A High Performance Open API platform for Disaster Management, integrating UAVs, Mobile and IOT devices

Sarantis Kyritsis

Mineral Resources Engineering

Technical University of Crete

This dissertation is submitted for the degree of

Master of Science
July 2020

I would like to dedicate this thesis to my loving parents . . .

Acknowledgements

This thesis has been a long and sometimes difficult journey.

I would like to express ...
my gratitude to my parents, grandparents, brother and sisters for supporting my every endeavor
my unconditional love to my brother and sisters
my respect to my teachers for the guidance and patience
my support to SenseLab Research Group, this team has been the foundation of myself as a researcher
my sincerest admiration to my mentor Dr. Partsinevelos, for all the knowledge and support throughout these years

and last but not least my unconditional love and gratitude to Vicky for the emotional support during this journey.

The end of a journey, is always the beginning of a new one...

Abstract

Search and rescue (SAR) constitutes a crucial, recurrent and integral challenge for civil protection entities. With the use of current technology, multiple parts of a SAR mission can be inter-connected in real-time, by using multiple aspects from the world of GIS systems, as well as leveraging the capabilities of embedded systems for on the fly application deployment. Multiple approaches have been undertaken so far towards bridging these fields, but the proposed system is utilizing the advantageous elements of primary memory databases, as well as the performance of statically typed compiled languages, such as Golang.

The goal of this thesis is the creation of an Open API, real time and highly performant disaster management system. The key elements of this system are that it is easily deployable, easily configurable and implements a variety of different communication protocols. The drive for this implementation is that the resulting system can be deployed on any Linux-based embedded system, works with a variety of messaging protocols and is completely agnostic of the underlying network infrastructure. For the creation of this platform, a variety of technologies have been used. The deployment system is based on docker containerization, the programming language is Golang, while the main database used is Redis.

The resulting platform has been thoroughly stress tested in handling loads of transactions, simulating high load real world scenarios. Through this platform, multiple field agents (i.e. drones, rescuers, volunteers, teams of people) can coordinate their actions by utilizing whatever available network infrastructure is still operating, or can be easily deployed on the spot. Moreover, people that are facing problems (i.e. cannot move, are injured, or are trapped) can directly send their position, as well as their status through this platform, so that the responding rescuers can approach them efficiently. Performance tests indicate that the system can handle 33269 concurrent requests per second regarding creations and updates of data in the system and 57543 concurrent requests per second fetching data from the platform. To put things into perspective an ArcGIS Server REST 2D Vector mapping services supports up to 79,710 requests per hour, and a heavily optimized PostGIS installation can reach up to 7432 requests per second.

The final system offers a real-time disaster management platform, heavily optimized for fast messaging performance between the users of the platform, and mainly oriented towards team coordination in SAR operations.

Περίληψη

Η έρευνα και η διάσωση (SAR) αποτελούν μια κρίσιμη, επαναλαμβανόμενη και αναπόσπαστη πρόκληση για τις οντότητες πολιτικής προστασίας. Με τη χρήση της τρέχουσας τεχνολογίας, πολλαπλά τμήματα μιας αποστολής SAR μπορούν να επικοινωνήσουν σε πραγματικό χρόνο, χρησιμοποιώντας πολλαπλές πτυχές από τον κόσμο των συστημάτων GIS, καθώς και αξιοποιώντας τις δυνατότητες των ενσωματωμένων συστημάτων για την ανάπτυξη της εφαρμογής σε πολύ σύντομο χρονικό διάστημα.

Μέχρι σήμερα έχουν υπάρξει πολλαπλές προσεγγίσεις για τη γεφύρωση αυτών των πεδίων, αλλά το προτεινόμενο σύστημα χρησιμοποιεί τα πλεονεκτήματα των βάσεων δεδομένων πρωτογενούς μνήμης, καθώς και τις επιδόσεις μεταγλωττισμένων γλωσσών στατικών τύπων, όπως η Golang.

Σκοπός αυτής της διπλωματικής εργασίας είναι η δημιουργία ενός συστήματος Ανοιχτού API, πραγματικού χρόνου και υψηλής απόδοσης για την διαχείριση καταστροφών. Τα βασικά στοιχεία αυτού του συστήματος είναι ότι είναι εύκολα αναπτυσσόμενο, διαμορφώσιμο και υλοποιεί μια ποικιλία διαφορετικών πρωτοκόλλων επικοινωνίας. Ο στόχος για αυτήν την εφαρμογή είναι το προκύπτον σύστημα να μπορεί να αναπτυχθεί σε οποιοδήποτε ενσωματωμένο σύστημα που βασίζεται στο Linux, να λειτουργεί με πολλαπλά πρωτόκολλα ανταλλαγής μηνυμάτων και είναι απόλυτα ανεξάρτητο της υποκείμενης υποδομής δικτύου. Για τη δημιουργία αυτής της πλατφόρμας, χρησιμοποιήθηκαν διάφορες τεχνολογίες. Το σύστημα ανάπτυξης βασίζεται στο σύστημα Docker, η γλώσσα προγραμματισμού είναι η Golang, ενώ η κύρια βάση δεδομένων που χρησιμοποιείται είναι η Redis.

Η πλατφόρμα που προέκυψε έχει δοκιμαστεί εκτενώς στο χειρισμό φορτίων συναλλαγών, που προσομοιώνουν σενάρια που πλησιάζουν τις πραγματικές απαιτήσεις και εστιάζουν στον υψηλό φόρτο πακέτων. Μέσω αυτής της πλατφόρμας, πολλοί πράκτορες (π.χ. αεροσκάφη, διασώστες, εθελοντές, ομάδες ανθρώπων) μπορούν να συντονίσουν τις ενέργειές τους με τη χρήση οποιασδήποτε διαθέσιμης υποδομής δικτύου εξακολουθεί να λειτουργεί ή μπορεί να αναπτυχθεί εύκολα επί τόπου. Επιπλέον, τα άτομα που αντιμετωπίζουν προβλήματα (δηλαδή δεν μπορούν να κινηθούν, τραυματίστηκαν ή βρίσκονται παγιδευμένα) μπορούν να στείλουν απευθείας τη θέση τους καθώς και την κατάστασή τους μέσω αυτής της πλατφόρμας, έτσι ώστε οι διασώστες να μπορούν να τα προσεγγίσουν αποτελεσματικά. Οι δοκιμές απόδοσης υποδεικνύουν ότι το σύστημα μπορεί να χειριστεί 33269 ταυτόχρονες αιτήσεις ανά δευτερόλεπτο για δημιουργίες και ενημερώσεις δεδομένων στο σύστημα και 57543 ταυτόχρονες αιτήσεις ανά δευτερόλεπτο για την εξαγωγή δεδομένων από την πλατφόρμα. Συγκριτικά, οι υπηρεσίες χαρτογράφησης του ArcGIS Server REST 2D Vector υποστηρίζουν έως και 79710 αιτήσεις ανά ώρα και μια εξαιρετικά παραμετροποιημένη εγκατάσταση PostGIS μπορεί να φτάσει έως και 7432 αιτήματα ανά δευτερόλεπτο.

Το τελικό σύστημα προσφέρει μια πλατφόρμα διαχείρισης καταστροφών πραγματικού χρόνου, βελτιστοποιημένη για γρήγορη απόδοση μηνυμάτων μεταξύ των χρηστών της πλατφόρμας και κυρίως προσανατολισμένη προς τον συντονισμό των ομάδων σε επιχειρήσεις SAR.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

In the modern age, the acquisition of data from multiple sources has surpassed all expectations. The rapid growth of the Internet of Things (IoT) is disrupting virtually every industry as innovative companies create new business models that exploit the wealth of data these networks generate. At the same time, the reduction in the costs of acquiring and using small versatile Unmanned Aerial Vehicles (UAVs) has created new and innovative ways to quickly survey dangerous areas, minimizing the risk of human life. Those technologies, in accord with the help of firefighters, civil protection agencies and volunteers seem to create a vast human and machine network, which could work as a unified protection system against natural disasters.

1.2 Problem Statement

Although, the sources of data have become richer, the processing of them is still based on traditional GIS systems, thus creating huge limitations in terms of time and space complexities. This means, that even if we have an ever expanding availability of different and complex information, those data take a long time to be processed and finally lead to actionable decisions. Furthermore, traditional GIS systems are not equipped to handle real-time events.

This means that with today's means, the way to handle events caused by the movements of different objects would need complex middlware solutions, communicating with a traditional GIS platform to somehow handle event driven messages.

1.3 Motivation

Natural disasters have always been one of humanity's biggest problems. That's why many steps have been taken towards minimizing the danger of human life, regarding the management and resolution of those disasters. But in the last years, and even more this past year, it seems that there is still much room for improvement. Especially in this past year, we have seen multiple disasters done by wildfires in Greece, Brazil and Siberia. Those disasters all account to the further declination of the already unstable climate around the planet Earth. Moreover, through my exposure to real-time data sources, such as drones, IoT devices and wearable devices, I have come to the conclusion that all those data streams could be used in a unified platform to provide us with better means of resolving more efficiently such disasters, and further help the protection and preservation of our planet, as well as minimize the risk of human life in the process.

1.4 Goal and Hypothesis Objectives

The realization of the aforementioned limitations in actual real-life scenarios led me to the idea of creating a new platform that would solve the problems of modern real-time, cooperative disaster management applications. In order to do that, the new platform should be able to handle and process geospatial data in real-time. This means that all the storage, recovery and spatial functions should be done with really strong time restrictions. At the same time, the platform should be able to handle multiple bidirectional communication channels between all the users that take part in a disaster management operation. Finally, since a large scale natural disaster might cause large network infrastructure problems, the platform must be completely network infrastructure agnostic.

1.5 Research Approach

In order to conclude on the final architecture and supported functions of the system, multiple things must be researched. For starters, an evaluation of already existing systems and approaches must be taken into account. Furthermore, multiple real-life scenarios must be examined in order to find possible weak spots in the scenario resolutions. As an overall, this platform's goal is to make all relevant information readily available to all the parties that need that kind of information in order to act as efficiently as possible. Moreover, the platform's performance is of great importance as all the supported functions must perform as efficiently as possible in order to provide real-time functionality and easy on the go deployment.

1.6 Applications

This platform's primary focus is the management of multiple agents in a disaster management scenario. But, taking into account the functions from which it consists it could be used for multiple applications. It could be used to help self-driving cars, acquire data about other cars on the road, and get messages in time for traffic jams, accidents or special events. It could also be used by air-traffic control systems as a lightweight and fast alternative that would automate the process of accident avoidance.

1.7 Organization of the Remaining Chapters

In this chapter we focused on the problems that exist on the field of real-time data handling in traditional GIS systems, and how these can be solved with the proposed solution. In **Chapter 2** we are going to investigate what has been done in the field by previous works. Moreover, we are going to examine multiple tools and techniques that will prove useful in the creation and optimization of our system. In **Chapter 3** our main objective will be to present the system's components and architecture. Specifically, we are going to focus on the technical structural components of the system, and how those components are connected in order to synthesize our final solution. **Chapter 4** will present some characteristic use-case scenarios that will help in better understanding the intended use and capabilities of the system. Furthermore, in this chapter some characteristic benchmarks will be presented which will help better understand the performance of the system under stress. We will conclude with **Chapter 5**, which will act as a summary of what has been presented and what was achieved. In this chapter, some ideas for future work will also be presented.

Chapter 2

Relevant Work

2.1 Background

2.1.1 A dive into the Geographic Information System

Geographic Information System (GIS) standards have previously been described by variations in the definition of an IS (information system) or an integration of hardware, software and telecommunications networks that have developed and used to gather, produce and propagate relevant knowledge in an organizational context. The difference between the "IS" and the "Geographic Information System" is that the latter integrates the concept of space into the system. (Schneider & Valacich, 2010). For instance, software and hardware made to accomplish map development, perform geographic analysis, create mapping applications (Longley et al. 2010), and examine statistical analysis with spatial constructions.

2.1.2 Explaining the response to disasters

Disaster Management consists of a series of actions to react to and mitigate an ongoing crisis. Additional information on the evolution from response to direct restoration is elaborated by Anthopoulos et al. (2013). "Disaster Response" involves a range of concepts suggested by different government policies and institutional frameworks. It is focused on adjustable, expandable, and evolving practises. For example, the NIMS (National Incident Management

System) aligns key duties and responsibilities throughout the US region. This system defines details for warnings and industry requirements for crisis responses ranging from serious but local emergencies to severe acts of violence or major natural disasters (U.S. Department of Homeland Security 2013: p. I). Under this context, the GIS is explicitly referred to as a system that integrates both the National Response Framework (NRF) and Emergency Support Function Annexes or ESFs (United States Department of Homeland Security 2013). For instance, the ESF 5 "Emergency Management Officers" states that "The design development personnel focus on providing, managing and modifying location information (Federal Emergency Management Agency (FEMA) 2008: p. 5-1)."
Furthermore, in other areas such as in Germany, “Disaster Response” is incorporated in the overall disaster management process. Disaster management can be described as all efforts to mitigate, track, avoid, respond to, and recover from further disasters. To increase its efficiency, Germany has merged regions, and has separated responsibilities between national and state governance. As a consequence, there are often so many definitions of disaster response within the different laws. These laws can be summed up as tools for tracking individual, environmental, and other natural or man-made disasters and providing help in the event of a crisis. Disaster Response situations involve a concerted effort to interpret GIS data, and make sure that those data are properly communicated throughout all the units taking part in the response effort. The primary responsibility for disaster mitigation lies in countries where the federal government has a duty to protect them. During major accidents, national resources (e.g. an Associate Assistance Department / Police Service or a National Information and Communication Centre) are allocated by the Federal State.
Based on the global scale, the UNISDR (UN Office for Disaster Risk Reduction) identified and responded to disaster management by defining it as : ’The urgent protection and government assistance provision during a crisis to save many human lives, minimize health effects, protect citizens and meet the needs of people directly affected.
Common issues in the aforementioned examples are the welfare of rapidly changing societies and the delivery of essential human needs and facilities. In the next section, we will include

examples which illustrate the huge role of the GIS in disaster response.

2.1.3 Geographic Information System for disaster management

Catastrophic events still expose the conceptual and long-term need for the GIS in disaster management. For instance, the initial reaction to the 9/11 attacks demonstrates a list of fundamental GIS actual examples to integrate reaction organizational activities, but also highlights the need for relevant information exchanging of pre-event action plans (Kevany 2003).
During the Haiti quake in 2010, it was at the very first instance, when non GIS Personnel got involved in Disaster and Emergency Mapping Activities. This involvement stressed the significance of consolidating structured disaster information with available data, identifying disaster areas and evaluating catastrophic damage from a variety of different information sources.
During that year, a massive oil spill occurred as a result of the Deep-Water Horizon oil spill in the Gulf of Mexico. Such an event has a surprising environmental effect that cannot be resolved without local temporal data on the location of the explosion, its potential escape, and reliable geospatial data.
In addition, in some publications on the use of the GIS systems in disaster management, there is a large amount of knowledge that is used to identify where GIS information has been used. In the broader field of Disaster Response, GIS systems can provide information regarding the peace and safety of an area, as well as point out possible dispute threats by analyzing geospatial data. Disaster risk management and dispute resolution have some important factors in common and have a similar purpose. The most significant aspect is that the objective is to improve the living conditions of people affected by and after a natural disaster or civil conflict (Kobayashi 2012). Security studies are highly dependent on space and spatial analysis, including the vital positions of violent commencement or sectarian violence and post-disaster circumstances. By linking space and conflict, both numerically and factually, peace and conflict researchers aim to develop a better understanding of danger circumstances as well as

to provide decision-makers with greater tools to prevent or, at least, reduce danger.
At the start of the 21st century, conflict and security researchers have examined the appropriate boundaries and procedures of various conflicts (Ross, 2004). The assessment of conceptual factors such as topography, geographic position and dispersion of land, facilities and credibility of national borders as well as the effect on mental struggles is described in Baechler et al., 2002. Other areas of expertise play a vital role in government systems and sustainable development and their significance to sectarian violence. The link among changes in the environment and dispute has been already established in the scientific community (Vandergeest and Peluso 1995). As we progress, and more data are gathered, climate change and conflict tend to show strong signs of correlation.
As a result, of the plethora of gathered GIS data the conflict and security research focus has shifted from a nation-wide scope to a sub-provincial level (Theisen, 2012). This shift resulted in the growing use of GIS systems as a data analysis and visualisation method. The list of risk zones with high conflict probability is rising. Most of the examples show three similar traits in the management of disaster while using Geographic Information Systems:

Situation Perception improved by Geographic Information System

Situational awareness (or SA) is a term used in both the army and disaster management teams. It is done so, because these formations show similarities in terms of organisation and operational tactics. Another phrase akin to SA is "common operating picture" or COP. Within the army COP, was precisely defined as "the shared visual representation of the pertinent data provided by a unified program that enables close coordination and helps all units realize the orientation of the situation" (Endsley 1995; Endsley and Garland 2000). Spatial-based COP representations enable the accomplishment and preservation of SA for groups of individuals, during a disaster response, via the use of spatial systems and GIS in general. For instance, maps and fundamental location data may provide a range of services, such as visual disaster management in earthquake situations or national emergency situations (Green and Parrish 2013), and enable disaster emergency workers to make choices and undertake the appropriate actions as reaction circumstances become time critical.

Time Sensitive Data Geographic Information Systems for Disaster Management

Use of GIS in disaster management has a number of special hurdles. In the event of a disaster, there is the urgent need for data regarding the afflicted areas and communities which might be in high risk. The initial reaction to a disaster is associated with a loss of information, particularly during a huge crisis. A huge proportion of the time is utilized to gather actionable data and incorporate them into a comprehensive spatial representation. GIS can assist in this task by providing the infrastructure to import all this data for a particular geographical area. However, setting up a GIS database (e.g. defining a set of data structures) or researching an appropriate way to analyze when a catastrophic event occurs takes a significant amount of time. Disaster management entities can only use what is currently in place at the time of disaster on the basis of technology, infrastructure and organizational interpretations. Paddy et al. (2014) report that, especially during a sudden occurrence of a disaster, any circumstantial information becomes invalid or irrelevant within a short period of time. It was found that just after 3-4 days, the emergency management personnel found the information regarding critical infrastructure completely useless. Unfortunately, the same conclusion was drawn for digital images that were just 3-4 days old. The geographical information used (and needed) for disaster management are identified as follows:
-Spatial data (e.g., basic position information)
-Sensitivity-specific data for the location and/or type of disaster (e.g., location information, key infrastructure, confined spaces, land use, etc.)
-Real-time status data (e.g., location of services and units, actual injuries, traffic status, etc.)
The general approach is to set up a local database containing the above-mentioned data and to gain access to real-time data feeds prior to a disaster. In addition, it is important to ensure that geographical information is available during the event (Zerger and Smith, 2003). Challenges can arise as a result of data availability, scale constraints, and access to information. Kevany (2005) summarizes the numerous lessons learned since the 9/11 incident on the use and application of GIS and spatial data during the New York disaster management period (Diehl and van der Heide 2005; Kevany 2005; National Economic Council 2007a, b).

Cross-disciplinary data sharing, coordination, and collaboration

Shared among all disaster relief activities is that they require collaboration between various agencies both directly and indirectly. Theoretically, all stakeholders as well as the public and affected industries have similar information requirements: incident outcomes, stakeholder expectations, and infrastructure or restoration status. As a result, massive quantities of data are produced and accessed during reaction efforts; however, the larger part of that data is presented as official statements, PowerPoint slides or inconsistent data tables. In certain cases, however, the key data include a spatial attribute, to a certain extent and accuracy, that can be used as a connection to the aforementioned information and provide a better response to spatial related questions. The main goal in the field of disaster response is to achieve a personalised view of the relevant data for each different respondent, depending on his needs and demands. But this can only be accomplished via internet-based data acquisition and inter-system data sharing (Neuvel et al. 2012). The first solitary system for the direct sharing of geographic information was launched after hurricane katrina (Mills et al. 2008). Around the same time, efforts were made to establish GIS based in the Netherlands and Germany to address obstacles in distributing spatial data to various companies (Diehl and van der Heide 2005; Köhler 2005). The rapid proliferation of web-based technologies to promote collaboration, data sharing, and interoperability would only exacerbate this phenomenon (examples such as GeoPlatform.gov and subsequently FEMA GeoPlatform).

2.2 The Current State of Geographic Information System for Disaster Response

In the subsequent section, we explain the latest state of GIS for disaster management and how much we have managed to improve since the initial reviewed literature. We are intending to discuss using a future framework, such as the involvement of GIS in disaster management. First, we will iterate over how GIS disaster response is seen in related articles or status reports. After that, we will test GIS for disaster management within the information of the two system components, users who create and concentrate on GIS and data. We will pay particular attention to these two aspects of the general GIS approach, as we will argue that the range of information and the range of actors involved in GIS driven disaster response, is an excellent example of how the involvement of GIS in disaster response is increasing.

2.2.1 Geographic Information System Disaster Response in Literature

As described in Tomaszewski et al., 2015 the creation of the term "geographic data system" is often directly linked to Geographer Roger Tomlinson and his determination to include it, in the 1960s, in the Canadian GIS. Over the years, GIS software has surfaced in its modern incarnation, from a combination of theoretical perspectives such as geography, architectural design, computers, and information technology. The specific definition of space and analytical capacity provided by GIS in the area of disaster management has a long history of disaster risk work, leading to simple GIS disaster management papers (or related words such as "emergency management") appearing in the GIS literature. GIS emergency management involves a 20-year discourse in research journals depending on specific research. For our evaluation, we concentrated on critical articles in geographic informatics journals, and also on the affirmation of peer-reviewed articles in risk management journals. The research in most well-known disaster management articles shows a significant number of publications on GIS topics. The vast majority of disaster management journals (including accident studies) are available only from 2000 onwards. The search shows that GIS play an important role in scientific discussions in the field of risk management and risk research.
Over the next section, we present significant educational papers that access the GIS disaster relief efforts.
Reports dealing only with the role of GIS in disaster management are most often traced back to 1992, just before GIS software had become accessible to many viewers for the first time thanks to the advances in Technological innovation (Johnson 1992). Coppock (1995) was the first Geographic Information System and Environmental Threat Analysis study to highlight a range of problems that continue to exist today in data shortages, GIS availability, user requirements, and network failures.

In reality, GIS emergency management has never been a trivial issue in Geography related research and has originally gained attention from many other perspectives such as safety research (Johnson 1995: p. 133). GIS can be a national emergency management system that can be used completely in the stages of the crisis cycle. Moreover, Dash (1997) was the first study to look at the use of GIS in scientific qualitative studies.

Cova (1999) was the first GIS user to be a multidisciplinary expert at all stages of disaster mitigation (which include, but are not confined to, disaster response) and identified unique GIS experiments with risk assessment models that still relate to the migration design process. Cutter, 2003 was among the main papers in the 9/11 era who proposed constructs that would merge GIS and disaster response principles. The events of 9/11 and Hurricane Katrina intrigued a great deal of attention to the concerns of disaster risk management in GIS related sciences and triggered progress in GIS scientific research on subjects as diverse as 3D GIS simulation in urban rescue operations (Lee and Zlatanova, 2008), demographic map representations on financing group work (MacEachren and Cai, 2006) and the forest fire suppression model (Cova et al., 2005).

Breakthroughs in web 2.0 technology, data infrastructure building, local voluntary work, mainstream press, and data analytics that started in the early 2000s have led to numerous research topics focusing on disaster management actors, visual analytics, and in general GIS disaster management awareness. Garb et al. (2007) presented research on realistic ways to assess human risk in the ability to forecast disasters. The research on data infrastructure development and data models explicitly for disaster management was addressed by Neuvel et al. (2012), Zlatanova and Dilo (2010) and Frigerio et al. (2013). Goodchild and Glennon (2010) were also the first ones to reach the audience and use GIS strategies to fill the information gap, but stumbled upon data quality problems. Tomaszewski (2011) explored the use of digital maps with random sources of data, and evaluated the benefits and caveats of such a practise being used in disaster management scenarios. Also, at the time, Liu and Palen (2010) identified new ways to "use the map" to help disaster response by combining various web features such as Google Maps, timelines, and YouTube broadcasts. Robinson et al. (2011) presented a study to fully map the characteristics of the map symbols to specific interactions. MacEachren et al. (2011) identified work on the distribution, and portrayal of Twitter updates in support of emergency preparedness. Twitter feed can be an alternative sensor design which would distribute information regarding the time and location (Crooks et al. 2013). Community information systems could also facilitate interaction between camp counselors and staging areas, as suggested by the studies by Annunziato et al. (2010), Frassl et al. (2010) and Kiltz and Smith (2011).

The review in Tomaszewski et al., 2015 -supported by GIS variability in disaster response- has demonstrated distinct scientific papers circulating three GIS related pillars: the people, the data and the usage. Cutter (2003) considered that future research activities should be mostly focused on the needs of users. The risk management process and recommended solutions for it can be traced back to a large number of publications. There are currently a lot of user interfaces online, though the actual emergency situations that GIS were properly used are scarce.

2.2.2 Spatial Data

Spatial data are the most significant and vital characteristics of the GIS. Expenditures around the spatial data include money and time needed to obtain, process, and maintain a wide range of geographical information. In addition, expenses include IS facilities, training for GIS employees and other advancements used to guide GIS, backup and restore systems, computer technology, and other information and communication Technology (ICT) (desktops, servers, radios, tablets, GPS linebackers) (National Council 2007a, b).

A main pattern over the last 5 years is the availability of public open source spatial data. This significant contribution is driven by the community and it is enhancing our understanding of disaster management circumstances. In order to have access, many of this publicly available information and data have been generated by non-specialist producers who are often referred to as VGI or " volunteered geographic information" (Goodchild, 2007). Maybe the most substantial VGI effort identified in the framework of the disaster management was the Open Street Map (OSM) 3 project (Haklay and Weber, 2008). The concept behind OSM would be that in the "wiki" map, anybody can make a contribution and edit a map of the world, which is very comparable to the Wikipedia page. OSM attracted particular attention in 2010 for its capacity to use its wiki-map method to generate adequate sets of data to fill data gaps during the 2010 Haiti earthquake (OpenStreetMap Team Zook et al. 2010). Humanistic OpenStreetMap Team 4 (H.O.T.) was established as a particularly unique OSM team, which supports humanitarian actions and provides data on all significant events (mostly international and European). Use of VGI to acquire disaster management data continues to grow with the use of disaster response-management of circumstances where local data don't actually exist (such as in developing nations) or are inaccessible (due to security or government reasons). For instance, the observer project began focusing on "tragedy mapping" via the use of web-based layouts and a crowd-sourcing platform to regulate violent conflicts in Kenya (Okolloh 2009). Tragedy mapping appears to be a big way to fill in the gaps in information produced by the Libyan disaster of 2011, the continuing Syrian war, Hurricane Haiyan 2013 and the continuing Ebola pandemic of 2014 (Meier 2012; Hodson 2014; Zastrow 2014). But those approaches have their drawbacks. The literature indicates that mapping natural hazards and VGI information obtained by the public cannot be considered reliable, accurate or consistent and corporate governance should be regarded when assessing the use of VGI in evaluating rescue efforts.

Satellites, Space Technology and Unmanned Air Vehicles (UAVs)

Satellites offer fast and stable data transmission, positioning and surveillance tools, which is of great value in cases when the land based infrastructure has suffered damage (Mandl et al. 2013). Risk analysis, disaster risk reduction, and preventative measures also greatly benefit from space driven data. As an overview, the use of space technologies supports disaster response in its different fields. This is achieved via the use of satellites for the purposes of analysis and image processing, telecommunications, positioning and path planning. Remote sensing data are used to map disaster zones for damage assessment as well as for the generation of relevant GIS information.

For instance, as mentioned above, remote sensing techniques have been shown to provide significant data and solve challenges in helping and organizing search and rescue operations during the Indian Ocean tsunami (Kelmelis et al. 2006). Google maps has been identified in publications as a method to provide visual imagery to support disaster relief efforts (Nourbakhsh et al. 2006).

Satellite systems are used to keep the relief teams connected to the status of the disaster areas (Robert Backhaus et al. 2011) as well as provide an early warning to secluded areas, for a possible incoming disaster (Ravan et al., 2011; Szarzynski and Stevens, 2009). The availability of GNSS is also really important to relief teams that approach (or are already in) a disaster area.

Among the newest developments for disaster response real-time data acquisition is the use of Unmanned Aerial Vehicles (UAVs) which support or even incorporate conventional remote sensing technology, like satellite imagery. The literature representing this domain, points to issues with system effectiveness and cooperation of multiple UAVs for better monitoring (Quaritsch et al. 2011) and uses UAVs for real-time flood forecasting (Abdelkader et al. 2013).

2.2.3 GIS Users, Producers, and Specialists

GIS products are empowered and/or used by different categories of people. The first category of people associated with GIS systems are the producers. The producers can be a single person, or a group of people, that do not necessarily use a GIS system, but assist in the creation, management and removal of GIS data, that could in turn be used to assist in disaster management activities. The GIS specialists are connoisseurs of GIS with deep knowledge in the use cases and benefits of GIS related products and data. The GIS users use GIS related products, data and technologies to perform mission tasks without the need for deeper understanding of the principles of the systems.

Producers

The producers are a diverse, global group that has received a significant amount of attention in recent days for the ability to incorporate GIS data that can be used to predict, visualize and simulate disaster scenarios and its appropriate responses. They often function without structured disaster management procedures. Even so, the significance of community access to disaster map data as supplementary disaster response is evident and "disaster maps" are now much more common, especially in the case of global disasters with big data gaps due to media and/or legal restrictions like in the 2011 Libyan Crisis (Lohr 2011).

GIS Specialists

GIS Specialists can be individuals or institutions able to judge and evolve the status of GIS, and in this specific case their uses in disaster management. Specialists are able to leverage the data gathered at the location to provide better insight regarding the status of the ongoing situation. One such example of a GIS Specialists institution, is the UK based NGO Map Action. Map Action is available to immediately respond to a crisis with a volunteer team of GIS professionals qualified in disaster management. Their plan involves a very well-designed, fast response which must be incorporated with the efficient method of other current disaster response institutions (e.g., UNDAC) in order to be effective. The different phases of the response plan are as follows: - Mobilization: in this phase, a small agile team is assembled and in parallel, the central organisation is creating data sources for the disaster area, and prepares them for the team.
- Base Station: the team is deployed to the disaster area and sets up its base station. Some members of the team then spread on the field to gather real-time data, while others analyze the data gathered from the central organization.
- Data collection status: reviews are obtained from a variety of sources. These reviews then are published, analyzed and plotted to provide answers to mission related crucial questions.
- Map Distribution: the spatial data that are gathered and evaluated are added into GIS so that they can be readily available to various field agents. These GIS data are constantly being kept up to date with the purpose of having time sensitive information in real-time.
In disaster responders, Map Action plays a crucial role in emergency preparedness, for instance, by educating social workers or offering educational material particularly to non-GIS professionals (Map Action 2011a, b).

GIS Users

The most recognisable group of GIS Users is those who battle the disaster with the purpose of saving human lives. These people can be firemen, healthcare or social welfare units. They rely on information regarding the current overview of the situation as well as potential outcomes of the situation. Researchers often study this team of GIS Users (Diehl et al. 2006; Annunziato et al. 2010; Kiltz and Smith 2011). The knowledge requirements for this community of users is very time-critical, but it is mostly centered on their specific objectives. Severe and/or long-term disasters, where many relief groups are implicated and the effects are considerable, require great communication. As the scale of the disaster grows, and the results affect various sites, the information propagation between the different relief groups must be more detailed but at the same time, fast enough. One of the very few researches done by Paddy et al. (2014) studied the use of geographical data within the period of Emergency Control Offices in the United States. Already in 2011, 90% of ECOs had at least part-time employees with GIS skills, and GIS was recognized as a very important subject.

Chapter 3

Implementation

3.1 Introduction

Having identified the existing needs in the management of real-time mission critical data, and taking into account the already implemented solutions and approaches to the needs of Search and Rescue platforms, we conclude to the following architectural approach. This approach consists of three basic pillars : the DBMS (Redis), the API (Golang) and the Messaging Systems. Following is a bottom up overview of the implementation, structured in such a way as to make the inter connectivity between the underlying components more profound to the reader.

3.1.1 The DBMS (Redis)

Redis is an in-memory data structure project implementing a distributed, in-memory key-value database with optional durability. Redis supports different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes. Redis made popular the idea of a system that can be considered at the same time a store and a cache, using a design where data is always modified and read from the main computer memory, but also stored on disk in a format that is unsuitable for random access of data, but only to reconstruct the data back in memory once the system restarts.

At the same time, Redis provides a data model that is very unusual compared to relational database management system (RDBMS), as user commands do not describe a query to be executed by the database engine, but specific operations that are performed on given abstract data types, hence data must be stored in a way which is suitable later for fast retrieval, without help from the database system in form of secondary indexes, aggregations or other common features of traditional RDBMS. The Redis implementation makes heavy use of the Fork (system call), to duplicate the process holding the data, so that the parent process continues to serve clients, while the child process creates a copy of the data on disk. Something that is of great importance to our specific implementation, is that Redis was configured to never store data on disk, in order to reduce the number of I/O cycles used by the data storage, which in return means that those CPU cycles would be used in the other components of the system such as the messaging mechanisms, as well as the facilitation of the API calls.

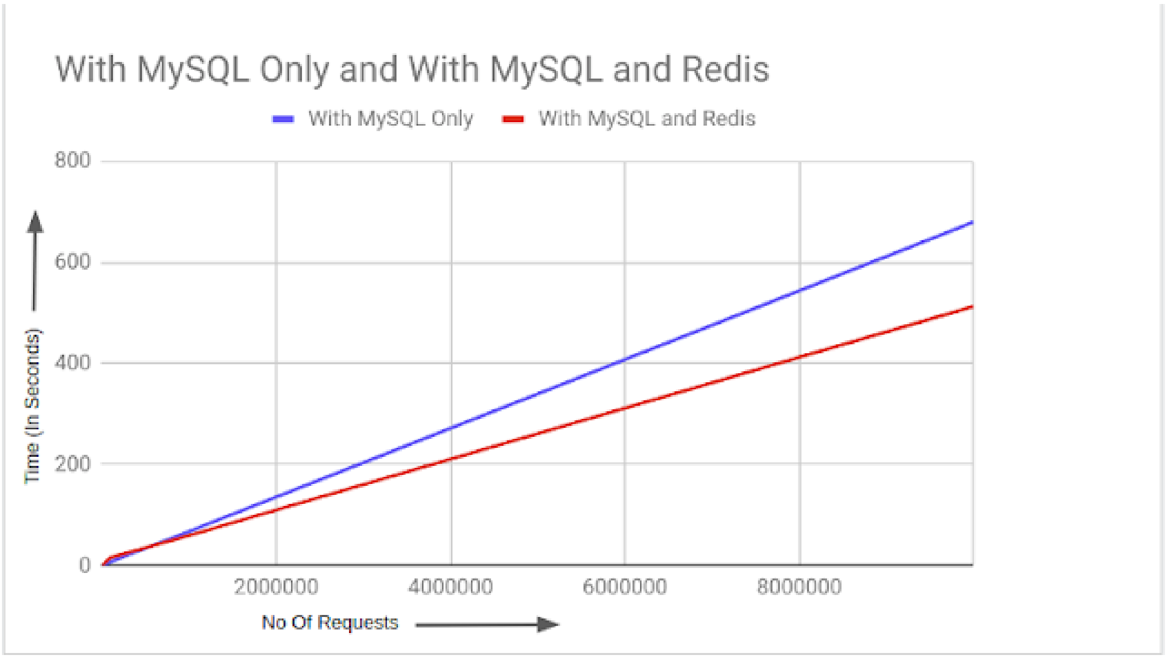

Figure 3.1 Performance upgrade of MySQL while using Redis as primary memory caching

3.1.2 The API (Golang)

Golang (Go), is a statically typed, compiled programming language designed at Google. It is syntactically similar to C, but with memory safety, garbage collection, structural typing, and CSP-style concurrency. In order to facilitate the needs for the API, multiple components where used along with Go to achieve real-time performance and reasonable resource management.

Echo

Echo is a high performance, extensible, minimalist Go web framework. It is used as the primary API router and Logger.

Paho MQTT

A Go client library for the Eclipse Paho MQTT broker

Protobuf

A Go library which adds support for Protocol Buffers, implementing Google's data interchange format

Redigo

A Go client library for the Redis database

gRPC

The Go implementation of gRPC: A high performance, open source, general RPC framework that puts mobile and HTTP/2 first.

3.1.3 Messaging Systems

The application messaging systems, are divided into two types : event based and request based. The event based messages, are triggered when a mission critical notification needs to be delivered to all subscribed members. The request based messages are transmitted, whenever a member actively requests a specific piece of information.

Event Based Messages

MQTT

MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices, and for mobile applications where bandwidth and battery power are at a premium. To be able to use the MQTT protocol, a broker needs to be set up, to facilitate as the messaging router. This implementation has used the opensource Eclipse Mosquitto, as a message broker.

WebSocket

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection. The main Golang application acts as a websocket server. For each member that opens up a new websocket connection, a websocket client is created server-side, in order to keep the connection between the member and the server. Websocket connections are used to facilitate end to end data transmissions, and are not preferred in cases of multiple member notifications, due to the iteration overhead.

gRPC

gRPC (gRPC Remote Procedure Calls) is an open source remote procedure call system initially developed at Google. It uses HTTP/2 for transport, Protocol Buffers as the interface description language, and provides features such as authentication, bidirectional streaming and flow control, blocking or nonblocking bindings, and cancellation and timeouts. It generates cross-platform client and server bindings.

Request Based Messages

RESTful API

RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data. A RESTful API, also referred to as a RESTful web service, is based on representational state transfer (REST) technology, an architectural style and approach to communications often used in web services development.

3.2 Architecture

Now that the underlying components have been established, we should elaborate on the architectural dependencies between them. The main data storage of the system is implemented using a Redis dbms. To store the individual member Ids, simple key - value storage facilities are used. Member Collection Ids are indexed and indexes are stored in B-trees. Individual member Ids, are also indexed in B-trees (child trees of Collection indexes). For the geospatial data R-trees are used for indexing, in order to speed up the queries involving intersections and joins between polygons and or points.
After the setup of the database and the choice of the specific data structures that will be used for storage and indexing purposes, Golang comes into play. The first component, is the establishment of connection between Goland and Redis. Something accomplished with the use of Redigo. After the database connection, the RESTful API is built.
The system incorporates three item classes to model the input data for the API and the database. Those three classes are the **Collection** class, the **Actor** class and the **Area** class.
The **Collection** class is a generic model that is used in order to form groups of underlying models (in this case Actors). So in a way they are used in order to provide an abstraction

layer to retrieve groups of relative information.
The **Collection** class supports the following API actions :

- Collection
  - Create a new Collection
  - Retrieve a Collection
  - Retrieve all Actors of a Collection
  - Retrieve all Collections
  - Update an existing Collection
  - Remove an existing Collection
  - Remove all Actors of a Collection
  - Remove all Collections
  - Set Action Notifications for Specific Collection

The **Actor** class is the model that contains the information for a specific system actor. As actors, we consider all the moving components in the field (i.e. Drones, IoT devices, Survivors, Rescuers).
The **Actor** class supports the following API actions :

- Actor
  - Create a new Actor and add him to a collection
  - Retrieve an Actor from a Collection
  - Update an existing Actor
  - Delete an existing Actor from a collection
  - Set Action Notifications for Specific Actor

Finally, the **Area** class is used to model areas of interest. Those areas are usually accompanied by relative events. For example, an area of interest could be a "no fly zone", which would be

accompanied by a relative event that would inform all the subscribed users in case a flying object enters the aforementioned area.
The **Area** class supports the following API actions :

- Area
    - Create a new Area and add it to a collection
    - Retrieve an Area from a Collection
    - Update an existing Area
    - Delete an existing Area from a collection
    - Set Action Notifications for Specific Area

The messaging systems described in 3.1.3 are used after a user has subscribed to a notification event. That way, a user can be notified in real time about all the relevant mission critical information that he has shown interest to. It is really important to point out that a user can some times be a non-human agent. For example, a drone can have a specific area assigned to it, and thus be notified whenever a victim appears inside its assigned area. In such a case, the drone would receive the event and perform the allocated actions, such as surveillance of the victim or even implementation of an actual rescue routine.
It should be pointed out, that though the system is oriented towards the real-time data acquisition and distribution between all the associated parties, at the same time it implements all the basic functionality that is usually found in traditional GIS systems.

3.3 Optimization

In the previous chapter, we discussed the way the data are saved in the system and distributed throughout the different end-clients, but in order to provide better filtering and search times for relevant results, certain optimizations must be taken into account.
For a single object to be retrieved from the database (anyone of Collection, Actor or Area),

taking into account that its unique identifier is known, the time complexity is $O(1)$, since it's retrieved from a dictionary like database construct. The problem arises when we want to retrieve a set of objects based on a regular expression over the id. In such a case, without the use of any indexes, the time complexity is O(N) where N is the number of components saved in the system. Now, in order to improve this, we use Btree indexes on the unique ids of all the components in the system. That way, the previous example performs much better with a time complexity that amounts to $O(logN)$.
Apart from the simple id queries, the system also allows geospatial queries. In order to better facilitate those kinds of results, rtree indexes are used over the spatial data, thus bringing the time complexity of spatial data retrieval to $O(log_{\mathrm{M}}n)$
Finally, the system allows the user to perform ranged queries on Actor properties. In order to better understand this, the following user scenario is proposed.
"A system coordinator, needs to find all the drones that are flying at a speed between 1m/s and 5 m/s."
In order to fetch the results for this the time complexity is O(N), where N is all the number of drones. To mitigate this time complexity problem btree indexes where used on all the properties accompanying an Actor. This way, the information regarding the previous example can now be retrieved with a time complexity of O(logN).

3.4 Replication

Due to the volatile nature of primary memory storage, the system needed some kind of redundancy to avoid failures. This redundancy was added through the means of real-time replication. The replication function allows for multiple nodes of the platform to run simultaneously, and all of them keep concurrent copies of the data on the system. The coordination between them is done by a simple governor, whose authority is to designate the main node of the system which acts as the primary server, and all others act as concurrent backups. In case the primary server fails, then the governor assigns one of the secondary server as the new primary, and all the other secondary servers, replicate the data of the new primary server from this point on. This gives the advantage of having multiple interchangeable instances up and running at the same time, to guarantee a far more redundant architecture opposing to the single server architecture.

Chapter 4

Use Case Scenarios / Results

In order to better understand the usage of the system described in Chapter 3, some use case scenarios will be presented. After that, some benchmarking results will be demonstrated focusing mostly on the usage around the use case scenarios. More on benchmarking can be found on Appendix A.

4.1 Use Cases

In this section three use case scenarios will be presented. These scenarios will have escalating demands in order to elaborate on the capabilities of the system. All the scenarios are loosely based on events that have already happened in the past, though the place and exact conditions have been altered in order to maintain anonymity.

4.1.1 Use Case 1 : Simple Scenario

For our first scenario, we are going to test the following hypothesis.
“A small team of rescuers is trying to locate a lost person in a park.”
In such a scenario, the system needs to have a real-time location on all the rescuers. Apart from that, the specified search area needs to be set in the system, so that the rescuers can be notified in case they venture outside of that specified area. When the lost person is found, a notification is sent towards all the system users about the location of the rescued person.

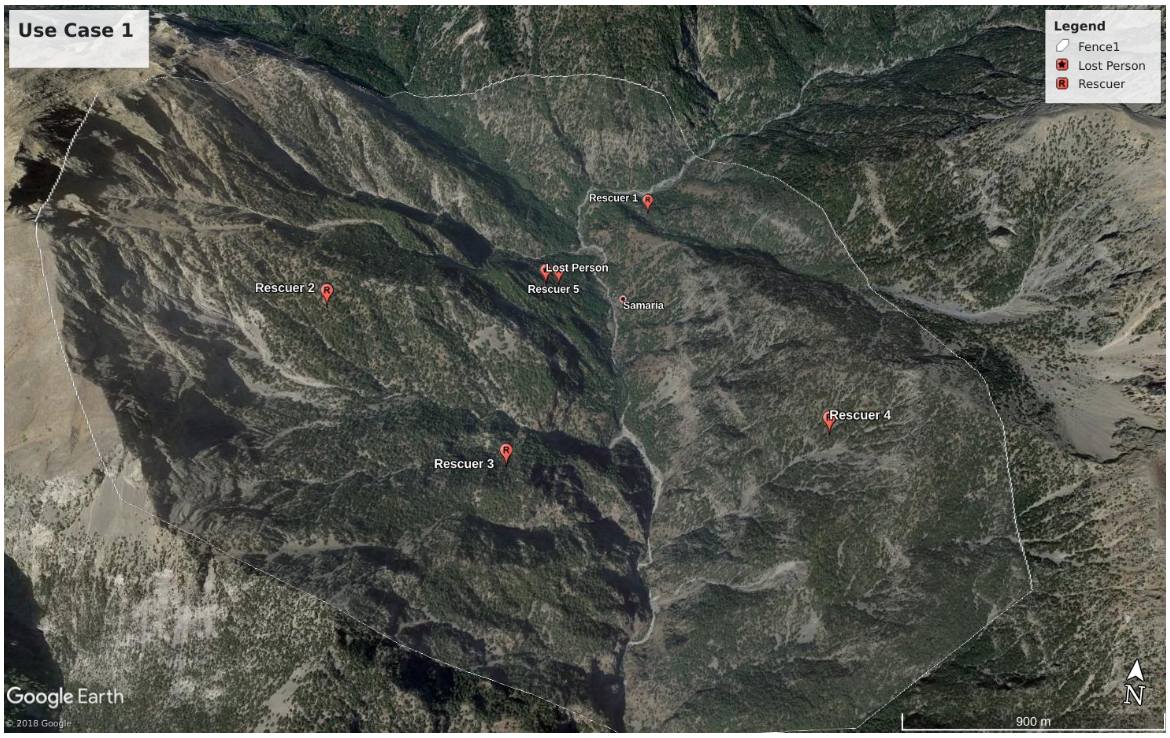

Figure 4.1 Graphical Representation of the Data for Use Case 1 Scenario

4.1.2 Use Case 2 : Medium Size Scenario

For our second scenario, we are going to test the following hypothesis.
"A fire broke out in an area with dense vegetation. Around the danger area, multiple populated areas exist that need to be evacuated and may contain people in need."
In order to mitigate such a crisis several different layers of data need to be in place for the better coordination of the rescuing services. First of all, a no flight zone fence is created around the center of the fire. This is needed so that drones won't meddle with the fire fighting airplanes that will be dropping water over the fire zone. Secondly, two evacuation zones need to be fenced. These serve the purpose of indicating to the rescuers and the service drones where they should be headed in order to locate and help any people in need evacuate the surrounding area. After that, the positions of the service drones and the rescuer teams are constantly updated in order to provide real time tracking of the situation until it is resolved.

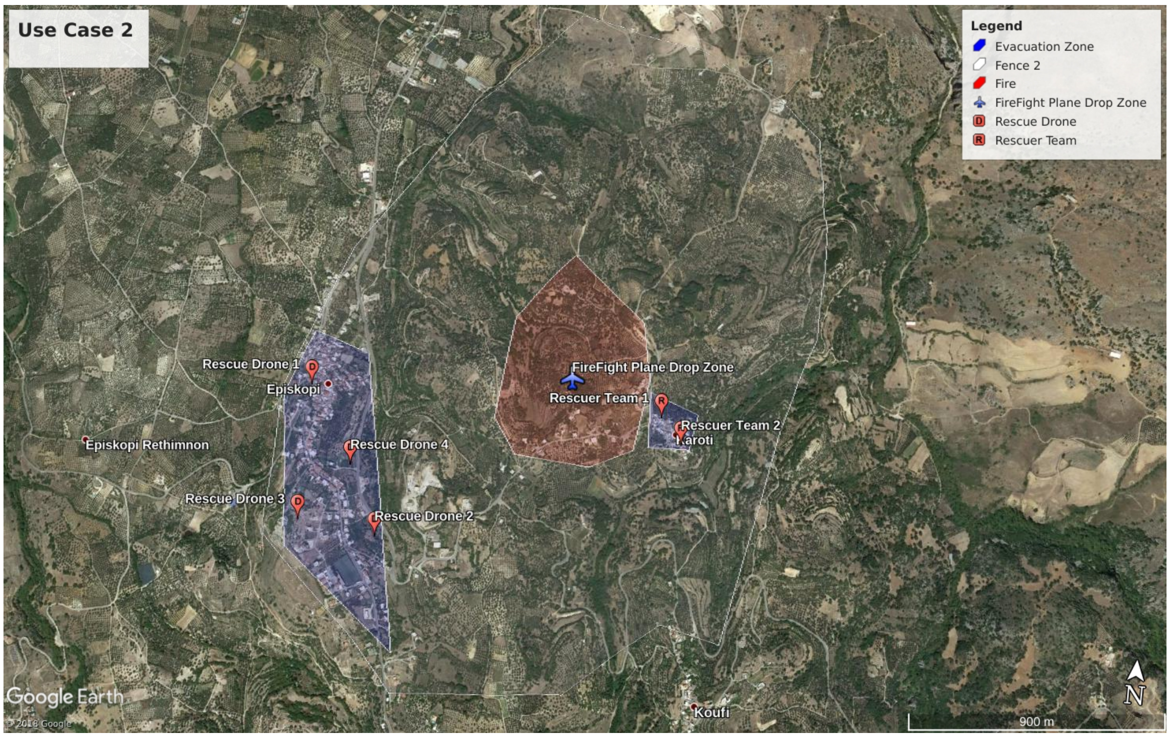

Figure 4.2 Graphical Representation of the Data for Use Case 2 Scenario

4.1.3 Use Case 3 : Large Scale Scenario

For our third scenario, we are going to test a larger example, imagining a much bigger crisis. “After an earthquake near a populated area, multiple disasters have happened. Building have collapsed, and so there are people trapped on injured. As a result of the earthquake, multiple fires have started and have merged into one big wildfire. This wildfire is progressing and is closing on the populated area. A plan for extinguishing the fire, finding possible victims and evacuating the area needs to be concocted.”

This scenario is designed to demonstrate the full fledged capabilities of the system. Multiple evacuated boats are positioned along the coastline, and are subscribed to the channel indicating the location changes of the groups of survivors. These boats are there to receive the survivors. At the same time, multiple groups of surveillance drones are positioned over the populated area to locate any new survivors and indicate their positions to the rescue groups that are deployed there. Those drones are subscribed to the channel of the survivors, as well as the no fly zones. These no fly zones, indicate the places where the fire fighter planes are going to be deploying water for the raging fire. Around the populated area, multiple fire fighter units are positioned in order to fight the fire on the ground. At the same time, the ground rescuer teams are searching for trapped people in the aftermath of the earthquake, and are subscribed to the channel reporting trapped people reports, generated either by the drones or by the people themselves.

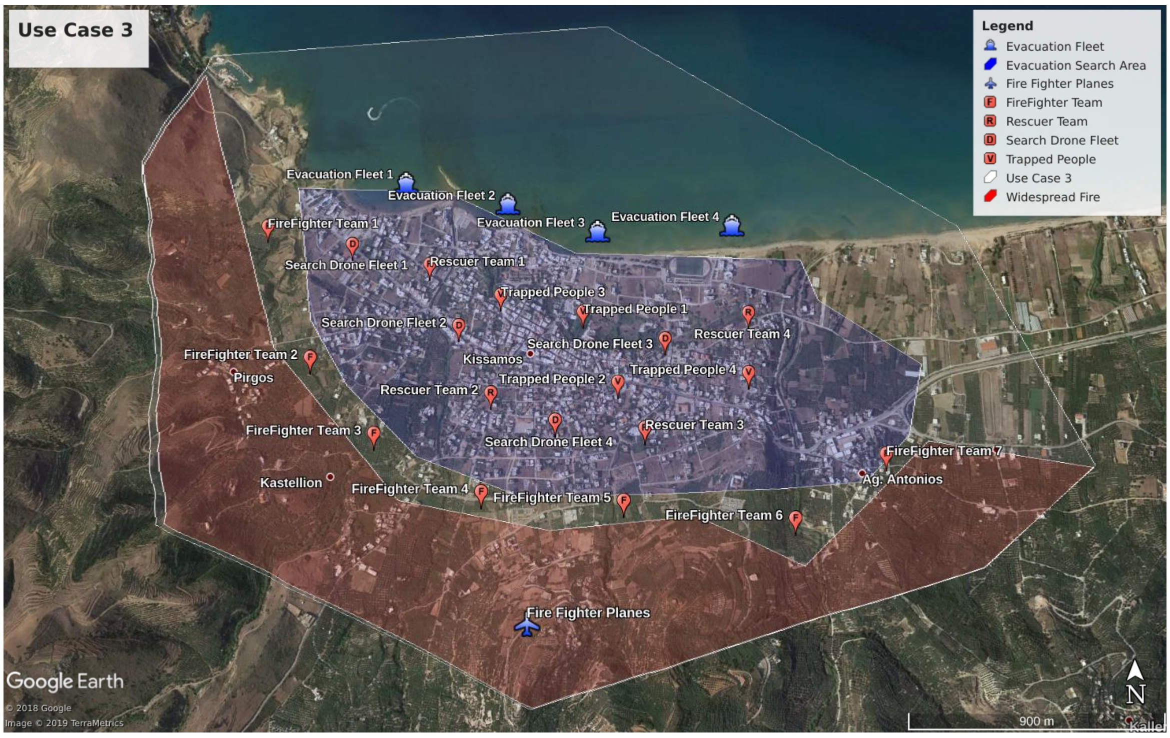

Figure 4.3 Graphical Representation of the Data for Use Case 3 Scenario

4.2 Performance Benchmarks

In the previous chapter, multiple use cases were presented. In this chapter, we are going to examine how the system reacts when in strenuous activity. In order to do that, we are going to present some important system benchmarks, that show the performance of the system on the most requested actions. All the system benchmarks can be found in Appendix A.

Actor Creation / Update / Retrieval

One of the most common uses of the system, is the creation, update and retrieval handling of an Actor.

```
====== SET (point) ======
  100000 requests completed in 2.90 seconds
  50 parallel clients
  82 bytes payload
  keep alive: 1

44.79% <= 0 milliseconds
62.61% <= 1 milliseconds
89.99% <= 2 milliseconds
96.98% <= 3 milliseconds
98.90% <= 4 milliseconds
99.41% <= 5 milliseconds
99.68% <= 6 milliseconds
99.81% <= 7 milliseconds
99.83% <= 8 milliseconds
99.84% <= 9 milliseconds
99.86% <= 13 milliseconds
99.90% <= 14 milliseconds
99.91% <= 15 milliseconds
99.91% <= 16 milliseconds
99.92% <= 17 milliseconds
99.94% <= 18 milliseconds
99.95% <= 19 milliseconds
99.96% <= 22 milliseconds
99.97% <= 34 milliseconds
99.99% <= 35 milliseconds
```

```
100.00% <= 39 milliseconds
34505.36 requests per second

====== GET (point) ======
  100000 requests completed in 1.38 seconds
  50 parallel clients
  52 bytes payload
  keep alive: 1

83.84% <= 0 milliseconds
97.73% <= 1 milliseconds
99.01% <= 2 milliseconds
99.43% <= 3 milliseconds
99.67% <= 4 milliseconds
99.76% <= 5 milliseconds
99.86% <= 6 milliseconds
99.89% <= 7 milliseconds
99.94% <= 8 milliseconds
99.96% <= 9 milliseconds
99.97% <= 10 milliseconds
99.97% <= 12 milliseconds
99.98% <= 13 milliseconds
99.99% <= 20 milliseconds
100.00% <= 25 milliseconds
72687.47 requests per second
```

In this case, we can see that our system has a throughput of 34505.36 requests per second for creation and update requests. That means, that with an update interval of 1 second, the system can handle 34505 subscribed Actors. Moreover, the system has a throughput of 72687.47 requests per second for retrieval requests.

Area Creation / Update / Retrieval

Another important use of the system, is the creation, update and retrieval handling of an Area.

```
====== SET (rect) ======
  100000 requests completed in 3.01 seconds
  50 parallel clients
  114 bytes payload
  keep alive: 1

44.89% <= 0 milliseconds
63.24% <= 1 milliseconds
89.45% <= 2 milliseconds
96.50% <= 3 milliseconds
98.46% <= 4 milliseconds
99.11% <= 5 milliseconds
99.48% <= 6 milliseconds
99.62% <= 7 milliseconds
99.68% <= 8 milliseconds
99.72% <= 9 milliseconds
99.73% <= 10 milliseconds
99.75% <= 11 milliseconds
99.77% <= 14 milliseconds
99.81% <= 15 milliseconds
99.83% <= 16 milliseconds
99.88% <= 19 milliseconds
99.90% <= 20 milliseconds
99.90% <= 21 milliseconds
99.92% <= 25 milliseconds
99.94% <= 26 milliseconds
```

```
99.95% <= 27 milliseconds
99.98% <= 35 milliseconds
100.00% <= 36 milliseconds
100.00% <= 37 milliseconds
33269.42 requests per second

====== GET (rect) ======
  100000 requests completed in 1.74 seconds
  50 parallel clients
  55 bytes payload
  keep alive: 1

73.59% <= 0 milliseconds
94.20% <= 1 milliseconds
97.50% <= 2 milliseconds
98.59% <= 3 milliseconds
99.30% <= 4 milliseconds
99.59% <= 5 milliseconds
99.71% <= 6 milliseconds
99.83% <= 7 milliseconds
99.88% <= 8 milliseconds
99.92% <= 9 milliseconds
99.93% <= 10 milliseconds
99.95% <= 12 milliseconds
99.96% <= 14 milliseconds
99.98% <= 15 milliseconds
99.99% <= 16 milliseconds
99.99% <= 17 milliseconds
100.00% <= 18 milliseconds
```

```
57543.51 requests per second
```

In this case, we can see that our system has a throughput of 33269.42 requests per second for creation and update requests. That means, that with an update interval of 1 second, the system can handle 33269 Areas. Moreover, the system has a throughput of 57543.51 requests per second for retrieval requests.

Chapter 5

Conclusions / Future Work

5.1 Conclusions

As we saw in the previous chapter, even the most extensive of scenarios are really far from proving to be an actual stress for the system. The chosen technologies, make sure that the system always compiles to a binary, and thus can always have the best possible resource management on every platform that it is deployed. This makes the system easily deployable even to cheap embedded systems, such as a RaspberryPi, meaning that the main node of the system can be deployed on the fly on the field. Moreover, since the system does not depend on any specific network infrastructure, it can be used on any circumstance provided that the administrator provides some kind of networking. For example, in case where there is no landline internet connectivity, mobile networking can be used. Even in a case where none of the previous mentioned networks is available, networks such as bluetooth low energy (ble) or even lora can be used to achieve network connectivity. This makes the system reliable, portable and more than able to handle thousands of requests per second, even on cheap hardware.

5.2 Future Work

The system at the moment is a self sustainable platform that can handle multiple data sources at the same time. The main concept that it was built around was real-time performance. For this reason, one of the main sacrifices that was made, was to use the primary memory of the system as the only data storage facility. In the future, it would be really interesting to pursue a somewhat different approach,where a hybrid storage would be used, with some level of concurrency and caching between the primary and secondary memory of the system. Moreover, it would be interesting to see the performance of the systems on world wide events where the possible actors and areas would scale up to billions. If that was to be achieved, then this system could be a viable alternative to even the basic functionality of all traditional GIS systems.

References

- Valacich, Joe and Christoph Schneider (2010) Information Systems Today: Managing in the Digital World: Prentice Hall.
- Longley, P, Michael F. Goodchild, David J. Maguire and David W. Rhind (2010) Geographic Information Systems and Science. 3rd ed: Wiley.
- Anthopoulos, Leonidas G, Efrosini Kostavara and John-Paris Pantouvakis (2013) "An Effective Disaster Recovery Model for Construction Projects," Procedia-Social and Behavioral Sciences, 74:21 – 30.
- United States Department of Homeland Security (2013) National Response Framework 2nd edition
- Federal Emergency Management Agency (FEMA) (2008) Emergency Support Function #5 – Emergency Management Annex.
- Kevany, M. J. (2003) "GIS in the World Trade Center Attack – Trial by Fire," Computers, Environment and Urban Systems, 27(6):571 – 583.
- Kobayashi, Audrey (2012) Geographies of Peace and Armed Conflict. New York: Routledge.
- Ross, Michael L (2004) "What Do We Know about Natural Resources and Civil War ?" Journal of peace research 41(3):337 – 356.
- Baechler, Günther, Kurt R Spillmann and Mohamed Suliman (2002) Transformation of Resource Conflicts: Approach and Instruments. Bern, Berlin, Bruxelles, Frankfurt/M., New York, Oxford, Wien: Peter Lang.
- Vandergeest, Peter and Nancy Lee Peluso (1995) "Territorialization and State Power in Thailand," Theory and society, 24(3):385 – 426.

- Theisen, Ole Magnus (2012) “Climate Clashes ? Weather Variability, Land Pressure, and Organized Violence in Kenya, 1989 – 2004,” Journal of peace research, 49(1):81 – 96.
- Endsley, M. R. (1995) “Toward a Theory of Situation Awareness in Dynamic Systems,” Human Factors, 37(1):32 – 64.
- Endsley, M. R. and Garland D. J. (Eds.) (2000) “Situation Awareness Analysis and Measurement,” Mahwah, NJ: Lawrence Erlbaum Associates.
- Breen, Joseph J. and David R Parrish (2013) “GIS in Emergency Management Cultures: An Empirical Approach to Understanding Inter-and Intra-Agency Communication During Emergencies,” Journal of Homeland Security and Emergency Management, 10(2):477 – 495.
- Zerger, A. and DI Smith (2003) “Impediments to Using GIS for Real-Time Disaster Decision Support,” Computers, Environment and Urban Systems, 27(2):123 – 141.
- Kevany, Michael J. (2005) “Geo-Information for Disaster Management: Lessons From 9/11.” In Geo-information for disaster management : Springer.
- Diehl, Stefan and Jene van der Heide (2005) “Geo Information Breaks through Sector Think.” In: Geo-Information for Disaster Management. Springer.
- National Research Council (2007a) Improving Disaster Management: The Role of IT in Mitigation, Preparedness, Response, and Recovery. In: (Committee on Using Information Technology to Enhance Disaster Management, ed.) Washington, DC: National Academies Press.
- National Research Council (2007b) Successful Response Starts With a Map: Improving Geospatial Support for Disaster Management. In: (Committee on Planning for Catastrophe: A Blueprint for Improving Geospatial Data Tools and Infrastructure, ed.) Washington, DC: National Academies Press.
- Neuvel, JeroenM M., HenkJ Scholten and Adri van den Brink (2012) “From Spatial Data to Synchronised Actions: The Network-centric Organisation of Spatial Decision Support for Risk and Emergency Management,” Applied Spatial Analysis and Policy, 5(1):51 – 72.
- Mills, Jacqueline, Andrew Curtis, John C Pine and Barrett Kennedy (2008) “The Clearinghouse Concept: A Model for Geospatial Data Centralization and Dissemination in a Disaster,” Disasters, 32(3):467 – 479.

- Köhler, Petra (2005) “User-Oriented Provision of Geo-Information in Disaster Management: Potentials of Spatial Data Infrastructures Considering Brandenburg/Germany as an Example.” In: Geo-information for Disaster Management. Springer.

- Tomaszweski, Brian & Judex, Michael & Szarzynski, Jörg & Radestock, Christine & Wirkus, Lars. (2015). Geographic Information Systems for Disaster Response: A Review. Journal of Homeland Security and Emergency Management. Volume 12. 571-602. 10.1515/jhsem-2014-0082.

- Johnson, G. O. (1992) “GIS Applications in Emergency Management,” URISA Journal, 4:66 – 72.

- Coppock, J Terry (1995) “GIS and Natural Hazards: An Overview from a GIS Perspective.” In: Geographical information systems in assessing natural hazards, Springer.

- Johnson, Glenn O (1995) “GIS applications in emergency management.” In: Computer Supported Risk Management: Topics in Safety, Risk, Reliability and Quality. Springer.

- Dash, Nicole (1997) “The Use of Geographical Information Systems in Disaster Research,” International Journal of Mass Emergencies and Disasters, 15(1):135 – 146.

- Cova, Thomas J (1999) “GIS in Emergency Management,” Geographical information systems, 2:845 – 858.

- Cutter, Susan L (2003) “GI Science, Disasters, and Emergency Management,” Transactions in GIS, 7(4):439 – 446.

- Lee, J. and Sisi Zlatanova (2008) “A 3D Data Model and Topological Analyses for Emergency Response in Urban Areas.” In: (Z. L. eds.) Geospatial information technology for emergency response (ISPRS book series). London, UK: Taylor & Francis Group.

- MacEachren, Alan M and Guoray Cai (2006) “Supporting Group Work in Crisis Management: Visually Mediated Human-GIS-Human Dialogue,” Environment and Planning B Planning and Design, 33(3):435.

- Cova, Thomas J, Philip E Dennison, Tae H Kim and Max A Moritz (2005) “Setting Wildfire Evacuation Trigger Points using Fire Spread Modeling and GIS,” Transactions in GIS, 9(4):603 – 617.

- Garb, Jane L, Robert G Cromley and Richard B Wait (2007) “Estimating Populations at

Risk for Disaster Preparedness and Response," Journal of Homeland Security and Emergency Management, 4(1).
- Zlatanova, Sisi and Arta Dilo (2010) "A Data Model for Operational and Situational Infomration in Emergency Response: The Dutch Case." In: Proceedings of the Gi4DM Conference – Geomatics for Disaster Management, February 2010, Torino.
- Frigerio, Ivan, Stefano Roverato and Amttia Amicis (2013) "A Proposal for a Geospatial Database to Support Emergency Management," Journal of Geographic Information System, 5(4):396 – 403.
- Goodchild, Michael F and J Alan Glennon (2010) "Crowdsourcing Geographic Information for Disaster Response: A Research Frontier," International Journal of Digital Earth, 3(3):231 – 241.
- Tomaszewski, Brian (2011) "Situation Awareness and Virtual Globes: Applications for Disaster Management," Computers and Geosciences, 37:86 – 92.
- Liu, S. B. and L. Palen (2010) "The New Cartographers: Crisis Map Mashups and the Emergence of Neogeographic Practice," Cartography and Geographic Information Science, 37(1):69 – 90.
- Robinson, Anthony C, Robert E Roth and Alan M MacEachren (2011) "Understanding User Needs for Map Symbol Standards in Emergency Management,"Journal of Homeland Security and Emergency Management, 8(1).
- MacEachren, A. M., A. Jaiswal, A. C. Robinson, S. Pezanowski, A. Savelyev, P. Mitra, X. Zhang and J. Blanford (2011) SensePlace2: GeoTwitter Analytics Support for Situational Awareness. Paper read at IEEE Conference on Visual Analytics Science and Technology (IEEE VAST), at Providence, RI.
- Crooks, Andrew, Arie Croitoru, Anthony Stefanidis and Jacek Radzikowski (2013) "#Earthquake: Twitter as a Distributed Sensor System," Transactions in GIS, 17(1):124 – 147.
- Annunziato, Alessandro, Simone Gadez, Daniele Al Galliano, Roberto Guana and Francisco Igualada (2010) "Field Tracking Tool: A Collaborative Framwork from the Field to the Decision Makers." In: (M. Konecny, S. Zlatanova and T. L. Bandrova, eds.) Geographic Information and Cartography for Risk and Crisis Management. Towards Better Solutions.

Beidelberg, Dortrecht, London, New York: Springer. - Frassl, Martin, Michael Lichtenstern, Mohammed Khider and Michael Angermann (2010) "Developing a System for Information Management in Disaster Relief – Methodology and Requirements." In: Proceedings of the 7th International ISCRAM Conference – Seattle USA, May 2010.
- Kiltz, Linda and Richard Smith (2011) "Experimenting with GIS in Doing Damage Assessments: A Trial Run at Disaster City," Journal of Homeland Security and Emergency Management, 8(1).
- Goodchild, Michael F (2007) "Citizens as Sensors: The World of Volunteered Geography," Geo Journal, 69(4):211 – 221.
- Haklay, Mordechai and Patrick Weber (2008) "Openstreetmap: User-Generated Street Maps," Pervasive Computing, IEEE, 7(4):12 – 18.
- Humanitarian OpenStreetMap Team (n.d.). Haiti. Available at: http://hot.openstreetmap.org/projects/haiti-2 .
- Okolloh, Ory (2009) "Ushahidi, or 'Testimony' : Web 2.0 Tools for Crowdsourcing Crisis Information," Participatory learning and action, 59(1):65 – 70.
- Meier, Patrick (2012) "Crisis Mapping in Action: How Open Source Software and Global Volunteer Networks are Changing the World, One Map at a Time," Journal of Map & Geography Libraries, 8(2):89 – 100.
- Hodson, Hal (2014) "Mapping in a Crisis," New Scientist, 222(2964):19.
- Zastrow, Mark (2014) "Crisis mappers turn to citizen scientists," Nature, 515(7527):321.
- Mandl, Daniel, Stuart Frye, Pat Cappelaere, Matthew Handy, Fritz Policelli, M Katjizeu, Guido Van Langenhove, Guy Aube, J Saulnier, Rob Sohlberg, Julie Silva, Nataliia Kussul, Sergii Skakun, Stephen Ungar, Robert Grossman and Joerg Szarzynski (2013) "Use of the Earth Observing One (EO-1) Satellite for the Namibia SensorWeb flood Early Warning Pilot," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing – Ten Years in Space, 6(2):298 – 308.
- Kelmelis, John A., Lee Schwartz, Carol Christian, Melba Crawford and Dennis King (2006) "Use of Geographic Information in Response to the Sumatra-Andaman Response to the Sumatra-Andaman Earthquake and Indian Ocean Earthquake and Indian Ocean Tsunami of

December 26, 2004," Photogrammetric Engineering and Remote Sensing, 72(8):862 – 877.
- Nourbakhsh, I, R Sargent, A Wright, K Cramer, B McClendon and M Jones (2006) "Mapping Disaster Zones," Nature, 439(7078):787 – 788.
- Robert Backhaus, Lorant Czaran, Natalie Epler, Michael Leitgab, David Stevens and Joerg Szarzynski (2011) The 4C-Challenge: Communication – Coordination – Cooperation – Capacity Development. Selected contributions to the Fourth United Nations International UN-SPIDER Bonn Workshop on Disaster Management and Space Technology, 2010 - 2011. Available at: http://www.un-spider.org/4c-challengecommunication-coordination-cooperation-capacity-development.
- Ravan, S., J. Szarzynski and D. Stevens (2011) Space technology to support disaster risk reduction and emergency medical and rescue teams. Paper read at Proceedings of the 17th World Congress on Disaster and Emergency Medicine, at Beijing, China.
- Szarzynski, J. and D. Stevens (2009) "Space-Based Solutions for Disaster and Emergency Medicine," Prehosp Disasters Medicine, 24(2):79.
- Quaritsch, Markus, Robert Kuschnig, Hermann Hellwagner, Bernhard Rinner, A Adria and U Klagenfurt (2011) Fast Aerial Image Acquisition and Mosaicking for Emergency Response Operations by Collaborative UAVs. Paper read at Proceedings of the 8th International ISCRAM Conference-Lisbon.
- Abdelkader, Mohamed, Mohammad Shaqura, Christian G Claudel and Wail Gueaieb (2013) A UAV Based System for Real Time Flash Flood Monitoring in Desert Environments using Lagrangian Microsensors . Paper read at 2013 International Conference on Unmanned Aircraft Systems (ICUAS) at Atlanta, Georgia.
- Lohr, Steve (2011) Online Mapping Shows Potential to Transform Relief Efforts. Available at: http://www.nytimes.com/2011/03/28/business/28map.html?_r=0.
- MapAction (2011a) Field Guide to Humanitarian Mapping. London.
- MapAction (2011b) What We Do. Available at: http://www.mapaction.org/about/about-us.html.
- Diehl, Stefan, Jeroen Neuvel, S. Zlatanova and H. Scholten (2006) "Investigation of User Requirements in the Emergency Response Sector: the Dutch Case." In: Second Symposium

on Geo-information for Disaster management (Gi4DM).

Appendix A

Full System Benchmark Log

```
PING: 0.00


PING: 108015.27


PING: 110948.61


PING: 111578.36


PING: 112112.49


PING: 111930.66
====== PING ======
  100000 requests completed in 0.89 seconds
  50 parallel clients
  14 bytes payload
  keep alive: 1

96.35% <= 0 milliseconds
99.33% <= 1 milliseconds
```

```
99.82% <= 2 milliseconds
99.89% <= 3 milliseconds
99.94% <= 4 milliseconds
99.97% <= 5 milliseconds
99.98% <= 6 milliseconds
99.99% <= 8 milliseconds
100.00% <= 11 milliseconds
111930.66 requests per second


SET (point): 0.00

SET (point): 31738.46

SET (point): 34092.94

SET (point): 35111.16

SET (point): 33211.25

SET (point): 33917.75

SET (point): 33941.94

SET (point): 34179.67

SET (point): 34216.35

SET (point): 34316.59
SET (point): 34396.44


SET (point): 33775.62


SET (point): 34042.70


SET (point): 34359.10


SET (point): 34448.13


SET (point): 34505.36
====== SET (point) ======
  100000 requests completed in 2.90 seconds
  50 parallel clients
  82 bytes payload
  keep alive: 1

44.79% <= 0 milliseconds
62.61% <= 1 milliseconds
89.99% <= 2 milliseconds
96.98% <= 3 milliseconds
98.90% <= 4 milliseconds
99.41% <= 5 milliseconds
99.68% <= 6 milliseconds
99.81% <= 7 milliseconds
99.83% <= 8 milliseconds
99.84% <= 9 milliseconds
99.86% <= 13 milliseconds
```

```
99.90% <= 14 milliseconds
99.91% <= 15 milliseconds
99.91% <= 16 milliseconds
99.92% <= 17 milliseconds
99.94% <= 18 milliseconds
99.95% <= 19 milliseconds
99.96% <= 22 milliseconds
99.97% <= 34 milliseconds
99.99% <= 35 milliseconds
100.00% <= 39 milliseconds
34505.36 requests per second


SET (rect): 0.00

SET (rect): 34006.02

SET (rect): 34649.42

SET (rect): 30008.48

SET (rect): 31193.48

SET (rect): 31784.46

SET (rect): 32456.64

SET (rect): 32678.28
SET (rect): 32978.58


SET (rect): 33199.14


SET (rect): 33524.86


SET (rect): 32932.17


SET (rect): 33137.22


SET (rect): 33141.33


SET (rect): 33224.59


SET (rect): 33272.82


SET (rect): 33269.42
====== SET (rect) ======
  100000 requests completed in 3.01 seconds
  50 parallel clients
  114 bytes payload
  keep alive: 1

44.89% <= 0 milliseconds
63.24% <= 1 milliseconds
89.45% <= 2 milliseconds
96.50% <= 3 milliseconds
98.46% <= 4 milliseconds
99.11% <= 5 milliseconds
```

```
99.48% <= 6 milliseconds
99.62% <= 7 milliseconds
99.68% <= 8 milliseconds
99.72% <= 9 milliseconds
99.73% <= 10 milliseconds
99.75% <= 11 milliseconds
99.77% <= 14 milliseconds
99.81% <= 15 milliseconds
99.83% <= 16 milliseconds
99.88% <= 19 milliseconds
99.90% <= 20 milliseconds
99.90% <= 21 milliseconds
99.92% <= 25 milliseconds
99.94% <= 26 milliseconds
99.95% <= 27 milliseconds
99.98% <= 35 milliseconds
100.00% <= 36 milliseconds
100.00% <= 37 milliseconds
33269.42 requests per second


SET (string): 0.00


SET (string): 39329.63


SET (string): 40599.97


SET (string): 38512.84
SET (string): 38340.48


SET (string): 38781.69


SET (string): 39372.82


SET (string): 39715.18


SET (string): 40089.51


SET (string): 40354.17


SET (string): 40371.95


SET (string): 39818.02


SET (string): 39646.50


SET (string): 39702.54
====== SET (string) ======
  100000 requests completed in 2.52 seconds
  50 parallel clients
  64 bytes payload
  keep alive: 1

51.94% <= 0 milliseconds
69.39% <= 1 milliseconds
92.13% <= 2 milliseconds
97.65% <= 3 milliseconds
```

```
99.06% <= 4 milliseconds
99.46% <= 5 milliseconds
99.69% <= 6 milliseconds
99.78% <= 7 milliseconds
99.80% <= 8 milliseconds
99.85% <= 10 milliseconds
99.89% <= 15 milliseconds
99.90% <= 16 milliseconds
99.90% <= 17 milliseconds
99.91% <= 21 milliseconds
99.93% <= 22 milliseconds
99.95% <= 23 milliseconds
99.95% <= 24 milliseconds
99.96% <= 38 milliseconds
99.98% <= 42 milliseconds
100.00% <= 43 milliseconds
100.00% <= 44 milliseconds
39702.54 requests per second


GET (point): 0.00


GET (point): 76506.00


GET (point): 76196.41


GET (point): 69811.44


GET (point): 69311.24
GET (point): 71827.93


GET (point): 72718.23


GET (point): 72687.47
====== GET (point) ======
  100000 requests completed in 1.38 seconds
  50 parallel clients
  52 bytes payload
  keep alive: 1

83.84% <= 0 milliseconds
97.73% <= 1 milliseconds
99.01% <= 2 milliseconds
99.43% <= 3 milliseconds
99.67% <= 4 milliseconds
99.76% <= 5 milliseconds
99.86% <= 6 milliseconds
99.89% <= 7 milliseconds
99.94% <= 8 milliseconds
99.96% <= 9 milliseconds
99.97% <= 10 milliseconds
99.97% <= 12 milliseconds
99.98% <= 13 milliseconds
99.99% <= 20 milliseconds
100.00% <= 25 milliseconds
72687.47 requests per second
```

```
GET (rect): 0.00

GET (rect): 64093.71

GET (rect): 61180.11

GET (rect): 57239.77

GET (rect): 53574.60

GET (rect): 56572.00

GET (rect): 58110.01

GET (rect): 56068.14

GET (rect): 56860.98

GET (rect): 57543.51
====== GET (rect) ======
  100000 requests completed in 1.74 seconds
  50 parallel clients
  55 bytes payload
  keep alive: 1

73.59% <= 0 milliseconds
94.20% <= 1 milliseconds
97.50% <= 2 milliseconds
```

```
98.59% <= 3 milliseconds
99.30% <= 4 milliseconds
99.59% <= 5 milliseconds
99.71% <= 6 milliseconds
99.83% <= 7 milliseconds
99.88% <= 8 milliseconds
99.92% <= 9 milliseconds
99.93% <= 10 milliseconds
99.95% <= 12 milliseconds
99.96% <= 14 milliseconds
99.98% <= 15 milliseconds
99.99% <= 16 milliseconds
99.99% <= 17 milliseconds
100.00% <= 18 milliseconds
57543.51 requests per second


GET (string): 0.00


GET (string): 81430.20


GET (string): 77645.99


GET (string): 70778.97


GET (string): 72358.89


GET (string): 74212.30
GET (string): 75365.90


GET (string): 75456.72
====== GET (string) ======
  100000 requests completed in 1.33 seconds
  50 parallel clients
  55 bytes payload
  keep alive: 1

85.23% <= 0 milliseconds
97.54% <= 1 milliseconds
99.00% <= 2 milliseconds
99.49% <= 3 milliseconds
99.72% <= 4 milliseconds
99.80% <= 5 milliseconds
99.86% <= 6 milliseconds
99.88% <= 7 milliseconds
99.92% <= 8 milliseconds
99.95% <= 9 milliseconds
99.95% <= 10 milliseconds
99.99% <= 11 milliseconds
99.99% <= 12 milliseconds
100.00% <= 17 milliseconds
75456.72 requests per second



INTERSECTS (intersects-circle 1km): 0.00


INTERSECTS (intersects-circle 1km): 30595.85
INTERSECTS (intersects-circle 1km): 36461.52

INTERSECTS (intersects-circle 1km): 37882.21

INTERSECTS (intersects-circle 1km): 38060.74

INTERSECTS (intersects-circle 1km): 39410.56

INTERSECTS (intersects-circle 1km): 39575.30

INTERSECTS (intersects-circle 1km): 39311.11

INTERSECTS (intersects-circle 1km): 40175.53

INTERSECTS (intersects-circle 1km): 40109.72

INTERSECTS (intersects-circle 1km): 40984.49

INTERSECTS (intersects-circle 1km): 39839.63

INTERSECTS (intersects-circle 1km): 40165.38

INTERSECTS (intersects-circle 1km): 40356.06
====== INTERSECTS (intersects-circle 1km) ======
  100000 requests completed in 2.48 seconds
  50 parallel clients
  98 bytes payload
  keep alive: 1
```

```
53.22% <= 0 milliseconds
85.19% <= 1 milliseconds
93.77% <= 2 milliseconds
96.73% <= 3 milliseconds
98.09% <= 4 milliseconds
98.79% <= 5 milliseconds
99.19% <= 6 milliseconds
99.52% <= 7 milliseconds
99.69% <= 8 milliseconds
99.79% <= 9 milliseconds
99.90% <= 10 milliseconds
99.93% <= 11 milliseconds
99.97% <= 12 milliseconds
99.98% <= 13 milliseconds
99.99% <= 15 milliseconds
100.00% <= 37 milliseconds
40356.06 requests per second


INTERSECTS (intersects-circle 10km): 0.00


INTERSECTS (intersects-circle 10km): 24340.17


INTERSECTS (intersects-circle 10km): 31356.64


INTERSECTS (intersects-circle 10km): 28802.86


INTERSECTS (intersects-circle 10km): 30116.22
INTERSECTS (intersects-circle 10km): 28082.19

INTERSECTS (intersects-circle 10km): 29462.23

INTERSECTS (intersects-circle 10km): 28768.89

INTERSECTS (intersects-circle 10km): 29586.96

INTERSECTS (intersects-circle 10km): 29189.08

INTERSECTS (intersects-circle 10km): 29016.63

INTERSECTS (intersects-circle 10km): 29373.96

INTERSECTS (intersects-circle 10km): 28749.38

INTERSECTS (intersects-circle 10km): 29223.28

INTERSECTS (intersects-circle 10km): 28864.31

INTERSECTS (intersects-circle 10km): 29116.93

INTERSECTS (intersects-circle 10km): 29068.25

INTERSECTS (intersects-circle 10km): 29110.24

INTERSECTS (intersects-circle 10km): 29146.15
====== INTERSECTS (intersects-circle 10km) ======
  100000 requests completed in 3.43 seconds
  50 parallel clients
  99 bytes payload
  keep alive: 1

41.24% <= 0 milliseconds
70.26% <= 1 milliseconds
85.40% <= 2 milliseconds
92.49% <= 3 milliseconds
95.98% <= 4 milliseconds
97.53% <= 5 milliseconds
98.28% <= 6 milliseconds
98.82% <= 7 milliseconds
99.10% <= 8 milliseconds
99.42% <= 9 milliseconds
99.62% <= 10 milliseconds
99.72% <= 11 milliseconds
99.80% <= 12 milliseconds
99.86% <= 13 milliseconds
99.89% <= 14 milliseconds
99.92% <= 15 milliseconds
99.94% <= 16 milliseconds
99.96% <= 17 milliseconds
99.96% <= 18 milliseconds
99.97% <= 19 milliseconds
99.98% <= 20 milliseconds
99.99% <= 22 milliseconds
100.00% <= 35 milliseconds
29146.15 requests per second
```

```
INTERSECTS (intersects-circle 100km): 0.00

INTERSECTS (intersects-circle 100km): 5091.95

INTERSECTS (intersects-circle 100km): 5852.11

INTERSECTS (intersects-circle 100km): 5880.37

INTERSECTS (intersects-circle 100km): 5820.82

INTERSECTS (intersects-circle 100km): 5838.70

INTERSECTS (intersects-circle 100km): 5813.94

INTERSECTS (intersects-circle 100km): 5814.14

INTERSECTS (intersects-circle 100km): 5710.52

INTERSECTS (intersects-circle 100km): 5754.45

INTERSECTS (intersects-circle 100km): 5790.19

INTERSECTS (intersects-circle 100km): 5845.09

INTERSECTS (intersects-circle 100km): 5933.63

INTERSECTS (intersects-circle 100km): 5944.90
INTERSECTS (intersects-circle 100km): 5921.44

INTERSECTS (intersects-circle 100km): 5920.93

INTERSECTS (intersects-circle 100km): 5944.59

INTERSECTS (intersects-circle 100km): 5979.25

INTERSECTS (intersects-circle 100km): 5966.94

INTERSECTS (intersects-circle 100km): 5988.26

INTERSECTS (intersects-circle 100km): 5924.30

INTERSECTS (intersects-circle 100km): 5919.82

INTERSECTS (intersects-circle 100km): 5895.34

INTERSECTS (intersects-circle 100km): 5919.23

INTERSECTS (intersects-circle 100km): 5893.76

INTERSECTS (intersects-circle 100km): 5895.04

INTERSECTS (intersects-circle 100km): 5881.92

INTERSECTS (intersects-circle 100km): 5899.96
INTERSECTS (intersects-circle 100km): 5861.72

INTERSECTS (intersects-circle 100km): 5839.68

INTERSECTS (intersects-circle 100km): 5819.77

INTERSECTS (intersects-circle 100km): 5826.89

INTERSECTS (intersects-circle 100km): 5814.89

INTERSECTS (intersects-circle 100km): 5823.01

INTERSECTS (intersects-circle 100km): 5818.49

INTERSECTS (intersects-circle 100km): 5822.56

INTERSECTS (intersects-circle 100km): 5802.95

INTERSECTS (intersects-circle 100km): 5830.88

INTERSECTS (intersects-circle 100km): 5833.12

INTERSECTS (intersects-circle 100km): 5831.40

INTERSECTS (intersects-circle 100km): 5859.25

INTERSECTS (intersects-circle 100km): 5860.34

INTERSECTS (intersects-circle 100km): 5875.32
INTERSECTS (intersects-circle 100km): 5866.85

INTERSECTS (intersects-circle 100km): 5852.82

INTERSECTS (intersects-circle 100km): 5841.43

INTERSECTS (intersects-circle 100km): 5869.62

INTERSECTS (intersects-circle 100km): 5865.83

INTERSECTS (intersects-circle 100km): 5879.18

INTERSECTS (intersects-circle 100km): 5876.45

INTERSECTS (intersects-circle 100km): 5870.40

INTERSECTS (intersects-circle 100km): 5871.35

INTERSECTS (intersects-circle 100km): 5873.35

INTERSECTS (intersects-circle 100km): 5864.91

INTERSECTS (intersects-circle 100km): 5865.74

INTERSECTS (intersects-circle 100km): 5869.35

INTERSECTS (intersects-circle 100km): 5856.90
INTERSECTS (intersects-circle 100km): 5862.30

INTERSECTS (intersects-circle 100km): 5855.31

INTERSECTS (intersects-circle 100km): 5863.14

INTERSECTS (intersects-circle 100km): 5858.77

INTERSECTS (intersects-circle 100km): 5859.97

INTERSECTS (intersects-circle 100km): 5864.42

INTERSECTS (intersects-circle 100km): 5872.46

INTERSECTS (intersects-circle 100km): 5868.17

INTERSECTS (intersects-circle 100km): 5882.22

INTERSECTS (intersects-circle 100km): 5883.41

INTERSECTS (intersects-circle 100km): 5888.85

INTERSECTS (intersects-circle 100km): 5882.19

INTERSECTS (intersects-circle 100km): 5880.56

INTERSECTS (intersects-circle 100km): 5879.98

INTERSECTS (intersects-circle 100km): 5877.37
INTERSECTS (intersects-circle 100km): 5865.95

INTERSECTS (intersects-circle 100km): 5866.20

INTERSECTS (intersects-circle 100km): 5868.72

INTERSECTS (intersects-circle 100km): 5873.91

INTERSECTS (intersects-circle 100km): 5869.62

INTERSECTS (intersects-circle 100km): 5883.72

INTERSECTS (intersects-circle 100km): 5888.48

INTERSECTS (intersects-circle 100km): 5887.35

INTERSECTS (intersects-circle 100km): 5880.61

INTERSECTS (intersects-circle 100km): 5878.54

INTERSECTS (intersects-circle 100km): 5883.88

INTERSECTS (intersects-circle 100km): 5876.92

INTERSECTS (intersects-circle 100km): 5878.08

INTERSECTS (intersects-circle 100km): 5873.89
INTERSECTS (intersects-circle 100km): 5873.03
====== INTERSECTS (intersects-circle 100km) ======
  100000 requests completed in 17.03 seconds
  50 parallel clients
  100 bytes payload
  keep alive: 1

28.84% <= 0 milliseconds
32.04% <= 1 milliseconds
34.88% <= 2 milliseconds
38.40% <= 3 milliseconds
42.67% <= 4 milliseconds
47.34% <= 5 milliseconds
52.17% <= 6 milliseconds
57.12% <= 7 milliseconds
61.85% <= 8 milliseconds
66.41% <= 9 milliseconds
70.49% <= 10 milliseconds
74.27% <= 11 milliseconds
77.80% <= 12 milliseconds
80.99% <= 13 milliseconds
83.90% <= 14 milliseconds
86.48% <= 15 milliseconds
88.70% <= 16 milliseconds
90.58% <= 17 milliseconds
92.11% <= 18 milliseconds
93.28% <= 19 milliseconds
94.16% <= 20 milliseconds
94.85% <= 21 milliseconds
```

```
95.42% <= 22 milliseconds
95.87% <= 23 milliseconds
96.26% <= 24 milliseconds
96.57% <= 25 milliseconds
96.83% <= 26 milliseconds
97.08% <= 27 milliseconds
97.29% <= 28 milliseconds
97.47% <= 29 milliseconds
97.63% <= 30 milliseconds
97.77% <= 31 milliseconds
97.91% <= 32 milliseconds
98.06% <= 33 milliseconds
98.20% <= 34 milliseconds
98.31% <= 35 milliseconds
98.42% <= 36 milliseconds
98.52% <= 37 milliseconds
98.61% <= 38 milliseconds
98.70% <= 39 milliseconds
98.78% <= 40 milliseconds
98.88% <= 41 milliseconds
98.94% <= 42 milliseconds
99.02% <= 43 milliseconds
99.07% <= 44 milliseconds
99.13% <= 45 milliseconds
99.19% <= 46 milliseconds
99.24% <= 47 milliseconds
99.29% <= 48 milliseconds
99.33% <= 49 milliseconds
99.37% <= 50 milliseconds
```

```
99.41% <= 51 milliseconds
99.45% <= 52 milliseconds
99.49% <= 53 milliseconds
99.52% <= 54 milliseconds
99.55% <= 55 milliseconds
99.58% <= 56 milliseconds
99.62% <= 57 milliseconds
99.64% <= 58 milliseconds
99.67% <= 59 milliseconds
99.70% <= 60 milliseconds
99.71% <= 61 milliseconds
99.73% <= 62 milliseconds
99.74% <= 63 milliseconds
99.76% <= 64 milliseconds
99.77% <= 65 milliseconds
99.80% <= 66 milliseconds
99.82% <= 67 milliseconds
99.84% <= 68 milliseconds
99.85% <= 69 milliseconds
99.86% <= 70 milliseconds
99.86% <= 71 milliseconds
99.87% <= 72 milliseconds
99.88% <= 73 milliseconds
99.89% <= 74 milliseconds
99.90% <= 76 milliseconds
99.91% <= 77 milliseconds
99.92% <= 80 milliseconds
99.93% <= 83 milliseconds
99.94% <= 87 milliseconds
```

```
99.95% <= 88 milliseconds
99.96% <= 94 milliseconds
99.97% <= 102 milliseconds
99.98% <= 113 milliseconds
99.99% <= 160 milliseconds
100.00% <= 266 milliseconds
5873.03 requests per second


INTERSECTS (intersects-bounds 1km): 0.00


INTERSECTS (intersects-bounds 1km): 61146.91


INTERSECTS (intersects-bounds 1km): 52777.40


INTERSECTS (intersects-bounds 1km): 54825.81


INTERSECTS (intersects-bounds 1km): 56767.41


INTERSECTS (intersects-bounds 1km): 57115.82


INTERSECTS (intersects-bounds 1km): 53263.02


INTERSECTS (intersects-bounds 1km): 54580.54


INTERSECTS (intersects-bounds 1km): 55270.36


INTERSECTS (intersects-bounds 1km): 55732.83
====== INTERSECTS (intersects-bounds 1km) ======
  100000 requests completed in 1.79 seconds
  50 parallel clients
  117 bytes payload
  keep alive: 1

69.89% <= 0 milliseconds
94.84% <= 1 milliseconds
98.07% <= 2 milliseconds
98.95% <= 3 milliseconds
99.34% <= 4 milliseconds
99.45% <= 5 milliseconds
99.57% <= 6 milliseconds
99.77% <= 7 milliseconds
99.90% <= 8 milliseconds
99.94% <= 9 milliseconds
99.95% <= 10 milliseconds
99.97% <= 11 milliseconds
99.98% <= 12 milliseconds
99.99% <= 14 milliseconds
100.00% <= 21 milliseconds
55732.83 requests per second


INTERSECTS (intersects-bounds 10km): 0.00

INTERSECTS (intersects-bounds 10km): 41391.96

INTERSECTS (intersects-bounds 10km): 51561.56
INTERSECTS (intersects-bounds 10km): 54916.46


INTERSECTS (intersects-bounds 10km): 57910.48


INTERSECTS (intersects-bounds 10km): 57874.81


INTERSECTS (intersects-bounds 10km): 56614.35


INTERSECTS (intersects-bounds 10km): 58344.49


INTERSECTS (intersects-bounds 10km): 59467.24


INTERSECTS (intersects-bounds 10km): 59774.44
====== INTERSECTS (intersects-bounds 10km) ======
  100000 requests completed in 1.67 seconds
  50 parallel clients
  115 bytes payload
  keep alive: 1

73.54% <= 0 milliseconds
95.31% <= 1 milliseconds
98.54% <= 2 milliseconds
99.26% <= 3 milliseconds
99.67% <= 4 milliseconds
99.80% <= 5 milliseconds
99.86% <= 6 milliseconds
99.90% <= 7 milliseconds
99.93% <= 8 milliseconds
99.93% <= 9 milliseconds
```

```
99.97% <= 10 milliseconds
99.98% <= 11 milliseconds
99.99% <= 12 milliseconds
100.00% <= 23 milliseconds
59774.44 requests per second


INTERSECTS (intersects-bounds 100km): 0.00

INTERSECTS (intersects-bounds 100km): 55345.37

INTERSECTS (intersects-bounds 100km): 60295.72

INTERSECTS (intersects-bounds 100km): 62168.69

INTERSECTS (intersects-bounds 100km): 63075.04

INTERSECTS (intersects-bounds 100km): 62092.87

INTERSECTS (intersects-bounds 100km): 58705.89

INTERSECTS (intersects-bounds 100km): 59787.30

INTERSECTS (intersects-bounds 100km): 60293.02

INTERSECTS (intersects-bounds 100km): 60363.65
====== INTERSECTS (intersects-bounds 100km) ======
  100000 requests completed in 1.66 seconds
  50 parallel clients
  119 bytes payload
  keep alive: 1

75.28% <= 0 milliseconds
95.88% <= 1 milliseconds
98.66% <= 2 milliseconds
99.16% <= 3 milliseconds
99.48% <= 4 milliseconds
99.71% <= 5 milliseconds
99.81% <= 6 milliseconds
99.83% <= 7 milliseconds
99.85% <= 9 milliseconds
99.96% <= 11 milliseconds
99.98% <= 12 milliseconds
100.00% <= 16 milliseconds
60363.65 requests per second


INTERSECTS (intersects-az limit 5): 0.00

INTERSECTS (intersects-az limit 5): 57447.66

INTERSECTS (intersects-az limit 5): 49590.44

INTERSECTS (intersects-az limit 5): 51813.89

INTERSECTS (intersects-az limit 5): 53089.59

INTERSECTS (intersects-az limit 5): 53839.54
INTERSECTS (intersects-az limit 5): 49998.18


INTERSECTS (intersects-az limit 5): 50216.36


INTERSECTS (intersects-az limit 5): 50947.22


INTERSECTS (intersects-az limit 5): 51240.16


INTERSECTS (intersects-az limit 5): 50856.18
====== INTERSECTS (intersects-az limit 5) ======
  100000 requests completed in 1.97 seconds
  50 parallel clients
  102 bytes payload
  keep alive: 1

62.85% <= 0 milliseconds
93.36% <= 1 milliseconds
97.43% <= 2 milliseconds
98.83% <= 3 milliseconds
99.35% <= 4 milliseconds
99.55% <= 5 milliseconds
99.73% <= 6 milliseconds
99.80% <= 7 milliseconds
99.84% <= 8 milliseconds
99.90% <= 9 milliseconds
99.94% <= 10 milliseconds
99.97% <= 11 milliseconds
99.97% <= 12 milliseconds
```

```
99.98% <= 13 milliseconds
99.99% <= 14 milliseconds
100.00% <= 19 milliseconds
50856.18 requests per second


WITHIN (within-circle 1km): 0.00

WITHIN (within-circle 1km): 47397.53

WITHIN (within-circle 1km): 40169.05

WITHIN (within-circle 1km): 39209.37

WITHIN (within-circle 1km): 39438.82

WITHIN (within-circle 1km): 38947.86

WITHIN (within-circle 1km): 38296.22

WITHIN (within-circle 1km): 38748.34

WITHIN (within-circle 1km): 38061.46

WITHIN (within-circle 1km): 38943.82

WITHIN (within-circle 1km): 38100.68

WITHIN (within-circle 1km): 38374.99
WITHIN (within-circle 1km): 38373.88

WITHIN (within-circle 1km): 38069.53

WITHIN (within-circle 1km): 38107.55
====== WITHIN (within-circle 1km) ======
  100000 requests completed in 2.62 seconds
  50 parallel clients
  93 bytes payload
  keep alive: 1

47.93% <= 0 milliseconds
82.62% <= 1 milliseconds
93.46% <= 2 milliseconds
97.15% <= 3 milliseconds
98.42% <= 4 milliseconds
98.99% <= 5 milliseconds
99.37% <= 6 milliseconds
99.58% <= 7 milliseconds
99.71% <= 8 milliseconds
99.81% <= 9 milliseconds
99.89% <= 10 milliseconds
99.92% <= 11 milliseconds
99.95% <= 12 milliseconds
99.96% <= 13 milliseconds
99.97% <= 14 milliseconds
99.98% <= 15 milliseconds
100.00% <= 20 milliseconds
```

```
100.00% <= 36 milliseconds
38107.55 requests per second


WITHIN (within-circle 10km): 0.00

WITHIN (within-circle 10km): 34301.48

WITHIN (within-circle 10km): 37800.82

WITHIN (within-circle 10km): 36087.03

WITHIN (within-circle 10km): 36472.81

WITHIN (within-circle 10km): 35217.94

WITHIN (within-circle 10km): 34897.69

WITHIN (within-circle 10km): 34452.26

WITHIN (within-circle 10km): 33821.99

WITHIN (within-circle 10km): 34299.56

WITHIN (within-circle 10km): 33725.96

WITHIN (within-circle 10km): 34013.32

WITHIN (within-circle 10km): 33613.37
WITHIN (within-circle 10km): 33792.45


WITHIN (within-circle 10km): 33451.90


WITHIN (within-circle 10km): 33574.00
====== WITHIN (within-circle 10km) ======
  100000 requests completed in 2.98 seconds
  50 parallel clients
  94 bytes payload
  keep alive: 1

44.47% <= 0 milliseconds
76.90% <= 1 milliseconds
90.62% <= 2 milliseconds
95.88% <= 3 milliseconds
97.59% <= 4 milliseconds
98.36% <= 5 milliseconds
98.87% <= 6 milliseconds
99.16% <= 7 milliseconds
99.41% <= 8 milliseconds
99.61% <= 9 milliseconds
99.71% <= 10 milliseconds
99.77% <= 11 milliseconds
99.84% <= 12 milliseconds
99.88% <= 13 milliseconds
99.89% <= 14 milliseconds
99.91% <= 15 milliseconds
99.93% <= 16 milliseconds
```

```
99.94% <= 17 milliseconds
99.95% <= 18 milliseconds
99.96% <= 19 milliseconds
99.99% <= 20 milliseconds
99.99% <= 21 milliseconds
100.00% <= 32 milliseconds
33574.00 requests per second


WITHIN (within-circle 100km): 0.00

WITHIN (within-circle 100km): 5054.43

WITHIN (within-circle 100km): 5209.42

WITHIN (within-circle 100km): 5459.24

WITHIN (within-circle 100km): 5405.17

WITHIN (within-circle 100km): 5508.51

WITHIN (within-circle 100km): 5450.05

WITHIN (within-circle 100km): 5557.92

WITHIN (within-circle 100km): 5629.23

WITHIN (within-circle 100km): 5413.84
WITHIN (within-circle 100km): 5192.99

WITHIN (within-circle 100km): 5132.60

WITHIN (within-circle 100km): 5091.97

WITHIN (within-circle 100km): 5147.74

WITHIN (within-circle 100km): 5169.75

WITHIN (within-circle 100km): 5269.04

WITHIN (within-circle 100km): 5235.59

WITHIN (within-circle 100km): 5181.82

WITHIN (within-circle 100km): 5133.31

WITHIN (within-circle 100km): 5199.20

WITHIN (within-circle 100km): 5215.82

WITHIN (within-circle 100km): 5273.95

WITHIN (within-circle 100km): 5325.43

WITHIN (within-circle 100km): 5377.26

WITHIN (within-circle 100km): 5426.89
WITHIN (within-circle 100km): 5389.04

WITHIN (within-circle 100km): 5367.99

WITHIN (within-circle 100km): 5293.40

WITHIN (within-circle 100km): 5271.44

WITHIN (within-circle 100km): 5291.82

WITHIN (within-circle 100km): 5326.08

WITHIN (within-circle 100km): 5328.35

WITHIN (within-circle 100km): 5288.33

WITHIN (within-circle 100km): 5256.24

WITHIN (within-circle 100km): 5267.43

WITHIN (within-circle 100km): 5283.53

WITHIN (within-circle 100km): 5300.84

WITHIN (within-circle 100km): 5328.11

WITHIN (within-circle 100km): 5321.60
WITHIN (within-circle 100km): 5336.40

WITHIN (within-circle 100km): 5328.09

WITHIN (within-circle 100km): 5368.85

WITHIN (within-circle 100km): 5390.04

WITHIN (within-circle 100km): 5399.24

WITHIN (within-circle 100km): 5385.22

WITHIN (within-circle 100km): 5369.29

WITHIN (within-circle 100km): 5385.30

WITHIN (within-circle 100km): 5380.87

WITHIN (within-circle 100km): 5367.11

WITHIN (within-circle 100km): 5327.98

WITHIN (within-circle 100km): 5298.27

WITHIN (within-circle 100km): 5293.65

WITHIN (within-circle 100km): 5303.86

WITHIN (within-circle 100km): 5313.60
WITHIN (within-circle 100km): 5318.93

WITHIN (within-circle 100km): 5298.74

WITHIN (within-circle 100km): 5264.71

WITHIN (within-circle 100km): 5255.30

WITHIN (within-circle 100km): 5254.06

WITHIN (within-circle 100km): 5271.33

WITHIN (within-circle 100km): 5268.66

WITHIN (within-circle 100km): 5286.32

WITHIN (within-circle 100km): 5296.02

WITHIN (within-circle 100km): 5269.62

WITHIN (within-circle 100km): 5251.56

WITHIN (within-circle 100km): 5249.34

WITHIN (within-circle 100km): 5252.15

WITHIN (within-circle 100km): 5255.51
WITHIN (within-circle 100km): 5244.73

WITHIN (within-circle 100km): 5252.78

WITHIN (within-circle 100km): 5246.65

WITHIN (within-circle 100km): 5261.78

WITHIN (within-circle 100km): 5262.33

WITHIN (within-circle 100km): 5271.02

WITHIN (within-circle 100km): 5284.75

WITHIN (within-circle 100km): 5277.66

WITHIN (within-circle 100km): 5266.53

WITHIN (within-circle 100km): 5262.86

WITHIN (within-circle 100km): 5266.99

WITHIN (within-circle 100km): 5275.34

WITHIN (within-circle 100km): 5287.15

WITHIN (within-circle 100km): 5285.69

WITHIN (within-circle 100km): 5299.68
WITHIN (within-circle 100km): 5300.88

WITHIN (within-circle 100km): 5309.13

WITHIN (within-circle 100km): 5318.09

WITHIN (within-circle 100km): 5317.41

WITHIN (within-circle 100km): 5289.13

WITHIN (within-circle 100km): 5267.64

WITHIN (within-circle 100km): 5243.55

WITHIN (within-circle 100km): 5239.81

WITHIN (within-circle 100km): 5231.03

WITHIN (within-circle 100km): 5206.36

WITHIN (within-circle 100km): 5191.40

WITHIN (within-circle 100km): 5172.53

WITHIN (within-circle 100km): 5143.71

WITHIN (within-circle 100km): 5118.67
WITHIN (within-circle 100km): 5106.65


WITHIN (within-circle 100km): 5080.87


WITHIN (within-circle 100km): 5058.13
====== WITHIN (within-circle 100km) ======
  100000 requests completed in 19.77 seconds
  50 parallel clients
  95 bytes payload
  keep alive: 1

29.90% <= 0 milliseconds
33.08% <= 1 milliseconds
35.41% <= 2 milliseconds
38.22% <= 3 milliseconds
41.64% <= 4 milliseconds
45.50% <= 5 milliseconds
49.66% <= 6 milliseconds
53.89% <= 7 milliseconds
57.89% <= 8 milliseconds
61.96% <= 9 milliseconds
65.73% <= 10 milliseconds
69.36% <= 11 milliseconds
72.71% <= 12 milliseconds
75.84% <= 13 milliseconds
78.79% <= 14 milliseconds
81.63% <= 15 milliseconds
84.19% <= 16 milliseconds
86.37% <= 17 milliseconds
```

```
88.15% <= 18 milliseconds
89.67% <= 19 milliseconds
90.90% <= 20 milliseconds
91.88% <= 21 milliseconds
92.74% <= 22 milliseconds
93.40% <= 23 milliseconds
93.97% <= 24 milliseconds
94.46% <= 25 milliseconds
94.91% <= 26 milliseconds
95.28% <= 27 milliseconds
95.68% <= 28 milliseconds
96.02% <= 29 milliseconds
96.32% <= 30 milliseconds
96.59% <= 31 milliseconds
96.84% <= 32 milliseconds
97.05% <= 33 milliseconds
97.25% <= 34 milliseconds
97.42% <= 35 milliseconds
97.58% <= 36 milliseconds
97.73% <= 37 milliseconds
97.84% <= 38 milliseconds
97.97% <= 39 milliseconds
98.06% <= 40 milliseconds
98.15% <= 41 milliseconds
98.24% <= 42 milliseconds
98.34% <= 43 milliseconds
98.42% <= 44 milliseconds
98.50% <= 45 milliseconds
98.57% <= 46 milliseconds
```

```
98.66% <= 47 milliseconds

98.73% <= 48 milliseconds

98.79% <= 49 milliseconds

98.86% <= 50 milliseconds

98.92% <= 51 milliseconds

98.98% <= 52 milliseconds

99.02% <= 53 milliseconds

99.06% <= 54 milliseconds

99.10% <= 55 milliseconds

99.14% <= 56 milliseconds

99.19% <= 57 milliseconds

99.23% <= 58 milliseconds

99.27% <= 59 milliseconds

99.30% <= 60 milliseconds

99.33% <= 61 milliseconds

99.35% <= 62 milliseconds

99.38% <= 63 milliseconds

99.41% <= 64 milliseconds

99.43% <= 65 milliseconds

99.45% <= 66 milliseconds

99.48% <= 67 milliseconds

99.50% <= 68 milliseconds

99.52% <= 69 milliseconds

99.56% <= 70 milliseconds

99.59% <= 71 milliseconds

99.62% <= 72 milliseconds

99.64% <= 73 milliseconds

99.65% <= 74 milliseconds

99.67% <= 75 milliseconds
```

```
99.69% <= 76 milliseconds

99.70% <= 77 milliseconds

99.72% <= 78 milliseconds

99.74% <= 79 milliseconds

99.75% <= 80 milliseconds

99.76% <= 81 milliseconds

99.77% <= 82 milliseconds

99.78% <= 83 milliseconds

99.79% <= 84 milliseconds

99.80% <= 85 milliseconds

99.81% <= 86 milliseconds

99.82% <= 88 milliseconds

99.83% <= 89 milliseconds

99.84% <= 90 milliseconds

99.85% <= 91 milliseconds

99.86% <= 92 milliseconds

99.87% <= 94 milliseconds

99.88% <= 95 milliseconds

99.89% <= 96 milliseconds

99.90% <= 97 milliseconds

99.91% <= 99 milliseconds

99.92% <= 102 milliseconds

99.93% <= 104 milliseconds

99.94% <= 109 milliseconds

99.95% <= 112 milliseconds

99.96% <= 116 milliseconds

99.97% <= 121 milliseconds

99.98% <= 129 milliseconds

99.99% <= 153 milliseconds
```

```
100.00% <= 190 milliseconds
5058.13 requests per second


WITHIN (within-bounds 1km): 0.00


WITHIN (within-bounds 1km): 37715.16


WITHIN (within-bounds 1km): 40494.85


WITHIN (within-bounds 1km): 40662.19


WITHIN (within-bounds 1km): 38074.19


WITHIN (within-bounds 1km): 37206.78


WITHIN (within-bounds 1km): 39173.96


WITHIN (within-bounds 1km): 39850.97


WITHIN (within-bounds 1km): 39044.38


WITHIN (within-bounds 1km): 38535.70


WITHIN (within-bounds 1km): 39532.98


WITHIN (within-bounds 1km): 39163.10


WITHIN (within-bounds 1km): 40316.85
WITHIN (within-bounds 1km): 40290.67
====== WITHIN (within-bounds 1km) ======
  100000 requests completed in 2.48 seconds
  50 parallel clients
  112 bytes payload
  keep alive: 1

56.17% <= 0 milliseconds
86.15% <= 1 milliseconds
93.30% <= 2 milliseconds
96.37% <= 3 milliseconds
98.01% <= 4 milliseconds
98.91% <= 5 milliseconds
99.25% <= 6 milliseconds
99.50% <= 7 milliseconds
99.67% <= 8 milliseconds
99.77% <= 9 milliseconds
99.88% <= 10 milliseconds
99.94% <= 11 milliseconds
99.95% <= 12 milliseconds
99.97% <= 13 milliseconds
99.98% <= 14 milliseconds
99.99% <= 15 milliseconds
99.99% <= 16 milliseconds
100.00% <= 35 milliseconds
40290.67 requests per second
```

```
WITHIN (within-bounds 10km): 0.00


WITHIN (within-bounds 10km): 41756.85


WITHIN (within-bounds 10km): 44512.23


WITHIN (within-bounds 10km): 49184.56


WITHIN (within-bounds 10km): 51092.26


WITHIN (within-bounds 10km): 52176.82


WITHIN (within-bounds 10km): 49600.49


WITHIN (within-bounds 10km): 46916.38


WITHIN (within-bounds 10km): 48056.74


WITHIN (within-bounds 10km): 47895.78


WITHIN (within-bounds 10km): 48735.47


WITHIN (within-bounds 10km): 48822.60
====== WITHIN (within-bounds 10km) ======
  100000 requests completed in 2.05 seconds
  50 parallel clients
  112 bytes payload
  keep alive: 1
```

```
64.77% <= 0 milliseconds
91.58% <= 1 milliseconds
96.45% <= 2 milliseconds
98.14% <= 3 milliseconds
98.90% <= 4 milliseconds
99.35% <= 5 milliseconds
99.60% <= 6 milliseconds
99.76% <= 7 milliseconds
99.81% <= 8 milliseconds
99.85% <= 9 milliseconds
99.91% <= 10 milliseconds
99.94% <= 11 milliseconds
99.95% <= 12 milliseconds
99.96% <= 13 milliseconds
99.96% <= 14 milliseconds
99.97% <= 15 milliseconds
99.99% <= 18 milliseconds
99.99% <= 21 milliseconds
100.00% <= 26 milliseconds
48822.60 requests per second


WITHIN (within-bounds 100km): 0.00


WITHIN (within-bounds 100km): 55096.61


WITHIN (within-bounds 100km): 48348.26


WITHIN (within-bounds 100km): 51320.50
WITHIN (within-bounds 100km): 52559.90


WITHIN (within-bounds 100km): 52811.25


WITHIN (within-bounds 100km): 52572.96


WITHIN (within-bounds 100km): 50642.49


WITHIN (within-bounds 100km): 49080.28


WITHIN (within-bounds 100km): 49915.38


WITHIN (within-bounds 100km): 50356.91
====== WITHIN (within-bounds 100km) ======
  100000 requests completed in 1.99 seconds
  50 parallel clients
  116 bytes payload
  keep alive: 1

65.13% <= 0 milliseconds
91.72% <= 1 milliseconds
96.82% <= 2 milliseconds
98.45% <= 3 milliseconds
99.34% <= 4 milliseconds
99.66% <= 5 milliseconds
99.89% <= 6 milliseconds
99.95% <= 7 milliseconds
99.97% <= 8 milliseconds
```

```
99.98% <= 9 milliseconds

99.98% <= 10 milliseconds

99.99% <= 13 milliseconds

100.00% <= 28 milliseconds

50356.91 requests per second



NEARBY (limit 1): 0.00


NEARBY (limit 1): 17345.04


NEARBY (limit 1): 17602.57


NEARBY (limit 1): 16098.60


NEARBY (limit 1): 16395.41


NEARBY (limit 1): 18075.32


NEARBY (limit 1): 17013.66


NEARBY (limit 1): 17660.49


NEARBY (limit 1): 18326.23


NEARBY (limit 1): 17554.09


NEARBY (limit 1): 17883.79
NEARBY (limit 1): 17220.22

NEARBY (limit 1): 17774.81

NEARBY (limit 1): 17977.73

NEARBY (limit 1): 17801.13

NEARBY (limit 1): 18223.58

NEARBY (limit 1): 17786.06

NEARBY (limit 1): 18166.11

NEARBY (limit 1): 18430.28

NEARBY (limit 1): 18151.06

NEARBY (limit 1): 18465.54

NEARBY (limit 1): 18608.30

NEARBY (limit 1): 18400.57

NEARBY (limit 1): 18693.21

NEARBY (limit 1): 18578.95

NEARBY (limit 1): 18632.18
NEARBY (limit 1): 18856.95

NEARBY (limit 1): 18571.51
====== NEARBY (limit 1) ======
  100000 requests completed in 5.38 seconds
  50 parallel clients
  100 bytes payload
  keep alive: 1

40.29% <= 0 milliseconds
64.54% <= 1 milliseconds
74.74% <= 2 milliseconds
80.22% <= 3 milliseconds
84.12% <= 4 milliseconds
87.44% <= 5 milliseconds
90.18% <= 6 milliseconds
92.28% <= 7 milliseconds
94.04% <= 8 milliseconds
95.36% <= 9 milliseconds
96.37% <= 10 milliseconds
97.22% <= 11 milliseconds
97.78% <= 12 milliseconds
98.25% <= 13 milliseconds
98.63% <= 14 milliseconds
98.92% <= 15 milliseconds
99.15% <= 16 milliseconds
99.31% <= 17 milliseconds
99.42% <= 18 milliseconds
```

```
99.50% <= 19 milliseconds

99.57% <= 20 milliseconds

99.64% <= 21 milliseconds

99.68% <= 22 milliseconds

99.72% <= 23 milliseconds

99.76% <= 24 milliseconds

99.80% <= 25 milliseconds

99.83% <= 26 milliseconds

99.85% <= 27 milliseconds

99.86% <= 28 milliseconds

99.88% <= 29 milliseconds

99.89% <= 30 milliseconds

99.91% <= 31 milliseconds

99.92% <= 32 milliseconds

99.93% <= 34 milliseconds

99.94% <= 35 milliseconds

99.95% <= 36 milliseconds

99.96% <= 37 milliseconds

99.97% <= 39 milliseconds

99.98% <= 42 milliseconds

99.99% <= 59 milliseconds

100.00% <= 75 milliseconds

18571.51 requests per second



NEARBY (limit 10): 0.00


NEARBY (limit 10): 15485.23
NEARBY (limit 10): 16053.68

NEARBY (limit 10): 12700.89

NEARBY (limit 10): 13276.06

NEARBY (limit 10): 13359.68

NEARBY (limit 10): 13157.88

NEARBY (limit 10): 13740.29

NEARBY (limit 10): 13583.69

NEARBY (limit 10): 13502.27

NEARBY (limit 10): 13851.89

NEARBY (limit 10): 13779.46

NEARBY (limit 10): 13631.02

NEARBY (limit 10): 13790.96

NEARBY (limit 10): 13143.70

NEARBY (limit 10): 12620.29

NEARBY (limit 10): 12607.92
NEARBY (limit 10): 12124.76

NEARBY (limit 10): 11998.37

NEARBY (limit 10): 11772.80

NEARBY (limit 10): 11393.44

NEARBY (limit 10): 11389.94

NEARBY (limit 10): 11171.81

NEARBY (limit 10): 10851.97

NEARBY (limit 10): 10977.50

NEARBY (limit 10): 10784.57

NEARBY (limit 10): 10974.73

NEARBY (limit 10): 11211.15

NEARBY (limit 10): 11144.17

NEARBY (limit 10): 11258.44

NEARBY (limit 10): 11251.98
NEARBY (limit 10): 11284.95

NEARBY (limit 10): 11458.79

NEARBY (limit 10): 11335.70

NEARBY (limit 10): 11405.75

NEARBY (limit 10): 11517.67

NEARBY (limit 10): 11476.46

NEARBY (limit 10): 11609.11

NEARBY (limit 10): 11714.01

NEARBY (limit 10): 11606.79

NEARBY (limit 10): 11739.74

NEARBY (limit 10): 11741.25

NEARBY (limit 10): 11775.74

NEARBY (limit 10): 11776.68
====== NEARBY (limit 10) ======
  100000 requests completed in 8.49 seconds
  50 parallel clients
  101 bytes payload
  keep alive: 1

32.84% <= 0 milliseconds
50.11% <= 1 milliseconds
63.18% <= 2 milliseconds
70.84% <= 3 milliseconds
75.65% <= 4 milliseconds
78.98% <= 5 milliseconds
81.86% <= 6 milliseconds
84.31% <= 7 milliseconds
86.36% <= 8 milliseconds
88.20% <= 9 milliseconds
89.82% <= 10 milliseconds
91.20% <= 11 milliseconds
92.43% <= 12 milliseconds
93.50% <= 13 milliseconds
94.46% <= 14 milliseconds
95.26% <= 15 milliseconds
95.98% <= 16 milliseconds
96.51% <= 17 milliseconds
96.94% <= 18 milliseconds
97.31% <= 19 milliseconds
97.63% <= 20 milliseconds
97.90% <= 21 milliseconds
98.14% <= 22 milliseconds
98.34% <= 23 milliseconds
98.52% <= 24 milliseconds
98.68% <= 25 milliseconds
98.81% <= 26 milliseconds
```

```
98.94% <= 27 milliseconds
99.04% <= 28 milliseconds
99.11% <= 29 milliseconds
99.19% <= 30 milliseconds
99.27% <= 31 milliseconds
99.33% <= 32 milliseconds
99.40% <= 33 milliseconds
99.45% <= 34 milliseconds
99.49% <= 35 milliseconds
99.54% <= 36 milliseconds
99.58% <= 37 milliseconds
99.63% <= 38 milliseconds
99.66% <= 39 milliseconds
99.69% <= 40 milliseconds
99.72% <= 41 milliseconds
99.74% <= 42 milliseconds
99.77% <= 43 milliseconds
99.78% <= 44 milliseconds
99.79% <= 45 milliseconds
99.81% <= 46 milliseconds
99.82% <= 47 milliseconds
99.84% <= 48 milliseconds
99.85% <= 49 milliseconds
99.86% <= 50 milliseconds
99.88% <= 51 milliseconds
99.88% <= 52 milliseconds
99.89% <= 53 milliseconds
99.90% <= 54 milliseconds
99.92% <= 56 milliseconds
```

```
99.92% <= 57 milliseconds

99.94% <= 59 milliseconds

99.94% <= 60 milliseconds

99.95% <= 61 milliseconds

99.96% <= 64 milliseconds

99.97% <= 71 milliseconds

99.98% <= 79 milliseconds

99.99% <= 85 milliseconds

100.00% <= 123 milliseconds

11776.68 requests per second



NEARBY (limit 100): 0.00


NEARBY (limit 100): 5735.29


NEARBY (limit 100): 6042.82


NEARBY (limit 100): 6171.93


NEARBY (limit 100): 5342.22


NEARBY (limit 100): 5507.05


NEARBY (limit 100): 5654.64


NEARBY (limit 100): 5340.18


NEARBY (limit 100): 5363.02
NEARBY (limit 100): 5479.74

NEARBY (limit 100): 5355.47

NEARBY (limit 100): 5292.58

NEARBY (limit 100): 5381.11

NEARBY (limit 100): 5453.28

NEARBY (limit 100): 5252.11

NEARBY (limit 100): 5328.91

NEARBY (limit 100): 5399.93

NEARBY (limit 100): 5290.49

NEARBY (limit 100): 5298.50

NEARBY (limit 100): 5305.05

NEARBY (limit 100): 5185.44

NEARBY (limit 100): 5238.50

NEARBY (limit 100): 5272.09
NEARBY (limit 100): 5118.20

NEARBY (limit 100): 5146.75

NEARBY (limit 100): 5076.51

NEARBY (limit 100): 5056.47

NEARBY (limit 100): 5104.44

NEARBY (limit 100): 5118.03

NEARBY (limit 100): 5050.23

NEARBY (limit 100): 5087.95

NEARBY (limit 100): 5100.87

NEARBY (limit 100): 5050.59

NEARBY (limit 100): 5070.85

NEARBY (limit 100): 5017.68

NEARBY (limit 100): 5013.90

NEARBY (limit 100): 5042.42

NEARBY (limit 100): 4964.82
NEARBY (limit 100): 4991.50

NEARBY (limit 100): 5026.11

NEARBY (limit 100): 4952.02

NEARBY (limit 100): 4979.57

NEARBY (limit 100): 5006.97

NEARBY (limit 100): 4986.24

NEARBY (limit 100): 4980.05

NEARBY (limit 100): 5007.76

NEARBY (limit 100): 5031.37

NEARBY (limit 100): 4975.09

NEARBY (limit 100): 4991.80

NEARBY (limit 100): 4992.47

NEARBY (limit 100): 4967.00

NEARBY (limit 100): 4985.28
NEARBY (limit 100): 4971.99

NEARBY (limit 100): 4966.83

NEARBY (limit 100): 4978.10

NEARBY (limit 100): 4922.33

NEARBY (limit 100): 4930.11

NEARBY (limit 100): 4862.24

NEARBY (limit 100): 4834.97

NEARBY (limit 100): 4799.81

NEARBY (limit 100): 4748.95

NEARBY (limit 100): 4757.70

NEARBY (limit 100): 4698.46

NEARBY (limit 100): 4669.03

NEARBY (limit 100): 4651.35

NEARBY (limit 100): 4599.85

NEARBY (limit 100): 4590.22
NEARBY (limit 100): 4569.63

NEARBY (limit 100): 4595.83

NEARBY (limit 100): 4618.96

NEARBY (limit 100): 4593.34

NEARBY (limit 100): 4613.94

NEARBY (limit 100): 4637.48

NEARBY (limit 100): 4617.07

NEARBY (limit 100): 4635.20

NEARBY (limit 100): 4658.61

NEARBY (limit 100): 4656.80

NEARBY (limit 100): 4645.91

NEARBY (limit 100): 4656.31

NEARBY (limit 100): 4619.07

NEARBY (limit 100): 4635.17
NEARBY (limit 100): 4643.94

NEARBY (limit 100): 4627.53

NEARBY (limit 100): 4644.21

NEARBY (limit 100): 4667.19

NEARBY (limit 100): 4673.37

NEARBY (limit 100): 4660.77

NEARBY (limit 100): 4678.90

NEARBY (limit 100): 4695.12

NEARBY (limit 100): 4692.05

NEARBY (limit 100): 4690.66

NEARBY (limit 100): 4707.75

NEARBY (limit 100): 4723.90

NEARBY (limit 100): 4707.84

NEARBY (limit 100): 4720.60

NEARBY (limit 100): 4739.97
NEARBY (limit 100): 4737.54


NEARBY (limit 100): 4731.90


NEARBY (limit 100): 4739.92


NEARBY (limit 100): 4713.48


NEARBY (limit 100): 4726.16


NEARBY (limit 100): 4732.31


NEARBY (limit 100): 4717.41


NEARBY (limit 100): 4724.50


NEARBY (limit 100): 4700.83


NEARBY (limit 100): 4713.31


NEARBY (limit 100): 4705.22


NEARBY (limit 100): 4701.35
====== NEARBY (limit 100) ======
  100000 requests completed in 21.27 seconds
  50 parallel clients
  102 bytes payload
  keep alive: 1
```

```
22.06% <= 0 milliseconds
31.45% <= 1 milliseconds
37.99% <= 2 milliseconds
44.64% <= 3 milliseconds
50.43% <= 4 milliseconds
55.61% <= 5 milliseconds
60.49% <= 6 milliseconds
64.61% <= 7 milliseconds
67.99% <= 8 milliseconds
70.53% <= 9 milliseconds
72.49% <= 10 milliseconds
74.00% <= 11 milliseconds
75.34% <= 12 milliseconds
76.51% <= 13 milliseconds
77.52% <= 14 milliseconds
78.56% <= 15 milliseconds
79.53% <= 16 milliseconds
80.45% <= 17 milliseconds
81.39% <= 18 milliseconds
82.30% <= 19 milliseconds
83.18% <= 20 milliseconds
84.04% <= 21 milliseconds
84.92% <= 22 milliseconds
85.81% <= 23 milliseconds
86.72% <= 24 milliseconds
87.59% <= 25 milliseconds
88.45% <= 26 milliseconds
89.23% <= 27 milliseconds
```

```
90.03% <= 28 milliseconds
90.77% <= 29 milliseconds
91.45% <= 30 milliseconds
92.17% <= 31 milliseconds
92.79% <= 32 milliseconds
93.36% <= 33 milliseconds
93.89% <= 34 milliseconds
94.36% <= 35 milliseconds
94.76% <= 36 milliseconds
95.13% <= 37 milliseconds
95.47% <= 38 milliseconds
95.75% <= 39 milliseconds
95.99% <= 40 milliseconds
96.23% <= 41 milliseconds
96.47% <= 42 milliseconds
96.66% <= 43 milliseconds
96.82% <= 44 milliseconds
97.00% <= 45 milliseconds
97.16% <= 46 milliseconds
97.31% <= 47 milliseconds
97.44% <= 48 milliseconds
97.57% <= 49 milliseconds
97.69% <= 50 milliseconds
97.81% <= 51 milliseconds
97.93% <= 52 milliseconds
98.01% <= 53 milliseconds
98.10% <= 54 milliseconds
98.17% <= 55 milliseconds
98.26% <= 56 milliseconds
```

```
98.33% <= 57 milliseconds
98.40% <= 58 milliseconds
98.48% <= 59 milliseconds
98.55% <= 60 milliseconds
98.62% <= 61 milliseconds
98.70% <= 62 milliseconds
98.77% <= 63 milliseconds
98.83% <= 64 milliseconds
98.88% <= 65 milliseconds
98.94% <= 66 milliseconds
98.99% <= 67 milliseconds
99.05% <= 68 milliseconds
99.09% <= 69 milliseconds
99.13% <= 70 milliseconds
99.17% <= 71 milliseconds
99.20% <= 72 milliseconds
99.23% <= 73 milliseconds
99.27% <= 74 milliseconds
99.30% <= 75 milliseconds
99.33% <= 76 milliseconds
99.36% <= 77 milliseconds
99.39% <= 78 milliseconds
99.42% <= 79 milliseconds
99.45% <= 80 milliseconds
99.47% <= 81 milliseconds
99.50% <= 82 milliseconds
99.52% <= 83 milliseconds
99.55% <= 84 milliseconds
99.56% <= 85 milliseconds
```

```
99.58% <= 86 milliseconds

99.60% <= 87 milliseconds

99.61% <= 88 milliseconds

99.63% <= 89 milliseconds

99.64% <= 90 milliseconds

99.65% <= 91 milliseconds

99.67% <= 92 milliseconds

99.68% <= 93 milliseconds

99.69% <= 94 milliseconds

99.70% <= 95 milliseconds

99.71% <= 96 milliseconds

99.72% <= 97 milliseconds

99.73% <= 98 milliseconds

99.75% <= 99 milliseconds

99.76% <= 100 milliseconds

99.77% <= 101 milliseconds

99.78% <= 102 milliseconds

99.79% <= 104 milliseconds

99.80% <= 105 milliseconds

99.82% <= 106 milliseconds

99.82% <= 107 milliseconds

99.84% <= 109 milliseconds

99.85% <= 111 milliseconds

99.86% <= 113 milliseconds

99.87% <= 115 milliseconds

99.88% <= 118 milliseconds

99.89% <= 120 milliseconds

99.90% <= 122 milliseconds

99.91% <= 124 milliseconds
```

```
99.92% <= 127 milliseconds
99.93% <= 128 milliseconds
99.94% <= 131 milliseconds
99.95% <= 137 milliseconds
99.96% <= 140 milliseconds
99.97% <= 144 milliseconds
99.98% <= 152 milliseconds
99.99% <= 162 milliseconds
100.00% <= 191 milliseconds
4701.35 requests per second


NEARBY (point 1km): 0.00

NEARBY (point 1km): 27780.57

NEARBY (point 1km): 20947.12

NEARBY (point 1km): 19797.13

NEARBY (point 1km): 22146.81

NEARBY (point 1km): 25717.52

NEARBY (point 1km): 25451.20

NEARBY (point 1km): 26321.88

NEARBY (point 1km): 28638.12
NEARBY (point 1km): 28745.79


NEARBY (point 1km): 28684.63


NEARBY (point 1km): 30376.14


NEARBY (point 1km): 30581.53


NEARBY (point 1km): 28972.66


NEARBY (point 1km): 27888.09


NEARBY (point 1km): 27660.20


NEARBY (point 1km): 28747.00


NEARBY (point 1km): 28750.52


NEARBY (point 1km): 28516.40
====== NEARBY (point 1km) ======
  100000 requests completed in 3.51 seconds
  50 parallel clients
  92 bytes payload
  keep alive: 1

47.54% <= 0 milliseconds
75.70% <= 1 milliseconds
85.52% <= 2 milliseconds
```

```
90.85% <= 3 milliseconds
93.95% <= 4 milliseconds
95.68% <= 5 milliseconds
96.79% <= 6 milliseconds
97.65% <= 7 milliseconds
98.21% <= 8 milliseconds
98.61% <= 9 milliseconds
98.94% <= 10 milliseconds
99.17% <= 11 milliseconds
99.35% <= 12 milliseconds
99.46% <= 13 milliseconds
99.57% <= 14 milliseconds
99.71% <= 15 milliseconds
99.79% <= 16 milliseconds
99.85% <= 17 milliseconds
99.88% <= 18 milliseconds
99.92% <= 19 milliseconds
99.93% <= 20 milliseconds
99.95% <= 21 milliseconds
99.96% <= 24 milliseconds
99.97% <= 26 milliseconds
99.99% <= 28 milliseconds
99.99% <= 29 milliseconds
100.00% <= 51 milliseconds
28516.40 requests per second


NEARBY (point 10km): 0.00
NEARBY (point 10km): 40085.27

NEARBY (point 10km): 32403.98

NEARBY (point 10km): 31765.02

NEARBY (point 10km): 29598.88

NEARBY (point 10km): 28922.64

NEARBY (point 10km): 26406.27

NEARBY (point 10km): 26992.30

NEARBY (point 10km): 26223.16

NEARBY (point 10km): 25934.92

NEARBY (point 10km): 26546.15

NEARBY (point 10km): 25869.33

NEARBY (point 10km): 26792.32

NEARBY (point 10km): 25815.89

NEARBY (point 10km): 25755.93

NEARBY (point 10km): 26126.21
NEARBY (point 10km): 25778.13


NEARBY (point 10km): 26248.08


NEARBY (point 10km): 25486.28


NEARBY (point 10km): 26106.77


NEARBY (point 10km): 26140.04
====== NEARBY (point 10km) ======
  100000 requests completed in 3.83 seconds
  50 parallel clients
  93 bytes payload
  keep alive: 1

45.22% <= 0 milliseconds
73.34% <= 1 milliseconds
82.79% <= 2 milliseconds
88.12% <= 3 milliseconds
91.53% <= 4 milliseconds
93.93% <= 5 milliseconds
95.62% <= 6 milliseconds
96.89% <= 7 milliseconds
97.68% <= 8 milliseconds
98.25% <= 9 milliseconds
98.64% <= 10 milliseconds
98.96% <= 11 milliseconds
99.22% <= 12 milliseconds
```

```
99.40% <= 13 milliseconds
99.52% <= 14 milliseconds
99.66% <= 15 milliseconds
99.74% <= 16 milliseconds
99.79% <= 17 milliseconds
99.82% <= 18 milliseconds
99.86% <= 19 milliseconds
99.89% <= 20 milliseconds
99.91% <= 21 milliseconds
99.92% <= 22 milliseconds
99.94% <= 23 milliseconds
99.96% <= 25 milliseconds
99.97% <= 26 milliseconds
99.98% <= 30 milliseconds
99.99% <= 32 milliseconds
100.00% <= 37 milliseconds
26140.04 requests per second


NEARBY (point 100km): 0.00


NEARBY (point 100km): 14476.87


NEARBY (point 100km): 17341.03


NEARBY (point 100km): 19672.58


NEARBY (point 100km): 17845.57
NEARBY (point 100km): 18505.78

NEARBY (point 100km): 19743.95

NEARBY (point 100km): 18466.47

NEARBY (point 100km): 18875.01

NEARBY (point 100km): 18469.26

NEARBY (point 100km): 18461.59

NEARBY (point 100km): 18794.42

NEARBY (point 100km): 17922.53

NEARBY (point 100km): 18093.67

NEARBY (point 100km): 17853.43

NEARBY (point 100km): 18258.92

NEARBY (point 100km): 18112.63

NEARBY (point 100km): 18200.47

NEARBY (point 100km): 18348.49

NEARBY (point 100km): 17948.26
NEARBY (point 100km): 18141.70


NEARBY (point 100km): 17854.56


NEARBY (point 100km): 18157.78


NEARBY (point 100km): 18147.41


NEARBY (point 100km): 18118.17


NEARBY (point 100km): 18389.75


NEARBY (point 100km): 18136.75


NEARBY (point 100km): 18273.80


NEARBY (point 100km): 18236.52
====== NEARBY (point 100km) ======
  100000 requests completed in 5.48 seconds
  50 parallel clients
  94 bytes payload
  keep alive: 1

38.71% <= 0 milliseconds
62.14% <= 1 milliseconds
73.61% <= 2 milliseconds
79.75% <= 3 milliseconds
84.05% <= 4 milliseconds
```

```
87.39% <= 5 milliseconds
89.99% <= 6 milliseconds
92.05% <= 7 milliseconds
93.80% <= 8 milliseconds
95.19% <= 9 milliseconds
96.24% <= 10 milliseconds
97.10% <= 11 milliseconds
97.77% <= 12 milliseconds
98.31% <= 13 milliseconds
98.65% <= 14 milliseconds
98.90% <= 15 milliseconds
99.11% <= 16 milliseconds
99.26% <= 17 milliseconds
99.39% <= 18 milliseconds
99.50% <= 19 milliseconds
99.57% <= 20 milliseconds
99.64% <= 21 milliseconds
99.70% <= 22 milliseconds
99.76% <= 23 milliseconds
99.79% <= 24 milliseconds
99.83% <= 25 milliseconds
99.86% <= 26 milliseconds
99.89% <= 27 milliseconds
99.90% <= 28 milliseconds
99.92% <= 30 milliseconds
99.92% <= 32 milliseconds
99.93% <= 33 milliseconds
99.94% <= 34 milliseconds
99.95% <= 35 milliseconds
```

```
99.96% <= 39 milliseconds
99.97% <= 42 milliseconds
99.98% <= 43 milliseconds
99.99% <= 46 milliseconds
100.00% <= 70 milliseconds
18236.52 requests per second

Scripts to run:
GET SCRIPT: return tile38.call('GET', KEYS[1], ARGV[1], 'point')
GET FOUR SCRIPT: local a = tile38.call('GET', KEYS[1], ARGV[1], 'point');local b = ti
SET SCRIPT: return tile38.call('SET', KEYS[1], ARGV[1], 'point', ARGV[2], ARGV[3])

EVAL (set point): 0.00

EVAL (set point): 20622.39

EVAL (set point): 20269.78

EVAL (set point): 18042.89

EVAL (set point): 18251.10

EVAL (set point): 18460.00

EVAL (set point): 17497.85

EVAL (set point): 17500.00

EVAL (set point): 17827.01
EVAL (set point): 17762.15

EVAL (set point): 17386.05

EVAL (set point): 17366.42

EVAL (set point): 17268.36

EVAL (set point): 17162.90

EVAL (set point): 17116.40

EVAL (set point): 17234.68

EVAL (set point): 17361.81

EVAL (set point): 17121.16

EVAL (set point): 17291.41

EVAL (set point): 17420.54

EVAL (set point): 17465.19

EVAL (set point): 17207.42

EVAL (set point): 17356.07
EVAL (set point): 17465.83


EVAL (set point): 17387.11


EVAL (set point): 17491.31


EVAL (set point): 17607.71


EVAL (set point): 17668.91


EVAL (set point): 17504.13


EVAL (set point): 17528.78
====== EVAL (set point) ======
  100000 requests completed in 5.70 seconds
  50 parallel clients
  156 bytes payload
  keep alive: 1

14.15% <= 0 milliseconds
23.47% <= 1 milliseconds
68.02% <= 2 milliseconds
85.52% <= 3 milliseconds
92.24% <= 4 milliseconds
95.44% <= 5 milliseconds
97.23% <= 6 milliseconds
98.01% <= 7 milliseconds
98.60% <= 8 milliseconds
98.86% <= 9 milliseconds
```

```
99.18% <= 10 milliseconds

99.27% <= 11 milliseconds

99.37% <= 12 milliseconds

99.49% <= 13 milliseconds

99.59% <= 14 milliseconds

99.65% <= 15 milliseconds

99.67% <= 16 milliseconds

99.71% <= 17 milliseconds

99.76% <= 18 milliseconds

99.83% <= 19 milliseconds

99.83% <= 20 milliseconds

99.84% <= 22 milliseconds

99.87% <= 23 milliseconds

99.90% <= 24 milliseconds

99.92% <= 25 milliseconds

99.93% <= 26 milliseconds

99.94% <= 27 milliseconds

99.94% <= 28 milliseconds

99.95% <= 30 milliseconds

99.96% <= 34 milliseconds

99.97% <= 39 milliseconds

99.99% <= 40 milliseconds

100.00% <= 49 milliseconds

17528.78 requests per second



EVALNA (set point): 0.00


EVALNA (set point): 24348.23
EVALNA (set point): 20161.42

EVALNA (set point): 21243.14

EVALNA (set point): 21731.35

EVALNA (set point): 20206.46

EVALNA (set point): 19806.49

EVALNA (set point): 20432.03

EVALNA (set point): 20358.60

EVALNA (set point): 19825.54

EVALNA (set point): 20207.45

EVALNA (set point): 20451.87

EVALNA (set point): 20043.84

EVALNA (set point): 20198.17

EVALNA (set point): 20364.58

EVALNA (set point): 20491.18
EVALNA (set point): 20236.82


EVALNA (set point): 20325.39


EVALNA (set point): 20499.95


EVALNA (set point): 20183.71


EVALNA (set point): 20263.32


EVALNA (set point): 20382.29


EVALNA (set point): 20455.01


EVALNA (set point): 20229.71


EVALNA (set point): 20333.37


EVALNA (set point): 20406.52
====== EVALNA (set point) ======
  100000 requests completed in 4.90 seconds
  50 parallel clients
  159 bytes payload
  keep alive: 1

24.97% <= 0 milliseconds
41.58% <= 1 milliseconds
73.23% <= 2 milliseconds
87.06% <= 3 milliseconds
```

```
92.61% <= 4 milliseconds
95.12% <= 5 milliseconds
96.71% <= 6 milliseconds
97.71% <= 7 milliseconds
98.42% <= 8 milliseconds
98.88% <= 9 milliseconds
99.09% <= 10 milliseconds
99.28% <= 11 milliseconds
99.44% <= 12 milliseconds
99.53% <= 13 milliseconds
99.59% <= 14 milliseconds
99.63% <= 15 milliseconds
99.66% <= 16 milliseconds
99.69% <= 17 milliseconds
99.70% <= 18 milliseconds
99.74% <= 19 milliseconds
99.77% <= 20 milliseconds
99.79% <= 21 milliseconds
99.82% <= 22 milliseconds
99.85% <= 23 milliseconds
99.85% <= 24 milliseconds
99.87% <= 25 milliseconds
99.90% <= 26 milliseconds
99.90% <= 27 milliseconds
99.92% <= 28 milliseconds
99.94% <= 29 milliseconds
99.96% <= 30 milliseconds
99.97% <= 31 milliseconds
99.97% <= 32 milliseconds
```

```
99.98% <= 34 milliseconds
99.99% <= 35 milliseconds
100.00% <= 38 milliseconds
20406.52 requests per second


EVALRO (get point): 0.00

EVALRO (get point): 27697.11

EVALRO (get point): 36312.69

EVALRO (get point): 34549.03

EVALRO (get point): 33900.18

EVALRO (get point): 35008.26

EVALRO (get point): 32652.77

EVALRO (get point): 33493.52

EVALRO (get point): 33928.91

EVALRO (get point): 32810.85

EVALRO (get point): 33418.72

EVALRO (get point): 33646.06
EVALRO (get point): 32723.21


EVALRO (get point): 33309.00


EVALRO (get point): 33635.78


EVALRO (get point): 32843.44


EVALRO (get point): 32803.14
====== EVALRO (get point) ======
  100000 requests completed in 3.05 seconds
  50 parallel clients
  112 bytes payload
  keep alive: 1

46.69% <= 0 milliseconds
78.93% <= 1 milliseconds
88.82% <= 2 milliseconds
93.40% <= 3 milliseconds
96.06% <= 4 milliseconds
97.51% <= 5 milliseconds
98.40% <= 6 milliseconds
98.95% <= 7 milliseconds
99.26% <= 8 milliseconds
99.50% <= 9 milliseconds
99.63% <= 10 milliseconds
99.72% <= 11 milliseconds
99.80% <= 12 milliseconds
```

```
99.84% <= 13 milliseconds
99.90% <= 14 milliseconds
99.92% <= 15 milliseconds
99.94% <= 16 milliseconds
99.95% <= 17 milliseconds
99.96% <= 18 milliseconds
99.97% <= 19 milliseconds
99.98% <= 20 milliseconds
99.99% <= 24 milliseconds
100.00% <= 31 milliseconds
32803.14 requests per second


EVALRO (get 4 points): 0.00

EVALRO (get 4 points): 43153.10

EVALRO (get 4 points): 36244.48

EVALRO (get 4 points): 32513.28

EVALRO (get 4 points): 33561.42

EVALRO (get 4 points): 33472.47

EVALRO (get 4 points): 30991.88

EVALRO (get 4 points): 31847.58
EVALRO (get 4 points): 32477.00

EVALRO (get 4 points): 31099.00

EVALRO (get 4 points): 31156.20

EVALRO (get 4 points): 31947.86

EVALRO (get 4 points): 30947.45

EVALRO (get 4 points): 31227.98

EVALRO (get 4 points): 31815.85

EVALRO (get 4 points): 31015.76

EVALRO (get 4 points): 31229.69
====== EVALRO (get 4 points) ======
  100000 requests completed in 3.20 seconds
  50 parallel clients
  338 bytes payload
  keep alive: 1

44.69% <= 0 milliseconds
75.51% <= 1 milliseconds
87.23% <= 2 milliseconds
92.75% <= 3 milliseconds
95.75% <= 4 milliseconds
97.48% <= 5 milliseconds
```

```
98.50% <= 6 milliseconds
99.09% <= 7 milliseconds
99.42% <= 8 milliseconds
99.62% <= 9 milliseconds
99.76% <= 10 milliseconds
99.82% <= 11 milliseconds
99.87% <= 12 milliseconds
99.92% <= 13 milliseconds
99.95% <= 14 milliseconds
99.96% <= 15 milliseconds
99.97% <= 17 milliseconds
99.98% <= 22 milliseconds
99.99% <= 24 milliseconds
100.00% <= 89 milliseconds
31229.69 requests per second


EVALNA (get point): 0.00


EVALNA (get point): 46920.46


EVALNA (get point): 35803.49


EVALNA (get point): 39153.75


EVALNA (get point): 39503.49


EVALNA (get point): 34135.96
EVALNA (get point): 31233.01


EVALNA (get point): 30994.45


EVALNA (get point): 32712.32


EVALNA (get point): 31371.54


EVALNA (get point): 31727.37


EVALNA (get point): 31946.23


EVALNA (get point): 31672.87


EVALNA (get point): 30631.03


EVALNA (get point): 31353.06


EVALNA (get point): 32263.34


EVALNA (get point): 32085.64
====== EVALNA (get point) ======
  100000 requests completed in 3.12 seconds
  50 parallel clients
  112 bytes payload
  keep alive: 1

50.18% <= 0 milliseconds
79.17% <= 1 milliseconds
```

```
88.48% <= 2 milliseconds
92.84% <= 3 milliseconds
95.19% <= 4 milliseconds
96.48% <= 5 milliseconds
97.39% <= 6 milliseconds
98.16% <= 7 milliseconds
98.64% <= 8 milliseconds
99.05% <= 9 milliseconds
99.31% <= 10 milliseconds
99.52% <= 11 milliseconds
99.66% <= 12 milliseconds
99.73% <= 13 milliseconds
99.78% <= 14 milliseconds
99.83% <= 15 milliseconds
99.87% <= 16 milliseconds
99.91% <= 17 milliseconds
99.92% <= 18 milliseconds
99.94% <= 19 milliseconds
99.95% <= 20 milliseconds
99.96% <= 22 milliseconds
99.97% <= 24 milliseconds
99.98% <= 28 milliseconds
99.99% <= 30 milliseconds
100.00% <= 34 milliseconds
32085.64 requests per second
```