BENCHMARKOVÁNÍ

Tento materiál slouží jako doplněk k přednáškám předmětu VSP o testování výkonnosti HW a SW.

ÚVOD

„Benchmarkovat“ v počítačovém světě znamená spouštět program nebo sadu programů s cílem určit výkon počítačového systému. Slovo benchmark je používáno jak pro označení programu použitého pro testování výkonu, tak (méně často) pro označení naměřeného hodnocení. Jako benchmark může sloužit speciálně připravený program, nebo libovolně zvolená aplikace, obvykle ta, která má být na testovaném systému nasazena.

Obvykle se mluví zejména o benchmarkování hardware, ale přesnější je mluvit o benchmarkování systému – software a hardware je od sebe často neoddělitelný. I v případě použití specializovaných programů testujících nějakou HW komponentu je zároveň spuštěn i operační systém a ovladače, což může výsledky testu ovlivňovat. Stejně tak mohou být výsledky ovlivněny například verzí překladače použitou pro překlad benchmarku.

ZÁKLADNÍ POJMY

V textu jsou používány následující pojmy:

- **Systém**: souhrn veškerého použitého HW a SW.
- **Uživatel**: Entita, která systém využívá. Nemusí jít nutně o člověka, může to být i jiný software.
- **Zátěž**: Požadavky zasílané uživatelem systému. Co je konkrétně požadavkem může být velmi variabilní, počínaje jednoduchým dotazem do databáze, přes transakce až po spouštění rozsáhlých dávkových úloh.
- **Metrika**: kritérium zvolené pro hodnocení kvality systému nebo pro porovnávání systémů.

Později budou popsány podrobnněji.

BENCHMARKOVACÍ EXPERIMENT

Benchmarkovací experiment má určit výkon testovaného sytému, případně poskytnout údaje pro porovnání několika systému. Pro tvorbu takového experimentu platí stejná pravidla jako pro jakýkoliv jiný experiment. V případě benchmarků je třeba zaměřit se zejména na:

- **Opakovatelnost**: benchmark musí být zvolen tak, aby mohl být opakován. To se týká zejména zvolené zátěže.
- **Izolace**: experiment by měl být připraven tak, aby testoval jen to, co doopravdy chceme. Pozor na procesy, které mohou nebo dokonce musí běžet zároveň s testovacím softwarem (například JVM – Java rozhodně není ideální jazyk pro psaní testů HW).
- **Měřitelný výsledek**: výsledky experimentu by mělo být možné měřit, je třeba dávat pozor na to, co vlastně měříme.

MOŽNÉ POSTUPY

Existují tři základní postupy, které je možné k benchmarkování použít – měření, vytvoření analytického modelu a simulace. Liší se použitou úrovní abstrakce a náročností na jejich vytvoření. Jako vždy platí, že všechno je „něco za něco“. Pokud je benchmark jednoduchý, je pravděpodobné, že jeho výsledky nebudou tak dobré jako u složitějšího. Vzhledem k obtížnosti izolace v případě benchmarků je rozhodně vhodné využít nejméně dva z uvedených způsobů.

Analytický model je nejlevnější co do nákladů na přípravu, ale vhodný jen v případě že skutečně dobře známe modelovaný systém a že je znám postup pro jeho efektivní modelování (např. konstrukce markovského řetězce). Není potřeba mít skutečný systém k dispozici, takže se může hodit i ve fázi návrhu. Na druhou stranu, tvorba analytického modelu nevyhnutelně znamená zjednodušování a to může ovlivnit vypovídající hodnotu výsledků. Využití modelu je obvykle nejlevnější a nejrychlejší, bohužel ale také náchylné k chybám plynoucím ze zjednodušení.

Simulace je o něco složitější, opět ji ale lze využít bez přístupu k reálnému systému. Na rozdíl od tvorby analytického modelu je návrh simulace intuitivnější, problém ale nastává při kalibraci. Simulaci je třeba nastavit tak, aby co nejvěrněji odpovídala modelovanému systému. To velmi často znamená určit (odhadem nebo měřením) doby, které může nějaký činnost trvat (typicky doby obsluhy nebo doby mezi příchody požadavků). Na kvalitě těchto nastavení závisí kvalita získaných výsledků, pokud je kalibrace provedena špatně, simulace bude k ničemu. Tvorba simulace může být ve srovnání s tvorbou analytického modelu náročnější na čas, ale pokud je systém složitý, je často snazší vytvořit simulaci než analytický model.

Měření znamená spouštění testovacích programů na skutečném HW. To znamená, že pro provedení měření je nudné mít přístup k testovanému systému. Při návrhu měření tedy využijete těžko a provedení testů je nejnákladnější a na přípravu experimentu nejnáročnější. Na druhé straně, výsledky získané měřením jsou nebližší realitě, pokud je spouštěn vhodný testovací SW. Obecně platí, že nejlepším benchmarkem je spustit program který se skutečně bude používat na HW na kterém skutečně bude běžet a sledovat jeho chování.

Nevěřte simulaci bez ověření měřením na reálném systému nebo analytickým modelem
Nevěřte analytickému modelu bez ověření simulací nebo reálným experimentem
Nevěřte simulaci bez ověření analytickým modelem nebo měřením na reálných datech

V dalším textu je popsán ideální postup pro přípravu benchmarkovacího experimentu. Stejně jako v případě vývoje SW se i na tento postup nedá dívat jako na vodopádový model, ale spíš jako na iterativní proces, ve kterém je nutné se k předchozím krokům občas vracet, přehodnotit je a podle toho upravit následující činnost.

Určení cílů

Volba síle je pro benchmarkování zcela zásadní, protože z ní vyplývá jaké prostředky a postupy budou použity. Jedním z hlavních vodítek k určení cílů je uvědomit si, pro koho se benchmark dělá a proč ho vlastně potřebuje. Uživatele obvykle nezajímá kolik MIPS má jeho procesor, spíš jestli bude na otevření okna Wordu čekat 1 nebo 5 vteřin. Stejně tak do tohoto kroku patří vymezení hranic systému. I to je velmi důležité, pro správnou přípravu zátěže.

Popis služeb

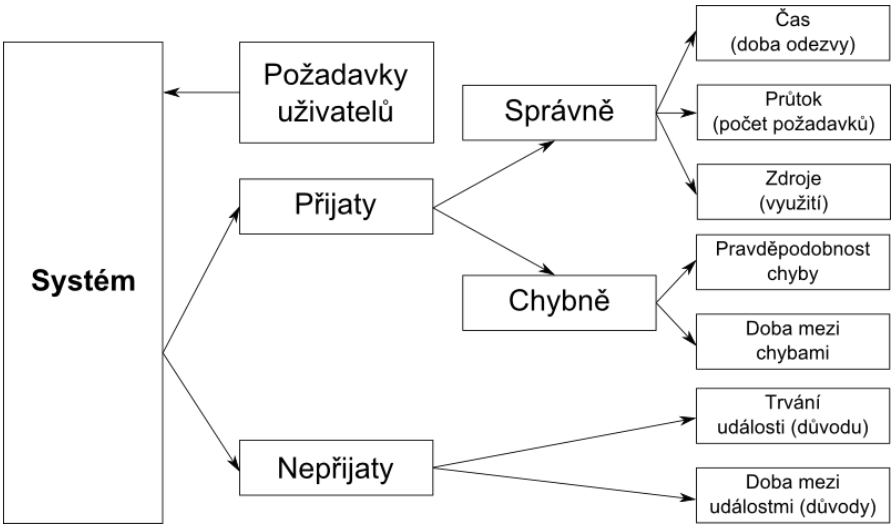
Při přípravě benchmarku je třeba mít představu o tom, které služby jsou uživatelem požadovány a jaké mohou mít výsledky. To ovlivňuje volbu metriky i testů. V seznamu by se měly objevit všechny důležité služby, ale neměl by být zahlcen podrobnostmi – čím víc služeb najednou je testováno, tím složitější je příprava testů.

Služby mohou být popsány na libovolné úrovni – může jít o dávkové zpracování výpočtu, přenesení paketu sítí nebo provedení rozsáhlé databázové transakce. Zátěž je později konstruována právě na základě popisu služeb, takže je třeba věnovat mu dostatek času. Další důležitá věc jsou možné výsledky – může se stát, že se služba neprovede vůbec? Že proběhne chybně? Nebo nás zajímá jen doba jejího provádění? I tohle ovlivňuje pozdější přípravu zátěže. Pokud hrozí, že služba nebude vůbec provedena, nestačí testovat jen dobu za jakou je prováděna, ale je nutné sledovat i jestli k jejímu provedení vůbec došlo.

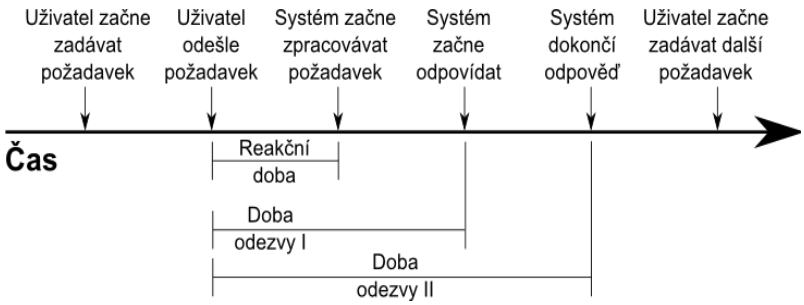
Volba metriky

Existuje řada metrik, používaných v tisku nebo samotnými výrobci HW a SW. Ne všechny ale poskytují informace, které jsou doopravdy potřeba. Zejména v tisku je patrná snaha sahat po nejsnáze dostupné metrice

jako po vhodné charakteristice systému a to ne vždy přináší dobré výsledky (například frekvence CPU není v žádném případě dobrá charakteristika výkonu počítače). Metriky bývají založené na čase (jak dlouho něco trvá udělat), na výkonu (kolik je toho možné udělat) a na spolehlivosti (s jakou pravděpodobností se něco rozbije).



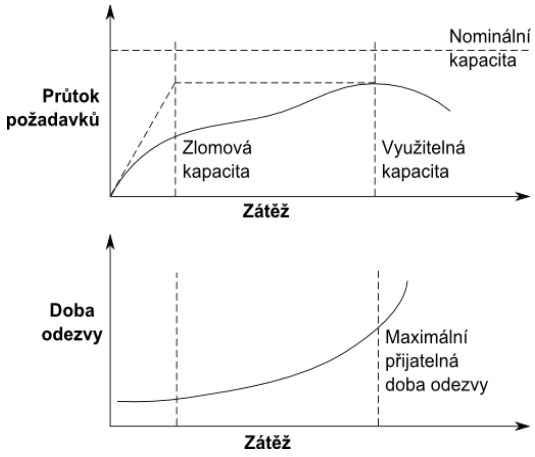
Neexistuje žádný jednoznačný návod pro volbu metriky, ale tento obrázek může trochu napovědět. Pokud systém požadavky vždy přijímá a je jisté že vrátí správnou odpověď (například DB server připojený přes TCP), pak má smysl zajímat se hlavně o doby odezvy nebo počet požadavků, které dokáže za nějakou dobu zpracovat. V případě že je možné dimenzovat HW podle počtu požadavků, lze se ptát i na efektivitu využití zdrojů (tedy jakou dobu systém tráví čekáním). Jestliže existuje riziko chybné odpovědi, je důležité se ptát na dobu mezi chybami nebo pravděpodobnost chyby. A konečně pokud existuje možnost, že systém požadavek nepřijme vůbec (např. zahlcený směrovač), má smysl ptát se jak dlouho trvá doba, kdy požadavky nepřijímá, nebo za jakých okolností je přijímat přestane.



Při použití časových metrik se typicky sleduje doba odezvy nebo reakční doba. Význam těchto pojmů je jasný z časové osy. Pozor na to že reakční doba může být definovaná, jako čas do doby kdy systém začne odpovídat nebo do doby kdy odpověď dokončí. Obojí může mít význam, ale při čtení výsledků je třeba vědět, o jakou reakční dobu jde. Doba do začátku odpovědi může být užitečná třeba pro webový server – už z prvních přijatých řádek html souboru dokáže prohlížeč vytvářet stránku, takže uživatel vidí, že se něco děje. Někdy je třeba ale přijmout všechna data než se odpověď zobrazí uživateli a pak je důležitější druhá varianta.

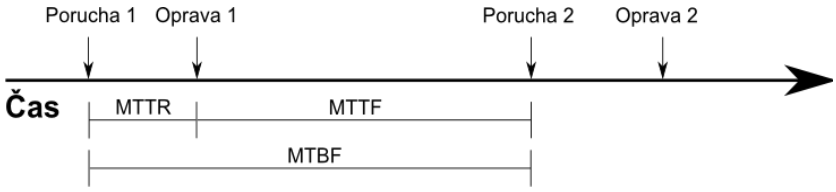
Metriky zaměřené na kapacitu se typicky měří při třech situacích. Nejčastěji uváděná je *nominální* (nebo maximální) kapacita, oblíbená zejména výrobci HW. To je maximálně počet požadavků které mohou být obslouženy při ideálních podmínkách (tedy systém je neustále vytěžován, ale tak aby žádný požadavek nebyl odmítnut a všechny se zpracovaly). Většinou ale systém takto pracovat nebude – zátěž často přichází ve vlnách, požadavky mají různé nároky na zdroje a tak je systém střídavě vytížen a uvolňován. Proto je důležitější *využitelná* kapacita. To je maximální kapacita dosažitelná se zachováním požadované doby odezvy. Například hypotetický DB server může být schopen obsloužit v ideálním případě 100 transakcí za minutu, ale doba

odpovědi se bude pohybovat mezi 1 a 10 sekundami. Pokud ale bude odesláno jen 50 transakcí, doba odezvy nestoupne přes 2 sekundy. 100 transakcí za minutu tedy bude maximální kapacita, ale 50 transakcí je využitelná kapacita – uživatel při delší době odezvy může být nervózní odesílat požadavky znovu v domnění, že se neodeslaly a tak systém přetěžovat. Poslední údaj, který se používá, je *zlomová* kapacita, tedy zátěž při které zůstává doba obsluhy velmi krátká až neznatelná. Volba času potřebného pro určení využitelné a zlomové kapacity závisí především na uživatelích, kteří budou se systémem pracovat, není nijak dopředu definována.



Další skupina metrik jsou metriky zaměřené na výkon. Sem patří především doba zpracování, zrychlení a průtok. Doba zpracování nebo doba obrátky má význam hlavně pro dávkové zpracování úloh. Jde o dobu od zadání úlohy k výpočtu k vrácení výsledku. Ačkoliv by se zdálo, že v dnešní době se s podobným přístupem moc nesetkáte, není to pravda. Dávkové zpracování je typické třeba pro překládací a kompilační algoritmy. V případě že je třeba překládat rozsáhlý projekt, dedikovaný server s překladačem je obvyklé řešení. Zrychlení se používá v souvislosti s paralelizmem a představuje poměr odezvy paralelizované a neparalelizované úlohy. A konečně průtok je počet požadavků za nějakou jednotku času. Tato metrika je velmi oblíbená a právě sem spadají např. MIPS, FLOPS, transakce za vteřinu a podobně.

Benchmarky jsou často popisovány z hlediska výrobců HW – tedy čím víc MIPS nebo čím kratší doby odezvy tím lépe. Ale z pohledu zákazníků je zároveň důležité neutrácet víc než je nezbytně nutné. Výkonný HW je dražší, levný HW zase neposkytne dostatečný výkon. Proto se mluví o efektivitě a využitelnosti. Efektivita je poměr využitelné a nominální kapacity (případně poměr výkonu na n procesorech k výkonu téže úlohy na jednom procesoru). Ukazuje kolik výkonu HW je skutečně využito pro spouštěné úlohy. Využitelnost je podíl času kdy jsou zdroje systému využity ke zpracování úlohy k době čekání, ukazuje tedy, jestli systém netráví příliš doby zastaven.



Poslední skupinou metrik jsou spolehlivostní metriky, zejména střední doba mezi chybami, střední doba do selhání a střední doba do opravy. Zejména střední doba do opravy bývá často podceněna, ale u systémů které mají poskytovat např. bankovní služby je naprosto zásadní. V případě že dojde k selhání, je důležité, aby bylo možné uvést systém do použitelného stavu co nejdříve.

PARAMETRY A FAKTORY

Určení parametrů je základní pro tvorbu analytického modelu a simulace a zároveň pomáhá při přípravě zátěže pro měření. Parametry systému charakterizují systém, nemění se během jeho práce a nemění se ani v různých

instalacích systému (když stejný HW a SW používají různí lidé). Parametrem je například počet požadavků, které dokáže směrovač přeposlat za jednotku času.

Kromě parametrů systému se dá mluvit i o parametrech zátěže. Ty charakterizují požadavky (a tedy chování) uživatelů systému. V různých instalacích se liší, podle toho jak je systém využíván.

Parametrů může být celá řada, proto se z nich vybírají ty nejdůležitější a označují se jako faktory. Ideální by bylo pro každý test mít vybraný jeden faktor a s ním pracovat. Bohužel se ale často musíme potýkat s tím, že faktorů je několik. Při testování se snažíme omezit vnější vlivy (např. odpojit nepotřebná zařízení, zabránit běhu nepotřebných procesů atd.), ale ne všechny se dají odstranit. Faktor je typicky parametr s velkým dopadem na výkon. Během benchmarkování se faktor mění a sledujeme, jak se při jeho změně systém chová. Benchmark je třeba provádět pro různé hodnoty, tzv. hladiny faktoru. Hladiny by měly postihnout důležité hraniční hodnoty – typicky systém s malou zátěží, systém s typickou zátěží a přetížený systém.

Příprava zátěže

Dobře zvolená zátěž je zásadní pro to aby měl benchmark vypovídající hodnotu. Zátěž musí co nejlépe napodobovat chování reálného systému. Je důležité, aby byla opakovatelně použitelná, takže v případě porovnání různých systémů bude možné použít tutéž zátěž.

Druh zátěže závisí na použité technice benchmarkování. Pro analytický model je zátěž popisována způsobem, který se dá matematicky dál zpracovávat, tedy obvykle pravděpodobnostním rozdělením a jeho charakteristikami. Pro simulace se navíc používá trasování požadavků uživatelů, které lze následně vpouštět do simulovaného systému. Trasované požadavky mohou být uloženy buď v podobě logu z nějakého reálného systému, nebo mohou být opět popsány pravděpodobnostními údaji. V případě testů nad reálným systémem se využívají prostředky napodobující chování skutečných uživatelů, velmi často skripty, logy uložených požadavků, vzorové problémy, nebo pokud to jinak není možné, testeři.

Zátěž se dá rozdělit do dvou skupin – reálná a syntetická. Reálná zátěž je zachycení běhu skutečného systému. K tomu je ale třeba mít jednak systém s uživateli (není tedy příliš použitelná při návrhu něčeho nového), jednak prostředky k jejímu zachycení a popisu. Druhou možností je syntetická zátěž. Tedy snaha napodobit reálnou zátěž použitím nějaké omezené množiny úloh. Syntetickou zátěž lze snáze řídit a měnit, je snadno modifikovatelná pro různé systémy, ale nese s sebou jedno riziko. V případě že se příliš odchýlí od reálné zátěže, znehodnotí výsledky benchmarku.

Typické jednoduché zátěže jsou instrukční mixy odvozené od skutečných programů. Statistickou analýzou programu lze zjistit, jaké instrukce program nejčastěji provádí a vytvořit program, který bude mít odpovídají statistické zastoupení instrukcí pro skoky, aritmetiku nebo podmínky. Takovou zátěží je například benchmark Dhrystone nebo Wheetstone. Další možností je využití jader skutečných programů. Obě zmíněné možnosti trpí několika neduhy. Především nevyužívají I/O operace a navíc se obvykle vejdou do cache paměti procesoru. Odpovídají tedy chování jednoduchých vědeckých výpočtů, ale nepřipomínají reálnou zátěž například na kancelářském PC.

Proto se objevily aplikační benchmarky, zastoupené např. systémem debit/kredit. Tyto benchmarky napodobují skutečné chování transakčních aplikací (např. debit/kredit napodobuje banku vydávající peníze v síti bankomatů a transakce s touto činností spojené). Ty se hodí zejména pro rozsáhlé databázové aplikace. Z podobné myšlenky vychází i benchmarky pro kancelářské a domácí PC, postavené nad ořezanou verzí aplikací jako je MS Word nebo programy z balíku Adobe Creative Suite.

Analýza výsledků

Analýza výsledků představuje podobný problém jako popis zátěže systému – opět jde zejména o využití statistických postupů. Výsledek benchmarku je v podstatě náhodná veličina a tak je také třeba ho popisovat.

Tedy nespokojit se jen s průměrem nebo podobnou charakteristikou, ale pokud možno ho popsat jako rozdělení, doplněné o údaje jako je střední hodnota, průměr, medián nebo percentily. Důležitý může být i rozptyl a odchylka. Naměřené výsledky jsou obvykle závislé na zvoleném faktoru a tak by měly být prezentovány. Pokud tedy měříte např. dobu odezvy při různé zátěži, je třeba zobrazit naměřené výsledky pro různé zátěže a ne z nich jen počítat průměr nebo něco podobného.

PREZENTACE VÝSLEDKŮ

Prezentace výsledků závisí na cílovém publiku, které obvykle statistice příliš nerozumí, měla by tedy být udělaná tak, aby byla srozumitelná, přehledná a především ne zavádějící. Pokud je to jen trochu možné, je nejlepší využít grafickou prezentaci, grafy, tabulky a schémata. Je ale třeba je alespoň lehce okomentovat, kdyby nic jiného aby bylo jasné jaká hodnota je v grafu nejlepší možná – největší (např. MIPS) nebo nejmenší (např. doba odezvy).

Je důležité zohlednit typ zobrazované veličiny. Diskrétní veličina by nikdy neměla být zobrazena jako křivka. U spojité veličiny je zase třeba dávat pozor na extrapolace – zatímco interpolace je relativně bezpečná, extrapolace může být velmi zavádějící.

S prezentovanými výsledky by měl být neoddělitelně spojen popis použité zátěže a také co nejpřesnější popis testovaného systému. Bez těchto údajů nemají výsledky příliš velký význam, není poznat co vlastně bylo ve skutečnosti naměřeno.