Comparing Optimizations of GTC for the Cray X1E and XT3

James B. White III (Trey)
trey@ornl.gov

Nathan Wichmann, Cray
Stephane Ethier, PPPL
Acknowledgement

Research sponsored by the Mathematical, Information, and Computational Sciences Division, Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.
Comparing Optimizations of GTC for the Cray X1E and XT3

• X1E and XT3 overview
• GTC overview
• X1E optimization
  – 40+ slides
  – Lots of code
  – Security, lock the doors!
• XT3 performance
• GTC weak-scaling benchmark
NCCS systems

• Cray X1E (Phoenix)
  – 1024 MSPs
  – 2 GB/MSP
  – Programming Environment 5.4

• Cray XT3 (Jaguar)
  – 5212 compute processors
    • 2.4 GHz Opterons
    • 2 GB/processor
  – PGI 6.0 and 6.1.3 compilers
GTC

- Simulates turbulent transport in fusion plasmas
- Intrinsically global 3D gyrokinetic particle-in-cell code
- Scales to 10,000+ processors
- Fortran and MPI
- Earlier vector version exists, but started from current non-vector version
  - Controlled tuning experiment
  - Used some techniques from earlier vector version
GTC optimization on Cray X1E

- Run benchmark
  - 64 processors

- Generate by-function, by-line profile
  - `pat_report -b functions,lines`

- Commit to Subversion repository
  - “r2” = Revision 2

- See loopmark for expensive functions

- Tune
  - Vectorize, multistream, see loopmark

- Repeat
## Profile r2

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Cum.Samp%</th>
<th>Samp</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>22049624</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>61.5%</td>
<td>61.5%</td>
<td>13557450</td>
<td></td>
<td>chargei_</td>
</tr>
<tr>
<td>12.2%</td>
<td>12.2%</td>
<td>2698466</td>
<td>line.140</td>
<td></td>
</tr>
<tr>
<td>9.2%</td>
<td>21.5%</td>
<td>2031530</td>
<td>line.132</td>
<td></td>
</tr>
<tr>
<td>6.4%</td>
<td>27.8%</td>
<td>1404587</td>
<td>line.102</td>
<td></td>
</tr>
<tr>
<td>4.8%</td>
<td>32.6%</td>
<td>1047693</td>
<td>line.133</td>
<td></td>
</tr>
<tr>
<td>3.8%</td>
<td>36.4%</td>
<td>833888</td>
<td>line.105</td>
<td></td>
</tr>
<tr>
<td>3.3%</td>
<td>39.7%</td>
<td>735487</td>
<td>line.137</td>
<td></td>
</tr>
<tr>
<td>3.2%</td>
<td>42.9%</td>
<td>705596</td>
<td>line.48</td>
<td></td>
</tr>
<tr>
<td>3.0%</td>
<td>45.9%</td>
<td>663002</td>
<td>line.136</td>
<td></td>
</tr>
<tr>
<td>2.8%</td>
<td>48.7%</td>
<td>627426</td>
<td>line.67</td>
<td></td>
</tr>
<tr>
<td>2.8%</td>
<td>51.5%</td>
<td>610015</td>
<td>line.127</td>
<td></td>
</tr>
<tr>
<td>2.1%</td>
<td>53.6%</td>
<td>452091</td>
<td>line.128</td>
<td></td>
</tr>
<tr>
<td>2.0%</td>
<td>55.5%</td>
<td>438478</td>
<td>line.95</td>
<td></td>
</tr>
<tr>
<td>1.9%</td>
<td>57.5%</td>
<td>429772</td>
<td>line.141</td>
<td></td>
</tr>
<tr>
<td>1.7%</td>
<td>59.2%</td>
<td>381375</td>
<td>line.129</td>
<td></td>
</tr>
<tr>
<td>0.7%</td>
<td>59.9%</td>
<td>150991</td>
<td>line.65</td>
<td></td>
</tr>
</tbody>
</table>
Loopmarks for chargei

90.  1------------<   do m=1,mi
...
93.  1            kk=kzion(m)
...
97.  1Vs----------<  do larmor=1,4
...
127.  1Vs            ij=jtion0(larmor,m)
128.  1Vs            densityi(kk,ij) = densityi(kk,ij) + wz0*wt00
129.  1Vs            densityi(kk+1,ij) = densityi(kk+1,ij) + wz1*wt00
130.  1Vs
131.  1Vs            ij=ij+1
132.  1Vs            densityi(kk,ij) = densityi(kk,ij) + wz0*wt10
133.  1Vs            densityi(kk+1,ij) = densityi(kk+1,ij) + wz1*wt10
...
144.  1Vs------------>  enddo
145.  1------------->  enddo

ftn-6289 ftn: VECTOR File = chargei.F90, Line = 90
A loop starting at line 90 was not vectorized because a recurrence was found
on "DENSITYI" between lines 128 and 129.
Vectorization of chargei

... #ifndef _UNICOSMP
15. integer, parameter :: vlen = 256
16. integer :: mv, v
17. real(wp) dnipart(mgrid,0:mzeta,vlen)
18. #endif
... #ifndef _UNICOSMP
98. m------------- < do mv=1,mi,vlen
99. m !dir$ prefervector
100. m MVs--------- < do m=mv,min(mv+vlen-1,mi)
101. m MVs
102. m MVs v=m-mv+1
103. m MVs #else
104. m MVs do m=1,mi
105. m MVs #endif

Add a vector dimension
Vectorization of chargei

... 
140. m MVs 3  \hspace{1cm} ij=jtion0(larmor,m) 
141. m MVs 3  \hspace{1cm} dnipart(ij,kk,v) = dnipart(ij,kk,v) + wz0*wt00 
142. m MVs 3  \hspace{1cm} dnipart(ij,kk+1,v) = dnipart(ij,kk+1,v) + wz1*wt00 
... 
194. ir----------< \hspace{1cm} do v=1,vlen 
195. ir 2----------< \hspace{1cm} do kk=0,mzeta 
196. ir 2 \hspace{1cm} !dir$ preferstream 
197. ir 2 MV-------< \hspace{1cm} do ij=1,mgrid 
198. ir 2 MV \hspace{1cm} densityi(kk,ij) = densityi(kk,ij) + dnipart(ij,kk,v) 
199. ir 2 MV------> enddo 
200. ir 2--------> enddo 
201. ir----------> enddo 

\textit{Sum over extra dimension}

vlen=256 
\hspace{1cm} mzeta=1 for 64+ processors 
\hspace{1cm} mgrid=32,449
GTC Runtime
**Profile r3**

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Cum.Samp%</th>
<th>Samp</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>8768310</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>30.3%</td>
<td>30.3%</td>
<td>2660905</td>
<td>shifti_</td>
<td></td>
</tr>
<tr>
<td>12.3%</td>
<td>12.3%</td>
<td>1080482</td>
<td>line.154</td>
<td></td>
</tr>
<tr>
<td>7.3%</td>
<td>19.7%</td>
<td>644222</td>
<td>line.206</td>
<td></td>
</tr>
<tr>
<td>6.1%</td>
<td>25.8%</td>
<td>535231</td>
<td>line.128</td>
<td></td>
</tr>
<tr>
<td>2.1%</td>
<td>27.9%</td>
<td>186416</td>
<td>line.207</td>
<td></td>
</tr>
<tr>
<td>1.4%</td>
<td>29.3%</td>
<td>120175</td>
<td>line.132</td>
<td></td>
</tr>
<tr>
<td>0.2%</td>
<td>29.5%</td>
<td>17864</td>
<td>line.190</td>
<td></td>
</tr>
</tbody>
</table>

...  
| 7.5% | 93.1% | 660755 | chargei_ |

...
Loopmarks for shifti

Loop over particles that might move to another processor

128. r----------------< do m=m0,mi
129. r zetaright=min(2.0*pi,zion(3,m))-zetamax
130. r zetaleft=zion(3,m)-zetamin
131. r
132. r if( zetaright*zetaleft > 0 )then
133. r zetaright=zetaright*0.5*pi_inv
134. r zetaright=zetaright-real(floor(zetaright))
135. r msend=msend+1
136. r kzi(msend)=m
137. r

Does it move to another processor?
Loopmarks for shifti

138. r          if( zetarigh < 0.5 )then
139. r          ! # of particle to move right
140. r          msendright(1)=msendright(1)+1
141. r          iright(msendright(1))=m
142. r          ! keep track of tracer
143. r          if( nhybrid == 0 .and. m == ntracer )then
144. r          msendright(2)=msendright(1)
145. r          ntracer=0
146. r          endif
147. r
148. r          ! # of particle to move left
149. r          else
150. r          msendleft(1)=msendleft(1)+1
151. r          ileft(msendleft(1))=m
152. r          if( nhybrid == 0 .and. m == ntracer )then
153. r          msendleft(2)=msendleft(1)
154. r          ntracer=0
155. r          endif
156. r          endif
157. r          endif
158. r----------->  enddo
Loopmarks for shifti

Is the tracer particle about to leave this processor?

142.  r        ! keep track of tracer
143.  r            if( nhybrid == 0 .and. m == ntracer ) then
144.  r                msendright(2)=msendright(1)
145.  r                ntracer=0
146.  r            endif

ftn-6005 ftn: SCALAR File = shifti.F90, Line = 128
    A loop starting at line 128 was unrolled 2 times.

ftn-6254 ftn: VECTOR File = shifti.F90, Line = 128
    A loop starting at line 128 was not vectorized because a recurrence was found
    on "NTRACER" at line 145.
Vectorization of shifti

123. #elif defined _UNICOSMP
124.
125. V----------------< do m=m0,mi
126. V zetaright=min(2.0*pi,zion(3,m))-zetamax
127. V zetaleft=zion(3,m)-zetamin
128. V if (zetaright*zetaleft > 0) then
129. V msend=msend+1
130. V kzi(msend)=m
131. V endif
132. V----------------> enddo

Find indices of particles that will move
Similar to Fortran pack function
Vectorizes but doesn’t multistream
Vectorization of shifti

134. do i=1,msend
135.     m=kzi(i)
136.     zetaright=min(2.0*pi,zion(3,m))-zetamax
137.     zetaright=zetaright*0.5*pi_inv
138.     zetaright=zetaright-real(floor(zetaright))
139.     if (zetaright < 0.5) then
140.         ! # of particle to move right
141.             msendright(1)=msendright(1)+1
142.             iright(msendright(1))=m
143.         ! # of particle to move left
144.             else
145.             msendleft(1)=msendleft(1)+1
146.             ileft(msendleft(1))=m
147.             endif
148. enddo

Decide between sending left or right
Generalization of Fortran pack function
Vectorizes but doesn’t multistream
Vectorization of shifti

150. ! keep track of tracer
151. if ((nhybrid == 0) .and. (ntracer > 0)) then
152.      MV-------------<
153.      do i=1,msendright(1)
154.      if (iright(i) == ntracer) msendright(2)=i
155.      enddo
156.      if (msendright(2) /= 0) then
157.      ntracer=0
158.      else
159.      MV-------------<
160.      do i=1,msendleft(1)
161.      if (ileft(i) == ntracer) msendleft(2)=i
162.      endif
163.      endif

See if any sent particles are the tracer
Reductions (like sum) of logical values
Profile r5

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Cum.Samp%</th>
<th>Samp</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>7126472</td>
<td><strong>Total</strong></td>
<td></td>
</tr>
<tr>
<td>34.0%</td>
<td>34.0%</td>
<td>2422227</td>
<td><strong>pushi_</strong></td>
<td></td>
</tr>
<tr>
<td>9.6%</td>
<td>9.6%</td>
<td>683490</td>
<td><strong>line.93</strong></td>
<td></td>
</tr>
<tr>
<td>4.6%</td>
<td>14.2%</td>
<td>330526</td>
<td><strong>line.80</strong></td>
<td></td>
</tr>
<tr>
<td>4.3%</td>
<td>18.6%</td>
<td>309770</td>
<td><strong>line.92</strong></td>
<td></td>
</tr>
<tr>
<td>3.8%</td>
<td>22.4%</td>
<td>269578</td>
<td><strong>line.94</strong></td>
<td></td>
</tr>
<tr>
<td>1.9%</td>
<td>24.2%</td>
<td>131984</td>
<td><strong>line.62</strong></td>
<td></td>
</tr>
<tr>
<td>1.5%</td>
<td>25.7%</td>
<td>107877</td>
<td><strong>line.148</strong></td>
<td></td>
</tr>
<tr>
<td>0.9%</td>
<td>26.6%</td>
<td>62389</td>
<td><strong>line.105</strong></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.7%</td>
<td>81.3%</td>
<td>973509</td>
<td><strong>shifti_</strong></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.2%</td>
<td>90.5%</td>
<td>654013</td>
<td><strong>chargei_</strong></td>
<td></td>
</tr>
</tbody>
</table>
Loopmarks for pushi

60. M---------------< do m=1,mi
...
67. M
68. M Vs-------------< do larmor=1,4  Vector length of 4?!
69. M Vs
70. M Vs                i=jtion0(larmor,m)
71. M Vs                wp0=1.0-wpion(larmor,m)
72. M Vs                wt00=1.0-wtio0(larmor,m)
73. M Vs                e1=e1+wp0*wt00*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
74. M Vs                e2=e2+wp0*wt00*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
75. M Vs                e3=e3+wp0*wt00*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))
...
96. M Vs-------------> enddo
97. M
98. M                    wpv(1,m)=0.25*e1
99. M                    wpv(2,m)=0.25*e2
100. M                    wpv(3,m)=0.25*e3
101. M
102. M---------------> enddo
Vectorization for pushi

58.          !dir$ prefervector
...
61.        MV----------<  do m=1,mi
...
69.        MV 2----------<  do larmor=1,4
70.        MV 2
71.        MV 2              ij=jtion0(larmor,m)
72.        MV 2              wp0=1.0-wpion(larmor,m)
73.        MV 2              wt00=1.0-wtio0(larmor,m)
74.        MV 2              e1=e1+wp0*wt00*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij))
75.        MV 2              e2=e2+wp0*wt00*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij))
76.        MV 2              e3=e3+wp0*wt00*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij))
...
97.        MV 2---------->  enddo
98.        MV
99.        MV              wpi(1,m)=0.25*e1
100.        MV              wpi(2,m)=0.25*e2
101.        MV              wpi(3,m)=0.25*e3
102.        MV
103.        MV---------->  enddo
GTC Runtime

Time (seconds)

XT3 r2
X1E r2
X1E r3
X1E r5
X1E r6
## Profile r6

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Cum.Samp%</th>
<th>Samp</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>5146367</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25.4%</td>
<td>25.4%</td>
<td>1306364</td>
<td>MPI_CRAY_fast_barrier</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22.2%</td>
<td>47.6%</td>
<td>1144908</td>
<td>rng_number_s1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.2%</td>
<td>66.9%</td>
<td>989615</td>
<td>shifiti_</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.3%</td>
<td>59.9%</td>
<td>631146</td>
<td>line.248</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2%</td>
<td>64.1%</td>
<td>214897</td>
<td>line.249</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.0%</td>
<td>65.1%</td>
<td>51156</td>
<td>line.128</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Not relevant for science
Not timed for performance

Problem setup
Reproducible random #s

Relevant bottleneck
Part of timing

“I’m baaack!”
Loopmarks for shifti

Pack local particles into holes left by sent particles

240. 1------------------ lasth=m-send
241. 1------------------ do i=1,m-send  Loop over sent particles
242.   1              m=kzi(i)
243.   1              if (m > mi) exit  !Break out of the DO loop if m > mi
244.   1 2-------------- do while(mtop == kzi(lasth))
245.   1 2              mtop=mtop-1
246.   1 2              lasth=lasth-1
247.   1 2-------------- enddo
248. 1              if( nhybrid == 0 .and. mtop == ntracer )ntracer=m
249. 1 r V M---<><><> zion(1:nparam,m)=zion(1:nparam,mtop)
250. 1 f----------<> zion0(1:nparam,m)=zion0(1:nparam,mtop)
251. 1              mtop=mtop-1
252. 1              if (mtop == mi) exit  !Break out of the DO loop
253. 1-------------- enddo

ftn-6254 ftn: VECTOR File = shifti.F90, Line = 241
A loop starting at line 241 was not vectorized because a recurrence was found on "NTRACER" at line 248.
Vectorization of shifti

Pack local particles into holes left by sent particles

Find particles from the end to fill in holes
(Don’t use a hole to fill in a hole)
Another pack-like operation
Vectorization of shifti

**Copy end particles into holes**

**Vectorize over holes, multistream over nparam**

257. !dir$ concurrent, prefervector
258. Vm------------------<  do i=1,nholes
259. Vm r 3 M--<><><>  zion(1:nparam,kzi(i))=zion(1:nparam,jzi(i))
260. Vm f------------<>  zion0(1:nparam,kzi(i))=zion0(1:nparam,jzi(i))
261. Vm------------------>  enddo
262. if (nhybrid == 0 .and. ntracer > mi) then
263.  itracer=0
264. MV------------------<  do i=1,nholes
265. MV  if (ntracer == jzi(i)) itracer=i
266. MV------------------>  enddo
267. ntracer=kzi(itracer)
268. endif

**Move the tracer, if necessary**

**Another reduction over logical values**
Loopmarks for shifti

Copy received particles from message buffers

332. M------------< do m=1,mrecvleft(1)
333. M V 3------<> zion(1:nparam,m+mi)=recvleft(1:nparam,m)
334. M f------<> zion0(1:nparam,m+mi)=recvleft(nparam+1:nzion,m)
335. M----------> enddo

... 339. M------------< do m=1,mrecvright(1)
340. M V 3------<> zion(1:nparam,m+mi+mrecvleft(1))=recvright(1:nparam,m)
341. M f------<> zion0(1:nparam,m+mi+mrecvleft(1))=recvright(nparam+1:nzion,m)
342. M----------> enddo

ftn-6294 ftn: VECTOR File = shifti.F90, Line = 332
A loop starting at line 332 was not vectorized because a better candidate was
found at line 333.

ftn-6294 ftn: VECTOR File = shifti.F90, Line = 339
A loop starting at line 339 was not vectorized because a better candidate was
found at line 340.

nparam = 6 or 7
Vectorization of shifti

363. !dir$ prefervector
364. Vm--------------<  do m=1,mrecvleft(1)
365. Vm r 3 M--<><><>  zion(1:nparam,m+mi)=recvleft(1:nparam,m)
366. Vm f----------<>  zion0(1:nparam,m+mi)=recvleft(nparam+1:nzion,m)
367. Vm------------>  enddo

371. !dir$ prefervector
372. Vm--------------<  do m=1,mrecvright(1)
373. Vm r 3 M--<><><>  zion(1:nparam,m+mi+mrecvleft(1))=recvright(1:nparam,m)
374. Vm f----------<>  zion0(1:nparam,m+mi+mrecvleft(1))=recvright(nparam+1:nzion,
375. Vm------------>  enddo

Vectorize over particles, multistream over nparam
GTC Runtime

Time (seconds)

- XT3 r2
- X1E r2
- X1E r3
- X1E r5
- X1E r6
- X1E r8
### Profile r8

<table>
<thead>
<tr>
<th>Sampler%</th>
<th>Cum. Sampler%</th>
<th>Sampler</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>4205531</td>
<td>Total</td>
</tr>
</tbody>
</table>

Now revisit chargei and pushi

No obvious fixes for chargei
Loopmarks for pushi

Looks OK, right?

259. MV----------<
260. MV          do m=1,mi
261. MV          ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
262. MV          dtem(ip)=dtem(ip)+wpi(2,m)*zion(5,m)
263. MV---------->  enddo

301. MV----------<
305. MV          do m=1,mi
306. MV          ip=max(1,min(mflux,1+int((r-a0)*d_inv)))
310. MV          ii=max(0,min(mpsi,int((r-a0)*delr+0.5)))
313. MV          rmarker(ip)=rmarker(ip)+zion0(6,m)
314. MV          eflux(ip)=eflux(ip)+vdrenergy
324. MV          dmark(ip)=dmark(ip)+wpi(1,m)*r
325. MV          dden(ip)=dden(ip)+1.0
326. MV---------->  enddo
Loopmarks for pushi

Not so fast!

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 259
A vectorized loop contains potential conflicts due to indirect addressing at
line 262, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 259
A vectorized loop contains potential conflicts due to indirect addressing at
line 261, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 301
A vectorized loop contains potential conflicts due to indirect addressing at
line 310, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 301
A vectorized loop contains potential conflicts due to indirect addressing at
line 314, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 301
A vectorized loop contains potential conflicts due to indirect addressing at
line 325, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 301
A vectorized loop contains potential conflicts due to indirect addressing at
line 313, causing less efficient code to be generated.

ftn-6371 ftn: VECTOR File = pushi.f90, Line = 301
A vectorized loop contains potential conflicts due to indirect addressing at
line 324, causing less efficient code to be generated.
Section 2.1.5: Isoparametric and Nonlinear Behavior

Not permutations!
Vectorization of pushi

Add a vector dimension, like for chargei

19. ifdef _UNICOSMP
20. integer, parameter :: vlen=256
21. integer :: mv, v
22. real(wp) :: vdtem(vlen,mflux), vdden(vlen,mflux)
23. real(wp) :: vhfluxpsi(0:msi,vlen),
      vrmarker(vlen,mflux), veflux(vlen,mflux),
      vdmrk(vlen,mflux)
24. endif

mpsi ≤ 192, vectorize over independent sums
mflux = 5, vectorize each vlen sum (reduction)
Vectorization of pushi

So this:

259. MV----------< do m=1,mi
260. MV          ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
261. MV          dtem(ip)=dtem(ip)+wpi(2,m)*zion(5,m)
262. MV          dden(ip)=dden(ip)+1.0
263. MV----------> enddo
Vectorization of pushi

Becomes this:

```c
#define _UNICOSMP

Vw V M------<><><> vdtem=0
f-------------<> vdden=0
m--------------<> do mv=1,mi,vlen
m MVs----------<> do m=mv,min(mv+vlen-1,mi)
m MVs v=m-mv+1
m MVs ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
m MVs vdtem(v,ip)=vdtem(v,ip)+wpi(2,m)*zion(5,m)
m MVs vdden(v,ip)=vdden(v,ip)+1.0
m MVs----------> enddo
m--------------> enddo
M--------------<> do i=1,mflux
M Vw V 4------<><><> dtem(i)=sum(vdtem(:,i))
M f-------------<> dden(i)=sum(vdden(:,i))
M--------------> enddo
```

Update temporaries
Sum results

And similarly for the larger code block
GTC Runtime

Time (seconds)

- XT3 r2
- X1E r2
- X1E r3
- X1E r5
- X1E r6
- X1E r8
- X1E r9
Profile r9

<table>
<thead>
<tr>
<th>16.3%</th>
<th>77.1%</th>
<th>651612</th>
<th>chargei_</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.4%</td>
<td>86.5%</td>
<td>375015</td>
<td>pushi_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6%</td>
<td>93.2%</td>
<td>64880</td>
<td>shifti_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6%</td>
<td>96.1%</td>
<td>22933</td>
<td>poisson_</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2%</td>
<td>98.1%</td>
<td>9468</td>
<td>poisson_initial_</td>
</tr>
</tbody>
</table>

Search for remaining low-hanging fruit
Tweaks

• pushi
  – Undo last example (!), not worth overhead
  – But keep analogous mods to larger code block

• shifti
  – Manually block pack operation for multistreaming
  – Move multistreaming away from `nparam` loops

• smooth
  – Add dimension to temporary to remove dependence
  – Block loop to use added dimension
Loopmarks for poisson

86. m iW M----------------< do i=1,mgrid
87. m iW M ptilde(i)=0.0
88. m iW M Vr----------< do j=1,nindex(i,k)
89. m iW M Vr ptilde(i)=ptilde(i) +ring(j,i,k)*phitmp(indexp(j,i,k))
90. m iW M Vr---------> enddo
91. m iW M-------------> enddo

nindex ≤ 65, reduction
mgrid = 32,449
Better to vectorize over i
Need constant j bounds
Vectorization of poisson

312. V M---------<><> max_nindex=maxval(nindex)

87. m iW MVR--------< do i=1,mgrid
88. m iW MVR ptilde(i)=0.0
89. m iW MVR #ifdef _UNICOSMP
90. m iW MVR r--------< do j=1,max_nindex
91. m iW MVR r #else
92. m iW MVR r do j=1,nindex(i,k)
93. m iW MVR r #endif
94. m iW MVR r ptilde(i)=ptilde(i)
   +ring(j,i,k)*phitmp(indexp(j,i,k))
95. m iW MVR r--------> enddo
96. m iW MVR--------> enddo

Do extra work (maybe) but vectorize and multistream
### Profile r12

<table>
<thead>
<tr>
<th>Samp%</th>
<th>Cum.Samp%</th>
<th>Samp</th>
<th>Function</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.0%</td>
<td>100.0%</td>
<td>3994895</td>
<td>Total</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Time for tougher changes to chargei (and others)**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16.3%</td>
<td>77.1%</td>
<td>651612</td>
</tr>
<tr>
<td>3.6%</td>
<td>64.5%</td>
<td>145767</td>
</tr>
<tr>
<td>2.3%</td>
<td>66.8%</td>
<td>93233</td>
</tr>
<tr>
<td>2.0%</td>
<td>68.8%</td>
<td>80616</td>
</tr>
<tr>
<td>1.9%</td>
<td>70.8%</td>
<td>77887</td>
</tr>
<tr>
<td>1.1%</td>
<td>71.9%</td>
<td>44914</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.4%</td>
<td>86.5%</td>
<td>375015</td>
</tr>
<tr>
<td>1.3%</td>
<td>78.4%</td>
<td>50273</td>
</tr>
<tr>
<td>1.2%</td>
<td>79.6%</td>
<td>48823</td>
</tr>
</tbody>
</table>
Dimension ordering in chargei

265. M--------------< do i=0,mpsi
266. M !dir$ prefervector
267. M V--------------< do j=1,mtheta(i)
268. M V r------------< do k=1,mzeta
269. M V r         ij=igrid(i)+j
270. M V r         zonali(i)=zonali(i)+0.25*densityi(k,ij)
271. M V r         densityi(k,ij)=0.25*densityi(k,ij)*marker(i,k,ij)
272. M V r------------> enddo
273. M V------------> enddo
274. M-------------> enddo

Vectorizing over second dimension of densityi and markeri
Change order of dimensions
Dimension ordering in charge

101. m MVs--------<  do m=mv,min(mv+vlen-1,mi)
102. m MVs            v=m-mv+1
...
112. m MVs 3--------<  do larmor=1,4
113. m MVs 3          wp1=wpion(larmor,m)
114. m MVs 3          wp0=1.0-wp1
...
140. m MVs 3          ij=jtion0(larmor,m)
141. m MVs 3          vdensityi(ij,kk,v) = vdensityi(ij,kk,v) + wz0*wt00
...
175. m MVs 3-------->  enddo
176. m MVs-------->  enddo
177. m              #ifdef _UNICOSMP
178. m--------->  enddo

Vectorizing over second dimension of wpion,jtion0, ...
(larmor loop is over vector operations)
Change order of dimensions
Changing dimension order

• Make vector dimension first

• Modified variables: densityi, dntmp, markeri, jtion0, jtion1, wpion, wtion0, wtion1, markere, densitye, zion, zion0

• Modified files (.F90): shifte, shifti, pushi, poisson, chargee, snapshot, setup, chargei, smooth, restart, tracking, set_random_values

• Changes affect all systems
  – Unlike directives and #ifdefs
  – Degrades performance on XT3
GTC Runtime

Time (seconds)

- XT3 r2
- X1E r2
- X1E r3
- X1E r5
- X1E r6
- X1E r8
- X1E r9
- X1E r12
- X1E r24
GTC on XT3

• Try running with small pages
  – yod -small_pages ...

• Effects of r24 dimension changes?

• Results for newest compiler and OS
GTC Runtime

Time (seconds)

- XT3 r2 6.0
- X1E r2
- X1E r3
- X1E r5
- X1E r6
- X1E r8
- X1E r9
- X1E r12
- X1E r24
- XT3 r2 6.1.3
- XT3 r2 small
- XT3 r24
- XT3 r24 small
Why do small pages help on XT3?

• GTC has irregular memory patterns
  – Particles move around

• Opteron TLB has just 8 entries for large pages
  – 512 entries for small pages

• GTC working set has more than 8 variables per loop

• TLB thrashing with large pages
Optimizing GTC for X1E and XT3

• X1E [performance improvement]
  – Vectorize overlapping updates by adding extra dimension [3.2x]
  – Replace if hierarchies with pack-like operations [1.4x]
  – Use prefervector directive to select loop for vectorization [1.8x]
  – Combine previous two techniques in a different procedure [1.6x]
  – Fix “less efficient” vectorization, use added-dimension trick [1.1x]
  – Vectorize outer loop, constant bounds (extra work) on inner [1.06x]
  – Change order of dimensions to make vector dimension first [1.26x]

• XT3
  – Run with small pages [1.4x, 2.15x for r24 (still slower)]
  – Don’t rearrange array dimensions [0.87x, 0.56x with large pages]
Compute Power of the Gyrokinetic Toroidal Code

Number of particles (in million) moved 1 step in 1 second

Latest vector optimizations
Not tested on Earth Simulator

S. Ethier, PPPL, Nov. 2005
Last slide*

• Caveat! Weak scaling!

• Cray X1E runs GTC fastest per processor
  – But it took lots of work to get there

• Earth Simulator still fastest overall
  – But XT3 is gaining

• 16,000-node BG/L similar to 1000-MSP X1E
  – BG/L computing on one processor per node
  – BG/L using 1/10 the particles per process

• 4000-processor XT3 beats 1000-MSP X1E
  – Production science runs on 4800 processors

* Except for the next one.
GTC science: Convergence study

- 400 particles/cell
- 28 billion particles
- 4800 XT3 processors
- Millions of processor hours

80 particles/cell vs. 200 particles/cell
Comparing Optimizations of GTC for the Cray X1E and XT3

James B. White III (Trey)
trey@ornl.gov

Nathan Wichmann, Cray
Stephane Ethier, PPPL