ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/261336440

Distributed T-way test suite data generation using exhaustive search method with map and reduce framework

Conference Paper · October 2010

DOI: 10.1109/ISIEA.2010.5679443

CITATION
1

READS
44

4 authors:

Zainal Hisham Che Soh
Universiti Teknologi MARA (Pulau Pinang) Permatang Pauh, Malaysia
22 PUBLICATIONS **22** CITATIONS
SEE PROFILE

Syahrul Afzal Che Abdullah
Universiti Teknologi MARA
19 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Kamal Z Zamli
Universiti Malaysia Pahang
167 PUBLICATIONS **1,250** CITATIONS
SEE PROFILE

Mohammed I. Younis
University of Baghdad
48 PUBLICATIONS **351** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Bat-inspired t-way Strategy (BTS) View project

Computational Search Techniques for Solving Optimization Problems View project

All content following this page was uploaded by Kamal Z Zamli on 16 July 2014.

The user has requested enhancement of the downloaded file.

Distributed T-Way Test Suite Data Generation using Exhaustive Search Method with Map and Reduce Framework

Zainal Hisham Che Soh
Faculty of Electrical Engineering
Universiti Teknologi MARA
Penang, Malaysia

Syahrul Afzal Che Abdullah
Faculty of Electrical Engineering
Universiti Teknologi MARA
Shah Alam, Malaysia

Kamal Zuhairi Zamli and Mohammed I. Younis
School of Electrical and Electronics
Universiti Sains Malaysia
Penang, Malaysia

Abstract--- ***In this paper, a new distributed algorithm for generating test suite for t-way testing utilizing a shared tuple space server as transport layer between master and worker is presented. The design of the algorithm is based on the Map and Reduce software framework implemented using tuple space technology. The proposed algorithm requires listing all the possible interaction elements into all partitioned space to ensure maximum interaction coverage in all partition. The exhaustive test cases is generated and loaded into their respective partition using hash based routing algorithm. The final test suite is produced from a reduced the test case obtained from the worker calculation on maximum interaction coverage. An encouraging simulation result is obtained on effectiveness of using the tuple space technology to parallelize the test suite generation on single machine and multi machine for different input parameter.***

Keywords---***Combinatorial Interaction Testing, Worker-Master Design Pattern, Map and Reduce, Tuple Space Technology***

I. INTRODUCTION

In highly customizable and configurable software system, such as web application, the number of test cases can be enormous in testing all the possible interaction among the different input parameter in the whole system involving the web browser client, web server and business logic server. The unintended interaction among input parameter can remain undetected and can cause the whole software system failure. Therefore, an efficient approach is needed to minimise the number of test suite size.

Combinatorial testing is an effective technique to reveal faulty interaction inside a given software system by covering all possible interaction element combination of input parameters by at least one of generated test cases, thus ensuring detection of errors due to their faulty combination. The pairwise or 2-way testing is one form of *t*-way combinatorial testing that is used to generate all possible pairwise interaction among two different input parameters.

For real world and complex software system, the possible size and combination of input parameters value are likely to be huge and can lead toward the combinatorial explosion problem. Hence, the generation of test suite will require a lot of computing power and memory resources. Therefore, limited memory resources can cause the generation to be halted. In order to overcome the problem, we propose to utilize the parallel processing to generate the test suite. The main goal of utilizing parallel processing is to increase the computing power and memory resources to overcome the combinatorial explosion problem for large input parameters and parameter values.

In this paper, we present our approach in to combinatorial testing in designing a distributed algorithm to generate test suites for t-way testing for software system under test. The algorithm described in this paper is based on Map and Reduce framework on distributed tuple space environment. The remainder of the paper is organized as follows. Section 2 gives some insight on the topic and recently published related works. Section 3 describes our distributed shared memory for generation of test suite approach called Tuple Space Exhaustive Search Method (TS-ESM) technique. Section 4 discussed the experimental result of the distributing the computing work on single and different machine. Finally, section 5 draws our conclusions and point out the ideas for future extension of this work.

II. RELATED WORKS

In general, there have been two strategies for generation of test suites either computational or algebraic approach. In computational approach such as, Automatic Efficient Test Generator (AETG) builds a test suite using greedy algorithm by iteratively adding one test case at a time until all the interaction element combinations are covered [3]. In [2], the IPO strategy has been introduced, which builds an exhaustive test for the first two parameters, extends the test suite to cover the first three parameters, and continues to extend the test suite until it builds a test suite for all the parameters.

Heuristic search techniques such as SA, ACA, GA usually start from a pre-existing test suite and then apply a series of transformations determine using a fitness function to the test suite until a complete test suites is reached that covers all the combinations.

Other than computational approaches, algebraic approaches have also been constructed. These approaches construct test suite using pre-defined rules using mathematical function such as Latin square, orthogonal

array and graph theory to produce a covering array. Other algebraic approaches are based on the idea of recursive construction, which allows larger test sets to be constructed from smaller ones.

Both computational and algebraic approaches have their own advantages and disadvantages such as computational approaches can be applied to any system configurations, but the computation can be intensive. Algebraic approaches on the other hand usually involved lightweight computations and in some cases, algebraic approaches can produce optimal test sets. However, algebraic approaches often impose restrictions on the system configurations to which they can be applied.

Nowadays, mostly available works on Combinatorial Interaction Testing exploits sequential algorithm in solving the problem such as AETG, IPO, IRPS[1], DDA[4], IPOG[5] and G2Way[6]. However, there only a few works that utilize the parallel algorithm to solve this problem such as GMIPOG[7] and MC-MIPOG[8]. GMIPOG distributed the test sets generation process into the grid by partitioning the work based on parameter value. It's generates more optimum test set and hence lesser combinations. It also supports higher order of interaction strength up to 13 ways.

Parallel Automation Testing can be used to reduce time required by both, concurrently generating test suites and concurrently performing the test execution. The execution of test suites is automatically done after their generation by delegating the generated test suites. The technique presented below is the first milestone to explore the work into Parallel Automation Testing.

III. THE DESIGN APPROACH FOR TS-ESM

We now describe about our work here on algorithm for tuple space based combinatorial interaction testing called Tuple Space Exhaustive Search Method (TS-ESM) technique. The design pattern used is Map and Reduce paradigm. Map and Reduce is a software framework that introduced by Google to search a large data set from a cluster of workstation. In our design approach, the InputDataLoader preloads the interaction strength, *t* value and the input parameter data, *ParmSet* into all space partition in POJO format. The TestDataFeeder; the master process obtains the parameter set, *ParmSet*; the strength of interaction coverage, *t* and partition number, *p* from space. The number of partition determines the number of worker processes. The master remotely executed all worker process to generate all possible interaction elements and stores it back into their respective partition space.

Nearly on the same instance, the master generates all the exhaustive test case and delegate the test case into workers partition based on hash based routing algorithm. The master then remotely executes a worker process to calculate the maximum interaction coverage among the test cases data stored in their partition. A result containing the maximum interaction coverage and the selected test case in their respective partition is sends back to calling master via a reducer that select a test case with highest interaction coverage value among results returned by all available worker process. The master store the selected result containing one test case, τ and its maximum interaction coverage value, *maxIEC* into the final test suite, *TS* and delete all interaction element covered by selected test case in all partition space in order to ensure optimal solution. The master then continues to remotely execute all workers to obtain τ and its *maxIEC* at their respective partition until all interaction elements are covered. The generation of test suite will stop when the shared interaction element data is empty.

A. Master/Client Process

In this section, we describe how the master application works:

a) The master sends *ParmSet*, and interaction strength *t*, to all partition space.

b) The master generates the exhaustive test cases set and delegate a different set to respective worker partition space based on hash based data routing mechanism.

*c)*The master starts remotely executes all the worker processor at their respective partition using map/reduce implementation and waits for one test case τ and its maximum interaction element coverage value *maxIEC*.

d) The master stores the selected test case τ corresponding to the highest *maxIEC* value to be added to final test suite, *TS*.

*e)*The master deletes the selected test case and all interaction elements covered by in *IESet*.

f) The master checks if the *IESet* is empty or not. If empty, the master will consider the latest test suite, *TS* as the final generated test suite. Otherwise, the master re-executes all the worker process again at their respective partition space.

For clarity, the complete algorithm for master is given below:

Algorithm *TS-ESM Master (ParameterSet ParmSet, t)*
begin
1. initialize test suite *TS* to be an empty set;
2. denote the parameters in *ParmSet,* in an arbitrary order, as *P1*, *P2*, …, and *Pn;*
3. send *ParmSet* and *t* to all partition space;
4. execute a worker processor to load all possible interaction element to all partition space;
5. sends the generated test set to respective partition space;
6. **while** *(IESet is* not empty*)* **do**

begin

7. execute a worker processor at their respective partition space to find to test case, τ with highest interaction coverage;
8. wait for result of τ with *maxIEC* from a reducer*;*
9. add *t* value to *TS;*
10. delete the selected τ *of* highest *maxIEC;*
11. delete covered interaction element in all partition of their respective *IESet;*

end

12. display *TS;*

end

B. Worker Process

In this section, we now describe how the worker process works:

a) Each worker process reads the *ParmSet* and *t* in their respective partition space.

b) Each worker generates all possible interaction elements and sends back to their space as *IESe*t.

*c)*Each worker reads all test case in their partition space and the worker determines the *maxIEC* of among available test case and returns the highest *maxIEC* with selected test case, τ to a reducer.

The complete algorithm for the worker is as in right hand.

Algorithm *TS-ESM Worker (ParameterSet ParmSet,t)*
begin
1. read *ParmSet* and *t* from their partition space;
2. generate all possible interaction element and send into their space as *IESet*;
3. **for** *(1..ts)* **do**

begin
4. check previous *pmaxIEC* value;
5. calculate τ with highest *maxIEC;*
6. if(maxIEC equal pmaxIEC)
7. send τ with highest *maxIEC* to shared data space;

end
end

TABLE I

RESULTS FOR FIXED INTERACTION STRENGTH, T=3 AND 6 PARAMETERS WITH 2 TO 5 VALUES ON FOUR DIFFERENT MACHINES.

Machine	One		Two		Three		Four	
Parameter Value	Size	Generation time(s)	Size	Generation time(s)	Size	Generation time(s)	Size	Generation time(s)
2	12	0.54	14	1.06	15	1.063	15	1.39
3	45	25.12	42	13.77	44	11.062	45	10.97
4	103	541.78	107	261.50	103	199.06	104	168.61

TABLE II

RESULTS FOR FIXED PARAMETER VALUE, V=4 AND 6 PARAMETERS WITH INTERACTION STRENGTH OF 2 TO 4 ON FOUR DIFFERENT MACHINES

Machine	One		Two		Three		Four	
Interaction strength, t	Size	Generation time(s)	Size	Generation time(s)	Size	Generation time(s)	Size	Generation time(s)
2	23	29.35	22	12.53	24	8.47	28	10.14
3	103	541.79	107	261.50	103	199.07	104	168.61
4	414	1763.26	414	1007.11	411	922.00	412	877.33

IV. RESULTS

In this section, the results of the experiment carried out are presented and analyzed in relation to speedup gain from distributing the workload among the different machine. Two experiments were carried out to determine the speedup gain from different input parameter value and different interaction strength running on single or multi machine settings. Speedup is deduced from execution time of single machine per execution time of multi machines.

A. Increasing parameter value on up to 4 machines

This experiment is carried out to investigate the speedup gain as the number of parameter values increases from 2 to 4 with fixed parameter of 6 and fixed interaction strength of 3. Firstly, we apply the TS-ESM strategy on single machine using GigaSpaces service grid and then on multi workstation up to 4 machines. The result of the number of generated test size and test suite generation time are shown on the Table I. The speedup value is determined by taking the time ratio between single machine generation time over the multi machine generation time as illustrated in Figure 1. It demonstrated that the speedup is more significance for higher parameter value compares to low parameter value.

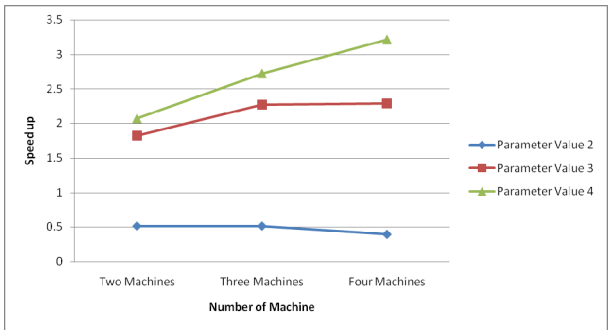

Figure 1. The speedup vs number of machine with varying parameter value from 2 to 4.

It is shown in the value of speedup for parameter value of 4 is higher than parameter value of 3 and 2 for all number of machines from two to four machines. As for parameter value of 2 no speedup is gained due to time to setup the distribution framework of both data distribution and task distribution is bigger compare to the test generation time.

B. Increasing interaction strength on up to 4 machines

In this computational simulation, we determined the speedup as the number of interaction strength is increases with fixed parameter and their parameter value. The TS-ESM strategy is run on single machine and then distributed using Gigaspaces service grid on multi workstation up to 4 machines. The input parameter consist of fixed 6 parameter and fixed parameter value, v=4 with varying interaction strength from t= 2 to 5. The result is shown on the Table II. In general as shown Figure 2, speed up is gained when running on multi workstation environment for all interaction strength value from 2 to 4. Although the value of speedup is higher for small interaction strength of 2 as compare to large interaction strength 3 and 4, it speedup is decreasing while running on four machine. In contrast, the speedup is always increasing and more attainable for higher interaction strength of 3 and 4 when running on high number of machine. The small value of speedup for higher interaction strength is due to computational complexity caused by combinatorial explosion problem thus slow down the test suite generation time.

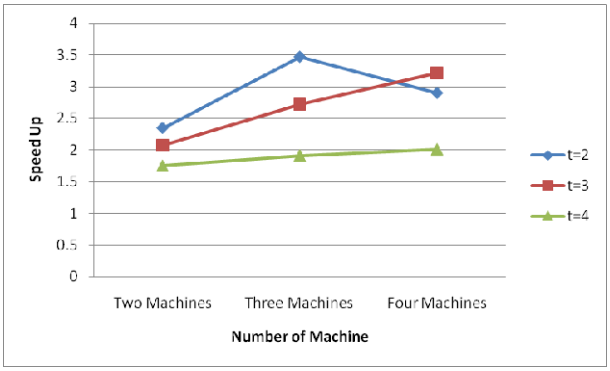

Figure 2. The speedup vs number of machine with varying interaction strength value from 2 to 4.

V. CONCLUSION

In this paper, we investigated a distributed algorithm for t-way combinatorial interaction testing by delegating the test cases among worker process to obtain the faster determination of maximum interaction coverage of the test case. Using Map and Reduce framework provided by GigaSpaces middleware, we were able to speedup the generation time while running the application on multi machine compare to single machine. As parameter value increase, the speed up gained as increased until it reaches it scaling linearity limit. Medium size of input parameter is suitable for this exhaustive search method and not for large input parameter. In addition, the test suite generation time is still too slow due to exhaustive search. Although the size of test suite is quite optimum and small, it is not deterministic in nature. In between each generation there are differences in the number of test sizes in range of 0 to 4. This happened due to the randomization nature of test case selection among test cases with same interaction coverage value. Furthers work will be carried out to redesign the algorithm so that it suitable for high input parameter. At the moment the developed algorithm is still not suitable for high input parameter due to high usage of memory during runtime. Therefore for high input parameter, the generation is halted due to out of memory.

ACKNOWLEDGMENT

This research is partially funded by the generous grants – "Investigating Heuristic Algorithm to Address Combinatorial Explosion Problem" from Ministry of Higher Education (MOHE), Malaysia and USM Research University Postgraduate Research Grant Scheme on "Distributed Strategy for Software and Hardware Test Planning Tool using Tuple Space Technology" from IPS, USM.

REFERENCES

[1] M. I. Younis, K. Z. Zamli, and N. A. M. ISA, "IRPS - An Efficient Test Data Generation Strategy for Pairwise Testing ", in Proc. of the 12th Intl. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems (KES2008), Zagreb, LNAI Vol. 5177, pp. 493-500.

[2] Y. Lei and K. C. Tai , "In-parameter-order: a test generation strategy for pairwise testing," *Proceedings of 3rd IEEE Intl. Conf. on High-Assurance Systems Engineering Symposium*, 1998, pp. 254-261.

[3] D. M. Cohen, S. R. Dalal, J. Parelius, G. C. Patton, "The Combinatorial Design Approach to Automatic Test Generation," *IEEE Software,* Vol. 13, No. 5, pp. 83-87, September 1996.

[4] R.C. Bryce and C.J. Colbourn, "The Density Algorithm for Pairwise Interaction Testing," *Journal of Software Testing, Verification, and Reliability*, 17(3): 159-182, 2007.

[5] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: A General Strategy for T Way Software Testing", in Proc. of the 14th Annual IEEE Intl. Conf. and Workshops on the Engineering of Computer-Based Systems, Tucson, AZ, March 2007, pp. 549-556.

[6] M.F.J. Klaib, K. Z. Zamli, N. A. Mat .Isa, M. I..Younis, and R. Abdullah, "G2Way – A Backtracking Strategy for Pairwise Test Data Generation", in the 15th IEEE Asia-Pacific Software Engineering Conference, Beijing, China, 2008, pp. 463-470.

[7] M. I. Younis, K. Z. Zamli, and N. A. M. Isa, "A strategy for grid based T-Way test data generation," in Proceedings the 1st IEEE International Conference on Distributed Frameworks and

Application (DFmA '08), pp. 73–78, Penang, Malaysia, October 2008.

[8] Mohammed I. Younis, and Kamal Z. Zamli, "MC-MIPOG: A Parallel t-Way Test Generation Strategy for Multicore Systems," ETRI Journal, vol.32, no.1, Feb. 2010, pp.73-83.

View publication stats