Extreme Scaling Workflows in Scientific Computing

Background and Methodology

This report describes Work Package 1.5 of the DiRAC Federation Project, whose goal is engagement with computational scientists in other fields (biomedical, materials, climate, environmental, energy) to share and exchange insight and experience, and in particular to explore the extent to which common workflows can be the basis of defining UKRI-wide computing services in a future UKRI Digital Research Infrastructure. Following consultation with senior figures within DiRAC, individual scientists in these domains were approached on the basis of their expertise and/or standing, and then invited to one of two one-day workshops held in central London during February 2022. This report covers the second workshop, held on 23rd February, focussing on "extreme scaling workflows'', by which is signified problems requiring tightly-coupled systems of CPUs/GPUs, exemplified by lattice QCD, currently served by the DiRAC3 *Tursa* facility at Edinburgh.

Due to the prevalent COVID situation, not everyone was able to attend the meeting in person, so the event was designed to run in hybrid mode, with remote participants attending via Zoom, and questions for speakers submitted via a web-based interface.

In advance of the workshop, participants within a particular discipline were organised into groups and invited to collaborate on presenting and summarising the state-of-the-art in their particular field in a specified timeslot during the morning of the workshop, prompted by the question "what kind of machine is ideally suited to your problem?". The participants and groupings are listed in Appendix A. Stress was laid on the importance of explaining the problems being tackled to a non-expert, if informed and receptive, audience, stressing computational techniques, challenges and bottlenecks. Scientific quality, competitiveness and importance was understood; speakers were urged not to spend precious time underscoring those aspects. Presentation slides are available to view at www.dirac.ac.uk/federation-project-workshops-wp1-5/.

During the afternoon, participants broke up into groups to consider and discuss particular issues of interest. In the first half of the afternoon the questions were scripted, but participants were given the opportunity to propose their own questions, and the most popular (on the basis of a web-poll) were discussed in the second half of the afternoon, with participants free to join whichever group best matched their interests. All questions considered are listed in Appendix B. Responses were collected, again using a web-based tool, and form the basis for the discussion summary in the second part of this report.

Scientific Fields - State of Play

Lattice QCD

- Lattice QCD addresses STFC Grand Challenges concerning the interaction of fundamental constituents of matter: applications include high precision hadronic physics, strong dynamics Beyond the Standard Model, and holographic cosmology.
- The Euclidean path integral underlying local quantum field theory is formulated on a spacetime lattice and then importance sampled using Monte Carlo Markov Chain methods. A typical system size is 64^4 implying e.g. a quark field has O(500M) real components. In realistic problems the compute cost is dominated by solution of the discretised Dirac equation, requiring inversion of a large, sparse, and poorly-conditioned matrix. Pre-conditioning techniques such as eigenvalue deflation (which potentially require O(10TB) data volumes) or multi-grid are by now mandatory.
- A specialist data parallel C++11 library called *Grid* has been developed to write efficient LQCD code on extreme scaling machines. It permits platform-agnostic code, and implements many

distinct different fermion actions and sparse solvers as part of a HMC/RHMC application programming interface. Grid works within a Single Instruction Multiple Thread (SIMT) paradigm, (permitted by the translational invariant nature of the problem) exploiting a cartesian lattice layout, high-efficiency halo exchange, a shared buffer and multi-endpoint communications. Grid admits high-level expressions with site-wise operations parallelised automatically, with loops over sites multi-threaded or offloaded to GPUs. The resulting code exhibits excellent scaling and was used as a benchmark in procurement of the ES service at Edinburgh.

- Current LQCD workflows face the challenge of large quantities of heterogeneous data and multi-level requirements (production, post-processing, data analysis...), presenting software challenges in modular automation, cataloguing, and fast data access. A workflow management system called *Hadrons,* built around the Grid library, has been developed to automate the necessary computations with optimal memory footprint. It is anticipated the need for such automation will increase as LQCD projects increase in scale and complexity. There are also many challenges ahead for curating and sharing increasingly large lattice datasets.

Chemistry, Biology and Materials Science

- Computational materials science applies distinct approaches according to the relevant length and time scales of the problem, ranging from "exact" quantum-based methods at few nm scales (post Hartee-Fock, CI, GW, coupled-cluster, Quantum Monte Carlo...); through Hartree-Fock and density functional theory; methods employing semi-empirical and machine learnt interaction potentials; classical molecular dynamics (MD); up to coarse-grained and continuum methods at micron scales.
- Many applications employ specialised workflow schedulers and management tools such as Aiida, Fireworks, more generic tools like Apache Airflow, and others to process many thousands of compounds in a single large run or create complex workflows (from geometry to spectroscopy). Large memory, I/O and compute requirements are bottlenecks. One solution involved porting key routines to accelerators (NVIDIA GPUs) — a recent electronic structure calculation of InAs nanowire on Al substrate achieved 90% of peak performance on Juwels Booster (using Questaal code developed at KCL and part of CCP9). Packages such as ChemShell combine quantum mechanical and molecular mechanical calculations; here better performance arises if the available processors are divided between multiple simultaneous energy/gradient calculations. In addition to the scientific challenge resulting from coupling, such approaches have the additional complication of coupling methods with very different complexity behaviour (see report on MI workshop for discussion of complexity of methods in Materials Science).
- One approach to “Extreme Scaling” is to use an entire HPC facility for a fixed duration, with embarrassingly parallel workloads enabling hundreds of drug-protein MD simulations, each performed on a single GPU using a suitably-designed binding affinity calculator workflow and task-farming. It is found that system maturity, software settings, hardware variations and user/operator experience all impact performance when migrating between systems, eg. for an equivalent core count ARCHER2 currently takes 6 times longer to run the HemeLB initialisation step than SuperMUC-NG.
- STFC’s Computational Science Centre for Research Communities (CoSeC) has a key role in assuring software developments and access to expert advice for the materials community via various Collaborative Computational Projects such as CCP9, CCP5, MCC and UKCP, each tackling different time and length scales.
- HPC use in biomolecular science as measured by, e.g. ARCHER usage is disproportionately low compared to e.g. chemistry. This must be addressed by incorporating suitable training into undergraduate curricula, to establish a pipeline of users and applications from Tier 5 up to Tier 0 and exascale.

BioMolecular Modelling

- This involves MD simulation of large molecules such as proteins, DNA, and thereby links fundamental principles of physics and chemistry with emergent behaviour of large complex systems of soft matter, with applications in health and biomedical research. Interesting biological phenomena occur over timescales ranging from fs to s. A recent Gordon Bell Special Prize was awarded for simulations of SARS-CoV-2.
- Biomolecular research needs very large, well-coupled, flexible, heterogeneous architectures. Conventional MD does not scale well on CPUs (strong scaling poor, weak scaling better). Porting to GPUs has yielded speedups of 50—100×. However, "slow" events on macroscopic timescales in practice correspond to "rare" events on the microscale, so that large ensembles of short-time simulations (Markov State Models) can achieve the desired results, ie. capacity is equivalent to capability. Multiscale simulations, in which subsystems are modelled at different resolutions, are increasingly important, e.g. QM/MM studies of enzyme reaction mechanisms, or MM/FE studies of how molecular motors transport payloads within cells (with MM on GPUs communicating with FE on CPUs at every timestep).
- Specialist workflows are needed for biomolecular simulations featuring large scale gather-scatter and interleaved simulation and analysis steps, with the various steps often having very different hardware requirements. There is a range of robust, flexible, workflow tools under development, but nothing yet that could be called "industry standard". Very large, well-coupled, flexible, heterogeneous architectures are exactly what cutting-edge, impactful, biomolecular simulation research needs.
- The HECBioSim consortium allocates time on UK Tier1/Tier 2 facilities (which until recently has offered limited GPU availability), supports software development, and supports new HPC users. Important international consortia are BioExcel (EU), CompBioMed (EU) and MolSSI (US).

Climate and Meteorology

- Global weather prediction including extreme events has huge economic importance; annual cost of natural catastrophes ~O(US10^{11}). The computational approach is based on well-understood atmospheric physics applied on a spatial grid, with timestep constrained by the Courant condition. Costs associated with computation, memory, storage and power rise in a predictable way with increasing model resolution. In the future, the aim is to increase coupling with the ocean. Ensemble forecasting, needed to counter the system's chaotic nature, is by now standard. Reliability of long-term forecasting is improving by roughly one day per decade.
- Earth system modelling exploits many HPC paradigms: both shared-memory and message-passing parallelism, vectorization, memory hierarchies, synchronous and asynchronous I/O, and use of optimised libraries. Recent trends include use of domain specific languages (e.g. PSyclone, GridTools, Atlas), reduced precision, porting to GPUs and use of Machine Learning for emulators, data analysis, data compression and feature identification. Weather and climate models have low arithmetic intensity and are limited by off-chip memory bandwidth.
- Typical simulations currently require O(10Pflop) facilities, e.g. ECMWF runs 50-member ensembles @ 9km requiring >1Pflop sustained. Deterministic simulations @2.5km are also employed. A future need for exascale is anticipated.
- Operational workflow for weather forecasting is complex and time-critical: data-acquisition (satellite observations) → assimilation into models → forecast production → generation of output products → dissemination of products → long-term archiving → user-specific data-driven analytics.
- Climate modelling has distinct requirements, with simulations spanning 10^2—10^3 years. Models incorporating clouds, circulation, climate-sensitive feedback, and 1km resolution will require an order of magnitude increase in resource. Multi-model comparison is needed to assess the impact of hazards due to extreme events. Bottlenecks include storage availability and slow I/O. Digital Twins is a new methodology employing Machine Learning as a means of reducing the

excessive data-writing required by multi-model intercomparison. Power consumption is an increasing concern; a conventional “cloud-resolving” simulation costs about 3000 MWh/simulated year, roughly the energy used by 900 households in one year. Novel energy-efficient simulations can bring this down by an order of magnitude.
- The current goals of the physical climate modelling community require an international, coordinated approach. We are currently a factor 100 short of the required machine performance and must reduce our power footprint very substantially. We need to motivate and educate a new cohort of researchers to better bridge the science and technology expertise required to tackle these daunting challenges.

Plasma and Fusion Science

- The simulation of power generation is a multi-scale problem ranging from atomic/nuclear scales up to tokamak systems of O(10m). The current ITER experiment is a O(US10^{10}) project. Relevant science includes plasma physics, thermodynamics, neutron transport, fluid mechanics, cryogenics, materials science. Representative fusion-specific problems include use of DFT and spin lattice dynamics to model the variation of response of magnetic dynamics in structural steels to the field from magnets and the plasma at high temperature; a typical petascale computation involves 10^6 atoms, but exascale resources are required for the 10^9 systems needed to model magnetic domains. Another example is the study of crystal defects resulting from intense neutron irradiation, modelling atomic collisions using empirical many-body potentials; O(10^4) such simulations are needed to inform FEM-based models for component-scale prediction.
- The CAD diagrams required to model the reactor are huge: e.g, the ITER Ion Cyclotron Resonant Heating (ICRH) Port Plug has 78 million elements. Modelling the entire facility at scale would require 3EB memory and 5×10^6 CPUs. For now the focus is on the key components: blanket, divertor, selected magnets, optimisations, lifetime predictions.
- The direction of travel: increasing use of GPUs and high bandwith memory; algorithms implementing parallelism in time (Cf. Markov State Models); digital twinning; surrogate modelling (ML).

Discussion Topics

Common Features of Workflows

Workflows represented at the workshop were relatively bimodal with regards to how reliant on communications they are: some are trivially parallel (sometimes called “embarrassingly parallel”) with very little coupling between computational tasks, allowing them to scale out to an arbitrary number of nodes without stressing even a modest interconnect. Others require significant coupling between computational elements, and so the number of computational elements (e.g. GPUs) that can be scaled to depends on the communications fabric between them, as well as the problem geometry. Lattice quantum field theory, the use case defining the DiRAC Extreme Scaling Service, falls into the latter category. Machines specified for one use case will not be an ideal fit for the other, so greater precision in this definition will be important when defining services, and when organising future workshops.

Common to many of the communities represented in the discussion was the need to understand the software’s balance of demands on raw compute, memory latency, and memory bandwidth, to fully understand its performance. Modern HPC systems have increasing complexity in their transport layers (e.g. PCI Express, NVLink, UCX, Infiniband); it was recognised as being important to have libraries to abstract some of this complexity

away, but it currently is not clear how to do this in a way applicable to many research domains.

Another common issue was around coupling of different pieces of software from different scientific domains. This will have impact on areas such as digital twins, where many different effects must be simulated to give a good representation of the real world.

Participants identified a need for more powerful job and task schedulers, to tackle issues including tracking provenance and metadata around data production, to better place individual tasks on (or even move them between) resources depending on their compute demands, and to deal with very large numbers of tasks being run in a massively parallel way.

Access to technical expertise

The main agreement of this discussion was that there needs to be greater possibilities for technical team members to work on problems for the long term, e.g. 3–5 years. RSEs switching contexts (between different projects within a position on fractional FTE, or between different contracts when taking short-term positions) is a hidden cost to productivity. Options suggested to enable this are a larger number of RSE fellowships, more career pathways, and opportunities for permanent RSE positions. It was also observed that the skillset required to enable software to use exascale resources is different to the “typical” RSE profile, and that domain-specific RSEs are likely to become important in this space.

Data

Participants identified the need to be precise about the distinction between data management (e.g. FAIR data), data retention, data exhibition (e.g. open data), and data curation (ensuring that data remain readable on long timescales). Each of these is a specific skillset, and each project must decide which elements are required.

A common concern was resources for long-term storage—institutional and project funding are poor fits to fund storage of data from national facilities for periods longer than a particular grant.

Other questions raised where what to store and for how long. For fields that are purely simulation-based, in principle storing only the input parameters and software used would be sufficient to reproduce all outputs; this would translate a storage requirement for the producer into a compute requirement for the consumer. This balance is affected by the growth of computational capability—what required national facilities to generate a decade ago can now be regenerated with relatively modest effort. Conversely where multiple researchers require results for the same simulation period in a short period of time, retaining the data and adjusting the scheduler so that repeated runs will not execute (and instead will return the data from the previous run) can give better throughput with fixed computational resources.

Participants agreed that the retention times, definitions of what should be retained, and similar questions will depend strongly on the field, and cannot be set at a UKRI level; each discipline needs to define the standards that it can work by.

GPUs

While GPUs form the bulk of compute on the DiRAC Extreme Scaling service, and many of those in the room were making heavy use of GPUs, not all workflows are yet running there. For those that are, the transition has in some cases depended on waiting for others to develop and release GPU versions of their software, as the research community is a consumer rather than a developer of software. For some of these, the bulk of computation wall time is now on data analysis and post-processing; there is an opportunity to improve productivity by porting these workflows to GPUs as well, but how to do so isn't fully understood. Concerns were raised around the portability, as NVIDIA CUDA is the most performant target for GPU ports, but is also proprietary to NVIDIA devices; Chronos SYCL and Intel oneAPI are alternatives that are in principle more portable. NVIDIA on its side is also pushing for standard language parallelism, to allow easier performance portability and porting of existing software to GPUs. While more recent GPUs have moved towards enabling more general-purpose computations, they are still not a universal panacea—the first step of any port should be verification that there is the potential to see performance gains.

Cloud

One use case discussed for cloud services was access for benchmarking of new architectures—cloud providers generally have a wider variety of platforms than is available through traditional HPC routes, and access is available on demand for very short periods. Outside of benchmarking, the ability to test new platforms with very small investment is an opportunity that is not available through more traditional routes. Another use case was for large arrays of loosely- or uncoupled problems, where cloud-oriented tools such as Kubernetes can scale to very large numbers of job-steps better than traditional schedulers, and the problem will easily scale to as many nodes as the cloud provider can make available, giving very good speedups.

The point was made that having a unified interface to computational resources reduces the overhead of learning and adapting tooling for new clusters, each of which has a subtly different operating environment. The counterpoint, however, was that moving between commercial cloud providers (e.g. after changing institution) can be correspondingly harder, since each provider provides its own proprietary tooling. A related issue relates to sharing of software; projects that develop a lot of robust tooling that could be deployed by others, end up not being adopted because potential users either do not have budget to pay a commercial cloud provider (even if they have local computational resources they could run on), or cannot find a route to access that is compliant with institutional procurement and finance rules. There are significant concerns about the cost of data storage and data movement, as these costs and can be both surprising and substantial if not planned for—e.g. if costs are estimated purely on the basis of core-hours as is sometimes done for HPC services. Cloud resources could in principle be a good fit for analysis of data products coming off a homogenous HPC system, where the analysis doesn't map well onto the system's architecture (e.g. post-processing requiring GPUs with data from a CPU-only machine, or serial post-processing requiring large memory with data from a machine with many cores but limited memory per node); however, the data movement problems mean that it becomes more cost-effective to store and analyse the data *in situ*.

Exascale

A number of participating communities already have software that is approaching exascale-readiness. Some areas of concern remained. One was I/O, where asynchronous parallelism can avoid significant loss of performance, but most users are unaware of it; it was identified that performant middleware can remove the effort required from developers of individual pieces of software to benefit from this. Another was resilience—as numbers of computational units increase, the likelihood of one failing during a computation of non-negligible length increases, and so software must be resilient against this. This is discussed less than in previous years, but if it is still a concern on exascale machines then a huge software effort would be needed to adapt code that is near-universally based on an assumption that lines of code will successfully execute as written. Other areas touched on as needing attention to enable efficient exascale workflows included containerisation, workflow management, data cataloguing, continuous integration/deployment, and hardware abstraction.

Quantum computing

Participants generally recognised this to be a substantial distance from being deployable. Some research areas are quantum in nature, so there is proof-of-concept work ongoing with quantum simulators to see how they could perform on a quantum computer; the unavailability of hardware (in particular hardware that solves the problem of decoherence of qubits) is the primary barrier. Other areas could benefit if the core libraries they depend on were able to be quantumly accelerated (e.g. if there were a “quantum FFTW”), but otherwise don’t see quantum on their roadmap. The materials community, while it does not envisage QC replacing classical computers in the immediate future (next ~10—15 years), is actively involved with the QC community (via CCP-QC) in exploring possible algorithms mainly related to optimisation. The current consensus here is that QC will be an accelerator rather than a replacement of classical computing.

Storage

Most fields have substantial storage requirements; some of these are for data to be processed, others data generated. The storage-related concerns can be separated into capacity and performance. An argument was made that I/O performance should be benchmarked as part of the peer review process alongside other performance criteria; this is, however, particularly challenging due to the strong dependence of I/O performance on the mix of workloads running on a system. There are options that will help with both constraints; for example, compression would reduce the volume of data being stored, and also the amount needing to be read from or written to disk in a given computation. It was suggested that there could be compression algorithms specific to scientific computation that may perform better than general-purpose ones. Another suggestion raised was to separate concerns, by using well-optimised libraries for I/O rather than implementing routines from scratch.

Resource allocation

Most HPC resources have more demand for their time than they have capacity; as such, a process for prioritisation of access is needed. There is a question of how to define efficiency for the purposes of prioritisation. Measuring “science per unit compute time” is most likely to match the priorities of funders, but “amount of science” is relatively unquantifiable, and researchers are less pressed to make efficient use of the machine in terms of the parallel efficiency of the algorithm and implementation. It was proposed that allocations could be made (and efficiency measured) in terms of units of energy (e.g.

science per joule—notwithstanding the difficulties with this metric discussed above—or FLOP/s per watt) instead of time. It was suggested that peer review panels did not necessarily need to be discipline-specific.

Software sustainability

It was recognised that software should ideally be modular, small, reusable, replaceable, easy to find, and easy to install, as well as performant. Routes discussed to this include registries such as Pip, Anaconda, and Spack—but with space for improvement in these. The possibility was raised that there is such a thing as too small, as more collaborating institutions increases the "elephant factor"—the likelihood that one will recognise and highlight a glaring problem (the elephant in the room).

Since libraries will interface with each other via an API, there is a tension between providing a maximally compatible API with flat interfaces—for example, that accepts blocks of floating-point numbers of some precision—and providing a more structured interface with more metadata to assist users and tooling in ensuring that the library is used correctly. A proposed solution to this was to have a minimal, performant library (for example, in C++20) maintained by RSEs or other experts, and then a set of wrappers (for example, Python with or without JAX, Julia, etc.) that present a more developer-friendly interface.

There is a frequent trend to reinvent the wheel—many pieces of software reimplement matrix multiplication, and many of these don't need to. While in a few cases this can help with performance on new architectures with poor library support, or give better control of data placement and movement in very precisely-optimised code, in the majority of cases using open-source or vendor-specific libraries would give improved performance. Part of the problem is suggested to be one of recognition—in some cases those writing the code do not realise that what they are writing can be expressed as a matrix product. In other cases it is that those writing the software are not aware that libraries are available, or how to use them. Training can help with all of these cases.

Software citation was also discussed; one question was whether it is reasonable to cite large numbers of dependencies that an author hasn't interacted with directly, even if they were essential parts of the computation. For scientific software it is common to cite the paper introducing a piece of software, which does not adequately recognise contributors who have enhanced or extended the software since the original work was published. Improved rates of software citation can include discoverability of software—by reading a paper related to a problem of interest, it should be clear what software is used in the field. However, this is likely to only work within a field, and could also result in it being harder for new software to become known and displace older software that it outperforms.

This discussion ties into points raised around the need for software to be considered as a first-class research output. Publishing software directly rather than only the paper enabled by it can help (if not solve) the issue of recognition for maintainers. There is a distinction between those writing/maintaining community codes, and those who write software to enable their own research; the latter will receive more recognition in terms of citations as software begins to be cited directly, but the problem of research culture and funding both rewarding papers means that the problem of how to retain talent and resources to maintain software remains a concern.

Summary

Rather than attempt a narrative synthesis of the many topics and themes explored in the workshop, we summarise our findings in the form of a table with one column for each science domain and one row for each computational issue or workflow characteristic. The results of the companion workshop on Memory Intensive Workflows are also included (left-hand panel). The corresponding element is populated based solely on the documentary evidence from the workshop; absence of an 'x' need not imply that a particular issue is not important or characteristic of a particular domain. Some overarching conclusions are clear, however: distributed-memory parallelism as a means to high performant computational science is almost ubiquitous, there is a clear direction of travel towards running high arithmetic intensity simulations on GPUs, and memory capacity and bandwidth, and storage capacity will continue to be significant bottlenecks.

	Cosmology	Chemistry/Materials MI	Fluid Dynamics	Neuroscience	Climate/Meteorology MI	Plasma/Fusion MI	Lattice QCD	Chemistry/Materials ES	Biomolecular	Climate/Meteorology ES	Plasma/Fusion ES
Unstructured mesh		x	x			x					x
Adaptive mesh refinement	x										
Mesh-free method	x	x	x					x			
Memory capacity constrained	x	x		x	x	x			x	x	x
Memory bandwidth/latency constrained	x	x	x		x			x	x	x	
Disk I/O constrained	x				x		x			x	
GPUs	x		x	x	x		x	x	x	x	x
Machine learning/artificial intelligence				x	x	x				x	x
Cloud				x	x						
Smart data movement/mixed precision	x			x	x						
Large sparse matrix operations		x	x			x	x				
Ensemble forecasting/time parallelism					x				x	x	x
Time critical					x					x	
Task-based parallelism	x										
Code coupling		x	x		x			x	x		
Workflow management system				x	x		x	x	x	x	
SIMD/SIMT							x				
Whole-machine via trivial parallelism				x				x	x		
Training needs					x			x	x		
Distributed-memory parallelism	x	x	x		x	x	x	x	x	x	x
Storage-constrained	x	x	x	x	x	x	x	x	x	x	x

It remains to thank workshop participants for their willingness to rise to the challenge of an unusual exercise, before an unfamiliar audience, in some cases with previously-unmet collaborators; for generously sharing their time and their knowledge (both precious commodities); and finally for providing invaluable feedback for early drafts of this report. Needless to say, any remaining inaccuracies are entirely our responsibility.

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

Appendix A: Names and affiliations of participants

Lattice QCD:	Luigi del Debbio	U. Edinburgh
	Biagio Lucini	U. Swansea
	Antonin Portelli	U. Edinburgh
	James Richings	U. Edinburgh
Biology, Chemistry & Materials:	Vassil Alexandrov	STFC Hartree Centre
	Alin Marin Elena	STFC Scientific Computing
	Dimitar Pashov	KCL
	Andrea Townsend-Nicholson	UCL
	Scott Woodley	UCL
Biomolecular Modelling:	Charles Laughton	U. Nottingham
Climate & Meteorology:	Pier Luigi Vidale	NCAS Reading
	Nils Wedi	ECMWF
Plasma & Fusion Science:	Max Boleininger	UKAEA
	James Cook	UKAEA
	Andy Davis	UKAEA
	Shaun DeWitt	UKAEA
	Leo Ma	UKAEA
	Joseph Parker	UKAEA
	Michael Ball	BBSRC
DiRAC:	Ed Bennett	U. Swansea
	Simon Hands	U. Liverpool
	Clare Jenner	UCL
	Mark Wilkinson	U. Leicester
	Laura Pecorone	
	Alastair Williams	

Appendix B: List of discussion questions (scripted and unscripted)

- Which aspects of the other workflows do you recognise? Is ES the right model?
- How is your community managing/planning the transition from petascale to exascale computing?
- What are the storage requirements of your field? To what extent is this a bottleneck/inhibitor for progress?
- How are you responding to the growing requirement for effective data management/curation?
- How would your community ensure the best science is given priority access to resource? How might a peer review process look?
- Will you be using Machine Learning/Artificial Intelligence to advance your research?
- Do you see an advantage in migrating from CPUs to GPUs? How are you managing the transition?
- What use could you make of cloud services?
- Do you foresee any role or need for quantum computing and/or quantum algorithms?
- What role do you foresee for emerging architectures, e.g. FPGAs?
- Do you have sufficient access to technical expertise, e.g. RSE support, to pursue your goals effectively?
- Software sustainability: how do we make code modular: small; reusable & replaceable; easy to use; easy to fond; easy to install?