H2020 FETHPC-1-2014

An Exascale Programming, Multi-objective Optimisation and Resilience Management Environment Based on Nested Recursive Parallelism

Project Number 671603

D5.6 – Implementation and Evaluation of Application Specific Resilience Techniques (b)

WP5: Cross layer resilience and online analysis for non functional parameters

Version: 1.0
Author(s): Charles Gillan (QUB)
Date: 29/03/18

Due date:	31st March 2018 (PM30)
Submission date:	day/month/year
Project start date:	01/10/2015
Project duration:	36 months
Deliverable lead organization	QUB
Version:	1.0
Status	Draft
Author(s):	Charles Gillan (QUB)
Reviewer(s)	Roman Iakymchuk (KTH), Stephane Monte (NUM)

Dissemination level	
PU/PP/RE/CO	*PU*

Acknowledgements

The work presented in this document has been conducted in the context of the EU Horizon 2020. AllScale is a 36-month project that started on October 1st, 2015 and is funded by the European Commission.

The partners in the project are UNIVERSITÄT INNSBRUCK (UBIK), FRIEDRICH-ALEXANDER-UNIVERSITÄT ERLANGEN NÜRNBERG (FAU), THE QUEEN'S UNIVERSITY OF BELFAST (QUB), KUNGLIGA TEKNISKA HÖGSKOLAN (KTH), NUMERICAL MECHANICS APPLICATIONS INTERNATIONAL SA (NUMEXA), IBM IRELAND LIMITED (IBM).

The content of this document is the result of extensive discussions within the AllScale Consortium as a whole.

More information

Public AllScale reports and other information pertaining to the project are available through the AllScale public Web site under http://www.allscale.eu.

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	22/03/18	First draft	Charles Gillan
0.2	26/03/18	Version after input from reviewers	Charles Gillan
1.0	29/03/18	Final version	Charles Gillan

Table of Contents

Index of Figures

Executive Summary

This deliverable describes work performed as part of task T5.4 (Development of Application Specific Resilience techniques), which in turn follows from the results of Task 5.1 (Application Specific Resilience Analysis). Deliverables D5.1 and D5.5 are precursors to this deliverable, originating from the activities in WP5. Deliverable 5.5 reported on a generic resilience protocol that has been implemented in AllScale, an approach that matches to the task based architecture that is employed in the AllScale system. This is further refined in D5.7 which reports on the implementation of the resilience manager.

At the mid-term project review in Brussels, the EC reviewers suggest that we consult more widely with the HPC community to understand how they perceive, indeed if at all, the emerging ecosystem of ExaScale computing and the challenges that it will present for resilience of their applications.

Following the methodology of D5.1 (Section 3.1), we established a questionnaire and distributed it widely to harvest the opinions of different HPC groups.

The current deliverable focuses on the implementation and evaluation of resilience planned by the groups which answered the survey. It looks beyond the checkpoint restart mechanisms that emerged as the preferred mechanism of the AllScale pilot applications. The mechanisms that emerged from the survey will be a matter for future consideration by the project, they will not be integrated into the pilot applications within the lifetime of the project.

1 Introduction

This deliverable follows from the work already reported in project deliverables D5.1 and D5.5. In D5.1, which was a horizon scanning type activity that took place in the first six months of the project, we reviewed the literature on resilience techniques as applicable to ExaScale computing and surveyed the AllScale pilot applications on their current resiliency mechanisms and on their plans for the emerging ExaScale systems. It emerged that checkpoint re-start for hard faults was the preferred solution for the three pilot applications in AllScale.

Deliverable 5.5 reported on a generic resilience protocol that has been implemented in AllScale, an approach that matches to the task based architecture that is employed in the AllScale system. This is further refined in D5.7 which reports on the implementation of the resilience manager.

As mentioned in D5.5, and reproduced for clarity here, the EC reviewers made some comments on WP5 at the mid-term review of the project. Specifically, in Section 1 of the Overall Assessment, General Comments, on page 2 of the report issued by the Project Officer after the review meeting, the expert reviewers point out that:

> ….. there are some deviations on the work plan in WP5. Some sub-tasks in T5.4, specifically the algorithmic approaches to resilience and self-stabilizing techniques, have been withdrawn but the justification given in the deliverables was confusing and impact on resources were not discussed. The withdrawal was based on the results of a questionnaire made for 3 applications which however showed a **lack of understanding of the various resilience aspects by most application developers**. However, the higher priority given to resiliency aspects related to hard failure and consequences on resource usage have been well justified during the review.

The boldface and underlining in the above are our additions to the text of the review report. The reviewers help us identify the fact that we should seek the opinions of a wider group of application developers than those in the AlScale pilot applications. Following the methodology of D5.1 (Section 3.1), we established a questionnaire and distributed it widely to harvest the opinions of different HPC groups.

The current deliverable reports on our findings from the survey. We discuss both the current and future plans for the implementation and evaluation of resilience planned by the groups which answered the survey. It looks beyond the checkpoint restart mechanisms that emerged as the preferred mechanism of the AllScale pilot applications. It is widely recognized in the literature that the possibilities for return on investment in ExaScale computing are very significant but that the challenges for scientific software development are enormous [1,2].

2 User Survey

Conducting surveys is a well-known research method in the social sciences and we borrowed from this pool of expertise in order to define a suitable set of questions to attempt to elucidate the information that we sought on resilience mechanisms.

We adopted this method, as opposed to focus group type discussions, because it allowed us to conduct quantitative research hoping to achieve conclusive results. However we recognized that a challenge we faced was whether we could engage a sufficient number of groups to answer. We required only one answer per group, typically from the lead developer. To this end, we adopted two design criteria:

1. Keep the survey as short as possible, in the hope that more people would engage with it if it did not distract from their available productive work time. After all there was no reward to the participants, other than contributing broadly to scientific research. We settled on ten questions as a relatively short number that would attract, we believed, participants due to the brevity.

2. We circulated the request to participate to any groups with which we had existing scientific collaborations in the United States, Canada and Europe. Unfortunately none of the project team had links to HPC teams in Asia or the Middle East or Australia. We recognized that this may lead to unconscious bias, so we also asked HiPEAC to notify its mailing list about the survey and we invited the groups mentioned above to distribute the survey further to their collaborators.

   We kept the survey participation to be anonymous as we felt this was the best way to engage participants and to avoid ethics and governance issue on data handling. In this sense, we cannot give a breakdown by continent on how many answered the survey. We felt it was not appropriate to ask participants to self-identify by asking questions on their location.

We conducted the survey using the cloud based provider SurveyMonkey. A service that is well-known and respected and has been operating since 1999. SurveyMonkey provides free, customizable surveys, as well as a suite of paid back-end programs that include data analysis, sample selection, bias elimination, and data representation tools. QUB paid a subscription to access the analysis toolset and as a university received an educational discount. The figures and charts that are presented in the next section have been generated using these tools.

We published the survey in early February 2018 setting an end date of Friday 16th March 2018. By having the survey one for one month we believed that we provided sufficient time to allow participants to respond at their leisure.

3 Analysis of results

In this section we present each question in the survey and the responses to that question.

3.1 Q1: In which field of computational science do you work

Table 1: Q1 Responses – field of computational science

Area of computational science	Number of responses
Numerical mathematics	1
Nuclear physics	1
Materials Science and solid state physics	1
Computational chemistry and plasma physics	7
Medical and engineering science	1
Physical oceanography	2
Condensed matter physics	1
Fluid dynamics, turbulent atmospheric flows	11
Planetary atmospheres	1
Climate modelling	1
Programming models and runtimes	1
Total	28

Analysis: There is a fairly wide spread of applications in the list of respondents and therefore in number of groups surveyed All of these are well known field in HPC, perhaps with the exception of medical engineering. Obviously the responses reflect the fact that the survey was circulated mainly via large HPC user communities, which is reflected in the dominance of computational chemistry and CfD.

3.2 Q2: In which languages are you applications written?

Multiple answers were allowed per respondent, so this captures the total space.

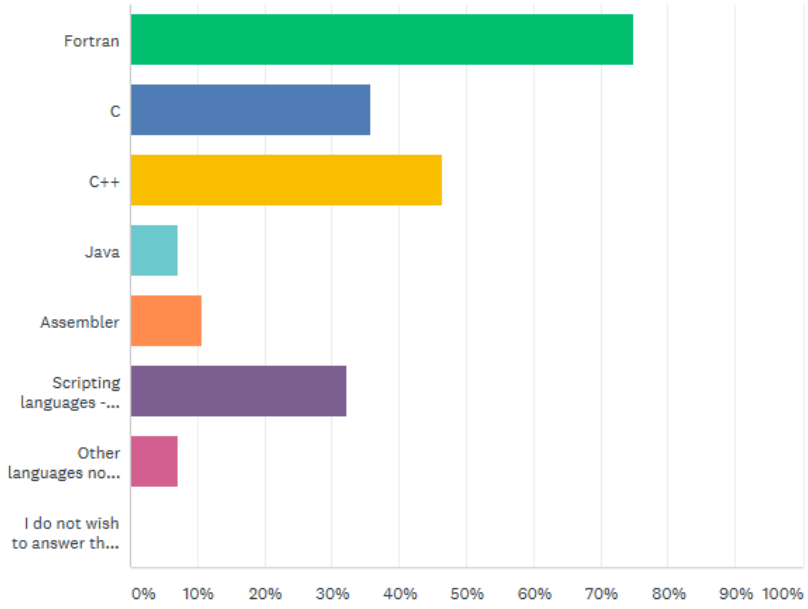

Figure 1: Distribution of languages used for program implementation

A**nalysis:** The dominance of Fortran is no surprise particularly given the high proportion of computational chemists in the list of respondents. The fact that C++ has overtaken C to be the second most prevalent language is encouraging for the AllScale project.

The presence of scripting languages reflects that fact that many HPC suites are steered by scripts, particularly Linux shell scripts. Perhaps the most interesting outcome here is that Assembler is still in use and at 10% come in fourth place.

3.3 Q3: Which of the following software technologies do you use in your application today?

As with the previous question, multiple answers were allowed.

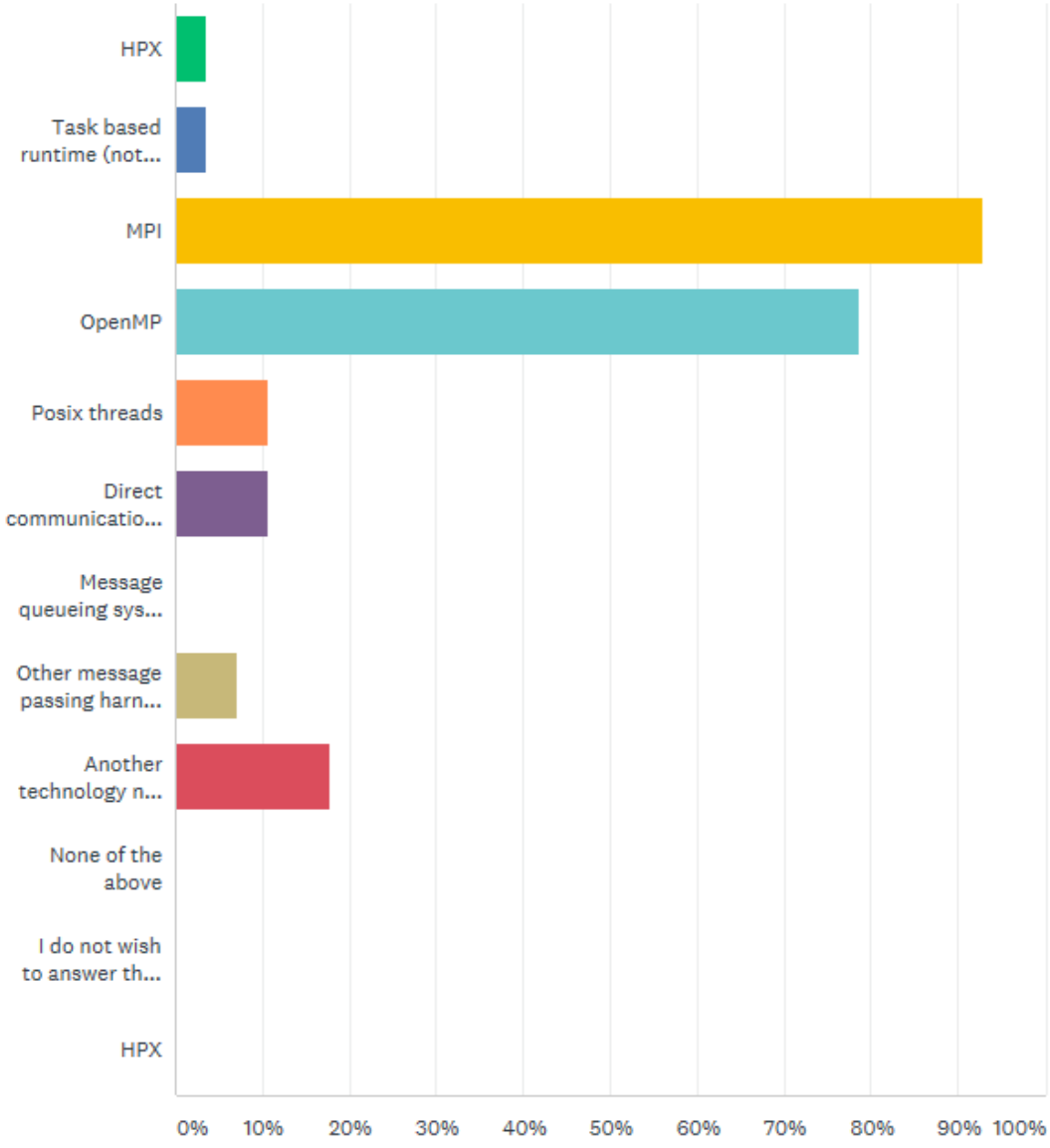

Figure 2: Analysis of the types of parallel runtime environments uses

Analysis: It is clear from the figure that MPI is the dominant implementation tool, closely followed by OpenMP. Together these far outweigh the sum of all of the other technologies. Task based runtimes are clearly in a minority.

3.4 Q4: Which are the principal mathematical kernels that dominate your applications

Multiple answers are allowed for this question.

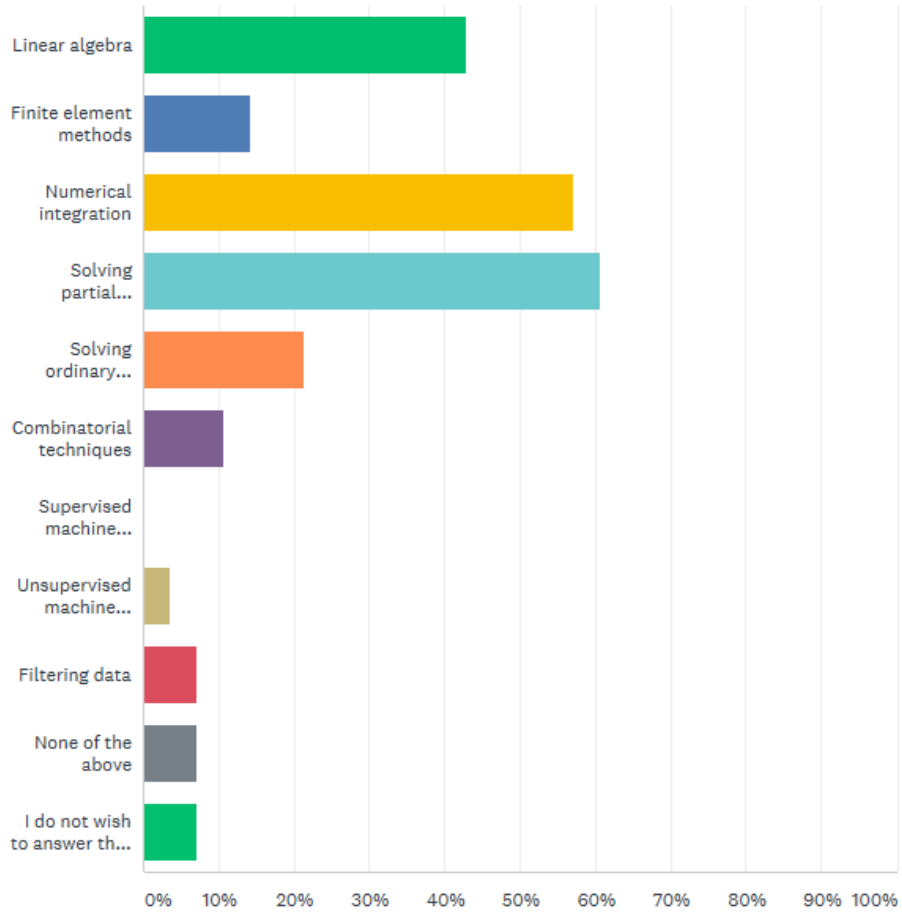

Figure 3: Analysis of the types of mathematical kernels used

Analysis: Solution of differential equations, numerical integration and linear algebra are the dominant kernels, no doubt correlating to the fact that the majority of respondents are in the fields of computational fluid dynamics or computational chemistry.

3.5 Q5: Which numerical representation is most common in your applications today

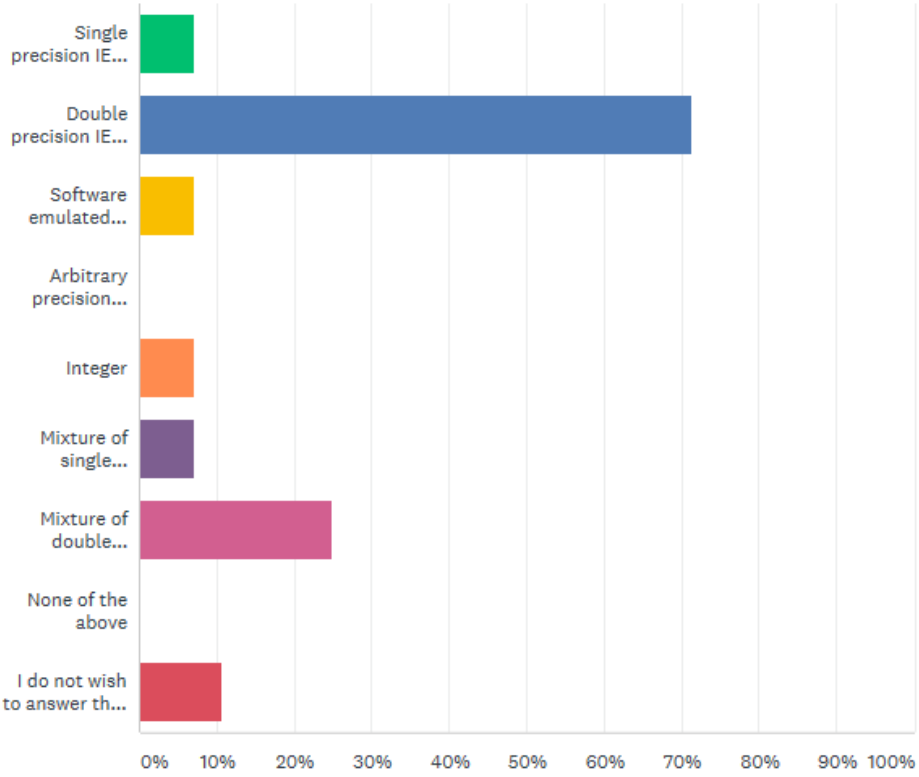

Figure 4: Analysis of the different numerical representations

Analysis: The dominance of double precision over all other representations is clear. It is noteworthy that a significant number of applications use a mixture of double and single precision.

3.6 Q6: How many CPUs do you use in typical jobs today (more than one answer allowed)

Respondents were asked to consider all of the types of jobs that they regularly run. The information is presented as a Pie chart.

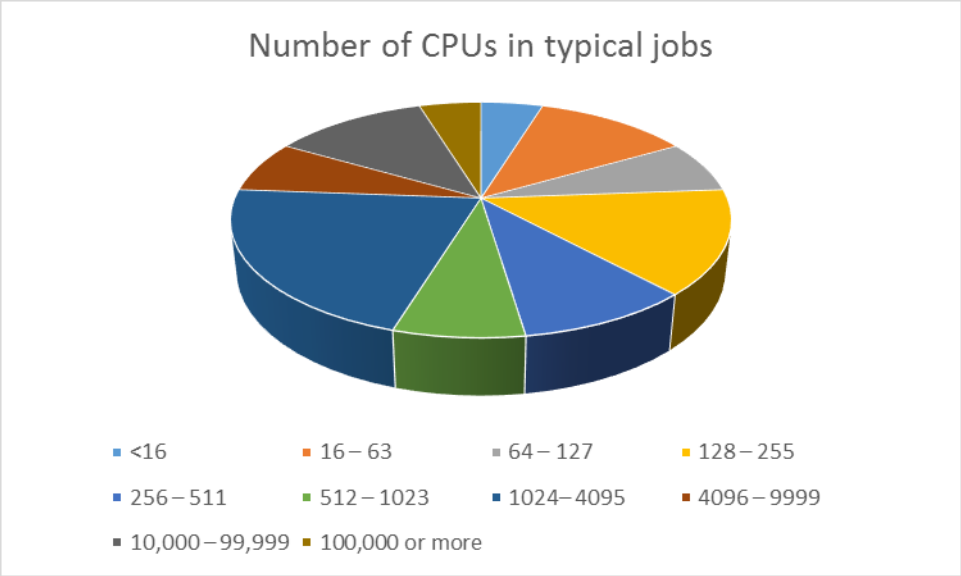

Figure 5: Distribution of jobs by number of CPUs used

Analysis: The largest block here is jobs which use 1024-4095 CPUs followed by those using 256-511 CPUs and then 128-255 CPUs. This reflects the fact that many of the surveyed users test out their code bases on smaller systems, probably local university systems, before moving to larger national or PRACE type systems. It is notable that some users report using over 100,000 CPUs per job and a similar number between 10,000 and 99,999 CPUs.

3.7 Q7: Have you considered the challenges of Exascale computing and are actively considering changes to bring the code to readiness for ExaScale

This relatively simple question produced the following result

Yes – 46.43%
No – 25%
Have consider changes but are not carrying out (too risky/costly) – 10.71%
Do not wish to answer : 17.86%

This provides a much wider picture than what we had found in surveying the AllScale pilot applications in D5.1

3.8 Q8: Have you considered the impact of HARD and SOFT faults in your application.

Table 2: Determining if users have considered hard and soft faults

My application software caters for neither HARD faults nor SOFT faults	39.29%
My application software only attempts to detect failures nodes (HARD faults)	28.57%
My application software attempts to detect both PERSISTENT and TRANSIENT SOFT faults	7.14%
My application software attempts to detect only PERSISTENT SOFT faults	0.00%
My application software attempts to detect only TRANSIENT SOFT faults	0.00%
My application attempts to verify its final computed results but not explicitly consider HARD and SOFT faults.	10.71%
I do not wish to answer this question	14.29%

Analysis: Approximately 40% of the users make no attempt to handle faults in their code. Written responses suggest that they simply resubmit the job. It is intriguing to note that 15% of users did not wish to answer the question. No reasons for this are available in the survey.

3.9 Q9: Resiliency against HARD faults can be provided by using a generic checkpoint/restart mechanism. Does your application software employ checkpoint/restart mechanisms. If so please describe briefly.

We received responses from all 28 respondents to the survey. Of these 17 indicated that a checkpoint restart mechanism was indeed part of their software package. The implementation varied significantly. For example one respondent replied that

> Job chains are realized. Checkpoints can be used via MPI_BARRIER statements.

while many others replied that periodic checkpoint files were written or in a few cases a checkpoint at every timestep. Interestingly one respondent, working in the field of computational chemistry reported (including a publication reference) that

> in the area of non-deterministic global optimization, we do better than that: our program is fully resistent against hard faults,
> cf. DOI 10.1021/acs.jctc.6b00716 [This is reference 3 below]

This is reference [3] below, reporting on an efficient massively parallel implementation of genetic algorithms for chemical and materials science problems, solely based on Java virtual machine (JVM) technologies and standard networking protocols. The paper point outs that the lack of complicated dependencies allows for a highly portable solution exploiting strongly heterogeneous components within a single computational context. At runtime, the implementation is almost completely immune to hardware failure, and additional computational resources can be added or subtracted dynamically, if needed. This raises an interesting concept of using resilient virtual machines, such as JVM, to decouple the implementation from the runtime completely.

3.10 Q10: Do you use any of the following error checking mechanisms?

This question sought to determine the different kinds of mechanisms in use by the user applications. Applications may use multiple entries from this list.

Table 3 Analysis of the different types of error checking mechanisms incorporated

Computation of checksums on critical data items	20.00%
Performing the same computation two or more times, perhaps on different nodes, and comparing results (often called redundant computation)	20.00%
Checking numerical results against the mathematical/physical constraints of the problem (perhaps as simply as just thresholding) and rolling back using a previous check point.	50.00%
Apply numerical analysis techniques, if possible, to make the algorithm resilient to soft faults	35.00%
I do not wish to answer this question	25.00%
Finite state machines	0.00%
Other techniques not listed above	5.00%

Analysis: It is interesting to note here that numerical accuracy or computation of application specific constraints together represent the main techniques used to check for errors.

3.11 Summarizing the results of the survey

In this section we bring together the results of each question in our survey and present some findings from it.

Forty percent of all applications surveyed, reported that they make no attempt to cater for hard or soft faults at present. However almost fifty-percent of the applications attempt to validate the numerically accuracy of their results. Double precision computing remains the dominant mathematical representation used. Given the distribution of languages reported, this means that double precision data representations are used in all languages.

We find that almost 60% of all groups surveyed have already investigated the changes that they will need to apply to their code base when moving it to run in an ExaScale environment, including the challenges that resiliency will impose. The fact that Fortran is a dominant language in this response group, it implies that even Fortran users are considering the challenges.

Interestingly 10% of all respondents have decided that the required changes are either too costly or too risky to implement at this time. We believe that this will prove to be a false economy and that eventually these groups will have to make changes.

We note that relatively few groups use a task based run time model, with OpenMP and MPI being not surprisingly the most widely used tools with which to exploit parallelism today. It was interesting to see the response from materials science, to Q9, reporting on resiliency against hardware faults using a JVM based approach. In that work [3] different parallelization technologies are supported: shared-memory parallelization based on the Java thread model and distributed-memory parallelization based on the Java wrapper of the MPI API (mpiJava and MPJ Express) and both implementations were shown to scale linearly.

Conclusions and Future DirectionsWe have surveyed 28 groups of application scientists from across Europe and North America to determine their understanding of resiliency requirements in the emerging field of ExaScale computing. This represents a much wider sample of the scientific software development community that the three AllScale pilot applications and addresses the issue raised by the EC reviewers at the mid-term review.

While we attempted to reach as many application groups as possible, we realize that there is a strong bias towards computational physics and chemistry in our responses. However this does correlate approximately to the distribution of workloads at many national and PRACE HPC facilities. Clearly, we are unable to compete with leading market analysts [4] in conducting our survey and were unable to engage the defense sciences community or the emerging use of data analytics in commercial computing [5].
In so far as AllScale WP5 is concerned, this deliverable completes the activity on investigating resiliency techniques. However the output from this deliverable raises interesting new avenues for research, work that could be carried on outside the AllScale project.

We believe that AllScale is well placed to market itself to the community that we have surveyed. In particular we have learnt from this survey about the kind of mathematical kernels that are widely used and AllScale can take this into account in future work when developing new training materials in an effort to market the system to a wider community. It is noteworthy that solving partial differential equations, numerical integration and linear algebra, generally in double precision floating point arithmetic, are the dominant mathematical kernels in use by this group of application scientists. On a different note, the continuing wide use of the Fortran language, most likely due to having large legacy code bases, may present a barrier to the uptake of C++ tools such as AllScale [6,7]. The implementation of the AllScale resiliency protocol is fully discussed D5.7, a document that should be read in conjunction with this one to define the AllScale approach.

4 Bibliography

1. R. F. Barrett et al., "Navigating an Evolutionary Fast Path to Exascale," 2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, 2012, pp. 355-365. doi: 10.1109/SC.Companion.2012.55
2. Ahern, Sean, Alam, Sadaf R, Fahey, Mark R, Hartman-Baker, Rebecca J, Barrett, Richard F, Kendall, Ricky A, Kothe, Douglas B, Mills, Richard T, Sankaran, Ramanan, Tharrington, Arnold N, and White III, James B. Scientific Application Requirements for Leadership Computing at the Exascale. United States: N. p., 2007. Web. doi:10.2172/1081802.
3. Johannes M. Dieterich and Bernd Hartke, Error-Safe, Portable, and Efficient Evolutionary Algorithms Implementation with High Scalability, J. Chem. Theory Comput., 2016, 12 (10), pp 5226–5233
4. Impact of Exascale Computing trends in Key sectors (TechVision), Frost and Sullivan, 2015, available on the web at: https://store.frost.com/impact-of-exascale-computing-trends-in-key-sectors-techvision.html, last accessed 20th March 2018.
5. S Rubenoff, How Deep Learning is Causing a 'Seismic Shift' in the Retail Industry, InsideHPC, Special Report, available on the web at: https://insidehpc.com/2018/02/deep-learning-shift-retail-industry/ last accessed 20th March 2018
6. S. X. Yang, D. Gannon, S. Srinivas, F. Bodin and P. Bode, "High Performance Fortran interface to the parallel C++," Proceedings of IEEE Scalable High Performance Computing Conference, Knoxville, TN, 1994, pp. 301-308. doi: 10.1109/SHPCC.1994.296658
7. M. Spencer, R. Ferreira, M. Beynon, T. Kurc, U. Catalyurek, A. Sussman, J. Saltz, "Executing Multiple Pipelined Data Analysis Operations in the Grid", Supercomputing ACM/IEEE 2002 Conference, pp. 54-54, 2002, ISSN 1063-9535.

[End Document]