A Scalable Middleware for Context-aware Mobile Applications

Loris Belcastro

DIMES, University of Calabria,
Rende, Italy
E-mail: lbelcastro@dimes.unical.it

Fabrizio Marozzo*

DIMES, University of Calabria,
Rende, Italy
E-mail: fmarozzo@dimes.unical.it
*Corresponding author

Paolo Trunfio

DIMES, University of Calabria,
Rende, Italy
E-mail: trunfio@dimes.unical.it

Abstract: A core functionality of context-aware mobile applications is storing, indexing, and retrieving information about users, places, events and other resources. The goal of this work is to design and provide a service-oriented middleware, called Geocon, which can be used by mobile developers to implement such functionality. To represent information about users, places, events and resources of context-aware applications, Geocon defines a metadata model that can be extended to match specific application requirements. The middleware includes a *geocon-service* for storing, searching and selecting metadata about users, resources, events and places of interest, and a *geocon-client* library that allows mobile applications to interact with the service through the invocation of local methods. The paper describes the Geocon middleware and presents an experimental evaluation of its scalability on a cloud platform with a real-world mobile application.

Keywords: Context-aware; Mobile applications; Middleware; Scalability; Cloud computing.

1 Introduction

In the last ten years, mobile applications became an essential part of human life as they provide a wide range of tools for communication, productivity, entertainment, social networking and many other purposes. The large number of mobile applications being constantly released and the ever increasing power of mobile devices (smartphones, tablets), have significantly improved the user experience in many fields that just a few years ago were monopolized by standard web applications (e.g., e-commerce, traveling services, online games, etc.).

The advent of location-based services, which provide ubiquitous access to context-aware information, has given a great impulse to the development of a new generation of mobile applications. Mobile context-aware computing is a paradigm in which mobile applications can discover and take advantage of contextual information (e.g., date and time, user position, nearby users) [1, 2, 3]. Some examples of context-aware mobile applications are: interactive trolleys to help shoppers finding groceries [4], monitors to remind medication for elderly [5], location-aware telephone call forwarding [6], and targeted advertisement based on social group information [7].

A core functionality of any context-aware ubiquitous system is storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). The goal of this work is to design and provide a service-oriented middleware, called *Geocon*[1], which can be used by mobile application developers to implement such functionality. Geocon can be used to discover location-aware content, to share context-related information, and to facilitate interaction among users of mobile apps. Some examples of services that can be implemented in a mobile app using Geocon are: *i*) discovery of cultural places to be visited during a trip; *ii*) publication of user reviews about hotels and restaurants; *iii*) find nearby free-time activities

[1] `https://github.com/SCAlabUnical/Geocon`

of a user and his/her friends; *iv*) sharing of real-time information about events, traffic, etc.

A key benefit for developers using Geocon is the possibility to focus on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data storing, indexing and searching, since they are provided by the middleware. In order to represent information about users, places, events and resources of mobile context-aware applications, Geocon defines a metadata model that can be extended to match most application requirements [8]. The widely-used JavaScript Object Notation (JSON) format is employed to represent such metadata. The architecture of the middleware includes a *geocon-service* that exposes methods for storing, searching and selecting metadata about users, resources, events and places of interest, and a *geocon-client* library that allows mobile applications to interact with the service through the invocation of local methods. The interaction between service and client is based on the REST model.

Given the huge number of users, places, events and resources that may be involved in context-aware ubiquitous applications, scalability plays a fundamental role [9]. Geocon was designed to ensure scalability through the use of a NoSQL indexing and search engine, Elasticsearch [10], that can scale horizontally on multiple nodes as the system load increases.

Compared to related work, Geocon is the only publicly available (and open source) Cloud-oriented system that provides a scalable middleware for context-aware mobile applications. Geocon was designed to be deployed on a public/private Cloud infrastructure, thus allowing an elastic resource allocation in a pay-per-use manner [11].

The scalability of Geocon is the results of three main elements: *i*) a scalable software architecture, which includes a *geocon-service* natively designed to support replication on multiple processing nodes; *ii*) a scalable data indexing and searching service provided by Elasticsearch; *iii*) the use of a scalable cloud infrastructure, which provides auto-scaling and load balancing to run Geocon and Elasticsearch services. To assess the scalability of Geocon in a real case scenario, we used it to develop a location-aware mobile application, called *GeoconView*. The application allows users to share information about events exploiting the Geocon middleware for storing, indexing, and retrieving such information.

We evaluated the performance of Geocon varying the number of Cloud machines used to run the Geocon software components, the number of events stored in the middleware, and the number of queries submitted to the system. The experimental results that will be described later in detail, show that the latency speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the metadata model. Section 4 describes the middleware architecture and components. Section 5 describes the performance experiments carried out to assess the scalability of Geocon. Finally, Section 6 concludes the paper.

2 Related Work

Several research projects and software systems have been proposed to support the implementation of context-aware mobile applications.

CRUMPET [12] (*Creation of User-Friendly Mobile Services Personalised for Tourism*) was a European research project aimed to deal with issues related to the mobility of tourists. In particular, the system provides tourists with information filtered by mobile users' positions and interests. Yu and Chang [22] extended CRUMPET to overcome the limitations of handheld devices regarding the screen size and transmitting bandwidth.

The COMPASS system [13] (*Context-aware Mobile Personal Assistant*) was developed to provide users with relevant information and services. The relevance is determined exploiting information extracted from the user profile (e.g., preferences, interests, locations visited). COMPASS uses two search criteria for selecting relevant services/information: *i*) strict criteria, for discarding irrelevant results; and *ii*) soft criteria, for sorting and assigning a score to remaining results.

Driver and Clarke [14] proposed a framework to support the development of mobile trails-based applications. A trail is a scheduled collection of activities, such as to-do lists, that can be properly reordered when context change. The framework supports context-based activity schedule composition, identification of whether or not schedule reordering is required following context change and subsequent automatic schedule reordering as appropriate.

MobiSoC [15] is a service-oriented middleware for capturing, managing, and sharing the social state of physical communities. This state is composed of people profiles, place profiles, people-to-people affinities, and people-to-places affinities. The middleware provides real-time recommendations about people, places, and events, and delivers customized information based on users' geo-social context. The latency time of MobiSoC has been evaluated varying the query type and the number of users, showing the limitation of having a fixed number of computing nodes.

Context Toolkit [16] is a framework designed to support the development of context-aware applications. It consists in a set of widgets, which are software components with a common interface used to separate applications from context acquisition issues. The toolkit provides developers different components responsible for acquiring, aggregating and interpreting context information.

CaMWAF [17] is a framework designed to support the development of context-aware mobile applications

Table 1 Comparison with related systems.

System	Goal	Cloud	Scalable	Publicly available
CRUMPET [12]	Content filtering based on mobile user's position	No	No	No
COMPASS [13]	Services and information filtering on user's preferences and interests	No	No	No
Driver and Clarke [14]	Context-aware management of user activities	No	No	No
MobiSoC [15]	Middleware for developing mobile social applications	No	Yes	Yes
Context Toolkit [16]	Composition of widgets for accessing context information	No	No	Yes
CaMWAF [17]	Context-aware applications using HTML5, CSS3 and JS	No	No	No
Malcher et al. [18]	Client middleware for local and remote data exchange	No	No	No
SALES [19]	Contextual data dissemination in heterogeneous wireless networks	No	Yes	No
CARISMA [20]	Context-aware applications exploiting reflection and micro-economic policies	No	Yes	No
EgoSpaces [21]	Agent-based middleware for applications in ad-hoc mobile environments	No	No	Yes
Geocon	Scalable middleware for context-aware mobile applications	Yes	Yes	Yes

and simplify the exchange of context information in heterogeneous environments. It allows developers to easily create cross-platform context-aware applications using common web technologies (e.g., HTML5, CSS3, JavaScript). To deal with the resource limitation of mobile devices, CaMWAF delegates the execution of intensive tasks to the server.

Malcher et al. [18] proposed a client middleware for developing context- and location-aware applications with capabilities of data sharing, dynamic deployment of new components, and combination of basic collaboration services. Given its client-side approach, the middleware does not take into account the server side architecture and related scalability issues.

SALES [19] is a middleware for contextual data dissemination in heterogeneous wireless communication networks. It proposes a hierarchical distributed architecture, some caching techniques for reducing context data traffic (e.g., a locality-based policy to speed up accesses to context data strictly related with locality), and two models for representing data (i.e., key-value and object-based model). Concerning data representation, the key-value model allows reducing management overhead, especially in terms of required bandwidth, while the object-oriented model facilitates development by supporting extendibility.

CARISMA [20] is a mobile computing middleware that exploits the reflection principle for enhancing the development of adaptive and context-aware mobile applications. It provides developers with a set of primitives for describing how context changes should be handled using policies. Such policies use a micro-economic approach, which relies on a particular type of sealed-bid auction to take decision during application execution.

EgoSpaces [21] is an agent-based middleware for developing applications in ad-hoc mobile environments. It proposes an agent-centered notion of context, called a view, which is a collection of relevant data (or context). Each agent can operate over multiple views (which can be redefined over time as needs change) that include data/resources associated with the agent.

Table 1 summarizes the main features of the related systems discussed above, in comparison with Geocon's features. For each system, the table indicates: (*i*) the main goal of the system; (*ii*) whether or not the system was designed to be deployed on the Cloud; (*iii*) whether or not the server side focuses on scalability; (*iv*) whether or not the system is publicly available. In particular, the second feature ("Cloud") is important as it allows mobile application developers to know whether a system can be natively deployed on a Cloud. In fact, mobile context-aware applications store, index, and retrieve information about entities very often on Cloud storage services and only rarely on on-promises private services (e.g., private servers, private clusters).

As shown in the table, Geocon is explicitly designed for a Cloud, with a set of back-end components for data storing, indexing and searching, that can be easily deployed on any public/private Cloud infrastructure. Moreover, Geocon provides ad hoc scalability mechanisms, which are fundamental to provide satisfactory services as the amount of users and/or data to be managed grow. The test results demonstrate that Geocon scales well, thus allowing the development of mobile applications with a large number of users. It is worth noticing that three related works ([15] [19] [20]) highlight the importance of the scalability problem, but do not provide experimental evaluations on this aspect. It is also important to point out that context-aware systems that are not natively designed for the Cloud, may be ported to the Cloud, but this does not mean that they become automatically scalable. In particular, when the workload grows and can only be managed efficiently using multiple servers (scale out), legacy systems need to be fully redesigned to support this kind of scalability. In contrast, Geocon is natively able to scale its workload on multiple servers since its software architecture has been designed to scale out using Cloud facilities (for more details, see Section 4.2).

In summary, Geocon is the only publicly available (and open source) Cloud-oriented system that provides a scalable middleware for context-aware mobile applications. Another important added value, not highlighted in the table, is the methodology provided by Geocon that defines an expendable metadata model supported by scalable set of back-end components for data storing, indexing and searching. This allows developers to focus on the front-end functionality provided by their mobile applications, without worrying on low-level back-end aspects and scalability issues that are managed transparently by Geocon.

3 Metadata Model

We defined a metadata model for representing information about users, places, events and resources of mobile context-aware applications. The model identifies a number of categories for indexing items in the domain of interest, which are generic enough to satisfy most of the application contexts. In particular, the metadata model is divided into four categories:

- *User*: defines basic information about a user (e.g., name, surname, e-mail).
- *Place*: describes a place of interest (e.g., square, restaurant, airport), including its geographical coordinates.
- *Event*: describes an event (e.g., concert, exhibition, conference), with information about time and location.
- *Resource*: defines a resource (e.g., photo, video, web site, web service) associated to a given place and/or event, including its Uniform Resource Identifier (URI).

Tables 2-5 present the basic metadata fields for each of the four categories listed above. Metadata are meant to be extensible, i.e., it is possible to include additional fields based on the specific application. For example, the user schema may be extended to include birth date, city, linked social network accounts, etc.

To represent metadata, the *JavaScript Object Notation* (JSON) is used. JSON is a widely-used text format for the serialization of structured data that is derived from the object literals of JavaScript [23]. Figure 1 shows an example of JSON metadata describing a User. Beyond the basic metadata (id, name, etc.), it includes some additional fields (city, linked accounts and food preferences).

Figure 2 shows an example of Place metadata, regarding the "Kabuki" restaurant in Washington, DC, USA, which is tagged as a Japanese and sushi specialties restaurant using an additional "tags" field.

Table 2 Basic User metadata.

Name	Type	Description
id	String	Unique user identifier
name	String	Given name
surname	String	Family name
email	String	E-mail
token	String	Authentication token

Table 3 Basic Place metadata.

Name	Type	Description
id	String	Unique place identifier
name	String	Name of the place
description	String	Textual description of the place
latitude	Real	Latitude of the place
longitude	Real	Longitude of the place
address	String	Full address of the place
user_id	String	Id of the user who created the place

Table 4 Basic Event metadata.

Name	Type	Description
id	String	Unique event identifier
name	String	Name of the event
description	String	Textual description of the event
start_date	String	Date and time when the event begins
end_date	String	Date and time when the event ends
place_id	String	Id of the place where the event is held
user_id	String	Id of the user who created the event

Table 5 Basic Resource metadata.

Name	Type	Description
id	String	Unique resource identifier
name	String	Name of the resource
description	String	Textual description of the resource
URI	String	Link to the resource
place_id	String	Id of the place to which the resource is associated
event_id	String	Id of the event to which the resourceis associated
user_id	String	Id of the user who created the resource

```
{ "id": "jdoe",
  "name": "John",
  "surname": "Doe",
  "email": "john.doe@example.com",
  "token": "19800308",
  "city": "New York, NY, USA",
  "linked-accounts": [
    {"name":"facebook", "token":"424911363"},
    {"name":"google", "key":"23467223454"}
  ],
  "food-preferences": ["sushi", "pizza"],
  "date-created": "2016-03-27T08:05:43.511Z"}
```


Figure 1 Example of User metadata in JSON.


```
{ "id": "534",
  "name": "Kabuki",
  "description": "Japanese Restaurant",
  "latidude": "38.897683",
  "longitude": "-77.006081",
  "address": "Union Station 50, Washington, DC, USA",
  "user_id": "jdoe",
  "tags": ["Japanese", "sushi"]}
```


Figure 2 Example of Place metadata in JSON.

4 Middleware

This section describes the software components of the Geocon middleware and how these components are deployed within a distributed architecture.

4.1 Software components

Figure 3 shows the software structure of the middleware, which includes two main components:

- *geocon-service*, which contains a central registry for indexing users, resources, events and places of interest, and exposes methods for storing, searching and selecting metadata about these entities.
- *geocon-client*, which is a client-side library allowing mobile applications to interact with *geocon-service* through the invocation of local methods.

The interaction between service and client is based on the REST model [24]. To this end, a complete support to *CRUD* (Create, Read, Update, and Delete) operations on the metadata has been defined through Java APIs.

4.1.1 Geocon-service

The geocon-service has been implemented as a JSON RESTful Web service application developed using *Jersey*[2]. Jersey is a Java open source framework that uses annotations to map a Java class to a Web resource, and natively supports JSON representations through

[2] http://jersey.java.net/

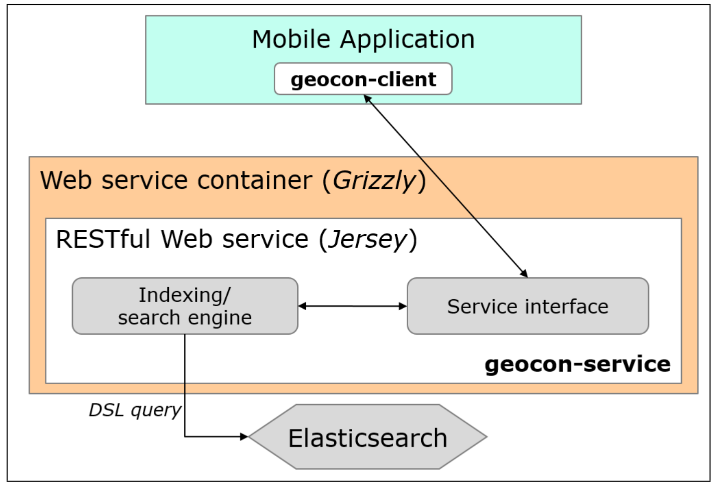


Figure 3 Software components of the Geocon middleware.

the integrated library *Jackson*[3]. The Jersey Web service application has been exposed using the Grizzly container[4].

The core component of *geocon-service* is the indexing and search engine, which has been implemented using Elasticsearch[5]. Elasticsearch is an open-source, distributed, scalable, and highly available search server based on Apache Lucene[6], and provides a RESTful web interface. Elasticsearch has been chosen because of several benefits, including:

- it is document-oriented, which means that entities can be structured as JSON documents;
- it is schema-free, which means it is able to detect the data structure automatically without need to specify a schema before indexing documents;
- it is horizontally scalable: if more power is needed, other nodes can be added and Elasticsearch will reconfigure itself automatically;
- it has APIs for several programming languages, including Java, which makes it easily integrable with other systems.

Geocon-service uses the query language provided by Elasticsearch, which is a full Query DSL (*Domain Specific Language*) based on JSON. Therefore, queries can be defined through the following main commands:

- *term*: returns all the documents whose specified field contains a given term. The following example returns all the documents whose field *name* contains the word “Mary”:

```
{"term" : { "name" : "Mary" }}
```

[3] http://jackson.codehaus.org/
[4] https://grizzly.java.net
[5] https://www.elastic.co/
[6] https://lucene.apache.org/

- *prefix*: returns all the documents whose specified field contains a term beginning with a given prefix. The following example returns all the documents whose field *surname* begins with “Ro”:

```
{"prefix" : { "surname" : "Ro" }}
```

- *bool*: returns all the documents containing a boolean combination of queries. It is built using one or more boolean clauses (i.e., *must*, *must_not*, *should*, and the parameter *minimum_should_match* that is the minimum number of clauses to be met). The following example returns all the users whose *name* is “Mary”, that are not between 10 and 20 years old, and that like eating sushi or pizza:

```
{"bool" : {
  "must" : { "term" : { "name" : "Mary" } },
  "must_not" : {
    "range" : {"age" : {"from": 10, "to": 20}}
  },
  "should" : [
    {"term" : {"food-preferences" : "sushi"}},
    {"term" : {"food-preferences" : "pizza"}}
  ],
  "minimum_should_match" : 1
  }}
```

- *filter*: returns all the documents filtered according to a given condition. The following example returns all the documents whose field *location* falls within 50km from the center of Los Angeles sorting by distance.

```
{"query": {
    "filtered" : {
      "filter" : {
        "geo_distance" : { "distance" : "50km",
        "location" : {"lat" : 34.052235,
          "lon" : -118.243683
        }}}}},
  "sort": [{
    "_geo_distance": {
      "location": { "lat" : 34.052235,
        "lon" : -118.243683},
      "order": "asc", "unit": "m",
        "distance_type": "plane"}}]
}
```

4.1.2 Geocon-client

Geocon-client is the library used by mobile applications to interact with *geocon-service*. The library aims to facilitate communication with the *geocon-service* methods, hiding some low-level details (e.g., authentication, REST invocation, etc.) and providing users with a complete set of functions for executing CRUD operations. These functions are implemented using a set of objects and methods provided by the client library to the application layer.

Geocon-client consists of five classes: four classes are used to represent the metadata categories (*User*, *Place*, *Event* and *Resource*), while a fifth class (*SearchEngine*) is used to expose the methods for storing and searching data on *geocon-service*. For each class representing a metadata category, the *SearchEngine* class provides a set of CRUD methods: *register*, *get*, *update*, and *delete*. As an example, Table 6 shows the CRUD methods provided to register, get, update and delete Resource elements in the service.

Table 6 CRUD methods for Resource elements.

Method	**Description**
`register(Resource r)`	Registers a resource to the service
`get (Resource r)`	Returns the metadata of a resource
`update (Resource r)`	Updates the metadata of a resource
`delete (Resource r)`	Deletes a resource

4.2 Distributed architecture

This section describes how the Geocon components are deployed within a distributed architecture. As described above, Geocon exploits Elasticsearch (ES) as indexing and search engine that can scale horizontally on a very large number of nodes as the system load increases. ES implements a clustered architecture that uses partitioning to distribute data across multiple nodes, and replication to provide high availability.

There are three types of ES nodes: *i*) *ES Data nodes*, which can hold one or more partitions containing index data; *ii*) *ES Client nodes*, that do not hold index data but handle incoming requests made by client applications to the appropriate data node; and *iii*) *ES Master node* that performs cluster management operations, such as maintaining routing information, coordinating recovery after node failure, relocating data partitions among nodes.

As shown in Figure 4, four types of nodes are present in the distributed Geocon architecture:

- *Mobile devices*, which interact with the geocon-service using the geocon-client library.
- *Load Balancer*, which evenly distributes requests from mobile devices to a pool of server nodes. Even though it is represented as a single node, it is actually implemented as a redundant Cloud service. This fact, coupled with the lightweight task it performs, fully prevents the Load Balancer to become a bottleneck for the system.
- *Server nodes*, which are a pool of virtual machines handling the mobile devices’ requests. Each request is managed by a geocon-service instance that translates it into an Elasticsearch query. The query is processed by an ES Client node that interacts with the appropriate ES Data nodes. One of Server nodes hosts the ES Master node.

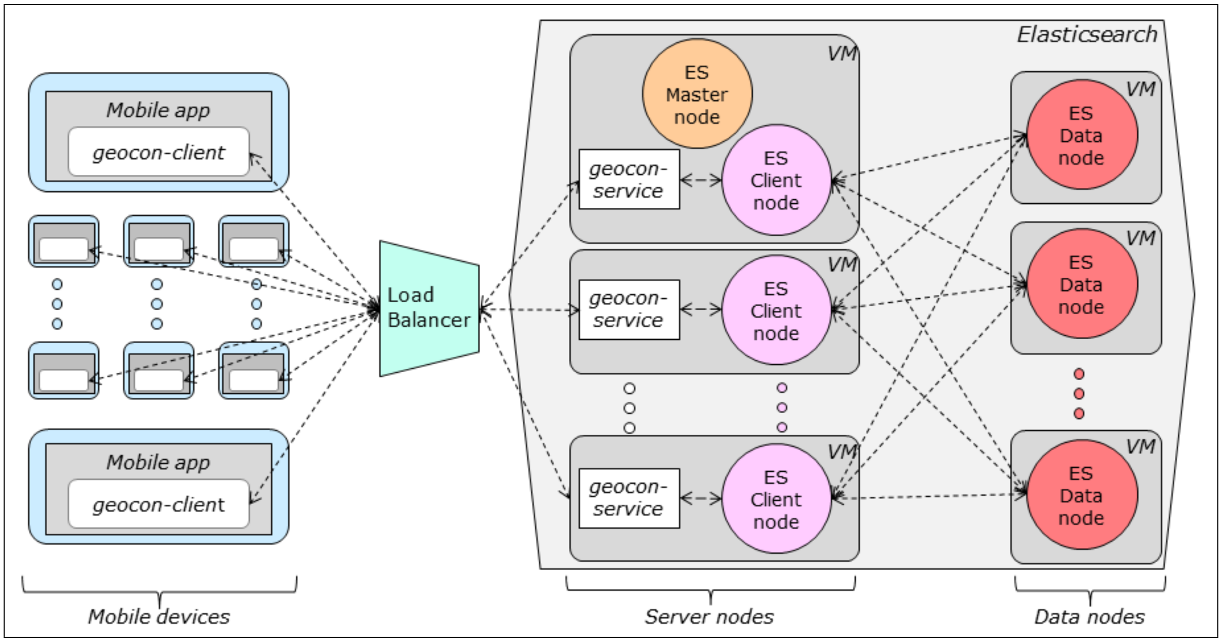


Figure 4 Distributed architecture of the Geocon middleware.

- *Data nodes*, which are a pool of virtual machines running the ES Data nodes, which process the queries upon request of the ES Client nodes.

5 Performance evaluation

An experimental performance evaluation was carried out to assess the scalability of the Geocon middleware in a real case scenario. The section is structured into three parts: *i*) introduction to the mobile application used as real case; *ii*) description of experimental setup and performance parameters; *iii*) presentation and discussion of the performance results.

5.1 Mobile application

In order to assess the scalability of Geocon in a real case scenario, we developed a location-aware mobile application, called *GeoconView*, which allows users to share information about events, exploiting the Geocon middleware for storing, indexing, and retrieving such information. More in detail, a GeoconView user can share events characterized by the following features: (*i*) *place*: the place where the event will happen; (*ii*) *images*: one or more photos representing the event; (*iii*) *datetimes*: the range of dates and times when the event will occur; (*iv*) *tags*: a set of keywords describing the event; and (*v*) *comments*: user comments and ratings about the event.

Figure 5 shows some screenshots of GeoconView. In particular, Figure 5(a) visualizes a number of GeoconView's events on a map. Figure 5(b) shows a preview of the *Arco Magno's sunset*, a daily event that occurs on a beach in San Nicola Arcella (Italy). Figure 5(c) provides full details about this event. Figure 5(d) shows a second event describing the arrival of grey herons in Tarsia (Italy) in December.

To implement the GeoconView application, the basic Event metadata scheme has been extended with additional fields to store URLs of images, descriptive tags, and information about the periodicity (e.g., weekly, monthly, yearly) of the events. Comments and ratings associated to the events have been stored as Resource metadata instances.

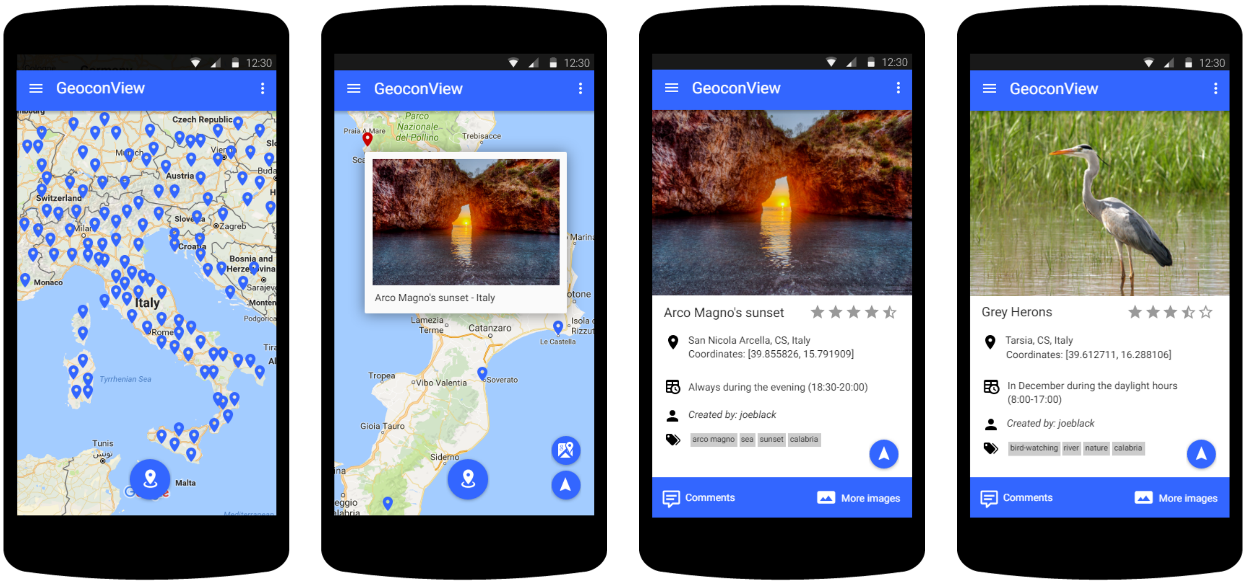
5.2 Experimental setup and performance parameters

The distributed architecture used for the evaluation is composed of 9 cloud machines hosted by the Microsoft Azure platform [25], each one equipped with a single-core 1.66 GHz CPU, 3.5 GB of memory, and 50 GB of disk space.

Table 7 shows the system parameters that have been used during the evaluation. As shown in the table, we used from 1 to 8 Data nodes, each one running on a separate cloud machine. An additional cloud machine was used to run a Server node. The number of geo-located events stored in the Geocon middleware ranges from 250k to 2000k. A varying number of queries per second (from 15 to 120) was submitted to the system, so as to evaluate its performance under different load levels. Every query asks for the ten active events that are closest to a location that changes randomly from query to query.

Table 7 System parameters.

Description	**Values**
Number of data nodes	1, 2, 4, 8
Number of events	250k, 500k, 1000k, 2000k
Queries per second	15q/s, 30q/s, 60q/s, 120q/s



(a) Geotagged events on a map. (b) Arco Magno's sunset (preview). (c) Arco Magno's sunset (details). (d) Grey herons arrival (details).

Figure 5 GeoconView: A location-aware mobile application based on the Geocon middleware.

For executing the load tests and measuring the performance of the system we used Apache JMeter[7]. According to the evaluation approach proposed in [26], the following performance parameters have been considered:

- *Latency time*: the average amount of time elapsed from query submission to query answer;
- *Speed-up*: the ratio of the latency time using 1 data node to the latency time using n data nodes, which indicates how much performance gain is obtained by distributing data over an increasing number of cloud machines;
- *Scale-up*: the latency time when the problem size is increased linearly with the number of data nodes, which measures the capability of the system to manage increasing loads when machines are added to accommodate that growth.

5.3 Performance results

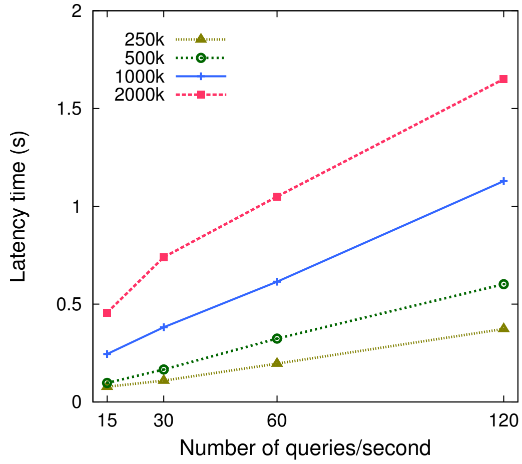
Figure 6 shows how the latency time changes using a fixed number of data nodes and varying the number of events stored and the number of queries per second submitted to the system. Figure 6(a) presents the results obtained using 2 data nodes. For the smallest number of events (250k), the latency time increases from 0.078 seconds with 15q/s, to 0.373 seconds with 120 q/s. For the largest number of events (2000k) the latency time ranges from 0.457 seconds to 1.651 seconds. Figure 6(b) shows the results obtained using 8 data nodes. For 250k events, the latency time ranges from 0.063 seconds with 15q/s, to 0.300 seconds with 120 q/s, while for 2000k events the latency time increases from 0.168 seconds to 0.621 seconds. In both cases, the latency time increases linearly with the number of requests per second, independently from the number of events stored in the system.

The scalability of Geocon can be evaluated through Figure 7, which shows the speedup obtained varying the number of data nodes and the number of queries per second submitted to the system.

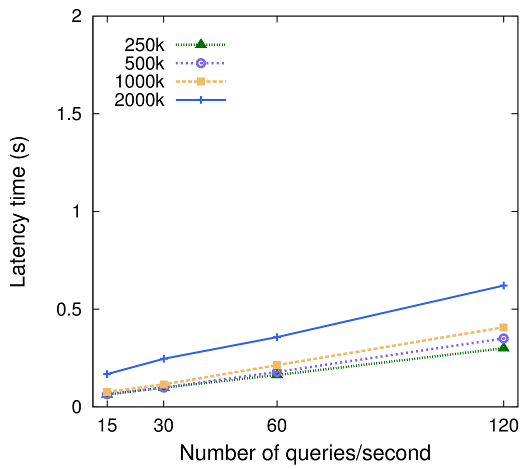
Figure 7(a) presents the results obtained with 500k events stored in the system. With a load of 15 queries per second, the speedup increases from 1.54 using 2 data nodes, to 2.29 using 8 data nodes. With the highest load (120 q/s) the speedup passes from 1.62 using 2 nodes, to 2.79 using 8 nodes. Figure 7(b) shows the results obtained when the number of events is increased to 2000k. With 15 queries per second, the speedup passes from 1.74 using 2 data nodes, to 4.74 using 8 data nodes. With 120 queries per second, the speedup passes from 1.45 using 2 nodes, to 3.86 using 8 nodes. As expected, the speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events (e.g., 2000k vs 500k).

Figure 8 measures the application scaleup by showing the latency times obtained when the number of events stored in the system increases proportionally to the number of data nodes used (i.e., from 250k events stored on 1 data node, to 2000k events stored on 8 data nodes). The results show that, for any number of queries per second submitted to the system, the latency time is almost constant. These results demonstrate that the system can manage an amount of data that increases almost linearly with available data nodes.

[7] http://jmeter.apache.org/

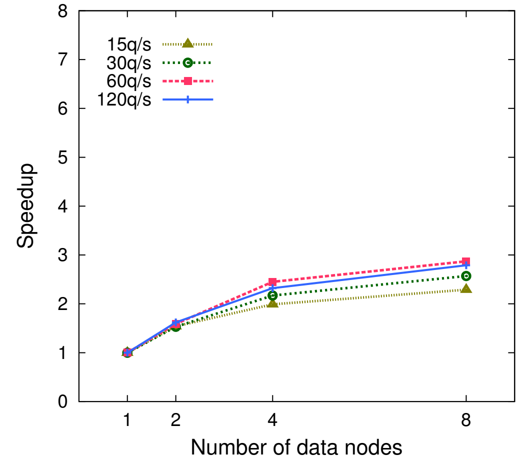


(a)

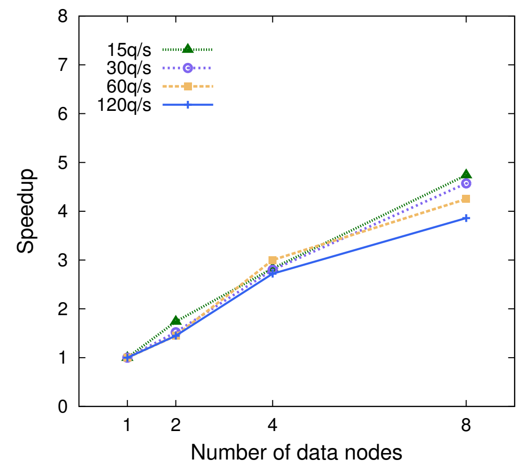


(b)

Figure 6 Latency time vs number of queries per second, for different numbers of events stored in the system, using: a) 2 data nodes; b) 8 data nodes.



(a)



(b)

Figure 7 Speedup vs number of data nodes, for different numbers of queries per second, using: a) 500k events; b) 2000k events.

6 Conclusions

The increasing power of mobile devices and the huge number of mobile applications available today have significantly improved the mobile users' experience. In particular, the advent of location-based services has given a great impulse to the development of context-aware mobile applications.

Geocon is a service-oriented middleware designed to help developers to implement context-aware mobile applications. Geocon provides a service and a client library for storing, indexing, and retrieving information about entities that are commonly involved in these scenarios, such as (mobile) users, places, events and other resources (e.g., photos, media, comments). A key benefit for developers using Geocon is the possibility to focus on the front-end functionality provided by their mobile application, without the need of implementing by scratch back-end components for data management and querying, which are provided by the middleware.

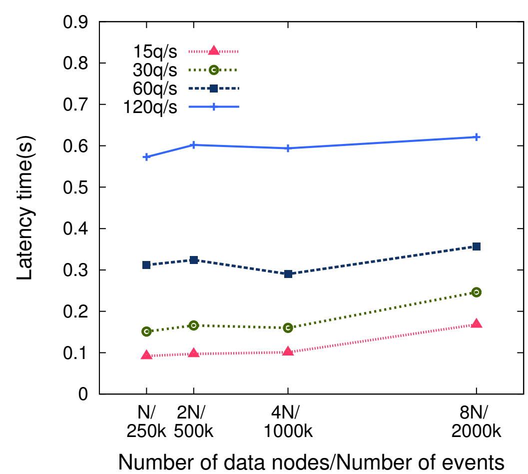


Figure 8 Latency time vs number of data nodes/number of events (scaleup), for different numbers of queries per second.

Geocon defines a metadata model to represent information about users, places, events and resources of mobile context-aware applications, which can be easily extended to match specific application requirements. In order to ensure a high level of decoupling and efficient communication between client and service, the REST model has been adopted. Moreover, given the huge number of users, places, events and resources that may be involved in context-aware mobile applications, Geocon uses the Elasticsearch engine that can scale horizontally on a multiple nodes.

To assess the scalability of Geocon in a real-world scenario, we developed a location-aware mobile application, called *GeoconView*. The application allows users to share information about events exploiting the Geocon middleware for storing, indexing, and retrieving such information. The experimental results show that the latency speedup is basically independent from the number of queries per second, but is significantly higher when the system stores a larger number of events (e.g., 2000k vs 500k). For instance, when the system stores 2000k events, with 15 queries per second, the latency speedup passes from 1.74 using 2 data nodes, to 4.74 using 8 data nodes. The Geocon middleware is available as open-source software at `https://github.com/SCAlabUnical/Geocon`.

References

[1] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, WMCSA '94, pages 85–90, 1994.

[2] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, January 2001.

[3] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4):263–277, 2007.

[4] Darren Black, Nils Jakob Clemmensen, and Mikael B. Skov. Pervasive computing in the supermarket: Designing a context-aware shopping trolley. *Int. J. Mob. Hum. Comput. Interact.*, 2(3):31–43, July 2010.

[5] Anand Agarawala, Saul Greenberg, and Geoffrey Ho. The context-aware pill bottle and medication monitor. 2004.

[6] Roy Want, Andy Hopper, Veronica Falcão, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, January 1992.

[7] Paul Adams. *Grouped: How small groups of friends are the key to influence on the social web.* New Riders, 2011.

[8] Loris Belcastro, Giulio Di Lieto, Marco Lackovic, Fabrizio Marozzo, and Paolo Trunfio. Geocon: A middleware for location-aware ubiquitous applications. In *Proc. of the First International Workshop on Ultrascale Computing for Early Researchers (UCER 2016)*, 2016.

[9] F. Marozzo, D. Talia, and P. Trunfio. Using clouds for scalable knowledge discovery applications. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7640 LNCS:220–227, 2013.

[10] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide.* " O'Reilly Media, Inc.", 2015.

[11] Domenico Talia, Paolo Trunfio, and Fabrizio Marozzo. *Data Analysis in the Cloud.* Elsevier, October 2015.

[12] Stefan Poslad, Heimo Laamanen, Rainer Malaka, Achim Nick, Phil Buckle, and Alexander Zipl. CRUMPET: creation of user-friendly mobile services personalised for tourism. In *3G Mobile Communication Technologies, 2001. Second International Conference on (Conf. Publ. No. 477)*, pages 28–32, 2001.

[13] Mark Van Setten, Stanislav Pokraev, and Johan Koolwaaij. Context-aware recommendations in the mobile tourist application compass. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 235–244. Springer, 2004.

[14] Cormac Driver and Siobhan Clarke. An application framework for mobile, context-aware trails. *Pervasive and Mobile Computing*, 4(5):719–736, 2008.

[15] Ankur Gupta, Achir Kalra, Daniel Boston, and Cristian Borcea. Mobisoc: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14(1):35–52, 2009.

[16] Anind K Dey, Gregory D Abowd, and Daniel Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, 16(2):97–166, 2001.

[17] Jianchao Luo and Hao Feng. A web-based framework for lightweight context-aware mobile applications. *International Journal of Database Theory and Application*, 9(4):119–134, 2016.

[18] Marcelo Malcher, Juliana Aquino, Hubert Fonseca, Lincoln David, Allan Valeriano, and Markus Endler. A middleware supporting adaptive and location-aware mobile collaboration. In *Mobile Context Workshop: Capabilities, Challenges and Applications, Adjunct Proceedings of UbiComp*, 2010.

[19] A. Corradi, M. Fanelli, and L. Foschini. Implementing a scalable context-aware middleware. In *2009 IEEE Symposium on Computers and Communications*, pages 868–874, July 2009.

[20] L. Capra, W. Emmerich, and C. Mascolo. Carisma: context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, 29(10):929–945, Oct 2003.

[21] C. Julien and G. C. Roman. Egospaces: facilitating rapid development of context-aware mobile applications. *IEEE Transactions on Software Engineering*, 32(5):281–298, May 2006.

[22] Chien-Chih Yu and Hsiao-Ping Chang. Personalized location-based recommendation services for tour planning in mobile tourism applications. In *International Conference on Electronic Commerce and Web Technologies*, pages 38–49. Springer, 2009.

[23] ECMA. Ecma-262: ECMAscript Language Specification. Fifth edition. *ECMA (European Association for Standardizing Information and Communication Systems)*, 2009.

[24] Leonard Richardson and Sam Ruby. *RESTful web services*. O'Reilly Media, Inc., 2008.

[25] F. Marozzo, D. Talia, and P. Trunfio. A cloud framework for big data analytics workflows on azure. *Advances in Parallel Computing*, 23:182–191, 2013.

[26] Albino Altomare, Eugenio Cesario, Carmela Comito, Fabrizio Marozzo, and Domenico Talia. Trajectory pattern mining for urban computing in the cloud. *Transactions on Parallel and Distributed Systems (IEEE TPDS)*, 28(2):586–599, 2017.