
Bruce Palmer, William Perkins, Yousu Chen, Shuangshuang Jin, David Callahan, Kevin Glass, Ruisheng Diao, Mark Rice, Stephen Elbert, Mallikarjuna Vallem, Zhenyu (Henry) Huang
The Power Grid
Motivation

- Power grid is enormously complex but is still mostly modeled using serial code
- To fit models onto a single core, numerous approximations and aggregations have been used
- Using parallel code would allow engineers to relax some of the current restrictions in their models but the barrier to creating parallel code is high
Considerations

- The power grid is represented as a graph.
- Graph nodes (buses) and edges (branches) are highly heterogeneous. Different buses and branches can have very different qualities.
- Most models describing the power grid are expressed in terms of non-linear algebraic equations involving complex variables.
- Need to capture a diverse range of models and numerical approaches to cover power grid applications.
GridPACK™ Framework

- Object-oriented design implemented in C++
- Use software templates and inheritance to create application-specific versions of most framework modules
- Hide all communication and minimize the number of parallel concepts that application developers must deal with
- Focus on “local” calculations in application
- Wrap math libraries to separate libraries from the rest of the framework and allow seamless substitution of other libraries in the future
Software Hierarchy

Matrix-Vector Interface

Base Component

Base Bus Component

Application Bus

Base Branch Component

Application Branch

Base Network<AppBus, AppBranch>

Network Parser<AppNetwork>

Base Factory<AppNetwork>

App Factory<AppNetwork>

Mapper<AppNetwork>

Serial IO<AppNetwork>
Network Module

- Templated class that takes application-specific bus and branch objects as template arguments
- Manages partitioning of network and exchange of data between processors
- Assigns internal indices that are used by other framework modules
Schematic Diagram of Network Object

Framework-defined interface

User-defined model
Network Partition
GridPACK™ Mappers

- Construct matrices from contributions from buses and branches
- Manage index transformations between grid location and matrix
- Construct matrices with sensible row partitions based on network partition
GridPACK™ Mappers: Initial Network
GridPACK™ Mappers: Matrix Contributions

No matrix contribution

No matrix contribution

No matrix contribution

No matrix contribution
GridPACK™ Mappers: Matrix Generation

Initial Placement

Final Matrix
Matrix Contributions

- Calculations are “local”
  - Only involve sums over neighboring branches or sums over the two buses at the ends of a branch
  \[ Y_{ii} = -\sum_j Y_{ij} \]

- Each bus/branch evaluates its local block, the mapper builds the matrix in a consistent manner
Math Module

- Wraps a parallel solver library (currently PETSc) and provides high level interface for manipulating matrices and vectors
- Provides interface for setting up distributed matrices and vectors
- Supports basic matrix/operations such as matrix-vector multiply, vector norms, matrix transpose, dot products, etc.
- Supports linear and non-linear solvers and preconditioners
Powerflow Application

1. `typedef BaseNetwork<PFBus,PFBranch> PFNetwork;`
2. `typedef PFFactory<PFNetwork> PFFactory;`
3. `Communicator world;`
4. `shared_ptr<PFNetwork> network(new PFNetwork(world));`

Read in network from external file and partition network

5. `PTI23_parser<PFNetwork> parser(network);`
6. `parser.parse("network.raw");`
7. `network->partition();`

Initialize network components

8. `PFFactory factory(network);`
9. `factory.load();`
10. `factory.setComponents();`
11. `factory.setExchange();`

Evaluate matrix components and create right hand side vector

12. `network->initBusUpdate();`
13. `factory.setYBus();`
14. `factory.setSBus();`
15. `factory.setMode(RHS);`
16. `BusVectorMap<PFNetwork> vMap(network);`
17. `shared_ptr<Vector> PQ = vMap.mapToVector();`

Create Jacobian matrix

18. `factory.setMode(Jacobian);`
19. `FullMatrixMap<PFNetwork> jMap(network);`
20. `shared_ptr<Matrix> J = jMap.mapToMatrix();`
21. `shared_ptr<Vector> X(PQ->clone());`
26 double tolerance = 1.0e-6;
27 int max_iteration = 20;
28 ComplexType tol;
29 LinearSolver solver(*J);
30
31 int iter = 0;
32
33 solver.solve(*PQ, *X);
34 tol = PQ->normInfinity();
35
36 while (real(tol) > tolerance && iter < max_iteration) {
    factory.setMode(RHS);
    vMap.mapToBus(X);
    network->updateBuses();
    vMap.mapToVector(PQ);
    factory.setMode(Jacobian);
    jMap.mapToMatrix(J);
    solver.solve(*PQ,*X);
    tol = PQ->normInfinity();
    iter++;
37 }
Component Hierarchy

Matrix-Vector Interface

Base Component

Base Bus Component

Base Branch Component

Y-matrix Bus

Y-matrix Branch

Powerflow Bus

Powerflow Branch
Powerflow Scaling for Artificial 777646 Bus Network

![Graph showing the relationship between Time (seconds) and Number of Processors for different stages: Parsing, Partitioning, Solver, and Total. The graph demonstrates a decrease in time as the number of processors increases.](attachment://image.png)
Dynamic Simulation

Simulation of 16351 bus WECC network
Current Activities

- Development of object-oriented Fortran 2003 interface
- Development of more generalized matrix-vector interface to support applications where dependent and independent variables are associated with both buses and branches (not just buses)
- Investigating new methods for distributing data to the network (distributed hashing algorithms)
Conclusions

- A software framework for developing parallel power grid applications has been developed.
- Several different types of power grid applications have been developed using the framework. These applications demonstrate parallel speedup.
- GridPACK™ is available for download at https://gridpack.org.
- Contact bruce.palmer@pnnl.gov.
Acknowledgments

- This work is supported by the U.S. Department of Energy (DOE) through its Advanced Grid Modeling Program.
- Computing resources were provided by Pacific Northwest National Laboratory through its PNNL Institutional Computing program
- GridPACK™ is available for download at https://gridpack.org