JAMES COOK UNIVERSITY

SCHOOL OF ENGINEERING &
PHYSICAL SCIENCES

EG4012

WebGIS Visualisation Techniques: Comparison and Optimisation for Large Data Sets

Robert Pyke

Thesis submitted to the School of Engineering & Physical Sciences in partial fulfilment of the requirements for the degree of

Bachelor of Engineering and Information Technology with Honours

(Computer Systems)

October 2013

Statement of Access

I, the undersigned, author of this work, understand that James Cook University may make this thesis available for use within the University Library and, via the Australian Digital Theses network, for use elsewhere.

I understand that, as an unpublished work, a thesis has significant protection under the Copyright Act and I do not wish to place any further restriction on access to this work.

Signature

30/10/2013

Date

Declaration of Sources

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Signature

30/10/2013

Date

Abstract

Through this report, we highlight an existing need for high performance visualisation techniques with the capability of responding to a GeoJSON or Well Known Test (WKT) request for a 1,000,000 point data set within 1 second. We demonstrate that existing geospatial visualisation techniques are unable to process and transmit GeoJSON or WKT requests of this size, in the desired time frame.

We demonstrate that through the use of a common geospatial visualisation technique abstraction, we are able to create a test platform, capable of testing a visualisation technique's pre-processing, database usage, request processing, and transmission requirements. This common abstraction is abstract enough as to allow visualisation techniques with vastly different requirements to be directly compared.

Five techniques were designed and implemented to allow for the testing of the benefits of using bounding box, gridded clustering and caching strategies to process geospatial visualisation requests. Testing demonstrated that using the strategies in isolation did improve performance, but performance improvements capable of reaching our design goal were only possible through coupling the strategies together.

Results showed that, when not using any strategies, a GeoJSON request for a dataset of up to 22,000 points could be processed within a second. When using a bounding box strategy in isolation, a GeoJSON request with a defined bounding box containing up to 5,200 points could be processed within a second, regardless of the number of points in the entire data set. When using the gridded clustering and bounding strategies together, a request with up to 100,000 points within the defined bounding box could be processed within a second. When using a bounding box to select from a cache table of pre-processed gridded clusters, a GeoJSON request could be processed with up to 1,000,000 points. Significant draw-backs of using a cache table were identified: pre-processing took up to 160 minutes, and the database grew up to 567% in size.

We proposed that for most data sets with up to 1,000,000 data points, using a combination of the bounding box and gridded clustering strategies would result in significantly improved performance, and reduced transmission times. We also proposed that for data sets that change infrequently, and where sufficient database storage was available, a cache table could be introduced to further improve performance.

Acknowledgments

I would like to thank Dr Owen Kenny, my supervisor, for providing direction and feedback; Dr Jeremy VanDerWal, for introducing me to Geospatial Information Systems; and Laura Pyke, for her enduring support.

Table of Contents

STATEMENT OF ACCESS ..... **II**

DECLARATION OF SOURCES ..... **III**

ABSTRACT ..... **IV**

ACKNOWLEDGMENTS ..... **V**

INTRODUCTION **1**

1.1 PROBLEM DEFINITION 1

1.2 PROJECT AIMS 1

1.3 PROJECT OUTCOMES 2

2 LITERATURE REVIEW **3**

2.1 WEB GIS 3

2.1.1 GIS Database *3*

2.1.2 Web Map Server *4*

2.1.3 Web Server *6*

2.1.4 Web GIS Client *7*

2.1.5 Web GIS Performance Optimisations *8*

2.1.6 Bounding Box *8*

2.1.7 Clustering *9*

2.1.8 Caching *9*

2.2 CONCLUSION 9

3 METHODOLOGY **10**

3.1 OVERVIEW 10

3.1.1 Proposed Method of Comparison *10*

3.1.2 Proposed High Performance Visualisation Technique *11*

3.2 ABSTRACTION OF GEOSPATIAL VISUALISATION TECHNIQUES 13

3.2.1 Extensible and Pluggable *13*

3.2.2 Analysis of Performance *13*

3.2.3 Abstraction Description *13*

3.2.4 Abstraction Implementation *14*

3.3 IMPLEMENTED GEOSPATIAL VISUALISATION TECHNIQUES 16

3.3.1 Mappable Point *16*

3.3.2 Bound Mappable Point *18*

3.3.3 Gridded Mappable Point *20*

3.3.4 Gridded and Bound Mappable Point 24
3.3.5 Cached, Gridded and Bound Mappable Point 27
3.4 TEST SYSTEM 33
3.4.1 Web GIS Foundation 33
3.4.2 Reproducible and Consistent Testing Environment 33
3.4.3 Input Data Sets 33
3.4.4 Test Scenario 36
4 SUMMARISED RESULTS 40
4.1 WORLDWIDE 40
4.1.1 Pre-Processing 40
4.1.2 Clusters 42
4.1.3 GeoJSON 43
4.1.4 WKT 47
4.2 AUSTRALIA WIDE 49
4.2.1 Pre-Processing 49
4.2.2 Clusters 51
4.2.3 GeoJSON 52
4.2.4 WKT 55
4.3 BRISBANE WIDE 58
4.3.1 Pre-Processing 58
4.3.2 Clusters 60
4.3.3 GeoJSON 61
4.3.4 WKT 64
5 DISCUSSION 68
5.1 MAPPABLE POINT 68
5.1.1 Processing Requirements 68
5.1.2 Transmission Requirements 69
5.1.3 Processing and Transmission Requirements Combined 69
5.1.4 Summary 69
5.2 BOUND MAPPABLE POINT 70
5.2.1 Processing Requirements 70
5.2.2 Transmission Requirements 71
5.2.3 Processing and Transmission Requirements Combined 71
5.2.4 Summary 72
5.3 GRIDDED MAPPABLE POINT 72

5.3.1 Processing Requirements *72*
5.3.2 Transmission Requirements *74*
5.3.3 Processing and Transmission Requirements Combined *74*
5.3.4 Summary *75*
5.4 GRIDDED AND BOUND MAPPABLE POINT 75
5.4.1 Processing Requirements *75*
5.4.2 Transmission Requirements *77*
5.4.3 Processing and Transmission Requirements Combined *77*
5.4.4 Summary *77*
5.5 CACHED, GRIDDED AND BOUND MAPPABLE POINT 78
5.5.1 Processing Requirements *78*
5.5.2 Transmission Requirements *80*
5.5.3 Processing and Transmission Requirements Combined *80*
5.5.4 Summary *81*
5.6 COMPARISON OF VISUALISATION TECHNIQUES 81
5.7 ABSTRACT TESTING FRAMEWORK DISCUSSION 83
5.8 LIMITATIONS 83
5.8.1 Clustering – GeoJSON versus WKT *83*
5.8.2 Point Accuracy – Decimal Places *84*
5.8.3 Performance Relative to Hardware Specifications *84*
5.8.4 Cached, Gridded and Bound Mappable Point – Number of Grid Sizes *84*
6 CONCLUSION 86
7 REFERENCES 87
8 APPENDICES 88
8.1 RISK ASSESSMENT 88
8.2 RESULTS 90
8.2.1 Worldwide *90*
8.2.2 Australia Wide *105*
8.2.3 Brisbane Wide *120*
8.3 SOURCE CODE REPOSITORY DETAILS 135

Word Count: 18,737

Introduction

Geographic Information Systems (GIS) are used to store, manipulate, visualise and analyse spatial data [1]. Web GIS focuses on access to GIS via the World Wide Web (WWW) [2]. Web GIS enhances GIS spatial data access and dissemination; spatial data exploration and visualisation; and spatial data processing, analysis and modelling. A commonly used WebGIS application is Google Maps.

It can be difficult to design Web GIS services with good performance, as they typically require heavy central processing unit (CPU) usage, often transmit large data sets, and their client applications are usually complex [3].

1.1 Problem Definition

James Cook University eResearch Centre is developing a web-based application to provide interactive maps for Australian bird species. These maps will be required to show the recorded occurrences of species with more than 500,000 sightings. In addition to rendering these points on a map, it is also a requirement that the map allow clients to query and interact with the rendered points.

Without additional processing, geospatial data sets of only a limited size can be transmitted to and effectively rendered on a web browser. This limitation comes from various factors, including network speed, data set size and client rendering performance. Preliminary testing has shown that data sets in excess of 20,000 points cannot be rendered as feature layers without significant transmission delays, exceeding ten seconds for an ADSL (8 Mbit/s) connection, and a high probability of browser crash during rendering.

1.2 Project Aims

The goal of this project is to support James Cook University's development efforts by producing an effective solution to allow for very large geospatial data sets, in the order of 1,000,000 points, to be transmitted, and rendered on a web browser, such that the data can be interacted with in a user acceptable time frame. For the purpose of this project, we will define a user acceptable time frame as one second for an ADSL (8 Mbit/s) user.

To achieve this overall goal, the project consists of two aims. The first aim is to develop a set of metrics for analysing the performance of a geospatial technique. These metrics will be used to compare the visualisation techniques. These metrics should cover, at a minimum, data fidelity, server-side storage, server-side processing, and transmission requirements. The

second aim is to develop a toolbox to allow users to perform analysis on their geospatial data sets using the metrics developed. A user of the toolbox should be able to input his or her geospatial data into the toolbox. Once inserted, the user should be able to perform an automated analysis of their data, such that they are provided with a report on the effectiveness of the different geospatial processing techniques.

1.3 Project Outcomes

Once complete, this project will provide several key benefits to the wider GIS community. The tool suite developed will allow developers to compare different geospatial visualisation techniques simply by inputting their data set, rather than through the implementation of each visualisation technique they wish to compare. The new visualisation technique will allow developers to produce applications that interface with much larger geospatial data sets than they can conventionally, allowing for a range of new applications to be developed.

On its completion, this thesis will have answered two research questions. 1. How can geospatial visualisation techniques be compared? 2. What geospatial visualisation technique can be used to render and interact with geospatial data sets exceeding one million points?

2 Literature Review

This literature review will consist of three parts: Web GIS, Web GIS Performance Optimisations, and Conclusion. Web GIS will explore the existing standards and technologies used in modern Web GIS platforms, and will propose technologies for use in this project. Web GIS Performance Optimisations will explore the limitations within the GIS ecosystem that are pertinent to this project, and will propose methods for overcoming these limitations. The Literature Review's Conclusion will summarise the findings of this review.

2.1 Web GIS

A typical Web GIS will consist of a GIS database, providing geospatial data storage; a map server, for rendering raster maps; a web server, for responding to client requests; and a client application, for the client to interact with the geospatial data [4].

2.1.1 GIS Database

A Spatial Database Management System (DBMS) allows for the storage and querying of spatial data types, provides spatial indexing, and implements spatial queries [5]. A database (DB) that meets these requirements is referred to as spatially aware.

The Open Geospatial Consortium (OGC) defines the simple features specification (SFS) [6]. The OGC SFS standard defines an SQL schema that supports the storage, retrieval, query, and update of feature collections via SQL. In this context, a feature is defined as having both spatial and non-spatial attributes. The OGC SFS is followed by most Spatial DBMS [7]. Figure 1 shows the geometry type hierarchy as defined by the OGC SFS.

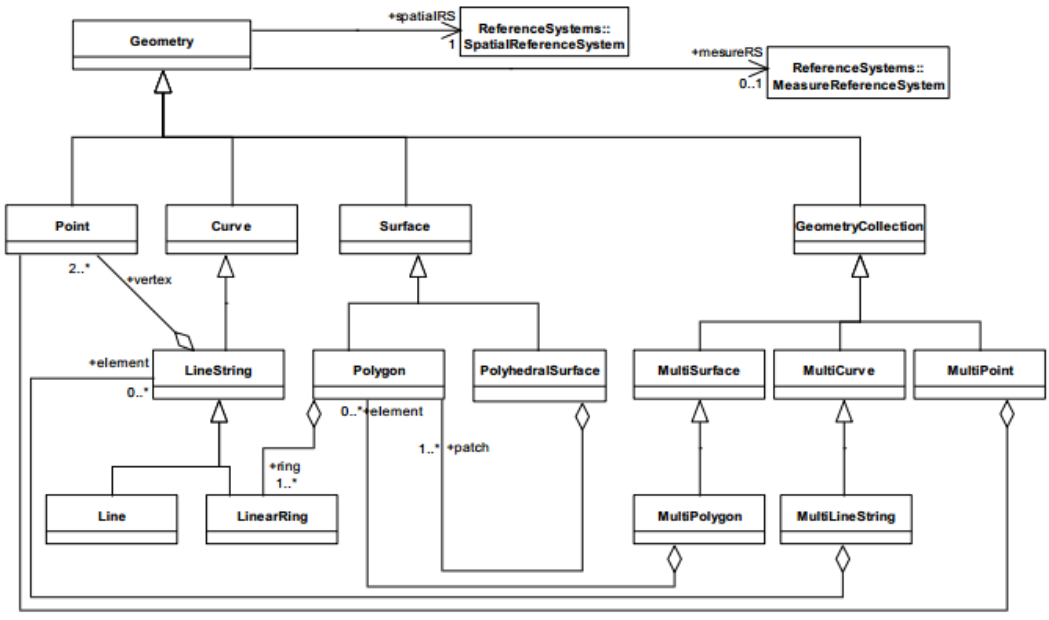

Figure 1: SQL Geometry Type Hierarchy [6]

Three popular Spatial DBMS are Oracle Spatial and Graph, PostGIS and MySQL [7]. Oracle Spatial and Graph is an optional component of Oracle Database. Both Oracle Database and Oracle Spatial and Graph are licensed software. PostGIS is a freely available extension to the open source DBMS PostgreSQL. MySQL is an open source DBMS that can be spatially enabled via inbuilt spatial extensions [8]. Both PostGIS and MySQL are fully compliant with the SQL implementation of the OGC SFS [7].

Surveys conducted by Ballatore et al. [7] evaluated the learning curve, stability, performance, scalability, interoperability, extendibility, standards, documentation, community support, and frequency of updates of PostGIS and MySQL. Ballatore et al. found that while PostGIS and MySQL had similar learning curves, PostGIS was rated higher in all other categories. A radar representation of the responses can be seen in Figure 2.

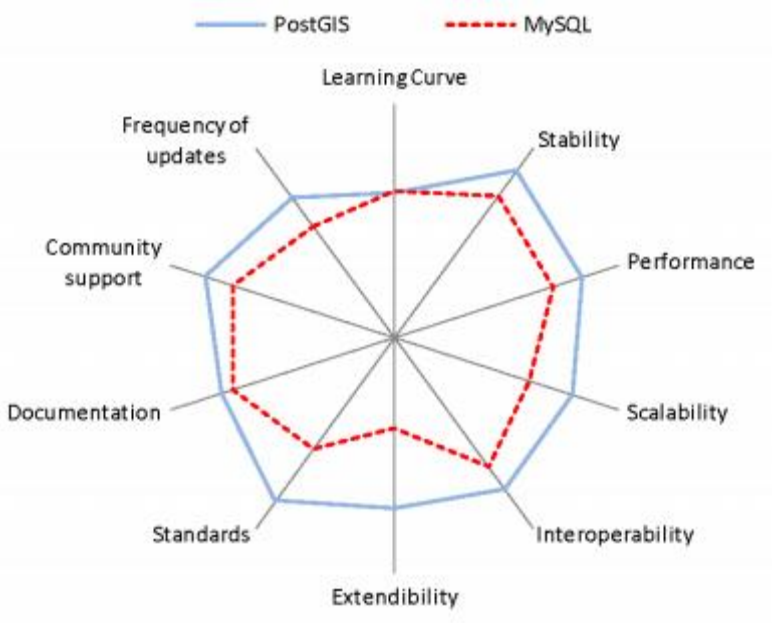

Figure 2: Spatial DBMS Comparison [7]

We propose using PostGIS as the Spatial DBMS for this project, as it is fully compliant with OGC SFS standards, it is open-source, and survey results show that developers prefer it to other open-source solutions.

2.1.2 Web Map Server

A web map server is a specialised web server that implements one or more of the following OGC standards: Web Mapping Services (WMS [9]), Web Feature Services (WFS [10]) and Web Coverage Services (WCS [11]) [12].

The WMS standard defines a HTTP interface for requesting geospatial map images from one or more geospatial DBMS [9]. A WMS request is a HTTP request that specifies the geographic layers, and area of interest to be rendered. The HTTP response to the request is one or more rendered map images (JPEG, PNG, etc.). These images are compatible with browser applications, and can be displayed directly. The WMS interface allows for various query options, including transparency, which allows multiple layers to be rendered to a single map. The OGC standard, Styled Layer Descriptor (SLD [13]), is an extension to WMS that allows for user defined symbolisation and colouring of geographic features.

The WFS standard defines a HTTP interface to allow for access and manipulation of geographic features [10]. A WFS interface allows a client to combine, use and manage geospatial data. Using WFS, a client can create features, delete features, update features, lock features, and query features based on spatial and non-spatial constraints. WFS query responses are returned as Geography Markup Language (GML [14]), an Extensible Markup Language (XML) grammar for expressing geographical features [12].

The WCS standard defines an interface for retrieving geospatial data as coverages [11]. A coverage is geospatial information representing space and time-varying phenomena. WCS provides access to coverage data in a variety of forms, allowing for client-side rendering, input into scientific models, etc. The WCS standard can output data as features, similar to WFS, or as images, similar to WMS [12].

Two of the best known web map servers are MapServer and GeoServer [12]. Both web map servers provide vector and raster support. Both conform to the WMS, WFS, WCS, GML and SLD OGC standards.

Surveys conducted by Ballatore et al. [7] evaluated the learning curve, stability, performance, scalability, interoperability, extendibility, standards, documentation, community support, and frequency of updates of GeoServer and MapServer. Ballatore et al. found that GeoServer was easier to extend, and was perceived to follow the standards more closely. MapServer was thought to be more stable, more scalable, and performed better. MapServer received more positive feedback than GeoServer. A radar representation of the responses can be seen in Figure 3. Note, the survey conducted by Ballatore et al. also evaluated JTS and GeoTools, two spatial libraries, the results associated with these software packages can be ignored.

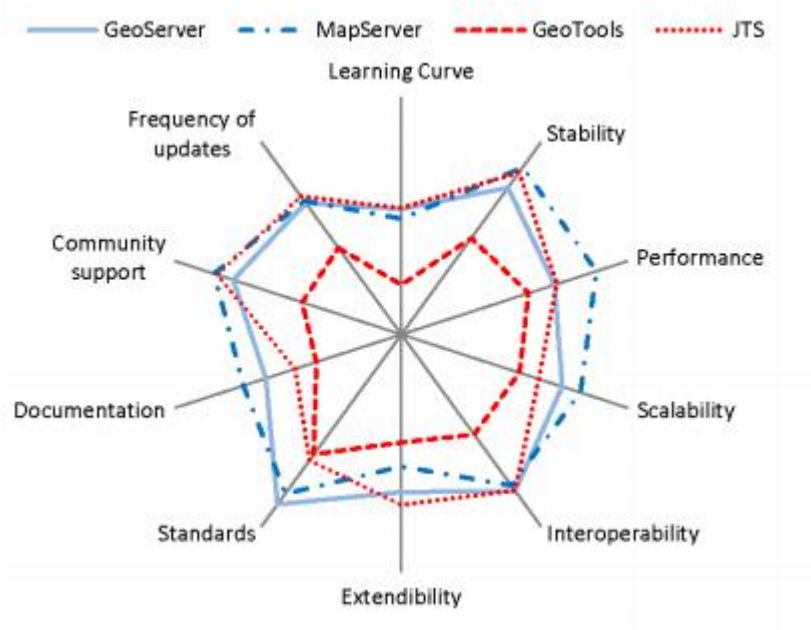

Figure 3: Web Mapping Servers and Spatial Libraries Comparison [7]

For the purpose of this project, our data will be time invariant, so we will not have any use of the WCS standard. Our input data consists of features, and as the WFS standard will allow us to query and interact with our features, it is an obvious choice for this project. The WMS standard will allow us to render our features as map images. Depending on what strategies we implement to interact with our map data, we should consider that that our performance may be better with rendered maps, rather than GML. For this reason, it is worth ensuring that our chosen web map server is WMS compliant. We propose using MapServer, and implementing a solution that makes use of the WMS and WFS standards.

2.1.3 Web Server

Grails and Ruby on Rails are two leading web application frameworks used to implement Web GIS servers [7]. Ruby on Rails is an open source web application framework developed in Ruby. Grails is a Ruby on Rails inspired open source web application framework developed in Groovy [15]. Grails is fully compatible with Java, running on the Java Virtual Machine (JVM). This is considered a significant advantage over Rails, as it allows Grails to leverage existing Java geospatial tools.

Surveys conducted by Ballatore et al. [7] evaluated the learning curve, stability, performance, scalability, interoperability, extendibility, standards, documentation, community support, and frequency of updates of Ruby on Rails and Grails. Ballatore et al. found that Grails was better in all areas except for stability. A radar representation of the responses can be seen in Figure 4. Note, the survey conducted by Ballatore et al. also

evaluated Hibernate and Hibernate Spatial, two object-relational mapping libraries, the results associated with these software packages can be ignored.

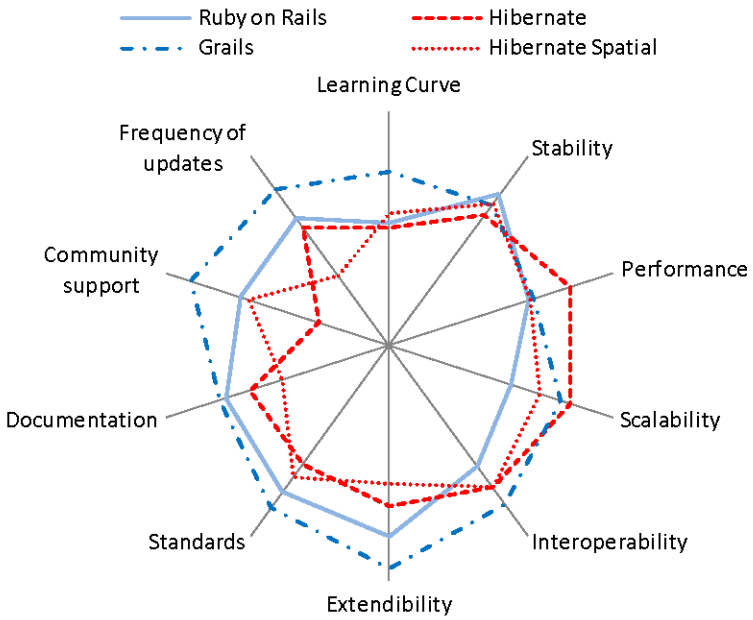

Figure 4: Web application frameworks and object-relational mapping [7]

Björemo et al. [16] evaluated the web frameworks CakePHP, Grails, Ruby on Rails, Stripes, Spring Roo and Wicket. Björemo et al. concluded that no framework is superior to the others, and that a skilled developer of a specific programming language should use a framework in the same language instead of learning a new language just to use a recommended framework.

We propose using Pyramid as the web application framework for this project. Pyramid is a light-weight web application framework written in Python. We propose using Pyramid because the authors have extensive experience in the programming language Python. If we did not have this previous experience in Python, we would recommend selecting the web application framework Grails.

2.1.4 Web GIS Client

The Web GIS client is responsible for allowing the user to interact with geospatial data through a virtual map. To do this, the Web GIS client must query the web server and map server to access the geospatial data. Geospatial data access is achieved using WMS, WFS, GeoJSON and other industry-standard protocols for geographic data access [7, 17].

Steiniger et al. [12] compared the popular Web GIS clients OpenLayers, Leaflet, and ModestMaps/Wax. Steiniger et al. noted that Leaflet and MostMaps/Wax compete with

OpenLayers only with respect to map tile rendering, and that OpenLayers offers significantly more functionality with respect to interactive and vector-based mapping tools than its competitors.

For the purpose of this project, we expect to be processing geospatial data sets consisting of point features. Point features are commonly rendered as points using a vector layer. We propose using OpenLayers to develop our Web GIS client application, as it has superior vector-based mapping tools, and allows the development of highly interactive applications.

2.1.5 Web GIS Performance Optimisations

Tu et al. [3] found that it can be difficult to design Web GIS services with good performance, as they typically require heavy CPU usage, often transmit large data sets, and their clients are usually complex.

Tu et al. [3] suggest that simply establishing connections between the individual components of Web GIS is not sufficient, and that performance should be a major consideration in the design of Web GIS services. To achieve higher performance, Tu et al. propose that as the data to transmit is often large, granularity of services deserves particular attention. A higher level of granularity can be achieved by providing services that deliver groups of capabilities, rather than a single capability. Additionally, Tu et al. suggest that the services provided should target specific attributes of the client, such as a bounding box.

2.1.6 Bounding Box

A bounding box, also referred to as a window or viewport, is a rectangular area, often defined by the co-ordinates minimum x (minx), minimum y (miny), maximum x (maxx), and maximum y (maxy). By targeting the client's bounding box, we can generally ignore all data points outside of the client's view. Depending on the client's bounding box, relative to the overall data set, this can significantly reduce the number of features that require processing.

The units of the client's bounding box co-ordinates are dependent on the projection of the client's virtual map. The projection allows the user to view the earth as a map on a flat surface, in this case a screen, and defines how the earth, a 3d spherical system, can be expressed in a 2d Cartesian co-ordinate system. Two common projections are The World Geodetic System 1984, EPSG:4326 (WGS 84), and Spherical Mercator, EPSG:3857 (WGS 84 / Pseudo-Mercator) [18]. In a latitude-longitude projection, such as WGS 84, the projection's unit of measurement is degrees latitude and longitude. In a Mercator projection, such as WGS 84 / Pseudo-Mercator, the projection's unit of measurement is meters.

2.1.7 Clustering

Clustering involves representing many points with a single cluster. OpenLayers provides built in support for clustering. OpenLayers' clustering strategy is client-based, meaning that the entire data set must be sent to the client, and then the client must process the data to determine the clusters. Client based clustering can increase the rendering performance of the client, as they only need render a smaller set of features, but it will not reduce the size of the data set that must be transmitted to the client. Clustering at the server however, reduces the amount of data that needs to be transmitted to the client, as only the clusters need to be transmitted to the client, and not the individual points. Clustering at the server also means that no client post-processing need occur.

2.1.8 Caching

Caching is a common technique used in web applications to reduce the amount of data transmitted to the user. Caching can occur at many locations, including at the client, a proxy, an Internet service provider (ISP), an app server, a web server and a DBMS. One of the main goals of the OpenGIS Web Map Tile Service Implementation Standard (WMTS) is to provide better cache support for repetitive requests, so as to allow the WMTS server to save resources, and improve performance [19]. The WMTS recommends, where possible, using the existing HTTP cache expiration and cache control headers to allow the client to cache the results of map requests.

2.2 Conclusion

In conclusion, we have examined the existing Web GIS technologies, and have proposed using technologies that are appropriate for this project. These technologies are PostGIS, MapServer, Pyramid and OpenLayers. We found that recommendations exist for increasing the performance of Web GIS systems. These recommendations included developing granular services, targeting client attributes and making use of caching techniques. Based on these recommendations, we have examined a variety of methods of developing a granular service that targets client attributes, and propose developing a granular web service that uses clustering, bounding boxes, and caching. We believe that by coupling these strategies, we can develop a Web GIS service that is capable of processing data sets with in excess of one million points. This process should act as pre-processor, meaning that the process can be executed prior to client requests, and that all client requests should be handled by an existing result cache. Theoretically, by caching the result of our clustering algorithm, this process should only be limited by the frequency of data updates, and data storage, and not by the size of the original data set. This process and its implementation will be discussed in further detail in the Methodology of this report.

3 Methodology

The Methodology will consist of four parts: Overview, Test System, Abstraction of Geospatial Visualisation Techniques, and Implemented Geospatial Visualisation Techniques. The Overview will provide an outline of the system, which will assist in understanding the implementation of both the test system, and the visualisation techniques. Test System will describe the software and hardware that was used for testing, the reproducibility of the system, what input data was used for testing, and what the test entailed. Abstraction of Geospatial Visualisation Techniques will describe why an abstraction was used, what benefits it provided, and how it was implemented. Implemented Geospatial Visualisation Techniques will describe the different visualisation techniques that were implemented, and why they were implemented.

3.1 Overview

Before we discuss the method, as it was applied, it is important to have an understanding of the overall system.

3.1.1 Proposed Method of Comparison

We will outline a method of comparing visualisation techniques, providing an answer to our first research question, 'How can geospatial visualisation techniques be compared?'

Through an underlying geospatial visualisation technique abstraction, we will measure and compare three types of performance. We will measure the computation time it takes to perform specific processes; the size of transmission ready result sets in different geospatial language formats, including GeoJSON, and WKT; and the size of the database storage required by the technique. The transmission ready result set sizes will also be described in terms of download times for specific network speeds: 4.7 Mbit/s, the average download speed of Australians [20]; and 8 Mbit/s ADSL.

The input data sets will be generated such that they are of different sizes, with different distributions. The output of the visualisation technique, at each zoom level, will be analysed to see how accurately the clustered data represents the original data set. This analysis will consist of measuring the number of clusters shown, versus the number of individual points those clusters represent.

We will define a set of test runs that test the different visualisation techniques under different use cases, varying in total required panning and zooming. The test runs, and their associated results, will be available via the toolkit.

3.1.2 Proposed High Performance Visualisation Technique

This section outlines a high performance method of visualising geospatial data sets. As noted in the Literature Review's Conclusion, section 2.2, we propose developing a Web GIS service that couples caching, clustering and bounding box strategies. We believe that testing will show that this method is effective for very large geospatial data sets, providing an answer to our second research question, 'What geospatial visualisation technique can be used to render and interact with geospatial data sets exceeding one million points?'

We propose using the client's bounding box to determine the user's viewable area. Based on this bounding box, we can determine the appropriate level of clustering to ensure that an appropriate number of clusters are transmitted to the client. The appropriate number of clusters is that which ensures the total data to be transmitted is small enough that it can be transmitted within the user acceptable timeframe, one second for an 8 Mbit/s connection. There is a linear relationship between the number of clusters and the size of the data to be transmitted. The appropriate number of clusters will be determined through experimentation. Preliminary results show that limiting transmission to 5,000 clusters is appropriate to ensure quick transmission of data to the client, and limit information overload. As the user zooms in, their bounding box will decrease in area, and they will be presented with a more accurate representation of the data. This will be performed by decreasing the coarseness of the clustering algorithm as the user zooms in.

The clustering algorithm will be implemented using the SQL implementation of the OGC SFS. PostGIS provides a function called ST_SnapToGrid [21]. ST_SnapToGrid snaps all points of an input geometry to a rectangular grid. The size of the grid can be specified, and will be used to define our clustering coarseness. Once these points have been snapped, they form an aggregate cluster. These clusters can be iterated over, allowing for additional processing of the points within the cluster. We propose iterating over these clusters, and computing the cluster's centroid. The centroid will represent the cluster as a single point, meaning we can significantly reduce our transmission requirements. The centroid can be seen in Figure 5.

Figure 5: Centroid of a multipoint input geometry [21]

In addition to using clustering, we also propose using caching. We propose coupling the clustering and caching strategies by determining the clusters worldwide for the different zoom levels, and storing the results in cache tables within our database. When a client then makes a request, we will use the area of their bounding box to normalise the request to the appropriate cached zoom level. We then query the cached table using the client's bounding box to limit the result to only clusters in the viewable area. When the original source of the cached data is updated, we will invalidate our cache, requiring reprocessing of the data set. The number of zoom levels we produce cached cluster results for will define the size of the data sets we need to store.

Now that we have given an overall description of how we intend to use the strategies, we will describe how the process will operate, step by step. When the data set is initially added to the Spatial DBMS, and whenever the data set changes, the pre-process will be executed. This pre-process consists of executing the clustering algorithm against the data set once for each zoom level we wish to cache. Each zoom level will have a defined degree of coarseness associated with it. As each of the clustering algorithm executions complete, the result will be stored in the database, referencing the coarseness that was used to produce the result set. This concludes the pre-processing phase. At this point, the data set has been clustered and cached for various zoom levels, and is ready for client queries.

We will now look at the client-interaction phase of the process. The client will initially load a webpage with a map on it, provided via OpenLayers. The user will be viewing a certain section of earth on the map. The bound of the client's viewable area is the bounding box. When the user initially loads the map, and whenever the user's bounding box changes, via zooming or panning, OpenLayers will request the features for the map using a Geospatial standard, such as WKT or GeoJSON, and will include the client's bounding box in the

request. This request will be sent to the Web Server, Pyramid. The Web Server will then examine the bounds of the request, and will determine the appropriate clustering coarseness for the client's bounds. As the server will only have cached clustering results for specific coarseness values, the server will normalise the client's clustering coarseness to a coarseness that has been cached. Once the clustering coarseness has been determined, the Web Server will then make an OGC SFS feature request to the Spatial DBMS. This feature request will be made against the cache table for the determined clustering coarseness. The feature request will use the client's bounding box as a spatial overlap to the DBMS request. The spatial overlap operator will ensure that only the clusters within the client's bounds will be returned via the request. The web server will then forward the DBMS response onto the client. This response will contain the position and size of the clusters. Once OpenLayers receives the response, it will render the cluster features, using dot size to represent cluster size.

3.2 Abstraction of Geospatial Visualisation Techniques

In order to directly compare visualisation techniques, we abstracted the core elements of visualisation techniques. In object-orientated-programming, this abstraction is referred to as a base class.

3.2.1 Extensible and Pluggable

The use of an abstraction means that another developer can produce extensions to the abstraction. These extensions can be directly plugged into our toolkit, as they extend the common abstraction. This means that the toolkit is future proofed, allowing the set of understood geospatial visualisation techniques to expand beyond those implemented in this thesis.

3.2.2 Analysis of Performance

By using an abstraction, we were able to embed performance analysis code directly into the underlying abstraction. This performance analysis code exists as a wrapper around the extension's implementation of the visualisation technique, and allows us to analyse the implementation. This means that regardless of how the individual geospatial visualisation technique is implemented, we can rely on the consistency of the performance analysis provided by the underlying abstraction. This consistency allows us to directly compare different visualisation techniques, without the need for special case handling, regardless of the implementation's complexity.

3.2.3 Abstraction Description

We defined a common abstraction that understood three functions. These functions are:

1. Pre-process (*pre_process*)

2. Get points as GeoJSON (*get_points_as_geojson*)
3. Get points as WKT (*get_points_as_wkt*).

3.2.3.1 Pre-Process

The *pre_process* function will be called against a visualisation technique once prior to any other requests. This function provides the visualisation technique with an opportunity to perform pre-processing logic prior to any client requests. No return value is expected from this function, and the implementation of this function is optional, i.e. a visualisation technique is not required to implement any pre-processing logic.

3.2.3.2 Get Points as GeoJSON

The *get_points_as_geojson* function will be called on every GeoJSON request, and will be provided, at a minimum, the client's bounding box (bbox). The visualisation technique is expected to implement this function. This function is expected to return a query object containing a set of clusters, and their associated centroid and cluster size. The cluster centroid must be a GeoJSON string representing an OGC SFS point object and the cluster size must be an integer describing the number of points the cluster represents.

3.2.3.3 Get Points as WKT

The expectations of the visualisation technique's implementation of the *get_points_as_wkt* function are identical to that of the *get_points_as_geojson* function, except that the cluster centroid must be a WKT string.

3.2.4 Abstraction Implementation

In addition to defining the expectations of the three functions detailed in the Abstraction Description, section 3.2.3, we also implemented other functions that provide the underlying test harness, allowing for automated performance testing. These functions are:

1. Test pre-process (*test_pre_process*)
2. Get points as GeoJSON string (*get_points_as_geojson_str*)
3. Test get points as GeoJSON (*test_get_points_as_geojson*)
4. Test get points as GeoJSON string (*test_get_points_as_geojson_str*)
5. Get points as WKT string (*get_points_as_wkt_str*)
6. Test get points as WKT(*test_get_points_as_wkt*)
7. Test get points as WKT string (*test_get_points_as_wkt_str*)

3.2.4.1 Test Pre-Process

The *test_pre_process* function is a light wrapper around the visualisation technique's *pre_process* function that analyses the time it takes to perform the *pre_process* function. Figure 6 contains the pseudo-code description of this function.

```
# arguments** are the input arguments (bounding box, etc.)
# class refers to the implementation
function test_pre_process(arguments**):

  start_time = datetime.now()                  # pre-process start time
  result     = class.pre_process(arguments**)  # call the implementation's function
  end_time   = datetime.now()                  # pre-process end time

  delta_time = end_time - start_time           # how long pre-process took

  log(delta_time)                              # log how long pre-processing took

  return result

end_function
```

Figure 6 Test pre-process pseudo code

Additionally, the test harness uses the start and end of the *test_pre_process* function to capture database size information. This allowed us to determine how much additional database space the individual visualisation techniques require.

3.2.4.2 Get Points as GeoJSON String

The *get_points_as_geojson_str* function is responsible for converting the visualisation technique's *get_points_as_geojson* function result into a single GeoJSON string, consumable by all standards compliant map clients. Effectively, this function is responsible for putting all the cluster results into a single feature collection represented by a GeoJSON string.

GeoJSON allows for the transmission of non-geospatial data, and as such, we also embed the cluster size information in the GeoJSON string result.

3.2.4.3 Test Get Points as GeoJSON

This function is a light wrapper around the *get_points_as_geojson* function described in the Abstraction Description, section 3.2.3. This wrapper captures the amount of time spent in the *get_points_as_geojson* function.

3.2.4.4 Test Get Points as GeoJSON String

This function is a light wrapper around the *get_points_as_geojson_str* function. By incorporating this function and the *test_get_points_as_geojson* function, we can determine how much time was spent at the database layer, capturing the cluster information, versus how much time is taken building the resulting GeoJSON string.

3.2.4.5 Get Points as WKT String

The *get_points_as_wkt_str* function is responsible for converting the visualisation technique's *get_points_as_wkt* function result into a single WKT string. Specifically, the centroid of each cluster, represented by a WKT POINT, is joined to form a single WKT GEOMETRYCOLLECTION. This WKT GEOMETRYCOLLECTION can then be consumed by any standards compliant map client.

It should be noted that, unlike GeoJSON, the WKT representation of the clusters cannot contain any information beyond geospatial information, and as such, the WKT representation of the clusters does not contain the cluster sizes.

3.2.4.6 Test Get Points as WKT

This function is a light wrapper around the *get_points_as_wkt* function described in the Abstraction Description, section 3.2.3. This wrapper captures the amount of time spent in the *get_points_as_wkt* function.

3.2.4.7 Test Get Points as WKT String

This function is a light wrapper around the *get_points_as_wkt_str* function. By incorporating this function and the *test_get_points_as_wkt* function, we can determine how much time was spent at the database layer, capturing the cluster information, versus how much time is taken building the resulting WKT string.

3.3 Implemented Geospatial Visualisation Techniques

As noted in the Literature Review's Conclusion, section 2.2, we proposed developing a Web GIS service that couples bounding box, clustering via gridding, and caching strategies. In order to analyse the benefits of using these strategies individually and combined, we implemented the following five visualisation techniques:

1. Mappable Point
2. Gridded Mappable Point
3. Bound Mappable Point
4. Gridded and Bound Mappable Point
5. Cached, Gridded and Bound Mappable Point

3.3.1 Mappable Point

Mappable Point is the most basic of the visualisation technique implementations. The Mappable Point technique does not employ bounding box, clustering via gridding or caching strategies. The Mappable Point technique will act as our control, providing a benchmark for all other visualisation technique implementations.

3.3.1.1 Pre-Process

The Mappable Point technique does not have any pre-processing logic, i.e. the Mappable Point's pre-process function is empty.

3.3.1.2 Get Points as GeoJSON

The Mappable Point technique's *get_points_as_geojson* function groups all points by their location, creating clusters of points with the exact same location. As the Mappable Point does not understand the client's bounding box, it will return every point, clustered by exact location, for every request. The one benefit of this is that all future requests the client would have normally made can be served by the result of their initial request, i.e. the client needs to only make a single request for the data set regardless of the number of interaction steps they intend to make.

The *get_points_as_geojson* function returns the centroid of each cluster as a GeoJSON string, and the number of points within each cluster as an integer. As all the points within each cluster have the same location, we can assume the centroid is the location of the points. Figure 7 contains the pseudo code for this function.

```
# Query the database for:
#   - the point's location as GeoJSON, and label it "centroid"
#   - the number of points at this location, and label it "cluster_size".
#
# To get the number of points at a given location, we should
# group the query by location.
function get_points_as_geojson():

    query = new database_query(
        group_by Point.location label_it Cluster

        get Cluster.location as GeoJSON label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 7: Mappable Point's *get_points_as_geojson* function pseudo code

3.3.1.3 Get Points as WKT

The Mappable Point technique's *get_points_as_wkt* function is identical to the *get_points_as_geojson* function, except that the cluster centroids are returned as WKT, rather than GeoJSON. Figure 8 contains the pseudo code for this function.

```
# Query the database for:
#   - the point's location as WKT, and label it "centroid"
#   - the number of points at this location, and label it "cluster_size".
#
# To get the number of points at a given location, we should
# group the query by location.
function get_points_as_wkt():

    query = new database_query(
        group_by Point.location label_it Cluster

        get Cluster.location as WKT label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 8: Mappable Point's *get_points_as_wkt* function pseudo code

3.3.2 Bound Mappable Point

The Bound Mappable Point visualisation technique makes use of the client's bounding box, only returning points within the bounds of the user's request. The Bound Mappable Point technique does not employ clustering via gridding or caching strategies. The Bound Mappable Point technique will evaluate the bounding box strategy used in isolation.

3.3.2.1 Pre-Process

The Bound Mappable Point technique does not have any pre-processing logic, i.e. the Bound Mappable Point's pre-process function is empty.

3.3.2.2 Get Points as GeoJSON

The Bound Mappable Point technique's *get_points_as_geojson* function filters out all points outside the client's bounding box. Once this filtering has been done, the points are then grouped by their location, creating clusters of points with the exact same location. Unlike Mappable Point, the Bound Mappable Point technique will only return the points within the client's bounding box. This means that the client will need to make a new request for every interaction that is outside the bounds of their previous requests.

The *get_points_as_geojson* function returns the centroid of each cluster as a GeoJSON string, and the number of points within each cluster as an integer. As all the points within each cluster have the same location, we can assume the centroid is the location of the points. Figure 9 contains the pseudo code for this function.

```
# Query the database for:
#   - the point's location as GeoJSON, and label it "centroid"
#   - the number of points at this location, and label it "cluster_size".
#
# Filter the query so that only points that intersect with the
# client's bounding box are processed.
#
# To get the number of points at a given location, we should
# group the query by location.
#
# Inputs
#   - bbox: The client's input bounding box
function get_points_as_geojson(bbox):

    query = new database_query(
        filter_by Point.location that_intersect_with bbox

        group_by Point.location label_it Cluster

        get Cluster.location as GeoJSON label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 9: Bound Mappable Point's *get_points_as_geojson* function pseudo code

3.3.2.3 Get Points as WKT

The Bound Mappable Point technique's *get_points_as_wkt* function is identical to the *get_points_as_geojson* function, except that the cluster centroids are returned as WKT, rather than GeoJSON. Figure 10 contains the pseudo code for this function.

```
# Query the database for:
#   - the point's location as WKT, and label it "centroid"
#   - the number of points at this location, and label it "cluster_size".
#
# Filter the query so that only points that intersect with the
# client's bounding box are processed.
#
# To get the number of points at a given location, we should
# group the query by location.
#
# Inputs:
#   - bbox: The client's input bounding box
function get_points_as_wkt(bbox):

    query = new database_query(
        filter_by Point.location that_intersect_with bbox

        group_by Point.location label_it Cluster

        get Cluster.location as WKT label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 10: Bound Mappable Point's *get_points_as_wkt* function pseudo code

3.3.3 Gridded Mappable Point

The Gridded Mappable Point visualisation technique uses the PostGIS function, ST_SnapToGrid [21] to perform clustering. ST_SnapToGrid snaps all points of an input geometry to a rectangular grid. The size of the grid can be specified, and is used to define clustering coarseness. Once the points have been snapped, they form an aggregate cluster. The clusters can then be iterated over, allowing for additional processing of the points within the cluster.

The Gridded Mappable Point makes use of the client's bounding box to determine an appropriate clustering coarseness, but does not use the bounding box to filter the processed points. The Gridded Mappable Point technique will evaluate using the ST_SnapToGrid function, as a clustering strategy, in isolation.

3.3.3.1 Pre-Process

The Gridded Mappable Point technique does not have any pre-processing logic, i.e. the Gridded Mappable Point's pre-process function is empty.

3.3.3.2 Get Points as GeoJSON

The Gridded Mappable Point technique's *get_points_as_geojson* function produces clusters using the PostGIS function, ST_SnapToGrid. In order to use the ST_SnapToGrid function, we must first determine a grid size. The grid size is determined by taking the user's viewable bounds, and calculating a grid size, such that, assuming every grid contains a point, the user will see at most 100 items in their viewable window.

There is a linear relationship between the number of clusters and the size of the data to be transmitted. One hundred items was selected as the target number of viewable items, as we believe that it will provide sufficient visual information to the user without overwhelming, the user visually, or the transmission requirements of the system. The transmission requirements of the system are that the total data be transmitted within a user acceptable timeframe, one second for an 8 Mbit/s connection. The implications of this decision will be described in the Discussion, section 5.

The calculation to determine the grid size is as follows:

1. The user's bounds are broken into a latitude range, and a longitude range.
2. The average of these ranges is found.
3. Once the average is found, we then divide it by the square root of the number of clusters we would ideally like to see. In this case, as we would like to see at most 100 clusters, we divide the average range by 10 (√100).

Figure 11 contains the pseudo code for this function.

```
# The grid size is the span of the window divided by GRID_SIZE_WINDOW_FRACTION
# The total number of grids will, on average, be GRID_SIZE_WINDOW_FRACTION^2
GRID_SIZE_WINDOW_FRACTION = 10

# Given a bounding box (bbox), determine an appropriate
# grid size to use.
#
# Inputs:
#   - bbox: the client's bounding box
function get_cluster_grid_size(bbox):

    # Get the West, South, East and North values out of the bounding box
    w, s, e, n = bbox

    latitude_range = absolute_value_of(w - e)
    longitude_range = absolute_value_of(n - s)

    latitude_longitude_range_avg = (latitude_range + longitude_range) / 2

    grid_size = latitude_longitude_range_avg / GRID_SIZE_WINDOW_FRACTION
    grid_size = round grid_size to 3 decimal_places

    return grid_size

end_function
```

Figure 11: Gridded Mappable Point's *get_cluster_grid_size* function pseudo code

Unlike Mappable Point, the Gridded Mappable Point technique will only return the clusters appropriate for the current zoom level and window size. This is because the client's bounding box was used to determine the grid size. As the result is not bound, the client can use the result to pan the map. A new request should be made if the client's zoom level changes, or if the map window is resized.

The *get_points_as_geojson* function returns the centroid of each cluster as a GeoJSON string, and the number of points within each cluster as an integer. As the points within a cluster do not all share the same location, the cluster centroid will need to be calculated. The calculation of a weighted centroid is performed by the PostGIS function, ST_Centroid [21]. ST_Centroid calculates the arithmetic mean of the input co-ordinates. Figure 12 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function
#
# Query the database for:
#   - the centroid of the cluster's locations as GeoJSON, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size".
#
# To cluster the points, we group the points into clusters
# using the ST_SnapToGrid function.
#
# Inputs
#   - bbox: The client's input bounding box
function get_points_as_geojson(bbox):

    grid_size = get_cluster_grid_size(bbox)

    query = new database_query(
        group_by ST_SnapToGrid(Point.location, grid_size) label_it Cluster

        get ST_Centroid(Cluster.locations) as GeoJSON label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 12: Gridded Mappable Point's *get_points_as_geojson* function pseudo code

3.3.3.3 Get Points as WKT

The Gridded Mappable Point technique's *get_points_as_wkt* function is identical to the *get_points_as_geojson* function, except that the cluster centroids are returned as WKT, rather than GeoJSON. Figure 13 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function
#
# Query the database for:
#   - the centroid of the cluster's locations as WKT, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size".
#
# To cluster the points, we group the points into clusters
# using the ST_SnapToGrid function.
#
# Inputs
#   - bbox: The client's input bounding box
function get_points_as_wkt(bbox):

    grid_size = get_cluster_grid_size(bbox)

    query = new database_query(
        group_by ST_SnapToGrid(Point.location, grid_size) label_it Cluster

        get ST_Centroid(Cluster.locations) as WKT label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )

    return query

end_function
```

Figure 13: Gridded Mappable Point's *get_points_as_wkt* function pseudo code

3.3.4 Gridded and Bound Mappable Point

The Gridded and Bound Mappable Point visualisation technique is a hybrid, combining the Bound Mappable Point's use of the bounding box for filtering points with the Gridded Mappable Point's use of a grid for clustering.

3.3.4.1 Pre-Process

The Gridded and Bound Mappable Point technique does not have any pre-processing logic, i.e. the Gridded and Bound Mappable Point's pre-process function is empty.

3.3.4.2 Get Points as GeoJSON

The Gridded and Bound Mappable Point technique's *get_points_as_geojson* function filters out all points outside the client's bounding box, and then produces clusters of the remaining points using the PostGIS function, ST_SnapToGrid.

The grid size used as an input to the ST_SnapToGrid function is determined using the same method as described for the Gridded Mappable Point technique. The Gridded and Bound Mappable Point technique will only return the clusters appropriate for the current zoom

level, within the current bounds. This means that every interaction will require a new request.

The *get_points_as_geojson* function returns the centroid of each cluster as a GeoJSON string, and the number of points within each cluster as an integer. As was the case with the Gridded Mappable Point, the points within the clusters do not all share the same location, and so the cluster centroid will need to be calculated. The calculation of the centroid is performed using the PostGIS function, ST_Centroid. Figure 14 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function
#
# Query the database for:
#   - the centroid of the cluster's locations as GeoJSON, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size"
#
# Filter the query so that only points that intersect with the
# client's bounding box are processed.
#
# To cluster the points, we group the points into clusters
# using the ST_SnapToGrid function.
#
# Inputs
#   - bbox: The client's input bounding box
function get_points_as_geojson(bbox):

    grid_size = get_cluster_grid_size(bbox)

    query = new database_query(
        filter_by Point.location that_intersect_with bbox

        group_by ST_SnapToGrid(Point.location, grid_size) label_it Cluster

        get ST_Centroid(Cluster.locations) as GeoJSON label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )


    return query

end_function
```

Figure 14: Gridded and Bound Mappable Point's *get_points_as_geojson* function pseudo code

3.3.4.3 Get Points as WKT

The Gridded Mappable Point technique's *get_points_as_wkt* function is identical to the *get_points_as_geojson* function, except that the cluster centroids are returned as WKT, rather than GeoJSON. Figure 15 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function
#
# Query the database for:
#   - the centroid of the cluster's locations as WKT, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size".
#
# Filter the query so that only points that intersect with the
# client's bounding box are processed.
#
# To cluster the points, we group the points into clusters
# using the ST_SnapToGrid function.
#
# Inputs
#   - bbox: The client's input bounding box
function get_points_as_wkt(bbox):

    grid_size = get_cluster_grid_size(bbox)

    query = new database_query(
        filter_by Point.location that_intersect_with bbox

        group_by ST_SnapToGrid(Point.location, grid_size) label_it Cluster

        get ST_Centroid(Cluster.locations) as WKT label_it "centroid"
        count Cluster.locations label_it "cluster_size"
    )


    return query

end_function
```

Figure 15: Gridded and Bound Mappable Point's *get_points_as_wkt* function pseudo code

3.3.5 Cached, Gridded and Bound Mappable Point

The Cached, Gridded and Bound Mappable Point visualisation technique adds a cache layer to the Gridded and Bound Mappable Point. The Gridded and Bound Mappable Point implementation will be used to determine the clusters worldwide for a fixed set of different zoom levels. The results will be stored in cache tables within our database. When a client makes a request, we use the area of their bounding box to normalise the request to the appropriate cached zoom level, and query the cached table using the client's bounding box to limit the result to only clusters in the viewable area. When the original source of the cached data is updated, we invalidate our cache, requiring reprocessing of the data set. The number of zoom levels we produce cached cluster results for defines the size of the data sets we need to store.

The cache table will consist of Cache Records, and Cached Mappable Point Clusters. Each cached cluster will be stored as a single Cached Mappable Point Cluster. The set of Cached Mappable Point Clusters for a single zoom level will all be referenced through a single Cache Record, meaning that for each zoom level, a single Cache Record will exist. The Cache Record provides a zoom level index, making it possible to find all Cached Mappable Point Clusters for a single zoom level.

The Cached Mappable Point Cluster consists of: a centroid, represented by an OGC SFS Point; a cluster size, represented by an integer; and a mandatory reference to a Cache Record. The pseudo code description of this class is contained in Figure 16.

```
# The Cached Mappable Point Cluster represents a Database Record.
#
# A Cached Mappable Point MUST refer to a Cache Record.
#
# A Cached Mappable Point must:
#  - refer to a Cache Record
#  - contain a cluster size (as an Integer)
#  - contain a centroid, as a OGC SFS Point
class CachedMappablePointCluster():

    # reference to this cached cluster's cache record
    cache_record = required_relationship CacheRecord

    # The number of points this cluster represents
    cluster_size = database_column_of_type Integer

    # The weighted centroid of this cluster
    centroid     = database_column_of_type Point

end_class
```

Figure 16: Cached Mappable Point Cluster pseudo code

The Cache Record consists of a zoom level, represented by a floating point number (Float). The Cache Record also contains a function, *get_clusters*, to find all Cached Mappable Point Clusters associated with it. The pseudo code description of this class is contained in Figure 17.

```
# The Cache Record represents a database record.
#
# A Cache Record will be referenced by many Cached Mappable Point Clusters
#
# A Cache Record must:
#  - contain a zoom level
class CacheRecord():

    # The zoom_level this cache record represents
    zoom_level = database_column_of_type Float

    # The get_clusters function should return all the
    # Cached Mappable Point Clusters associated with this Cache Record
    function get_clusters():

        # Get all the clusters that reference us as their
        # Cache Record
        clusters = new database_query(
            get CachedMappablePointCluster where cache_record references THIS:CacheRecord
        )

        return clusters

    end_function

end_class
```

Figure 17: Cache Record pseudo code

3.3.5.1 Pre-Process

The Cached, Gridded and Bound Mappable Point uses the *pre_process* function to fill the aforementioned cache table. This involves producing and storing the Worldwide clusters for every zoom level we wish to support.

As zoom levels differ between different client implementations, the supported zoom levels are inferred from the set of supported GRID_SIZES defined in the Cached, Gridded and Bound Mappable Point. The Gridded and Bound Mappable Point's *get_points_as_wkt* function is used to determine the clusters for each grid size. The resulting clusters are stored in the cache table as Cached Mappable Point Clusters. The pseudo code description of this function is contained in Figure 18.

```
# The possible grid sizes that should be used (the normalised grid sizes)
# Note: values in decimal degrees (latitude/longitude)
GRID_SIZES = [0, 0.015, 0.03125, 0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128]

# Iterate over each of the possible GRID_SIZES.
#
# For each grid_size, create a cache record, and calculate
# all known clusters across the world.
#
# Once we have the clusters, iterate over those, and store them as
# CachedMappablePointClusters. Each CachedMappablePointCluster should
# have a reference to the appropriate Cache Record.
function pre_process():

    # Iterate over every supported grid size
    for_each grid_size in GRID_SIZES:

        # Create an empty CacheRecord for this grid size
        cache_record = new CacheRecord with grid_size defined

        # Get the clusters for this zoom level, using the
        # existing GriddedAndBoundMappablePoint technique
        clusters = GriddedAndBoundMappablePoint.get_points_as_wkt(
            with grid_size: grid_size
            with bounds:    WORLD-WIDE
        )

        # Iterate over every cluster that was returned
        for_each cluster in clusters:

            # Create a CachedMappablePointCluster from the cluster
            cached_mappable_cluser = new CachedMappablePointCluster(cluster)
            # Store the CachedMappablePointCluster against the current CacheRecord
            cache_record.append cached_mappable_cluster

        end_for_each

    end_for_each

end_function
```

Figure 18: Cached, Gridded and Bound Mappable Point's *pre_process* function pseudo code

As noted above, the supported zoom levels are inferred from the set of supported GRID_SIZES. This means that should you wish to support more, or less, zoom levels, you need only alter the GRID_SIZES variable accordingly. This ease of customisation makes this visualisation technique very flexible.

3.3.5.2 Get Points as GeoJSON

The Cached, Gridded and Bound Mappable Point technique's *get_points_as_geojson* function uses the cached clusters that were produced as a result of the *pre_process*. This makes this technique unique, in that during the client request, it will not need to interact with the individual points directly.

In a similar fashion to the Gridded and Bound Mappable Point technique, the Cached, Gridded and Bound Mappable Point will process the client's bounding box, determining the most appropriate grid size to use. As the Cached, Gridded and Bound Mappable Point technique only has a limited number of grid sizes cached, the determined grid size will be normalised to match one of those stored. Figure 19 contains the pseudo code for normalising the grid size.

```
# The possible grid sizes that should be used (the normalised grid sizes)
# Note: values in decimal degrees (latitude/longitude)
GRID_SIZES = [0, 0.015, 0.03125, 0.0625, 0.125, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128]

# Given an input grid size (in_grid_size), find the same size,
# or next largest grid size we have cached.
#
# Inputs
#   - in_grid_size: The grid size we would ideally use (if we had infinite cache)
function normalise_grid_size(in_grid_size):

    # The grid_size we we will eventually return
    return_grid_size = 0

    # Sort the grid sizes such that the smallest candidate grid sizes are first
    sorted_grid_sizes = sort_smallest_to_largest(GRID_SIZES)

    # Iterate over the possible grid sizes
    for_each candidate_grid_size in sorted_grid_sizes:

        # If we are larger (or equal) to this grid size, normalise to it.
        if in_grid_size greater_or_equal_to candidate_grid_size:
            return_grid_size = candidate_grid_size
        end_if

    end_for_each

    # This will be the last grid size we found that we were larger than (or equal to).
    return return_grid_size

end_function
```

Figure 19: Gridded and Bound Mappable Point's *normalise_grid_size* function pseudo code

The Cached, Gridded and Bound Mappable Point uses the client's bounding box to filter out cached clusters that fall outside the client's bounding box. As this technique only returns the

clusters appropriate for the current zoom level, within the current bounds, every interaction will require a new request.

The *get_points_as_geojson* function returns the centroid of each cluster as a GeoJSON string, and the number of points within each cluster as an integer. As this technique operates against the Cached Mappable Point Clusters, neither the centroid, nor the cluster count will need to be calculated. Figure 20 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function.
#
# Once the ideal grid size is determined, normalise it
# to one of the cached grid sizes.
#
# Find the cache record that is for the normalised grid size.
#
# Query the database for:
#   - the centroid of the cluster's locations as GeoJSON, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size"
#
# Filter the query so that only clusters with centroids that intersect with the
# client's bounding box are processed.
#
# Filter the query so that only clusters for the appropriate Cache Record
# are processed.
#
# Inputs:
#   - bbox: The client's input bounding box
function get_points_as_geojson(bbox):

    # If no grid size was provided, calculate it from the bbox
    grid_size = get_cluster_grid_size(bbox)

    # Normalise our grid size to one of the standard grid sizes
    normalised_grid_size = normalise_grid_size(grid_size)

    cache_record = CacheRecord where grid_size is normalised_grid_size

    query = new database_query(
        filter_by CachedMappablePointCluster.centroid that_intersect_with bbox

        filter_by CachedMappablePointCluster.cache_record references cache_record

        get CachedMappablePointCluster.centroid as GeoJSON label_it "centroid"
        get CachedMappablePointCluster.cluster_size label_it "cluster_size"
    )

    return query

end_function
```

Figure 20: Cached, Gridded and Bound Mappable Point's *get_points_as_geojson* function pseudo code

3.3.5.3 Get Points as WKT

The Cached, Gridded and Bound Mappable Point technique's *get_points_as_wkt* function is identical to the *get_points_as_geojson* function, except that the cluster centroids are returned as WKT, rather than GeoJSON. Figure 21 contains the pseudo code for this function.

```
# Determine the appropriate grid size for this bounding box (bbox)
# using the get_cluster_grid_size function.
#
# Once the ideal grid size is determined, normalise it
# to one of the cached grid sizes.
#
# Find the cache record that is for the normalised grid size.
#
# Query the database for:
#   - the centroid of the cluster's locations as WKT, and label it "centroid"
#   - the number of points in this cluster, and label it "cluster_size"
#
# Filter the query so that only clusters with centroids that intersect with the
# client's bounding box are processed.
#
# Filter the query so that only clusters for the appropriate Cache Record
# are processed.
#
# Inputs:
#   - bbox: The client's input bounding box
function get_points_as_wkt(bbox):

    # If no grid size was provided, calculate it from the bbox
    grid_size = get_cluster_grid_size(bbox)

    # Normalise our grid size to one of the standard grid sizes
    normalised_grid_size = normalise_grid_size(grid_size)

    cache_record = CacheRecord where grid_size is normalised_grid_size

    query = new database_query(
        filter_by CachedMappablePointCluster.centroid that_intersect_with bbox

        filter_by CachedMappablePointCluster.cache_record references cache_record

        get CachedMappablePointCluster.centroid as WKT label_it "centroid"
        get CachedMappablePointCluster.cluster_size label_it "cluster_size"
    )

    return query

end_function
```

Figure 21: Cached, Gridded and Bound Mappable Point's *get_points_as_geojson* function pseudo code

3.4 Test System

This section of the report will describe the overall test system, including the hardware and software materials used, as well as how the testing was performed. Web GIS Foundation will describe the software components that form the Web GIS we used. Reproducible and Consistent Testing Environment will describe the benefits of using Amazon's Elastic Compute Cloud (EC2) for producing test results. Input Data Sets will describe the input data sets that were used, and how they were generated. Test Scenario will describe how the individual test runs were conducted, and what they consisted of.

3.4.1 Web GIS Foundation

As outlined in the Literature Review, section 2.1, a typical Web GIS will consist of a GIS database, a map server, a web server, and a client application. These technologies comprise our Web GIS and are the foundation of our geospatial visualisation web toolkit. We have selected PostGIS as our GIS database, as it is fully compliant with OGC SFS, and survey results show that developers prefer it to other open-source solutions. We have selected MapServer as our map server, as it complies with both the WFS and WMS standards. We have selected Pyramid as our web application framework, as we have extensive experience in Python development, and Pyramid is a flexible web framework that is compatible with PostGIS. We have selected OpenLayers as our client application, as it has superior vector-based mapping tools, and allows for the development of highly interactive applications.

3.4.2 Reproducible and Consistent Testing Environment

The performance analysis was performed on Amazon Web Services' (AWS) Elastic Compute Cloud (EC2). The performance analysis was performed on a single EC2 instance type, High-Memory Double Extra Large (m2.2xlarge). This instance is optimised for database use, and has large amounts of random access memory (RAM), 32 GB, allowing us to conduct memory intensive tests. The test environment instance installation process was scripted, remaining the same across all tests. The installation script is available from our public source code repository. See Appendix 8.3 for the source code repository details. All tests were executed multiple times, to ensure that the results were consistent. These measures ensure that the results are comparable, and reproducible.

3.4.3 Input Data Sets

Input data sets were produced in sizes of 10, 100, 1,000, 10,000, 100,000 and 1,000,000 points. The points were generated at random positions across three different areas: Worldwide, Australia wide and Brisbane wide. It should be noted that these names are used for illustrative purposes. The actual bounds used in this experiment are not exact matches for the defined bounds of these geographic regions.

Worldwide data was generated to cover the area between -180 longitude, -90 latitude and +180 longitude, +90 latitude. Figure 22 illustrates the area Worldwide covers.

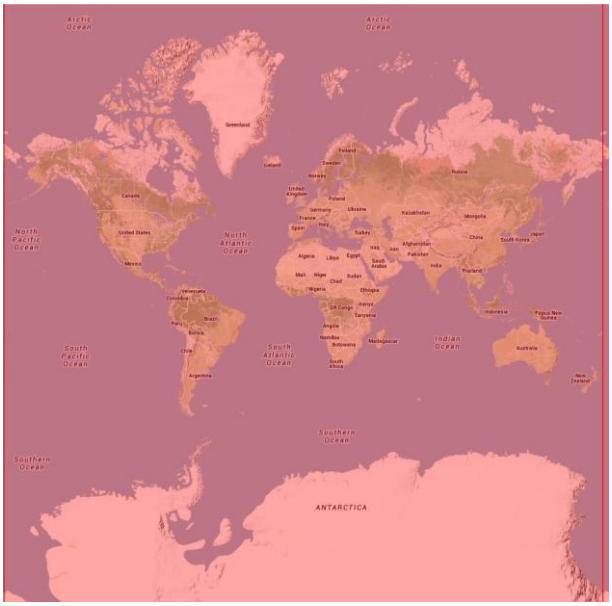

Figure 22: Worldwide bounds [-180, -90, 180, -90] [22]

Australia wide data was generated to cover the area between +113 longitude, -43 latitude and +153 longitude, -10 latitude. Figure 23 illustrates the area Australia wide covers.

Figure 23: Australia wide bounds [113, -43, 153, -10]

Brisbane wide data was generated to cover the area between +151.766402 longitude, -28.141693 latitude and +153.831832 longitude, -26.855379 latitude. Figure 24 illustrates the area Brisbane wide covers.

Figure 24: Brisbane wide bounds [151.766402, -28.141693, 153.831832, -26.855379]

These three areas were selected to allow for comparison of how the different visualisation techniques perform when the points are distributed at varying distances from each other; closer together for Brisbane wide and further apart for Worldwide.

The input data sets were generated as comma separated value (CSV) files. All input data sets were generated using a script. The script was configured with the number of points to generate, and the bounds to generate the points within. Figure 25 contains the pseudo-code description of this script.

```
number_of_points =  100         # the number of points to generate
min_longitude    = -180.0       # the minimum longitude value to generate
min_latitude     = -90.0        # the minimum latitude  value to generate
max_longitude    =  180.0       # the maximum longitude value to generate
max_latitude     =  90.0        # the maximum latitude  value to generate

loop number_of_points times:

  longitude = random_number_in_range(min_longitude .. max_longitude)
  latitude  = random_number_in_range(min_latitude  .. max_latitude)

  print "<longitude>,<latitude>"

end_loop
```

Figure 25: Input Data Generation Script - Pseudo Code

These input data sets are very large, exceeding 30 megabytes for 1 million points. Due to the size of these input data sets, they have not been attached as appendices to this report. All

input data sets used have been stored in our public source code repository. See Appendix 8.3 for the source code repository details.

3.4.4 Test Scenario

In order to test how the different visualisation techniques performed across the different input data set sizes and areas, we defined a common user interaction that could be automated. This interaction involves a user starting at a Worldwide zoom level, and zooming in to a single block within Brisbane. The interaction steps and their associated bounding boxes are defined in Table 1 and are illustrated in Figure 26 through Figure 33. In the figures, the shaded area is the client's bounding box, and represents the user's viewable area, the area of the map they would be able to see.

Interaction Step	Client Bounding Box [West, South, East, North]
Step 1	[-180, -90, 180, 90]
Step 2	[113, -43, 153, -10]
Step 3	[137.834, -10.7, 153.513336, -28.17]
Step 4	[151.766402, -28.141693, 153.831832, -26.855379]
Step 5	[152.557418, -27.738907, 153.266036, -27.271171]
Step 6	[152.913787, -27.679941, 153.077209, -27.586867]
Step 7	[153.051803, -27.611664, 153.078582, -27.595387]
Step 8	[153.077359, -27.597061, 153.080556, -27.596623]

Table 1: Interaction steps and their associated bounding boxes

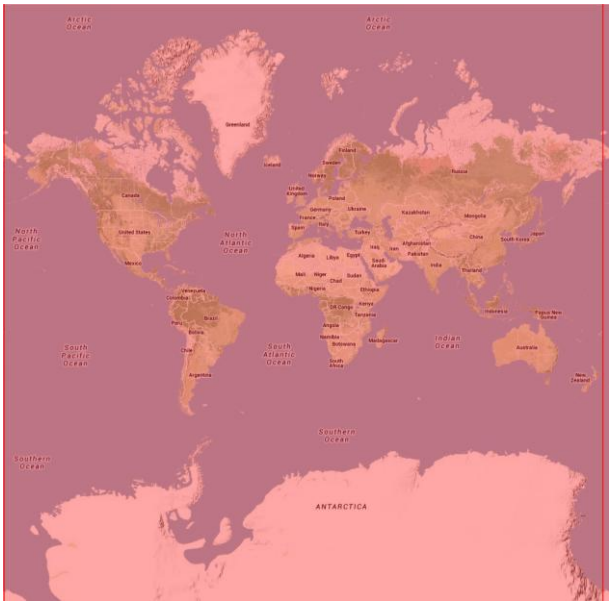

Figure 26: Interaction step 1 (Worldwide) [-180, -90, 180, -90]

Figure 27: Interaction step 2 (Australia) [113, -43, 153, -10]

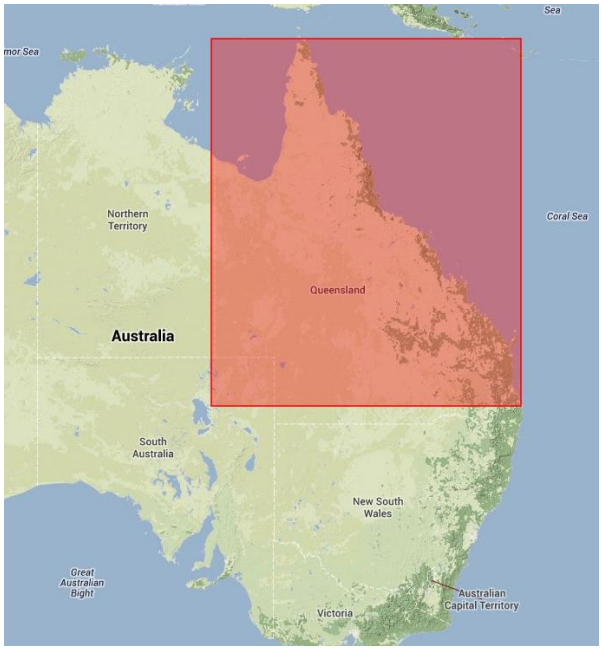

Figure 28: Interaction step 3 (Queensland) [137.834, -10.7, 153.513336, -28.17]

Figure 29: Interaction step 4 (Brisbane – zoomed out) [151.766402, -28.141693, 153.831832, -26.855379]

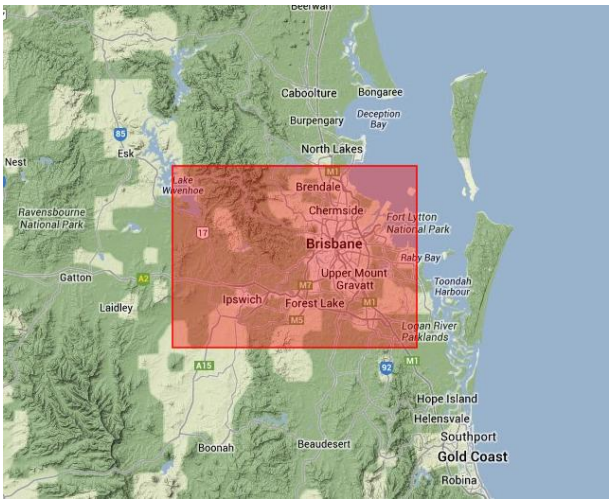

Figure 30: Interaction step 5 (Brisbane) [152.557418, -27.738907, 153.266036, -27.271171]

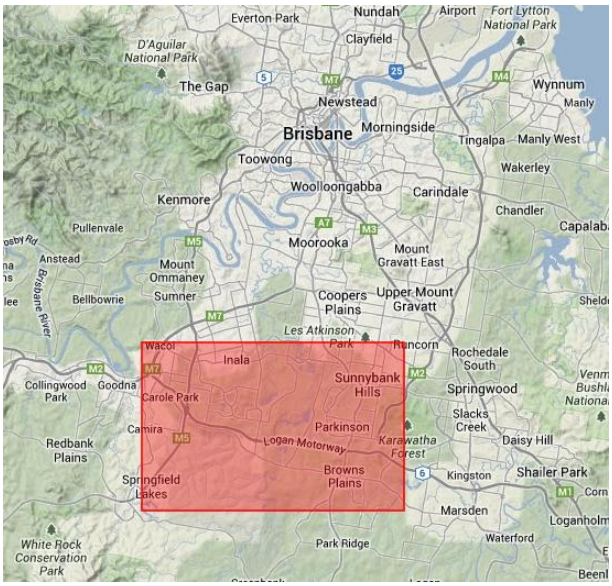

Figure 31: Interaction step 6 (Brisbane - suburb) [152.913787, -27.679941, 153.077209, -27.586867]

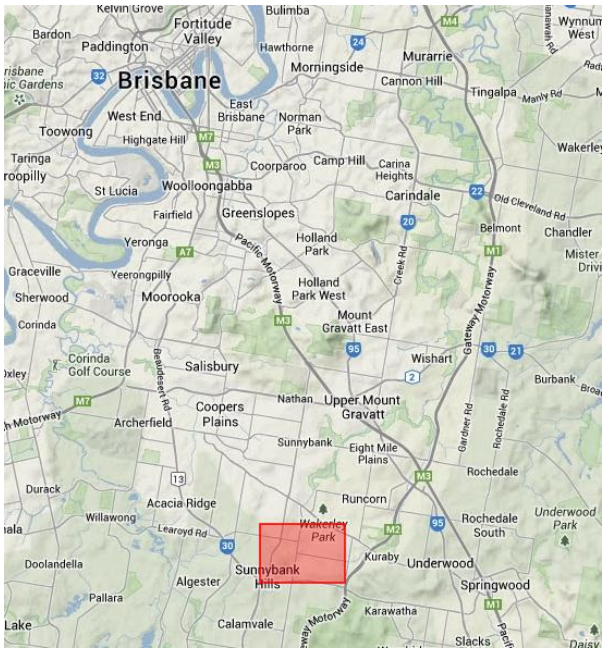

Figure 32: Interaction step 7 (Brisbane – a few blocks) [153.051803, -27.611664, 153.078582, -27.595387]

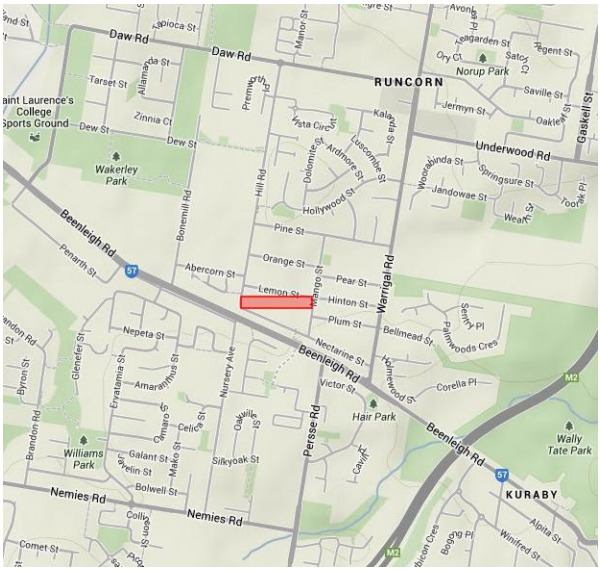

Figure 33: Interaction step 8 (Brisbane – a single block) [153.077359, -27.597061, 153.080556, -27.596623]

At each step of the user interaction, the client will request the geospatial data, providing the defined bounding box as an input to the geospatial visualisation technique. The visualisation technique can then optionally use the bounding box in its processing of the request.

4 Summarised Results

This section of the report will only provide a summary of the results. The full results are available in appendix 8.2 - Results. The summarised results will focus on the input data sets: 10,000 points, 100,000 points and 1,000,000 points. Unless otherwise specified, all results are plotted with a logarithmic vertical axis.

The following abbreviations shall be used extensively in this section of the report.

- M → Mappable Point
- B → Bound Mappable Point
- G → Gridded Mappable Point
- GB → Gridded and Bound Mappable Point
- CGB → Cached, Gridded and Bound Mappable Point

4.1 Worldwide

The following results were produced with the worldwide input data set. The worldwide input data set consists of 10, 100, 1,000, 10,000, 100,000 or 1,000,000 points randomly generated between -180 longitude, -90 latitude and +180 longitude, +90 latitude, i.e. across the world.

4.1.1 Pre-Processing

Figure 34 shows how long it took, in seconds, for the visualisation techniques to perform their *pre_process* function.

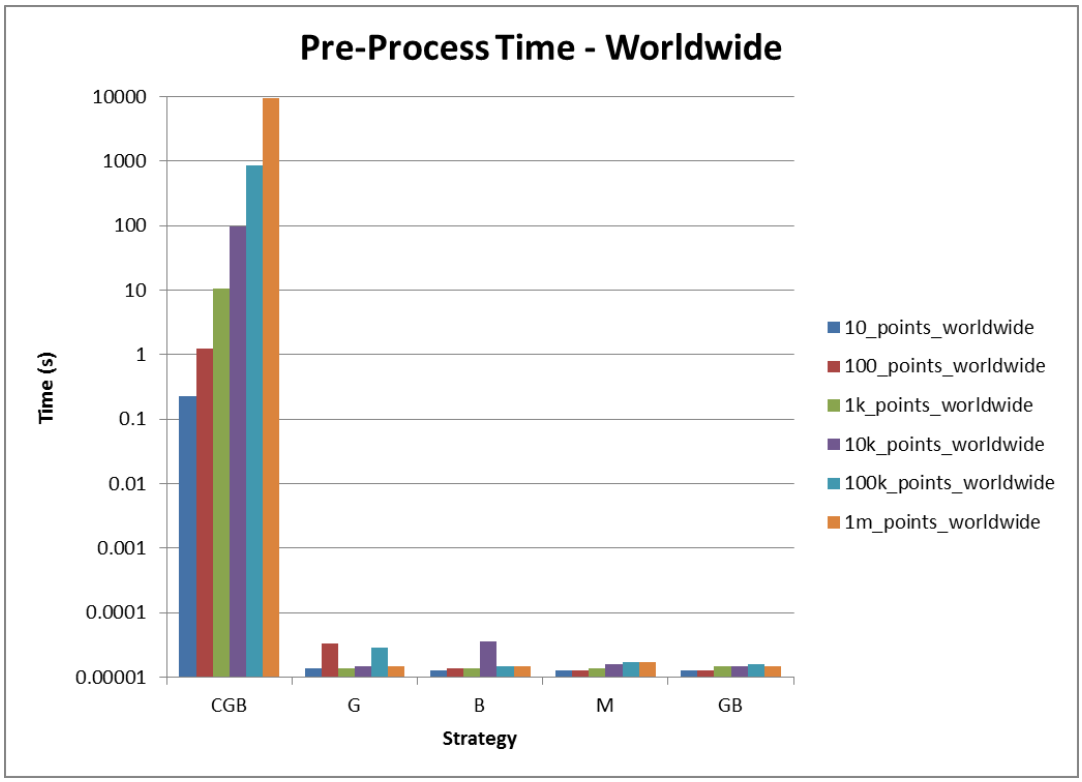

Figure 34: Pre-Processing Time – Worldwide

The CGB technique had a pre-process time of approximately 1 second for every 100 points. This resulted in a pre-process time of approximately 160 minutes for 1,000,000 points. No other technique implemented the pre-process function.

Figure 35 shows, for the CGB technique, the size of the database before and after the *pre_process* function was called.

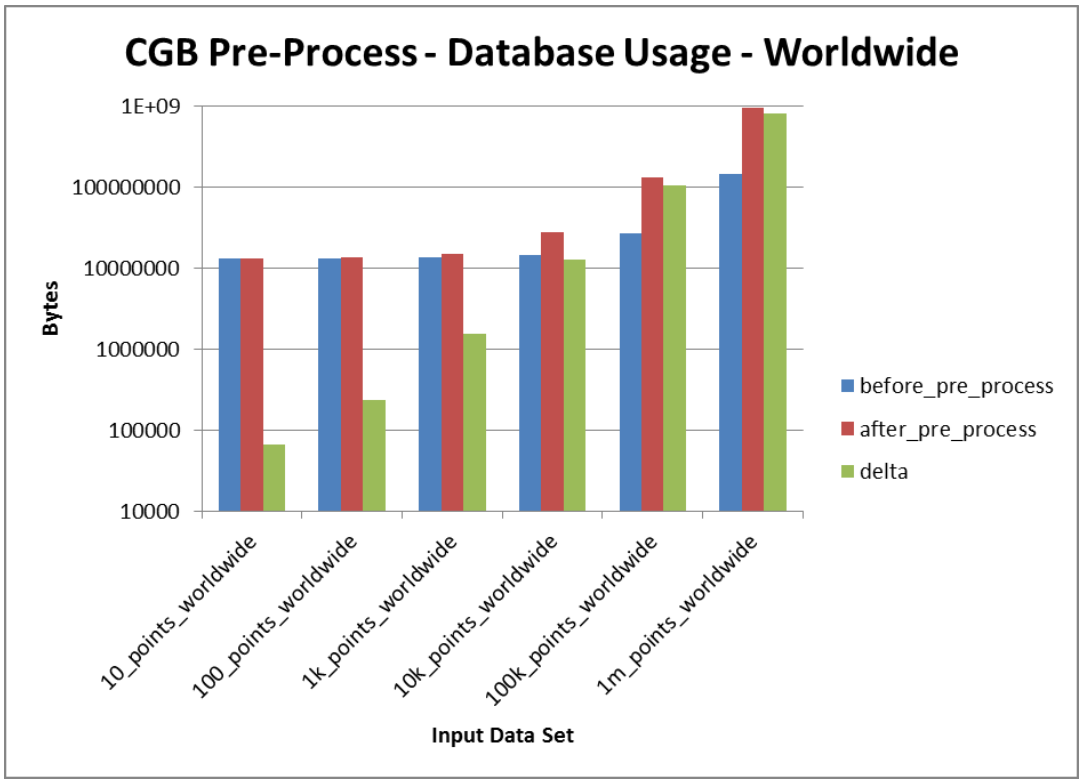

Figure 35 : CGB Pre-Processing Database Usage – Worldwide

For the 1,000,000 point data set, the CGB technique's *pre_process* function increased the database size from 138.2 megabytes to 921.4 megabytes. This is a difference of 783.2 megabytes, which equates to 821 bytes per point. For the 10,000 point data set, the database size increased from 13.7 megabytes to 26 megabytes. This is a difference of 12.3 megabytes, which equates to 1293 bytes per point.

4.1.2 Clusters

Figure 36 shows the number of clusters produced by the visualisation techniques for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 3 of the appendix.

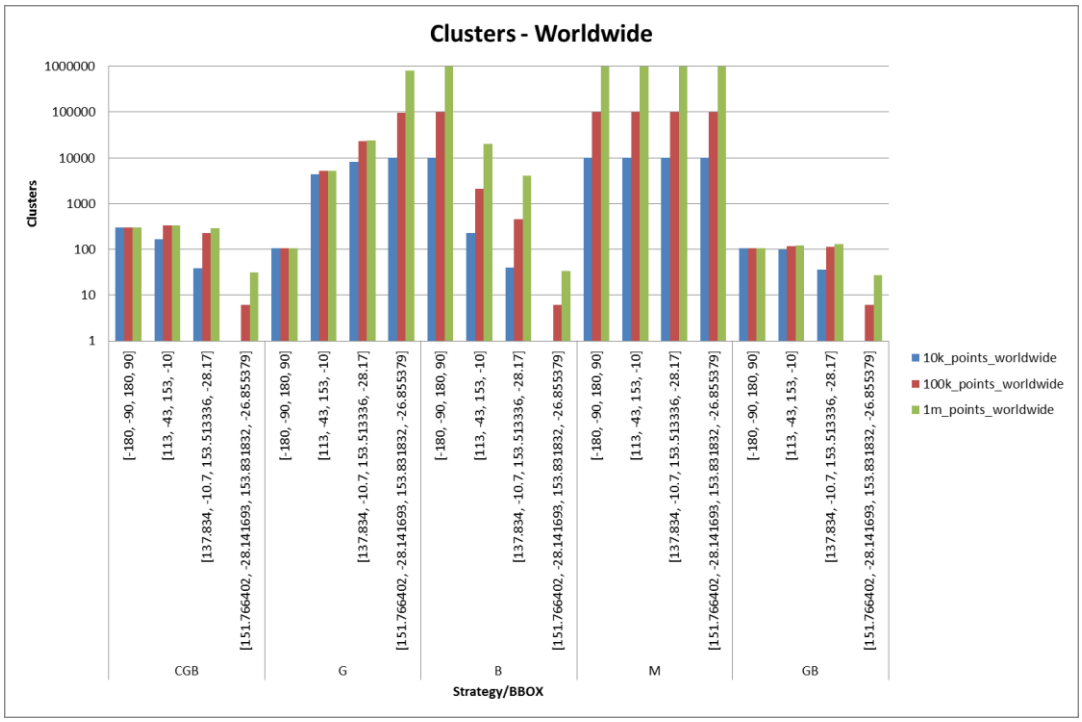

Figure 36: Clusters Generated – Worldwide; Summarised

The CGB technique had a maximum of 327 clusters; the GB technique had a maximum of 129 clusters; and the M, G, and B techniques all had a maximum of 1,000,000 clusters.

4.1.3 GeoJSON

Figure 37 shows how long it took, in seconds, to return from the *get_points_as_geojson* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 4 of the appendix.

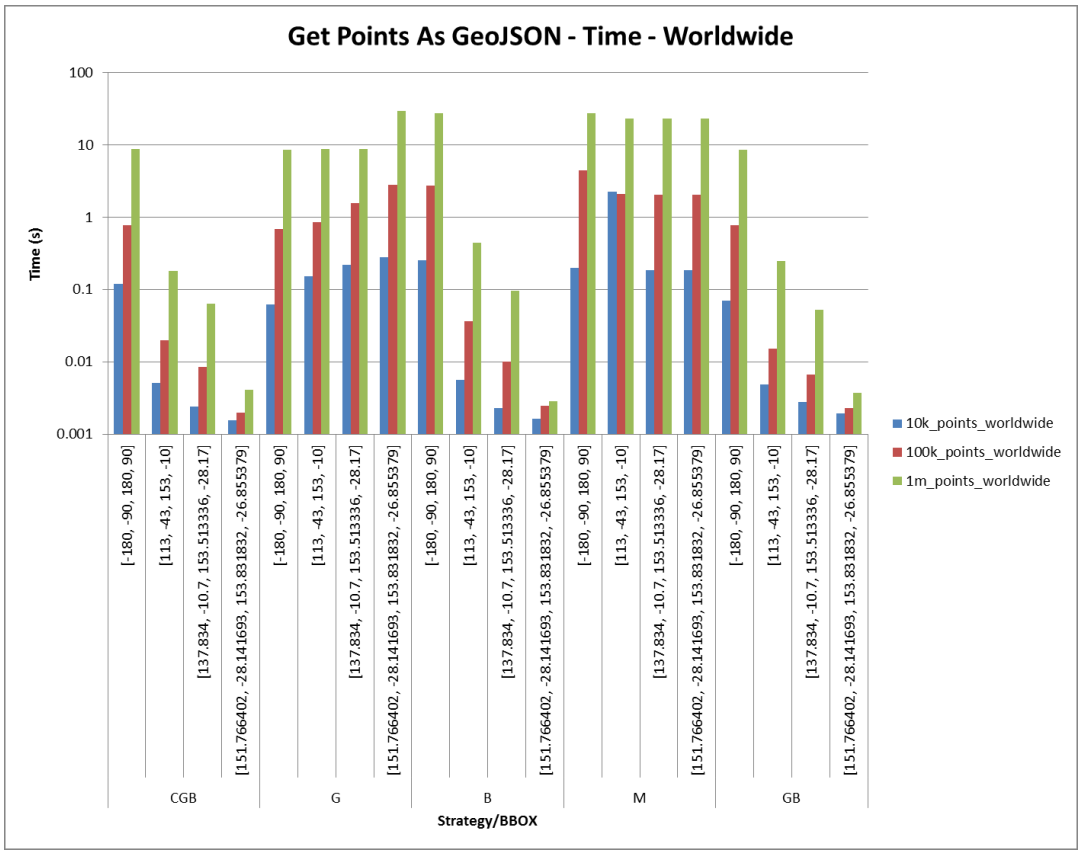

Figure 37: Get Points as GeoJSON – Time (s) – Worldwide; Summarised

The CGB, and GB techniques both had a maximum time of less than 10 seconds; and the M, G and B techniques all had a maximum time exceeding 25 seconds.

Figure 38 shows how long it took, in seconds, to return from the *get_points_as_geojson_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 5 of the appendix.

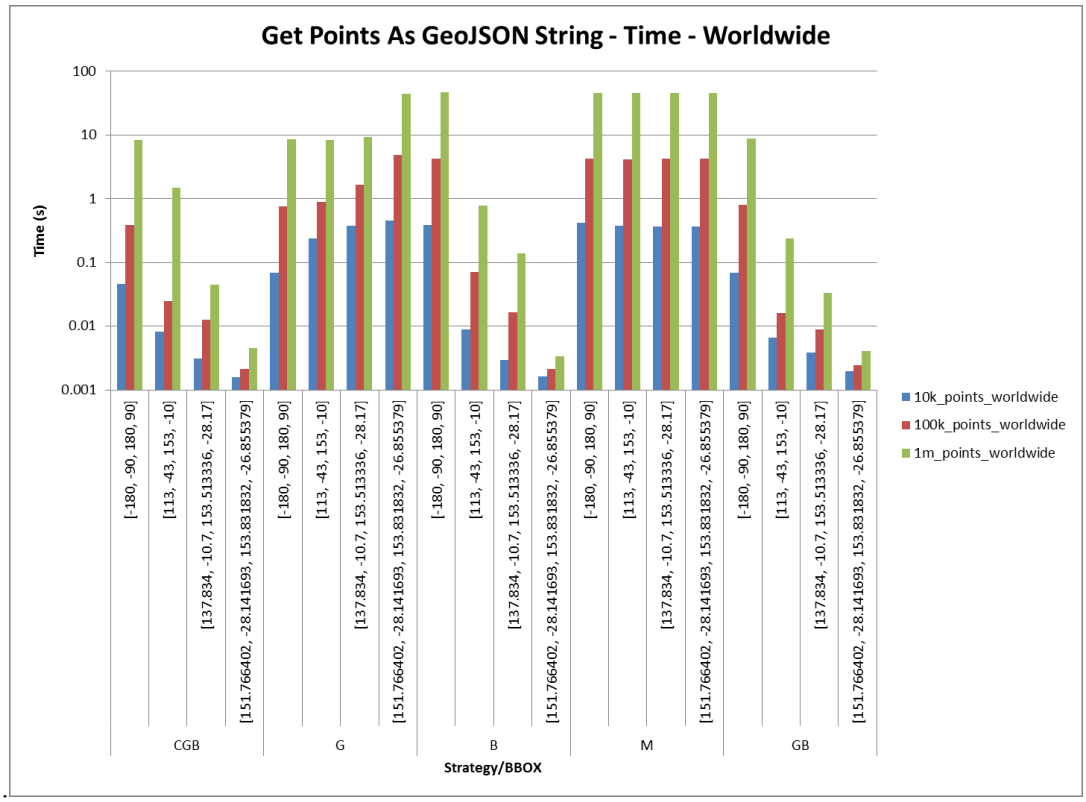

Figure 38: Get Points as GeoJSON String – Time (s) – Worldwide; Summarised

The CGB, and GB techniques both had a maximum time of less than 10 seconds; and the M, G and B techniques all had a maximum time exceeding 40 seconds.

Figure 39 shows how long the GeoJSON string returned from the *get_points_as_geojson_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 6 of the appendix.

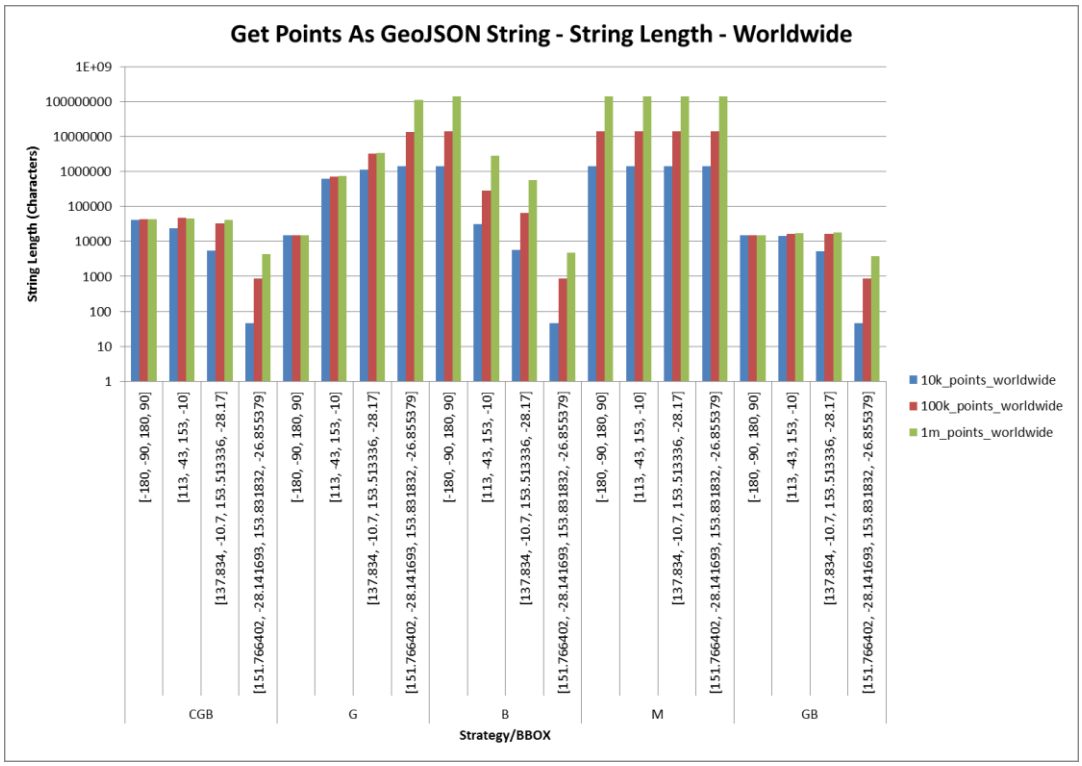

Figure 39: Get Points as GeoJSON String – String Length (Characters) – Worldwide; Summarised

The M, G and B techniques all had a maximum string size of 140,796,255 characters, or 134 megabytes; the CGB technique had a maximum string size of 46,401 characters, or 45.3 kilobytes; and the GB technique had a maximum string size of 18,322 characters, or 17.9 kilobytes.

4.1.4 WKT

Figure 40 shows how long it took, in seconds, to return from the *get_points_as_wkt* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 7 of the appendix.

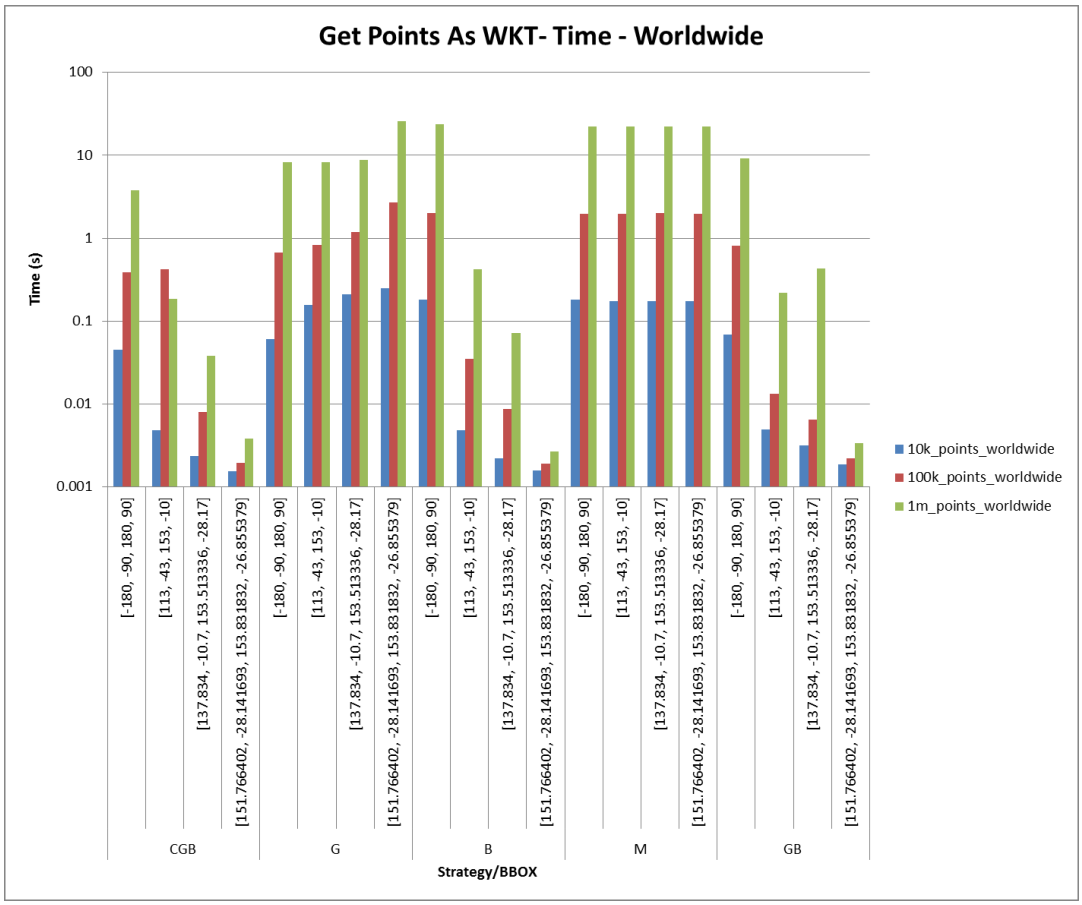

Figure 40: Get Points as WKT – Time (s) – Worldwide; Summarised

The CGB technique had a maximum time of 3.8 seconds; the GB technique had a maximum time of 9.2 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 41 shows how long it took, in seconds, to return from the *get_points_as_wkt_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 8 of the appendix.

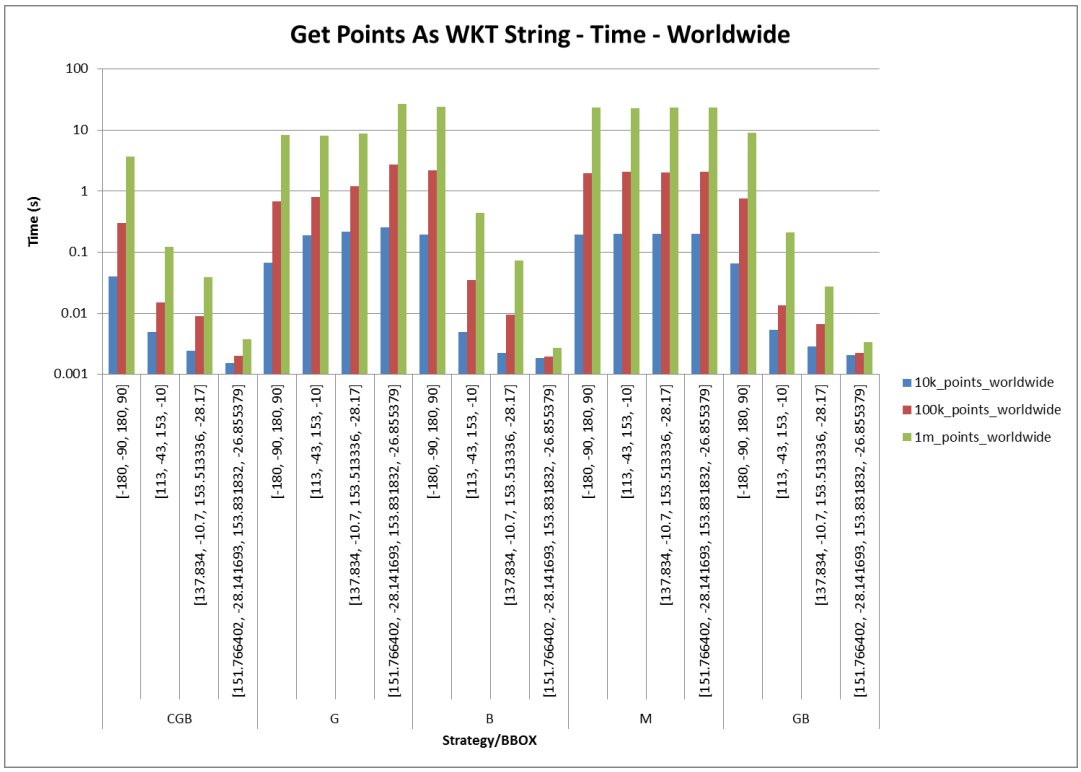

Figure 41: Get Points as WKT String – Time (s) – Worldwide; Summarised

The CGB technique had a maximum time of 3.7 seconds; the GB technique had a maximum time of 8.9 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 42 shows how long the WKT string returned from the *get_points_as_wkt_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 9 of the appendix.

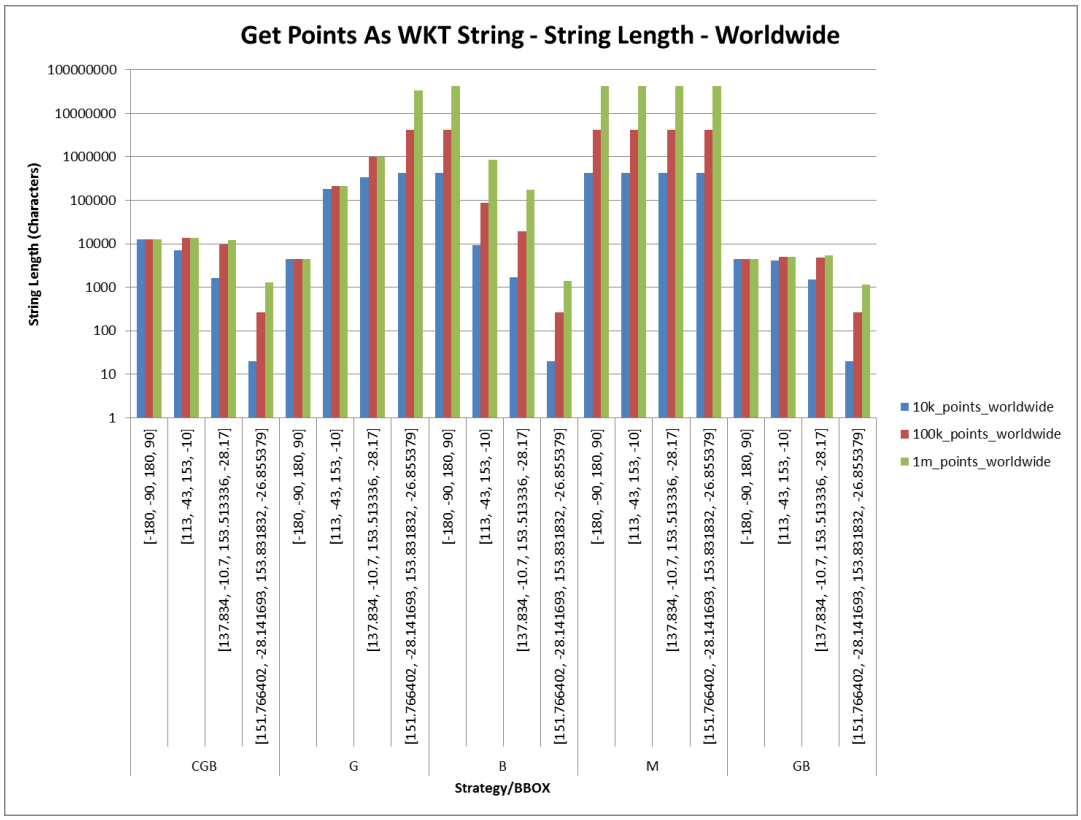

Figure 42: Get Points as WKT String – String Length (Characters) – Worldwide; Summarised

The M, G and B technique all had a maximum string size of 41,797,983 characters, or 39.9 megabytes; the CGB technique had a maximum string size of 13,811 characters, or 13.5 kilobytes; and the GB technique had a maximum string size of 5,416 characters, or 5.3 kilobytes.

4.2 Australia Wide

The following results were produced with the Australia wide input data set. The Australia wide input data set consists of 10, 100, 1,000, 10,000, 100,000 or 1,000,000 points randomly generated between +113 longitude, -43 latitude and +153 longitude, -10 latitude, i.e. approximately across Australia.

4.2.1 Pre-Processing

Figure 43 shows how long it took, in seconds, for the visualisation techniques to perform their *pre_process* function.

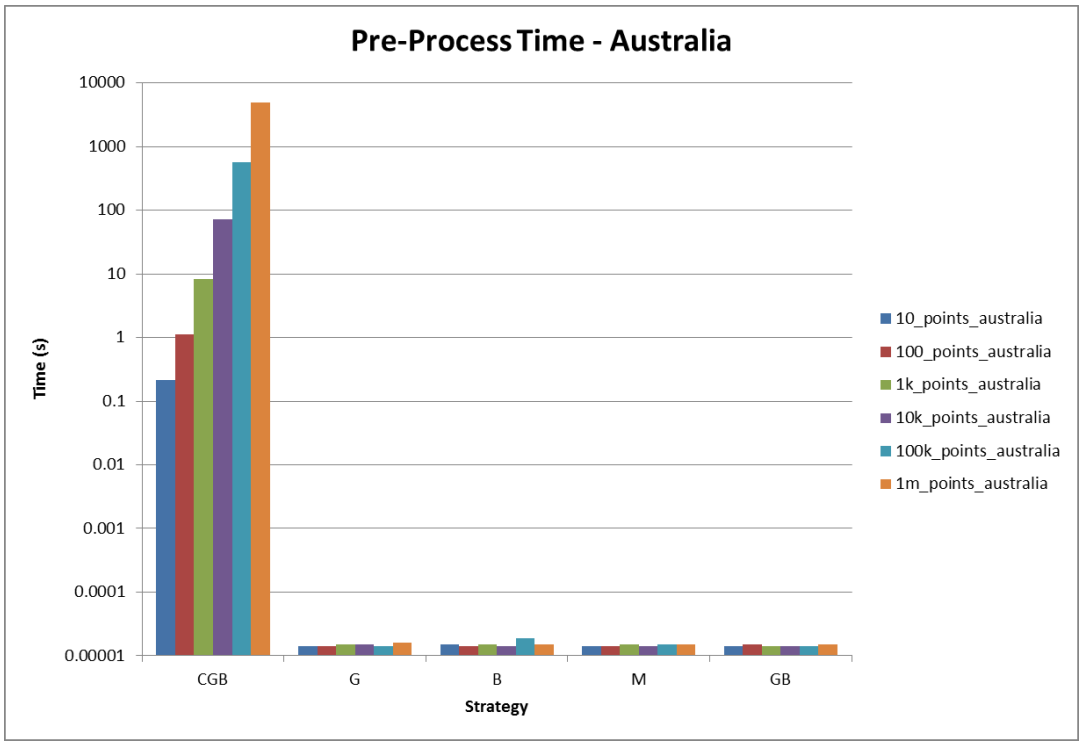

Figure 43: Pre-Processing Time – Australia wide

The CGB technique had a pre-process time of 82 minutes for 1,000,000 points, which equates to approximately 200 points every second. For the 100,000 point data set, approximately 180 points were processed every second. This dropped to approximately 141 points every second for the 10,000 point data set.

Figure 44 shows, for the CGB technique, the size of the database before and after the *pre_process* function was called.

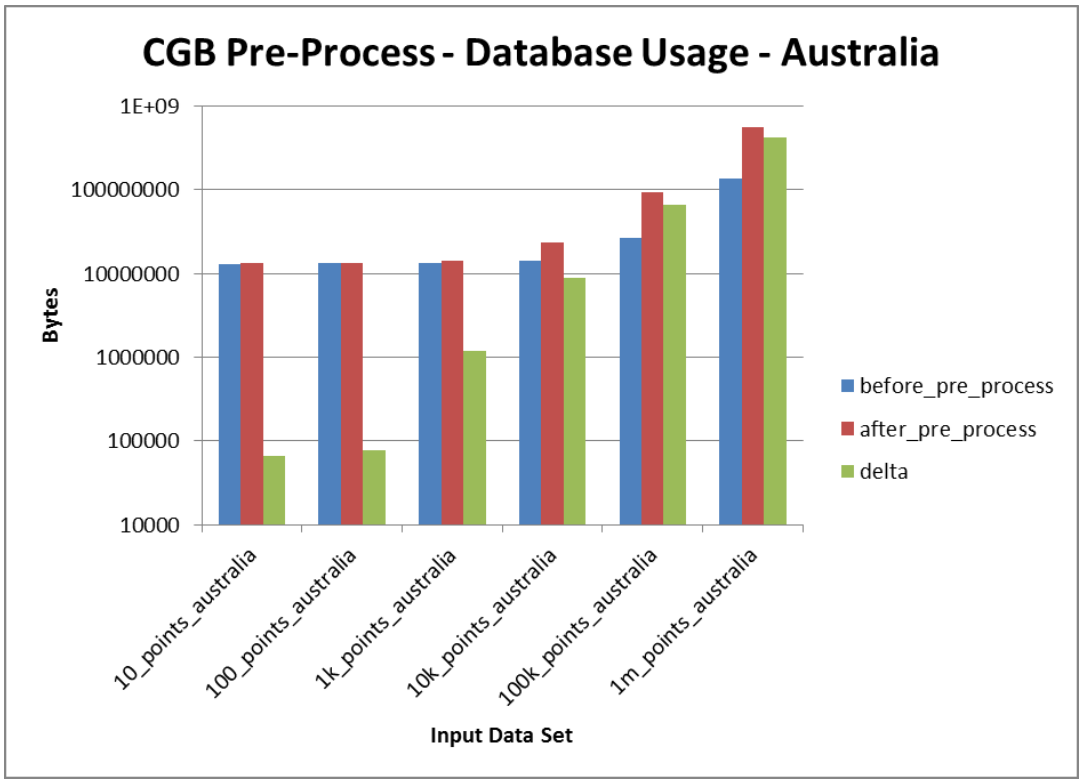

Figure 44: CGB Pre-Processing Database Usage – Australia wide

For the 1,000,000 point data set, the CGB technique's *pre_process* function increased the database size from 131.38 megabytes to 539.3 megabytes. This is a difference of 407.9 megabytes, which equates to 427 bytes per point. For the 10,000 point data set, the database size increased from 13.8 megabytes to 22 megabytes. This is a difference of 8.36 megabytes, which equates to 877 bytes per point.

4.2.2 Clusters

Figure 45 shows how many clusters the visualisation techniques produced for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 11 of the appendix.

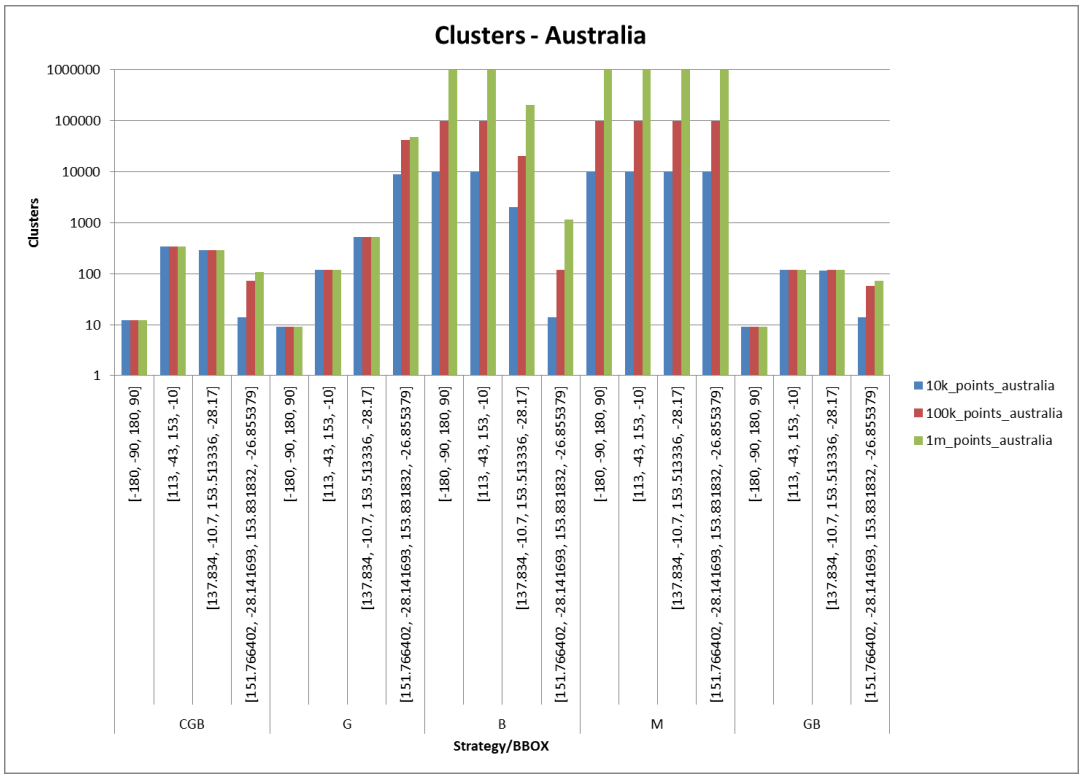

Figure 45: Clusters Generated – Australia wide; Summarised

The CGB technique had a maximum number of 340 clusters; the GB technique had a maximum of 120 clusters; and the M, G, and B techniques all had a maximum of 1,000,000 clusters.

4.2.3 GeoJSON

Figure 46 shows how long it took, in seconds, to return from the *get_points_as_geojson* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 12 of the appendix.

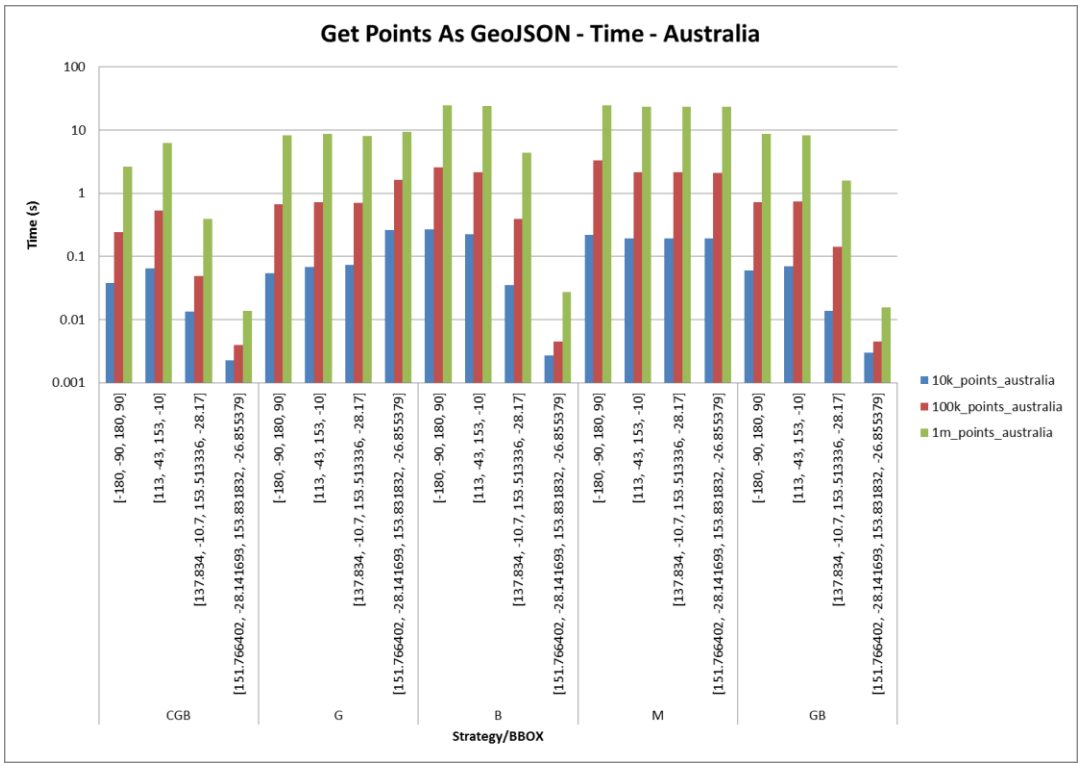

Figure 46: Get Points as GeoJSON – Time (s) – Australia wide; Summarised

The CGB, and GB techniques both had a maximum time of less than 10 seconds; and the M, G and B techniques all had a maximum time exceeding 24 seconds.

Figure 47 shows how long it took, in seconds, to return from the *get_points_as_geojson_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 13 of the appendix.

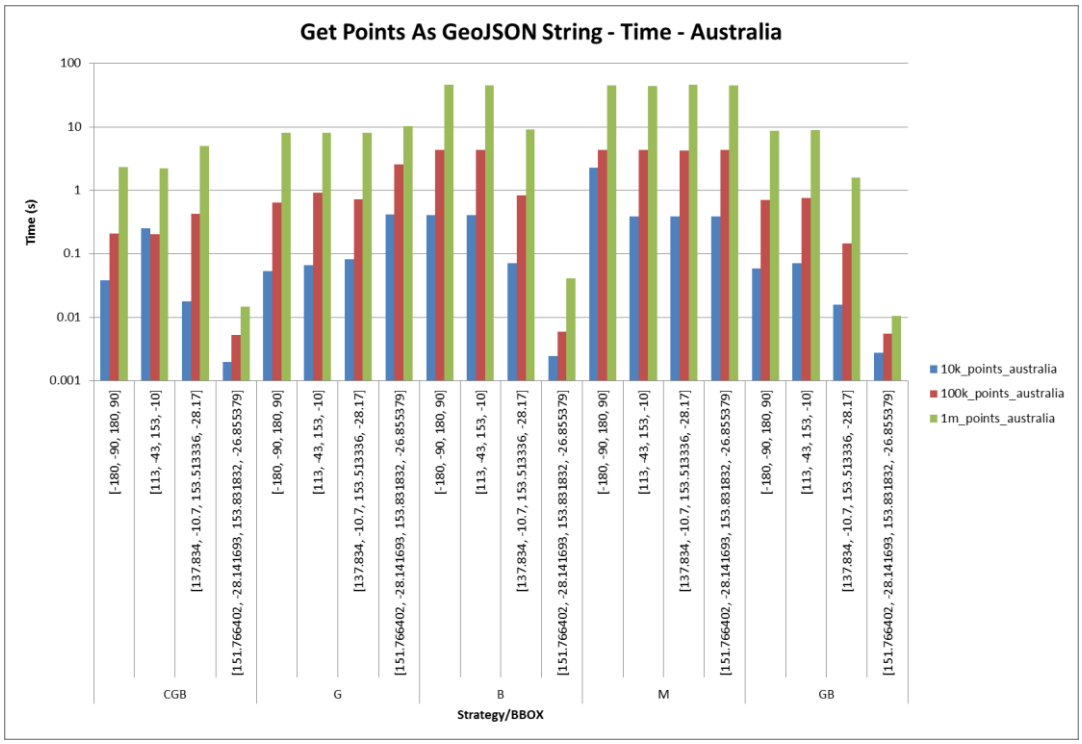

Figure 47: Get Points as GeoJSON String – Time (s) – Australia wide; Summarised

The CGB, and GB techniques both had a maximum time of less than 10 seconds; and the M, G and B techniques all had a maximum time exceeding 45 seconds.

Figure 48 shows how long the GeoJSON string returned from the *get_points_as_geojson_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 14 of the appendix.

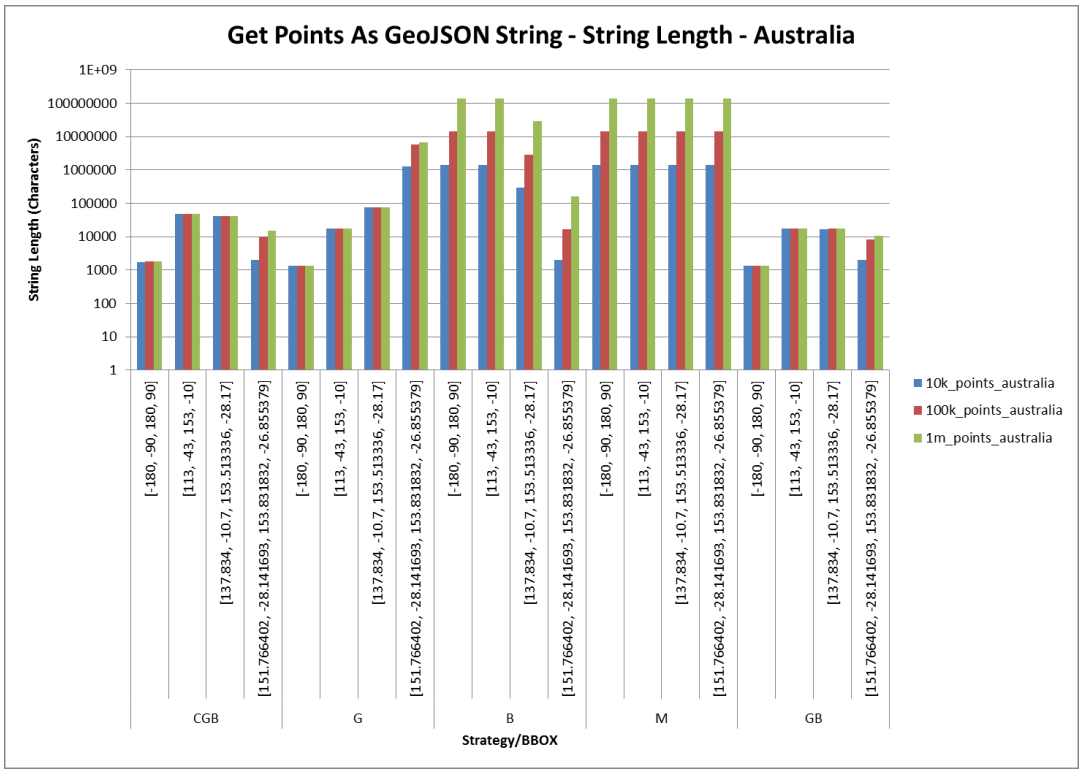

Figure 48: Get Points as GeoJSON String – String Length (Characters) – Australia wide; Summarised

The M, G and B techniques all had a maximum string size of 140,777,390 characters, or 134 megabytes; the CGB technique had a maximum string size of 48,924 characters, or 47.8 kilobytes; and the GB technique had a maximum string size of 17,300 characters, or 16.9 kilobytes.

4.2.4 WKT

Figure 49 shows how long it took, in seconds, to return from the *get_points_as_wkt* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 15 of the appendix.

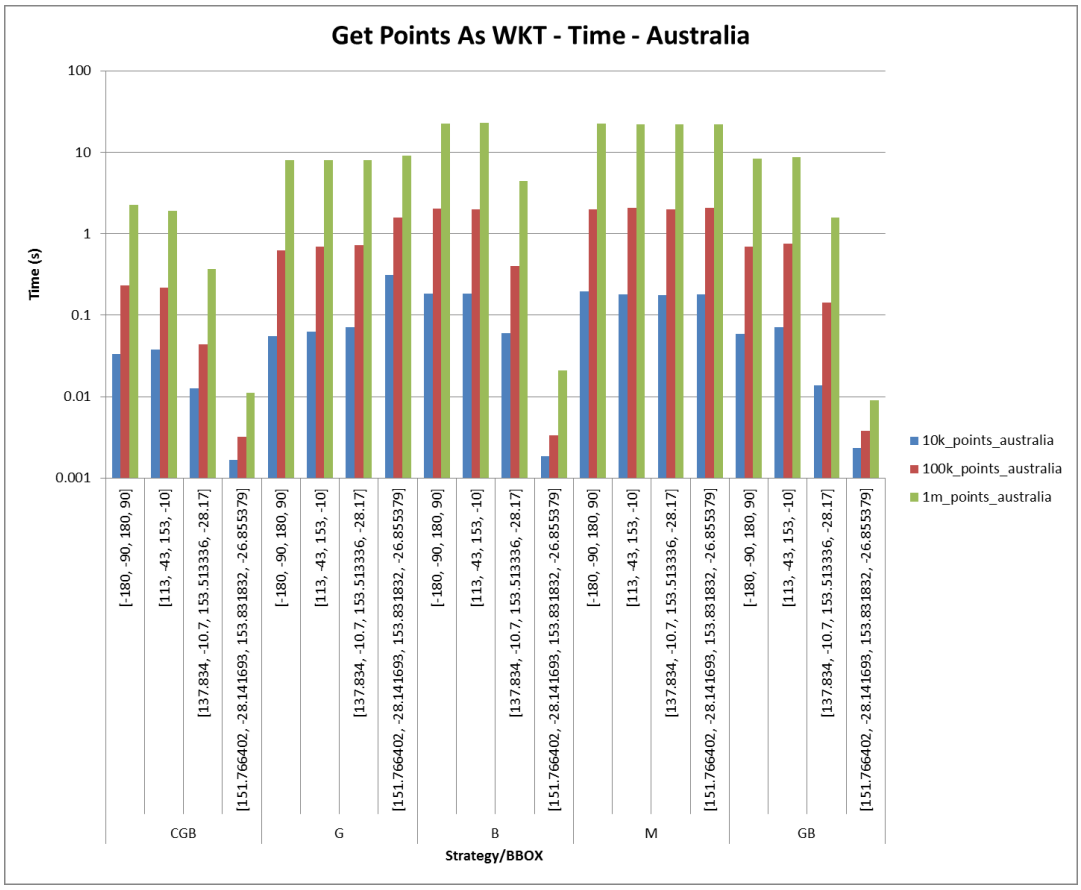

Figure 49: Get Points as WKT – Time (s) – Australia wide; Summarised

The CGB technique had a maximum time of 2.3 seconds; the GB technique had a maximum time of 8.7 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 50 shows how long it took, in seconds, to return from the *get_points_as_wkt_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 16 of the appendix.

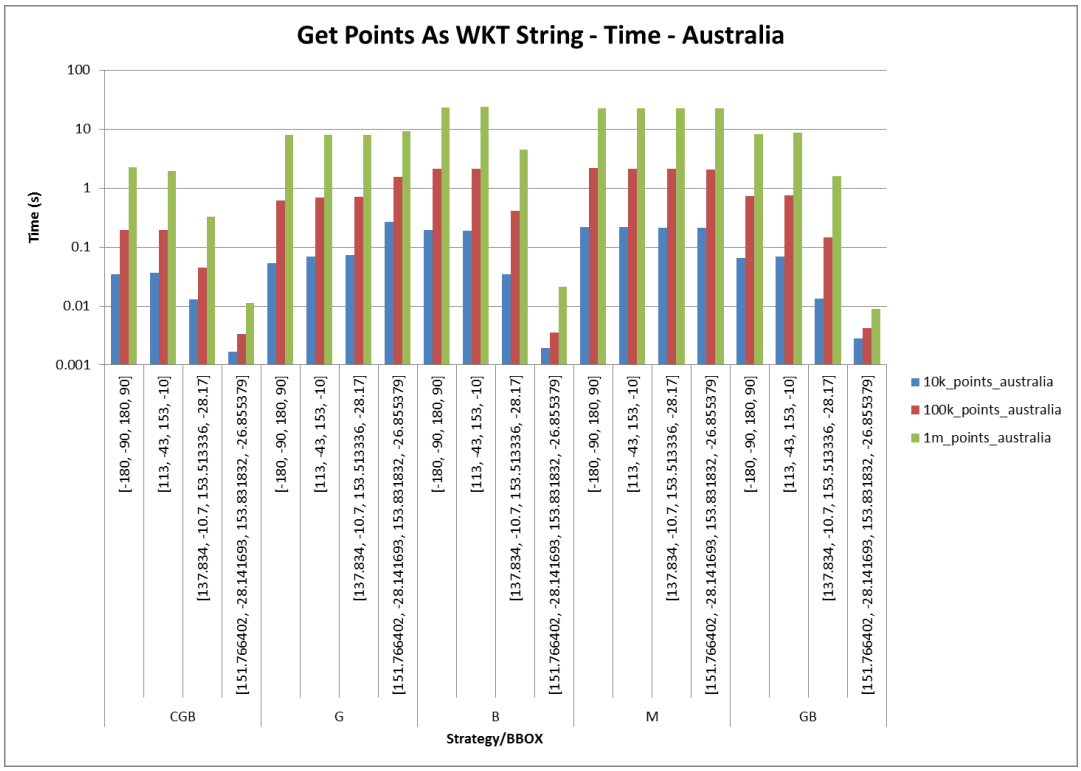

Figure 50: Get Points as WKT String – Time (s) – Australia wide; Summarised

The CGB technique had a maximum time of 2.3 seconds; the GB technique had a maximum time of 8.7 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 51 shows how long the WKT string returned from the *get_points_as_wkt_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 17 of the appendix.

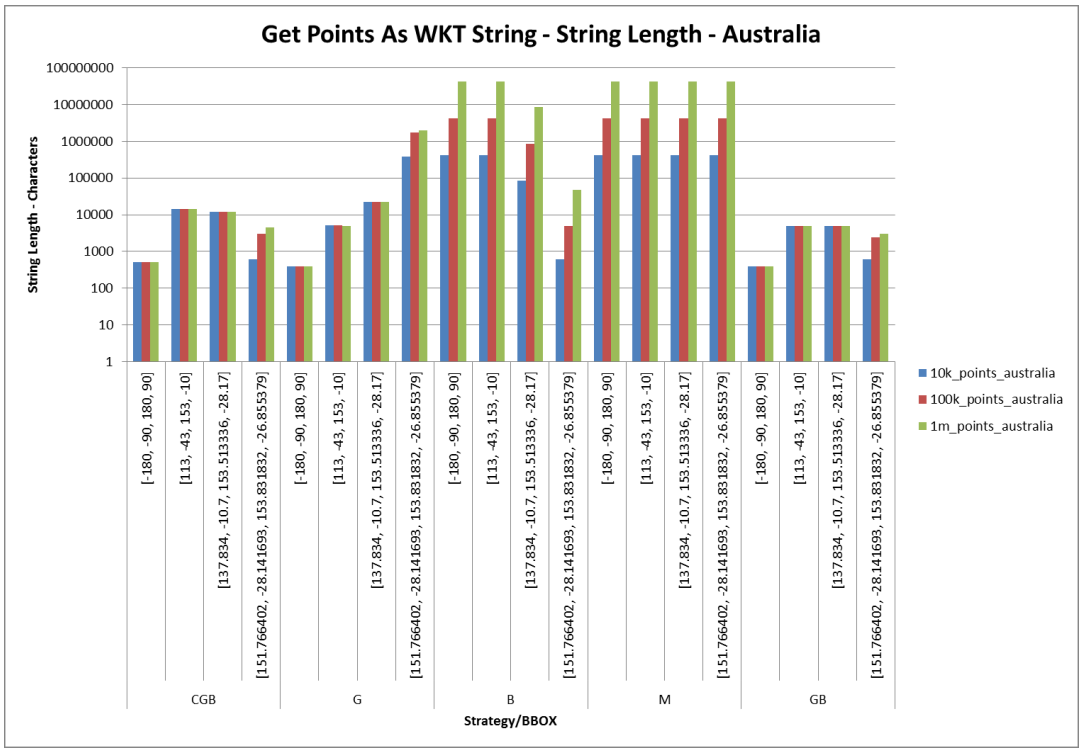

Figure 51: Get Points as WKT String – String Length (Characters) – Australia Wide; Summarised

The M, G and B technique all had a maximum string size of 41,777,366 characters, or 39.9 megabytes; the CGB technique had a maximum string size of 14,220 characters, or 13.9 kilobytes; and the GB technique had a maximum string size of 5,042 characters, or 4.9 kilobytes.

4.3 Brisbane Wide

The following results were produced with the Brisbane wide input data set. The Brisbane wide input data set consists of 10, 100, 1,000, 10,000, 100,000 or 1,000,000 points randomly generated between +151.766402 longitude, -28.141693 latitude and +153.831832 longitude, -26.855379 latitude.

4.3.1 Pre-Processing

Figure 52 shows how long it took, in seconds, for the visualisation techniques to perform their *pre_process* function.

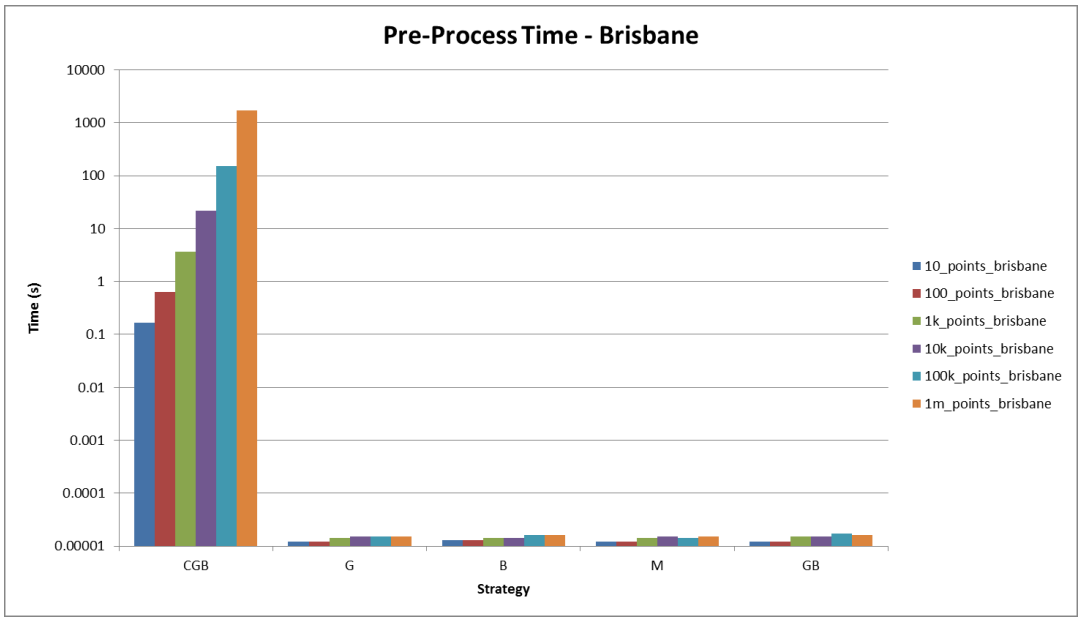

Figure 52: Pre-Processing Time – Brisbane wide

The CGB technique had a pre-process time of 29 minutes for 1,000,000 points, which equates to approximately 578 points every second. For the 100,000 point data set, approximately 651 points were processed every second. This dropped to approximately 460 points every second for the 10,000 point data set. This drops further to approximately 275 points every second for the 1,000 point data set.

Figure 53 shows, for the CGB technique, the size of the database before and after the *pre_process* function was called.

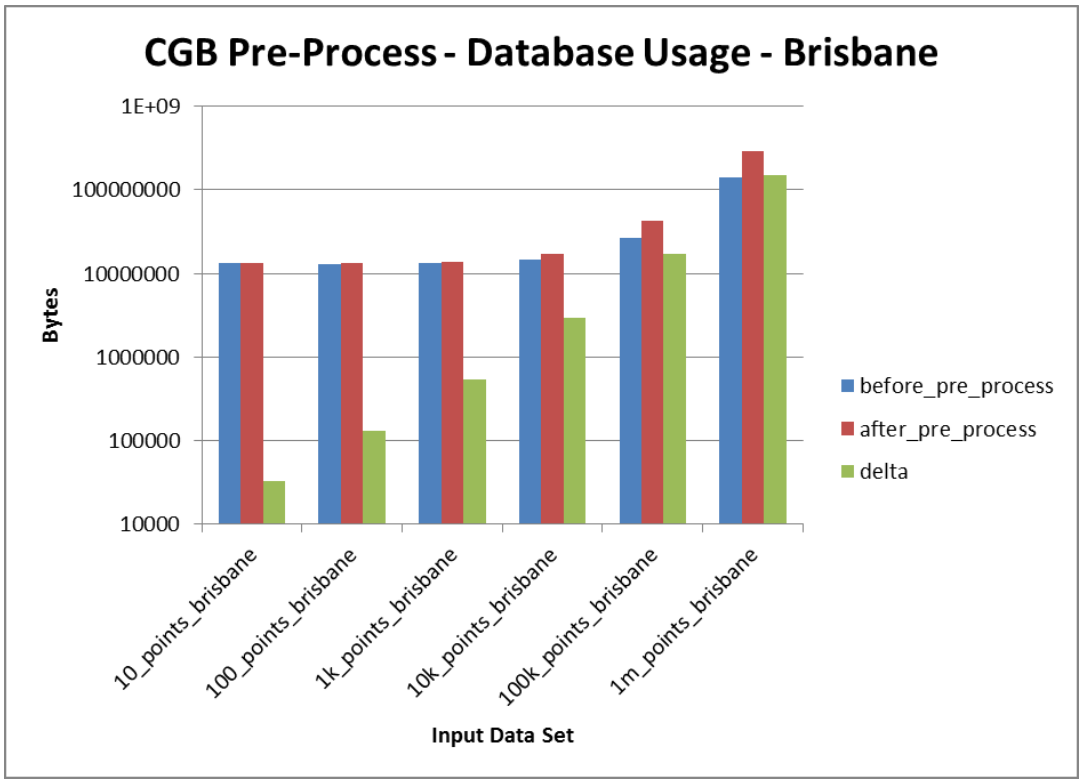

Figure 53 : CGB Pre-Processing Database Usage – Brisbane wide

For the 1,000,000 point data set, the CGB technique's *pre_process* function increased the database size from 135 megabytes to 276 megabytes. This is a difference of 141.3 megabytes, which equates to 148 bytes per point. For the 10,000 point data set, the database size increased from 13.8 megabytes to 16.53 megabytes. This is a difference of 2.73 megabytes, which equates to 291 bytes per point.

4.3.2 Clusters

Figure 54 shows how many clusters the visualisation techniques produced for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 19 of the appendix.

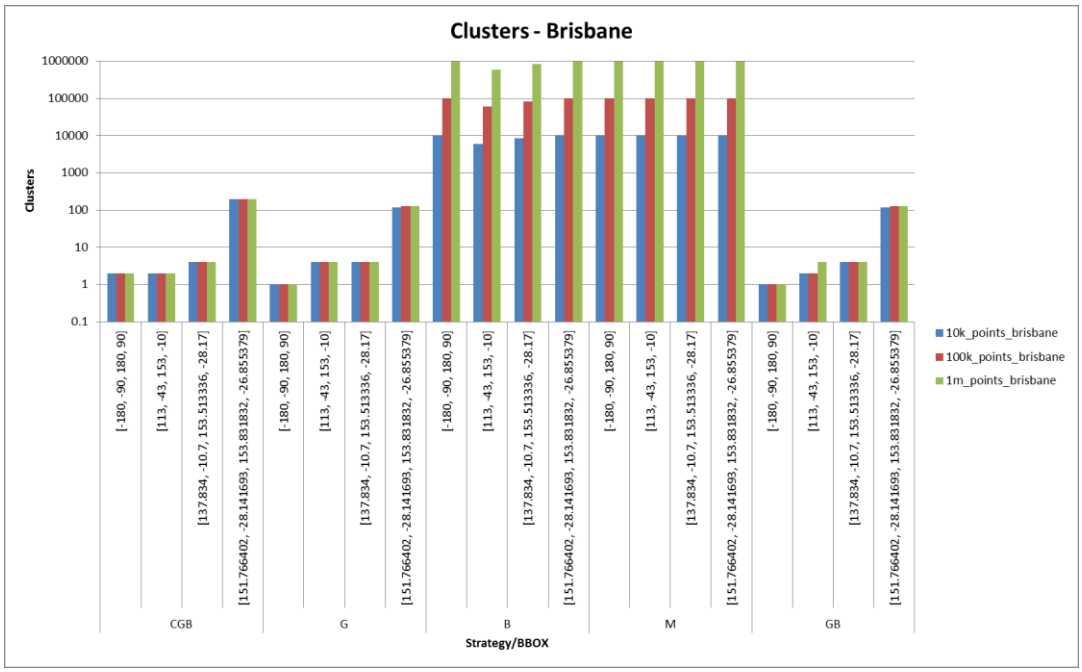

Figure 54: Clusters Generated – Brisbane wide; Summarised

The CGB and GB techniques had a maximum number of 5,791 clusters; and the M, G, and B techniques all had a maximum of 999,999 clusters.

4.3.3 GeoJSON

Figure 55 shows how long it took, in seconds, to return from the *get_points_as_geojson* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 20 of the appendix.

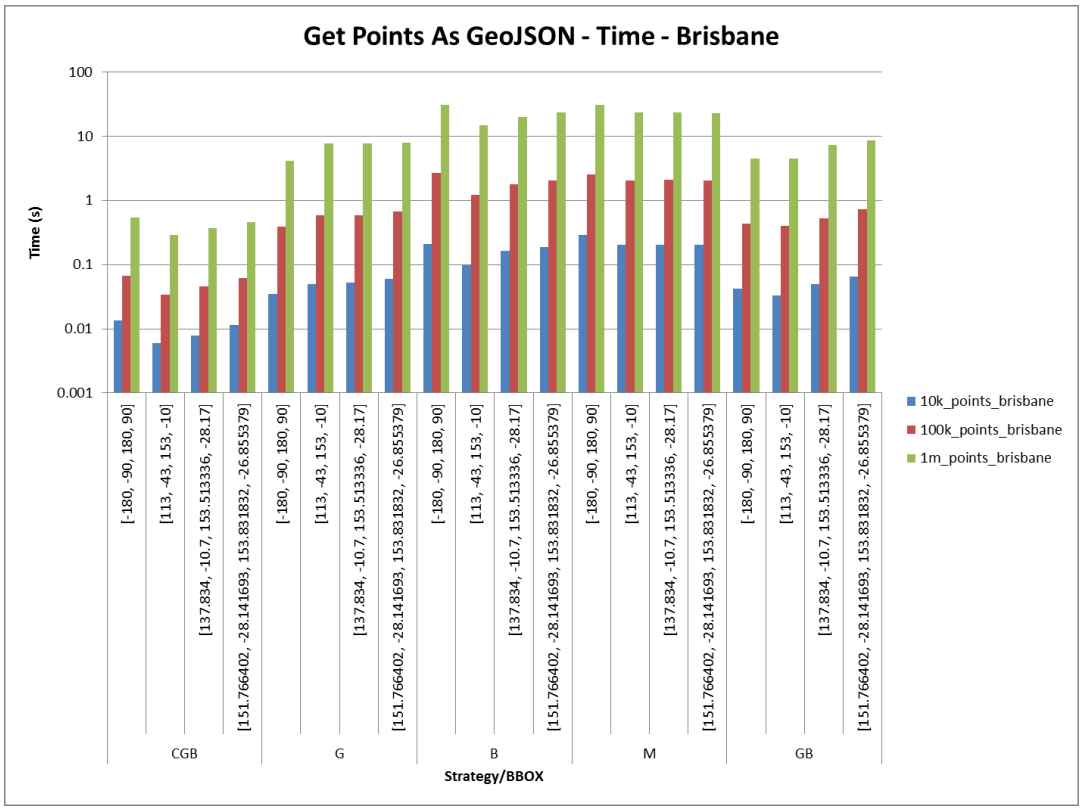

Figure 55: Get Points as GeoJSON – Time (s) – Brisbane wide; Summarised

The CGB technique had a maximum time of 0.54 seconds; the GB technique had a maximum time of 8.54 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 56 shows how long it took, in seconds, to return from the *get_points_as_geojson_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 21 of the appendix.

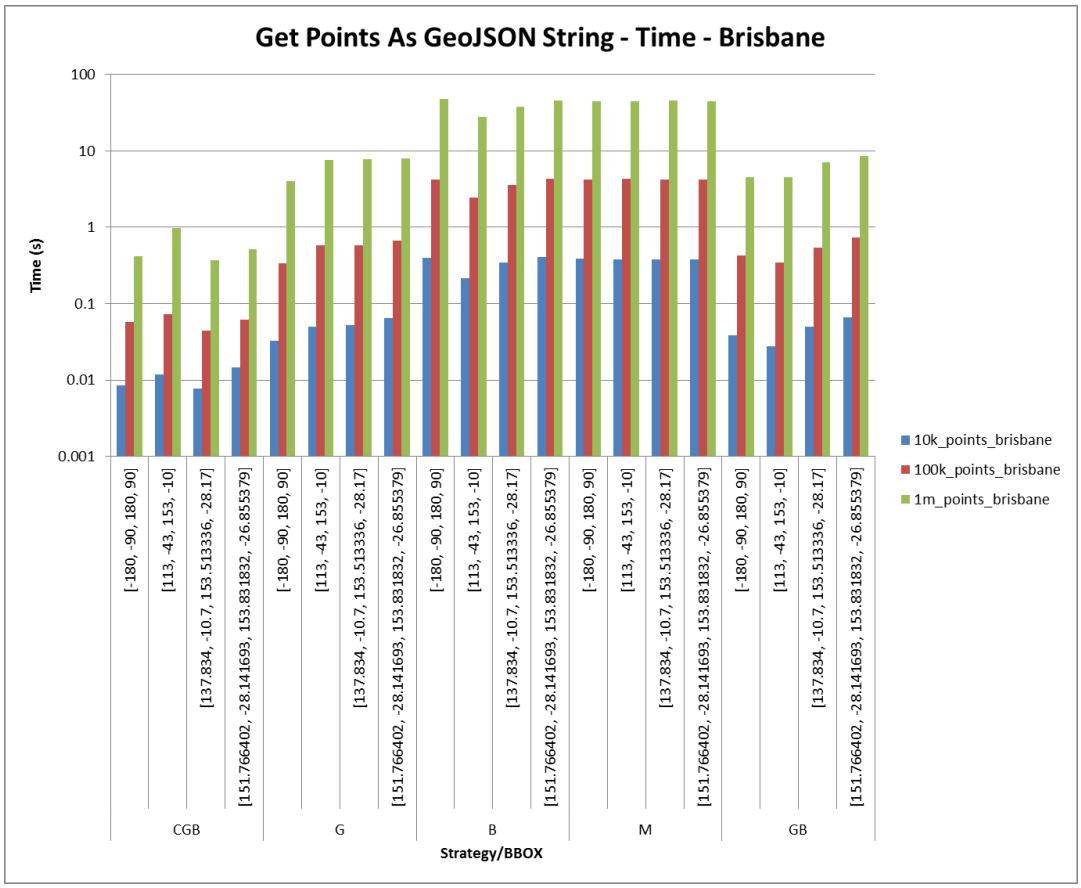

Figure 56: Get Points as GeoJSON String – Time (s) – Brisbane wide; Summarised

The CGB technique had a maximum time of 0.99 seconds; the GB technique had a maximum time of 8.55 seconds; and the M, G and B techniques all had a maximum time exceeding 45 seconds.

Figure 57 shows how long the GeoJSON string returned from the *get_points_as_geojson_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 22 of the appendix.

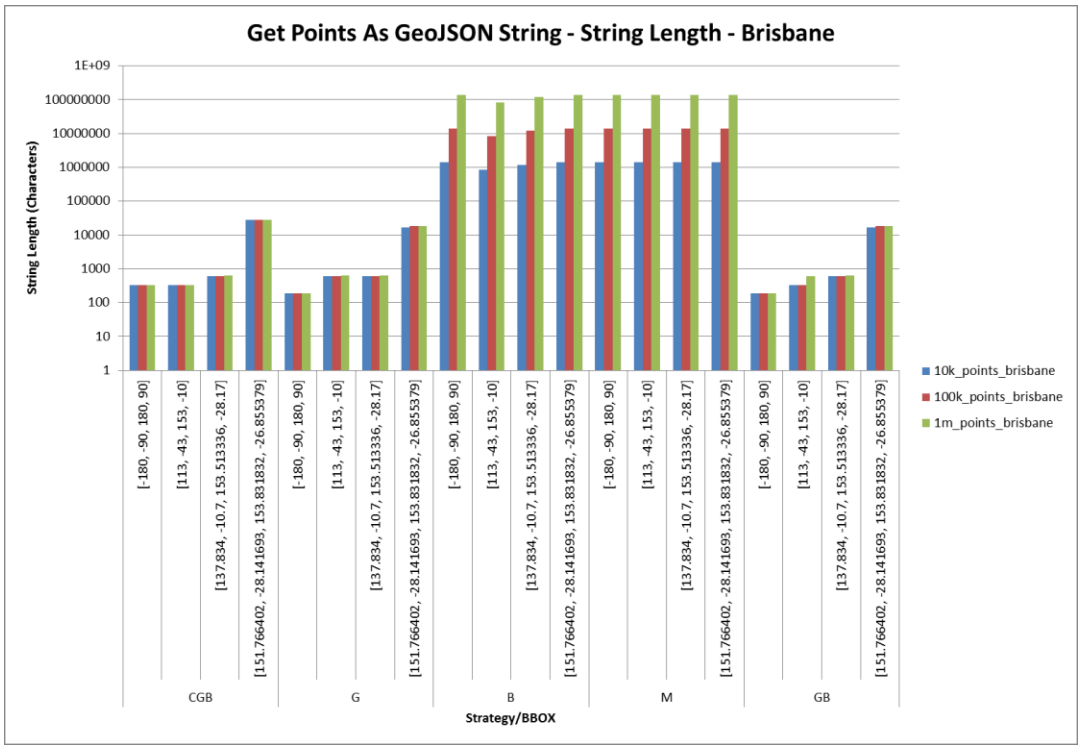

Figure 57: Get Points as GeoJSON String – String Length (Characters) – Brisbane wide; Summarised

The M, G and B techniques all had a maximum string size of 140,777,312 characters, or 134.3 megabytes; and the CGB and GB techniques had a maximum string size of 815,263 characters, or 796 kilobytes.

4.3.4 WKT

Figure 58 shows how long it took, in seconds, to return from the *get_points_as_wkt* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 23 of the appendix.

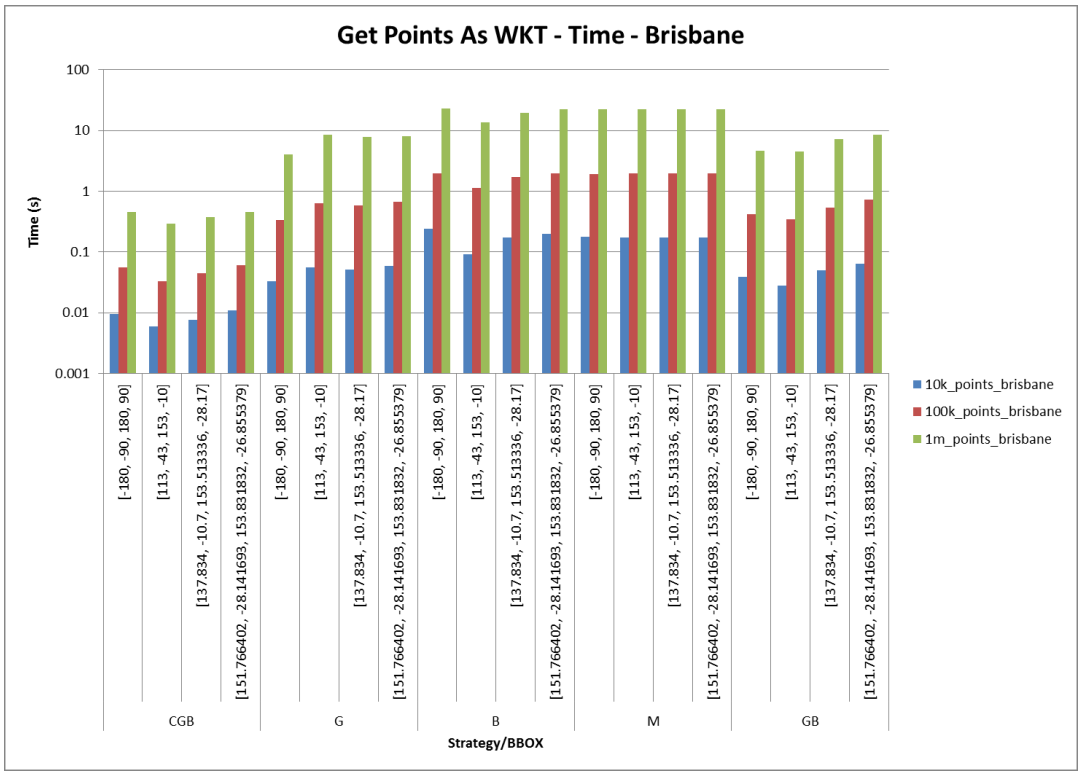

Figure 58: Get Points as WKT – Time (s) – Brisbane wide; Summarised

The CGB technique had a maximum time of 3.97 seconds; the GB technique had a maximum time of 8.5 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 59 shows how long it took, in seconds, to return from the *get_points_as_wkt_str* function for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 24 of the appendix.

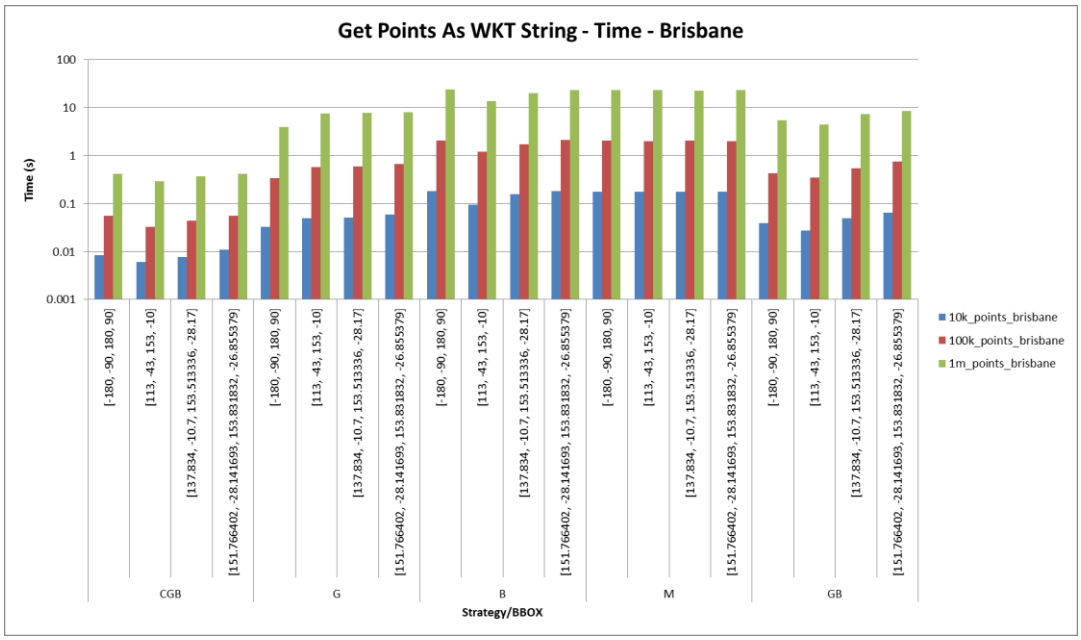

Figure 59: Get Points as WKT String – Time (s) – Brisbane wide; Summarised

The CGB technique had a maximum time of 0.4 seconds; the GB technique had a maximum time of 8.5 seconds; and the M, G and B techniques all had a maximum time exceeding 20 seconds.

Figure 60 shows how long the WKT string returned from the *get_points_as_wkt_str* function was, in characters, for the first four steps of the test scenario. Steps 5 through 8 have been omitted from the summary, and are available in Table 25 of the appendix.

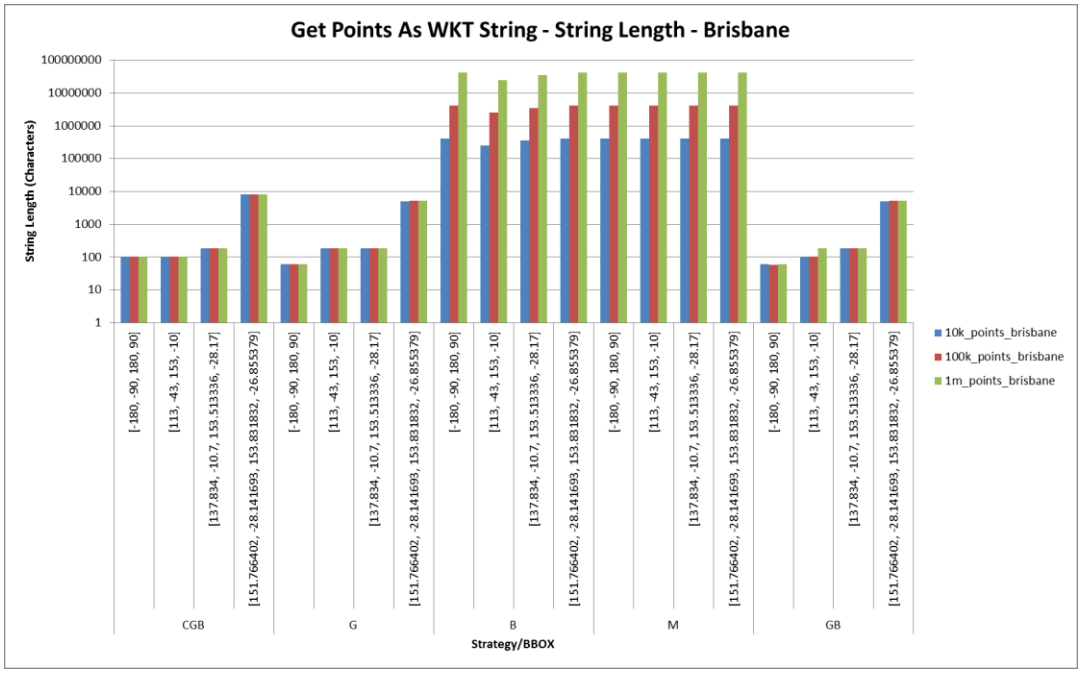

Figure 60: Get Points as WKT String – String Length (Characters) – Brisbane Wide; Summarised

The M, G and B techniques all had a maximum string size of 41,777,387 characters, or 39.8 megabytes; and the CGB and GB techniques had a maximum string size of 241,930 characters, or 236.26 kilobytes.

5 Discussion

In the Proposed Method of Comparison, section 3.1.1, we described that through our abstraction, we would measure three things: the computation time it takes to perform specific processes, the size of the transmission ready result sets, and the size of the database storage required by the technique. We will now discuss the result of those measures for each visualisation technique.

5.1 Mappable Point

Mappable Point was intended to act as our control, as it does not use bounding box, gridded clustering, or caching strategies. As this technique only clusters locations with the exact same location, it, along with the Bound Mappable Point technique, are the two techniques that provide the most accurate representation of the input data set. As the response to a request doesn't vary with the client's bounding box, the client using this technique needs only to request the data set once. The result can be then used for all future interactions with the data set.

5.1.1 Processing Requirements

The Mappable Point has no pre-processing requirements. This means that this technique can be instantly applied, without any set up, and the underlying points can change at any time. This is especially beneficial if the underlying data set is frequently changing.

If we consider requesting the GeoJSON string representation of the input data sets, this technique was able to produce a result for the worldwide 10,000 point input data set in approximately 0.4 seconds. The 10,000 point Australia wide and Brisbane wide input data sets also took 0.4 seconds. This increased to 4.2 seconds for the 100,000 point input data set, and 45 seconds for the 1,000,000 point data set. This equates to approximately 0.045 seconds per 1000 points. Assuming that the number of seconds per thousand points is 0.045 between 10,000 points and 100,000 points, this technique would be able to process a GeoJSON request for approximately 22,000 points within a second. This, as expected, falls significantly short of our research goal of processing a 1,000,000 point data set request within 1 second.

If we now consider requesting the WKT string representation of the points, this technique was able to produce a result for the worldwide 10,000 point input data set in approximately 0.2 seconds. The 10,000 point Australia wide and Brisbane wide input data sets also took 0.2 seconds. This increased to 2 seconds for the 100,000 point input data set, and 23 seconds for the 1,000,000 point data set. This equates to approximately 0.02 seconds per 1000 points.

Assuming that the number of seconds per thousand points is 0.02 between 10,000 points and 100,000 points, this technique would be able to process a WKT request for approximately 50,000 points within a second. This falls significantly short of our research goal of processing a 1,000,000 point data set request within 1 second.

5.1.2 Transmission Requirements

We should also consider the transmission requirements of this technique. The GeoJSON request for the 1,000,000 point Worldwide data set resulted in a 134 megabyte string. This string would take 2 minutes, 27 seconds to download with an 8 Mbit/s connection and 4 minutes, 10 seconds with a 4.7 Mbit/s connection. In regards to this technique, the size of the GeoJSON string is linearly related to the number of points. We can see that there are approximately 140 bytes for every point. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive at most 1,048,576 bytes, which equates to 7500 points.

In regards to the transmission requirements of a WKT request, we can see that a request for the 1,000,000 point Worldwide data set resulted in a 39.9 megabyte string. This string would take 43 seconds to download with an 8 Mbit/s connection and 1 minute, 14 seconds with a 4.7 Mbit/s connection. In regards to this technique, the size of the WKT string is linearly related to the number of points. We can see that there are approximately 42 bytes for every point. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive at most 1,048,576 bytes, which equates to 25,000 points.

5.1.3 Processing and Transmission Requirements Combined

We will now discuss the combined time to process the client's request, and transmit the result. It will take 3 minutes, 12 seconds for an 8 Mbit/s connection to receive the response to a GeoJSON request for the 1,000,000 point data set. It will take 1 minute, 6 seconds for an 8 Mbit/s connection to receive the response to a WKT request for the 1,000,000 point data set.

5.1.4 Summary

In summary, the Mappable Point technique: is unaffected by the client's bounds; did not require pre-processing; is unaffected by the distribution of points; had approximately linear performance between 100 and 1,000,000 points; took 0.045 seconds per 1,000 points to process a GeoJSON request; took 0.02 seconds per 1,000 points to process a WKT request; produced 140 bytes of GeoJSON per point; and produced 42 bytes of WKT per point. Based on these results, we can clearly see that, as expected, this technique is unable to meet our goal of processing a 1,000,000 point data set request within 1 second.

5.2 Bound Mappable Point

Bound Mappable Point was intended to evalute using a bounding box in isolation. This technique does not use gridded clustering, or caching strategies.

5.2.1 Processing Requirements

The Bound Mappable Point has no pre-processing requirements. This means that, like the Mappable Point, it does not require set up, and the underlying points can change at any time.

When requesting GeoJSON for the Worldwide input data set, with a bbox of [-180, -90, 180, 90], this technique produced results almost identical to that of the Mappable Point; 46 seconds for 1,000,000 points, 4.3 seconds for 100,000 points and 0.39 seconds for 10,000 points. This was also true for the Australia wide and Brisbane wide input data sets. It is important to understand that for this bbox value, all points fall within the bbox.

As soon as we adjust the bbox, and subsequently change the number of points that fall within the bbox, we start to see very significant differences between this technique and the Mappable Point technique. When requesting GeoJSON for the Worldwide 1,000,000 point input data set, with a bbox of [113, -43, 153, -10], we find that the processing time drops to 0.78 seconds. Within these bounds, 20,273 clusters were produced. If now consider the Australian input data set, the processing time is approximately 45 seconds for the same bbox. This bbox contains 1,000,000 clusters for the Australia input data set. This suggests that the Bounding Mappable Point technique's processing time is almost entirely dictated by the number of points found within the bbox. Based on our results, we can say that there exists an approximately linear relationship between the number of points within the bbox, and the time it takes to process the request. It will take approximately 0.045 seconds for every 1,000 points found within the bbox. This suggests that this technique will be able to process a GeoJSON request within a second for any bbox that contains up to 22,000 points.

When requesting WKT for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique, again, produced results almost identical to the Mappable Point technique; 23.5 seconds for 1,000,000 points, 2.16 seconds for 100,000 points and 0.19 seconds for 10,000 points. As we previously described, this technique is almost entirely influenced by the number of points within its bbox. If we reduce the bbox to [113, -43, 153, -10], we find that the processing time drops to 0.43 seconds for 1,000,000 points, 0.035 seconds for 100,000 points and 0.005 seconds for 10,000 points. These data sets had 20,273, 2,077, and 225 points within them respectively. This equates to approximately 0.02 seconds for every

1,000 points within the bbox. This suggests that this technique will be able to process a WKT request within a second for any bbox that contains up to 50,000 points.

5.2.2 Transmission Requirements

We will now discuss the transmission requirements for this technique. For the [-180, -90, 180, 90] bbox, this technique produces the same string as the Mappable Point technique. This means that a GeoJSON request for the 1,000,000 point Worldwide data set produced a 134 megabyte string. This string would take 2 minutes, 27 seconds to download with an 8 Mbit/s connection. With a bbox of [113, -43, 153, -10], we find that the string produced by a GeoJSON request for the Worldwide data set is only 2.7 megabytes. As we would expect, there is a direct relationship between the number of points within the bbox, and the size of returned string. There are approximately 140 bytes produced for every point found within the bbox. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive, at most, 7,500 points. Unlike the Mappable Point technique, this requirement is for the number of points within the bbox, regardless of the total number of points within the data set.

In regards to the transmission requirements of a WKT request, we can see that a request for the 1,000,000 point Worldwide data set resulted in a 39.9 megabyte string. This is, once again, identical to that of the Mappable Point technique. This string would take 43 seconds to download with an 8 Mbit/s connection and 1 minute, 14 seconds with a 4.7 Mbit/s connection. With a bbox of [113, -43, 153, -10], we find that the string produced by a WKT request for the Worldwide data set is only 827 kilobytes. Once again, we note a direct relationship between the number of points within the bbox, and the size of returned string. There are approximately 42 bytes produced for every point found within the bbox. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive, at most, 25,000 points. Again we note that unlike the Mappable Point technique, this requirement is for the number of points within the bbox, regardless of the total number of points within the data set.

5.2.3 Processing and Transmission Requirements Combined

The Bound Mappable Point technique has identical processing and transmission requirements for the bbox [-180, -90, 180, 90], as the Mappable Point technique has: 3 minutes, 12 seconds for an 8 Mbit/s connection to receive the response to a GeoJSON request for the 1,000,000 point data set and 1 minute, 6 seconds for an 8 Mbit/s connection to receive the response to a WKT request for the 1,000,000 point data set. The time required scales approximately linearly with the number of points within the bbox, equating to 0.192

seconds per 1,000 points within the bbox for GeoJSON, and 0.066 seconds per 1,000 points within the bbox for WKT.

5.2.4 Summary

In summary, the Bound Mappable Point technique: has a processing time almost entirely dictated by the number of points within the client's bounds; did not require pre-processing; had approximately linear performance between 100 and 1,000,000 points falling within the client's bounds; took 0.045 seconds per 1,000 points within the bbox to process a GeoJSON request; took 0.02 seconds per 1,000 points within the bbox to process a WKT request; produced 140 bytes of GeoJSON per point within the bbox; and produced 42 bytes of WKT per point within the bbox. Based on these results, we can see that this technique would only be able to meet our goal of processing a 1,000,000 point data set request within 1 second under special circumstances. Specifically, the client's zoom level would need to be restricted so that the client could only request bounding boxes with at most 5,200 points for GeoJSON, and 15,000 points for WKT. To restrict the client's zoom level appropriately, the data set's distribution would need to be analysed as a pre-processing step. These restrictions mean that, when interacting with very large data sets, such as 1,000,000 points, this technique would likely only be useful for interacting with highly localised versions of worldwide data. Should the client need to interact with the data set in a way that requires understanding larger portions of the data set, such as generalisations and trends, then this technique will not be sufficient. For example, assuming we have a data set of all restaurants Worldwide, this technique may be useful for viewing what restaurants are within your suburb, but you could not zoom out to view how many restaurants were within your state compared to another state.

5.3 Gridded Mappable Point

Gridded Mappable Point was intended to demonstrate the qualities of using a gridded clustering strategy in isolation. This technique does not use bounding box or caching strategies. Whilst this technique does not use the client's bounding box to filter the data points, it does use the bounding box to calculate the appropriate grid size for the gridded clustering strategy.

5.3.1 Processing Requirements

The Gridded Mappable Point has no pre-processing requirements. This means, like the Mappable Point, it does not require set up, and the underlying points can change at any time.

When requesting GeoJSON for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 8.65 seconds for 1,000,000 points, 0.75 seconds for

100,000 points and 0.068 seconds for 10,000 points. All three of these results produced 105 clusters. For the Australia wide input data sets, this technique produced results in 8.14 seconds for 1,000,000 points, 0.64 seconds for 100,000 points and 0.053 seconds for 10,000 points. All three of these results produced 9 clusters. For the Brisbane wide input data sets, this technique produced results in 4 seconds for 1,000,000 points, 0.34 seconds for 100,000 points and 0.033 seconds for 10,000 points. All three of these results produced only 1 cluster. These results suggest that the Gridded Mappable Point is faster when the distribution of the points is reduced.

As we reduce the bbox, we effectively shrink the size of the grid used by the gridded clustering strategy, and in turn increase the number of clusters produced. When requesting GeoJSON for the Worldwide 1,000,000 point input data set with a bbox of [151.766402, -28.141693, 153.831832, -26.855379], we find that the processing time is 44.6 seconds, and 810,639 clusters are produced. If we now consider the Australia wide input data set, the processing time is approximately 10.3 seconds for the same bbox, and 47,080 clusters are produced. The Brisbane wide input dataset required a processing time of approximately 8 seconds, and produced 126 clusters.

This suggests that the Gridded Mappable Point technique's processing time is primarily determined by the number of clusters in the result. The number of clusters produced is determined by the size of the grid used for clustering, and the distribution of the points that are being processed. As the distribution of the points becomes tighter, such that fewer grids contain points, the processing time decreases. Similarly, as the grid size decreases, in this case by reducing the area of the client's bounding box, the points fall into more discrete grids, and the processing time increases.

When requesting WKT for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 8.23 seconds for 1,000,000 points, 0.68 seconds for 100,000 points and 0.068 seconds for 10,000 points. All three of these results produced 105 clusters. As we previously described, this technique is influenced by the distribution of the points, and the client's bbox. If we reduce the bbox to [151.766402, -28.141693, 153.831832, -26.855379], while maintaining the same distribution, we find that the processing time increases to 26.7 seconds for 1,000,000 points, 2.8 seconds for 100,000 points and 0.25 seconds for 10,000 points. The resulting data sets had 810,639, 97,782, and 9,975 clusters respectively.

5.3.2 Transmission Requirements

With a bbox of [-180, -90, 180, 90], this technique produced 105 clusters for the 10,000, 100,000 and 1,000,000 point Worldwide input data sets. For a GeoJSON request, this resulted in a string 15,234 characters long. This equates to 14.88 kilobytes. This string would take 0.014 seconds to download for an 8 Mbit/s connection, and 0.025 seconds for a 4.7 Mbit/s connection. With a bbox of [137.834, -10.7, 153.513336, -28.17] for the 10,000, 100,000 and 1,000,000 point data sets, this technique produced 8,152, 23,449, and 23,870 clusters respectively and 1,147,739, 3,301,900, and 3,384,428 characters respectively. As we would expect, there is a direct relationship between the number of clusters produced and the size of the returned string. There are approximately 140 bytes produced for every cluster produced. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive, at most, 7,500 clusters. For a 4.7 Mbit/s connection, this drops to 4400 clusters.

When making a WKT request with a bbox of [137.834, -10.7, 153.513336, -28.17] for the 10,000, 100,000 and 1,000,000 point data sets, this technique produced 8,152, 23,449, and 23,870 clusters respectively, and 340,691, 980,284, and 997,910 characters respectively. We note that, as expected, there is a direct relationship between the number of clusters returned, and the size of the returned string. There are approximately 42 bytes produced for every cluster in the result. To receive the transfer within 1 second for an 8 Mbit/s connection, we would only be able to receive, at most, 25,000 clusters. For a 4.7 Mbit/s connection, this would reduce to approximately 14,500 clusters.

5.3.3 Processing and Transmission Requirements Combined

In regards to the Worldwide 1,000,000 point data set, the quickest Gridded Mappable GeoJSON request processing time was 8.65 seconds. This occurred for the bbox [-180, -90, 180, 90]. The transmission size associated with this request was 14.88 kilobytes, and the transmission time was 0.014 seconds with an 8 Mbit/s connection. This means that it would take 8.664 seconds to receive a response to the fastest Worldwide data set request, assuming an 8 Mbit/s connection. This reduces to 4 seconds for the Brisbane wide data set with the same bbox. A WKT request for the Worldwide 1,000,000 point data set with a supplied bbox of [-180, -90, 180, 90] would take 8.2342 seconds. This reduces to 3.977 seconds for the Brisbane wide data set with the same bbox. For these bounding boxes, the number of clusters in the response is small, and so the time to receive the response is primarily dictated by the processing time of the request.

We note that as the number of clusters increases, so does the time required to transmit them. Considering a GeoJSON request for the Worldwide data set with an 8 Mbit/s connection, we believe that a tipping point exists at approximately 75,000 clusters. At this point, the time required to process the result will be approximately 11 seconds, and the time required to transmit the result will be approximately 10 seconds. Any GeoJSON request for this data set with more than 75,000 clusters will have its total request time determined primarily by the transmission time, and not by the processing time. As WKT is a lighter protocol, requiring a smaller transmission size per cluster, the tipping point occurs at approximately 300,000 points, where the processing time will be approximately 12 seconds, and the transmission time will be approximately 12 seconds.

5.3.4 Summary

In summary, the Gridded Mappable Point technique: had a processing time that is a factor of both the total number of points in the data set, and the number of clusters returned by a request; did not require pre-processing; for a given data set, had a processing time that scaled approximately linearly with the number of clusters produced; had a processing time that scaled approximately linearly with the number of points in the total data set; produced 140 bytes of GeoJSON per cluster returned; and produced 42 bytes of WKT per cluster returned. Based on these results, we don't believe this technique would be able to meet our goal of processing a 1,000,000 point data set request within 1 second for any bbox.

5.4 Gridded and Bound Mappable Point

The Gridded and Bound Mappable Point technique provides the benefit of coupling the gridded clustering and bounding box strategies, without requiring caching.

5.4.1 Processing Requirements

The Gridded and Bound Mappable Point has no pre-processing requirements. This means that it does not require set up, and the underlying points can change at any time.

When requesting GeoJSON for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 8.85 seconds for 1,000,000 points, 0.79 seconds for 100,000 points and 0.068 seconds for 10,000 points. All three of these results produced 105 clusters. For the Australia wide input data sets, this technique produced results in 8.63 seconds for 1,000,000 points, 0.71 seconds for 100,000 points and 0.059 seconds for 10,000 points. All three of these results produced 9 clusters. For the Brisbane wide input data sets, this technique produced results in 4.5 seconds for 1,000,000 points, 0.43 seconds for 100,000 points and 0.038 seconds for 10,000 points. All three of these results produced only 1 cluster. These results show that for this bbox, where every point is within the bounds, the

Gridded and Bound Mappable technique has marginally worse performance than the Gridded Mappable Point, and produces the same results.

Unlike the Gridded Mappable Point technique, this technique generally shows improved performance as the bbox is reduced. This is because, as the bbox reduces, the number of points within the bounding box is generally reduced, and so the technique needs to only perform gridded clustering on a smaller number of points.

As we reduce the size of the bbox, we reduce the total number of points that need to be processed by the clustering strategy. By reducing the bbox, we also reduce the size of the grid used by the gridded clustering strategy. This, as has previously been mentioned, is to ensure that approximately 100 clusters are returned, regardless of the size of the bbox. When requesting GeoJSON for the Worldwide 1,000,000 point input data set with a bbox of [113, -43, 153, -10], we find that the processing time is 0.24 seconds, and 120 clusters are produced. If we now consider the Australia wide input data set, the processing time is approximately 8.8 seconds for the same bbox, and 120 clusters are produced. We note that for the Worldwide data set, this bbox contains 20,273 points, but for the Australia wide input data set, this bbox contains all 1,000,000 points. This suggests that the Gridded and Bound Mappable Point technique's processing time is primarily determined by the number of points that fall within the client's bbox. Based on these results, we can say that there exists an approximately linear relationship between the number of points within the bbox, and the time it takes to process the request. It will take approximately 0.01 seconds for every 1,000 points found within the bbox. This suggests that this technique will be able to process a GeoJSON request within a second for any bbox that contains up to 100,000 points.

When requesting WKT for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 8.94 seconds for 1,000,000 points, 0.76 seconds for 100,000 points and 0.066 seconds for 10,000 points. All three of these results produced 105 clusters. As we previously described, this technique is primarily influenced by the number of points within the client's bbox. If we reduce the bbox to [113, -43, 153, -10], while maintaining the same distribution, we find that the processing time reduces to 0.21 seconds for 1,000,000 points, 0.013 seconds for 100,000 points and 0.0053 seconds for 10,000 points. The resulting data sets had 120, 119, and 100 clusters respectively, and the data sets had 20,273, 2,077, and 225 points within the bounding boxes respectively. We see that, unlike the other techniques already discussed, this technique has very similar processing times for both the WKT and GeoJSON. This is because the result sets are relatively small, only consisting of approximately 100 clusters. This means that the processing time

associated with bounding and clustering, dwarfs the amount of time necessary to render the result as WKT or GeoJSON.

5.4.2 Transmission Requirements

With a bbox of [-180, -90, 180, 90], this technique produced 105 clusters for the 10,000, 100,000 and 1,000,000 point Worldwide input data sets. For a GeoJSON request, this resulted in a string 15,234 characters long. This equates to 14.88 kilobytes. This string would take 0.014 seconds to download for an 8 Mbit/s connection, and 0.025 seconds for a 4.7 Mbit/s connection. For the Worldwide input data set, the largest result data set was 129 clusters. This resulted in a string 18,322 characters long. This equates to 17.89 kilobytes. This string would take 0.0175 seconds to download for an 8 Mbit/s connection. The equivalent WKT request would result in only 5,416 characters. This equates to 5.29 kilobytes. This string would take 0.005 seconds to download for an 8 Mbit/s connection. As the Gridded and Bound technique is designed to produce, at most, approximately 100 clusters, we can expect transmission requirements to always be this low, regardless of the size of the input data set.

5.4.3 Processing and Transmission Requirements Combined

For the Worldwide 1,000,000 point input data set, with a bbox of [-180, -90, 180, 90], a GeoJSON request would receive a response in 8.864 seconds, when made with an 8 Mbit/s connection. This would extend to 8.875 seconds, when made with a 4.7 Mbit/s connection. For the same data set, but with a bbox of [137.834, -10.7, 153.513336, -28.17], the request response would be received after 0.05 seconds, when made with an 8 Mbit/s connection.

As previously noted, all requests should produce, at most, approximately 100 clusters. This equates to a transmission time of 0.0175 seconds for a GeoJSON request, when made with an 8 Mbit/s connection. As the processing time is approximately 0.01 seconds for every 1,000 points within the bbox, a user should receive a response to a GeoJSON request within a second, as long as there are less than 98,250 points within the client's bounds. This is assuming the client has an 8 Mbit/s connection. This increases to 99,500 points for a WKT request.

5.4.4 Summary

In summary, the Gridded and Bound Mappable Point technique: has a processing time almost entirely dictated by the number of points within the client's bounds; did not require pre-processing; had approximately linear performance when considering the number points within the client's bounds; took 0.01 seconds per 1,000 points within the bbox to process a GeoJSON or WKT request; produced 140 bytes of GeoJSON per cluster produced; and

produced 42 bytes of WKT per cluster produced. Based on these results, we can see that this technique would only be able to meet our goal of processing a 1,000,000 point data set request within 1 second under special circumstances. The client's zoom level would need to be restricted so that the client could only request bounding boxes with at most 100,000 points. To restrict the client's zoom level appropriately, the data set's distribution would need to be analysed as a pre-processing step. In our test scenario, when considering the Worldwide input data sets, the response time would have been under a second for 7 of the 8 interaction steps.

5.5 Cached, Gridded and Bound Mappable Point

The Gridded, Cached and Bound Mappable Point technique demonstrates the benefit of coupling gridded clustering, bounding box, and caching strategies. The Gridded, Cached and Bound Mappable Point is the only technique to use all three of these strategies.

5.5.1 Processing Requirements

Unlike any other technique, this technique requires a pre-processing step. The pre-processing step is responsible for producing the needed cache tables. For the Worldwide 1,000,000 point data set, the pre-processing time was approximately 160 minutes. This reduced to 82 minutes for the Australia wide input data set, and 29 minutes for the Brisbane wide input data set. Pre-processing the Worldwide input data set would have produced significantly more clusters than the Brisbane wide input data set. As they contain the same number of points, this suggests that the pre-processing time is primarily dictated by the total number of clusters generated by the pre-processing phase, and not by the total number of points within the data set. Assuming that your points are evenly spread across the world, the required pre-processing time is approximately 10 seconds per 1,000 points.

In addition to the processing time associated with the pre-processing function, we can also discuss the database size requirements. In the case of the Worldwide 1,000,000 point input data set, the database increased from 138.2 megabytes to 921.4 megabytes. This equates to a 567% increase in database size. For the Australia wide input data set, the database increased from 131.4 megabytes to 539.3 megabytes. This equates to a 310% increase in database size. For the Brisbane wide input data set, the database increased from 135 megabytes to 276.3 megabytes. This equates to a 105% increase in database size. As the number of clusters produced by the pre-processing defines the increase in database size, these results provide further evidence to the fact that the processing time is primarily dictated by the number of clusters produced by the pre-processing function, and that this relationship is linear.

When requesting GeoJSON for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 8.2 seconds for 1,000,000 points, 0.38 seconds for 100,000 points and 0.046 seconds for 10,000 points. These requests produced 299, 299, and 298 clusters respectively. For the Australia wide input data sets, this technique produced results in 2.3 seconds for 1,000,000 points, 0.207 seconds for 100,000 points and 0.0384 seconds for 10,000 points. All three of these results produced 12 clusters. For the Brisbane wide input data sets, this technique produced results in 0.415 seconds for 1,000,000 points, 0.057 seconds for 100,000 points and 0.0086 seconds for 10,000 points. All three of these results produced 2 clusters. These results show that for this bbox, the Cached, Gridded and Bound Mappable technique has the best performance of any of the techniques.

As the bbox is reduced, the grid size will normalise to a smaller grid size, and a different Cache Record will be processed. With a bbox of [113, -43, 153, -10], for the Worldwide data sets, this technique produced its results in 1.48 seconds for 1,000,000 points, 0.024 seconds for 100,000 points and 0.008 seconds for 10,000 points. The number of clusters produced was 327, 330 and 167 respectively. For the same bbox, and the Australia wide input data sets, this technique produced its results in 2.196 seconds for 1,000,000 points, 0.202 seconds for 100,000 points and 0.25 seconds for 10,000 points. The number of clusters produced was 340 clusters for each of the input data sets. For the Brisbane wide input data sets, this technique produced its results in 0.98 seconds for 1,000,000 points, 0.073 seconds for 100,000 points and 0.0118 seconds for 10,000 points. The number of clusters produced was 2 for each of the input data sets. We can see that the for the Worldwide input data sets, the Cached, Gridded and Bound Mappable Point had similar performance to the Gridded and Bound Mappable Point. For the Australia wide and Brisbane wide input data sets however, this technique had significantly better performance than the Gridded and Bound Mappable Point. In the case of the Brisbane wide input data set, this technique had a process time of less than 1 second for all client bbox values.

When requesting WKT for the Worldwide input data set with a bbox of [-180, -90, 180, 90], this technique produced its results in 3.68 seconds for 1,000,000 points, 0.3 seconds for 100,000 points and 0.04 seconds for 10,000 points. These requests produced 299, 299, and 298 clusters respectively. For the Australia wide input data sets, this technique produced its results in 2.27 seconds for 1,000,000 points, 0.19 seconds for 100,000 points and 0.035 seconds for 10,000 points. All three of these requests resulted in 12 clusters. For the Brisbane wide input data sets, this technique produced its results in 0.4 seconds for 1,000,000 points, 0.056 seconds for 100,000 points, and 0.008 seconds for 10,000 points. All three of these results produced 2 clusters. In regards to WKT requests, this technique

performed better than the Gridded and Bound Mappable Point. For the Worldwide 1,000,000 point input data set, this technique had a processing time approximately twice as fast as the Gridded and Bound Mappable Point for the first two interaction steps. The remaining interaction steps had performance that was similar to the Gridded and Bound Mappable Point. For the Brisbane wide 1,000,000 point input data set, the performance was between 10 and 20 times faster than the Gridded and Bound Mappable Point for the first five interaction steps. The performance was similar to the Gridded and Bound Mappable Point for the three reaming interaction steps.

These results show that the processing time required for this technique is, in almost all cases, faster than any other technique. If we consider GeoJSON requests for the 1,000,000 point data sets, this technique was slower than the Gridded and Bound Mappable Point technique in only 6 of 24 interaction step tests. In the case of WKT requests, this technique was slower than the Gridded and Bound Mappable Point technique in only 4 of 24 interaction steps. For the Brisbane wide 1,000,000 point input data set, this technique was faster than every other technique in every test.

5.5.2 Transmission Requirements

As has already been established, there is a linear relationship between the number of clusters in a response, and the transmission size of the response. For a GeoJSON request, there are approximately 140 bytes produced for every cluster in the result. For a WKT request, there are approximately 42 bytes produced for every cluster in the result. Unlike the Gridded and Bound Mappable Point technique, this technique doesn't always produce a result of approximately 100 clusters or less. This technique is required to normalise to an existing Cache Record's grid size. This means that in some cases, the number of clusters returned can exceed 100. For the Worldwide data sets, the maximum number of clusters returned was 327. For the Australia wide data sets, the maximum number of clusters returned was 340. For the Brisbane wide data sets, the maximum number of clusters returned was 345. For 345 clusters, the GeoJSON response string length is approximately 48,300 characters, or 47.17 kilobytes. This response would take approximately 0.046 seconds for an 8 Mbit/s connection to receive. For a WKT request, the string length would be approximately 14,490 characters, or 14.15 kilobytes. The WKT response would take approximately 0.0138 seconds for an 8 Mbit/s connection to receive.

5.5.3 Processing and Transmission Requirements Combined

For the Worldwide 1,000,000 point input data set, with a bbox of [-180, -90, 180, 90], a GeoJSON request would receive a response in 8.252 seconds, when made with an 8 Mbit/s connection. This would extend to 8.28 seconds, when made with a 4.7 Mbit/s connection.

This is the worst case response for time Worldwide data set. For the same data set, but with a bbox of [137.834, -10.7, 153.513336, -28.17], the request's response would be received after 0.083 seconds, when made with an 8 Mbit/s connection. For the Australia wide data set, the worst case response time to a GeoJSON request was 5.01 seconds for an 8 Mbit/s connection. For the Brisbane wide data set, the worst case response time to a GeoJSON request was 0.983 seconds for an 8 Mbit/s connection.

5.5.4 Summary

In summary, the Cached, Gridded and Bound Mappable Point technique: has a processing time that is generally better than the Gridded and Bound Mappable Point; has a pre-processing time that increases as the distribution of the points increases; produced 140 bytes of GeoJSON per cluster produced; and produced 42 bytes of WKT per cluster produced. Based on these results, we can see that this technique would be able to meet our goal of processing a 1,000,000 point data set within a second if the data set is closely distributed, such as with the Brisbane wide input data set. This technique is the only technique that is able to meet our goal of processing and transmitting a 1,000,000 point data set within a second for an 8 Mbit/s connection. Unfortunately, this technique was not able to meet this standard for the 1,000,000 point Australia wide or Worldwide input data sets. This technique performed the best of all the techniques described within this report.

5.6 Comparison of Visualisation Techniques

We would only recommend using the Gridded and Bound Mappable Point or the Cached, Gridded and Bound Mappable Point technique. These two techniques performed significantly better than the alternatives in almost all tests. They were also the only techniques to produce response outputs of an appropriate size, capable of being transmitted within a second, to all requests.

The perceived benefit of using the Mappable Point, or Bound Mappable Point, is that the input data set is more accurately represented by not using clusters. This benefit can easily be achieved through a minor modification of the Gridded and Bound techniques. The number of points that are returned within the clusters is known at the time of processing the request. This means the Gridded and Bound techniques could be modified to check the number of points the clusters represent, and if it is less than a certain threshold, for example 10,000 points, the un-clustered points could be returned. The overhead this would introduce would be approximately the equivalent of running the Bound Mappable Point technique in addition to the applicable Gridded and Bound technique. This overhead would only apply in situations where the clusters represent less than 10,000 points. Based on our Bounding

Mappable Point results, assuming a threshold of 10,000 points, the overhead would be approximately 0.39 seconds for GeoJSON, and 0.19 seconds for WKT.

The Cached, Gridded and Bound technique had process times comparable to the Gridded and Bound technique for the Worldwide input data set. Two noteworthy exceptions to this are, the WKT request with a bbox of [-180, -90, 180, 90], which the cached technique was able to process in a third of the time, 3 seconds compared to 9 seconds, and the GeoJSON request with a bbox of [113, -43, 153, -10], which the cached technique processed slower than the un-cached technique, 1.5 seconds compared to 0.25 seconds. As we captured the database process time, and the GeoJSON string generation time independently, we are able to see one of factors leading to why the cached technique was slower in this instance. Due to the necessary normalisation of grid sizes, the cached technique produced 327 clusters, to the un-clustered technique's 120 clusters. This resulted in the cached technique needing to spend an increased amount of time generating the GeoJSON string. The amount of time spent generating the GeoJSON does not fully explain this discrepancy, as such, we believe this outlier may have been produced by an unexpected disruption on the test machine.

The Cached, Gridded and Bound technique has significantly better performance than the Gridded and Bound technique for the Australia wide and Brisbane wide input data sets. As the data set becomes less distributed, i.e. the points become closer together on average, the performance of the cached technique improves. As the Worldwide data set represents the worst case scenario for data set distribution, this suggests that the cached technique will outperform the un-cached technique in virtually all cases.

The Cached, Gridded and Bound technique has two main drawbacks when compared to the Gridded and Bound technique. The first is that the cached technique requires a set of defined grid sizes. The number of these grid sizes will define the number of zoom levels you will support for your client. The number of grid sizes also influences the size of the resulting clusters, and the pre-processing time of the cached technique. The second drawback of the cached technique is the pre-processing implications. The cached technique required in excess of 2.5 hours to pre-process the Worldwide data set, and increased the database size to over 6 times its original size. The implication of this should be considered, especially for frequently changing data sets.

We recommend using the Gridded and Bound technique as your default visualisation technique. Where you have sufficient database space and where the data set is stable, i.e. it changes infrequently, we would recommend using the Cached, Gridded and Bound

technique for increased performance. The Cached, Gridded and Bound Mappable Point technique, and the Gridded and Bound Mappable Point technique, provide an answer to our research question, “what geospatial visualisation technique can be used to render and interact with geospatial data sets exceeding one million points?”

5.7 Abstract Testing Framework Discussion

Through our use of a standardised test suite, and common visualisation abstraction, we have been able to successfully compare different visualisation techniques. This demonstrates that our methodology provides an answer to our research question, “how can geospatial visualisation techniques be compared?”

5.8 Limitations

We will now discuss the limitations of our methodology and test system, specifically focusing on how these limitations should be interpreted, and their associated implications.

5.8.1 Clustering – GeoJSON versus WKT

When using a clustered technique, we recommend using GeoJSON to carry the cluster size information. While WKT is a lighter protocol, in regards to transmission size, and performed significantly better in all tests, it is unable to transmit non geospatial information, and as such, cannot transmit the cluster size. By transmitting the number of points in the cluster, we can vary the size of the rendered cluster based on the number of points. This conveys important visual information regarding the data sets. Figure 61 shows an example of using the visual cluster size to convey the number of points within the cluster.

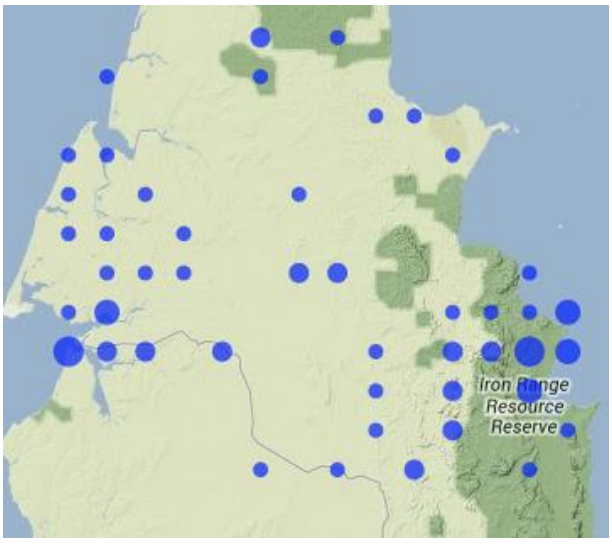

Figure 61: Example use of visual cluster size

This limitation means that in practice, the clustering techniques will likely only be used via the GeoJSON protocol, and not via WKT.

5.8.2 Point Accuracy – Decimal Places

The input data sets generated will not accurately reflect the performance of all real world data sets. The input data sets were generated with 14 decimal places. This meant that the likelihood of 2 points having the same location is extremely low. It also means that the GeoJSON and WKT string representations of these points are very long. This in turn increases the length of the output response strings.

We expect that the way in which the input data was generated is a reasonable depiction of a worst case scenario. It is also the case that all visualisation techniques were tested against common data sets. For these reasons, we believe that while the test input data sets may not accurately represent real world data sets, they do provide an appropriate input to test against, and draw conclusions from.

5.8.3 Performance Relative to Hardware Specifications

It is the case that the performance of these visualisations will be dependent on the system upon which they are executed. We made use of an Amazon EC2 instance, in order to ensure that these tests are standardised, and reproducible. We expect that the processing performance of all visualisation techniques will scale according to the hardware they are executed on. Importantly, the response size will remain constant, meaning that the transmission size will not vary. We believe that the conclusions that have been drawn from the results will still be accurate, regardless of the hardware used. As the test system source code is freely available, users can run the test suite against their own hardware, should they so wish.

5.8.4 Cached, Gridded and Bound Mappable Point – Number of Grid Sizes

The Cached, Gridded and Bound Mappable Point used a fixed set of 15 grid sizes. These grid sizes had an initial value of 0 decimal degrees, and then ranged between 0.015 decimal degrees and 128 decimal degrees. The grid sizes were selected to provide a smooth transition between fully zoomed out, and fully zoomed in. They were chosen to be powers of 2, such that as the user zooms out, the height and width of the cluster grid doubles. This means that as the user zooms in, a single grid should split from the centre into up to 4 new clusters. Bing Maps uses a similar power of 2 system to increase the detail of their maps as the user zooms in [23]. Figure 62 illustrates how the grid sizes may expand.

Figure 62: Approximate grid size expansion at three zoom levels [23]

Our results show that for conventional use, 15 grid sizes generally produced up to 300-400 clusters. These results had an acceptable transmission requirement. Future work should be done to examine the effects of reducing or expanding the number of grid sizes. We expect that as you reduce the number of grid sizes, you will see reduced pre-processing times and database usage. We also expect that the client will normalise to grid sizes further from their ideal grid size, resulting in responses with more clusters. Similarly, we expect that as you increase the number of grid sizes, you will see increased pre-processing requirements, and the client will normalise to a grid size closer to their ideal grid size.

6 Conclusion

We have shown that existing visualisation methods only allow for limited data set sizes. Our testing has shown that conventional techniques can take in excess of 40 seconds to process a request for 1,000,000 points, producing 134 megabytes of GeoJSON output. This demonstrates that there is a need for improved methods of visualising large geospatial data sets via HTTP.

We believe that through our use of a standardised test suite, and common visualisation abstraction, we have been able to successfully compare different visualisation techniques. The benefits of using a common visualisation abstraction, and standardised test suite, include the ability to extend the existing toolkit, the ability to plug-in new visualisation techniques, and the ability to automatically analyse and compare the performance of visualisation techniques with different underlying implementations.

We have also proposed and demonstrated two high performance geospatial visualisation techniques that we believe allow for the visualisation of data sets with in excess of 1,000,000 points. The first proposed technique, Cached, Gridded and Bound Mappable Point, couples bounding, clustering and caching strategies. Through the use of this technique, we have demonstrated the ability to service a WKT request for 1,000,000 points in a worst case time of under 4 seconds. This technique improves significantly to produce responses in less than 1 second for the vast majority of data distributions and client request zoom levels. While this technique had the best performance, it also had a large pre-processing requirement, as such, we have proposed the use of the Gridded and Bound Mappable Point technique, which couples bounding and clustering strategies, for use when the input data set frequently changes. Using either of these two techniques, it will be possible for users to effectively interact with large geospatial data sets.

7 References

[1] S. Dragicevic, "The potential of Web-based GIS," *Journal of Geographical Systems*, vol. 6, pp. 79-81, 2004.

[2] Z. Peng, "An assessment framework for the development of Internet GIS," *Environment and Planning B*, vol. 26, pp. 117-132, 1999.

[3] S. Tu, M. Flanagin, Y. Wu, M. Abdelguerfi, E. Normand, and V. Mahadevan, "Design strategies to improve performance of GIS Web services," in *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, 2004, pp. 444-448.

[4] H. Helali, "Design and Implementation of a Web GIS for the City of Tehran," *Department Of Geodesy And Geomantic Engineering KN Toosi University Of Technology, Submitted thesis for degree of Master Of Science*, 2001.

[5] S. Shekhar and S. Chawla, *Spatial databases: a tour* vol. 2003: Prentice Hall Englewood Cliffs, 2003.

[6] J. R. Herring, "OpenGIS Implementation Specification for Geographic information-Simple feature access-Part 2: SQL option," *Open Geospatial Consortium Inc*, 2006.

[7] A. Ballatore, A. Tahir, G. McArdle, and M. Bertolotto, "A comparison of open source geospatial technologies for web mapping," *International Journal of Web Engineering and Technology*, vol. 6, pp. 354-374, 2011.

[8] A. MySQL, "MySQL 5.6 reference manual," 2013.

[9] J. d. l. Beaujardiere, "OpenGIS Web Map Server Implementation Specification," *Open Geospatial Consortium Inc*, 2006.

[10] P. P. A. Vretanos, "OpenGIS Web Feature Service 2.0 Interface Standard," *Open Geospatial Consortium Inc*, 2010.

[11] P. Baumann, "OGC® WCS 2.0 Interface Standard - Core," *Open Geospatial Consortium Inc*, 2010.

[12] S. Steiniger and A. J. Hunter, "The 2012 free and open source GIS software map–A guide to facilitate research, development, and adoption," *Computers, Environment and Urban Systems*, 2012.

[13] D. M. Lupp, "Styled Layer Descriptor profile of the Web Map Service Implementation Specification," *Open Geospatial Consortium Inc*, 2007.

[14] C. Portele, "OGC® Geography Markup Language (GML) — Extended schemas and encoding rules," *Open Geospatial Consortium Inc*, 2012.

[15] G. Rocher, P. Ledbrook, M. Palmer, J. Brown, L. Daley, and B. Beckwith, "Grails 2.2.1 Framework - Reference Documentation," 2013.

[16] M. Björemo and P. Trninić, "Evaluation of web application frameworks-Evaluation of web application frameworks with regards to rapid development," 2010.

[17] E. Hazzard, *Openlayers 2.10 beginner's guide*: Packt Publishing Ltd, 2011.

[18] G. S. o. t. G. C. o. t. I. A. o. O. G. P. (OGP), "EPSG Geodetic Parameter Dataset v8.1," 2012.

[19] J. Masó, K. Pomakis, and N. Julià, "OpenGIS Web Map Tile Service Implementation Standard," *Open Geospatial Consortium Inc*, 2010.

[20] A. Technologies, "The State of the Internet," vol. 6, 2013.

[21] P. Ramsey, "PostGIS manual," *Postgis. pdf file downloaded from Refractions Research home page*, 2005.

[22] Google, "Google Maps," 2013.

[23] J. Schwartz, "Bing Maps Tile System."

8 Appendices

8.1 Risk Assessment

JAMES COOK UNIVERSITY
AUSTRALIA

Townsville Campus
Townsville Qld 4811 Australia
School of Engineering & Physical Sciences

Risk Assessment
Name of Test Modelling of Geospatial Visualisation Techniques

Purpose: Thesis			
Operator: Robert Pyke		**Duration: 10 months (March 2013 – Dec 2013)**	
SDS Attached: (Tick one)	Yes	No	N/A ✓
Major Hazard Types: (Tick at least one)			
Chemical		Mechanical	
Electrical		Thermal	
Environmental ✓		Other: **Ergonomic ✓**	

SUMMARY OF RISKS

Specific Task/Activity	Potential Hazards/Consequences	Assessed Risk	Risk Control Measures	Reassessed Risk
Use of desktop workstation (PC, Monitor, Keyboard and Mouse)	Eyestrain; musculoskeletal problems; repetitive movement strain; tripping and fatigue.	MEDIUM	Use ergonomic office equipment, including: chairs, wrist rests, and keyboards; Ensure adequate lighting and window blinds; Take a break every 30 minutes to look away from the workstation; Take a break every 2 hours to walk around.	LOW
Movement (walking) around workstation	Tripping; falling materials and housekeeping hazards.	MEDIUM	Ensure good office organisation and layout; Ensure all work surfaces are in good condition; Ensure liquid spillages are cleared up promptly; Ensure adequate lighting and window blinds; Ensure that items are appropriately stored; Ensure cabinet doors & drawers are kept shut when not in use.	LOW

SUMMARY OF REQUIREMENTS

Personal Protective Equipment	N/A
Is Training Required If YES, please state requirements	No
Training Manual Location	N/A

Revised: 1 May 2013

SUMMARY OF ACTIVITY

Perform modelling of geospatial visualisation techniques using JCU's High Performance Computing (HPC) infrastructure via remote terminal control.

ASSESSMENT:

OPERATOR (Student or Technician):

ROBERT PYKE — Date 01/5/13 Contact No: 0427835642

Name — Signature

SUPERVISOR:

Owen Kenny — Date: 1/5/13 Contact No: 14279

Name — Signature

SAFETY ADVISOR:

Warren O'Donnell — Date: 1/5/13 Contact No: 14118

Name — Signature

HEAD OF DISCIPLINE:

AHMAD ZAHEDI — Date: 02/05/2013 Contact No: 16907

Name — Signature

THIS FORM IS TO BE DISPLAYED IN THE IMMEDIATE VICINITY OF THE EXPERIMENT BEING UNDERTAKEN

Revised: 1 May 2013

8.2 Results

8.2.1 Worldwide

Strategy	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	0.223858	1.265362	10.72881	98.48752	860.1655	9590.002
G	1.40E-05	3.30E-05	1.40E-05	1.50E-05	2.90E-05	1.50E-05
B	1.30E-05	1.40E-05	1.40E-05	3.60E-05	1.50E-05	1.50E-05
M	1.30E-05	1.30E-05	1.40E-05	1.60E-05	1.70E-05	1.70E-05
GB	1.30E-05	1.30E-05	1.50E-05	1.50E-05	1.60E-05	1.50E-05

Table 2: Pre-Processing – Time (s) – Worldwide

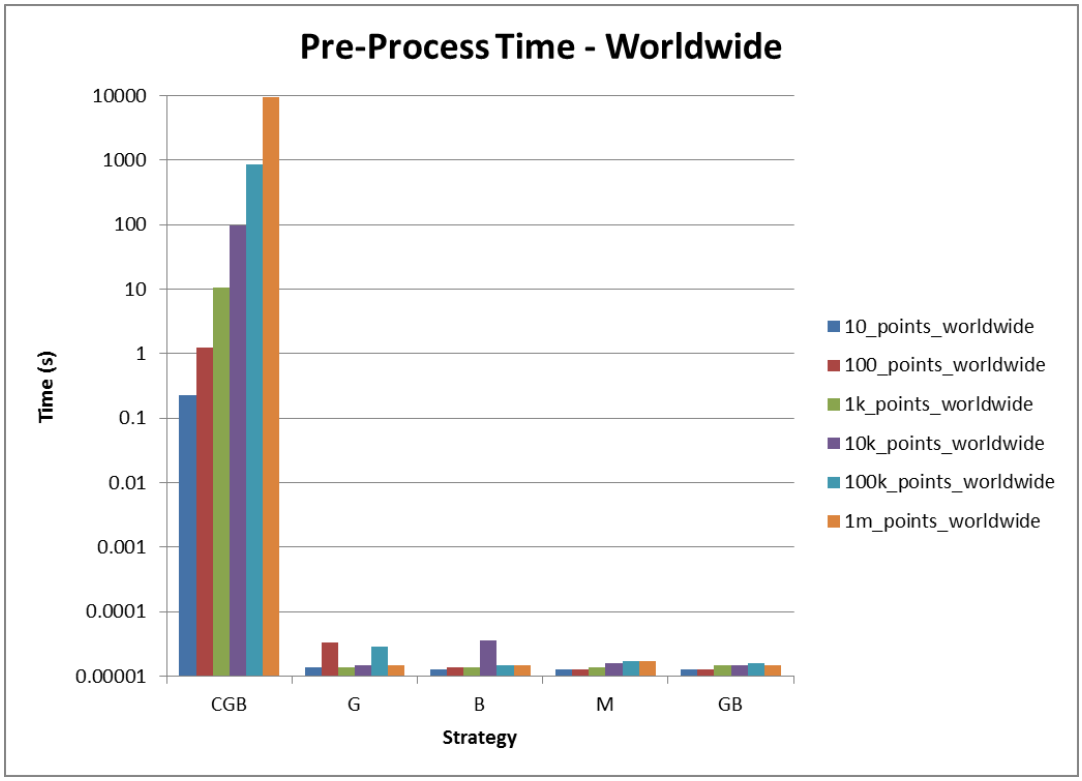

Figure 63: Pre-Processing – Time (s) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	10	84	258	298	299	299
CGB	[113, -43, 153, -10]	0	2	18	167	330	327
CGB	[137.834, -10.7, 153.513336, -28.17]	0	0	2	38	230	287
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0	0	0	0	6	31
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	0	0	4	3
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	0
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0
G	[-180, -90, 180, 90]	9	59	104	105	105	105
G	[113, -43, 153, -10]	10	98	900	4360	5151	5151
G	[137.834, -10.7, 153.513336, -28.17]	10	99	978	8152	23449	23870
G	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	1000	9975	97782	810639
G	[152.557418, -27.738907, 153.266036, -27.271171]	10	100	1000	9997	99728	973513
G	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	1000000
G	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	1000000
G	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	1000000
B	[-180, -90, 180, 90]	10	100	1000	10000	100000	1000000
B	[113, -43, 153, -10]	0	3	20	225	2077	20273
B	[137.834, -10.7, 153.513336, -28.17]	0	0	2	40	462	4125
B	[151.766402, -28.141693, 153.831832, -26.855379]	0	0	0	0	6	33
B	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	0	0	4	3
B	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	0
B	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
B	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0
M	[-180, -90, 180, 90]	10	100	1000	10000	100000	1000000
M	[113, -43, 153, -10]	10	100	1000	10000	100000	1000000
M	[137.834, -10.7, 153.513336, -28.17]	10	100	1000	10000	100000	1000000
M	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	1000	10000	100000	1000000
M	[152.557418, -27.738907, 153.266036, -27.271171]	10	100	1000	10000	100000	1000000
M	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	1000000
M	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	1000000
M	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	1000000
GB	[-180, -90, 180, 90]	9	59	104	105	105	105
GB	[113, -43, 153, -10]	0	2	17	100	119	120
GB	[137.834, -10.7, 153.513336, -28.17]	0	0	2	36	115	129
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0	0	0	0	6	27
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	0	0	4	3
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	0
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0

Table 3: Clusters Generated – Worldwide

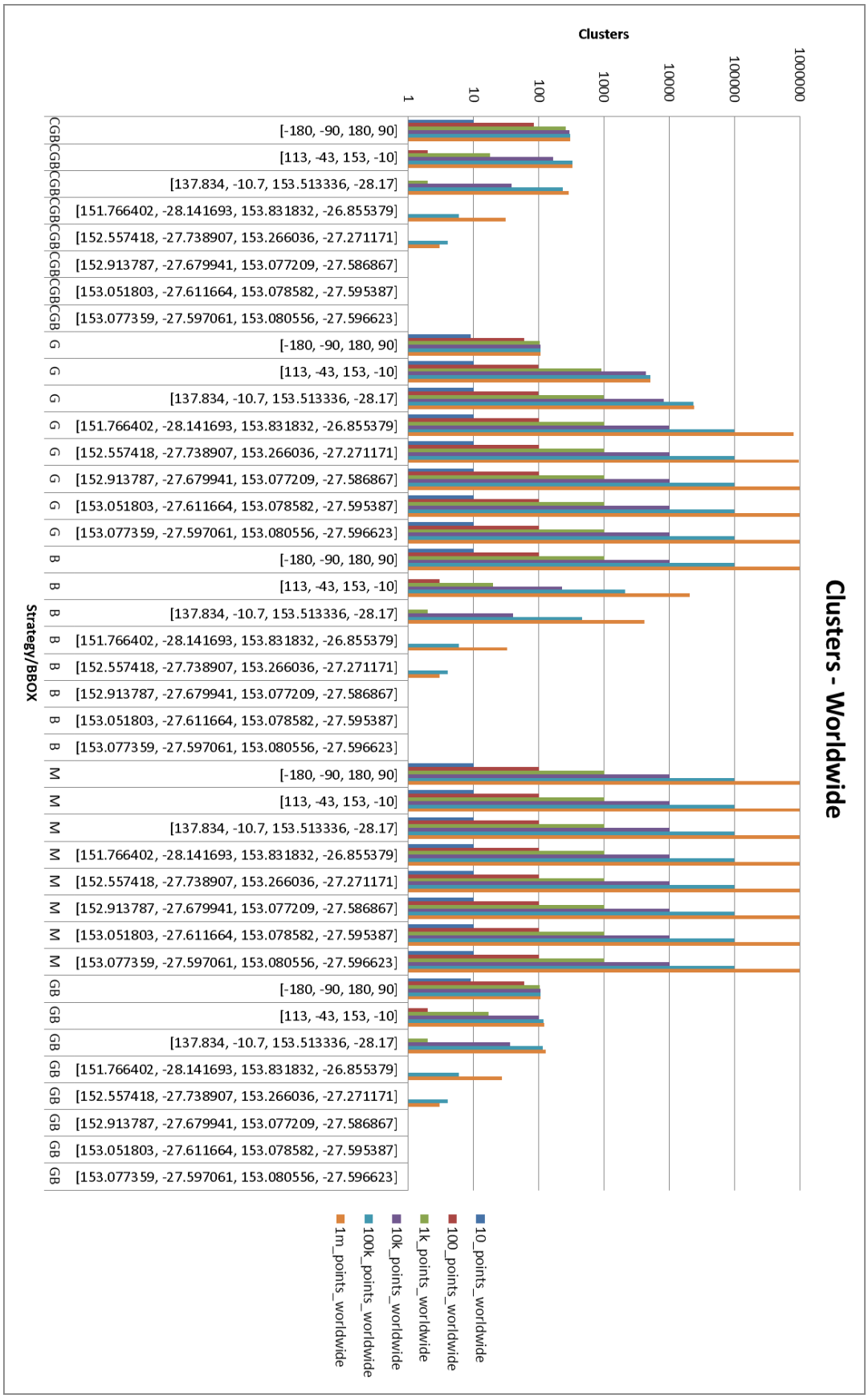

Figure 64: Clusters Generated – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.003007	0.004337	0.014693	0.120534	0.784828	8.744032
CGB	[113, -43, 153, -10]	0.001374	0.001444	0.001847	0.005185	0.020005	0.179669
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001389	0.001392	0.001512	0.002452	0.008455	0.063271
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001375	0.00172	0.00146	0.001569	0.001996	0.004106
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001679	0.00138	0.001444	0.001933	0.001857	0.002389
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001379	0.001367	0.001453	0.001538	0.001478	0.001826
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001365	0.001375	0.001452	0.001531	0.001475	0.00174
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001362	0.0017	0.001447	0.001496	0.001465	0.001705
G	[-180, -90, 180, 90]	0.002335	0.003324	0.007232	0.061756	0.685716	8.532206
G	[113, -43, 153, -10]	0.001794	0.004266	0.02366	0.154378	0.848984	8.708809
G	[137.834, -10.7, 153.513336, -28.17]	0.00207	0.0038	0.024342	0.221799	1.588351	8.774199
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001871	0.003837	0.024702	0.279356	2.808026	29.50946
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.002076	0.003934	0.02523	0.278458	2.861875	30.81089
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001796	0.003832	0.025104	0.287193	2.973437	33.06865
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001778	0.00383	0.044588	0.28752	3.004445	32.74536
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.002032	0.003826	0.024892	0.268719	2.999309	32.73291
B	[-180, -90, 180, 90]	0.001976	0.003319	0.017058	0.251809	2.711924	27.58168
B	[113, -43, 153, -10]	0.001619	0.001692	0.002007	0.005635	0.036473	0.443279
B	[137.834, -10.7, 153.513336, -28.17]	0.001619	0.001641	0.001769	0.002297	0.010166	0.097665
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001624	0.001599	0.001695	0.001636	0.002453	0.002899
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001612	0.001625	0.0017	0.001609	0.001906	0.002124
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.0024	0.001623	0.001688	0.001631	0.00176	0.001901
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001622	0.001619	0.001692	0.001629	0.001753	0.001897
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001621	0.001616	0.001711	0.00162	0.001759	0.001895
M	[-180, -90, 180, 90]	0.001528	0.002899	0.016535	0.199921	4.497775	27.68949
M	[113, -43, 153, -10]	0.001391	0.002668	0.016134	2.257196	2.117611	23.32011
M	[137.834, -10.7, 153.513336, -28.17]	0.001399	0.0027	0.015725	0.184599	2.051624	23.18437
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001395	0.00284	0.015817	0.183702	2.029498	23.42439
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001387	0.002697	0.038684	0.184712	2.061113	23.8416
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001406	0.002693	0.01614	0.183873	2.100731	23.31434
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001416	0.002684	0.01568	0.184328	2.102845	23.27052
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.00143	0.00268	0.01576	0.184635	2.195387	23.25816
GB	[-180, -90, 180, 90]	0.002271	0.003601	0.008162	0.070241	0.786551	8.581553
GB	[113, -43, 153, -10]	0.001854	0.001979	0.002414	0.004916	0.015303	0.251445
GB	[137.834, -10.7, 153.513336, -28.17]	0.001879	0.002173	0.002353	0.002825	0.006715	0.052054
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001962	0.001877	0.001961	0.001933	0.002301	0.003733
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001865	0.001875	0.001961	0.002202	0.061779	0.00308
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001872	0.002161	0.002584	0.001894	0.001988	0.002384
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001852	0.00186	0.00197	0.002237	0.002329	0.002658
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001867	0.002167	0.002254	0.002052	0.001996	0.002222

Table 4: Get Points as GeoJSON – Time (s) – Worldwide

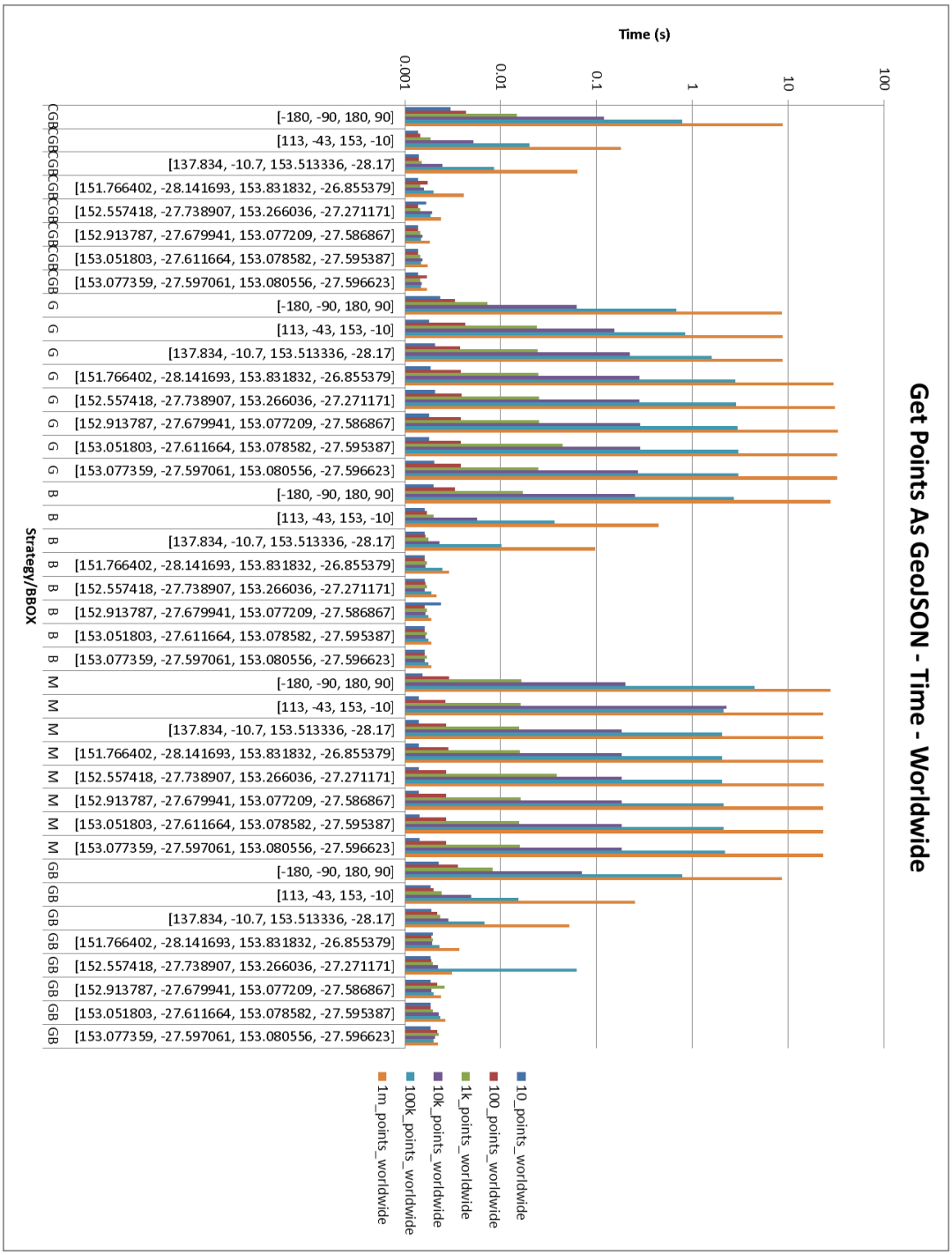

Figure 65: Get Points as GeoJSON – Time (s) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001756	0.005111	0.012717	0.045682	0.381206	8.212023
CGB	[113, -43, 153, -10]	0.001789	0.001486	0.002171	0.008108	0.024419	1.47939
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001391	0.001413	0.001629	0.003142	0.012729	0.044738
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001417	0.00146	0.00146	0.001572	0.002118	0.004483
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001395	0.001384	0.001482	0.00157	0.001787	0.002451
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001389	0.001735	0.001472	0.001548	0.001497	0.001827
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001372	0.001384	0.001479	0.001539	0.00148	0.001758
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001385	0.001396	0.001463	0.001504	0.001469	0.001755
G	[-180, -90, 180, 90]	0.001988	0.004018	0.008833	0.067822	0.752348	8.649185
G	[113, -43, 153, -10]	0.001999	0.005358	0.037219	0.236513	0.900133	8.291716
G	[137.834, -10.7, 153.513336, -28.17]	0.001995	0.005399	0.04	0.378351	1.672829	9.267905
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001995	0.00564	0.040829	0.451952	4.912801	44.6013
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.002003	0.005627	0.040916	0.457178	5.028052	51.88854
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001987	0.005986	0.040458	0.4609	5.187717	55.07505
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001978	0.005599	0.041569	0.456315	5.18681	54.22458
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001972	0.005608	0.040939	0.454103	5.176556	54.65801
B	[-180, -90, 180, 90]	0.002013	0.005555	0.032722	0.38959	4.270437	46.00326
B	[113, -43, 153, -10]	0.002029	0.001773	0.002344	0.008885	0.069972	0.780336
B	[137.834, -10.7, 153.513336, -28.17]	0.001624	0.001969	0.001821	0.002933	0.016317	0.138788
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001962	0.001626	0.001708	0.001631	0.002115	0.003335
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.002512	0.001962	0.001715	0.001624	0.002008	0.002201
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.002059	0.001636	0.001711	0.001638	0.00177	0.001912
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.002484	0.001632	0.001714	0.00163	0.001771	0.001917
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001607	0.002461	0.0017	0.00164	0.001685	0.001928
M	[-180, -90, 180, 90]	0.001615	0.005132	0.032932	0.421163	4.195377	45.2246
M	[113, -43, 153, -10]	0.001589	0.004522	0.031938	0.375898	4.141046	45.80649
M	[137.834, -10.7, 153.513336, -28.17]	0.002078	0.004571	0.031389	0.368004	4.213317	44.85817
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001592	0.00448	0.031414	0.369454	4.257227	45.23047
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001609	0.00449	0.032116	0.368724	4.229111	44.7858
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001898	0.004581	0.031832	0.352313	4.145214	44.91813
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001585	0.004766	0.030967	0.352599	4.313476	45.32323
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.00189	0.00448	0.031588	0.391198	4.28408	44.50327
GB	[-180, -90, 180, 90]	0.002324	0.005003	0.009597	0.068079	0.787407	8.85413
GB	[113, -43, 153, -10]	0.001867	0.00204	0.002704	0.006567	0.016046	0.238289
GB	[137.834, -10.7, 153.513336, -28.17]	0.001879	0.001881	0.00212	0.003892	0.008971	0.032798
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001897	0.002504	0.00229	0.001961	0.002438	0.004011
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001881	0.001891	0.001978	0.001909	0.002317	0.002976
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001872	0.001866	0.001957	0.002235	0.002347	0.002786
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001875	0.001881	0.001979	0.001901	0.001998	0.002332
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001877	0.001892	0.001968	0.002262	0.00237	0.002691

Table 5: Get Points as GeoJSON String – Time (s) – Worldwide

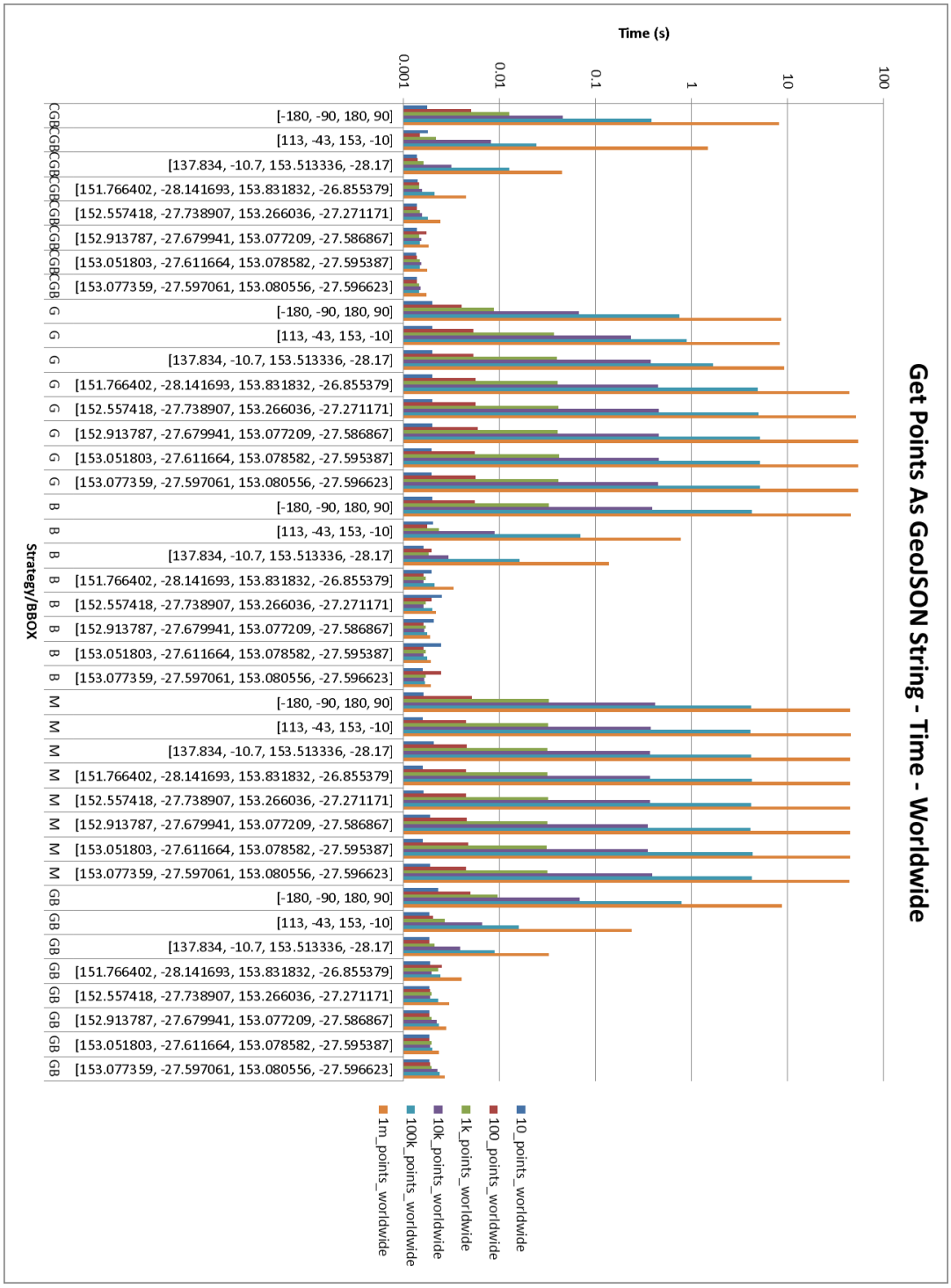

Figure 66: Get Points as GeoJSON String – Time (s) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	1450	11876	36380	42278	42725	43007
CGB	[113, -43, 153, -10]	45	325	2579	23564	46541	46401
CGB	[137.834, -10.7, 153.513336, -28.17]	45	45	325	5397	32419	40721
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	45	45	45	45	887	4408
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	45	45	607	466
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	45
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45
G	[-180, -90, 180, 90]	1310	8353	14749	15013	15121	15234
G	[113, -43, 153, -10]	1450	13850	126748	613893	730357	735505
G	[137.834, -10.7, 153.513336, -28.17]	1450	13990	137722	1147739	3301900	3384428
G	[151.766402, -28.141693, 153.831832, -26.855379]	1450	14129	140817	1404403	13767556	1.14E+08
G	[152.557418, -27.738907, 153.266036, -27.271171]	1450	14129	140817	1407510	14041534	1.37E+08
G	[152.913787, -27.679941, 153.077209, -27.586867]	1450	14129	140817	1407932	14079827	1.41E+08
G	[153.051803, -27.611664, 153.078582, -27.595387]	1450	14129	140817	1407932	14079827	1.41E+08
G	[153.077359, -27.597061, 153.080556, -27.596623]	1450	14129	140817	1407932	14079827	1.41E+08
B	[-180, -90, 180, 90]	1450	14129	140817	1407932	14079827	1.41E+08
B	[113, -43, 153, -10]	45	464	2860	31726	292429	2854067
B	[137.834, -10.7, 153.513336, -28.17]	45	45	325	5678	65075	580737
B	[151.766402, -28.141693, 153.831832, -26.855379]	45	45	45	45	887	4690
B	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	45	45	607	466
B	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	45
B	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
B	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45
M	[-180, -90, 180, 90]	1450	14129	140817	1407932	14079827	1.41E+08
M	[113, -43, 153, -10]	1450	14129	140817	1407932	14079827	1.41E+08
M	[137.834, -10.7, 153.513336, -28.17]	1450	14129	140817	1407932	14079827	1.41E+08
M	[151.766402, -28.141693, 153.831832, -26.855379]	1450	14129	140817	1407932	14079827	1.41E+08
M	[152.557418, -27.738907, 153.266036, -27.271171]	1450	14129	140817	1407932	14079827	1.41E+08
M	[152.913787, -27.679941, 153.077209, -27.586867]	1450	14129	140817	1407932	14079827	1.41E+08
M	[153.051803, -27.611664, 153.078582, -27.595387]	1450	14129	140817	1407932	14079827	1.41E+08
M	[153.077359, -27.597061, 153.080556, -27.596623]	1450	14129	140817	1407932	14079827	1.41E+08
GB	[-180, -90, 180, 90]	1310	8353	14750	15013	15120	15234
GB	[113, -43, 153, -10]	45	325	2436	14118	16897	17156
GB	[137.834, -10.7, 153.513336, -28.17]	45	45	325	5114	16235	18322
GB	[151.766402, -28.141693, 153.831832, -26.855379]	45	45	45	45	887	3844
GB	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	45	45	607	466
GB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	45
GB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
GB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45

Table 6: Get Points as GeoJSON String – String Length (Characters) – Worldwide

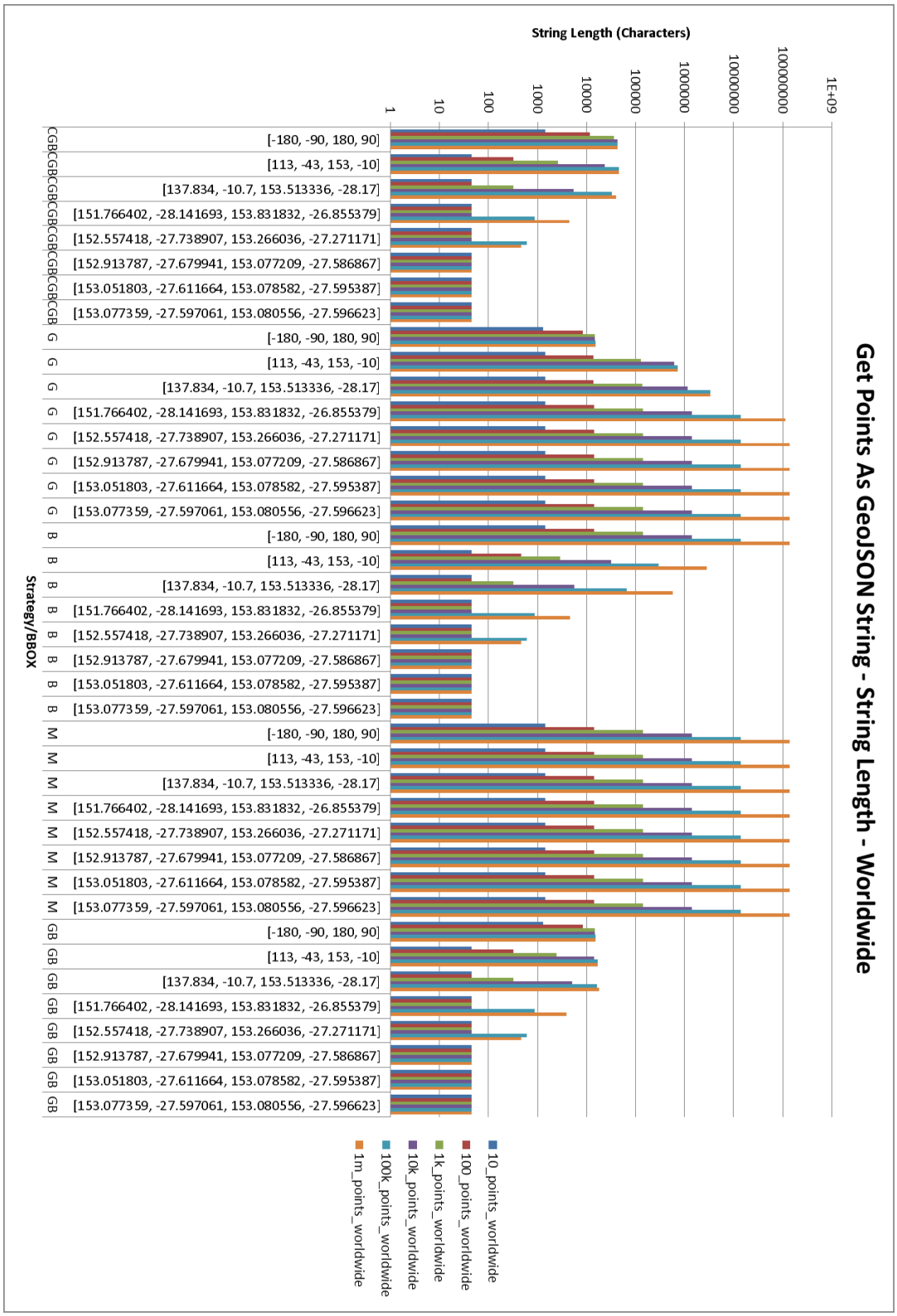

Figure 67: Get Points as GeoJSON String – String Length (Characters) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.00152	0.002639	0.007998	0.045267	0.38894	3.771646
CGB	[113, -43, 153, -10]	0.001344	0.001403	0.001778	0.004798	0.419282	0.183889
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001347	0.001358	0.001538	0.002384	0.008084	0.03855
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001345	0.001372	0.001761	0.001539	0.001947	0.003804
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001332	0.001346	0.001431	0.001532	0.001627	0.002343
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.00134	0.001328	0.001428	0.001502	0.001886	0.002928
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001352	0.001339	0.001423	0.001489	0.001445	0.001617
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001336	0.001328	0.001787	0.001471	0.001434	0.00162
G	[-180, -90, 180, 90]	0.001745	0.00294	0.006649	0.060084	0.676475	8.281416
G	[113, -43, 153, -10]	0.001759	0.00365	0.021941	0.156057	0.824073	8.202208
G	[137.834, -10.7, 153.513336, -28.17]	0.001742	0.003844	0.023452	0.212095	1.184085	8.752359
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001794	0.003677	0.023783	0.246697	2.701893	25.99111
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001772	0.003669	0.023813	0.248395	2.79532	29.16699
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.00175	0.003657	0.024136	0.25725	2.946921	32.0516
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001735	0.003659	0.024447	0.256869	3.036145	31.82806
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001716	0.003662	0.024037	0.256864	2.943902	31.66798
B	[-180, -90, 180, 90]	0.001751	0.002924	0.015449	0.183122	2.025757	23.74109
B	[113, -43, 153, -10]	0.001575	0.001692	0.001947	0.004827	0.035016	0.42328
B	[137.834, -10.7, 153.513336, -28.17]	0.001565	0.001587	0.001713	0.002204	0.008646	0.071528
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001586	0.001577	0.001665	0.001593	0.001912	0.002683
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001586	0.001589	0.001661	0.001574	0.001859	0.002076
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.002472	0.001571	0.001658	0.001589	0.001744	0.001852
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001586	0.001584	0.001653	0.001593	0.001718	0.001871
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001577	0.001582	0.001671	0.001598	0.001734	0.001862
M	[-180, -90, 180, 90]	0.001362	0.002562	0.015128	0.183469	1.96373	22.10462
M	[113, -43, 153, -10]	0.001346	0.002512	0.014946	0.173883	1.957898	22.25116
M	[137.834, -10.7, 153.513336, -28.17]	0.001347	0.002533	0.015007	0.172685	1.997204	22.43995
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001392	0.002501	0.014925	0.172886	1.949983	22.30589
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.00136	0.002528	0.015276	0.173228	2.003085	22.12166
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001365	0.002527	0.01481	0.197079	1.957009	22.28051
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001341	0.002494	0.014783	0.195033	1.959871	22.33882
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001338	0.002527	0.014839	0.19331	1.976285	22.80578
GB	[-180, -90, 180, 90]	0.0021	0.003325	0.007403	0.068622	0.816214	9.228245
GB	[113, -43, 153, -10]	0.001833	0.001927	0.002345	0.004963	0.013357	0.218927
GB	[137.834, -10.7, 153.513336, -28.17]	0.002123	0.001853	0.002026	0.003159	0.006465	0.429302
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001842	0.001842	0.001943	0.001859	0.002221	0.003376
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001851	0.001841	0.00192	0.001853	0.002302	0.002599
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.002099	0.001848	0.001913	0.00186	0.001956	0.002317
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001826	0.001852	0.001923	0.001841	0.001949	0.002255
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.002519	0.001831	0.001917	0.001885	0.001947	0.002292

Table 7: Get Points as WKT – Time (s) – Worldwide

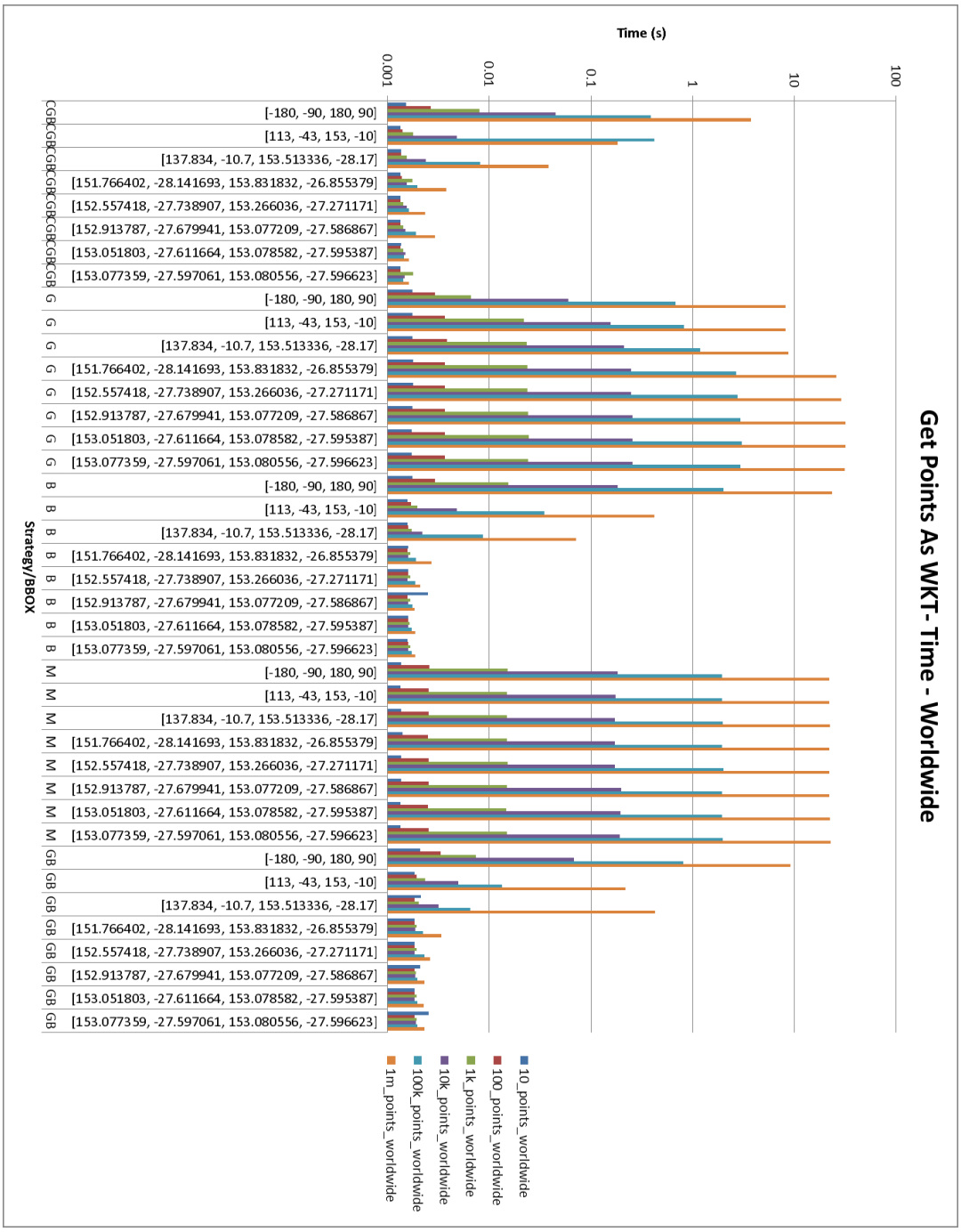

Figure 68: Get Points as WKT – Time (s) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001547	0.002677	0.008048	0.039837	0.301053	3.680209
CGB	[113, -43, 153, -10]	0.001333	0.001428	0.001801	0.004882	0.014881	0.121797
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001352	0.001357	0.001561	0.002401	0.009016	0.03883
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001348	0.001352	0.001426	0.001542	0.001995	0.003773
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001338	0.001342	0.001442	0.001518	0.001674	0.002327
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001355	0.001362	0.001806	0.001536	0.001463	0.0018
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001332	0.001357	0.001433	0.001486	0.001851	0.002164
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.00134	0.001463	0.001427	0.001477	0.001425	0.0017
G	[-180, -90, 180, 90]	0.001768	0.00296	0.006715	0.067981	0.681083	8.231727
G	[113, -43, 153, -10]	0.001757	0.003685	0.022191	0.187066	0.793564	8.171559
G	[137.834, -10.7, 153.513336, -28.17]	0.001874	0.003697	0.023659	0.218105	1.194787	8.75534
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001778	0.003746	0.024137	0.253084	2.75161	26.65866
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001775	0.003738	0.024387	0.252881	2.809712	30.08876
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001769	0.003707	0.024499	0.263536	2.951132	32.9429
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001735	0.003731	0.024473	0.287419	2.964372	32.96984
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001752	0.003727	0.024173	0.286296	2.956724	32.80911
B	[-180, -90, 180, 90]	0.001749	0.00298	0.016166	0.192396	2.170993	24.1821
B	[113, -43, 153, -10]	0.001542	0.001657	0.001949	0.004954	0.035114	0.434552
B	[137.834, -10.7, 153.513336, -28.17]	0.001575	0.001604	0.002009	0.002234	0.009469	0.073862
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001572	0.001577	0.001664	0.001858	0.001931	0.002733
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.002492	0.001588	0.001949	0.001581	0.002112	0.002313
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001589	0.001591	0.001668	0.001871	0.001715	0.001871
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001582	0.001593	0.001669	0.001865	0.001741	0.001859
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001566	0.00159	0.001952	0.0016	0.001998	0.00213
M	[-180, -90, 180, 90]	0.001374	0.002619	0.0155	0.194701	1.992243	23.39067
M	[113, -43, 153, -10]	0.001358	0.002591	0.01549	0.202229	2.065112	22.51437
M	[137.834, -10.7, 153.513336, -28.17]	0.001361	0.002599	0.015493	0.201494	2.023461	23.33137
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001363	0.002576	0.015699	0.200151	2.06733	23.16482
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001389	0.00258	0.015337	0.198569	2.022843	23.42156
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001385	0.002604	0.015282	0.198527	2.069489	23.35058
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001355	0.002569	0.015361	0.19888	1.977762	23.15853
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001364	0.002588	0.015281	0.179091	2.064203	23.46284
GB	[-180, -90, 180, 90]	0.002484	0.003374	0.007478	0.065722	0.763635	8.94024
GB	[113, -43, 153, -10]	0.001849	0.001942	0.002374	0.005395	0.013422	0.210053
GB	[137.834, -10.7, 153.513336, -28.17]	0.001855	0.001852	0.002041	0.002861	0.006658	0.02762
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.002123	0.00183	0.001924	0.002044	0.002253	0.003409
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001862	0.001901	0.001851	0.001862	0.002163	0.002519
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001846	0.001842	0.001925	0.001851	0.001959	0.002299
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001838	0.001846	0.001935	0.001864	0.001963	0.002271
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001823	0.00184	0.001842	0.001862	0.001945	0.002301

Table 8: Get Points as WKT String – Time (s) – Worldwide

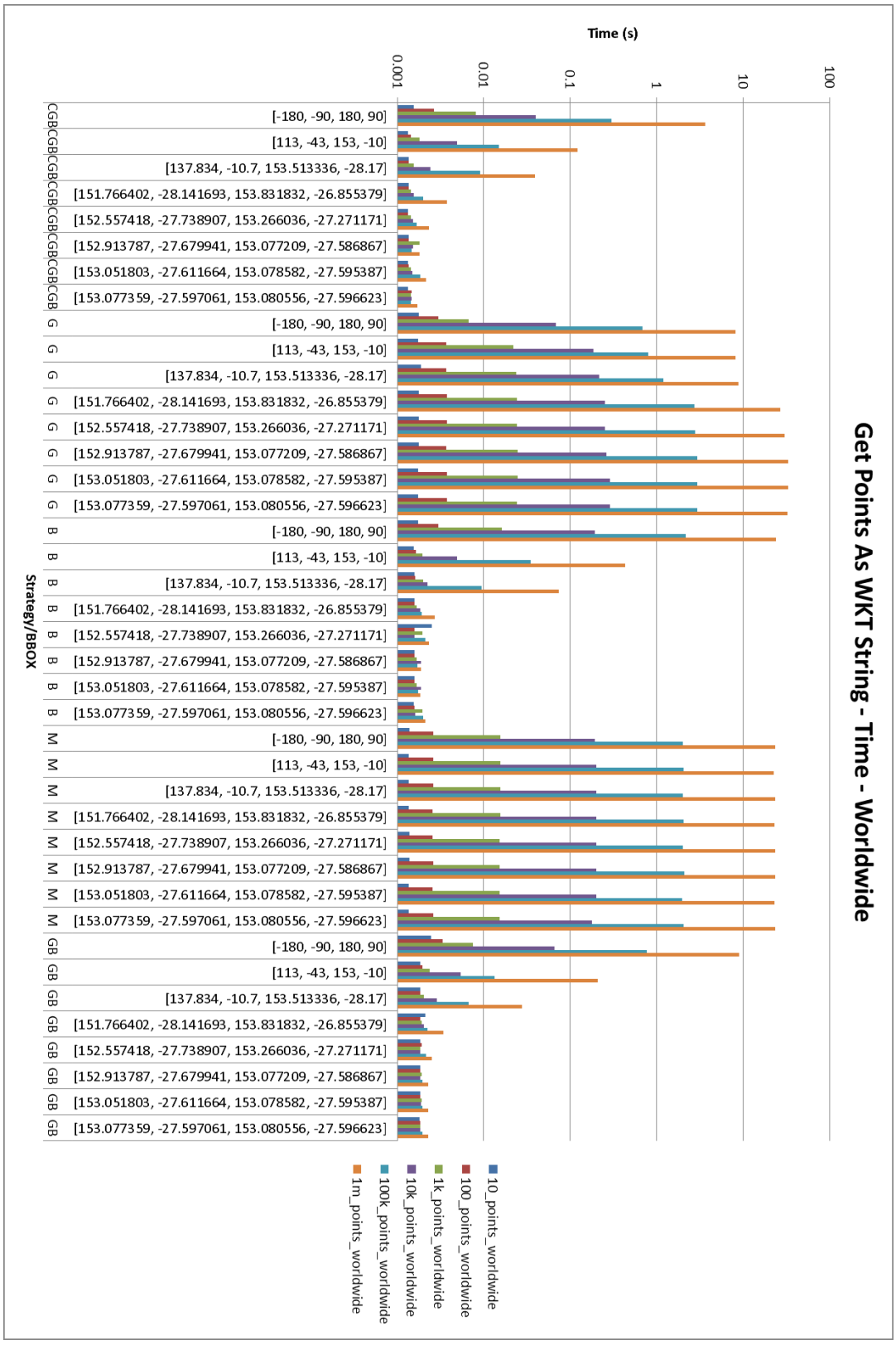

Figure 69: Get Points as WKT String – Time (s) – Worldwide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	436	3536	10816	12499	12558	12566
CGB	[113, -43, 153, -10]	20	103	773	7007	13811	13677
CGB	[137.834, -10.7, 153.513336, -28.17]	20	20	103	1611	9625	12010
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	20	20	20	20	269	1315
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	20	20	187	145
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	20
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20
G	[-180, -90, 180, 90]	395	2488	4377	4425	4435	4453
G	[113, -43, 153, -10]	436	4124	37626	182241	215444	215532
G	[137.834, -10.7, 153.513336, -28.17]	436	4165	40878	340691	980284	997910
G	[151.766402, -28.141693, 153.831832, -26.855379]	436	4205	41795	416874	4087279	33882792
G	[152.557418, -27.738907, 153.266036, -27.271171]	436	4205	41795	417803	4168603	40690842
G	[152.913787, -27.679941, 153.077209, -27.586867]	436	4205	41795	417928	4179969	41797983
G	[153.051803, -27.611664, 153.078582, -27.595387]	436	4205	41795	417928	4179969	41797983
G	[153.077359, -27.597061, 153.080556, -27.596623]	436	4205	41795	417928	4179969	41797983
B	[-180, -90, 180, 90]	436	4205	41795	417928	4179969	41797983
B	[113, -43, 153, -10]	20	143	856	9427	86782	847016
B	[137.834, -10.7, 153.513336, -28.17]	20	20	103	1694	19313	172338
B	[151.766402, -28.141693, 153.831832, -26.855379]	20	20	20	20	269	1399
B	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	20	20	187	145
B	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	20
B	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
B	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20
M	[-180, -90, 180, 90]	436	4205	41795	417928	4179969	41797983
M	[113, -43, 153, -10]	436	4205	41795	417928	4179969	41797983
M	[137.834, -10.7, 153.513336, -28.17]	436	4205	41795	417928	4179969	41797983
M	[151.766402, -28.141693, 153.831832, -26.855379]	436	4205	41795	417928	4179969	41797983
M	[152.557418, -27.738907, 153.266036, -27.271171]	436	4205	41795	417928	4179969	41797983
M	[152.913787, -27.679941, 153.077209, -27.586867]	436	4205	41795	417928	4179969	41797983
M	[153.051803, -27.611664, 153.078582, -27.595387]	436	4205	41795	417928	4179969	41797983
M	[153.077359, -27.597061, 153.080556, -27.596623]	436	4205	41795	417928	4179969	41797983
GB	[-180, -90, 180, 90]	395	2488	4378	4426	4434	4453
GB	[113, -43, 153, -10]	20	103	729	4194	4992	5034
GB	[137.834, -10.7, 153.513336, -28.17]	20	20	103	1526	4822	5416
GB	[151.766402, -28.141693, 153.831832, -26.855379]	20	20	20	20	269	1147
GB	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	20	20	187	145
GB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	20
GB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
GB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20

Table 9: Get Points as WKT String – String Length (Characters) – Worldwide

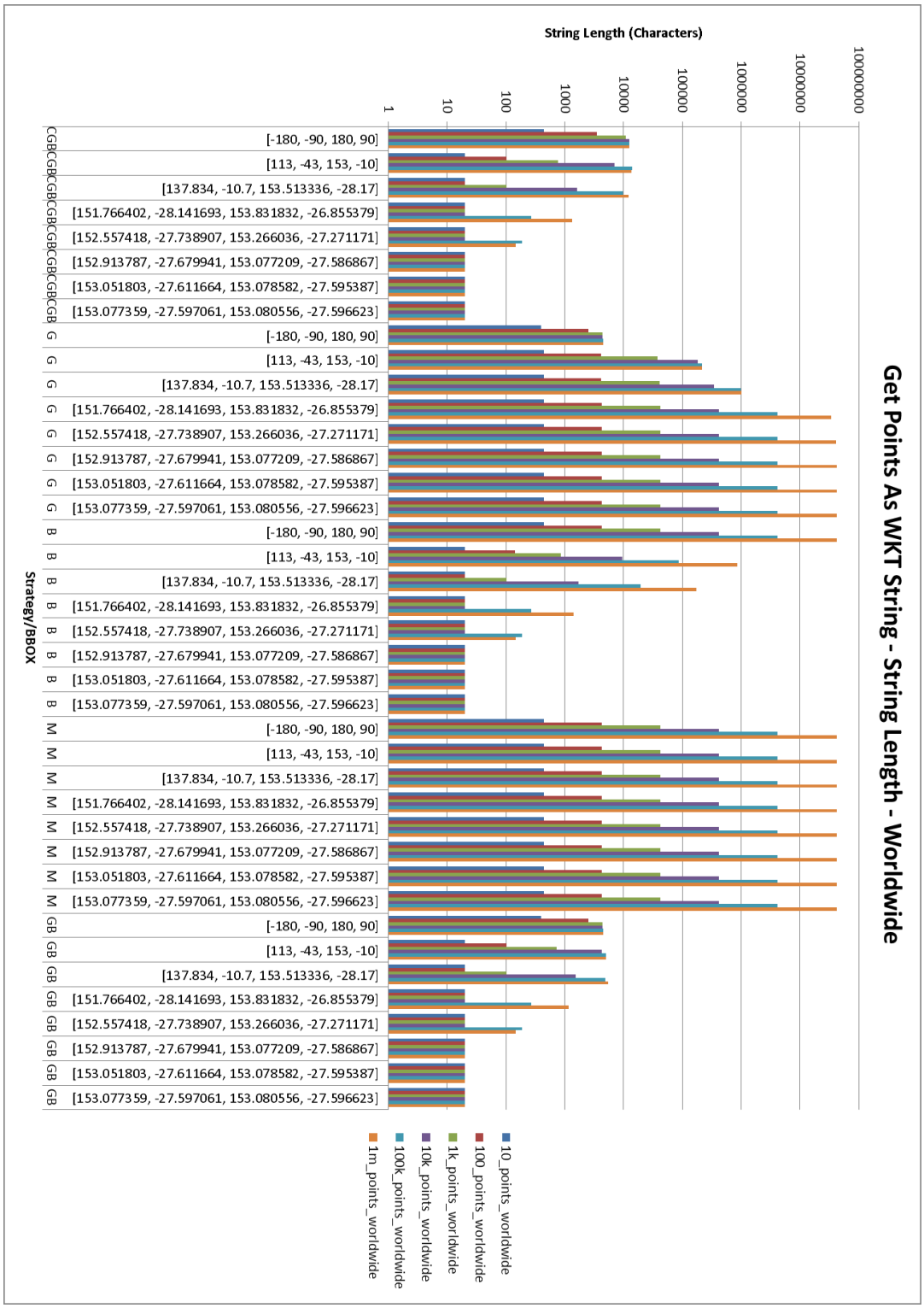

Figure 70: Get Points as WKT String – String Length (Characters) – Worldwide

8.2.2 Australia Wide

Strategy	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	0.217658	1.13251	8.332264	70.77112	558.3035	4948.624
G	1.40E-05	1.40E-05	1.50E-05	1.50E-05	1.40E-05	1.60E-05
B	1.50E-05	1.40E-05	1.50E-05	1.40E-05	1.90E-05	1.50E-05
M	1.40E-05	1.40E-05	1.50E-05	1.40E-05	1.50E-05	1.50E-05
GB	1.40E-05	1.50E-05	1.40E-05	1.40E-05	1.40E-05	1.50E-05

Table 10: Pre-Processing – Time (s) – Australia Wide

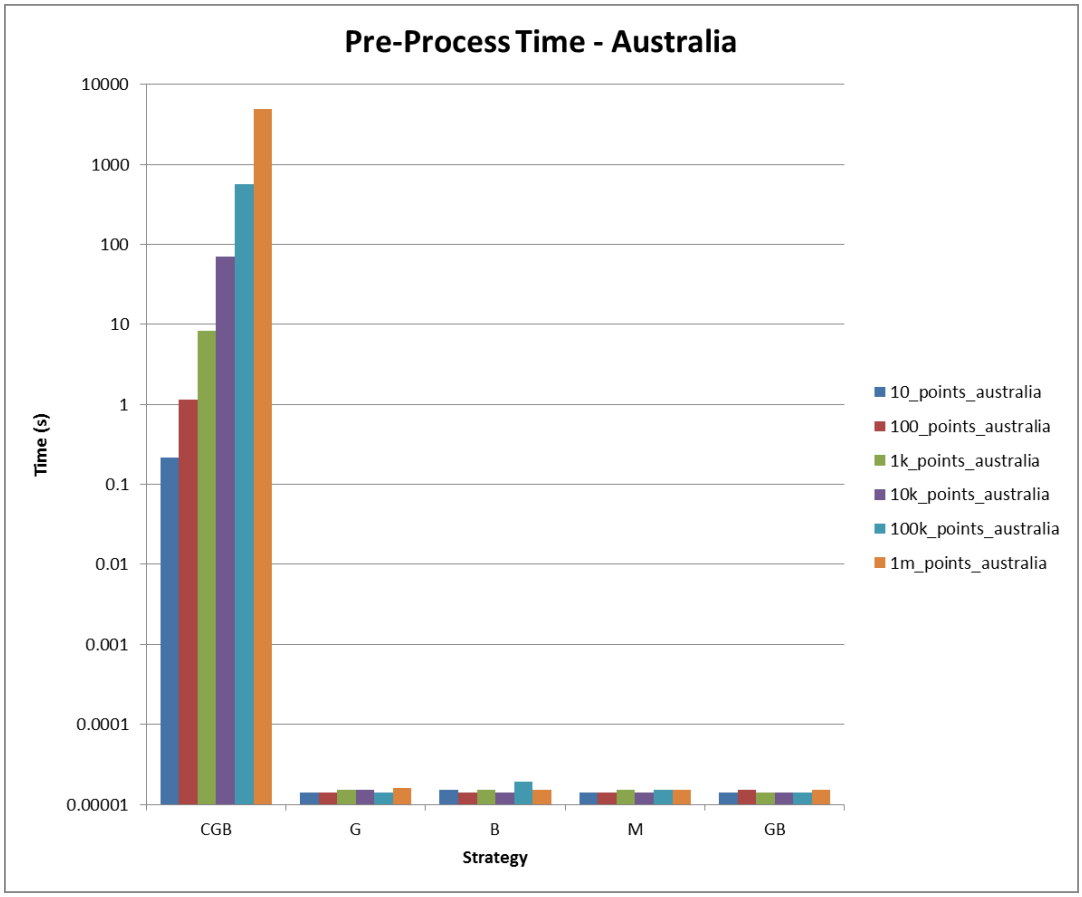

Figure 71 : Pre-Processing Time – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	5	10	12	12	12	12
CGB	[113, -43, 153, -10]	10	86	319	340	340	340
CGB	[137.834, -10.7, 153.513336, -28.17]	2	23	142	286	288	288
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0	1	2	14	72	106
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	1	0	9	104
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	5
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0
G	[-180, -90, 180, 90]	3	7	9	9	9	9
G	[113, -43, 153, -10]	10	70	117	120	120	120
G	[137.834, -10.7, 153.513336, -28.17]	10	93	428	525	525	525
G	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	985	8984	41398	47080
G	[152.557418, -27.738907, 153.266036, -27.271171]	10	100	1000	9882	87837	352619
G	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	1000000
G	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	1000000
G	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	1000000
B	[-180, -90, 180, 90]	10	100	1000	10000	100000	1000000
B	[113, -43, 153, -10]	10	100	1000	10000	100000	1000000
B	[137.834, -10.7, 153.513336, -28.17]	2	23	214	2043	20169	200593
B	[151.766402, -28.141693, 153.831832, -26.855379]	0	1	2	14	119	1158
B	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	1	0	9	131
B	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	5
B	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
B	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0
M	[-180, -90, 180, 90]	10	100	1000	10000	100000	1000000
M	[113, -43, 153, -10]	10	100	1000	10000	100000	1000000
M	[137.834, -10.7, 153.513336, -28.17]	10	100	1000	10000	100000	1000000
M	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	1000	10000	100000	1000000
M	[152.557418, -27.738907, 153.266036, -27.271171]	10	100	1000	10000	100000	1000000
M	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	1000000
M	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	1000000
M	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	1000000
GB	[-180, -90, 180, 90]	3	7	9	9	9	9
GB	[113, -43, 153, -10]	10	70	117	120	120	120
GB	[137.834, -10.7, 153.513336, -28.17]	2	22	89	116	120	120
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0	1	2	14	57	73
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0	0	1	0	9	61
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	0	0	0	5
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	0	0	0
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	0

Table 11: Clusters Generated – Australia Wide

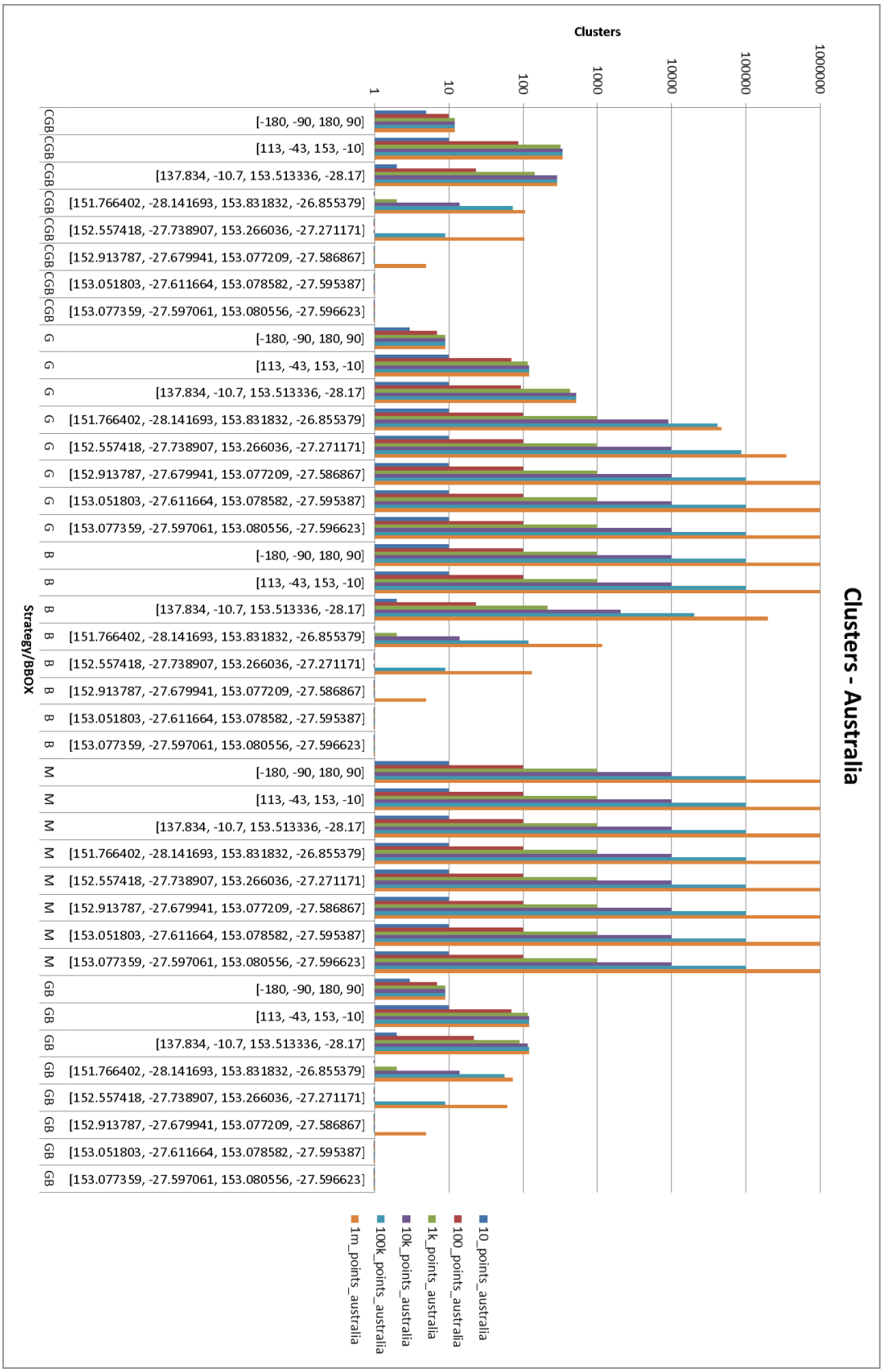

Figure 72: Clusters Generated – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.003194	0.003853	0.007337	0.038424	0.244738	2.653093
CGB	[113, -43, 153, -10]	0.001565	0.003723	0.011602	0.064576	0.536406	6.182612
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001476	0.002035	0.004421	0.013279	0.049116	0.390442
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001364	0.001485	0.001648	0.002242	0.003932	0.013909
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001774	0.001355	0.001489	0.001446	0.001861	0.005489
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001352	0.001376	0.001911	0.001738	0.001455	0.002182
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001357	0.001353	0.00139	0.001402	0.001394	0.002331
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001353	0.001386	0.001372	0.00137	0.00138	0.001584
G	[-180, -90, 180, 90]	0.002163	0.00253	0.005083	0.054366	0.663014	8.269174
G	[113, -43, 153, -10]	0.001858	0.004272	0.008163	0.067709	0.720138	8.593903
G	[137.834, -10.7, 153.513336, -28.17]	0.001825	0.003968	0.015451	0.072698	0.701859	8.038047
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001797	0.004214	0.026981	0.259988	1.629237	9.410453
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001799	0.004164	0.028049	0.275236	2.728089	17.19308
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001802	0.004029	0.027976	0.284252	3.223997	33.81173
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001799	0.004079	0.027612	0.283	3.097854	33.48979
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001804	0.004163	0.027933	0.284529	3.135012	33.32814
B	[-180, -90, 180, 90]	0.002253	0.003688	0.017937	0.269429	2.574674	24.77885
B	[113, -43, 153, -10]	0.0018	0.003153	0.017556	0.222412	2.157713	24.02525
B	[137.834, -10.7, 153.513336, -28.17]	0.001683	0.002043	0.005673	0.035561	0.394343	4.420758
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001738	0.001951	0.002227	0.002736	0.004467	0.027371
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.00161	0.00166	0.001794	0.001698	0.001837	0.004897
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001611	0.001658	0.001653	0.001616	0.00173	0.002092
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001628	0.001689	0.001638	0.001634	0.001623	0.001773
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001607	0.001794	0.001608	0.001624	0.001621	0.001831
M	[-180, -90, 180, 90]	0.001809	0.003165	0.017383	0.218901	3.306338	24.75509
M	[113, -43, 153, -10]	0.00149	0.002769	0.017696	0.191376	2.12146	23.17934
M	[137.834, -10.7, 153.513336, -28.17]	0.001409	0.002828	0.017865	0.191613	2.160407	23.20334
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001413	0.002774	0.017498	0.193106	2.106132	23.112
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001413	0.002753	0.017593	0.190426	2.152609	23.12829
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001474	0.003341	0.01739	0.190881	2.188472	23.09968
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.00142	0.002886	0.017394	0.192248	2.116066	23.16815
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001437	0.002846	0.017445	0.190603	2.112725	23.26256
GB	[-180, -90, 180, 90]	0.002534	0.003015	0.005752	0.060362	0.722508	8.680796
GB	[113, -43, 153, -10]	0.002792	0.005351	0.007967	0.069026	0.740862	8.308388
GB	[137.834, -10.7, 153.513336, -28.17]	0.002076	0.00259	0.005197	0.013829	0.141732	1.579055
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001921	0.001992	0.001979	0.002977	0.004498	0.015619
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.003084	0.002435	0.002999	0.002027	0.002468	0.004014
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001908	0.001916	0.001916	0.001936	0.002386	0.002253
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.002355	0.002397	0.002367	0.001912	0.001949	0.002009
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001935	0.001919	0.00187	0.00194	0.035001	0.002051

Table 12: Get Points as GeoJSON – Time (s) – Australia Wide

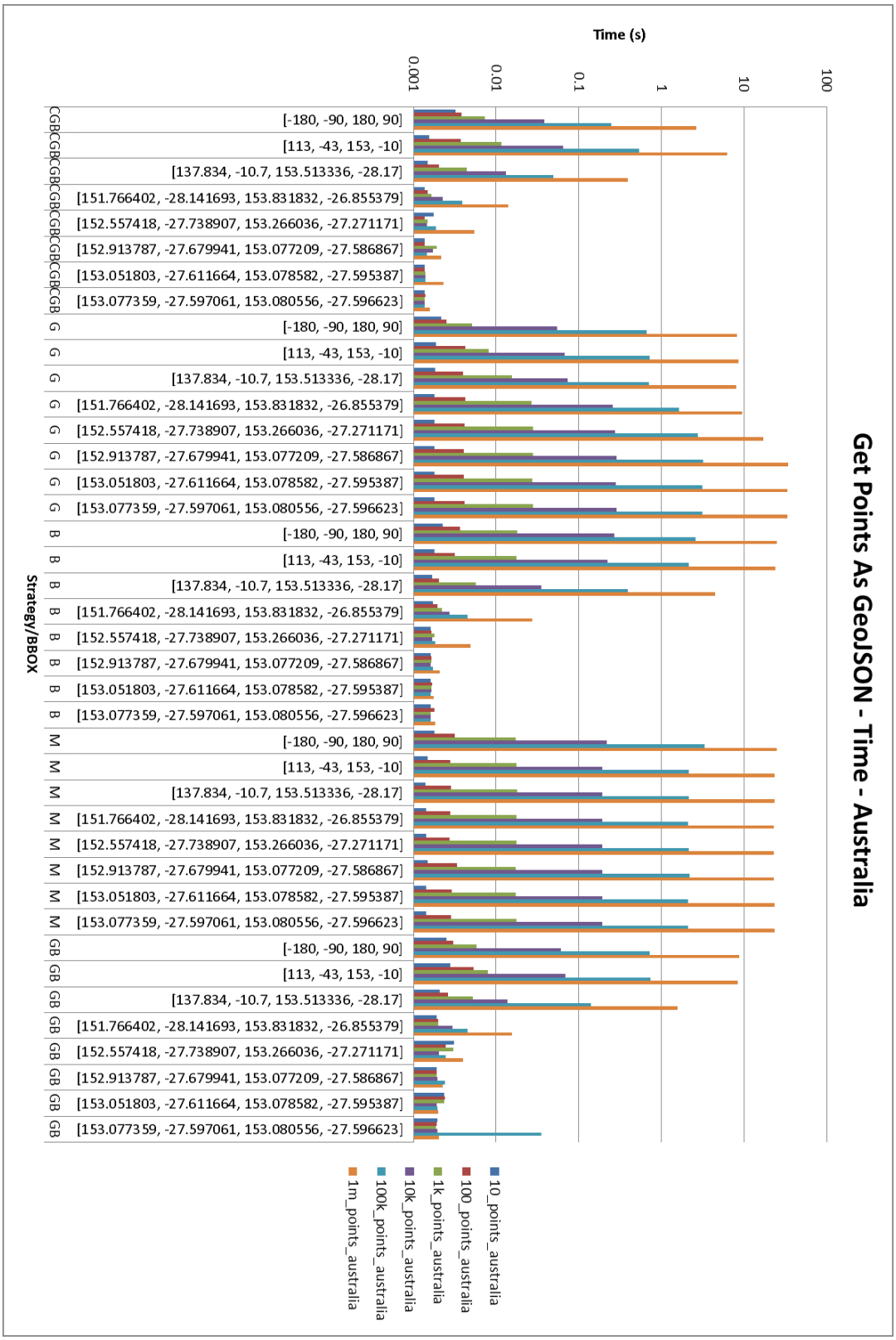

Figure 73: Get Points as GeoJSON – Time (s) – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001664	0.002076	0.004643	0.038449	0.207135	2.300839
CGB	[113, -43, 153, -10]	0.002192	0.004504	0.018128	0.251425	0.20175	2.196141
CGB	[137.834, -10.7, 153.513336, -28.17]	0.00157	0.002334	0.006562	0.017564	0.42367	4.97123
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001438	0.002079	0.001605	0.001993	0.005201	0.014601
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.00136	0.001393	0.00152	0.001431	0.002091	0.007532
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001367	0.001399	0.001467	0.001433	0.001447	0.002068
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001366	0.001382	0.00199	0.001754	0.001438	0.001612
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001435	0.001881	0.001409	0.001778	0.001419	0.001699
G	[-180, -90, 180, 90]	0.001876	0.002345	0.005063	0.053711	0.638976	8.142467
G	[113, -43, 153, -10]	0.002113	0.00485	0.010401	0.066209	0.920211	8.09474
G	[137.834, -10.7, 153.513336, -28.17]	0.002089	0.005753	0.022979	0.082361	0.720454	8.058333
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.002033	0.00632	0.044728	0.414157	2.539615	10.30022
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.002041	0.006328	0.045952	0.463905	4.739075	24.15295
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.002129	0.006456	0.044428	0.462051	5.295395	57.15148
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.002051	0.006328	0.045627	0.485239	5.321778	56.48641
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.002045	0.006372	0.045554	0.507648	5.370639	55.03475
B	[-180, -90, 180, 90]	0.002091	0.006388	0.034464	0.408681	4.327131	45.75002
B	[113, -43, 153, -10]	0.002553	0.006046	0.034899	0.407232	4.337482	44.89509
B	[137.834, -10.7, 153.513336, -28.17]	0.00182	0.002571	0.008865	0.070377	0.824307	9.115384
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.003184	0.002315	0.001904	0.002418	0.005949	0.040791
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001614	0.001654	0.002162	0.001665	0.002029	0.006438
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.002111	0.002111	0.001697	0.001638	0.002042	0.002104
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.002126	0.002245	0.001634	0.001619	0.001966	0.001825
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.00164	0.001688	0.002238	0.001619	0.001604	0.002128
M	[-180, -90, 180, 90]	0.001769	0.006245	0.036172	2.265221	4.326902	44.5899
M	[113, -43, 153, -10]	0.001661	0.004949	0.035471	0.391796	4.34527	44.49076
M	[137.834, -10.7, 153.513336, -28.17]	0.002317	0.00493	0.035516	0.391991	4.253881	45.65319
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001622	0.005028	0.034663	0.38887	4.358476	45.01312
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001656	0.004977	0.034811	0.393171	4.414159	44.31342
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.002158	0.005039	0.035024	0.363536	4.263181	63.54026
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001651	0.005003	0.034408	0.363136	4.282552	46.05018
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.002145	0.005022	0.062585	0.404107	4.337537	45.29899
GB	[-180, -90, 180, 90]	0.002174	0.00274	0.005279	0.059114	0.709778	8.634506
GB	[113, -43, 153, -10]	0.002493	0.005379	0.010646	0.071472	0.754415	8.805295
GB	[137.834, -10.7, 153.513336, -28.17]	0.002618	0.003539	0.006512	0.01573	0.146747	1.580137
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.00195	0.002053	0.002053	0.00278	0.005509	0.010557
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001925	0.001923	0.002002	0.002019	0.00251	0.005568
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.002343	0.002412	0.002379	0.001937	0.00197	0.002411
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.00192	0.001938	0.001907	0.002316	0.001942	0.002388
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.002423	0.002402	0.002223	0.001946	0.001912	0.002048

Table 13: Get Points as GeoJSON String – Time (s) – Australia Wide

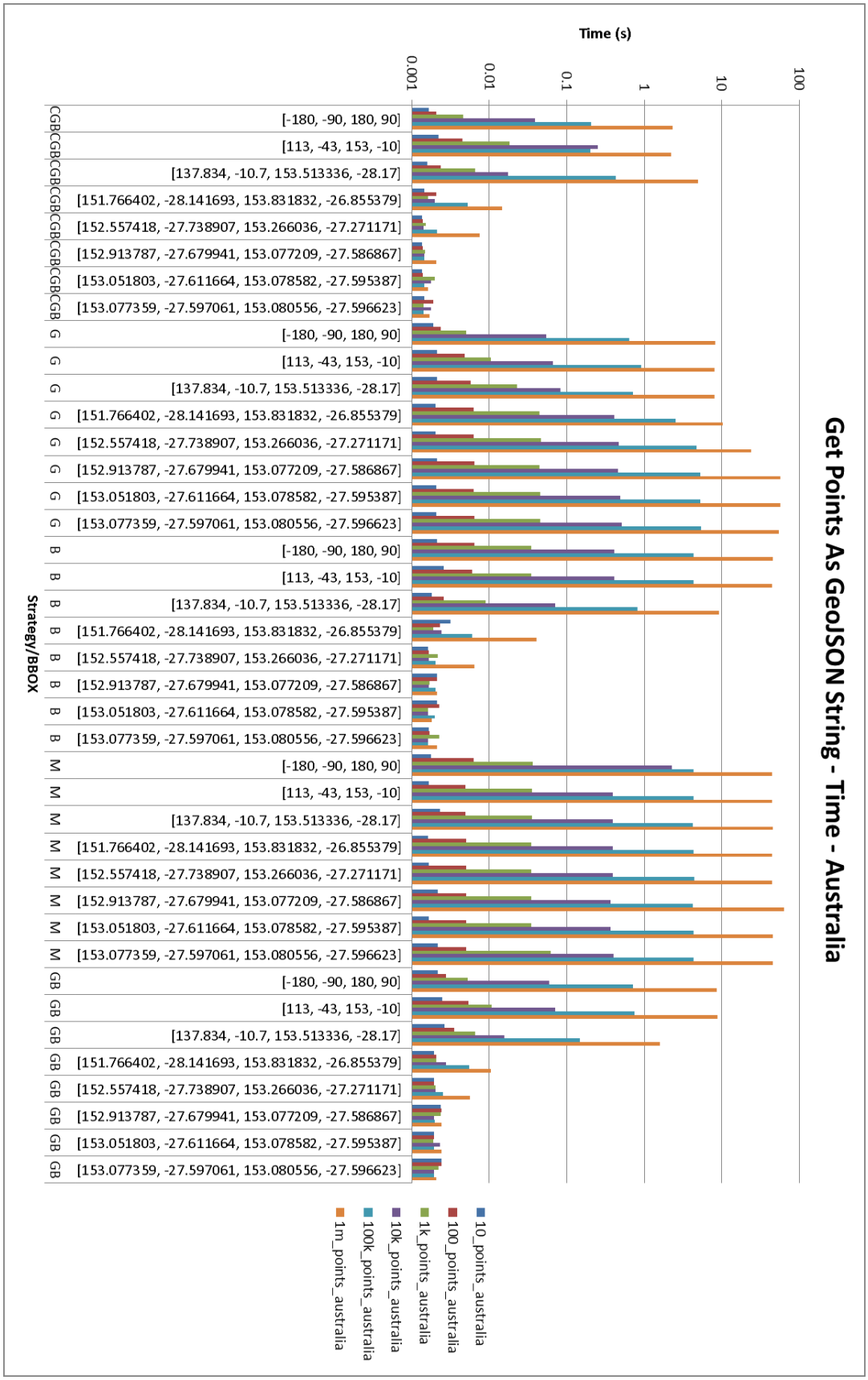

Figure 74: Get Points as GeoJSON String – Time (s) – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	747	1456	1749	1756	1773	1785
CGB	[113, -43, 153, -10]	1452	12140	44951	48241	48575	48924
CGB	[137.834, -10.7, 153.513336, -28.17]	325	3278	20041	40373	40863	41169
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	45	184	325	2012	10175	15037
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	184	45	1309	14686
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	747
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45
G	[-180, -90, 180, 90]	465	1029	1322	1329	1339	1348
G	[113, -43, 153, -10]	1452	9895	16567	17096	17208	17300
G	[137.834, -10.7, 153.513336, -28.17]	1452	13125	60295	74415	74971	75463
G	[151.766402, -28.141693, 153.831832, -26.855379]	1452	14111	138722	1264822	5828023	6674561
G	[152.557418, -27.738907, 153.266036, -27.271171]	1452	14111	140838	1391277	12365714	49641398
G	[152.913787, -27.679941, 153.077209, -27.586867]	1452	14111	140838	1407882	14077985	1.41E+08
G	[153.051803, -27.611664, 153.078582, -27.595387]	1452	14111	140838	1407882	14077985	1.41E+08
G	[153.077359, -27.597061, 153.080556, -27.596623]	1452	14111	140838	1407882	14077985	1.41E+08
B	[-180, -90, 180, 90]	1452	14111	140838	1407882	14077985	1.41E+08
B	[113, -43, 153, -10]	1452	14111	140838	1407882	14077985	1.41E+08
B	[137.834, -10.7, 153.513336, -28.17]	325	3278	30165	287667	2839483	28238993
B	[151.766402, -28.141693, 153.831832, -26.855379]	45	184	325	2012	16796	163069
B	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	184	45	1309	18490
B	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	747
B	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
B	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45
M	[-180, -90, 180, 90]	1452	14111	140838	1407882	14077985	1.41E+08
M	[113, -43, 153, -10]	1452	14111	140838	1407882	14077985	1.41E+08
M	[137.834, -10.7, 153.513336, -28.17]	1452	14111	140838	1407882	14077985	1.41E+08
M	[151.766402, -28.141693, 153.831832, -26.855379]	1452	14111	140838	1407882	14077985	1.41E+08
M	[152.557418, -27.738907, 153.266036, -27.271171]	1452	14111	140838	1407882	14077985	1.41E+08
M	[152.913787, -27.679941, 153.077209, -27.586867]	1452	14111	140838	1407882	14077985	1.41E+08
M	[153.051803, -27.611664, 153.078582, -27.595387]	1452	14111	140838	1407882	14077985	1.41E+08
M	[153.077359, -27.597061, 153.080556, -27.596623]	1452	14111	140838	1407882	14077985	1.41E+08
GB	[-180, -90, 180, 90]	465	1029	1322	1330	1338	1348
GB	[113, -43, 153, -10]	1452	9895	16567	17094	17203	17300
GB	[137.834, -10.7, 153.513336, -28.17]	325	3137	12573	16467	17148	17261
GB	[151.766402, -28.141693, 153.831832, -26.855379]	45	184	325	2012	8063	10373
GB	[152.557418, -27.738907, 153.266036, -27.271171]	45	45	184	45	1309	8633
GB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	45	45	45	747
GB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	45	45	45
GB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	45

Table 14: Get Points as GeoJSON String – String Length (Characters) – Australia Wide

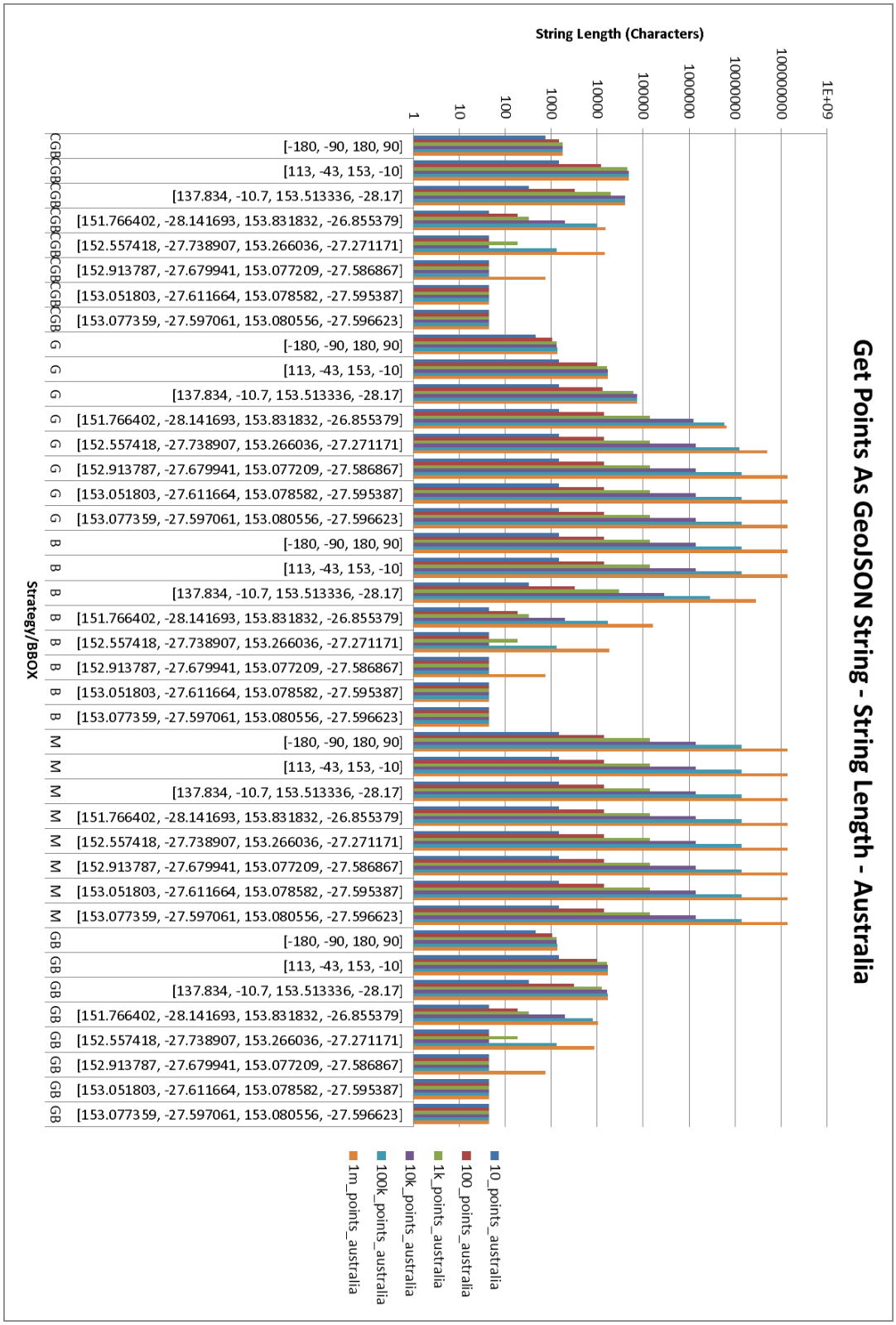

Figure 75: Get Points as GeoJSON String – String Length (Characters) – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001476	0.001803	0.004468	0.033189	0.234075	2.270617
CGB	[113, -43, 153, -10]	0.001504	0.002836	0.00891	0.037476	0.217745	1.917155
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001386	0.001774	0.003546	0.012621	0.043875	0.372366
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001342	0.001429	0.001472	0.001655	0.003197	0.011074
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001322	0.001326	0.001438	0.001395	0.001793	0.004709
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001334	0.001352	0.00139	0.001387	0.002005	0.001935
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001338	0.001313	0.001375	0.001331	0.001382	0.001539
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001322	0.001342	0.00135	0.001696	0.001363	0.001546
G	[-180, -90, 180, 90]	0.001744	0.002077	0.004779	0.055447	0.620321	7.976591
G	[113, -43, 153, -10]	0.001793	0.003505	0.007605	0.062711	0.691331	8.076125
G	[137.834, -10.7, 153.513336, -28.17]	0.001788	0.003987	0.01446	0.071266	0.724163	8.031563
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001735	0.003906	0.025528	0.314492	1.582076	9.209822
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001756	0.003872	0.025854	0.2549	2.599333	16.30489
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001752	0.003883	0.053005	0.302074	3.002938	31.8606
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001738	0.003918	0.025576	0.295801	2.966958	32.12085
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001744	0.003822	0.025844	0.29549	3.018788	31.87385
B	[-180, -90, 180, 90]	0.001816	0.003204	0.016081	0.183708	2.055438	22.71157
B	[113, -43, 153, -10]	0.001745	0.003132	0.015935	0.185975	2.018943	23.20722
B	[137.834, -10.7, 153.513336, -28.17]	0.001627	0.00198	0.004756	0.060151	0.398816	4.425289
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.0016	0.001945	0.001712	0.00184	0.003366	0.020736
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001566	0.0016	0.001664	0.001622	0.001794	0.003892
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.00157	0.001613	0.00162	0.00159	0.0016	0.001955
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001597	0.001616	0.001568	0.001558	0.00156	0.001769
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001568	0.001603	0.001574	0.001586	0.00155	0.001783
M	[-180, -90, 180, 90]	0.001388	0.002626	0.016255	0.19704	2.02125	22.48481
M	[113, -43, 153, -10]	0.001389	0.002593	0.016279	0.179855	2.074094	22.1042
M	[137.834, -10.7, 153.513336, -28.17]	0.001338	0.002576	0.043231	0.178035	2.010736	22.03684
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001341	0.002598	0.016559	0.178581	2.096865	22.16089
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001367	0.00259	0.016352	0.180022	2.022357	22.53324
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001437	0.002927	0.015915	0.209914	2.013522	22.70933
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001415	0.002682	0.016518	0.209106	1.990682	22.11977
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001359	0.002589	0.016319	0.204677	2.050752	22.23218
GB	[-180, -90, 180, 90]	0.002034	0.002517	0.005015	0.058995	0.699151	8.475151
GB	[113, -43, 153, -10]	0.002248	0.003707	0.007993	0.071377	0.750716	8.687756
GB	[137.834, -10.7, 153.513336, -28.17]	0.001989	0.002477	0.004533	0.013591	0.142509	1.575546
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001876	0.001951	0.001933	0.002317	0.00378	0.008989
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001868	0.001902	0.002108	0.00193	0.002215	0.00388
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001865	0.00188	0.001875	0.002306	0.001898	0.002234
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001851	0.001889	0.001844	0.002896	0.001896	0.002028
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001902	0.001885	0.001841	0.002375	0.00203	0.002007

Table 15: Get Points as WKT – Time (s) – Australia Wide

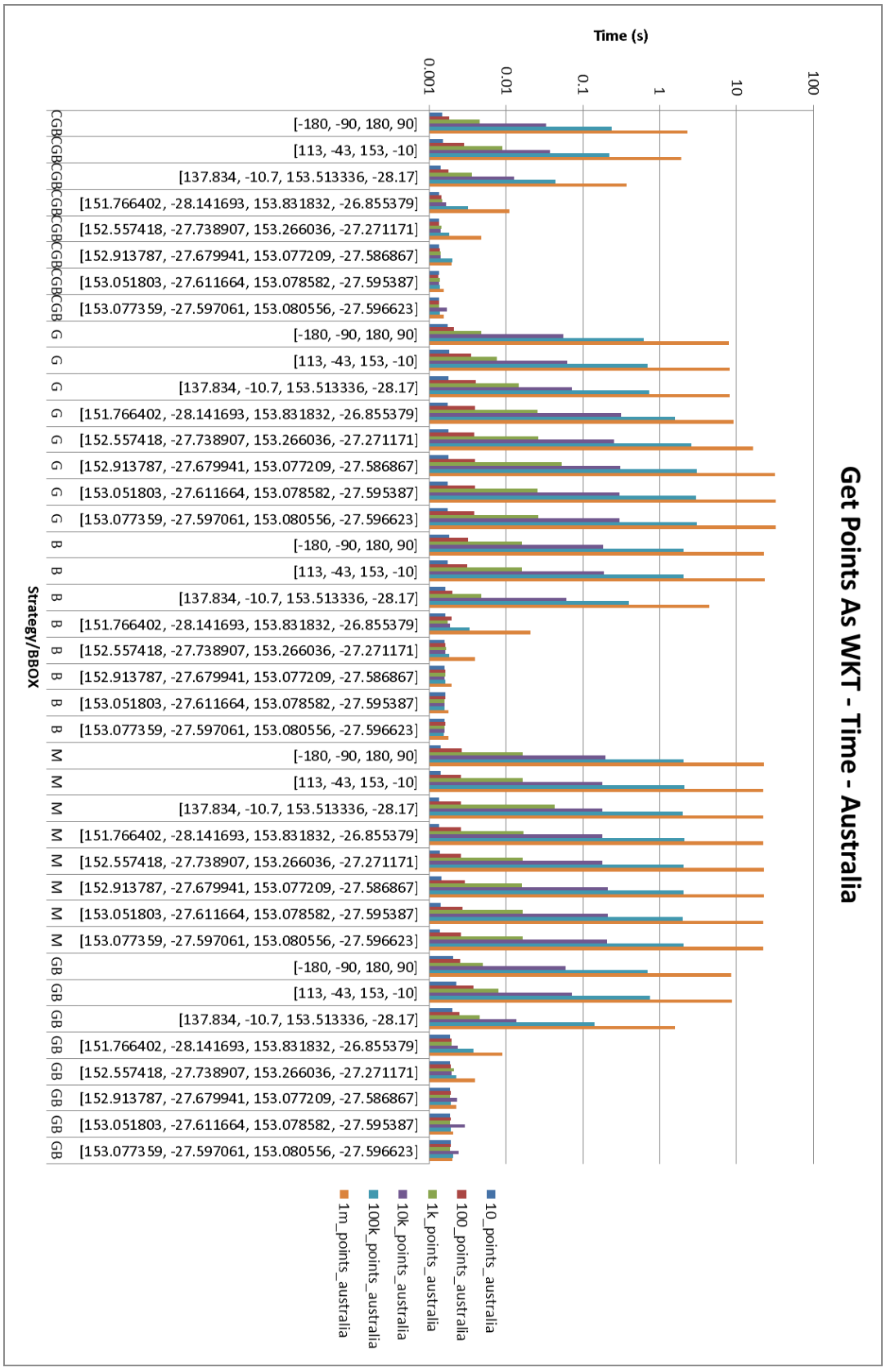

Figure 76: Get Points as WKT – Time (s) – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001462	0.001848	0.004191	0.034755	0.193726	2.26707
CGB	[113, -43, 153, -10]	0.001491	0.003095	0.009235	0.036626	0.193061	1.978238
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001422	0.001855	0.003755	0.013089	0.044475	0.325291
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001364	0.001406	0.001447	0.001686	0.003339	0.0113
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001378	0.001353	0.001415	0.001392	0.001836	0.005251
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001333	0.00135	0.001417	0.001396	0.001389	0.001921
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001325	0.001323	0.001332	0.001346	0.001977	0.001553
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001374	0.001325	0.001354	0.001329	0.001358	0.001544
G	[-180, -90, 180, 90]	0.002256	0.002095	0.004775	0.053541	0.614501	8.140069
G	[113, -43, 153, -10]	0.001802	0.003424	0.00807	0.068974	0.691109	8.063631
G	[137.834, -10.7, 153.513336, -28.17]	0.002226	0.003987	0.015236	0.072564	0.716035	8.053971
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001786	0.004262	0.025962	0.271382	1.577215	9.30697
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.00222	0.00406	0.026574	0.299517	2.601609	16.31092
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001797	0.004043	0.026419	0.303725	3.086547	33.39919
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001781	0.004166	0.026417	0.299048	3.081642	32.93143
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.002242	0.004226	0.026492	0.27425	3.038951	33.55688
B	[-180, -90, 180, 90]	0.001784	0.00358	0.017219	0.192555	2.161404	23.53042
B	[113, -43, 153, -10]	0.001739	0.003609	0.018093	0.191498	2.157822	24.08449
B	[137.834, -10.7, 153.513336, -28.17]	0.001768	0.002043	0.005516	0.034233	0.41672	4.505318
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001604	0.00168	0.001754	0.001919	0.003586	0.021376
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001578	0.001593	0.001634	0.001937	0.001954	0.004194
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001579	0.001598	0.001622	0.001602	0.0016	0.001963
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.0016	0.001656	0.001564	0.001574	0.00157	0.001788
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001588	0.001611	0.001545	0.001911	0.001562	0.00175
M	[-180, -90, 180, 90]	0.001431	0.003145	0.016997	0.217641	2.216735	22.99472
M	[113, -43, 153, -10]	0.001373	0.00274	0.017214	0.217539	2.141562	22.97803
M	[137.834, -10.7, 153.513336, -28.17]	0.001371	0.002756	0.016997	0.213431	2.132764	22.92697
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.00137	0.002994	0.017236	0.211104	2.061938	22.73019
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001469	0.003183	0.017001	0.209827	2.028578	23.12695
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001404	0.002947	0.017066	0.208114	2.129535	23.05414
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001377	0.002748	0.017242	0.212223	2.121439	22.28099
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001398	0.002933	0.017326	0.184996	2.110353	22.81689
GB	[-180, -90, 180, 90]	0.002031	0.002527	0.005623	0.064454	0.731915	8.232003
GB	[113, -43, 153, -10]	0.002197	0.003866	0.008104	0.068997	0.758526	8.703573
GB	[137.834, -10.7, 153.513336, -28.17]	0.002012	0.00248	0.004445	0.013453	0.145872	1.591164
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001873	0.001978	0.00197	0.002811	0.004225	0.008847
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.0019	0.001898	0.001934	0.00194	0.002243	0.003928
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001877	0.001873	0.001863	0.001878	0.001952	0.002242
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001915	0.001885	0.001861	0.001873	0.001876	0.001985
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.00186	0.001874	0.001816	0.001875	0.001924	0.002036

Table 16: Get Points as WKT String – Time (s) – Australia Wide

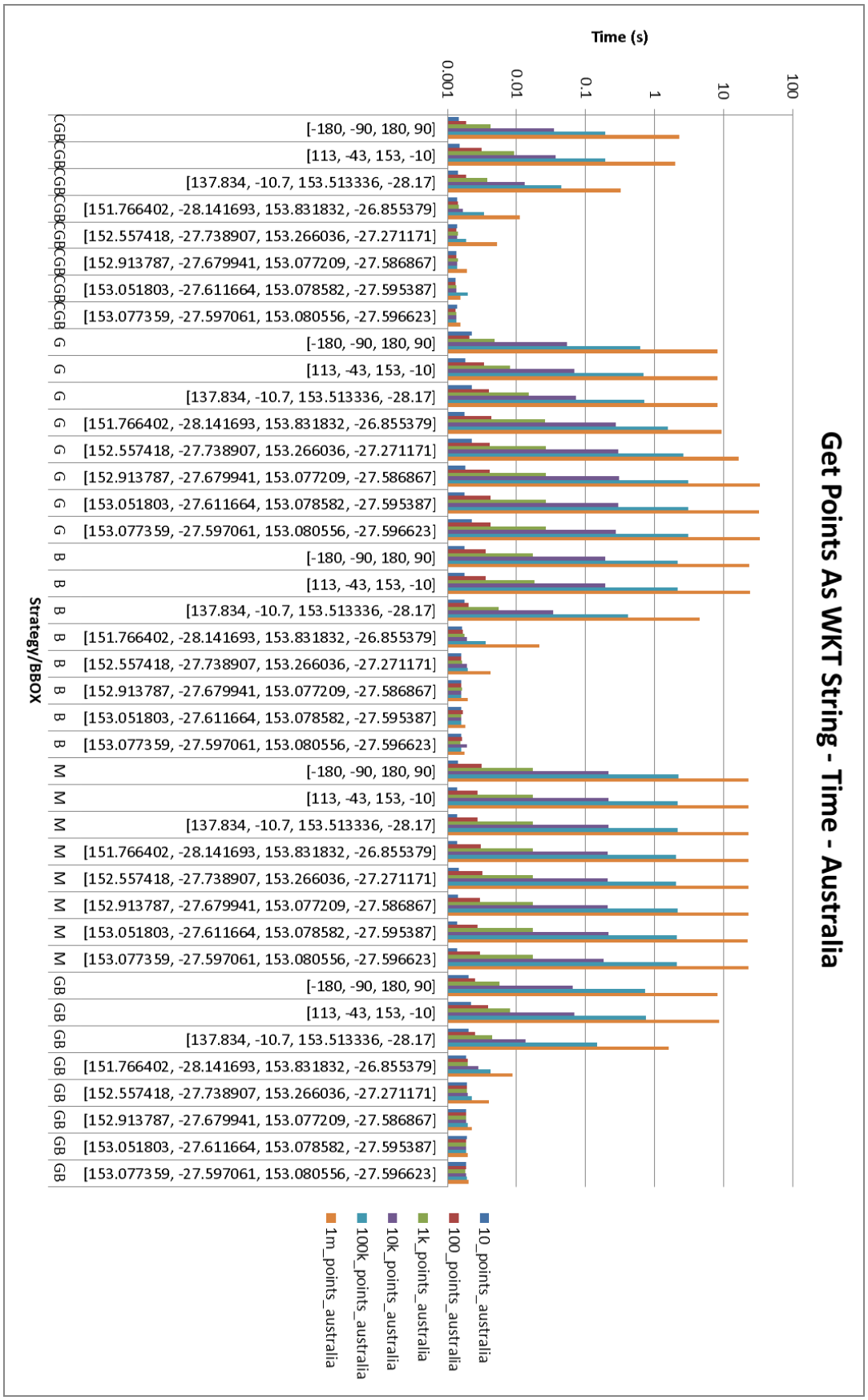

Figure 77: Get Points as WKT String – Time (s) – Australia Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	228	437	522	517	522	522
CGB	[113, -43, 153, -10]	438	3602	13346	14218	14211	14220
CGB	[137.834, -10.7, 153.513336, -28.17]	103	977	5959	11969	12038	12057
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	20	61	103	602	3023	4448
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	61	20	394	4366
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	228
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20
G	[-180, -90, 180, 90]	144	309	396	395	396	396
G	[113, -43, 153, -10]	438	2941	4914	5044	5044	5036
G	[137.834, -10.7, 153.513336, -28.17]	438	3894	17899	21927	21970	21955
G	[151.766402, -28.141693, 153.831832, -26.855379]	438	4187	41183	375382	1729593	1966936
G	[152.557418, -27.738907, 153.266036, -27.271171]	438	4187	41814	412935	3669827	14731952
G	[152.913787, -27.679941, 153.077209, -27.586867]	438	4187	41814	417858	4177961	41777366
G	[153.051803, -27.611664, 153.078582, -27.595387]	438	4187	41814	417858	4177961	41777366
G	[153.077359, -27.597061, 153.080556, -27.596623]	438	4187	41814	417858	4177961	41777366
B	[-180, -90, 180, 90]	438	4187	41814	417858	4177961	41777366
B	[113, -43, 153, -10]	438	4187	41814	417858	4177961	41777366
B	[137.834, -10.7, 153.513336, -28.17]	103	977	8955	85386	842728	8380262
B	[151.766402, -28.141693, 153.831832, -26.855379]	20	61	103	602	4991	48403
B	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	61	20	394	5497
B	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	228
B	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
B	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20
M	[-180, -90, 180, 90]	438	4187	41814	417858	4177961	41777366
M	[113, -43, 153, -10]	438	4187	41814	417858	4177961	41777366
M	[137.834, -10.7, 153.513336, -28.17]	438	4187	41814	417858	4177961	41777366
M	[151.766402, -28.141693, 153.831832, -26.855379]	438	4187	41814	417858	4177961	41777366
M	[152.557418, -27.738907, 153.266036, -27.271171]	438	4187	41814	417858	4177961	41777366
M	[152.913787, -27.679941, 153.077209, -27.586867]	438	4187	41814	417858	4177961	41777366
M	[153.051803, -27.611664, 153.078582, -27.595387]	438	4187	41814	417858	4177961	41777366
M	[153.077359, -27.597061, 153.080556, -27.596623]	438	4187	41814	417858	4177961	41777366
GB	[-180, -90, 180, 90]	144	309	396	396	395	396
GB	[113, -43, 153, -10]	438	2941	4914	5042	5039	5036
GB	[137.834, -10.7, 153.513336, -28.17]	103	935	3738	4863	5039	5028
GB	[151.766402, -28.141693, 153.831832, -26.855379]	20	61	103	602	2396	3065
GB	[152.557418, -27.738907, 153.266036, -27.271171]	20	20	61	20	394	2570
GB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	20	20	20	228
GB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	20	20	20
GB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	20

Table 17: Get Points as WKT String – String Length (Characters) – Australia Wide

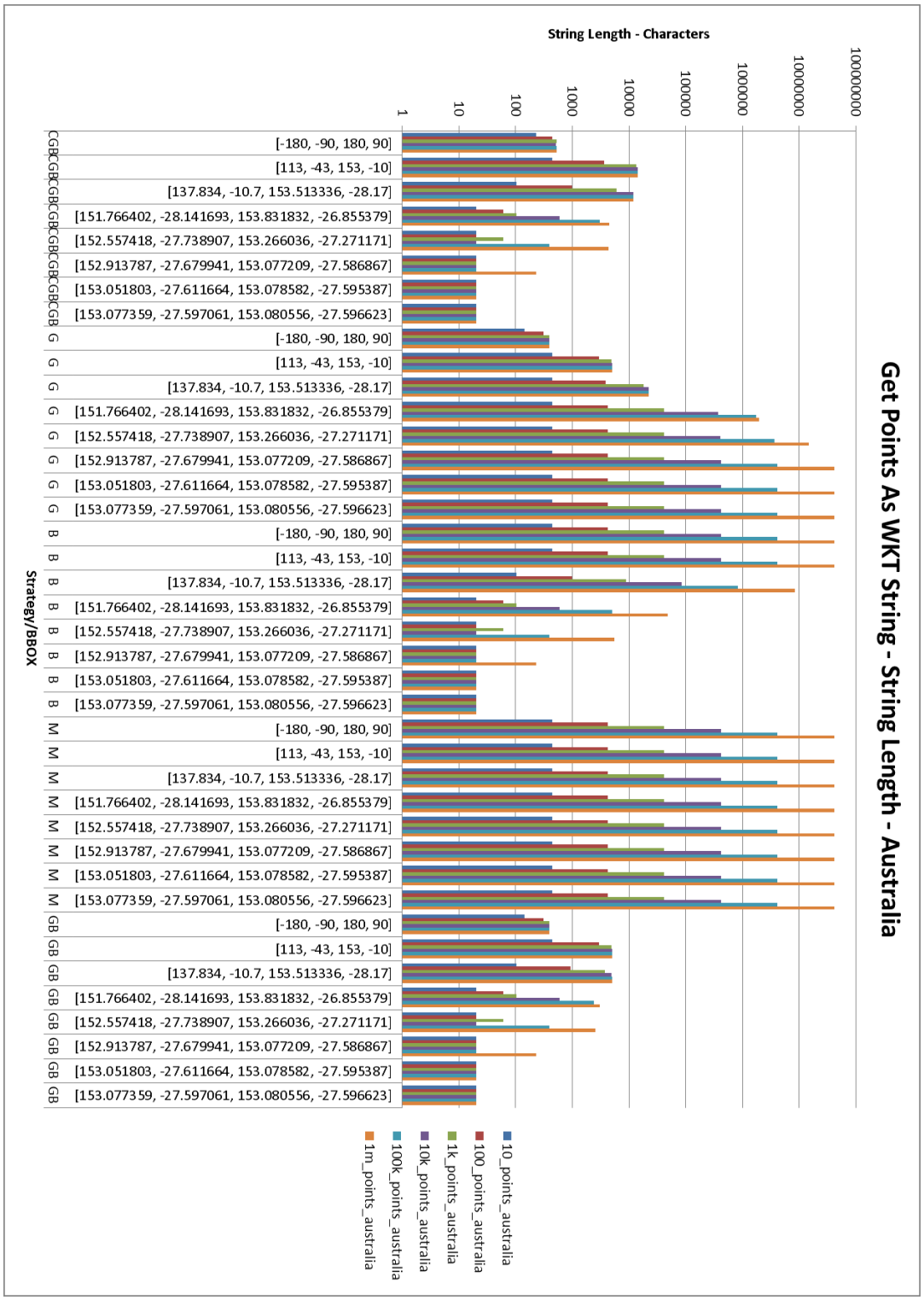

Figure 78: Get Points as WKT String – String Length (Characters) – Australia Wide

8.2.3 Brisbane Wide

Strategy	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	0.166418	0.634273	3.630482	21.72394	153.5975	1729.685
G	1.20E-05	1.20E-05	1.40E-05	1.50E-05	1.50E-05	1.50E-05
B	1.30E-05	1.30E-05	1.40E-05	1.40E-05	1.60E-05	1.60E-05
M	1.20E-05	1.20E-05	1.40E-05	1.50E-05	1.40E-05	1.50E-05
GB	1.20E-05	1.20E-05	1.50E-05	1.50E-05	1.70E-05	1.60E-05

Table 18: Pre-Processing – Time (s) – Brisbane Wide

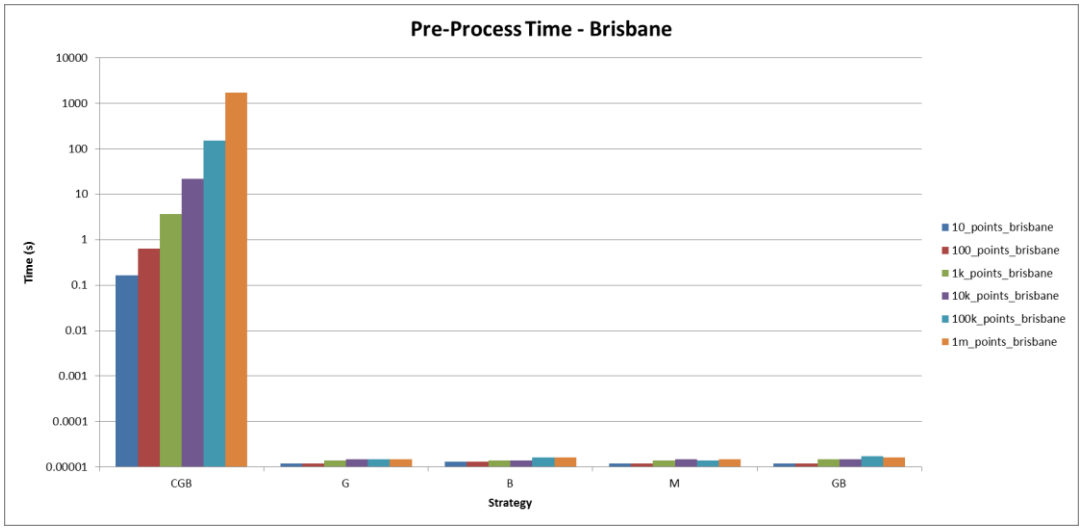

Figure 79: Pre-Processing – Time (s) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	2	2	2	2	2	2
CGB	[113, -43, 153, -10]	2	2	2	2	2	2
CGB	[137.834, -10.7, 153.513336, -28.17]	4	4	4	4	4	4
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	10	78	192	198	198	198
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	1	14	115	337	345	345
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	3	69	560	5791
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	1	16	162
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	1
G	[-180, -90, 180, 90]	1	1	1	1	1	1
G	[113, -43, 153, -10]	3	4	4	4	4	4
G	[137.834, -10.7, 153.513336, -28.17]	3	4	4	4	4	4
G	[151.766402, -28.141693, 153.831832, -26.855379]	9	60	112	119	126	126
G	[152.557418, -27.738907, 153.266036, -27.271171]	10	95	572	825	828	828
G	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	999999
G	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	999999
G	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	999999
B	[-180, -90, 180, 90]	10	100	1000	10000	100000	999999
B	[113, -43, 153, -10]	8	57	592	6086	59925	597389
B	[137.834, -10.7, 153.513336, -28.17]	9	81	859	8517	84624	846485
B	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	1000	10000	100000	999999
B	[152.557418, -27.738907, 153.266036, -27.271171]	1	14	132	1252	12596	125386
B	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	3	69	560	5791
B	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	1	16	162
B	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	1
M	[-180, -90, 180, 90]	10	100	1000	10000	100000	999999
M	[113, -43, 153, -10]	10	100	1000	10000	100000	999999
M	[137.834, -10.7, 153.513336, -28.17]	10	100	1000	10000	100000	999999
M	[151.766402, -28.141693, 153.831832, -26.855379]	10	100	1000	10000	100000	999999
M	[152.557418, -27.738907, 153.266036, -27.271171]	10	100	1000	10000	100000	999999
M	[152.913787, -27.679941, 153.077209, -27.586867]	10	100	1000	10000	100000	999999
M	[153.051803, -27.611664, 153.078582, -27.595387]	10	100	1000	10000	100000	999999
M	[153.077359, -27.597061, 153.080556, -27.596623]	10	100	1000	10000	100000	999999
GB	[-180, -90, 180, 90]	1	1	1	1	1	1
GB	[113, -43, 153, -10]	2	2	2	2	2	4
GB	[137.834, -10.7, 153.513336, -28.17]	3	4	4	4	4	4
GB	[151.766402, -28.141693, 153.831832, -26.855379]	9	60	112	119	126	126
GB	[152.557418, -27.738907, 153.266036, -27.271171]	1	13	74	114	117	117
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0	0	3	69	560	5791
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0	0	0	1	16	162
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0	0	0	0	0	1

Table 19: Clusters Generated – Brisbane Wide

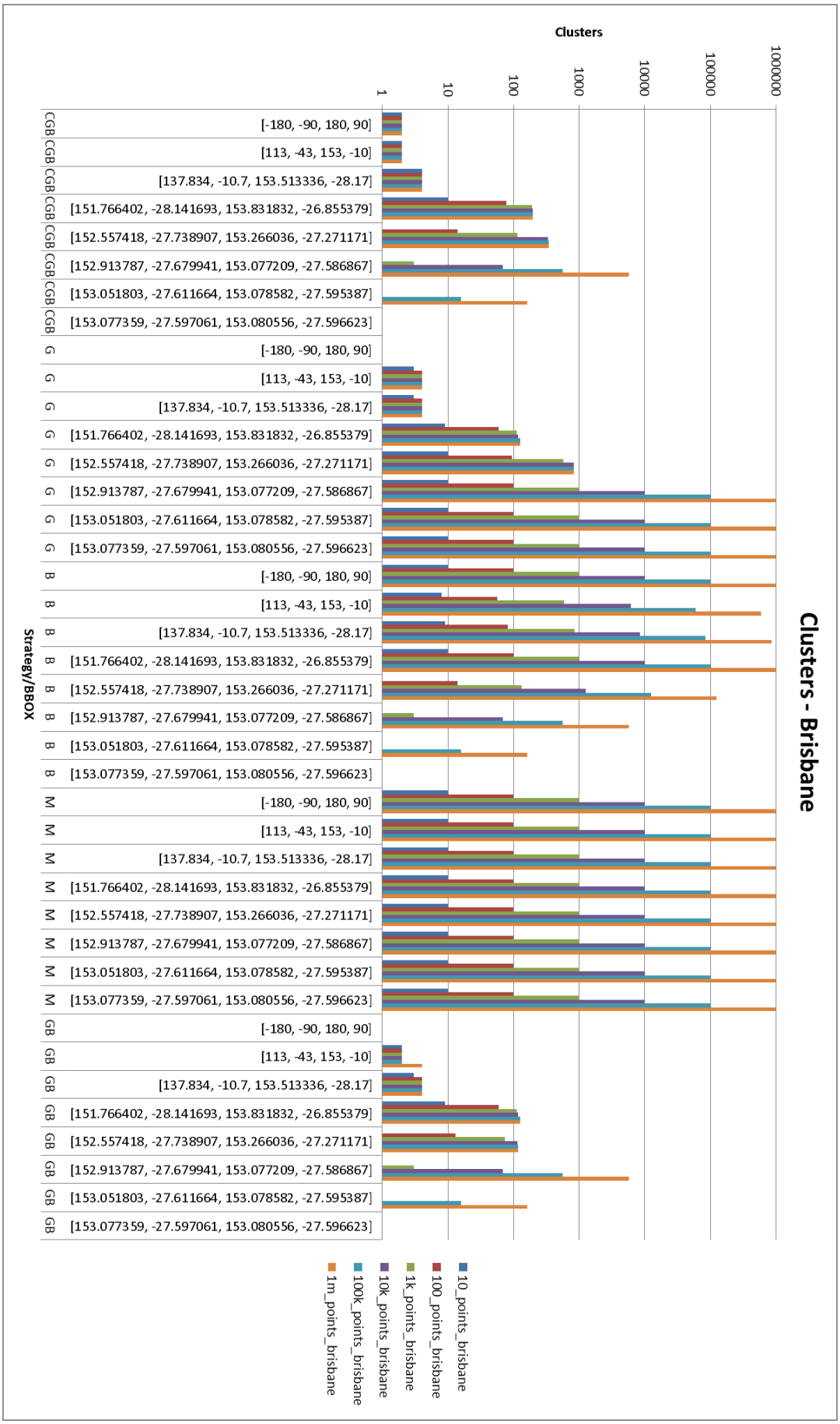

Figure 80: Clusters Generated – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.002861	0.003053	0.004359	0.013481	0.066838	0.535955
CGB	[113, -43, 153, -10]	0.00141	0.001504	0.002065	0.006002	0.033509	0.286693
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001454	0.001573	0.002367	0.007782	0.045985	0.368689
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.00151	0.002444	0.004967	0.011535	0.061431	0.459021
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001713	0.001621	0.003029	0.007038	0.014186	0.089423
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001365	0.001388	0.001481	0.002436	0.009454	0.082755
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001335	0.001394	0.001664	0.001467	0.001741	0.004612
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001348	0.001393	0.001353	0.001395	0.001431	0.001473
G	[-180, -90, 180, 90]	0.001745	0.001991	0.004412	0.034488	0.385268	4.187582
G	[113, -43, 153, -10]	0.001633	0.00191	0.004157	0.050048	0.588041	7.676447
G	[137.834, -10.7, 153.513336, -28.17]	0.001642	0.001915	0.004163	0.051839	0.583478	7.815538
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001759	0.003306	0.008024	0.059986	0.674293	7.897121
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001797	0.003755	0.016878	0.078938	0.749008	7.89832
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001756	0.003857	0.025808	0.277235	3.427542	33.64132
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001769	0.003858	0.025778	0.273207	3.053387	36.27075
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.00177	0.00385	0.025792	0.274367	3.055136	33.2354
B	[-180, -90, 180, 90]	0.001872	0.003116	0.018042	0.209125	2.673826	30.70534
B	[113, -43, 153, -10]	0.00171	0.002458	0.010301	0.098169	1.229218	14.98117
B	[137.834, -10.7, 153.513336, -28.17]	0.001738	0.003006	0.014202	0.163169	1.798115	19.96169
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001769	0.003018	0.016736	0.188862	2.067721	23.83514
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001656	0.001821	0.003563	0.021573	0.232892	2.715689
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001578	0.001594	0.001699	0.002865	0.01148	0.129833
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001594	0.001572	0.00162	0.001806	0.002076	0.004341
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001568	0.001591	0.00162	0.001755	0.001772	0.001843
M	[-180, -90, 180, 90]	0.001513	0.002778	0.018105	0.286573	2.55457	31.16223
M	[113, -43, 153, -10]	0.001375	0.002646	0.016181	0.206144	2.045646	23.52791
M	[137.834, -10.7, 153.513336, -28.17]	0.001413	0.00264	0.015964	0.204652	2.109738	23.26592
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001394	0.002662	0.016118	0.203756	2.038193	23.0674
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.00139	0.002662	0.017055	0.203246	2.074814	23.20964
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001378	0.002643	0.016781	0.203386	2.050308	23.16729
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001387	0.002662	0.015865	0.183754	2.040416	23.00087
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001375	0.002665	0.016144	0.183798	2.106249	23.35015
GB	[-180, -90, 180, 90]	0.002058	0.002329	0.00528	0.042104	0.434984	4.502786
GB	[113, -43, 153, -10]	0.002258	0.002896	0.00415	0.032826	0.397324	4.487481
GB	[137.834, -10.7, 153.513336, -28.17]	0.002	0.00224	0.00456	0.049254	0.527684	7.41987
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.00214	0.003438	0.007809	0.064683	0.738181	8.540057
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.002208	0.002224	0.003919	0.009169	0.083731	1.048129
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001844	0.002165	0.002005	0.003621	0.016707	6.288318
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.002179	0.001848	0.001896	0.002271	0.002492	0.006243
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001851	0.002171	0.001849	0.001894	0.002073	0.002123

Table 20: Get Points as GeoJSON – Time (s) – Brisbane Wide

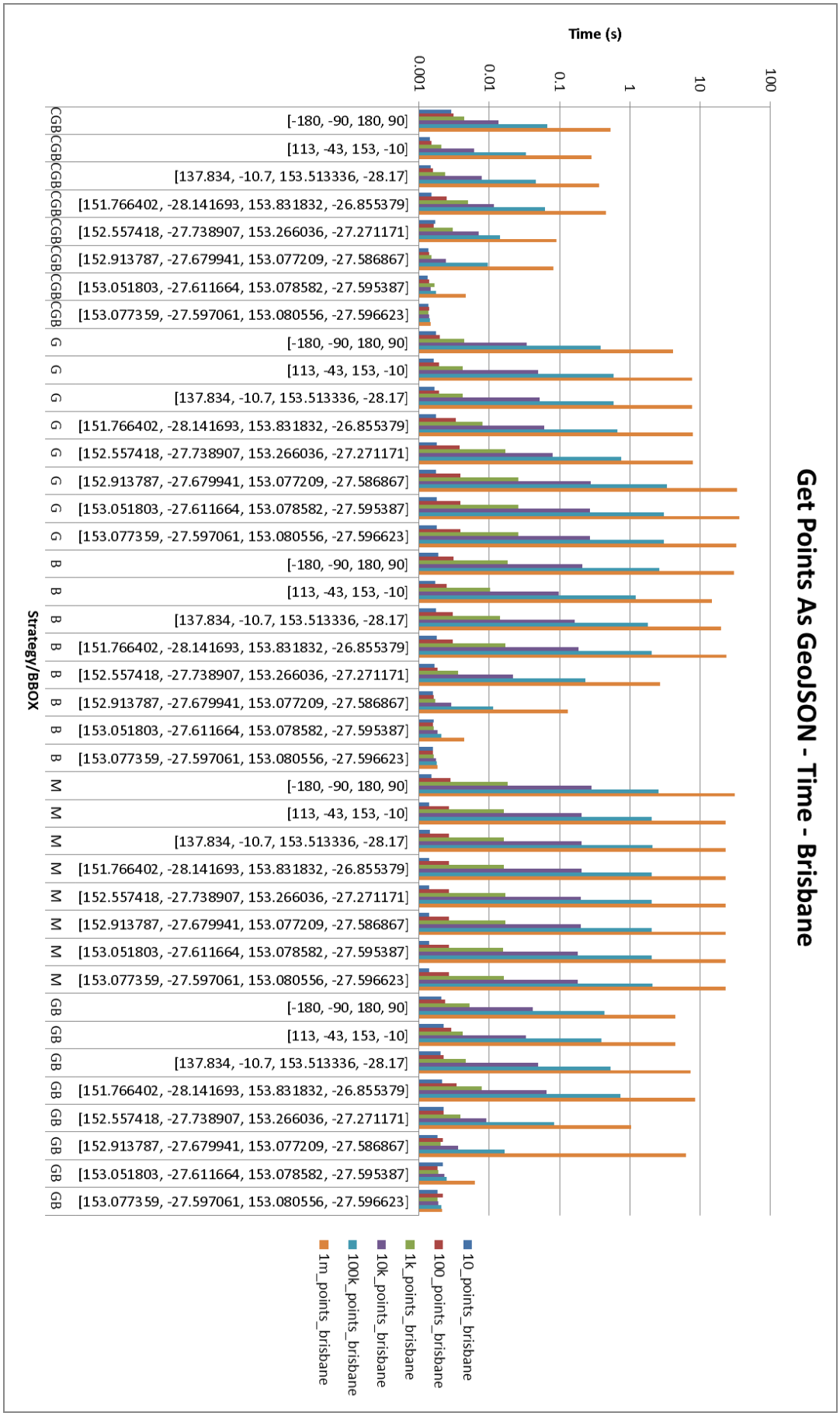

Figure 81: Get Points as GeoJSON – Time (s) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001522	0.001634	0.002512	0.008564	0.05686	0.414975
CGB	[113, -43, 153, -10]	0.001871	0.002271	0.003318	0.011755	0.073435	0.982441
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001557	0.001679	0.002467	0.007792	0.044711	0.372794
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001721	0.003706	0.009136	0.01471	0.0612	0.520421
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.00145	0.001871	0.005013	0.017624	0.020105	0.913339
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001377	0.001375	0.001535	0.003581	0.020586	0.179386
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001346	0.001363	0.001348	0.001501	0.0022	0.006881
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001365	0.001384	0.001346	0.001389	0.001437	0.001522
G	[-180, -90, 180, 90]	0.001636	0.001875	0.004024	0.032663	0.337124	3.998506
G	[113, -43, 153, -10]	0.001706	0.002684	0.004241	0.050289	0.575226	7.577074
G	[137.834, -10.7, 153.513336, -28.17]	0.001766	0.002007	0.004262	0.051946	0.582823	7.806749
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.00198	0.004023	0.009101	0.064788	0.67399	8.000547
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001993	0.005252	0.02633	0.089689	0.708383	7.906655
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001962	0.005676	0.042565	0.445678	5.265275	55.03086
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001964	0.005641	0.041574	0.44464	5.20109	55.48275
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.001959	0.005622	0.041619	0.48578	5.24831	55.37644
B	[-180, -90, 180, 90]	0.001958	0.005276	0.033232	0.399022	4.18066	47.32974
B	[113, -43, 153, -10]	0.002927	0.003354	0.019926	0.217371	2.443439	27.56252
B	[137.834, -10.7, 153.513336, -28.17]	0.001918	0.004017	0.028043	0.341092	3.556498	38.07793
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.002255	0.004596	0.032778	0.402543	4.271647	45.34407
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.00167	0.002103	0.005828	0.040554	0.482506	5.526547
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001878	0.001594	0.001778	0.004055	0.01943	0.223337
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001895	0.001612	0.001614	0.001859	0.002583	0.007018
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001584	0.001602	0.00191	0.001781	0.001787	0.001866
M	[-180, -90, 180, 90]	0.001576	0.004828	0.03314	0.385424	4.179769	44.79899
M	[113, -43, 153, -10]	0.001587	0.004396	0.032728	0.37917	4.300196	44.55579
M	[137.834, -10.7, 153.513336, -28.17]	0.002021	0.004725	0.032104	0.375413	4.171117	45.61817
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001578	0.004435	0.033223	0.375968	4.228747	44.55152
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001573	0.004373	0.034709	0.375686	4.141548	45.53814
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001852	0.004368	0.045246	0.374175	4.255553	44.98722
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001568	0.004416	0.031556	0.369095	4.238602	44.34472
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001862	0.004383	0.031793	0.371302	4.196919	45.85289
GB	[-180, -90, 180, 90]	0.002003	0.002261	0.004724	0.038441	0.426953	4.557255
GB	[113, -43, 153, -10]	0.002036	0.002179	0.003695	0.027628	0.34096	4.467612
GB	[137.834, -10.7, 153.513336, -28.17]	0.002842	0.00266	0.004589	0.049306	0.535386	7.011088
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.002323	0.004683	0.009642	0.066723	0.737903	8.553837
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001979	0.002458	0.005104	0.011265	0.083901	1.166317
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.002161	0.001868	0.002068	0.005011	0.081374	0.276385
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001866	0.001882	0.001881	0.002009	0.002799	0.008706
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.002172	0.001862	0.001891	0.002243	0.00209	0.002168

Table 21: Get Points as GeoJSON String – Time (s) – Brisbane Wide

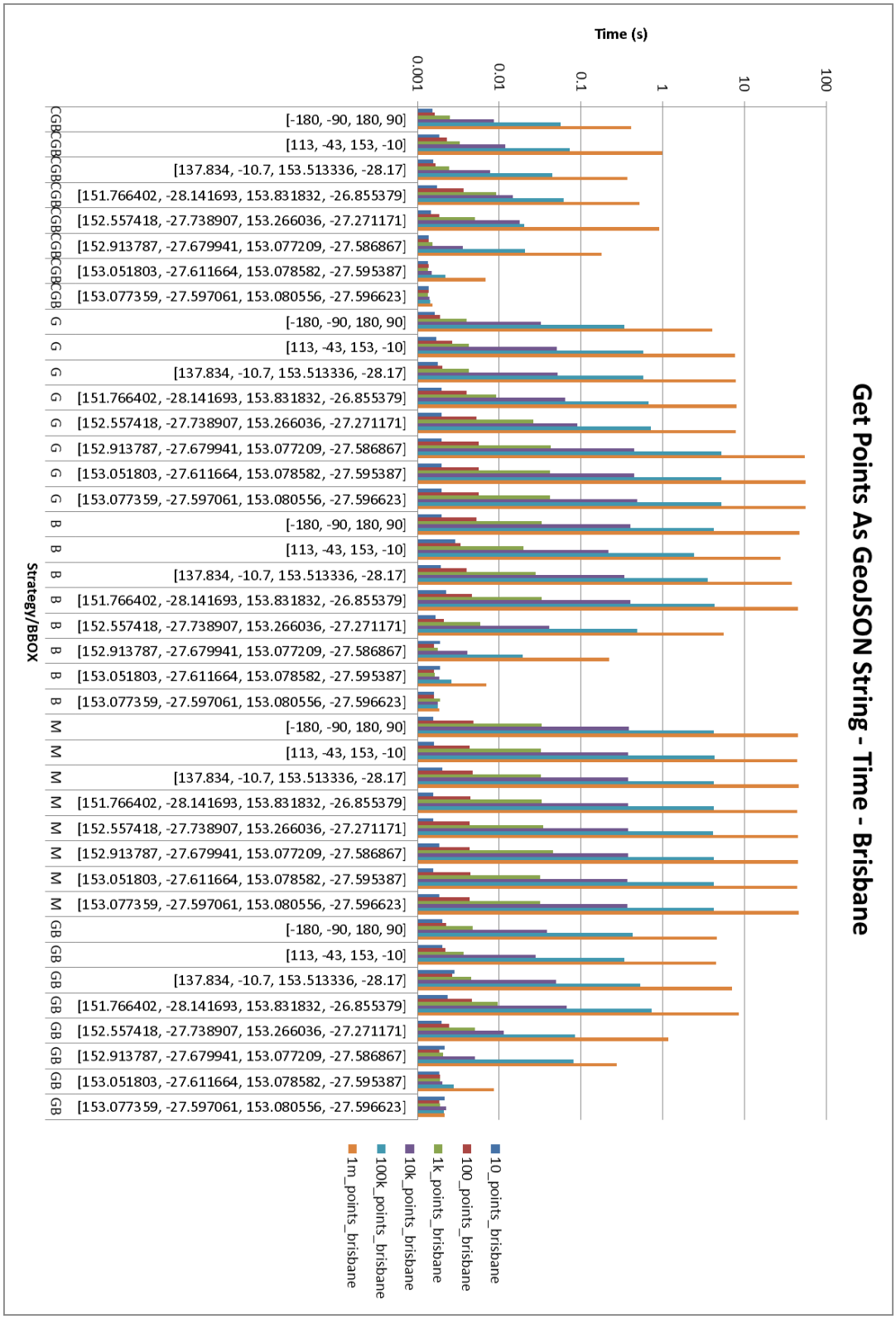

Figure 82: Get Points as GeoJSON String – Time (s) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	325	326	328	331	332	334
CGB	[113, -43, 153, -10]	325	324	327	328	332	333
CGB	[137.834, -10.7, 153.513336, -28.17]	606	611	614	618	623	624
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	1453	11029	27081	28110	28305	28500
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	184	2013	16238	47487	48962	49311
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	466	9754	78878	815263
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	184	2297	22847
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	184
G	[-180, -90, 180, 90]	185	186	187	188	189	190
G	[113, -43, 153, -10]	466	606	611	615	620	624
G	[137.834, -10.7, 153.513336, -28.17]	466	611	612	616	622	625
G	[151.766402, -28.141693, 153.831832, -26.855379]	1312	8496	15856	16981	18094	18230
G	[152.557418, -27.738907, 153.266036, -27.271171]	1453	13418	80576	116806	118173	119012
G	[152.913787, -27.679941, 153.077209, -27.586867]	1453	14123	140794	1407837	14077916	1.41E+08
G	[153.051803, -27.611664, 153.078582, -27.595387]	1453	14123	140794	1407837	14077916	1.41E+08
G	[153.077359, -27.597061, 153.080556, -27.596623]	1453	14123	140794	1407837	14077916	1.41E+08
B	[-180, -90, 180, 90]	1453	14123	140794	1407837	14077916	1.41E+08
B	[113, -43, 153, -10]	1171	8066	83371	856788	8436050	84098271
B	[137.834, -10.7, 153.513336, -28.17]	1312	11445	120952	1199046	11913233	1.19E+08
B	[151.766402, -28.141693, 153.831832, -26.855379]	1453	14123	140794	1407837	14077916	1.41E+08
B	[152.557418, -27.738907, 153.266036, -27.271171]	184	2013	18629	176291	1773275	17651586
B	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	466	9754	78878	815263
B	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	184	2297	22847
B	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	184
M	[-180, -90, 180, 90]	1453	14123	140794	1407837	14077916	1.41E+08
M	[113, -43, 153, -10]	1453	14123	140794	1407837	14077916	1.41E+08
M	[137.834, -10.7, 153.513336, -28.17]	1453	14123	140794	1407837	14077916	1.41E+08
M	[151.766402, -28.141693, 153.831832, -26.855379]	1453	14123	140794	1407837	14077916	1.41E+08
M	[152.557418, -27.738907, 153.266036, -27.271171]	1453	14123	140794	1407837	14077916	1.41E+08
M	[152.913787, -27.679941, 153.077209, -27.586867]	1453	14123	140794	1407837	14077916	1.41E+08
M	[153.051803, -27.611664, 153.078582, -27.595387]	1453	14123	140794	1407837	14077916	1.41E+08
M	[153.077359, -27.597061, 153.080556, -27.596623]	1453	14123	140794	1407837	14077916	1.41E+08
GB	[-180, -90, 180, 90]	185	186	187	188	187	190
GB	[113, -43, 153, -10]	325	324	327	328	332	616
GB	[137.834, -10.7, 153.513336, -28.17]	466	609	613	616	621	625
GB	[151.766402, -28.141693, 153.831832, -26.855379]	1312	8496	15856	16984	18094	18230
GB	[152.557418, -27.738907, 153.266036, -27.271171]	184	1872	10463	16159	16715	16828
GB	[152.913787, -27.679941, 153.077209, -27.586867]	45	45	466	9754	78878	815263
GB	[153.051803, -27.611664, 153.078582, -27.595387]	45	45	45	184	2297	22847
GB	[153.077359, -27.597061, 153.080556, -27.596623]	45	45	45	45	45	184

Table 22: Get Points as GeoJSON – String Length (Characters) – Brisbane Wide

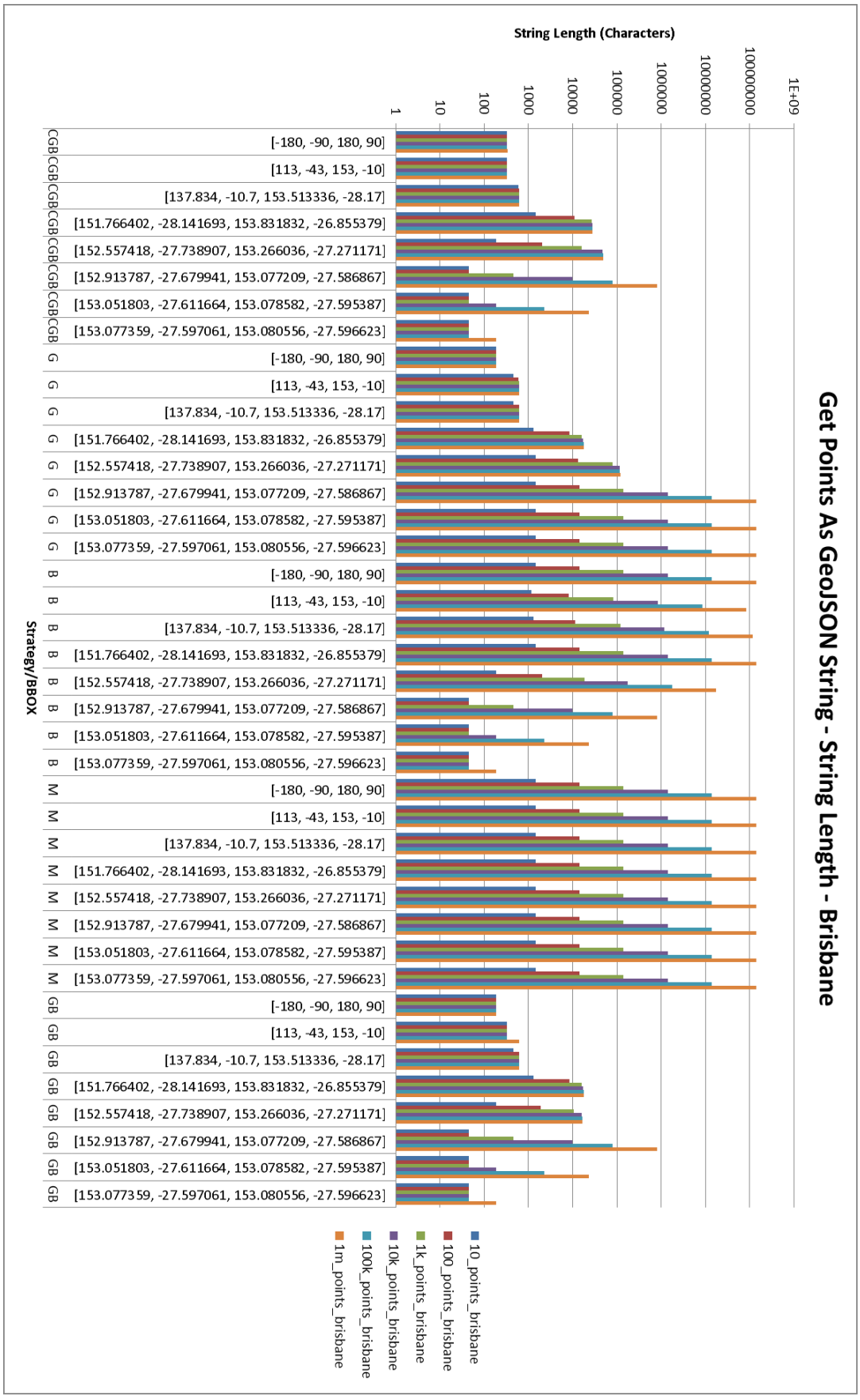

Figure 83: Get Points as GeoJSON – String Length (Characters) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001366	0.001499	0.00242	0.009557	0.055206	0.455557
CGB	[113, -43, 153, -10]	0.001365	0.001463	0.001997	0.005946	0.032599	0.291376
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001407	0.00154	0.002341	0.007667	0.044207	0.375312
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001482	0.002629	0.004492	0.010848	0.060964	0.457559
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001368	0.001556	0.002807	0.008415	0.013232	0.089086
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001313	0.001329	0.001415	0.002319	0.0457	3.974975
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001313	0.001329	0.001316	0.001772	0.001707	0.00394
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001311	0.001673	0.001331	0.001343	0.001766	0.001473
G	[-180, -90, 180, 90]	0.001565	0.001775	0.003904	0.032527	0.334369	4.063005
G	[113, -43, 153, -10]	0.0016	0.001871	0.004695	0.054886	0.626327	8.505617
G	[137.834, -10.7, 153.513336, -28.17]	0.001687	0.001862	0.004175	0.051695	0.583065	7.780399
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001714	0.002932	0.00681	0.059236	0.674308	7.924652
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.001762	0.003775	0.015931	0.073152	0.693471	14.00461
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001719	0.003647	0.024332	0.323549	2.97341	32.02096
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001719	0.003618	0.024386	0.280254	2.880304	31.62855
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.00172	0.003613	0.024174	0.275966	2.927857	31.83462
B	[-180, -90, 180, 90]	0.001711	0.002821	0.015442	0.241477	1.973006	22.6129
B	[113, -43, 153, -10]	0.001678	0.002282	0.009717	0.09074	1.142912	13.67533
B	[137.834, -10.7, 153.513336, -28.17]	0.001719	0.002608	0.013355	0.172336	1.685205	19.30744
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001716	0.003096	0.015367	0.198038	1.986125	22.48877
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001594	0.001772	0.003368	0.019277	0.241517	2.600995
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.00153	0.001858	0.001681	0.002747	0.009791	0.139706
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001556	0.001867	0.001572	0.001782	0.001997	0.004287
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001524	0.00159	0.001576	0.001722	0.001735	0.001807
M	[-180, -90, 180, 90]	0.001339	0.002452	0.014918	0.176502	1.932229	22.32299
M	[113, -43, 153, -10]	0.001337	0.00248	0.014947	0.172894	1.982134	22.3531
M	[137.834, -10.7, 153.513336, -28.17]	0.001331	0.002434	0.015314	0.172754	1.941626	22.12759
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001326	0.002473	0.014881	0.172667	1.983576	22.57356
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001345	0.002469	0.01646	0.172942	1.943571	22.13299
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.00132	0.002494	0.057148	0.172763	1.980492	22.05655
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001349	0.002465	0.014739	0.171166	2.014667	22.33217
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001355	0.002473	0.014772	0.171197	1.946024	22.1091
GB	[-180, -90, 180, 90]	0.001921	0.002142	0.004625	0.038377	0.422644	4.574022
GB	[113, -43, 153, -10]	0.001916	0.002062	0.003619	0.02753	0.340814	4.43824
GB	[137.834, -10.7, 153.513336, -28.17]	0.001987	0.00221	0.004751	0.049604	0.528465	7.087815
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.002075	0.003338	0.007714	0.064046	0.734599	8.478143
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001971	0.002216	0.004028	0.008801	0.081654	1.00578
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001809	0.001813	0.001986	0.00347	0.015429	0.175399
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001814	0.001824	0.002147	0.001906	0.002764	0.005827
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001807	0.0018	0.001835	0.001855	0.002032	0.002093

Table 23: Get Points as WKT – Time (s) – Brisbane Wide

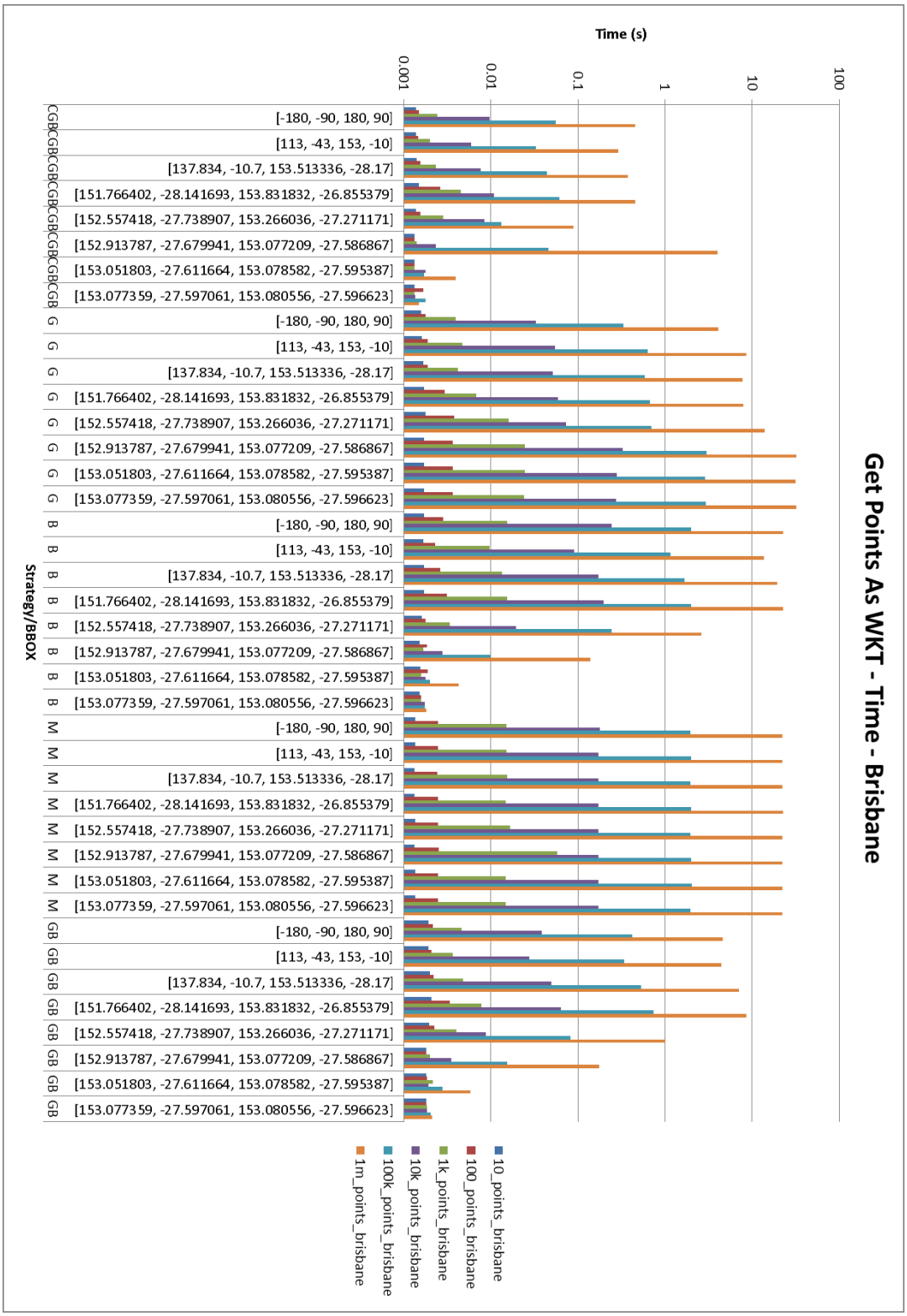

Figure 84: Get Points as WKT – Time (s) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	0.001398	0.001553	0.002401	0.008442	0.056472	0.412616
CGB	[113, -43, 153, -10]	0.001378	0.001473	0.002009	0.00599	0.032723	0.293739
CGB	[137.834, -10.7, 153.513336, -28.17]	0.001422	0.001553	0.002347	0.007654	0.044178	0.371777
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	0.001501	0.002404	0.004557	0.010985	0.056028	0.412729
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001361	0.001596	0.002919	0.007042	0.013456	0.090381
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	0.001318	0.001665	0.001437	0.002357	0.009131	0.077398
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001332	0.001345	0.001334	0.001414	0.00172	0.004053
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	0.001307	0.001334	0.001304	0.001356	0.001408	0.001436
G	[-180, -90, 180, 90]	0.001935	0.001777	0.003895	0.032534	0.335212	3.976995
G	[113, -43, 153, -10]	0.001609	0.00187	0.004125	0.049981	0.572106	7.586037
G	[137.834, -10.7, 153.513336, -28.17]	0.001899	0.001864	0.004178	0.051681	0.586059	7.785947
G	[151.766402, -28.141693, 153.831832, -26.855379]	0.001718	0.002992	0.00687	0.059465	0.675828	8.026065
G	[152.557418, -27.738907, 153.266036, -27.271171]	0.002006	0.003643	0.01649	0.07393	0.69297	7.892361
G	[152.913787, -27.679941, 153.077209, -27.586867]	0.001737	0.00371	0.024576	0.281652	2.961095	32.61959
G	[153.051803, -27.611664, 153.078582, -27.595387]	0.001757	0.003718	0.02452	0.28215	2.99842	32.94593
G	[153.077359, -27.597061, 153.080556, -27.596623]	0.002033	0.003676	0.024875	0.261727	2.969031	32.65733
B	[-180, -90, 180, 90]	0.001711	0.002915	0.015948	0.183568	2.074537	23.98837
B	[113, -43, 153, -10]	0.00167	0.002342	0.010043	0.095708	1.209079	13.75799
B	[137.834, -10.7, 153.513336, -28.17]	0.001716	0.002701	0.013715	0.158474	1.744642	19.93562
B	[151.766402, -28.141693, 153.831832, -26.855379]	0.001734	0.002928	0.015842	0.182897	2.100875	23.44095
B	[152.557418, -27.738907, 153.266036, -27.271171]	0.001594	0.001791	0.003412	0.020075	0.223009	2.651702
B	[152.913787, -27.679941, 153.077209, -27.586867]	0.001541	0.001556	0.001664	0.002821	0.010088	0.125344
B	[153.051803, -27.611664, 153.078582, -27.595387]	0.001562	0.001561	0.00157	0.001785	0.002015	0.004236
B	[153.077359, -27.597061, 153.080556, -27.596623]	0.001561	0.001564	0.001583	0.002015	0.00174	0.00181
M	[-180, -90, 180, 90]	0.001341	0.002506	0.01591	0.17729	2.039085	23.17386
M	[113, -43, 153, -10]	0.001344	0.002533	0.015477	0.178105	2.015786	23.28206
M	[137.834, -10.7, 153.513336, -28.17]	0.00135	0.002559	0.015677	0.177439	2.071479	22.84952
M	[151.766402, -28.141693, 153.831832, -26.855379]	0.001361	0.002523	0.015431	0.176811	1.996299	23.14054
M	[152.557418, -27.738907, 153.266036, -27.271171]	0.001345	0.002512	0.016494	0.177565	2.071042	22.84491
M	[152.913787, -27.679941, 153.077209, -27.586867]	0.001343	0.002529	0.015832	0.202964	2.013765	23.25235
M	[153.051803, -27.611664, 153.078582, -27.595387]	0.001334	0.002518	0.015259	0.198914	2.009206	23.13559
M	[153.077359, -27.597061, 153.080556, -27.596623]	0.001364	0.002518	0.015221	0.198047	2.045661	22.8405
GB	[-180, -90, 180, 90]	0.001914	0.002146	0.004635	0.038492	0.422474	5.467193
GB	[113, -43, 153, -10]	0.001951	0.002074	0.003656	0.027664	0.343482	4.440845
GB	[137.834, -10.7, 153.513336, -28.17]	0.001981	0.002207	0.004449	0.049171	0.54558	7.387973
GB	[151.766402, -28.141693, 153.831832, -26.855379]	0.002114	0.003443	0.007555	0.064066	0.742099	8.507974
GB	[152.557418, -27.738907, 153.266036, -27.271171]	0.001891	0.002189	0.00385	0.008914	0.081296	1.209753
GB	[152.913787, -27.679941, 153.077209, -27.586867]	0.0018	0.001804	0.002244	0.003548	0.017083	0.195091
GB	[153.051803, -27.611664, 153.078582, -27.595387]	0.001814	0.001817	0.00189	0.001935	0.002496	0.005931
GB	[153.077359, -27.597061, 153.080556, -27.596623]	0.00183	0.001833	0.002178	0.001834	0.002391	0.002101

Table 24: Get Points as WKT String – Time (s) – Brisbane Wide

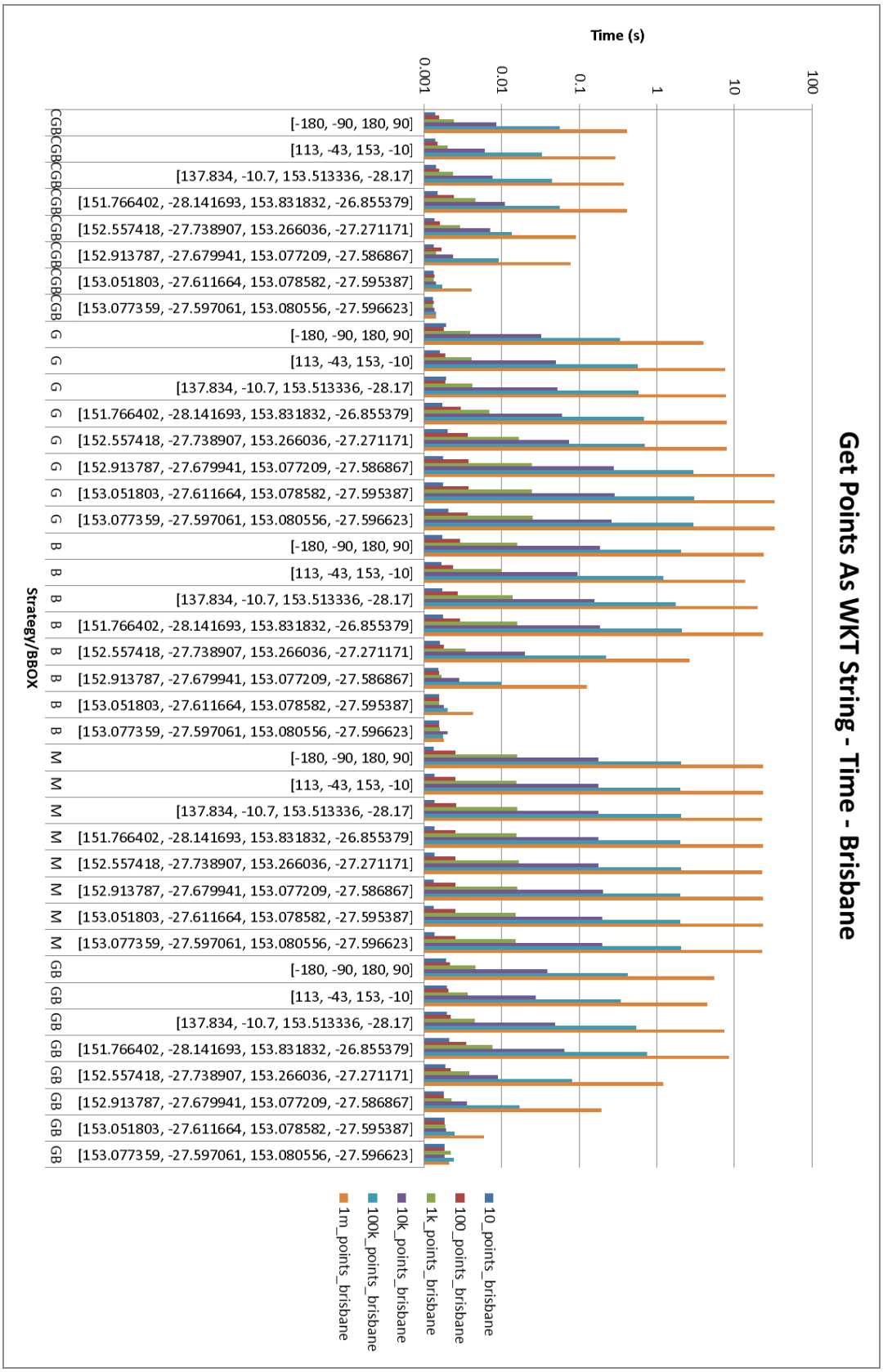

Figure 85: Get Points as WKT String – Time (s) – Brisbane Wide

Strategy	BBOX	10 pts	100 pts	1k pts	10k pts	100k pts	1m pts
CGB	[-180, -90, 180, 90]	103	102	102	103	102	102
CGB	[113, -43, 153, -10]	103	101	102	101	103	102
CGB	[137.834, -10.7, 153.513336, -28.17]	186	187	186	186	187	184
CGB	[151.766402, -28.141693, 153.831832, -26.855379]	439	3283	8036	8294	8294	8291
CGB	[152.557418, -27.738907, 153.266036, -27.271171]	61	603	4829	14098	14438	14442
CGB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	145	2899	23414	241930
CGB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	61	689	6785
CGB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	61
G	[-180, -90, 180, 90]	61	61	61	61	61	61
G	[113, -43, 153, -10]	145	184	185	185	186	186
G	[137.834, -10.7, 153.513336, -28.17]	145	187	185	185	186	185
G	[151.766402, -28.141693, 153.831832, -26.855379]	397	2532	4689	4999	5284	5290
G	[152.557418, -27.738907, 153.266036, -27.271171]	439	3989	23924	34493	34619	34627
G	[152.913787, -27.679941, 153.077209, -27.586867]	439	4199	41770	417813	4177892	41777387
G	[153.051803, -27.611664, 153.078582, -27.595387]	439	4199	41770	417813	4177892	41777387
G	[153.077359, -27.597061, 153.080556, -27.596623]	439	4199	41770	417813	4177892	41777387
B	[-180, -90, 180, 90]	439	4199	41770	417813	4177892	41777387
B	[113, -43, 153, -10]	355	2399	24739	254250	2503451	24956736
B	[137.834, -10.7, 153.513336, -28.17]	397	3402	35887	355839	3535433	35363635
B	[151.766402, -28.141693, 153.831832, -26.855379]	439	4199	41770	417813	4177892	41777387
B	[152.557418, -27.738907, 153.266036, -27.271171]	61	603	5537	52319	526247	5238348
B	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	145	2899	23414	241930
B	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	61	689	6785
B	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	61
M	[-180, -90, 180, 90]	439	4199	41770	417813	4177892	41777387
M	[113, -43, 153, -10]	439	4199	41770	417813	4177892	41777387
M	[137.834, -10.7, 153.513336, -28.17]	439	4199	41770	417813	4177892	41777387
M	[151.766402, -28.141693, 153.831832, -26.855379]	439	4199	41770	417813	4177892	41777387
M	[152.557418, -27.738907, 153.266036, -27.271171]	439	4199	41770	417813	4177892	41777387
M	[152.913787, -27.679941, 153.077209, -27.586867]	439	4199	41770	417813	4177892	41777387
M	[153.051803, -27.611664, 153.078582, -27.595387]	439	4199	41770	417813	4177892	41777387
M	[153.077359, -27.597061, 153.080556, -27.596623]	439	4199	41770	417813	4177892	41777387
GB	[-180, -90, 180, 90]	61	61	61	61	59	61
GB	[113, -43, 153, -10]	103	101	102	101	103	186
GB	[137.834, -10.7, 153.513336, -28.17]	145	187	187	186	187	187
GB	[151.766402, -28.141693, 153.831832, -26.855379]	397	2532	4689	5002	5284	5290
GB	[152.557418, -27.738907, 153.266036, -27.271171]	61	561	3113	4778	4911	4905
GB	[152.913787, -27.679941, 153.077209, -27.586867]	20	20	145	2899	23414	241930
GB	[153.051803, -27.611664, 153.078582, -27.595387]	20	20	20	61	689	6785
GB	[153.077359, -27.597061, 153.080556, -27.596623]	20	20	20	20	20	61

Table 25: Get Points as WKT String – String Length (Characters) – Brisbane Wide

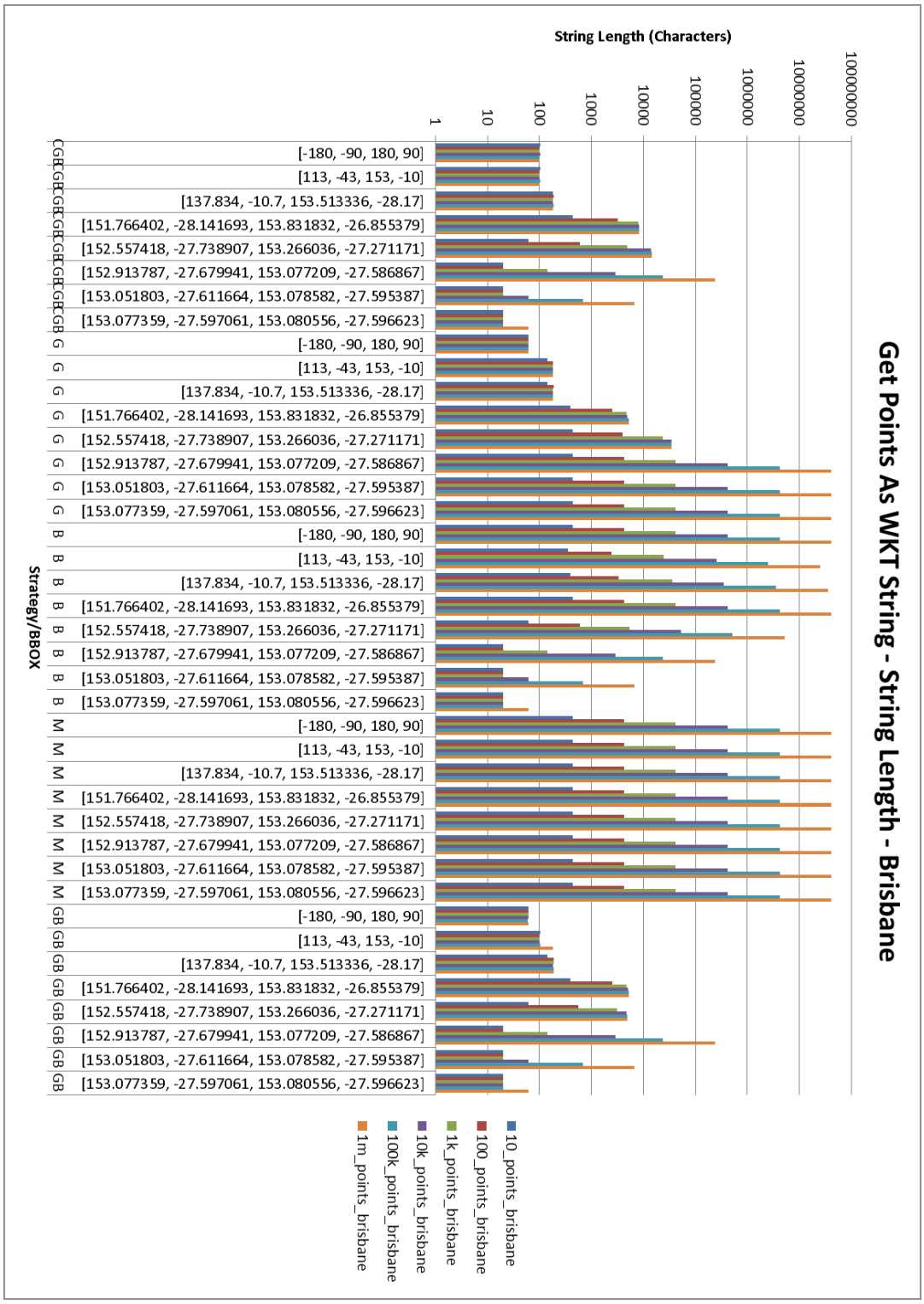

Figure 86: Get Points as WKT String – String Length (Characters) – Brisbane Wide

8.3 Source Code Repository Details

The source code repository associated with this thesis is publicly available via GitHub. The following is the repository's URL: https://github.com/robertpyke/PyThesis. Please see the repository's README for further information regarding the use of the repository.