Modeling the Behavior of Large Scale Reasoning Systems Using Clustering and Regression

Raluca Brehar, Ion Giosan, Andrei Vatavu, Mihai Negru, Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca
E-mail: firstname.lastname@cs.utcluj.ro

***Abstract*—Modeling the performance of large scale systems is the core idea of this paper. We focus on modeling the performance specific behavior of LarKC [1]- The Large Knowledge Collider a platform for large scale integrated reasoning and Web-search. A set of instrumentation and monitoring tools are employed to collect metrics related to execution time, resources, and specific platform measurements like running workflows and plug-ins. Our method performs machine learning on top of instrumented data and tries to find relations between input defined metrics and output metrics that describe the instrumentation observations of the LarKC platform, plug-ins or workflows. The proposed method is a combination of clustering and regression techniques.**

I. Introduction

LarKC is developing a platform that enables the development of large-scale reasoning applications using and combining techniques from various Semantic Web related research fields. The pluggable architecture of LarKC enables the interested LarKC users to test their ideas for doing reasoning. It allows them to integrate and deploy their own components, known as plug-ins in LarKC terminology, in the platform, to flexibly connect them in order to build workflows, to run and to test them.

The instrumentation and monitoring of such a system is extremely important. LarKC platform comes with a collection of tools that allow users get real-time data about the resources, execution time and other specific behavior of the platform, plug-ins and workflows. It offers the means for developers to specify the metrics of interest, to instrument the code, to collect and observe how well the system and its components are performing. For example, LarKC developers can find out how are their plugins and workflows, how many resources they use, how much data they consume and produce, etc.

The instrumented data is further used by a novel component described in this paper. It performs machine learning on top of the data and it can be used to model the behavior of LarKC platform, plug-ins and workflows given a certain input workload to the platform.

Modeling behavior of systems like LarKC is a complex task because of the high variability in workload and input pattern configurations, the available resources and system configuration vary among different machines on which the platform is run, the different components interacting with the platform should not have conflicting hardware/software or network requirements, massive parallelization and workload distribution.

The instrumentation measurements try to capture various aspects of the platform, plug-ins and workflow behavior, that is we have generated workloads for which the system provides a successful execution, workloads for which the system fails (because of lack of resources or because of network problems), workloads for which the execution takes long time, but we still get a result. Using all this collected data we try to model the behavior of the system on new, unknown workload configuration that are susceptible to a set of predefined constraints.

The structure of the document is as follows: section II presents the state of the art methods in modeling system performance, section III defines the mathematical model employed by our method, section IV provides details about the clustering and regression methods we have used, section V details our experimental methodology and finally section VI concludes the paper and provides directions for future work.

II. Related Work

Modeling system performance is a new and active field of research and it can be applied to numerous system models: distributed systems, networking, database systems, parallel I/O systems etc.

Statistical machine learning techniques, namely clustering, regression, principal component analysis and kernel canonical correlation analysis are employed by [1] and [2] to optimize multicore performance or to accurately predict the performance metrics of database queries in vary large data warehouse.

Frequent pattern mining and principal component analysis are also used by [3] for detecting large-scale system problems. They use information retrieval to transform free-text console logs into numerical features and based on these features identify the operational problems.

Neural networks, support vector machines and decision tree-based regression models have been experimented by [4] in order to asses the performance of TOB protocols (Total Order Broadcast - is a main building block for developing strongly consistent replicated systems).

Neural networks have been experimented in other types of systems. For example [5] apply neural networks and Bayesian networks for modeling the response time of service-oriented

[1] http://www.larkc.eu

computing facilities in the construction of dynamic, complex distributed systems. A radial basis function neural network has been applied by [6] for modeling the performance of a parallel I/O system with experiments on IBM SP.

Techniques based on a Instance Based Learning algorithm and several improvements are proposed and empirically evaluated by [7] for workload modeling and performance prediction in space-shared, data-intensive Grid environments.

TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) machine learning algorithm evaluated on KDD Cup 1999 dataset by [8] is used for anomaly detection in network.

Predictive models used to identify parts of a Java system with a high fault probability are proposed by [9] that assess three aspects on how to build and evaluate fault-proneness models in the context of the large Java legacy system development project.

III. Method Description

Our method models the behavior of a large scale reasoning systems - namely LarKC[2]. LarKC is developing a platform that enables the development of large-scale reasoning applications using and combining techniques from various Semantic Web related research fields. The pluggable architecture of LarKC enables the interested LarKC users to test their ideas for doing reasoning. It allows them to integrate and deploy their own components, known as plug-ins in LarKC terminology, in the platform, to flexibly connect them in order to build workflows, to run and to test them.

A. General context

A general scenario in which LarKC can be used is shown in Figure 1. In this situation each user may "ask" one or more queries that are send serially to the platform. Each query is "solved" by a workflow that is formed of several plug-ins that are running serially or in a remote manner. Even if LarKC supports more interaction patterns between users and the platform, we will use the previous general scenario: query – workflow (list of plugins) – response.

Queries have the form of SPARQL [3] queries.

B. Metrics

LarKC comes with a set of instrumentation tools that measure the parameters of the system at runtime. We call these parameters metrics. They can be grouped with respect to the type of entity for which they are generated. For example we may have:

- query metrics – provide information from the SPARQL query or from the execution of the query; The information extracted from SPARQL can be: QuerySizeInTriples, QueryNamespacesNb, QueryNamespace, QueryVariablesNb, QueryDataSetSourcesNb, QueryResultOrderingNb, QueryResultLimitNb, QueryResultOffsetNb. Other metrics related to the query include: QueryTotalResponseTime, QueryCompletionStatus, QueryBlockedTime.
- workflow metrics – offer information about the behavior of the workflow at runtime. They include WorkflowPluginsNb, WorkflowDuration.
- plug-in metrics – supply data about the plug-ins' behavior at runtime. They comprise: PluginInputSizeInTriples, PluginOutputSizeInTriples, PluginInputSizeInBytes, PluginOutputSizeInBytes, PluginDataLayerAccess, PluginBlockedTime.
- platform metrics – describe the performace of the platform at runtime. They include: PlatformCPULoad, PlatformMemoryUsage, PlatformMemoryDim, PlatformGarbageCollectingTime.

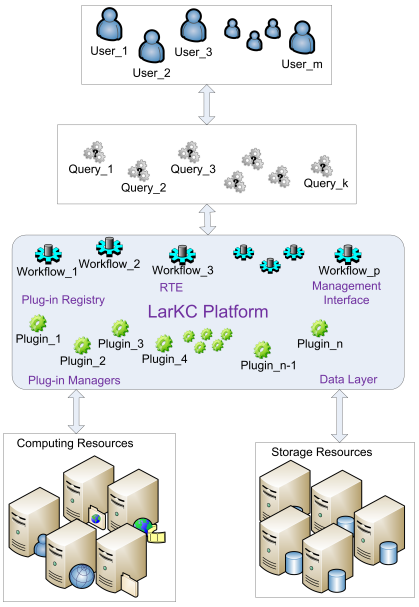

Fig. 1. LarKC - use case for modeling the system behavior

C. System model

We consider that the LarKC processing system has as input a workload that can be a query or a workflow as depicted in Figure 2. The query is in the SPARQL format and it should be a valid query. The input workflow can be given as a combination of plug-ins that the user wants to utilize in order to solve a certain computation task.

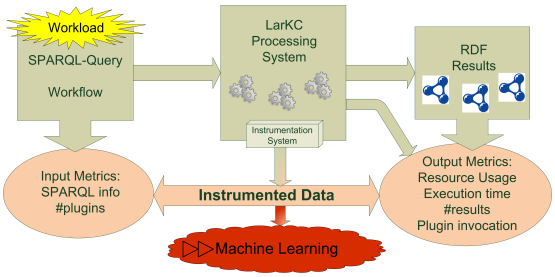

Fig. 2. System model

[2]http://www.larkc.eu
[3]http://www.w3.org/TR/rdf-sparql-query

Using instrumentation we may extract:

- Input metrics that quantify the characteristics of the workload (comprise query related metrics and workflow related metrics or plug-in related metrics if the input workload contains the parameters of the workflow).
- Output metrics: capture the behavior of the LarKC processing system (performance,resources) during the execution of tasks required by the input workload.

Our purpose is to use the recorded historical behavior of the system (given by instrumented data) to develop a predictive model (LarKC Processing Relevance Feedback Model) that finds relations between input metrics and output metrics. Figure 3 presents the idea behind the predictive model or relevance feedback model. Our method is applicable for several use-case scenarios like:

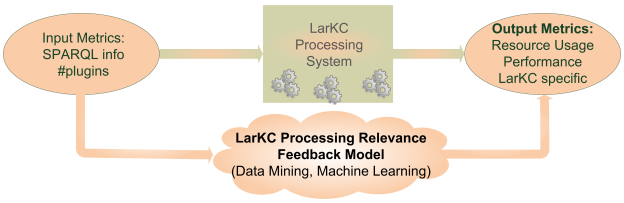

Fig. 3. Predictive Model

- Execution status analysis captured by the success or failure rate of the system. Using the relevance feedback model one may find the relation between input workload parameters and success/failure rate of the system. Success shows that the execution corresponding to a certain workload was completed by the system successfully while failure marks that the execution corresponding to a workload has not been accomplished because a component of the system crashed.
- Scalability analysis:
  - Given a workload find the relationship between the metrics that quantify the workload and the resources needed by the system to handle the workload.
  - Given a constraint on resources/performance find the input workload that can be handled by the system.

D. Mathematical model

In order to find the relation between input metrics and output metrics we deploy a mathematical model containing clustering and regression. Consider the following notations:

- The set of input metrics: $I = I_1, ..., I_n$
- The set of output metrics: $O = O_1, ..., O_m$
- The measurements/observations obtained from instrumentation and monitoring module are a matrix with p instances:

$$\begin{pmatrix} I_{11} & I_{12} & ... & I_{1n} & O_{11} & O_{12} & ... & O_{1m} \\ I_{21} & I_{22} & ... & I_{2n} & O_{21} & O_{22} & ... & O_{2m} \\ ... & ... & ... & ... & ... & ... & ... & ... \\ I_{p1} & I_{p2} & ... & I_{pn} & O_{p1} & O_{p2} & ... & O_{pm} \end{pmatrix}$$

Another notation:

$$\begin{pmatrix} Inst_{11} & Inst_{12} & ... & Inst_{1u} \\ Inst_{21} & Inst_{22} & ... & Inst_{2u} \\ ... & ... & ... & ... \\ Inst_{p1} & Inst_{p2} & ... & Inst_{pu} \end{pmatrix}, u = m + n$$

Generally, the relevance feedback model that we propose can be described as a function that takes a set of inputs and tries to define an output value:

$$O_j = f_j(I), j = \overline{1,m}$$

In the above formula, the function f_j can be: a)

1) a clustering rule followed by a regression model which return a numerical value
2) a clustering rule and a numbering (ranking) to:
   b1) return a probability
   b2) return a list of nominal values

The clustering process takes the set of instances and forms q groups (clusters):

$$Cluster(Inst, K) \Rightarrow C_1, ..., C_q$$

where:

1) q is the total number of clusters
2) K is the set of clustering criteria:

$$K = \{K_i | K_i \in I, K_i \ is \ a \ clustering \ criterion\} \Rightarrow K \subseteq I \quad (1)$$

The cluster number p on criteria K is:

$$C_p^K = \{Inst_i\} \text{ where } Inst_i \in Inst, \forall Inst_j \in C_p^K, \text{similar}_K(Inst_i, Inst_j) \bigcup_{j=1}^{q} C_j^K = Inst, \bigcap_{j=1}^{q} C_j^K = \phi \quad (2)$$

$C^K = \{C_p^K | p = \overline{1,q}\}$, where q is the cardinality of C^K

Given a new workload Q characterized by a set of input parameters (metrics):

$$I_W = (I_{W1}, I_{W2}, ..., I_{Wn})$$

The objective is to find the cluster where the new workload belongs to:

$$C_b^K = C_j^K | similar_K(I_W, I_i), \forall I_i \in C_j^K, \text{ where } j = \overline{1,q}, \ q = |C^K|$$

The way that the prediction is applied in all previously mentioned cases is described in the next paragraphs.

Case a)

Find a regression function which returns a numeric value. It will be used in prediction of a numeric value of an output metric O_p:

$$O_p \in O, p \in \{1, 2, ..., m\}$$

$$f_p : I \to \Re$$

$$O_p = f_p(I)$$

The function f_p is obtained by a regression method applied on the instances from the selected cluster C^K:

$$f_p(I) = \sum_{i=1}^{n} a_i I_i + a_0$$

The predicted value of O_p for the given input I_w is:

$$O_{pW} = f_p(I_W)$$

Case b1)

Find a probability function which returns a real value between 0 and 1. It is used in prediction of a nominal output metric O_p.The values of O_p are from the set $\{n_1, ..., n_s\}$.

Define s probability functions $f_{p1}, ..., f_{ps}$ which output the probability values for $\{n_1, ..., n_s\}$.

Inside the cluster C_b^K perform a selection on fixed nominal values $\{n_1, ..., n_s\}$. The result is a set of selection clusters:

$$C_{b1}^{K1} = Select(C_b^K, K1), K1 = \{n_1\}$$

$$...$$

$$C_{bs}^{Ks} = Select(C_b^K, Ks), Ks = \{n_s\}$$

$$C_{bt}^{Kt} = \{Inst_i\} \quad \text{having} \quad Inst_i \in C_b^K, \forall Inst_j \in C_b^K, exact_{Kt}(Inst_i, Inst_j) \quad t = \overline{1,s} \tag{3}$$

The appearance probability of the nominal value n_t for input parameters I_W is n_{tW}

$$f_{pt}(I) = \frac{|C_{bt}^{Kt}|}{|C_b^K|}$$

$$n_{tW} = f_{pt}(Inst_W) = \frac{|C_{bt}^{Kt}|}{|C_b^K|}$$

Case b2)

This case is used for getting the best configuration (nominal attribute) that maximizes the values from a set of constraints applied on the input metrics. The constraints set $Z \subseteq I$ (cost functions, performance metrics, etc.) is:

$$Z = \{Z_1, ..., Z_c\}$$

Consider the previous case b1) and the possible values for that nominal attribute $n_1, ..., n_s$. The selection clusters are known: $C_{bt}^{Kt}, t = \overline{1,s}$.

Compute the mean performance for each input configuration for each cluster:

$$\mu_t = avg_Z(C_{bt}^{Kt}), t = \overline{1,s}$$

$$\mu_t = (\mu_{t1}, ..., \mu_{tc})$$

$$Kt = \{n_t\}$$

Sort descending the selection clusters $C_{b1}, ..., C_{bs}$ by considering the mean performance $\mu_t, t = \overline{1,s}$ as sorting criteria $\Rightarrow$ the sorted result will be the set of selection clusters $S_{b1}, ..., S_{bs}$. The output consist in the ordered list of nominal values from $S_{b1}, ..., S_{bs}$:

$$n(S_{b1}), ..., n(S_{bs})$$

which represent the ordered list of nominal values from the best one to the worst one.

IV. Theoretical Background - Clustering and Regression

A. Clustering

In our mathematical model we have used clustering for grouping similar instances. In our implementation we use Expectation Maximization (EM) Clustering method.

EM is a mixture based algorithm that attempts to maximize the likelihood of the model. It makes the assumption that attributes are independent random variables. An instance is characterized by a set of attributes. EM is performed in two steps:

1) expectation: calculation of the cluster probabilities (which represent the "expected" class values)
2) maximization: computation of the distribution parameters that is the maximization of the likelihood of the distributions given the data.

EM can decide how many clusters to create by cross validation. For this prototype, the cross validation performed to determine the number of clusters is done in the following steps [10]:

1) the number of clusters is set to 1
2) the training set is split randomly into 10 folds.
3) EM is performed 10 times using a certain number of folds folds .
4) the log-likelihood is averaged over all 10 results.
5) if log-likelihood has increased the number of clusters is increased by 1 and the program continues at step 2.

B. Regression

Within each cluster a regression model is build.

The regression model is used to predict the result of an unknown dependent variable, given the values of the independent variables. The estimation target is a function of the independent variables called regression function. In our prediction problem we use a linear regression model. The Akaike criterion determines the model selection by choosing from a set of candidates (e.g. total least squares, generalized least squares, adaptive estimation, principal component regression, ridge regression) that one with the minimum AIC (Akaike Information Criterion) value, and is able to deal with weighted instances. Each instance is represented by a vector of input metrics (attributes) values "x" and a value "y" of the output metric (class). The linear regression model determines a linear equation of the form $y = ax + b$ for finding the relation between the values from input vector "x" and the corresponding value for output "y". The parameters that are computed by

the regression model refer to the vector "a" which is used for weighting the input attributes values and the constant value "b" that is added to the weighted sum. An attribute selection using M5's[11] method that steps through all the attributes and removes the one with the smallest coefficient until no improvement is observed in the estimate of the error given by the AIC and a greedy selection using the Akaike information metric are also used.

V. Experiments and results

A. Model training

In order to build our relevance feedback model we have instrumented several LarKC workflows. We have collected data for the generated queries and we have trained our combined clustering and regression model.

The data used in the predictor's training is loaded from a specific ARFF file. An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances corresponding to a set of attributes. In the following we describe an experiment regarding training and prediction for a scalability scenario.

The attributes considered are divided in two sections: input metrics and output metrics, as described in the mathematical model section.

The input metrics are: *QueryTimestamp, QueryContent, QueryNamespaceNb, QueryNamespaceKeys, QueryNamespaceValues, QueryVariablesNb, QueryDataSetSourcesNb, QueryResultOrderingNb, QueryResultLimitNb, QueryResultOffsetNb, QuerySizeInTriples*.

The output metrics (the predicted metrics) are: *WorkflowDurationFromPlatform, QueryTotalResponseTimeFromClient, QueryCompletionStatus, DeciderTotalExecutionTime, TransformerTotalExecutionTime, IdentifierTotalExecutionTime, SelecterTotalExecutionTime, ReasonerTotalExecutionTime, WorkflowPluginNb, DeciderThreadsStartedNB, WorkflowDuration*.

Each metric has a specific type: Date (e.g. *QueryTimestamp*), String (e.g. *QueryNamespaceValues*), Numeric (e.g. *WorkflowDuration*) or nominal values (e.g. *QueryCompletionStatus* with values in the set SuccessfulQuery, FailedQuery).

In the training phase, the instances' values for both input and output metrics are known by analyzing the input query and by instrumenting the platform. The first step consist in preprocessing the ARFF file and generate unique attributes for namespaces. The values that represent compound inputs (e.g. query namespaces) are tokenized using a string-to-word tokenizer, resulting a set of atomic entities. A preprocessing function that converts nominal values to numeric indexes is also applied on the data. The second step is the clustering based only on the query's attributes (already numeric values). After this step the result is a set of clusters. The third step comes and computes a linear regression model inside each formed cluster.

For the given data four clusters have been formed. Within each cluster the regression model has been build.

B. Model evaluation and application

For the prediction phase, a query is given and we try to predict the output metrics values. First we convert the query in ARFF format, then the namespaces are tokenized in atomic values and nominal values are converted in numeric indexes. The nearest cluster to the given query is found. Last step consist in selecting the regression model or the probabilistic model which is already computed for that nearest cluster. The result is returned by applying that model with the input values from the given query.

The models have been evaluated on several test data. In the Figure 4, 5 and 6 are presented some prediction results on a set of few given queries and corresponding workflows. The errors between the predicted values and the real values obtained by executing those given queries are subjectively low. The results are promising and they can be improved by training on a larger amount of instrumentation data.

For example the predicted values for the metric "QueryTotalResponseTimeFromClient" are given in Figure 4. The values represents the time elapsed (in milliseconds) for a query to complete its execution at the client side. The prediction is realized by applying the already computed linear regression model of the nearest cluster where the given query belongs to. There are some cases where the predicted value is below 0 due to the fact that the result is a pure linear combination of some input attributes' values. In these situations the predicted value can be considered equal to 0.

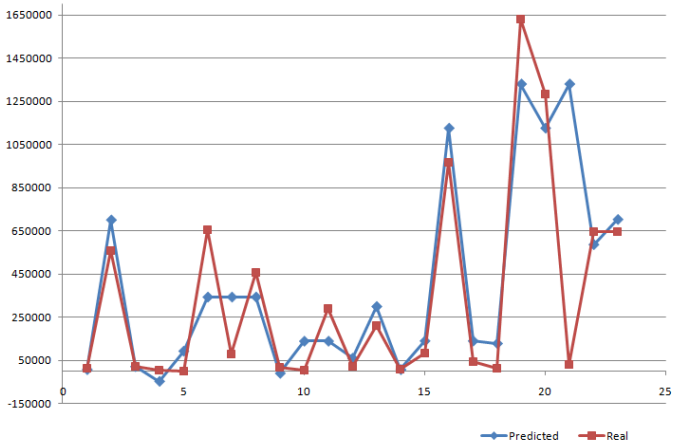

Fig. 4. Predicted outputs vs. true values for the "QueryTotalResponseTimeFromClient" metric.

The predicted values for the metric "WorkflowDurationFromPlatform" are given in Figure 5. The values represents the platform workflow duration (in milliseconds) for completing the execution of the given query. This case is similar to the previous one, having the same remarks. Overall, the predicted results in these two cases are good.

The predicted values for the metric "QueryCompletionStatus" are given in Figure 6. The real values for this metric are from the set {FailedQuery, SuccessfulQuery}. These two

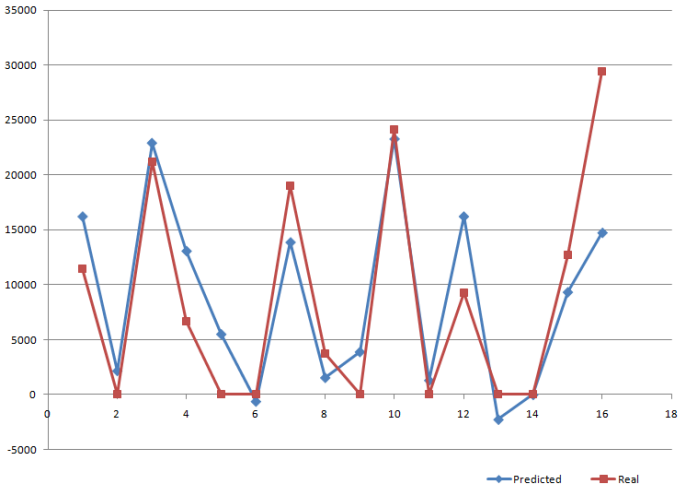

Fig. 5. Predicted outputs vs. true values for the "WorkflowDurationFromPlatform" metric.

nominal values correspond to the set of two probabilities {0–FailedQuery, 1–SuccessfulQuery}. Due to the fact that the prediction model finds the success completion probability of the given query, the results are values between 0 and 1 (in a fuzzy manner).

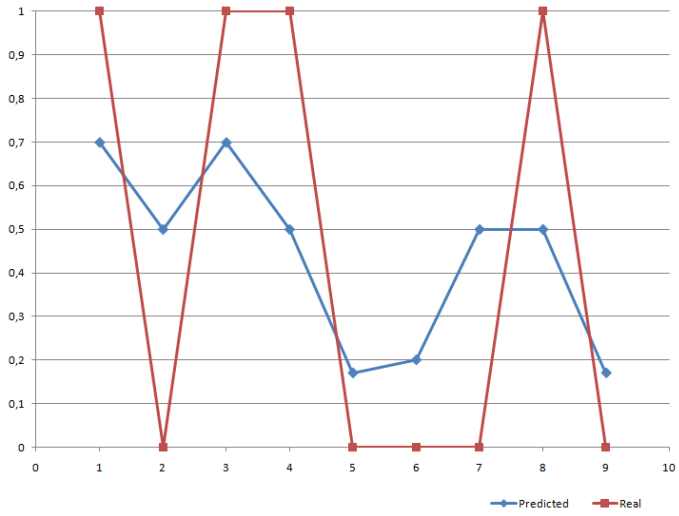

Fig. 6. Predicted outputs vs. true values for the "QueryCompletionStatus" metric.

The model we have trained can be applied on new workload parameters. For example, the user can give a query and chooose a scenario for which the prediction can be made. This is done via a visualization component responsible for sending input metrics to the relevance feedback module and displaying the results provided by the relevance feedback. The Visualization Component is a real-time web application that allows an easier interpretation of the historical data collected during instrumentation and monitoring. It includes the following main components:

- Client-Side Component is the front end part that runs in the end user's web browser as a rich content AJAX application. For the client-side component implementation we choose Google Web Toolkit (GWT). By using GWT the front end application is written entirely in Java and deployed as a highly optimized JavaScript that runs across all browsers.
- Server-Side Component includes a controller of business logic which coordinates requests from clients, as well as the data layer queries and responses. Based on client requests and data base (data layer) query results, actions are carried out by the server.

Figure 7 offers a screen shot of the on-line demo.

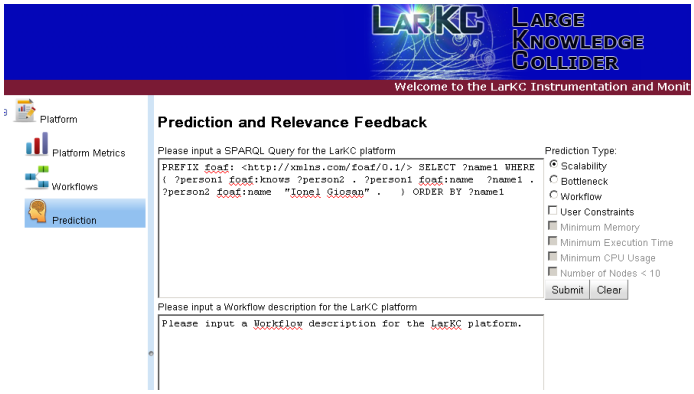

Fig. 7. Relevance feedback integrated in the visualization component

The visualization interface offers us the possibility to interactively extract and identify important patterns in a large amount of rough data about.

- LarKC platform in general.
- Queries that were passed to the LarKC platform.
- Workflows used to solve the queries.
- Plug-ins that compose the workflows.

Furthermore, visualization results allow users to discover the potential anomalies and make appropriate decisions.

VI. Conclusions and Future Work

The paper has presented a machine learning based approach for modeling the behavior of LarKC - a platform for distributed large scale reasoning and Web-search. The novelty of the approach resides in the application of clustering and regression on instrumentation specific data collected for the LarKC platform and its satellite components: the plug-ins and the workflows. The obtained results are promising and the efficiency of the method can be improved by increasing the number of instrumented workflows in the LarKC platform in order to generate data with a high variability, by applying other machine learning techniques (like multi-variate dataset analysis), or by testing out feature selection and outlier detection methods before the actual prediction is made.

VII. Acknowledgment

This work has been founded by the international research project LarKC[4].

References

[1] A. Ganapathi, K. Datta, A. Fox, and D. Patterson, "A case for machine learning to optimize multicore performance," in *Proceedings of the First USENIX conference on Hot topics in parallelism*, ser. HotPar'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 1–1. [Online]. Available: http://portal.acm.org/citation.cfm?id=1855591.1855592

[2] A. Ganapathi, H. Kuno, U. Dayal, J. L. Wiener, A. Fox, M. Jordan, and D. Patterson, "Predicting multiple metrics for queries: Better decisions enabled by machine learning," in *Proceedings of the 2009 IEEE International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 592–603. [Online]. Available: http://portal.acm.org/citation.cfm?id=1546683.1547490

[3] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 117–132. [Online]. Available: http://doi.acm.org/10.1145/1629575.1629587

[4] M. Couceiro, P. Romano, and L. Rodrigues, "A machine learning approach to performance prediction of total order broadcast protocols," in *Proceedings of the 2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, ser. SASO '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 184–193. [Online]. Available: http://dx.doi.org/10.1109/SASO.2010.41

[5] R. Zhang and A. J. Bivens, "Comparing the use of bayesian networks and neural networks in response time modeling for service-oriented systems," in *Proceedings of the 2007 workshop on Service-oriented computing performance: aspects, issues, and approaches*, ser. SOCP '07. New York, NY, USA: ACM, 2007, pp. 67–74. [Online]. Available: http://doi.acm.org/10.1145/1272457.1272467

[6] S. Yu, M. Winslett, J. Lee, and X. Ma, "Automatic and portable performance modeling for parallel i/o: a machine-learning approach," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, pp. 3–5, December 2002. [Online]. Available: http://doi.acm.org/10.1145/605521.605524

[7] H. Li, "Performance evaluation in grid computing: A modeling and prediction perspective," in *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, ser. CCGRID '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 869–874. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2007.84

[8] Y. Li, B. Fang, L. Guo, and Y. Chen, "Network anomaly detection based on tcm-knn algorithm," in *Proceedings of the 2nd ACM symposium on Information, computer and communications security*, ser. ASIACCS '07. New York, NY, USA: ACM, 2007, pp. 13–19. [Online]. Available: http://doi.acm.org/10.1145/1229285.1229292

[9] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A systematic and comprehensive investigation of methods to build and evaluate fault prediction models," *J. Syst. Softw.*, vol. 83, pp. 2–17, January 2010. [Online]. Available: http://portal.acm.org/citation.cfm?id=1663656.1663909

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10–18, 2009.

[11] Y. Wang and I. H. Witten, "Induction of model trees for predicting continuous classes," in *Poster papers of the 9th European Conference on Machine Learning*. Springer, 1997.

[4] http://www.larkc.eu