Approche Informatique de l'optimisation de code sur nymphea

Tina Odaka (Ifremer)

How do we improve the performance of a code??

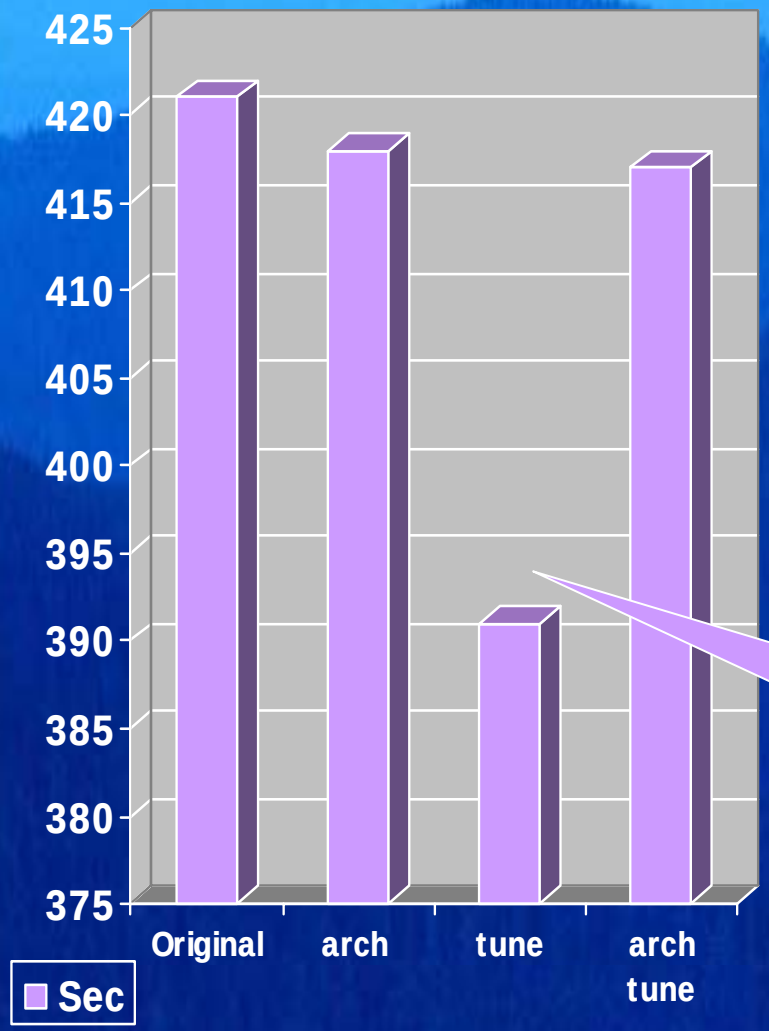
- Compiler optimisation options
- Manual optimisation
  - Analyse program using profiler.
  - Find redundant lines and improve the code.
- Use more than one CPU.
  - OpenMP (max 4 CPU on nymphea .i.e. max 4 threads)
    - Easy to parallelise but limited amount of resource.
  - MPI (max 4 CPU*11 machine on nymphea .i.e. max 44 nodes)
    - Takes time to parallelise but unlimited amount of resource.

Example:

- **Optimization of the operational version of MARS3D**
- **Project SOCOM** (Systeme d'Oceanographie Cotiere Operationel en Mediterranee)
  - **MARS3D models the actual hydrodynamic situation of the north occidental Mediterranean Sea.**
  - **F77, Parallelised using OpenMP. (max. 4 threads on nymphea.)**

Compiler optimisation options

Machine specific compiling options for Nymphea



- Original code: “-O5”
- Nymphea have Alpha EV68 1.25 GHz CPUs. Ev68 processor have the same instruction set as ev67 processors.
  - -tune ev67
  - -arch ev67

“-tune ev67” gain 7% of total calculation time

Manual optimisation: Tools for analysing the program

- Traditional
  - prof: need to re-compile with “–p” option
  - hiprof: no need to re-compile. Hierarchy profiler.
    - hiprof exe => creates exe.hiprof
    - ./exe.hiprof => creates exe.hiout
    - gprof exe.hiprof exe.hiout
- Graphical interface
  - dxprof
    - Can use prof, gprof, hiprof, pixie, with « click »
  - Visual Threads (dxthreads)
    - Dynamically analyze a program using threads, cf.: OpenMP.

Analysis of the Progra

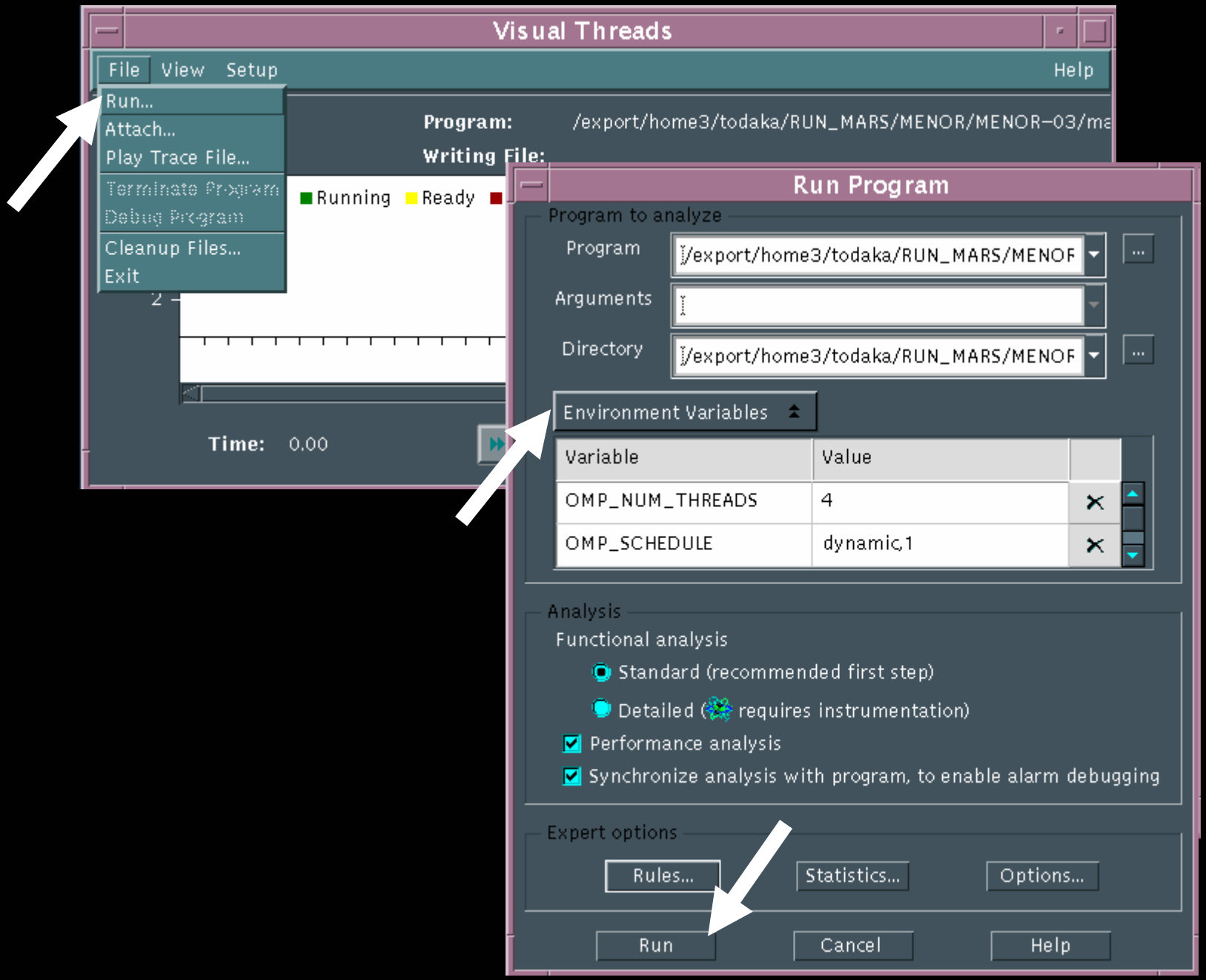
Why smagorinsky.f is costing this much??

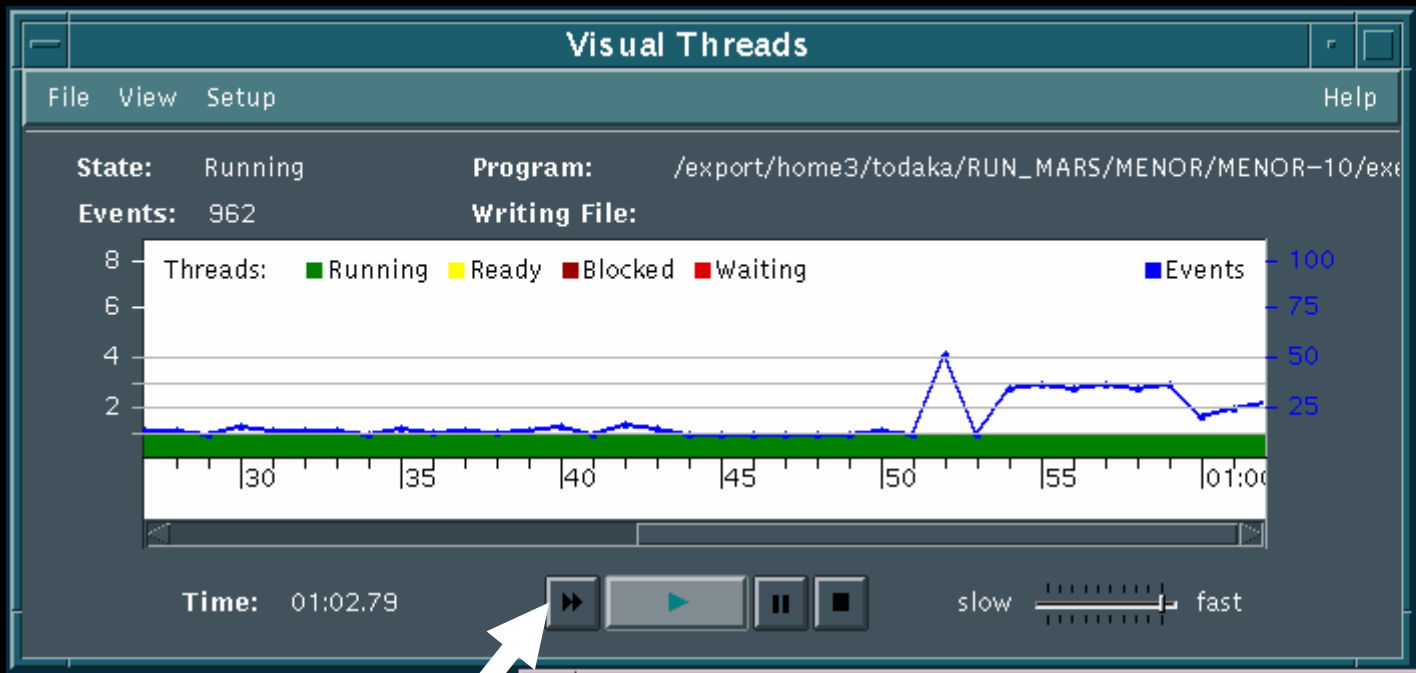
```
Profile listing generated Fri Sep  9 09:53:01 2005 with:
   prof -numbers exe2001prof mon.out

------------------------------------------------------------------------
*  -p[rocedures] using pc-sampling;                                     *
*  sorted in descending order by total time spent in each procedure;    *
*  unexecuted procedures excluded                                       *
------------------------------------------------------------------------

Each sample covers 8.00 byte(s) for 0.00019% of 508.7021 seconds

%time     seconds  cum %   cum sec  procedure (file)

 10.7     54.5039   10.7     54.50 fluy_ (<exe2001prof>:"./petitf/fluy.f":3)
 10.2     51.6514   20.9    106.16 _46_smagorinsky_ (<exe2001prof>:"./petitf/smagorinsky.f":46)
  8.8     44.8203   29.7    150.98 _83_adv_ (<exe2001prof>:"./petitf/adv.f":83)
  8.7     44.3623   38.4    195.34 equeta_ (<exe2001prof>:"./petitf/equeta.f":3)
  7.5     38.2090   45.9    233.55 flux_ (<exe2001prof>:"./petitf/flux.f":3)
```

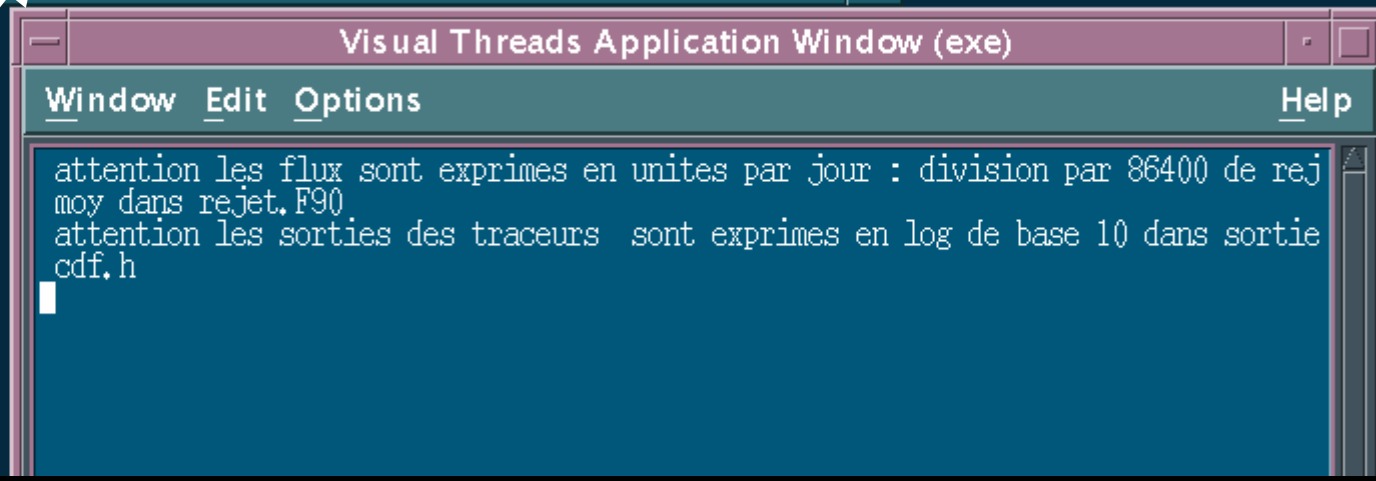
- “prof” gives you the CPU usage of each part of the code.
  - Subroutine smagorinsky; 10 % of CPU usage

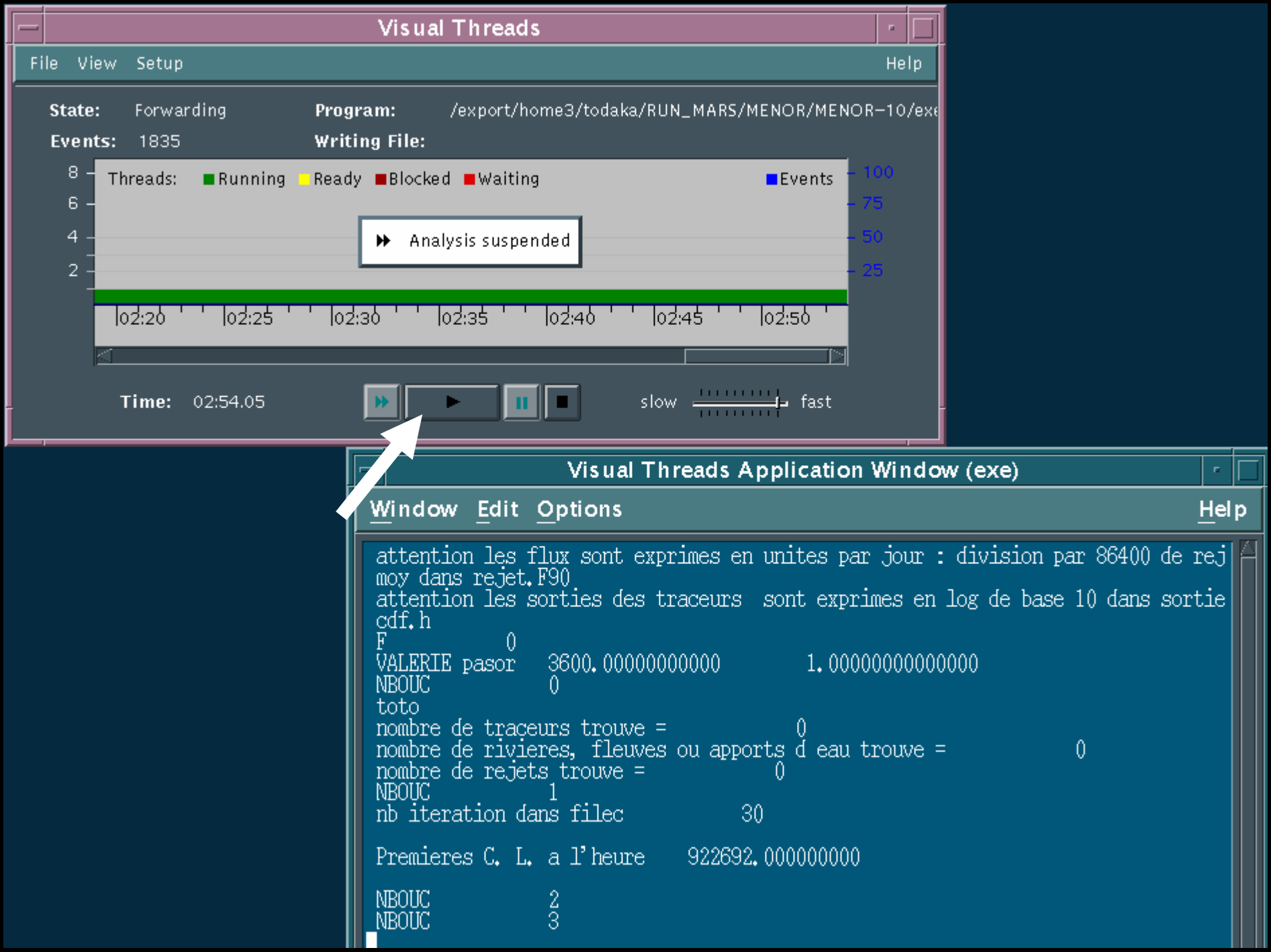
Visual Threads

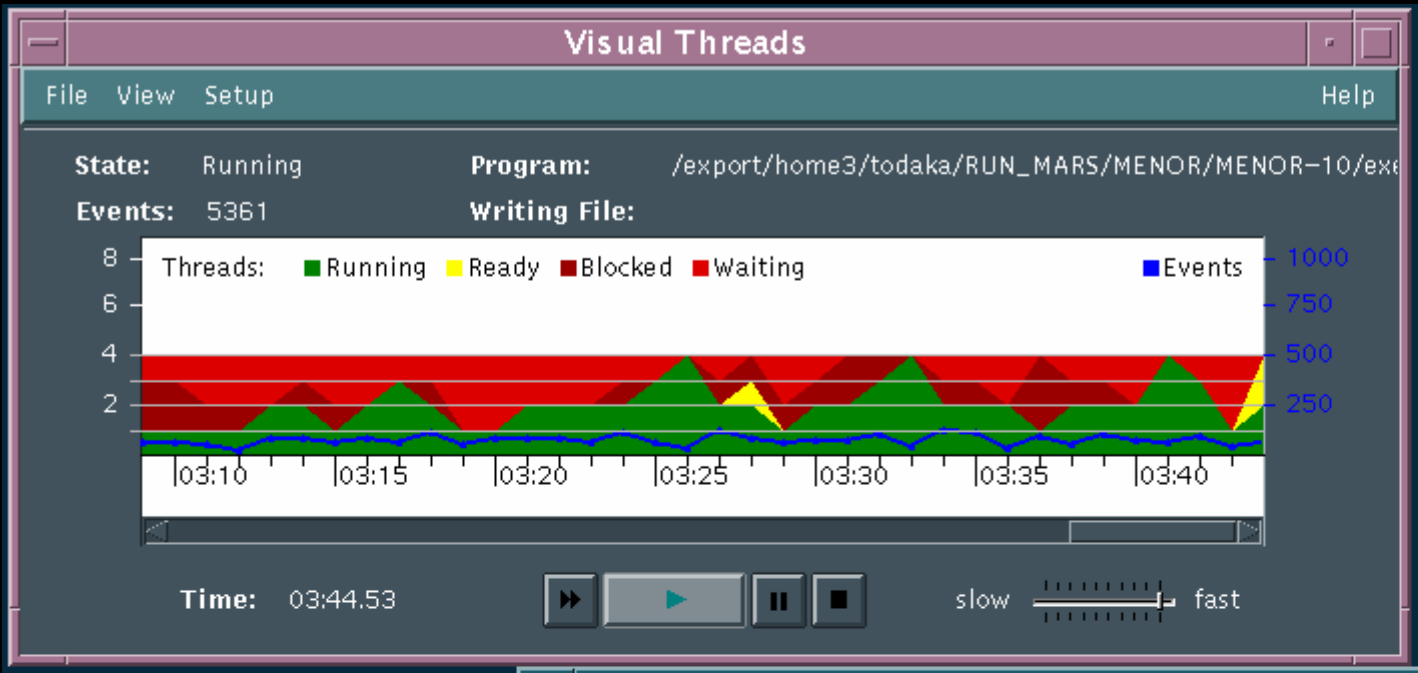
- nymphea0>62% dxthreads

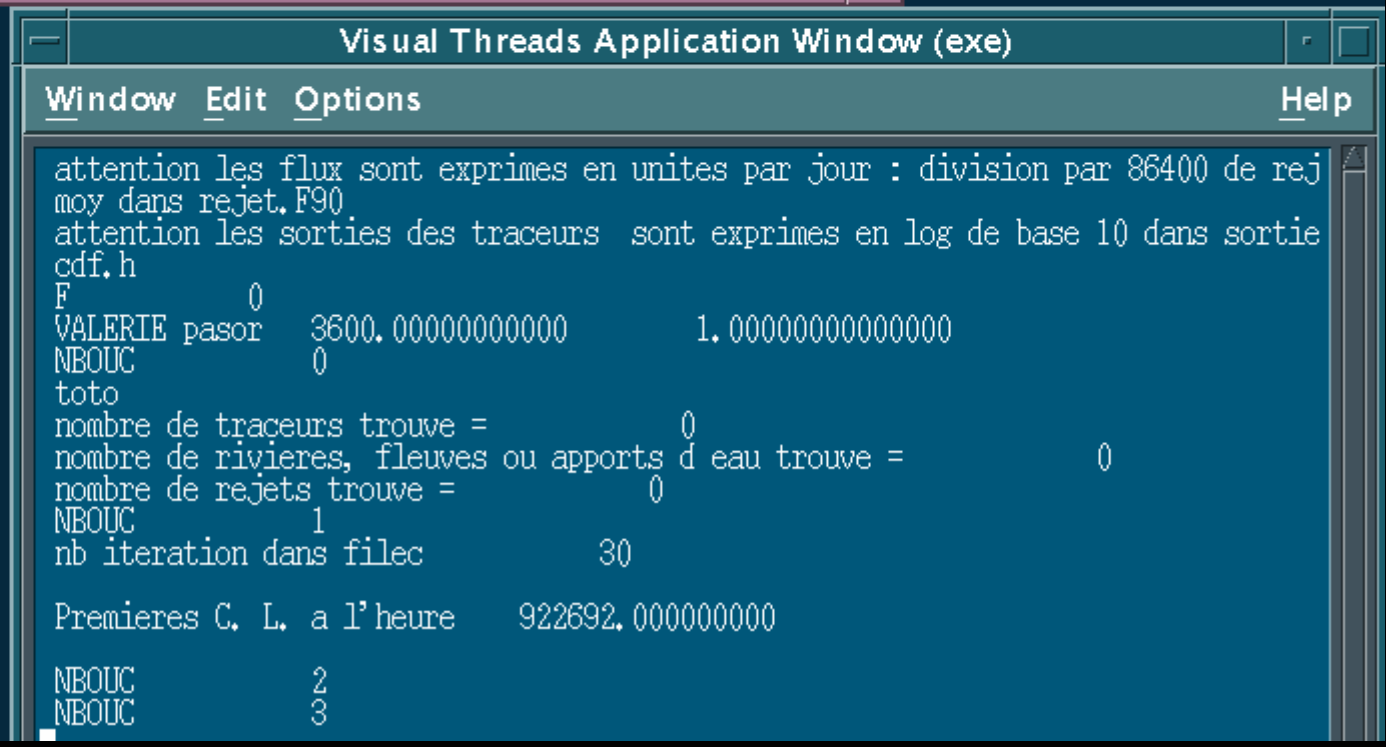

Visual Threads
File
View
Setup
Help
Run...
Attach...
Play Trace File...
Terminate Program
Debug Program
Cleanup Files...
Exit
Program:
/export/home3/todaka/RUN_MARS/MENOR/MENOR-03/ma
Writing File:
Running
Ready
Time:
0.00
Run Program
Program to analyze
Program
/export/home3/todaka/RUN_MARS/MENOF
Arguments
Directory
/export/home3/todaka/RUN_MARS/MENOF
Environment Variables
Variable
Value
OMP_NUM_THREADS
4
OMP_SCHEDULE
dynamic,1
Analysis
Functional analysis
Standard (recommended first step)
Detailed (requires instrumentation)
Performance analysis
Synchronize analysis with program, to enable alarm debugging
Expert options
Rules...
Statistics...
Options...
Run
Cancel
Help

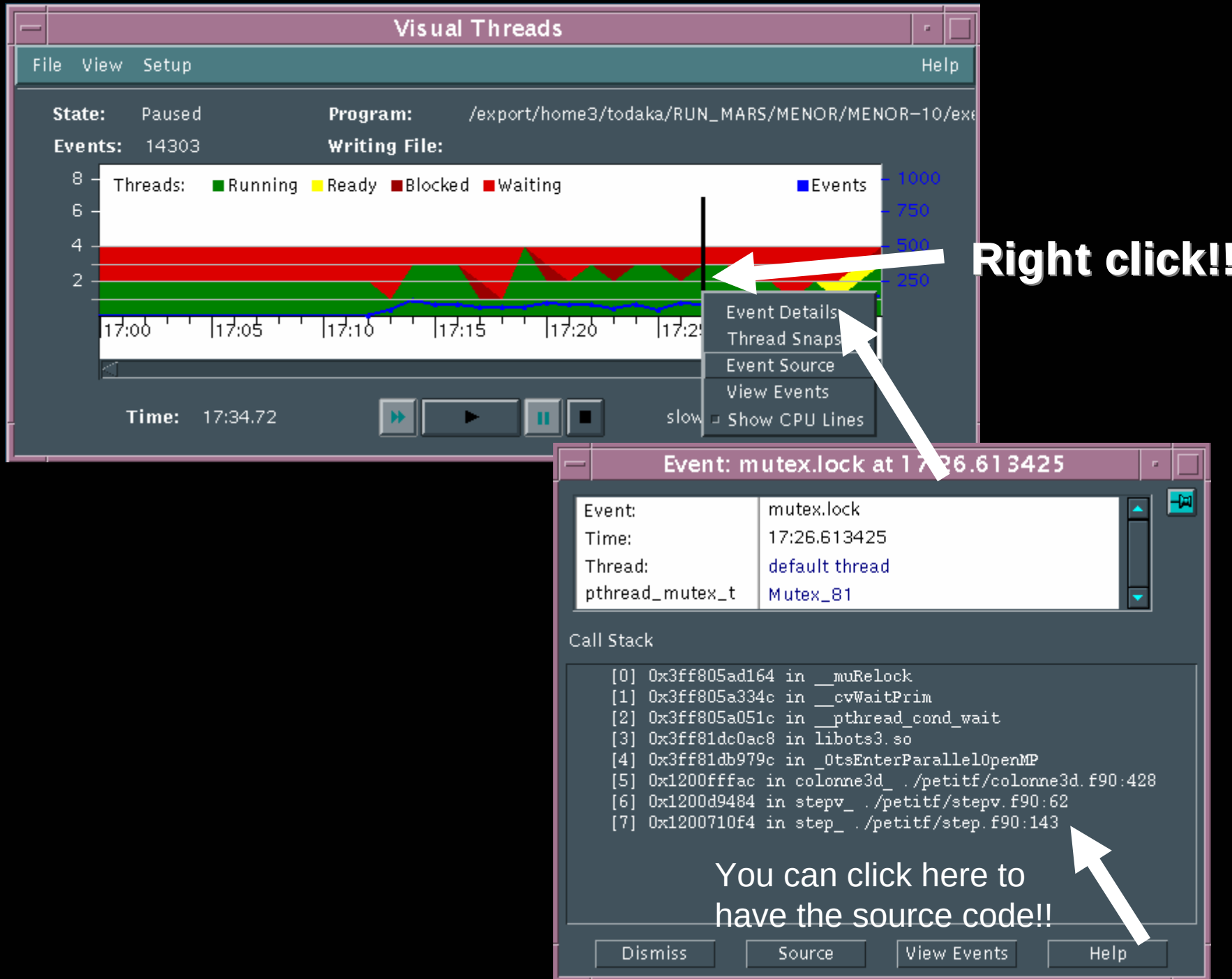

Visual Threads
File View Setup Help
State: Running
Program: /export/home3/todaka/RUN_MARS/MENOR/MENOR-10/exe
Events: 962
Writing File:
Threads: Running Ready Blocked Waiting
Events
8
6
4
2
100
75
50
25
30
35
40
45
50
55
Time: 01:02.79
slow
fast

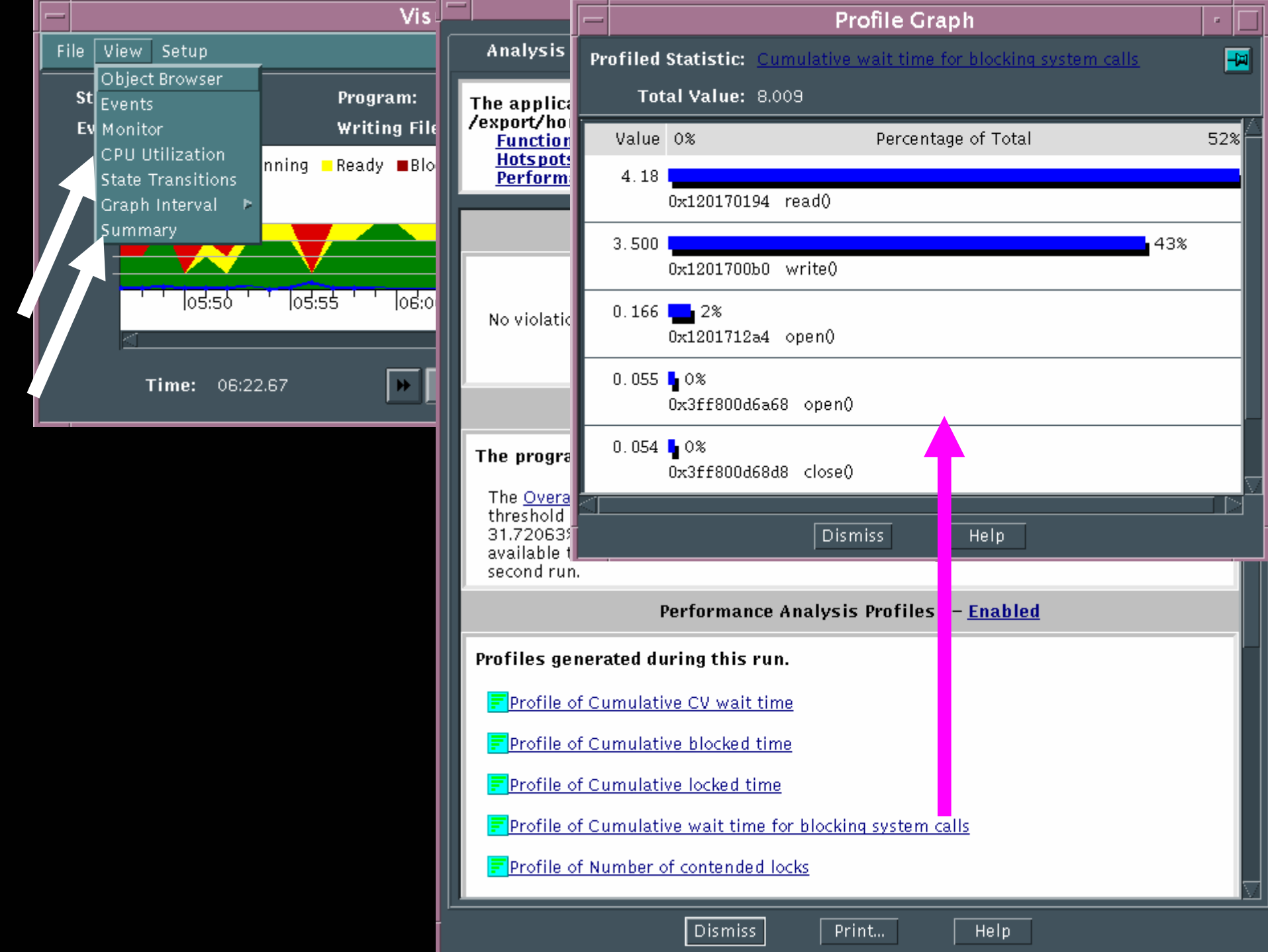

Visual Threads Application Window (exe)
Window Edit Options Help
attention les flux sont exprimes en unites par jour : division par 86400 de rej moy dans rejet.F90
attention les sorties des traceurs sont exprimes en log de base 10 dans sortie cdf.h


Visual Threads
File View Setup Help
State: Forwarding
Program: /export/home3/todaka/RUN_MARS/MENOR/MENOR-10/exe
Events: 1835
Writing File:
Threads: Running Ready Blocked Waiting
Events
8
6
4
2
100
75
50
25
Analysis suspended
02:20
02:25
02:30
02:35
02:40
02:45
02:50
Time: 02:54.05
slow
fast
Visual Threads Application Window (exe)
Window Edit Options Help
attention les flux sont exprimes en unites par jour : division par 86400 de rej
moy dans rejet.F90
attention les sorties des traceurs sont exprimes en log de base 10 dans sortie
cdf.h
F 0
VALERIE pasor 3600.00000000000 1.00000000000000
NBOUC 0
toto
nombre de traceurs trouve = 0
nombre de rivieres, fleuves ou apports d eau trouve = 0
nombre de rejets trouve = 0
NBOUC 1
nb iteration dans filec 30
Premieres C. L. a l'heure 922692.000000000
NBOUC 2
NBOUC 3


Visual Threads
File View Setup Help
State: Running
Program: /export/home3/todaka/RUN_MARS/MENOR/MENOR-10/exe
Events: 5361
Writing File:
Threads: Running Ready Blocked Waiting
Events
8
6
4
2
1000
750
500
250
03:10
03:15
03:20
03:25
03:30
03:35
03:40
Time: 03:44.53
slow
fast


Visual Threads Application Window (exe)
Window Edit Options Help
attention les flux sont exprimes en unites par jour : division par 86400 de rej
moy dans rejet.F90
attention les sorties des traceurs sont exprimes en log de base 10 dans sortie
cdf.h
F 0
VALERIE pasor 3600.00000000000 1.00000000000000
NBOUC 0
toto
nombre de traceurs trouve = 0
nombre de rivieres, fleuves ou apports d eau trouve = 0
nombre de rejets trouve = 0
NBOUC 1
nb iteration dans filec 30
Premieres C. L. a l'heure 922692.000000000
NBOUC 2
NBOUC 3


Visual Threads
File View Setup Help
State: Paused
Program: /export/home3/todaka/RUN_MARS/MENOR/MENOR-10/exe
Events: 14303
Writing File:
Threads: Running Ready Blocked Waiting Events
8
6
4
2
1000
750
500
250
17:00
17:05
17:10
17:15
17:20
Event Details
Thread Snaps
Event Source
View Events
Show CPU Lines
Time: 17:34.72
slow
Right click!!
Event: mutex.lock at 17:26.613425
Event: mutex.lock
Time: 17:26.613425
Thread: default thread
pthread_mutex_t Mutex_81
Call Stack
[0] 0x3ff805ad164 in __muRelock
[1] 0x3ff805a334c in __cvWaitPrim
[2] 0x3ff805a051c in __pthread_cond_wait
[3] 0x3ff81dc0ac8 in libots3.so
[4] 0x3ff81db979c in _OtsEnterParallelOpenMP
[5] 0x1200fffac in colonne3d_ ./petitf/colonne3d.f90:428
[6] 0x1200d9484 in stepv_ ./petitf/stepv.f90:62
[7] 0x1200710f4 in step_ ./petitf/step.f90:143
You can click here to have the source code!!
Dismiss
Source
View Events
Help

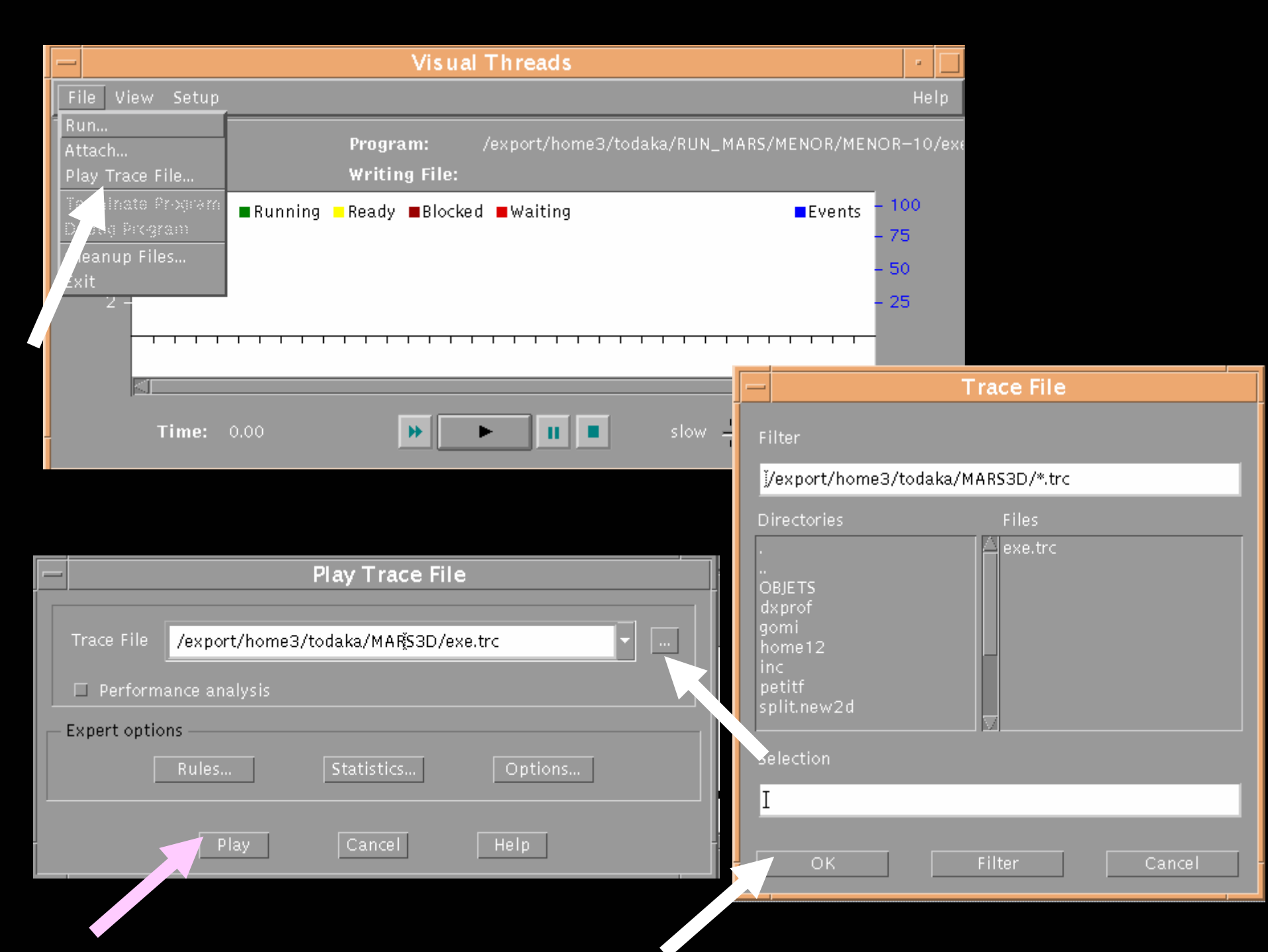

Vis
File
View
Setup
Object Browser
Events
Monitor
CPU Utilization
State Transitions
Graph Interval
Summary
Program:
Writing File
Ready
05:50
05:55
Time: 06:22.67
Analysis
No violatic
The progra
Profile Graph
Profiled Statistic: Cumulative wait time for blocking system calls
Total Value: 8.009
Value 0%
Percentage of Total
52%
4.18
0x120170194 read()
3.500
43%
0x1201700b0 write()
0.166
2%
0x1201712a4 open()
0.055
0%
0x3ff800d6a68 open()
0.054
0%
0x3ff800d68d8 close()
Dismiss
Help
Performance Analysis Profiles – Enabled
Profiles generated during this run.
Profile of Cumulative CV wait time
Profile of Cumulative blocked time
Profile of Cumulative locked time
Profile of Cumulative wait time for blocking system calls
Profile of Number of contended locks
Dismiss
Print...
Help

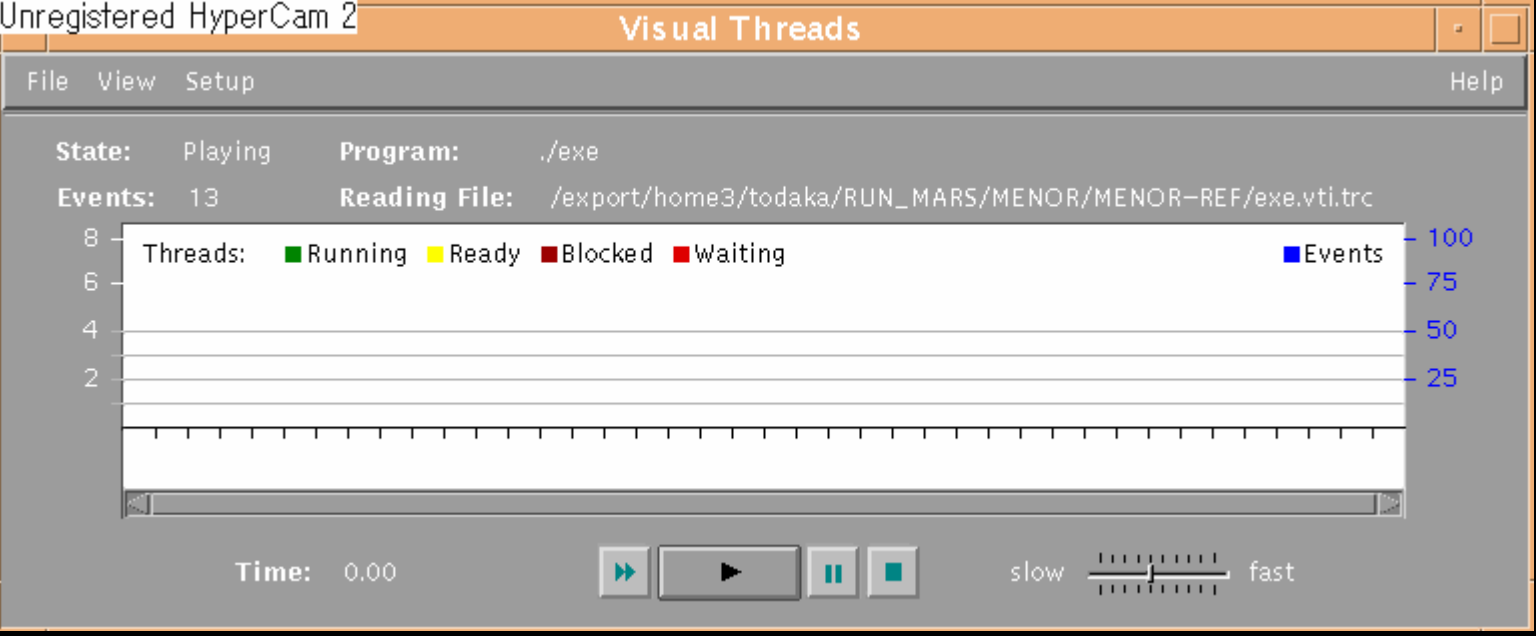
Q: Can’t I use batch job to run dxthreads?? Nymphea0 is crowded!!

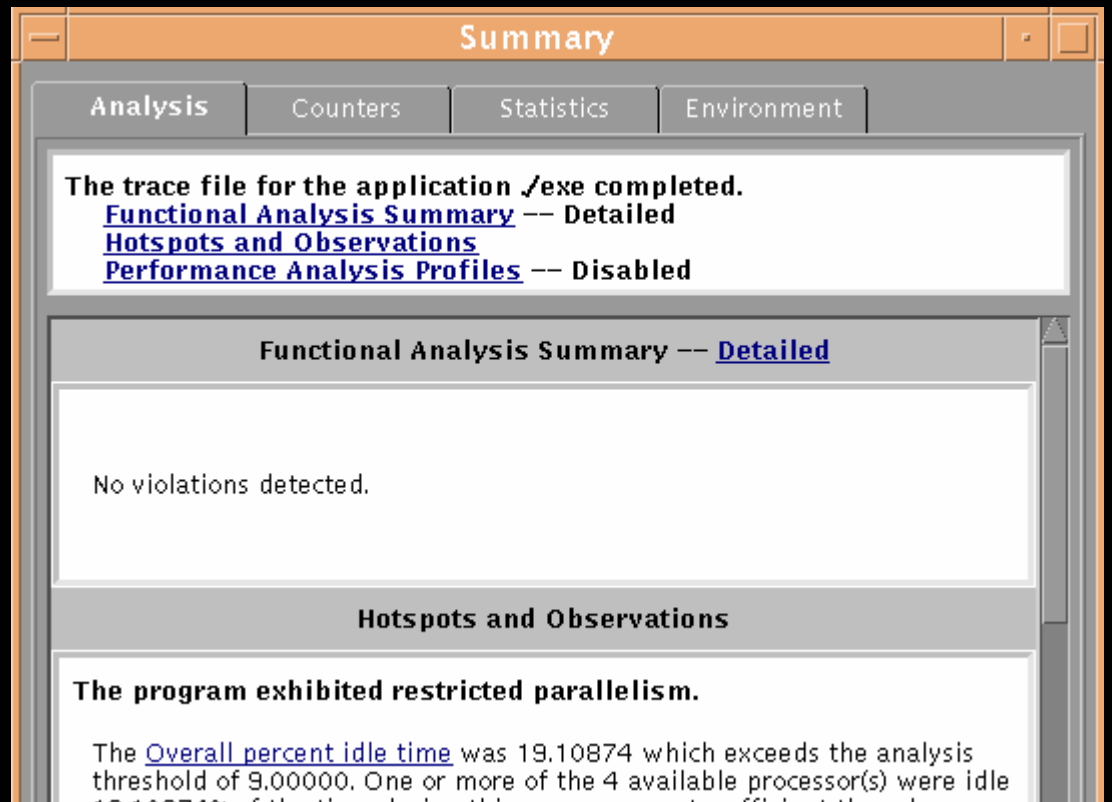
A: You can not run the dxthreads in the batch job, but you can create a trace file (exe.trc) by vttrace:

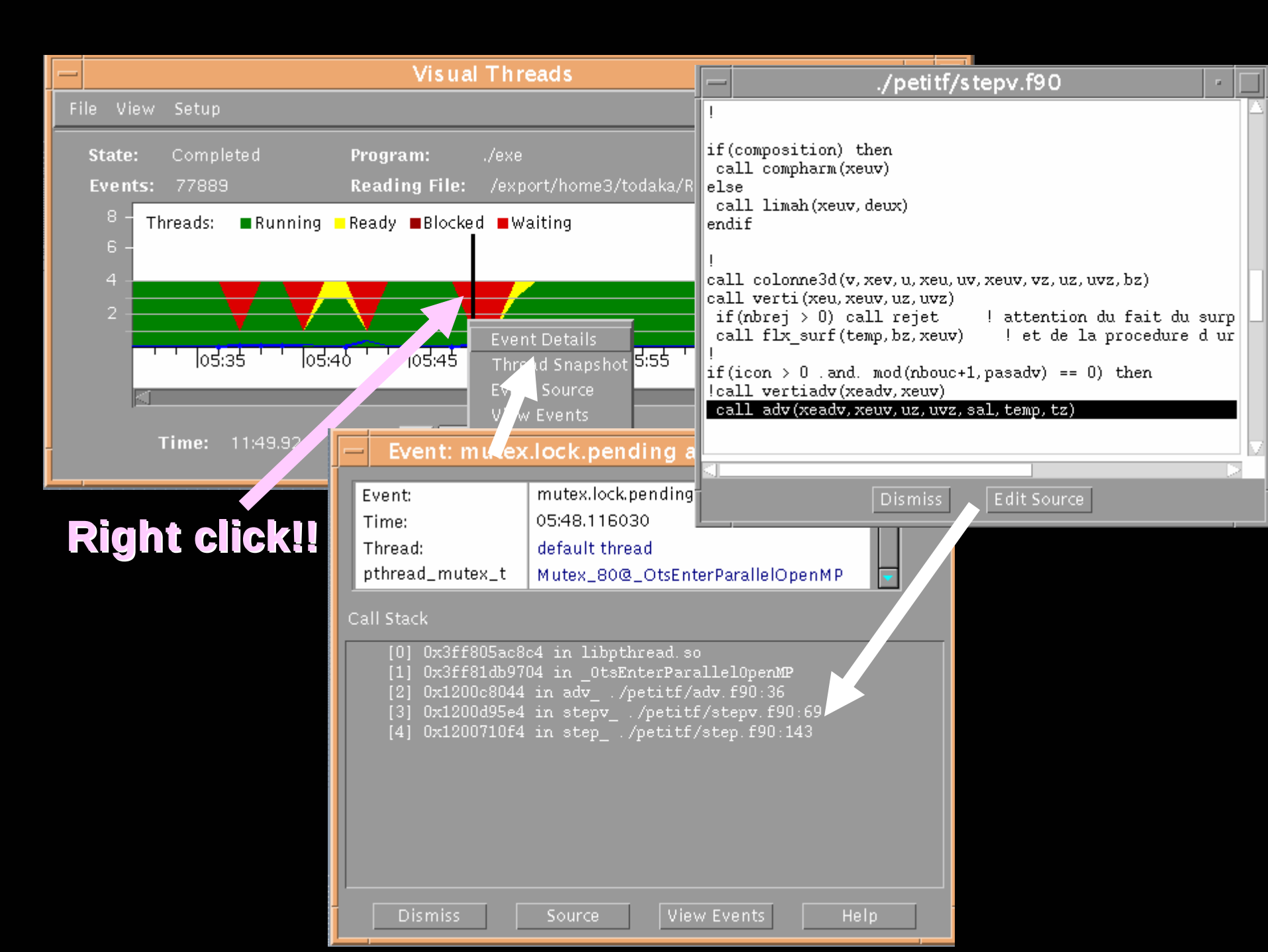
vttrace –e all ./exe

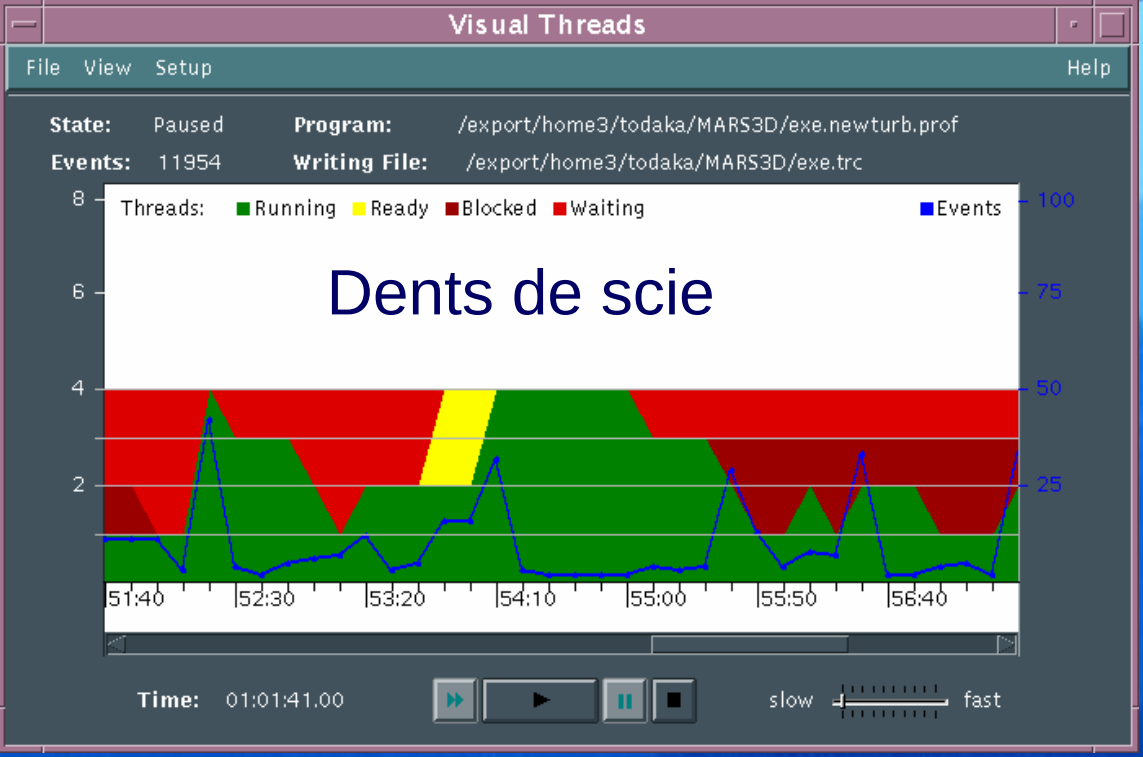
and play the trace file with dxthreads.


Visual Threads
File View Setup Help
Run...
Attach...
Play Trace File...
Exit
Program: /export/home3/todaka/RUN_MARS/MENOR/MENOR-10/exe
Writing File:
Running Ready Blocked Waiting Events
100
75
50
25
Time: 0.00
slow
Play Trace File
Trace File /export/home3/todaka/MARS3D/exe.trc
Performance analysis
Expert options
Rules... Statistics... Options...
Play Cancel Help
Trace File
Filter
/export/home3/todaka/MARS3D/*.trc
Directories
Files
.
..
OBJETS
dxprof
gomi
home12
inc
petitf
split.new2d
exe.trc
Selection
OK Filter Cancel



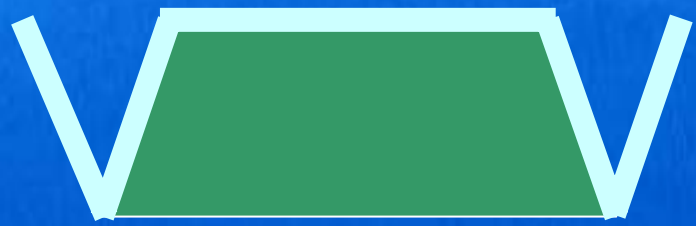



Visual Threads
File View Setup
State: Completed
Program: ./exe
Events: 77889
Reading File: /export/home3/todaka/R
Threads: Running Ready Blocked Waiting
8
6
4
2
05:35
05:40
05:45
5:55
Event Details
Thread Snapshot
Source
View Events
Time: 11:49.92
./petitf/stepv.f90
!
if(composition) then
call compharm(xeuv)
else
call limah(xeuv, deux)
endif
!
call colonne3d(v, xev, u, xeu, uv, xeuv, vz, uz, uvz, bz)
call verti(xeu, xeuv, uz, uvz)
if(nbrej > 0) call rejet ! attention du fait du surp
call flx_surf(temp, bz, xeuv) ! et de la procedure d ur
!
if(icon > 0 .and. mod(nbouc+1, pasadv) == 0) then
!call vertiadv(xeadv, xeuv)
call adv(xeadv, xeuv, uz, uvz, sal, temp, tz)
Dismiss
Edit Source
Event: mutex.lock.pending a
Event: mutex.lock.pending
Time: 05:48.116030
Thread: default thread
pthread_mutex_t Mutex_80@_OtsEnterParallelOpenMP
Call Stack
[0] 0x3ff805ac8c4 in libpthread.so
[1] 0x3ff81db9704 in _OtsEnterParallelOpenMP
[2] 0x1200c8044 in adv_ ./petitf/adv.f90:36
[3] 0x1200d95e4 in stepv_ ./petitf/stepv.f90:69
[4] 0x1200710f4 in step_ ./petitf/step.f90:143
Dismiss
Source
View Events
Help
Right click!!




```
!$OMP DO
do  i=1,1000
  A(i)=b(i)+c(i)
enddo
!$OMP END DO
!$OMP DO
do  i=1,1000
  d(i)=b(i)+c(i)
enddo
!$OMP END DO
```

We need full load!!



Barrier!!
Bottle neck

Smagorinsky

```
!$OMP PARALLEL
!$OMP DO SCHEDULE(RUNTIME)
      do j=jmin+1,jmax-1
       do i=imin+1,imax-1
        do k=1,kmax
       uz_v(k,i,j)=0.25*(uz(k,i,j)+uz(k,i-1,j)+uz(k,i,j+1)+uz(k,i-1,j+1))
       vz_u(k,i,j)=0.25*(vz(k,i,j)+vz(k,i,j-1)+vz(k,i+1,j)+vz(k,i+1,j-1))
        end do
       end do
      end do
!$OMP END DO
!$OMP DO SCHEDULE(RUNTIME)
      do j=jmin+1,jmax-1
        do k=1,kmax
         uz_v(k,imin,j)=uz_v(k,imin+1,j)
         vz_u(k,imin,j)=vz_u(k,imin+1,j)
         uz_v(k,imax,j)=uz_v(k,imax-1,j)
         vz_u(k,imax,j)=vz_u(k,imax-1,j)
        end do
      end do
!$OMP END DO
!$OMP DO SCHEDULE(RUNTIME)
      do i=imin+1,imax-1
        do k=1,kmax
         uz_v(k,i,jmin)=uz_v(k,i,jmin+1)
         vz_u(k,i,jmin)=vz_u(k,i,jmin+1)
         uz_v(k,i,jmax)=uz_v(k,i,jmax-1)
         vz_u(k,i,jmax)=vz_u(k,i,jmax-1)
        end do
      end do
!$OMP END DO
```

```
!$OMP DO SCHEDULE(RUNTIME)
      do j=jmin+leponge*epongs+1,jmax-leponge*epongn-1
      do i=imin+leponge*epongw+1,imax-leponge*eponge-1
      if (igd(i,j).ne.4) then
      do k=1,kmax
        vish_xe(k,i,j)=
     &   cosmag*dy*dxu(j)*sqrt(((uz(k,i,j)-uz(k,i-1,j))/dxu(j)
     &     -tfiusr(j)*0.5*(vz(k,i,j)+vz(k,i,j-1)))**2
     &    +((vz(k,i,j)-vz(k,i,j-1))/dy)**2
     &    +0.5*((uz_v(k,i,j)-uz_v(k,i,j-1))/dy
     &     +tfiusr(j)*0.5*(uz(k,i,j)+uz(k,i-1,j))
     &          +(vz_u(k,i,j)-vz_u(k,i-1,j))/dxu(j))**2)
        vish_xe(k,i,j)=min(max(vish_xe(k,i,j),vismin),vismax)
      enddo
      endif
      enddo
      enddo
!$OMP END DO
!$OMP DO SCHEDULE(RUNTIME)
      do j=jmin+leponge*epongs+1,jmax-leponge*epongn-1
      do i=imin+leponge*epongw+1,imax-leponge*eponge-1
      if (igd(i,j).ne.4) then
      do k=1,kmax
        vish_phi(k,i,j)=
     &   cosmag*dy*dxu(j)*sqrt(((uz_v(k,i+1,j)-uz_v(k,i,j))/dxv(j)
     &     -tfivsr(j)*0.5*(vz(k,i+1,j)+vz(k,i,j)))**2
     &    +((vz_u(k,i,j+1)-vz_u(k,i,j))/dy)**2
     &    +0.5*((uz(k,i,j+1)-uz(k,i,j))/dy
     &        +tfivsr(j)*0.5*(uz(k,i,j+1)+uz(k,i,j))
     &         +(vz(k,i+1,j)-vz(k,i,j))/dxv(j))**2)
        vish_phi(k,i,j)=min(max(vish_phi(k,i,j),vismin),vismax)
      enddo
      endif
      enddo
      enddo
!$OMP END DO
!$OMP END PARALLEL
```

- Original subroutine contains **5 OMP do loops**
  - Reduced to **3** on smagorinsky version 1 => **- 4 %**
  - Reduced to **1** on smagorinsky version 2 => **- 2 %**

Accessing matrix in good order

ij order

- **flux.f**

```
do j=jmin+2,jmax-1
      do i=ig(j),id(j)-1
      ..........
      do k=1,kmax
       uu=u(k,i,j)
```

- **fluy.f**

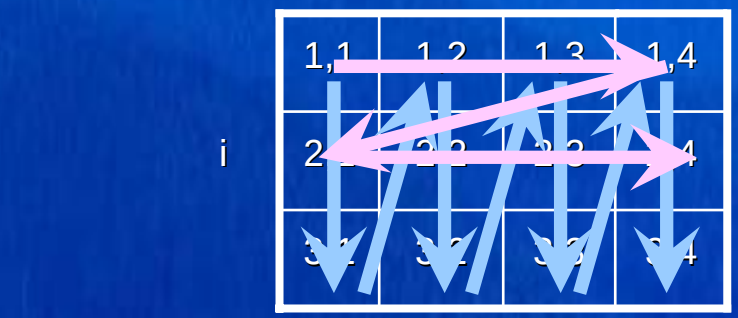
```
do i=imin+2,imax-1
      do j=jb(i),jh(i)-1
      ..........
      do k=1,kmax
       vv=v(k,i,j)
```

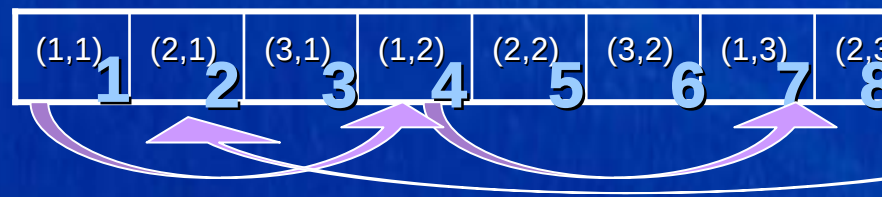
fluy.f do loop index is changed to

```
do j=jmin, jmax
   do i=imin+2,imax-1
   if(j.ge.jb(i).and.j.le.jh(i)-1) then
   ..................
```

(i,j)	Order
(1,1)	1
(1,2)	4
(1,3)	7
(1,4)	10
(2,1)	2
...	...

matrix a(i,j)





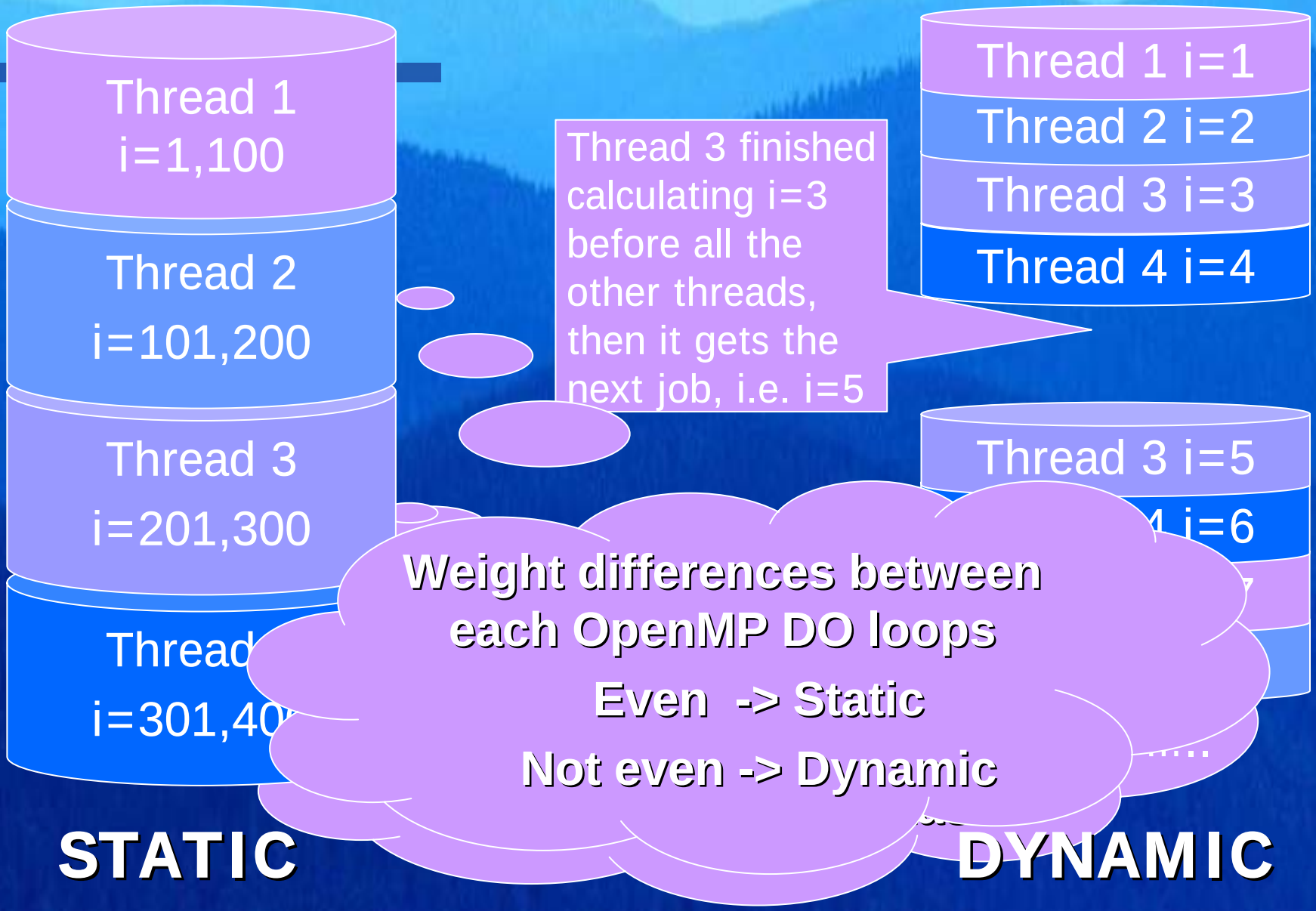
- Subroutine flux.f and fluy.f have very similar task
- Difference: order of access to 3 dimensional matrices

Suppressing redundant lines
Dkey_ntra0

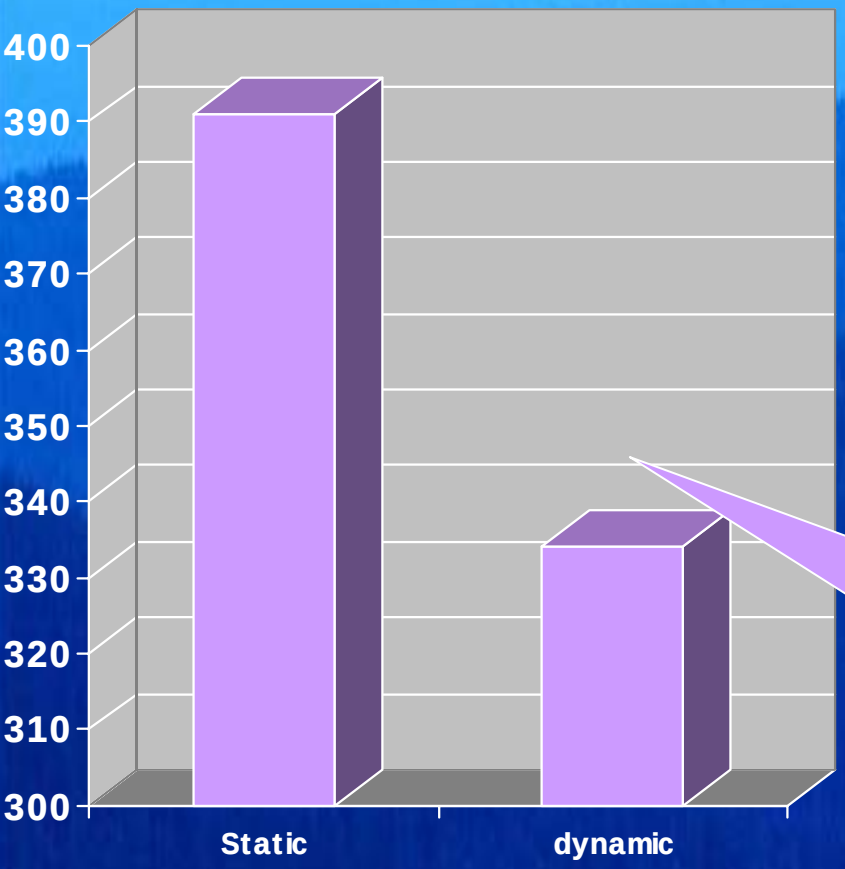
- "Partie traceurs"
  - Simulate the transportation or dilution of some pollutants or nutrients
  - Not used for the operational run! (ntra=0)
- CPU % used in "partie traceurs"
  - adv.f 1.82%
  - fluy.f +5%
  - flux.f 3.62%
  - ...
- Pre-processor 'Dkey_ntra0' created to suppress the redundant lines

```
do l=1,ntra
  u0(1)=y(1,l)/b(1)
  do k=2,kmax
    u0(k)=(y(k,l)-a(k)*u0(
.....
```

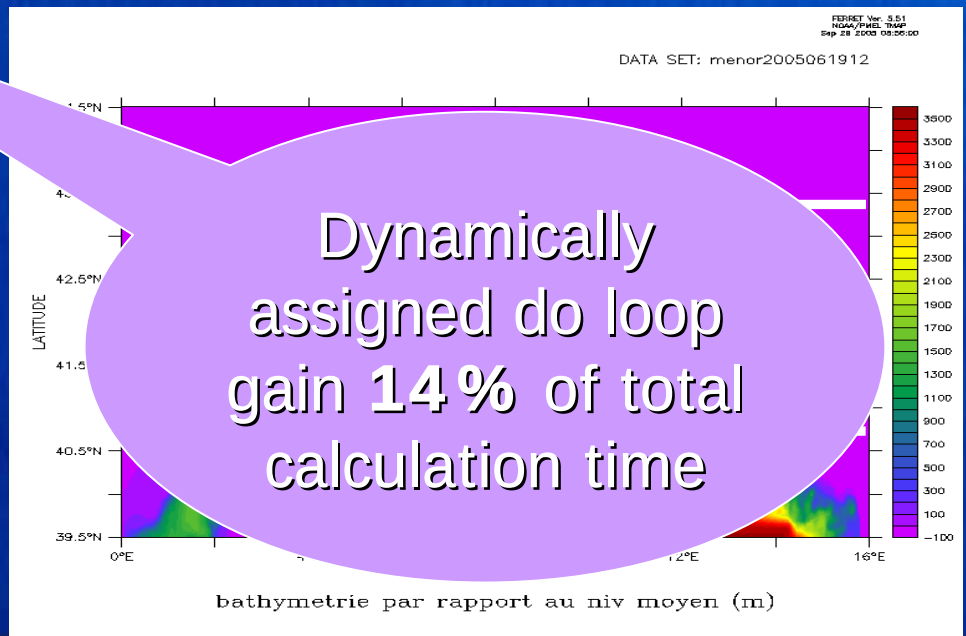
Scheduling OpenMP Do loop



MARS3D (operational version) Scheduling OpenMP Do loops

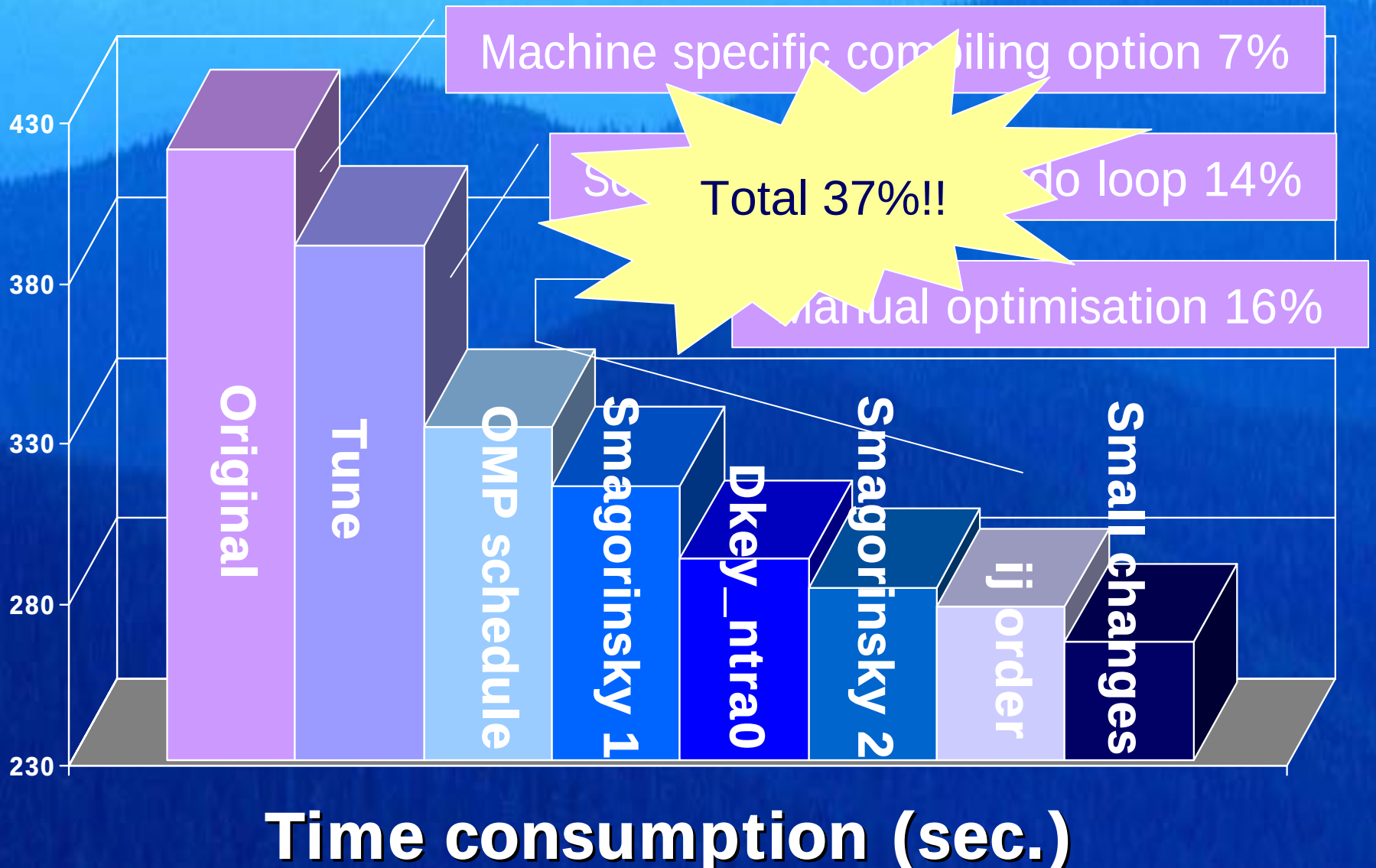


- Original code:270% of CPU usage with “static”
- Shape of the coast gave uneven load to each do loop.



Dynamically assigned do loop gain **14%** of total calculation time

SUMMARY


Machine specific compiling option 7%
do loop 14%
manual optimisation 16%
Total 37%!!
430
380
330
280
230
Original
Tune
OMP schedule
Smagorinsky 1
Dkey_ntra0
Smagorinsky 2
ij order
Small changes
Time consumption (sec.)

Any questions??

- Contact me!
  - todaka@ifremer.fr