Automating Sparse Linear Solver Selection with Lighthouse

Kanika Sood\textsuperscript{1}, Pate Motter\textsuperscript{2}, Elizabeth Jessup\textsuperscript{3}, Boyana Norris\textsuperscript{1}

University of Oregon\textsuperscript{1}, University of Colorado Boulder\textsuperscript{2}

Abstract
Solving large, sparse linear systems efficiently is a challenging problem in scientific computing. Accessible, comprehensive, and usable interfaces and tools for high quality code production for this computation are not available. Lighthouse is the first framework that offers an organized taxonomy of software components for linear algebra, providing functionality and performance-based search and generates code templates for optimized one-level kernels. We present the integration of PETSc [2] and Trilinos [3] iterative solvers for sparse linear systems into the Lighthouse framework.

Motivation
- Current HPC linear algebra software is based on years of research.
- The power of these libraries is increasing, but they are becoming harder to use.

Our Solution: Lighthouse
- Is a taxonomy that guides users in finding the appropriate numerical method
- Generates customized code templates from user-specified input and output requirements
- Makes it easier to use advanced libraries
- Uses modern web-based application techniques for a user-friendly experience

Approach
Our approach to selecting sparse linear solvers in Lighthouse proceeds as follows:
1. Create a dataset of preconditioner-solver pair timings and other characteristics using PETSc and Trilinos on a collection of sparse matrices [2].
2. Collect features for each matrix such as its structure, variability across rows and columns, and spectral properties.
3. Reduce the feature set to remove features that do not contribute significantly to the accuracy of the classification.
4. Use a subset of the data as a training set to build classifiers using machine learning methods.
5. Verify the accuracy of the classifiers using the remaining data as the testing set.
6. Use the best classifier to predict well-performing preconditioner-solver pairs for new matrices arising in applications.

Results
- PETSc: BayesNet performs best using Anamorphic-computed features.
- Trilinos: Voting Feature Intervals (VFI) performs best using Trilinos-computed features.

Reduced Feature Sets

| Feature | Reducer Set
|---------|------------------|
| dominant rows | PETSc 1.4
| Total global norms | PETSc 1.4
| Local diagonal dominance | PETSc 1.4
| Diagonal dominance * local variability | PETSc 1.4
| Total one norm % | PETSc 1.4

Performance of the best classifiers for predicting good solvers

<table>
<thead>
<tr>
<th>Data Quality</th>
<th>PETSc with Anamorphic Features</th>
<th>Trilinos with Trilinos Features</th>
<th>Trilinos with Trilinos Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Good Predictors</td>
<td>1853</td>
<td>1795</td>
<td>1936</td>
</tr>
<tr>
<td>Bad Predictors</td>
<td>23</td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

Definitions
- TRF = TPRP, where TPR is the True Positive Rate / Sensitivity, TP is the number of "good" instances correctly classified as "good", and P is the actual number of good instances.
- Classification Algorithms: algorithms that automatically learn how to make accurate estimates of classification predictions based on past observations.
- Good solver: Solver whose true class is within the threshold value of the best.

Overall classifier accuracy

<table>
<thead>
<tr>
<th>All features</th>
<th>PETSc with Anamorphic Features</th>
<th>Trilinos with Trilinos Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS3</td>
<td>85%</td>
<td>75%</td>
</tr>
<tr>
<td>PS1</td>
<td>95%</td>
<td>85%</td>
</tr>
</tbody>
</table>

References

Workflows
- WEKA [4] knowledge flow components used to generate the results.

Conclusions and Future Work
Results to date indicate that machine learning-based classification can produce up to 93% accurate predictions of well-performing sparse linear system solution methods. We also observed that, for PETSc, no single solver configuration was consistent across different solvers. As we expand the set of input problems and solution methods, we expect the accuracy to improve. In future work, we plan to do the following:
- Exploit the Lighthouse taxonomy with more HPC routines and libraries.
- Carry out additional tuning of machine learning algorithm parameters.
- Investigate hierarchical machine learning approaches to provide scalable support for all possible solvers and levels of parallelism.
- Incorporate scalability and hardware resource information into predictions.