ISSN No.-2321-8711

International Journal of Software Engineering and Systems

Volume No. 12

Issue No. 2

May - August 2024

ENRICHED PUBLICATIONS PVT.LTD

JE - 18,Gupta Colony, Khirki Extn,
Malviya Nagar, New Delhi - 110017.
E- Mail: info@enrichedpublication.com
Phone :- +91-8877340707

ISSN No.-2321-8711

International Journal of Software Engineering and Systems

Aims and Scope

Software Engineering has become very important with the ever-increasing demands of the software development to serve the millions of applications across various disciplines. For large software projects, innovative software development approaches are vital importance. In order to gain higher software standards and efficiency, software process adaptation must be derived from social behavior, planning, strategy, intelligent computing, etc., based on various factors. International journals of software engineering address the state of the art of all aspects of software engineering, highlighting the all tools and techniques for the software development process. The journals aims to facilitate and support research related to software engineering technology and the applications. International journals of software engineering welcomes the original research paper, review papers, experimental investigation , surveys and notes in all areas relating to software engineering and its applications. The following list of sample-topics its by no mean to be understood as restricting contributions to the topics mentioned:

Ø Aspect-oriented software development for secure software

Ø Dependable systems

Ø Experience related to secure software system

Ø Global security system

Ø Maintenance and evolution of security properties

Ø Metrics and measurement of security properties

Ø Process of building secure software

ISSN No.-2321-8711

ISSN No.-2321-8711

International Journal of Software Engineering and Systems

(Volume No. 12, Issue No. 2, May - August 2024)

Contents

ISSN No.-2321-8711

Temporal Rule Mining In Clinical Databases

Prof.(Dr.) T. Muthukumar
International Institute of Health Management Research, New Delhi - 110075
Email: tmkumar13@yahoo.co.in, Mobile: 0-9871969455.

ABSTRACT

This paper deals with temporal rule mining with periodicity detection from data available in heterogeneous data sources containing clinical data, in my work, two types of periodicity are considered namely Symbol Periodicity "and Segment Periodicity. Data cleaning and transformation are performed on each clinical data source and then they are stored in a temporal database designed specifically to store clinical data. Data retrieval is carried out through TSQL that combines the features of Structured Query Language (SQL) as well as Allen's Interval Algebra. The main advantage of this system is that it stores and manipulates current data as well as historical data, which are a tedious process in conventional database systems. Moreover, Temporal rules are generated from the data stored in the temporal database using an extended version of apriori algorithm. A user interface has been designed for ease of use and to perform summery as well as analysis operations on data, and also to display the results to the user in a comprehensible manner.

Keywords: Temporal rule mining, Temporal reasoning, Time series analysis, Clinical Databases.

1. INTRODUCTION

Conventional databases were designed to capture the most recent data that is current data. On the other hand, temporal databases have been designed to handle past, present and future data effectively. The insight denoting the precision in a temporal database is called the granularity of the time [3]. Temporal database supports two forms of time namely valid time and transaction time and every tuple is associated with time stamp. In my work, I take into consideration both valid time and transaction time. The reason is a patient would have undergone the same test more than once on the same day. For example, a patient would undergo the test for ECG more than once on a particular day. Temporal Data Mining is a rapidly evolving area of research that is an the intersection of several disciplines, including statistics, time series analysis, temporal pattern recognition, optimization, visualization, high-performance computing, and parallel computing. Periodicity mining is a data mining technique that is used for predicting trends in the clinical data stored in the temporal database.

Here, two types of periodicity are considered namely Symbol Periodicity and Segment Periodicity [1]. Segment periodicity concerns the periodicity of the entire time series, whereas symbol periodicity concerns the periodicities of various symbols or values of the time series. Data retrieval is carried out through TSQL that combines the features of SQL as well as Allen's Interval Algebra [3].

Currently, clinical databases are predominantly queried directly using Structured Query Language (SQL).

Physicians, researchers, and administrators who need information retrieved from a database have to request that an experienced SQL programmer, and also has intimate knowledge of the underlying database structure as well as at strong medical foundation, perform the queries for them. However SQL is not suitable if physician would like to perform a "what if analysis or prediction Rule generation finds its role here".

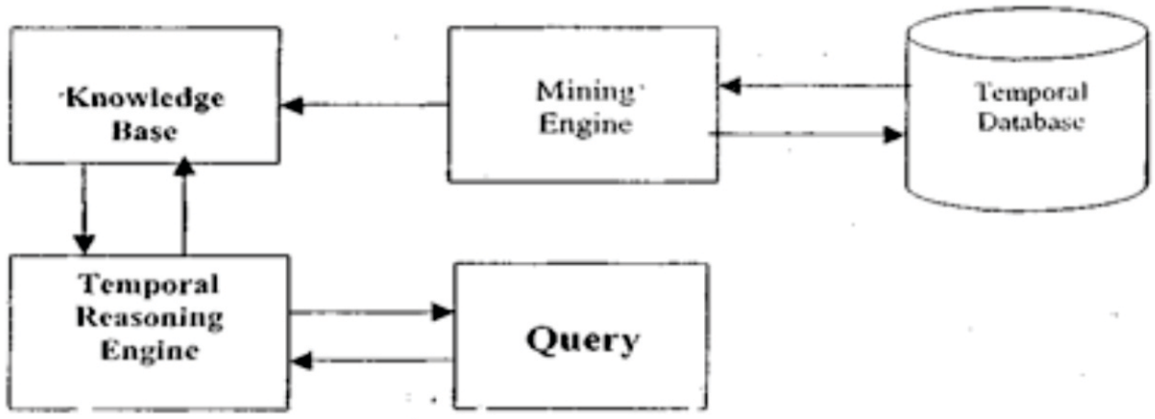


Fig 3.1- System Architecture

2. RELATED WORK

Symbol periodicity and segment periodicity for time series data and algorithms for periodicity detection in time series data has been discussed in [1]. In [1], symbol periodicity addresses the periodicity of the symbols in the time series and segment periodicity addresses the periodicity of the entire time series regarding its segments. A general model for mining asynchronous periodic patterns in temporal databases has been discussed in [2], [4]. Contributions in the area of linear temporal sequences and their interpretation using midpoint relationships have been discussed in [3]. [3] Uses Allen's Interval relationship algebra, to refine the overlaps relationship to consider midpoints for equal length intervals. A web based health data analysis tool, which was developed by CSIRO, has been discussed in [5]. A temporal data model and temporal query language TSQL with extensive time-processing capabilities

has been discussed in [6]. The summary of spatial-temporal data models, state oriented view of historical databases, and temporally grouped and ungrouped models has been discussed in [7]. Temporal reasoning with an example is discussed in [8], [9], and [10]. How ever, they used forest data and statistical techniques for prediction. Predictive Data Mining techniques discussed in [11] gives technical aspects and requirements of common mining tools for predictive data mining process and they have proposed some useful algorithms.

3. SYSTEM DESIGN

The architecture of my rule mining system is shown in Fig.3.1. The major components of my system include Temporal Database, Database Manager, and Mining Engine. Knowledge Based, and Temporal Reasoning Engine. Mining Engine is the heart of our system. Mining Engine is shown in Fig 3.2. It is further divided in to following subsystems namely Rule Generation Engine, Singular Periodic Pattern Mining Engine, Multi event Periodic Pattern Mining Engine, Complex Periodic Pattern Mining Engine, and Asynchronous Sequence Mining Engine.

Data cleaning and transformation is performed on each clinical data source and then stored in a temporal database designed specifically to store clinical data. Database Manager is a general-purpose software system that facilitates processes of defining, constructing and manipulating databases for various applications. Temporal Reasoning Engine is linked with Allen's interval Algebra [3] to execute WHEN clause. Knowledge discovered from a database can be represented by a set of rules. Rule Mining Engine generates two types of rules namely Static Rule Mining, and Temporal Rule Mining

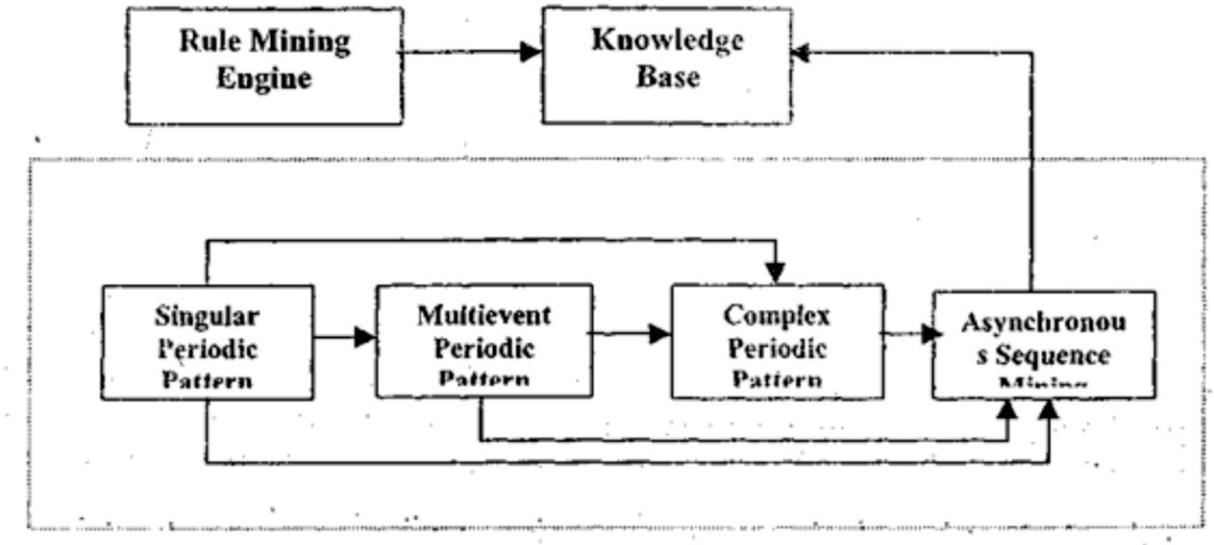


Figure 3.2: Architecture of Mining Engine

Static rule mining generates rules of the form 80% of patients who had symptoms X and Y also had symptom Z. Temporal rule mining generates rules of the form 80% of patients who had symptoms X and Y will develop symptom / during the temporal interval Tl, T2.

Temporal rules aid physicians in decision making. Mining Engine is responsible for performing periodic pattern mining. Periodic pattern may be singular event, multi event or complex pattern. SP Miner (Singular Periodic Pattern Mining) is utilized to discover all valid segments for each single event from database using potential cycle detection and hash-based validation mechanism. Then, two algorithms, MP Miner (Multi event Periodic Pattern Mining) and CP Miner (Complex Periodic Pattern Mining), are devised to discover valid segments for multi event 1-patterns and complex patterns.

In this system, Segment periodicity concerns the periodicity of the entire time series, whereas symbol periodicity concerns the periodicities of various symbols or values of the time series. This work has been developed by extending the work discussed in [1], in which for each periodicity type, a convolution-based algorithm has been proposed and analyzed, both theoretically and empirically. Segment periodicity detection needs a mapping scheme simpler than that the symbol periodicity detection since what matter is that the total number of matches is considered rather than, the symbols that match. Moreover, Segment periodicity detection algorithm proposed in this work takes less execution time whereas symbol periodicity detection algorithm proposed in this work is capable of detecting more periods. In my work, periodicity is applied in the temporal database, containing clinical data to find out interval, and instant tuple values.

4. EXPERIMENTAL RESULTS

Dataset used for testing my system has been collected from scan centers, hospitals and from the websites. In my system, database is created in oracle is used as a buck end and Java is used as a front end. TSQL is simulated using Java and translator is used to convert TSQL queries in to SQL queries. Execution times for queries range from approximately two seconds to ten minutes, depending on the complexity of the database query and the number of patients retrieved. Multithreading techniques are used to allow multiple queries to execute concurrently, preventing "downtime" while complex retrievals are underway.

5. CONCLUSION AND FUTURE WORK

The major contribution of this work is the provision of the special query language TSQL that can automatically invoke the temporal reasoning and temporal data-mining operators. This system used Allen's interval Algebra for temporal query processing. Data mining concepts have been used for performing reasoning using temporal intervals of care relevant to hospitalization episodes. The periodicity detection feature provided in this system used Symbol and Segment Periodicity detection algorithms. Further work in this direction could be the provision of distributed database features and intelligent agents for effective decision-making.

6. REFERENCES

[I] Molnuned G.EIfeky. Wulid G.Aref Ahmed A. Eimagcirinid. "Periodicity Detection in Time Series Databases", IEEE Transactions on Knowledge and Data Engg. 'Col. 17, No. 7,pp.875-887. July2005.

[2] Kuo-Yu /lining and Cha-Chu Chang,."SMCA: A General Model for Mining Asynchronous Periodic Patterns in Temporid tkilabascs ", IEEE Transactions on Know/edge and Data Engg, Vol.17, No.6, pp.7'74-785 , June 2005.

[3] John Roddick and Carl II. Mooney, "Linear temporal Sequences and Their Interpretation Using Midpoint Relations/lips", IEEE Transactions on Knowledge and Data Engg, Vol. 17, No.l pp.132-155. January 2005.

[4] J. Yang. JK Wang, and P.S. Yu, "Mining Asynchronous Periodic Patterns in Time Scries Data ". IEEE Transactions on Knowledge and Data Engg, Vol.15, No.3. pp. 613-628, 2003.

[5] A Grid Infrastructure for mixed bioinformatics Data & Text Mining By Moustafa Ghanem, Yike Guo, Anthony Rowe - 2011.

[6] Jaiwei Han, Micheline Kamber.Data Mining concepts and Techniques. (Morgan Kaufmann publishers - 2011).

[7] Damicn McAullay, G.Williams, and Jie Chen. " A Delivery framework for Health Dalit Mining and Analytics", 28t Australian Computer Science Conference, Vol.38, pp.,i-7, 2002.

[8] A.Kannan and T.V.Geetha, lemporal Reasoning with intelligent Databases". Artificial Intelligence Quarterly. India. Vol.11. No. 4. pp. i 1-20. October 1999.

[9] A Kuumoi and T.V.Geetha. " A logic for Tcn!f>und Reasoning", Communicated in IEEE Transactions on Knowledge and Data r.ngy.. I '.VI, 1999.

[10] .1 Kannan and T.V.Geetha. An iunlligKa; ,rw'i,-;; for Teinporiil Reasoning", National System Conference, pp. 201-206,October 1999.

[11] N.Jovonovic et al, " Foundations of Predictive Data Mining", 6th Seminar on Neural Network Applications in Electrical Engineering, pp.53-58. September 2002.

ISSN No.-2321-8711

The Impact Of Coding Guideline For Developing A Proficient System Integrity

Rajeev Kumar Singh
Assistant Professor, (Department of IT), L. N. Mishra College of Business Management, Muzaffarpur (An autonomous institute runs under BRA Bihar University)

A B S T R A C T

The principles and concepts that guide the coding standards are an integral part of today's safety - critical computer systems. Software verification and validation practices significantly impact the cost of achieving human - rated levels of system integrity. The stringent technical demands and expertise of the domain and the limited and specialized availability of industrial grade software products are the primary limiting factors. Robustness of software is influenced by choices in what programming languages, coding standards and testing, which is the primary form of V&V practice, are used. As a result, in these systems the state – of – the - art software uses less current programming features and techniques than those found, on the whole, in the software industry. Application of coding standards in high integrity systems software development is central to such practices. The goal of these is to make the software robust and safe and thereby contributing to overall system integrity. This paper examines the role and impact of interactions between use of the C++ programming language, the organization's own coding standards, and how these fit within V&V processes and procedures on robustness and testing as recently observed in software project. These observations will help future good software managers and engineers to make better application of coding standards throughout the V&V life cycle and optimize their impact on robustness and affordability.

Keywords: C, C++, Coding, software industry, invention, languages, Standard, optimize

1. INTRODUCTION

Coding standards grew out of the need to manage software from the earliest days of the invention of programming languages. Programmers found that it was essential to have common guidelines for understanding source code that was written by another. Although the nature of computer program source code is a distant cousin to that of literature as prose, both share the notion of style. Style found in source code represents aesthetics of human readability and comprehension based on a writer's or an engineer's taste and judgment. The expressiveness of features of a programming language is found in its specification. Languages with few and simple features were less complex to use; however, the complexity of applications and their hosting computer environment have historically been drivers for

the use of languages with more features. Such languages may satisfy one application domain but not another because the complexity of its features are found hard to understand nor safe nor predictable. This is often the case with hard real - time embedded systems. Current solutions in the use of programming languages such as C, C++ and others consist of a series of constraints that narrow the use of the features of a programming language. There are three funnels that distill down to the baseline – programming environment for the project's software developer. The first step is to start with the specification for the programming language. Depending on the needs of the domain a language subset may be defined. In practice this is done with significant investment on the part of a software vendor or the open source community. This subset specification is used to implement a compiler that only supports source code that is compliant withthe narrowed specification. With this subset there may be a need for the creation of a coding standard. It may be asked why the needs for a coding standard have. If the language or its subset were sufficiently matched to the application domain then, in principle, the coding standard would not be needed. In practice, however, it is needed because programmers must have project specific conventions that allow them to share one others' code and for the need for flexibility. As the life cycle of a project goes through its phases, experience may uncover additional rules to add to the standard based on lessons learned. The places where the project's software stakeholders can exert on their use of the compiler are in the rules of the coding standard. If it is not feasible to sufficiently cover software robustness and safety through the application of the coding standard then a remedy has to be sought with the vendor.

Most serious software development projects use coding guidelines. These guidelines are meant to state what the ground rules are for the software to be written: how it should be structured and which language features should and should not be used. Curiously, there is little consensus on what a good coding standard is. Among the many that have been written there are remarkable few patterns to discern, except that each new document tends to be longer than the one before it. The result is that most existing guidelines contain well over a hundred rules, sometimes with questionable justification. Some rules, especially those that try to stipulate the use of white-space in programs, may have been introduced by personal preference; others are meant to prevent very specific and unlikely types of error from earlier coding efforts within the same organization. Not surprisingly, the existing coding guidelines tend to have little effect on what developers actually do when they write code. The most dooming aspect of many of the guidelines is that they rarely allow for comprehensive tool-based compliance checks. Tool-based checks are important, since it is often infeasible to manually review the hundreds of thousands of lines of code that are written for larger applications.

3. PURPOSE OF THE RESEARCH

The benefit of existing coding guidelines is therefore often small, even for critical applications. A verifiable set of well-chosen coding rules could, however, make critical software components more thoroughly analyzable, for properties that go beyond compliance with the set of rules itself. To be effective, though, the set of rules has to be small, and must be clear enough that it can easily be understood and remembered. The rules will have to be specific enough that they can be checked mechanically. To put an easy upper-bound on the number of rules for an effective guideline, I will argue that we can get significant benefit by restricting to no more than eighteen rules. Such a small set, of course, cannot be all-encompassing, but it can give us a foothold to achieve measurable effects on software reliability and verifiability. To support strong checking, the rules are somewhat strict – one might even say Draconian. The trade-off, though, should be clear. When it really counts, especially in the development of safety critical code, it may be worth going the extra mile and living within stricter limits than may be desirable. In return, we should be able to demonstrate more convincingly that critical software will work as intended.

The choice of language for a safety critical code is in itself a key consideration, but we will not debate it much here. At many organizations, JPL included, critical code is written in C. With its long history, there is extensive tool support for this language, including strong source code analyzers, logic model extractors, metrics tools, debuggers, test support tools, and a choice of mature, stablecompilers. For this reason, C is also the target of the majority of coding guidelines that have been developed. For fairly pragmatic reasons, then, our coding rules primarily target C and attempt to optimize our ability to more thoroughly check the reliability of critical applications written in C. The following rules may provide benefit, especially if the low number means that developers will actually adhere to them. Each rule is followed with a brief rationale for its inclusion.

This paper covers the use of a coding standard in a e-commerce software project and provides a balance of introduction and detail of its specific rules; it is not meant to serve as a treatise. It also provides discussion of a tool used to automatically check source code compliance according to the standard. The standard used by the software project serves as the definitive writing style for onboard source code. However, the standard encompasses more than just writing style. Within the scope of the flight critical project discussed below, design and testing are part of it. The rules covering design and testing, although only a small percentage of the total number, have ramifications to program and project level stakeholders across multiple organizations and disciplines within NASA and other government agencies and contractors with responsibility for high integrity systems. The paper assumes that the

background of the audience is similar to these stakeholders and includes individuals from disciplines of system engineering and integration (SE&I), and flight software managers and software developers. When working with onboard flight software, these stakeholders are tied together through the application of the coding standard. This is best seen in the context of software V&V lifecycle. One point of this paper is to make a significant case of how critical just one coding rule can impact the operational integrity of a mission. A historical case study is provided in terms of how stakeholders' requirements collectively led to a deviation in coding practice with unintended consequences in flight and ultimately to a mishap.

V&V process changes can modify the project's coding standard. This paper discusses how the impact of deviations is managed within the life cycle of a high integrity flight software project. The paper discusses how design and testing rules may result in source code that is not compliant to the standard and how resolution of compliance deviations may need the consensus of stakeholders' to choose what trade - offs are appropriate.

3. CODING GUIDELINE FOR PROFICIENT SYSTEM INTEGRITY

The following guideline may provide benefit, especially if the low number means that developers will actually adhere to them. There are number of fundamentals guideline which may be followed with a brief rationale for its inclusion.

- **Guideline:** When use loops then all loops must have a fixed upper-bound. It must be trivially possible for a checking tool to prove statically that a preset upper-bound on the number of iterations of a loop cannot be exceeded. If the loop-bound cannot be proven statically, the rule is considered violated.

- **Rationale:** The absence of recursion and the presence of loop bounds prevent runaway code. This rule does not, of course, apply to iterations that are meant to be non-terminating (e.g., in a process scheduler). In those special cases, the reverse rule is applied: it should be statically provable that the iteration cannot terminate. One way to support the rule is to add an explicit upper-bound to all loops that have a variable number of iterations (e.g., code that traverses a linkedlist). When the upper-bound is exceeded an assertion failure is triggered, and the function containing the failing iteration returns an error.

- **Guideline:** Restrict all code to very simple control flow constructs – do not use goto statements and

- **Rationale:** Simpler control flow translates into stronger capabilities for verification and often results in improved code clarity. The banishment of recursion is perhaps the biggest surprise here. Without recursion, though, we are guaranteed to have an acyclic function call graph, which can be exploited by code analyzers, and can directly help to prove that all executions that should be bounded are in fact bounded.

- **Guideline:** Can not use dynamic memory allocation after initialization.

- **Rationale:** This rule is common for safety critical software and appears in most coding guidelines.

The reason is simple: memory allocators, such as malloc, and garbage collectors often have unpredictable behavior that can significantly impact performance. A notable class of coding errors also stems from mishandling of memory allocateion and free routines: forgetting to free memory or continuing to use memory after it was freed, attempting to allocate more memory than physically available, overstepping boundaries on allocated memory, etc. Forcing all applications to live within a fixed, pre-allocated, area of memory can eliminate many of these problems and make it easier to verify memory use. Note that the only way to dynamically claim memory in the absence of memory allocation from the heap is to use stack memory. In the absence of recursion (Rule 2), an upper-bound on the use of stack memory can derived statically, thus making it possible to prove that an application will always live within its pre-allocated memory means.

- **Guideline:** The assertion density of the code should average to a minimum of two assertions per function. Assertions are used to check for anomalous conditions that should never happen in real-life executions. Assertions must always be side-effect free and should be defined as Boolean tests. When an assertion fails, an explicit recovery action must be taken, e.g., by returning an error condition to the caller of the function that executes the failing assertion. Any assertion for which a static checking tool can prove that it can never fail or never hold violates this rule. (I.e., it is not possible to satisfy the rule by adding unhelpful "assert (true)" statements.)

- **Rationale:** Statistics for industrial coding efforts indicate that unit tests often find at least one defect per 10 to 100 lines of code written. The odds of intercepting defects increase with assertion density. Use of assertions is often also recommended as part of a strong defensive coding strategy. Assertions can be used to verify pre- and post-conditions of functions, parameter values, return values of functions, and loop-invariants. Because assertions are side-effect free, they can be selectively disabled after testing in performance-critical code.

A typical use of an assertion would be as follows:

```
if (!c_assert(p >= 0) == true) {

return ERROR;

}
```

With the assertion defined as follows:

```
#define c_assert(e) ((e) ? (true) : \

tst_debugging("%s,%d: assertion '%s' failed\n", \

   FILE , LINE , #e), false)
```

In this definition, FILE and LINE are predefined by the macro preprocessor to produce the filename and line-number of the failing assertion. The syntax #eturns the assertion condition e into a string that is printed as part of the error message. In code destined for an embedded processor there is of course no place to print the error message itself – in that case, the call to tst_debugging is turned into a no-op, and the assertion turns into a pure Boolean test that enables error recovery from anomalous behavior.

- **Guideline:** Should be declared data objects at the smallest possible level of scope.

- **Rationale:** This rule supports a basic principle of data-hiding. Clearly if an object is not in scope, its value cannot be referenced or corrupted. Similarly, if an erroneous value of an object has to be diagnosed, the fewer the number of statements where the value could have been assigned; the easier it is to diagnose the problem. The rule discourages the re-use of variables for multiple, incompatible purposes, which can complicate fault diagnosis.

- **Guideline:** The return value of non-void functions must be checked by each calling function, and the validity of parameters must be checked inside each function.

- **Rationale:** This is possibly the most frequently violated rule, and therefore somewhat more suspect as a general rule. In its strictest form, this rule means that even the return value of print f statements and file close statements must be checked. One can make a case, though, that if the response to an error would rightfully be no different than the response to success, there is little point in explicitly

checking a return value. This is often the case with calls to print f and close. In cases like these, it can be acceptable to explicitly cast the function return value to (void) – thereby indicating that the programmer explicitly and not accidentally decides to ignore a return value. In more dubious cases, a comment should be present to explain why a return value is irrelevant. In most cases, though, the return value of a function should not be ignored, especially if error return values must be propagated up the function call chain. Standard libraries famously violate this rule with potentially grave consequences. See, for instance, what happens if you accidentally execute strlen(0), or strcat(s1, s2, -1) with the standard C string library – it is not pretty. By keeping the general rule, we make sure that exceptions must be justified, with mechanical checkers flagging violations. Often, it will be easier to comply with the rule than to explain why non-compliance might be acceptable.

- **Guideline:** The use of pointers should be restricted. Specifically, no more than one level of dereferencing is allowed. Pointer dereference operations may not be hidden in macro definitions or inside typedef declarations. Function pointers are not permitted.

- **Rationale:** Pointers are easily misused, even by experienced programmers. They can make it hard to follow or analyze the flow of data in a program, especially by tool-based static analyzers. Function pointers, similarly, can seriously restrict the types of checks that can be performed by static analyzers and should only be used if there is a strong justification for their use, and ideally alternate means are provided to assist tool-based checkers determine flow of control and function call hierarchies. For instance, if function pointers are used, it can become impossible for a tool to prove absence of recursion, so alternate guarantees would have to be provided to make up for this loss in analytical capabilities.

- **Guideline:** The use of the preprocessor must be limited to the inclusion of header files and simple macro definitions. Token pasting, variable argument lists (ellipses), and recursive macro calls are not allowed. All macros must expand into complete syntactic units. The use of conditional compilation directives is often also dubious, but cannot always be avoided. This means that there should rarely be justification for more than one or two conditional compilation directives even in large software development efforts, beyond the standard boilerplate that avoids multiple inclusion of the same header file. Each such use should be flagged by a tool-based checker and justified in the code.

- **Rationale:** The C preprocessor is a powerful obfuscation tool that can destroy code clarity and befuddle many text based checkers. The effect of constructs in unrestricted preprocessor code can be extremely hard to decipher, even with a formal language definition in hand. In a new implementation

of the C preprocessor, developers often have to resort to using earlier implementations as the referee for interpreting complex defining language in the C standard. The rationale for the caution against conditional compilation is equally important. Note that with just ten conditional compilation directives, there could be up to 210 possible versions of the code, each of which would have to be tested– causing a huge increase in the required test effort.

- **Guideline:** No function should be longer than what can be printed on a single sheet of paper in a standard reference format with one line per statement and one line per declaration. Typically, this means no more than about 60 lines of code per function.

- **Rationale:** Each function should be a logical unit in the code that is understandable and verifiable as a unit. It is much harder to understand a logical unit that spans multiple screens on a computer display or multiple pages when printed. Excessively long functions are often a sign of poorly structured code.

- **Guideline:** All code must be compiled, from the first day of development, with all compiler warnings enabled at the compiler's most pedantic setting. All code must compile with these setting without any warnings. All code must be checked daily with at least one, but preferably more than one, state-of-the-art static source code analyzer and should pass the analyses with zero warnings.

- **Rationale:** There are several very effective static source code analyzers on the market today, and quite a few freeware tools as well. 2There simply is no excuse for any software development effort not to make use of this readily available technology. It should be considered routine practice, even for non-critical code development. The rule of zero warnings applies even in cases where the compiler or the static analyzer gives an erroneous warning: if the compiler or the static analyzer gets confused, the code causing the confusion should be rewritten so that it becomes more trivially valid. Many developers have been caught in the assumption that a warning was surely invalid, only to realize much later that the message was in fact valid for less obvious reasons. Static analyzers have somewhat of a bad reputation due to early predecessors, such as lint, that produced mostly invalid messages, but this is no longer the case. The best static analyzers today are fast, and they produce selective and accurate messages. Their use should not be negotiable at any serious software project.

- **Guideline:** Understanding the software architecture and create interfaces that are consistent with it.

- **Guideline:** Select meaningful variables name and follow other local coding standards.

- **Guideline:** Create nested loops in a way that makes them easily testable.
- **Guideline:** Constrain your algorithms by following structured programming practice.\
- **Guideline:** Select data structures that will meet the needs of the design.
- **Guideline:** Keep conditional logic as well as simple.
- **Guideline:** Write code that is self documented.
- **Guideline:** Create a visual layout.

The first two rules guarantee the creation of a clear and transparent control flow structure that is easier to build, test, and analyze. The absence of dynamic memory allocation, stipulated by the third rule, eliminates a class of problems related to the allocation and freeing of memory, the use of stray pointers, etc. The next few rules (4 to 7) are fairly broadly accepted as standards for good coding style. Some benefits of other coding styles that have been advanced for safety critical systems, e.g., the discipline of "design by contract" can partly be found in rules 5 to 7. These ten rules are being used experimentally at JPL in the writing of mission critical software, with encouraging results. After overcoming a healthy initial reluctance to live within such strict confines, developers often find that compliance with the rules does tend to benefit code clarity, analyzability, and code safety. The rules lessen the burden on the developer and tester to establish key properties of the code (e.g., termination or boundedness, safe use of memory and stack, etc.) by other means. If the rules seem Draconian at first, bear in mind that they are meant to make it possible to check code where very literally your life may depend on its correctness: code that is used to control the airplane that you fly on, the nuclear power plant a few miles from where you live, or the spacecraft that carries astronauts into orbit. The rules act like the seat-belt in your car: initially they are perhaps a little uncomfortable, but after a while their use becomes second-nature and not using them becomes unimaginable.

REFERENCES

1. *ISO/IEC 14882-2011(E) Information technology—Programming languages—C++, 3rdedition, 2011- 09-01*
2. *ISO/IEC 14882-2011(E) Information technology—Programming languages—C++, 3rdedition, 2011- 09-01*
3. *Stroustrup, B., "Software Development for Infrastructure," Computer",IEEE Computer Society Press, Vol. 45, Jan, pp. 47,58*
4. *NASA Procedural Requirements 7150.2B,URL:http://nodis3.gsfc.nasa.gov/*
5. *Avionics Application Software Standard Interface ARINC Specification 653 -1, October 16, 2003, Published by Aeronautical Radio, Inc.*
6. *Joint Strike Fighter Air Vehicle C++ Coding Standards for the System Development and Demonstration Program, Document Number 2RDU00001*
7. *LDRA Software Technology, http://www.ldra.com/*
8. *RTCA/DO - 178B, "Software Considerations in Airborne Systems and Equipment Certification," December 1, 1992, URL: www.rtca.org*
9. *Lions, J.L., Ariane 5 Flight 501 Failure Report, http://www.di.unito.it/~damiani/ariane5rep.html*
10. *Embedded C++ Homepage Official Website, http://www.caravan.net/ec2plus/*
11. *Hayhurst,K. J., Veerhusen, D.S., Chilenski, J.,J., Rierson,L.k., NASA/TM-2001-210876, A Practical Tutorial on Modified Condition/Decision Coverage, May 2001, http://www.sti.nasa.gov*
12. *URL: www.scrum.org*
13. *IEEE 829, Standard for Software and System Test Documentation, URL: http://standards.ieee.org/findstds/standard/829 -2008.html*
14. *Storey, N., Safety - Critical Computer Systems, Addison Wesley, 1996, pp. 314. Rev C, December 2005, www.stroustrup.com/JSF-AV-rules.pdf*
15. *NASA Systems Engineering Handbook NASA-SP-2007-6105 Rev1,URL:http://www.acq.osd.mil /se/docs/NASA-SP-2007-6105-Rev-1-Final-31Dec2007.pdf*
16. *Software Engineering – A Practitioner's Approach - 6th Edition – Roger S. Pressman Page No. 144 - 146*
17. *Oron Exploration Flight Test-1, http://www.nasa.gov/pdf/663703main_flighttest1_fs_051812.pdf*
18. *[WIR71] Wirth, N., "Program Development by Stepwise Refinement," CACM, vol. 14, no. 4, 1971, pp. 221-22719.[ROS75] Ross, D., J. Goodenough, and C. Irvine, "Software Engineering: Process, Principles, and Goals," IEEE Computer, vol. 8, no. 5, may 1975*
20. *[DAV95] Davis, A., 201 Principles of Software Development, McGraw-Hill, 1995*
21. *[MCC93] McConnell, S., Code Complete, Microsoft Press, 1993*

ISSN No.-2321-8711

Multi Domain Analysis Tool

R. Rajashekaran, Ms B Jayanthi
M.Phil Scholar, Department of Computer Science ,Rathnavel College of Arts & Science College, Sulur,Coimbatore ,Tamilnadu, India.
Head - Computer Science, School of Computer Studies(UG),Rathnavel Subramaniam College of Arts and Science,Sulur,Coimbatore -641 023 , Tamil Nadu

ABSTRACT

This paper refers Multi Domain Analysis Tool" (MDAT) to analyze the multiple domain codes such as Java ,.Net , HTML. The tool is a platform independent tool developed using Data mining techniques in Java, that helps to detect performance bottlenecks by graphically displaying profiling data. The Tool checks for the efficiency of the program. The efficient code reduces memory usage and increases speed , efficiency of applications and is examines based on time and memory factors also fetch system details on which the tool is running. This is an automated tool and hence reduces the time for the client to gain details regarding the system.Testing is a process of executing a program with the intent of finding an error. Testing is the analysis of source/executable code and the controlled execution of executable code to reveal defects that compromise a program's executable integrity.

1. INTRODUCTION

I.1 INTRODUCTION:

Multi Domain Analysis Tool (MDAT) is a user - friendly and fully atomized tool for analyzing performance of Java applications. This Performance Analysis Tool is a GUI-based tool. Thus the execution of the Analysis Tool could generate a view that the user is in need of to judge the total performance of Application.

Memory and Time are the main constraints that are considered at the execution of any program. These basic and inevitable constraints are considered as the main part of computerizing and programming. So a program is expected to occupy less memory space and execute as fast as Possible. The objective of this analysis tool is to tune the program so that it can efficiently utilize memory and execute as fast as possible.

I.2 DATA MINING :

data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

II. EXISTING SYSTEM

Performance of the applications (.Net,Java,HTML) is usually done in some of the tools by the programmer or the analyst, such as Program analyzer tool , Performance analyzer tool ,Memory Analyzer , Doxygen ,Source CodeNavigator in now a days .The above tools processing takes more time and it is not possible to analyze accurately. Time consumption is more and also the application is not completely tested. The existing tool cost is very high and could not be affordable to very small applications. The available systems are useful and applicable to large sized programming and very well standardized applications and hence could not fit through all small programs. In the existing system any application is analyzed or tested slowly. Here the performance of any application is analyzed slowly & without full security.

II.1. DISADVANTAGES:

- The tools are not user friendly
- Time overhead and time consumption is more
- Higher chances of missing some conditions in the application
- Complete analysis of the program is not sometimes possible
- Not High security
- The performance and testing tool exist individually
- The cost factor is very high for the existing system
- Some times may leave certain simple flaws undetermined which may later rise as a complex problem
- The tools are not user friendly
- The analysts or programmers alone check for the efficiency which may not be flexible for clients or end users.

III. PROPOSED SYSTEM

The proposed system is a automated tool. That designed for the performance analysis and testing of Various (,Net, Java,Html) applications together with the system configuration details. As this system runs in Java environment it overcomes the drawbacks of the existing system by its features. The tool is machine and platform independent. It is GUI based and is at a very economical, affordable cost. The proposed system being user friendly, it enables the entry level users and clients to access easily. The efficiency of the various applications is determined depending upon the system on which it is installed. The tool can be extended to analyze the performance and also conduct testing for Serves, .Net,HTML, EJB and JSP applications.

III.1 DISADVANTAGES:

- User friendly
- Reduces testing and debugging
- Simple and compact
- Platform and machine independent
- Improves the application reliability
- Prevents the simple errors from becoming complex
- Increases the code reusability
- Any level of users can easily access the tool
- Reduces the maintenance time

IV. MODULES

IV.1. PERFORMANCE ANALYSIS:

This module checks for the efficiency of the program. The efficient code reduces memory usage and increases speed. The performance analysis module checks for the efficiency of applications (.Net,Java,Html) and is examines based on time and memory factors. The factors that involve performance of Various (.Net, Java, Html) applications. Such as the used variables, unused variables, optimizing loops, file compiling, memory usage, time computation etc. are determined. This analysis is carried out so as to provide the client with the best optimized application. Time based execution of the loops is done to optimize the loops. A programmer utilizes this timing detail to list out unnecessary loops. The classes and methods used in the applications (.NET, Java, Html) are also listed out.

MDAT lists the used and unused variables by which the unused variables can be eliminated by the programmer for efficient memory Utilization. It gives the details about number of classes and methods used in a source code and also the timing details for compilation and execution.

IV.1 SYSTEM CONFIGURATION DETAILS:

A system configuration detail gives system details on which the tool is running. This is an automated tool and hence reduces the time for the client to gain details regarding the system. This helps in comparing the performance of the application by executing in systems with different configurations. This module provides detail regarding memory that is memory used and frees memory. The operating system installed is also known. The drives in the system are known and the memory usage is listed in this module. System configuration details are generated by the tool. As this tool is developed using java and hence it can run on any platform and just pressing a tab lists the details about Memory capacity and RAM details

IV 3.TESTING:

Testing is a process of executing a program with the intent of finding an error. Testing is the analysis of source/executable code and the controlled execution of executable code to reveal defects that compromise a various (.NET ,JAVA ,HTML)program's executable integrity. Defects often lead to erratic behavior or the premature termination of an executing program. A good test is one that has a high probability of finding an error. The objective is to design tests that systematically uncover different classes of errors and to do with a minimum Amount of time and effort. This module performs Testing Is Carried out in This Module

- White-Box Testing

V. ALGORITHM USED: K-NEAREST NEIGHBOR (DATA MINING)

Classification (generalization) using an instance- based classifier can be a simple matter of locating the nearest neighbor in instance space and labeling the unknown instance with the same class label as that of the located (known) neighbor. This approach is often referred to as a nearest neighbor classifier. The downside of this simple approach is the lack of robustness that characterizes the resulting classifiers. The high degree of local sensitivity makes nearest neighbor classifiers highly susceptible to noise in the training data.

More robust models can be achieved by locating k, where k > 1, neighbors and letting the majority vote decide the outcome of the class labeling. A higher value of k results in a smoother, less locally sensitive, function. The nearest neighbor classifier can be regarded as a special case of the more general k-nearest neighbors classifier, hereafter referred to as a kNN classifier. The drawback of increasing the value of k is of course that as k approaches n, where n is the size of the instance base, the performance of the classifier will approach that of the most straightforward statistical baseline, the assumption that all unknown instances belong to the class most frequently represented in the training data.

An arbitrary instance is represented by $(a_1(x), a_2(x), a_3(x),.., a_n(x))$ ai(x) denotes features

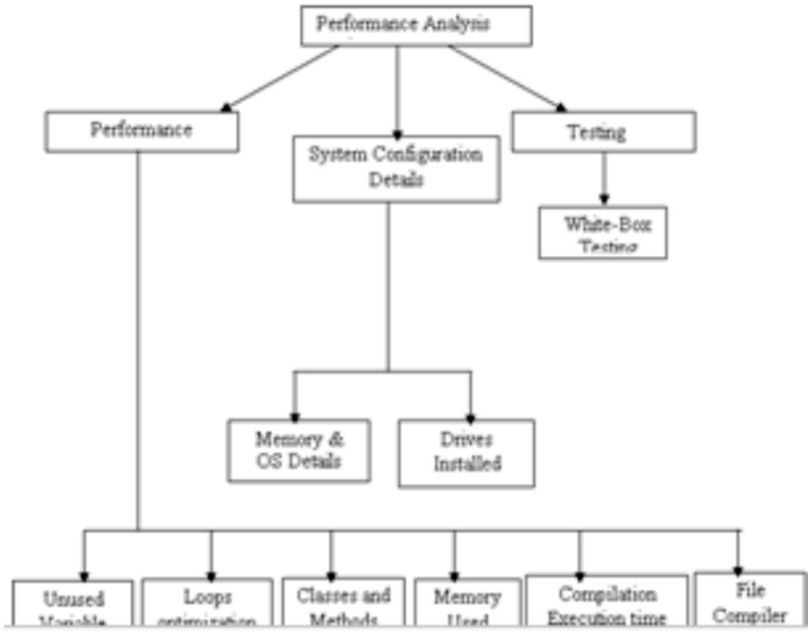
Euclidean distance between two instances d(xi, xj)=sqrt (sum for r=1 to n (ar(xi) - ar(xj))2)Distance-Weighted Nearest Neighbor Algorithm

- Assign weights to the neighbors based on their 'distance' from the query point Weight 'may' be inverse square of the distances
- All instances correspond to points in the n-D space
- The nearest neighbor are defined in terms of Euclidean distance, dist(X_1, X_2)
- Target function could be discrete- or real- valued
- For discrete-valued, k-NN returns the most common value among the k training examples nearest to x_q

Distance Calculated:

Co efficient = p/p+q+R
Distance =Q+R/P+Q+R
P=No. of Variables Positive for Both Objects
Q= No. of Variables Positive in Q not R
R= No. of Variables Positive in R not Q

VI. RESULT:



VII. CONCLUSION:

The MDAT tool is designed with the future in mind. There may occur changes to all the systems. Likewise this system is also subjected to changes and advancements to meet ever the requirements of the customer. The generation of reports is under process which can be an enhancement for the application. It helps the Care has been taken to assist in the needs for future development. The software is constructed along the lines suggested by the users. By making necessary changes efficiency of the system can be improved This system can be extended as per advancements and requirements. the future changes the changes in the languages can be made. The security can be highly increased. The generation of methods is under process which can be an enhancement for the application. New modules can be added to the existing system with less effort.

VIII. ABBREVIATIONS:

MDAT - Multiple Domain Analysis Tool
JDK - Java Developer's Kit
JVM - Java Virtual Machine
OS - Operating System
RAM - Random Access Memory
MS - Mille Second – ms units

IX. REFERENCES

1. Elias Awath, "SYSTEM ANALYSIS AND DESIGN", Tata Mc Graw Hill Publication, Sixth Edition, 2003.
2. Ivon Horton, "JAVA PROGRAMMING", Tata McGraw Hill Publication, Second Edition 1999.
3. O'Reilly, "JAVA SWINGS", Tata McGraw Hill Publication, Fourth Edition, 2003.
4. Patrick Naughton and Herbert Schildth, "JAVA 2", Tata McGraw Hill Publication, 1999
5. Data mining Basic Concepts https://en.wikipedia.org
6. Data mining Classification https://en.wikipedia.org
7. Data mining techniques tutorial http://www.tutorialspoint.com
8. Datamining Packages in Java http://www.jcp.org.

ISSN No.-2321-8711

Survey On Grid Computing And Parallel Computing

Basant Namdeo
Lecturer International Institute of Professional Studies, Devi Ahilya University, Takshashila Parisar, Indore, MP, India

A B S T R A C T

Grid computing is useful for scientific workloads. It has a basic foundation of distributed computing, parallel computing, mainframe and many more computing paradigms. Understanding of proper classification of grid computing services and scheduling algorithms are needed to exploit it. This paper focuses on survey based on scheduling of grid and parallel computing. Thus this paper provides a roadmap for students, and researchers to study available literature.

Key words- Parallel computing, Grid computing, Applications of Grid, Application of Parallel Computing, Parallel computer.

1. INTRODUCTION

Parallel computing could be a sort of computation during which several calculations are administered at the same time, in operation on the principle that giant issues will typically be divided into smaller ones, that are then resolved at the same time ("in parallel") [1]. In science, it is always required to solve the big problems in a reasonable amount of time. This requirement led us to develop massively parallel system to solve problems like videos processing to scientific calculation. Computer architecture is divided by their number of instruction and number of data on which these instruction works. They are divided by Flynn's taxonomy. Flynn's taxonomy is a classification of computer architectures, proposed by Michael J. Flynn in 1966[7]. This taxonomy say there are 4 type of Computer system SISD (Single Instruction Single Data), SIMD (Single Instruction Multiple Data), MISD (Multiple Instruction Single Data) and MIMD(Multiple Instruction Multiple Data).

Grid computing is that the collection of computing resources from multiple locations to achieve a standard goal. The grid may be thought of as a distributed system with non-interactive workloads that involve an oversized range of files [2]. In grid computing environment each computing resource is fully capable of doing its works. Grid computing is always done in loosely coupled systems.

APPLICATIONS

GRID COMPUTING APPLICATIONS

Typical applications that need grid computing are forecasting and military state of affairs simulations, database etc, where we need our works has to done fast and with less resources:

High-throughput computing support:- High-throughput computing support permits applications to use grids to place unused processor cycles to figure in usually loosely coupled [3]. For example there are 3 computing devices and each can do all sort of work, and there are two computing devices who are working on full capacity and one is sitting ideal, then the overall throughput is less than that can be achieved.

Resource sharing: In Grid computing there may be chances that all the computing locations have different number / sort of resources. They can be shared in several completely different organizations that enable alternative organizations (i.e. users) to access them. This will reduce the cost of computation.

Data Driven computing: Modern database servers also use grid computing for various purpose. They can use remote execution-program and data is sent to the remote server for execution, remote data extraction- data can be extracted from remotely available server by distributed SQL and remote process execution- execute PL/SQL block of code on least loaded remote server.

Parallel Computing Applications

Parallel computing can be used in various fields, where we can divide the problem in sub-parts and each part can be solved parallel. Some of fields are:

Computer graphics and Gaming: creating graphics in computer is very lengthy task. If we create some scene sequentially then it will took lots of time, although we know, we can create various objects parallel. In computer games, we have to create new scene vary fast in user interaction.

Weather Forecasting: Weather forecasting also require lots of processing of data to get conclusion and it should be done in time bound manner.

Database Centric Applications: Extracting useful information from database is also requires applying complex business rules and processing. Database software must also runs its queries parallel, so that we can get right information in right time.

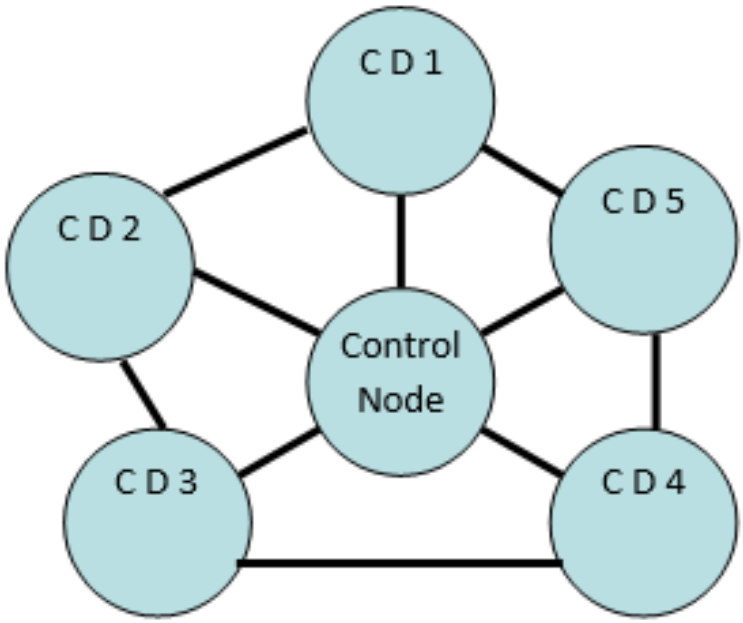
Statistics: Parallel processing can be used in statistical problems also like Least Median Squares Regression, Random Number Generators, Parallel Monte Carlo and Extensions, Monte Carlo Estimation of Multidimensional Integrals [5].

There are lots of other areas where parallel computing is used like Medical imaging and diagnosis, Pharmaceutical design, advanced graphics and virtual reality, particularly in the entertainment industry, Bioscience, Biotechnology and Genetics etc[6].

Areas Where Grid and Parallel Computation Cannot be Used: There are some areas or some scene where this type of computing methodology can not be used :

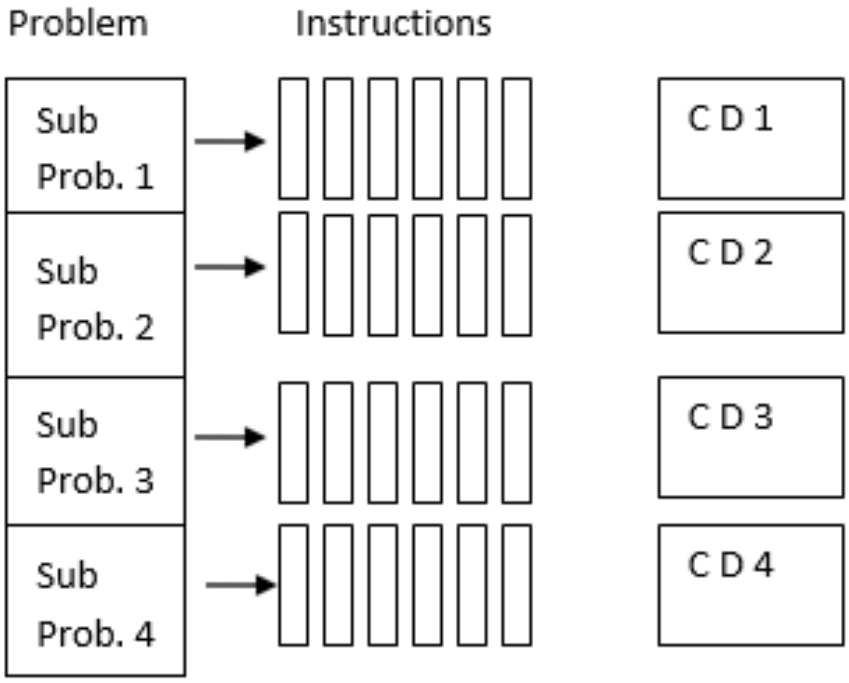
- If data required for one sub-problem is dependent on previous or other sub problems output. Then here is data dependency.

- If problem cannot be sub divided in parallel executable parts.



C D : Computing Device

Figure 1 Grid Computing



C D: Computing Device

Figure 2 Parallel Computing

CONCLUSION:

Grid computing and parallel computing is the requirement of today's computing environment. There are lots of application areas where they can be used. For using these methodology we have to first develop algorithms which can be executed in concurrent manner.

REFERENCES:

1 *https://en.wikipedia.org/wiki/Parallel_computing*
2 *http://en.wikipedia.org/wiki/Grid_computing*
3 *Bote-Lorenzo, Miguel L., Yannis A. Dimitriadis, and Eduardo Gómez-Sánchez. "Grid characteristics and uses: a grid definition." Grid Computing. Springer Berlin Heidelberg, 2004.*
4 *Navarro, Cristobal A., Nancy Hitschfeld-Kahler, and Luis Mateu. "A survey on parallel computing and its applications in data-parallel problems using GPU architectures." Communications in Computational Physics 15.02 (2014): 285-329.*
5 *Beddo, Vanessa. Applications of parallel programming in Statistics. Diss. University of California Los Angeles, 2002.*
6 *https://computing.llnl.gov/tutorials/parallel_comp/*
7 *Flynn, M. J. (September 1972). "Some Computer Organizations and Their Effectiveness". IEEE Trans. Comput. C–21 (9): 948–960. doi:10.1109/TC.1972.5009071*

ISSN No.-2321-8711

Web Data Extraction – Tools And Techniques

Dr. Jatinder Kumar
Assistant Prof., A. S. College, Khanna

ABSTRACT

: In order to make effective decisions, the data should be in strand and structured form, but web data are semi- structured or even unstructured, which cannot be manipulated by traditional database techniques, it is imperative to extract web data to port them into databases for further handling. The purpose of Web Information Extraction (IE) in our web mining research support system is to extract a specific portion of web documents useful for a research project. Such techniques if implemented backed with good data mining techniques will prove to be boon for the decision makers.

Keywords: Data extraction, Web Mining, Decision Making

A specific web data set can be scattered among different web hosts and have different formats. IE takes web documents as input, identifies a core fragment, and transforms that fragment into a structures and unambiguous format.

1. WRAPPERS

Information extraction on the web brings us new challenges. The volume of web documents is enormous, documents change often, and contents of documents are dynamic. Designing a general web IE system is a hard task. Most IE systems use a wrapper to extract information from a particular web site. A wrapper consists of a set of extraction rules and specific code to apply rules to a particular site. A wrapper can be generated manually, semi- automatically or automatically.

The manual generation of a wrapper requires writing ad hoc codes based on the creator's understanding of the web documents. Programmers need to understand the structure of a web page and write codes to translate it. Techniques have been developed to use expressive grammars, which describe the structure of a web page, and to generate extraction codes based on the specified grammar. TSIMMIS is an example of manual wrapper. TSIMMIS focuses on developing tools to facilitate the integration of heterogeneous information sources. A simple self- describing model. Languages and tools are developed to support the wrapping process.

The manual way of wrapper cannot adapt to the dynamic changes of web sites. If new pages appear or the format of existing sources is changed, a wrapper must be modified to adapt the new change.

Semi- automatic wrapper generation uses heuristic tools to support wrapper generation. In such a system, users to hypothesize the underlying structure of the whole web site provide sample pages. Based on the hypothesized structure, wrappers are generated to extract the information from the site. This approach does not require programmer knowledge about web pages, but demonstration of sample pages are required for each new site.

Using machine learning and data mining techniques to learn extraction rules or patterns can generate wrappers automatically. These systems can train themselves to learn the structure of web pages. Learning algorithms must be developed to guide the training process.

2. WRAPPER GENERATION TOOLS

Many tools have been created to generate wrappers. Such tools include Languages for Wrapper development, NLP based Tools, Modeling based Tools and Ontology- based Tools.

Some languages are specifically designed to assists users to address the problem of wrapper generation. Minerva combines the benefits of a declarative and grammar- based approach with the edibility of procedural programming by enriching regular grammars with an explicit exception- handling mechanism. Minerva defines the grammar in Extended Backus- Naur Form (EBNF) style. Which adds the regular expression syntax of regular languages to the BNF notation, in order to allow very compact specifications. It also provides an explicit procedural mechanism for handling exceptions inside the grammar parser. Web OQL (Object Query Language) is a declarative query language, which aimed at performing SQL- like queries over the web. A generic HTML wrapper parses a web page and produces an abstract HTML syntax tree. Using the syntax of the language, users can write queries to locate data in the syntax tree and output data in some formats i.e. tables. Such tools require users to examine web documents and write a program to separate extraction data.

Natural language processing (NLP) techniques are used to learn extraction rules existing in natural language documents. These rules identify the relevant information within a documents by using syntactic and semantic constraints. WHISK is an extraction system using NLP techniques. In this system,a set of extraction rules is induced from a given set of training example documents. At each iteration, instances are selected for users to tag; users use a graphical interface to add a tag for each attribute to be extracted from the instance WHISK uses the tagged instances to create rules and test the accuracy of the rules.

Web structure mining is the process of using theory to analyse the node and connection structure of a web site. According to the type of web structural data web structure mining can be divided into two kinds.

The first kind of web structure mining is extracting patterns from hyperlinks in the web. A hyperlink is a structural component that connects the web page to a different location. The other kind of the web structure mining is mining the documents structure. It is using the tree- like structure to analyse and describe the HTML (Hyper Text markup Language) or XML (eXtensible Markup Language) tags within the web page.

Modeling based tools locate portions of data in a web page according to a given target structure. The structure is provided based on a set of modeling primitives, which conform to an underling data model. NoDoSe is an interactive tool for semi- automatically determining the structure of such documents and extracting their data. Using a GUI, the user hierarchically decomposes the file, outlining its interesting regions and then describing their semantic. In the decomposition process, users build a complex-structured object and decompose it into s simpler object. Such decomposition is a training process through which NoDoSe can learn to construct object and identify other objects into the documents. This task is expedited by a mining component that attempts to infer the grammar of the file from the information the user has input so far.

All tools presented above generate extraction rules based on the structure of the data in a document. Ontology- based tools rely directly on the data. Given a specific domain application, an ontology tool can locate constants in the page and construct object with them. In that they present an approach to extract information from unstructured documents based on an application ontology that describes a domain of interest. By using the ontology, rules are formulated to extract constants and context keywords from unstructured documents on the web. For each unstructured documents of interest a recognizer is applied to organize extracted constants as attribute values of tuples in a generated database schema. This tools requires the construction of ontology by experts and the ontology construction needs to be validated.

3. OSS WEB DATA EXTRACTION

Information extraction is used to extract relevant data from web pages. A Web mining research support system should be able to search for and extract specific contents on the web efficiently and effectively. In our Open Source Software study, a web wrapper was developed to help extracting useful information from web resources.

3.1 A SIMPLE MANUAL WEB WRAPPER

A wrapper should be developed for a web mining research support system to find and gather the research related information from web sources. Although Different research projects have different web documentation formats, which lead to different web wrappers, those wrappers still have some common designs. A wrapper can automatically traverses web documents, extract information and follow links to other pages. Java, Perl, Python, etc can implement a wrapper.

Our wrapper starts with a URL, identifies other links on the HTML page visits those links, extracts information from web pages and stores information into Databases. Thus, the wrapper consists of a URL access method, a web page parser with some extractors, and databases. The access function of a wrapper should prevent repeatedly accessing the same web address and should identify dead links. The parser recognizes start tags, end tags, text and comments. The databases provide storage for extracted web information.

The key component of our web wrapper is the parser, which includes a word extractor, a table extractor and a link extractor. The word extractor is used to extract word information. It should provide a string checking function. Tables are used commonly in web pages to align information. A table extractor identifies the location of the data in the table. A link extractor retrieves links contained in a web page. There are two types of link (absolute links and relative links. An absolute Link gives the full address of a web page, while a relative link needs to be converted to a full address by adding a prefix.

3.2 OSS DATA COLLECTION

After informing Source Forge of our plans and receiving permission, we gathered data monthly at Source Forge. Source Forge provides project management tools, bug tracking, mail list services, discussion forums, and version control software for hosted project. Data is collected at the community, project, and developer level, characterizing the entire OSS phenomenon, across multiple numbers of projects, investigating behaviors and mechanisms at work at the project and developer levels.

A wrappers, implemented by Perl and CPAN (Comprehensive Perl Archive- the repository of Perl module/libraries) modules, traversed the Source Forge web server to collect the necessary data. All project home pages in Source Forge have a similar top – level design. Many of these pages are dynamically generated from a database. The web crawler uses LWP, the libwww-perl library, to fetch each project's homepage. CPAN has a generic HTML parser to recognize start tags end tags, text and

comments, etc. Because both statistical and member information is stored in tables, the web crawler uses an existing Perl Module called HTML; Table Extract and string comparisons provided by Perl to extract information. Link extractors are used if there are more than one page of members.

HYBRID INFORMATION EXTRACTION

The manual wrapper is easy to develop and implement. However users must create different wrappers for use with each change of web documents. Maintenance cost will be too high for this manual wrapper approach. Moreover, many text- like documents exist on the web, e.g. discussion board, news group.

Web contents as two types: structured/ semi- structured text and free text. The first type ahs data item (e.g. names, SSN, etc.) Example web documents of this type include on-line statistics, tables, etc. The second type consists of free languages, e.g. advertisements, messages, etc. Techniques such as wrapper induction and modeling based tools are suitable with pages of the first type because such tools rely on delimiters of data to create extraction rules. NLP techniques are based on syntactic and semantic constraints can work with both types. However grammar analysis and learning rules generations are complex. These techniques are costly for structured text. A web mining research support system must deal with both types, because both contain useful information for research. Wrapper induction techniques will be developed to extract information from a structured/ semi structured text, while NLP techniques are used for free text.

For a free text type web document, grammar and syntax are analyzed by NLP learning. Then extraction rules are generated based on the analysis. The extractor extracts data according to those extraction rules and stores extracted data into the database.

The extraction of a structured/ semi structured web documents is as follows. Firstly the parser creates a parse tree for the documents. Secondly, users input sample pattern, which they want to extract. Then, extraction heuristic are generated to match the sample patterns. Wrappers are created based on extraction heuristics to extract data. If the extracted data contains free text which needs further extraction, the process will be changes to use NLP techniques. Otherwise, data are stored into the database.

OSS APPLICATION

The hybrid information extraction system will be applied on our Open Sources Software study. Our OSS study need to collect data from several OSS web sites such as Source Forge, Savanah, Linux, Apache, Mozilla etc. These sites offer structured/ semi structured documents. e.g. membership tables, statistics tables. However, we also need to study information hiding in some free text documents. For example, we want to analyze developers activity by collecting data from their messages. We believe our hybrid information extraction system will provide efficient extraction on those sites.

REFERENCES

(01) Srivastava Data preparation for mining World Wide Web browsing patterns. Knowledge and information Systems.
(02) J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data.
(03) S.K. Pal, V Talwar, and P. Mitra Web mining in soft computing framework Relevance.
(04) K. Snadhu, and M. Shih, Clustering of web users based on access patterns.
(05) S. Soderland. Learning information extraction rules for semi- structured and free text. Machine Learning.
(06) R. Albert, H. Jeong, and A.L. Barabasi. Diameter of the World Wide Web.
(07) A. L. Barabasi Linked: The Science of Networks. Perseus Boston, 2002.
(08) Gengbank homepage. http://www.mcbi.nlm.mih.gov/ Genbank/ indec. html.
(09) Free/Open Source Software Development Phenomenon Homepage. http"//www.nd.edu/oss
(10) Swarm Homepage. http://www.swarm.org
(11) S. Lawrence and C.L. Giles Searching the World Wide Web Science.
(12) Oracle. http://oracle.com
(13) Repast homepage. http://repast.sourceforge.net/.
(14) Sourceforge homepage. http://sourceforge.net.

Instructions for Authors

Essentials for Publishing in this Journal

1 Submitted articles should not have been previously published or be currently under consideration for publication elsewhere.

2 Conference papers may only be submitted if the paper has been completely re-written (taken to mean more than 50%) and the author has cleared any necessary permission with the copyright owner if it has been previously copyrighted.

3 All our articles are refereed through a double-blind process.

4 All authors must declare they have read and agreed to the content of the submitted article and must sign a declaration correspond to the originality of the article.

Submission Process

All articles for this journal must be submitted using our online submissions system. http://enrichedpub.com/ . Please use the Submit Your Article link in the Author Service area.

Manuscript Guidelines

The instructions to authors about the article preparation for publication in the Manuscripts are submitted online, through the e-Ur (Electronic editing) system, developed by **Enriched Publications Pvt. Ltd**. The article should contain the abstract with keywords, introduction, body, conclusion, references and the summary in English language (without heading and subheading enumeration). The article length should not exceed 16 pages of A4 paper format.

Title

The title should be informative. It is in both Journal's and author's best interest to use terms suitable. For indexing and word search. If there are no such terms in the title, the author is strongly advised to add a subtitle. The title should be given in English as well. The titles precede the abstract and the summary in an appropriate language.

Letterhead Title

The letterhead title is given at a top of each page for easier identification of article copies in an Electronic form in particular. It contains the author's surname and first name initial .article title, journal title and collation (year, volume, and issue, first and last page). The journal and article titles can be given in a shortened form.

Author's Name

Full name(s) of author(s) should be used. It is advisable to give the middle initial. Names are given in their original form.

Contact Details

The postal address or the e-mail address of the author (usually of the first one if there are more Authors) is given in the footnote at the bottom of the first page.

Type of Articles

Classification of articles is a duty of the editorial staff and is of special importance. Referees and the members of the editorial staff, or section editors, can propose a category, but the editor-in-chief has the sole responsibility for their classification. Journal articles are classified as follows:

Scientific articles:

1. Original scientific paper (giving the previously unpublished results of the author's own research based on management methods).
2. Survey paper (giving an original, detailed and critical view of a research problem or an area to which the author has made a contribution visible through his self-citation);
3. Short or preliminary communication (original management paper of full format but of a smaller extent or of a preliminary character);
4. Scientific critique or forum (discussion on a particular scientific topic, based exclusively on management argumentation) and commentaries. Exceptionally, in particular areas, a scientific paper in the Journal can be in a form of a monograph or a critical edition of scientific data (historical, archival, lexicographic, bibliographic, data survey, etc.) which were unknown or hardly accessible for scientific research.

Professional articles:

1. Professional paper (contribution offering experience useful for improvement of professional practice but not necessarily based on scientific methods);
2. Informative contribution (editorial, commentary, etc.);
3. Review (of a book, software, case study, scientific event, etc.)

Language

The article should be in English. The grammar and style of the article should be of good quality. The systematized text should be without abbreviations (except standard ones). All measurements must be in SI units. The sequence of formulae is denoted in Arabic numerals in parentheses on the right-hand side.

Abstract and Summary

An abstract is a concise informative presentation of the article content for fast and accurate Evaluation of its relevance. It is both in the Editorial Office's and the author's best interest for an abstract to contain terms often used for indexing and article search. The abstract describes the purpose of the study and the methods, outlines the findings and state the conclusions. A 100- to 250-Word abstract should be placed between the title and the keywords with the body text to follow. Besides an abstract are advised to have a summary in English, at the end of the article, after the Reference list. The summary should be structured and long up to 1/10 of the article length (it is more extensive than the abstract).

Keywords

Keywords are terms or phrases showing adequately the article content for indexing and search purposes. They should be allocated heaving in mind widely accepted international sources (index, dictionary or thesaurus), such as the Web of Science keyword list for science in general. The higher their usage frequency is the better. Up to 10 keywords immediately follow the abstract and the summary, in respective languages.

Acknowledgements

The name and the number of the project or programmed within which the article was realized is given in a separate note at the bottom of the first page together with the name of the institution which financially supported the project or programmed.

Tables and Illustrations

All the captions should be in the original language as well as in English, together with the texts in illustrations if possible. Tables are typed in the same style as the text and are denoted by numerals at the top. Photographs and drawings, placed appropriately in the text, should be clear, precise and suitable for reproduction. Drawings should be created in Word or Corel.

Citation in the Text

Citation in the text must be uniform. When citing references in the text, use the reference number set in square brackets from the Reference list at the end of the article.

Footnotes

Footnotes are given at the bottom of the page with the text they refer to. They can contain less relevant details, additional explanations or used sources (e.g. scientific material, manuals). They cannot replace the cited literature.
The article should be accompanied with a cover letter with the information about the author(s): surname, middle initial, first name, and citizen personal number, rank, title, e-mail address, and affiliation address, home address including municipality, phone number in the office and at home (or a mobile phone number). The cover letter should state the type of the article and tell which illustrations are original and which are not.

Notes: