MolSSI Workshop Report

Executive Summary

A workshop to understand the broader cyberinfrastructure requirements of computational molecular sciences was organized at Rice University, Oct 8 and 9. The opening session of the workshop had several talks illustrating the importance of a robust, functional and scalable ecosystem of cyberinfrastructure capabilities that goes beyond simple "single code" performance. These requirements ranged from the ability to run scalable ensemble based simulations to machine learning analysis to improve the parameterization of force-fields. At the conclusion of the first session there was a "talk from the trenches" that conveyed a group perspective of the computational challenges and software development lifecycle challenges inherent in molecular sciences. The challenges were perceived to be representative of groups across many size scales. The second session of the workshop was devoted to discussion of the wide range and requirements of workflows; this session did not discuss specific tools or technologies but tried to capture the richness of the requirements and importance of workflow capabilities.

The second day of the workshop had two sessions comprised of invited talks. The first was focussed on a wide range of resource and software capabilities providers; the second day was focussed on the data analysis state-of-the-art and software development requirements. Resource providers included Blue Waters, the Open Science Grid and DOE's OLCF; each of these discussed the resources and how they could be useful for Molecular Sciences community. Resource providers discussed requirements of the community they were well suited to support. Resource providers were in agreement on the importance of community capabilities and activities as a way to improve their ability to support computational molecular science.

The workshop concluded with a robust discussion on the following topics: (i) What are the primary Software Development, deployment challenges that different projects face? And, (ii) From the participants perspective, how and where should/could MolSSI make a difference? An important point that emerged in discussions with the criticality of changing the culture and best practises around software development as much focus on specific software development.

Workshop Proceedings:

Science Drivers: State of the Art

[Cecilia, Eric]
Domain-specific software challenges in molecular science include ``the sampling problems'', long-range interactions, many-body problems, and other parallelization challenges, the need for thermodynamic correctness in sampling, and visualization (different for every domain). Additionally, computational molecular scientists face the same cyber-infrastructure challenges as other users of nationally funded compute facilities but can more effectively address them by concentrating on a more limited number of tools within the domain.
MolSSI can benefit from efforts of broader computing communities when supporting domain-specific infrastructure development by focusing on middleware inserted between existing domain science codes and generalized workflow and resource managers designed to support domain state-of-the-art compute methodologies. Domain compute needs are diverse, including HPC, HTC, and small urgent tasks as well as enormous low-priority tasks.
Mol.Sci. groups benefit from the efforts of many large multinational software projects as well as employing small ad hoc software projects. Researchers face challenges running mature software in diverse computing environments with different resource management and job scheduling paradigms. Less mature projects struggle to identify the strengths and pitfalls of different computing environments, manage software dependencies, build requirements, distribution, deployment, and ultimately usability. It will be easier for computing service providers to help our scientists with virtualization technologies when MolSSI can provide a point of contact and communications conduit to a unified community.
After overcoming the challenges of choosing the right tool(s) for the job, a researcher struggles with interoperability of software and file formats from non-cooperating sources. A research group wishing to contribute new methods to the community as well-engineered and efficient code faces limitations on project continuity and programmer hours due to the limited time of PhD students / post-docs and conflicting demands on time/energy. This conflict is partly due to the mandates of a particular academic program and a culture that undervalues research software engineering versus research data.

Testing, validating, and benchmarking software can require securing additional compute resources.

Workflows Requirements and Challenges

[John, Julien, Peter]

A. Goal is to **create an ecosystem where entities can share workflows and components to enable**:
    a. **Reusability**: Allow investment in software/methods to see maximum return
    b. **Reproducibility**: Allow other researchers to reproduce and evaluate published work
    c. **Training**: Enable researchers to train new scientists on best practices workflows
    d. **Provenance**: Track how data was generated by specific chain of tools
    e. **Citability**: Everything should have citable DOIs (see e.g. Zenodo)
    f. **Sharing and evaluation of best practices** for various common tasks in comp chem
    g. **Allow scientists to focus** on specific problems without having to reengineer entire workflow
    h. Automated **testing of individual components** using standard testing data
B. **Define standards for interoperable components** that are portable across different workflow systems. Could use container + metadata that describes how I/O and features are exposed
C. **Define data format standards** that allow exchange of data among multiple workflow components. Also providing useful tools (e.g. core libraries) that can read/write these standards with hooks in multiple languages may be necessary for making it easy for the community to adopt these standards.
D. **Capture requirements** for common computational chemistry workflows
    a. Workshop focusing on workflows in comp chem?
E. **Evaluate workflow systems** to see if they meet requirements. Implement new solutions if needed, contribute to existing workflow systems, or disseminate/support workflow systems that meet requirements. Requirements may include:
    a. facile deployment across infrastructures and platforms; utilize cloud, local, HPC, HTC
    b. Tool runtime system isolation via container infrastructures (Docker/Singularity/Shifter/Kubernetes)
    c. automatic parallelization
    d. Support for adaptive execution; non-static workflows (e.g. adaptive sampling strategies)
    e. balance ease of use with ease of extensibility
    f. consideration of easy of use to enable use by non-experts
    g. ease of interoperability with data analysis tools (IPython notebooks, web services analytics)
    h. scalability
    i. portability
    j. fault tolerance
    k. access to cloud and local data storage
    l. workflow and component versioning
    m. pata provenance tracking / metadata
    n. How do we cope with non-free components?
        i. If MolSSI provides workflow infrastructure, could charge users for non-free software costs plus infrastructure costs through central clearinghouse.
        ii. A central entity with unified license agreement (like the Apple “app store”) greatly simplifies licensing effort from industry (e.g. pharma) instead of having to sign separate license agreements/contracts with every tool vendor
        iii. “Pay for privacy” model? Everything is free if results of computation become public. Can pay to keep them private / generate IP.
    o. Who pays for compute resources and how? Subsidies from grant agencies?
    p. Component and workflow discovery; “App store”?
    q. Performance, especially for very quick operations where container overhead is significant
    r. Can we regenerate intermediate results if needed?
F. **Provide a mechanism for sharing datasets**
    a. Find a sustainable funding model (e.g. users pay egrees charges; valuable datasets paid for by public NIH/NSF money). See NIH Commons experiment?
    b. Standard benchmark datasets, e.g.
        i. D3R/SAMPL datasets: https://drugdesigndata.org/
        ii. high-quality QM datasets
        iii. experimental physical property datasets like ThemoML Archive: http://trc.nist.gov/ThermoML.html

   c. Publication datasets
   d. Blind community challenge datasets (e.g. D3R, SAMPL)
   e. Everything should have citable DOIs (see e.g. Zenodo)
G. How can we **incentivize toolmakers** to encapsulate their tools as reusable workflow components?
   a. Make it as easy as possible for users to encapsulate their tools
      i. Provide infrastructure that makes it easy
      ii. Good tutorials and training
      iii. Lots of examples
   b. Work with grant agencies (NIH/NSF):
      i. Carrot: “Well give you extra money/resources to do science if you put your tool into this reusable form”
      ii. Stick: “We won’t give you money unless you put your tool into this reusable form” (e.g. NIH unfunded mandates)
   c. Provide **metrics of usage** to justify value to funding agencies
H. **Examples** of computational chemistry workflows to look to for inspiration:
   a. OpenEye **Orion**: http://www.eyesopen.com/orion
   b. **Copernicus**: http://copernicus.gromacs.org/
   c. **FireWorks**: http://pythonhosted.org/FireWorks/
   d. **MDWorks**: https://github.com/Becksteinlab/mdworks
   e. **Pegasus**: https://pegasus.isi.edu/
   f. **Kepler**: https://kepler-project.org/
   g. **Ensemble Toolkit**: https://radical-cybertools.github.io/entk/index.html
   h. **Pipeline Pilot**: http://accelrys.com/products/collaborative-science/biovia-pipeline-pilot/
   i. Knime: https://www.knime.org/
I. Some example use cases of interest (totally incomplete):
   a. D3R/SAMPL automated blind community challenges
      i. Participants would submit workflows or components that are automatically used as part of a “blind challenge” workflow harness that does automated evaluation
   b. CELPP automated docking challenge using weekly RCSB releases

Data Analytics

[Arvind, Oliver]

A large number of CPU cycles on national supercomputing resources are consumed by biomolecular simulations. Analysis of MD trajectories is becoming a major bottleneck while modern HPC resources have made it much easier to produce terabyte-sized datasets with comparative ease. A wide range of software exists that allows the user to read trajectories in one of the many file formats that exist in the community. Monolithic tools (e.g., VMD, Gromacs, cpptraj, CHARMM) have a long history and are used by many researchers but only provide fixed analysis tools. Libraries (such as LOOS, MDAnalysis, mdtraj, pytraj) provide unlimited freedom to implement novel tools but are often harder to use, although the prevalence of Python (with the NumPy or a buffer with the array interface as the common data structure for coordinares) as the language of choice has made this avenue much more approachable. Most modern analysis libraries already adhere to good software engineering practices, including code being stored in software revision control systems, (git on GitHub is prevalent) and use of continuous integration with automatic unit testing is common. As such, these codes can already serve as good examples of many of the **best practices** that the MolSSI might want to propagate.

The large number of high-quality projects represent a thriving and competitive environment. This competition is generally considered good for users, is felt to overall increase the standards in the field, leads to percolation of new solutions throughout the software eco-system (if available under an open source licence), and creates a community of developers with broadly common goals. On the other hand, it leads to competition for resources (developers, funding) and to duplication of effort. There is an opportunity for the MolSSI to catalyze efforts to unify common software parts into lower level libraries that can be used by all projects. As an example, most codes libraries that analyze MD trajectories implement their own code to read/write various file formats. A low level *mdfileformats* library that combines the best approaches from the community would be an important step in the right direction. The MDAnalysis project, for instance, is moving towards refactoring “readers/writers” towards a lower level without any MDAnalysis-specific code.

Analysis tools should allow users to utilize existing resources, ranging from many-core desktop systems to hybrid HPC

systems with GPU accelerators. At the moment, support for GPUs is not standard, because GPU programming requires substantial expertise and investment of time although some tools such as specialized dimensionality reduction algorithms (such as quasi-anharmonic analysis (QAA)) already make use of accelerators. The most common approach to accelerating analysis code is to write performance critical parts in a compiled language and use OpenMP thread-level parallelism. Parallelizing over trajectories when analyzing ensembles of trajectories is supported by various tools (e.g., cpptraj or radical.pilot). Parallelizing over slices of trajectories is more difficult and can sensitively depend on the underlying file system and the file format as shown for an example of MDAnalysis with dask.delayed. Some other tools appear to have made some progress in this direction (e.g., HiMach, pytraj) but more standardized benchmarks would be desirable.

Overall, there seems to be much room to improve the compute bottleneck at the analysis step. Integration with workflows and approaches to organize and curate simulation data also appear to become more important.

Resource Providers Perspectives

[Brett, Lauren, Sanjay, Dmitry, Sudhakar]

a. On Blue Waters official support is provided for software packages requested by multiple science teams, so far no workflow software has met that bar. Multiple packages are in use, but each team is solving the problem independently. Existing teams are making use of workflow tools including Condor, Swift and RADICAL-Pilot, but mostly to the extent of bundling tasks within one batch job. As a resource provider we would like to see adoption of common tools. Common tools enable better support and testing by resource provider staff and a better experience for science teams. Workflow systems need to enable efficient science output and hide as many of the individual system idiosyncrasies as possible while working with the system resource manager to improve overall resource utilization. In addition, the workflow tools must manage faults effectively. Node and jobs will fail. The workflow system should manage those failures rather than the user, yet provide sufficient logging to easily diagnose failures. Such as system would have significant benefits for users while also lowering the support burden for resource providers.
b. New (to HPC) technologies such as Spark, machine learning and containers offer opportunities to ease the burden of transitioning to a new system. However, it is not clear what limitations containers might impose on scalability or low level hardware usage (such as GPUs).
c. Resource providers would love to see good software development practices adopted by software providers across the board. Developers with good software development skills are more likely to write quality, portable, and testable code that can be installed and tested on a new system without a major effort by center staff.

Science Gateways

Science Gateways Community Institute (SGCI) funded under the SI2 program has just started functioning. It is an online community space for science and engineering research and education for accessing data, software, and computing services. Science Gateways Community Institute consists of 5 focus areas – An Incubator, Extended Developer Support, a Software Framework, Community Engagement and Exchange, and Workforce Development. The incubator provided planning for technologies, business and client management. Specifically technology planning include security, cyberinfrastructure, evaluation and impact assessment and resource development. The business model development, project management and licensing and sustainability are covered under business planning. Customer facing interfaces and support infrastructures will be covered under client management. Extended developer support is modeled as as extension and complementary to the highly successful XSEDE ECSS program to provide front-end development, technology integration and prototyping, “burst” support and gateways deployment and dev-ops support. The Software Fraweworks area is to serve end-to-end solutions for portable and reproducible community products encompassing community standards

and help organize gateway discovery through a registry and promote gateways across community through forums and provide documentation, user guides and organized outreach. Community engagement area is responsible for organizing workshops, symposia and showcase gateways for the communities and obtain feedback in terms of surveys and provide specific way to continually improve services and user experience. The work force development consists of bringing up developer and user communities and engage in training for a sustained support of human capital by providing skills development and work-study opportunities and internships and organize gateway related job-boards. A specific component here is to address skills and workforce development among MSIs and under represented groups. Using these focus areas the goal of the institute is to provide a sustainable operation of science gateways to maximize the impact of NSF investments in HPC and software development in broadening and deepening the community participation.

NSF XSEDE program has Extended Collaborative Support Service model has been successful in providing specific and targeted consulting support for allocated grants in maximizing usage of NSF funded HPC resources with significant benefits to the PI as well as to the community. Exemplars of such collaborative engagement deployed scientific software optimized for specific complex HPC environments and established community access through Science Gateways.

Apache Airavata is a software framework adopted under NSF funded SciGaP project to provide Science Gateways as a platform providing gateway hosting capabilities in a multi-tenanted fashion. Apache airavata generalizes services across gateways and deploy them in a community developed open environment with the Apache Foundation's discipline in open software management for maximizing the sustainability. Several Science Gateways have been deployed under SciGaP project and being operated using Apache Airavata and the SEAGrid Science Gateway is one that is mainly focused on molecular and materials applications.

SEAGrid Science Gateway provides seamless access to a wide variety of popular molecular sciences applications deployed at NSF HPC systems available under XSEDE and Bluewaters projects as well as resources hosted at local campus level to communities and individuals with secure access. SEAGrid provides both web based and desktop application based interfaces to provide services such as authentication, application specific input preparation, job submission and monitoring and post processing as well as research data management systems. The SEAGrid data pipeline also provide a way to automatically parse the job result data and provide interfaces to discover and re-engage the data for reuse, sharing among collaborators and publication.

Workshop Discussion:

1. What are the primary Software Development/Deployment/Challenges that you/your project face?
    a. Heterogeneous environments (compatibility, optimization, runtime configuration)
    b. Managing compute resources / mapping workload to available computes
2. How can and where should MolSSI make a difference?
    a. Everything Daniel said
    b. Software?

i. perform software development that improves research efficiency in ways for which project developers are not motivated
    1. libraries and APIs for file formats already in the wild
    2. contribute code that improves interoperability between software
    3. consider information security issues
    4. what about experimental research computational tool needs?
        a. micrograph / tomography analysis
        b. diffraction measurement analysis, simulation
        c. data interpretation, modeling
ii. develop infrastructure such as information portals
iii. participate in non-domain-specific projects while representing the domain-specific interests: help make sure cloud compute services, workflow managers, APIs, etc. have the hooks and optimizations most useful to domain software and workflows

c. Beyond Software?
    i. produce white papers on current and upcoming challenges
        1. playing nice with Lustre filesystems
        2. starting large parallel jobs with large input data
        3. ways to reduce domain communication / synchronous operations
        4. ways to better exploit cloud computing resources
        5. robustness against failures on small parts of parallel resources
    ii. Establishing software development best practices / training
        1. Integrate recommended practices (e.g. GitHub, continuous integration, unit tests, package deployment)
        2. Produce written online materials intended for early grad students to overcome limitations in undergraduate and graduate education in scientific computing
        3. Summer bootcamps focused on software engineering practices?
    iii. Encourage good software development by recognizing good software development. Conventions on author contributions and citations _are_ domain-specific, so this _is_ a task for MolSSI. Maybe MolSSI can't help software development lead to authorships and citations in scientific research journals, but maybe an alternative standard of recognition can be established or promoted. **Product idea:** Formalize and provide tool or portal to identify code contribution: present commit logs in a citable, clearly formatted way. Work with publishers in our domain on ideas for the right way to recognize software development effort.
    iv. A research group can be very successful by building a unique tool-set and then identifying low-hanging fruit, finding problems to fit the tools rather than finding the right tool for the job. Can MolSSI drive a research agenda by choosing what tools to develop? Does this give MolSSI a power that it can leverage for a seat at the table in discussions previously hard to participate in? E.g. funding large-scale compute facilities with capabilities that map to research requirements.
    v. build community
        1. Sklogwiki, alchemistry.org as a model/inspiration for molecular simulation knowledge and MolSS development (a living sequel to Allen & Tildesley, Frenkel & Smit)
        2. presence at conferences as outreach and to present talks
        3. coordinate workshops like CECAM and VSCSE
        4. suggest standards / conventions
            a. file formats, APIs, user interfaces (Python, visualization libraries)
            b. curricula for computational science certificate programs
    vi. represent the interests of research software engineers
        1. liaise with public and private funding sources, compute facilities, hardware/software providers, industry, government
        2. work with cluster administrators to ease software build and deployment, virtualization
        3. participate in working groups for library / API / language specification revision.
        4. provide a point of contact for entities like national HPC resources, HPC vendors, who need to understand the computational needs of molecular simulation or to convey upcoming shifts in computing paradigms

d.

3. Lessons for MolSSI