نظا

جامعة السودان للعلوم والتكنولوجيا
كلية علوم الحاسوب وتقانة المعلومات

تطوير نظام لإدارة البيانات الضخمة بإستخدام الأدوات المفتوحة المصدر

Develop a System for Managing Big Data using Open Source Tools

بحث مقدم كأحد متطلبات الحصول على درجة الشرف في علوم الحاسوب / هندسة البرمجيات

أكتوبر 2018

بسم الله الرحمن الرحيم

جامعة السودان للعلوم و التكنولوجيا

كلية علوم الحاسوب و تقانة المعلومات

قسم علوم الحاسوب/هندسة البرمجيات

تطوير نظام لإدارة البيانات الضخمة بإستخدام الأدوات المفتوحة المصدر

Develop a System for Managing Big Data using Open Source Tools

إعداد :

- **رهام الهادي عبدالله مصطفى**
- **محمد أدم الخير محمد الشريف**
- **ياسمين فتحي إبراهيم حسن**

بحث مقدم كأحد متطلبات الحصول على درجة الشرف في علوم الحاسوب / هندسة البرمجيات

إشراف : د.هشام عبدالله منصور

التاريخ .../.../..... **التوقيع.....**

الآية

قال تعالى :

لَهُ اللهُ ﴿إِلاَّ بُشْرَى لَكُمْ وَلِتَطْمَئِنَّ قُلُوبُكُم بِهِ
نْدِ اللهِ الْعَزِيزِ الْحَكِيمِ ﴾

صدق الله العظيم

سورة آل عمران الأية (126)

الأهداء

إلي من بها أكبر و عليها أعتمد

إلى شمعة متقدة تنير ظلمة حياتي

إلى من أرضعتني الحب و الحنان

إلى رمز الحب و بلسم الشفاء

إلى القب الناصع بالبياض

أمهاتنا الفاضلات : سلوى ، سهير و نعمات

إلي من أحمل اسمه بفخر

إلى من جرع الكأس فارغاً ليسقيني قطرة حب

إلى من كلّت أنامله ليقدم لنا لحظة سعادة

إلى من حصد الأشواك عن دربي ليمهد لي طريق العلم

أبائنا الأفاضل : آدم ، الهادي و روح الوالد فتحي

إلي سندي و قوتي و ملاذي بعد الله

إلى من أثروني على نفسهم

إلى من أظهرو لي ما هو أجمل من الحياة **"إخوتي"**

إلى من تذوقت معهم أجمل اللحظات

إلى من سأفتقدهم و أتمنى أن يفقدوني

إلى من جعلهم الله أخوتي بالله .. ومن أحببتهم في الله **"زملائي و زميلاتي"**

شكر و عرفان

الحمدلله الذي وفقنا لهذا البحث و لم نكن لنصل إليه لولا فضل الله علينا

ثم الشكر أجزله للدكتور هشام عبدالله منصور الذي تفضل بإشرافه على هذا البحث ، فجزاه الله عنا كل خير و الذي نقول له بشراك الرسول صلى الله عليه و سلم :

"إن الحوت في البحر، و الطير في السماء ليصلون على معلم الناس الخير"

كما نخص بالشكر سارة حسبو – مركز النيل على مساعدتها و تقديم العون لنا و تزويدنا بالمعلومات اللازمة لإتمام هذا البحث .

المستخلص

في السنوات الاخيرة اصبحت البيانات تتولد بكميات ضخمة بعد دخول مواقع التواصل الاجتماعي والتجارة الألكترونية وعمليات البحث اليومية وعمليات دراسة السلوك..الخ . وأن هذه البيانات يستفاد منها بطرق شتى من الناحية التجارية فإن الشركات بدأت تولي اهتماماً لبياناتها والبيانات التي تتحصل عليها حيث أن هذه البيانات تساعد الشركة في فهم سلوك عملائها وتحليل جودتها وتحسين سمعتها بشكل عام، ولكن التعامل مع هذه البيانات الضخمة ليس بالأمر السهل إذ أن أدوات إدارة البيانات التقليدية لا تستطيع إدارة هذه البيانات بالكفاءة المطلوبة حيث أن هذه الأنظمة تسعى لتحقيق التكاملية والإتساق مما يقلل بشكل واضح من إتاحية البيانات في أغلب الأوقات وأن هذا الأمر قد يزعج العميل بشكل أو بأخر، ولذلك بدأت هذه الشركات في البحث عن أنظمة أخرى أفضل تحقق إتاحية أكبر إلى حد مرضي للعملاء ولكن الانظمة التي ظهرت لحل هذه المشكلة تضحي بالإتساق مما قد يؤدي إلى خسارة الشركة ، وأن معظم الشركات تضطر للإختيار ما بين الإتساق والإتاحية أي ما بين رضا العملاء وربحية الشركة.

في هذا البحث تم اقتراح وتصميم معمارية جديدة توفر حلاً متوازناً ما بين الإتساق والإتاحية ’ حيث تم الاعتماد في المعمارية المقترحة على أدوات مفتوحة المصدر ومجانية و تم ربطها مع بعضها البعض لإيجاد نظام إدارة البيانات المناسب، ومن ثم تم اختبار النتائج بإستخدام الأداة المطورة في هذا البحث RYM للتأكد من أن النظام يعمل بالكفاء المطلوبة.

Abstract

Data in recent years are being generated in large quantities after the entry of social networking sites, e-commerce, daily searches and behavioral study processes etc. These data are used commercially in various ways.Companiesarebeginning to pay attention to their data and the data acquired because this data help the company to: study their Customer behaviors, quality analysis and overall reputation improvement, but dealing with these large data is not easy. Traditional data management tools cannot manage this data efficiently, as these systems strive for Integrity and consistency, which clearly reduces the availability of data in most of the times and this may bother the customer in one way or another, and therefore began to search for other systems that achieve more satisfactory to the customers, but the systems that emerged to resolvethis problem sacrifices consistency, which may lead to the company loss, and most companies have to choose between consistency and availability, that means between customer satisfaction and profitabilityof the company .

In this research a new architectural design proposal is proposed to provide a balanced solution between consistency and availability. This architecture it will rely on free and open source tools and then integrate them together to create the appropriate data management system then

test the resultsusing the tool RYM developed in thisresearch to ensure that the system works efficiently.

فهرس الأشكال

رقم الشكل	موضوع الجدول	رقم الصفحة
(1.1.2)	خصائص البيانات الضخمة الأربعة 4V	5
(1.3.2)	أمثلة لقواعد البيانات غير العلائقية	9
(2.3.2)	نظرية CAP	10
(3.3.2)	معمارية HDFS	12
(4.3.2)	معمارية Map Reduce	13
(1.4.2)	يوضح الفرق في الأداء بين Cassandra و Riak	15
(2.4.2)	يوضح تحسن زمن الوصول عند استخدام تقنية IRM	16
(3.4.2)	المعمارية المقترحة لتوفير قدر عالي من الإتاحية	17
(4.4.2)	مثال لكيفية حساب الـstaleness	19
(1.3)	المنهجية المتبعة في بناء النظام المقترح	21
(2.3)	هيكل بيانات Mongo DB	22

(3.3)	مثال لـ MongoDB document	23
(4.3)	عمليتي التوزيع و الاستنساخ في Mongo DB	23
(1.4)	معمارية النظام المقترح	27
(1.5)	خوارزمية حساب الـLatency و الـ Throughput	36
(1.6)	يوضح الـstaleness عند إجراء عملية Update على 1Thousand Records عند حجم بيانات 50GB	41
(2.6)	يوضح الـstaleness عند إجراء عملية Update على10Thousand Records عند حجم بيانات 50GB	41
(3.6)	يوضح الـstaleness عند إجراء عملية Update على100Thousand Records عند حجم بيانات 50GB	42
(4.6)	يوضح الـstaleness عند إجراء عملية Update على 1Million Records عند حجم بيانات 50GB	42
(5.6)	ي يوضح الـstaleness عند إجراء عملية Update على 20Million Records عند حجم بيانات 50GB	43
(6.6)	يوضح الـstaleness عند إجراء عملية Update على 1Thousand Records عند حجم بيانات 100GB	44
(7.6)	يوضح الـstaleness عند إجراء عملية Update	44

	على10Thousand Records عند حجم بيانات 100GB	
(8.6)	يوضح الـstaleness عند إجراء عملية Update على100Thousand Records عند حجم بيانات 100GB	45
(9.6)	يوضح الـstaleness عند إجراء عملية Update على 20Million Records عند حجم بيانات 100GB	45

فهرس الشاشات

رقم الشاشة	موضوع الشاشة	رقم الصفحة
(1.5)	ملف Hosts	29
(2.5)	خطوات Secure Shell	29
(3.5)	ملف Core-site	30
(4.5)	ملف hdfs-site	30
(5.5)	توضح تهيئة الـ hdfs	31
(6.5)	توضح تهيئة الأمر start– dfs.sh	31
(7.5)	توضح تهيئة الأمر start- yarn.sh	32
(8.5)	ملف الاعدادات mongo.conf	32
(9.5)	توضح خطوات تحديد الـMaster في MongoDB Replica Set	33
(10.5)	توضح تحديد بقية الـ Nodes في MongoDB Replica Set	33
(11.5)	توضح كتابة واستيراد البيانات من والي الـ MongoDB باستخدام الـMap Reduce	34

فهرس الجداول

رقم الجدول	موضوع الجدول	رقم الصفحة
(1.2.2)	أمثلة على برامج مفتوحة المصدر	7
(1.3.2)	مقارنة بين قواعد البيانات العلائقية و غير العلائقية	10
(1.4.2)	طرق عملية الإستنساخ	18
(2.4.2)	امثلة لقواعد البيانات و الخواص التي توفرها من نظرية PACELC	19
(3.4.2)	أوجه التشابه و الإختلاف بين البحث و الدراسات السابقة	20
(1.5)	مواصفات العقد	37
(1.6)	يوضح الـThroughput لكل عنقود عند حجم بيانات 50GB	38
(2.6)	يوضح الـLatency لكل عنقود عند حجم بيانات 50GB	39
(3.6)	يوضح الـThroughput لكل عنقود عند حجم بيانات 100GB	39
(4.6)	يوضح الـLatency لكل عنقود عند حجم بيانات 100GB	40
(5.6)	يلخص نتائج الـstaleness عند حجم بيانات 50GB	43
(6.6)	يلخص نتائج الـstaleness عند حجم بيانات 100GB	46

جدولالمحتويات

الآية.....أ

الأهداء.....ب

شكروعرفان.....ج

المستخلص.....د

Abstract.....ه

فهرسالأشكال.....و

فهرسالشاشات.....ط

فهرسالجداول.....ي

الباب الأول : المقدمة

1.1تمهيد:.....1

2.1 مشكلةالبحث :.....1

3.1 أهميةالبحث:.....1

4.1 أهدافالبحث:.....2

5.1 حدودالبحث:.....2

6.1 منهجيةالبحث:.....2

7.1 فروضالبحث:.....2

8.1 هيكليةالبحث:.....2

الباب الثاني : الخلفية النظرية

الباب الثاني : المبحث الأول : إدارة البيانات و البيانات الضخمة

1.2 المقدمة.....3

1.1.2 البياناتData.....3

1.1.1.2 أنواعالبيانات.....3

2.1.1.2 مصادرالبيانات.....3

2.1.2 المعلوماتInformation.....3

3.1.2 ادارةالبياناتData Management.....3

4.1.2 ادارةالمعلوماتInformation Management.....4

5.1.2 قواعدالبياناتDatabase.....4

7.1.2 قواعدالبياناتالمركزيةCentralized Database.....4

8.1.2 معالجةالبياناتData Processing.....4

1.8.1.2 المعالجةالمتوازيةParallel Processing.....4

9.1.2 البياناتالضخمةBig Data 4
1.9.1.2 خصائصالبياناتالضخمة 5
2.9.1.2 مصادرالبياناتالضخمة 5

الباب الثاني : المبحث الثاني: الأنظمة المفتوحة المصدر

2.2المقدمة 6
1.2.2شفرةأوكودالمصدرsource code 6
2.2.2 الأنظمةمغلقةالمصدرclosed source software 6
3.2.2 الأنظمةمفتوحةالمصدرopen source software 6
1.3.2.2 معاييرالأنظمةمفتوحةالمصدر 6
2.3.2.2 مزاياالأنظمةمفتوحةالمصدر 7

الباب الثاني : المبحث الثالث: قواعد البيانات غير العلائقية و بيئة Hadoop

3.2 المقدمة 8
1.3.2 قواعدالبياناتالغيرعلائقيةNO SQL Database 8
2.3.2 اقسامقواعدالبياناتالغيرعلائقية 8
3.3.2 الإتساقونظريةCAP 9
4.3.2 قواعدالبياناتالعلائقيةالجديدة NewSQL:- 11
5.3.2 بيئةنظام Hadoop :- 11
6.3.2 مكوناتنظامHadoop :- 12

الباب الثاني : المبحث الرابع :الدراسات السابقة

4.2المقدمة 14
1.4.2الدراسةالأولى: 14
2.4.2الدراسةالثانية: 14
3.4.2الدراسةالثالثة: 15
4.4.2الدراسةالرابعة: 16
5.4.2 الدراسةالخامسة : 16
6.4.2الدراسةالسادسة: 17
7.4.2الدراسةالسابعة: 17
8.4.2 الدراسةالثامنة : 19

الباب الثالث: منهجية البحث

1.3 المقدمة : 21
2.3 المنهجية : 21

3.3 المصادرالمفتوحةالمستخدمةفيالبحث: 22
1.3.3 قاعدةالبياناتMongoDB : 22
2.3.3أداةالقياسYCSB 24
3.3.3 Mongo-hadoop connector 24
4.3.3 Hadoop 25

الباب الرابع: تحليل و تصميم النظام

1.4متطلباتالنظام: 26
1.1.4المتطلباتالوظيفية: 26
2.1.4المتطلباتغيرالوظيفية: 26
2.4تحليلالمتطلبات: 26
3.4 معماريةالنظام: 27

الباب الخامس: تطبيق و اختبار النظام

1.5مقدمة: 29
2.5 تهيئةبيئةعملالـHadoop: 29
1.2.5 ربطالعقدNodes : 29
3.2.5 تهيئةالـHDFS : 31
4.2.5تشغيلالـHadoopفيالـcluster: 31
3.5 إعدادبيئةتشغيلMonogDB 32
1.3.5 إعدادبيئةMongoDBللعملفيمجموعةعملReplica Set : 32
2.3.5 إنشاءالـMaster : 33
3.3.5 إضافةبقيةالـNodesإلىالـReplica Set : 33
4.5 ربطبيئتيالعملMongoDBوHadoop : 34
1.4.5 إضافةملفاتالربطإلىمكتباتالـHadoopوالـMap Reduce: 34
2.4.5 كتابةوإستيرادالبياناتمنوإلىالـMongoDBبإستخدامالـ Map Reduce : 34
5.5 اختبارالنظام : 35

الباب السادس: النتائج و التوصيات

1.6المقدمة:- 38
2.6النتائج:- 38
3.6التوصيات: 46
4.6الخاتمة: 46
المصادروالمراجع 47

الباب الأول

المقدمة

1.1تمهيد:

مع تطور التقنيات وإقتحام التكنولوجيا لمختلف المجالات وإعتماد الأنظمة في معظم العمليات وفي الحياة اليومية أصبحت هذه الانظمة تولد كميات كبيرة من البيانات بشكل يومي وظهر مصطلح البيانات الضخمة ، وبدأت الدراسات حول كيفية التعامل مع هذه البيانات بطرق جديدة أكثر فعالية لبطء أنظمة إدارة البيانات الموجودة وعدم قدرتها للتعامل مع هذه البيانات بالكفاءة المطلوبة ، وتم إنشاء قواعد بيانات تسمى بقواعد البيانات الغير علائقية NoSqlللتعامل مع هذه البيانات الضخمة . وقد لوحظ أن قواعد البيانات هذه توفر قدر عالي من الإتاحة Availability وإمكانية التوسع أو التقسيمpartition-toleranceولكنها تضحي بجوانب أخرى كالإتساقConsistency ، وبعض الانظمة تولد بيانات ضخمة والتي بدورها تحتاج إلى درجة عالية من الإتساق ، ومن هنا ظهرت نظريات تدرس هذه الظاهرة مثل نظرية الـCAP، والتي على أساسها تم تقسيم هذه الانظمة إلى أنظمة توفر الإتاحية وإمكانية التقسيم و أنظمة توفر التقسيم والإتساق ، ولكن هناك نوع جديد من التقنيات التي تسمى قواعد البيانات العلائقية الجديدةNewSql والتي تحاول أن توفر الإتاحية وقابلية التوسع والأساق معاً ، ونجد أن عدد من الشركات التجارية مثل Oracle قد نجحت في تطبيق هذه التقنية وتوفيرها كخدمة سحابية.

كما تم إنشاء عدد من بيئات العمل لمعالجة وتحليل هذه البيانات والإستفادة منها في إتخاذ القرارات، إكتشاف الأنماط وفي تحسين العمل بشكل عام ، و من هذه الأنظمة ما هو تجاري ومنها ما هو مجاني ومنها ماهو مفتوح المصدر ومنها ما هو غير مفتوح المصدر .

2.1 مشكلة البحث :

تتميز بعض التطبيقات بتوليد بيانات ضخمة والحلول المطروحة لإدارة مثل هذه البيانات بعضها ذات تكلفة عالية والبعض الأخر يعطى أولوية للإتاحية Availability على الإتساق Consistency .

3.1 أهمية البحث:

توفير نظام ذا كفاءة عالية ومقدرة على إدارة البيانات الضخمه ، قادر على مقابلة إحتياجات سوق العمل المحلى والإقليمى والعالمى بتكلفة أقل.

4.1 أهداف البحث:

1. تطوير نظام يقوم بمعالجة البيانات بإستخدام نماذج تتوافق مع البيانات الضخمة شبه المهيكلة.
2.توفير حل يوازن ما بين الإتساق والإتاحية بأقل تكلفه.

5.1 حدود البحث:

تصميم و تطبيق نظام مبني على المصادر المفتوحه للتعامل مع البينات الضخمه ثم توليد البيانات و من ثم إدخالها على النظام المقترح وإجراء القياسات المطلوبة عليها وتحليل النتائج ، ثم قياس أداء النظام بإجراء نموذج عمليات على الـMapReduce .

6.1 منهجية البحث:

- إجراء إستقراء بحثي لتحديد الحل المناسب والأدوات التي ستستخدم من الدراسات السابقة
- تصميم معمارية النظام .
- تطبيق و إختبار النظام
- تحليل النتائج.

7.1 فروض البحث:

بناء نظام موزع متوازى ذا كفاءة عاليه يوازن بين الإتساق والإتاحية بإستخدام الأدواتالمفتوحة المصدر .

8.1 هيكلية البحث:

يتكون البحث من ستة أبواب بالإضافة إلى الأجزاء التمهيدية و الختامية على النحو الأتي :

- **الباب الأول:** يتناول هذا الباب المقدمة و يتمثل في مشكلة البحث ، أهدافه و أهميته .
- **الباب الثاني:** يتكون هذا الباب من أربعة مباحث ، المبحث الأول يتناول البيانات و إدارة البيانات لأن هذا البحث جزء من علم إدارة البيانات و البيانات الضخمة ، و المبحث الثاني يتناول الأنظمة مفتوحة المصدر ، المبحث الثالث يتناول بيئة عمل Hadoop و قواعد البيانات غير العلائقية و المبحث الرابع يتناول الدراسات السابقة .
- **الباب الثالث:** يتناول هذا الباب منهجية البحث و المصادر المفتوحة المستخدمة فيه .
- **الباب الرابع:** يتناول تحليل و تصميم النظام المقترح .
- **الباب الخامس:** يتناول تطبيق و اختبار النظام .
- **الباب السادس:** يتناول النتائج و التوصيات و الخاتمة .

الباب الثاني

الخلفية النظرية

مدخل:

هذا الباب سيتناول الخلفية النظرية و المفاهيم العامة المتعلقة بالبحث و كذلك سيتطرق الباب للدراسات السابقة ذات العلاقة و يحتوي البحث على أربع مباحث .

حيث يتناول **المبحث الأول** إداراة البيانات و البيانات الضخمة و يتناول **المبحث الثاني** قواعد البيانات الغيرعلائقية و مقارنة بينها و بين قواعد البيانات العلائقية بالاضافة الي بيئة عمل Hadoop و يتناول **المبحث الثالث**المصادر المفتوحة المصدر و اخيراً يتناول **المبحث الرابع** الدراسات السابقة ذات الصلة .

المبحث الأول

إدارة البيانات و البيانات الضخمة

Data Management & Big Data

1.2 المقدمة

هذا المبحث يقوم بالتعرف على البيانات و أقسامها و مصادرها و كيفية تحويلها لمعلومات يمكن الاستفادة منها في إتخاذ القرارات الهامة اعتماداً على تراكمها ، و كلما زاد حجم البيانات زادت كفاءة اتخاذ القرار .

1.1.2 البياناتData

هي مجموعة الحقائق الخام والقياسات والمشاهدات التي تكون على شكل أرقام وحروف ورموز وأشكال خاصّة ، فالبيانات لا يكون لها معنى، لهذا يتم تجميعها و تنظيمها و تفسيرها و معالجتها حتى يتم استخدامها.[1]

1.1.1.2 أنواع البيانات

1- بيانات مهيكلية (structured data) : وهي البيانات التي يمكن تنظيمها في الجداول ذات أنواع بيانات data types .

2- بيانات غير مهيكلة (unstructured data): و هي مثل الصور و الفيديو.

3- بيانات شبه مهيكلة (semi structured): و تتمثل في ملفات xml و josn . [2]

2.1.1.2 مصادر البيانات

هو المكان الذي تتواجد أو تأتي منه البيانات المستخدمة فأما أن تكون قواعد بيانات تخزن البيانات أو اجهزة مراقبة و إستشعار تقوم بتوليد البيانات . [1]

2.1.2 المعلومات Information

هي البيانات التي تمّ معالجتها بتصنيفها وتنظيمها وتحليلها، وأصبح لها معنى لتحقق هدف معين وتُستعمل لغرض معيّن حتى توفّر ما يسمى بالمعرفة. [1]

3.1.2 ادارة البيانات Data Management

هي عملية إدارية تتضمن الحصول على البيانات المطلوبة والتحقق منها وتخزينها وحمايتها ومعالجتها لضمان إمكانية الوصول إلى البيانات وموثوقيتها وتوقيتها بالنسبة لمستخدميها. [3]

4.1.2 ادارة المعلوماتInformation Management

هي جمع المعلومات وتخزينها ونشرها وحفظها وتدميرهافهو يمكّن الفُرق وأصحاب المصلحة من استغلال وقتهم ومواردهم وخبراتهم بفعالية في إتخاذ القرارات والقيام بأدوارهم. [1]

5.1.2 قواعد البيانات Database

هي مجموعة من المعلومات التي تم تنظيمها بحيث يمكن الوصول إليها وإدارتها وتحديثها بسهولة. [2]

6.1.2 قواعد البيانات الموزعة Distributed Database

هي مجموعةمتعددة من قواعد البيانات المترابطة منطقياً وموزعة على شبكة الحواسيب. [2]

7.1.2 قواعد البيانات المركزية Centralized Database

هي قاعدة بيانات موجودة في موقع ، مخزنة و يمكن تعديلها في مكان واحد . هذا الموقع في الغالب يكون جهاز حاسوب مركزي او قاعدة بيانات. [2]

8.1.2 معالجة البيانات Data Processing

عبارة عن سلسلة من العمليات على البيانات ، لإسترداد المعلومات أو تحويلها أو تصنيفها . بصورة أخرى هي عبارة عن جمع ومعالجة عناصر البيانات لإنتاج معلومات مفيدة[1].

1.8.1.2 المعالجة المتوازيةParallel Processing

هي طريقة لتقسيم مهام البرامج وتشغيلها في وقت واحد على معالجات متعددة ، وبالتالي تقليل وقت المعالجة .يمكن إنجاز المعالجة المتوازية عبر الحاسوب الذي يحتوي على معالجين أو أكثر أو عبر شبكة حاسوب. [1]

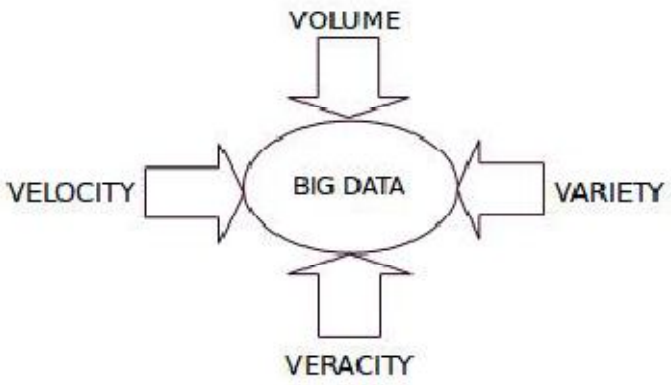
9.1.2 البيانات الضخمة Big Data

هيمصطلح يشير الي مجموعات البيانات أو مجموعات من مجموعات البيانات التي يجعل حجمها (volume) وتعقيدها (variability) ومعدل نموها (velocity) و دقتها (Veracity) من الصعب

التقاطها أو إدارتها أو معالجتها أو تحليلها بواسطة التقنيات والأدوات التقليدية مثل قواعد البيانات العلائقية وإحصاءات سطح المكتب أو حزم التصور ، في الوقت اللازم لجعلها مفيدة[4].

1.9.1.2 خصائص البيانات الضخمة

1- **الحجم volume** : يعني حجم البيانات أو الكمية الكبيرة من البيانات التي يتم توليدها في كل ثانية و يجب أن تفوق التيرابايت.
2- **السرعة Velocity** : و هي السرعة التي يتم بها توليد البيانات.
3- **الاختلاف Variety** : و يشير إلى تنوع البيانات المتولدة بين مهيكلية و غير مهيكلية و شبه مهيكلية.
4- **الدقة Veracity** : وتعني دقة البيانات, حيث أن البيانات غير دقيقة بسبب عدم الإتساق وعدم الإكتمال.[4]



شكل رقم (1.1.2) خصائص البيانات الضخمة الأربعة 4V[4]

2.9.1.2 مصادر البيانات الضخمة

- شبكات أجهزة الإستشعارو أجهزة التتبع.
- وسائل التواصل الاجتماعي.
- قواعد بيانات أنظمة تخطيط موارد المؤوسسه.
- مصادر البيانات السلوكية، على سبيل المثال مرات البحث على الإنترنت. [4]

المبحث الثاني

الأنظمة مفتوحة المصدر

Open Source System

2.2المقدمة

في السبعينات[5] كانت البرمجيات الإحتكارية لا تسمح للمستخدمين بإعادة توزيعها ، او تعديلها أوالوصول إلى شفرة المصدر لذلك كان لابد من وجود البرمجيات مفتوحة المصدر كردٍّ فعلٍ على حقيقة أن التغييرات أو التحسينات لا يمكن أن تتم في البرمجيات الاحتكارية من قبل المطورين أو المستخدمين.

1.2.2 شفرة أو كود المصدر sourcecode

هو اللغة المتخصصة التي تسمح لمطوري البرمجيات بإنشاء وتعديل برامج الحاسوب. ويحتاج الشخص للحصول على حق الوصول القانوني إلى شفرة المصدر التي تعبر عن الأوامر والتعليمات المكتوبة بلغة من لغات البرمجة. [5]

2.2.2 الأنظمة مغلقة المصدرclosed source software

هي التي لا تتيح شفرة المصدر الخاصة بها ، ولا تسمح إلا للمطورين الرسميين بالتعديل عليها[5].

3.2.2 الأنظمة مفتوحة المصدرopen source software

هي البرامج التي تتيح شفرة المصدرsourcecode الخاصة بها للمستخدمين، ألا و هي الأوامر البرمجية التي تم كتابة البرامج بها، وعندما يتم إتاحة هذه الشفرة فإن بإمكان أي شخص التعديل على هذه البرامج كما يريد حسب الترخيص الذي يحدده مطوري تلك البرامج[5].و الجدول رقم (1.2.2) يبين بعض الأمثلة لهذه البرامج.

1.3.2.2 معايير الأنظمة مفتوحة المصدر

- الوصول إلى شفرة المصدر.
- تعديل شفرة المصدر.
- توزيع النسخة المعدلة من البرنامج.
- إعادة توزيع البرنامج دون قيود. [5]

2.3.2.2 مزايا الأنظمة مفتوحة المصدر

1-التكاليف المنخفضة:

عادةً لا تتطلب رسوم ترخيص، وتكلفتها منخفضة عمومًا ، وذلك هو أحد الأسباب الرئيسية لاختيار الشركات الصغيرة الاعتماد على تلك الفئة من البرامج.

2-المرونة:

يمكنللمبرمج أن يأخذ حزمة البرامج القياسية وتعديلها لتتناسب بشكلٍ أفضل مع المهام المطلوبة. ويمكن للشركة عادةً استئجار مبرمج لإضافة وظيفة معينة إلى برنامج مفتوح المصدر.

3-الأمان والخصوصية:

لن يكون بإمكان المطورين وضع أي برامج ضارة أو تجسسية لأنبإمكان أي شخص الاطلاع على شفرة المصدر واكتشاف تلك البرامج. وعلى العكس، فإن مطوري البرامج مغلقة المصدر باستطاعتهم إضافة أي شيء يريدونه دون الخوف من معرفة المستخدمين لهذه الإضافات.

4-توافر الدعم الخارجي:

يتوفر الدعم الفني الخارجي للعديد من المصادر المفتوحة.[5]

جدول رقم(1.2.2) أمثلة على برامج مفتوحة المصدر

النوع	الأمثلة	
أنظمة تشغيل	Android	Linux
متصفحات انترنت	Chromium	Mozilla Firefox
تعديل الصور	MyPaint	Gimp
برامج مكتبية	Apache OpenOffice	LibreOffice
ضغط وفك ضغط	Universal Extractor	7-Zip
عرض الفيديو	Media Player Classic	VLC Media Player
تصوير الشاشة	Shutter	Greenshot
تحميل التورنت	Deluge	Transmission
تحرير النصوص	Vim	Notepad++
إضافات للمتصفحات	Greasemonkey	Adblock Plus

المبحث الثالث

قواعد البيانات
غير العلائقية NOSQL
و بيئة عمل Hadoop

3.2 المقدمة

فيهذا المبحث سيتم التعرف على قواعد البيانات الغيرعلائقية التي طورت لتسد الفراغ الذي أوجدته قواعد البيانات العلائقية في عدم استوعابها للبيانات الضخمة بالاضافة للتعرف على بيئة عمل نظام Hadoop.

1.3.2 قواعد البيانات الغيرعلائقية NO SQL Database

نظرًا لعدم مقدرة RDBMS في تلبية متطلبات الأداء والقابلية والمرونة التي تتطلبها تطبيقات الجيل القادم التي تتطلب بيانات مكثفة ، تم تبني قواعد بيانات NoSQL لسد هذه الثغرات ، ففي الأصل تم إنشاء تقنية NoSQL واستخدامها من قبل قادة الإنترنت مثل Facebook و Google و Amazon وغيرها ممن إحتاجوا إلى أنظمة إدارة قواعد البيانات التي يمكنها كتابة وقراءة البيانات في أي مكان في العالم ، مع توسيع نطاق الأداء وتقديمه عبر مجموعات بيانات ضخمة وملايين المستخدمين.

حيث تعد NoSQL مفيدة بشكل واضح في تخزين البيانات غير المنظمة والتي تنمو بسرعة أكبر بكثير من البيانات المنظمة ولا تتوافق مع المخططات العلائقية لـRDBMS . وهي اختصار لـ Not Only Sql[6]

2.3.2 اقسام قواعد البيانات الغير علائقية

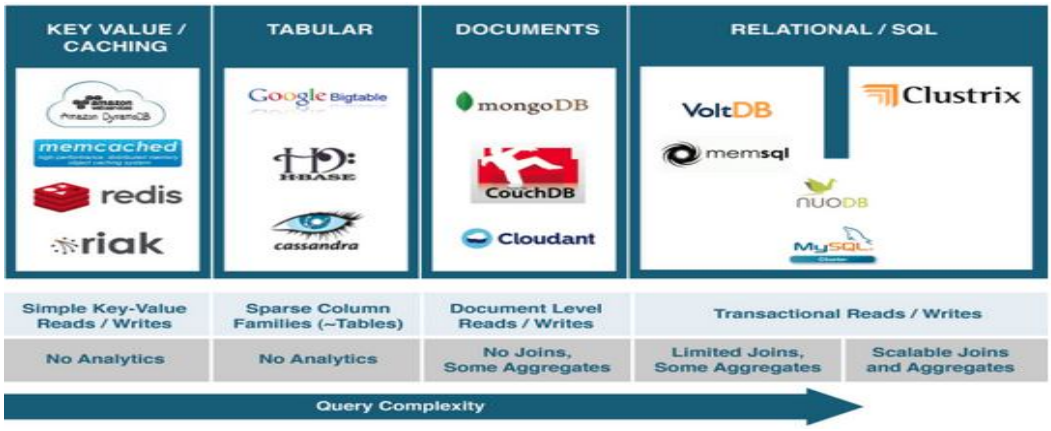
هنالك العديد من الأنواع المختلفة لقواعد بيانات NoSQL لدعم احتياجات محددة وحالات الاستخدام .تقع هذه الانواع في أربع فئات رئيسية هي:

1-Key-value data stores : هيأنظمةفعالةللغايةيتمفيهاالوصوللقيمvaluesعنطريقمفاتيحkeys و من امثلتها Cassandra .

2-Document stores : يتم فيها تخزين البيانات في شكل document بصيغ بيانات مثل josn او xml و من من امثلتها MongoDB .

3-Wide-column store :تقوم بتخزين البيانات في جداول ذات صفوف وأعمدة ، مشابهة لـRDBMS، ولكن يمكن أن تختلف أسماء وتنسيقات الأعمدة من صف إلى آخر عبر الجدول. ومن امثلتها HBase .

4-Graph stores : تمثل فيها البيانات في شكل Graph هي توفر اتصالاً بدون مؤشر ، بحيث يتم ربط العناصر المجاورة مادةً بدون استخدام index مثل Neo4j .[6]



شكل رقم (1.3.2) أمثلة لقواعد البيانات غير العلائقية

3.3.2 الإتساق و نظرية CAP

إن المهام الرئيسية لجميع قواعد البيانات هي تخزين و إسترجاع البيانات صحيحة و بصورة فعالة . القدرة على القيام بذلك مرتبطة بإتساق البيانات consistency و الإتاحية availability و قابلية التوسعscalability .

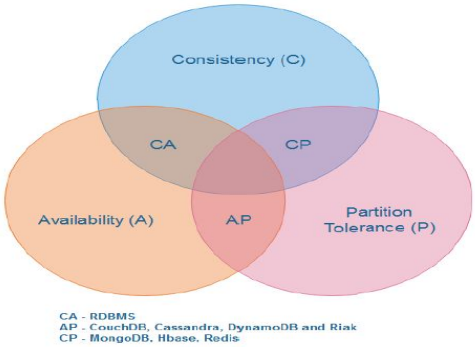
فالاتساق يعني توافق جميع نسخ البيانات في انظمة البيانات المتوازية الناتجة عن عملية الاستنساخ.

وهناك نماذج للإتساق في الأنظمة ومن اشهر هذه النماذج نموذجي ال ACIDو ال BASE ، حيث يمثل الـ ACID النموذج الأكثر قوة والذي يرتكز على اربعة خصائص اساسية : Atomicity ، Consistency ، Isolation ، Durability وهو يضمن ان تكون البيانات متسقة في كل حين على عكسنموذج الBASE والذي يعني الإتساق ولو بعد حين، والذي ظهر مع قواعد البيانات الغيرعلائقية NOSQL ، ففي سياق قواعد البيانات NOSQL الموزعة، الاتساق له معنى خاص . حيث أن قواعد البيانات RDBMS تقدم اتساق عالي strong consistency ، هذا الاتساق العالي يتساوى مع الاتساق المتوافق مع ACID و هو يضمن ان تكون البيانات متسقة في جميع الاوقات ، أما أنظمة NOSQL هي Eventually Consistent أي أن عملية القراءة قد لا تعكس أخر تحديث للبيانات و لكن سينعكس التغير في وقت ما في المستقبل . Eventually Consistent عبارة عن ضعف لأن بعض الأنظمة تحتاج للإتساق في جميع الأوقات مثل Banking System.

نظرية CAPهي نظرية خاصة بقواعد البيانات الموزعة و تقوم علي ان يتم تحقيق خاصيتين علي الاكثرمن اصل ثلاثة ألا و هي :

1- **الاتساق consistency:** اي توافق كل النسخ الناتجة عن عملية الاستنساخ في جميع الاجهزة.

2- **الاتاحية availability** : تعني عندما يتم طلب عملية قراءة او كتابة اما ان تتم بنجاح او اعطاء رسالة توضح عدم اكمال العملية .
3- **امكانية التقسيمpartition tolerance** : تمكين النظام من المواصلة في عمله حتي عند حدوث اي عطل في احد الاجهزة .



شكل رقم(2.3.2) نظرية CAP

كل ما سبق هي خصائص مرغوبة ، ولكن نصت النظرية على أنه لا يمكن تواجد الإتساق و الإتاحية في وقت واحد في وجود partition-prone network لأنه لا يمكن افتراض أن الشبكة موثوق بها و لا يمكن فشلها أبداً . هذا يعني وجوب الاختيار بين لإتساق و الإتاحية . أي أن الإحتمالات المتبقية هي (AP) و (CP) و بالتالي يجب أن تتم التضحية اعتماداً على إحتياجات التطبيق .[7]

جدول رقم (1.3.2) مقارنة بين قواعد البيانات العلائقية و غير العلائقية [8]

	قواعد البيانات العلائقيةRDBMS	**قواعد البيانات غير العلائقية NoSql**
Schema	تتطلب وجود schema معرفة مسبقاً	لا تتطلب وجود schema معرفة مسبقاً
لغة الاستعلامات sql	تعتمد على استعلامات sql	لا تعتمد عليها (not only sql)
نماذج البيانات	تعتمد على ACID	تعتمد على Base
التوسع Scalability	توسع رأسي Vertical	توسع عمودي Horizontal
الاداء Performance	يقل الاداء مع البيانات الضخمة بالاضافة الى بطئ الاستعلامات quarries عند استخدام العمليات المعقدة	تعمل بكفاءة أكبر في وجود البيانات الضخمة و تمثل الخيار الأفضل لتخزين البيانات مما يؤدي إلي اداء أفضل

4.3.2 قواعد البيانات العلائقية الجديدة NewSQL:-

تعتبر تقنية الـ NewSql تقنية جديدة تهدف الى جعل قواعد البيانات العلائقية قابلة للتوسع مع التمسك بخاصية الاتساق القوية ACID وقد ظهرت عدة مشاريع بحثية و منتجات تجارية تستخدم هذا المصطلح استخداماً غير صحيحاً ولذلك قام الباحثون بعرض تصنيفات لـ NewSql و توضيح أنأنظمة إدارة قواعد البيانات هي فعلا ينطبق عليها معايير الـ NewSql و قد قامات بوضع ثلاثة تصنيفات :

1- **New architecture :** وهي عبارة عن انظمة جديدة تم بناءها بمعمارية جديدة لتقوم بإدارة عدة nodes بإستخدام الـconcurency control protocls.
2- **Transparent Sharding Middleware:**و هي عبارة عن تقنية تسمح للمؤسسة بأن تقوم بتقسيم قواعد بياناتها إلى مجموعة من shards والتي تخزن في cluster التي تشترك في نفس النسخة من الـ RDBMS.
3- **Database-as-a-Service:**وهي عبارة عن قواعد بيانات مبنية على اساس تقنية NewSqlومتوفرة كخدمة تقوم بتقديمها الشركات الكبيرة كـ[9] amazon, Microsoft

5.3.2 بيئة نظام Hadoop :-

بيئة نظام Hadoop أو Hadoop Ecosystem هي عبارة عن منصة او إطار عمل Framework يسمح بالمعالجة المتوازية (Parallel Processing) للبيانات الضخمة عبر مجموعات من الاجهزة Cluster باستخدام نماذج بسيطة . و هي مصممة بغرض التوسع باستخدام عدد من الخوادم بدلاً عن خادم واحد , كل منها يقوم بالحوسبة المحلية local computation و التخزين المحلي local storage فبدلاً من الاعتماد على الأجهزة Hardware لتوفير اتاحية عالية High Availability ، فقد تم تصميم هذه البيئة لأكتشاف و معالجة الأعطال في طبقة التطبيقات application layer و بالتالي تقديم خدمة عالية من التوفر high available service على cluster مكون من مجموعة من الاجهزة كل منها معرض للعطل.

و جاءت بيئة Hadoop لمعالجة مشاكل البيانات الضخمة التي فشلت التقنيات القديمة عن حلها ، حيث يمكن اعتبارها جناحاً يشمل عدد من الخدمات مثل تخزين و تحليل و معالجة البيانات في داخله .

و أصبحت بيئة Hadoop مستخدمة من قبل العديد من الشركات العالمية المعروفة مثل Oracle و Google و twitter[10]

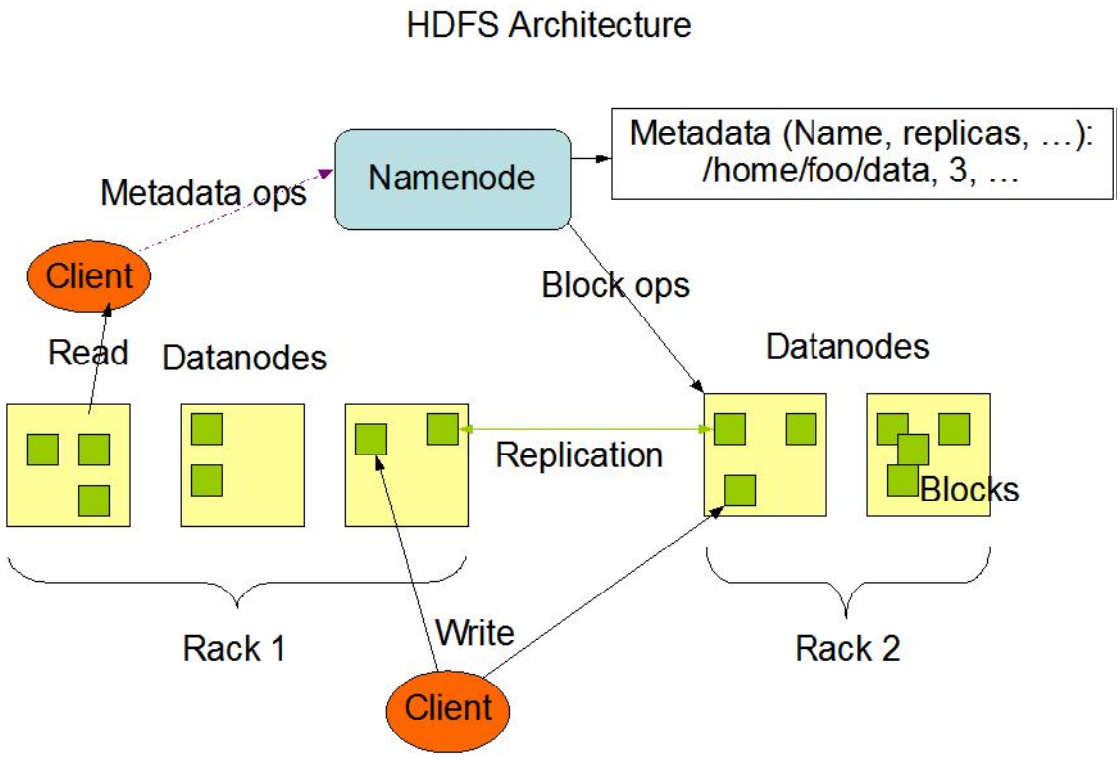
6.3.2 مكونات نظام Hadoop :-

يتكون النظام من مكونين اساسيين هما :

1- نظام الملفات الموزع HDFS (Hadoop Distributed File System) :

هو نظام التخزين الأساسي المستخدم من قبل تطبيقات Hadoop و الذي يوفر أداء عالي للوصول للبيانات بالاضافة لقابلية التوسع و توفير إتاحية عالية .

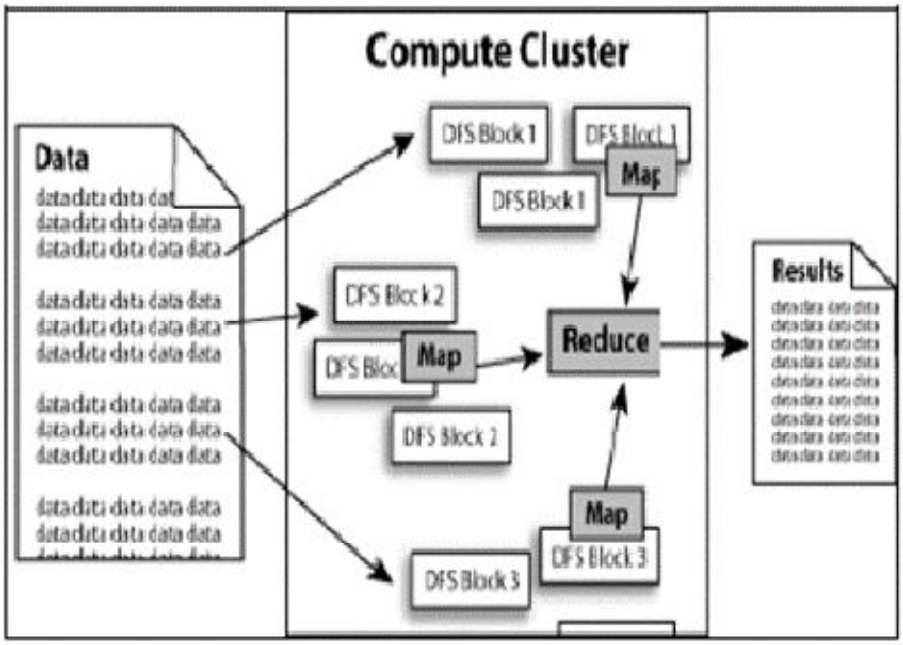
يعتمد HDFS على معمارية Master/Slave حيث يتكون من عدد من العقد Nodes أحداهما تسمى Master أو Name Node و هي العقدة الرئيسية و تحتوي على وصف البيانات و توزيعها في الخوادم ، أما بقية العقد فهي تدعى Slave او Data Node و هي عقد تابعة للعقدة الرئيسية حيث تحتوي على البيانات الحقيقية الموزعة عليها.[10]



شكل رقم(3.3.2) معمارية HDFS

2- نموذج البرمجة Map Reduce :

هو نموذج قدمته Google في 2004 حيث يعمل على معالجة البيانات الضخمة الموجودة في HDFS عن طريق تقسيمها الي اجزاء أصغر و من ثم معالجتها باستخدام المعالجة المتوازية ، حيث يتكون من مرحلتين احداهما Map Phase و هنا يتم أخذ البيانات الخام و تنظيمها في شكل (key/values) و المرحلة الثانية هي Reduce Phase و هي التي تعالج البيانات بالتوازي. [10]



شكل رقم(4.3.2) معمارية Map Reduce

المبحث الرابع

الدراسات السابقة

4.2المقدمة

في هذا المبحث سيتم عرض الدراسات السابقة ذات الصلة بالبحث .

1.4.2الدراسة الأولى: Towards Comprehensive Measurement of Consistency Guarantees for Cloud-Hosted Data Storage Services[11]

قامت الدراسة ببناء مؤشر اساسي شامل لمدى ضمان الاتساق في انظمة التخزين السحائبية ، فحددت مقاييس دقيقة و ذات معنى، كما قامت بتوضيح التحديات والمتطلبات لهذا النوع من المؤشرات واقترحت معمارية لإجراء انظمة قياس مشابهة وبعد ذلك وضحت كيفية بناء ادوات قياس شاملة عن طريق إعادة استخدام اجزاء معيارية و مؤكدة ،و بعد ذلك قامت باستخدام هذه الادوات لتقييم مدى تاثير النسخ المختلفة وأعباء عمل workloads مختلفة على اثنان من قواعد البيانات Nosql هما Mongo DB و Cassandra و ايضاً قامت بدراسة كيفية الاختلاف عند استخدام اعدادات مختلفة ل Cassandra .

2.4.2 الدراسة الثانية: Consistency issues on NOSQL databases: problems and possible solutions [12]

يقوم البحث بعرض مجموعة من الاسباب التي قد تسبب عدم اتساق البيانات في قواعد البيانات الموزعة ، كما يتطرق الي جودة الطرق التي يتم اتباعها للحفاظ علي الاتساق QUORUM،BASE و كيفية تحسين الطرق الحالية .

توصل البحث الي ان اسباب ضعف الاتساق هي :

- Data Redundancy
- Network Latency

في التجربة تم استخدام اثنين من قواعد البيانات اللاعلائقية (NOSQL) هما CASSANDRA و RAIK . تم الاختبار علي مجموعة من دفعات اليبانات ، و اخذت المقاييس الـ latency و throughput عن طريق الاداة YCBS benchmarking tool.

تمت المقارنة و عرضت النتائج في اشكال جداول و رسومات بيانية و اظهرت نتائج هذة الدراسة ان Cassandra تقدم اتساق بيانات افضل من . Riak

Workload	Operation	Cassandra	Riak
A	Read	4	3
	Update	4	2
B	Read	4	3
	Update	4	2
D	Insert	1	4
	Read	4	4

شكل رقم (1.4.2) يوضح الفرق في الأداء بين Cassandra و Riak .

3.4.2 الدراسة الثالثة: An Experimental PerformanceComparison of NoSQL and RDBMS Data Storage Systems in the ERP System Odoo [13]

هذا البحث قام نتيجة لتقرير شركة Inotos Gesellschaft mit beschrankterr Haftung الذي أشار الي وجود ضعف في احدى انظمة Odoo التي قد طورت الي أحد العملاء و بعد التحليل كشف ان المشكلة ناتجة عن وحدتي Mail messages Module و Attachments Moduleاللتي يستخدما بواسطة الوحدات الاخرى في النظام ، حيث أن حجم البيانات كان يتضخم و يزداد بصورة أسرع من الوحدات الأخرى مما أدى الي ضعف أداء النظام .

تم حل مشكلة ضعف الأداء التي سببتها كل من الوحدتين بنقل تخزين الوحدتين ال نظام تخزين أخر و هو قواعد البيانات الغير علائقية NoSql (تم اختيار HBase حسب المتطلبات)و تطوير وحدة تمكن ORM للتواصل مع NOSql .و قد تم اختيار بيئة Hadoop كبيئة تخزين لهذه الوحدتين ، أما بقية الوحدات ظلت كما هي في قاعدة البيانات الخاصة بنظام Odoo و هي PostgreSql.

اجريت مقارنة في الاداء بين النظام الحالي و النظام المقترح فخرج البحث منها بأن النظام المقترح يكون حلاً أفضلاً بالنسبة لوحدتي Mail messages Module و Attachments Module عندما توصفا بالتضخم وان النظام الحالي أفضل اذا كان حجم البيانات قليلاً .كذلك أوصى البحث الي استخدام بيئة تخزين كنظام ثانوي و تخزين البيانات الحديثة و بالتالي يمكن استخدام هذه المنظومة في توليد التقارير التي يمكن ان تؤثر في اداء ال PostgreSql. و اينما ضعف أداء ال PostgreSql فإن العميل يستطيع التحويل و استخدام بيئة Hadoop بسلاسه.

4.4.2الدراسة الرابعة: Performance Evaluation of Unstructured NoSQL data over distributed framework [14]

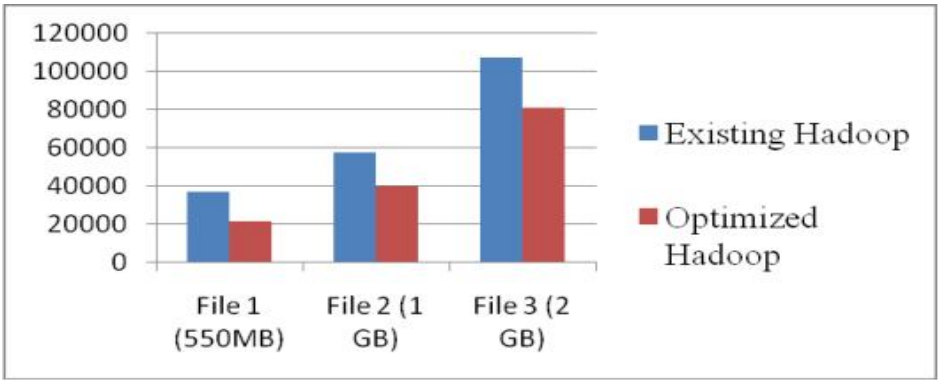
قامت هذه الورقة العلمية بإعطاء نظرة عامة على بيئات العمل وقواعد البيانات الموزعة ومميزات كل منهما ، وقيمت زمن عملية البحث search على مجموعات العمل لقاعدة البيانات Mongo DB (cluster set) و وضحت أن الترتيبات configuration تعطي نتائج مختلفة في اداء عملية البحث ، كما بينت التحليلات ان الاداء يقل بصورة سريعة مع زيادة عبارات النداء call of querries بعد اي بداية و أن عمليتا الترتيبات configuration و التجزئة في قاعدة بيانات Mongo مهمة لزيادة الإنتاجية .
كما تضمن البحث مقارنة بين قواعد البيانات العلائقية و قاعدة بيانات MongoDB وبينت ان MongoDB تتفوق من ناحية سرعة الادخال والبحث .

5.4.2 الدراسة الخامسة : File replication and consistency maintenance in the HADOOP cluster using IRM technique[15]

ينظر البحث الي المشاكل التي تواجه البيانات الضخمة الموزعة خلال عملية ال REPLICATION و مشكلة اتساق البيانات و يقدم تقنية التي قد تعتبر حلاً جزئيا للمشكلة .

تقوم التقنية المسماة INTEGERATED DATA REPLICATION & CONSISTENCY MAINTENANCE (IRM) علي زيادة النسخ الموجودة من البيانات علي حسب معدل الوصول اليها ACCESS RATE(POPULARITY) بينما تبقي البيانات ذات الوصول القليل بدون نسخ و تزيد بالتدريج مع زيادة الطلب علي البيانات .

اظهرت النتائج ان تقنية ال IRM تقلل من زمن الوصول الي البيانات و تحسن من الاتساق كما انها لا تزال تحافظ علي الاتاحية و الاعتمادية و لكنها تقلل من اتاحية بعض البيانات بصورة جزئية .

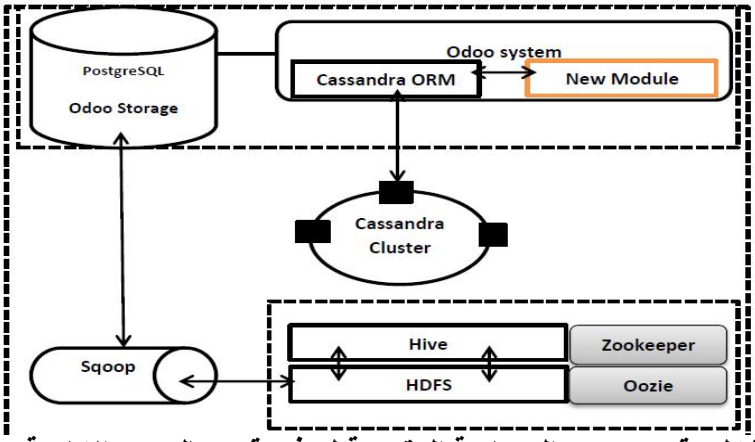


شكل رقم (2.4.2) يوضح تحسن زمن الوصول عند استخدام تقنية IRM .

6.4.2الدراسةالسادسة: Storing and Processing Big Data for Enterprise Resource Planning Systems[16]

أجري هذا البحث لحل بعض المشاكل و الصعوبات التي تتمثل في أنظمة موارد المؤسسة التي تعتمد على قواعد البيانات العلائقية حيث سعى الي بناء مستودعات بيانات التي تحتاج في الحقيقة الي وقت و جهد و تكلفه ، بالأضافة الي النظر في صعوبة معالجة و إدارة الحجم الضخم من البيانات بالطرق التقليدية و بالتالي تأخير تنفيذ العمليات المعقدة .

اعتمد البحث علي استخدام نظام Odoo كدراسة حالة ، حيث تم استراد البيانات من قواعد البانات الخاصة به PostgreSql واجراء عمليات التخزين لأغراض الأرشفه في نظام الملفات الموزع HDFS و معالجة البيانات عن طريق نموذج البرمجة MapReduce، كما أتاح التخزين و التعامل المباشر عن طريق وحدة Odoo جديدة تم تطويرها لتواصل مع قاعدة التخزين غير العلائقية Cassandra .



شكل رقم (3.4.2) المعمارية المقترحة لتوفير قدر عالي من الإتاحية.

خرج البحث بأرشفة البيانات في أنظمة التخزين Nosql هو خيار اقتصادي يقلل الجهد و الزمن المبذولين في بناء مستودعات خاصة و أن التخزين المباشر لبيانات وحدات نظام Odoo في أنظمة NOsql يوفر خصائص الاتاحية و التوسع و سرعة تنفيذ العمليات علي البيانات .

7.4.2 الدراسة السابعة: Consistency Tradeoffs in Modern Distributed Database System Design [17]

أشارت هذه الورقة البحثية الي مدى الاحتياج لأنظمة البيانات الموزعة DDBSs في الآونه الاخيرة ، حيث كانت الدوافع من استخدامها أن التطبيقات الحديثة تتطلب زيادة البيانات و العمليات الانتاجية و التي بدورها تحتاج الي قواعد بيانات موسعة باالاضافة الي الزيادة في العولمة GLOBALIZATION و تقدم الاعمال الذي ادى الي ابتغاء توفر البيانات بالقرب من ال clients الموزعون علي نطاق مختلفة .

و اشارت الورقة استناداً علي نظرية CAP ان معظم DDBS لا تدعم خاصية الأتساق و توصلت الي بعض المعتقدات الخاطئة منها :

-افتراض ان DDBS التي تقلل من ال consistency في غياب ال partitionانها تفعل ذلك استناداً على نظرية CAP .

- افتراض ان DDBS التي يجب ان تكون tolerant of network partition يستوجب ذلك علي النظام الاختيار بين availability وconsistency .

و توصلت الي وجود مفاضلة (Tradeoff) بين consistency و latency والسبب فيها انه لتحقيق high availability يجب فرض النظام بعملية استنساخ البيانات و التي يمكن ان تتم ثلاثة طرق لخصت في الجدول التالي :

جدول رقم (1.4.2) طرق عملية الإستنساخ .

(1) Data updates sent to all replicas at the same time	• Updates do not first pass through a preprocessing layer (lack of consistency) • Updates first pass through a preprocessing layer (increase latency)
(2) Data updates sent to an agreed-upon location first	• Synchronous (master node waits until all updates sent to replicas) • Asynchronous (system treats the updates as if it were completed) • Combination of both (system sends updates to some subset of replicas synchronously, and the rest asynchronously)
(3) Data updates sent to an arbitrary location first	• Different from (2) in that the location the system sends updates to is not always the same.

و في صورة اكثر تكاملاً لوصف consistency-Tradeoff في DDBS يمكن اعادة صياغ CAP ب PACELC و التي تعني:

“If there is a partition, how does the system trade off availabilityand consistency; else, when the system is running normally in the absence of partitions, how does the system trade off latencyand consistency”

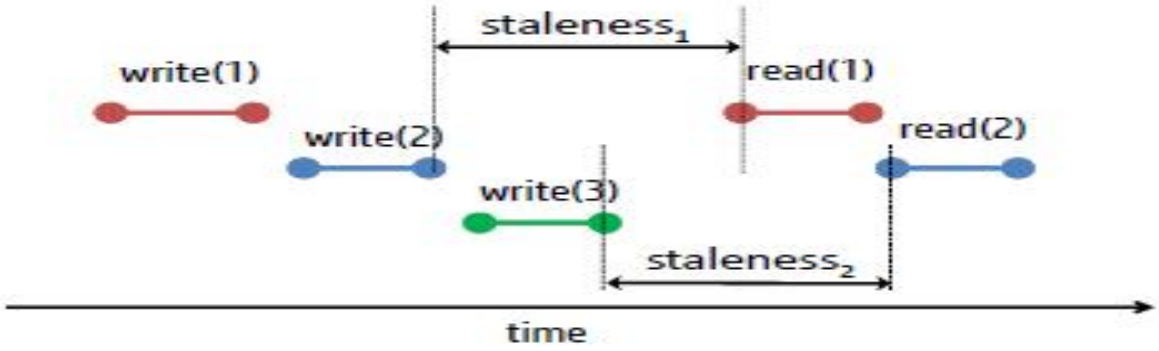
و الجدول التالي يبين بعض انواع قواعد البيانات و الخواص التي توفرها :

جدول رقم (2.4.2) امثلة لقواعد البيانات و الخواص التي توفرها من نظرية PACELC .

Dynamo, Cassandra, and Riak	PA/EL	-If a partition occurs, give up consistency for availability. -Under normal operation, give up consistency for lower latency
VoltDB/H-Store and Megastore, HBase	PC/EC	Refuse to give up consistency, will pay the availability and latency costs to achieve it
MongoDB	PA/EC	Guarantees reads and writes to be consistent.
PNUTS	PC/EL	Gives up consistency for latency. If a partition occurs, it trades availability for consistency

8.4.2 الدراسة الثامنة : Toward a Principled Framework for Benchmarking Consistency[18]

هذه الورقةقدمت منهجية قياس مرجعية لفهم التناسق في أنظمة التخزينذات البيانات الموزعة ، وانها تعتمد على قراءات معيار الـYCSB لقياس الإتساق بطريقة كمية،حيث نقوم بحساب ما يسمى بالstaleness وهو الفرق بين زمن عملية الكتابة و زمن قراءة القيمة الجديدة ، وكلما كانت قيمته قليلة كان الاتساق عالي وفي حالة وجود اكثر من عقدة فاننا نحسب الstaleness لكل عقدة واعتماد الاكبر ، وقد تم تطبيق هذه المنهجية في قياس الإتساق في الـCassandra وقد اعطت قراءات سلبية حيث إن الإتساق يحدث بعد زمن طويل نسبياً .



شكل رقم (4.4.2) مثال لكيفية حساب الـ staleness .

جدول رقم (3.4.2) أوجه التشابه و الإختلاف بين البحث و الدراسات السابقة .

رقم الدراسة	أوجه التشابه	أوجه الإختلاف
(1.4.2)	قياس درجة الإتساق consistency	عدم قياس الـLatency و الـThroughput
(2.4.2)	قياس الـLatency و الـThroughput	عدم قياس درجة الإتساق consistency
(4.4.2)	قياس الـLatency	عدم قياس الـconsistency و الـThroughput
(5.4.2)	قياس الـLatency	عدم قياس الـconsistency و الـThroughput
(8.4.2)	قياس درجة الإتساق consistency	عدم قياس الـLatency و الـThroughput

خلاصة الباب:

هذا الباب تناول الخلفية النظرية و المفاهيم العامة المتعلقة بالبحث والدراسات السابقة ذات الصلة .

الباب الثالث

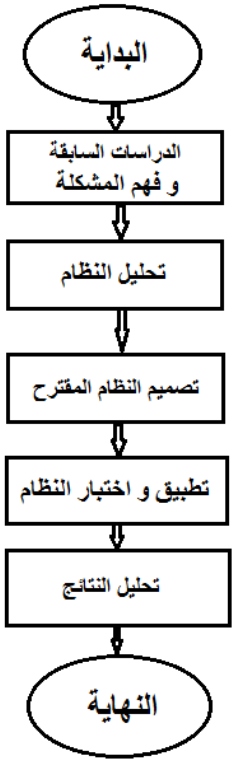
منهجية البحث

1.3 المقدمة :

في هذا الباب سيتم عرض المنهجية التي تم اتباعها في البحث ، بالأضافة لتناول أهم الأدوات مفتوحة المصدر التي استخدمت في هذا البحث .

2.3 المنهجية :

بدأت المنهجية بالتعرض و البحث في الدراسات السابقة للتعرف و فهم المشكلة ، ومن بعدها تصميم النظام المقترح بربط المكونات components ببعضها البعض و من ثم اختبار النظام و تحليل النتائج التي سيتم الوصول إليها .



شكل رقم(1.3) المنهجية المتبعة في بناء النظام المقترح.

وقد تم إتباع المنهجية الموضحة بالشكل اعلاه ، ففي البدء تم البحث في الدراسات السابقة ودراسة المشاكل التي تواجه نظم إدراة البيانات الحالية وجوانب القصور في كل منها , وكما تم البحث عن متطلبات الشركات التي تتعامل مع البيانات الضخمة والتي ذكرت في الباب التالي"تحليل وتصميم النظام"، وبعد ذلك تمت عملية تحليل النظام التي كان ناتجها إختيار الادوات المناسبة بناءً على فهم المشكلة والمتطلبات وصممت معمارية النظام المقترح لتوضيح كيفية ربط هذه الأدوات بعد ذلك تمت تهيئة البيئة ثم تطبيق النظام واختباره وتحليل وعرض النتائج.

3.3 المصادر المفتوحة المستخدمة في البحث:

هنا سنتعرض الأدوات مفتوحة المصدر التي تم استخدامها في تصميم النظام المقترح مع ذكر وظيفة و مميزات كل منها .

1.3.3 قاعدة البيانات MongoDB :

هي نوع من أنواع قواعد البيانات الغير علائقية مفتوحة المصدر التي نشأت في منتصف 2000 و التي تستخدم نموذج document oriented الذي يدعم مختلف أشكال البيانات . و هي واحدة من العديد تحت لافتة NOSQL لاستخدامها في تطبيقات البيانات الضخمة و غيرها من مهام المعالجة التي تتضمن البيانات التي لا تتلائم بشكل جيد في النموذج العلائقي. [7]

و بدلاً من استخدام Tables و Rows كما في قواعد البيانات العلائقية فإن معمارية MongoDB مكونة من Collections و documents . [7]

شكل رقم(2.3) هيكل بيانات Mongo DB

-**Database :** عبارة عن سياق Mongo DB و هي الحاوية الخارجية و تتكون من مجموعة من الـCollections .

-**Collections:** مجموعة من مستندات قاعدة البيانات Mongo Documents .

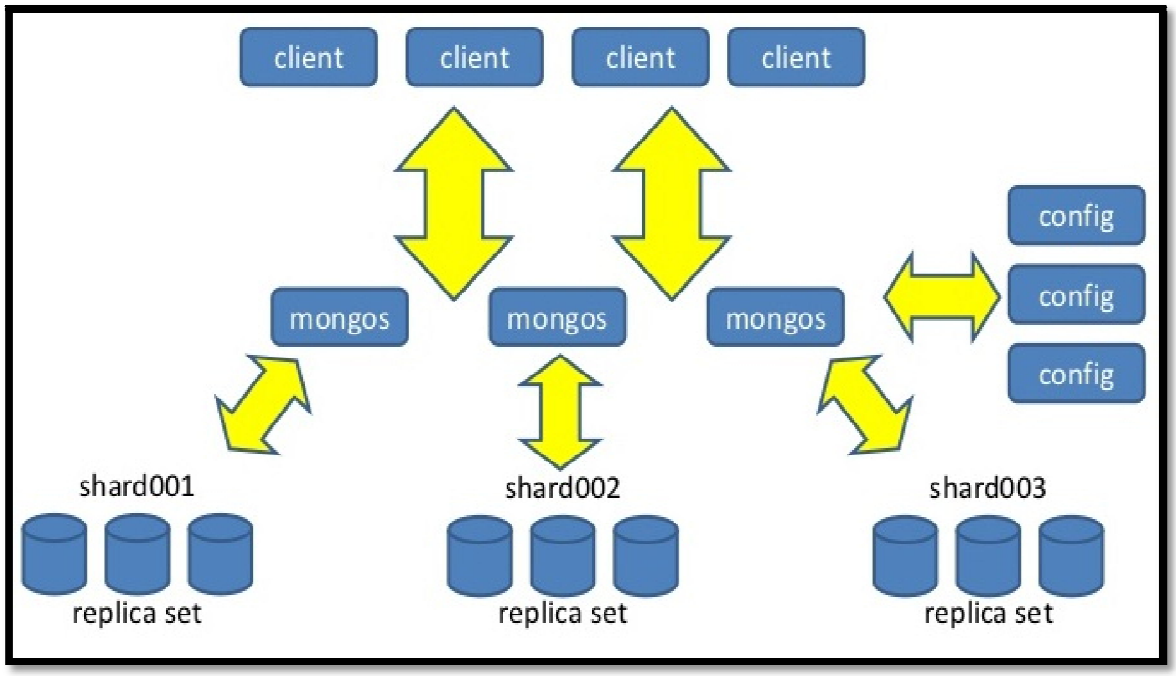
-**Documents :** البنية الأساسية لتخزين وحدة واحدة من البيانات . حيث أن Mongo Document تستخدم صيغة BSON و هي عبارة عن مجموعة من (fields: value-pairs) . بالاضافة الي أن أي document يجب أن يحتوي علي unique identifier . [7]

```
{
   _id : <ObjectIdX>,                          <---------- field : value
   title : "MongoDB document model",           <---------- field : value
   size :  "3.4MB",                                          :
   duation : "00:03:12",                                     :
   details :
       {
          text : "Some descriptive text",
          contact : "contact@mail.com"
       },
    related : [<ObjectIdY>,<ObjectIdZ>],
    tags : ["mongodb","document","tutorial"]
}
```

شكل رقم(3.3) مثال لـ MongoDB document .

معمارية :MongoDB

قاعدة البيانات MongoDB تدعم خاصية التوسع scalability عن طريق تنفيذ عمليتي التوزيع sharding و الاستنساخ replication .



شكل رقم(4.3) عمليتي التوزيع و الاستنساخ في MongoDB .

مميزات قاعدة البيانات :Mongo DB

1. تسمح بالوصول السريع للبيانات بسبب طبيعتها في إستخدام الذاكرة الداخلية للتخزين.
2. تدعم ACID و لها إتساق عالي.
3. هيكلية الكائن (object)الواحد واضحة جداً .
4. تسمح بوجود إختلاف بين عدد الحقول و المحتوى و الحجم من document لأخر.
5. لا تحتاج لتشغيل Virtual Machine .
6. بما انها عبارة عن Nosql ف بالتأكيد هي آمنه لعدم إمكانية حدوث sql injection.
7. لا تحتاج لعمليات JOINS معقدة نسبة لهيكلة البيانات (fields: value-pairs).
8. سهولة التوسع .
9. توفر كمية من documentation مما يشجع الباحثين على استخدامها.
10. تتبع MongoDB دورة منتظمة في انشاء اصداراتها الجديدة . [7]

تحليل الأداء لقواعد البيانات العلائقيةRDBMS و MongoDB :

- يتم استخدام الجداول Tables كوحدة تخزين بينما يتم استخدام Collections في قاعدة البيانات MongoDB.
- يوجد العديد من ال schema في RDBMS و في كل منها يتم إنشاء الجداول لتخزين البيانات , بينما في MongoDB و التي تعتبر document oriented فإن البيانات تكتب في صيغة BOSNو التي تشبه كثيراً صيغة josn .
- أنظمة Mongo DB أسرع 100مرة تقريباً مقارنة بقواعد البيانات التقليدية. [7]

2.3.3أداة القياس YCSB

هي عبارة عن اختصار لـ **(Yahoo Cloud Serving Benchmark)** وتستخدم لتقييم أداء و قياس قواعد البيانات , حيث تمتلك نماذج بيانات data models افتراضية بالاضافة الي workloads لتنفيذ الإختبار. [18]

Mongo-hadoop connector 3.3.3

وهو نظام تابع لـHadoop يقوم بربط ال Hadoop مع MongoDB .

Hadoop 4.3.3

سيتم استخدامه في استيراد البيانات من ال MongoDB و اجراء العمليات عليها عن طريق الـ MapReduceثم تخزينها مجدداً .

خلاصة الباب:

في الباب السابق تم عرض المنهجية المتبعة في البحث و المصادر المفتوحة التي تم استخدامها في البحث و مميزات كل منها.

الباب الرابع

تحليل وتصميم النظام

1.4 متطلبات النظام:

1.1.4 المتطلبات الوظيفية:

1. إمكانية تخزين البيانات الضخمة بكفاءة عالية .
2. إمكانية معالجة وتحليل البيانات الضخمة بكفاءة عالية .
3. إمكانية إسترجاع البيانات الضخمة بكفاءة عالية .
4. التحديث للتغيرات السريعة التي تحدث اثناء تفاعل المستخدم مع النظام , من غير الحوجة لإعادة كتابة المجموعات الداخلية .
5. الوصول للنظام من عدة أجهزة موزعة في نفس الوقت.

2.1.4 المتطلبات غير الوظيفية:

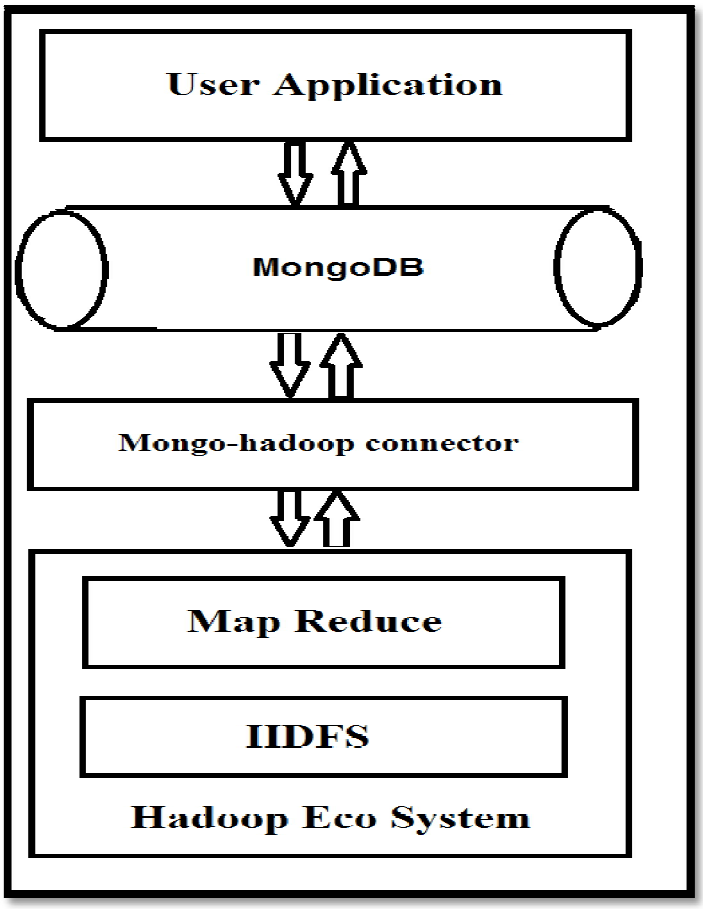
1. توفير درجة عالية من الاتساق (consistency).
2. قابلية التوسع (Scalability).
3. سرعة الأداء في وجود البيانات الضخمة .
4. التوفر والأتاحية (availability).
5. سهولة فهم المستخدم لكيفية تطبيق الحل على نظامه (learnability).
6. سهولة التعامل مع النظام (usability).

2.4 تحليل المتطلبات:

وكما ورد في المتطلبات الوظيفية للنظام فإن هناك حوجة لتخزين ومعالجة وإسترجاع البيانات بكفاءة عالية و إمكانية الوصول للنظام من عدة اجهزة في الوقت نفسه وهذا ما توفره أدوات إدارة البيانات الضخمة وهنا نجد أن Hadoop يوفر بيئة جيدة للقيام بهذه العمليات على النحو المطلوب ولكن نجد أن Hadoop يعتمد نظام الملفات الموزعة HDFS و هو ما لا يحقق متطلب التحديث اللحظي و الدرجة العالية من الإتساق ولذلك سنقوم بابدال نظام التخزين اللحظي هنا باستخدام قاعدة التخزين الغيرعلائقيةMongoDB . ولأن MongoDB هي ليست نظام التخزين الاساسي لـHadoopفإننا سنضطر لإضافة Mongo-hadoop connector وهو نظام تابع لـHadoop .

3.4 معمارية النظام:

وهنا تم توضيح كيفية الربط بين الاجزاء المقترحة للنظام و كيفية ربط النظام ببرنامج المستخدم وكيفية سريان البيانات والعمليات .



شكل رقم(1.4) معمارية النظام المقترح

نجد أن المعمارية مقسمة إلى مستويات ، في المستوى الأول برنامج المستخدم User Application والذي يتعامل مع النظام المقترح عن طريق قاعدة البيانات MongoDB فيخزن فيها البيانات ويسترجعها منها ، مما يضمن الإتساق الدائم للبيانات لأن MongoDB تستخدم نموذجACID الذي تم التحدث عنه سابقا . أما في المستوى التالي يأتي نظام Mongo-hadoop connector والذي يساعد في نقل البيانات المخزنة في MongoDB الى النظام HDFS التابع لـHadoop لمعالجتها بواسطة نظام المعالجة MapReduce وتحليلها ومن ثم إعادة النتائج الى الـMongoDBمجدداً لتقوم MongoDB بتخزينها في المستوى الأخير Hadoop .

خلاصة الباب :

وقد تم في هذا الباب تحليل متطلبات النظام و اقتراح الادوات المناسبة التي يمكن ان تستخدم في الحل، كما تم توضيح كيفية الربط بينها مما سيساعد بشكل كبير في تطبيق النظام وهذا ما سيتم إيراده في الباب القادم.

الباب الخامس

تطبيق و اختبار النظام

1.5 مقدمة:

في هذا الباب سيتم توضيح الطريقة التي نفذبها النظام المقترح اعتماداً على المعمارية المقترحة والأدوات المذكورة في الباب السابق ، وكما سنوضح الخطوات المتبعة لإجراء الإختبار .

سنقوم بعرض الخطوات بالترتيب التالي :

1.تهيئة بيئة عمل الـ Hadoop.

2.تهيئة بيئة عمل الـMongoDB.

3.ربط بيئتي العمل MongoDB وHadoop.

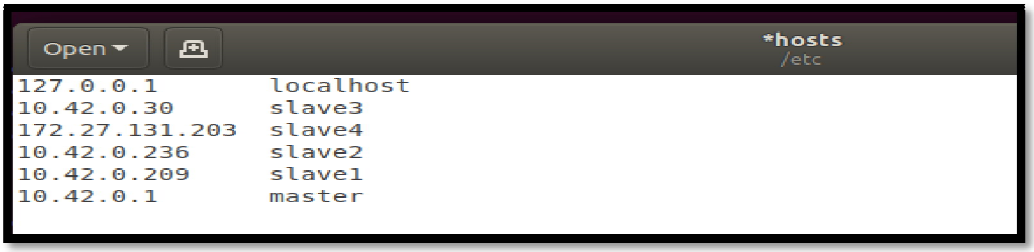
4.إختبار النظام .

2.5 تهيئة بيئة عمل الـ Hadoop:

وتتكون هذه الخطوة من مجموعة من الخطوات:

1.2.5 ربط العقد Nodes :

1. في هذه الخطوة تم تغير التعريفات الموجودة بملف الـHosts كما موضح في الشكل**(1.5)** بإضافة البيانات الخاصة بالعقد المستخدمة.



الشاشة رقم(1.5) ملف Hosts.

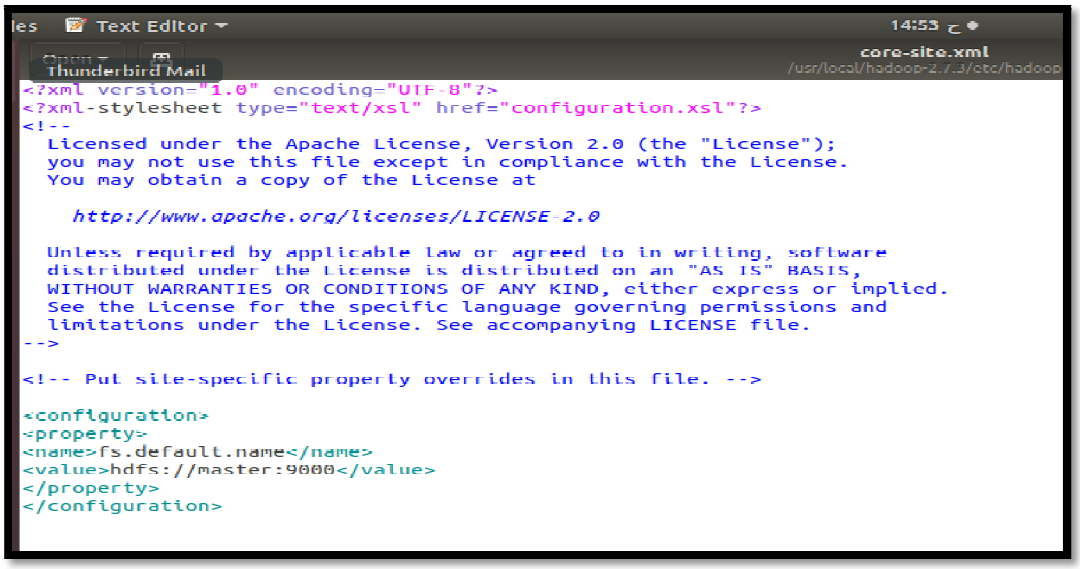
2. تم ربط الأجهزة مع بعضها البعض بإستخدام نظام الربط secure shell بإتباع الخطوات الموضحة في الشكل التالي :

```
$ sudo apt-get install openssh-server
$ ssh-keygen -t rsa
$ ssh-copy-id -i ~/.ssh/id_rsa.pub ubuntu@master
$ chmod 0600 ~/.ssh/authorized_keys
$ sudo reboot
```

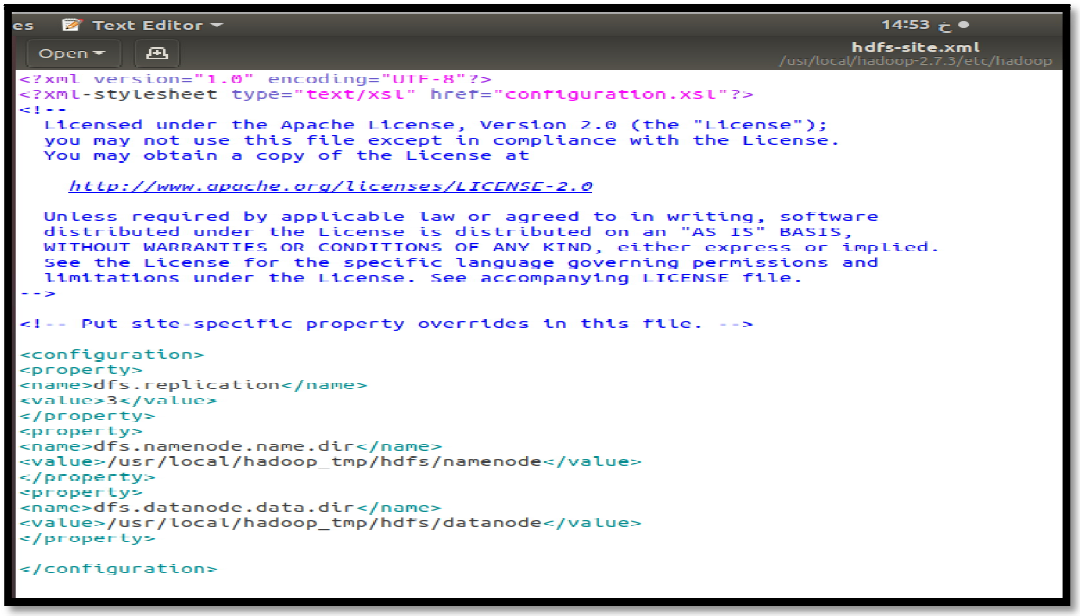
الشاشة رقم(2.5) خطوات Secure Shell.

2.2.5 تهيئة ملفات إعدادات Hadoop:

تم تغير بعض الاعدادت في ملفات إعدادت Hadoop للتوافق مع الـcluster الجديد كما موضح بالأشكال الأتية:



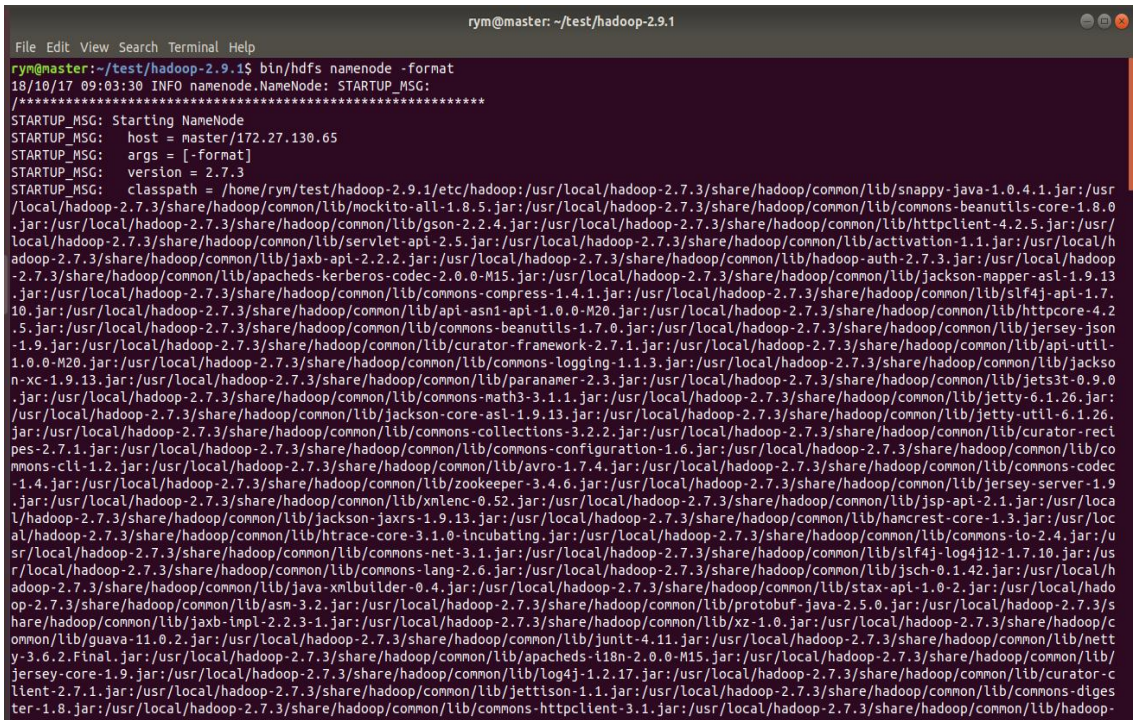
الشاشة رقم(3.5) ملف Core-site .



الشاشة رقم(4.5) ملف hdfs-site.

3.2.5 تهيئة الـHDFS :

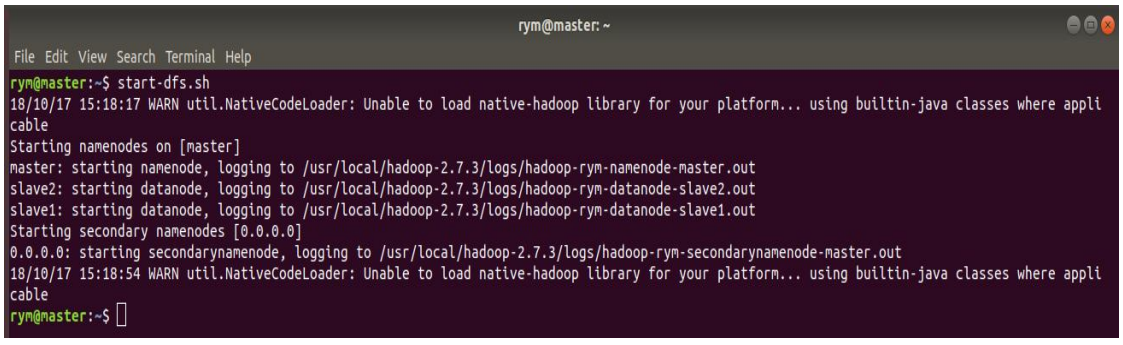
تمت تهئيت الـHDFS بإستخدام الأمر namenode –format كما موضح في الشكل أدناه :



الشاشة رقم(5.5) توضح تهيئة الـ hdfs

4.2.5 تشغيل الـ Hadoop في الـ cluster:

لتشغيل الـ Hadoop في الـ cluster تمت هذه العملية في الـmaster node ،حيث تم تشغيل الـhadoop في الـ master node باستخدام الأمر start –dfs.sh والأمر start –yarn.sh وهو بدوره تلقائيا يقوم بتشغيله في بقية الـ Nodes كما مبين في الشكل أدناه :



الشاشة رقم(6.5) توضح تهيئة الأمر start –dfs.sh

الشاشة رقم(7.5) توضح تهيئة الأمر start –yarn.sh

3.5 إعداد بيئة تشغيل MonogDB

ولأعداد هذه البيئة تم اجراء الخطوات التالية :

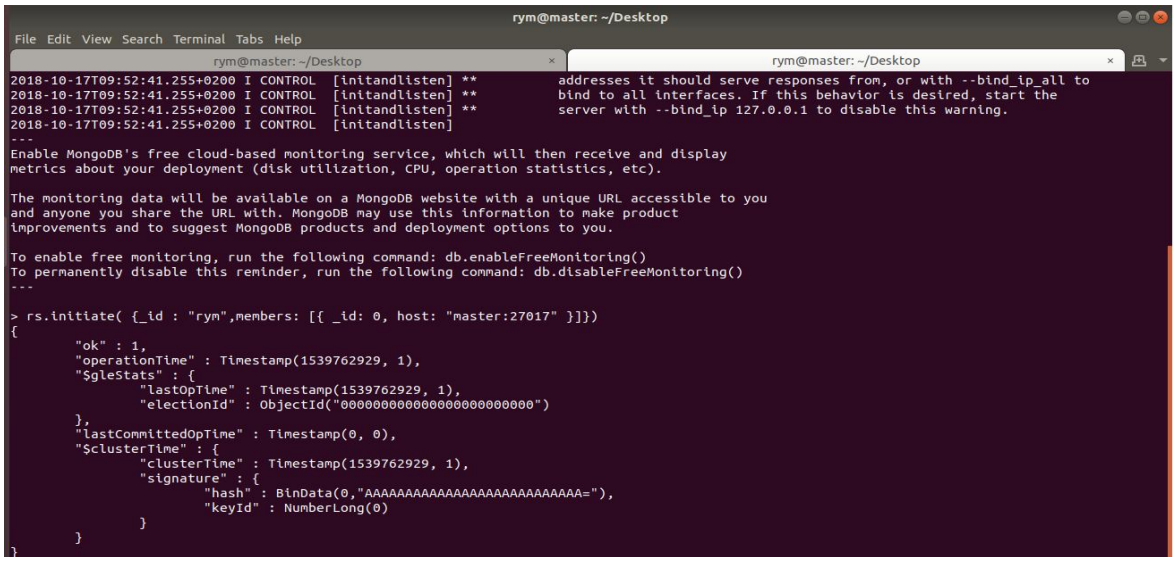
1.3.5 إعداد بيئة MongoDB للعمل في مجموعة عمل Replica Set :

هذه الخطوة تشمل تعديل بعض الإعدادات في الملف الخاص بإعدادات MongoDb والذي يسمى mongo.conf

الشاشة رقم(8.5) ملف الاعداداتmongo.conf

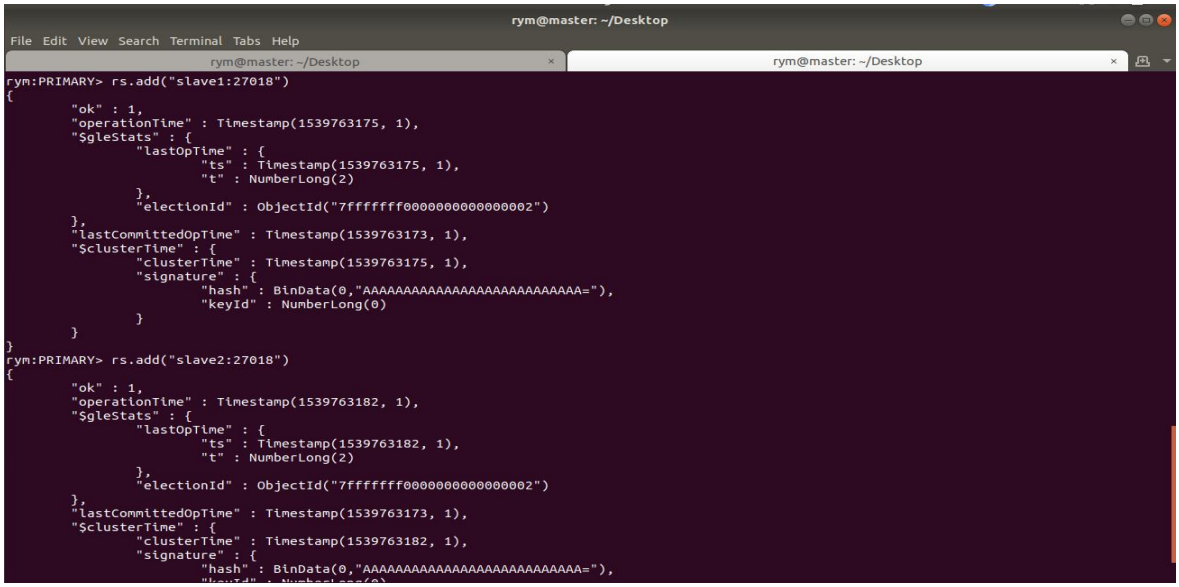
2.3.5 إنشاء الـ Master :

في هذه الخطوة تم القيام بالخطوات الموضحة في الشاشة التالية لتعيين أحد الأجهزة كـ Master للمجموعة MongoDB Replica Set.



الشاشة رقم(9.5) توضح خطوات تحديد الـ Masterفي MongoDB Replica Set

3.3.5 إضافة بقية الـ Nodes إلى الـ Replica Set :



الشاشة رقم(10.5) توضح تحديد بقية الـ Nodes في MongoDB Replica Set

4.5 ربط بيئتي العمل MongoDB و Hadoop :

هناك مجموعة من الخطوات التي تم اتباعها و هي :

1.4.5 إضافة ملفات الربط إلى مكتبات الـ Hadoop و الـ Map Reduce:

في هذه الخطوة تم تحميل مكتبة Hadoop-Mongo Connector وإضافتها الى مكتبات الـHadoop والـMap Reduce .

2.4.5 كتابة وإستيراد البيانات من وإلي الـMongoDB بإستخدام الـ Map Reduce :

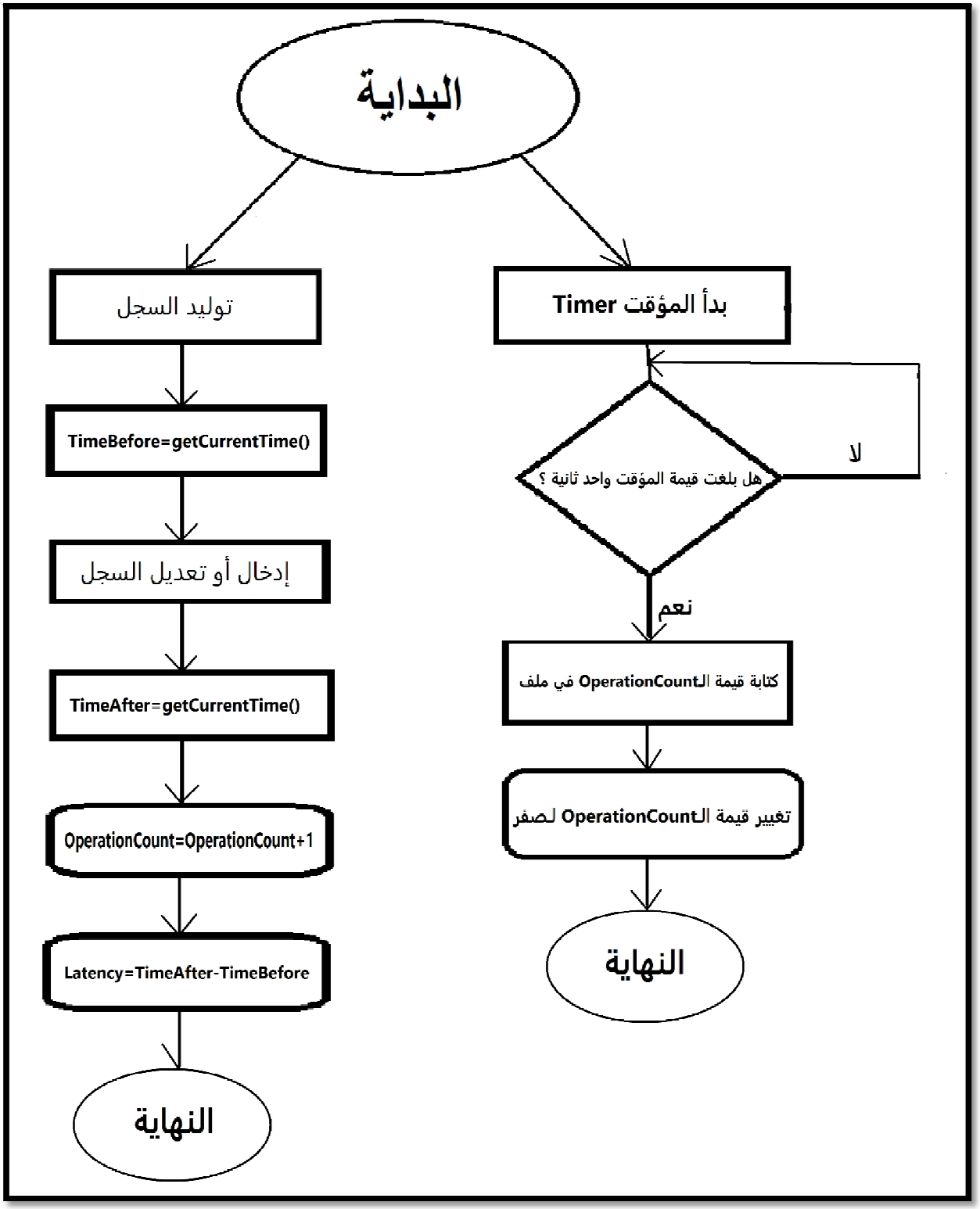
```
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    MongoConfigUtil.setInputURI(job.getConfiguration(), "mongodb://master:27017/mongo_hadoop.collectic
    MongoConfigUtil.setOutputURI(job.getConfiguration(), "mongodb://master:27017/mongo_hadoop.collecti
    job.setJarByClass(Mongo_MR.class);
    job.setNumReduceTasks(0);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);
    job.setInputFormatClass(com.mongodb.hadoop.MongoInputFormat.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    FileOutputFormat.setOutputPath(job, new Path(args[0]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

الشاشة رقم(11.5) توضح كتابة واستيراد البيانات من والي الـMongoDB باستخدام الـ Map Reduce

5.5 اختبار النظام :

في عملية إختبار النظام سيتم استخدام ثلاثة معايير وهي الانتاجية Throughput(عدد عمليات الكتابة أو التحديث في الثانية الواحدة) ،والـlatency(مقدار التاخير لكل عملية)، والـstaleness(الزمن المستغرق لتحقيق الاتساق)، لقياس الأداء و الاتساق.

تمت الاستعانة بأداة القياس الـYSCB نظراً لوجود بعض القصور في هذه الأداة من ناحية قياس الإتساق ، بالإضافة إلى عند توليد البيانات في كل مرة تحدث مشاكل توقف التنفيذ وعند إعادة توليد البيانات فإن الأداة تقوم بإعادة توليد البيانات التي ولدتها سابقاً مما يسبب مشاكل في قاعدة البيانات لأنها لا تقبل التكرار ، ولذلك كان لابد من مسح البيانات المولدة سابقاً وإعادة عملية التوليد من الصفر ، ومن الواضح أن هذه الطريقة غير عملية خصوصاً في محاولة توليد بيانات كبيرة بما يكفي لإجراء الإختبار ولذلك كان لابد من التعديل على الأداة YCSB ولأنها مفتوحة المصدر فقد تم أخذ المصدر وتعديله وتطوير أداة تعالج هذا القصور . كما تقوم بقياس الإتساق إعتمادا على نظرية الـStaleness وسميت الأداة الجديدة RYM. فتقومRYM بتوليد بيانات عشوائية وإدخالها في قاعدة البيانات Mongo DB ، وأثناء عملية الإدخال تقوم بقياس الـLatency والـThroughput كما موضح بالشكل (1.5) ، كما تقوم RYM بعمل تعديل على البيانات في العنقود Primary وتقوم بتسجيل الزمن الذي تم فيه التعديل ومن ثم تسجيل الزمن الذي تم فيه التعديل في العناقيد الـSecondaryو أخذ أكبر زمن من الـSecondary ، ومن ثم ايجاد القيمة المطلقة لحاصل طرح الزمنين ، والقيمة الناتجة من عملية الطرح هي الـStaleness .



شكل رقم(1.5) خوارزمية حساب الـLatency و الـ Throughput .

جدول رقم (1.5) مواصفات العقد

Specification	Primary	Secondary1	Secondary2
CPU	Core i7	Core i5	Core i7
RAM	8	8	8
Hard Disk	1 TB	1 TB	1 TB
Ubuntu	18.04LTS	18.04LTS	18.04LTS
Hadoop Version	2.7.3	2.7.3	2.7.3
MongoDB Version	4.0.2	4.0.2	4.0.2

خلاصة الباب:

في الباب السابق تم توضيح الطريقة التي نفذبها النظام المقترح اعتماداً على المعمارية المقترحة والأدوات المذكورة في الباب الثالث، كما تم توضيح الخطوات المتبعة لإجراء الإختبار .

الباب السادس

النتائج و التوصيات

1.6المقدمة:-

في هذا الباب سيتم عرض و مناقشة نتائج الاختبار .

2.6النتائج:-

أُجري الاختبار لمجموعتين من البيانات ذات أحجام مختلفة 50 GB و100GBتم توليدهم عشوائياً عن طريق مكتبة تابعة لـApache تسمى common-lang3 ، ووجد أنه كلما زادات قيمة الـThroughput كانذلك دليلاً على سرعة الأداء .أما بالنسبة للـlatency فنجد أنه كلما قل كان الأداء أفضل.

الجداولالتالية توضح قيماً متدنية للـLatency ومرتفعة للـThroughput مما يعني وبصورة واضحة أن أداء النظام ممتاز .

جدول رقم (1.6) يوضح الـThroughput لكل عنقود عند حجم بيانات 50GB .

Throughput (operation/second) 50GB			
Volume(Records)	**Primary Update**	**Secondary1 Read**	**Secondary2 Read**
10 Thousand	3206	7286	4694
100 Thousand	3633	5012	3102
1 Million	3644	3801	2513
20 Million	2787	4255	2280

جدول رقم (2.6) يوضح الـLatency لكل عنقود عند حجم بيانات 50GB .

Latency (Microsecond) 50GB			
Volume (Records)	**Primary**	**Secondary1**	**Secondary2**
1Thousand	632	143	202
10 Thousand	318	150	187
100 Thousand	238	145	196
1Million	239	141	221
20Million	321	140	242

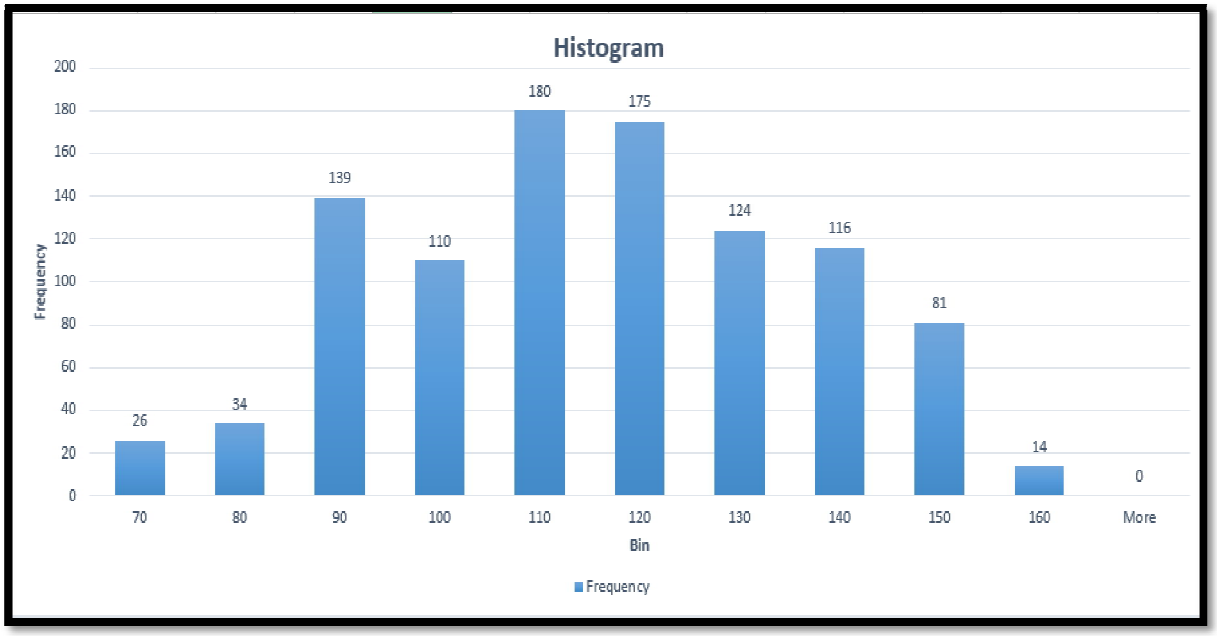
جدول رقم (3.6) يوضح الـThroughput لكل عنقود عند حجم بيانات 100GB .

Throughput (operation/second) 100GB			
Volume (Records)	**Primary Update**	**Secondary1 Read**	**Secondary2 Read**
1 Thousand	1000	8434	5546
10 Thousand	2193	5580	4415
100 Thousand	2605	7047	4008
20Million	3077	4139	2325

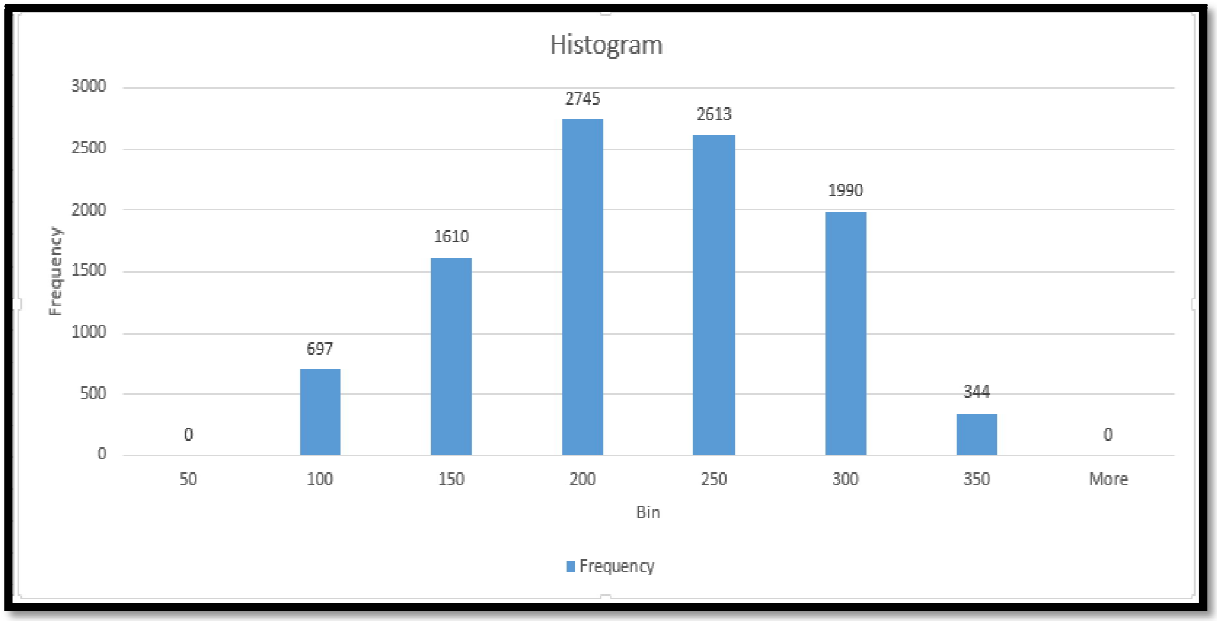
جدول رقم (4.6) يوضح الـLatency لكل عنقود عند حجم بيانات 100GB .

Latency (Microsecond) 100GB			
Volume (Records)	**Primary**	**Secondary1**	**Secondary2**
1 Thousand	971	130	181
10 Thousand	422	126	208
100 Thousand	239	131	187
1Million	422	185	193
20Million	292	135	225

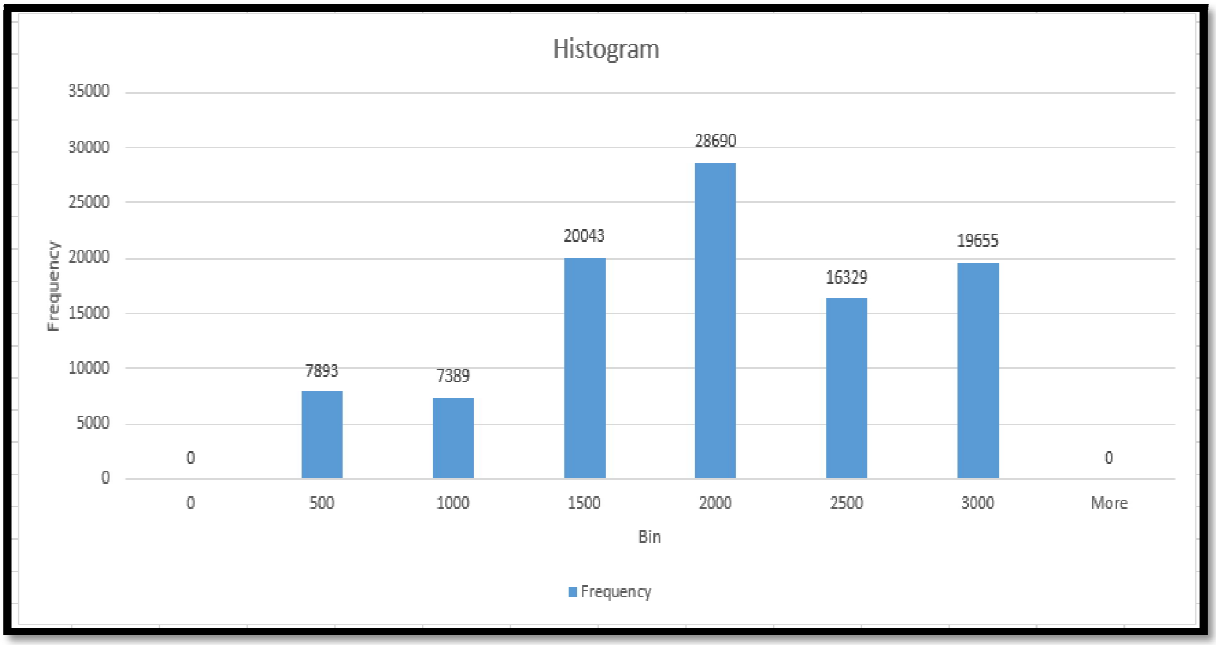
الصور أدناه توضح الـStaleness لمجموعة من البيانات ونسبة تكرار كل منها ، أن اغلب بيانات النتائج توضح تمركز البيانات في قيمة ضئيلة وبناء على النظرية المذكورة في**[17]** نجد ان الاتساق عالي جدا في النظام.



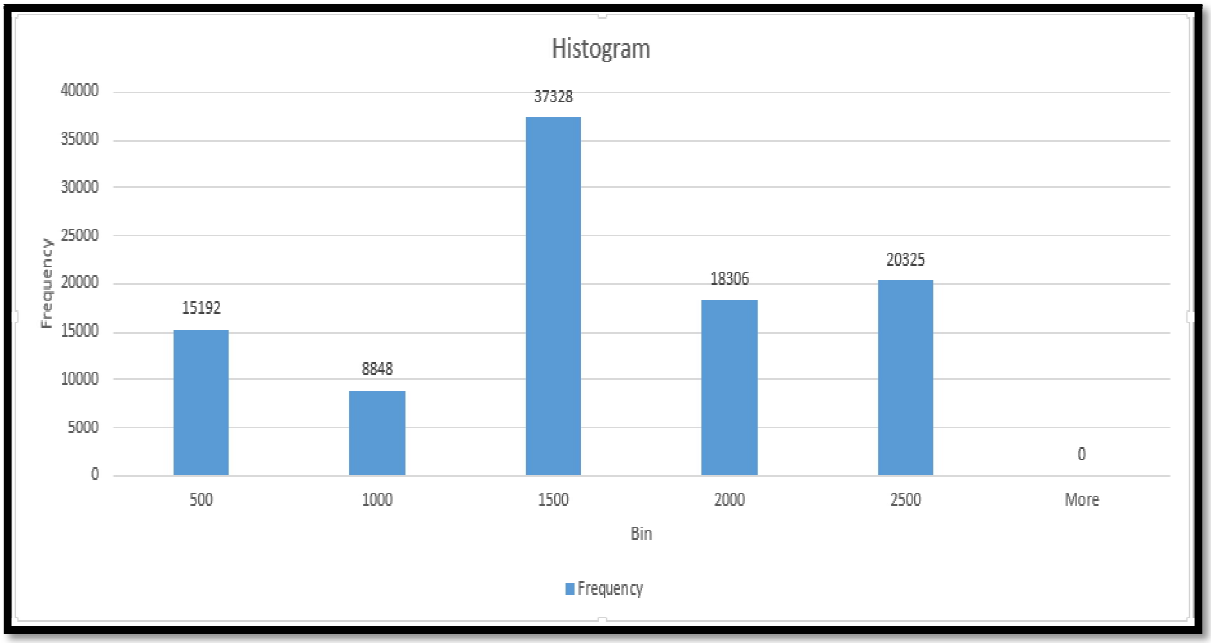
شكل رقم (1.6) يوضح الـstaleness عند إجراء عملية Update على1Thousand Records عند حجم بيانات 50GB .



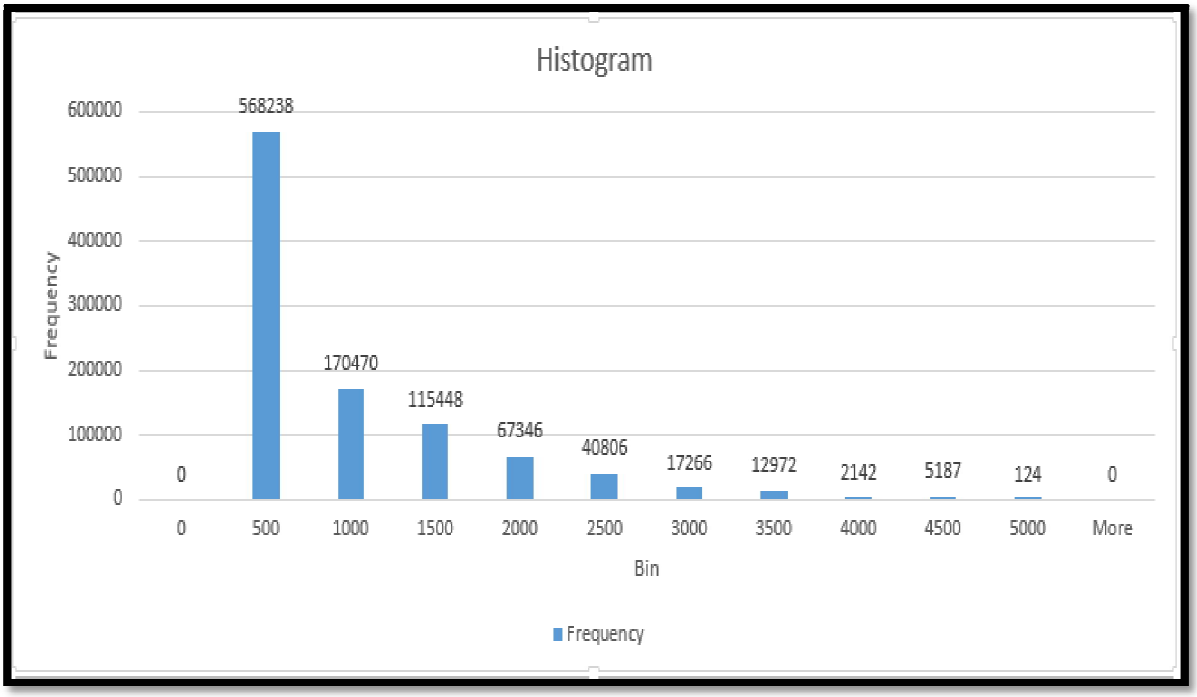
شكل رقم (2.6) يوضح الـstaleness عند إجراء عملية Update على 10Thousand Records عند حجم بيانات 50GB .



شكل رقم (3.6) يوضح الـstaleness عند إجراء عملية Update على 100Thousand Records عند حجم بيانات 50GB .



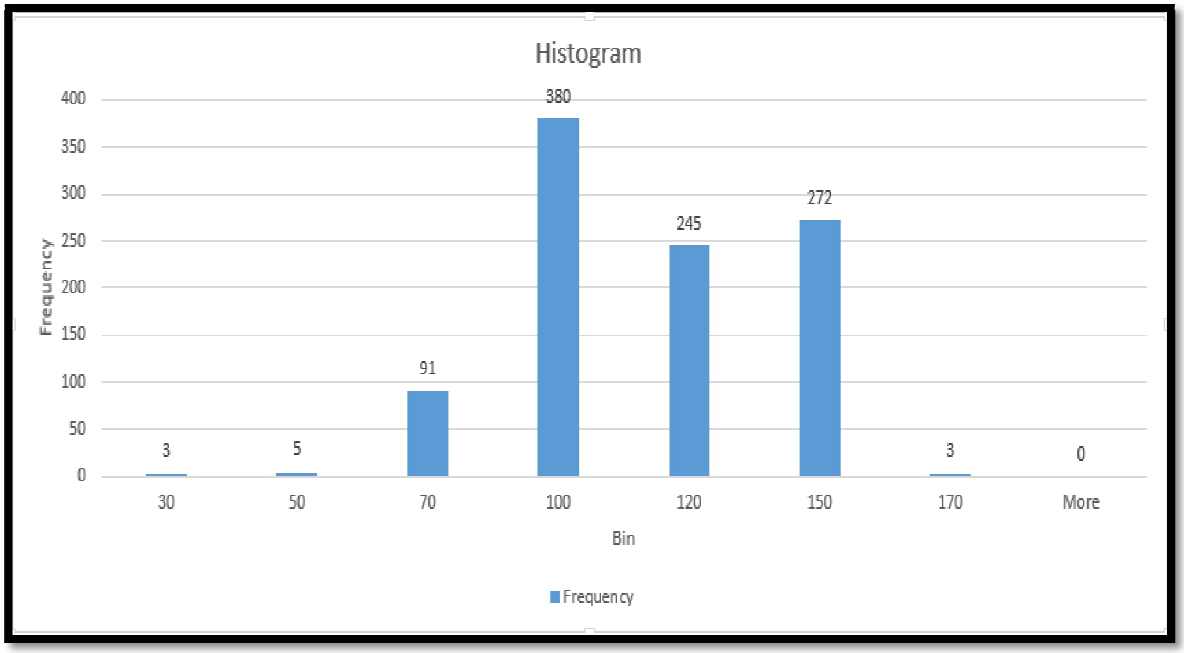
شكل رقم (4.6) يوضح الـstaleness عند إجراء عملية Update على1Millon Records عند حجم بيانات 50GB .



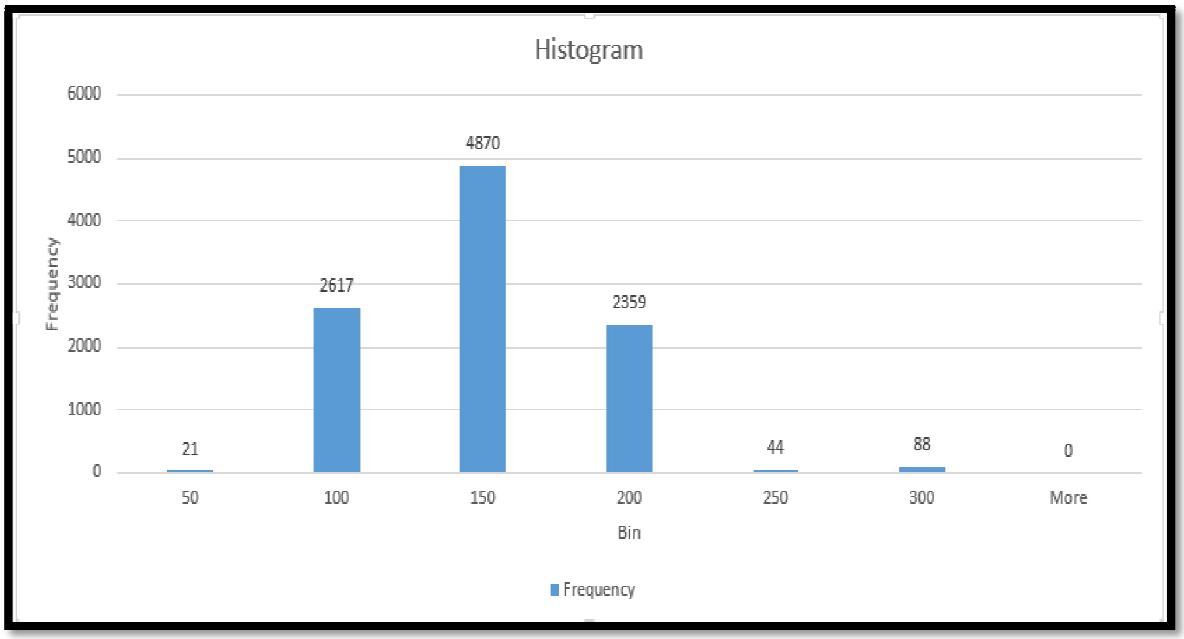
شكل رقم (5.6) يوضح الـstaleness عند إجراء عملية Update على 20Milion Records عند حجم بيانات 50GB .

جدول رقم (5.6) يلخص نتائج الـstaleness عند حجم بيانات 50GB .

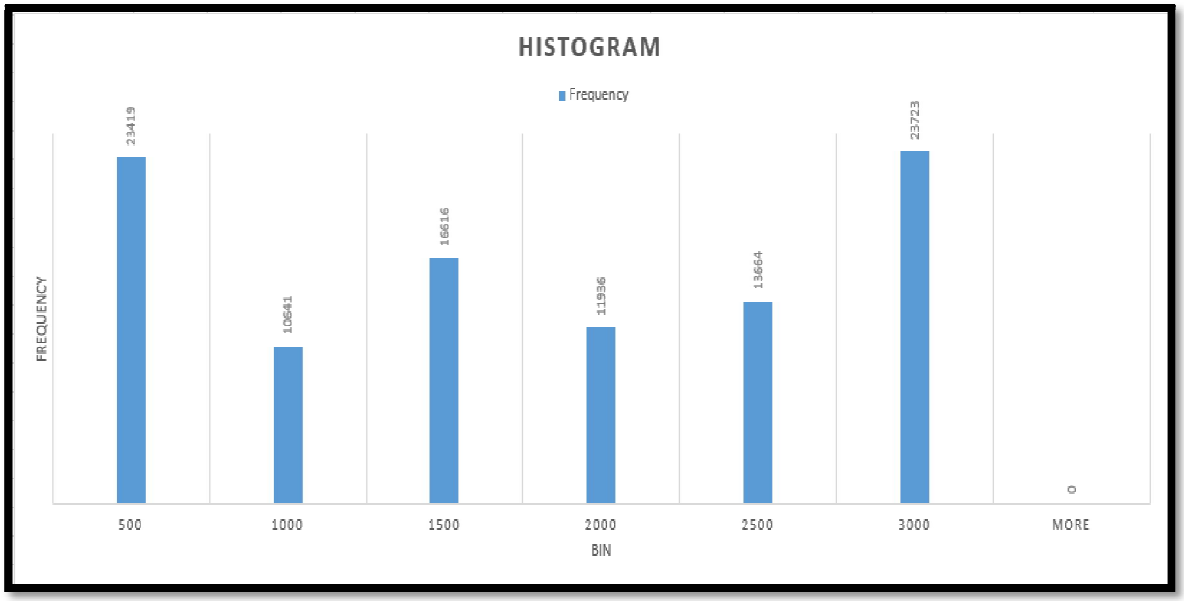
Volume (Records)	Millisecond	Second
1 Thousand	110	0.11
10 Thousand	200	0.20
100 Thousand	2000	2
1Million	1500	1.50
20Million	500	0.50



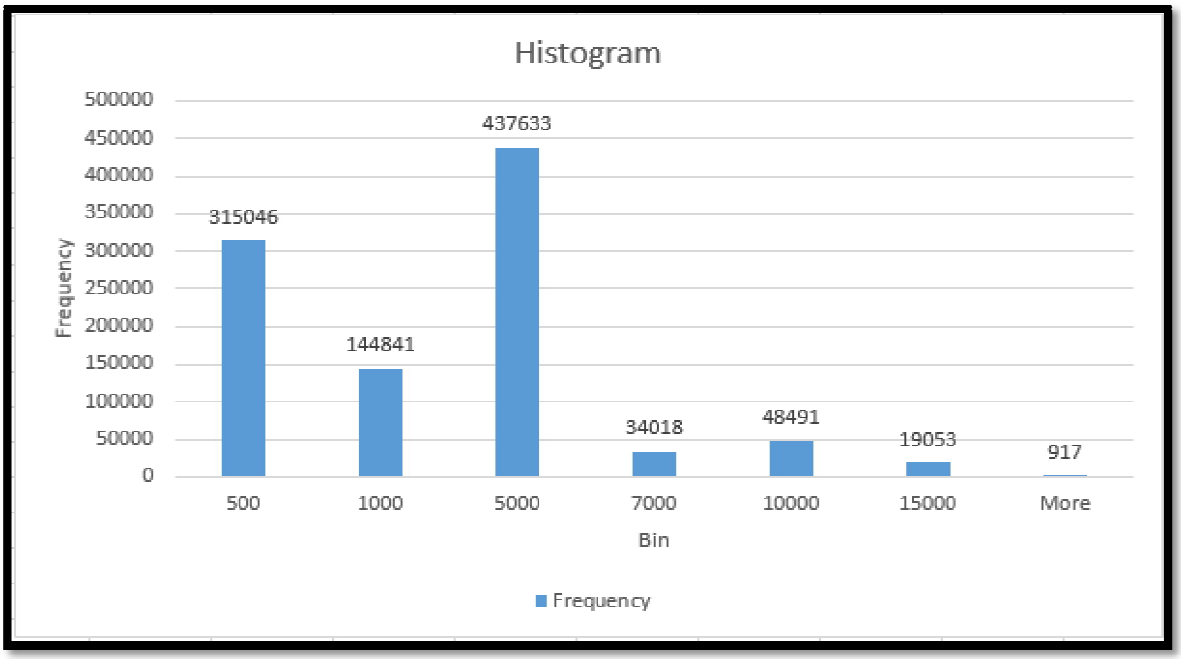
شكل رقم (6.6) يوضح الـstaleness عند إجراء عملية Update
على 1Thousand Records عند حجم بيانات 100GB .



شكل رقم (7.6) يوضح الـstaleness عند إجراء عملية Update
على 10Thousand Records عند حجم بيانات 100GB .



شكل رقم (8.6) يوضح الـstaleness عند إجراء عملية Update على 100Thousand Records عند حجم بيانات 100GB .



شكل رقم (9.6) يوضح الـstaleness عند إجراء عملية Update على 20Million Records عند حجم بيانات 100GB .

جدول رقم (6.6) يلخص نتائج الـstaleness عند حجم بيانات 100GB .

Volume (Records)	Millisecond	Second
1 Thousand	100	0.10
10 Thousand	150	0.15
100 Thousand	3000	3
20 Million	5000	5

3.6التوصيات:

1.تطوير الأداة RYM و جعلها قادرة على حساب معايير أخرى إضافية الي التي تم قياسها في هذا البحث.

2. تطوير و إضافة واجهات المستخدم للأداة RYM .

3. أن يتم الإستفادة من هذا النظام المقترح وتطبيقه على بيئة حقيقية .

4. أن تتجه الشركات في السودان لتطبيق تقنيات البيانات الضخمة والتي لا شك ستحقق لها ربحية اكبر.

4.6الخاتمة:

في هذا البحث تم بحمد الله اقتراح نظام يوازن بين الاتساق والاتاحيةفي إدارة البيانات الضخمة يواجه إحتياجات سوق العمل الحالي فيساعد المؤسسات في إدارة بياناتهم الضخمة بفعالية ، كما أن النظام يعمل بشكل موزع ومن أهم مميزات النظام أنه بني باستخدام أدوات مفتوحة المصدر مما سيقلل من تكلفته , ومن ثم تم اختبار النظام باستخدام الأداة RYM التي تم تطويرها في هذا البحث لقياس الإتساق والإتاحية بإستخدام خوارزميات معيارية. وقد اثبتت النتائج لعملية القياس أن النظام يوفر قدر عالي من الإتساق والإتاحية .

المصادر و المراجع

المصادر و المراجع

[1]Keith Gordon, “Principles of Data ManagementFacilitatingInformation Sharing”, 2007, ISBN 978-1-902505-84-8, British Cataloguing in Publication Data.

[2]Ramez Elmasri ,Shamkant B. Navathe , “Fundamentals Of DatabaseSystems”, SixEDITION, ISBN-13: 978-0-136-08620-8, Library of Congress Cataloging-in-Publication Data.

[3]IGGI,“The Principles of Good Data Management”, 2nd Edition,July 2005, Office of the Deputy Prime Minister: London, Product Code: 05 PLUS 03146.

[4]Rahul Beakta, “Big Data and Hadoop: A Review Paper”, (2015), e-ISSN: 1694-2329, CSE Deptt. Baddi University of Emerging Sciences & Technology, Baddi, India.

[5]Mohammed Anas Tawela, “Open Source”, Boosla Site, CC BY-NC-SA 3.0, Version 1,2006.

[6]Deka Ganesh Chandra, “BASE analysis of NoSQL database”, Journal Future Generation Computer Systems, November 2015.

[7]Kavita Bhamra, “A Comparative Analysis of MongoDB and Cassandra”, Department of Informatics University of Bergen, November 2017.

[8]Mohammed A. Mohamed,Obey G. Altrafi “Relational vs. NoSQL Databases: A Survey”, International Journal of Computer and Information Technology, (ISSN: 2279 – 0764), Volume 03 – Issue 03, May 2014.

[9]Andrew Pavlo,Matthew Aslett, “What’s Really New with NewSQL?” Newsletter ACM SIGMOD Record, June 2016.

[10]Harshawardhan S. Bhosale1, Prof. Devendra P. Gadekar, “A Review Paper on Big Data and Hadoop”, International Journal of Scientific and Research Publications, Volume 4, Issue 10, October 2014 1 ISSN 2250-3153.

[11]David Bermbach,and Sherif Sakr, “Towards Comprehensive Measurement of Consistency Guarantees for Cloud-Hosted Data Storage Services”,Karlsruhe Institute of Technology ,Karlsruhe, Germany.

[12]Jabir Al Fatah, “Consistency Issues on NoSQL Databases: Problems with Current Approaches and Possible Solution(s)”, Independent thesis, 15 HE credits, for degree of Bachelor in Computer Science,Spring Term 2016.

[13]MHD Fawaz Enaya , “An Experimental Performance Comparison of NoSQL and RDBMS Data Storage Systems in the ERP System Odoo”,University of Magdeburg,November 28, 2016.

[14]Suyog S. Nyati; Shivanand Pawar, “Performance evaluation of unstructured NoSQL data over distributed framework”, 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI).

[15]Ashish R. Varma, Dr. A. K. Shrivastava, “File Replication and Consistency Maintenance in the Hadoop cluster using IRM Technique”,International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), July 2014.

[16]Sara Hassab Elrasool , Esra and etal , “Storing and Processing Big Data for Enterprise Resource Planning Systems”, 2017,Sudan University for science and Technology , College of computer science.

[17]Daniel J. Abadi, “Consistency Tradeoffs in Modern Distributed Database System Design”,Yale University,FEBRUARY 2012.

[18]Muntasir Raihan Rahman and Wojciech Golab, “Toward a Principled Framework for Benchmarking Consistency”, HotDep'12 Proceedings of the Eighth USENIX conference on Hot Topics in System Dependability,2012.

[19]Russell Sears, Brian F. Cooper,“ Benchmarking Cloud Serving Systems with YCSB”,Proceedings of the 1st ACM Symposium on Cloud Computing, June 2010.