Analyzing Array Accesses for GPGPUs

- A formal system to track array accesses that are linear polynomials of thread and block ids.
- Our system formalizes the analyses for intra-kernel optimizations like memory coalescing and thread/block merging.
- We show how our system can be used for *kernel merging*.

A Simple GPGPU Language

Types:

$$\text{T'} := \text{uint} \mid \text{int} \mid \text{float} \mid \text{bool}$$

$$\text{T} := \textbf{global}\ T'[] \mid \textbf{local}\ T'[] \mid T' \mid \textbf{const}\ T' \mid \boxed{\textbf{ID}(\text{a, b, c})}$$

Terms

$$\begin{aligned}
e &:= x \mid 1 \ldots \mid tid \mid bid \mid a[e] \\
\text{stmt} &:= \langle\rangle \mid \text{cmd; stmt} \\
\text{cmd} &:= \textbf{if}\ e\ \textbf{then}\ \text{stmt}\ \textbf{else}\ \text{stmt} \\
&\quad \mid \textbf{for}\ x\ \textbf{in}\ [e, e, e]\{\text{stmt}\} \\
&\quad \mid \textbf{sync} \\
&\quad \mid x := e \\
&\quad \mid a[e] := e \\
&\quad \mid \textbf{decl}\ T\ x\ \textbf{in}\ \text{stmt}
\end{aligned}$$

Typing Expressions

$$\frac{}{\Gamma \vdash \mathsf{tid} : \mathbf{ID}(1,0,0)} \qquad \frac{}{\Gamma \vdash \mathsf{bid} : \mathbf{ID}(0,1,0)}$$

$$\frac{\Gamma/\Delta_1 \vdash e_1 : \mathbf{ID}(a_1,b_1,c_1) \qquad \Gamma/\Delta_2 \vdash e_2 : \mathbf{ID}(a_2,b_2,c_2)}{\Gamma/\Delta_1 \cup \Delta_2 \vdash e_1 + e_2 : \mathbf{ID}(a_1+a_2, b_1+b_2, c_1+c_2)}$$

$$\frac{\Gamma/\Delta_1 \vdash e_1 : \mathbf{ID}(0,0,c') \qquad \Gamma/\Delta_2 \vdash e_2 : \mathbf{ID}(a,b,c)}{\Gamma/\Delta_1 \cup \Delta_2 \vdash e_1 * e_2 : \mathbf{ID}(a*c', b*c', c*c')}$$

$$\boxed{\frac{\Gamma/\Delta \vdash a : \mathbf{global}\ T'[] \qquad \Gamma/\Delta \vdash e : \mathbf{ID}(a',b',c')}{\Gamma/(a,(a',b',c')),\Delta \vdash a[e] : T'}}$$

Typing Commands

$$\frac{\Gamma/\Delta_r^1 \vdash e : \mathsf{bool} \qquad \Gamma/\Delta_r^2/\Delta_w^2 \vdash S_1 \text{ ok} \qquad \Gamma/\Delta_r^3/\Delta_w^3 \vdash S_2 \text{ ok}}{\Gamma/\Delta_r^1 \cup \Delta_r^2 \cup \Delta_r^3/\Delta_w^2 \cup \Delta_w^3 \vdash \textbf{if } e \textbf{ then } S_1 \textbf{ else } S_2 \text{ ok}}$$

$$\frac{\begin{array}{c}\Gamma/\delta_r^1 \vdash e_1 : \mathsf{int} \qquad \Gamma/\delta_r^2 \vdash e_2 : \mathsf{int} \qquad \Gamma/\delta_r^3 \vdash e_3 : \mathsf{int} \\ \Gamma/\Delta_r^0/\Delta_w^0 \vdash S[e_1/x] \text{ ok} \qquad \Gamma/\Delta_r^1/\Delta_w^1 \vdash S[e_1 + e_3/x] \text{ ok} \\ \ldots \qquad \Gamma/\Delta_r^{15}/\Delta_w^{15} \vdash S[e_1 + 15 * e_3/x] \text{ ok}\end{array}}{\Gamma/\bigcup \Delta_r \cup \bigcup \delta_r/\bigcup \Delta_w \vdash \textbf{for } x \textbf{ in } [e_1, e_2, e_3]\{S\} \text{ ok}}$$

$$\frac{x : T \in \Gamma \qquad T = \textbf{global } T'[] \qquad \Gamma/\Delta_r^1 \vdash e_1 : \textbf{ID}(a', b', c') \qquad \Gamma/\Delta_r^2 \vdash e_2 : T'}{\Gamma/\Delta_r^1 \cup \Delta_r^2/(a, (a', b', c')) \vdash a[e_1] := e_2 \text{ ok}}$$

Applications

- Our analysis can be used to check the applicability of several optimizations
- **Memory Coalescing:** Check if a half warp (16 threads) accesses an entire memory segment
- **Thread/Block Merging:** If neighbouring blocks access some common memory segments, blocks can be merged, reducing global memory accesses
- **Kernel Merging:** If corresponding threads in two independent kernels accss the same memory segments, the two kernels can be merged