Comparing Optimizations of GTC for the Cray X1E and XT3

Scaling to New Heights

CUG 2006

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

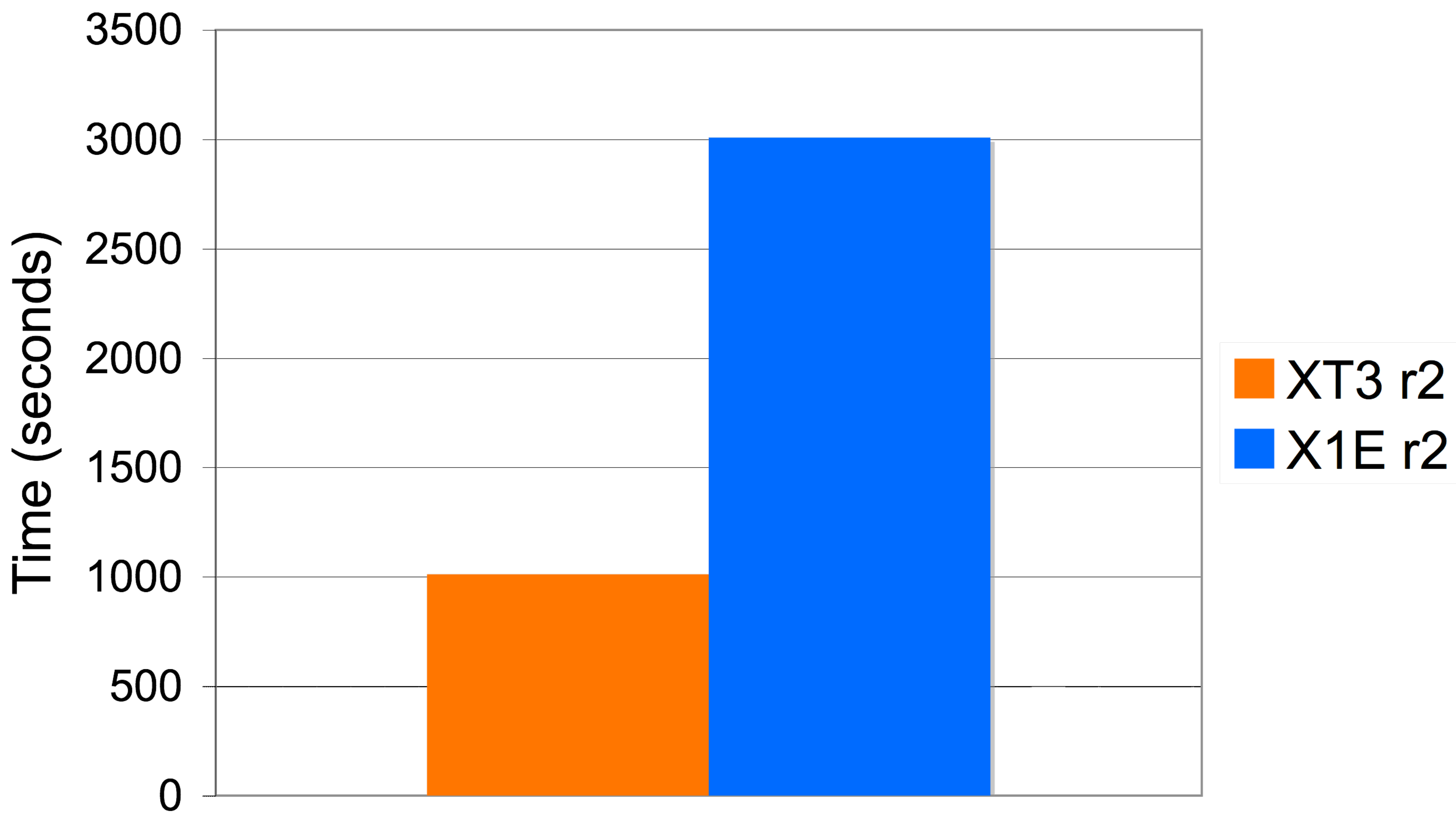
GTC

- 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

GTC Runtime



Profile r2

```
 Samp% | Cum.Samp% |      Samp |Function
                                |Line

100.0% |    100.0% | 22049624 |Total
|------------------------------------------
|  61.5% |     61.5% | 13557450 |chargei_
||-----------------------------------------
||  12.2% |     12.2% |  2698466 |line.140
||   9.2% |     21.5% |  2031530 |line.132
||   6.4% |     27.8% |  1404587 |line.102
||   4.8% |     32.6% |  1047693 |line.133
||   3.8% |     36.4% |   833888 |line.105
||   3.3% |     39.7% |   735487 |line.137
||   3.2% |     42.9% |   705596 |line.48
||   3.0% |     45.9% |   663002 |line.136
||   2.8% |     48.7% |   627426 |line.67
||   2.8% |     51.5% |   610015 |line.127
||   2.1% |     53.6% |   452091 |line.128
||   2.0% |     55.5% |   438478 |line.95
||   1.9% |     57.5% |   429772 |line.141
||   1.7% |     59.2% |   381375 |line.129
||   0.7% |     59.9% |   150991 |line.65
```

Loopmarks for chargei

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

Not permutations

```
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

```
...
                      #ifdef _UNICOSMP
                        integer, parameter :: vlen = 256
                        integer :: mv, v
                        real(wp) dnipart(mgrid,0:mzeta,vlen)
                      #endif
...
                      #ifdef _UNICOSMP
     m-------------<    do mv=1,mi,vlen
     m                  !dir$ prefervector
     m MVs---------<    do m=mv,min(mv+vlen-1,mi)
     m MVs                 v=m-mv+1
     m MVs            #else
     m MVs              do m=1,mi
     m MVs            #endif
```

Add a vector dimension

Vectorization of `chargei`

```
 ...
m MVs 3                 ij=jtion0(larmor,m)
m MVs 3                 dnipart(ij,kk,v) = dnipart(ij,kk,v) + wz0*wt00
m MVs 3                 dnipart(ij,kk+1,v)   = dnipart(ij,kk+1,v)   + wz1*wt00
 ...
ir------------<   do v=1,vlen
ir 2----------<      do kk=0,mzeta
ir 2                    !dir$ preferstream
ir 2 MV-------<         do ij=1,mgrid
ir 2 MV                    densityi(kk,ij) = densityi(kk,ij) + dnipart(ij,kk,v)
ir 2 MV------->         enddo
ir 2---------->      enddo
ir------------>   enddo
```

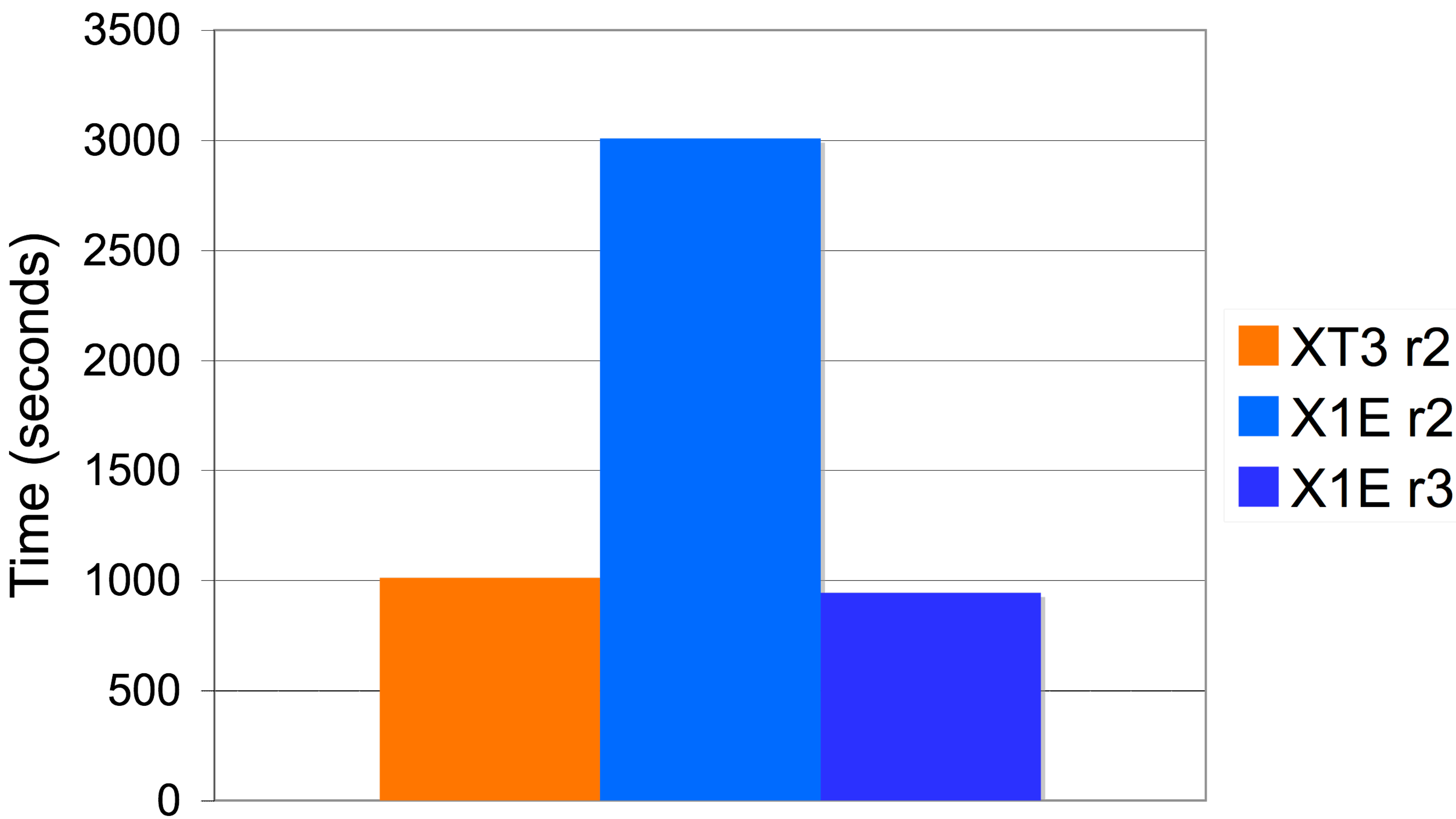
Sum over extra dimension

vlen=256

mzeta=1 for 64+ processors

mgrid=32,449

GTC Runtime


Time (seconds)
3500
3000
2500
2000
1500
1000
500
0
XT3 r2
X1E r2
X1E r3

Profile r3

```
  Samp% | Cum.Samp% |     Samp  |Function
                                |Line

 100.0% |    100.0% | 8768310 |Total
|----------------------------------------
|  30.3% |      30.3% | 2660905 |shifti_
||---------------------------------------
||  12.3% |      12.3% | 1080482 |line.154
||   7.3% |      19.7% |  644222 |line.206
||   6.1% |      25.8% |  535231 |line.128
||   2.1% |      27.9% |  186416 |line.207
||   1.4% |      29.3% |  120175 |line.132
||   0.2% |      29.5% |   17864 |line.190
...
|   7.5% |      93.1% |  660755 |chargei_
...
```

Loopmarks for `shifti`

Loop over particles that might move to another processor

```
r--------------<      do m=m0,mi
r                        zetaright=min(2.0*pi,zion(3,m))-zetamax
r                        zetaleft=zion(3,m)-zetamin
r
r                        if( zetaright*zetaleft > 0 )then
r                           zetaright=zetaright*0.5*pi_inv
r                           zetaright=zetaright-real(floor(zetaright))
r                           msend=msend+1
r                           kzi(msend)=m
r
```

Does it move to another processor?

Loopmarks for shifti

```
r                            if( zetaright < 0.5 )then
r               ! # of particle to move right
r                               msendright(1)=msendright(1)+1
r                               iright(msendright(1))=m
r               ! keep track of tracer
r                               if( nhybrid == 0 .and. m == ntracer )then
r                                  msendright(2)=msendright(1)
r                                  ntracer=0
r                               endif
r
r               ! # of particle to move left
r                            else
r                               msendleft(1)=msendleft(1)+1
r                               ileft(msendleft(1))=m
r                               if( nhybrid == 0 .and. m == ntracer )then
r                                  msendleft(2)=msendleft(1)
r                                  ntracer=0
r                               endif
r                            endif
r                         endif
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.
```

NATIONAL CENTER FOR COMPUTATIONAL SCIENCES

Vectorization of `shifti`

```
                    #elif defined _UNICOSMP

V-------------<          do m=m0,mi
V                           zetaright=min(2.0*pi,zion(3,m))-zetamax
V                           zetaleft=zion(3,m)-zetamin
V                           if (zetaright*zetaleft > 0) then
V                              msend=msend+1
V                              kzi(msend)=m
V                           endif
V------------->          enddo
```

Find indices of particles that will move

Similar to Fortran `pack` function

Vectorizes but doesn't multistream

Vectorization of `shifti`

```
V-------------<      do i=1,msend
V                       m=kzi(i)
V                       zetaright=min(2.0*pi,zion(3,m))-zetamax
V                       zetaright=zetaright*0.5*pi_inv
V                       zetaright=zetaright-real(floor(zetaright))
V                       if (zetaright < 0.5) then
V                 ! # of particle to move right
V                          msendright(1)=msendright(1)+1
V                          iright(msendright(1))=m
V                 ! # of particle to move left
V                       else
V                          msendleft(1)=msendleft(1)+1
V                          ileft(msendleft(1))=m
V                       endif
V------------->      enddo
```

Decide between sending left or right

Generalization of Fortran `pack` *function*

Vectorizes but doesn't multistream

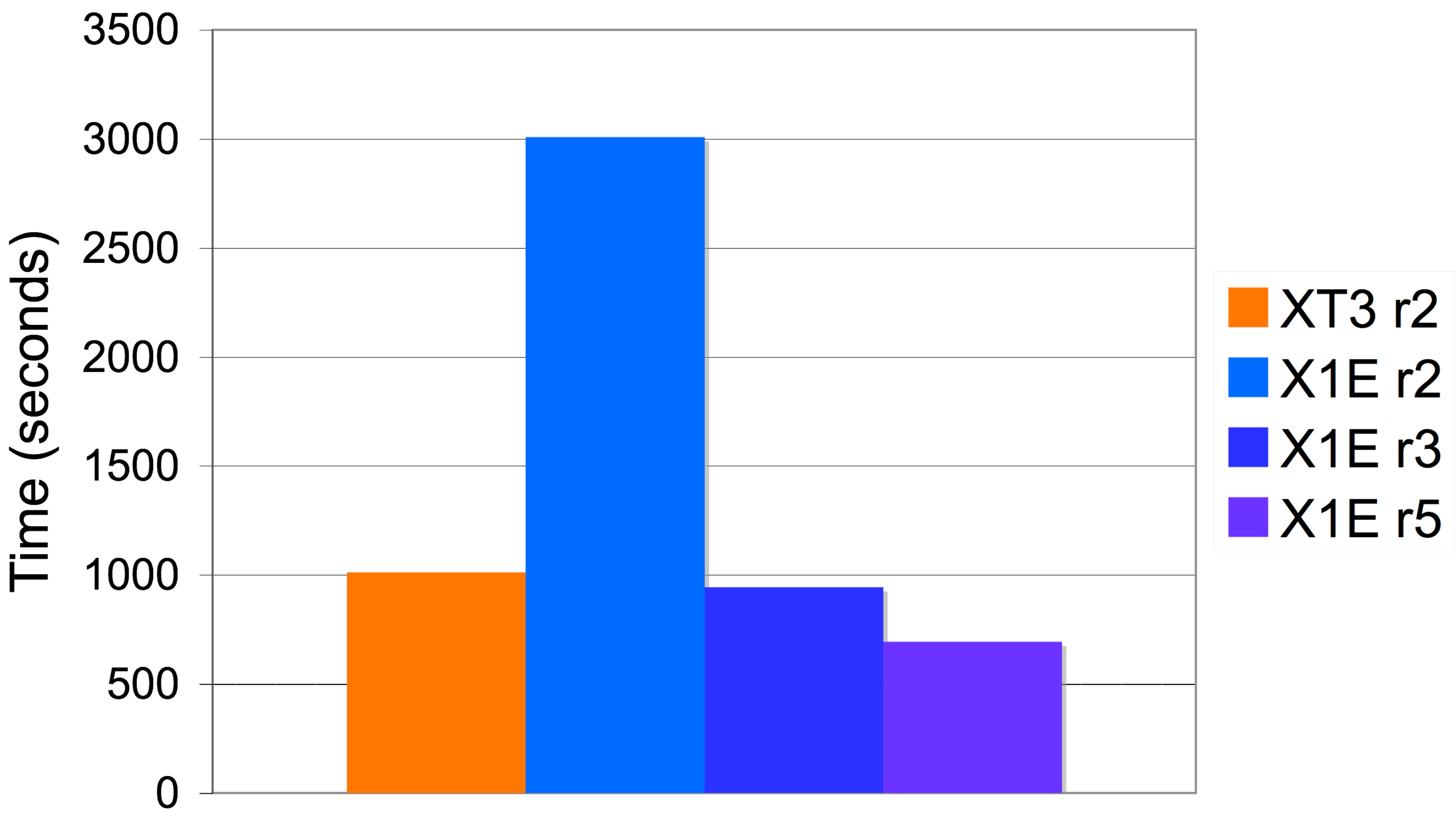
Vectorization of `shifti`

```
                    ! keep track of tracer
                         if ((nhybrid == 0) .and. (ntracer > 0)) then
MV------------<             do i=1,msendright(1)
MV                             if (iright(i) == ntracer) msendright(2)=i
MV------------>             enddo
                            if (msendright(2) /= 0) then
                               ntracer=0
                            else
MV------------<                do i=1,msendleft(1)
MV                                if (ileft(i) == ntracer) msendleft(2)=i
MV------------>                enddo
                               if (msendleft(2) /= 0) ntracer=0
                            endif
                         endif
```

See if any sent particles are the tracer

Reductions (like `sum`) of `logical` values

GTC Runtime


3500
3000
2500
2000
1500
1000
500
0
Time (seconds)
XT3 r2
X1E r2
X1E r3
X1E r5

Profile r5

```
 Samp% | Cum.Samp% |    Samp |Function
                             |Line

 100.0% |    100.0% | 7126472 |Total
|------------------------------------------
|  34.0% |     34.0% | 2422227 |pushi_
||----------------------------------------
||   9.6% |      9.6% |  683490 |line.93
||   4.6% |     14.2% |  330526 |line.80
||   4.3% |     18.6% |  309770 |line.92
||   3.8% |     22.4% |  269578 |line.94
||   1.9% |     24.2% |  131984 |line.62
||   1.5% |     25.7% |  107877 |line.148
||   0.9% |     26.6% |   62389 |line.105
...
|  13.7% |     81.3% |  973509 |shifti_
...
|   9.2% |     90.5% |  654013 |chargei_
```

Loopmarks for pushi

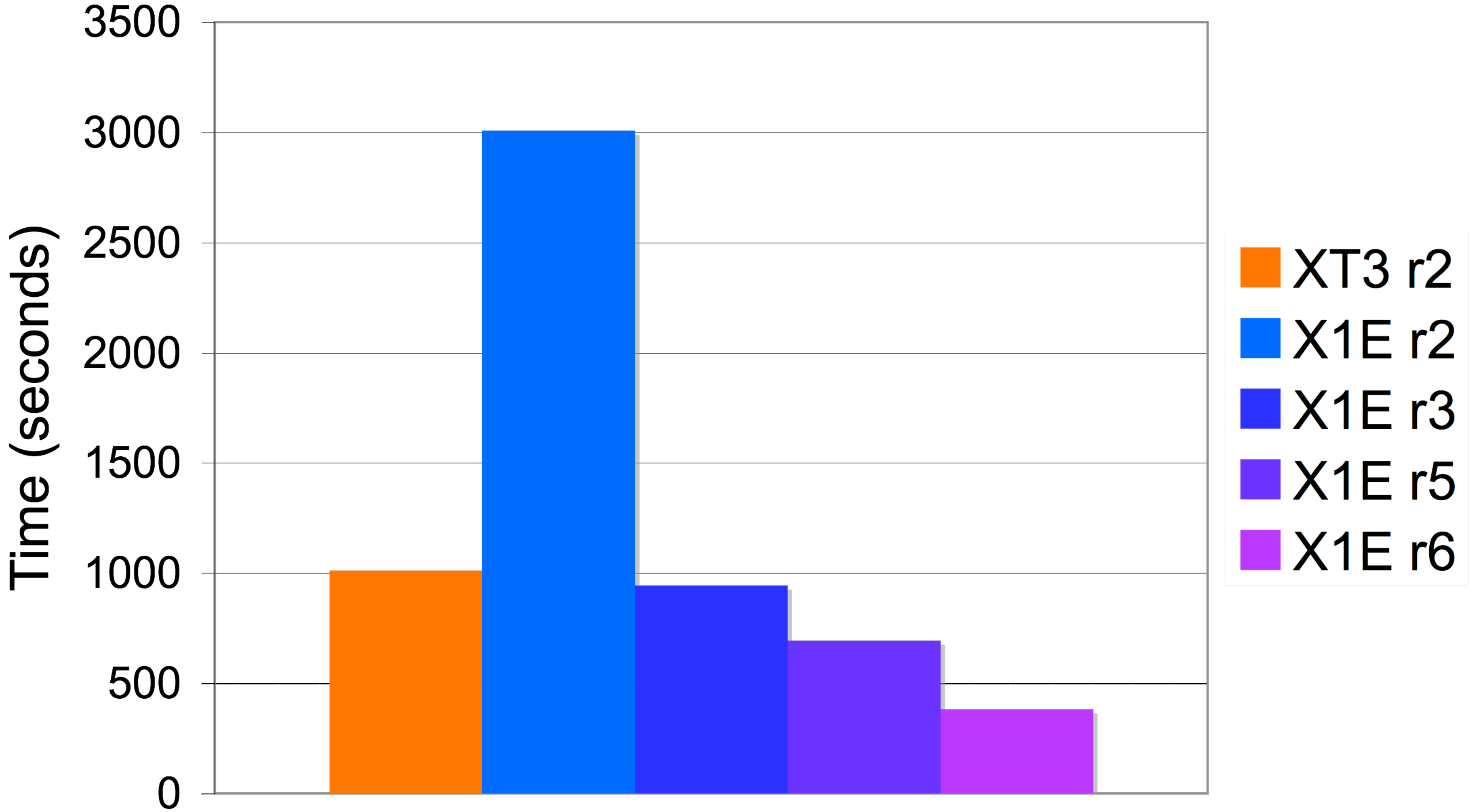
```
 60.  M----------<    do m=1,mi
...
 67.  M
 68.  M Vs--------<      do larmor=1,4
 69.  M Vs
 70.  M Vs                 ij=jtion0(larmor,m)
 71.  M Vs                 wp0=1.0-wpion(larmor,m)
 72.  M Vs                 wt00=1.0-wtion0(larmor,m)
 73.  M Vs                 e1=e1+wp0*wt00*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij)
 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                  wpi(1,m)=0.25*e1
 99.  M                  wpi(2,m)=0.25*e2
100.  M                  wpi(3,m)=0.25*e3
101.  M
102.  M---------->    enddo
```

Vector length of 4?!

Vectorization for pushi

```
                    !dir$ prefervector
...
MV----------<   do m=1,mi
...
MV 2--------<      do larmor=1,4
MV 2
MV 2                  ij=jtion0(larmor,m)
MV 2                  wp0=1.0-wpion(larmor,m)
MV 2                  wt00=1.0-wtion0(larmor,m)
MV 2                  e1=e1+wp0*wt00*(wz0*evector(1,kk,ij)+wz1*evector(1,kk+1,ij)
MV 2                  e2=e2+wp0*wt00*(wz0*evector(2,kk,ij)+wz1*evector(2,kk+1,ij)
MV 2                  e3=e3+wp0*wt00*(wz0*evector(3,kk,ij)+wz1*evector(3,kk+1,ij)
...
MV 2-------->      enddo
MV
MV                 wpi(1,m)=0.25*e1
MV                 wpi(2,m)=0.25*e2
MV                 wpi(3,m)=0.25*e3
MV
MV---------->   enddo
```

GTC Runtime


3500
3000
2500
2000
1500
1000
500
0
Time (seconds)
XT3 r2
X1E r2
X1E r3
X1E r5
X1E r6

Profile r6

```
 Samp% | Cum.Samp% |     Samp |Function
                                |Line

 100.0% |    100.0% | 5146367 |Total
|-----------------------------------------
|  25.4% |      25.4% | 1306364 |MPI_CRAY_fast_barrier
...
||========================================
|  22.2% |      47.6% | 1144908 |rng_number_s1_
...
||========================================
|  19.2% |      66.9% |  989615 |shifti_
||----------------------------------------
||  12.3% |      59.9% |  631146 |line.248
||   4.2% |      64.1% |  214897 |line.249
||   1.0% |      65.1% |   51156 |line.128
...
```

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

Loop over sent particles

```
240.                       lasth=msend
241.  1-------------<      do i=1,msend
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
```

nparam = 6 *or* 7

```
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

```
                         #ifdef _UNICOSMP
r V M-----<><><>               zmask(mi+1:mtop) = .true.
V M---------<><>               i0=count(kzi(1:msend) <= mi)
r V M-----<><><>               zmask(kzi(i0+1:msend)) = .false.
                               nholes=0
V--------------<               do m=mtop,mi+1,-1
V                                 if (zmask(m)) then
V                                    nholes=nholes+1
V                                    jzi(nholes)=m
V                                 endif
V-------------->               enddo
```

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`

```
                           !dir$ concurrent, prefervector
Vm-------------<           do i=1,nholes
Vm r 3 M--<><><>              zion(1:nparam,kzi(i))=zion(1:nparam,jzi(i))
Vm f----------<>              zion0(1:nparam,kzi(i))=zion0(1:nparam,jzi(i))
Vm------------->           enddo
                           if (nhybrid == 0 .and. ntracer > mi) then
                              itracer=0
MV-------------<              do i=1,nholes
MV                               if (ntracer == jzi(i)) itracer=i
MV------------->              enddo
                              ntracer=kzi(itracer)
                           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

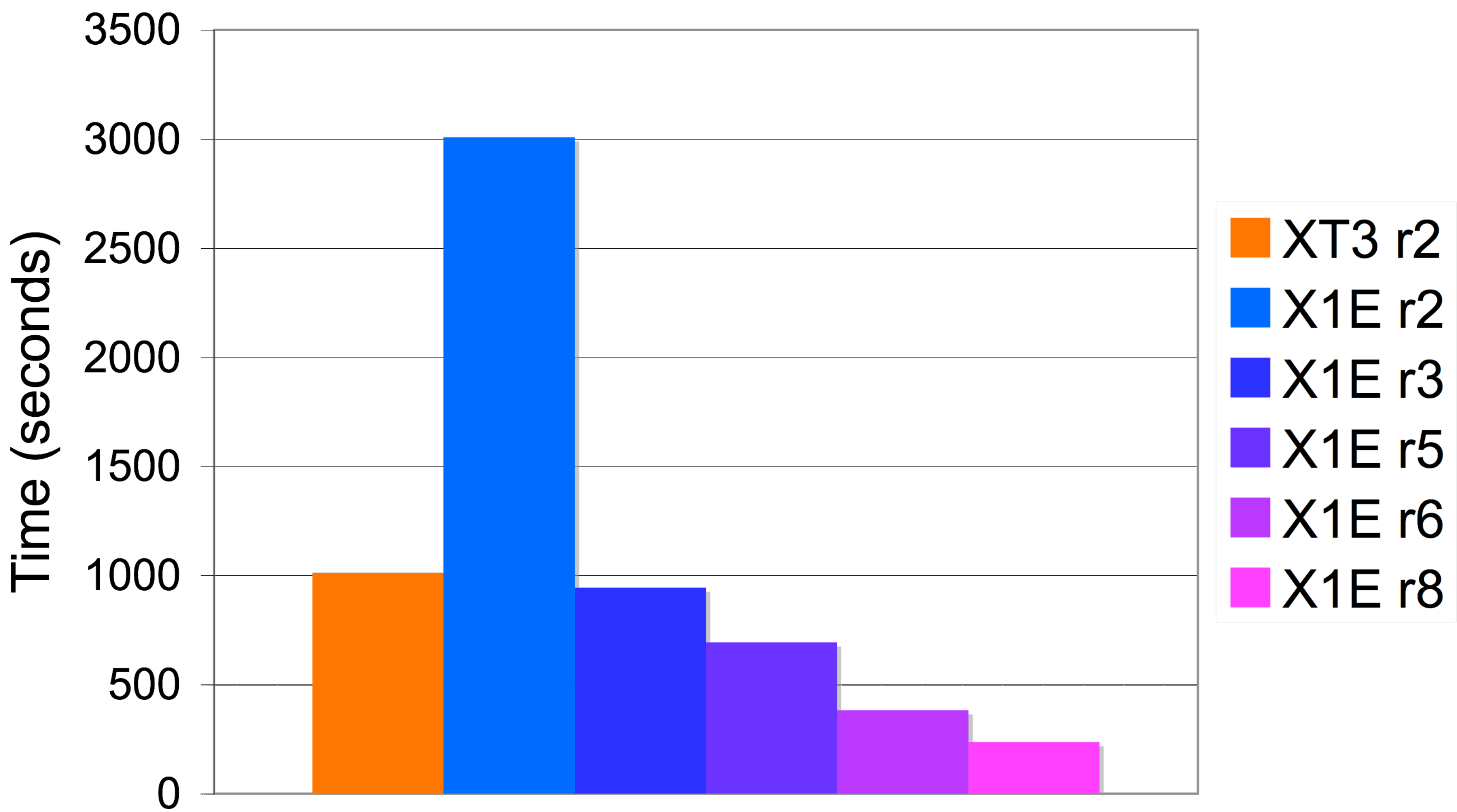
Vectorization of `shifti`

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

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

Vectorize over particles, multistream over `nparam`

GTC Runtime


3500
3000
2500
2000
1500
1000
500
0
Time (seconds)
XT3 r2
X1E r2
X1E r3
X1E r5
X1E r6
X1E r8

Profile r8

Now revisit chargei *and* pushi

No obvious fixes for chargei

```
 Samp% | Cum.Samp% |     Samp |Function
                               |Line

100.0% |    100.0% | 4205531 |Total
|-----------------------------------------
|  30.9% |      30.9% | 1297644 |MPI_CRAY_fast_barrier
||----------------------------------------
...
||========================================
|  26.8% |      57.7% | 1127354 |rng_number_s1_
||----------------------------------------
...
||========================================
|  15.5% |      73.2% |  652792 |chargei_
||----------------------------------------
...
||========================================
|  11.6% |      84.7% |  486147 |pushi_
||----------------------------------------
...
||========================================
|   2.4% |      92.4% |  100005 |shifti_
||----------------------------------------
...
```

Loopmarks for pushi

Looks OK, right?

```
MV----------<           do m=1,mi
MV                         ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
MV                         dtem(ip)=dtem(ip)+wpi(2,m)*zion(5,m)
MV                         dden(ip)=dden(ip)+1.0
MV---------->           enddo

MV----------<     do m=1,mi
 ...
MV                   ip=max(1,min(mflux,1+int((r-a0)*d_inv)))
MV                   ii=max(0,min(mpsi,int((r-a0)*delr+0.5)))
 ...
MV                   hfluxpsi(ii)=hfluxpsi(ii)+vdrenergy ! energy flux profile
 ...
MV                   rmarker(ip)=rmarker(ip)+zion0(6,m)
MV                   eflux(ip)=eflux(ip)+vdrenergy
 ...
MV                   dmark(ip)=dmark(ip)+wpi(1,m)*r
MV                   dden(ip)=dden(ip)+1.0
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.
```

Loopmarks for pushi

Not permutations!

```
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

301.  MV----------<     do m=1,mi
 ...
305.  MV                   ip=max(1,min(mflux,1+int((r-a0)*d_inv)))
306.  MV                   ii=max(0,min(mpsi,int((r-a0)*delr+0.5)))
 ...
310.  MV                   hfluxpsi(ii)=hfluxpsi(ii)+vdrenergy ! energy flux profile
 ...
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
```

Vectorization of pushi

Add a vector dimension, like for `chargei`

```
#ifdef _UNICOSMP
  integer, parameter :: vlen=256
  integer :: mv, v
  real(wp) :: vdtem(vlen,mflux), vdden(vlen,mflux)
  real(wp) :: vhfluxpsi(0:mpsi,vlen),
           vrmarker(vlen,mflux), veflux(vlen,mflux),
           vdmark(vlen,mflux)
#endif
```

`mpsi` ≤ 192, *vectorize over independent sums*

`mflux` = 5, *vectorize each* `vlen` *sum (reduction)*

Vectorization of pushi

So this:

```
MV----------<            do m=1,mi
MV                          ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
MV                          dtem(ip)=dtem(ip)+wpi(2,m)*zion(5,m)
MV                          dden(ip)=dden(ip)+1.0
MV---------->            enddo
```

Vectorization of pushi

Becomes this:

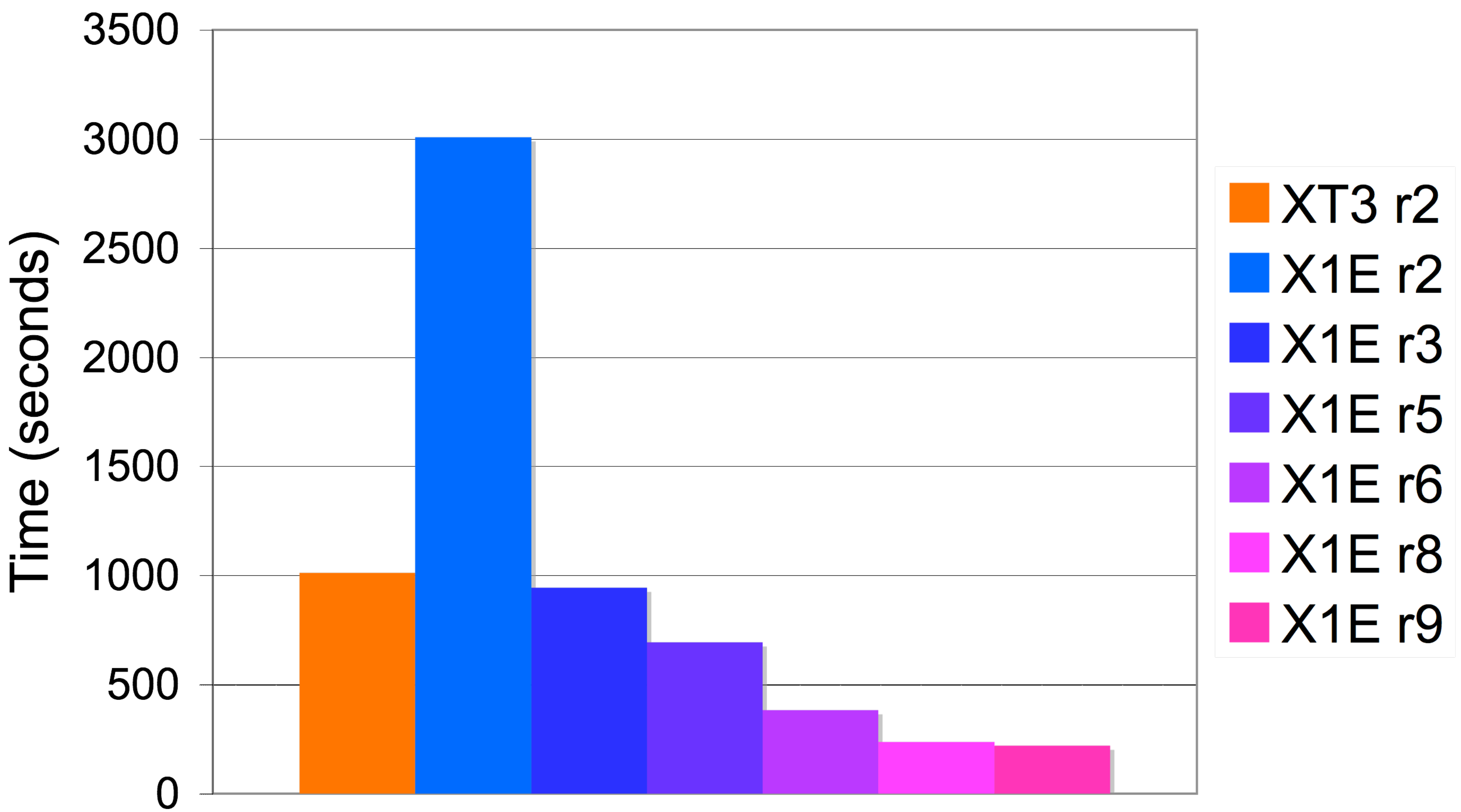
```
                        #ifdef _UNICOSMP
Vw V M----<><><>            vdtem=0
f-------------<>            vdden=0
m--------------<            do mv=1,mi,vlen
m MVs----------<               do m=mv,min(mv+vlen-1,mi)
m MVs                             v=m-mv+1
m MVs                             ip=max(1,min(mflux,1+int((wpi(1,m)-a0)*d_inv)))
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)
3500
3000
2500
2000
1500
1000
500
0
XT3 r2
X1E r2
X1E r3
X1E r5
X1E r6
X1E r8
X1E r9

Profile r9

```
|  16.3% |      77.1% |  651612 |chargei_
...
|   9.4% |      86.5% |  375015 |pushi_
...
|   1.6% |      93.2% |   64880 |shifti_
...
|   0.6% |      96.1% |   22933 |poisson_
...
|   0.2% |      98.1% |    9468 |poisson_initial_
```

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`

```
m iW M-----------<          do i=1,mgrid
m iW M                         ptilde(i)=0.0
m iW M Vr--------<             do j=1,nindex(i,k)
m iW M Vr                         ptilde(i)=ptilde(i)
                                   +ring(j,i,k)*phitmp(indexp(j,i,k))
m iW M Vr-------->             enddo
m iW M----------->          enddo
```

`nindex` ≤ 65, *reduction*

`mgrid` = 32,449

Better to vectorize over `i`

Need constant `j` *bounds*

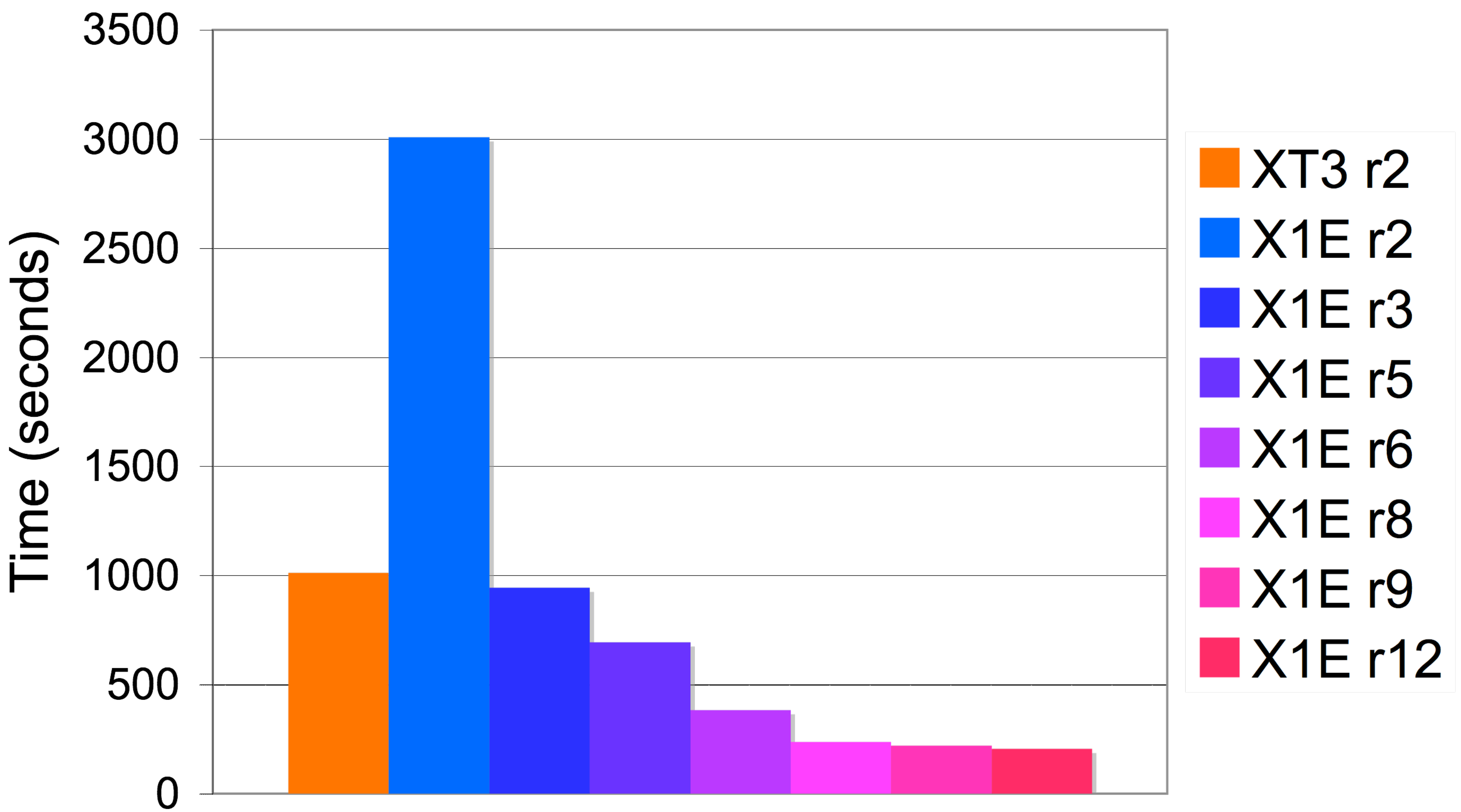
Vectorization of poisson

```
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

GTC Runtime


Time (seconds)
3500
3000
2500
2000
1500
1000
500
0
XT3 r2
X1E r2
X1E r3
X1E r5
X1E r6
X1E r8
X1E r9
X1E r12

Profile r12

Time for tougher changes to chargei (and others)

```
  Samp% | Cum.Samp% |    Samp |Function
                               |Line

 100.0% |    100.0% | 3994895 |Total
|---------------------------------------
...
||======================================
|  16.3% |     77.1% |  651612 |chargei_
||--------------------------------------
||   3.6% |     64.5% |  145767 |line.120
||   2.3% |     66.8% |   93233 |line.76
||   2.0% |     68.8% |   80616 |line.55
||   1.9% |     70.8% |   77887 |line.153
||   1.1% |     71.9% |   44914 |line.149
...
||======================================
|   9.4% |     86.5% |  375015 |pushi_
||--------------------------------------
||   1.3% |     78.4% |   50273 |line.99
||   1.2% |     79.6% |   48823 |line.100
```

Dimension ordering in `chargei`

```
M-------------<   do i=0,mpsi
M                    !dir$ prefervector
M V-----------<      do j=1,mtheta(i)
M V r---------<         do k=1,mzeta
M V r                      ij=igrid(i)+j
M V r                      zonali(i)=zonali(i)+0.25*densityi(k,ij)
M V r                      densityi(k,ij)=0.25*densityi(k,ij)*markeri(k,ij)
M V r--------->         enddo
M V----------->      enddo
M------------->   enddo
```

Vectorizing over second dimension of `densityi` *and* `markeri`

Change order of dimensions

Dimension ordering in `chargei`

```
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`*, …*

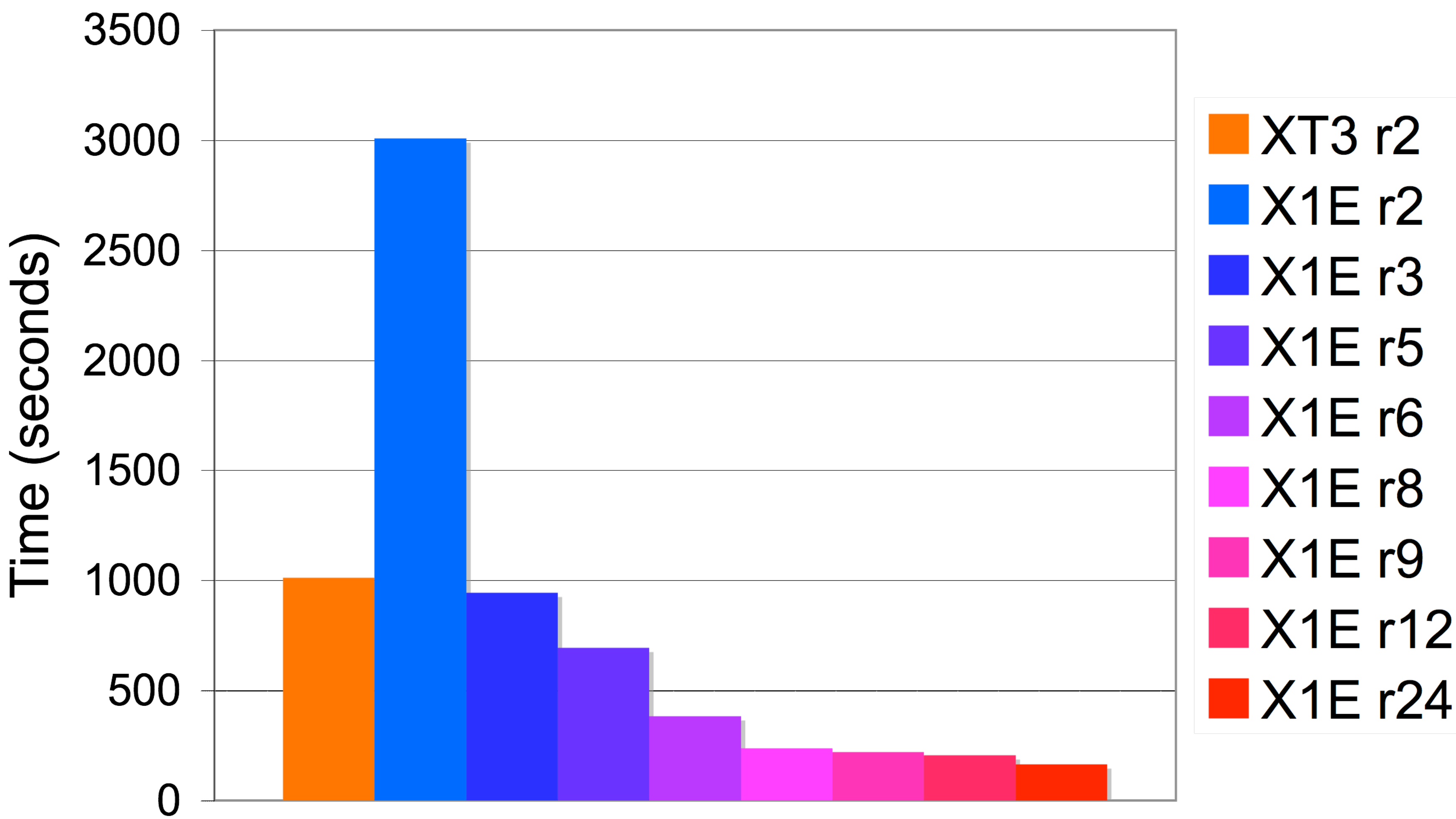
(`larmor` *loop is over vector operations)*

Change order of dimensions

Changing dimension order

- Make vector dimension first
- Modified variables: `densityi, dnitmp, 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

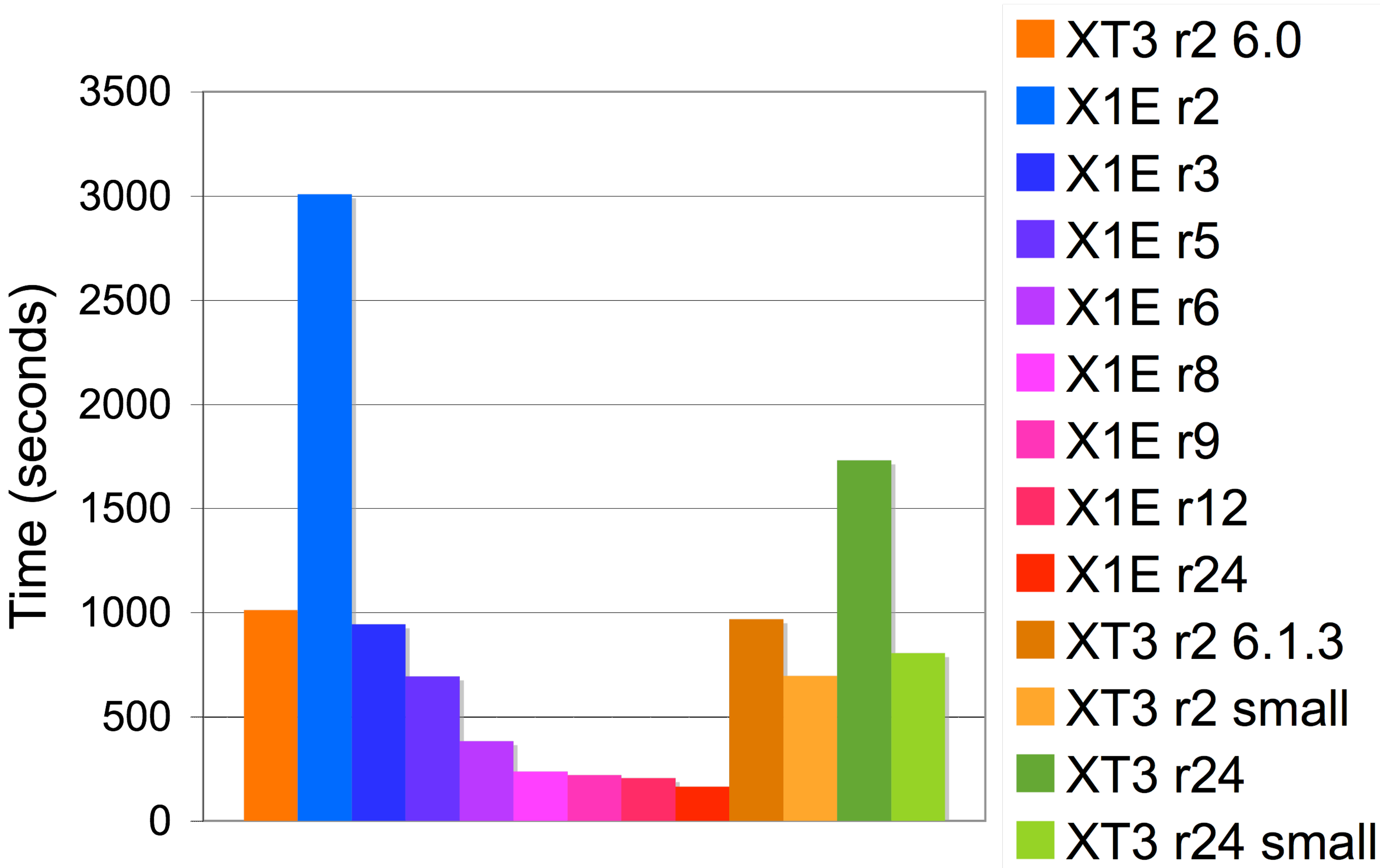
GTC Runtime


Time (seconds)
3500
3000
2500
2000
1500
1000
500
0
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



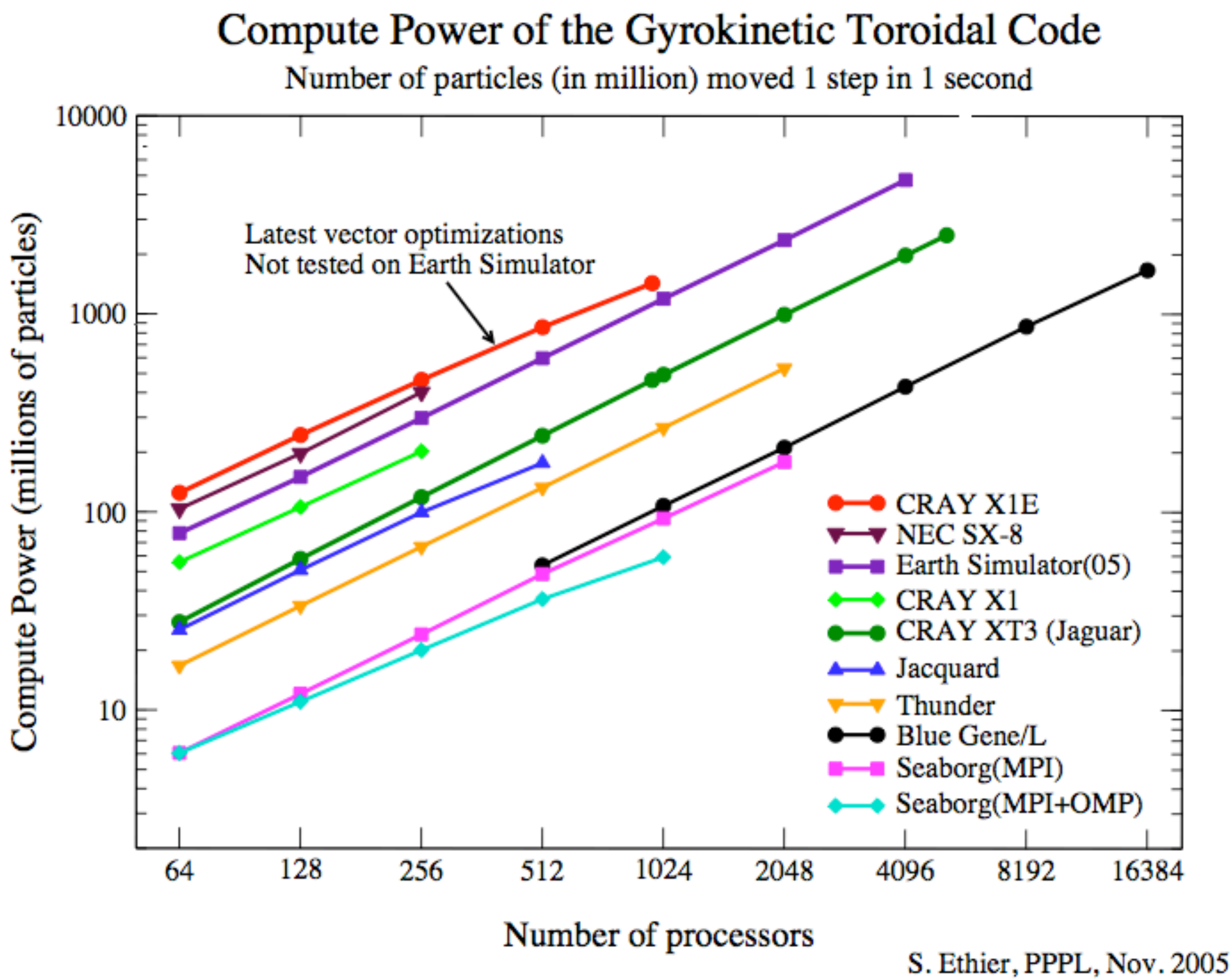
Time (seconds)
3500
3000
2500
2000
1500
1000
500
0
XT3 r2 6.0
X1E r2
X1E r3
X1E r5
X1E r6
X1E r8
X1E r9
X1E r12
X1E r24
XT3 r2 6.1.3
XT3 r2 small
XT3 r24
XT3 r24 small

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]

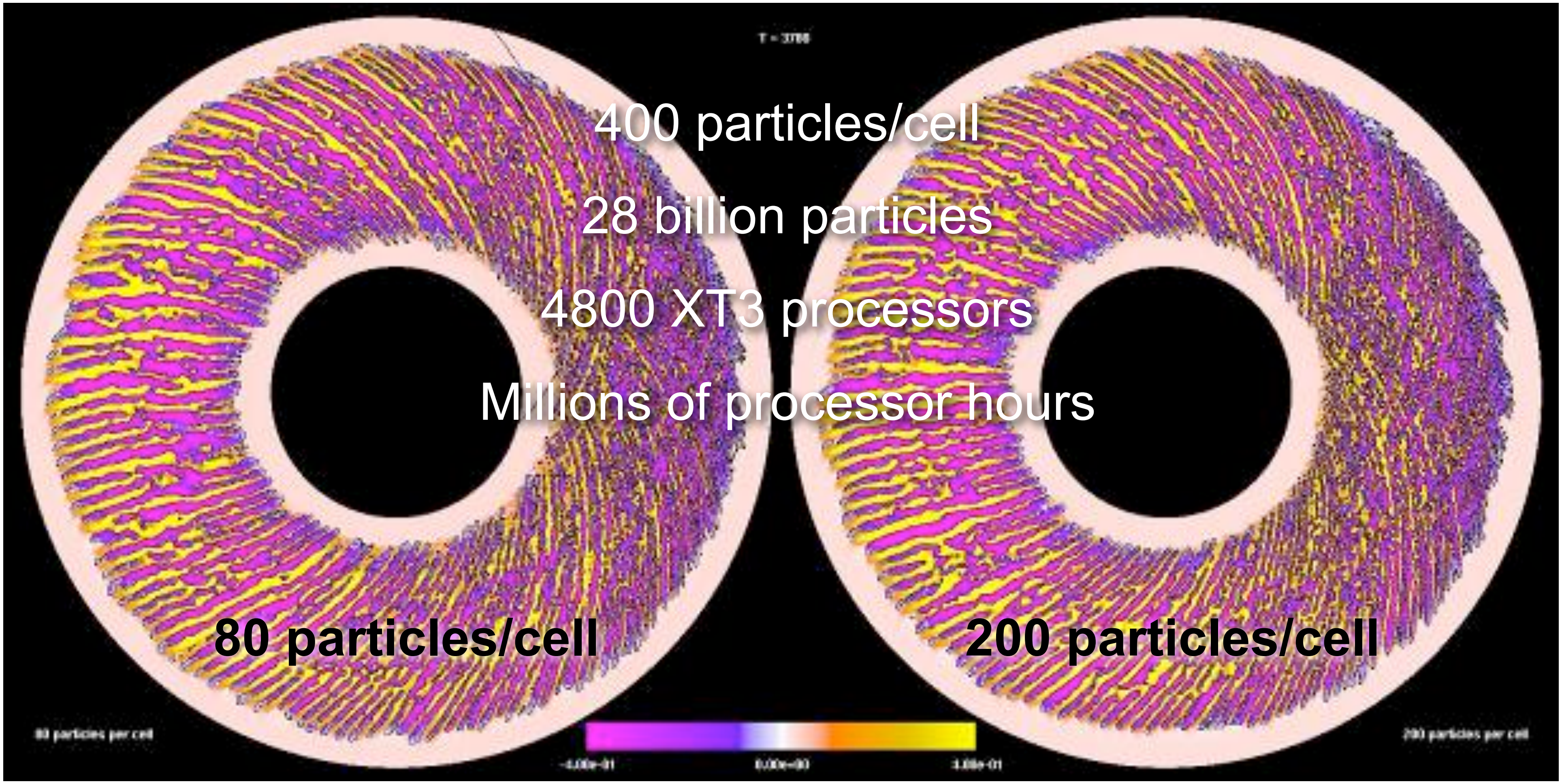

Compute Power of the Gyrokinetic Toroidal Code
Number of particles (in million) moved 1 step in 1 second
Latest vector optimizations
Not tested on Earth Simulator
Compute Power (millions of particles)
10000
1000
100
10
64
128
256
512
1024
2048
4096
8192
16384
Number of processors
CRAY X1E
NEC SX-8
Earth Simulator(05)
CRAY X1
CRAY XT3 (Jaguar)
Jacquard
Thunder
Blue Gene/L
Seaborg(MPI)
Seaborg(MPI+OMP)
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



Comparing Optimizations of GTC for the Cray X1E and XT3

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

Nathan Wichmann, Cray
Stephane Ethier, PPPL