Prof. A. E. Trefethen, University of Oxford
Prof. N. J. Higham, University of Manchester
Prof. I. S. Duff, Rutherford Appleton Laboratory
Prof. P. V. Coveney, University College London

Developing a high performance computing/numerical analysis roadmap

Abstract

A Roadmap Activity in the UK has leveraged US and European efforts for identifying the challenges and barriers in the development of high-performance computing algorithms and software. The activity has identified the Grand Challenge to provide:

- Algorithms and software that application developers can reuse in the form of high-quality, high performance, sustained software components, libraries and modules
- a community environment that allows the sharing of software, communication of interdisciplinary knowledge, and the development of appropriate skills.

Through a series of workshops and discussions with UK HPC application groups and numerical analysts five areas of challenge have emerged.

The HPC/NA Roadmap Activity

Methodology

A full description of the methodology used to develop the roadmap is provided in Annex 1. In essence the effort has been a combination of background desk work, a series of workshops and a collaborative community site. The latter has not provided input to this version of the roadmap but should provide significant input in the future. The three workshops held in Oxford, Manchester and London are described in full in Annex 1; they brought together applications developers, numerical analysts, computer scientists, industry scientists and computer vendors. The outputs from the workshops have been distilled and circulated to the broader community.

Roadmap Themes

A number of themes have been developed through the consultation. Not surprising these are largely mirrored in other international activities although some are local to the UK. We note particularly that the general themes that have emerged appear to match those in the French activity "Thinking for the Petaflop". It is important that we focus on understanding and considering the UK areas of strength within the context of these themes to make sure that investment and development build on them.

Cultural

Cultural issues around sharing

- Some application domain scientists are used to sharing models and codes, and reusing other people's software. For other domains this approach is almost completely alien with codes being entirely developed within a particular group and little use being made of libraries or other third-party software.

International boundaries/collaborations

- Many of the application groups have international collaborators or in some cases depend upon software developed in other countries (particularly the US) that may or may not continue to be supported. It is suggested that a map of international developments is created and a repository of information about ongoing activities is developed.

Development of a Community

- There was a general desire to have activities such as this workshop to develop more of a community across applications and across application/numerical analysis and computer science borders. Bringing together these interdisciplinary groups is very valuable and allows a transfer of knowledge from one field to another. A sequence of events and activities should be developed to assist in the communications across the community.

Applications and Algorithms

This section provides a high-level view of some of the issues and challenges for applications and algorithms; a more detailed section follows that will provide a more in-depth consideration.

Cross application commonality

- The following section identifies and articulates the commonality of algorithms across the applications considered and the challenges for the future.

Integration across models

- Many applications involve multiple models at different scales or for different elements of the application. Bringing independent codes together can be difficult due to any number of issues – lack of standards for data models and formats, interoperability of programming models, and lack of knowledge of error propagation through the integrated system.
- Integration of application pipeline (e.g. CAD, simulation, Visualisation) again these components often require different data formats and the like.
- In many applications there is a pipeline of activities: first, setting up the model; then the actual calculation; finally visualisation and analysis. A common concern was the lack of integration of the pipeline thus requiring a lot of effort to go, for example from the calculations/simulations to the analysis.

Error propagation across mathematical and simulation models

- It was recognised that there is a great deal to be understood regarding error propagation through a given model. This is compounded in the integration across models and pipelines.
- As architectures become heterogeneous there is also the need for algorithms that support mixed arithmetic.

Adaptivity

- There is a need to have adaptive algorithms to adapt to problem characteristics and also architectural constraints. This may include dynamic algorithms that adapt at runtime and algorithms that might adapt according to experience.

Efficiency

- As architectures become more heterogeneous and components might be power hungry, there is a need to develop algorithms that are energy efficient.

Scalability

- As noted above scalability is a huge problem for many application areas, and a desire for all application areas. The desire to solve bigger problems faster is one of the main drivers of this community. Most applications do not scale beyond a few hundred processors, and this is widely perceived as inadequate as we move to petaflop-scale machines.
- As applications scale there is a need to develop algorithms that minimize communications to enable that scaling.

Partitioning & Load balancing

- As systems become larger and more heterogeneous, load balancing and problem partitioning will be increasingly difficult. The need for load balancing may arise from from the hardware, faults, and or the applications/algorithms requirements. Methods for dynamic concurrency generation and dynamic runtimes that default to a static model as needed will be required.

Data management

- As applications scale so often too does the data be it analyzed data, output or other, and there are many issues around data distribution, replication, integration, integrity and security that need to be addressed. This includes management of metadata and ontologies.
- Ability to manage locality and data movement will be of increasing importance as memory hierarchies increase in complexity, making efficient use of bandwidth and scheduling for latency hiding will continue to be important.

Scalable i/o

- Input and output is important for applications not just in terms of writing out results but also in terms of enabling efficient and effective checkpointing. As applications scale to larger number of processors, this capability will become increasingly important.

Exemplar applications

- It is suggested that baseline models for a set of specified applications are developed to enable communication and benchmarking of new algorithms.

Software

Language issues

- There is a need for mixed language support: A variety of languages are used for application development. There is a need to consider how best to support this mixed language environment to allow better code re-use. This needs to allow composability, portability and support for standards.
- Similarly there is a need for sustainable software that through backward compatible provides interoperability.

Ease of Use

- Higher level abstractions should allow application developers an easier development environment. The provision of efficient, portable "plug-and-play" libraries would also simplify the application developers' tasks.

Efficiency and Performance

- Ability to manage locality and data movement and to schedule for latency hiding.
- Performance transparency and feedback providing the user with a layering of capability and tuning.
- Capability to control energy efficiency.

Support for development of software libraries and frameworks

- More effective code reuse is essential. This could be achieved by supporting software library development and frameworks for reuse.

Validation of software and models

- There were concerns from many application developers that there are not well defined methods and techniques for validating scientific software and the underlying models. In some application areas observational data can play a role in validation, but for many this is not the case.

Software engineering

- It is often the case that application teams developing scientific software are not as skilled in software engineering as would be desired. Guidance on best practice for software engineering development would be a step to assist the community.

Standards and Compilers

- There is a need for standards to enable composability of models and it is clear that there will be a need for more sophisticated compiler and development suites. (The latter is likely to be an industry development.)

Active libraries & code generation

- In order to be able to move from one platform to another it would be beneficial to have underlying libraries that "do the right thing" for any given platform. This is becoming increasingly important with the plethora of new architectures that need to be considered.

Sustainability

There is general concern regarding the sustainability of application codes, software libraries and skills (we consider skills in the next section).

There is a need to develop models for sustainable software that might include

- Long term funding
- Industrial translation
- Open community support
- Other

The question of sustainability is also linked to the issues identified above in programming models and the need to maintain compatibility and interoperability.

Knowledge Base

Lack of awareness of existing libraries/packages

- It became clear through the workshops that there is patchy awareness of what is already available. It would helpful to the community to develop mechanisms for collecting information on existing software and tools and disseminating effectively.

Skills and training

- All presentations at workshops mentioned skills in academic research groups and industry alike. There are simply insufficient students being trained with the required skills, mathematical, software engineering and high-performance computing. Approaches to this include MSC and graduate training, computational science internships and short courses or summer schools.
- As well as integrated approaches to high-performance algorithms it was noted that there were some specific areas such as optimization where there is scant education for graduate and postdoctoral researchers, but which is likely to be an area of increasing importance across a number of application areas.

Lack of awareness of expertise

- Providing a repository of expertise of numerical analysis and application domains in the UK may assist in developing appropriate teams for activities.

Conclusions

This activity is ongoing and the UK roadmap for algorithms and applications will continue to evolve. For more a more detailed analysis of the findings and additional information see http://www.oerc.ox.ac.uk/research/hpc-na.

Acknowledgements

We are grateful for the support provided by EPSRC for the algorithm/application roadmapping activity. The activity has been successful thanks to the work of the team at OeRC and in particular Mark Hylton and Stef Salvini.