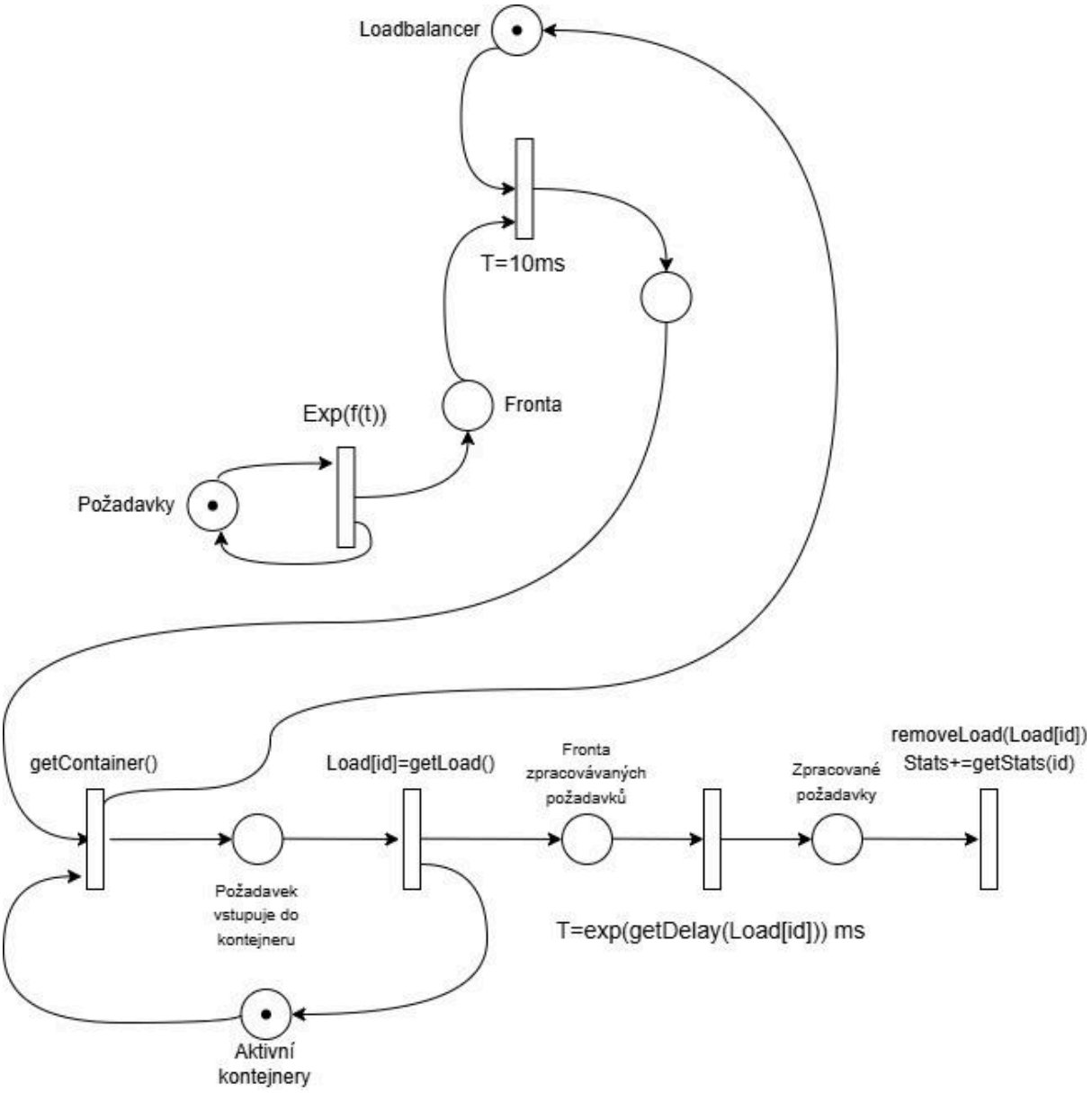
Obecné poznámky

- v dokumentaci neocením "úvodní stránku s logem fakulty" a obsah. Obvzláště obsah jako jedna stránka u čtyř-pěti stránkového dokumentu působí směšně. Tyto dvě strany pochopitelně smíte zprávě předřadit, nepočítám je ovšem do počtu stran zprávy.
- grafy mají své náležitosti - identifikační název grafu (případně jeho číslo), cejchované osy s názvem veličiny na dané ose (včetně její jednotky). V případě grafu kombinujícího více jevů i legenda dokumentující grafické vyjádření jevů v grafu.
- **veškeré tabulky a grafy musí být komentovány v textu - čtenáři musí řečeno, co v tom grafu uvidí a čeho si má všimnout.**

Datové zabezpečení naší práce:

Proměnlivost předpokládané vytíženosti Rozdělení požadavků: Typické webové aplikace mohou mít průměrné zatížení mezi 20 % a 60 % CPU během běžného provozu, s krátkodobými špičkami až 90 % CPU při neočekávaných událostech RESEARCHGATE : https://www.researchgate.net/publication/366562529_A_Cloud-Based_Container_Microservices_A_Review_on_Load-Balancing_and_Auto-Scaling_Issues . *Směrodatná odchylka oproti předpokladu* Odhady v simulaci: Model prediktivního škálování může zahrnovat rozdíl mezi předpokládaným a skutečným zatížením. Směrodatná odchylka u odhadovaného zatížení (např. pro metriky CPU) se typicky pohybuje mezi 5 % až 25 %, v závislosti na kvalitě dat a složitosti modelu AKAMAI : https://www.linode.com/docs/guides/load-balancing-fundamentals/ . *Vnitřní režie zpřístupnění kontejneru* Čas inicializace: Vytvoření a zpřístupnění nového kontejneru na běžné platformě, jako je Kubernetes, trvá mezi 5 až 15 sekundami v ideálním případě. Při zatížení systému se tento čas může zvýšit až na 30 sekund. RESEARCHGATE : https://www.researchgate.net/publication/366562529_A_Cloud-Based_Container_Microservices_A_Review_on_Load-Balancing_and_Auto-Scaling_Issues TECHNOLOGY FOCUSED HUB : https://network-insight.net/2015/02/26/load-balancing-and-scale-out-architectures/ . *Algoritmy LoadBalancer a Autoscaler* Latence přesměrování: Dynamické algoritmy LoadBalanceru, jako Weighted Round Robin, mají přidanou latenci přesměrování obvykle <10 ms na požadavek. Tato hodnota se může zvýšit při přetížení na 30-50 ms AKAMAI : https://www.linode.com/docs/guides/load-balancing-fundamentals/ . Doba odezvy na škálování: Reaktivní autoscalery reagují obvykle v časech 1-5 minut, což zahrnuje čas na detekci potřeby škálování a spuštění nových instancí TECHNOLOGY FOCUSED HUB : https://network-insight.net/2015/02/26/load-balancing-and-scale-out-architectures/ .

Petriho síť 1. část - správa požadavků (LoadBalancer):



Petriho síť 2. část - správa aktivních kontejnerů (Autoscaler):