Observations on the Publicity and Usage of Parallel Programming Systems and Languages: A Survey Approach

Technical Report KIS No. 1 / 2007

Michael Süß, Alexander Podlich and Claudia Leopold
University of Kassel
Research Group Programming Languages / Methodologies
Wilhelmshöher Allee 73, D-34121 Kassel, Germany
{msuess, leopold}@uni-kassel.de; podlich@student.uni-kassel.de

Abstract: In this publication, we report on an online survey that was carried out among parallel programmers. More than 250 people worldwide have submitted answers to our questions, and their responses are analyzed here. Although not statistically sound, the data we provide give useful insights about which parallel programming systems and languages are known and in actual use. For instance, the collected data indicate that for our survey group MPI and (to a lesser extent) C are the most widely used parallel programming system and language, respectively.

1 Introduction

Very few data seem to exist on the usage of parallel programming systems by application programmers. Some presumptions are floating around the community, e.g.:

- The majority of parallel applications uses MPI nowadays
- Java threads is growing strong

There are few real data available to back up any of these claims, though. Studies with a limited number of participants have been conducted in the past (e.g. by Sodan [Sod05]), yet their intentions are different from ours. Books about the different parallel programming systems, such as [Leo01], show which systems are available, but have no data about which ones are in actual use.

Furthermore, different approaches to parallel programming like algorithmic skeletons or parallelizing compilers are raising a lot of interest in the scientific community, but no data about how many application programmers actually know and / or use these systems are available.

Therefore, we have conducted a non-representative survey among the programmers of parallel applications. Sec. 2 explains how the survey was carried out. Sec. 3, the main part of the paper, goes through the questions, shows the submitted answers, and comments on

their relevance. We also wanted to combine several questions with each other to show even more detailed analysis, but decided against it due to the limited amount of participants. Finally Sec. 4 sums up our results.

It is important to point out, that each and every observation we sketch out in this paper is merely a thesis. Our findings are not backed up statistically and should therefore be treated merely as data points, not as a scientific proof. We believe, however, that any data is better than no data on the subject at all, as it can at least be used to come up with hypotheses, which can be proved at a later date, possibly using a more representative study.

2 Survey Methodology

The survey was carried out online, with a simple HTML form, and a PHP script to put the submissions into a MySQL database. It was hosted on the web-server of our research group, where it can still be viewed for reference [Süß05]. Submissions were accepted from February 2nd to November 7th, 2005. Visitors of the site were asked to fill out the survey only if they are developing parallel applications. Of course, we have no way to verify that. If participants submitted their answers along with a working e-mail address, they could win one of two 50$ gift certificates from amazon.com. We also promised to deliver a report of the results.

To raise more interest for the survey and generate submissions, we have sent messages to several discussion groups, once shortly after the start of the survey and once shortly before the end:

- **Usenet:** comp.parallel, comp.parallel.mpi, comp.parallel.pvm, comp.sys.super, aus.computers.parallel, comp.programming.threads
- **Mailing Lists:** beowulf@beowulf.org, lam@lam-mpi.org, omp@openmp.org, compunity@lists.rwth-aachen.de, bspall@bsp-worldwide.org
- **Forums:** CodeGuru Forum, Intel Developer Services Forum
- **Websites:** slashdot.org

It was necessary to send out these messages, because otherwise our survey would not have been noticed by enough members of the parallel programming community to make the results meaningful. The distribution of these messages influenced our results, though. An example: we did not find a discussion forum for java threads or for algorithmic skeletons. If we had found one, the results for these systems probably would have been higher. For this reason alone, please consider all results of this survey merely as data points, as they are not in any way statistically sound! For statistical significance, we would have to sample a proportional part of the parallel programming population, and we know of no way to do so (at least not within our budget). It is for this reason, that you will not find any statistical measures applied to the data in this paper.

Other sites might have reported on the survey as well. On the Erlang mailing list, a link to the survey has been posted, which may have significantly increased the participation from this community. We got over 30 responses from Erlang programmers, whereas few other communities not specifically targeted by the survey were mentioned more than three times.

Before the data was processed and evaluated, some entries had to be disqualified. Firstly, we had four double submissions, identifiable by the submitted name and email address. Only the last submission was taken into account in these cases. Secondly, two of our students participated in the survey, although all they know about parallelization is from our classes. Since we did not want to influence the results ourselves, these submissions were taken out as well. This left us with 256 out of 262 submissions.

Of course, we cannot rule out the possibility entirely that more of our students participated or that there were more double submissions, since a valid e-mail address or name was not required on the submission form. Judging from the IP-addresses of our participants, this has at least not happened on a noticeable scale, though.

3 Survey Analysis

In this section, we present the results of the survey and add some analysis to each question. Sec. 3.1 considers languages for parallel programming, and Sec. 3.2 is about parallel programming systems. Sec. 3.3 evaluates which operating systems parallel applications are developed for, and Sec. 3.4 shows our data about which hardware platforms are targeted. In order to learn more about the participants of the survey, we have collected data on the organizations they work in (Sec. 3.5), the average time they spend on parallel programming (Sec. 3.6), and the specific field they work in (Sec. 3.7). Finally, Sec. 3.8 highlights the problems people had with parallel programming. The last question of the survey, which provided the possibility to add remarks, was seldom used by our participants, and is therefore omitted here.

3.1 Question 1 — Programming Languages

The first question was formulated as shown in Fig. 1. Results are depicted in Fig. 2. The left graph labeled *Usage* is calculated by weighting the submissions like this: *N/K* and *never* are not counted, *sometimes* is counted as is, *about half the time* with a factor of 2, *often* with a factor of 3 and *for every parallel application* with a factor of 4. The height of the bar for each language is the sum of the weighted submissions. This graph presents a view on which languages are actually used by parallel programmers. Please note that we stick to this definition of *Usage* throughout the paper, although it is of course somewhat arbitrary.

The right graph in Fig. 2 labeled *Publicity* simply counts all submissions except the votes for *N/K (not known)*. It shows, how many members of our survey group know the lan-

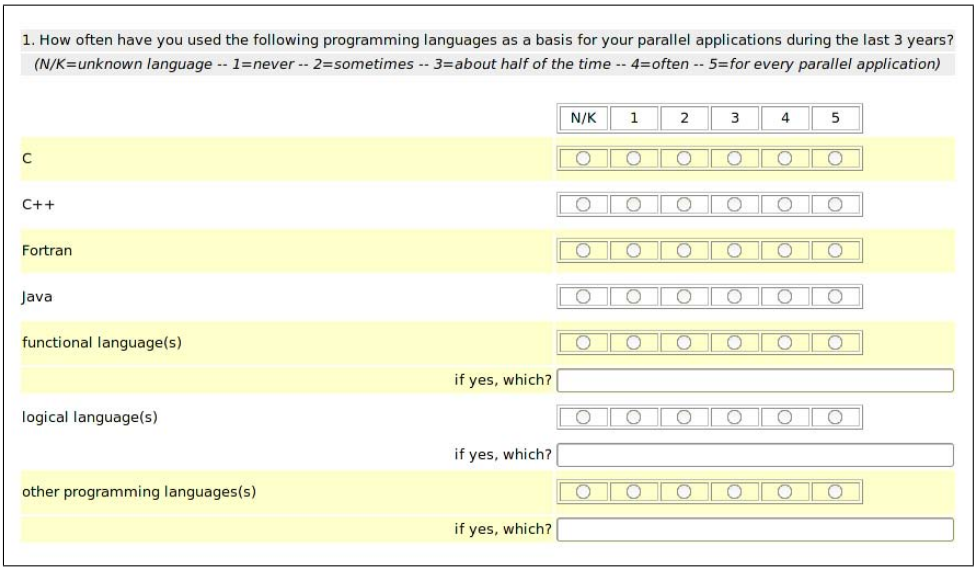

Figure 1: Survey Question 1

guage.

C comes out as the winner in both graphs for our specific survey group, followed by C++ and Fortran. Parallel programming with logical languages is obviously not very widespread (or if it is, we have not managed to reach this group). The numbers for Java are interesting. As can be seen in the publicity-graph, many participants know Java, almost as many as Fortran and more than the functional languages, yet when it comes to actually using it, Java falls behind Fortran by a great margin and is even surpassed by the functional languages. This trend can also be observed in the first part of Fig. 3, where the distribution of votes for Java is shown. Note the large number of votes for *never*, meaning the programmer has heard about the language but is not using it for parallel programming. Obviously the hype around Java has managed to make the language known to many programmers, but has not convinced too many of them to actually use it for parallel programming.

The second part of Fig. 3 is interesting as well. It is the distribution of votes for functional languages. Obviously there are two camps of parallel programmers: the ones who do not use functional languages at all (this is the larger camp and it contains the votes for *N/K* and *never*), and the camp that uses them *often* or even *for every application*. As can be seen in the graph, there are not many votes in-between the two (for *sometimes* or *half the time*), which seems to confirm the common belief that once a programmer starts using functional languages, he will never go back to using any other language.

The outcome of these graphs can be explained in part by the discussion groups we contacted for this survey, as many of the parallel programming systems for which we found and contacted mailing lists (e.g. MPI, OpenMP, PVM or POSIX threads) have C / C++ / Fortran as their base languages. We can, however, compare the results of these languages,

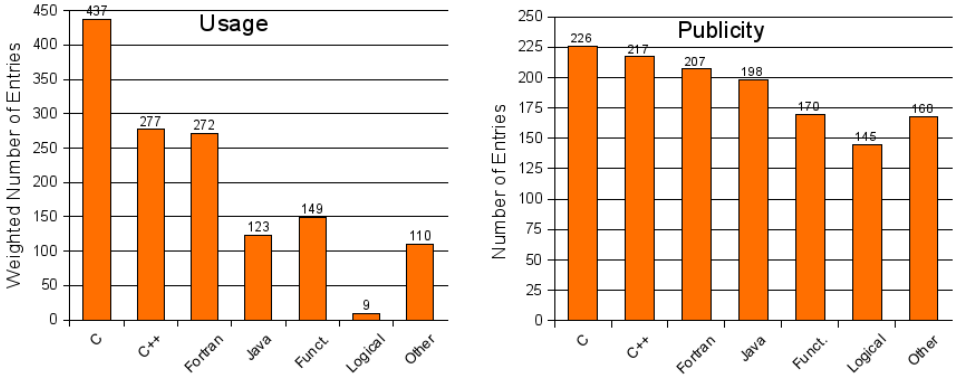

Figure 2: Parallel Programming Languages

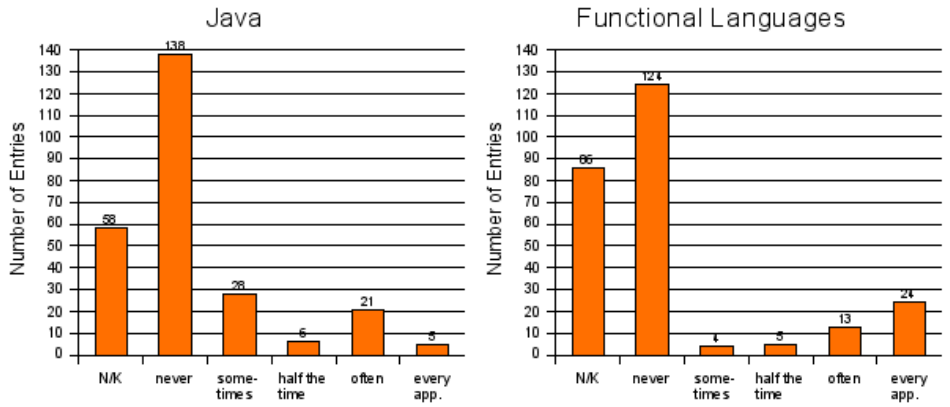

Figure 3: Parallel Programming Languages in detail

and C emerges as the clear winner between the three.

But what about the other languages? As can be seen in Fig. 1, participants had the ability to enter their own choices of languages into various input fields. Fig. 4 and 4 show, which languages people have entered. We only present languages that were submitted more than three times, since otherwise the graph would have become too big. We have also decided against dividing the graph into categories (functional, logical, others), for the sake of clarity and because the distinction does not lead to any new insights here.

We have already told you a possible reason for the relatively high numbers for Erlang when compared to the other languages in Sec. 2. What can be deduced from these numbers, though, is that Erlang has an active user community.

Please note, that it would be unfair to compare the languages explicitly mentioned in question one with the other languages entered in textboxes. If we would have explicitly asked for e.g. Python in this question, it would have been voted higher. Therefore, the languages

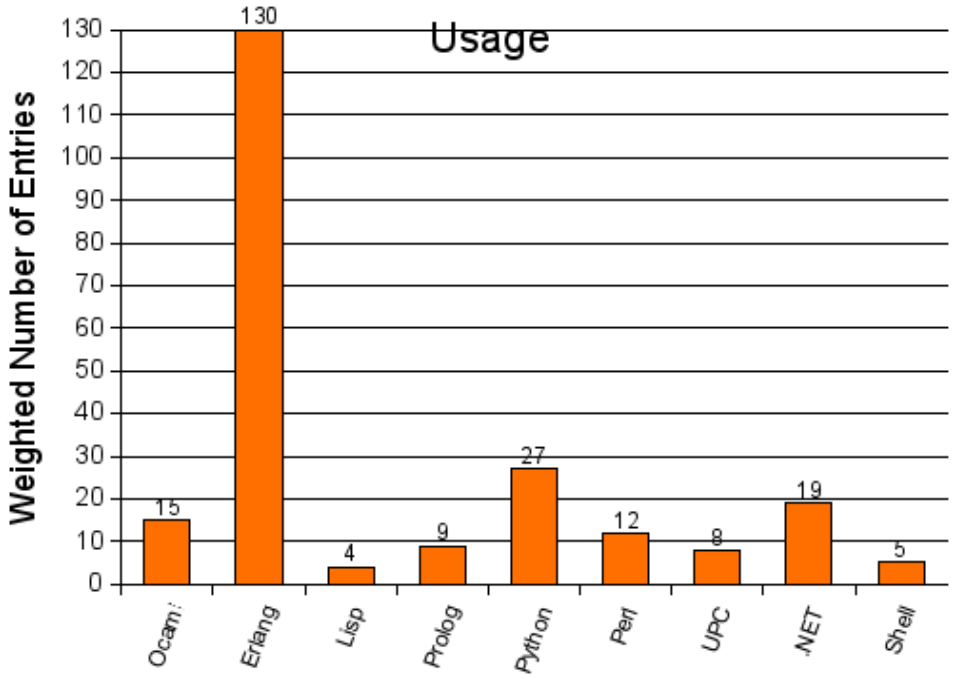

Figure 4: Usage of Other Languages for Parallel Programming

depicted in Figures 2 and 4 can only be compared fairly to the languages in their respective figures.

All of these graphs together give us at least some indications about our initial question, which languages are known and in use for parallel programming today. All of C / C++ / Fortran and Java are used and known to the community at large, while at least a few other languages (starting with Erlang, but also including languages like Perl and Python) are on the radar of some parallel programmers.

3.2 Question 2 — Parallel Programming Systems

The second question was phrased as:

How often have you used the following parallel programming systems during the last 3 years?

The possible answers as well as their respective number of submissions (weighted exactly as explained in Sec. 3.1) are shown in Fig. 6 and 7.

The results of the usage graph seem pretty clear: MPI wins by a great margin, for our survey group, followed by POSIX Threads and OpenMP. We suspect that 5 years ago, the

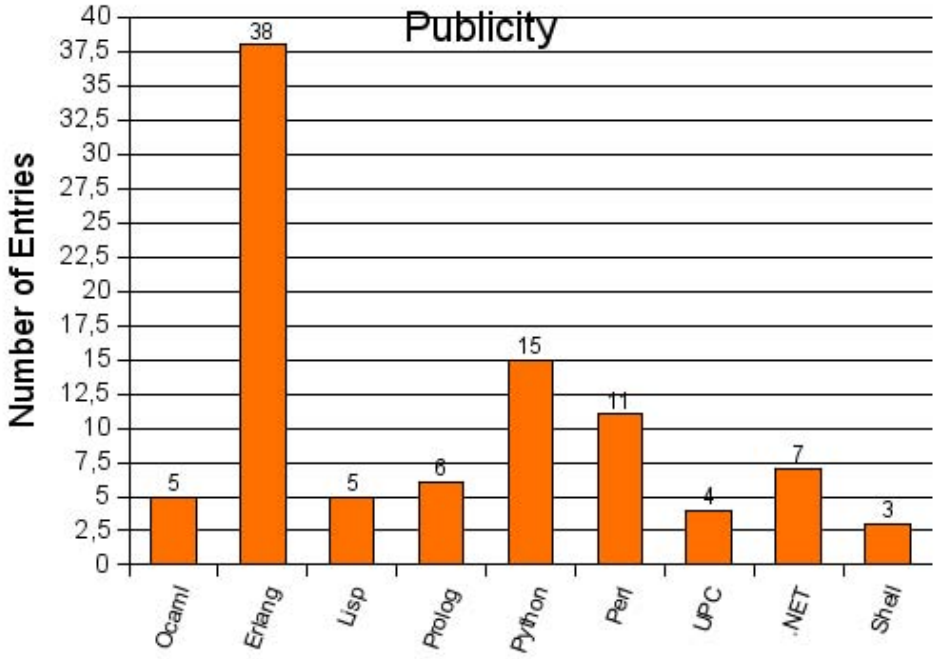

Figure 5: Publicity of Other Languages for Parallel Programming

numbers for PVM would have been much higher, but for now the battle for the predominant message passing system seems to have been settled in favour of MPI. The high usage numbers for MPI are due to a large number of exclusive users, as can be seen in Fig. 8. There are other systems which are also widely known (see Fig. 7), but all of them fall behind in actual usage.

For shared memory programming, POSIX Threads and OpenMP are close together, and it will be interesting to see, whether or not one of the systems manages to become the dominant one, especially since their usage distribution is about equal (see the bottom of Fig. 8). High Performance Fortran (HPF) and BSP do not seem to play a significant role for shared memory programming today, for our survey group.

The picture for Java threads is similar to what has been observed in Sec. 3.1. The system is widely known, yet seldom used.

We were somewhat surprised by the low numbers for algorithmic skeletons. These are generating a lot of attention in the research community, yet their publicity and even more usage are low, in our survey group. We were not even able to create a thesis about which system is the most widely known, because all three submitted systems (P3L, DatTeL, FreePOOMA) have been mentioned only once!

A similar observation can be made for automatically parallelizing compilers. Compared to other systems, their publicity and usage are relatively low. This is surprising, since they are

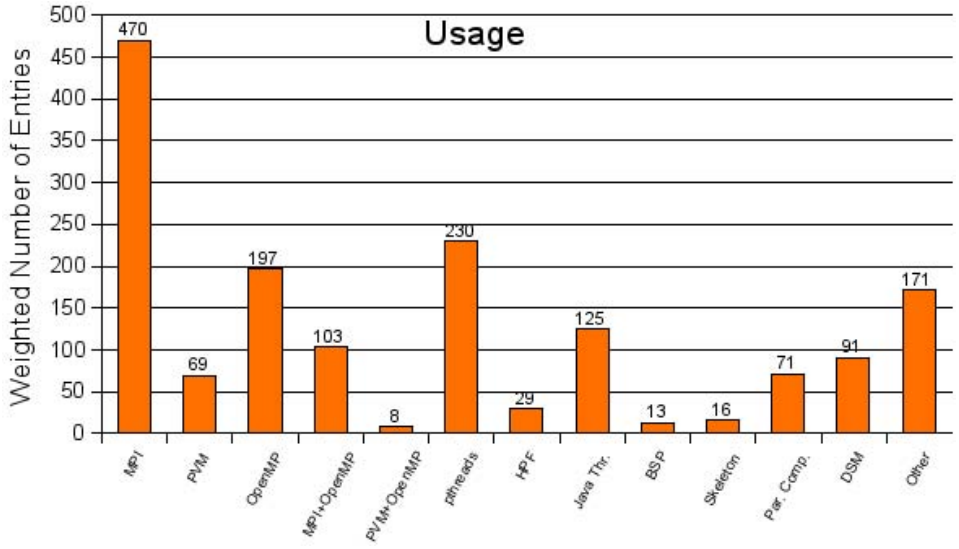

Figure 6: Usage of Parallel Programming Systems

commonly available by now and all it takes to use them is setting a switch in the command line. A detailed list of the parallelizing compilers submitted is shown in Fig. 9. Note that, once again, only compilers that were submitted more than three times are depicted.

Distributed shared memory systems seem to suffer a similar fate in our survey group. They are heavily researched, yet not widely known or used. Only SGI Altix and IBM pSeries have been mentioned more than three times, and both are actually hardware architectures with built-in support for distributed shared memory. No pure software solutions were mentioned more than three times, which indicates that they are not widely accepted or used in our survey group.

The other parallel programming systems submitted include a wide variety of systems, yet only Erlang (26 submissions, accumulating to a usage of 89) and .NET (four submissions with a usage of 10) managed to be mentioned more than three times. Noteworthy is the fact that Erlang is one of the very few programming languages for which parallelism is an integral part of the language, and it therefore has high submissions for both questions one and two. When interpreting the numbers, please keep in mind that systems to be entered by users can only be compared fairly among themselves, and not to systems explicitly asked for in question two.

Using these figures and observations, let us reconsider our initial question about the usage and publicity of parallel programming systems: It is safe to conclude that MPI, POSIX Threads, OpenMP, and Java Threads are widely known and used for parallel programming, with MPI being the most popular programming system from our submissions (but remember that we have not sampled a proportional part of the parallel programming population!). Algorithmic skeletons, parallelizing compilers and distributed shared memory systems do not seem to be widely accepted and used (or, if they are, we have not managed

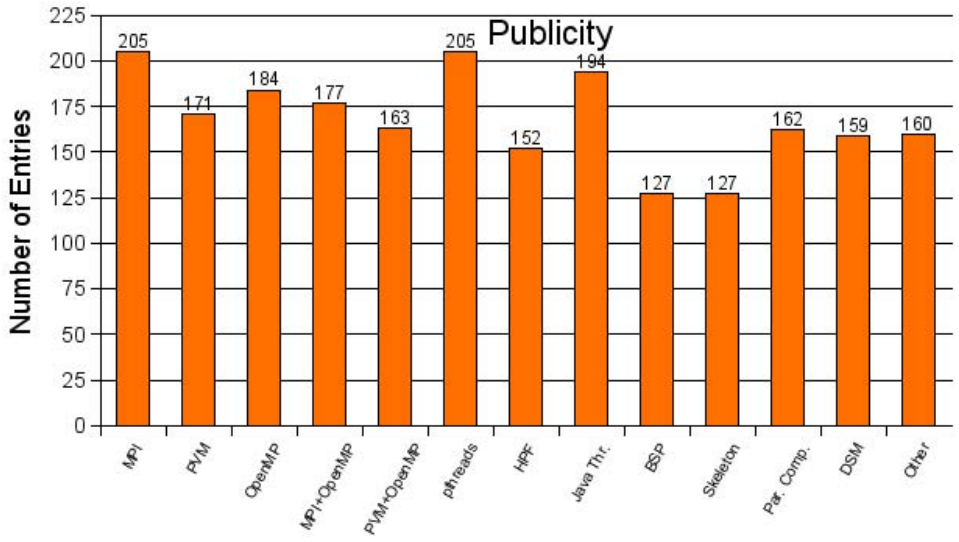

Figure 7: Publicity of Parallel Programming Systems

to reach their user communities). Erlang has significant usage numbers, but we can not fairly judge its popularity, from our survey methodology.

3.3 Question 3 — Operating Systems

Our next question was:

- What operating systems are your parallel applications intended for?

The possible answering options and the submissions are shown in Fig. 10. Note that only radio buttons were provided for the answers, therefore it is not possible to distinguish between usage and publicity here as for questions one and two.

Linux is the dominant operating system for this survey by a wide margin, followed by Solaris and Windows. It seems safe to conclude that most parallel applications are developed for a flavour of UNIX, as even the operating systems that were hand-submitted consist mainly of variants of UNIX (as can be seen in the right part of Fig. 10). It would be interesting to see if these numbers change in time with the introduction of Windows for Supercomputers and the increase in parallel systems on the desktop due to the advent of multi-core processors.

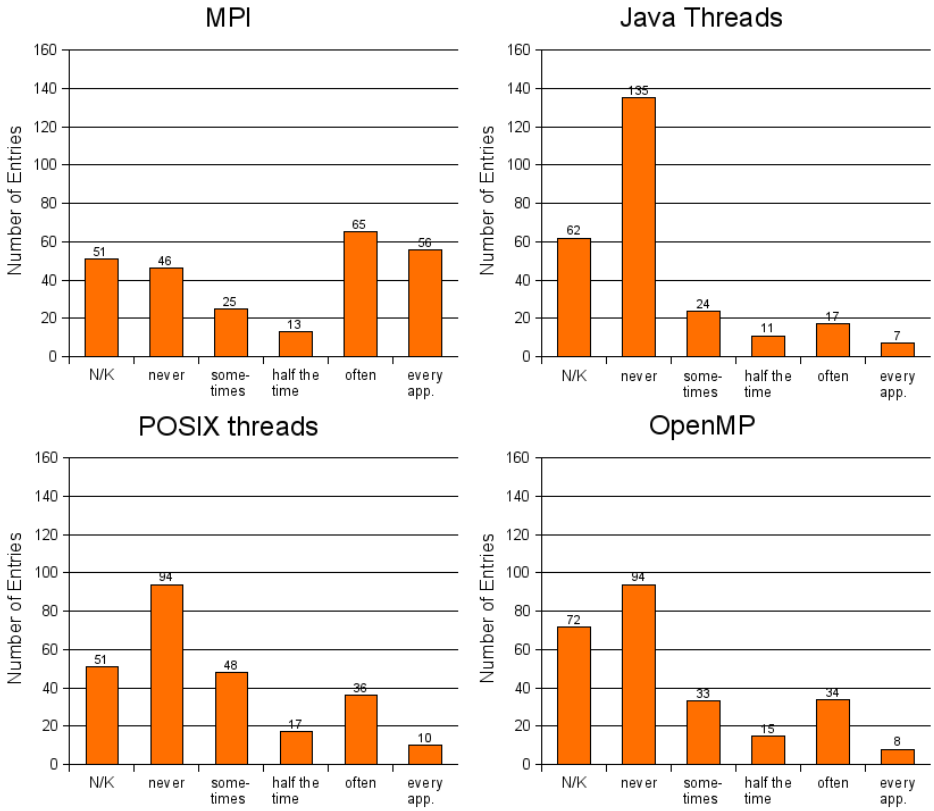

Figure 8: Parallel Programming Systems in detail

3.4 Question 4 — Hardware Platforms

The next question we wanted to investigate was:

- What hardware platforms are your parallel applications intended for?

Possible answers were shared memory architectures (e.g. SMPs, NUMAs), distributed memory architectures (e.g. clusters) and grids. Multiple answers were allowed. Distributed memory architectures won our survey with 199 submissions, followed by shared memory architectures with 160 submissions. Grids came in at a distant third place with only 47 submissions.

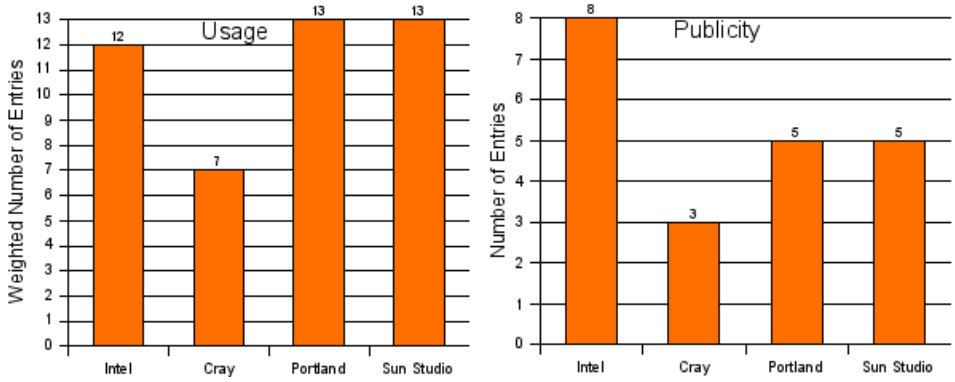

Figure 9: Parallelizing Compilers

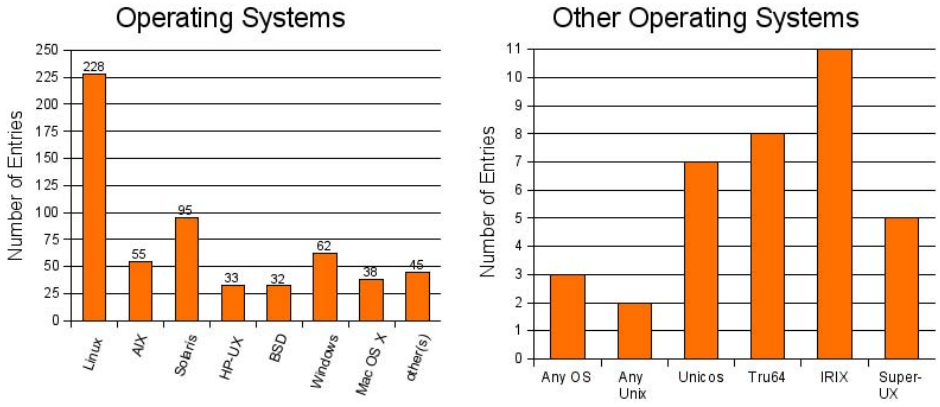

Figure 10: Operating Systems

3.5 Question 5 — Organization

In this question we wanted to know:

- In what kind of organization are you developing parallel applications?

Answers were: University (122 submissions), Company (88 submissions), Other Research Institute and Other Organization (6 submissions total). This question is most useful to put the results we have gathered so far into perspective, as obviously our survey is biased in favour of systems used at universities.

3.6 Question 6 — Time spent

Our next question was:

- How much of your programming time is spent on the development of parallel applications?

Answers were: $0 - 20\%$ (81 submissions), $20 - 40\%$ (48 submissions), $40 - 60\%$ (46 submissions), $60 - 80\%$ (37 submissions) and $80 - 100\%$ (no submission). Multiple answers were not allowed. There either are no people who are employed to work on parallel projects full-time, or we have not managed to reach them.

3.7 Question 7 — Fields

In this question we wanted to know:

- For which scientific or economic fields are you developing parallel applications?

We did not define these fields beforehand, but rather evaluated what people have entered in a text-field. This of course led to a wide variety of fields. We therefore classified them into bigger categories and came up with the following list: Physics (41 submissions), Computer Science (31 submissions), Biology (20 submissions), Chemistry (19 submissions), Mathematics (17 submissions), Engineering (14 submissions) and Astronomy (10 submissions). All other fields had less than ten submissions and are therefore left out here. Once again, our survey results are obviously very biased in favour of the systems used for either engineering or the natural sciences - please keep that in mind while putting them into perspective.

3.8 Question 8 — Problems

In another question, participants were asked to enter their problems regarding parallel programming in a text field. The question was formulated as:

- What major problems do you see with the currently available parallel programming systems?

For the analysis, we classified the entered problems into categories, of which six were entered more than ten times: parallelization overhead (41 submissions), need for better debugging tools (36 submissions), need for bug-free compilers or libraries (22 submissions), lack of support by a community/vendor or lack of documentation (15 submissions), need

for higher-level development tools (11 submissions), and problems with special hardware (10 submissions).

It turns out, that the biggest problem with the currently available parallel programming systems is parallelization overhead. Some participants went on to mention that writing parallel programs is very hard in the comments section of the survey.

4 Summary

This paper has reported on a survey that we have conducted among the programmers of parallel applications. 256 valid answers to nine questions were submitted, of which eight have been analyzed in this paper. The following observations were made for our survey group:

- C, C++ and Fortran are well known and widely used in the parallel programming community, the most widely used language in our survey group was C.
- Some other languages (starting with Erlang, but also including languages like Perl and Python) are known and used by at least a few programmers.
- Java is well-known, but not frequently used.
- MPI appears to be the most popular parallel programming system, followed by POSIX Threads, OpenMP and Java Threads.
- Algorithmic skeletons, parallelizing compilers and distributed shared memory systems do not seem to be widely accepted and used.
- The target platform for most parallel applications is a variant of UNIX, predominantly Linux.
- Both distributed memory architectures and shared memory architectures are used in the parallel programming community, grids are not yet common.
- The major problems with parallel programming for our survey group were parallelization overhead, need for better debugging tools and the need for bug-free compilers or libraries.
- Parallel programming was still felt to be hard by many respondents.

It must be kept in mind (once again) that the data presented is not statistically backed up and therefore only valid for our specific survey group!

Acknowledgements

We thank Holger Bischof for extensive comments and improvements prior to the launch of the survey, Björn Knafla for enlightening discussions, and Raffaele Biscosi for taking care of the server and database. Last but not least, we thank all the participants of the survey: Without your valuable time and commitment, this work would not have been possible!

References

[Leo01] Claudia Leopold. *Parallel And Distributed Computing: A Survey of Models, Paradigms, and Approaches*. John Wiley & Sons, 2001.

[Sod05] Angela C. Sodan. Message-Passing and Shared-Data Programming Models - Wish vs. Reality. In *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS'05)*, pages 131–139, Washington, DC, USA, 2005. IEEE Computer Society.

[Süß05] Michael Süß. Parallel Programming Survey. `http://www.plm.eecs.uni-kassel.de/parasurvey/`, February 2005.