SCALABLE CONTEXT-AWARE SERVICES

Thomas Buchholz*

Abstract. *Most of the prior work in ubiquitous computing focused on small Context-Aware Services (CASs). However, there is a trend to build also large-scale applications where many context sources, CASs, and users are spread over a large area. These kinds of CASs are subject to strong scalability problems. In my work I show how scalable CASs can be built. The main idea is to dynamically replicate and distribute CASs within the main scalability infrastructure of the world wide web, namely Content Delivery Networks (CDNs). Unfortunately, today's CDNs are not able to deliver CASs mainly due to two reasons: First, it is not solved how replicas of CASs can access context information from within the CDN. Second, the distribution algorithms employed in CDNs are not well suited for distributing CASs.*

In my work I address these two issues. I designed an integration layer that resides on the servers of the CDNs, binds to the heterogeneous regional context sources, and provides context information in a unified form to the replicas of the CASs. Furthermore, I developed a new heuristics for the distribution of CASs within CDNs that bases its replica placement decisions on a profit maximization calculation and uses statistical process control techniques.

1. Introduction

Context-Aware Services (CASs) are applications that use context information to adapt their behavior or to customize the content they provide. Early research in ubiquitous computing focused on small applications where the context sources, the CASs, and users are located in each other's spatial proximity, and where all entities are administrated by the same organization (see e.g. [7] for a survey). In such a setting there is no scalability problem.

However, there are large-scale applications where many context sources, CASs, and users are spread over a large area. Examples for these kinds of applications are context-based push information services like friend-finders or dating services, contextualized web pages (e.g. contextualized portals), contextualized yellow pages like restaurant finders, and server-based navigation systems. These kinds of CASs are subject to strong scalability problems.

According to [14] the scalability problem can be divided into a numerical (N), a geographical (G), and an administrative dimension (A). The numerical dimension refers to the number of users, deployed objects like e.g. context sources and services in a system. The geographical dimension means the distance between the farthest nodes, while the administrative dimension consists of the number of organizations that exert control over parts of the system. A large administrative dimension in general leads to heterogeneity. The various organizations are likely to use different hardware, software, pro-

*Mobile and Distributed Systems Group, Institute for Informatics, Ludwig-Maximilian-University Munich, Oettingenstr. 67, 80538 Munich, Germany, email: thomas.buchholz@ifi.lmu.de, http://www.mobile.ifi.lmu.de/

tocols, and information models. [14] defines a system to be scalable if users, objects and services can be added, if it can be scattered over a larger area, and if the chain of value creation can be divided among more organizations without the system "suffering a noticeable loss of performance or increase in administrative complexity."

The mentioned large-scale CASs have scalability problems in all three dimensions: A single CAS may be used by many users (N) that are scattered over a large area (G). Collectively, a workload (N) might arise that is too large for a single server. Since it cannot be predicted from where (G) a CAS will be invoked and the context sources are often in the user's region (G), the CAS must be able to interact with a plethora (N) of context sources. Often many interactions (N) over a long distance (G) are needed leading to a high user-perceived latency. Furthermore, the context sources in the various regions may possess heterogeneous access interfaces (A) since it is very likely that they will be operated by diverse organizations. Thus, service interoperability becomes a problem. The interoperability problem is classically divided into a signature level, a protocol level, and a semantic level [21]. On the signature level context sources may differ in the syntax of their interfaces. The name of the provided operations and the type and sequence of the stipulated parameters may vary. Often context information needs to be composed by combining context sources sequentially. For example, first it needs to be found out where a user is located before a service can search for nearby bus stops. Thus, context sources need to be orchestrated. This interoperability problem is dealt with on the protocol level. Furthermore, the various context sources may employ different information models. This means that they use different terms to refer to the same concepts which leads to a divergent understanding and interpretation. This is the semantic level of the interoperability problem.

2. Approach and Methodology

This work addresses how large-scale CASs can be built. Its key idea is to reuse the major scalability infrastructure of the world wide web, namely Content Delivery Networks (CDNs) [15]. A CDN is a platform of globally distributed servers located at strategically important points within the Internet. CDNs distribute replicas of the content of their customers, which are the large content providers, on their infrastructure. Client requests are routed to the edge servers that are closest to the client and that already hold a replica of the requested resource with a high probability. Unfortunately, today's CDNs are not able to deliver CASs mainly due to two reasons: First, it is not solved how replicas of CASs can access context information from within the CDN. Second, the distribution algorithms employed in CDNs are not well suited for distributing CASs. In this work these two issues are addressed:

To grant the replicas of the CASs a homogeneous access to context information throughout the CDN, I developed a concept for an integration layer. The integration layer was prototypically implemented and comprises an infrastructure that resides on the servers and a context diffusion infrastructure that is installed on the handhelds. It was evaluated by implementing a restaurant finder on top of it. Furthermore, I conceived a new heuristics that efficiently distributes CASs within CDNs based on a profit maximization calculation. It takes a threshold based approach and uses techniques from statistical process control to react only to persistent and significant changes of the request rate. The heuristics' performance is currently evaluated by simulation.

3. Related Work

Many other researchers recognized the need to make context-aware systems scalable. Most of the work focused on scalable context provision infrastructures like ConFab, Solar, GLOSS, SCI, IrisNet, and iQueue (see [6] for an overview). However, to build scalable context-aware systems not only the context provision process must be scalable, but also the CAS itself. Thus, large-scale context provision infrastructures must be combined with large-scale service provision infrastructures. Candidates are Peer-to-Peer Networks [13], Grids [3], and CDNs [15]. The only infrastructure that is globally in place and can be reused are CDNs. CDNs have the additional advantage that a CAS provider can achieve a global reach of his application by closing contracts with a single organization (the CDN provider). Little research has been done in the area of scalable CASs so far. The only works I am aware of are [2], [20], and [9]. [2] and [20] built a health-care application and exposed the sensors to the system as grid services. A grid service is a Web Service that conforms to a set of conventions. By using grid technology interoperability on the signature level is reached and the scalability properties of grids can be reused. Regardless of which large-scale service provision infrastructure is used, an integration layer is needed to couple it with the context provision infrastructures. Furthermore, algorithms need to be developed that decide when and where to replicate a CAS.

4. Contribution

4.1. Integration Layer

Requirements: To support building large-scale CASs an infrastructure is needed that resides on the servers of the CDN and provides context information (1.) for the replicas of the CASs. Since a developer of a CAS cannot predict on which servers his CAS will be deployed the access to context information must be homogeneous throughout the CDN. This means that the infrastructure must act as an integration layer. It must bind to the heterogeneous regional context sources, map the retrieved information to a standard information model, and provide it to the CASs in a unified way (2.). Very often high level context information needs to be derived from many pieces of low level context information. Thus, an infrastructure should also be able to execute workflows that generate higher-level context information. This means that it must orchestrate the various context sources (3.). Since the context sources will provide context information in various formats (e.g. WGS84 vs. UTM coordinates) the infrastructure must transparently translate between those (4.). Some pieces of context information will exist near the user. These are most efficiently gathered by the user's handheld via wireless technologies. However, there will also be many pieces of context information that are too far away from the user to be captured wirelessly. For example, a restaurant finder might incorporate the crowdedness of candidate restaurants into its recommendations or it might suggest only restaurants that can be reached within 20 minutes with public transportation means taking into account the proximity of bus stops etc. and the delays of buses. These types of information are generated too far away from the user to be retrieved wirelessly. Thus, an infrastructure must be able to combine wireless with wired context retrieval (5.). How context information was obtained (wirelessly or wired) should be transparent for the CASs (6.). Furthermore, an infrastructure should be fault-tolerant (7.), independent from computing platforms and languages (8.), deployable in a web environment (9.), easy to use (10.), and expandable (11.).

Realization: The CoCo infrastructure [5] presented in earlier work is an integration layer that resides on the servers on which the CASs are replicated. It binds to the heterogeneous regional context sources and provides context information to the CAS in a unified form. The CoCo infrastructure is accessed by passing a document in the XML-based CoCo language that specifies which pieces of context information are needed and how they must be combined to generate higher-level context information. The CoCo language is inspired by Web Service Orchestration Languages like the Business Process Execution Language [1] and the Web Service Choreography Description Language [12]. The CoCo language document is parsed and executed by the CoCo infrastructure that responds with the required context information.

It is combined with a Context Diffusion Infrastructure [4] that resides on the handhelds of the users. The main principle of Context Diffusion is that static server nodes announce information via local wireless radio technologies like Bluetooth or IEEE 802.11. Mobile nodes listen to this information and cooperate to disseminate the information by word of mouth. When a user subscribes to a CAS he downloads a thin client that registers its context information requirements at the Context Diffusion Infrastructure. This infrastructure searches in its immediate vicinity for context information via wireless technologies. When the user invokes the CAS, the thin client passes the locally gathered context information to the server-based backend of the CAS. The backend hands the context information to the CoCo infrastructure where it is cached and used when the CAS request context information by passing a CoCo document. In this way it remains transparent for the CAS how context information was retrieved.

To evaluate the integration layer and to show its appropriateness a restaurant finder was implemented based on J2EE technology. The application communicates with the integration layer via a Web Services interface.

4.2. CAS Distribution

Granting a homogeneous access to context information throughout the CDN is one problem that needs to be solved to reuse CDNs for CAS distribution. Another is that the replica placement algorithms that are applied in today's CDNs are suboptimal for the distribution of CASs. The market leader in this industry, Akamai [8], employs a caching based replication strategy. With this approach a CAS would be replicated on a server when the first request for it arrives to this server. Due to the size of a typical CAS this would likely add an unacceptable delay for the user. Furthermore, such on-demand application replication might result in transferring a large application and its data to fulfill a single user request, increasing instead of decreasing the required bandwidth and wasting storage capacity [15].

Other placement strategies have been proposed (see [11] for an overview). Many algorithms are not practically applicable because they assume that the global state of the system is known at a single point within the system where the optimization takes place. This assumption is unrealistic because the incurred overhead to gather and distribute the required information would be too high. More applicable approaches are presented in [17] and [16]. [17] shows that by choosing the best connected nodes within the Internet to set replicas very good user-perceived latencies can be reached. [16] employs a threshold based approach. If more than X requests per time period arrive from a region a replica is set there. However, the threshold must be manually configured. How the threshold is calculated is not addressed in [16].

In my work I take the approach to organize the data centers in a tree (similar to [10] and [18]). The

server that holds the original copy of the CAS becomes the root node. Well connected nodes are more likely to receive a high position within the hierarchy. Thus, I build on [17], but also add some randomness to achieve load balancing. Requests are routed to the nearest data center. If it does not hold a replica, the data center propagates the request to its parent in the tree. Each data center logs how many request per time period arrived. If this request rate λ exceeds a certain threshold, an additional replica is set.

The threshold is determined based on an economical calculation. The latency improvement Δ_L for each user request that is reached by setting an additional replica is multiplied with the request rate λ and the lifetime of the replica τ to derive the total latency improvements that a replica generates over its lifetime. This total latency improvement is monetarily evaluated by multiplying it with a constant ρ_1 that has the unit [€/second]. The resulting value can be interpreted as the revenues a replica generates. Placing and storing a replica also incurs some costs. The storage costs are proportional to the size R of the replica and the storage time which is again τ. Thus, the storage costs are calculated as $\rho_2 * R * \tau$, where ρ_2 is a constant with the unit [€/(second*byte)]. The replica placement costs account for the consumed bandwidth for setting a replica. It is proportional to the size R of the replica. It is calculated as $\rho_3 * R$. ρ_3 carries the unit [€/byte]. Using the revenues and the costs the profit that is generated by an additional replica can be derived: $profit = \rho_1 * \Delta_L * \lambda * \tau - \rho_2 * R * \tau - \rho_3 * R$. The profit must be positive to justify the placement of a replica. Thus, a replica is set if the $profit$ persistently and significantly exceeds 0. This can be transformed to:

$$\lambda > \frac{(\rho_2 * \tau + \rho_3) * R}{\rho_1 * \Delta_L * \tau}$$

Since the heuristics should not react to short term fluctuations of the request rate, an exponentially weighted moving average of λ is calculated. To test whether the moving average significantly exceeds the threshold, approaches from statistical process control are adapted.

A simulation was set up based on the Scalable Simulation Framework [19] to evaluate the new distribution heuristics . It operates on a synthetic Internet topology generated by the Internet Topology Generator [22]. CDN servers, context access points, and client nodes are placed within this topology. Currently, I am in the process of executing simulation runs with varying parameters. The outcomes are encouraging. The complete heuristics as well as the simulation results will be published soon.

5. Conclusion

In my work I show how scalable Context-Aware Services (CASs) can be built. The main idea is to reuse Content Delivery Networks (CDNs). To allow for the distribution of CASs within CDNs an integration layer must be introduced that provides a uniform interface to context information throughout the CDN. I developed a concept for such an integration layer. The integration layer was prototypically implemented and comprises an infrastructure that resides on the servers and a context diffusion infrastructure that is installed on the handhelds. Furthermore, I conceived a new heuristics that efficiently distributes CASs within CDNs based on a profit maximization calculation. It takes a threshold based approach and uses techniques from statistical process control to react only to persistent and significant changes of the request rate. The heuristics' performance is currently evaluated by simulation.

Literatur

[1] Andrews, T. et al. Business process execution language for web services, version 1.1, May 2003. www-106.ibm.com/developerworks/library/ws-bpel/.

[2] Barratt, C. et al. Extending the Grid to Support Remote Medical Monitoring. In *2nd UK eScience All hands meeting*, Nottingham, U.K., September 2003.

[3] Fran Berman, Geoffrey Fox, and Anthony J.G. Hey, editors. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley, April 2003.

[4] T. Buchholz, I. Hochstatter, and G. Treu. Profile-based Data Diffusion in Mobile Environments. In *Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, Fort Lauderdale, USA, October 2004.

[5] T. Buchholz, M. Krause, C. Linnhoff-Popien, and M. Schiffers. CoCo: Dynamic Composition of Context Information. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous 2004)*, Boston, Massachusetts, USA, August 2004.

[6] Thomas Buchholz and Claudia Linnhoff-Popien. Towards realizing global Scalability in Context-Aware Systems. In *International Workshop on Location- and Context-Awareness (LoCA 2005) in cooperation with Pervasive 2005*, Munich, Germany, May 2005.

[7] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

[8] Homepage of Akamai, Inc. www.akamai.com.

[9] Kerry Jean, Alex Galis, and Alvin Tan. Context-Aware GRID Services: Issues and Approaches. In *International Conference on Computational Science*, pages 166–173, Krakow, Poland, June 2004.

[10] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, and Rina Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[11] Magnus Karlsson, Christos Karamanolis, and Mallik Mahalingam. A Framework for Evaluating Replica Placement Algorithms. Technical Report HPL-2002-219, HP Labs, Palo Alto, CA, USA, 2002.

[12] Kavantzas, N. et al. Web service choreography description language version 1.0. W3C Working Draft, December 2004.

[13] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE communications survey and tutorial*, March 2004.

[14] B. Clifford Neuman. *Scale in distributed systems*. IEEE Computer Society, Los Alamitos, CA, 1994.

[15] Michael Rabinovich and Oliver Spatscheck. *Web Caching and Replication*. Addison Wesley, 2002.

[16] Michael Rabinovich, Zhen Xiao, and Amit Aggarwal. Computing on the Edge: A Platform for Replicating Internet Applications. In *The 8th Int. Workshop on Web Content Caching and Distribution*, September 2003.

[17] Pavlin Radoslavov, Ramesh Govindan, and Deborah Estrin. Topology-Informed Internet Replica Placement. In *Proceedings of WCW'01: Web Caching and Content Distribution Workshop*, Boston, MA, June 2001.

[18] Kavitha Ranganathan and Ian T. Foster. Identifying dynamic replication strategies for a high-performance data grid. In *Proceedings of the International Grid Computing Workshop*, pages 75–86, Denver, Colorado, November 2001.

[19] Scalable Simulation Framework Homepage. http://www.ssfnet.org/homepage.html.

[20] Oliver Storz, Adrian Friday, and Nigel Davies. Towards 'Ubiquitous' Ubiquitous Computing: an alliance with the Grid. In *System Support for Ubiquitous Computing Workshop at the Fifth Annual Conference on Ubiquitous Computing (UbiComp 2003)*, Seattle, October 2003.

[21] Thomas Strang and Claudia Linnhoff-Popien. Service Interoperability on Context Level in Ubiquitous Computing Environments. In *SSGRR2003w*, L'Aquila/Italy, January 2003.

[22] Jared Winick and Jamin Sugih. Inet-3.0: Internet topology generator. Technical Report CSE-TR-456-02, EECS Department, University of Michigan, 2002.