České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Tomáš Jiřík**

Studijní program: Otevřená informatika (magisterský)
Obor: Počítačové inženýrství

Název tématu: **Optimalizace spotřeby elektrické energie datového centra**

Pokyny pro vypracování:

1. Seznamte se s problematikou implementace datových center.
2. Vyberte vhodné servery a proveďte jejich konfiguraci.
3. Navrhněte optimalizační algoritmus pro minimalizaci spotřeby energie datového centra.
4. Otestujte funkcionalitu optimalizačního algoritmu.

Seznam odborné literatury:

[1] BARNHART, Cynthia, et al. Branch-and-price: Column generation for solving huge integer programs. Operations research, 1998, 46.3: 316-329.
[2] SCHRIJVER, Alexander. Theory of linear and integer programming. Wiley. com, 1998.
[3] SAVELSBERGH, Martin. A branch-and-price algorithm for the generalized assignment problem. Operations Research, 1997, 45.6: 831-841.

Vedoucí: Ing. Daniel Novák, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

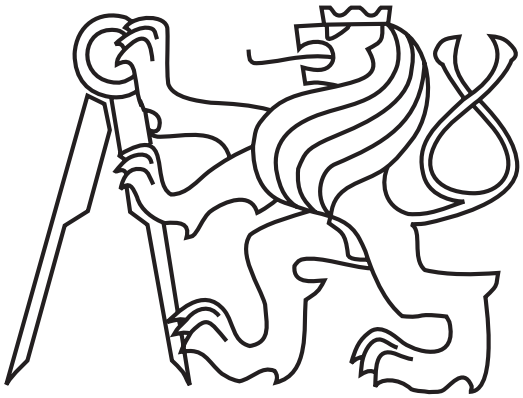
prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2014

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ
KATEDRA ŘÍDÍCÍ TECHNIKY



DIPLOMOVÁ PRÁCE

Optimalizace spotřeby elektrické energie datového centra

Tomáš Jiřík

Vedoucí práce: Ing. Daniel Novák PhD.

Studijní obor: Otevřená informatika, Magisterský

Obor: Počítačové inženýrství

12. května 2014

Poděkování

Děkuji všem, kteří mi pomohli při tvorbě této diplomové práce a rodině za jejich plnou podporu a pochopení. Chtěl bych také poděkovat za příležitost podílet se na projektu datového centra, který realizují ve firmě Haug-land, spol. s.r.o. V neposlední řadě děkuji Ing. Danielu Novákovi za vedení mé diplomové práce.

Prohlášení

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

V Praze, dne 12. května 2014 Podpis .....  .....

Abstrakt

Předmětem této diplomové práce je implementace programu pro optimalizaci spotřeby elektrické energie. Cílem je vytvořit program, který pomocí algoritmu dokáže rozvrhnout virtuální servery na fyzické tak, aby spotřeba byla co nejmenší. Program je implementován v jazyce C#.

Abstract

The subject of this diploma thesis is implementation of program, which is going to help reduce consumption of electrical power. The aim is create program, which according to algorithm will assign virtual servers on physical ones and in manner, that electrical consumption will be minimal. The program is implemented in C# language.

Obsah

Seznam Obrázků

1 Úvod

1.1 Předmluva

Cílem této diplomové práce je vytvořit program, který bude na základě zadaných dat schopný efektivně rozdělit virtuální servery na fyzické. Při umisťování virtuálních serverů bude kladen důraz na úsporu spotřeby elektrické energie. V této práci si tedy kladu za cíl vytvořit program, který bude minimalizovat spotřebu elektrické energie fyzických serverů. Tato práce vznikla k projektu datového centra společnosti Haug-land, spol. s.r.o.

V ideálním případě bude využíváním programu, či jeho budoucí verzí, dosaženo snížení spotřeby elektrické energie datového centra, a tím i snížení nákladů na jeho provoz.

Program je vyvinut v jazyce C#. Pro uchování a zpracování vložených informací slouží databáze SQLite. Pro práci s SQLite databází je použita knihovna pro jazyk C# System.Data.SQLite (1).

Vzhledem k neustále rostoucímu počtu datových center a výkonu serverů je zapotřebí se zamyslet nad spotřebou těchto center. Spotřeba roste díky nárůstu výkonu jednotlivých komponent. Další velký konzument energie je chladicí systém, který musí chladit stále výkonnější součástky. Dnes si již výrobci uvědomují jak je spotřeba elektrické energie důležitá, a proto nové produkty bývají i šetrnější k životnímu prostředí. Snižují jejich energetickou náročnost a zároveň je vyrábějí tak, aby byly schopny pracovat i při vyšších teplotách, čímž snižují nároky na chlazení systému.

1.2 Datové centrum

1.2.1 Definice

Datové centrum je zařízení, ve kterém jsou umístěny počítačové systémy a přidružená zařízení, jako například telekomunikační systémy a uložiště. Datové centrum obecně zahrnuje redundantní nebo záložní zdroje, redundantní datové komunikační spojení, systémy pro zajištění chlazení a protipožární zařízení a jiné další bezpečnostní prvky.

1.2.2 Úrovně Data centra

Telecommunications Industry Association je obchodní sdružení akreditované ANSI (American National Standards Institute). V roce 2005 publikovalo ANSI/TIA-942, telekomunikační standard pro datová centra, ve kterém byly stanoveny čtyři úrovně (tzv. vrstvy[1]) datových center. TIA-942 byla změněna v roce 2008 a znovu v roce 2010. *TIA-942:Data Center Standards Overview* popisuje požadavky na infrastrukturu datového centra. Nejjednodušší je TIER 1 datového centra, které je v podstatě serverovnou, dodržující základní pokyny pro instalaci počítačových systémů. Nejpřísnější úrovní je TIER 4, která je navržena tak, aby mohla hostit kritické počítačové systémy. Datové centrum je vybaveno plně redundantními podsystémy a je rozčleněno bezpečnostními zónami, do nichž je přístup zajištěn přes biometrické zabezpečení.

Zde viz Tabulka 1, je seznam jednotlivých úrovní s jejich požadavky:

Úroveň	Popis
TIER1	Datové Centrum bez redundantních prvků, garantuje 99,6% dostupnost (průměrná doba výpadku je 28,8 hodiny).
TIER2	Datové Centrum s jediným napájecím a chladícím distribučním procesem, který má na sebe navázánu podporu redundantních prvků. Datové Centrum poskytuje 99,7% dostupnost a průměrná doba výpadku je 22,0 hodin
TIER3	Datové Centrum obsahující záložní bezpečnostní systém, resp. více aktivních napájecích a chladících prvků a to včetně redundantních komponent, dostupnost dosahuje minimálně 99,98% (průměrná doba výpadku je 1,6 hodiny).
TIER4	Nejlépe zabezpečené Datové Centrum, nalezneme v něm více aktivních napájecích a chladících prvků včetně redundantních komponent a se systémem prevence výpadků s dostupností 99,99% (průměrná doba výpadku je 15 minut).

Tabulka 1 Úroveň datových center (2)

1.2.3 Energetické centrum

[1] Z anglického TIERS – definuje úroveň zabezpečení a provozní dostupnosti

Energetické centrum zajišťuje napájení všech celků a technologií instalovaných v Datovém centru. Design Energetického centra určuje výslednou třídu dostupnosti celého Datového centra. Zvolená třída - úrovně dostupnosti, je rozhodující pro návrh struktury a funkčnosti Energetického centra. Dále je zde koeficient energetické účinnosti (PUE - Power Usage Effectiveness)

$$PUE = \frac{Total\ Facility\ Energy}{IT\ Equipment\ Energy} \tag{1.1}$$

Konkrétně se jedná o poměr celkové spotřeby energie ke spotřebě IT vybavení (hardware). Tato hodnota je pak ukazatelem toho, jak efektivní je datové centrum jako celek, a také, jaký ještě zbývá prostor ke zvyšování jeho efektivity.

Ideální hodnota PUE by měla být samozřejmě 1,0, to znamená, že by veškerá spotřebovaná energie byla využita přímo jen na provoz samotného IT, ale běžně mají moderní datová centra ve světě hodnotu PUE obvykle kolem 1,4.

UPS + baterie:

Pro UPS a baterie se v projektech Datových center vyhrazuje samostatná místnost, či více místností. Místnost vyhrazená pro baterie musí obsahovat prvky řízeného prostředí – teplota, odvětrávání, vlhkost.

Motorgenerátor:

Motorgenerátor je standardně umísťován ve strojovně budovy. Dle požadavků akustické studie, nebo na přání zákazníka, je motorgenerátor instalován s kapotáží. Pokud je vyžadováno umístění vně budovy, je motorgenerátor umísťován ve venkovním provedení. Většinou se jedná o montáž v kontejneru, která umožňuje různé stupně odhlučnění.

Stejnosměrné systémy:

Stejnosměrné systémy jsou v Datových centrech instalovány pro zajištění napájení instalovaných zařízení, která nejsou napájena standardními napájecími zdroji, ale pro svou činnost potřebují stejnosměrné napětí, zpravidla 48V. Tyto stejnosměrné napájecí systémy nejsou zálohovány pomocí UPS, pouze pomocí motorgenerátoru.

Trafostanice:

V rámci platných norem a zvyklostí jsou trafostanice, určené pro napájení Datových center, stavěny s oddělenými prostory pro jednotlivá trafa. Pro kabelové rozvody jsou určeny samostatné kabelové žlaby.

1.2.4 Chlazení

Technologie chlazení je v systému Datového centra stejně kritická, jako technologie napájení. Zvolená cílová teplotní hustota (kW/rack) je určující na topologii technologie chlazení. Samotná technologie chlazení je pak členěna do těchto základních celků:

- jednotky chlazení určené pro chlazení IT infrastruktury
- venkovní chladící jednotky
- v některých případech i zásobníky chladu – nádrže.

Díky vysoké energetické náročnosti chlazení (30% až 40% z celkové spotřeby elektrické energie instalovaných IT technologií) se v praxi napájení technologie chlazení nezálohuje pomocí UPS, ale pouze pomocí motorgenerátoru. V některých instalacích, kde provozovatel požaduje vysokou dostupnost provozované IT technologie, se pomocí UPS zálohují čerpadla a ventilátory vnitřních chladících jednotek, což v případě výpadku napájení a přechodu na napájení z motorgenerátoru zajistí dostatečné chlazení IT technologií do okamžiku obnovení plného provozu chladících technologií.

1.2.5 Infrastruktura (fyzická infrastruktura)

Pro bezproblémový provoz IT technologií v Datovém centru je důležitá fyzická infrastruktura. Do této kategorie řadíme následující technické prostředky:

- zdvojená podlaha (nosnost a povrchová úprava podlahy je určena typem provozovaných IT technologií)
- oddělené kabelové žlaby pro vedení napájení a datových vedení
- rackové skříně, apod.

1.2.6 Zabezpečení

Požární bezpečnost:

Vnitřní prostory Datového centra a technologické místnosti jsou rozčleněny do samostatných požárních úseků, abychom maximálně předešli materiálním škodám v případě požáru. Pro prostory Datových sálů je používána technologie zhášecího plynu nebo vodní mlhy.

Fyzická bezpečnost:

Pro zajištění fyzické bezpečnosti musí zdi, stropy a dveře poskytovat dostatečnou ochranu proti cizímu vniknutí, či proti vnějšímu požáru. V Datovém centru musí být instalován systém kontroly vstupu a pohybu osob. Projekt datového centra by měl obsahovat kamerový systém s možností nahrávání.

Bezpečnost IT:

Zabezpečení IT infrastruktury je důležitým článkem v projektu Datového centra. Samotná definice zabezpečení IT je určena druhem provozované IT infrastruktury a poskytovaných služeb zákazníkům.

1.2.7 Monitoring

Monitoring všech technologických celků je nedílnou součástí projektu Datového centra. Pomocí instalovaných čidel a snímačů dostává obsluha všechny potřebné informace o provozních stavech. Výstupy z monitorovacích systémů slouží jako vstupy pro systém řízení budovy (BMS – Building Management system). Výstupy z monitoringu též slouží jako kritéria pro měření parametrů SLA uzavíraných se zákazníky.

2 Výběr serveru

2.1 Kritéria

Když je vybírán server (rozuměno fyzické zařízení) pro datové centrum, je zapotřebí zhodnotit několik kritérií. Zodpovědět několik otázek ohledně toho, co je od serveru očekáváno.

- Škálovatelnost hardware
- Virtualizace
- Poměr záruka/cena

2.1.1 Škálovatelnost hardware

Při výběru serveru pro datové centrum je třeba si ujasnit, jaký typ služeb bude v datovém centru provozován. Existuje mnoho druhů služeb, které mohou být provozovány; od služeb webhostingu až po komplexní cloudové služby. Pro každou službu je vhodná trochu odlišná konfigurace serveru. Pokud není zřejmé, jaká služba bude poskytována, nebo bude zákazníkům nabízen kompletní servis, pak v obou daných případech je dobré mít server takový, kde bude možno snadno vyměnit procesor, jestliže přestane dostačovat jeho výkon či bude možné přidat nebo vyměnit paměťové moduly za větší (myšleno kapacitně), pokud začne být kapacita pamětí nedostatečná.

2.1.2 Virtualizace

Virtualizace je dnes vcelku běžná věc. V dnešní době se většina serverů provozuje jako virtuální, protože vývoj technologií, rychlost, více-jadernost procesorů umožňují provozovat několik virtuálních serverů a to bez větších problémů. Většina datových center dnes provozuje na svém hardwaru některou z virtualizačních technologií (VMWare, Hyper-V, Citrix a jiné). To dovoluje maximální využití hardwaru. V každém případě musejí být servery na virtualizaci připravené. Nutno podotknou, že v dnešní době je většina serverů již připravena.

2.1.3 Poměr záruka/cena

V neposlední řadě musí být zahrnuta do úvah i cena pořizovaného hardware, a jaké benefity budou obdrženy, při koupi hardwaru od renomovaných a velkých výrobců. V potaz musí být bráno také finanční hledisko, čili jestli je bezpodmínečně nutné platit za podporu od společnosti, kde bude hardware pořizován nebo zda postačí „standartní“ služby a záruka a za ušetřenou cenu bude pod stolem připraven jeden náhradní server pro případ, kdy dojde

k technické závadě. I toto je součástí rozhodnutí, zda bude použita virtualizace, či nikoliv. Pokud bude použito virtualizace, pak není problémem vyměnit nefunkční jednotku za funkční s minimálním nebo žádným výpadkem služeb.

2.2 Vybraná sestava

Jelikož firma již několik let provozuje virtuální servery, nebyla volba serveru nikterak složitá. Při výběru hardwaru byl kladen důraz na škálovatelnost jednotlivých kusů hardware. Aby bylo možno systém rozšířit, v případě potřeby, o další CPU [2], z toho plyne výběr nejméně dvouprocesorové základní desky. V nejlepším případě učinit výběr tak, aby deska podporovala rodinu procesorů, která má v sobě obsaženu řadu různě výkonných procesorů a to i procesorů s různým počtem jader (například rodina Intel Xeon E5-2400, která čítá procesory od čtyř jádrových až po osmi jádrové, s různou variací výkonů).

Dále pak je důležitá dostatečná paměť RAM. Byla vybrána základní deska, tak že obsahuje množství paměťových slotů, minimálně však šest. Do těchto slotů mohou být instalované paměťové moduly v rozmezí 2 – 64 GB.

Dalším důležitým faktorem je zdroj elektrické energie. Zdroj byl volen s důrazem kladeným na redundanci (v případě poruchy jednoho zdroje se přepne na druhý, který je do té doby neaktivní).

2.2.1 Hardwarová sestava

Výsledná sestava je tedy následující:

Kategorie	Produkt	#
Procesor	Intel Xeon E5-2407 2,2GHz 10MB L2, 4core, 80W, LGA1356	1
Základní deska	X9DBU-iF 2S-B2, PCI-E16+E8(g3), 2GbE, 6sATA,12 DDR3-1600,IPMI	1
Paměť	8GB 1600MHz DDR3 ECC Registered 2R×4, LP(30mm), Hynix	3
Ethernet	Síťová karta Intel i350 2× GbE na základní desce	1
VGA	Nuvoton G200 WPCM450 BMC na základní desce	1
Management	IPMI 2.0 modul s KVM-over-LAN na základní desce	1
HDD/SSD	250GB VelociRaptor WD2500BHTZ 2,5" 10000RPM, 64MB	1
Skříň	SC113TQ-R700U 1U UIO 8SFF, slimCD, rPS 700W	1

Tabulka 2 Vybraná konfigurace serveru[3]

[2] CPU – Central Processing Unit

[3] Konfigurace proběhla na stránce http://www.abacus.cz

2.3 Startup sestava

Sestava, která byla použita pro rozjezd datového centra, je ovšem jiná. Ke koupi naskytla výhodná sestava blade centra od společnosti DELL Inc. Konkrétně se jedná o Dell PowerEdge M1000e Blade Server, s Fiber chanelovým datastorem, který má k sobě ještě dvě diskové expanze.

Při použití virtualizačních technologií je využíváno sítí SAN [4], jako propojovací technologie se využívá Fiber Chanel.

2.3.1 Fiber Chanel

Je vysokorychlostní síťová technologie, která běžně využívá přenosové rychlosti 2,4,8 a 16 Gb/s. Tato technologie se primárně využívá pro připojení k SAN. Fiber Chanel je standartizován v T11 Technical Committee of the International Committee for Information Technology Standards (INCITS) a v American National Standards Institute (ANSI)-accredited standards committee. I přes to, že Fiber Chanel signalizuje funkčnost na optických kabelech, je možné jej provozovat i na elektrickém rozhraní.

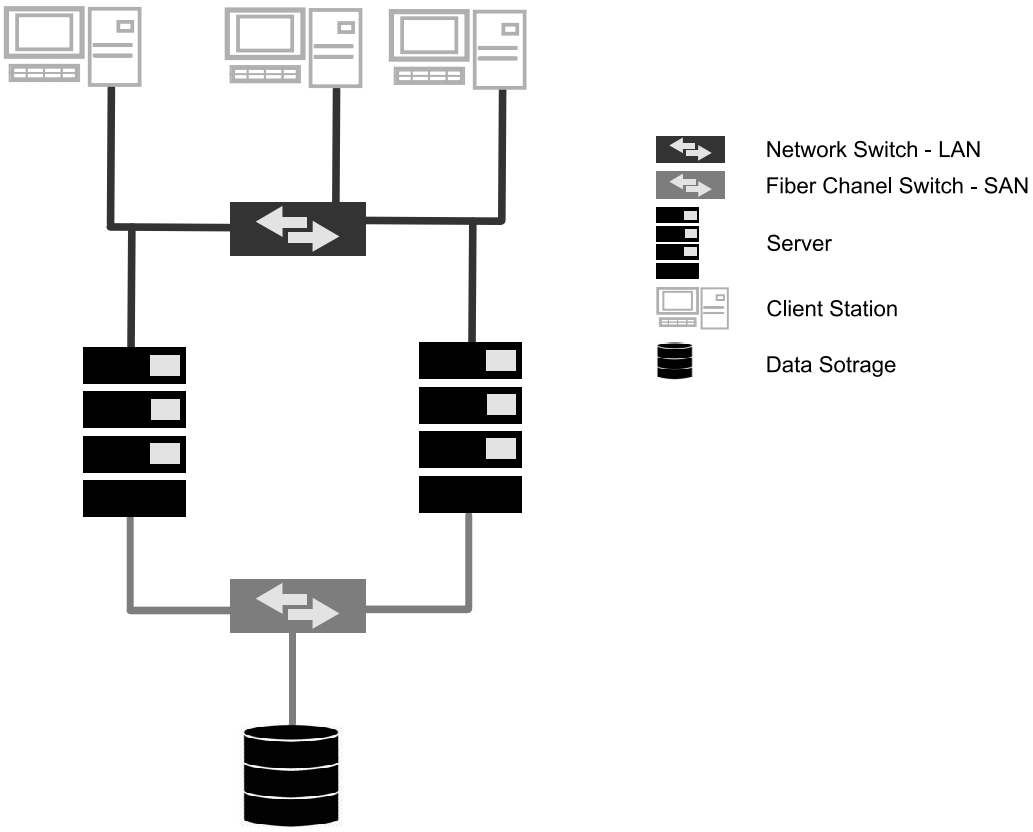
2.3.2 Storage Area Network

SAN je dedikovaná datová síť, která slouží pro připojení externích zařízení k serverům (disková pole, páskové knihovny a jiná zálohovací zařízení). SAN vznikla hlavně kvůli narůstajícím potřebám na zabezpečení a konsolidaci dat. Díky poměrně vysokým pořizovacím nákladům se SAN budují hlavně ve větších společnostech (bankovní sektor, automobilový průmysl, média), které vyžadují vysokou dostupnost svých služeb, rychlé odezvy v transakčně orientovaných aplikacích. V poslední době se však díky klesající ceně začínají malé SAN rozšiřovat i do stále menších společností. SAN zařízení tak postupně nahrazují tzv. DAS (Direct Attach Storage) technologii, kde součástí serveru byly všechny disky, či zálohovací zařízení nejčastěji připojené pomocí SCSI řadiče, či interního RAID řadiče.

SAN oproti DAS nabízí:

- Fyzické oddělení dat a serverů
- Sdílení zdrojů mezi jednotlivými servery
- Vyšší propustnost
- Definici redundantních cest ke zdrojům
- Podporu pro clusterová řešení
- Podpora pro tzv. Disaster Recovery sites

[4] SAN – Storage Area Network



Obrázek 1 Schéma SAN, LAN, FC

2.3.3 Hardwarová konfigurace Blade Centra

Blade Centrum má šestnáct šachet pro šestnáct jednotlivých Blade Serverů. V pořízeném Blade Centru se nachází čtyři Blade Servery, z toho dva jsou v provozu. Konfigurace jednoho takového Blade Serveru se nachází viz Tabulka 3:

Kategorie	Produkt	#
Procesor	Intel Xeon E5450, QuadCore 3GHz, 1333MHz 12MB cache	1
Základní deska	Dell MY736 PowerEdge M600 Motherboard	1
Paměť	Kingston 2GB CL6 ECC Registered, DDR2, 800MHz	8
Ethernet	Dual-port Broadcom NetXtreme II 5708 Gb Ethernet NIC, iSCSI	1
VGA	ATI RN50 video controller 32MB paměti, na základní desce	1
Management	OpenManage Server Administrator, Storage Services, Deployment Toolkit, Dell Server Assistant, VMedia, vKVM, IPMI 2.0 support	1
HDD/SSD	2.5" SAS 10000 RPM 73GB	2
Skříň	Dell PowerEdge M600 Blade Server	1

Tabulka 3 Konfigurace Blade Serveru

3 Optimalizační algoritmus

3.1 Předešlá práce

Existuje článek o efektivním přiřazení virtuálních serverů na fyzické (3), kde autoři definují Virtual Machine Assigment problém a dokazují, že je NP těžký. Autoři se zabývají přiřazením virtuálních serverů na identické fyzické servery, kde virtuální server nemůže být rozdělen mezi několik fyzických. Testují celkem 4 varianty, v závislosti na to, zda počet fyzických serverů je omezen či nikoliv. Dále jsou v práci rozebírány polynomiální časově aproximační schémata (PTAS a FPTAS) pro tyto problémy.

V článku (4) se autoři zabývají vytvořením modelu a posléze algoritmu, který bude rozdělovat virtuální servery na fyzické v předefinovaném poměru efektivity spotřeby elektrické energie a výkonu.

3.2 Analýza problému

Vycházím z předpokladu, že fyzické servery nejsou uniformní, tedy jinak řečeno, každý server může obsahovat jiné komponenty, jako jsou různě výkonné procesory s různými počty jader apod. Dále každý fyzický server pak také může mít různé konfigurace pamětí RAM.

Problém přiřazení virtuálního serveru na fyzický je v podstatě general assigment problém. Každý fyzický server má svůj fond zdrojů, a virtuální server má své požadavky na zdroje, proto jsem vybral pro toto přiřazování algoritmus, který řeší problém batohu. V mém případě je rozdíl oproti klasickému problému batohu takový, že tam kde se v problému batohu hledá maximální cena na nosnost batohu, já hledám minimální cenu (spotřebu elektrické energie) na zatížení fyzického serveru.

3.3 Problém batohu

Problém batohu je úloha kombinatorické optimalizace, jejímž cílem je umístění podmnožiny předmětů (předmět má specifikován svoji cenu a váhu) do přepravky omezené kapacity (nosnosti) tak, aby cena nákladu byla maximální. Název toho problému je odvozen od problému někoho, kdo je omezen na pevné velikosti batohu a musí jej naplnit nejcennějšími předměty.

Problém se často vyskytuje při alokování zdrojů, kde se bere v potaz cena. Častými obory jsou kombinatorika, teorie složitosti, kryptografie, aplikovaná matematika a počítačové vědy.

3.4 Definice

3.4.1 0 – 1 problém batohu

Nejběžnějším problémem, který se řeší je 0 – 1 problém batohu, který omezuje počet kopií x_i každého předmětu na nula nebo jedna.

Matematická definice

Maximalizuj sumu hodnot v batohu tak, aby suma jejich vah byla menší nebo rovna kapacitě batohu.

Mějme celkem n předmětů.

$$\max \sum_{i=1}^{n} v_i \, x_i \tag{2.1}$$

$$z.p.: \sum_{i=1}^{n} w_i x_i \leq W, \; x_i\{0,1\} \tag{2.2}$$

- z_1 – z_n, jsou předměty kde z_i má hodnotu v_i a váhu w_i
- x_i označuje počet kopií předmětu z_i

3.4.2 Ohraničený problém batohu

Problém v této úpravě odstraňuje omezení ohledně toho, že může být pouze jedna kopie předmětu. Zavádí ovšem počet kopií každého předmětu x_i, kde x_i je maximálně celočíselná hodnotu c_i

Matematická definice

Mějme celkem n předmětů.

$$\max \sum_{i=1}^{n} v_i \, x_i \tag{3.1}$$

$$z.p.: \sum_{i=1}^{n} w_i x_i \leq W, \; x_i\{0,1, \dots, c_i\} \tag{3.2}$$

- $z_1 - z_n$, jsou předměty kde z_i má hodnotu v_i a váhu w_i
- x_i označuje počet kopií předmětu z_i a to do počtu až c_i.

3.4.3 Neohraničený problém batohu

Odstraňuje omezení na počet předmětů každého druhu, a může být formulován, viz výše, kde jediné omezení se týká x_i , které musí být nezáporné celé číslo.

3.5 Minimalizace pomocí 0 – 1 problému batohu.

Ve své práci pro optimalizaci spotřeby serverů v datovém centru jsem potřeboval algoritmus, který rozdělí dostupné virtuální stroje na fyzické a to tak aby výsledná spotřeba byla minimální. Rozhodl jsem se použít 0 – 1 problém batohu, kde algoritmus počítá s cenami $\frac{1}{v_i}$, kde v_i je původní cena. Jelikož algoritmus ze své podstaty maximalizuje, bude teď s novými cenami celkovou sumu minimalizovat.

Důkaz: Mějme ceny $v_1, v_2, \dots, v_i$. Algoritmus vybere položky tak, že výsledná suma cen je maximální vzhledem k maximální nosnosti batohu. Jistě platí, že pokud $v_k > v_l$ poté $\frac{1}{v_k} < \frac{1}{v_l}$ a to platí pro všechny ceny v_i. Dáme tedy algoritmu řešit problém s cenami $\frac{1}{v_i}$, do výsledku pak vybereme ceny nepřevrácené. Výsledná suma pak bude minimální s maximálním využitím kapacity batohu.

3.6 Složitost 0 – 1 problému batohu

Polynomiální redukcí můžeme dojít od Úlohy SAT přes problém celočíselného lineárního programování k problému batohu. Řekneme tedy, že problém batohu náleží do třídy NP úplných úloh.

Tvrzení: Říkáme, že rozhodovací úloha U je NP úplná jestliže:

1. U je ve třídě NP
2. Každá z NP úloh se polynomiálně redukuje na U

Třída všech NP úplných úloh se značí NPC.

Zhruba řečeno, NP úplné úlohy jsou ty "nejtěžšími" mezi všemi NP úlohami.

Tvrzení: Redukce a polynomiální redukce úloh. Jsou dány dvě rozhodovací úlohy U a V. Řekneme, že úloha U se redukuje na úlohu V, jestliže existuje algoritmus, který pro každou instanci I úlohy U zkonstruuje instanci I_0 úlohy V a to tak, že:

$$I\ je\ ANO - instance\ U \Leftrightarrow I_0\ je\ ANO - instance\ V.$$

Tvrzení: NP obtížné úlohy. Jestliže o některé úloze U pouze víme, že se na ní polynomiálně redukuje některá NP úplná úloha, pak říkáme, že U je NP těžká, nebo též NP obtížná. Poznamenejme, že to vlastně znamená, že U je alespoň tak těžká jako všechny NP úlohy.

Věta: Cookova věta. Úloha SAT, splňování formulí v konjunktivním normálním tvaru, je NP úplná úloha. (5)

4 Program pro optimalizaci spotřeby

Program je implementován v jazyce C# (*vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework, později schválený standardizačními komisemi ECMA (ECMA-334) a ISO (ISO/IEC 23270). Microsoft založil C# na jazycích C++ a Java*)

Aby program mohl být spuštěn, je třeba splnit několik požadavků na systém a software:

- Operační systém Windows Vista a novější
- Microsoft .NET Framework, ve verzi 3.5.1

Další požadavky nejsou, knihovnu pro práci s SQLite databází si program přináší s sebou.

Adresářová struktura programu:

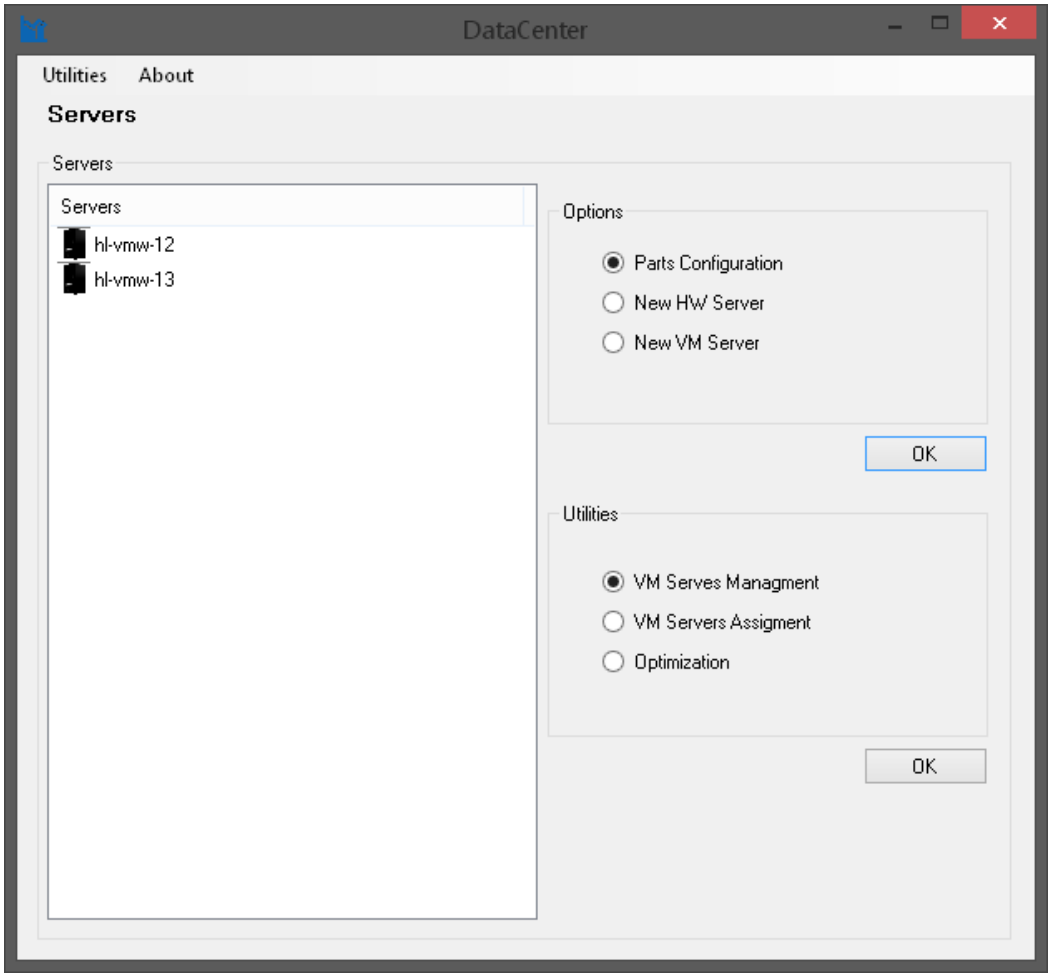
```
---/                                - Kořenový adresář
   |___files                        - Složka e interní databází + pracovní složka
          |--- test_DP.sqlite       - SQLite databáze
          |--- Server_Type.xml      - Xml soubor s typy serverů
   |--- DataCenter_DP.exe           - Spustitelný program
   |--- SQLite.Interop.dll          - Knihovna pro práci s SQLite databází
   |--- System.Data.SQLite.dll      - Knihovna pro práci s SQLite databází
```

Do složky files se kopírují soubory se zátěžovou charakteristikou jednotlivých hardwarových serverů.

4.1 Popis funkcionality

Program v sobě implementuje několik funkčních celků. Základem je hlavní okno aplikace (viz Obrázek 2), které so zobrazí po spuštění programu. Zde můžeme vidět seznam již přidaných fyzických serverů a můžeme se dostat k dalším funkcím programu.

Po dvojkliku na fyzický server v seznamu se nám ukážou informace o něm, levým tlačítkem vyvoláme kontextovou nabídku, která umožňuje editovat server, získat o něm informace, nebo jej smazat.



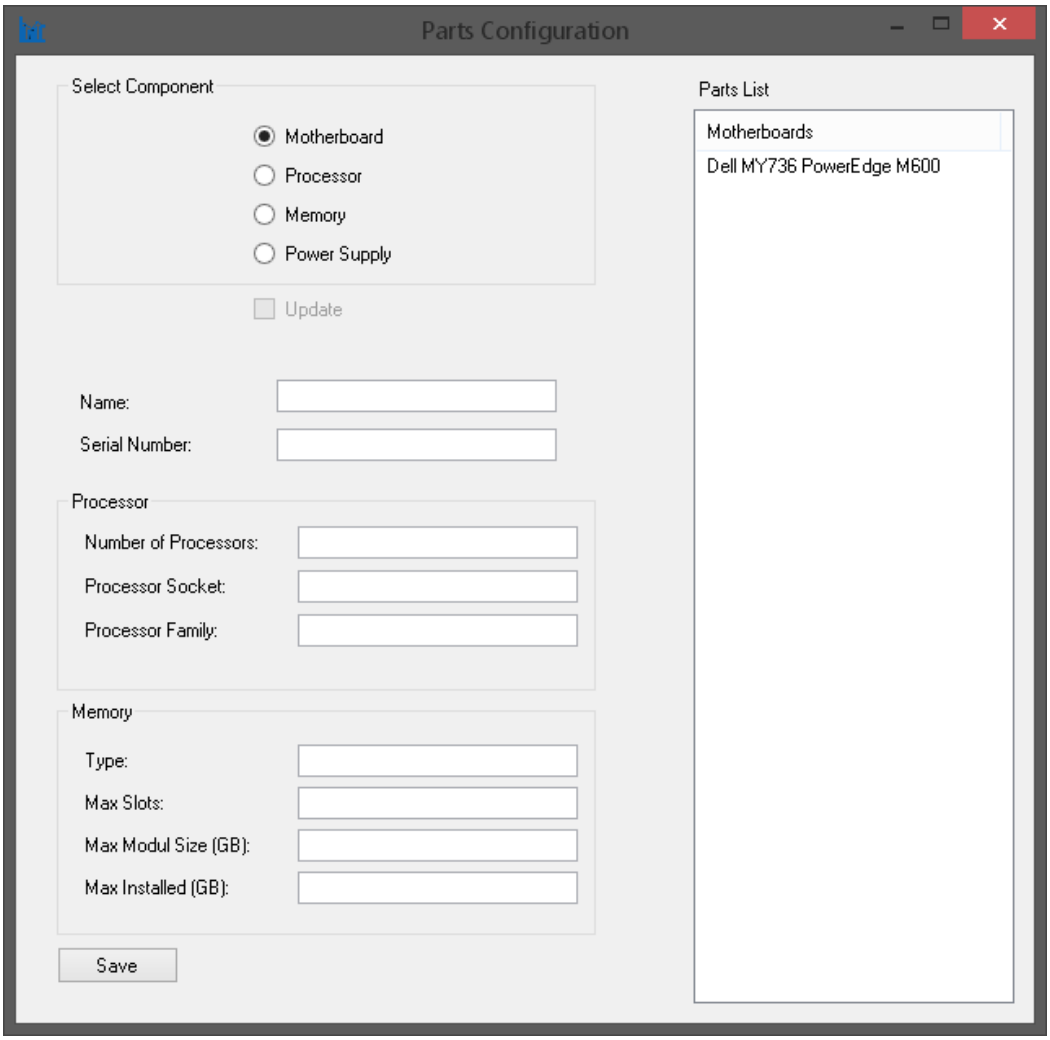
Obrázek 2 Hlavní okno programu

4.2 Konfigurace součástek

Toto je část, ve které se provádí konfigurace jednotlivých součástí fyzického serveru. Uživatel si zde může nakonfigurovat tyto součástky:

- Základní desku
- Procesor
- Paměti
- Zdroj napětí

Raido buttonem si uživatel vybere, jakou součástku si přeje přidat, v pravém seznamu vidí součástky, které již v minulosti vytvořil a po dvojkliku na ně je může upravit a znovu uložit.



Obrázek 3 Nabídka pro konfiguraci součástek

4.3 Konfigurace fyzického serveru

Další funkcí (viz Obrázek 4) je vytvoření fyzického serveru. Server se konfiguruje ze součástek vytvořených dříve. Uživatel si zvolí základní desku a na základě tohoto výběru a hodnot zadaných u té konkrétní desky se vyberou procesory, které jsou s deskou kompatibilní (výběr probíhá na základě socketu procesoru). Stejně se vyberou paměti, ty se vyselektují na základě typu paměti, který deska podporuje.

Zde se společně s jednotlivými součástkami nahrává též zátěžová charakteristika serveru. Tato charakteristika je požadovaná ve formátu XML souboru, kde jsou uvedeny hodnoty pro každé procento vytížení procesoru, to znamená hodnoty pro 0 – 100 % vytížení. Tyto hodnoty jsou uzavřené v tagu <Vlaue>XXX</Value>

Obrázek 4 Konfigurace fyzického serveru

4.4 Konfigurace virtuálního serveru

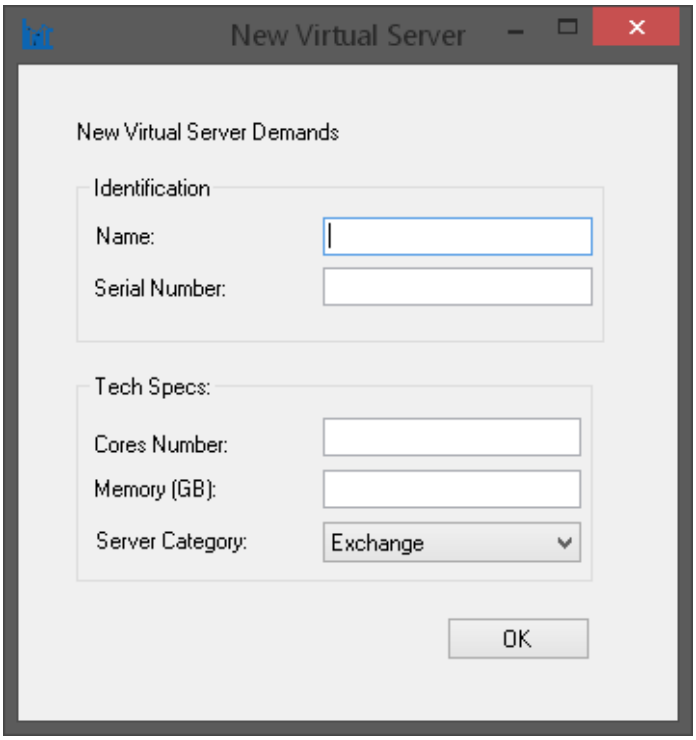
Další funkcí je nadefinování virtuálního serveru (viz Obrázek 5). Každý virtuální server má svoji identifikaci, a dále své požadavky na přidělení zdrojů.

Požadavky se týkají počtu jader, která budou virtuálnímu serveru přidělena. Dalším požadavkem je požadavek na alokaci paměti RAM z fyzického serveru. Posledním parametrem je typ serveru. Typ je předem definován a načítá se ze souboru Server_Type.xml. Jeho struktura vypadá takto:

```
<Servers>
   <Type>Exchange</Type>
   <Value>0,90</Value>
   <Type>File Server</Type>
   <Value>0,25</Value>
   <Type>Firewall</Type>
   <Value>0,25</Value>
   <Type>SQL Server</Type>
   <Value>0,75</Value>
   <Type>Application Server</Type>
   <Value>0,50</Value>
</Servers>
```

Kde v tagu `<Type>XXX</Type>` se nachází pojmenování kategorie serveru, které se zobrazuje v okně konfigurace a hodnota v tagu `<Value>XXX</Value>` je uložena do databáze pokud je daná položka vybraná.

Hodnota určuje, jak moc virtuální server vytěžuje prostředky jemu přidělené. Hodnoty jsou vždy z rozmezí 0,1 – 1,0. Tato čísla vycházejí z praxe a dlouhodobých zkušeností firmy Haug-land, spol. s.r.o.



Obrázek 5 Konfigurace virtuálního serveru

4.5 Přiřazení virtuálních serverů.

Další možnost v tomto programu je zobrazení přiřazení virtuálních serverů (viz Obrázek 6). Zobrazení závislostí je zde formou stromové struktury, kde rodič je fyzický server, a jeho potomci jsou virtuální servery, k němu přiřazené. Ve spodní části okna jsou vidět nepřiřazené servery.

Kliknutím pravým tlačítkem myši na virtuální server se zobrazí kontextová nabídka s možnostmi:

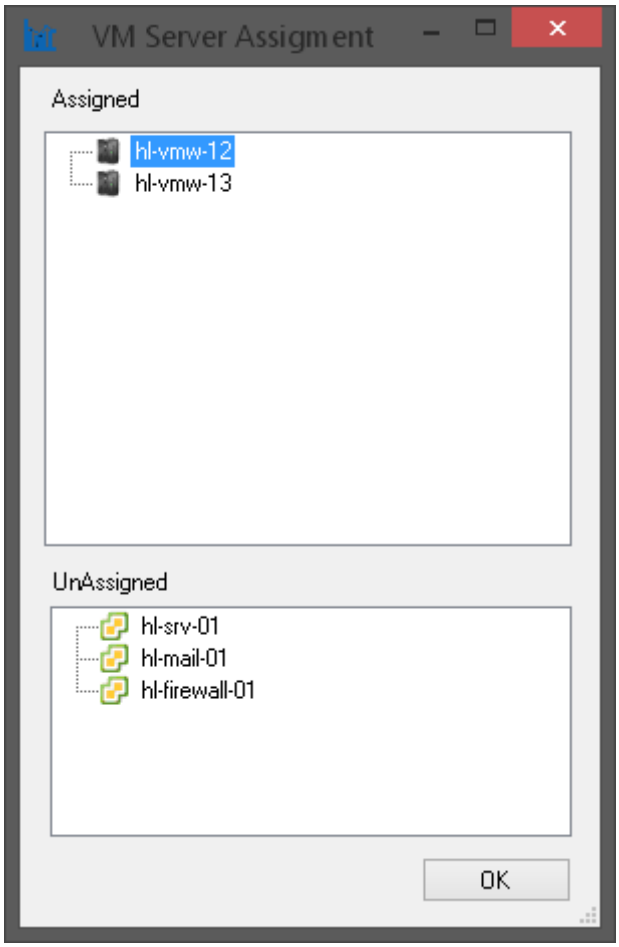
- zobrazit informace
- editovat
- smazat
- označit jako nepřiřazený

V nabídce zobrazit informace se ukáže dialogové okno s informacemi o virtuálním serveru a jeho požadavcích.

Možnost editovat virtuální server zobrazí okno pro vytváření virtuálních serverů a předvyplní informace z databáze.

Nabídka smazat odstraní záznam z interní databáze aplikace. Možnost označit jako nepřiřazený (unassigned) nastaví server jako nepřiřazený. Hodnota ve sloupci „hw_server_assigment“ bude nastavena na „-1“.

V nabídce nepřiřazených virtuálních serverů je také možné vyvolat kontextovou nabídku, která obsahuje, podobně jako viz výše, možnosti zobrazit informace; editovat a smazat.



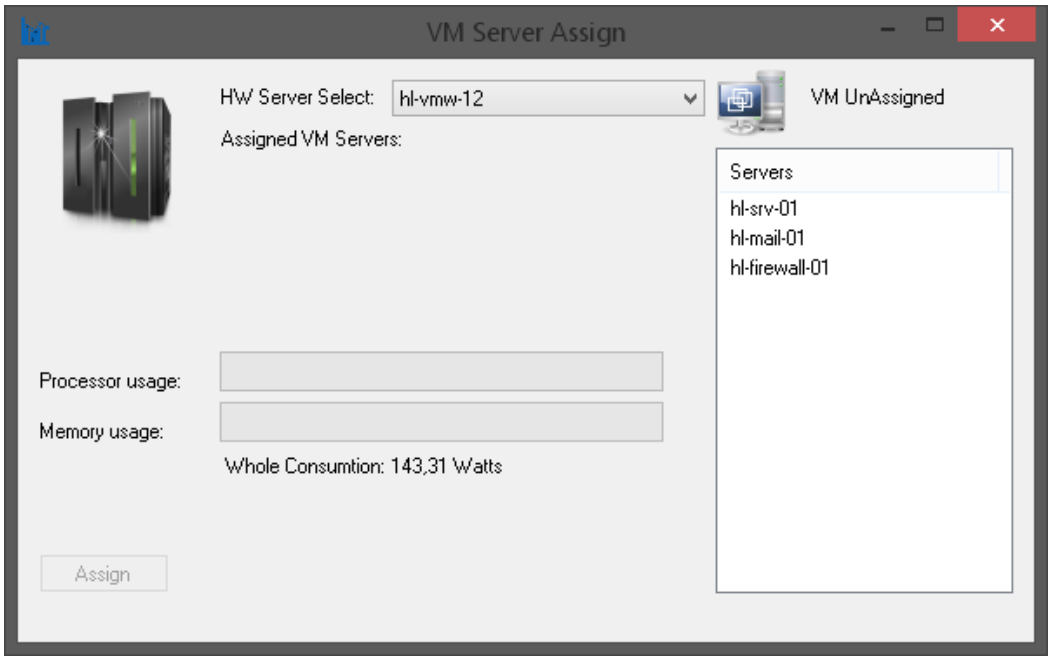
Obrázek 6 Přiřazení virtuálních serverů

4.6 Manuální přiřazení virtuálních serverů

Funkce manuálního přiřazení virtuálního serveru funguje tak, že v pravé části okna máme seznam nepřiřazených virtuálních serverů. V comboboxu se nachází seznam hardwarových serverů.

Dvojklikem na jeden z virtuálních nepřiřazených serverů program otestuje, co se stane při pokusu přidat tento virtuální server na vybraný fyzický. Pokud jsou na fyzickém serveru dostatečné prostředky, program vypíše pod combobox s vybraným fyzickým serverem, že je možné sem zvolený virtuální server přiřadit a vypíše, na kolik Wattů vzroste celková spotřeba daného hardwarového serveru. Tlačítkem Assign se daný virtuální server přiřadí ke zvolenému fyzickému. Pokud na zvolený hardwarový server není možné přiřadit zvolený virtuální server, program vypíše, z jakého důvodu to není možné. Pokud jsou volné paměťové sloty, pak vypíše kolik je volných, pokud nejsou, doporučí upgrade. Pokud není dostatečný výkon procesoru a je volný slot, doporučí instalaci dalšího procesoru, pokud ne, pak doporučí upgrade.

Dva progressbary ukazují zaplněnost fyzického serveru.

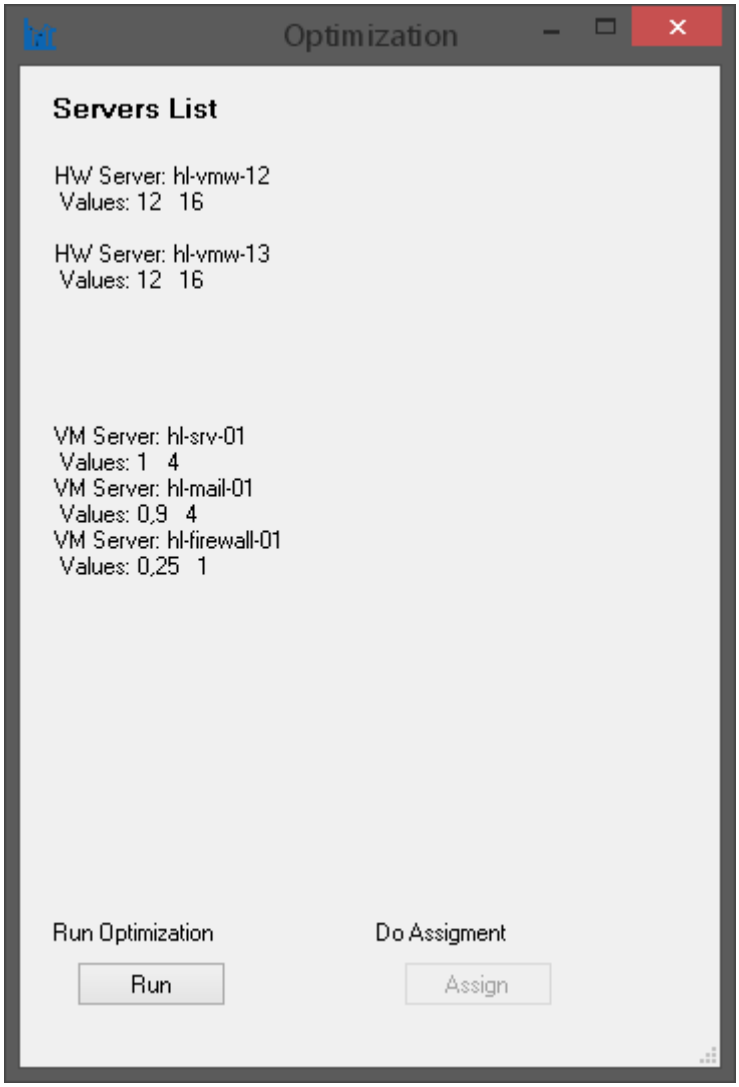


Obrázek 7 Manuální přiřazení serverů

4.7 Automatické přiřazení virtuálních serverů

Další funkcí je funkce automatického přiřazení virtuálních serverů na fyzické.

Po spuštění okna (Obrázek 8) vidíme seznam hardwarových serverů s jejich kapacitami, první hodnota je výsledkem součinu hodnot $počet\_jader * taktovací\_frekvence$, druhá hodnota znamená celkovou kapacitu paměti, tedy $počet\_slotů * kapcita\_paměťového\_modulu$. Dále je zde seznam virtuálních serverů, kde první hodnota je $pčet\_jader * typ\_serveru$ a druhá hodnota požadovaná kapacita paměti.

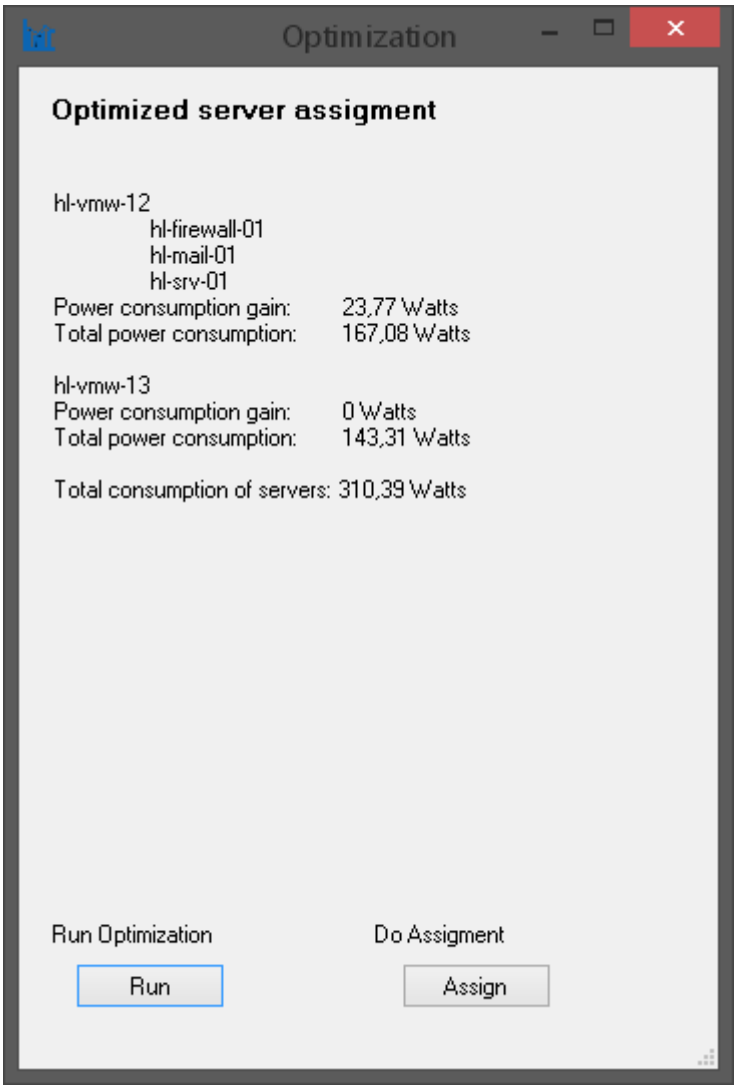


Obrázek 8 Automatické přiřazení virtuálních serverů před spuštěním

Po kliknutí na tlačítko Run, se spustí optimalizační algoritmus, výše popsaný problém batohu, který minimalizuje cenu, zde cenu reprezentuje přírůstek spotřeby elektrické energie. Ten provede rozmístění virtuálních serverů na fyzické tak, aby co nejlépe využil kapacitu fyzických serverů a přitom minimalizoval spotřebu.

Výsledné navržené přiřazení program vypíše do okna společně se spotřebou jednotlivých fyzických serverů. Dále program vypíše předpokládanou celkovou spotřebu všech fyzických serverů. Pokud zbydou nepřiřazené virtuální servery, program je následně vypíše společně s jejich celkovými požadavky na alokaci paměti, a také vypíše součet náročnosti serveru násobený počtem jader. (Obrázek 9)

Tlačítkem Assign se provede přiřazení, tak jak bylo programem navrženo.



Obrázek 9 Automatické přiřazení serverů po spuštění

4.8 Databáze

Veškeré informace se ukládají do SQLite databáze (*SQLite je softwarová knihovna, navržená D. Richardem Hippem, která implementuje soběstačný, bez-serverový, s nulovou konfigurací, transakční SQL databázový stroj. SQLite je nejrozšířenější SQL databázový stroj na světě. Zdrojový kód pro SQLite je šířen pod licencí public domain*[5]. (6) D. Richard Hipp ji navrhl v roce 2000, když pracoval na kontraktu pro Americké námořnictvo, na softwaru pro odpalovače naváděných raket).

Tato databáze má vlastní tabulku pro každou jednotlivou součástku. Databáze také obsahuje tabulky jak pro hardwarové servery, tak pro ty virtuální.

Databáze obsahuje tyto tabulky[6]

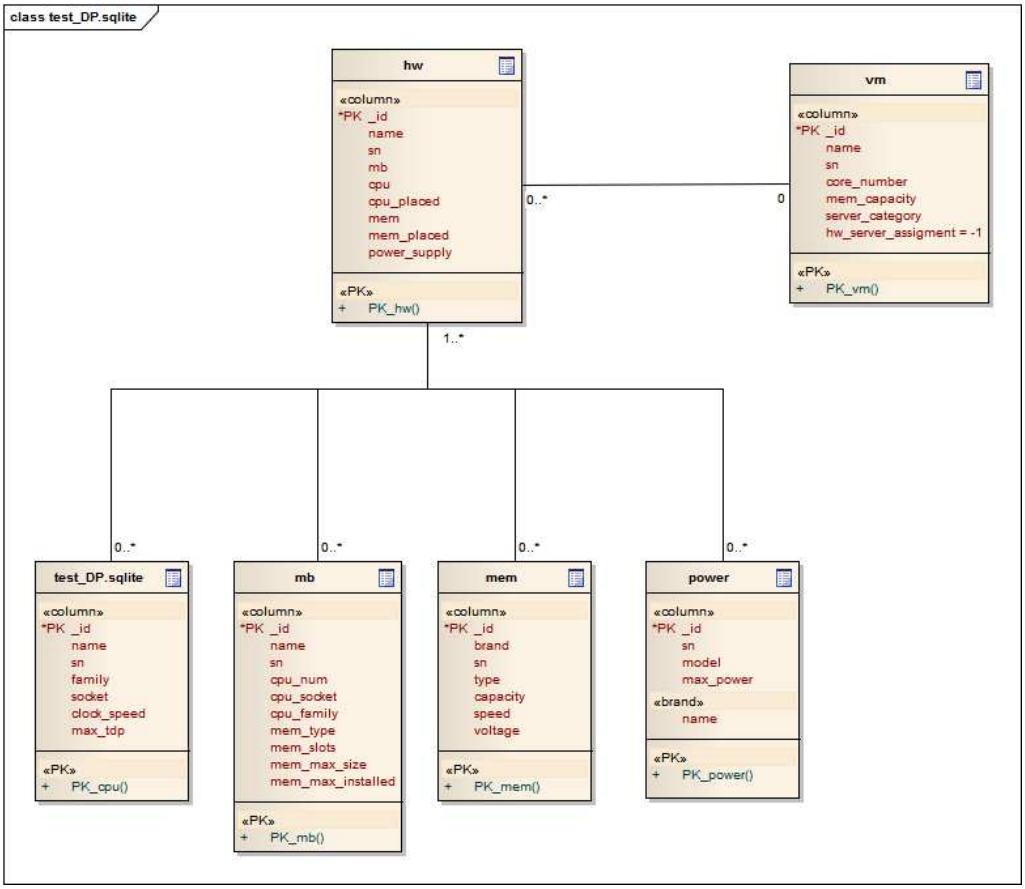
- cpu – zde jsou uloženy informace o procesorech
- hw – zde jsou uloženy informace o hardwarových serverech
- mb – tabulka, v níž jsou uloženy základní desky
- mem – zde jsou uloženy informace o pamětech
- vm – tabulka obsahující potřeby virtuálních serverů
- power – tato tabulka obsahuje záznamy o napájecích zdrojích

Dále je zde jedna tabulka „system“, která obsahuje verzi databázového schématu. Po startu programu se vyčte hodnota z této tabulky, tato hodnota určuje verzi databáze a porovná se s naposledy uloženou, použitou verzí databáze. V případě, že se neshodují, provede se ugrade na novější verzi.

[5] Tento pojem znamená, že autor díla se rozhodl, že dovolí svoje dílo volně užívat, bez nároku na další ochranu díla.

[6] Jeden řádek v tabulce znamená jednu konkrétní součástku/server

4.8.1 Struktura tabulek SQLite databáze



Obrázek 10 ER-diagram databáze

CPU – create statement

```
CREATE TABLE "cpu" (
"_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL,
"name" VARCHAR,
"sn" VARCHAR,
"family" VARCHAR,
"socket" VARCHAR,
"clock_speed" REAL,
"max_tdp" REAL, "cores" INTEGER);
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
name	– slouží k identifikování procesoru, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name, informativní zamýšlená pokud máme třeba několik revizí
family	– sloupec označuje, do které rodiny procesorů daný procesor patří (např.: E5-2400)
socket	– označení pro patici[7] procesoru, slouží jako klíč pro kompatibilitu se základní deskou
clock_speed	– označuje taktovací rychlost procesoru je zadávána v GHz
max_tdp	– maximální termodynamický výkon procesoru, udává se ve Wattech

[7] Patice – konektor na základní desce určený pro připojení procesorů

HW – create statement

```
CREATE TABLE "hw" (
"_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL
, "name" VARCHAR,
"sn" VARCHAR, "source" INTEGER, "characteristic"
TEXT, "mb" INTEGER,
"cpu" INTEGER,
"cpu_placed" INTEGER,
"mem" INTEGER,
"mem_placed" INTEGER,
"power_supply" INTEGER);
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
name	– slouží k identifikování fyzického serveru, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name, informativní
mb	– ve sloupci je uložena hodnota odkazující na základní desku (_id základní desky)
cpu	– ve sloupci je uložena hodnota odkazující na procesor (_id procesoru)
cpu_placed	– hodnota říká kolik procesorů je v serveru osazeno
mem	– ve sloupci je uložena hodnota odkazující na paměťový modul (_id pamětí)
mem_palced	– hodnota říká kolik paměťových modulů je v serveru osazeno
power_supply	– hodnota odkazuje na použitý napájecí zdroj (_ id zdroje)

MB – create statement

```
CREATE TABLE "mb" (
"_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL
, "name" VARCHAR,
"sn" VARCHAR,
"cpu_num" INTEGER,
"cpu_socket" VARCHAR,
"cpu_family" VARCHAR,
"mem_type" VARCHAR,
"mem_slots" INTEGER,
"mem_max_size" INTEGER,
"mem_max_installed" INTEGER);
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
name	– slouží k identifikování základní desky, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name, informativní zamýšlená pokud máme třeba několik revizí
cpu_num	– hodnota určuje kolik cpu lze do desky osadit
cpu_socket	– označení pro patic procesoru, slouží jako klíč pro kompatibilitu s procesorem
cpu_family	– označuje rodinu procesorů, které jsou deskou podporovány
mem_type	– označuje typ pamětí, které jsou deskou podporovány
mem_slots	– hodnota udává počet paměťových slotů
mem_max_size	– maximální velikost paměťového modulu, který je deskou podporován udává se v GB
mem_max_installed	– maximální hodnota paměti, která může být do desky naistalována

MEM – create statement

```
CREATE TABLE "mem" (
"_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL
, "brand" VARCHAR,
"sn" VARCHAR,
"type" VARCHAR,
"capacity" REAL,
"speed" INTEGER,
"voltage" REAL);
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
brand	– slouží k identifikování paměti, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name, informativní zamýšlená pokud máme třeba několik revizí (verzí)
type	– typ pamětí, slouží k určení kompatibility se základní deskou
capacity	– velikost paměťového modulu, udává se v GB
speed	– pracovní frekvenci zadává se v MHz
voltage	– napětí, pod kterým paměti pracují, udává se ve V

POWER – create statement

```
CREATE TABLE "power" (
"_id" INTEGER PRIMARY KEY  AUTOINCREMENT  NOT NULL
, "brand" VARCHAR,
"sn" VARCHAR,
"model" VARCHAR,
"max_power" INTEGER);
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
brand	– slouží k identifikování procesoru, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name, informativní zamýšlená pokud máme třeba několik revizí
model	– označení modelové řady
max_power	– maximální výkon zdroje, udává se ve Wattech

VM – create statement

```
CREATE TABLE "vm" (
"_id" INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL ,
"name" VARCHAR,
"sn" VARCHAR,
"core_number" INTEGER,
"mem_capacity" REAL,
"server_category" REAL,
"hw_server_assigned" INTEGER DEFAULT (-1) );
```

Popis sloupců:

_id	– jednoznačný identifikátor v rámci tabulky, automaticky se inkrementuje a slouží jako primární klíč.
name	– slouží k identifikování virtuálního serveru, uchopitelný název pro člověka
sn	– doplňková informace ke sloupci name
core_number	– definuje, kolik jader bude alokováno pro daný virtuální server
mem_capacity	– určuje nároky na alokaci paměti virtuálního serveru
server_category	– číslo určující náročnost virtuálního serveru (jak moc daný server vytěžuje jemu přidělené prostředky[8])
hw_server_assigment	– určuje, ke kterému fyzickému serveru je daný virtuální server přiřazen, DEFAULT hodnota -1 znamená nepřiřazne

[8] Předem definované kategorie – vychází z dlouhodobého pozorování firmy

4.9 Popis algoritmu pro automatické přiřazení

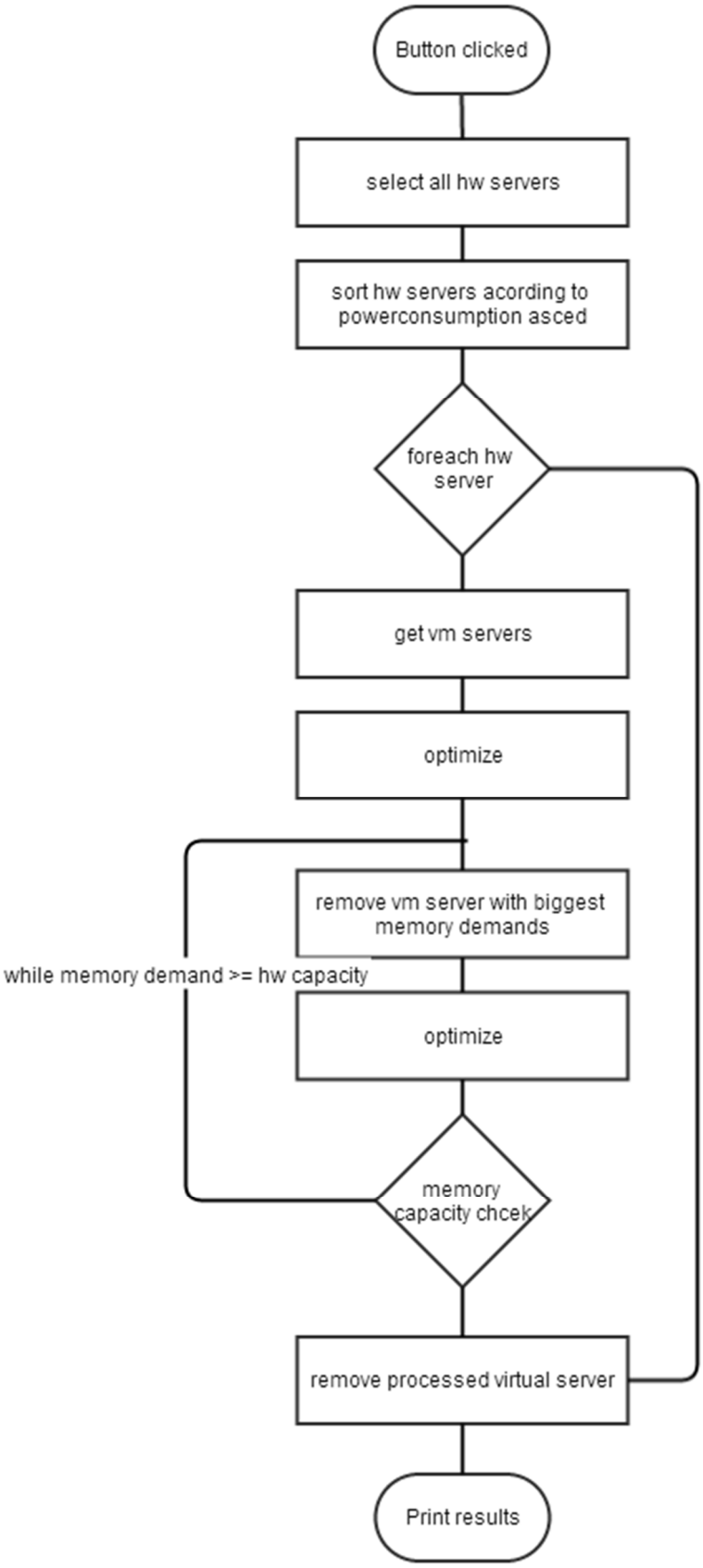
Algoritmus automatického přiřazení využívá algoritmu, pro řešení problému batohu. Po otevření optimalizačního okna je zde jedno aktivní tlačítko, Run, po stisku tohoto tlačítka se uvede v činnost algoritmus, jehož průběh viz Obrázek 11.

Po stisku tlačítka se vyberou všechny fyzické servery uložené v databázi v tabulce *hw*. Tyto servery se seřadí sestupně podle spotřeby v klidovém stavu. Poté se pro každý fyzický server provede přiřazení virtuálních serverů zavoláním metody `callOptimizer(vmservers_to_place),` kde metoda přijímá seznam virtuálních serverů, které se má pokusit přidělit.

Po návratu z této metody se provede kontrola, zda paměťové nároky takto vybraných virtuálních serverů nepřekračují kapacitu fyzického serveru. Pokud nepřekračují, je vše v pořádku a virtuální servery se označí jako zpracované a pokračuje se s dalším fyzickým serverem a seznamem virtuálních serverů, zmenšeným o již zpracované virtuální servery. Jestliže jsou paměťové nároky příliš velké, odebere se server s nejmenšími paměťovými nároky a spustí se znovu metoda pro optimalizaci. To se provádí tak dlouho, dokud existuje virtuální server, který by bylo možné přiřadit.

Zde je ukázka metody, která vytváří seznam virtuálních serverů, které budou přiřazovány na konkrétní fyzický server.

```
private List<Item> prepareVmServers(KeyValuePair<string, double[]> hwserver)
{
   double[] power_usage;
   powerHW.TryGetValue(hwserver.Key, out power_usage);
   List<Item> vmservers_to_place = new List<Item>();
   foreach (KeyValuePair<string, double[]> vmserver in serversVM)
   {
       if (!globalList.Contains(vmserver.Key)&&!tempGlobalList.Contains(vmserver.Key))
       {
          double vmdemand_cpu = vmserver.Value[0] * hwserver.Value[2];
          double index = (vmdemand_cpu / (hwserver.Value[0] - (1.0f *
hwserver.Value[2]))) * 100;
          double vmdemand_mem = vmserver.Value[1];
          int upindex = (int)Math.Ceiling(index);
          int downindex = (int)Math.Floor(index);
          if (upindex == downindex)
          {
             downindex--;
          }
          double price = upindex * (((power_usage[upindex] + (3.5 * hwserver.Value[5]))
- (power_usage[downindex] + (3.5 * hwserver.Value[5]))) * hwserver.Value[4]);
          vmservers_to_place.Add(new Item(vmserver.Key, upindex, price, vmdemand_mem));
       }
      }
      return vmservers_to_place;
     }
```



Obrázek 11 Vývojový diagram automatického přiřazovacího algoritmu

4.10 Popis algoritmu pro manuální přiřazení

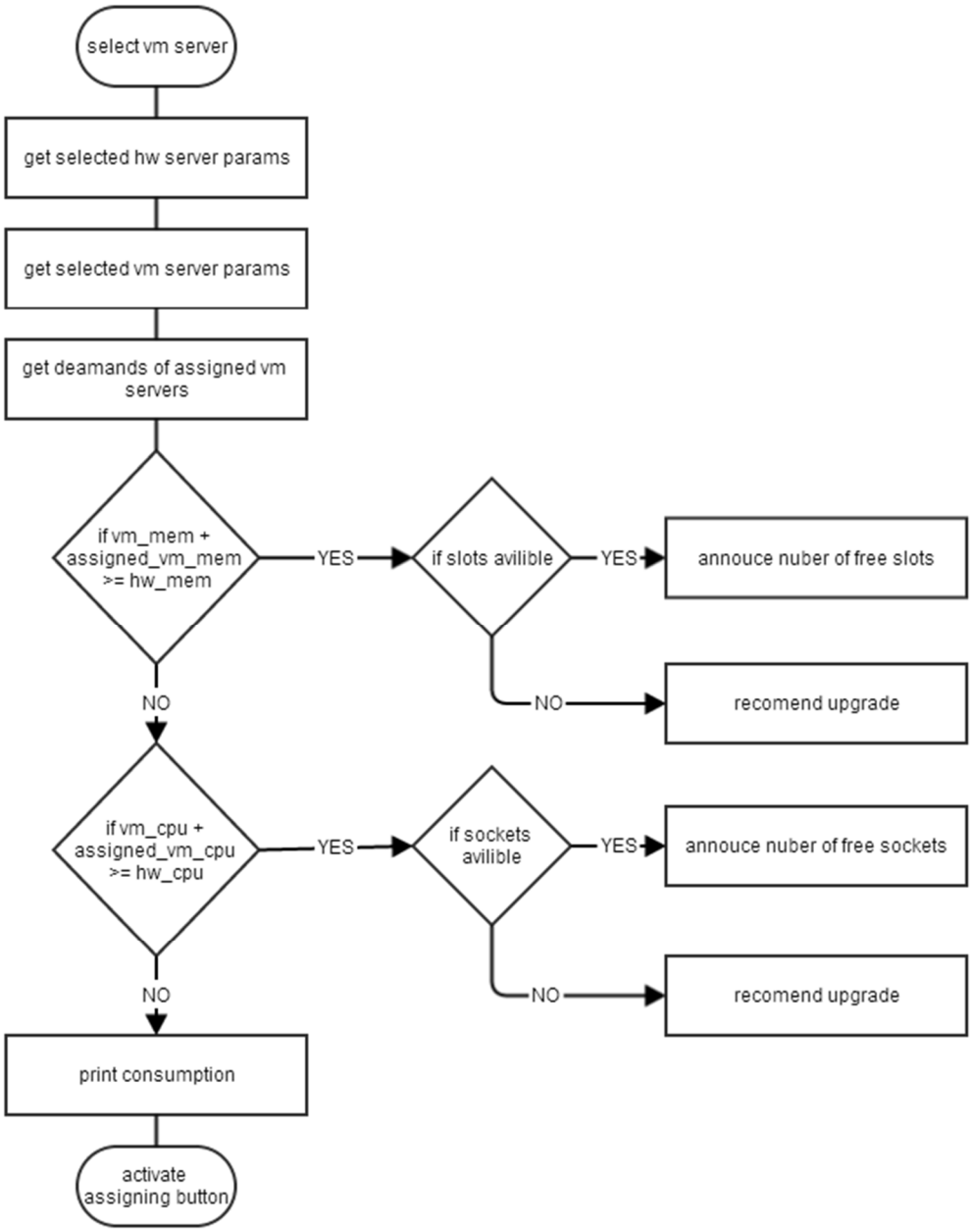
Algoritmus pro manuální přiřazení neobsahuje, na rozdíl od automatického algoritmu, žádný řešící algoritmus. Jeho průběh viz Obrázek 12.

Manuální algoritmus pro přiřazení vezme parametry vybraného fyzického serveru a požadavky na zdroje virtuálního serveru. Následně si algoritmus vyčte všechny nároky již přiřazených virtuálních serverů. Dále se algoritmus pokusí přiřadit vybraný virtuální server na fyzický a kontroluje, zda je toto umístění možné.

Je provedena kontrola, zda nebyly překročeny paměťové nároky, v případě, že byly, zkontroluje počet volných paměťových slotů. Jestliže existují volné paměťové sloty, uvědomí uživatele o tom, že umístění virtuálního serveru na fyzický není možné, ale jsou volné paměťové bloky, které mohou být obsazeny. V případě, že volné paměťové sloty nejsou k dispozici, taktéž o tom uvědomí uživatele a doporučí upgrade paměťových modulů.

Dále se provede kontrola, jestli nebyl překročen výkon instalovaných procesorů. Pokud byl, zkontroluje se počet instalovaných procesorů s počtem patic pro procesory na základní desce. Pokud existuje možnost přidat procesor, pak je o tom uživatel vyrozuměn. V případě, že další procesory není možné přidat, pak je uživateli doporučen upgrade.

V případě, velikost paměti a výkon procesorů dostačuje, pak je uživatel informován o tom, že vybraný virtuální server je možné umístit na zvolený fyzický a zároveň vypíše, na kolik se zvedne předpokládaná spotřeba.



Obrázek 12 Vývojový diagram algoritmu pro manuální přiřazování

4.11 Měření spotřeby

V celém objektu je velký počet elektrických zařízení, která spotřebovávají elektrickou energii. Její kompletní spotřeba je měřena zařízením SDS MICRO Light2

4.11.1 SDS MICRO Light2

Modul má nahrán firmware, který nabízí webové rozhraní, řadu komunikačních protokolů pro vyčítání informací (web, xml, txt, SNMP atd.) a modul můžete programovat vlastním SDS-C programem. Navíc jsou pravidelně vydávány nové verze firmware s novými funkcemi.

Připojení k modulu je možné přes pájecí body (rozteč 2.54mm), modul je určen k vestavbě do dalšího zařízení nebo jen k připojení vodičů na pájecí body.

Funkce:

- webový teploměr (sběrnice 1-Wire),
- měření spotřeby elektrické energie (vstupy pro S0),
- binární opto-vstupy,
- výstup pro ovládání dvou externích relé,
- funkce IP watchdog,
- 4x napěťový vstup 0-30Vdc (rozsah lze dodatečně rozšířit).

Pomocí vlastního SDS-C programu je možné realizovat jakékoliv řídící funkce, nebo odesílat data na vlastní portál atd.

Verze "LIGHT 2" má osazenou paměť DATAFLASH. Může tak být využit velký paměťový prostor pro SDS-C program a pro vlastní webovou stránku.

4.11.2 Měření spotřeby fyzických serverů

Pro měření spotřeby elektrické energie fyzického serveru jsem použil software od společnosti VMWare, VMware vSphere Client, kde je možné v záložce Performance, sledovat, jak vytížení jednotlivých periferií, tak spotřebu jednotlivých fyzických serverů.

Abych zjistil spotřebu elektrické energie, vytvořil jsem na fyzickém serveru v datovém centru, jeden virtuální server a přiřadil jsem mu maximu zdrojů serveru hardwarového.

Parametry virtuálního serveru:

Operační systém:	Linux CentOS version 6.5
Počet jader:	4
Alokovaná paměť (GB):	4

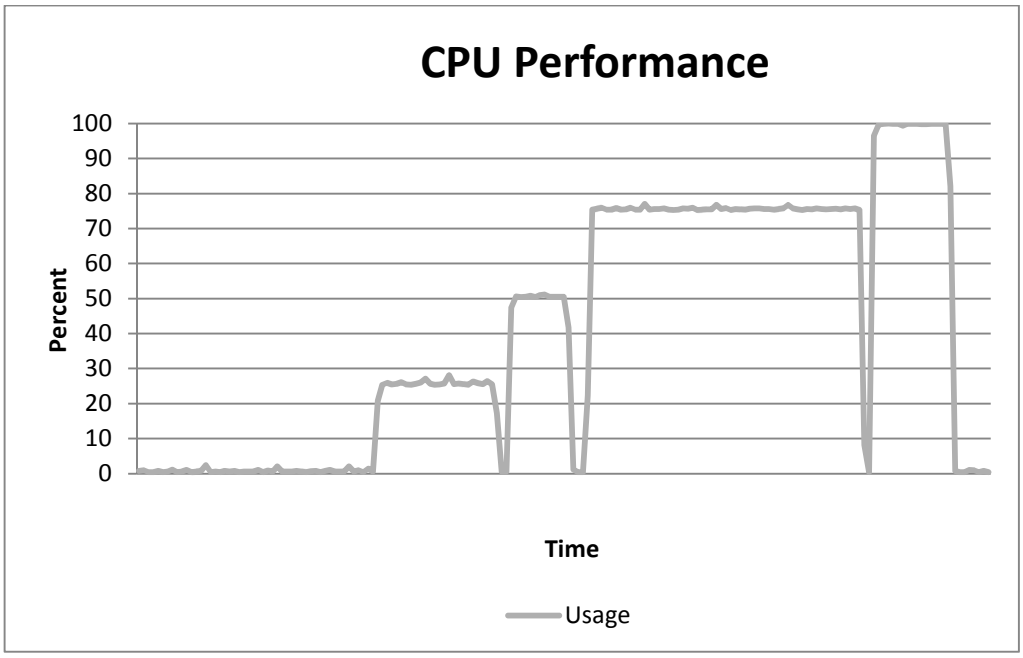
Pro vytížení procesorů jsem použil benchmarkovací nástroj SysBench – konkrétně jsem spustil test cpu:

```
sysbench --test=cpu --cpu-max-prime=200000 --num-threads=X run
```

`x` – označuje počet vláken, `x` je z rozmezí `1 - 4`

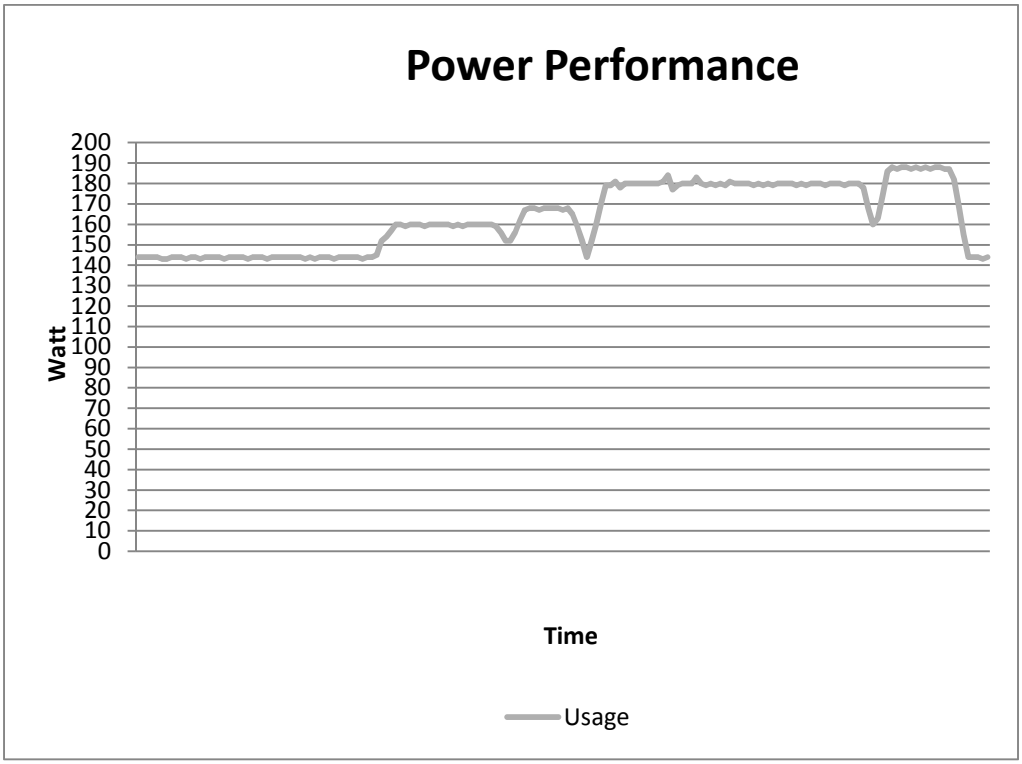
Pomocí počtu vláken jsem byl schopen vytížit systém na 25% pro jedno vlákno, na 50% pro dvě vlákna, na 75% pro 3 vlákna a na 99,8% pro vlákna čtyři (viz Obrázek 13). Výsledky jednotlivých testů jsou k vidění v příloze číslo 8.1.

Naměřenou spotřebu je možno vidět na Obrázek 14



Obrázek 13 Graf vytížení procesoru[9]

[9] Exportováno z programu VMware vSphere Client



Obrázek 14 Graf spotřeby elektrické energie při vytížení procesoru

SysBench (7)

SysBench je modulární cross-platformní a multi-vláknový benchmarkovací nástroj pro vyhodnocení parametrů systému, které jsou důležité pro běh databázového stroje pod těžkou zátěží

Myšlenka stojící za touto benchmarkovací sadou nástrojů, je získat dojem o výkonosti bez nutnosti nastavovat komplexní databázové benchmarkery nebo dokonce bez nutnosti instalovat databázový stroj.

Současné nástroje umožňují otestovat následující systémové parametry:

- Výkonost při souborových I/O operacích
- Výkonost plánovače
- Alokaci paměti a přenosové rychlosti
- Výkonost implementaci POSIX-ových vláken

Z naměřených hodnot jsem metodou lineární interpolace dostal spotřebu při každém procentu vytížení procesoru (Pro účely práce vycházím z předpokladu, že spotřeba je blízká lineární).

Lineární interpolace

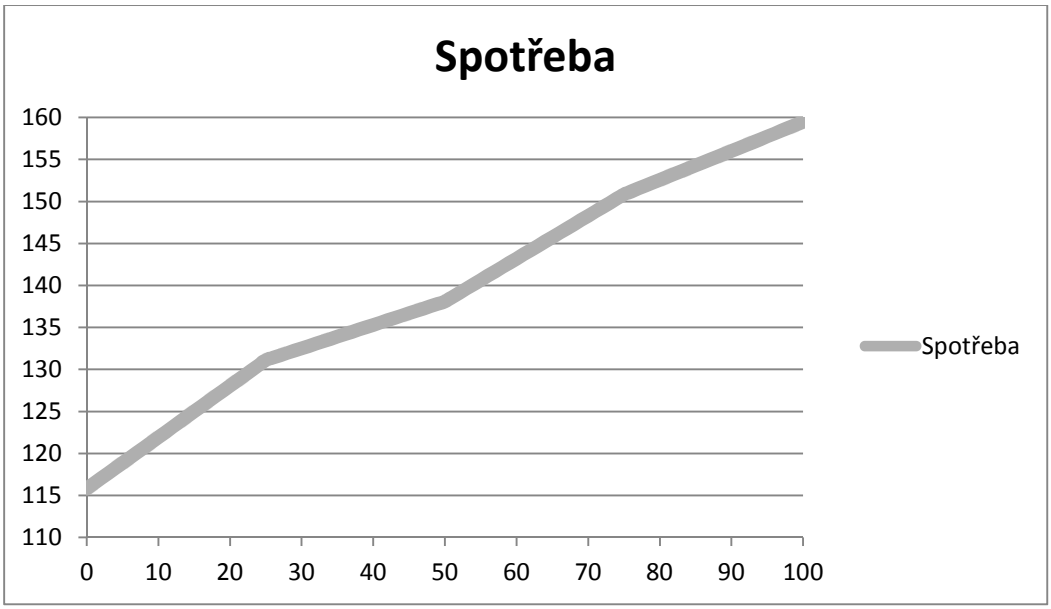
je metoda prokládání křivek za použití lineárních mnohočlenů. Této metody se často využívá v matematice (konkrétně v numerické analýze) a v početních aplikacích včetně počítačové grafiky. Jedná se o jednoduchou formu interpolace.

Pokud jsou dány dva známé body souřadnicemi (x_0, y_0) a (x_1, y_1), lineární interpolace je přímka mezi těmito dvěma body. Pro hodnotu x je interval (x_0, x_1). Hodnota y podél přímky je dána rovnicí:

$$\frac{y - y_0}{y_1 - y_0} = \frac{x - x_0}{x_1 - x_0} \tag{4.1}$$

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{4.2}$$

Díky systému sběru dat, který ve firmě připravujeme, se budou hodnoty zpřesňovat postupem času, jak budeme moci odměřovat systém v různých situacích.



Obrázek 15 Lineární interpolace spotřeby z naměřených dat

5 Diskuze

Ve své diplomové práci jsem implementoval program, který by měl pomoci optimalizovat spotřebu elektrické energie datového centra.

Program optimalizuje přiřazení virtuálních serverů na fyzické. Vzhledem k nedávnému dokončení stavby datového centra se začíná s postupným zprovozňováním. Přesnější výsledky ohledně spotřeby serverů budou získávány průběžně podle toho, jak budou v datovém centru přibývat jak fyzické tak virtuální stroje.

Po dokončení systému pro sběr dat, kde kromě spotřeby elektrické energie budou rovněž shromažďovány další veličiny, jako výkon chladicího systému apod., bude možné přesněji určovat spotřebu.

Pro rozvržení virtuálních serverů na fyzické jsem zvolil algoritmus, který řeší problém batohu. Konkrétně jsem volil variantu 0 – 1 problému batohu. Problém batohu má výhodu v tom, že maximálně využívá kapacity batohu, v tomto případě kapacity fyzického serveru. Algoritmus, po úpravě cen na převrácené hodnoty, maximalizuje jejich sumu. Ve výsledku tedy algoritmus minimalizuje spotřebu na daném fyzickém serveru.

Tento přístup má i nevýhodu a to například v tom, že se musí spustit, pro každý fyzický server zvlášť. Dále také je u serverů třeba počítat s paměťovou kapacitou. Algoritmus bere v potaz při výpočtu pouze co nejlepší využití instalovaných procesorů.

Paměťovou kontrolu se mi povedlo vyřešit pomocí zjištění paměťových nároků virtuálních serverů, které označil algoritmus jako vhodné pro přiřazení na konkrétní fyzický server. Zjištěnou hodnotu vždy porovnávám s kapacitou fyzického serveru a příslušně reaguji na vzniklou situaci.

6 Závěr

Vytyčený cíl, vytvořit program, který bude minimalizovat spotřebu elektrické energie datového centra, byl splněn. Domnívám se, že program, který jsem vytvořil, je dobrým základem pro program, který pomůže při snižování spotřeby elektrické energie datového centra.

Tato verze programu optimalizuje spotřebu fyzických serverů, na základě jejich vytížení. Doufám, že tento vyvinutý software přispěji k úspoře elektrické energie a ke snižování nákladů na provoz datového centra.

Jako algoritmus pro minimalizaci jsem zvolil problém batohu. Tento problém je ze své podstaty maximalizující. Vhodnou manipulací s cenami (převrácení její hodnoty) jsem získal minimalizující algoritmus, který si zachovává tu vlastnost, že se snaží využít celý potenciál fyzického serveru s minimální cenou (v tomto případě spotřebou).

Management firmy se těžko přesvědčuje o úspornosti řešení, či potřebě nákupu nového zařízení, proto i výsledky z programu budou moci být použity při rozhodování o nákupu nového hardware a o účelném a úsporném využití již stávající infrastruktury. Úspornost a efektivní využití se velmi těžko dokazují, pokud nejsou k dispozici exaktní čísla a model, který dokáže říci, zda je využití optimálně minimalizované. Doufám, že díky tomuto programu bude dokazování úspornosti jednodušší.

6.1 Záměry do budoucna

V budoucnu bych se rád věnoval dalšímu vývoji a rozšíření programu. Mám v plánu zahrnout i jiné periferie než pouze fyzické servery. Rád bych do aplikace zaimplementoval spotřebu diskových polí, síťových prvků apod. Na spotřebě všech zařízení se určitě nemalou měrou podílí větrací zařízení, která jsou důležitá pro chlazení periferií. Proto bych také rád v budoucnu zahrnul jejich spotřebu do programu. Je možné, že pro tyto účely bude potřeba změnit optimalizační algoritmus za jiný, který dokáže počítat s více zájmovými faktory.

7 Literatura

1. **System.Data.SQLite.** System.Data.SQLite Home. *System.Data.SQLite.* [Online]
https://system.data.sqlite.org/index.html/doc/trunk/www/index.wiki.

2. **ITbiz.cz.** Jsou datacentra s TIER certifikací opravdu bezpečnější? . *ITbiz.cz.* [Online] 3. Duben 2014. [Citace: 25. Duben 2014.] http://www.itbiz.cz/clanky/jsou-datacentra-s-tier-certifikaci-opravdu-bezpecnejsi.

3. **Aroca, Jordi Arjona, a další, a další.** Power-efficient Assignment of Virtual Machines to Physical Machines. [Online] [Citace: 5. Květen 2014.] http://arxiv.org/pdf/1304.7121v2.pdf.

4. **Jansen, R. a Brenner, P.R.** Energy efficient virtual machine allocation in the cloud. *ieeexplore.ieee.org.* [Online] [Citace: 5. Květen 2014.] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6008550&isnumber=6008545.

5. **Demlová, Marie, prof.** Přednášky A4M01TAL. *Výuka - předmět Teorie algoritmů.* [Online] [Citace: 5. Květen 2014.] math.feld.cvut.cz/demlova/teaching/tal.

6. **SQLite.** SQLite Home Page. *SQLite.* [Online] [Citace: 5. Květen 2014.] http://www.sqlite.org/.

7. **Kopytov, Alexey.** SysBench. *SourceForge.net.* [Online] SourceForge.net. [Citace: 1. Květen 2014.] http://sysbench.sourceforge.net/.

8. **IBM.** Introduction to Storage Area Networks and System Networking - sg245470.pdf. *IBM Redbooks.* [Online] [Citace: 5. Květen 2014.] http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf.

9. **Dawande, M., a další, a další.** Springer. *Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions.* [Online] [Citace: 6. Květen 2014.] http://link.springer.com/article/10.1023/A:1009894503716.

10. **Nagel, Christian, a další, a další.** *Professional C# 2008.* New Jersey : Wrox, 2008. 978-0470191378.

11. **Chekuri, Chandra a Khanna, Sanjeev.** A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem. *Society for Industrial and Applied Mathematics.* [Online] 27. Červenec 2006. [Citace: 6.

Květen 2014.] http://dx.doi.org/10.1137/S0097539700382820. ISSN-1095-7111.

12. **Principled Technologies.** Dell PowerEdge M600 Blade Server. *Dell.* [Online] [Citace: 5. Květen 2014.] http://www.dell.com/downloads/global/products/pedge/en/pe_blades_spec jbb2005.pdf.

8 Přílohy

8.1 Výsledky benchmarkeru

```
[root@ref-srv ~]# sysbench --test=cpu --cpu-max-prime=200000 --
num-threads=1 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 1
Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 200000

Test execution summary:
    total time:                          508.3018s
    total number of events:              10000
    total time taken by event execution: 508.2958
    per-request statistics:
         min:                                  50.78ms
         avg:                                  50.83ms
         max:                                  57.79ms
         approx.  95 percentile:               50.89ms

Threads fairness:
    events (avg/stddev):           10000.0000/0.00
    execution time (avg/stddev):   508.2958/0.00
```

```
[root@ref-srv ~]# sysbench --test=cpu --cpu-max-prime=200000 --
num-threads=2 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 2
Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 200000

Test execution summary:
    total time:                          254.4207s
    total number of events:              10000
    total time taken by event execution: 508.8321
    per-request statistics:
         min:                                  50.77ms
         avg:                                  50.88ms
         max:                                  54.74ms
         approx.  95 percentile:               51.01ms

Threads fairness:
    events (avg/stddev):           5000.0000/3.00
    execution time (avg/stddev):   254.4160/0.00


    [root@ref-srv ~]# sysbench --test=cpu --cpu-max-prime=200000
--num-threads=3 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 3
Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 200000

Test execution summary:
    total time:                          169.8104s
    total number of events:              10000
    total time taken by event execution: 509.3596
    per-request statistics:
         min:                                  50.76ms
         avg:                                  50.94ms
         max:                                 132.21ms
         approx.  95 percentile:               50.98ms

Threads fairness:
    events (avg/stddev):           3333.3333/2.36
    execution time (avg/stddev):   169.7865/0.02
```

```
[root@ref-srv ~]# sysbench --test=cpu --cpu-max-prime=200000 --
num-threads=4 run
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 200000


Test execution summary:
    total time:                          128.1415s
    total number of events:              10000
    total time taken by event execution: 512.4596
    per-request statistics:
         min:                                  50.78ms
         avg:                                  51.25ms
         max:                                  97.68ms
         approx.  95 percentile:               51.86ms

Threads fairness:
    events (avg/stddev):           2500.0000/2.35
    execution time (avg/stddev):   128.1149/0.02
```