Also check “joseconto.blogspot.com”

MPjobs allows several instances of PSSe to run same PSSe script (\*.idv or \*.py) on different data sets using multi-processing modules available in Python 2.7 and later (installed with PSSe).

A power system study has several processing tasks where some tasks are done in series. Tasks suitable for parallel processing can use MPjobs to minimize overall study runtime but they might still go through a final process to aggregate the output data for reports, plotting, etc. as needed.

In a dynamic run demo, MPjobs will run in parallel 16 fault events for a 5-sec. simulation on a 8-cpu pc, by running first 8 faults using 8-cpus and then the remaining 8 faults on the 8-cpus.

Timing Performance

|  |  |
| --- | --- |
| CPU’s  Tn | |
| PSSe v33  5k-bus case,  16 normal clearing faults for 5 sec. simulation,  1/4 cycle step size,  busdim = 80k,  3249 channels | |  |  |  | | --- | --- | --- | | **time sec** | **CPU** | **Time norm.** | | 3386.13 | 1 | 1 | | 1234.89 | 2 | 0.36 | | 786.64 | 4 | 0.23 | | 581.68 | 8 | 0.17 | |

As an example for loadflow, MPjobs would run activity ACCC (contingency analysis) on n-cases using the same set of input data files (\*.sub,\*.mon, \*.con) on a fixed number of CPUs set by the users (of course, less than total number of available cpus ;)

MPjobs uses two python scripts and one data input file in addition to other user data needed for the study:

MPjobs.py – It contains the code for parallelizing the multiple calls to the target python script. Its designed is such a way that it does not have a dependency on PSSe or the target python script. No changes to this code are expected for common parallel runs.

\*.ini (i.e.: mpjobs.ini) - text file with data input to run the PSSe process. It also contains the name of the target python script and files names to read the independent variables (up to three, X, Y, Z vars) being sent to the target python script for each run.

Data entry in an INI file:

Xvec = [5,10,15,20,25,30] /<- list input

//Xvec = frange(5.0,35.0,5.0] /<- commented data line

mycnv = CASEs\savnw\_flat33\_cnv.sav /<-string, no quotes

LOGspath = LOGs\ /path name

initLoading= 0.6 /float

x.py (i.e.: run\_accc1D.py) – the target python script that effectively run the PSSe process. In the case of a contingency analysis, this python script will include all PSSe activities to create a DFAX files and then run an ACCC activity at the same time, it will receive from MPjobs.py all data needed, contained in a dictionary structure (python data type).

Data received by x.py from mpjobs after reading the INI file will retain its data type:

In INI file in x.py value type

initLoading My[‘INITLOADING’] 0.6 float

LOGspath My[‘LOGSPATH’] ‘LOGs\’ string

All functions and module imports made at mpjobs are available to x.py, together with the study data in dictionary form, My.

In general, if an application can be run under DOS, with line argument or data contained in a file, then such application runs can be also parallelized using MPjobs. It is all about how the target python script is coded!!

MPjobs.py has been configured to run data sets with three independent variables (a 3D run or a triple loop run of Z, Y, X vars):

* 1D runs: a single list of variables, in addition to other study data.

(i.e.: In dynamics runs, a list of faults names = X vars, say 100 names -> 100 dynamic runs)

* 2D runs: two independent lists of variables

(i.e.: In dynamics runs, a list of faults names = X vars (100 names) and

a list of base case names = Y vars (10 cases) -> 1000 dynamic runs)

* 3D runs: three independent lists of variables

(i.e.: In dynamics runs, a list of faults names = X vars,

a list of base case names = Y vars, and

a list of a study region load levels = Z vars)

MPjobs can also run in python 2.5 (installed with PSSe v.32) after installation of the add-on module ‘multiprocessing’ (at the Net:” <https://pypi.python.org/pypi/multiprocessing/>”) which might require admin right on your pc.

# PSSe v.33 or PSSe v.34 with python v.2.7 or v.3.7

At the working folder, open a CMD window (=DOS window) by double-clicking on the link appropriate for your set up:

\_dos33 link: to run PSSe v.33 with python 2.7

\_dos3427 link: to run PSSe v.34 with python 2.7

\_dos3437 link: to run PSSe v.34 with python 3.7

Once the CMD window is open, run the demo data set as:

[Warning: update the \*.ini file used as data input and activate the corresponding "psspypath" for your set up:

//PsspyPath = C:\Program Files (x86)\PTI\PSSE33\PSSBIN

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY27

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY37

]

Use the command "mpjobs27" to run to run PSSe v.33 with python 2.7:

[In \*.ini file:

PsspyPath = C:\Program Files (x86)\PTI\PSSE33\PSSBIN

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY27

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY37

]

c:\..>mpjobs27 cmld

Use the command "mpjobs27" to run to run PSSe v.34 with python 2.7:

[In \*.ini file:

//PsspyPath = C:\Program Files (x86)\PTI\PSSE33\PSSBIN

PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY27

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY37

]

c:\..>mpjobs27 cmld

Use the command "mpjobs37" to run to run PSSe v.33 or PSSe v.34 with python 2.7:

[In \*.ini file:

//PsspyPath = C:\Program Files (x86)\PTI\PSSE33\PSSBIN

//PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY27

PsspyPath = C:\Program Files (x86)\PTI\PSSE34\PSSPY37

]

c:\..>mpjobs27 cmld

# Example 1 - Loadflow demo

Data preparation

For parallelizing multiple ACCC run where the base case is the independent variable, a text file named ‘sav.lst ’ is created, containing the names of the base cases to be used:

Sav.lst:

Savnw

Savnw33a

The ‘mpACCC\_1d.ini’ file has all data needed for this run, including the name of the target python script:

..

Script = SCRIPTs\accc\_1D.py

Xfile = CASEs\sav.lst

..

File names can have full paths or new vars can be defined to hold those path values.

A few rules about data input using an INI file:

* comments start with or from /
* path ending with '\' required
* in-line comments are stripped before reading keyword value
* string shall be empty, not set to be ='' which is read to has two chars of '

The target python script (‘accc\_1D.py’) will contain the code to run PSSe activities [execution of DFAX and ACCC activities], utilizing the data made available by MPjobs throught the dictionary My. This target python code will also be able to parse data and create new variables based on the received independents variable, like creating unique file names to output data:

xvarpath= os.path.dirname(My['XFILE']) #=CASEs\

xvarkey,xext = os.path.splitext(My['XVAR']) #= savnw & ‘’

..

My['MYSAV'] = '%s\%s'%(xvarpath,My['XVAR']) # = CASEs\ savnw

This way, we are able to locate the base case to open for the first instance of a PSSe run.

To run this loadflow demo in PSSe 33 with python 2.7, open a CMD window using the link “\_dos33” and type the command + inifile, without extension:

C:..>mpjobs27 mpACCC\_1d

Once the parallel runs are completed, some output files, \*.dfx, \*.acc, will be created in the “ACCC” folder.

# Example 2 – Dynamic run demo

If no inifile is entered, the program default to read study data from ‘mpjobs.ini’

To run this loadflow demo in PSSe 33 with python 2.7, open a CMD window using the link “\_dos33” and type the command:

C:..>mpjobs27

The data in ‘mpjobs.ini’ will call ‘run\_faults1D.py’ as the target python script, to run dynamics runs for faults scenarios.

As an example of a 2D process, using two independent variables [a double loop run], the data in ‘mp2D.ini’ will identify ‘run\_faults2D.py’ as the target python script, to run dynamics runs for two different base cases and 16 fault scenarios.

To run this loadflow demo in PSSe 33 with python 2.7, open a CMD window using the link “\_dos33” and type the command + ini filename with no extension:

C:..>mpjobs27 mp2d

# MPjobs & Plotting

Plotting in PSSe utilizes PSSPLT.exe, a tool that does not support python. ‘Plotting automation’ is done using idvs files containing the PSSPLT activities leading to a plot on screen or as a postscript file (\*.ps).

For a few plots, PSSPLT is recommended, but for those instances where parallel processing will help, MPjobs could be used to run PSSPLT in parallel as the following demo:

mpplots.bat – batch file to call mpjobs with ‘mpplots1D’ as the ini file argument

mpplots1D.ini – data input to plot from OUTs files generated using mpjobs.ini

Plots \*.ps files are saved at a location specified in “\_plot\_faults.idv”, the idv for PSSPLT

To convert \*.ps to \*.pdf, I use Adobe Distiller.

[Note: Feel free to edit this document, add your own examples and pictures, …, but send me a copy;]