February 2007

Optimized Dynamic Allocation Management for ERP Systems and Enterprise Services

Valentin Nicolescu
Technische Universität München, nicolesc@in.tum.de

Martin Wimmer
Technische Universität München, wimmerma@in.tum.de

Raphael Geissler
Technische Universität München, geisslra@in.tum.de

Daniel Gmach
Technische Universität München, gmach@in.tum.de

Matthias Mohr
Technische Universität München, mohrm@in.tum.de

See next page for additional authors

Authors
Valentin Nicolescu, Martin Wimmer, Raphael Geissler, Daniel Gmach, Matthias Mohr, Alfons Kemper, and Helmut Krcmar

In: Oberweis, Andreas, u.a. (Hg.) 2007. *eOrganisation: Service-, Prozess-, Market-Engineering*; 8. Internationale Tagung Wirtschaftsinformatik 2007. Karlsruhe: Universitätsverlag Karlsruhe

ISBN: 978-3-86644-094-4 (Band 1)
ISBN: 978-3-86644-095-1 (Band 2)
ISBN: 978-3-86644-093-7 (set)

Optimized Dynamic Allocation Management for ERP Systems and Enterprise Services

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

Abstract

To ensure the operability and reliability of large scale Enterprise Resource Planning Systems (ERP) and enterprise services, a peak-load oriented hardware sizing is often used, which results in low average utilization. The evaluation of historical load data revealed that many applications show cyclical resource consumption. The identification of load patterns can be used for static as well as dynamic allocation optimization.
In this paper we show the extraction of load patterns and present self-organizing service allocation concepts. This practical evaluation of theoretical adaptive computing concepts is of particular importance for the configuration of emerging service oriented architectures (SOA).

1 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 enterprise services. These new computing paradigms provide higher levels of flexibility, but also demand for new administration principles and more powerful resources resulting in an increased administration overhead [Erl05, 67]. 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 and their services. 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 Technische Universität München (TUM). Furthermore we achieved to extract central monitoring data for realtime anaysis and pattern identification purposes. Thus we focused on the runtime behaviour of a large ERP landscape. The cyclical behaviour of applications allows us to optimize the static application-to-server allocation and to supervise the deployment at runtime. Moreover, 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.

2 Related Work

The analysis of monitoring data and the detection of application specific load characteristics are beneficial for the prediction of future system states. Such load patterns can be used to predict future resource requirements. A flexible architecture supporting the statistical evaluation of log data is presented in [BFBL05]. In [DiOH; SmFT98], CPU usage and application runtime have been evaluated and in [VaSh03] the data requirements of applications were examined. In this paper we present our pattern extraction algorithms by means of CPU monitoring data. Furthermore, we evaluated various log data like memory usage and quality of service parameters like the number of served users and application response times. These evaluations show that in particular business services oftentimes reveal quite regular behaviours. As shown in [HeZS01; RZAA04], load traces of such applications in many cases demonstrate clear daily and weekly patterns. In [ShHe00], also the trend and residual process of load traces was analyzed. Corresponding to the results presented in [CSCD05] and [CCDS05], our experience is that business applications oftentimes show future load demands that are similar to their requirements in the past, meaning that the trend process has minor impact on short-term forecasts. As we will show in this paper, load forecasts allow warning of potential critical situations, like service crashes and overload situations. Thus, sophisticated feedback and feedforward control mechanisms can

be realized [HDPT04]. For instance, in case of a server overload situation, servers that can satisfy the applications' resource requirements can be determined reliably. Future overload situations can be anticipated and avoided through a dynamic allocation management. The authors of [CZGS05] show how system state patterns can be extracted automatically at runtime. Failure diagnosis can be leveraged through the evaluation of such system "signatures". Case studies on the application of prediction schemes were presented in [VAHM02] and [XZSW06]. The forecasts of failures or anomalities have for example been analyzed in [CKFF02; KiFo04] and [BFBL05]. In [CKFF02; KiFo04], anomality detection methods are employed to determine problems of internet services. In [BFBL05], the authors show, how HTTP logs can be analyzed and, e.g., certain site failures can be inferred from changes in the user behaviour. Monitoring and analysis of log data are crucial for the realization of self-organizing computing principles – also refered to as adaptive or autonomic computing [Horn01]. Astrolabe [ReBV03] is an example of a self-organizing computing infrastructure. It provides self-management computing concepts for distributed environments. Our main focus is rather on vast distributed applications than on classical intra-organizational enterprise resource planning (ERP) systems and enterprise grids. In [SGKK06] we presented our self-organizing computing infrastructure called AutoGlobe. AutoGlobe is a research prototype, used to evaluate upcoming adaptive computing trends, mainly by means of simulations. In the context of a long-term collaboration with the SAP HCC [MWNK06] – which is an application service provider for the academia (see Section 3) – self-organizing techniques examined in the AutoGlobe project are also step by step evaluated on the basis of productive systems. First results of this collaboration were presented in [WNGM06]. In this paper we present further technical details about the extraction of application load patterns and the current state of the collaboration. In particular, the evaluation of application load patterns, which we present in this paper, can constitue the basis for a sophisticated admission control [SeBH06].

The demand for flexible, self-organizing infrastructures stems from the requirement of enterprises to sooner react on changes of the market. Therefore, leading hardware and software vendors provide self-organizing products in their portfolio, like IBM Director [MeYa04], Sun N1 Grid [Sun06] or Fujitsu-Siemens FlexFrame [FuSi06]. The SAP HCC provides classical ERP (training) applications. But the evaluation results gained in this environment are supposed to apply to a large extent also to the emerging service oriented architectures [BeCT05; Erl05].

These provide higher levels of service flexibility, thus, supporting dynamic allocation schemes even more comprehensive than classical ERP systems.

3 Target Infrastructure

We examined the CPU load caused by the ERP training applications hosted by the SAP HCC at TUM. The SAP HCC acts as an application service provider (ASP) for academia [BlJa04]. As shown in Table 1, it provides support for 61 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 [MoSK05].

- 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. This effect can also be recognized in productive enterprise environments, with the load depending on the work time of the employees.

- *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. In an enterprise environment this effect can for example be seen, if sales representatives work in-house or a number of batch jobs are worked off.

- 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 but for training environments like the one regarded in this work.

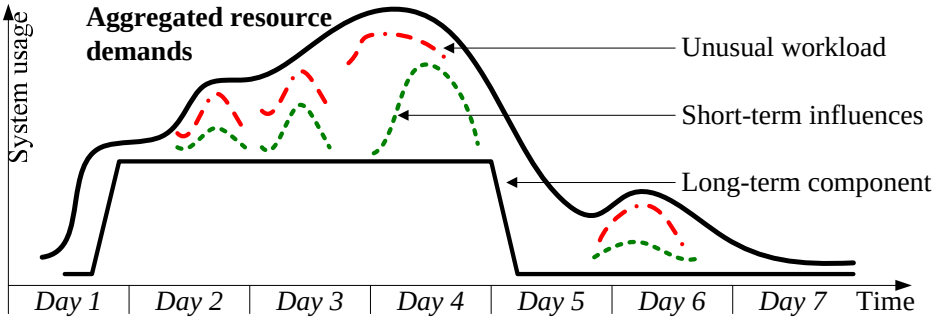

Figure 1: Schematic composition of a 7-day load pattern of an ERP training application

If several of these characteristics occur at the same time, a so-called *whiplash effect* can be observed, meaning that high load peaks appear after short build-up times (see Figure 1).

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). Initially, only one instance runs on each server. 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.

Institutions (partially more than one per system)	Universities	13
	Schools for applied sciences	27
	Vocational schools	17
	University of cooperative education	3
	Total	61
	SAP systems	55
Users	Students	18,390
Application servers	Blade Servers (Sun B100s)	80
	Others (Sun T2000, V40z, X2100, X4200)	17
DB servers	Sun Fire V210, V240, V880, V890	50
Storage	RAID capacity	42 TB

Table 1: SAP HCC customers and infrastructure (June 2006)

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 to July). As the predominantly homogeneous HCC landscape is running in a separate subnet, allocation constraints regarding different subnets and performance of servers were not taken into account.

4 Extraction of Application Load Patterns and Application Classification

We developed a three-staged approach for the extraction and evaluation of load patterns. Thereby, we kept the parametrization at a minimum level – a prerequisite for its integration into a self-organizing infrastructure. During the *preprocessing phase* the historical load information is transformed into time series of equidistant sampling rate. Hypothetical load patterns are determined for each time series during the *analysis phase*. Finally, in the *classification phase*, the quality of the extracted patterns is estimated.

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 paragraphs we therefore look at time series referring to generic historical load information.

4.1 The Preprocessing Phase

At runtime, all application instances are monitored and the load induced by them is logged. Let S be the set of all ERP training systems. In order to determine the characteristics of a certain ERP training 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 $\left(l^{(s)}(t_i)\right)_{1 \le i \le N}$ is calculated, representing its entire load influence over a monitored period of length $T = t_N - t_1$.

4.2 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. The stages of the analy-

sis phase are exemplified for two time series in Figure 2. The left part of the figure represents an ERP training system with periodic characteristics, while the right part represents a system with irregular load characteristics. 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, modelling an overall upwards or downwards development. We focus on the cyclical component that describes the periodic load characteristics of a system.

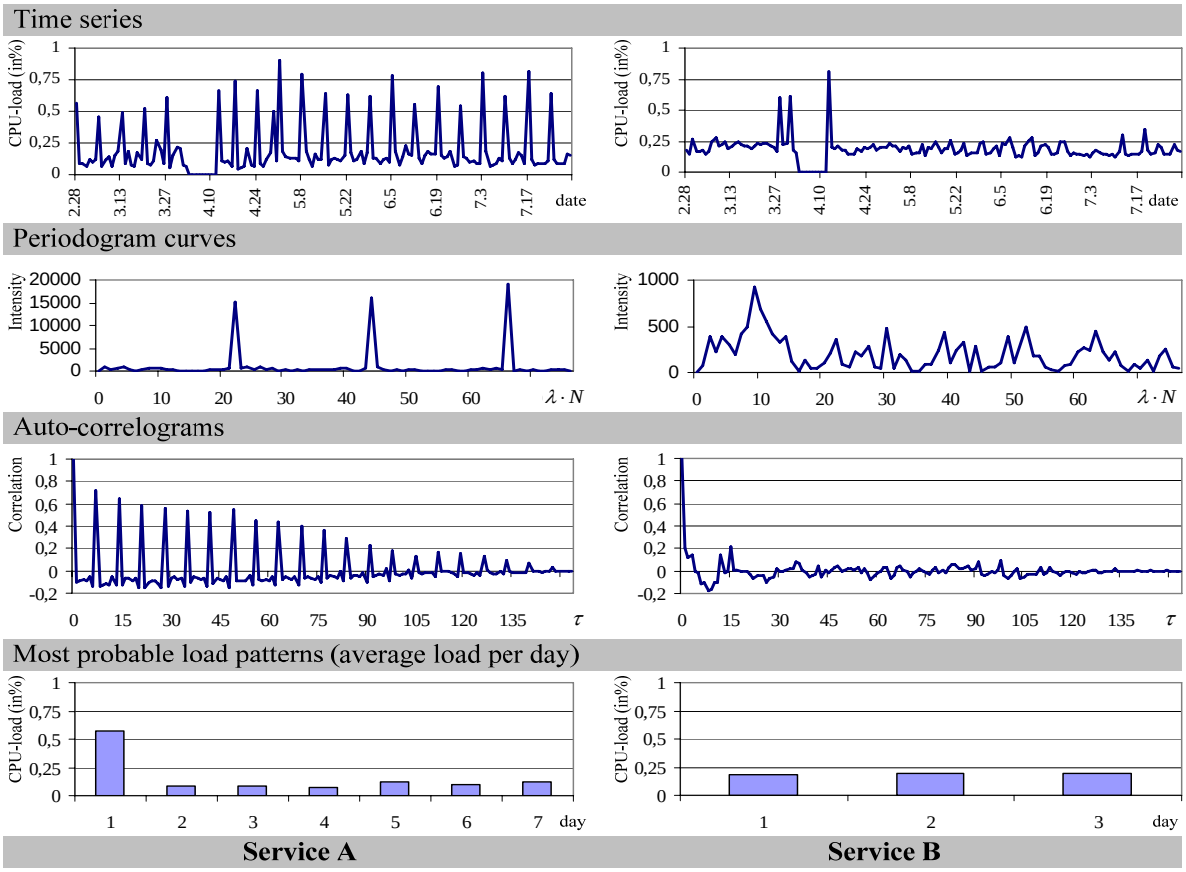

Figure 2: Illustration of the analysis phase – left: system with statistically period, right: system with irregular load characteristics

First, we determine the yet unknown duration of the load pattern, starting with the *evaluation of the periodogram function*: Via Fourier Transformation the time series can be represented as an overlay of harmonics. The periodogram function

$$I(\lambda) = \frac{1}{N} \cdot \left[\sum_{1 \le i \le N} \left(l^{(s)}(t_i) - \bar{l} \right) cos\, 2\pi\lambda i \right]^2 + \frac{1}{N} \cdot \left[\sum_{1 \le i \le N} \left(l^{(s)}(t_i) - \bar{l} \right) sin\, 2\pi\lambda i \right]^2 \qquad (1)$$

defines the intensity, with which the harmonic of frequency $\lambda > 0$ is present in the time series that is normalized by its average value $\bar{l}$. The most dominant frequencies provide information

about the duration of the potential pattern. Intuitively, if I has a local maximum at λ_m, then it is likely that there exists a pattern of length $T \cdot \lambda_m$. In general, taking the position of the global maximum (named $\max_I$) is error-prone. Thus, we determine the set $\Lambda = \{\lambda_1,...,\lambda_n\}$ of local maxima positions, with $I(\lambda_j) > \max I/2$ for every $1 \le j \le n$. The local maxima positions are computed through evaluating the periodogram at i/N with $1 \le i \le N/2$ (using Fast Fourier Transformation) and subsequent iterative approximation.

We then derive, which $\lambda_j \in \Lambda$ determines the pattern duration. For this purpose we *evaluate the auto-correlation* (r_τ) that is defined as

$$r_\tau = c_\tau / c_0 \text{, with } \tau \in \mathbb{Z},\ -N < \pi < N \text{ and } c_\tau = \frac{1}{N}\sum_{1 \le i \le N}\left(l^{(s)}(t_i) - \bar{l}\right)\left(l^{(s)}(t_\tau) - \bar{l}\right) \qquad (2)$$

The auto-correlation describes linear dependencies within the time series. If the auto-correlation shows local extrema at a quite regular lag τ_e (see system A in Figure 2), it is a sign that there exists a temporal dependency of length $\tau_e \cdot T / N$. In a quite similar way as for the evaluation of the periodogram, we determine the set $\Pi = \{\tau_1,...,\tau_m\}$ of significant local extreme positions.

As a next step, the compliance of the elements of Λ and Π is evaluated and the best matching $\lambda^{(s)}$ is determined. Intuitively, $\lambda^{(s)}$ is the best matching frequency, if the elements of Π exhibit a lag that is (approx.) equal to $\lambda^{(s)}N$, i.e., $\tau_j = j \cdot \lambda^{(s)}N$. We estimate the relationship of $\lambda \in \Lambda$ and Π through:

$$\delta(\lambda, \Pi) \overset{def}{=} \frac{d_{Cal}(\lambda)}{card(\Pi)} \cdot \sum_{\tau \in \Pi}\left|\tau - rnd\left(\frac{\tau}{\lambda N}\right) \cdot \lambda N\right| \qquad (3)$$

In the above definition, *rnd* is the rounding operation and *card* determines the cardinality of a finite set. $d_{Cal}(\lambda)$ is used to take calendar specific time periods into account, what will be discussed in more detail below. Let $\lambda^{(s)}$ be the frequency that minimizes δ, i,e., $\delta^{(s)} \overset{def}{=} \delta(\lambda^{(s)}, \Pi) = \min\{\delta(\lambda, \Pi) \mid \lambda \in \Lambda\}$. Assuming that s shows periodic characteristics, $T_1^{(s)} \overset{def}{=} T \cdot \lambda^{(s)}$ 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 cal-*

endar specific periods. Therefore, a rounding operation is required that determines the nearest calendar interval for $T_1^{(s)}$. As core intervals, hours, days, and weeks are used. When calculating the best matching calendar dependent interval, longer core intervals are prioritized, e.g., the approximation of 13½ days by two weeks is favored compared to 14 days and especially compared to 324 hours. Let $T_2^{(s)}$ denote the best matching period regarding $T_1^{(s)}$. The enforcement of calendar related periods is also expressed in the above definition of δ by the weight $d_{Cal}(\lambda)$. There, $d_{Cal}(\lambda)$ describes the difference of $\lambda \cdot T$ and the best matching calendar interval relative to the respective core interval.

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. Figure 3 illustrates the proceeding for selecting the k^{th} starting point. Let (t,l) be the supposed starting point according to the estimated pattern duration, i.e., $t = t_1 + k \cdot T_1^{(s)}$ or $t = t_1 + k \cdot T_2^{(s)}$. Instead of approximating the derivation function, we use a more descriptive geometric approach: The distinctive starting point (t',l') in the figure is determined as the point with the maximum distance ε to the line, which is determined by $(t,\bar{l})$ and the local extremum (t_e,l_e) (with $l_e > \bar{l}$ in case of a local maximum and $l_e < \bar{l}$ in case of a local minimum). 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 $\mathbb{N} \to \mathbb{R}_0^+$, with timestamps as input parameters (e.g., milliseconds and thus $\in \mathbb{N}$). The quality of a pattern $P = (l_P(t))$ is defined as the normalized distance to the original time series, estimated as $\Delta(P) = \frac{1}{N} \sum_{1 \le i \le N} \left| l^{(s)}(t_i) - l_P(t_i) \right|$. The best matching pattern $P^{(s)}$ is the one with

$$\Delta^{(s)} \overset{def}{=} \Delta\left(P^{(s)}\right) = \min\left\{\Delta\left(P_1^{(s)}\right), \Delta\left(P_2^{(s)}\right)\right\}. \quad (4)$$

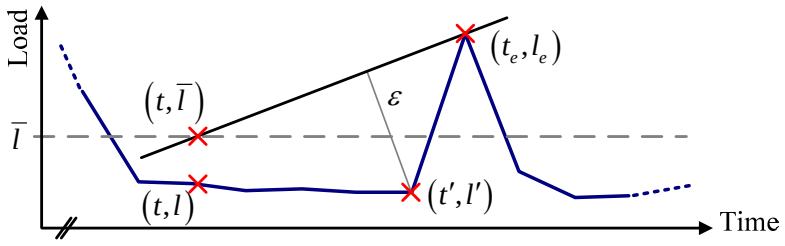

Figure 3: Detection of significant starting points

4.3 The Classification Phase

Separating time series with cyclical characteristics from those without is done in the classification phase. The likelihood for a time series (denoting the load induced by an ERP training system $s \in S$) showing a characteristic cyclical component is estimated by

$$\Omega^{(s)} \overset{def}{=} \omega \cdot \overbrace{\delta^{(s)} / \max\left\{\delta^{(s')} \mid s' \in S\right\}}^{\substack{compliance\ of\ periodogram \\ and\ auto-correlation}} + (1-\omega) \cdot \overbrace{\Delta^{(s)} / \max\left\{\Delta^{(s')} \mid s' \in S\right\}}^{pattern\ quality}. \quad (5)$$

$\omega \in [0,1]$ is a weighting factor and in fact the only parameter that has to be chosen for pattern extraction. In our experiments, we used $\omega = 0.5$.

The lower $\Omega^{(s)}$, the higher the probability of s inducing statistically periodic load. Using a classical Lloyd-Max quantizer, the services are clustered.

4.4 Experimental Results

We evaluated the characteristics of 38 ERP training systems based on monitoring data that was generated during a German summer semester. As mentioned in Section 3, the examined time series represent the aggregated CPU load that was measured on the application servers. Our evaluation showed that CPU load is more suitable for detecting cyclical characteristics than other monitoring data like the number of registered users. We calculated three application clusters: Regarding their likelihood of showing periodic characteristics, we classified the ERP training systems as *periodic*, *fuzzy* and *non-periodic*. Considering Figure 2, system A is classified as *periodic*, while system B is *non-periodic*. The periodogram and auto-correlogram diagrams support this decision: Considering system A the graphs reveal a regular behavior in contrast to the diagrams for system B. Table 2 gives an overview of the results we gained based on the evaluation of CPU monitoring data.

The ranking of applications is useful for the static deployment as discussed in the next section. The table below 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. In contrast to more traditional pattern recognition applications, like image processing, wrongly classified patterns do not have such a strong impact on the optimization technique: False negatives, i.e., services that were not classified as periodic though they actually reveal a cyclical characteristics, reduce optimization possibilities, as for them a standard peak-load oriented distribution will be applied, but do not prohibit optimization at all. False positives, i.e., services that were classified as periodic although they show more or less unpredictable characteristics, are tolerable through the dynamic allocation management that is described in Section 5.2.

Cluster sizes	Periodic	17
	Fuzzy	8
	Non-periodic	13
	Total	**38**
Pattern lengths	7-days	23
	More than 30 days	2

Table 2: Experimental results

5 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 [SGKK06]. AutoGlobe is a research project focusing on the design and evaluation of self-organizing computing concepts for emerging service oriented architectures.

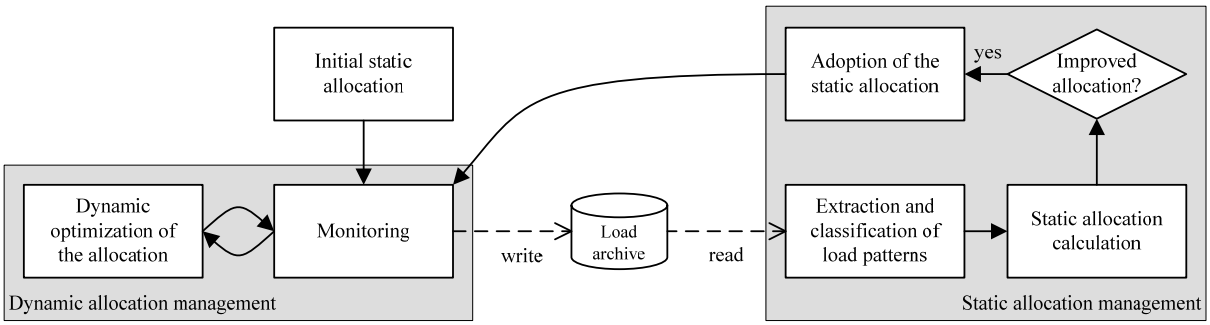

Figure 4: The AutoGlobe framework

Our aim is to successively integrate self-organizing computing techniques into the SAP HCC infrastructure for (semi-) automatic allocation optimization. The goal of bringing AutoGlobe and the SAP HCC together is to reduce hardware and maintenance costs and to evaluate theoretical concepts by means of a real world application scenario. Figure 4 illustrates the basic concepts of the AutoGlobe framework that are grouped into techniques for static and dynamic allocation optimization.

5.1 Static allocation management

Initially, a system is typically set up with a predefined configuration, e.g., designed by the system administrator. Alternatively, if further knowledge about the characteristics of applications is given, AutoGlobe can calculate an initial static allocation. A prerequisite for the computation of a static application-to-server allocation is knowledge about the estimated behavior of applications. Therefore, the load induced by the different ERP training systems is monitored at runtime and protocolled in a load archive. The historical load information is then analyzed and course specific load patterns are determined as described in the previous section. If 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. By use of these load characteristics, an optimized static allocation for a predefined timeframe is calculated. In this regard, optimized means that the workload will be almost balanced and especially critical situations like overloaded systems will be prevented. One further optimization criterion is that the available hardware shall not be lavishly utilized. That is, hardware, which can be used for further applications, has to be determined.

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 as presented in Section 3. Finding the optimal static allocation then is in $O\left(card(C)^{card(S)}\right)$ a complexity that cannot be handled in the general case. Therefore, we employ a greedy approach 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 4.4. One necessary parameter of the greedy heuristics 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 $\left(l^{(s)}(t)\right)_{1 \le t \le T}$. Analogously, $\left(l^{(c)}(t)\right)_{1 \le t \le T}$ 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:

$$\Delta(s,c) \overset{def}{=} \max_{1 \le t \le T}\left(l^{(c)}(t) + l^{(s)}(t)\right) - \max_{1 \le t \le T}\left(l^{(c)}(t)\right) \tag{6}$$

The so-called *best-match-server* is the one that minimizes cost function (6). In case all servers show a resource utilization higher than AL, the server with minimal exceedance is chosen. The greedy heuristics terminates when all systems have been allocated and, thus, a new static allocation is determined. Figure 5 shows the AutoGlobe user interface and illustrates the optimization calculation. The right part of the image shows 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.

AutoGlobe supports optimized allocations to be applied automatically, e.g., at regular maintenance intervals. At the test stage, the allocations calculated by the AutoGlobe framework are used as recommendations. Before adapting the allocation, AutoGlobe evaluates, whether the new allocation provides significant benefit and is thus worth being applied. Therefore, the actual application-to-server assignment is compared with the new alternative considering the expected overload and idle situations and the administrative costs for modifying the system landscape (e.g., application downtimes).

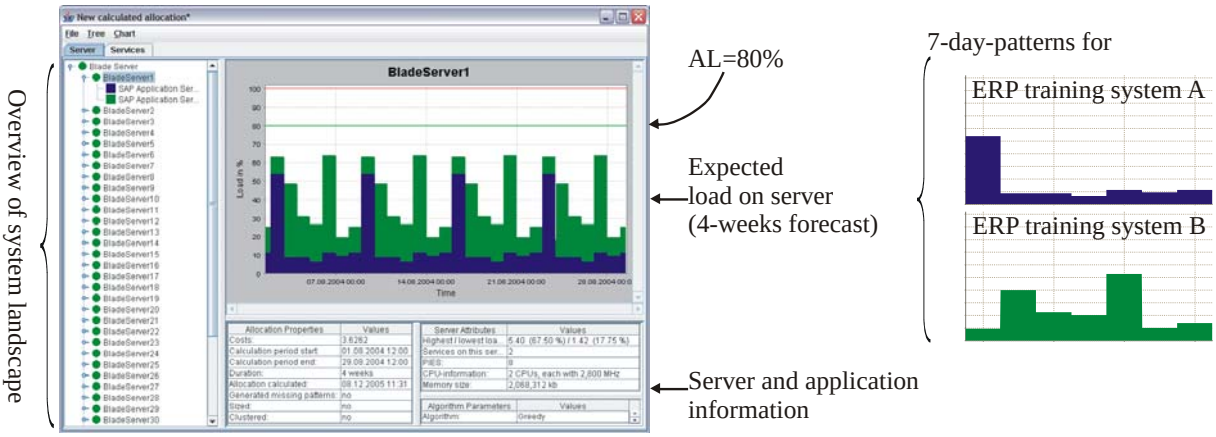

Figure 5: AutoGlobe-GUI: shown is an example for an allocation of two applications onto one server

5.2 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. Therefore, the system is supervised by a controller that reacts on exceptional situations. For example, in case of system breakdowns applications have to be restarted automatically. If an overload situation is detected, i.e., user requests can't be handled sufficiently because a training system s is executed on an overloaded host c, the controller calculates recommendations to remedy the situation. Depending on the flexibility the application provides, it is suggested to either start additional instances of s on other servers or to move instances onto a new host that provides enough free computational resources. For the selection of an appropriate host, load patterns are evaluated for estimating the load on hosts within $[t, t+\Delta t]$ for t being the time the exception occurs and Δt being a predefined look-ahead period. The best matching host is the one, on which an additional instance of s does not cause an exceptional situation during $[t, t+\Delta t]$ and the load increase is at minimum according to cost estimation (6). In case such a host does not exist, the system cannot react and a warning message is created. 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 3).

The described approach is exception-triggered, i.e., the controller reacts in case of an exceptional situation. Moreover, predictive and proactive capabilities are provided through load-forecasting by regularly estimating the load induced on hosts $c \in C$ in between t and $t+\Delta t$. In case overload situations are predicted, either warning messages are created, or in the sense of an automatism, the system reacts by autonomously migrating applications or starting additional instances.

The first step towards an automatic realtime service allocation, is the access to performance data of the HCC system landscape. Therefore we built a Java based interface to the central monitoring system of the landscape. Within this system all performance data is collected and stored in the SAP Central Performance History (CPH). The interface was built using the library SAP Java Connector, which provides access to the SAP development platform via Java. This interface can be used to extract real-time monitoring data, e.g. CPU performance data, from an ERP system for dynamic as well as for static allocation optimization. The current version of the realtime analyzing tool provides a wizard for the configuration of the connection parameters, the polling

intervall and the type of montoring data to be transferred, e.g. CPU performance or number of users logged on. After closing the wizard, the monitoring values are extracted from the central monitoring system and the received values are analyzed regarding their periodical load patterns. In a final step these, patterns and analysis parameters are shown in a graphical user interface. Thus the methods that compute the patterns can be verified visually and the practical use can be tested.

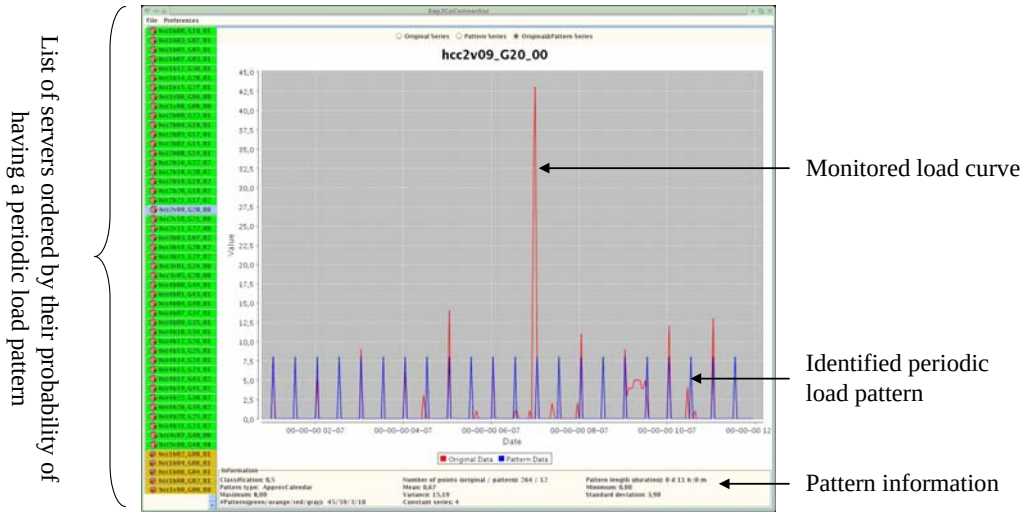

Figure 6: Identified load pattern within the GUI of the realtime analyzing tool

In the next phase, AutoGlobe's optimization and allocation capabilities will be combined with the realtime analyzing tool to supervise the SAP HCC system in order to create realtime recommendations for allocating systems respectively services. These recommendations based on decision rules will then be further controlled by a system administrator prior to their realization. Later on, the reaction process shall be automated as far as possible.

6 Conclusion and Future Work

The extensive evaluation of the SAP HCC monitoring data supports our assumption that many applications exhibit periodic performance characteristics, which can be used to optimize the application-to-server allocation and, thus, to achieve a better balanced system landscape. Using our pattern classification approach, we are able to automatically identify system clusters that show a reliable cyclical behaviour and to separate them from applications with more or less unpredictable load behavior. This helps to reduce hardware costs as combinations of applications onto the same servers can be determined reliably. Thus, free resources are identified which

can be used for additional applications or economized,e.g., by ordering less resources from an ASP. 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. This will allow us to prove theoretical adaptive computing techniques in a representative real world ERP environment and to evaluate, how far automatism for system administration can be driven.

The SAP HCC is a service provider for academia, but classical ERP systems are likely to reveal applications with periodic characteristics as well. Concrete examples are classical OLTP applications with regard to financial operating services (FI) or the logistics execution system (LES). Therefore, the results obtained in the academic field of application will also provide vital benefits for load-balancing requirements of classical ERP landscapes in particular hosted ERP applications. Currently, monolithic ERP systems still dominate enterprise system landscapes and restrict a flexible application-to-server allocation. It becomes apparent that a lot of such systems will be replaced by service oriented architectures (SOA), like SAP's Enterprise Service Architecture (ESA) with its underlying technology platform SAP NetWeaver. Such an enterprise grid architecture provides a higher level of modularity by successively decomposing monolithic applications into fine-grained services that can be allocated in a more flexible way. These services are cleanly partitioned and consistently represented software components [Erl05]. Due to the split-up of applications into a multitude of services the need for intelligent and highly adaptive allocation management will gain in importance.

Our analysis is a step towards evaluating adaptive computing concepts for SOA. We analyzed the characteristics of applications of a large scale academic landscape with a multitude of individual systems. The multitude of systems/services is also characteristic for SOA and it is likely that many enterprise services show cyclical load characteristics as the regarded ERP training systems do. Thus, 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.

Bibliography

[BeCT05] Benatallah, B.; Casati, F.; Traverso, P. (Ed.). (2005). *Proceedings of Third International Conference Service-Oriented Computing – ICSOC 2005, Amsterdam, The Netherlands, December 12-15, Springer 2005.*

[BFBL05] Bodik, P.; Friedman, G.; Biewald, L.; Levine, H.; Candea, G.; Patel, K.; Tolle, G.; Hui, J.; Fox, A.; Jordan, M.I.; Patterson, D. (2005). *Combining Visualization and Statistical Analysis to Improve Operator Confidence and Efficiency for Failure Detection and Localization*. In: *Proceedings of* Second International Conference on Autonomic Computing 2005 (ICAC 2005), 89-100.

[BlJa04] Bleek, W.-G.; Jackewitz, I. (2004). *Providing an E-Learning Plattform in a University Context - Balancing the Organisational Frame for Application Service Providing*. In: *Proceedings of* 37th Annual Hawaii International Conference on System Sciences, Los Alamitos, CA

[CCDS05] Castellanos, M.; Casati, F.; Dayal, U.; Shan, M.-C. (2005). *A Platform for Business Operation Management*. In: *Proceedings of* 21st International Conference on Data Engineering 2005 (ICDE 2005), 5.-8.04.2005 1084-1095

[CKFF02] Chen, M.; Kiciman, E.; Fratkin, E.; Fox, A.; Brewer, E. (2002). *Pinpoint: Problem determination in large, dynamic internet services*. In: *Proceedings of* Dynamic Internet Services: International Conference on Dependable Systems and Networks (DSN'02), 595-604.

[CSCD05] Castellanos, M.; Salazar, N.; Casati, F.; Dayal, U.; Shan, M.-C. (2005). *Predictive Business Operations Management*. In: *Proceedings of* Databases in Networked Information Systems (DNIS 2005), 4th International Workshop, Aizu-Wakamatsu, Japan, 28.-30.03.2005, 1-14.

[CZGS05] Cohen, I.; Zhang, S.; Goldszmidt, M.; Symons, J.; Kelly, T.; Fox, A. (2005). *Capturing, indexing, clustering, and retrieving system history*. In: *Proceedings of* 20th ACM symposium on Operating systems principles, Brighton, United Kingdom, 105-118.

[DiOH] Dinda, P.A.; O'Hallaron, D.R. (1999). *An Evaluation of Linear Models for Host Load Prediction*. In: *Proceedings of* HPDC '99: The 8th IEEE International Symposium on High Performance Distributed Computing, 87-96.

[Erl05] Erl, T. (2005). *Service-Oriented Architecture - Concepts, Technology, and Design*, New York: Prentice Hall.

[FuSi06] Fujitsu-Siemens (2006). FlexFrame for Oracle, http://www.fujitsu-siemens.com/campaigns/flexframe_oracle/index.html, Accessed at: 2006-07-12.

[HDPT04] Hellerstein, J.L.; Diao, Y.; Parekh, S.; Tilbury, D.M. (2004). *Feedback Control of Computing System*: Wiley-IEEE Press.

[HeZS01] Hellerstein, J.L.; Zhang, F.; Shahabuddin, P. (2001). A Statistical Approach to Predictive Detection. *Computer Networks: The International Journal of Computer and Telecommunications Networking, 35*(1), 77-95.

[Horn01] Horn, P. (2001). Autonomic computing: IBM's perspective on the state of information technology, *IBM Corporation*, http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf, Accessed at: 2006-07-10.

[KiFo04] Kiciman, E.; Fox, A. (2004). Detecting application-level failures in component-based internet services. *IEEE Trans Neural Netw., 16*(5), 1024-1041.

[MeYa04] Melenovsky, M.; Yang, J. (2004). IBM Director : Driving Efficiencies in Scale-Out Computing, *IDC*, http://www-8.ibm.com/my/express/IDC_Sys_Mgt_WhitePaper_0324.pdf, Accessed at: 2006-07-12.

[MoSK05] Mohr, M.; Simon, T.; Krcmar, H. (2005). *Building an Adaptive Infrastructure for Education Service Providing*. In: *Proceedings of* Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety, Bamberg, 847-859.

[MWNK06] Mohr, M.; Wittges, H.; Nicolescu, V.; Krcmar, H.; Schrader, H. (2006). Einbindung und Motivation informeller Multiplikatoren im IT-Training am Beispiel Education Service Providing. *Wirtschaftsinformatik-Ausbildung mit SAP®-Software - Publikation zum Track der Multikonferenz Wirtschaftsinformatik 2006 in Passau*, 1-24. Lohmar: Joseph Eul Verlag.

[ReBV03] Van Renesse, R.; Birman, K.P.; Vogels, W. (2003). Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Comput. Syst.*, *21*(2), 164-206.

[RZAA04] Rolia, J.; Zhu, X.; Arlitt, M.; Andrzejak, A. (2004). Statistical service assurances for applications in utility grid environments. *Performance Evaluation*, *58*(2+3), 319-339.

[SeBH06] T.Setzer; Bichler, M.; Hühn, O. (2006). Adaptive Zugriffskontrollverfahren - Ein Entscheidungsmodell für die Kontrolle des Zugriffs auf gemeinsam genutzte IT-Infrastrukturen. *Wirtschaftsinformatik*, *48*(4).

[SGKK06] Seltzsam, S.; Gmach, D.; Krompass, S.; Kemper, A. (2006). *AutoGlobe: An Automatic Administration Concept for Service-Oriented Database Applications*. In: *Proceedings of* 22nd International Conference on Data Engineering, ICDE 2006, Atlanta, GA, USA, 3.-8. April, 90.

[ShHe00] Sheng, D.; Hellerstein, J.L. (2000). Predictive Models for Proactive Network Management: Application to a Production Web Server. *NOMS 2000 IEEE/IFIP Network Operations and Management Symposium*.

[SmFT98] Smith, W.; Foster, I.; Taylor, V. (1998). *Predicting Application Run Times Using Historical Information.* In: *Proceedings of* IPPS/SPDP '98: Workshop on Job Scheduling Strategies for Parallel Processing, 122-142.

[Sun06] Sun Microsystems (2006). Sun N1 Software, *Sun*, http://www.sun.com/software/n1gridsystem/, Accessed at: 2006-07-12.

[VAHM02] Vilalta, R.; Apte, C.V.; Hellerstein, J.L.; Ma, S.; Weiss, S.M. (2002). Predictive algorithms in the management of computer systems. *IBM Systems Journal*, *41*(3), 461-474.

[VaSh03] Vazhkudai, S.; Schopf, J.M. (2003). Using Regression Techniques to Predict Large Data Transfers. *International Journal of High Performance Computing Applications*, *17*(3), 249-268.

[WNGM06] Wimmer, M.; Nicolescu, V.; Gmach, D.; Mohr, M.; Kemper, A.; Krcmar, H. (2006). *Evaluation of Adaptive Computing Concepts for Classical ERP Systems and Enterprise Services*. In: *Proceedings of* IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services (CEC'06 and EEE'06), San Francisco, California, June 26-29, 352-355.

[XZSW06] Xu, W.; Zhu, X.; Singhal, S.; Wang, Z. (2006). *Predictive control for dynamic resource allocation in enterprise data centers*. In: *Proceedings of* 10th IEEE/IFIP Network Operations & Management Symposium (NOMS'06), Vancouver, Canada, 3.-7.April 2006