Integrated Retail and Wholesale Power System Operation with Smart-Grid Functionality

Project Update Report
May 21, 2010
Chengrui Cai

Outline:

1. Introduction to GridLab-D
2. GridLab-D structure
3. Climate Module
4. Metronome example
5. Case study: IEEE 14 Bus transmission power flow
6. Work plan for following weeks

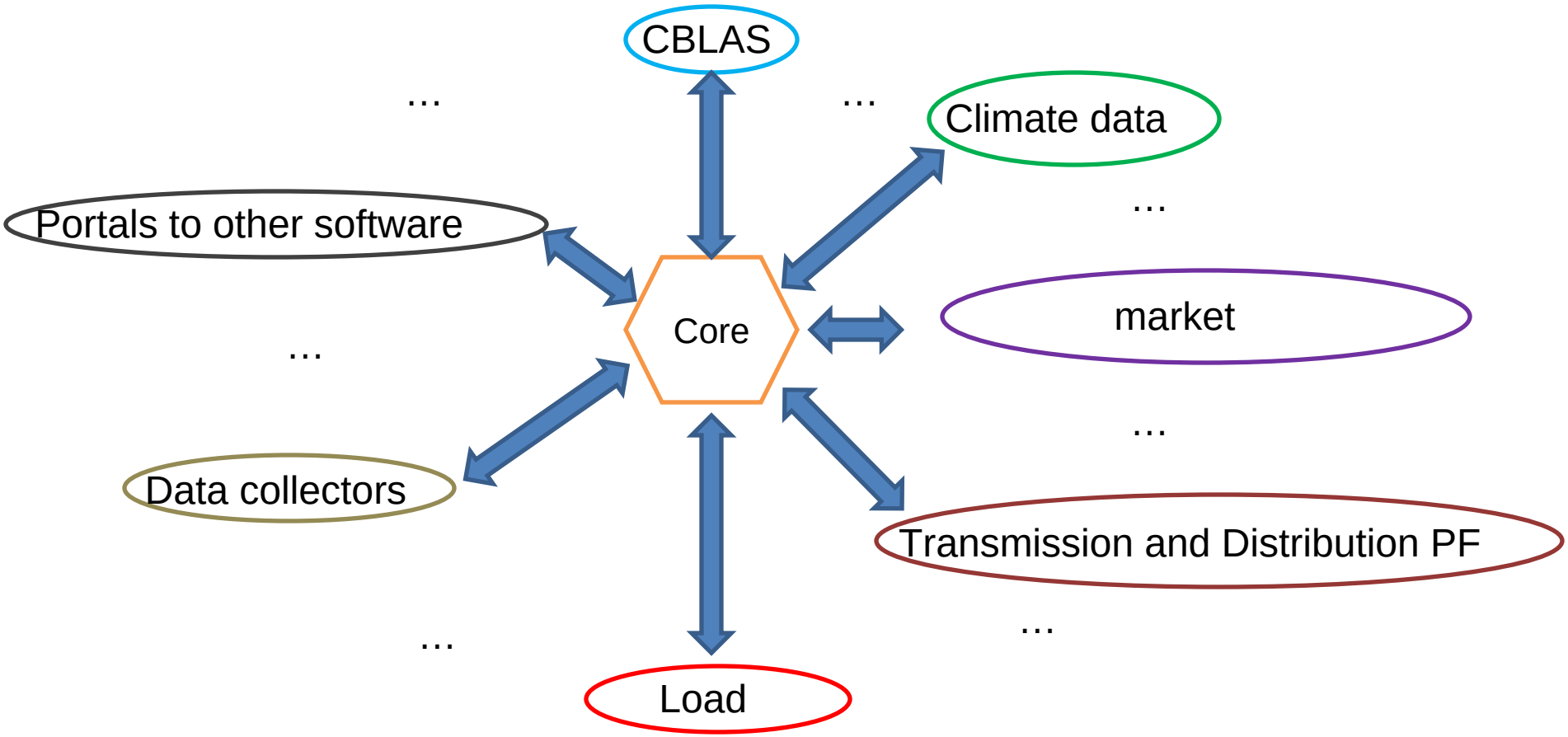
Introduction to GridLab-D

1. GridLAB-D is a flexible agent-based simulator.

2. GridLAB-D will continue advancing the clock and allowing objects to update themselves until all the object report that they are at equilibrium and the clock need not be advanced further.

3. Use .GLM file as Input file and output in xml, csv and other types of files

Structure of GridLab-D



Structure of GridLab-D

1. Module, Class, object and model

   Module: Network(Transmission PF), Market, Climate and so on

   Class: Node, link and other classes in Network Module

   Object: realizations declared in GLM file

   Model: Described by GLM file

2. GLM file:

   1 or more module loaded;
   1 or more classes;
   1 or more objects;
   only one clock;
   zero or more directives
   (import cmd, module, class, object and clock)

Climate module

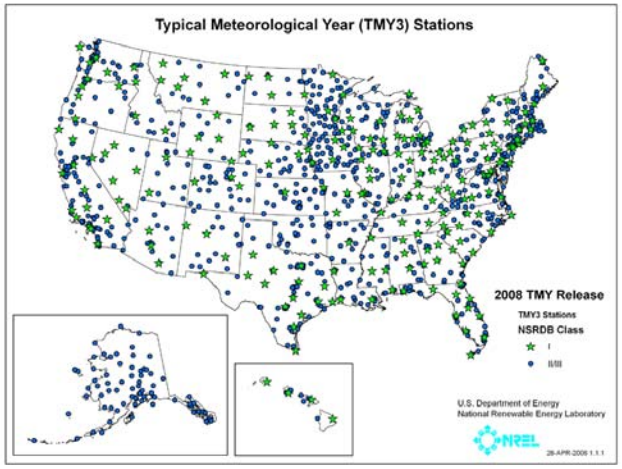
Give other objects the access to weather data specified by the global variable global_clock

```
class climate {
    char1024 tmyfile;
    char32   city;
    double   temperature[F];
    double   humidity[%];
    double   solar_flux[W/sf]; // 9 values array (S,SE,SW,E,W,NE,NW,N,HZ)
    double   wind_speed[mph];
    double   wind_dir[deg];
    double   wind_gust[mph];
}
```

The climate class currently supports only the Typical Meteorological Year (TMY) weather data using the TMY2 weather data format.

TMY1: 1952-1975
TMY2: 1961-1990
TMY3: 1991- 2005 and continue to add

TMY, TMY2 and TMY3 data sets cannot be used interchangeably.



Iowa

725457	Algona	Class III	725456	Keokuk Muni	Class III
725453	Atlantic	Class III	725493	Knoxville	Class III
725486	Boone Muni	Class III	725484	Le Mars	Class III
725455	Burlington Municipal AP	Class II	725485	Mason City Municipal Arpt	Class I
725468	Carroll	Class III	725475	Monticello Muni	Class III
725450	Cedar Rapids Municipal AP	Class II	725487	Muscatine	Class III
725469	Chariton	Class III	725464	Newton Muni	Class III
725463	Charles City	Class III	725488	Oelwen	Class III
725479	Clarinda	Class III	725489	Orange City	Class III
725473	Clinton Muni (AWOS)	Class II	725465	Ottumwa Industrial AP	Class II
725497	Council Bluffs	Class III	725494	Red Oak	Class III
725474	Creston	Class III	725495	Sheldon	Class III
725476	Decorah	Class III	725467	Shenandoah Muni	Class III
725477	Denison	Class III	725570	Sioux City Sioux Gateway AP	Class I
725460	Des Moines Intl AP	Class I	726500	Spencer	Class II
725470	Dubuque Regional AP	Class I	725496	Storm Lake	Class III
726499	Estherville Muni	Class II	725454	Washington	Class III
726498	Fair Field	Class III	725480	Waterloo Municipal AP	Class I
725490	Fort Dodge (AWOS)	Class II	725478	Webster City	Class III
725483	Fort Madison	Class III			

A metronome example

Objective: To generate a metronome running at specified rate and number of counts

Pseudo code:

```
class Metronome {
      enum sound {tick, tock};
      TIMESTAMP last_time;
      double bpm;
      int count;
 }
```

+

An initialization function to initialize the clock value of a metronome object

+

An sync function where we can define the behaviors of the metronome

A metronome example

Metronome.glm

```
#set pauseatexit=1

clock {
   timezone PST+8PDT;
   timestamp '2000-01-01 0:00:00';
}

class metronome {
   double bpm; // beats per minute
   int16 count; // beats left
   enumeration {TIC=0, TOC=1} side;
   timestamp last_t;

   intrinsic create (object parent)
   {
      last_t = gl_globalclock;
      return SUCCESS;
   };

   intrinsic sync(TIMESTAMP t0, TIMESTAMP t1)
   {
      TIMESTAMP next_t = (TIMESTAMP)(last_t + 60/bpm);
      if (t1==next_t)
      {
         side = (side==TIC) ? TOC : TIC;
         last_t = next_t;
         count--;
      }

   return count>0 ? (TIMESTAMP)(last_t + 60/bpm) : TS_NEVER;};
};

module tape;

object metronome{
   bpm 60;
   count 10;
   object recorder
   {
      property side;
      file "metronome1.csv";
   };
};
```

Macro #

Clock: define the time zone and simulation start time

Definition of the metronome class according to the pseudo code

Tell the core the time when the metronome acted last time

Create runs when a new metronome object is decaled in GLM file

Behaviors of the metronome

Tell the core when the metronome will update its state next time

Load the tape module which is a kind of data collector

Nested object indicates a parent and child relationship

A metronome example

```
# file..... metronome1.csv
# date..... Wed May 19 01:22:13
2010
# user..... ccai
# host..... (null)
# target.... metronome 0
# trigger... (none)
# interval.. -1
# limit..... 0
# timestamp,side
2000-01-01 00:00:00 PST,TIC
2000-01-01 00:00:01 PST,TOC
2000-01-01 00:00:02 PST,TIC
2000-01-01 00:00:03 PST,TOC
2000-01-01 00:00:04 PST,TIC
2000-01-01 00:00:05 PST,TOC
2000-01-01 00:00:06 PST,TIC
2000-01-01 00:00:07 PST,TOC
2000-01-01 00:00:08 PST,TIC
2000-01-01 00:00:09 PST,TOC
2000-01-01 00:00:10 PST,TIC

Bpm  = 60;
Count  = 10;
```

```
# file..... metronome1.csv
# date..... Wed May 19 01:40:11
2010
# user..... ccai
# host..... (null)
# target.... metronome 0
# trigger... (none)
# interval.. -1
# limit..... 0
# timestamp,side
2000-01-01 00:00:00 PST,TIC
2000-01-01 00:00:02 PST,TOC
2000-01-01 00:00:04 PST,TIC
2000-01-01 00:00:06 PST,TOC
2000-01-01 00:00:08 PST,TIC
2000-01-01 00:00:10 PST,TOC
2000-01-01 00:00:12 PST,TIC
2000-01-01 00:00:14 PST,TOC
2000-01-01 00:00:16 PST,TIC
2000-01-01 00:00:18 PST,TOC
2000-01-01 00:00:20 PST,TIC

Bpm = 30;
Count = 10
```

Metronome.xml

```
# file..... metronome2.csv
# date..... Wed May 19 01:43:26 2010
# user..... ccai
# host..... (null)
# target.... metronome 0
# trigger... (none)
# interval.. -1
# limit..... 0
# timestamp,side
2000-01-01 00:00:00 PST,TIC
2000-02-01 09:15:01 PST,TOC
2000-02-01 09:15:02 PST,TIC
2000-02-01 09:15:03 PST,TOC
2000-02-01 09:15:04 PST,TIC
2000-02-01 09:15:05 PST,TOC
2000-02-01 09:15:06 PST,TIC
2000-02-01 09:15:07 PST,TOC
2000-02-01 09:15:08 PST,TIC
2000-02-01 09:15:09 PST,TOC
2000-02-01 09:15:10 PST,TIC
```

A metronome example

Summary:

1. Use macro command to change default core setting
2. Different directives: clock, module, class, object
3. How to define new class in GLM file
4. Indispensible functions (create & sync) for a class in Gridlab-D
5. Time series control via TIMESTAMP
6. Load existing module and declare parent-child relation
7. Use data collector “tape” to record output

Case study: transmission power flow module

Properties definition required
by Network Module:

1.Bus type (PV, PQ, swing)

2.Voltage phasor for swing bus

3.Complex power for PQ

4.Voltage magnitude and real

power injection for PV

5. Limits (Max.Var and Min.Var)

6. Parent-child relation ship

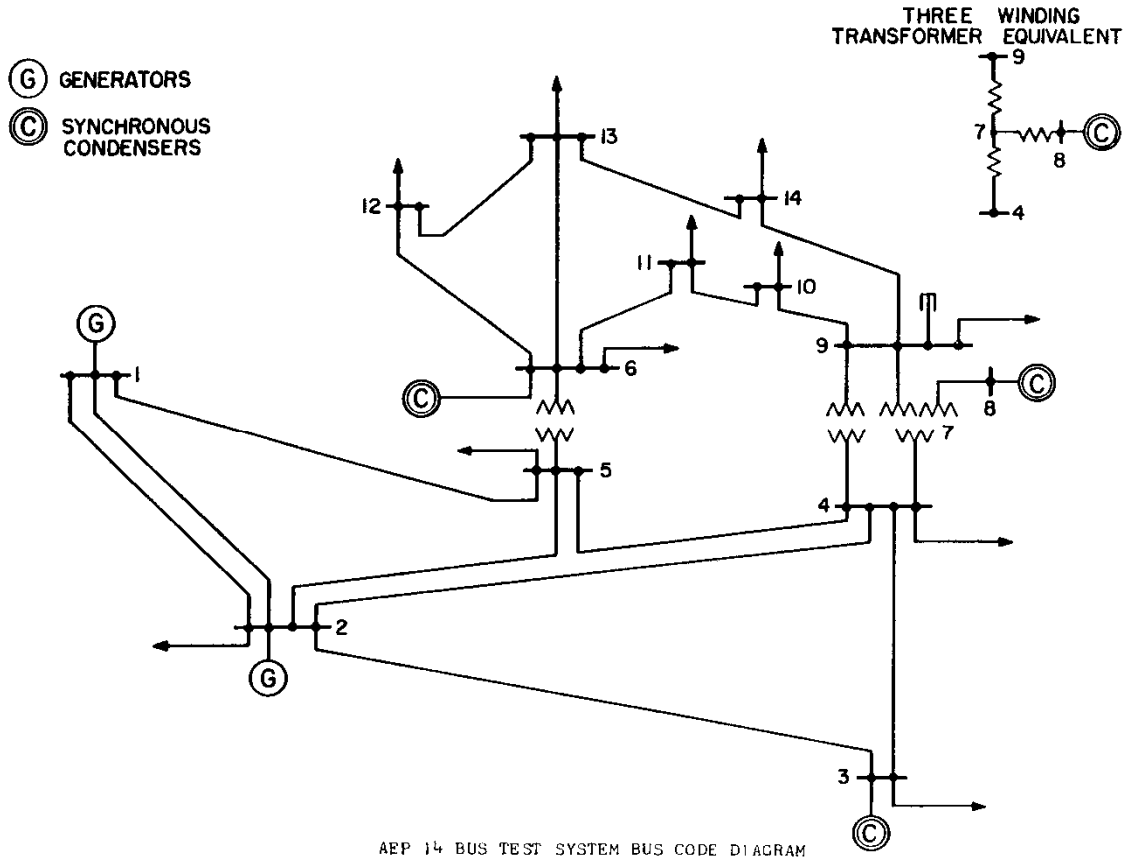
7. Node shunt element

8. Branch's admittance

9. Branch's line charging reactance

10. Branch's start and end bus

11. Guess initial value for unspecified states



Case study: transmission power flow module

```
#set pauseatexit=1

clock {
     timestamp '2000-01-01 0:00:00';
}

module network {
    acceleration_factor 1;
    convergence_limit 0.000000001;
}

object node {
    name Bus1;
    V 1.060;
    S -2.324+0.169j;
    type SWING;
}

object node {
    name Bus2;
    parent Bus1;
    V 1.04106-.0907144j;
    S -0.183-0.297j;
    type PV;
    Qmax_MVAR 50;
    Qmin_MVAR -40;
}

… \\ other nodes object

object link {
    name link1-2;
    from Bus1;
    to Bus2;
    Y 4.99913-15.26309j;
    B 0.0528;
}

… \\ other branch object
```

Name	V	S	G	B	Qmax_MVAR	Qmin_MVAR	type	flow_area_num	base_kV
Bus1 (#0)	+1.06+0i	+2.31362-0.166949j	+0	+0	+0	+0	SWING	1	+0
Bus2 (#1)	+1.045-4.94424d	-0.183-0.262911j	+0	+0	+50	-40	PV	1	+0
Bus3 (#2)	+1.01-12.5911d	+0.942+0j	+0	+0	+40	+0	PV	1	+0
Bus4 (#3)	+1.02725-10.3952d	+0.478-0.039j	+0	+0	+0	+0	PQ	1	+0
Bus5 (#4)	+1.03228-8.93033d	+0.076+0.016j	+0	+0	+0	+0	PQ	1	+0
Bus6 (#5)	+1.07-14.6687d	+0.112-0.343138j	+0	+0	+24	-6	PV	1	+0
Bus7 (#6)	+1.05489-13.5348d	+0+0i	+0	+0	+0	+0	PQ	1	+0
Bus8 (#7)	+1.09-13.5348d	+0-0.21727j	+0	+0	+24	-6	PV	1	+0
Bus9 (#8)	+1.04872-15.152d	+0.295+0.166j	+0	+0.19	+0	+0	PQ	1	+0
Bus10 (#9)	+1.04504-15.3508d	+0.09+0.058j	+0	+0	+0	+0	PQ	1	+0
Bus11 (#10)	+1.05389-15.1359d	+0.035+0.018j	+0	+0	+0	+0	PQ	1	+0
Bus12 (#11)	+1.05462-15.5127d	+0.061+0.016j	+0	+0	+0	+0	PQ	1	+0
Bus13 (#12)	+1.04935-15.5702d	+0.135+0.058j	+0	+0	+0	+0	PQ	1	+0
Bus14 (#13)	+1.03094-16.3421d	+0.149+0.05j	+0	+0	+0	+0	PQ	1	+0

GridLab-D result

```
================================================================================
|     Bus Data                                                                 |
================================================================================
 Bus      Voltage          Generation             Load
  #   Mag(pu) Ang(deg)   P (MW)   Q (MVAr)   P (MW)   Q (MVAr)
----- ------- --------  --------  --------  --------  --------
    1  1.060    0.000    232.39    -16.55       -         -
    2  1.045   -4.983     40.00     43.56     21.70     12.70
    3  1.010  -12.725      0.00     25.08     94.20     19.00
    4  1.018  -10.313       -         -       47.80     -3.90
    5  1.020   -8.774       -         -        7.60      1.60
    6  1.070  -14.221      0.00     12.73     11.20      7.50
    7  1.062  -13.360       -         -         -         -
    8  1.090  -13.360      0.00     17.62       -         -
    9  1.056  -14.939       -         -       29.50     16.60
   10  1.051  -15.097       -         -        9.00      5.80
   11  1.057  -14.791       -         -        3.50      1.80
   12  1.055  -15.076       -         -        6.10      1.60
   13  1.050  -15.156       -         -       13.50      5.80
   14  1.036  -16.034       -         -       14.90      5.00
                        --------  --------  --------  --------
               Total:    272.39     82.44    259.00     73.50

================================================================================
```

MatPower result

Future work plan:

Residential load module

PV modeling

Program a runtime class in GLM file to read data from outside data files

Thank you
May 21, 2010