INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2016-06-03

EXAMENSARBETE A Quantitative Evaluation and Proposition of Cache Policies in Mobile GPUs
STUDENTER Fredrik Paulsson, Shan Senanayake
HANDLEDARE Michael Doggett (LTH), Fuad Tabba (ARM Sweden AB)
EXAMINATOR Flavius Gruian (LTH)

Bättre Prestanda i Mobila Grafikprocessorer

POPULÄRVETENSKAPLIG SAMMANFATTNING **Fredrik Paulsson, Shan Senanayake**

De senaste 50 åren har gapet i prestanda mellan processorn och primärminnet ökat, vilket gjort primärminnet till en flaskhals för processorn. För att lindra problemet används cachar - små snabba minneselement - placerade mellan processorn och primärminnet. Genom att utvärdera algoritmerna som väljer ut innehållet i cacharna - och välja ut de bästa - kan man öka prestandan för processorn.

Förbättring med cachar

Under de senaste 50 åren har datorarkitekturen utvecklats enormt. Prestandan har mångdubblerats för både processorer och primärminnen men inte lika fort. Primärminnet har inte haft samma hastiga utveckling som processorn vilket lett en växande klyfta mellan processorns och primärminnets prestanda. Resultatet blir att processorn tvingas vänta på primärminnet vid minnesaccesser. Eftersom minnet används ofta betyder detta att en stor del av den potentiella prestandan i processorer går upp i rök.

Lösningen för att överbrygga klyftan i prestanda mellan processorn och primärminnet är att använda s.k. cachar. Cachar är små minneselement som är mycket snabbare är primärminnet. Dessa placeras sedan i flera nivåer mellan processorn och primärminnet. En minnesaccess går sedan igenom alla nivåer av cachar från processorn till primärminnet tills den når den första nivån som innehåller den efterfrågande datan. Datan skickas därefter tillbaka till processorn genom alla nivåer igen.

När ny data kommer till en nivå där den inte redan finns i cachen måste den placeras i cachen. Om cachen vid detta tillfälle är full måste gammal data bytas ut mot den nya. Algoritmerna som väljer ut vilken gammal data som skall bytas ut eller som bestämmer om det är värt att byta ut gammal data för den nya datan kallas för *Cache Policies*.

Cachar i grafikprocessorn

Mycket forskning har gjorts inom cache policies men inte inom grafikprocessorer och därför finns det ett behov av att undersöka dessa algoritmer för cacharna i en sådan. Grafikprocessorer arbetar med helt andra minnesaccessmönster än vanliga processorer och använder samtidigt betydligt mycket mer bandbredd för accesser till primärminnet. Dessa två faktorer bidrar till att det kan finnas cache policies som fungerar bättre för grafikprocessorer än för vanliga processorer. Det är genom undersökningar liknande den vi har gjort som man kan hitta bättre cache policies för grafikprocessorer.

Det är extra viktigt att förbättra prestandan i cacharna för grafikprocessorer i mobiltelefoner. Skälet till detta är att grafikprocessorer arbetar mycket med primärminnet och genom att hantera minnesaccesserna så mycket som möjligt från cacharna får man ökad prestanda för hela grafikprocessorn. Detta är extra viktigt för mobiltelefoner då det även innebär att grafikprocessorerna blir mer energisnåla och därmed sparar på energi som är väldigt viktigt för mobiltelefoner.

Vår undersökning

Vi har gjort vår undersökning genom att implementera olika cache policies i två simulatorer, en som simulerade en cache och en annan som simulerade en hel grafikprocessor Simulatorerna kördes med hjälp av inspelad data från riktiga benchmarks och spel.

Bättre algoritmer

Vi har gjort en undersökning av en mängd olika cache policies med fokus på grafikprocessorer i mobiltelefoner. Vi har även föreslagit nya algoritmer som kan fungera som cache policies baserat på resultaten från vår studie.

I vår studie har vi sett att det är svårt att hitta en cache policy som fungerar bättre än *Least-Recently Used* som är den vanligaste cache policyn som används i dagens cachar. De cache policies som vi ser skulle kunna fungera bättre än Least-Recently Used består i att antingen behandla minnessaccesserna olika beroende på typ eller att spara minnesaccesserna i en buffer innan de flyttas till cachen för att kunna avgöra vilken data som är bäst att flytta.