Parallel Software usage on UK National HPC Facilities 2009-2015: How well have applications kept up with increasingly parallel hardware?

Dr Andrew Turner
EPCC
University of Edinburgh
Edinburgh, UK
a.turner@epcc.ed.ac.uk

Abstract—**By common consent, one of the largest challenges facing the computational science community on moving from terascale, through petascale towards exascale HPC facilities is the ability of parallel software to meet the scaling demands placed on it by modern HPC architectures. Understanding how well current applications scale and how their scaling and use pattern has changed (or not changed) through the recent rise of multicore processor architectures provides insight into how well current HPC users may be able to exploit future systems. In addition, understanding both the requirements and limitations of users and their applications is critical to research agencies and other organisations provisioning shared facilities for their user communities.**

In this paper we analyse the usage of parallel software across two UK national HPC facilities: HECToR (initially a Cray XT and latterly Cray XE system) and ARCHER (a Cray XC30 system). These systems have spanned the rise of multicore architectures: the initial HECToR installation was based on single-socket, dual core AMD Opteron technology and the current ARCHER installation is based on dual-socket, 12-core Intel Ivy Bridge technology. Both are general-purpose systems that support a wide range of research communities and more than 1500 users. In particular, we analyse and comment on:

- **Trends in application usage over time: which applications have declined in use and which have become more important to particular research communities; and why might this be?**
- **Trends in the sizes of jobs: which applications have been able to increase their scaling properties in line with architecture changes and which have not? Can we identify why this is the case?**
- **Changes in research areas on the systems: which areas have appeared/increased and which have declined?**

The in-house Python tool that is used to collect and analyse the application usage statistics from the Cray ALPS scheduler is described. This tool does not depend on the monitored software being installed in a central location; rather it uses regular expressions to identify the executables and so includes data on software installed locally by users. Using this method we are typically able to automatically associate more than 70% of the core hours used on the systems with known applications. Increasing the coverage is solely dependent on contacting users for details of the applications that they use. Each known application also has metadata associated with it that describes properties such as programming language, parallel programming models used, research area, license type, etc. and so we are also able to analyse usage based on these properties. Amongst other analyses, this provides insight into changes in prevalence of parallel programming models in different research areas.

The analysis reveals shows that there are two broad classes of scaling limitations on applications: the first is obviously due to limitations in the applications themselves and can potentially be overcome with a suitable investment of development effort; the second is due to limitations in the scale of the scientific problem being studied that have an intrinsic limitation in the parallelism available to the applications.

We conclude with a look forward to future HPC facilities and comment on how this may impact particular research areas and applications based on the preceding analysis.

Keywords: HPC, Applicatons

I. INTRODUCTION

Supercomputer processor technology has undergone a large amount of change over the past decade – from single-core architectures, through dual- and quad-core architectures to current multicore architectures with 10's of cores per processor. The UK national supercomputing services HECToR [1] and ARCHER [2] have spanned the change from low count multicore processors to the current day where we have two 12-core processors per compute node. As both HECToR and ARCHER have been Cray systems supporting similar research communities this provides an opportunity to analyse how much effect this change has had on the applications used by researchers.

If we can understand how applications have (or have not) been able to cope with changes in requirements due to changes in processor technology we will have a better understanding of how future architecture changes (for

example, the availability of many-core processors) will affect different applications and research communities.

The UK national supercomputing services are provided by a number of partners:

For ARCHER, the partners are EPSRC [3] and NERC [4], the service is run by EPCC [5] and STFC Daresbury Laboratory [6]; Computational Science and Engineering support is provided by EPCC; and the hardware is provided by Cray Inc. [7].

For HECToR, the partners were the UK research councils: EPSRC, NERC and BBSRC [8], the service was run by EPCC at the University of Edinburgh and STFC Daresbury Laboratory; Computational Science and Engineering support was provided by NAG Ltd. [9]; and the hardware was provided by Cray Inc.

II. System Architecture Details

Table 1 summarises the different architectures for the systems considered in this paper. The major impact on applications has been the increase in core counts (particularly in moving from HECToR Phase 2a to HECToR Phase 2b) and the corresponding increase in concurrency on node. This has also impacted applications as the amount of memory per core (and memory bandwidth per core) decreased across the HECToR Phases; although this trend has been reversed somewhat on ARCHER due to the increased memory per node and increased memory bandwidth available with the Intel processor technology. The change in interconnect technology has had little impact on most applications although the increase in stability that accompanied the switch from SeaStar+ (XT) to Gemini (XE) has benefitted all users.

III. Overall Comparisons

Figure 1 shows the percentage of core hours used on the systems broken down by research area.

The largest growth areas as time has progressed are materials science and biomolecular simulation; the latter has grown from very modest use on HECToR Phase 2a to over 10% of all core hours used on ARCHER. This increase is at least partially due to the significant development of applications (e.g. Gromacs) that have been able to exploit MPP systems such as HECToR and ARCHER for this research area. Computational Fluid Dynamics (CFD) have increased their usage going from HECToR to ARCHER and, as we will see below, this is mostly due to a large increase in the number of cores that the CFD codes are able to exploit. Climate/ocean modelling has stayed largely constant across systems. In the "Others" category the area that has grown the most from HECToR to ARCHER is Medical Physics (grown from 0.01% to 0.5%). Although this area is not currently using large amounts of time we expect to continue to see growth as the codes used to simulate, for example, blood flow using Lattice Boltzmann method, have the potential to scale to very large numbers of cores.

Astrophysics, Cosmology and Particle Physics are not represented, as they do not form part of the partners that fund and run the UK national supercomputing services; they have their own HPC facilities in the UK including an IMB BG/Q for QCD and various smaller HPC clusters for astronomical research.

Table 2 shows the top 10 codes used on each of the systems (ordered by total core hours used).

The highly used codes across the four systems generally fall into a number of broad classes:

- **Periodic Electronic Structure:** VASP, CP2K, CASTEP, CRYSTAL, CASINO. These codes are generally used for materials science or chemistry research and usually use 3D parallel FFTs and dense/sparse linear algebra.
- **Classical atomic structure modelling:** Gromacs, NAMD, DL_POLY, LAMMPS. These *N*-body simulation codes are generally used for biopolymer simulation (Gromacs, NAMD) or materials science (DL_POLY, LAMMPS) and often also require parallel 3D FFT for electrostatic interactions.
- **Climate Simulation and Ocean Modelling:** UM (MET Office Unified Model) WRF, NEMO, MITgcm, Oasis. These codes generally employ a structured grid to simulate the ocean or atmosphere and also include Earth system models that couple atmosphere and ocean grid-based models.
- **Computational Fluid Dynamics**: (CFD): Hydra, INCOMPACT3D, PDNS3D, HiPSTAR. These codes employ structured and/or unstructured grids.

Table 1: Architecture details for the systems considered in this paper.

System (Type)	Processor Arch. (Clock Speed)	Cores per Node (Sockets)	Memory/Node (Bandwidth/Core)	Nodes (Cores)	Interconnect	R_{peak} (Tflop/s)
HECToR Phase 2a (Cray XT4)	AMD Barcelona (2.3 GHz)	4 (1)	8 GB (3.2 GB/s)	5664 (22656)	SeaStar+	63.4
HECToR Phase 2b (Cray XE6)	AMD Magny-Cours (2.1 GHz)	24 (2)	32 GB (3.6 GB/s)	1856 (44544)	Gemini	372.8
HECToR Phase 3 (Cray XE6)	AMD Interlagos (2.3 GHz)	32 (2)	32 GB (2.7 GB/s)	2816 (90112)	Gemini	829.0
ARCHER (Cray XC30)	Intel Ivy Bridge (2.6 GHz)	24 (2)	64 GB (4.9 GB/s)	4920 (118080)	Aries	2550.5

Figure 1: Breakdown of usage by research area.

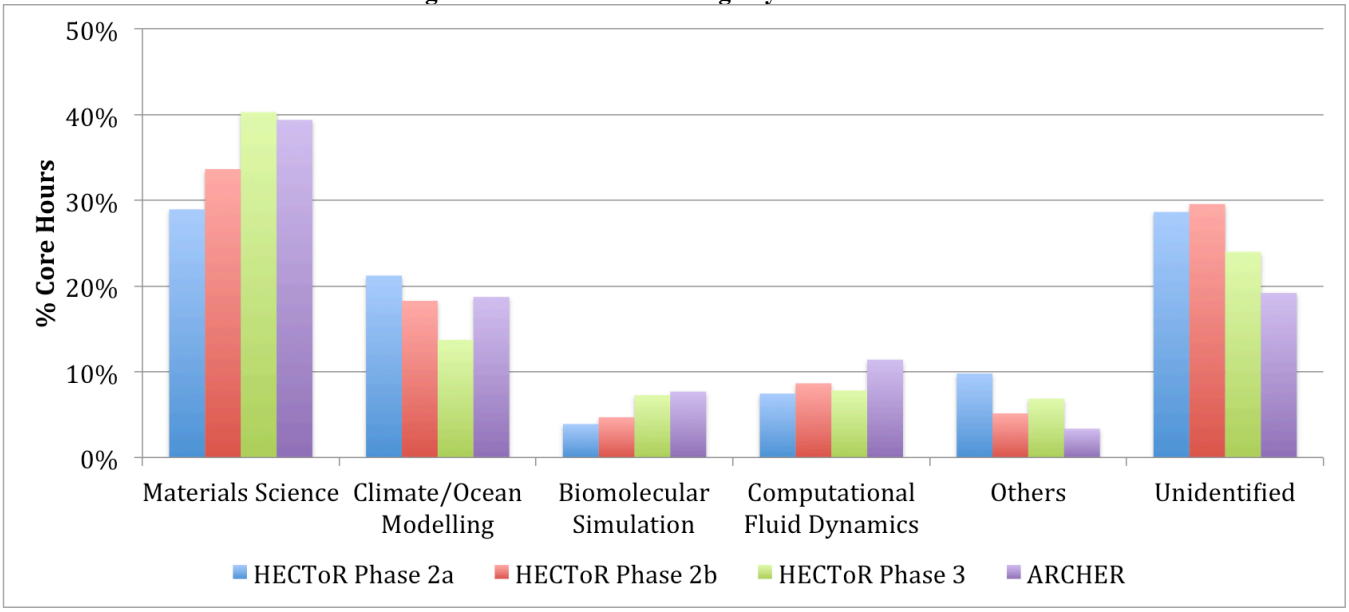


Table 2: Top ten codes by core hours used for each of the systems.

HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
UM	VASP	VASP	VASP
VASP	UM	CP2K	CP2K
CASTEP	CASTEP	UM	UM
Hydra	CP2K	CASTEP	Oasis
CP2K	INCOMPACT3D	Gromacs	Gromacs
Chroma	NEMO	DL_POLY	CASTEP
NAMD	Gromacs	PDNS3D	HiPSTAR
ChemShell	MITgcm	MITgcm	NEMO
WRF	ChemShell	NEMO	LAMMPS
DL_POLY	PDNS3D	CRYSTAL	CASINO

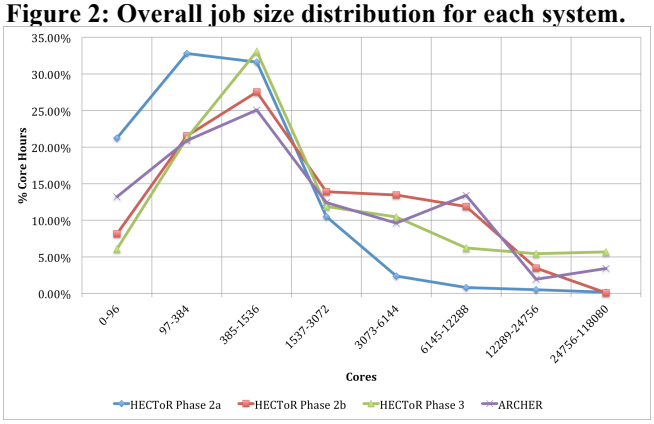
Codes in the top ten lists above that do not fall into these categories are: ChemShell (QM/MM computational chemistry) and Chroma (lattice QCD). These areas do not show up across all systems in a large way and so are not further analysed in this paper.

Figure 2 shows the job size distribution for each of the systems (by core hours used) and reveals that as time has gone on, the distribution of core hours has used has generally shifted to larger job sizes although the change has not been dramatic. It is also clear that not many users are really managing to use the very largest jobs possible on the systems for "production" jobs (the codes that do run at these very large scales are discussed in more detail in the sections below). In general, this capability is being used for testing scaling properties and for benchmarking.

This graph reveals that the biggest single change in job size profile was moving from HECToR Phase 2a to HECToR Phase 2b and that, since this step, the change has been modest. This can be understood by looking at the processor architecture, this corresponds to the change from 4 cores per node (single socket Cray XT nodes) to 24 cores per node (dual socket Cray XE nodes). This change is a 6× increase in concurrency per compute node leading to an enforced requirement on users (and hence applications) to be able to scale to higher core counts. The distributions above taken with the top code table suggest that, on the whole, applications were able to cope with this requirement for increased parallelism. Applications, CP2K for example, that were able to exploit this increased parallelism have been able to grow their share of usage.

Table 3 shows a breakdown of time usage on the systems by programming languages employed by parallel applications. The distribution is relatively stable across system with 60-70% of usage attributed to Fortran applications and 5-15% attributed to C/C++. The only significant change has been the increase in C usage from less than 1% on HECToR Phase 2a to 6% on ARCHER. This growth is due to the increase in the use of the Gromacs code (which is written in C).

Figure 2: Overall job size distribution for each system.



Finally, Table 4 shows the breakdown of usage by parallel programming models across the systems. MPI, of course, dominates; accounting for at least 70% of the use on all systems and it is a reasonable assumption to make that the remaining unidentified usage has a similar breakdown and this would push the MPI usage number above 95%. The only model used other than MPI are codes that interface with the Cray interconnect at a low level using the DMAPP API. We also monitor the usage by codes that can use a hybrid message passing plus shared memory approach and their usage has roughly doubled from 16% on HECToR Phase 2a to 28% on ARCHER. This increase has been a key feature in most codes that are able to scale to very high core counts (O(10,000)).

We now look at particular application types that have high usage across all systems in more detail.

IV. Application Types

A. *Periodic Electronic Structure*

Periodic electronic structure codes generally account for more than 30% of the time used on all the systems covered here and so their scaling properties are key in determining the overall distribution of jobs on the system. All of the codes in this class are parallelised using message passing via MPI (CP2K also supports a hybrid MPI + OpenMP model and CASTEP can also use a hybrid MPI + System V Shared Memory Segments model. Figure 3 shows how different periodic electronic structure codes have changed their share of the usage over the systems and Table 5 the median job sizes for each of the codes on the systems. CRYSTAL has a much larger median job size than all other applications in this area possibly reflecting the different approach it has (using purely localised basis functions).

Table 3: Breakdown of usage by programming language.

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
Fortran	63.3%	65.2%	66.8%	69.3%
C++	8.9%	2.7%	4.4%	7.4%
C	0.4%	3.6%	5.4%	6.3%
Unidentified	29.1%	30.0%	24.2%	19.4%

Table 4: Breakdown in usage by parallel model.

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
MPI	56.0%	46.2%	48.0%	54.3%
MPI +OpenMP	10.7%	15.8%	21.7%	22.8%
MPI +SharedMem	5.6%	9.0%	5.5%	5.0%
DMAPP	0.2%	0.5%	1.4%	0.6%
Unidentfied	29.1%	30.0%	24.2%	19.4%

Figure 3: Periodic electronic structure code usage across systems as a function of % core hours used.

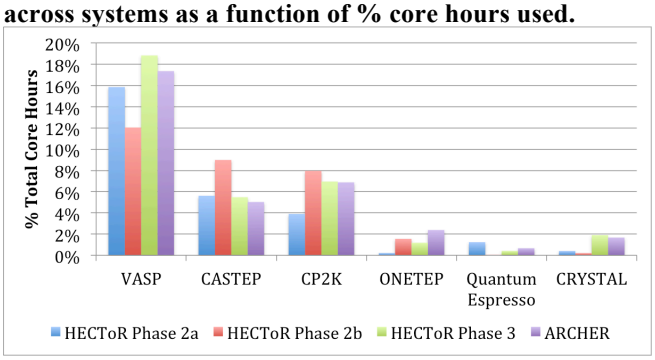


Table 5: Median job sizes (in cores) for periodic electronic structure codes on each of the systems.

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
VASP	240	456	480	240
CASTEP	252	720	512	360
CP2K	224	1320	608	672
ONETEP	104	504	416	864
Quantum Espresso	60	72	448	192
CRYSTAL	144	4032	3648	2808

VASP is by far the dominant code on all systems. CASTEP usage has stayed pretty constant, apart from a brief increase on HECToR Phase 2b, while CP2K, ONETEP and CRYSTAL have all increased their share from the quad-core to the multi-core systems.

The change in used time as a function of job size for VASP is shown in Figure 4. There was a distinct shift to larger job sizes in going from quad-core to multi-core nodes as users were forced to adapt to the new architecture. The balance of the parallel decomposition in VASP can be altered at run time and this flexibility allows the code to adapt somewhat to changing architectures.

If we compare CASTEP and CP2K job sizes (CASTEP: Figure 5, CP2K: Figure 6) we can see that the ARCHER CASTEP use tails off by 3072 cores while CP2K jobs have significant usage up to the 6145-12288 core range. The trend across systems for the two codes also shows that both codes increased their job sizes on the move from quad-core to multi-core architecture. It is, of course, difficult to say if the difference in job sizes between CASTEP and CP2K are due to inherent scaling limits in the codes or due to the scaling limits in the scientific problems that the codes are used to treat. In addition, as CASTEP has been around longer than CP2K there may be some user "inertia" that means that the users continue to run the same job sizes that they have always run as they have an acceptable time to solution for their research.

The remaining periodic electronic structure codes have relatively low usage so we have not explored their data in detain in this paper but the distribution data is available online [10].

Figure 4: VASP job size distribution for each system.

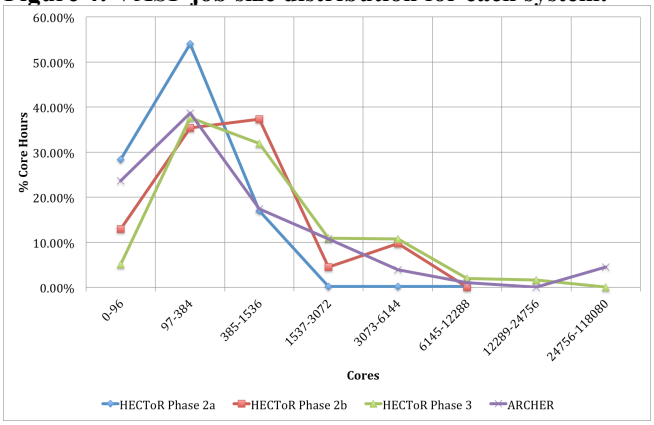


Figure 5: CASTEP job size distribution for each system.

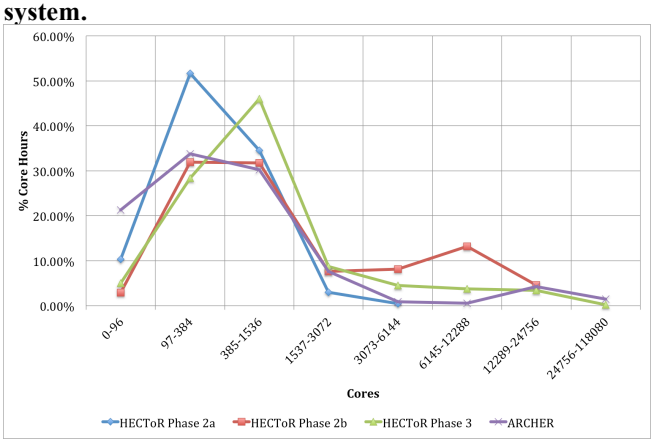
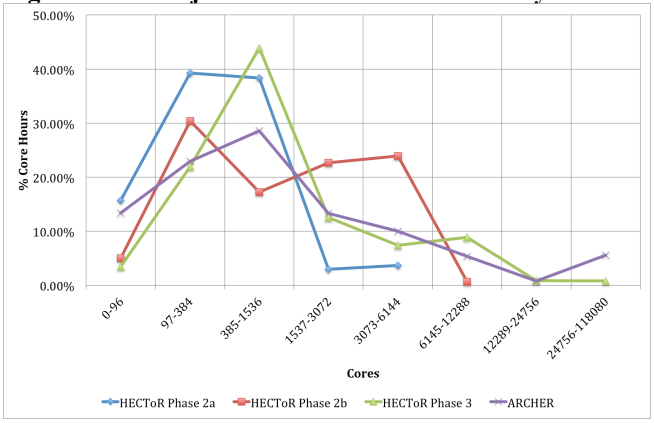


Figure 6: CP2K job size distribution for each system.



B. Classical Atomic Structure Modelling

The usage profile of the *N*-body classical atomic structure modelling codes is shown in Figure 7 and the median job sizes in Table 6. There are really two sub-classes of code here: Gromacs, NAMD and Amber of generally used for biomolecular simulation (proteins, DNA/RNA, lipids, etc.); and DL_POLY, LAMMPS are used for materials science applications. For the biomolecular codes, the main story is the growth in the use of Gromacs from the quad-core to multi-core systems; NAMD, in contrast, has decreased its share on the systems. Moving from HECToR to ARCHER it is noticeable that the use of DL_POLY has dropped while the use of LAMMPS has grown. All of the codes discussed here employ hybrid MPI+OpenMP and this area is the least Fortran-centric area on UK supercomputers with two of the codes (NAMD, LAMMPS) being C++ based and one (Gromacs) using C.

For these codes, the scalability is strongly influenced by the scientific problem being treated. This can be illustrated by looking at the job distribution for DL_POLY (Figure 8).

Although the intrinsic scalability of the code has not changed on going from HECToR to ARCHER, the job size mix is fundamentally different. In particular, a large amount of time was spent on very large jobs on HECToR Phase 2b/3. This is a direct consequence of a particular research problem being investigated using very large DL_POLY simulations. Since then there has not been that requirement for such large systems to be studied using DL_POLY and hence there have not been such large jobs on the system. This illustrates that the variation in problem size for these *N*-body codes in the materials science area is very large.

In contrast, the biomolecular area have much more constant, constrained simulation sizes (as proteins and other biopolymers have a finite size) and this is reflected in the job size distribution for Gromacs (Figure 9) where, once it was the dominant code in the space, the jobs sizes do not vary much.

Figure 7: *N*-body code usage on each system.

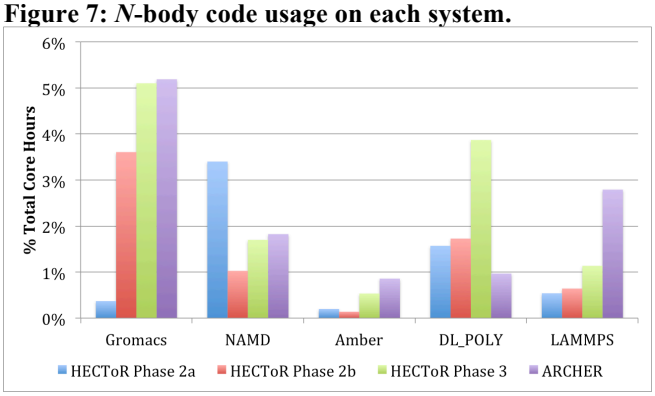


Table 6: Median job sizes (in cores) for *N*-body codes on each of the systems.

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
Gromacs	56	1152	640	432
NAMD	32	72	352	480
Amber	16	72	96	24
DL_POLY	128	4104	32000	72
LAMMPS	96	384	480	456

Figure 8: DL_POLY job distribution for each system.

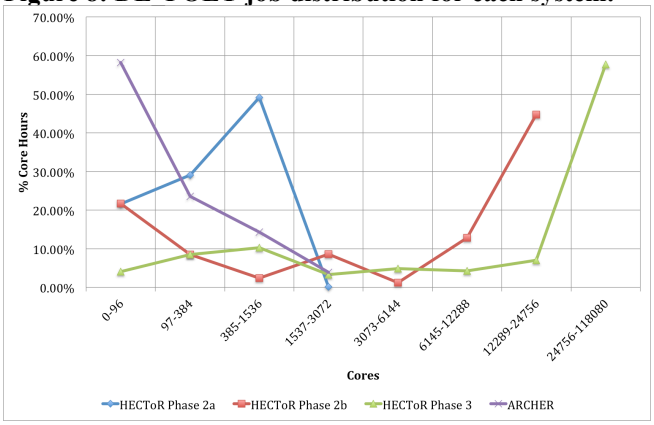


Figure 9: Gromacs job size distribution for each system.

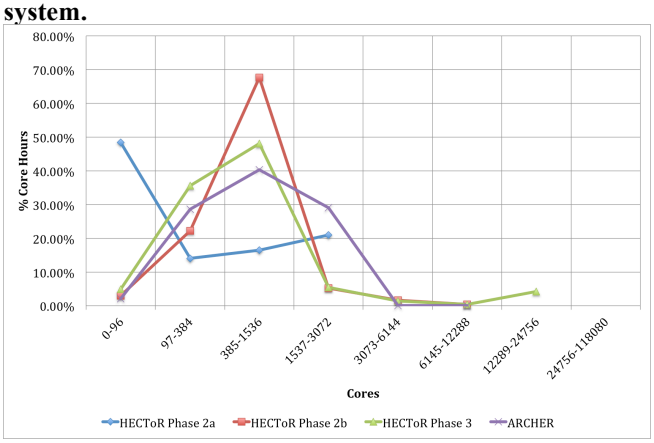


Figure 10: Climate/ocean code usage on each system.

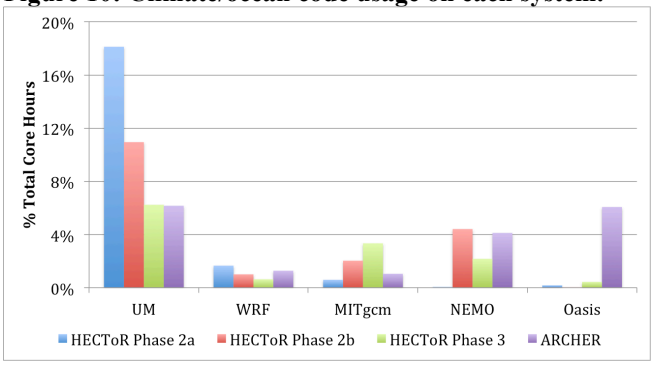


Table 7: Median job sizes (in cores) for climate/ocean modelling codes on each of the systems

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
UM	256	864	224	1392
WRF	400	400	704	2064
MITgcm	16	504	96	384
NEMO	64	1536	1376	1920
Oasis	32	144	384	5232

There has been a small growth in job sizes on ARCHER as people start to simulate larger systems (for example, proteins embedded in membranes) but the job size is essentially static even though the weak scaling properties of the code mean that for large systems it can scale to huge numbers of cores. The peak job size here reflects the natural size of the scientific problem and not the code scaling properties. The move to larger HPC systems does open up the opportunity to use the increased potential throughput to implement more sophisticated statistical analyses.

C. Climate Simulation and Ocean Modelling

In this area, where structured grids are the norm, there is often a drive to higher and higher resolution (finer grids) and so the potential for useful weak scaling should be strong. We would expect any limits on the scaling of these codes to be due to code features and/or design rather than limits inherent in the scientific problem being studied. The majority of the codes in this area employ pure MPI as their parallel method of choice although there are exceptions such as the ECMWF IFS code that employs Coarray Fortran.

The reduction in Climate/Ocean Modelling usage across the HECToR phases seen in Figure 1 above can be traced mainly to the gradual reduction in usage of the MET Office Unified Model (UM) as shown in Figure 10 below. The increase in the % of ARCHER used for Climate/Ocean Modelling compared to HECToR Phase 3 is mainly due to the emergence of the Oasis coupler that is used to couple the UM atmospheric model to the NEMO ocean model. The job size distribution for the UM for the various systems (Figure 11) demonstrates how the drive to higher resolution in these models allows the jobs to scale up as the size of the resource increases. The job size increases tend to lag a bit behind the increase in hardware for a number of reasons including: code development to ensure that the solver, load balancing and memory use work properly as the codes scale up; a requirement to verify the models on the new hardware; and the increase in stochastic sampling that also accompanies the increase in resolution. The same trends are also visible in the data for the other codes: WRF, MITgcm and NEMO (see the usage reports online [10]).

The distribution for the Oasis coupler (Figure 12) looks very different. In this case, the use on HECToR was very low and seems to have been purely for code porting and, latterly, benchmarking. On ARCHER the code has been used at scale for large amounts of time at a single job size suggesting that it is now being used for "production" research. The parallel nature of the coupler allows it to exploit larger numbers of cores as it combines the parallelism of the UM and NEMO leading to one of the few high scaling production codes on the ARCHER service.

Figure 11: UM job size distribution for each system.

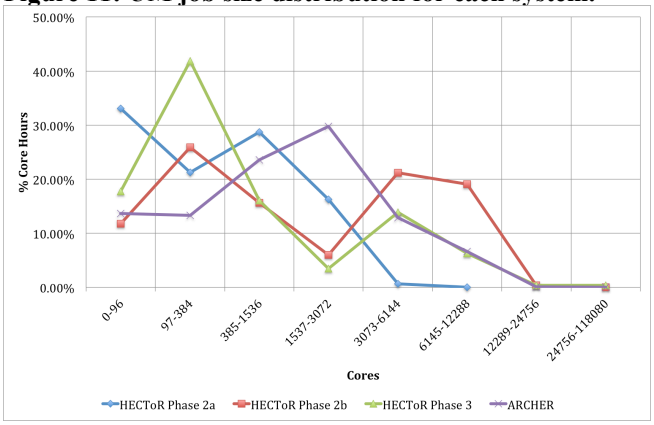
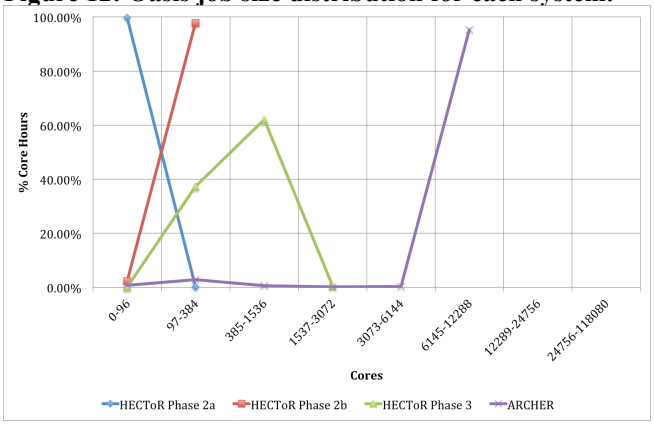


Figure 12: Oasis job size distribution for each system.



D. Computational Fluid Dynamics (CFD)

As with the climate and ocean modelling codes, CFD codes of various sorts tend to employ structured or unstructured grids. As well as the tendency to increase the resolution of the grids there is often a drive to simulate larger systems (i.e. larger grids). This potential weak scaling provides opportunities for these codes to scale to high core counts. There is also a push to greater complexity with more complex geometries that include dynamic elements (for example rotating turbine blades) and these increase the difficulties of maintaining acceptable load balancing and avoiding global communications as the codes scale to larger and larger numbers of cores.

This research area employs a wider variety of codes than any other area on the systems, the major codes and their use on the various systems are listed in Figure 13 and the median job sizes for the codes listed in Table 8.

A different code has had the largest usage on each of the three systems: Hydra on HECToR Phase 2a, Incompact3D on HECToR Phase2b, PDNS3D on HECToR Phase 3 and HiPSTAR on ARCHER. The large increase in "Others" on ARCHER is due to increased usage of Code_Saturne and Nektar++.

Figure 13: CFD code usage on each system.

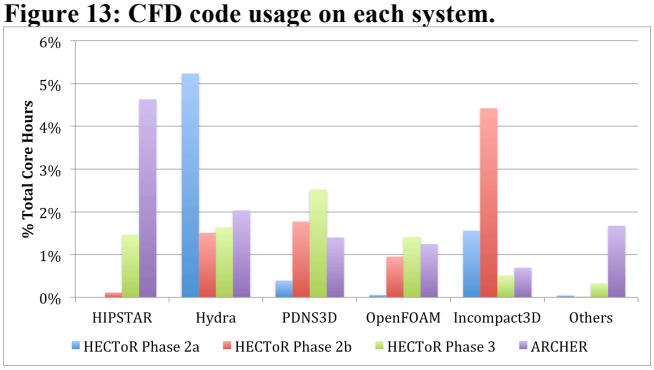


Table 8: Median job sizes (in cores) for CFD codes on each of the systems

	HECToR Phase 2a	HECToR Phase 2b	HECToR Phase 3	ARCHER
HiPSTAR		768	7776	10344
Hydra	256	1056	800	1248
PDNS3D	508	7200	12544	6144
OpenFOAM	512	768	992	288
Incompact3D	2048	6912	3616	2064

Most of the codes in this area are Fortran based and use MPI parallelism, exceptions include: OpenFOAM and Nektar++, which are both C++ codes and HiPSTAR, which is Fortran-based but uses a hybrid MPI+OpenMP approach.

As can be seen in Figure 14, the job size distribution for the Hydra code has not changed across different systems. One reason for this is that as there are so many different codes in the CFD space a single code tends to be used by a particular research group for a particular problem. The uniformity of the Hydra job sizes suggests that it is being used to study the same research problems across all the systems.

As we progress from Phase 3 to ARCHER, HiPSTAR becomes the dominant code in this area, its job size distribution is shown in Figure 15. The distribution and usage levels reflect that code development took place on HECToR Phase 2b/3 before the code went into full production on ARCHER. This code is able to exploit larger numbers of cores for the scientific research of interest than almost any other on ARCHER. This reflects the fact that for codes in this area the factor limiting the scaling is often not the scientific problems themselves but rather the development of the code itself. There has been a large amount of software development work on this application to improve the scaling using hybrid MPI +OpenMP parallelism.

Figure 14: Hydra job size distribution for each system.

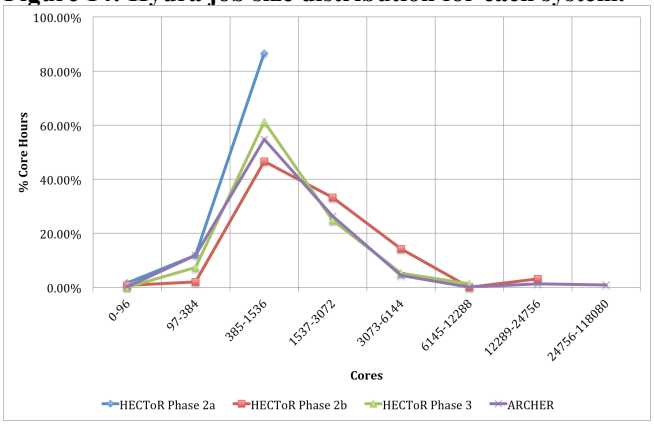
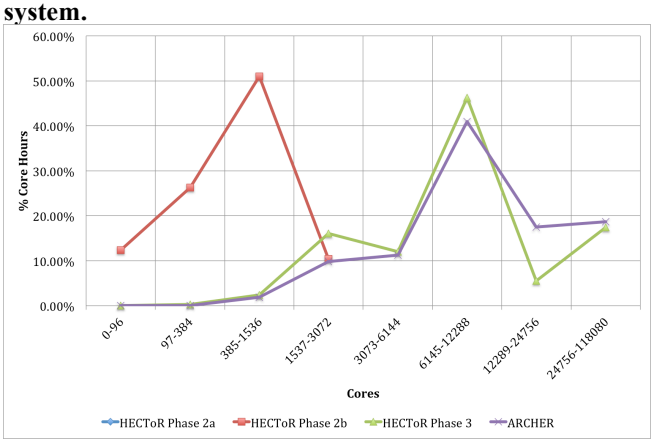


Figure 15: HiPSTAR job size distribution for each system.



V. CONCLUSIONS

Almost all of the codes that have significant usage on UK national supercomputing facilities coped well with the change in architecture from quad-core to high core count multi-core and have been able to increase the core counts that they can exploit. Most have not been able to increase the core counts six-fold (in line with the increase in parallelism on a node) but the majority has managed to increase their job sizes at least two-fold. The codes that have been able to increase their scaling in line with architecture changes are those that have a drive towards weak scaling from their research problems.

The scaling prospects for codes on the UK national supercomputing facilities tend to fall into two broad classes:

1. Those where the scaling is naturally limited by the scientifically interesting problems that are to be studied – most of the periodic electronic structure codes and the classical atomistic simulation codes fall into this category particularly when applied to biomolecular systems.
2. Those where the scaling is limited by the code development – most of the grid-based problems (climate/ocean modelling and CFD) fall into this category. Here the science tends to drive to problems that have the potential for better scaling but code development is usually required to provide the scaling in an efficient way.

Looking to the future, it may seems that scientific problems (or codes) that fall into the first class have no real prospects for exploiting even larger parallel resources in a useful way but this is not generally true. These codes and research problems will be able to exploit future architectures to employ more sophisticated sampling techniques. The trend to increasing computational power per node should serve them well in this regard and emerging sampling frameworks that are code agnostic (e.g. PLUMED, VOTCA) will continue to develop to meet this need. Codes and problems in the second class will require continued code development to exploit future HPC architectures but the potential for scaling driven by research need is inherent in the research problems they are used for.

A number of code development initiatives in the UK have provided support to allow codes to exploit new architectures. The ARCHER eCSE programme [11] and its predecessor, the HECToR dCSE programme [12] have provided focused effort to improve code specifically for the UK national supercomputing services. Many of the applications in this paper have benefitted from funding through this route: particularly CASTEP, CP2K, DL_POLY and NEMO. The Cray Centre of Excellence at the University of Edinburgh has helped develop a number of codes including the UM and HiPSTAR. In addition, EPSRC has a wider ranging software strategy [13] that takes a broader approach to improving research software: improving both the simulation software itself and the supporting software (libraries, pre- and post-processing tools). For both the classes of software discussed above continuing software development effort is key to allowing them to exploit future HPC technologies.

VI. FUTURE WORK

We will work with the ARCHER user community to decrease the number of unidentified codes on the system. We also plan to work with specific communities to understand if the picture of code usage matches their expectations, and if not, why not. We will also work with the UK research councils to understand how this analysis can help feed into future UK HPC system procurements. Finally, we would like to generalise the analysis tool so that it can generate the same information from the range of HPC services available in the UK and compare application usage across a range of HPC services.

VII. DATA COLLECTION AND ANALYSIS

The data collection and analysis tool is written in-house in Python and is available via GitHub [14]. Interrogating ALPS every hour via the apstat command collects is used to collect usage data.

The analysis tool can then process these usage logs to produce statistics on code usage. Each application is defined in a separate file that includes:

- The name of the application
- The regex that is used to identify the application from the logs (the logs store executable names)
- The primary programming language (and version) for the application
- The parallel programming model employed in the application
- The application type (e.g. structured grid)
- The primary research area for the application (e.g. materials science)
- The license type for the application

Any executable names that are encountered in the logs that do not match any of the known applications are stored and usage accumulated against them. The report then lists any of these unknown executables that have a large amount of time attributed to them so that they can be investigated and, hopefully, added to the list of described applications.

Additional options to the analysis tool include allowing the reporting to be limited to a specific period, to a particular project or group of users, specifying a custom set of histogram bins, and producing the reports in CSV format for import into other software. By default, the tool also produces plots in PNG format for quick visual inspection of the data.

The text reports submitted along with this paper provide examples of the output produced by the reporting tool.

ACKNOWLEDGMENT

Thanks to the EPCC User Support and Liaison team and the Cray CoE staff in Edinburgh for interesting discussions and comments on this paper.

REFERENCES

[1] http://www.hector.ac.uk, accessed 4 April 2015
[2] http://www.archer.ac.uk, accessed 4 April 2015
[3] http://www.epsrc.ac.uk, accessed 4 April 2015
[4] http://www.nerc.ac.uk, accessed 4 April 2015
[5] http://www.epcc.ed.ac.uk, accessed 4 April 2015
[6] http://www.stfc.ac.uk/1903.aspx, accessed 4 April 2015
[7] http://www.cray.com, accessed 4 April 2015
[8] http://www.bbsrc.ac.uk, accessed 4 April 2015
[9] http://www.nag.com, accessed 4 April 2015
[10] http://www.archer.ac.uk/documentation/white-papers/, accessed 23 April 2015
[11] http://www.archer.ac.uk/community/eCSE/, accessed 4 April 2015
[12] http://www.hector.ac.uk/cse/distributedcse/, accessed 4 April 2015
[13] http://www.epsrc.ac.uk/research/ourportfolio/themes/researchinfrastructure/subthemes/einfrastructure/software/, accessed 4 April 2015
[14] https://github.com/aturner-epcc/archer-monitor, accessed 4 April 2015