Requirements for the Future DRI Ecosystem

Simon Hands and Ed Bennett

Contents

Foreword

This report is an important contribution to the realisation of the UKRI vision for "a coherent state-of-the-art national Digital Research Infrastructure (DRI) that will seamlessly connect researchers and innovators to the computers, data, tools, techniques and skills that underpin the most ambitious and creative research." Such a DRI has the potential to transform many areas of research across UKRI and the wider UK economy.

The workshop whose discussions are presented here was supported by the DiRAC HPC Facility as part of the DiRAC Federation (DFed) Project, funded by the UKRI DRI Programme. It built on workshops held in 2022, also under the DFed Project auspices. The workshop reflected the DiRAC vision of broadening discussions on the shape of the future DRI to ensure that it will be able to support activities across all UKRI themes. In particular, the DRI should provide the right resources for both compute and data, at the right scales, to enable all UKRI researchers to carry out their research in the most cost-effective, efficient and environmentally sustainable way.

The report reflects the diversity of specific DRI requirements across the UKRI community, including: the need for significant investment in software support; the need to address skills shortages for re-factoring of codes to make use of GPUs; the importance of community building and training to ensure that researchers are equipped to make use of new computing opportunities; and the value of a workflow-based approach to system design.

The workshop included in-depth discussions of important issues beyond hardware and software challenges, such as: career paths for DRI professionals; access modes for large-scale computing services; data challenges for computational projects; training.

I would like to thank Simon Hands and Ed Bennett for their hard work in compiling this report and all the participants in the workshop whose insights have been captured here. I expect that this report will be an immensely valuable resource to support the work of defining the UKRI DRI over the coming years.

Mark Wilkinson

DiRAC Director

1 Introduction

In 2022, DiRAC hosted two workshops to explore the use of workflow mapping as a means to define the technical requirements of future UKRI large-scale computing services. The goals of the follow-up workshop in late 2023, and of this report, were to expand the discussion to include all aspects of the Digital Research Infrastructure (DRI) Ecosystem and to move the discussion of computing requirements forward ahead of the various formal requirements gathering exercises which are planned for early 2024 (e.g. STFC Exascale Requirements; UKRI Pathway to Exascale requirements).

Given the broad scope of the topics covered and areas of research represented, there is a natural diversity in some of the requirements for the DRI ecosystem if it is to support the full range of UK science. As will be seen in this report, there are cases where these differences appear rather stark and may require multiple, complementary solutions to be deployed in order to meet mutually inconsistent requirements. At this stage, we have therefore retained the conflicting requirements in this document to encourage further discussion of potential solutions as the DRI ecosystem is defined and deployed.

Our intention is that this report can be the basis for additional consultation leading to a science-driven DRI ecosystem which will deliver for UK researchers in both academia and industry.

1.1 Methodology

The main purpose of this part of the Federation Project was engagement with computational scientists across several fields (biomedical science, engineering, condensed matter and materials science, particle physics, cosmology, plasma and fusion science, large language modelling) to share and exchange insight and experience, and in particular to attempt to set out design principles and strategies for UKRI-wide computing services in a future UKRI Digital Research Infrastructure.

Individual scientists from these domains were invited to participate on the basis of their expertise and/or standing, and in many cases previous engagement with DiRAC. A subset of those approached are currently actively implementing domain-specific workflows using a discretionary allocation on DiRAC facilities made available under Phase 2 of the DIRAC3 procurement, having attended a 2-day onboarding workshop alongside DiRAC technical staff in June 2023.

The invitees attended a residential workshop held in Leicester on 4th-5th October 2023, whose proceedings form the basis for this document. In advance of the workshop, participants were sent a questionnaire (Appendix C) seeking:

- a brief description of their scientific area;

- quantitative information of the system and service requirements supporting their current workflows;
- software requirements including an estimate of the effort needed to adapt to future anticipated computing resources;
- hardware or architectures of particular importance for their workflow.

The questionnaire also asked participants to identify relevant numerical methods from a list of computational "motifs", as set out in the table below. The purpose of this pre-workshop "homework" was not so much to obtain definitive answers but rather to prepare participants for the meeting, and perhaps to encourage them to consult in advance with other technically-inclined people in their research consortia.

Following an introductory presentation by DiRAC Director Mark Wilkinson, participants worked in one of four groups depending on which table they were sitting at. At each table sat a rapporteur identified in advance of the meeting, charged with leading the conversation and with keeping a record using a template pro-forma prepared for each individual table using a Google doc. Participants were given access to the doc to allow the addition of comments. The groups focussed on a series of questions set out in an agenda: on Day One these concerned the scientific problems being addressed, the current methodology and system requirements, and a forward look to future needs, finishing with a first attempt at defining a set of common ecosystem requirements. On Day Two the same format was used to discuss what is still needed to be known before the ecosystem can be properly specified, and how and where to find the missing information; and whether there are any immediate actions or investments which could improve the suitability of current systems.

The workshop format on the morning of Day Two was switched so that groups circulated around the four tables where the rapporteurs, each charged with leading the discussion of a specific issue, were based. Participants were invited to record their thoughts using coloured Post-it notes on a flip-chart, the colours chosen so that responses of later groups to the initial ideas of preceding groups could be traced. The flip-charts were retained at the end, along with a sequential photographic record.

The four issues tackled in this approach were:

- *Developing career paths for DRI professionals;*
- *Access modes for large-scale computing services;*
- *Allocation processes for large-scale computing services;* and
- *Data challenges for computational projects.*

The report that follows is based on all the material collected: questionnaires, Google Docs and Post-It notes. Due to the differing formats and also because the workshop was mainly conducted through parallel rather than plenary sessions, it has proved almost impossible to keep to a uniform style, and there is inevitably both repetition and contradiction. Rather, the document records the focussed engagement of a group of research leaders and professionals with a moving target.

2 Why is DRI needed?

Which science goals depend on large-scale computational resources?

2.1 Plasma & Fusion Science

Fusion plants are expensive capita projects (ITER is currently costed at £40B) and due to the extreme operating environments must be designed *in silico.* Two current simulation projects concern tokamak design (STEP) and studies of inertial confinement. High performance computing is also integrated into operations via plasma control systems. It is of key importance that systems comply with regulatory requirements.

2.2 Cosmology

The large-scale structure and mapping of the Universe present fundamental questions, such as the nature of dark energy and dark matter, and the correct theory of gravity at large scales. Observational programmes (e.g. Euclid, DESI, LSST) all require virtual simulation counterparts to exploit their full potential. The models require large-scale simulations, but also large samples in order to quantify statistical deviations. The long-range nature of gravity means equations of motion need to be solved on the whole system, with no compartmentalization possible. There are challenges handling huge datasets, associated with checkpointing, storage and long-term labelling needed to ensure data is readily accessible to external users.

2.3 Engineering

Companies such as Rolls Royce now conduct virtual certification of engines. The computational cost of system and component design rises rapidly as further detail is incorporated into the model. Forward models, which predict outcome actions based on input control parameters, currently exhaust computer capacity. Within the commercial sector there is inertia against transitioning to parallel open source codes, with perhaps a 50-fold performance gain required to initiate change. All data needs to be monetized, because of the practical difficulty of storing everything.

Studies of phenomena such as turbulence feeds into aircraft design aeroplane (wings of tomorrow), and wind energy generation; here different physics domains may be coupled, such as weather forecast with wind turbulence. Looking forward, it will be important to move from deterministic models into models with uncertainty quantification; our primary requirement will be not so much *accuracy* but rather *fidelity*.

2.4 Experimental Particle Physics

There is a distributed, federated infrastructure already in place to process the data generated by particle physics experiments (estimated volume 2 Exabytes), with tasks ranging from raw data handling to more advanced analyses determined by the end-user. Most computational effort is required for the simulation of either physical processes or detectors. The goal is for the statistical precision of the simulation *not* to be the limiting factor in any measurement, which should rather be set by the performance of the accelerator.

2.5 Large Language Modelling

Strategically, UK needs LLM support to be economically competitive. Currently the field is dominated by US-based companies such as OpenAI (ChatGPT). Can other countries or geopolitical entities design such systems that are tailored to their needs and preserve a necessary degree of independence?

The development cost for LLM is typically $10M-$100M; how generalisable are these models? Training data are often opaque. Can aspects of these models be "translated" to other languages or social/political/legal contexts? We need to be able to support languages other than English (e.g. in the UK, the Welsh language has legal protection, so we need models which also work in Welsh). All these are currently-open scientific questions of national importance, which can only be answered via access to sufficient computational resources.

2.6 Biomedical Science

There is growing use of -omics data, ie. data relevant to the entirety of processes within a cell or organism (e.g. multi-omics projects bringing together genetic, metabolic, lipid and protein data), alongside patient data, care pathways, and environmental data. Modelling is important — the Covid pandemic has demonstrated the value of molecular modelling, evidenced by the award of the 2021 Gordon Bell prize. There are also projects modelling cellular and physiological systems. An example grand challenge is the large-scale modelling of the entire human vascular system (e.g. on Frontier).

The computational challenges include the Integration of multimodal data; accessing or federating with sensitive data; creation and integration of datasets representing individuals and groups rather than just averages. Strong scaling delivers the opportunity to run for longer and thus study effects over different temporal scales.

2.7 Condensed Matter & Materials Science

Quantum chemistry/physics is a large-scale computational project, with HPC the workhorse technology. Increasing precision is needed to reach the point where model predictions match experiment—many results from 10 years ago don't manage this, due to effects that could not be captured within the performance envelope available at the time.

The Materials Chemistry Consortium has identified several "exascale" themes where chemical and physical accuracy matter, and it's important to have a large system size:

- **Bio & soft matter**: developing force-fields
- **Bulk materials**: defects & disordered material models
- **Power**: energy conversion & storage (battery materials, PhotoVoltaics...)
- **Reactivity**: predictive computational catalysis
- formation of protective oxides on aluminium
- interactions of CO_2 with specific surface substrates.

Modelling realistic systems is challenging, because there isn't the capability to scale up current methodologies to attain the precision needed, because resources aren't available to handle 1000s or 10000s of atoms, and because we can't get results in reasonable time. Moving to "classical" molecular dynamics enables bigger systems at the cost of precision.

3 Computational methodologies

How do the scientific workflows map to numerical methods?

Here delegates were invited to match their scientific workflows against numerical methods, which for the purposes of this exercise were encapsulated in computational templates known as "motifs". For further background please consult https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf.

Classically there are seven motifs, but these have been extended by the addition of a further seven more sophisticated variants. We have chosen to present the outcome of the discussions and submitted information in tabular form, with each row corresponding to one of fourteen motifs. A single tick indicates that the method is in use, and there are two ticks if there are workflows where more than 40% of the effort resides with a particular motif.

	Plasma/ Fusion Science	Cosmology	Engineering	Particle Physics (HTC)	Large Language Models	Bio-medical Science	Lattice QCD	Condensed Matter/ Materials Science
Dense Linear Algebra	✓	✓	✓		✓	✓		✓✓
Sparse Linear Algebra			✓✓	✓	✓	✓✓	✓✓	✓✓
Spectral Methods	✓	✓	✓	✓				✓✓
N-body Methods	✓	✓✓	✓✓			✓		✓✓
Structured Grids	✓	✓	✓✓			✓✓	✓	✓
Unstructured Grids	✓	✓	✓✓			✓		
MapReduce/Monte Carlo	✓			✓		✓	✓	✓
Combinatorial Logic								
Graph Traversal			✓			✓		
Dynamic Programming	✓					✓		
Backtrack & Branch-and-Bound								
Graphical Models						✓		
Finite State Machines								
Uncertainty Quantification	✓							
Others								

Clearly this is a coarse-grained picture, and individual responses to the question reveal much more detail. The dominance of the classical motifs is, however, somewhat striking.

4 Current Computational Requirements and Issues

What do you need for this at present?

Again, it's very hard to present a unified overview, but the following remarks provide a useful snapshot of current practice and technical sophistication. Some delegates also responded to the request for long term storage requirements, on a per-project basis.

4.1 Plasma & Fusion Science

UKAEA is the biggest contributor of hardware to CSD3 at Cambridge, mainly x86 (10000 cores) and GPUs (44xA100), with 6xL40 nodes for advanced visualization, and a Lustre parallel file system. The UKAEA allocation at CSD is managed by Cambridge, with the possibility of exceeding allocation by 25% provided it's paid back the following quarter. This is supplemented by a local cluster with 3000 cores and Lustre. The resource is currently divided between two projects: STEP (tokamak design) and inertial confinement studies. There is a need for more visualization capacity.

The simulations require a mix of GPU and CPU. The Marconi system is entirely GPU-based – support is needed to adapt codes to GPU development. LAMPS/VASP are single-node and consume a lot of Archer-2 time.

4.2 Engineering

There's no immediate shortage of compute power – but tasks run at extreme scale. There is a need for more HTC than current systems can handle. Also, it is crucial that the system reports back when nodes are down, since otherwise remedial action is difficult. Often the default MPI library doesn't scale well beyond 16 nodes, yet alternative libraries leak memory. MPI can take 10 minutes to initialize, with the time scaling quadratically with number of nodes. Configuration of jobs requiring communication between GPUs is challenging for many users.

4.3 Cosmology

Current requirements are a large memory footprint and fast checkpointing. Most algorithms are not obviously suited for GPUs in the sense that no large performance gains are anticipated from changing architecture; however, the codes are portable and switching to GPUs may well yield an improved energy-to-solution than that obtained currently.

4.4 Experimental Particle Physics

Particle physics is largely supported through GridPP for large-scale compute, with approximately 100,000 x86 cores across a Tier 1 and a distributed Tier 2 infrastructure. GPU workflows are nascent, being anticipated over the coming years but not currently a large part of processing requirements. Software has co-evolved with hardware and requires only modest memory per core. Hierarchical storage (fast local temporary storage, tape storage) with distributed data management is critical for many workflows. Jobs typically run across ~8 cores at a time, with large parallelism overall but no strong coupling between jobs. The long history of large-scale compute means the user base is reasonably familiar and most experiments include training on accessing large-scale compute.

4.5 Biomedical Science

Lattice-Boltzmann codes are RAM-limited, so benefit from high RAM per node. RAM per core is an issue for many materials modelling packages.
Memory bandwidth (rather than capacity) is a limiting step for, e.g. FFTs.
GPU performance is working well, with runs on 74,000 GPUs on Frontier demonstrating both strong & weak scaling. GPU has a higher software maintenance cost (OneAPI, CUDA, HiP).

Frustrations in accessing and using Research Organisation resources and the affordability of computing in the past has led to smaller scale HPC users in some fields procuring and running their own machines. Increasingly, competitive research requires collaboration and access to more significant computing resource; community building is required to facilitate key enablers such as data standards, community code development, and computing access.

- An example; code written using a sorting algorithm originally performed adequately, but running on a bigger problem caused the machine to get stuck in single-core mode for 20 minutes, because the sort step didn't parallelise.
- An optimal algorithm depends heavily on a specific data structure. Different steps in a computation have a different optimal data structure, so a significant part of workflow is transforming data structures from one step to the next. Perhaps a better solution is to pick a different algorithm/data structure that better matches the available hardware?

Long term storage requirements - More than 100 TB! Input alone might be 100 TB, snapshots similar (50-100 TB) – maybe 1 PB required in total.

4.6 Condensed Matter & Materials Science

There are many codes (eg. CRYSTAL, CP2, SIESTA, VASP) which work well on a single node, but their scaling to many nodes is less well-understood. New methodologies are needed to attain acceptable modelling fidelity on sufficiently large systems. CPU-based machines are very bad for this, but there are new methodologies which are very optimised for GPU running. E.g. new methods based on ML (ACE, MACE, NICE) employ the PyTorch and TensorFlow packages; the resulting performance on GPU is absurdly better than CPU.

The optimal solution is strongly dependent on the specific task; some calculations are well-defined in size, and parallel scaling limits define the maximum core count; other calculations are restricted at present to "ideal" models, so more CPUs means that larger, more complex and therefore realistic models could be used.
Long term storage requirement ~ 100 TB.

What would happen to the materials community if we unilaterally picked e.g. a performant GPU-based machine and required everyone to adjust to it? Here there is an issue with the codebase: many codes would lag and only be ready to use once the machine was obsolete.

4.7 Large Language Modelling

LLM training is inherently massively parallel, with each node ideally hosting the largest number of GPUs possible in order to minimise the data to be sent between nodes, which greatly slows the

process (as was the case for the Japanese LLM efforts). Currently, a model with moderate research impact needs about 256 high-memory GPUs located over 32 nodes. However, for experiments at the cutting edge we would need closer to 128 nodes, and multiply that by a factor of up to 10 in order to make a complete exploration of the experimental parameter space. New cards with greater processing speed and memory are being released each year, enabling models with further capabilities.

While training a model the storage requirements are significant, firstly because the model can hypothetically ingest a near-infinite amount of text that needs to be stored and processed. Secondly, to understand better the training dynamics and whether adjustments need to be made to improve the model, or to avoid wasting training resources on a model that is doomed before it finishes, it is necessary to checkpoint at regular intervals, by saving the model in its entirety to disk. This can easily require hundreds of TBytes, approaching a PByte for more ambitious LLM projects.

5 Anticipated Future Needs

How will these needs evolve over the next 5 years?

5.1 Cosmology

- Incorporation of radiative transfer
- Incorporation of gas-photon interaction, requiring embarrassingly-parallel small matrix inversions on individual resolution elements
- Decoupling of regions in time to improve strong scaling
- More memory will be needed since everything is held in RAM because of the continual need to solve the Poisson equation on the entire system.
- Improvement in data movement speed and possibly reformatting between the requirements of MPI, CPU and GPU.
- Investigate use of matrix intrinsics on Sapphire Rapids (next-generation CPU).

5.2 Engineering

- There is some frustration with Archer-2 due to poor performance of MPI libraries and reliability/documentation issues
- Investigation of new architectures and new algorithms
- Increasing sophistication of systems will present challenges across a wide user base
- ARM chip on RAM to reformat data into different orders for improved data mobility.

5.3 Plasma & Fusion Science

- Need an accessible interface, eg. working libraries, compiler, etc.
- Need more CPUs and GPUs, but on a timescale ~3 years, CPU requirements will start to decrease with the advent of AI-based surrogate models to replace numerical models
- Need more memory to permit finer resolution to expose, eg. micro-tearing modes of plasma.

- Increased use of High Bandwidth Memory (HBM), currently under-exploited, but may become more valuable following transition to GPUs.
- Grace-Grace may outperform Grace-Hopper.

5.4 Experimental Particle Physics

- Need more GPUs – the current fleet is running at full capacity. Current work is looking at rewriting core codes to work on GPU architectures. This will be needed for future data-taking at the LHC due to high event rates and increased complexity. It may prove the only affordable way to meet future computing needs and is also anticipated to improve energy-to-science.
- Need hierarchical storage.
- Some work has been done looking at other accelerators, but on a relatively small scale. Examples include Grace-Hopper, also FPGA arrays.

5.5 Biomedical Science

- 1 PByte storage is currently adequate but anticipate a need for 2-5x increase over next 5 years.
- More CPUs will yield more capacity.
- more GPUs to implement ML.
- Flexible systems, heterogeneous architectures (eg. CPUs and GPUs) that enable short-term episodic access as well as large time allocations.
- Infrastructure to enable the use of sensitive data, secure research environment, and pipelines to enable the use of distributed data with HPC.
- Support refactoring code to make the most of emerging hardware

5.6 Condensed Matter & Materials Science

- 1 PByte storage is adequate for next 5 years
- More CPUs will yield more capacity
- Codes to rely more on sparsity and locality of interactions,
- more use of AI, exploiting newer generations of hardware.
- More fault-tolerance and resilient algorithms.
- Move to use more GPUs in a hybrid fashion, on ML training part.
- ML input datasets are small, but the output is very large. They need to access a lot of distributed databases.
- There is a skills shortage in porting codes to accelerated architectures.

5.7 Large Language Modelling

- Need frequent refreshment of GPU hardware –
  huge performance increase as P100 -> V100 -> A100 -> H100, enabling new research questions and model categories to be studied
- steady increase in GPU numbers (10x every 4 years)

6 Can we identify Common Ecosystem Requirements?

6.1 Data Curation & Management

- Large metadata database which can be easily interrogated, and returns sensible information when a search fails so that ongoing workflows can continue
- Data management
- Persistent memory close to the node (eg. local disks) is key for ML and large-data workflows
    - Co-processors for IO or intelligent networks would enable us to hold data locally. For many workflows, pushing extreme scaling is of limited value. We don't need more nodes, but a steady growth of GPU numbers would help.
- In summary, where possible switch from data movement to a better orchestration of computations

6.2 High Throughput Computation

- Exploit network-light computing, ie. prioritise compute over network communication.
- Need for HTC system where the key characteristic is per-node (or few-node) performance, e.g. high memory bandwidth, low memory latency, high core performance (clock speed?), less emphasis on interconnect; could be CPU or GPU.

6.3 Exascale

- Will need several exascale systems over a period of years, which will ensure expertise and infrastructure is distributed, better management and delivery of training, and avoid vendor lock-in and maximise the chance of having the best hardware for the problem at hand.
- Recognise the need for continuity, ie. overlap between a service and its successor.
- Exascale support infrastructure will be needed, working with software library developers.
- For many workflows, pushing extreme scaling is of limited value. Such workflows don't need more nodes per job, but a steady growth of GPU numbers would help. There are plenty of problems that push the limits of the scalability of their algorithms to multiple nodes already; larger machines give better throughput and enable big ensemble calculations but the number of true strong-scaling exascale hero problems is small. In this case having denser compute (more GPUs per node) can give more performance boost.

6.4 Testing & Development

- There is a need for Testing & Development HPC, with fast queue turnaround to facilitate rapid code-test-debug cycles, run-profile-optimise cycles, etc.
- multi-architectures are very useful.
- Reproducibility of computations and data handling as well as validation and verification.

6.5 Private/Commercial Cloud Integration

- Cluster-as-a-service model, so that workflows can temporarily access commercial cloud services if on-site/private cloud HPC provision is insufficient.

6.6 Software

- Code ownership will become more of an issue. Many important codes are driven by consortia based overseas, and we can only operate within constraints imposed by them. We will need to understand where international code is suitable for research communities, ensuring the resources enable adaptation and use for UK research questions, and where communities are limited by their reliance on code and require further development support.
- Support by UKRI for long-term code development is missing; Hartree et al. have no core money to support research outside of labs and are therefore centre-centric rather than outward looking. We'd need the opportunity to have at least 1FTE per large-scale code in a central, national unit.

6.7 Machine of the future

- It's not clear if we need hybrid chips since they cause difficulty on the software side.
    - Finally give us a "write once run everywhere" API; or
    - Integrate USPs of GPUs into CPUs.
- Either have many GPUs and few cores or many cores and few GPUs (current trends are discouraging). Two specialised machines might be better than one that suits all.
- Cost of tight GPU integration is worrying.
    - For example, NVIDIA Grace Hopper carries a high financial price tag, and also sacrifices flexibility; each GPU is tied to 72 CPU cores, even if not all of them can be used by a particular problem.
- There is clearly a tension between hardware procurement and software maintenance. Machine refreshment should not enforce a software rewrite. Therefore, there is a driver for tighter integration of GPUs and CPUs and more bandwidth. Upscaling solely compute capability without also improving memory and interconnect bandwidth, etc. will not give an appreciable benefit to the majority of codes.
- Future workflows will have (close to) sequential phases running on small local workstations taking turns with massively parallel steps. Integration of the two modes will be a challenge.

7 Building an effective DRI

In the following subsections, text marked in **bold** are direct quotes from workshop participants, while the surrounding text is presented to provide sufficient context to understand their intent and the connections between them.

7.1 How do we develop career paths for DRI professionals?

Work around the Technician Commitment and by the Society of Research Software Engineering has, in recent years, sought to develop new job titles, job descriptions, and career pathways for DRI professionals (among others). However, there is not a **national consensus** around what this pathway looks like: while there are commonalities, each institution has its own pathway, and

some still suffer from the same problems that have been discussed in the Research Software Engineering community since its inception over a decade ago.

At many institutions, there remains a hard division between *academic* staff, who are appointed onto a pathway (Lecturer → Senior Lecturer → Associate Professor → Professor), and may apply for and be awarded promotions based on specific guidelines, and ***professional services* staff,** [who] **"cannot" be promoted, "must" apply for a new role if they wish to advance to a new grade, and this new role "has to be" advertised externally**. Discussion among participants suggested a need to more generally **fix HR processes**. If providing to DRI professionals a career pathway comparable to that available for academics is to be a priority, then **sites should only be eligible to host a service if their HR model** [and other bureaucracy] **allow for promotion in place for DRI staff**, and/or **they have a defined pathway for this**. The aim should be to **recognise levels of experience and skills, and reward** [them] **appropriately**.

There is a need for an evaluation of **best practices** [and] **approaches for inclusion, resourcing, development, and mobility of DRI skills and teams at different career stages, highlighting universities doing good work**, to provide **blueprints, case studies, and impact reports** on which institutions can pattern their models to gain eligibility.

DRI professionals carry with them a body of **skills, experience, and knowledge within large DRI projects** that is not easily replaced, similarly to other technical staff working with and on large facilities. A strong incentive to retain and promote RSEs based within a research group is continuity; too often time is spent training RSEs not in technical skills but rather in providing insight into the scientific applications on which they are working. While in most institutions (as a result of HERA grading requirements or otherwise), higher-graded positions require a level of management of people (and in the case of academic roles, significant amounts of teaching) as well as technical work, a model that has been developed in the private sector and applied in some institutions is to **establish technical promotion paths**, allowing DRI professionals to focus on developing technical excellence rather than compromising technical development to develop and apply people management and teaching skills instead; the position of **staff scientist** is used at many institutions internationally to describe such roles, and a question was raised as to whether funding for staff scientists focusing on specific tools or communities could be provided as part of a future DRI service.

Collaborative Computational Projects (CCPs) and **High-End Computing Consortia** (HECs) are another way that has successfully retained computational skills within communities. Identifying technical professionals as **"core" staff on consolidated grants** is another approach that has been used to retain vital technical skills in research communities, although to date this has seen limited use for DRI professionals specifically. Additionally, there was discussion on how competing codes in similar areas should be supported; whether there should be prioritisation of software based on popularity, efficiency, or if the primary development activity is in the UK.

Where DRI professionals do seek to move institution (in search of career advancement, due to limited opportunities at their current institution, or otherwise), it is currently difficult to identify

equivalence of roles. Due to the lack of consensus identified above, each institution will have its own set of roles—even where job titles are consistent (e.g. Research Software Engineer), the associated role descriptions are bespoke to each institution, and the technical nature of the work means that each role will have a very different set of responsibilities and required criteria. A framework for understanding the skillset of a DRI professional is vital to be better able to compare different roles and one's relative suitability for them. Options suggested by discussion participants included a **national accreditation scheme**, and **skills and competency-based awards**. Work is ongoing within the Research Software Engineer community to develop such a framework for RSEs, but a more general one (either as a superset or a parallel, complementary classification) would be needed to cover the wider space of DRI professionals.

There may also be a benefit to **increasing academic–industrial interactions and collaborations in DRI**, to give more cross-pollination of best practices, skills, and people between academia and the private sector. It was also recognised that the significant upcoming investment in AI and exascale facilities will require significantly skilled research software engineers who **both know the domain but can also implement algorithms** to prepare the code to run on these facilities; it was recommended to fund a **CDT in the area of developing exascale RSEs**, including in **disciplines without a computational heritage**, but a concern was raised that the current **PhD education has the wrong focus and is the wrong route to acquire these skills**—one possible resolution discussed was that software should be recognised as an output of the funded PhDs, potentially submitted in lieu of the narrative thesis. Other alternatives raised were **new job descriptions** for developing experienced HPC-facing RSEs, and to develop **better separation of concerns** so it is less essential for the same individuals to fully understand the implementation specifics and the domain to which they apply—the latter of these may be considered as aspirational rather than immediately achievable, however, requiring as it does a paradigm shift in the way computational research is conducted.

Despite work by, for example, the Hidden REF, there remains little recognition of the work of DRI professionals outside of their projects and disciplines. The value of this work and the people performing it could be highlighted via **awards, badges,** or **recognition** schemes, such as a **UK version of the Gordon Bell award**. However, the impact of DRI is not cleanly captured via the current REF system—individual institutions' DRI may be captured in the environment statement, but this cannot be collated across the sector and may not be included depending on other competing interests. There is a need for a **strategy to influence REF to include DRI impacts**. Similarly, software outputs constituted only 11 of the 185,290 outputs submitted to REF 2021. DRI professionals are more likely to be valued by institutions when they are able to directly contribute to metrics by which institutions are assessed.

Relatedly, despite changing guidance from funders recognising the need to **add DRI staff support and named data professionals to grants**—for example, to ensure that software development is sustainable and reproducible, and that data outputs are curated appropriately—there is an impression that many panels do not give appropriate weight to these concerns. Suggestions in this direction included reviewing the aims and structure of Data Management Plans, stronger

guidance to reviewers and funding panels, and also a concerted effort to include **DRI** professionals **on panels and higher committees (e.g. Science Board)**.

7.2 Access modes for large-scale computing services

Historically, access to high-performance computing services has near-exclusively used the Unix shell via a Secure Shell connection, with workloads being submitted as shell scripts to a batch scheduler such as Slurm, PBS Pro, or Grid Engine. However, this is not the only way by which workloads can be defined and scheduled, and can present a learning curve to those not already familiar with these technologies—this may act as a barrier to new communities making use of services. Additionally, there may be demand to go beyond what the conventional technologies offer by default—for example, defining workloads via data rather than via code, and defining workflows that span multiple clusters and/or workstations. While any change in access mode for new or existing services should be made as a result of **gathering actual requirements** from users and potential users, some alternative options were explored.

Alternatives discussed include a **programmable interface to create jobs**, other non-Kubernetes **text file-based** routes to specifying workloads in a data-defined way, a **GitHub listener** that would allow a push-driven workflow defined similarly to GitHub Actions, and **graphical user interfaces** or web **portals** for common community workflows. Additionally, some communities may be accustomed to **interactive use**, a pattern that is not well-suited to current schedulers in terms of efficient resource utilization. Increasing numbers of facilities in the UK and internationally are making some portion of resources available for interactive use, at the expense of less intensive use of these resources, including through graphical interfaces like Jupyter Notebooks. It was recognized that **no one solution fits all** users, and most systems would likely need to support more than one access route. Each would also require a different level of support; for example, a GUI or web interface setup would require significant support from DRI professionals either within the service delivery team or within the community to deliver something well-adapted to the specific workloads that the community needs to be able to run.

Kubernetes is a container orchestration and high-availability cluster management tool that has become a de-facto standard for deploying complex, highly scalable software solutions to cloud services. Its high scalability and wide adoption, and in particular its enabling of data-defined infrastructure and services (rather than imperative scripted definitions), has led it to be considered in conversation around schedulers for high-performance computing. There is a section of the potential user community who either already make use of Kubernetes (for example, on a commercial cloud) or whose workflows would map well to it, and who would benefit from having this access model on future UK DRI. However, there is not yet a consensus as to what role it could play in this space; participants suggested it may be **too complex** for users, and it does not currently solve some of the problems HPC-specific schedulers target, while making some assumptions that do not necessarily fit the HPC use case.

As discussed in a previous section, **sequential phases…taking turns with massively parallel steps** is another challenge whose solution may benefit from changes to established access modes. Some technologies discussed above may contribute to possible approaches to this challenge, but

allowing these workflows to progress at a reasonable speed while still making efficient use of the available resources is an open challenge.

In addition to high-level frontends, some specific recommendations were discussed that are applicable to most frontend technologies. A common concern in HPC facility provision is how to ensure that the most efficient and effective use is made of the available resources. While on most facilities an interested user is able to obtain data to understand how efficiently their workloads are performing, the people most likely to benefit from this knowledge (with large potential efficiency gains) are also those who are least likely to understand how to interrogate the available data to understand this. Some HPC facilities provide data on a periodic data, or an **efficiency report after every job**, as to what proportion of the allocated resources were used, **to drive better practice and identify problems**; however, this is not currently universal. Providing these data to project leaders in addition to the individual users submitting the jobs would allow projects to manage their utilization more holistically, and allow more experienced users to mentor newcomers who may not otherwise realise there is an opportunity for improvement.

Additionally, sometimes reporting after a job is insufficient to understand where the performance bottlenecks are; better **monitoring of running jobs** would be helpful here. In addition to providing more information, it was mentioned that **training** would be needed to enable users to understand how to make improvements, and **incentives** put in place so users have a reason (beyond good citizenship) to apply the techniques; these are discussed further in other subsections. It is possible that technical solutions could also contribute here: in some cases, not tying users to a single system and instead presenting an abstracted interface may allow the scheduler to pick resources that best match the demands of the job. However, this is likely to require development work both by service providers on improving schedulers' abilities to make these decisions, and by long-standing users to adapt to this new model and learn to inform the scheduler of their workloads' requirements.

Containerisation is increasingly used in many contexts to bypass much of the complexity associated with maintaining compatibility in a complex software stack. For complex software that would otherwise require an intricate installation process needing input from system operators, containers can significantly reduce the workload required from facility staff. However, it was recognised that containers introduce specific challenges in a HPC environment. While reducing the global clutter associated with a large module setup serving a diverse user base (which participants acknowledged as being a challenge on most systems), containers can duplicate large amounts of data for each user making use of them, meaning larger filestores are needed. And as most containerisation technologies are designed around services running on a single (typically virtual) machine, the abstractions they introduce don't necessarily interact well with technologies for multi-node computation, which need to sit very close to the hardware.

Many systems currently allocate nodes in *exclusive* mode, where no other users' jobs are permitted to start on the same node (even if the workload only requests a small fraction of the resources on the node). Since many users' workloads are not able to fully saturate a node's resources, this leaves the unused capacity idle, reducing the overall machine efficiency. While

exclusive mode is valuable for proprietary or confidential data, as **most users don't need exclusive mode**, disabling it by default (with the option to enable when necessary) may improve utilisation of the resource. Users are also frequently not well placed to understand what resources they will need for how long at the point of submitting a proposal; providing **tools to support requirement calculations** would likely increase the quality and utility of these estimates.

The final component of this discussion centred around **Quality of Service** (QoS) provided to, and experienced by, users. Multiple discussions raised the importance of a **debug queue** with very rapid turnaround (but strong constraints on the number, size, and length of jobs permitted), so that a workload can be tested and adapted in a rapid cycle. This will enable better optimisation of workloads, as the optimisation process requires iterative work, which is greatly slowed down when each test needs to wait in a multi-hour queue with production jobs. It was suggested that nodes be specifically reserved for this purpose, with the reservation being "topped up" from the general pool where possible as nodes are allocated, to ensure that a rapid start is always available. Those making use of discretionary allocations should not necessarily be cut out from this partition; if access without a dedicated allocation is given to enable pump-priming, R&D, and prototype projects, then these will particularly need access to a rapid develop-debug cycle to bring their tooling to the point where it can be scaled up in production.

More generally it was suggested that the **perceived QoS** or **responsiveness** [can be] **more important than throughput**—a user submitting a large array of production jobs may prefer one to start immediately at the expense of others finishing slightly later, to ensure that the job starts correctly, rather than all jobs starting simultaneously out of working hours, and immediately failing. There is also the opportunity to quasi-marketise the scheduling algorithm, with users able to trade **flexibility** [in allowing a] **later completion time for a cheaper run**. Poor quality of service can also be a barrier to projects completing on time or using their full allocation, as a project plan may assume running one large workload near-continuously, with breaks only due to limits on the lengths of individual jobs, but periods of heavy utilisation may lead to a delay in job steps starting. A possible resolution to this could be to give each project a dedicated partition or reservation, with other projects able to run lower-priority, preemptible workloads thereon only when it is not in use for its primary project.

In general, it was recommended that **policies** [should be] **user driven**, rather than imposed based on an assumption of what users need or what is convenient for vendors or system managers. Where possible, resources should be provided to enable **local expertise to work with users, understand requirements, and advise how to use/define the environment**. Users are unlikely to be able to become experts on all the systems they gain access to; DRI professionals recruited to understand a facility in detail are in a much better position to understand the most effective ways to make use of the resource and advise users on this.

7.3 Allocation processes for large-scale computing services

DiRAC currently uses a relatively bureaucratic process to allocate the majority of computing time on its facilities, involving a written scientific proposal which is externally peer reviewed, and a

technical assessment of the software to be used which is assessed by RSEs familiar with the DiRAC systems, before being assessed by a Resource Allocation Committee which prioritises proposals. This gives a strong impression of robustness and impartiality, but provides a significant hurdle for new researchers to overcome. Alternative mechanisms used in some other disciplines are block allocations to communities such as the High-End Compute consortia (HECs); directly providing allocations of time as part of grant proposals; and directly awarding time at the Director's discretion (for example, as seedcorn in support of preparing a full proposal).

Each of these places hurdles in front of **bringing new communities on board**; there is an open question of how to **get** [such communities] **over the activation energy to apply**. It was also raised that there is a **recency bias**, where applicants are more likely to **go back to machines they have used previously**, regardless of whether these are the most suitable facilities for the workload at hand at a given moment.

Having unified, **federated access to all Tier 0–2 systems under DRI/UKRI** may help both of these—reducing the number of processes for researchers to become familiar with and easing the process of switching between different resources.

Some projects expressed a need for a **something between seedcorn and full proposal**, where a piece of work is concise and timely, but beyond what can be granted on a discretionary basis. One suggestion that could help with this, as well as lowering the barrier to entry for other communities, would be allowing the **use of low-priority** (possibly pre-emptible) **jobs with short duration** in the absence of a full peer-reviewed proposal, but with full acknowledgement of resources used.

Participants within HECs expressed great satisfaction with the HEC process for resource allocation; however other participants expressed concerns over how scalable this process is, and whether there may be negative implications for transparency and inclusivity. Participants also specified a need to give **fair access to the HPC resources, not favouring specific communities**.

Regarding efficient utilisation of resources, it was suggested that there should be **strong technical oversight banning inefficient codes when others exist**. However, there should also be other mechanisms to drive users to make the most effective and efficient utilisation of the available resources. **Currently we allocate** [resources] **by CUs** (compute units, i.e. proportionally to number of compute elements used and the time for which they are used), which may raise considerations for sustainability. Alternatives may be to allocate by energy usage, or to identify other mechanisms that better reflect the fact that some classes of resource (e.g. large memory nodes) have a higher associated capital and operational cost, and so drive behaviour to use less expensive or more energy-efficient resources where possible.

Finally, in any allocation process it is also important to prevent double-jeopardy; viz. a re-tensioning of science that has already been tensioned by Research Councils.

7.4 Data challenges for computational projects

There was a recognition among participants that **data curation is a challenge**, and is perhaps not as well recognised or considered in some disciplines as the computation-related challenges. Dedicated resources (both capital and human) are likely necessary to address these. The specific challenges that these resources are needed to address can be divided roughly into those associated with making use of data in the context of large HPC facilities, and those associated with long term archiving and availability of data, with a few challenges relating to both areas.

7.4.1 Data within large facilities

Historically, data storage and compute were physically separated within a supercomputer; however, the volumes of data used as input to many current computational research problems means that this creates significant time wastage. As such, **storage needs to be more tightly integrated with compute**. Simply adding storage to compute nodes is not sufficient, however; there needs to be careful thought as to how the software stack can enable the storage to be used effectively, staging data to be in the right place at the right time. This may necessitate **dedicated nodes for data movement**, as well as **data-aware scheduling** and **automated transfer between systems so workflows can start once data has arrived**.

A better awareness of the cost of retaining data is likely needed. It was suggested that the model of providing a storage quota to a project may not be the best way to achieve this; instead, a **charge per terabyte per hour** may better focus decisions on how long retention is valuable for a given dataset. (Of course, this charge may be against a consumable budget similarly to CPU time allocations, rather than a financial transaction.)

The available **external network connection** to a facility can also pose a challenge: in some cases different stages of a computation are better fits for different facilities, but limitations on the available bandwidth between machines (and/or lack of filesystem integration) mean that it is more practical for researchers to use a sub-optimal machine for one or more aspects of a computation to avoid needing to transfer large volumes of data between facilities.

Tiering of data storage, where data migrate (automatically or otherwise) between different categories of storage (in terms of speed, latency, and energy cost of accessing data) was discussed as a possible technique to help both with the above problem and with the high energy cost of keeping large storage arrays powered to provide instant access to large volumes of data which are infrequently used.

7.4.2 Data archiving

Large scale computational and experimental facilities can both produce colossal volumes of data. Retaining all of these data would require resources well beyond what most facilities are able to provide, and in many disciplines the complete volume of generated data is likely to provide no value to other researchers. Therefore it is incumbent on researchers and disciplines to identify methods of thinning and compressing the generated data to those that are useful, and for which the trade-off of paying for long-term storage is worthwhile—for data from some fields of computational research in particular, **recomputing** results can be viewed as an extreme form of

compression, where terabytes of output can be encoded in kilobytes of input file (but where the decompression requires huge computational resources on current facilities).

However, even compressed data can be very large, and there is currently a lack of any unifying approach for where such data can be retained for the periods recommended in UKRI data management plans—for example, there may be a need to **keep data generated under a three-year grant for 15 years**, which isn't achievable with current services and funding models. Many **current-generation portals and similar services are inconvenient ("or worse") at providing access for independent researchers** to data from publicly-funded research, and **a lot of data is at risk without a curation plan**.

As such, the need was identified for a **national data repository**, to be operated pan-research council and pan-discipline, funded as part of the national Digital Research Infrastructure. Data transfer should be via an open protocol such as Globus. This would help to avoid disciplinary silos, and better aid cross-funding-stream data sharing. However, as metadata standards are very discipline-specific, such a service would need to account for this—potentially acting as a backend service to **federated**, **discipline-specific, community-led front-ends** implementing appropriate metadata standards, and potentially with diverse retention policies appropriate to the field, provided that these implemented common protocols for synchronization, access control, etc.

Specific challenges occur with large volumes of archived data that need to be visualized or otherwise reduced in volume—provisioning sufficient local resource to enable such data to be downloaded and processed may be prohibitive, even if the actual computations to be performed are relatively modest. Solutions for **remote visualization, navigation, and interrogation** of large data, as well as APIs to better enable **subsetting of data**, would significantly enhance the utility of large datasets to other researchers and enable them to be leveraged into more research; this would require integrating computation with long term data storage, and developing appropriate access models. It was recognized that currently the technologies to enable this still need to be developed.

Many classes of data have specific requirements that may make them unsuitable for storage in a general-purpose public data repository—for example, export restrictions, data protection/GDPR, or Material Transfer Agreements under which access to underlying data was granted. In particular, **data in Trusted Research Environments and similar would not be allowed to leave those environments**; however, where possible they should still be **interoperable** both with each other and with public repositories, rather than researchers needing to learn multiple technologies and make superfluous conversions to use public and private data together.

It was discussed that many, most, or all **disciplines are falling short of the FAIR data standards**—this may be due to lack of resource, lack of awareness, or lack of appropriate standards. A lack of metadata standardisation in particular hinders both data reuse within disciplines and working across multiple disciplines. In some cases, there are data practices that can hinder methodology innovation—for example, in machine learning, it is common to discard intermediary checkpoints

in model training that could be used to better understand and improve models and training algorithms.

7.4.3 Archiving data from large facilities

Some challenges and possible solutions fall into both of the above categories. Strategies for long term retention and curation need to be integrated with those for computation, so as not to cause one problem while solving another. Providing APIs for access to tape storage and making better connections to metadata silos and national data storage facilities can help enable joined-up thinking in this space.

7.5 Training

Provision of training for users and potential users of DRI was a common thread in many discussions despite not being specified as the topic for any individual discussion on the agenda. The outcomes of these discussions will be summarised here.

Nobody is born knowing how to use digital research infrastructure, and so at some point each user must receive training on how to effectively use the infrastructure needed for their research. In communities with an established history of using DRI, this frequently takes the form of mentoring of PhD students by more experienced researchers; this approach however has a number of disadvantages. Firstly, it does not adapt quickly to changes in technology; methods and patterns of working that are no longer efficient or appropriate may be passed down and repeated. Secondly, it is exclusionary of research groups and entire research communities that for whatever reason have historically not used DRI, even if they may now be able to get significant benefits from doing so. As such, it is essential to provide an effective training programme both for new and existing users of DRI, and to ensure that users take advantage of this when needed.

Some topics suggested are specific to individual services—such as using the Slurm scheduler, and the design of a particular system. Others are specific to particular needs—for example, not every application makes use of CMake, but training on it is useful for researcher using or developing applications that rely on it. A third group are applicable more widely—for instance, making use of a debugger.

It is necessary to incentivise researchers to take up training opportunities. For researchers already making use of the DRI, the need for training can be identified via monitoring of the efficiency with which they use the resources they request, and participation in training could be encouraged by imposing more restrictive quotas on usage (up to and including suspending access) until the user engages. (To minimise the friction, both to the user and to the service provider, participation in training may be tracked via the same software—e.g. SAFE—that is used to manage the user database for the DRI in question.) Care is needed to ensure that the training is seen as supportive in achieving scientific goals more effectively, rather than as a punishment for misbehaviour. Monitoring could also help identify whether the system a researcher is using is the best fit for their workloads, or whether other available resources would give them better throughput.

For researchers not already familiar with high-end computing, it was proposed to **link ECR fellowships with training and DRI support, prioritising emerging fields**, as well as introducing an **interdisciplinary doctoral training programme** in DRI, and **cross-disciplinary schemes and training**.

8 What unknowns are there, and how can they be better understood?

8.1 What don't we know?

- What are the emerging hardwares? Is the need for CPU decreasing everywhere, or will some science applications always need them? How large a memory will be available?
- What will architectures look like in 5 years? How do we balance tried and tested architectures and solutions against emerging technologies?
- What software stacks will be available? Will core libraries work at scale? At Exascale? eg. FFTW, HDF5, MPI.
- What are the emerging algorithms eg. AI? What programming models will prevail? How much storage will be required?
- Problems exacerbated by the diversity of codes, even within a given science area -- need to nest and benchmark variety of codes on various advanced architectures.
- What back-system environments, in terms of hardware and management software, are required for HTC?
- Is it even possible to have a top-down approach? Should we rather focus on bottom-up funding streams instead and let the communities bid for it?
- A key resource is the people/RSEs to implement this based on strategy and long-term plan.
- Knowledge exchange is very important. Communities will need adequate RSE support.
- Who are the competitors nationally and internationally?
    - Do we want to compete with Amazon and Google in some areas (AI)? We need a strong case why universities should not migrate over to these companies.
    - Do we want to benchmark our systems against European and US machines?
- Will UK join EuroHPC? How will this inform design of future UK systems?
- How should we ensure that all potential DRI user communities can have appropriate input into its design, including modelling and simulation, AI, experimental communities and others.
- STFC has some policies and experience on data curation and storage. But it is not clear how universities implement this.

8.2 How can we find out?

- Talking to vendors
- Horizon scanning in research councils (priority areas) should clarify what kind of computing problems/support they see.
- Providing access to small-scale testing facilities integrated within the large-scale DRI, permitting researchers to test and develop codes and workflows against emerging technologies

- Providing benchmarking facilities for a more formalised evaluation of emerging technologies

9 Are there gaps in current services that can be quickly filled?

9.1 Hardware

- Regular augmentation/refreshment of GPU hardware. AMD GPUs don't yet have enough library support to become the default choice.
- Memory (RAM): 20TB per node available on Superdome Flex (shared memory machine). CFD applications can't split the mesh.
- Composable RAM systems.
- Capability to virtualize clusters to guarantee when a job will run at the cost of under-utilised processors.
- It is very difficult to catch up with HPC evolution.
    - AMD, NVIDIA and Intel all present us with hardware solutions, and then we try to find out how to handle these; closer collaboration would help.
    - We permanently fix codes, just to find out that the hardware evolution changes the character of the challenge a few months later; vendors would have known that before and could have guided development (co-design).

9.2 Software

- What do we do about support and long-term maintenance? Need to recognise that porting to GPU does not usually replace CPU, it typically adds extra code paths, algorithms etc. and these add to the software's computational complexity, technical debt, etc. How do we support this?
- Software sustainability and maintaining portability to upcoming novel architectures are both important issues.
- How can we support lower-level software development, e.g. library development, so that it is long-lived, developed actively, updated, and ported to new hardware?
- Need parallelizable software versions of eg. Ansys and similar codes.
- need a way to capture R&D in software development as a successful component of the project, even if it wasn't a successful method to do the science, because it was a correct, useful research output.
- eCSE-style, light-touch collaborative software development projects.
- Determine best practice for key kernels used across multiple domains (e.g. FFTs on large, parallel GPU machines).

9.3 Systems & Access

- Current codes on Archer or any Tier 2 system are performance-limited by the file system.
- Dedicated full-system benchmarking eg. "petascale hackathon"
    - 1 week workshop permitting sole access to whole system;

    - Vendors, sysadmins, RSEs, users all participate;
    - Annual event on existing systems, and in commissioning of new systems;
    - Need to stress filesystem too.
- Some smaller systems where people can prototype and test.
- Rolling calls with simpler application process.

9.4 Data

- Data curation front end for public access to data.
- Spare nodes for data transfer between file systems.
- Connect everything to Globus (service for file transfer/syncing)
    - It's free of charge, but possible security implications for ISO institutions.
- Improved support for PB-scale data intensive workloads – network to get it on/off the nodes & machines; storage itself – local vs network storage? What about GPUs, are we generating data faster than we can get them to storage?

9.5 Training & Community

- Need to carefully manage manpower cost of the projects too, so that if someone spends 2 years porting a code to GPUs, their CV/track record correctly reflects their productivity. How do we support the relevant staff in their careers, what technical/research outputs etc are appropriate?
- National leadership setting standards/guidance for open research best practice - that scales as people develop their practices, while ensuring a low entry-level requirement.
- More widespread knowledge about testbeds of ExCALIBUR.
- Pilots and case studies. People would like to read more about success stories for their grant proposals, bids, but also strategic decisions.

10 Conclusions

It remains to thank workshop participants for their willingness to rise to the challenge of an unusual exercise, and for generously sharing their time and their knowledge (both precious commodities); and finally for providing invaluable feedback for early drafts of this report. We'd particularly like to record our appreciation of and gratitude for the expertise, support and goodwill of the four rapporteurs, who threw themselves into the fray with little or no questions asked, and whose energy and hard work were essential to the workshop's success:

- Vassil Alexandrov
- Alastair Basden
- Phil Hasnip
- Tobias Weinzierl

Needless to say, any remaining inaccuracies are entirely our responsibility.

Ed Bennett (STFC RSE Fellow)
Simon Hands (Community Development Director, DiRAC)
February 2024

Appendix A

Glossary

AI	Artificial Intelligence
AMD	Advanced Micro Devices; a provider of CPUs and GPUs
Amazon	Large multinational company that in addition to online shops also operates commercial public cloud infrastructure
API	Application Programming Interface
Arm	Formerly Advanced RISC Machine; a provider of standards and reference designs for microprocessors, and by extension microprocessors using these.
CCP	Collaborative Computational Projects
CDT	Centre for Doctoral Training
CFD	Computational Fluid Dynamics
ChatGPT	Commercial AI service that provides text responses to prompts using the GPT (Generative Pre-trained Transformer) 3.5 and 4 models
Cloud	Computational resources available on-demand, without direct active management by the user. Typically involves large amounts of homogenous resources, which a user can self-allocate and deallocate portions. Can be hosted within/dedicated to a single organization (private cloud) or hosted commercially with general availability (public cloud).
CP2K	Open source molecular dynamics software
CPU	Central Processing Unit; typically the primary, all-purpose computational unit of a device, featuring a relatively small number of relatively powerful cores.
CRYSTAL	A computational tool for solid state chemistry and physics
CSD3	Cambridge Service for Data-Driven Discovery
CU	Compute Unit
CV	Curriculum Vitae
CUDA	Compute Unified Device Architecture
DESI	Dark Energy Spectroscopic Instrument
DiRAC	Distributed Research utilizing Advanced Computing
DRI	Digital Research Infrastructure
ECR	Early Career Researcher
eCSE	Embedded Computational Software Engineer
EuroHPC	The European High-Performance Computing Joint Undertaking
Exascale	Computing at a scale involving $O(10^{18})$ or $O(2^{60})$ bytes of data or arithmetic operations per second.
FAIR	Findable, Accessible, Interoperable, and Reusable
FFT	Fast Fourier Transforms
FFTW	Fastest Fourier Transform in the West
FTE	Full-Time Equivalent
Front-end	The part of a service with which a user directly interacts
Globus	A company providing solutions for grid computing, including an implementation of the GridFTP (Grid File Transfer Protocol) standard for transferring large volumes of data between HPC facilities

Google	Large multinational company that, in addition to a search engine and other online services, also operates commercial public cloud infrastructure
Gordon Bell Award	Annual prize awarded by the Association for Computing Machinery recognising outstanding achievement in HPC
Grace Hopper	New architecture from NVIDIA that tightly couples an Arm CPU (Grace) and NVIDIA GPU (Hopper, the same architecture as the H100) on the same chip.
GridPP	Distributed computing facility for data-intensive research, based in the UK and CERN
GPU	Graphics Processing Unit; typically employed as an accelerator, having a relatively large number of relatively simple computational cores. Very performant and power-efficient for specific classes of large-volume numerical computation.
HDF5	Hierarchical Data Format 5
HEC	High-End Compute consortium
HIP	Heterogeneous Interface for Portability
HPC	High-Performance Computing, typically involving tasks each of which requires substantial, tightly-coupled computational resource
HR	Human Resources
HTC	High-Throughput Computing, typically involving large numbers of tasks each requiring modest resources, and with little or no coupling between them.
IO	Input/output
IRIS	Digital research infrastructure supporting STFC science
ISO	International Organization for Standardization. May also be used to refer to the ISO/IEC 27001 standard for information security.
ITER	International Thermonuclear Experimental Reactor
LAMMPS	Large-scale Atomic/Molecular Massively Parallel Simulator
LHC	Large Hadron Collider, a particle accelerator at CERN.
LLM	Large Language Model
LSST	Legacy Survey of Space and Time [or Large Synoptic Survey Telescope, former name of the Vera C. Rubin Observatory where the survey is conducted]
ML	Machine Learning
MPI	Message Passing Interface
NVIDIA	Provider of GPU hardware solutions
P100, V100, A100, H100	Successive generations of NVIDIA GPU architecture (released 2016, 2017, 2020, 2022 respectively)
PB	Petabytes
PBS	Portable Batch System
QoS	Quality of Service
R&D	Research and Development
RAM	Random Access Memory
REF	Research Excellence Framework
RISC	Reduced Instruction Set Computing
RSE	Research Software Engineer

SAFE	[HPC account and project management system developed by the University of Edinburgh]
SIESTA	Spanish Initiative for Electronic Simulations with Thousands of Atoms
STEP	Spherical Tokamak for Energy Production
STFC	Science and Technology Facilities Council
SuperDome Flex	Line of large-memory high-performance servers from HPE (Hewlett Packard Enterprise)
TB	Terabytes
UKAEA	United Kingdom Atomic Energy Authority
UKRI	United Kingdom Research and Innovation
USP	Unique Selling Point
VASP	Vienna Ab initio Simulation Package

Appendix B

List of attendees

Cosmology

Matthieu Schaller, University Lecturer, Leiden University

Plasma & Fusion Science

Shaun DeWitt, Head of High-Performance Data Analytics, UKAEA
Andrew Davies, Advanced Engineering Simulation Lead, UKAEA

Computational Biology

Crispin Miller, Head of Computational Biology, Cancer Research UK Scotland Institute
Xiao Xue, Research Fellow, University College London

Computational Chemistry

Charles Laughton, Professor of Computational Pharmaceutical Science, University of Nottingham

Engineering

Benedict Rogers, Professor of Computational Hydrodynamics, University of Manchester
Stefano Rolfo, Principal Computational Scientist, STFC Daresbury Laboratory
Garth Wells, Hibbitt Professor of Solid Mechanics, University of Cambridge

Large Language Models

Pontus Stenetorp, Professor in Natural Language Processing, University College London

Materials Science

Alin-Marin Elena, Computational Scientist, STFC Daresbury Laboratory
Dimitar Pashov, Research Associate, King's College London
Scott Woodley, Professor of Computational Chemistry and Physics, University College London

Particle Physics

Jon Hays, Director of Particle Physics Research Centre, Queen Mary University of London, and Scientific Director of IRIS, representing workloads across astro, nuclear and particle physics.

Rapporteurs

Vassil Alexandrov, Chief Science Officer, STFC Hartree Centre
Alastair Basden, DiRAC Senior Technical Manager, Durham University
Phil Hasnip, EPSRC Research Software Engineering Fellow, University of York
Tobias Weinzierl, Professor in the Department of Computer Science, Durham University

DiRAC

Ed Bennett, STFC Research Software Engineering Fellow, Swansea University
Simon Burbidge, DiRAC Research Software Engineering Team Lead, University College London
Simon Hands, DiRAC Community Development Director, University of Liverpool
Clare Jenner, DiRAC Deputy Director, University College London
Mark Wilkinson, DiRAC Director, University of Leicester

UK Research and Innovation

Ben Yarnall, Medical Research Council
Luke Davis, Engineering and Physical Sciences Research Council

Appendix C

Questionnaire

Name:

Affiliation:

[**Delete as appropriate**] I am responding about my personal research / I am responding on behalf of
< *Insert name of project/facility/community/collaboration* >

Research Area *(Title***):**

1. Brief description of science/research goals for non-specialist audience:

2. Description of workflow(s) associated with this work with ***quantitative*** details on system and service requirements as set out below *(ie. not just "lots of cores/high RAM").*

- RAM
- CPU cores
- Interconnect bandwidth/latency
- Storage and I/O
- Accelerators,
  e.g. GPUs
- Enabling software,
  e.g. parallel and numerical libraries
- Support required,
  e.g. *Research Software Engineering, Data Curation*
- Any hard dependencies,
  e.g. workflow is reliant on hand crafted x86 assembly language

There's no need to specify particular hardware solutions – we are focusing on high-level requirements at this stage

3. How do your workflows map to numerical methods?
Choose as many from the list below as are relevant.

- **Dense linear algebra** - dense matrix and vector operations,
  e.g. BLAS, MATLAB

- **Sparse linear algebra** - solves the same problem as dense linear algebra but has matrices with few non-zero entries,
  e.g. SpMV, OSKI, or SuperLU
- **Spectral methods** - transform data from/to either a spatial or temporal domain,
  e.g. FFT
- **N-body methods** - interactions between many discrete points,
  e.g., Barnes-Hut, Fast Multipole Method
- **Structured grids** - organise data in a regular multidimensional grid,
  e.g. Cactus or Lattice-Boltzmann Magneto-hydrodynamics
- **Unstructured grids** - possess data structures,
  e.g. linked list of pointers, ABAQUS, FIDMAP
- **MapReduce** - captures the repeated independent execution of a "map" function and results are aggregated at the end via a "reduce" function,
  e.g. Monte Carlo
- **Combinational logic** - exploits bit-level parallelism in order to achieve high throughput
- **Graph traversal** - visits and evaluates a number of objects in a graph,
  e.g. Quicksort
- **Dynamic programming** - solves a complex problem by solving a series of simpler subproblems
- **Backtrack & branch-and-bound** - approaches generally search a very large search space to find a globally optimal solution
- **Graphical models** - map variables into nodes and conditional probabilities into edges,
  e.g., Bayesian networks and Hidden Markov Models
- **Finite state machines** - capture a system whose behaviour is defined by states, transitions defined by inputs and the current state, and events associated with transitions or states

For further discussion of these various models or "dwarves" you might find it helpful to consult https://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf]

4. Software requirements – does this activity have sufficient effort available to future anticipated computing resources efficiently? If not, how much additional effort would be required?
Specify e.g. #PDRAs #PhDs #RSEs

5. Any other requirements that you would like to highlight?

6. Based on your knowledge of the technology landscape, are there particular architectures particularly relevant for your workflow?
e.g. GPU, CPU, flash-accelerated storage