OpenMP and the Stommel Model
Timothy H. Kaiser, Ph.D.
tkaiser@mines.edu
Talk Overview

- What we look for when using OpenMP
- Why the Stommel model is a good candidate for OpenMP
- Timings for serial version
- Adding our directive
- Pitfalls. What happened?
- Fixing the problem
- Final runtimes
What we look for when using OpenMP

• Do a profile to find time dominant routines
• Better yet find time dominant loop
  • Best case:
    • No dependency between iterations
    • A lot of work in each iteration
  • May want to rewrite your loops
    • Remove dependencies
    • Merge loops to reduce synchronization costs
  • Can you rewrite your program in a task level parallel fashion?
Why OpenMP?

- Stommel Model well suited for OpenMP
- Single nested do loop dominates the runtime
- No dependencies between iterations

```fortran
subroutine do_jacobi(psi,new_psi,diff,i1,i2,j1,j2)
  ...
  ...
  do j=j1,j2
    do i=i1,i2
      new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
      a3*psi(i,j+1) + a4*psi(i,j-1) - &
      a5*for(i,j)
      diff=diff+abs(new_psi(i,j)-psi(i,j))
    enddo
  enddo
psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
```
Some Results

- These are “old” results
- At the time OpenMP compilers were immature
- Illustrates some points
  - Check your results
- You may have lied to the compiler
  - It might not be working correctly
- Check your run times
Serial runtime

- Run on Dec Alpha
- Fast processors 533 MHz Alpha
- 4-way SMP nodes
- Good OpenMP support
- `f90 stf_00.f90 -O4 -o str_00.exe`
- Took out write_grid routine

"Standard" Input:

```
200 200
2000000 2000000
1.0e-9 2.25e-11 3.0e-6
75000
```
Dec Alpha Stommel Model
Serial Time

```
f90 stf_00.f90 -O4 -o str_00.exe
timex str_00.exe < st.in
run time = 194.52
real  194.6
user  194.4
sys   0.0
```
Our directives

• Put directive around outer loop
  • A1-A5 are private and defined outside the loop
  • diff is a reduction variable
  • We want i to be private also

• Our directive is:

```
$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
  firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
do j=j1,j2
  do i=i1,i2
    new_psi(i,j)= a1*psi(i+1,j) + a2*psi(i-1,j) + &
    a3*psi(i,j+1) + a4*psi(i,j-1) - &
    a5*for(i,j)
    diff=diff+abs(new_psi(i,j)-psi(i,j))
  enddo
enddo
```
Dec Alpha Stommel Model OpenMP

```
f90 -omp stf_00.f90 -O4 -o stf_00.omp
timex str_00.exe < st.in
run time = 363.95
real    364.2
user    1451.6
sys  0.5
```

Serial runtime:
```
run time = 194.52
```

Ouch! What happened?
Ouch! What Happened?

- Program spent much time in synchronization after do loops at line:
  \[
  \psi(i_1:i_2,j_1:j_2) = \text{new}_\psi(i_1:i_2,j_1:j_2)
  \]

- This should not happen but shows immaturity of the compilers

- It should work better on current compilers
Work around

- Rewrite the array syntax line and insert directives

```fortran
!     psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
!$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
do j=j1,j2
  do i=i1,i2
    psi(i,j)=new_psi(i,j)
  enddo
enddo
!$OMP END PARALLEL DO
```
Dec Alpha Stommel Model OpenMP

```bash
f90 -omp stf_00.f90 -O4 -o stf_00.omp
timex str_00.exe < st.in

run time =  50.10
real    50.3
user    200.3
sys     0.1
```

Serial runtime:
run time =  194.52

Factor of 4 Speedup!
Our final subroutine
header stuff

```
subroutine do_jacobi(psi,new_psi,diff,i1,i2,j1,j2)
    use numz
    use constants
    implicit none
    integer,intent(in) :: i1,i2,j1,j2
    real(b8),dimension(i1-1:i2+1,j1-1:j2+1):: psi
    real(b8),dimension(i1-1:i2+1,j1-1:j2+1):: new_psi
    real(b8) diff
    integer i,j
    real(b8) y
    diff=0.0_b8
```
$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
  firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
  do j=j1,j2
    do i=i1,i2
      new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
      a3*psi(i,j+1) + a4*psi(i,j-1) - &
      a5*for(i,j)
      diff=diff+abs(new_psi(i,j)-psi(i,j))
    enddo
  enddo
$OMP END PARALLEL DO

psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)

$OMP PARALLEL DO SCHEDULE (STATIC,50) private(i)
  do j=j1,j2
    do i=i1,i2
      psi(i,j)=new_psi(i,j)
    enddo
  enddo
$OMP END PARALLEL DO

end subroutine do_jacobi
implied do loop
outside of parallel section

!$OMP PARALLEL DO SCHEDULE (STATIC) private(i)
firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
do j=j1,j2
  do i=i1,i2
    !
    y=j*dy
    new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
    a3*psi(i,j+1) + a4*psi(i,j-1) - &
    a5*for(i,j)
    !
    a5*force(y)
    !
    diff=diff+abs(new_psi(i,j)-psi(i,j))
  enddo
enddo
!$OMP END PARALLEL DO
psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)

StomOmpf_00a.f
second do loop in its own parallel section

```fortran
!$OMP PARALLEL DO SCHEDULE (STATIC) private(i) firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
   do j=j1,j2
     do i=i1,i2
       y=j*dy
       new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
                    a3*psi(i,j+1) + a4*psi(i,j-1) - &
                    a5*for(i,j)
       !
       diff=diff+abs(new_psi(i,j)-psi(i,j))
     enddo
   enddo
!$OMP END PARALLEL DO
!$OMP PARALLEL DO SCHEDULE (STATIC) private(i)
   do j=j1,j2
     do i=i1,i2
       psi(i,j)=new_psi(i,j)
     enddo
   enddo
!$OMP END PARALLEL DO
```

StomOmpf_00b.f
implied do loop in first parallel section

```c
!$OMP PARALLEL
!$OMP DO SCHEDULE (STATIC) private(i) firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
   do j=j1,j2
      do i=i1,i2
         !
         y=j*dy
         new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
         a3*psi(i,j+1) + a4*psi(i,j-1) - &
         a5*for(i,j)
         !
         diff=diff+abs(new_psi(i,j)-psi(i,j))
      enddo
   enddo
!$OMP END DO
psi(i1:i2,j1:j2)=new_psi(i1:i2,j1:j2)
!$OMP END PARALLEL
```

StomOmpf_00c.f
!$OMP PARALLEL and two separate parallel do

!$OMP PARALLEL
!$OMP DO SCHEDULE (STATIC) private(i) firstprivate(a1,a2,a3,a4,a5) reduction(+:diff)
do j=j1,j2
  do i=i1,i2
    !
    y=j*dy
    new_psi(i,j)=a1*psi(i+1,j) + a2*psi(i-1,j) + &
    a3*psi(i,j+1) + a4*psi(i,j-1) - &
    a5*force(i,j)
    !
    diff=diff+abs(new_psi(i,j)-psi(i,j))
  enddo
enddo
!$OMP END DO
!$OMP DO SCHEDULE (STATIC) private(i)
do j=j1,j2
  do i=i1,i2
    psi(i,j)=new_psi(i,j)
  enddo
enddo
!$OMP END DO
!$OMP END PARALLEL

StomOmpf_00d.f
### Times using 8 threads

<table>
<thead>
<tr>
<th>Name</th>
<th>Concept</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>StomOmpf_00a.f</td>
<td>implied do loop outside of parallel section</td>
<td>22.81</td>
</tr>
<tr>
<td>StomOmpf_00b.f</td>
<td>second do loop in its own parallel section</td>
<td>3.54</td>
</tr>
<tr>
<td>StomOmpf_00c.f</td>
<td>implied do loop in first parallel section</td>
<td>57.79</td>
</tr>
<tr>
<td>StomOmpf_00d.f</td>
<td>!$OMP PARALLEL and two separate parallel do</td>
<td>2.99</td>
</tr>
</tbody>
</table>

ifort -O3 -openmp -free
serial time 18.79 seconds

Examples in “hybrid” directory
We can combine MPI and OpenMP

- A possible programming methodology for clustered SMP’s and NUMA architectures
Summary

• Simple changes enabled a 4x speedup
• Need to be careful because of immaturity of the compilers and you can introduce bugs
• We can combine OpenMP with MPI to take advantage of hybrid machines, both NUMA and clustered SMPs