International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-2 Issue-1, March 2013

Substance Cache Mechanism in the Cloud by Using Substance Provisioning Framework

Sudeep K. Hase, Milind B. Vaidya

Abstract - The substance (context) information in the context-aware systems may be about human, entity and computing situations has a powerful temporal aspect i.e. for a specific period of time it remains valid. This property can be exploited in caching mechanisms that desired to exploit such locality of reference. Substance information The substance information have varying temporal validity durations and a varied spectrum of access frequencies. This variation affects the suitability of a single caching strategy and an ideal caching mechanism should utilize dynamic strategies based on the type of substance data, access patterns and quality of service heuristics and frequencies of context consuming applications. This paper investigates the various context-caching strategies and proposes a novel bipartite caching mechanism in a Cloud-based substance provisioning system. The bipartite context caching mechanism is achieved through both simulation and deployment in a Cloud platform.

Keywords– Cloud Computing, Substance (context) provisioning, Intelligent caching, substance aware clouds

I. INTRODUCTION

In terms of complexity, substance (context) is the information which is related to the users of computing systems. This includes their personal situations, and environmental characteristics. Context-aware Systems provides the acquisition, representation, distribution and the aggregation of the contextual information in a wide-spread environment. The context-aware system largely makes use of a broker or a contextual server for providing context provisioning from providers of context information to the context consumers. The distributed nature of sensors and services which provides raw data for creation of context is the reason for the provisioning of contextual information to be a non-trivial task.

The present context-aware systems mainly focus on small geographical and conceptual domains and hence the context provisioning function of such systems has not attracted the attention in-depth. The temporal properties of contextual data are not been utilized by the context-aware system for improving the context provisioning performed through caching, grid cloud based platforms. The provisioning of contextual information about anything, anytime and anywhere[2] is one of the key challenge in the context aware system. The context information, depending on the type of content data, remains temporarily valid for some duration of time. For improving the overall performance of the system, this property of context data can be exploited. This can be done employing context caches in the context provisioning systems. The contextual data is central to the fundamental relevance of context aware system. Due to significant increases in number of users device and data sources and the services which are involved in end to end acquisition cycle, inadequate infrastructure support in all terms can prove to be biggest hurdle in the adoption aware system on a large scale. Caching is a well established mechanism for performance improvement in the distributed system. The cloud based context provision with caching can boost its infrastructure strength and improve further the context provisioning function. Context information is mostly represented using name value pairs, structured or semi-structured records and software objects. The context information has a temporal property which is independent of the representation format. Consider example, wherein the location context of a user remains valid only till the user is in that location or Wi-Fi context of a device is present only till the device is connected to a Wi-Fi hotspot. Hence, for improving the context provisioning performance this temporal validity can be exploited. So the Caching contextual data at a context broker side can permit for exploitation of locality of reference to reduce the contextual query satisfaction time and reduce overall context traffic related to the system. As the caches are practically limited in size, the cache replacement policies need to set up at the time of context query response. The variances in scope distribution of the queries, the rate of queries and the validity period influences the effectiveness of the cache replacement policies.

The cloud based context provisioning system permits access to context information by standardized and interoperable interfaces, that will provide device and location independence. It will also provide high reliability and scalability through the elastic and redundant resource. This paper narrates the delivery of caching functionality through cloud based context provisioning system and focuses mainly on the establishment of suitability and reducing effectiveness of various caching strategies for various caching types of contextual data. After this, the performance of the caching strategies in a prototype is analyzed. A novel caching strategy i.e. bipartite context cache is being used for the utilization in context provision system as well as its dynamic reconfiguration based operation is also discussed. The evaluation of the different cache replacement policies in a cloud based environment of the context provisioning architecture is carried out.

Revised Manuscript Received on 30 March 2013.
* Correspondence Author
Sudeep K. Hase*, Department of Computer Science Engineering, University of Pune, AVCOE, Sangamner,, India.
Prof. Milind. B. Vaidya, Department of Computer Science Engineering, University of Pune, AVCOE, Sangamner, India.

II. EXISTING SYSTEM

MobiLife [7], SOCAM [4], PACE [6], CoBrA [3] likewise the number of broker-based context provisioning systems have been structured. The MobiLife structured system contains context (substance) caching where the context provider component has been placed. This working environment reduces the computational load but unable to reduce the communication cost. If we build the collective cache which is a collection of smaller caches at context provider cache (CxP) then query given by the context consumer can be traverse complete cycled from the context provider via a context broker.

How can we improve the quality of context caches has been elaborated by Buchholz et al. [8] and how the context aware system plays an important role that has been discussed by Ebling et al. [9]. validation of the caching context system can be included by using temporal data that means for the particular period of time data remains in the cache. MobiLife provides the architectural level but its context representation does not contain metadata that uses to distinguish temporal properties.

Another system context based service composition (CB-Sec) [12] that contains a cache engine. The cache engine provides context based service composition. Context based service composition stores context service not the context information. Caching service with the internal data application is provided by ScaLaDE middleware. This system supports central cache in the network along with mobile devices cache consistency.

The importance of cache consistency elaborated by some of the researchers but definite result does not find in the context provisioning system. The existing deficiency has served as the reason to do more study in this area.

III. SUBSTANCE PROVISIONING FRAMEWORK

The context victual framework is centered on provider and consumer criterion in which context providers and context consumers plays an important role. This framework is interact with context broker that is responsible for query resolution, routing, event management and lookup services.

This framework consist of three measure components. A context consumer (CxC) which is context based application uses substance data. An on-demand query and substance information is sent when and if it is available by (CxC) or by sending an acceptance to the context broker (CxB) context consumer can fetches the information. The context provider can provide the substance information. The collection of network or cloud services, sensors and other sources have some data and that data is gathered by context provider (CxP). A CxP might use numerous aggregation and reasoning mechanisms to infer context from raw sensing element, network or alternative supply knowledge. A CxP provides context information only to a particular invocation or subscription and is typically specialized during a particular context domain (e.g. Place). A Context Broker (CxB) is the main coordinative element of the framework. CxB has to ease substance flow among all the components, which it achieves by allowing CxCs to accept to or query substance information and CxPs to deliver responses.

The described framework components are shown in fig 1 emphasizing the complementary provision of synchronous and asynchronous context-related communication facilities. A lots of applications have been emerged on this framework. Industrial trials and detailed information elaborated in [15],[16]. CxCs and CxPs register with a context broker by specifying its communication end point and the type of substance they provide or require. This will turn a special function called brokering function in which CxB can find a particular context provider that a CxC may be interested in (i.e. Based on the requested substance). The CxB can cache recently produced substance, in order to exploit the fundamental locality of reference.

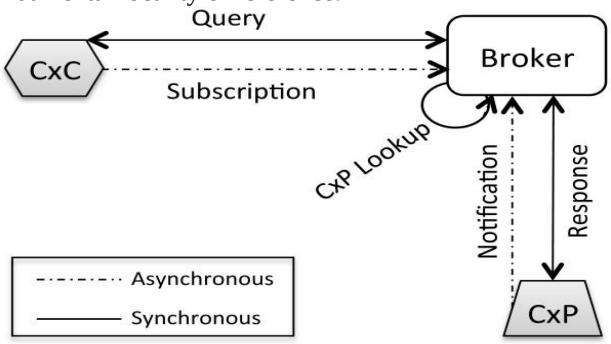

Fig 1 : Basic Broker based substance provisioning component interaction.

To improve the scalability of the overall system and to provide location transparency to the CxCs and CxPs of each broker, the framework of multiple context brokers can be formed. Called as overlay of network of brokers shown in fig 2. This association of CxB is acquire with a coordination framework that is centered on routing of context queries called as subscription and responses called as notification across distributed brokers, lookup function and discovery which is described in proposed system [17]. This alliance of CxB can be directly related to cloud alliances in which two or more geographically distinct or administratively independent clouds operates in resource sharing , hence foundation of alliance of context aware cloud that exchange cross-domain substance information for serving their mobile users.

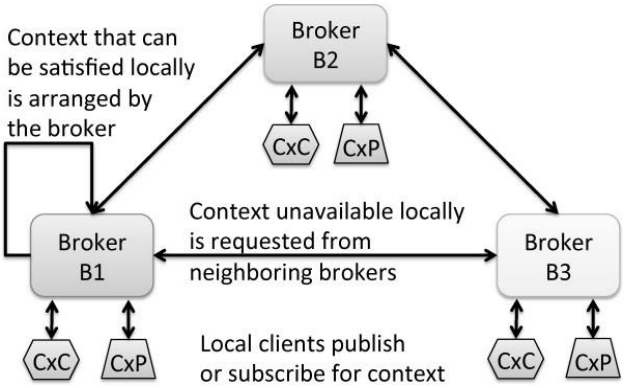

Fig 2 : Simplified view of the federated broker based interaction.

Context data is symbolized in the substance victual framework using an XML based schema entitled ContextML. The purpose of the ContextML is that substance information relates to an entity and if of certain scope. The entity may be a consumer, user, user name, email address and scope defines the weather, places, activity and user preferences. ContextML is consist of the temporal validity associated with ContextML encoded context information through the timestamp and expiry tags, which specifies the time period during which a substance instance is considered valid.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

Thus ContextML creates the basis of utilizing the caching function in the context brokers of the framework.

The substance information about a scope is encoded using entitled parameters, parameter arrays and complex parameter structures in ContextML elements. The parser called ContextML Parser, has been carried out as a Java library for Android platforms and Java EE, SE that can be used by the substance producing and consuming applications for the processing of substance information and other messages encoded in ContextML. The framework of the substance data-related elements and various dimensions of ContextML is presented in an proposed work [18].

A single broker based paradigm of the substance victual framework has been arranged on a Cloud platform and work is under progress to enable a alliance of many such Cloud-based instances to be aliased together in order to exploit the scalability, reliability, performance and interoperability related advantage offered by the Cloud platform. In fig 3 the visionary diagram of how the system components may operate in a alliance of CxB in the Cloud framework for the delivery of substance data to CxC. Each CxB may be under the control of a different administrative authority but the alliance between these CxB (semi-private Clouds) can allow the CxC to utilize these brokers for achieve substance data. The alliance features are apart from scope of this paper.

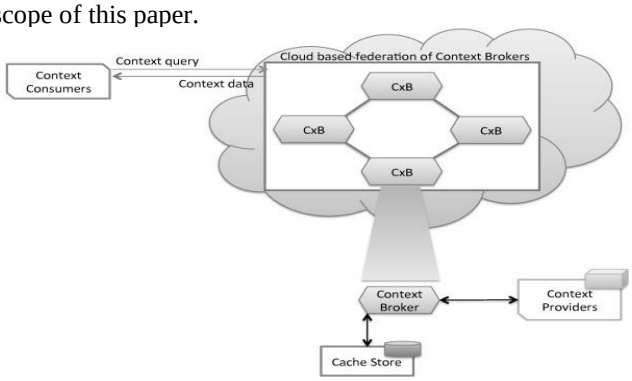

Fig 3 : Architectural components of the Context Provisioning Architecture in the Cloud infrastructure.

IV. SUBSTANCE CACHE

CxC request for substance about a specific object (entity) and scope by forwarding a ContextML encoded query to the CxB. Then the broker forwards the inquiry to the appropriate CxP that can satisfy it. The CxP sends the context response to the CxB when the query-satisfying context information is available. When the caching facility is not present, the CxB simply forwards the query to the querying consumer.

The Substance Provisioning Framework utilizes a caching component that caches recently received substance data in response to substance queries, in advance for forwarding the response to the querying consumer. The substance data remain in the cache for the validity period unless it is altered by more recent substance of the same scope/entity or has to be removed to free the cache due to cache size limits. Processing and notification operations from the CxB's point of view are described in Algorithms 2 respectively.

A. Algorithm 1 (CxB query processing)

WHERE P={$P1$,$P2$,…,$P3$},
P is the set of all providers in the framework
WHERE a query Q={Iq,Ie,Is,$ICxC$,Qp} # $I_q$is the query ID, $I_e$is the entity ID, $I_s$is the scope ID,$I_{CxC}$is the consumer component's ID and Q_p consists of other query parameters.
WHERE Tq={Iqi,$ICxCi$} # $T_q$is a table where the CxB stores the query ID to consumer ID mappings of the form {Iq,$ICxC$}
subscribe(Q) # Query appears at the broker
record(T_q,I_q,I_{CxC}) # Query is recorded in the queries table
CXT_f=*searchCache*(Q,I_e,I_s) # See if cache can satisfy the query
ifCXT_f**then**
notify(I_{CxC},CXT_f) # Notify the CxC in case of cache hit
Check weather item increased or not : incrementUseCount(CXT_f) # Increment the use count of theparticular item
else
P_s=*lookup*(P,I_e,I_s) # CxB looks up an suitable provider
query(Q,P) # And forwards the query to that provider
end if

B. Algorithm 2 (Context Broker notification processing)

WHERE Tq={Iqi,$ICxCi$} # T_q is a table where the CxB stores the query ID to consumer ID mappings of the form {Iq,$ICxC$}
WHERE $T_{Ins}$is the cached item inclusion time
WHERE $T_{Exp}$is the cached item's validity expiry time
WHERE CXT*in* is the cached item's use count
publish(I_q,CXT_p) # Substance response arrives from the CxP
storeInCache(CXT_p,T_{Exp},T_{Ins}) # Store the substance item in the cache
I_{CxC}=*resolve*(T_q,I_q) # Find out which consumer requested this substance item
notify(I_{CxC},CXT_p) # Notify the CxC

The development and real-world deployment of the Substance Provisioning Framework system, a simulation model has been developed to evaluate the system under various conditions. OMNET++[19] is the base of this simulator. The conclusion of the experiments carried out with this simulated setup will aid in establishing the suitability and relative effectiveness of caching strategies for context provisioning. Caching approach can then be readily implemented in a Cloud-based Substance Provisioning to augment the accuracy, scalability and device/location independence benefits that are provided by the Cloud setup.

C. Concept of Simulation

The simulation model consists of a CxB module, CxP and CxC connected by communication channels. The simulator contains the core utility of substance caching, substance querying service, provider registration and lookup service. The ContextML structure is also fully modeled. Provider modules provide substance on invocation by the broker and provide substance about particular single scope only. This model comprises various input parameters that can be set individually for each simulation run allowing several conditions to be carried out and compared against each other. Each scope of parameter contains numerical scope ID (integer) and its validity duration (seconds). Parameters for each Provider consist of a CxP ID(integer), the ID of the substance scope that it provides (integer) and the average time taken to process a query and respond to it (millisecond). The substance broker module parameters include the lookup time for finding CxPs for satisfying queries(ms), cache access time [ms], caching enabled(Boolean), maximum cache size and the cache strategy.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

$$ScopeID = \left[maxScopeID \cdot \left(\frac{\varepsilon}{randUniform\,(0,1)}\right)^{-\sigma}\right] \quad (1)$$

In this evaluation, there are three main caching strategies that we will evaluate, including *remove oldest first (OF), remove least used first (LU), and remove soonest expiring first (SE)*, in addition to the non-practical strategy of having an infinite cache size thus requiring no replacement policy. When the condition occurs that cache is full and space is required for a more recent context item, the *OF* policy the oldest item from the cache. Substance caching functionality in our framework therefore records the time of insertion of each item in the cache. *SE* policy deletes the substance item from the cache store whose *expiry* time will be up the soonest. Case is different in *LU* policy, the substance item which has been accessed the least number of times. The policy is applicable when the caching function in our system has to record each substance item's access frequency.

D. Algorithm 3

Substance cache insertion and replacement procedure (storeInCache)
WHERE T_{ins} is the cached item's insertion time
WHERE T_{Exp} is the cached item's validity expiry time
WHERE CXT_{in} is the cached item's use count
WHERE policy=`LU' ∨ `OF' ∨`SE' # The cache replacement policy
If filledSpace<maxSpacethen insert(MD5(*Ie*||*Is*),CXT,T_{Exp})
else
ifpolicy==`SE'then
CXT_{Rem}=Minimum(T_{Exp}) #Select the item with the soonest reaching expiry time remove(CXTRem)
end if
else
if policy==`OF' then
CXT_{Rem}=Maximum(T_{Ins}) # Select the item with the oldest insertion time remove(CXTRem)
end if
else
if policy==`LU' then
CXT_{Rem}=Minimum(CXT_{in})# Select the item with the least usage count
end if
end if

We have built the usefulness of caching substance data in principle in our earlier work [14], but did not analyze the effect of variance in the scope validity durations in detail. Results show that, different access patterns from users can have a significant influence on the performance of the cache (cache-hit rate). By using this model, we intend to build suitable strategies for varying access patterns and devise a caching strategy that can accommodate a combination of these access patterns. The substance consumers are configured to request substance a constant rate λ[/s]. The substance scope specified by the consumers in the queries is determined using a Pareto distribution with a selectable shape α and scale ξ(1).

Pareto distribution has been elected because it allows us to model scope distribution in substance queries with tunable parameters. In this experiment twelve different scopes are used and the scope distribution in substance queries is managed by changing the Pareto shape parameter α while the scale parameter ξ is kept constant at 1. In each simulation run 5,000 substance requests distributed across 10 entities are instantiated. When the response is received by CxCs, simulation is terminated. The broker cache access time and CxP lookup time are assumed to be 10ms. In each simulation run, a caching approach is elected and the Pareto distribution for selecting the requested scopes (α) is varied to select a certain percentage of short validity (SV) and long validity (LV) category scopes. Simulation is repeated for each strategy and the record is maintained for a query satisfaction time, time elapsed between the issuance of a query from a CxC and receipt of a response to that query. Therefore, the performance of the selected cache strategies is investigated with varying scope distribution in the context requests.

V. THE BIPARTITE CONTEXT CACHE

We have considered different cache replacement policies for SV and LV scope categories, we divided the physical cache into two parts, one catering for the SV scoped substance data items and the other for LV scoped items. Caching approach is then configured to utilize OF replacement policy for SV scoped data and SE policy for LV scoped data items. The bipartite cache gives a marginally improved overall performance over the SE and OF. Therefore use of two different cache replacement policies suited to the scope validity durations of the data items results in an improved performance and provides a fairly constant mean query satisfaction time across all scope distribution patterns.

We have further evaluated the bipartite caching mechanism with a dynamic scaling of the size of the two partitions that is based on the distribution of scopes in the incoming context queries. Increasing or decreasing the size of a partition dynamically based on the ratio of a particular scope validity category in the incoming queries tunes the cache to accommodate the pattern of queries that exists in a particular situation. The time of query satisfaction improves marginally by the application of bipartite caching with dynamic partitioning from the case of equally sized bipartite cache.

Experiments are carried out in a simulated environment (OMNET++) have established that the bipartite caching mechanism, with dynamically as well as fixed re-sizable partitions, handling different replacement strategies in both partitions provides a better hit ratio. Therefore the satisfaction time of mean query improves in comparison to the SE and OF policies operating independently.

VI. CLOUD BASED EVALUATION

The archetype implementation of the Substance Provisioning Framework has been carried as a collection of applications and services conforming to the Enterprise JavaBean [20] specification. The CxB EJB application consists of functional entities that provide the querying, notification, registration, brokering as well as caching functions through RESTful HTTP interfaces. Cache Service is carried out as a Singleton Session Bean of the EJB specification with appropriate interfaces for deleting, adding and fetching context information. Design of singleton session beans are made for conditions in which a single enterprise bean instance is shared across and concurrently accessed by clients. clients are the constituent components of the CxB. Cache Service has built in cache replacement policies and the administrator can configure which policy is to be used during execution. Generic CxP and CxC applications have also been developed that can be programmed for querying and providing context information related to a particular scope and entities at specified rates.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The CxB is deployed on a Glassfish [21] which is an application server, hosted on a compute node of the OpenStack cloud platform (Diablo release). OpenStack cloud platform is implemented on a server consisting of a nova-compute virtual machine. The CxP's are deployed on a server that is accessible to the CxB via a local area network through the host OS. The CxC are deployed on a workstation on the public network available to the Cloud nodes.

VII. CONCLUSION

In this paper we have worked to improve established mechanism of caching in distributed system. However, What is the effectiveness and use of substance caches has not been evaluated or demonstrated. The Substance Provisioning Framework employs a caching mechanism at the CxB, which positively affects the mean query satisfaction time between CxC and CxP. By using OMNET++ discrete event simulator we have analyzed the relative performance of various cache replacement policies. Analysis talks about caching strategies display contrasting behavior under different scope distribution scenarios, with *OF* policy performs better for short scoped substance data and *SE* performs better for long scoped substance data. By using this information, we have devised a novel bipartite caching strategy for use in substance data provisioning that allows utilization of the *OF* and *SE* policies for SV and LV scoped substance data during context provisioning.

Bipartite cache is then improved by allowing dynamic resizing of the bipartite cache partitions based on the scope distribution scenario of the incoming substance queries. This novel strategy can assist in designing a Cloud based substance provisioning system that *productively* utilizes the temporal validity aspects of the substance data, exploit the principle of locality to improve query-response times and hence positively influence the quality of service of the substance-aware system. We have repeated the simulation supported experiments on a deployment of the Substance Provisioning Framework in a Cloud platform.

FUTURE WORK

The Substance Provisioning Framework enables multiple CxB's to be aligning together in an overlay network and the geographically distributed CxC's and CxP's can be attached to different brokers. In such a composition, there will be multiple distributed caches in the system, one at each CxB. The caching advantages cannot be maximized if such distributed caches are not synchronized amongst each other. Such a synchronization activity does not exist in our system and is a aim of our future work. The synchronized and distributed caches may also affect the suitability of cache replacement policies to certain distributions of scopes in the substance queries and we also expect that an investigation into this aspect may open further avenues of research and development in this field.

REFERENCES

1. Saad Liaquat Kiani, Ashiq Anjum, Kamran Munir, Richard McClatchey (2102) Context Caches in the Clouds http://www.journalofcloudcomputing.com/content/pdf/2192-113X-1-7.pdf
2. Weiser M (1991) The computer for the twenty-first century. Sci Am 265(3) : 94–104
3. Chen H (2004) An Intelligent Broker Architecture for Pervasive Context-Aware Systems. University of Maryland, Baltimore County
4. Gu T, Pung HK, Zhang DQ (2005) A Service-oriented middleware for building context aware Services. J Netw Comput Appl 28: 1–18
5. Bardram JE (2005) The Java Context Awareness Framework (JCAF) – a service infrastructure and programming framework for context-aware applications. In Pervasive Computing, Volume 3468 of *LNCS* 98–115. Springer
6. Henricksen K, Indulska J, McFadden T, Balasubramaniam S (2005) Middleware for distributed context-aware systems. In On the Move to Meaningful Internet Systems 2005: CoopIS, DOA , and ODBASE, Volume 3760 of Lecture Notes in Computer Science, ed. Meersman R Tari Z 846–863. Berlin / Heidelberg: Springer. http://dx.doi.org/10.1007/11575771 53
7. Floreen P, Przybilski M, Nurmi P, Koolwaaij J, Tarlano A, Wagner M, Luther M, Bataille F, Boussard M, Mrohs B, *et al* (2005). Towards a context management framework for mobiLife. 14th IST Mobile & Wireless Summit 7.
8. Buchholz T, K¨upper A, Schiffers M (2003) Quality of Context: What It Is and Why We Need It. In Workshop of the HP OpenView University Association
9. Ebling M, Hunt GDH, Lei H (2001) Issues for Context Services for Pervasive Computing. In Workshop on Middleware for Mobile Computing, Heidelberg. http://www.mobilesummit.de/authors.php
10. Lei H, Sow DM, Davis I, John S, Banavar G, Ebling MR (2002) The design and applications of a context services. ACM SIGMOBILE Mobile Comput Commun Rev 6(4): 55
11. Kernchen R, Bonnefoy D, Battestini A, Mrohs B, Wagner M, Klemettinen M (2006) Context-awareness in mobiLife. In Proceedings of the 15th IST Mobile Summit. IST Mobile Summit. Mykonos, Greece
12. Most´efaoui SK, Tafat-Bouzid A, Hirsbrunner B (2003) Using context information for service discovery and composition. In 5th International Conference on Information Integration and Web-based Applications and Services (iiWAS). O¨ stereichische Computer Gesellschaft, ISBN 3-85403-170-10, ed. Kotsis G, Bressan S, Catania B, Ibrahim IK
13. Bellavista P, Corradi A, Montanari R, Stefanelli C (2006) A mobile computing middleware for location and context-aware internet data services. ACM Trans Internet Technol (TOIT) 6(4): 380
14. Kiani SL, Knappmeyer M, Reetz E, Baker N (2010) Effect of Caching in a Broker based Context Provisioning System. In Proceedings of The 5th European Conf. on Smart Sensing and Context, Vol 6446, LNCS 108–121
15. Zafar M, Baker N, Moltchanov B, Jo˜ao Miguel Goncalves SL, Knappmeyer M (2009) Context Management Architecture for Future Internet Services. In: ICT Mobile Summit 2009. Santander, Spain
16. Knappmeyer M, T¨ onjes R, Baker N (2009) Modular and extendible context provisioning for evolving mobile applications and services. In: 18th ICT Mobile Summit
17. Kiani SL, Knappmeyer M, Baker N, Moltchanov B (2010) A Federated Broker Architecture for Large Scale Context Dissemination. In: 2nd Int'l Symp. on Advanced Topics on Scalable Computing. Bradford, UK
18. Knappmeyer M, Kiani SL, Fr´a C, Moltchanov B, Baker N (2010). In: Proceedings of IEEE International Symposium on Wireless Pervasive Computing
19. Varga A (2001) The OMNeT++ discrete event simulation systems. In: Proceedings of the European Simulation Multiconference (ESM'2001) 319–324
20. Emmerich W, Kaveh N (2002) Component technologies: Java beans, COM, CORBA , RMI , EJB and the CORBA component model. In: Proceedings of the 24th International Conference on
21. Software Engineering (ICSE 2002). IEEE 691–692
22. Goncalves A (2009) Beginning Java EE 6 Platform with GlassFish 3: from novice to professional. From Novice to Professional Series, Apress

AUTHOR PROFILE

Sudeep K. Hase completed his B.E.(I.T.) from Amrutvahini College of Engineering, Sangamner and perusing his Master degree (Computer Science and Engineering) from the same institute. He has got 1st rank in department of IT (AVCOE, Sangamner 2011-2012). He has 1 year of industry experience in ESDS software solutions, Nashik (Cloud Computing)

Prof. Milind B. Vaidya is a associate professor at Amrutvahini College of Engineering (Computer Science). He has 18 years of teaching experience.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)