

Generalized Points-to Graphs: A New Abstraction of Memory in the Presence of Pointers

Pritam M. Gharat, Indian Institute of Technology Bombay
 Uday P. Khedker, Indian Institute of Technology Bombay
 Alan Mycroft, University of Cambridge

Additional Material

A Augmented GPU Reduction Algorithm for Computing Queued GPUs

Calculating the set of GPUs **Queued** for dead GPU elimination can be performed parallelly with edge reduction. We thus define a new algorithm given in Definition 1 for GPU reduction which augments the method of computing set **Queued**.

A GPU $p \in \text{Queued}$ could belong to any of the following categories:

- Composition $c \circ^\tau p$ may be postponed because p may be a GPU blocked by the presence of a barrier. It is possible that the barrier which may be simplified after Δ is inlined in a caller and may not block p any more enabling its composition with c .
- Composition $c \circ^\tau p$ may be *undesirable*. It is possible that p may be simplified after Δ is inlined in a caller making the composition *desirable*.

In the first case, a GPU composition is *admissible* when RGln is used for GPU reduction but with the GPU p being blocked ($p \in \overline{\text{RGln}}$), the composition is postponed. These conditions are checked at line numbers 19 and 20 in Definition 1 and accordingly the flag *postpone* is set.

In the second case, we identify a *valid* but *undesirable* GPU composition using the predicate **Undes_comp** which checks that a pivot exists ($v = x$ for *TS* and $u = x$ for *SS*) and the composition is *undesirable* ($l > k$). This check is performed at line number 26 in Definition 1 and accordingly the flag *postpone* is set.

$$\text{Undes_comp}(\tau, u \xrightarrow{i|j}{t} v, x \xrightarrow{k|l}{s} y) = \begin{cases} \text{true} & (\tau = \text{ts}) \wedge (v = x) \wedge (l > k) \\ \text{true} & (\tau = \text{ss}) \wedge (u = x) \wedge (l > k) \\ \text{false} & \text{otherwise} \end{cases} \quad (1)$$

Example 1. The set of GPUs reaching statement 04 is $\{p \xrightarrow{1|0}{02} a, x \xrightarrow{2|0}{03} b\}$. The barrier GPU (in this case $x \xrightarrow{2|0}{03} b$) blocks the GPU $p \xrightarrow{1|0}{02} a$ and hence and hence $\overline{\text{RGln}}_{04} = \{x \xrightarrow{2|0}{03} b\}$. Thus, the composition between GPUs $c : q \xrightarrow{1|1}{04} p$ and $p : p \xrightarrow{1|0}{02} a$ is *admissible* if RGln_3 is used for GPU reduction and the condition on line 19 of Definition 1 is **true**. However since $p \in \overline{\text{RGln}}_3$, the condition on line 20 is **false** and the flag *postpone* is set to **true** indicating that a composition is postponed and p should be added to **Queued** as seen on line numbers 9 and 10 of the algorithm.

```

01 void h()
02 { p = &a;
03   *x = &b;
04   q = p;
05 }
```

```

Input:   $\mathbf{c}$            // The consumer GPU to be simplified
           $R$            // The set of GPUs using which  $\mathbf{c}$  is to be simplified
           $\bar{R}$           // The set of GPUs that have been blocked by a barrier
Output: Red           // The set of simplified GPUs equivalent to  $\mathbf{c}$ 
          Queued         // The set of GPUs which may be used later

01 Augmented_GPU_reduction ( $\mathbf{c}, R, \bar{R}$ )
02 { Red = Queued =  $\emptyset$ 
03    $W = \{\mathbf{c}\}$ 
04   while ( $W \neq \emptyset$ )
05   { extract  $w$  from  $W$ 
06     for each  $\gamma \in R$ 
07     {  $\langle W, tscomp, tspost \rangle = \text{Compose\_GPUs}(ts, w, \gamma, W, \bar{R})$ 
08        $\langle W, sscomp, sspost \rangle = \text{Compose\_GPUs}(ss, w, \gamma, W, \bar{R})$ 
09       if ( $tspost$  or  $sspost$ )
10         Queued = Queued  $\cup \{\gamma\}$ 
11     }
12     if ( $\neg (tscomp$  or  $sscomp)$ )
13       Red = Red  $\cup \{w\}$ 
14   }
15   return (Red , Queued)
16 }
17 Compose_GPUs( $\tau, w, \gamma, W, blocked$ )
18 { composed = postpone = false
19   if ( $r = w \circ \gamma$ ) succeeds
20   { if ( $\gamma \notin blocked$ )
21     {  $W = W \cup \{r\}$ 
22       composed = true
23     }
24     else postpone = true
25   }
26   else if (Undes_comp( $\tau, w, \gamma$ ))
27     postpone = true
28   return  $\langle W, composed, postpone \rangle$ 
29 }

```

Figure 1: Definition of Augmented edge reduction algorithm for computing Queued GPUs

Example 2. The composition between GPUs $\mathbf{c} : p \xrightarrow{\frac{1|2}{3}} y$ and $\mathbf{p} : y \xrightarrow{\frac{1|2}{4}} x$ is *undesirable* because the result of composition is a GPU $p \xrightarrow{\frac{1|3}{3}} x$ whose *indlev* exceeds that of $\mathbf{c}$. This composition will be performed once the $\mathbf{p}$ is simplified. The predicate Undes_comp returns **true** because ($l > k$) (in this case $l = 2$ and $k = 1$) indicating that the composition is *undesirable* and adds $\mathbf{p}$ to the set Queued.

B Additional Data

We have evaluated our GPG-based points-to analysis implementation on SPEC CPU 2006 benchmarks. The data is presented in the form of graphs. However, the number of procedures varies significantly across the benchmark programs. Besides, the number of GPUs and GPBs varies across GPGs. Hence we have plotted such data in terms of percentages. The actual procedure counts are given in the tables below.

Program	0 GPUs	0 GPBs	% of Dead GPUs					% of empty GPBs eliminated					% of GPBs reduced because of coalescing					% of back edges reduced because of coalescing			
			0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-80	80-100
lbm	15	0	4	0	0	0	0	4	0	15	0	0	19	0	0	0	0	0	0	0	0
mcf	12	0	10	1	0	0	0	14	0	8	0	1	17	2	3	1	0	1	0	0	3
libquantum	45	0	32	2	1	0	0	31	0	47	2	0	61	7	4	5	3	0	0	0	6
bzip2	78	8	9	0	2	0	0	22	0	59	0	0	78	2	1	0	0	1	0	0	0
milc	184	3	12	0	0	0	0	61	0	131	1	0	190	2	1	0	0	0	0	0	1
sjeng	101	2	29	1	2	0	0	8	0	101	18	4	124	1	3	3	0	0	0	0	9
hmmer	241	5	31	0	1	0	2	68	0	202	0	0	265	2	1	1	1	1	0	0	1
h264ref	439	3	123	2	1	1	0	195	2	343	18	5	534	13	10	4	2	5	0	0	3
gobmk	1437	2	1260	2	0	0	0	1268	0	1380	45	4	2144	241	164	89	59	7	1	0	97

Table 1: Effectiveness of redundancy elimination optimizations.

3

Program	Proc. count for different buckets of # of GPBs						Proc. count for different buckets of # of GPUs								Proc. count for different buckets of % of CI					
	0	1-3	4-10	11-25	26-35	>35	0	1-3	4-6	7-10	11-30	31-50	51-70	>70	0-20	20-40	40-60	60-80	80-100	
lbm	0	18	1	0	0	0	15	4	0	0	0	0	0	0	4	0	0	0	0	
mcf	0	22	1	0	0	0	12	6	2	2	1	0	0	0	8	0	0	1	2	
libquantum	0	80	0	0	0	0	38	42	0	0	0	0	0	0	38	2	0	0	2	
bzip2	8	79	2	0	0	0	78	10	1	0	0	0	0	0	11	0	0	0	0	
milc	3	191	2	0	0	0	184	6	5	1	0	0	0	0	4	0	0	0	8	
sjeng	2	128	3	0	0	0	101	26	1	3	2	0	0	0	1	0	2	3	26	
hmmer	5	254	15	1	0	0	242	27	5	1	0	0	0	0	26	2	1	0	4	
h264ref	3	531	23	7	2	0	434	81	20	8	18	3	1	1	78	3	4	0	47	
gobmk	2	2568	120	3	0	6	1436	83	87	972	120	1	0	0	1077	33	60	31	62	

Table 2: Measurment of the goodness of procedure summaries.

Program	Time (in seconds)													
	FSCS with blocking						FSCS without blocking						FICI	FICS
	Total	Streng. Reduc.	Redundancy Elim				Total	Stren. Reduc.	Redundancy Elim					
			Total	Dead GPU Elim	Empty GPB Elim	Coalesce			Total	Dead GPU Elim	Empty GPB Elim	Coalesce		
lbm	0.085	0.043	0.001	0.000	0.000	0.000	0.084	0.041	0.001	0.000	0.000	0.000	0.072	0.108
mcf	5.815	5.052	0.032	0.002	0.003	0.017	3.053	2.529	0.021	0.001	0.003	0.008	13.224	47.088
libquantum	0.830	0.518	0.066	0.001	0.012	0.030	0.601	0.348	0.033	0.001	0.006	0.014	0.067	0.393
bzip2	1.267	0.582	0.010	0.000	0.002	0.003	1.183	0.530	0.009	0.000	0.001	0.003	0.867	2.787
milc	1.329	0.664	0.017	0.001	0.003	0.006	1.301	0.640	0.016	0.000	0.003	0.006	3.301	4.538
sjeng	4.304	2.184	0.081	0.001	0.004	0.068	4.001	1.861	0.077	0.001	0.003	0.066	1.057	4.073
hmmer	6.225	4.344	0.087	0.002	0.015	0.037	6.006	4.117	0.081	0.001	0.015	0.035	27.453	26.521
h264ref	80.319	64.542	0.388	0.005	0.046	0.188	88.464	76.217	0.397	0.002	0.057	0.182	234.175	841.067
gobmk	462.875	353.583	49.372	0.232	3.463	40.522	248.393	169.199	24.784	0.127	1.319	20.773	36.193	707.428

Table 3: Time measurements.

Program	Proc. count for different buckets of ratio of GPBs/BBs in CFG and GPG after inlining					Proc. count for different buckets of ratio of GPBs/BBs in CFG and optimized GPG					Proc. count for different buckets of ratio of GPBs in GPG after inlining and optimized GPG				
	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100
lbm	9	3	3	1	3	11	5	3	0	0	2	1	15	0	1
mcf	14	5	2	1	1	22	1	0	0	0	3	5	13	2	0
libquantum	40	16	12	1	11	56	13	11	0	0	15	21	41	2	1
bzip2	55	15	8	4	7	70	12	7	0	0	12	2	68	4	3
milc	118	21	14	9	34	138	24	34	0	0	10	6	174	2	4
sjeng	85	19	7	3	19	105	9	19	0	0	16	14	102	0	1
hmmer	194	41	22	1	17	235	23	16	0	1	29	37	194	4	11
h264ref	401	69	49	10	37	474	50	40	2	0	26	65	437	13	25
gobmk	2315	279	29	9	67	2594	37	66	1	1	141	464	1494	579	21

Table 4: Relative size of GPGs with respect to the size of corresponding procedures in terms of GPBs and basic blocks.

Program	Proc. count for different buckets of ratio of GPUs/stmts in CFG and GPG after inlining					Proc. count for different buckets of ratio of GPUs/stmts in CFG and optimized GPG					Proc. count for different buckets of ratio of GPUs in GPG after inlining and optimized GPG				
	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100
lbm	16	3	0	0	0	19	0	0	0	0	18	0	0	1	0
mcf	21	0	1	1	0	23	0	0	0	0	17	0	3	0	3
libquantum	75	4	0	0	1	80	0	0	0	0	45	10	7	6	12
bzip2	89	0	0	0	0	89	0	0	0	0	85	0	0	0	4
milc	195	1	0	0	0	196	0	0	0	0	191	0	0	0	5
sjeng	131	0	2	0	0	133	0	0	0	0	105	0	1	2	25
hmmer	273	0	1	0	1	275	0	0	0	0	266	7	1	0	1
h264ref	540	12	10	1	3	563	2	1	0	0	505	3	1	1	56
gobmk	2690	4	1	0	4	2698	1	0	0	0	1460	1	5	21	1212

Table 5: Relative size of GPGs with respect to the size of corresponding procedures in terms of GPUs and pointer assignments.

Program	Proc. count for different buckets of ratio of CF edges in CFG and GPG after inlining					Proc. count for different buckets of ratio of CF edges in CFG and optimized GPG					Proc. count for different buckets of ratio of CF edges in GPG after inlining and optimized GPG				
	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100	0-20	20-40	40-60	60-80	80-100
lbm	13	4	2	0	0	19	0	0	0	0	18	0	0	0	1
mcf	21	1	1	0	0	23	0	0	0	0	16	3	3	1	0
libquantum	61	8	2	0	9	80	0	0	0	0	64	11	4	0	1
bzip2	74	8	3	0	4	89	0	0	0	0	79	2	3	2	3
milc	183	6	4	0	3	195	0	1	0	0	188	2	1	0	5
sjeng	124	5	1	0	3	133	0	0	0	0	130	1	1	0	1
hmmer	246	24	3	0	2	274	1	0	0	0	254	3	4	3	11
h264ref	508	27	13	1	17	560	0	2	1	3	506	13	9	12	26
gobmk	2560	78	31	1	29	2695	1	2	1	0	1773	278	265	353	30

Table 6: Relative size of GPGs with respect to the size of corresponding procedures in terms of control flow edges.