2009 IEEE International Conference on Web Services

Scalable and Reliable Location Services Through Decentralized Replication

Gong Zhang, Ling Liu, Sangeetha Seshadri, Bhuvan Bamba, and Yuehua Wang
College of Computing, Georgia Institute of Technology

***Abstract*—One of the critical challenges for service oriented computing systems is the capability to guarantee scalable and reliable service provision. This paper presents Reliable GeoGrid, a decentralized service computing architecture based on geographical location aware overlay network for supporting reliable and scalable mobile information delivery services. The reliable GeoGrid approach offers two distinct features. First, we develop a distributed replication scheme, aiming at providing scalable and reliable processing of location service requests in decentralized pervasive computing environments. Our replica management operates on a network of heterogeneous nodes and utilizes a shortcut-based optimization to increase the resilience of the system against node failures and network failures. Second, we devise a dynamic load balancing technique that exploits the service processing capabilities of replicas to scale the system in anticipation of unexpected workload changes and node failures by taking into account of node heterogeneity, network proximity, and changing workload at each node. Our experimental evaluation shows that the reliable GeoGrid architecture is highly scalable under changing service workloads with moving hotspots and highly reliable in the presence of both individual node failures and massive node failures.**

I. Introduction

As the cost of the mobile devices and its accessories continue to decrease, there is a growing demand for high performance location based service architecture, aiming at providing scalable and reliable location based information delivery in large scale pervasive computing environments. In contrast to centralized client-server architecture, decentralized management and provision of location based services have gained lot of attentions in the recent years due to its low cost in ownership management and its inherent scalability and self-configurability.

Most of the research and development in decentralized service computing systems has been focused on unstructured overlay network computing, exemplified by Skype and BitTorren, and structured overlay network systems. Measurements performed on deployed overlay networks show that node characteristics such as availability, capacity and connectivity, present highly skewed distribution [1] and such inherent dynamics creates significant variations, even failures, on the services provided by the overlay systems. For example, a sudden node failure that causes the service interruption may lead the system to exhibit dramatic changes in service latency or return inconsistent results. Furthermore, increasing population size of mobile users and diversity of location-based services available to mobile users have displayed rapidly changing user interests and behavior patterns as they move on the road, which creates moving hot spots of service requests and dynamically changing workloads. Thus an important technical challenge for scaling location service network is to develop a middleware architecture that is both scalable and reliable, on top of a regulated overlay network with node dynamics and node heterogeneity, for large scale location based information delivery and dissemination. By scalable, we mean that the location service network should provide effective load balancing scheme to handle the growing number of mobile users and the unexpected growth and movements of hot spots in service demand. By reliable, we mean that the location service network should be resilient in the presence of sudden node failures and network partition failures.

In this paper we present Reliable GeoGrid, a decentralized and geographical location aware overlay network service architecture for scalable and reliable delivery of location based services (LBSs). The main contributions of this paper are two folds. First, we describe a distributed replication scheme which enables the reliable location service request processing in an environment of heterogeneous nodes with continuously changing workloads. Our replication framework provides failure resilience to both individual node failures and massive node failures, aiming at keeping the service consistently accessible to users and eliminating the sudden interruption of the ongoing tasks. Second, we present a dynamic replica-based load balancing technique, which utilizes a parameterized utility function to control and scale the system in the presence of varying workload changes by taking into account of several workload relevant factors. Our experimental evaluation demonstrates that Reliable GeoGrid architecture is highly scalable under changing workloads and moving hotspots, and highly reliable in the presence of both individual node failures and massive node failures.

II. System Overview

Reliable GeoGrid comprises of a network of computing nodes such as personal computer or servers with heterogeneous capacities. The system consists of four core components: topology management module, routing module, replication module and load balancing module.

Topology management.

All nodes are represented as points in a two dimensional geographical coordinate space, which bears a one-to-one mapping to the physical coordinate system. At any time instant, the network of N nodes will dynamically partition the entire GeoGrid coordinate space into N disjoint rectangles such that

978-0-7695-3709-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICWS.2009.57

each node manages its own rectangular region within the entire coodinate space based on its geographical information and handles all location service requests mapped to its region based on the geographical information of the requests. Fig. 1 shows a two dimensional geographical coordinate space partitioned among 17 GeoGrid nodes (for simplicity, we denote each node and its mangaged region with the same number).

GeoGrid is constructed incrementally. It starts with one node who owns the entire GeoGrid space. As a new node p joins the system, it first obtains its geographical coordinate by using services like GPS (Global Positioning System) and then obtains a list of existing nodes in GeoGrid from a bootstrapping server. Then node p initiates a joining request by contacting an entry node selected randomly from this list. The joining request is routed to node q whose region covers the coordinate of the new node. The region owned by q now is split into two halves, one half owned by q and the other half owned by p. In addition to neighbor list, Reliable GeoGrid node also maintains a replica list to provide recovery capability and a routing list for fast routing in the presence of large network size. Mobile users obtain GeoGrid location service by connecting to a GeoGrid node, either through wireless or wired network connections.

In the first prototype design of Reliable GeoGrid, each node is equipped with the capability for submitting location service requests in the form of *Location Query*, routing and processing location service requests, and delivery of results to the mobile users. For example, a car driver may post a service request "send me the traffic conditions within 5 miles every 10 minutes in the next 1 hour". We assume that location-dependent information sources, such as traffic monitoring cameras, owners of gas stations, and restaurants, and so forth, are external to the service network of Reliable GeoGrid.

Routing Protocol.

Routing in a GeoGrid network works by following the straight line path through the two dimensional coordinate space from source to destination node. A routing request is forwarded initially from its source initiator node to one of its immediate neighbors, say q, which is the closest to the destination location (x, y). If (x, y) is covered by the region owned by the chosen routing node q, then the node q will be the owner node of this request. Otherwise, q starts the forwarding process again until the request reaches the node whose region covers (x, y). For example, in Fig. 1, a routing request is initiated by node 7 for a point covered by node 3 is forwarded through nodes 12,2,4,6 in order and finally arrives its owner node 3.

The other two components of Reliable GeoGrid are replication module and load balancing module, which use replication to provide reliability and scalability for location service. Due to space constraints, the rest of the paper focuses on these two components.

Each location service request is issued only once and once it is installed into the system, it is read-only and need to be persistent until it expires. The proposed distributed replication scheme replicates all location requests across multiple selected replica hosts in the network and all the replica hosts get the same copy of the read-only location service request, though each location service is only executed at one executor node at any given time. In Reliable GeoGrid, every location service request has an initiator node, an owner node and an executor node. We call the node that receives the location service requests from the mobile users residing in its region *the initiator node* of the services. Each location service request will be routed to the destination node whose region covers its geographical coordinate or the coordinate of the center of the spatial area if the request is querying on a spatial region. The destination node is called *owner node* of the location query, which chooses one of its replica nodes to execute the nodes working as *executor node*. When the executor node of a location query fails unexpectedly, one of the replica nodes will be chosen as the new executor node.

III. Replication and Replica Management

The GeoGrid replication scheme follows two design principles. First, we want to control the replica management cost by creating and maintaining a constant number of replicas for all services. Second, the replica nodes should be selected from both nearby nodes and remote nodes such that we can take advantage of geographical proximity inherent in the Reliable GeoGrid system to reduce the routing cost involved in recovery and at the same time we can increase the failure resilience of GeoGrid against network partition failures.

A. Failure Patterns and Risk Analysis

A failure is defined as *an abrupt disconnection* from the Reliable GeoGrid service network without issuing explicit notifications. In practice, such sudden behavior may be caused by computer node crash, network connectivity problems, or improper software termination. Reliable GeorGrid network supports a fail-stop assumption and failures can be captured by prolonged heart-beat messages. By fail-stop assumption we mean that node will stop execution, and lose the contents of volatile storage whenever a failure occurs and node never acts an erroneous action against the system due to a failure [2].

Two types of failures are most common in overlay networks: individual node failures and massive node failures. By individual node failure, we mean that a single node experiences failures independently under fail-stop assumption. Individual node failure may render part of the service interrupted or cause permanent loss of service or service state information or query results, if there is no failure resilience protection employed at the individual node level. By massive node failures we mean that when the underlying IP network partitions, overlay network may partition as well and under such network partitions, the nodes in one partition component are separated from the nodes in another partition component and all the messages across the different network partition components fail to get response and thus create significant delay or critical failures. We argue that a higher level failure resilience mechanism, which can mask the simultaneous failures of multiple nodes,

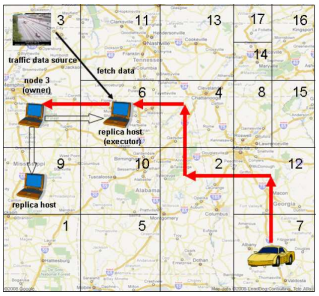

Fig. 1. Service Model

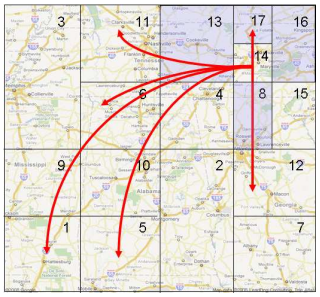

Fig. 2. Random replication scheme

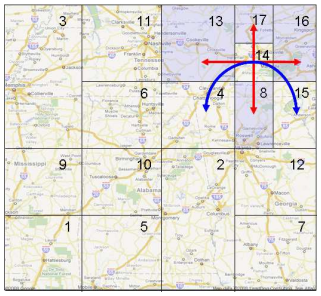

Fig. 3. Neighbor replication scheme

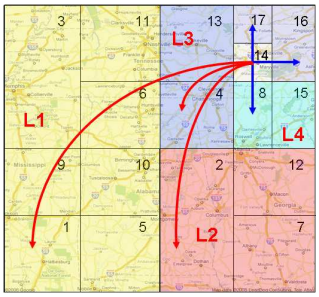

Fig. 4. Neighbor-shortcut replication scheme

is critical for any long lived large scale systems to maintain service availability.

B. Baseline Replication Methods

In order to understand the intrinsic factors impacting the design of an effective replication scheme in terms of benefit and cost, we analyze two basic replication methods – neighbor replication scheme and random replication scheme, each of which achieves some degree of failure resilience but suffers from either weak failure resilience or high replica maintenance overhead.

Random Replication Approach.

Random replication is a widely adopted replication method in distributed systems [3], [4]. Given an owner node and a replication factor rf, a random replication scheme will randomly select rf nodes as its replica hosts using a hashing function H. Fig. 2 shows an example in which node 14 randomly selects 6 nodes as its replica hosts. Because randomly selected replicas are often far away from the host node, random replication exhibits some natural resilience against both individual node failures and massive node failures. However, this approach incurs high replica maintenance cost. First, the random replica hosts may be far away from the owner node in terms of both network proximity and geographical proximity, thus this approach incurs much higher communication and synchronization overheads. Second, if the replica owner crashes, higher overheads in searching and migration are involved to restore the service in the new owner node.

Neighbor Replication Approach.

This replication scheme places replicas in the direct neighbors of the owner node or multi-hop neighbors if the number of direct neighbors is not sufficient to accomplish the replication requirements. The replication factor rf defines the number of neighboring nodes that act as the replica hosts, which represents the desired redundancy level of the service. If rf is relatively small, the number of direct adjacent neighbor nodes is sufficient to accomplish the replication requirement. In the case that rf is large, the number of direct neighboring nodes is insufficient, we will select replica hosts from multi-hop neighbors of the owner node. As shown in Fig. 3, the replicas hosts of node 14 consists of its direct neighbors: node 8, 13, 16, 17 and its neighbor's neighbors: node 4 and 15. By choosing replica hosts clustered around the owner node, this scheme greatly reduces synchronization and search overheads compared with the random replication scheme. However, it suffers from the relatively weak resilience to massive node failures. For example, when network partition occurs, if an executor node and its neighboring replica hosts are within the same network segment, then nodes outside this network segment will have no way to reach the location services hosted by this executor node or the service replicas located around this executor node, leading to the unavailability of the services.

C. Neighbor and Shortcut Replication Scheme

The design objectives of Reliable GeoGrid replication scheme is to provide durable location query maintenance, offer uninterrupted location query processing and enhance the partition tolerance capability. Directed by these objectives, we exploit a hybrid replica placement scheme by combining replication by *"neighboring nodes"* and replication by *"shortcut nodes"*. The former emphasizes that the replica placement should enable fast replica-based recovery and keep the replica maintenance cost low in the presence of high churn rates and node dynamics. The later promotes the use of shortcut nodes to reach GeoGrid regions that are far away from the region of the current node, which can greatly strengthen failure resilience against severe network partition failures. Next we describe our replica placement algorithm that chooses neighboring nodes and shortcut nodes as the replica hoarding destination. Finally, we present how Reliable GeoGrid replication scheme dynamically maintains rf invariant replicas in the presence of node departure or node failures.

Overview of Shortcut.

Similar to a path shorter than usual one in real world, *routing shortcut* trims the routing space and reduce the redundant routing hops through maintaining more routing information such as shortcuts to other other larger regions at each node such that these routing entries can be used as the shortcuts in forwarding routing requests. To build shortcut, in addition to the rectangular regions owned by each GeoGrid node, the entire geographical space is virtually partitioned into a sequence of larger regions such that each region is half size of the previous region in order and are not overlapping with each other, called *shortcut region*. Thus each node stores

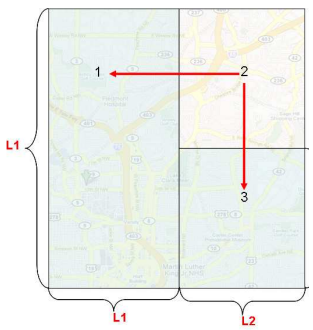

Fig. 5. Shortcut list initialization

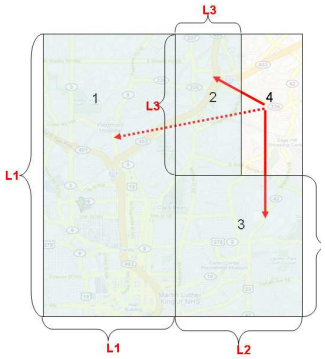

Fig. 6. Maintain obsolete neighbor link in joining the system

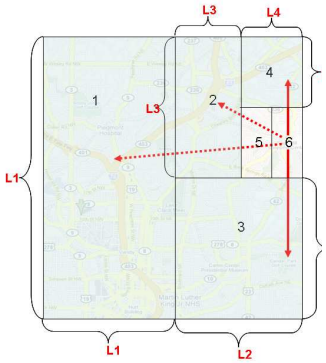

Fig. 7. Inherit shortcuts in splitting region

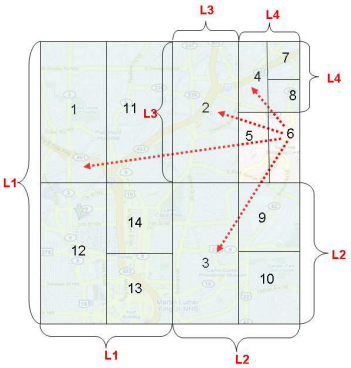

Fig. 8. Maintain obsolete neighbor links when other other nodes joining the system

the addresses of its immediate neighboring nodes but also addresses of one or more residents, *shortcut nodes* for each shortcut regions.

The shortcuts of a node p is organized into a list $L_p < s_1, s_2, \ldots, s_m >$, denoted by $ShortcutList(p)$. m is the number of shortcuts in the shortcut list of p. Each shortcut s_i points to a node in a geographical partition of $1/2^i$ the size of the geographical plane. There are no overlapping among the partitions pointed to by the shortcuts of p.

In Reliable GeoGrid, nodes may have their shortcut lists in different size. The exact length of the shortcut list L_p for a node p is determined by the relative size of the region R owned by p. When the region R is $1/2^m$ of the size of the geographical plane, the length of the shortcut list L_p is m. This allows the shortcut list of p to cover the entire geographical plane by the shortcuts of p according to the following equation: $\sum_{i=1}^{m} 1/2^i + 1/2^m = 1$. Based on this analysis, we can estimate the average length of the shortcut list maintained by each node. The size of a region in a GeoGrid of N regions is $\frac{1}{N}$ of the geographical plane, assuming a uniform region size distribution. Thus the length of the shortcut list maintained by each node can be estimated by $O(\log_2 N)$. As an eample, in Fig.4 node 14 maintains a shortcut pointing to node 1 which is not its direct neighbor. If node 14 is routing a request towards node 9, it can forward this request to node 1 directly which then forwards to node 9. Such routing path effectively trims the half search space, compared with the normal routing path passing node $8, 4, 6, 10$ to reach node 9.

Shortcut Construction and Maintenance.

The construction of shortcuts is a part of the topology construction algorithm. When a new node p joins and splits an existing region L into halves and inherits one half of the region L from the original owner of L, the shortcut list of this new node p is created in two steps. First, the new node inherits the shortcut list of the owner node of region L. Second, it examines the inherited neighbor list, and identifies those nodes whose regions are no longer its direct neighbors. The new node p will add these nodes into its shortcut list and then remove them from its neighbor list. For example, starting from a 3 nodes system visualized in Fig.5, node $1, 2, 3$ initialize their shortcut list as an empty list. As shown in Fig.6, as node 4 joins and splits node 2's region and inherits the neighbor list from node 2, node 1 is not new node 4's direct neighbor and an obsolete neighboring relationship occurs, illustrated by the dotted line in Fig.6. Instead of deleting the link pointing to node 1, node 4 adds this link into its shortcut list. Next, in Fig.7 when node 5 joins, it inherits the link pointing to node 1 from node 4 when it splits node 4's region. Node 6 inherits node 5's shortcut list when it splits node 5's region and at the same time, node 6 integrates the obsolete neighboring link pointing to node 2 as another shortcut. In Fig.8, as node 9 joins and splits node 3's region, node 6 integrates the obsolete link to node 3 to its shortcut list, and node 6 finally acquires the shortcut list which owns pointers to nodes residing in $L1, L2, L3, L4$, as shown by Fig.8. The maintenance of shortcuts, upon node join, departure, or failure, is similar to the maintenance of neighbor list. Heart beat messages are used to detect node failures or departures.

Replication Factor rf. Given a location query request LQ, let node p be either the owner node or executor node of LQ. LQ will be replicated at the following set of nodes:

$$ReplicationList(p, lq) = [(p_1), (p_2), \ldots, (p_{rf})], \text{ where}$$
$$\bigwedge_{k=1}^{rf} p_k \subset \{\text{NeighborList(p)} \cup \text{ShortcutList(p)}\}$$

This set is called the *Replication List* of node p, denoted by $ReplicationList(p)$. As a location query lq is issued and received by its owner node, it is replicated to $ReplicationList$ of the owner node p. The length of the replication list is defined by the *replication factor* rf, which is a tunable system supplied parameter, set in the system initialization time and continually tuned according to failure rate, throughput of the system, and the latency of the messages. Setting rf to a large value may cause the system to pay higher replica maintenance cost for fault tolerance and such cost can be further aggravated due to high churn rate of the overlay network or fast movement of hotspots in terms of request patterns of mobiles. Another

important design consideration is to keep the ratio of neighbor replica and shortcut replica to be relatively proportional and constant for each node p, because shortcut nodes are usually further away from node p and it is important to keep sufficient number of neighboring nodes as the replica hosts. In the situation where the rf value is large, and combining both the neighbor list and the shortcut list of a node p is insufficient to fulfill the replication requirement, i.e., size(NeighborList(p)) + size(ShortcutList(p)) $< rf$, we will extend $NeighborList$ to the multi-hop neighbor list which maintains the i-hop neighbor list of a node p for $i = 1, \ldots, k$. As shown in Fig.4, with $rf = 6$ and the minimum neighbor replica to be 50%, node 14 selects its 3 direct neighbors: node 8,16,17, and its 3 shortcut nodes: node 1,2, 4 to compose its replica list.

The dynamic replica management module maintains the rf replicas for each node in the network when node leaves or joins the network by monitoring the rf number of replica hosts in the $ReplicationList$ with the help of lower level Reliable GeoGrid operations such as periodic heartbeat messages. Due to the space constraint, we refer the readers to our technical report [5] for further details.

IV. Load Balancing Through Replication

An important challenge in scaling pervasive location service is the system-level capability in handling continuously changing hot spots in terms of service demands and access patterns of mobile users. In Reliable GeoGrid, we designed a dynamic utility-aware, replica-based load balancing scheme, which takes into account load balance relevant factors to exploit the service processing capabilities of replicas. For each newly arrived LQ, a utility value is computed for each replica host based on a weighted utility function and the node with the largest utility value is selected as the query executor of the LQ for load balance purpose. The selection of executor node for a LQ takes into account three factors that have critical impacts on load balance and system resource utilization. The first factor is the load per node, namely how much load does a replica host currently have. The second factor is the cache affinity factor, which states whether the data items interested by the location query is in the cache of the replica host. The third factor is the network proximity of the replica host to the remote data source that provides the data items of the given query. By taking into account the runtime load level of node p and its replica hosts, we can avoid the risk of offloading the LQ from the node p to another heavily loaded replica node and create unwanted query drops, at the same time we increase the probability of assigning the LQ to a node that has more resources and yet less loaded. By taking into account the cache affinity factor, the node with required cache items will be ranked higher in its load balancing utility value, thus we avoid repeated and blindly data fetching from the remote data source and effectively reduce the query processing overheads of the system. By considering the network proximity between the replica host node and the remote data source being queried, better system utilization is achieved.

More specifically, a system-initiated workload threshold parameter is used to control when the load balancing scheme should be turned on and how long the load adaptation should be. By monitoring workload level in runtime, each node can invoke the dynamic load balance scheme when it detects that its workload exceeds the "alert" threshold. Upon the detection of overload at a node, the load balance algorithm will be turned on and the overloaded node computes utility value for each replica host from its rf replicas by applying the utilization function which assigns different weights to the three factors. The overloaded node selects the replica host with largest utility value and passes (offloads) the location request to the newly selected replica host for execution. This load balancing process repeats until the specified number of epochs is reached or there exisits no significant load imbalance across nodes in the Reliable GeoGrid overlay network. Due to the space constraint, we refer the readers to our technical report [5] for further details.

V. Experimental Results

This section reports our experimental results for Reliable GeoGrid service network by simulating a geographical region of 64 miles $\times$ 64 miles. The population of end users in this region ranges from 1×10^3 to 1.6×10^4. For each population, we simulated 100 randomly generated Reliable GeoGrid networks. Each end user connects into the Reliable GeoGrid system through a dedicated proxy node. The capacities of those proxies follow a skewed distribution using a measurement study documented in [1]. We report two sets of experiments that evaluate the effectiveness of Reliable GeoGrid approach to scaling location service networks. We first study the fault tolerance of our replication scheme against individual node failures and massive node failures. Then we evaluate the effectiveness of our utility-aware, replica-based load balancing scheme.

A. Failure Resilience

There are two common types of node failures: individual node failure, and massive node failure. In the case of individual node failure, without replication, the LQs hosted by the failed node p will be lost, though the geographical region, for which the failed node is responsible, will be taken over by one of the neighbor nodes upon detecting the departure of node p. However, with our replication scheme, such individual node failure will ensure no interruption of the system operation at all since all LQs hosted by the failed node will be recovered by one of its rf replica nodes, assuming that not all rf replicas failed together. Otherwise, a critical failure will occur.

Fig. 9 and Fig. 10 plot the total number of critical failures captured during this experiment under different settings of mean service time (st), restoration time (rt) and replication factors (rf). We observe that the neighbor and shortcut replication scheme can help the system to significantly reduce the occurrence of critical failures. With larger replication factor, smaller restoration time and longer service time, we can achieve better reliability and incur less number of critical

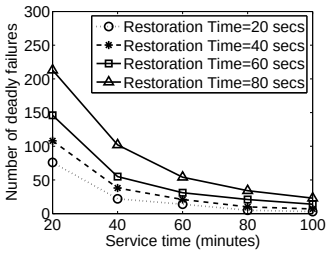

Fig. 9. Critical failures, $rf = 1$, system size=4000 nodes

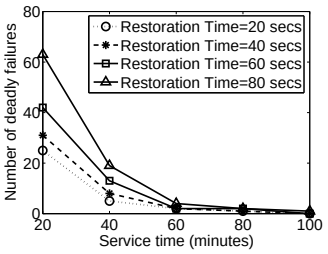

Fig. 10. Critical failures, $rf = 3$, system size=4000 nodes

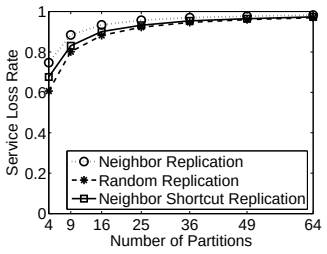

Fig. 11. Service Loss Rate in massive node failures (rf=2, system size=10000 nodes)

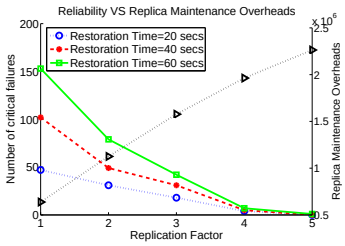

Fig. 12. Reliability VS Replica maintenance overheads (system size=4000 nodes, service time=20 mins)

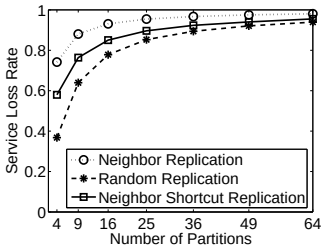

Fig. 13. Service Loss Rate in Network Partition Failures (rf=4,System Size=10000 nodes)

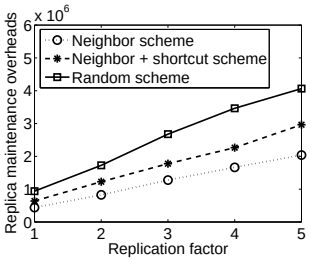

Fig. 14. Repliation overhead comparison on replication schemes (System Size=4000 nodes)

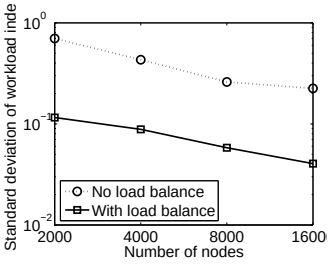

Fig. 15. Standard deviation of workload index (no hot spot) on rf=4

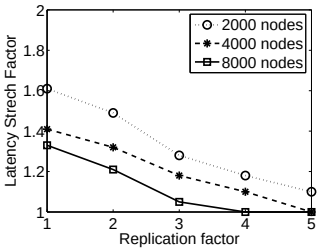

Fig. 16. Impact of replication factor (rf) on load balance

failures. As defined earlier, critical failure occurs when all replica hosts of a LQ fail within the restoration time interval Δt_r. In Fig. 10 a moderate replication factor 3 will reduce the number of critical failures to almost zero for system with service time more than 40 mins. This shows that the proposed replication scheme can significantly reduce the number of critical failures and achieves reasonable reliability through placing moderate number of replicas.

Everytime the system migrates a replica from one node to another node data movement overheads occur, we denote it as individual replica migration overheads, which is defined as $dataSize * communicationLatency$. Thus the system replica maintenance overheads is defined as the sum of all the individual replica migration overheads. Fig.12 shows the comparison between achieved reliability and replication maintenance overheads. As shown by the dotted line with hollow circle marker, dashed line with solid circle marker and solid line with square marker, higher reliability is achieved as rf increases. The dotted line with triangle marker shows how the replication maintenance overheads increases as the replication factor increments. For a system with 4000 nodes with service time around 20 mins (high node dynamics), replication with rf=4 introduces relatively low overheads while achieving good reliability.

Fig. 11 and Fig. 13 compare the failure resilience of the three replication schemes discussed earlier in the presence of network partition failures. We examine the cases with the number of network partitions ranging from 4 to 64 for a 10000 nodes network with 5000 random LQs. We use the *service loss rate* to measure the failure resilience in such scenario, which is defined as the percentage of unsuccessful LQs. Overall, we observe that as the number of overlay network partitions increases, the service loss rate increases in all three replication schemes, and the loss rate of random replication approach and the lost rate of the neighbor-shortcut replication approach start to converge. For an overlay network with 36 network partitions or higher, the random replication approach performs only slightly better than the neighbor-shortcut replication approach in terms of service lost rate. We also observe that the higher the replication factor rf is, the more effective our replication scheme performs in terms of the service loss rate reduction. Fig.14 compares the replication maintenance overheads among these three replication schemes. The result confirms that random replication incurs the most maintenance overheads and neighbor replication scheme introduces the least amount of maintenance traffic while neighbor and shortcut replication scheme is in the middle. Combining the failure resilience towards both individual node failure and network partition failure and the moderate maintenance overheads, we can conclude that neighbor and shortcut replication scheme takes the advantages of the two replication schemes while avoiding their weakness and can achieve reasonable good reliability through placing moderate number of replicas.

B. Evaluation of Load Balance Scheme

To evaluate the effectiveness of the proposed load balance scheme in dealing with continuously changing workload in pervasive computing environment, we built a discrete event simulator that models the allocation of the location query events to nodes. The workload is a mixture of regular location

query service requests and moving hot spot query requests.

Fig. 15 examines load balance scheme performed on regular uniform workload distribution (without hot spot) and we set rf as 4 and epoch length as 20 seconds with a total of 50000 regular location query requests. We observe that as the system size increases, the standard deviation values of the workload index decreases and the load balance scheme reduces the standard deviation of the workload index to almost 10 percent of the case without load balance. For a system without load balance scheme equipped, the occurence of hot spot may introduce longer service latency because of the longer waiting queues. To study how well the load balance scheme helps to reduce the query latency when using load balance scheme, we define "latency stretch factor" as the ratio between the average latency in the case without load balance scheme activated and the average latency when the replica based load balance scheme is activated. In other words, higher latency stretch factor indicates higher query latency caused by the hot spot. Fig.16 shows the experimental results in different system size. We can observe that a replication factor 4 for 8000 nodes system and replication factor 5 for 4000 nodes system can eliminate the extra query latency caused by hot spot and reduces the query latency to the same scale as the case without hot spot, indicating by query latency stretch factor equivalent to 1. This shows that the replica based load balance scheme greatly helps the system to reduce the query latency and improve the system performance even when hot spot occurs.

VI. Related Work

There have been an intensive research effort on serving location based services and applications in mobile environments through an overlay network or a network of caches [6], [7]. However, most of these research works focus on improving the scalability and proximity awareness of location based services by utilizing its decentralized, self-managing features. Only few research has been focusing on improving the failure resilience of overlay network related location based service.

Existing research on replication scheme for overlay network can be classified into two categories: random replication and cluster replication. Random replication scheme is widely used in distributed files systems, autonomous replication [8]. Cluster replication places the replicas around the owner node. The most representative examples are the neighbor replication in Chord [9].

Most of existing load balance schemes [10] and [11] focus on random workload reassignment among nodes which incurs high overheads while reduces the burden of some heavily loaded nodes. Although these solutions can reduce the workload from heavily loaded nodes, it requires to maintain a few centralized directory nodes and each node needs to keep the identification information of all the directory nodes. Furthermore, the selection of the load shedding node fails to consider the communication bandwidth and promixity distance factor which may also lead to high communication overheads.

VII. Conclusion

We have presented Reliable GeoGrid, a location-based service overlay network for scaling location based services and enhancing reliability of pervasive computing applications. This paper makes three original contributions. First, we developed a methodical approach to building a reliable and scalable location service network with neighbor-shortcut based replications. Second, we develop a dynamic replica-based load balancing scheme with an utility-aware model, which takes into account of node heterogeneity, network proximity, and changing workload at each node to scale the system in the presence of unexpected workload changes and node failures. Third but not the least, our prototype and experimental study demonstrate that the Reliable GeoGrid framework is highly scalable in terms of changing hotspots and highly reliable in the presence of both node failures and network partition failures.

Acknowledgment

This work is partially supported by grants from NSF CyberTrust program, NSF CNS program, IBM faculty award, IBM Sur grant, and a grant from Intel.

References

[1] S. Saroiu, P. Gummadi, and S. Gribble, "A measurement study of Peer-to-Peer file sharing systems," in *Proceedings of MMCN*, San Jose, CA, August 2002.

[2] R. D. Schlichting and F. B. Schneider, "Fail-stop processors: An approach to designing fault-tolerant computing systems," *Computer Systems*, vol. 1, no. 3, pp. 222–238, 1983.

[3] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *ICS '02: Proceedings of the 16th international conference on Supercomputing*. New York, NY, USA: ACM, 2002, pp. 84–95.

[4] A. Ghodsi, L. O. Alima, and S. Haridi, "Symmetric replication for structured peer-to-peer systems," in *DBISP2P*, 2005, pp. 74–85.

[5] G. Zhang and L. Liu, "Reliable geogrid: Scaling location service network through decentralized replication," Georgia Institute of Technology, Research Report, 2008.

[6] D. Spence, J. Crowcroft, S. Hand, and T. Harris, "Location based placement of whole distributed systems," in *CoNEXT '05: Proceedings of the 2005 ACM conference on Emerging network experiment and technology*. New York, NY, USA: ACM, 2005, pp. 124–134.

[7] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney, "Giggle: a framework for constructing scalable replica location services," in *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 1–17.

[8] F. M. Cuenca-Acuna, R. P. Martin, and T. D. Nguyen, "Autonomous Replication for High Availability in Unstructured P2P Systems," in *The 22nd IEEE Symposium on Reliable Distributed Systems (SRDS-22)*. IEEE Press, Oct. 2003.

[9] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building peer-to-peer systems with Chord, a distributed lookup service," 2001, pp. 81–86.

[10] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in structured p2p systems," 2003.

[11] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load balancing in dynamic structured P2P systems," in *Proc. IEEE INFOCOM*, Hong Kong, 2004.