Západočeská univerzita v Plzni

FAKULTA PEDAGOGICKÁ

KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

Posouzení vlivů vybraných internetových aplikací na vytížení počítače

BAKALÁŘSKÁ PRÁCE

Lukáš Nimberger

Informatika se zaměřením na vzdělávání

2010-2013

Vedoucí práce: Mgr. Denis Mainz

Plzeň, 30. červen 2013

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

Plzeň, 30. červen 2013

……
vlastnoruční podpis

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu své kvalifikační práce Mgr. Denisi Mainzovi za cenné rady, připomínky a trpělivost při vedení této práce. Dále bych chtěl poděkovat všem vyučujícím Katedry výpočetní a didaktické techniky Pedagogické fakulty Západočeské univerzity v Plzni, za jejich pedagogickou činnost a poskytnuté znalosti v průběhu studia.

Lukáš Nimberger

OBSAH

SEZNAM ZKRATEK

AES ..... Advanced Encryption Standard.

AJAX ..... Asynchronous JavaScript and XML.

API ..... Application Programming Interface.

ASCII ..... American Standard Code for Information Interchange.

ASP ..... Active Server Pages.

AST ..... Abstract Syntactic Tree.

C# ..... C sharp.

CPU ..... Central Processing Unit.

CSS ..... Cascading Style Sheets.

DDR ..... Double Data Rate.

DNS ..... Domain Name Server.

DOM ..... Document Object Model.

FPS ..... Frames Per Second.

GB ..... Gigabyte.

GDDR ..... Graphics Double Data Rate.

GPU ..... Graphics Processing Unit.

HD ..... High-definition.

HTML ..... HyperText Markup Language.

HW ..... Hardware.

JPEG ..... Joint Photographic Experts Group.

JSON ..... JavaScript Object Notation.

kB ..... Kilobyte.

MB ..... Megabyte.

PC ..... Personal Computer.

PDF..... Portable Document Format.

PHP..... Hypertext Preprocessor.

PrtScr ..... Print Screen.

RAM ..... Random-Access Memory.

RIA..... Rich Internet Application.

RPM..... Revolutions Per Minute.

SD..... Standard-definition.

VB..... Visual Basic.

WMV..... Windows Media Video.

XAML..... Extensible Application Markup Language.

SEZNAM POJMŮ

ActiveX ..... Technologie společnosti Microsoft. Je používána pro komunikaci mezi jednotlivými aplikacemi společnosti Microsoft.

Benchmark ..... Ukazatel výkonnosti sloužící k porovnání určitých výsledků.

Cache paměť ..... Vyrovnávací paměť sloužící pro dočasné uložení často používaných dat.

Cookies ..... [z angličtiny „koláčky, sušenky“], soubory malé velikosti, jenž server odesílá prohlížeči uživatele, který jej ukládá. Nejčastěji se používá k identifikaci uživatele.

Drag and drop ..... [z angličtiny „táhni a pusť“], technika pro přemísťování objektů využívající stisku tlačítka myši, tažení a následného puštění tlačítka.

E-learning aplikace ..... Elektronické výukové aplikace.

Engine ..... Jádro programu, prohlížeče, hry.

Framework ..... Sada nástrojů, knihoven či programů sloužících k vývoji dalšího software.

Garbage collector ..... Specializovaná část běhového prostředí realizující automatickou správu paměti.

Haldy (heaps) ..... Speciální datová struktura pro uchovávání množiny čísel s definovaným uspořádáním.

Hašování ..... Algoritmus pro převod vstupních dat na číslo sloužící zejména pro prohledávání databází.

Chipset ..... [z angličtiny „čipová sada“], sada elektronických integrovaných obvodů, které řídí komunikaci mezi jednotlivými prvky základní desky.

Hardware ..... Soubor fyzických komponentů počítače.

Interpret ..... Program specializovaný na vykonávání zdrojového kódu nebo jeho překládání.

JavaScript jádro ..... Jedna z komponent internetového prohlížeče, která pracuje s JavaScript kódem.

JavaScript Object Notation ... Jazykově nezávislý formát pro výměnu dat na webu, který umožňuje zpracovávat různé datové struktury a převádět je na řetězec.

jQuery ..... JavaScriptová knihovna.

Kodek ..... Složenina slov Kodér a DEKodér. Počítačový program specializovaný na transformaci dat a signálů.

Log ..... Výstupní soubor záznamů, zpráva o činnosti.

Memory leak ..... [z angličtiny „únik paměti“], chování programu, kdy dochází k využití operační paměti, která ale není uvolněna, ani když už není využívána.

Parsování ..... Slangový výraz pro syntaktickou analýzu. Je to proces, při kterém se analyzuje posloupnost prvků (obvykle ze vstupního řetězce), která se následně transformuje do stromové struktury dle předem známých pravidel formální gramatiky.

Pixel ..... Obrazový bod.

Platforma ..... Skupina software, hardware či obojího, která umožňuje běh počítače jako celku.

Plug-in ..... Zásuvný modul prohlížeče. Rozšiřuje funkční schopnosti prohlížeče.

Ray-tracing ..... [z angličtiny „sledování paprsku“], metoda výpočtu a zobrazení světelných efektů při vytvářené 3D grafiky.

Renderovací jádro ..... Jádro zodpovědné za zobrazování požadovaného obsahu na obrazovku.

Rich Internet Application ..... [z angličtiny „bohatá internetová aplikace“], druh internetové aplikace, která se v některých parametrech shoduje nebo alespoň velmi podobá lokálně instalovaným aplikacím.

Score ..... Výsledek, v našem případě udávající počet bodů.

Scrollování..... Pohyb, posouvání obsahu okna pomocí kolečka na myši nebo posuvníku.

Server ..... Počítač se síťovým operačním systémem, který poskytuje služby počítačům v síti.

Shader..... Program sloužící k řízení činnosti GPU.

Smartphone ..... [z angličtiny „chytrý telefon“], označení mobilního telefonu s operačním systémem, širokou škálou podporovaných aplikací a interaktivním uživatelským rozhraním.

Software..... Programová výbava počítače.

Streaming ..... Kontinuální doručování multimediálních dat mezi poskytovatelem služby (streamovacím serverem) a jejím příjemcem.

Streamovací server ..... Servery poskytující své služby pomocí streamingu.

Tablet ..... Přenosný plochý počítač s dotykovou obrazovkou.

Tag cloud ..... [z angličtiny „oblak štítků“], navigace na webové stránce reprezentovaná zejména klíčovými slovy, jejichž velikost je úměrná jejich důležitosti.

Utilita ..... Pomocný program.

Virtuální stroj ..... Software vytvářející virtualizované prostředí, díky čemuž je umožněn například běh operačního systému na platformě, pro kterou nebyl původně určen.

1 ÚVOD

Téma kvalifikační práce bylo vybráno s ohledem na možnost skloubit teoretické znalosti s praktickým měřením. Při výběru tématu mne zaujala možnost vyzkoušet a otestovat tak často používaný software, jako jsou internetové prohlížeče a dále vliv jejich používání na výkon počítače.

Cílem práce je zjistit pomocí série testů a monitoringu, do jaké míry zatěžují webové aplikace vybrané počítačové komponenty. Stanovíme si nejdůležitější sledované parametry a ty otestujeme.

Testování softwaru i hardwaru v oblasti informatiky obecně má relativně krátkou dobu aktuálnosti. Aktuálnost celé práce závisí na rychlosti vývojářů a uvolňování nových verzí webových aplikací a zejména internetových prohlížečů. Dalším omezením bude vlastní způsob měření. Ideální by bylo obsáhlé měření všech internetových prohlížečů včetně různých verzí na větším množství strojů a také na rozdílných platformách operačních systémů. Avšak i v odborných médiích se většinou setkáváme s testy, ve kterých se internetové prohlížeče testují na jedné referenční sestavě a na vybraném operačním systému.

Výstupem této práce bude souhrn výsledků testování internetových prohlížečů a webových aplikací spuštěných v jednom z otestovaných prohlížečů, jež bude vykazovat v průměru nejlepší výsledky. Dále výstupem práce budou obecná doporučení pro výběr programů a zobecnění, co vše má vliv na vytížení počítačových komponent.

2 WEBOVÉ APLIKACE

„Webová aplikace je dynamický web. Dynamický web je takový, do něhož se informace dynamicky doplňují a to nejčastěji z databází. V takovém webu stačí pouze změnit informace, na základě kterých pracuje a které předává návštěvníkům, a příslušné stránky se mění samy. Například můžete do formuláře zadat novinku, která by měla být přehledně zobrazena na Vašem webu na příslušných místech. V dynamickém webu stačí zadat údaj jen jednou, do jednoduchého formuláře, a změny se objeví všude tam, kde mají. Vznikají často ve skriptovacích jazycích PHP nebo ASP. Tyto jazyky poskytují velké množství užitečných funkcí, které se starají o prezentaci dat. Oddělení logiky od prezentace je hlavním trendem moderních webových aplikací, protože usnadňuje údržbu a zpřehledňuje aplikaci.“ [1]

„Webové aplikace jsou programy vytvořené pro použití pouze v rámci prohlížeče. Pomocí aplikací můžete například vytvářet dokumenty, upravovat fotografie a poslouchat hudbu bez toho, abyste museli instalovat složitý software.“ [2]

Pod pojmem „webová aplikace“ si tedy můžeme v širším kontextu představit prakticky veškeré dynamické webové stránky. Častěji se pod tímto pojmem rozumí program spuštěný pomocí internetového prohlížeče, využívající internetového připojení a spolupracujícího s ostatními zařízeními na této síti, v případě web aplikací především se servery.

2.1 VÝZNAM, BUDOUCNOST A PŘÍKLADY WEBOVÝCH APLIKACÍ

Oblíbenost webových aplikací mezi uživateli jednoznačně roste a jejich vývojáři jsou si tohoto faktu dobře vědomi. Proto se stále internetové aplikace inovují a transformují. Také škála zařízení, které využívají internetové aplikace je větší, než dříve. Kromě klasických desktop počítačů a notebooků jsou to mobilní telefony s operačním systémem, tablety, ale například i tzv. chytré televize. Můžeme předpokládat, že se v budoucnu výčet zařízení pracujících s internetovými aplikacemi ještě rozšíří.

Rozvoj webových aplikací hojně podporuje rozšíření a rozvoj nových přenosných zařízení, zejména smartphone telefonů a tabletů. Nejenom, že uživatelé tráví mnohem více času na webu díky stále lepší dostupnosti internetového připojení, ale hojně využívají

služby, které vznikly jako protějšky klasických lokálních (nativních) aplikací, které musí uživatel sám instalovat a updatovat. Jedná se především o online kancelářské balíky či editační programy pro audio a video soubory, ale mění se také způsob myšlení, jak se soubory nakládat nebo je zálohovat. Rozhodně nezanedbatelným trendem se stalo sdílení dat, které zejména díky sociálním sítím a online galeriím umožňuje zhlédnutí dat velkému množství uživatelů, přičemž sdílení je možné omezit například jen pro určitou skupinu přátel. Klienti pro prohlížení sociálních sítí se pak stali jednou ze základních předinstalovaných aplikací většiny mobilních telefonů.

Ochrana dat a jejich dosažitelnost se stala přirozeným vyústěním stále více užívaných online služeb. Proto vznikly některé servery umožňující zálohování, ale i zmíněné sdílení. Mezi tyto servery patří například DropBox, Microsoft SkyDrive, Amazon Cloud Drive, Apple iCloud nebo jeden z nejnovějších a nejrozsáhlejších – Google Drive. Google Drive navíc nabízí ještě integraci jiných služeb společnosti Google, především Gmail a Google Docs.

Příklady webových aplikací a služeb jsou:

- Elektronické obchody.
- Diskusní fóra.
- Online aukce.
- Blogy.
- E-learning aplikace.
- Online aplikace pro zpracování audio/video dat, případně pro konverzi dat.
- Internetové přehrávače atd.

2.2 VÝHODY A NEVÝHODY WEBOVÝCH APLIKACÍ

Internetové aplikace přinášejí řadu výhod. Mezi nejdůležitější z nich patří:

- Odpadá nutnost instalovat na klientský počítač rozsáhlé softwarové balíky, pro práci s internetovými aplikacemi stačí pouze internetový prohlížeč, případně doplněný o zásuvné moduly a rozšíření.

- Jednoduchá správa a aktualizace webových aplikací bez nutnosti stahovat a instalovat další softwarové balíčky na klientský počítač. Aktualizace a správa aplikací se provádí v centrále provozovatele.
- Díky umístění webové aplikace mimo klientský počítač je zajištěna její určitá ochrana před poškozením uživatelem.
- Tytéž webové aplikace mohou používat tisíce uživatelů najednou.
- Jsou nezávislé na operačním systému. Dokonce mohou fungovat zcela bezchybně i na systému poškozeném.
- Z hlediska poskytovatele webové aplikace jsou jejími výhodami možnosti přidělování práv uživatelům, vyhodnocování statistik jejich užívání a v neposlední řadě také různé způsoby jejich zpoplatnění.
- Možnost šifrované komunikace mezi uživateli, a to i v reálném čase.
- Možnost zálohování dat mimo hardware klientského počítače, ochrana dat a možnost jejich snadného sdílení.

Přestože se internetové aplikace neustále vyvíjí a vylepšují, ze své podstaty mají i některé nevýhody:

- Nutnost připojení k internetu.
- Webová aplikace může reagovat pomaleji, než aplikace lokální.
- V případě výpadku centrály se zastaví provoz na všech klientských počítačích.
- Otázka zabezpečení se netýká jen klienta. [3]

3 HARDWARE

Každý osobní počítač (PC – Personal Computer) je tvořen skupinou komponent, jejichž parametry mají klíčový vliv na stabilitu a výkon celého počítačového systému. Určujícími vlastnostmi počítačů však není jen hardware, tedy technické vybavení počítače, ale také software, neboli vybavení programové. Bezproblémový provoz počítače závisí na správné spolupráci softwaru, hardwaru a používaném operačním systému, který tvoří jejich rozhraní.

Přestože je výčet počítačových komponent a používaných periferních zařízení poměrně rozsáhlý, budeme vycházet z faktu, že základními komponentami osobního počítače jsou:

- Základní deska.
- Procesor.
- Paměť RAM.
- Grafická karta (nebo jen integrovaný grafický čip).
- Pevný disk.
- Napájecí zdroj.

V architektuře počítačového systému se vyskytuje počet dalších komponentů, jsou to zejména zvukové, síťové, televizní a střihové karty, přídavné chladiče a mechaniky. Dále některé komponenty (typicky grafická, zvuková a síťová karta) bývají integrované na základní desce. [4] Na tomto místě je třeba zdůraznit, že hardware má klíčový vliv na absolutní výsledky jednotlivých testů. Pokud by například uživatel chtěl podobnými testy podrobit svůj notebook, který obsahuje jak integrovaný grafický čip na základní desce, a současně grafickou kartu samostatnou - dedikovanou (dnes se tato kombinace často používá z důvodu prodloužení výdrže baterie), budou výsledky testů výrazně ovlivněny tím, který ze zobrazovacích adaptérů bude používán.

3.1 PROBLEMATIKA ZÁTĚŽE ANEB JE TŘEBA SE ZATÍŽENÍM ZABÝVAT?

Pokud je počítač zatížen, zpracování procesů může být pomalejší nebo/a méně plynulé, může také docházet k nespojitostem vedoucím k pádu programu nebo celého

systému. V kapitole 2.2 jsem zmínil, že internet je čím dál více využíván mobilními zařízeními. Většina poskytovatelů mobilního internetu ještě stále praktikuje datové limity, a proto by uživatele mělo zajímat, kolik dat spotřebovávají internetové aplikace a práce s internetem vůbec.

Dále můžeme pohlížet na problematiku zátěže jako na úměrnost spotřeby energie k zatížení počítače. Tento pohled je pak důležitý zejména u mobilních zařízení, kde vytížení zařízení přímo ovlivňuje výdrž baterie. Tuto výdrž pak v takovém případě samozřejmě neovlivňují jen webové aplikace, ale veškerý běh systému a ostatních programů. Můžeme se však zamyslet nad tím, zda bychom nemohli odlehčit zátěži hardware například používáním textových prohlížečů bez nadbytečných zásuvných modulů a rozšíření nebo používat takové prohlížeče, které dokážou rozložit zátěž mezi více komponent, zejména CPU a GPU. Dále je zřejmé, že již samotná volba webového prohlížeče bude mít na zatížení počítače vliv. O jednotlivých prohlížečích bude pojednáno v některé z následujících částí této práce, nyní pouze zmíníme, že různé prohlížeče (včetně různých verzí téže platformy) používají různá renderovací a JavaScript jádra, podporují či nepodporují hardwarovou akceleraci, webové standardy (např. HTML5) či audio a video kodeky.

My se však budeme zabývat především vytížením hardwaru počítače. Sledování této oblasti se může zdát v dnešní době, kdy se i v PC sestavách střední třídy objevují vícejádrové procesory a RAM paměti o kapacitě několik GB, nedůležité. Berme však v potaz, že stejně, jako se vyvíjí hardware, vyvíjí se i software, web aplikace a prohlížeče nevyjímaje. S tím souvisí i jejich vzrůstající minimální systémové nároky.

3.2 HARDWARE ZATĚŽOVANÝ WEBOVÝMI APLIKACEMI

Při testování zátěže webových aplikací na výkon počítače můžeme sledovat především vytížení těchto počítačových komponent.

- Procesor.
- Grafický procesor.
- Paměť RAM.

3.2.1 PROCESOR A ZPŮSOB SPOUŠTĚNÍ PROCESŮ PROHLÍŽEČI

Procesor (CPU – Central Processing Unit) je integrovaný obvod, který plní řídící funkci počítače. Provádí výpočty, zpracovává data, ovládá další komponenty počítače a komunikuje s nimi. [5] Když chceme říci procesoru, co má dělat, musíme mluvit v takzvaném instrukčním kódu. Počítač si instrukce tohoto kódu načítá z operační paměti a potom je velmi rychle zpracovává. Dnešní procesory zvládnou splnit za jednu vteřinu až desítky miliard instrukcí. [6]

Procesor zpracovává vlákna jednotlivých procesů, které jsou spuštěny při startu a běhu prohlížeče. Nejedná se obvykle jen o běh vlastního procesu prohlížeče, například *firefox.exe*. Firefox spouští ještě proces *plugin-container.exe*, který umožňuje běh prohlížeče i v případě, když některý zásuvný modul spadne. Tento proces obaluje dále dvě instance *FlashPlayerPlugin.exe* (v případě, že prohlížíme stránku s obsahem spustitelným pomocí Adobe Flash). Dvě instance proto, že každá běží s různým stupněm integrity – *High* a *Low* (vysoká a nízká). Adobe Flash Player totiž s prohlížečem Firefox na platformě Windows běží v tzv. chráněném režimu. Uživatel spustí proces s integritou vysokou, ale samotný proces běží s nízkou integritou, což zabraňuje procesu zápis do většiny oblastí uživatelského profilu a registrů, které potřebují k přístupu vysokou úroveň integrity. [7] Na principu spuštěného .exe souboru prohlížeče a procesů zajišťující běh zásuvných modulů funguje i Opera a Safari.

Jiným způsobem se chová Google Chrome. Každý panel i každý použitý zásuvný modul otevírá jako samostatný proces. Toto chování lze potlačit spuštěním prohlížeče z příkazové řádky s parametrem *-single-process*. Díky skutečnosti, že je běh panelů a zásuvných modulů takto rozdělen se nabízí možnost monitorovat, jakým způsobem vytěžují počítač jednotlivé webové aplikace. Google Chrome navíc nabízí vlastního Správce úloh. Více instancí spouští také Internet Explorer 10. Tyto prohlížeče označujeme souhrnně jako **multi-procesové**. Protože tento princip implementoval Google do Chrome a jasně popsal, proč a jak jsou procesy spouštěny, budeme v dalším textu brát jako multi-procesový právě Chrome.

Celkové zatížení procesoru ale stále závisí především na náročnosti webové aplikace a případné distribuci zatížení na jiné počítačové komponenty, především na

grafickou kartu. Rozdělení spuštěných panelů a běžících zásuvných modulů do samostatných procesů nemá primárně výkonnostní opodstatnění, ale přispívá ke stabilitě programu. Každé jádro procesoru může v jeden okamžik zpracovávat pouze jeden proces resp. jedno vlákno (*thread*). Námi použitý procesor AMD Athlon 64 X2 má dvě fyzická jádra, takže v jednom okamžiku může obsluhovat dva procesy zároveň nebo dvě vlákna jednoho procesu. O tom, jaká vlákna budou v dané chvíli zpracovávána, rozhoduje plánovač (*scheduler*) především na základě priority procesu. Ta lze sice měnit z uživatelské strany, ale obecně se změna priority nedoporučuje. Největší vliv na distribuci aplikačních vláken mezi více jader procesoru má sám programátor. Existují však algoritmy, u nichž není možné použít jiné, než jednovláknové zpracování. Typickým příkladem jednovláknového zpracování instrukcí je JavaScript. Jednovláknové je také téměř celé renderovací jádro prohlížeče, s výjimkou jeho síťového subsystému. Síťové operace mohou využívat několik paralelních vláken, zpravidla 2-6. Právě renderovací jádro je hlavním vláknem celého prohlížeče. Hlavní vlákno je pak nekonečná smyčka, která udržuje proces naživu a čeká na události, které následně zpracovává. [8]

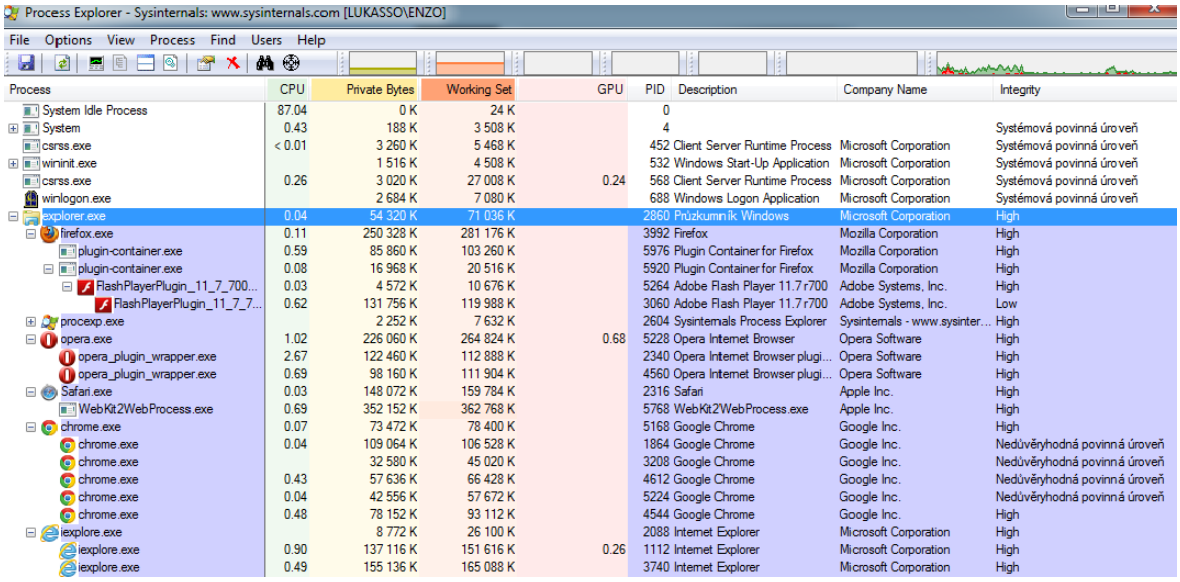
Pro lepší ilustraci chování jednotlivých prohlížečů jsem použil program Process Explorer, který umí zobrazit stromovou strukturu spuštěných programů a jimi spuštěných procesů a mnoho dalších informací o nich. V každém prohlížeči jsem spustil dva panely, v jednom se přehrávalo video ze stránky youtube.com, využívající přehrávač Adobe Flash (zásuvný modul Shockwave Player) a ve druhém video využívající zásuvný modul Microsoft Silverlight. Z obrázku níže je jasně vidět způsob vytváření procesů jednotlivými prohlížeči.

Firefox pro otevření dvou různých zásuvných modulů vytvořil dva procesy *plugin-container.exe*. Jeden zajišťuje běh Silverlightu, druhý Flashe. Proč existují dvě instance *FlashPlayerPlugin.exe* je popsáno výše. Naprosto stejně se chová i Opera a chovalo by se tak i Safari, kdyby podporovalo Silverlight. Nicméně počet spuštěných procesů doposud závisí na počtu použitých zásuvných modulů. Otevření dalších panelů prohlížeče by samo o sobě nemělo na vznik nových procesů žádný vliv.

Internet Explorer naopak nespouští nové procesy použitím zásuvných modulů, ale otevřením nového panelu (o zásuvných modulech bude pojednáno později, nyní pouze

zmíním, že Internet Explorer nepoužívá Adobe Flash jako zásuvný modul, ale jako ovladač ActiveX).

Google Chrome spravuje procesy ještě více separátně. Samostatně běží nejen jednotlivé zásuvné moduly a rozšíření, ale také panely a navíc i proces nazvaný *Proces GPU*, který běží, když je zobrazován hardwarově akcelerovaný obsah jako HTML, CSS či jiné grafické prvky webové stránky. Již při otevření prázdného prohlížeče vzniknou dva procesy – proces *Prohlížeč*, který zajišťuje běh hlavních funkcionalit prohlížeče a druhý je počáteční renderovací proces, který zajišťuje běh renderovacího jádra WebKit a JavaScript jádra V8. Pokaždé, když otevřeme nový panel prohlížeče, spustíme také nový renderovací proces.



Obrázek 1: Spuštěné procesy jednotlivými prohlížeči.

VYUŽITÍ VÍCE JADER PROCESORU

Jednou z otázek, které se často řeší na internetu, je využití více jader procesoru. *Má-li v paralelním systému probíhat několik procesů současně, je potřeba, aby algoritmus činnosti byl k tomu uzpůsoben, tj. aby byla provedena tzv. paralelizace algoritmu. Jedná se o to, aby jednotlivým subprocesorům byla přidělena činnost tak, aby výsledný proces proběhl v paralelním systému co nejefektivněji. Efektivností je zde myšlena především rychlost, ale může to být v některých případech bezpečnost, spolehlivost, příp. přesnost výsledku.* [9]

Aby bylo možné dosáhnout výše popsaného zrychlení, je třeba vytvořit program s více vlákny. Vlákno je posloupnost instrukcí, které můžou být zpracovány současně s dalšími vlákny. Proces obsahuje vždy alespoň jedno, častěji více vláken. Vlákna uvnitř procesu mohou vzájemně sdílet stejná data, procesy nikoliv. [10] *„K tomu, aby vícevláknové aplikace dělaly něco užitečného, je obvykle nezbytné, aby vlákna sdílela nějaký druh společného stavu. Potřebná úroveň sdílení závisí na konkrétní úloze. Jedním extrémem může být jediné číslo, jež označuje úlohu, která se má provést. Například vláknu na webovém serveru může být oznámeno pouze číslo portu, na který má odpovídat. Druhým extrémem může být fond vláken neustále předávající informace mezi vlákny pro signalizaci, které úlohy jsou dokončené a která práce ještě stále čeká na dokončení.“* [11] Fakt, že jsou sdílena určitá data mezi vlákny, vede k problémům se sdílením dat mezi vlákny, zejména k tzv. data races (soupeření o data), což jsou situace, kdy více vláken aktualizuje stejná data nebezpečným způsobem. [11]

Dnešní internetové prohlížeče využívají vícejádrových procesorů jen tehdy, pokud spouští více instancí (procesů), kdy každý proces může využít jedno jádro. Z mých jednoduchých měření však vyplývá, že jeden proces prozatím neumí využít více jader procesoru. K tomuto závěru jsem dospěl díky jednoduchému pokusu. Využil jsem online aplikace, tzv. stres testu, který generoval několik úrovní zatížení CPU (0, 25, 50 a 100 %). 100% zatížení bylo vyvoláno během nekonečné smyčky. V prohlížeči Opera, jehož běh zajišťuje prakticky jeden proces, jsem spustil test se zvolenou zátěží 100 %, což vedlo dle programu Process Explorer i Správce úloh Windows pouze k 50 % vytížení CPU tímto procesem. Hranice 50% vytížení CPU nebyla překonána i při běhu 100% zátěže ve více panelech, z čehož usuzuji, že byl proces zpracováván jediným jádrem. Stejná situace nastala v případě stres testu prohlížeče Internet Explorer. Zde běžel zátěžový test také ve 2 panelech, každý vytěžoval procesor na necelých 50 %. Po zavření jednoho panelu však vytížení druhým panelem (procesem) taktéž nepřekonalo hranici 50 %. Google Chrome taktéž nepodporuje vícevláknové zpracování, navíc využívá technologie aktivních oken, kdy běží renderovací proces pouze v aktivním okně, díky čemuž zatěžuje CPU pouze jednou instancí.

Bohužel jsem nenalezl dostatek relevantní literatury, proto jsem se touto problematikou nadále nezabýval. Pro úplnost jen doplním, že je v současné době

společností Mozilla Foundation vyvíjeno nové renderovací jádro s názvem Servo, které dle zdroje [12] využívá programovacího jazyka Rust a jenž je výrazně paralelizováno.

3.2.2 OPERAČNÍ PAMĚŤ

Operační paměť je volatilní (nestálá) vnitřní elektronická paměť číslicového počítače typu RWM-RAM, určená pro dočasné uložení zpracovávaných dat a spouštěného programového kódu. Tato paměť má obvykle rychlejší přístup, než vnější paměť (např. pevný disk). Tuto paměť může procesor adresovat přímo, pomocí podpory ve své instrukční síti. Strojové instrukce jsou adresovány pomocí instrukčního ukazatele a k datům se obvykle přistupuje pomocí adresace prvku paměti hodnotou uloženou v registru procesoru nebo je adresa dat součástí strojové instrukce. Operační paměť je spojena s procesorem pomocí sběrnice, obvykle se mezi procesor a operační paměť vkládá rychlá vyrovnávací paměť typu cache. [13]

S rozdílným přístupem k procesům se dostáváme k dalšímu sledovanému parametru – využití operační paměti. Zde se zároveň dostáváme k tomu, proč se zaměřujeme před testováním na webové prohlížeče jako takové. Protože jsou některé prohlížeče multi-procesové a jiné ne, dochází k odlišnostem ve způsobu využívání i uvolňování paměti.

Nevýhodou „multi-procesových“ prohlížečů je fakt, že běh každého procesu zvyšuje režii. Proces si vyhradí část paměti a spotřebovává procesorový čas. Tato nevýhoda je znatelná hlavně proto, že se replikují některé vnitřní komponenty prohlížeče, jako například cache, haldy JavaScript virtuálního stroje a vnitřní datové struktury, které musí být duplikovány uvnitř více procesů. Problematický je pak zejména JavaScript, jehož haldy (*heaps*) jsou relativně velké a musí být replikovány mezi všemi procesy prohlížeče. Nejznatelněji se tento fakt projeví, pokud je spuštěno mnoho panelů, které zpracovávají velké množství JavaScript kódu z různých stránek. [14]

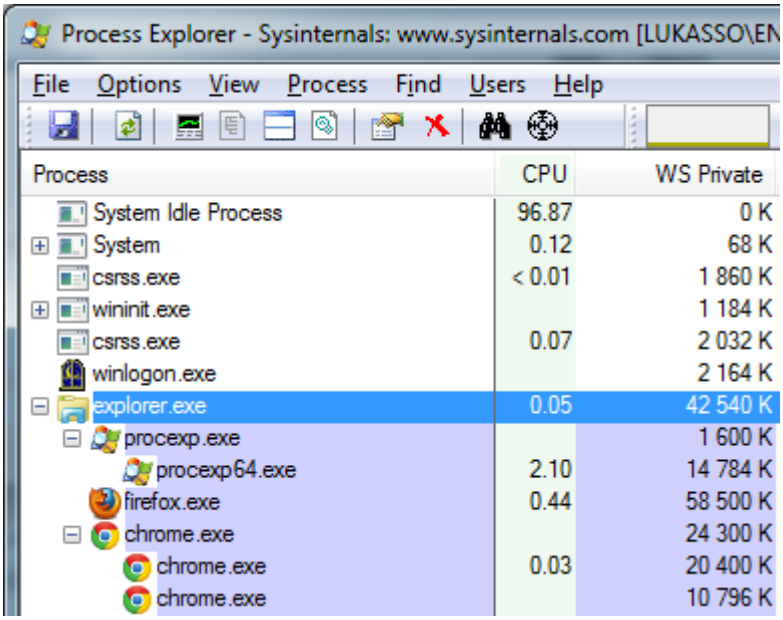
Naopak výhodou multi-procesových prohlížečů je, že mnohem ochotněji uvolňují paměť nevyužívaných procesů. Pokud je v prohlížeči otevřeno několik záložek, z nichž je například pouze jedna aktivně využívána, vyhrazená paměť neaktivních záložek je z podstatné části navrácena systému. „Jedno-procesové“ prohlížeče mezi aktivními a nevyužívanými záložkami nerozlišují, nemohou proto automaticky redukovat

vyhrazenou paměť. Další výhodou multi-procesových prohlížečů je uvolňování paměti při zavírání jednotlivých panelů. Při zavření některého z panelů se automaticky ukončí proces (nebo procesy), které zajišťovali běh stránky a jejich paměť je kompletně navrácena systému. Jedno-procesové prohlížeče si při otevírání různých panelů zaberou část paměti, která je systému plně vrácena až při zavření hlavního procesu, tedy celého prohlížeče. Při postupném zavírání panelů se využívaná paměť sice snižuje, ale méně, než u multi-procesových prohlížečů. [14]

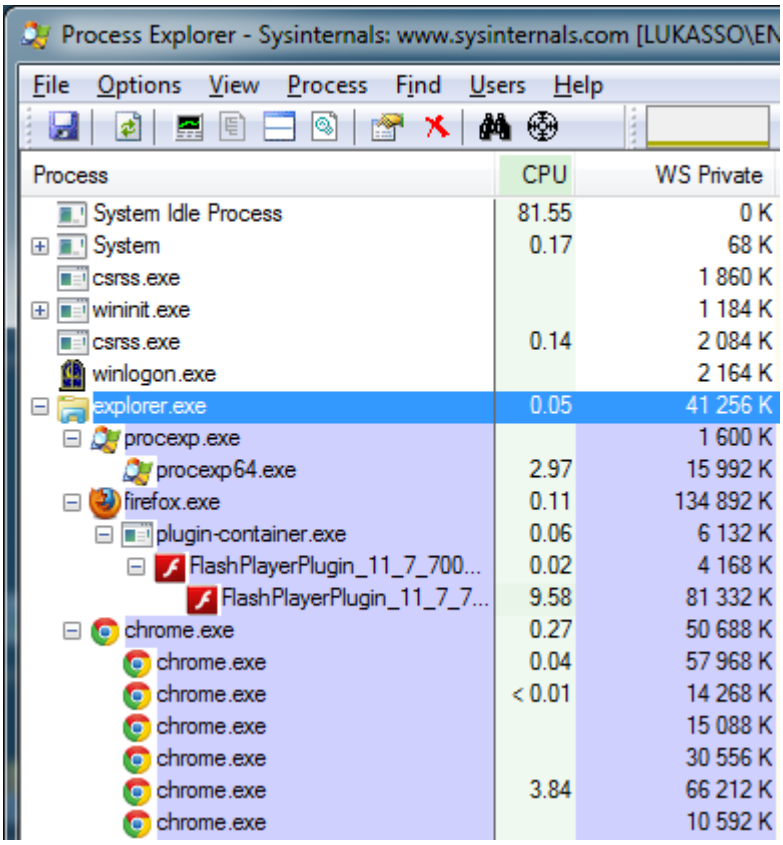
Pro názornost vyzkoušíme výše popsané chování na malém příkladu. Otevřeme prohlížeč Google Chrome a Mozilla Firefox, oba prohlížeče defaultně otevírají jednu záložku s prázdnou stránkou (*about:blank*). Zaznamenáme hodnotu velikosti paměti (obr. 2). Otevřeme v každém prohlížeči další 3 panely, *seznam.cz*, *aukro.cz* a *youtube.com*. Po minutě opět zaznamenáme hodnotu využívané paměti (obr. 3). Nakonec všechny 3 stránky postupně zavřeme, necháme spuštěné prohlížeče pouze s panelem v režimu prázdné stránky a hodnoty zaznamenáme po třetí (obr. 4).

Ve většině monitorovacích programů (Sp*rávce úloh Windows, Sledování výkonu Windows, Process Explorer*) rozlišujeme několik druhů paměti.

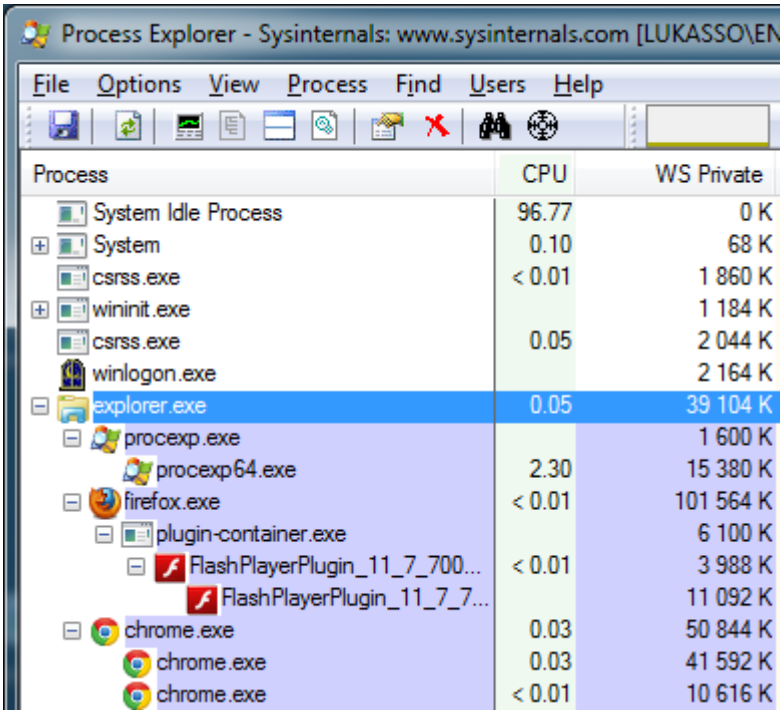
- **Pracovní sada:** Velikost soukromé pracovní sady plus velikost procesem používané paměti, která může být sdílena s jinými procesy. Pracovní sada je tedy složená ze Soukromé pracovní sady a Sady ke sdílení.
- **Nejvyšší velikost pracovní sady:** Maximální velikost pracovní sady používané procesem.
- **Soukromá pracovní sada:** Podsada pracovní sady, jež konkrétně popisuje velikost paměti používané procesem, kterou nelze sdílet s jinými procesy.
- **Velikost po spuštění:** Velikost virtuální paměti vyhrazené pro proces.
- **Stránkovaný fond:** Velikost virtuální paměti přidělené procesu, kterou lze zapsat na jiné médium, například pevný disk.
- **Nestránkovaný fond:** Velikost virtuální paměti přidělené procesu, kterou nelze zapsat na jiné médium. [15]



Obrázek 2: Demonstrace hospodaření s pamětí – spuštění prohlížeče.



Obrázek 3: Demonstrace hospodaření s pamětí - 3 nově otevřené panely.



Obrázek 4: Demonstrace hospodaření s pamětí - po zavření panelů.

V našem pokusu se zaměříme na sledování soukromé pracovní sady (v programu Process Explorer označované jako WS Private). Právě tento druh paměti je zobrazen i ve výchozím nastavení Správce úloh operačního systému Windows 7 a určuje, kolik paměti je používáno jedním konkrétním procesem.

Paměť – soukromá pracovní sada [kB]				
Prohlížeč	**Po spuštění prohlížeče**	**Otevření 3 panelů**	**Po zavření panelů**	**Rozdíl (spuštění – zavření)**
Google Chrome	54 172	245 372	103 052	48 880
Mozilla Firefox	58 500	226 524	122 744	64 244

Tabulka 1: Práce s pamětí vybraných prohlížečů.

Z tabulky vidíme, že multi-procesový prohlížeč skutečně při otevření několika záložek spotřebovává více paměti, než jedno-procesový a v souladu s teorií výše, po zavření panelů i více paměti uvolní. Ještě razantněji by se toto chování projevilo, pokud bychom využívali mnohem více (řádově desítky) panelů. Na obrázku 2 vidíme tři okna, ve kterých jsou zachyceny hodnoty využití paměti ve výše uvedených třech krocích tak, jak byly postupně vykonávány.

Dostáváme se k otázce, co ovlivňuje spotřebu paměti RAM, kromě způsobu spouštění a ukončování procesů. Obecně lze říci, že záleží na tom, jakým způsobem jsou webové aplikace a prohlížeče naprogramovány a implementovány. To souvisí především s implementací AJAX kódu, funkcí garbage collectorů, způsobem, jakým prohlížeč alokuje a dealokuje paměť, a v neposlední řadě také s memory leaks aplikací, včetně rozšíření a zásuvných modulů. Velikost využité paměti samozřejmě ovlivňuje také vlastní obsah web aplikace, například množství obrázků nebo jiného multimediálního obsahu, použití zásuvných modulů apod. Testováním vytížení paměti jednotlivými prohlížeči se budeme zabývat v samostatné kapitole. Jako poslední faktor ovlivňující vytížení operační paměti zmíním přítomnost samostatné (dedikované) grafické karty s vlastní pamětí (typu DDR či GDDR) v níž je uložen obraz, textury, mezivýsledky při vykreslování obrazu a další. V případě integrovaného grafického čipu na základní desce je však využívána pro ukládání těchto informací operační paměť.

3.2.3 GRAFICKÝ PROCESOR A JEHO VYUŽITÍ POMOCÍ HARDWAROVÉ AKCELERACE

Využití grafického procesoru (GPU – Graphics Processor Unit) je při běhu prohlížeče a internetových aplikací realizováno ze dvou hlavních důvodů:

1. zrychlení zobrazování grafického obrazu,
2. snížení zátěže CPU.

Protože je CPU primárně zaměřen na zpracování složitých operací a skoky v instrukcích, zabírá velkou část čipu kontrolní logika a vyrovnávací paměť (cache), která slouží pro ukládání dat a programů určených k bezprostřednímu zpracování procesorem. Tím je omezen prostor na čipu pro implementaci velkého množství aritmeticko-logických jednotek, což výrobci ve snaze podpořit paralelismus řeší implementací více jader na společný čip procesoru.

Grafická karta paralelismus podporuje ze své hardwarové podstaty, obsahuje stovky tzv. streamovacích (SM) multiprocesorů – v případě grafické karty naší sestavy je počet SM multiprocesorů 320. *„Tyto multiprocesory dokážou zpracovávat v jednom okamžiku něco přes tisíc vláken a dohromady tak může běžet na jednom grafickém čipu desetitisíce vláken. Většinou jsou bloky vláken omezeny na vykonávání jedné instrukce nad různými daty (stejná instrukce, ale jiná data pro každé vlákno - tzv. SIMD).* [16] Paralelní

systém *„SIMD (Single Instruction stream, Multiple Data stream) představuje zpracování jednoho programu větším množstvím stejných jednotek (subprocesorů).“* [9] GPU tedy dokáže zpracovávat velké množství paralelních instrukcí najednou, nejčastěji po blocích se stejnou instrukcí, ale různými daty. Tohoto faktu se využívá zejména v situaci, kdy má velký počet pixelů podobný nebo stejný postup výpočtu. Příkladem může být vykreslování pozadí HTML dokumentu malým obrázkem. Využití grafické karty dává ještě větší smysl při vykreslování 3D grafiky, kde se tělesa pokrývají texturou, dochází ke stínování a práci se světly.

O využití hardwarové akcelerace, jejím testování a dopadu na běh aplikace bude pojednáno později. Na tomto místě prozatím zmíníme, že HW akcelerace se objevila teprve v roce 2010, ale ani dnes není podporována všemi nejznámějšími prohlížeči.

4 PŘÍPRAVA NA TESTOVÁNÍ, VYBRANÝ HARDWARE A SOFTWARE

Vlastnímu testování předcházel výběr vhodného software a hardware. Referenční počítačová sestava byla sestavena převážně ze starších komponentů s ohledem na minimální nároky testovaných i testovacích programů. Výpis použitých komponent je obsažen v tabulce 2. Doplňující informací je, že RAM paměťové moduly byly zapojeny v tzv. *Dual-channel*. V tomto zapojení jsou vlastně využívány dvě sběrnice spojující operační paměť s chipsetem. V tomto zapojení je propustnost pamětí teoreticky dvojnásobná. [17] K využití této funkce bylo nutné použít dva páry identických pamětí zakoupené jako tzv. *Dual Channel Kit*. [18] Veškerá programová výbava a operační systém se nacházely na jednom oddílu jednoho fyzického disku.

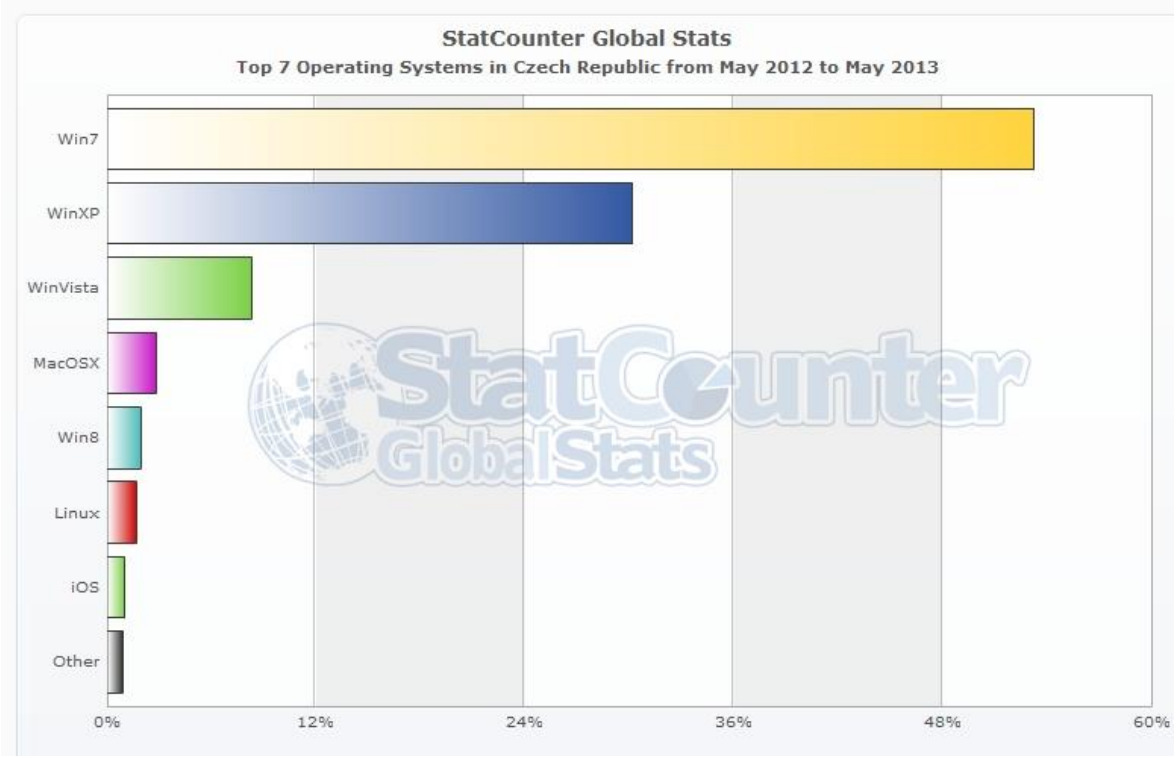
Nedílnou součástí celého testování bylo vyhledávání vhodných programů pro monitorování a záznam naměřených hodnot. Tento úkol byl časově náročný, vyžadoval instalaci a zkoušení mnoha programů a vyhodnocování, jaké z nich budou nejvhodnější pro naše testování. Vhodnost jsem posoudil na základě schopností programů (například možnost záznamu dat) a úpravě jejich uživatelského rozhraní (například možnost volby zobrazených informací a jejich pořadí v programu Process Explorer).

4.1 TESTOVANÁ PC SESTAVA, VÝBĚR HARDWARE A OPERAČNÍHO SYSTÉMU

Aby testování bylo co možná nejpřesnější, bylo by ideální zkoušet zátěž webových aplikací na různých strojích a na různých platformách operačních systémů. Protože však nemám k dispozici tolik počítačů a časová náročnost na přípravu a vlastní testování by byla mnohem obsáhlejší, než předpokládaný rozsah práce, proběhne testování pouze na jednom stroji, avšak učiníme takové kroky, aby byly výsledky testů co nejméně zkresleny.

Dalším důvodem, proč bylo testování pojato tímto způsobem, je fakt, že předpokládaný výsledek práce by měl zrcadlit reálné používání internetových aplikací skutečnými uživateli. Podle toho jsem také zvolil referenční sestavu s přiměřeným výkonem a jako operační systém byl použit Windows 7 64-bit. Statistiky společnosti StatCounter [příloha 1] ukazují, že za období květen 2012 až květen 2013 byl v České republice podíl používání operačního systému Windows XP, Vista, 7 a 8 v součtu téměř

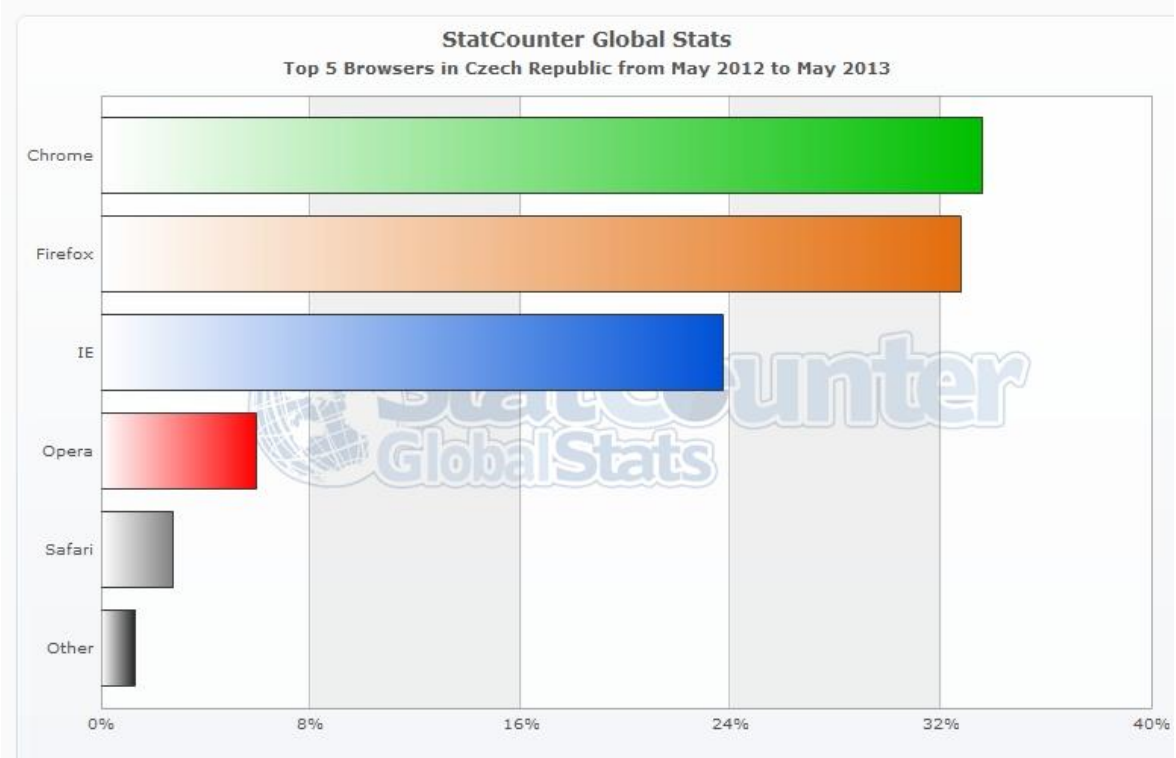
94 %, z čehož Windows 7 tvoří 53,22 %. Jinými slovy pouze šest ze sta počítačů používá jiný operační systém, než Windows.



Obrázek 5: Statistika používaných OS (Zdroj: statcounter.com)

Další statistika [příloha 2], ze které vycházíme, se týká oblíbenosti internetových prohlížečů (obr. 6). Za období květen 2012 až květen 2013 byl v České republice nejpoužívanější Google Chrome (33,58 %) a Firefox (32,73 %). Položka Other (Ostatní) dosahuje pouze 1,31 %.

S ohledem na výše uvedené statistiky budeme testovat 5 nejpoužívanějších prohlížečů běžících v operačním systému Windows 7.



Obrázek 6: Statistika používaných prohlížečů (Zdroj: statcounter.com)

Testování proběhlo na následující sestavě:

Testovaná PC sestava	
Operační systém	Microsoft Windows 7 Ultimate (64-bit)
Procesor	AMD Athlon 64 X2 4400+, 2530 MHz
Základní deska	ASUS M2R32-MVP (AMI BIOS 0804)
Operační paměť	6 GB, A-DATA Extreme edition, DDR2-800 MHz (2x1 GB + 2x2 GB)
Grafický adaptér	ATI Radeon HD 3870 (512 MB), AMD Catalyst 13.1
Displej	U2311H, rozlišení 1920x1080 obrazových bodů
Pevný disk	HD321KJ ATA (320 GB, 7200 RPM, SATA-II)
Optická mechanika	Sony DVD-RW DRU-820A
Napájecí zdroj	CoolerMaster eXtreme Power Plus 460W
Router	SMCWBR14-G2

Tabulka 2: Seznam použitých komponent a operačního systému.

Jak je popsáno výše, operační systém nebyl vybrán dle data vydání, ale dle současné oblíbenosti. Zbylý software byl vyhledán a nainstalován ve své poslední stabilní verzi. Protože se software velmi rychle vyvíjí a jsou často uvolňovány nové verze, bylo

třeba tuto skutečnost brát v úvahu a aktivně sledovat případné aktualizace. V průběhu tvorby této práce se například změnila verze prohlížeče Google Chrome, doplněk Adobe Flash a Java.

Software	Verze
Google Chrome	28.0.1500.52 m
Mozilla Firefox	21.0
Internet Explorer	10.0.9200.16540
Opera	12.15
Safari	5.1.7 (7534.57.2)
AMD Catalyst	13.1
Adobe Flash Player	11.7.700.224 (11.7.700.225 pro Chrome)
Microsoft Silverlight	5.1.20125.0
Microsoft Office	2010

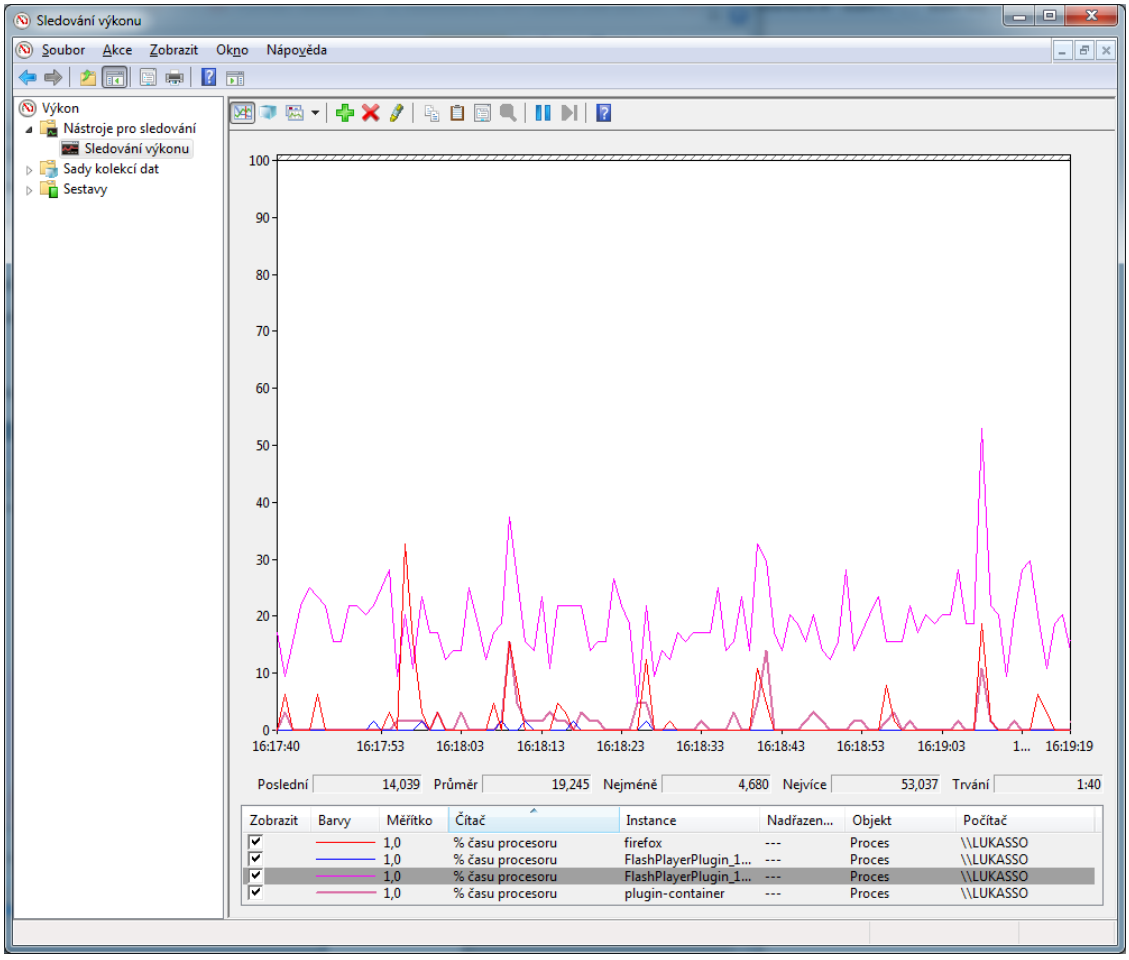
Tabulka 3: Programové vybavení v době testování.

4.2 VYHLEDÁVÁNÍ A VÝBĚR VHODNÝCH PROGRAMŮ PRO TESTOVÁNÍ APLIKACÍ

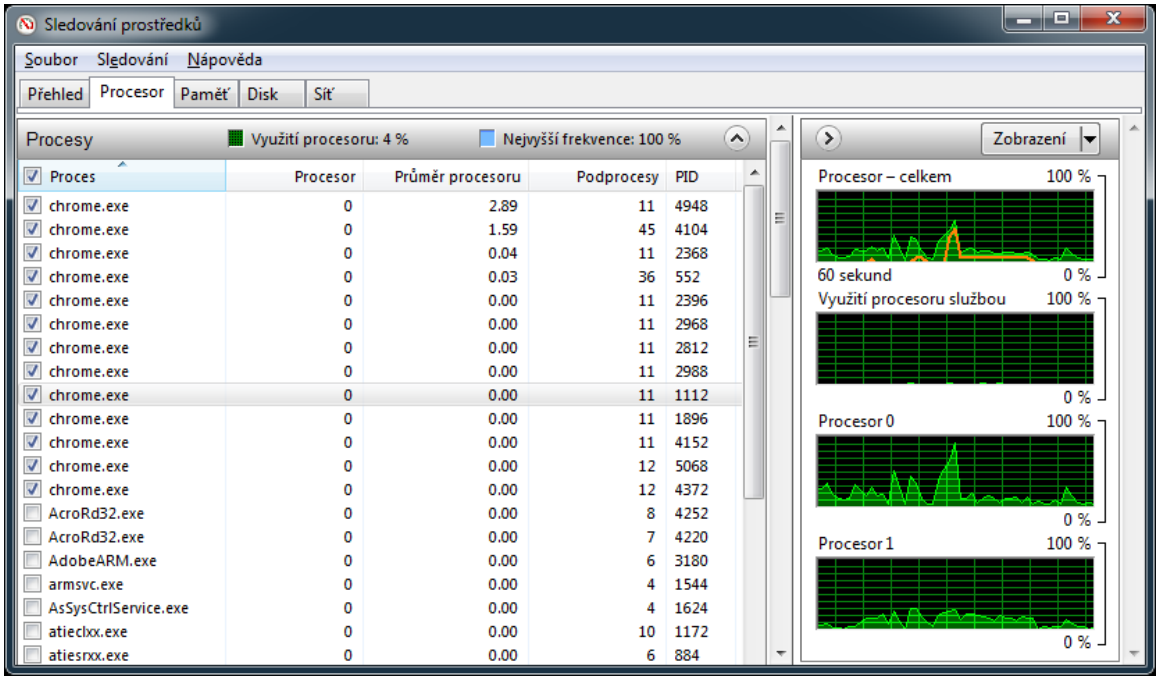
Značně problematický úkol spočíval ve výběru programů vhodných pro zachytávání mezihodnot vytížení jednotlivých komponent při testování aplikací. Operační systém Windows 7 nabízí nástroj Sledování výkonu, který je součástí balíku Nástroje pro správu systému Windows. Tento nástroj má velkou výhodu v tom, že umožňuje velmi rozsáhlé sledování vytížení procesoru, paměti, sítě, ale třeba i různých protokolů, jako TCP, UDP apod. Bohužel však neumí zaznamenat vytížení grafické karty. Síla tohoto nástroje spočívá v možnosti sledování vytížení hardware jednotlivými procesy.

Prostřední programu Sledování výkonu má přehledné grafické rozhraní (obr. 7). Nabízí obrovské množství sledovaných parametrů, v našem případě můžeme například zvolit čítače vykreslující procentuální vytížení procesoru jednotlivými procesy, které jsou spuštěny při běhu prohlížeče Firefox a zásuvného modulu Adobe Flash. Program vykresluje graf a pod ním jsou zobrazeny hodnoty Poslední, Průměr, Nejméně, Nejvíce a Trvání.

Tento nástroj jsem se nakonec rozhodl nepoužít, protože pro naše účely postačí i jednodušší program Sledování prostředků Windows (obr. 8), kde můžeme jednotlivé procesy snadněji vyhledat a označit, což nám ušetří práci s manipulací s čítači. Program Sledování výkonu je však neobyčejně komplexní a velmi vhodný pro sledování různých procesů a služeb. K programům měřící vytížení procesoru je nutno dodat, že jsem nenalezl program, který by uměl měřit vytížení více jader procesoru konkrétním procesem. Monitorovací programy, které jsem nalezl a testoval, zobrazují pouze celkovou zátěž procesoru vyvolanou konkrétním procesem.



Obrázek 7: Sledování výkonu MS Windows - uživatelské rozhraní.



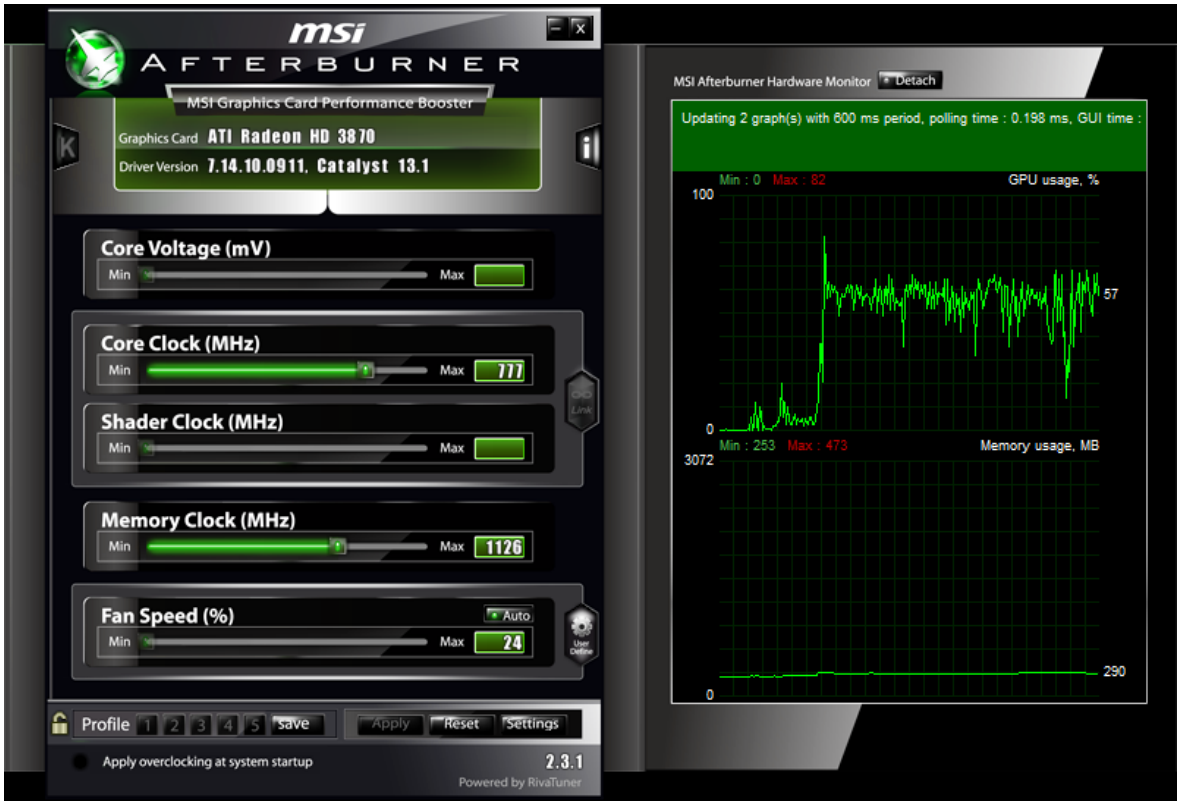
Obrázek 8: Sledování prostředků MS Windows – uživatelské rozhraní.

Výběr programů také determinoval metodiku měření. Například v testu, kdy jsme chtěli sledovat, jak je vytěžován GPU jediným procesem, neexistuje (alespoň veřejně dostupný) program, který by uměl sledovat tento jediný proces a poskytoval výstup hodnot z určitého časového intervalu. Například velmi oblíbený program Process Explorer sice umí zobrazit, jak jednotlivé procesy vytěžují GPU, ale zobrazuje pouze aktuální hodnoty, nestanovuje minima, maxima ani průměr a proto by bylo mimořádně obtížné a nepřesné používat ke sledování vytížení GPU právě tento program. Dále existují programy, které poskytují výstup hodnot vytížení GPU, ale tyto výsledky jsou znepřesněny tím, že zaznamenávají celkové vytížení GPU bez ohledu na to, který proces toto vytížení způsobuje. Nedostupnost programu, který by uměl sledovat jednotlivé procesy a současně zaznamenávat jejich vliv na vytížení GPU celé měření komplikuje.

Pro sledování vytížení GPU jsme použili program MSI Afterburner (obr. 9). Tento program umí zobrazit nejenom identifikační informace o používané grafické kartě (v našem případě dokonce od jiného výrobce – Sapphire), ale umožňuje zobrazovat a logovat například:

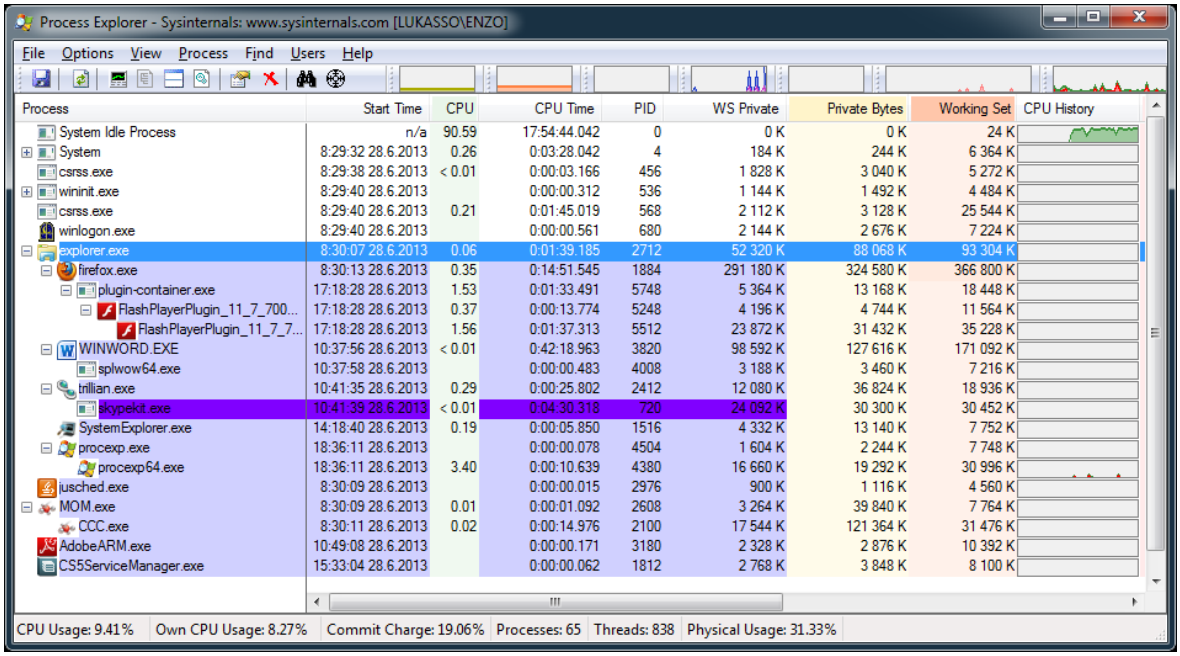
- Teplotu GPU.
- Procentuální využití GPU.

- Rychlost ventilátoru.
- Takt GPU.
- Takt paměti.



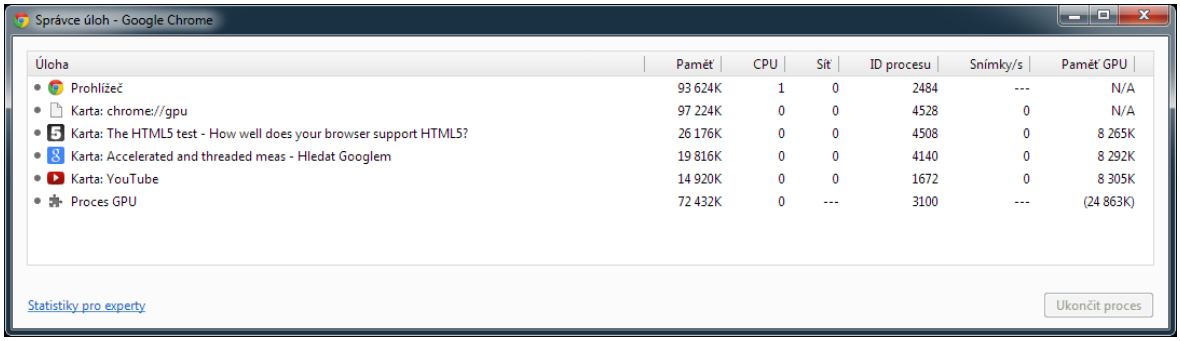
Obrázek 9: MSI Afterburner - uživatelské rozhraní.

Dalším používaným programem pro sledování vytížení paměti, ale také k případné kontrole aktuálního vytížení GPU a CPU a kompletní diagnostice konkrétních procesů a jejich struktuře, použijeme uživatelsky velmi oblíbený program Process Explorer (obr. 10).

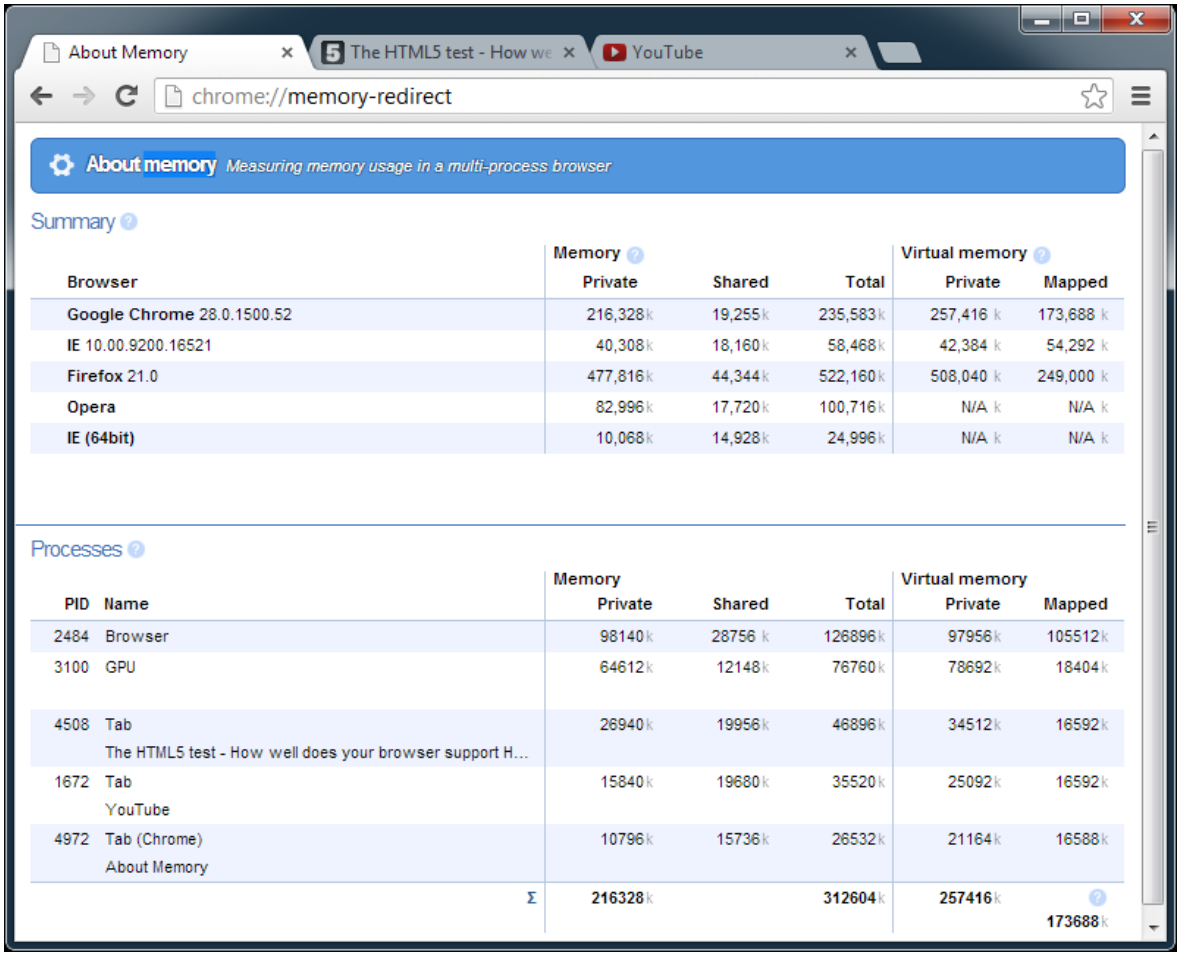


Obrázek 10: Process Explorer - uživatelské rozhraní.

Kromě výše uvedených programů je také možno využít některé vestavěné utility prohlížečů. Asi nejpokročilejší je v tomto ohledu Google Chrome a jeho Správce úloh (obr. 11), a dále lze přímo v prohlížeči vyvolat příkazem *about:memory* podrobný výpis využití paměti všech nainstalovaných prohlížečů (obr. 12).



Obrázek 11: Správce úloh Google Chrome.



Obrázek 12: Výpis vytížení paměti v prohlížeči Google Chrome.

Pro úplnost dodejme, že pro analýzu výsledků programu MSI Afterburner budeme ještě potřebovat program Microsoft Word. Pro tvorbu grafů a tabulek byl použit program Microsoft Excel, obojí z kancelářské sady Microsoft Office 2010.

4.3 VÝCHOZÍ PODMÍNKY PRO TESTOVÁNÍ

Aby výsledky testů byly co nejméně ovlivněny jinými faktory, byly před začátkem testování stanoveny jednotné počáteční podmínky. V případě testování internetových prohlížečů se jednalo o smazání cookies a vymazání jejich paměti cache. Prohlížeče také neobsahovaly žádné rozšíření ani alternativní vzhled a obsahovaly jen potřebné zásuvné moduly, jako Adobe Flash, Microsoft Silverlight a Java. Kromě prohlížeče Opera byly ostatní ponechány ve výchozím nastavení. V prohlížeči Opera bylo nutné pomocí vnitřního nastavení vyvolaného příkazem *about:config* zapnout hardware akceleraci a WebGL. Tyto technologie jsou sice podporovány, ale defaultně vypnuty. O jednotlivých technologiích

bude pojednáno později, nyní pouze zmíníme, že důvodem pro tuto změnu nastavení bylo zajištění rovnocennosti při testech rychlosti vykreslování grafiky.

Kromě prohlížečů, monitorovacích programů a poznámkového bloku pro záznam naměřených hodnot, nebyly žádné jiné aplikace spuštěny. V případě testování prohlížeče Internet Explorer je dobré nespouštět před ním žádný program z kancelářského balíku Microsoft Office nebo Průzkumníka Windows. Tyto aplikace totiž využívají stejného renderovacího jádra, jako Internet Explorer a v případě, že bychom testovali i rychlost spuštění prohlížeče, byl by Internet Explorer zvýhodněn tím, že jeho jádro by bylo již spuštěné.

5 WEBOVÉ PROHLÍŽEČE A TESTOVÁNÍ JEJICH FUNKCIONALIT

Webový prohlížeč je software, který umožňuje prohlížení WWW stránek. Umožňuje komunikaci se serverem. Zpracovává přijatý kód ve formátu HTML, XHTML nebo XML, který podle daných standardů zformátuje a zobrazí webovou stránku. Pro zobrazení některých doplňujících částí stránky, jako jsou Flash a Java animace (aplikace), je třeba prohlížeč doplnit o specializované zásuvné moduly. [19]

„Webový klient není ani zdaleka jen programem, ve kterém jsou pouze zobrazovány stránky. Rozhraní WWW se běžně používá pro množství různých aplikací, od prostého výběru (a posílání) e-mailů, přes práci s elektronickým bankovnictvím až po složité podnikové aplikace nebo konfigurační nástroje síťových zařízení.“ [20] Dnešní ideou je běh aplikací v prostředí prohlížeče. Běžně takto fungují například kancelářské aplikace. Jejich otevírání přímo v okně prohlížeče urychluje práci a usnadňuje přístup k dokumentům, neboť jsou prezentovány stejným rozhraním, kterým byl na serveru nalezen a stáhnut. To s sebou však přináší nevýhodu ve formě vyšších nároků na hardware počítače. „*“Spotřeba“ systémových prostředků Acrobat readerem běžícím v prostředí Internet Exploreru je vyšší, než kolik by jich obě tyto aplikace vyžadovaly samostatně.“* [20]

TESTOVANÉ PROHLÍŽEČE A JEJICH VLASTNOSTI

Všechny testované prohlížeče jsou stabilní verze. Vývojové (testovací, v angličtině označované jako RC – Release Candidate) verze nebyly testovány, neboť mohou obsahovat chyby implementací nových funkcí, není garantována stabilita a také v těchto verzích nejsou naplno využívány optimalizace JavaScript enginů. To vše může mít za následek nesprávný průchod a nedokončení vybraných testů.

	Mozilla Firerfox	Google Chrome	Internet Explorer	Opera	Safari
Vývojář	Mozilla Corp.	Microsoft Corp.	Google	Opera Software ASA	Apple Inc.
Verze	21	28	10	12.15	5.1.7
Renderovací jádro	*Gecko*	*WebKit*	*Trident*	*Presto*	*WebKit 2*

JavaScript jádro	IonMonkey	V8	Chakra	Carakan	Nitro
HTML5 HW akcelerace	Ano	Ano	Ano	Ano	Ne
WebGL	Ano	Ano	Ne	Ano	Ne
Architektura	32-bit	32-bit	32bit, 64-bit	32bit, 64-bit	32-bit

Tabulka 4: Souhrn vlastností internetových prohlížečů.

5.1 CO VŠE LZE TESTOVAT U PROHLÍŽEČŮ

Prohlížeč je velmi komplexní software a proto lze testovat jeho různé funkcionality. Z dostupných testů internetových magazínů jsem zjistil, že je testování nejčastěji zaměřeno na:

- **Rychlost startu**. Může být testován tzv. studený start, kdy je prohlížeč spuštěn poprvé po zapnutí PC či teplý start, tedy opětovné spuštění již dříve spuštěného a zavřeného prohlížeče.
- **Rychlost načtení stránky**. Obvykle se využívá kombinovaného měření rychlosti načtení s vymazanou a nevymazanou cache prohlížečů.
- **Výkon JavaScript interpreta**. Jedna z nejdůležitějších vlastností mající vliv na celkový výkon prohlížeče. Mnohé srovnávací testy vycházejí pouze z výsledků testů testujících výkon JavaScriptu. Touto problematikou se budeme zabývat v samostatné kapitole, ve které o JavaScript jádrech pojednáme teoreticky a přikročíme k praktickému testování.
- **Rychlost práce s DOM a CSS.** Testy probíhají obvykle jako simulace vytváření a transformace objektového DOM stromu, přidávání nových uzlů do tohoto stromu, dále posouvání stromové struktury s pomocí různých knihoven, například jQuery. Testování rychlosti DOM je důležité, neboť je součástí renderovacího procesu, tedy reálného užívání a chování internetového prohlížeče. Řada testovacích sad, včetně nejznámějších (SunSpider, Octane) však rychlost DOM vůbec netestuje, a proto výsledky takových testů méně odrážejí reálné chování prohlížečů. V syntetických testech se také může zkoumat rychlost získávání a nastavování vzhledu

pomocí CSS. CSS lze také nastavovat právě pomocí DOM. O renderovacím procesu a DOM bude pojednáno později.

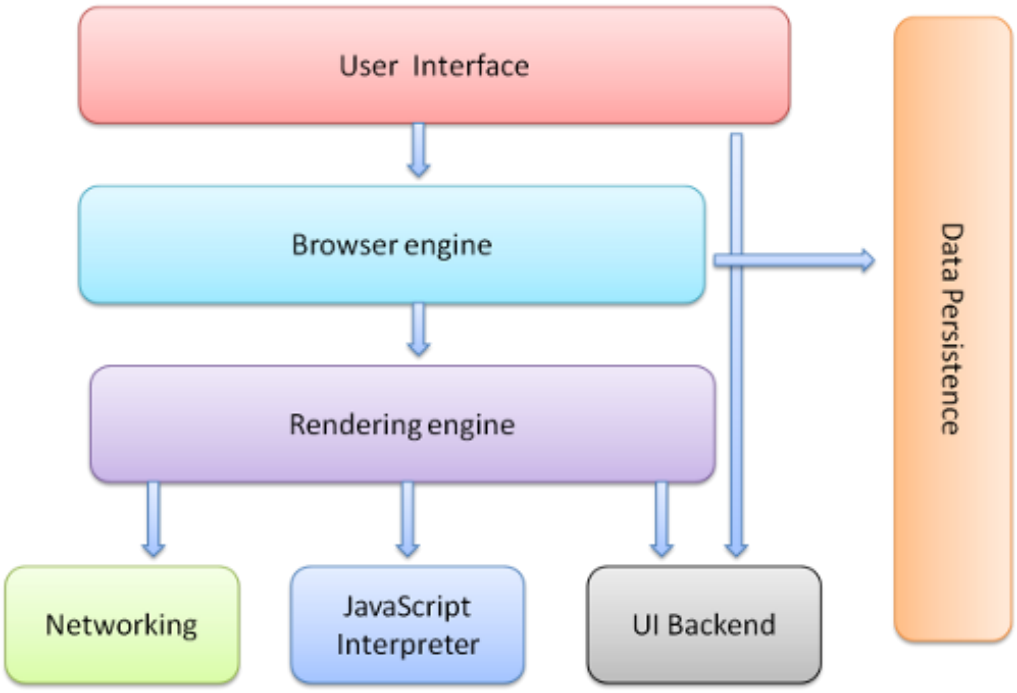
- **Rychlost a validace HTML5.** Některé online testy, např. *html5test* nebo částečně benchmark Peacekeeper se zaměřují na míru podpory jazyka html5. Dále se zkoumá výkon některých objektů HTML5, například rychlost vykreslování pomocí objektu <*canvas*>.
- **Využití hardwarové akcelerace**. Z pohledu zatížení počítačových komponent je podpora hardwarové akcelerace jedna z klíčových oblastí, která ovlivňuje výkon prohlížeče. Pokud prohlížeč podporuje hardwarovou akceleraci, je k vykreslování veškerého obsahu na obrazovku využíván grafický procesor na místo CPU. Díky tomu je možné dosáhnout plynulejšího zobrazování videí nebo her. Této oblasti se též budeme věnovat v samostatné kapitole.
- **Výkon zásuvných modulů.** Především Adobe Flash, Microsoft Silverlight, dnes méně často i Java. Protože jsou tyto zásuvné moduly velmi často využívané a najdeme je ve většině dynamických webů, budeme se jimi zabývat teoreticky a změříme i výkon jejich implementací v jednotlivých prohlížečích.
- **Využití paměti**. Obvykle je sledována velikost využité paměti, kterou zabírá proces (nebo více procesů) při otevření určitého počtu záložek a jejich následném zavření.

5.2 CO OBSAHUJE WEBOVÝ PROHLÍŽEČ

Hlavními komponentami webových prohlížečů jsou (obr. 13):

- **Uživatelské rozhraní (User interface)** – obsahuje adresní řádku, tlačítka na přechod o stránku zpět nebo dopředu, menu záložek, tlačítko Domů, Obnovit apod.
- **Jádro prohlížeče (Browser engine)** – rozhraní mezi uživatelským rozhraním a renderovacím jádrem.

- **Renderovací jádro (Rendering engine)** – je zodpovědné za zobrazování požadovaného obsahu. Pokud je například požadovaným obsahem HTML, má na starosti analýzu HTML a CSS a zobrazování analyzovaného obsahu na obrazovku. O renderovacím jádru bude pojednáno později.
- **Síťový subsystém (Networking)** – používá se k síťovým voláním, například HTTP požadavkům. Je platformě nezávislý.
- **Backend uživatelského rozhraní (UI backend)** – používá se pro vykreslování základních ovládacích prvků, jako jsou pole a okna. Zpřístupňuje obecné rozhraní, které není platformě specifické. Využívá metod uživatelského rozhraní operačního systému.
- **JavaScript interpret** – slouží k analýze a provádění JavaScript kódu.
- **Datové uložiště (Data persistence)** – stálá vrstva, které slouží k ukládání všech druhů dat na pevný disk, například cookies, záložek a cache. [8]

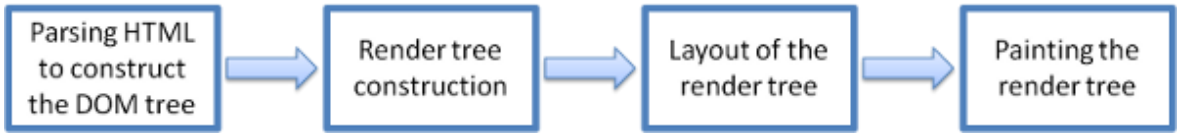


Obrázek 13: Hlavní komponenty prohlížeče (Zdroj: taligarsiel.com)

5.3 RENDEROVACÍ PROCES A JÁDRA PROHLÍŽEČŮ

Renderovací jádro prohlížeče má za úkol zobrazit požadovaný obsah na displej. Defaultně může renderovací jádro zobrazovat HTML a XML dokumenty a obrázky.

Využitím zásuvných modulů může zobrazovat i jiné formáty dat, například PDF dokumenty použitím příslušeného doplňku (Adobe Acrobat). Průběh renderovacího procesu je zobrazen na obrázku 14.



Obrázek 14: Schématický rozbor renderovacího procesu. (Zdroj: taligarsiel.com)

Renderovací jádro začne parsovat (parsování = syntaktická analýza, upřesněno ve slovníku pojmů) HTML dokument a z tagů (značek) vytvoří DOM uzly ve stromě, který se nazývá obsahový strom (content tree). Ten dále zpracovává data o stylech pomocí externích CSS souborů a tagů HTML určujících styly. Tato data slouží k vzniku dalšího stromu – tzv. render tree (renderovací strom). Renderovací strom obsahuje obdélníky s vizuálními atributy, jako jsou barva a rozměry. Tyto obdélníky jsou seřazeny v pořadí, v jakém mají být zobrazeny na displej. Po konstrukci renderovacího stromu je třeba přiřadit každému uzlu tohoto stromu správné souřadnice pro zobrazení. Poslední fází renderovacího procesu je vykreslování (painting). Při procházení renderovacího stromu se každý uzel vykresluje pomocí Backendu uživatelského rozhraní. [8]

Je třeba dodat, že celý proces je urychlován tím, že se nečeká na analýzu HTML, aby mohl být sestaven renderovací strom. Tento proces probíhá průběžně, kdy se část obsahu analyzuje a zobrazí, zatímco zbytek obsahu průběžně přichází ze sítě. [8] Každé renderovací jádro takto principiálně funguje, ale rozdíly mezi jádry jsou. Například jádro Gecko sestavuje strom z rámů, zatímco WebKit využívá objekty. [8]

DOM

V odstavci výše jsme použili termín DOM. DOM je zkratka z anglického Document Object Model, což je organizací W3C standardizovaná objektová reprezentace HTML a XML dokumentů. Tato reprezentace je platformě i jazykově nezávislá. Umožňuje dynamicky přistupovat a měnit obsah, strukturu a styl dokumentu. Standard vznikl jako reakce na používání rozdílných vlastností a metod pro popis objektu, které v 90. letech používali prohlížeče Netscape Navigator a Microsoft Explorer.

Protože je tvorba DOM, respektive jeho stromová reprezentace součástí renderovacího procesu, je důležitá i rychlost práce s DOM. Rychlost DOM je testována např. v testovací sadě Dromaeo nebo Peacekeeper. [21]

5.4 TESTOVÁNÍ RYCHLOSTI A TEORIE JAVASCRIPTU

Jeden z nejzásadnějších faktorů, který má vliv na rychlost a výkon prohlížeče, je implementace JavaScriptu. Interprety tohoto jazyka obsažené ve webových prohlížečích se ale liší. Cílem vývojářů je co nejlepší optimalizace a urychlení spouštěného kódu. Potřeba optimalizace vzrostla především s rozšiřováním webových aplikací a dynamických stránek vůbec. JavaScript byl v minulosti, zhruba od poloviny 90. let minulého století) na svém počátku pouhým okrajovým doplňkem pro tvorbu jednoduchých dynamických prvků webových stránek. Jednalo se především o různé efekty zobrazení obrázků nebo práce s formuláři.

Postupem času se však obliba dynamických webů rozšiřovala, zejména díky webovým aplikacím jako je Gmail, a dále nástupem technologie AJAX a knihovnám jQuery. Tím však také rostla náročnost a z krátkých programů o několika řádcích kódu se staly stovky kilobajtů dat. Přestože se poměrně dlouho vývojáři prohlížečů nezabývali optimalizací implementací, situace posledních let je jiná. Používání webových aplikací často komplikovala pomalost JavaScript interpretů, což vedlo spolu s rozšířeností dynamických webů k tlaku na vývojáře, kteří se zrychlováním JavaScriptu začali zabývat. Inovace v této oblasti resultovaly ve vývoj jednotlivých JavaScript enginů, tedy JavaScript jader prohlížečů, které spolu s renderovacím jádrem tvoří jejich základní součást. JavaScript se stal prvním dynamickým jazykem, který byl tak důležitý a dostatečně rozšířený na to, aby se vyplatil výzkum v oblasti jeho zrychlování. První velké změny v oblasti vývoje a implementací JavaScriptu probíhaly zhruba od roku 2007. Naplno se konkurenční souboj v rychlosti JavaScript enginů rozběhl v roce 2008. [22]

Problematika implementace JavaScript interpretů je však velmi rozsáhlá a její analýza není předmětem této práce. Uvedeme tedy pouze nejdůležitější faktory, které mají vliv na dynamičnost a výkon JavaScript interpretu. Jedná se především o problematiku volání metod, přístup k atributům objektů a dynamické typování

proměnných. Kromě jazykových vlastností však ovlivňují rychlost JavaScriptu i vlastnosti technické, především to, jakým způsobem probíhá samotná interpretace.

Zjednodušeně lze způsob psaní interpretu rozdělit na tři základní techniky:

1. Využití abstraktního syntaktického stromu (AST). Strom se postaví ze zdrojového kódu a reprezentuje strukturu programu. Strom je složen z uzlů, které představují jednotlivé elementy programu (např. příkaz *if*) a poduzlů reprezentující součásti těchto příkazů (např. součástí příkazu if je podmínka a obě jeho větve). Strom je pak interpretem procházen a vykonává jednotlivé příkazy.
2. Rychlejším způsobem je průchod vystavěného stromu a z něj následné vygenerování *bajtkódu*, což jsou sekvenční instrukce, např. volání funkcí, vykonávání matematických operací atd. Ty jsou následně interpretovány virtuálním strojem, na jehož návrhu také závisí celkový výkon interpretu. Virtuální stroje buď jsou zásobníkové, nebo registrové. Nelze jednoznačně říci, který typ virtuálního stroje je lepší, ale v posledních letech se využívá především strojů registrových. Kromě volby a návrhu virtuálního stroje dále závisí výkon na použité instrukční sadě a na optimalizacích používaných při zpracovávání instrukcí.
3. Ještě rychlejší intepretace je možné dosáhnout, pokud z AST nebo z bajtkódu vygenerujeme přímo strojový (nativní) kód procesoru. Tento způsob je však složitější, přináší závislost na konkrétní platformě a využívá se jej až v posledních letech. Efektivnějšího využití se dosáhne tím, že se program kompiluje po částech za běhu, nikoliv předem, jako u klasických kompilátorů. Díky tomu se šetří čas, který by byl zapotřebí ke kompilaci všech funkcí a metod, tedy i těch, které nejsou vůbec volány. Tato technologie se nazývá Just-In-Time (JIT) a je dnes běžnou kompilační technikou JavaScript jader. [23]

5.4.1 SCÉNÁŘ TESTOVÁNÍ

Testovací sady SunSpider, Kraken Benchmark a Octane spustíme 3x, vždy po novém startu počítače. Výsledné grafy (obr. 15) budou reprezentovat průměry těchto tří

průběhů. Testovací sadu Dromaeo jsem chtěl též spustit 3x, ale jelikož ani jednou nedoběhlo testování do konce u prohlížečů Google Chrome a Safari, usoudil jsem, že průběh zbylými prohlížeči bude mít pouze doplňující charakter a nebude uvažován při vyhodnocování výkonu JavaScript interpretů. Protože význam sady Dromaeo tkví mimo jiné v testování výkonu DOM, které patří k součásti renderovacího procesu a mělo by být v testech zahrnuto. Práce s objekty DOM je zahrnuta například v testu Peacekeeper.

5.4.2 TESTOVACÍ SADY, KRÁTKÉ PŘEDSTAVENÍ

SUNSPIDER – testovací sada pocházející od tvůrců WebKitu obsahuje sadu testů, které se zaměřují na většinu operací, s nimiž se musí JavaScript jádro potýkat při běžném užívání. Jedná se především o kryptografické testy, testy dekomprimace, 3D ray-tracing nebo tvorba tag cloudu z JSON vstupu. [24] Výsledkem je rychlost zpracování udaná v milisekundách.

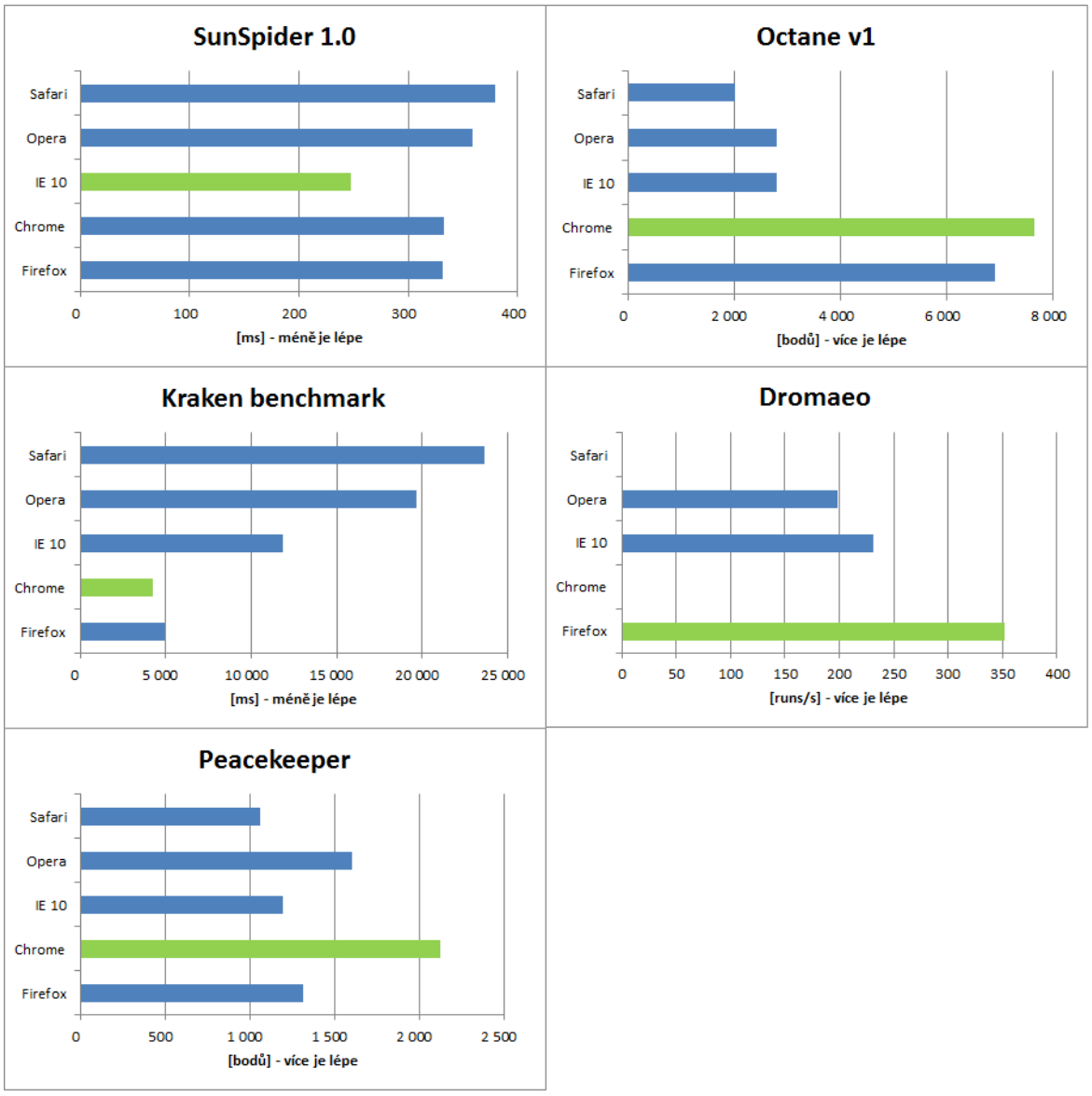
OCTANE V1 – testovací sada vytvořená Google Developers vychází z dříve úspěšného V8 Benchmark Suite (V8 BS). Obsahuje 13 testů, z nichž 8 pochází právě ze zmíněného V8 BS. Kromě klasických testů (šifrování, dešifrování, ray-tracing apod.) měří např. rychlost dekódování a interpretace PDF Readeru. Výsledkem je počet bodů.

DROMAEO – testovací sada vyvinuta společností Mozilla Foundation nabízí uživateli čtyři kolekce testů – doporučené testy, ryze JavaScriptové, ryze DOM a všechny dostupné. Doporučené testy, které jsme zvolili, obsahují všechny testy, kromě testů CSS selektorů. Těchto testů je celkem 51 a testují například šifrování a dešifrování AES, práci s poli, regulární výrazy, práci s 2D i 3D objekty a samozřejmě manipulaci s DOM. Výsledkem je počet průběhů za sekundu, více je lépe.

KRAKEN BENCHMARK – také testovací sada společnosti Mozilla. Méně rozsáhla sada testů vycházející ze sady SunSpider. Zajímavostí je např. test A* (A star) vyhledávacího algoritmu, který hledá a vykresluje cestu mezi dvěma body a míjí překážky v cestě. Dále obsahuje testy pracující se zvukem, obrazem, JSON a šifrovacími algoritmy. Výsledkem je rychlost zpracování udaná v milisekundách.

PEACEKEEPER – testovací sada společnosti Futuremark testuje rychlost JavaSciptu, do níž je zahrnuta i práce s DOM. Na rozdíl od sady Dromaeo tímto testem prošli všechny prohlížeče bez problému. Kromě rychlosti JavaScriptu také testuje podporu některých

HTML5 prvků, například podporované formáty videa nebo WebGL. Tyto výsledky však nejsou zahrnuty do celkového skóre rychlosti JavaScriptu, ale jsou vyjádřeny mimo hlavní výsledek jako výše podpory HTML5, kdy je prohlížeč ohodnocen max. 7 body.



Obrázek 15: Souhrn výsledných grafů testovacích JavaScript sad.

Z výsledků (obr. 15) je patrné, že nejpomalejší v interpretaci JavaScriptu je Safari. Velmi vyrovnané výsledky vykazoval Firefox a Chrome, Internet Explorer však bodoval v testu SunSpider.

5.5 TESTOVÁNÍ ZÁSUVNÝCH MODULŮ

5.5.1 ADOBE FLASH PLAYER

Adobe Flash Player je multiplatformní aplikační běhové prostředí založené na prohlížeči a určené pro prohlížení obsahově bohatých aplikací, obsahu a videa na různých obrazovkách a prohlížečích. [25] Je to patrně nejrozšířenější doplněk webových prohlížečů vůbec. Na svých stránkách [26] společnost Adobe uvádí, že tento produkt využívá více než 1,3 miliardy lidí.

Adobe Flash používá objektově orientovaný skriptovací jazyk ActionScript (dále jen AS), který je nyní ve třetí vývojové generaci označován jako ActionScript 3.0. Protože není tématem této práce rozbor jazyka AS, uveďme pouze, že oproti verzi 1 se AS 3.0 provádí zhruba 10x rychleji a to především díky využití vysoce optimalizovaného virtuálního stroje ActionScript Virtual Machine 2 a jinému principu dědičnosti založeném na existenci dvou oddělených mechanismů dědičnosti: dědičnosti pevné vlastnosti a dědičnosti prototypu. *„V předcházejících verzích jazyka ActionScript mohlo být vyhledání názvu, kdy přehrávač Flash Player prochází řetězec prototypu, časově náročným procesem. V jazyce ActionScript 3.0 je vyhledání názvu daleko účinnější a méně časově náročné, protože zděděné vlastnosti jsou kopírovány z nadřazených tříd do objektů vlastností podtříd.“*[27]

Rozdílná je také implementace Adobe Flash. Dnes existují tři základní – ve formě zásuvných modulů pro Firefox, Chrome a Opera, jako ovladač ActiveX pro Internet Explorer anebo v poslední době tzv. Pepper-based implementace v Google Chrome. Pepper je aplikační programové rozhraní (API), které slouží ke zvýšení robustnosti webového prohlížeče díky skutečnosti, že je Adobe Flash implementován přímo v něm. Přímá implementace také poskytuje jistý uživatelský komfort, neboť odpadá nutnost doplněk samostatně instalovat a aktualizovat. Tato technologie má zajistit lepší interakci mezi prohlížečem a běhovým prostředím Flash a očekává se, že společnost Adobe bude Pepper prosazovat v budoucnu i u jiných prohlížečů. [28]

5.5.2 MICROSOFT SILVERLIGHT

„Microsoft® Silverlight™ je nejmodernější technologie pro internetové prohlížeče. Je to platforma určená pro tvorbu dynamického online obsahu a interaktivní práce s ním. Kombinuje text, vektorovou i bitmapovou grafiku, animace a video.

Pomocí malé stažitelné komponenty (plug-in) umožní interaktivní ovládání her nebo aplikací a přehrávání multimédií ve většině současných webových prohlížečů (Internet Explorer, Firefox, Safari, Opera, Chrome) na platformách Windows a Mac OS X. Na Linuxu je dostupný pod názvem Moonlight, vyvinutý společností Novell." [29] Tento text je citován z oficiálních stránek společnosti Microsoft. Je však otázkou samotná budoucnost projektu Silverlight, neboť používání tohoto doplňku není na internetu rozšířené tak, jako jiné konkurenční doplňky. Například velká část videí užívající ke svému přehrávání Silverlight se nachází přímo na stránkách Microsoft nebo na blogách jejich pracovníků. Trend dnešní doby jde ale jiným směrem – Silverlight je postupně zapomínán a i Adobe Flash ztrácí půdu pod nohama díky HTML5. Sice zatím neexistuje oficiální vyjádření k budoucnosti Silverlight, ale některé zdroje uvádějí, že s podporou současné verze 5 do roku 2021 tato platforma zanikne. [30]

5.5.3 FLASH VS. SILVERLIGHT

Uživatelé se s aplikací Silverlight setkávají především ve formě přehrávače, podobně, jako si lidé k technologii Adobe Flash/Flex přiřadí pouze přehrávač Flash Player. Je však třeba zdůraznit, že výše uvedené technologie nejsou jen přehrávače, ale primárně běhové prostředí, ve kterém je možné vytvářet například obchodní aplikace, animace, hry a multimediální RIA aplikace. [31]

Následně si popíšeme několik zásadních rozdílů mezi Flash a Silverlight technologiemi.

ANIMACE

Flash využívá *frame-based* animační model, tedy model založený na animaci rámec po rámci. To znamená, že musíme vytvářet objekt pro každý rámec, spočítat dle trvání animace počet rámců a tyto rámce rozdělit podle zvolené metriky – trasy pohybu. Jako program pro tvorbu Flash animací se používá především Adobe Flash Professional, nicméně objekty používané pro animaci je možné vytvořit pomocí dalších programů z rodiny Adobe, zejména Adobe Illustrator.

Silverlight používá vlastní WPF (Windows Presentation Found) model, založený na .NET framework, využívající jazyk XAML. Animace je zde založena na existenci časové osy. Stanoví se pouze počáteční a koncový pohyb, trajektorie pohybu a vlastní výpočet pohybu

je realizován programem. Vývojové prostředí pro tvorbu konzolových aplikací a jejich grafického rozhraní se využívá zejména program Microsoft Visual Studio a Microsoft Expression.

VELIKOST SOUBORU

Flash využívá komprimovaný formát s příponou .swf (kromě toho je také možné exportovat Flash do .exe souboru spustitelného v prostředí Windows. Silverlight ke spuštění potřebuje přehrávač vždy). Tento soubor však vyžaduje přítomnost Adobe Flash Player plug-in. Velikost tohoto souboru je relativně malá díky faktu, že veškeré obrázky a text jsou zahrnuty přímo do výsledného videa. Tento fakt měl dříve nežádoucí efekt pří SEO optimalizaci. Protože text je součástí videa a nebyl separován jako samostatná komponenta, stával se praktický neviditelným pro vyhledávače – nemohl být indexován. Přelomem se stala spolupráce společností Adobe a Google, kdy byla vyvinuta sada nástrojů pro Flash umožňující indexaci vyhledávačem Googlebot.

Uživatelské rozhraní Microsoft Silverlight je definováno pomocí deklarativního jazyka XAML využívající část .NET framework. Text je na server umístěn nezávisle a je dosažitelný nezávisle na zbytku grafiky, což umožňuje jeho indexaci a tedy SEO přívětivost od samého počátku. Text v Silverlightu tedy není kompilován, ale reprezentován jako text prostřednictvím jazyka XAML. Výsledný soubor není komprimovaný, a proto je jeho velikost obvykle větší. [32]

SKRIPTOVÁNÍ

Flash využívá objektově-orientovaného jazyka ActionScript, který obsahuje řadu ovládacích prvků pro tvorbu uživatelského rozhraní. Může být integrován s technologiemi, které využívají jiné jazyky a frameworky, např. PHP, ASP.NET nebo Ruby. Díky tomu je možné využívat rozsáhlou knihovnu tříd pro vývoj jak webových, tak samostatných desktop aplikací

Silverlight lze programovat s využitím několika programovacích jazyků, především Visual C#.NET a Visual Basic.NET, včetně skriptovacího jazyka pro skriptování z klientské strany pomocí JavaScriptu. Jazyky C# a VB.NET mohou využívat všech možností a potenciálu frameworku Microsoft.NET. [33]

OSTATNÍ

Další rozdíly spočívaly dříve například v podpoře webových kamer a mikrofonů, kterou Silverlight podporuje až od verze 4. Velmi proměnlivá byla také podpora formátů obrázků, kde jako první nabízel širší podporu Flash, ale s každou vydanou verzí se škála podporovaných formátů rozrůstala i v případě Silverlightu. Stejná situace byla například s audio/video kodeky. Co se týče podpory platforem, podporuje Flash jak Windows, tak Linux a Solaris. Silverlight jako takový je podporován pouze platformou Windows, existuje však verze Moonlight, což je svobodná re-implementace Silverlightu pro platformu Linux. [32]

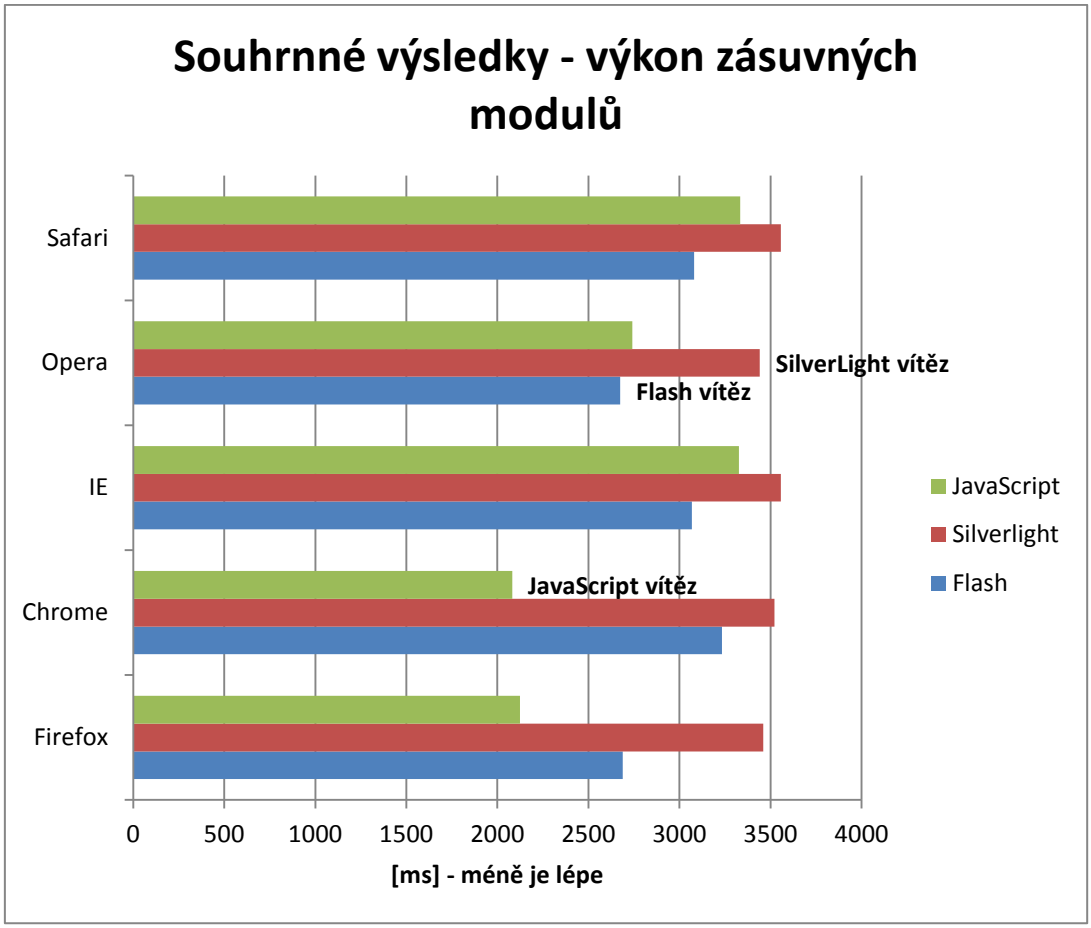
5.5.4 TESTOVÁNÍ FLASH VS. SILVERLIGHT VS. JAVASCRIPT

Protože se v této práci zaměřujeme zejména na vytížení hardware, otestujeme právě ty doplňky, jejichž běh má největší vliv na využití systémových zdrojů. O jednotlivých technologiích bylo pojednáno výše, JavaScript a výkon jednotlivých interpretů byl navíc testován zvlášť v kapitole 5.5.2.

Testování těchto technologií proběhlo pomocí testovací sady RIABench, která se zaměřuje na testování technologií JavaScript, Flash, Silverlight a Java. Výhodou této sady je, že obsahuje stejné testy využívající různé technologie. Proto můžeme vyhodnotit výsledky jednotlivých měření nejenom z absolutního hlediska, ale můžeme sledovat, jak jsou jednotlivé technologie výkonné ve spojení s jednotlivými prohlížeči.

Vybráno bylo 5 testů spustitelných pomocí Flash, Silverlight a JavaScript, a každý byl spuštěn 3x. Jednalo se o testy:

- **Primetest** – generování 20 000 prvočísel.
- **Focustest** – komplexní sada testů využívající řetězce, pole a matematické operace.
- **MD5 encoding** – hašování 455 kB ASCI textu s využitím MD5 algoritmu.
- **Random Key generator** – generování 1 000 000 náhodných čísel využitím matematické metody lineární kongruence.
- **Run-length encoding** – komprese 12 kB ASCI textu s kompresním poměrem 13,2 %.



Obrázek 16: Výsledky výkonu zásuvných modulů.

Výsledky jednotlivých testů jsou uvedeny v příloze [příloha 3]. Pro analýzu jednotlivých testů zde není dostatečný prostor, samotná analýza by vydala na samostatnou práci. Na tomto místě alespoň uvedeme, že žádná technologie neměla absolutní náskok ve všech testech. Největší vliv na výsledky měla povaha testů – zatímco byl Silverlight oproti JavaScriptu a Flash násobně rychlejší v testech MD5 encoding a Random Key generator, v testu Run-lenght encoding byl násobně pomalejší oproti JavaScriptu. Nelze tedy s určitostí říci, že jedna technologie předčí všechny ostatní ve všech ohledech. Graf souhrnných výsledků je pak součtem délek jednotlivých testů prováděných jednotlivými technologiemi. Dále interpretace celkových výsledků může zkreslovat fakt, že i když v součtu celkového průběhu zvítězil Chrome a dvakrát Opera, 9 z 15 dílčích testů (5 testů * 3 technologie) vyhrál Firefox.

5.6 MODERNÍ TECHNOLOGIE PROHLÍŽEČŮ A JEJICH TESTOVÁNÍ

5.6.1 ZNAČKA *<CANVAS>*

Příchod HTML5 a znamenal příchod některých nových elementů. Mezi nejdůležitější patří značka *<canvas>*, která umožňuje vykreslování 2D a 3D objektů pomocí skriptovacího jazyka, zejména pomocí JavaScriptu. Canvas je obdélníková plocha na HTML stránce, která sama o sobě nemá žádný obsah ani hranice. Kreslení probíhá pomocí tzv. kontextů: 2D kontext a 3D kontext (WebGL). 2D kontext je podporovaný všemi námi testovanými prohlížeči a jeho použití je mnohem častější. *„Tento kontext nabízí jednoduché, ale poměrně mocné API, pomocí něhož kreslíte do 2D bitmapového „plátna“. Nemá specifikovaný „formát souboru“ a lze kreslit jen skripty. Není tu žádný DOM pro nakreslené tvary, vše je pouze v bitmapě, jako pixely. To znamená, že budete-li kreslit další a další čáry, křivky, objekty, …, nebude se nikde v paměti vytvářet stále složitější objektový model.“* [34] To přináší výhodu v podobě stálosti výkonu, neboť výkon klesá pouze s rostoucími rozměry obrázku a nikoliv se složitostí, jako je tomu v případě použití vektorového formátu SVG. Canvas je tedy vhodný pro bitmapovou grafiku (hry, editory obrázků…), navíc všechny námi testované prohlížeče kromě Safari podporují HTML5 hardware akceleraci, kam patří také akcelerace 2D kontextu. Jestli se akcelerace projeví na výkonu prohlížeče. [34]

5.6.2 WEBGL

WebGL (Web Graphics Library) je rozšíření značky *<canvas>*, využívající rozhraní DOM. WebGL je JavaScriptové API pro vykreslování 2D i 3D grafiky v prohlížeči bez nutnosti používat plug-in. Největší výhodou WebGL je, že programy obsahují kromě obslužného kódu napsaného v JavaScriptu i kód shaderu, který je prováděn GPU. [35] Díky tomu se na vykreslování grafiky přímo podílí grafická karta, v čemž spočívá podstata hardwarové akcelerace. Tu však musí podporovat prohlížeč.

Přesto, že WebGL umožňuje zefektivnění práce počítačových komponent a tvorbu uživatelsky zajímavých stránek, ne všechny prohlížeče jej podporují. Námi testovaný Firefox 21 a Google Chrome 28 podporuje WebGL defaultně, Opera 12.15 nikoliv, ale je možné jej zapnout. Internet Explorer 10 WebGL nepodporuje vůbec a Safari pouze od verze 6, která však prozatím není dostupná pro platformu Windows.

5.6.3 TESTOVÁNÍ HARDWARE AKCELERACE

Pro zjištění, jak si prohlížeče vedou při vykreslování grafiky a jak umí spolupracovat s grafickou kartou, bylo vybráno několik testů, které se zaměřují jak na samostatné testování výkonu canvasu a WebGL, tak i na HW akceleraci jako takovou. Pro testování výkonu canvasu a WebGL jsem zvolil tyto testy:

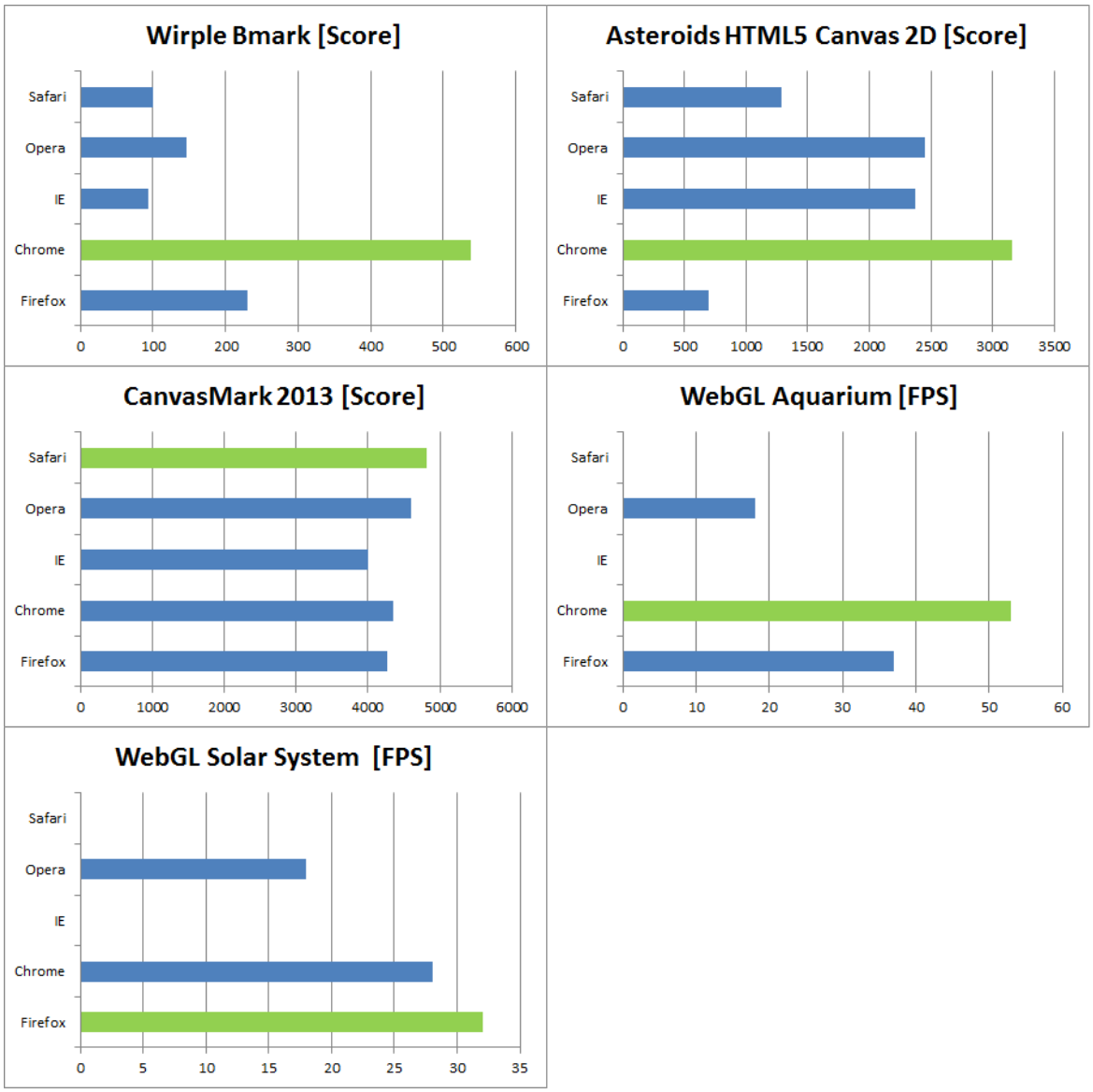
- Wirple Bmark.
- Asteroids HTML5 Canvas 2D Rendering and JavaScript Benchmark.
- CanvasMark 2013.
- WebGL Aquarium.
- WebGL Solar Systém.

a pro test HW akcelerace:

- FishIE Tank.
- Psychadellic Browsing.
- WebVizBench.

Všechny testy proběhly 3x, výsledné hodnoty jsou průměrem bodů (Score), snímků za sekundu (FPS – Frames Per Second) nebo otáček za minutu (RPM – Revolutions Per Minute). Jednotky výsledků jsou uvedeny u názvu grafu.

5.6.4 VYHODNOCENÍ VÝSLEDKŮ TESTŮ VÝKONU HTML5 A WEBGL



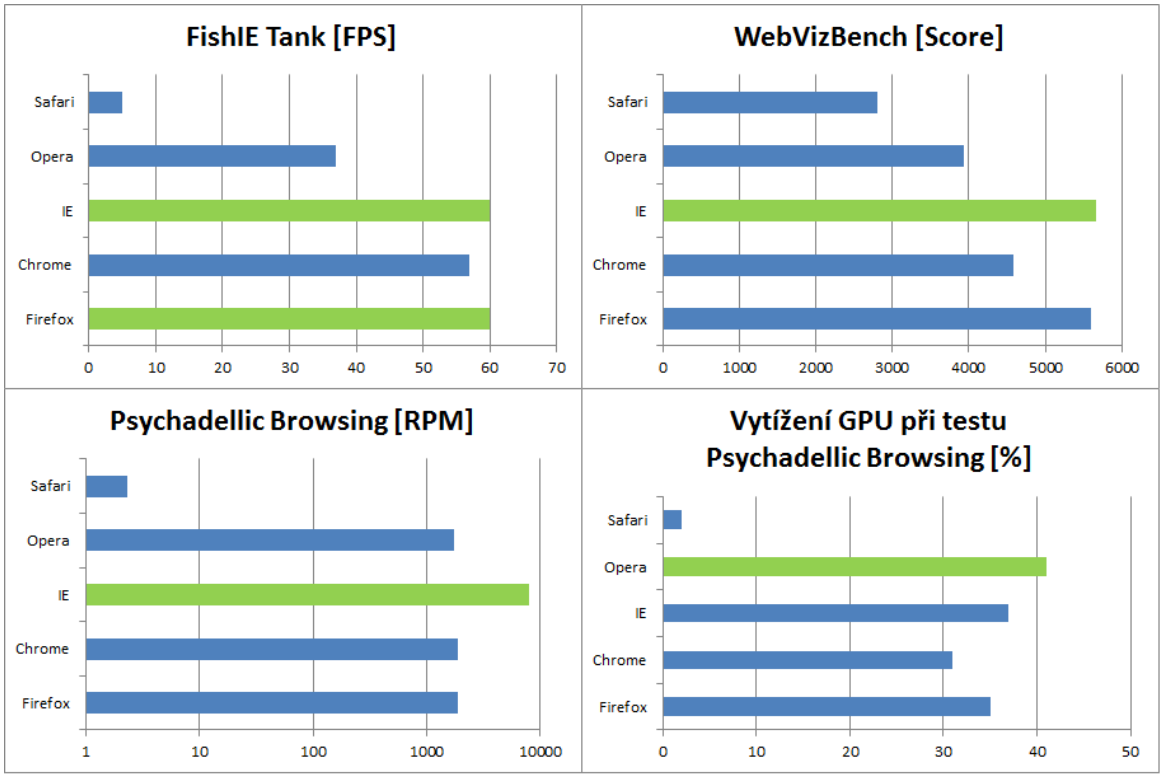
Obrázek 17: Výsledky testů výkonu HTML5 a WebGL.

Test Wirple Bmark zkoumá jako jediný z výše uvedených současně výkon HTML5 i WebGL v poměru 2 testy HTML a 2 testy WebGL. Prohlížeče Internet Explorer a Safari jsou poznamenány tím, že polovina testu neproběhla. S výraznou převahou zde zvítězil Chrome, stejně jako v testu Asteroids HTML5 Canvas 2D Rendering and JavaScript Benchmark. Zajímavé je, že v následujícím testu CanvasMark 2013, který z předchozího přímo vychází, byly výsledky velmi vyrovnané a dokonce největší výkon vykázal prohlížeč Safari.

Testování WebGL se účastnili z výše uvedených důvodů jen tři prohlížeče. Z obrázku 17 je patrné, že v obou testech WebGL poměrně výrazně zaostávala Opera. Tento prohlížeč sice podporuje HTML5 HW akceleraci, ale ne ve výchozím stavu. Možným vysvětlením toho, proč Opera nepodporuje HW akceleraci a WebGL výchoze, může být pokračující práce na odlaďování renderovacího jádra Presto společností Opera Software. V testu WebGL Aquarium zvítězil Chrome s průměrem 53 FPS, zatímco v testu WebGL Solar Systém získal body Firefox s průměrem 32 FPS.

Celkově nejlépe zvládal testy prohlížeč Chrome. Ten jako jediný ze dvou prohlížečů defaultně podporuje jak HTML5 HW akceleraci, tak WebGL. Z grafů vidíme, že vyhrál tři z pěti testů, ve zbylých dvou se umisťoval velmi vyrovnaně.

5.6.5 VYHODNOCENÍ VÝKONU HW AKCELERACE



Obrázek 18: Výsledky testů hardwarové akcelerace.

Stejně jako v testu výkonu HTML5 a WebGL, i v tomto testu výrazně zaostávalo Safari, ale i Opera. V zátěžovém testu FishIE Tank bylo nastaveno zobrazení 1000 rybek a byla sledována hodnota snímků za sekundu. Maximální hodnoty 60 FPS dosáhl jak Firefox, tak Internet Explorer. Test WebVizBench realizoval různé druhy manipulace s obrázky, změnu jejich pozice, otáčení a třídění. Zvítězil Internet Explorer s 5666 body.

Nejzajímavější však bylo testování Psychadellic Browsing. Tento test během 14 sekund roztočil a následně zpomalil barevné kolo. Výsledkem je počet otáček, které kolo vykonalo. Safari samozřejmě bez podpory HW akcelerace nemělo příliš šanci dostihnout své konkurenty a kolo otočilo jen 2x. Zarážejícím byl však propastný bodový rozdíl, mezi Internet Explorerem a trojicí Firefox, Chrome, Opera. Výsledky tohoto uvedeného tria byly velmi podobné, pohybovaly se okolo 1800 RPM, ale Internet Explorer vykazoval více než čtyřnásobný počet otáček (v průměru 8011 RPM). Protože se vizuálně nezdálo, že by byl pohyb kola v Internet Exploreru plynulejší nebo rychlejší, rozhodl jsem se ještě udělat vlastní test s využitím monitorovacího programu MSI Afterburner.

Zopakoval jsem měření testu Psychadellic Browsing se spuštěným programem Process Explorer a MSI Afterburner. Pomocí Process Explorer jsem mohl určit přesný čas startu testu, a program MSI zachytil průběh vytížení GPU během tohoto testu. Dle času startu jsem pak v textovém výstupu programu MSI přesně vyhledal začátek testu, a protože test probíhal 14 sekund, bylo určení času dokončení jednoduché. Díky těmto informacím jsem určil procentuální vytížení GPU během testu a zjistil jsem, že ještě více než Internet Explorer, umí vytížit GPU prohlížeč Opera a o jednotky procent méně než Internet Explorer vytěžoval GPU Firefox a Chrome. Nedovedu si tedy vysvětlit, jak je možné, že Internet Explorer získal čtyřnásobný počet bodů. Test Psychadellic Browsing pochází z dílny vývojářů Internet Exploreru, ale zda je právě tento fakt důvodem k bodové nadsazenosti, je bezpochyby zajímavým tématem diskuse.

5.6.6 AUDIO A VIDEO KODEKY PODPORUJÍCÍ HW AKCELERACI

Z hlediska přehrávání videa je důležité, jaké kodeky podporuje příslušný prohlížeč a také na tom, jakých kodeků využívají pro streaming jednotlivé streamovací servery. *„Když na YouTube streamujete video ve Flash Playeru, máte jistotu, že SD video bude nejspíše ve formátu FLV a HD video v MP4/H.264. Toto jsou výchozí kodeky Flash Playeru na všech platformách.*

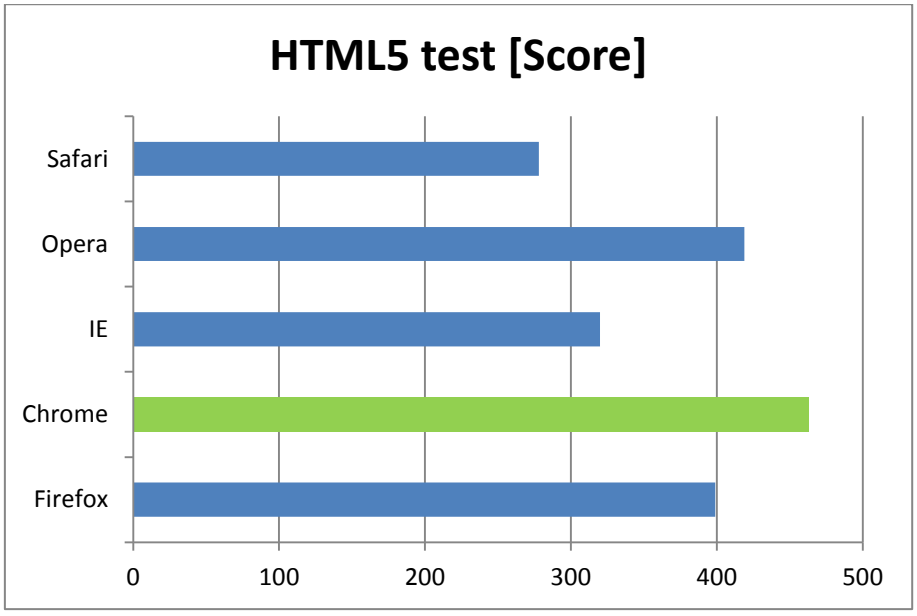
Jenže HTML5 video žádný kodekový standard nemá a záleží pouze na službě samotné, v jakém formátu pošle prohlížeči stream videa. Každý prohlížeč zároveň podporuje různé kodeky a používá různě kvalitní dekodéry videa.“ [36] Jak Flash, tak HTML používá kodeky využívající HW akceleraci. V případě Flash se jedná o kodek H.264,

u HTML5 se jedná o kontejner WebM, který obsahuje video kodek VP8/VP9 a audio kodek Vorbis. Zda jsou tyto kodeky podporovány prohlížečem lze snadno zjistit pomocí online testu *html5test.com*. O tom, v jakém formátu jsou poskytována videa, je obvykle možné naleznout informace přímo na stránkách služby, v případě *youtube.com* se jedná o stránku *youtube.com/html5*.

5.6.7 PODPORA HTML5

V tomto testování věnujme pozornost míře podpory standardu HTML5. Úkolem zde není určit výkon prohlížečů, ale výsledek nám poskytne povědomí o tom, jak nejnovější verze prohlížečů podporují tento standard.

Ke zjištění míry podpory HTML5 byl využit test *html5test.com*, který testuje například podporu audio/video formátů, podporu různých typu vstupních polí, uživatelskou interakci (metoda drag and drop, možnosti editace HTML, API), proměnné, možnosti historie a navigace, ale i zabezpečení. Maximální bodové ohodnocení je 500. Vítězem tohoto testu je Chrome se 463 body.



Obrázek 19: Výsledky HTML5 test.

5.7 TESTOVÁNÍ VYTÍŽENÍ OPERAČNÍ PAMĚTI

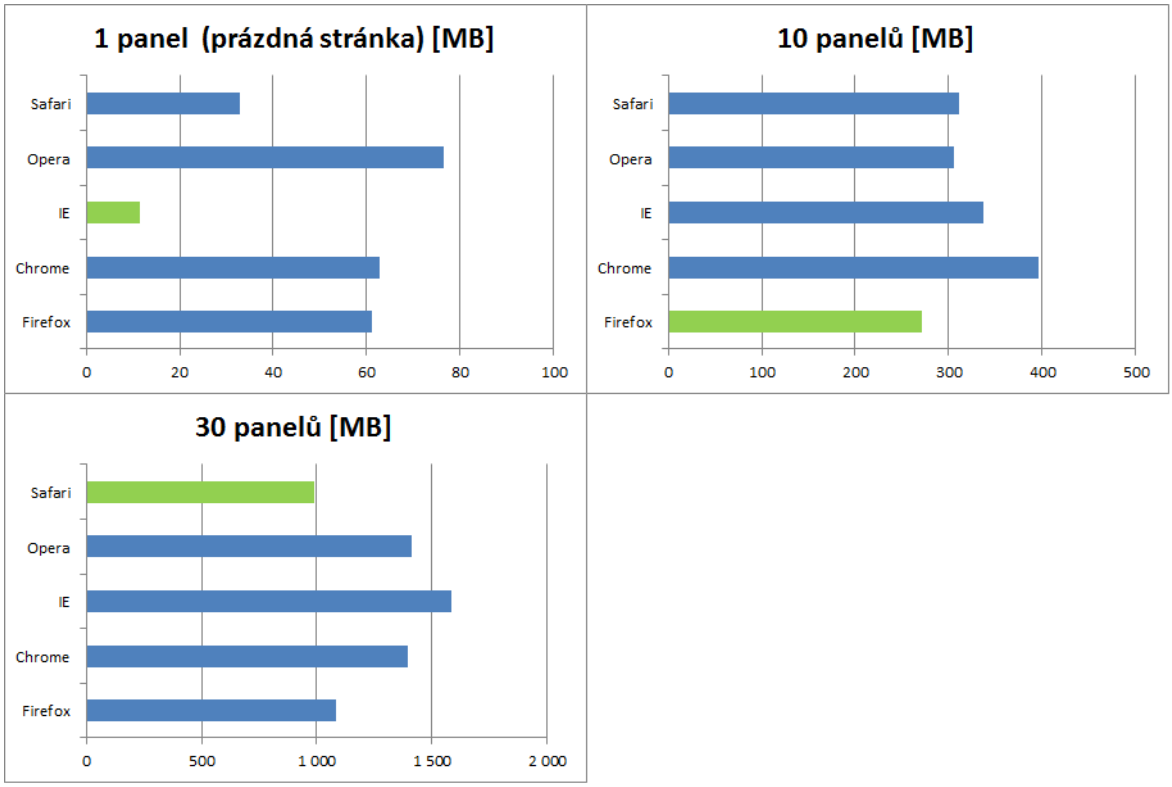
V této kapitole se zaměříme na testování vytížení operační paměti jednotlivými prohlížeči. Na základě jednotného scénáře budeme sledovat, kolik operační paměti využijí jednotlivé prohlížeče. Abychom se co nejvíce přiblížili uživatelskému chování, vytvořili

jsme 3 testy – zachycení hodnot při startu prohlížeče s prázdnou stránkou, po otevření 10 panelů a nakonec při 30 otevřených panelech s různým obsahem. V každém prohlížeči byly otevírány stejné stránky, v nichž jsme prováděli stejnou činnost. Před, a po každém testování byla vymazána jejich cache. Hodnoty jsme zachytili programem Process Explorer.

V případě běhu prohlížeče s prázdnou stránkou není třeba scénář rozvíjet. Pro test spotřeby paměti při 10 otevřených panelech jsme zvolili tyto stránky a činnosti v nich:

- *seznam.cz* – vyhledání vybraného hesla vyhledávačem.
- *youtube.com* – spuštění doporučeného videa (přehrávač Flash Player).
- *heureka.cz* – procházení kategorií Počítače a Notebooky.
- *aukro.cz* – procházení kategorie Elektro, PC a Mobily.
- *idnes.cz* – otevření hlavní zprávy dne, scrollování.
- *autosalon-tv.cz* – spuštění posledního dílu seriálu Autosalon (přehrávač Silverlight).
- *motorkari.cz* – přechod do sekce Články, otevření hlavního článku, scrollování.
- *idos.cz* – vyhledání trasy Plzeň – Hamburg autobusem.
- *autorevue.cz* – otevření hlavního článku, scrollování.
- *novinky.cz* – otevření hlavní zprávy dne, scrollování.

V případě testování 30 otevřených panelů jsem použil stejné stránky ve stejném pořadí, jen každá z výše uvedených stránek byla otevřena 3x. Otevírané články či videa v každém jednom prohlížeči však nebyly stejné – například v každém ze tří panelů se stránkou youtube.com běželo jiné video, které jsem vybral z doporučených. Stejná 3 různá videa byla použita v každém prohlížeči, aby se předešlo zvýhodnění vlivem různě dlouhých či kvalitních videí. Stejný princip byl praktikován například u redakčních serverů, kde byly otevírány trojice doporučených zpráv atd.



Obrázek 20: Výsledky testů spotřeby operační paměti.

Z výsledků (obr. 20) vidíme, že při spuštění jedné prázdné schránky s přehledem vítězí Internet Explorer s cca 11,5 MB, naopak nejvíc paměti spotřebovala Opera (76,5 MB). Rozdíly se začnou vyrovnávat při 10 otevřených panelech. Zde se do vedení dostává Firefox (271 MB), nejhůře si vede Google Chrome (396 MB). Při 30 otevřených záložkách už jsou rozdíly markantní, v řádu stovek MB. V tomto případě vítězí Safari (991 MB) a nejhůře dopadl Internet Explorer (1 583 MB).

Z hlediska spotřeby paměti bych ale nejlépe hodnotil Firefox, protože při 10 otevřených panelech spotřeboval nejméně paměti a i při 30 otevřených panelech spotřeboval o zhruba 500 MB méně, než Internet Explorer.

Otázkou však zůstává, zda vyšší náročnost na paměť znamená automaticky horší výsledek. Když uvážíme, že naše referenční sestava obsahuje 6 GB paměti RAM a i v nejhorším případě, kdy bylo v Internet Exploreru spuštěno 30 obsahově náročných stránek využívající různé zásuvné moduly bylo využito 1,5 GB, využili jsme paměť jen z jedné čtvrtiny. Dále bychom měli brát v úvahu, že každý prohlížeč hospodaří s pamětí jinak. Chrome například může paměť nečinných procesů uvolnit systému, ale zároveň má

například funkci předpřeložení názvů DNS, což se může na celkové spotřebě paměti také projevit.

6 DALŠÍ WEBOVÉ APLIKACE A JEJICH TESTOVÁNÍ

V předchozích kapitolách jsme otestovali jednotlivé funkcionality webových prohlížečů. Získali jsme tak přehled o vlastnostech jednotlivých prohlížečů a teoretický základ o fungování jejich nejdůležitějších komponent.

Díky tomu víme, jakým způsobem zatěžuje hardware počítače prohlížeč sám o sobě. Z dosavadních výsledků měření je zřejmé, že ne každý prohlížeč si umí poradit s velkou zátěží a dělit ji mezi více komponent.

6.1 VÝBĚR INTERNETOVÉHO PROHLÍŽEČE

Prohlížeče jsme podstoupili komplexnímu testování. Jejich výkon a schopnost práce s různými standardy a formáty, ale i plynulost běhu, využití HW akcelerace, to vše má vliv nejen na uživatelský komfort, ale také na to, jak prohlížeč nakládá s webovými aplikacemi.

Na výsledky testů prohlížečů lze pohlížet z různých pohledů. Často se na webu setkáváme s množstvím recenzí prohlížečů, jejichž závěry pocházejí pouze z testování rychlosti JavaScript interpretů. Tato informace je sice podstatná, nikoliv však jediná. Pokud se pohybujeme v obsahově bohatém prostředí a využíváme webové aplikace, sledujeme online videa nebo hrajeme online hry, měly by nás zajímat i jiné faktory. Mezi ty klíčové patří HW akcelerace, podpora formátů audia a videa, a pokud patříme mezi uživatele, jež mají trvale spuštěny stovky panelů v prohlížeči, tak také spotřeba operační paměti.

Na základě výše uvedeného jsem po provedeném testování a pečlivé úvaze vybral prohlížeč Google Chrome. Z testovaných prohlížečů má největší podporu HTML5, což je v dnešní době neoddiskutovatelná výhoda. Vykazuje také v průměru nejvyšší výkon JavaScriptu. V testech výkonu vykreslování 2D a 3D grafiky zvítězil ve 3 z 5 testů a v ani testech HW akcelerace neměl žádné velké výkonnostní (např. rozdíl pouhých 3 FPS mezi ním a vítěznými prohlížeči).

Google Chrome má z hlediska testování také výhodu ve své „multiprocesovosti.“ V případě, že bychom testovali aplikace využívající zásuvné moduly, můžeme díky členění procesů pozorovat, kolik systémových prostředků spotřebovaly tyto jednotlivé procesy.

Poslední výhoda hovořící ve prospěch Chrome byla zmíněna již v kapitole 4.2. Jedná se o možnosti prohlížeče zobrazovat vytížení operační paměti a obsahuje také vlastní nástroj Správce prostředků. Ani jeden z těchto nástrojů nebyl před testováním prohlížečů v plánu využívat, neboť nebylo jisté, který z prohlížečů bude nakonec použit. Proto jsem vyhledal a průběžně i vyzkoušel jiné programy, které svojí funkcionalitou plně vyhovují charakteru následujících testů.

6.2 SCÉNÁŘ TESTOVÁNÍ – SPOLEČNÁ CHARAKTERISTIKA

Pro testování vlivu zátěže internetových aplikací na výkon počítače budeme potřebovat následující programové vybavení:

- Process Explorer.
- Sledování prostředků (MS Windows).
- MSI Afterburner.
- Microsoft Office.

Před vlastním testováním web aplikací otevřeme programy Process Explorer, Sledování prostředků a MSI Afterburner. V posledním jmenovaném programu musí být povolena možnost „*Log history to file*“. Nastavíme hodnotu 1000ms pro záznam dat – každou vteřinu zachytí program příslušnou hodnotu a posune graf. Záznam po vteřinách nám také umožní vyhledat v logu na sekundu přesný čas startu procesu. Čas startu procesu je zobrazen v Process Exploreru.

Test každé webové aplikace bude trvat přesně 5 minut. Sledovat budeme následující parametry vybraných komponent:

- **CPU** – průměrné procentuální vytížení pomocí programu Sledování prostředků a spotřebovaný procesorový čas pomocí Process Exploreru. Procesorový čas vyjadřuje množství času, který procesor stráví vykonáváním instrukcí. Procesorový čas je také jistým indikátorem, jak je aplikace schopna pracovat s procesy na pozadí a do jaké míry podporuje multitasking.

- **RAM** – velikost soukromé pracovní sady využité webovou aplikací. Bude-li navíc web aplikace využívat některý plug-in, bude se hodnota velikosti spotřebované paměti mezi procesy sčítat.
- **GPU** – procentuální vytížení grafického procesoru. Není možné sledovat a zaznamenávat vytížení GPU jediným procesem (v našem případě by tento úkol byl ještě velmi ztížen separovaným spouštěním procesů prohlížečem Chrome), chyba měření vytížení GPU celým systémem bude uvedena ve vstupních podmínkách.

6.2.1 VSTUPNÍ PODMÍNKY

Před testováním každé internetové aplikace nepoběží žádný jiný program, kromě prohlížeče a monitorovacích programů. Cache prohlížeče bude vymazána před testováním každé web aplikace.

Aby nebyl ovlivňován výkon ostatními programy, například překreslováním grafu programu Sledování prostředků nebo MSI Afterburner, poběží tyto aplikace na pozadí.

6.2.2 ZAHÁJENÍ A UKONČENÍ TESTOVÁNÍ

Když všechny potřebné monitorovací programy běží, spustíme prohlížeč a přejdeme na příslušnou stránku. Po průběhu připraveným 5min. scénářem přepneme co nejrychleji na okna programů Process Explorer a Sledování prostředků a pro uchování co nejčerstvějších výsledků uděláme jejich printscreen. Je dobré před zahájením vlastního testování rozvrhnout rozmístění oken tak, aby se rušivě nepřekrývala a všechny hodnoty mohly být zaznamenány jedním stiskem klávesy PrtScr. Prodleva mezi přepínáním okna prohlížeče a okny monitorovacích programů může vést ke zkreslení výsledků, kdy spotřeba CPU námi sledovaným procesem nepatrně poklesne.

6.2.3 VYHODNOCENÍ VÝSLEDKŮ

Hodnoty procesorového času, procentuálního vytížení procesoru i spotřeby operační paměti jsou zachyceny snímkem obrazovky. Veškerá data jsou vidět ze snímku a stačí je pouze převést do tabulkového procesoru Microsoft Excel. Analýza výstupního logu programu MSI Afterburner je nepatrně složitější. Data jsou programem uložena v textovém souboru s příponou *.HML*. Soubor otevřeme pomocí programu Microsoft Word. Data jsou zaznamenána v řádcích v pořadí datum, čas, číselná hodnota

sledovaného parametru (v našem případě % GPU). Abychom mohli z dokumentu získat pouze hodnoty % GPU a pracovat s nimi samostatně, označíme si celý obsah dokumentu a zvolíme horní záložku Vložení, Tabulka, Převést text na tabulku. Vybereme počet sloupců, do kterých chceme dokument rozdělit, a vznikne tabulka, jejíž sloupec s hodnotami % GPU snadno zpracujeme.

Tento postup nám ušetří velké množství práce, neboť umožňuje hromadné zpracování 300 hodnot najednou. Navíc program MSI Afterburner běží několik sekund před startem prohlížeče, takže výsledek můžeme zpřesnit pomocí programu Process Explorer, díky kterému lze určit start procesu a vyhledat příslušný čas v tabulce.

6.2.4 PŘEDPOKLÁDANÉ OMEZENÍ

Zachytávání hodnot funguje částečně ručně. Je nutné přepínat okna programů, což může vést k drobným nepřesnostem.

Vlastní testování více programů probíhá podle jednotného scénáře. Protože je každý program jiný po stránce uživatelského rozhraní i vlastního obsahu, je tato simulace ovlivněna především tím, že se nechováme v různých programech zcela identicky. Není to ani možné. Navíc mějme na paměti, že stále testujeme aplikace tak, jak by je pravděpodobně využíval běžný uživatel.

6.3 TESTOVÁNÍ: ONLINE FOTO EDITORY

6.3.1 SCÉNÁŘ TESTOVÁNÍ

Vstupem byl obrázek o velikosti 1920x1080 pixelů ve formátu JPEG. S ním jsem postupně v každém programu prováděl následující operace:

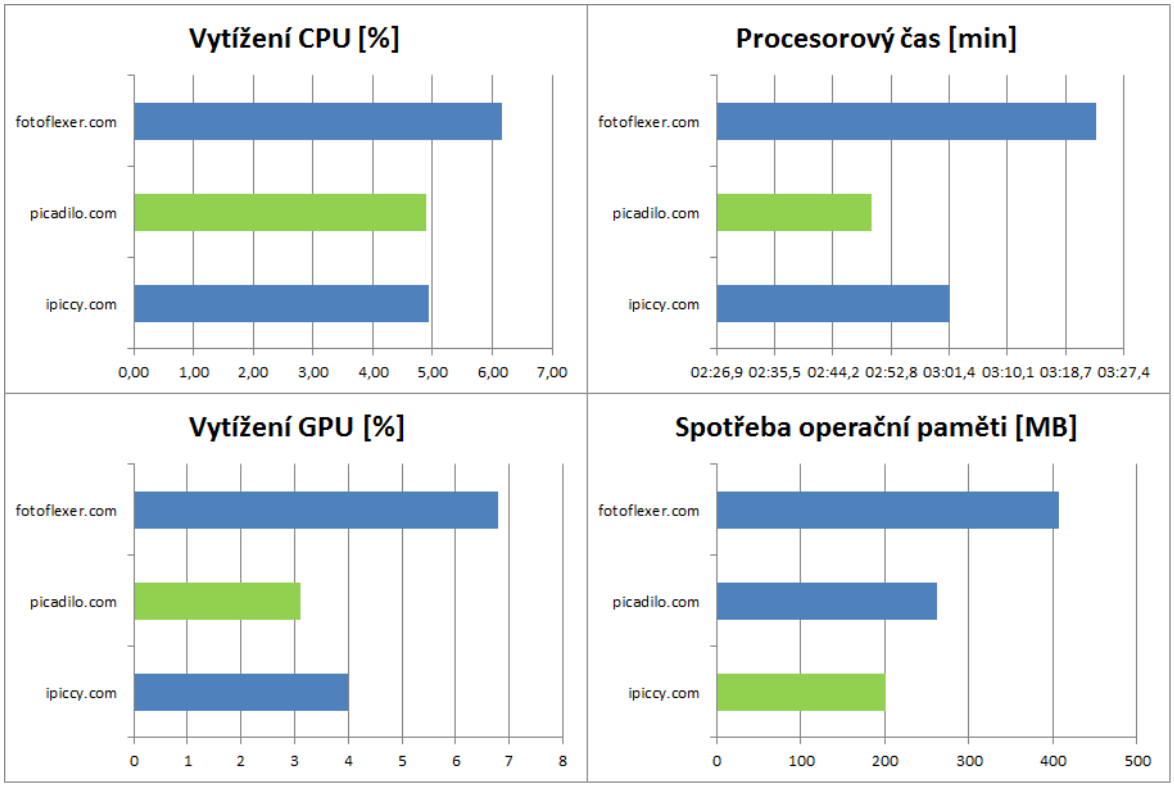
- Otáčení obrázku.
- Vertikální a horizontální převrácení.
- Přiblížení a oddálení.
- Úprava kontrastu a jasu.
- Úprava odstínu a sytosti.
- Vložení textového pole, změna fontu a velikosti.
- Průchod všemi přednastavenými efekty.

- Použití dvou filtrů.
- Oříznutí obrázku na polovinu.
- Uložení v JPEG.

6.3.2 VYBRANÉ ONLINE FOTO EDITORY

Pro testování jsem vybral aplikace, které jsou si rozhraním velmi podobné, navíc všechny využívají Adobe Flash. Všechny vybrané aplikace obsahují nástroje, jejž umožňují projít předem stanovený scénář. Těmito aplikacemi jsou:

- iPiccy (ipiccy.com).
- Picadilo (picadilo.com).
- FotoFlexer (fotoflexer.com).



Obrázek 21: Vytížení počítače online foto editory.

Z grafů (obr. 21) je patrné, že nejméně procesor a GPU zatěžovala aplikace Picadilo. Nejméně operační paměti však spotřebovala aplikace iPiccy a zcela nejvíce prostředků využil program FotoFlexer.

6.4 TESTOVÁNÍ: ONLINE VIDEO EDITORY

6.4.1 SCÉNÁŘ TESTOVÁNÍ

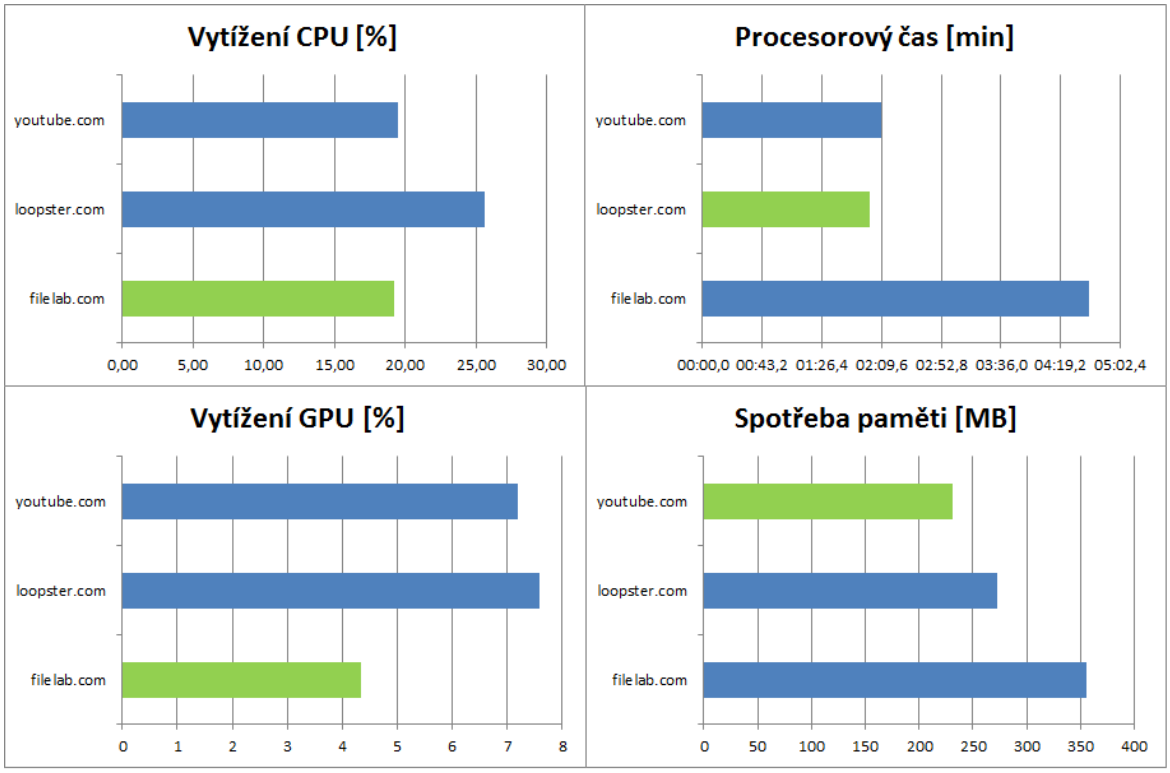
Vstupem bylo 30 sekund dlouhé video s velikostí 1280x720 pixelů ve formátu WMV. V každé z testovaných aplikací jsem prováděl následující úkony:

- Úprava vloženého videa – jas a kontrast.
- Použití filtrů.
- Přidávání a odebírání nabízených efektů.
- Přidání textu do videa.
- Průběžné přehrávání po každé úpravě.

6.4.2 VYBRANÉ ONLINE VIDEO EDITORY

Pro testování jsem opět vybral nabízenými funkcemi co nejpodobnější editory a všechny využívají zásuvného modulu Adobe Flash. FileLab navíc používá ještě vlastní plug-in, což zvyšuje počet spuštěných procesů. Testované video editory jsou:

- YouTube editor (youtube.com/editor).
- Loopster (loopster.com).
- FileLab (filelab.com).



Obrázek 22: Vytížení počítače online video editory.

Nejmenší procentuální vytížení procesoru i GPU vykazovala aplikace FileLab, zatímco nejmenší spotřebu procesorového času měla aplikace Loopster. Editor YouTube spotřeboval z této skupiny aplikací nejméně operační paměti.

6.5 TESTOVÁNÍ: ONLINE GRAFICKÉ EDITORY

6.5.1 SCÉNÁŘ TESTOVÁNÍ

Jak bude vysvětleno v následující kapitole, byli jsme při vytváření scénáře omezeni nabízenými funkcemi programů. Nakonec byl však stanoven rámec funkcí, které jsme při testování používali. Při testování jsme prováděli následující operace:

- Kreslení tvarů kruh, obdélník.
- Kreslení různými stopami využitím nástrojů Tužka, Štětec.
- Změna vlastností stop (velikost, průhlednost).
- Vyplňování barvou celku pomocí nástroje Plechovka barvy.
- Vyplňování barevným přechodem.
- Mazání pomocí nástroje Guma.

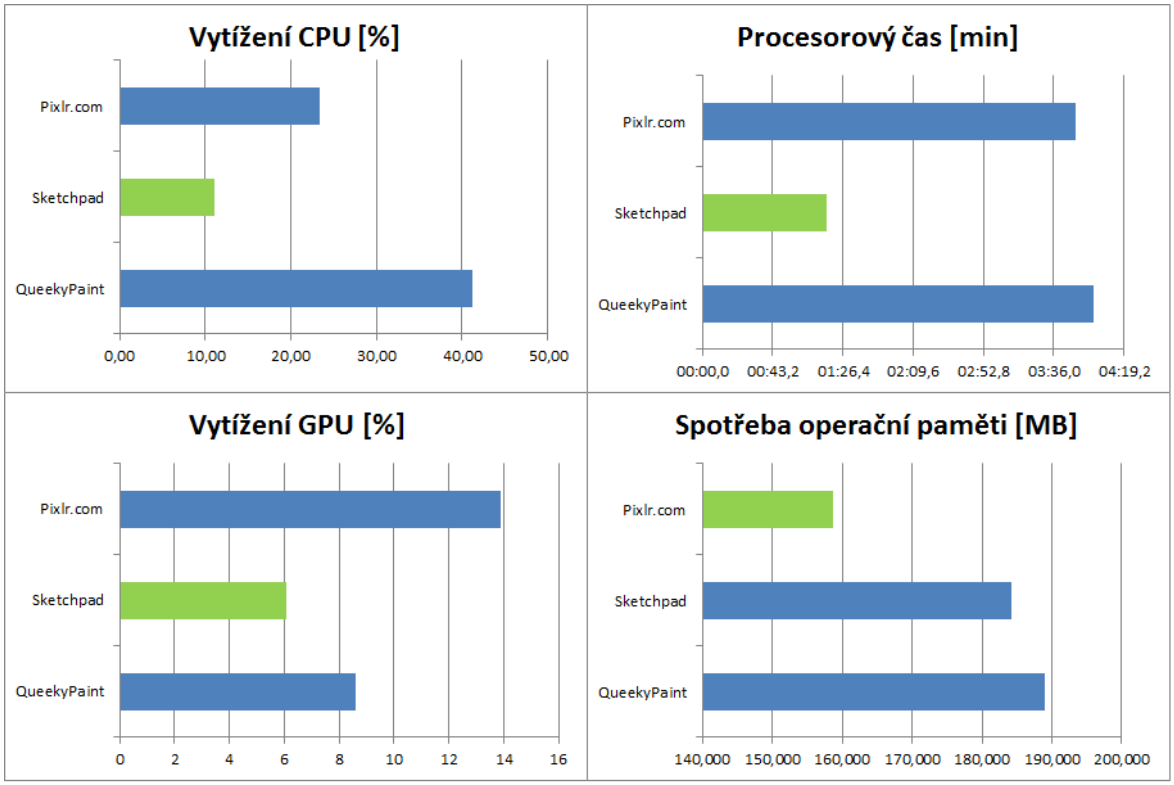
- Výběr barvy v obrázku pomocí nástroje Kapátko.
- Vložení textu, změna velikosti písma.
- Výběr částí obrázku a v tomto výběru vykonávání výše jmenovaných operací.

6.5.2 VYBRANÉ ONLINE GRAFICKÉ EDITORY

Z množství online grafických editorů jsem vybral 3, které obsahovali námi testované funkce.

- QueekyPaint (queeky.com).
- Sketchpad (mudcu.be/sketchpad/).
- Pixlr (pixlr.com).

Jak aplikace Pixlr, tak QueekyPaint využívají technologie Adobe Flash, zatímco Sketchpad je editor napsaný v jazyce HTML5. Díky tomu, že nevyužívá žádný plug-in, běží o jeden proces méně. V rámci objektivity je nutné dodat, že Sketchpad je z testovaných aplikací této skupiny nejjednodušší a nabízí nejmenší škálu funkcí. Také jako jediný z této skupiny neumožňuje otevírat nebo importovat jiné obrázky, proto jsme při tvorbě scénáře byli omezeni pouze na vytváření grafiky.



Obrázek 23: Vytížení počítače online grafickými editory.

Z naměřených hodnot reprezentovanými výše uvedenými grafy (obr. 23) vyplývá, že nejméně procesor i GPU vytěžovala aplikace Sketchpad. Nejméně operační paměti spotřebovala aplikace Pixlr, což je s ohledem na množství nástrojů této aplikace v porovnání s jednoduchým prostředím aplikace Sketchpad pozoruhodné.

7 ZÁVĚR

Internetové aplikace nabízejí velké množství funkcí usnadňující život uživatelům, kteří preferují online dostupnost aplikací před jejich lokální instalací. V dnešní době je možné pomocí internetových aplikací upravovat fotografie a video, tvořit grafiku nebo tvořit textové dokumenty, tabulky či prezentace. Z mého pohledu se během několika posledních let tyto aplikace výrazně vyvinuly. Upravovat fotografie je dnes možné online a navíc zdarma, bez nutnosti instalovat složité a často drahé programy, aniž by běžný uživatel postrádal zásadní funkce těchto programů. Tento fakt spolu se současným pokrytím internetových sítí umožňuje uživatelům přistupovat k internetovým aplikacím takřka odkudkoliv a s pomocí různých zařízení.

V této práci jsme se zabývali otázkou vytížení počítačových komponent internetovými aplikacemi. Pro čtenáře této práce by mohly být zajímavé některé vyplývající závěry. Například uživatel zvyklý na otevírání mnoha panelů v okně prohlížeče na počítačích s malou operační pamětí by mohl zvážit použití prohlížeče Safari, který v testu 30 otevřených panelů vykazoval největší úsporu paměti, dokonce o třetinu menší, než prohlížeč Internet Explorer. Dále jsme při sledování vytížení GPU jednotlivými prohlížeči ověřili, že prohlížeče podporující hardwarovou akceleraci, umí vytížit GPU na desítky procent. To se výsledně projeví nejen na zátěži procesoru, ale z uživatelského hlediska také na rychlosti vykreslování stránek, plynulosti při sledování videí ve vysokém rozlišení, při hraní online her apod. Používáme-li graficky náročné web aplikace, rozhodně bychom se měli zaměřit na prohlížeče podporující hardwarovou akceleraci. Zde se jako zcela nevhodný jeví prohlížeč Safari. Pokud k využívání hardwarové akcelerace přidáme ještě podporu 3D grafiky WebGL a standardu HTML5, je nejlepší volbou prohlížeč Google Chrome. Z testovaných skupin webových aplikací bychom s ohledem na vytížení CPU a GPU měli používat následující trio v pořadí foto editor/video editor/grafický editor programy Picadilo, FileLab a Sketchpad. Nejméně operační paměti pak spotřebovávají ve stejném pořadí aplikace iPiccy, YouTube editor a Pixlr.

8 SEZNAM OBRÁZKŮ

9 SEZNAM TABULEK

10 RESUMÉ

Web applications offer many features to make life easier for users who prefer the online availability of applications before local installation. Nowadays it is possible to use Internet applications to edit photos and video, create graphics or create text documents, spreadsheets and presentations. From my perspective, over the last few years, these applications have evolved significantly. Edit photos is now possible online, plus free, without having to install complicated and often expensive programs without a common user lacked the essential function of these programs. This fact, together with the current coverage of Internet networks allows users to access Internet applications and almost anywhere using different devices.

In this work we deal with the issue of workload of computer components by Internet applications. For example, the user used to open many panels in the browser on computers with low memory could consider using Safari, which opened 30 test panels showed the greatest memory savings, even one-third smaller than the browser Internet Explorer. During monitoring of GPU workload by browsers that supports hardware acceleration, we verified that GPU can be workload to tens percents. To the resulting impact not only on the CPU load, but from the user's point of view also the speed of rendering pages, fluency when watching HD videos, playing online games, etc. If you use graphics-intensive web applications, surely we should focus on browsers that support hardware acceleration. Here, as seems entirely inappropriate Safari browser. If you use the hardware acceleration add even more support for 3D graphics, WebGL and HTML5 is the best choice browser Google Chrome. The test groups of web applications we with respect to workload of CPU and GPU should use the following trio in order photo editor / video editor / graphic editor programs Picadilo, FileLab and Sketchpad. At least then consume memory in the same order by iPiccy, YouTube editor and Pixlr.

11 SEZNAM LITERATURY A ZDROJŮ

[1] Webová aplikace - podrobnosti. In: ŠLECHTA, Jan. *Janslechta.net: webdesign, grafika, reklama* [online]. 2007 [cit. 2013-02-19]. Dostupné z: http://www.janslechta.net/?p=productsMore&iProduct=3&sName=Webov%E1-aplikace

[2] O webových aplikacích. Chrome [online]. 2013 [cit. 2013-06-28]. Dostupné z: https://support.google.com/chrome/answer/1050586?hl=cs

[3] Výhody webových aplikací. Vladimír Macek [online]. [cit. 2013-06-28]. Dostupné z: http://macek.sandbox.cz/weby-sluzby/

[4] *Elektrotechnická měření*. 1. vyd. Praha: BEN - technická literatura, 2002, 255 s. ISBN 80-730-0022-9.

[5] MINASI, M. *IBM PC velký průvodce hardwarem: učebnice pro pokročilé*. Vyd. 1. Praha: Grada, 1994, 712 s. ISBN 80-716-9038-4.

[6] Jak funguje počítač - díl 1. - procesor a paměť. Levná pc [online]. 2006 [cit. 2013-06-28]. Dostupné z: http://www.levnapc.cz/jak-funguje-pocitac-procesor-pamet.html

[7] Adobe Secure Software Engineering Team (ASSET) Blog. UHLEY, Peleus. Adobe [online]. 2012 [cit. 2013-06-28]. Dostupné z: http://blogs.adobe.com/asset/2012/06/inside-flash-player-protected-mode-for-firefox.html

[8] How Browsers Work: Behind the scenes of modern web browsers. GARSIEL, Tali a Paul IRISH. HTML5 Rocks Tutorials [online]. 2011 [cit. 2013-06-28]. Dostupné z: http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

[9] MICHALÍK, Petr. *Technika počítačů 1: učebnice pro pokročilé*. 4., upr. vyd. Plzeň: Západočeská univerzita, Pedagogická fakulta, 2002, 97 s. ISBN 80-708-2883-8.

[10] Co je to Thread (vlákno)?. LINHART, Ondřej. VB Net [online]. 2008 [cit. 2013-06-29]. Dostupné z: http://www.vbnet.cz/clanek--96-vicevlaknove_aplikace_dil_1_uvod.aspx

[11] GOVE, Darryl. *Programování aplikací pro vícejádrové procesory: učebnice pro pokročilé*. Vyd. 1. Brno: Computer Press, 2011, 416 s. ISBN 978-80-251-3487-0.

[12] Samsung teams up with Mozilla to build browser engine for multicore machines. BRIGHT, Peter. ARS technica [online]. 2013 [cit. 2013-06-29]. Dostupné z: http://arstechnica.com/information-technology/2013/04/samsung-teams-up-with-mozilla-to-build-browser-engine-for-multicore-machines/

[13] PAVLÍK, Michal. Hardware pc: Operační paměť. [online]. [cit. 2013-06-28]. Dostupné z: http://lide.uhk.cz/fim/student/pavlimi3/htm/operacni_pamet.htm

[14] Memory Usage Backgrounder. The Chromium Projects [online]. [cit. 2013-06-28]. Dostupné z: http://www.chromium.org/memory-usage-backgrounder

[15] Význam sloupců udávajících velikost paměti v programu Správce úloh. Windows [online]. 2013 [cit. 2013-06-29]. Dostupné z: http://windows.microsoft.com/cs-cz/windows-vista/what-do-the-task-manager-memory-columns-mean

[16] Co je to hardwarová akcelerace?. ZACHAR, Martin. Stahuj.cz Magazín [online]. 2010 [cit. 2013-06-29]. Dostupné z: http://magazin.stahuj.centrum.cz/co-je-to-hardwarova-akcelerace/

[17] HORÁK, Jaroslav. *Hardware: učebnice pro pokročilé*. 4. aktualiz. vyd. Brno: Computer Press, 2007, 360 s. ISBN 978-80-251-1741-5.

[18] ŠNOREK, Miroslav. *Připojování periferií k PC: učebnice pro pokročilé*. 1. vyd. Praha: Grada Publishing, 1996, 303 s. ISBN 80-716-9146-1.

[19] Slovníček pojmů a zkratek: Webový prohlížeč. In: Wizards [online]. 2009 [cit. 2013-06-28]. Dostupné z: http://www.wizards.cz/slovnicek-pojmu/#par_75

[20] BEDNÁŘ, Vojtěch. *Alternativní webové prohlížeče: Firefox, Opera, Mozilla, Maxthon a další*. Vyd. 1. Brno: Computer Press, 2006, 168 s. ISBN 80-251-0566-0.

[21] Document Object Model (DOM). HÉGARET, Philippe. W3C [online]. 2005 [cit. 2013-06-29]. Dostupné z: www.w3.org/DOM

[22] Do hlubin implementací JavaScriptu: 1. díl. MAJDA, David. Zdroják [online]. 2008 [cit. 2013-06-28]. Dostupné z: http://www.zdrojak.cz/clanky/do-hlubin-implementaci-javascriptu-1-dil-uvod/

[23] Do hlubin implementací JavaScriptu: 3. díl – výkonnostně nepříjemné konstrukce. MAJDA, David. Zdroják [online]. 2008 [cit. 2013-06-28]. Dostupné z: http://www.zdrojak.cz/clanky/javascript-vykonnostne-neprijemne-konstrukce/

[24] SunSpider 1.0 JavaScript Benchmark. Webkit [online]. [cit. 2013-06-29]. Dostupné z: http://www.webkit.org/perf/sunspider/sunspider.html

[25] Adobe Flash Player 11.7.700.224. Adobe [online]. 2013 [cit. 2013-06-28]. Dostupné z: http://get.adobe.com/cz/flashplayer/

[26] Adobe Flash Player 11. Adobe [online]. 2013 [cit. 2013-06-28]. Dostupné z: http://www.adobe.com/cz/products/flashplayer.html

[27] Objektově orientované programování v jazyce ActionScript: Pokročilá témata. Adobe [online]. [cit. 2013-06-28]. Dostupné z: http://help.adobe.com/cs_CZ/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7f3f.html

[28] Adobe AIR and Adobe Flash Player Team Blog: News, Commentary and Insights from the Flash Player and AIR Product and Engineering Team. Adobe [online]. 2012 [cit. 2013-06-28]. Dostupné z: http://blogs.adobe.com/flashplayer/2012/02/adobe-and-google-partnering-for-flash-player-on-linux.html

[29] Microsoft Silverlight. Microsoft [online]. 2013 [cit. 2013-06-29]. Dostupné z: http://www.microsoft.com/cze/web/silverlight/

[30] Microsoft releases Silverlight 5. FOLEY, Mary Jo. ZD Net [online]. 2011 [cit. 2013-06-29]. Dostupné z: http://www.zdnet.com/blog/microsoft/microsoft-releases-silverlight-5/11391

[31] ŠINDLER, M. *Srovnání Silverlight vs. Flash*. Liberec, 2008. Dostupné z: http://sindler.name/docs/rocnikovy_projekt_sindler_final_version.pdf

[32] Flash vs. Silverlight: What Suits Your Needs Best?. ALAM, Muhammad Usama. Smashing magazine [online]. 2009 [cit. 2013-06-29]. Dostupné z: http://www.smashingmagazine.com/2009/05/09/flash-vs-silverlight-what-suits-your-needs-best/

[33] Silverlight vs HTML5 – bitva, která se nekoná. K, Dalibor. MSDN Blogy [online]. 2010 [cit. 2013-06-29]. Dostupné z: http://blogs.msdn.com/b/kaci/archive/2010/11/10/silverlight-vs-html5-bitva-ktera-se-nekona.aspx

[34] MALÝ, Martin. SVG, nebo Canvas? Vyberte si. [online]. 2010 [cit. 2013-06-30]. Dostupné z: http://www.zdrojak.cz/clanky/svg-nebo-canvas-vyberte-si

[35] OpenGL ES 2.0 for the Web. Khronos Group [online]. [cit. 2013-06-29]. Dostupné z: http://www.khronos.org/webgl/

[36] Test: Flash není horší než HTML5. Ale ani lepší. ČÍŽEK, Jakub. Zive.cz [online]. 2012 [cit. 2013-06-29]. Dostupné z: http://www.zive.cz/clanky/test-flash-neni-horsi-nez-html5-ale-ani-lepsi/sc-3-a-165091/default.aspx

12 SEZNAM PŘÍLOH

Operační systémy, květen 2012 až květen 2013	
Operační systém	**Procentuální zastoupení**
Win7	53,22
WinXP	30,19
WinVista	8,29
MacOSX	2,78
Win8	1,96
Linux	1,66
iOS	0,97
Android	0,43
Win2003	0,17
Unknown	0,14
Win2000	0,11
Playstation	0,04
SunOS	0,02
Win98	0,01
WinNT	0,01

Příloha 1: Tabulka oblíbenosti operačních systémů (zdroj: statcounter.com)

Oblíbenost prohlížečů v ČR, květen 2012 až květen 2013	
Prohlížeč	**Procentuální zastoupení**
Chrome	33,58
Firefox	32,73
IE	23,7
Opera	5,92
Safari	2,76
Maxthon	0,38
Android	0,29
Chromium	0,16
Phantom	0,12
SeaMonkey	0,1
Sony PS3	0,04
Iron	0,03
Konqueror	0,02
Comodo Dragon	0,02
Pale Moon	0,02
RockMelt	0,02
360 Safe Browser	0,02
Mozilla	0,01
Netscape	0,01
AppleWebKit	0,01
Lunascape	0,01
Yandex Browser	0,01

Příloha 2: Tabulka oblíbenosti internetových prohlížečů (zdroj: statcounter.com)

	Primetest			Focustest			MD5 encoding			Random Key genereator			Run-length encoding		
	FL	SL	JS	FL	SL	JS	FL	SL	JS	FL	SL	JS	FL	SL	JS
Firefox	V		V			V		V	V	V	V	V	V		
Chrome															V
IE		V						V							
Opera				V	V		V							V	
Safari															

Příloha 3: Souhrn jednotlivých výsledků testování výkonu zásuvných modulů.

Vysvětlivky k příloze 3:

FL .....technologie Flash.

SL .....technologie Silverlight.

JS.....technologie JavaScript.

V.....vítěz příslušného testu s využitím dané technologie.