pallas

Competence in High Performance Computing

Portabel MPI Tools at Work
- Cracking Performance Problems -

Werner Krotz-Vogel <krotz@pallas.com>

ICTP Cluster-Workshop, Trieste
02002-02-08

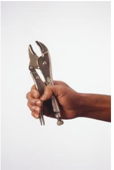
Pallas GmbH
Hermülheimer Straße 10
D-50321 Brühl, Germany

info@pallas.com
www.pallas.com

Tools for development and optimization

- Parallel Systems
  - from single PCs
  - Linux Farms
  - Clusters (PC2, Beowulf, Cplant)
  - to ASCI, RWC, Top500
  - and other places ...

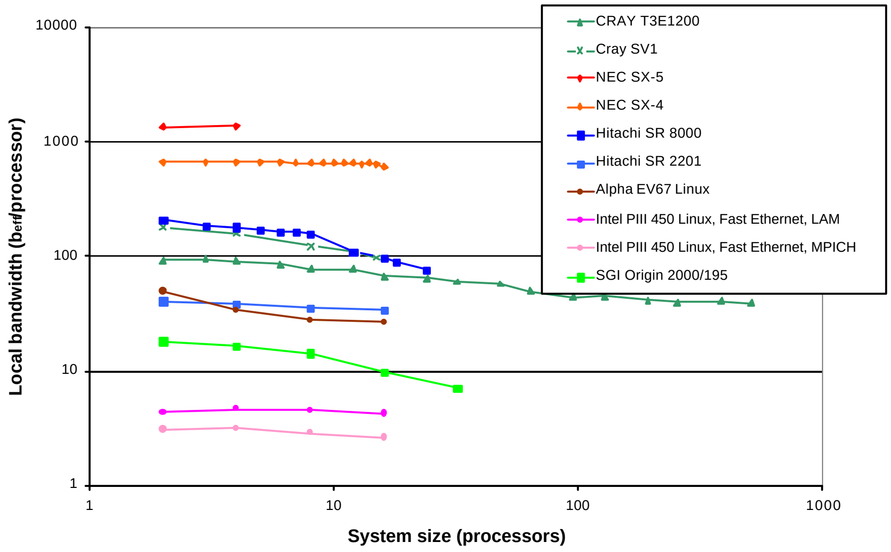
... and what do people use to program it ?

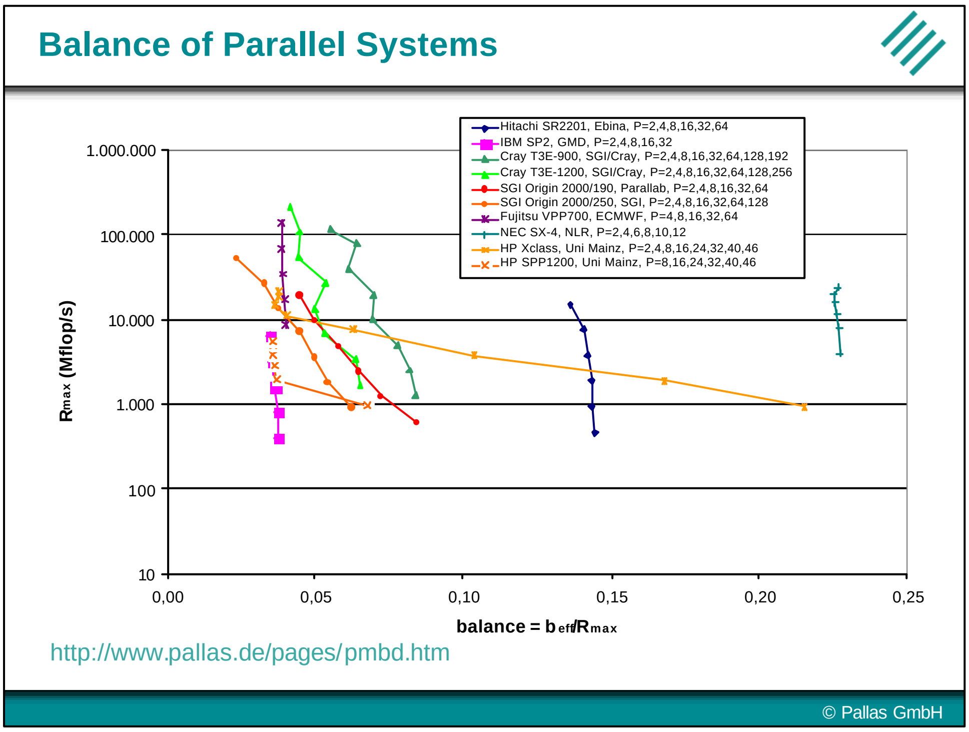


Pallas Company Background

- Leading European HPC software company
- World-wide market
- HPC Tool Products
- Porting and tuning of applications
- HPC training, Performance consulting, test and evaluation
- Implementation of MPI and profiling tools
- Studies (market, performance, competition, ...)

PMB@pallas.com - Pallas MPI Benchmarks (free)

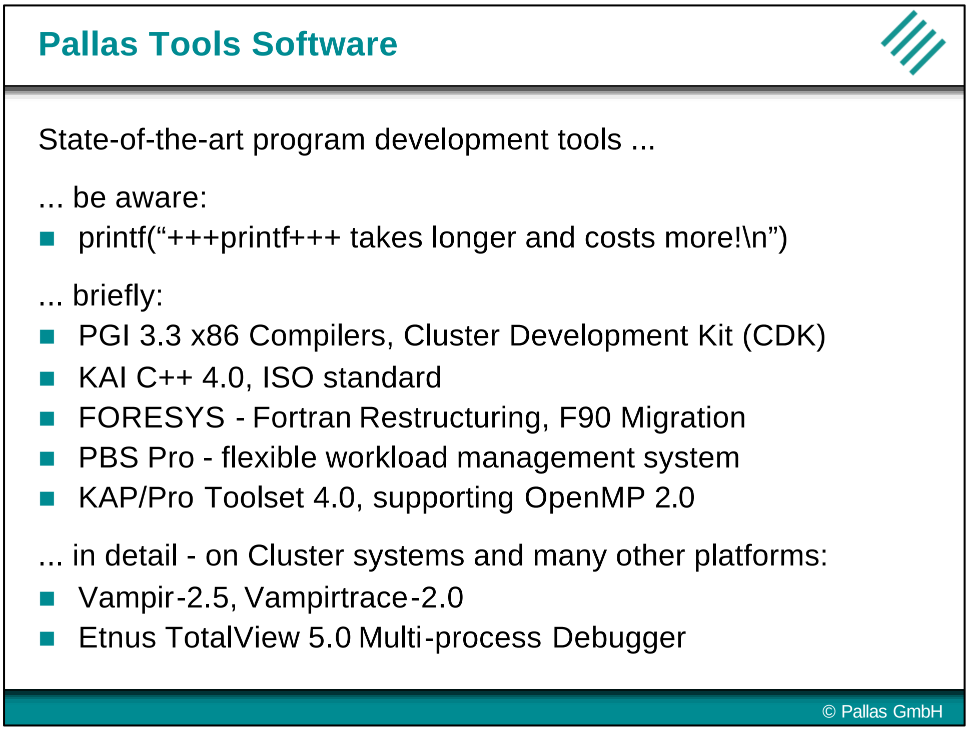




Balance of Parallel Systems

http://www.pallas.de/pages/pmbd.htm

© Pallas GmbH



Pallas Tools Software

State-of-the-art program development tools ...

... be aware:

- printf(“+++printf+++ takes longer and costs more!\n”)

... briefly:

- PGI 3.3 x86 Compilers, Cluster Development Kit (CDK)
- KAI C++ 4.0, ISO standard
- FORESYS - Fortran Restructuring, F90 Migration
- PBS Pro - flexible workload management system
- KAP/Pro Toolset 4.0, supporting OpenMP 2.0

... in detail - on Cluster systems and many other platforms:

- Vampir-2.5, Vampirtrace-2.0
- Etnus TotalView 5.0 Multi-process Debugger

© Pallas GmbH

PGI Cluster Development Kit (CDK)

Compilers & Tools ...

- PGI 3.3 x86 compilers , C, C++, F77, F90, HPF, pgrof, pgdbg
- SMP/OpenMP support for C, C++, F77, F90
- optimized CodeGen for P4, Athlon. New profile-feedback feature

... plus convenient add-on´s:

- parallel ScaLAPACK
- optimized BLAS, LAPACK
- MPI/mpich
- PVM
- PBS
- Tutorial, examples
- Cluster management utilities

KAI C++

The most modern, best performing, platform independant C++

- ISO C++ standard syntax, including exeptions and member templates
- ISO C++ standard class library
- multi-platform support
- meet C performance requirements
- thread safety (on most platforms)

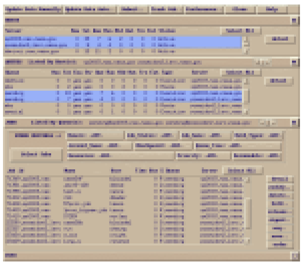
FORESYS

- Translates FORTRAN code (F77 - F95) into abstract syntax tree (ForLib)
- FORTRAN code consistency checks (definitions of functions, common blocks etc.)
- Interactive visualization & analysis of inconsistencies
- Upgrading from FORTRAN 77 to FORTRAN 90
- Interactive/batch analysis of parallelization possibilities
- Automatic code quality analysis/improvements

PBS Pro - The Portable Batch System

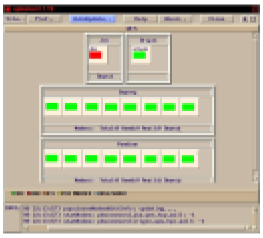
Flexible workload management system

www.pbspro.com

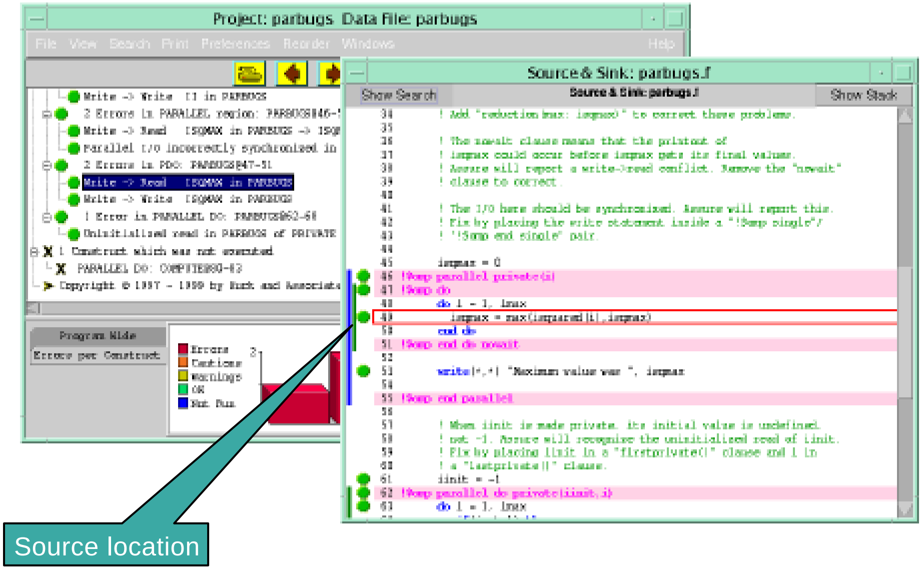


- Unified interface to all computing resources
  - POSIX batch standard, all major UNIXs supported, heterogeneous environment, SMPs & clusters, parallel jobs (MPI & Globus & UNICORE)
  - Single interface handles both interactive and batch processing
  - GUI tools for user and administrator

- Fully configurable scheduler module -- any site policy
  - fair share, load balancing, priorities, back-filling, meta-scheduling
- Sophisticated fault tolerance, accounting, security (ACLs), automatic file staging
- Professional services: commercial support & training



KAP/Pro Toolset - Assure OpenMP Example

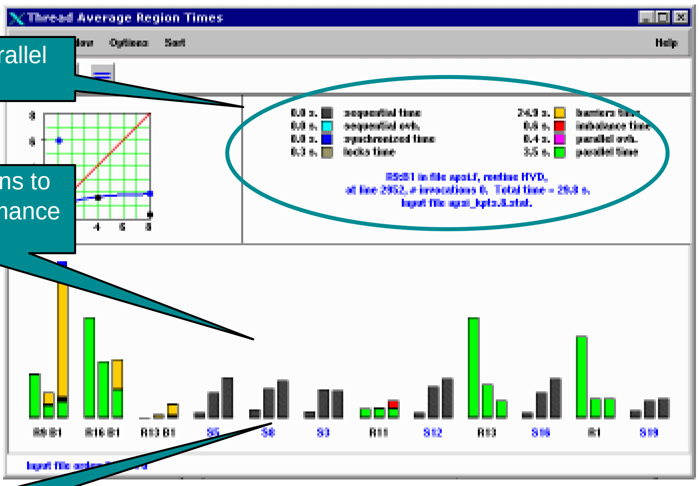


```
Write -> Write  [I] in PARBUGS
2 Errors in PARALLEL region: PARBUGS@46-5
Write -> Read  ISQMAX in PARBUGS -> ISQ
Parallel I/O incorrectly synchronized in
2 Errors in PDO: PARBUGS@47-51
Write -> Read  ISQMAX in PARBUGS
Write -> Write  ISQMAX in PARBUGS
1 Error in PARALLEL DO: PARBUGS@62-68
Uninitialized read in PARBUGS of PRIVATE
1 Construct which was not executed
PARALLEL DO: COMPTIE@80-83
Copyright © 1997 - 1999 by Kuck and Associates
```

```
      ! Add "reduction(max: iqmax)" to correct these problems.

      ! The nowait clause means that the printout of
      ! iqmax could occur before iqmax gets its final value.
      ! Assure will report a write->read conflict. Remove the "nowait"
      ! clause to correct.

      ! The I/O here should be synchronized. Assure will report this.
      ! Fix by placing the write statement inside a "!$omp single"/
      ! "!$omp end single" pair.

      iqmax = 0
!$omp parallel private(i)
!$omp do
      do i = 1, imax
        iqmax = max(iqsquared(i),iqmax)
      end do
!$omp end do nowait

      write(*,*) "Maximum value was ", iqmax

!$omp end parallel

      ! When iinit is made private, its initial value is undefined,
      ! not -1. Assure will recognize the uninitialized read of iinit.
      ! Fix by placing iinit in a "firstprivate()" clause and i in
      ! a "lastprivate()" clause.
      iinit = -1
!$omp parallel do private(iinit, i)
      do i = 1, imax
```

Source location

KAP/Pro Toolset - GuideView OpenMP Example



0.0 s. ▨ sequential time
0.0 s. ▨ sequential ovh.
0.0 s. ▨ synchronized time
0.3 s. ▨ locks time

24.9 s. ▨ barriers time
0.6 s. 🟥 imbalance time
0.4 s. ▨ parallel ovh.
3.5 s. 🟩 parallel time

R9:B1 in file apsi.f, routine HYD,
at line 2952, # invocations 0. Total time = 29.8 s.
Input file apsi_kpts.8.stat.

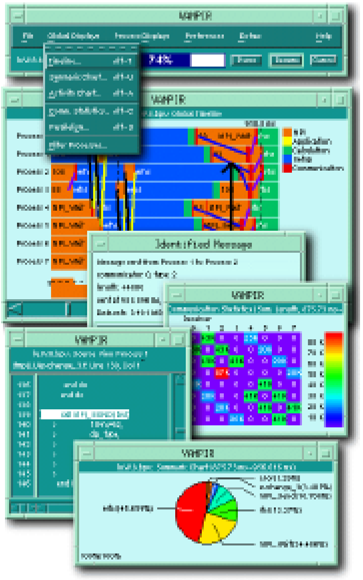
Analyse each parallel region

Sort or filter regions to navigate to performance hotspots

Identify serial regions that hurt scalability pallas

Vampir 2.5

Visualization and
Analysis of
MPI
Programs



Vampir Features

- Offline trace analysis for MPI (and others ...)
- Traces generated by Vampirtrace tool (`ld ... -lVT -lpmpi -lmpi`)
- Convenient user–interface
- Scalability in time and processor–space
- Excellent zooming and filtering
- High–performance graphics
- Display and analysis of MPI and application events:
  - execution of MPI routines
  - point–to–point and collective communication
  - MPI–2 I/O operations
  - execution of application subroutines (optional)
- Easy customization

How to use Vampir

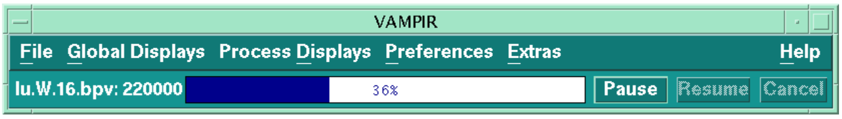
- Step 1: generate instrumented binary
  - re–link your application with Vampirtrace and libmpi.a:

    ```
    f90 -o test test.o -lVT -lpmpi -lmpi
    ```

    OR
  - instrument application, recompile and relink
- Step 2: produce tracefile
  - run the instrumented binary
  - can specify configuration file to filter data
- Step 3: analyze tracefile with Vampir (on workstation)
  - look at global statistics
  - identify interesting application phases
  - investigate those in–depth

Vampir Main Window

Vampir 2.5 main window

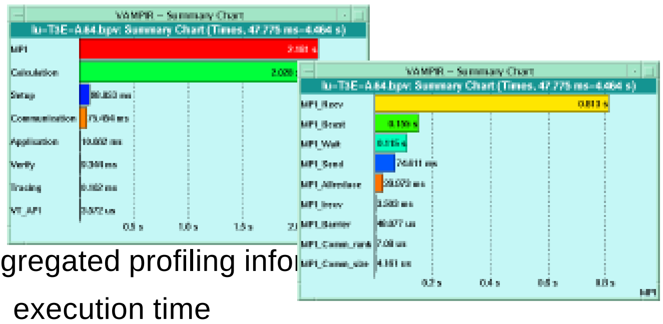


- Tracefile loading can be interrupted at any time
- Tracefile loading can be resumed
- Tracefile can be loaded starting at a specified time offset
- Tracefile can be re–written (re–grouped symbols)

Vampir Displays

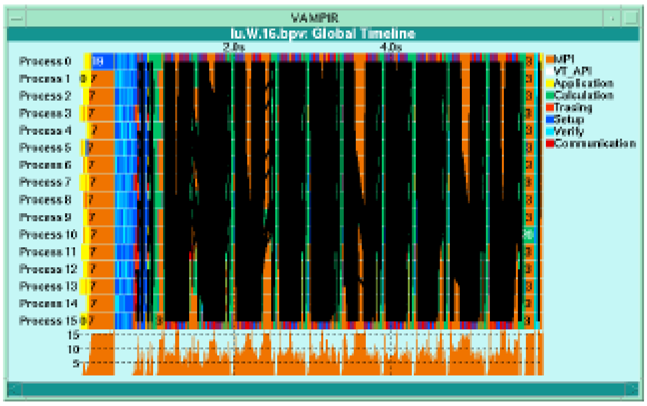
- Global displays show all selected processes
  - Summary Chart: aggregated profiling information
  - Activity Chart: presents per–process profiling information
  - Timeline: detailed application execution over time axis
  - Communication statistics: message statistics for each process pair
  - Global Comm. Statistics: collective operations statistics
  - I/O Statistics: MPI I/O operation statistics
  - Calling Tree: draws global or local dynamic calling trees
- Process displays show a single process per window
  - Activity Chart
  - Timeline
  - Calling Tree

Summary Chart



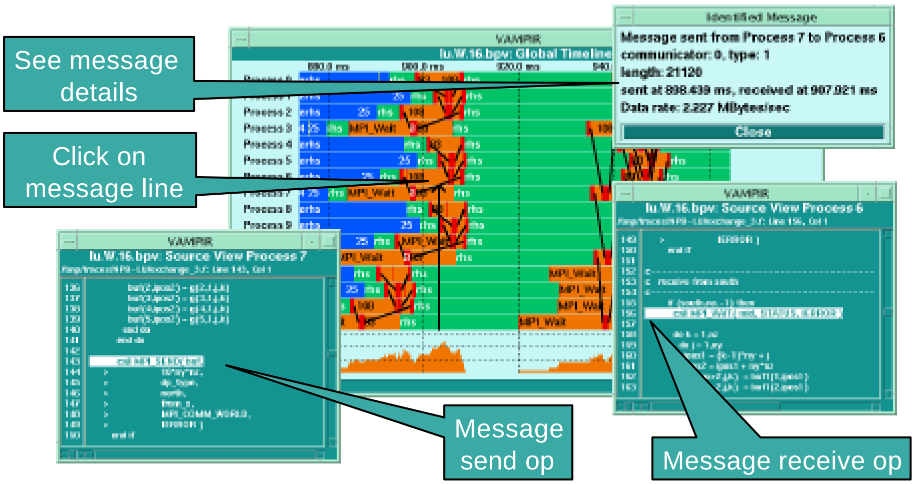
- Aggregated profiling infor
  - execution time
  - number of calls
- Inclusive or exclusive of called routines
- Look at all/any category or all states
- Values can be exported/imported
- Tracefiles can be compared

Timeline Display



- Now displays MPI collective and I/O operations
- To zoom, draw rectangle with the mouse
- Also used to select sub–intervals for statistics

Timeline Display (Message Info)


VAMPIR
lu.W.16.bpv: Global Timeline
Identified Message
Close
VAMPIR
lu.W.16.bpv: Source View Process 7
VAMPIR
lu.W.16.bpv: Source View Process 6
MPI_Wait

Message sent from Process 7 to Process 6
communicator: 0, type: 1
length: 21120
sent at 898.439 ms, received at 907.921 ms
Data rate: 2.227 MBytes/sec

See message details

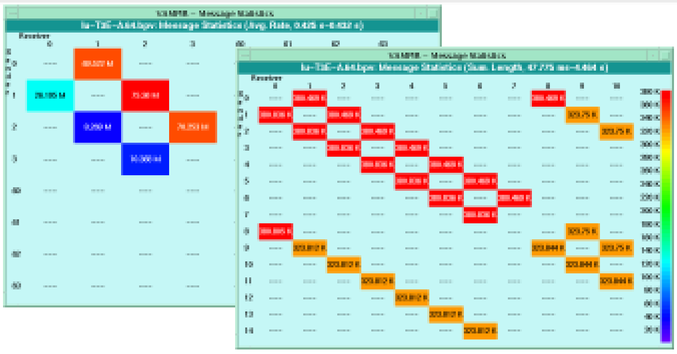
Click on message line

Message send op

Message receive op

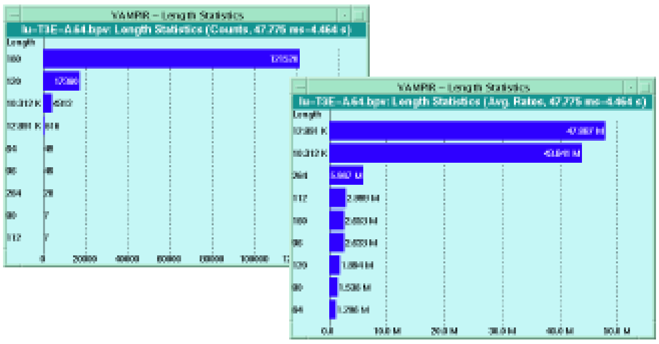
- Source–code references are displayed if recorded by Vampirtrace

Communication Statistics


VAMPIR – Message Statistics
lu-T3E-A64.bpv: Message Statistics (Avg. Rate, 0.425 s–0.432 s)
Receiver
VAMPIR – Message Statistics
lu-T3E-A64.bpv: Message Statistics (Sum. Length, 47.775 ms–4.464 s)
Receiver

- Message statistics for each process pair:
  - Byte and message count
  - min/max/avg message length
  - min/max/avg bandwidth
- Filter for message tags or communicators

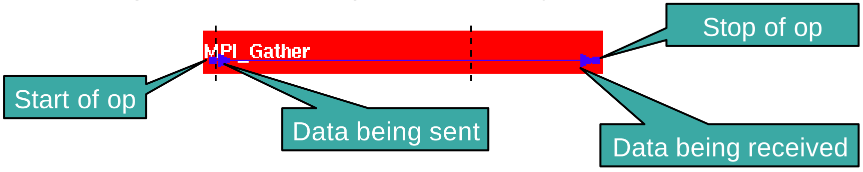
Message Histograms


VAMPIR – Length Statistics
lu-T3E-A64.bpv: Length Statistics (Counts, 47.775 ms–4.464 s)
Length
VAMPIR – Length Statistics
lu-T3E-A64.bpv: Length Statistics (Avg. Rates, 47.775 ms–4.464 s)
Length

- Message statistics by length, tag or communicator
  - Byte and message count
  - min/max/avg bandwidth
- Filter for message tags or communicators

Collective Operations

- For each process: mark operation locally



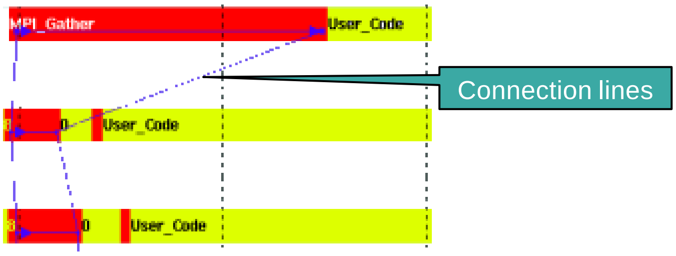
Stop of op

Start of op

Data being sent

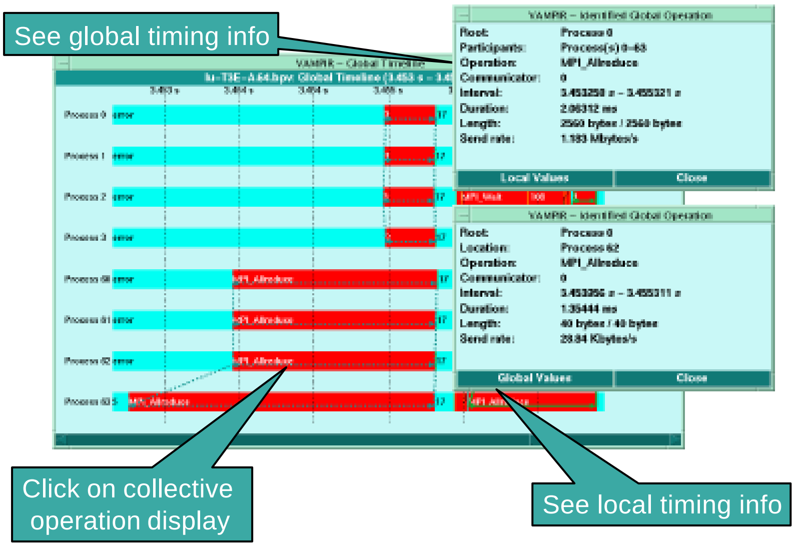
Data being received

- Connect start/stop points by lines



Connection lines

Collective Operations



See global timing info

VAMPIR – Identified Global Operation

Root: Process 0
Participants: Process(s) 0–63
Operation: MPI_Allreduce
Communicator: 0
Interval: 3.453258 s – 3.455321 s
Duration: 2.06312 ms
Length: 2560 bytes / 2560 bytes
Send rate: 1.183 Mbytes/s

VAMPIR – Identified Global Operation

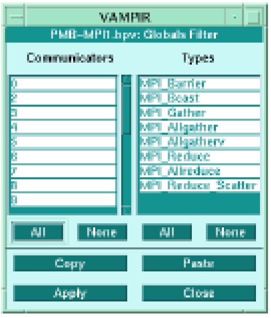
Root: Process 0
Location: Process 62
Operation: MPI_Allreduce
Communicator: 0
Interval: 3.453356 s – 3.455311 s
Duration: 1.35444 ms
Length: 40 bytes / 40 bytes
Send rate: 28.84 Kbytes/s

Click on collective operation display

See local timing info

Collective Operations

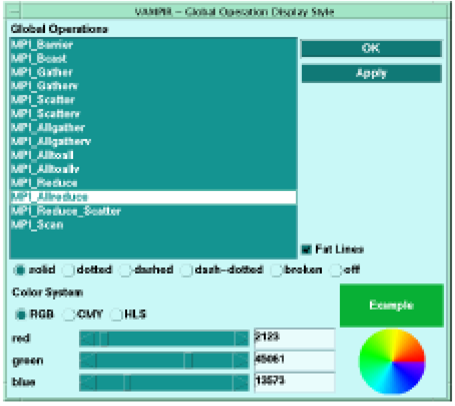
- Collective operations can be filtered



VAMPIR

PMB-MPI1.bpv: Globals Filter

Communicators	Types
0	MPI_Barrier
2	MPI_Bcast
3	MPI_Gather
4	MPI_Allgather
5	MPI_Allgatherv
6	MPI_Reduce
7	MPI_Allreduce
8	MPI_Reduce_Scatter
9	



VAMPIR – Global Operation Display Style

Global Operations

MPI_Barrier
MPI_Bcast
MPI_Gather
MPI_Gatherv
MPI_Scatter
MPI_Scatterv
MPI_Allgather
MPI_Allgatherv
MPI_Alltoall
MPI_Alltoallv
MPI_Reduce
MPI_Allreduce
MPI_Reduce_Scatter
MPI_Scan

☑ Fat Lines

● solid ○ dotted ○ dashed ○ dash-dotted ○ broken ○ off

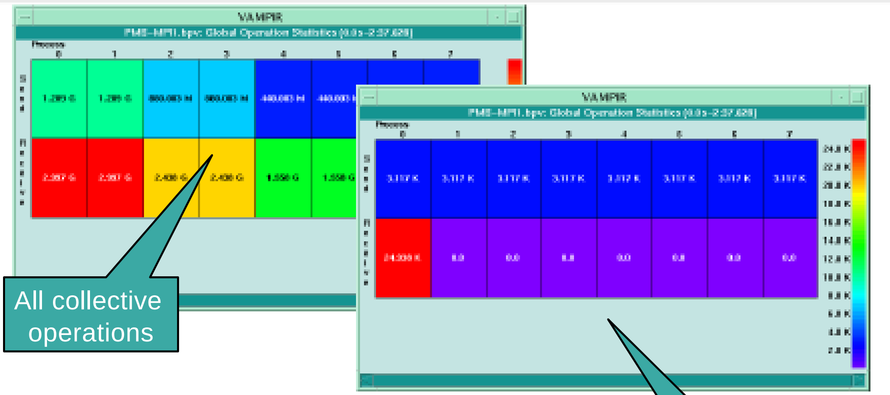
Color System

● RGB ○ CMY ○ HLS

red 2123
green 45061
blue 13573

- The display style can be adapted for each collective operation

Global Communication Statistics



VAMPIR

PMB-MPI1.bpv: Global Operation Statistics [0.0 s–2:37.028]

Process	0	1	2	3	4	5	6	7
Send	1.289 G	1.289 G	880.083 M	880.083 M	440.083 M	440.083 M		
Receive	2.387 G	2.387 G	2.408 G	2.408 G	1.558 G	1.558 G		

VAMPIR

PMB-MPI1.bpv: Global Operation Statistics [0.0 s–2:37.028]

Process	0	1	2	3	4	5	6	7
Send	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K	3.117 K
Receive	24.936 K	0.0	0.0	0.0	0.0	0.0	0.0	0.0

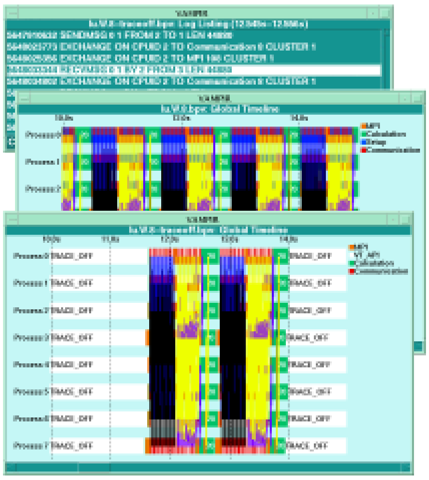
All collective operations

MPI_Gather only

- Statistics for collective operations:
  - operation counts, Bytes sent/received
  - transmission rates
- Filter for collective operation

Vampirtrace

Tracing of
MPI and
Application
Events



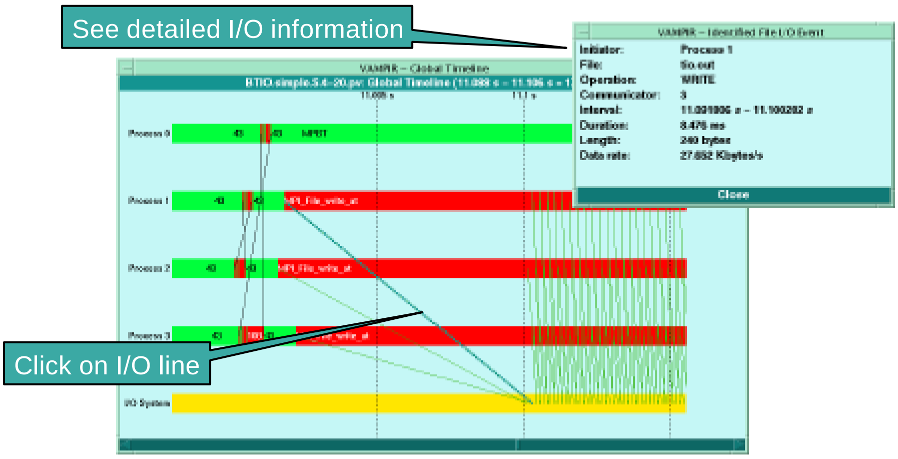
lu.W.8-traceoff.bpv: Log Listing (12.540s - 12.556s)

```
5647910632 SENDMSG 0 1 FROM 2 TO 1 LEN 44800
5648023773 EXCHANGE ON CPUID 2 TO Communication 0 CLUSTER 1
5648025056 EXCHANGE ON CPUID 2 TO MPI 186 CLUSTER 1
5648033348 RECVMSG 0 1 BY 2 FROM 3 LEN 44800
5648034802 EXCHANGE ON CPUID 2 TO Communication 0 CLUSTER 1
```

Vampirtrace

- Current: Vampirtrace 2.0
- Significant new features:
  - records collective communication
  - enhanced filter functions
  - extended API
  - records source–code information (selected platforms)
  - support for shmem (Cray T3E)
  - records MPI–2 I/O operations
- Available for all major MPI platforms

MPI–I/O Operations



See detailed I/O information

VAMPIR – Identified File I/O Event

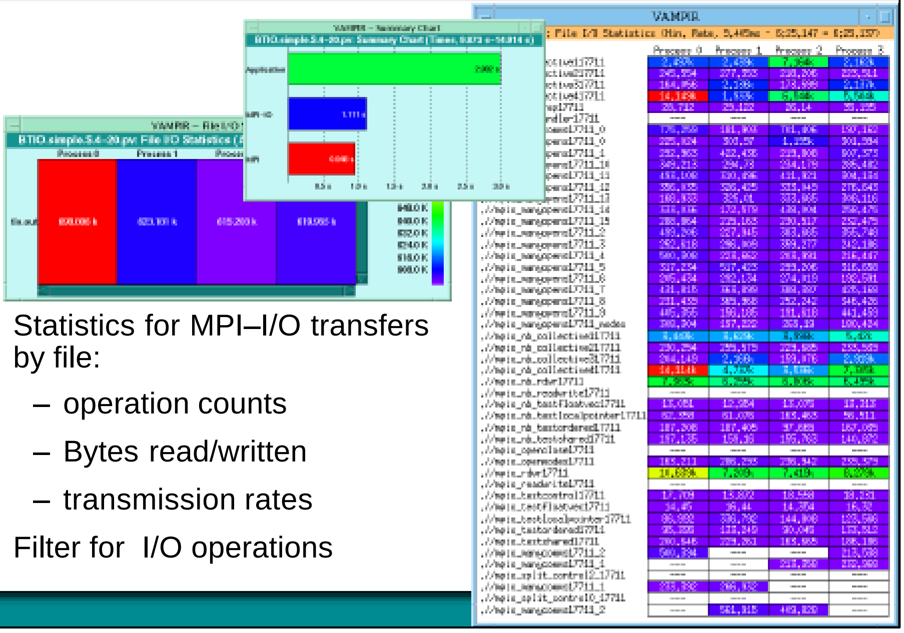
Initiator: Process 1
File: tio.out
Operation: WRITE
Communicator: 3
Interval: 11.091806 s – 11.100282 s
Duration: 8.476 ms
Length: 240 bytes
Data rate: 27.652 Kbytes/s

Click on I/O line

- I/O transfers are shown as lines

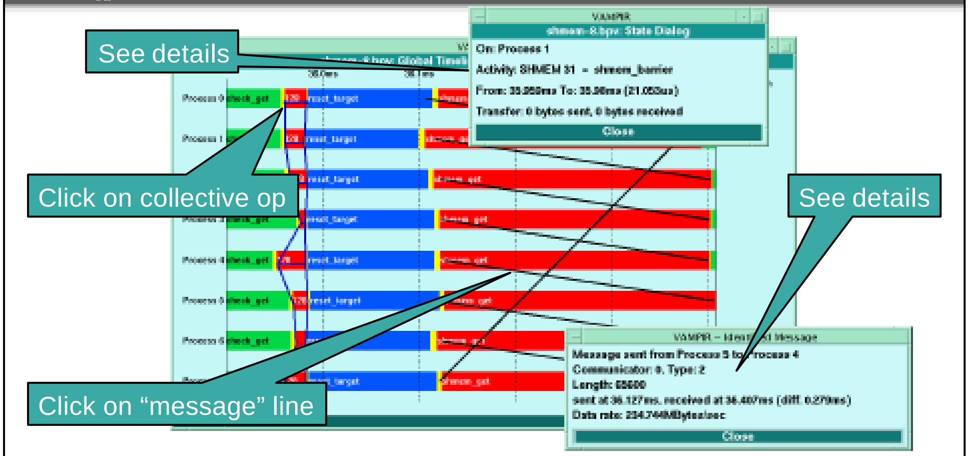
© Pallas GmbH

MPI–I/O Statistics



- Statistics for MPI–I/O transfers by file:
  - – operation counts
  - – Bytes read/written
  - – transmission rates
- Filter for I/O operations

shmem Operations



See details

Click on collective op

See details

Click on “message” line

VAMPIR

shmem-8.bpv: State Dialog

On: Process 1

Activity: SHMEM 31 - shmem_barrier

From: 35.959ms To: 35.98ms (21.053us)

Transfer: 0 bytes sent, 0 bytes received

VAMPIR – Identified Message

Message sent from Process 5 to Process 4

Communicator: 0, Type: 2

Length: 65600

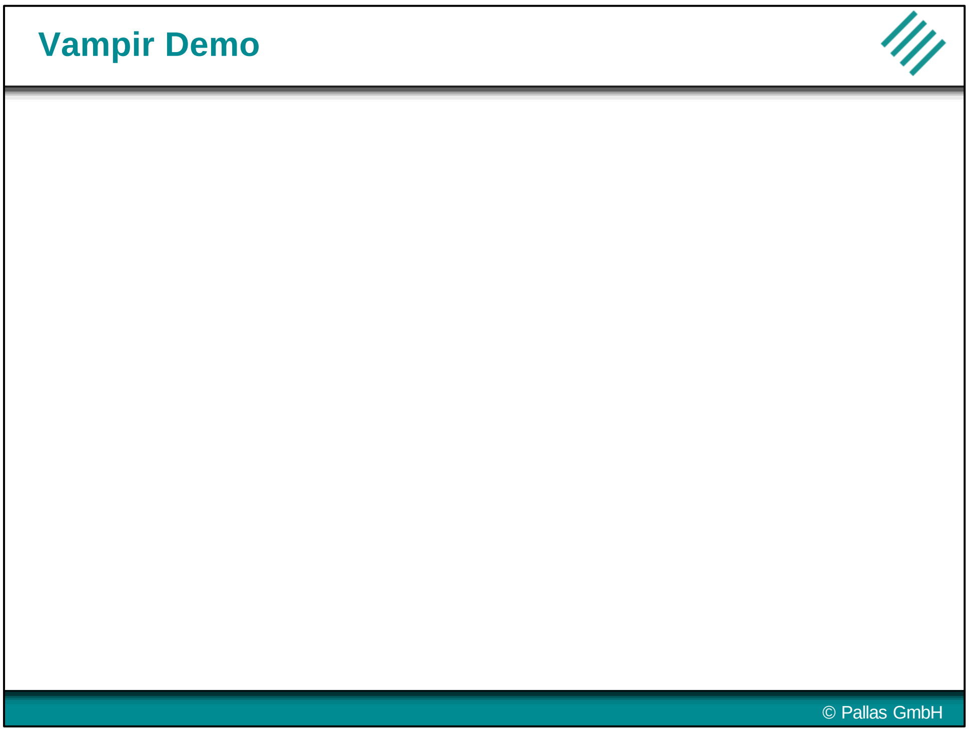
sent at 36.127ms, received at 36.407ms (diff. 0.279ms)

Data rate: 234.744MBytes/sec

- Display one–sided transfers as messages
- Display shmem global operations

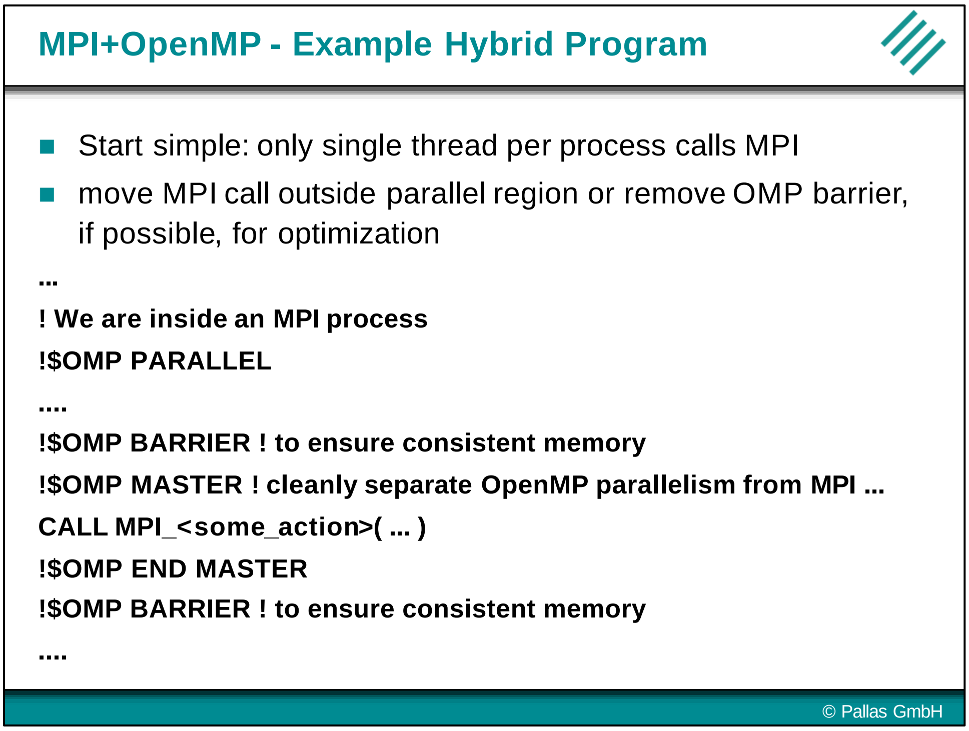
Vampir Track Record

- Reference customers: ARL, ARSC, CEWES, LANL, LLNL, MHPCC, NASA, NCSC, NERSC, SDSC, Cornell TC, Oregon Univ., CEA, DWD, ECMWF, GMD, HLRS, LRZ, PC^2, RUKA, ...
- URLs:
  - www.llnl.gov/sccd/lc/DEG/vampir/vampir.html
  - www.uni-karlsruhe.de/~Vampir
  - www.lrz-muenchen.de/services/software/parallel/vampir
  - www.hlrs.de/structure/support/parallel_computing/tools/performance/vampir.html



Vampir Demo

© Pallas GmbH



MPI+OpenMP - Example Hybrid Program

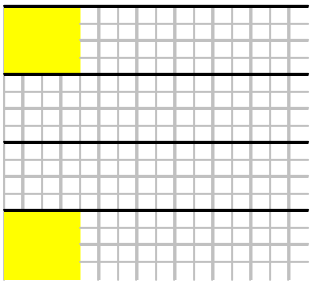
- Start simple: only single thread per process calls MPI
- move MPI call outside parallel region or remove OMP barrier, if possible, for optimization

```
...
! We are inside an MPI process
!$OMP PARALLEL
....
!$OMP BARRIER ! to ensure consistent memory
!$OMP MASTER ! cleanly separate OpenMP parallelism from MPI ...
CALL MPI_<some_action>( ... )
!$OMP END MASTER
!$OMP BARRIER ! to ensure consistent memory
....
```

© Pallas GmbH

MPI+OpenMP - Example Hybrid Program

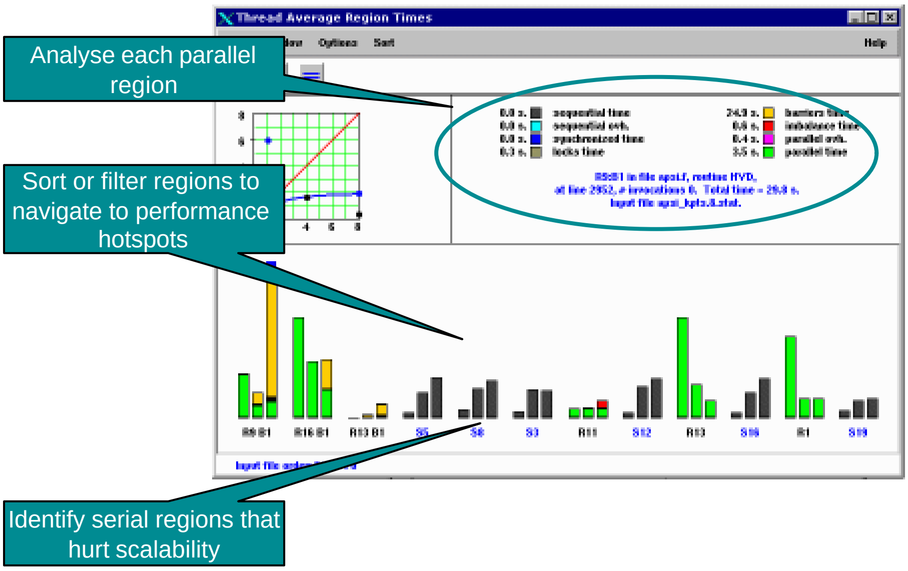
- Parallel Poisson solver in MPI and OpenMP
- certain sections of the grid with more work load
- MPI split not easy, except 2x1 horizontal
- 4 MPI processes run not balanced



MPI+OpenMP - how to measure quality ?

- Judge quality of parallelization by means of tools:
  - Vampir, GuideView --> performance
  - Assure --> correctness
- Yes, the tools work for hybrid approaches. When running, e.g., a 2 by 2 hybrid model (2 MPI processes, 2 threads per process), one gets
  - one Vampir tracefile and,
  - for each (!) MPI process, an own GuideView profile. GuideView can combine these into one single graph. (--> setenv KMP_STATS_FILE guide_%I)

KAP/Pro Toolset - GuideView Example

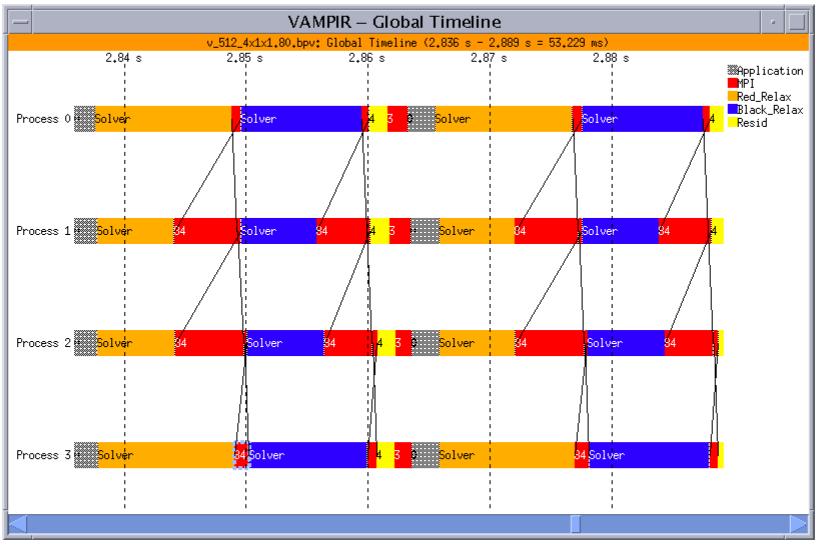


Analyse each parallel region

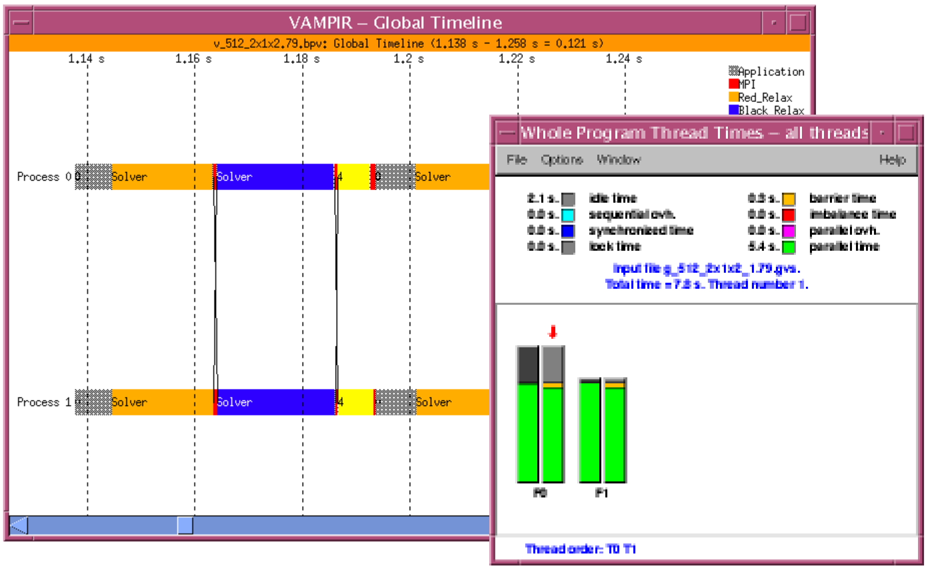
Sort or filter regions to navigate to performance hotspots

Identify serial regions that hurt scalability

MPI+OpenMP Example - 4 MPI x 1 OpenMP



MPI+OpenMP Example - 2 MPI x 2 OpenMP

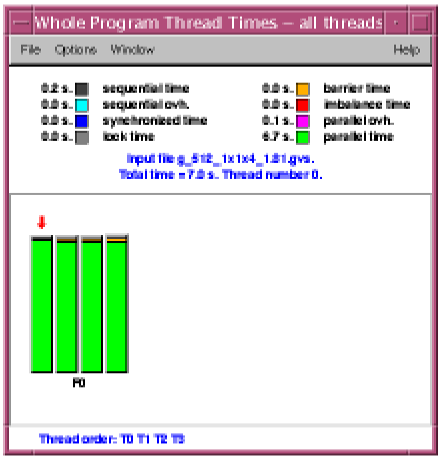


v_512_2x1x2.79.bpv: Global Timeline (1.138 s - 1.258 s = 0.121 s)

```
2.1 s. ■ idle time               0.3 s. 🟧 barrier time
0.0 s. ■ sequential ovh.         0.0 s. 🟥 imbalance time
0.0 s. 🟦 synchronized time      0.0 s. ■ parallel ovh.
0.0 s. ■ lock time               5.4 s. 🟩 parallel time
          Input file g_512_2x1x2_1.79.gvs.
          Total time = 7.8 s. Thread number 1.
```

Thread order: T0 T1

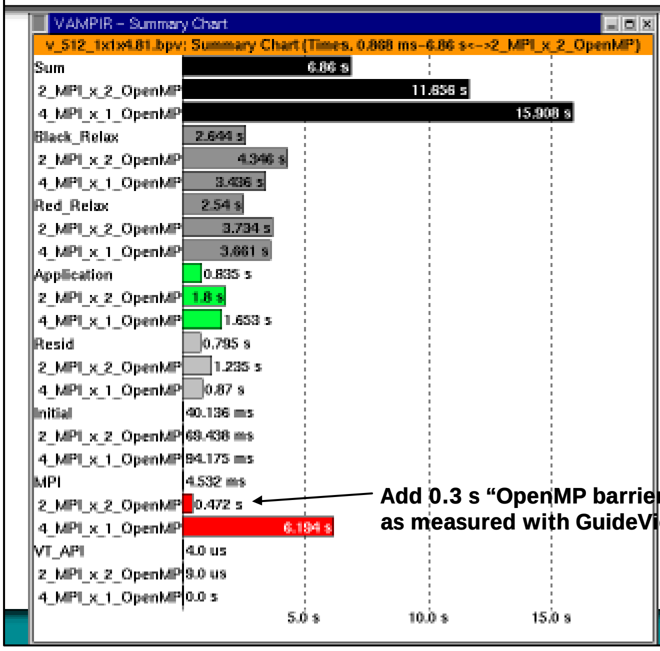
MPI+OpenMP Example - 1 MPI x 4 OpenMP



```
0.2 s. ■ sequential time         0.0 s. 🟧 barrier time
0.0 s. ■ sequential ovh.         0.0 s. 🟥 imbalance time
0.0 s. 🟦 synchronized time      0.1 s. ■ parallel ovh.
0.0 s. ■ lock time               6.7 s. 🟩 parallel time
          Input file g_512_1x1x4_1.81.gvs.
          Total time = 7.0 s. Thread number 0.
```

Thread order: T0 T1 T2 T3

MPI+OpenMP Example - Vampir compares the runs

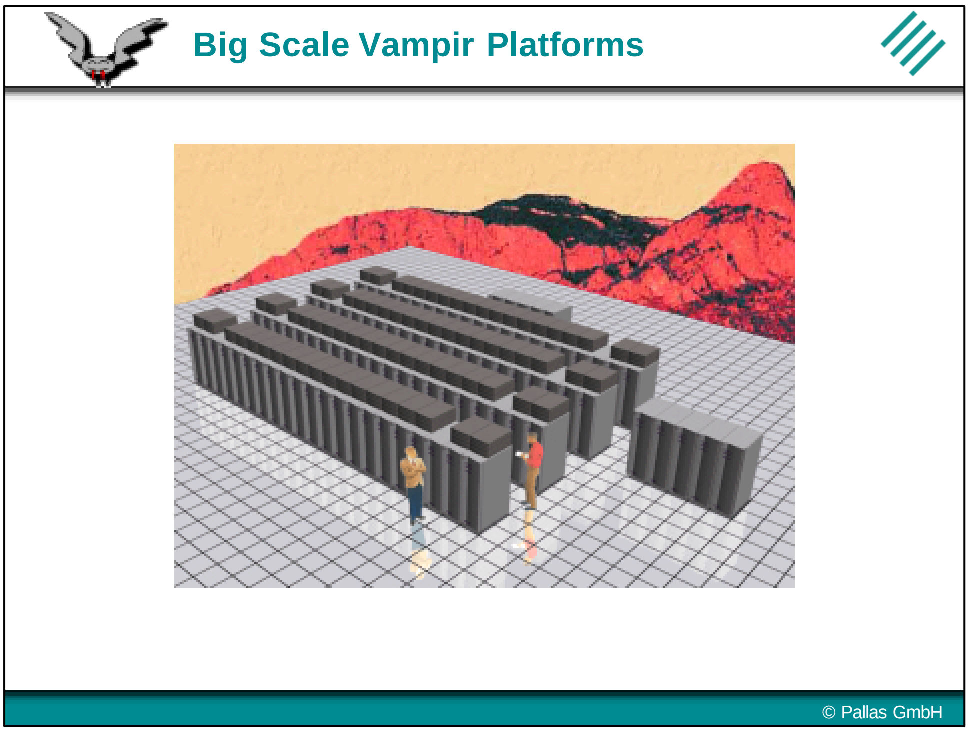


Add 0.3 s “OpenMP barrier -time“
as measured with GuideView

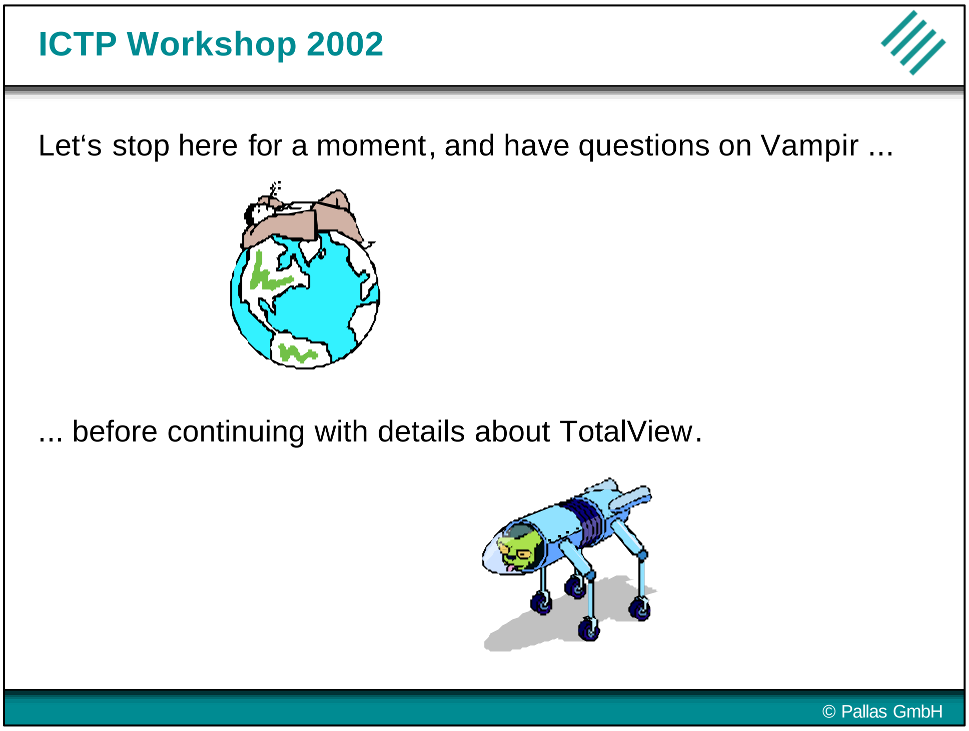
pallas

Future plans - Vampir

- Towards automatic performance analysis
  - improve user guidance in Vampir
  - add “assistant” module for inexperienced users
- Support for clustered shared–memory systems
  - support shared–memory programming models (threads, OpenMP)
  - expose cluster structure
  - aggregate information on SMP nodes
- Support for (very) large systems
  - new structured tracefile format
  - fine–grain interactive control over tracing
  - scalable displays
  - new Vampir structure (can exploit parallelism)



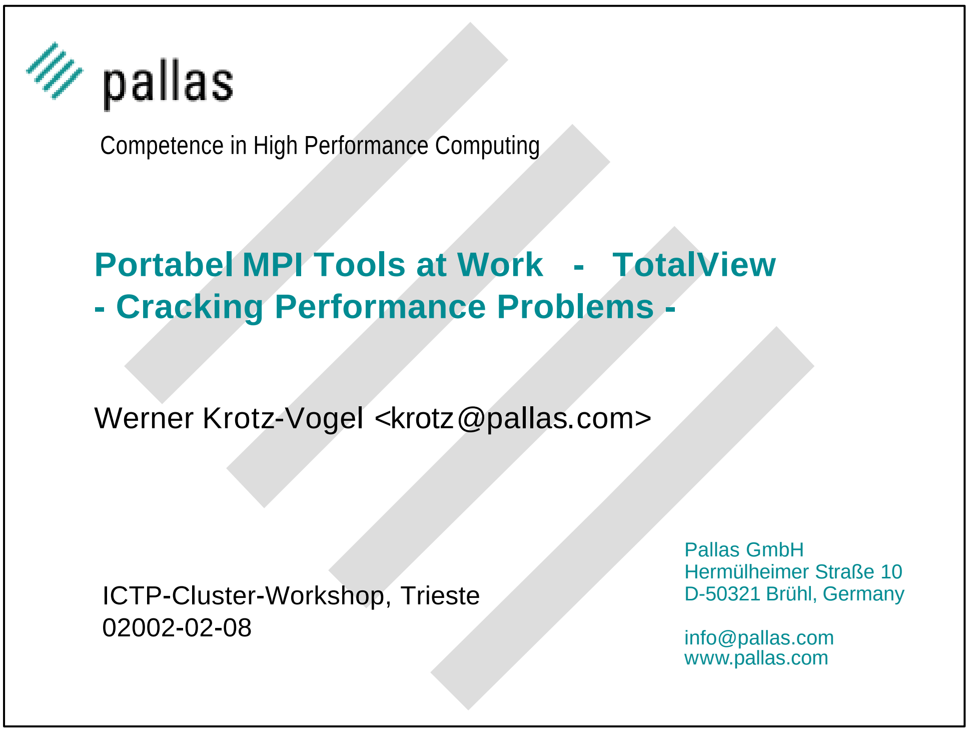
Big Scale Vampir Platforms



ICTP Workshop 2002

Let‘s stop here for a moment, and have questions on Vampir ...

... before continuing with details about TotalView.



pallas

Competence in High Performance Computing

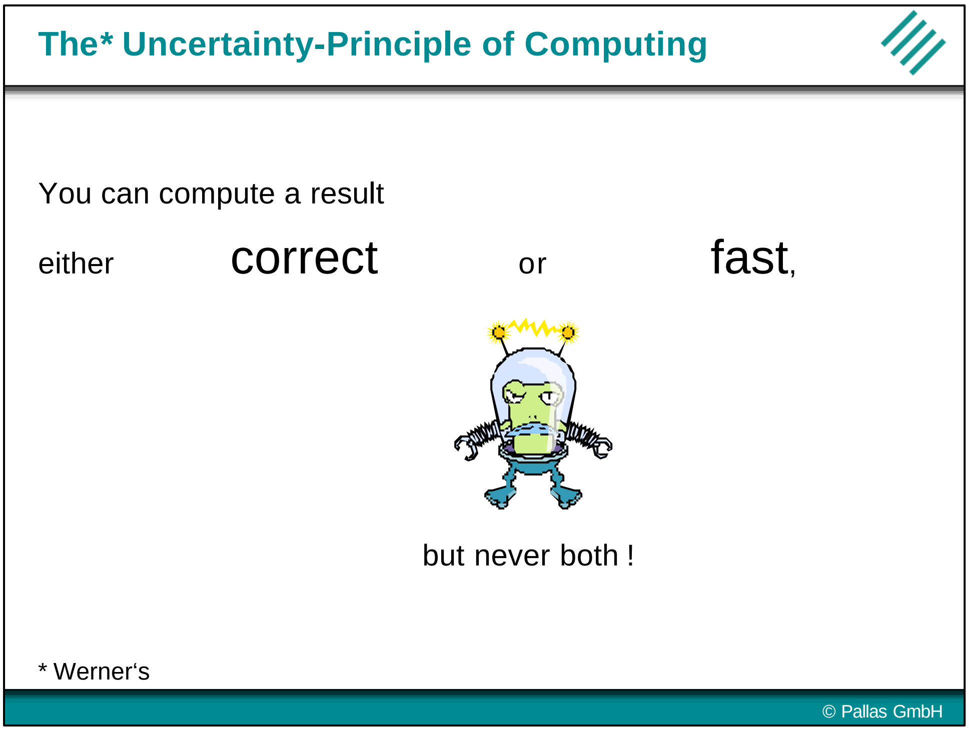
Portabel MPI Tools at Work - TotalView
- Cracking Performance Problems -

Werner Krotz-Vogel <krotz@pallas.com>

ICTP-Cluster-Workshop, Trieste
02002-02-08

Pallas GmbH
Hermülheimer Straße 10
D-50321 Brühl, Germany

info@pallas.com
www.pallas.com



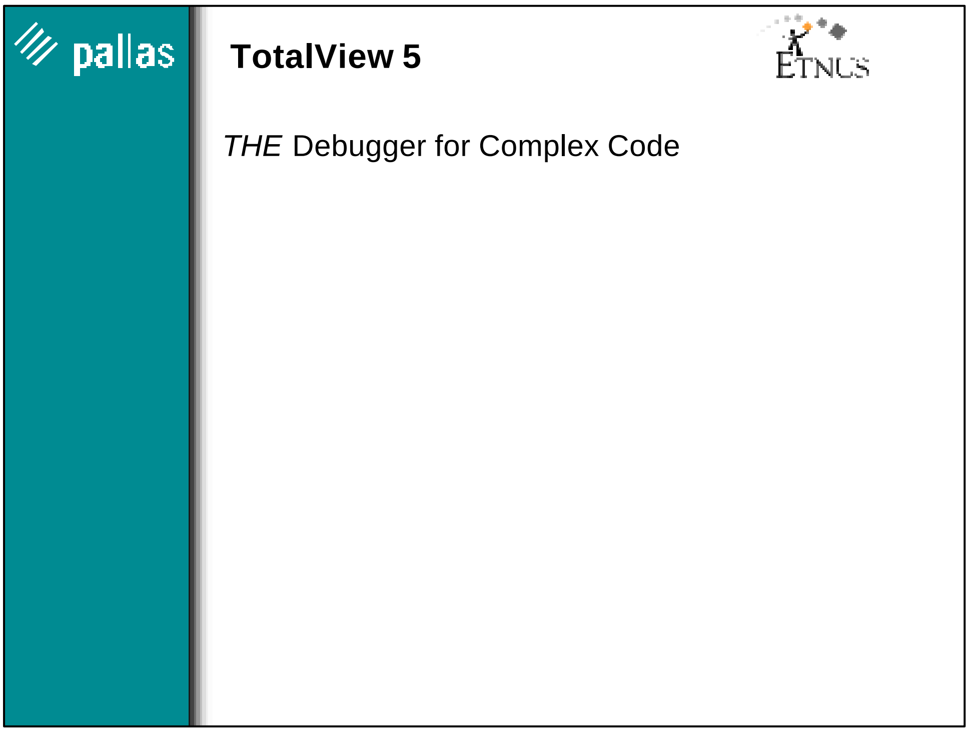
The* Uncertainty-Principle of Computing

You can compute a result

either correct or fast,

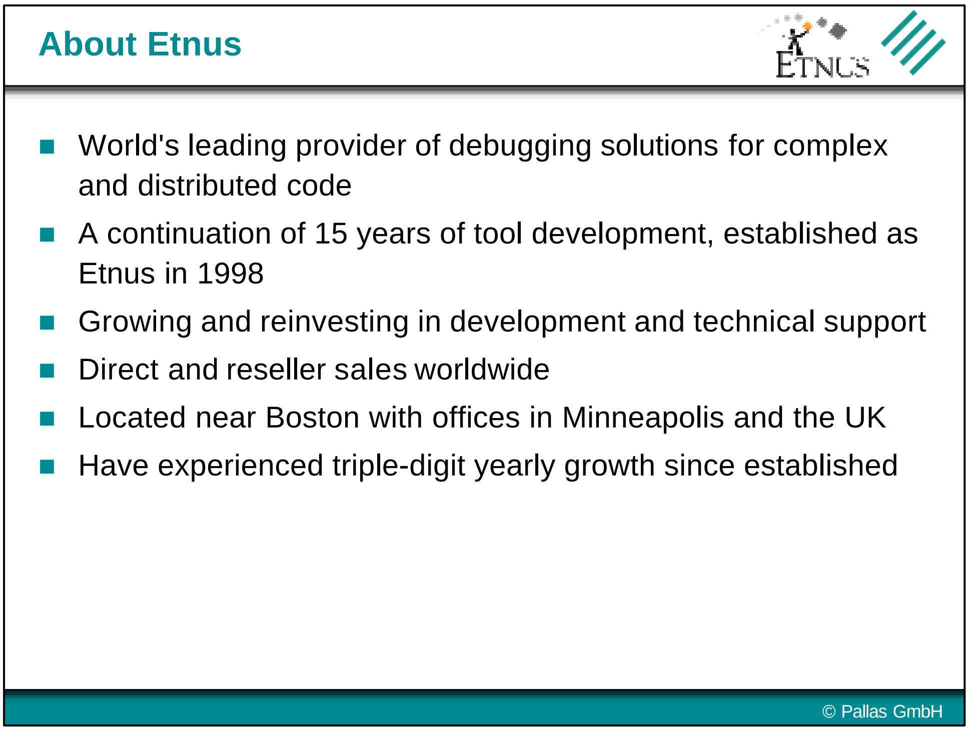
but never both !

* Werner's



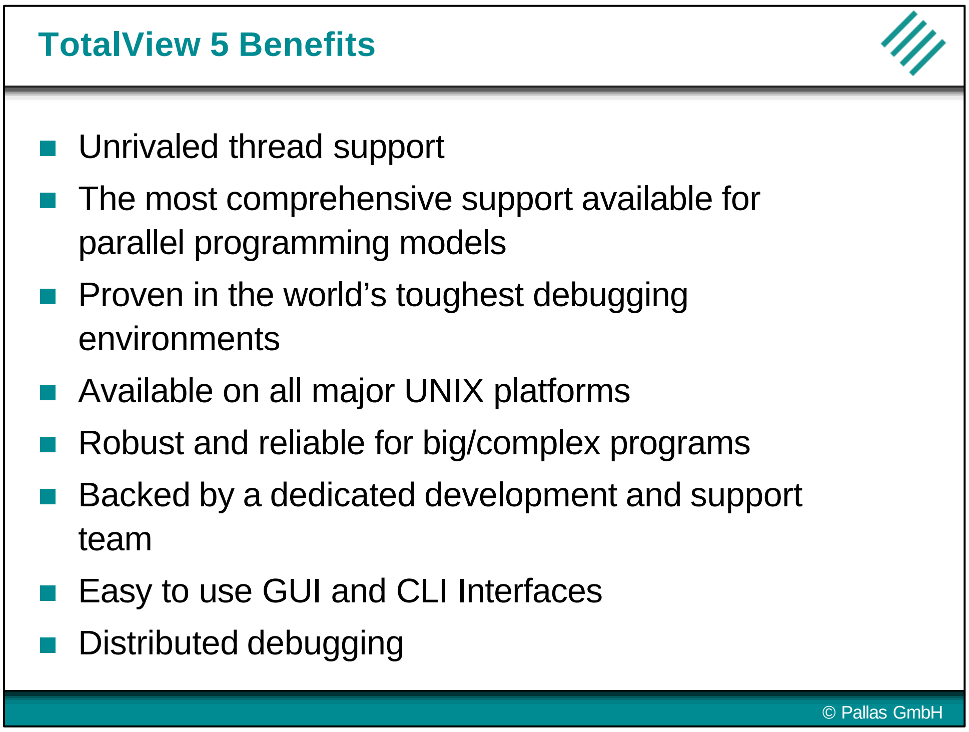
TotalView 5

THE Debugger for Complex Code



About Etnus

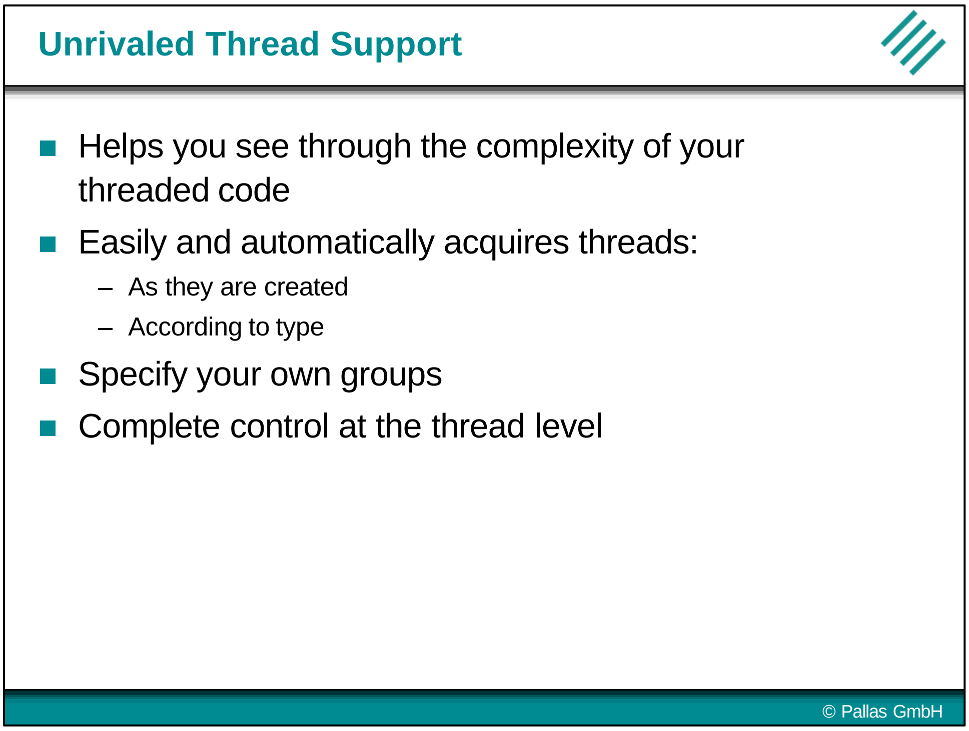
- World's leading provider of debugging solutions for complex and distributed code
- A continuation of 15 years of tool development, established as Etnus in 1998
- Growing and reinvesting in development and technical support
- Direct and reseller sales worldwide
- Located near Boston with offices in Minneapolis and the UK
- Have experienced triple-digit yearly growth since established



TotalView 5 Benefits

- Unrivaled thread support
- The most comprehensive support available for parallel programming models
- Proven in the world’s toughest debugging environments
- Available on all major UNIX platforms
- Robust and reliable for big/complex programs
- Backed by a dedicated development and support team
- Easy to use GUI and CLI Interfaces
- Distributed debugging

© Pallas GmbH



Unrivaled Thread Support

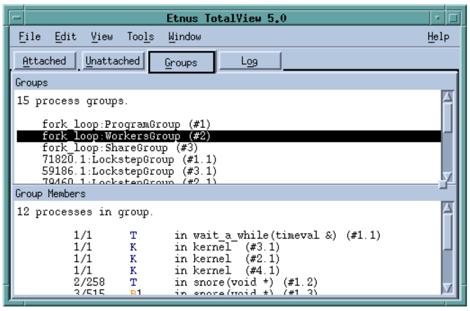
- Helps you see through the complexity of your threaded code
- Easily and automatically acquires threads:
  - As they are created
  - According to type
- Specify your own groups
- Complete control at the thread level

© Pallas GmbH

Complete Control at the Thread Level

- Control individual threads and set breakpoints at the thread level*
- Control groups of threads
- Dig into critical sections of your code, mutexes and queues
- Dynamically manage thread groups and add newly spawned threads to groups
- Switch between threads, view objects, and control execution

*not available on all platforms



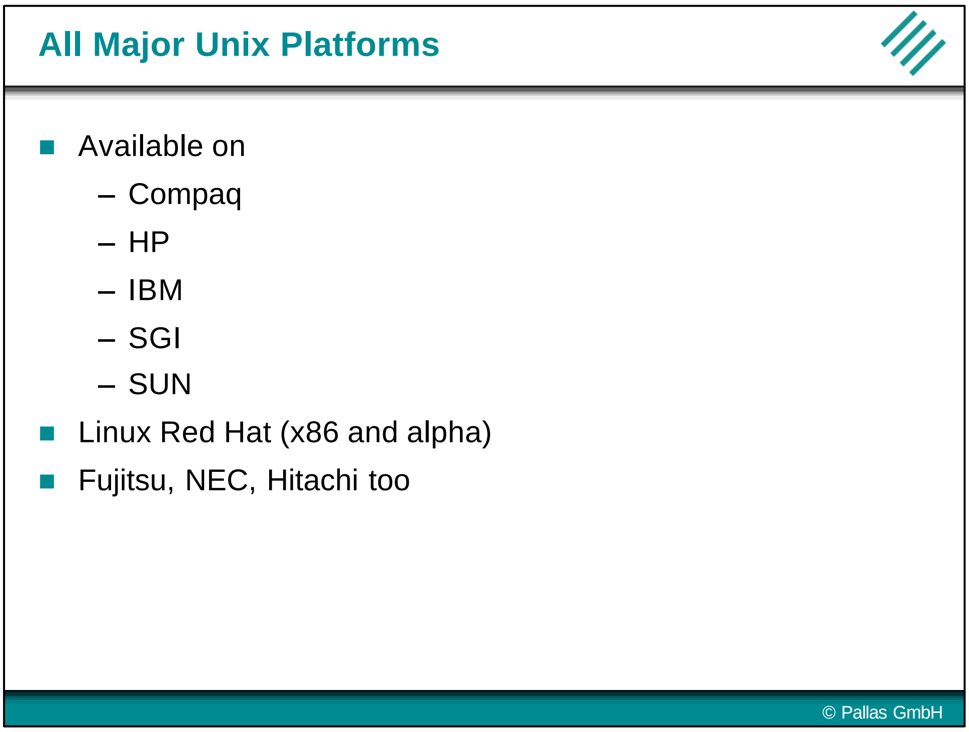
```
15 process groups.

    fork_loop:ProgramGroup (#1)
    fork_loop:WorkersGroup (#2)
    fork_loop:ShareGroup (#3)
    71820.1:LockstepGroup (#1.1)
    59186.1:LockstepGroup (#3.1)
    79460.1:LockstepGroup (#2.1)
```

```
12 processes in group.

         1/1       T      in wait_a_while(timeval &) (#1.1)
         1/1       K      in kernel  (#3.1)
         1/1       K      in kernel  (#2.1)
         1/1       K      in kernel  (#4.1)
         2/258     T      in snore(void *) (#1.2)
         3/515     B1     in snore(void *) (#1.3)
```

Dynamic Groups

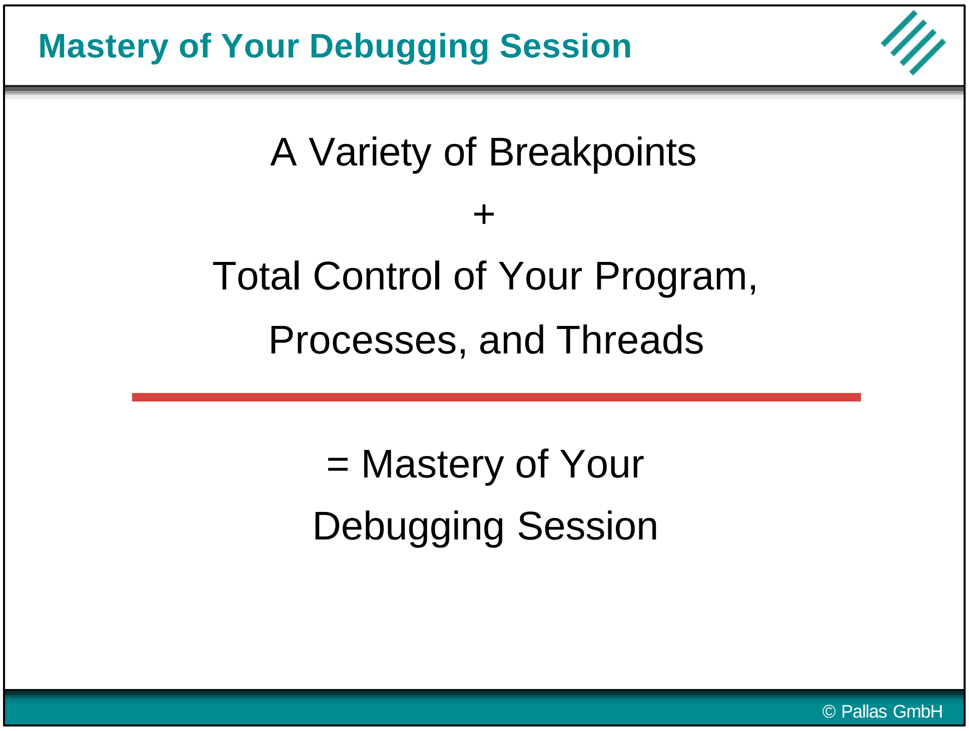
Group Processes or Threads & Control at the Group Level

- Called Process/Thread Sets (P/T sets)
- Create a P/T set through the CLI
  - `dgroups -new thread –g mygrp 2.3`
- Add processes or threads to a group
  - `dgroups -add –g mygrp 3.3`
- Set breakpoints on that group or control it in other ways



All Major Unix Platforms

- Available on
  - Compaq
  - HP
  - IBM
  - SGI
  - SUN
- Linux Red Hat (x86 and alpha)
- Fujitsu, NEC, Hitachi too



Mastery of Your Debugging Session

A Variety of Breakpoints

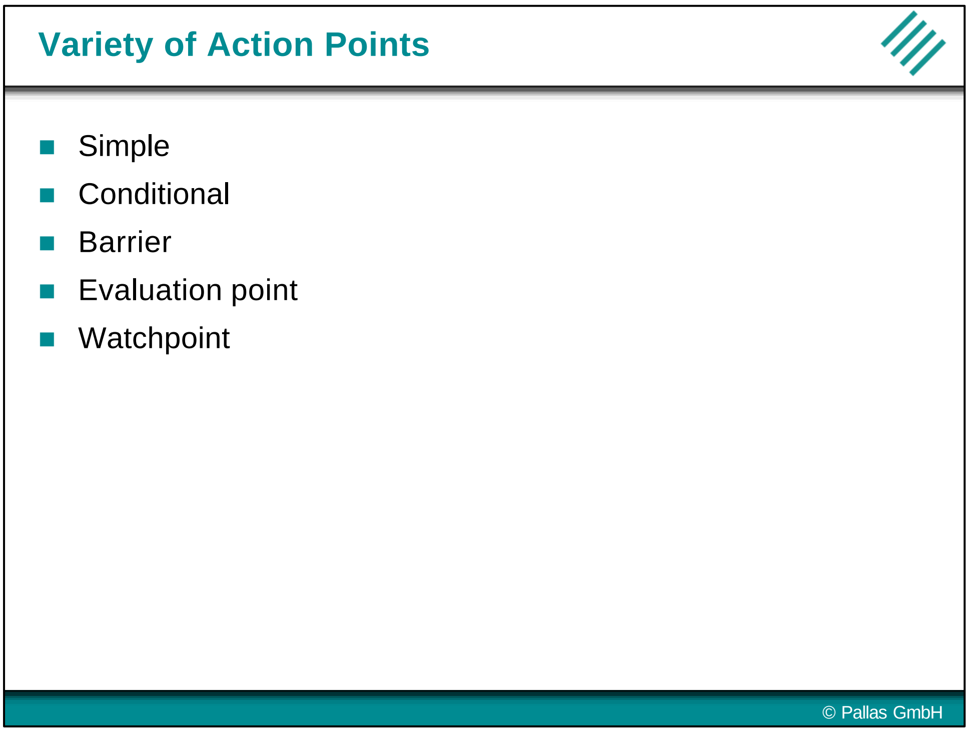
+

Total Control of Your Program,

Processes, and Threads

= Mastery of Your

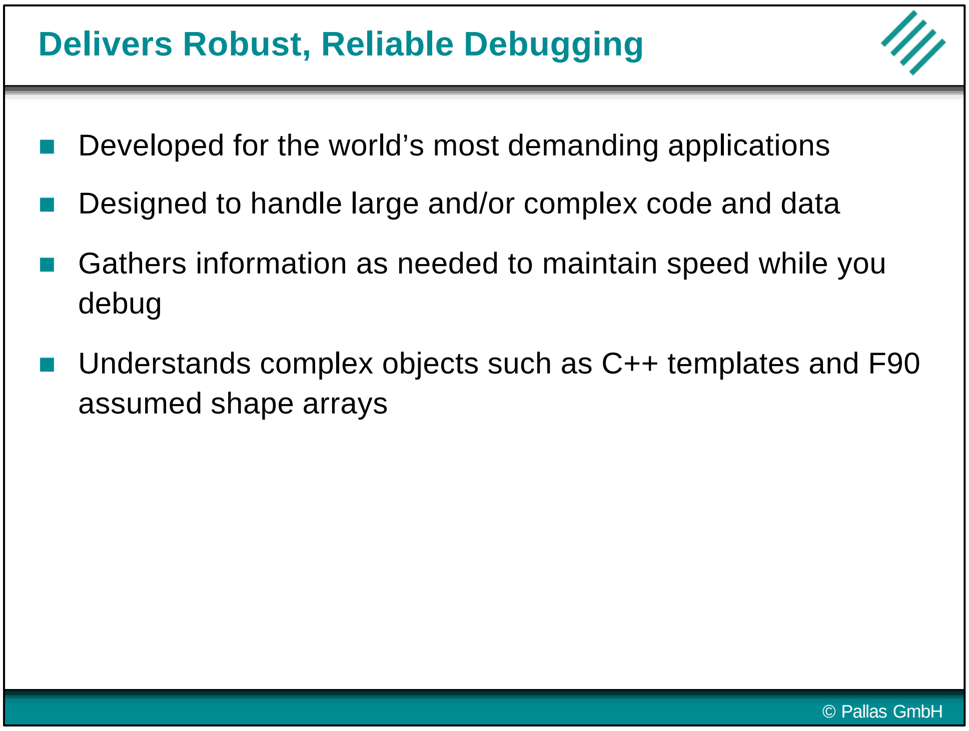
Debugging Session



Variety of Action Points

- Simple
- Conditional
- Barrier
- Evaluation point
- Watchpoint

© Pallas GmbH



Delivers Robust, Reliable Debugging

- Developed for the world’s most demanding applications
- Designed to handle large and/or complex code and data
- Gathers information as needed to maintain speed while you debug
- Understands complex objects such as C++ templates and F90 assumed shape arrays

© Pallas GmbH

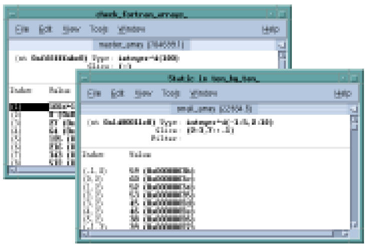
Powerful Data Analysis Features

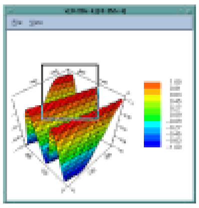
- *Erroneous data can cause bugs*
- *Searching big data can be difficult or impossible*

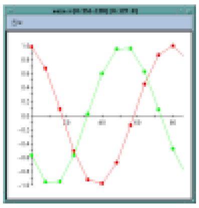
- TotalView lets you explore your data:
  - Filter out data you don’t need
  - Find values out of range
  - Set data watchpoints on memory locations
  - View statistics on arrays
  - Slice arrays to view portions of interest
  - Sort arrays

Explore Your Data

- *Dive* through complex structures
- Edit or cast values on the spot
- Slice, filter, sort arrays
- Plot array data
- View data objects across processes (laminated data)

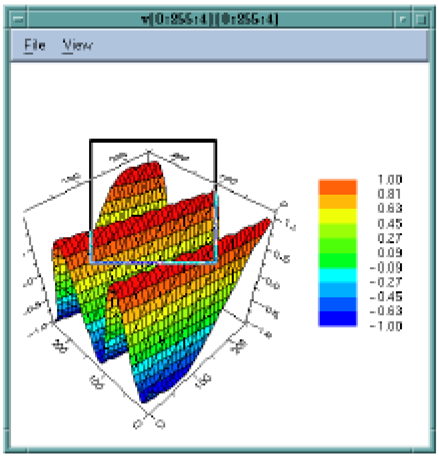






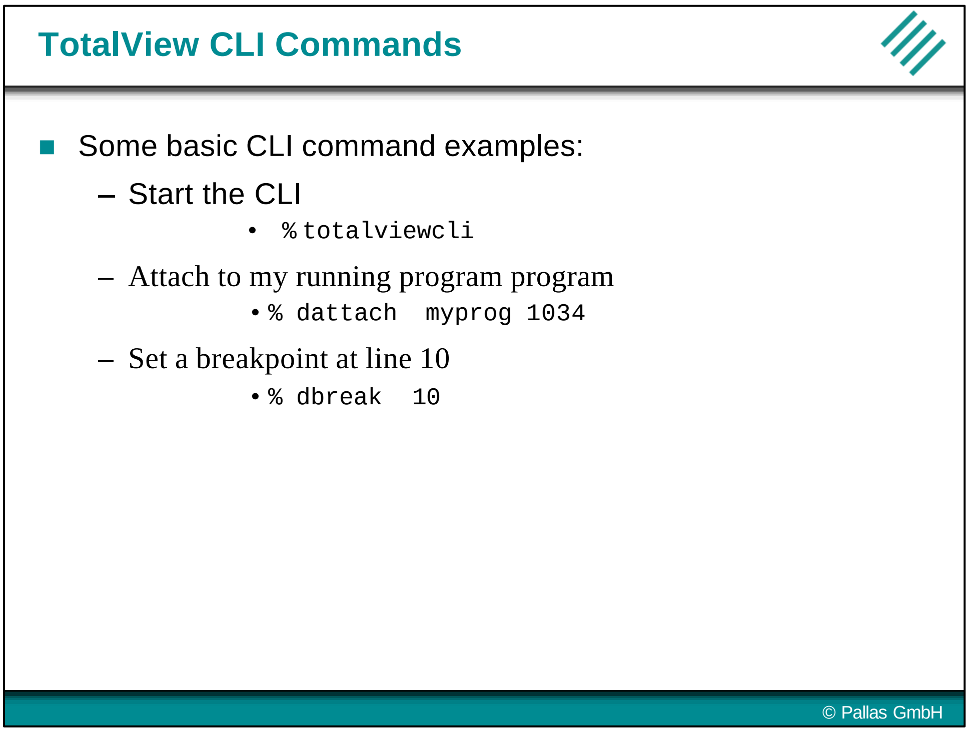
TotalView's Data Visualizer Reveals Bugs

- The visualizer illuminates patterns, anomalies, or outliers in your data



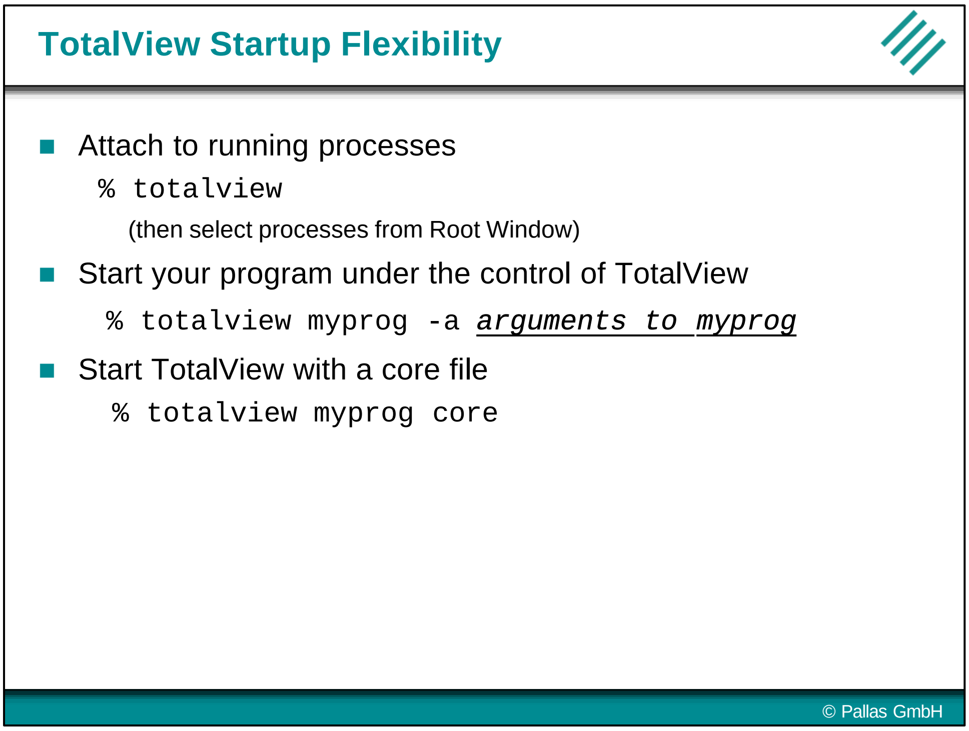
TotalView Interfaces Help You Find Bugs Quickly

- Easy to use GUI and CLI interfaces
- Scripting languages for powerful debugging macros and scripts (through CLI)
- Accommodates large scale programs with hierarchical presentation in GUI… *you* decide how much information you want displayed
- Famous "Dive" feature lets you explore your program, each click giving you more details
- Pre-program CLI input to create completely non-interactive debugging scripts



TotalView CLI Commands

- Some basic CLI command examples:
  - Start the CLI
    - `% totalviewcli`
  - Attach to my running program program
    - `% dattach  myprog 1034`
  - Set a breakpoint at line 10
    - `% dbreak  10`



TotalView Startup Flexibility

- Attach to running processes

```
% totalview
```

(then select processes from Root Window)

- Start your program under the control of TotalView

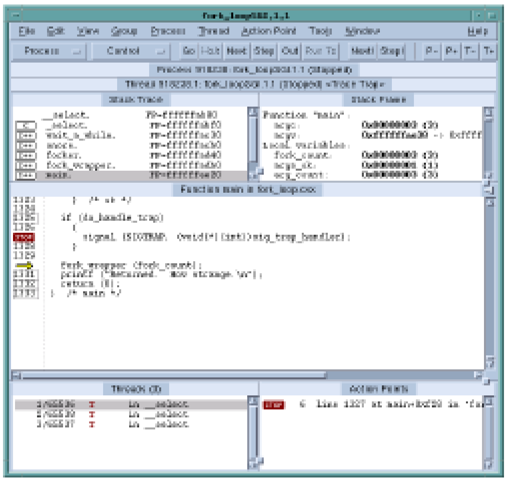
```
% totalview myprog -a arguments to myprog
```

- Start TotalView with a core file

```
% totalview myprog core
```

TotalView 5 Graphical Interface

- Feature-rich menus
- Toolbar & keyboard control
- *Dive* on variables or functions to drill deeper (right mouse click or double left click)


fork_loopSGI.1.1
File Edit View Group Process Thread Action Point Tools Window Help
Process Control Go Halt Next Step Out Run To Next Step P- P+ T- T+
Stack Trace
Stack Frame
Threads (3)
Action Points

Process 918238: fork_loopSGI.1.1 (Stopped)

Thread 918238.1: fork_loopSGI.1.1 (Stopped) <Trace Trap>

```
__select.        FP=ffffffab80
_select.         FP=ffffffabf0
wait_a_while.    FP=ffffffac30
snore.           FP=ffffffac30
forker.          FP=ffffffad40
fork_wrapper.    FP=ffffffad90
main.            FP=ffffffae20
```

```
Function "main":
  argc:            0x00000003 (3)
  argv:            0xffffffae30 -> 0xffff
Local variables:
  fork_count:      0x00000003 (3)
  args_ok:         0x00000001 (1)
  arg_count:       0x00000003 (3)
```

Function main in fork_loop.cxx

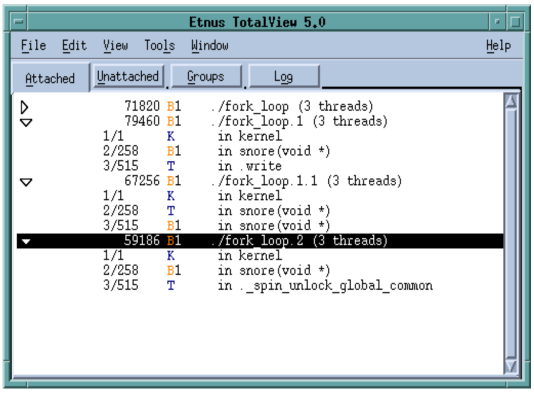
```
        } /* if */

    if (do_handle_trap)
      {
        signal (SIGTRAP, (void(*)(int))sig_trap_handler);
      }

    fork_wrapper (fork_count);
    printf ("Returned.  Now strange.\n");
    return (0);
} /* main */
```

```
1/65536  T    in __select
2/65538  T    in __select
3/65537  T    in __select
```

```
STOP  6  line 1327 at main+0x250 in "fa
```

Root Window

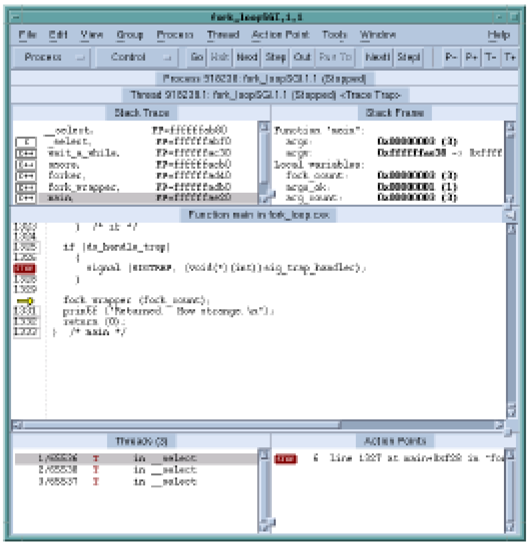
- Gives you hierarchical view of your program
- Combines previously separate windows into tabbed pages


Etnus TotalView 5.0
File Edit View Tools Window Help
Attached Unattached Groups Log

```
▷          71820 B1    ./fork_loop (3 threads)
▽          79460 B1    ./fork_loop.1 (3 threads)
        1/1      K      in kernel
        2/258    B1     in snore(void *)
        3/515    T      in .write
▽          67256 B1    ./fork_loop.1.1 (3 threads)
        1/1      K      in kernel
        2/258    T      in snore(void *)
        3/515    B1     in snore(void *)
▼          59186 B1    ./fork_loop.2 (3 threads)
        1/1      K      in kernel
        2/258    B1     in snore(void *)
        3/515    T      in ._spin_unlock_global_common
```

Process Window

Addition of toolbars for common actions


fork_loopSGI,1,1
File Edit View Group Process Thread Action Point Tools Window Help
Process Control Go Halt Next Step Out Run To NextI StepI P- P+ T- T+
Stack Trace
Stack Frame
Threads (3)
Action Points

Process 3182238: fork_loopSGI.1.1 (Stopped)

Thread 3182238.1: fork_loopSGI.1.1 (Stopped) <Trace Trap>

```
    __select,          FP=ffffffb80
C   _select,           FP=fffffabf0
C++ wait_a_while,      FP=fffffac30
C++ snore,             FP=fffffacb0
C++ forker,            FP=fffffad40
C++ fork_wrapper,      FP=fffffadb0
C++ main,              FP=fffffae20
```

```
Function 'main':
  argc:              0x00000003 (3)
  argv:              0x7ffffae38 -> 0x7fff
Local variables:
  fork_count:        0x00000003 (3)
  args_ok:           0x00000001 (1)
  arg_count:         0x00000003 (3)
```

Function main in fork_loop.cxx

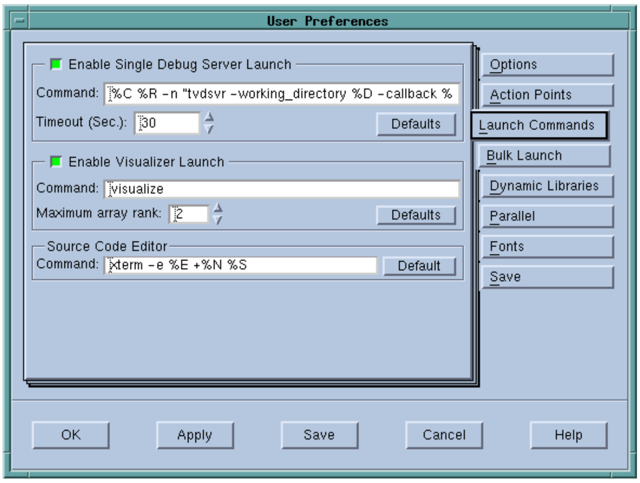
```
      }  /* if */

    if (do_handle_trap)
      {
        signal (SIGTRAP, (void(*)(int))sig_trap_handler);
      }

    fork_wrapper (fork_count);
    printf ("Returned.  How strange.\n");
    return (0);
  }  /* main */
```

```
1/65536   T   in __select
2/65538   T   in __select
3/65537   T   in __select
```

```
STOP  6  line 1327 at main+0xf28 in "fo
```

User Preferences Window

Many TotalView behaviors can be customized



☑ Enable Single Debug Server Launch

Command: %C %R -n "tvdsvr -working_directory %D -callback %

Timeout (Sec.): 30

☑ Enable Visualizer Launch

Command: visualize

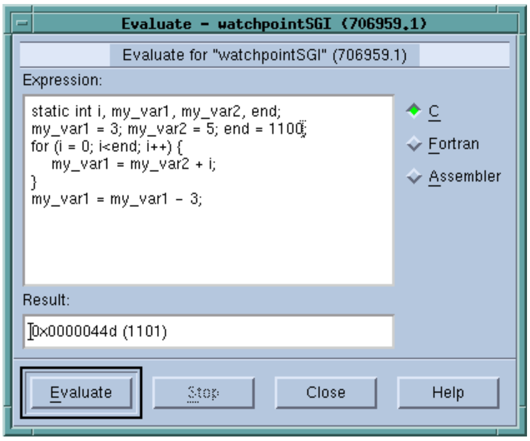
Maximum array rank: 2

Source Code Editor

Command: xterm -e %E +%N %S

Expressions Window

Evaluate expressions on the fly or test fixes before changing source



Evaluate - watchpointSGI (706959.1)

Evaluate for "watchpointSGI" (706959.1)

Expression:

```
static int i, my_var1, my_var2, end;
my_var1 = 3; my_var2 = 5; end = 1100;
for (i = 0; i<end; i++) {
   my_var1 = my_var2 + i;
}
my_var1 = my_var1 - 3;
```

● C ○ Fortran ○ Assembler

Result:

0x0000044d (1101)

Unprecedented Support for MPI and OpenMP

- Automatically acquires processes as they start
- Exposes MPI message queues with Message State Window and the Message Queue Graph
- Supports mixed MPI and OpenMP programs
- Relates OpenMP worker to master threads
- Allows debugging in parallel OpenMP regions
- Offers barrier breakpoints for synchronizing parallel tasks

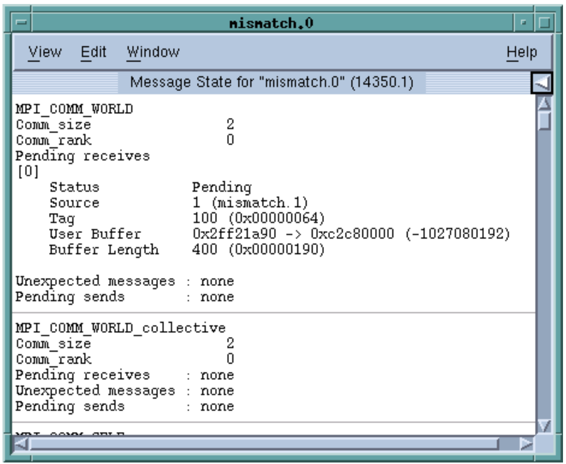
MPI and OpenMP

- Supports MPICH as well as selected vendor-specific versions of MPI
- Supports IBM, KAI, SGI OpenMP compilers

Please consult the release notes or web site for exact versions supported

Message State Window

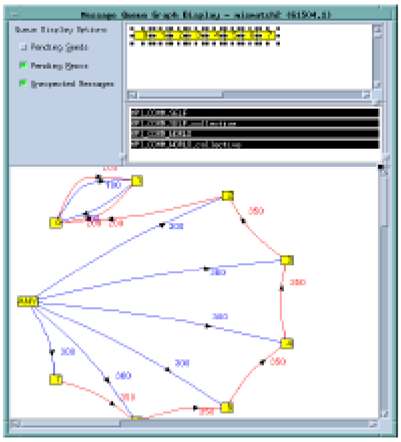
Exposes contents of messages queues so you can understand inter-process communication



```
MPI_COMM_WORLD
Comm_size                    2
Comm_rank                    0
Pending receives
[0]
    Status              Pending
    Source              1 (mismatch.1)
    Tag                 100 (0x00000064)
    User Buffer         0x2ff21a90 -> 0xc2c80000 (-1027080192)
    Buffer Length       400 (0x00000190)

Unexpected messages : none
Pending sends       : none

MPI_COMM_WORLD_collective
Comm_size                    2
Comm_rank                    0
Pending receives    : none
Unexpected messages : none
Pending sends       : none
```

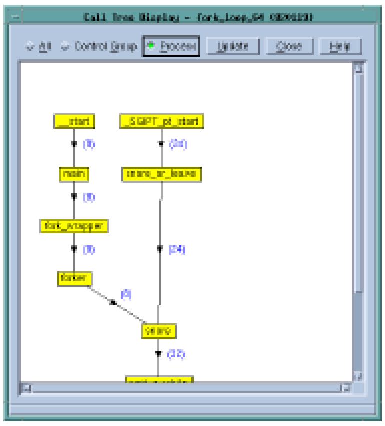
Message Queue Graph

- Visualization of message queues
- Depicts inter-process communication
- Immediately spot the source of deadlock or other inter-process communication problems



Call Tree Graph

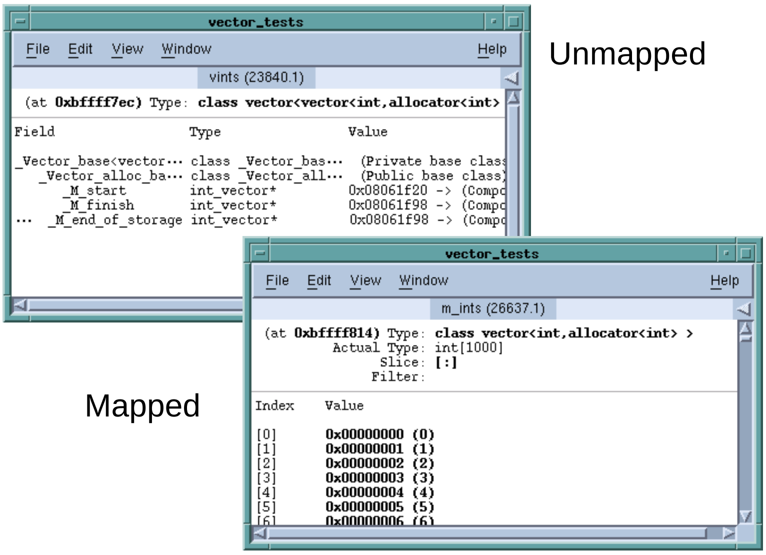
- Displays an amalgamated snapshot of your program’s execution
- See potential bottlenecks or conflicts
- Review flow of execution



Type Mapping

- Most debuggers don’t know how to display certain complex objects
- Now: create a prototype of these objects using the CLI
- TotalView displays them the way you want to see them
- Example uses: complex STL constructs, “recomposing” distributed MPI arrays

Type Mapping Example: std::vector

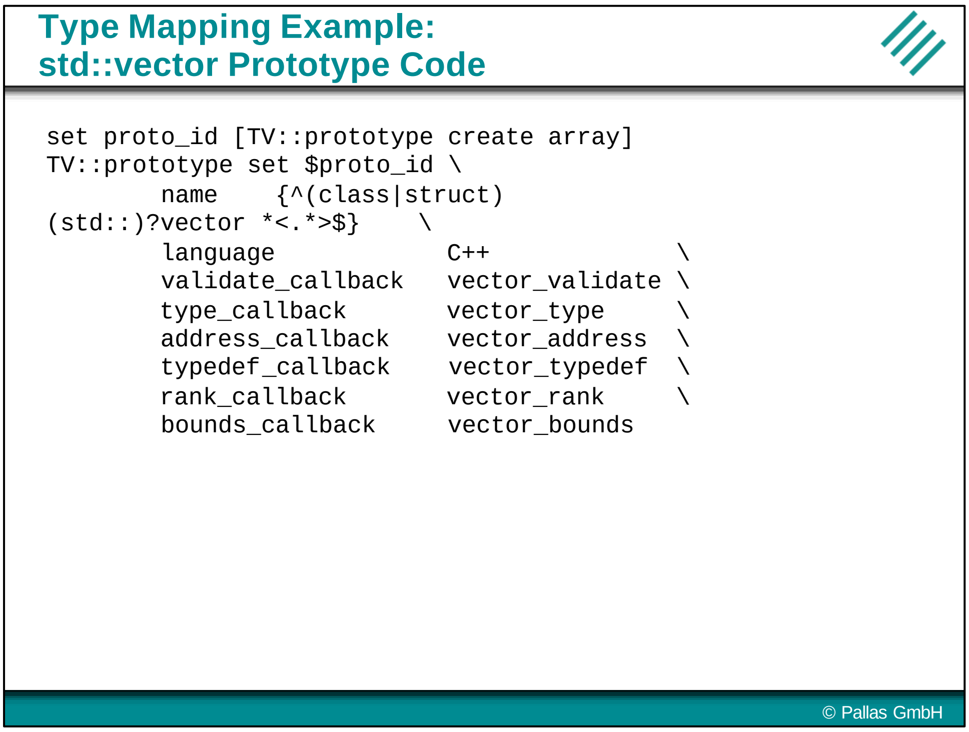


Unmapped

```
vector_tests
File   Edit   View   Window                                   Help
                     vints (23840.1)
 (at 0xbffff7ec) Type: class vector<vector<int,allocator<int>
Field                    Type                 Value

_Vector_base<vector… class _Vector_bas…  (Private base class
   _Vector_alloc_ba… class _Vector_all…  (Public base class)
      _M_start         int_vector*          0x08061f20 -> (Compo
      _M_finish        int_vector*          0x08061f98 -> (Compo
…    _M_end_of_storage int_vector*          0x08061f98 -> (Compo
```

Mapped

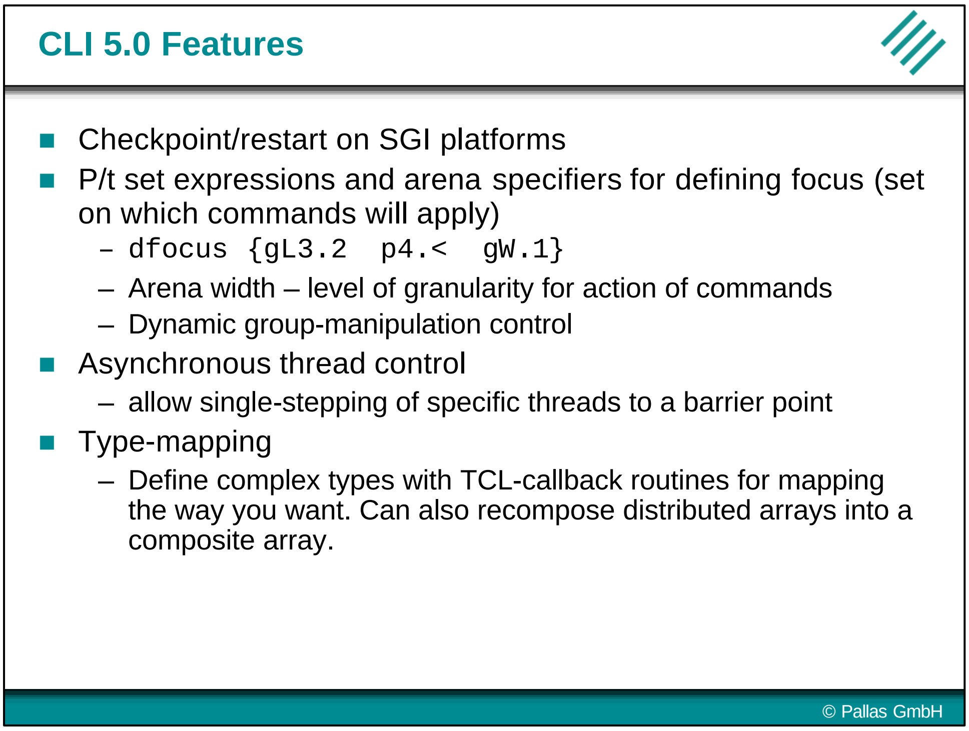
```
vector_tests
File   Edit   View   Window                                   Help
                     m_ints (26637.1)
 (at 0xbffff814) Type: class vector<int,allocator<int> >
          Actual Type: int[1000]
                Slice: [:]
               Filter:
Index     Value

[0]       0x00000000 (0)
[1]       0x00000001 (1)
[2]       0x00000002 (2)
[3]       0x00000003 (3)
[4]       0x00000004 (4)
[5]       0x00000005 (5)
[6]       0x00000006 (6)
```



Type Mapping Example: std::vector Prototype Code

```
set proto_id [TV::prototype create array]
TV::prototype set $proto_id \
        name    {^(class|struct)
(std::)?vector *<.*>$}    \
        language            C++             \
        validate_callback   vector_validate \
        type_callback       vector_type     \
        address_callback    vector_address  \
        typedef_callback    vector_typedef  \
        rank_callback       vector_rank     \
        bounds_callback     vector_bounds
```

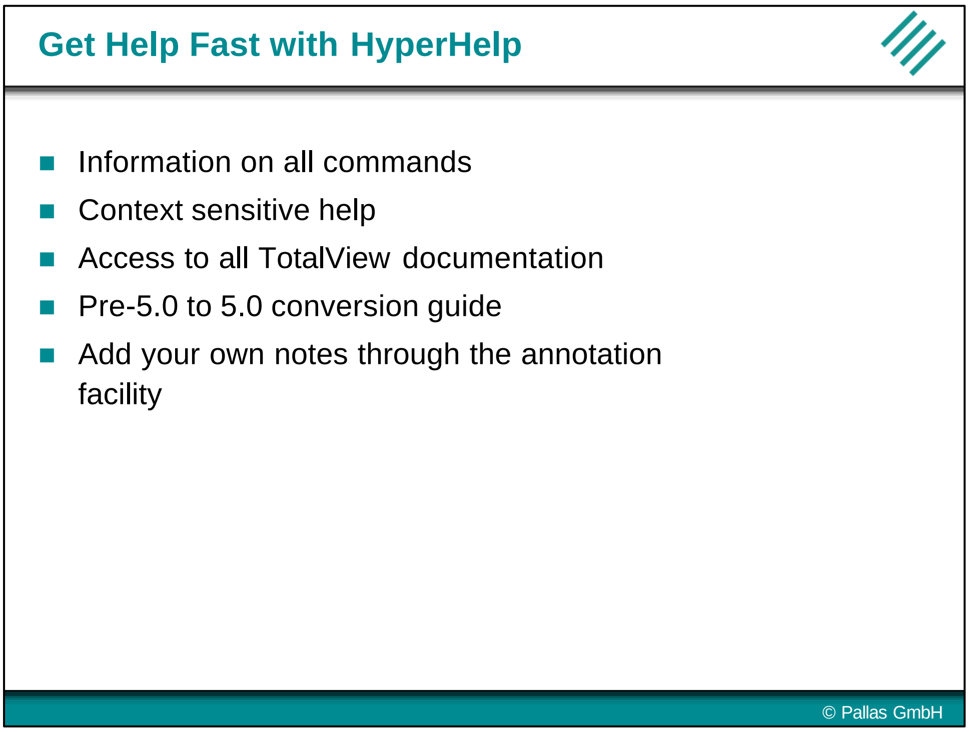
© Pallas GmbH



CLI 5.0 Features

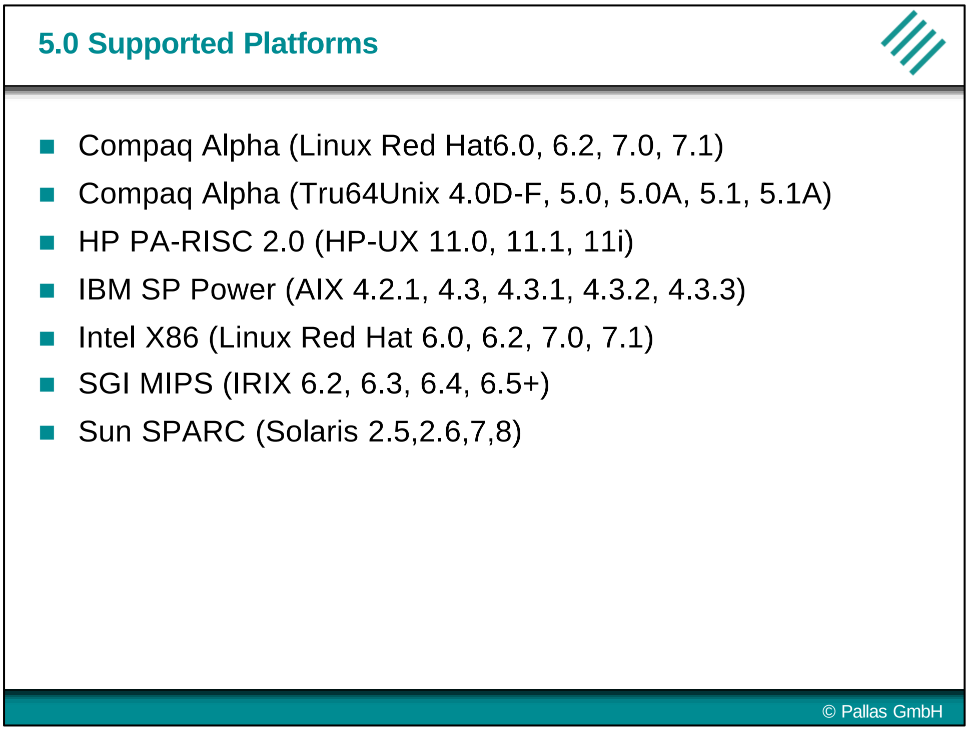
- Checkpoint/restart on SGI platforms
- P/t set expressions and arena specifiers for defining focus (set on which commands will apply)
  - `dfocus {gL3.2  p4.<  gW.1}`
  - Arena width – level of granularity for action of commands
  - Dynamic group-manipulation control
- Asynchronous thread control
  - allow single-stepping of specific threads to a barrier point
- Type-mapping
  - Define complex types with TCL-callback routines for mapping the way you want. Can also recompose distributed arrays into a composite array.

© Pallas GmbH



Get Help Fast with HyperHelp

- Information on all commands
- Context sensitive help
- Access to all TotalView documentation
- Pre-5.0 to 5.0 conversion guide
- Add your own notes through the annotation facility



5.0 Supported Platforms

- Compaq Alpha (Linux Red Hat6.0, 6.2, 7.0, 7.1)
- Compaq Alpha (Tru64Unix 4.0D-F, 5.0, 5.0A, 5.1, 5.1A)
- HP PA-RISC 2.0 (HP-UX 11.0, 11.1, 11i)
- IBM SP Power (AIX 4.2.1, 4.3, 4.3.1, 4.3.2, 4.3.3)
- Intel X86 (Linux Red Hat 6.0, 6.2, 7.0, 7.1)
- SGI MIPS (IRIX 6.2, 6.3, 6.4, 6.5+)
- Sun SPARC (Solaris 2.5,2.6,7,8)

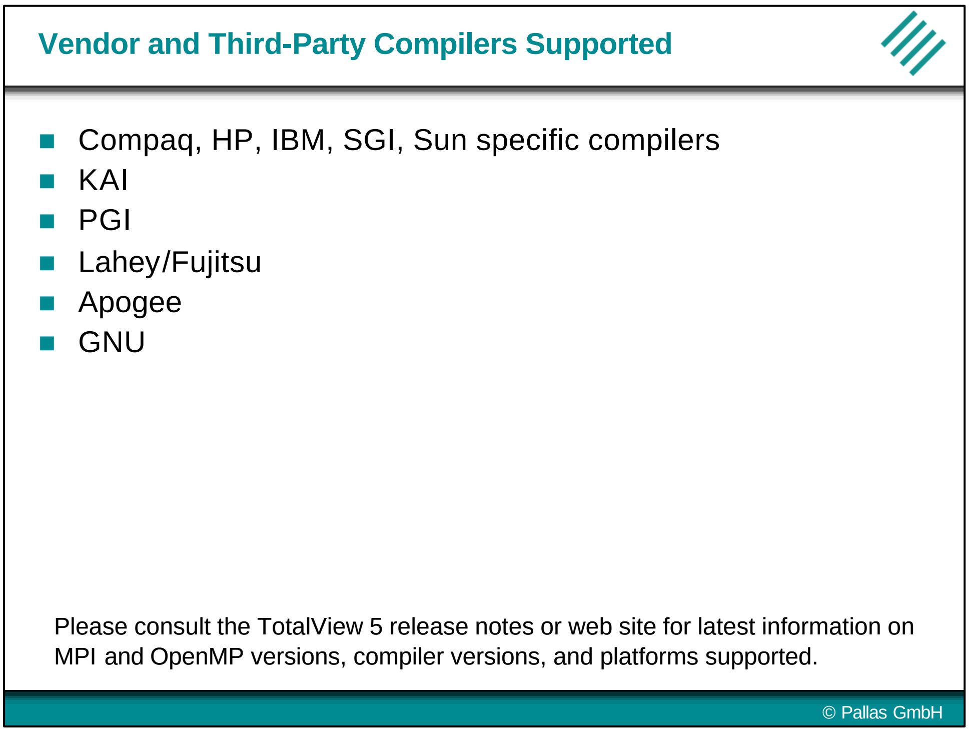
Programming Languages

- C
- C++
  - Templates
  - Inlined functions and code in header files
  - Base and virtual classes
- FORTRAN77
- Fortran90
  - F90 types
  - Modules
  - Assumed shape arrays
- Mixed C/C++ and Fortran
- Assembler

Other Parallel Models

In addition to MPI, OpenMP, and threads, TotalView supports:

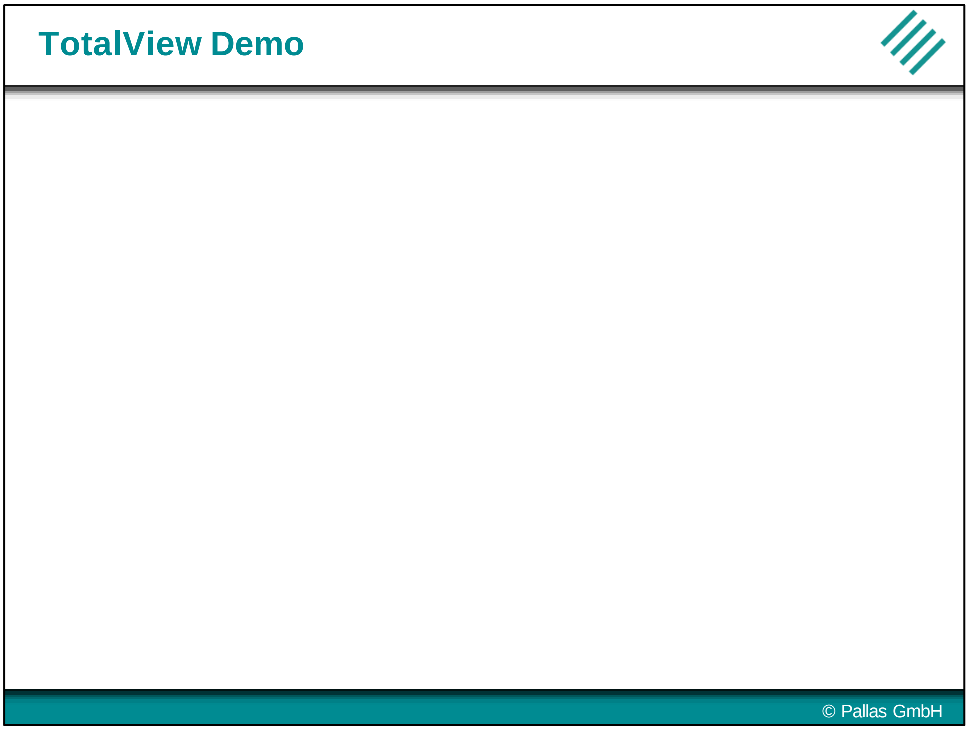
- HPF
- PVM
- Shmem
- Fork/exec



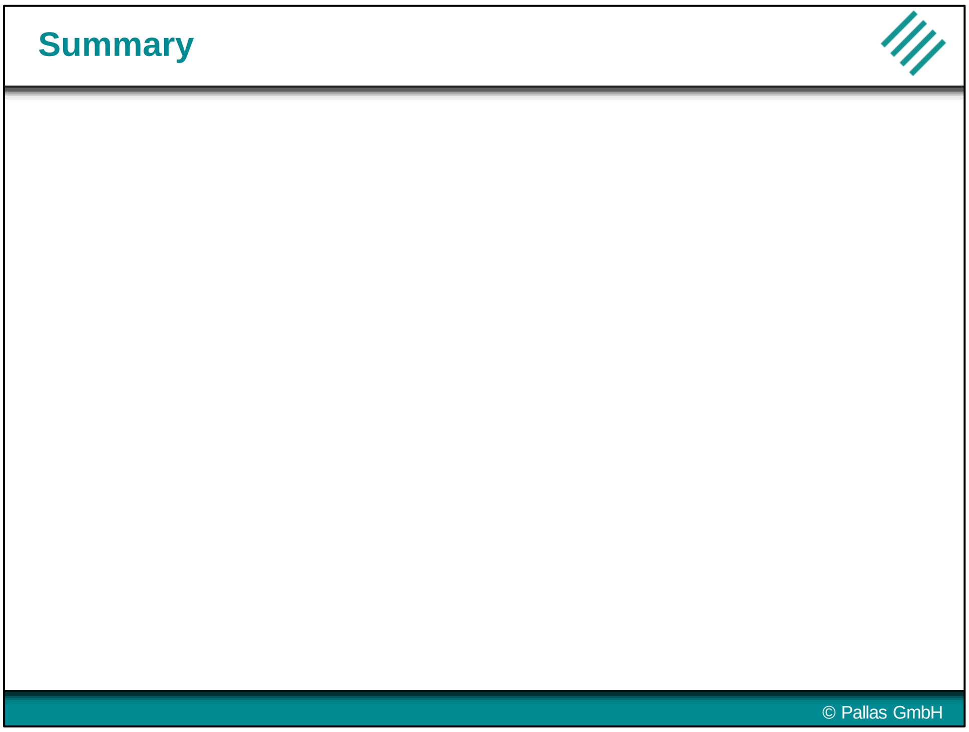
Vendor and Third-Party Compilers Supported

- Compaq, HP, IBM, SGI, Sun specific compilers
- KAI
- PGI
- Lahey/Fujitsu
- Apogee
- GNU

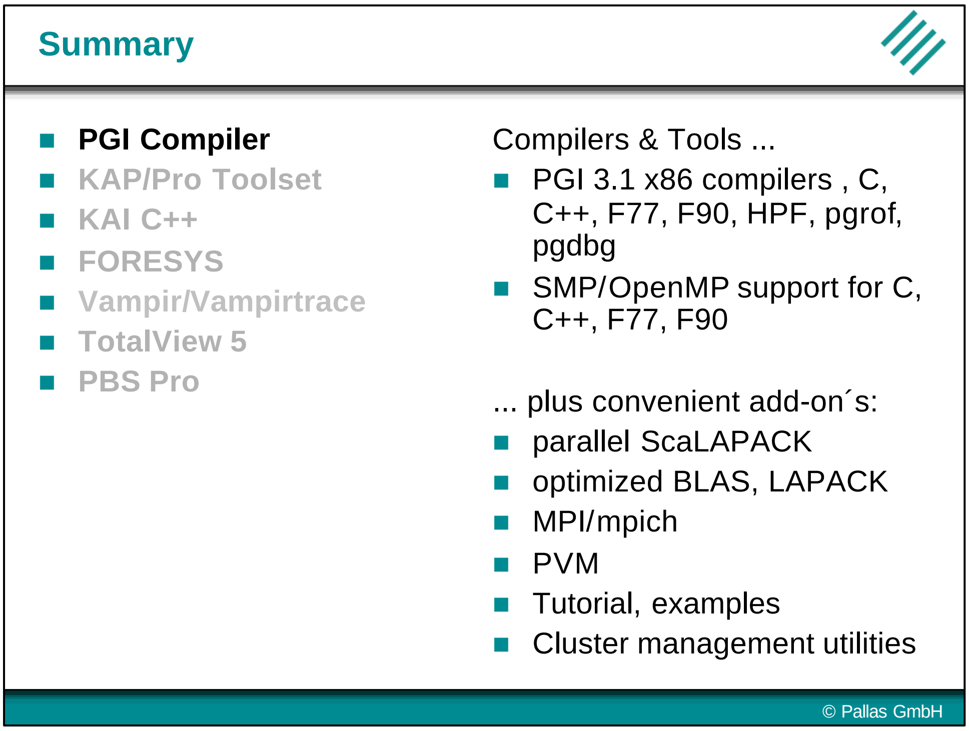
Please consult the TotalView 5 release notes or web site for latest information on MPI and OpenMP versions, compiler versions, and platforms supported.



TotalView Demo



Summary



Summary

- **PGI Compiler**
- KAP/Pro Toolset
- KAI C++
- FORESYS
- Vampir/Vampirtrace
- TotalView 5
- PBS Pro

Compilers & Tools ...

- PGI 3.1 x86 compilers , C, C++, F77, F90, HPF, pgrof, pgdbg
- SMP/OpenMP support for C, C++, F77, F90

... plus convenient add-on´s:

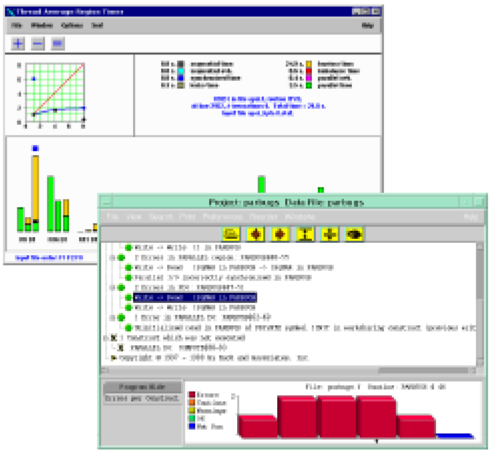
- parallel ScaLAPACK
- optimized BLAS, LAPACK
- MPI/mpich
- PVM
- Tutorial, examples
- Cluster management utilities

Summary

- PGI Compiler
- **KAP/Pro Toolset**
- KAI C++
- FORESYS
- Vampir/Vampirtrace
- TotalView 5
- PBS Pro

OpenMP 2.0

Compiler and Tools



© Pallas GmbH

Summary

- PGI Compiler
- KAP/Pro Toolset
- **KAI C++**
- FORESYS
- Vampir/Vampirtrace
- TotalView 5
- PBS Pro

The most modern, best performing, platform independant C++

- ISO C++ standard syntax, including exeptions and member templates
- ISO C++ standard class library
- multi-platform support
- meet C performance requirements
- thread safety (on most platforms)

© Pallas GmbH

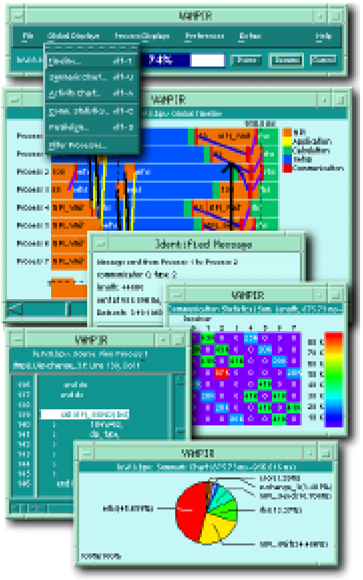
Summay

- PGI Compiler
- KAP/Pro Toolset
- KAI C++
- **FORESYS**
- Vampir/Vampirtrace
- TotalView 5
- PBS Pro

- Translate code from Fortran 77 to Fortran90
- FORTRAN code consistency checks
- Interactive visualization & analysis of inconsistencies
- Interactive/batch analysis of parallelization possibilities
- Automatic code quality analysis/improvements

Summary

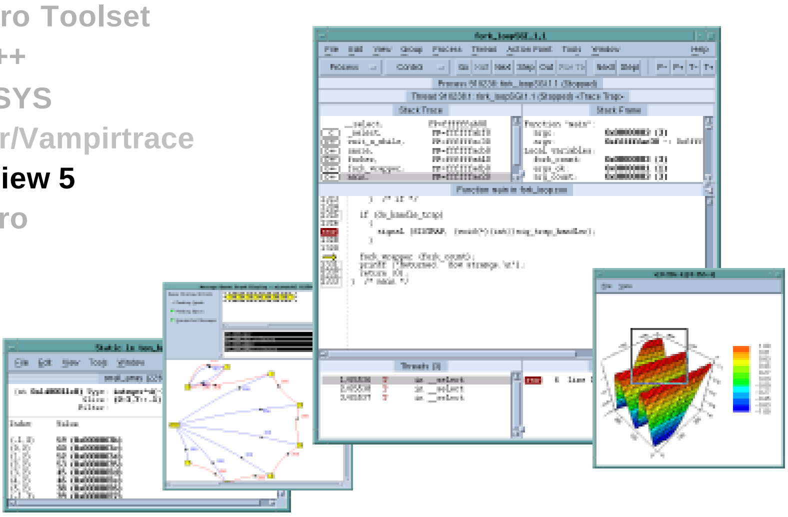
- PGI Compiler
- KAP/Pro Toolset
- KAI C++
- FORESYS
- **Vampir/Vampirtrace**
- TotalView 5
- PBS Pro



Summary

- PGI Compiler
- KAP/Pro Toolset
- KAI C++
- FORESYS
- Vampir/Vampirtrace
- **TotalView 5**
- PBS Pro

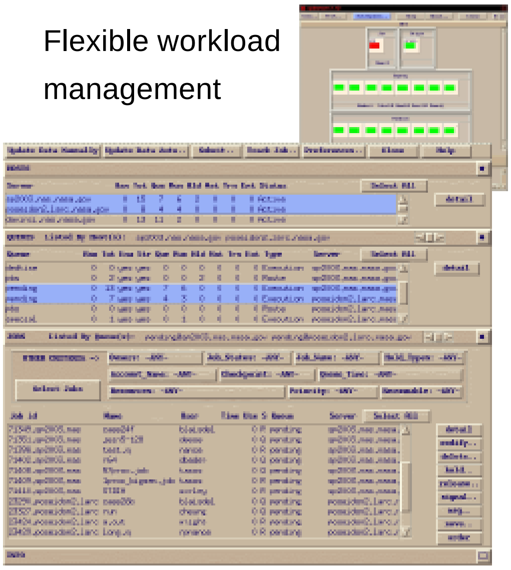
Debug complex applications

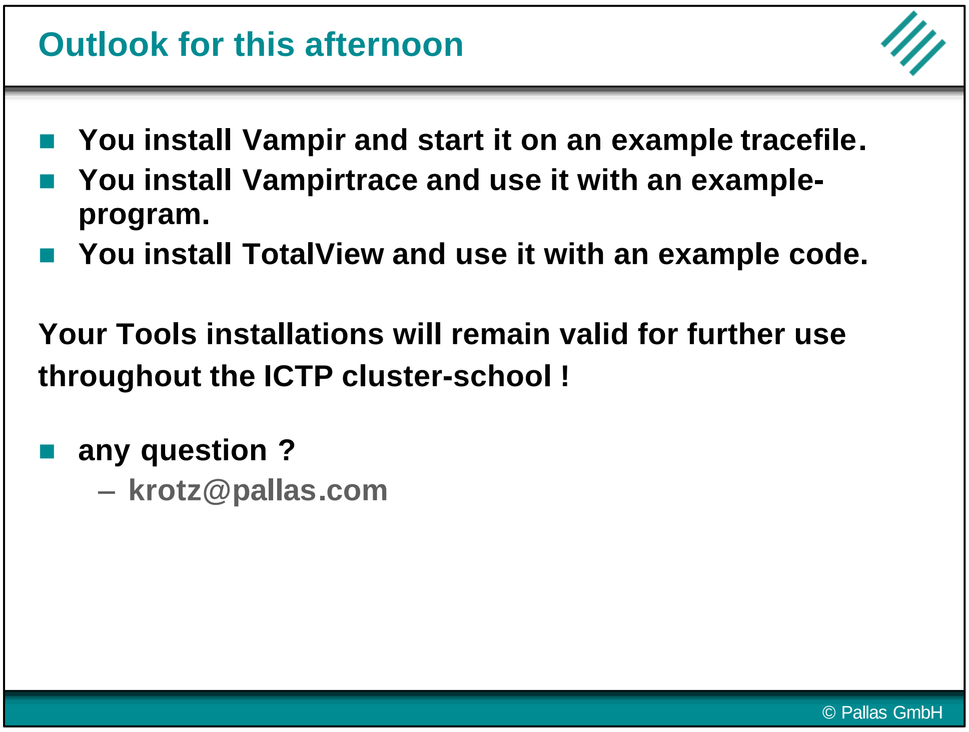


Summary

- PGI Compiler
- KAP/Pro Toolset
- KAI C++
- FORESYS
- Vampir/Vampirtrace
- TotalView 5
- **PBS Pro**

Flexible workload management





Outlook for this afternoon

- You install Vampir and start it on an example tracefile.
- You install Vampirtrace and use it with an example-program.
- You install TotalView and use it with an example code.

Your Tools installations will remain valid for further use throughout the ICTP cluster-school !

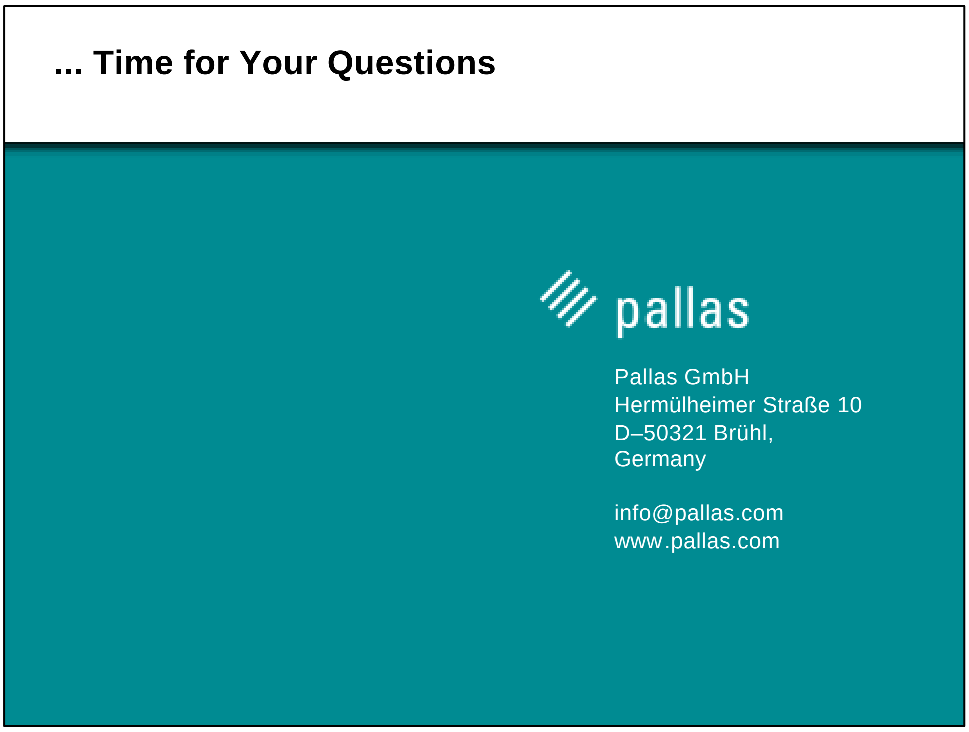
- any question ?
  - krotz@pallas.com

Access to Pallas Tools

Download free evaluation copies

http://www.pallas.com

Buy 1000 printf(), get one free ;-)



... Time for Your Questions

pallas

Pallas GmbH
Hermülheimer Straße 10
D–50321 Brühl,
Germany

info@pallas.com
www.pallas.com