7b: Technologies for Extreme Scale Computing (Software)
DOE FY19, Release 1 Phase I SBIR/STTR

Proposal

SASI: Smart Algorithm Selection through Inference

Ben O'Neill[1], PI, Gerald Sabin[1], Boyana Norris[2], Sub-contractor

[1] RNET Technologies
240 W Elmwood Dr
Dayton, OH 45459-4296
boneill,gsabin@rnet-tech.com

[2]Department of Computer and Information Sciences
1202 University of Oregon
Eugene, OR 97403-1202
norris@cs.uoregon.edu

1 Identification and Significance of the Problem or Opportunity, and Technical Approach

RNET Technologies Inc. (RNET) in Dayton, OH and Professor Boyana Norris from the University of Oregon (UO) are responding to the 2019 DOE SBIR/STTR (Release 1) Topic 7b: Technologies for Extreme Scale Computing (Software). RNET and UO are proposing the development of the Smart Algorithm Selection through Inference (SASI) toolkit; a machine learning framework for the run-time optimization of numerical algorithms in terms of performance metrics such as CPU-time, memory consumption, energy usage, and resilience. The SASI framework will be built based on the ideas, workflows and results obtained in the projects team previous work developing machine learning models for optimal algorithm selection in linear solver packages such as PETSc. In addition to this ongoing work, RNET has extensive experience in various aspects of High Performance Computing such as performance optimization of numerical softwares and libraries, development of fine-grained power monitoring tools for HPC infrastructure and large scale data analysis tools. Dr. Boyana Norris (UO) has extensive experience in enabling technologies for high-performance simulations in computational science and engineering. Specifically, her research on performance analysis of scientific codes spans a spectrum of topics, including static analysis, runtime performance monitoring, performance database management, postmortem performance analysis, and source transformation tools for performance tuning. Therefore, this team is well positioned to develop and commercialize the proposed smart algorithm selection package for large-scale numerical simulation.

1.1 Identification and Significance

In an effort to further improve the nations numerical simulation capabilities, the DOE, academia and other government agencies have invested heavily in the development of efficient algorithms and implementations for solving the broad spectrum of subproblems that arise in numerical simulation. The result of this investment is a robust collection of software packages designed to solve numerical subproblems across a broad spectrum of applications and compute architectures. One only needs to look at the expansive collection of linear solvers and preconditioners available in PETSc to see the shear volume of implementations and algorithms that have been developed to solve just one, albeit important, class of numerical subproblem.

While much work has taken place in optimizing the algorithms and implementations for specific problems and architectures, there is no simple governing theory for choosing between the numerous algorithms and configurations. Rather, the optimal method is, in practice, determined by experimentation and numerical folklore [2]. The experience and expertise provided by the domain scientists in tuning the algorithms cannot be underestimated, however, it is desirable to have toolkit-enabled functionality that automatically chooses the algorithm that is both appropriate for the computational problem at hand *and* the computer architecture that will be used. Automatic selection of an appropriate algorithm and configuration can lead to benefits such as reduced memory requirement, lower execution time, fewer synchronization points in a parallel computation and so forth.

As an example, consider the large sparse linear systems that arise when solving a nonlinear partial differential equation. One could take a guess as to the best linear solver based on coarse grained problem characteristics and experience, however, it is all but impossible to robustly de-

termine the best solver in all cases. In fact, it has been shown that there is no uniformly optimal linear solver for general matrix-vector systems and that as the simulation progresses and the linear systems produced by the simulation changes, the best preconditioned iterative solver to solve each linear system can also change [3]. Similarly, in applications with dynamic mesh adaptation, run-time changes in the structure of the mesh (both physical and geometrical) have also been shown to lead to changes in the linear system and hence, the optimal linear solver. The memory constraints of the solver can also change according to the code implementation or the architecture of the machine, highlighting the fact that the compute architecture also plays a huge role in the performance of different algorithms. In other words, for optimal performance, the choice of linear solver should not be a static, a-priori design parameter, but rather, a dynamic, architecture dependent decision made based on the properties of the problem and the performance goals of the user. The same argument holds true for the vast majority of numerical subproblem solvers, including eigensolvers, nonlinear solvers, graph algorithms and nonlinear optimization routines.

To address this need, RNET and the University of Oregon are proposing the Smart Algorithm selection through Inference (SASI) framework. The idea for SASI was born out of the ideas, lessons and workflows developed during the project teams previous work on automatic solver selection for linear solvers, eigen-solvers and graph algorithms. That previous work has continuously shown that SASI style performance models that form the foundation of the SASI framework are capable of repeatedly predicting, with an extremely high accuracy (i.e., 95%+), the performance of numerical algorithms and solver configurations across a wide range of applications and domains.

The typical workflow for a developer and end-user of advanced numerical simulations involves testing, development and production runs on a wide range on computational resources, ranging from laptops and locally hosted clusters, through to leadership class HPC resources. SASI models allow developers to create simulations that automatically adapt to both the arcitecture and the problem, removing the need for architecture and problem specific tuning, ultimately reducing the overall time to solution for numerical simulation and ensuring the effcient usage of the nations HPC resources.

1.1.1 Smart Algorithm Selection Through Inference

The proposed framework will guide the user through the process of building and using smart algorithm selection models for performance optimization in terms of metrics such as CPU-time, memory usage, energy efficiency and resillency, automating the process whereever possible. Such optimizations will allow developers to deploy simulations that are optimized to perform within the constraints of the given architecture; a feature that will be of particular importance on exascale machines, where factors such as excessive power consumption and and poor algorithmic resillency are predicted to cause major problems.

The model building pipeline the SASI framework will look to automate can be split into four distinct components; feature determination, data collection, model building and validation and efficient software integration, each of which is rife with intricate details and nuances that must be addressed to ensure optimal performance in the final simulation:

- **Feature Determination:** In machine learning terminology, a feature is any obtainable metric, measurement or observation that can be made about the target phenomenon. Determination of an informative, discriminating and independent set of features is a crucial step in the

creation of unbiased and useful classification and regression based machine learning algorithms such as those used in SASI. In SASI, a feature represents a computable metric of the computational model (i.e, the linear system in a linear solver or the graph in graph solvers), that can be used to indicate or predict the performance of an algorithm when applied to a specific problem and on a specific architecture. The process of determining the best features for a given algorithm must be completed in collaboration with a domain scientist and cannot be automated. However, there are numerous feature set optimization techniques that can be used to enhance the perfromance of the SASI models, all of which will be included in SASI.

- **Data Collection:** A large diverse training data set forms the foundation of every machine learning model. In the context of SASI, training data takes the form of a set of measurements obtained by applying the different algorithms and configurations to a diverse set of sample problems (e.g., for linear solvers, this means solving a large number of matrix-vector systems using a wide variety of linear solvers, preconditioners and input parameters.) Collecting this data is the most expensive component in building a machine learning model, and again, requires a domain scientist famillar with a wide range of applications for the given algorithm. To aid in this process, SASI will include a robust database management system, along with several novel tools for assessing the validity and suitability of the training data.

- **Model selection and Validation:** The third component of the SASI workflow involves choosing a machine learning algorithm, building the model, and testing the result. Previous results have shown that simple machine learning algorithms such as a single decision tree or a Random forest can be used to create cheap and highly accurate machine learning models for smart algorithm selection. For instance, the decision tree based J48 algorithm was the most accurate algorithm for the linear solver based SASI models developed in previous work. SASI will include a modular, extensible interface for connecting to external machine learning toolkits that takes care of building, serializing and loading of the models internally, allowing developers to create models without getting into the details of the machine learning algorithms. SASI will also include a collection of standard and novel model validation tools such as the k-fold cross validaation method and 66-33% split model validation method.

- **Model Integration**. Using a pre-compiled SASI model at runtime is as simple as extracting the required features from the given computational model and feeding them into the SASI model. The model will then return the optimal solver for the given problem and architecture that can be used to solve the problem.

1.1.2 SolverSelector: Automatic Solver Selection for Linear Solvers

The fundamental ideas and workflows proposed for SASI were developed during the project teams ongoing Phase II SBIR project entitled Automated Solver Selection for Nuclear Engineering Simulations (Contract #DESC0013869). In the follow section we will outline the results of that work, as well as highlight the key differences between that work and the work being proposed in this Phase I proposal.

The primary objective of the Solver Selection SBIR project was to develop models for automatically selecting the optimal linear solver configuration based on the given computational model and the performance goals of the user. For linear solvers, the computational model is a matrix

vector system, $A\mathbf{x} = \mathbf{b}$, hence, the feature set is comprised of structural and physical properties of the matrix. An example of matrix features used includes the number of non zeros, the max number of non zeros per row, the non zero symmetry pattern, the infinity norm and the trace.

The training data was collected from two sources; the SuiteSparse matrix collection (formally the Florida Sparse matrix collection) and a set of matrices extracted from the MOOSE testing suite. MOOSE is a multiphysics finite element code built on top of libMesh that provides a number of finite element examples across a broad range of application areas [1].

The system $A\mathbf{x} = \mathbf{b}$ was solved multiple times for each matrix in the datasets, each time using a different linear solver configuration. The RHS was a vector of ones and a random initial guess was used when needed. An entry in the training data set was classified as "good" or "bad" using a binary classification system. Specifically, a solver was classified as "good" for a given matrix if the measured value (i.e., CPU time, memory usage) for that solver was within 80% of the best solver for the given matrix. Eighty percent was chosen based on experimentation because it meant a small number of solvers are classified as good for each matrix while not being so restrictive that the runtime models cannot find a good solver.

Table 1: Convergence model accuracy and build time for 10-fold cross validation with SuiteSparse data set.

Method	Overall accuracy ($\%$)	Build time
BayesNet	96.1	0.11
RF(100)	97.0	3.70
ADT	93.1	0.70
knn(10)	92.0	0.01
J48	95.7	3.70

Table 1 shows 10-fold cross validation tests for the machine learning models built using the SuiteSparse matrix collection. The best method, Random forest with 100 trees, classified good solvers as good and bad solvers as bad with an accuracy of 97% over the 50000 individual entries in the data set.

The main research goal of the SolverSelector project was to create a blueprint through which smart algorithm selection models can built. To that end, one of the success stories of the project has been the creation of a robust, documented and published method for the creation of smart algorithm selection models that has successfully applied to other algorithm classes such as eigensolvers and graph algorithms.

The development of Solver Selector API, which was built as a means of integrating automatic solver selection in NEAMS tools and is the primary commercial product of the SBIR Phase II award, has been equally successful. The SolverSelector API provides a simple interface through which users can use the linear solver models in advanced numerical tools. In particular, the API has a simple database management system based on Sqlite3, an interface to the Waffles machine learning toolkit, and built in support for gathering linear solver based training data. Currently, the SolverSelector API has also been integrated into PETSc as a standalone KSP solver and can be used in any PETSc based code (MOOSE, libMesh, Fenics, Proteus, etc) through the addition of a copuple of lines of code and two command line parameters.

1.1.3 The need for a SASI framework

RNET and UO have been working on developing, optimizing and implementing the blueprint for linear solver based smart algorithm selection for just over three years. That blueprint has repeatedly proven to be a robust pathway towards creating models for new classes of numerical algorithms; however, it has also proven to be incredibly difficult to implement, even under the guidance of the creators of the approach. As an example, it took a graduate student 9-12 months to develop the first smart algorithm selection model for graph algorithms, with numerous hickups and difficulties along the way.

While the SolverSelector API does provide some levels of automation (i.e., automated model building using Waffles and integrated database management with Sqlite3), it was not designed for generic numerical algorithms and was never intended to be a general toolkit for building the SASI models. That is to say, the goal of the SolverSelector project was to create and sell the technology (i.e., the SASI models), whereas, the goals of the proposed Phase I project will be to develop and sell the mechanisms for making the technology accessible to the broad numerical simulation community. Currently RNET is in a position to build and sell prebuilt SASI models and custom integration software for any new application, but, the process for developing those manuels is completely manual. Based on the demostrated performance of the models, we feel there will be a large market for a framework that automates the model building process, making smart algorithm selection available in academic and industrial settings where developing a model from scratch would otherwise be unfeasible.

SASI will be designed to guide the user through the process of building and integrating the models for generic algorithms, providing tools and mechanisms for avoiding the numerous hick-ups and complications that often arise. This will include tools for verifying the discriminatory properties of the features in the feature set, tools for visualizing how well the dataset covers the feature space, and tools for performing efficient data collection on large scale machines.

The SASI framework will also look to address one key issue associated with the existing SASI models; the requirement to rebuild the training data set from scratch for each new architecture and for each new algorithm. To that end, the SASI models developed with the proposed toolkit will be designed to facilitate the development of numerical software that automatically adapts to the problem being solved and to the compute environment being used, irrespective of the size (laptop, petascale, exascale, cloud) and type (CPU, GPU, FPGA, etc) of the available resources. The ability to adapt to new architectures and algorithms will be of particular importance on exascale resources, where the cost of running exascale simulations and the expected high demand for the resources during the initial rollout will make it difficult to obtain the exascale compute time required to build accurate SASI models. The SASI framework will leave the definition of ”optimal” up to the user, allowing the developers to create simulations that take into account the strengths and weaknesses of the available resources. For example, on exascale resources, the end-user could configure the SASI models to pick algorithms based on minimizing the power consumption or maximizing resilliancy. Likewise, on a cloud based cluster where network speeds can be an issue, simulations could be configured to prioritise algorithms with a high ratio of computational to communication.

2 Anticipated Public Benefits

The proposed project will facilitate runtime performance tuning of numerical simulations based on both the computational model and the compuation architecture, in terms of any combination of measurable metrics (e.g., CPU time, Memory consumption, energy efficiency). Numerical simulations, widely applicable across various scientific disciplines, are a key component in the development of most, if not all, of the cutting edge technologies being developed by the DOE and other U.S government agencies. By automating the process of tuning simulations for a specific architecture, SASI will provide the the computational scientists, numerical software developers, and electrical engineers with a mechanism for letting the software decide the best performing method without the need for labor-intensive experimentation, and thus focus on more scientific aspects of their applications. Moreover, by ensuring applications are optimized for the architecture for which they are being used on, SASI will inform the efficient usage of the nations current and emerging computational resources. The targeted customers include nuclear power companies, DoD and its Prime Contractors, NASA divisions, DOE agencies, CFD software providers, oil and gas companies, semiconductor design companies etc.

3 Technical Objectives

To demonstrate the technical feasibility of the proposed approach, the Phase I effort will focus on developing a functional proof of concept for the SASI framework. The specific objectives RNET Technologies, and its subcontractor (UO) will pursue during the Phase I project are:

- Develop the core interfaces and software indirections required to create a framework that guides the user through the process of building and using an accurate and reliable SASI model.
- Prototype a range of tools for ensuring the SASI models are valid, including automated testing and analysis of the features, training data and model.
- Investigate of approaches for reusing, modifying and/or re-purposing training data obtained on other architectures and/or from alternative algorithms or application areas in new models and for new architectures.
- Demonstrate the effectiveness of the SASI framework by applying it to graph algorithms, a numerical subproblem that project team has experience developing SASI models for, but for which no sophisticated integration framework has been developed.

By achieving these four objectives, RNET will demonstrate the scientific and commercial potential of the proposed package. The investigation into approaches for re-using data through Baysian transfer learning represents the key and novel scientific contribution of the proposed project. The core framework and that additional tools for validation and verifitcation of the model represent the technology that RNET will look to commercialize.

I have a tendency to promise the world in these applications –

Do you think this is to much for a phase I, or

4 Work Plan

Development of the SASI framework will be composed of two connected but distinct components; the development of numerous tools, analysis routines and machine learning algorithms required to build, test and validate the proposed SASI models, and the development of the core framework for gluing together those tools. In what follows, we outline the approach that will be taken to achieve the overall objectives of the proposed project.

4.1 Model, Data and Feature Set Verification and Analysis

The number one requirement for the development of an accurate machine learning model is a robust, discriminating and informative set of training data, features and classification metrics. Speaking from experience, it is all to easy to create a machine learning model that performs wonderfully well in testing and validation tests, but that has little to no predictive capabilities in real-world applications. Based on previous research, the primary causes for poor performance in SASI models are; (1) a feature set with poor discriminating capabilities, (2) a data set that does not adequately cover the parameter space of the given feature set and (3) a feature set that is too expensive for use in run-time applications. To that end, the SASI framework will include a range of tools dedicated to assessing the validity of the given model, feature set and data set. In particular, the Phase I effort will focus on the development of the following two tools:

4.1.1 Feature Sampling

The process of determining which features are appropriate for a given algorithm is highly problem dependent and best left up to a domain expert. However, once the features have been determined, there are several existing methods for assessing the validity of the feature set and for optimizing it for use in SASI models. One example of this is known as feature sampling, whereby we remove features from the feature set that do not possess enough variability to be of any statistical importance. For SASI, removing irrelevant features is incredibly important because it reduces the possibility of overfitting by enabling the creation of less complex models whenever possible. Moreover, it dramatically reduces the runtime cost associated with feature extraction, thereby increasing the potential performance benefits of using SASI models.

In our previous work, feature sampling was completed manually using the many feature evaluation tools available in the Weka toolkit, including the CfsSubsetEval with BestFirst search method, Gain Ratio, Info Gain and the Principle Components evaluators with Ranker search method. In this Phase I effort we will develop a tool for the automation of this process. This will include the development of an interface for interacting with the Java based Weka toolkit programmatically, as well as mechanisms for automatically verifying the results. While Weka enables rapid model exploration through the large variety of ML methods it provides, we expect to extend the framework in the Phase II project to take a preliminary Weka model and transfer it to another available machine learning infrastructure that may be less portable, but is better suited to the production environment of the application.

This automated feature sampling tool will also include an optimization algorithm for optimizing the feature set based on maximizing the accuracy of the model and minimizing the computational costs of feature extraction. Feature set optimization is an offline task and, in previous

research, full feature sets have been composed of around 30 discrete features; hence, a simple brute force search of the parameter space will likely be efficient enough to complete this optimization. If the efficiency of the method becomes an issue, more complex optimization methods will be investigated.

4.1.2 Database Management and Sanitation

Management of the large datasets required in machine learning models can be a complex and time consuming task. In addition to simply parsing the data into the machine learning model, one must also be concerned with cleaning, sanitizing and validating the datasets. To ensure this is an easy task, SASI will include a generalized version of the sqlite3 database management system developed for SolverSelector. To address database sanitation, the project team will develop simple interfaces for performing data orientated tasks such as detecting classification conflicts, replacing or removing missing data, rescaling data to fix prescribed ranges and automatic discretization of data into classifications. These are all standard, albeit time consuming, machine learning tasks that are well suited for automation.

A low-quality data set is a major reason for poor performance of machine learning models. However, because machine learning models are generally tested using the same data set from which they are built, which can result in a highly accurate (but overfitted) model. While there are multiple ways to arrive at this stage, in Phase I we will focus on how well the training and test data cover the problem's input space. This will be achieved by employing a mixture of statistical metrics (e.g., variance), parsing the data-set and determining areas of the feature set parameter space for which there is limited data. This data will then be presented visually as an intuitive database coverage map. This will allow users and developers to quickly determine areas where additional data is required. We will also provide interfaces to enable developers to describe the data in more detail than currently supported by most ML toolkits (for example, the type of distribution or a function of parameter values, if known). In turn, this will allow them to focus the data collection efforts on applications that will have a real statistical impact on the performance of the model, rather than repeatedly gathering data covering the same area of the feature space.

4.2 Re-purposing training data for use on other applications and architectures

In previous work, the machine learning models and training data has been architecture specific. The consequence of this is that the entire training set must be recomputed for each new architecture, a task that is extremely expensive and time consuming. The situation is similar when applying SASI to a new model because the results from different application areas cannot be used in any way in the new application. This means that, even in cases where the computational model has core components that are similar (i.e., linear solvers and eigensolvers), the user must create an entirely new algorithm specific data set. Data collection is the most expensive component of the SASI workflow, hence, maximizing data reuse will go a long way towards increasing the appeal of the methods.

To that end, the Phase I effort will involve a in depth investigation into methods for using data obtained from one architecture and algorithmic class to infer informed predictions on a different architecture or algorithmic class.

The approach taken to address architecture differences will be to develop models whereby the characteristics of the architecture (i.e., the processor speed, available memory, cores-per-node, network speed, etc) are used as features in the machine learning model. The result will be a model that can, for example, predict that an algorithm will perform poorly on a new architecture because the memory allocations are to small or because the network is to slow. To test this, the project team will generate the linear solver training data set on a range of different architectures and use it to build a single model whereby the linear solver features as well as the machines internal specifications are used as the features.

To further enable efficient reuse of data we will also investigate the development of Bayesian transfer learning methods. Transfer learning leads to models capable of simultaneously learning from different source domains, making it possible to transfer relevant knowledge from domains with plenty of labeled data to new domains where training data is limited. A good analogy for transfer based learning is that of learning a new programming language, whereby the transfer of core skills from previous languages acts to reduce the learning curve associated with the new language. In the context of SASI, this means using data from a class of algorithms (such as linear solvers) and architecture, to improve prediction in a similar algorithm class or on another architecture. In Phase I we will use existing data for linear solvers, eigensolvers and graph algorithms to build and test models using several classes of Bayesian transfer methods. In particular, we will look at developing a model whereby the learning solvers dataset is used to kickstart the model for eigensolvers (on the initial assumption that the sparse linear features that are predictive of Krylov method convergence are also likely to be predictive of eigensolver convergence).

We discussed these Bayesian methods on the phone last week, and this was my best guess at what that might look like. But, I am a lot thin on the details. Any ideas? BN: I think it's pretty good, I'd keep it at a high level – the actual method application is pretty straightforward, the main question is would it work or not (which we won't know until we try). This is a useful reference: https://arxiv.org/pdf/1801.00857.pdf

4.3 The SASI API

Gluing all these tools as algorithms together will be the primary SASI API. Completion of this interface will require a large literature review to asses the best methods for developing an interface that is applicable to generic numerical algorithms. Currently, the idea is that the interface will require the user to specify a list of functions for tasks that must be completed to utilize the various tools given inside SASI. All interaction between the domain scientist and the internal SASI tools will go through this interface, with every effort being made to minimize the required number of function definitions. If a function definition is not provided, any tools using that method will be unavailable, but other methods will still be accessible. A similar interface has proven to be very succssessful in the SolverSelector API. Work on documentation and examples will be completed throughout the development of the interface as each new feature is added.

5 Performance Schedule and Task Plan

The goal of the Phase I effort will be to provide the reviewers with a clear idea of the scientific and commercial potential of SASI. The research and development topics described in Section 4 will

Time (Months):	Phase I											
	1	2	3	4	5	6	7	8	9	10	11	12
Task 1	X	X	X	X	X	X						
Task 2						X	X	X	X	X		
Task 3	X	X	X	X	X	X	X	X	X	X		
Task 4							X	X	X	X		
Admin											X	X

Figure 1: Overview of task dependencies and time-line.

be addressed by the tasks described in the remainder of this section. Figure 1 summarizes, at a high level, the dependencies among tasks and approximate anticipated task durations. The project duration is roughly divided into 1 month blocks and is 12 months overall, with Phase II proposals being due at the 10 month mark. The final two months (denoted admin in the task-dependencies table) will be dedicated to wrapping up the project, finalizing the documentation, writing the final report and preparing for a Phase II.

RNET would like to present the project ideas and research plan to the DOE Program Manager and other interested scientists interested in performance tuning through smart algorithm selection. This meeting will be scheduled soon after the Phase I contract is awarded. The Kickoff meeting will coincide with the Phase I SBIR PI meeting being hosted by the DOE. The meeting can be hosted at RNET, a DOE site suggested by the Program Manager or via a teleconference. RNET will submit a final report and present the report details along with a Phase II work plan to the DOE program manager and other interested scientists.

5.1 Task 1: Prototype the core SASI framework

In this task, the project team will begin the process of developing the core SASI framework. As discussed above, the framework will be developed based on the results of a comprehensive literature review focusing on the methods and interfaces used to solve a variety of subproblems in many of the more prominent solver packages. At the end of this task, the software interfaces and indirections for building and integration of SASI models will be set in stone with all remaining work simply adding functionality. A comprehensive user manual for SASI and the features therein will also be developed as part of this task.

RNET will take the lead on this task, with UO providing assistance and insight when needed.

5.2 Task 2: Model, Data and Feature set Analysis tools.

In this task, the project team will develop the tools required for verifying the data set, feature set and model as described in Section 4.1. Initial testing of these tools will be completed using the datasets obtained for linear solvers as part of the SolverSelector project. This data provides a good baseline for testing as it has been shown to be effective in informing an accurate SASI based model for linear solvers. The tools developed in this task will also be used in task 4 to verify the suitability of the graph algorithm data.

RNET will take the lead on this task, with UO providing assistance and insight when needed.

5.3 Task 3: Efficient Reuse of Data on new Architectures and Applications.

In this task, the project team will prototype the approaches for the re-purposing training data. As outlined in Section 4.2, this will include an investigation into using data obtained from similar algorithm classes and Bayesian methods to kick-start models for new algorithms. To test these methods, RNET and UO will attempt to kick-start a graph algorithm model utilizing training data obtained from linear solvers and applying it to eigensolvers. The Bayesian transfer models will be considered successful if, at the end of the task, we can use data obtained from linear solvers to improve the accuracy of models for eigensolvers when only a small subset of the data for eigensolvers is used. Additionally, the project team will develop models that use the machines internal capabilities as features in the overall feature set. This will involve collecting training data on a range of machines with various levels of computing prowess. The architecture transfer models developed in this task will be considered successful if, at the end of the project, we can predict the performance of an algorithm for a given matrix on a new architecture with an accuracy of over 90%. This task constitutes the major novel scientific contribution of the proposed project and is expected to form the bulk of the Phase I effort.

RNET and UO will collaborate closley on this task.

5.4 Task 4: Demonstration of SASI using Graph Algorithms

In this task, RNET and UO will demonstrate the capabilities of the SASI framework by applying it to graph algorithms that rely on the matrix representation of graphs. The solver team has tested smart algorithm selection for non-matrix graph algorithms with good results; however, this was a completely manual process and no SolverSelector-like framework was ever built. Using the SASI framework, the project team will recreate the graph-algorithm models, leading to the development of a SolverSelector like API for the increasingly growing set of matrix-based graph algorithms. This will allow the project team to create an informative and instructive proof of concept as to the capabilities of the SASI framework.

RNET will take the lead on this task, with UO assisting with data collection and integration.

@Boyana, does this seem like a good choice for a demonstration (and do you think we even need one at this point) @Ben, yes, it does

6 Related Work

The proposed project is based on the ideas and workflows developed as part of the ongoing SBIR Phase II grant entitled Automated Solver Selection for Nuclear Engineering Simulations (Contract #DESC0013869). Every effort has been made in the report to highlight the new work being proposed that was not funded as part of the Solver selection project, including the development of the numerous machine learning model verification tools, the development of models utilizing architecture specific fetures such as available memory and network speed, and most notably, the development of Baysian transfer learning models designed to encourage efficient data reuse across applications and architectures. Some additional related projects completed by RNET are briefly described below.

6.1 Verification and Validation Toolkit for Nuclear Engineering Simulations

To aid in the verification and validation of numerical simulations, RNET is developing the V&V toolkit. The toolkit will offer a complete set of verification and validation routines designed to rigorously test every aspect of numerical simulation. The toolkit will use the dyninstAPI for binary source code injection to inject the tests into a numerical simulations, producing an external verification binary. The results from running that binary will be automatically parsed by the toolkit to produce a Doxygen style HTML verification report, with the textual components provided both by the specific tests being run and through user specified description files. As such, the V&V toolkit will provide users with a mechanism for developing and maintaining a detailed and living verification document that can be displayed proudly along side any numerical simulation.

6.2 Cloud-based Scientific Workbench for Nuclear Reactor Simulation Life Cycle Management

The predictive modeling approaches and softwares being continually developed and updated by the DOE nuclear engineering scientists (under programs such as NEAMS, CASL, RISMC etc.) need to be efficiently transferred to the nuclear science and engineering community. An advanced workflow management workbench is required to allow efficient usage from small and large business and research groups. The workbench must manage inputs decks, simulation execution (on a local machine, a High Performance Compute cluster, or a Cloud cluster), intermediate results, final results and visualizations, and provenance of the tools and settings. CloudBench is a hosted simulation environment for large scale numeric simulations. CloudBench will augment existing simulation, Integrated Development Environment, and workbench tools being developed by the DOE and industry. It offers a complete set of simulation management features not available in open tools: sharing of configurations, simulation output, and provenance on a per simulation or per project basis; multi-simulation provenance history to allow simulations to be reconstructed, verified, or extended; and remote access to simulation tools installed on Cloud and HPC resources. The portal enables easy adoption of government codes.

6.3 Scaling the PETSc Numerical Library to Petascale Architectures

RNET has developed an extended version of the numerical library PETSc [4] in collaboration with Ohio State University and Argonne National Lab. PETSc is an MPI-based numerical library of linear and nonlinear solvers that is widely used in a variety of scientific domains. With the emergence of multicore processors and heterogeneous accelerators as the building blocks of parallel systems, it is essential to restructure the PETSc code to effectively exploit multi-level parallelism. Changes to the underlying PETSc data structures are required to leverage the multicore nodes and GPGPUs being added to the "cluster architectures".

This project was funded by Department of Energy under the STTR program from August 2010 (Contract Number DE-SC0002434) to May 2013. Dr. P. Sadayappan (OSU) and Dr. Boyana Norris (ANL) have played a key role in this effort by serving as technical advisors. As part of the project, the team has investigated ways for the PETSc library to fully utilize the computing power of future Petascale computers. Novel sparse matrix types, vector types, and preconditioning techniques that are conducive for GPU processing and SIMD parallelization have been integrated

into the PETSc library. The matrix vector operations have been optimized for specific architectures and GPUs by utilizing the autotuning tools.

6.4 A Map-Reduce Like Data-Intensive Processing Framework for Native Data Storage

RNET is currently under a DOE Phase II STTR contract for developing a MapReduce-like data-intensive processing framework for native data storage (Contract#: DE-SC0011312). The Ohio State University (OSU) is a collaborator on this STTR project. MapReduce is a very popular data analytic framework that is widely used in both industry and scientific research. Despite tanalysis applications.

This project is developing a Native data format MapREDuce-like framework, iNFORMER, based on SciMate architecture. The framework allows MapReduce-like applications to be executed over data stored in a native data format, without first loading the data into the framework. This addresses a major limitation of existing MapReduce-like implementations that require the data to be loaded into specialized file systems, e.g., the Hadoop Distributed File System (HDFS). The overheads and additional data management process.

7 Principal Investigator and other Key Personnel

7.1 Ben O'Neill

Ben O'Neill is a Research Scientist at RNET and will be the PI for this project. Ben is a full time employee at RNET and has sufficient time to dedicate to this project. Dr. O'Neill is a Permanent Resident of the United States and a Citizen of New Zealand. Dr. O'Neill, in collaboration with Dr. Boyana Norris, is currently leading the research effort for the ongoing Phase II DOE project (DE-FOA-001490) for the Automated Solver Selection for Nuclear Engineering Simulations described in section 1.1.2. Dr. O'Neill is also the PI for the ongoing Phase I DOE project for the development of verification and validation toolkit for large-scale numerical simulation. Other projects Dr. O'Neill has been involved in include the Phase I DOE Vera workbench project and as the lead developer in the ongoing Phase II DOE project for the development of Cloudbench, a web-enabled interface for remote execution and visualization for nuclear physics tools. His background is in Applied mathematics, high performance computing, and parallel-time integration. His work includes a detailed investigation into parallel-time-integration with MGRIT for nonlinear problems, an enhanced MGRIT algorithm based on Richardson extrapolation and he is also involved in implementing several features currently under development as part of the parallel in time XBraid project.

7.2 Dr. Gerald Sabin, Principal Investigator

Dr. Gerald Sabin, senior researcher at RNET and is a US Citizen. Dr. Sabin is a full time employee of RNET, and has sufficient time to dedicate to project tasks as indicated in the cost proposal. Currently, he is working on several Scientific Computing (HPC) SBIR/STTR projects at RNET. He

is the PI the ongoing Phase II DARPA project developing a linear solver library for graph applications and on the Phase II SBIR project (DE-SC0015748) developing a "Web Infrastructure for Remote Modeling and Simulation of Nuclear Reactors and Fuel Cycle Systems". He is also the PI on the ongoing Phase II DOE project (DE-FOA-001490) for the Automated Solver Selection for Nuclear Engineering Simulations. He has also worked on distributed memory, GPU, multi-core and SIMD optimizations to the Air Force's Kestrel code (DOD Contract#:FA9550-12-C-0028). He has also been the PI on several other related projects including a NASA Phase I project developing SIMD optimizations for Monte Carlo codes (NNX14CA44P), developing parallelization optimizations for PETSc (DOE Contract#: DE-SC0002434), and developing data virtualization support and bitmap indexing for massive Climate Modeling data sets (DOE Contract #:DE-SC0009520).

8 Facilities/Equipment

RNET currently has 9 development computers and a 10-node development cluster that can be used for development and testing in this effort. Each cluster node has two quad-core or hexa-core XEON CPUs, 24-32GB of DRAM, 500+GB of local disk. Two data networks are available, a COTS 1 Gbps Ethernet network and a 10 Gbps Ethernet network. The University of Oregon currently has several clusters connected with high performance networks (such as InfiniBand and 10GigE). The facilities in the CIS Department at University of Oregon including the High Performance Computing infrastructure will be available for this research.

The primary goal of this Phase I project is to develop the framework for developing and using the proposed machine learning based performance tuning models. To that end, RNET and UO has the tools (software and hardware) to evaluate and develop the technologies proposed as part of this Phase I project. However, it is important to note that large-scale testing on a variety of compute resources with a range of core counts, networks and hardware will be a key and important component of the Phase II proposal. The requests for the required resource allocations will be made as part of the Phase II application, but are mentioned here to indicate our desire to use ASCR resources should a Phase II award be granted.

8.1 University of Oregon

University of Oregon (Dr. Boyana Norris) will serve as a subcontractor for this SBIR/STTR. Dr. David O. Conover (Vice President for Research and Innovation, Sponsored Projects Services) is the certifying official for University of Oregon (Address: 5219 University of Oregon, Eugene, OR; Telephone: 541-346-5131; Fax 541-346-5138; Email: sponsoredprojects@uoregon.edu). The budget allocated for University of Oregon is $72K. An official budget and justification from UO are attached to this proposal submission. Dr. Norris will be actively involved in this project by guiding the students, participating in weekly meetings, and providing her expertise as needed. Her CV is attached to this proposal submission.

9 Other Consultants and Subcontractors

None.

References

[1] MOOSE - Open Source Multiphysics . http://mooseframework.org/.

[2] Victor Eijkhout and Erika Fuentes. Machine learning for multi-stage selection of numerical methods. In *Machine Learning*. Intech, 2010.

[3] Paul R. Eller, Jing-Ru C. Cheng, and Robert S. Maier. Dynamic linear solver selection for transient simulations using machine learning on distributed systems. In *IPDPS Workshops*, pages 1915–1924, 2012.

[4] D. Lowell, J. Holewinski J. Godwin, D. Karthik, C. Choudary, A. Mametjanov, B. Norris, G. Sabin, P. Sadayappan, and J. Sarich. Stencil-aware gpu optimization of iterative solvers. *SIAM Journal on Scientific Computing*, 35(5), 2013.