Posudek oponenta diplomové práce

Autor/autorka práce: **Jan Ambrož**

Název práce: **Výkonnostní a paměťová optimalizace nástroje JaCC**

Obsah práce

Práce se zaměřuje na popis nástroje JaCC a možnosti jeho optimalizace z hlediska spotřebovávané paměti. V první části jsou popsány využitelné technologie – tedy možnosti analýzy spotřeby zdrojů u Java aplikace, knihovna pro cachování Java objektů a fungování nástroje JaCC. Druhá část poměrně podrobně popisuje jak je cachování do JaCC implementováno a srovnává výsledky měření pro původní a optimalizovanou implementaci. Vzhledem k tomu že náročnější algoritmy mohou snadno narazit na paměťové nebo výkonnostní limity, jde o velmi zajímavé téma.

Teoretická část je zpracována poměrně přehledně, i když bych v ní očekával některé další informace. V kapitole 2.1.3 se mluví o určení velikosti objektu v paměti, jsou uvedeny tři základní způsoby jak měření provést, ale dva z nich jsou nespolehlivé (jak je i v textu uvedeno). Není ale zmíněna možnost využití Instrumentation rozhraní které je v Javě od verze 1.5 a jeho metody getObjectSize() která problém alespoň částečně řeší, ani využití heapwalku který mohou profilery provést a spočítat tak nejen velkost objektu ale i velikost všech objektů z něj referencovaných. Popis profilerů mi připadá lehce zmatený, v kapitole 2.2.1 je popsáno, jak profilery sledují běh aplikace (tedy jaké metody jsou spuštěny), ale nic o tom jak pracují se sledováním stavu paměti (možnost uložit heap a procházet ho, případně možnosti samotné Javy pro práci s heapem, jako je jmap a jhat), v následující kapitole jsou jako alternativa k profletům uvedeny možnosti měření stavu paměti přímo z aplikace. V podstatě se tedy srovnává schopnost profilerů zjistit, jaké metody program používá se schopností JVM zjistit kolik je aktuálně obsazeno paměti. Samotný JaCC je popsán pro mě srozumitelně, i když příliš nerozumím tomu, co znázorňuje obrázek 3.2 (respektive jak demonstruje rozdíl mezi dvěma způsoby vyhodnocování).

Ve druhé části je popsána vytvořená implementace, která do JaCC doplňuje cachování. Popis implementace je srozumitelný, ale chybí mi zde alespoň krátká diskuse toho, proč bylo zvoleno zrovna toto řešení – očekával bych také rozbor algoritmů JaCC a jeho způsobu nakládání s pamětí, tedy zjištění jestli by nebylo možné nějakou paměť ušetřit změnou těchto algoritmů. I kapitola věnovaná profilování a měření velikosti objektů v paměti naznačuje, že na začátku mělo být něco podobného provedeno – v práci je zmíněno, že měření profilerem a sledování objektů v paměti proběhlo, ale nejsou popsány žádné výsledky.

Kvalita řešení a dosažených výsledků

Nová implemetnace vedla podle naměřených výsledků ke snížení celkové paměťová náročnosti – ovšem jen cestou cachování, takže data jsou odkládána na disk a výrazně je tak zpomalen běh aplikace - jde tedy spíš o mechanickou úpravu „něco za něco" než o optimalizaci algoritmů JaCC, je ovšem otázka nakolik by byla tato optimalizace možná (práce se o tom nezmiňuje).

V práci prezentované výsledky ukazují, že cachování vede k výraznému nárůstu počtu garbage kolekcí, z většiny prezentovaných grafů ani není patrné, že ke skutečnému zlepšení došlo – peaky v grafech naopak ukazují, že nárazově aplikace využije víc paměti než před tím a že vzniká velké množství následně rušených objektů. Významný je tedy graf 4.19, na kterém je vidět že aplikace doběhne i

s menším množstvím paměti než před úpravou potřebovala a že úprava vedla ke skutečnému zlepšení (předchozí verze aplikace nebyla schopna s tímto množstvím pamětí výpočet dokončit).

Celkově práce ukazuje velmi obecnou možnost jak implementovat cachování v Java aplikacích, což považuji za přínosné, na druhé straně je škoda že se víc nezaměřuje na specifické algoritmy JaCC a možnosti jejich vylepšení.

Formální úroveň

Text práce je čitelný a bez velkých chyb, jen místy jsou používány zvláštní slova a obraty (niance místo nuance, „vyprezentovaná třída" a podobně). Pokud jde o zdrojové texty, řešení je součástí mnohem rozsáhlejšího projektu a diplomantova práce j v něm roztroušena na různých místech podle potřeby. Části, které dokážu určit jako dílo pana Ambrože, vypadají přehledně, čitelně a respektují stávající strukturu projektu. Projekt je opatřen značným množstvím unit testů a úprava implementace byla provedena i s ohledem na zachování jejich funkčnosti.

Grafy popisující výsledky experimentů jsou hůř čitelné, měření probíhala různou dobu a aplikace obsazovala různé množství paměti, takže v každém grafu mají osy jiné měřítko.

Práce s literaturou

Veškerá citovaná literatura je relevantní a tvrzení jsou v práci řádně citována.

Splnění zadání

Všechny body zadání považuji za splněné, s tím že výsledné řešení je lépe škálovatelné a umožňuje spustit rozsáhlejší výpočet za cenu dalšího prodloužení doby zpracování. Nicméně systémové zdroje nejsou ušetřeny, jen jinak využívány.

Dotazy k práci

1. Proběhla analýza využití paměti při běhu JaCC? Ukázalo se že úprava, která by snížila potřebu alokované paměti (změna datových struktur, zbytečně vytvářené objekty, ...) není možná?
2. Při popisu lazy loadingu je zmíněno opakované načítání, co je tím myšleno? Objekty, které jsou líně načítány, jsou po použití ihned odstraňovány? Zvažoval jste nějakou možnost řízené toho, které objekty mají být zachovány a které zahozeny (podobně jako když OS rozhoduje o stránkách v paměti) na základě jejich využití v JaCC?

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

V Plzni 20.5.2016

Ing. Richard Lipka, Ph.D.

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