TALKS WITH
DR. ROLF RABENSEIFNER

DR. ROLF RABENSEIFNER DEVELOPED METHODS FOR APPLYING THE BUTTERFLY PRINCIPLE TO ANY NUMBER OF PARALLELL PROCESSES

Were there scientific questions that your work on MPI contributed to solving? Can you give an example?

At the beginning of my career, I was focused on research. In reduction operations, I developed methods for applying the butterfly principle to any number of parallel MPI processes. In my Ph.D., I developed a method for posterior clock synchronization for trace files that could compensate not only for the linear clock errors of the clock quartzes in the range of 10E-6, but also for the additional fluctuations in the range of 10E-8 due to the fluctuating temperatures in our supercomputers. I was pleased that my work was then applied in another Ph.D. by a tool professional about 10 years later. In the "b_eff_io" benchmark, I applied the ideas of the Effective Bandwidth Benchmark to I/O.

Why did we need MPI and what did it bring to the table to overcome challenges which could not be solved by PVM?

I already belong to the post-PVM generation. It was important that the insights on message passing gained with PVM were incorporated into a universally accepted standard. This is MPI. All MPI application programs are portable and can now be executed with all MPI libraries. Just like an M10 screw can be turned into any M10 nut.
But the most important aspect is that MPI guides the application programmer to package the data to be exchanged with the fewest possible messages. There are still many parallelization approaches today, such as the PGAS languages, where this is not the case.
By using this packaging approach, MPI programs make the best use of the communication bandwidth of the hardware and do not waste resources by communicating in latency.

IMAGE CAPTION
1. Dr. Rolf Rabenseifner at the colloquium organised by HLRS in his honour in May 2022

What are MPI's major challenges at this point?

I think the many different ways to build hybrid HPC hardware with multi-core CPUs, GPUs, etc. The idea of a standard is also to abstract hardware in an efficient way. However, with the different hybrid approaches, application developers can no longer avoid building hardware-specific optimizations into applications.

2

> MPI'S MAJOR CHALLENGES?
> WITH DIFFERENT HYBRID APPROACHES APPLICATION DEVELOPERS CAN NO LONGER AVOID BUILDING HARDWARE-SPECIFIC APPLICATIONS.

ABOUT MPI

By **EuroCC Belgium**

The problem & the solution

Back in the early 1990s, hardware for scientific computing was very expensive. A handful of companies dominated the market and sold excellent systems, but at premium prices. A decade earlier, personal computers appeared on the market, and at NASA Thomas Sterling and Donald Becker came up with an idea that would revolutionize the world of high-performance computing (HPC).

Sterling and Becker realized that the limited power of individual personal computers (PCs) could be aggregated provided you connect them in a communication network so that they can exchange information. If in addition, you can break up your computation into many smaller computations that can be executed individually on a PC, combining the results to solve the original problem you end up with a cheap solution to a very hard problem. They built an experimental cluster according to the design and named it Beowulf after the hero of an Old-English epic poem. The name as well as the concept stuck, you can safely state that the HPC clusters to this day are sophisticated variations on that "simple" theme.

IMAGE CAPTION
2. Dr. Rolf Rabenseifner

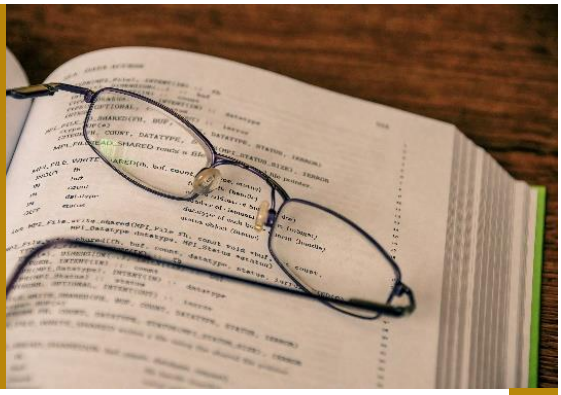
However, hardware is one thing, but using it efficiently is quite another. In the 1950s, computer users had to rewrite their code for each individual make and even type of computer. This was remedied by the advent of standardized programming languages such as FORTRAN. The first users of Beowulf architectures were faced with a very similar problem: there was no out-of-the-box solution to have computers conveniently exchange information for a scientific application. Just like a programming language provides an abstraction of the underlying hardware, an abstract model for communication was required, one that would be independent of the hardware of the interconnect as well as the low-level communication protocols. One of the tools developed in those days was Parallel Virtual Machine (PVM). In fact, the first version of PVM was written in 1989 at Oak Ridge National Laboratory and hence predates the first Beowulf cluster by half a decade. Although PVM enabled harnessing the power of this new computing paradigm it was soon eclipsed by the Message Passing Interface (MPI).

Enter MPI

The first draft specification of the MPI standard was presented at the Supercomputing conference in November 1993. After a number of changes, the first official MPI version 1.0 standard was published in June 1994. The MPI standard provided a communication model that was richer than that offered by PVM and hence MPI became the de-facto standard for distributed computing applications in the domain of science.

Since those early days, the MPI specification has gone through numerous iterations that improved both the quality of the initial version and added new features based on insights gained by experience and to accommodate improved networking hardware. The most recent version, MPI 4.0 was approved by the MPI Forum in June 2021.

> *"Dr. Rolf Rabenseifner's work in the MPI Forum since 1996 and his many contributions to the standard have provided a solid basis for his more than two decades of training activities."*
>
> **EuroCC Belgium**



3

IMAGE CAPTION
3. The MPI standards 2.1, 2.2, 3.0 (shown here), and 3.1 have been published in hardcover by HLRS, the High-Performance Computing Center Stuttgart.

The MPI specification defines language bindings for C and Fortran, and many implementations are available as open source software, e.g., Open MPI and MVAPICH, as well as commercial libraries, e.g., from Intel and HPE. Since all these implementations adhere to the MPI Forum's specification, your source code is portable, i.e., it can be built using any MPI implementation. However, typically some tuning is required to get the most out of the specific hardware platform you are using.

Typically, specifications do not really make for an exciting read, and most programmers revert to tutorials. However, the MPI specification is a rare exception to this: it is in fact very well written and a pleasure to read. The text provides information for programmers that use MPI to develop their applications but also hints to implementors of MPI libraries. This additional information is very useful for developers as well since it provides insights into optimization strategies used for those libraries. This in turn helps the programmer to make informed choices when designing the communication strategy of his application. In addition, the MPI standard is a very important – and the most accurate – source for developing MPI training materials and courses.

Dr. Rolf Rabenseifner's work in the MPI Forum since 1996 and his many contributions to the standard have provided a solid basis for his more than two decades of training activities. They are the subject of the next article in this series. Given his talent as a trainer, we have little doubt that he contributed greatly to the excellent style and content of the MPI specification text.