exampi-2020.pdf

MPI International Survey Report

Atsushi Hori
Riken Center for Computational Science
Email: ahori@riken.jp

Emmanuel Jeannot
Inria
Email: emmanuel.jeannot@inria.fr

George Bosilca
The University of Tennessee, Knoxville
Email: bosilca@icl.utk.edu

Balazs Gerofi
Riken Center for Computational Science
Email: bgerofi@riken.jp

Jie Yin
Riken Center for Computational Science
Email: jie.yin@riken.jp

Takahiro Ogura
Riken Center for Computational Science
Email: t-ogura@riken.jp

Yutaka Ishikawa
Riken Center for Computational Science
Email: yutaka.ishikawa@riken.jp

***Abstract*—The Message Passing Interface (MPI) plays a crucial part in the parallel computing ecosystem, a driving force behind many of the high-performance computing (HPC) successes. To maintain its relevance to the user community—and in particular to the growing HPC community at large—the MPI standard needs to identify and understand the MPI users' status, concerns, dissatisfaction, and expectations, and adapt accordingly. This questionnaire survey was conducted starting from February 2019 by using two online questionnaire frameworks, and more than 850 answers from 42 countries has gathered at the time of this writing. Some of preceding work in surveying MPI uses are questionnaire surveys like ours, while others are conducted either by analyzing MPI programs to reveal static behavior or by analyzing dynamic runtime behavior of MPI jobs by using profiling tools. Our survey is different from the other questionnaire survey in terms of geographically wide-spread and the much larger number of participants. As a result it is possible to illustrate the current status of MPI users more accurately and with a wider geographical distribution. In this report, we will show some interesting findings, comparing the results with preceding studies when possible.**

I. Background

Existing studies on MPI uses are focused on a restricted target domain, such as the Exascale Computing Project (ECP) [1] study conducted in 2017 [2] that focused on MPI usage in the context of ECP applications; and/or are generally geographically constrained to a single laboratory, funding agency or at best, country. As such they provide sporadic, disconnected views on the real uses of MPI across the world. Interestingly enough, and mostly by coincidence, simultaneously with the ECP study another survey was conducted in Japan targeting HPCI [3] users which included several questions asking about MPI [4]. HPCI is an infrastructure for HPC users in Japan connecting major supercomputers owned by universities and governmental research institutes. If both questionnaire surveys would have the same questions, we could have compared the answers to reveal the differences between US and Japan MPI user communities. Unfortunately, a single question was similar in both studies, limiting the correlations between the two surveys.

Such studies inspired us to conduct a larger study, reaching across many diverse community of MPI users. Unlike it's predecessors this study not only focused on high-end HPC, but targeted a wider audience and involved a larger spectrum of geographically distinct users. Since MPI has been a widely used vehicle for high-performance computing for decades, this larger-scale questionnaire survey would be beneficial not only for deciding the future direction of MPI, but also for understanding the feature differences of MPI users among countries and/or regions of the world.

Our project was started to conduct such a study as a project at JLESC [5] which is an international research collaboration framework. The international nature of this survey matches the concept of JLESC. Co-authors are a member of JLESC and responsible for the country and/or region where they belong. For the design of the questionnaire, we consulted two social scientists, Prof. Marshall Scott Poole at Illinois Univ., and Prof. Iftekhar Ahmed at Univ. of North Texas, participating JLESC workshops to investigate how researchers can collaborate in the JLESC framework.

TABLE I
SURVEY COMPARISON

	Field of Interests	Geographic Target	Number of Questions	Number of Participants
ECP	Exascale Computing	US	64 (max.)	77
HPCI	Computing Environment	Japan	75 (max.)	105
Ours	MPI (w/o MPI-IO)	Earth	30	851

To give an order of comparison with preceding studies, our MPI International Survey, ECP survey, and HPCI survey are summarized in Table I.

II. Related Work

So far, there are three MPI survey classes; [Q] questionnaire survey asking questions to MPI users such as ours and the ECP survey, [S] analyzing MPI program statically, and [R] analyzing MPI behavior at runtime by using a profiling tool.

Apparently our survey and the ECP survey are categorized in [Q]. The surveys categorized as [S] are [6], [7]. The [R] class surveys are [8], [9].

In the [S] category, [6] statically investigated 110 open-source MPI programs. [7] investigated 14 MPI programs chosen from the ECP Proxy Applications Suite 2.0 [10].

In the [R] category, [8] collected and analyzed the runtime behavior by running more than 100K MPI jobs. [9] analyzed behavior of DOE Mini-apps based on the trace data which DOE made public.

The target of the questionnaire surveys are MPI users, the target of [S] is MPI programs, and the target of [R] is MPI jobs. In spite of these target differences, we dare to compare some results of those non-questionnaire-based surveys and ours in the following sections if appropriate.

III. Survey

The social scientists suggested that the number of questions must be limited around 30 not to loose the participants' concentrations. This number is much smaller that those of ECP and HPCI surveys. So, we focused on MPI communications and excluded MPI-IO related questions. Additionally, we designed the questionnaire for participants can easily answer questions as much as possible, and the questions to force participants doing some work, such as counting the lines of code, are eliminated.

ECP and our questionnaires are implemented by using Google Forms. Later in our project, we also implemented the same questionnaire by using Microsoft Forms for those who cannot access Google Forms. All graphs in this paper were generated by Python and R programs from the CSV files obtained by using Google Forms and Microsoft Forms.

The draft questionnaire was tested by MPI Forum members, and researchers at Inria and Riken Center for Computational Science (R-CCS). We also interviewed with Dr. Kento Sato at R-CCS. The questionnaire was started distributing and receiving answers from February 17, 2019 and it is still open. The most recent answer was obtained at June 22, 2020. All questions, their choices, and abbreviations of the choices are listed in Appendix A.

We announced this survey by posting emails to major mailing lists such as hpc-annouce. Soon after started, we realized the number of responses was not as many as we expected. Hence, we asked people who can reach local regions. Distribution using local mailing-lists worked much better than that of major mailing lists. Fig. 1 shows the transition of the number of answers at the first three months. As shown there are several steps and those steps came out after asking local distribution.

This local distribution strategy worked very well on some regions but did not work universally. Table II shows the number of participants of top 11 countries. In this report, the countries having more than 50 participants are called *major countries (regions)* and are the objects of cross-tab analysis. There is a trade-off between the number of participants of each major country and the number of the major countries in

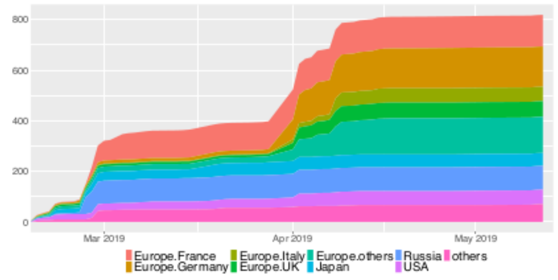

Fig. 1. Time series in first 90 days

the cross-tab analysis. The threshold of 50 participants was selected to balance this trade-off but this is not our intention of saying 50 is enough. Hence, some cross-tab analysis may not be reliable enough.

Comparing with Table III listing the top 10 countries in the performance share in the Top500 [11], the three major countries, USA, China and Japan, are not even the top 5 in out survey. Especially China has only 18 participants including Taiwan (2). We tried to increase the number of participants of those countries, especially China, as much as we could, but we failed.

TABLE II
Top 11 Countries of Participants

#	Country	#Ans	[%]
1	Germany	159	18.7
2	France	125	14.7
3	Russia	94	11.1
4	UK	67	7.9
5	Japan	64	7.5
6	USA	58	6.8
7	Italy	57	6.6
8	Switzerland	40	5.8
9	South Korea	27	3.2
10	Austria	26	3.1
11	China (+Taiwan)	18	2.1

42 countries, 851 participants

TABLE III
Top500 Performance Share (June 2020)

#	Country	[%]
1	USA	28.2
2	China	25.6
3	Japan	23.9
4	Italy	4.0
5	France	3.6
6	Germany	3.1
7	UK	1.4
8	Canada	1.3
9	Netherlands	1.1
10	Switzerland	1.1

Fig. 2 shows the graph of Q1 asking participants' occupation. As shown, roughly 80% of participants are working at universities or governmental research institutes.

Fig. 3 shows the result of asking "Which fields are you mostly working in?" Generally speaking, most participants are working on numerical applications and/or libraries. In Japan and US, the percentages of parallel languages and OS/runtimes are higher than the other countries and regions.

IV. Comparison with the ECP Survey

Although the ECP questionnaire and our questionnaire were designed independently, there are several questions quite similar. It is also assumed that some of the participants of our survey also participated in the ECP survey. However, significant differences between them can be found. Before going

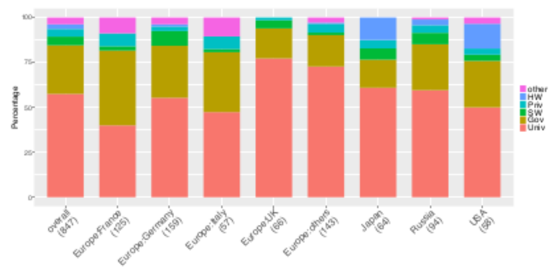

Fig. 2. Q1: Occupations *(single)*

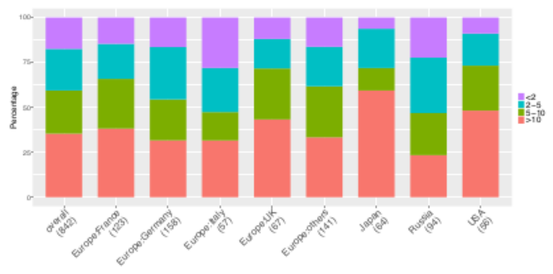

Fig. 5. Q6: MPI Experience *(single)*

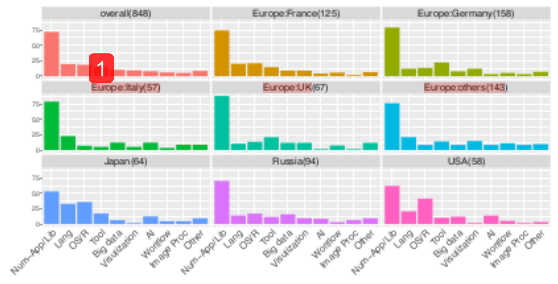

Fig. 3. Q7: Working Fields *(multiple)*

into the details, we clarify some profiles of the participants in our survey.

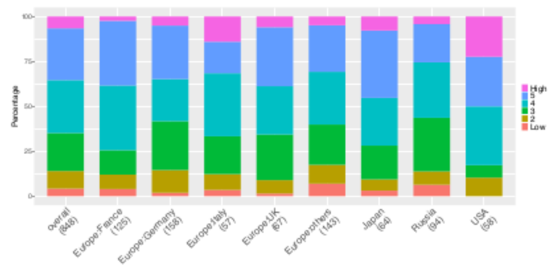

Fig. 4. Q3: Self assessment of MPI Skill *(single)*

We assumed that the participants in the ECP survey have more experiences on MPI programming than the participants of ours. Fig. 4 shows the results of Q3 asking to rate the participants MPI skill by themselves in our survey. In the US case [1] (right most bar), almost half of participants rate 5 or 6 (6 is the highest). This percentage in having high MPI skill (larger than or equal to 5) in the US is the highest among the other countries and nobody in US marked the lowest skill.

Fig. 5 shows the more interesting result. This graph is the result of asking "How long have you been writing MPI programs?" and the choices are 'more than 10 years' (denoted as '>10'), 'between 5 and 10 years' (denoted as '5-10'), 'between 2 and 5 years' (denoted as '2-5') and 'less than 2 years' (denoted as '<2'). Interestingly 9% of US participants have less than 2 years MPI experience, but they do no rank their MPI expertise the lowest (Fig. 4). Japan followed by USA has the highest percentage in more than 10 years and has the lowest percentage in less than 2 years experience. Russia followed by Italy has the highest percentage in less than 5 years experience (including the less than 2 years experience case).

We chose three questions from both of out survey and ECP survey that are very similar and thus the results are comparable (Table IV). We will discuss those results in the following subsections.

TABLE IV
COMPARABLE QUESTIONS

A. Layering MPI calls	
Q21: In most of your programs, do you pack MPI function calls into their own file or files to have your own abstraction layer for communication? *(single)*	Q22: Do you have an abstraction layer that hides the MPI calls? Or do most of your developers write MPI calls directly? *(single)*
B. Using MPI Aspects	
Q17: What aspects of the MPI standard do you use in your program in its current form? *(multiple)*	Q15: What aspects of the MPI standard do you use in your application in its current form? *(multiple)*
C. Multi-threading	
Q18: Which MPI thread support are you using? *(multiple)*	Q59: Which MPI threading option are you using? *(single)*
Our Survey	ECP Survey

A. *Layering MPI calls*

Fig. 6 shows the result of our survey and Table V shows the comparison between ours and ECP survey. In the ECP survey, the participants are categorized into two groups; application development (AD) and system technology (ST). It is interesting that the percentage of the participants having MPI layer(s) in our survey is roughly 50% even in the US, whilst the ratio of yes and no is approximately 6 : 4 in the ECP survey. Having a closer look at Fig. 6, the answer, 'No, my program is too small to do that,' dominates in Russia. In the other countries, the participants having a packing layer occupies 40-50%.

[1]Note that we asked the workplaces in past 5 years, not asking nationalities.

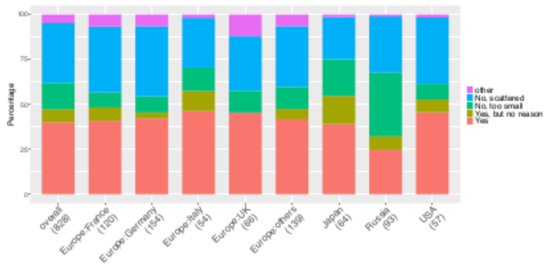

Fig. 6. Q21: Layering MPI calls *(single)*

TABLE V
LAYERING MPI CALLS

Choice		Our Survey [%]		ECP [%]		
		overall	USA	AD	ST	AD+ST
Yes	-	40	46			
	no reason	7	7			
	(sum)	47	53	79	46	62
No	too small	15	9			
	-	33	37			
	(sum)	48	46	21	54	38
Other	-	5	2	-	-	-

B. *Using MPI Aspects*

The Q35 in the ECP survey and Q17 in our survey are almost equivalent questions, although choices are somewhat different. Fig. 7 shows the result of our survey and Table VI shows the comparison between ECP's and ours. As shown in Fig.7, the using aspects can be categorixed in three groups; A) more frequently used (point-to-point and collectives), B) second frequently used ('Datatypes', 'with OpenMP', 'Communicator', and 'One-sided'), and C) less frequently used ('PMPI', 'Persistent', and 'dyn. process' (dynamic process). It should be noted that all these less frequently used features were already introduced and standardized in MPI 2.2 which was release in 2009. Despite the 10-year appearance, those features failed to get popular.

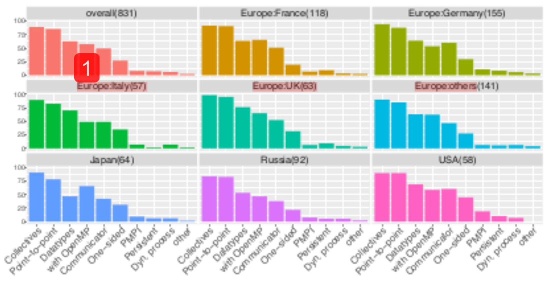

Fig. 7. Q17: Using MPI Aspects *(multiple)*

Most notable difference between our survey and the ECP survey is the datatype (Table VI). The percentage in datatype in ECP is 23% and the percentage in our survey is more than 60% in both overall and USA. In the ECP survey, there are three questions asking using MPI aspects in different usage scenarios; current, exascale, and performance critical. In all three questions, the percentages of using datatype are low.

TABLE VI
USING MPI ASPECTS

Choice	Our Survey [%]		ECP [%]		
	overall	USA	AD	ST	AD+ST
Collectives	89	90	86	75	80
Point-to-point	85	90	96	79	88
Datatype	63	69	25	21	23
Communicator	50	60	68	54	61
One-sided (RMA)	27	45	36	7	21
PMPI	8	19	11	0	14

Fig. 8 is the addition. This graph is a heatmap of the cross-tab analysis between Q3 asking MPI skill and Q16 asking unknown MPI features. The darker the color of a cell, the higher the frequency (Legend combined with color bar can be found at the bottom of the figure. The numbers in the legend cells are percentages). Less frequent rows (skill low (1) and skill high (6)) in this figure are omitted to increase readability. Natural thinking may conclude that the higher the MPI skill, lesser the unknown MPI features. The result is quite interesting, the less used features in Fig. 7, PMPI, persistent, and dynamic process, are almost independent from the MPI skill. This may indicate that the MPI standard is very profound. Even the most basic send/receive functions, although they look very simple and natural, they require deep knowledge such as possibility of deadlock, timing of buffer access, blocking and non-blocking, and so on.

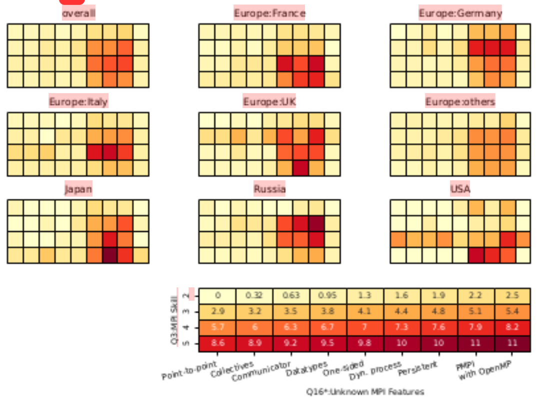

Fig. 8. Q3-Q16: MPI Skill *(single)* and Unknown MPI Features *(multiple)*

Fig. 9 is the result of Q27 asking "What MPI feature(s) are NOT useful for your application?" Although many participants think MPI has no "not useful features", fairly amount of participants think the dynamic process feature not useful. Thinking of the fact that the dynamic process feature is not used by the most participants (Q17, Fig. 7), this result is quite interesting. This tendency is also reported in [6].

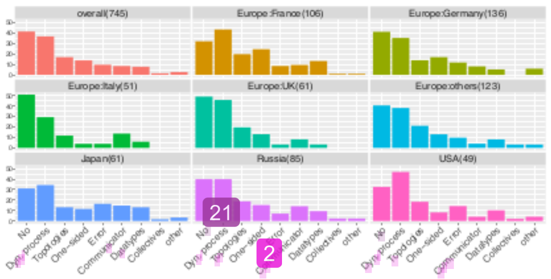

Fig. 9. Q27: Useless Features *(multiple)*

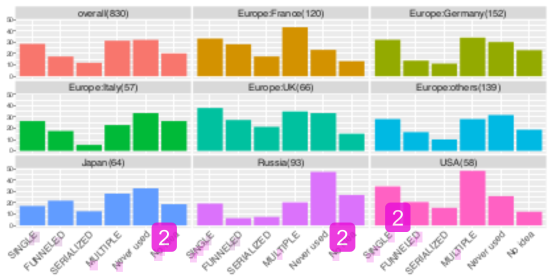

Fig. 10. Q18: Multi-threading *(multiple)*

The dynamic process feature is on the border of process management and communication, since the process creation itself is obviously out of the scope of the MPI standard, while the communication between the existing (MPI) processes and newly create (MPI) processes must be defined in the standard. Indeed, the implementation of dynamic process creation spreads many parts of a computing system: MPI library, process manager, job scheduling system and system operation. This complexity might make the use of the dynamic process creation hard and impractical for MPI users.

C. *Multi-threading*

The difference between our survey and the ECP survey can be found on the question asking multi-thread support. Fig. 10 shows the result of our survey and Table VII shows the difference. Note that our question is multiple-answer and the ECP question is single-answer. In both surveys (USA and ECP), the percentage of using `MULTIPLE` is the highest excepting the AD case. However, the percentage of the choice 'I don't know' is fairly larger than ours. This may sound contradictory because the ECP participants would be more experienced MPI users.

TABLE VII
MULTI-THREADING

Choice	Our Survey [%]		ECP *(single)* [%]		
	overall	USA	AD	ST	AD+ST
SINGLE	29	22	(no corresponding choice)		
FUNNELED	18	13	18	18	18
SERIALIZED	12	10	18	18	18
MULTIPLE	22	31	18	32	25
never used	23	16	(no corresponding choice)		
not know	14	8	25	25	25

The usage of `MULTIPLE` in US is also the highest among the major regions (Fig. 10). France and Germany have the same trend. In Italy, Japan, Russia and the other countries in Europe, the percentages of 'I don't know' are the highest. In UK, the percentage of using `SINGLE` is the highest.

Remember this is a multiple answer question. Table VIII shows the top 7 of the percentages of raw answers (combined answers) of the overall. These top 7 percentages occupy about 85% in total. Nearly half participants answered 'never used' or 'no idea.' The numbers in parenthesis in this table are the percentage of participants excluding those who answered 'never used' or 'no idea.' Half of threading-aware participants are using `SINGLE` and/or `MULTIPLE`. Although many participants ignore the thread mode, however, some participants use a particular thread support (`SINGLE` or `MULTIPLE`) and some other participants select one of supported thread capabilities positively.

TABLE VIII
MULTI-THREADING - RAW ANSWERS

Threading Support	Overall Percentage
'never used' + 'no idea'	48
`MULTIPLE`	12 (23)
`SINGLE, MULTIPLE`	8 (16)
`SINGLE`	7 (14)
`SINGLE, FUNNELED, SERIALIZED, MULTIPLE`	4 (8)
`SINGLE, FUNNELED`	3 (7)
`SERIALIZED`	3 (5)

Numbers in parenthesis are percentages excluding 'never used' and 'no ides'

In [8], approximately 75% of their target executables (not number of jobs) on Mira (total of 68) are using `SINGLE`, 15% use `FUNNELED` and 4% use `MULTIPLE`. In [6], approximately 60% of their target programs use `FUNNELED`, 30% use `MULTIPLE`, 20% use `SINGLE` and only few percent use `SERIALIZED`. Thus, the thread support usage varies on each survey and further investigation is needed to state a result.

V. OTHER FINDINGS

A. *MPI Implementations*

Q13 is asking "why did you choose the MPI implementation(s)" and its answers are shown in Fig. 11. The highest percentage of US participants selected 'Familiar.' Many Italian participants also selected 'Familiar.' Many Russian participants selected 'No reason.' The largest part of UK and Germany participants selected 'No choice.' In general, more than half of participants excepting US select MPI implementation(s) without any reason nor freedom of choice.

B. *Learning MPI*

Fig. 12 shows the percentages of how participants learned MPI. In this graph, 'Other lec.' indicates the choice 'Other lectures or tutorials (workplace, conference)'. The UK and Russia

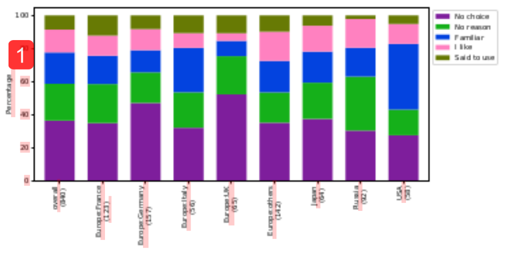

Fig. 11. Q13: Choosing MPI Implementations *(single)*

participants preferred to learn by Internet. The participants of Germany and other European countries preferred to have other lectures. The percentage of reading 'Books' in US is the highest. Taking a look at the other answers, 18 participants learned by reading existing code and 8 participants learned by doing[2].

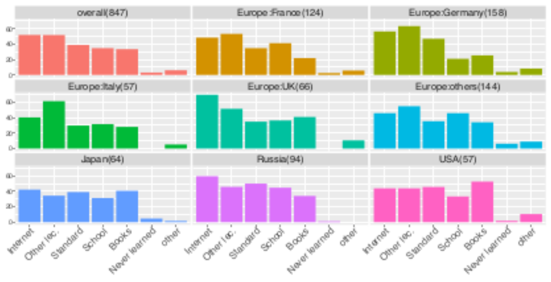

Fig. 12. Q10: Learning MPI *(multiple)*

Fig. 13 shows the graph of Q9, asking if participants have read the MPI standard document. Not surprisingly, around 60% of participants, independent from countries, read the standard partly. Most interesting country is UK, where the percentage of participants reading all document and, at the same time, the percentage of the answer 'No, and I will not read it' are the highest among the countries and regions.

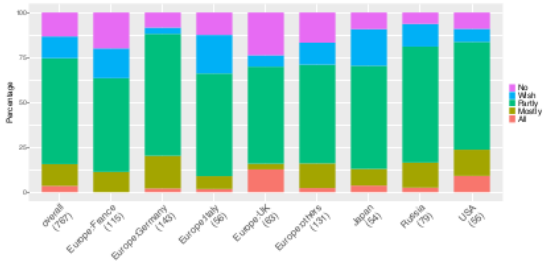

Fig. 13. Q9: Reading MPI Standard *(single)*

Now let's see how people are writing MPI programs. Fig. 14 shows the percentages of Q14 asking "How do you check MPI specifications when your are writing MPI programs?" Most users are checking MPI specifications by reading online documentations (e.g., man pages), searching Internet and reading the standard. As shown in the previous figure (Fig. 13), users are reading the standard partly because of checking the MPI specifications.

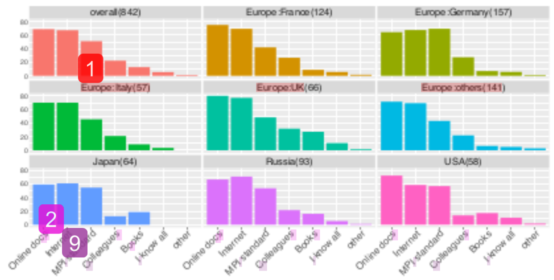

Fig. 14. Q14: Checking Specification *(multiple)*

Currently the MPI standard documents are available in PDF format and hardcover books [12]. There are some MPI tutorial web sites ([13] as an example). [14] pointed out most of such online documents are out-dated and not thorough. It might be a good idea for MPI Forum to officially publish some supplemental document(s) in another online hypertext form.

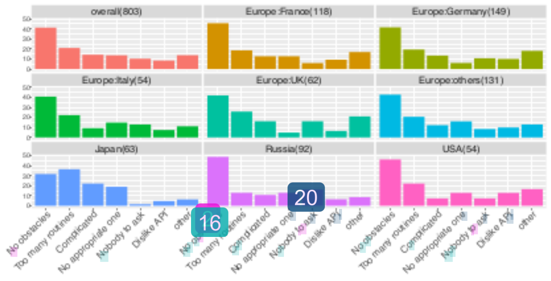

Fig. 15. Q19: Learning Obstacles *(multiple)*

Generally speaking, there can be a case where a particular question in a survey may ignite for participants to explode their dissatisfaction by writing messages into a free text field. In our survey Q19 is such a question. The largest number of 'other' inputs in our survey can be found at Q19 asking "What are your obstacles to mastering MPI?" (Fig. 15). Although the largest answer is the choice of 'No obstacles,' we got 111 'other' inputs exceptionally. Q4 and Q7 are the second largest (70), but these are because of the variety of answers. More than 20 participants answered 'other' raise 'time' (to master MPI). Many other participants pointed out the need of MPI programming guideline ('clear doc.', 'internal doc.', 'implementation doc.', 'performance guideline', and so on, in their words). Some participants complaints about implementations and the (performance or specification) differences among implementations.

[2]One participant answered 'reverse engineering!'

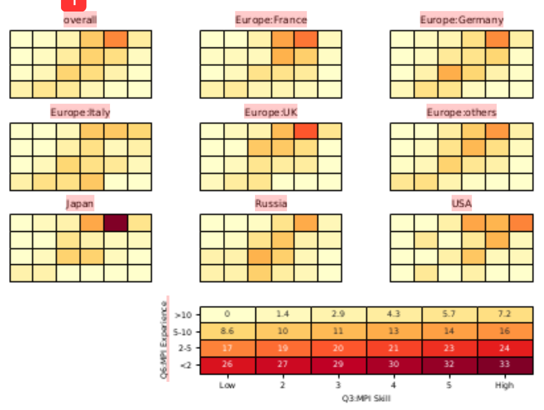

Fig. 16. Q6-Q3: MPI Experience *(single)* and MPI Skill *single*

As shown in Fig. 16, the cross-tab heatmap graphs between Q6 (Fig. 5 and Q3 (Fig. 4), there is a strong correlation, from lower-left to higher-right, between those two questions regardless to countries or regions. And these graphs tell that it takes more than 10 years of MPI programming experience to reach the high MPI skill ('4' or 'High') in most countries and regions. Many MPI users in US can reach the high ('4') MPI skill in 5 to 10 years. Hence, it takes more than 5 or 10 years to reach the high MPI skill mostly. Considering this fact and the profound nature of MPI specification (Subsection IV-B), MPI could be said to be very difficult library to use.

C. MPI Programming Difficulty and Tuning

Fig. 17 shows the result of Q15 asking "What is the most difficult part of writing an MPI program?" and Fig 18 shows the result of Q23 asking "Is there any room for performance tuning in your MPI programs?" The largest part of US and UK participants chose 'algorithm design' whilst the participants of the other countries and regions chose 'debugging'. In US, the second largest choice was 'Domain Decomposition'. In Japan, the second largest is 'Tuning'.

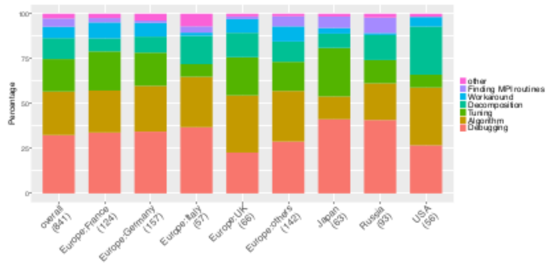

Fig. 17. Q15: MPI Programming Difficulty *(single)*

Fig. 18 has more divergence than Fig 17. The participants selected 'my MPI programs are well-tuned' is only around 10% excepting Japan and Russia. There seems to be a lots of room to tune MPI programs in general, however, around 40% of participants said they do not have enough resource to do that. In Japan, the percentage of well-tuned program is only few percent. Contrastingly, Russian percentage of choosing 'well-tuned' is the highest among the countries and regions.

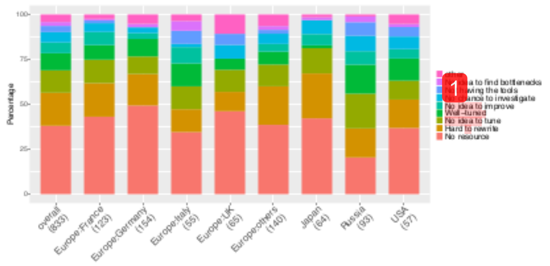

Fig. 18. Q23: Performance Tuning *(single)*

D. MPI+X and Alternatives

Fig. 19 shows the result of Q22 asking "Have you ever written MPI+"X" programs?" Most participants have the experiences of writing MPI+OpenMP programs. 'CUDA' is the second largest in US and the percentage of 'No' in US is the lowest among the others. Considering the low percentage of 'No' (approx. 25% in overall), most participants using MPI with something else. [6] reported that the approximately 3/4 of the target programs use the hybrid model of MPI+OpenMP. This percentage is very close to our survey result.

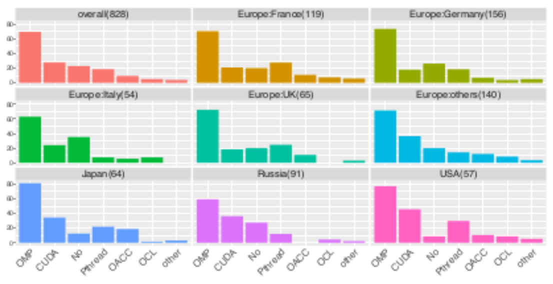

Fig. 19. Q22: MPI+X *(multiple)*

When the participants were asked the question "What, if any, alternatives are you investigating to indirectly call MPI or another communication layer by using another parallel language/library?" they exhibited as shown in Fig. 20. In overall, almost half participants do not investigate the alternatives. The second largest answer was 'Framework' (i.e. a framework or library using MPI) followed by 'PGAS'. The differences over the countries and regions are not so big.

Fig. 21 shows the cross-tab analysis of the above Q22 and Q24. A certain percentage of participants of Germany, Italy, Russia and other European countries using MPI+OpenMP without investigating the MPI alternative (upper right corners of the heatmaps in the figure).

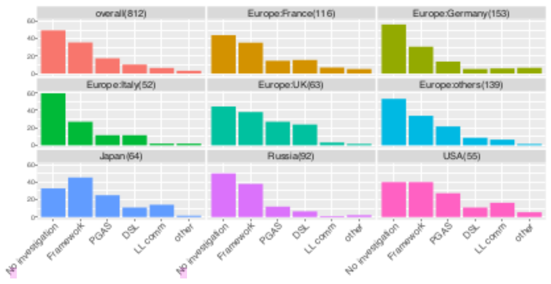

Fig. 20. Q24: MPI Alternatives *(multiple)*

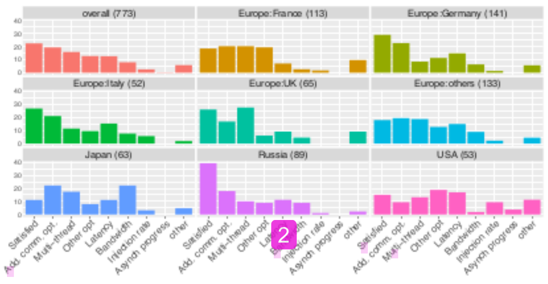

Fig. 22. Q25: Features to improve *(single)*

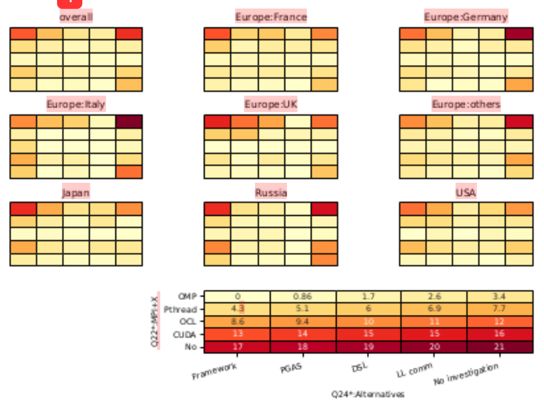

Fig. 21. Q22-Q24: MPI+X *(multiple)* and MPI Alternatives *(multiple)*

E. Missing Features and Semantics

It is a general concern how MPI provides optimization opportunities in terms of hardware capabilities such as being able to handle the various topologies of hardware components more efficiently. To answer this, Q25 asking "If there were one communication aspect which is not enough in the current MPI could improve the performance of your application, what would you prioritize? Or . . ." (Fig. 22), and Q26 asking "Is MPI providing all the communication semantics required by your application? If not, what is missing?" (Fig. 23) were prepared.

In Fig. 22, only 23% of overall MPI users are satisfied with the current situation. Interestingly enough the second largest percentage is 'additional comm. opt.' ('Additional optimization opportunities in terms of communication (network topology awareness, etc.)', followed by 'Multi-thread' and 'Other opt.' ('Optimization opportunities except communication (architecture awareness, dynamic processing, accelerator support, etc.)').

Q26 is somewhat similar to Q25, but looking for more precise answers. This question tackles the issue on which semantic feature is missing from MPI. Overall a very similar picture emerges with Q25, almost one third of the participants are satisfied with the current situation. There is a high discrepancy between Japan where users are the least satisfied with the current situation and Russia which is the most satisfied. The same situation can be seen in Q25. The highest given answer concerns 'Add. opt' ('Additional optimization opportunities in terms of communication (topology awareness, locality, etc.)'). This is coherent with Q25 and managing efficiently the topology and the locality seem a major concern to many users. Then comes the concerns about the lack of resilience, a concern shared by more than 20% of the participants. Hiding latency through generalization of asynchrony over the whole set of functions is another point raised repeatedly. 16% of the users think that a simpler and easier API would be desirable.

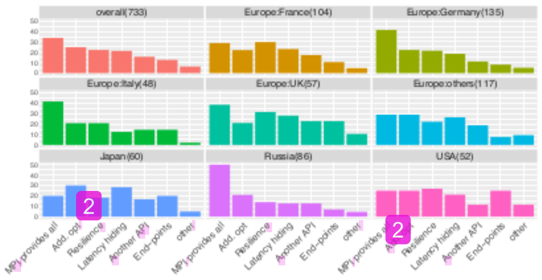

Fig. 23. Q26: MPI missing Semantics *(multiple)*

Finally, the least desired feature concerns the notion of endpoints, as discussed in the MPI standardization effort. However taking into account the extremely technical aspect of this question, and it's intricate evolution in the standard, it might be possible that most people answering this question knew little, and possible imprecisely, what this feature was exactly about.

F. Compatibility vs. Performance

In somewhat long history of MPI, maintaining the backward compatibility can be obstacles when to introducing new features to enhance MPI capabilities. Fig. 24 shows the result of the question asking which is more important, performance or compatibility. Fig. 25 shows the percentages of Q28 asking backward compatibility.

In Fig 24, let's consider three groups; *performance group* choosing 'Performance' or '4', *compatibility group* choosing

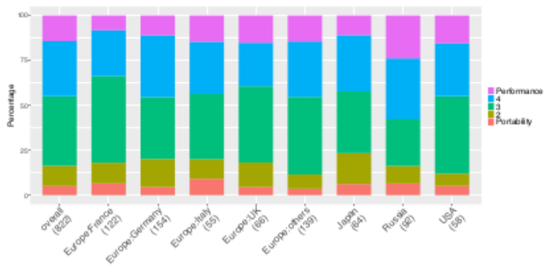

Fig. 24. Q29: Performance vs. Compatibility *(single)*

'Portability' or '2'), and *grey group* who chose '3'. Apparently the grey group dominates in most countries and regions (excepting Russia), however, the performance group occupies more percentage. Regarding to Russia, a certain percentage of Russian participants answered 'my program is too small' in Q21 (Fig. 6). If program is small enough, then the compatibility does not cause a big problem.

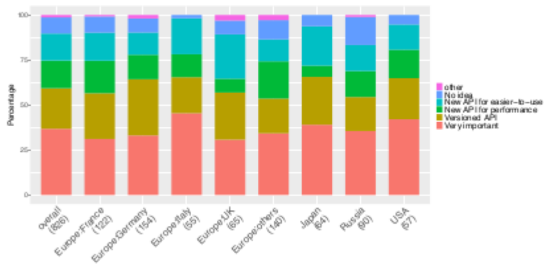

Fig. 25. Q28: Backward Compatibility *(single)*

As shown in Fig. 25, around 40% of participants answered that the compatibility is very important, while the rest of participants may accept the incompatibility conditionally. The incompatibility forces users to update their program. The result of Q28 may suggests that users would accept incompatibility if the users could get some kind of benefit from the incompatible updates of MPI specification.

VI. SUMMARY

We have conducted a questionnaire survey and succeeded to gather more than 850 participants from more than 40 countries and regions. By analyzing the collected data, we could get several findings. As for the MPI features, the dynamic process feature is considered not only as a less-used feature but also a useless feature (highlighting that the MPI programming model is seen as *static*). By asking several questions how participants obtain MPI knowledge and experiences, it is revealed that MPI is a very difficult-to-use library. Most participants read the MPI standard only partially to check the specification of MPI functions. Instead many MPI users desire to have practical programming guideline, online documents in hypertext form, and useful sample programs. Most important (and most difficult) thing is those supplemental documents in any form must be up-to-dated and thorough. For the compatibility, many MPI users may accept to sacrifice the compatibility to get more performance.

All collected answers, the programs to analyze the survey data and to generate graphs, and all published reports are available at `https://github.com/bosilca/MPIsurvey.git`.

ACKNOWLEDGMENTS

We thank to those who participated in this survey and those who helped us to distribute the questionnaire to their local communities. We especially thank to MPI Forum members who gave us many significant comments on the draft questionnaire. This research is partially supported by the NCSA-Inria-ANL-BSC-JSC-Riken-UTK Joint-Laboratory for Extreme Scale Computing [5].

REFERENCES

[1] Exascale Computing Project, "Exascale Computing Project," https://exascaleproject.org/.
[2] D. E. Bernholdt, S. Boehm, G. Bosilca, M. Gorentla Venkata, R. E. Grant, T. J. Naughton, III, H. P. Pritchard, M. Schulz, and G. R. Vallee, "A Survey of MPI Usage in the U.S. Exascale Computing Project," June 2018.
[3] Research Organization for Information Science and Technology (RIST), "High-Performance Computing Infrastructure," http://www.hpci-office.jp/folders/english.
[4] RIST, "Report of the fourth survey on the K computer and the other HPCI systems," http://www.hpci-office.jp/materials/k_chosa_4th.pdf?4th, 2018, (in Japanese).
[5] JLESC, "Joint Laboratories for Extreme-scale Computing," https://jlesc.github.io/.
[6] I. Laguna, R. Marshall, K. Mohror, M. Ruefenacht, A. Skjellum, and N. Sultana, "A large-scale study of mpi usage in open-source hpc applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3295500.3356176
[7] N. Sultana and A. Skjellum, "Understanding the usage of mpi in exascale proxy applications," 2018.
[8] S. Chunduri, S. Parker, P. Balaji, K. Harms, and K. Kumaran, "Characterization of mpi usage on a production supercomputer," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, ser. SC '18. IEEE Press, 2018. [Online]. Available: https://doi.org/10.1109/SC.2018.00033
[9] B. Klenk and H. Fröning, "An overview of mpi characteristics of exascale proxy applications," in *High Performance Computing*, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds. Cham: Springer International Publishing, 2017, pp. 217–236.
[10] D. F. Richards, O. Aaziz, J. Cook, H. Finkel, B. Homerding, P. McCorquodale, T. Mintz, S. Moore, A. Bhatele, and R. Pavel, "Fy18 proxy app suite release. milestone report for the ecp proxy app project," 10 2018.
[11] Top500.org, "Top 500," https://www.top500.org.
[12] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 3.1*. High Performance Computing Center Stuttgart (HLRS), 2015.
[13] W. Kendall, D. Nath, and W. Bland, "A Comprehensive MPI Tutorial Resource," https://mpitutorial.com.
[14] W. Kendall, "MPI Tutorial Introduction," https://mpitutorial.com/tutorials/mpi-introduction/.

APPENDIX

The followings are the list of all questions associated with choices. The question numbers suffixed by '*' are multiple-answer questions. The choices are followed by corresponding abbreviations in square brackets.

Q1: What is your main occupation **C**1) College/University [Univ] **C**2) Governmental institute [Gov] **C**3) Hardware vendor [HW] **C**4) Software vendor [SW] **C**5) Private research institute [priv] **C**6) Other [other]

Country: Select main country or region of your workplace in past 5 years. Choose one from the country list.

Q2: Rate your overall programming skill (non-MPI programs). Choose one in the range of 1 to 6. [Low-High]

Q3: Rate your MPI programming skill. Choose one in the range of 1 to 6. [Low-High]

Q4*: What programming language(s) do you use most often? **C**1) C/C++ [C(++)] **C**2) Fortran 90 or newer [>=F90] **C**3) Fortran (older one than Fortran 90) [<F90] **C**4) Python [Py] **C**5) Java [Java] **C**6) Other [other]

Q5: How long have you been writing computer programs (incl. non-MPI programs)? **C**1) more than 10 years [>10] **C**2) between 5 and 10 years [5-10] **C**3) between 2 and 5 years [2-5] **C**4) less than 2 years [<2]

Q6: How long have you been writing MPI programs? **C**1) more than 10 years [>10] **C**2) between 5 and 10 years [5-10] **C**3) between 2 and 5 years [2-5] **C**4) less than 2 years [<2]

Q7*: Which fields are you mostly working in? **C**1) System software development (OS, runtime library, communication library, etc.) [OS/R] **C**2) Parallel language (incl. domain specific language) [Lang] **C**3) Numerical application and/or library [Num-App/Lib] **C**4) AI (Deep Learning) [AI] **C**5) Image processing [Image Proc] **C**6) Big data [Bg Data] **C**7) Workflow and/or In-situ [Workfflow] **C**8) Visualization [Visualization] **C**9) Tool development (performance tuning, debugging, etc.) [Tool] **C**10) Other [other]

Q8*: What is your major role at your place of work? **C**1) Research and development of application(s) [Apps] **C**2) Research and development software tool(s) [Tools] **C**3) Parallelization of sequential program(s) [parallelize] **C**4) Performance tuning of MPI program(s) [Tuning] **C**5) Debugging MPI programs [Debug] **C**6) Research and development on system software (OS and/or runtime library) [OS/R] **C**7) Other [other]

Q9: Have you ever read the MPI standard specification document? **C**1) I read all. [All] **C**2) I read most of it. [Mostly] **C**3) I read only the chapters of interest for my work. [Partly] **C**4) I have not read it, but I plan to. [Wish] **C**5) No, and I will not read it. [No]

Q10*: How did you learn MPI? **C**1) I read the MPI standard document. [Standard] **C**2) I had lecture(s) at school. [School] **C**3) I read articles found on Internet. [Internet] **C**4) I read book(s). [Books] **C**5) Other lectures or tutorials (workplace, conference). [Other lec.] **C**6) I have not learned MPI. [Never learned] **C**7) Other [other]

Q11*: Which MPI book(s) have you read? **C**1) Beginning MPI (An Introduction in C) [Beginning MPI] **C**2) Parallel Programming with MPI [Parallel Programming] **C**3) Using MPI [Using MPI] **C**4) Parallel Programming in C with MPI and OpenMP [Parallel Programming in C] **C**5) MPI: The Complete Reference [MPI: Complete Ref] **C**6) I have never read any MPI books [(no book)] **C**7) Other [other]

Q12*: Which MPI implementations do you use? **C**1) MPICH [MPICH] **C**2) Open MPI [OMPI] **C**3) Intel MPI [Intel] **C**4) MVAPICH [MVA] **C**5) Cray MPI [Cray] **C**6) IBM MPI (BG/Q, PE, Spectrum) [IBM] **C**7) HPE MPI [HPE] **C**8) Tianhe MPI [Tianhe] **C**9) Sunway MPI [Sunway] **C**10) Fujitsu MPI [Fujitsu] **C**11) NEC MPI [NEC] **C**12) MS MPI [MS] **C**13) MPC MPI [MPC] **C**14) I do not know [No idea] **C**15) Other [other]

Q13: Why did you choose the MPI implementation(s)? **C**1) I like to use it. [I like] **C**2) I was said to use it. [Said to use] **C**3) I could not have any choice (the one provided by a vendor). [No choice] **C**4) I am familiar with it. [Familiar] **C**5) I have no special reason. [No reason]

Q14*: How do you check MPI specifications when you are writing MPI programs? **C**1) I read the MPI Standard document (web/book). [MPI standard] **C**2) I read online documents (such as man pages). [Online docs] **C**3) I search the Internet (Google / Stack Overflow). [Internet] **C**4) I ask colleagues. [Colleagues] **C**5) I read book(s) (except the MPI standard). [Books] **C**6) I know almost all MPI routines. [I know all] **C**7) Other [other]

Q15: What is the most difficult part of writing an MPI program? **C**1) Algorithm design [Algorithm] **C**2) Debugging [Debugging] **C**3) Domain decomposition [Decomposition] **C**4) Finding appropriate MPI routines [Finding MPI routines] **C**5) Implementation issue workaround [Workaround] **C**6) Performance tuning [Tuning] **C**7) Other [other]

Q16*: Which MPI features have you never heard of? **C**1) Point-to-point communications [Point-to-point] **C**2) Collective communications [Collectives] **C**3) Communicator operations (split, duplicate, and so on) [Communicator] **C**4) MPI datatypes [Datatypes] **C**5) One-sided communications [One-sided] **C**6) Dynamic process creation [Dyn. process] **C**7) Persistent communication [Persistent] **C**8) PMPI interface [PMPI] **C**9) MPI with OpenMP (or multi-thread) [with OpenMP] **C**10) Other [other]

Q17*: What aspects of the MPI standard do you use in your program in its current form? **C**1) Point-to-point communications [Point-to-point] **C**2) Collective communications [Collectives] **C**3) Communicator operations (split, duplicate, and so on) [Communicator] **C**4) MPI datatypes [Datatypes] **C**5) One-sided communications [One-sides] **C**6) Dynamic process creation [Dyn. process] **C**7) Persistent communications [Persistent] **C**8) MPI with OpenMP (or multithread) [with OpenMP] **C**9) PMPI interface [PMPI] **C**10) Other [other]

Q18*: Which MPI thread support are you using? **C**1) MPI_THREAD_SINGLE [SINGLE] **C**2) MPI_THREAD_FUNNELED [FUNNELED] **C**3) MPI_THREAD_SERIALIZED [SERIALIZED] **C**4) MPI_THREAD_MULTIPLE [MULTIPLE] **C**5) I have never called `MPI_INIT_THREAD` [never used] **C**6) I do not know or I do not care. [Np idea] **C**7) Other [other]

Q19*: What are your obstacles to mastering MPI? **C**1) I have no obstacles. [No obstacles] **C**2) Too many routines. [Too many routines] **C**3) No appropriate lecture / book / info. [No appropriate one] **C**4) Too complicated and hard to understand. [Complicated] **C**5) I have nobody to ask. [Nobody to ask] **C**6) I do not like the API. [Dislike API] **C**7) Other [other]

Q20: When you call an MPI routine, how often do you check the error code of the MPI routine (excepting MPI-IO)? **C**1) I rely on the default Errors abort error handling [Default] **C**2) Always [Always] **C**3) Mostly [Mostly] **C**4) Sometimes [Sometimes] **C**5) Never [Never] **C**6) Other [other]

Q21: In most of your programs, do you pack MPI function calls into their own file or files to have your own abstraction layer for communication? **C**1) Yes, to minimize the changes of communication API. [Yes] **C**2) Yes, but I have no special reason for doing that. [Yes, but no reason] **C**3) No, my program is too small to do that. [No, too small] **C**4) No, MPI calls are scattered in my programs. [No, scattered] **C**5) Other

Q22*: Have you ever written MPI+X programs? **C**1) OpenMP [OMP] **C**2) Pthread [Pthread] **C**3) OpenACC [OACC] **C**4) OpenCL [OCL] **C**5) CUDA [CUDA] **C**6) No [No] **C**7) Other [other]

Q23: Is there any room for performance tuning in your MPI programs? **C**1) No, my MPI programs are well-tuned. [Well-tuned] **C**2) Yes, I know there is room for tuning but I should re-write large part of my program to do that. [Hard to rewrite] **C**3) Yes, I know there is room for tuning but I do not have enough resources to do that. [No resource] **C**4) I think there is room but I do not know how to tune it. [No idea to tune] **C**5) I do not have (know) tools to find performance bottlenecks. [Not having the tools] **C**6) I have no chance to investigate. [No chance to investigate] **C**7) I do not know how to find bottlenecks. [No idea to find bottlenecks] **C**8) I do not know if there is room for performance tuning. [No idea to improve] **C**9) Other [other]

Q24*: What, if any, alternatives are you investigating to indirectly call MPI or another communication layer by using another parallel language/library? **C**1) A framework or library using MPI. [Framework] **C**2) A PGAS language (UPC, Coarray Fortran, OpenSHMEM, XcalableMP, ...). [PGAS] **C**3) A Domain Specific Language (DSL). [DSL] **C**4) Low-level communication layer provided by vendor (Verbs, DCMF, ...). [LL comm] **C**5) I am not investigating any alternatives. [No investigation] **C**6) Other [other]

Q25: If there were one communication aspect which is not enough in the current MPI could improve the performance of your application, what would you prioritize? Or is MPI providing all the communication semantics required by your application? If not, what is missing? **C**1) Latency [Latency] **C**2) Message injection rate [Injection rate] **C**3) Bandwidth [Bandwidth] **C**4) Additional optimization opportunities in terms of communication (network topology awareness, etc.) [Additional comm. opt.] **C**5) Optimization opportunities except communication (architecture awareness, dynamic processing, accelerator support, etc.) [Other opt.] **C**6) Multi-threading support [Multi-thread] **C**7) Asynchronous progress [Asynch progress] **C**8) MPI provides all semantics I need [Satisfied] **C**9) Other [other]

Q26*: Is MPI providing all the communication semantics required by your application? If not, what is missing? **C**1) Latency hiding (including asynchronous completion) [Latency hiding] **C**2) Endpoints (multi-thread, sessions) [Endpoints] **C**3) Resilience (fault tolerance) [Resilience] **C**4) Additional optimization opportunities in terms of communication (topology awareness, locality, etc.) [Additional opt] **C**5) Another API which is easier and/or simpler to use [Another API] **C**6) MPI is providing all the communication semantics required by my application [MPI provides all] **C**7) Other [other]

Q27*: What MPI feature(s) are NOT useful for your application? **C**1) One-sided communication [One-sided] **C**2) Datatypes [Datatypes] **C**3) Communicator and group management [Communicator] **C**4) Collective operations [Collectives] **C**5) Process topologies [Topologies] **C**6) Dynamic process creation [Dyn. process] **C**7) Error handlers [Error] **C**8) There are no unnecessary features [No] **C**9) Other [other]

Q28: Do you think the MPI standard should maintain backward compatibility? **C**1) Yes, compatibility is very important for me. [Very important] **C**2) API should be clearly versioned. [Versioned API] **C**3) I prefer to have new API for better performance. [New API for performance] **C**4) I prefer to have new API which is simpler and/or easier-to-use. [New API for easier-to-use] **C**5) I do not know or I do not care. [No idea] **C**6) Other [other]

Q29: In the tradeoff between code portability and performance, which is more or less important for you to write MPI programs? Choose one in the range of 1 to 6. [Portability-Performance] exampi-2020.pdf

ORIGINALITY REPORT

15%

SIMILARITY INDEX

PRIMARY SOURCES

1	hal.inria.fr Internet	894 words — 10%
2	William McClain. "Symmetry Theory in Molecular Physics with Mathematica", Springer Science and Business Media LLC, 2008 Crossref	68 words — 1%
3	d-nb.info Internet	39 words — < 1%
4	Ziqing Luo, Stephen F. Siegel. "Towards Deductive Verification of Message-Passing Parallel Programs", 2018 IEEE/ACM 2nd International Workshop on Software Correctness for HPC Applications (Correctness), 2018 Crossref	38 words — < 1%
5	Waseem Ahmed. "Extending Shared-Memory Computations to Multiple Distributed Nodes", International Journal of Advanced Computer Science and Applications, 2020 Crossref	36 words — < 1%
6	Nuria Losada, Aurelien Bouteiller, George Bosilca. "Asynchronous Receiver-Driven Replay for Local Rollback of MPI Applications", 2019 IEEE/ACM 9th Workshop on Fault Tolerance for HPC at eXtreme Scale (FTXS), 2019 Crossref	36 words — < 1%
7	Ian Karlin, Carlos Costa, Johann Dahm, Giacomo Domeniconi et al. "Preparation and optimization of a diverse workload for a large-scale heterogeneous system",	34 words — < 1%

Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '19, 2019
Crossref

8	David E. Bernholdt, Swen Boehm, George Bosilca, Manjunath Gorentla Venkata et al. "A survey of MPI usage in the US exascale computing project", Concurrency and Computation: Practice and Experience, 2018 Crossref	24 words — < 1%
9	docplayer.net Internet	22 words — < 1%
10	link.springer.com Internet	20 words — < 1%
11	www.ideals.illinois.edu Internet	17 words — < 1%
12	mafiadoc.com Internet	17 words — < 1%
13	Kenji Ono, Jorji Nonaka, Tomohiro Kawanabe, Masahiro Fujita, Kentaro Oku, Kazuma Hatta. "HIVE: A cross-platform, modular visualization framework for large-scale data sets", Future Generation Computer Systems, 2020 Crossref	15 words — < 1%
14	Balazs Gerofi, Masamichi Takagi, Yutaka Ishikawa. "Toward Operating System Support for Scalable Multithreaded Message Passing", Proceedings of the 22nd European MPI Users' Group Meeting on ZZZ - EuroMPI '15, 2015 Crossref	13 words — < 1%
15	Lacoste, Xavier, Mathieu Faverge, George Bosilca, Pierre Ramet, and Samuel Thibault. "Taking Advantage of Hybrid Systems for Sparse Direct Solvers via Task-Based Runtimes", 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, 2014. Crossref	13 words — < 1%

16	www.groundai.com Internet	12 words — < 1%
17	export.arxiv.org Internet	9 words — < 1%
18	Ignacio Laguna, Ryan Marshall, Kathryn Mohror, Martin Ruefenacht, Anthony Skjellum, Nawrin Sultana. "A large-scale study of MPI usage in open-source HPC applications", Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '19, 2019 Crossref	9 words — < 1%
19	Masamichi Takagi, Norio Yamaguchi, Balazs Gerofi, Atsushi Hori, Yutaka Ishikawa. "Adaptive transport service selection for MPI with InfiniBand network", Proceedings of the 3rd Workshop on Exascale MPI - ExaMPI '15, 2015 Crossref	9 words — < 1%
20	"Euro-Par 2016: Parallel Processing", Springer Science and Business Media LLC, 2016 Crossref	8 words — < 1%
21	es.scribd.com Internet	8 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	OFF		