EXAMENSARBETE Code positioning in LLVM
STUDENT Henrik Lehtonen och Klas Sonesson
HANDLEDARE Jonas Skeppstedt (LTH)
EXAMINATOR Krzysztof Kuchcinski (LTH)

Kodpositionering i LLVM

POPULÄRVETENSKAPLIG SAMMANFATTNING **Henrik Lehtonen och Klas Sonesson**

Det finns en enorm prestandaskillnad mellan processorn och primärminnet i dagens datorer. Varje processor idag har därför cacheminne som mellanlagrar data från primärminnet. Cacheminne är mycket snabbare än primärminnet, men är också mycket mindre. Att använda cacheminnet effektivt kan därför ge stora prestandavinster. Detta arbete undersöker två optimeringar som försöker förbättra cachebeteendet hos program.

Majoriteten av alla program idag är skrivna i ett högnivåspråk, till exempel C eller Java. För att processorn ska kunna köra ett sådant program måste det översättas till maskinkod. Denna process kallas för *kompilering*, och utförs av en *kompilator*. Under kompileringen kan programmet automatiskt optimeras för att t.ex. bli snabbare eller mindre. Maskinkod består av instruktioner som lagras i minnet precis som annan data. De är dock ofta mellanlagrade i ett separat cacheminne som kallas för en instruktionscache.

1990 publicerade Pettis och Hansen en artikel som beskriver ett antal optimeringar vars syfte är att förbättra instruktionscachebeteendet hos ett program genom att automatiskt omorganisera dess kod under kompileringen. Vårt arbete fokuserar på implementering och testning av två av dessa optimeringar för att se om de fortfarande är relevanta idag.

Vissa optimeringar beror på information om hur programmet som kompileras beter sig, vilket kan sammanfattas i en så kallad profil. En profil kan skapas genom att samla information under programkörningen. Några exempel på vad en profil kan berätta är vilka delar av programmet som körs ofta och vilka delar som kanske inte har körts alls. De två optimeringarna som vi undersöker använder sig av denna profilinformation för att kunna organisera koden.

En av optimeringarna grupperar kod som är relaterad enligt programmets profil. Den andra flyttar bort kod som inte körs ofta så att resten av koden är mer kompakt. I teorin kan programmet då använda cacheminnet bättre.

LLVM är ett open-source kompilatorramverk. De två optimeringarna som vi valde ut implementerades i LLVM för att testa deras inverkan på olika program.

För att testa optimeringarna på en så varierad mängd av program som möjligt använde vi oss av benchmarksviten SPEC CPU2000 och databassystemet PostgreSQL.

Vi fann att optimeringarna hade en varierande inverkan på prestandan av testprogrammen. Hos vissa program förbättrades prestandan med upp till 3%. Andra program blev däremot långsammare.

Vår slutsats är att optimeringarna är relevanta för vissa program. Optimeringarnas inverkan är kraftigt beroende av vissa egenskaper hos programmen, exempelvis deras storlek och vilken typ av arbete de utför.