مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (4) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (4) 2015

دراسة تأثير تغير حجم البيانات على أداء نظم إدارة قواعد البيانات السحابية والهجينة والتقليدية

الدكتورة ألفت جولحة*
فاطمة أبوشامة**

(تاريخ الإيداع 20 / 5 / 2015. قُبِل للنشر في 30/ 8 / 2015)

□ ملخّص □

مع النمو المتسارع لحجم البيانات المخزنة في الأنظمة السحابية تصبح الحاجة إلى المعالجة الفعالة للبيانات أمراً حرجاً وملحاً. يقدم هذا البحث دراسة أهم خصائص نظم إدارة قواعد البيانات الآتية: Hive و SQLMR وMariaDB Galera. إن Hive هو نظام إدارة قواعد بيانات سحابي. وSQLMR هو نظام هجين يعتمد على الدمج بين قدرات النظم السحابية والتقليدية. أما MariaDB Galera فهو من نظم إدارة قواعد البيانات التقليدية المطورة لمواكبة الخصائص السحابية. تم في هذا البحث عرض أهم التطويرات التي تمت على تلك النظم ومقارنة أدائها في معالجة البيانات بالاعتماد على قياس زمن تنفيذ عمليات استعلام مع تغير حجم البيانات، وذلك من أجل التعرف على أداء تلك الأنظمة عملياً ومعرفة متطلبات تطويرها للوصول إلى نظام إدارة بيانات أمثلي، ولمساعدة المستخدمين في اختيار نظام قواعد البيانات الذي يحقق متطلباتهم من ناحية التوافرية والتدرجية.

الكلمات المفتاحية: الحوسبة السحابية، نظم إدارة قواعد البيانات السحابية، Apache Hadoop، Hive، SQLMR، عنقود MariaDB Galera، Shard-Query.

* مدرس- قسم هندسة الحاسبات والتحكم الآلي- كلية الهمك- جامعة تشرين- اللاذقية - سورية.
** طالبة دراسات عليا (ماجستير)- قسم هندسة الحاسبات والتحكم الآلي- كلية الهمك- جامعة تشرين- اللاذقية -سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (37) العدد (4) 2015
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (37) No. (4) 2015

Study of Data Size Changing Effect on Performance of Cloud, Hybrid, And Traditional DBMSs

Dr. Oulfat Jolaha*
Fatima Abo Shameh**

(Received 20 / 5 / 2015. Accepted 30 / 8 / 2015)

□ ABSTRACT □

With the rapid growth of the size of the data stored in the cloud systems, the need for effective data processing becomes critical and urgent. This research introduces a study of the most important characteristics of databases management systems: Hive, SQLMR, and MariaDB Galera. Hive is a cloud database management system. SQLMR is a hybrid system, which depends on the integration between the cloud and traditional systems capabilities. While MariaDB Galera is a traditional database management system developed to cope with the cloud characteristics. In this research, we show the most important developments that have been on those systems, and then we compare their performance in data processing based on the execution time of query operations with the change of the volume of data. That is to identify the performance of those systems practically and to know the developing requirements for access to optimized data management system, and to help users in the selection of the database system that achieves their requirements in terms of availability and scalability.

Key Wards: cloud computing, cloud DBMS, Apache Hadoop, Hive, SQLMR, MariaDB Galera cluster, Shard-Query.

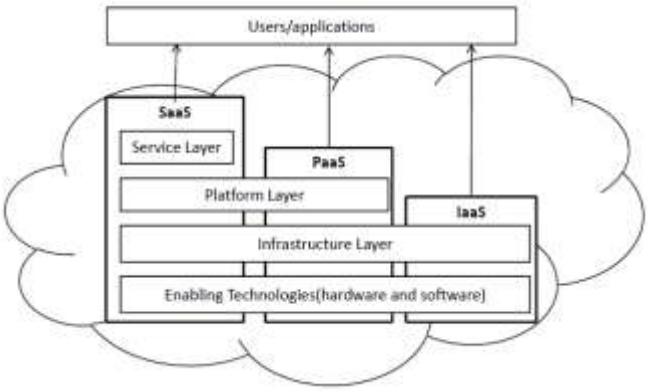
***Assistant Professor – Department of Computer and Automatic Control engineering- faculty of mechanical and electrical engineering –Tishreen University – Lattakia- Syria.**
****Postgraduate student (Master degree)- Department Of Computer And Automatic Control Engineering- Faculty Of Mechanical And Electrical Engineering –Tishreen University – Lattakia- Syria**

مقدمة:

تعرف الحوسبة السحابية بأنها نموذج يتيح الوصول الدائم والمريح وحسب الطلب إلى الشبكة لمجموعة من موارد الحوسبة القابلة للتشكيل مثل البنية التحتية والتطبيقات والخدمات والتخزين والشبكات، والتي يمكن توفيرها بسرعة بأدنى حد من مجهود الإدارة أو التفاعل مع مزودي خدمه هذه الموارد [1] .

تتميز الحوسبة السحابية بمجموعة من الخصائص التي تمكّن الزبائن من الحصول على قدرات الحوسبة تلقائياً حسب الحاجة وبشكل فردي مثل وقت المخدم والتخزين الشبكي، ومن دون تطلب التفاعل البشري مع كل مزود خدمة. ويمكن الوصول إلى الإمكانات المتاحة عبر الشبكة عن طريق الآليات القياسية التي تدعم استخدام الشبكة بواسطة منصات العملاء المختلفة مثل الهواتف الخلوية والحواسيب اللوحية والحواسيب المحمولة والبنية التحتية ومحطات العمل. يتم تجميع موارد حوسبة المزود لخدمة عدة زبائن باستخدام نموذج متعدد المستأجرين multi-tenant model للموارد الفيزيائية والافتراضية المختلفة، وذلك بتخصيصها وإعادة تخصيصها بشكل ديناميكي وفقاً لطلب المستهلك. يمكن التزويد بالإمكانيات وإطلاقها بمرونة (وفي بعض الحالات بشكلٍ تلقائي) لتوسيع نطاق المدخلات والمخرجات بسرعة وبما يتناسب مع الطلب. وكذلك تمكّن النظم السحابية من التحكم التلقائي باستخدام الموارد بشكل أمثلي من خلال الاستفادة من القدرة على القياس بمستوى معين من التجريد المناسب لنوع الخدمة كالتخزين والمعالجة[1].

تنقسم خدمات الحوسبة السحابية إلى ثلاث فئات أساسية، كما يبين الشكل (1)، وهي: البنية التحتية كخدمة IaaS (Infrastructure as a Service) والمنصة كخدمة PaaS (Platform as a Service) والبرمجيات كخدمة SaaS (Software as a Service).



الشكل (1) الخدمات الأساسية للحوسبة السحابية [3].

يعتبر نمط البنية التحتية كخدمة هو النمط الأساسي، بينما تكون الأنماط الأخرى، وكلما تدرجنا نحو الأعلى، عبارة عن تجرد من التفاصيل الموجودة في الأنماط الأدنى، وهذا الأمر يتضح من عرض مفهوم تلك الأنماط وماهية خدمة الحوسبة السحابية التي يتم تقديمها في كل نمط.

البرمجيات كخدمة SaaS: تقدم للمستهلك تطبيقات مزود الخدمة التي تعمل على البنية التحتية السحابية. إن البنية التحتية السحابية هي مجموعة من الأجهزة والبرمجيات التي تحقق الخصائص الأساسية للحوسبة السحابية المذكورة أعلاه. تحتوي البنية التحتية السحابية على كل من الطبقة المادية وطبقة التجريد، وتتكون الطبقة المادية من أجهزة الموارد الضرورية لدعم الخدمات السحابية المقدمة، وتشمل عادة الخادم والتخزين ومكونات الشبكة. كما تتكون

طبقة التجريد من البرامج المنتشرة في الطبقة المادية، التي تُظهر خصائص السحابة الأساسية. يمكن الوصول للتطبيقات من مختلف أجهزة العميل مثل متصفح الإنترنت أو واجهة البرنامج. لا يستطيع المستهلك إدارة أو التحكم بالبنية التحتية السحابية الأساسية بما في ذلك الشبكة والمخدّمات وأنظمة التشغيل والتخزين، أو حتى قدرات تطبيق الفردية، مع احتمال استثناء بعض إعدادات تكوين التطبيق الخاصة بالمستخدم [3,2].

المنصة كخدمة PaaS: هو المستوى الثاني من الحوسبة السحابية وتقدم المنصة السحابية والبيئة البرمجية التي يستطيع المستخدمون من خلالها إنشاء واختبار وتشغيل تطبيقاتهم. يقوم مزود الخدمة بإدارة والتحكم بنظم التشغيل ومكونات الشبكة والعتاد الصلب (البنية التحتية الأساسية). أما إعدادات التطبيقات وخصائص بيئة العمل المرتبطة بها يتم إدارتها من قبل المستخدمين[3,2].

البنية التحتية كخدمة IaaS: هي الأساس أو الطبقة السفلية من الحوسبة السحابية وأحيانا يشار إليها باسم الأجهزة كخدمة Hardware as a Service، وهي تمكّن مستخدمي السحابة من استخدام موارد الحوسبة الأساسية مثل التخزين ووحدات المعالجة والشبكات وغيرها، وتمكّن المستخدم من تنصيب وتشغيل البرامج المختلفة التي تشمل أنظمة التشغيل والتطبيقات والتحكم بخصائصها. ولا يملك المستخدم القدرة على التحكم بالبنية التحتية الأساسية للسحابة ولكن يمكن له التحكم ببعض خصائص الشبكة مثل الجدار الناري للمزود[3,2].

مع زيادة الطلب على نظم إدارة قواعد البيانات السحابية اتجه العديد من المطورين إلى دراستها والعمل على تطوير بنيتها وتحسين أدائها من خلال القدرة على المعالجة المتوازية للبيانات الموزعة على العقد المخدمة، والتي يمكن أن تنضم بشكل تلقائي، وفي حال خروج أي عقدة عن الخدمة لا يتأثر الأداء العام للعنقود المعني. كذلك اتجه العمل على ضمان التوافرية العالية والاتساق القوي وسماحية للخطأ Fault–Tolerance، والتي على أساسها صممت نظم إدارة البيانات السحابية. لقد تمت دراسة طرق تخزين البيانات في نظم السحابة مثل Amazon، Yahoo، Hadoop وغيرها وتصنيفها ومقارنتها، وتلخيص المناهج البحثية لإدارة البيانات في هذه النظم [4]. وقدمت بعض المقترحات لتعديل بعض التقنيات مثل 2PC (two phase commit protocol) التي تم تجاهلها لعدم ملائمتها مع قواعد البيانات المبنية على أسس العنقدة. بالإضافة إلى تطوير بعض تقنيات النسخ المتماثل replication في قواعد البيانات لزيادة فعالية التناقلات في السحابة ROWA protocols إما باستخدام certification–based أو primary copy ones، وذلك لدعم المناقلات في قواعد البيانات السحابية [5]. كذلك تم تصميم نظام SQLMR الذي يعتمد على الدمج بين قدرات النظم السحابية والتقليدية وتمت مقارنة أدائه مع أشهر نظم قواعد البيانات التقليدية SQL cluster والسحابية (Hbase و Hive) بالاعتماد على زمن تنفيذ عمليات الاستعلام [6].

أهمية البحث وأهدافه:

يتم في هذا البحث عرض أهم خصائص وآلية عمل كل من نظام تخزين Hadoop مع التركيز على أشهر تطبيقاته Hive، ونظام SQLMR الهجين، وعنقود MariaDB Galera المطور حديثاً والذي يدعم الحوسبة السحابية. ومن ثم تمت مقارنة أداء تلك النظم في معالجة البيانات عملياً من خلال إجراء عمليات استعلام من خلالها وقياس زمن التنفيذ مع تغيير حجم تلك البيانات، وذلك لمعرفة متطلبات تطويرها للوصول إلى نظام إدارة بيانات أمثلي يواكب التغييرات المتسارعة ومتطلبات المستخدمين وضمان التوافرية والتدرجية والاستقرار التي تَعِد بها الحوسبة السحابية.

طرائق البحث ومواده:

توجد العديد من أنظمة إدارة قواعد البيانات السحابية المدفوعة بالإضافة للعديد من الأنظمة المفتوحة المصدر بهدف تطويرها والتعديل عليها من قبل المطورين والباحثين. يتم في هذا البحث اعتماد تلك الأنظمة مفتوحة المصدر، حيث يتم تناول دراسة طريقة عمل وأهم خصائص أنظمة إدارة قواعد البيانات السحابية التالية Apache Hadoop وأشهر تطبيقاته Hive، ومن ثم نظام SQLMR الهجين، يليه نظام عنقود MariaDB Galera الذي يدعم الحوسبة السحابية. ومن ثم إجراء عمليات استعلام على كل منها لاختبار أدائها في معالجة البيانات متغيرة الحجم والمقارنة فيما بينها، أضف إليها مقارنة أدائها مع أحد النظم التقليدية MySQL والتطويرات التي تمت على هذه الأنظمة ومدى تحسن أدائها مع تلك التطويرات بالمقارنة مع النظم أخرى.

1-نظام إدارة قواعد البيانات السحابية Apache Hadoop:

يعمل المشروع الذي يسمى Hadoop من Apache على تطوير برمجيات مفتوحة المصدر بهدف تحقيق الحوسبة الموزعة ذات الموثوقية الكبيرة والقدرة على التوسع والقدرة على الاستشعار والكشف والتعامل مع حالات الفشل وذلك ضمن مستوى التطبيقات، مما يقدم خدمة توافرية عالية لأجهزة البنية العنقودية، والتي قد يكون كل منها عرضة للفشل. ومن أهم الخصائص التي تميز Hadoop هي قدرته على تقسيم البيانات وإجراء عمليات الحوسبة عليها بالتوازي على آلاف الأجهزة المضيفة.

يتكون مشروع Hadoop من [7]:

- أدوات Hadoop المشتركة: وهي الأدوات التي تدعم عمل الأجزاء الأخرى في المشروع.
- نظام ملفات Hadoop الموزع Hadoop Distributed File System (HDFS): وهو يؤمن وصولاً عالي الإنتاجية لبيانات التطبيقات.
- Hadoop Yarn: هو عبارة عن بيئة عمل تؤمن جدولة المهمات وإدارة موارد البنى العنقودية.
- Hadoop MapReduce: هو نظام يعتمد على YARN يقدم معالجة متوازية لمجموعات كبيرة من البيانات.

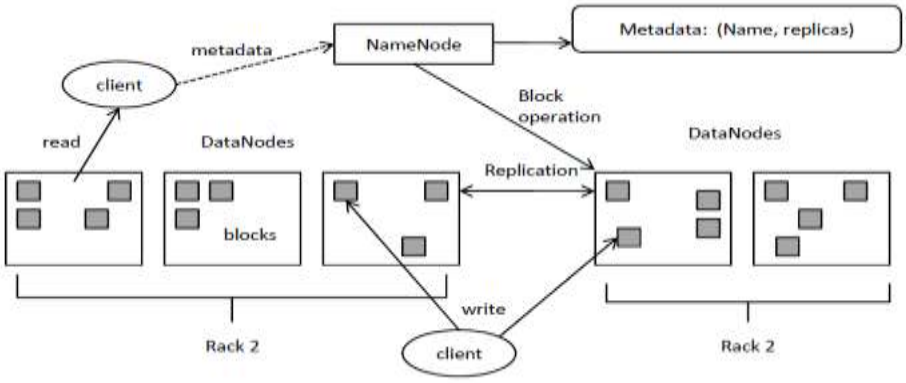
بالإضافة إلى عدة مشاريع أخرى متعلقة به مثل Hive وHBase وZooKeeper وCassandra.

فيما يلي يتم عرض آلية عمل نظام ملفات Hadoop الموزع، ومن ثم طريقة عمل Hadoop MapReduce حيث تم شرح كيفية عمل تطبيق معتمد على Yarn وتمت المقارنة بين Yarn و MapReduce1. بعد ذلك تم عرض نظام Hive وآلية معالجته للبيانات.

1.1. نظام ملفات Hadoop الموزع Hadoop Distributed File System (HDFS)

يعمل نظام HDFS على أجهزة تجارية ويقدم وصولاً عالي الإنتاجية لمجموعات كبيرة من البيانات. يمتلك هذا النظام بنية سيد/تابع master/slave التي تتألف من مخدم رئيسي واحد وعقدة اسمية واحدة وعدة عقد بيانات. يبين الشكل (2) هيكلية نظام ملفات Hadoop. يقوم المخدم الرئيسي بإدارة فضاء الأسماء name space الخاص بنظام الملفات وينظم الوصول للملفات من قبل العملاء [8]. تقوم عقد البيانات بتنفيذ طلبات القراءة والكتابة التي يتم طلبها من قبل العميل، وكذلك تقوم بتنفيذ طلبات الإنشاء أو الحذف أو النسخ المتماثل للكتل replicate block التي تصدرها العقدة الاسمية NameNode. يتم تجزئة كل ملف إلى عدة كتل يتم تخزينها ضمن مجموعة من عقد البيانات. تقوم العقدة الاسمية بتخزين البيانات التعريفية meta data الخاصة بنظام الملفات، وهي تعدّ مسؤولةً عن فتح وإغلاق

وإعادة تسمية الملفات والمجلدات directories. ويكون حجم جميع الكتل متساوٍ (عادة 64MB) ما عدا الكتلة الأخيرة. يتم النسخ المتماثل للكتل من أجل جعل المنظومة أكثر قدرة على التعامل مع تصحيح الأخطاء. يتم ضبط عدد الكتل والنسخ المتماثلة لكل ملف على حدة. ويتم تحسين وأمثلَة توضع النسخ المتماثلة بتطبيق سياسة rack–aware، مما يحقق الاعتمادية والتوافرية العالية والاستخدام الأمثل للشبكة [8,9]. يتم تلخيص استراتيجية توضع النسخ المتماثلة بأنه لا توجد عقدة بيانات تحوي أكثر من نسخة متماثلة للكتلة ولا يوجد رف rack يحوي أكثر من نسختين متماثلتين لنفس الكتلة شرط وجود عدد كافٍ من الرفوف في العنقود[8].



الشكل (2) هيكلية نظام ملفات Hadoop الموزع [8].

2.1. Hadoop MapReduce

إن Hadoop MapReduce هو نموذج برمجي لمعالجة البيانات، فهو يقوم بتسهيل وتبسيط معالجة كميات هائلة من البيانات الموزعة على بنى عنقودية كبيرة بطريقة موثوقة وكذلك يقوم بتصحيح للأخطاء، حيث يصل حجم هذه البيانات إلى رتبة بيتا بايت والآلاف من العقد. كما أنه يقوم بمعالجة حوسبية متوازية على كل من البيانات غير الهيكلية كنظام الملفات والبيانات الهيكلية كقواعد البيانات وذلك عن طريق تقسيم المهام إلى مهام جزئية وتوزيعها على العقد الخادمة التي تعيد نتيجة معالجتها، فتقوم العقدة الرئيسية بتجميع النتائج. يقوم Hadoop بتشغيل برنامج MapReduce المكتوب بلغات مختلفة مثل Java, Ruby Python, C++ [4,9].

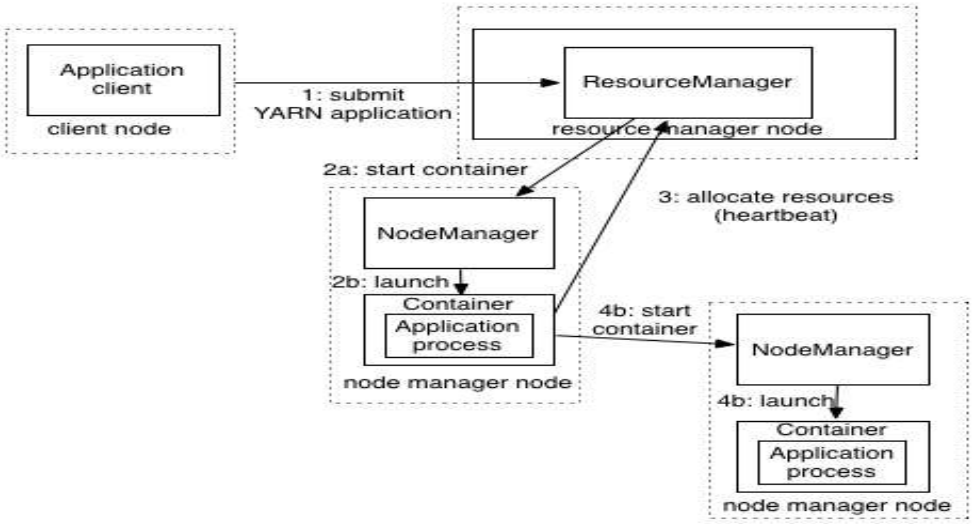
إن MapReduce سيصل إلى حالة الاختناق من أجل البنى العنقودية في المنطقة الواحدة والمؤلفة من أكثر من 4000 عقدة. لذا قامت مجموعة Yahoo بالبدء بتصميم جيل جديد من MapReduce. وبعد تعديل شامل تماماً نتج عنه نسخة جديدة اعتباراً من نسخة Hadoop–0.23 هي النسخة MarReduce2 (MRv2) أو YARN، أما النسخة القديمة فتدعى MapReduce الكلاسيكية أو (MRv1) MarReduce1. وفيما يلي سيتم عرض كيفية إدارة عملية MapReduce لكل من النسختين المذكورتين أعلاه.

3.1. كيفية عمل تطبيق معتمد على Yarn [9,10]

تؤمن Yarn في صلب عملها نوعين من العمليات المخفية daemons (أي التي تجري في الخلفية): مدير للموارد (مدير واحد لكل عنقود) يعمل على تنظيم استخدام الموارد عبر كامل العنقود، ووحدات لإدارة العقد تتواجد في كافة العقد وتعنى بعمليات إقلاع الحاويات containers ومراقبتها. وتعمل الحاوية على تنفيذ عمليات خاصة بالتطبيقات بمجموعة من الموارد التي تفرض عليها قيوداً معينة (الذاكرة، المعالج وغيرها). ويمكن أن تكون الحاوية –

اعتماداً على الإعدادات التي ضبطت وفقها Yarn– إما عبارة عن عملية من نمط Unix أو مجموعة cgroup الخاصة بـLinux.

ومن أجل تشغيل تطبيق على Yarn، يقوم الجهاز الزبون بالاتصال بمدير الموارد ويطلب منه تشغيل عملية من نوع application master أي قائد التطبيقات (كما هو موضح في الخطوة 1 من الشكل (3)). بعد ذلك يقوم مدير الموارد بإيجاد مدير عقدة قادر على تشغيل العملية المذكورة في حاوية خاصة (كما في الخطوتين 2a و 2b في الشكل (3)). عند هذه النقطة يجب القول إن ما يفعله قائد التطبيقات بالتحديد عند بداية تشغيله في الحاوية يعتمد على ماهية التطبيق الذي يُنفذ. فقد يعمل ببساطة على إجراء عمليات حسابية ويعيد النتيجة إلى الجهاز الزبون، أو قد يطلب المزيد من الحاويات من وحدات إدارة الموارد (الخطوة الثالثة) بهدف إجراء عمليات حسابية موزعة (الخطوة 4a و 4b في الشكل (3)). المهمة الأخيرة المذكورة هي ما يفعله تطبيق MapReduce المعتمد على Yarn، وهذا ما سننظر إليه بشيء من التفصيل في دراستنا لتشغيل مهمة من نوع MapReduce.



الشكل(3) كيفية تشغيل تطبيق في Yarn[10].

يلاحظ من الشكل(3) أن Yarn لا يقدم أي وسيلة للتواصل بين أجزاء التطبيق (الجهاز الزبون، قائد التطبيقات، والعملية نفسها). حيث تستخدم معظم تطبيقات Yarn التي لا تتصف بالبساطة أو البديهية شكلاً ما من أشكل التواصل عن بعد (مثل اعتماد طبقة Hadoop والمسماة RPC) لتمرير تحديثات عن سير العمل ونتائج المهمات إلى الأجهزة الزبونة، إلا أن هذه النتائج والتحديثات تعتمد على التطبيق المعني.

4.1. مقارنة بين Yarn و MapReduce 1 [9,10]

لقد تم عرض كيفية تشغيل تطبيق ما باستخدام Yarn في الفقرة السابقة، وللمقارنة بين Yarn و MapReduce 1 لابد من عرض آلية عمل MapReduce 1 التي يمكن تلخيصها كالآتي. يوجد في MapReduce 1 نوعان من الخدمات المخفية daemon التي تتحكم في تنفيذ الأعمال: متابع الأعمال jobtracker ومتابع واحد أو أكثر للمهام tasktracker. ينسق متابع الأعمال كافة الأعمال التي تعمل في المنظومة عير جدولة المهمات العاملة بدورها على متابعي المهام. ويقوم متابعي المهام بتشغيل المهام وإرسال تقارير عن سير العمل إلى متابع الأعمال، الذي يراقب سير العمل الإجمالي لكل عمل. وعند فشل مهمة ما، يمكن لمتابع الأعمال أن يعيد جدولتها في متابع مهمات آخر.

يعنى متابع الأعمال في MapReduce 1 بكل من جدولة الأعمال (أي تخصيص متابع لكل مهمة) وأيضاً مراقبة سير تنفيذ المهمات (وهذا يتضمن متابعة كل مهمة، وإعادة إقلاع المهمات التي فشل تنفيذها أو التي تنفذ ببطء، والعناية بتسجيل المهمات، مثل متابعة مقادير العدادات). في حين يتم تقسيم هذه المسؤوليات بين كيانين منفصلين: مدير الموارد وقائد التطبيقات (قائد لكل عمل MapReduce). ويكون متابع الأعمال أيضاً مسؤولاً عن تخزين سجل للأعمال المنجزة، على الرغم من إمكانية تشغيل مخدم مخصص لسجل الأعمال المنجزة كخدمة مخفية منفصلة لتخفيف العبء عن متابع الأعمال. أما في Yarn فيلعب مخدم الخط الزمني timeline server الدور المكافئ للمهمة المذكورة آنفاً عبر تخزينه لسجل يحوي ما تم تشغيله من تطبيقات. حيث يتم تشغيل العمليات MapReduce 1 في slot أما في Yarn فيتم تشغيلها في حاويات خاصة. يكافئ مدير العقد في YARN متابع الأعمال في MapReduce 1. ويلخص الجدول(1) عملية المقارنة والتكافؤ بين المنظومتين.

الجدول (1) عملية المقارنة والتكافؤ بين MapReduce 1 وYARN [10]

MapReduce 1	YARN
متابع الأعمال	مدير العقد، قائد التطبيقات مخدم السير الزمني
متابع المهام	مدير العقد
المجال slot	الحاوية

لقد تم تصميم Yarn مع الأخذ بالحسبان نقاط ضعف MapReduce 1. يتميز Yarn بالتوافرية العالية وقابلية التوسع حيث يصل عدد العقد إلى حدود 10000عقدة وعدد المهمات 100000 مهمة في Yarn، وبالمقابل يصل عدد العقد إلى حدود 4000 عقدة و 4000 مهمة في MapReduce1. بالإضافة للاستخدام الأمثل للموارد وقابلية تغير التوضع أي أنه يجعل Hadoop متاحاً لتطبيقات توزيعية أخرى بخلاف MapReduce1، والذي يُعَد تطبيقاً واحداً من بين العديد من التطبيقات الأخرى التي تقدمها Yarn. حتى أنه من الممكن للمستخدمين أن يشغلوا إصدارات مختلفة من MapReduce على نفس عنقود Yarn، مما يسهل إدارة عمليات التحديث المطبقة على MapReduce.

2. نظام Apache Hive

يُعد نظام Hive مستودعاً للبيانات بني على قمة نظام Hadoop. يستخدم في معالجة السجلات واستخراج المعلومات من النصوص وفهرسة الوثائق والنمذجة التنبؤية [4] . لقد بني نمط بيانات Hive بالاعتماد على الجداول، حيث إن أعمدة الجداول إما أن تنتمي لأنواع SQL أو أنواع مركّبة، وإن الأخيرة تتضمن الربط والقوائم التي تسمح بتحميل بيانات شبه مهيكلة (مثل ملفات JSON). ويكون لكل جدول مسار في نظام الملفات HDFS، حيث يتم ترتيب بيانات الجدول بشكل متسلسل وتخزن في ملف ضمن مسار محدد. ويتم تجزئة الجداول بناءً على الأعمدة، وكل قيمة فريدة لأزواج جزء–مفتاح partition–key تحدد جزءاً من الجدول. ولا يتم تخزين الأعمدة المجزأة مع بياناتها بل يتم تعريف مسارات فرعية لتخزين البيانات، حيث إن كل جزء يتم تمثيله بمسار فرعي ضمن مسار الجدول. ويمكن تجزئة الجزء الناتج تجزئة إضافية ذو طبيعة مبعثرة إلى buckets. وإن كل bucket يخزن في ملف ضمن المسار الفرعي للجزء المطروح. ويتم استخدام معلومات buckets لتحقيق عمليات الأمثلة من قبل مخطط الاستعلام وخصوصاً من أجل أمثلة عمليات ضم الجداول [9,11].

لقد طور نظام Hive لغة استعلام شبيهة بـ SQL تسمى لغة استعلامات Hive (HiveQL) تسمح لمستخدمي SQL بإجراء عمليات الاستعلام. كما وتسمح أيضاً للمبرمجين بكتابة أكواد MapReduce لإجراء تحليل أعقد وأرقى للبيانات قد لا يكون مدعوماً من قبل الخصائص التي بنيت عليها HiveQL في الأصل [11].

في إدارة موارد إطار عمل المعالجة يتم تنفيذ أعمال MapReduce بالاعتماد على نسخة hadoop، فإذا كانت النسخة المستخدمة هي hadoop1 أو hadoop2 يتم تحويل الاستعلامات إلى أعمال MapReduce ويتم تنفيذها باستخدام MRv1 أو MRv2، على الترتيب. وأخيراً لابد من ذكر أن نظام التخزين المستخدم في Hive هو HDFS.

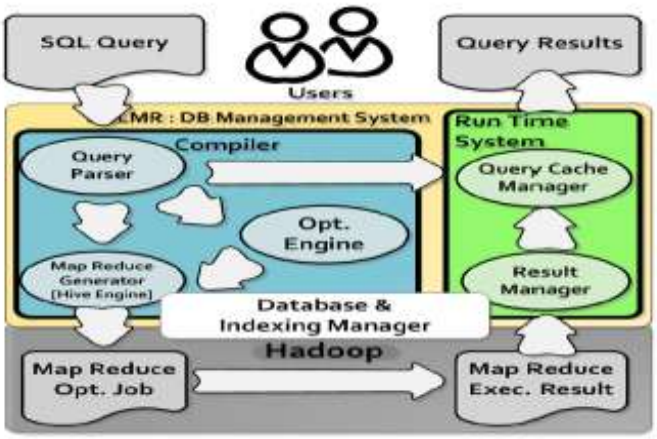
3. نظام SQLMR الهجين:

SQLMR (Structured Query Language Mapreduce) هو نظام هجين يدمج بين قدرات البرمجة الخاصة بـ SQL مع قدرات التدرج والعمل في عناقيد غير متجانسة واستيعاب الخطأ التي تتمتع بها MapReduce. إنه يمكّن المستخدمين من كتابة برامج لإدارة البيانات باستخدام لغة SQL المألوفة أو حتى تشغيل البرامج الموجودة أصلاً دون الحاجة إلى القيام بتعديلات. ولتحقيق الأداء العالي في معالجة البيانات تم أيضاً استخدم عدد من طرق الأمثلة.

يبين الشكل (4) البنية المعمارية لنظام SQLMR، حيث نلاحظ أن أهم ما يميز بنية SQLMR هو معرب أكواد SQL الذي يحول تلك الأكواد إلى أكواد MapReduce. حالياً يدعم نظام SQLMR مجموعة جزئية من استعلامات SQL؛ وإن تلك الاستعلامات كافية بشكل تقريبي لدعم تطبيقات إدارة البيانات ذات النطاق الواسع التي تتمتع بصفة تحليلية، مثل المعالجة التحليلية online analytical processing (OLAP) التي يمكن إجراؤها على الشبكة والتنقيب في البيانات data mining...والخ. يقوم SQLMR بالاتصال بعدة مخدمات SQL ليشكل نظاما هجيناً لإدارة البيانات يتم فيه معالجة الاستعلامات وعمليات الكتابة الصغيرة الحجم في مخدمات SQL، في حين تعالج الاستعلامات والعمليات الكبيرة عبر نظام SQLMR. كما أن نظام SQLMR يدعم تقنية إنشاء ملفات بيانات تتمتع بأعباء وتكاليف منخفضة، تعمل على التحويل الديناميكي السريع من ملفات قواعد بيانات SQL إلى ملفات ذات نمط نظام الملفات الموزع Hadoop أي HDFS والتي يمكن قبولها من قبل البيئة التشغيلية لنظام MapReduce كملفات دخل. لذلك فإن هذه التقنية تقلل بشكل كبير الوقت المستهلك في تحويل الملفات بين SQL وMapReduce [6].

كذلك يقوم نظام SQLMR بعمليات فهرسة وتقسيم فعالة لقاعدة البيانات بما يضمن إيجاد سريع للبيانات المستعلم عنها في نظام HDFS وتقليل عمليات الدخل والخرج المطلوبة عند الاستعلام على مجالات من البيانات. يتم استيداع نتائج الاستعلام في ذواكر مخبئية تجنباً لإعادة معالجة الاستعلامات المكررة والزائدة عن الحاجة. فعندما يدخل استعلام جديد إلى نظام SQLMR، يقوم المعرب بتمرير الاستعلام إلى وحدة إدارة نتائج الاستعلام التي تحوي الاستعلامات السابقة في السجل. وفي حال وجود نتائج متوافقة مع هذا الاستعلام في الذاكرة المخبئية يتم إعادة هذه النتائج إلى المستخدم بدون إعادة تنفيذ الاستعلام. وإلا يقوم المعرب بتحليل الاستعلام ويولد كود MapReduce أمثلي. وتصبح النتائج في الذاكرة غير صالحة عندما يقوم المستخدم بالحذف أو التعديل في قاعدة البيانات. تم تطبيق تقنيات أمثلة على نظام HDFS التشغيلي الخاص بـ MapReduce من أجل التقليل الإضافي لزمن معالجة الاستعلامات. حيث يقوم المعرب بتوليد مهام MapReduce أمثلية وينفذها على نظام Hadoop. وقد وظفت عدة تقنيات أمثلة، ومنها نذكر تقنية تواصل الرفوف المتقاطعة cross-rack communication optimization، لتحسين

الأداء العام لنظام Hadoop. حيث إن نظام Hadoop المحسّن هو عبارة عن إطار عمل برمجي مخصص للمعالجة الموزعة لمجموعات كبيرة من البيانات المخزنة على عناقيد حاسوبية [6].



الشكل(4) البنية المعمارية لنظام SQLMR [6].

4. نظام MariaDB Galera [12,13]

للتعرف على عنقود MariaDB Galera في البداية لابد من التعرف على عنقود Galera، إنه عنقود قواعد بيانات متزامن متعدد السادة multi-master، يعتمد في عمله على النسخ المتماثل المتزامن synchronous replication ومحرك تخزين أوراكل MySQL/InnoDB. يتألف عنقود Galera من مخدم قواعد بيانات (MySQL أو MariaDB أو Percona XtraDB) ويستخدم Galera Replication Plugin لإدارة النسخ المتماثل للبيانات. وقد تم إضافة امتداد لـ API MySQL replication plugin لتأمين جميع المعلومات و hooks التي يتطلبها النسخ المتماثل المتزامن المتعدد الرؤساء. يدعى امتداد API بـ Write-Set Replication أو wsrep API، وهو الذي من خلاله يؤمن عنقود Galera النسخ المتماثل المعتمد على المصادقة certification-based replication. تحوي مناقلة النسخ المتماثل (write-sets) على صفوف البيانات التي يتم نسخها بالإضافة إلى جميع معلومات الإقفال التي تجرى من قبل قواعد البيانات خلال المناقلة. يتم تطبيق write-set بعد مصادقة كل العقد على النسخ المتماثل، فإذا لم يكن هناك أي تضارب بإقفال الصفوف تعتبر المناقلة قد تمت عند هذه المرحلة، وبعدها تطبق كل التغييرات على فضاء الجداول tablespace الخاص بها[14]. حيث يضمن النسخ المتماثل المتزامن عدم ضياع البيانات أو حدوث تأخر من قبل العقد المخدمة وذلك عند انهيار عقدة ما.

تم تصميم Galera Cluster استناداً على بيئة السحب السائدة في هذه الأيام وذلك بتأمينه توافرية عالية والقدرة على ضم العقد وتركها بسهولة دون التأثير على أداء العقود أو أي مجهود إداري.

في دراستنا هذه سوف نستخدم مخدم MariaDB وذلك لأنه نظام محسن ومطور صمم خصيصاً ليكون بديلاً توافقياً لمخدمات MySQL، وهو مفتوح المصدر ومتاح للمطورين تحت رخصة GPL V2. ومن الجدير بالذكر أن المطورين الأساسيين الذين عملوا على إنشاء MySQL شاركوا في تطوير MariaDB.

إن نظام MariaDB Galera Cluster يعتبر منظومة عنقودية متزامنة الطابع خاصة بـ MariaDB تتيح وجود أكثر من مخدم رئيسي master في نفس البنية العنقودية، وهي متاحة على نظام linux فقط وتدعم محركات التخزين XtraDB/InnoDB مع وجود نسخ تجريبية تعمل مع محرك التخزين MyISAM. تستخدم هذه المنظومة

مكتبة Galera المطروحة أيضاً في MySQL مع النسخة MariaDB 10.0. يمكن لأي تطبيق متصل مع أي مخدم في البنية العنقودية أن يرسل بياناته إلى هذا المخدم لتقوم Galera Cluster بالنسخ المتماثل للبيانات وتوزيعها على كافة المخدمات في العنقود، وتتميز عملية النسخ المتماثل هذه بأنها عملية نسخ متماثل حقيقية على مستوى صفوف الجداول. يحدث النسخ المتماثل في نظام Galera عند بدء تنفيذ المناقلة وذلك عبر بث إعدادات الكتابة write set الخاصة بالمناقلة عبر كامل البنية العنقودية، بحيث يكون الجهاز الطرفي على اتصال مباشر مع نظام إدارة قاعدة البيانات وظاهرياً يبدو كأنه على منصة MySQL تقليدية .

تتميز منظومة النسخ المتزامن لدى Galera بشفافيتها العالية والقدرة الكبيرة على التدرج مما يسمح لها بتقديم أداء وجهوزية عالية وبحسب التطبيق المستخدم يمكن تحقيق تدرجية قريبة من الخطية.

يعتبر Galera نظاماً عالي التوافرية وليس نظام sharding، بالإضافة إلى عدم قدرته على معالجة الاستعلامات على التوازي الأمر الذي يعتبر مشكلة الأنظمة التقليدية عند ازدياد حجم البيانات. لذا فقد تم تطوير عدد من البرامج الفرعية لكي تدعم عملية تجزئة البيانات ومعالجتها على التوازي، من هذه البرامج Shard-Query الذي صمم لتحسين أداء المعالجة بالإضافة لكونه مدعوما من قبل MariaDB، لذا سنركز في دراستنا هذه عليه .

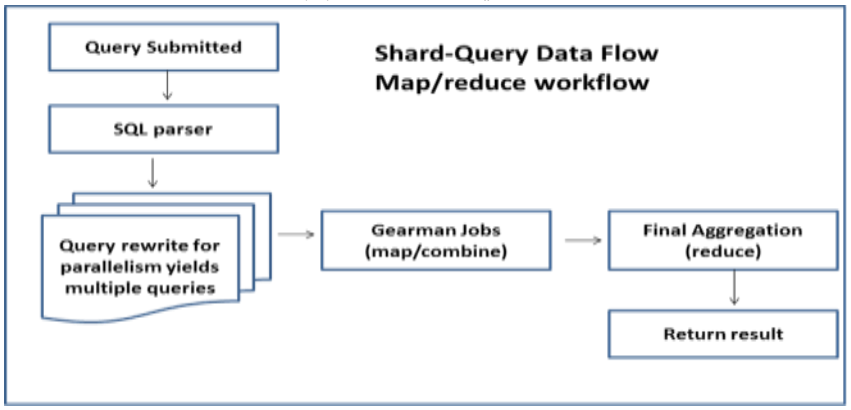
1.4. Shard-Query:

Shard-Query هو محرك استعلام متوازي المعالجة على نطاق واسع لمخدمات MariaDB و MySQL، وتم تنفيذه باستخدام Gearman و PHP. وقد صمم لإجراء الاستعلامات على مجموعات البيانات المجزأة أو sharded[15]. وإن Shard-Query يزيد من التوازي بالاستفادة من ميزات MySQL/MariaDB (التجزئة و shard) وشروط الاستعلام المشترك[16]. يتطلب عقدة مصممة لتكون مخزن التشكيل configuration repository. حيث إن مخزن التشكيل هو مجموعة من جداول MySQL التي تحوي تشكيل ومعلومات الـ Shard-Query.

تخزن البيانات ضمن عقد التخزين التي يجب أن تكون متماثلة المخطط schema. وتحوي عقد التخزين على الجداول ذات الصفة sharded والتي تحوي مفتاح/عمود shard وتوزع ضمن الشبكية grid، بالإضافة إلى الجداول ذات الصفة unshard التي لا تحوي مفتاح shard والتي يتم إجراء النسخ المتماثل عليها عند كل عقدة [15].

يعمل Shard-Query مع البيانات المجمعة باستخدام المخطط النجمي star schema ومع المخططات التي تحوي جداول كبيرة. ويجب عند إنشاء المخطط الأخذ بالاعتبار وجود مفتاح/عمود shard الذي يستخدم لتجزئة البيانات إلى shards مستقلة. حيث إن وجود هذا العمود في صف سيحدد الجدول بأنه sharded. لا يوجد مخدم واحد يحوي نسخة كاملة من الجداول الـ sharded ويستخدم مفتاح shard لتحديد المخدم الذي سيخزن الصف، وإن الجداول التي لها نفس المفتاح تخزن على نفس المخدم. يضيف Shard-Query التوازي للاستعلامات على الجداول المجزأة والـ sharded، حيث تقسم البيانات إلى buckets عن طريق تقسيم المفتاح والتقسيم الهاشي. عند اختيار مفتاح/عمود shard يجب اختيار طريقة للربط إما عن طريق الربط البعثرة أو ربط directory. عند تحميل البيانات أو إدخال صفوف جديدة يقوم Shard-Query بفحص هذه البيانات وإرسالها إلى shardالمناسب ويعمل التحميل بشكل متواز على نطاق واسع، حيث يتم تحميل الملفات المحددة في chunks على التوازي. حيث إن ادخال صفوف جديدة إلى الجداول unshard تذهب إلى كل shards، وإذا كان الاستعلام يحوي على جداول غير sharded يرسل الاستعلام إلى shard واحد فقط لمنع الازدواجية في البيانات[15].

إن مخدم الأعمال Gearman هو الذي يقدم خاصية التوازي، بالإضافة إلى ضمان الاتصال بين الواجهة والعاملين، ويستخدم هيكلية شبيهة بـ map/reduce والتي يبينها الشكل (5).



الشكل (5) تدفق بيانات Shard–Query وطريقة عمل MapReduce like [16].

5. الإعداد للتجربة:

نستخدم في هذه التجربة قاعدة بيانات SysBench كعينة مقارنة ونقوم بمقارنة SQLMR مع أنظمة قواعد البيانات الأخرى، بما فيها Hive وعنقود MariaDB Galera.

إن أداة SysBench هي آلية تركيبية قابلة للتغير تعمل على عدة أنظمة تشغيل متعددة المهام لتقييم معايير الحكم على أداء أنظمة التشغيل والتي تعتبر مهمة لنظام يعمل على تشغيل قاعدة بيانات تفرض عليها أحمال شديدة. نستخدم وحدة OLTP (Online Transaction Processing) الخاصة بـ SysBench. من أجل تعيين نقاط علامة مرجعية تقيس مستوى أداء قاعدة بيانات حقيقية. ويمكن لوحدة OLTP أن تولد مقداراً كبيراً من البيانات المتسلسلة المفهرسة بواسطة رقم العمود column id. كما يمكن لها أن تولد استعلامات تناقلية.

تم في التجربة المقارنة بين الأنظمة المعتبرة Hive(Yarn) و SQLMR وعنقود MariaDB Galera فقط في عمليات القراءة، بما في ذلك عمليات الجمع المطبقة على مجالات البيانات. كما جرت مقارنة النتائج مع الدراسة [6] التي تم فيها مقارنة نظام SQLMR مع Hive باستخدام maprdeduce1 وعنقود Mysql في بيئة تشغيلية مشابهة لبيئة تجربتنا.

تتضمن التجربة قياس مستوى تدرجية البيانات إي إظهار مستوى التدرجية مع ازدياد حجم البيانات وثبات عدد العقد عند خمس عقد، كل عقدة تحتوي أربع نوى معالجة تعملان عند تردد 2.27GHZ ذات ذواكر 4GB وأقراص تخزينية بسعة 200GB وكلها متصلة مع بعضها بواسطة محول Gigabit Ethernet واحد. حيث تم قياس زمن تنفيذ عملية SELECT والمرور على كل نقطة بيانات 10 مرات وحساب زمن التنفيذ الوسطي.

النتائج والمناقشة:

تم دراسة تأثير حجم البيانات على زمن التنفيذ عند إجراء عملية SELECT عن طريق القيام بمجموعة من التجارب لقياس التدرجية مع الأخذ بالحسبان الازدياد الحاصل في أحجام البيانات، مع وجود عدد ثابت من العقد هو 5 عقد في حين تتراوح أحجام البيانات بين 512MB و 128GB.

يظهر الجدول(2) زمن التنفيذ المستغرق عند إجراء عملية SELECT لكل من Hive باستخدام نسخة Yarn وعنقود MariaDB Galera باستخدام Shard-Query و SQLMR على بيانات ذات أحجام مختلفة. ويبين الشكل (9) الرسم البياني لتغير زمن تنفيذ عملية SELECT من أجل كل من الأنظمة المدروسة.

الجدول(2) مقارنة زمن تنفيذ عملية SELECT بين الأنظمة Hive وعنقود MariaDB Galera و SQLMR .

النظام / حجم البيانات	زمن التنفيذ مقاس بالثانية		
	Hive(yarn)	**MariaDB Galera (Shard-query)**	**SQLMR [6]**
512MB	30.67	1.68	32.14
1GB	29.01	3.06	31.64
2GB	29.89	6.12	33.37
4GB	29.15	12.01	33.41
8GB	39.94	23.64	37.29
16GB	45.52	52.97	41.89
32GB	52.25	118.50	48.23
64GB	85.61	252.55	70.39
128GB	142.24	336.3	122.19

كما يظهر في الجدول(3) زمن التنفيذ المستغرق لكل من Hive باستخدام نسخة Mapreduce1 وعنقود MySQL و SQLMR عند إجراء عملية SELECT على بيانات ذات أحجام مختلفة من الدراسة [6].

وإن الاستعلام المنفذ في كلا التجربتين هو كالتالي:

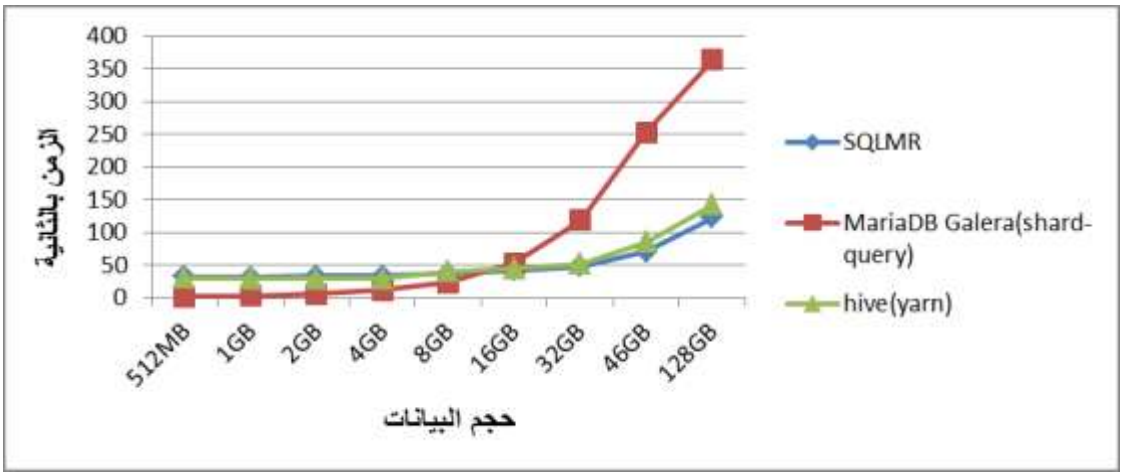
```
SELECT sum(id) FROM table
WHERE id >= max(id)/2 and id <= max(id)
```

من أجل بيانات ذات أحجام صغيرة تم التوصل من الشكل(6) إلى أن عنقود MariaDB Galera باستخدام Shard-Query يتفوق في أدائه على الأنظمة القائمة على MapReduce وذلك عند حجم بيانات أقل من 8GB. في حين ينعكس الوضع عند بيانات ذات أحجام أكثر من 8GB. ويعود السبب في ذلك إلى أن عنقود Galera MariaDB باستخدام Shard-Query سوف يعمل بالتوازي على جلب البيانات من shard واحد وجزء واحد من البيانات ومع ازدياد حجم البيانات سوف يتزايد زمن التنفيذ.

الجدول(3) مقارنة زمن تنفيذ عملية SELECT بين الأنظمة عنقودMySQL وHive وSQLMR [6]

النظام / حجم البيانات	زمن التنفيذ مقاس بالثانية		
	عنقودMySQL	Hive(MP1)	SQLMR
512MB	3.32	34.34	32.14
1GB	6.02	33.27	31.64
2GB	12.11	34.24	33.37
4GB	23.98	34.27	33.41
8GB	47.72	40.08	37.29
16GB	94.72	49.53	41.89
32GB	188.98	58.11	48.23
64GB	378.50	91.50	70.39
128GB		157.43	122.19

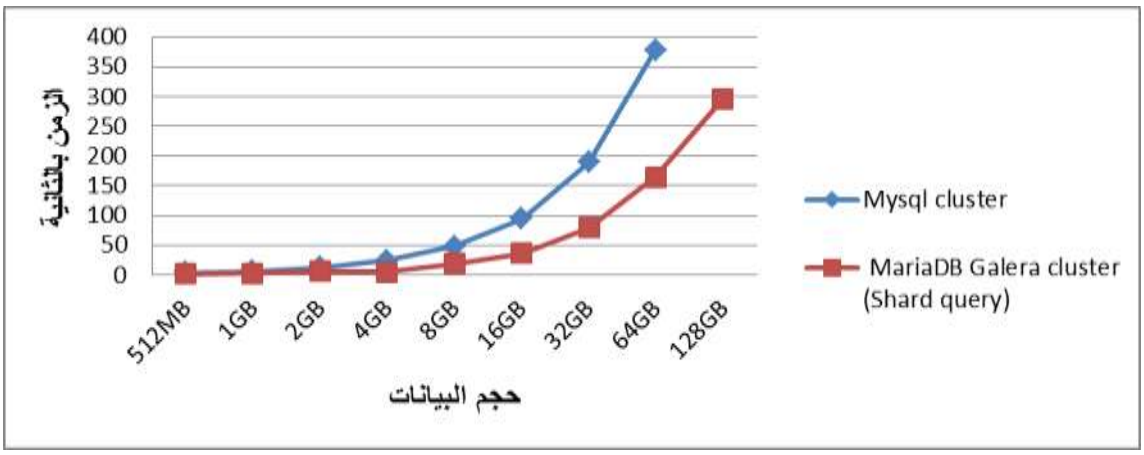
كذلك نلاحظ من الجدول(1) والشكل(6) أن كلاً من SQLMR وHive باستخدام Yarn يتماثلان بالأداء بشكل تقريبي من أجل البيانات ذات الأحجام المختلفة وذلك لأن كلاً منهما يعتمد على استخدام mapreduce التي تجري استخدام التعليمات بالتوازي كما تم شرح العملية سابقا.



الشكل (6) مقارنة زمن تنفيذ عملية SELECT بين SQLMR[6] وعنقودGalera MariaDB وHive على بيانات ذات أحجام مختلفة.

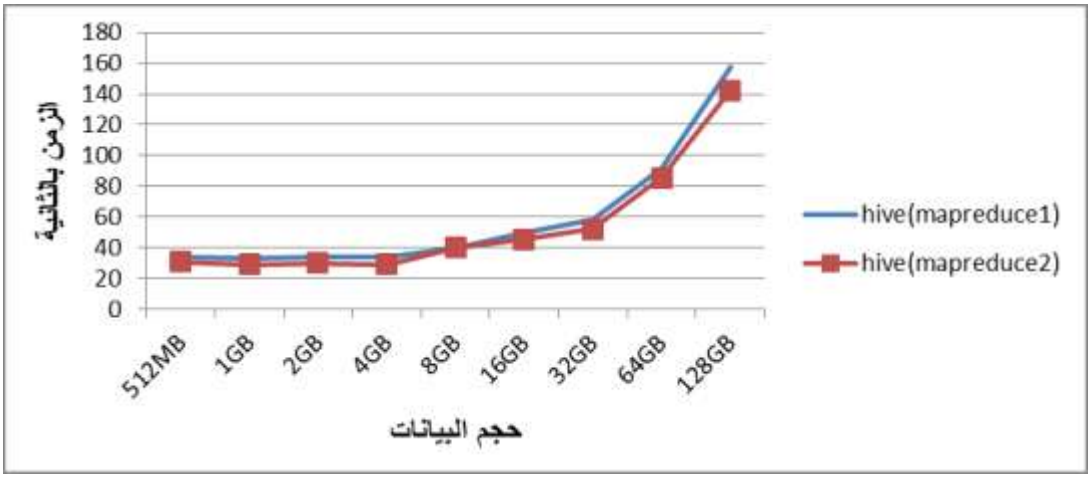
ويبين الشكل(7) مقارنة زمن تنفيذ عملية SELECT في كل من عنقود MySQL وعنقود Galera MariaDB باستخدام Shard-Query من أجل بيانات ذات أحجام مختلفة. ويلاحظ من هذا الشكل أنه من أجل

بيانات ذات حجم صغير يتفوق عنقود MariaDB Galera باستخدام Shard-Query في أدائه على عنقود MySQL، ويعود السبب في ذلك إلى أن MySQL لا تجري معالجات متوازية لاستعلام واحد. يوظف عنقود MySQL تقنيات قواعد البيانات العاملة داخل الذاكرة، والتي تقوم بكتابة كافة البيانات إلى الذاكرة قبل بدء عمل قاعدة البيانات مما يجعلها مقيدة بحجم الذاكرة الفيزيائية. حيث إن نظام عنقود MySQL سينهار بسبب نفاد الذاكرة عندما تصل البيانات إلى أحجام بحدود 64GB. بينما يسرع عنقود MariaDB Galera باستخدام Shard-Query تنفيذ عملية select بما يقارب بين 2 إلى 3 مرة بسبب إجراء عمليات الاستعلام على التوازي.



الشكل (7) مقارنة زمن تنفيذ عملية SELECT بين عنقود MySQl[6] و MariaDB Galera على بيانات ذات أحجام مختلفة.

وتم في الشكل (8) مقارنة زمن تنفيذ عملية SELECT في نظام hive بين نسختي الmapreduce 1 و2، حيث نلاحظ أن التطويرات التي تمت على نسخ ال mapreduce . لم تقدم أي تحسن ملحوظ بالأداء لأنه لم يتم التحسين بطريقة عمل الMapReduce بل كان مقتصراً على تحسين استخدام الموارد وزيادة قابلية التوسع والتوافرية.



الشكل(8) مقارنة زمن تنفيذ عملية SELECT في نظام Hive باستخدام نسختي MapReduce 1 و2 على بيانات ذات أحجام مختلفة.

الاستنتاجات والتوصيات:

1. بينت هذه الدراسة تفوق أداء الأنظمة السحابية المعتمدة على الـ MapReduce. على الأنظمة التقليدية من حيث سرعة تنفيذ عمليات الاستعلام مع تغير حجم البيانات.

2. تقارب أداء كل من SQLMR و hive باستخدام yarn من أجل البيانات ذات الأحجام المختلفة (من 512MB إلى 128GB) وهذا يعني أن الأنظمة السحابية والأنظمة الهجينة من الممكن أن يتقارب أدائها.

3. تفوق نظام Galera MariaDB باستخدام Shard-Query في أدائه على نظام MySQL من أجل البيانات بأحجام مختلفة، وقد سرع تنفيذ عملية SELECT إلى ما يقارب ضعفين أو ثلاث أضعاف بسبب أن نظام Galera MariaDB باستخدامShard-Query يجري الاستعلامات على التوازي.

4. تقارب أداء نسختي Hive وهما MapReduce 1 و 2 عند مقارنتهما وذلك لأن طريقة عملها بقيت نفسها ولكن تم التحسين في طريقة استخدام الموارد وزيادة قابلية التوسع والتوافرية.

5. تكمن أهمية الأنظمة التقليدية في أنها مألوفة للمستخدمين والمطورين وتتمتع ببعض الخواص المهمة للكثير من الأعمال لذا من المهم تطويرها لتواكب الأنظمة السحابية إما عن طريق دمج الخواص السحابية والتقليدية وخلق أنظمة هجينة أو عن طريق إدخال mapreduce إلى آلية عملها والتي بينت التجربة مدى تحسن الأداء عند تنفيذ الأعمال على التوازي.

6. يمكن تحسين أداء الأنظمة السحابية بتطبيق خوارزميات أمثلة للوصول للبيانات وفي طريقة عمل mapreduce ، وهذا ما يمكن إنجازه مستقبلاً.

المراجع:

[1] MELL, P.; GRANCE, T., *The NIST Definition of Cloud Computing*. NIST Special Publication 800-145, September 2011.

[2] SADASHIV,N.; KUMAR, S. M.D., *Cluster, Grid and Cloud Computing: A Detailed Comparison*, The 6th International Conference on Computer Science & Education, SuperStar Virgo, Singapore (ICCSE 2011) August 3-5, 2011.

[3] FURHT, B.; ESCALANTE, A., *Hand Book of Cloud Computing*. Springer, 2010, pp: 3-45.

[4] MOHAMMAD, S., *A Survey and Classification of Data Management Research Approaches in the Cloud*: Master Thesis. Otto von Guericke University Magdeburg, September 11/ 2011.

[5] ARMEND´ARIZ-I˜NIGO, J. E.; RUIZ-FUERTES, M.I., *Transaction Consistency in the Cloud: Old Paradigms Revisited*, Technical Report TR-ITI-SIDI-2010/004, September, 2010.

[6] HSIEH,M.J.; CHANG,C.R.; HO,L.Y.; WU,J.J., LIU,P., *SQLMR : A Scalable Database Management System for Cloud Computing*. International Conference on Parallel Processing (ICPP), IEEE, Taipei City, 13-16, Sept. 2011, 315 – 324.

[7] Apache Hadoop Documentation, <https://hadoop.apache.org>,15 jan.2015

[8] BORTHAKUR, D., *The Hadoop Distributed File System: Architecture and Design*, The Apache Software Foundation 2007, pp:3-14.

[9] WHITE, T., *Hadoop: The Definitive Guide*. SECOND EDITION, O'Reilly Media, Inc, may 2012 ,189-198.

[10] WHITE, T., *Hadoop: The Definitive Guide*. THIRD EDITION, O'Reilly Media, Inc, march 2015, p 686.

[11] Apache hive documentation,
<https://cwiki.apache.org/confluence/display/Hive/Home;jsessionid=E7E4CE667A69CCE005855011E5816C17>, 11/1/2015.

[12] MariaDB documentation, *what is MariaDB cluster*,
< https://mariadb.com/kb/en/mariadb/what-is-mariadb-galera-cluster>, 1/1/2015.

[13] RAZZOLI, F**.,** *Mastering MariaDB*. Packt Publishing Ltd, ISBN 978-1-78398-154-0, September 12, 2014, p 384.

[14] Galera documentation, <http://galeracluster.com/documentation-webpages>. 11/1/2015.

[15] SWANHART, J. *Shard Query Manual*. May 7, 2014. <https://github.com/greenlion/swanhart-tools/wiki/Shard-Query-Manual#Loading>

[16] SWANHART, J., *Shard-Query AN MPPDATABASE FOR THE CLOUD USING THE LAMP STACK*. Jul 30, 2014.