A Glimpse into Performance Testing

Tomáš Vepřek
Tietoevry Ostrava
tomas.veprek@tietoevry.com

Agenda – Topics

- Part 1
  - Motivation for performance testing
- Part 2
  - What is performance?
  - Fundamental metrics and concepts
- Part 3
  - How to do performance testing

About Myself …

- Graduated from VŠB Ostrava in 2003
- Since 2005 employed at Tieto Czech

- Java developer
- Functional software tester
- 3rd tier support of EMC Documentum applications
- Performance tester (2007 - present)

PART 1

Why Performance Testing?

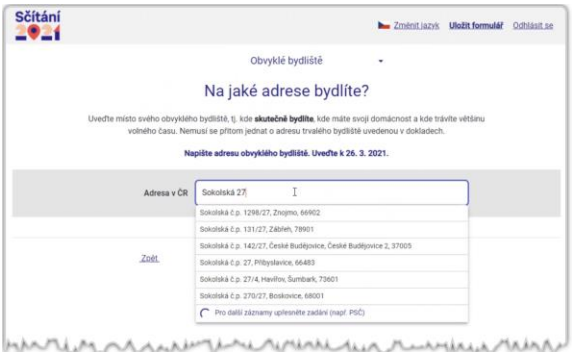
Real-life Example #1

- **Czech online census system (27 March 2021)**
  - System overloaded a few hours after officially launched
  - It took half a day before the problem was fixed
  - Overloading problem occurred again
  - Address predictive search was the culprit

Source: zive.cz

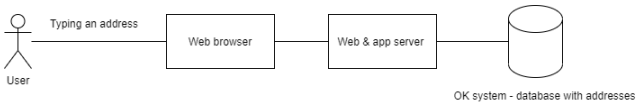
- V roce 2021 (27. března – 11. května) bylo sčítání poprvé provedeno pomocí elektronického formuláře.
- První problémy po spuštění se projevily první den kolem 6. hodiny ráno
- Do té doby bylo odesláno a uloženo několik tisíc formulářů
- Formuláře bylo možné opět odesílat od 17:15 téhož dne
- Od 19:35 je systém opět velmi vytížený
- Do sobotní 22.hodiny bylo odesláno 250 000 formulářů (~ 500 000 sečtených občanů)
- Příčinou problému byl modul pro prediktivní hledání adres, který po každém znaku odesílal požadavek do databáze

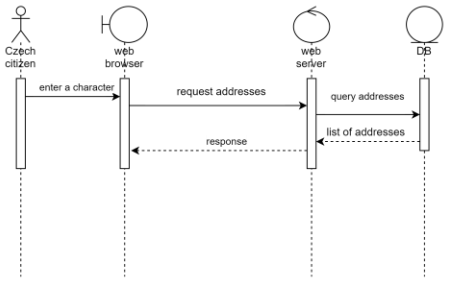
Real-life Example #1 (contd.)



Source: lupa.cz

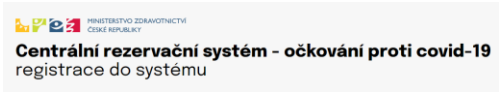
Real-life Example #1 (contd.)





Real-life Example #2

- Czech Covid-19 vaccination registration system (May 17, 2021)
  - Registration could not be completed for about 50 minutes for the age category 40-45
  - Text message with PIN not delivered
  - No field available for entering PIN
  - Website unavailable entirely



Source: https://registrace.mzcr.cz/

Real-life Example #2 (contd.)

Source: https://www.irozhlas.cz/

Real-life Example #3

- **HealthCare.gov web site**
  - Fails miserably after released on 1st October, 2013
  - Only 1% of people who visited the site during the first week was able to enroll
  - Stress tests conducted by contractors 1 day before the go-live date revealed that the website became significantly slow for only 1100 simultaneous users. The target was 50000 – 60000

Source: http://sphweb.bumc.bu.edu

- Webová aplikace byla vytvořena, aby fungovala jako prostředek pro centralizaci dat, které umožní Američanům porovnat ceny plánů zdravotního pojištění v jednotlivých státech a také aby jim umožnila přihlásit se k jednomu z nich a souběžně zjistit, jestli mají nárok na vládní dotaci.
- Webová apllikace byla uvolněna 1.října 2013, což byl poslední termín stanovený Obamovou administrativou.
- Nejzažší termín pro výběr zdravotního pojištění, který měl začít v lednu 2014, byl stanoven na 23.prosince.
- Pouze 1% lidí, kteří navštívili stránku během prvního týdne po uvedení aplikace, dokončili celou transakci – výběr zdravotního pojištění.
- Stresové testy, které byly provedeny třetí stranou den před spuštěním, odhalily, že webová stránka se významně zpomaluje už pří zátěži 1100 aktivních uživatelů. Přitom očekávána zátěž byla mnohonásobně větší – 50000 až 60000 aktivních uživatelů.
- Webová aplikace byla vytvořena celou řadou federálních dodavatelů, zejména

dodatvatelů od CMS (Centers for Medicare and Medicaid Services).

- Původní rozpočet byl $93.7 million, ale celkové náklady webové aplikace dosáhly $1.7 billion (podle the Office of Inspector General)

Real-life Example #5

- **Validators for Litacka public transport card (2016)**
  - Machines for transferring Open card coupons to Litacka cards (aka validators) slow or not working at all
  - Peak hours 8-9 and 17-18 o'clock
  - Validation system overloaded

Source: https://www.litacka.cz

- Zdroj: https://www.irozhlas.cz/regiony/aktivovat-litacku-muze-byt-problem-rano-a-vecer-kvuli-karte-vypadavaji-validatory_201609030600_pjadrny (září 2016)
- Občané, kteří si nechali poslat Lítačku (náhradu za Open Card) poštou, mají problémy s jejích aktivací pomocí tzv. validátorů v prostorách pražského metra
- Validátory často vypadávají, nejčastěji mezi 8.-9. a 17.-18. hodinou
- Jednou z možností je (potvrdil i mluvčí dopravního podniku), že systém validátorů je během zmíněných časů přetížen

Causes of Failures

- Poorly planned projects
- Bad risk management
- Rigid processes
- Too many subcontractors
- Enormous time pressure
- Not enough skilled people
- Decisions based on wrong or little information

Uvedené 4 případy jsou jen kapkou v moři softwarových projektů, které nedosáhly kýženého cíle tak, jak jsi na začátku představovaly. Jedna z příčin obrovských problémů, které tyto projekty musely řešit při zprovoznění zmíněných systémů, bylo podcenění nebo nekvalitní provedení zátěžového testování a ignorování výsledků zátěžových testů. A nejen zátěžových testů, ale testování obecně. Mezi další příčiny patřily špatné plánování, řízení rizik, těžkopádné procesy, příliš mnoho dodavatelů, nedostatek času a nedostatečně kvalifikovaní nebo vyškolení lidé.

Všechny tyto zmíněné faktory ovlivňují **kvalitu** jakéhokoli softwarového systému.

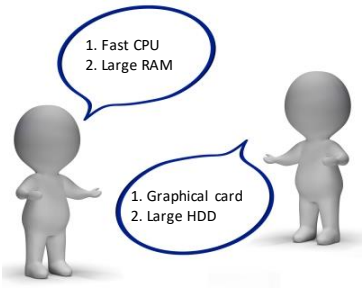
Quality

- Subjective and context-driven term
- Relationship of many different aspects, many of which are out of our control
- Needs regular reviews
- Multidimensional

Testování je způsob, pomocí kterého můžeme zjistit informace o skutečném stavu a rizicích softwarového produktu a předat je osobám, které jsou odpovědné za rozhodování (manažerům). Jako testeři musíme být připraveni, že náš pohled na kvalitu produktu nemusí být vždy akceptován manažery. Může být zcela legitimní, když nás manažeři „přehlasují“, protože kromě našich vstupů posuzují i ekonomické aspekty, například náklady za odložené uvedení produktu na trh.

Quality – Subjective

Buying a new computer …



Kvalita je subjektivní termín. To, co jeden člověk považuje za kvalitní, může druhý vidět naprosto odlišně.

Quality - Context-driven

Smart meter

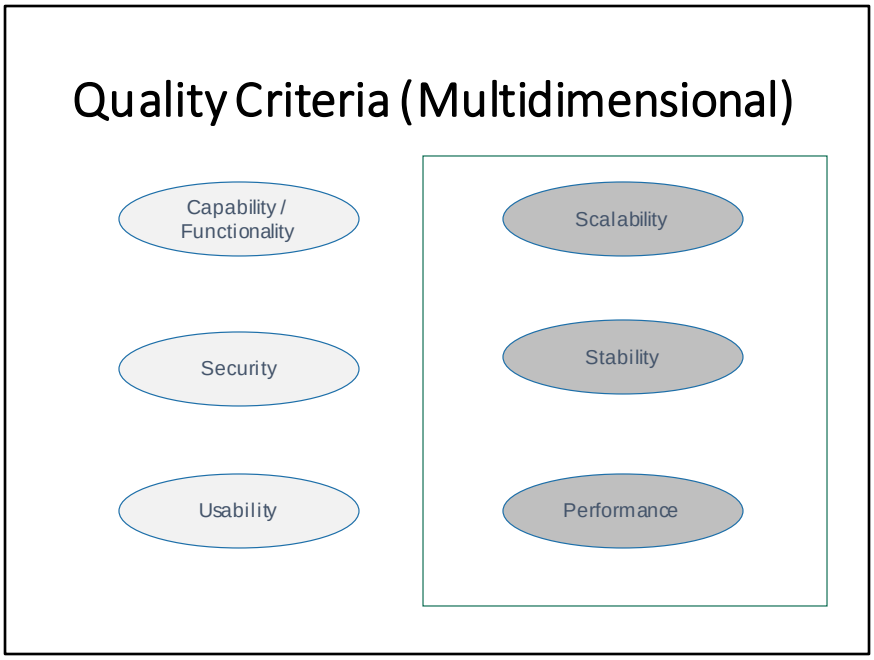
Pacemaker

Vnímání kvality je ovlivněno kontextem. Příklad: Software instalovaný v kardiostimulátoru bude mít zcela jistě přísnější kritéria oproti softwaru, který je nainstalovaný v chytrém elektroměru. Důsledky selhání srdečního stimulátoru jsou nesrovnatelně vážnější oproti selhání softwaru v chytrém elektroměru.

Quality – Different Factors

- Development team
  - Skills
  - Experience
  - Personality
- Customer
- Management
- Rewarding system
- Allotted time
- Working environment

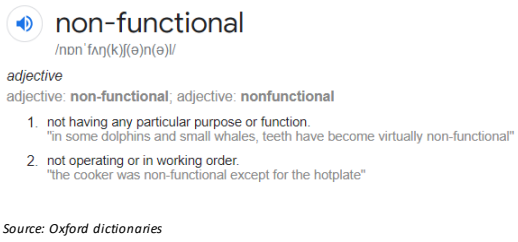
Kvalita softwaru bezesporu závisí na celé řadě faktorů, například způsob sestavení týmu, znalosti, dovednosti, ohodnocení jeho členů, prostředí, ve kterém pracují a dostupný čas.



- Kvalitu softwarového produktu lze posuzovat na základě celé řady kritérií (vícerozměrná)
- Ta nejčastější je funkčnost, nebo-li schopnost softwaru vykonávat to, co se po něm očekává
- Příklady:
    - Eshop nabízející knihy musí umožňovat hledání v katalogu knih a jejich nákup
    - Mailový klient musí umožňovat zaslání emailů, příjem emailů a správu emailů
- Velmi často se při testování zapomíná na další kritéria, jako je bezpečnost, výkonnost (rychlost zpracování), škálovatelnost, stabilita atd.
- Často se o těchto dalších kritériích hovoří jako o nefunkčních. Pojem nefunkční je však nevhodně zvolený termín.

Non-functional Criteria / Testing

- Please, do not use this term!



non-functional

/nɒnˈfʌŋ(k)ʃ(ə)n(ə)l/

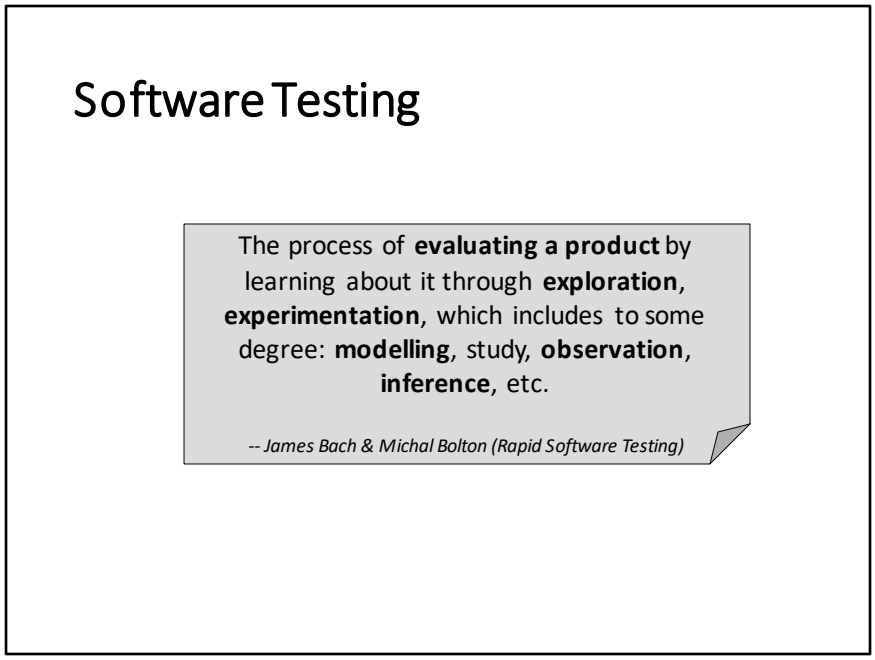
adjective

adjective: **non-functional**; adjective: **nonfunctional**

1. not having any particular purpose or function.
   "in some dolphins and small whales, teeth have become virtually non-functional"
2. not operating or in working order.
   "the cooker was non-functional except for the hotplate"

Source: Oxford dictionaries

- V softwarovém inženýrství se používají termíny, které jsou zavádějící
- Kromě pojmu nefuční testování / kritéria / požadavky, je to například RAM (paměť s náhodným přístupem), kde mnohem vhodnější termín by byl paměť s libovolným přístupem (arbitrary access memory) – viz přednáška od Martina Thompsona (Designing for Performance, https://www.youtube.com/watch?v=03GsLxVdVzU)



Evaluate: posouzení produktu - jak je dobrý nebo špatný? Vyhodnocení jeho kvalitativních kritérií

Exploration: při zkoumání produktu používáme strategii, která nám dává velkou pravděpodobnost, že objevíme problémy, na kterých našim klientům záleží. Každý test by měl být navžen tak, aby přinesl hodnotnou informaci, jinými slovy, aby snížil entropii produktu, který zkoumáme.

Experimentation: testování je experimentální činnost

Modelling: během testování si vytváříme o zkoumaném produktu různé modely. Příklady: stavový automat, případy užití, diagram činností, množina funkcí, časové závislosti, rozhraní. Pojem test coverage (pokrytí) se vždy vztahuje ke konkrétnímu modelu.

Observation: sledování toho, jak se systém chová a jak reaguje během našich experimentů.

Inference: na základě výsledků našeho pozorování si vytváříme závěry a teorie o tom,

jak systém funguje nebo nefunguje.

Bug (Defect)

Anything that threatens the value of the product

-- James Bach & Michael Bolton (Rapid Software Testing)

Bug is something that bugs somebody… who matters.

-- James Bach & Michael Bolton (Rapid Software Testing)

PART 2

What is Performance?

Throughput

- The rate at which work is completed

Throughput (propustnost) je jedna ze zásadních metrik, které měříme, když chceme posoudit, jakou výkonnost má softwarový produkt nebo jakákoli jeho část.

Obrázek zobrazuje zákazníky čekající u několika pokladen. Propustnost jedné pokladny definujeme jako počet zákazníků, které jsou u ní obslouženi za jednotku času (např. hodinu). Pro zjištění celkové propustnosti, vynásobíme propustnost jedné pokladny počtem pokladen (za předpokladu, že všechny pokladní odbavují stejně rychle) nebo sečteme propustnosti jednotlivých pokladen.

Throughput (cont'd)

- Examples:
  - Book orders per second
  - API calls per second
  - SQL queries per second
  - Disk read / write operations per second
  - Bytes / bits per second

V případě softwarových systémů můžeme hovořit o průchodnosti z pohledu business transakcí, API volání a dalších zdrojů, které jsou uvedeny ve výčtu.

Response Time

- **Latency** = time the request for an operation spent waiting in a queue
- **Service time** = time an operation takes to complete
- **Response time** = Latency + Service time

- Easily quantifying degradation or improvement

Vedle průchodnosti je další zásadní metrikou odezva. Je definována jako čas nutný k tomu, aby softwarový produkt nebo jeho jakákoli část dokončila operaci. Na příkladu s lidmi čekajícími u pokladny v potravinách by byla odezva definována součtem času, během kterého zákazník čeká ve frontě, než přijde na řadu, a doby, kterou potřebuje pokladní na jeho obsloužení.

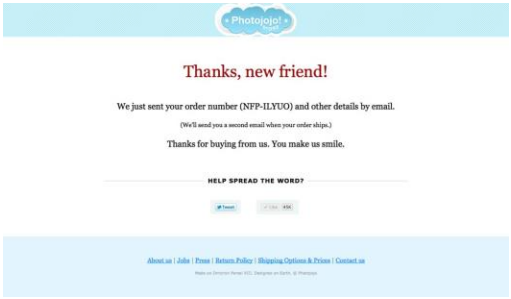
Čas strávený čekáním ve frontě se nazává latence a doba na vyřízení požadavku se nazývá service time.

Výhodou odezvy je snadnost, s jakou se dá použít při kvantifikaci zlepšení nebo zhoršení výkonnosti systému.

Definice odezvy závisí na kontextu.

Response Time (cont'd)

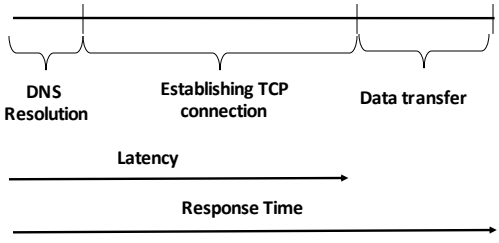
- **Order a Book Example:** the amount of time from placing our order until receiving confirmation



Na úrovni business transakcí se u eshopu pro prodej knih může odezva definovat jako čas nutný pro zaslání objedávky a obdržení potvrzení.

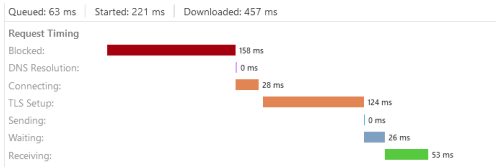
Response Time (cont'd)

- **HTTP Request Example:** Time elapsed from sending an HTTP request until receiving the last byte of the response



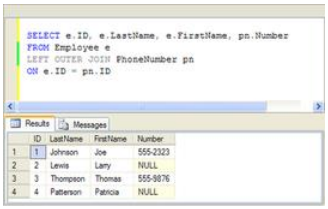
Na úrovni protokolu HTTP (Hyper Text Transfer Protocol) lze odezvu definovat jako čas, který uplynul od odeslání HTTP požadavku, do přejetí **posledního bytu** HTTP odpovědi.
Latence je definována jako čas, který uplynul od odeslání HTTP požadavku, do přijetí **prvního bytu** HTTP odpovědi

Response Time – denikn.cz



Response Time (cont'd)

- **SQL Statement Example:** Time elapsed from sending an SQL statement until receiving the result



U posuzování výkonnosti relační databáze lze odezvu definovat jako čas pro zpracování SQL dotazu a zaslání odpovědi.

Little's Law

$$L = \lambda W$$

L – average number of items in the queuing system
W – average waiting time for an item in the system
λ – average arrival rate

Source: http://web.mit.edu/~sgraves/www/papers/Little%27s%20Law-Published.pdf

John Dutton Conant Little (narozen 21.února 1928) je emeritním profesorem na MIT
V roce 1961 definoval zákon, které se nazývá Little's Law
Formule Littlova zákona umožňuje vypočítat, průměrný počet požadavků, které se nacházejí v systému (čekají ve frontě nebo jsou právě zpracovány) za předpokladu, že jsou do systému zasílány s frekvencí lambda a zpracování každého z nich trvá W
Littlův zákon je možné aplikovat v jakékoli oblasti, ve které se vyskytují fronty
Upozornění: Littlův zákon pracuje s průměrnými hodnotami

Little's Law – Shopping Mall

- Suppose there is one cashier in a shopping mall. Let us assume that one customer per minute arrives to the cash desk and it is served in 10 minutes. How many customers on average stand in the queue?

- λ = 1 customer / min
- W = 10 min
- L = 10 customers

Source: https://www.scienceabc.com/humans/queuing-theory-how-to-choose-a-faster-queue-at-the-grocery-shop-the-science-of-queues.html

Little's Law – Wine Archive

- How long will a bottle of wine on average last in the wine rack?
  - Capacity: 240 bottles
  - Average utilisation: 2/3
  - 8 bottles supplied every month

- L = 160 bottles
- λ = 8 bottles / month
- W = 160 / 8 = 20 months

Little's Law – Web Site

- Suppose https://ct24.ceskatelevize.cz/ is opened by users every 5 seconds on average. The average visit duration is 15 minutes. How many users are working with the site on average?

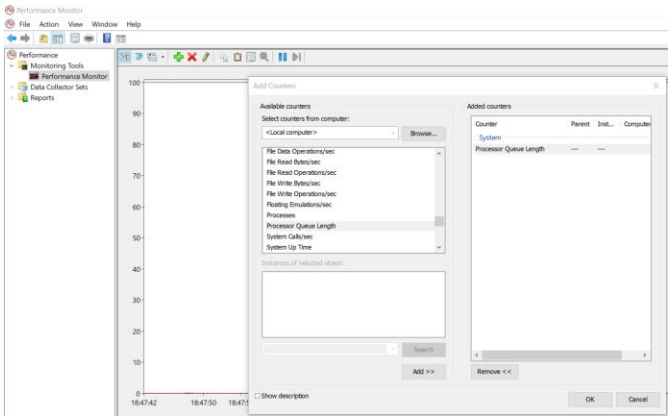
- λ = 12 users / min
- W = 15 min
- L = 180 users

Source: https://ct24.ceskatelevize.cz/

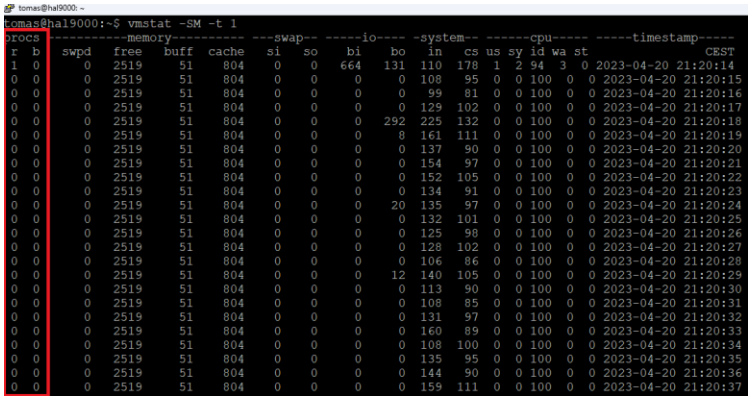
Queues in Software Systems

- System-level queues:
  - CPU queue
  - Disk queue
  - Network card queue
  - Keyboard
- Application-level queues:
  - Producer-consumer queues in messaging systems
    - IBM MQ, RabbitMQ, ActiveMQ
  - Request queues in Tomcat and Nginx

CPU Queue – Windows - perfmon



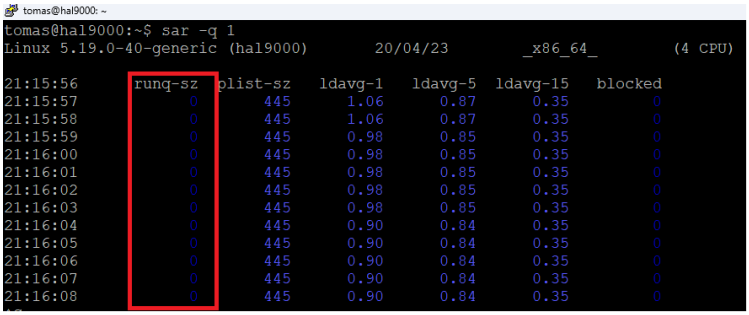
CPU Queue – Linux - vmstat



```
tomas@hal9000:~$ vmstat -SM -t 1
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu----- -----timestamp-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st                 CEST
 1  0      0   2519     51    804    0    0   664   131  110  178  1  2 94  3  0 2023-04-20 21:20:14
 0  0      0   2519     51    804    0    0     0     0  108   95  0  0 100  0  0 2023-04-20 21:20:15
 0  0      0   2519     51    804    0    0     0     0   99   81  0  0 100  0  0 2023-04-20 21:20:16
 0  0      0   2519     51    804    0    0     0     0  129  102  0  0 100  0  0 2023-04-20 21:20:17
 0  0      0   2519     51    804    0    0     0   292  225  132  0  0 100  0  0 2023-04-20 21:20:18
 0  0      0   2519     51    804    0    0     0     8  161  111  0  0 100  0  0 2023-04-20 21:20:19
 0  0      0   2519     51    804    0    0     0     0  137   90  0  0 100  0  0 2023-04-20 21:20:20
 0  0      0   2519     51    804    0    0     0     0  154   97  0  0 100  0  0 2023-04-20 21:20:21
 0  0      0   2519     51    804    0    0     0     0  152  105  0  0 100  0  0 2023-04-20 21:20:22
 0  0      0   2519     51    804    0    0     0     0  134   91  0  0 100  0  0 2023-04-20 21:20:23
 0  0      0   2519     51    804    0    0     0    20  135   97  0  0 100  0  0 2023-04-20 21:20:24
 0  0      0   2519     51    804    0    0     0     0  132  101  0  0 100  0  0 2023-04-20 21:20:25
 0  0      0   2519     51    804    0    0     0     0  125   98  0  0 100  0  0 2023-04-20 21:20:26
 0  0      0   2519     51    804    0    0     0     0  128  102  0  0 100  0  0 2023-04-20 21:20:27
 0  0      0   2519     51    804    0    0     0     0  106   86  0  0 100  0  0 2023-04-20 21:20:28
 0  0      0   2519     51    804    0    0     0    12  140  105  0  0 100  0  0 2023-04-20 21:20:29
 0  0      0   2519     51    804    0    0     0     0  113   90  0  0 100  0  0 2023-04-20 21:20:30
 0  0      0   2519     51    804    0    0     0     0  108   85  0  0 100  0  0 2023-04-20 21:20:31
 0  0      0   2519     51    804    0    0     0     0  131   97  0  0 100  0  0 2023-04-20 21:20:32
 0  0      0   2519     51    804    0    0     0     0  160   89  0  0 100  0  0 2023-04-20 21:20:33
 0  0      0   2519     51    804    0    0     0     0  108  100  0  0 100  0  0 2023-04-20 21:20:34
 0  0      0   2519     51    804    0    0     0     0  135   95  0  0 100  0  0 2023-04-20 21:20:35
 0  0      0   2519     51    804    0    0     0     0  144   90  0  0 100  0  0 2023-04-20 21:20:36
 0  0      0   2519     51    804    0    0     0     0  159  111  0  0 100  0  0 2023-04-20 21:20:37
```

Při použití nástroje vmstat sledujeme frontu CPU ve sloupci r. Ten ukazuje součet bežících a čekajících procesů na CPU. Pokud tato hodnota je větší než počet procesorů (jader) po „delší“ dobu, znamená to, že je procesor saturován.

CPU Queue – Linux - sar



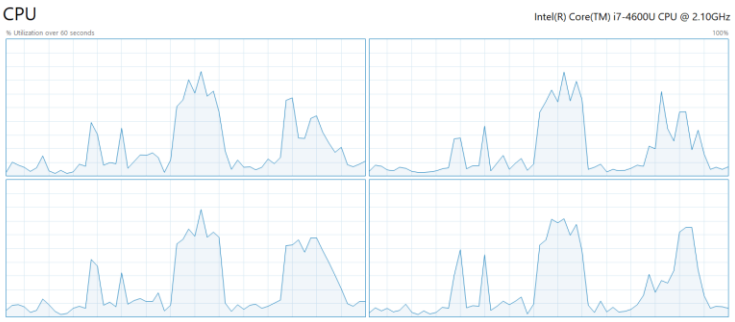
Utilisation

- **Time-based definition:** How busy a resource is during a time interval (e.g. CPU, disk I/O)
- **Capacity-based definition:** The extent to which the capacity of a resource was used during a time period (e.g. physical memory, disk space)

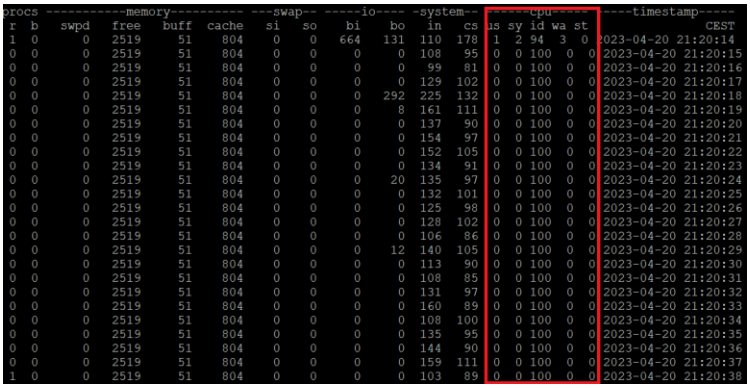
Využití (utilizaci) hardwarového či softwarového zdroje (prostředku) lze definovat dvěma způsoby:

1. Časová závislost – procento aktivity zdroje během stanoveného časového intervalu
2. Kapacitní závislost – procento využití kapacity zdroje během stanoveného časového intervalu

CPU Utilisation – Windows



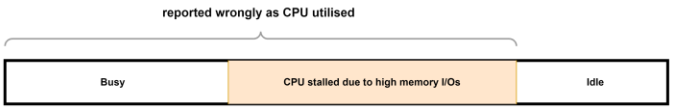
CPU Utilisation – Linux - vmstat



```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu----- -----timestamp-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st                 CEST
 1  0      0   2519     51    804    0    0   664   131  110  178  1  2 94  3  0 2023-04-20 21:20:14
 0  0      0   2519     51    804    0    0     0     0  108   95  0  0 100  0  0 2023-04-20 21:20:15
 0  0      0   2519     51    804    0    0     0     0   99   81  0  0 100  0  0 2023-04-20 21:20:16
 0  0      0   2519     51    804    0    0     0     0  129  102  0  0 100  0  0 2023-04-20 21:20:17
 0  0      0   2519     51    804    0    0     0   292  225  132  0  0 100  0  0 2023-04-20 21:20:18
 0  0      0   2519     51    804    0    0     0     8  161  111  0  0 100  0  0 2023-04-20 21:20:19
 0  0      0   2519     51    804    0    0     0     0  137   90  0  0 100  0  0 2023-04-20 21:20:20
 0  0      0   2519     51    804    0    0     0     0  154   97  0  0 100  0  0 2023-04-20 21:20:21
 0  0      0   2519     51    804    0    0     0     0  152  105  0  0 100  0  0 2023-04-20 21:20:22
 0  0      0   2519     51    804    0    0     0     0  134   91  0  0 100  0  0 2023-04-20 21:20:23
 0  0      0   2519     51    804    0    0     0    20  135   97  0  0 100  0  0 2023-04-20 21:20:24
 0  0      0   2519     51    804    0    0     0     0  132  101  0  0 100  0  0 2023-04-20 21:20:25
 0  0      0   2519     51    804    0    0     0     0  125   98  0  0 100  0  0 2023-04-20 21:20:26
 0  0      0   2519     51    804    0    0     0     0  128  102  0  0 100  0  0 2023-04-20 21:20:27
 0  0      0   2519     51    804    0    0     0     0  106   86  0  0 100  0  0 2023-04-20 21:20:28
 0  0      0   2519     51    804    0    0     0    12  140  105  0  0 100  0  0 2023-04-20 21:20:29
 0  0      0   2519     51    804    0    0     0     0  113   90  0  0 100  0  0 2023-04-20 21:20:30
 0  0      0   2519     51    804    0    0     0     0  108   85  0  0 100  0  0 2023-04-20 21:20:31
 0  0      0   2519     51    804    0    0     0     0  131   97  0  0 100  0  0 2023-04-20 21:20:32
 0  0      0   2519     51    804    0    0     0     0  160   89  0  0 100  0  0 2023-04-20 21:20:33
 0  0      0   2519     51    804    0    0     0     0  108  100  0  0 100  0  0 2023-04-20 21:20:34
 0  0      0   2519     51    804    0    0     0     0  135   95  0  0 100  0  0 2023-04-20 21:20:35
 0  0      0   2519     51    804    0    0     0     0  144   90  0  0 100  0  0 2023-04-20 21:20:36
 0  0      0   2519     51    804    0    0     0     0  159  111  0  0 100  0  0 2023-04-20 21:20:37
 1  0      0   2519     51    804    0    0     0     0  103   89  0  0 100  0  0 2023-04-20 21:20:38
```

Incorrect CPU Utilisation

- CPU stalls due to intensive memory operations
- Memory significantly slower than CPU

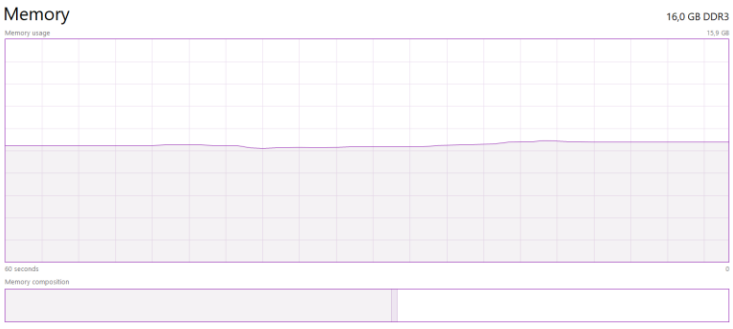


Time Comparison Table

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	3 ns	10 s
Level 3 cache access	10 ns	33 s
Main memory access (DRAM, from CPU)	100 ns	6 min
Solid-state disk I/O (flash memory)	10–100 µs	9–90 hours
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Lightweight hardware virtualization boot	100 ms	11 years

Source: Systems Performance, 2nd edition, Brendan Gregg

Memory Utilisation – Windows



Memory Utilisation – Linux – vmstat

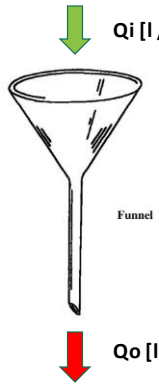
```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu----- -----timestamp-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st                 CEST
 1  0      0   2519     51    804    0    0   664   131  110  178  1  2 94  3  0 2023-04-20 21:20:14
 0  0      0   2519     51    804    0    0     0     0  108   95  0  0 100  0  0 2023-04-20 21:20:15
 0  0      0   2519     51    804    0    0     0     0   99   81  0  0 100  0  0 2023-04-20 21:20:16
 0  0      0   2519     51    804    0    0     0     0  129  102  0  0 100  0  0 2023-04-20 21:20:17
 0  0      0   2519     51    804    0    0     0   292  225  132  0  0 100  0  0 2023-04-20 21:20:18
 0  0      0   2519     51    804    0    0     0     8  161  111  0  0 100  0  0 2023-04-20 21:20:19
 0  0      0   2519     51    804    0    0     0     0  137   90  0  0 100  0  0 2023-04-20 21:20:20
 0  0      0   2519     51    804    0    0     0     0  154   97  0  0 100  0  0 2023-04-20 21:20:21
 0  0      0   2519     51    804    0    0     0     0  152  105  0  0 100  0  0 2023-04-20 21:20:22
 0  0      0   2519     51    804    0    0     0     0  134   91  0  0 100  0  0 2023-04-20 21:20:23
 0  0      0   2519     51    804    0    0     0    20  135   97  0  0 100  0  0 2023-04-20 21:20:24
 0  0      0   2519     51    804    0    0     0     0  132  101  0  0 100  0  0 2023-04-20 21:20:25
 0  0      0   2519     51    804    0    0     0     0  125   98  0  0 100  0  0 2023-04-20 21:20:26
 0  0      0   2519     51    804    0    0     0     0  128  102  0  0 100  0  0 2023-04-20 21:20:27
 0  0      0   2519     51    804    0    0     0     0  106   86  0  0 100  0  0 2023-04-20 21:20:28
 0  0      0   2519     51    804    0    0     0    12  140  105  0  0 100  0  0 2023-04-20 21:20:29
 0  0      0   2519     51    804    0    0     0     0  113   90  0  0 100  0  0 2023-04-20 21:20:30
 0  0      0   2519     51    804    0    0     0     0  108   85  0  0 100  0  0 2023-04-20 21:20:31
 0  0      0   2519     51    804    0    0     0     0  131   97  0  0 100  0  0 2023-04-20 21:20:32
 0  0      0   2519     51    804    0    0     0     0  160   89  0  0 100  0  0 2023-04-20 21:20:33
 0  0      0   2519     51    804    0    0     0     0  108  100  0  0 100  0  0 2023-04-20 21:20:34
 0  0      0   2519     51    804    0    0     0     0  135   95  0  0 100  0  0 2023-04-20 21:20:35
 0  0      0   2519     51    804    0    0     0     0  144   90  0  0 100  0  0 2023-04-20 21:20:36
 0  0      0   2519     51    804    0    0     0     0  159  111  0  0 100  0  0 2023-04-20 21:20:37
 1  0      0   2519     51    804    0    0     0     0  103   89  0  0 100  0  0 2023-04-20 21:20:38
```

Saturation

- The extent to which a resource has queued work because it cannot accept more work
- Increases latency and response time
- Bottleneck = the resource that limits performance

- Saturace je situace, ve které hardwarový či softwarový zdroj (prostředek) odkládá další úlohy, protože již nemá další kapacitu je zpracovat.
- Negativní efekt saturace je prudké zvýšení odezvy a tímto degradace výkonnosti celého systému
- Bottleneck (úzké hrdlo) je zdroj, který byl saturován.
- Nejslabší článek v systému určuje jeho maximální výkonnost

Saturation (cont'd)



If Qi > Qo => saturation

Utilisation Law

$$U = XS$$

U – utilisation of a resource
X – throughput of the resource
S – request service time

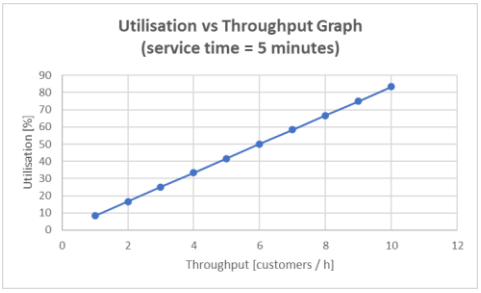
Source: https://homes.cs.washington.edu/~lazowska/qsp/Images/Chap_03.pdf

- Zákon utilizace je odvozen z Little's law.
- Umožnuje vypočítat utilizaci jakéhokoli zdroje, pokud známe jeho propustnost (throughput) a servisní čas (service time)

Utilisation Law (cont'd)

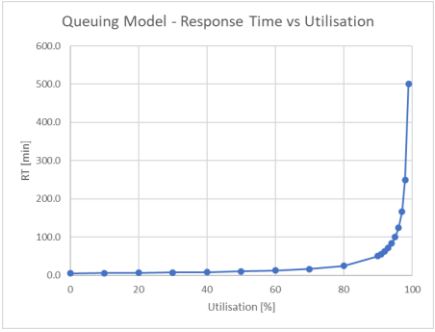
Service time [s]	300

Throughput [customers / h]	Utilisation [%]
1	8
2	17
3	25
4	33
5	42
6	50
7	58
8	67
9	75
10	83



- Tento příklad demonstruje, do jaké míry je pokladní v obchodu vytížena při různých počtech zákazníků, přicházejících k pokladně během každé hodiny
- Průměrný servisní čas pokladní je 5 minut

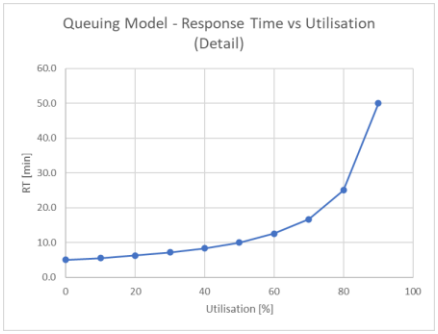
Response Time vs Utilisation



$W = S / (1 - U)$

- Výše uvedený graf vyjadřuje závislost odezvy na utilizaci pro model M/D/1, kterému se ve skutečnosti blíží diskové zařízení
- M/D/1 znamená, že požadavky jsou přicházejí podle Markovskou distribuce (exponenciální), servisní čas je konstantní a v systému existuje jedna služba (prostředek) pro obsloužení požadavků

Response Time vs Utilisation (zoomed-in)



PART 3

How to do performance testing?

Performance Testing Activities

- Problem statement clarification
- Workload modelling
- Performance test design and implementation
- Test data creation
- Monitoring setup

Test preparation activities

Tento a následující slide vyjmenovává zásadní činnosti zátěžového testování

Performance Testing Activities (cont'd)

- Test execution
- Observing the system under load
- Test results analysis
- Reporting to stakeholders
- …

Test execution and reporting activities

How Does It Begin?

Client **Performance tester**

Typický rozhovor mezi zátěžovým testerem a klientem (např.: programátor, architekt, projektový manažer)

Conversation with Client

- **Client:** “We are just about to release a new version of application and we want to do performance testing.”
- **Tester:** “When exactly are you releasing?”
- **Client:** “In one week.”
- **Tester:** “What do you want to get from performance testing?”
- **Client:** “How fast is the application compared to the previous version.”

Conversation with Client (2)

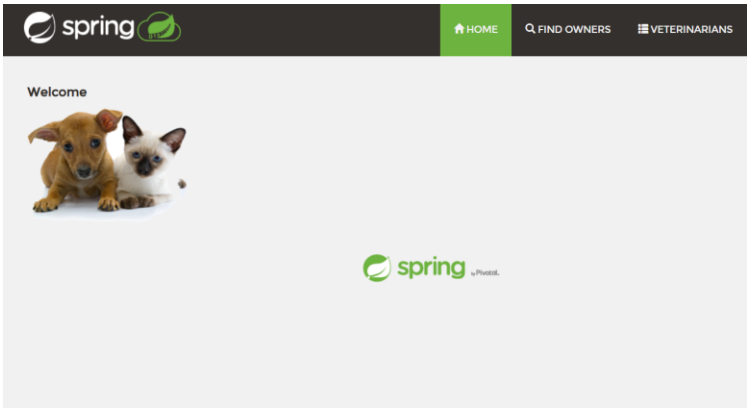
- **Tester:** "Did you performance test the previous version?"
- **Client:** "No. We didn't. I thought you would do it now."
- **Tester:** "Do you realise there's only one week left before the go-live date?"
- **Client:** "Yes, the schedule is very tight, but you can just do simple performance tests."
- **Tester:** "Ugh!… What do you mean by simple tests?"
- ….

Problem Clarification

- What system will be scrutinized for performance?
- What information are you interested in?
- What response time should be for a given throughput? (performance requirements)
- Context-related questions:
  - Clients
  - Time and budget
  - System status
  - Development team & lifecycle

Na začátku zátěžového testování je nezbytné vyjasnit si zadání
Jaký systém se bude testovat?
Jaké informace má zátěžové testování poskytnout? Například, posoudit výkonnost systému pro očekávanou zátěž v produkčním prostředí a kromě toho zjistit, jakou maximální zátěž je systém schopen zvládnout a přitom stále fungovat spolehlivě.
Jak rychle by měl systém provést konkrétní operace (například zaslání objednávky) při dané zátěži?
Zátěžový tester by měl také zjistit informace o situaci, ve které se testovaný systém nachází (kontext), protože ty mohou mít významný dopad na strategii testování

Pet Clinic Web Application

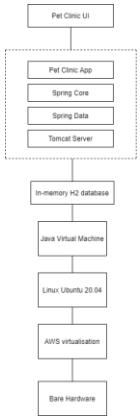


https://github.com/spring-projects/spring-petclinic

Pet Clinic Application

- Simple web-based application for maintaining data about owners, their pets and visits in a veterinary office
- Spring framework hosted on Tomcat
- Operations / functions:
  - Add a new owner
  - Add a new vet to a specific owner
  - Add details about a visit
  - Edit an owner
  - Edit a vet
  - Search for owners
  - View the details of an owner

Pet Clinic Web App – Software Stack



Problem Clarification – Testing Mission

- Evaluate the performance of the Pet Clinic application under a load of 20 users who start working with it every minute

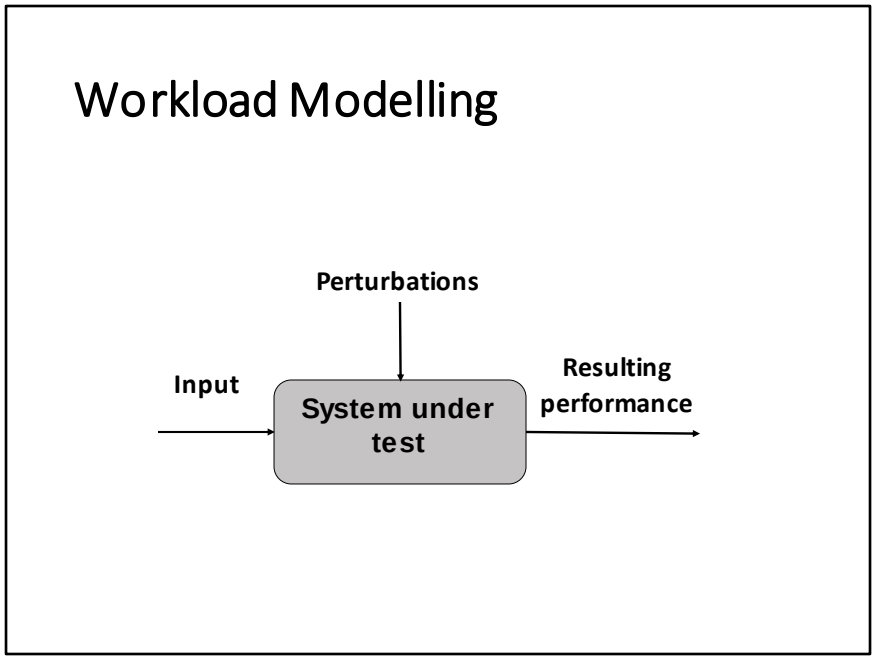
User arrival rate - λ

Problem Clarification – Performance Requirements

- Under the specified load, the Pet Clinic application should complete the operations as follows:
  - Add a new owner: <= 0.5 s in 99% (max = 1 s)
  - Search for an owner: <= 1 s in 99% (max = 2 s)
  - View the details of an owner: <= 0.5 s in 99% (max = 1 s)
  - Add a new visit: <= 1 s in 99% (max = 2 s)

Response Time

Při specifikaci odezev jenotlivých operací je mnohem vhodnější uvést rozsahy hodnot (intervaly), kterým systém má vyhovět než jen jedno číslo. Aritmetický průměr je nevhodnou statistikou, protože ze své podstaty je velmi náchylný na odlehlá pozorování. Medián ve spojení s vyššími percentily (například 90, 95, 99, 99.9 percentily) a maximem je způsob, jakým zachytit rozptyl naměřených hodnot.



Netradiční ukázka ovlivnění výkonnosti diskových polí hlukem (perturbation) od Brendana Gregga (bývalý senior performance engineer v Netflixu) : https://www.youtube.com/watch?v=tDacjrSCeq4

Modelování zátěže je proces, pomocí kterého se snažíme pochopit její charakter a původ. Zjišťujeme, kdo a co se systémem pracuje, a co vše může ovlivnit jeho výkonnost.

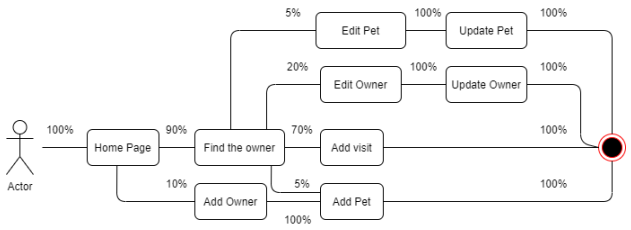
Workload Characterisation (cont'd)

- Who and what uses the system?
  - All actors should be considered
- What operations are performed and how often?
  - Business critical operations
  - Popular operations
  - Performance-intensive operations
- Perturbations (Interferences)
  - External systems
  - Scheduled jobs
  - Multiple virtual tenants sharing the same bare hardware

Při modelování zátěže analyzujeme každou uživatelskou roli a zároveň identifikujeme jaké operace se systémem provádí. Protože operací (funkcí), které systém nabízí, může být celá řada, je nutné z nich vybrat jen určitý vzorek. To provádíme na základě znalostí, jak ta či ona operace je kritická pro business (důvod, proč byl systém vytvořen), populární (často používaná) nebo časově náročná (zpracování velkého množství dat).

Po výběru reprezentativních operací analyzujeme s jakou pravděpodobností je uživatelé provádí.

Workload Model – Pet Clinic App



Pro zachycení zátěžového modelu se často používá diagram chování, který znázorňuje, jak uživatel se systémem komunikuje, aby provedl nějakou činnost. Procento přiřazené ke každé hraně stanovuje s jakou pravděpodobností se uživatel vydá touto cestou.

Performance Test Design

- What tests do we carry out in order to fulfil our mission?
- Performance test types
  - Load test
  - Scalability test
  - Stress test
  - Endurance test

Existuje několik typů zátěžových testů:
Load test – slouží pro stanovení výkonností pod stabilní zátěží
Scalability test (škálovací test) – slouží především pro získání informace, jakou kapacitu systém má. Zjišťujeme jakou maximální zátěž je systém schopen zvládnout a přitom stále fungovat dostatečně rychle.
Stress test – slouží pro zjištění, co se se systémem stane, pokud buden vystaven nadměrné (výjimečně) zátěži. Zkoumáme, kdy se začnou vyskytovat chyby, čeho se budou týkat, jak dlouho systém vydrží nadměrnou zátěž než totálně spadne a jestli je schopen se zotavit potom, co se zátěž sníží na normální úroveň.
Endurance test – slouží pro zjištění stability výkonnosti systému v čase. Zátěž se generuje po dobu několika hodin nebo dnů. Typickým problémem, který se objeví v tomto testu, je memory leak (systém spotřebovává více a více paměti a neuvolňuje ji zpět).

Closed Workload Model

- Arrival rate is a function of throughput
- System controls maximum concurrency level

Open Workload Model

- Arrival rate is independent of throughput
- No virtual users / threads set in advanced
- Majority of public web applications

- Pet Clinic implements the open workload model

Performance Test Implementation

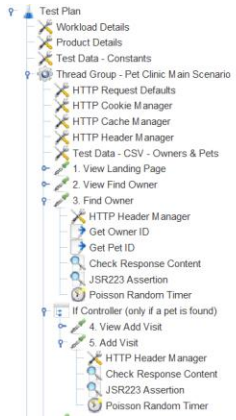
- Converting a workload model and test design ideas into a script that is performed automatically by a load test tool
- Load test tools:
  - Commercial (LoadRunner, WebLOAD, NeoLoad)
  - Open source (Apache JMeter, Gatling, k6, Locust)
- Load test tool is just the tip of an iceberg!

Dostáváme se do bodu, ve kterém musíme koncepční návrh zátěžového testu převést do podoby skriptu, který se spustí v nějakém zátěžovém nástroji.
Zátěžovým nástrojem nemusí být vždy softwarový nástroj. Může to být skupina lidí, která bude dostávat pokyny, jak mají se systémem pracovat. Tento přístup lze použít pro malé zátěže, kdy počet lidí nebude velký. Jeho nevýhoda je v nemožnosti přesně opakovat sekvenci kroků v případě, pokud se objeví nějaký problém. Z tohoto důvodu se pro vytvoření zátěže používají softwarové nástroje. Jejich základní rozlišení je na komerční a volně dostupně.

Load Test Tool - Apache JMeter

- Java-based, free and open-source load generation tool
- Suitable for HTTP(S), FTP, SMTP, TCP, JDBC, MongoDB, shell scripts, …
- Scripts stored as XML files
- Current version 5.4.1
- Out-of-box bundle missing advanced graphs
- JMeter Plugins enhance JMeter functionalities
- JMeter cloud solutions: Blazemeter, Octoperf, flood IO
- Twitter: @ApacheJMeter, @jmeter_plugins

Test Implementation – Pet Clinic



Test Data Population

- Data amount should be close to reality
- Diversity and accuracy of data is important
- Some must be unique, some data can be recycled

Kvalita a množství testovacích dat jsou pro zátěžové testování velmi důležité. Pokud databáze neobsahuje žádná data nebo jen malé množství, protom výsledky testů mohou být do značné míry zkreslené.

Test Data Population – Pet Clinic

- **1000 owners and pets populated into the db**

- **Owner's last name** – a random string of 12 characters
- **Address** – constant prefix + random int (1-999)
- **Phone number** – random string of 9 digits
- **Pet's name** – constant prefix + Unix timestamp
- **Pet species** – CSV file with 6 items

Monitoring Setup

- Provides insight into the system under load
- System-level monitoring:
  - CPU, memory, disk, network card
- Application-level monitoring
  - Heap memory
  - Web server
  - Database

Během zátěžového testu je nutné sledovat, jak se systém pod zátěží chová. Obecně řečeno, čím více výkonnostních metrik budeme sledovat, tím budeme lepší šanci objevit případný problém a také najít jeho příčinu.
Mezi základní výkonnostní metriky, které bychom měli sledovat vždy, patří:
Utilizace CPU, operační paměti a diskových zařízení, počet diskových operací čtení a zápis, propustnost síťové karty.
Kromě těchto systémových zdrojů bychom měli sledovat také metriky na vyšší úrovní abstrakce, například specifické metriky webového serveru, databázového serveru a spotřebu a čistění Java heapu.

Monitoring Setup (cont'd)

- Tools:
  - Provided with operating systems
    - Windows: perfmon,
    - Linux: vmstat, iostat, sar
  - Standalone applications:
    - Telegraf + influxdb + Grafana
    - Sitescope
    - Nagios
    - Datadog
    - New Relic
    - AppDynamics

Nástrojů pro monitorování výkonnosti je celá řada. Některé z nich jsou nainstalovány společně s operačním systémem. Některé se instaluji jako samostatné aplikace.

Performance Test Execution

- Activate monitoring
- Start the simulation of a workload model on a computer that is dedicated for performance testing to avoid disturbances
- Actively observe the system under load and in case of a problem, collect as much information about it as possible

Po spuštění zátěžového testu by měl zátěžový tester aktivně sledovat, co se systémem děje, a v případě výskytu něčeho zajímavého by měl začít zjišťovat příčinu problému.

Test Results Analysis

- Depends on the purpose of a performance test
- Was the objective of the performance test met? If not, why?
- Load test: Was the course of response times stable during the test? If peaks occurred, what might the cause have been?
- Interpretation of performance test results is teamwork

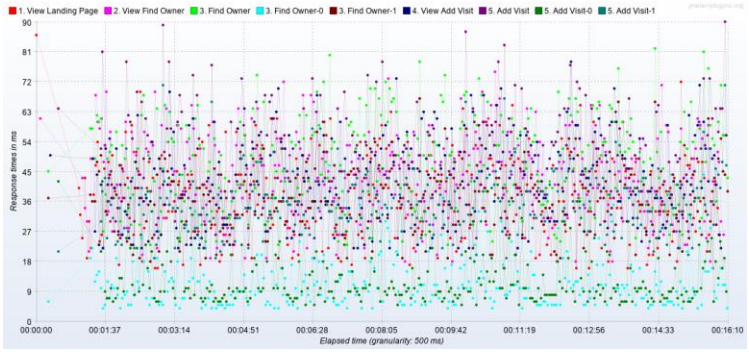
Analýza výsledků zátěžového testu je komplexní aktivita, během které zátěžový tester zpracuje data ze zátěžového nástroje (response times, throughput) a monitorovací data. Všímá si anomálií, které mohou ukazovat na závažný problém. Snaží se hledat jejich příčinu společně s dalšími členy vývojového týmu (programátoři, architekti)

Test Results Analysis – Pet Clinic

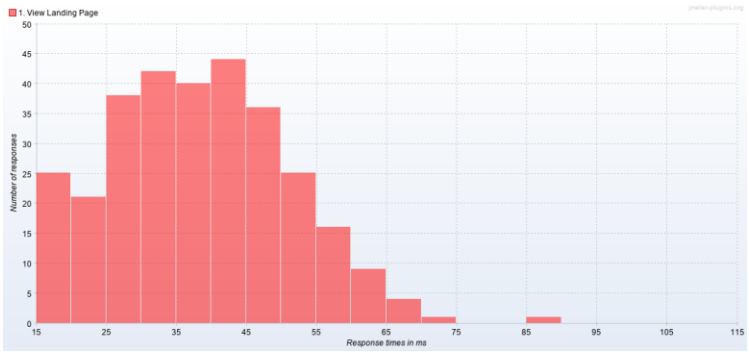
Table with response time statistics – arrival rate of 20 users / min

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB/sec	Sent KB/sec
1. View Landing Page	302	37	37	55	59	65	16	86	0.00%	18.9/min	1.03	0.17
2. View Find Owner	302	43	44	60	65	71	18	75	0.00%	18.9/min	1.21	0.17
3. Find Owner	300	48	50	66	69	78	23	82	0.00%	18.9/min	1.56	0.35
3. Find Owner-0	300	11	10	21	28	35	4	47	0.00%	18.9/min	0.06	0.18
3. Find Owner-1	300	36	36	50	53	61	18	67	0.00%	18.9/min	1.50	0.17
4. View Add Visit	296	42	42	59	63	70	20	78	0.00%	18.9/min	1.63	0.18
5. Add Visit	295	49	47	67	73	83	25	90	0.00%	18.9/min	1.59	0.40
5. Add Visit-0	295	13	10	25	31	42	5	46	0.00%	18.9/min	0.06	0.23
5. Add Visit-1	295	35	35	46	50	66	19	71	0.00%	18.9/min	1.54	0.17

Test Results Analysis – Pet Clinic Response Time Graph



Test Results Analysis – Pet Clinic Histogram



Useful Links

- Scott Barber's User Experience, not Metrics Series: http://www.perftestplus.com/pubs.htm
- Martin Thompson, Performance Testing Java Applications https://www.youtube.com/watch?v=1DuMvpwWHH4
- Martin Thompson, Designing for Performance: https://www.youtube.com/watch?v=d50SxnI1D6Y

Useful Links (cont'd)

- Brendan Gregg, personal web site: https://www.brendangregg.com/index.html
- Brendan Gregg's study materials: https://www.brendangregg.com/books.html
- Apache JMeter: https://jmeter.apache.org/