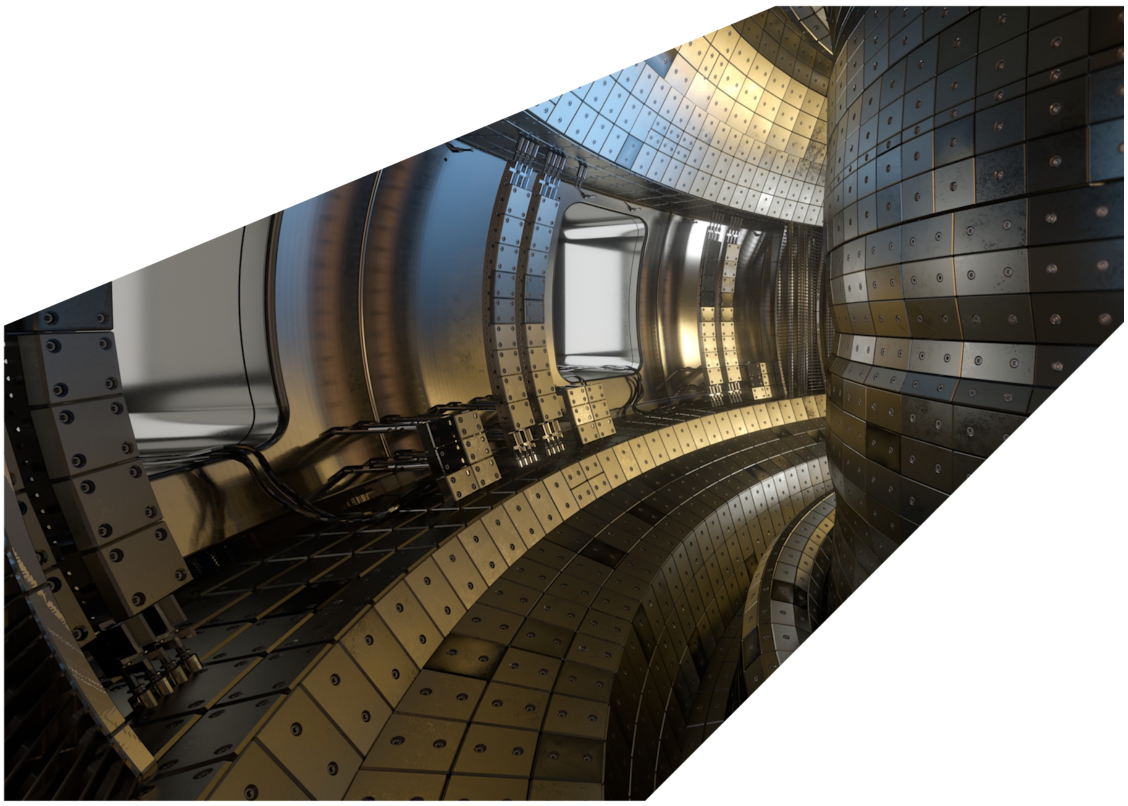
UK Atomic
Energy
Authority

ExCALIBUR

Survey of code generators and their suitability for NEPTUNE

M3.2.1

Abstract
The report describes work for ExCALIBUR project NEPTUNE at Milestone 3.2.1. Report on the status of programming models and code generators for the various node architectures focused on performance, usability, maintainability and availability.



UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0039
	Issue:	1.00
	Date:	June 28, 2021

Project Name: ExCALIBUR Fusion Modelling System

	Name and Department	Signature	Date
Prepared By:	Ed Threlfall Wayne Arter BD	N/A N/A	June 28, 2021 June 28, 2021
Reviewed By:	Rob Akers Advanced Computing Dept. Manager		June 28, 2021
Approved By:	Rob Akers Advanced Computing Dept. Manager		June 28, 2021

1 NEPTUNE Meeting, 15 June 2021 14.00-15.00 BST

Present

- Wayne Arter, UKAEA
- Steven Wright, University of York
- Ed Threlfall, UKAEA

2 Minutes

This was a one-on-one meeting between UKAEA and Steven Wright, holder of the grant T/NA086/20, to discuss progress.

For clarity, the minutes have been rearranged so as to group discussion by topic.

WA explained that this meeting was the basis for a report to the Met Office. He explained that UKAEA expected T/NA086/20 to help decide technical options for writing the code. For NEPTUNE, separation of concerns is a prime consideration. WA has read thoroughly the reports from York [1, 2] and is now looking for steer from SW on how to write code for NEPTUNE. The target hardware is clearly a consideration, however, the actual choice of platform is likely to be constrained by whatever should become available (and clearly there is no Exascale platform yet). So overall WA requested a discussion of DSL and code generation aspects, with one eye on hardware (the latter at a fairly generic level). SW said a major point was that the machine evaluations at hand do not actually need access to Exascale hardware and a lot of the testing can be done at the node level; one reason is that current and likely future parallel programming models use ubiquitously MPI+X. SW said that on-node considerations are interesting eg. Nvidia A100, V100; AMD; Arm A64FX.

2.1 Programming models

The majority of NEPTUNE code is expected to use an abstraction layer (eg. SYCL, Kokkos, Raja) plus MPI. There was a discussion on the merits of SYCL compared to its rivals. Kokkos can generate SYCL code (and also CUDA, OpenMP, AMD etc.) and so comes with some degree of future-proofing. SYCL has Intel support so it should become significant. There is a chance Sandia may abandon Kokkos and also a question of how manageable Kokkos-generated SYCL code is. To some extent, Raja/Kokkos/SYCL may be interchangeable. Regarding the HPC architecture that UKAEA currently uses, Cambridge CSD3 is Intel CPU and Nvidia GPU. WA made the point that Intel might switch their loyalties in future. Re. SYCL, SW mentioned that the Codeplay compiler almost excludes Intel though he has not investigated it extensively. ET indicated that one of the compiler's strengths is that it ships with a Nvidia back end. SW is working on a grant to research energy-efficient computing exploring Intel FPGAs (compilation time is a big issue here as this is basically an NP-hard circuit design problem). Raja/Kokkos support for FPGAs is doubtful; although of course it is possible that FPGAs will remain a minority platform.

WA mentioned being impressed by presentations concerning Fugaku - A64FX, no GPUs - where it is possible to generate efficient code quickly. He asked whether it was possible to look beyond the incipient generation of (heterogeneous) US Exascale machines, noting that some users favour homogeneous platforms. SW said homogeneous Exascale machines were currently unlikely due to power constraints, and that near-future alternatives would be KNL-like with the accelerator on the CPU. There was a discussion of energy budgeting and cost in general; SW indicated that the A64FX was of order twice as expensive as the computationally-equivalent GPU hardware (though his experience of these costings was not extensive). SW said energy usage is broadly the same across platforms - he has done work on this, though it is perhaps a little out-of-date - KNL vs double Broadwell - concluding that the performance is similar overall, though per-core performance differs. GPUs generally give more compute than CPUs for the same money but it is 'simpler' compute - WA said this suits us as much of our code is just matrix multiplications.

WA asked whether SW expected any paradigm shift from the current status quo eg. mainstream turning to ARM instead of heterogeneous platforms. ET pointed out that Nvidia now own ARM. SW answered no. WA asked whether any surprises were in store on the heterogeneous side: again, it seemed not; GPUs are expected to continue in the current mould, possibly with < 7 nm fabrication; Nvidia are still pushing CUDA, likewise AMD promote ROCm and Intel promote Xe SYCL. The US seems firmly set on heterogeneous hardware. Homogeneous machines produced say in China are a (small) possibility.

SW asked whether NEPTUNE code was to be developed from a clean sheet, or expected to be a fusion of existing apps eg. Nektar++ and EPOCH. WA explained that the original concept (from 5 years ago) was to convert finite-difference BOUT++ to use spectral/hp elements. The scope of the NEPTUNE project is however significantly larger and there is increasing awareness of the importance of community participation. Regarding physical complexity, there is almost no limit in the edge because of the likely presence of different species of neutrals, molecules, and all their ionised and excited states. Hence NEPTUNE needed to engage with the atomic physics data-collection community also.

2.2 DSL related

BOUT++ has a DSL whereas Nektar++ currently does not. For particle codes there is currently not known to be a DSL, but there are many issues eg. electromagnetic field representation (for NEPTUNE, use of Nektar++ will prescribe the latter); a big problem is the back-reaction represented by particle-field and field-particle coupling.

2.2.1 UFL

Regarding code generators, SW had spoken to Patrick Farrell who wants only something that he can program easily eg. UFL which generates C code via OPS. SW said Patrick's interest was simply having something useable (ie. he is not particularly interested in the underlying code) and cited a demo by Patrick in which the latter derived some math equations and coded them into UFL, which produced C code (SW used the word 'magic' for the process but also indicated that the generated code was not appealing to read). Re. UFL, note that it is a compiled language (in two senses: it is compiled to generate C, which is itself compiled). WA asked whether SW thought of UFL as existing on the same sort of level as Python - SW agreed, saying that UFL offers a Python-esque front end for scientist users. WA mentioned that the problem with UFL was the need to understand significant details of finite-element modelling (the weak formulation of equations) in order to use it. ET pointed out that especially turbulence modelling is resistant to automation in that the user has say to understand the implications of the choosing of $h-$ and $p-$refinements in Nektar++.

2.2.2 Users

WA made a point about the 'wide spectrum' of computing skills among physicists. Hence, UKAEA's aim is for a code fit for various user classes (eg. scientist, coder, HPC specialist) while also making the coding itself as easy as possible. There are different levels for users to want to interact with software: at a low level are the code authors, probably using SYCL; at higher levels are the theoretical and experimental plasma physicists, not using SYCL. SW made the point that the latter users would have to learn *something* eg. a DSL. WA emphasized the need to keep the amount of new material to learn as small as possible.

Illustrative is the question of DSLs to be used by eg. atomic physicists to interface with their ADAS database in order to produce fits on request. Rob Akers is concerned that database-code interface may not scale well. It would also be desirable for these scientists to contribute back to the code eg. providing validations and not just data (thus more justification for making the code accessible). SW asked whether these database queries were Python and WA answered that a variety of interfaces exist eg. Python, Fortran and IDL for ADAS. ADAS code is Fortran (probably much is Fortran 77). It seems that whenever physicists are involved, Python is unavoidable (supported by SW's personal experiences when engaged with Warwick physics). Julia could also be attractive here as regarded as good for extracting material from databases.

2.3 Miscellaneous

SW has avoided Python HPC thus far (he cited that it cannot multi-thread). WA mentioned that there is also Julia (SW has not used it yet) and also that Jupyter notebooks are becoming very common in modern Python projects. WA mentioned also cross-language claims for Jupyter that make it attractive to NEPTUNE. It was suggested that a comparison between Julia and Python would be useful (eg. Patrick Farrell uses both).

There was a discussion of compilers: WA suggested a scoping exercise on the science of compiler writing (ie. how long would it take to write a great DSL compiler). SW said Gihan Mudalige has expertise via his OPS library: OPS is a stand-alone DSL using LLVMs. WA mentioned that LLVMs are excellent for user flexibility. SW said also it is not that hard to translate C++ to IR (intermediate representation) but that optimizing for specific architectures is hard and LLVMs remove some of this difficulty. Intel etc. spend much time optimizing their back ends (eg. introducing patented technology) so compiling to IR is a good strategy, rather than trying to beat Intel etc. compiler writers. WA agreed LLVMs will form our foundations eg. SYCL on LLVMs; SW agreed that this approach is logical (SYCL can compile to IR eg. DPC++ uses LLVMs and SPIR to target Intel hardware, with alternative back ends for non-Intel).

WA mentioned also that the concept of data-driven programming kept surfacing and asked how this might fit into NEPTUNE work. SW admitted he had not worked on this but said he felt there were significant challenges involved. He agreed that it was likely that there was already significant intelligence gone into the use of eg. buffering versus in-situ processing at compiler level. WA also asked about asynchronous programming - SW mentioned AMT (asynchronous multi-threading); he knew of a Sandia group working on DARMA/vt, which is an AMT library. This is useful for PIC workloads, which can become very unbalanced eg. if particles bunch up. SW opined that there was nothing particularly outstanding in DARMA/vt and said the benefits of AMT depend on balancing issues vs. the cost of moving data; also, the proxyapps do not have major balancing issues. SW also mentioned that SYCL can do AMT on a node level as it has a job queue (inter-node is MPI).

2.3.1 Wrap-up

WA noted that another DSL workshop was planned, this time with perhaps a more structured program leading to a concrete plan for how to proceed. SW agreed that another workshop would be of benefit and proposed presentations on UFL, SYCL and other abstraction layers (N.B. Gihan Mudalige has already delivered a presentation on OPS). WA opined that discussion should be focussed and that there should not be a large selection of presentations; perhaps proposing an approach and offering it up for debate would be good, though he noted that there should always be opportunities to try alternative technologies (the suggestion was to propose SYCL, with Kokkos as a reserve). SW noted that an alternative approach was to base as much of the code as possible on pre-existing libraries (many of which have SYCL implementations) and so avoid doing any SYCL programming unless something novel is needed (he cited the example of the EMPIRE-PIC code from Sandia, where the field solve is via Trilinos). SW agreed that it could well be the case that making a timely decision was more important than which option was chosen.

Acknowledgement

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

References

[1] S. Wright, B. Dudson, P. Hill, D. Dickinson, and G. Mudalige. Approaches to Performance Portable Applications for Fusion. Technical Report 2047358-TN-01-02, UKAEA Project Neptune, 2021.

[2] S. Wright, B. Dudson, P. Hill, D. Dickinson, and G. Mudalige. Identification of Testbed Platforms and Applications. Technical Report 2047358-TN-02, UKAEA Project Neptune, 2021.