INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2015-06-04

EXAMENSARBETE Processor Models for Instruction Scheduling using Constraint Programming
STUDENT Karl Hylén
HANDLEDARE Jonas Skeppstedt (LTH)
EXAMINATOR Krzysztof Kuchcinski (LTH)

Modeller för optimal schemaläggning av maskininstruktioner

POPULÄRVETENSKAPLIG SAMMANFATTNING **Karl Hylén**

Vid översättning från kod till program uppstår svåra problem som normalt löses approximativt. I det här arbetet har metoder för att lösa ett av dessa problem optimalt tagits fram med speciellt fokus på betydelsen av detaljerade modeller.

Det här arbetet är också först med att utvärdera optimala metoder genom mätningar på program skapade med dem. Resultatet visar att det finns mycket att vinna på att använda dessa metoder. Prestandaförbättringarnas stabilitet beror starkt på modellens detaljrikedom.

Idag finns programvara överallt, inte bara i mobiltelefoner och datorer. Bilar, tvättmaskiner och olika typer av sjukhusutrustning är exempel på saker som innehåller så kallade *inbyggda system*. För de flesta tillämpningarna är prestanda mycket viktigt. Dessutom möjliggör bättre prestanda att billigare hårdvara kan användas.

Programs prestanda beror mycket på hur bra översättningen från kod till *maskinkod* är. Maskinkod är formatet en dator förstår, och består av en lista av *instruktioner*. Varje instruktion utför en enkel operation, som till exempel addition av två tal. Översättningen till maskinkod kallas *kompilering* och utförs av ett program som kallas *kompilator*. Under kompileringen utför kompilatorn en rad optimeringar på programmet. En av dessa optimeringar är schemaläggning av instruktioner, så att de kommer i en ordning som passar datorns räkne-enhet, *processorn*.

Instruktioner i ett program är normalt beroende av varandra. Resultatet av en addition kan till exempel användas i en senare multiplikation. Bara oberoende instruktioner kan ordnas om. På grund av hur processorn är konstruerad kan en ordning vara snabbare än en annan. En konstruktionsteknik som har den effekten är *pipelining*. Det innebär att instruktioner utförs i steg, som kan liknas vid steg i ett fabriksband. Om oberoende instruktioner placeras nära varandra kan en instruktion påbörjas i varje tidssteg. Instruktioner som beror av en annan instruktion måste vänta på att denna ska bli klar.

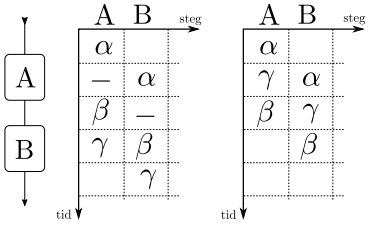

Illustration av pipelining. Pipelinen består av två steg, A och B och utför tre instruktioner α, β och γ, där β beror på α och måste vänta tills den är klar.

Att hitta den bästa ordningen för instruktionerna är ett mycket svårt problem. Det tillhör en klass av problem inom datavetenskapen som kallas NP-svåra problem. Att hitta en effektiv algoritm för problem i den här klassen, eller bevisa att det inte finns någon är ett av milleniumproblemen inom matematiken. När optimala metoder används, kan vi alltså bara hoppas konstruera en algoritm som fungerar tillräckligt bra för de vanligaste, minsta programmen, och kompilatorn kommer kräva mycket tid.

Under arbetet konstruerades modeller med olika detaljrikedom av processorn. För att göra detta användes *constraint* programmering, som är en form av programmering där man uttrycker sig med hjälp av krav på en sökt lösning. Modellerna byggdes in i en öppen kompilator av industriell styrka, *LLVM*. Med hjälp av LLVM genereras program som kan användas för mätning och utvärdering av modellerna.

Arbetet kan fungera som en grund för vidare utveckling av optimala metoder. Det har visat vikten av att modellen överensstämmer väl med processorn. Dessutom fungerar arbetet som en påminnelse om hur viktigt det är att mäta prestandan på riktiga program när man forskar om kompilatoroptimeringar.