GridPACK™ Toolkit for Developing Power Grid Simulations on High Performance Computing Platforms

Bruce Palmer
GridPACK™ Development Team

- Bruce Palmer (PI): Parallel code development
- William Perkins: Parallel code development
- Yousu Chen: Power grid application development
- Shuangshuang Jin: Power grid application development
- David Callahan: Data integration
- Kevin Glass: Data integration and optimization
- Ruisheng Diao: Power grid engineering and model validation
- Stephen Elbert: Optimization and economic modeling
- Mallikarjuna Vallem: Synthetic data and model validation
- Nathan Tenney: Automatic builds and testing
- Kevin Lai: Webpage development
- Zhenyu (Henry) Huang: Program management
Why GridPACK™?

- The power grid, despite its size and complexity, is still being modeled primarily using workstations.
- Serial codes are limited by memory and processor speed and this limits the size and complexity of existing models.
- Modeling large systems using small computers involves substantial aggregation and approximations.
- Parallel computing can potentially increase memory and computing power by orders of magnitude, thereby increasing the size and complexity of power grid models that can be simulated using computing.
- Parallel computing is more complex than writing serial code and the investment costs are relatively high.
- Parallel software is a rapidly changing field and keeping up with new developments can be both expensive and time-consuming.
Objectives

- Simplify development of HPC codes for simulating power grid
- Create high level abstractions for common programming motifs in power grid applications
- Encapsulate high performance math libraries and make these available for power grid simulations
- Promote reuse of power grid software components in multiple applications to reduce development and maintenance costs
- Incorporate as much communication and indexing calculations as possible into high level abstractions to reduce application development complexity
- Compartmentalize functionality to reduce maintenance and development costs
Impact

- Access to larger computers with more memory and processing power
- Models containing larger networks and higher levels of detail can be simulated
- Reduced time to solution
- Greater capacity for modeling contingencies and quantifying uncertainty
Contributing to GridPACK™

- GridPACK™ is open-source, releases include all source files
- BSD license allows users to incorporate GridPACK™ in their software, both proprietary and open-source
- Development tree will be available via a public server in the late summer to fall time frame (most likely via GitHub)
- Files can be contributed in the meantime by getting in touch with a member of the development team
GridPACK™ Framework

GridPACK™ Applications

Application Driver
- Base Factory
  - Network-wide Operations
- Application Factory
- Base Network Components
  - Neighbor Lists
  - Matrix Elements

Application Components
- Y-matrix
- Powerflow
- Dynamic Simulation

GridPACK™ Framework

Import Module
- PTI Formats
- Dictionary

Task Manager

Network Module
- Exchanges
- Partitioning

Configure Module
- XML

Math and Solver Module
- PETSc

Mapper

Export Module
- Serial IO
- PTI Formats

Utilities
- Errors
- Profiling

Core Data Objects

Matrices and Vectors

Power Grid Network
Partitioning of Network

WECC (Western Electricity Coordinating Council) network partitioned between 16 processors
Matrix Contributions from Network Components

No matrix contribution

No matrix contribution

No matrix contribution
Distribute Component Contributions and Eliminate Gaps
Powerflow Jacobian from Mapper
(1 Processor)

16351 bus
WECC system
Powerflow Jacobian from Mapper (4 Processor)

16351 bus
WECC system
Powerflow Jacobian from Mapper (16 Processor)

16351 bus WECC system
typedef BaseNetwork<PFBus,PFBranch> PFNetwork;

Communicator world;

shared_ptr<PFNetwork> network(new PFNetwork(world));

PTI23_parser<PFNetwork> parser(network);
parser.parse("network.raw");

network->partition();

PFFactory factory(network);
factory.load();
factory.setComponents();
factory.setExchange();

network->initBusUpdate();
factory.setYBus();
factory.setMode(YBus);

FullMatrixMap<PFNetwork> mMap(network);
shared_ptr<Matrix> Y = mMap.mapToMatrix();

factory.setSBus();
factory.setMode(RHS);
BusVectorMap<PFNetwork> vMap(network);
shared_ptr<Vector> PQ = vMap.mapToVector();

factory.setMode(Jacobian);
FullMatrixMap<PFNetwork> jMap(network);
shared_ptr<Matrix> J = jMap.mapToMatrix();

shared_ptr<Vector> X(PQ->clone());

double tolerance = 1.0e-6;
int max_iteration = 100;
ComplexType tol = 2.0*tolerance;
LinearSolver solver(*J);

int iter = 0;

// Solve matrix equation J*X = PQ
solver.solve(*PQ, *X);
tol = X->norm2();

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 = X->normInfinity();
  iter++;
}

Proudly Operated by Battelle Since 1965
Performance Results

Applications
- Powerflow
- Dynamic Simulation
- Dynamic Contingency Analysis

Strong Scaling Performance
- Fixed problem size, increasing number of processors
Powerflow Scaling for Artificial 777646 Bus Network

![Graph showing the relationship between time and the number of processors. The graph includes lines for Parsing, Partitioning, Solver, and Total time.](image)
Dynamic Simulation

Simulation of 16351 bus WECC network

Number of Processors

Time (seconds)
Dynamic Contingency Analysis

Simulation of 16 contingencies on 16351 bus WECC network

2 levels of parallelism

Number of Processors

Time (seconds)
Conclusions

- A flexible framework for developing power grid applications that run on advanced computer architectures has been developed.

- The framework supports most of the basic data structures and data manipulations common to many power grid applications.

- Several power grid applications have been developed within the framework and show scaling behavior on multiple processors.

- Documentation and downloads for GridPACK™ are available at https://gridpack.org.
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