

- EigAdept -
The Expert Eigensolver Toolbox

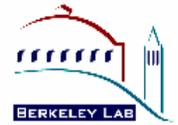
Sherry Li Osni Marques
Lawrence Berkeley National Laboratory

Yeliang Zhang
University of Arizona

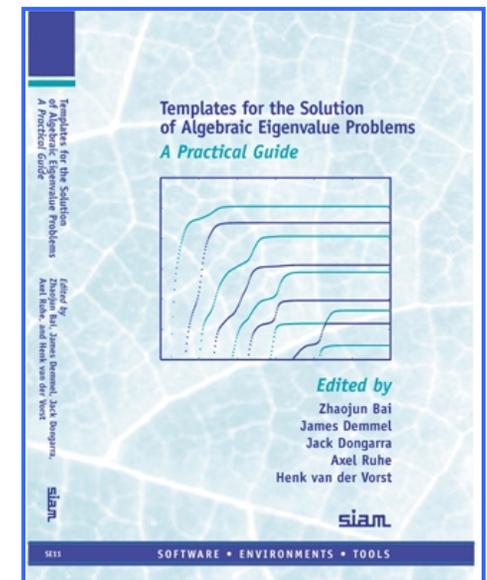
Acknowledgements: Parry Husbands, Konrad Malkowski



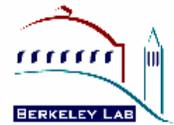
EigAdept: $Ax = \lambda Bx$



- The computation of eigenvalues and eigenvectors is important and time-consuming
- Widely used in multi-domain science
 - Nuclear reactor dynamics (stability of neutron fluxes)
 - Finite element dynamic analysis of structural models
 - Next generation of particle accelerators
 - And more...
- Many numerical eigenvalue solving libraries have been developed
 - Serial
 - Parallel



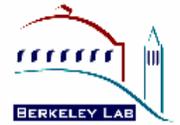
A (Partial) List of Available Eigensolvers



Name	Method	Version	Date	Language	Parallel
ABLEPACK	Adaptive Block Lanczos (ABLE)	1.0	1996	Matlab	
ARNCHEB	Arnoldi Chebychev				
ARPACK	Implicitly Restarted Arnoldi/Lanczos	2	1996	F77	MPI
BLZPACK	Block Lanczos, PO+SO	04/00	2000	F77	MPI
DVDSON	Davidson	-	1995	F77	
EIGIFP	Inverse free precondition Krylov sub proj	2.0	2002	Matlab	
EIGTOOL	GUI for eigs			Matlab	
GUPTRI	Generalized Upper Triangular Form	-	1999	F77	
IETL	Power/RQI, Lanczos-Cullum	2.1	2003	C++	
INSYLAN	Indefinite Symmetric Lanczos	1.0	2000	Matlab	
IRBL	Block Lanczos with Implicit Restart	1.0	2002	Matlab	
JDBSYM	Jacobi-Davidson (symmetric)	0.14	1999	C	
JDQR	Jacobi-Davidson	-	1998	Matlab	
LANCZOS	Lanczos (Cullum, Willoughby)	-	1992	F77	
LANZ	Lanczos, PO	1.0	1991	F77	
LASO	Lanczos	2	1983	F77	
LOBPCG	Preconditioned Conjugate Gradient	4.10	2004	Matlab/C	MPI
LOPSI	Subspace Iteration	1	1981	F77	
MPB	Conjugate Gradient / Davidson	1.4.2	2003	C	
NAPACK	Power Method			F77	
PDACG	Deflation-accel. conjugate gradient	-	2000	F77	MPI
PEIGS	Jacobi (Dense), Inverse Iteration	3.0			MPI
PLANSO	Lanczos, PO	1.0	1997	F77	MPI
PROPACK	Lanczos, PO	-	2001	F77,Matlab	
QMRPACK	Nonsymmetric Lanczos with lookahead	-	1996	F77	
SLEPc	Power/RQI, Subspace, Arnoldi	2.2.1	2004	C/F77	MPI
SPAM	Subspace Projected Approx. Matrix		2001	F90	
SRRIT	Subspace Iteration	1	1997	F77	
SVDPACK	SVD via Lanczos, Ritzit & Trace Minim.	-	1992	F77,C	
SYISDA	Symmetric Invariant Subspace Decom				
TRLAN	Lanczos, dynamic thick-restart	1.0	1999	F90	MPI
Underwood	Block Lanczos		1992	F77	

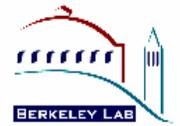
Ref: V. Hernandez et. al., A Survey of Software for Sparse Eigenvalue Problems, SLEPs Technical Report STR-6.

Software Challenges



- No unified software framework.
- A large number of parameters associated with each algorithm.
- Different architectures/applications need different configurations.
- Hard to choose the “best” algorithm for a particular application.

EigAdept: Goals



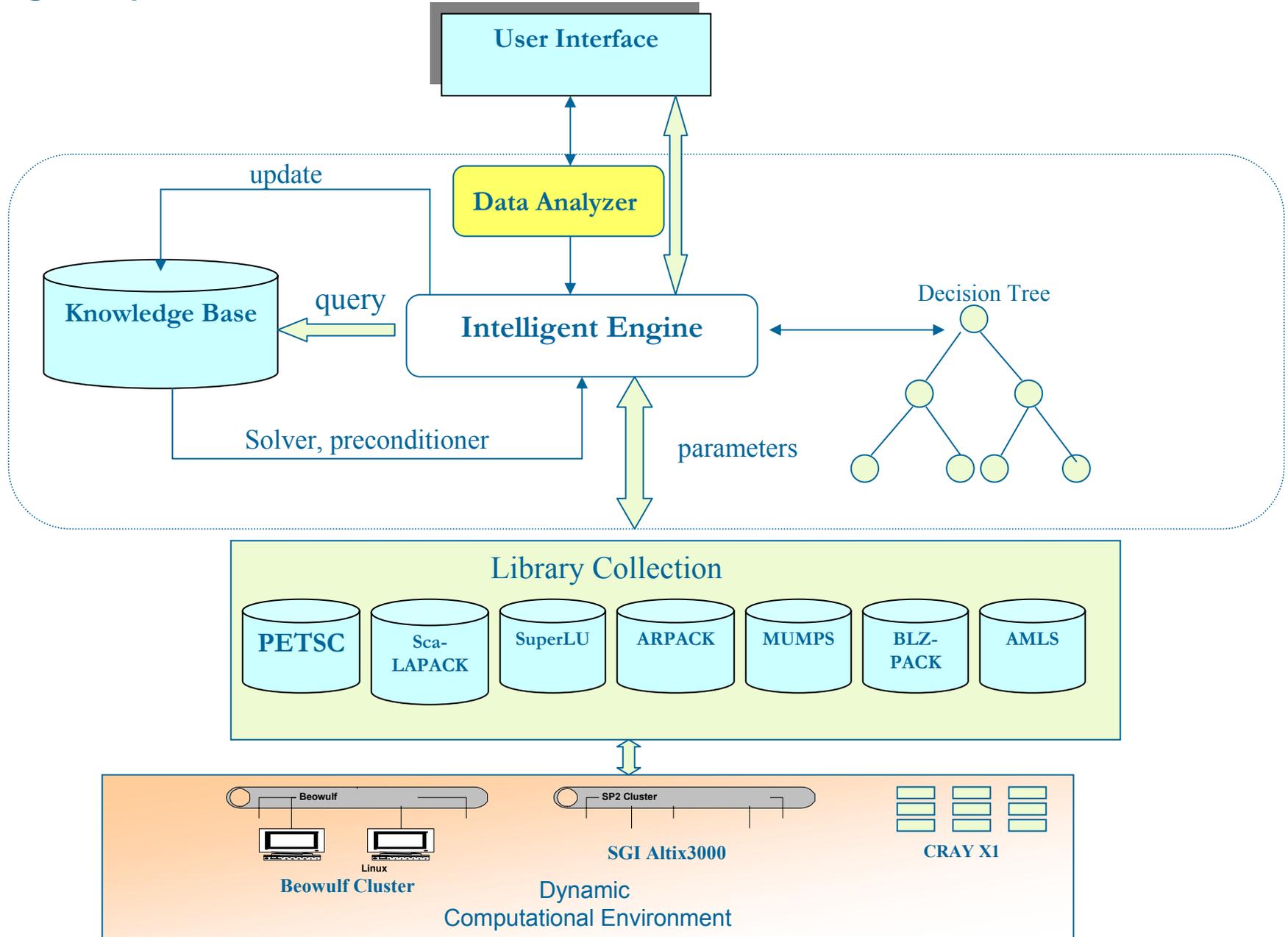
1. Uniform interface

- Easy to use
- Hide unnecessary information

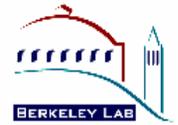
2. Intelligent engine

- Guide user through the maze of different numerical libraries
- Help user to achieve the best performance for his/her application

EigAdept Architecture



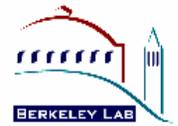
EigAdept: Major Components



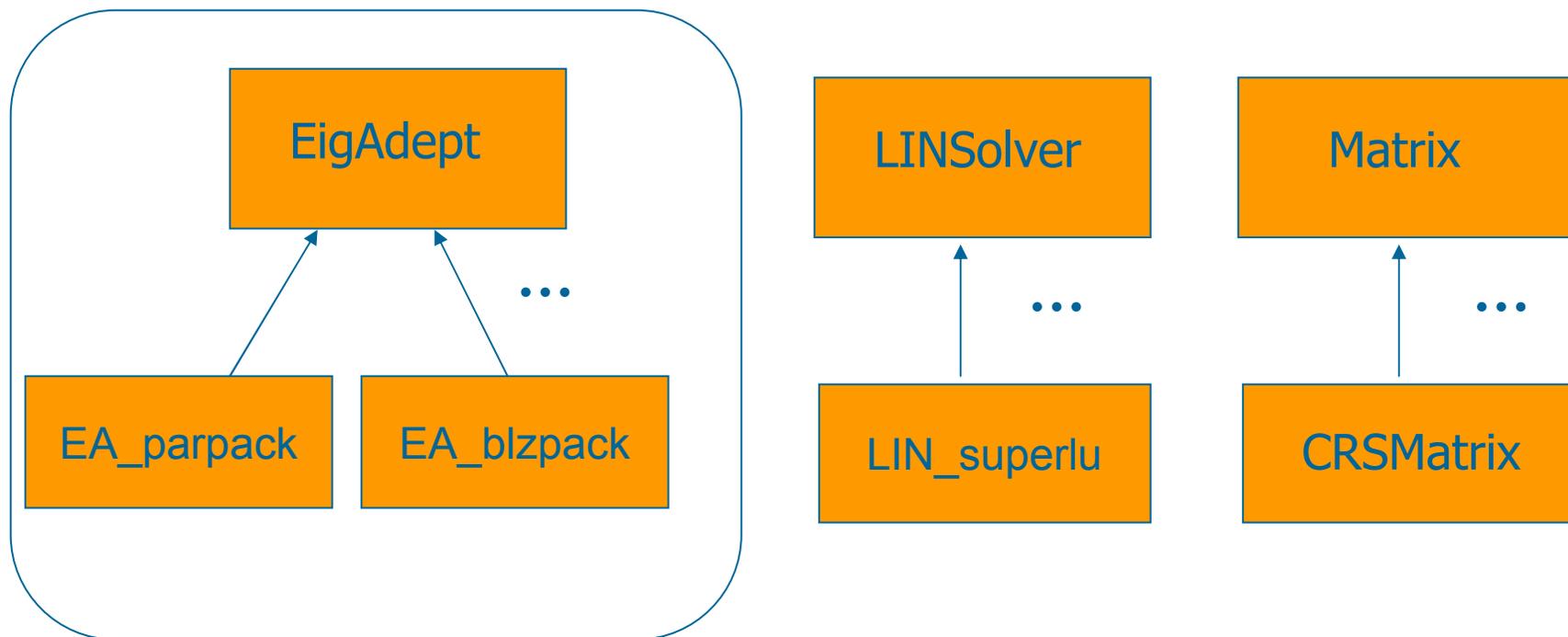
- *User Interface*: unified interface implemented in C++ (callable from other languages)
- *Intelligent Engine*
 - Process data from analyzer, submit to knowledge base, pick the result from knowledge base.
- *Knowledge Base*: Rule + database
 - ◆ Rules are like if-then paradigm
 - ◆ Database is a MySQL relational database
- *Decision Tree*
 - Based on mathematical properties of the problem and approximate spectral information (if available), decision tree could provide recommendations on choosing one of the algorithms.

Ref: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, editors. Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia, 2000.

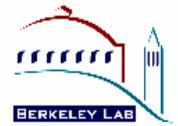
EigAdept: Library Collection



● C++ class hierarchy

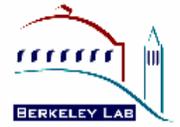


EigAdept: Base Class



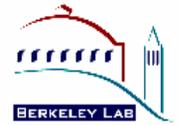
- Mathematical properties
 - Real, complex, symmetric, Hermitian, ...
 - Standard, generalized
 - Matrices A, B
- Desired spectral information
 - Number of eigenvalues/vectors
 - Which part of the spectrum
- Application domain
- Common solver parameters
 - tolerance, maximum iterations, ...
- Common output statistics
- Subclass EA_parpack
 - Parameters specific to PARPACK
- Subclass EA_blzpack
 - Parameters specific to BLZPACK

EigAdept: Extensibility



- New eigensolver can be added in **EigAdept** base class.
- New linear solver can be added in **LINSolver** base class.
- New matrix format can be added in **Matrix** base class.

How to Generate the Knowledge Base?



● Training Set

- Eigenvalue problems collected from different sources.
- Use EigAdept to obtain performance data on these problems and record all the parameters used.
- Generate rules for these problems and store them in the database.

● Feedback Set

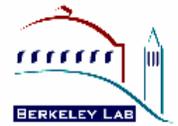
- Initially, the application is solved based on the knowledge of the training set.
- The performance data will be added into the knowledge base as a new rule.
- Provide mechanisms to allow the user change the rules.

EigAdept: Input Parameters



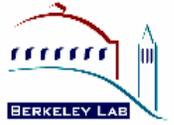
- Automatic:
 - The intelligent engine assigns the default values for the selected library.
- Provided by the user:
 - The user set all library parameters in user's own code.
- Hybrid:
 - The parameters not set by the user use system default.

How to Use EigAdept?



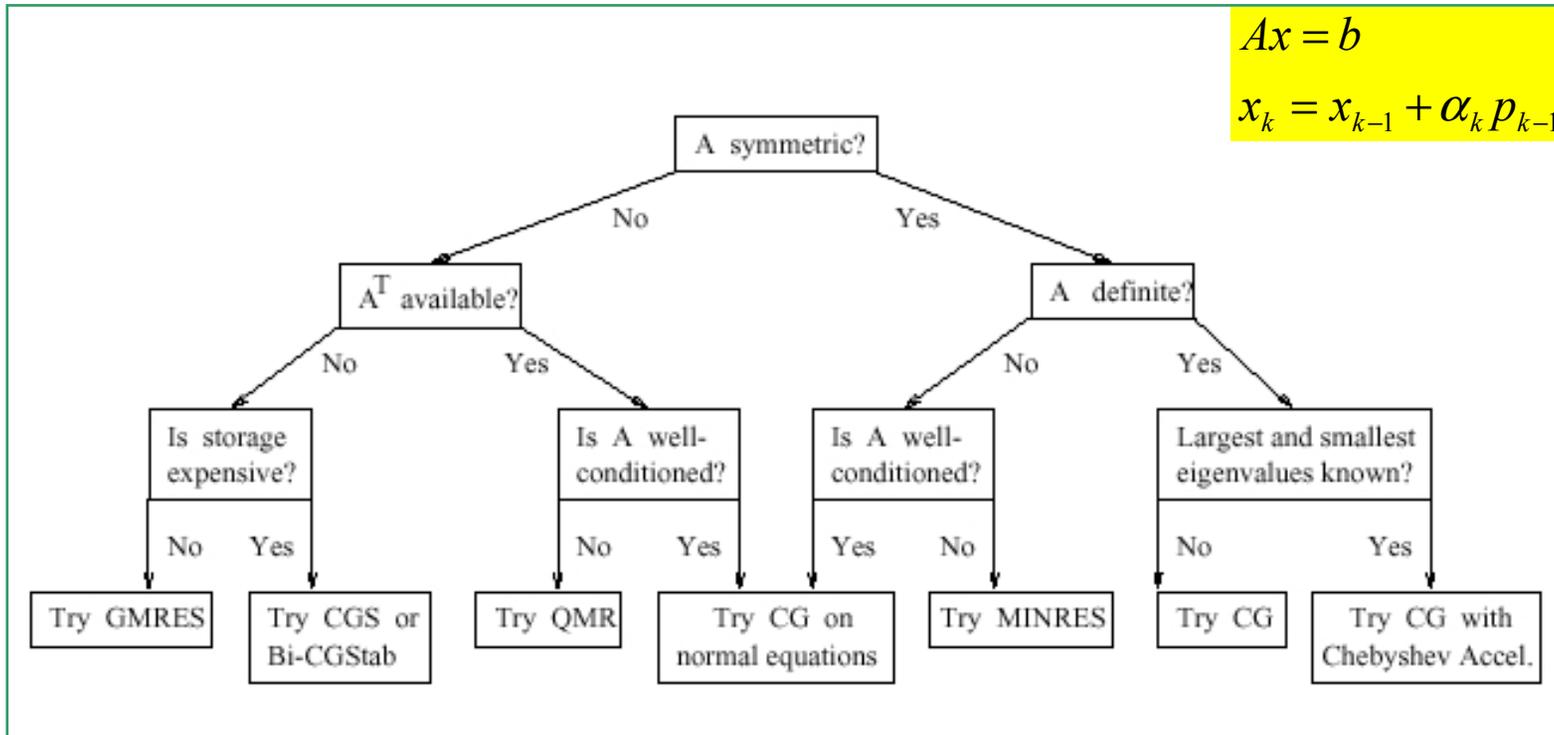
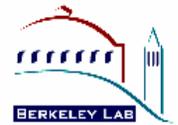
- Simplest: intelligent engine selects solver
EigAdept myeig(A, B);
Call "Set" methods to set system properties
myeig.solve();
- User selects eigensolver: bypass intelligent engine
EA_parpack myeig(A, B);
myeig.setNev(nev); /* "Set" methods to modify parameters */
myeig.solve();
- User selects both eigensolver and linear solver
EA_parpack myeig(A, B);
LIN_superlu mylin(A- σ B);
myeig.solve(mylin);

Case Study: Intelligent Linear Solver



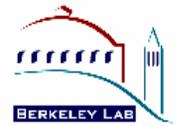
- Why a linear solver?
 - PETSc already provides uniform interfaces to many linear solvers and is widely used.
 - Good exercise for the implementation of the intelligent engine and the knowledge base.

Decision Tree for Case Study



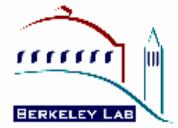
Ref: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, R. Barrett et. al

Database Description



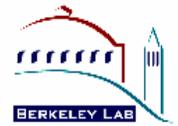
Field	Type
Row	Int
Col	Int
Datatype	Varchar
Property	Varchar
Hermitian	Int
Standard	Int
Solver	Varchar
Preconditioner	Varchar
Execution_time	Double
Sparsity	int

Database Initialization



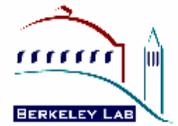
- Training Set
 - 20 matrices obtained from Matrix Market (<http://math.nist.gov/MatrixMarket>)
 - ◆ Reaction-Diffusion
 - ◆ BCS Structural Engineering
 - ◆ Bounded Finline Dielectric Waveguide
- Solving methods: 8 iterative solvers and 4 preconditioners provided by PETSc (32 combinations)
 - Solvers: CG, BiCG, GMRES, BCGS, CGS, TFQMR, CR, LSQR
 - Preconditioners: None, Jacobi, Bjacobi, ASM

Procedure to Choose Solver and Preconditioner



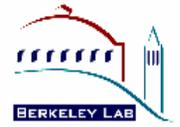
- User submits application with some information regarding the problem using a uniform interface
- The system finds out further information helpful for the knowledge base to decide the solver and preconditioner
- Inquire the database and decision tree to find out the best solver and preconditioner available in PETSc
 - In case knowledge base does not have solver and preconditioner information for current application, a default one is assigned
 - Off-line process verifies a best solver and update the knowledge base accordingly for future use

Performance of Case Study / Intelligent Engine



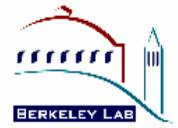
Matrix	Best Time	Worst Time	Percent Failure	Automatic Selection
Rdb32001	2.54 (bicg+none)	22.16 (bcgs+asm)	56.2%	bicg+none
Bcsstk26	1.62 (cr+bjacobi)	11.78 (bicg+none)	43.8%	cr+bjacobi
bfw782b	0.0645 (cr+jacobi)	0.0942 (lsqr+asm)	9.3%	cr+jacobi

Future Work



- Continue building infrastructure
- Build eigensolver decision tree
- Use empirical information to reduce the search space and improve the knowledge base rules
- Architecture awareness (performance tuning)

Knowledge Base



● Utility Class

■ mySQL initiation

- ◆ `mysql_init(MYSQL *mysql)`

■ mySQL connection

- ◆ `mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned long client_flag)`

■ Knowledgebase_act(string, res_info)

- ◆ `mysql_query(MYSQL *mysql, const char *query)`

- ◆ String is used to find best solver in decision tree

- ◆ If no solver is found, string is passed into database to find a solver based our heuristic information

- ◆ The same method is used for off-line database update

- ◆ `res_info` is a data structure carries the return solver package information back to the user program