"PARALLEL" ENERGYPLUS AND THE DEVELOPMENT OF A PARAMETRIC ANALYSIS TOOL

Yi Zhang[1]
[1]Institute of Energy and Sustainable Development, De Montfort University, Leicester, UK

ABSTRACT

Parametric analysis is a powerful method for exploring alternative design options and establishing variable dependency therefore design guide. The text-based user interface of EnergyPlus makes it a perfect simulation tool for automated (or scripted) parametric analysis. Since the number of simulations required for parametric analysis tend to be large, a software utility that may take advantage of the ever-increasing desktop computing power is desirable. "Parallelism", in its broad sense of running more calculations simultaneously, comes naturally into our view. Two implementations have been tested on a single-box dual-core PC and a 256-core Beowulf Cluster. This paper presents the development of the Java tool that prepares the parametric runs, as well as the performance enhancement achieved on different platforms.

INTRODUCTION

Parametric analysis is often needed for exploring design options, especially when a global optimization method is unavailable, or the optimization result is in doubt. Parametric analysis can also be applied to all design variables simultaneously, which forms an exhaustive search approach that, providing that the search grid is fine enough, will guarantee the global optimal solution. This is potentially a very useful method.

In order to perform complex parametric analysis on multiple parameters with more than a handful alternative values each, two crucial ingredients are required – a simulation model that supports parametric runs, and a tool to generate commands for those runs and collect results afterwards. EnergyPlus (EERE, 2009) is suitable tool to run parametric models because of its command-line interface and text-based model definition. It also works with Linux therefore can be used on computer clusters. We shall see the benefit of this in the "Experiment" section. There have been a few tools available for EnergyPlus users.

EP-Macro (EERE, 2008)

According to the user manual of the auxiliary programs of EnergyPlus, the EP-Macro program is intended for advanced users who need to prepare input manually, by providing the following functions that are relevant to parametric runs:

- Incorporating blocks of IDF in external files
- Conditioned activation / deactivation of blocks
- Defining parametric blocks
- Defining parameters and performing arithmetic and logic operations on parameters and values

EP-Macro itself is not a parametric tool because it does not implement any loop nor specify alternative parameter values. However, it can be used with an external parametric shell to define any complex parametric runs. This will be further discussed in the later part of the paper.

EzPlus-Parm (EERE, 2002)

"EzPlus-Parm" was developed by the Derringer Group at Berkeley, California in 2002. It is one of earliest parametric tools for EnergyPlus. Although still listed on EnergyPlus' website, the tool is no longer available. EzPlus-Parm claimed to "simplify running multiple parametric EnergyPlus simulations" by assisting a user to organize and edit all needed files, and write AWK scripts to specify parametric runs and result collection. Essentially EzPlus-Parm streamlined the processes of a parametric analysis by linking various software tools (EnergyPlus, AWK interpreter, Ms Excel) in one GUI. It does require the user to be familiar with AWK, however, because all input and output methods have to be specified in this syntax. The existing user base of EzPlus-Parm is unclear.

COMFEN 2.0 Beta

"COMFEN is a tool designed to support the systematic evaluation of alternative fenestration systems for project-specific commercial building applications." (LBNL Windows and Daylighting, 2009). It allows users to specify details of up to 4 different fenestration façade systems in an Excel-based user interface and compare the results of EnergyPlus simulations. Although it is not a tool for parametric analysis, it is relatively easy to be converted to one with some extra VBA (Microsoft Visual Basic for Applications) scripts. It can also be used for defining alternative glazing systems and shading controls with the provided libraries.

GenOpt

GenOpt (LBNL, 2008), developed by Dr Michael Wetter, was first introduced in 2000. It is a collection of optimization algorithms bundled with a generic interface that can work with many simulation tools including EnergyPlus and TRNSYS. GenOpt supports parametric runs on an orthogonal, equidistant grid. Some casual users may find its sophisticated and powerful mechanism for defining optimization problems and coupling with simulation tools overly complex. The most significant limitation, however, is that GenOpt does not support non-numeric variables, nor arbitrary list of alternative values.

DesignBuilder V2

Providers of commercial frontend to EnergyPlus, e.g. DesignBuilder (DesignBuilder Sofware Ltd., 2009), are also considering implementing full parametric capability in their software, e.g. selecting variables and report results in one screen. Speculatively this could be the best solution since the quality of the software can be controlled by the vendor. To the research community, however, commercial software often means less flexibility and more cost.

This paper describes the development of a parametric shell called "jEPlus". This tool interfaces with EnergyPlus in the same way as GenOpt does, except giving users more control on the graphical interface. First, a user prepares an IDF file by putting tags (special search strings) at the places of the parameters. jEPlus chooses the next set of values for the parameters according to the information provided by the user. It then searches the IDF file for the tags and accordingly replaces them with the new values. EnergyPlus is subsequently called to run the simulation and produce results. A number of distinct features are offered by jEPlus:

- Unique parameter tree for defining complex parametric runs;
- Simulation results are collected in both tables and databases;
- Completely written in Java to be platform-neutral

PARAMETRIC DEFINITION

Obviously, the ability to define parametric runs is the elementary function of a parametric analysis tool. jEPlus uses a *Tree* structure to organize parameters and their values.

The Parameter Tree

Traditional parametric studies are designed for analysing sensitivity of a model to a number of independent parameters. In engineering design however, investigations of the effects of different combinations of dependent parameters are often required. For example, to study the effect of window sizes, four parameters have to be considered together: the coordinates (x, y) of one corner, and the height and width of the window. The choices of the four values are constrained by the geometry of the wall, as well as the overall size (in m^2) of glazing. To encode the dependency between parameters, a *Tree* structure is necessary.

The definition of window size parameters is not a particularly good example because it can be easily handled using one parameter (e.g. the glazing ratio in DesignBuilder) with some pre-processing. However, this example is used to explain the concept of the Parameter Tree.

Assume that, on a $5x3m^2$ wall, the impact of glazing area is to be evaluated. The lower edge and the height of the window are fixed to 1.0m and 1.5m, respectively. There are two adjustable parameters, the left edge ($\boldsymbol{x}$) and the width ($\boldsymbol{w}$). The two parameters are constrained by the width of the wall (5m). For example, if $\boldsymbol{x}$=2m, $w \in [0,3]$m. Now consider $\boldsymbol{x}$ is varied between 1.0m and 2.0m at a step of 1.0m; whereas $\boldsymbol{w}$ is also varied at a step of 0.5m, to encode this without arithmetic calculations, the following syntax have to be used: {{$\boldsymbol{x}$=1.0, $\boldsymbol{w}$={0.5, 1.0, …4.0}}, {$\boldsymbol{x}$=2.0, $\boldsymbol{w}$={0.5, 1.0, …3.0}}}.

It can be presented as part of the *Tree* structure in *Figure 1*, i.e. P2 represents $\boldsymbol{x}$ with value {1.0}; P3 represents $\boldsymbol{w}$ with alternative values {0.5, 1.0, …4.0}; P4 represents $\boldsymbol{x}$ with value {2.0}; and finally, P5 represents $\boldsymbol{w}$ with alternative values {0.5, 1.0, …3.0}. A traverse of the tree will give us all combinations of the alternative values of the parameters.

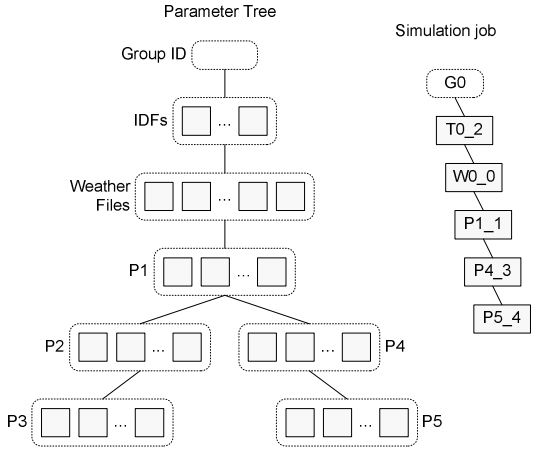

Figure 1 Parameter tree and simulation job

Figure 1 illustrates the full structure of the Parameter Tree used in jEPlus. The first three nodes in the tree, i.e. the ***Group ID*** (identified by "***G***" + user-specified integer), the ***IDFs*** ("***T***") and the ***Weather Files*** ("***W***") are implicit and default to all projects.

Each simulation job is a path from the root node of the tree to a leaf (the end of a branch) of the tree, with each node containing an optional value of the corresponding parameter. As a result, the total

number of jobs encoded in the tree equals the total number of paths from the root to the leaves. *Figure 2* shows an example Parameter Tree defined in a project. Each row contains the definition of a parameter.

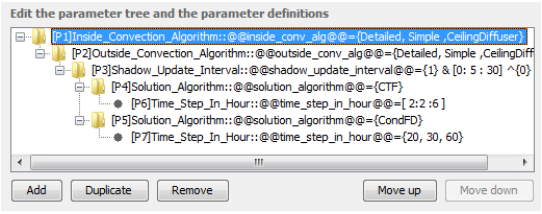

Figure 2 An example parameter tree

Parameter definition

A parameter to be incorporated in the parametric analysis is specified with three essential elements, a unique ***ID***, a ***Search string***, and a list of ***Alternative values***. The ***ID*** is short string used for identifying the parameter. It is also used to form part of the job title as well as the work directory name in which the job is to be executed. The formulation of the job titles will be further explained in the "Result collection" section.

The ***Search string*** (or "tag") is a character sequence to be planted in the IDF files to identify the location of a value to be later inserted. This string must not naturally occur in an IDF file; therefore it is recommended to include special characters (e.g. '`@`') that are not used in the standard EnergyPlus syntax. Note that jEPlus only searches and replaces one occurrence of a search string in each job. A user must ensure that there is only one instance of a search string, as well as all search strings in the IDF will be replaced within the job. A validation facility has been provided in the jEPlus GUI.

The "***Alternative values***" is a list of strings to be used one at a time in the parametric jobs. jEPlus supports three types of alternative values: ***Discrete***, ***Integer*** and ***Double***. The syntax for specifying the list of values is explained next. *Figure 3* shows an example of parameter definition. There are two extra fields, i.e. "***Name***" and "***Description***", which are recorded in the output files for reference, but not used in the simulation.

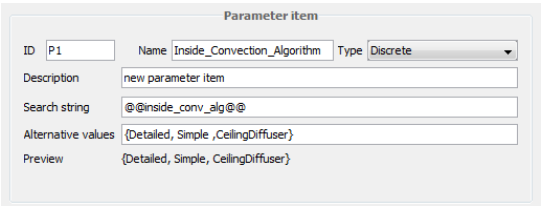

Figure 3 Parameter definition

Syntax for Alternative Values

For the "***Discrete***" type of parameters, the values can be specified with a comma ('`,`') delimited list enclosed in a pair of curly brackets ('`{}`'), e.g.

```
{Detailed, Simple, CeilingDiffuser}
```

For the "***Integer***" and "***Double***" parameters, square brackets ('`[]`') and union/exclusion operations ('`&`' '`^`') are accepted in addition to the curly brackets ('`{}`'). The square brackets are used to define a number series with a uniform interval. For example, the list `{1,3,5,7,9}` can be specified using `[1:2:9]`. Colons ('`:`') are used to separate the ***Start Value***, the ***Interval***, and the ***End Value***. Please note the last value in the resultant list is unnecessarily the ***End Value***.

The union operator ('`&`') combines the elements in two lists. For example, `[1:2:5]&{2,4,6}` is equivalent to `{1,3,5,2,4,6}` (Note that the list is not sorted). The exclusion operator ('`^`') removes elements in the right-hand list from the left-hand list, e.g. `[-2:1:6]^{2,4,6}` gives `{-2,-1,0,1,3,5}`. The operators are processed in the left-to-right order. In the current version, grouping with parentheses is not supported. The following example shows the use of all supported operations: `{1}&[0:5:30]^{0}`, which results in `{1,5,10,15,20,25,30}`.

More Possibilities with EP-Macro

Using EP-Macro in the pre-processing step can significantly extend the ability of jEPlus. First, it allows replacement of IDF code blocks by using the "`##include`" clause. Secondly, the "`##def`" command allows a block of input text to be defined with arguments, therefore enables associating values at different locations with one search string. Used in conjunction with the "`##if/##elseif/##else/##endif`" and "`##ifdef/ ##ifndef`", "`##def`" can also be used to include multiple buildings/systems in one input file, therefore reducing the number of IDF files to be handled. The arithmetic functions will be useful for calculating dependent parameters, therefore reducing the total number of parameters to be specified for parametric runs. EP-Macro will be incorporated in jEPlus in the near future.

RESULT COLLECTION

It is almost impossible to develop a generic GUI for the post-processing of the results of parametric runs. This is mainly because each research problem will need its own way to process and present data. As a result, jEPlus provides only a convenient way to transfer the (large amount of) results to another software tool (e.g. Excel, Database, or Matlab) for post processing.

Index files in CSV Format

Information of jobs to be executed is stored in a series of index files in both Comma Separated Values (CSV) and Structured Query Language (SQL)

formats. The definition of each parameter is stored in a CSV file, in which the first row contains the column headers. Figure 3 shows an example parameter index file. Figure 4 shows an IDF index, which's fields are different to other parameter's index files.

IndexP6.csv

```
Index, ID, NAME, Type, DISCRIPTION, SEARCHSTRING, VALUE
0, P6, Time_Step_In_Hour, 0, new parameter item, @@time_step_in_hour@@, 2
1, P6, Time_Step_In_Hour, 0, new parameter item, @@time_step_in_hour@@, 4
2, P6, Time_Step_In_Hour, 0, new parameter item, @@time_step_in_hour@@, 6
```

Figure 4 Content of a parameter index file

IndexIDF.csv

```
ID, GROUP, NAME, DIRECTORY, FILE
0, G_1, T0, ./, tmp1.idf
```

Figure 5 Content of an IDF index file

IndexJobs.csv

```
INDEX, GROUP_ID, IDF_ID, WTHR_ID, P1, P2, P3, P4, P5, P6, P7, JOBNAME, WORKDIR,
0, 1, 0, 0, 0, 0, 0, 0, null, 0, null, G_1-T_0-W_0-P1_0-P2_0-P3_0-P4_0-P6_0, ~/E
1, 1, 0, 0, 0, 0, 0, 0, null, 1, null, G_1-T_0-W_0-P1_0-P2_0-P3_0-P4_0-P6_1, ~/E
2, 1, 0, 0, 0, 0, 0, 0, null, 2, null, G_1-T_0-W_0-P1_0-P2_0-P3_0-P4_0-P6_2, ~/E
3, 1, 0, 0, 0, 0, 0, 0, null, 0, null, 0, G_1-T_0-W_0-P1_0-P2_0-P3_0-P5_0-P7_0, ~/E
4, 1, 0, 0, 0, 0, 0, 0, null, 0, null, 1, G_1-T_0-W_0-P1_0-P2_0-P3_0-P5_0-P7_1, ~/E
5, 1, 0, 0, 0, 0, 0, 0, null, 0, null, 2, G_1-T_0-W_0-P1_0-P2_0-P3_0-P5_0-P7_2, ~/E
6, 1, 0, 0, 0, 0, 0, 1, 0, null, 0, null, G_1-T_0-W_0-P1_0-P2_0-P3_1-P4_0-P6_0, ~/E
7, 1, 0, 0, 0, 0, 0, 1, 0, null, 1, null, G_1-T_0-W_0-P1_0-P2_0-P3_1-P4_0-P6_1, ~/E
8, 1, 0, 0, 0, 0, 0, 1, 0, null, 2, null, G_1-T_0-W_0-P1_0-P2_0-P3_1-P4_0-P6_2, ~/E
9, 1, 0, 0, 0, 0, 0, 1, null, 0, null, 0, G_1-T_0-W_0-P1_0-P2_0-P3_1-P5_0-P7_0, ~/E
10, 1, 0, 0, 0, 0, 0, 1, null, 0, null, 1, G_1-T_0-W_0-P1_0-P2_0-P3_1-P5_0-P7_1, ~/
11, 1, 0, 0, 0, 0, 0, 1, null, 0, null, 2, G_1-T_0-W_0-P1_0-P2_0-P3_1-P5_0-P7_2, ~/
12, 1, 0, 0, 0, 0, 0, 2, 0, null, 0, null, G_1-T_0-W_0-P1_0-P2_0-P3_2-P4_0-P6_0, ~/
13, 1, 0, 0, 0, 0, 0, 2, 0, null, 1, null, G_1-T_0-W_0-P1_0-P2_0-P3_2-P4_0-P6_1, ~/
14, 1, 0, 0, 0, 0, 0, 2, 0, null, 2, null, G_1-T_0-W_0-P1_0-P2_0-P3_2-P4_0-P6_2, ~/
15, 1, 0, 0, 0, 0, 0, 2, null, 0, null, 0, G_1-T_0-W_0-P1_0-P2_0-P3_2-P5_0-P7_0, ~/
16, 1, 0, 0, 0, 0, 0, 2, null, 0, null, 1, G_1-T_0-W_0-P1_0-P2_0-P3_2-P5_0-P7_1, ~/
17, 1, 0, 0, 0, 0, 0, 2, null, 0, null, 2, G_1-T_0-W_0-P1_0-P2_0-P3_2-P5_0-P7_2, ~/
18, 1, 0, 0, 0, 0, 3, 0, null, 0, null, G_1-T_0-W_0-P1_0-P2_0-P3_3-P4_0-P6_0, ~/
```

Figure 6 The Jobs index file

The complete list of jobs identified for the parametric run is listed in the file named "`IndexJobs.csv`". Each job has a unique serial number ("***Index***"), as well as a unique "***JobName***" in the form of:

`G_[group num]-T_[IDF file num]-W_[Weather file num]-[Parameter 1's ID]_[val num]-…`

Index numbers of the Group, IDF file, Weather file and Parameter values used in each job are listed in the same row as the ***JobName*** and the serial number. This information can later be used to reference the actual file names or values in the IDF/Weather and parameter index files.

SQL File for Indexes

A SQL file will be generated for easy importing of job indexes into database software. The commands in the SQL file ("`jobdb.sql`" by default) perform the following operations:

1. Create a new database with a user-specified name (e.g. "`EpResultDB`");
2. Use the database;
3. Create table "`[prefix]_IndexIDF`". [prefix] is a user specified string to distinguish tables for this batch of jobs from others;
4. Insert records (list of IDF files used in the jobs);
5. Create table "`[prefix]_IndexWthr`";
6. Insert records (list of Weather files used in the jobs);
7. Create table "`[prefix]_Index[Param 1's ID]`";
8. Insert records (list of values for Parameter 1);
9. (Repeat steps 7 and 8 for all parameters)

The generated SQL commands have been tested to work with *MySQL*.

Get Result Data

By default, the utility `ReadVarsESO` supplied with EnergyPlus is used to extract results from the standard output file ("`eplusout.eso`"). The `ReadVarsESO` is controlled by the contents of "`my.rvi`" file. For more information, please refer to the Output section in the Input Output Reference in the EnergyPlus documentation. `ReadVarsESO` produces a result table in CSV format. An extra column containing the serial number of the current job is added to the table, which will be subsequently renamed to "`[the job's name].csv`", and copied to the root directory. A user can then import the files to the database (or other software), where the results can be accessed in conjunction with the indexes.

IMPLEMENTATION

Some details of the implementation including the GUI are discussed bellow.

System Configuration

Minimum configuration is needed to setup the work environment for EnergyPlus. jEPlus uses the binary directory and commands that are default to the operating system on which it is running. If EnergyPlus is not installed in the default directory, a user can manually locate the executables by pressing the browse button.

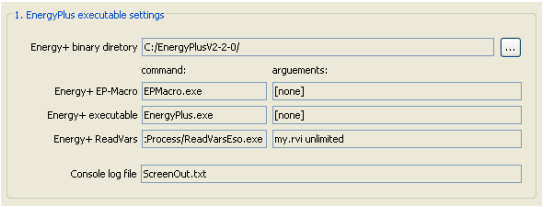

Figure 7 System configuration

Execution Control

jEPlus is primarily designed to run parametric jobs on the local computer, either it has a Windows or Linux platform. The Internal Batch Controller is designed to take advantage of the modern multi-processor/multi-core systems. A user can specify the number of processor cores to use for running the parametric jobs. The batch controller will start a new job as soon as an allocated core becomes available. A small delay time (5,000ms by default) is inserted to minimize the chances that several jobs start at the same time, which may cause congestion in the file

system. A batch controller for running jobs on computer clusters is currently under development.

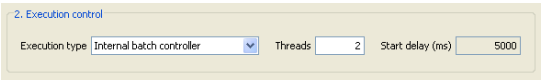

Figure 8 Execution control

Parametric Run Details

Figure 9 shows the basic settings for a parametric run. A user can specify a group ID that helps identify different batches. As mentioned before, jEPlus allows parametric runs with multiple IDF files and Weather files. In the current version, a user has to select these files using the file browsers. A simple editor is provided so the user can edit the IDF files and the "`my.rvi`" file without leaving the GUI.

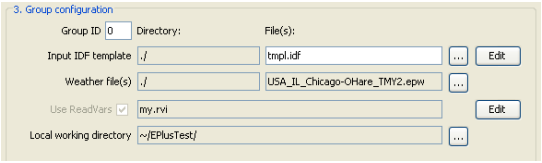

Figure 9 Job group configuration

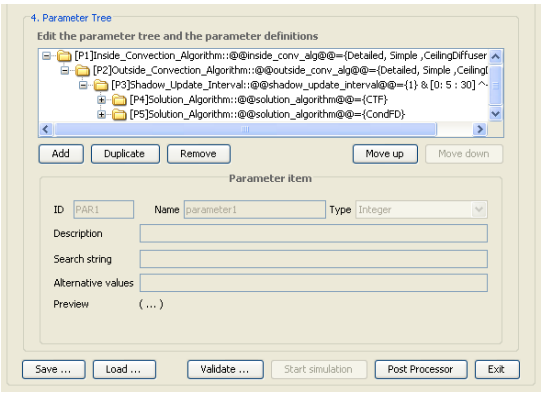

Figure 10 Parameter definition

Figure 10 shows the Parameter Tree editor, in which parameters can be defined while some commands will help a user arrange the parameters in the tree. The specifications of a parametric run can be saved to or loaded from the files system. The "`Validate`" command generates the list of jobs from the parameter tree. Checks are carried out during the compilation. The user is prompted for the total number of jobs to be executed if validation has been successful, upon which the "`Start Simulation`" command is enabled.

EXPERIMENT

A simple experiment was carried out to demonstrate the use of jEPlus. We were also interested in how different simulation algorithms and options may affect the computing time. The DOE Benchmark Commercial Building (Deru et al, 2008), the model of a 47-zone secondary school, was used to run annual simulation. The parameters considered include following:

1. Solution algorithm – choice between Conduction Transfer Functions (*CTF*) and Conduction Finite Difference (*CondFD*)
2. Inside and outside convection algorithms – *Detailed, simple* or *CeilingDiffuser*
3. Shadow update interval – 1 to 30 at step of 5
4. Time steps in hour – 2/4/6 for *CTF,* 20/40/60 for *CondFD*.

There are total 189 jobs with *CTF* algorithm, and 189 jobs with *CondFD* algorithm. The *CTF* jobs were ran on both a dual-core desktop PC and a 256-core cluster. The *CondFD* jobs were only run on the cluster. Only simulation time was collected from the results.

Table 1
Computing times

Time (hr)		CTF jobs (189)		CondFD jobs (189)
		PC Core2D E6600 2 cores 2 threads	Cluster Xeon E5440 256 cores upto 128 threads	Cluster Xeon E5440 256 cores upto 128 threads
Job	Mean	0.72	1.24	6.01
	SD	0.05	0.14	1.84
	Sum	135.47	230.70	1120.64
Batch (time saved)		68.45 (-49.5%)	*8.20* (-96.4%*)*	*15.90* (-98.6%*)*

* The CTF jobs and the CondFD jobs are submitted at the same time to the Cluster. The total computing hours are counted from the time of submission to the completion of the last job in the group. '15.90hrs' is also the total time for all jobs on the Cluster.

Table 1 shows a summary of the simulation times, where "job" stands for one simulation; "Batch" includes all simulation runs in the category. The mean and standard deviation of the CPU time (hours) required by the jobs are shown in the first two rows. The sum of CPU core-hours for all jobs is listed in the third row. Since the jobs in a batch are ran in parallel, the actual time required to complete the execution of a batch is significantly less than the total CPU-hours.

To perform a single simulation, the PC required only 60% of the time required for the cluster, despite the slightly lower CPU speed (2.4GHz vs. 2.66GHz). This may be a result of the overhead associated with the communications between the nodes of the cluster.

As a batch, however, the speed increase achieved on the PC by running more jobs at once is almost linear to the number of cores employed. On the cluster, running 128 threads in parallel achieved 85 times

speed boost. Although the ratio is lower than that achieved on the PC, the cluster has nevertheless finished 378 jobs in 15.9 hours, which would otherwise take nearly 2 months if executed in a single thread.

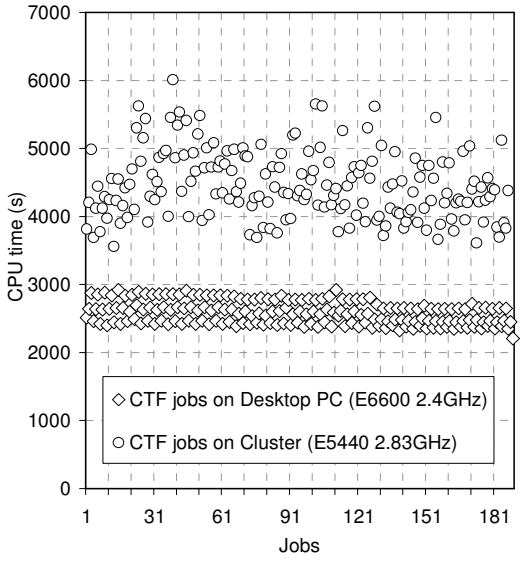

Figure 11 CPU times for the jobs

Figure 11 shows the computing time of each job on the PC and the cluster. On the PC, a tight pattern is observed, in which jobs are roughly grouped in three rows that represent 2, 4 and 6 time steps in an hour, respectively. Slight inverse correlation between the job number and the computing time is directly visible, too. The job numbers are primarily ordered by Inside Convection Algorithm ("*Detailed*", "*Simple*", "*CeilingDiffuser*"), and secondarily ordered by Outside Convection Alogrithm ("*Detailed*", "*Simple*", "*CeilingDiffuser*").

No clear pattern of computing times is observed on the cluster. This may be attributed to the existing loads on the cluster nodes. At the time of the experiment, half of the processor cores (128) have already engaged to other jobs. The computing time of each EnergyPlus job was dependent on the type and characteristics of other jobs running on the same node.

Despite the noises, correlations are still observed statistically between the choices of the Inside Wall Convection Algorithms, Shadow Update Interval and Time Step in Hour, and the computing time. Figure 12 shows the 50-point moving average of the option numbers (e.g. 0-2 for Convection Algorithm and Time Step; 0-6 for Shadow Update Interval) against computing time on the x-axis (expressed as percent increase based on the shorted run). The computing time shows a positive relation to the options of Time Steps in Hour, whereas negatively correlated to the options for the Convection Algorithm and the Shadow Update Interval.

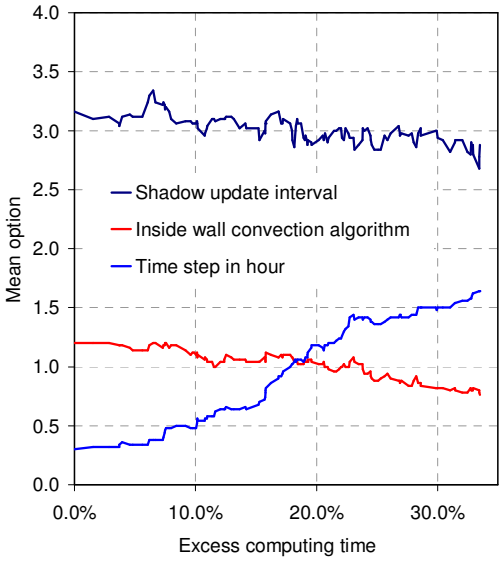

Figure 12 Correlation between CPU time and parameters

DISCUSSION

In this paper, we presented the development of a Java shell (jEPlus) for EnergyPlus. jEPlus is designed to assist building researchers preparing and executing parametric runs with EnergyPlus. Some useful and distinctive features have been implemented. Firstly, the parameters are organized in a Tree structure, which makes it possible to define complex dependencies between parameters. Secondly, the simulation results can be collected in both CSV tables and databases with associated indexes for the IDF files, the Weather files and the Parameter values. jEPlus also provides a Batch Job Controller that can take advantage of the modern multi-core computers. The software tool is completely written in Java and works on multiple platforms.

More features have been identified for future implementation. One of such opportunities is to make use of the EP-Marco utility, which's pre-processing functions, may substantially extend the ability of jEPlus.

REFERENCES

DesignBuilder Software Ltd. (2009) "DesignBuilder Features", http://www.designbuilder.co.uk/content/ view/ 6/14/

Deru, M.; Griffith, B.; Long, N.; Benne, K.; Torcellini, P; Halverson, M.; Winiarski, D.; Liu, B.; Crawley, D. (2008). "DOE Commercial Building Resarch: Benchmarks for Commercial Buildings." Washington, DC: U.S. Department of Energy, Energy Efficiency and Renewable Energy, Office of Building Technologies.

EERE, (2009) “EnergyPlus Energy Simulation Software”, http://apps1.eere.energy.gov/ buildings/ energyplus/

EERE, (2008) “EnergyPlus Auxiliary Programs Manual”, http://apps1.eere.energy.gov/ buildings/ energyplus/ pdfs/auxiliaryprograms.pdf

EERE, (2002) “Third-Party EnergyPlus Tools - EzPlus-Parm”, http://apps1.eere.energy.gov/ buildings/ energyplus/ third_party_tools.cfm #drawezplus

LBNL Windows and Daylighting, (2009) “COMFEN 2.0 Beta”, http://windows.lbl.gov/ software/ comfen/ 2/

LBNL, (2008) “GenOpt – Generic Optimization Program”, http://gundog.lbl.gov/GO/