STUDENTER Anton Botvalde, Andreas Larsson
HANDLEDARE Jonas Skeppstedt (LTH)
EXAMINATOR Per Andersson (LTH)

Prestandaevaluering av ISO C restrict på Power-arkitekturen

POPULÄRVETENSKAPLIG SAMMANFATTNING **Anton Botvalde och Andreas Larsson**

Introduktion

Moderna C-kompilatorer kan utföra avancerade optimeringar som gör att ett C-program blir mycket snabbare. I C så tillåts det att flera olika pekare pekar på samma objekt i minnet, vilket är känt som alias. Detta försvårar möjligheterna för en kompilator att spara ett värde som en pekare pekar på i ett register då kompilatorn har svårt att avgöra om någon annan pekare eventuellt pekar på samma minnesplats. Vårt arbete har visat att optimeringen kan ge en prestandaökning men också, något oväntat, en prestandaförlust.

Bakgrund

För att underlätta pekaroptimeringar introducerades ett nytt nyckelord, restrict. Med restrict garanterar programmeraren både kompilatorn och andra programmerare att om en restrict-pekare ändrar värdet på det pekade objektet så är det den enda pekaren som kommer åt detta objekt. Med en sådan garanti blir det lättare för kompilatorn att utföra optimeringar som t.ex. spara undan värdet i ett register, vilket ofta leder till en prestandaökning. En läsning från ett register är avsevärt snabbare än en läsning från RAM-minnet.

I det här examensarbetet har vi undersökt om C-kompilatorerna GCC, Clang och XL C har implementerat optimeringar med användning av restrict på IBMs Power-arkitektur.

För att undersöka prestandapåverkan av restrict så konvertade vi ett Fortran-prestandamätningsprogram, mer känt som Livemore loops, till C. Looparna passar utmärkt som grund till utvärdering av restrict då de innehåller många möjligheter för restrict-baserade optimeringar.

Prestandaresultat

Totalt sett så gav restrict en prestandaökning men i vissa enskilda loopar så gav det en prestandaminskning. I loop 1 med GCC visade det sig att optimeringen bytte en minnesläsning till en registerläsning vilket ledde till en sämre schemaläggning av koden. Restrict i Clang hade liten påverkan på prestandan då Clang utför en egen aliasanalys av pekare men det fanns även fall där analysen inte räckte till. I de fallen gjorde restrict både positiva och negativa förändringar.

Type	Double			Int		
	Restrict	Unroll	Unroll restrict	Restrict	Unroll	Unroll restrict
GCC	16.31	8.15	24.14	11.73	12.35	14.32
XL C	23.18	5.21	31.04	13.71	15.78	24.50
Clang	1.16	-	-	-0.94	-	-

Tabell Genomsnittlig förändring av exekveringstid mot ingen restrict i procent. Positiva värden innebär en minsking av exekveringstiden medan negativ innebär en ökning.

Slutsats

Alla tre kompilatorer har implementerat pekaroptimeringar som utförs vid användningen av restrict. IBM har generellt lyckats bäst när det kommer till prestandaökningen, vilket inte var oväntat då de har mest kännedom om arkitekturen.

Eftersom restrict kan försämra prestanda så är det inte rekommenderat som programmerare att lägga till restrict i sin kod hur som helst. Vid arbete med en befintlig kodbas rekommenderar vi att identifiera loopar som ser ut att vara lämpliga kandidater för restrict och undersöka hur prestandan påverkas.

För kompilatorskrivare är restrict lite av ett dilemma. Prestandan kan påverkas negativt och själva optimeringen kan vara svår att implementera. Eftersom det inte alltid lönar sig att ersätta en läsning från minnet till en registerläsning behöver kompilatorn antagligen en modell av arkitekturen för att kunna schemalägga koden effektivt. Att som kompilatorskrivare behöva lägga mycket tid på en optimering är inte en självklarhet och bör därför noggrant övervägas innan eventuell implementering. ■