A Comparison of API of Key-Value and Document-Based Data-Stores with Python Programming

Ehsan Azizi Khadem[1], Emad Fereshteh Nezhad[2]

[1.] MSc of Computer Engineering, Department of Computer Engineering, Lorestan University, Iran
[2.] MSc of Computer Engineering, Communications Regulatory Authority of I.R of Iran

Abstract: Today, NoSql data-stores are widely used in databases. NoSql, is based on NoRel (Not Only Relational) concept in database designing. Relational databases are most important technology for storing data during 50 years. But nowadays, because of the explosive growth of data and generating complex type of it like Web2, RDBMS are not suitable and enough for data storing. NoSql has different data model and API to access the data, comparing relational method. NoSql data-stores can be classified into four categories: key-value, document-based, extensible record or column-based and graph-based. Many researches are analyzed and compared data models of these categories but there are few researches on API of them. In this paper, we implement python API to compare performance of key-value and document-based stores and then analyze the results.

[Ehsan Azizi Khadem, Emad Fereshteh Nezhad. **A Comparison of API of Key-Value and Document-Based Data-Stores with Python Programming.** *N Y Sci J* 2015;8(7):42-48]. (ISSN: 1554-0200). http://www.sciencepub.net/newyork. 7

Keywords: NoSql, DataBase, Python, Key-value, Document-based

1. Introduction

Databases re very important part of any software that is used by an organization or a simple user. For years, relational database management systems (RDBMS) has been the only solution for data engineers to design and implement a data store [1]. But in 21th century, some problems occur that RDBMS can't solve them completely. For example, in social networks we face Web2 applications. It means that we must store and retrieve user's activities and relations in addition to their profiles. Indeed, we need schema-less and very flexible database management system with simple replications, high availability, horizontal scaling and different access methods against RDBMS. NoSql (Not Only Sql) is the result of NoRel (Not Only Relational) that focuses on persistence, high availability and data partitioning based n distributed file systems like GFS, Hadoop, and Dynamo. Today there are more than fifty NoSql data-store systems available that each of them has its data model, characteristics, API, and etc[2][3]. But according to their data model, we can classified them into four categories:

- Key-Value: The key-value store is a simple model for NoSql databases. It stores pairs keys to values in the same way that an unique id is assigned to an object in some programming languages like Python. It provides a retrieval service like hash table and each node in the system can retrieve the value associated with the key. This model is used in caching content and main advantage of it is early access. Some of tools in this category are Redis, OracleBDB, Voldemort, TokyoVabinet/Tyrant, Riak, Memcached-DB[4].
- Document-Based: In this model, data stored in document format. Each document can consist of scalar values, metadata lists or even nested documents. This systems is uniquely named fields and of values can be different. There is no limitation in size of text and number of elements. The syntax of document is JSON or XML. This model is used in web applications and content representation in social networks. The advantage of it is resistant against incomplete content and weakness are slow querying and has no standard query language. Some of tools in this category are Couch-DB and Mongo-DB[5][6].
- Extensible Record or Column-Based: This model is a hybrid concept between RDBMS and document-based stores. Data is stored in columns and extensible rows. Extensible row means that each row has its own set of columns. Some terms like super-column and super-table are defined in this model. It is used in common distributed file systems like GFS and Hadoop. The advantages of it are fast searching and retrieval, and benefit distributed storing data and weakness is low level API. Some of tools, in this category are Cassandra, Hbase, and Riak[7].
- Graph-Based: When the data can be represented in the form of a graph with interlink elements, this model is mapped to graph theory in mathematics. For example, when we need to find shortest route between two persons in a social network like Linked-In, graph-based model do it perfectly. It is used in social networks and links FAQ. The advantages of the model are availability of graph algorithms like find shortest route, the connection's degree and weaknesses are navigating hole of graph to find results and difficult clustering. Some tools of this model are Neo4j, InfoGrid, InfiniteGraph[8][9].

In this paper, we peruse Redis as a key value store and MongoDB as a Document-Based store systems and implement an API for access the data in them by Python programming and then use the API for storing and retrieval several rows in these systems then compare and analyze the results[10][11].

2. A Key-Value Store DBMS: Redis

Redis is a famous and mostly used key-value store DBMS that easy to use. It has sophisticated set of commands and when comes to speed, is hard to bit. It supports advanced data structures, though on the degree document-based stores would. It provides set-based query operations but not with the granularity or type support in a relational database[12]. It is very fast. Redis is a blocking queue (or stack) and a publish-subscribe system. It is written with C programming language and its engine has tremendous speed. It is an in-memory key-value store and open source. It can handle up to 232 keys. Redis exposes five different data structures: strings, hashes, lists, sets, sorted sets. Strings are the mostly used basic data structure available in Redis. Hashes are like strings but the the important difference is that they provide an extra level of indirection: a field. Lists let you store and manipulate an array of values for a given key. You can add value to the list, get the first or last value and manipulate values at a given index. Sets are used to store unique values and provide a number of set-based operations. Sets aren't ordered but they provide value-based operations[13][14]. The most powerful data structure of Redis is sorted set. Sorted sets are like sets but with a score. The scores provides sorting and ranking capabilities. Redis supports replication, which means that as you write to one Redis instance (the master), one or more other instances (the slaves) are kept up-to-date by the master. Backing up Redis is very simple because by default Redis saves its snapshots to a file named 'dump.rdb'[15][16]. Redis distributes your keys across multiple Redis instances which could be running on the same box. It Not only will offer horizontal scaling, include rebalancing.

3. A Document-Base Store: MongoDB

MongoDB is a famous and popular NoSql DBMS. It is open source and written with Java. It has flexible data model (JSON) and rich query language. It supports auto-sharding and replication with automatic failure. It hasn't transaction nor joins. MongoDB is closer to MySql than other NoSql tools. It has main advantages over RDBMS. Any thing you can do in JSON, you can do in MongoDB. Within a MongoDB instance, you can have several databases, each acting as a high-level containers for everything else[17]. A database has some collections. A collection likes a table in relational model but it is a dynamic schema that can storing different data for several rows. Collections are made up 0 or more documents. Document is similar to row or tuple in relational model. A document is made up one or more fields which like a column in relational model. Indexes in MongoDB function mostly like their RDBMS counterparts. When you ask MongoDB for data, it returns a pointer to the result set called a cursor, which we can do things, to , such as counting or skipping ahead, before actually pulling down data[18]. You can use insert or create command to input a new row in a document of a collection and find command with a JSON query to retrieve data from it. When you want alternate a relational database with full text indexing, MongoDB is a good choice. An important benefit of document-based databases is schema-less. This makes then much more flexible than RDBMS because of lack of setup and reduced friction with object oriented programming. For example, Python's dynamism already reduce much of the object-relational impedance mismatch. So MongoDB is a very good match for Python. When you want to save an object, you serialize it to JSON and send it to MongoDB. There is no property mapping or type mapping. Another area where MongoDB is useful, is in logging. There are two aspects of MongoDB which make writes quite fast. First, you have an option to send a write command and have it return immediately without waiting for the write to be acknowledged. Secondly, you can control the write behavior with respect to data durability. These setting, in addition to specifying now many servers should get your data before being considered successful, are configurable pre-write, giving you a great level of control over write performance and data durability. In addition to these performance factors, log data is one of those datasets which can often take advantage of schema-less collections. MongoDB has something called a capped collection. Notice that all of the implicitly created collections are called normal collections. We can create a capped collection by using 'db. create collection' command and flagged it as capped. When you use a capped collection, you can update a document but it can't change its size. You can tail a capped collection the way you tail a file in Unix. Finally, you can run MongoDB in a multi-server setup[19].

4. An Object Oriented Programming Language: Python

Python is a general-purpose, interpreted, interactive, object oriented and high level programming language. It is open source. Python is easy to learn, read, maintain and has a broad standard library. It is portable and has cross-platform compatible on Unix, Windows, and Macintosh. It supports interactive mode and extendable. Python provides interfaces to all major commercial databases [20]. It supports GUI applications that can be created and ported to many system calls, libraries and windows systems. Python provides a better structure and support for large programs than shell scripting. It can be used as a scripting language or can be compiled to byte-code for building large applications. It has very high level dynamic data types and supports dynamic type checking and automatic garbage collection. It can easily integrate with C, C++, Java and etc. Python interpreter is written with C programming language. It offers much more error checking than C. Python has high level data types built in, such as flexible arrays and dictionaries. It allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your program or as examples to start to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like TK. Programs in Python are typically much shorter than equivalent C, C++ or Java programs because the high level data types allow you to execute complex operations in a single statement; Statement grouping is done by indentation instead of beginning and ending brackets and no variable or argument declarations are necessary [21]. Python is very suitable for using as an API programming language for NoSql because of some reasons. First, in Python each object has a unique id that generated by interpreter can be mapped on key in the key-value or in document-based store NoSql databases. With calling function id (object), you can retrieve the id of each object in Python. Second, Python has a data structure called dictionary. Python's dictionary are kind of hash table type. They would like associated arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type but are usually numbers of strings. Values, on the other hand, can be arbitrary Python objects. Dictionaries are enclosed by curly brackets ({}) and values can be assigned and accessed using square braces ([]). Python's dictionaries like documents in MongoDB and accept JSON format very well. So we can use Python for generate an API to manipulate data in MongoDB and Redis. For this, we can use some libraries known as connector. PyMongo is a connector for connect Python interpreter to MongoDB server. A library is called redis-py connect Python interpreter to Redis server. With these connectors, we can process the data stored in MongoDB or Redis with Python programs.[22]

5. Algorithm and Implementation:

We decide to design an algorithm and implementation a Python code to access to NoSql data stores and then compare the results. We want to compare performance and time consuming of similar operations on key-value and document-based stores. Redis is our choice from key-value stores and MongoDB from document-based stores. We implement a Python API to access each of them we want to investigate time consuming of our API in first: insert rows, second: retrieve rows and third: insert and retrieve rows simultaneously. We do it for 10 to 100000 rows and then compare the time duration of each execution. We use JetBrains PyCharm for coding and append these libraries with import command:

```
Import redis, pymongo, string, time
```

We import redis and pymongo for using redis-py and pymongo connectors that mentioned pervious section. Furthermore we import string to generate simple data for Redis and MongoDB and import time library for calculate time duration of each execution. For interaction with Redis, we write these statements:

```
r = redis.Redis()
start_time = time.time()
for i in range (10):
r.set(i,i*2)
m=r.get(i)
print("Redis Time = " ,time.time()-start_time)
```

First, we create an instance of Redis called 'r'. Then we determine time of start of accessing data in Redis instance and assign it to 'start_time'. Next step, we implement a 'for' loop with count value that is variable between 10 to 100000 and in the 'for' block, we write 'set' and 'get' command that can be converted to comment with #. When we want to execute set and get together, we don't write # but when we only want to store, write # before get and when only want to retrieve, write # before set. Finally, we print time duration of executions that is current time minus start time. For interaction with MongoDB, we write these statements:

```
conn = pymongo.MongoClient()
start_time = time.time()
db = conn.test1
coll = db.collection1
for i in range (10):
coll.insert({str(i):i*2})
cc = coll.find_one({str(i):i*2})
print("Mongo Time = " , time.time()-start_time)
```

First, we create an instance of MongoDB client called 'conn' and determine 'start_time' with calling time() method, from time library. Then we create a database in MongoDB called 'db' and create a collection in it called 'coll'. Next step, we implement a 'for' loop like that was implemented for Redis. The storing and retrieving statements in pymongo and redis-py is different. In pymongo, 'insert' is for storing and 'find' is for retrieving. Strings must be in JSON format.

6. Results:

In this section, we peruse result of running the algorithm mentioned in pervious section. First, we study the results of only storing data for several numbers of rows. See table 1 and figure 1:

Table 1: Set and Insert

Number of Rows	10	100	1000	10000	50000	100000
Redis Time (Seconds)	1.005	1.029	1.121	1.981	6.209	10.917
MongoDB Time (Seconds)	0.006	0.023	0.191	1.776	9.138	17.762

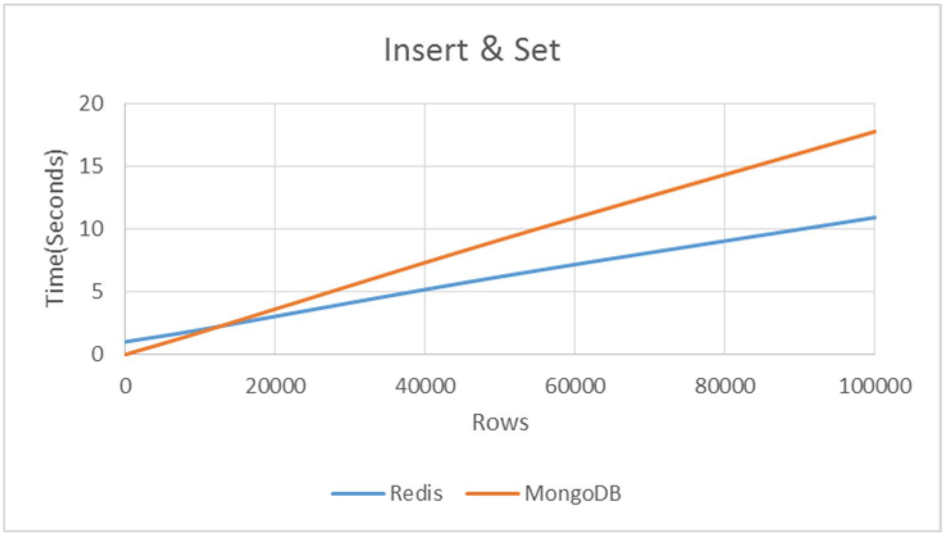

Figure 1: Set and Insert Diagram

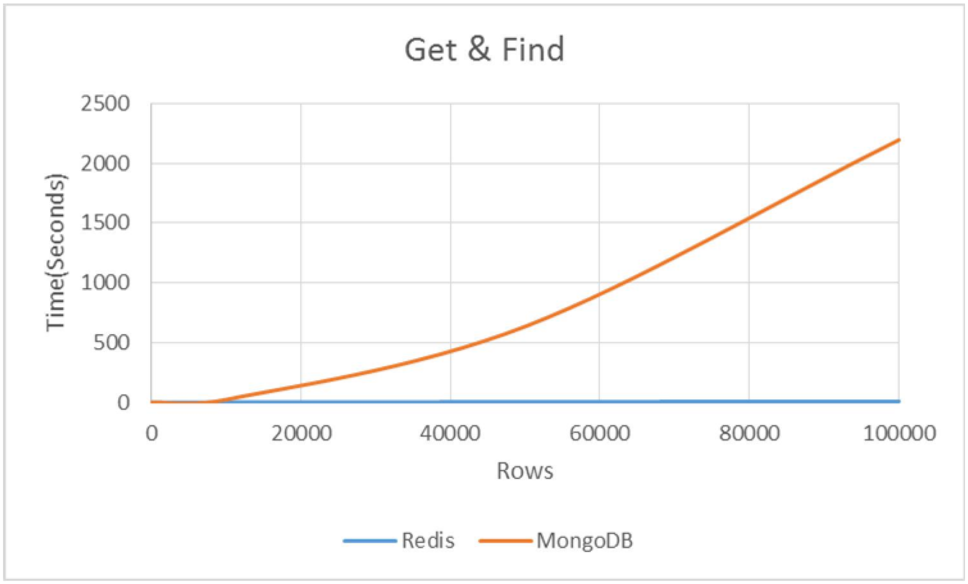

Figure 2: Get and Find Diagram

MongoDB is faster than Redis in storing few number of rows (less than 20000 rows). But when number of rows is increasing, Redis stores data faster. So for huge volume of data, Redis has higher speed than MongoDB when using same Python API. Also the runtime difference isn't very significant.

Second, we check the result of only retrieving data for several numbers of rows. See table 2 and figure 2:

Table 2: Get and Find

Number of Rows	10	100	1000	10000	50000	100000
Redis Time (Seconds)	1.002	1.029	1.122	1.936	5.584	10.022
MongoDB Time (Seconds)	0.006	0.035	0.439	24.344	635.444	2195.227

If you have less than 1000 rows, MongoDB retrieve data faster but it isn't Big Data! For more than 1000 rows, Redis retrieves faster and for more than 50000 rows, Redis works very very faster! For huge amount of data, if you want to retrieve data with Python API, Redis have higher speed than MongoDB and the speed difference is surprisingly high. Redis more than 200 time faster than MongoDB for 100000 rows. Time consuming grows exponentially for MongoDB in huge number of data and it means Redis works with Python API very faster and is a better choice to implement a retrieval system than MongoDB.

Third, we study speed of Python API for Redis and MongoDB for storing and retrieving data simultaneously. See table 3 and figure 3:

Table 3: Set and Insert - Get and Find

Number of Rows	10	100	1000	10000	50000	100000
Redis Time (Seconds)	1.011	1.053	1.273	3.103	10.639	19.339
MongoDB Time (Seconds)	0.1	0.057	0.739	26.839	635.295	2205.618

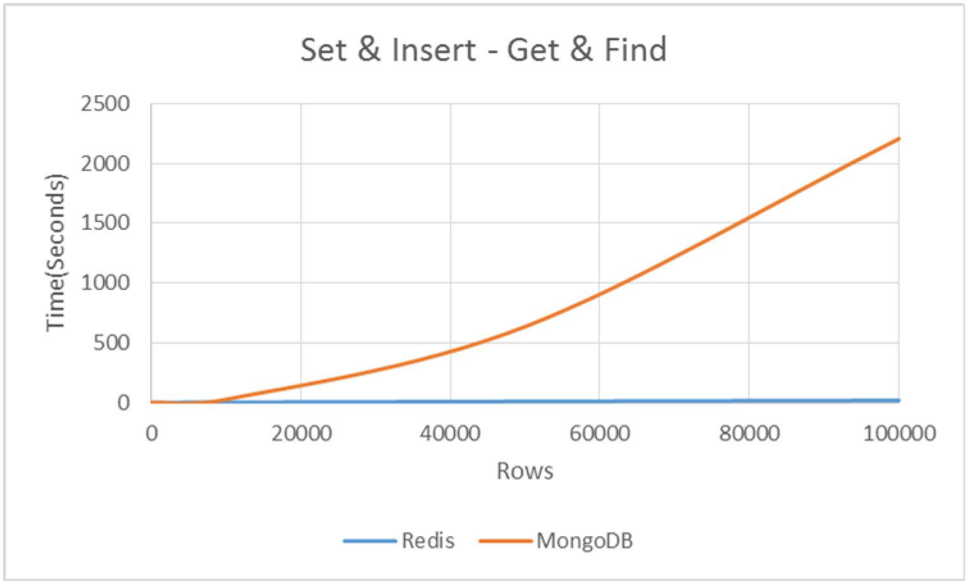

Figure 3: Set and Insert - Get and Find Diagram

For less than 5000 rows, MongoDB is a little faster, but for more than 10000 rows, Redis is very faster and for more than 50000 rows, Redis has much higher speed than MongoDB. For 100000 rows, Redis is more than 100 times faster than MongoDB.

The results obtained from the above steps are:

- When we want to retrieve data from huge sets of information with Python API, Redis is an optimize option compared with MongoDB and this is independent from that we want to store data or not.
- When we only want to store data, Redis is still better than MongoDB for huge volume of data but they don't have much difference. Also for small data sets, MongoDB works better.

7. Conclusion and Future Works:

Nowadays, NoSql data stores are very important technologies for storing and retrieving data. Partitioning the data in several servers, persistence of data and consistency are three goals that Big Data and NoSql technologies follow them. NoSql technology is used when we have huge sets of data for processing. There are several categories of NoSql data models: key-value, document-based, column-based and graph-based. We investigated key-value and document-based in this research. From key-value store, we choose Redis and from document-based store choose MongoDB. We want to study about their behavior when a same API used for them because most or researches are about their data models. Our API is implemented with Python programming language. When our algorithm is implemented and executed, we see some arresting results. In data retrieval, Redis is more than 200 times faster than MongoDB in interaction with API. In data storing, Redis is better when we want to store huge volume of data but the difference isn't significant. Future, we will study about speeds of other data stores in addition to key-value and document-based and their memory consuming.

References:

1. Yan Carri_ere-Swallow and Felipe Labb_e. Nowcasting with Google Trends in an emerging market. Journal of Forecasting, 2011. doi: 10.1002/for.1252. URL http://ideas.repec.org/p/chb/bcchwp/588.html. Working Papers Central Bank of Chile 588.
2. Sharad Goel, Jake M. Hofman, Sbastien Lahaie, David M. Pennock, and Duncan J. Watts. Predicting consumer behavior with web search. Pro-ceedings of the National Academy of Sciences, 2010. URL http://www.pnas.org/content/107/41/17486.full.
3. Rebecca Hellerstein and Menno Middeldorp. Forecasting with internet search data. Liberty Street Economics Blog of the Federal Reserve Bank of New York, January 2012. URL http://libertystreeteconomics.newyorkfed.org/2012/01/forecasting-with-internet-search-data.html.
4. Brewer, E. A. 2000. Towards robust distributed systems (abstract). In Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing (Portland, Oregon, United States, July 16 - 19, 2000). PODC '00. ACM, New York, NY, 7. DOI= http://doi.acm.org/10.1145/343477.343502.
5. Gilbert, S. and Lynch, N. 2002. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News 33, 2 (Jun. 2002), 51-59. DOI=http://doi.acm.org/10.1145/564585.564601.
6. Pritchett, D. 2008. BASE: An Acid Alternative. Queue 6, 3 (May. 2008), 48- DOI=http 55.://doi.acm.org/10.1145/1394127.1394128
7. A. Lakshman, P. Malik, and K. Ranganathan. Cassandra: A Structured Storage System on a P2P Network, product presentation at SIGMOD 2008.
8. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall,P., and Vogels, W. 2007. Dynamo: amazon's highly available key-value store. In Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles (Stevenson, Washington, USA, October 14 - 17, 2007). SOSP '07. ACM, New York, NY, 205-220. DOI=http://doi.acm.org/10.1145/1294261.1294281.
9. Morris, R. 1968. Scatter storage techniques.Commun. ACM 11, 1 (Jan. 1968), 38-44. DOI= http://doi.acm.org/10.1145/362851.362882.
10. Agrawal, P., Silberstein, A., Cooper, B. F., Srivastava, U.,and Ramakrishnan, R. 2009. Asynchronous view maintenance for VLSD databases. In Proceedings of the 35th SIGMOD international Conference on Management of Data (Providence, Rhode Island, USA, June 29 - July 02, 2009). C. Binnig and B.Dageville, Eds. SIGMOD '09. ACM, New York, NY, 179-192.DOI= http://doi.acm.org/10.1145/1559845.1559866.
11. Message in Redis mailing list http://groups.google.com/group/redisdb/msg/ca398a90ea78bfc5.
12. Armbrust, M., Lanham, N., Tu, S., Fox, A., Franklin, M., and Patterson, D. A. Piql: A performance insightful query language for interactive applications. First Annual ACM Symposium on Cloud Computing (SOCC).
13. Acharya, S., Carlin, P., Galindo-Legaria, C., Kozielczyk, K., Terlecki, P., and Zabback, P. 2008. Relational support for flexible schema scenarios. Proc. VLDB Endow. 1, 2 (Aug. 2008), 1289-1300. DOI= http://doi.acm.org/10.1145/1454159.1454169.
14. Avrilia Floratou, Nikhil Teletia, David J. DeWitt, Jignesh M. Patel, Donghui Zhang, *Can the elephants handle the NoSQL onslaught?*, Proceedings of the VLDB Endowment, VLDB Endowment Hompage archive, Volume 5 Issue 12, August 2012, Pages 1712-1723.
15. Bogdan Tudorica, Bucur Cristian – A comparison between several NoSQL databases with comments and notes,The proceedings of "2011 – Networking in Education and Research" IEEE International Conference, June

23, 2011– June 25, 2011, Alexandru Ioan Cuza University from Iasi.
16. Bogdan Tudorica - Challenges for the NoSQL systems: Directions for Further Research and Development, The International Journal of Sustainable Economies Management (IJSEM), Volume 2: Issue 1 (2013), DOI:10.4018/IJSEM.2013010106, ISSN:2160-9659, EISSN: 2160-9667.
17. MongoDB. http://www.mongodb.org. Accessed 2013.
18. Banker, Kyle ; Chodorow, Kristina ; Merriman, Dwight et al.: *mongoDB Manual – Admin Zone – Replication – Replica Sets – Replica Set Tutorial*. August 2010. – Wiki article, version 22 of 2010-08-11. http://www.mongodb.org/display/DOCS/Replica+Set+Tutorial.
19. Banker, Kyle; Merriman, Dwight ; Horowitz, Eliot: *mongoDB Manual – Admin Zone Replication – Halted Replication*. August 2010. – Wiki article, version 18 of 2010-08-04. http://www.mongodb.org/display/DOCS/Halted+Replication.
20. Copeland, Rick: *How Python, TurboGears, and MongoDB are Transforming Source- Forge.net*. Feburary 2010. – Presentation at PyCon in Atlanta on 2010-02-20. http://us.pycon.org/2010/conference/schedule/event/110/.
21. Python Software Foundation, 2012. Python/C API Reference Manual. Python Software Foundation.
22. Vanovschi, V., 2013. Parallel python software. http://www.parallelpython.com. Accessed: 10/02/2013.

7/17/2015