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 are proposing the development of SASI (Smart Algorithm Selection through Inference); a machine learning based toolkit 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 number of different 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 [3]. While the experience and expertise provided by the domain scientists in tuning the algorithms cannot be underestimated, 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. While one could take a guess as to the best linear solver based on coarse grained problem characteristics and experience, it is all but impossible to robustly deter-

mine 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. Moreover, Eller [4] showed 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. Similarly, in applications with dynamic mesh adaptation, runtime 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 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 all numerical subproblems 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 were capable of repeatedly predicting, with an extremely high accuracy (i.e., 98%+), the performance of numerical algorithms and solver configurations across a wide range of applications and domains.

Overall, SASI will be a generic toolkit that guides the user through the process of building and using smart algorithm selection models in modern numerical simulations. While SASI will be portable to any compute architecture (GPU, Cloud FPGA, etc.), the framework developed in this project will target the automatic optimization of numerical algorithms designed for execution on multi-node distributed memory systems ranging on user owned clusters up to leadership class exa-scale machines. SASI provides developers with a mechanism for allowing the problem and architecture to determine the optimal solver, removing the need for time consuming parameter tuning. Ultimately, SASI will allow developers to ship tools that run optimally on any architecture, speeding up development across the entire design pipeline; from early stage testing on single desktops through to final exascale production runs.

1.1.1 Smart Algorithm Selection Through Inference

The SASI framework will guide the user through the process of training, building an using machine learning models targeted at algorithm optimization, automating the process where ever possible. The process of building a SASI model can be split into four distinct components; feature determination, data collection, model building 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:** Perhaps the most difficult component of building a SASI model is that of feature determination and evaluation. 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. world application (see Section 1.1.2).

- Data Collection: 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. For linear solvers, this means solving a large number of matrix-vector systems using a wide variety of linear solvers, preconditioners and input parameters. Likewise, for graph algorithms, the training data set consists of measurements and features obtained by solving a collection of graph problems using a range of graph algorithms. Data collection is the most expensive step in SASI workflow; however, once an initial dataset has been built, it can be reused and extended continuously, quickly making up for the initial cost.

- 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 a a single decision tree or a Random forest can be used to create cheap and highly accurate machine learning models. For instance, the decision tree based J48 algorithm was the most accurate algorithm for the linear solver based SASI models outlined in Section 1.1.2.

- Model Integration. The final component of the SASI workflow is software integration. Integrating SASI into existing simulations is somewhat software dependent. Using a precompiled 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 a 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 primarily during the project teams ongoing Phase II SBIR project entitled Automated Solver Selection for Nuclear Engineering Simulations (Contract #DESC0013869). In that work, the project team developed a robust framework for building and integrating machine learning models for smart algorithm selection of the linear solvers used in nuclear engineering. 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. ly.

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.

The first step in creating a SASI model is to determine the features. Recall that the features are a set of metrics that, in some way or another, infer the performance of the given algorithm on the given computational model. 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 second step in model building is to collect the data. For linear solvers, 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 by scaling the appropriate input parameters. 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].

Training data was obtained by solving the system $A\mathbf{x} = \mathbf{b}$ for each of the matrices in the data sets using a range of linear solver configurations available in PETSc [2]. In each case the RHS was a vector of ones and a random initial guess was used when needed. A solver 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 is 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.

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 Florida Sparse 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 product generated from the Phase II prject is called SolverSelector. SolverSelector is designed to act as an interface between linear solver packages such as PETSc and machine learning toolkits such as TensorFlow. For that reason, SolverSelector was designed to mimic the behaviour of a linear solver. As such, integration of a automatic solver selection into an existing solver package is as simple as implementing the interface and calling SolverSelectors solve function whenever a linear solve is required. SolverSelector has implemented in PETSc as a standalone KSP solver, and has been used successfully used inside PETSc based simulation software such as MOOSE, LibMesh and PROTEUS.

The main research goal of the SolverSelector project was to create a blueprint through which smart algorithm selection models can be built for individual numerical algorithms. 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 been 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. Moreover, included with the API is an interface for integration with PETSc, wherein, the API is integrated as a standalone KSP solver.

1.1.3 The SASI framework

RNET and UO have been working on developing, optimizing and implementing the blueprint for smart algorithm selection in linear solvers for just over three years. That blueprint has proven time and time again 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. Rather it was designed as a plugin for linear solver packages to include smart algorithm selection. Similarly, service contracts offered by RNET for supporting new algorithms and applications are designed such that RNET will design and build the entire system, from determining the features through to gathering the data, building the model and performing software integration; this is simply no way that an industry based company would have the time and resources to otherwise implement a new model.

The SASI framework will be a sophisticated machine learning toolkit designed to make SASI integration feasible for developers across the broad spectrum of numerical simulation algorithms and applications. Overall, the models built using this approach will ensure the efficient usage of large-scale resources (both in terms of time and energy).

The defining design guidelines of SASI will be simplicity and generality. That is to say, 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 hickups 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.

One of the key issues associated with smart algorithm selection models, (and machine learning in general) is that the training data must come from the same domain for which it will be used. That is to say, it is not easy to reuse data from one domain when transitioning to a new domain. This is a particular problem with the smart algorithm selection models because the cost of data collection is so large (i.e, generating the MOOSE training data set required the completion of over 150000 linear solves) and because the data set must be regenerated on each new architecture.

Because SASI will be designed for generic algorithms, addressing these issues will be a key area of research for the Phase I and Phase II projects. Whereas the dataset of the SolverSelector models needs to be regenerated on each new architecture, SASI 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, with applications across all areas of numerical simulation.

An additional benefit of the proposed adaptive models will be that they will equip SASI models with a high level of predictive capabilities. That is to say, users will be able to predict the performance of a solver or algorithm on a new architecture based solely on previous results. The ability to adapt to new architectures and algorithms and to make use of previous simulation results will be of particular importance on exascale resources, where the cost of running exascale simulations combined with the expected high demand for the resources during the initial resources will make it difficult to obtain the exascale compute time required to build accurate SASI models.

In summary, SASI will be a machine learning toolkit designed to optimize the process of building a smart algorithm selection model for any numerical algorithm capable of determining the optimal algorithm and configuration for the given problem and the given compute architecture. To encourage uptake of SASI in numerical simulation, the core SASI framework will be released as Open Source software, with the commercialization plan being to offer support, training, extension and/or integration contracts to government agencies and private companies looking for automated optimization of existing simulation toolkits. Open source, contract based commercialization is quickly becoming the norm in the numerical simulation community, primarily because it appeals to a user-base that almost exclusively deals in free open source software. In return, the results, papers and simulation toolkits released utilizing SASI models act as free, no-cost marketing for SASI, further driving uptake of SASI while also increasing the likelihood of obtaining new service contracts. In addition to this, RNET will also maintain a database of simulation based training data that will be made available at an additional cost to any users looking for data to kickstart a new SASI model.

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 three objectives, RNET will demonstrate the scientific and commercial potential of the proposed package. The investigation into approaches for re-using data as well as the development of automated tools for assessing the validity of the model and data represent the key and novel scientific contributions of the proposed project requiring. Similarly, the development of . While we will be able to reuse some components of the SolverSelector API (i.e., the Sqlite3 database interface), the development of the core the core framework represents a difficult computer science problem that will required an in-depth literature review regarding the best way to implement the framework such that we can support a wide range of numerical algorithms

The Phase II will focus on generalizing and improving the generic machine learning tools proposed in this proposal, including full scale development and testing of the tools developed during Phase I. This will include investigating efficient data collection algorithms and implementations such as a fork-join model to provide fast in-line data collection that efficiently utilizes the available computational resources and an investigation into ranking based machine learning models. Mechanisms and software indirections required for utilizing and managing automatic algorithm selection using implementations from separate and distinct solver packages will also be developed as part of the Phase II effort.

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 of poor performance in SASI models can most often be attributed to; (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. The Phase I effort will focus on the development of two of those tools, each of which is described below:

4.1.1 Feature Sampling

A key component of any machine learning model is an accurate, informative and discriminating feature set capable of inferring knowledge about the given phenomenon. 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 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 programatically, as well as mechanisms for automatically verifying the results.

This automated feature sampling tool will also include an optimization algorithm for optimizing the feature set based on maximizing the accuracy while 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 up to 30 discrete features; hence, a simple brute force search of the parameter space will likely be used 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, Sanitation and Validation

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 database management component built using Sqlite3.

For the most part, the core framework for the sqlite3 database management system developed for SolverSelector is ready for use is SASI; however, there is some requirement for generalizing the schema by which data is stored to the database. As such, the Phase I effort will focus on developing the database sanitation and validation tools.

To address database sanitation, the project team will develop simple interfaces for performing standard machine learing task 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.

SASI will also include a variety of tests for assessing data validity supported by a wide range of statistical metrics. For Phase I, the project team will focus on fixing issues associated with data variability. A narrow data set is the number one 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, it can be very difficult to assess the variability of the method. The Phase I effort will look to address this issue by developing a tool for visualizing and quantifying the variability of a given dataset. This will be achieved by 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 training data is required. 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 even worse when applying SASI to a new model because the results from application area 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), we cannot use the results or data from the old domain in the new domain; instead requiring the user to build a new set of training applications from scratch.

This is the key research issue the project team will need to address for SASI to be a success. To that end, the phase I effort will involve a concerned investigation into methods for using data obtained from data on other architectures and/or from other algorithmic classes to infer informed predictions of algorithm performance on architectures and/or algorithmic classes for which limited training data is available.

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 in addition to the computational model based features. This will allow for the creation of models whereby the features of the machine are incorporated into the prediction, while also adding little to no computational overhead to the overall feature extraction process. 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 do this, the project team will generate the linear solver training data set on a range of different architectures. Using that data from the various machines, the user will create a single model whereby the linear solver features as well as the machines internal specifications are used as the features.

To futher enable efficient reuse of data we will also investigate the development of Baysian machine learning models, specifically Baysian transfer learning methods. A major assumption in the machine learning models used in previous work is that the training data must be in the identical feature space and distribution. That is to say, the training data must come from the same domain. This is less than ideal in situations where limited training data is available because it

prohibits the reuse of existing data in the new domain. In contrast, transfer learning leads to models capable of simultaneously learning form 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, the 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. The Phase I use pre-existing training data sets for linear solvers and eigensolvers to test these new algorithms

4.3 The SASI API

Gluing all these tools as algorithms together will be the primary SASI API.

Several components from the core SolverSelector API will reused in the SASI API. However, completion of this task 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. All in, the SASI API will provide the user with a single place for interacting with and user the numerous SASI based tools.

The goal will be to develop the interface such that it forms the guiding star for users looking to build SASI models for new algorithms and simulations. However, due to the complex and domain dependent nature of SASI models, a comprehensive set of documentation and examples will also be required. 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 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. Specific details are included in the description of each task. Phase II proposals will be due at the of the 10th month, hence, it is our intention to have the technical components of the proposal complete before the end of the 10'th month as that is when Phase II proposal will be due. The final two months (denoted admin) 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.

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.

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: Protype 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 package. 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 implement the approaches for the re-purposing training data obtained on other architectures and/or from alternative algorithms or application areas in new models and for new architectures. As outlined in Section 4.2, this will include an investigation into using data obtained from similar algorithm classes and Baysian 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 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%. The Baysian 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.

This task constitutes the major novel scientific contribution of the proposed project, hence, RNET and UO will collaborate together 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. The solver team has tested smart algorithm selection for 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. 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.

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 a generic API not tied to any simple numerical algorithm, 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. To use the tests in existing simulations, users will simply provide a set of function pointers defining implementations of test dependent tasks that are likely already implemented by the underlying simulation library (i.e., a matrix-vector multiplication or printing a mesh to file) and a xml file defining which tests should be used and on which functions, all of which will be created through a user friendly GUI. Given this, the toolkit will use the dyninstAPI for binary source code injection to inject the tests into an existing binary producing an external verification binary. The verification 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. All in, the V&V toolkit will provide users with a mechanism for developing and maintaining a detailed

and living verification document specifically designed to test each and every component of the given numerical simulation or library.

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 [5] 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. 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, will be the PI on this project 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

@Boyana – I copied this from the MLNEAMS proposal – change as needed

University of Oregon (Dr. Boyana Norris) will serve as a subcontractor for this SBIR/STTR. Dr. Brad Shelton (Interim 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] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2018.

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

[4] 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.

[5] 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.