A survey of application memory usage on a national supercomputer: an analysis of memory requirements on ARCHER

Andy Turner[1] and Simon McIntosh-Smith[2]

[1] EPCC, University of Edinburgh, EH9 3JZ, UK
a.turner@epcc.ed.ac.uk
[2] Department of Computer Science, University of Bristol, BS8 1UB, UK
S.McIntosh-Smith@bristol.ac.uk

Abstract. In this short paper we set out to provide a set of modern data on the actual memory per core and memory per node requirements of the most heavily used applications on a contemporary, national-scale supercomputer. This report is based on data from the UK national supercomputing service, ARCHER, a 118,000 core Cray XC30, in the 1 year period from 1st July 2016 to 30th June 2017 inclusive. Our analysis shows that 80% of all usage on ARCHER has a maximum memory use of 1 GiB/core or less (24 GiB/node or less) and that there is a trend to larger memory use as job size increases. Analysis of memory use by software application type reveals differences in memory use between periodic electronic structure, atomistic N-body, grid-based climate modelling, and grid-based CFD applications. We present an analysis of these differences, and suggest further analysis and work in this area. Finally, we discuss the implications of these results for the design of future HPC systems, in particular the applicability of high bandwidth memory type technologies.

Keywords: HPC, Memory, Profiling

1 Introduction

Memory hierarchies in supercomputer systems are becoming increasingly complex and diverse. A recent trend has been to add a new kind of high-performance memory but with limited capacity, to high-end HPC-optimised processors. Recent examples include the MCDRAM of Intel's Knights Landing Xeon Phi, and the HBM of NVIDIA's Pascal P100 GPUs. These memories tend to provide 500-600 GBytes/s of STREAM bandwidth, but to only about 16 GiB of capacity per compute node.

To establish whether these fast but limited capacity memories are applicable to mainstream HPC services, we need to revisit and update our data on the typical memory requirements of modern codes. This is an area where conventional wisdom abounds, yet it is likely to be out of date. The underpinnings of this conventional wisdom were recently reviewed by Zivanovic *et al.* [1]. One of the key findings from this previous study is that the amount of memory provisioned on large HPC systems is a consequence of a desired high performance for HPL, where larger memory is required to achieve good scores, rather than the actual memory requirements of real HPC applications.

There are many factors which affect the memory capacity requirements of any scientific code, and these factors are likely to have been changing rapidly in recent years. For example, the ratio of network performance to node-level performance tends to influence how much work each node needs to perform, and as the node-level performance tends to grow faster than the network-level performance, the trend is for each node to be given more work, typically implying larger memory requirements. Because of these changes, we cannot rely on conventional wisdom, nor even older results, when estimating future memory capacity requirements. Instead, we need up-to-date, good quality data with which to reason and then to inform our predictions.

In this study we have used ARCHER, the UK's national supercomputer, as an example of a reasonably high-end supercomputer. ARCHER reached #19 in the Top500 upon its launch in 2013. It is a 4,920 node Cray XC30, and consists of over 118,000 Intel Ivy Bridge cores, with two 2.7 GHz, 12-core E5-2697 v2 CPUs per node[3]. 4,544 of the 4,920 nodes have 64 GiB per node (2.66 GiB per core), while the remaining 376 'high memory' nodes have 128 GiB each (5.32 GiB per core).

We set out to analyse all of the codes running on ARCHER for their current memory usage, in the hope that this will inform whether future processors exploiting smaller but faster HBM-like memory technologies would be relevant to ARCHER-class national services. Zivanovic *et al.* [1] also studied the memory footprints of real HPC applications on a system of similar scale to ARCHER. Their approach differs from ours in that they used profiling tools to instrument a particular subset of applications using a standard benchmark set (PRACE UEABS [2]). In contrast, we are sampling the memory usage of *every* job run on ARCHER in the analysis period. Thus our data should complement that from Zivanovic's study.

2 Data collection and analysis

We use Cray Resource Usage Reporting (RUR) [3] to collect various statistics from all jobs running on ARCHER. This includes the maximum process memory used across all parallel processes in a single job. It is this data that provides the basis of the analysis in this paper. Unfortunately, RUR does not include details on the number of processes per node, executable name, user ID and project ID which allow the memory use to be analysed in terms of application used and research area (for example). Tying the RUR data to these additional properties of jobs on ARCHER requires importing multiple data feeds into our service management and reporting database framework, SAFE [4]. All of the data reported in this paper rely on multiple data feeds linked together through

[3] https://www.archer.ac.uk/about-archer/hardware/

SAFE. Applications are identified using a library of regexp against executable name that has been built up over the ARCHER service with help from the user community. With this approach we are currently able to identify around 75% of all usage on ARCHER.

Memory usage numbers below are presented as maximum memory use in GiB/node. As there are 24 cores per node on ARCHER, a maximum memory use of 24 GiB/node corresponds (if memory use is homogeneous) to 1 GiB/core. Note that, as described above, the actual value measured on the system is maximum memory use across all parallel processes running in a single job. The measured value has then been converted to GiB/node by multiplying by the number of processes used per node in the job. This is a reasonable initial model as the majority of parallel applications on ARCHER employ a symmetric parallel model, where the amount of memory used per process is similar across all processes. However, if an application has asymmetric memory use across different parallel processes, this will show up as an overestimation of the maximum memory use per node. Indeed, we discuss an example of exactly this effect in the section on grid-based climate modelling applications below.

We have analysed memory usage data from Cray RUR for all applications run on ARCHER in the 1 year period from 1st July 2016 to 30th June 2017 inclusive.

3 Application memory usage

First we look at overall memory usage for all jobs on ARCHER in the period, and then go on to look at the data for the top 10 applications used on the service (these 10 applications cover over 50% of the usage). We have broken the applications down into four broad types to facilitate this initial analysis:

– Periodic electronic structure: VASP, CASTEP, CP2K
– N-body models: GROMACS, LAMMPS, NAMD
– Grid-based climate modelling: Met Office UM, MITgcm
– Grid-based computational fluid dynamics: SBLI, OpenFOAM

Due to space restrictions we are not able to include memory usage figures for all applications listed above. Instead we plot the data that best represents the trends for that application class, or that we use to illustrate a particular point. An expanded version of this paper that includes plots for all the applications listed above (along with the numerical data that was used to produce the plots) can be found online [5].

3.1 Overall memory use

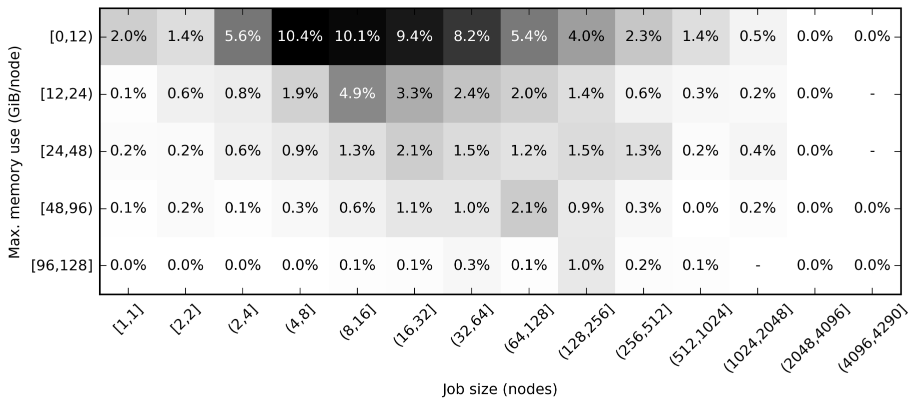
Table 1 shows a breakdown by memory use for all jobs on ARCHER in the 12 month analysis period. Additional columns show the usage for *Small* jobs (32 nodes or less) and *Large* jobs (more than 32 nodes). Just under 80% of all usage in the period uses a maximum of 24 GiB/node (1 GiB/core). Memory usage for larger jobs is generally higher, with large jobs showing only 70% of usage at a maximum of 24 GiB/node, and over 25% of usage in the range [24,96) GiB/node. These results generally echo the results from Zivanovic *et al.* [1] with the exception that we do not observe large memory requirements for smaller jobs, as seen in their application benchmarks. This could be due to the benchmarks chosen in the previous study not being representative of the usage pattern of those applications on ARCHER (see, for example, the results for CP2K below which is also one of the applications in the PRACE UEABS).

Table 1. % usage breakdown by maximum memory use per node for all jobs run on ARCHER during the analysis period. (Small: 32 nodes or less; Large: more than 32 nodes.)

Max. memory use	Usage		
(GiB/node)	All	Small	Large
(0,12)	61.0%	69.5%	53.0%
[12,24)	18.6%	19.4%	16.9%
[24,48)	11.5%	7.7%	14.8%
[48,96)	6.9%	3.0%	11.2%
[96,128)	2.0%	0.4%	4.2%

Figure 1 shows a heatmap of the usage broken down by job size versus overall memory use in GiB/node (extrapolated from the maximum process memory usage). The trend for higher maximum memory use as job size increases can be seen as a diagonal feature running from top left to bottom right.

Fig. 1. Usage heatmap of maximum memory versus job size for all jobs in the period.



3.2 Periodic electronic structure (PES) applications

The top three of the top 10 used applications on ARCHER are PES modelling applications: VASP, CASTEP, and CP2K. Although the implementation of the theory differs across the three applications, the algorithms used are similar, involving dense linear algebra and spectral methods (generally small Fourier transforms). Table 2 shows the breakdown of usage by maximum memory use for these three applications combined.

Table 2. % usage breakdown by maximum memory use per node for VASP, CASTEP and CP2K jobs run on ARCHER during analysis period. (Small: 32 nodes or less; Large: more than 32 nodes.)

Max. memory use	Usage		
(GiB/node)	All	Small	Large
(0,12)	65.4%	68.6%	55.4%
[12,24)	21.4%	20.0%	25.7%
[24,48)	9.4%	8.5%	12.1%
[48,96)	3.7%	2.7%	6.7%
[96,128)	0.1%	0.1%	0.1%

Comparing to the overall distribution (Table 1), we can see that this distribution is very similar, with a large majority of usage at 24 GiB/node (1 GiB/core) or less. This is unsurprising, as PES applications make up such a large part of the use of ARCHER (almost 30% from just these three applications, and over 40% if all similar applications are included). Only 13% of usage needs more than 24 GiB/node, and this only increases to 19% for larger jobs. The heatmaps of usage broken down by maximum memory use and job size for VASP, CASTEP and CP2K are shown in Figure 2, Figure 3 and Figure 4 respectively. When compared to the overall heatmap (Figure 1) the PES applications do not mirror the trend that larger job sizes lead to increased memory use per node. For PES applications, the larger jobs have similar memory use per node as smaller jobs.

It is interesting to compare our results for CP2K (Figure 4) with those reported in Zivanovic *et al.* [1]. In particular, they report that the small CP2K benchmark (Test Case A: bulk water) has a memory requirement of approx. 6 GiB/core running on a single node (16 cores), whereas on ARCHER, small CP2K jobs generally have maximum memory requirements of less than 0.5 GiB/core. This would suggest that, generally, the size of problem people are using these low core-count jobs to study on ARCHER is substantially smaller than the small CP2K benchmark in the PRACE UEABS.

3.3 N-body atomistic simulation applications

The N-body atomistic modelling applications, GROMACS, LAMMPS, and NAMD, also appear in the top 10 applications on ARCHER. Two of these (GROMACS

Fig. 2. Usage heatmap of maximum memory versus job size for VASP jobs in the period.

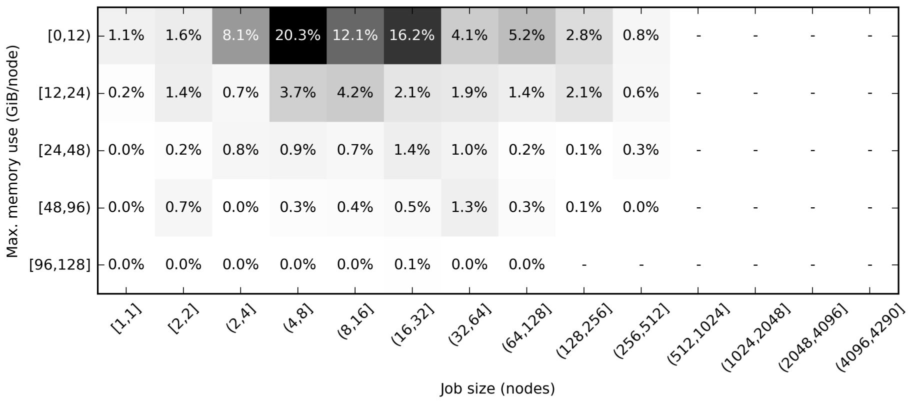
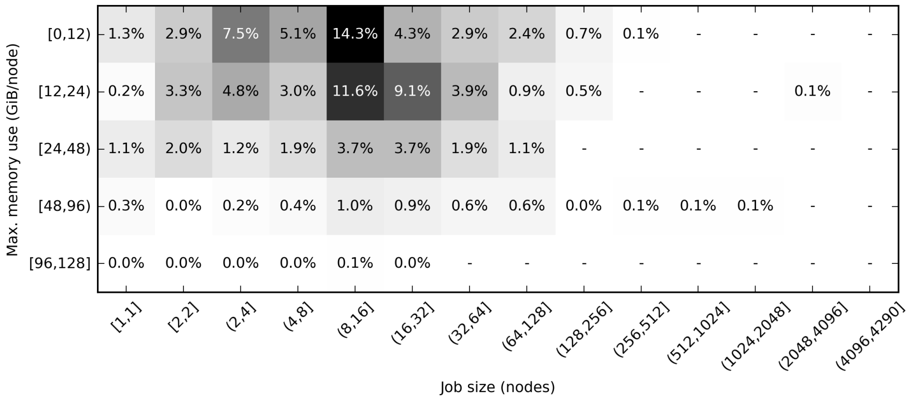


Fig. 3. Usage heatmap of maximum memory versus job size for CASTEP jobs in the period.



and NAMD) are almost exclusively used for biomolecular simulations, while LAMMPS is used more broadly across a number of research areas. All three applications use very similar algorithms, with pairwise evaluation of short-range forces and energies, and Fourier transforms for long range electrostatic forces. The parallelisation strategies differ across the applications. Table 3 shows the breakdown of usage by maximum memory use for these three applications combined.

These applications generally have the lowest memory demands on ARCHER, with over 90% of usage requiring less than 12 GiB/node (0.5 GiB/core). Even for larger jobs, only 20% of jobs require more than 12 GiB/node. Figure 5, Figure 6 and Figure 7 show the heatmaps of usage for GROMACS, LAMMPS and NAMD respectively. The heatmaps show that each of these applications have a class of

Fig. 4. Usage heatmap of maximum memory versus job size for CP2K jobs in the period.

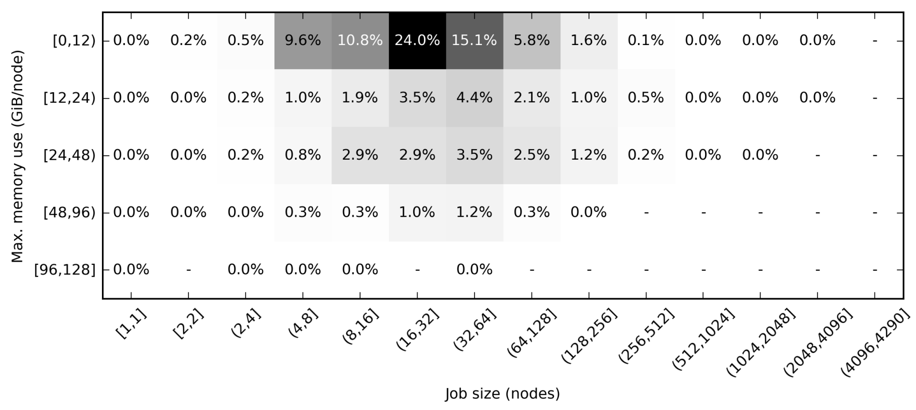


Table 3. % usage breakdown by maximum memory use per node for GROMACS, LAMMPS and NAMD jobs run on ARCHER during analysis period. (Small: 32 nodes or less; Large: more than 32 nodes.)

Max. memory use (GiB/node)	Usage All	Usage Small	Usage Large
(0,12)	91.6%	96.6%	80.7%
[12,24)	2.7%	3.1%	2.1%
[24,48)	0.5%	0.4%	0.6%
[48,96)	4.8%	0.0%	15.5%
[96,128)	0.1%	0.0%	0.1%

large calculations that have higher memory demands (48-96 GiB/node, around 4 times higher than the majority of jobs). This is particularly prominent in the NAMD heatmap (Figure 7). We plan to contact users to understand what this use case is and why it has such a large memory requirement. It is worth noting that these jobs with a larger memory requirement only represent 0.5% of the total node hours used on ARCHER in the period.

3.4 Grid-based climate modelling applications

Both of the grid-based climate modelling applications analysed (Met Office UM and MITgcm) show a very different profile from the other application classes studied in this paper. As shown in Table 4, a much higher proportion of usage use large amounts of memory, and that use of higher memory is almost always for the largest jobs. The heatmaps (Figure 8 and Figure 9) clearly reveal a very distinct split for both applications with two classes of job existing: small jobs (less than 32 nodes) with low memory requirement (24 GiB/node or less) and very large jobs (above 128 nodes) with very large memory requirements (96-128

Fig. 5. Usage heatmap of maximum memory versus job size for GROMACS jobs in the period.

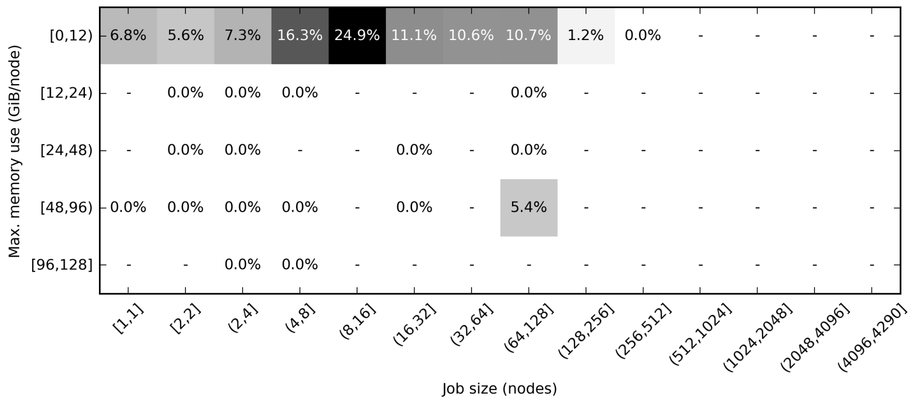
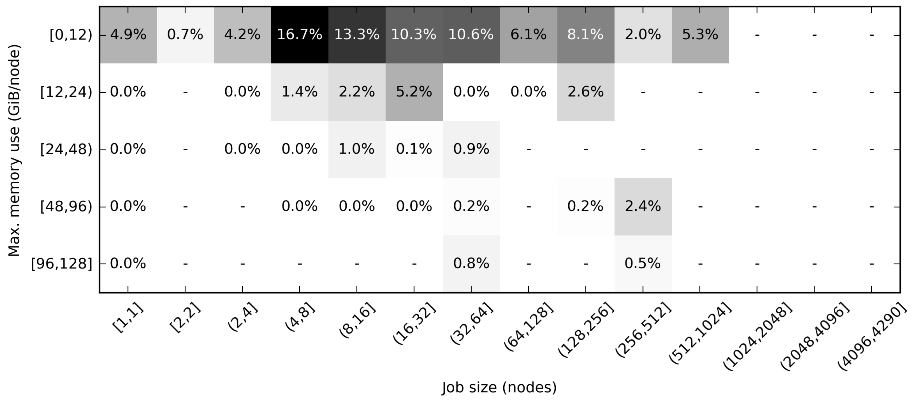


Fig. 6. Usage heatmap of maximum memory versus job size for LAMMPS jobs in the period.



GiB/node for Met Office UM, 24-48 GiB/node for MITgcm). We have performed initial investigations into this phenomenon for the Met Office UM jobs and found that it is due to asymmetrical memory use across parallel processes in the jobs. These jobs have a small number of parallel processes that have much higher memory requirements. These high-memory processes work as asynchronous I/O servers that write data to the file system while other processes continue the computational work.

3.5 Grid-based computational fluid dynamics (CFD) applications

Finally, we look at the grid-based CFD applications. Two applications appear in the top 10 applications on ARCHER: SBLI and OpenFOAM. Table 5 reveals

Fig. 7. Usage heatmap of maximum memory versus job size for NAMD jobs in the period.

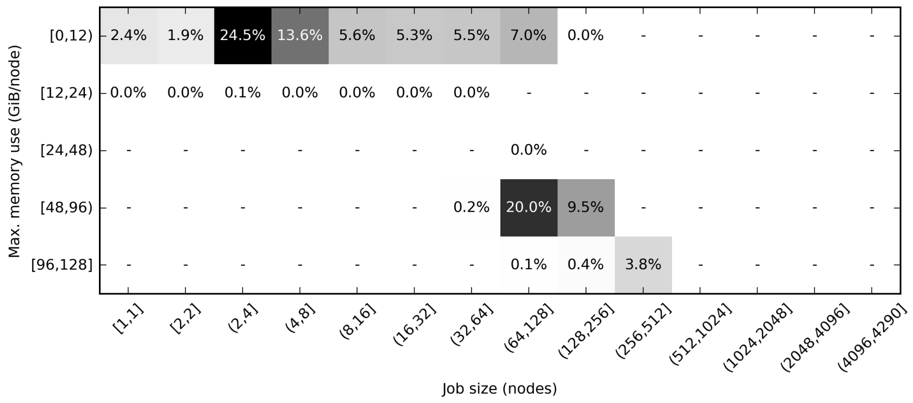


Table 4. % usage breakdown by maximum memory use per node for Met Office UM and MITgcm jobs run on ARCHER during analysis period. (Small: 32 nodes or less; Large: more than 32 nodes.)

Max. memory use (GiB/node)	Usage All	Usage Small	Usage Large
(0,12)	53.5%	66.4%	18.8%
[12,24)	25.0%	33.1%	3.3%
[24,48)	6.0%	0.2%	21.5%
[48,96)	0.2%	0.2%	0.0%
[96,128)	15.3%	0.0%	56.4%

Fig. 8. Usage heatmap of maximum memory versus job size for Met Office UM jobs in the period.

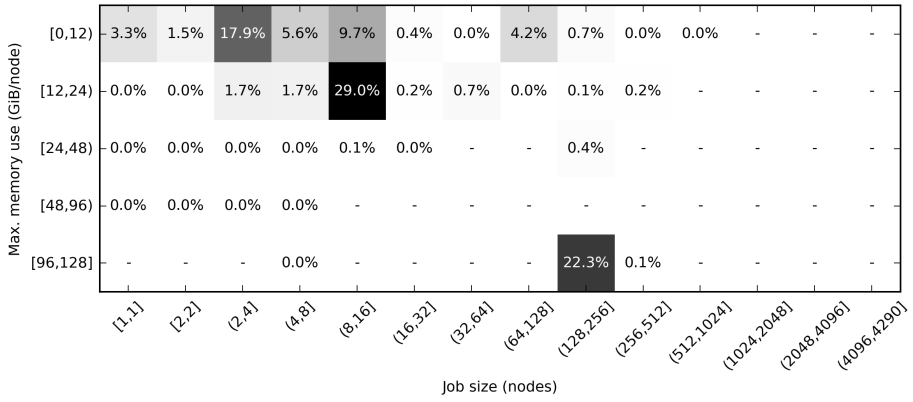
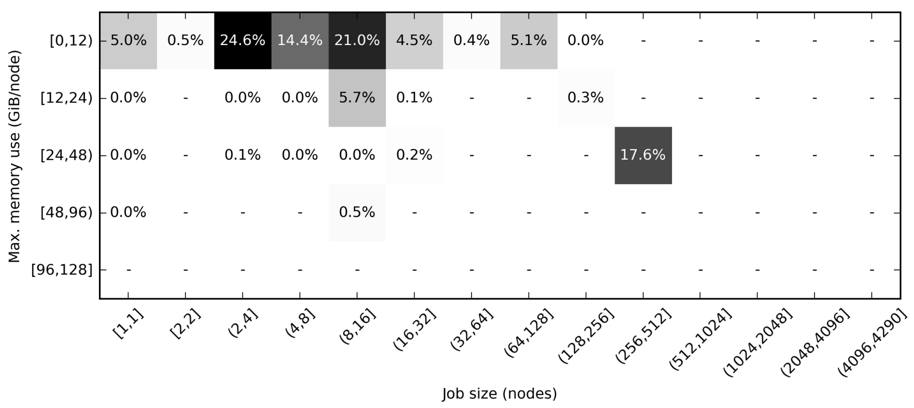


Fig. 9. Usage heatmap of maximum memory versus job size for MITgcm jobs in the period.



that they do not follow the same memory usage trend as the climate modelling applications even though both classes of application use grid-based methods and both have the same drive to higher resolution and, generally, larger jobs. The usage heatmap for SBLI (Figure 10) shows that the large jobs can have a larger memory requirement (24-96 GiB/node), but this is not always required (as was seen for the climate applications), as a reasonable proportion of the large jobs also have low memory requirement (up to 12 GiB/node). We plan to contact SBLI users to understand the differences between the jobs that have large memory requirements and those having low memory requirements. The heatmap for OpenFOAM (Figure 11) shows no clear link between increasing job size and increased memory requirements, with 94% of usage requiring less than 24 GiB/node.

Table 5. % usage breakdown by maximum memory use per node for SBLI and OpenFOAM jobs run on ARCHER during analysis period. (Small: 32 nodes or less; Large: more than 32 nodes.)

Max. memory use	Usage		
(GiB/node)	All	Small	Large
(0,12)	64.2%	71.8%	62.2%
[12,24)	14.8%	8.6%	16.4%
[24,48)	13.4%	5.6%	15.4%
[48,96)	7.7%	14.0%	6.0%
[96,128)	0.0%	0.0%	0.0%

Fig. 10. Usage heatmap of maximum memory versus job size for SBLI jobs in the period.

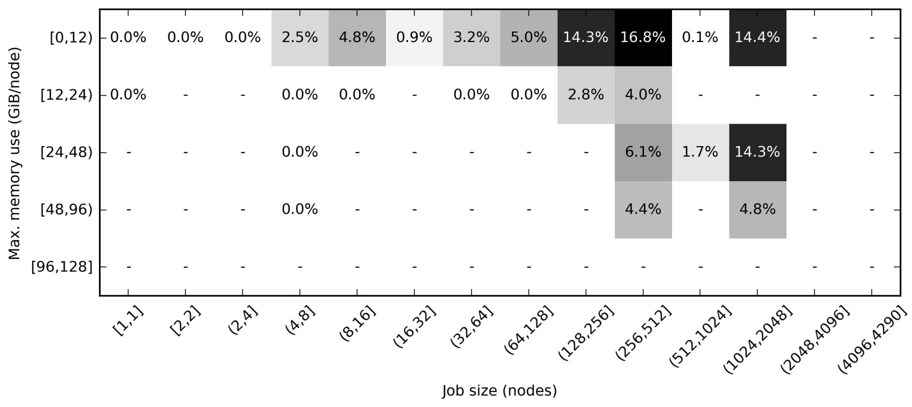
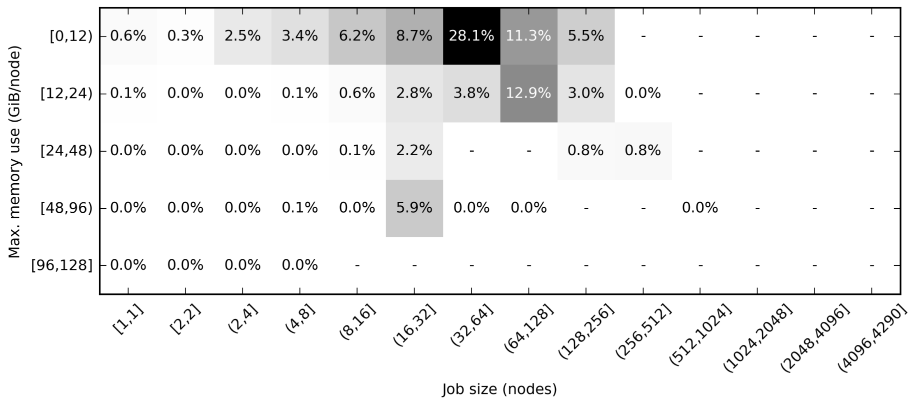


Fig. 11. Usage heatmap of maximum memory versus job size for OpenFOAM jobs in the period.



4 Conclusions and future work

Our initial analysis of memory use of applications running on ARCHER has shown that a large amount of use (80%) is under 24 GiB/node (1 GiB/core), with a significant fraction (60%) using less than 12 GiB/node (0.5 GiB/core). There seems to be a trend to increased memory requirements as jobs get larger, although some of this increase may be due to asymmetric memory use across processes. Another possible reason for this phenomenon is that larger jobs are usually larger simulations, and so the memory requirement may generally be larger. These results are generally in line with those reported for specific applications benchmarks in Zivanovic *et al.* [1], with the exception that we do not see large memory requirements for small jobs as reported in their study.

We also illustrated one weakness in our current analysis, when memory use between parallel processes is very asymmetric. As the analysis is based on maximum process memory use extrapolated to a per-node value, parallel processes with very different memory use within the same application can produce misleading estimated memory use figures. We plan to refine our analysis methodology to take this type of asymmetric memory use into account for future work.

Our analysis leads us to conclude that there is an opportunity for exploiting emerging, high bandwidth memory technologies for most of the research on ARCHER. Many applications from a broad range of research areas have performance that is currently bound by memory bandwidth and would therefore potentially see significant performance improvements from this type of technology. The data in this paper suggests that, even memory was as low as 0.5 GiB/core, two-thirds of the current workload on ARCHER would be in a position to exploit this, without any software changes. Expanding to 1.0 GiB/core would address nearly 80% of ARCHER's current workload. Our results (and results from previous studies) suggest that a future ARCHER service could even benefit from architectures where HBM-like technologies with limited capacity *replace* main memory, rather than using a hybrid solution (such as the MCDRAM+DRAM seen on the Intel Xeon Phi). The reasoning here is that using HBM technologies as a main memory replacement allows applications to access the best performance *without application code modifications* whereas in the hybrid approach the only way to use the HBM without code modification is as an additional, large cache level, which can limit the performance gains available [6]. Another option would be to use a combination of processors with high memory bandwidth alongside processors with high memory capacity. Ideally the processors would have identical cores in them, making it easy for a job scheduler to allocate any job to either the high memory bandwidth partition, or the high memory capacity partition, as necessary.

In addition to refining our analysis technique using this new data from ARCHER, we need to work with the user community to understand the different memory use classes for particular applications and research problems. This work will help us make sure that future UK national supercomputing services provide the best resource for researchers.

References

1. Darko Zivanovic, Milan Pavlovic, Milan Radulovic, Hyunsung Shin, Jongpil Son, Sally A. Mckee, Paul M. Carpenter, Petar Radojkovi, and Eduard Ayguad? Main Memory in HPC: Do We Need More or Could We Live with Less?. ACM Trans. Archit. Code Optim. 14, 1, Article 3 (March 2017) DOI: https://doi.org/10.1145/3023362
2. PRACE. 2013. Unified European Applications Benchmark Suite. http://www.prace-ri.eu/ueabs/ (accessed 21 Sep 2017)
3. XC30 Series System Administration Guide (CLE 6.0.UP01) S-2393https://pubs.cray.com/content/S-2393/CLE%206.0.UP01/xctm-series-system-administration-guide-cle-60up01/resource-utilization-reporting (accessed 21 Sep 2017)

4. Stephen Booth, Analysis and reporting of Cray service data using the SAFE. Cray User Group 2014 Proceedings. https://cug.org/proceedings/cug2014_proceedings/includes/files/pap135.pdf (accessed 21 Sep 2017)
5. ARCHER. 2017. Memory usage on the UK national supercomputer, ARCHER: analysis of all jobs and leading applications. http://www.archer.ac.uk/documentation/white-papers/ (accessed 21 Sep 2017)
6. Milan Radulovic, Darko Zivanovic, Daniel Ruiz, Bronis R. de Supinski, Sally A. McKee, Petar Radojkovi, and Eduard Ayguad? 2015. Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?. In Proceedings of the 2015 International Symposium on Memory Systems (MEMSYS '15). ACM, New York, NY, USA, 31-36. DOI: http://dx.doi.org/10.1145/2818950.2818955