GS2

Improved Data Distribution Routines for Gyrokinetic Plasma Simulations

Adrian Jackson
EPCC
a.jackson@epcc.ed.ac.uk

DCSE Background

- Previous DCSE: Upgrading the FFTs in GS2
  - Discovered issue with some communication functionality
  - `c_redist_22` and `c_redist_32` 8% on each (1024 cores)
- Organise and send data for transpose
  - Extensive use of indirect addressing
  - Used to copy local-to-local
  - Rearranging data local to processor
  - Remote copy also
- Tested on HECToR Phase 2a
  - Cray XT4

DCSE Background

Time %	Time	Imb. Time	Imb. Time %	Calls	Group Function PE='hide'
100%	1937.148			152415504.9	Total
56.6%	1095.712			31752.0	USER
12.1%	234.504	1.101836	0.5%	1.0	main
8.3%	161.292	12.236146	7.1%	3968.0	redistribute_c_redist_22_inv_
8.1%	155.955	20.772738	11.8%	3971.0	redistribute_c_redist_32_inv_
7.8%	151.572	17.199143	10.3%	3968.0	ffttest_
6.5%	125.364	1.143944	0.9%	3968.0	redistribute_c_redist_22_
5.0%	96.416	1.968193	2.0%	3971.0	redistribute_c_redist_32_

DCSE Background

```
c_redist_22_inv routine:

do i = 1, r%to(iproc)%nn

   to_here(r%from(iproc)%k(i), r%from(iproc)%l(i)) &

   = from_here(r%to(iproc)%k(i), r%to(iproc)%l(i))

end do


c_redist_32 routine:

do i = 1, r%from(iproc)%nn

   to_here(r%to(iproc)%k(i), r%to(iproc)%l(i)) &

   = from_here(r%from(iproc)%k(i), r%from(iproc)%l(i),  &

   r%from(iproc)%m(i))

end do
```

DCSE Background

- c_redist_22
  - 5 loads, 1 store, 1 increment, 1 comparison
  - Inhibits compiler optimisation: No re-order of iterations, no pre-fetch, etc..
- Optimal loop:

```
do i = 1, upperi
  do j = 1, upperj
    to_here(i,j) = from_here(i,j)
  end do
end do
```

  - 1 load, 1 store, 1 increment, 1 comparison
  - Also possibly better cache re-use
- Data layout investigated in Totalview
  - Optimised version suggested
  - Given problem, 128 cores used

DCSE Background

```
c_redist_22:
do kxj = 1, r%from(iproc)%nn, 32*96
  valueF = r%from(iproc)%l(kxj) - 1
  valueT = r%to(iproc)%l(kxj) - 1
  do jxj = 1, 32
    do ixj = 1, 96
      to_here(jxj, valueT + ixj ) &
      = from_here(ixj, valueF + jxj)
    enddo
  enddo
enddo

c_redist_22_inv:
do kxj = 1, r%to(iproc)%nn, 32*96
  valueF = r%from(iproc)%l(kxj) - 1
  valueT = r%to(iproc)%l(kxj) - 1
  do ixj = 1, 96
    do jxj = 1, 32
      to_here(ixj, valueF + jxj) &
      = from_here(jxj, valueT + ixj)
    enddo
  enddo
enddo
```

GS2

- Flux-tube gyrokinetic code
  - Initial value code
  - Solves the gyrokinetic equations for perturbed distribution functions together with Maxwell's equations for the turbulent electric and magnetic fields
  - Linear (fully implicit) and Non-linear (dealiased pseudo-spectral) terms
  - 5D space – 3 spatial, 2 velocity
  - Different species of charged particles
- Advancement of time in Fourier space
  - FFTs only in two spatial dimensions perpendicular to the magnetic field
- Non-linear term calculated in position space
  - Requires FFTs

GS2

- Different data layouts
  - Can decompose: species, energy, ky, kx, lambda differently (right to left)
  - xyles, yxles, lyxes, yxels, lxyes, lexys
- K space
  - Main domain for computations
  - g_lo(ig,isgn,iglo): ig, isgn local; iglo decomposed according to “layout”
  - Have to go to real space for non-linear computations
- FFTs
  - Map from K space to xxf space
    - 3D $\rightarrow$ 2D: c_redist_32
    - xxf_lo(it,ixxf): it (kx) local; ixxf decomposed by “layout” (minus xy), then isgn, ig, and ik (ky)
  - Map from xxf space to yxf space
    - 2D $\rightarrow$ 2D: c_redist_22
    - yxf_lo(ik,ixyf): ik local; iyxf decomposed by layout then isgn, ig, and it

GS2 Initial Performance

Number of Cores:			128	192	256	512	528	1024	1080
_22	LC	Min	44.94	33.17	24.06	11.19	11.75	5.88	2.84
		Max	55.23	37.00	28.46	14.26	15.40	7.70	7.90
		Average	**53.71**	**34.73**	**27.17**	**13.73**	**12.75**	**6.81**	**5.16**
	RC	Min	0.04	0.33	0.10	0.16	0.30	0.34	0.59
		Max	0.08	23.38	0.15	0.30	15.65	0.59	11.42
		Average	**0.07**	**4.55**	**0.13**	**0.26**	**3.63**	**0.48**	**4.99**
_22_inv	LC	Min	9.79	8.71	5.90	2.47	3.10	1.49	0.78
		Max	14.01	9.31	7.04	3.56	3.43	1.82	1.72
		Average	**13.73**	**9.09**	**6.90**	**3.46**	**3.34**	**1.76**	**1.36**
	RC	Min	0.01	0.07	0.02	0.03	0.05	0.08	0.13
		Max	0.02	1.43	0.04	0.07	0.71	0.14	2.28
		Average	**0.01**	**0.45**	**0.03**	**0.05**	**0.21**	**0.10**	**1.07**
_32	LC	Min	41.15	26.67	21.93	9.80	5.76	3.19	1.57
		Max	61.93	35.80	31.66	14.31	14.65	5.14	4.61
		Average	**49.85**	**30.15**	**25.60**	**12.46**	**9.03**	**3.92**	**3.04**
	RC	Min	0.04	5.25	0.08	0.14	1.30	6.37	7.89
		Max	0.07	25.72	0.12	0.24	28.23	13.06	21.89
		Average	**0.05**	**12.97**	**0.10**	**0.19**	**13.10**	**9.22**	**11.84**
_32_inv	LC	Min	9.78	5.92	5.40	1.95	1.13	0.59	0.18
		Max	14.60	7.77	7.06	3.41	3.30	0.86	0.78
		Average	**12.31**	**6.93**	**6.26**	**2.79**	**1.98**	**0.77**	**0.55**
	RC	Min	0.01	1.03	0.02	0.03	0.28	1.46	1.72
		Max	0.02	6.22	0.04	0.07	4.68	2.27	4.16
		Average	**0.01**	**3.00**	**0.03**	**0.05**	**2.74**	**1.99**	**2.54**

xyles layout

RC: remote copy

LC: local copy

Proc/core count important: 1,2,4,8,16,32,64, 128,256,512,1536, 3584,4608

Calculating Indirect Addresses

```
select case (layout)
case ('yxels')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(ie-1 + xxf_lo%negrid*(il-1 + xxf_lo%nlambda*(is-1)))))
case ('yxles')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(il-1 + xxf_lo%nlambda*(ie-1 + xxf_lo%negrid*(is-1)))))
case ('lexys')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(il-1 + xxf_lo%nlambda*(ie-1 + xxf_lo%negrid*(is-1)))))
case ('lxyes')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(il-1 + xxf_lo%nlambda*(ie-1 + xxf_lo%negrid*(is-1)))))
case ('lyxes')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(il-1 + xxf_lo%nlambda*(ie-1 + xxf_lo%negrid*(is-1)))))
case ('xyles')
   ixxf = ik-1 + xxf_lo%naky*(ig+xxf_lo%ntgrid + (2*xxf_lo%ntgrid+1)*(isign-1 &
   + xxf_lo%nsign*(il-1 + xxf_lo%nlambda*(ie-1 + xxf_lo%negrid*(is-1)))))
end select
```

Calculating Indirect Addresses

```
select case (layout)
case ('yxels')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(ie-1 + yxf_lo%negrid*(il-1 + yxf_lo%nlambda*(is-1)))))
case ('yxles')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(il-1 + yxf_lo%nlambda*(ie-1 + yxf_lo%negrid*(is-1)))))
case ('lexys')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(il-1 + yxf_lo%nlambda*(ie-1 + yxf_lo%negrid*(is-1)))))
case ('lxyes')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(il-1 + yxf_lo%nlambda*(ie-1 + yxf_lo%negrid*(is-1)))))
case ('lyxes')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(il-1 + yxf_lo%nlambda*(ie-1 + yxf_lo%negrid*(is-1)))))
case ('xyles')
   idx_yxf = it-1 + yxf_lo%nx*(ig+yxf_lo%ntgrid + (2*yxf_lo%ntgrid+1)*(isign-1 &
   + yxf_lo%nsign*(il-1 + yxf_lo%nlambda*(ie-1 + yxf_lo%negrid*(is-1)))))
end select
```

Calculating Indirect Addresses

```
do iglo = g_lo%llim_world, g_lo%ulim_world
   do isign = 1, 2
      do ig = -ntgrid, ntgrid
         call gidx2xxfidx (ig, isign, iglo, g_lo, xxf_lo, it, ixxf)
         if (idx_local(g_lo,iglo)) then
            ip = proc_id(xxf_lo,ixxf)
            n = nn_from(ip) + 1
            nn_from(ip) = n
            from_list(ip)%first(n) = ig
            from_list(ip)%second(n) = isign
            from_list(ip)%third(n) = iglo
         end if
         if (idx_local(xxf_lo,ixxf)) then
            ip = proc_id(g_lo,iglo)
            n = nn_to(ip) + 1
            nn_to(ip) = n
            to_list(ip)%first(n) = it
            to_list(ip)%second(n) = ixxf
         end if
      end do
   end do
end do
```

Calculating Indirect Addresses

```
do ixxf = xxf_lo%llim_world, xxf_lo%ulim_world
   do it = 1, yxf_lo%nx
      call xxfidx2yxfidx (it, ixxf, xxf_lo, yxf_lo, ik, iyxf)
      if (idx_local(xxf_lo,ixxf)) then
         ip = proc_id(yxf_lo,iyxf)
         n = nn_from(ip) + 1
         nn_from(ip) = n
         from_list(ip)%first(n) = it
         from_list(ip)%second(n) = ixxf
      end if
      if (idx_local(yxf_lo,iyxf)) then
         ip = proc_id(xxf_lo,ixxf)
         n = nn_to(ip) + 1
         nn_to(ip) = n
         to_list(ip)%first(n) = ik
         to_list(ip)%second(n) = iyxf
      end if
   end do
end do
```

Solution – c_redist_22

```
do while (i .le. r%from(iproc)%nn)
   itmin = r%from(iproc)%k(i)
   ixxf = r%from(iproc)%l(i)
   ik = r%to(iproc)%k(i)
   iyxf = r%to(iproc)%l(i)
   it_nlocal = (yxf_lo%ulim_proc+1) - iyxf
   itmax = min((itmin-1)+it_nlocal,yxf_lo%nx)
   do it = itmin,itmax
      to_here(ik,iyxf) = from_here(it,ixxf)
      iyxf = iyxf + 1
      i = i + 1
   end do
end do
```

Solution – c_redist_22

- For large enough inner loop
  - 1 load, 1 store, 3 increments, 1 comparison
- Original
  - 5 loads, 1 store, 1 increment, 1 comparison
- Ideal
  - 1 load, 1 store, 1 increment, 1 comparison

Solution – c_redist_32

```
do while(i .le. r%from(iproc)%nn)
      f2 = r%from(iproc)%l(i)
      f3 = r%from(iproc)%m(i)
      t1 = r%to(iproc)%k(i)
      do while (f2 .le. f2max)
         f1 = r%from(iproc)%k(i)
         t2 = r%to(iproc)%l(i)
         thigh = ceiling(((xxf_lo%ulim_proc+1) - t2)*nakyrecip)
         thigh = thigh + (f1-1)
         fhigh = min(thigh,r%from_high(1))
         do k = f1,fhigh
            to_here(t1,t2) = from_here(k,f2,f3)
            t2 = t2 + naky
            i = i + 1
         end do
         if(thigh .gt. r%from_high(1)) then
            f2 = f2 + 1
         else
            f2 = f2max + 1
         end if
      end do
   end do
```

Solution – c_redist_32

- For large enough inner loop
  - 1 load, 1 store, 3 increments, 1 comparison
- Original
  - 6 loads, 1 store, 1 increment, 1 comparison
- Ideal
  - 1 load, 1 store, 1 increment, 1 comparison

Preliminary results

Number of Cores:			128	192	256	512	528	1024	1080
_22	OC	Min	44.94	33.17	24.06	11.19	11.75	5.88	2.84
		Max	55.23	37.00	28.46	14.26	15.40	7.70	7.90
		Average	**53.71**	**34.73**	**27.17**	**13.73**	**12.75**	**6.81**	**5.16**
	NC	Min	17.82	17.07	10.39	4.00	5.00	2.34	1.17
		Max	36.22	23.53	17.87	8.93	8.54	4.35	3.71
		Average	**32.68**	**21.19**	**16.15**	**8.05**	**6.98**	**3.54**	**2.46**
_22_inv	OC	Min	9.79	8.71	5.90	2.47	3.10	1.49	0.78
		Max	14.01	9.31	7.04	3.56	3.43	1.82	1.72
		Average	**13.73**	**9.09**	**6.90**	**3.46**	**3.34**	**1.76**	**1.36**
	NC	Min	4.58	5.22	3.14	1.17	1.86	0.79	0.37
		Max	9.05	5.98	4.54	2.29	2.21	1.15	1.08
		Average	**8.77**	**5.79**	**4.39**	**2.21**	**2.12**	**1.10**	**0.84**
_32	OC	Min	41.15	26.67	21.93	9.80	5.76	3.19	1.57
		Max	61.93	35.80	31.66	14.31	14.65	5.14	4.61
		Average	**49.85**	**30.15**	**25.60**	**12.46**	**9.03**	**3.92**	**3.04**
	NC	Min	44.46	25.81	23.56	10.07	5.29	2.13	1.14
		Max	63.05	37.48	33.36	16.63	15.94	3.81	3.36
		Average	**54.33**	**31.11**	**27.93**	**13.68**	**8.72**	**2.84**	**2.30**
_32_inv	OC	Min	9.78	5.92	5.40	1.95	1.13	0.59	0.18
		Max	14.60	7.77	7.06	3.41	3.30	0.86	0.78
		Average	**12.31**	**6.93**	**6.26**	**2.79**	**1.98**	**0.77**	**0.55**
	NC	Min	6.90	4.63	3.56	1.64	1.06	0.41	0.16
		Max	10.97	6.02	5.30	2.55	2.32	0.61	0.63
		Average	**8.82**	**5.36**	**4.48**	**2.33**	**1.63**	**0.55**	**0.44**

Conclusions

- Indirect addressing
  - Very useful tool for programming and simplicity
  - Can impact performance, especially in core routines
- Possible to replace indirect addressing
  - Breaks simplicity of code
  - Need to understand so can modify if layout code modified (modularity compromised)
  - However, can significantly improve performance
- Need to investigate the c_redist_32 functionality
  - Work out performance problem
  - Simplify if possible
- Remote copy code needs attention
- Also scope for improving decompositions
  - Core sweet spots for g_lo not necessarily ideal for ixxf and ixyf
  - Build load imbalance into ixxf and ixyf decomposition to remove remote copy calls: e.g. 32 * 2 * 31 * 8 * 32 * 2 (31 = (2*ntgrid) + 1)

Thanks

Questions?