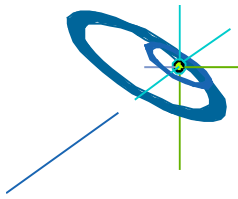
Automatic MPI Counter Profiling

Rolf Rabenseifner

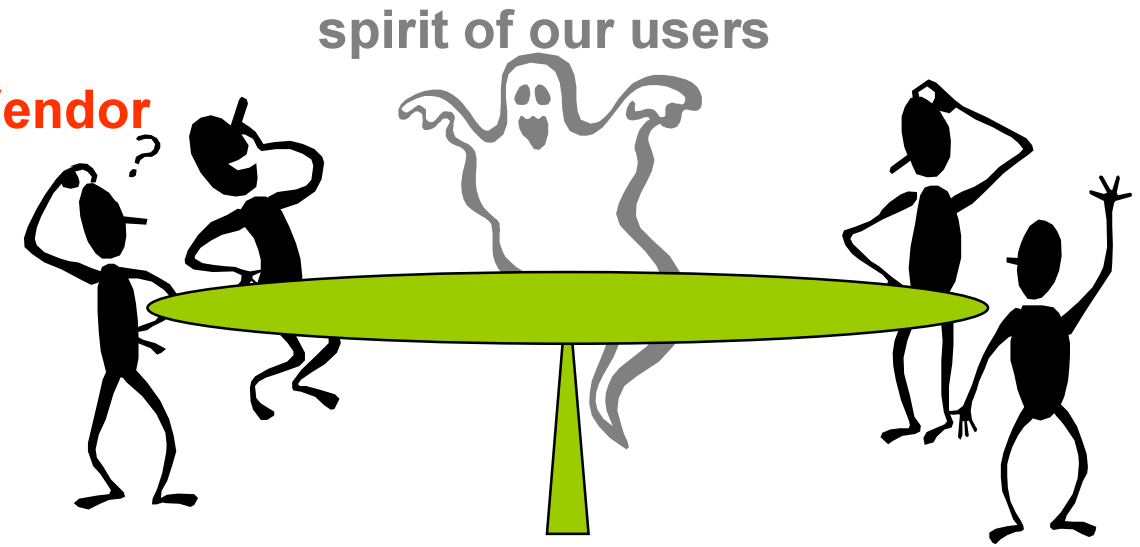
High-Performance Computing-Center Stuttgart (HLRS), University of Stuttgart,

CUG Summit 2000

Noordwijk, The Netherlands, May 22-26, 2000



The beginning – at CUG Spring 1998



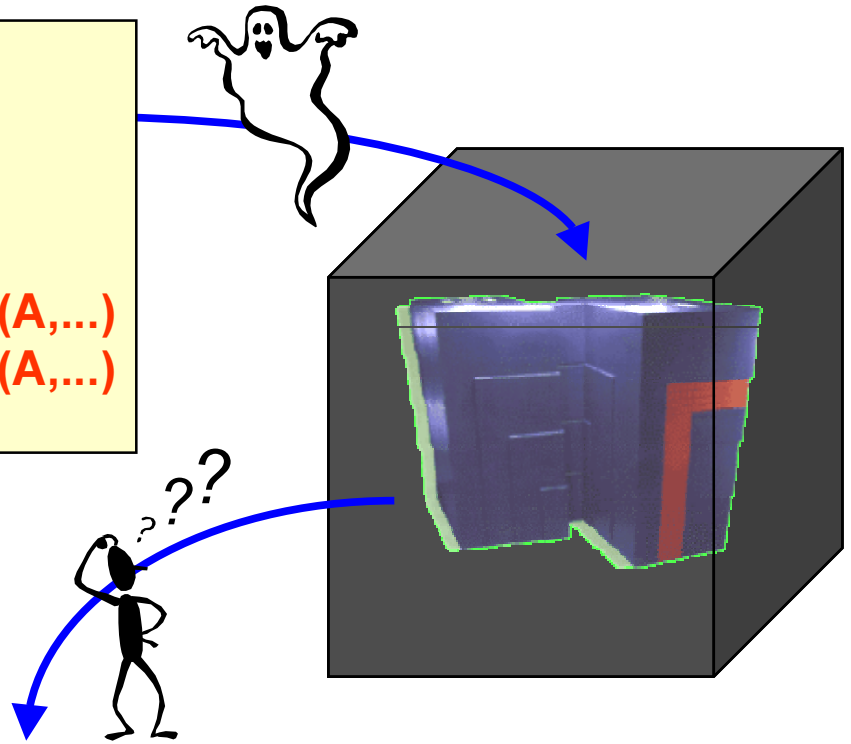
MPI on T3E – quo vadis?

What is more important for our users?

- Optimization of existing MPI routines?
  - and which MPI routines are most important?
- New MPI-2 features?

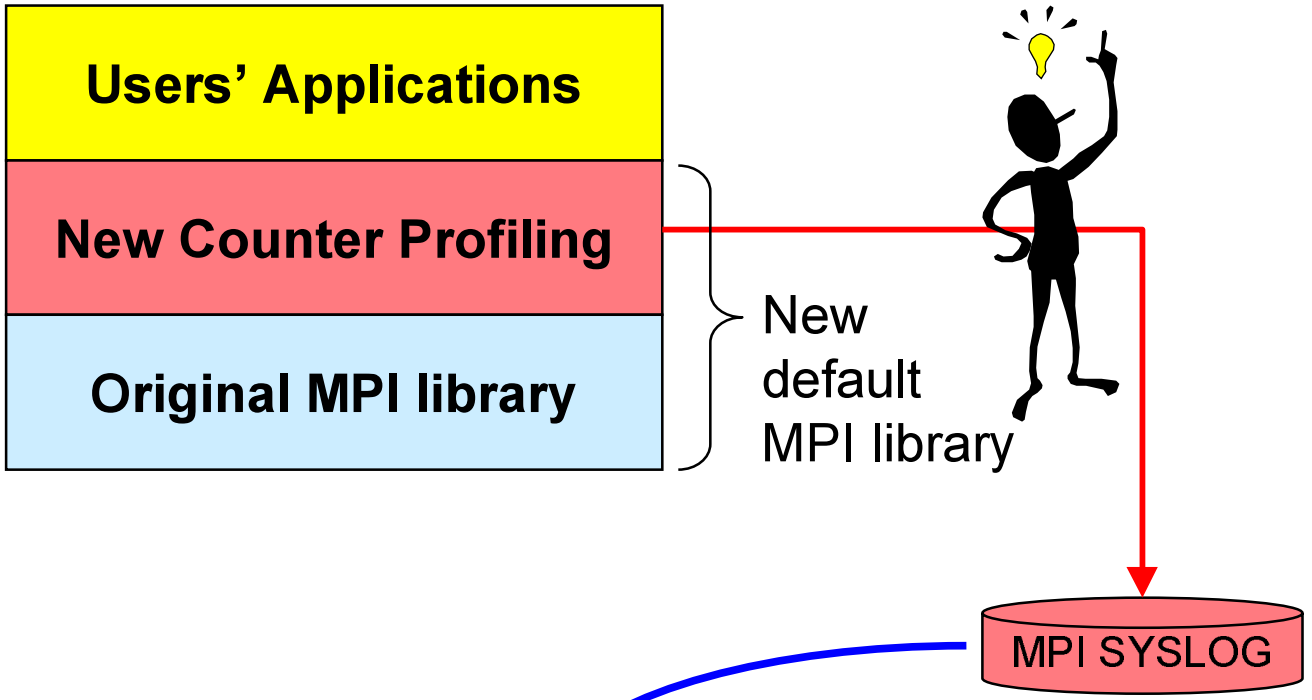
Communication percentage

```
WHILE ...
  DO I=1,100000
    A(I) = B(I)*C(I)
  ENDDO
  CALL MPI_SEND(A,...)
  CALL MPI_RECV(A,...)
ENDWHILE
```



$$\text{Communication percentage} = \frac{\text{Communication time}}{\text{Total execution time}}$$

Adding a counter profiling layer – writing to SYSLOG

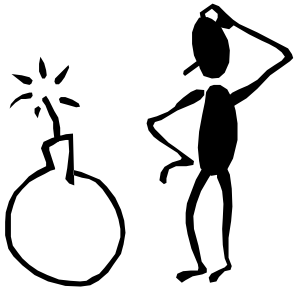


$$\text{Communication percentage} = \frac{\text{Communication time}}{\text{Total execution time}}$$

How to add a profiling layer?

Each MPI library has a standardized **P**MPI interface for profiling!

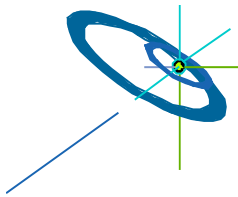
Good idea! Looks very easy!



But this PMPI profiling must be available for user's profiling!

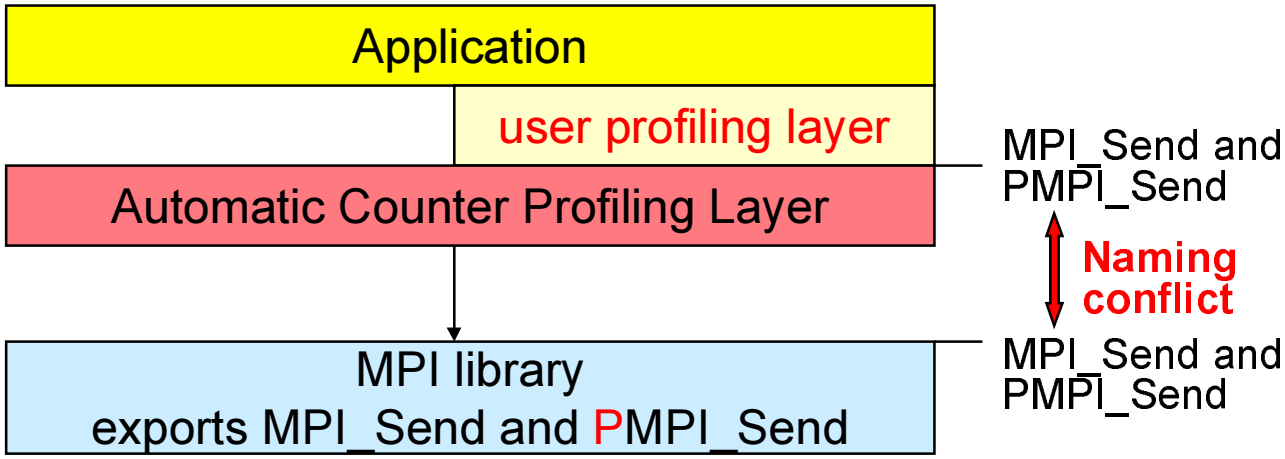
It can not be used for the automatic profiling of the computing center!

But this problem can be solved!



HLRS

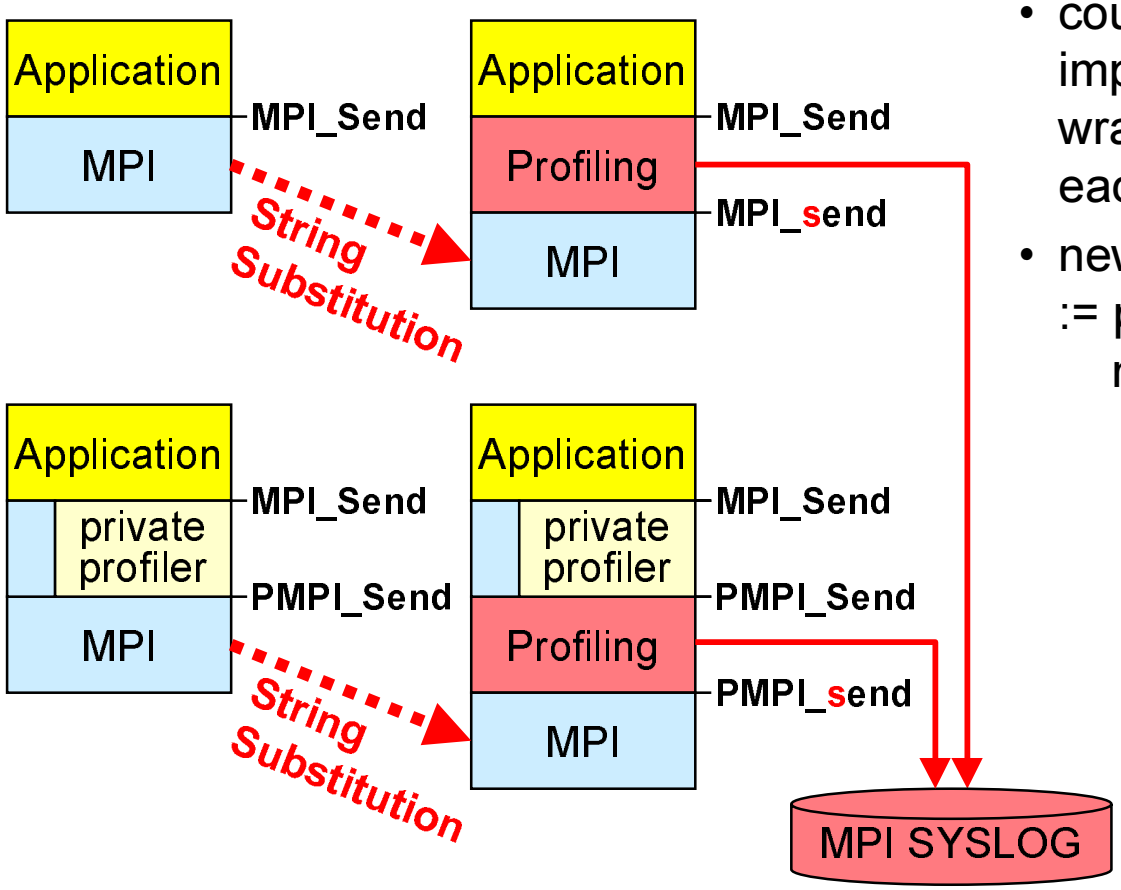
Additional profiling layer for automatic profiling



- Simple layering without consuming PMPI: Naming conflict
- Solution: for library libmpi.a (e.g., on Cray T3E)
  - renaming routine names in libmpi.a
    with a string substitution program for binary files

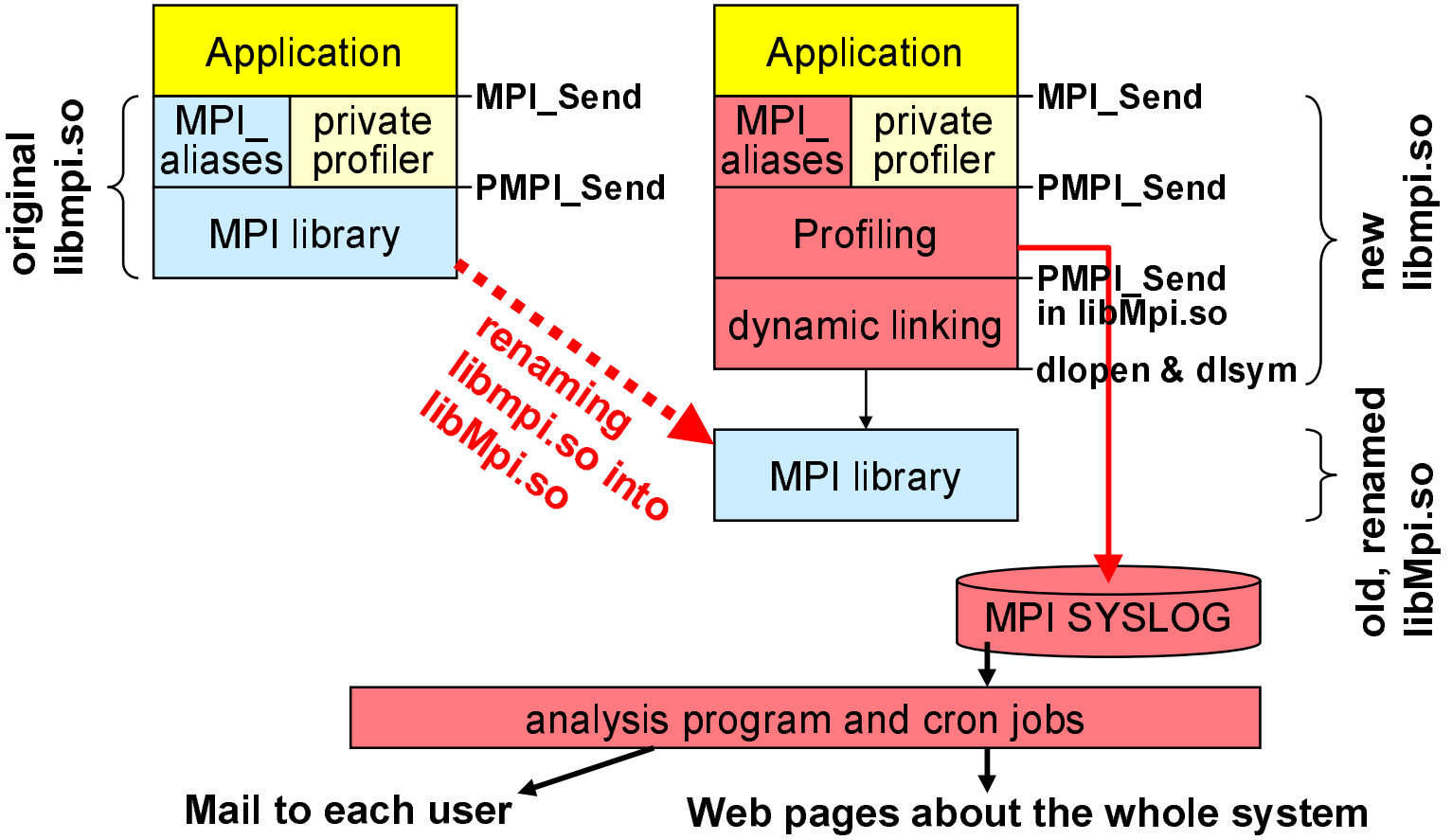
  for dynamic shared objects (DSO) libmpi.so (e.g., on IRIX)
  - the counter profiling layer can dynamically link the original MPI DSO

Instrumentation: Software design for libmpi.a



- counter profiling is implemented as wrapper routines to each MPI routine
- new libmpi.a
  := profiling +
  modified libmpi.a

Instrumentation: Software design for libmpi.so

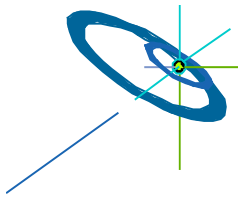


What is counted?

Instrumentation:

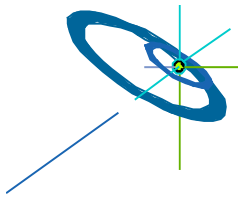
- for each MPI routine
  - **the number of calls**
  - **the time spent in the routine**
  - **the number of transferred bytes**

- hardware performance counters
  - **floating-point instructions**
  - **total instruction count**
  - **separated counting:**
    - user's application
    - MPI routines



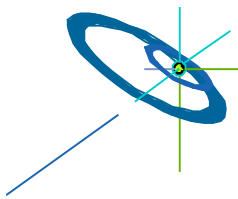
Results
on
Cray T3E-900/512

Conclusions
to the
discussions at CUG Spring ‘98

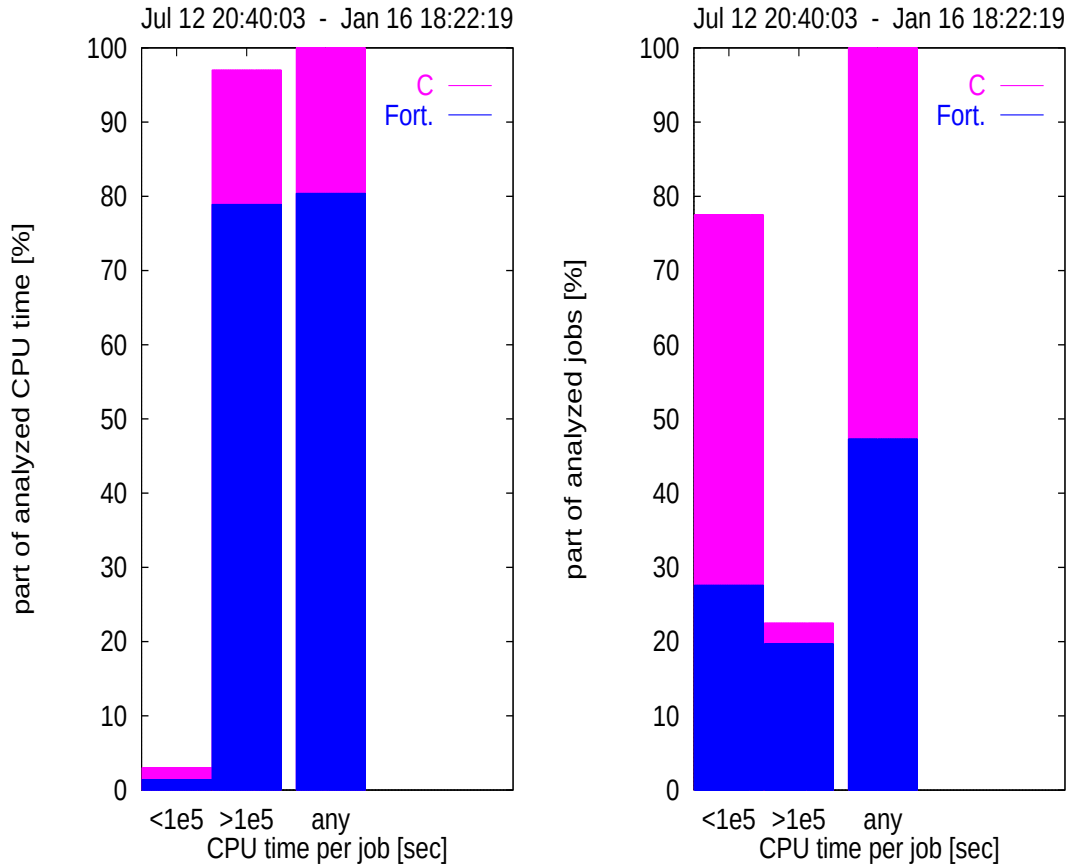


Half-year results

- On the CRAY T3E 900-512 at the HWW/HLRS Stuttgart
- 18.4 % of a half-year (**July 12, 1998 - Jan 16, 1999**) was analyzed
- ~ 80 % not analyzed due to
  - **not linked with profiling library, or**
  - **HPF applications, or**
  - **PVM applications, or**
  - **shmem application, or**
  - **aborted, or**
  - **not calling MPI_Finalize**
  - **system maintenance**
  - **idle time**



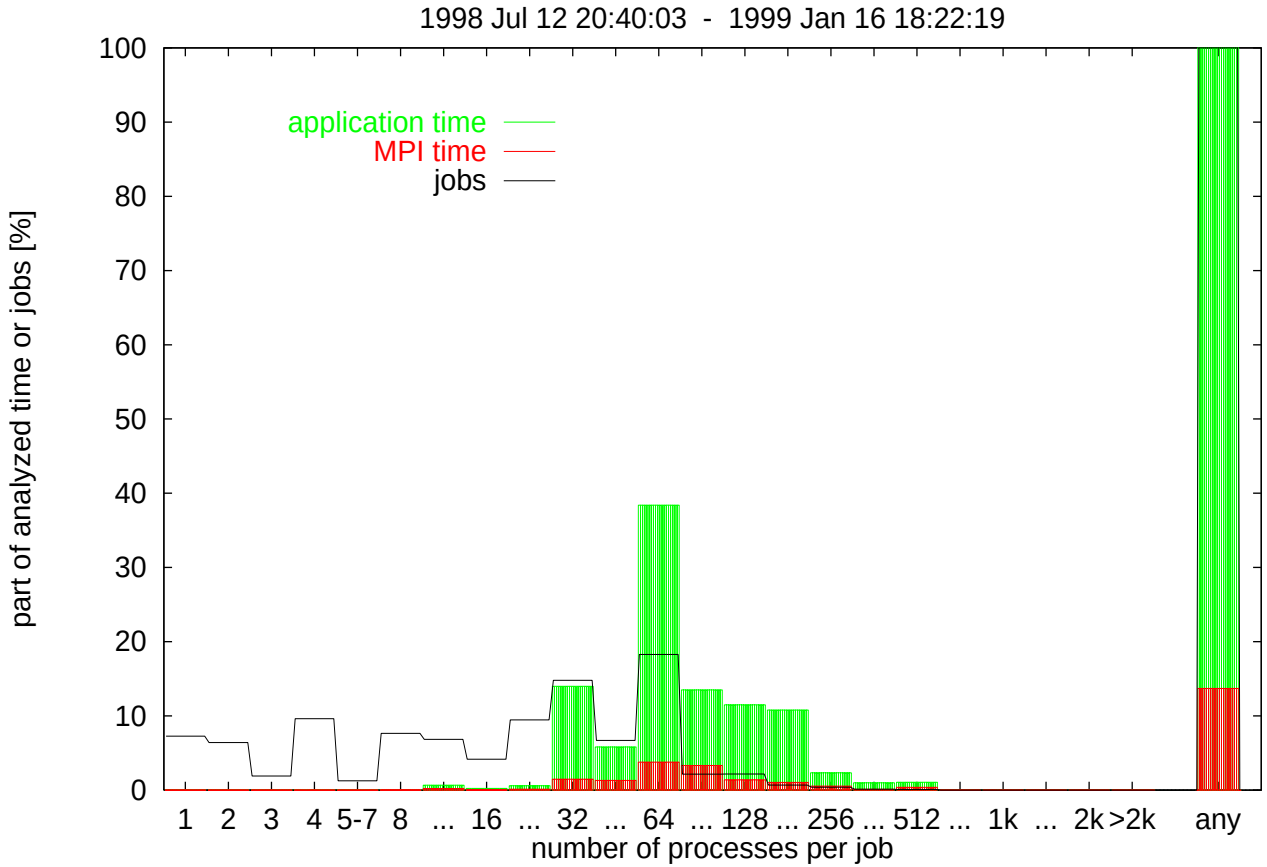
Half-year results: **Fortran & C – time** (left) **& jobs** (right)



MPI jobs:

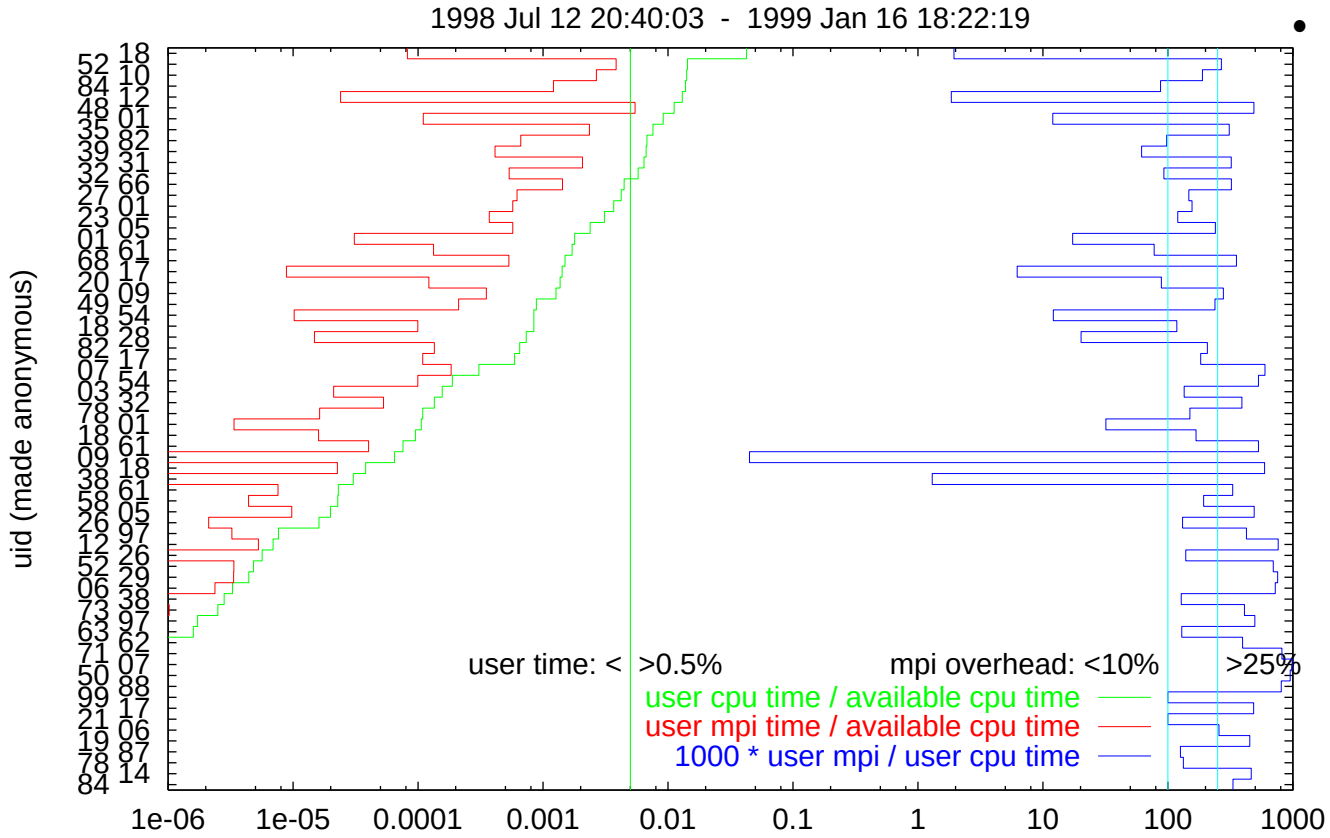
- Most of the **CPU time** is accounted for by long running Fortran jobs (production)
- but most of the **jobs** are short running C programs (development)

Half-year results: **Usage of different partition sizes**



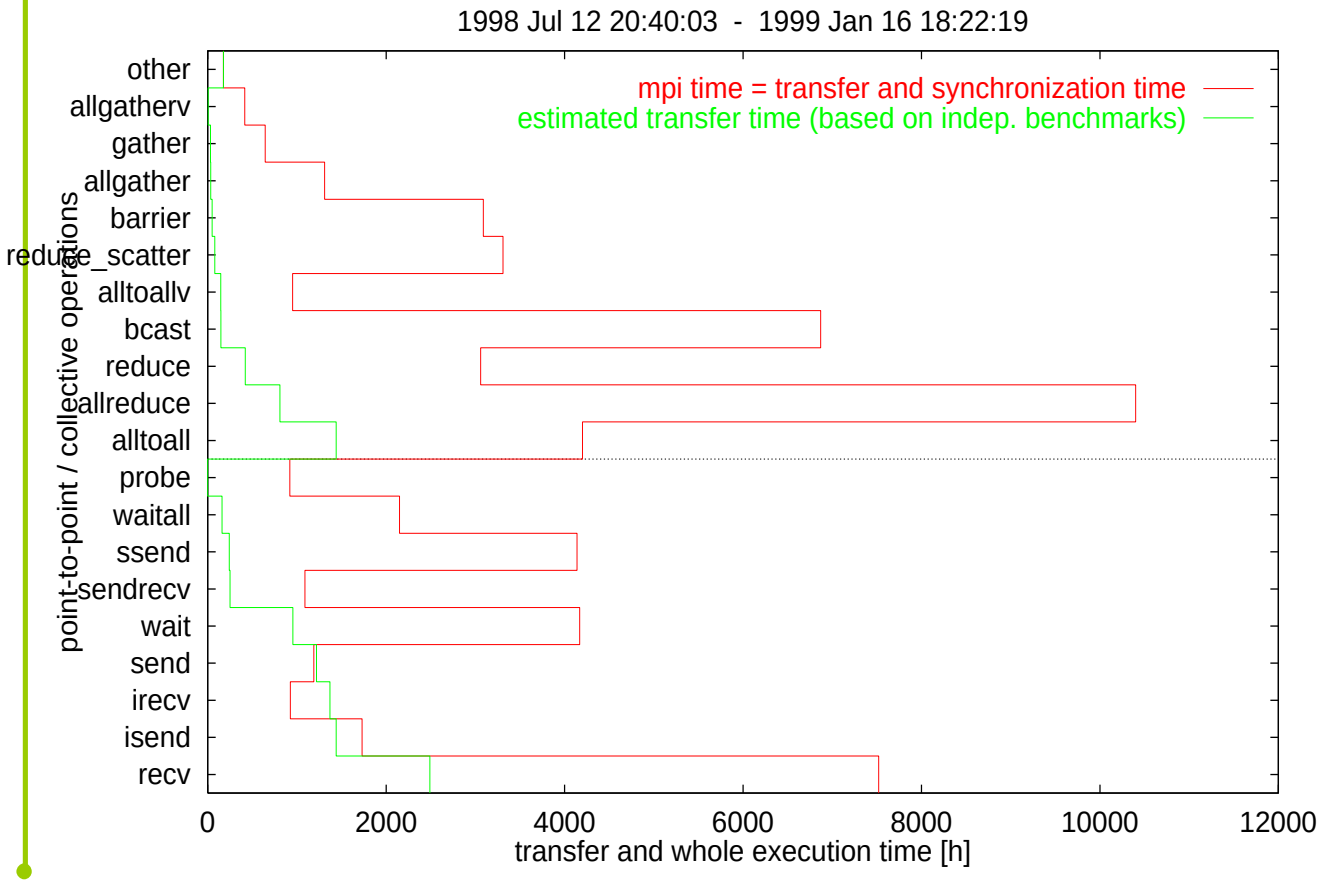
MPI time is 14% of analyzed CPU time

Half-year results: **User's profiles**



- Most Users with less than 30 % MPI percentage

Half-year results: **MPI routines' profiles**

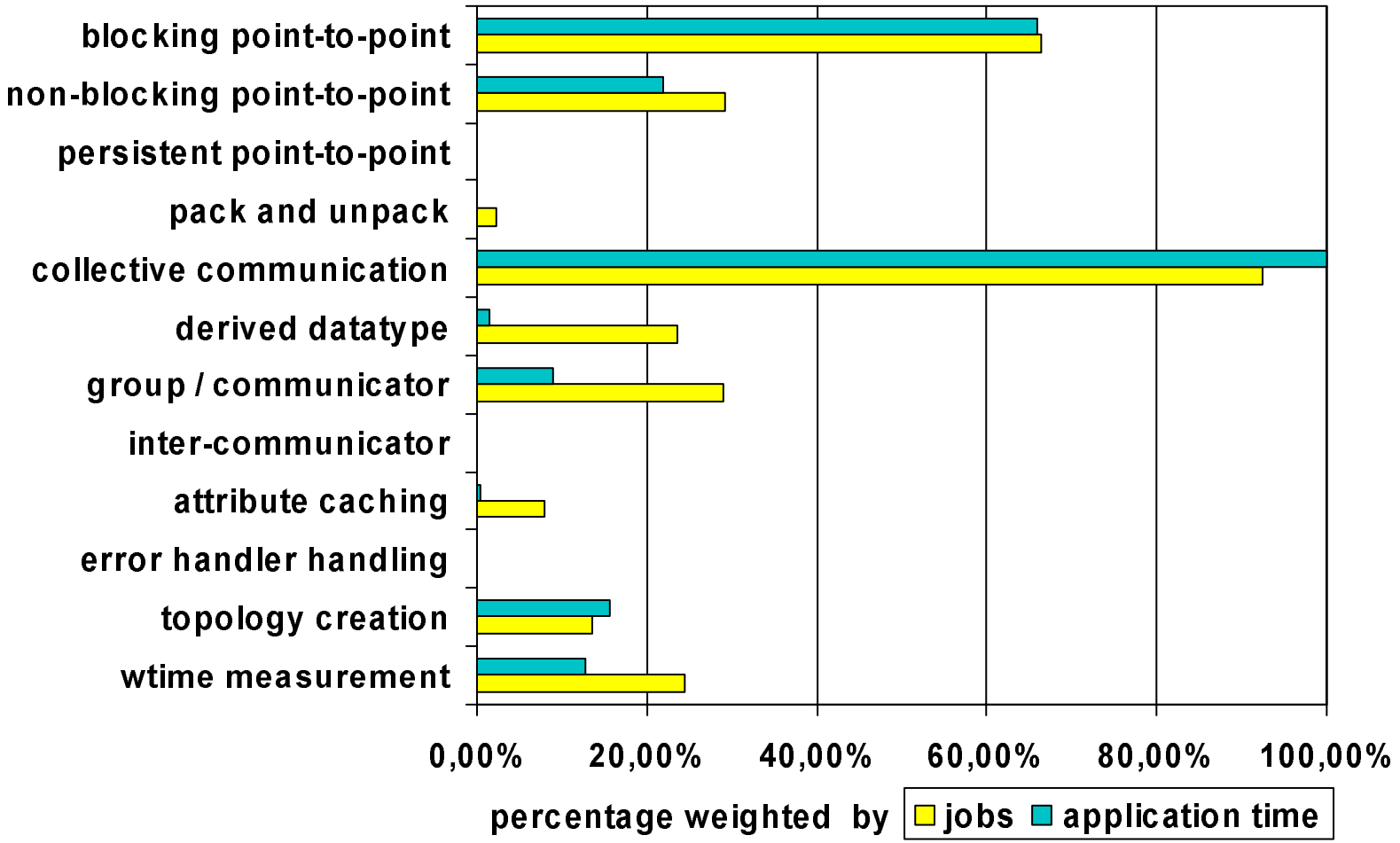


- 99% of MPI time was spent in 19 MPI routines
- ~ 20% of MPI time used for commu-nication[1)]
- ~ 80% for synchro-nization[1)]

[1)] Based on a rough estimate

HLRS

Half-year results: **Usage of the MPI routines' groups**

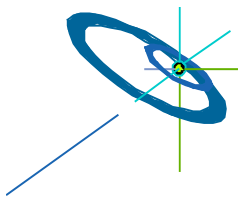


Half-year results: **Details in the proceedings**

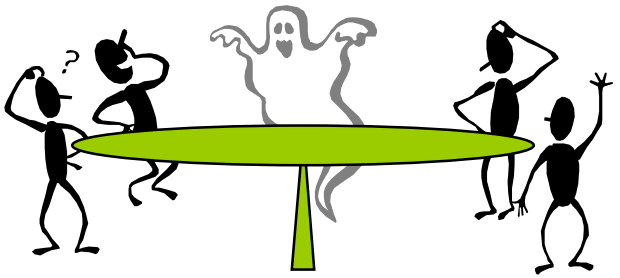
- 10289 jobs with
- 425,288 hours CPU time
- 68 users in all
- 21 production-users have consumed 95% of the CPU time
- 92 MPI routines

  --> detailed usage statistics in the proceedings

- **minimal overhead** on a T3E-900:
  - about 1/1000 of the execution time of our users
  - 0.3 - 0.5 µsec / MPI call (without hardware counters)
  - 2 µsec / MPI call (hardware counters included)
  - 0.1 sec / mpirun
  - 300 kbyte / process



Back to our discussion



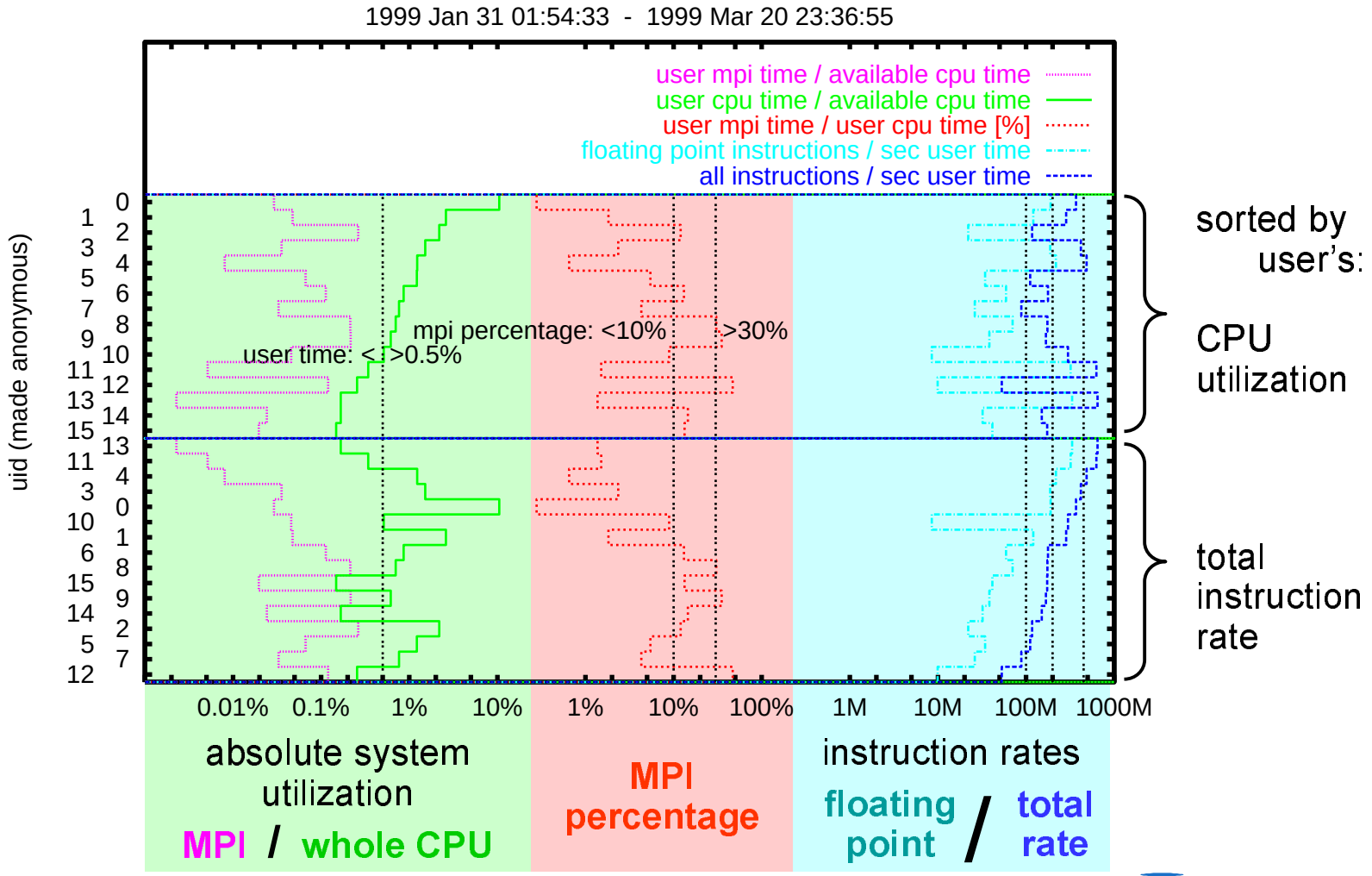
- 14 % MPI percentage
  —> Optimization of Cray's MPI library
    —> no first priority!
  —> we want MPI-2 features
    e.g., **optimized** MPI-I/O
    —> accumulated MPI-I/O bandwidth: **www.hlrs.de/mpi/b_eff_io**
- Accumulated latencies < 1/10 of the MPI percentage
  —> we didn't install mpich ported by Shane Hebert et al.
    - **mpich: 7 µsec latency on T3E (see MPIDC'99)**
    - **mpt.1.3.0.4: 16 µsec latency on T3E**
- Usage of C in the development jobs
  —> we expect more C and C++ in future production jobs
- Looking at the 19 MPI routines with 99 % of the MPI time
  —> all major MPI features are really used

Integrating the hardware performance counters

- Goal
  - provide users not only with **communication** timings,
  - but also with timing data about the **computation**
- each micro-processor has 2-4 hardware performance counters
- e.g. as profiling default:
  - total instruction count
  - floating point instruction[1)] count
- necessary to separate counting of instructions
  - inside the user code
  - inside the MPI routines

[1)] **1 instruction = 1 or 2 operations**

First result with hardware counters on a T3E900-512



HLRS

Summary of the computing center's interface

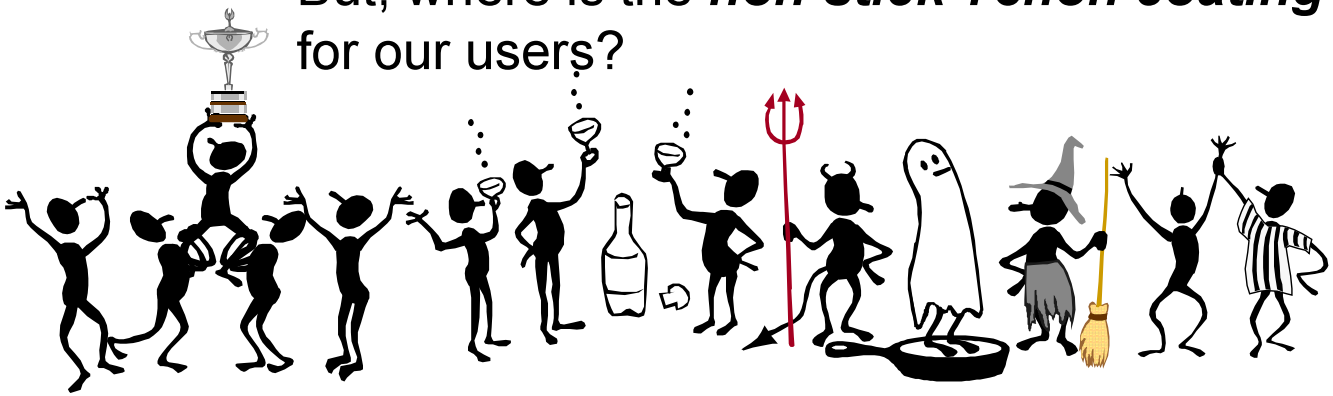
- Counter Profiling layer
  - integrated in the default MPI library
  - counting each MPI call
  - floating point / total instructions count
  - writing all data at the end of a job to a special syslog file

- Automatic weekly and monthly analysis of the syslog file
  - web-page with all statistical data
  - summary-e-mail to the computing center

And our Users?

A small step for a man...

But, where is the ***non-stick Teflon coating*** for our users?



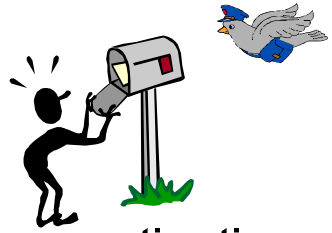
User's Interface

- Weekly e-mail
  - based on the SYSLOG file, a cron-job sends
    - **to each user**
    - **a summary of all his MPI jobs**
    - **each weekend**

- Counter data on a user's output file
  - by setting the environment variable MPIPROFOUT
  - additional printings via MPI_Pcontrol()

- from byproduct to main product!

Scalable user interface



Level 1: Reading the summary of a weekly e-mail
- **e.g. MPI percentage, floating point / total instruction rate**

Level 2: Reading the details of the e-mail
- **e.g. for each MPI routine: number of calls, transferred bytes, execution time**

Level 1+2 are generated on the basis of the profiling syslog file

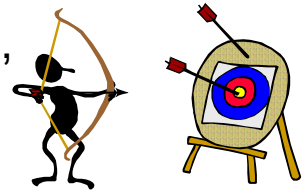
Level 3: generating per-job output

Level 4: additional printings in the per-job output
by using MPI_Pcontrol (standard MPI profiling API)

Level 5: selecting other hardware counters

*Level 3-5 provide the information base for **basic optimizations** of the application and to **decide whether additional tools** should be used*

Level 6: using additional profiling tools,
e.g., VAMPIR



Summary

- Automatic counter based MPI profiling
- timing of all MPI routines & hardware performance counters
- no source code modifications
- detailed statistics for the computing center
- scalable user-interface

  - **to get the right amount of performance data**
  - **to the right person**
  - **in time**

- basis for the further optimization process,
  e.g. with VAMPIR (traced-based MPI profiling)
- closes the gap between *job accounting* and *trace based profiling*
- additional profiling layer for MPI libraries (T3E) and MPI DSOs (IRIX)
- used and accepted by our customers

Further information: www.hlrs.de/mpi/mpi_t3e.html#profiling