STM JOURNALS

Journal of Advances in Shell Programming

Volume 1, Issue 2
www.stmjournals.com

Automation in Computation Over Linux Integrated Environment

Ankush Rai*, Siddarth Choubay
Department of Applied Science, CRIAD Laboratories, Smriti Nagar, Bhilai, Durg, Chhattisgarh, India
Shri Shankaryacharya College of Engineering, Chhattisgarh, India

Abstract

The challenges of building an automated computational system for an effective controller of job scheduling and execution of computational tasks in heavy computing research projects, either requires external controllers or must function within the restriction bound of the utility service offered by APIs. We address the problem of the automation in computation over a Unix environment using effective shell programming techniques. The approach is mainly based on dividing computational jobs by explicitly reducing the problem to the satisfiablity through strings of feedback control policies for a variety of services.

Keywords: *Automation, Computation, Linux Shell, Dynamic Programming.*

**Author for Correspondence* E-mail: ankushressci@gmail.com

INTRODUCTION

Research on dissimilar regularities, in particular, different variants of the computing tasks like that of GENOME sequencing can be retrieved from various important techniques [1, 2]. It still requires a hefty chunk of manual interference for the selection, extraction and matching up of the desired result during iterative run of the simulation. We address this issue in this study and proposed an algorithm to automate the process of computation in dynamic run flow such that it eliminates the requirement of the human interference, which is more or less prone to error and subjected to the risk of data disturbances; essentially it is more important and will be helpful for the scientific community associated with computational fields.

However, in the past few methods have been attempted in this scenario to achieve automation but the full automation hasn't yet been achieved. One of them is to employ a Tomcat server cluster that serves dynamic content to clients which didn't require purchasing one's own infrastructure to run its service. Thus, the authority provided a slice of resources on lease from a cloud hosting provider to reduce capital and operating costs. The advantage it offers is that its application is horizontally scalable. Mainly, the cloud API also offers support for zones to guide the placement of VM instances in the network, connecting several research groups and various computational jobs is achieved over it simultaneously but it is more prone to insignificant disadvantages as the VM installation it is mounted is influenced badly by insignificant out communications for its services. In this scenario, the cloud hosting provider runs its own control system to arbitrate resource requests but application control is factored out of the cloud platform and left on the client. This points towards the notion of clean decoupling of application control policy from the cloud platform mechanism, which is a necessary architectural choice to prevent cloud platforms from becoming brittle as guest demands change [3]. Its control policies must be stable and effective under the wide range of conditions that is enormously encountered in practice. Adaptive resource provisioning from Cloud services is just one example of the need for feedback driven application control [4–6].

METHODOLOGY

Following Points are to be considered before designing an automated controller for any computational jobs:-

a. How much internal knowledge does the controller need?
b. How do we integrate the internal guest constraints to the guest control?
c. How does the control deals with multi-tier interactions?

Let us suppose that C, B, X, and Y be matrix of some strings, X and Y be the matrix of strings for stimulant variable and response variable for each X, such that C = XBY. Then C is a superstring of B. The length of a string S is equivalent to the total number of letters in it and is denoted by |S|.
Now,

$$X_{x_0} = \begin{bmatrix} {X_{x_0}}^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {X_{x_0}}^{(r)} \end{bmatrix}, \quad {X_{x_0}}^{(j)} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \end{bmatrix}, j = 1,2, \ldots r$$

where, X_{x_0} is the covariance map which asserts to the association formed between the strings $S(t)$with that of stimulant and response variable. X_i is the position of input record. C_l is the cluster value which contains various values from 1 to l.

Now, at each step we calculate:

$$S(t) = \sum_{i=-\infty}^{+\infty} \sum_{k=1}^{N_{SC}} C_{li} S_l(t - iT_s)$$

$$S_l(t) = \prod(t) e^{j2\pi f_l t}$$

$$\prod(t) = \begin{cases} 1, 0 < t \leq T_s \\ 0, t \leq 0, t > T_s \end{cases}$$

where, C_{li} is the i^{th} information symbol at the l^{th} subcarrier (when output of one iteration is propagated to the input of the other), T_s is the symbol period, S_l is the waveform for the l^{th} subcarrier, N_{sc} is the number of subcarriers (number of matching iterations), f_l is the frequency of the subcarrier, and π(t) is the pulse shaping function. Follow this process to complete the dataset in all records. Thus, the dynamics of the equation for a computational job of quantum chemistry is automated and shown in Figure 1 below; where the intersection points reveals the desirable convergence output for the after the iterative run of the source code [7–10].

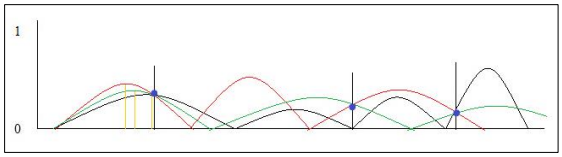

Fig. 1: *Dynamics of the Automated Script for the Quantum Chemistry Computational Task During the 1500 Iterative Run x-axis; The Color Waveform Represents Different Strings during the Script's Dynamic Flow.*

CONCLUSION

We have presented issues that eliminate the use of feedback control and its consequently hefty computing for greater workload and CPU utilization through the proposed scripting method. As shown in Figure 2 it is inferred that the effective infrastructure utilization is accomplished while still being different from feedback control of other computer systems. It provides coupled control and granularity. It is highly adaptable to its work environment whether it is for parallel processing or for cloud services.

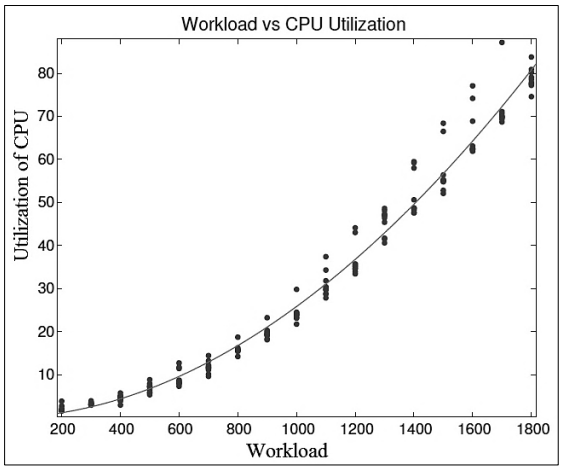

Fig. 2: *CPU Utilization v/s Workload Distribution by the Proposed Automated Script shows the Scalable Even Consumption during the Computational Operation.*

REFERENCES

1. Popov V. Yu., Computational Complexity of Problems Related to DNA Sequencing by Hybridization, *Doklady Math.* 2005; 72: 642–644p.
2. Popov V., Sorting by Prefix Reversals, IAENG, *Int. J. Appl. Math.* 2010; 40: 247–250p.

3. Gorbenko A., Popov V., On the Problem of Placement of Visual Landmarks, *Appl. Math. Sci*. 2012; 6: 689–696p.
4. Gorbenko A., Popov V., Computational Experiments for the Problem of Selection of a Minimal Set of Visual Landmarks, *Appl. Math. Sci*. 2012; 6: 5775–5780p.
5. Gorbenko A., Popov V., Task-resource Scheduling Problem, *Int. J. Auto. Comp*. 2012; 9: 429–441p.
6. Gorbenko A., Popov V., SAT Solvers for the Problem of Sensor Placement, *Adv. Stud. Theor. Phy*. 2012; 6: 1235–1238p.
7. Popov V., the Approximate Period Problem for DNA Alphabet, Theoretical Computer *Science* 2003; 304: 443–447p.
8. Popov V., The Approximate Period Problem, IAENG *Int. J. Comp. Sci*. 2009; 36: 268–274p.
9. Popov V., Approximate Periods of Strings for Absolute Distances, *Appl. Math. Sci*. 2012; 6: 6713–6717p.
10. Popov V., Multiple Genome Rearrangement by Swaps and by Element Duplications, *Theor. Comp. Sci*., 2007; 385: 115–126p.