**KES Transactions on Sustainable Design and Manufacturing I**
Sustainable Design and Manufacturing 2014 : pp.386-397 : Paper sdm14-033

# Parallel computing for supporting complex energy simulation and optimization

Ioan Petri[1], Haijiang Li[1], Yacine Rezgui[1], Chunfeng Yang[1,2] , Bejay Jayan[1] and Baris Yuce[1]

[1]School of Engineering, Cardiff University, UK
BRE Institute of Sustainable Engineering,
Cardiff University, UK
[2]State Key Laboratory of Structural Analysis for Industrial Equipments
Department of Engineering Mechanics, Faculty of Vehicle Engineering and Mechanics Dalian University of Technology, Dalian, China. 116024

**Abstract** With the emergence of the new major challenge on communities regarding global warming, significant researching efforts have been recently directed towards simulation based building energy optimisation with the overall objective of reducing energy consumption. Energy simulation and optimisation identifies a class of applications that demands high performance processing power in order to be realised within a feasible time-frame. The problem becomes increasingly complex when addressing energy simulation and optimisation in large scale buildings such as sport facilities where the generation of optimal set points can be timing inefficient. Parallel processing on computing systems has enabled a wider range of complex simulations to be undertaken and are particularly useful due to their capability to handle and analyse massive amounts of data at high speed. In particular, such systems can provide the mechanisms for reducing the time for running complex simulations and optimising the various aspects within a problem. In this paper we present how parallel computing systems can be efficiently used for running and deploying EnergyPlus based simulations and optimisations in order to fulfil a number of energy related objectives. We adopt a comparison based approach by evaluating the performances of our system over a number of relevant scenarios.

**Keywords**: High Performance Computing, Cluster Computing, EnergyPlus, Simulation, Optimisation

## 1. Introduction

The adoption of computing systems in the engineering community is at early stages. In the context of building energy simulation based optimisation there are a number of proposed computing solutions that have been developed, enhanced and are in use throughout the building energy community [1], [2], [3]. Scientists tend to use standard computational methods for solving the increasing complexity of the various energy problems. Lately, as with the new major challenge on communities regarding global warming, significant researching efforts have been directed towards simulation based building energy optimisation with the overall objective of reducing energy consumption.
Parallel computing provides the means for running complex processes simultaneously by dividing large chunks of problems into smaller ones. With the multiple classes residing such as cluster computing and grid computing, parallel computing seeks to divide tasks and deploy them on existing forms of computing infrastructures. Whereas parallel computing on cluster based systems is widely

explored and used, parallel computation on High Throughput Computing (HTC) infrastructures such as Condor systems has not been enterely explored. This is mainly because High Throughput Computing systems are focusing on computational tasks which can be processed over a long period of time and imply additional work for schedulling and monitoring. However, in the engineering domain the diversity of tasks and computational problems can be highly increased. Deciding on which computational infrastructure should be used to maximise performances and scalability can be difficult to achieve especially when the problems have various degrees of complexity [4]. On the other hand, High Performance Computing (HPC) based systems work with several nodes, up to thousands, dedicated for groups of users allowing parallel tasks to be executed. Whereas HPC systems save a lot of time for large tasks in terms of the number of floating point operations per second compared to personal computers or small servers, the nodes in HTC systems are regular machines such as desktop computers and usually they do not support parallel algorithms[1].

Thus, HTC systems and HPC systems can provide a number of advantages when used for the right type of problems. However, the decision on the type of infrastructure to be chosen can be difficult to determine especially when the problem to be solved requires the execution of algorithms over a large number of data sets. An example for such a problem is the energy optimisation context where the input data sets can be continuously changing and the algorithm running necessitates sensitivity analysis for each data set result. Nevertheless, a key scenario is the optimisation with genetic algorithms (GA) using principles of natural evolution. In GA optimisation, the process deals with a randomly generated population of individual solutions where good solutions are selected and mixed by recombination to form new population of better solutions. For such an optimisation type, HPC systems are suitable only from the perspective of the associated process duration (the run time is short and the number of runs is huge). On the other end of the spectrum, HTC systems are suitable from the perspective of algorithm complexity where parallelism is necessary, the run time is large and the number of complete runs is relatively short so the correspondent mechanism of queue waiting becomes extremely practical [2].

We tackle the problem of energy optimisation with a hybrid approach by developing an algorithm compatible with the two computing environments -- HTC and HPC systems. More specifically we have created an infrastructure where HTCondor [6] and Torque [5] can run on different types of Energy Plus optimisation tasks. In this paper we propose a framework for deploying Energy Plus [21] based simulation optimisation and enable users to compute tasks within a HTC or HPC based infrastructure. As time and load are extremely important factors especially when dealing with real-time energy simulation and optimisation, we present and test our algorithm in different scenarios. We believe that our solution is applicable not only for complex energy optimisation problems but can be easily extended to other types of problems. The reminder of the paper is as follows: Section 2 and Section 3 explore related models and present the approach we follow in this paper; Section 5 presents the methodology followed by a detailed use case in Section 6. Section 7 presents the results. We present the conclusions in Section 8.

## 2. Related work

In this section we explore several related approaches in the field of energy simulation and optimisation from two perspectives: (i) computing based architectures and (ii) optimisation based solutions.

### 2.1 Computer Architectures

The architectures of parallel machines have emerged over the time from computer architectures with Single-Instruction-Multiple-Data (SIMD) to Multiple-Instruction-Multiple-Data (MIMD) models. The SIMD has appeared as a simple parallel computer model with one instruction unit and several processing units, while the MIMD processors have the ability to execute their tasks in the memory. Shared-memory machines (SIMD) are configured to enable the processors to address the whole memory space and the communication between the tasks is done through read and writes operations on the shared memory. The distributed memory machines (MIMD) have their memory physically distributed among the processors. Each processor can only address its own memory, and communication among the processes executed on different processors is performed by messages passed through the communication network [9].
On the other hand, the SMP machines provide a new mechanism for connection via the fast local networks (Gigabit Ethernet) adding fault-tolerance and excellent cost/performance ratios. This has led to the emergence of clusters of computers (based on Linux OS, Myrinet, off-the-shelf PCs) which is a current trend in parallel computing as the fastest and most powerful computers according to various studies [16]. Parallel computers mainly use a type of UNIX4 operating system with an X-Window based interface. On the market, there are also other operating systems than the UNIX/Linux mainstream, Microsoft Windows 2003 Server, Apple Mac OS, Open VMS, but they do not compete successfully in High-Performance Computing due to various reasons (price, hardware support, scalability, maintenance) [15].

### 2.2. Optimization solutions

Genetic algorithms (GA) are based on principles from natural evolution and can generate an optimal solution from a population of candidate solutions. Generally, GAs begin with a randomly generated population of individual solutions where solutions are selected and mixed by recombination to form new population of better solutions. Genetic algorithms do not guarantee optimal solutions, but they can produce high quality solutions in a reasonable amount time [7]. A disadvantage of GAs is that they require a large number of, sometimes thousands of, evaluations to find adequate solutions for complex optimisation problems [8].
A number of different researching attempts have been developed around generic algorithms and their efficiency in solving optimisation problems. Sefrioui et al. [11] developed a Hierarchical Genetic Algorithms (HGAs) with multi-layered hierarchical topology and multiple models for optimisation problems. The result was composed by a mix of a simple and complex model with a significant improvement in regards to the processing time when compared to complex models. For the problem of

synchronism for migration of various parallel distributed GAs, Alba and Troya (2001) [10] extended the existing results to structured-population GAs and demonstrated linear and even super-linear speedup when run in a cluster of workstations. From the application perspective, Jelasity et al. [12] proposed a tool for automatic learning of algorithm components based on distributed evolutionary algorithms for problem classes. The tool was based on a multi-objective conceptual framework implemented in Java and capable of running distributed experiments on the Internet. Arenas at al. (2002) [13] developed a framework for the automatic distribution of evolutionary algorithm processing through a virtual machine built from a large number of individual computers on the Internet. The benefits of parallel genetic algorithms have been also tested by comparison with a traditional non-parallel GA alongside an optimisation problem [14]. It was identified that parallel GA provide gains not only in terms of computational time, but also in the optimisation outcome. Thus, parallel algorithms can be reliable methods for reducing calculation time of simulation-based GA optimisation. In this paper we develop a parallel genetic algorithm based on NSGA-II compatible with two computing frameworks -- HTCondor and Torque.

## 3. Computing systems and applications

In our study we use two environments such as HTCondor and Torque for deploying our algorithm.
HTCondor is an open-source high throughput computing workload management software framework for a cluster of distributed computer resources. As most of personal computers have more processing power and storage space than the supercomputer of last century is has become possible to build a network of such computers. HTCondor is widely used to exploit the distributed computers to their full potential for computational intensive tasks, such as simulation calculations [3].
The mechanism of task scheduling in HTCondor is different to the mechanism of related systems. In HTCondor system, after the submission, the jobs run until a user tries to use the computer interrupting the processed job and then restarting it on another available machine. HTCondor can be used to manage a cluster of dedicated compute nodes (our case) greatly enhancing the completion time and balancing the load. In our scenario, the machines are part of a dedicated cluster and every task submitted to HTCondor will be processed without interruption.
In Torque system a task contains both the details of the processing to be carried out (name and version of the application, input and output, etc.) and directives for the computer resources needed (number of CPUs, amount of memory). Tasks are run as batch jobs, i.e. in an unattended manner by the user submitting a job to the execution queue. Tasks are managed by a task scheduler, a piece of software which is in charge of allocating the computer resources requested for the task, running the task and reporting back to the user the outcome of the execution. Running a Torque task involves at the minimum the following steps: (i) preparing a submission script and (ii) submitting the task to execution [5].
From the energy simulation perspective, EnergyPlus has been validated as an efficacious tool for running energy simulations [19], [20]. EnergyPlus is an energy

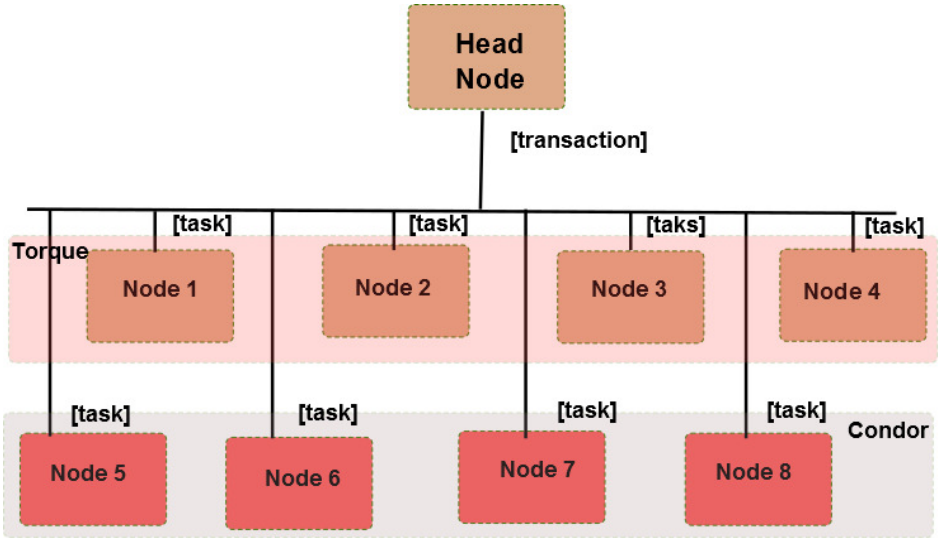


**Figure 1 Cluster infrastructure**

analysis and thermal load simulation tool that can calculate the heating and cooling loads necessary to maintain thermal control setpoints, conditions throughout an secondary HVAC system and coil loads, and the energy consumption of primary plant equipment. In EnergyPlus, inputs and outputs are handled as ASCII (text) files as follows:

- the Input Data Dictionary (IDD) that describes the types (classes) of nput objects and the data associated with each object;
- the Input Data File (IDF) that contains all the data for a particular simulation.
- the Weather Data File (EPW) that contains all the data for exterior climate of a building.

From a computational perspective EnergyPlus necessitates reliable computational infrastructure to run. When dealing with small EnergyPlus models (with small umber of surfaces, zones, and systems), which do not require large amounts of computer memory, computers with faster CPUs are more effective in reducing run time than computers with more memory. For large models, more and faster computer memory including RAM and internal cache may be more effective in reducing run time. The amount of computer memory only helps to a certain point as if the simulation produces multiple reports, the hard drive access speed also becomes important in reducing the run time.

## 5. Methodology

We consider a number of transactions within the set $T = \{t_1, t_2, t_3, ..., t_n\}$ that can take place within the system, where each transaction $t_i$ maps a complex optimisation problem (see figure 2). Each transaction $t_i$ can have an objective (single objective) $ob_i$ or a set of objectives (multi-objective) $[ob_1, ob_2, \ldots, ob_j]$. Each transaction $t_i$ is composed by a set of tasks $t_i = \{ts_1, ts_2, ...,ts_m\}$ which generate a result $r_i = \{ts_1 +$

$t_{s2}$+,..., +$t_{sp}$}, where *n* is determined by the complexity of the optimisation problem to be resolved. Each task $t_{si}$ is mapped as an EnergyPlus simulation.

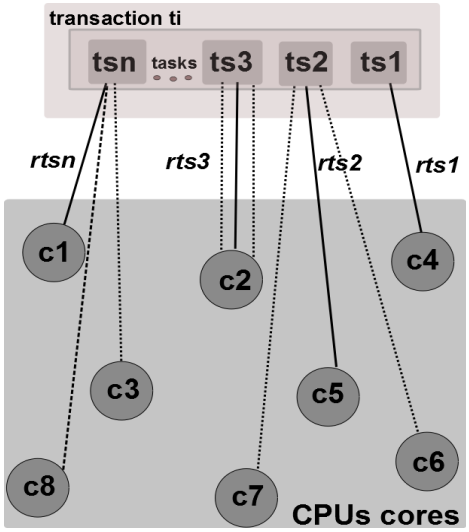


**Figure 2 Algorithm deployment**

The result $r_i$ is determined in accordance with the objectives *obj* of the transaction. The tasks associated with one transaction $t_i$ are run for a number of rounds *noOfRounds* until the result of the round satisfy the objectives. The tasks of a new generated round $r'_i = \{t_{s1}' + t_{s2}'+, ..., t_{sp}'\}$ are based on the results obtained from round $r_i$ where $t_{si'} = f[r(t_{si})]$, $t_{si'}$ is a function of result $r(t_{si})$. The termination criterion of each transaction resides on the following two conditions:

(i) Maximum number of rounds (*MaxNoOfRounds*) -- number of rounds after which a transaction finishes

(ii) Similarity of results ($r_i \sim r_{i+1}$) -- the level of similarity between results in consecutive rounds; if two consecutive rounds are very similar the transaction processing will stop.

In our system each $t_i$ has a set of associated parameters such as:

- Functional parameters: [taskId; objective]
- Non-functional parameters: [time; load; CPU usage]

Each transaction $t_i$ has a number of tasks and a corresponding computation time $p_{tj}$ calculated as $p_{tj} = noOfRounds * noOfTasks$, where *noOfRounds* determines the number of rounds of transaction $t_i$ and *noOfTasks* determines the number of tasks composing the transaction.

The number of tasks to be run within one transaction is dependent on the objective and the variables of the simulation model. At each stage the simulation model is updated by refining the input values of current simulation round with previous simulation round results until the optimisation objective is achieved (see Algorithm 1). Our framework is based on a cluster system composed by a number of servers $S = \{s_1, s_2, s_3,\ldots,s_m\}$ which can support a number of transactions. Each $s_i \in S$ is a cluster server containing 12 CPU cores such as $s_i = \{c_1, c_2, c_3,\ldots,c_p\}$, p=12. The cluster is monitored by a scalable distributed system called Ganglia [17] which

helps us in giving a graphic representation of the various metrics we report in Section 6.

Algorithm 1: Parallel computation algorithm
1: Initialise parameters of the algorithm;
2: **while** Transaction iteration is not completed do
3: Generate a new $t_i$;
4: **for all** $t_{si}$ := 1 to $t_{sn}$; **do**
5: Get variables of the $t_{si}$;
6: Create a working folder for the computing task $t_{si}$;
7: Clear the computing transaction working folders;
8: Create input data file for task $t_{si}$ working folder;
9: Copy necessary files into the task $t_{si}$ working folder;
10: **end for**
11: Submit transaction $t_i$
12: Waiting for the transaction $t_i$ to finish;
13: **for all** $t_{si}$ := 1 to $t_{sn}$ **do**
14: Parse task $t_{si}$ output files;
15: Set objective values and constraint values;
16: **end for**
17: **if** Meet transaction terminate condition **then**
18: Transaction iteration complete;
19: **end if**
20: **end while**

Within the system, servers are organised in two parallel based environments such as Torque and HTCondor, where two types of transactions can take place:

(i) Torque based transactions -- containing tasks to be completed within the Torque based system - identified as $t_j = \{t_{s1} + t_{s2}+, \ldots, t_{sn}\}$. Each $t_{si}$ is a task representing an EnergyPlus simulation process, whereas $t_j$ is a transaction representing an optimisation process composed by many $t_{si}$ tasks.

(ii) HTCondor based transactions -- containing tasks to be completed within the HTCondor based system - identified as $t_k = \{t_{s1} + t_{s2}+ \ldots + t_{sm}\}$.

## 6. Optimisation Use Case

We present a use case from the SportE$^{2}$[1] project pilot called FIDIA, a public sport building facility, located in Rome, Italy. The building we have used in the pilot study has wooden external walls of 9cm and a wooden external roof of 9cm. The floor is made of concrete. The windows are single glass with a thermal transmittance of 5:7W=m2K and a solar gain of 0.7. The geometry of the building is composed of an Gable roof with Hmin = 3m and Hmax = 6m with window surfaces of about 70 m2.
In this use case energy consumption *E* is considered as the main objective of the optimisation model composed by four different sub-objectives such as: the energy of heating $E_h$, the energy of cooling $E_c$, the energy consumed by fans $E_f$, and the energy consumption of generators $E_g$.

[1] SportE$^{2}$ is a research project co-financed by the European Commission FP7 programme under the domain of Information Communication Technologies and Energy Efficient Buildings.

$E = E_h + E_c + E_f + E_g$

The optimisation model is shown as following:

**Objective** : min Obj = $E_h + E_c + E_f + E_g$

**Design variables** : xi; i = 1- 6;

The optimisation model, composed by four objectives is based on Energy Plus simulations with a number of six design variables. These variables are updated after each round of simulation until the ending conditions of the optimisation process are triggered.

**Objective** :

minimize $E = E_h + E_c + E_f + E_g$

**Design variables :**

x1 ∈ [10, 1000]

x2 ∈ [10, 1000]

x3 ∈ {0:002, 0:003, 0:004, 0:005, 0:006, 0:008, 0:01, 0:012, 0:015, 0:018}

x4 ∈ {0:002, 0:003, 0:004, 0:005, 0:006, 0:008, 0:01, 0:012, 0:015, 0:018}

x5 ∈ [0:0017, 0:0036]

x6 ∈ [0:012, 0:16]

The sports facility is equipped with sensors and actuators for monitoring, control and optimisation of the facility. The building has metering capability to determine consumption of electricity, gas, biomass, water and thermal energy. This data can be accessed through a specialist interface and recorded for analysis

## 7. Experiments

We deploy a number of transactions with an associated objective *obj* on the infrastructure with dedicated cluster machines. In our system there are 8 dedicated cluster machines each having 12 CPU cores and 3201 MHz CPU speed.

We use Ganglia to measure various parameters such as: completion time, load, average utilisation, CPUs usage.

Experiment 1: Load and time based comparison - This experiment presents the load generated by the algorithm within the two infrastructures. From Figure 3 is observed how the load(axis o-y) and time(axis o-x) affected with the transaction evolves. Whereas for the HTCondor based execution the load is higher than for Torque, the execution time associated with the HTCondor execution is shorter that for Torque system. This is determined by the scheduling policies of the two infrastructures. While HTCondor distributes the task on all the CPU Cores, the Torque schedules tasks according to a time frame.

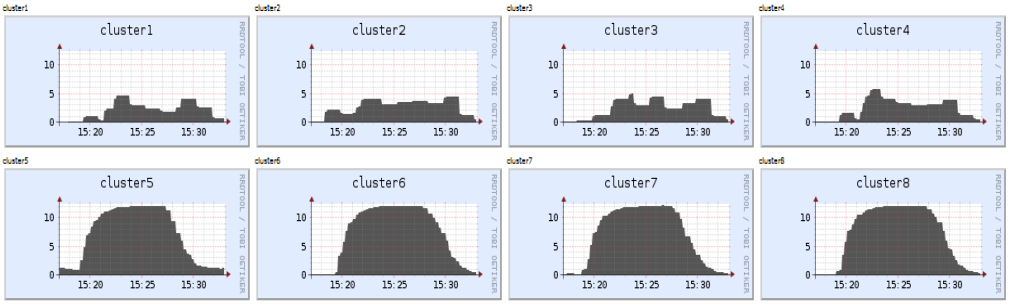


**Figure 3 Load and time for multiple transactions**

Experiment 2: Computation load and time - In this experiment we compute the algorithm on the HTCondor and Torque infrastructure and measure the time parameter.

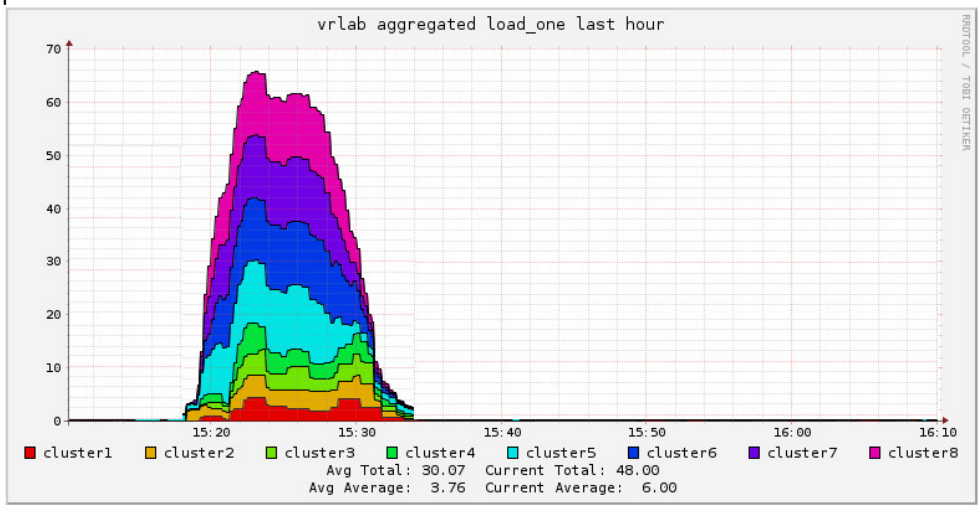


**Figure 4 Computation load and time in cluster**

This experiment shows the performances of the two infrastructures when deploying transactions with similar complexity degree(same number of objectives). This experiment provides a general overview on the load(axis o-y) and the computation time(axis o-x) associated with a transaction. Figure 4 demonstrates that on HTCondor the load is signiffcantly higher comparing to Torque. In what concerns the overall computation time is observed that the completion of a transaction takes longer on Torque.

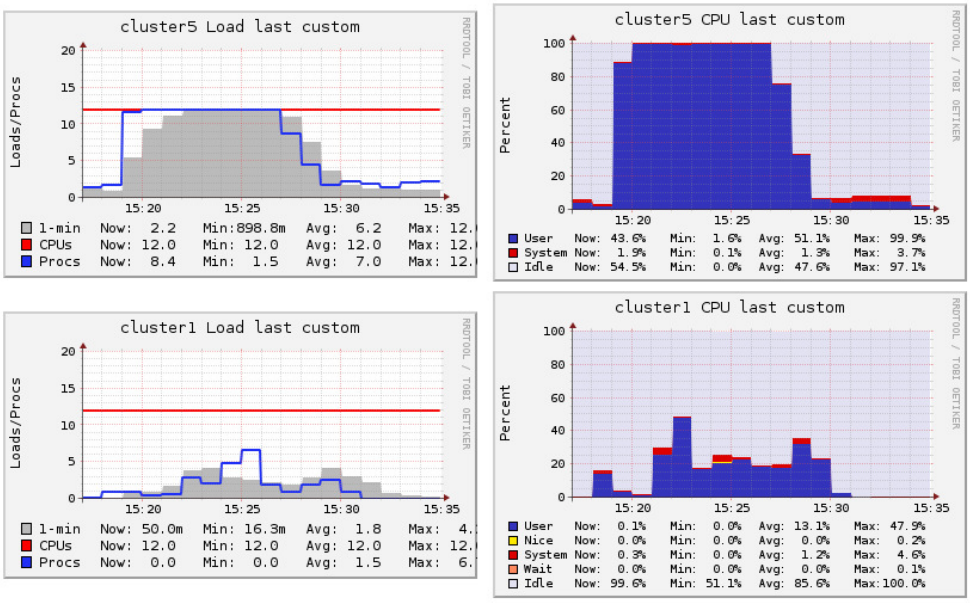


**Figure 5 Comparing hosts in load and CPU**

Experiment 3: Host based comparison - In this experiment we compare the load and CPU parameters of the two execution systems. By using a similar degree of complexity for both cases is observed that CPU load has different trajectories according to the execution environment used to deploy the transaction. Whereas the first two experiments provide an evaluation from the perspective of load(axis o-y) and time affected with a transaction(axis o-x), in this experiment we present a detailed comparison of the indivual host machines.
From the comparison illustrated by Figure 5 it can be concluded that HTCondor system can provide a higher effciency in what concerns the load and computation time. On the other hand, Torque system has the advantage of a better scheduling mechanism, signiffcantly reducing the computation load but with an additional delay concerning transaction completion.

Experiment 4: Comparison of single algorithm versus parallel algorithm - This experiment compares single CPU process with parallel process on HTCondor emphasising the capability of the parallel algorithm to use all the CPU cores comparing to the case when a transaction is deployed as a regular process with a single process execution. This experiment presents the load(axis o-y) and execution time(axis o-x) within two different experimental setups: (i) single CPU process and (ii) parallel algorithm.


**Figure 6 Comparing single vs parallel algorithm**

From Figure 6 is observed that the execution time of the optimisation process takes longer when deploying the algorithm on a single CPU process. The interval of computation time is signiffcantly reduced when deploying the algorithm to a parallel process. It can be observed that a single process takes around 4 days to complete while the parallel process reduces the execution time of the optimisation to couple of hours.

## 8. Conclusions

This paper presents a parallel framework for computing complex simulation based optimisation problems. As existing today, the computing capabilities can solve large-scale, complex engineering problems in a significantly shorter time compared with single-processor machines. As stated, energy simulation and optimisation identifies a class of applications demanding high performance processing power in order to be realised within a feasible time-frame.
We present our parallel algorithm by measuring the performances in a number of different scenarios. We test the performances of the algorithm both on a HTCondor system and Torque system and evaluate the impact of the load it produces and the computation time. Is observed that our algorithm performs better on a HTCondor

environment due to a key capability of using the available cores. When testing the algorithm on a Torque environment is observed that the load is lower but computation time increases comparing to HTCondor. We demonstrate that our algorithm can represent a realisable solution not only for complex energy optimisation problems but can be easily extended to other types of problems. When evaluating the single process and parallel process case of the algorithm is observed that our parallel algorithm reduces the computation time significantly. Althrough in the evaluated case we only use continuous variables and discrete optimisation, our system can cope with diffferent types of variables and optimisations. The advantages of our solution rely not only on the performances of the algorithm but also on the unique combination of different submission systems such as HTCondor and Torque. By combining the two environments within a single solution we facilitate an optimum between the high execution time and robust scheduling.

Acknowledgements: The research presented in this work is supported by EU FP7 SportE2 project, ICT for Energy Efficiency in European Sport Facilities.

## References

[1] Rezvan, A.T., Gharneh, N.S., and Gharehpetian, G., Optimization of distributed generation capacities in buildings under uncertainty in load demand. Energy and Buildings, 57 (0), pp. 58 -64. 2013
[2] Magnier, L. and Haghighat, F., Multiobjective optimization of building design using TRNSYS simulations, genetic algorithm, and Artificial Neural Network. Building and Environment, 45 (3), 739-746, 2010.
[3] Hong, T., Chou, S., and Bong, T., Building simulation: an overview of developments and information sources. Building and Environment, 35 (4), 347 -361, 2000.
[4] Raicu, I.; Foster, I.T.; Yong Zhao, "Many-task computing for grids and supercomputers," Many-Task Computing on Grids and Supercomputers, 2008. MTAGS, pp.1-17 Nov. 2008
[5] Jim Basney and Miron Livny, "Managing Network Resources in Condor", Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), Pittsburgh, Pennsylvania, August 2000, pp 298-299.
[6] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley, 2003. ISBN: 0-470-85319-0
[7] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience" Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.
[8] TORQUE Resource Manager, http://www.clusterresources.com/products/torque-resource-manager.php, Last accessed: June 2013.
[9] High Throughput Computing, http://research.cs.wisc.edu/htcondor/, Last accessed: June 2013.
[10] A. Taghipour Rezvana, N. Shams Gharneha, G.B. Gharehpetianb. Optimization of distributed generation capacities in buildings under uncertainty in load demand. Energy and Buildings 57 (2013) pp.58-64.
[11] Laurent Magnier, Fariborz Haghighat. Multiobjective optimization of building design using TRNSYS simulations, gentic algorithm, and Artical Neural Network. Building and Environment 45 739 - 746 (2010)
[12] P. Tvrdik. Parallel systems and algorithms. Czech Technical Publishing House, 167,1994.
[13] E. Alba, J. M. Troya. Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Generation Computer Systems, 17(4):451-465, 2001.
[14] M. Sefrioui, J. Periaux. A Hierarchical Genetic Algorithm Using Multiple Models for Optimization. Parallel Problem Solving from Nature VI., Paris, France, 879-888, 2000.
[15] M. Jelasity, M. Preub, A. Eiben. Operator Learning for a Problem Class in a

[16] Distributed Peer-to-Peer Environment. Parallel Problem Solving from Nature VII, Granada, Spain, 172-183, 2002.
[17] M. Arenas, P. Collet, A. Eiben, M. Jelasity, J. Merelo, B. Paechter, M. Preub, M. Schoenauer. A Framework for Distributed Evolutionary Algorithms. Parallel Problem Solving from Nature VII, Granada, Spain, 665-675, 2002.
[18] C. Pereira, C. Lapa. Coarse-grained parallel genetic algorithm applied to a nuclear reactor core design optimization problem. Annals of Nuclear Energy, 30(5):555-565, 2003.
[19] T. Sterling, J. Salmon, J. D. Becker, F. D. Savarese. How to Build a Beowulf. A Guide to the Implemnetation and Application of PC Clusters. The MIT Press, 239, 1999.
[20] G. A. Sena, D. Mergherbi, G. Isern. Implementation of a parallel genetic algorithm on a cluster of workstations: Traveling salesman problem, a case study. Future Generation Computer Systems, 17:477-488, 2001.
[21] Ganglia Monitoring System: http://ganglia.sourceforge.net/.
[22] Fumo, N.; Mago, P.; Luck, R. "Methodology to Estimate Building Energy Consumption Using EnergyPlus Benchmark Models." Energy and Buildings; (42:12); pp. 2331-2337, 2010.
[23] Garg, V.; Chandrasen, K.; Tetali, S.; Mathur, J. "Energyplus Simulation Speedup Using Data Parallelization Concept." ASME Energy Sustainability Conference, New York: American
[24] Energy Plus, Available at: http://apps1.eere.energy.gov/buildings/energyplus/