Hodnocení vedoucího diplomové práce

Autor práce: **Jan Ambrož**
Název práce: **Výkonnostní a paměťová optimalizace nástroje JaCC**

Autor se v práci detailně seznamoval s nástrojem JaCC a navrhoval jeho modifikaci pro úsporu systémových prostředků. V původní verzi program pro zpracování byte-code v řádu stovek MB potřeboval i 10 GB operační paměti, čímž se stával nepoužitelný pro reálné nasazení a proto se autor zaměřil zejména na paměťové optimalizace.

Autor v textu detailně popisuje současnou strukturu nástroje JaCC a navrhuje zavedení cache pro úsporu operační paměti. Následně popisuje implementaci a výsledky optimalizace. Celkově se autorovi povedlo ušetřit zhruba 20-30% paměti avšak za cenu 2-3x zpomalení běhu. Velikost potřebného mistra na disku pro cache autor neuvádí.

Z výsledků je patrné, že k jednoznačnému zlepšení stavu nedošlo. Pokud nebylo možno lepších výsledků dosáhnout, nelze to považovat za chybu autora. Má hlavní výhrada k práci však je, že autor málo analyzuje další možnosti optimalizace a proto není celkově jasné, zda opravdu dosáhl nejlepších možných výsledků.

Konkrétně, kapitola 4.1 analyticky zjišťuje příčiny velké paměťové náročnosti. Přestože uvedené důvody znějí racionálně, nejsou podpořeny daty. Je proto otázka zda bylo slabé místo v JaCCu nalezeno správně. Autor zakládá další rozhodnutí na informacích získaných studiem zdrojového kódu, nikoliv na profilování aplikace. Sice na straně 33 uvádí, že měření proběhlo, ale není potom jasné, proč výsledky nepublikuje.

Jako hlavní problém JaCC autor uvádí příliš velkou provázanost objektů v doménovém modelu. Autor se však nepokusil model upravit, aby se snížila provázanost a tím množství uložených dat. Bez větší diskuse jako jediné možné řešení navrhuje použití cache pro odkládání objektů z paměti na disk. Toto je hlavní slabina práce, neboť měl autor navrhnout více řešení a pečlivěji, na základě měření, se pro jedno rozhodnout.

Autor pro cachování používá nástroj Ehcache, který serializuje objekty na disk. Na straně 34 je toto řešení odůvodněno jako efektivnější, než opakované načítání přímo byte-code. Vzhledem k jiným částem práce to však nezní logicky. Autor totiž jinde uvádí, že příčina velkého zpomalení výsledného řešení je dána nutností diskových operací. Byte-code však na disku zabírá (předpokládám řádově) méně místa než serializované objekty. Logicky se pak nabízí, že opakované načítání byte-code je výhodnější než serializace a deserializaceí. Samozřejmě zpracování byte-code vyžaduje jistou režii, ale stejně tak serializace a deserializace není laciná operace. Navíc výsledky na straně 59 a dále spíše ukazují na poměrně velkou aktivitu garbage kolektoru při zavedení cache. Nemohlo toto společně s režií serializace být skutečným důvodem zpomalení, namísto diskových operací? Asi ano, neboť autor dále tvrdí, že knihovna Kryo o

SOUHLASÍ
S ORIGINÁLEM

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
katedra informatiky a výpočetní techniky

polovinu zkrátila běh programu oproti pomalé Java serializaci. Konečně autor použil k měření rychlý SSD disk.

Nutno konstatovat, že kdyby autor faktům z předchozích kapitol věnoval větší pozornost, zejména pečlivěji analyzoval a zvolil způsob optimalizace, mohla vzniknout nadstandardně dobrá práce. Teoretická část práce je totiž (v dalších směrech) vyvedena velice kvalitně. Nejenom že detailně popisuje nástroj JaCC, ale je i bohatě ilustrována diagramy a celkově může v budoucnu sloužit jako dokumentační zdroj pro nástroj JaCC a pro práci dalších studentů. Také výsledky, pomineme-li výhrady výše, jsou pečlivě zpracovány. Autor provedl měření na několika reálných aplikacích a zjistil paměťovou úsporu v různých částech JaCCu. Jelikož spotřeba paměti je v Javě těžko měřitelná z důvodu spouštění garbage kolektoru mimo dosah programátora, autor velice správně tento problém řeší opakovaným měřením a statistickým zpracováním výsledků.

Autor pracoval samostatně a využíval společné schůzky k řešení dotazů, návrh řešení, realizaci, výsledky navrhl a zpracoval zcela samostatně.

Souhrnně lze tedy říci, že autor některé části práce vyvedl v nadstandartní kvalitě, bohužel však zanedbal některé části úzce související s hlavním smyslem práce.

Navrhuji hodnocení známkou **velmi dobře** a práci doporučuji k obhajobě.

Otázky:

1. Bylo provedeno měření pro zjištění skutečně nejvíce paměťově náročných částí JaCCu? Pokud ano, které části to byly a proč nejsou výsledky uvedeny v textu?
2. Proč jste se nepokusil omezit paměťovou náročnost optimalizací vazeb v doménovém modelu a tím omezit množství dat držených v paměti?
3. Kolik místa na disku je potřeba pro cache pro testované aplikace? Jaký je poměr vůči velikosti byte-code?
4. Nebylo důvodem zpomalení aplikace ve skutečnosti velká aktivita garbage kolekce a také režie serializace (např. výpočet hashcode a equals u HashMap, kterých je v JaCC velké množství)?

V Plzni 27.5.2016

Ing. Kamil Ježek, Ph.D.