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 generic numerical algorithms in terms of performance metrics such as CPU-time, memory consumption, energy usage, and resilience. SASI will be a generalization and extension of the project teams ongoing SBIR Phase II SolverSelector project (DOE TODO), developing performance based solver selection algorithms for 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. Recent work in this area includes the development of high performance computing optimizations for the CFD solver, CSD solver, and fluid structure interaction of the Air Force's Kestrel simulation tool. During the project, these optimizations were shared with the Kestrel team and integrated into Kestrel. Dr. Boyana Norris has extensive experience in enabling technologies for high-performance simulations in computational science and engineering. Specifically, her research on performance analysis of scientific codes span 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 performance tuning package for large scale numerical simulation with applications of exascale computing resources.

1.1 Identification and Significance

The modern-day numerical simulation tool-chain consists of an interconnected set of advanced numerical packages and toolkits designed to solve a complex but specific set of numerical subproblems. By allowing developers to focus on their specific area of expertise, the compartmentalization of the numerical simulation pipeline has lead to the rapid development of numerous simulation toolkits with unprecedented predictive capabilities, fidelity and accuracy. 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 generic and specific numerical subproblems across a wide range of applications and on a broad spectrum of 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.

The paradox of such rapid development is that it has quickly become impossible to fully understand the intricacies and details of the numerous subproblem solvers, let alone the mysterious sets of input parameters and configuration options that often accompany advanced numerical tools. 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 nu-

merous algorithms and configurations. Rather, the optimal method is, in practice, determined by experimentation and numerical folklore [**?**]. 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 transient nonlinear partial differential equation. In many cases, the linear solver used to solve these systems is the key computational kernel of the entire simulation, taking place multiple times per nonlinear solve. 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 determine 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 [**?**] 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. A similar argument holds for almost 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. SASI will use machine learning to inform smart runtime algorithm selection and optimization in existing numerical simulations and libraries. The SASI framework will be a generalization 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. In that work, it was 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.

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. The large costs associated with exa-scale computing will make SASI based performance optimizations particularly important on exascale machines. In addition to the aforementioned performance optimization routines, SASI models will give developers an avenue for predicting the performance of numerical algorithms on exa-scale resources. In that case, developers will be able to access the merits of integrating a new solver algorithm or library with little to no overhead. Moreover, SASI will allow developers to optimize simulations based on any measurable metric. The billion way concurrency that will be available on exa-scale is expected to amplify issues related to concurrency, power consumption, resiliency and latency; hence, being

able to optimize simulations based on these metrics will be essential if users are to fully test the limits of this new and exciting technology. 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 The SASI Framework

SASI will guide the user through the process training, building an using machine learning models targeted at algorithm optimization, automating the process where ever possible. As shown in Figure **??**, 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. With regards to SASI, a feature must represent 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.

  Because feature extraction is required during data collection *and* at runtime, it is extremely important that it be as computationally efficient as possible; overly expensive feature calculations can quickly eat up the run-time savings associated with picking the optimal solver if one is not very careful. For linear solvers, metrics such as the number of nonzeros and the absolute trace of the matrix where found to be both cheap and informative, whereas metrics such as the two norm or the spectral radius where to expensive to be used in a real-world application (see Section **??**.

- Data Collection: Also key for an accurate and informative machine learning model is a large and diverse set of training data. 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. The SASI API will include components that simplify the progress of gathering this data, including a SQL based database management.

  It is the accuracy, volume and variability of the training data that ultimately determines the accuracy of the machine learning model. As such, it is essential that the training data collected be analyzed to ensure it is acceptable for use in a machine learning model; it is

all to easy to create a large, but narrow set of training data capable of informing a machine learning model with high levels of accuracy in cross validation tests, but with little to no predictive capabilities in real world situations. As such, SASI will also include a set of database analysis tools designed to access the variability and appropriateness of the training data (see section **??**).

- Model building and Optimization. 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 descision tree or a Random forest can be used to create cheap, serializable and highly accurate machine learning models for SASI based problems. For instance, the decision tree based J48 algorithm was the most accurate algorithm for the matrix-free based linear solver based SASI models outlined in Section **??**.

  Rather than recreating the wheel, SASI will build the machine learing models by interfacing to one (or more) of numerous machine learning frameworks (the python based sci-kit, Facebook's Torch, Google's TensorFlow, the university of Waikato's WEKA, etc) already in existence. In doing so, SASI will provide users with access to a wide range of fully tested and verified machine learning tools. To aid in algorithm selection, SASI will be equipt with tools for automatically building and testing models using standard machine learing based verification methods such as k-fold cross validation and the 66-33% split testing methods. Feature set reduction tools will also be included. These tools will automatically determine the most efficient feature set for which the model still produces a pre-set level of accuracy. Feature set reduction is important as it reduces the computational burden associated with using SASI models, thereby increasing the performance increases that can be obtained.

- Model Integration. The final component of the SASI workflow is software integration. Integrating SASI into existing simulations is somewhat software dependent. For that reason, SASI will implement the software indirections and interfaces required for loading the pre-built model at run-time, and for obtaining a algorithm prediction, but will leave it up to the user to integrate the required software calls into existing software. At runtime, the user will be required to implement three software calls; (1) extract the required features from the computational model and (2) set the algorithm based on the result returned from the the machine learning model. Users will be free to pick and choose when the machine learning model is used to optimize the algorithm (i.e, optimization of a linear solver routine could occur once per Newton iteration, once per time step, or even once per simulation). This more general approach to integration will allow SASI to be integrated into a wide range of software applications, without enforcing restrictions on the formating and style of the intended simulation code.

1.1.2 SolverSelector: Automatic Solver Selection for Linear Solvers

The SASI workflow outlined above is built upon the experience, lessons and work flows developed by the project team as part its ongoing DOE Phase II SBIR project developing automated solver selection algorithms and software for the linear solvers in nuclear engineering applications (RNET and UO, DOE #TODO), as well as for additional models for eigensolvers and for graph algorithms

(UO). In what follows, we provide a brief overview of the methods used and results obtained from our work with linear solvers.

As described above, the first step in the SASI workflow is to determine a set of features for the chosen computational model. For linear solvers, the computational model is represented by the matrix vector system $A\mathbf{x} = \mathbf{b}$. As such, the features represent metrics based around the matrix and the initial guess. Note the the RHS vector b was not examined because, other than potentially creating a nullspace, the RHS has little bearing over the performance of the linear solver. In contrast, the initial guess, or more precisely, the initial residual, $||A\mathbf{x} - \mathbf{b}||$, is a good, albeit somewhat expensive, feature useful for distinguishing between the performance of iterative and direct solvers. Table **??** shows two feature sets that were found to lead to an accurate machine learning model. In both cases, the feature sets were designed such that they maximized the accuracy per computational cost of the feature extraction and solver prediction process. SASI will provide automated mechanisms for determining the optimal feature set based on cross-validation tests that optimize the accuracy of the machine learning models. As part of those proceedures, SASI will also attempt to minimize instances of conflicting classifications. Conflicting classification occurs when two separate matrices have the same feature set (and are thereby identical as far as the model is concerned), but are classified differently for a given solver. Conflicts such as these must be minimized, either by adding new or changing existing features, to ensure the feature set exhibits the discriminating qualities required for use in high accuracy machine learning models.

The second step in the workflow is data collection. To test and build the linear solver based models we used data collected from two sources; the SuiteSparse matrix collection and a set of matrices extracted from the MOOSE testing suite by scaling the appropriate input parameters. The matrices were solved using various configurations of linear solvers and preconditioners available in PETSc with measurements regarding metrics such as CPU time and power consumption being recorded. As an example, the MOOSE dataset contains approximately 50000 independent data points made up of TODO solver configurations applied to TODO different matrices. The size and breadth of these datasets is continuously increasing, with new data points being added on a regular basis.

The SolverSelector API uses binary classification to differentiate between a good and a bad linear solver for a given matrix. A solver is classified as "good" for a given matrix if the measured value (i.e., CPU time) for that solver is within $p \in (0, 1)\%$ of the best solver for the given matrix. In that way, we obtain a list of solvers that are considered to have acceptable performance for a given set of feature values. Choosing p is an important and difficult process that depends heavily on the training set. As p approaches 1, the number of acceptable solvers decreases, increasing the chance that no good solvers are found for a given matrix. Likewise, choosing p to small leads to a large number of acceptable solvers, potentially reducing the performance gains that can be achieved. We have found that setting $p \approx 80\%$ leads to a good balance whereby a small number of solvers are classified as good. However, SASI will provide a more robust mechanism for determining the optimal value of p using a mixture of steepest decent optimization methods and 10-fold cross validation tests.

The downside of a classification based machine learning models is that they do not rank the solvers in order of effectiveness. Rather, the model can only be used to predict the performance of a single solver. This means that for any given feature set, there may be multiple solvers classified as "good". Moreover, the only way to find the list of "good" solvers is to loop through and test every solver configuration available in the database. To address this issue, SolverSelector simply

returns the first "good" solver as the optimal solver, a decision that balances the risk of not getting the very best solver against the costs of testing every solver in the database. The project team has developed a ranking based machine learning algorithm for graph problems; however, the results of those tests are yet to be verified. If successful, ranking based machine learning models will be integrated into the SASI framework in Phase II.

Table **??** shows 10-fold cross validation and 66-33% split validation tests for the machine learning models built using the data sets. Table **??** shows the accuracy. In this case, we show the accuracy obtained from various machine learning algorithms, built using Weka [**?**], based on optimizing for CPU Time using the MOOSE and PETSc based training data. SolverSelector has built in machine learning algorithms utilizing the Waffles machine learning code, but also provides an interface for integration of other third party machine learning toolkits. As you can see, the models obtained for linear solvers where capable of classified "good" solvers as "good" with an accuracy of up to 99% in both the 10-fold cross validation and 33-66% split validation tests.

Table **??** shows some performance gains obtained using the model for run-time algorithm selection inside some of the tests in the MOOSE test suite when compared to the default linear solvers specified by the developers of those tests. It is important to note that these tests are, in most cases, not indicative of a real world application and are likely not highly optimized; however, these results do highlight the fact that the integration of SASI based models is an effective mechanism for the optimization of linear solvers.

The goal of the SolverSelector project was to create a plugin like infrastructure for the automatic detection and selection of the optimal linear solver configuration inside nuclear physics simulations. The realization of that infrastructure is the SolverSelector API. The SolverSelector API, written utilizing C++11 templates, is designed to mimic a standard linear solver routine. To integrate Solver Selector into an existing application, the user must implement a template based interface, wherein the template parameters must represent the matrix and the vector associated with the given application. The user must implement several simple interface functions such as a functions for initializing a "Vector" and performing a matrix vector multiplication. The two most difficult interface functions are the function for setting the solver and the function for extracting the features. As discussed above, it is essential that feature extraction be as efficient as possible, thus, the SASI framework will make no effort to automate or generalize this proceedure, opting instead to provide guidance in the form of documented examples and best practices.

Once SolverSelector has determined the optimal linear solver configuration, the users Solve routine is called. In this routine, the user is given a "Solver" object that specifies the solver name and parameters as specified in the database. It is up to the user to then set up a linear solver to suit those specifications. It is also up to the user to call the SolverSelector API's model prediction functions at the appropriate point in the simulation process. For example, the user could choose to integrate model prediction into the existing solve routine or, to replace the internal solve call with a call to the SolverSelector API's Solve function, which will call a user specified Solve function once model prediction has been completed.

The SolverSelector API has been successfully integrated into PETSc. In that case, we opted to integrate SolverSelector as an additional Krylov subspace solver (KSP). In this way, users can build, validate and use SASI based machine learning models inside PETSc through a single command line parameter and a simple input file. Internally, the SolverSelector KSP extracts the features, predicts the optimal solver, sets up an internal KSP solver, and solves the system. This is done in a completely non-intrusive way requiring no modification of the PETSc source code.

The PETSc interface for SolverSelector supports feature extraction and classification in parallel simulation. Feature extraction is completed in parallel based on PETScs internal domain decomposition. The actual machine learning prediction stage is completed using a single broadcaster to ensure the same solver is used on all processors. As such, the PETSc code uses a single MPI_Reduce call to accumulate features on the root processor, and a single MPI Broadcast step to distribute information about the chosen solver. All of the machine learning models used so far in SolverSelector are deterministic, i.e., the result is fixed for a given input, hence model prediction could be completed in a distributed fashion. However, the benefit of such an approach would likely be mitigated by the additional all-to-all communication needed to gather the feature values on all processors. An MPI_AllReduce style sanity check on the model prediction would likely also be required given the catastrophic nature of attempting to utilize a different solver across different processors.

Currently PETSc versions 3.4 and 3.7 are supported with patches for additional versions available on request. The interface is designed for use with both standard matrix based and matrix-free based iterative and direct solvers, and has been successfully used inside PETSc based simulation software such as MOOSE, LibMesh and PROTEUS.

1.1.3 SASI

In summary, SASI will 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. Overall, the models built using this approach will ensure the efficient usage of large-scale resources (both in terms of time and energy). The predictive capabilities of the model will also reduce the computational burden associated with the development of new algorithms by providing developers with a fast, low cost method for predicting the performance of the algorithm configuration on large-scale machines.

Whereas the SolverSelector API is specifically designed for linear solvers; SASI will be designed as a generic toolkit for the development of smart algorithm detection models. In many ways, the specific interfaces of the SolverSelector API, i.e., the PETSc interface described above, will represent an example of SASI model and interface that can be built using the SASI framework. The SolverSelector project and API is scheduled for completion in July 2019 with the commercialization plan being to sell service contracts for the integration of SolverSelector into existing simulation toolkits, as well as the API itself. Because SolverSelector is specifically designed for linear solvers, SolverSelector will continue to be the tool of choice for that domain.

RNET and UO have been working on SolverSelector for three years, highlighting the difficult nature of building a SASI model from scratch. SASI was born out of the desire to extend the work completed on linear solvers to a more generic audience and to drive the uptake, notoriety and usability of machine learning models for performance tuning in advanced numerical simulation. To that end, our hope is that SASI will address the needs of a large customer base while also increasing sales of the SolverSelector API. For that reason,As such, we believe SASI is best 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.

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 feasibility of the proposed approach, we will develop a proof of concept for the SASI framework as introduced in the previous section.

The requirements being addressed are leveraging the project teams extensive research into automatic solver selection for linear solvers to develop a machine learning framework for automatic performance tuning of both generic numerical algorithms and simulations that is applicable in a broad range application areas. In order to demonstrate feasibility of the proposed approach in Phase I, RNET Technologies, and its subcontractor (UO) will pursue the following objectives.

- Generalize and, where-ever possible, automate, the SolverSelector API to create a SASI framework that guides the user through the process of building and using an accurate and reliable SASI model, including a robust set of machine learning analysis tools that test, analyze and optimize the feature set, data set, and model based on a user specified set of performance metrics.

- Investigation 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. This will include an investigation into implementation and architecture specific features such as the processor speed, available memory and memory access patterns to provide informed predictions when switching to an architecture for which limited training data is available.

- Demonstrate the effectiveness of the SASI framework by applying it to graph algorithms, a numerical subproblem that project team has experience developing SASI models, 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. In particular, the investigation into architecture and implementation features will allow us to demonstrate the capability of SASI models to automatically adapt to not only changes in the computational model, but also to changes in the compute architecture. Likewise, the demonstration of SASI applied to graph algorithms, an area where the project team has prior experience developing SASI models, but for which model building and integration was a manual process, will all the project team to fully highlight the simplicity of the SASI framework, as well as the performance gains that can be achieved using SASI models.

The Phase II will be primarily focused on further 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

4.1 The SASI framework

4.1.1 Feature Set Detection

4.1.2 Efficient Data Extraction and Management of HPC resources

4.2 Model, data and feature set verification and analysis

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

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 **??** 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.

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: Generalization and extension of the SolverSelector Framework

In this task, the project team will begin the process of generalizing the SolverSelector framework for use with generic algorithms and simulations. This will include a large litriture review focusing on the methods and interfaces used to solve a variety of subproblems in many of the more prominent solver packages (i.e., TODO for graph algorithms, TODO for optimization problems, TODO for ODE solvers, etc). At the end of this task, the software interfaces and indirections for building and integration of SASI models will be completed. All remaining work will look to add functionality, verification, analysis and usability features to the overall toolkit

5.2 Task 2: Model Verification Toolkit

In this task, the project team will develop the tools required for model verification, including the feature and data set analysis tools outline in Section **??**. 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.

5.3 Task 3:

In this task, the project team will investigate 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 **??**, 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.

In addition, this task will also include an investigation into architecture and implementation features such as memory access patterns and/or the structure of the memory on the given architecture. RNET and UO will use the data sets previously obtained for linear solvers to assess the applicability of the developed methods. In those tests, training data will be obtained on the variety of small scale compute facilities available to the project team as outlined in Section **??**. The investigation will be considered a success when RNET can predict the performance of a algorithm on a different computer without requiring specific training data obtained on that machine and when existing training data from linear solvers can be used to inform an accurate machine model for graph algorithms. As above, testing will be done existing data obtained for linear solvers and graph algorithms, as well as will new data generated on various complete architecture. An accuracy of 90% for the prediction of solver performance across architectures will be considered a success. An accuracy of 90% was chosen because that is sufficient to kickstart a model on a new architecture while dramatically reducing the costs associated with data collection.

A version of this task will likely extend into any phase II project, with the ultimate goal being to create models that are capable of predicting performance on exascale machines utilizing training data obtained on more moderately sized resources using modest core-counts. As such, the phase II effort will require larger scale testing on terra and peta scale compute resources. Previous research suggests achieving this goal will require a combination of the archtecture and implementation backed features developed here with either analytical or heuristic based performance models (also to be investigated in Phase II).

5.4 Task 4: Development of SASI model for 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.

6 Related Work

RNET has past and current experience in several SBIR/STTR projects on modeling and simulation, high performance computing, and large data formats. The proposed project is an extension and generalization of RNETs Phase II automated linear solver selection project as outlined in Section **??**. Some additional related projects completed by RNET are briefly described below.

6.1 Verification and Validation Toolkit for Nuclear Engineering Simulations

numerical simulation codes being used to design real world nuclear reactors, erroneous simulations can result in design errors that can be extremely expensive to fix, damage the environment, and ultimately result in loss of human life. 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 [1] 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 **??**. 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 (section **??**). 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 $33K. An official budget and justification from UO are attached to this proposal submission. The entire budget for University of Oregon is allocated to the students of Dr. Norris. However, 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] 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.