What is Driving Programming System Technology for Exascale and Beyond

Mary Hall September 11, 2019

Collaborators and Acknowledgements

Stencils, Bricks and Geometric Multigrid

Protonu Basu (Facebook), Tuowen Zhao, Sam Williams, Brian Van Straalen, Lenny Oliker, Phil Colella, Hans Johansen

Autotuning Search and Pragma Autotuner

Prasanna Balaprakash, Paul Hovland, Vinu Sreenivasan, Rajath Javali

LLVM and Polly Optimization

Michael Kruse, Hal Finkel, Vinu Sreenivasan

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

This research used resources in Lawrence Berkeley National Laboratory and the National Energy Research Scientific Computing Center, which are supported by the U.S. Department of Energy Office of Science's Advanced Scientific Computing Research program under contract number DE-AC02-05CH11231.









Which version would you prefer to write?



Code A: miniGMG baseline smooth operator approximately 13 lines of code

256((__n256i*)(RedBlackHask+ij+ 4)))); 256((__n256i*)(RedBlackWask+ij+ &)))); 56_load_si256((__m256i*)(RedBlackHask+ij+12)))); Spin locks in OpenMP rev. 12 = _m256_blendv_pdfphi_12,nev_12,RedBlack_12" m256 store pd/phi+iik+ 0.nev (0); _m256_store_pd(phi+ijk+ 4,nev_04) _m256_store_pd(phi+iik+ 4,new_00); _m256_store_pd(phi+ijk+12,nev_12); **Code B:** miniGMG optimized smooth operator approximately 170 lines of code

helmholtz_00 = _m256_mul_pd(helmholtz_00,b_h2inv_splot4);

M = m256 mil od'heloholtz 84 h h2inv s

m256 load ad(rhs+iik+ 8))):

n256_load_pd(rts+ijk+12)));

(_n256i*)(RedBlackWask+ij+ ())));

And now GPU code?

elfdef VL	double * _ beta_k = gpu_subdomains[box].levels[lavel].grids[beta_k] + (plans+pencil+1);	if(threadids.x++0) temp[threadids.y+1][0] = pti[i]k+1];
##orning using vectorized smooth)	double * lombda = ģpu_subdomains[box].levels[level].ģrids[lombdd] + (plane+pencil+1); #endif	if(threadids.y==0) temp[0][threadids.ss1] = phi[i]x-pencil]; if(threadids.x==0) temp[threadids.ys1][THDIMX+1] = phi[i]x+THDIMX];
double b2(nv = 1, 2/(0*5); 19 hov = hore(fit = v / fitha 4 1)	shared double hera i six[TRDIW_][[RDIW_1]]-	<pre>if(threadIdx.y==0) temp[T80104-1][threadIdx.x+1] = phi[i]k+T801047*pencil]; bets_i_i]k(threadIdx.y_][threadIdx.x_] = bets_i[i]k];</pre>
int ubdradin dim = dou ubdradinifon].leveli[devel].dim.i:		<pre>if(threadIdx.x==0)bets_1_i[s[threadIdx.y_]] TBODMX] = bets_1[i]s+TBODMX]; bets_1_i[s[threadIdx.y_]][threadIdx.y_] = bets_1[i]s];</pre>
int pencil = gpu_subdomains[box].levels[level].pencil; int plane = gpu_subdomains[box].levels[level].plane;	double beta_k_1j%,beta_k_1j%p1; double = \$m1.1j%,p1.j%p1,pm.ij%m1;	i/(threadidx.y==0)teta_j.j[x] TBUDW][threadids.x] = beta_j[t]%TBUDW*percil]; syncthreads(): // ANN seers
// gancliiJ3frideiganclii??i h= 13fride(V)	// when the publication is recommended to recilize them a thread block resticts theorem was's receil their secul	if (within Bounds)/
int 1; - ((I)Stride*BlockIdx,*,epencil pencil) & -WeW) + threadIdx,x; // i.e. shift vector so that thread0 is 128-byte aligned //int i = 1; B pencil; // pencil is even and I will AND off all but the ISB	<pre>// whit the set it withinfounds = 1; if(is) which is a set in the set in the set is a set in the set is a set in the set is a set in the set in the set is a set in the s</pre>	double helmholiz,ijk = drolpholijk)*phi,ijk - b*h2inv*(heto i iSilthreadida,v][threadida,val] * (temolthreadida,val]*threadida,val] - shi ijk) -
int $i = i$] & Bat; // pencil is even and I will AND off all but the LSB int $j = i j$ / pencil:	if(j>=subdomsin_dim)withinSounds = 0;	bets.i.i.Stitreadidx.y [[threadidx.s] + (phi.i.S - temp[threadidx.y4][threadidx.y4][threadidx.s]) + here i iStitreadidx y4][threadidx s] + (seen[threadidx y4][threadidx y4]] = th
int x;	k = 0; int ijk k^* plane + j*pencil + i;	bets_j_ip[threadidx.y][threadidx.x] + (phi_jp - temp[threadidx.y][threadidx.s+1]) + bets_ipitat
_shoreddouble * sht: _shoreddouble * rhs;	ph_ijph = ph[i]k plon]; ph_ijk = ph[i]k =	intractifie (picifie) (picifie)
sArred double • slpha; sArred double • sleta::	ph:[jps:=ph:[jkplas]; beta,kj% - beta,kj%];	// (SNB double new abi u shi i% - lambdo[i%]*/selsholty i% - rhs[i%]):
storeddouble * Deta_K: thoreddouble * immda;	beta_k_1jkg1 = beta_k[jk+plone];	phi[ijk] = (RedBlackUpdote) ? new_phi : phi_ijk;
if(threadidx.x==0){ phi= gpu_subcompins[box].levels[ievel].grids[phi_id] + [plone];	in wantercover - (c-)-meny = ext.	//if/withinBounds && (fRedBlackladote/VIBBul))/
elpha = ppu_subdemeins[box].levels[level].grids[olpha] + (plume): bets.l = ppu_subdemeins[box].levels[level].grids[bets.l] + (plume):	syncthreads(); // #AR guard	// phi[ijk] = phi_ijk - lambda[ijk]*(helmhaliz_ijk - rhs[ijk]); // GSNB //]
beta_i = gpu_subdomains[box].levels[evel].grids[beta_] + (plane); beta_k = gpu_subdomains[box].levels[evel].grids[beta_k] + (plane);	// cory prices setup; its source term velocit, x+1) = phi_5(x; if threads x=a) = terms(dx; x+1) = phi_5(x; 1)	//double new_phi = phi_ijk - lambda[ijk]*(helmholtz_ijk - nhs(ijk)); //if(withinSounds)/
temoda = gpu_subodmini[box].teveis[tevei].grids[temsdd] + (piono); 5.wxr/Themosfy)	if(threadids.y==0) temp[= 0[threadids.y=1] = ph[[jk=ph(zk]); if(threadids.y==0) temp[threadids.y=1] [ThOMA:] = ph[[jk=TROMA];	// phi[ijk] = ((Medilackipdete*k)&0x1) ? new_phi : phi_ijk; //}
relie double * phi = gpu_subdomains[box].levels[level].grids[phi_id] + (plane);	if(threadids.y=0) temp[TDSIM+1][threadids.x+1] = ph[[jk+TDSIM*penci]; beta_lijk[threadids.x][threadids.x] = beta_l[1]x[j]	RedBlacklpdote=RedBlacklpdote*1;
double nrs - gpu_subdomoins[bax].levels[level].arids[nrs_id] + (plane); double alpha = gpu_subdomoins[bax].levels[level].arids[- alpha];	i (integrada, vende da) (integrada) menada) = beta (integrada) if(integrada, vende da) (integrada) = beta (integrada)	tjk+plane; // rotate register stpaline
double · bets_i = gpu_subdomoins[box].levels[level].griddbets_i + (plane); double · bets_i = gpu_subdomoins[box].levels[level].griddbets_k + (plane);	Syncthreads(); // KAW guárd	phi_ijkml = phi_ijk; phi_ijk = phi_ijkpl;
double * lambda = gpu_subdomains[box].levels[level].grids[lambda] + (plane); #endif	if(kithidouds)! double nelemoiz;jik = d*aleha[ijk]*phl.jk= = b*h2im*(double nelemoiz;jik = d*aleha[ijk]*phl.jk= - b*h2im*(phi_ijkp1 = phi[ijkkolone]; beto_k_ijk = beto_k_ijkp1;
shareddouble beta_L_11k[YL]:	beta.Likithreadida.y threadida.x + (abi.[ik] beta.Jikithreadida.y+1[threadida.x + (abi.[ik]) + (betatia.y+1](threadida.y+1](threadida.y+1](threadida.y+1)	<pre>beta.k.ijkp1 = beta.k[ijk+plane];</pre>
shored double heto_j_ijs(V_j) _shored_ double teop(v_j)	beta_j_ij(threadIdx.y][threadIdx.x] { ani_ik - test[threadIdx.y][threadIdx.x+1] + beta.k_ijka - sni_ika - sni_ika - sni_ika	} // for k } // GSR8 kernel
ddubis petd.x.1;%.eetd.x.1;%elphi_tjkm1; ddubla phi_tjka, phi_tjkp1 phi_tjkm1;	para_k_ijk (pni_ijk -pni_ijkmi)) (come	fela //
<pre>// when the subdomain is coarsened to smaller than a thread block, certain threads won't commit their result</pre>	 double new_phi = phi_ijk - lambd(jk)(Chelmholtz.jk - nhs[ijk]); phi[ijk] = (Kediscupdata) ? new_phi : phi_ijk; 	// cache version (relies solely on the 11/12 cache hierorchy)
if (threadiax.x = pencil) MithiaGunds = 0; if (threadiax.x = pencil+JStrie) MithiaGunds = 0;		global vaidsmooth_once_GSH(subdomain_type * gpu_subdomains, int phi_id, int rhs_id, double &, double &, double k, int sweep, int level){
<pre>iii (* pencii* if () >pencii*(subdomain_dir+1)-1) #iiinisdomas = 0; iii () >pencii*(subdomain_dir+1)-1) #iiinisdomas = 0;</pre>	//fylitinsouds.kk.(teksidouppotexjAks))/ // pill(jk) = ph.[i](/helimoliz_jjk - rhs[ijk]); // GSRs	double h2(nv = 1,0/(h*h); (nt box = blockIds.z; // CUDA 4
k = 0:	//double new_phi = phi_ijk - lambd(ijk)*(helmholtz_ijk - rhs[ijk]); //if(withinBound);	int 1,j,k; // (0,0,0) = first non ghost zone element 1 = TBO19X*blockIdx.x + threadIdx.x;
int ijk = 13 + krylobe; abi ijkw = abiijk-nlanel:	// psl[ijk] = ((RedBlockUpdote-k)ABk1) ? nem_pni : phi_ijk; //)	j = TBDIM"BlockIds.y + threadIds.y;
phi_is = phi[is]; phi_isot = phi[isot = phi];	RedBlackUpdate=RedBlackUpdate^1; 1jk==plone;	int subdomain,dim = gpu_subdomains[box].levels[level].dim.i; int pencil = gpu_subdomains[box].levels[level].pencil;
8484_L1X = 8484_K1X btta_L15pi = btta_K(1x)	// rotate register pipeline ph.ljkkg = ph.ljkt	int plane = gpu_suddandins[box].levels[level].plane;
int RedBlockUpdste = (1^j^iweep)&Ex1	philis = philipping); philis = philipping);	// construct pointers to element (W,W,W) in each orray for the current suddomain mifdefPEUMTERS_W_SERED
for(k=0;k <subdomain_dim;k++)[< td=""><td>bata_k_ijapi = bata_k[ija+plane];</td><td></td></subdomain_dim;k++)[<>	bata_k_ijapi = bata_k[ija+plane];	
iyncthreads(); // WAR guard temp[threadidx.x] = phi_ijt;	} // for k } // Sits kernel #alse	
<pre>if(Cthreadiax.x=pect1)Ak(threadiax.xcpanct1+13xtride+1) Detalik(threadiax.x] = beta[i]x[: if(cthreadiax.x=pect1)</pre>	////////////////////////////////	
if(withinBounds){	globdl voidsmooth_once_GSRB(subdomain_type * gpu_subdomains, int phi_id, int rhs_id, double a, double b, double h, int sweep, int level){	<pre>if((inredidx.vw0)k(inredidx.yw0)){</pre>
double nelmoltz_ijk = d*ulphe[jk]=ph_ijk = b*nZinv*(beta_ijk[predda.* + 1] = (tengihreolds.* + 1] = phi_jk =	double h2inv = 1.0/(h*h); int box = blockIdx.1; // (UDA 4 111	rhs = gpu_subdents[box].tweis[evel.grids] rhs_id] + (plorespective); rhs = gpu_subdents[box].tweis[evel].grids[rhs_id] + (plorespective); alabe = au undentsis[box].tweis[au].grids] = rhs_id] + (plorespective);
betaiişk[threadidx.x=pencii] • { temp[threadidx.x=pencii] • phi.];k 5 · Betaiişk[threadidx.x] • { phi.];k • temp[threadidx.x=pencii] } +	int $i, j, k; // (0, 0, 0) = first non ghost zone element i = TBOINKPlockBAx. H threadBax. Since the element i = TBOINKPlockBAx. The threadBax. The threadBax is the thread thr$	beto_i = gpu_subdomins[box].levels[level].grids[beto_i] + (planexpencilvi); heto i = opu_subdomins[box].levels[level].grids[beto_i] + (planexpencilvi);
Deta_K_15pp1 • [ph.15p1 - ph.15k] - beta_K_15p1 • [ph.15pk - ph.15m])	j = index = Subdemoliants, y = Limited empirical states and the subdemoliant states an	beta,k = gpu_subdomains[box].levels[level].grids[beta,k] + (plane+pencil+1); lombda = gpu_subdomains[box].levels[level].arids[lambda] + (plane+pencil+1);
<pre>// GS88 double new_phi = phi_ijk - lambda[ijk]*(helmholtz_ijk - rhs[ijk]);</pre>	int peneti = gpu_subdomsins[box].leveli[tevel].peneti; int plane gpu_subdomsins[box].leveli[tevel].plane;	_pyncthreads();
ph[[jk] = (KedBlackUpdate) ? new_phi : phi_ijk; }	// construct politers to element $(0,0,0)$ in each array for the current subdomain error. DUBTERS is verticed.	Falsa double * _ phi = gpu_subdomsins[box].levels[level].grids[_phi_id] + (plane+pencil+1);
RedNackUpdate=RedNackUpdate^1; (]%==)lane;	_shareddouble * sht: _shareddouble * nh;	double * rhs = gpu_subdomains[box].levels[level].grids[rhs_id] + (planexpencil+1); double * olphs = gpu_subdomains[box].levels[level].grids[s[sho] + (planexpencil+1);
<pre>// rotate register pipeline philipki = philipki </pre>	_shoreddouble * olpho; _shoreddouble * beta_t;	double ' beta_i = gpu_subdomains[box].levels[level].grids[beta_i] + (plane+pencil+1); double ' beta_j = gpu_subdomains[box].levels[level].grids[beta_j] + (plane+pencil+1);
phi_(1) = pri(1)+pi(nn); bt(1, 1) = bt(1, 1) = bt(1) =		double ' beta,k = gpu_subdomains[box].levels[level].grids[beta,k] + (plane+pencil+1); double ' lombda = gpu_subdomains[box].levels[level].grids[lambda] + (plane+pencil+1);
beta_k_jkpl = beta_k(tjk+plane);	if([threadidx.i===9kk[threadidx.y==#0]){ phi = gu_subdemins[@xx]ivels[[evt].grids[phi_id] + (plane+gencil+1);	Redif
} // GSRS kernel	rms = gww_swawnmuns(swa,i.versis;[eVel],grids[_rd5], rd5_10] + (ploHetpOnt[i=1]) alpha = gpu_subdomains(box).levels[ievel],grids[alpha] + (plane+pencil+1); beto.i = dwu_subdomains(box).levels[ievel].grids[_beto.i] + (plane+pencil+1);	// when the subdomein is corrected to smaller than a thread black, certain threads won't commit their result
#else #ffdefLDCALITY_VIA_SHARED	<pre>beta_j = gpu_subdomains(bax).levels(level).grids(beta_j) + (plane+pencil+1); beta_k = gpu_subdomains(bax).levels(level).grids(beta_k) + (plane+pencil+1);</pre>	Int withings = 1; If()>=subdomsin,fis)=(1):Storeds = 0;
// shared+11 version (relies on shared memory but you must still favor:1)	" lambda = gpu_subdomins[box].levels[level].grids[lambda] + (plane+pencil+1);] "	<pre>tr(p+suddmdin,otn)withindownds + w;</pre>
global voidsmooth_ence_GSRB(subdomain_type * gpu_subdomains, int phi_id, int rhs_id, double a, double b, double h, int sweep, int level){		(at RedBlackIndete - (1)()(resear) 1 det)
wearing matrix a sector matrix	#else double − phi = gpu_subdomdins[box].levels[level].grids[phi_id] + (plane+pencil+1);	int RedilickUppdte = (i^jsmeed) & dkl; Conficitiversholmente Alexies V.
int box = blockids.2; // Coxe = 11 int 1.1k: // (B.0) = (int non dboxt zone element	file follot pli « pu_iuddomini(box).level](revel,gridi; pli_id) + (planespanci)+1); follot ns « pu_iuddomini(box).level](revel,gridi; nis.(d) + (planespanci)+1); follot ns = apu.iuddomini(box).level](revel,gridi; nis.(d) + (planespanci)+1); follot ns = apu.iuddomini(box).level](rev	111 Reflectpoors = (1')=ee() & Rd; for(ioi);cusdoons(,dir(i+)) int (i) = kibine + y=bendi = 1;
int dok = Diotclast.; // close = [i] nt [] // (B_1)@ = [i] first non ghost zone element 1 = TIDIXXTPIOCCLAST, + threadIax.x; = TIDIXYTPIOCCLAST, + threadIax.x;	Alia Participation Participation 001 Participation Participation Participation	In manufacture of the second sec
111 [1,2,1/2](2,0,0) - ('1''') ang ipan zong elment 1 - 1000/9711012(x, - 1'') ('2,0,0) - ('1'') ('1''') ('1'') ('1'') ('1''') ('1''') ('1'') ('1'')	nis	In Mallocapae. (*)(*)(*)(*) At() (*) () + (*)(*)(*)(*)(*)(*)(*)(*)(*)(*)(*)(*)(*)(
<pre>tit i, j, j, j/(d, d) = /(triangle) as zet element i = 100 xylisite.i = (triad(d, t)) = 100 xy</pre>		The Mathematic (1) ((1) ((1))
<pre>Nit [1, 1///CRAG] - fritan part zets filters Nit [1, 1///CRAG] - fritan part zets filters Nit [1, 1///CRAG] - fritan part zets filters Nit parters.filt - gr.patersingSoci.leti[(evi].det.]; Nit parters.filt - gr.patersingSoci.leti[(evi].det.]; Nit parters.filters.fi</pre>	<pre>ref:</pre>	In Mallocapae. (//)weary A Mc; free/a-management (//)weary A Mc; free/a-management (//) / ///////////////////////////////
<pre>iii ii j, j // (d, d) = / triane jest zew element i ii j, j // (d, d) = / triane jest zew element i ii j // (d, d) = / triane jest zew element ii ii j // (d, d) = / triane jest zew element ii ii j // (d, d) = / triane jest zew element ii j // (d, d) = / triane jest zew element ii j // (d, d) = / triane jest zew element i j / triane jest zew elemen</pre>	<pre>refs refs r</pre>	In Mail(content) (((((((((((((((((((((((((((((((((((
<pre>Nit [1, 1, 1/2] (2, 1/2] - / (1/2, 1/2) - / (1</pre>	<pre>rise</pre>	In Markagene ((')=mail) A ki; '(')=('-k-acasene ((')=mail) A ki; '(')=(')=(')=(')=(')=(')=(')=(')=(')=(')
<pre>Nit [</pre>	<pre>ref: r</pre>	In Maticapane. (*)*(*)*(*) (*)*(*)*(*)*(*)*(*)*(*)*(*)*(*)*(*)*(*)
11 1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.	<pre>File File</pre>	<pre>the Maticogene. (c)/mem() A Mc; for d-avaaanse / reveare / for maticogene. (c)/mem() A Mc; for d-avaanse / reveare / for maticogene. (c)/mem() A Mc; maticogene. (c)/mem() A Mc; maticogene. (c)/mem() A Mc; maticogene. (c)/mem() A Mc; mem() A Mc; mem() A Mc; for any and avaal / for any and avaal / for mem() A Mc; for any and avaal / for any any any any any any any any any any</pre>
<pre>11 (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</pre>	<pre>ris ris ris ris ris ris ris ris ris ris</pre>	In Markongan, (*)*(*)*(*) A ki; *(*)*(*)*(*)*(*)*(*)*(*)*(*)*(*)*(*)*(*
<pre>11 (1 , 1 , 1 / (2 , 4) / (2 , 1 / (2 , 1) / (2 , 1 , 2 , 1) / (2 , 4 , 2 , 1 / (2 , 4 , 3) / (2 , 1 , 2 , 1) / (2 , 4 , 3) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1) / (2 , 1 , 2 , 1 , 2 , 1) /</pre>	<pre>File File</pre>	<pre>the Maticogene (://wear) A Mc; /// Contact A Maticogene (:///wear) A Mc; // Contact A Maticogene (:////////////////////////////////////</pre>
<pre>[10] [11] [12] [12] [12] [12] [12] [12] [12</pre>	<pre>ris ris ris ris ris ris ris ris ris ris</pre>	<pre>Int Maticipant: (')'(')'(')'(') (') (') (') (') (') (')</pre>
<pre>11 [1, 1, 7, 7, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,</pre>	<pre>ris ris ris ris ris ris ris ris ris ris</pre>	In Markovski, (* 1999) A (* 1999) (* 1997) A (* 1997) A (* 1997) A (* 1997) Markovski (* 1997) Markovsk
<pre>11 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1</pre>	<pre>Aligned to the set of the se</pre>	<pre>http://www.jt.ki; //wikawa.jt.ki; //wikaw</pre>

Code C: miniGMG optimized smooth operator for GPU, 308 lines of code for just kernel

Goal of Research



Programming system derives

Code B (CPU)

Code A

for (k=0; k<N; k++) for (j=0; j<N; j++) for (i=0; i<N; i++){ if ((i+j+k+color)%2 == 0)

for (k=0; k<N; k++) for (j=0; j<N; j++) for (i=0; i<N; i++

/* Helmholz */ for (k=0; k<N; k++) for (j=0; j<N; j++) for (i=0; i<N; i++)

temp[k][j][i] = b * h2inv * (

temp[k][i][i];

Also, Codes D, E and F....

(temp[k][i][i] - rhs[k][i][i]);}

		- O stated and find the
panet, who cannot and a second a second seco	842 ¹	(Constanting and Constanting) (Constanting (
All Sec Manifeld J., "If the All II. In annexis (III., particular), (Intelligent), Ann. I.	- Dense, Andre Mari, 1978 - Carlor (1980) - Dense, Andre Mari, 1978 - Carlor (1980) - Dense, Andre Mari, 1978 - Carlor (1980)	1(mate-effec.)(0)mate- 86(0)mate- 86(0)mate- 1(0)
A THE R. CONTROL OF CO	BATE PLUS, PLUM, PLUM	"Statement" 1 of Res. [Lowers 1 - art's (Lowers 2)".
AN ADDRESS OF ADDRESS OF ADDRESS A	1 Mar H Andrews S Spectra & Andrews S 1998 Fight, 2012 1008 APA 1000 100 100 100 100	The second secon
	1 () Constanting () () () () () () () () () () () () ()	Harrison and the second
100 - 100 - 100	E10*1202244	2017 103* 23A
	and the second s	ALL AND THE ADDRESS AND ADDRESS ADDRES
Const. Barris - Carlos Martineza - Carlos Martineza - Carlos	The Reference of Children (Children) (Children)	
A 10000 0000 00 24 000	(Antering of the second s	
NAME AND ADDRESS OF A DECEMPTION OF A DECEMPTI	timeta org	The set of the second s
And a second sec	100000 1-0 100	A CONTRACTOR OF A CONTRACTOR OFTA CONTRACTOR O
	Constant of State (1) The Constant of Cons	Contraction -
	PROPERTY AND ADDRESS OF ADDRESS ADDRES	
-004. 800 300 11000	BALL DOWNERS & DOWNERS - DOWNERS AND ADDRESS AND ADDRESS ADDRE	March (March (March))
	And a second sec	1 Contraction
HAR SPICE PRODUCT AND A STREET AND A STREET, AND A STREET AND AND AND AND AND AND	The second secon	
A DESCRIPTION AND A DESCRIPTIO	and an and a second second second	"Tomo" and "man but the contraction and a first second of the second second second second second second second
N. C. Hanni Franzens, Jones I. December 8.	 product - product - induction control and a control of the induction of the in	No. 2010 Control of Co
an (b = 1) a regime.	2. 40(3)(-10400000000000000000000000000000000000	- REFERENCE - FRANK -
	Audit up hydroxy Audit un hydroxy 1, h Audit Audit Audit un hydroxy 1,	
Record Constant Const	E.C. E.B.	T AND A REAL ADDRESS ADDRE
And the Association of Association of Association and As	1010 Sec 1000 (Plane)	100.00110
(1) Direction or and (increasing converticity and increasing the AUSIAN STREET STREET,	Contraction of the second seco	
Annual of the part	- and drive links with a de cris and taxeful gran. Art. derivation (Marines) og 1 gr. shamers, in pr. 4, 10 hr. 6, 2011 1, 2011 1, 2011 1, 10 hr. 11 1011	
Manual Contraction of the Contra	And a close of a particular sector of the se	5 - 2 - 2000 10 - 2010 12 - 5.2 - 5000010
NULL CONTRACTOR AND A C	1 Million Control of Control o	B. B. SHARES MICH. Phys. Rev. B 10000011
l an		(and - (a), and only (and (and (and - (and only (and)))))))
and of a line linear and a second second	CAPTAL ADDRESS AND ADDRESS	man and a second condition of the second sec
Constraints (constraints)		and an example roles and an example
		HART * HEL - DO LANDER NO. HEL CALL AND A CONTRACT OF A CO
BALLINE - BALLINGER		
		2 and its sampler is serviced in address for a little field, proof thead and sample reactions.
	The second secon	
pert, with construction definition of a production of an origin of the result water to make to the second s		A CONTRACTOR OF A CONTRACTOR O
No. 2 And A Control of		in product (see in)
an anter an anter the local and the local and	101 - 114 - B. (1000) (101 (101) (101) (101) (101)	Market Control of Cont
IN YORE I BELIEBERGERE HERDERE BERGERE BURG. 2 wennet minste in Kannet Burder (Burd) is met wenn for the secret antibility conservation conservation conservations and an	If the is allowed a provide a set of the Cried State, senar implementation and the registrone	
And Address of the Ad		80.000 etc 2010
Contraction and the Contraction of C	An Instructional Conference Conference Conference	
		800 900 - 0.0 January (Charles Charles
The second secon	Part of a research of Academic	And and the product of the second sec
	The second	Bell and and a set of the set
		 Belle and variable and conservation with the model of the second second second second second second and the second second second second second second second second and the second sec
		Microsoftware and a management of the second

Theme 1: Performance Portability

Can the same program perform well on diverse supercomputing platforms? (e.g., Top 500 list, top500.org)



#1: Summit, IBM Power9+V100 GPUs



#3: TaihuLight, Sunway



#6: Piz Daint, Intel Xeon+P100 GPUs

#8: ABCI Intel Xeon Gold And V100 GPUs



#4: Tianhe-2, Intel Xeon Phis



What's Coming Next?



Fugaku (Riken), ARM + custom optimizations



Aurora, Intel Xeon + Intel X Compute



Frontier, AMD EPYC CPU + AMD GPU

Theme 2: Data Movement



Communication wall will get worse (dominates energy and time)

- Optimizing for memory/network more important than ever
- Automatic data movement (caches, VM) can be wasteful
- Autotuning (search) helps reach bandwidth limits

Slide source: Kathy Yelick, UC Berkeley, More Data, More Science and... ! Moore's Law, 2015.

Stencil Computations

- Solve partial differential equations
 - Points are computed using neighbors
- Low order stencil
 - Lower accuracy
 - Low arithmetic intensity (FLOP per byte) typically memory bound

• High order stencil

- High arithmetic intensity and could be compute bound
- Conventional wisdom: memory optimization is not as important
- Diameter of stencil related to order of stencil
 - Low order stencil smaller diameter
 - High order stencil larger diameter



Data Movement Arising from Stencils

- Hardware prefetching streams
- TLB entries
- Worst case usage (cube-shaped):
 - ~ diameter in 2D
 - \sim diameter² in 3D

- Usage limits parallelism
 - e.g. number of threads < streams
- Problem exacerbated with tiling
- Tiling factors are architecture specific
 - Size of cache, page size, number of prefetching stream





Data Movement Arising from Stencils

- Hardware prefetching streams
- TLB entries
- Worst case usage (cube-shaped):
 - ~ diameter in 2D
 - $\sim \text{diameter}^2$ in 3D

- Usage limits parallelism
 - e.g. number of threads < streams
- Problem exacerbated with tiling
- Tiling factors are architecture specific
 - Size of cache, page size, number of prefetching stream



Approach #1: Code A to Codes B & C

- Extended compiler transformation and code generation framework with *domain-specific specialization* (supports C-like C++)
 - Target is loop-based scientific applications and related tensor computations such as CNNs
 - Composable transformations
- Optimization strategy can be specified or derived with *transformation recipes*
 - Also optimization parameters exposed
 - Separates code from mapping!

• Autotuning

- Systematic exploration of alternate transformation recipes and their optimization parameter values
- Search technology to prune combinatorial space

for (i=0;i<N;i++) { for (j=1;j<M;j++) { S0: a[i][j] = b[j] a[i][j-1]; I = {[i,j] | 0<=i<N 1<=j<=M}



Transformation Recipes, Codes B & C

/* jacobi_box_4_64.py, 27-pt stencil, 64³ box size */
from chill import *

#select which computation to optimize
source('jacobi_box_4_64.c')
procedure('smooth_box_4_64')
loop(0)
original() # fuse wherever possible

#create a parallel wavefront skew([0,1,2,3,4,5],2,[2,1]) permute([2,1,3,4])

#partial sum for high order stencils and fuse result distribute([0,1,2,3,4,5],2) stencil_temp(0) stencil_temp(5) fuse([2,3,4,5,6,7,8,9],1) fuse([2,3,4,5,6,7,8,9],2) fuse([2,3,4,5,6,7,8,9],3) fuse([2,3,4,5,6,7,8,9],4) /* gsrb.lua, variable coefficient GSRB, 64³ box size */
init("gsrb_mod.cu", "gsrb",0,0)
dofile("cudaize.lua") # custom commands in lua

set up parallel decomposition, adjust via autotuning TI=32 TJ=4 TK=64 TZ=64

tile_by_index(0, {"box","k","j", "i"},{TZ,TK, TJ, TI},{I1_control="bb", I2_control="kk", I3_control="jj", I4_control="ii"}, {"bb","box","kk","k","jj","j","ii","i"})

cudaize(0, "kernel_GPU",{_temp=N*N*N*N,_beta_i=N*N*N*N, _phi=N*N*N*N},{block={"ii","jj","box"}, thread={"i","j"}},{})

Similar Idea is Gaining Traction in Domain-Specific Frameworks

Halide

a language for image processing and computational photography

vectorize(x_inner, factor), equivalent to gradient.split(x, x, x_inner, 4); gradient.vectorize(x_inner); gradient.parallel(tile_index); gradient.split(x, x_outer, x_inner, 2); gradient.unroll(x_inner), equivalent to gradient.unroll(x, 2); gradient.tile(x, y, x_outer, y_outer, x_inner, y_inner, 4, 4); gradient.reorder(y, x); // similar to transpose gradient.split(x, x_outer, x_inner, 2) fuse(x, y, fused)

SamI TVM Stack

tile(x_parent, y_parent, x_factor, y_factor)

Perform tiling on two dimensions

The final loop order from outmost to inner most are [x_outer, y_outer, x_inner, y_inner]

- Parameters: x_parent (IterVar) The original x dimension
 - y_parent (IterVar) The original y dimension
 - x_factor (Expr) The stride factor on x axis
 - y_factor (Expr) The stride factor on y axis

Returns:

- x_outer (IterVar) Outer axis of x dimension
- y_outer (IterVar) Outer axis of y dimension
- x_inner (IterVar) Inner axis of x dimension
- p_y_inner (IterVar) Inner axis of y dimension

unroll(var)

Unroll the iteration.

Parameters: var (IterVar) – The iteration to be unrolled.

vectorize(var)

Vectorize the iteration.

Parameters: var (IterVar) - The iteration to be vectorize

Communication Avoiding: Sometimes Code A Beats Code B!

- miniGMG w/CHiLL
 - Fused operations
 - Communication-avoiding
 wavefront
 - Parallelized (OpenMP)
- Autotuning finds the best implementation for each box size
 - wavefront depth
 - nested OpenMP configuration
 - inter-thread synchronization (barrier vs. point-to-point)
- For fine grids (large arrays) CHiLL attains nearly a 4.5x speedup over baseline

GSRB Smooth (Edison)



Basu et al., HiPC 2013, IPDPS 2015.

Retargetable and Performance Portable: Optimized Code A can beat Code C!

- CHiLL can obviate the need for architecture-specific programming models like CUDA
 - CUDA-CHiLL took the sequential GSRB implementation (.c) and generated CUDA that runs on NVIDIA GPUs
 - CUDA-CHiLL autotuned over the thread block sizes and is ultimately 2% faster than the hand-optimized minigmg-cuda (Code C)
 - Adaptable to new GPU generations



Basu et al., PARCO 2017.

Approach #2: Code A to Codes B & C



Brick Data Layout + Code Generator

- A brick is a 4x4x4 mini domain without a ghost zone
- Application of a stencil reaches into other bricks (affinity important)
- Implemented with contiguous storage and adjacency lists



Code A uses Brick Domain-Specific Library

- Bricks are programmed using brick library for 3D stencils
 - Creation
 - Deletion
 - Access

```
Array
float c = prev[k][j][i] * coeff[0] + (
    prev[k][j][i+1] + prev[k][j][i-1] +
    prev[k][j+1][i] + prev[k][j-1][i] +
    prev[k+1][j][i] + prev[k-1][j][i]) *
    coeff[1];
next[k][j][i] = c * vel[k][j][i];
```

- Brick library handles cases when access across brick boundary
- Vector code generation is carried out by a code generator

Brick

```
float c = prev.elem(b,k,j,i)*coeff[0]+(
    prev.elem(b,k,j,i+1)+prev.elem(b,k,j,i-1)+
    prev.elem(b,k,j+1,i)+prev.elem(b,k,j-1,i)+
    prev.elem(b,k+1,j,i)+prev.elem(b,k-1,j,i))*
    coeff[1];
next.elem(b,k,j,i)=c*vel.elem(b,k,j,i);
    The index of brick
```

Bricks Address Themes 1 and 2

- Performance portability
 - Automation of architecture-specific code generation
 - Same abstraction, but different low-level instructions and "vector" widths
- Data movement
 - Contiguous storage of subdomain reduces overhead of automatic data movement (prefetch, TLB, cache)
 - Adjustable brick size adapts to node architecture limits
 - Indirection to represent neighbor lists gives freedom to adapt co-located bricks to architecture
 - (Ongoing) And to adapt layout to optimize communication

Performance Results (Node)



- Bricks achieve best performance for higher-order stencils, up to 5X!
- Always profitable on P100

Roofline Performance Results



- Bricks achieve performance close to memory bandwidth limit
- 125pt stencil approaches compute limit, has non-float operations

Zhao et al., PP3HPC 2018. Zhao et al., SC19.

More on Autotuning Research: Automating Finding Codes B and C

- Bricks
 - What brick size?
 - How many bricks per core? Per node?
- Program transformations
 - Which transformations to use?
 - Parameters to optimizations, such as tile size?
- Other things to tune
 - Pragmas, e.g., OpenMP
 - Application parameters, e.g., in a library like SuperLU

Pragma Autotuner

 Search Using Random Forest (SuRF) for autotuning search (may not involve compiler)



Autotuning Barriers to Adoption in HPC

- Overhead
 - Tuning search can be expensive
 - Off-line tuning expensive, programmer burden
 - Specifying search space, transformations
 - Selection and configuration of algorithms
- Scope
 - Tuning must be repeated for new execution contexts
 - Exascale resources vary during execution, platform may not be available for training
 - Economies of data scale: Learning based on a community's code
- Other programmer concerns
 - Correctness concerns with dynamically-changing code
 - Long-term tool availability

Autotuning in High-Performance Computing Applications, Balaprakash, Dongarra, Gamblin, Hall, Hollingsworth, Norris, Vuduc, Special Issue of Proceedings of the IEEE, Nov. 2018.

Conclusion

- The plethora of architecture technologies will make programming future supercomputers even more of a nightmare
- Programming system technology is desperately needed to address programmer productivity
 - Separating specification from architecture mapping
 - Architecture-specific code generation
 - Autotuning
- HOW TO BUILD THIS TECHNOLOGY???

Theme 3: Leveraging Investment in Deep Learning Compilers

POSTED ON SEP 13, 2018 TO AI RESEARCH, ML APPLICATIONS

Glow: A community-driven approach to AI infrastructure



A typical framework

NNVM Compiler

Various

frontends

NNVM



GPU-optimized * and + CPU-optimized * and +

Amazon

Various hardware backends

Google

Multi-Level Intermediate Representation Compiler Infrastructure

2019 European LLVM Developers Meeting

Challenges and opportunities:

- Domain-specific
- Many frontends
- Many target architectures
- Abundant parallelism and data reuse
- Must scale to large problems

http://code.fb.com/ml-applications/glow-a-community-driven-approach-to-ai-infrastructure/ http://aws.amazon.com/blogs/machine-learning/introducing-nnvm-compiler-a-new-open-end-to-end-compiler-for-ai-frameworks/

Convolutional Neural Network Forward Layer Code (in C)

for (n=0; n<N; n++) { // minibatch size for (k=0; k<K; k ++) { // output feature map for (c=0; c<C; c ++) { // input feature map for (p=0; p<P; p ++) { // output height ij = p * u; // input height for $(q = 0; q < Q; q + +) \{ // output width \}$ ii = q * v; // input width for (r=0; r<R; r ++) { // filter height for (s =0; s< S; s ++) $\{// filter width$ output_seq[n][k][p][q] += input [n][c][i]+r][ii+s] * weight[k][c][r][s];} } } } }