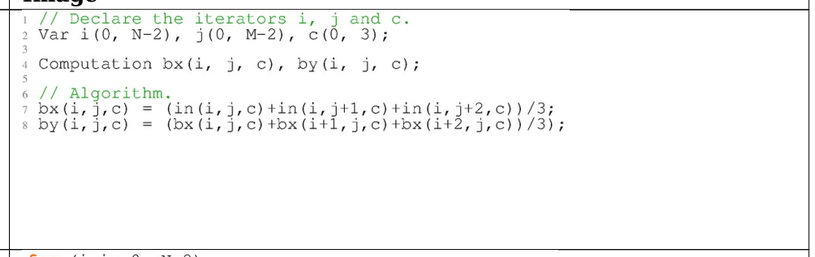
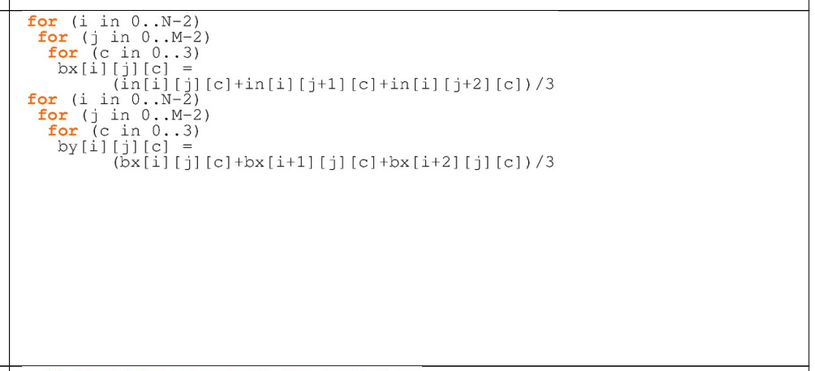
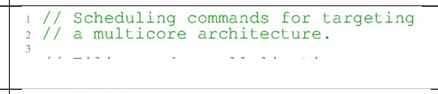
1804.10694v5: image evidence

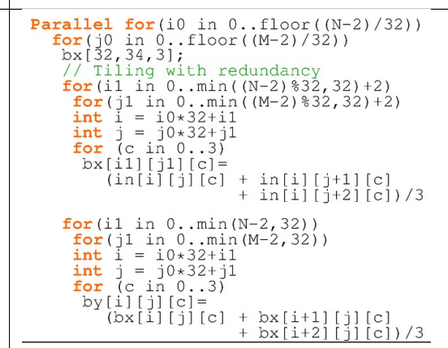
Image regions

Identifier	Page	Conf.	LaTeX source	Rendered	Image

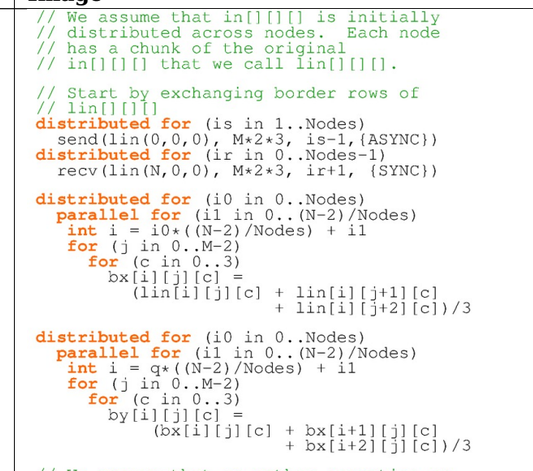
1804.10694v5_DIA_0001	3	—	// Declare the iterators i, j and c.
			Var i(0, N-2), j(0, M-2), c(0, 3);
			Computation bx(i, j, c), by(i, j, c);
			// Algorithm.
			bx(i,j,c) = (in(i,j,c)+in(i,j+1,c)+in(i,j+2,c))/3;
			by(i,j,c) = (bx(i,j,c)+bx(i+1,j,c)+bx(i+2,j,c))/3);
				// Declare the iterators i, j and c.
				Var i(0, N-2), j(0, M-2), c(0, 3);
				Computation bx(i, j, c), by(i, j, c);
				// Algorithm.
				bx(i,j,c) = (in(i,j,c)+in(i,j+1,c)+in(i,j+2,c))/3;
				
				by(i,j,c) = (bx(i,j,c)+bx(i+1,j,c)+bx(i+2,j,c))/3);
					

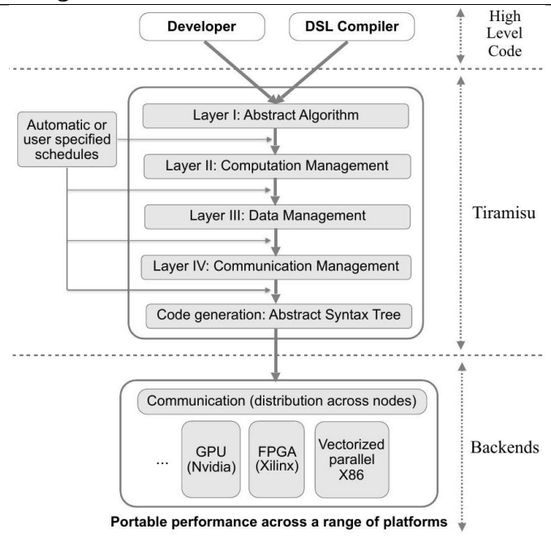
1804.10694v5_DIA_0002	4	—	for (i in 0..N-2)
			for (j in 0..M-2)
			for (c in 0..3)
			bx[i][j][c] =
			(in[i][j][c]+in[i][j+1][c]+in[i][j+2][c])/3
			for (i in 0..N-2)
			for (j in 0..M-2)
			for (c in 0..3)
			by[i][j][c] =
			(bx[i][j][c]+bx[i+1][j][c]+bx[i+2][j][c])/3
				for (i in 0..N-2)
				for (j in 0..M-2)
				for (c in 0..3)
				bx[i][j][c] =
				(in[i][j][c]+in[i][j+1][c]+in[i][j+2][c])/3
				for (i in 0..N-2)
				for (j in 0..M-2)
				for (c in 0..3)
				by[i][j][c] =
				
				(bx[i][j][c]+bx[i+1][j][c]+bx[i+2][j][c])/3
					

1804.10694v5_DIA_0003	5	—	// Scheduling commands for targeting
			// a multicore architecture.
				// Scheduling commands for targeting
				
				// a multicore architecture.
					

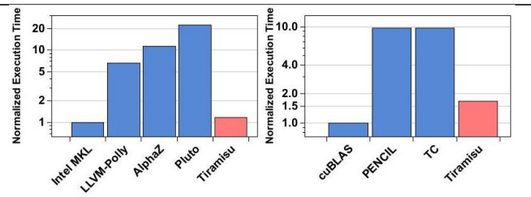
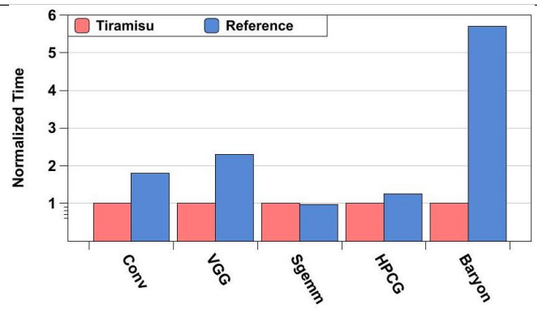
1804.10694v5_DIA_0004	5	—	Parallel for(i0 in 0..floor((N-2)/32))
			for(j0 in 0..floor((M-2)/32))
			bx[32,34,3];
			// Tiling with redundancy
			for(i1 in 0..min((N-2):32,32)+2)
			for(j1 in 0..min((M-2)%32,32)+2)
			int i = i0*32+i1
			int j = j0*32+j1
			for (c in 0..3)
			bx[i1][j1][c]=
			(in[i][j][c] + in[i][j+1][c]
			+ in[i][j+2][c])/3
			for(i1 in 0..min(N-2,32))
			for(j1 in 0..min(M-2,32))
			int i = i0*32+i1
			int j = j0*32+j1
			for (c in 0..3)
			by[i][j][c]=
			(bx[i][j][c] + bx[i+1][j][c]
			+ bx[i+2][j][c])/3
				Parallel for(i0 in 0..floor((N-2)/32))
				for(j0 in 0..floor((M-2)/32))
				bx[32,34,3];
				// Tiling with redundancy
				for(i1 in 0..min((N-2):32,32)+2)
				for(j1 in 0..min((M-2)%32,32)+2)
				int i = i0*32+i1
				int j = j0*32+j1
				for (c in 0..3)
				bx[i1][j1][c]=
				(in[i][j][c] + in[i][j+1][c]
				+ in[i][j+2][c])/3
				for(i1 in 0..min(N-2,32))
				for(j1 in 0..min(M-2,32))
				int i = i0*32+i1
				int j = j0*32+j1
				for (c in 0..3)
				by[i][j][c]=
				(bx[i][j][c] + bx[i+1][j][c]
				
				+ bx[i+2][j][c])/3
					

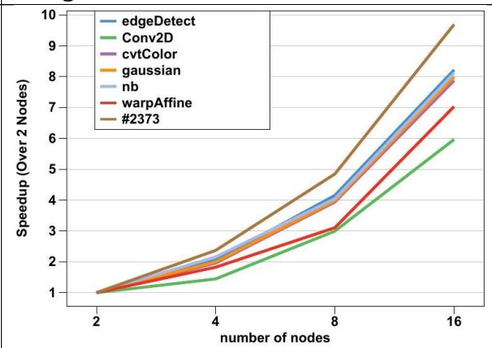
Identifier	Page	Conf.	LaTeX source	Rendered	Image

1804.10694v5_DIA_0005	5	—	// We assume that in[][][] is initially
			// distributed across nodes. Each node
			// has a chunk of the original
			// in[][][] that we call lin[][][].
			// Start by exchanging border rows of
			// lin[][][]
			distributed for (is in 1..Nodes)
			send(lin(0,0,0), M*2*3, is-1,{ASYNC})
			distributed for (ir in 0..Nodes-1)
			recv(lin(N,0,0), M*2*3, ir+1, {SYNC})
			distributed for (i0 in 0..Nodes)
			parallel for (il in 0.. (N-2) /Nodes)
			int i = i0*((N-2)/Nodes) + i1
			for (j in 0..M-2)
			for (c in 0..3)
			bx[i][j][c] =
			(lin[i][j][c] + lin[i][j+1][c]
			+ lin[i][j+2][c])/3
			distributed for (i0 in 0..Nodes)
			parallel for (il in 0..(N-2)/Nodes)
			int i = q*((N-2)/Nodes) + i1
			for (j in 0..M-2)
			for (c in 0..3)
			by[i][j][c] =
			(bx[i][j][c] + bx[i+1][j][c]
			+ bx[i+2][j][c])/3
			// We assume that no gather operation on
			// by[][][] is needed
				// We assume that in[][][] is initially
				// distributed across nodes. Each node
				// has a chunk of the original
				// in[][][] that we call lin[][][].
				// Start by exchanging border rows of
				// lin[][][]
				distributed for (is in 1..Nodes)
				send(lin(0,0,0), M*2*3, is-1,{ASYNC})
				distributed for (ir in 0..Nodes-1)
				recv(lin(N,0,0), M*2*3, ir+1, {SYNC})
				distributed for (i0 in 0..Nodes)
				parallel for (il in 0.. (N-2) /Nodes)
				int i = i0*((N-2)/Nodes) + i1
				for (j in 0..M-2)
				for (c in 0..3)
				bx[i][j][c] =
				(lin[i][j][c] + lin[i][j+1][c]
				+ lin[i][j+2][c])/3
				distributed for (i0 in 0..Nodes)
				parallel for (il in 0..(N-2)/Nodes)
				int i = q*((N-2)/Nodes) + i1
				for (j in 0..M-2)
				for (c in 0..3)
				by[i][j][c] =
				(bx[i][j][c] + bx[i+1][j][c]
				+ bx[i+2][j][c])/3
				// We assume that no gather operation on
				
				// by[][][] is needed
					

Identifier	Page	Conf.	LaTeX source	Rendered	Image
1804.10694v5_DIA_0006	7	—	—	—	
1804.10694v5_DIA_0007	9	—	`for(q in 1..N-1) {...} // distribute on q`	`for(q in 1..N-1) {...} // distribute on q`	`for(q in 1..N-1) {...} // distribute on q`
1804.10694v5_DIA_0008	9	—	`q = get_rank(); if (qP1 and q<N-1) {...}`	`q = get_rank(); if (qP1 and q<N-1) {...}`	`q = get_rank(); if (q≥1 and q<N-1) {...}`

1804.10694v5_DIA_0009	10	—	`/* Ring Blur Filter */`
			`R(i,j) = (Img(i-1,j-1) + Img(i-1,j) + Img(i-1,j+1)+`
			`Img(i,j-1) + Img(i,j+1) +`
			`Img(i+1,j-1) + Img(i+1,j) + Img(i+1,j+1))/8`
			`/* Roberts Edge Detection Filter */`
			`Img(i,j) = abs(R(i,j) - R(i+1,j-1)) +`
			`abs(R(i+1,j)- R(i,j-1))`
				`/* Ring Blur Filter */`
				`R(i,j) = (Img(i-1,j-1) + Img(i-1,j) + Img(i-1,j+1)+`
				`Img(i,j-1) + Img(i,j+1) +`
				`Img(i+1,j-1) + Img(i+1,j) + Img(i+1,j+1))/8`
				`/* Roberts Edge Detection Filter */`
				`Img(i,j) = abs(R(i,j) - R(i+1,j-1)) +`
				`abs(R(i+1,j)- R(i,j-1))`
					`/* Ring Blur Filter */`
					`R(i,j) =(Img(i-1,j-1) + Img(i-1,j) + Img(i-1,j+1)+`
					`Img(i,j-1) + Img(i,j+1) +`
					`Img(i+1,j-1) + Img(i+1,j) + Img(i+1,j+1))/8`
					`/* Roberts Edge Detection Filter */`
					`Img(i,j) = abs(R(i,j) - R(i+1,j-1)) +`
					`abs(R(i+1,j)- R(i,j-1))`

1804.10694v5_PIC_0001	1	—	—	—	
1804.10694v5_PIC_0002	9	—	—	—	

Identifier	Page	Conf.	LaTeX source	Rendered	Image
1804.10694v5_PIC_0003	11	—	—	—	

1804.10694v5_LST0003	5	—	// Scheduling commands for targeting // a multicore architecture.
			// Tiling and parallelization. Var i0, j0, i1, j1; by .tile(i, j, 32, 32, i0, j0, i1, j1); by.parallelize(i0); bx.compute_at(by, j0);
				// Scheduling commands for targeting // a multicore architecture.
				// Tiling and parallelization. Var i0, j0, i1, j1; by.tile(i, j, 32, 32, i0, j0, i1, j1);
				by.parallelize(i0); bx.compute_at(by, j0);
					—

1804.10694v5_LST0004	5	—	Parallel for(i0 in 0..floor((N-2)/32)) for(j0 in 0..floor ((M-2)/32)) bx[32,34,3]; // Tiling with redundancy for(i1 in 0..min((N-2):32,32)+2) for(j1 in 0..min((M-2)%32,32)+2) int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) bx[i1][j1][c]= (in[i][j][c] + in[i][j+1][c] + in[i][j+2][c])/3 for(i1 in 0..min(N-2,32)) for(j1 in 0..min(M-2,32)) int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) by[i][j][c]= (bx[i][j][c] + bx[i+1][j][c] + bx[i+2][j][c])/3
			host_to_device_copy(in_host, in); GPUBlock for (i0 in 0 ..floor (N-2) / 32)) GPUBlock for (j0 in 0..floor ((M-2) / 32)) shared bx [3,32,34] ; // Tiling with redundancy GPUThread for(i1 in 0..min \((((\mathrm{N}-2) \% 32,32)+2)\) GPUThread for(j1 in 0..min \((((\mathrm{M}-2) \% 32 ,32)+2)\) int i = i0*32+i1 int j = j0*32+j1 for (c in 0 ..3) bx[c][i1][j1]= (in[i][j][c] + in [i][j+1][c] + in [i] [j+2] [c]) / 3 GPUThread for(i1 in 0..min(N-2,32)) GPUThread for(j1 in 0 ...min(M-2,32)) int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) by[c][i][j]= (bx[c][i][j] + bx[c][i+1][j] + bx [c] [i+2] [j]) / 3 device_to_host _copy(by, by_host);
				Parallel for(i0 in 0..floor((N-2)/32)) for(j0 in 0..floor((M-2)/32)) bx[32,34,3]; // Tiling with redundancy for(i1 in 0..min((N-2):32,32)+2) for(j1 in 0..min((M-2)%32,32)+2) int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) bx[i1][j1][c]= (in[i][j][c] + in[i][j+1][c] + in[i][j+2][c])/3 for(i1 in 0..min(N-2,32)) for(j1 in 0..min(M-2,32)) int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) by[i][j][c]= (bx[i][j][c] + bx[i+1][j][c] + bx[i+2][j][c])/3
				host_to_device_copy(in_host, in); GPUBlock for (i0 in 0..floor (N-2) / 32)) GPUBlock for (j0 in 0..floor ((M-2) / 32)) shared bx [3,32,34] ; // Tiling with redundancy GPUThread for(i1 in 0..min $((\mathrm{N}-2)\%32,32)+2)$ GPUThread for(j1 in 0..min $((\mathrm{M}-2)\%32,32)+2)$ int i = i0*32+i1 int j = j0*32+j1 for (c in 0..3) bx[c][i1][j1]= (in[i][j][c] + in [i][j+1][c]
					—

1804.10694v5_LST0008	10	—	/* Ring Blur Filter */ R(i,j) = (Img(i-1,j-1) + Img(i-1,j) + Img(i-1,j+1)+ Img(i,j-1) + Img(i,j+1) + Img(i+1,j-1) + Img(i+1,j) + Img(i+1,j+1))/8 /* Roberts Edge Detection Filter */ Img(i,j) = abs(R(i,j) - R(i+1,j-1)) + abs(R(i+1 ,j)- R(i,j-1)) edgeDetector creates a cyclic dependence graph with a cycle length \(\geq 1\) (R is written in the first statement and read	/* Ring Blur Filter */ R(i,j) = (Img(i-1,j-1) + Img(i-1,j) + Img(i-1,j+1)+ Img(i,j-1) + Img(i,j+1) + Img(i+1,j-1) + Img(i+1,j) + Img(i+1,j+1))/8 /* Roberts Edge Detection Filter */ Img(i,j) = abs(R(i,j) - R(i+1,j-1)) + abs(R(i+1,j)- R(i,j-1)) edgeDetector creates a cyclic dependence graph with a cycle length ≥ 1(R is written in the first statement and read	—