IEEE PES hpcGrid WG Webinar
July 16, 2018

GridPACK™: Framework and Library for Accelerating HPC in Grid Applications

Bruce Palmer, William Perkins, Yousu Chen, Renke Huang, Zhenyu (Henry) Huang

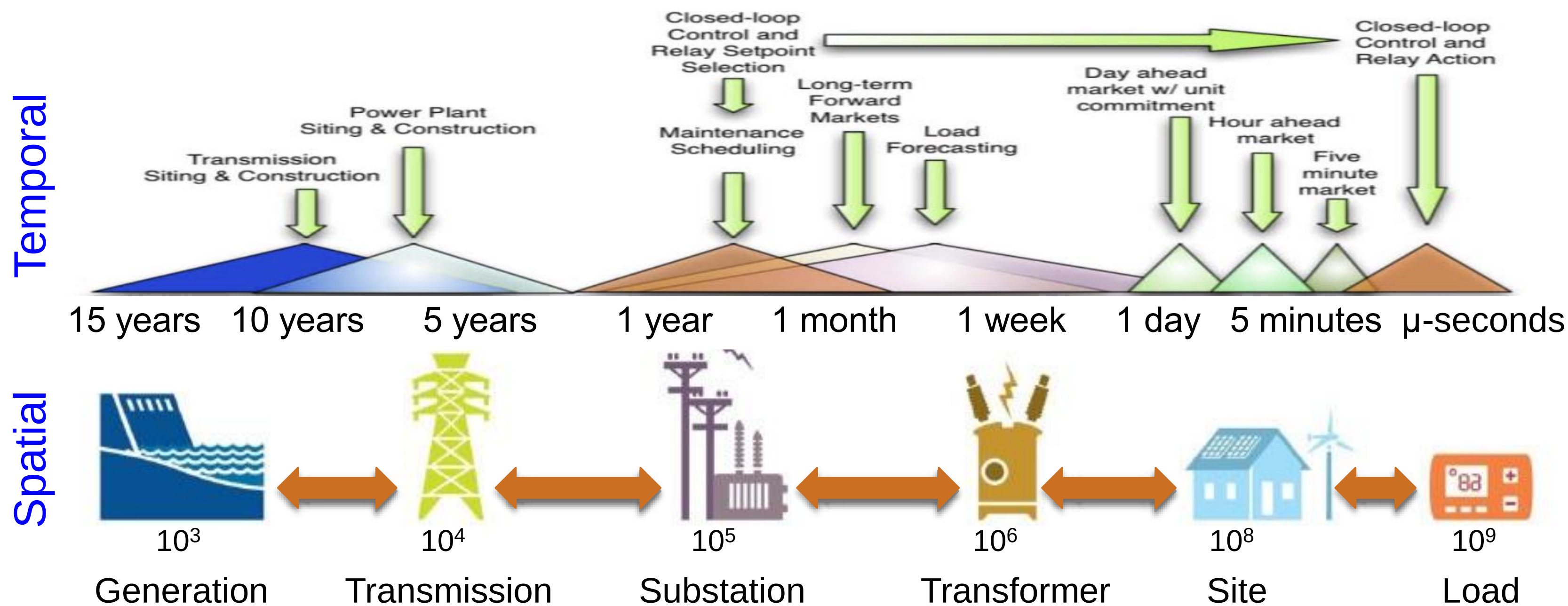
Pacific Northwest
NATIONAL LABORATORY

Acknowledgement

- Webinar Sponsor:
  - IEEE Power and Energy Society Working Group on High Performance computing for Grid Applications (hpcGrid WG)
- Funding Sponsors:
  - US Department of Energy (DOE) Office of Electricity (OE) Advanced Grid Modeling (AGM) Program
  - US Department of Energy (DOE) Grid Modernization Laboratory Consortium (GMLC)
  - US Department of Energy (DOE) Office of Advanced computing Scientific Research (ASCR)
  - US Department of Energy (DOE) Advanced Research Program Agency – Energy (ARPA-E)
  - Bonneville Power Administration (BPA) Technology Innovation Program

Math and computing challenges in modeling and simulation of the future grid

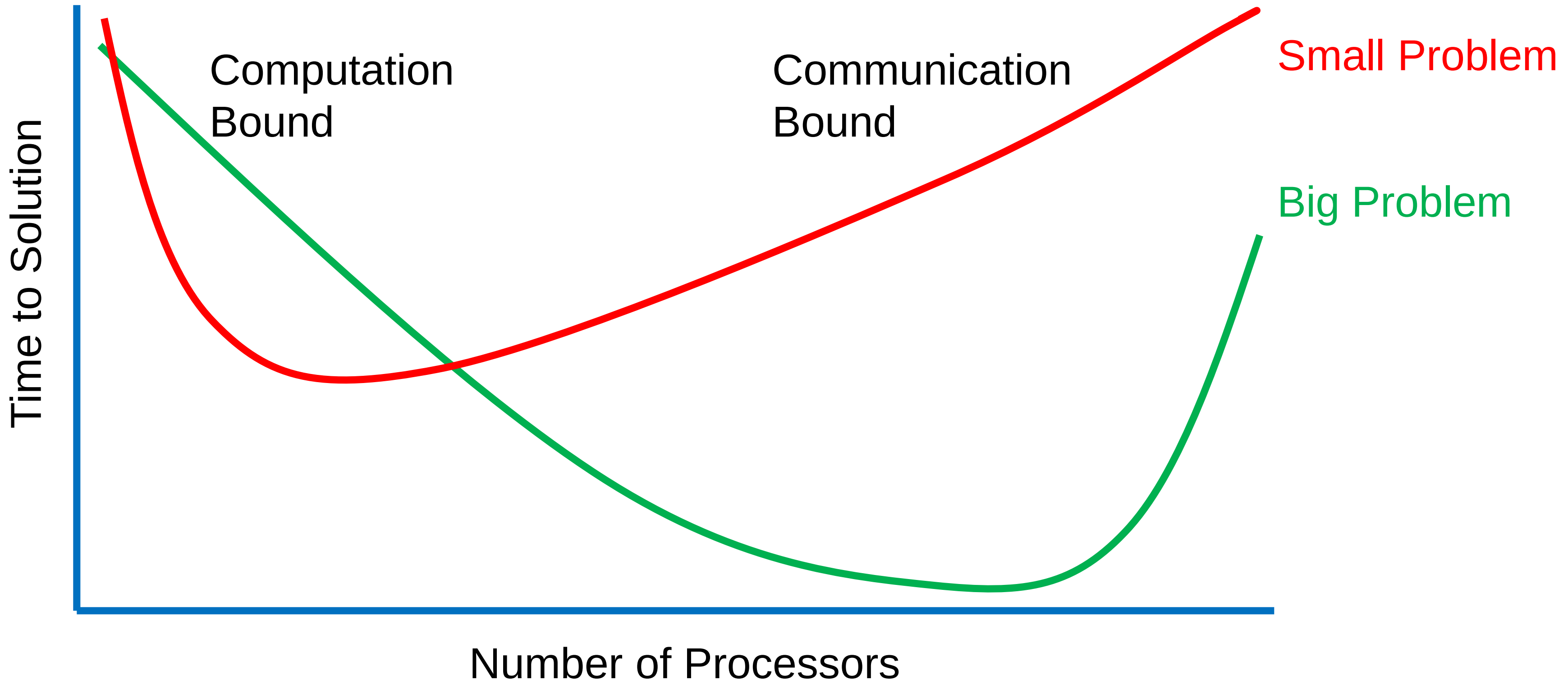
- Multi-scale spatio-temporal modeling and simulation with stochasticity
  - From micro-second to decades
  - From 10^3 generators nodes to 10^9 end-use devices
- Large-scale data assimilation for state and parameter calibration
  - Petabyte data/year from high-speed sensors and smart meters.
- Modeling of multi-system dynamics and dependency
  - Grid, buildings, communication, gas pipelines, weather/wind/solar, water



Why use High Performance Computing (HPC) for the Power Grid?

- The power grid is growing more complex
  - More renewables
  - Smart grid technology
- Serial codes are no longer enough
  - Simulation of larger and more complex models
  - Reduced time to solution for operations
  - Evaluation of thousands (N-1) or millions (N-2) of contingencies
  - Large scale optimization

Does Parallel Computing Help?

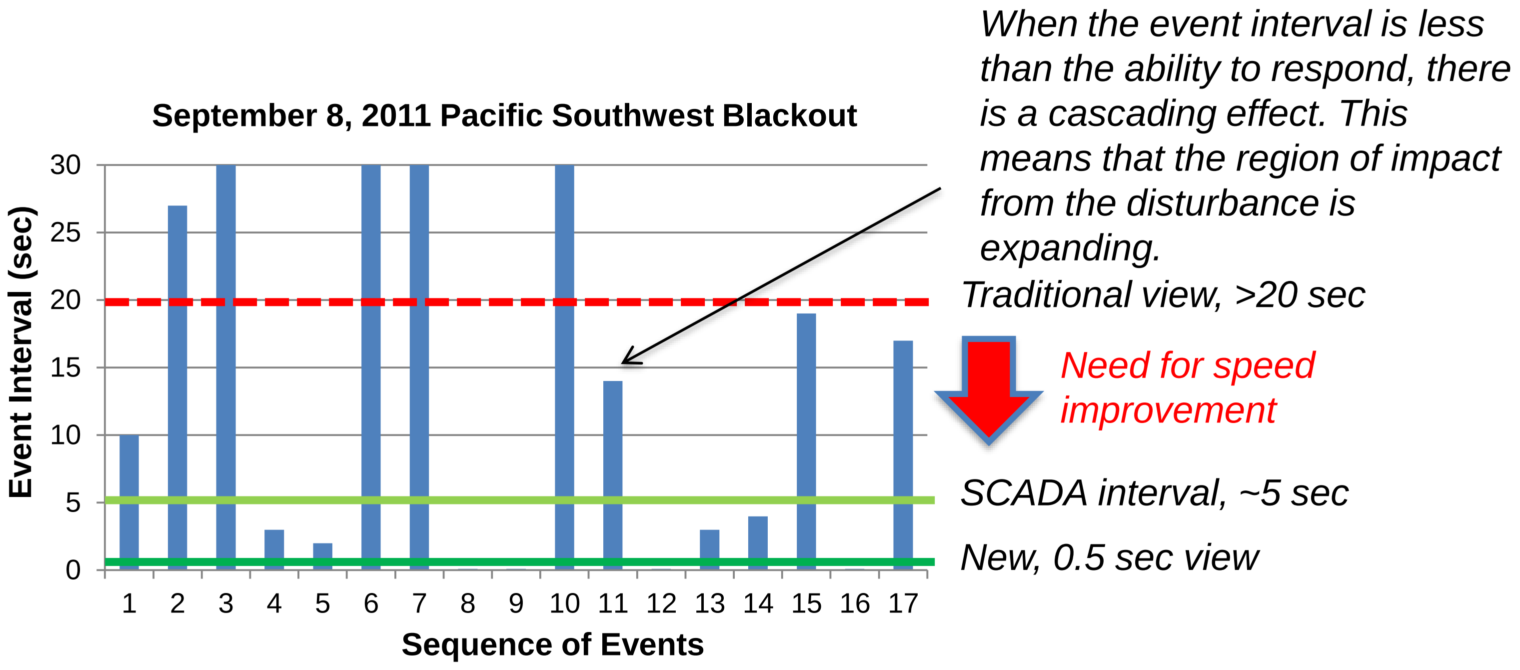


Power Grid Computing Challenges

- Faster time to solution for operations
- Large number of scenarios to account for uncertainties and increasing variability
- More complicated algorithms and more detailed models
- Optimization problems are increasing in size and complexity
- Significant gap in program complexity in going from serial to parallel code

Example Success: Fast State Estimation captures the changes and offers an opportunity to stop cascading

- For the first time, the core function in control rooms – State Estimation – can run at a unprecedented 0.5s speed (>40x faster).



Example Success: Visual analytics of massive contingency analyses for real-time decision support

Contingency	# of scenarios	Serial on 1 core	Parallel on 512 cores	Parallel on 10,240 cores
WECC N-1 (full)	20,000	39 minutes	4.8 seconds	
WECC N-2 (partial)	1,000,000	68.5 hours	8 minutes (511x speedup)	25 seconds (9871x speedup)
ERCOT	1,000,000	~4 hours (estimated)	~0.5 minute (estimated)	<2 seconds (estimated)
Eastern Interconnect	1,000,000	~1100 hours (estimated)	~128 minutes (estimated)	~400 seconds (estimated)

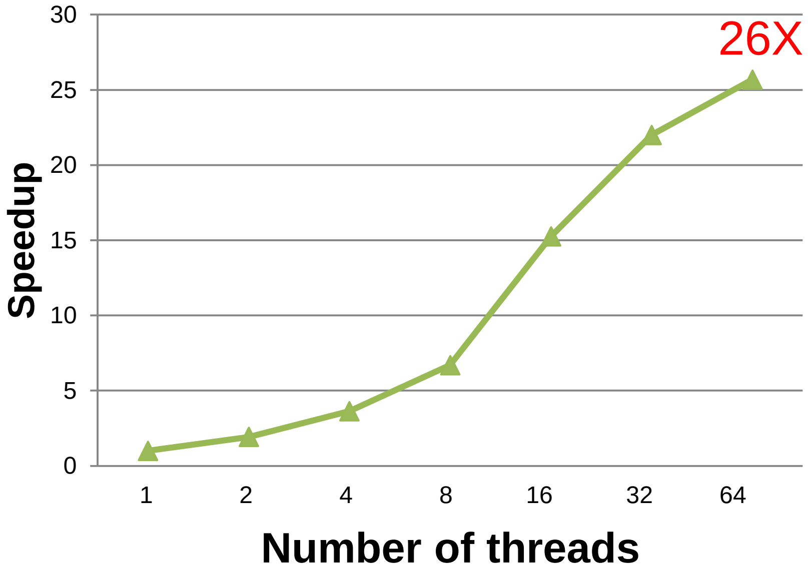
- Easy-to-interpret visualization with prioritized concerns & recommendations
- Operators reported 30% improvement in emergency response

Current tabular format presents data, not information

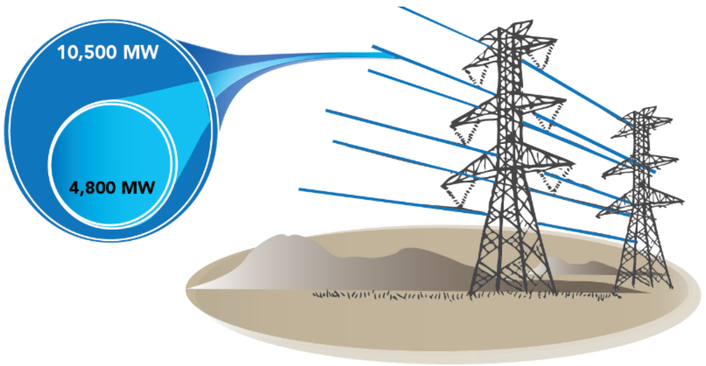
New visualization tool converts data to actionable info

Example Success: Real-time path rating through fast computation to manage transmission congestion

- Look-Ahead Dynamic Simulation
  - 16,000-bus w/ simplification
  - 9 sec for 30-sec simulation
  - 13X faster than today's commercial tools



- Real-Time Path Rating in 10 min
  - Significant congestion cost: NYISO $1.1B/2010; PJM $1.4B/2012
  - Transmission expansion?
  - Realistic ratings: +1000 MW = +$240M/year
  - Avoid renewable curtailment



tap into unused capacities

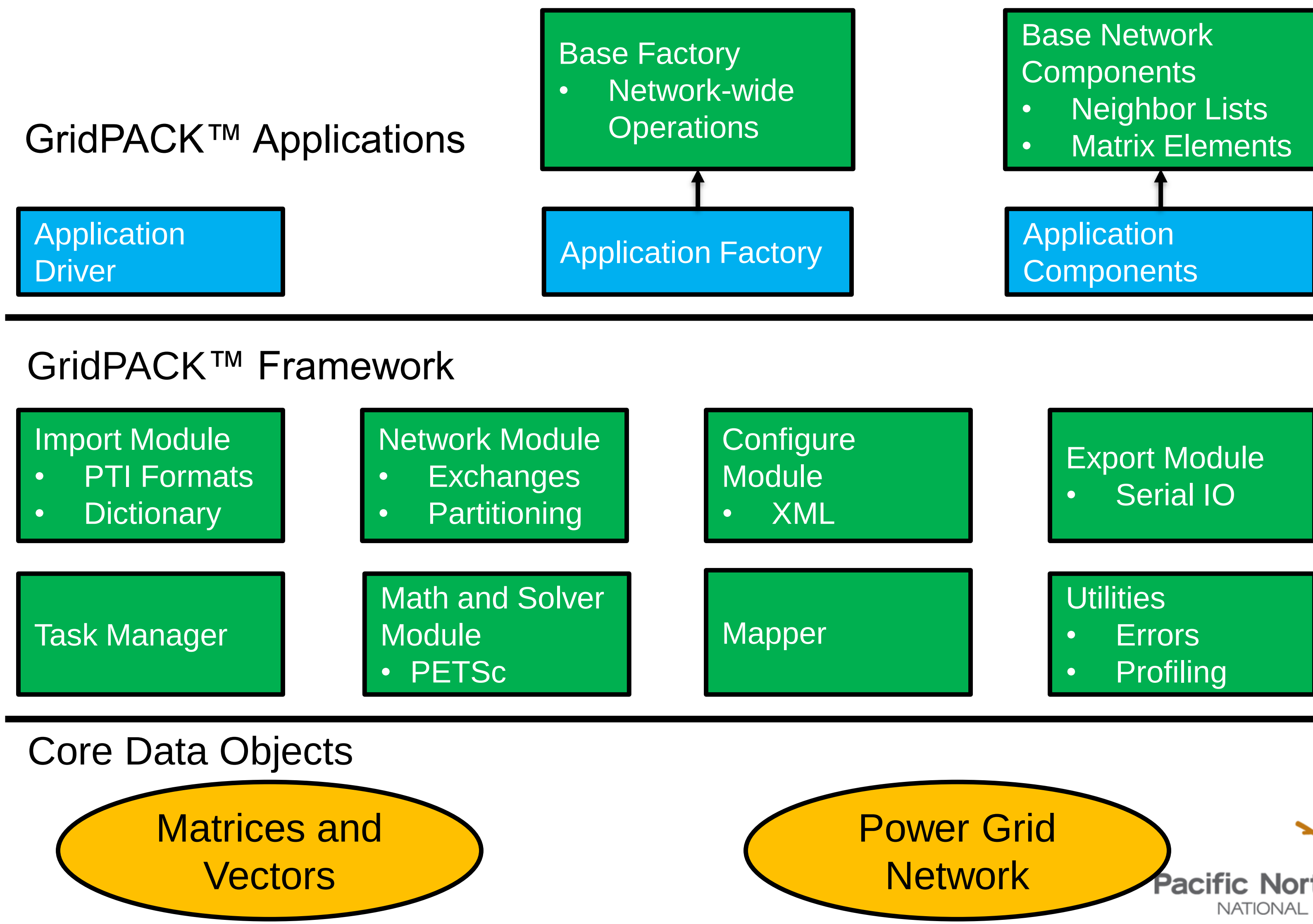
GridPACK Approach

- Lower the threshold for development of HPC codes
- Create high level abstractions for common programming motifs
- Encapsulate high performance math libraries
- Compartmentalize functionality and promote reuse of software components
- Hide communication and indexing

GridPACK Framework

- GridPACK is written in C++ and is highly customizable
  - Inheritance
  - Software templates
- Runs on Linux-based clusters and workstations.
- Wide variety of solvers and parallel linear algebra available through PETSc suite of software.
- Prebuilt modules
  - Power flow
  - State estimation
  - Dynamic simulation
  - Dynamic state estimation (Kalman filter)
- Robust support for task-based execution.

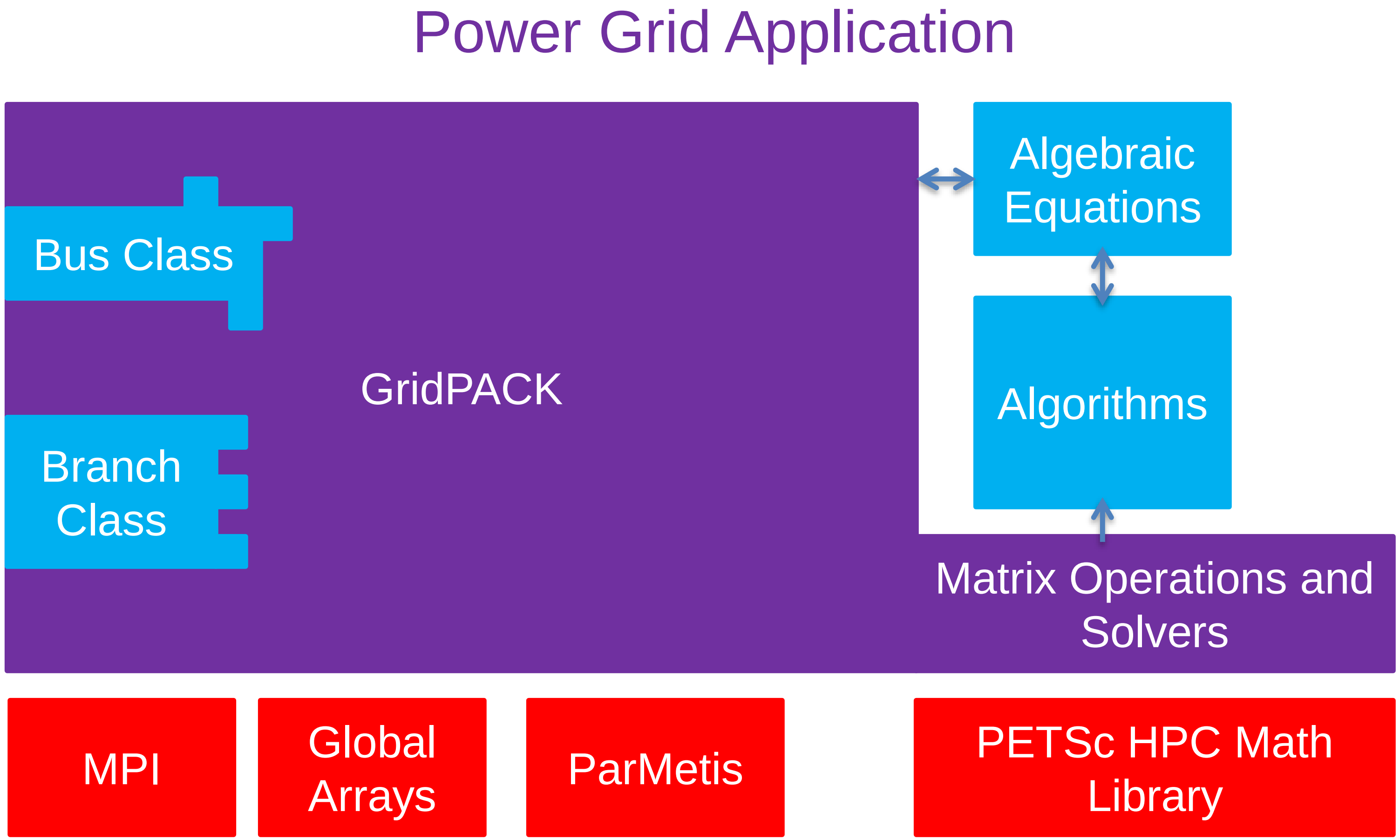
GridPACK Core Framework



GridPACK Functionality

- What you supply
  - Bus and branch classes that define your power system application
  - High level application driver describing solution procedure
- What you get
  - Parallel network distribution and setup
  - Data exchanges between processors
  - Parallel matrix builds and projections
  - I/O of distributed data
  - Parallel solvers and linear algebra operations
  - Distributed task management
  - Application modules for use in more complex workflows

GridPACK Application

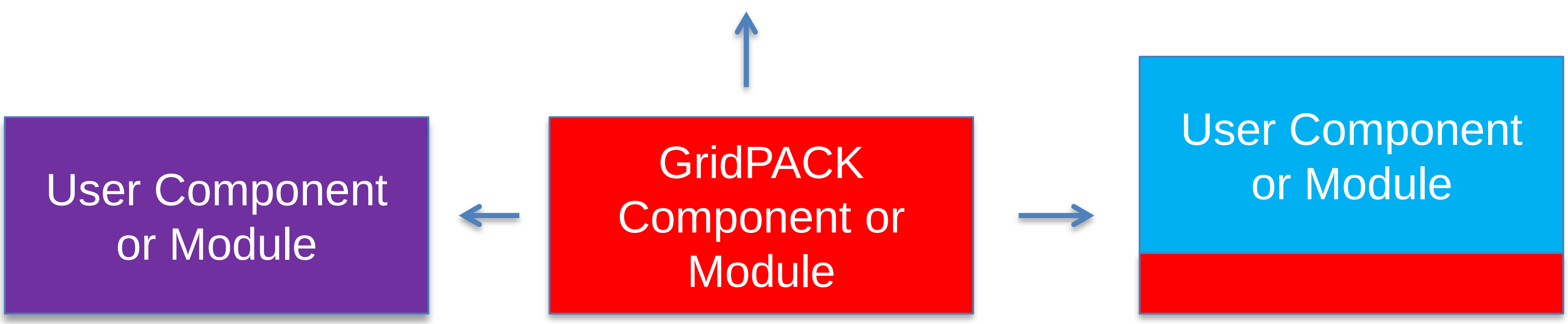


Software Reuse

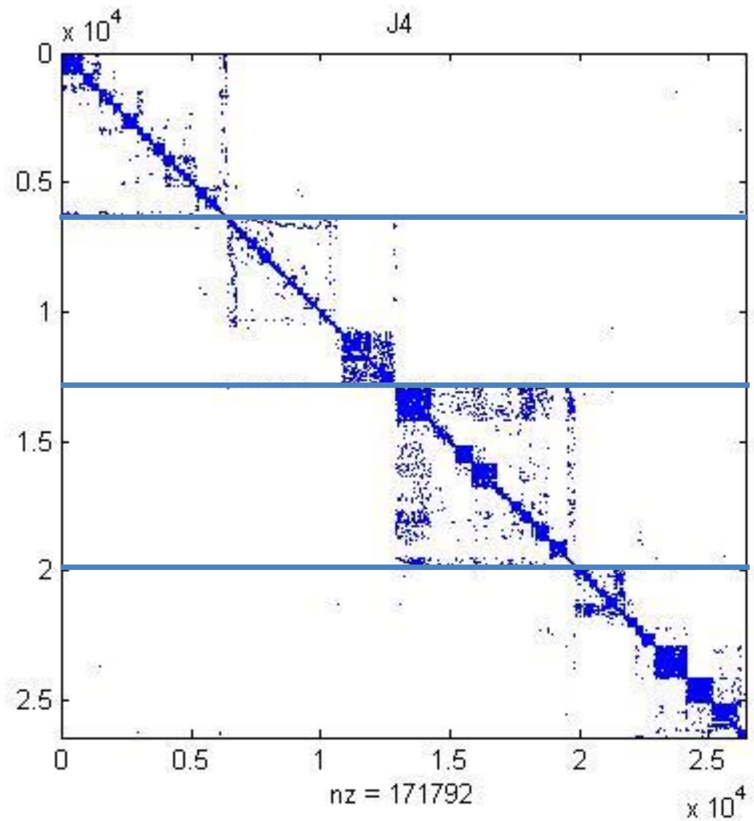
Rename and modify existing GridPACK component or module to create a new application

Use GridPACK components and/or modules as is

Inherit from GridPACK component or module to create a new application



Distributed Power Flow Jacobian from Mapper

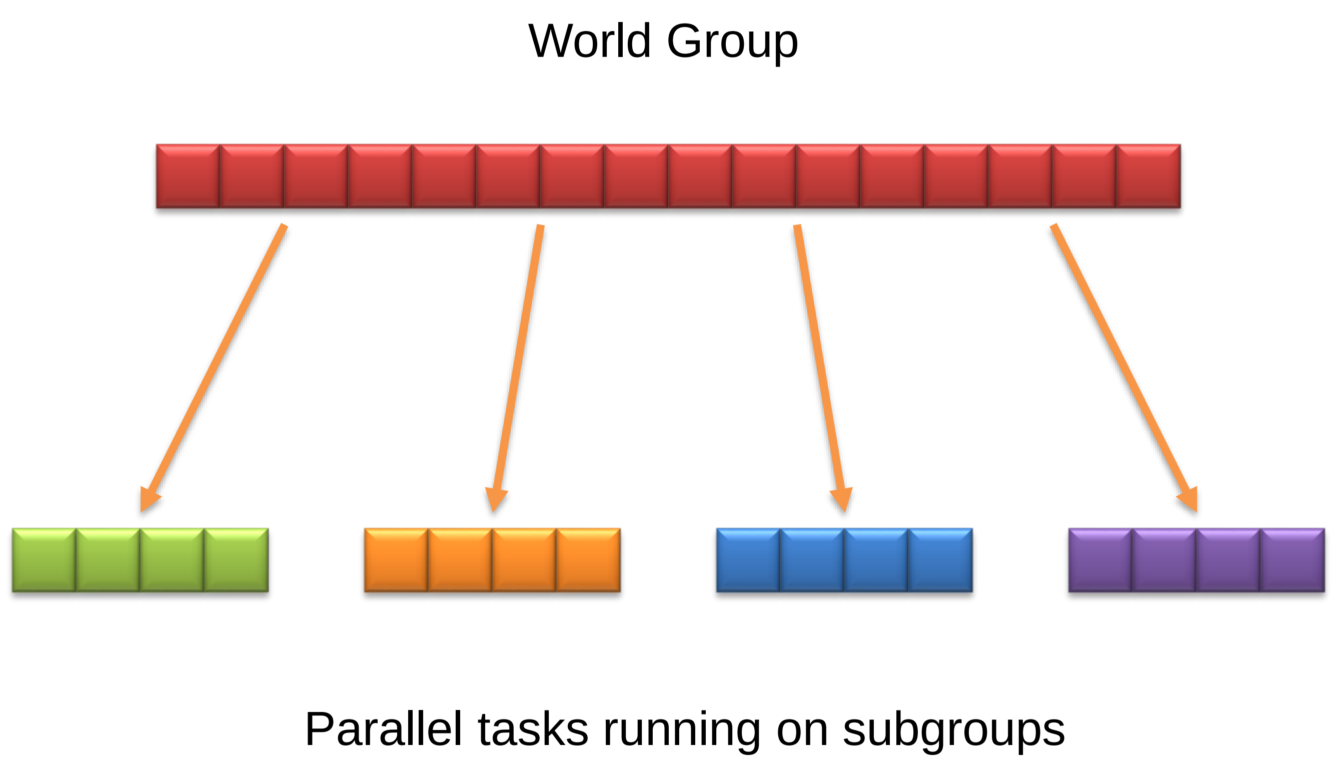


16351 bus
WECC
system

Power Flow Code

```
typdef BaseNetwork<PFBus,PFBranch> PFNetwork;
Communicator world;
shared_ptr<PFNetwork>
    network(new PFNetwork(world));

PTI23_parser<PFNetwork> parser(network);
parser.parse("network.raw");
network->partition();
typedef BaseFactory<PFNetwork> PFFactory;
PFFactory factory(network);
factory.load();
factory.setComponents();
factory.setExchange();

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

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->normInfinity();

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++;
}
```

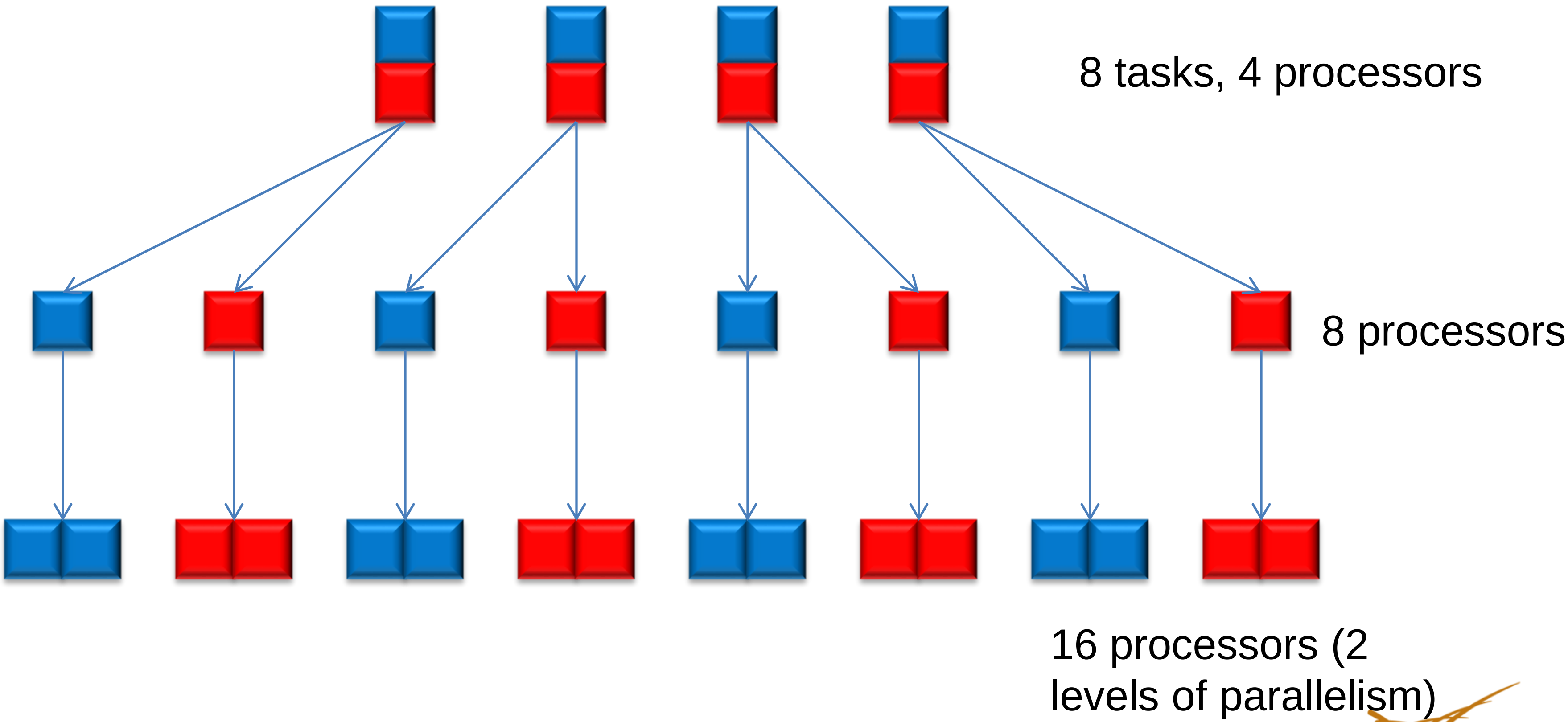
GridPACK Task Manager Support



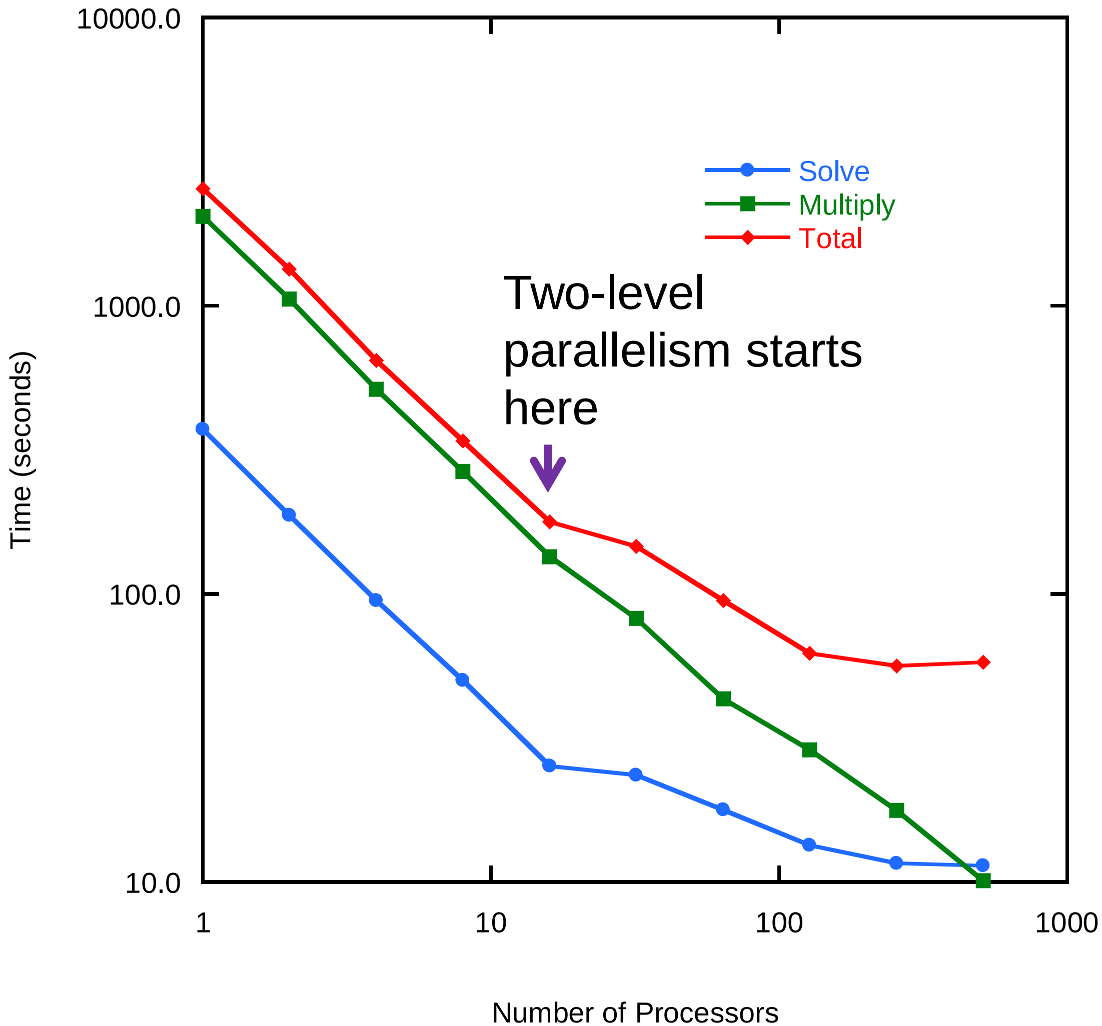
Pacific Northwest NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Multiple Levels of Parallelism



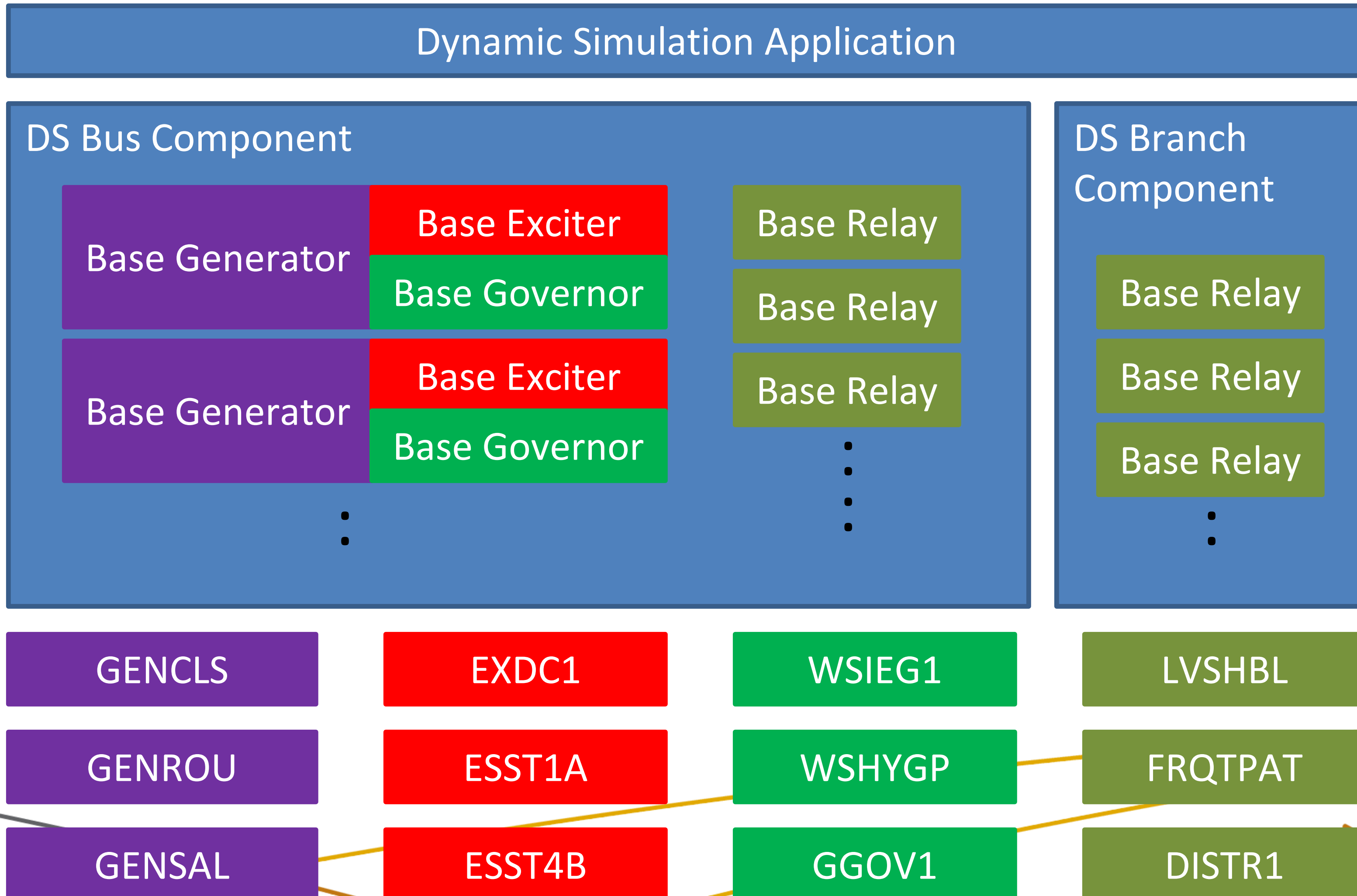
Dynamic Contingency Analysis



Simulation of 16 contingencies on 16351 bus WECC network

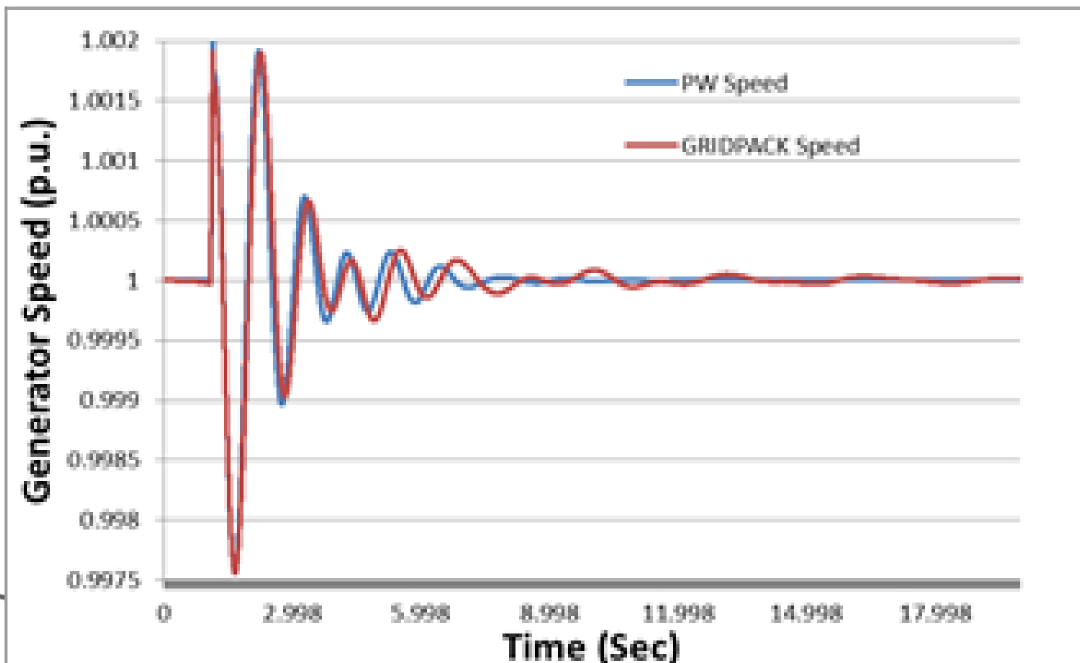
GridPACK Examples

Dynamic Simulation Mini-Framework



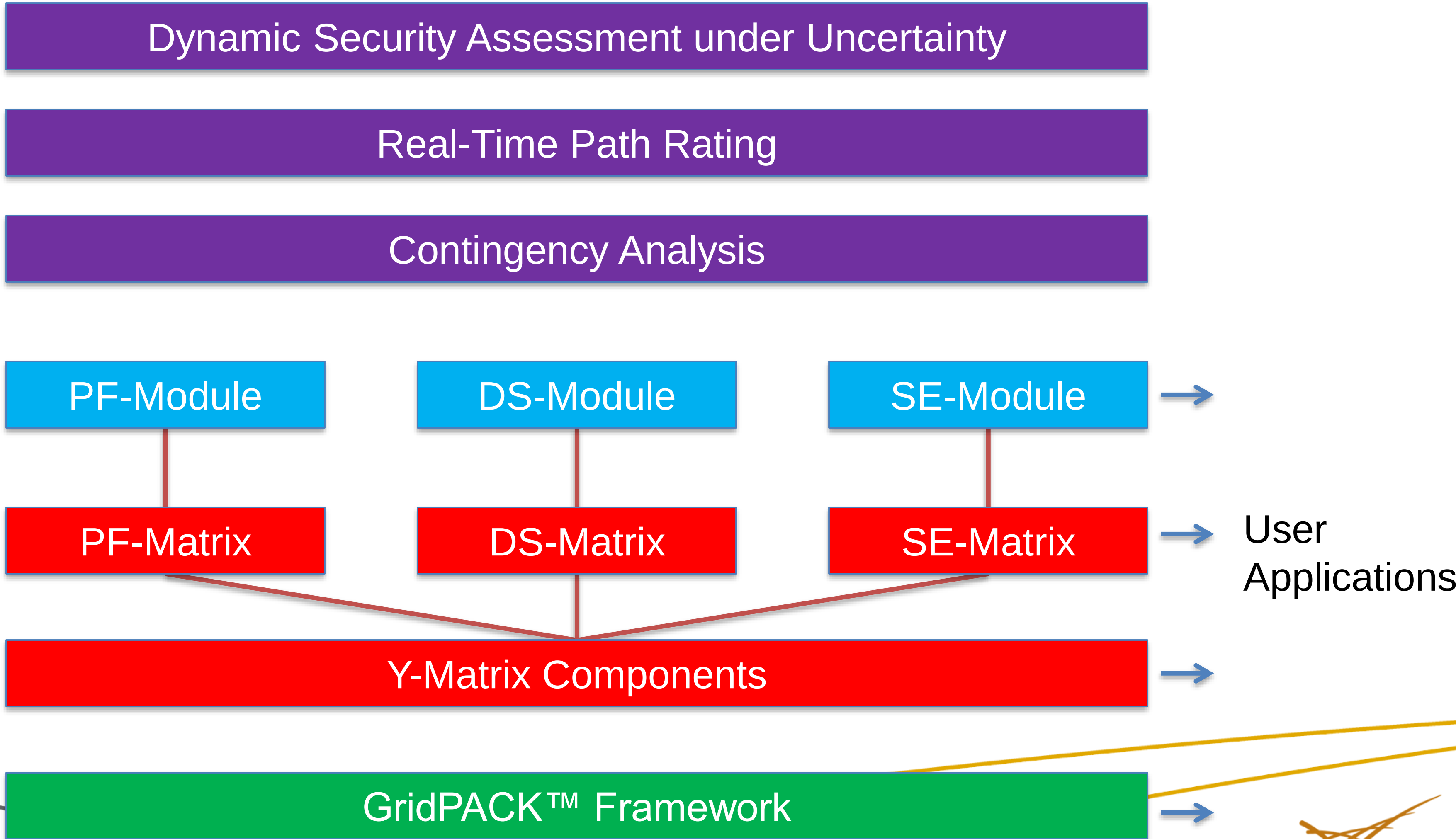
Dynamic simulation of WECC system

- WECC system of 17,000 buses with detailed dynamic models, 20 seconds simulation, results compared with PowerWorld.
- Achieve Faster-than-real-time simulation with **16 cores**.

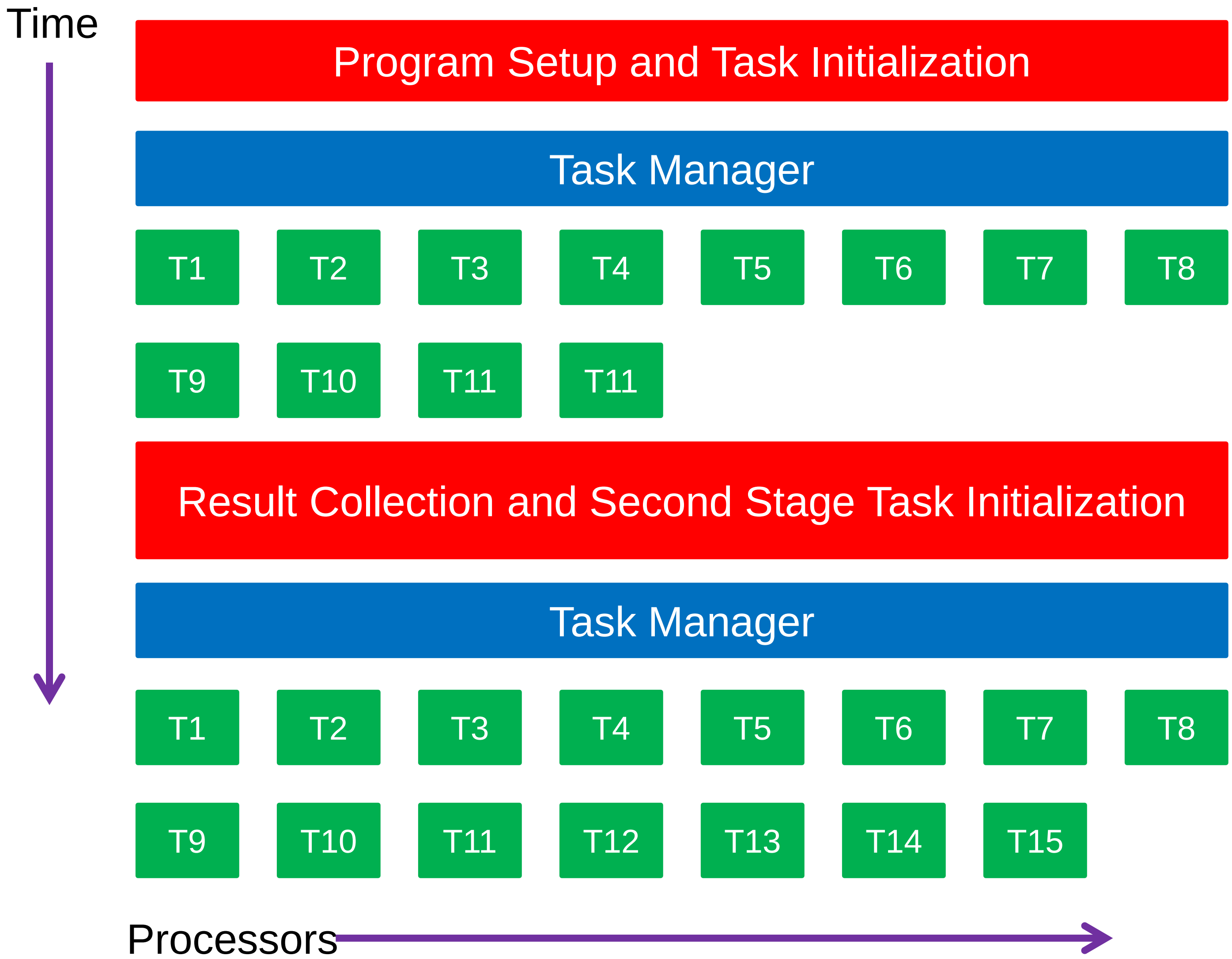


No. of Cores	Total Solution Time (seconds)
1	72.92
2	45.04
4	30.96
8	22.95
16	19.57

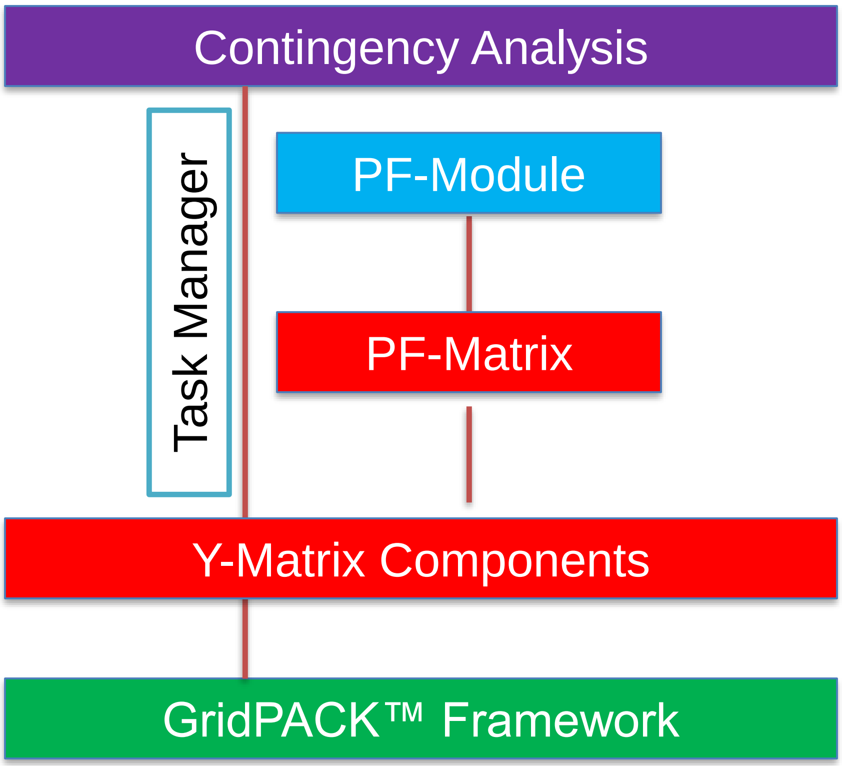
Module-Based Simulations

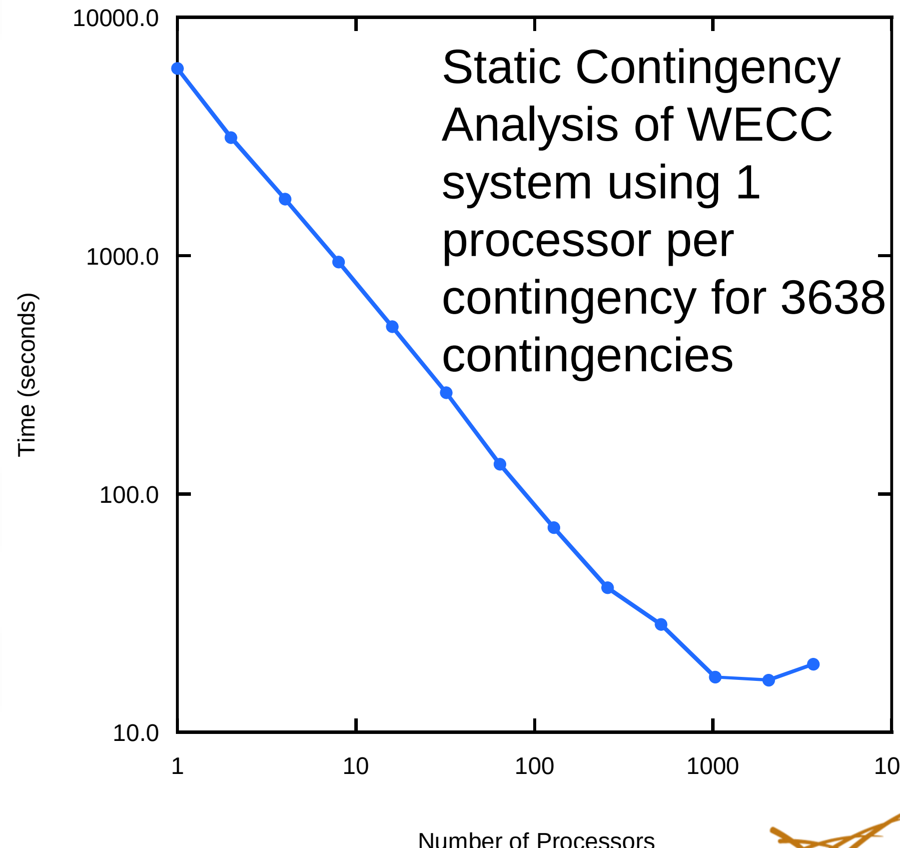


Many Task Simulations

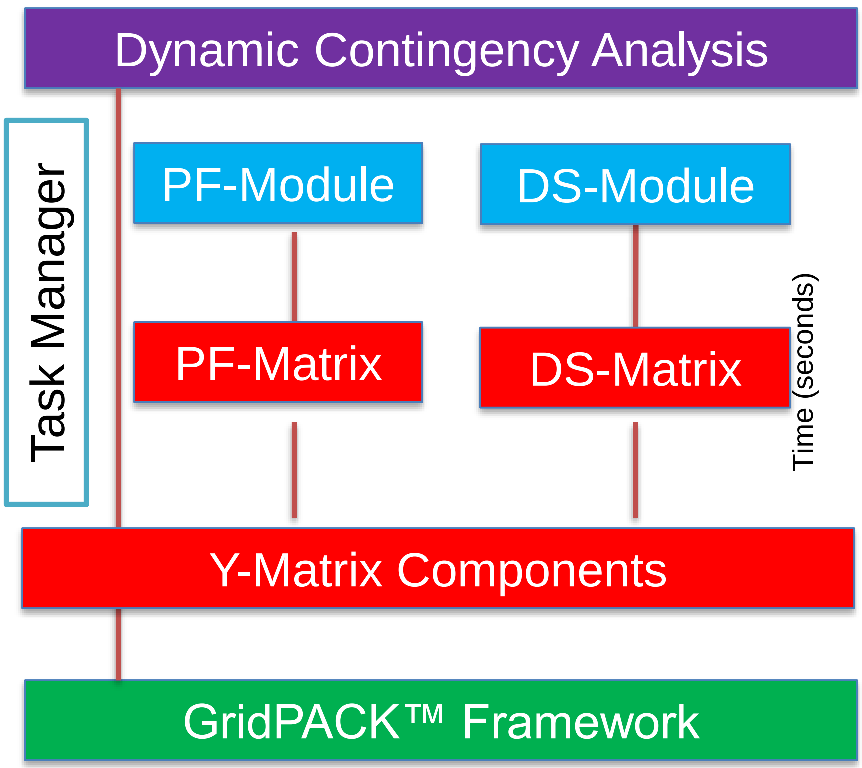


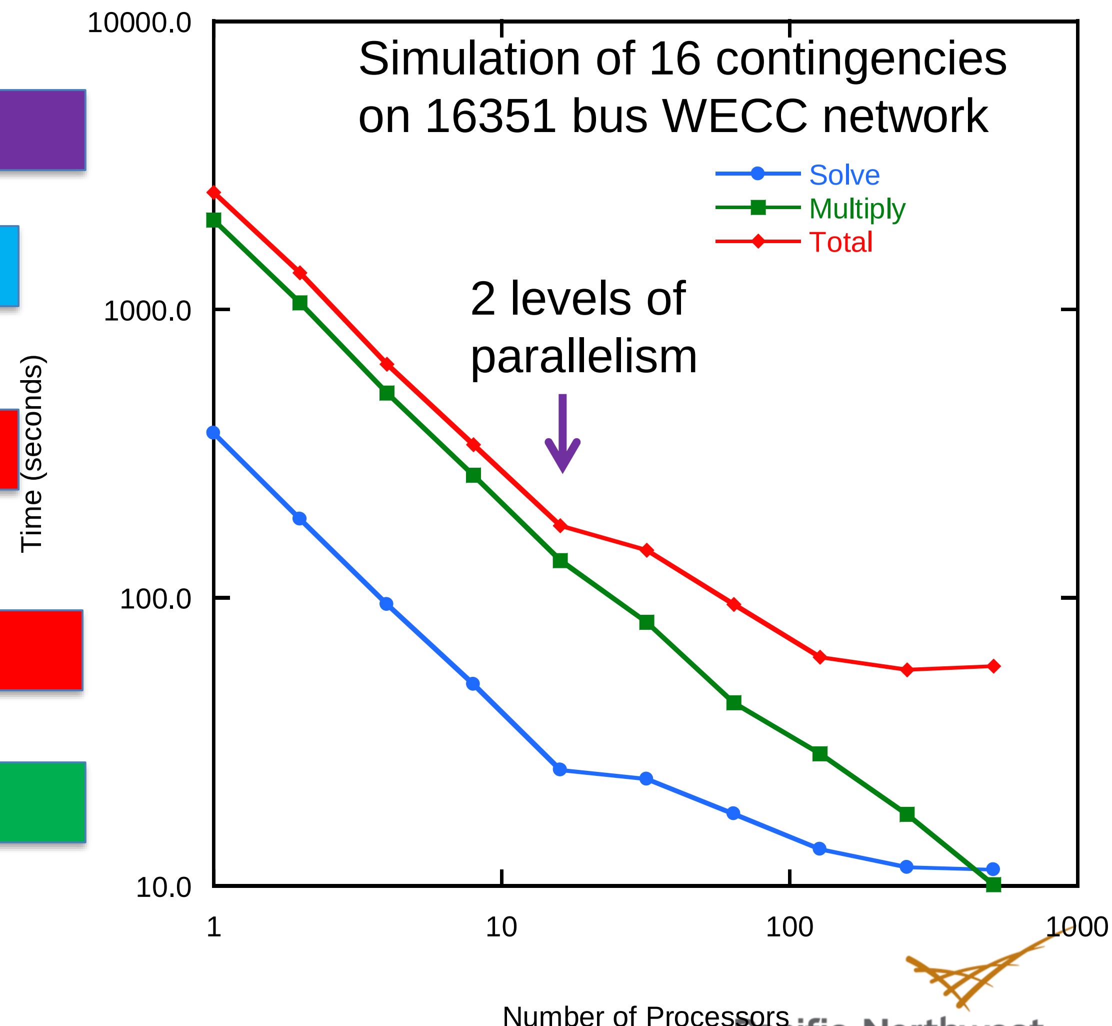
Application (1): Contingency Analysis



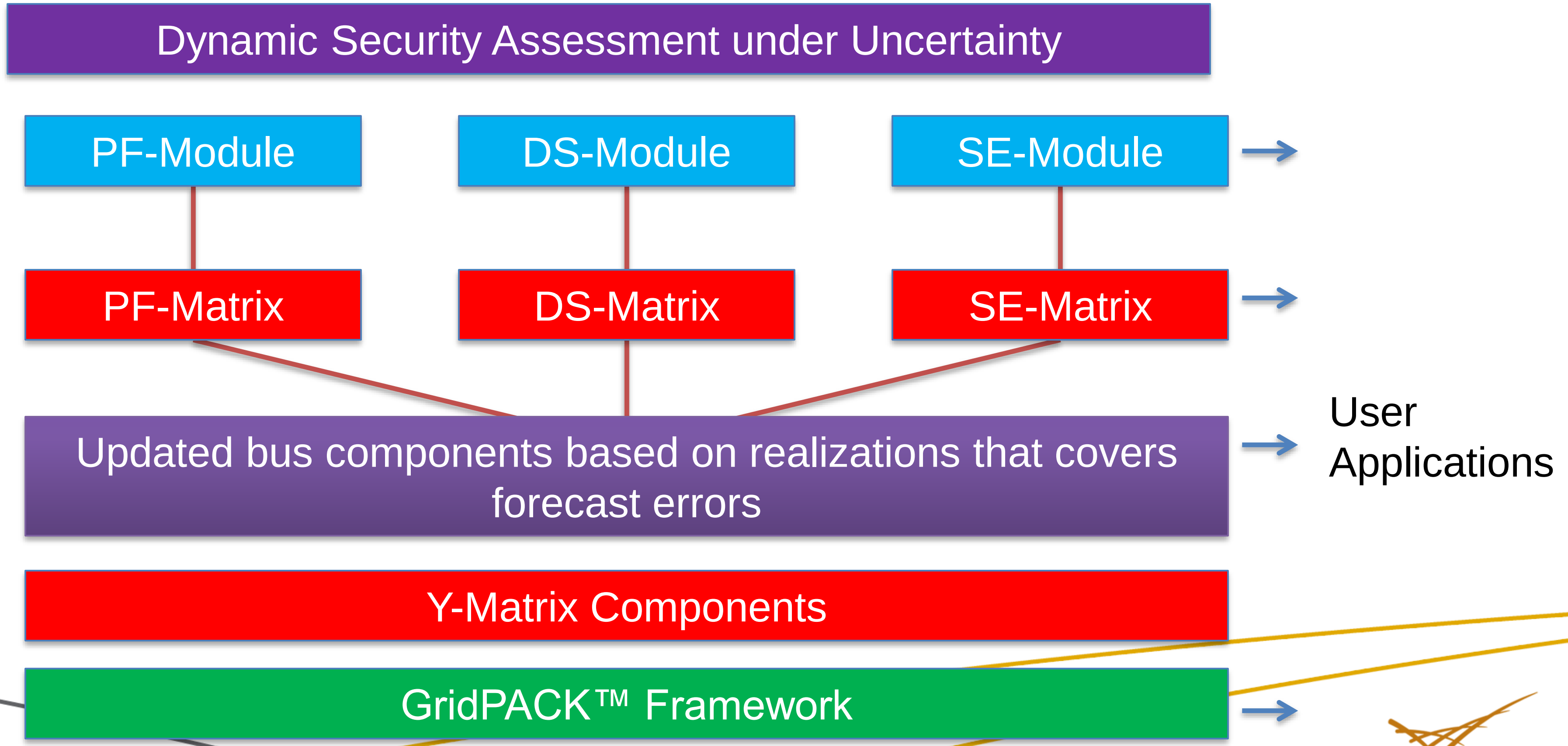


Example (2): Dynamic Contingency Analysis

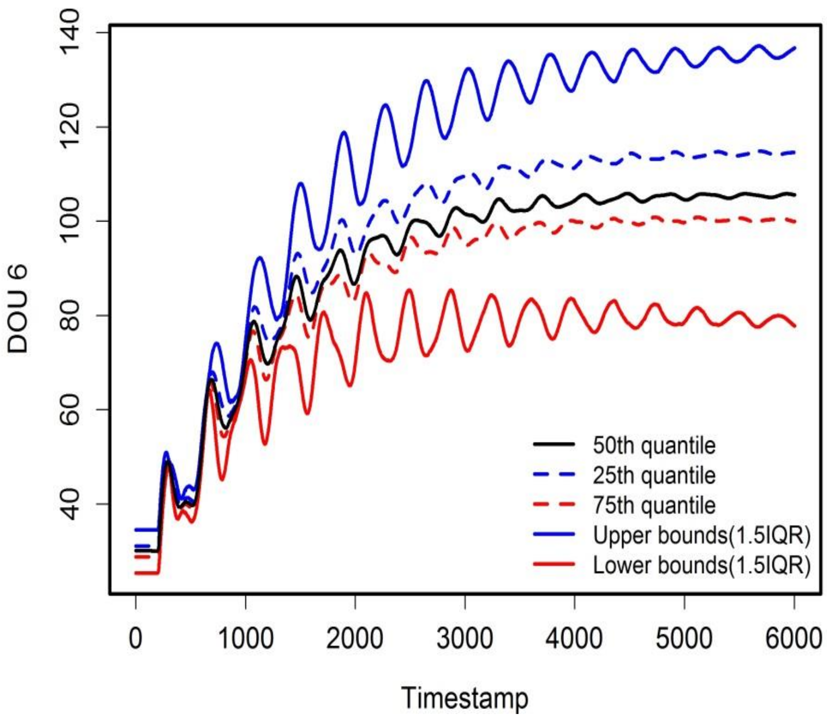




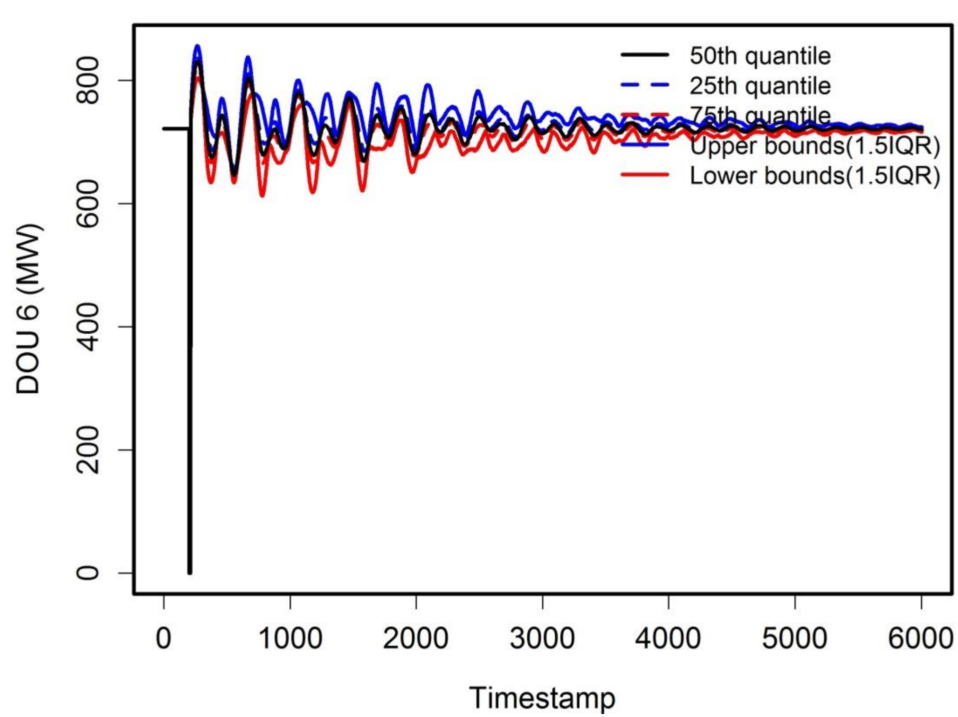
Application (3): Dynamic Security Assessment (DSA) under Uncertainty



Application (3) DSA under Uncertainty



Rotor speed under different contingencies with different quantiles and IQR



Line flow under different contingencies with different quantiles and IQR

- The quantiles of all the contingencies for each generator are used to represent the statistical characteristics at each time stamp.
- The lower/upper bounds use 1.5 interquartile range (IQR).

GridPACK Summary

- Open source software for running HPC power grid simulations
- Written in C++ and designed to run on Linux platforms with MPI
- Many applications already available
  - Power flow
  - Dynamic simulation
  - State estimation
  - Kalman Filters
- Can be reused to develop own applications
- Download and documentation at www.gridpack.org
- Contact us at gridpack.account@pnnl.gov

Accessing GridPACK

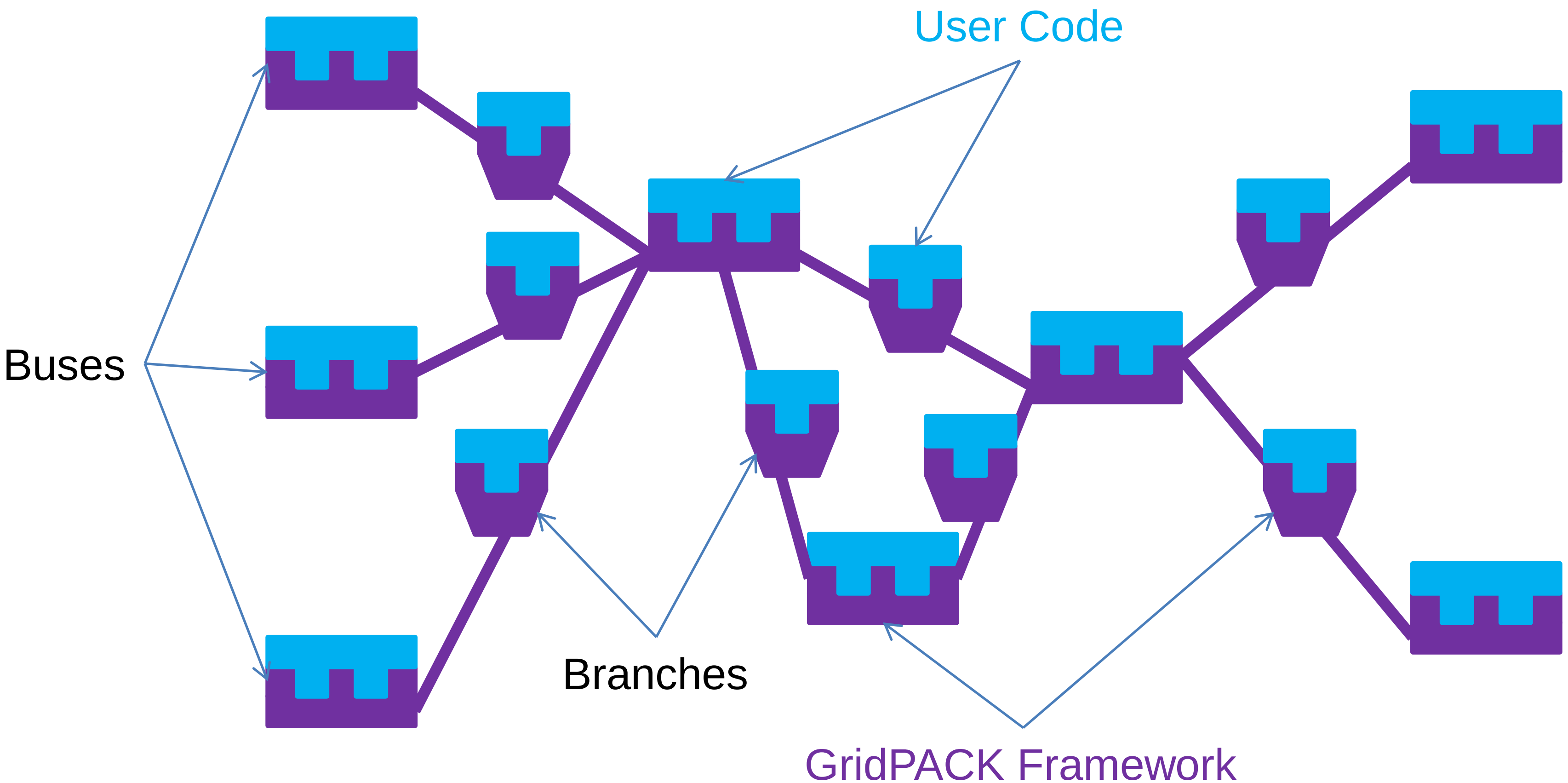
- Download from www.gridpack.org
- Extensive documentation in GridPACK user manual
- Documentation on building GridPACK on numerous different platforms. If you run into problems, we can help
- Contact us at gridpack.account@pnnl.gov

Extra Slides

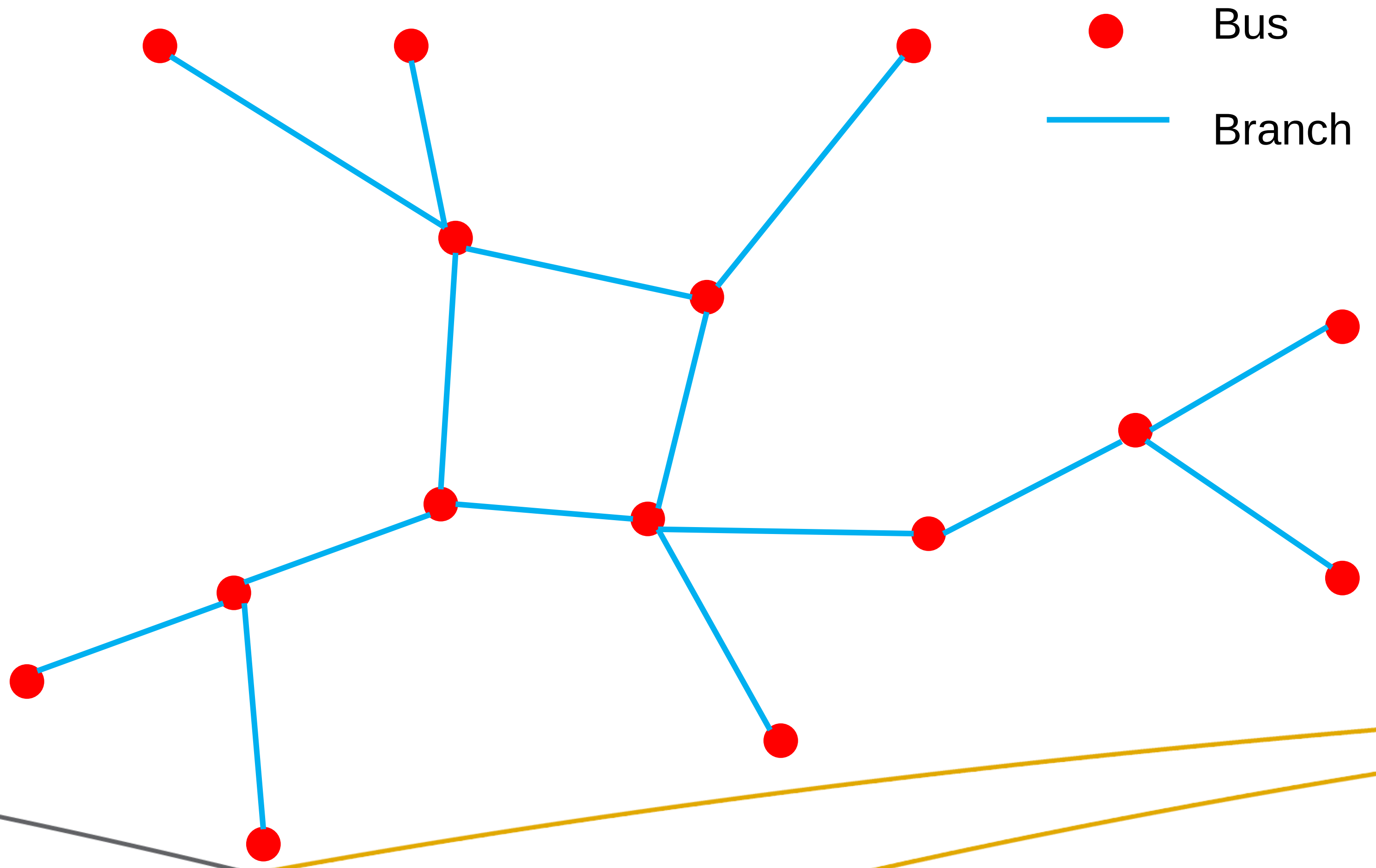
Customizing Networks using the BaseNetwork Class

- Template class that can be created with arbitrary user-defined types for the buses and branches
  - `BaseNetwork<MyBus, MyBranch>(const Communicator &comm)`
- Implements partitioning of network between processors
  - Create highly connected sub-networks on each processor with minimal connections between processors
- Implements data exchanges between buses and branches on different processors
- Manages indexing of network components

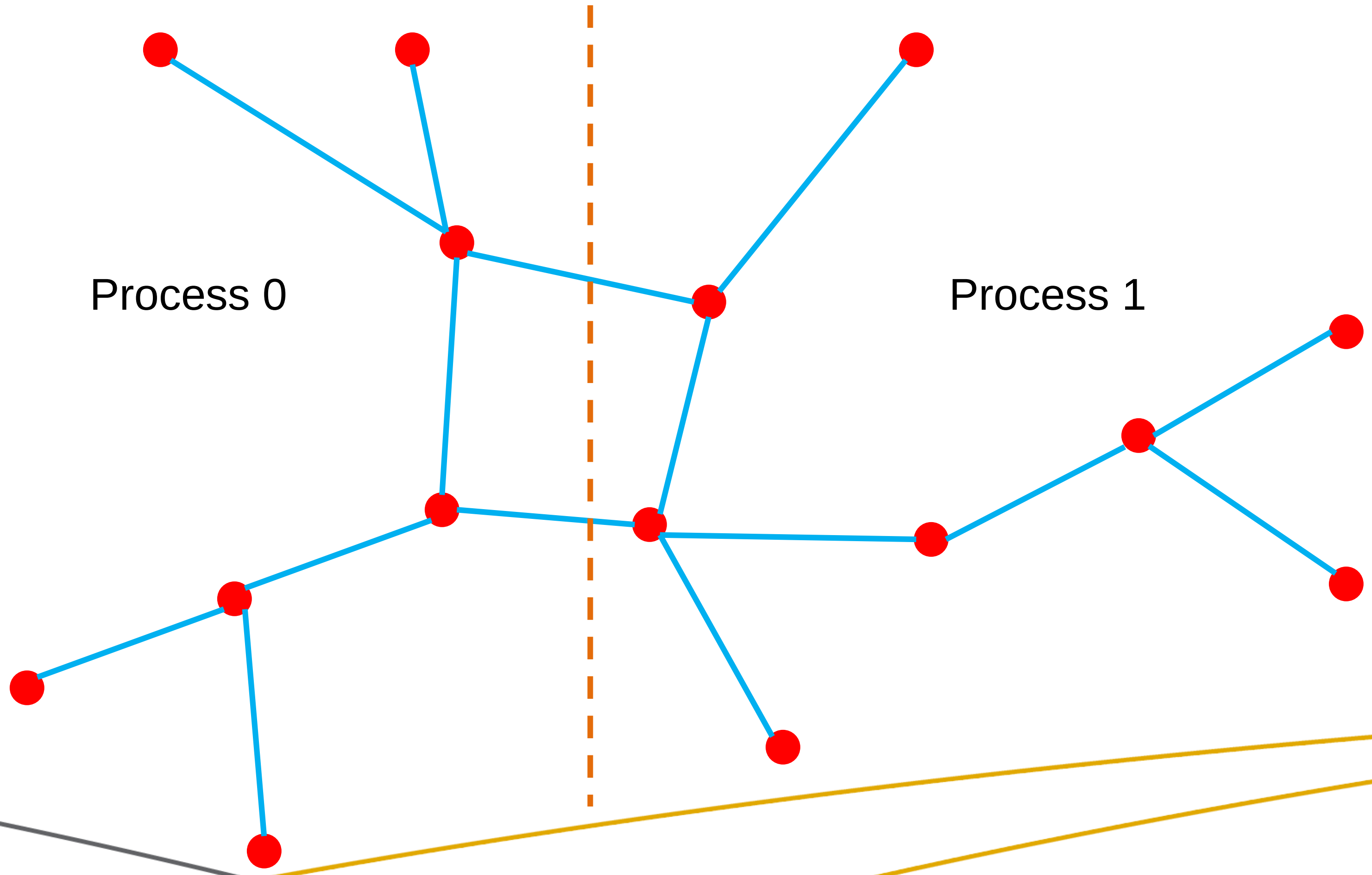
Customization Through Templates



Network Topology

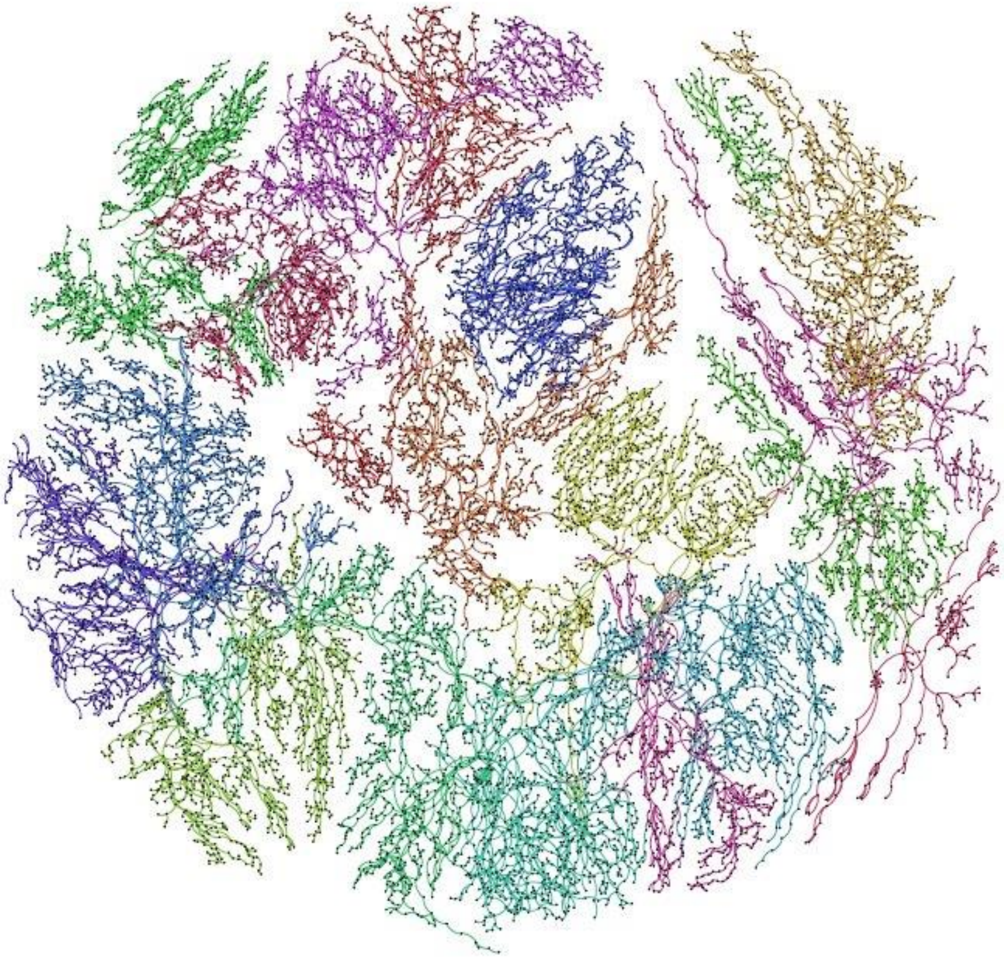


Partitioning the Network



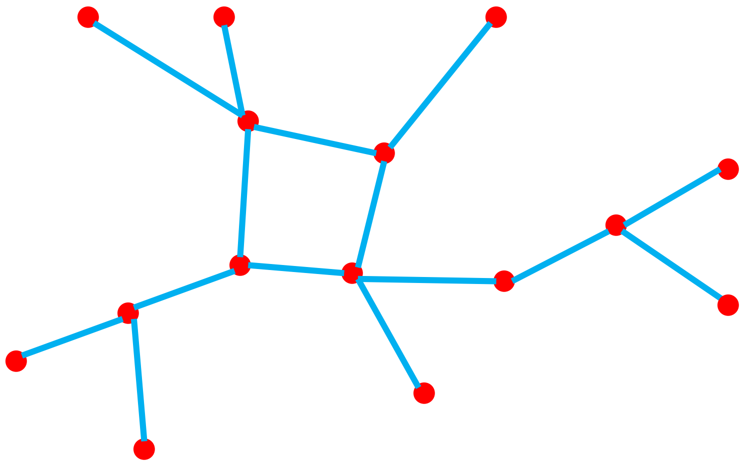
Partitioning of Network

WECC (Western Electricity Coordinating Council) network partitioned between 16 processors

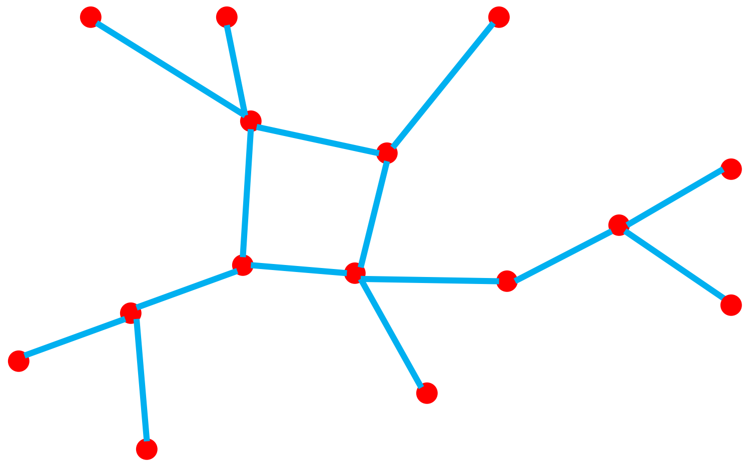


Multiple Networks

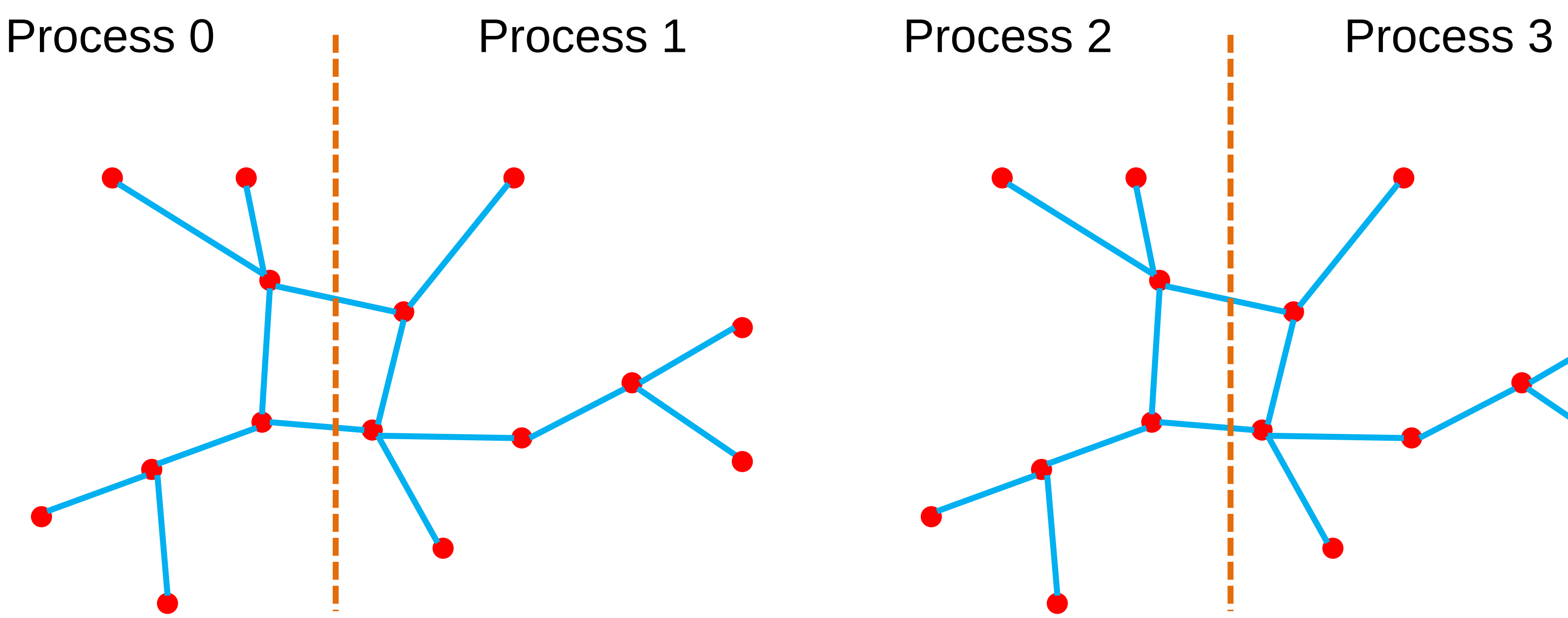
Process 0



Process 1



Multiple Distributed Networks



Component Classes

- Each bus and branch in the network has an associated bus or branch object. They also have an associated DataCollection object
- The bus and branch classes are written by the user. Access to other functionality in GridPACK is through functions defined in the bus and branch interfaces
  - Creation of matrices and vectors
  - IO
  - Data exchange between processors
  - Creation of optimization equations

Traditional Programming to Evaluate Y_{ii}

```
integer nbus
integer attached_branch_start(nbus+1)
integer attached_branch_nghbrs(n_neighbors)

for ibus = 1, nbus
   ibeg = attached_branch_start(ibus)
   iend = attached_branch_start(ibus+1)-1
   ydiag(ibus) = 0.0
   for j = ibeg, iend
      ibranch = attached_branch_nghbrs(j)
      x = reactance(ibranch)
      r = resistance(ibranch)
      c = cmplx(r,x)
      ydiag(ibus) = ydiag(ibus)+1.0/c
   end do
end do
```

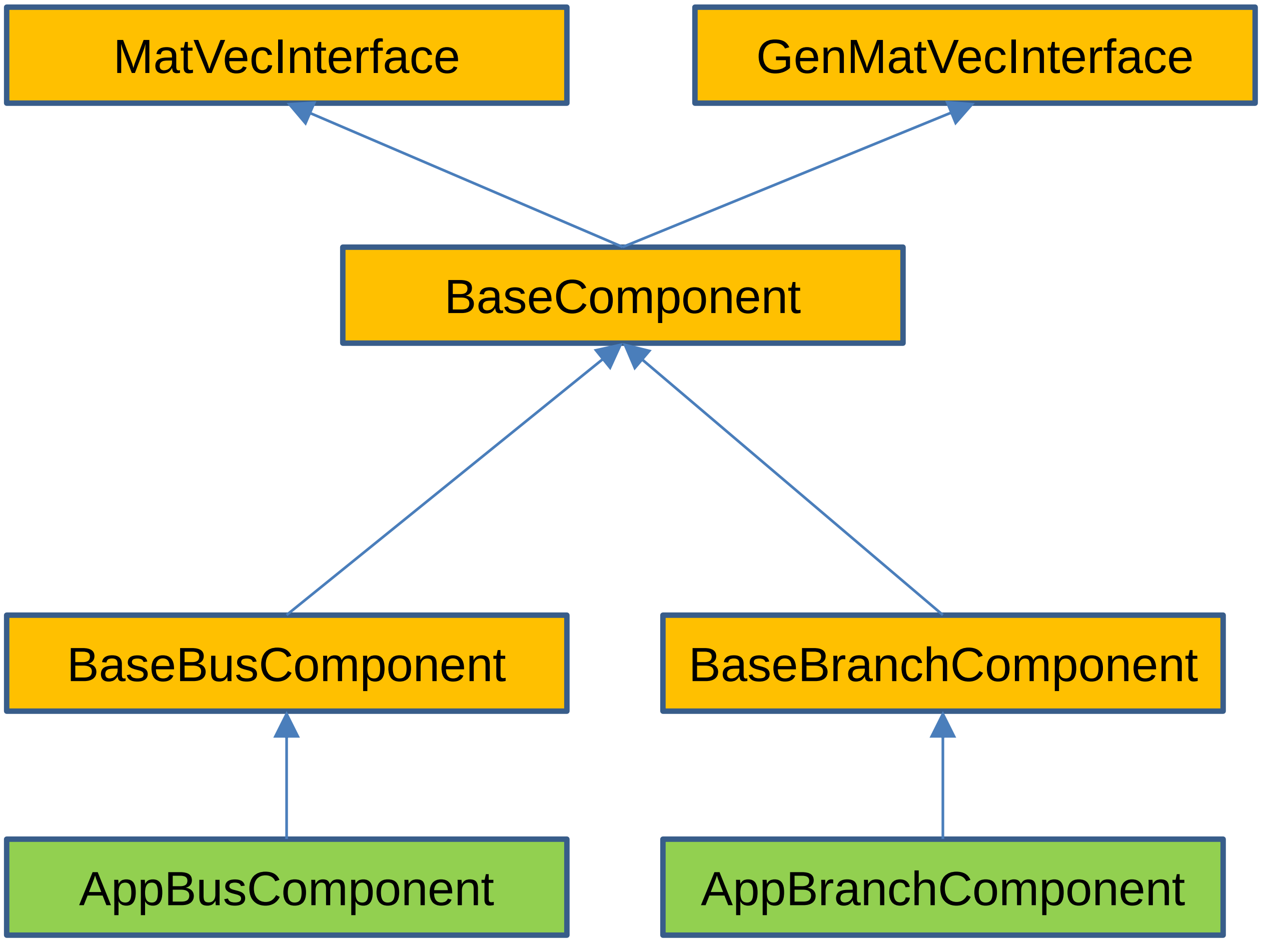
Properties are in large lists with auxiliary data structures maintaining relationships between different pieces of data

C++ Programming to Evaluate Y_{ii}

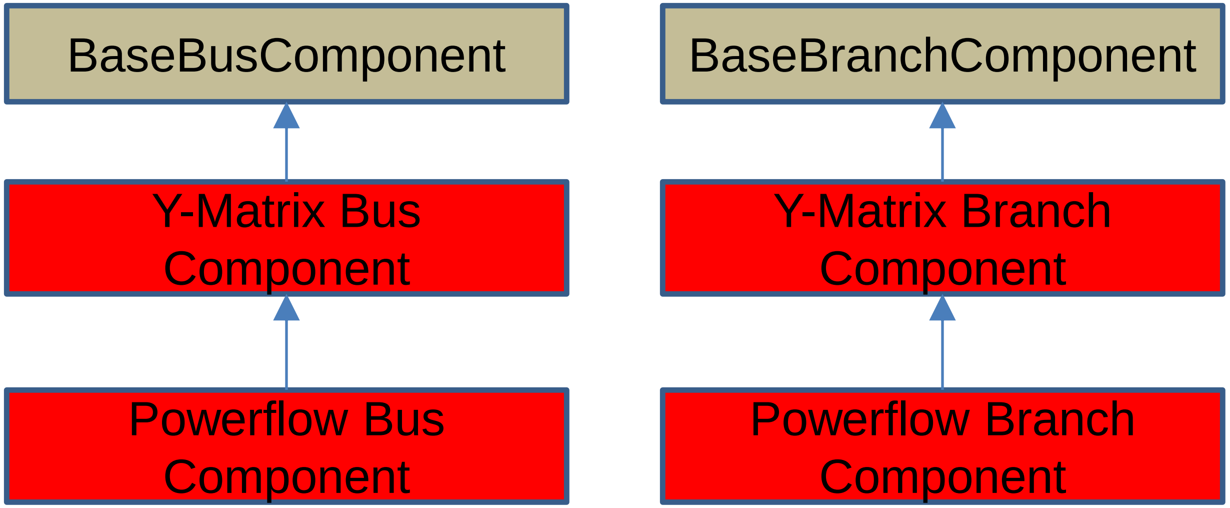
```
int nbus = network->numBuses()

for (int ibus=0; ibus<nbus; ibus++) {
  Bus *bus = network->getBus(i);
  std::vector<Branch> branches = bus->getNeighborBranches();
  DoubleComplex y(0.0,0.0);
  for (j=0; j<branches.size(); j++) {
    Branch *branch = branches[j];
    double r = branch->getResistance();
    double x = branch->getReactance();
    DoubleComplex c(r,x);
    y += 1.0/c;
  }
  bus->setYdiag(y);
}
```

Data is tied to individual objects instead of being associated with large. global lists

Component Class Hierarchy



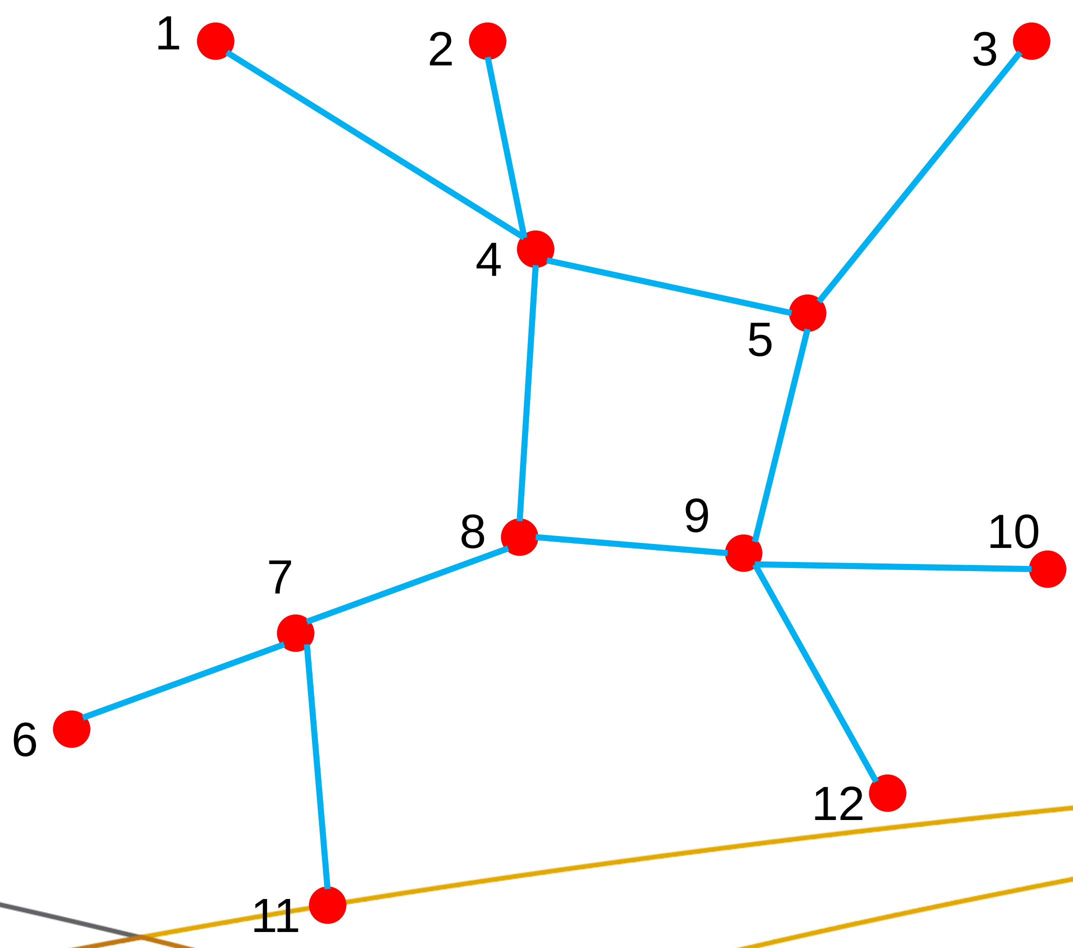
Component Reuse



Mapper

- Provides a flexible framework for constructing matrices and vectors representing power grid equations
- Hide the index transformations and partitioning required to create distributed matrices and vectors from application developers
- Developers can focus on the contributions to matrices and vectors coming from individual network elements

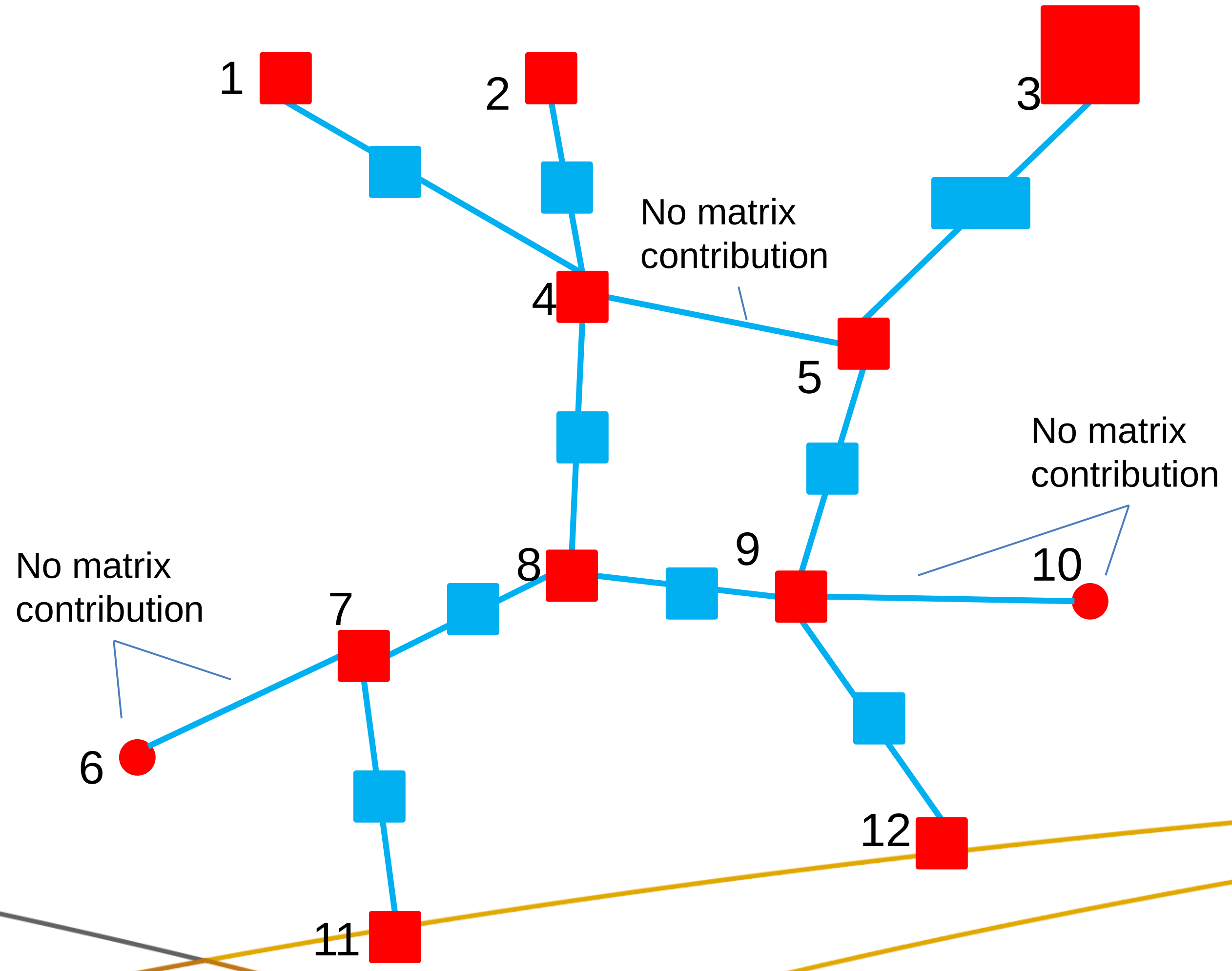
Mapper



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle *Since 1965*

Matrix Contributions from Components



MatVecInterface

```
// Implemented on buses
virtual bool matrixDiagSize(int *isize,
                            int *jsize) const
virtual bool matrixDiagValues(ComplexType *values)

// Implemented on branches
virtual bool matrixForwardSize(int *isize,
                               int *jsize) const
virtual bool matrixReverseSize(int *isize,
                               int *jsize) const
virtual bool matrixForwardValues(ComplexType *values)
virtual bool matrixReverseValues(ComplexType *values)
```

Distribute Component Contributions and Eliminate Gaps