Oponentský posudek BP
Lukáš Haringer

Objektvní analýza výkonu překladačů vybraných programovacích jazyků

Cílem práce bylo v různých jazycích co nejpodobněji implementovat několik algoritmů a následně detailně porovnat kód vyprodukovaný různými překladači dle následujících bodů.

1. **Seznamte se s problematikou objektivního posuzování výkonnosti překladačů programovacích jazyků a s technikami profilingu získaného strojového kódu.**

V práci není nikde nadefinováno, co je to výkon překladače – zda rychlost překladu, nebo velikost paměti potřebné k překladu, atd. V práci se nevysloveně pracuje s předpokladem, že si všichni myslíme, že výkon překladače je totéž co efektivita překladačem vyprodukovaného strojového kódu (která však silně závisí na zdrojovém programovém kódu).

O technikách profilingu se v práci nepíše vůbec nic. Nebyla provedena žádná instrumentace kódu, nebyly zaznamenány žádné hardwarové countery procesoru, nebylo nijako diskutováno využití L1 – L3 mezipamětí, nebyla provedena žádná analýza velikosti dat a velikostí kódu funkcí a smyček – tzv. hot code. Student se vůbec nezajímal o to, jaké strojové instrukce byly vygenerovány – zda např. došlo k autovektorizaci či zvýšenému využití cmov instrukcí – např. u překladače firmy Intel. Navzdory zadání práce („…různými datovými typy s různě těsnou vazbou na architekturu procesoru.“) dokonce nedošlo ani k porovnání výkonnosti pro 32-bitové a 64-bitové formáty čísel v plovoucí čárce dle standardu IEEE 754-1985, nebo k porovnání vlivu přesnosti výsledků matematických operací dle zvolené optimalizace (ačkoliv jsou operace v plovoucí čárce v kompilovaných programech použity). U jazyka Pascal navíc student používá datový typ real, který je závislý na platformě. Na IA-32 se v použitých překladačích sice mapuje na datový typ double, ale student ho použil např. u kódu GEM, kde v jiných jazycích použil datový typ single/float.

2. **Zvolte několik reprezentativních překladačů běžně používaných programovacích jazyků a vyberte vhodné algoritmy pro posouzení jejich výkonnosti.**

Student zvolil jazyky C, C++, C#, Java, Pascal a Scala. Jazyky C, C++, C# a Java lze akceptovat jako běžně používané jazyky. Ale nijak nezdůvodnil, proč je mezi ně zařazen jazyk Pascal (který je navíc už reálně několik let na výrazném ústupu) a jazyk Scala - proč dostal přednost před častěji používaným např. JavaScriptem, Pearlem, Matlabem či Swiftem?

U zvolených algoritmů student nijak nediskutuje, proč vybral právě tyto algoritmy a jaký aspekt práce překladače mají otestovat. Dále u algoritmů v sekci 6.3 píše:

> „u algoritmu v Javě musel být použit cyklus pro nulování pole, i když není potřeba, aby javovské překladače nebyly zvýhodněny například oproti překladačům jazyka C.“

Proč? Jednak v jazyce C existuje funkce calloc, která alokovaný blok paměti nuluje stejně, jako to dělá operátor new v Javě, a jednak bylo cílem práce otestovat výkonnost překladačů a ne ohýbat programové paradigma zvolených jazyků tak, aby jejich výsledky vyšly podobně. Když

např. Java dělá oproti C zbytečnou práci tam, kde jí není potřeba, tak to má být ve výsledcích vidět, a ne že se výsledky uměle zkreslí. Zejména, když je v zadání práce uvedeno, že se má provést objektivní analýza.

3. Implementujte vybrané algoritmy v příslušných jazycích a proveďte výkonnostní testy.

Student hodnotil dobu překladu, paměťovou náročnost přeloženého programu, velikost přeloženého programu a dobu běhu přeloženého programu. Student nijak nedokumentuje použité volby překladače. Např. rozdíly mezi /Od a /O3 u C++ jsou tak propastné jak v rychlosti, paměťové náročnosti i velikosti výsledného kódu, že samy o sobě stačí k tomu, aby šlo o předložených výsledcích konstatovat, že jsou naprosto k ničemu. Stejně tak provedené testy nijak nedokumentují rozdíly mezi šablonami a generiky – alespoň u C++, Object Pascalu a Javy.

Student v tabulkách předkládá u každého měřeného ukazatele jedno číslo na každou kombinaci vybraného algoritmu a překladače. Přitom by objektivní zpracování vyžadovalo vícenásobné spuštění programů, aby mohl vyhodnotit všechny kvartily, průměr a střední odchylku měřených ukazatelů. Jedno měření nijak nevypovídá o chybě měření, a tudíž i z tohoto pohledu jsou prezentované výsledky k ničemu.

V návaznosti na přechozí nedostatky je třeba poznamenat, že měřené veličiny student získával pomocí dalších programů, které předcházely spuštění jím přeložených algoritmů. Výsledné časy tak obsahují režii operačního systému, která je díky konceptu Bottom-Half nedeterministická, a tudíž významně zkresluje naměřené hodnoty. Např. u měření doby běhu měl student použít tzv. High-Performance Counter přímo ve svém kódu a zajistit, že kód poběží dostatečně dlouhou dobu.

V implementovaných algoritmech jsou použity napevno zadané vstupní hodnoty těchto algoritmů. Student nijak neanalyzoval, zda některé úseky překladač nevypočítal předem a následně je zcela neeliminoval jako mrtvý kód.

4. Získané poznatky důkladně zdokumentujte a proveďte jejich rozbor.

Student naměřené hodnoty zanesl do několika tabulek. Následný text pak de-facto říká to samé, co už je obsaženo v těchto tabulkách. Není zde žádný náznak o zformulování hypotézy, proč výsledky vyšly tak, jak vyšly – tj. vedoucím práce požadovaný rozbor získaných poznatků.

Práce dle mého názoru nesplnila ani jeden bod zadání a tudíž nemám jinou možnost než ji hodnotit známkou

n e v y h o v ě l

a práci tak nedoporučuji k obhajobě.

V Plzni dne 29. června 2016

Doc. Ing. Tomáš Koutný, Ph.D.
KIV-FAV-ZČU