Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 1
February 2009

Complexity Analysis in Heterogeneous System

Kuldeep Sharma & Deepak Garg
Department of Computer Science and Engineering
Thapar University
Patiala-147004, India

Abstract

Complexity analysis in heterogeneous system is one of the more complicated topics in the subject mathematics for computing. It involves an unusual concept and some tricky algebra. This paper shows how various algorithms take part in heterogeneous system and how they affect the overall system complexity? All analysis and methods are situation specific. Here we show a comparative study of various method and techniques of algorithm analysis giving their specific advantages and disadvantages. We will specify general guidelines for calculating the complexity of heterogeneous system which still gives sleepless nights to the researchers.

Keywords: Heterogeneous system, Complexity analysis, Time complexity, Space complexity

1. Introduction

In recent years heterogeneous system has begun to play a vital role in computing. The potential use of heterogeneous system in safety-critical applications increases the importance of their reliability. The system includes telecommunications, consumer products, robotics, and an automotive control system is called heterogeneous system. Heterogeneous system contains many different kinds of hardware and software working together in cooperative fashion to solve problems. There may be many different representations of data in the system. This might include different representations for integers, byte streams, floating point numbers, and character sets. Most of the data can be marshaled from one system to another without losing significance. Attempts to provide a universal canonical form of information is lagging. There may be many different instructions sets. An application compiled for one instruction set cannot be easily run on a computer with another instruction set unless an instruction set interpreter is provided. Heterogeneous system may be based on software as well as hardware, e.g. we have a system

in which our data is on site A and on site B, Site A having the SQL Server and Site B having DB2. In this scenario we may have same hardware but platforms are different. In another case we may have SQL server or DB2 on both site but hardware may be different.

An *algorithm* is a well-defined sequence of steps to solve a problem of interest. The complexity of an algorithm means a function representing the number of steps (times) required to solve a problem under the worst-case behavior. The worst-case behavior of an algorithm, means the maximum number of steps executed (or time taken) to solve a problem. Hence, for any algorithm, even before implementing it, its complexity function should be analyzed. Such analysis will help us predict the maximum magnitude of time required to solve a problem using the algorithm. There is one more type of complexity function, namely *volume complexity function*, which represents the maximum primary memory requirement while executing the algorithm (Design And Analysis of Algorithms).

Time complexity function of an algorithm gives the worst-case estimate in terms of the number of steps to be executed for the algorithm. The order of the algorithm is defined like (n^2), $O(n!)$, $O(2^n)$, etc. The big *O* means the maximum order of the algorithm. $O(n^2)$ means that order of the algorithm is n^2, which indicates the maximum number of steps required to solve any problem by that algorithm is n^2 where n is the problem size. The time complexity function may be either polynomial or exponential.

Volume complexity: The Volume complexity function of an algorithm represents the amount of prime memory space required while executing the algorithm. Again this may be either polynomial or exponential. In the case of the branch and bound technique, if Breath First Search (BFS) is used, the function representing the memory space required to store the sub-problems will be in exponential from; if Depth First Search (DFS) is used, the function representing the memory space required to store the sub-problems will be in polynomial form. The type of the algorithm as well as data structure affects the volume complexity. This analysis will be helpful in deciding the types of computer to be used for implementing the algorithm.

2. Methods for Analysis

Apriori means designing then making. The principle of apriori Analysis was introduced by *Donald Knuth*. In apriori Analysis first we analyze the system or problem then we design or write the code for the problem.

Posterior Analysis refers to the technique of coding a given solution and then measuring its efficiency. Posterior Analysis provides the actual time taken by the program. This is useful in practice. The drawback of posterior analysis is that it depends upon the programming language, the processor and quite a lot of other external parameters.

Micro analysis refers to perform the instruction count for all operations. It counts each and every operation of the program. Micro analysis takes more time because it is complex and tedious (Average lines of codes are in the range of 3 to 5 million lines).Those operations which are not dependent upon the size or number of the input will take constant time and will not participate in the growth of the time or space function, So they need not be part of our analysis.

Macro Analysis Perform the instruction count only for dominant operations or selective instructions which are costliest. The following are a few examples of dominant (or basic) operations e.g. Comparisons and Swapping are basic operations in sorting algorithms. B**est**, **worst** and **average cases** of a given algorithm express what the resource usage is *at least*, *at most* and *on average*, respectively.

Amortized analysis refers to finding the average running time per operation over a worst-case sequence of operations. In this kind of analysis we do not include probability we always use actual data so it guarantees the time per operation over worst-case performance, hence it is different from average case.

E.g. In the implementation of the dynamic array, we double the size of the array each time it fills up. Because of this, array reallocation may be required, and in the worst case an insertion may require $O(n)$. However, a sequence of n insertions can always be done in $O(n)$ time, so the *amortized* time per operation is $O(n) / n = O(1)$.(Allan Bording and Ran El-Yaniv. 1998.) *Aggregate analysis* determines the upper bound $T(n)$ on the total cost of a sequence of n operations, and then calculates the average cost to be $T(n) / n$.

Accounting method determines the individual cost of each operation, combining its immediate execution time and its influence on the running time of future

operations. Usually, many short-running operations accumulate a "debt" of unfavorable state in small increments, while rare long-running operations decrease it drastically (Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein.).

Potential method is like the accounting method, but overcharges operations early to compensate for undercharges later.

3. Complexity of various algorithms in the system

The example given in figure 1 there are number of algorithm like authentication, encryption, fault tolerant, routing algorithms, ACID (atomicity, consistency, isolation, durability) properties for transaction and, communication protocol etc. In figure 1.terminals at a branch connected to a set of clustered web servers for authentication. The web servers are in turn connected to a set of application servers for implementing banking rules and policies. The application server's access mirrored and/or replicated data storage. Redundancy is present in the network also. Telephone banking using an Interactive Voice Response System is used as a backup if the branch terminals break down. The design of the authentication hardware and software requires fault tolerance – the users should not have to Re-login if one or more servers fail – some state should be stored in the form of cookies in non-volatile storage somewhere. The banking calculations require very high accuracy (30+ digit accuracy). Various kinds of fault tolerance schemes are used for storage. For example, two mirrored disks always keep identical data. A write to one disk is not considered complete till the other is written also. The servers have to respond within seconds to each user level request (deposit, withdrawal, etc) – the real time response of the system has to be evaluated using queuing theory and similar techniques. For TTS, the response output speech samples have to be guaranteed to be delivered at periodic time intervals, say every 125*ms* (micro-seconds). We have to calculate the complexity of overall system.

In a communication network routing is one of the latency sources. Routing algorithm has task to determine the suitable port for delivering message address to one network node. Correct delivery of message from source node to destination node is considered as optimal routing. The aim is to have the fastest and optimal routing available in the network.

The principle of routing with routing tables is straightforward. Ever node keeps a table with entries for each node. Using these entries, it could be determined via which outgoing port a message had to be sent. Interval routing is a space efficient routing strategy used in computer networks. The basis of interval routing idea is to label all of the nodes with integer from one set (for example $\{1, 2, 3...n\}$) and labeling of all arcs with interval (in range of number of nodes). These intervals means that a message addressed to a node labeled with u, via a port labeled with interval which includes a node u is forwarded.

The mechanism used for detecting and correcting faults is called fault tolerance. It can be further classified based on place of deployment, fault coverage, fault or error latency, cost of recovery .In real time systems fault tolerance can be achieved by online fault (K. Chandy, J. Browne, C. Dissly and W. Uhrig. 1975.) detection followed by a hardware-based checkpoint and rollback recovery mechanism. Cost is measured in terms of hardware and or time overheads (F.P. Preparata, G. Metze, and R.T. Chien. 1967.). The complexity for k faults $O(n^2RL)$, where n is the number of periodic tasks in a task set, R is the ratio of largest task period to smallest task period, and L is the number of frequency levels supported by the processor (K. Shin and Y. Lee).

4. Measure complexity in Heterogeneous System

In table 3 we have five heterogeneous systems and each system has five different modules, in the table their module wise complexity is given in the table. Now we will calculate the overall complexity of the system (T_C).

Scenario1. In H_system 1 we have a very simple system and no module is running in parallel, all the modules are running in sequence, simply by adding the complexity of each module will give the overall complexity of the system *(i.e.* $T_c = n^2 + n \log n + 2^n + \log n + n$*).* $T_C = 2^n$ .It is an easy example and these types of systems are rare in use.

Now we have more dynamic system with inter connections and multiple dependencies with distributed, parallel or other techniques of advance computing.

Scenario2. In H_system2 some of the modules are running parallel, so approach to measure the complexity will be different. If module 3 and module 5 are running in parallel fashion, so the highest complexity from both the module will be taken in to the account (the complexity of module 3 and module 5 is n^3 and n respectively so will take n^3 highest one), so the total complexity (T_c) of the system H_system 2 will be $T_c = \log n + n! + n^3 + n \log n$, $T_C = n^3$. These types of systems are most common one. Some time all the modules are running in parallel manner then the highest complexity will be the Total complexity *(*T_c*)* of the system. In actual it may not be that simple and many more parameters may affect overall complexity.

Scenario3. Heterogeneous system may include one or more critical module like security where it cannot be compromised in context to complexity, so in this type of cases one module can affect the overall system's complexity. Like in banking system or online payment system security, atomicity, accuracy, consistency, isolation and durability are modules those cannot be compromised we have to take care these modules, they can increase the total complexity of the system.

Scenario4. In this scenario we may have some dependencies. One module may dependent on another one so the complexity will increase. Like in H_system 3 we can't start the second module until first module is not finished if the complexity of module one is high then automatically the overall system's complexity will get effected and output of Module 3 is dependent module 2 if output varies then it may has effect on T_C.

Scenario5. Now we have system in which two modules (module 1and module 2) are running in parallel and have one complex module 3 the output of module 5 is dependent on module 4. If we do some computations on client side rather than to do all computations on server side, we do so the complexity of module 3 is decreased n^4 to n. the overall system complexity will automatically decreased.

There are various other factors that will play significant roles in the overall complexity. Communication is one of them. Depending upon the distance of locations where servers may be placed due to security or feasibility of the application. Backup is another issue where system needs some time to reconsolidate with the back end links or system. Depending upon the types of backup (hot or cold backup) different timings may be taken up by the system. In case of database application there may be delays due to commit and other updates on various types of storage media, which needs to be taken into account. These, will be looked into in our future study of this area.

Conclusion

This paper gives relative comparison of various techniques to analyze the complexity like Apriori Analysis, Posterior Analysis ,Micro Analysis ,Macro Analysis, Amortized Analysis ,Big *O* Notation ,Theta Notation, Potential Method ,Accounting Method and also lists there advantages and disadvantages, But still people face problem determining the complexity of complex systems. Many points has come up relating to various specific problems formulation and various design technique elements giving an insight into the area of complexity analysis of algorithms.

In this type of system, there are many algorithms that are running simultaneously as we discussed earlier like fault tolerance, authentication, encryption, routing tables, communication protocols and for error recovery. Every algorithm contributes in the system, so every algorithm have time and space complexity, if the algorithms are running parallel then the complexity will be highest out of them. E.g. we have complexities n, n+1, n2 then the complexity will be n^2. If we have N algorithms and they are running sequentially then the complexity will be sum of all these.eg. We have N algorithms running simultaneously then the total complexity will be $Tc = 1+2+\ldots\ldots+N$.

References

Allan Bording and Ran El-Yaniv(1998). Online Computation and Competitive Analysis Cambrige University Press.20, 141. ISBN 0-521-56392-5.

Complexity Analysis of Routing Algorithms in computer network, M. Bieliková (Ed.), IIT.SRC 2005, April 27, 2005, pp. 69-76.

Design And Analysis of Algorithms By R.Panneerselvam Prentice-Hall India. ISBN:978-81203-3278-2.

F.P. Preparata, G. Metze, and R.T. Chien, "On the connection assignment problem of diagnosahlc systems." IEEE Truns. Eleclron. Compirt.,Vol. C-16. pp. 848-854, Dec. 1967.

K. Chandy, J. Browne, C. Dissly and W. Uhrig, "Analytic models for rollback and recovery strategies in data base systems," *IEEE Transactions on Software Engineering,* Vol. 1, pp. 100-110, Mar. 1975.

K. Shin and Y. Lee, "Error detection process-model, design and its impact on computer Performance," *IEEE Transactions on Computers.*

Task feasibility analysis and dynamic voltage scaling in fault-tolerant real-time embedded systems Ying Zhang; Chakrabarty, K.Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings Volume 2, Issue, 16-20 Feb. 2004 Page(s): 1170 - 1175 Vol.2

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms,* Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Chapter 17: Amortized Analysis, pp.405–430.

Vol. C (33), pp. 529-540, Jun 1984.

Table 1. Comparison of all Methods

Apriori Analysis	Require less effort because we don't actually implement it. Less risky,Simple and Easy
Posterior Analysis	Whole effort may go waste if analysis gives negative data. Actually resources or infrastructure is required. Not applicable in many instances.
Micro Analysis	Checks all instruction. It takes more time,Not useful for large code.
Macro Analysis	Check only dominant operation It takes less time comparative to Micro Analysis As useful as micro analysis
Big*O* Notation	Represent worst case growth of an algorithm in time and space when they are dependent on *n.* Big *O* represents an upper bound. Easy to calculate and widely acceptable.
Theta Notation	It represents tight bound. Useful for many problems.
Omega Notation	It represents a lower bound. Not used very often.
Amortized Analysis	Probability is not involved. Guarantees the time per operation over worst-case performance
Accounting Method	It determines the individual cost of each operation. Balancing can be done between heavy and light operations.
Potential Method	Use in Online Algorithms [1]. Good to calculate the overall cost of a data structure.

Table 2. Complexity of latency of message passing in wide area networks.

	Routing Tables	*Interval routing*
Space complexity	*O(n log Δ)*	*O(k Δ log n)*
Time complexity	*O(1)*	*O(log(k Δ))*

Note: - Δ is maximum degree of the graph, n is number of nodes in the graph, k is interval.

Table 3. Heterogeneous Systems and module wise complexity.

	Module 1	*Module 2*	*Module 3*	*Module 4*	*Module 5*
H_system 1	n^2	*n log n*	2^n	*log n*	*N*
H_system 2	*log n*	*n!*	n^3	*n log n*	*n*
H_system 3	n^3	*N*	*log n*	*O(1)*	*n log n*
H_system 4	n^2	*(log n) / 2*	n^4	*N*	*O(1)*
H_system 5	n^2	*n log n*	$n^2 \log n$	$(\log n)^2$	*n log n*

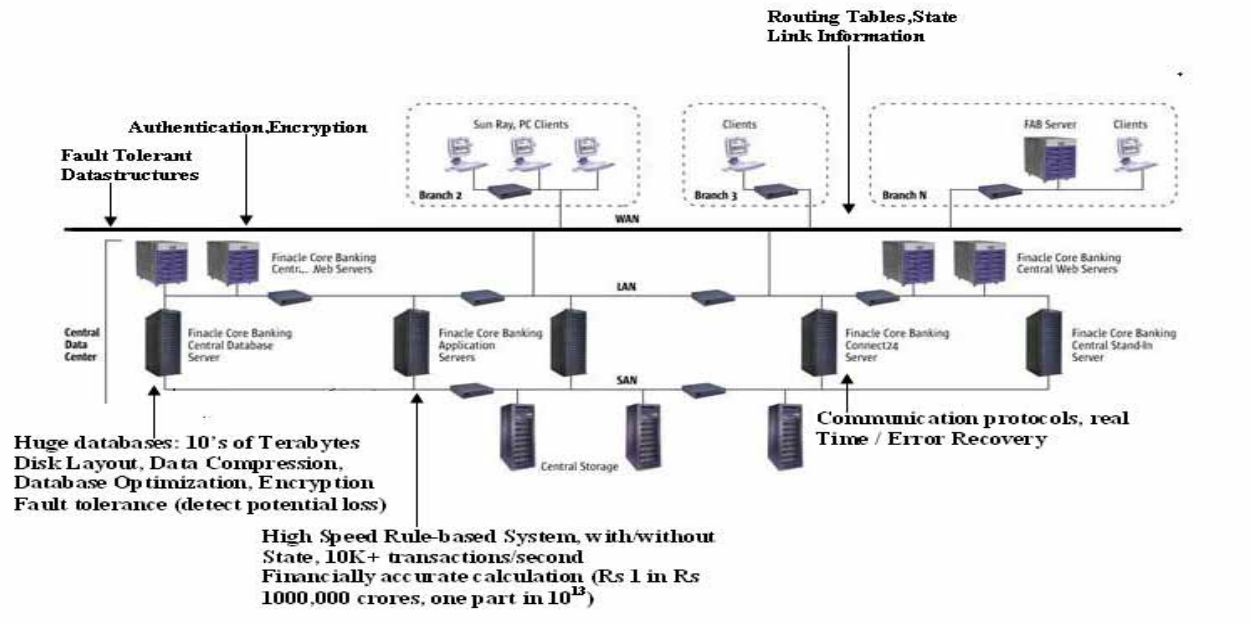

Figure 1. An Hetrogeneous system