Линь М., Чжао С., Цзы С., Го П., Фань Ц.

КЛАССИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПО ОБРАБОТКЕ ДАННЫХ МЕТЕОРОЛОГИЧЕСКИХ СПУТНИКОВ / CLASSIFICATION OF METEOROLOGICAL SATELLITE GROUND SYSTEM APPLICATIONS

Аннотация. *Эффективная работа метеорологических спутниковых систем предполагает наличие не только наземных комплексов управления с системой телеметрии, но и многочисленных прикладных программных продуктов по передаче, обработке и распространению данных самих метеорологических наблюдений. Оптимизация функционирования орбитальных и наземных подсистем получения метеорологических данных с точки зрения потребления системных ресурсов и улучшения их рационального распределения требует решения задачи классификации таких программных приложений на основе характеристик их функционирования. Решение задачи классификации предполагает построение концептуальной схемы (онтологии) линейки программных продуктов путем формирования структуры данных, содержащей выбранные классы объектов, их связи и правила функционирования.*

Для решения задачи использованы многомерные статистические процедуры в виде совокупности алгоритмов упорядочивания данных на основе иерархической кластеризации программных приложений.

Для классификации выбрано несколько групп параметров: кривые загрузки ЦПУ, распределения оперативной памяти, объемов данных систем ввода-вывода и передачи по сетям связи. На основе оценки степени близости объектов в рамках итерационной процедуры обучения построен классификатор приложений (дерево принятия решений). В свою очередь, применение алгоритма большинства голосов дало возможность выявления наиболее ресурсно-емкого приложения. На конечном этапе анализа использована процедура эталонной оценки потребления ресурсов (построения рейтинга программных приложений). Предложенные алгоритмы классификации значительно повышают эффективность наземной системы обработки метеорологических данных.

Ключевые слова: *Спутник, Наземная прикладная система, Классификация, Иерархическая кластеризация, Верификация программной модели, Параметры приложения, Загрузка центрального процессора, Загрузка памяти, Загрузка ввода-вывода, Загрузка сети.*

Abstract. *Meteorological satellite ground application system carries a large number of applications. These applications deal with a variety of tasks. In order to classify these applications according to the resource consumption and improve the rational allocation of system resources, this paper introduces several application analysis algorithms. Firstly, the requirements are abstractly described, and then analyzed by hierarchical clustering algorithm. Finally, the benchmark analysis of resource consumption is given. Through the benchmark analysis of resource consumption, we will give a more accurate meteorological satellite ground application system.*

Keywords: *Satellite, Ground Application System, Classification, Hierarchical Clustering, Software Model Verification, Application Parameters, CPU-loading, Memory-loading, IO-loading, Network-loading.*

1. Introduction

Meteorological satellite application system is large and complex. Different types of applications need different resources. Accurate classification of meteorological satellite ground application systems will play a vital role in optimizing the resources of the entire system. How to classify these applications has become a more critical issue.

There have been some successes in the study of application classification. Thomas Thüm and Juristo N. [1] [2] give a classification and survey of analysis strategies for software product lines. A software product line is a family of software products that share a common set of features. Software product lines challenge traditional analysis techniques, such as type checking, model checking, and theorem proving, in their quest of ensuring correctness and reliability of software. Gabmeyer S. [3] proposed a feature-based classification of software model verification approaches. They classify a verification approach in a system-centric view according to the pursued verification goal. Srinivas C. [4], his main idea is to cluster the software components and form a subset of libraries from the available repository. These clusters thus help in choosing the required component with high cohesion and low coupling quickly and efficiently. Rashwan A. [5] contains two significant contributions: 1) a new gold standard corpus containing annotations for different NFR types, based on a requirements ontology, and 2) a Support Vector Machine (SVM) classifier to automatically categorize requirements sentences into different ontology classes. Pancerz K. [6] introduced the first version of a computer tool called CLAPSS (Classification and Prediction Software System) for solving different classification and prediction problems using, among others, some specialized approaches based mainly on rough set theory. A comparison framework is proposed as Wahono R. S. [7] studied, which aims to benchmark the performance of a wide range of classification models within the field of software defect prediction. For the purpose of this study, 10 classifiers are selected and applied to build classification models and test their performance in 9 NASA MDP datasets. Naufal M. F. [8] proposed a software defect classification using Fuzzy Association Rule Mining (FARM) based on complexity metrics. However, not all complexity metrics affect on software defect, therefore it requires metrics selection process using Correlation-based Feature Selection (CFS) so it can increase the classification performance.

This paper introduces several application analysis algorithms. We first describe the requirements in an abstract manner. Then analyzed by hierarchical clustering algorithm. Finally, the benchmark analysis of resource consumption is given. Through the above analysis, improve the classification accuracy of meteorological satellite ground application system.

2. Calculation Method of Application Type Curve

The application classification analysis function is used as the decomposition requirement of the multidimensional mode evaluation. It mainly implements the data mining of the type and operation characteristics of the application, and discovers the change rule of various performance indexes applied to the platform [9] [10]. So as to realize the classification and analysis of the running status of the application system. Through the application of classification analysis, you can visually get the target operation in the hardware resources on the consumption situation. Thus helping to determine the operational behavior of the application model. To help provide software system and hardware system optimization and recommendations. In general, the application of classification analysis includes application type curve and application type evaluation of two parts.

The application type curves appear in the product overview tables in the Jobflow Task Report. It complements the application type attribute of the product. This shows the CPU job usage (CPU_All_Idle) when the target job is running.

Apply the classification analysis to classify the target job. The results differ from the expected classification of the application, reflecting the actual deviation of the application. Application Classification describes the resource consumption tendency of the application. Upgrading or optimizing the hardware performance of such resources or optimizing the software specifications, will have a greater impact on the performance of the application.

The application of the type of curve, is based on the application of CPU resources for the analysis of the target. By summarizing the proportion of CPUs in the idle state during all scheduling of the application, a rough CPU occupancy trend and pattern is given during the entire run. By observing the application type curve, you can get an overview of the CPU usage during the run. Such as whether it is in a long time high load occupancy, whether the change over time jitter and so on.

The calculation method of the application type curve is calculated based on the time series of CPU_All_Idle (CPU idle occupancy) during each scheduling of the application. Considering that the time of each

DOI: 10.7256/2453-8817.2018.1.26055

dispatch may not be strictly consistent, we take the length of the application type curve T = min (T_1, T_2, ..., T_n). T_i is the application of the first run of the scheduling time, n is the application of the number of scheduling. Therefore, the application type curve is calculated as:

$$Y_i = \frac{1}{n}\sum_{k=1}^{n} x_{ki}, i = 1, 2, ..., T \qquad (1)$$

X_{ki} is the value of the i-th point in the CPU_All_ Idle curve when the job is run at the k-th time.

When the length of the application type curve is longer (greater than 100 seconds), the original application type curve is smoothed out for the convenience of the front display and not requiring high precision. To retain the main morphological features of the curve, and remove the curve glitches and noise and other irrelevant details.

3. Classification Analysis Algorithm

At present, application classification analysis can be divided into CPU-intensive, memory-intensive, IO-intensive, network-intensive and non-intensive 5. According to the existing data at this stage, we'll train classifier like decision tree. The classifier divides the specific classification type of an application based on the relevant characteristics extracted from the application parameters. At this stage in accordance with the needs of the classifier is now the specific details of the fixed. Waiting for subsequent updates to consider whether you need to learn and update the classifier.

3.1. Descriptive Descriptions of Demand

Application classification analysis, in essence, is a classification problem. For the target application, the application of the classification analysis needs to construct a model that analyzes the resource operating parameters (such as the CPU occupancy curve, the memory footprint curve, etc.) of the target application, constructs and learns a classifier, The data is mapped to a collection of sort labels, which can actually be abstracted into an unsupervised process. In the course of model learning, the main feature of the original data is extracted by using the feature extraction configuration item. The number of classification labels (that is, all the original data can be divided into several categories) is determined by hierarchical clustering, and then through the decision tree building classifiers, and finally adding artificial strategies, and experts to determine the specific meaning of each category.

3.2. Hierarchical Clustering Algorithm

General clustering methods, such as mixed Gaussian model (GMM) and K- Means (M-Means), are more or less faced with the choice of the number of clustering clusters or by the greater impact of initialization and other issues. In the application classification problem, you can roughly think that the name of the classification label (cluster label) includes CPU-intensive, memory-intensive and so on. Whether there are other classification labels in the model learning stage cannot be completely decided. Therefore, in the case where the number of clustering clusters cannot be completely determined, the general clustering method is difficult to achieve the effect.

Hierarchical clustering is a structured clustering method. It is based on the distance between the two sample data calculation method, based on the distance between the different samples linked to the distance. And through the separation and fusion to build a tree structure.

Assuming there are N samples to be clustered, the basic steps for hierarchical clustering are:

- (Initialize) Classify each sample into a class. Calculate the distance between two classes. That is, the similarity between the sample and the sample;
- Find the nearest two classes between classes. Classify them as a class (the total number of such classes is less);
- Recalculate the similarity between the newly generated class and each old class;
- Repeat 2 and 3 until all sample points are classified as a class. End.

In the model learning process of the classification analysis configuration item, the tree structure obtained by hierarchical clustering is shown in Figure 1.

The leaf nodes with the same color in the tree structure represent the corresponding samples belonging to a cluster.

Theoretically, most of the job streams are non-intensive for running normal workflows. Memory-intensive and CPU-intensive than IO-intensive and network-intensive occur more often. Thus, in a hierarchical clustered tree structure as shown in

Figure 1, the tree is cut by cutting at a specified tree height. All leaf nodes in a subtree are considered to belong to the same cluster. As shown in Figure 1, about 10,000 training samples are divided into approximately five categories.

CPU active state occupancy rate (CPU_ All_Us + CPU_All_Sys), the IO read/write rate (Disk_All_Read and Disk_All_ Write), network transmit and receive rates (Net_All_Recv and Net_All_Send), and other parameters as the sample feature vector elements.

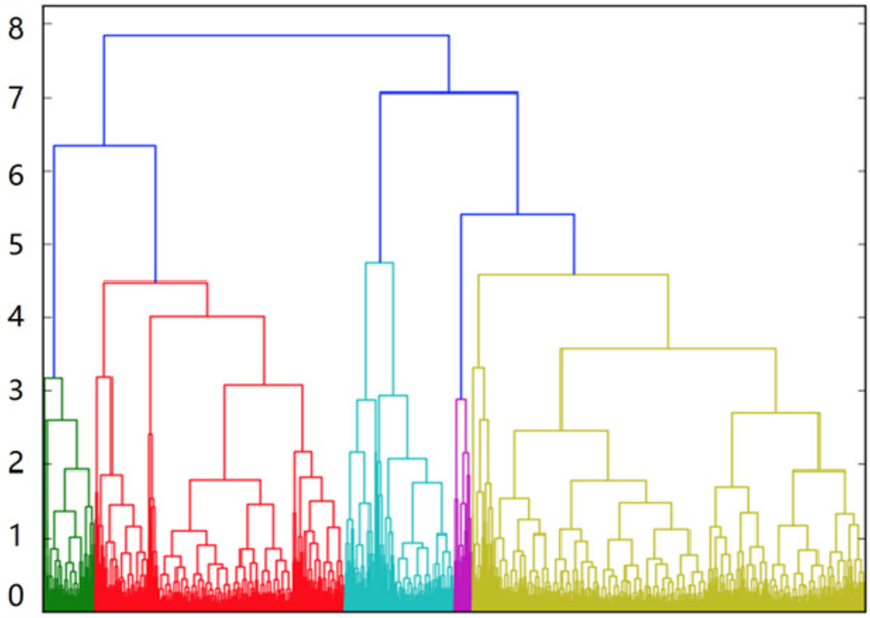

Figure 1. The picture shows the hierarchical clustering of the tree structure. (Each leaf node of the tree is a separate sample point. A total of about 10,000 sample points).

3.3. Decision Tree Classifier

A decision tree is a tree structure that describes the classification of sample data. It can be seen as a collection of if-else strategies. Any path from the root node of the decision tree to the leaf node is a rule that corresponds to an if-else. It can be shown that all the if-else rules in the decision tree are mutually exclusive and complete.

In Section 2.2, the result of the hierarchical clustering model is that for each training sample it is possible to give a given classification label without practical significance. However, considering the clustering algorithm itself is poor generalization ability, and the complexity of online learning algorithm is very high, so the application of classification analysis configuration item in the decision tree classifier, is based on hierarchical clustering training results based on the construction and trained.

Specifically, the sample data containing the non-meaningful classification tags processed by the hierarchical clustering model, the application memory occupancy rate (Mem_All_MemRatio), the And then use the CART decision tree algorithm to construct a simpler decision tree to meet the needs of generalization and online updating of subsequent models.

The decision tree classifier has been constructed in Figure 2.

The above decision tree produces the classification type of the application's scheduling. For the entire application classification, a majority voting algorithm is used. Selecting the category in which the application has the highest number of occurrences in all schedules as the category of the app itself.

4. Resource Consumption Benchmark Analysis

In the above application classification decision tree, it indicates the resource consumption reference value of a certain parameter. The so-called resource con- sumption benchmark is based on resource consumption load. Taking into account the time factor, it characterizes the total amount of resource

DOI: 10.7256/2453-8817.2018.1.26055

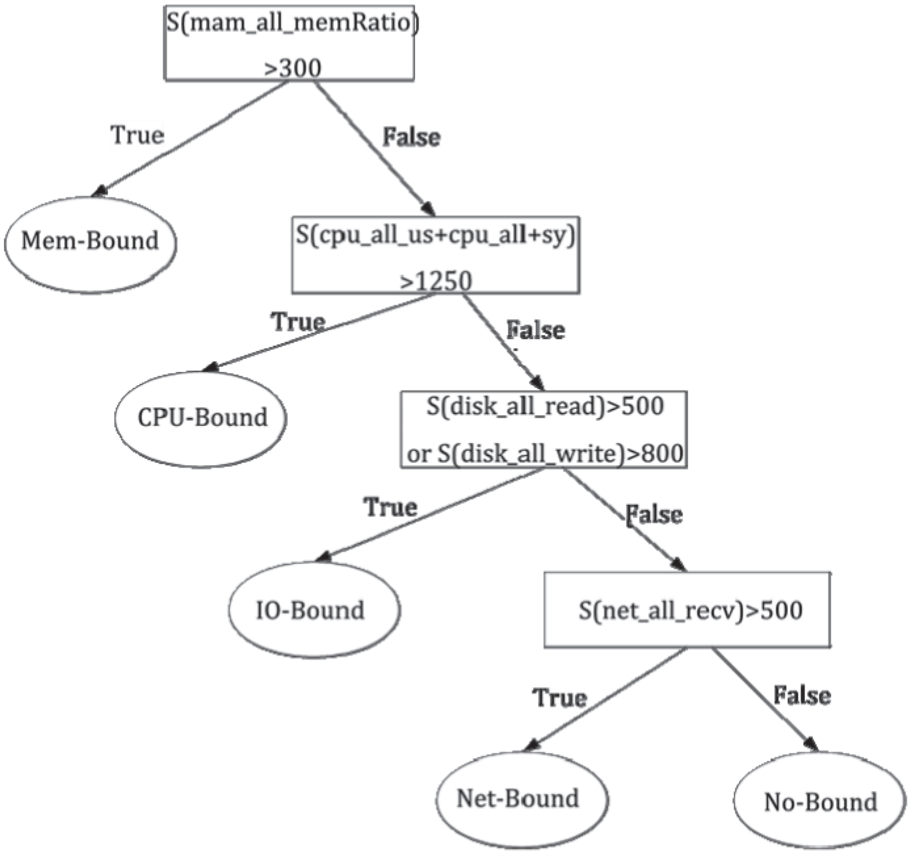

Figure 2. Decision tree classification model based on classification analysis which indicates the resource consumption baseline for a parameter.

load for the target job over the entire run time. Assuming that the resource load of a resource is X and the run time is T, the resource consumption reference value is

$$Y_i = \frac{1}{n}\sum\nolimits_{k=1}^{n} x_{ki}, i = 1, 2, \ldots, T$$
$$\text{Benchmark} = T \cdot X \qquad (2)$$

For the sake of intuition, the resource consumption baseline can be normalized. The value obtained in a [0, 1] interval is called the resource consumption score. It characterizes how much of the target resource consumes relative resources to other resources.

Theoretically, for a normal operation, the CPU resource consumption score and the memory resource consumption score are more consistent and higher relative to other resources. Therefore, you can think that when the CPU resource consumption score and memory resource consumption score is close to less than 0.4 when the operation is normal.

5. Conclusion

Ground application systems are a very important part of meteorological satellite systems engineering. It is responsible for the operation and management of satellite payloads after satellite launch, as well as receiving, processing, distributing, applying and servicing data from satellite. Especially for stationary meteorological satellites, its observing function is done through satellite and terrestrial applications. The ground application system is an integral part of the star system. Aiming at the complexity of application types and resource requirements in meteorological satellite application system, this paper studies the types of applications. In order to classify these applications according to the resource consumption and improve the rational allocation of system resources, this paper introduces several application analysis algorithms. We first described the requirements in an abstract manner and then analyzed them through hierarchical clustering algorithms. And we introduced the decision tree classifier on this basis. Finally, the benchmark analysis of resource consumption is given. The analysis shows that this paper has high accuracy for the application of resource type analysis.

Acknowledgments

The work presented in this study is supported by National High-tech R&D Program (2011AA12A104).

Статья впервые опубликована:
Atmospheric and Climate Sciences. 2017. Vol. 07. No. 3.

Библиография

1. Thüm, T., Apel, S., Schaefer, I., et al. (2014) A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Computing Surveys, 47, 1-45. https://doi.org/10.1145/2580950.
2. Gómez, O.S., Juristo, N. and Vegas, S. (2014) Understanding Replication of Experiments in Software Engineering: A Classification. Information & Software Technology, 56, 1033-1048. https://doi.org/10.1016/j.infsof.2014.04.004.
3. Gabmeyer, S., Kaufmann, P. and Seidl, M. (2013) A Classification of Model Checking-Based Verification Approaches for Software Models. Proceedings of the STAF Workshop on Verification of Model Transformations (VOLT 2013), Budapest, 17 June 2013, 1-7.
4. Srinivas, C., Radhakrishna, V. and Rao, C.V.G. (2014) Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries. Procedia Computer Science, 31, 1044-1050. https://doi.org/10.1016/j.procs.2014.05.358.
5. Rashwan, A. and Ormandjieva, O. (2013) Ontology-Based Classification of Non-Functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. IEEE, Computer Software and Applications Conference. IEEE Computer Society, Kyoto, 22-26 July 2013, 381-386. https://doi.org/10.1109/COMPSAC.2013.64.
6. Pancerz, K. (2015) On Selected Functionality of the Classification and Prediction Software System (CLAPSS). International Conference on Information and Digital Technologies, IEEE, Zilina, 7-9 July 2015, 278-285. https://doi.org/10.1109/DT.2015.7222984.
7. Wahono, R.S., Herman, N.S. and Ahmad, S. (2014) A Comparison Framework of Classification Models for Software Defect Prediction. Advanced Science Letters, 20, 1945-1950. https://doi.org/10.1166/asl.2014.5640.
8. Naufal, M.F. and Rochimah, S. (2016) Software Complexity Metric-Based Defect Classification Using FARM with Preprocessing Step CFS and SMOTE a Preliminary Study. International Conference on Information Technology Systems and Innovation. IEEE, Al Ain, 28 November 2016, 1-6.
9. Peng, J., Elias, J.E., Thoreen, C.C., et al. (2003) Evaluation of Multidimensional Chromatography Coupled with Tandem Mass Spectrometry (LC/LC-MS/MS) for Large-Scale Protein Analysis: The Yeast Proteome. Journal of Proteome Research, 2, 43-50. https://doi.org/10.1021/pr025556v.
10. Lorencatto, F., Gould, N.J., McIntyre, S.A., et al. (2016) A Multidimensional Approach to Assessing Intervention Fidelity in a Process Evaluation of Audit and Feedback Interventions to Reduce Unnecessary Blood Transfusions: A Study Protocol. Implementation Science, 11, 163. https://doi.org/10.1186/s13012-016-0528-x.

DOI: 10.7256/2453-8817.2018.1.26055

References

1. Thüm, T., Apel, S., Schaefer, I., et al. (2014) A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Computing Surveys, 47, 1-45. https://doi.org/10.1145/2580950.
2. Gómez, O.S., Juristo, N. and Vegas, S. (2014) Understanding Replication of Experiments in Software Engineering: A Classification. Information & Software Technology, 56, 1033-1048. https://doi.org/10.1016/j.infsof.2014.04.004.
3. Gabmeyer, S., Kaufmann, P. and Seidl, M. (2013) A Classification of Model Checking-Based Verification Approaches for Software Models. Proceedings of the STAF Workshop on Verification of Model Transformations (VOLT 2013), Budapest, 17 June 2013, 1-7.
4. Srinivas, C., Radhakrishna, V. and Rao, C.V.G. (2014) Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries. Procedia Computer Science, 31, 1044-1050. https://doi.org/10.1016/j.procs.2014.05.358.
5. Rashwan, A. and Ormandjieva, O. (2013) Ontology-Based Classification of Non-Functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier. IEEE, Computer Software and Applications Conference. IEEE Computer Society, Kyoto, 22-26 July 2013, 381-386. https://doi.org/10.1109/COMPSAC.2013.64.
6. Pancerz, K. (2015) On Selected Functionality of the Classification and Prediction Software System (CLAPSS). International Conference on Information and Digital Technologies, IEEE, Zilina, 7-9 July 2015, 278-285. https://doi.org/10.1109/DT.2015.7222984.
7. Wahono, R.S., Herman, N.S. and Ahmad, S. (2014) A Comparison Framework of Classification Models for Software Defect Prediction. Advanced Science Letters, 20, 1945-1950. https://doi.org/10.1166/asl.2014.5640.
8. Naufal, M.F. and Rochimah, S. (2016) Software Complexity Metric-Based Defect Classification Using FARM with Preprocessing Step CFS and SMOTE a Preliminary Study. International Conference on Information Technology Systems and Innovation. IEEE, Al Ain, 28 November 2016, 1-6.
9. Peng, J., Elias, J.E., Thoreen, C.C., et al. (2003) Evaluation of Multidimensional Chromatography Coupled with Tandem Mass Spectrometry (LC/LC-MS/MS) for Large-Scale Protein Analysis: The Yeast Proteome. Journal of Proteome Research, 2, 43-50. https://doi.org/10.1021/pr025556v.
10. Lorencatto, F., Gould, N.J., McIntyre, S.A., et al. (2016) A Multidimensional Approach to Assessing Intervention Fidelity in a Process Evaluation of Audit and Feedback Interventions to Reduce Unnecessary Blood Transfusions: A Study Protocol. Implementation Science, 11, 163. https://doi.org/10.1186/s13012-016-0528-x.