Evaluation of Adaptive Computing Concepts for Classical ERP Systems and Enterprise Services

Martin Wimmer, Valentin Nicolescu, Daniel Gmach, Matthias Mohr, Alfons Kemper and Helmut Krcmar
Technische Universität München,
85748 Garching b. München, Germany,
Email: {wimmerma | nicolesc | gmach | mohrm | kemper | krcmar}@in.tum.de

***Abstract*— To ensure the operability and reliability of large scale Enterprise Resource Planning Systems (ERP), a peak-load oriented hardware sizing is often used. Better utilization can be achieved by employing an adaptive infrastructure based on smaller computational units in combination with an intelligent allocation management. The SAP University Competence Center (German SAP HCC) at the Technische Universität München provides support for 55 ERP training systems. The evaluation of the historical load data revealed that many applications exhibit cyclical resource consumption. In this paper we show the extraction of load patterns and present self-organizing controlling concepts in the context of the SAP HCC.**

I. Introduction

Enterprise resource planning systems (ERP) undergo significant changes with regard to the hardware and software architecture: Clusters of commodity servers are displacing classical mainframe architectures and monolithic software systems are decomposed into smaller modular components. These new computing paradigms provide higher levels of flexibility, but also demand for new administration principles. Thereby, ERP systems have to respond with consistently low latency as requests are posted in a dialog mode. Thus, an integrated Quality of Service (QoS) management is an important issue for ERP systems. User requests can vary according to the frequency they occur and the load they induce. Nevertheless, using an aggregated view on the number of served users, ERP applications often show statistically periodic load characteristics.

In order to justify this theory, we analyzed the time dependent workload of monolithic ERP training systems (SAP R/3 Enterprise edition). Our contribution is an approach for the extraction of load patterns, which we evaluated on the basis of real-world monitoring data that was provided by the SAP Hochschul Competence Center (SAP HCC) at the Technische Universität München (TUM). The cyclical behavior of applications allows us to optimize the static application-to-server allocation and to supervise the deployment at runtime. Predictable critical situations like overload on servers can be prevented. Knowing the load characteristics of applications, those with complementary behavior can reliably be deployed onto the same servers.

Institutions (partially more than one per system)	Universities	13
	Schools for applied sciences	26
	Vocational schools	16
	University of cooperative education	3
	Total	58
	SAP systems	55
Users	Students	18,390
Application servers	Blade servers (Sun B100s)	96
	Others (Sun V40z)	5
DB servers	Sun Fire V210, V240, V880, V890	50
Storage	RAID capacity	27 TB

TABLE I
SAP HCC customers and infrastructure (Dec. 2005)

II. Target Infrastructure

We examined the CPU load caused by the ERP training applications hosted by the SAP HCC at the TUM. The SAP HCC acts as an application service provider (ASP) for academia [1]. As shown in table I, it provides support for 58 academic customers with an estimated number of about 18,400 users. An ERP training system in terms of the SAP HCC is typically an SAP R/3 Enterprise edition, which includes the classical enterprise functionalities. Some of these basic configurations are enhanced by business analytics covered by SAP's Business Warehouse (BW). We identified three different effects that determine the load curve of such an ERP training system [2].

1) The basic characteristics can be described as a *long-term component*. For example the number of user requests is noticeably reduced during weekends and holidays.
2) *Short-term influences* can be identified by the kind of performed tasks that vary during a day depending on the users' work rhythm, e.g., the course structure.
3) A third component describes *unusual workloads* that emerge when certain tasks are performed by all course participants simultaneously. In general, this does not apply to ERP production systems.

If several of these characteristics occur at the same time, a so-called *whiplash effect* can be observed, i.e., high load peaks appear after short build-up times.

The SAP HCC employs a Blade server architecture that allows to be adaptively extended. Normally, each of the 55 SAP systems consists of several dialog instances (DI) and one central instance (CI). While the DIs can be allocated dynamically on the available hardware, the CI is by default statically assigned to one of the database servers. However, the static deployment can be changed easily due to virtual host names and an SAN infrastructure. The CI provides basic functionality, like dispatching user requests onto the available instances. The assignment of requests to instances can be parameterized. For example, the percentage of user requests for each instance can be defined.

Initially, the hardware allocation was done in a traditional, peak-load oriented way. Better, cost-effective results can be achieved by considering the characteristics of courses. In this regard, applications with statistically complementary load patterns are supposed to be allocated onto the same servers. Therefore, we analyzed the historical load information and extracted application specific load patterns. The protocolled load represents 15 min average-aggregates of CPU load that was measured every 5 seconds. The basis for our experimental evaluation constitutes the monitoring data for 38 ERP training systems that was generated during a German summer semester (February till July).

III. Extraction of Application Load Patterns and Application Classification

In our experiments we analyzed the CPU load induced by ERP systems of the SAP HCC, but our approach is not restricted to a certain notion of load. For example, memory usage or Quality-of-Service (QoS)-relevant parameters like response time, system throughput, or the number of served users can be handled as well. In the following we look at time series referring to generic historical load information.

We developed a three-staged approach for the extraction and evaluation of load patterns, consisting of a *preprocessing phase*, an *analysis phase*, and a *classification phase*. Thereby, we kept the parametrization at a minimum level – a prerequisite for its integration into a self-organizing infrastructure. For the following considerations let S be the set of distinguished ERP training systems.

A. The Preprocessing Phase

At runtime, all application instances are monitored and the load induced by them is logged. In order to determine the characteristics of a system $s \in S$, the aggregated historical load information of all instances of s has to be determined. Thereby, in heterogeneous computing environments, the load induced on host machines of varying capability is considered through performance normalization. Consequently, for each system s an equidistantly sampled time series is calculated, representing its entire load influence over a monitored period T.

B. The Analysis Phase

The extraction of a load pattern proceeds under the assumption of the time series being cyclic. Whether this assumption holds is evaluated in the classification phase. According to the classical additive component model, a time series consists of a trend component, a cyclical component, and a remainder, e. g., characterizing the influence of noise. The trend is a monotonic function, modeling an overall upwards or downwards development. We focus on the cyclical component that describes the periodic load characteristics of a system.

First, we determine the yet unknown duration of the load pattern. In order to do so, we make a combined evaluation of the *periodogram function* and the *auto-correlation*. The periodogram function illustrates the intensities, with which the distinguished harmonics are present in the time series. Intuitively, if the periodogram function has a local maximum at frequency $\lambda > 0$, then it is likely that there exists a pattern of length $T \cdot \lambda$. In general, taking the position of the global maximum is error-prone. Thus, we determine a set of frequency hypotheses $\Lambda = \{\lambda_1, \dots, \lambda_n\}$ of local maxima positions. These hypotheses are then evaluated against the auto-correlation function. The auto-correlation describes linear dependencies within the time series. If the auto-correlation shows local extrema at a quite regular lag, it is a sign that there exists a temporal dependency of certain length. Thus, the characteristics described by the auto-correlation can be used to determine the most suitable frequency hypothesis $\lambda^{(s)}$ that describes the cyclical characteristics of a time series. Assuming that s shows periodic characteristics, $T_1^{(s)} \overset{\text{def}}{=} T \cdot \lambda^{(s)}$ is considered to be the most probable duration of the load pattern.

ERP training systems and business processes typically show a periodicity which is a multiple of hours, days, weeks and so forth. Due to unavoidable computational inaccuracies and the influence of noise, $T_1^{(s)}$ can diverge from the actual period. Thus, we perform a *comparison to calendar specific periods*, to determine the best matching calendar specific pattern length $T_2^{(s)}$, which is a multiple of the core intervals hours, days, and weeks.

Finally, two *patterns are extracted* from the original time series with respect to the supposed pattern durations $T_1^{(s)}$ and $T_2^{(s)}$. One possible approach is to take extracts of the respective lengths out of the original time series. The approach we follow, is to take the average over the pattern occurrences within the time series. In order to reduce the influence of noise and computational inaccuracies, the starting points of pattern recurrences have to be determined reliably. Candidates for distinctive starting points mark significant changes of the derivation. Having determined the starting points, patterns $P_1^{(s)}$ and $P_2^{(s)}$ that refer to $T_1^{(s)}$ and $T_2^{(s)}$ are extracted as the average over the respective occurrences. Thus, patterns are time series as well, but (e. g., through interpolation) they can be seen as functions $\mathbb{N} \rightarrow \mathbb{R}_0^+$, with timestamps as input parameters (e. g., milliseconds and thus $\in \mathbb{N}$). From the two pattern hypotheses we take the one that minimizes the distance to the original time series.

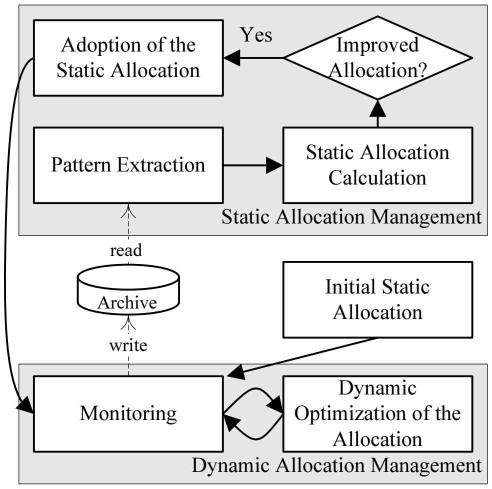

Adoption of the Static Allocation
Yes
Improved Allocation?
Pattern Extraction
Static Allocation Calculation
Static Allocation Management
read
Archive
write
Initial Static Allocation
Monitoring
Dynamic Optimization of the Allocation
Dynamic Allocation Management

Fig. 1. The AutoGlobe framework

C. The Classification Phase

Separating time series with cyclical characteristics from those without is done in the classification phase. The likelihood for a time series showing a characteristic cyclical component is estimated by a weighted evaluation of the pattern quality and the correlation of the frequency hypotheses with the auto-correlation function, as discussed in the previous section. Using a classical Lloyd-Max quantizer, the systems are assigned to clusters of *periodic*, *fuzzy periodic* and *non-periodic* systems.

D. Experimental Results

We evaluated the characteristics of 38 ERP training systems based on monitoring data that was generated during a German summer semester. Our evaluation showed that CPU load is more suitable for detecting cyclical characteristics than other monitoring data. Table II gives an overview of the results we gained based on the evaluation of CPU monitoring data.

Cluster sizes	*periodic*	17
	fuzzy	8
	non-periodic	13
	total	38
Pattern lengths	7-days	23
	more than 30 days	2

TABLE II
Experimental results

The ranking of applications is useful for the static deployment as discussed in the next section. The previous table illustrates that for 23 of the 25 ERP systems that were classified as *periodic* or *fuzzy* 7-day-patterns were determined, which corresponds to the nature of courses. For two of the ERP training systems patterns of length 30 days, respectively 33 days were determined.

IV. Self-organizing Allocation Management

We make use of application-specific load patterns to optimize the application-to-server allocation of large scale ERP landscapes. This idea was developed in the AutoGlobe project [3]. AutoGlobe is a research project focusing on the design and evaluation of self-organizing computing concepts for emerging service oriented architectures. The goal of bringing AutoGlobe and the SAP HCC together is to reduce hardware and maintenance costs. Figure 1 illustrates the basic concepts of the AutoGlobe framework that are grouped into techniques for static and dynamic allocation optimization.

A. Static allocation management

Static application-to-server allocations are calculated based on the estimated load induction of applications. In case no load pattern can be determined for an ERP training system, either the average, or – for more pessimistic estimations – the maximum of the time series is used. The goal of the static allocation management is to prevent overload situations as far as possible and to achieve almost balanced landscape. Thereby, the available hardware shall not be lavishly utilized, thus, providing resources for further applications.

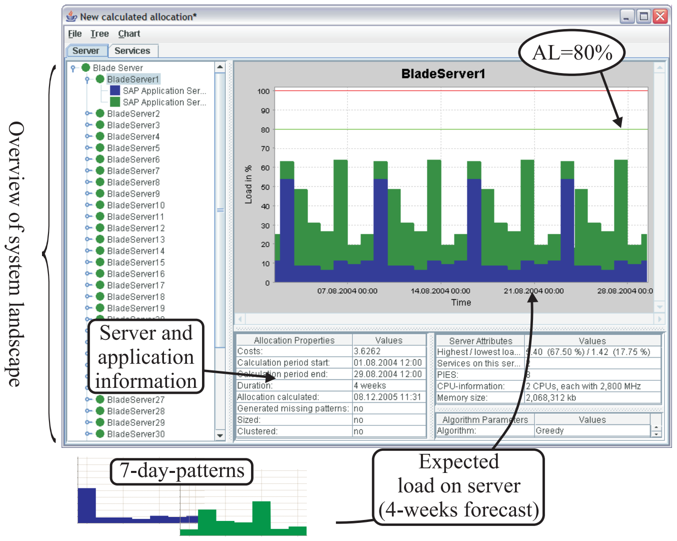
Let C denote the set of available servers and S the set of ERP training systems, i.e., applications. In our setting, the elements of C represent Blade servers. Finding the optimal static allocation is in $O(\mathrm{card}(C)^{\mathrm{card}(S)})$ – a complexity that cannot be handled in the general case. Therefore, we employ a greedy heuristics that successively assigns the systems $s \in S$ to the servers, whereby the systems are ranked according to the quality of their patterns as motivated in Section III-D. One necessary parameter of the greedy approach is the *allocation limit* (AL), which determines the upper bound of resource utilization, e.g., the maximum percentage of processor load. Any resource utilization higher than AL will be considered as overload situation.

Let T be the duration of the time period, for which a static allocation has to be calculated. Suppose that some systems have already been assigned to the available servers and that the next system to be deployed is $s \in S$. The load induced by s is described by the pattern $\left(l^{(s)}(t)\right)_{1\leq t\leq T}$. Analogously, $\left(l^{(c)}(t)\right)_{1\leq t\leq T}$ represents the aggregated load scheduled for systems that are already allocated on server $c \in C$. As mentioned in the previous section, these time series are normalized.

The load increase on server c, when allocating s on it, is estimated by:

$$\Delta\left(s,c\right) \overset{\mathrm{def}}{=} \max_{1\leq t\leq T}\left(l^{(c)}(t)+l^{(s)}(t)\right)-\max_{1\leq t\leq T}\left(l^{(c)}(t)\right) \quad (1)$$

The so-called *best-match-server* is the one that minimizes cost function (1). In case all servers show a resource utilization higher than AL, the server with minimal exceedance is chosen. Figure 2 shows the AutoGlobe user interface and illustrates the optimization calculation. Shown are two exemplary application patterns. As these reveal somehow complementary characteristics (system A shows a load peak on Monday, while system B has peaks on Tuesday and Friday) they can well be allocated on one host.



Allocation Properties	Values
Costs:	3.6262
Calculation period start:	01.08.2004 12:00
Calculation period end:	29.08.2004 12:00
Duration:	4 weeks
Allocation calculated:	08.12.2005 11:31
Generated missing patterns:	no
Sized:	no
Clustered:	no

Server Attributes	Values
Highest / lowest loa...	5.40 (67.50 %) / 1.42 (17.75 %)
Services on this ser...	2
PIES:	8
CPU-information:	2 CPUs, each with 2,800 MHz
Memory size:	2,068,312 kb

Algorithm Parameters	Values
Algorithm:	Greedy

Fig. 2. AutoGlobe-GUI: Example for an allocation of applications

B. *Dynamic allocation management*

Though a reliable static deployment has been arranged, still unforeseeable situations like crashes of applications, overload situations due to wrongly classified patterns or the start of further systems for additional courses can occur and have to be handled. Dynamic allocation adjustments can be performed exception-triggered, i.e., in cases when exceptional situations like overload situations are detected, or pro-actively, using load-forecasting. In such cases, either warning messages are created, or in the sense of an automatism, the system reacts by autonomously migrating applications or starting additional instances. When an ERP system is split into several instances, the dispatcher assigns user requests to instances of s depending on the performance of the host machines these instances are running on (see Section II).

At a first phase, AutoGlobe's controller for dynamic optimization will be used to supervise the SAP HCC system in order to create recommendations that are further controlled by the system administrator prior to their realization. Later on, the reaction process shall be automated as far as possible.

V. Related Work

Substantial efforts have been made in the research community regarding the statistical analysis of historic load data and applying this to adaptive load balancing. Load in this context either refers to CPU usage or the runtime of applications, e.g., covered by the work of [4], [5], [6], [7], the data requirements of applications [8] or characteristics of user requests [9]. Applications with statistically periodic load characteristics are evaluated in [4]. Instead of just making short-termed predictions, we extract and evaluate patterns out of historical load information, which we employ for a long-term static allocation management. Regarding fault tolerance, the influence of wrongly classified patterns is absorbed by a continuous dynamic allocation management, which is presented in [3]. In 2001 IBM coined the term autonomic computing [10] in analogy to the autonomic nervous system. Since then leading companies in the area of information technology integrated autonomic computing concepts in their products. In contrast to these commercial products that depend on vendor-specific hardware, we examine generic self-organizing computing solutions.

VI. Conclusion and Future Work

Regarding the SAP HCC landscape, we were able to automatically classify applications that relate to regularly held courses, which allows a sophisticated analysis and revision of the application deployment. Currently, we are working on a tighter integration of the AutoGlobe system into the HCC landscape, which will help to reduce hardware and maintenance costs. Our intention is to autonomously supervise the ERP training systems, to monitor their "health", report system breakdowns and to calculate optimization propositions. These can be evaluated and confirmed by the system administrator, who can decide to apply them. We assume that the obtained results are also applicable for a pure service oriented approach. Nevertheless, an in-depth analysis of the described concepts for SOA remains as future work.

References

[1] W.-G. Bleek and I. Jackewitz, "Providing an E-Learning Platform in a University Context - Balancing the Organisational Frame for Application Service Providing," in *Annual Hawaii Conference on System Sciences*, Los Alamitos, CA, 2004.

[2] M. Mohr, T. Simon, and H. Krcmar, "Building an Adaptive Infrastructure for Education Service Providing," in *7. Int. Tagung Wirtschaftsinformatik*, Bamberg, Germany, 2005.

[3] S. Seltzsam, D. Gmach, S. Krompass, and A. Kemper, "Autoglobe: An automatic administration concept for service-oriented database applications," in *Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006)*, Atlanta, Georgia, USA, Apr. 2006.

[4] M. Hailperin, "Load Balancing Using Time Series Analysis for Soft Real Time Systems with Statistically Periodic Loads," Ph.D. dissertation, Stanford University, Dec. 1993.

[5] W. Smith, I. T. Foster, and V. E. Taylor, "Predicting Application Run Times Using Historical Information," in *IPPS/SPDP '98: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. Springer-Verlag, 1998, pp. 122–142.

[6] P. A. Dinda and D. R. O'Hallaron, "An Evaluation of Linear Models for Host Load Prediction," in *HPDC '99: Proceedings of the The Eighth IEEE International Symposium on High Performance Distributed Computing*. IEEE Computer Society, Aug. 1999, pp. 87–96.

[7] J. Andersson, M. Ericsson, W. Löwe, and W. Zimmermann, "Lookahead Scheduling for Reconfigurable GRID Systems," in *10th International Euro-Par Conference*, ser. Lecture Notes in Computer Science (LNCS), vol. 3149. Pisa, Italy: Springer-Verlag, Sept. 2004.

[8] S. Vazhkudai and J. M. Schopf, "Using Regression Techniques to Predict Large Data Transfers," in *Int. Journal of High Performance Computing Applications*, vol. 17, 2003, pp. 249–268.

[9] J. R. Santos, K. Dasgupta, G. Janakiraman, and Y. Turner, "Understanding Service Demand for Adaptive Allocation of Distributed Resources," Hewlett-Packard Development Company, Tech. Rep. HPL-2002-85, 2002.

[10] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf, 2001.