Table 1: Take Home Performance Results										
Image	Execution Time (usecs)								Percent of Execution Time Used for Commit	
	CPU	Naïve	Shared		Global		Commit Type			
			TRUE	FALSE	TRUE	FALSE	Naïve	Opt	Shared	Global
Lena	16	13.09	4.16	3.84	5.15	3.68	60	9	7.7 %	28.5 %
Cat	169	99.55	12.06	6.4	18.69	10.88	613	68.5	46.9 %	41.8 %
Scope	398	416.8	29.73	12.64	43.74	27.65	1793	186.5	57.5 %	36.8 %
Mountain	1405	1280	187.26	44.26	227.58	69.02	6148.5	590.5	76.4 %	69.7 %

From Table 1 the percent of time spent committing data in private structures to global structures scales such that with larger amounts of data the commit process occupies more than half of the execution time on the accelerator device.

From Figures 1 and 2 the naïve access pattern will be slower than the optimized access pattern specifically when the data is sufficiently distributed meaning that the data spans multiple byte boundaries and multiple private copies cannot be efficiently loaded into the cache memory. The cache loads will be inefficient for the naïve implementation because the data which is required from successive private copies is beyond the boundary which is automatically fetched surrounding the currently requested data. Thus, to get the data from the second private copy requires another fetch from a higher cache level than the optimized access pattern will require.

Access Patterns

Naive

An example with 4 bins and 3 copies

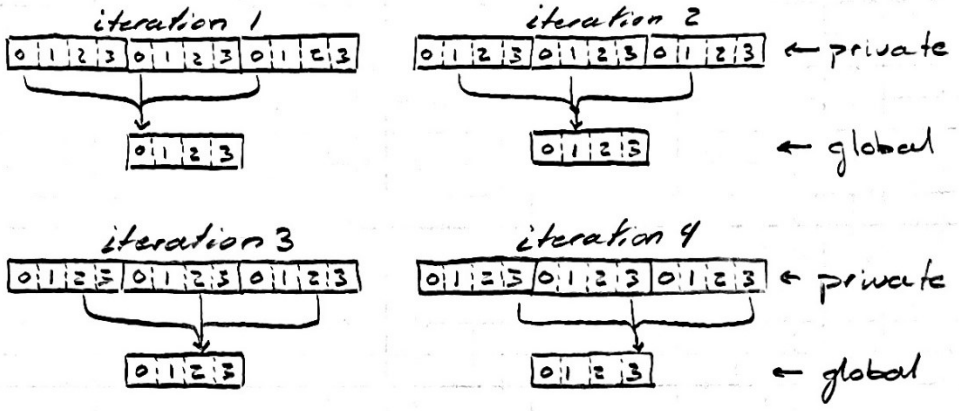

Fig. 1 Naive Access pattern

optimized

An example with 4 bins and 3 copies

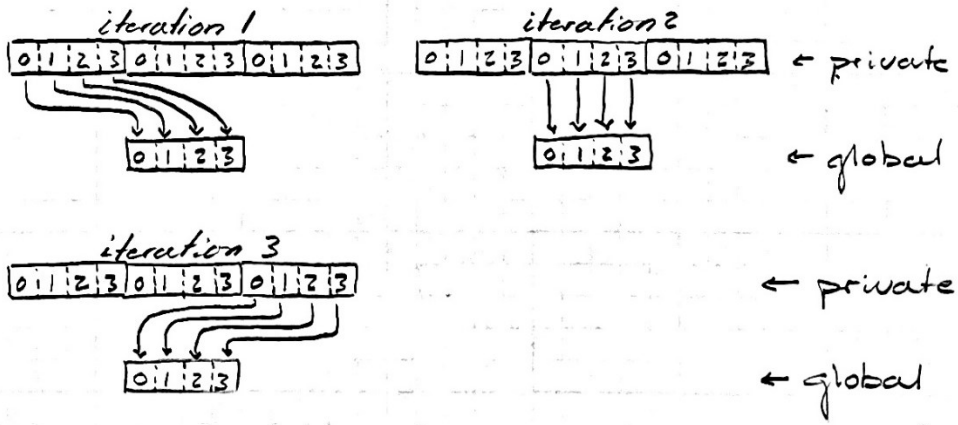

Fig. 2 Optimized Access Pattern