Posudek vedoucího na diplomovou práci
Martin Liška: Optimalizace rozsáhlých aplikací

Práce Martina Lišky se zabývá problematikou optimalizace velkých aplikací v prostředí vývojových nástrojů GNU. Současné nástroje jsou optimalizované a testované zejména na malých a středních programech. Velké programy přinášejí specifické problémy.

Existuje několik významných týmů (například v Google, v Mozilla Foundation i v komunitě vývojářů KDE či LibreOffice), které se téměř výhradně zabývají problematikou řešení problémů toolchainu pro danou aplikaci. Uveďme například nástroj ElfHack, který redukuje velikost relokací pomocí komprese relokací formátu ELF, KDE init obcházející dlouhé doby dynamického linkovaní pomocí zavedení dynamické knihovny s vlastním programem, a nebo LIPO implementující optimalizace mezi moduly specificky pro interní potřeby Google. Tyto nástroje však často fungují jen pro danou aplikaci.

Úkolem řešitele práce bylo zmapovat tuto oblast a vytvořit nástroje i metody pro analýzu jednotlivých problémů (jako je například doba startu aplikace, množství stránek nahraných z disku atd.). Na základě těchto měření pak navrhnout a částečně implementovat dlouhodobější řešení na úrovni oficiálního toolchainu.

V první části řešení projektu bylo nutné opravit všechny vybrané aplikace (LibreOffice, Chromium, Forefox, SPEC2006, a linuxové jádro) pro překlad vývojovou verzí GNU nástrojů se zapnutou link-time optimalizací a optimalizací na základě profilu. Chromium i LibreOffice byly s link-time optimalizací na Linuxu přeloženy vůbec poprvé. Podstatnou i časově náročnou součástí práce byla analýza a izolace chyb v překladači, linkeru i samotných aplikacích.

Analýza jednotlivých výkonnostních problémů se ukázala velmi přínosná. Martin Liška navázal komunikaci s jednotlivými vývojovými týmy, zopakoval starší experimenty v současných podmínkách, napsal několik skriptů pro zpracování a vizualizaci získaných dat a porovnal klasický model překladu s nově implementovanou link-time optimalizací. Naměřená data vedla k lepšímu pochopení problému mezi vývojáři GCC a GNU Binutils. Praktickým výsledkem je několik zlepšení, které byly v posledních měsících implementovány komunitou vývojářů GCC přímo za účelem řešení nalezených problémů.

- Byly odstraněny nedostatky měření profilu v GCC pro programy používající fork, vfork a inline funkce v C++.
- Byly opraveny problémy ve spekulativní devirtualizaci na základě profilu.
- Nová eliminace prokazatelně nedosažitelných virtuálních funkcí během kompilační fáze (místo při linkování) zmenšila nároky na paměť při link-time optimalizaci Firefoxu z 16GB na 4GB.
- Nový algoritmus na unifikaci typů výrazně zrychlil linkováni Chromia, Libreoffice i Firefoxu.

- Vlastní link-time optimalizace nyní využívá lépe více-jádrové procesory.
- Bylo projednáno několik drobných změn v implementaci C++ ABI, které vedou k podstatné redukci množstvích externích relokací v LibreOffice.

V důsledku těchto změn se v současné době aktivně pracuje na přechod k link-time optimalizaci pro oficiální vydání Firefoxu pro Linux i Android. Toto považuji za velký úspěch. Link-time optimalizace i optimalizace na základě profilu trpěly na malý počet uživatelů a tím i testerů. První experimenty s optimalizaci Firefoxu začaly v roce 2010, dodnes se však nepodařilo je prosadit pro produkční nasazení.

Podstatné jsou i dvě nově implementované optimalizace, které diplomant implementoval samostatně. První optimalizace uspořádá funkce v programu podle času prvního provedení získaného z profilu. V době odevzdání práce bylo možné prokázat zrychlení na startu středně velkých aplikací. Po odstranění výše zmíněných problémů v runtime lze dnes ukázat zlepšení ve spotřebě paměti při startu Firefoxu z 60MB na přibližně 30MB a po odstranění dalších nedostatků i více.

Druhá optimalizace implementuje unifikaci sémanticky identických funkcí. Tato optimalizace je motivována identickou transformací prováděnou na úrovni linkeru Gold. Řešení na úrovni překladače slibuje lepší unifikaci profilu a spolupráci se zbytkem optimimizeru. Jak se parametry překladu stávají více a více závislé na kontextu dané funkce, implementace na úrovni linkeru přestane být efektivní. Současná optimalizace dosahuje podobných výsledků jako implementace v linkeru Gold, ale ani jedna nedokáže nahradit druhou. Nevýhodou práce na úrovni mezikódu je nutnost zachovat konzistentní typovou informaci pro pozdější alias analýzu.

V současné době se aktivně pracuje na zařazení obou optimalizací do oficiální verze GCC 4.9.0.

Jako jazyk textové často práce byla zvolena angličtina, aby výsledky byly dostupné pro co nejširší komunitu vývojářů. Toto rozhodnutí se ale ukázalo poněkud problematické. Vlastní textová práce obsahuje množství jazykových nedostatků, které zbytečně degradují kvalitu prezentace jinak zajímavých výsledků. Na práci je také vidět, že velká část textu vznikla v posledních týdnech.

Vzhledem k tomu, že množství odvedené práce, náročnost celého projektu i praktické výsledky považuji za velmi nadprůměrné v rámci diplomových prací, navrhuji i přes zmíněné prezentační problémy textové části **přijmout** tuto práci jako diplomovou. Navrhuji hodnocení **výborně**.

V Táboře, 30. srpna 2013

Jan Hubička