Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Metody testování virtualizační technologie superpočítače CMU

Bakalářská práce

Autor: Jiří Cehák

Vedoucí práce: Ing. Jan Fesl

České Budějovice 2015

Zadání bakalářské práce:

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Jiří Cehák
.....
(jméno, příjmení, tituly)

Obor – zaměření studia: Aplikovaná informatika
.....

Katedra/ústav, kde bude práce vypracovávána:.UAI PRF JU
.....

Školitel: Ing. Jan Fesl
.....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF:
.....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Metody testování virtualizační technologie superpočítače CMU
.....

.....

Cíle práce:

Účelem zadávané práce je vytvoření metodiky testování efektivity virtuálních počítačů a její praktické ověření. V úvodní fázi se zaměřte na existující způsoby možného testování virtuálního počítače a jejich evaluaci, později se tuto metodiku pokuste rozšířit na testování distribuovaných systémů. Při tvorbě metodiky se snažte dbát na univerzálnost použití, tj. minimalizovat vliv použitého hypervizoru. Bude-li to použitá technologie umožňovat, prozkoumejte vliv a možnosti pokročilé virtualizace pomocí technologií Intel VT-x, popřípadě AMD-V.

Základní doporučená literatura:

1. Anala, M. R. and G. Shobha (2012). "Comparative Study of Application Performance on Virtual Machine and Physical Machine." 2012 Ieee International Conference on Computational Intelligence and Computing Research (Iccic): 359-364

2. Ye, K. J., et al. (2014). "Virt-B: Toward Performance Benchmarking of Virtual Machine Systems." Ieee Internet Computing 18(3): 64-72.

3. Xiao, P., et al. (2013). "Virtual machine power measuring technique with bounded error in cloud environments." Journal of Network and Computer Applications 36(2): 818-828.

4. Wen, C. J., et al. (2013). "System Power Model and Virtual Machine Power Metering for Cloud Computing Pricing." 2013 Third International Conference on Intelligent System Design and Engineering Applications (Isdea): 1379-1382.

Financování práce: .....

Vedoucí práce: .....podpis: .....

U externích vedoucích fakultní garant práce: .....podpis: .....

Garant oboru bak. studia, pokud je obor zajišťován jinou katedrou/ústavem, než ze které je školitel (nepožaduje se u oboru biologie): .....podpis: .....

Vedoucí katedry/ústavu, kde bude práce vypracovávána: .....podpis: .....

Případný souhlas vedoucího ústavu AV: .....podpis: .....

V Českých Budějovicích dne ..25.2. 2015.....Podpis studenta: .....

Bibliografické údaje
Jiří Cehák 2015: Metody testování virtualizační technologie superpočítače cmu.
[Methods for testing virtualization technology of supercomputer caled CMU. Bc. Theis, in Czech.] – 43 p., Faculty of Science, The University of South Bohemia, České Budějovice, Czech Republic.

Anotace:
Tato práce pojednává o metodách testování virtualizační technologie. Testování probíhá na virtuálních a fyzických strojích. Dále tato práce vytváří návod, jak testovat superpočítač ohledně výkonu virtualizační technologie. Testování se rozděluje na testování výkonu procesoru, testování rychlosti pamětí, zápisu a čtení disku a na konec rychlosti komunikace po síti. V závěru práce naleznete porovnání těchto strojů.

Abstract:
This work deals about testing of the virtualization technology. The testing proccess is realized on the physical and the virtual machine as well. The main idea behind this work stands in the testing of the cpu power, memory access throughput for reading and writing and the network utilization bandwidth. The end of this thesis deals about comparison of both machine types .

Prohlašuji, že svoji bakalářskou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích 20.4.2015

Podpis:

Poděkování:
Chtěl bych poděkovat panu Ing. Janu Feslovi za rady a pomoc při psaní této bakalářské práce.

Obsah

Úvod

Účelem této bakalářské práce je vymyšlení způsobu testování a kvantifikování superpočítače v závislosti na provozované virtualizační technologii. Pro tento účel jsme použili technologii od firmy Microsoft. Konkrétní Hyper - V. Jedná se o spuštění více virtuálních počítačů za pomoci virtualizace na jednom superpočítači (nevíme, kolik jich přesně budeme moci spustit) a my na těchto virtuálních počítačích budeme měřit takzvaný overhad, což znamená určitý poměr výkonu virtualizační technologie oproti fyzické. Předpokladem pro měření je určitá režie (overhad) virtuálních počítačů. Měření je dále závislé na druhu operací, které bude virtuální stroj provádět. Režie pro výpočet na procesoru může být jiná a režie pro přenos po síti může být také jiná.

Cíle práce

Tato bakalářská práce pojednává o vytvoření schématu, návodu, metody testování superpočítače CMU. Jelikož tento superpočítač by měl v budoucnu sloužit ke zlepšení výuky, zajímá nás, jak moc určitý hardware (případně i jiné konfigurace) můžeme zatížit virtuálními systémy oproti fyzickým systémům. Předpokládali jsme, že virtualizovaný systém bude zatěžovat ještě takzvané řízení virtualizace. Pro rozdíl výkonů mezi virtualizovaným systémem a fyzickým systémem jsme si stanovili heslo overhad, které bude stěžejní v této bakalářské práci. Pomocí metodiky testování otestujeme virtuální počítač vs. fyzický počítač a změříme overhady. Cílem této práce bude metodika výpočtu overhadu při použití virtualizovaného operačního systému a následné vyhodnocení.

Metodika

Předpokladem pro tuto práci je pochopení virtualizační technologie. S postupným nabýváním znalostí o virtualizační technologii bylo nutné se soustředit na implementaci technologie na superpočítači CMU a následně se zaměřit na hypervizor 1. typu s použitou technologií Hyper-V od firmy Microsoft [3].

Dalším krokem bylo zjistit, jak je vlastně možné tuto technologii otestovat. Následovalo hledání materiálů a návodů, jak se testuje virtualizační technologie. Bohužel moc lidí se touto problematikou nezabývalo, a pokud ano, tak vždy testovali konkrétní infrastrukturu a většinou na úrovní aplikace. Aplikace navíc byla psaná v programovacím jazyce, který přistupuje k hardwaru pomocí virtual machine, což znamená ještě další virtualizaci na úrovni software. Za pomoci těchto zjištění jsme usoudili, že bude nejlepší zajistit testy pomocí benchmarků, které nebudou tuto softwarovou virtualizaci používat (další overhad ovlivněný programovacím jazykem). Hledání benchmarků ztížil fakt, že bylo nutné použít multiplatformní testovací program. Benchmarky jsme si rozdělili na benchmark procesoru, benchmark pamětí, benchmark I/O operací a nakonec benchmark komunikace po síti. Toto rozdělení jsme si stanovili na základě operací, které běžně fyzický počítač zpracovává. Dalším požadavkem na testovací program bylo ovládání benchmarku z příkazové řádky, díky které můžeme testovací program ovládat na dálku „pouhým“ posíláním příkazů a můžeme tento testovací program spustit na více strojích takřka najednou.

Posledním krokem práce je samotné testování a vyhodnocení výsledků.

Současný stav problematiky

Testování počítačů (benchmarky) existuje už velmi dlouho, avšak testování virtualizační technologie nad distribuovaným systémem v podstatě neexistuje. V doporučené literatuře k bakalářské práci se můžeme dočíst o pokusu testovat podobný systém, nicméně se jedná pouze o testy konkrétního systému, konkrétní virtualizační technologie. V mnoha případech se jedná o technologii založenou na platformě Linux, konkrétně na virtualizéru Xen.[1][2]

Teoretická část

Superpočítač

Pojmem „Superpočítač“ značíme zapojení jednotlivých počítačů do distribuované počítačové sítě pomocí domény. Superpočítač obsahuje CORE, NODE, NAS, …

CORE

- Jedná se o server, který slouží k řízení a přístupu k výkonnostním uzlům a k datovému úložišti. Tento server je určen pro správu a je možné ho „neomezeně“ rozšířit o další servery

NODE

- Zde se jedná o server, který je určen k výkonnostním účelům. Je určen pro výpočty a chod virtuálních počítačů. I tento server lze „neomezeně“ rozšiřovat o další uzly

NAS

- Tento server je použit jako síťové úložiště. Zde jsou uloženy obrazy virtuálních počítačů a je zde i úložiště virtuálních počítačů. Stejně jako v předchozích serverech je lze „neomezeně“ rozšířit o další uzly.

Pomocí této struktury jsme schopni přidat takřka jakýkoliv uzel. Pokud by nedostačoval výkon jednotlivých uzlů, není problém přikoupit další hardware. Pouze by musel být přidán do domény, pomocí které jsou veškeré uzly spravovány. Jestliže bychom potřebovali nový uzel například pro zpracování grafických operací, jediné co by bylo potřeba, je připojit tento nový uzel do domény. Pojem superpočítač můžeme tedy definovat jako „živoucí personalizovaný stroj, který nemusí nikdy zastarat“.

Overhad (režie)

V této kapitole je vysvětleno, co vlastně znamená overhad a jak jsme se k tomuto výrazu dostali. Dále si vysvětlíme overhady jednotlivých částí a ke konci jak overhad vypočítat.

Základní myšlenkou pro tuto práci je porovnání výše zmíněných overhadů. Jedná se v podstatě o jakýsi poměr výkonu počítače fyzického oproti počítači

virtualizovanému. Předpokladem je, že pod virtuálním počítačem existuje virtualizační infrastruktura, která by měla mít vliv na výkon virtuálního počítače. A právě tento rozdíl výkonů jednotlivých strojů je overhad.

Bylo důležité si uvědomit, co takový počítač vlastně dělá a jak ho můžeme otestovat. V úvahu připadalo testovat různé výpočty, programy či další operace, které počítač zpracovává. Zjistili jsme, že tyto operace mají jedno společné. Zaměřují se na vytížení určitých periferií počítače a jejich kombinací. Konkrétně se jedná o operace na CPU, RAM (čtení, zápis), dále práce s diskovými operacemi I/O a v neposlední řadě komunikaci po síti NET. Zaměřili jsme se na tyto operace, protože pokud fyzický počítač neobsahuje určitý hardware, který se používá pro specializované účely, tak tyto operace pokrývají veškerý chod klasického počítače.

Pro výpočet overhadu jsme vycházeli z rovnice, kde overhad je poměr overhadu virtuálního vs. fyzického.

$$O = \frac{VPC}{FPC}$$

O = Overhad

VPC = virtuální počítač

FPC = fyzický počítač

Pokud bychom vycházeli pouze z této rovnice, můžeme zjistit, jaký overhad má jaký VPC, ale nezjistíme už z jakých operací. Pro tento účel bylo nutné základní rovnici obohatit o další prvky.

$$O = \frac{O_{CPU_{VPC}}}{O_{CPU_{FPC}}} + \frac{O_{RAM_{FPC}}}{O_{RAM_{VPC}}} + \frac{O_{I/O_{FPC}}}{O_{I/O_{VPC}}} + \frac{O_{NET_{FPC}}}{O_{NET_{VPC}}}$$

Touto rovnicí bereme v úvahu i to, zda se jedná o overhad při výpočtu na procesoru, nebo zda se jedná o overhad při diskových operacích atd.

Bohužel pro nás, každý počítač bývá vytížen jiným způsobem. Pokud počítač bude provádět více diskových operací a jiný počítač bude provádět více výpočtů na procesoru, bude vytížení jednotlivých prostředků na každém počítači rozdílné, přestože se celkové vytížení jednotlivých počítačů bude shodovat. Z této úvahy plyne, že rovnice uvedená výše nemůže fungovat. Rovnice by fungovala pouze v případě, pokud by všechny operace vytěžovaly počítač stejně. Po bližším pohledu na běh počítače jsme přidali ještě další proměnnou do rovnice. Tato proměnná nám zajišťuje, že bude v rovnici bráno v potaz to, pro co je počítač převážně určen. Zda se jedná

převážně o počítač určený pro výpočty, komunikaci po sítí či ukládání dat. Tato proměnná se nazývá váha. Může být pro každou operaci odlišná, nicméně její součet musí být vždy roven 1. Rovnice tedy vypadá následně:

$$O = \frac{O_{CPU_{VPC}}}{O_{CPU_{FPC}}} \times w_{CPU} + \frac{O_{RAM_{FPC}}}{O_{RAM_{VPC}}} \times w_{RAM} + \frac{O_{I/O_{FPC}}}{O_{I/O_{VPC}}} \times w_{I/O} + \frac{O_{NET_{FPC}}}{O_{NET_{VPC}}} \times w_{NET}$$

Výše zmíněnou rovnicí jsme zjistili už konečný overhad jednoho počítače. Pomocí vah, které si můžeme nastavit dle libosti (součet musí být jedna), můžeme vytvořit takřka jakýkoliv počítač. Pokud budeme mít například počítač provádějící více diskových operací a nebude využívat procesor, tak procesoru stanovíme váhu 0,1 a zápisu a čtení na disk váhu třeba 0,6. Zbytek rozdělíme mezi zápisem a čtením operační paměti a komunikaci po síti.

Overhad více virtuálních počítačů

Po předchozí úvaze bychom mohli říci, že overhad bude součtem overhadů virtuálních počítačů.

$$O_C = O_{VPC_1} + O_{VPC_2} + O_{VPC_3} + \cdots + O_{VPC_n}$$

S tímto výpočtem už bychom se mohli spokojit. U dílčích overhadů jsou již i váhy operací započítány. Pokud se však na problém budeme koukat ze širšího hlediska, zjistíme, že i každý virtuální počítač může mít určitou váhu a může mít i jiný virtuální hardware. Z tohoto důvodu jsme definovali ještě další proměnné a rovnici výše rozepsali a upravili do podoby, která sčítá dílčí overhady jednotlivých operací s určitou váhou a započítává, kolik prostředků nám daný proces využívá (počet jader procesoru, velikost ram) oproti tomu, kolik jich máme k dispozici na jednotlivých uzlech. Tento součet provedeme pro všechny virtuální stroje na uzlu.

$$O_{NODE_{CPU}} = \sum_{i=1}^{N} \left(\frac{O_{CPU_{i_v}}}{O_{CPU_{i_s}}} \times w_{CPU_i} \times \frac{CPU_i}{CPU_{NODE}} \right)$$

Součet overhadů procesoru pro všechny virtuální počítače.

$$O_{NODE_{RAM}} = \sum_{i=1}^{N} \left(\frac{O_{RAM_{i_s}}}{O_{RAM_{i_v}}} \times w_{RAM_i} \times \frac{RAM_i}{RAM_{NODE}} \right)$$

Součet overhadů operační paměti pro všechny virtuální počítače.

$$O_{NODE_{I/O}} = \sum_{i=1}^{N} \left(\frac{O_{I/O_{i_s}}}{O_{I/O_{i_v}}} \times w_{I/O_i} \times \frac{I/O_i}{I/O_{NODE}} \right)$$

Součet overhadů diskových operací pro všechny virtuální počítače.

$$O_{NODE_{NET}} = \sum_{i=1}^{N} \left(\frac{O_{NET_{i_s}}}{O_{NET_{i_v}}} \times w_{NET_i} \times \frac{NET_i}{NET_{NODE}} \right)$$

Součet overhadů komunikace po síti pro všechny virtuální počítače.

Konečná rovnice pro overhad určitého výpočetního uzlu (NODE) je součtem těchto overhadů.

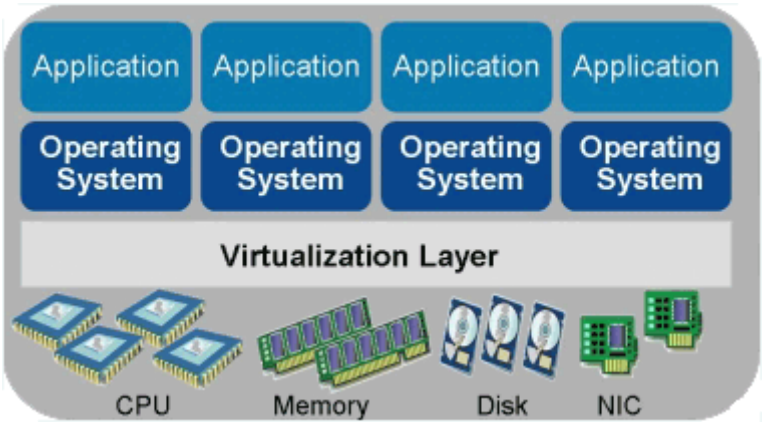
$$O_{NODE_n} = O_{n_{CPU}} + O_{n_{RAM}} + O_{n_{I/O}} + O_{n_{NET}}$$

Pokud bychom chtěli overhad spočítat pro celý systém, tak potom už je možné pouze sečíst overhad na jednom uzlu s dalšími uzly.

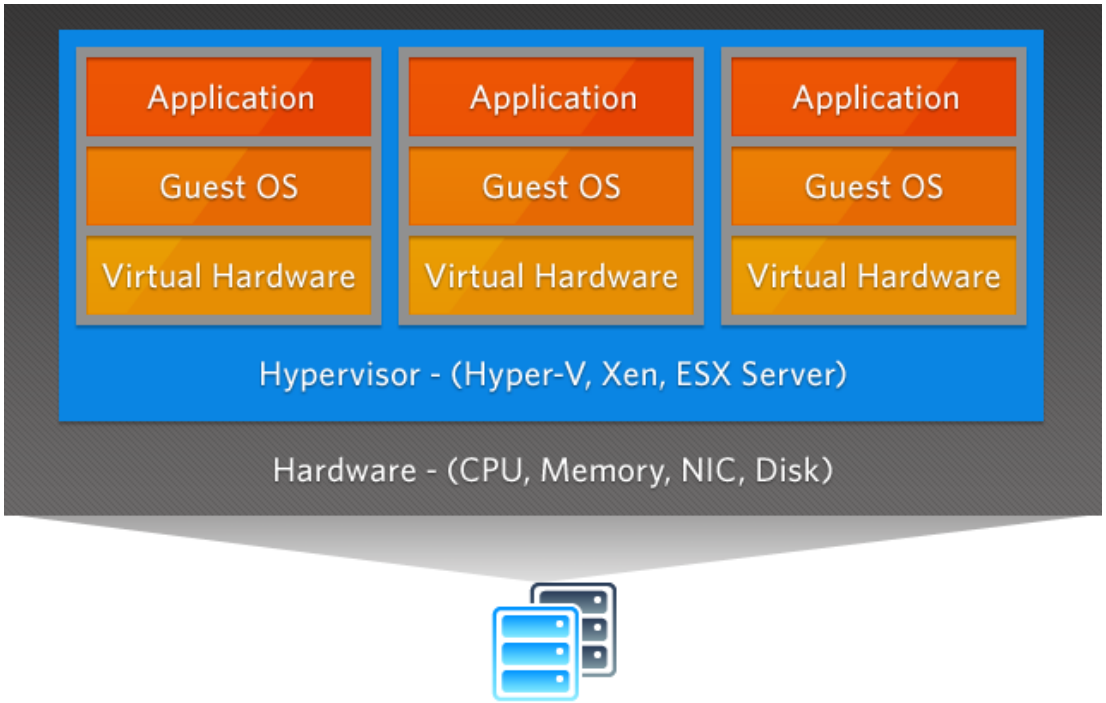
Virtualizace

O co se konkrétně jedná a k čemu slouží

Podstatou virtualizace je možnost provozu více oddělených operačních systémů najednou na jediném hardwaru. Existuje více metod virtualizace: paravirtualizace, emulace, aplikační virtualizace a plná virtualizace. Bližší popis k těmto úrovním virtualizací je k dispozici v bakalářské práci studenta aplikované informatiky Přírodovědecké fakulty Jihočeské univerzity pana Luboše Plcha. V této práci se nadále budeme zabývat pouze plnou virtualizací (částečně paravirtualizací) z důvodu využití superpočítače CMU, na kterém je plná virtualizace implementována. Plná virtualizace, nebo také někdy nativní virtualizace, se skládá z hypervizoru, který běží mezi hardwarem a virtualizovanými systémy. Slouží ke komunikaci virtualizovaného operačního systému s hardwarem. Pomocí tohoto hypervizoru jsme schopni přiřadit virtuálnímu počítači určité systémové prostředky a virtuální počítač k těmto prostředkům přistupuje, jako by byly jeho fyzické. Toto platí pouze pro hypervizor prvního typu. Hypervizor druhého typu k hardwaru přistupuje za pomoci hostitelského operačního systému, na kterém emuluje hardwarové prostředky. Z důvodu velkého overhadu jsme tuto technologii zavrhli. Posledním druhem zde uváděné virtualizace je paravirtualizace pracující na principu hypervizoru, na kterém je spuštěn operační systém sloužící ke správě a spuštění dalších operačních systémů. Tyto virtualizované systémy pracují v podstatě jako thready hostitelského operačního systému.



Plná virtualizace



Plná virtualizace prvního typu

Benchmark

Definice a kritéria

Pokud si rozdělíme pojem „benchmark“ na pojmy „bench“ a „mark“, tak lze tyto pojmy přeložit jako „soudci značek“. Již z tohoto překladu lze tedy vyvodit, že se jedná a jakousi metodu posuzování značek (veličin). Přesněji řečeno se jedná o obecné označení programu, který je schopen určitým algoritmem zjistit výkonnost či schopnost daného hardwaru. Benchmark lze použít i pro software. Je možné testovat databáze či překladače. Porovnáním dvou a více testů jsme poté schopni vyvodit, který hardware (software) je lepší, samozřejmě pro konkrétní účel (podle testu). V následné části stručně popíšeme části benchmarku:

1. Co vlastně chceme testovat?
    - Bude se jednat o zpracování textu nebo náročnější procesy jako úprava videa nebo jiné uživatelské aplikace? Položit si „jednoduchou“ otázku. Co chceme testovat? (Jak uvidíme poději, tato otázka nemusí být úplně jednoduchá)

2. Dílčí benchmark
    - Jedná se většinou o pouhý soupis hardwarových prostředků počítače (velikost vyrovnávací paměti, velikost cache, CPU, RAM, HDD, …)
3. Benchmark
    - Zde už program obsahuje testy jako takové
4. Výpis
    - Vypíše hodnoty, které měřením zjistil

Následující bod není pro každý benchmark povinný a většina starších benchmarků (bez přístupu na web) tento krok neobsahují a je třeba ho provézt manuálně.

5. Porovnání testů
    - V tomto posledním kroku benchmarku můžeme zjistit, jak na tom doopravdy náš hardware či software je v porovnání s ostatními výsledky. Většina „nových“ grafických benchmarků tento krok obsahuje.

Dělení benchmarků

Podle operačního systému

- Linux, Windows, Android, …

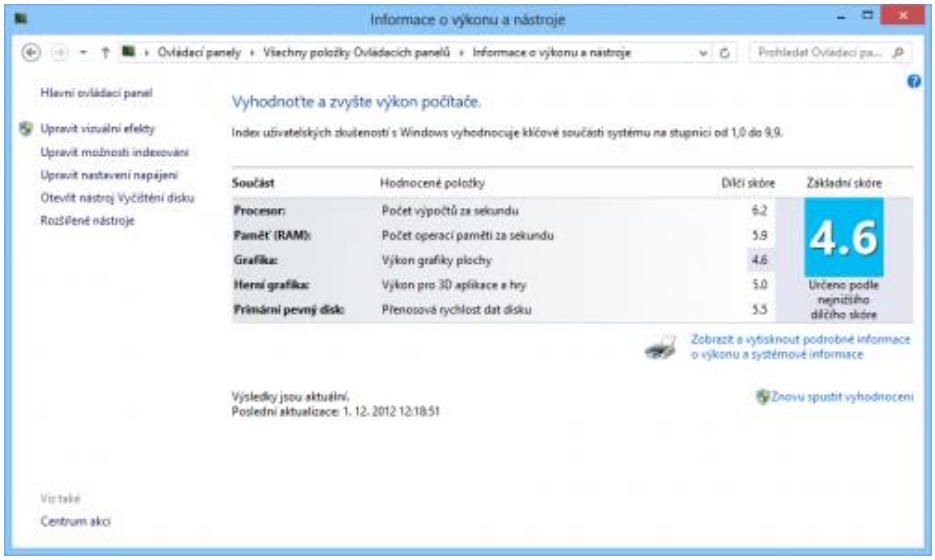
Podle druhů testu

- CPU
- RAM
- I/O operace
- Display
- NETWORK (Komunikace po síti)
- GPU
- …

Podle určení testovacího objektu

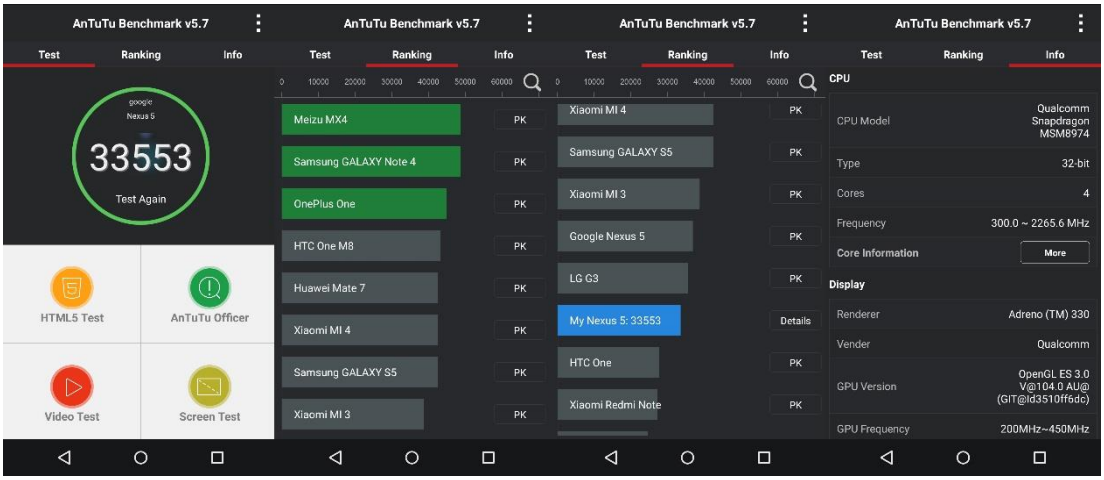
- Hardware
- Software

Rychlý přehled benchmarků

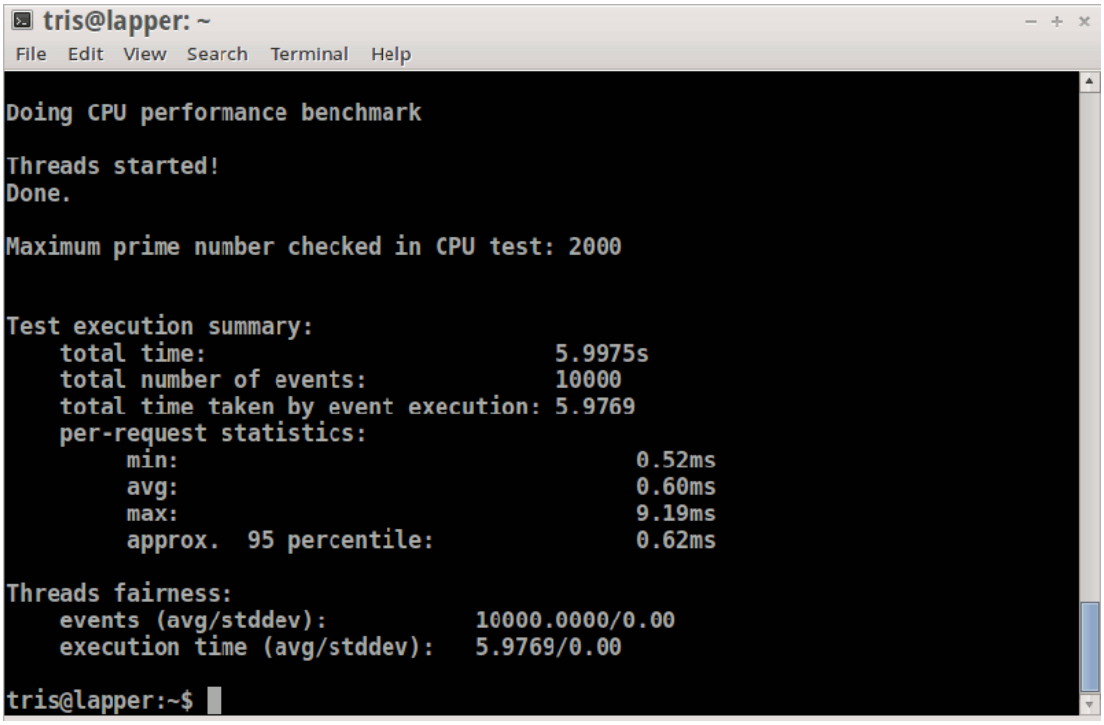


Na prvním místě bychom zde uvedli benchmark, který většina zná (a má ho dokonce naistalováný na osobním počítači), ale neví, že se ve skutečnosti jedná o benchmark: **Windows System Assessment Tool** (Index uživatelských zkušeností s Windows

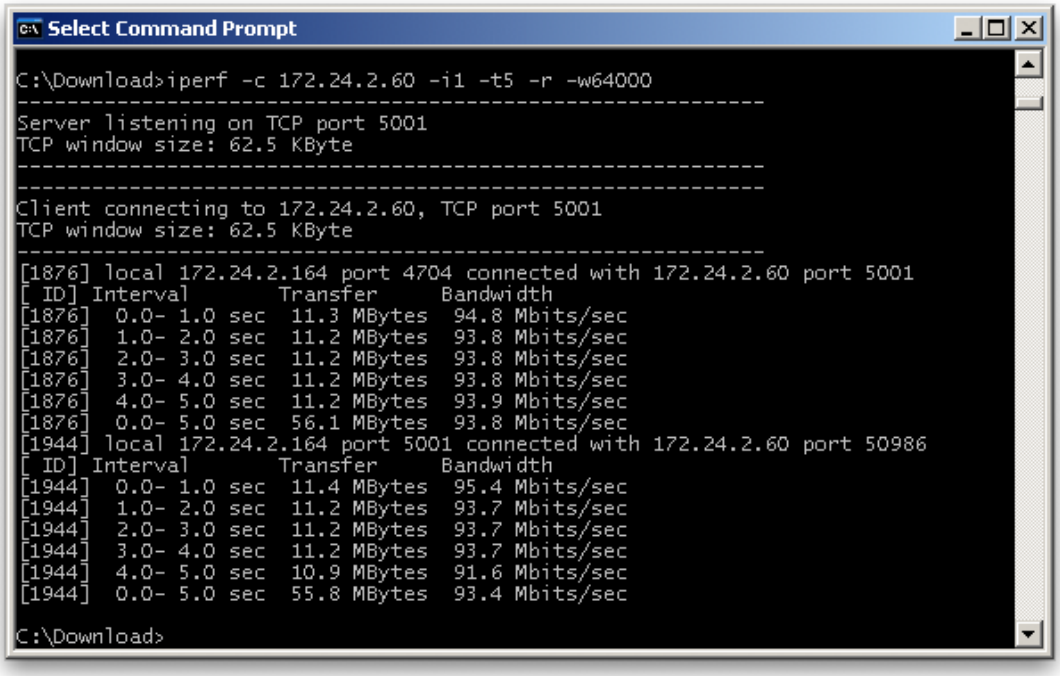
Dalším bechmarkem je známý **3DMark**. Jedná se o benchmark zaměřený na měření herního výkonu.



S vývojem chytrých telefonů a tabletů přišly benchmarky i na tato zařízení. Jeden z nejpoužívanějších benchmarků pro platformu Android je **AnTuTu** benchmark. Ovšem tato platforma je schopna testovat pouze procesory s architekturou ARM.



Sysbench - multiplatformní benchmark ovládaný pomocí příkazové řádky. Jedná se o benchmark, který je schopen testovat v podstatě veškerý HW kromě GPU a sítě. Později se tomuto programu budeme věnovat podrobněji, protože tento program jsme využili také pro testování CMU.



Poslední zmíněný benchmark je iperf. Tento benchmark testuje síť mezi dvěma uzly s tím, že jeden uzel vždy musí být server a druhý klient. Jako v předchozím případě se jedná o benchmark s možností multiplatformního využití.

Sysbench[4]

Jedná se o modulární, přenositelný a multithreadový testovací nástroj pro vyhodnocení výkonu hardwaru z hlediska intenzivního zatížení. Myšlenkou tohoto benchmarku je rychlé získání představy o výkonu systému.

Vlastnosti (Testy)

Výkon operací I/O

Výkon scheduleru (plánovače)

Rychlost alokace a přemístění paměti (RAM)

Implementace POSIX (Portable Operating System Interface) treads

Výkonost databázového serveru

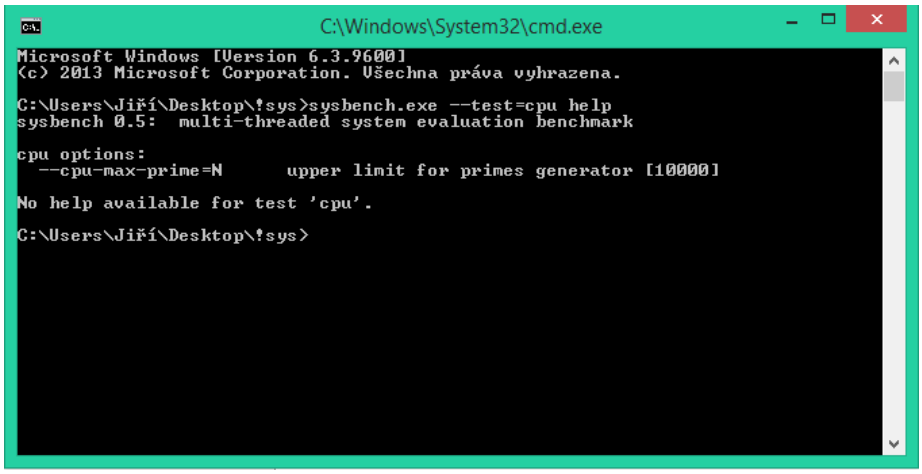
Základní syntaxe

Základní syntaxi bychom si ukázali na příkladu, jak by měl vypadat příkaz.

```
sysbench [common-options] --test=name [test-options] command
```

Na tomto příkazu můžeme vidět postup při psaní příkazů. V první části příkazu voláme testovací program, následuje typ testu. Jedná se o testy vyjmenované výše. Podrobnější popis bude pouze u testů, které jsme použili. Na konci příkazu musíme vždy napsat, co má daný příkaz provézt:

- Prepare
    - Tento druh příkazu použijeme pouze pro nastavení souborů pro testování I/O a databázových prostředků
- Run
    - Příkaz pro běh testu jako takového. Jakýkoliv test musí být zakončen tímto příkazem
- Cleanup
    - Příkaz pro odstranění dočasných souborů vytvořených příkazem *prepare*
- Help
    - Tento příkaz je možný zadat takřka kdykoliv. Pokud ho zadáme bez zadání testu, tak vypíše obecné možnosti nastavení a pokud už máme vypsaný druh testu, vypíše možnosti zadaného testu, viz obr.:



Popis základních příkazů

Příkaz	Popis	Základní hodnota
--num-threads	Nastavuje na kolika threadech by měl test běžet	1
--max-time	Jedná se o limit, po jaký čas má test běžet ve vteřinách (0 = neomezeně)	0
--debug	Vypíše další info. Kupříkladu vypíše info po threadech	off
--thread-stack-size	Velikost každého threadu	32K
--max-time	Časový limit provedení testu (0 = neomezený)	0

TEST CPU

Jedná se o jednoduchý test výpočtem prvočísel do určité hodnoty. Pro tento výpočet se používá 64-bit integers. Hodnotu prvočísla je možné definovat a tím ztížit (prodloužit) výpočet. Samozřejmě je možné použít výpočet ve více threadech pomocí pomocného příkazu. Celý příkaz na testování CPU může vypadat například takto:

```
sysbench --test=cpu --cpu-max-prime=20000 run
```

TEST THREADŮ

Tento test je psaný pro testování plánovače úloh, takzvaného scheduleru.

Sysbench vytvoří určitý počet vláken o určitém počtu mutexů. Všechna vlákna se potom spustí a je otázkou scheduleru, jak se s tím vypořádá.

```
sysbench --num-threads=64 --test=threads --thread-yields=100
--thread-locks=2 run
```

TEST MUTEX

Provedení tohoto testu je na principu napodobení situace, kdy všechny thready běží současně většinu času a získají zámek mutexu jen na krátký okamžik.

TEST RAM

Zde benchmark používá sekvenční zápis a čtení z operační paměti. Pro nastavení můžeme použít rozšiřující příkazy pro velikost bloku a celkovou velikost paměti, kterou využijeme k testu a nechybí ani možnost volby operace. Samozřejmostí je také možnost spuštění testu ve více threadech.

Příkaz	Popis	Základní hodnota
--memory-block-size	Velikost použitého bloku paměti pro test	1K
--memory-scope	Dvě možnosti: global, local. Specifikuje, jestli každý thread použije globálně alokovanou paměť anebo si ji alokuje lokálně	global
--memory-total-size	Celková přenesená data	100G
--memory-oper	Jaká se má provést operace. Zde jsou dvě možnosti: read, write	100G

TEST FILEIO

Tento test je velice jednoduchý. Pracuje na principu zápisu a čtení z disku. Toto čtení či zápis lze dále rozdělit a testovací program Sysbench to umožňuje. Proto lze tyto operace rozdělit na operace sekvenční a na operace náhodné. Podrobnější přehled operací ukáže následující tabulka.

seqwr	Sekvenční zápis

seqrewr	Sekvenční přepisování
seqrd	Sekvenční čtení
rndrd	Náhodné čtení
rndwr	Náhodný zápis
rndrw	Kombinace čtení a zápisu

Pro testování je důležité nejdříve si připravit testovací soubory. K této přípravě slouží příkaz *prepare*, který se zapíše na konec příkazu (namísto příkazu *run*).

Následuje příkaz *run* pro proběhnutí testů. Poté, co máme výsledky, je třeba testovací soubory (pokud je již nebudeme potřebovat), smazat a pro tento případ zde máme příkaz *cleanup*.

Při vytváření souborů si například můžeme zvolit, jak velké mají tyto soubory být a určit jejich počet. Tuto možnost nastavujeme pomocí přepínačů. Jedním z nich je celková velikost všech souborů, dalšími jsou počet souborů, velikost bloku… viz tabulka.

Příkaz	Popis	Základní hodnota
--file-num	Počet vytvořených souborů	128
--file-total-size	Velikost všech souborů (kolik nám to zabere na disku)	2G
--file-bock-size	Velikost bloku pro operace I/O	16K

Příklad testu:

```
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw prepare
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw run
$ sysbench --num-threads=16 --test=fileio --file-total-size=3G
--file-test-mode=rndrw cleanup
```

TEST OLTP

Tento test je zaměřený na testování databáze. Pro naše účely je v tuto chvíli nepodstatný, proto se tímto testem nebudeme zabývat.

Iperf[5]

Jedná se o velice jednoduchý testovací nástroj zaměřený na testování komunikace po síti.

Pro měření je zapotřebí minimálně dvou počítačů. Je jedno, jestli budou fyzické anebo virtuální. Jediné, co je třeba zajistit, aby byly ve stejné síti. Na jednom z počítačů zapneme testovací program iperf s přepínačem *–s*, a tím spustíme server, na který se budou dotazovat klienti. Klienty spustíme úplně stejně, nýbrž použijeme přepínač *–c* a za tento přepínač ještě doplníme IP adresu serveru (adresa počítače, na kterém běží test s přepínačem *-s*). Pro výpis nápovědy stačí napsat přepínač *–h*.

Ve chvíli, kdy spustíme server, tak začne naslouchat na všech dostupných IP adresách a na všech dostupných portech. Po spuštění klienta je komunikace prováděna pomocí protokolu TCP na portu 5001. Tento port i protokol lze nastavením přepínačů změnit.

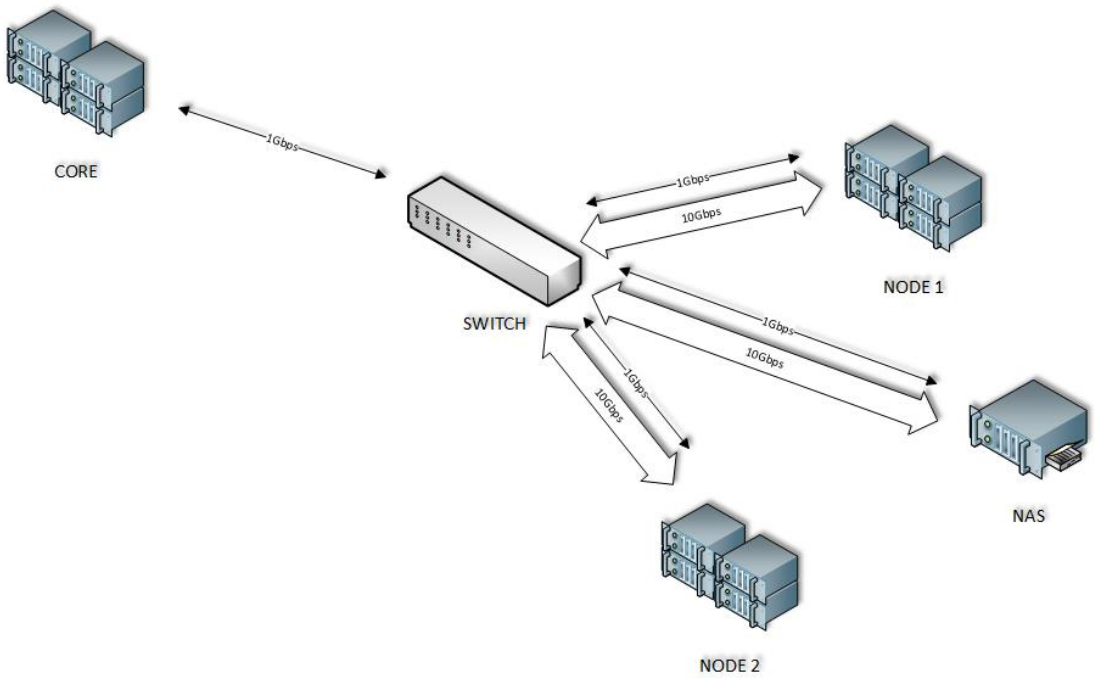
Přepínač	Popis
-u	UDP mód
-p (číslo portu)	Změna čísla portu

Iperf nabízí větší množství přepínačů pro zpřesnění testu.

Přepínač	Popis
-b (cílová rychlost)	Nastavení maximální rychlosti
-t (doba testu)	Jak dlouho má test běžet
-F (soubor)	Načíst data pro odeslání ze souboru
-P (počet klientů)	Simulace více klientů
-M (velikost)	Velikost TCP segmentu

Testování superpočítače CMU

Schéma zapojení



Zde můžeme vidět názorné schéma zapojení superpočítače CMU. Uprostřed schéma zapojení vidíme centrální switch, který zprostředkovává komunikaci mezi uzly. Konkrétní výpočetní uzly NODE a NAS jsou připojeny pomocí 10Gbps. Celý systém je ještě propojen pomocí 1Gbps sítě (CORE pouze 1Gbps). Propojení s CORE slouží pro řízení a správu, nicméně uzly jsou nastavené tak, že pokud se 10Gbps připojení nasytí, tak se může využít i linka pro správu. Za pomoci domény lze jakýkoliv server takřka neomezeně škálovat.

Testování

Na základě poznatků z teoretické části práce jsme si testování rozdělili na testy CPU, RAM, I/O (HDD) a NET.

Důležité pro testování virtuálních počítačů je celkový počet virtuálních počítačů, který poběží na daném hardwaru. Pro tento test jsme si vytvořili vždy určitý počet virtuálních počítačů definovaných nastavením procesorových jader. Podle nastavení

počtu procesorových jader na virtuálním počítači musíme nastavit daný test. Tento test lze spustit na více threadech (podle počtu procesorových jader daného počítače). Pro rozdělení virtuálních počítačů podle nastavení hardwaru (procesorová jádra) budeme vycházet z rovnice, kde máme tři proměnné. První proměnná se bude měnit se změnou testovacího uzlu (NODE), druhá se bude měnit podle nastavení počtu jader procesoru virtuálního počítače a třetí nám bude uvádět výsledek tohoto výrazu. Nastavení virtuálních počítačů bude vždy homogenní. To znamená, že veškeré nastavení virtuálních počítačů (případně i fyzických) bude pokud možno co nejshodnější. Na toto nastavení není třeba každý nainstalovaný virtuální stroj přenastavovat, nýbrž se vytvoří takzvaný vzorový virtuální počítač a ten se bude následně klonovat. Obraz těchto systémů je reálně uložený na síťovém úložišti NAS. Pro obrazy jsme použili operační systémy Windows 8.1 x64, Windows 8.1 x86, Debian 7 x64, Debian 7 x86 s tím, že všechny systémy jsou plně aktualizované. Rovnice pro rozdělení (nastavení) počítačů podle výkonu:

$$\frac{0{,}8 * CPU_{NODE}}{x} = V \qquad x \in <1; (0{,}8 * CPU_{NODE})>$$

V – možný počet spuštěných virtuálních počítačů (je možné, že nevyjde celé číslo, pak tuto desetinou část jednoduše ořízneme)

0,8*CPU_{NODE} – Jedná se zde o 80% z výkonu určitého NODE. 80% bereme v potaz z důvodu rezervy výkonu pro běh serveru. U serveru CMU, ať je to jakýkoliv node, to znamená 32 procesorových jader

x – jak vidíme v podmínce vedle rovnice, tak x může nabývat hodnot od 1 do „ořízlého“ počtu procesorových jader fyzického počítače. Tyto hodnoty budeme postupně dosazovat

Jednoduše řečeno, levá strana rovnice nám říká, jaké nastavení je na virtuálním počítači a pravá strana rovnice (výsledek) nám říká, na kolika virtuálních počítačích lze tento test spustit. Testovat lze i hraniční hodnoty a případně i přetížení tohoto systému jednoduchou úpravou rovnice. Tato úprava spočívá ve změně násobku v čitateli, kde násobíme 0,8 krát (tedy 80 % celkového výkonu).

Testování CPU probíhá na určitém počtu virtuálních počítačů *V* najednou. Pro toto jednotné spuštění slouží program, který je pro tento účel naprogramován. Program

pracuje ve dvou verzích. Klient, který se spouští okamžitě po spuštění a přihlášení virtuálního počítače. Další částí tohoto programu je server. Tato serverová část programu běží na serveru CORE a je schopná z tohoto serveru řídit veškeré testy. Program je schopen spuštění téměř jakéhokoliv příkazu na všech virtuálních počítačích takřka zároveň. Pro tuto komunikaci používá síťového rozhraní.

Test CPU

Z obecného nastavení testování vyplývá, že budeme postupovat v testech od jednoduchých jedno-jádrových virtuálních počítačů, až po jeden plně vytížený více-jádrový. Budeme se snažit testovat vždy maximální vytížení daného hardwaru. Pro tento test jsme si zvolili testovací program Sysbench 0.5, který je v krátkosti popsán výše. Zde je pouze uveden a vysvětlen příkaz, který jsme pro toto testování použili:

$$sysbench.exe \;\; --test = cpu \;\; --num - threads = x \;\; --cpu - max - prime = p \;\; run$$

V příkazu vidíme proměnné x a p. Proměnnou x (definovanou výše pro počet threadů při výpočtu počtu virtuálních počítačů) pouze dosadíme do zápisu testu. Proměnnou p nastavujeme maximální prvočíslo, do kterého se počítá test. Proměnná p se skládá z více částí. Statické části a části dynamické.

$$p = c + (d * x)$$

c – statická část proměnné p, která nabývá hodnoty 50 000

d*x – dynamická část proměnné p, kde dynamičnost je zaručena proměnou x (počet threadů = počet procesorů virtuálního počítače) násobenou proměnnou d, kde proměnná d zaručuje určitý násobek obtížnosti testu a nabývá hodnoty 30 000.

Celková rovnice potom zaručuje zvýšení obtížnosti testu se zvýšením počtu threadů (jader virtuálního počítače), aby byl test objektivní. Pokud bychom toto přenastavování testů ignorovali například při testu, kde by bylo prvočíslo moc veliké a test by probíhal na jednom threadu, tak by mohlo být čekání na výsledek testu velice zdlouhavé. Naopak, pokud by nastavené prvočíslo bylo malé (nastavené na test pro 1 thread), tak při výpočtu na více threadech by výsledek byl za příliš krátký časový interval a my bychom nemuseli zaznamenat rozdíl mezi testovanými systémy.

Sysbench vypisuje výsledky do jednoduché tabulky, která se strojově zhodnotí (pomocí programu pro řízení testů, který běží na serveru CORE). Pro představu je zde ukázka výpisu testu CPU:

```
sysbench 0.5:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 2
Random number generator seed is 0 and will be ignored


Primer numbers limit: 110000

Threads started!


General statistics:
    total time:                          253.5789s
    total number of events:              10000
    total time taken by event execution: 507.1233s
    response time:
         min:                                  36.80ms
         avg:                                  50.71ms
         max:                                  93.06ms
         approx.  95 percentile:               53.67ms

Threads fairness:
    events (avg/stddev):           5000.0000/31.00
    execution time (avg/stddev):   253.5616/0.01
```

Z výsledků testu můžeme vidět, na jakém počtu threadů byl test puštěn a jaký byl maximální limit pro hodnotu prvočísla. Za prvočíslem následuje tabulka s dílčími výsledky. Z těchto výsledků nás při testu CPU bude nejvíce zajímat hodnota, za jaký časový okamžik byl příkaz proveden. Tato hodnota se nazývá total time a v našem případě má hodnotu 253,5789s. Tuto hodnotu budeme porovnávat mezi virtuálními a fyzickými počítači.

Pro testování samotné jsme si naprogramovali jednoduchý skript, který nám postupně prochází veškeré možnosti nastavení testů pro CPU.

- Windows

```
for($a=1;$a-64 ;$a=$a*2)
{
$max=($a * 30000)+50000
echo $a $max>>pokusCpu.txt
echo "max prime $max thread $a is runing"
echo "sysbench.exe --test=cpu --num-threads=$a --cpu-
```

```
max-prime=$max run" >>pokusCpu.txt
C:\Users\win\Documents\sysbench\sysbench.exe --
test=cpu --num-threads=$a --cpu-max-prime=$max run
>>pokusCpu.txt
echo "max prime $max thread $a is done"
}
```

- Linux

```
#!/bin/bash


echo "x64">>pokusCpu.txt
for((a=1 ; $a-64 ;a=$a*2))
do
const=30000
j=50000
maxa=$(($a * $const))
max=$(($maxa + $j))
echo $a $max>>pokusCpu.txt
#K="K"
#by="$b$K"
echo "max prime "$max " thread " $a " is runing"
#echo "./sysbench --test=memory --num-threads=$a --
memory-block-size=$by --memory-total-size=1000G run"
>>pokus.txt
echo "./sysbench --test=cpu --num-threads=$a --cpu-
max-prime=$max run" >>pokusCpu.txt
./sysbench --test=cpu --num-threads=$a --cpu-max-
prime=$max run >>pokusCpu.txt
echo "max prime "$max " thread " $a " is done"
done

#./sysbench --test=cpu run >> ram.txt
```

Ve skriptu je vidět cyklus postupně procházející veškeré nastavení, které jsme si definovali výše. To znamená, že projde veškeré nastavení threadů a k nim přiřadí danou velikost prvočísla, do kterého se mají hodnoty počítat.

Následuje výpis hodnot testování fyzických a virtuálních počítačů. Podrobný výpis je obsahem přiloženého CD.

- **Fyzický počítač**

1 Thread	362,0651
2 Thready	253,5789
4 Thready	189,5382
8 Threadů	203,0986
16 Threadů	241,9213
32 Threadů	409,8460

- **Virtuální počítač Windows 8.1 x64**

1 Thread	256,5584
2 Thready	255,2437
4 Thready	228,5391
8 Threadů	224,4081
16 Threadů	332,0797
32 Threadů	413,4953

- **Virtuální počítač Windows 8.1 x86**

1 Thread	332,1014
2 Thready	254,9815
4 Thready	186,4905
8 Threadů	202,1175
16 Threadů	331,6312
32 Threadů	413,6656

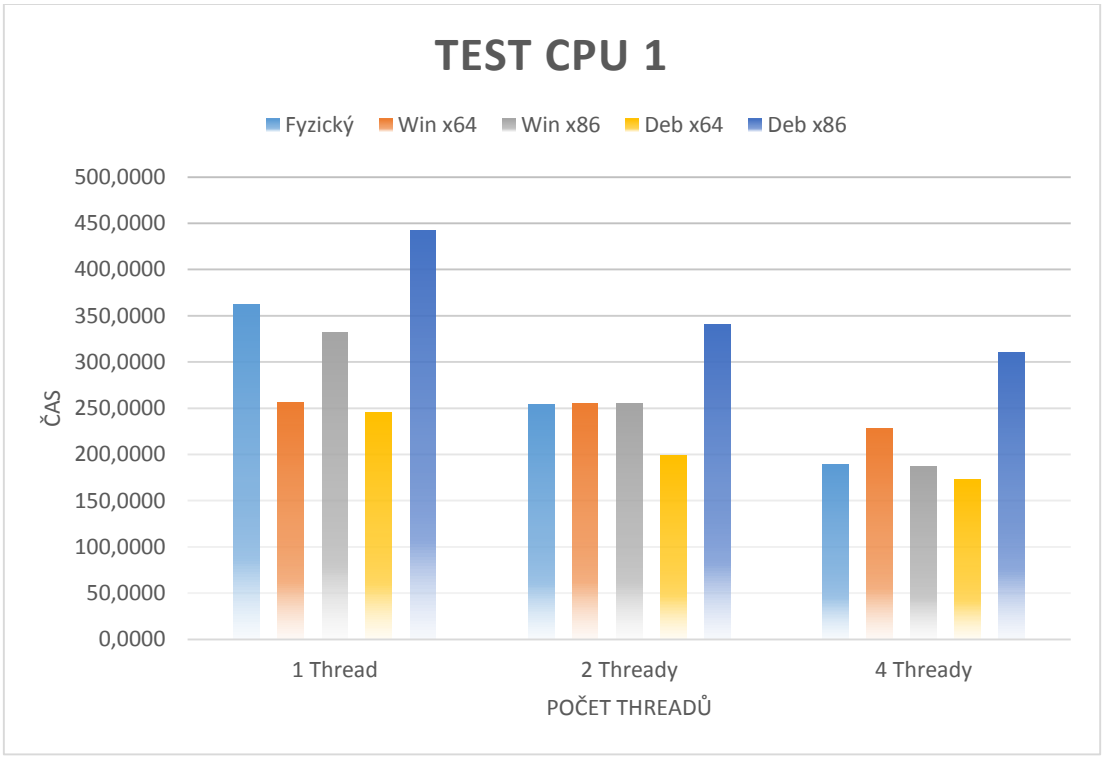
- **Virtuální počítač Debian 7 x64**

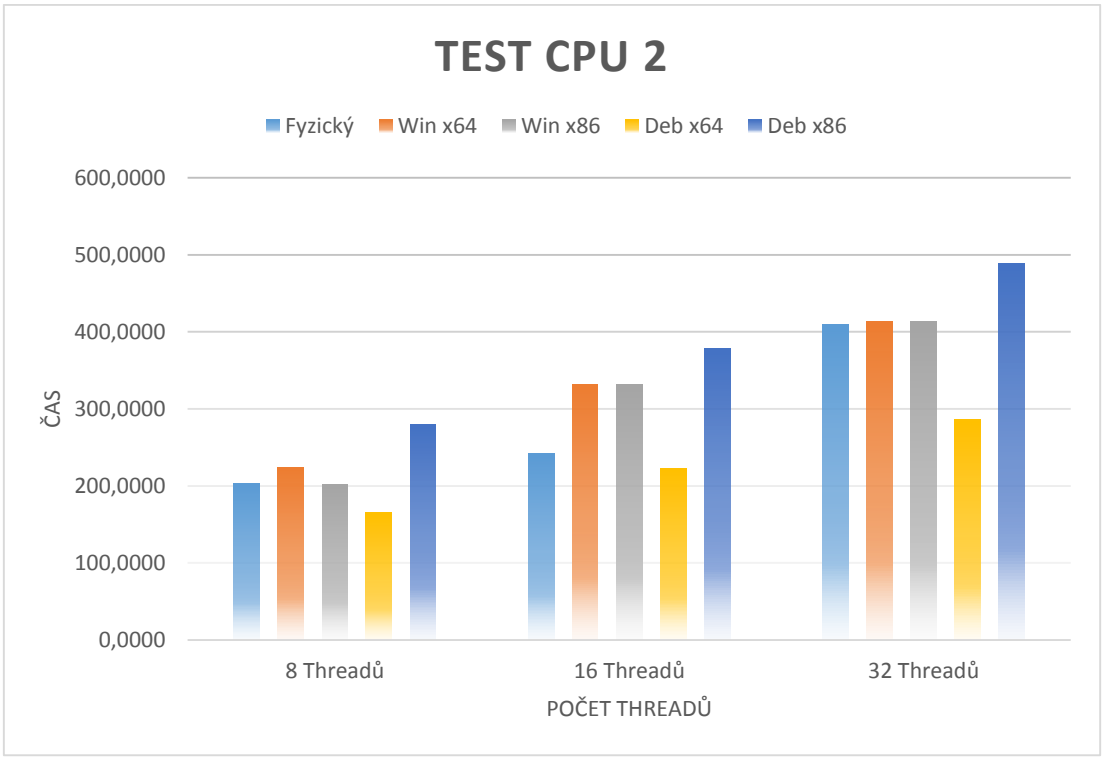
1 Thread	245,6426
2 Thready	198,8289
4 Thready	173,0316
8 Threadů	165,5946
16 Threadů	223,1496
32 Threadů	285,9788

- **Virtuální počítač Debian 7 x86**

1 Thread	442,2984
2 Thready	341,0260
4 Thready	310,5209
8 Threadů	280,2945
16 Threadů	378,0922
32 Threadů	488,7553

Grafy výsledků:





Test RAM

Pro tento test jsme opětovně využili programu Sysbench ve stejné verzi.

V testu operační paměti jsme použili, oproti testu procesoru, maximálně „pouze“ 4 thready. Toto nastavení je z důvodu fyzického zapojení pamětí, kde maximální počet kanálů paměti na uzel je 4. Z tohoto důvodu nemělo význam testovat více threadů, ledaže bychom chtěli vědět, jak se systém zachová při přetížení. Tento test pracuje na principu měření rychlosti přístupu do paměti. Pro tento účel si vytvoří soubory s velikostí 2^z, kde z je číslo od 4 do 10. Následně jsou soubory zpracovány RAM pamětí až do celkové velikosti v našem případě 1000GB.

$$sysbench.exe \;\; --test = memory \;\; --num - threads = x \;\; --memory - block - size = b \;\; --memory - total - size = t \;\; run$$

Výpis tesu je dosti podobný jako u testu CPU, nicméně nás nebude v tomto případě zajímat celkový čas provedení operace, ale rychlost provedení této operace. Pro představu je zde zase uveden výpis testu. Jedná se o test, kde jsme nastavili počet threadů pouze na jeden, velikosti testovacích souborů byly 32KB a celková velikost 1000GB.

```
sysbench 0.5:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1
Random number generator seed is 0 and will be ignored


Threads started!

Operations performed: 32768000 (158062.61 ops/sec)

1024000.00 MB transferred (4939.46 MB/sec)


General statistics:
    total time:                          207.3103s
    total number of events:              32768000
    total time taken by event execution: 165.9285s
    response time:
         min:                                  0.00ms
         avg:                                  0.01ms
         max:                                  7.52ms
         approx.  95 percentile:               0.01ms

Threads fairness:
    events (avg/stddev):           32768000.0000/0.00
    execution time (avg/stddev):   165.9285/0.00
```

Jak bylo uvedeno výše, nejdůležitější poznatek z tohoto testu je rychlost. Položka je uvedena zhruba uprostřed testu (v tomto případě 4939,46MB/sec).

Testování proběhlo zase za pomoci skriptů jak pro Windows, tak pro Linux:

- Windows

```
cd C:\Users\win\Documents\sysbench
for($i=1;$i -le 4;$i++)
{
for($j=16;$j-le 1024;$j=$j*2)
{
$k="K"
$ij="$j"+$k
echo $ij
echo "sysbench.exe --test=memory --num-threads=$i --
memory-block-size=$ij  --memory-total-size=1000G  run
">>test.txt
C:\Users\win\Documents\sysbench\sysbench.exe       --
```

```
test=memory --num-threads=$i --memory-block-size=$ij -
-memory-total-size=1000G run >>test.txt
}
}
```

- Linux

```
#!/bin/bash

for((a=1 ; $a-5 ;a=$a+1))
do for((b=16 ; $b-2048 ; b=$b*2))
do echo $a $b>>pokus.txt
K="K"
by="$b$K"
echo "block "$by " thread " $a " is runing"
echo "./sysbench --test=memory --num-threads=$a --
memory-block-size=$by --memory-total-size=1000G run"
>>pokus.txt
./sysbench --test=memory --num-threads=$a --memory-
block-size=$by --memory-total-size=1000G run
>>pokus.txt
echo "block " $by " thread " $a " is done "
done
done
#./sysbench --test=cpu run >> ram.txt
```

Ze zápisu skriptů vyplývá, že test se skládá ze dvou vzájemně vnořených cyklů, kde první prochází počtem threadů a druhý (vnořený) mění velikosti testovaných souborů.

Jako v předchozích případech následuje výpis hodnot testování fyzických a virtuálních počítačů. Podrobný výpis je obsahem přiloženého CD.

- **1 Thread**

1 Thread	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	985,78 MB/sec	589,14 MB/sec	552,20 MB/sec	4 245,22 MB/sec	3 772,36 MB/sec
32K	982,77 MB/sec	676,88 MB/sec	637,08 MB/sec	5 300,22 MB/sec	4 939,46 MB/sec
64K	1 007,88 MB/sec	624,84 MB/sec	575,01 MB/sec	6 238,11 MB/sec	6 018,56 MB/sec
128K	987,07 MB/sec	628,04 MB/sec	605,24 MB/sec	6 918,15 MB/sec	6 783,48 MB/sec
256K	1 012,98 MB/sec	635,09 MB/sec	641,95 MB/sec	6 903,01 MB/sec	7 028,68 MB/sec
512K	1 023,16 MB/sec	646,29 MB/sec	591,93 MB/sec	6 957,36 MB/sec	6 978,48 MB/sec
1024K	1 030,48 MB/sec	842,78 MB/sec	582,62 MB/sec	6 985,99 MB/sec	7 125,07 MB/sec

- **2 Thready**

2 Thready	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	1 893,59 MB/sec	1 730,70 MB/sec	1 229,91 MB/sec	3 793,10 MB/sec	1 916,98 MB/sec
32K	1 996,85 MB/sec	1 862,51 MB/sec	1 203,56 MB/sec	7 863,30 MB/sec	5 388,25 MB/sec
64K	2 029,35 MB/sec	1 934,37 MB/sec	1 203,90 MB/sec	10 841,37 MB/sec	10 458,80 MB/sec
128K	2 007,95 MB/sec	1 913,39 MB/sec	1 203,78 MB/sec	12 464,40 MB/sec	12 310,47 MB/sec
256K	2 045,74 MB/sec	1 977,15 MB/sec	1 210,87 MB/sec	13 118,78 MB/sec	13 151,95 MB/sec
512K	2 016,60 MB/sec	2 009,73 MB/sec	1 223,46 MB/sec	13 627,21 MB/sec	13 730,16 MB/sec
1024K	2 021,73 MB/sec	1 972,28 MB/sec	1 288,50 MB/sec	14 145,64 MB/sec	14 281,08 MB/sec

- **3 Thready**

3 Thready	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	2 750,75 MB/sec	2 653,03 MB/sec	2 465,05 MB/sec	2 657,28 MB/sec	3 186,27 MB/sec
32K	2 898,00 MB/sec	2 855,26 MB/sec	2 652,15 MB/sec	1 944,77 MB/sec	1 482,30 MB/sec
64K	2 843,53 MB/sec	2 973,28 MB/sec	2 820,86 MB/sec	14 865,00 MB/sec	18 045,42 MB/sec
128K	2 970,31 MB/sec	3 015,09 MB/sec	2 928,02 MB/sec	18 264,24 MB/sec	22 470,47 MB/sec
256K	3 076,26 MB/sec	3 029,90 MB/sec	2 867,09 MB/sec	19 647,50 MB/sec	24 438,76 MB/sec
512K	3 053,22 MB/sec	2 987,95 MB/sec	2 903,92 MB/sec	20 428,80 MB/sec	25 556,20 MB/sec
1024K	3 085,60 MB/sec	3 060,16 MB/sec	2 810,48 MB/sec	21 072,18 MB/sec	26 650,86 MB/sec

- **4 Thready**

4 Thready	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	3 593,15 MB/sec	3 531,90 MB/sec	4 355,08 MB/sec	2 469,32 MB/sec	1 555,82 MB/sec
32K	3 861,48 MB/sec	3 733,69 MB/sec	4 605,91 MB/sec	2 953,99 MB/sec	2 194,90 MB/sec
64K	3 978,14 MB/sec	3 921,63 MB/sec	4 859,61 MB/sec	12 935,82 MB/sec	8 537,10 MB/sec
128K	4 066,02 MB/sec	4 005,66 MB/sec	4 959,52 MB/sec	22 737,63 MB/sec	27 625,97 MB/sec
256K	4 099,81 MB/sec	4 037,09 MB/sec	4 992,74 MB/sec	25 764,43 MB/sec	31 831,80 MB/sec
512K	4 093,83 MB/sec	4 047,92 MB/sec	5 005,36 MB/sec	26 920,82 MB/sec	33 229,32 MB/sec
1024K	4 096,04 MB/sec	4 070,51 MB/sec	5 023,05 MB/sec	27 512,48 MB/sec	34 274,28 MB/sec

Test I/O (HDD)

Test harddisku je v tomto systému velmi netypický z důvodu umístění datového úložiště na jiném serveru. Reálné datové úložiště je ve skutečnosti připojené pomocí ethernetu 10Gbps na server datového úložiště (NAS). To znamená, že ve skutečnosti nejsme z virtuálního počítače schopni otestovat rychlost zápisu a čtení z lokálního disku, protože virtuální počítač žádný lokální disk nemá a veškerý diskový prostor je uložen na vzdáleném serveru, tudíž veškeré testy půjdou na tento server. Dalším podstatným problémem pro testování disku je toto datové úložiště samotné. NAS je nakonfigurován na softwarový RAID-5. To znamená, že výsledky testů záleží i na výkonu procesoru. Pro tyto případy jsou zde uvedeny dva testy fyzického stroje. Jeden s označením local a druhý SMB. Local testuje reálnou rychlost disků na uzlu NAS a test s označením SMB testuje celý systém úložiště (to znamená RAID pole).

Výpis programu Sysbench:

```
Running the test with following options:
Number of threads: 1
Random number generator seed is 0 and will be ignored


Extra file open flags: 0
128 files, 800Mb each
100Gb total file size
Block size 400Mb
Number of IO requests: 100
Read/Write ratio for combined random IO test: 1.50
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Threads started!

Operations performed:  60 reads, 40 writes, 128 Other = 228 Total
Read 23.438Gb  Written 15.625Gb  Total transferred 39.063Gb  (398.8Mb/sec)
    1.00 Requests/sec executed

General statistics:
    total time:                          100.3018s
    total number of events:              100
    total time taken by event execution: 95.7898s
    response time:
         min:                                  532.19ms
         avg:                                  957.90ms
         max:                                 8589.68ms
         approx.  95 percentile:              1310.33ms

Threads fairness:
    events (avg/stddev):           100.0000/0.00
    execution time (avg/stddev):   95.7898/0.00
```

Z výpise je zřejmé, že nás v tomto případě bude zajímat rychlost náhodného čtení a zápisu, což je hodnota opět cca uprostřed tabulky. V tomto konkrétním případě se jedná o hodnotu 398,8Mb/sec.

Pro tyto testy jsme si vytvořili pár jednoduchých testovacích skriptů:

- Windows

```
cd C:\Users\win\Documents\sysbench

C:\Users\win\Documents\sysbench\sysbench.exe --
test=fileio --num-threads=$a --file-total-size=100G -
-file-test-mode=rndrw prepare >>pokusio.txt
```

```
for($a=1;$a-64 ;$a=$a*2)

{

echo $a >>pokusio.txt

echo " thread $a is preparing"

echo "C:\Users\win\Documents\sysbench\sysbench.exe --
test=fileio --max-requests=100 --num-threads=$a --
file-total-size=100G --file-block-size=400M --file-
test-mode=rndrw run " >>pokusio.txt

echo "redy and go"

C:\Users\win\Documents\sysbench\sysbench.exe --
test=fileio --max-requests=100 --num-threads=$a --
file-total-size=100G --file-block-size=400M --file-
test-mode=rndrw run >>pokusio.txt

#echo "done and clean"

#C:\Users\win\Documents\sysbench\sysbench.exe --
test=fileio --num-threads=$a --file-total-size=100G -
-file-test-mode=rndrw cleanup >>pokusio.txt

echo "thread $a is done"

   }
```

- Linux

```
#!/bin/bash

./sysbench --test=fileio --num-threads=4 --file-total-
size=100G --file-test-mode=rndrw prepare >>pokusiodeb.txt
for((b=1 ; $b-64 ; b=$b*2))
do echo  $b>>pokusiodeb.txt
echo "thread "$b  " is runing"
echo "./sysbench --test=fileio --num-threads=$by --max-
requests=100 --file-block-size=400M --file-total-size=100G
--file-test-mode=rndrw run ">>pokusiodeb.txt
./sysbench --test=fileio --num-threads=$b --max-
requests=100 --file-block-size=400M --file-total-size=100G
--file-test-mode=rndrw run >>pokusiodeb.txt

echo "thread " $b " is done "
done
#./sysbench --test=cpu run >> ram.txt
```

Tyto skripty provádí test jednoduchým stylem. Nejprve si připraví soubory na testování a následně tyto soubory testuje. Pro testování jsme definovali velikost bloku na 400MB a celková velikost souborů je 100GB. Testy s tímto nastavením provádíme postupně nad všemi thready. To znamená, že používáme dělení threadů stejné, jako při

testování CPU.

	Fyzický Local	Fyzický SMB	Win x64	Win x86	Deb x64	Deb x86
1 Thread	39,95 Mb/sec	119,74 Mb/sec	518,13 Mb/sec	398,80 Mb/sec	44,04 Mb/sec	36,28 Mb/sec
2 Thready	40,14 Mb/sec	199,62 Mb/sec	762,21 Mb/sec	452,21 Mb/sec	50,06 Mb/sec	153,24 Mb/sec
4 Thready	48,92 Mb/sec	243,87 Mb/sec	495,30 Mb/sec	1 070,90 Mb/sec	65,28 Mb/sec	129,62 Mb/sec
8 Threadů	72,51 Mb/sec	246,36 Mb/sec	1 560,78 Mb/sec	1 226,85 Mb/sec	62,98 Mb/sec	128,71 Mb/sec
16 Threadů	67,60 Mb/sec	253,21 Mb/sec	907,90 Mb/sec	1 317,89 Mb/sec	72,43 Mb/sec	133,16 Mb/sec
32 Threadů	62,00 Mb/sec	231,20 Mb/sec	343,15 Mb/sec	701,25 Mb/sec	65,75 Mb/sec	99,24 Mb/sec

Test NET

Pro testování komunikace virtuálních počítačů po síti jsme použili testovací program iperf. Iperf je jednoduchý nástroj, který je pro naše účely naprosto ideální. Testování musí probíhat jak mezi fyzickým a virtuálním počítačem, tak mezi virtuálním a virtuálním počítačem. Tento test dále rozšiřujeme na test počítačů v rámci jednoho výpočetního uzlu, kde máme virtuální síťové karty, které by měly pracovat na principu zápisu a čtení z operační paměti a na test mezi výpočetními uzly. Pro lepší představu poslouží tabulky.

local	Debx86	Debx64	Win86	Winx64
Debx86	x			
Debx64		x		
Win86			x	
Winx64				x

remote	Debx86	Debx64	Win86	Winx64
Debx86	x			
Debx64		x		
Win86			x	
Winx64				x

Nastavení testů je naprosto jednoduché. Na jednom počítači spustíme iperf s přepínačem –s, to znamená, že se spustí jako server a bude naslouchat na všech portech. Následně se spustí klient, který se pouští přepínačem –c, a za tento přepínač se připíše IP adresa serveru, na kterém běží iperf v režimu server. Pro vytížení komunikace jsme použili navázání spojení na více threadech. Konkrétně jsme použili pro testování 10 threadů. Nastavení počítačů bylo shodné, nicméně v tomto testu není pro nás až tolik důležité.

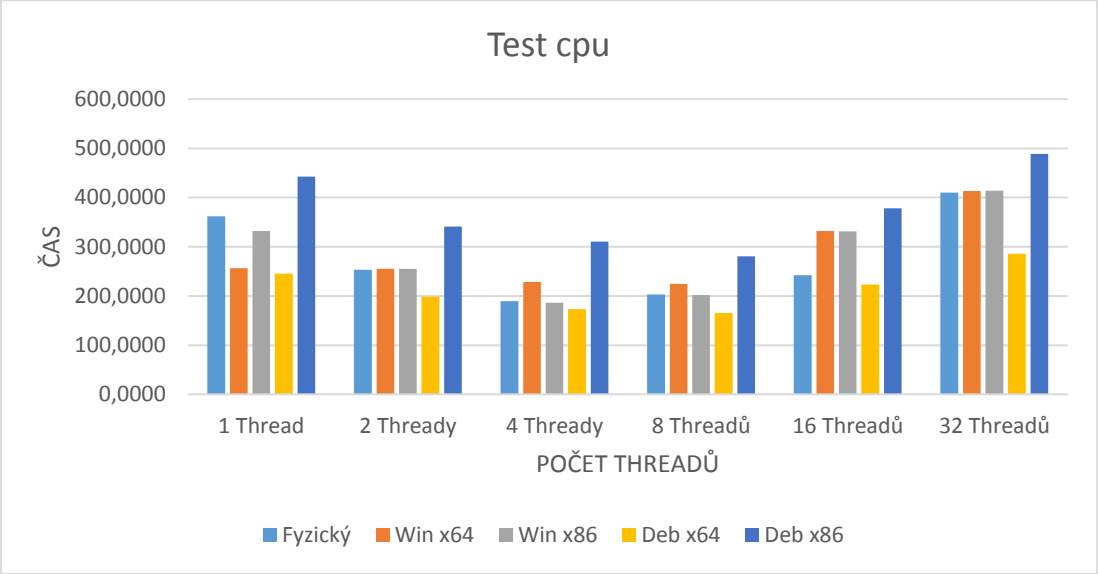
	propustnost sítě	Debx86	Debx64	Winx86	Winx64
local	10000,00 MBps	17920,00 MBps	23859,20 MBps	11366,40 MBps	10649,60 MBps
remote	10000,00 MBps	9260,00 MBps	9320,00 MBps	9130,00 MBps	8990,00 MBps

Výsledky testů vytížení síťové komunikace ukazují velikou převahu v rychlosti lokální komunikace mezi operačním systémem Linux oproti Windows na jednom uzlu, jak je vidět z tabulky. Obecně oba operační systémy jsou schopny komunikovat (mezi uzly) takřka na limitu propustnosti sítě.

Zhodnocení

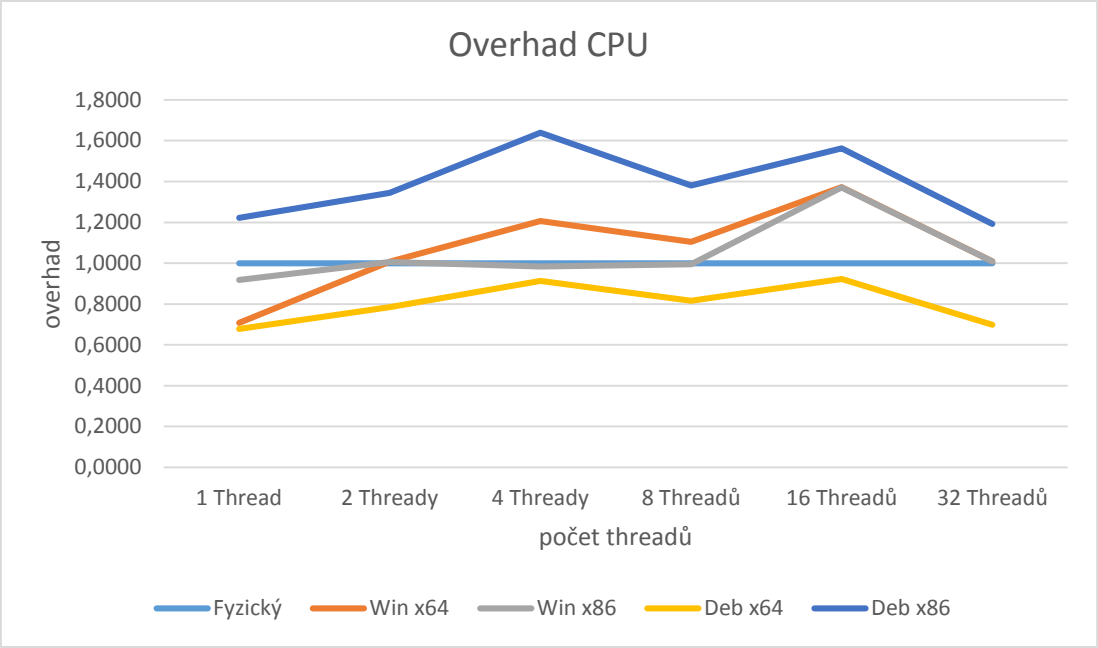
Hodnocení dílčích testů bude probíhat nejdříve pomocí grafu naměřených výsledků a následovat bude výpočet overhadu s grafem.

Zhodnocení CPU



V grafu nás nejvíce zajímá světle modrý sloupec (fyzický počítač), ke kterému budeme veškeré virtuální počítače porovnávat. Z prvního pohledu je tedy jasné, že ne vždy bude v tomto případě overhad větší než jedna. Toto zjištění je velmi zajímavé z hlediska virtualizace, protože můžeme říci, že co se týče procesoru, tak opravdu záleží na kvalitě implementace jádra operačního systému, který zprostředkovává komunikaci mezi OS a HW. Zde vidíme jasný příklad Debx64 vs. Debx86. Overhady těchto OS vychází z tabulky a jsou celkem jasné, nicméně je zajímavé zjištění, že pokud běží virtualizovaný Debian x64 v podstatě pod Windows, tak má lepší hodnoty, než samotný fyzický systém Windows. Viz tabulka a graf:

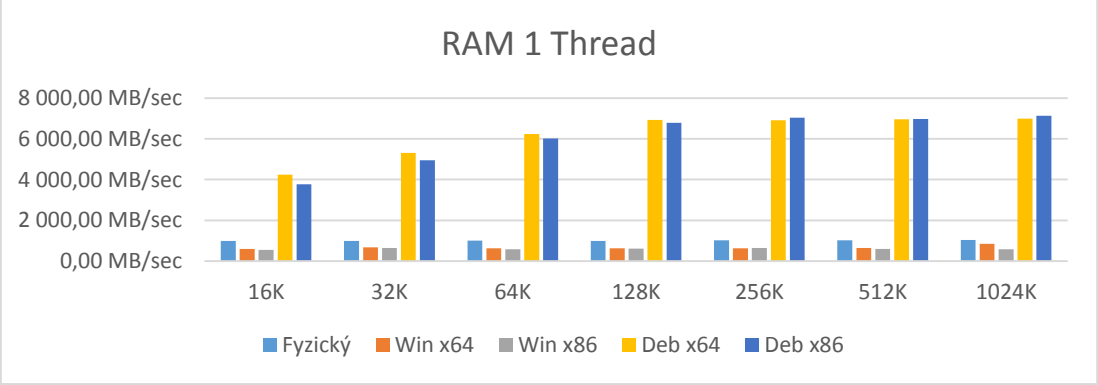
OVERHAD	Fyzický	Win x64	Win x86	Deb x64	Deb x86
1 Thread	1,0000	0,7086	0,9172	0,6784	1,2216
2 Thready	1,0000	1,0066	1,0055	0,7841	1,3449
4 Thready	1,0000	1,2058	0,9839	0,9129	1,6383
8 Threadů	1,0000	1,1049	0,9952	0,8153	1,3801
16 Threadů	1,0000	1,3727	1,3708	0,9224	1,5629
32 Threadů	1,0000	1,0089	1,0093	0,6978	1,1925



Zhodnocení RAM

Vyhodnocení operační paměti bude probíhat stejně, jako je proveden výpis testů. Tudíž budeme postupovat postupně podle počtu threadů.

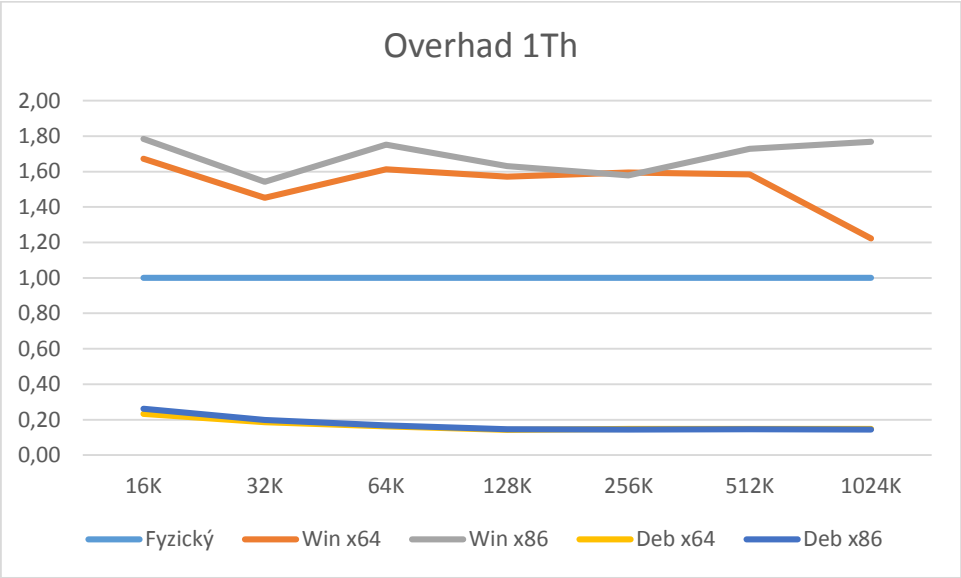
1 Thread



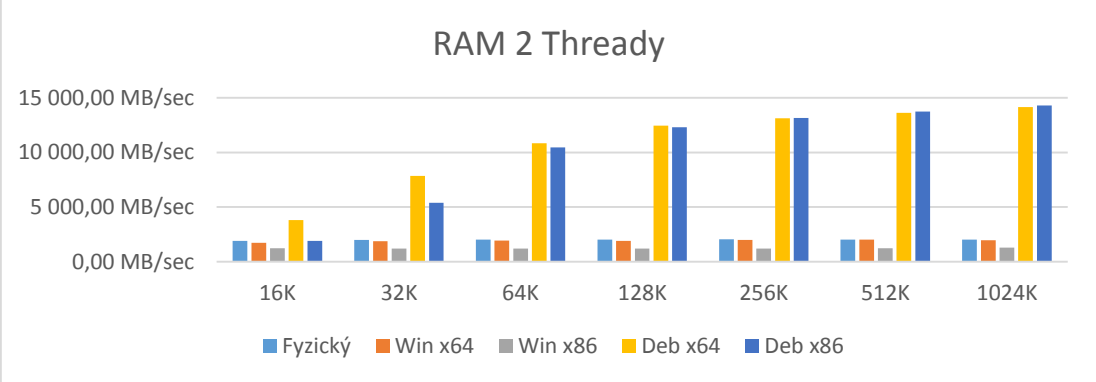
Z grafu můžeme vidět, že co se týče přístupu do paměti pro jeden thread, jednoznačně vyhrává Debian pro obě verze. Největší propad má virtualizovaný Windows, který je ve všech případech pomalejší (ne o moc, ale pomalejší), než fyzický operační systém. Z grafu je zřejmé, že stejně jako při testování CPU i zde záleží na jádru operačního systému, jak je schopno pracovat s HW, přičemž je obecně známé, že Linux je v tomto směru podstatně vyspělejší. Následuje tabulka overhadů a

porovnání těchto overhadů v grafech.

1 Thread	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	1,00	1,67	1,79	0,23	0,26
32K	1,00	1,45	1,54	0,19	0,20
64K	1,00	1,61	1,75	0,16	0,17
128K	1,00	1,57	1,63	0,14	0,15
256K	1,00	1,60	1,58	0,15	0,14
512K	1,00	1,58	1,73	0,15	0,15
1024K	1,00	1,22	1,77	0,15	0,14

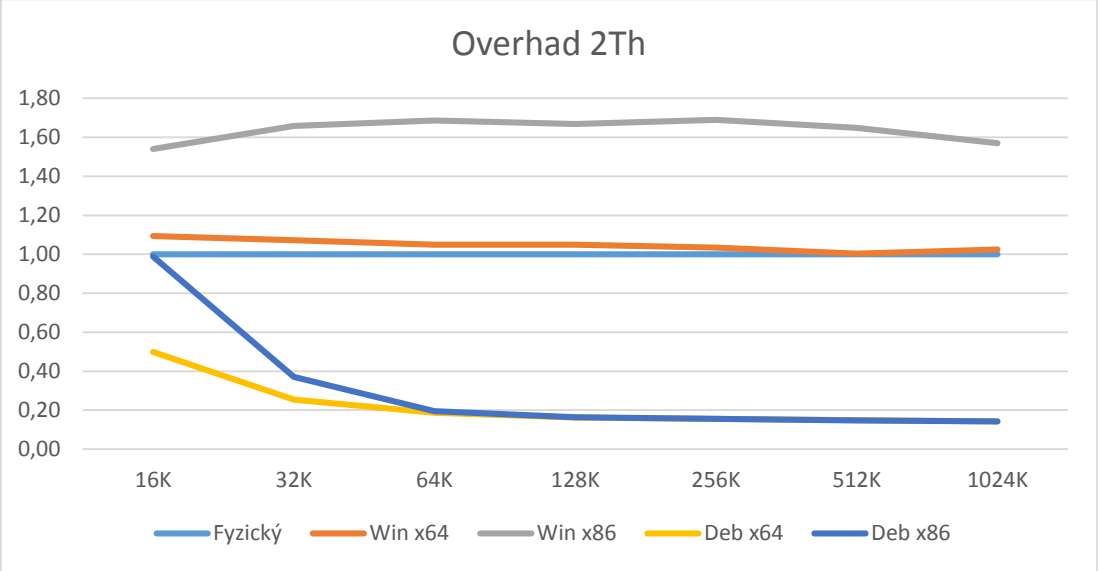


2 Thready

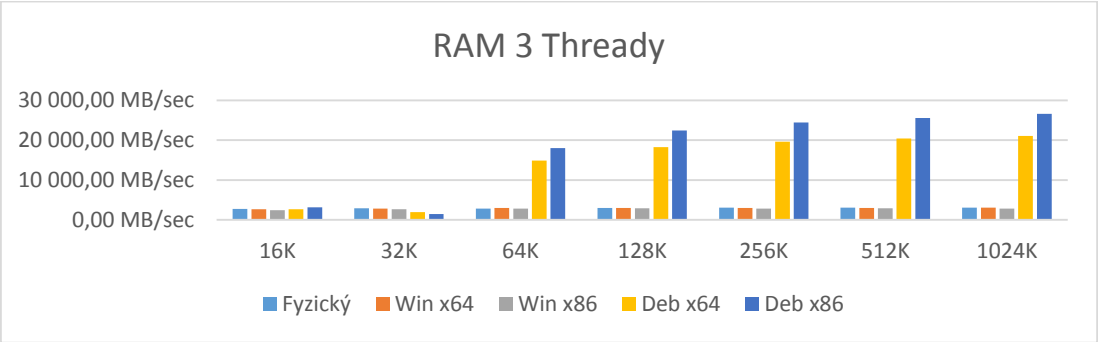


Zde se opakuje scénář, jako u testu na jeden thread, nicméně je zde pozvolnější start u obou verzí Linuxu. Windows x86 i fyzický zůstávají téměř na stejné úrovni, jako v přechozím testu. Zajímavý skok výkonnosti provedly Windows x64, které se přiblížily k fyzickému počítači. Tabulka a graf overhadů nám poví více:

2 Thread	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	1,00	1,09	1,54	0,50	0,99
32K	1,00	1,07	1,66	0,25	0,37
64K	1,00	1,05	1,69	0,19	0,19
128K	1,00	1,05	1,67	0,16	0,16
256K	1,00	1,03	1,69	0,16	0,16
512K	1,00	1,00	1,65	0,15	0,15
1024K	1,00	1,03	1,57	0,14	0,14

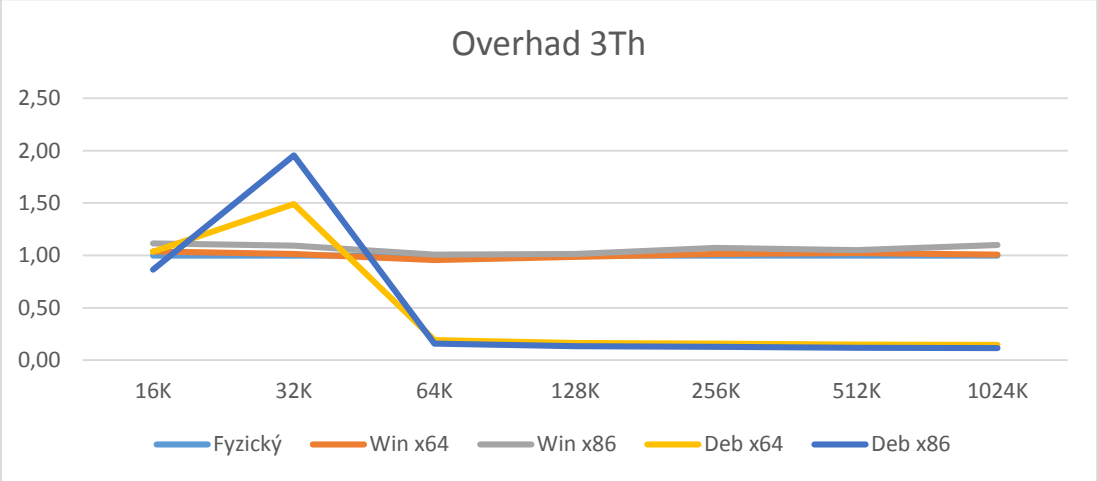


3 Thready

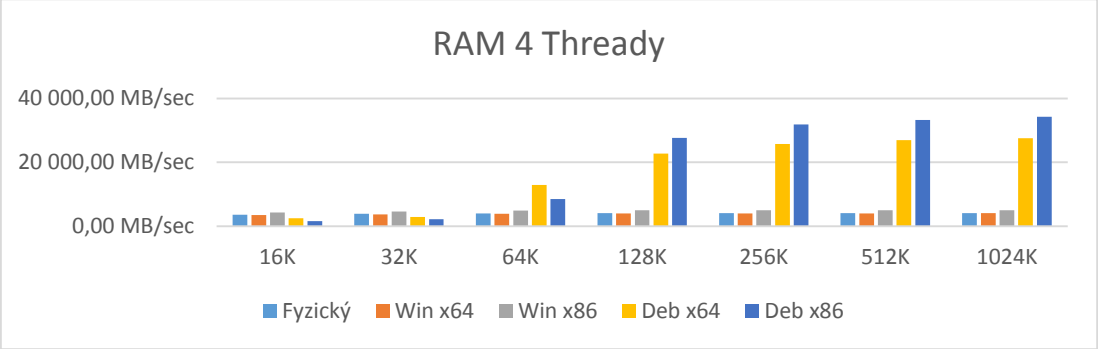


Oproti vyhodnocení testu RAM (2 threadů) je začáteční zpoždění u operačního systému Debian ještě markantnější. Pokud se zaměříme na oba operační systémy x86, tak si můžeme všimnout určitého zlepšení ke konci testu. Viz graf a tabulka overhadů.

3 Thready	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	1,00	1,04	1,12	1,04	0,86
32K	1,00	1,01	1,09	1,49	1,96
64K	1,00	0,96	1,01	0,19	0,16
128K	1,00	0,99	1,01	0,16	0,13
256K	1,00	1,02	1,07	0,16	0,13
512K	1,00	1,02	1,05	0,15	0,12
1024K	1,00	1,01	1,10	0,15	0,12

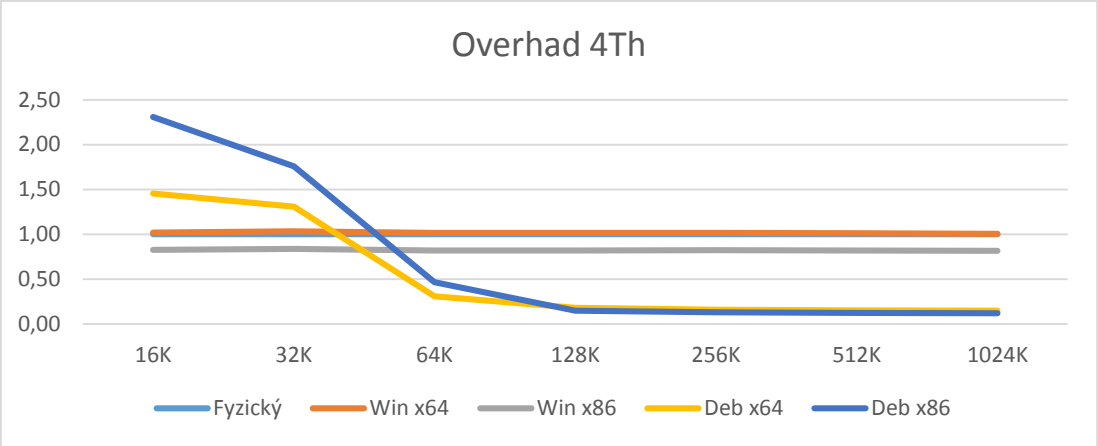


4Thready

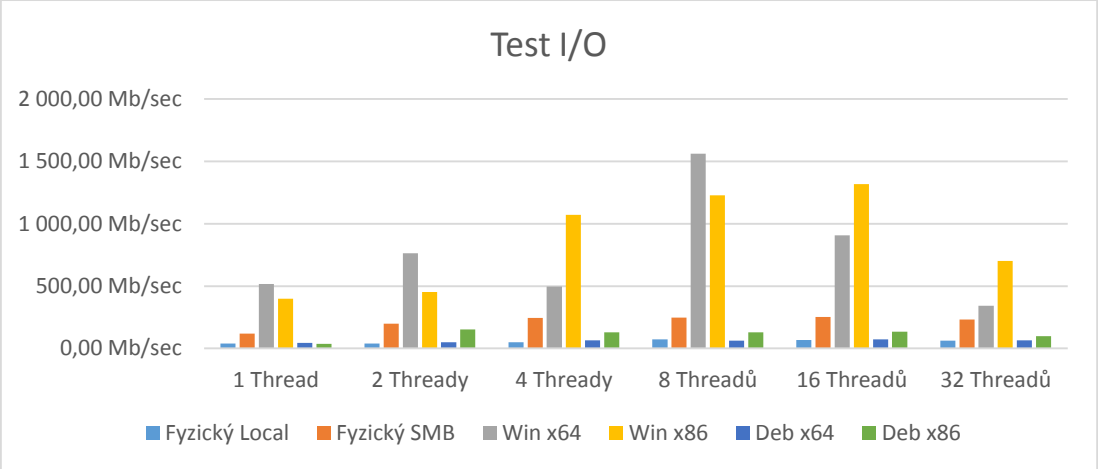


Tento test dopadl dosti podobně, jako test pro 3 thready, nicméně je zde vidět ještě vyšší nárůst rychlostí, co se týče zpracování větších částí bloku paměti operačními systémy x86.

4 Thready	Fyzický	Win x64	Win x86	Deb x64	Deb x86
16K	1,00	1,02	0,83	1,46	2,31
32K	1,00	1,03	0,84	1,31	1,76
64K	1,00	1,01	0,82	0,31	0,47
128K	1,00	1,02	0,82	0,18	0,15
256K	1,00	1,02	0,82	0,16	0,13
512K	1,00	1,01	0,82	0,15	0,12
1024K	1,00	1,01	0,82	0,15	0,12

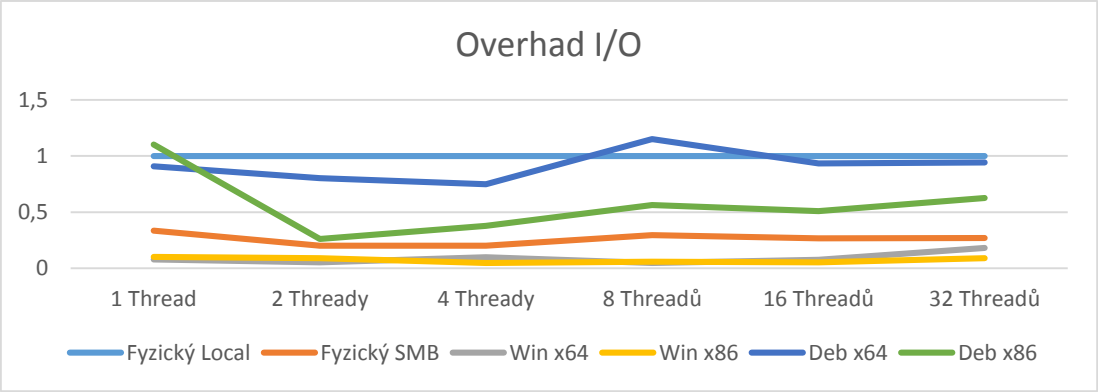


Zhodnocení I/O (HDD)

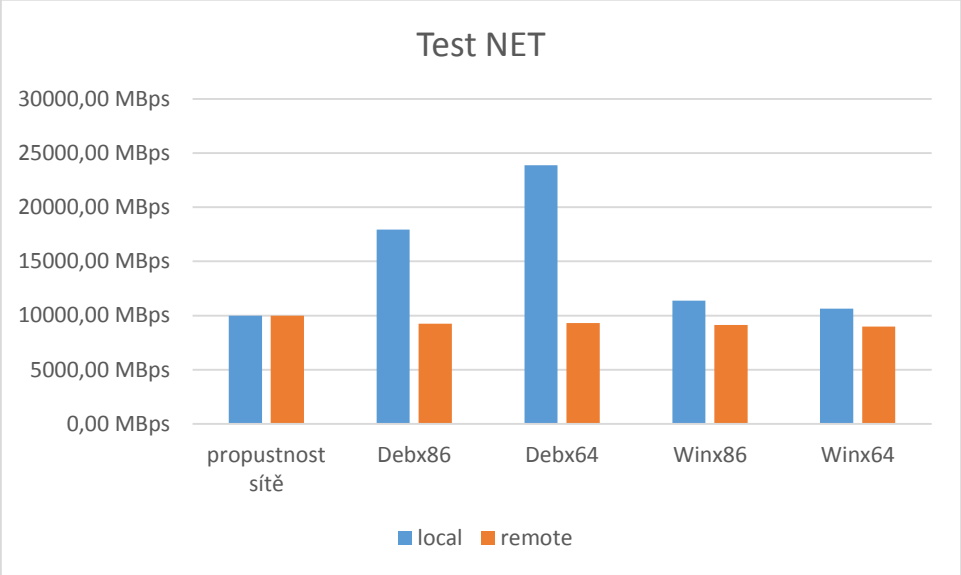


Oproti testům na výkonnost systému, kde vedl Debian, tak zde jednoznačně vedou obě verze Windows.

OVERHAD	Fyzický Local	Fyzický SMB	Win x64	Win x86	Deb x64	Deb x86
1 Thread	1	0,333664607	0,077109992	0,100183049	0,907198002	1,10121
2 Thready	1	0,201057008	0,052656092	0,088753013	0,801818	0,261909423
4 Thready	1	0,200582277	0,098760347	0,045677502	0,749314502	0,377380034
8 Threadů	1	0,294337555	0,046459439	0,059104813	1,151347232	0,563382799
16 Threadů	1	0,266968129	0,074456438	0,051293433	0,93332689	0,507652448
32 Threadů	1	0,268144464	0,180664432	0,088406417	0,942889734	0,624710292

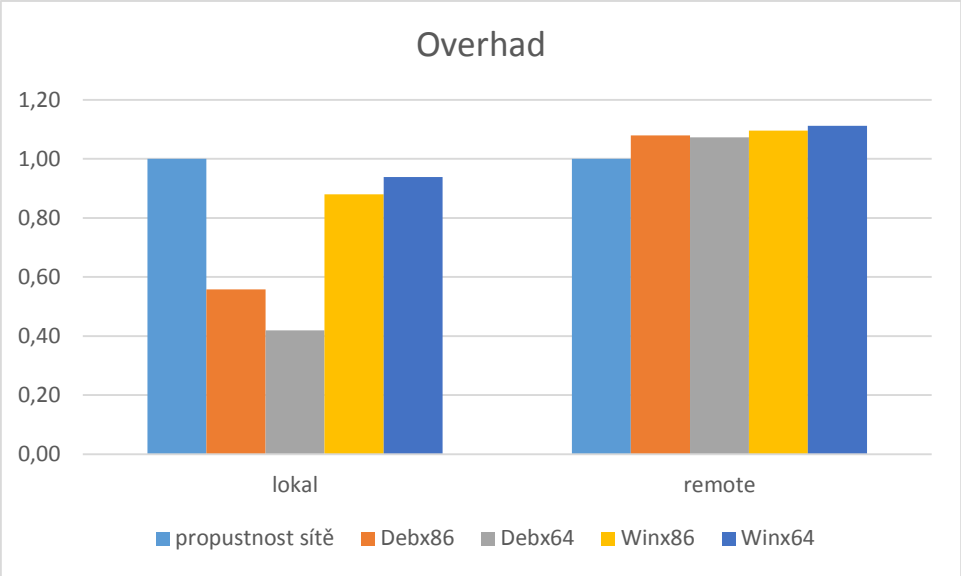


Zhodnocení NET



Z grafu můžeme jednoznačně vyvodit, že komunikace v rámci jednoho uzlu je závislá na jádru operačního systému. Protože reálná komunikace virtuální počítač vs. virtuální počítač je komunikace pouze za pomoci zápisu a čtení z RAM pamětí jednoho uzlu, což nám dokazují hodnoty u OS Debian x64.

OVERHAD	propustnost sítě	Debx86	Debx64	Winx86	Winx64
lokal	1,00	0,56	0,42	0,88	0,94
remote	1,00	1,08	1,07	1,10	1,11



Závěr

Testování superpočítače přineslo velice překvapivé výsledky. Ukázalo slabá místa využití této technologie a návrhu systému a naopak ukázalo, jak je hypervizor od Microsoftu opravdu velice dobře proveden. Zároveň jsme schopni říci, že výkon virtuálního systému extrémně závisí na provedení jádra operačního systému. Dokonce jsme schopni říci, že virtuální Linux, který běží pod Windows, nemá žádný overhad, ba dokonce má overhad menší než 1. To je dáno lepším zpracováním jádra operačního systému a precizním provedením hypervizoru ze strany Microsoftu. Z výsledků testů můžeme tvrdit, že i systém tvářící se na první pohled jako fyzický, je ve skutečnosti virtualizován a nemá žádnou podstatnou výhodu jeho použití oproti systému virtualizovanému.

Použitá literatura

[1]Xianghua Xu, Feng Zhou, Jian Wan, and Yucheng Jiang. "Quantifying Performance Properties of Virtual Machine," Fei Yu. *International Symposium on Information Science and Engineering: ISISE 2008 : [proceedings], 20-22 December 2008, Shanghai, China*, vol. 2008, no. 08, pp. 24-28, c2008.

[2]M R Anala, and G Shobha. "Comparative study of application performance on virtual machine and physical machine," *2012 IEEE International Conference on Computational Intelligence and Computing Research*, vol. 2012, 18-20, pp. 1-6, 2013.

[3](2015). Přehled technologie Hyper-V. Microsoft. *TN Přehled technologie Hyper-V* [online]. Available from: https://technet.microsoft.com/library/hh831531.aspx

[4]Vlastimil Marek. (2009). Něco v síti: fejetony, které vycházely od roku 1997 na internetu na adrese http://svet.namodro.cz. *Http://imysql.com* [online]. Available from: http://imysql.com/wp-content/uploads/2014/10/sysbench-manual.pdf

[5Vlastimil Marek. (2003). Něco v síti: fejetony, které vycházely od roku 1997 na internetu na adrese http://svet.namodro.cz. *Http://www.netcheif.com* [online]. Available from: http://www.netcheif.com/downloads/iperf.pdf]