A Report of MPI International Survey

Atsushi Hori
Takahiro Ogura
Balazs Gerofi
Jie Yin
Yutaka Ishikawa
Riken CCS

George Bosilca
The University of Tennessee

Emmanuel Jeannot
Inria

1 ABSTRACT

The Message Passing Interface (MPI) plays a critical 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' expectations and concerns, and adapt accordingly.

Existing studies on MPI uses are focused on a restricted target domain, such as the Exascale Computing Project (ECP) [3] study conducted in 2017 [2] that focused on MPI usage in the context of ECP applications; and/or those that are geographically constrained to a single laboratory, funding agency or at best, country. Such studies inspired us to conduct a larger study, not focused on high-end HPC, but targeting a wider audience and involving 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.

This survey was conducted from February to June 2019, and at the time of this writing has gathered more than 800 answers from 42 countries. This report, based the July 2020 results of the survey, focuses on the MPI users communities awareness of MPI features. This survey reveals the staggering fact that most MPI users make use of a small subset of the MPI API, a subset that mostly avoids features introduced after MPI 2.2, released in 2009, such as PMPI, dynamic process creation, and persistent communications. Further, the survey reveals that many MPI users learn MPI from the internet and/or some form of online documents, highlighting a shift in the education of the target community. These two outcomes seem to suggest that many MPI users failed to receive the information about new MPI features because most internet sources are outdated and lack support for them. It also suggests that one quick way to address this is to provide the necessary education, possibly as part of the MPI Forum effort.

2 SURVEY

The points we kept in mind while designing the survey's questions were: (a) minimizing the number of questions, (b) making them easy to answer, and (c) avoiding ambiguity. The questionnaire is implemented by using firstly Google Forms and later Microsoft Forms, and distributed via emails to major mailing lists such as `hpc-announce` and by personal contact to diverse academic and governmental institutions and vendors. All data, a Python program to analyze the answers, and all reports published so far (including this one) are available on GITHUB.[1]

Table 1: Top 10 Countries of Participants

Rank	Country	# Ans	Percent
1	Germany	159	18.68
2	France	125	14.69
3	Russia	94	11.05
4	UK	67	7.87
5	Japan	64	7.52
6	USA	58	6.82
7	Italy	57	6.60
8	Switzerland	40	5.76
9	South Korea	27	3.17
10	Austria	26	3.06
11	China (+Taiwan)	18	2.11

42 countries, 851 participants

Table 2: Top500 Performance System Share (June 2020)[6]

Rank	Country	Perf. Share [%]
1	USA	28.18
2	China	25.64
3	Japan	23.92
4	Italy	3.95
5	France	3.62
6	Germany	3.11
7	UK	1.40
8	Canada	1.27
9	Netherlands	1.12
10	Switzerland	1.05

Table 1 shows the number of answers from top-10 countries. It should be noted that the survey does not ask participants nationality, but instead ask the most recent (last 5 years) country of employment.

Table 2 shows the top-10 aggregated system performance per country as of June 2020. Comparing with Table 1 highlights a big disparity, possibly related to the number of users or to their dedication of answering an online survey. In particular it can be noted that only 18 participants from Chinese institutions answered. We noticed this immediately after publishing the survey and we have been trying to narrow the gap, but our efforts were unfortunately not able to change the trend. Overall, if aggregating the answers geographically, we currently have 65% of answers from Europe, 15% from Asia, 11% from Russia, and 7% from North America.

The countries having more than 50 participants (above the line in Table 1) are the target of cross-tab analysis. One may argue that the number of 50 is too small to conduct cross-tab analysis, however, this threshold was decided to have enough number of major countries.

Most answers, around 85%, come from research organizations (universities and governmental research institutes). We think this diversity reflects not the characteristics of the countries, but came from the biased questionnaire distribution.

Since the above profiles may bias the analysis of the results, readers must keep this in mind when considering the background.

[1] https://github.com/bosilca/MPIsurvey.git

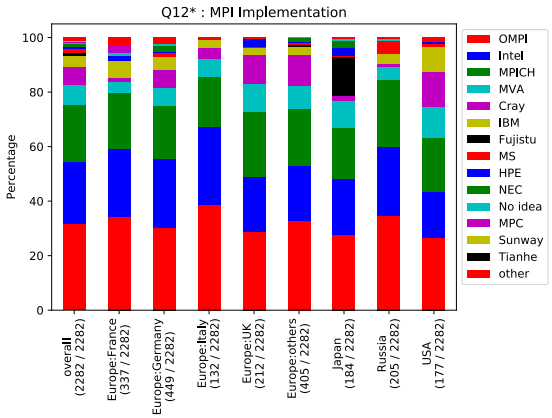


Figure 1: Which MPI implementations do you use?

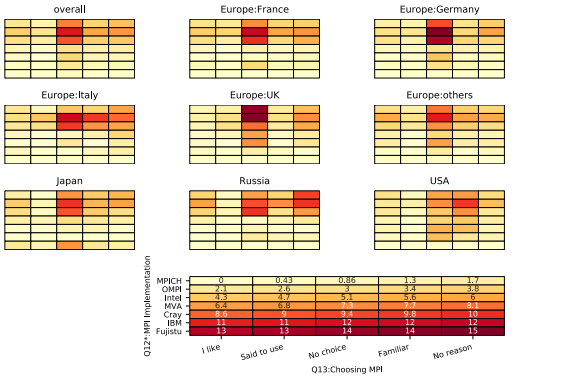


Figure 2: Cross-tab: MPI Implementations and choice reason

3 SURVEY RESULTS

3.1 Choice of MPI Implementation

Figure 1 shows the result of multiple answer question asking "Which MPI implementations do you use?" Note that the order in the legend and the orders of bars are reversed. With respect to the geographical distribution of MPI implementations, several patterns stand out. In all countries and regions, Open MPI (denoted as 'OMPI'), Intel MPI ('Intel') and MPICH dominate more than 60%. Intel MPI is the most widely used vendor provided MPI implementation all over the world. Cray MPI ('Cray') dominates around 10% in most countries and regions. From the less dominant vendors, Fujitsu MPI ('Fujitsu') enjoys wider usage than Cray MPI in Japan, while IBM has visible presence in both Europe and the US. On the other end of the spectrum are research oriented implementations, such as MadMPI with only a few mentions in France and ParaStation MPI used exclusively in Germany (they are not shown in the figure). Italy and Russia mostly use open source MPI implementations.

Figure 2 shows heatmaps of cross-tab analysis of the answers asking "which MPI implementations do you use?" (Figure 1 and "why did you choose the MPI implementation(s)?" Legend is located at the lower right corner. The numbers in the legend cells represent the percentage and darker color means higher percentage. USA has the most distributed MPI choices and more US users choose MPI implementations on their preferences than the other countries.

3.2 Threading Support

Figure 3 shows another heatmap of two questions; "Rate your MPI programming skill" (X axis) and "Which MPI thread support are you using?" (Y axis). France, Germany and UK show a very similar and expected outcome, the highest users programming skills the more they were tending to use a more complex programming approach, where threading is an integral part of the software stack. The US users exhibit a very differently behavior. Most users seem to favor the two extremes of the MPI threading support, either `MPI_THREAD_MULTIPLE` or `MPI_THERAD_SINGLE`, irrelevant of their programming skills. Most interestingly Japan exhibits a very curious result. Most Japanese users declared high MPI programming

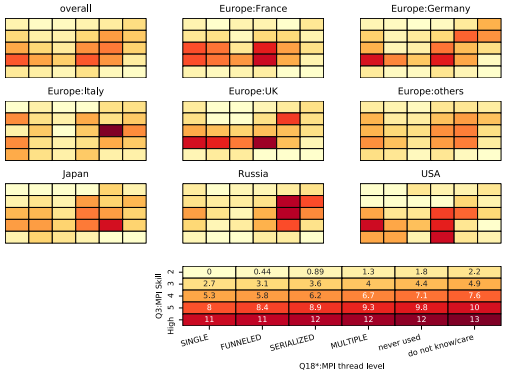


Figure 3: Cross-tab: MPI Skill and Thread Level

skills (5), however, most of them do not use MPI threading support. This Japanese anomaly is discussed in [1].

3.3 MPI Functions

Table 3, 4, 5, and 6 show the answer distribution for 4 questions out of 30. Note that the numbers in these tables reflect that these questions allow participants to choose multiple answers. The actual number of participants who answered each question is noted after the slash (/) of the total number at the end of each table.

Table 3 shows the result of the question asking "How did you learn MPI." It is a bit surprising that 40% (323/825) of participants refer to the MPI standard, despite it's lack of examples to walk-through. According to the original data (omitted due to the space limit), one quarter of participants learned MPI via the internet and/or online documents.

The result of the question asking "How do you check MPI specifications when writing MPI programs" is shown in Table 4. More than half of participants refer to the MPI standard. It is very natural for MPI users to check MPI specification by referring the MPI standard more often than reading it for learning. Most notably, the percentage of participants reading online documents and/or referring the internet occupies 46% (original data).

Table 3: How did you learn MPI?

Multiple Choice	# Answers
Articles found on Internet	439 (51.8%)
Other lectures or tutorials	438 (51.7%)
MPI standard docs	329 (38.8%)
Lecture(s) at school	294 (34.7%)
Book(s)	283 (33.4%)
Never learned MPI	26 (3.1%)
other	51 (6.0%)
total	1860 / 847

Table 4: How do you check MPI specifications when you are writing MPI programs?

Multiple Choice	# Answers
Online docs.	579 (68.8%)
Internet	566 (67.2%)
MPI standard docs.	430 (51.1%)
I ask colleagues	188 (22.3%)
I read book(s)	108 (12.8%)
I know all MPI routines	46 (5.5%)
other	11 (1.3%)
total	1928 / 842

Table 5: Which MPI features have you never heard of?

Multiple Choice	# Answers
PMPI interface	465 (66.1%)
Persistent comm.	433 (61.5%)
Dynamic process creation	383 (54.4%)
One-sided comm.	131 (18.6%)
Communicator operations	123 (17.5%)
Point-to-point comm.	91 (12.9%)
MPI datatypes	90 (12.8%)
Collective comm.	86 (12.2%)
MPI w/ OpenMP	86 (12.2%)
total	1,888 / 704

Table 6: What aspects of the MPI standard do you use in your program in its current form?

Multiple Choice	# Answers
Collective comm.	740 (89.0%)
Point-to-point comm.	710 (85.4%)
MPI datatypes	521 (62.7%)
w/ OpenMP (multithread)	478 (57.5%)
Communicator operations	412 (49.6%)
One-sided comm.	226 (27.2%)
PMPI interface	67 (8.1%)
Persistent comm.	61 (7.3%)
Dynamic process creation	51 (6.1%)
other	20 (2.4%)
total	3,240 / 831

Referring Table 5 and 6, the MPI features can be categorized into two groups: well-known ones (P2P, Collectives, and so on) and little-known ones (PMPI, Persistent and Dynamic process). Surprisingly, this tendency is almost independent from countries/regions of the participants.

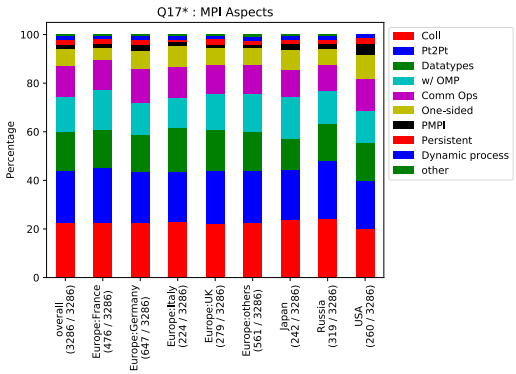


Figure 4: What aspects of the MPI standard do you use in your program in its current form?

Among the little-known MPI features, the persistent communication is one of the important directions being discussed on the MPI Forum[5]. The persistent communication can give implementors room for optimizing not only P2P but also collective communication performance. All those little-known MPI features already appeared in MPI 2.2 released in 2009. Despite the 10-year appearance, those features fail to be widely accepted. Why?

One possible answer to this question may come from the survey results asking participants "how did you learn MPI" (Table 3) and "how do you check MPI specifications when you are writing MPI programs" (Table 4). Here a significant number of participants refer to the internet and/or online documents. These are handy and allow users to get required information on the fly. However, these online medias can only be retrieved by search. To search something, a clue or some keywords must be given. Say someone wants to search something he/she does not know; how can he/she find the appropriate key words to obtain the right information? This could be the responsibility of the information providers. For example, there is no `See Also` link from the man page of `MPI_Irecv` to the corresponding persistent routines in many MPI implementations.

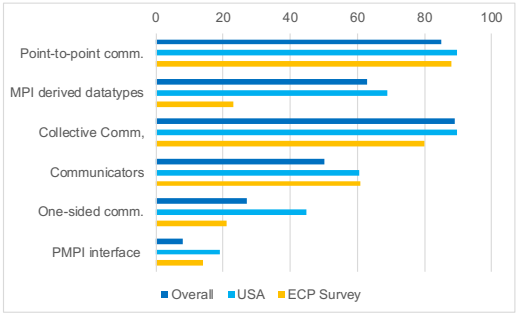


Figure 5: Comparison of the result of asking using MPI aspects between ours and ECP

There is a similar question of asking using MPI aspects in the ECP survey. Figure 5 compares the results of ours and ECP survey on the common MPI aspects. Trends of both survey are very similar excepting the usage of datatype. The percentage in our survey is much higher than that of the ECP survey.

4 SUMMARY

This large-scale international MPI survey reveals several interesting findings. We will present more results at the poster presentation.

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 [4].

REFERENCES

[1] Atsushi, H., George, B., Emmanuel, J., Takahiro, O., and Yutaka, I. Is japanese hpc another galapagos? - interim report of mpi international survey -. Tech. Rep. 34, Information Processing Society of Japan, SIGHPC, Jul 2019.
[2] Bernholdt, D. E., Boehm, S., Bosilca, G., Gorentla Venkata, M., Grant, R. E., Naughton, III, T. J., Pritchard, H. P., Schulz, M., and Vallee, G. R. A Survey of MPI Usage in the U.S. Exascale Computing Project.
[3] Exascale Computing Project. Exascale Computing Project. https://exascaleproject.org/.
[4] JLESC. Joint Laboratories for Extreme-scale Computing. https://jlesc.github.io/.
[5] MPI Forum. MPI Forum. https://www.mpi-forum.org.
[6] TOP500.org. Top 500. https://www.top500.org.

A Report of
MPI International Survey

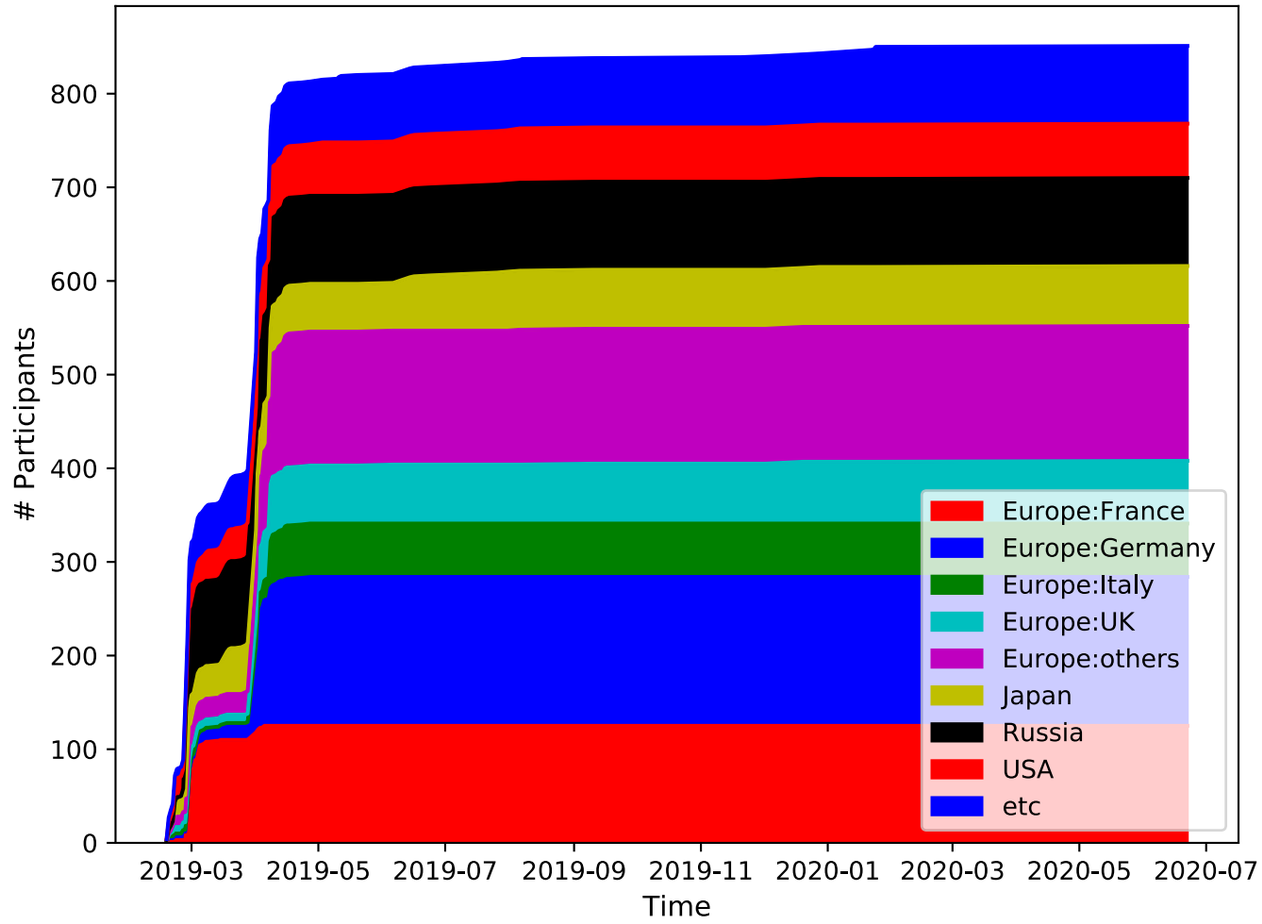
EuroMPI/USA
Sep. 2020

A. Hori, T. Ogura, B. Gerofi, Y. Jie and Y. Ishikawa - Riken CCS
G. Bosilca - UT/ICL
E. Jeannot - Inria

Objective and Survey Outline

- Beta-test
  - MPI Forum Dec. 2018 and
  - Riken CCS
- Release
  - Feb. 2019

- Up until now
  - 852 participants
  - 42 Countries and Regions



- Lessons Learned
  - Large mailing lists (e.g., hpc-announce) is not effective
  - Local mailing lists worked very well (European countries)

List of Questions

Q1: What is your main occupation?
Country: Select main country or region of your workplace in past 5 years
Q2: Rate your overall programming skill (non-MPI programs)
Q3: Rate your MPI programming skill
Q4*: What programming language(s) do you use most often?
Q5: How long have you been writing computer programs (incl. non-MPI programs)?
Q6: How long have you been writing MPI programs?
Q7*: Which fields are you mostly working in?
Q8*: What is your major role at your place of work?
Q9: Have you ever read the MPI standard specification document?
Q10*: How did you learn MPI?
Q11*: Which MPI book(s) have you read?
Q12*: Which MPI implementations do you use?
Q13: Why did you choose the MPI implementation(s)?
Q14*: How do you check MPI specifications when you are writing MPI programs?
Q15: What is the most difficult part of writing an MPI program?
Q16*: Which MPI features have you never heard of?
Q17*: What aspects of the MPI standard do you use in your program in its current form?
Q18*: Which MPI thread support are you using?
Q19*: What are your obstacles to mastering MPI?
Q20: When you call an MPI routine, how often do you check the error code of the MPI routine (excepting MPI-IO)?
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?
Q22*: Have you ever written MPI+"X" programs?
Q23: Is there any room for performance tuning in your MPI programs?
Q24*: What, if any, alternatives are you investigating to indirectly call MPI or another communication layer by using another parallel language/library?
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?
Q26*: Is MPI providing all the communication semantics required by your application? If not, what is missing?
Q27*: What MPI feature(s) are NOT useful for you application?
Q28: Do you think the MPI standard should maintain backward compatibility?
Q29: In the tradeoff between code portability and performance, which is more or less important for you to write MPI pro- grams?

Countries

- 42 countries and a region
  - The countries having 50 or more participants are cross-tab analyzed.
- Large disparity between our survey and Top500 performance share (below)
- We tried to increase the numbers of USA, China and Japan, but we failed.

Top500 Performance Share

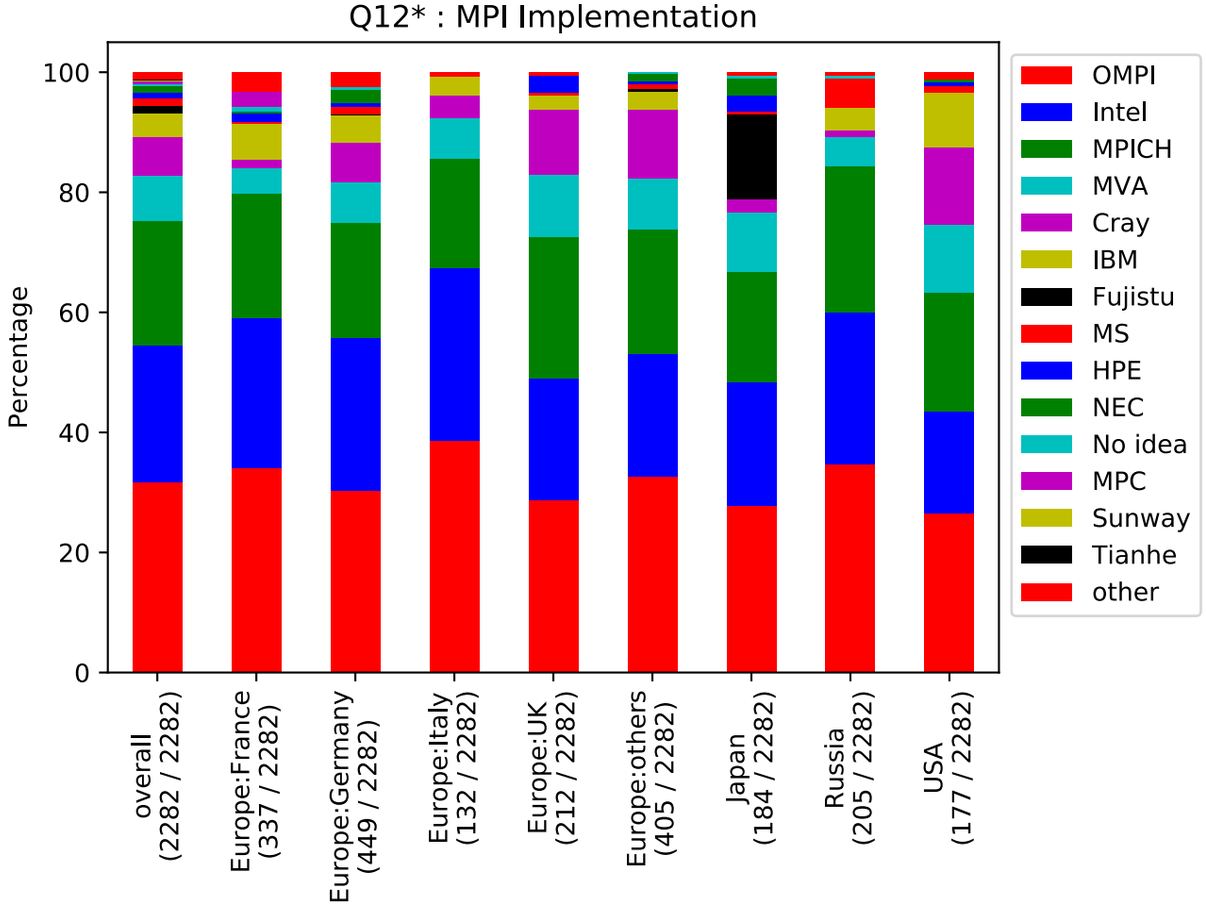
	Countries	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
1	China	226	45.2	565,553,102	1,184,700,707	31,277,940
2	United States	113	22.6	621,655,590	884,317,444	15,216,248
3	Japan	29	5.8	527,607,512	691,558,660	11,153,228
4	France	19	3.8	79,878,820	121,553,694	2,428,600
5	Germany	16	3.2	68,713,720	101,130,880	1,747,574
6	Netherlands	15	3	24,736,650	31,795,200	864,000
7	Ireland	14	2.8	23,087,540	29,675,520	806,400
8	Canada	13	2.6	27,926,060	50,348,729	767,296
9	United Kingdom	10	2	30,950,142	37,703,042	1,168,368
10	Italy	7	1.4	87,188,790	128,918,596	1,811,568
11	Brazil	4	0.8	10,991,000	19,270,566	214,040
12	Singapore	4	0.8	6,596,440	8,478,720	230,400
13	South Korea	3	0.6	18,720,660	31,496,620	709,220
14	Norway	3	0.6	7,718,070	10,432,512	287,232
15	Saudi Arabia	3	0.6	10,109,130	13,858,214	325,940
16	Australia	2	0.4	10,913,420	17,261,875	261,632
17	United Arab Emirates	2	0.4	9,013,750	12,164,803	142,368
18	Sweden	2	0.4	4,771,700	6,773,346	131,968
19	Finland	2	0.4	7,095,250	9,748,685	209,728
20	Switzerland	2	0.4	23,126,750	29,347,305	453,140

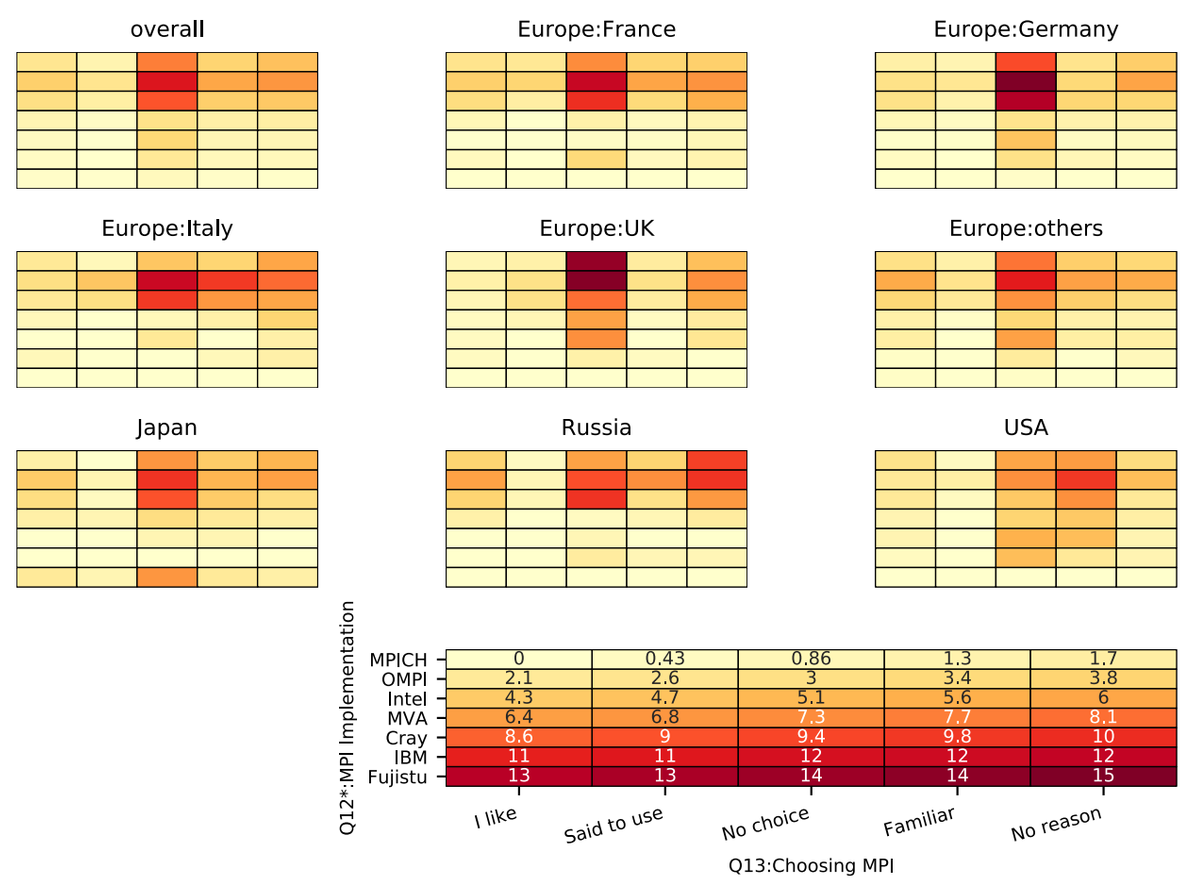
Our Survey

Country	Abbrv.	Region	# Answers
Germany	GR	Europe	159
France	FR	Europe	125
Russia	RU	Russia	94
UK	UK	Europe	67
Japan	JP	Japan	64
USA	US	USA	58
Italy	IT	Europe	57
Switzerland		Europe	40
Korea, South		South Korea	27
Austria		Europe	26
China		China	16
Sweden		Europe	15
Spain		Europe	14
India		India	12
Poland		Europe	10
Netherlands		Europe	8
Brazil		Central and South America	6
Denmark		Europe	6
Luxembourg		Europe	5
Czech Republic		Europe	5
Canada		North America	4
Finland		Europe	3
Australia		Australia	3
Argentina		Central and South America	3
Belgium		Europe	2
Greece		Europe	2
Egypt		Africa	2
Taiwan		China	2
Serbia		Europe	2
Pakistan		Asia	2
Croatia		Europe	1
Singapore		Asia	1
Peru		Central and South America	1
Denmark, Austria		Europe	1
Tunisia		Africa	1
Mexico		Central and South America	1
UAE		Asia	1
Portugal		Europe	1
Norway		Europe	1
Saudi Arabia		Asia	1
Estonia		Europe	1
Ukraine		Europe	1
42 countries			851 answers

MPI Implementation

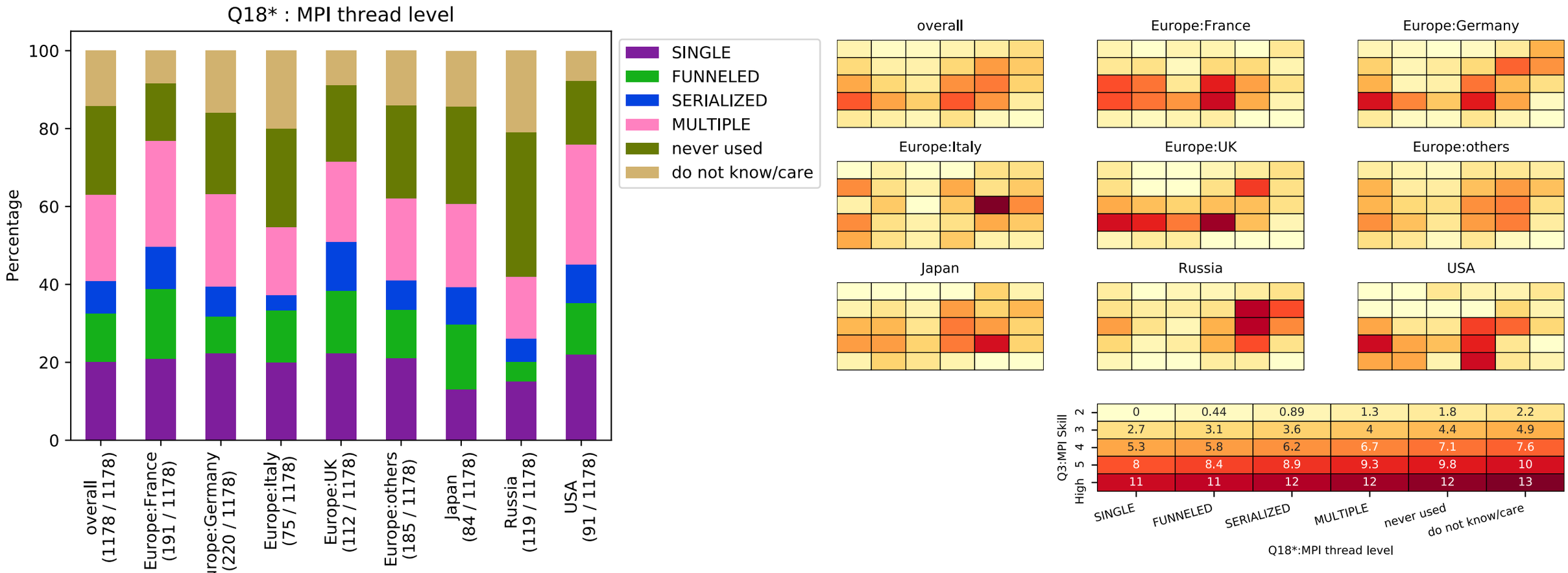
- Open source MPI implementations dominate around 80%
- Less frequently used MPI implementations;
  - Japan: Fujitsu
  - Others: Cray, IBM
- Not listed
  - France: Bull, MadMPI
  - Germany: Parastation





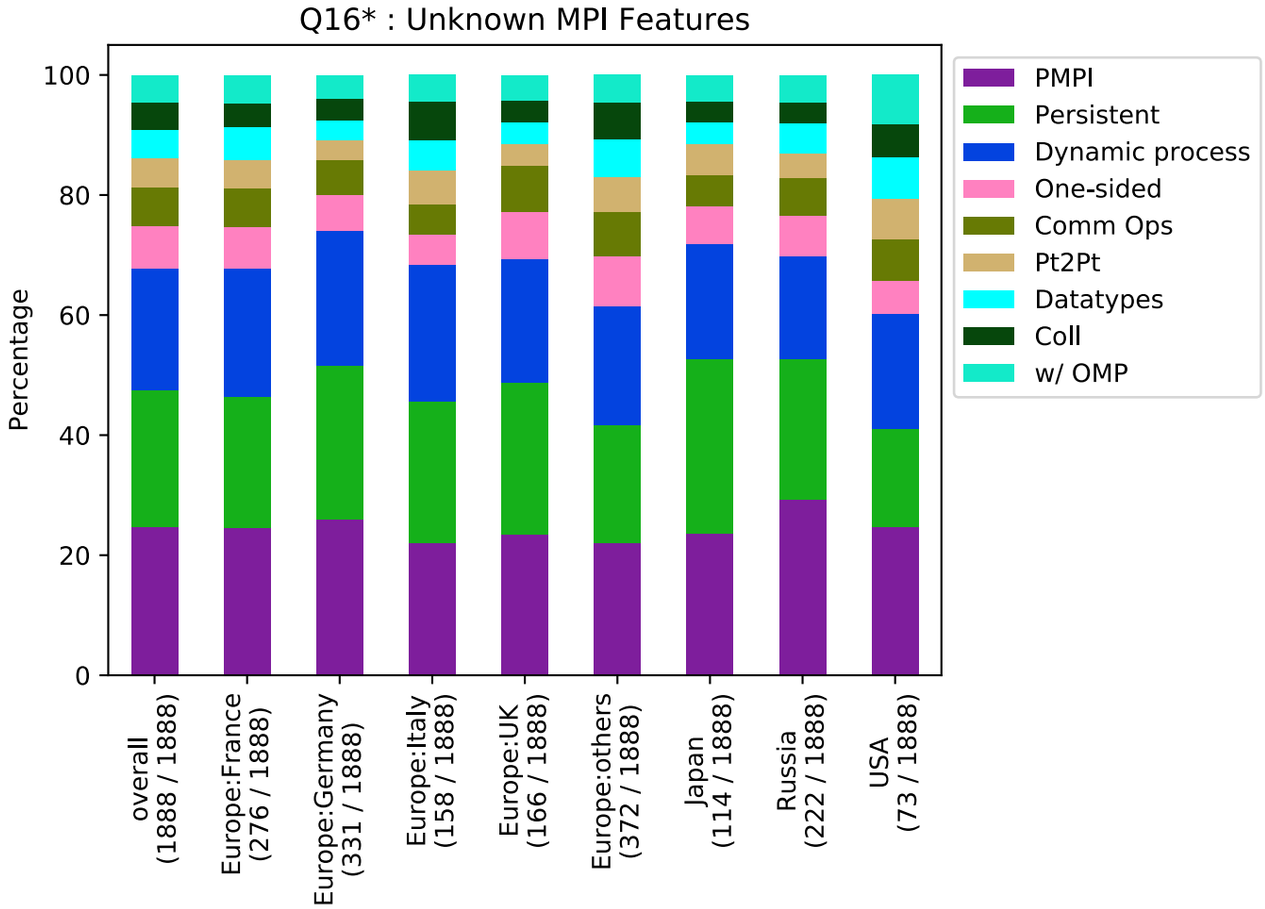
Threading Support

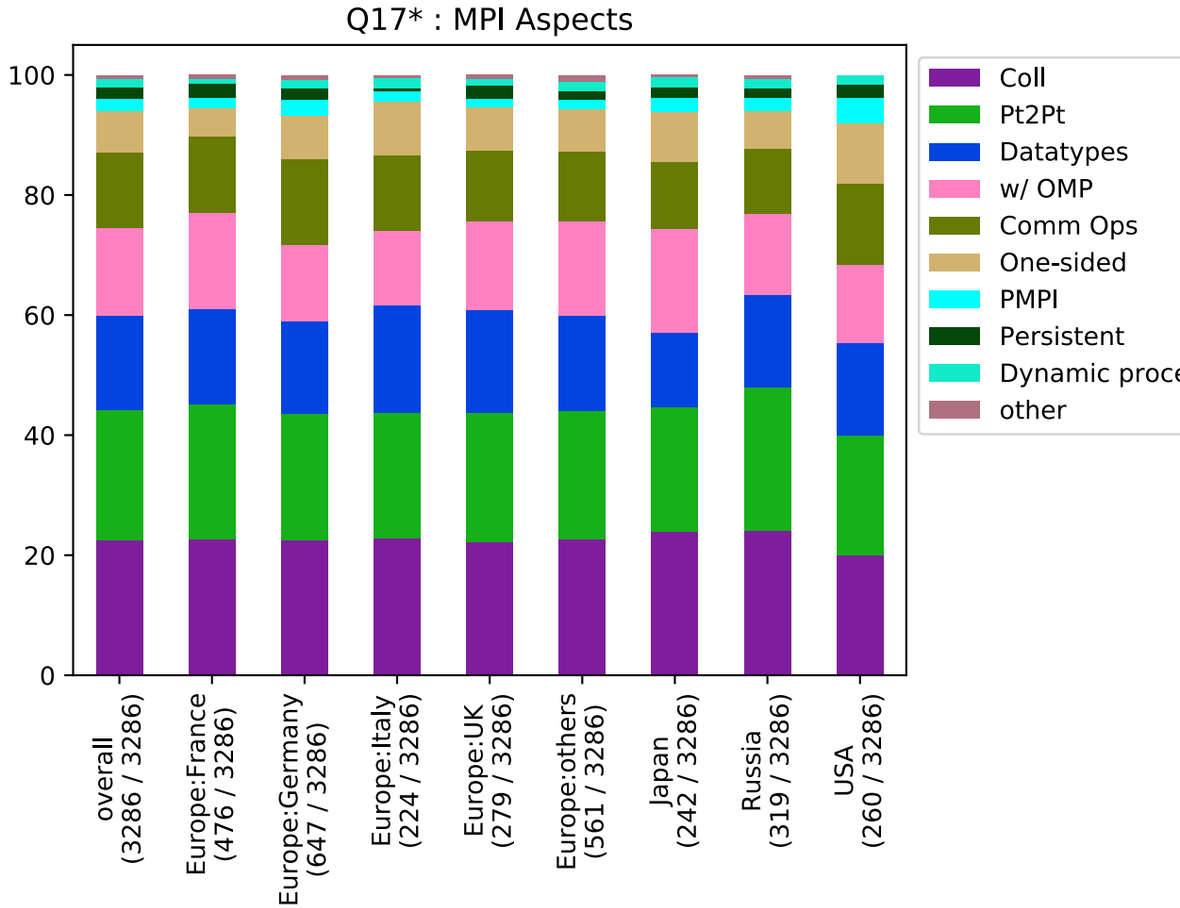
- France, Germany and UK
  - Higher the MPI skill, higher the frequency of using MULTIPLE
- USA
  - Highest percentage of using MULTIPLE
- Russia
  - Highest percentage of not using (specifying) thread level



MPI Featuress

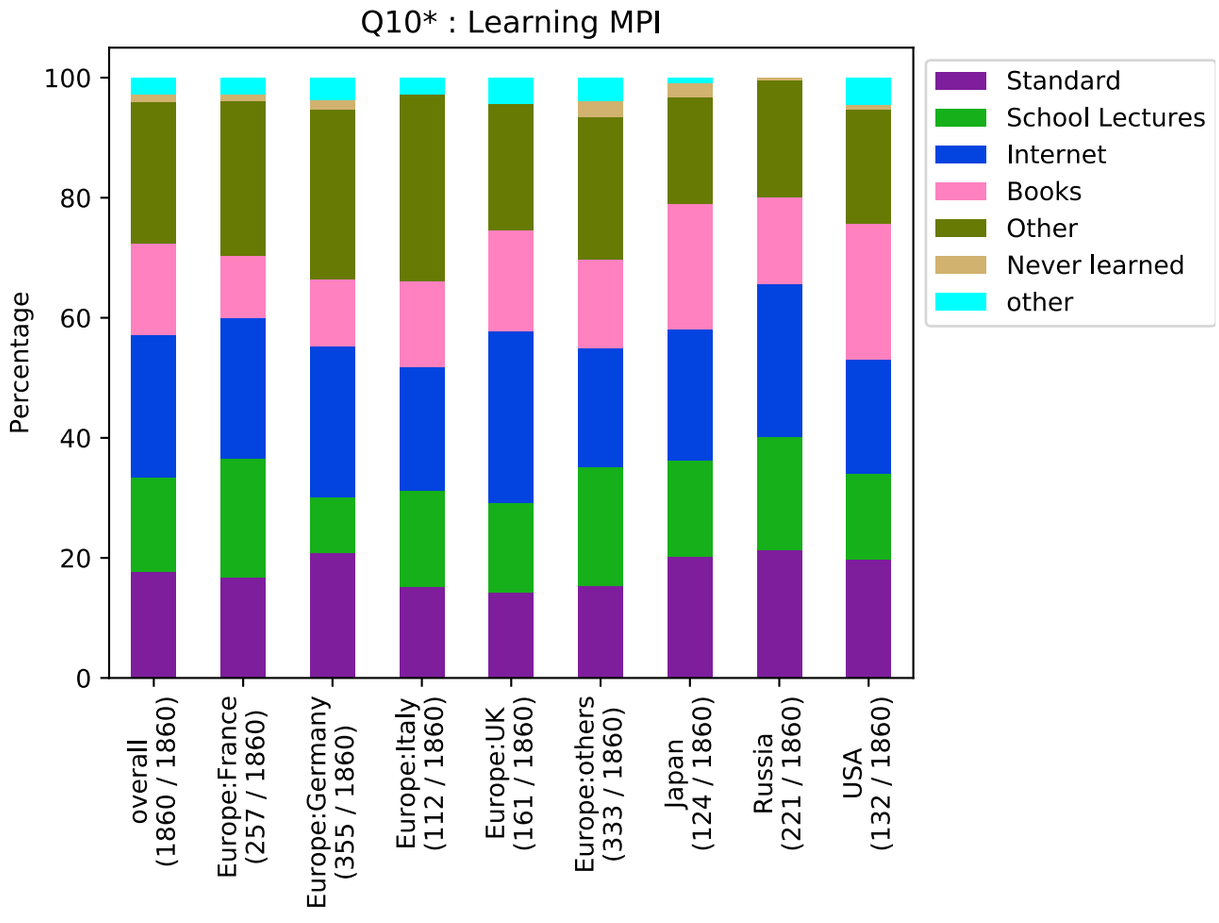
- Dynamic process creation, Persistent comm. and PMPI are the least known MPI features
  - Those features were introduced in MPI 2.2 (released 2002)

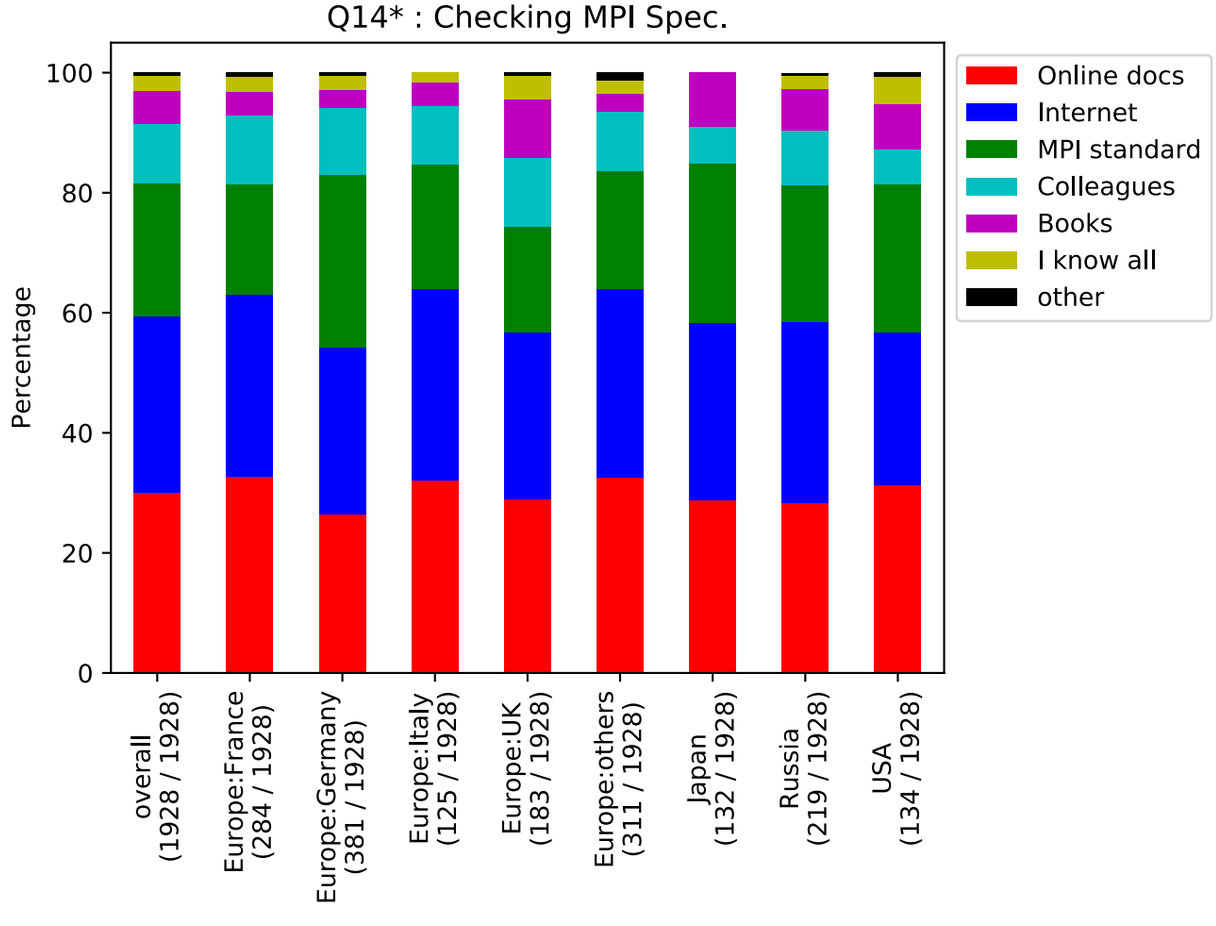




Learning and Checking MPI

- Learning
  - Certain percentage (≈20%) of people reading MPI standard
    - *Is the standard written for learning?*
  - More than 20% of people using Internet
- Checking
  - Online and Internet: ≈ 60%





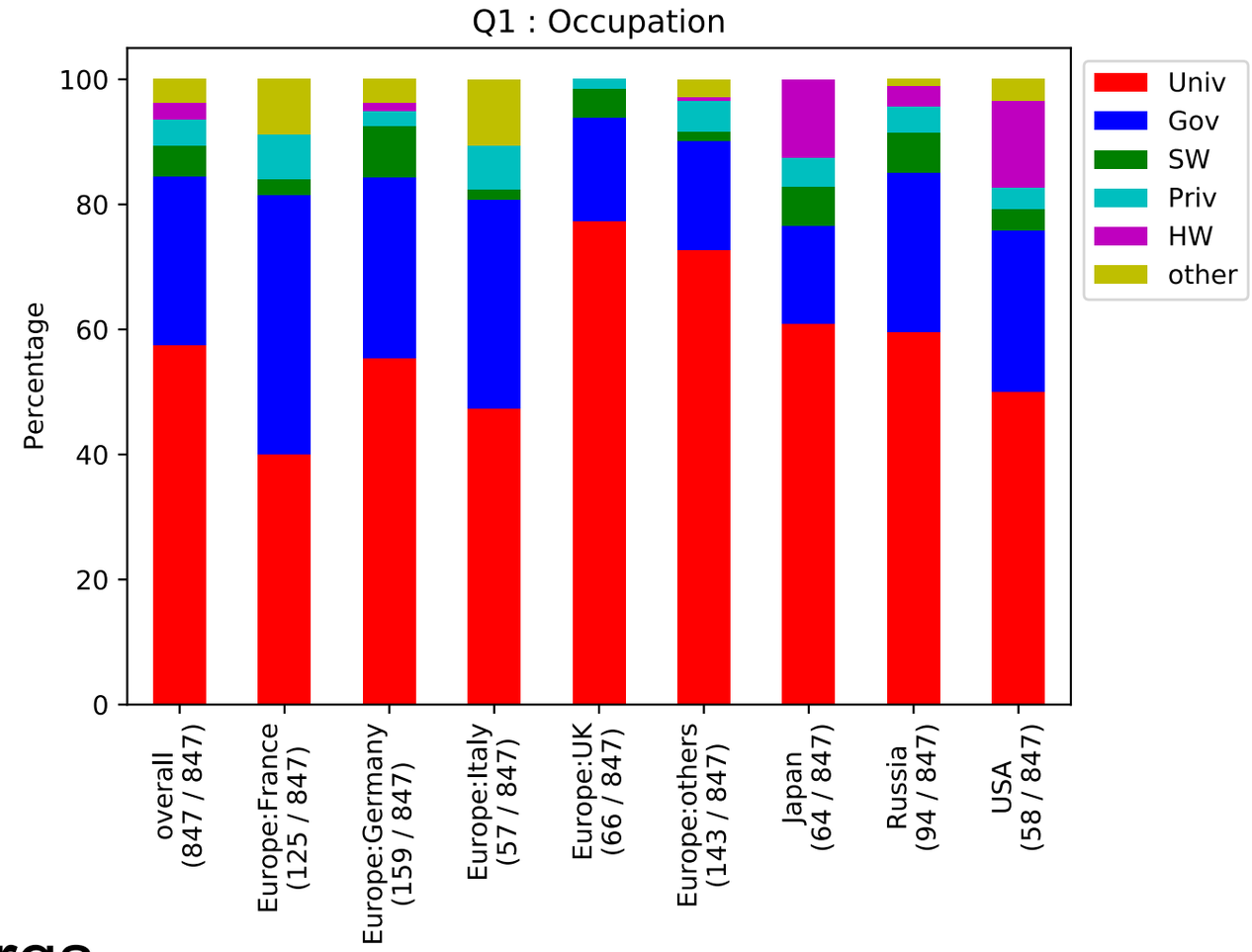
< Prev. << Index

Back Up Slides

All Questions and Simple Analysis

Q1: What is your main occupation ?

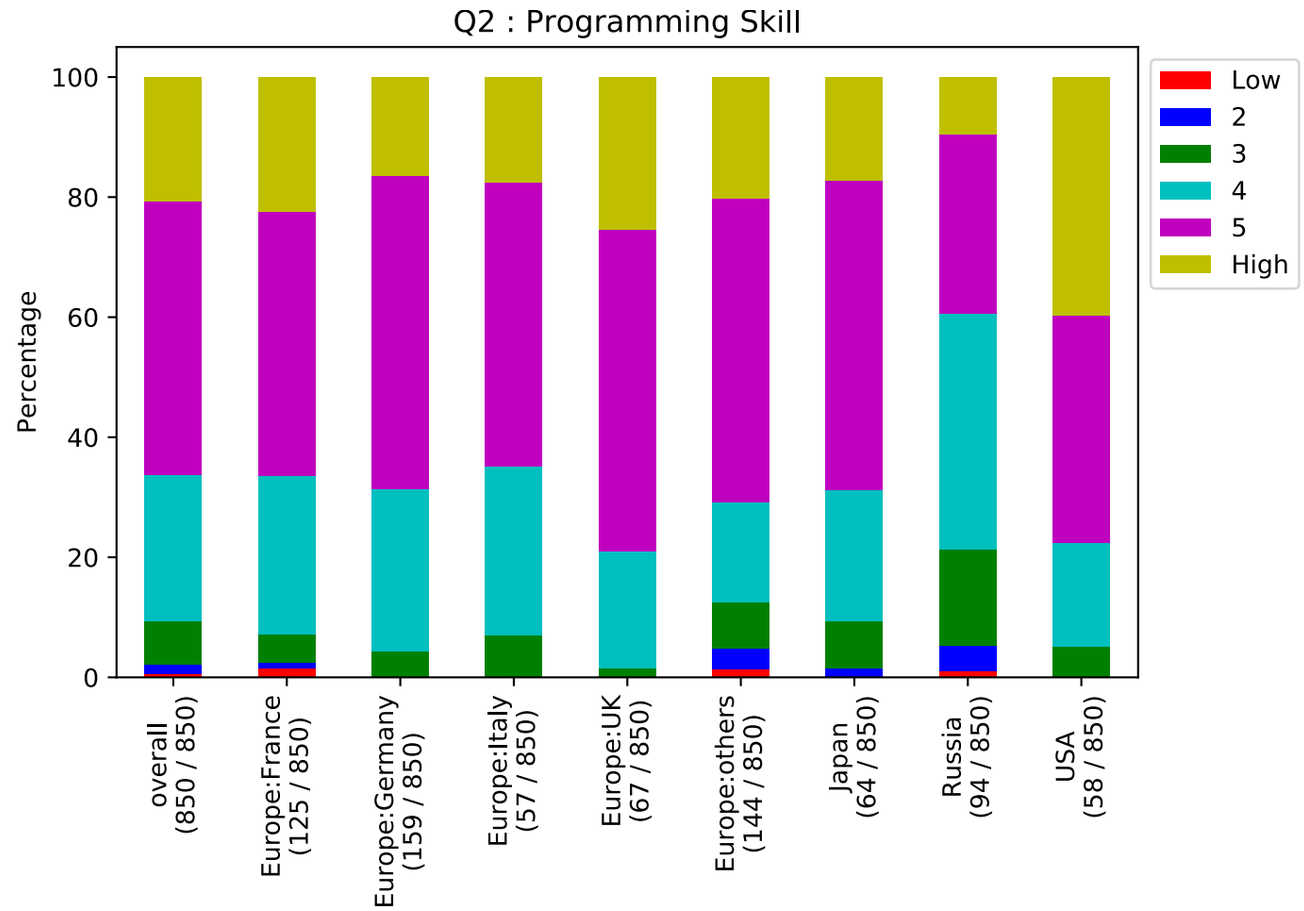
- Choice
  1. College/University
  2. Governmental institute
  3. Hardware vendor
  4. Software vendor
  5. Private research institute
  6. Other
- Fact
  - 80% comes from non-profit orgs



Q2: Rate your overall programming skill (non-MPI programs)

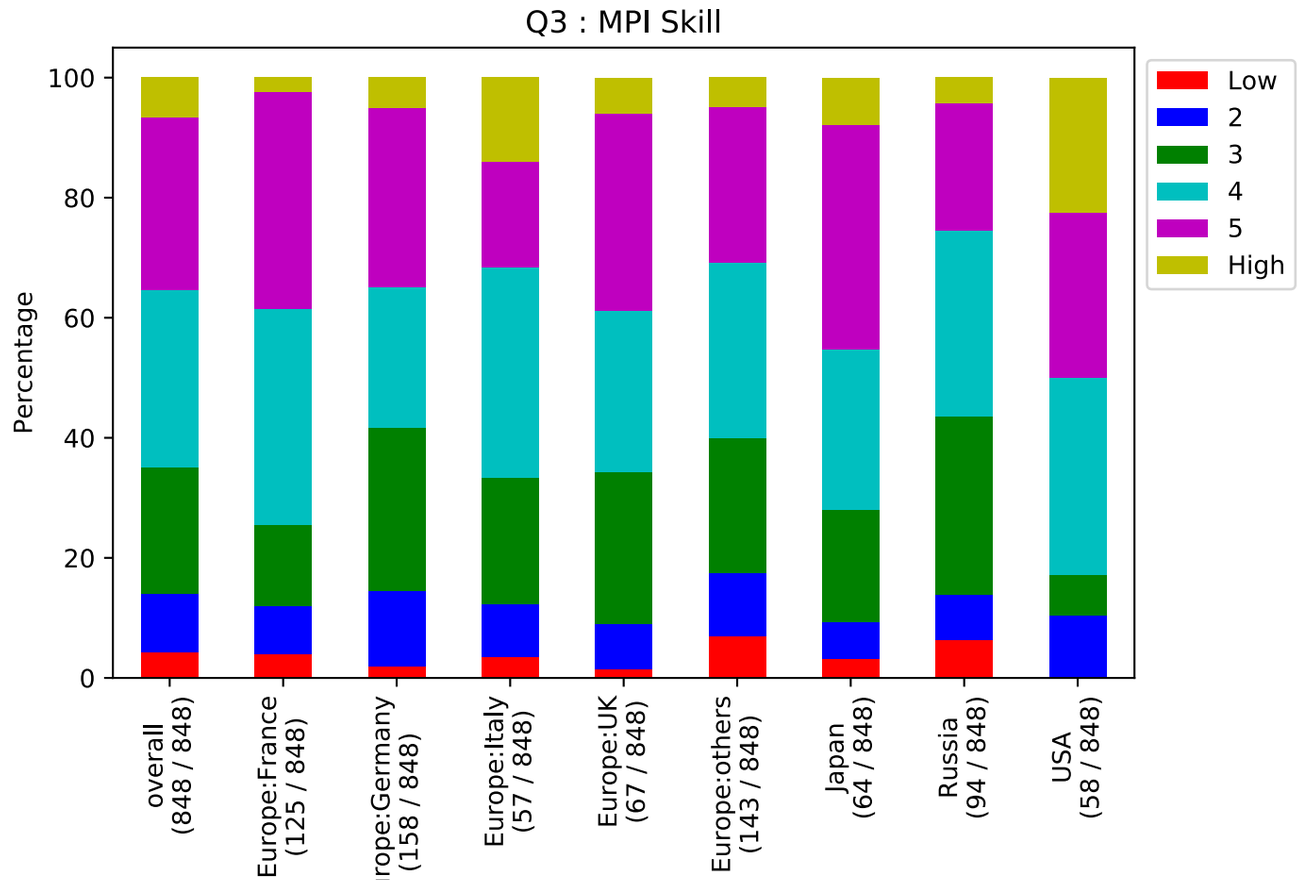
- Choice
  - from 1 (lowest) to 6 (highest)

- Findings
  - Russia is the least talented
  - USA is the most talented

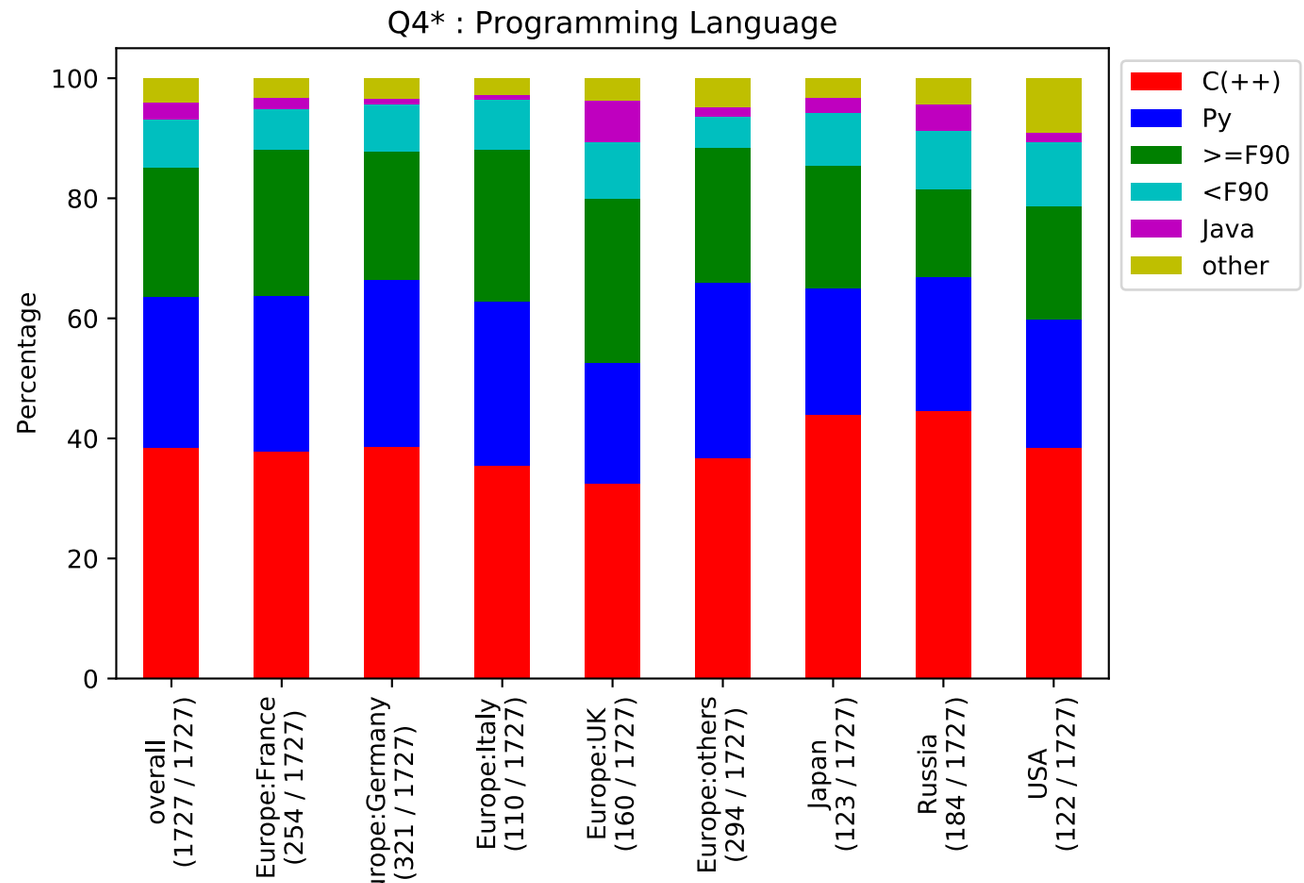


Q3: Rate your MPI programming skill.

- Choice
  - from 1 (lowest) to 6 (highest)

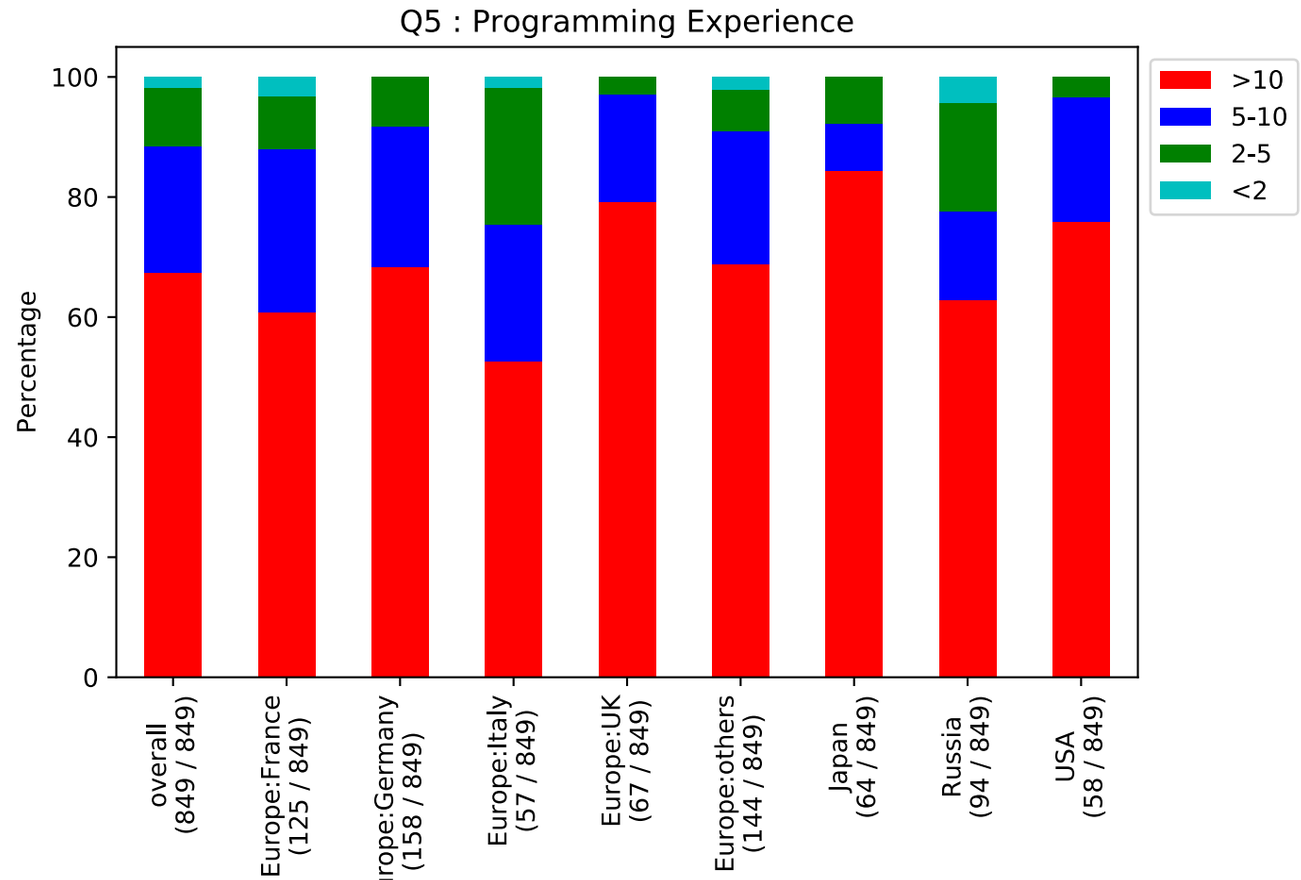


Q4: What programming language(s) do you use most often?



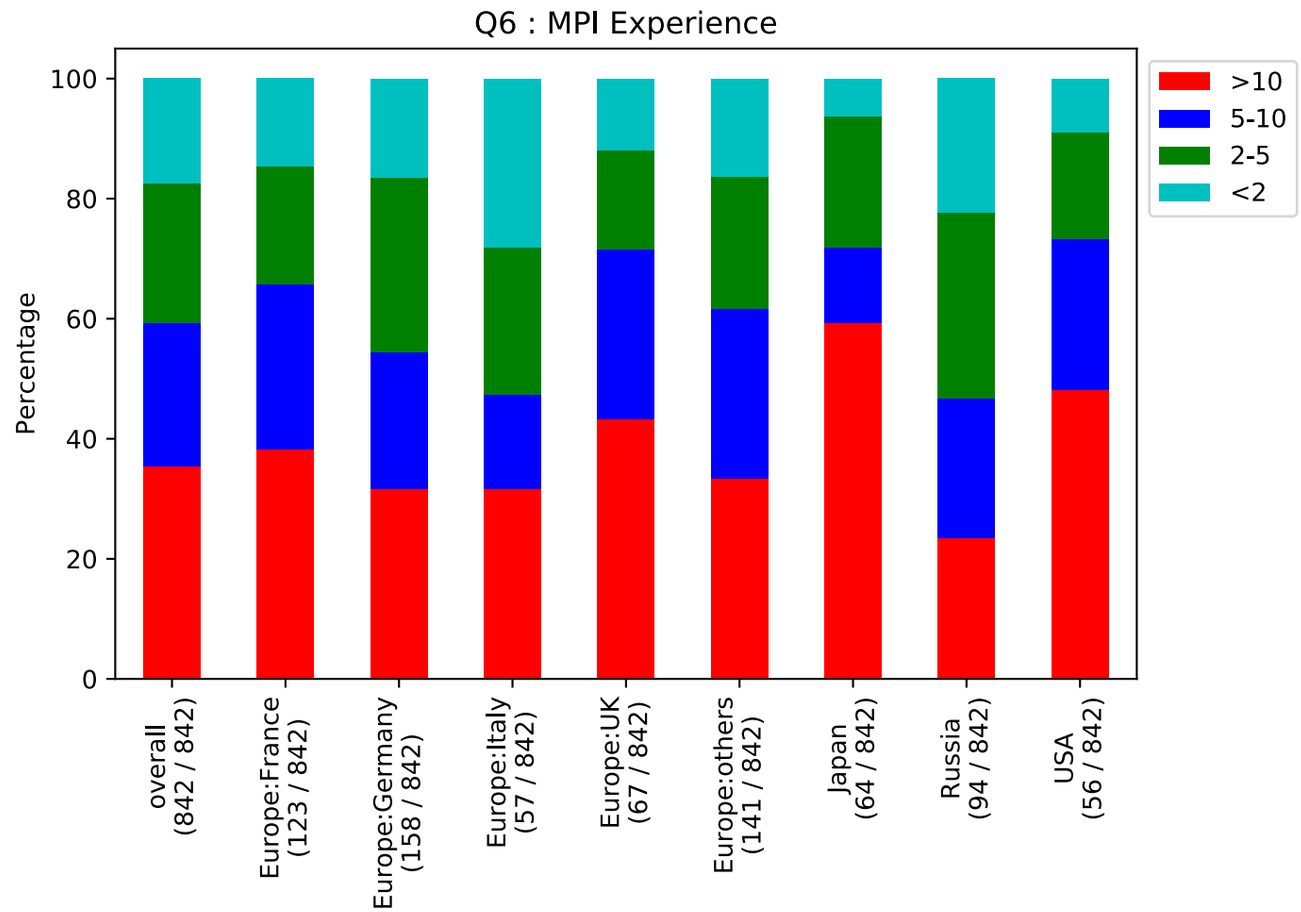
Q5: How long have you been writing computer programs (incl. non-MPI programs)?

- Choice
  1. more than 10 years
  2. between 5 and 10 years 3
  3. between 2 and 5 years 4
  4. less than 2 years
-

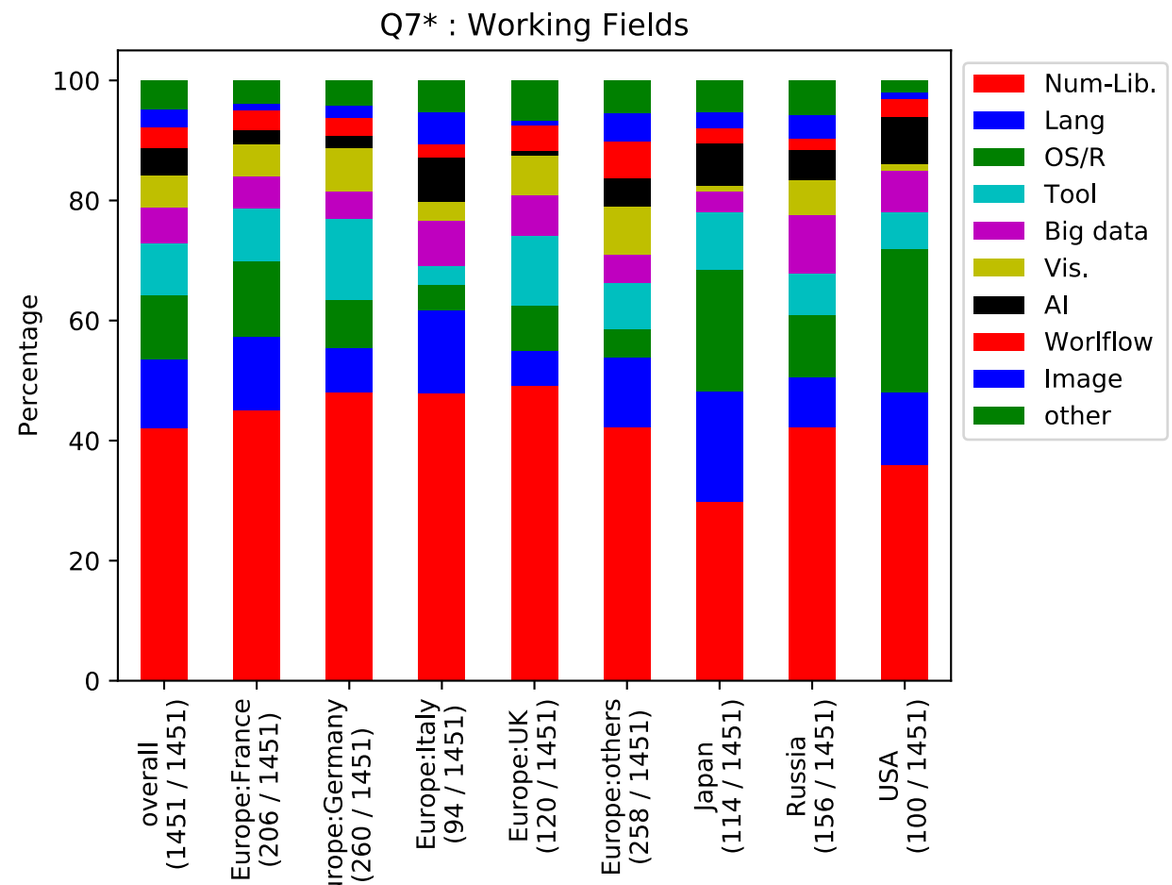


Q6: How long have you been writing MPI programs?

- Choice
  1. more than 10 years
  2. between 5 and 10 years 3
  3. between 2 and 5 years 4
  4. less than 2 years
-

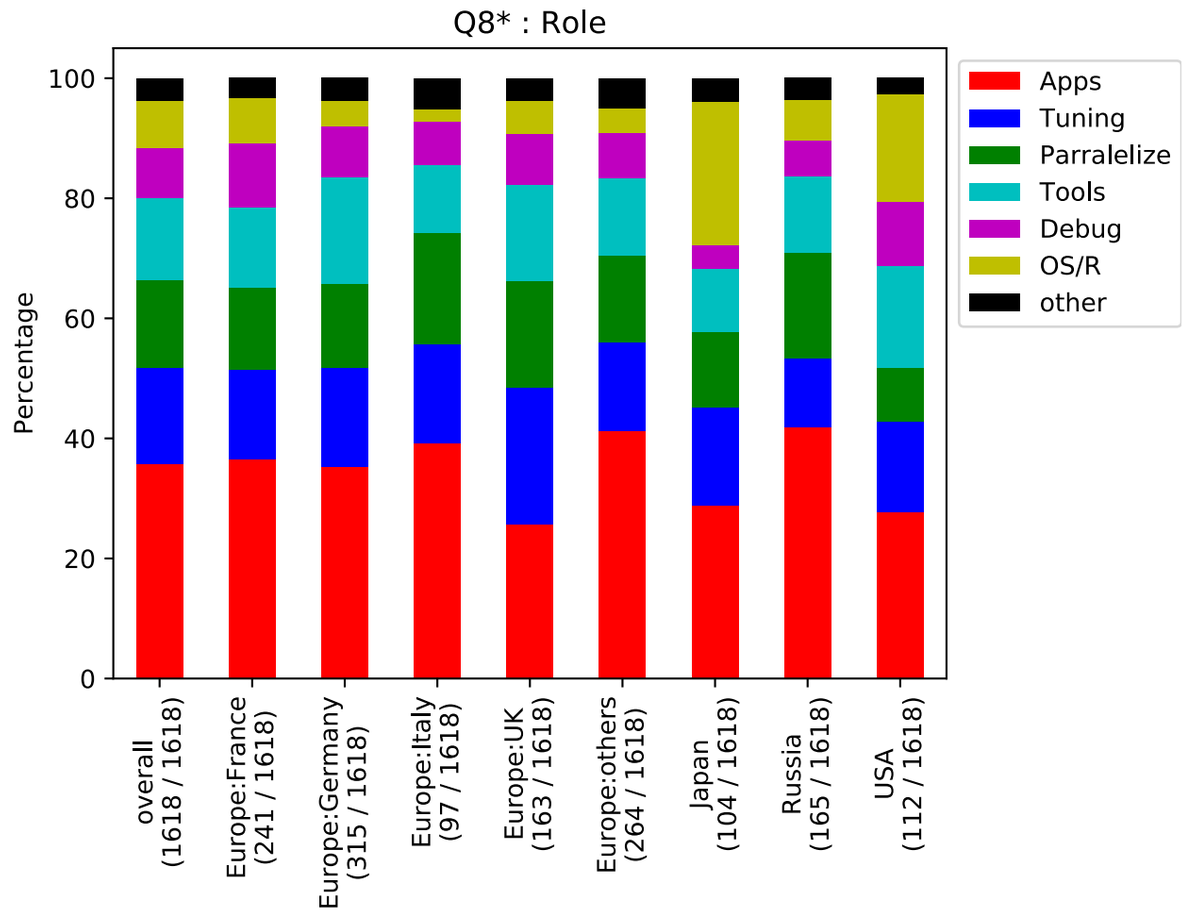


Q7: Which fields are you mostly working in?



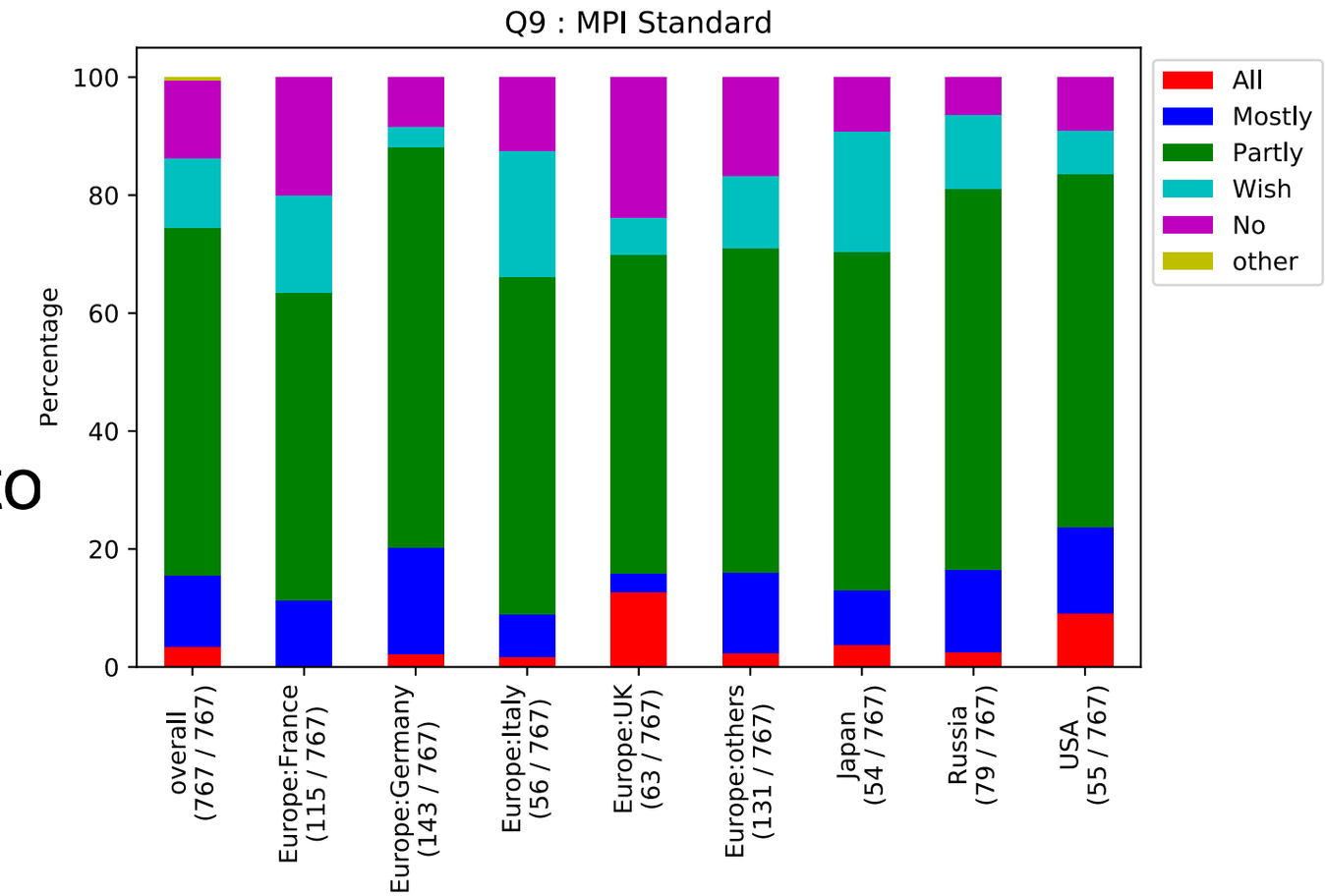
Q8: What is your major role at your place of work?

- Choices
  1. Research and development of application(s)
  2. Research and development software tool(s)
  3. Parallelization of sequential program(s)
  4. Performance tuning of MPI program(s)
  5. Debugging MPI programs
  6. Research and development on system software (OS/R)
  7. Other



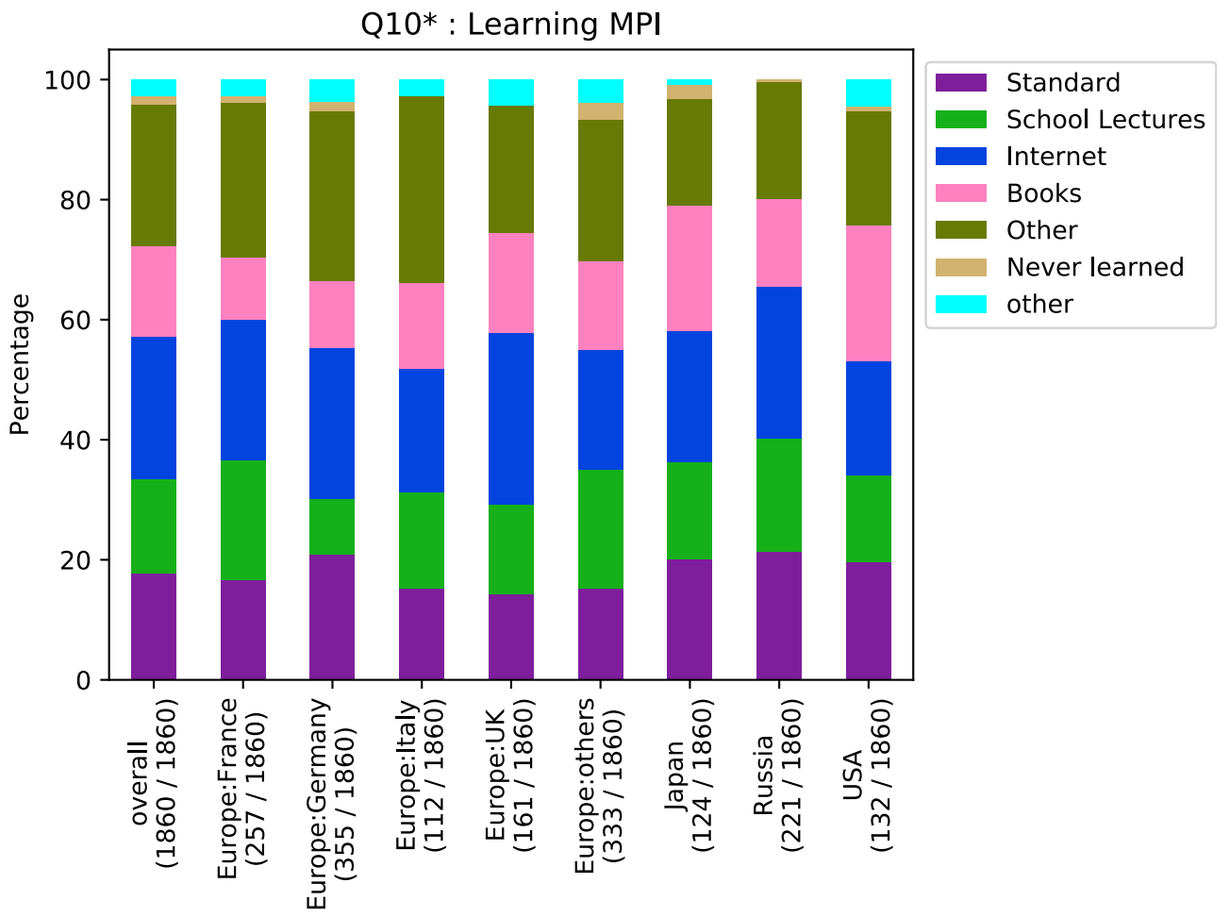
Q9: Have you ever read the MPI standard specification document?

- Choices
  1. I read all.
  2. I read most of it.
  3. I read only the chapters of interest for my work.
  4. I have not read it, but I plan to
  5. No, and I will not read it.



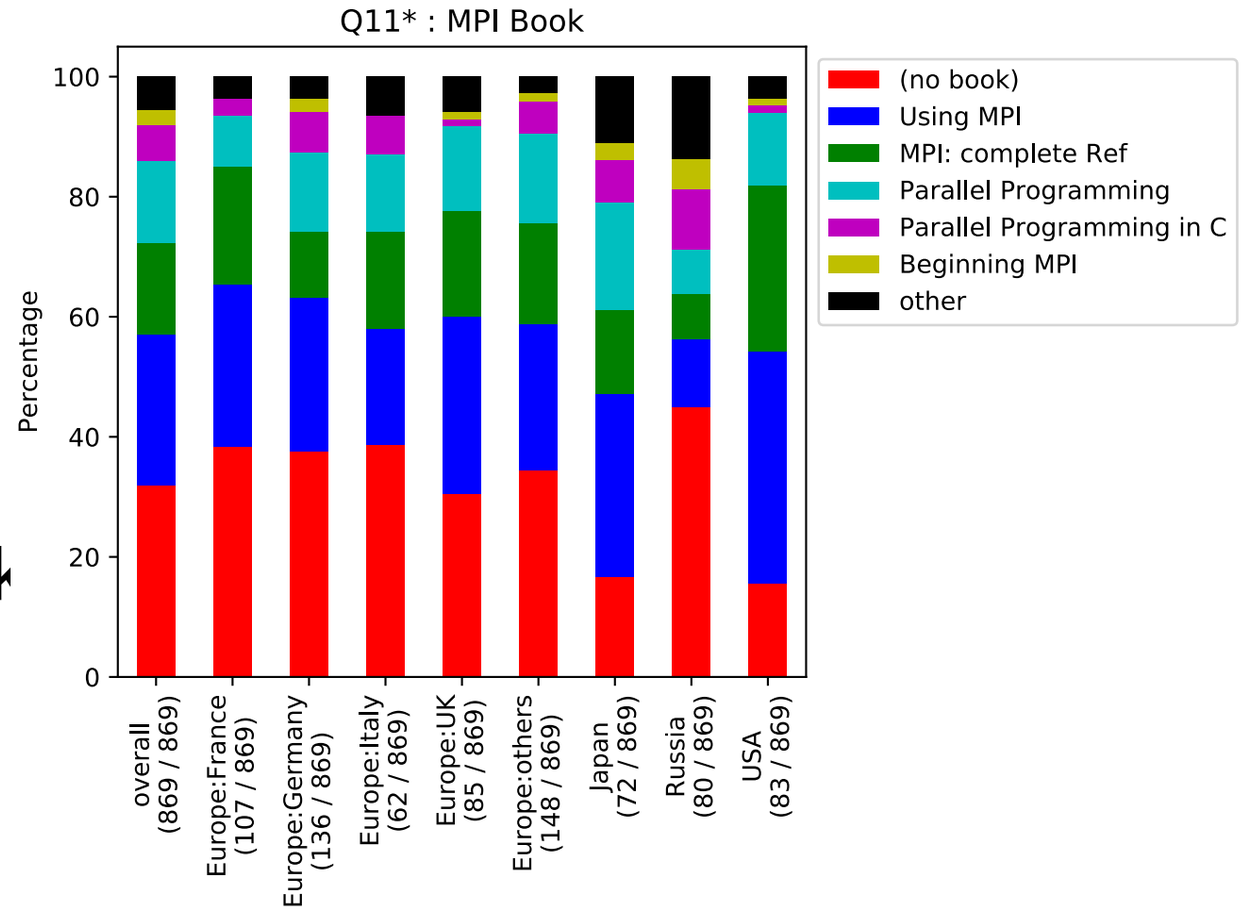
Q10: How did you learn MPI?

- Choices
  - I read the MPI standard document.
  - I had lecture(s) at school.
  - I read articles found on Internet.
  - I read book(s).
  - Other lectures or tutorials (workplace, conference).
  - I have not learned MPI.
  - Other



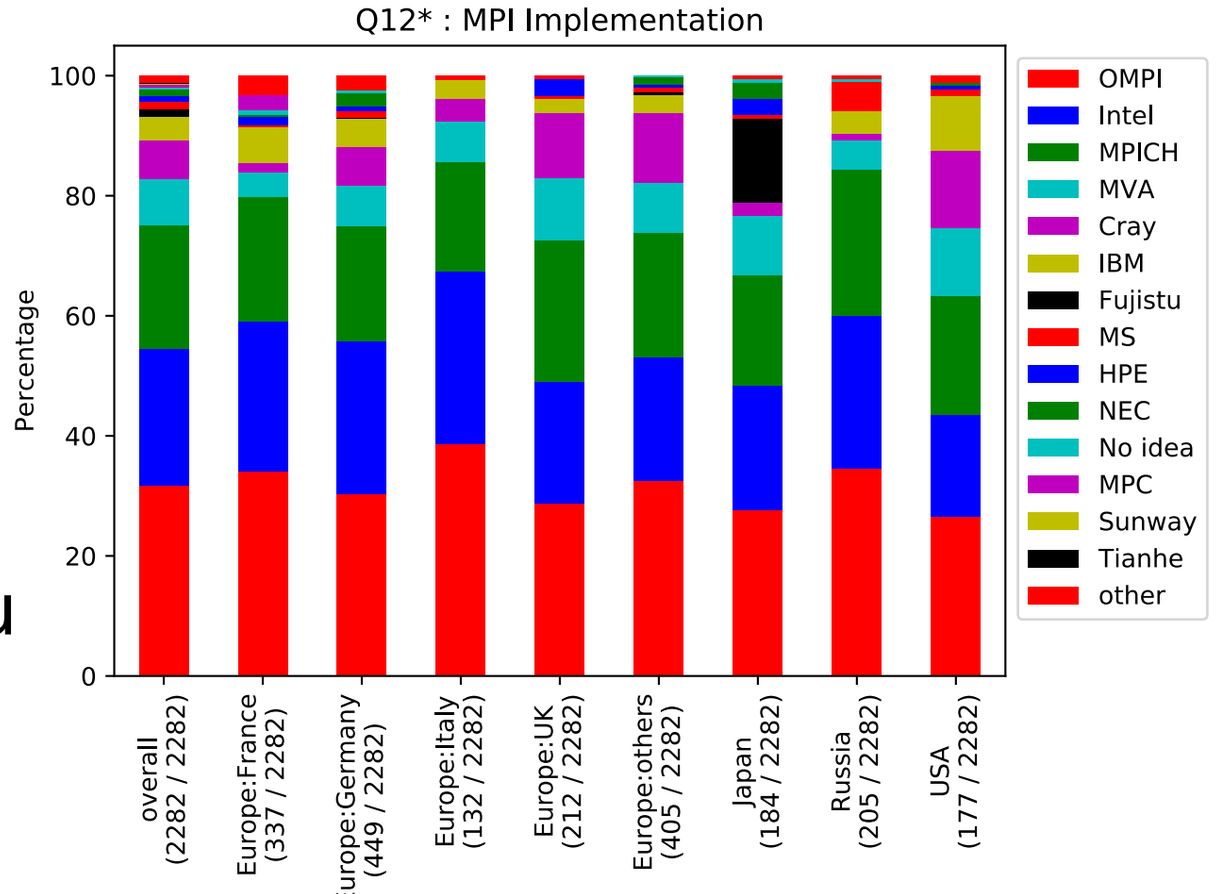
Q11: Which MPI book(s) have you read?

- Choices
  - Beginning MPI (An Introduction
  - Parallel Programming with MPI
  - Using MPI
  - Parallel Programming in C with and OpenMP
  - MPI: The Complete Reference
  - I have never read any MPI book
  - Other



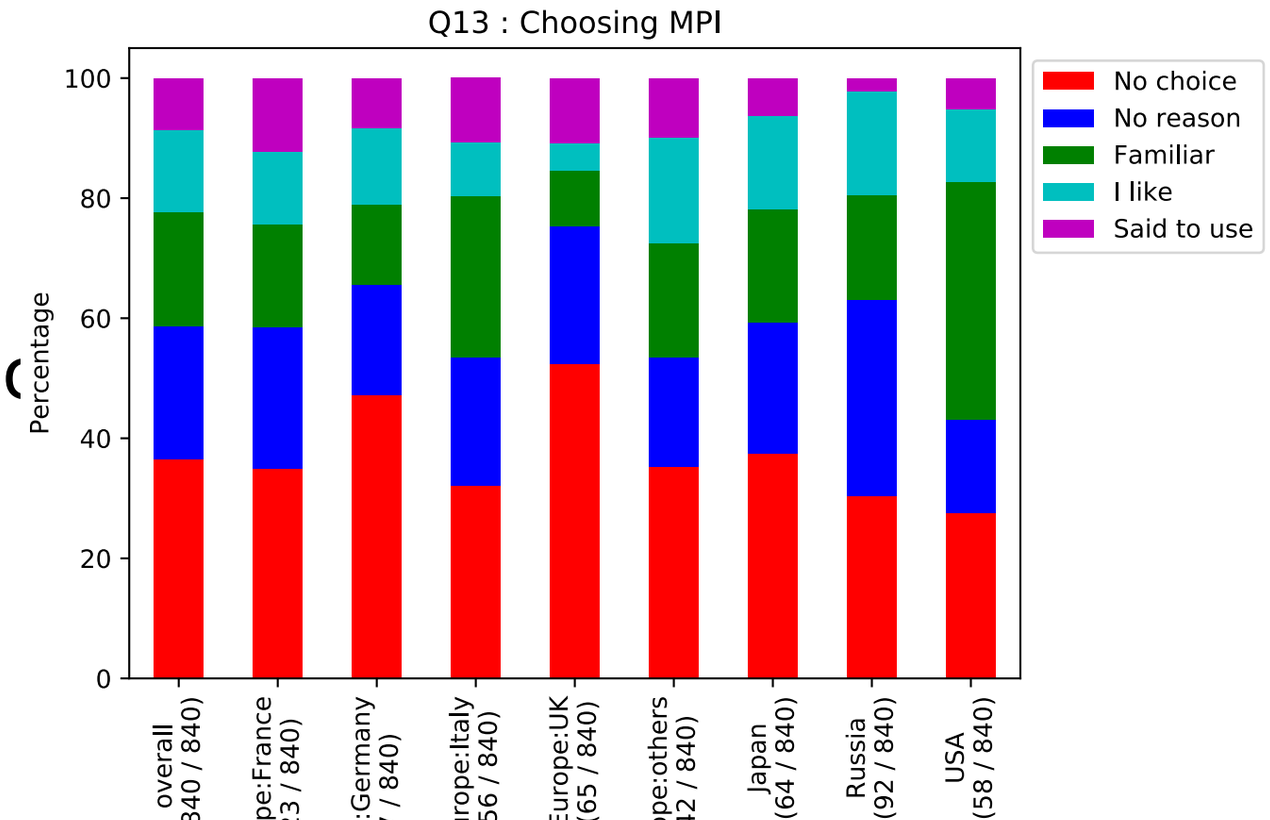
Q12: Which MPI implementations do you use?

- Choices
  - MPICH
  - Open MPI
  - Intel MPI
  - MVAPICH
  - Cray MPI
  - IBM MPI (BG/Q, PE, Spectru
  - HPE MPI
  - Tianhe MPI
  - Sunway MPI
  - Fujistu MPI
  - NEC MPI
  - MS MPI
  - MPC MPI
  - I do not know
  - Other



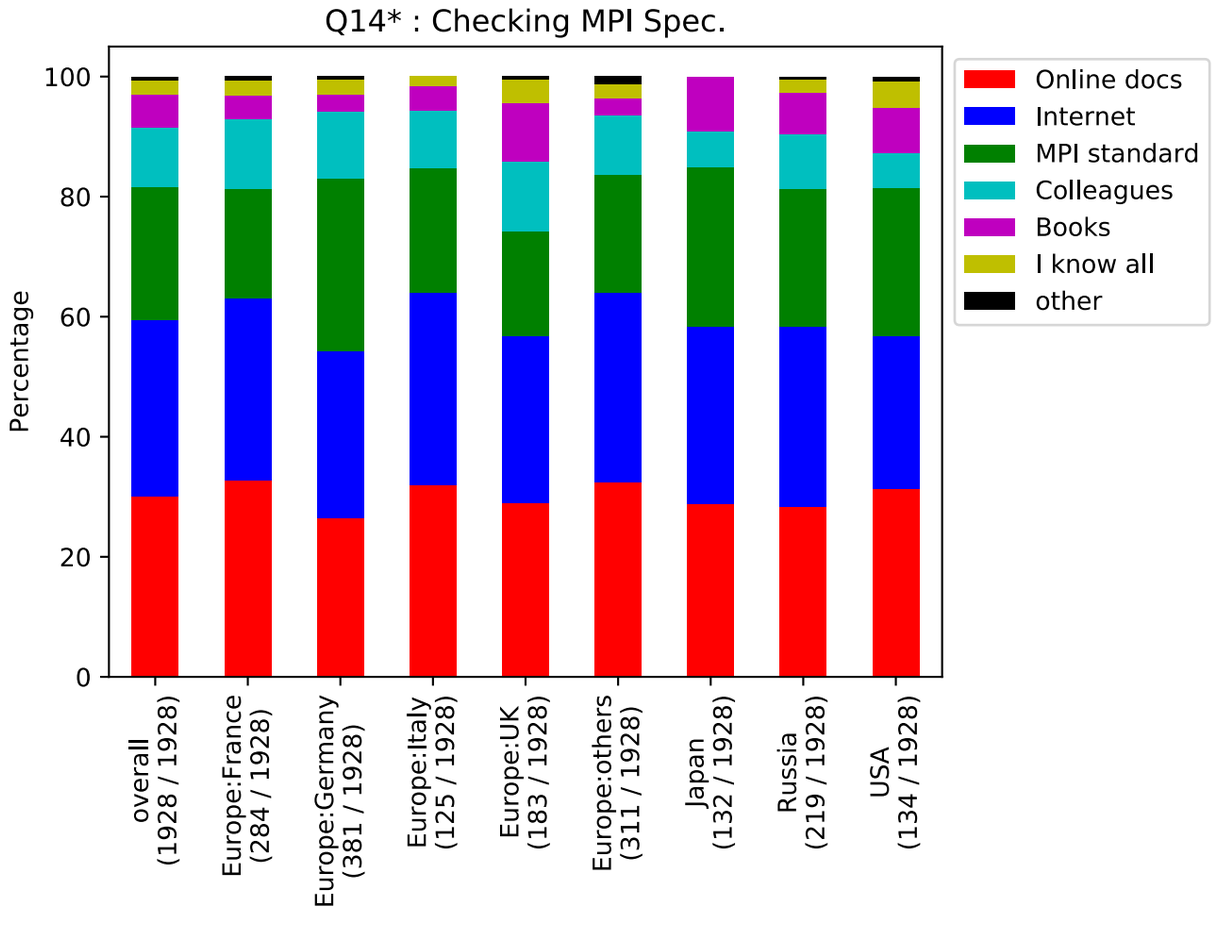
Q13: Why did you choose the MPI implementation(s)?

- Choices
  - I like to use it.
  - I was said to use it.
  - I could not have any choice (the one provided by a vendo
  - I am familiar with it.
  - I have no special reason.



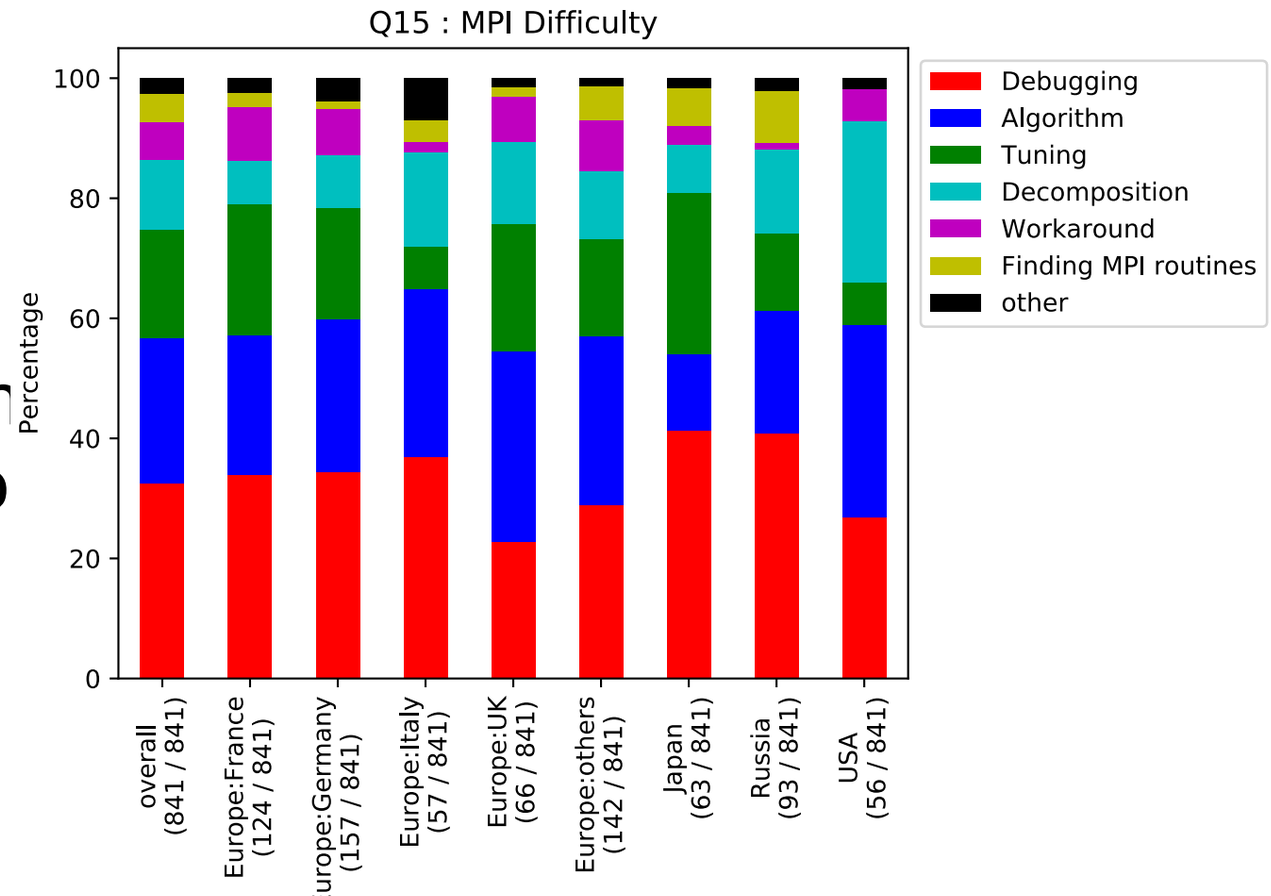
Q14: How do you check MPI specifications when you are writing MPI pro- grams?

- Choices
  - I read the MPI Standard document (web/book)
  - I read online documents (such as man pages)
  - I search the Internet (Google / Stack Overflow)
  - I ask colleagues.
  - I read book(s) (except the MPI sta
  - I know almost all MPI routines
  - Other



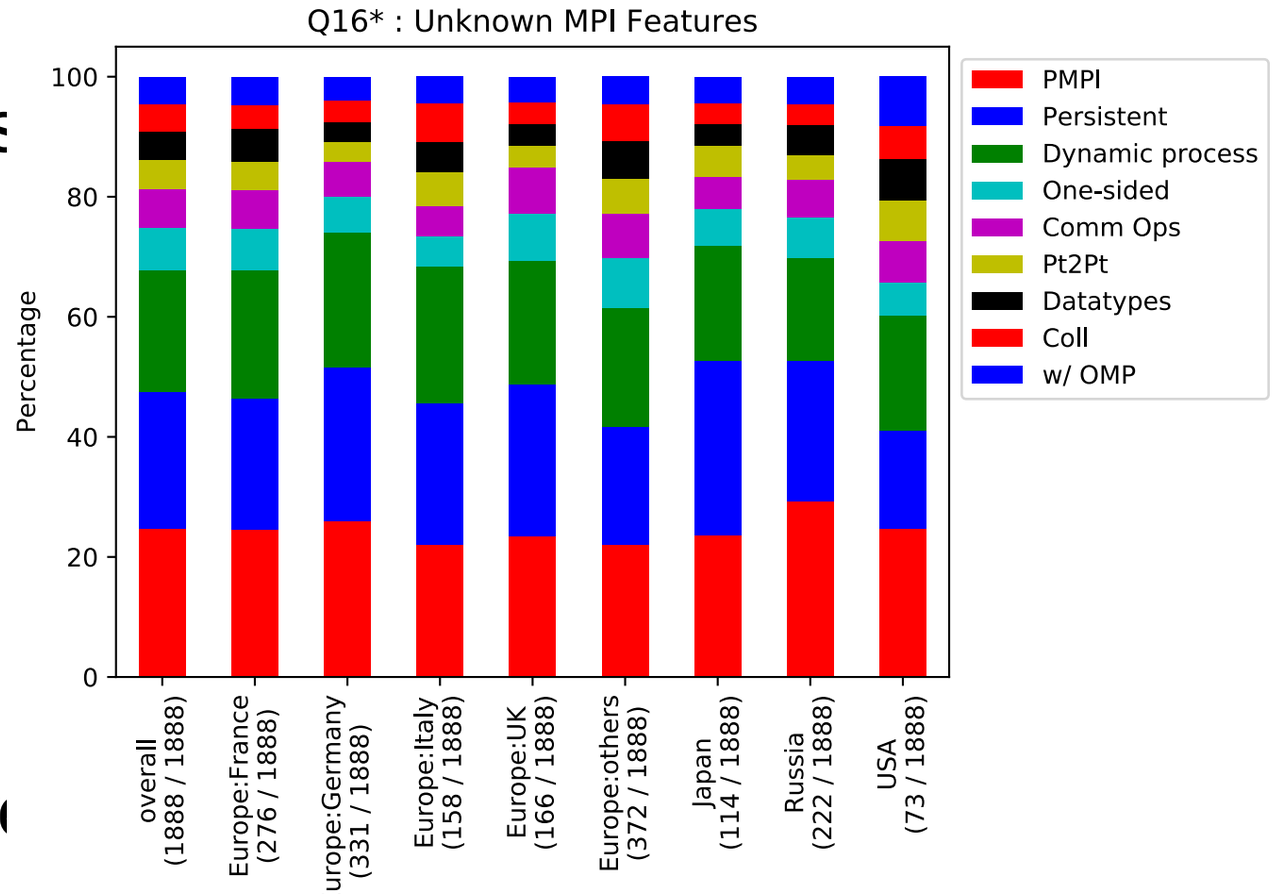
Q15: What is the most difficult part of writing an MPI program?

- Choices
  - Algorithm design
  - Debugging
  - Domain decomposition
  - Finding appropriate MPI routin
  - Implementation issue workaro
  - Performance tuning
  - Other



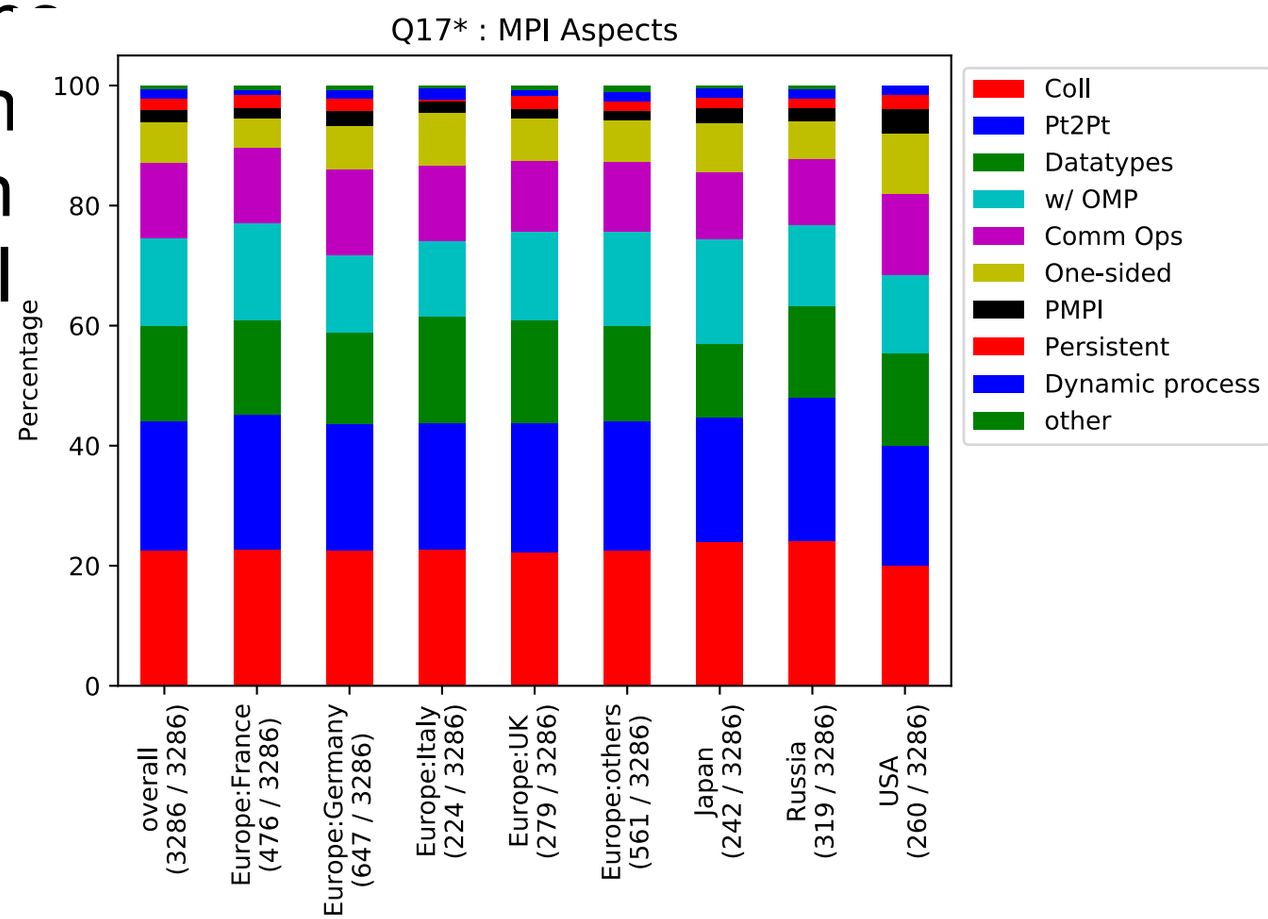
Q16: Which MPI features have you never heard of?

- Choices
  - Point-to-point communications
  - Collective communications
  - Communicator operations
  - MPI datatypes
  - One-sided communications
  - Dynamic process creation
  - Persistent communication
  - PMPI interface
  - MPI with OpenMP (or multithr
  - Other



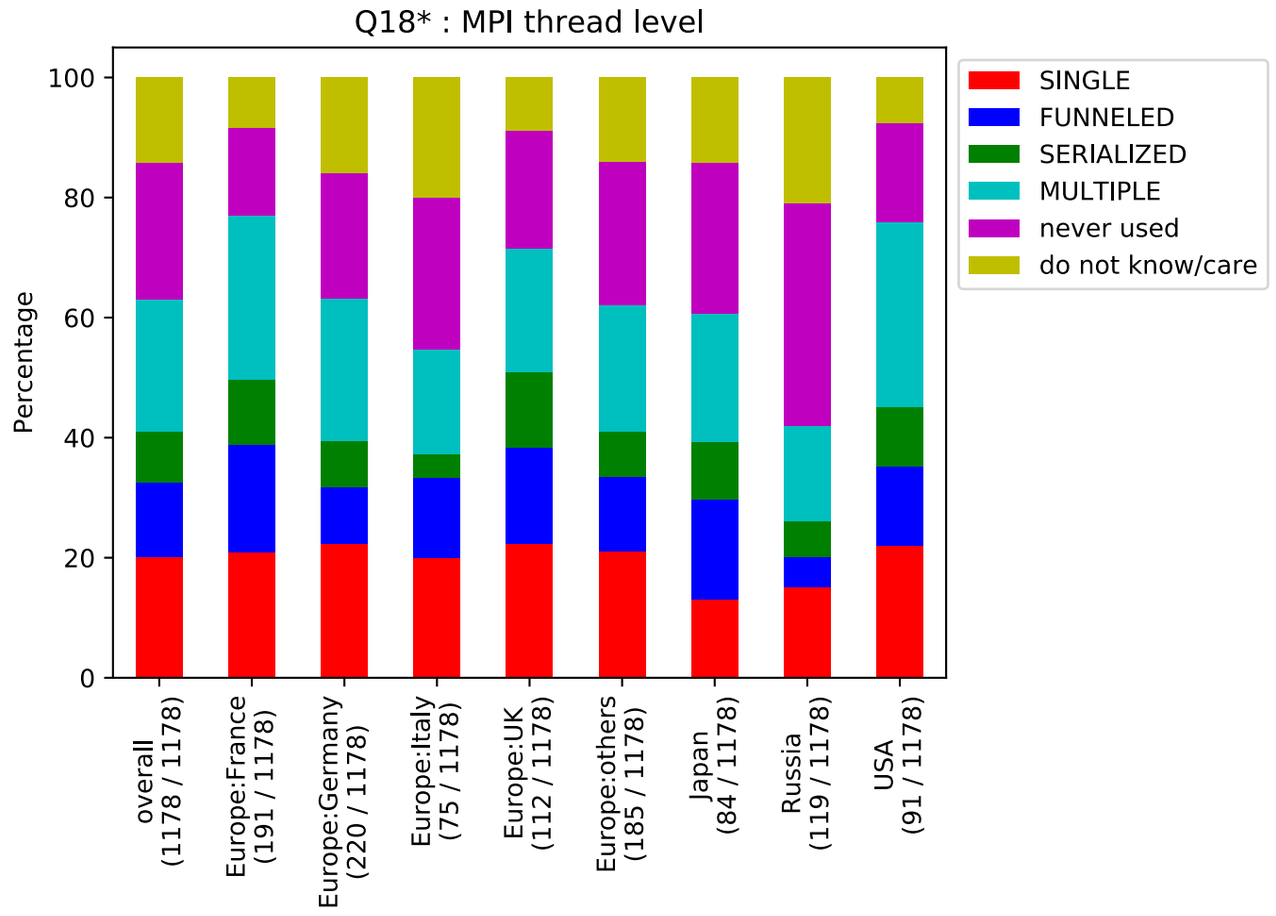
Q17: What aspects of the MPI standard do you use in your program in its current form?

- Choices
  - Point-to-point communications
  - Collective communications
  - Communicator operations (split, duplicate, and so on)
  - MPI datatypes
  - One-sided communicatior
  - Dynamic process creation
  - Persistent communication
  - MPI with OpenMP (or mul
  - PMPI interface
  - Other



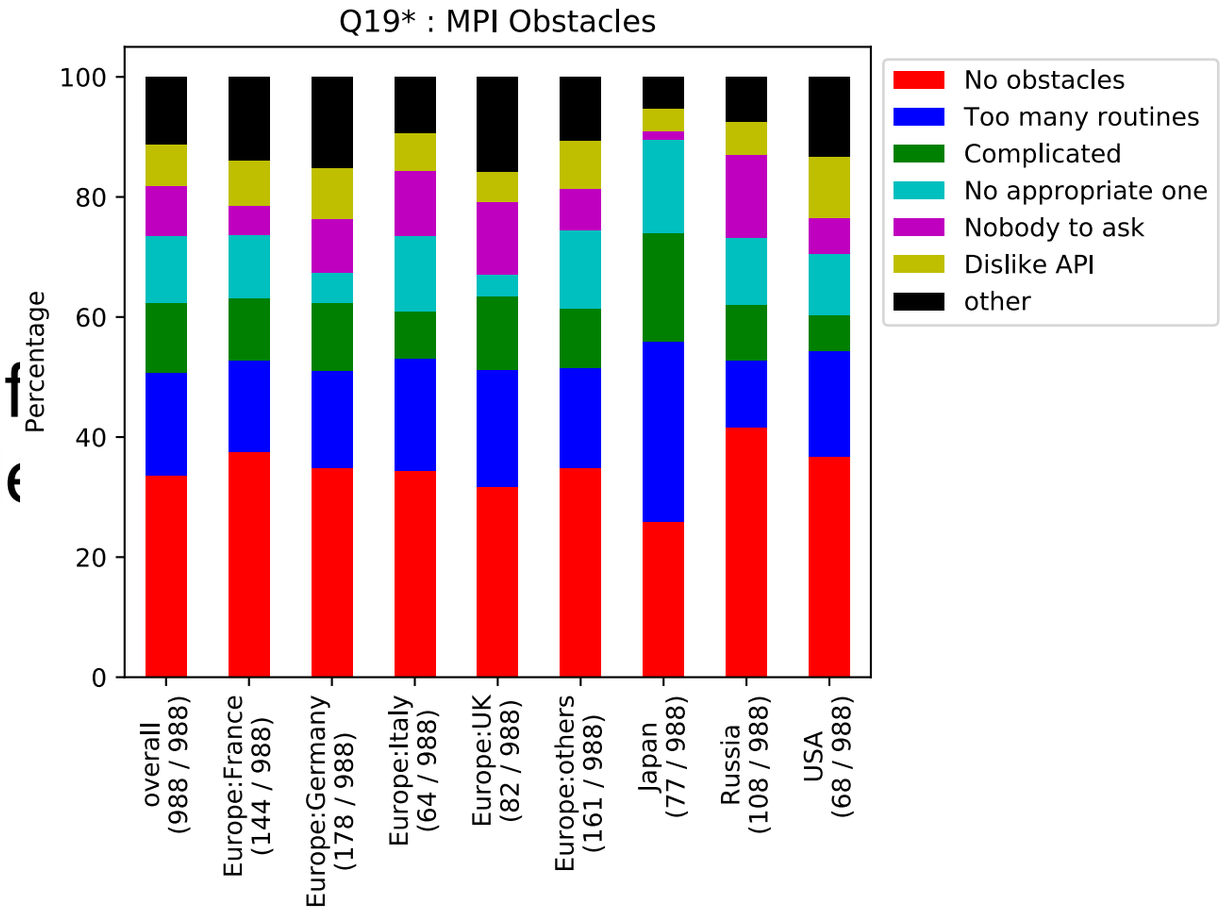
Q18: Which MPI thread support are you using?

- Choices
  - MPI_THREAD_SINGLE
  - MPI_THREAD_FUNNELED
  - MPI_THREAD_SERIALIZED
  - MPI_THREAD_MULTIPLE
  - I have never called MPI_INIT_THREAD
  - I do not know or I do not care
  - Other
  -



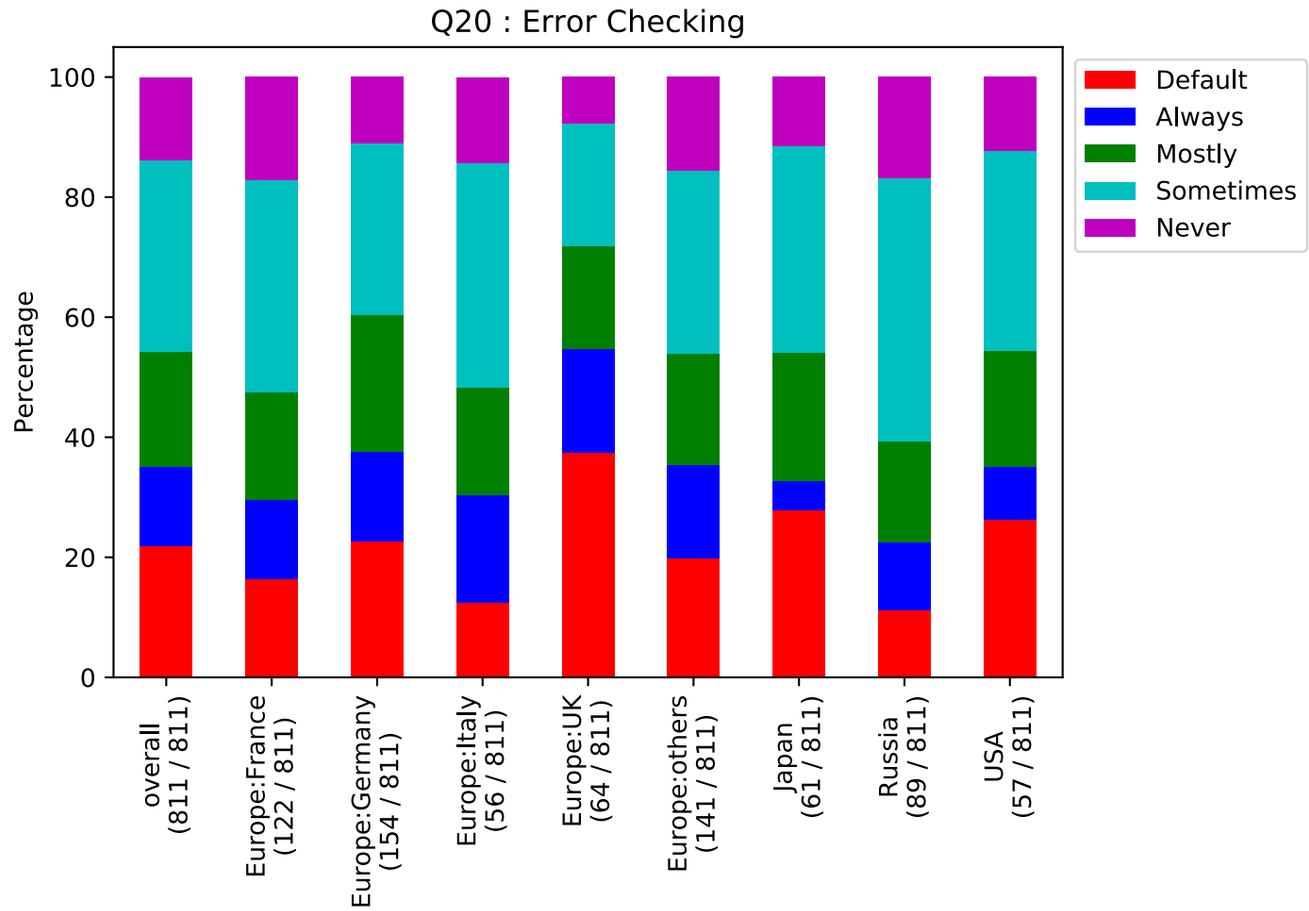
Q19: What are your obstacles to mastering MPI?

- Choices
  - I have no obstacles
  - Too many routines
  - No appropriate lecture / book / inf
  - Too complicated and hard to unde
  - I have nobody to ask
  - I do not like the API
  - Other



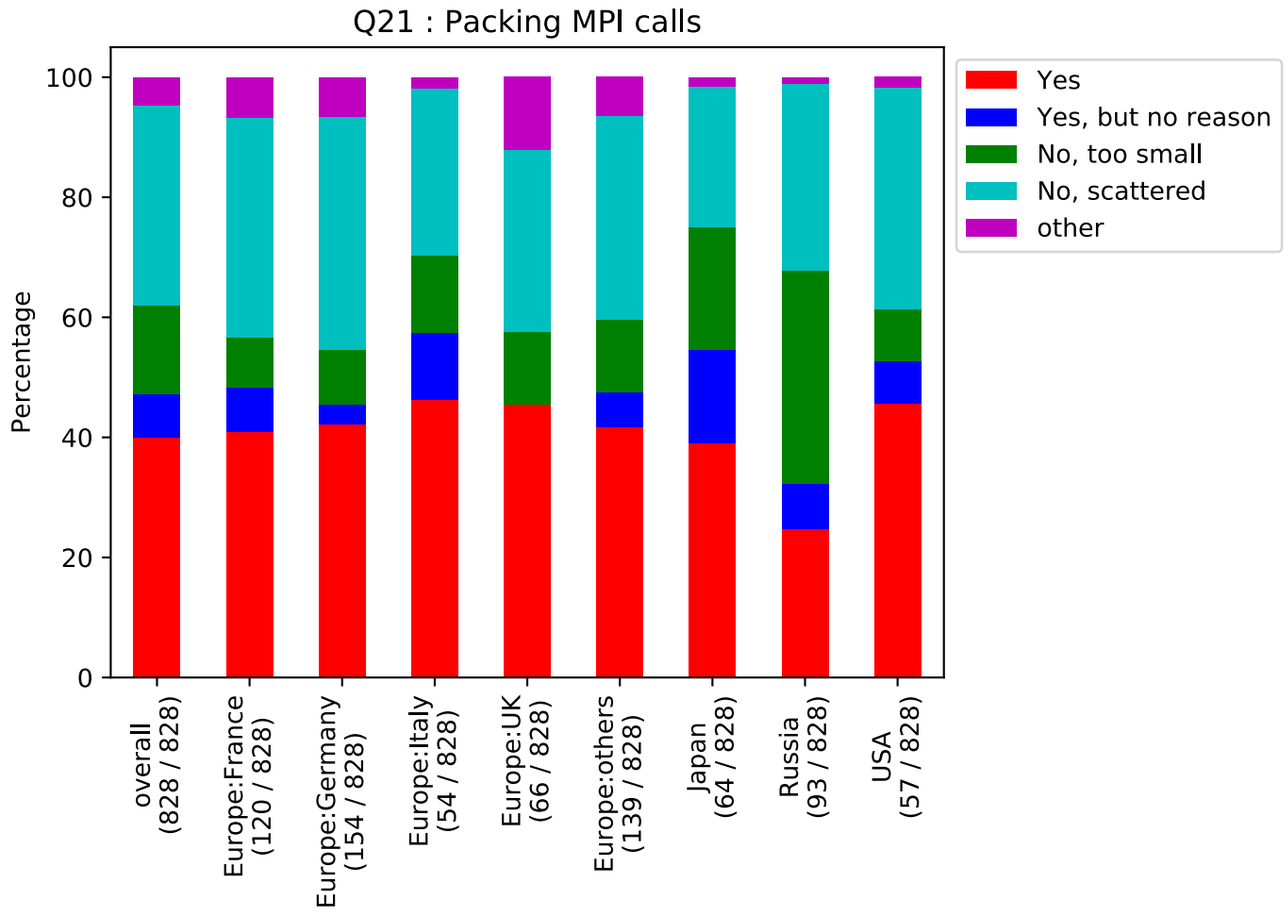
Q20: When you call an MPI routine, how often do you check the error code of the MPI routine (excepting MPI-IO)?

- Choices
  - I rely on the default ‘Errors abort’ error handling
  - Always
  - Mostly
  - Sometimes
  - Never
  - Other



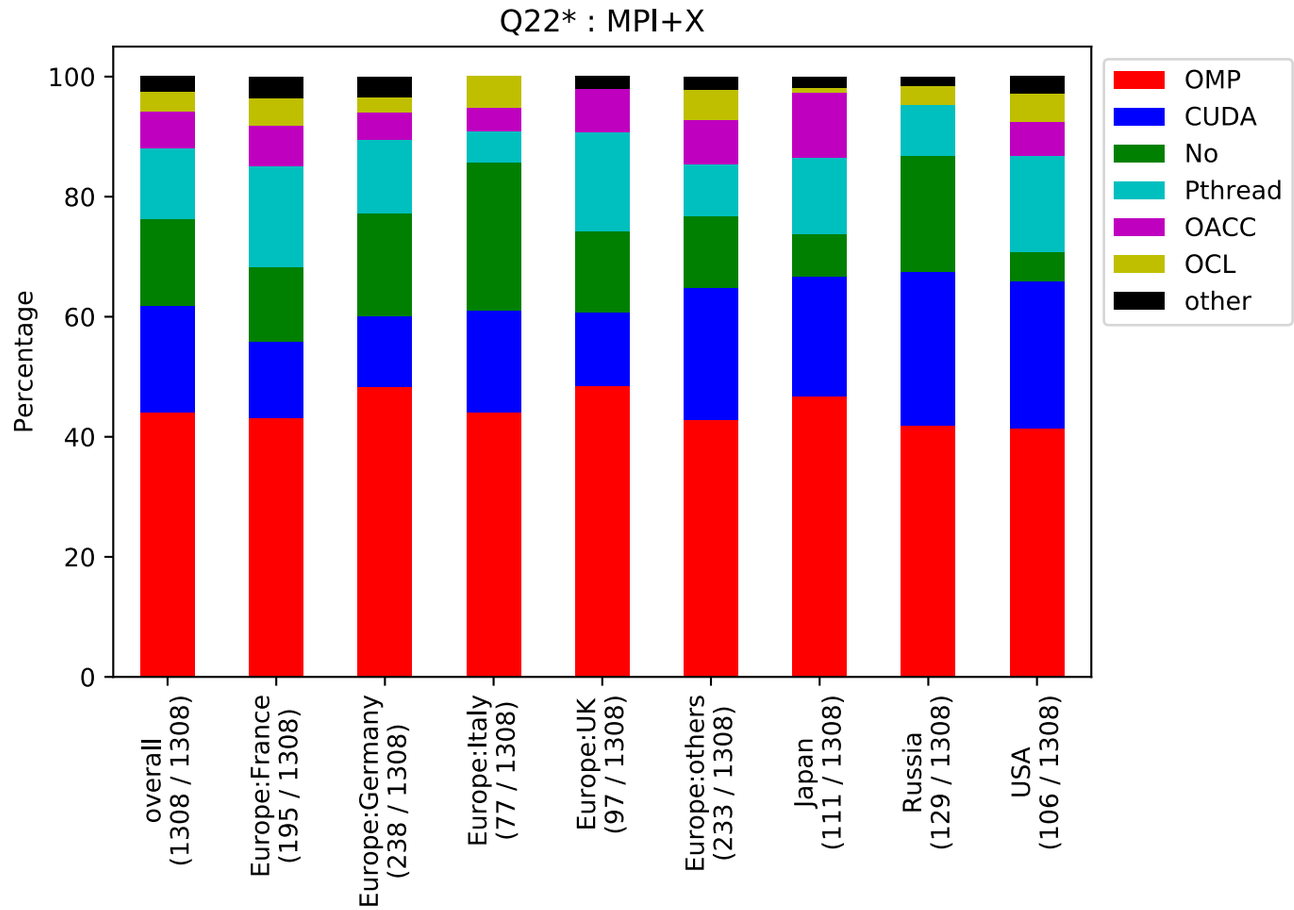
Q21: In most of your programs, do you pack MPI function calls into their own file or files to have your own abstraction layer for communication?

- Choices
  - Yes, to minimize the changes of communication API
  - Yes, but I have no special reason for doing that
  - No, my program is too small to do that
  - No, MPI calls are scattered in my programs
  - Other
  -



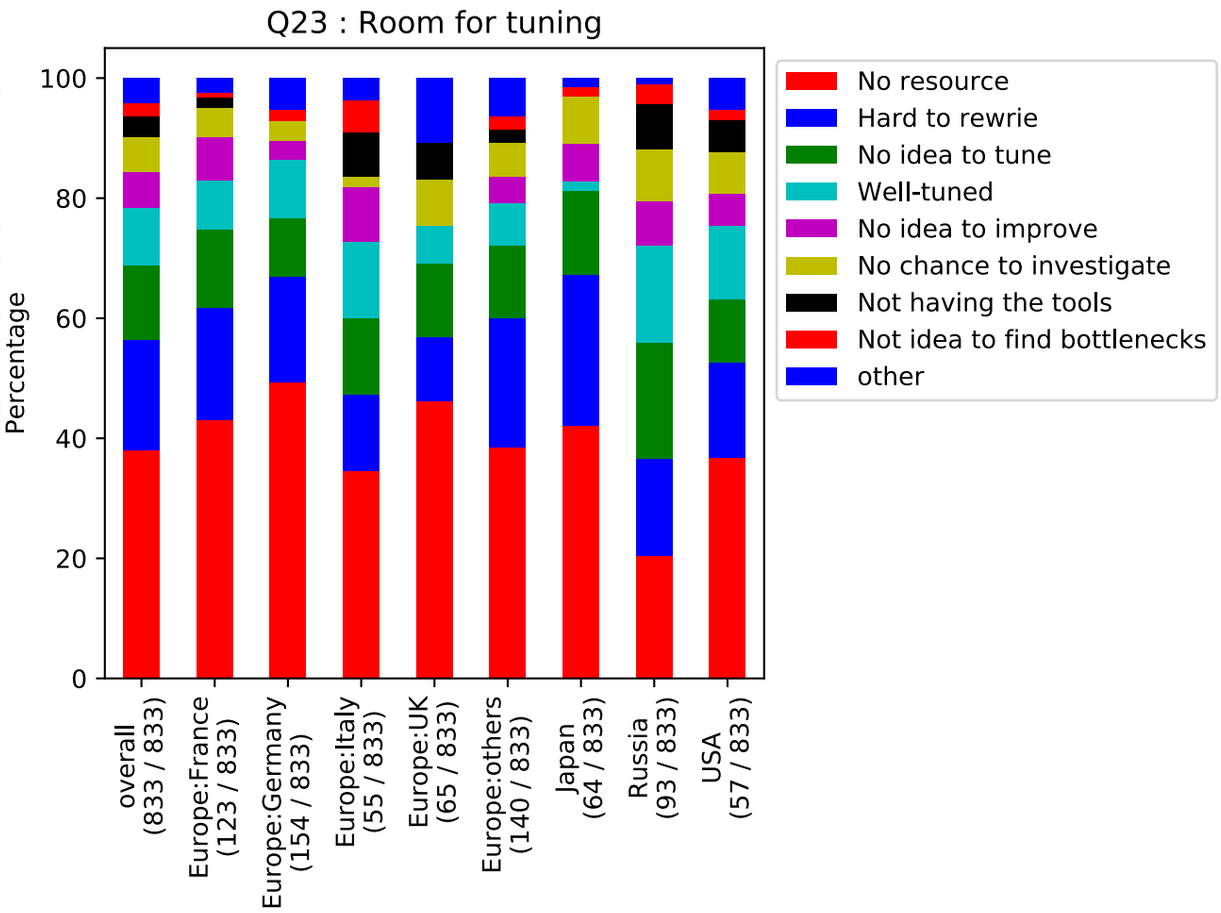
Q22: Have you ever written MPI+”X” programs?

- Choices
  - OpenMP
  - Pthread
  - OpenACC
  - OpenCL
  - CUDA
  - No
  - Other



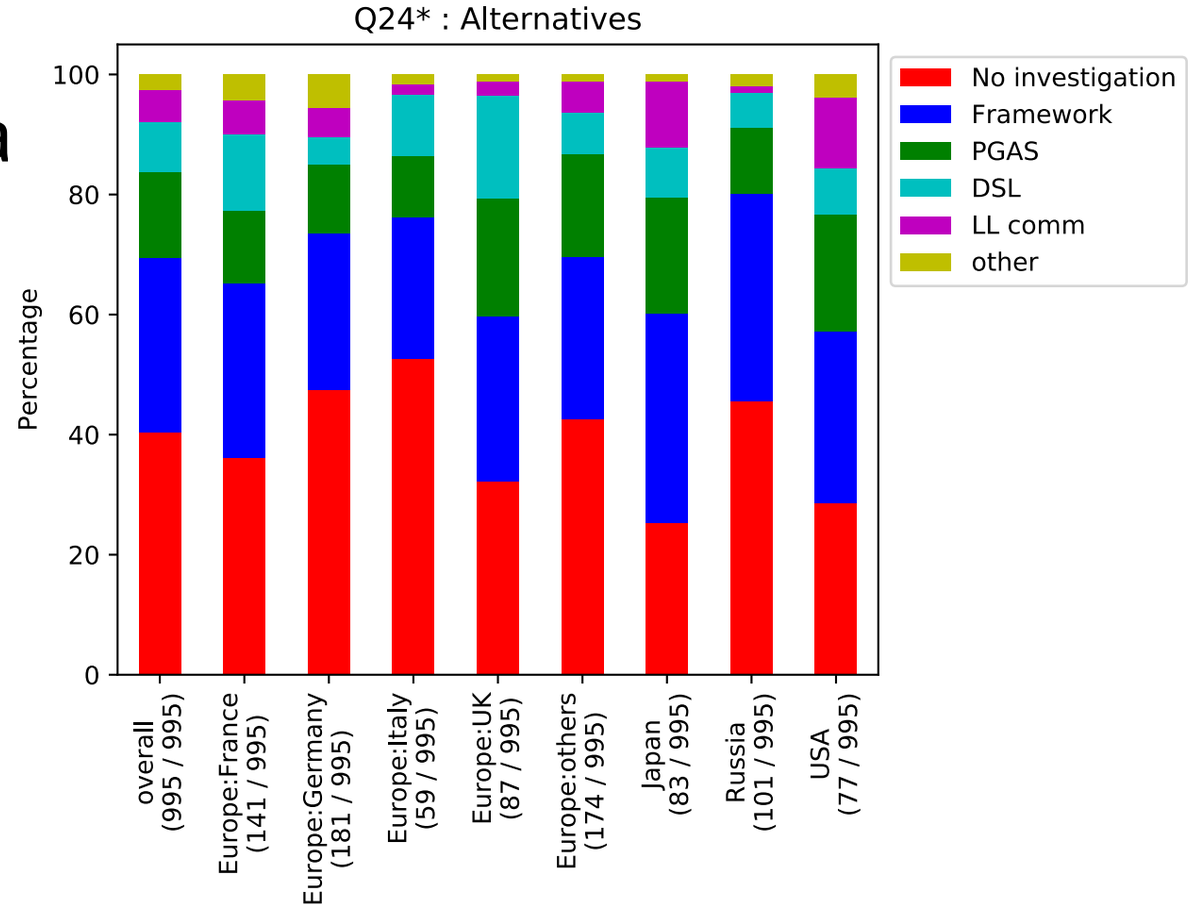
Q23: Is there any room for performance tuning in your MPI programs?

- Choices
  - No, my MPI programs are well-tuned.
  - Yes, I know there is room for tuning but I should re-write large part of my program to do that.
  - Yes, I know there is room for tuning but I do not have enough resources to do that.
  - I think there is room but I c
  - I do not have (know) tools
  - I have no chance to invest
  - I do not know how to find b
  - I do not know if there is ro
  - Other



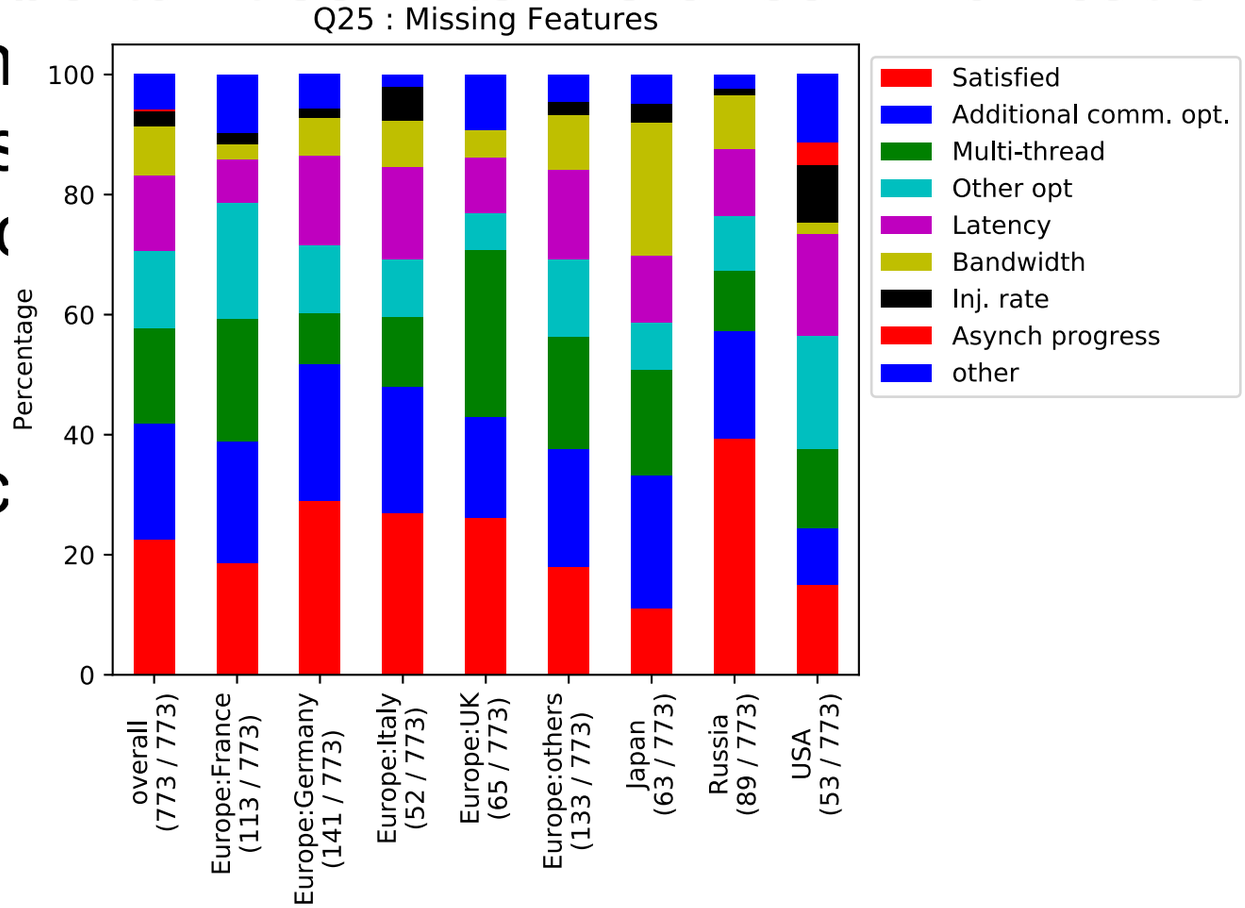
Q24: What, if any, alternatives are you investigating to indirectly call MPI or another communication layer by using another parallel language/library?

- Choices
  - A framework or library using MPI
  - A PGAS language
  - A Domain Specific Language (DSL).
  - Low-level communication layer provided by vendor (Verbs, DCMF, ...)
  - I am not investigating any a
  - Other



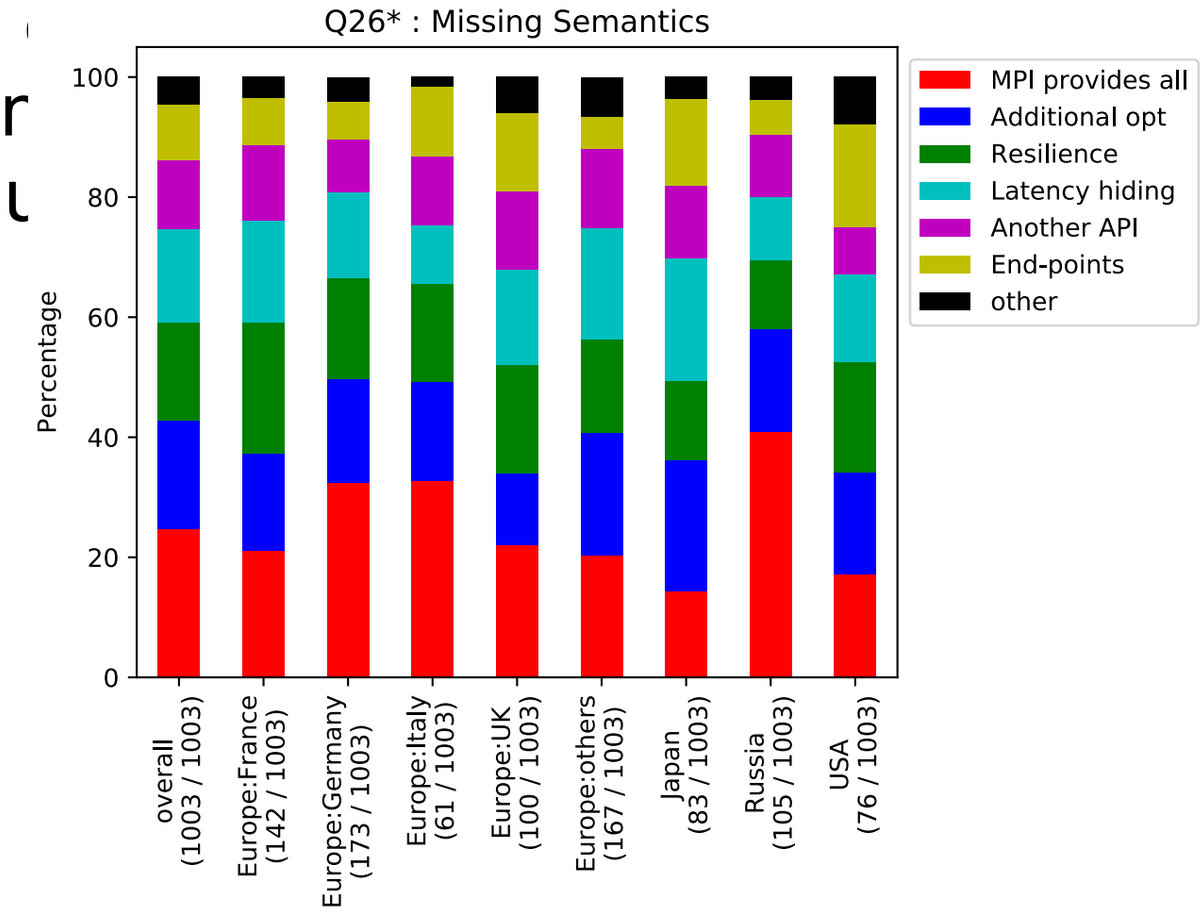
Q25: If there were one communication aspect which is not enough in the current MPI could improve the performance of your application, what would you prioritize? Or is MPI providing all the communication semantics required by your application? If not, what is missing?

- Choices
  - Latency
  - Message injection rate
  - Bandwidth
  - Additional optimization opportunities in terms of communication (network topology awaren
  - Optimization opportunities aware-ness, dynamic proc
  - Multi-threading support
  - Asynchronous progress
  - MPI provides all semantic
  - Other
  -



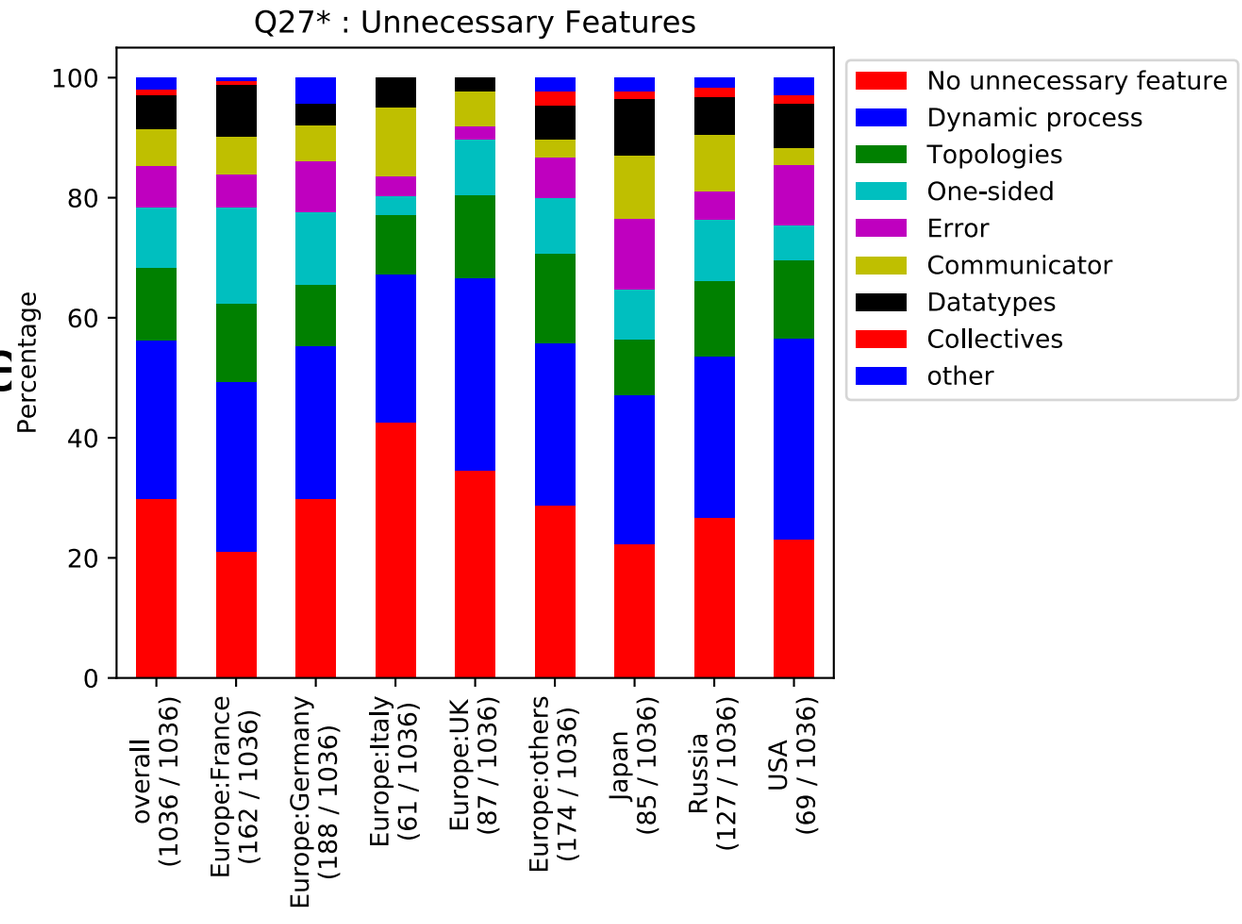
Q26: Is MPI providing all the communication semantics required by your application? If not, what is missing?

- Choices
  - Latency hiding (including asynchronous completion)
  - Endpoints (multi-thread, sessions)
  - Resilience (fault tolerance)
  - Additional optimization opportunities in terms of communication (topology awareness, locality,
  - Another API which is easier ar
  - MPI is providing all the commu application
  - Other



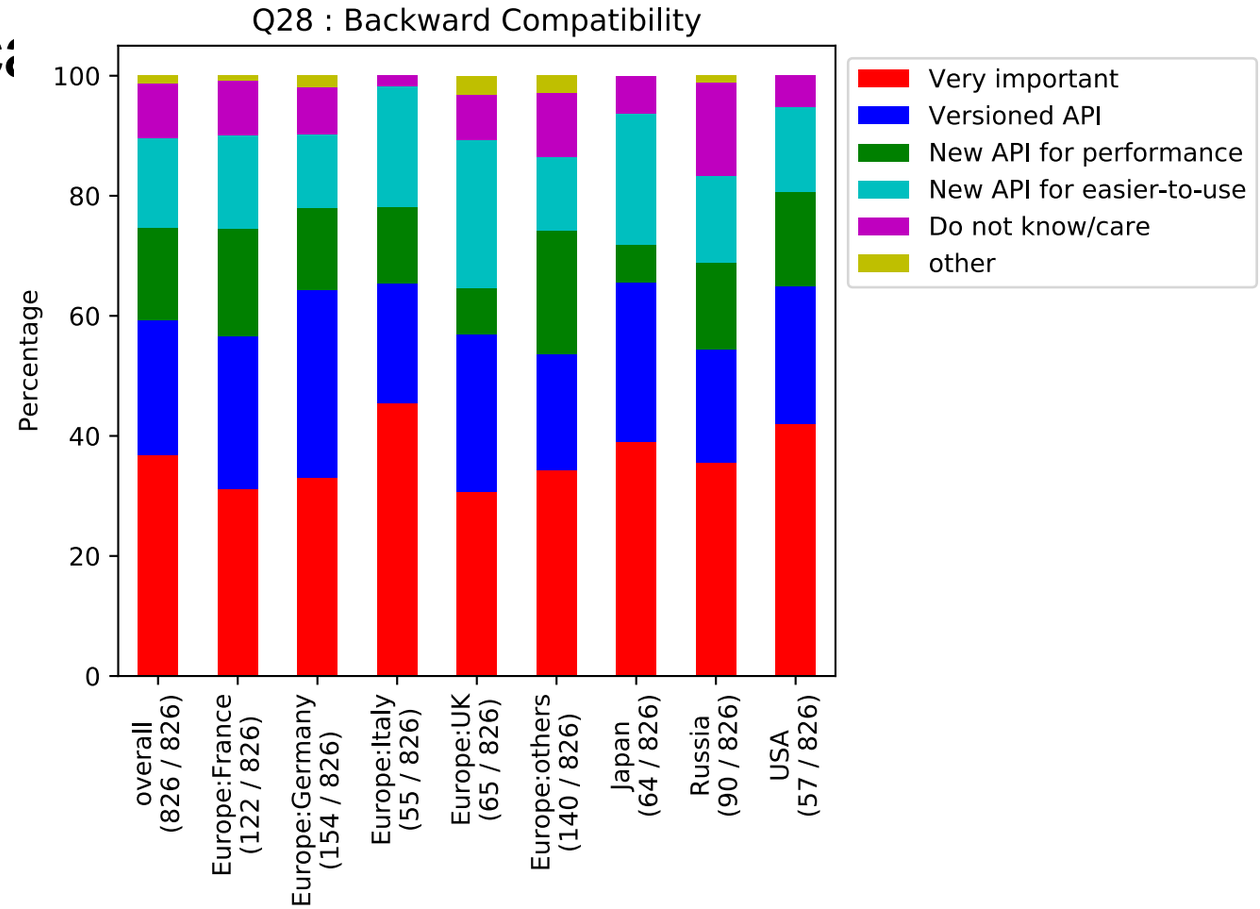
Q27: What MPI feature(s) are NOT useful for you application?

- Choices
  - One-sided communication
  - Datatypes
  - Communicator and group management
  - Collective operations
  - Process topologies
  - Dynamic process creation
  - Error handlers
  - There are no unnecessary fe
  - Other



Q28: Do you think the MPI standard should maintain backward compatibility?

- Choices
  - Yes, compatibility is very important for me
  - API should be clearly versioned
  - I prefer to have new API for better performance
  - I prefer to have new API which is simpler and/or easier-to-use
  - I do not know or I do not c
  - Other



Q29: In the tradeoff between code portability and performance, which is more or less important for you to write MPI programs?

- Choices
  - 1 - portability
  - 6 - performance