How to Parallelize Your Code

Beth Richardson
NCSA Consultant and Trainer
bethr@ncsa.uiuc.edu

NCSA
University of Illinois at Urbana-Champaign

Outline for the Workshop

1. Port your code
2. Tune the code to improve scalar performance
3. Parallelize the code
4. Tune the code to improve parallel performance

We will cover step 3 in this lecture

NCSA
University of Illinois at Urbana-Champaign

Outline

Automatic Compiler Parallelism
Data Parallelism by Hand
Task Parallelism by Hand
Specify the number of threads

NCSA
University of Illinois at Urbana-Champaign

Automatic Compiler Parallelism

Use a compiler option and let the compiler do the work

- On the Power Challenge and Origin
  - f77 -pfa ... prog.f
- On the Exemplar-1200
  - fc -O3 … prog.f
- On the SPP-2000
  - f77 +O3 +Oparallel … prog.f

NCSA
University of Illinois at Urbana-Champaign

Automatic Compiler Parallelism (Cont.)

Advantage of compiler parallelism

- its easy

Disadvantages of compiler parallelism

- It only does loop level parallelism, not task parallelism
- It wants to parallelize every loop in your code. If you have hundreds of do loops this creates too much parallel overhead.

Data Parallelism by Hand

Identify the loops that use most of the cpu time (refer to the lecture on Hot Spot Analysis)

Hand insert into the code a compiler directive just before the loops you want to go parallel

Some code modifications may be needed to remove data dependencies

Use your knowledge of the code and data to assist the compiler

Data Parallelism by Hand
Power Challenge Array and Origin

- Insert into the code the c$doacross compiler directive just before the loop that you want to go parallel

```
c$doacross local(i), share(a,n)
     do i=1,n
     a(i)=float(i)
     end do
```

- The local variables are those private to each thread
- Compile with the "mp" compiler option

```
f77 -mp ... prog.f
```


Data Parallelism by Hand (Cont.)
Power Challenge Array and Origin

- The compiler generates conditional code that will run with any number of threads.
- Set the number of threads with the setenv command

```
setenv MP_SET_NUMTHREADS 4
a.out > results
```

- and make sure you still get the right answers
- If you want to rerun with a different number of threads, you do not need to recompile.

Data Parallelism by Hand
Exemplar-1200

Insert into the code the c$dir loop_parallel compiler directive just before the loop you want to go parallel.

Specify variables that are private to each thread with the c$dir loop_private directive.

```
c$dir loop_parallel
c$dir loop_private (list of private vars)
     do i=1,1000
     … body of loop ...
     end do
```


Data Parallelism by Hand(Cont.)
Exemplar-1200

Compile with the O3 compiler option

```
fc -O3 …  prog.f
```

The compiler generates conditional code that can be run with any number of threads.

Specify the number of threads with the mpa command

```
mpa -max 4 a.out > results
```

and make sure you still get the right answers

If you want to rerun with a different number of threads, you do not need to recompile

Data Parallelism by Hand
SPP-2000

- Insert into the code the c$dir loop_parallel compiler directive just before the loop you want to go parallel.
- Specify variables that are private to each thread with the c$dir loop_private directive.

```
c$dir loop_parallel
c$dir loop_private (list of private vars)
    do I=1,1000
    … body of loop …
    end do
```


Data Parallelism by Hand
SPP-2000

- Compile as follows

```
f77 +O3 +Oparallel +Onoautopar … prog.f
```

- The compiler generates conditional code that can be run with any number of threads.
- Specify the number of threads with the setenv command

```
setenv MP_NUMBER_OF_THREADS 4
a.out > results
```

- and make sure you still get the right answers
- If you want to rerun with a different number of threads, you do not need to recompile.

Data Parallelism Reminders

Do not mix automatic compiler parallelism with hand parallelism. When this is done, the results are unpredictable.

The compiler directive begins in column 1.

A directive effects the loop that it immediately precedes, it does not effect loops that are nested within that loop.

The Exemplar-1200 and SPP-2000 compiler directives cannot be continued

An SGI compiler directive continuation line begins with C$&

University of Illinois at Urbana-Champaign

Task Parallelism
Power Challenge Array and Origin

There are no special compiler directives for task parallelism on the Power Challenge Array and Origin computers. You can accomplish task parallelism as follows:

```
c$doacross
    do i=1,4
    if (i=1) call sub1 (...)
    if (i=2) call sub2 (...)
    if (i=3) call sub3 (...)
    if (i=4) call sub4 (...)
    enddo
```

University of Illinois at Urbana-Champaign

Task Parallelism (Cont.)
Power Challenge Array and Origin

Use the "mp" compiler option

```
f77 -mp …  prog.f
```

Use the “setenv” command to specify the number of threads

```
setenv MP_SET_NUMTHREADS 4
a.out > results
```


Task Parallelism
Exemplar-1200 and SPP-2000

There are specific directives for task parallelism on the Exemplar-1200 and the SPP-2000.

```
     beginning of program
c$dir begin_tasks
     lots of code
c$dir next_task
     lots of code
c$dir next_task
     lots of code
c$dir end_tasks
     rest of program
```

Task Parallelism (Cont.)

Exemplar-1200

- **fc -O3 -noautopar … prog.f**
- **mpa -max 4 a.out > results**

SPP-2000

- **f77 +O3 +Oparallel +Onoautopar … prog.f**
- **setenv MP_NUMBER_OF_THREADS 4**
- **a.out > results**

NCSA
University of Illinois at Urbana-Champaign