Data Mining Based Diagnosis in Resource Management

Mathias Beck[1] and Jorge Marx Gómez[2]

[1] Technical University of Clausthal, Germany

[2] Otto-von-Guericke-Universität, Magdeburg, Germany

Abstract. There are different solutions to resource allocation problems in Resource Management Systems (RMS). One of the most sophisticated ways to solve these problems is an adjustment to Quality-of-Service (QoS) settings during runtime. These settings affect the trade-off between the resource usage and the quality of the services the executed tasks create. But, to be able to determine the optimal reactive changes to current QoS settings in an acceptable time, knowledge of the resource allocation problem's cause is necessary. This is especially significant in an environment with real-time constraints. Without this knowledge other solutions could be initiated, still an improvement to the current resource allocation, but the optimal compromise between resource requirements and QoS is likely to be missed. A resource management system (RMS) with the ability to adjust QoS settings can solve more resource allocation problems than one providing reallocation measures only. But problem-depending only optimal changes to QoS settings can solve the problem within timing constraints and thus prevent expensive system failures. Depending on the environment a RMS is used in, the failures could be a huge financial loss or even a threat to human lives. But the knowledge of a problem's cause does not only help to solve the problem within existing timing constraints and to guarantee feasibility of the executed tasks, but helps to maximize the quality of the generated services as well. To detect upcoming problems in time, forecasting mechanisms can be integrated into the RMS. They can predict a problem in the near future, early enough for the system to react. For diagnosis of resource management problems, data mining can be applied to determine the cause of an allocation problem. The techniques implemented in this work are the k-nearest neighbor analysis and decision trees. Both techniques will make their predictions based on prior created resource allocation snapshots referring to problem cases with known cause.

1 Introduction

There are different solutions to resource allocation problems in Resource Management Systems (RMS). One of the most sophisticated ways to solve these problems is, if supported by the RMS, an adjustment to Quality-of-Service (QoS) settings during runtime. These settings affect the trade-off between the resource usage and the quality of the services the executed tasks create. But, to be able to determine the optimal reactive changes to current QoS settings in an acceptable time, knowledge of the resource allocation problem's cause is necessary. This is especially significant in an environment

Beck M. and Marx Gómez J. (2005).
Data Mining Based Diagnosis in Resource Management.
In *Proceedings of the 5th International Workshop on Pattern Recognition in Information Systems*, pages 177-185
DOI: 10.5220/0002534801770185

with real-time constraints. Without this knowledge other solutions could be initiated, still an improvement to the current resource allocation, but the optimal compromise between resource requirements and QoS is likely to be missed. A resource management system (RMS) with the ability to adjust QoS settings can solve more resource allocation problems than one providing reallocation measures only. But problem-depending only optimal changes to QoS settings can solve the problem within timing constraints and thus prevent expensive system failures. Depending on the environment a RMS is used in, the failures could be a huge financial loss or even a threat to human lives. “The real-time and reliability constraints require responsive rather than best-effort metacomputing.” ([2]) In general problems in a RMS arise whenever over-load or under-load on an attached node occurs. Under-load occurs when the current resource usage falls below a certain minimum usage. This simply results in a high idle time and indicates potential for optimization. If other hosts are much busier or new tasks are waiting to be assigned to a computing node a resource reallocation can improve the overall performance. In case of a QoS aware RMS under-load could also, if necessary or possible, trigger an increase in the currently executed task’s QoS level. Overload is a problem indicating that the resources are currently operating close to their maximum capacity. If other nodes in the RMS still operate with higher idle times, a resource reallocation might be able to solve the problem. If all hosts are operating close to their capacity limits, a decrease of the currently executed task’s QoS level can improve the systems condition. While unsolved overload problems in a timing insensitive environment only result in longer response times, the effects in a real-time environment are more severe. In case of QoS aware RMS overload indicates that the current QoS level might not be feasible anymore in the near future and that the QoS level has to be reduced. If overload problems in a real-time environment can not be solved, timing sensitive deadlines of one or more tasks in the systems can be missed. Violating the timing constraints of a real-time application system can, especially in case of military defense systems or aviation control systems, result in very costly system failures. To prevent these violations it is important that the available resources are allocated in a way meeting all task deadlines that would result in an immediate system failure when missed. To detect upcoming problems in time, forecasting mechanisms can be integrated into the RMS. They can predict a problem in the near future, early enough for the system to react. To determine the cause of resource allocation problems or forecasted problems a technique already successfully applied to multiple business-related problems and in [4] to resource performance analysis can be used: data mining. The techniques implemented in this work are the k-nearest neighbor analysis and decision trees. Both techniques will make their predictions based on prior created resource allocation snapshots referring to problem cases with known cause. Once the cause of a resource allocation problem is determined, this knowledge can help to choose the optimal measures to solve the problem. Different solutions to a resource allocation problem are possible, thus the optimal one can be hard to determine without knowledge of the problem’s cause. A sub-optimal solution is not sufficient for all computing environments, RMS operating under real-time constraints while optimizing QoS attributes relies on optimal solutions rather than first-fit measures. Once the cause of a resource allocation problem has been determined, it can be used to limit the number of possible countermeasures to those who actually could help to solve the spe-

cific problem. The reduction to relevant solutions helps a QoS aware RMS to decide which service attributes to modify while still providing an optimal QoS level. Depending on the number of solutions known to the RMS, the reduction of alternative solutions also strengthens the real-time applicability. This performance improvement benefits the timing constraints in a real-time environment, requiring fewer solutions to be evaluated.

2 Requirements and Specifications

The resource management architecture QARMA, introduced in [3], consists of three major components: a System Repository Service (SRS), a Resource Management Service and an Enactor Service. The SRS stores both static and dynamic information concerning resource usage and software systems in the computing environment. Static information is known a priori and includes attributes like the number of available processing nodes or service replicas. Dynamic information is concerning the current state of the resources, often referred to as a snapshot, and includes monitor data measuring resource usage, resource availability, application performance and environmental conditions. To support the QARMA resource management service during the decision making, a QoS problem diagnosis component can be integrated in the functional structure of QARMA. This component analyzes resource allocation problems in order to determine their cause. The discovered knowledge enables QARMA to make more effective decisions, finding the optimal reactive measures faster and with increased accuracy. The knowledge of a problems cause helps to limit the number of possible countermeasures to those who actually fit the specific problem. The decreased set of possible solutions makes it easier for the decision-making component of QARMA to choose the optimal reaction and thus to provide longer-lasting solutions with a maximum level of QoS. A sub-optimal solution is not sufficient for all computing environments. Operating under real-time constraints and maximizing the QoS level provided by active tasks, QARMA relies on optimal countermeasures rather than first-fit measures. In addition QARMA can take benefit from the performance improvement, requiring fewer reactive measures to be evaluated for each problem. Whenever a detector discovers or predicts a violation to existing real-time constraints, it will invoke the QoS problem diagnosis component. Based on a snapshot of the current resource allocation, this module determines the exact cause of the problem. This information can be used by QARMAs resource management service to choose the optimal reactive measures for each problem. In general the QoS problem diagnosis component is independent of a specific technique used to determine the cause of a resource allocation problem. In this work a data mining module will be utilized by the diagnosis component to analyze the problem data.

3 A Data Mining Approach

To prove the aptitude of data mining to be used by the QoS problem diagnosis component, two according to [5] and [1] very common data mining techniques are applied to the resource allocation problems: the k-nearest neighbor technique and decision trees. Among the most common techniques, these two seem to be the most promising approaches to support the decision-making process of QARMA. They are not intended to

be the perfect choice out of all existing data mining techniques, but suitable to prove the aptitude of data mining for the given problem. The nearest neighbor technique makes a prediction based on distance between objects, in this case between snapshots of the training data and the one of the unclassified problem. Due to the identical layout of training data and the snapshot of the unclassified problems resource allocation and due to their restriction to numerical values only, a calculation of the distance is very simple. Each resource allocation can be seen as a point in a multi-dimensional vector space with one dimension for each resource attribute. While a graphical illustration of these points is impossible, the mathematical calculation of the distance between them is simple. It can be determined with the Euclidean distance metric shown in equation 1.

$$
\begin{aligned}
d_{ab} &= \sqrt{\sum_{i=1}^{N} |x_{ai} - x_{bi}|^2} \\
d_{ab} &: \text{distance between two points a and b} \\
N &: \text{Number of predictors}
\end{aligned}
\tag{1}
$$

For cause detection every resource attribute is equally important. Therefore it has to be prevented that attributes with large values outweigh ones with small values. This is done by normalizing the input values to lie in an equal interval for all attributes. Since all values in the system repository are positive, the interval [0,1] makes most sense. This normalization can be done with a linear transformation of all input values, the min-max normalization defined in equation 2.

$$
\begin{aligned}
x' &= x \cdot \left(\frac{1}{max_{in} - min_{in}} \right) \\
x &: \text{input value} \\
x' &: \text{output value} \\
min_{in} &: \text{lowest input value} \\
max_{in} &: \text{highest input value}
\end{aligned}
\tag{2}
$$

Without the normalization a 5 percent different amount of free memory would be rated much more important than the same difference for one of the load values. After the preprocessing of the input data is finished, the distance between each snapshot of the training data and the unclassified problem is calculated. Basing the prediction only on the one very closest match of the training data can case-dependent be too inaccurate. Thus the k closest records are taken into account for a prediction, usually k between 9 and 15 return much better predictions. But it is important that k matches the number of records in the training set and is not chosen too high. Otherwise the accuracy of the prediction will be reduced by too strong influences of records with other causes. Other than the k-nearest neighbor technique the decision tree prediction works with a prior created classifier model. Based on the training data, simple conditional rules are discovered. These rules separate groups of records with identical problem causes of the remaining group of unclassified records. The rule discovery algorithm tries to find attributes within the training data, which only have records with an identical cause above

or below a certain value for the attribute. A rule does not have to separate all records of a cause, the cause of the separated group might reoccur in the unclassified part of the training data or in other rules. The group of records classified by a rule is only restricted to a single cause. Searching for these rules, the algorithm tries to find the most significant ones, separating the biggest groups, first. The algorithm stops looking for further rules when only a single cause is left in the unclassified part of the training data. Once the rules are discovered, they can be used over and over again to predict the cause of an upcoming resource allocation problem. Given a suitable training set, decision trees return very accurate predictions. Since the predictions are made only based on the prior discovered rules, the performance of the prediction is rather independent of the amount of training records. The performance is only influenced by a limited number of rules. Thus decision trees are a very suitable approach even under the timing constraints of a real-time environment. The implementation of the decision tree technique consists of two different components. The first one searches the database for rules separating records of the database in groups with an identical cause. The second component predicts the cause of an unclassified problem by applying the resource usage snapshot to these rules. The big advantage of this separation with respect to performance issues is the fact, that the computation intensive rule discovery only has to be done once and that the discovered rules can be used over and over again by the prediction component. This provides a major performance improvement compared to techniques based on the complete training data.

4 Experimental Results

To prove the aptitude of the developed data mining module to be used in the QoS problem diagnosis component, and thus the aptitude of data mining in general, two different experiments were performed. The first experiment was set up to analyze the accuracy of the two implemented data mining techniques. It was used to determine the aptitude of the techniques to be used for problem diagnosis independent of their required execution time. The second experiment was set up to analyze the performance of the predictions in dependence of the amount of training data. The results of this experiment were used to determine how suitable the two data mining techniques are for a real-time environment. The dependence of the size of the training set is important, because the accuracy of the predictions made depends on the size of the training set. All experiments were repeated several times on a machine with a Pentium IV 2.4GHz CPU and 512MB RAM running under Knoppix Linux 3.3 or Knoppix Linux 3.4 respectively. Prior to both experiments the training data was created by deliberately causing resource allocation problems in QARMA and collecting the affiliated resource usage information. For each cause multiple SRS snapshots and their causes were stored. The biggest part of these snapshots was used as training data, providing the data mining techniques with resource usage information of classified problems. The remaining records were provided to the data mining techniques as unclassified problems without their known cause. To be correct, a predicted cause had to be equal to the real cause of these snapshots. The experiments regarding the accuracy of the predictions made by the two implemented data mining techniques were performed for the biggest part with small databases. Later the database

size was increased to capture possible dependencies between the accuracy of a prediction and the size of the training set. The minimum database size was 50 records; the maximum size was 500 records. The experiments were repeated multiple times using Knoppix 3.3 and Knoppix 3.4 as operating environments. Other than the results of the performance experiments, the ones concerning the accuracy of the chosen techniques were identical for both operating systems as expected. Thus neither average values are presented nor can a comparison of the predictions made under different operating systems be made. To evaluate the predictions made by the two data mining techniques, the predicted cause of each applied resource allocation snapshot was compared to the real cause the snapshot was created with. A prediction was deemed accurate, if it was either equal to the known cause or in case of the k-nearest neighbor technique was at least returned with a significant higher probability than all other causes. The performance experiments were set up with database sizes between 50 and 100.000 records. While the smaller databases contained each prior created training record only once, the bigger databases consisted of multiple copies of these records. This was easier than to manually create 100.000 problem records and still is a suitable setup since the performance depends on the database size only. The performance is determined by the number of records to be analyzed by the data mining algorithms, but not by reoccurrences in their content. The experiments were performed several times under Knoppix Linux 3.3 and Knoppix Linux 3.4. To determine the performance of an implemented data mining technique, the execution time of a prediction was measured using the Linux command time. For each technique, each database size and each operating system the experiments were repeated several times. After collecting the execution times for each combination of the three factors above, the average execution time of these combinations was calculated to determine the performance. The k-nearest neighbor prediction was tested with k={1,3,5,8,10,15} to determine if the size of k also influences the performance. The values for k were chosen to range from very low values to those offering the highest accuracy. With one exception the execution time needed to predict the cause of an unclassified resource allocation problem turned out to be independent of the chosen k. When executed under Knoppix 3.4 a prediction with k=1 performed significantly slower than all other k. Since the accuracy of a prediction with k=1 is lower than one with a higher k, this result has not been further investigated. For all other k the execution times are very similar among those with identical database size and operating system. The independence of the value k has a simple explanation. The most time consuming part of a prediction is the distance measurement between the resource usage of each database record and the unclassified problem. These calculations require almost the complete execution time. The time needed for the comparisons of the calculated distances to the temporary optima is insignificantly small. Thus a real difference in execution time for different k can not be measured. Figure 1 shows the performance graphs of the k-nearest neighbor prediction for execution under Knoppix 3.4. The execution times for the different k are with exception for k=1 so similar that a visual difference between the different k can not be determined. The execution times have a linear relation to the size of the database the k-nearest neighbor prediction is based on. Thus the time complexity of this approach is O(n)=n. Due to the results of the accuracy experiments and the linear time complexity the k-nearest neighbor prediction is a very suitable technique for

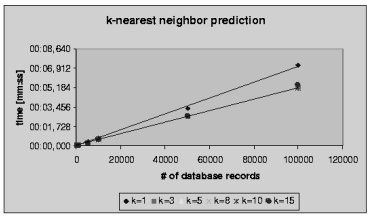

Fig. 1. Performance graph of the k-nearest neighbor prediction (Knoppix 3.4)

resource management environments operating without timing constraints. The fact that they operate without a prior created classifier model has the advantage that the training set needs less preparation while still making accurate predictions. The cause of resource allocation problems that do not perfectly fit the training data can be determined in the diagnosis nevertheless. But depending on the number of possible problems in the environment the resource management system is used in, the database needs to be rather large to store enough training records to make accurate predictions possible. Resulting in longer execution times, this makes the k-nearest neighbor prediction only in connection with small-scale databases suitable for an application in a real-time environment. The multiple-second execution time needed in a 100.000 record database is inappropriate for such an environment. The experiments measuring the decision tree performance have been executed in two steps. First a classifier model has been created based on the prior created training data. After the model had been build, the discovered rules have been used by the decision tree prediction component to determine the cause of the unclassified problem. Surprisingly the execution times of the rule discovery process under Knoppix 3.3 and Knoppix 3.4 turned out to be quite different. Especially under Knoppix 3.3 the creation of the classifier model was very slow, thus only up to 10.000 database records were used for the performance experiments under this operating system. The rule discovery process performed significantly better under Knoppix 3.4. The cause for these performance differences is expected to be the improved mySQL version provided by Knoppix 3.4. But since the performance of the rule discovery process is less important for a feasible usage of the decision tree technique in the QoS problem diagnosis component, this result has not been further investigated. Figures 2 and 3 show the performance graphs of the decision tree rule discovery process for Knoppix 3.3 and Knoppix 3.4 respectively. Other than the performance of data analysis done during the k-nearest neighbor prediction, the performance of this component differs significantly between the two operating systems. Under Knoppix 3.3 the rule discovery process has a clear polynomial increase in the execution time with respect to the database size. The creation of the decision tree classifier model has time complexity O(n)=n^2. For large databases the performance of the execution time needed for the rule discovery process under Knoppix 3.4 is in almost linear relation to the size of the database. Only for small databases with less than 1.000 records the relation turned out to be polynomial. Due to the range of records used for the experiments figure 3 only can visualize the linearity of this relation. Even though the time complexity for smaller training sets is quadratic, it almost reaches O(n)=n for larger scaled databases. The execution time of the decision tree prediction takes constant 2 milliseconds and is independent of the database size. The time complexity of this component's execution is O(n)=1 and thus performing very

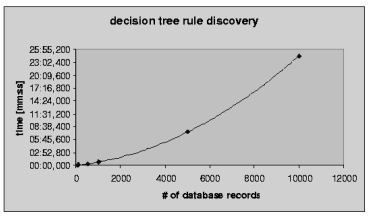

Fig. 2. Performance graph of the decision tree rule discovery (Knoppix 3.3)

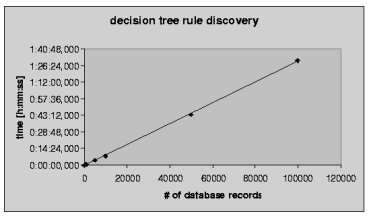

Fig. 3. Performance graph of the decision tree rule discovery (Knoppix 3.4)

well. The performance experiments have proven decision trees to be a very suitable technique to be used by the QoS problem diagnosis component to predict the cause of an resource allocation problem. The polynomial time complexity of the rule discovery process is no drawback for the aptitude of this technique. This step is required only once and not suspect to any timing constraints. Thus the long execution times of this component are justified, if the created classifier model can be used to make accurate predictions. Although the accuracy experiments have proven that either carefully prepared or extensive training sets are required to create an accurate classifier model, this does not contradict with the aptitude of the prediction to be used even in a real-time environment. The constantly fast execution time makes predictions with this technique feasible even for very timing sensitive environments. Thanks to the execution time being independent of the database size, the training set can be chosen large enough for an accurate model creation. Both data mining techniques implemented and analyzed in these experiments have proven to be suitable and accurate approaches to diagnose resource allocation problems. The k-nearest neighbor technique is not based on a prior created classifier model. This has the advantage that this technique is less demanding with respect to the provided training set. This approach makes accurate predictions even with a less careful prepared training set possible. On the other hand the linear complexity of the execution time restricts this technique to either a limited training set size or an environment with weaker timing constraints. If only a very limited number of problems can occur in the environment, the database can be kept small enough to comply with real-time requirements. For timing sensitive environments with numerous possible problem causes, decision trees are a better approach. The decision tree prediction is based on a prior created classifier model. This has the advantage that the time needed for a prediction is independent of the size of the training set and thus performing significantly better that a technique taking the whole training set into account for a prediction. Due to the constant time complexity of the prediction, this approach is feasible even under strong timing constraints.

5 Conclusion

This work introduced a QoS problem diagnosis component for Ohio University's real-time resource management system QARMA. For this problem diagnosis component a data mining module has been developed to determine a resource allocation problem's cause and to prove the aptitude of business informatics techniques to successfully sup-

port resource management. The performed experiments could verify data mining's aptitude in general and the single techniques' aptitude in specific. But the experimental results have also been able to show the techniques' limitations and their requirements to be able to make accurate predictions. While the k-nearest neighbor technique is a good choice for RMS operating without timing constraints, it is not the best choice for problem diagnosis in a real-time environment due to performance issues. Decision trees on the other hand are performing well enough to be applied to resource allocation problems in a resource management environment with timing constraints. But due to its tendency to over-fit the training data, this technique turned out to be very demanding to the training set. A rather large training set has to be carefully prepared to enable the decision tree rule discovery process to create an accurate classifier model. But with suitable training data, decision trees turned out to be a very fast and accurate prediction technique. The proposed QoS problem diagnosis component is expected to optimize the choice of reactive measures to resource allocation problems. Knowledge of a problem's cause can limit the number of possible reactions in general and further restrict their number to good or optimal choices in specific. This reduces the time needed to find the optimal reaction to each problem, which can not only solve the resource allocation problem, but also provides the highest QoS level possible. Thus the problem diagnosis component is expected to help to optimize the QoS level at which a real-time RMS can operate and to support adjustments to QoS settings during runtime. But the expected possibilities for QoS optimization are not only of importance for resource management alone. Especially business application systems that work under timing constraints and allow QoS adjustments to their tasks can take great benefit of the QoS optimization. For example video solutions like VoD systems or video conferencing solutions would be able to constantly provide the highest lag-free QoS level possible at each moment, while still being able to satisfy all clients. This QoS optimization would be an important cornerstone to the success of such a business application system. In a B2C environment this optimization could be a crucial part of a VoD system needed to gain a competitive advantage over other similar services. Thus the problem diagnosis is not only expected to help to constantly optimize QoS settings during runtime, but could also be of significant influence on the economic success of commercial services based on a QoS aware RMS like QARMA.

References

1. BERSON, A., SMITH, S. and THEARLING, K. (1999): *Building Data Mining Applications for CRM*. New York, NY, U.S.A.
2. BESTAVROS, A. (1996): Middleware Support for Data Mining and Knowledge Discovery in Large-scale Distributed Information Systems. Proceedings of ACM SIGMOD'96 Data Mining Workshop
3. FLEEMAN, D. and WELCH, L. et al. (w/o year): Quality-based Adaptive Resource Management Architecture (QARMA): A CORBA Resource Management Service. Ohio University, Athens, OH, U.S.A.
4. IMBERMAN, S. (2002): The KDD process and data mining for performance professionals. *Journal of Computer Resource Management, issue 107, p. 68-77*
5. KDNUGGETS (2003): Poll: Data Mining techniques http://www.kdnuggets.com/polls/2003/data_mining_techniques.htm 08/21/04