פרויקט בנושאים מתקדמים 6

אילון טל

322793910

מקבול CPU

על ההתחלה ניגשתי לשיפור מהיר דרך אופטימיזציות קומפיילר. הוספתי O3, -march=native כדי שיקמפל לארכיטקטורה שנתונה ויבצע אופטימיזציות לכך בעצמו. ניסיתי להוסיף דגלי SIMD – עמדתי להוסיף -ftree-vectorize אבל אז הבנתי שהוא כבר בO3. מצאתי את -ffast-math שהתאים לי ממש כי הוא עושה וקטוריזציה לפעולות float ובאמת יש המון כאלו בתכנית. הדגל האיץ לי בהמון את ההרצה (: אבל עשה המון שגיאות בקובץ של הvelocity :(. לא מצאתי דגלי וקטוריזציה רחבים כמו בקומפיילר של אינטל (השתמשתי בGNU כי היו לי מלא בעיות טכניות עם אינטל). ניסיתי גם בסוף הוספתי גם את -fno-math-errno כי קראתי שהוא יכול לעשות כל מיני אופטימיזציות נוספות למתמטיקה שיש המון בקוד, לא בטוח כמה זה עזר. ניסיתי גם -fassociative-math שלא עבד (כנראה התנגש עם משהו אחר). הוספתי גם את הדגל -msse2 שאמור לייצר וקטוריזציות נוספות בx64. סה"כ פקודת הקמפול:

```
if (CMAKE_Fortran_COMPILER_ID STREQUAL "GNU")
set(ENV{JSONFORTRAN} "~/json-fortran-example/json-fortran/build")
set(CMAKE_Fortran_FLAGS "-fPIC -cpp -O3 -fopenmp -msse2  -fno-math-errno -march=native -ffree-line-length-512")
else()
```

עכשיו פתחתי VTUNE ומצאתי בערך את המספרים הבאים (הם לא שמורים אצלי אז זה מה שאני זוכר):

Velocity – calculate derivatives	45%
Vertex mass – calculate vertex mass 3d	13%
Geometry- x proj	8%
Advect – calculate advect 3d	5%

ובהמשך היו לי עוד כמה מהמודל של גיאומטריה

החלטתי באופן טבעי להתחיל מcalculate derivatives. (הגיוני שבפעולות מתמטיות ילך הרבה זמן על חישוב נגזרות והייתה לי תחושה שזה יהיה מתאים למקבל).

```
!$omp parallel do collapse(3) num_threads(24) default(private) shared(nz,ny,nx,emf,dvel_x_dx,dvel_x_dy,dvel_x_dz) &
!$omp shared(dvel_y_dx,dvel_y_dy,dvel_y_dz,dvel_z_dx,dvel_z_dy,dvel_z_dz,velocity_x,velocity_y,velocity_z,x,y,z,vol,vof)
do k = 1, nz
    do j = 1, ny
        do i = 1, nx
            ip = i + 1
            jp = j + 1
```

ראיתי את הלולאה המקוננת והחלטתי להכניס מקבול קלאסי. מכיוון שהיו המון משתנים בתוך הלולאה שפרטיים לכל חוט החלטתי לעשות שהדיפולטיבי יהיה shared. חילקתי ל24 חוטים כי זהו מספר הליבות בפועל (hyperthreading). זה היה שיפור משמעותי מאוד בריצות. לקח לי בערך משהו בזמנים הבאים:

```
Cycle time:    1.9850372634828091
Cycle time:    2.0909837819635868
Cycle time:    2.0046888589859009
Cycle time:    1.9692413005977869
Cycle time:    1.9977749139070511
Cycle time:    1.9819984361529350
Cycle time:    1.9825180098414421
Cycle time:    1.9856570344418287
Cycle time:    1.9579121246933937
Cycle time:    1.9854083359241486
Total Time:   19.967929162085056
ncyc:           10
```

בפונקציה מתחת impose spherical symmetry ניסיתי למקבל לולאות וזה עזר בצורה מעטה יחסית לזמנים (אני מתעד בסדר לא כרונולוגי אלא שיהיה קל לעקוב מבחינת מודולים).

```
call coords%Point_to_data(x, y, z)
do k = 1, nzp
    k1 = virt_k_start + k
    !$omp parallel do num_threads(24) collapse(2) default(shared) private(vel_x_tmp,vel_y_tmp,vel_z_tmp)
    do j = 1, nyp
        do i = 1, nxp
                vel_x_tmp = velocity_x(i, j, k)
                vel_y_tmp = velocity_y(i, j, k)
                vel_z_tmp = velocity_z(i, j, k)
            vel_radial_vec(k1) = vel_radial_vec(k1) + (vel_x_tmp * x(i, j, k)  &
                                                     +  vel_y_tmp * y(i, j, k)  &
                                                     +  vel_z_tmp * z(i, j, k)) &
                                                     / (x(i,j,k) * x(i,j,k) + y(i,j,k)*y(i,j,k) + z(i,j,k) * z(i,j,k))

            index_vertex_vec(k1) = index_vertex_vec(k1) + 1

        end do
    end do
end do

do k = 1, nzp
```

כעת ניגשתי למקבל את המודל של vertex mass. הוספתי מקבול לcalculate_vertex_mass_3d ולאחר מכן גם לd2 (מתוך מחשבה שאולי זה יעזור כי משתמשים בשניהם הרבה):

:3d

```
   ! better to collapse(3) here because last iteration is small, needs to add t to private
!$omp parallel do num_threads(24) collapse(3) default(shared) private(t,kk,jj,ii,i1,i2,i3,j1,j2,j3,k1,k2,k3)
do k = 1, nzp
   do j = 1, nyp
      do i = 1, nxp
         do t = 1, 8
            kk = k - (8 - t) / 4
            jj = j - mod(t + 1, 4) / 2
            ii = i - mod(t, 2)
            if (cell_mass(ii, jj, kk) /= 0d0) then
               i1 = i + i_tetr(t, 1)
               i2 = i + i_tetr(t, 2)
```

:2d

```
    vertex_mass(1:nxp, 1:nyp, 1) = 0d0
  !$omp parallel do num_threads(24) collapse(2) default(private) shared(cyl,omcyl,k)
  do j = 1, nyp
     do i = 1, nxp
        x1 = x(i, j, 1)
        y1 = y(i, j, 1)
        r_factor_1 = x1 * cyl + omcyl
```

זה שיפר לי בעוד קצת את הזמנים. ראוי לשים לב שעשיתי collapse3 בd3 כי הלולאה התחתונה היא רק 8 איטרציות אז כשעושים collapse4 זה מעלה את הזמנים. ניסיתי כל מיני שיטות scheduling אבל דווקא הדיפולטיבי יצא הכי טוב.

כעת בproj x geometry ראיתי שאין שם שום לולאה, אז ניסיתי פשוט להכריז על זה כsimd וזה שיפר קצת, כל פונקציה אחרת שהוספתי לה simd בgeometry העלתה זמנים.

```
 making sedov-taylor
done mesh
Done diagnostics
finished building problem
Cycle time:    1.9671636763960123
Cycle time:    2.0641229748725891
Cycle time:    1.9818894397467375
Cycle time:    1.9579944666475058
Cycle time:    1.9425282925367355
Cycle time:    1.9398146327584982
Cycle time:    1.9566407781094313
Cycle time:    1.9659271296113729
Cycle time:    1.9679288286715746
Cycle time:    1.9930250830948353
Total Time:   19.768023077398539
ncyc:           10
```

ניסיתי למקבל את volume – calculate כי ראיתי בvtune שהוא הבא בתור אבל כל גם היא וגם כל פונקציה נוספת שניסיתי למקבל שזמן הריצה שלה קטן או שווה בסה"כ, התקורה על המקבול שלהן הייתה יותר משמעותית מאשר ההרצה הסדרתית שלה ולכן הזמנים עלו בהרבה. בגלל זה החלטתי שזה יהיה מקום טוב להפסיק.

מקבול GPU

בשביל לדעת מה לקבל בgpu נרצה למצוא קטע שנוח לבצע עליו simd הקטע הכי טוב לזה הוא בcalculate derivatives גם בגלל המסה שלו מבחינת זמן חישוב וגם מבחינת העובדה שהוא בעיקר חישובים מתמטיים שניתן להפוך לsimd כלומר ב"ת זה בזה (אולי גם היה אפשר בCalculate_Vertex_mass_3d אבל קצת קשה לדעת אם זה היה עובד טוב בגלל כל התנאים שם וכל העבודה שאינה מתבצעת בחישובים פשוטים יחסית).
ניתן לעשות זאת על ידי שורת הפראגמה הבאה: (נמפה למכשיר את כל מה שצריך לחישוב ונחזיר מה שחישבנו)

```
!$omp target  teams loop collapse(3) map(to:nx,ny,nz,x,y,z,velocity_x,velocity_y,velocity_z,vol)
map(from: dvel_z_dx,dvel_z_dy,dvel_z_dz, dvel_y_dx dvel_y_dy dvel_y_dz, dvel_x_dx dvel_x_dy
dvel_x_dz)
```

בונוס