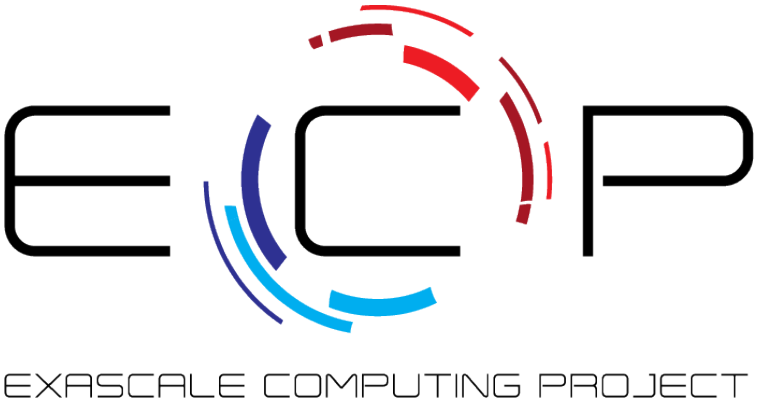
Survey of ECP Application Development Teams on Tools for Performance Analysis and Debugging

Peter McCorquodale[1], Greg Watson[2], and Shirley Moore[3]

[1]Lawrence Berkeley National Laboratory, Berkeley, CA
[2]Oak Ridge National Laboratory, Oak Ridge, TN
[3]University of Texas, El Paso, TX



1 Survey methodology

We conducted a survey of Application Development team members on the Exascale Computing Project, asking them to rate the importance of various functionalities of performance analysis tools and debugging tools, and to rate their satisfaction with such tools that they are currently using. This survey was done on SurveyMonkey over two iterations, the first in November of 2020, and the second in January of 2021. The full survey form is shown in the Appendix of this report.

1.1 Initial survey in November 2020 received inadequate response

What we sent: The first iteration of the survey was sent via a common link emailed to the `ecp-ad-pis` list by administrator Nate Noll at Oak Ridge. The members of this list are the Principal Investigators of the Application Development teams of the Exascale Computing Project. The email, shown in Figure 1, asked PIs to forward the survey link to members of their project teams. The email was sent on November 13, and the survey closed on November 30. No reminders were sent.

```
The ECP Proxy Applications team is seeking input from ECP applications
developers to determine how well current and emerging performance
analysis tools and debugging tools meet ECP application requirements,
and to identify deficiencies and gaps.

So in order for us to obtain feedback from as broad a group as
possible, could you please forward this survey link to the members of
your ECP application development project team?  The survey closes on
November 30, 2020:
https://www.surveymonkey.com/r/CTP2KSH

Peter McCorquodale and Greg Watson, for the ECP Proxy Applications team
```

Figure 1: Initial email sent to `ecp-ad-pis` on November 13, 2020.

The response: Only five responses were received. Since there are hundreds of ECP AD team developers, we figured that the low response rate indicated that few of these developers had even seen the survey, likely because not many PIs forwarded the surveys to their team members. It is impossible to know which PIs forwarded the survey, or how many did. However, the five who did respond reported being on different AD teams in Question 1 of the survey (Section 2 of this report). We do not know the identities of those who responded to the survey, but SurveyMonkey does give us their IP addresses.

1.2 Second survey in January 2021 had much better response rate

After the failure of the first survey, we decided to send the same survey to email addresses scraped from the `https://confluence.exascaleproject.org/` website. The email addresses were taken from the rosters listed for the 33 AD teams on the website, where such rosters were available. When a team's pages on the website did not list contact information for members, such as LatticeQCD, we took email addresses from ECP 2020 annual meeting posters, or in the case of Proxy Applications, from the team's own external website.

What we sent: We ended up with 613 email addresses that were fed into SurveyMonkey. SurveyMonkey sent a different link to each recipient, so responses could be tracked individually. The

survey was sent on January 11, with automatic reminder emails sent out on January 25, and closed on January 30. The email had the same words as the previous one in Figure 1, with the updated deadline date, and ending with: "`If you have already completed this survey, then thank you for your responses.`"

The response: According to SurveyMonkey, 27 (4.4%) of the 613 emails bounced, and 10 (1.6%) opted out of all emails from my SurveyMonkey account, so that left 576 email addresses that could possibly have responded. Out of these 576, there were 36 responses (6.3%).

1.3 Responses to the two surveys are combined

In this report, we are combining the 5 responses from the first survey with the 36 responses from the second, for a total of 41 responses. We consider it unlikely that anyone who completed the first survey would have also completed the second survey. This supposition is confirmed by the fact that none of the IP addresses of those who completed the second survey matched any of those who completed the first survey.

1.4 January respondents reflect the domain diversity of those surveyed

Table 1 breaks down by Internet domain the email addresses that were taken from the ECP Confluence website and used as targets for the January survey, and the number of these targets who responded to the survey. The breakdown by the three major categories of Office of Science labs, National Nuclear Security labs, and Other is very similar among those surveyed (53.5%, 24.0%, and 22.5%) and among those who responded (58.3%, 22.2%, and 19.4%, respectively). The number of respondents by each individual domain is too small to be able to draw conclusions about representativeness, but there was at least one response from every lab that had 10 or more targets.

		targets		respondents	
	domain	#	%	%	#
Office of Science Laboratories:		**328**	**53.5%**	**58.3%**	**21**
Argonne National Lab	anl.gov	87	14.2%	8.3%	3
Brookhaven National Lab	bnl.gov	18	2.9%	5.6%	2
Lawrence Berkeley National Lab	lbl.gov	78	12.7%	16.7%	6
Oak Ridge National Lab	ornl.gov	78	12.7%	11.1%	4
Pacific Northwest National Lab	pnnl.gov	35	5.7%	11.1%	4
Princeton Plasma Physics Lab	pppl.gov	22	3.6%	5.6%	2
SLAC National Accelerator Lab	slac.stanford.edu	9	1.5%	0.0%	0
Thomas Jefferson National Accelerator Facility	jlab.org	1	0.2%	0.0%	0
National Nuclear Security Administration Laboratories:		**147**	**24.0%**	**22.2%**	**8**
Lawrence Livermore National Lab	llnl.gov	55	9.0%	11.1%	4
Los Alamos National Lab	lanl.gov	48	7.8%	8.3%	3
Sandia National Laboratories	sandia.gov	44	7.2%	2.8%	1
Other:		**138**	**22.5%**	**19.4%**	**7**
National Energy Technology Lab	netl.doe.gov	6	1.0%	0.0%	0
National Renewable Energy Lab	nrel.gov	21	3.4%	2.8%	1
National Institutes of Health	nih.gov	3	0.5%	0.0%	0
National Institute of Standards and Technology	nist.gov	1	0.2%	0.0%	0
	.edu (not slac.stanford.edu)	77	12.6%	13.9%	5
	.com	27	4.4%	2.8%	1
	.net	2	0.3%	0.0%	0
	.org (not jlab.org)	1	0.2%	0.0%	0
TOTAL:		**613**			**36**

Table 1: Distribution by Internet domain of targets and respondents to the January survey.

2 Question 1: Responses specifying AD teams

The first question on the survey asked:

> 1. Which ECP Application Development project(s) are you working on? Check all that apply.

There were 32 choices, as shown in Figure 8 in the Appendix.

2.1 Table of results

Table 2 shows the distribution of responses to Question 1 on AD team affiliation, classified according to ECP's six application categories. The second-last column in the table gives the number of email addresses scraped from the team's web pages on the ECP Confluence website and used as January survey targets. Their total over all teams (or over all teams in an application category) exceeds the number of survey targets, because many people are on more than one team. Although Applications Assessment was not listed as a choice on the survey, because that project has ended, it is included in the table because our January survey targets included email addresses from its web pages on the ECP Confluence website.

2.2 Breakdown by application category

From the number of team members who were sent the January survey, and the number who responded and clicked on the box for each team, we are able to calculate a response rate for each team on the January survey, and this rate is given in the last column of Table 2. No respondent checked any of the boxes for the three National Security applications. Response rates for the other categories were relatively low for Data Analytics and Optimization (4.2% of 144 emails sent) and Earth and Space Science (4.7% of 85), and relatively high for Chemistry and Materials (8.7% of 104) and Energy (9.8% of 153). Co-Design, the largest category with 203 email addresses over eight teams, was in the middle with a 7.4% response rate.

2.3 "Other" responses

Question 1 also included a box for "Other: specify below." There were three responses given here, one each for ExaWorks and PETSc/TAO, which are ECP Software Technology teams, and one for Uintah, which is not an ECP project.

2.4 Conclusion: survey respondents reflect diversity of ECP AD teams

The results in Table 2 show that the survey respondents are broadly representative of the ECP AD teams, except for National Security applications. Of the 29 non-National-Security teams, 25 were checked at least once over the two surveys, the four exceptions being EXAALT, ExaSky, Subsurface, and Urban. In the rest of this report, we have chosen not to break down survey responses by AD team or category, because the number of responses for each AD team is far too small, and even grouping by the five represented application categories, the applications within a category often use very different methods, so any conclusions drawn from breaking down responses by category are likely to be spurious.

	Total responses	November responses	January responses	January emails sent	January response rate
Chemistry and Materials:	11	2	9	104	8.7%
LatticeQCD	1	0	1	10	10.0%
NWChemEx	3	1	2	21	9.5%
GAMESS	1	0	1	5	20.0%
EXAALT	0	0	0	9	0.0%
ExaAM	4	0	4	38	10.5%
QMCPACK	2	1	1	21	4.8%
Energy:	16	1	15	153	9.8%
ExaWind	2	0	2	33	6.1%
Combustion-Pele	2	0	2	26	7.7%
ExaSMR	2	0	2	14	14.3%
MFIX-Exa	2	0	2	11	18.2%
WDMApp	6	1	5	50	10.0%
WarpX	2	0	2	19	10.5%
Earth and Space Science:	5	1	4	85	4.7%
ExaStar	1	0	1	8	12.5%
ExaSky	0	0	0	25	0.0%
EQSIM	1	0	1	6	16.7%
Subsurface	0	0	0	13	0.0%
E3SM-MMF	3	1	2	33	6.1%
Data Analytics and Optimization:	7	1	6	144	4.2%
Urban	0	0	0	23	0.0%
ExaSGD	2	0	2	33	6.1%
CANDLE	3	0	3	49	6.1%
ExaBiome	1	0	1	14	7.1%
ExaFEL	1	1	0	25	0.0%
National Security:	0	0	0	7	0.0%
ATDM LANL	0	0	0	6	0.0%
ATDM LLNL	0	0	0	0	—
ATDM SNL	0	0	0	1	0.0%
Co-Design:	16	1	15	203	7.4%
Proxy Applications	1	0	1	19	5.3%
Application Assessment	0	0	0	11	0.0%
CODAR	4	0	4	39	10.3%
COPA	2	0	2	33	6.1%
AMREX	2	0	2	16	12.5%
CEED	2	0	2	32	6.3%
ExaGraph	1	0	1	4	25.0%
ExaLearn	4	1	3	49	6.1%
Other, named by respondent:	3	0	3		
ExaWorks (ST)	1		1		
PETSc/TAO (ST)	1		1		
Uintah (non-ECP)	1		1		
Total responses given:	58	6	52	696	7.5%
Number of respondents:	41	5	36	576	6.3%

Table 2: Table of survey responses for Question 1 on AD teams. The first column gives the number who checked the box for each team in either November of January, broken down in the second column for November and the third column for January. The fourth column shows the number of emails sent that were scraped from the ECP Confluence website for the January survey, and the last column shows the response rate of the January survey, as defined as the number of respondents who clicked the box for the AD team divided by the number of email addresses for that team.

3 Question 2: Importance of performance analysis functionalities

The second question on the survey was:

> 2. Performance analysis tools: How important is it for your application development to have tools with each of these functionalities?

There were ten functionalities listed, in the order shown in Figure 9 in the Appendix, and the options given for each were "essential", "important but not essential", "somewhat needed", and "not needed". There was also a space for respondents to write in other functionalities.

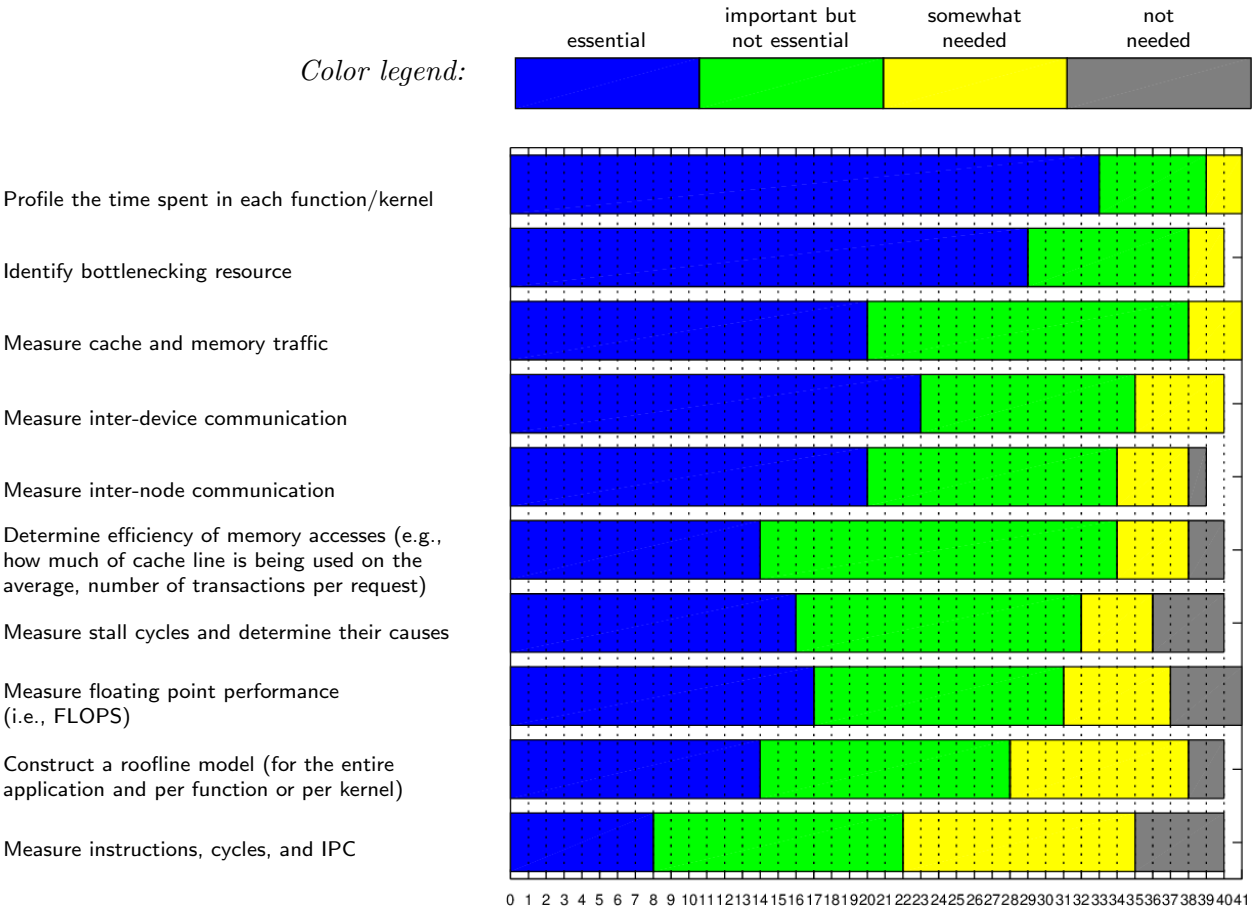


Figure 2: Distribution of responses to Question 2 on the importance of functionalities of performance analysis tools, listed from top to bottom in decreasing order of number of responses of either "essential" or "important but not essential", ties broken by number of responses of "essential".

3.1 Table of results

Figure 2 shows the responses checked on Question 2, with functionalities listed in order of number of responses of either "essential" or "important but not essential", with the highest at top. The number of respondents who checked a box for a given functionality varied from 39 to all 41 survey respondents. In the discussion below, when we report the fraction of respondents who gave a rating for any one functionality, we take it as a fraction of the total number, 41, inferring that a respondent who does not check a box for a particular functionality finds that functionality to be not needed.

3.2 What respondents find important: a majority say everything

The results in Figure 2 show us that:

- *Every* functionality on the list was rated as either essential or important by the majority of respondents (22 to 39 out of 41).
- *Every* functionality was rated as at least "somewhat needed" by at least 85% of respondents (35 of 41).
- The two functionalities that were rated as essential by the largest majorities were:
  - "Profile the time spent in each function/kernel" (80%); and
  - "Identify bottlenecking resource" (71%).

  These two functionalities were rated as at least important, but not essential by at least 93% of respondents.
- Other functionalities rated as essential by around half of respondents were:
  - "Measure inter-device communication" (56%);
  - "Measure cache and memory traffic" (49%); and
  - "Measure inter-node communication" (49%).

  All three of these were rated as at least important, but not essential by at least 83% of respondents.

3.3 Other functionalities given by respondents

At the bottom of Question 2, the survey asked:

> Please list any other tool functionalities for performance analysis that you need but are not listed above.

Four respondents gave answers here:

- "The performance analysis tools need to work well with python – and with python applications which are parallelized using MPI."
- "Besides performance, need to see scheduling of kernels as well as memory copies. e.g. See that two kernels are running simultaneously."
- "It would be nice to use an accessor API of the performance analysis tools to feed the data they collect into our logging system as the program runs. Otherwise connecting the data we collect with the data the performance analysis tools is painful post-processing."
- "Memory footprints on host and device. Memory bandwidth. % usage of host versus device."

4 Question 3: Satisfaction with performance analysis tools

The third question on the survey was:

> 3. Performance analysis tools: How do you rate the tools you currently use in terms of how they meet your requirements for each of these functionalities?

The same ten functionalities as in Question 2 were listed, and in the same order, as shown in Figure 10 in the Appendix. The options given for each were "very good", "good", "fair", "poor", "I don't have this, but would like", and "I don't need this". Then there was a space for respondents to enter answers to the question, "Which tools, if any, do you currently use to do this?"

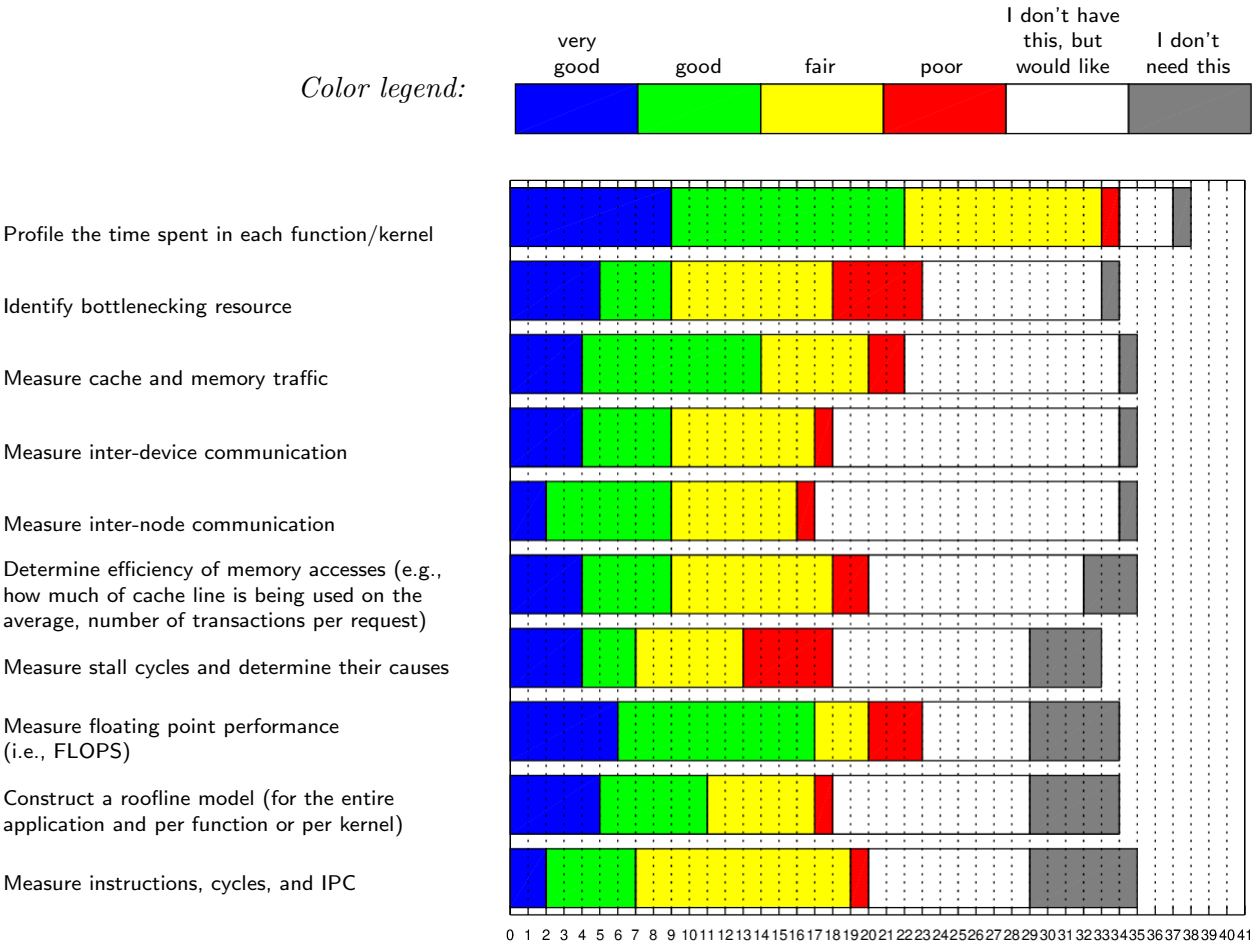


Figure 3: Distribution of responses to Question 3 on satisfaction with current performance analysis tools, listed in the same order as in Figure 2 for responses to Question 2, where functionalities have been ranked from most important down to least important.

4.1 Table of results

Figure 3 shows the multiple-choice responses received on Question 3, where functionalities are listed not in the order in which they appeared on the survey, but as in Figure 2 in Section 3, which is by decreasing order of importance as indicated by responses to Question 2. The number of respondents

who checked a box for a given functionality varied from 33 to 38, of the 41 total respondents to the survey.

4.2 Profile the time spent in each function/kernel: generally satisfied

Recall that "Profile the time spent in each function/kernel" was rated by respondents to Question 2 as the most important of the ten functionalities that were listed, with 80% classifying it as "essential". On Question 3, 38 respondents gave ratings of their current tools on this functionality, compared to 34 or 35 for the other functionalities. And the results displayed in Figure 3 show us that this is the only functionality with a majority of respondents rating their current tools as either "very good" (24%) or "good" (34%). Of the rest, 29% gave ratings of "fair", 3% as "poor", 8% opted for "I don't have this, but would like", and 3% for "I don't need this". It was on this functionality that those last two options received the fewest responses of all, indicating that respondents definitely need to do time profiling, and almost all of them have tools to do it.

4.3 Other functionalities: many report lacking tools, or at least good ones

There was much less satisfaction with performance analysis tools on the other nine functionalities on the list. As can be seen by a glance at the adjacent red and white bars in Figure 3, of those who reported a rating or that they "would like" a functionality, other than time profiling, roughly half the respondents rated their current tools as either "poor" or "I don't have this, but would like", with the combined fraction who gave those two responses ranging from 31% for "Measure floating point performance (i.e., FLOPS)" up to 55% for "Measure stall cycles and determine their causes".

The two functionalities to which respondents gave the highest fraction of "I don't have this, but would like" answers were measuring communication: 50% for "Measure inter-node communication" and 47% for "Measure inter-device communication", in both cases excluding one response each of "I don't need this".

4.4 NVIDIA Nsight tools received by far the most mentions, and rated highly

In the answer box asking respondents to name the tools they currently use for each functionality, the ones that were mentioned the most were NVIDIA's Nsight Compute and Nsight Systems, or sometimes just "Nsight". These tools received more mentions than all others put together, so it is worth including brief descriptions of them from NVIDIA's website:

> *NVIDIA Nsight Compute is an interactive kernel profiler for CUDA applications. It provides detailed performance metrics and API debugging via a user interface and command line tool. In addition, its baseline feature allows users to compare results within the tool. Nsight Compute provides a customizable and data-driven user interface and metric collection and can be extended with analysis scripts for post-processing results.* [1]

> *NVIDIA Nsight Systems is a system-wide performance analysis tool designed to visualize an applications algorithms, help you identify the largest opportunities to optimize, and tune to scale efficiently across any quantity or size of CPUs and GPUs; from large server to our smallest SoC.* [2]

[1] https://developer.nvidia.com/nsight-compute
[2] https://developer.nvidia.com/nsight-systems

In the rest of this section on Question 4, we count Nsight Systems and Nsight Compute together, because some respondents entered simply "Nsight" and did not specify either Nsight Systems or Nsight Compute.

In Figure 4, we show the reported satisfaction levels of performance tools on nine of the ten listed functionalities on the survey, excluding responses of either "I don't have this, but would like" or "I don't need this", and separating responses that mentioned Nsight from those that did not. In all cases, those who mentioned Nsight reported greater satisfaction (ratings of "very good" or "good") with performance analysis tools than those who did not mention Nsight. No one who mentioned using Nsight tools for any functionality reported a satisfaction level of "poor" on their tools for that functionality.

As shown in Figure 4, the number of respondents who mentioned Nsight performance analysis tools depended on which functionality was being asked about, and varied from 4 to 11 out of the 38 who responded to Question 3. The high levels of satisfaction reported by those who mentioned Nsight tools, relative to those who did not, may be simply a result of people who are satisfied with their tools being more likely to name those tools on the survey.

4.5 Performance analysis tools other than NVIDIA Nsight

For completeness, Table 3 lists all mentions by respondents of performance analysis tools other than Nsight tools, with the rating given for current tools on each functionality. Since many respondents named multiple tools, the ratings shown in the table should not necessarily be interpreted as indicating satisfaction level with a particular tool. There are too few mentions of non-Nsight tools to draw broader conclusions.

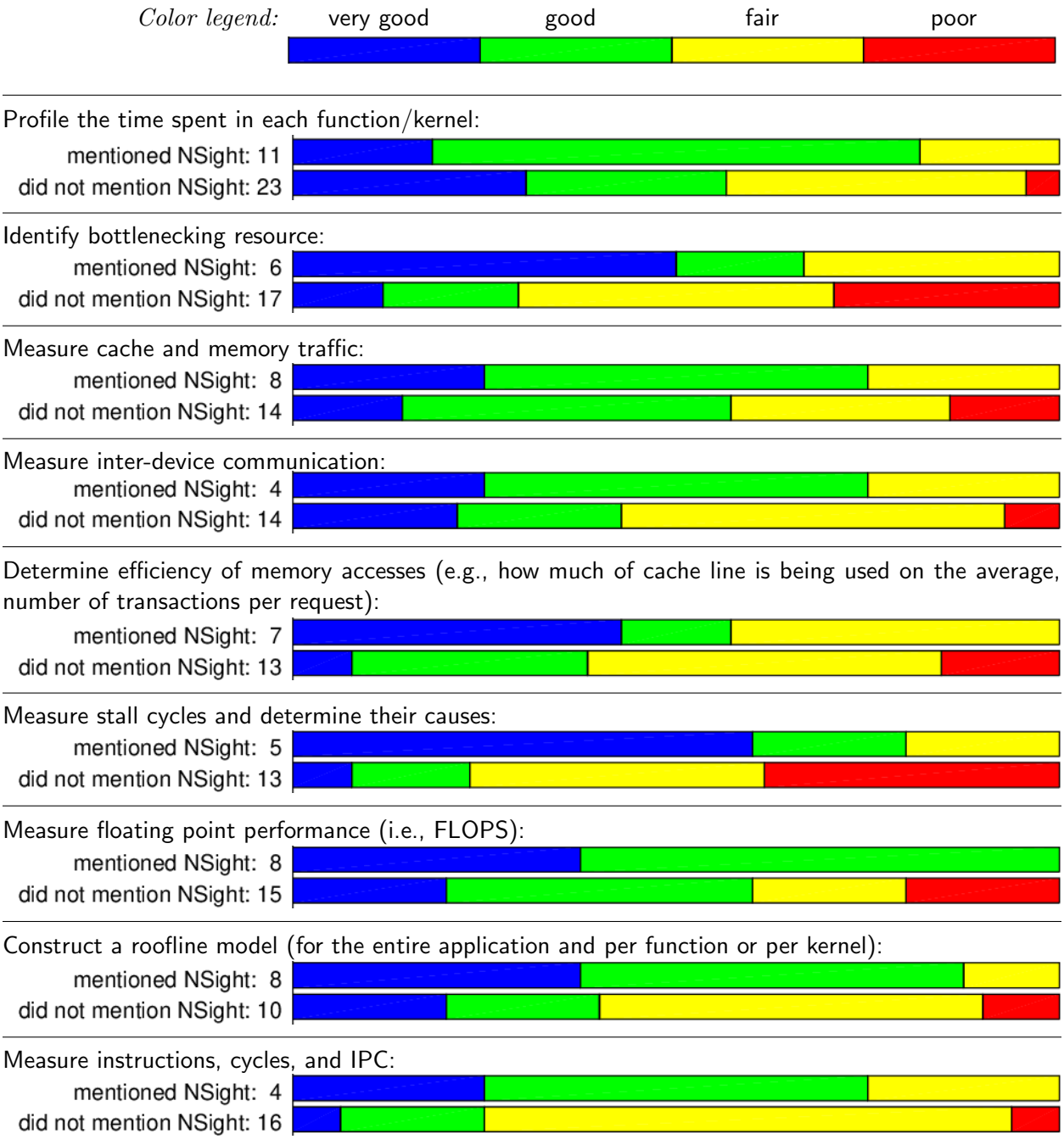


Figure 4: Comparison of satisfaction levels (for those who gave one) of those respondents who mentioned using Nsight tools, against those who did not mention Nsight tools. Functionalities are listed in the same order as in Figures 2 and 3, except that "Measure inter-node communication" is omitted, because no respondent reported using Nsight tools for that. These are not specifically ratings for Nsight tools; the ratings include everyone who mentioned using Nsight tools for the listed functionality.

	Allinea MAP	caliper	CUPTI	EP Analytics' PEBIL	gprof	HPCToolkit	Intel Advisor	IPM	kokkos profiling	mpip	nvprof	nvvp	PAPI	PSiNSTracer	Python cProfile and Pyinstrument	rocprof	TAU	timemory	valgrind	VTune	application timers	vendor tools
Profile the time spent in each function/kernel	G	F	G	G	F	G G			V		V G G	F			F	P	V G F				V V G F F	
Identify bottlenecking resource											P				V			V		V	V F P	
Measure cache and memory traffic				F							G P		F					G		G		
Measure inter-device communication															F			F			F	
Measure inter-node communication								G		G				G	G F		G F	G F			V G	
Determine efficiency of memory accesses (e.g., how much of cache line is being used on the average, number of transactions per request)											G		G							G		F
Measure stall cycles and determine their causes																					P	F
Measure floating point performance (i.e., FLOPS)				F		G	G						V			P		G			V G	
Construct a roofline model (for the entire application and per function or per kernel)							V															F
Measure instructions, cycles, and IPC						F							F						P			G

Table 3: Table of mentions by respondents to Question 3 of all performance analysis tools other than NVIDIA's Nsight tools. There is a letter entry in the table for each rating of a functionality of performance analysis tools of **V**ery good, **G**ood, **F**air, or **P**oor.

5 Question 4: Importance of debugging functionalities

The fourth question on the survey was:

> 4. Debugging tools: How important is it for your application development to have tools with each of these functionalities for HPC applications?

There were five functionalities listed, in the order shown in Figure 11 in the Appendix, and just as in Question 2 on performance analysis tools, the options given for each were "essential", "important but not essential", "somewhat needed", and "not needed". There was also a space for respondents to write in "any other tool functionalities for debugging HPC applications that you need but are not listed above, other than basic debugger functions such as setting breakpoints and displaying memory contents."

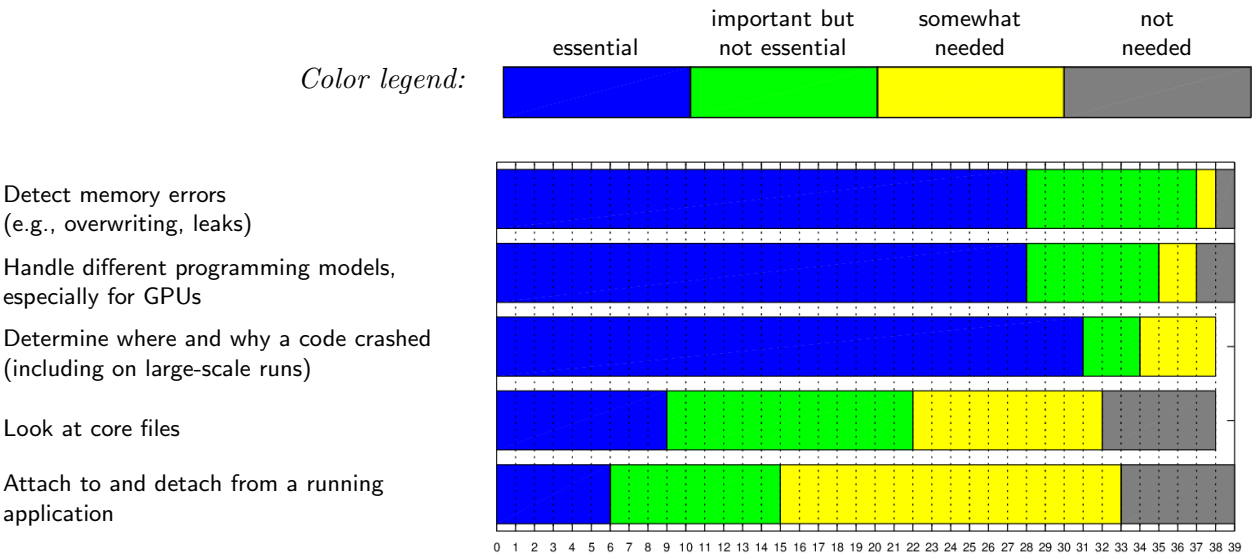


Figure 5: Distribution of responses to Question 4 on the importance of functionalities of debugging tools, listed from top to bottom in decreasing order of number of responses of either "essential" or "important but not essential".

5.1 Table of results

Figure 5 shows the responses received on Question 4, with functionalities listed in order of the number of responses of either "essential" or "important but not essential", with the highest at top. The number of respondents who checked a box for a given functionality was 38 or 39, out of 41 total respondents to the survey. In the discussion below, when we report the fraction of respondents who gave a rating for any one functionality, we take it as a fraction of the total number, 39, inferring that a respondent who does not check a box for a particular functionality finds that functionality to be not needed. We do not include the two survey respondents who skipped the whole question.

5.2 What respondents find important: most of the listed functionalities

Here are some results shown by Figure 5.

- At least 87% of respondents consider these functionalities to be either essential or important:

  - "Detect memory errors (e.g., overwriting, leaks)": 72% essential;
  - "Handle different programming models, especially for GPUs": 72% essential;
  - "Determine where and why a code crashed (including on large-scale runs)": 79% essential.
- The remaining two functionalities are still rated as at least somewhat needed by 82% or more respondents, but not as important:
  - "Look at core files": 23% essential, 56% at least important;
  - "Attach to and detach from a running application": 15% essential, 38% at least important.

5.3 Other functionalities given by respondents

At the bottom of Question 4, the survey asked:

> Please list any other tool functionalities for debugging HPC applications that you need but are not listed above, other than basic debugger functions such as setting breakpoints and displaying memory contents.

Two respondents gave answers here:

- "gdb backtraces output from gasnet and/or frozen application (generally on a single node)"
- "Monitoring where and how an application is running (i.e., which cores, threads, GPUs, etc). Synchronous GPU kernel execution for debugging."

6 Question 5: Satisfaction with debugging tools

The fifth and final question on the survey was:

> 3. Debugging tools: How do you rate the tools you currently use in terms of how they meet your requirements for each of these functionalities?

The same five functionalities as in Question 4 were listed, and in the same order, as shown in Figure 11 in the Appendix. The options given for each were "very good", "good", "fair", "poor", "I don't have this, but would like", and "I don't need this". Then there was a space for respondents to enter answers to the question, "Which tools, if any, do you currently use to do this?"

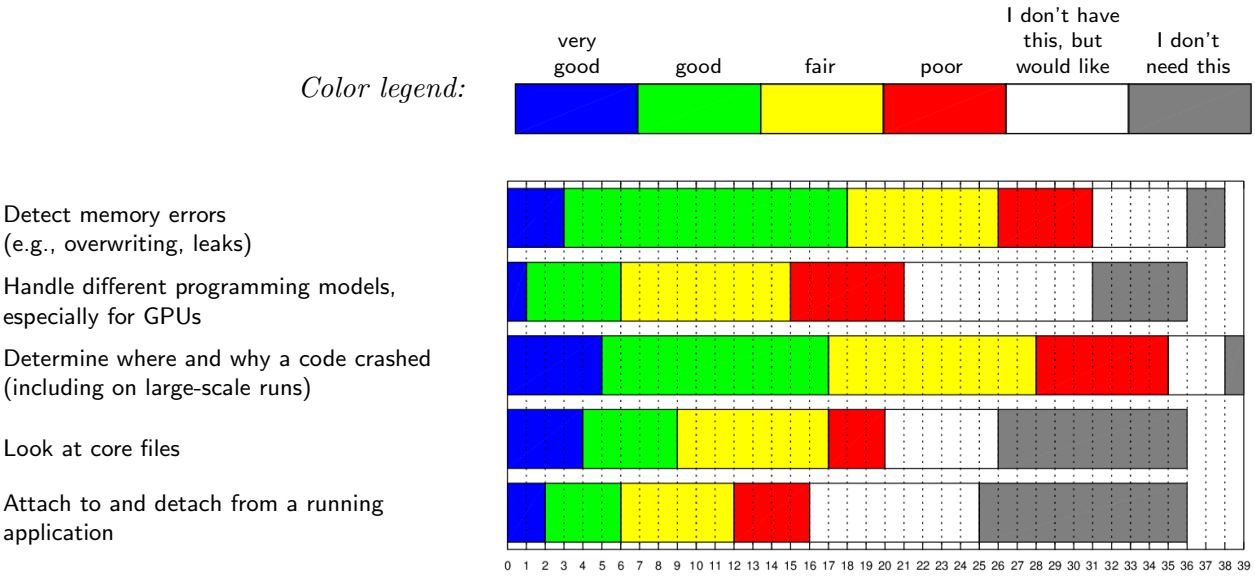


Figure 6: Distribution of responses to Question 5 on satisfaction with current debugging tools, listed in the same order as in Figure 5 for responses to Question 4, where functionalities have been ranked from most important down to least important.

6.1 Table of results

Figure 6 shows the multiple-choice responses received on Question 5, where functionalities are listed by decreasing order of importance as indicated by responses to Question 4, as in Figure 5 in Section 5. The number of respondents who checked a box for a given functionality varied from 36 to 39, of the 41 total respondents to the survey.

6.2 Valgrind and gdb named the most, but ratings not great when mentioned

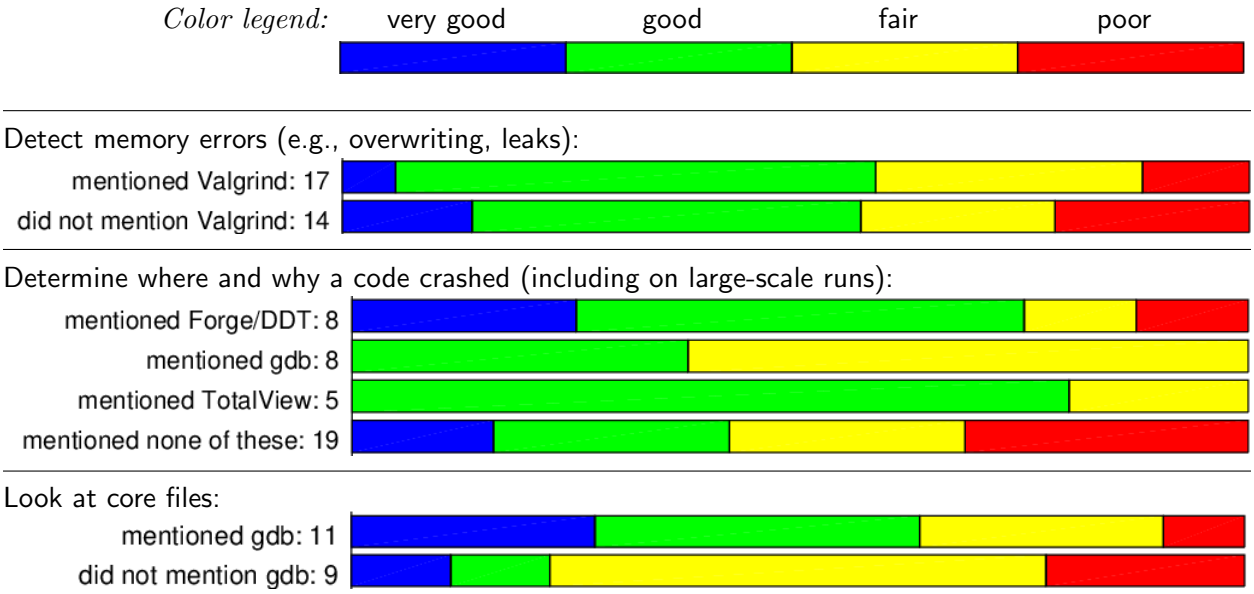


Figure 7: Comparisons of satisfaction levels, by those who gave them, of debugging tools applied to three different named functionalities on the survey, each broken down by those responses that mentioned the most frequently named debugging tools and those that did not. The ratings are not specifically for the tools mentioned; each is an overall rating of debugging tools for the functionality in question, by those who mentioned that tool. Many respondents named more than one debugging tool that they use for a given functionality.

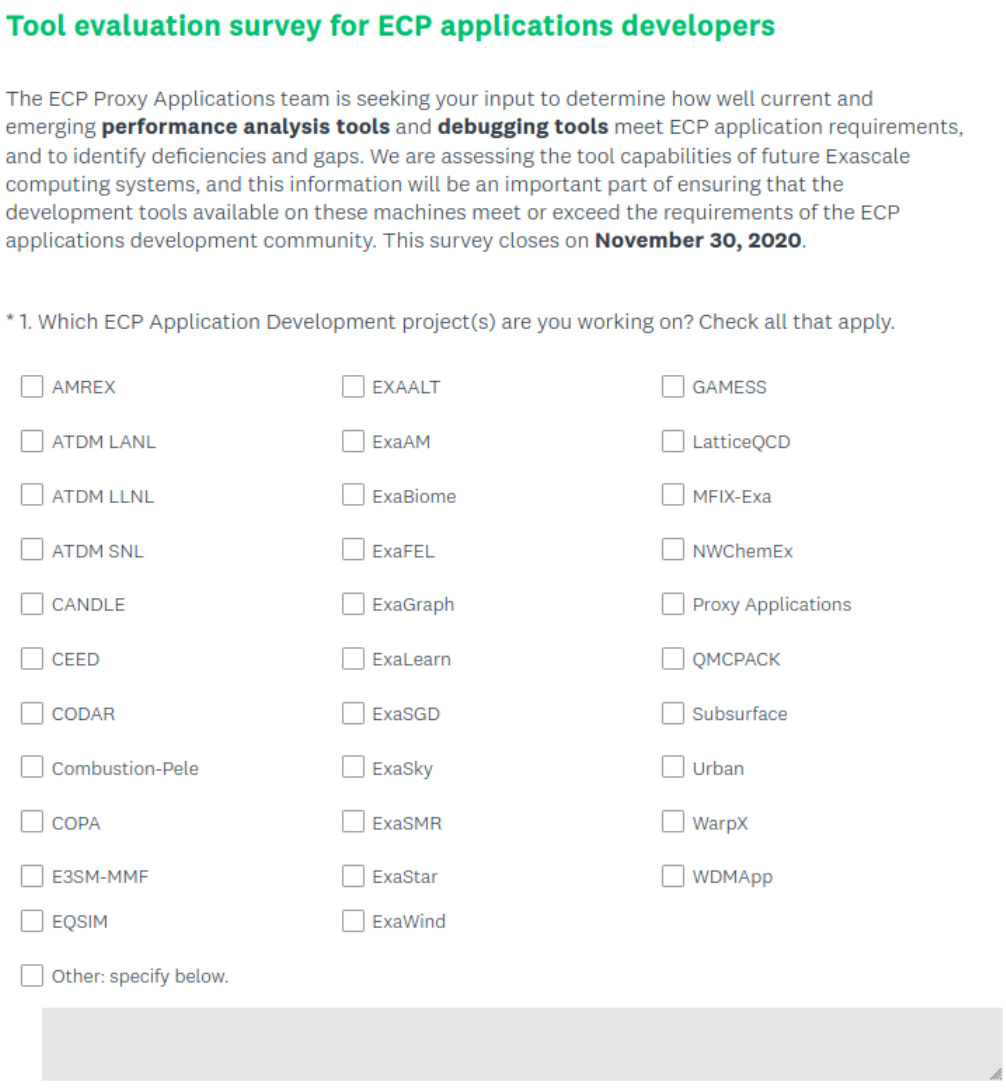
	ATP (by Cray)	asan Address Sanitizer	assertion checking	cuda-memcheck	cuda-gdb	gasnet crash reports	gdb4hpc (by HPE)	hip	kokkos	lldb (by LLVM)	NVIDIA Compute Sanitizer	Nvidia-gdb	openss	printf	Python toolchain	rocm	Valgrind	vendor tools
Detect memory errors (e.g., overwriting, leaks)		V G		G F	G			P			G		G			P		
Handle different programming models, especially for GPUs					G				G			P						F
Determine where and why a code crashed (including on large-scale runs)	V V		P			F	V V			G F F				G	F		G	
Look at core files					V					V								
Attach to and detach from a running application																		

Table 4: Table of mentions by respondents to Question 5 of all performance analysis tools other than those listed in the section whatever. There is a letter entry in the table for each rating of **V**ery good, **G**ood, **F**air, or **P**oor.

A Appendix: full survey form

The exact survey form as seen on the SurveyMonkey website is reproduced in this appendix.

- Figure 8: The preamble to the survey and Question 1 on project affiliation.
- Figure 9: Question 2 on importance of functionalities of performance analysis tools.
- Figure 10: Question 3 on satisfaction with performance analysis tools.
- Figure 11: Question 4 on importance of functionalities of debugging tools, and Question 5 on satisfaction with those tools.



Tool evaluation survey for ECP applications developers

The ECP Proxy Applications team is seeking your input to determine how well current and emerging **performance analysis tools** and **debugging tools** meet ECP application requirements, and to identify deficiencies and gaps. We are assessing the tool capabilities of future Exascale computing systems, and this information will be an important part of ensuring that the development tools available on these machines meet or exceed the requirements of the ECP applications development community. This survey closes on **November 30, 2020**.

* 1. Which ECP Application Development project(s) are you working on? Check all that apply.

☐ AMREX	☐ EXAALT	☐ GAMESS
☐ ATDM LANL	☐ ExaAM	☐ LatticeQCD
☐ ATDM LLNL	☐ ExaBiome	☐ MFIX-Exa
☐ ATDM SNL	☐ ExaFEL	☐ NWChemEx
☐ CANDLE	☐ ExaGraph	☐ Proxy Applications
☐ CEED	☐ ExaLearn	☐ QMCPACK
☐ CODAR	☐ ExaSGD	☐ Subsurface
☐ Combustion-Pele	☐ ExaSky	☐ Urban
☐ COPA	☐ ExaSMR	☐ WarpX
☐ E3SM-MMF	☐ ExaStar	☐ WDMApp
☐ EQSIM	☐ ExaWind	

☐ Other: specify below.

Figure 8: The preamble to the survey and Question 1 on project affiliation. This was for the November survey; the only difference between this form and the one sent in January was the closing date of the survey.

2. Performance analysis tools: How important is it for your application development to have tools with each of these functionalities?

	essential	important but not essential	somewhat needed	not needed
Profile the time spent in each function/kernel	○	○	○	○
Measure floating point performance (i.e., FLOPS)	○	○	○	○
Measure cache and memory traffic	○	○	○	○
Measure inter-device communication	○	○	○	○
Measure inter-node communication	○	○	○	○
Measure instructions, cycles, and IPC	○	○	○	○
Determine efficiency of memory accesses (e.g., how much of cache line is being used on the average, number of transactions per request)	○	○	○	○
Measure stall cycles and determine their causes	○	○	○	○
Identify bottlenecking resource	○	○	○	○
Construct a roofline model (for the entire application and per function or per kernel)	○	○	○	○

Please list any other tool functionalities for performance analysis that you need but are not listed above.

Figure 9: Question 2 on importance of functionalities of performance analysis tools.

3. Performance analysis tools: How do you rate the tools you currently use in terms of how they meet your requirements for each of these functionalities?

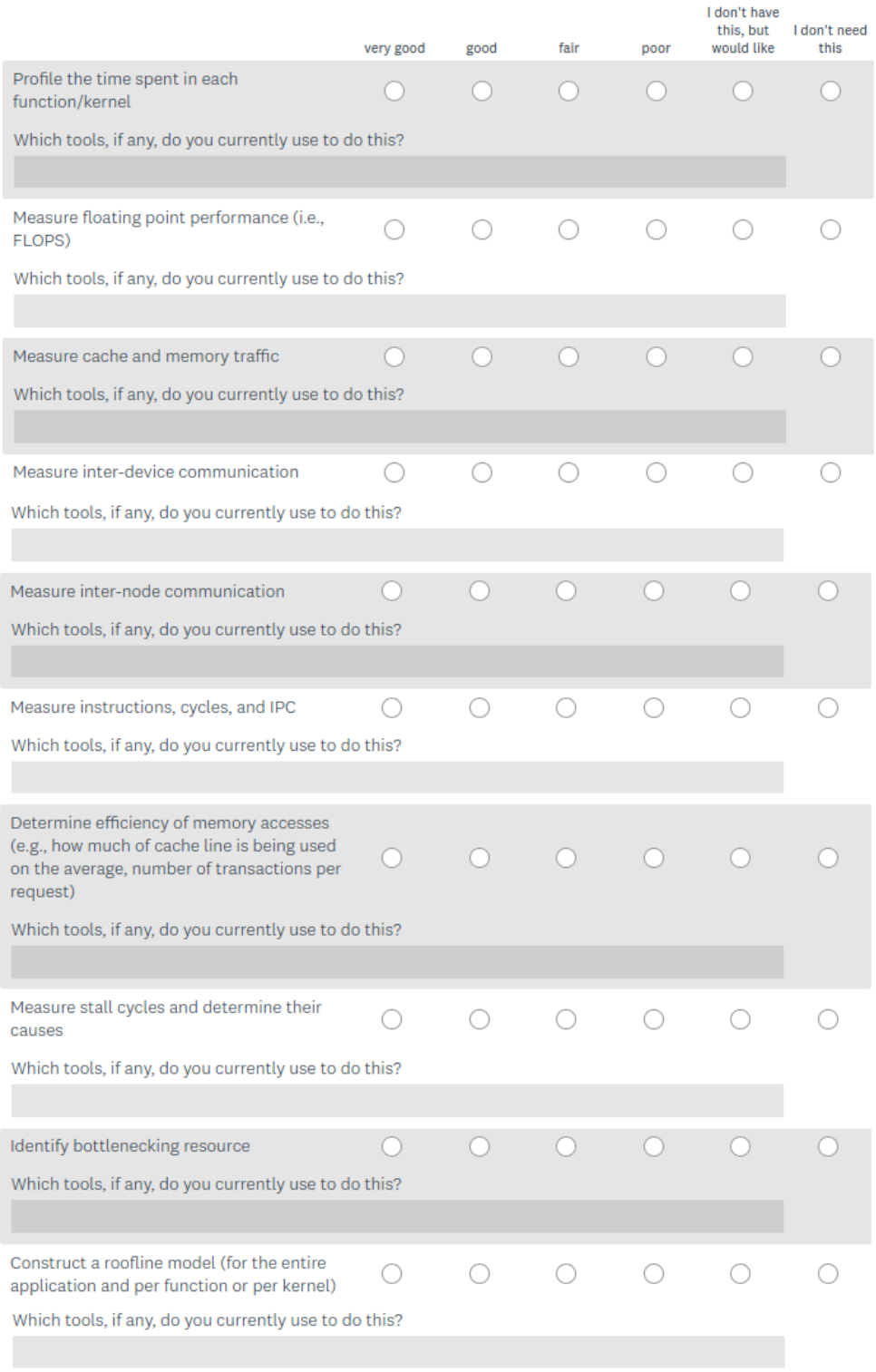


Figure 10: Question 3 on satisfaction with performance analysis tools.

4. Debugging tools: How important is it for your application development to have tools with each of these functionalities for HPC applications?

	essential	important but not essential	somewhat needed	not needed
Determine where and why a code crashed (including on large-scale runs)	○	○	○	○
Detect memory errors (e.g., overwriting, leaks)	○	○	○	○
Look at core files	○	○	○	○
Handle different programming models, especially for GPUs	○	○	○	○
Attach to and detach from a running application	○	○	○	○

Please list any other tool functionalities for debugging HPC applications that you need but are not listed above, other than basic debugger functions such as setting breakpoints and displaying memory contents.

5. Debugging tools: How do you rate the tools you currently use in terms of how they meet your requirements for each of these functionalities?

	very good	good	fair	poor	I don't have this, but would like	I don't need this
Determine where and why a code crashed (including on large-scale runs)	○	○	○	○	○	○
Which tools, if any, do you currently use to do this?						
Detect memory errors (e.g., overwriting, leaks)	○	○	○	○	○	○
Which tools, if any, do you currently use to do this?						
Look at core files	○	○	○	○	○	○
Which tools, if any, do you currently use to do this?						
Handle different programming models, especially for GPUs	○	○	○	○	○	○
Which tools, if any, do you currently use to do this?						
Attach to and detach from a running application	○	○	○	○	○	○
Which tools, if any, do you currently use to do this?						

Figure 11: Question 4 on importance of functionalities of debugging tools, and Question 5 on satisfaction with those tools.