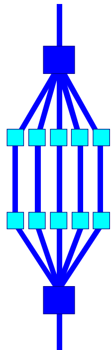
Sun microsystems
take it to the nth



How to use OpenMP on Sun

Ruud van der Pas

High-Performance and Technical Computing
Application Performance Specialist

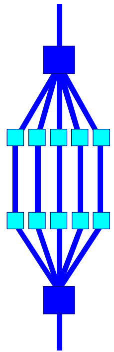
Enterprise System Products
Sun Microsystems

EWOMP2003
RWTH Aachen, Germany
September 22-26, 2003

 Copyright©2003 Sun Microsystems

Serial performance

Sun microsystems
take it to the nth



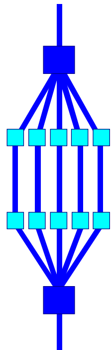
In general, one obtains very good performance out of the Sun compilers by <u>just using 3 options</u> on the compile and link line:

For the UltraSPARC-III Cu processor:

```
-fast -xarch=v8plusb    (32-bit addressing)
-fast -xarch=v9b        (64-bit addressing)
```


Copyright©2003 Sun Microsystems

Serial performance and OpenMP



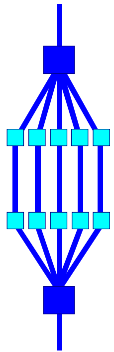
Activates OpenMP

```
-fast -xarch=v8plusb -xopenmp -xloopinfo   (32-bit addr.)

-fast -xarch=v9b     -xopenmp -xloopinfo   (64-bit addr.)
```

Loop Parallelization Messages

Copyright©2003 Sun Microsystems

Parallelization Options*



User Code	Perf. Library	Compile	Link**
Serial	Parallel		-xautopar
Auto-Parallel	Parallel	-xautopar	-xautopar
OpenMP	Parallel	-xopenmp	-xautopar
Auto+OpenMP	Parallel	-xautopar *and* -xopenmp	-xautopar
Parallel	Serial***	-xautopar	-xautopar

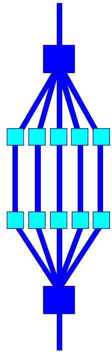
*) *It is assumed that you compile and link with -fast as well (for good serial performance)*

**) *Linking with -xautopar or -xparallel is equivalent*

***) *By default you'll get the parallel version if you're in a serial region; use Sun Perflib's routine "USE_THREADS" to override the number of threads*

Copyright©2003 Sun Microsystems

Example - Compile and Link



```
DO J = 1, N
   CALL DGEMM(.....)
END DO
CALL DGEMM
```

- -fast → **DGEMM serial**
- -fast -xparallel → **DGEMM parallel**

```
!$OMP PARALLEL ...
DO I = 1, N
   CALL DGEMM(.....)
END DO
(call use_threads(1))
CALL DGEMM
```

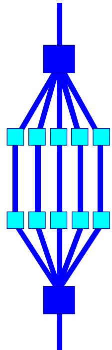
forces DGEMM to run on 1 processor

- -fast → **DGEMM serial**
- -fast -xopenmp → **DGEMM serial within loop, parallel outside of loop**

Copyright©2003 Sun Microsystems

OpenMP Compiler Options

Sun microsystems
take it to the nth



`-xopenmp[=parallel]`

Add this option for explicit OpenMP programs, requires -xO3 (at least)

`-xopenmp=stubs`

Use a runtime stub library only; useful for testing purposes

`-xopenmp=noopt -g`

To debug your OpenMP application

`-stackvar (Fortran only, included with -xopenmp)`

Allocate local data on stack

`-xcheck=stkovf`

Assists detecting stack overflow problems

`-xloopinfo`

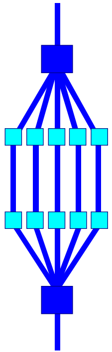
Show loop parallelization messages on the screen

`-XlistMP (Fortran only)`

Reports warnings about possible errors in OpenMP parallelization

Copyright©2003 Sun Microsystems

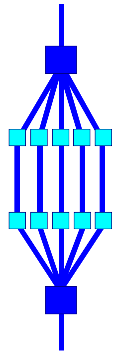
OpenMP Runtime Library



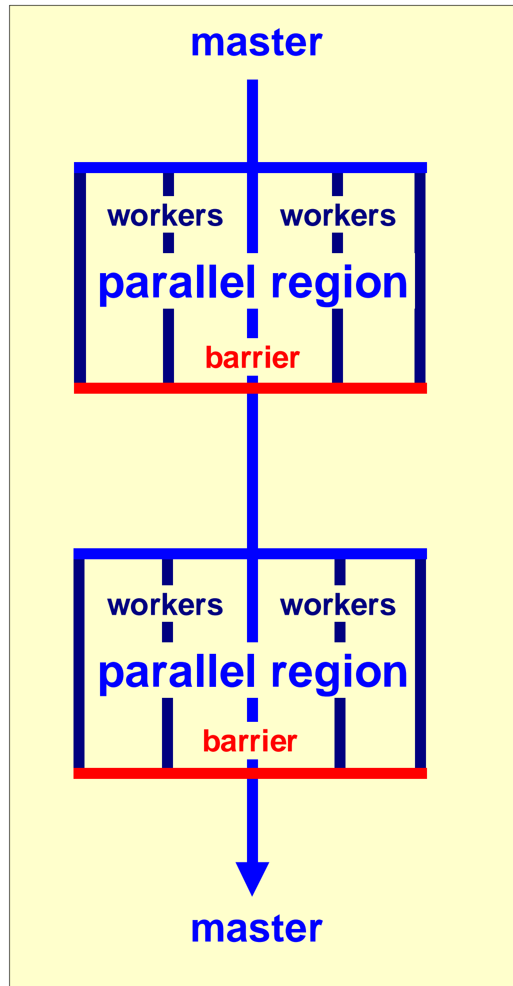
- ***OpenMP Fortran runtime routines are external functions***
- ***Their names start with OMP_ but usually have an integer or logical return type***
- ***Therefore these functions must be declared explicitly***
- ***On Sun systems the following features are available:***
  - ***USE omp_lib***
  - ***INCLUDE 'omp_lib.h'***
  - ***#include “omp_lib.h” (preprocessor directive)***
- ***C programs: #include <omp.h>***
- ***Use the -XlistMP (and -Xlist) option on the Fortran compiler to check your code***

Copyright©2003 Sun Microsystems

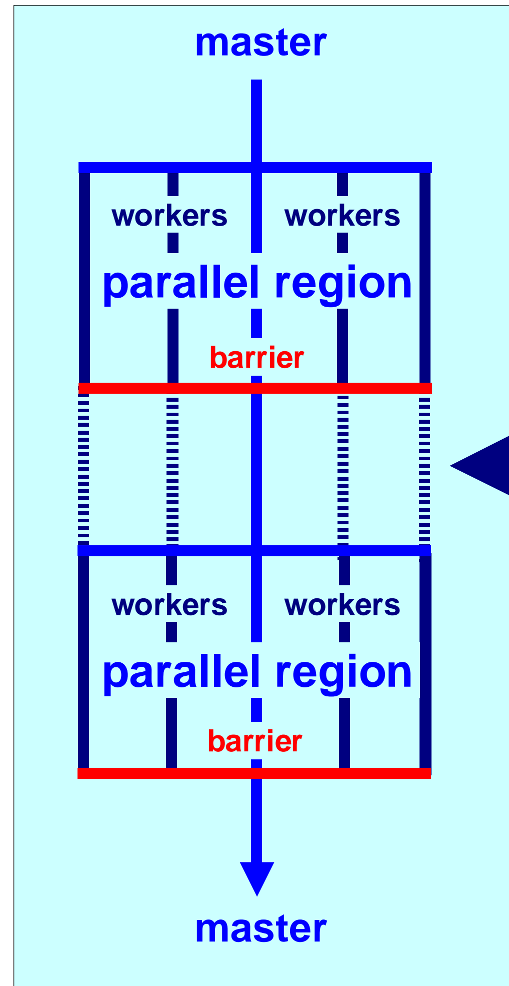
The Fork-Join Model



Parallel Model



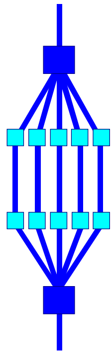
SUN's Implementation



Idle threads spin in the CPU by default

Copyright©2003 Sun Microsystems

The behaviour of idle threads



Environment variable to control the behaviour:

```
SUNW_MP_THR_IDLE
[ spin | sleep | sleep (ns) | sleep (nms) ]
```

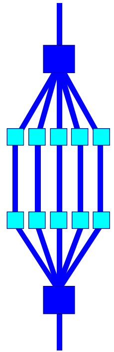
- *Default is to "spin" i.e. block the CPU*
- *Sleep: thread is put to sleep; awakened when new work arrives*
- *Sleep ('time'): spin for 'time' s (or ms), then go into sleep mode*

When to set SUNW_MP_THR_IDLE to sleep ?

- *To use the CPU time effectively (throughput versus single job performance)*
- *If the number of threads exceeds the number of processors*
- *Parallel regions constitute a small portion of the total execution time of the program (i.e. long serial portions in the code)*

Copyright©2003 Sun Microsystems

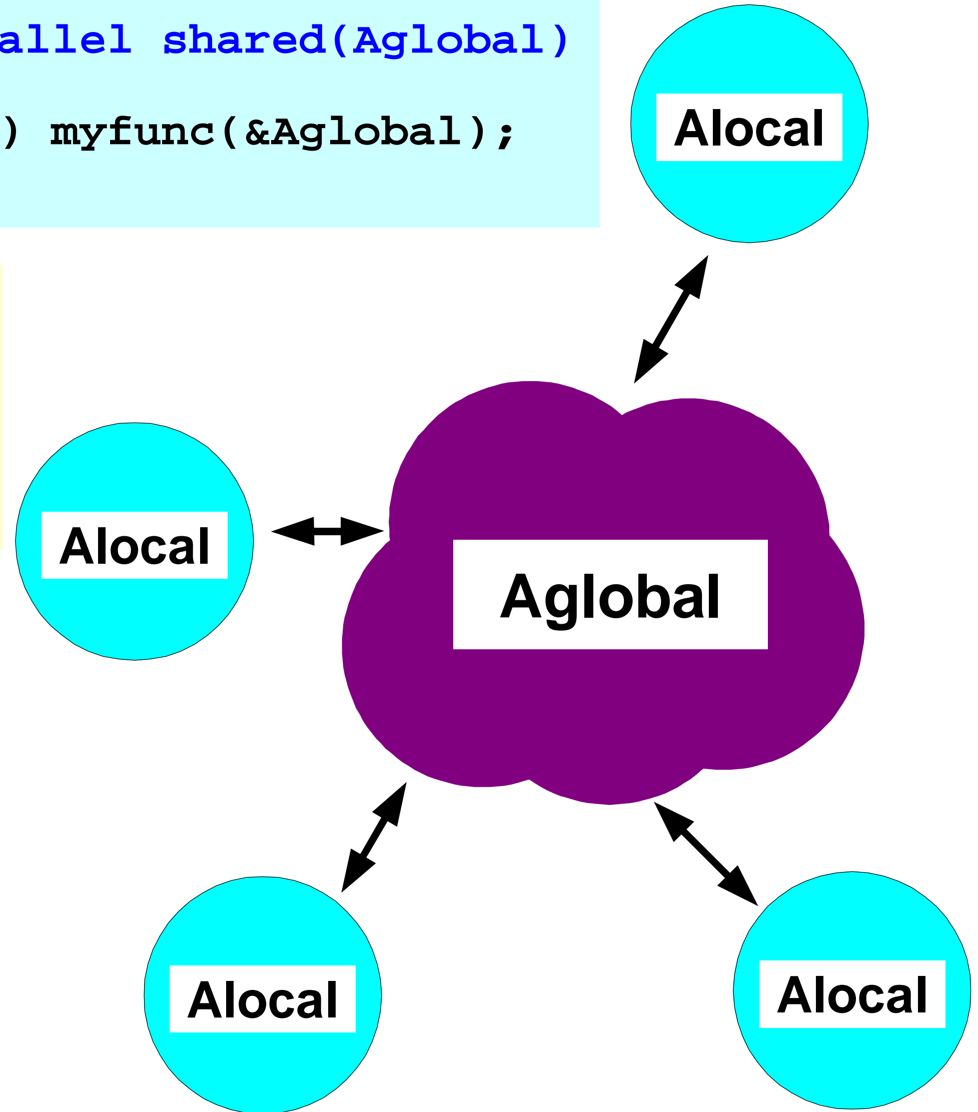
About the stack



```
#omp parallel shared(Aglobal)
{
   (void) myfunc(&Aglobal);
}
```

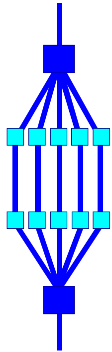
```
void myfunc(float *Aglobal)
{
    int Alocal;

    .......
}
```

Access to Alocal is managed through the so-called stack



Copyright©2003 Sun Microsystems

The STACKSIZE env. variable

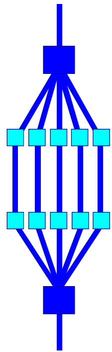


`STACKSIZE n` — *Set thread stacksize in n KByte*

- *Each thread has it's own private stackspace*
- *If a thread runs out of this stackspace, your program will crash with a segmentation violation*
- *Use the Unix "limit" command to increase the stacksize for the MAIN ("parent") program*
  - ✔ *Default is 8 MByte*
- *Use STACKSIZE to increase the thread stacksize*
- *Default value for STACKSIZE:*
  - ✔ *4 MByte for 32-bit addressing*
  - ✔ *8 MByte for 64-bit addressing*

Copyright©2003 Sun Microsystems

Example STACKSIZE



```
      program main()
      integer, parameter:: n=2000000
      integer, parameter:: p=2
      real(kind=8)       :: a(n), check(p)

!$omp parallel default(none) &
!$omp private(i) shared (check)
!$omp do
      do i = 1, p
         call suba(i,check(i))
      end do
!$omp end do
!$omp end parallel
      print *,'check=',check

      stop
      end
```

Main requires about 16 MByte stack space to run

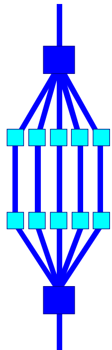
Subroutine requires about ~8 MByte stack space to run

```
subroutine suba(i,check)
integer                :: i
real(kind=8)           :: check
integer, parameter:: n=1000000
real(kind=8)           :: mystack(n)

do i = 1, n
   mystack(i) = i
end do
check = mystack(n)

return
end
```


Copyright©2003 Sun Microsystems

Runtime behaviour

microsystems
take it to the nth



```
% f95 -fast -g -xopenmp -xloopinfo main.f95
"main.f95", line 9: PARALLELIZED, user pragma used
```

```
% setenv OMP_NUM_THREADS 1
% limit stack 10k
% ./a.out
Segmentation Fault

% limit stack 16m

% ./a.out
 check= 500000.0 500000.0
```

Not enough stack for master thread

Now runs fine on 1 processor

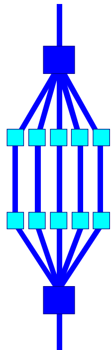
```
% setenv OMP_NUM_THREADS 2
% ./a.out
Segmentation Fault

% setenv STACKSIZE 8000
% setenv OMP_NUM_THREADS 1
% ./a.out
 check= 500000.0 500000.0
% setenv OMP_NUM_THREADS 2
% ./a.out
 check= 500000.0 500000.0
```

But crashes on 2 ....

Increase thread stacksize and all is well again

Copyright©2003 Sun Microsystems

Default Behaviour

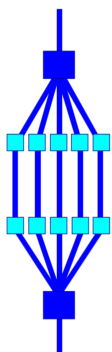


```
hpc% f95 -g -fast -xopenmp main.f95
hpc% setenv OMP_NUM_THREADS 2; unsetenv STACKSIZE
hpc% dbx a.out
   <... lines deleted ...>
(dbx) run
Running: a.out (process id 9585)
t@5 (l@6) signal SEGV (no mapping at the fault address) in suba (optimized)
     at line 26 in file "main.f95"
   26              mystack(i) = i
    dbx: read of 4 bytes at address fd0605f0 failed -- Error 0
(dbx) where
current thread: t@5
=>[1] suba(i = ???, check = ???) (optimized), at 0x13ed8 (line ~26) in "main.f95"
  [2] _$d1A8.MAIN_() (optimized), at 0x13f7c (line ~10) in "main.f95"
  [3] __mt_runLoop_int_(0xfd801a18, 0x6ff28, 0x1d424, 0x2, 0x13f60, 0xfd801bd4),
      at 0x1d684
  [4] __mt_run_my_job_(0x6ff28, 0xfd801a18, 0x40, 0x0, 0x0, 0x0), at 0x14b24
  [5] __mt_WorkSharing_(0xfd801b90, 0x13f60, 0xfd801bd4, 0xffbef960, 0x0, 0x0), at
      0x148c0
  [6] _$p1B6.MAIN_() (optimized), at 0x14008 (line ~8) in "main.f95"
  [7] __mt_run_my_job_(0x6ff28, 0xffbef7d8, 0x0, 0x0, 0xffbef960, 0x0), at 0x14c70
  [8] __mt_SlaveFunction_(0x6ff28, 0xfeeef690, 0xffbef7d8, 0x68800, 0x64800,
      0x68800), at 0x17e70
(dbx) quit
```

For example only !

Copyright©2003 Sun Microsystems

The -xcheck=stkovf option

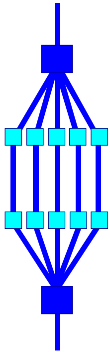


```
hpc% f95 -fast -g -xopenmp -xcheck=stkovf main.f95
hpc% setenv OMP_NUM_THREADS 2; unsetenv STACKSIZE
hpc% dbx a.out
    <... lines deleted ...>
(dbx) run
Running: a.out (process id 9590)
t@5 (l@6) signal SEGV (access to address exceeded protections) in
__stack_grow_probing_impl at 0x13ebc
0x00013ebc: __stack_grow_probing_impl+0x001c:   ldsb    [%o3], %g0
Current function is suba (optimized)
   19          subroutine suba(i,check)
(dbx) where
current thread: t@5
  [1] __stack_grow_probing_impl(0xfd0605d0, 0x13ea0, 0x64c00, ....)
=>[2] suba(i = ???, check = ???) (optimized), at 0x14158 (line ~19) in "main.f95"
  [3] _$d1A8.MAIN_() (optimized), at 0x14220 (line ~10) in "main.f95"
  [4] __mt_runLoop_int_(0xfd801a18, 0x702a0, 0x1d6cc, 0x2, 0x14200, ....)
  [5] __mt_run_my_job_(0x702a0, 0xfd801a18, 0x40, 0x0, 0x0, 0x0), at 0x14dc8
  [6] __mt_WorkSharing_(0xfd801b90, 0x14200, 0xfd801bd4, 0xffbef960, ...)
  [7] _$p1B6.MAIN_() (optimized), at 0x142ac (line ~8) in "main.f95"
  [8] __mt_run_my_job_(0x702a0, 0xffbef7d8, 0x0, 0x0, 0xffbef960, 0x0
  [9] __mt_SlaveFunction_(0x702a0, 0xfeeef690, 0xffbef7d8, 0x68c00, ...)
(dbx) quit
```

For example only !

Copyright©2003 Sun Microsystems

Tip: use pstack on core file



```
hpc% pstack core

core 'core' of 9579:     ./a.out
-----------------  lwp# 5 / thread# 4  --------------------
 00013ebc __stack_grow_probing_impl (fd0018ac, ffbefa40, 0, 0, 0, 0) + 1c
 00014220 _$d1A8.MAIN_ (fd001bd4, 3, 2, ffbefa30, 10, fd001bdc) + 20
 0001d92c __mt_runLoop_int_ (fd001a18, 702a0, 1d6cc, 2, 14200, fd001bd4) + 228
 00014dc8 __mt_run_my_job_ (702a0, fd001a18, 40, 0, 0, 0) + 84
 00014b64 __mt_WorkSharing_ (fd001b90, 14200, fd001bd4, ffbefa10, 0, 0) + 260
 000142ac _$p1B6.MAIN_ (ffbefa10, 14000, 2, 1, ffbefa34, 0) + 6c
 00014f14 __mt_run_my_job_ (702a0, ffbef888, 0, 0, ffbefa10, 0) + 1d0
 00018118 __mt_SlaveFunction_ (702a0, fe6ef690, ffbef888, 68c00, 64c00, 68c00) + fc
 fe6dbad0 _thread_start (702a0, 0, 0, 0, 0, 0) + 40
-----------------  lwp# 1 / thread# 1  --------------------

 00013ebc __stack_grow_probing_impl (ffbef47c, ffbefa38, 0, 0, 0, 0) + 1c
 00014220 _$d1A8.MAIN_ (ffbef7a4, 2, 1, ffbefa30, 8, ffbef7ac) + 20
 0001d92c __mt_runLoop_int_ (ffbef5e8, 6fc38, 1d6cc, 1, 14200, ffbef7a4) + 228
 00014dc8 __mt_run_my_job_ (6fc38, ffbef5e8, e0, a0, 0, a0) + 84
 00014b64 __mt_WorkSharing_ (ffbef760, 14200, ffbef7a4, ffbefa10, 0, 0) + 260
 000142ac _$p1B6.MAIN_ (ffbefa10, 14000, 2, 1, ffbefa34, 0) + 6c
 00014f14 __mt_run_my_job_ (6fc38, ffbef888, 0, a0, ffbefa10, 0) + 1d0
 00014748 __mt_MasterFunction_ (0, 0, 68c00, 0, 0, 0) + 38c
 000140ac MAIN_    (64c00, fe0b28e0, 14000, 102, ffbefa34, ffbefa30) + 4c
 00014034 main     (1, ffbefb1c, ffbefb24, 64c00, 0, 0) + 34
 00013e88 _start   (0, 0, 0, 0, 0, 0) + 108
-----------------  lwp# 2 / thread# 2  --------------------

                <... rest of output deleted ...>
```


Copyright©2003 Sun Microsystems