تطوير خوارزمية لمزامنة الملفات الموزعة في التخزين السحابي عند حدوث تعديل في المحتوى بين طرفين

م. طارق عيسى

كلية الهندسة المعلوماتية – جامعة البعث

اشراف الدكتور: أكرم المرعي + ناصر أبو صالح

الملخص:

غالباً يتم تحقيق التخزين السحابي عالي الإتاحية باستخدام أنظمة موزعة معقدة ومتعددة المستويات مبنية على مجموعات من المخدمات ومحركات الأقراص، هناك حاجة إلى تقنيات متطورة للإدارة وموازنة الحمل والاسترداد لتحقيق أداء وإتاحية عاليين وسط وفرة من مصادر الأعطال التي تشمل البرامج والأجهزة والاتصال بالشبكة ومشكلات الطاقة، ولكن تبقى مشكلة تزامن الملفات بين المخدمات أو بين الزبون والمخدم من أعقد المشاكل حيث تعتمد الحلول التقليدية المتبعة حالياً في مختلف الأنظمة الموزعة وأنظمة التخزين السحابي كما في Google drive، Dropbox على عملية رفع أو ارسال كامل الملف عند حدوث تعديل في محتوى الملف سواء كان هذا التعديل كبير أو صغير جداً وبالتالي استهلاك كبير في موارد الشبكة وضغط على عرض الحزمة المتوفر وخصوصاً في حالات الدفع مقابل حجم الارسال، لذلك كان لابد من البحث عن آلية او خوارزمية تعمل على التقليل من حجم البيانات المرسلة على الشبكة.

فقد قمنا في هذا البحث بتطوير آلية جديدة لمزامنة الملفات من خلال تقسيم الملف الى بلوكات والبحث عن مواضع التعديل وتفادي ارسال كامل الملف، والاستعاضة عنها بإرسال أجزاء فقط من الملف وعملنا على تحقيق الخوارزمية باستخدام لغة الجافا JAVA وتقييم عملها بناء على مجموعة من المعايير.

الكلمات المفتاحية:

التخزين السحابي، الأنظمة الموزعة، Google drive، Dropbox، نموذج العمل.

Development of an algorithm to synchronize files distributed in cloud storage when content modification occurs between two sides

Abstract:

High availability cloud storage is often achieved using complex, multi-tiered distributed systems built on clusters of servers and drives. Advanced management, load balancing, and recovery techniques are needed to achieve high performance and availability amid an abundance of fault sources including software, hardware, network connectivity, and power issues, but the The problem of file synchronization between servers or between the client and the server is one of the most complex problems. The traditional solutions currently used in various distributed and cloud storage systems, as in Google drive, Dropbox, depend on the process of uploading or sending the entire file when a modification occurs in the file's content, whether this modification is large or It is very small and thus a large consumption of network resources and

pressure on the available packet width, especially in cases of payment for the size of the transmission, so it was necessary to search for a mechanism or algorithm that works to reduce the volume of data sent on the network.

In this research, we have developed a new mechanism for synchronizing files by dividing the file into blocks, searching for modification locations and avoiding sending the entire file, and replacing it with sending only parts of the file. We worked on achieving the algorithm using JAVA and evaluating its work based on a set of criteria.

Key words:

Cloud storage, Distributed system, Google drive, Dropbox, work template

1 – مقدمة:

يعرف التخزين السحابي الموزع بأنه بنية تحتية مشتتة جغرافياً، تتم من خلال نشر بيانات الأشخاص عبر الشبكة، حيث يسمح بوضع البيانات بالقرب من المستخدم النهائي، وبالتالي يسرع عمليات النقل ويقلل من ازدحام الشبكة وخطر فقدان البيانات. بالإضافة إلى ذلك وعلى عكس نماذج السحابة المركزية، لا تعتمد السحابة الموزعة على مراكز البيانات المركزية.

تعتبر السحابة الموزعة تطبيق لتقنيات الحوسبة السحابية لربط البيانات والتطبيقات المقدمة من مواقع جغرافية متعددة، يعني التوزيع في سياق تكنولوجيا المعلومات (IT) أن شيئاً ما يتم مشاركته بين أنظمة متعددة قد تكون أيضاً في مواقع مختلفة. تعمل السحابة الموزعة على تسريع الاتصالات للخدمات العالمية وتمكين اتصالات أكثر استجابة لمناطق معينة.

يستخدم مزودو السحابة النموذج الموزع لتمكين زمن انتقال أقل وتوفير أداء أفضل للخدمات السحابية. خارج سياق مزود السحابة، هناك مثالان آخران على السحابة الموزعة هما حوسبة الموارد العامة public resource computing وسحابة المتطوعين volunteer cloud.

السحابة الموزعة تعني سحابة أكثر خصوصية (كل ملف مقسم، مشفر من طرف إلى طرف ثم ينتشر عبر الشبكة)، أكثر أماناً (بالاعتماد على أنظمة متعددة فهو أقل عرضة للخطر) وأكثر مراعاة للبيئة.

وتعد مشكلة التزامن من أهم القضايا المدروسة حالياً ففي التخزين السحابي الموزع وهناك حاجة لإيجاد آليات قادرة على مزامنة الملف (الملفات) بين المخدمات أو بين المخدم والزبون، حيث الهدف الرئيسي الأهم هو الحصول على تبادل بيانات أقل في عملية التزامن. لكن نظم التخزين السحابي التقليدية تعمل على تبادل الملفات وإعادة ارسالها بشكل كامل عند كل عملية تعديل.

حيث يتضمن بحثنا أولاً تحديد هدف البحث ومواد وطرائق البحث ومجموعة من المفاهيم النظرية يليها النتائج والمناقشة متضمنة توصيف الخوارزمية المقترحة والمعايير المستخدمة في تقييم عملها وأدائها وفي النهاية مجموعة من النتائج والتوصيات التي حصلنا عليها.

2 – هدف البحث:

يهدف هذا البحث الى تطوير خوارزمية قادرة على مزامنة الملفات بين طرفين والحصول على نسخة متطابقة من الملف (الملفات) بينهما، وذلك تفادياً لعملية إرسال كامل الملف عند كل عملية تعديل في محتوى الملف، بالاستفادة من تقسيم الملف الى بلوكات أصغر والتعامل مع الملف كأنه مجموعة من الملفات الأصغر وتحديد موضع التعديل الحاصل وبالتالي القيام بإرسال فقط الجزء (الأجزاء) الذي تم تعديلها. وبذلك نقلل من حجم استهلاك عرض الحزمة بين الطرفين مع عدد معين من العمليات الحسابية عند الطرف المالك للنسخة الاقدم.

3 – مواد وطرائق البحث:

قمنا في هذا البحث بالاستفادة من لغة الجافا في نمذجة عمل الخوارزمية المقترحة وتطبيق عملها على عدد من الملفات باستخدام برنامج netbeans 12.5 ونسخة الجافا jdk–11.0.13 والقيام بعدد من التجارب ورسم النتائج من خلال برنامج Gnuplot version 5.0 المسؤول عن رسم المخططات البيانية.

4 – المفاهيم النظرية:

4.1 – الحوسبة السحابية مقابل السحابة الموزعة:

في الحقيقة يمكن أن نقول عن الحوسبة السحابية والسحابة الموزعة أنهما نفس المفهوم لكنهم يستخدمون أنظمة مختلفة لتحقيق ذلك، تتطلب الحوسبة السحابية مركز بيانات به العديد من المخدمات للعمل عبر مهام متعددة للمستخدمين، مثل تخزين البيانات ومعالجتها وإدارتها، بينما توزع الحوسبة الموزعة المهام عبر شبكتها على أجهزة حاسوب فردية.

عند المقارنة بين المفهومين، هناك فائدة رئيسية واحدة للسحابة الموزعة: الموثوقية، نظراً لأن السحابة الموزعة تعتمد على العديد من الأنظمة بدلاً من نظام واحد ففي حالة حدوث خلل فني تكون بياناتك أكثر أماناً علاوة على ذلك فهي أسرع حيث يتم تقسيم كل مهمة وتنفيذها بواسطة العديد من أجهزة الحاسوب في وقت واحد.

4.2 – الحوسبة السحابية الموزعة مقابل الحوسبة الطرفية:

الحوسبة الطرفية مثال على السحابة الموزعة حيث تعني الحوسبة الطرفية حرفياً الحوسبة التي تحدث على حافة الشبكة، والتي تحدث على مصادر البيانات تماماً أو بالقرب منها.

تعمل الحوسبة الطرفية على تفريغ البيانات إلى السحابة أثناء فترات الذروة في حركة مرور الحوسبة لضمان السرعة والموثوقية أما السحابة الموزعة بدلاً من ذلك مهمتها الحساب والتخزين و تحقيق الاتصالات في سحابة صغيرة موجودة عبر الشبكة.

4.3 – التخزين السحابي:

يعتبر التخزين السحابي نموذج من نماذج تخزين البيانات، حيث يكون هذا التخزين تخزين فيزيائي على السحابة (مجموعة من المخدمات في موقع أو عدة مواقع بعيدة يتم إدارتها من قبل شركة مستضيفة)، مهمة التخزين السحابي بالنسبة للبيانات هو جعلها متوفرة مع إمكانية وصول إليها، أما بالنسبة للبيئة الفيزيائية فمهمتها أن تكون محمية وقيد التشغيل دائماً[4] .

مفهوم التخزين السحابي يعني أن المستخدم سيقوم بتخزين البيانات الخاصة به على السحابة عوضاً عن تخزينها على النظام المحلي، ويتم الوصول إلى هذه البيانات عن طريق اتصال شبكي بين المستخدم و المخدم (client – server).

تتميز السحابة بالخصائص الأساسية الأربع التالية:

1- المرونة: يتم توفير الخدمة أو التخزين عند الطلب عندما يحتاج المستخدمين إلى مزيد من الموارد، وسيتم توفيرها تلقائياً، من ناحية أُخرى سيتم تخفيض الخدمة إذا لم يكن المستخدم بحاجة إليها.

2- توفير الخدمة الذاتية وإلغاء التوفير التلقائي: يمكن للمستخدمين طلب أي مساحة من التخزين مباشرةً.

3- واجهات برمجة التطبيقات (APIs): تعتبر الواجهات القياسية ميزة لأن التطبيقات ومصادر البيانات يمكن أن تتصل مع بعضها البعض بسهولة.

4- الفواتير بناءً على استخدام الخدمة: تدفع الشركات بقدر ما تستخدم.

4.4 - البنية التحتية للتخزين الفيزيائي:

بداية كان التخزين مرتبط بالمخدمات وأجهزة الحاسوبية العملاقة والمركزية، وتستخدم التطبيقات التي تعمل على نفس الأجهزة هذا التخزين، بعد ذلك ظهرت شبكة منطقة التخزين (Storage Area Network)،وأصبح التخزين منفصلاً ولكنه متصل باستخدام اتصالات شبكية عالية الأداء، ثم تم التركيز على كلاً من البيانات والبنية التحتية الفيزيائية من خلال تقسيم هذه التخزين، مما أدى إلى فوائد عديدة مثل تقليل التكلفة والمرونة.

4.5 – التخزين كخدمة:

يمكن أن يُمنح التخزين على السحابة كخدمة للمستخدمين بدلاً من الاضطرار إلى شراء سعة تخزين فعلية بشكل محلي، زادت شعبية هذه الخدمة لأنها توفر حلاً رخيصاً للنسخ الاحتياطي والتكرار واستعادة البيانات بعد حدوث الكوارث، وبالتالي يتم تأجير التخزين باستخدام سعر عند الطلب من قبل مزودي السحابة مما يقلل من التكاليف التشغيلية.

هناك مجموعة كبيرة ومتنوعة من مزودين التخزين السحابي الذين يوفرون برنامج مستخدم لاستخدام التخزين عبر الشبكة. يسمح بعضها للمستخدمين بتخزين جميع

أنواع البيانات. في حين أن البعض الآخر مخصص فقط لرسائل البريد الإلكتروني أو الصور الرقمية حيث يتحمل المزودين مسؤولية توفير خطة التوفير وصيانة مئات أو آلاف من مخدمات البيانات للتأكد من أن جميع بيانات المستخدمين ستكون متاحة في أي وقت.

يوجد العديد من الفوائد لتخزين البيانات في السحابة على التخزين المحلي [6]:

1- تدفع الشركات فقط مقابل التخزين الذي تستخدمه، أي أن النفقات هي نفقات التشغيل فقط.

2- يمكن الوصول إلى البيانات بسرعة ويمكن الاعتماد عليها توجد البيانات على الويب عبر أنظمة تخزين متعددة بدلاً من موقع محلي.

3- حماية أفضل في حالة وقوع كارثة ما، ففي بعض الأحيان يكون لدى الشركة نسخة احتياطية محلية وفي حالات الحريق أو الكوارث الطبيعية لن يكون النسخ الاحتياطي متاحاً.

4- يوفر مزودو السحابة وفرة في العتاد وتجاوز فشل التخزين التلقائي، يساعد هذا في تجنب انقطاع الخدمة بسبب فشل الأجهزة. يعرف المزودون كيفية توزيع النسخ للتخفيف من أي فشل في الأجهزة.

5- سعات تخزين غير محدودة عملياً، إذا لم يضطر المستخدم إلى استخدام التخزين الإضافي فسوف تنخفض التكاليف.

6- توازن عبء العمل، يساعد مزودو السحابة المستخدمين على تحقيق أفضل أداء من خلال موازنة أعباء العمل.

7- عرض موحد للتخزين، يوفر مزودو السحابة تصديراً للحصول على عرض موحد لاستخدام التخزين.

من ناحية أخرى، هناك العديد من المساوئ لتخزين البيانات باستخدام السحابة على التخزين المحلي:

1- قلة الخبرة: كان على المزودين إعادة كتابة الحلول لحل بعض حالات عدم التوافق مع تخزين البيانات عبر الإنترنت، وقد شكّل ذلك صعوبة للشركات.

2- السعر والموثوقية: يجب على المستخدم حساب فعالية التكلفة للسحابة مقابل استضافة بياناته وصيانتها.

3- الأمان: هناك احتمال أن تتم سرقة البيانات أو عرضها من قبل أشخاص غير مصرح لهم.

4- حدود النطاق الترددي (عرض الحزمة): إذا لم يكن النطاق الترددي بالسرعة التي يحتاجها المستخدم، فلن يكون الحل مناسباً. حيث أن النطاق الترددي هو مقياس لمقدار البيانات التي يمكن نقلها من مكان إلى آخر في فترة زمنية معينة.

5- مسافة الشبكة (التأخير): يؤثر مجموع التأخيرات الزمنية في الانتشار ونقل الحزم داخل الشبكة على عملية التخزين السحابي.

4.6 - النموذج المرجعي للتخزين السحابي:

من المهم أن تدعم أي واجهة للتخزين السحابي مجموعة من السمات والتي سبق وأن تحدثنا عنها: الدفع حسب الاستخدام، المرونة وبساطة الاستخدام والإدارة، مع السماح بالعديد من حالات الأعمال والعروض لفترة طويلة في المستقبل.

يُظهر النموذج الذي تم إنشاؤه ونشره بواسطة Storage Networking Industry Association (SNIA) أنواعاً متعددة من واجهات تخزين البيانات السحابية قادرة على دعم التطبيقات القديمة والجديدة، تسمح جميع الواجهات بتوفير التخزين عند الطلب والمستمدة من مجموعة من الموارد، يتم سحب السعة من مجموعة من السعات التخزينية التي توفرها خدمات التخزين. يتم تطبيق خدمات

البيانات على عناصر البيانات الفردية على النحو الذي يحدده الوصف للنظام. يحدد هذا الوصف متطلبات البيانات على أساس عناصر البيانات الفردية أو مجموعات البيانات (الحاويات containers).

واجهة إدارة البيانات السحابية (Cloud Data Management Interface CDMI) هي الواجهة الوظيفية التي ستستخدمها التطبيقات لإنشاء عناصر البيانات واستردادها وتحديثها وحذفها من السحابة، كجزء من هذه الواجهة سيتمكن المستخدم من اكتشاف إمكانات عرض التخزين السحابي واستخدام هذه الواجهة لإدارة المجموعات والبيانات الموضوعة فيها، بالإضافة إلى ذلك يمكن تعيين البيانات الوصفية على الحاويات وعناصر البيانات الموجودة بها من خلال هذه الواجهة. من المتوقع أن تكون الواجهة قادرة على التنفيذ بواسطة غالبية عروض التخزين السحابي الحالية اليوم، يمكن القيام بذلك باستخدام ملائم adapter لواجهة الملكية الموجودة أو عن طريق تنفيذ الواجهة مباشرة، بالإضافة إلى ذلك يمكن تكييف مكتبات المستخدمين الحالية مثل طريقة الوصول القابلة للتمديد (XAM Extendable Access Methods) مع هذه الواجهة.

تُستخدم CDMIبواسطة التطبيقات الإدارية والتنظيمية لإدارة الحاويات والحسابات والوصول إلى الأمان ومعلومات المراقبة / الفواتير وحتى للتخزين الذي يمكن الوصول إليه بواسطة البروتوكولات الأخرى. يتم الكشف عن إمكانات خدمات التخزين والبيانات الأساسية حتى يتمكن المستخدمين من فهم العرض، قد تقدم عروض السحابة المتوافقة مجموعة فرعية من أي واجهة طالما أنها تكشف القيود في جزء القدرات من الواجهة.

4.7 – واجهة برمجة تطبيقات التخزين السحابي (API):

تعد واجهة برمجة تطبيقات التخزين السحابي (API) طريقة للوصول إلى نظام التخزين السحابي واستخدامه، أكثر هذه الأنواع شيوعاً هي REST (نقل الحالة التمثيلية REpresentational State Transfer) على الرغم من وجود أنواع أخرى تستند إلى SOAP (بروتوكول الوصول إلى الأغراض البسيط Simple Object Access Protocol)، ترتبط كل واجهات برمجة التطبيقات هذه بإنشاء طلبات الخدمة عبر الإنترنت REST هو مفهوم معترف به على نطاق واسع كمنهج لتصميم API القابل للتطوير "عالي الجودة".

من أهم ميزات REST أنها بنية "عديمة الحالة stateless"، هذا يعني أن كل ما يلزم لإكمال الطلب إلى سحابة التخزين مضمن في الطلب بحيث لا يلزم عقد جلسة بين مقدم الطلب وسحابة التخزين، تعتبر REST مهمة للغاية لأن حالة الشبكة لها وقت استجابة غير متوقع والاتصال ليس سريعاً بشكل عام عند مقارنته بشبكة المنطقة المحلية LAN.

REST هو منهج له صلة كبيرة بالطريقة التي يعمل بها الإنترنت. لا تعمل طرق الوصول إلى تخزين الملفات التقليدية التي تستخدم NFS (نظام ملفات الشبكة) أو CIFS (نظام ملفات الإنترنت العام) عبر الإنترنت، بسبب زمن الوصول.

التخزين السحابي مخصص للملفات، والتي يشير إليها البعض كأغراض والبعض الآخر يسميها البيانات غير المنظمة أو غير المهيكلة، مثلاً الملفات المخزنة على الحاسب الشخصي، مثل الصور وجداول البيانات والمستندات، هذه لها تنوع غير عادي وبالتالي فهي غير منظمة.

النوع الآخر من البيانات هو البيانات المنظمة أو الكتلة، على سبيل المثال بيانات قاعدة البيانات، وهي البيانات التي تغذي نظام المعاملات التي تتطلب أداءً معيناً مضموناً أو منخفض التأخير. التخزين السحابي ليس لحالة الاستخدام هذه. يقدّر

مركز التصميم الصناعي (IDC) أن ما يقارب من 70٪ من بيانات الجهاز المخزنة في العالم غير منظمة، وهذا أيضاً هو نوع البيانات الأسرع نمواً. لذا فإن التخزين السحابي هو تخزين للملفات التي يمكن الوصول إليها بسهولة عبر الإنترنت. هذا لا يعني أنه لا يمكنك الوصول إلى التخزين السحابي على شبكة خاصة أو شبكة محلية LAN، والتي قد توفر أيضاً الوصول إلى سحابة التخزين من خلال طرق أخرى، مثل NFS أو CIFS، هذا يعني أن الوصول الأساسي والمفضل يكون بواسطة واجهة برمجة تطبيقات REST.

تعد واجهات برمجة تطبيقات REST لغة محايدة وبالتالي يمكن للمطورين الاستفادة منها بسهولة بالغة باستخدام أي لغة تطوير يختارونها.

قد يتم العمل على الموارد داخل النظام من خلال عنوان URL، لذلك فإن واجهة برمجة التطبيقات ليست "لغة برمجة"، ولكنها طريقة استخدام لغة البرمجة للوصول إلى سحابة التخزين.

تتعلق واجهات برمجة تطبيقات REST أيضاً بتغيير حالة المزود من خلال تمثيل تلك الموارد، ولا يتعلق الأمر باستدعاء أساليب خدمة الويب بالمعنى الوظيفي،

حيث تتمثل الاختلافات الرئيسية بين واجهات برمجة تطبيقات التخزين السحابي المختلفة في عناوين URL التي تحدد الموارد وتنسيق التمثيلات.

أمثلة واجهات برمجة تطبيقات Amazon S3، Eucalyptus، Rackspace Cloud Files APIs، Mezeo APIs، Nivanix APIs، Simple Cloud API.

4.8 – الأنظمة الموزعة:

يتم تعريف النظام الموزع على أنه مجموعة من أجهزة الحواسيب المستقلة التي تنظر إلى مستخدميها على أنها نظام واحد متماسك [2]. ويشمل هذا التعريف العديد من الجوانب أهمها:

الجانب الأول هو أن النظام الموزع يحتوي على مكونات مستقلة، والمكونات هنا ليست سوى أنظمة الحاسوب. الجانب الثاني هو أن المستخدمين يعتقدون أنهم يديرون بنظام واحد، هذا يعني أنه بطريقة أو بأخرى، تحتاج أجهزة الحواسيب المستقلة إلى التعاون.

يمكننا تعريف المكونات الأساسية في النظام الموزع على الشكل التالي:

- البرنامج: كود نقوم بكتابته.
- العملية: على ماذا نحصل عندما نقوم بعملية التشغيل.
- الرسالة: تستخدم من أجل عملية الاتصال بين العمليات.
- الحزمة: جزء من الرسالة التي يمكن أن تنتقل.
- البروتوكول: هو وصف رسمي لتنسيقيات الرسائل والقواعد التي يجب على عمليتين اتباعها من أجل تبادل تلك الرسائل.
- الشبكة: هي البنية التحتية التي تربط أجهزة الحاسوب ومحطات العمل والمحطات الطرفية والمخدمات وما إلى ذلك و تتكون من أجهزة توجيه متصلة بواسطة روابط اتصال.
- العنصر: يمكن أن تكون عملية أو أي جزء من الأجهزة المطلوبة لتشغيل عملية أو دعم الاتصالات بين العمليات وتخزين البيانات، و غيرها.
- النظام الموزع: هو تطبيق ينفذ مجموعة من البروتوكولات لتنسيق الإجراءات من عمليات متعددة على الشبكة، بحيث تتعاون جميع المكونات معاً لأداء مجموعة واحدة أو صغيرة من المهام ذات الصلة

4.8.1 - ميزات النظام الموزع:

1- متسامح مع الخطأ: يبقى النظام قيد التشغيل حتى لو فشلت إحدى مكوناته.

2- متوفر: يمكنه استعادة العمليات، والسماح لها باستئناف تقديم الخدمات حتى في حالة فشل بعض المكونات.

3- قابل للاسترداد: يمكن للمكونات الفاشلة إعادة تشغيل نفسها والانضمام إلى النظام، بعد إصلاح سبب الفشل.

4- الاتساق: يمكن للنظام تنسيق الإجراءات بواسطة مكونات متعددة غالباً في وجود التزامن والفشل. هذا يكمن وراء قدرة النظام الموزع على التصرف كنظام غير موزع.

5- قابل للتطوير: يمكن أن يعمل بشكل صحيح حتى مع تغيير حجم بعض جوانب النظام إلى حجم أكبر. على سبيل المثال، قد نقوم بزيادة حجم الشبكة التي يعمل عليها النظام. يؤدي هذا إلى زيادة تواتر انقطاع الشبكة ويمكن أن يؤدي إلى تدهور النظام "غير القابل للتطوير". وبالمثل، قد نقوم بزيادة عدد المستخدمين أو

المخدمات، أو التحميل الكلي على النظام. في نظام قابل للتطوير، لا ينبغي أن يكون لهذا تأثير كبير.

6- الأداء المتوقع: القدرة على توفير الاستجابة المرغوبة في الوقت المناسب.

7- آمن: يصادق النظام على الوصول إلى البيانات والخدمات.

4.8.2 - أهداف النظام الموزع:

ليس الأمر أن الأنظمة الموزعة مبنية فقط لأن الناس لديهم إمكانية بنائها. يحتوي كل نظام موزع على عدد من الأهداف التي يجب تحقيقها أثناء عملية البناء الخاصة منها:

1- جعل الموارد متاحة: من هذه الموارد (الشبكات – صفحات الانترنت – الملفات – البيانات – مرافق التخزين – أجهزة الكمبيوتر – طابعات).

2- الشفافية: ينطبق مفهوم الشفافية على العديد من جوانب النظام الموزع، منها (الفشل – التزامن – التمكن من الوصول – الموقع – التكرار – النقل).

3- الانفتاح: فصل استراتيجية العمل عن الآلية.

4- قابلية التوسع: (مشاكل في قابلية التوسع – تقنيات التحجيم).

5- الأخطاء: تشرح الأخطاء التي ارتكبت أثناء تقديم النظام الموزع للمرة الأولى. تمت صياغة هذه الأخطاء على أنها افتراضات خاطئة وهي كما يلي: (لديها شبكة موثوقة – الشبكة مؤمنة بالكامل – الشبكة متجانسة- لديها زمن انتقال صفري – يوجد مسؤول واحد فقط – تكلفة النقل صفر – لديها عرض النطاق الترددي اللانهائي – الهيكل لا يتغير).

4.8.3 – أنواع الأنظمة الموزعة:

1- نظام الحوسبة الموزعة Distributed computing systems

نوع مهم من الأنظمة الموزعة حيث أنه يُستخدم لمهام الحوسبة عالية الأداء، بشكل تقريبي يمكن للمرء أن يميز بين مجموعتين فرعيتين:

a. أنظمة الحوسبة العنقودية

b. أنظمة الحوسبة الشبكية

في الحوسبة العنقودية تتكون الأجهزة الأساسية من مجموعة من محطات العمل أو أجهزة الحاسوب المماثلة المتصلة بشكل وثيق عن طريق شبكة محلية عالية السرعة، بالإضافة إلى ذلك تعمل كل عقدة على نفس نظام التشغيل.

يصبح الوضع مختلفاً جداً في حالة الحوسبة الشبكية، تتكون هذه المجموعة الفرعية من أنظمة موزعة يتم إنشاؤها غالباً على شكل اتحاد لأنظمة الحواسيب حيث قد يقع كل نظام ضمن مجال إداري مختلف، وقد يكون مختلفاً تماماً عندما يتعلق الأمر بالأجهزة والبرامج وتكنولوجيا الشبكة.

2- نظم المعلومات الموزعة Distributed information systems

a. أنظمة معالجة التحويلات

b. تكامل تطبيق المؤسسة

فئة مهمة أخرى من الأنظمة الموزعة في المؤسسات التي واجهت ثروة من التطبيقات المتصلة بالشبكة، ولكن تبين أن قابلية التشغيل كانت تجربة سيئة، العديد من حلول البرامج الوسيطة الحالية هي نتيجة العمل مع بنية تحتية كان من الأسهل فيها دمج التطبيقات في نظام معلومات على مستوى المؤسسة.

يتكون التطبيق المرتبط بالشبكة ببساطة من مخدم يقوم بتشغيل هذا التطبيق (غالباً ما يتضمن قاعدة بيانات) وإتاحته للبرامج البعيدة، التي تسمى العملاء أو المستخدمين، يرسل هؤلاء العملاء طلباً إلى المخدم لتنفيذ عملية محددة وبعد ذلك يتم إرسال استجابة مرة أخرى. من أجل العمل عند أدنى مستوى للعملاء يتم تغليف عدد من الطلبات، ربما لمخدمات مختلفة في طلب واحد أكبر وتنفيذها كمعاملة موزعة و هذا ما يسمى بالتكامل. الفكرة الأساسية هي أنه يتم تنفيذ جميع الطلبات أو عدم تنفيذها. نظراً لأن التطبيقات أصبحت أكثر تعقيداً وتم فصلها تدريجياً إلى مكونات مستقلة (لا سيما التمييز بين مكونات قاعدة البيانات ومكونات المعالجة)، أصبح من الواضح أن التكامل يجب أن يتم أيضاً عن طريق السماح للتطبيقات بالاتصال ببعضها البعض مباشرة. وقد أدى هذا الآن إلى صناعة ضخمة تركز على تكامل تطبيقات المؤسسة (EAI).

3- الأنظمة المنتشرة الموزعة Distributed pervasive systems

a. أنظمة منزلية

b. أنظمة الرعاية الصحية الإلكترونية

c. شبكات الاستشعار

تتميز الأنظمة الموزعة التي تمت مناقشتها حتى الآن إلى حد كبير باستقرارها حيث العقد ثابتة ولديها اتصال دائم وعالي الجودة إلى حد ما بالشبكة، نوعاً ما، يتحقق هذا الاستقرار نوعاً ما من خلال التقنيات المختلفة لتحقيق شفافية التوزيع، على سبيل المثال هناك العديد من الطرق التي يمكننا من خلالها خلق الوهم بأن المكونات في بعض الأحيان فقط قد تفشل، وبالمثل هناك جميع أنواع الوسائل لإخفاء موقع الشبكة الفعلي للعقدة، مما يسمح للمستخدمين والتطبيقات بشكل فعال بالاعتقاد بأن العقد تظل في مكانها، ومع ذلك فقد تغيرت الأمور منذ إدخال أجهزة الحوسبة المحمولة والمدمجة، مما أدى إلى ما يشار إليه عموماً بالأنظمة المنتشرة، كما يوحي اسمها تهدف الأنظمة المنتشرة إلى الاندماج بشكل طبيعي في بيئتنا، هي أيضا أنظمة موزعة بشكل طبيعي، و ما يجعلها فريدة بالمقارنة مع أنظمة الحوسبة والمعلومات الموصوفة حتى الآن هو أن الفصل بين المستخدمين ومكونات النظام أكثر غموضاً، غالباً لا توجد واجهة واحدة مخصصة مثل مجموعة الشاشة / لوحة المفاتيح، بدلاً من ذلك غالباً ما يكون النظام واسع الانتشار مزوداً بالعديد من المستشعرات التي تلتقط جوانب مختلفة من سلوك المستخدم، وبالمثل قد يكون لديها عدد لا يحصى من المشغلات لتقديم المعلومات والتعليقات، وغالباً ما تهدف بشكل مقصود إلى توجيه السلوك.

تتميز العديد من الأجهزة في الأنظمة المنتشرة بأنها صغيرة ومزودة بالبطارية ومتنقلة ولا تحتوي إلا على اتصال لاسلكي، على الرغم من عدم تطبيق كل هذه الخصائص على جميع الأجهزة، هذه ليست بالضرورة خصائص مقيدة كما هو الحال في الهواتف الذكية، ومع ذلك فإن حقيقة أننا غالباً ما نحتاج إلى التعامل مع تعقيدات الاتصالات اللاسلكية والمتنقلة تتطلب حلولاً خاصة لجعل النظام المنتشر شفافاً أو غير مزعج قدر الإمكان.

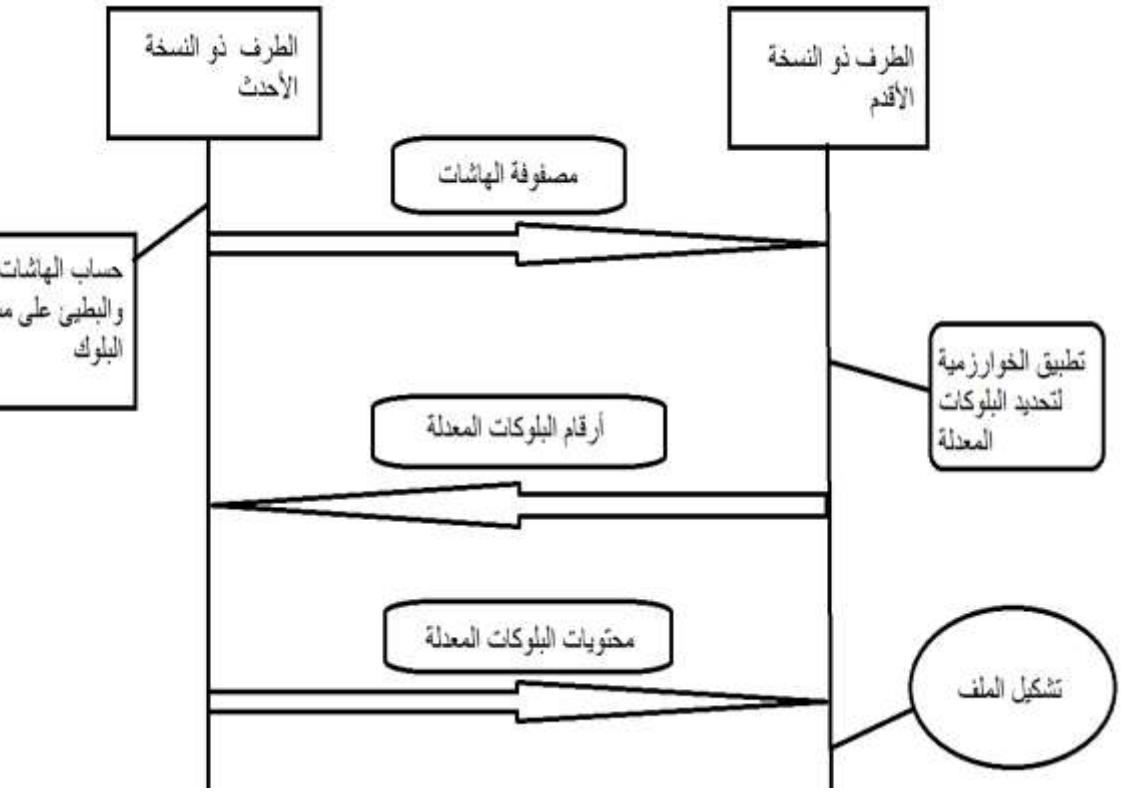
5 – النتائج والمناقشة:

5.1 – مقدمة:

قمنا في هذا البحث بالعمل على تطوير خوارزمية تعمل على مزامنة (تحديث) ملف أو مجموعة من الملفات بين طرفين حيث أحد الطرفين يحوي نسخة أحدث من النسخة المتوفرة عند الطرف الأخر و نعتمد على تاريخ التعديل في تحديد النسخة الأحدث لتبدئ الخوارزمية في التنفيذ ونحصل في النهاية على نسخة متطابقة من الملف عند الطرفين بعد تبادل عدد معين من الرسائل التي تدل على مواقع التعديل وتبادل محتوى البلوكات المعدلة فقط.

5.2 – توصيف الخوارزمية:

يمكننا توصيف الخوارزمية المقترحة بالشكل التالي:



وتتلخص خطوات الخوارزمية المقترحة بالبنود التالية:

- ✓ يقوم أحد الطرفين بإرسال تاريخ أخر تعديل للملف الى الطرف الأخر ليقوم بمقارنة التاريخين وتحديد الطرف المالك للنسخة الأحدث.
- ✓ يقوم الطرف المالك للنسخة الأحدث بحساب الهاشات (hashes) على مستوى البلوك وتشكيل مصفوفتين:

- مصفوفة الهاشات السريعة حجمها n هاش بحيث قمنا باختيار هاش سريع الحساب وهو أخذ أول ثلاث بايتات من بداية البلوك وآخر ثلاث بايتات من نهاية البلوك لتشكيل هاش سريع الحساب (كثير التطابق) بحجم 6 بايت.
- مصفوفة الهاشات البطيئة بحجم n هاش وبذلك اعتمدنا على خوارزمية MD5 المعروفة لحسابه حيث يتميز بصعوبة في الحساب ولكن صعب جداً وجود تطابق بين بلوكين في حال اختلاف البيانات بينهما.

✓ يقوم بعدها بإرسال المصفوفتين الى الطرف الأخر ذو النسخة الأقدم.

✓ يقوم الطرف ذو النسخة الأقدم وهو مسؤول عن تطبيق آلية للبحث عن البلوكات وتحديد البلوكات المعدلة وموضعها وذلك من خلال توليد مصفوفة هباشات سريعة على مستوى البايت ومن اجل كل هاش سريع تم استقباله يعمل على البحث عنه ويكون امام حالتين:

- حالة وجود تطابق مع أحد الهاشات السريعة لديه ليقوم بتوليد الهاش البطيء ومقارنته مع الهاش البطيء المستقبل وبالتالي أيضا يكون هنا أمام خيارين:

- ❖ في حال تطابق الهاشات البطيئة معاً: يتأكد أن محتويات البلوك موجودة لديه.
- ❖ في حال عدم التطابق بين الهاشات البطيئة: يقوم بإكمال البحث عن هاش سريع اخر متطابق مع الهاش السريع المستقبل.
- ➢ في حال عدم حدوث تطابق ابدا مع أية هاش سريع يعلم عندها أن البلوك تم تعدله ويسجل رقمه ليتم طلبه لاحقاً من الطرف ذو النسخة الاحدث.

- ✓ بعد العملية السابقة نكون ق حصلنا على أرقام البلوكات المعدلة وأرقام البلوكات غير المعدلة.
- ✓ يقوم عندها بإرسال أرقام البلوكات المعدلة الى الطرف ذو النسخة الأحدث طالباً منه محتويات البلوكات.
- ✓ يتم ارسال محتويات البلوكات المعدلة من الطرف ذو النسخة الأحدث الى الطرف الأخر.
- ✓ في النهاية: يعمل الطرف ذو النسخة الأحدث بإعادة تشكيل الملف من البلوكات الموجودة لديه إضافة الى البلوكات المستقبلة لنحصل أخيرا على

نسخة متطابقة من الملف عند الطرفين (تمت عملية المزامنة بنجاح دون الحاجة لإرسال كامل محتوى الملف والاستعاضة عنها بإرسال أجزاء محددة معدلة من الملف).

5.3 – معايير تقييم أداء عمل الخوارزمية:

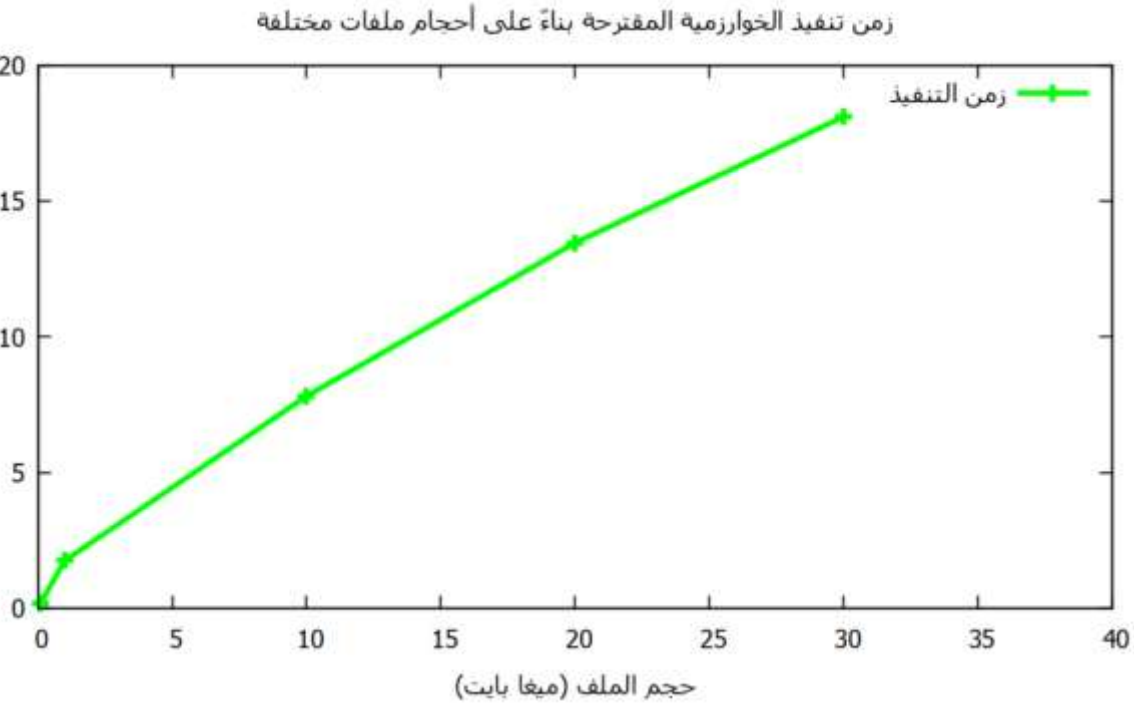
تم تقييم عمل وأداء الخوارزمية المقترحة بناءً على عدد من المتغيرات:

- ✓ استخدام احجام ملفات متنوعة (100KB–1MB–10MB…).
- ✓ استخدام تعديلات متنوعة على محتوى الملف (عشوائي – مواضع محددة).
- ✓ تعديل حجم البلوك ودراسة تأثيره على عمل الخوارزمية (1KB–10KB–100KB–1MB…).

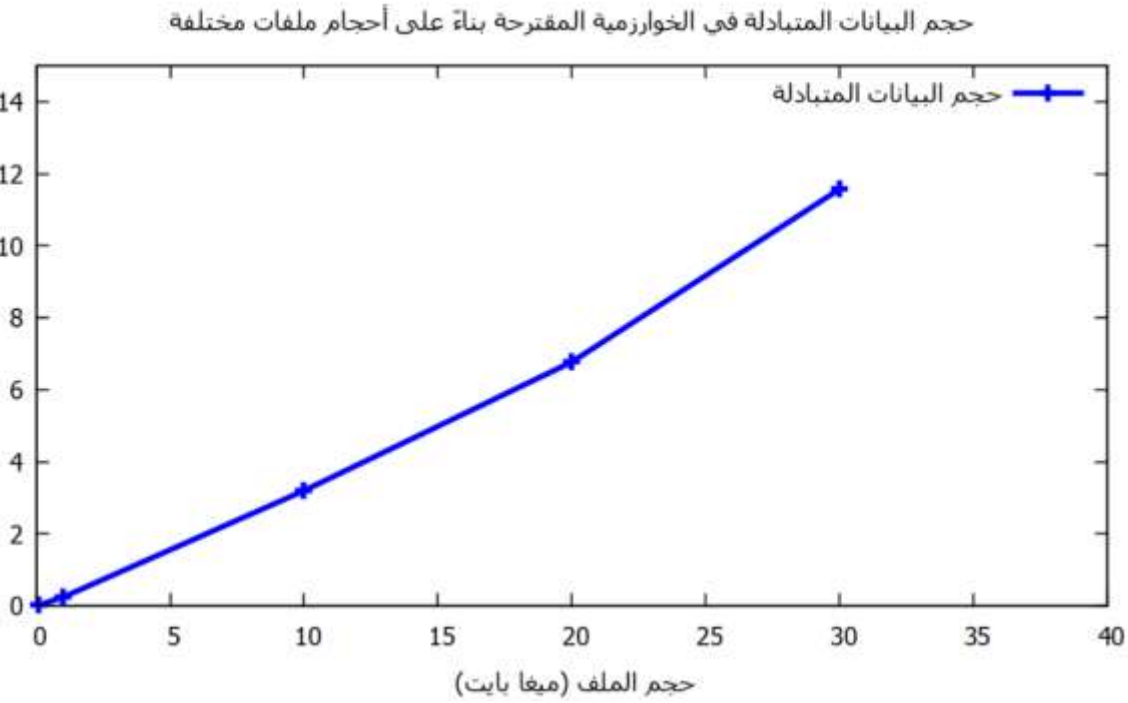
5.4 – النتائج العملية:

5.4.1 – تقييم عمل الخوارزمية بناءً على أحجام ملفات متنوعة:

تم بتثبيت حجم البلوك وحجم التعديلات والعمل على استخدام أحجام ملفات مختلفة ومقارنة زمن تنفيذ الخوارزمية المقترحة كما في الشكل (1) لنلاحظ أن زمن تنفيذها يزداد مع ازدياد حجم الملف المراد مزامنته ولكن يبقى حجم البيانات المتبادلة على الشبكة اقل بشكل ملحوظ مقارنة مع ارسال كامل الملف كما في الشكل (2).



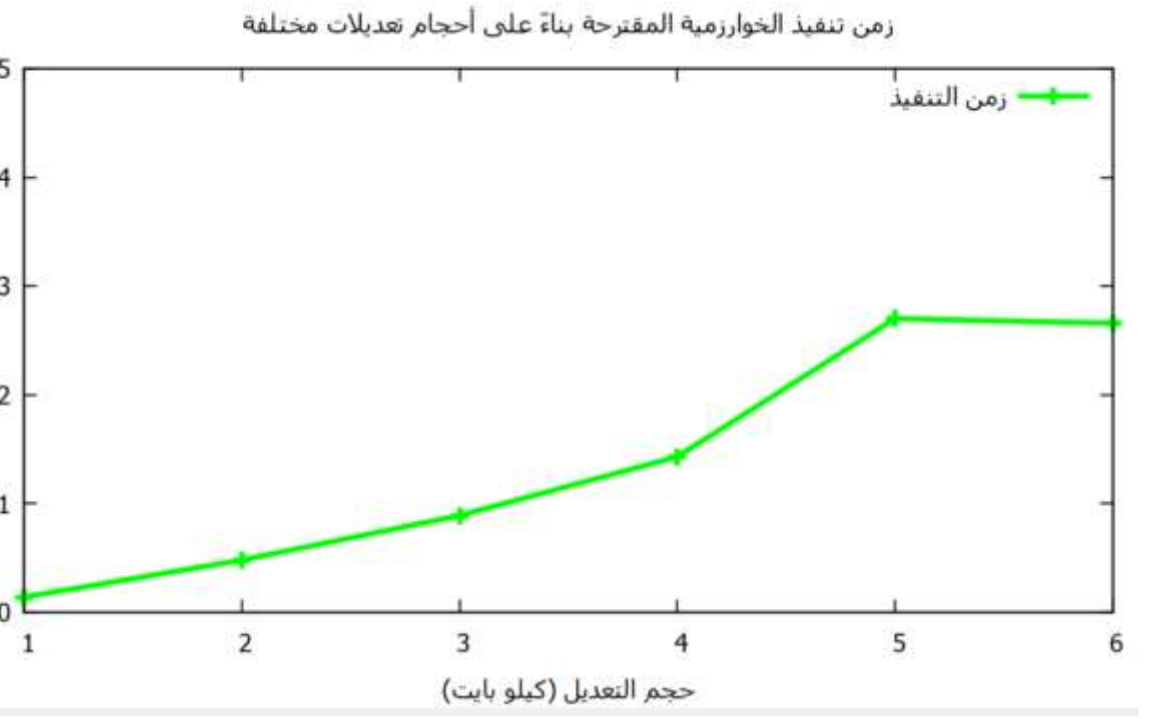
الشكل (1) زمن تنفيذ الخوارزمية المقترحة بناءً على أحجام ملفات مختلفة



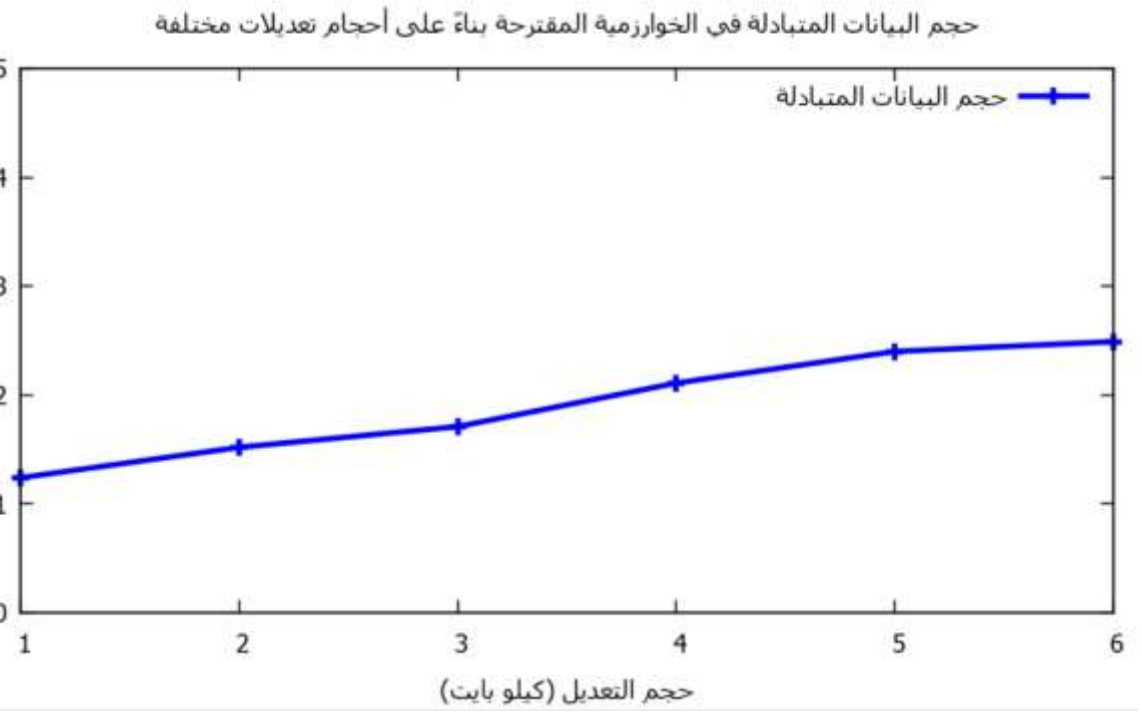
الشكل (2) حجم البيانات المتبادلة في الخوارزمية المقترحة بناءً على أحجام ملفات مختلفة

5.4.2 – تقييم عمل الخوارزمية المقترحة بناءً على تعديلات مختلفة في محتوى الملف :

تم الحفاظ على حجم الملف وحجم البلوك ثابت وتعديل حجم التعديلات (الفروق) الحاصلة في الملف ودراسة اداء الخوارزمية حيث نلاحظ من الشكل (3) زيادة في زمن تنفيذ الخوارزمية مع زيادة حجم التعديلات في محتوى الملف بسبب الحاجة الى مزيد من عمليات توليد الهاش الصعب (hard) ولكن تبقى الخوارزمية تقدم أداء أفضل من الآليات التقليدية المعتمدة على إرسال كامل الملف كما في الشكل (4).



الشكل (3) زمن تنفيذ الخوارزمية المقترحة بناءً على أحجام تعديلات مختلفة

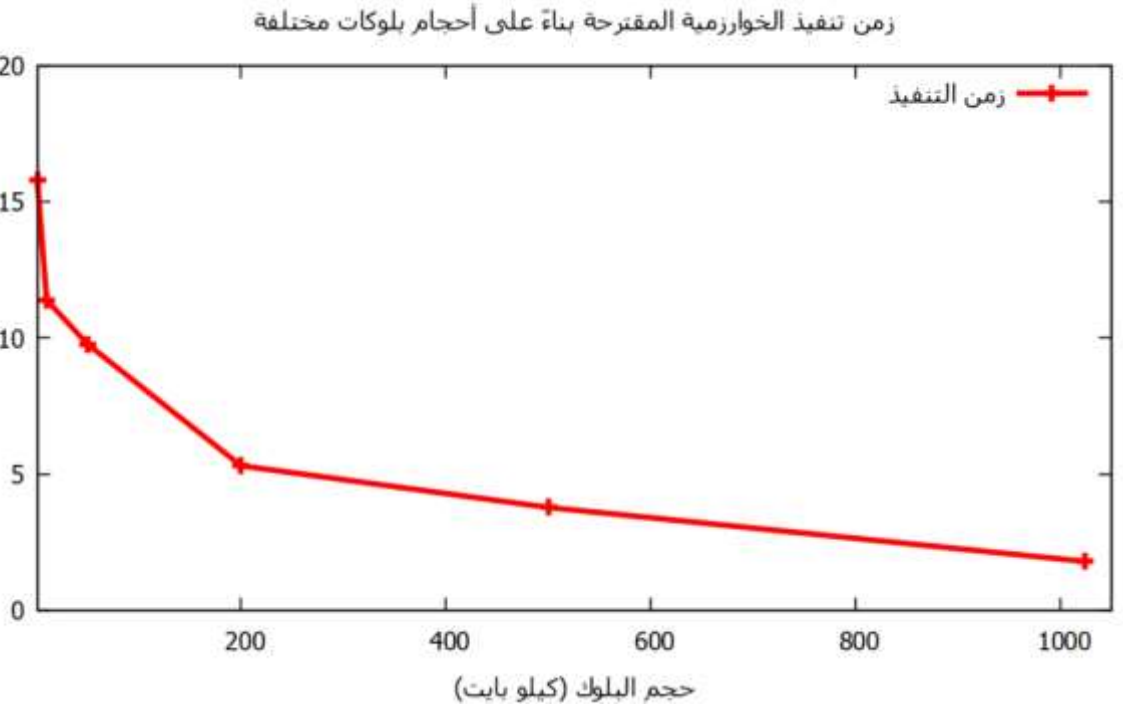


الشكل (4) حجم البيانات المتبادلة في الخوارزمية المقترحة بناءً على أحجام تعديلات مختلفة

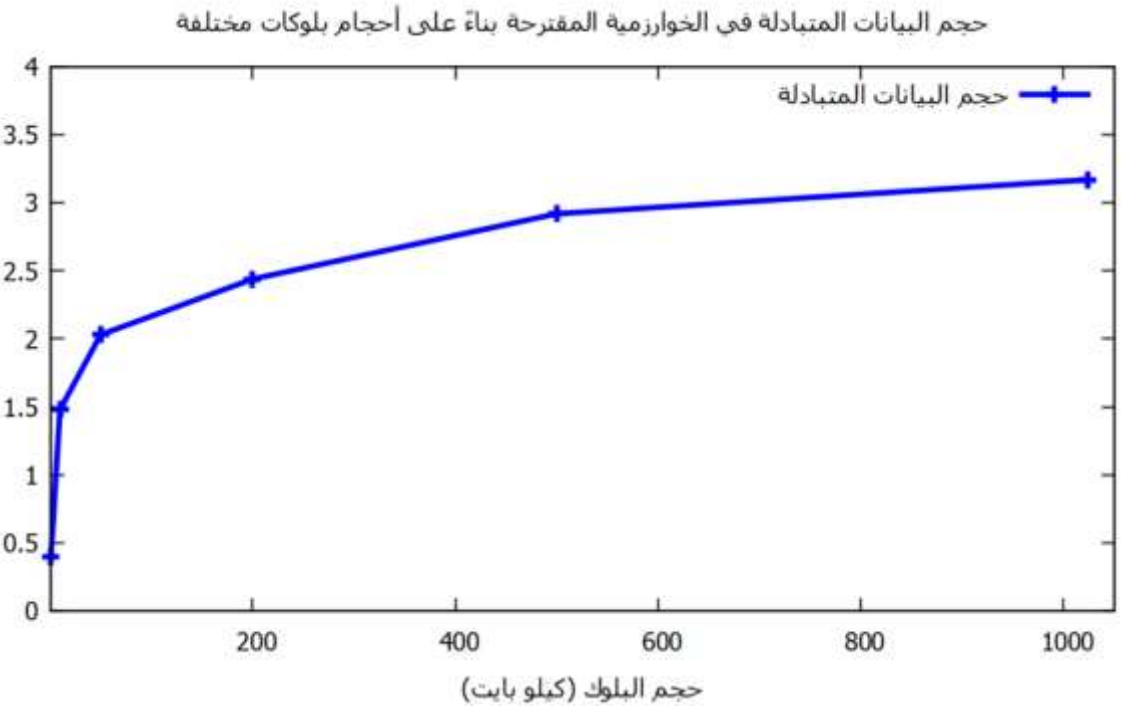
5.4.3 – تقييم عمل الخوارزمية المقترحة بناءً على أحجام بلوكات متنوعة:

تم تثبيت حجم الملف وحجم التعديلات في محتواه والعمل على استخدام أحجام بلوكات مختلفة لنلاحظ من الشكل (5) أن زمن تنفيذ الخوارزمية المقترحة ينقص مع زيادة حجم البلوك الذي بدوره يقلل من عمليات توليد الهاشات وتقليل في زمن

البحث عن تموضع البلوكات ولكن هذا النقصان في حجم البلوك يؤدي الى زيادة في حجم البيانات المتبادلة بشكل ملحوظ كون الحاجة الى ارسال بلوكات بأحجام أكبر عبر الشبكة كما في الشكل (6).



الشكل (5) زمن تنفيذ الخوارزمية المقترحة بناءً على أحجام بلوكات مختلفة

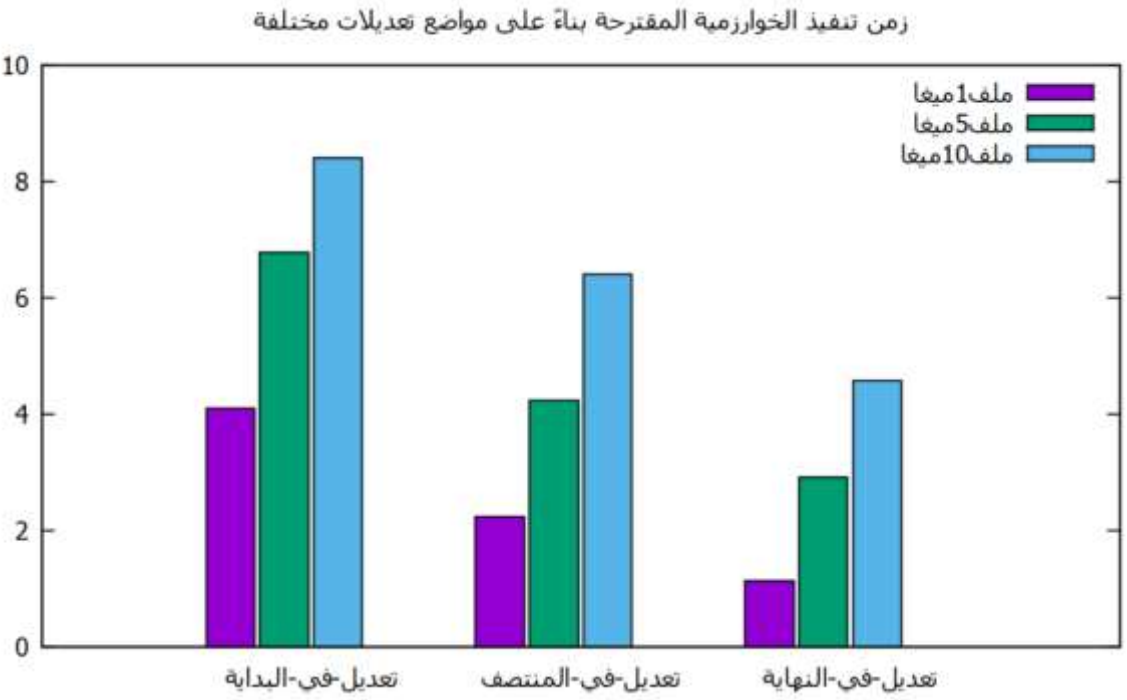


الشكل (6) حجم البيانات المتبادلة في الخوارزمية المقترحة بناءً على أحجام بلوكات مختلفة

5.4.4 – تقييم عمل الخوارزمية المقترحة بناءً على مواضع تعديلات مختلفة :

تم هنا العمل على ثلاث أحجام مختلفة من الملفات والقيام بتعديلات في محتوى الملف قبل عملية المزامنة في ثلاث مواضع مختلفة (البداية – المنتصف – النهاية) لنجد أن زمن التنفيذ يزداد مع اقتراب الاختلاف في محتوى الملف من

بدايته ولكن تبقى الخوارزمية تقدم أداء جديد جداً مقارنة مع عملية إرسال كامل الملف على الشبكة كما في الشكل (6).



الشكل (6) زمن تنفيذ الخوارزمية المقترحة بناءً على مواضع تعديلات مختلفة

6 – الاستنتاجات والتوصيات:

بناءً على ما سبق يمكننا تلخص مجموعة من الاستنتاجات والتوصيات:

- ❖ تعتبر عملية ارسال كامل الملف المتبعة في أنظمة التخزين السحابي الحالية غير فعالة وذلك عند الملفات ذات الأحجام الكبيرة كونها تستهلك عرض حزمة كبير وبالتالي كلفة عالية.
- ❖ قللت الخوارزمية المقترحة من حجم الرسائل المتبادلة بين طرفي المزامنة بشكل ملحوظ مقارنة مع أنظمة ارسال كامل الملف وبالتالي تقليل في حجم استهلاك عرض الحزمة المتوفر.
- ❖ أثبتت الخوارزمية المقترحة فعاليتها في الحفاظ على أداء جيد جدا مع زيادة حجم الملف بين الطرفين.

- مع زيادة حجم التعديلات في الملف المراد مزامنته تبقى الخوارزمية قادرة على العمل مع زيادة نسبية في حجم الحسابات عند الطرف الأقدم.
- زيادة حجم البلوك في الخوارزمية المقترحة يؤدي الى تقليل في عدد الحسابات اللازمة ولكن زيادة في حجم البيانات المرسلة.
- من الممكن تطوير عمل الخوارزمية لتكون قادرة على الاستجابة لأنواع أخرى من التعديلات التي ممكن حدوثها في الملف (إضافة – حذف).
- من الممكن الاستفادة من عمل الخوارزمية للعمل على مزامنة الملفات بين أكثر من طرفين.

7 – الخاتمة:

تم في هذا البحث العمل على تطوير خوارزمية .....

8 – المراجع:

1. D.Ford, et al. "**Availability in Globally Distributed Storage Systems**", Dept. of Industrial Engineering and Operations Research Columbia University, 2010.
2. M.V.Steen and A.S.Tanenbaum, "**A Brief Introduction to Distributed Systems**", Springer, 2016.
3. C.Yan , "**Cloud Storage Services**", CENTRIA UNIVERSITY OF APPLIED SCIENCES, Information Technology, 2017.
4. R.A.Rajan and S.Shanmugapriyaa, "**Evolution of Cloud Storage as Cloud Computing Infrastructure Service**", IOSR Journal of Computer Engineering (IOSRJCE), (May-June 2012), PP 38-45.
5. L.Posani, A.Paccoia, M.Moschettini, "The carbon footprint of distributed cloud storage", June 26, 2019.
6. S.L.Obrutsky, "**Cloud Storage: Advantages, Disadvantages and Enterprise Solutions for Business**", Eastern Institute of Technology, Hawke's Bay, New Zealand, 2016.

7. X.Gao,et al. , "**Building a Distributed Block Storage System for Cloud Infrastructure**", Conference Paper, January 2011.