VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SYSTÉM PRO OVĚŘENÍ MINIMÁLNÍCH POTŘEBNÝCH ZDROJŮ PRO BĚH APLIKACE

SYSTEM FOR VERIFYING THE MINIMUM RESOURCES REQUIRED TO RUN AN APPLICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ ŽÁK**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2022

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

ŽÁK, Jiří. *Systém pro ověření minimálních potřebných zdrojů pro běh aplikace*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Systém pro ověření minimálních potřebných zdrojů pro běh aplikace

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Jiří Žák
20. ledna 2022

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

1 Úvod **2**

2 Rozbor řešené problematiky **3**
2.1 Metody 4
2.2 Testování 5
2.3 Přehled nástrojů 5
2.3.1 Berkeley Packet Filter 5
2.3.2 Rozdíl mezi user spacem a kernal spacem 6
2.3.3 Netem 6
2.3.4 Virtuální počítač s omezenými zdroji 7

Literatura **8**

Kapitola 1

Úvod

Kapitola 2

Rozbor řešené problematiky

Počítačový výkon je účinnost daného systému neboli jak dobře funguje, když se vezmou v úvahu všechny možné aspokty. Hodnocení výkonu počítače je definováno jako proces při kterém se posuzují zdroje a výstupy počítače, aby se zjistilo, jestli systém funguje na optimální úrovni.

Počítačové vyhodnocování je složitý proces. Při vyhodnoconváůní výkonu počítače se k určení využívá řada aspektů.

Hlavní aspekty [2]

- škálovatelnost
- stabilita
- schopnost reagovat
- rychlost
- dostupnost
- propustnost
- doba odezvy
- spolehlivost
- provozuschopnost

Napsat to do vysvetlivek a vysvetlit i zbytek, popsat jeste nejake metriky nize Propustnost. Propustnost je počet jednotek dat, které systém zpracuje za určitou dobu. Doba odezvy. doba odezvy měří, kolik času trvá systému, než zareaguje na dotaz nebo poptávku. Spolehlivost, dostupnost a provozuschopnost (RAS). RAS označuje schopnost softwaru trvale plnit své specifikace; jak dlouho funguje vzhledem k očekávanému množství; a jak snadno se dá opravit nebo udržovat.

Ke zjištění těchto aspektů se využívají speciálně navržené programy, které posuzují relativní výkon daného objektu obvykle spuštěním řady testů. Tento proces se nazývá benchmark. Není jednoduché vytvořit měřítka pro hodnocení výkonu počítaču, protože technologické vlastnosti se stále vývýjí a obměňují. Toto znamené, že benchmark počítačů musí být také neustále vyvíjen a aktualizován.

Metriky k měření

Před každým měřením by mělo být jasně dáno jaké metriky budeme měřit a jakým způsobem se to bude vykonávat. Pokud metriky ani způsob nebude stanoven, tak je měření ztrátou času. Ověření těchto metrik se poté provádí testováním. Svoji důležitou roli hraje také složitost tesovaného softwaru. Mezi základní metriky k měření výkonu systému jsou:

Doba spuštění a doba načítání

Každý software vyžaduje nějaký čas k jeho načtení a zahájení provozu. Koncový uživatelé během této doby obvykle vidí animovaný kurzor nebo načítání obrazovky. Složité programy mohou vyžadovat nastavení a konfiguraci než uživatele pustí do práce. Software například kontroluje závislost s jiným softwarem nebo existenci potřebných souborů. Toto zvyšuje jeho dobu načítání. Doba načítání by proto měla být co nejkratší.

Doba odezvy nebo latence

Doba odezvy označováná jako latence je doba prodlevy mezi okamžikem, kdy uživatel zadá vstup, a okamžikem, kdy daný software na základě tohoto vstupu reaguje. V ideálním připadě by tato doba měla být pro uživatele neposřehnutelná. Tato doba také může záviset na hardaru nebo na rychlosti připojení.

Omezené zatížení nebo špatná škálovatelnost

Aplikace by měly obsluhovat omezený počet zařízení nebo uživatelů bez snížení výkonu softwaru. Škálovatelnost pomáhá zajistit, aby aplikace měla prostředky pro podporu očekávaného počtu uživatelů. Testování výkonu hodnotí software při různých podmínkách zátěže. Nedostatečné zdroje, jako je nedostatečný výpočetní výkon, mohou způsobit špatnou škálovatelnost. Výkon softwaru je však prostředky plus návrh a mnoho problémů se škálovatelností vyplývá ze špatné architektury aplikací. Každá aplikace by měla být napsaná tak, aby se dala v budoucnu škálovat.

Bottlenecks neboli uzká místa

napsat co to znamena v čestine

Každá aplikace přistupuje k omezenému zdrojů, jako třeba výpočetní výkon, paměťový prostor, šířka pásma, propustnost sítě, kapacita diskového úložiště a vstupy/výstupy a služby operačního systému. Úzká místa jsou nedostatky zdrojů, ke kterým dochází pokud software není vhodně navržen nebo pokud systém pro danou aplikaci není dostačující.

2.1 Metody

Vyhodnocování výkonu počítačů je důležitou technologii pro výzkum v oblasti počítačů. Neustálý vývoj počítačů dělá tento úkol stále složitější. Obecný problém rozvoje efektivní vyhodnocovací techniky lze vyjádřit jako hledání nejlepšího kompromisu mezi přesností a rychlostí. Tento kompromis závisí na použití vyhodnocovací metody. Správná efektivita může vést ke snížení nákladů při tvorbě systému pro běh aplikace.

2.2 Testování

Vzhledem k velkému potenciálu hrozeb v souvislosti s výkonem softwaru, mohou různé testy zaměřené na výkon vyhodnitit specifické chování softwaru. Mezi důležité typy testů patří:

- zátěžový
- odolnostní
- spike
- závislostní

Zátěžové testy

Tento typ testů obecně měří pohyb dat, také nazývaný jako propustnost. Zátěživý test pomáhá ověřit zda daný software funguje správně za typických podmínek. Cílem zátěžového testování je zajistit, aby klíčové metriky výkonu, jako propustnost dat za sekundu nebo přístup disku za sekundu, zůstaly přijatelné, když se připojí více zařízení nebo více uživatelů. Pokud testování významně ovlivňuje metriky výkonu, tak je to signál k úpravě softwaru. Případně zjištění maximálního limitu softwaru.

Odolnostní testy

Odolnostní testy jsou charakteristické tím, že se provádí po delčí dobu. Jejich cílem je ověřit, jestli se výkonnostní charakteristiky softwaru v průběhu provádění nemění. Odolnostní testy zachycují nedostatky jako je nedostatečná paměť. Při krátkém testování by se tento problém nemusel vůbec objevit.

Spike testy

Spike testy simulují náhle změny uživatelské zátěže za účelem měření odezvy softwaru.

Závislostní testy

Software zřídka funguje sám o sobě. Většinou spolého na jiné aplikace nebo případně na databázi. Testy závislosti vyhodnocují výkon softwaru při zátěži, ale vztahují se na závislosti, nikoliv jen na software.

2.3 Přehled nástrojů

2.3.1 Berkeley Packet Filter

Berkeley Packet Filter nebo také BPF je technologie, která umožňuje spouštět programy v jádře operačního systému na linuxových systémech. Efektivně a bezpečně rozšíří schopnosti jádra bez nutnosti měnit zdrojový kód jádra. Tato schopnost je často použita ke sledování využití systému či sledování síťového provozu. BPF je k dispozici na většině unixových operačních systémů a extended BPF je také k dispozici pro Microsoft Windows.

Sledování síťového provozu

BPF poskytuje rozhranní k datové lince, což umožňuje odesílání a přijímání paketů spojové vrstvy. BPF podporuje filtrování paketů a umožňuje user space(vysvětlit proč jsem to napsal anglicky) procesům, aby si samy vyfiltrovaly pakety, které chtějí obdržet a které naopak zahodit. Toto může vést ke zvýšení výkonu operačního systému, protože se tím zamezí kopírování nepotřebných paketů.

Verze BPF

BPF má více verzí. Výše zmíněnou extended BPF (eBPF) nebo také classic BPF (cBPF).

extended BPF - eBPF

Extended BPF povoluje spouštět programy v rámci operačního systému a přidávat další funkce za běhu. Operační systém pak zaručuje bezpečnost a efektivitu provádění. Díky to tomu je na eBPF je založeno hned několik projektů. eBPF se používá pro poskytování vysoce výkonných sítí a vyvažování zátěže v moderních datových centrech a cloudových nativních prostředích, získávání podrobných dat o pozorovatelnosti zabezpečení při nízké režii, pomáhá vývojářům aplikací sledovat aplikace, poskytování přehledů pro řešení problémů s výkonem a mnoho dalšího.

classic BPF - cBPF

Classic BPF je jenom přejmenovaná originální verze BPF.

2.3.2 Rozdíl mezi user spacem a kernal spacem

kernel space

Kernel space je prostor kde je uložen kód kernelu a kde je vykonáván.

user space

User space je sada míst, kde běží uživatelské procesy(všechno ostatní kromě jádra). Jádro řídí aplikace v tomto prostoru tak, aby se nepletly mezi sebou. Procesy, které běží v user spacu mají omezenou část paměti a také nemají přístup ke kernel spacu. Procesy běžící v user spacu mohou přistupovat do kernel spacu jedině přes rozhranní vystavenný kernel spacem - systémová volání. Pokud proces provede systémové volání, tak se do jádra odešle systémové přerušení a jádro poté obslouží příslušný proces. Jádro pokračuje ve své práci až po dokončení přerušení.

2.3.3 Netem

[1] Netem je softwarová utilita pomocí které lze vykonávat síťovou emulaci. Síťovou emulaci lze využít k omezení internetových zdrojů pro měřenou aplikaci pro lepší testování. Netem umí simulovat zpomalení, ztrátu, poškození nebo duplikaci paketů a případně změnu pořadí paketů. Netem je řízen pomocí nástroje 'tc', který se ovládá pomocí příkazové řádky. Jeho aktuální distribuce je multiplatformě podporována.

Zpoždění paketů

Tento příkaz přidá zpoždění všech paketů o 100 milisekund. Toto nastavení je ještě omezeno rozlišením jádra.

```
$ tc qdisc add dev <interface> root netem delay 100ms
```

Lze i nastavit zpoždění pomocí rozdělení. Příklad s normalním rozdělením:

```
$ tc qdisc change dev <interface> root netem delay 100ms 20ms distribution normal
```

Ztráta paketů

Tento příkaz zajistí ztrátu paketů. 1 paket z 1000 je zahozen. nejnižší možná nastavitelná hodnota je 0.0000000232

```
$ tc qdisc change dev <interface> root netem loss 0.1%
```

Duplikace paketů

Příkaz pro duplikaci paketů je podobný jako příkaz pro ztrátu paketů.

```
$ tc qdisc change dev <interface> root netem duplicate 1%
```

Poškození paketů

Tímto příkazem lze nastavit poškození paketů. Ten zavede bitovou chybu v náhodném offsetu v paketu.

```
$ tc qdisc change dev <interface> root netem corrupt 0.1%
```

Přeuspořádání paketů

Existují dva způsoby jak nastavit přeuřpořádání paketů. První způsob je pomocí metody gap, která používá přednastavenou sequekci a změní pořadí každého Ntýho paketu.

```
$ tc qdisc change dev <interface> root netem gap 5 delay 10ms
```

Druhá metoda je reorder způsobí, že určité procento paketů se špatně seřadí.

```
$ tc qdisc change dev <interface> root netem delay 10ms reorder 25% 50%
```

Obnovení rozhranní

Tento příkaz smaže všechno nastavení na rozhranní. Rozhranní bude poté fungovat jako by na něm žádné nastavení nebylo.

```
sudo tc qdisc del dev <interface> root
```

2.3.4 Virtuální počítač s omezenými zdroji

Pro zajištění více testovacích platforem je jedna z Bohužel výpočetní hardware bude stejný pro všechny virtuální platformy i když bude omezený.

Literatura

[1] AUTORA, P. *Netem* [online]. [cit. 2022-10-02]. Dostupné z: https://wiki.linuxfoundation.org/networking/netem.

[2] Black, R. *23 software development metrics to track today* [online]. [cit. 2022-10-02]. Dostupné z: https://searchsoftwarequality.techtarget.com/tip/23-software-development-metrics-to-track-today.