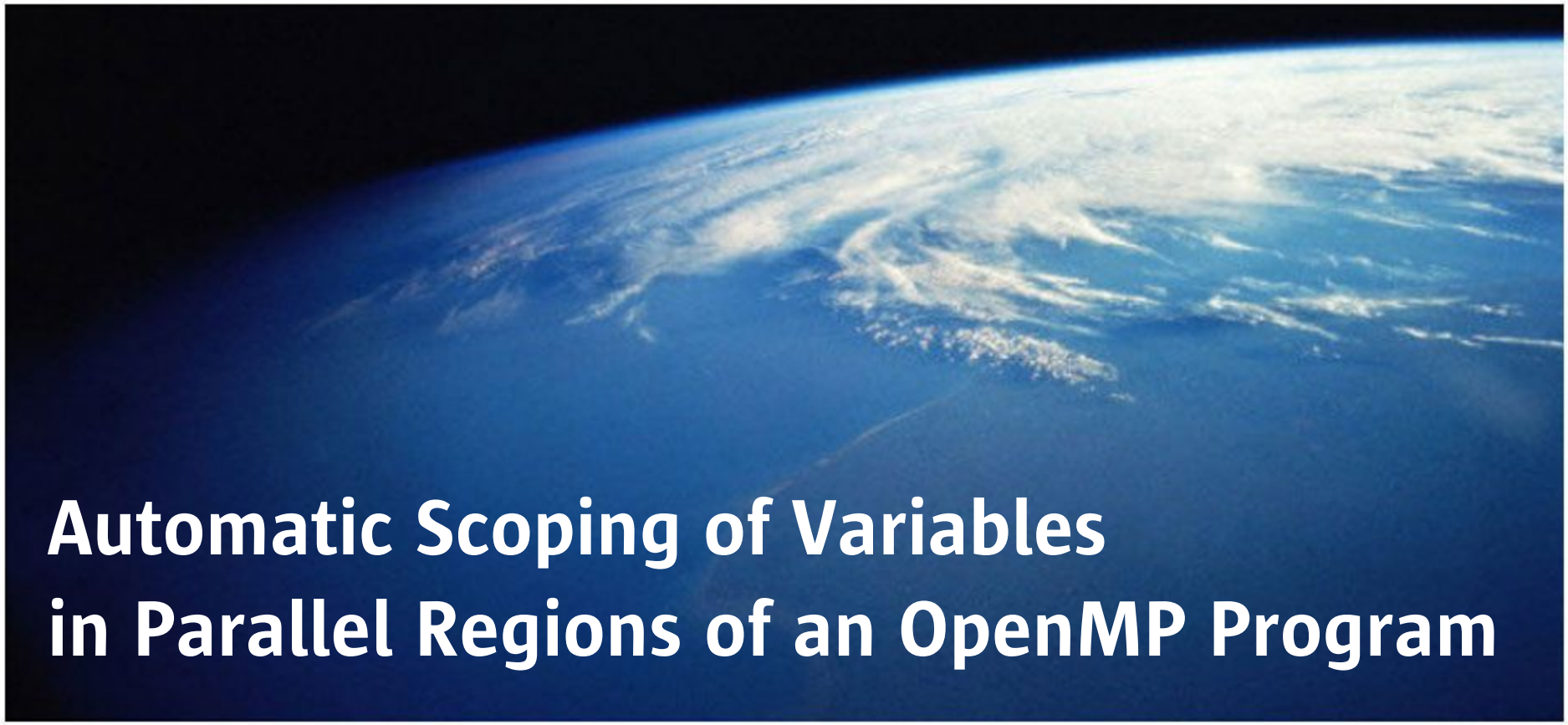


Automatic Scoping of Variables in Parallel Regions of an OpenMP Program

Yuan Lin[1]
Christian Terboven[2]
Dieter an Mey[2]
Nawal Copty[1]

[1]*Sun Microsystems, USA*
[2]*RWTH Aachen University, Germany*

RWTH AACHEN UNIVERSITY Center for Computing and Communication

Sun microsystems
We make the net work.

The Problem

- The process of manually specifying scopes of variables is tedious and error-prone.
- Any thing language/compiler/tool can do to improve productivity?

A solution: autoscoping

- User specifies which variables to be autoscoped
- Compiler determines the appropriate scopes for the variables

An example

```
C$OMP PARALLEL PRIVATE(T,I) SHARED(W,A,N)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

OMP Autoscoping: a brief history

- Proposed by Dieter an Mey to OpenMP ARB
  - "New Ideas for OpenMP", www.compunity.org, 2001

- First implementation:
  Sun Studio 9 Fortran 95 Compiler

Early Access Version Publicly Available Now.

http://developers.sun.com/prodtech/cc/ea/ss9/index.html

Agenda

- Autoscoping: design and implementation
- An experiment with autoscoping
- Conclusion

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
  - Synchronization
  - Communication

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
  - Communication

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication
    - SHARED, FIRSTPRIVATE, LASTPRIVATE, ...

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication
    - SHARED, FIRSTPRIVATE, LASTPRIVATE, ...

- Parallel execution Synchronization Communication

What is Autoscoping

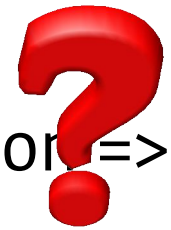
- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication
    - SHARED, FIRSTPRIVATE, LASTPRIVATE, ...

- Autoscoping:
  - Parallel execution + Synchronization => Communication

What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication
    - SHARED, FIRSTPRIVATE, LASTPRIVATE, ...

- Autoscoping:
  - Parallel execution + Synchronization => Communication



What is Autoscoping

- Three aspects of writing a parallel program
  - Parallel execution
    - PARALLEL, DO, SECTIONS, SINGLE, ...
  - Synchronization
    - BARRIER, CRITICAL, ...
  - Communication
    - SHARED, FIRSTPRIVATE, LASTPRIVATE, ...

- Autoscoping:
  - Parallel execution + Synchronization + Autoscoping Rules => Communication

Autoscoping Rules
Design philosophy

- Simple enough

  So that most users can understand them.

- Complete enough

  So that they can cover most useful cases.

The Rules in a Nutshell

- No data race -> SHARED
- Written before read in each thread -> PRIVATE
- Written before read in each thread, read before written after the parallel region -> LASTPRIVATE
- Reduction form -> REDUCTION
- Check in the above order
- Follow implicit scoping rules and restrictions in the OMP Specification

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

I: PRIVATE

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

T: PRIVATE

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

N: SHARED

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

W: SHARED

An example

```
C$OMP PARALLEL DEFAULT(AUTO)
      T = N*N
C$OMP SINGLE
      W = T * OMP_GET_NUM_THREAD()
C$OMP END SINGLE
C$OMP DO
      DO I=1, N
         A(I) = T + W + I
      END DO
C$OMP END DO
C$OMP END PARALLEL
```

A: SHARED

Autoscoping may not be able to scope all variables

- The use of the variable does not match any of the autoscoping rules.

- The compiler is not able to perform an accurate analysis.
  - Compile-time unknown data
  - Memory alias
  - Interprocedural issues
  - User-implemented synchronization

Autoscoping may not be able to scope all variables

- The use[illegible] match any of th[illegible]
- The com[illegible]rm an accurate
  - Compile-time unknown data
  - Memory alias
  - Interprocedural issues
  - User-implemented synchronization

Important :

Feedback to the user

OMP Autoscoping in Sun Studio 9 Fortran 95 Compiler

- Publicly available now
- Can handle general OMP parallel regions; Not just "!$OMP PARALLEL DO"
- Serialize a parallel region and issue a warning message if autoscoping cannot scope a variable
- Detailed autoscoping results reported via *compiler commentary*; cut-and-paste possible
- Some limitations

An Experiment with Autoscoping

Purpose of our experiments

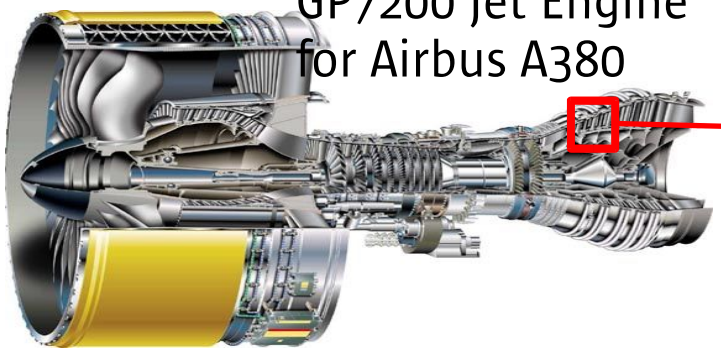
- Verify if expectations are fulfilled by Sun's current implementation:
  - Save work + prevent bugs:
    - fewer source code editing required?
    - fewer Assure test-runs required?
    -> productivity improved?
  - Impact on performance?

Overview of PANTA (1)

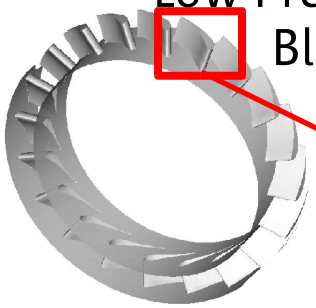
- Developed at the IST at RWTH Aachen Univ. for computation of turbomachinery flow
  -> solution of partial differential equations
- Several layers of parallelism in production, here: loop-level OpenMP (lin. eqn. system)
- Real-world example, also used in Dieter's proposal

Overview of PANTA (2)

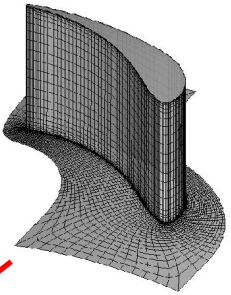
GP7200 Jet Engine
for Airbus A380



Low Pressure Turbine
Blade Row
MPI / OMP

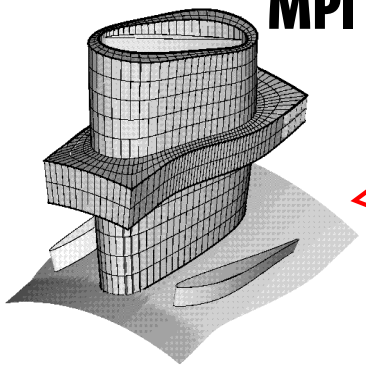


Low Pressure Turbine
Blade Channel
MPI / OMP

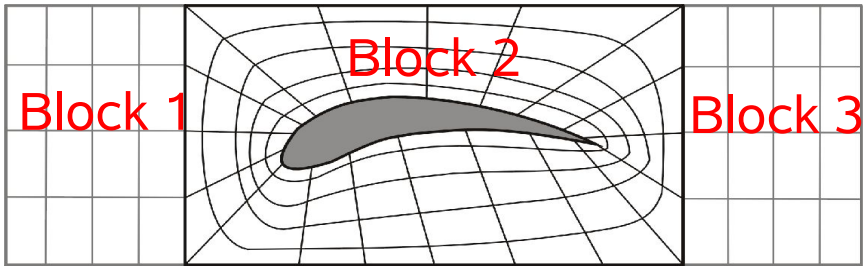


Linear eqn. solver
OMP

Alternating
Direction
Inversion Zones
MPI / OMP



Blocks MPI / OMP



Overview of PANTA (3)

- Source code statistics:
  - About 50,000 lines of Fortran 90 code
  - About 7000 lines code in the compute kernel in 11 subroutines
  - About 200 OpenMP statements in 370 lines
  - Default scope is SHARED

- About two weeks for the development of original OpenMP version

Experiment setup

- Sun Studio 9 Fortran 95 compiler
  (EA1 and EA2, on Sparc Solaris)

- Run on an idle Sun Fire 6800 server
  (using one to four threads)

- Different versions:
  - (original) OpenMP version, OMP
  - Autoparallel version, APA
  - Autoscope version, ASC
  - Edited autoscope version, ASM

Experiment setup

Compiler options OMP:

```
f90 -g -fast -openmp -dalign
-xarch=v9b -xchip=ultra3cu -xcache=...
```

Compiler options APA:

```
f90 -g -fast -xautopar -xreduction -dalign
-xarch=v9b -xchip=ultra3cu -xcache=...
```

Compiler options ASC:

```
f90 -g -vpara -fast -openmp -dalign
-xarch=v9b -xchip=ultra3cu -xcache=...
```

Compiler options ASM:

```
f90 -g -vpara -fast -openmp -dalign
-xarch=v9b -xchip=ultra3cu -xcache=...
```

- Edited autoscope version, ASM

Experiment results (1)

```
!$omp parallel private(local_QSIK,local_QSIE)
!$omp do &
!$omp private (lijk,xiaxc,xiayc,xiazc,xiaxeb,xiayeb,xiazeb,xiaxwb,xiaywb,xiazwb, &
!$omp & xibxnb,xibynb,xibznb,xibxsb,xibysb,xibzsb,xicxub,xicyub,xiczub,xicxdb, &
!$omp & xicydb,xiczdb,xibxc,xicxc,qjc,lijk2,lijk4,ueb,veb,web,psteb,teb,tkeb, &
!$omp & eeb,uwb,vwb,wwb,pstwb,twb,tkwb,ewb,unbp,unbm,vnbp,vnbm,wnbp,wnbm,pstnw, &
!$omp & pstne,tnbp,tnbm,tknbp,tknbm,enbp,enbm,usbp,usbm,vsbp,vsbm,wsbp,wsbm, &
!$omp & pstse,pstsw,tsbp,tsbm,tksbp,tksbm,esbp,esbm,uubp,uubm,vubp,vubm,wubp, &
!$omp & wubm,pstuw,pstue,tubp,tubm,tkubp,tkubm,eubp,eubm,udbp,udbm,vdbp,vdbm, &
!$omp & wdbp,wdbm,pstde,pstdw,tdbp,tdbm,tkdbp,tkdbm,edbp,edbm,uc,vc,wc,tc,tkc, &
!$omp & ec,amuelc,amuetc,amuegc,qmqc,amuekc,amueec,auebx,auwbx,aucx,avebx,avwbx, &
!$omp & avcx,awebx,awwbx,awcx,bunbx,busbx,buubx,budbx,bvnbx,bvsbx,bvubx,bvdbx, &
!$omp & bwnbx,bwsbx,bwubx,bwdbx,gunb,gusb,guub,gudb,gvnb,gvsb,gvub,gvdb,gwnb, &
!$omp & gwsb,gwub,gwdb,runbpx,runbmx,rvnbpx,rvnbmx,rwnbpx,rwnbmx,rusbpx,rusbmx, &
!$omp & rvsbpx,rvsbmx,rwsbpx,rwsbmx,ruubpx,ruubmx,rvubpx,rvubmx,rwubpx,rwubmx, &
!$omp & rudbpx,rudbmx,rvdbpx,rvdbmx,rwdbpx,rwdbmx,unb,usb,uub,udb,vnb,vsb,vub, &
!$omp & vdb,wnb,wsb,wub,wdb,diffru,aueby,auwby,aucy,aveby,avwby,avcy,aweby, &
!$omp & awwby,awcy,bunby,busby,buuby,budby,bvnby,bvsby,bvuby,bvdby,bwnby,bwsby, &
!$omp & bwuby,bwdby,runbpy,runbmy,rvnbpy,rvnbmy,rwnbpy,rwnbmy,rusbpy,rusbmy, &
!$omp & rvsbpy,rvsbmy,rwsbpy,rwsbmy,ruubpy,ruubmy,rvubpy,rvubmy,rwubpy,rwubmy, &
!$omp & rudbpy,rudbmy,rvdbpy,rvdbmy,rwdbpy,rwdbmy,diffrv,auebz,auwbz,aucz,avebz, &
!$omp & avwbz,avcz,awebz,awwbz,awcz,bunbz,busbz,buubz,budbz,bvnbz,bvsbz,bvubz, &
!$omp & bvdbz,bwnbz,bwsbz,bwubz,bwdbz,runbpz,runbmz,rvnbpz,rvnbmz,rwnbpz,rwnbmz, &
```

```
!$omp & rusbpz,rusbmz,rvsbpz,rvsbmz,rwsbpz,rwsbmz,ruubpz,ruubmz,rvubpz,rvubmz, &
!$omp & rwubpz,rwubmz,rudbpz,rudbmz,rvdbpz,rvdbmz,rwdbpz,rwdbmz,diffrw,auebe, &
!$omp & auwbe,auce,avebe,avwbe,avce,awebe,awwbe,awce,atebe,atwbe,atce,bunbe, &
!$omp & busbe,buube,budbe,bvnbe,bvsbe,bvube,bvdbe,bwnbe,bwsbe,bwube,bwdbe,btnbe, &
!$omp & btsbe,btube,btdbe,gtnb,gtsb,gtub,gtdb,runbpe,runbme,rvnbpe,rvnbme,rwnbpe, &
!$omp & rwnbme,rtnbpe,rtnbme,rusbpe,rusbme,rvsbpe,rvsbme,rwsbpe,rwsbme,rtsbpe, &
!$omp & rtsbme,ruubpe,ruubme,rvubpe,rvubme,rwubpe,rwubme,rtubpe,rtubme,rudbpe, &
!$omp & rudbme,rvdbpe,rvdbme,rwdbpe,rwdbme,rtdbpe,rtdbme,tnb,tsb,tub,tdb,prode, &
!$omp & diffe,akebk,akwbk,akck,bknbk,bksbk,bkubk,bkdbk,gknb,gksb,gkub,gkdb, &
!$omp & rknbpk,rknbmk,rksbpk,rksbmk,rkubpk,rkubmk,rkdbpk,rkdbmk,tknb,tksb,tkub, &
!$omp & tkdb,diffrk,aeebe,aewbe,aece,benbe,besbe,beube,bedbe,genb,gesb,geub, &
!$omp & gedb,renbpe,renbme,resbpe,resbme,reubpe,reubme,redbpe,redbme,enb,esb,eub, &
!$omp & edb,diffre,aux,avx,awx,auy,avy,awy,auz,avz,awz,aue,ave,awe,ate,akk,aee, &
!$omp & qdens,caqden,asonsq,velosq,omegaz,omegay,dudq1,dudq2,dvdq1,dvdq3,dwdq1, &
!$omp & dwdq4,dtdq1,dtdq2,dtdq3,dtdq4,dtdq5,dtdq6,dkdq1,dkdq6,dedq1,dedq7, &
!$omp & hfmuetsst,uyvx,uzwx,vzwy,vxuy,wxuz,wyvz,duxdq1,duxdq2,duydq1,duydq2, &
!$omp & duzdq1,duzdq2,dvxdq1,dvxdq3,dvydq1,dvydq3,dvzdq1,dvzdq3,dwxdq1,dwxdq4, &
!$omp & dwydq1,dwydq4,dwzdq1,dwzdq4,amt,omsqr,dmtdq1,dmtdq2,dmtdq3,dmtdq4,dmtdq6, &
!$omp & dmtdq7)
!$omp lastprivate(INE,IEE,ISE,ISW,IWW,INW,IUE,IUW,IDE,IDW,INU,IND,ISU,ISD)
    do i = is,ie
        [...]   1949 lines omitted
    end do
!$omp end do
!$omp end parallel
```

… with Autoscoping

```
!$omp parallel DEFAULT(__AUTO)
!$omp do
    do i = is,ie
        [...]   1949 lines omitted
    end do
!$omp end do
!$omp end parallel
```

… with Autoscoping

```
!$omp parallel DEFAULT(__AUTO)
!$omp do
    do i = is,ie
        [...]   1949 lines omitted
    end do
!$omp end do
!$omp end parallel
```

- No errors
  - at most one Assure test-run
  - speedup the development cycle
- Avoided a lot of “boring” work

Experiment results (2)

- Autoscoping is unable to scope some variables
  - No matching autoscoping rule
  - Limitation of the data-race detection method (false positive)
  - Compile-time unknown data
- BUT: Autoscoping version is about 1.7 times faster than the autoparallel version

Experiment results (3)

```
integer :: l3
l3(n1,n2,n3) = n1 + n2*D23 + n3*D33 + D43
```

```
!$omp parallel DEFAULT(__AUTO)
    do idir = 1,3
        do k = ks,ke+1
            do j = js,je+1
                !$omp do
                do i = is,ie+1
                    ADDFLG(l3(i,j,k),idir) = 1.
                end do
                !$omp end do nowait
            end do
        end do
    end do
!$omp end parallel
```

D23, D33 and D43 unknown at compile-time, but invariant in the parallel region, in l3 i=n1 is the loop-variable of the parallel loop.

Experiment results (3)

```
integer :: l3
l3(n1,n2,n3) = n1 + n2*D23 + n3*D33 + D43
```

```
!$omp parallel DEFAULT(__AUTO) SHARED(addflg)
    do idir = 1,3
        do k = ks,ke+1
            do j = js,je+1
                !$omp do
                do i = is,ie+1
                    ADDFLG(l3(i,j,k),idir) = 1.
                end do
                !$omp end do nowait
            end do
        end do
    end do
!$omp end parallel
```

D23, D33 and D43 unknown at compile-time, but invariant in the parallel region, in l3 i=n1 is the loop-variable of the parallel loop.

Experiment results (4)

```
integer :: l3
l3(n1,n2,n3) = n1 + n2*D23 + n3*D33 + D43
```

```
!$omp parallel do DEFAULT(__AUTO)
    do k = MAX(ks,KSTA(ib)),MIN(ke+1,KEND(ib))
        do j = MAX(js,JEND(ib)),MIN(je,JEND(ib))
            do i = MAX(is,IPERBI),MIN(ie,IPERTI-1)
                ADDFLG(l3(i,j+1,k),2) = 0.
            end do
        end do
    end do
!$omp end parallel do
```

D23, D33 and D43 unknown at compile-time and invariant in the parallel region, but this time in l3 k=n3 is the loop-variable of the parallel loop.

Experiment results (4)

```
integer :: l3
l3(n1,n2,n3) = n1 + n2*D23 + n3*D33 + D43
```

```
!$omp parallel DEFAULT(__AUTO) SHARED(addflg)
    do k = MAX(ks,KSTA(ib)),MIN(ke+1,KEND(ib))
        do j = MAX(js,JEND(ib)),MIN(je,JEND(ib))
            do i = MAX(is,IPERBI),MIN(ie,IPERTI-1)
                ADDFLG(l3(i,j+1,k),2) = 0.
            end do
        end do
    end do
!$omp end parallel do
```

D23, D33 and D43 unknown at compile-time and invariant in the parallel region, but this time in l3 k=n3 is the loop-variable of the parallel loop.

Experiment results (5)

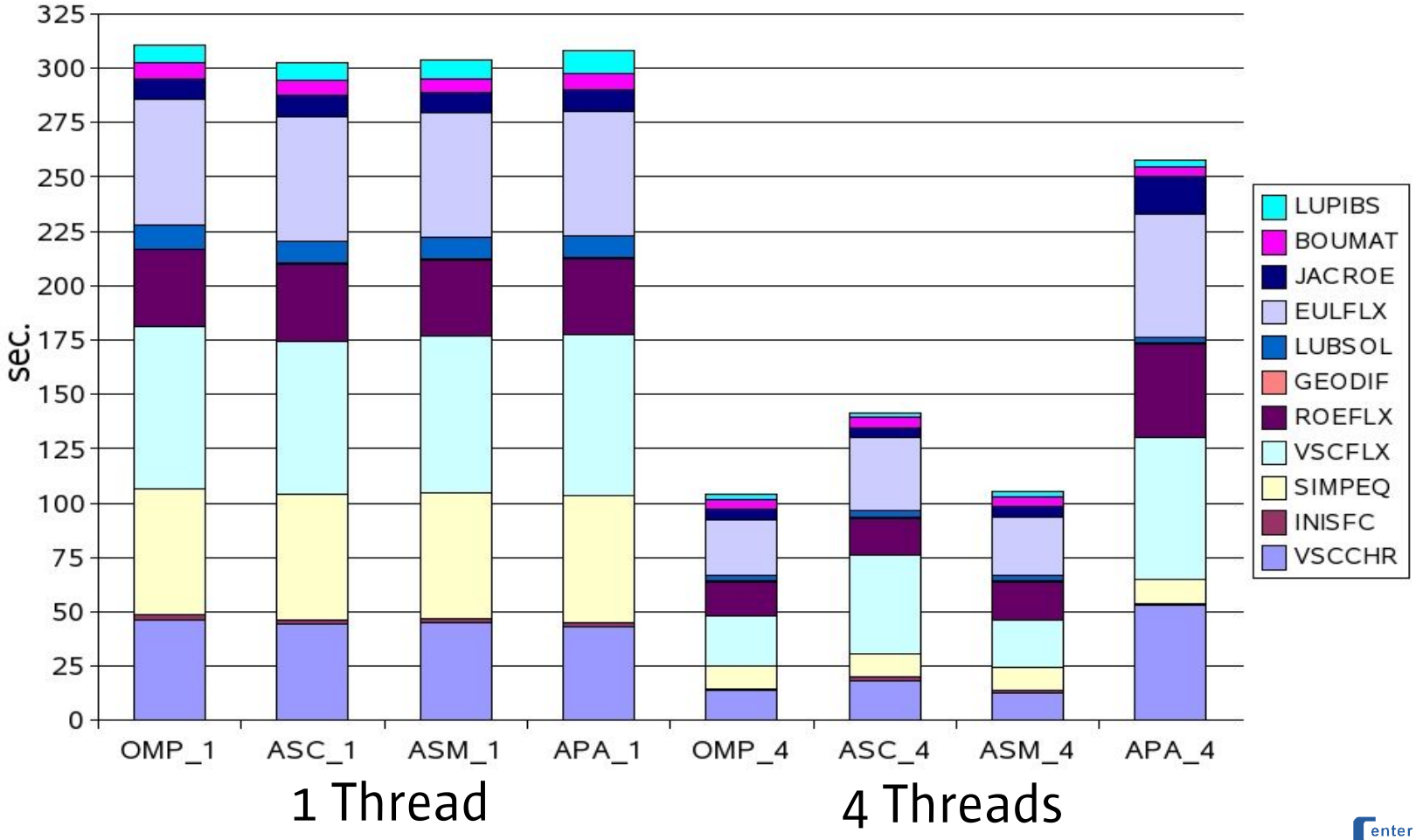
Routine	Lines of code	# parallel regions (Loops)	# serialized parallel regions	# explicitly privatized variables	# variables that could not be autoscoped
VSCCHR	**2984**	**2 (2)**	**0**	**880**	**0**
INISFC	244	4 (7)	4	16	4
SIMPEQ	386	9 (10)	0	48	0
VSCFLX	**466**	**2 (2)**	**2**	**10**	**2**
ROEFLX	879	2 (4)	0	220	0
GEODIF	132	1 (1)	1	15	3
LUBSOL	149	3 (3)	0	5	0
EULFLX	660	7 (10)	4	96	4
JACROE	422	2 (4)	0	42	0
BOUMAT	147	1 (2)	0	3	0
LUPIBS	693	1 (43)	0	54	0
Total	**7162**	**34 (88)**	**11**	**1389**	**13**

Experiment results (5)

Routine	Lines of code	# parallel regions (Loops)	# serialized parallel regions	# explicitly privatized variables	# variables that could not be autoscoped
VSCCHR	**2984**	**2 (2)**	**0**	**880**	**0**
INISFC	244	4 (7)	4	16	4
SIMPEQ	386	9 (10)	0	48	0
VSCFLX	**466**	**2 (2)**	**2**	**10**	**2**
ROEFLX	879	2 (4)	0	220	0
GEODIF	132	1 (1)	1	15	3
LUBSOL	149	3 (3)	0	5	0
EULFLX	660	7 (10)	4	96	4
JACROE	422	2 (4)	0		
BOUMAT	147	1 (2)	0	3	0
LUPIBS	693	1 (43)	0	54	0
Total	**7162**	**34 (88)**	**11**	**1389**	**13**

100x difference

Experiment results (6)



Conclusion

- Performance
  - Edited autoscoping equals original OpenMP
  - Autoscoping is better than autoparallel
- Amount of manual scoping work
  - 100x difference
  - Prevented many (possible) errors
- Autoscoping rocks ...

Thank you!

Yuan Lin[1]
Christian Terboven[2]
Dieter an Mey[2]
Nawal Copty[1]

[1]*Sun Microsystems, USA*
[2]*RWTH Aachen University, Germany*

RWTH AACHEN UNIVERSITY Center for Computing and Communication

Sun microsystems
We make the net work.