مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(١٣) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (13) 2023

تعزيز عامل ثقة العقد في توزيع مهام منصات معالجة البيانات الضخمة

د. ماهر إبراهيم*
م. محمد هيفه**

(تاريخ الإيداع ٢٠٢٣/٩/٢٦ . قُبِل للنشر في ٢٠٢٣/١٢/١١)

□ ملخّص□

أصبحت جدولة مهام البيانات الكبيرة في بيئات الحوسبة السحابية محل اهتمام في السنوات القليلة الماضية، نظرًا للنمو الكبير في عدد الشركات التي تعتمد على البنية التحتية السحابية كأساس لتخزين وتحليل البيانات الكبيرة . التحدي الرئيسي في جدولة مهام بيئات الحوسبة السحابية هو ضمان الحد الأدنى للوقت المستغرق في إنجاز عمليات المعالجة وتقليل كمية الموارد المستخدمة .تم اقتراح العديد من الطرق في محاولة لتجاوز هذا التحدي .والقيود الرئيسية لهذه الطرق تنبع من حقيقة أنها تتجاهل مستويات الثقة للعقد مما يعرض جودة الخدمة الكلية لعملية معالجة البيانات الكبيرة للخطر .

ضمن هذا البحث سنقدم نهج جدولة مهام معتمد على ثقة العقد باستخدام تقنيات , HADOOP CLOUDSIM بالإضافة للغة PYTHON . يزيد هذا النهج إنتاجية المعالجة في بيئات البيانات الضخمة .وتقوم الفكرة الأساسية للحل المقترح على ثلاث مراحل، في المرحلة الأولى نقوم بإعطاء قيم ثقة للعقد قبل إسناد المهام لها، بينما في المرحلة الثانية نعمل على ترتيب المهام حسب أولويتها، وفي المرحلة الثالثة والأخيرة نعمل على إسناد المهام المرتبة إلى العقد المناسبة بطريقة ذكية تضمن تقليل وقت الخدمة وتكلفة معالجتها.

أما عن مخرجات هذا البحث فيمكن الاستفادة منها في تحسين عمل المؤسسات التي تعتمد على نتائج تحليل البيانات الضخمة، وفي جميع مجالات الحوسبة السحابية الأخرى وانترنت الأشياء .

الكلمات المفتاحية: مركز بيانات، توزيع الأحمال، جدولة المهام، البيانات الضخمة، استهلاك الموارد، ثقة العقد

* مدرس في قسم هندسة تكنولوجيا المعلومات – كليّة هندسة تكنولوجيا المعلومات والاتصالات – جامعة طرطوس– سوريا
** طالب ماجستير _ كلية هندسة تكنولوجيا المعلومات والاتصالات _ جامعة طرطوس _ طرطوس _ سوريا.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (٧) العدد(١٣) ٢٠٢٣
Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (7) No. (13) 2023

Enhancing the trust factor in tasks distribution of BigData platforms

Dr.Maher Ebrahim*
Eng.Mohammed Haifa**

(Received 26/9/2023 . Accepted 11/12/2023)

□ ABSTRACT

The scheduling of big data tasks in cloud computing environments has gained significant attention in recent years, due to the substantial growth of companies relying on cloud infrastructure as a foundation for storing and analyzing big data. The main challenge in scheduling tasks in cloud computing environments is ensuring minimal completion time and reducing resource utilization. Several approaches have been proposed in an attempt to overcome this challenge. However, the main limitations of these approaches stem from the fact that they disregard contract trust levels, thereby jeopardizing the overall quality of big data processing service. In this research, we present a task scheduling approach that is contract-trust-aware, utilizing Hadoop, CloudSim, and the Python programming language. This approach enhances processing efficiency in big data environments. The core idea of the proposed solution involves three stages. In the first stage, we assign trust values to contracts before allocating tasks to them. In the second stage, tasks are prioritized based on their importance. Finally, in the third and last stage, the prioritized tasks are intelligently assigned to suitable contracts, ensuring a reduction in service time and processing cost. The outputs of this research can be utilized to improve the operations of institutions relying on big data analytics results, as well as in various other domains of cloud computing and the Internet of Things.

Key Words: Datacenter, Load Distribution, Task Scheduling, Big Data, Resource Consumption, trust of nodes

*Teacher, Information Technology Engineering Department, Information and communication Technology Engineering , Tartous University, Syria
** Master Student_ Faculty of Information and Communication Technology _ University of Tartous _ Tartous _ Syria.

١. مقدمــة:

ظهر مصطلح البيانات الضخمة في السنوات القليلة الماضية كمفهوم أساسي بسبب زيادة حجم البيانات التي يتم إنتاجها من قبل مختلف الشركات و المؤسسات يوميًا. حجم البيانات الضخم لا يمكن معالجته و تخزينه باستخدام التقنيات التقليدية المعروفة سابقاً، وهذا أدى إلى الحاجة لاعتماد حلول حديثة تمكن من تخزين وجدولة[1] ومعالجة حجوم كبيرة من البيانات. وقد كانت الحوسبة السحابية هي الخيار الأول لمعظم الشركات التي تسعى لمواكبة تطور البيانات الكبيرة، بفضل مجموعة واسعة من المزايا التي توفرها مثل التعددية، والمرونة، والافتراضية. وبالتالي، بدلاً من شراء وصيانة معدات الأجهزة المكلفة لتخزين وتحليل البيانات الكبيرة، يمكن للشركات الآن نقل هذه المسؤوليات إلى السحابة لتقليل النفقات الرأسمالية والتشغيلية والاستفادة من أداء أفضل. ومع ذلك، يترتب على ذلك مشكلة خطيرة لمسؤولي الحوسبة السحابية [2]، وهي كيفية جدولة مهام البيانات الكبيرة بكفاءة في البيئات الافتراضية لضمان أداء أفضل وتقليل إهدار الموارد.

٢. هدف البحث :

يهدف هذا البحث إلى تطبيق منهجية لجمع معلومات الأداء المتعلقة بمتوسط زمن نبضات الاستجابة، و متوسط تردد نبضات الاستجابة، و استخدام الموارد في عقد المعالجة لاشتقاق قيم الثقة لكل عقدة، بعد ذلك نقوم بتطبيق طريقة للتصنيف بناء على التعلم الآلي لتجميع المهمات في مجموعات اعتماداً على الموارد المطلوبة و كلفة التنفيذ، و في النهاية تحسين آلية توزيع المهام من خلال تعزيز عامل الثقة .

٣. أهمية البحث :

إن زيادة فعالية أنظمة معالجة البيانات الضخمة ينعكس إيجاباً على أنظمة دعم القرار المعتمدة على البيانات التي تم تحليلها و المعلومات التي تم استخلاصها، هذا يؤثر بدوره على عمل المؤسسات التي تستخدم تلك الأنظمة.

٤. طرق البحث و مواده :

قمنا باستخدام مجموعة بيانات جمعتها شركة Bitbrains، وهي مزود خدمة متخصص في استضافة الأعمال والحوسبة للشركات.[3] تتكون مجموعة البيانات من ١٧٥٠ آلة افتراضية تعمل في مركز بيانات موزع وتخدم حتى ٨٢٦٠ مهمة. تشمل المعلومات الواردة في مجموعة البيانات: الطوابع الزمنية timestamps، عدد النوى المركزية الظاهرية المقدمة CPUs، استخدام CPU لكل آلة افتراضية، كمية الذاكرة المقدمة لكل آلة افتراضية، استخدام الذاكرة لكل آلة افتراضية، سرعة قراءة وكتابة القرص لكل آلة افتراضية، وسرعة استقبال وإرسال الشبكة. نأخذ في الاعتبار كل من الآلات الافتراضية الموثوقة وغير الموثوقة، و نقوم بتغيير نسبة الآلات الافتراضية غير الموثوقة من ١٠% إلى ٥٠%.حيث تعتبر الآلات الافتراضية غير الموثوقة هي تلك التي تعرض أداءً سيئًا في خدمة طلبات العملاء وتستهلك كميات غير طبيعية من الموارد (على سبيل المثال، المعالجة المركزية، الذاكرة العشوائية، إلخ).

٥. ما هي البيانات الكبيرة BigData :

البيانات الكبيرة (BigData) هي مصطلح يُستخدم لوصف كميات ضخمة من البيانات التي تكون معقدة ومتنوعة وتتزايد بسرعة كبيرة. تتميز البيانات الكبيرة بثلاثة سمات رئيسية وهي الحجم (Volume) والتنوع (Variety) والسرعة(Velocity) .

يشير الحجم Volume إلى حجم البيانات الهائل الذي يتجاوز قدرة أدوات إدارة قواعد البيانات التقليدية على التعامل معه. قد تتضمن البيانات الكبيرة مجموعات ضخمة من السجلات و المعلومات التي قد تصل إلى مئات التيرابايتات أو حتى البتابايتات.

أما التنوع Variety فيشير إلى أن البيانات الكبيرة تأتي بتنسيقات متنوعة وتشمل نصوصًا، وصورًا، ومقاطع فيديو، وملفات صوتية، وبيانات اجتماعية، وبيانات جغرافية، وغيرها. هذا التنوع يزيد التحدي في تحليل واستخلاص المعلومات من تلك البيانات.

أما السرعة Velocity فتشير إلى سرعة تدفق البيانات وتوليدها، حيث يتم تجميع وتحديث البيانات بمعدلات عالية جدًا. فمثلاً، في مجالات مثل وسائل التواصل الاجتماعي والأجهزة الطبية والاقتصاد والتجارة الإلكترونية، يتم توليد كميات ضخمة من البيانات في وقت قصير جدًا.

إن إدارة وتحليل البيانات الكبيرة يشكل تحديًا كبيراً، ولكنها توفر أيضًا فرصاً هائلة للاستفادة من البيانات في استخلاص الأنماط والتوجهات واتخاذ القرارات الأعمق والأكثر دقة.

تعتمد BigData بشكل أساسي على الحوسبة السحابية والتي تعرف بأنها تقديم أو توصيل خدمات وموارد الحاسوب من خلال شبكة الإنترنت، ويتم توفير واجهات لإدارة الخدمات المقدمة من خلال تطبيقات ويب.

يتم تقديم هذه الخدمة من قبل الشركات بمقابل مادي يتم تحديده على أساس الاستخدام، وهذا ما يميز خدمة الحوسبة السحابية عن خدمة مركز البيانات Data Center، حيث في خدمة مركز البيانات يتم حجز موارد الحاسوب ويتم الدفع مقابل الموارد المحجوزة سواء تم استخدامها أو لم يتم استخدامها. [4]

٦. الدراسات المرجعية :

• **الدراسة الأولى:** في عام ٢٠١٨ حاول الباحثون [5] بتحقيق التوازن بين استهلاك الطاقة في مراكز المعالجة و تقليل زمن تنفيذ المهام بنفس الوقت. للوصول لهدفهم قاموا بتصميم حل يعتمد على تقنية تحجيم التردد و الجهد الديناميكي DVFS التي تمكن الأجهزة من العمل عند توليفات متنوعة من الترددات و الجهود، كما قاموا باستخدام خوارزمية Non-dominated Sorting Genetic Algorithm (NSGA-II) للحصول على الحل الأفضل، بالإضافة لاستخدام شبكة عصبية ذكية للتنبؤ بالأجهزة الافتراضية المناسبة بناءً على خصائص المهام و ميزات الموارد، و أظهرت النتائج التي حصل عليها الباحثون أن استهلاك الطاقة و الزمن الإجمالي للمهام هدفان متناقضان، ويتم اختيار الحل الأنسب بناءً على ظروف العمل.

• **الدراسة الثانية:** في ٢٠١٨ قام وانغ و فريقه [6] باقتراح نهجاً بايزياً Bayesian لجدولة المهام، يهدف لتحسين الأداء من خلال تقليل احتمالية الفشل في تنفيذ المهمة وضمان التنفيذ في بيئة آمنة، قام هذا النهج بتقييم العقد بناءً على سلوكياتها السابقة وتوصيات العقد الأخرى، أثبتت تجارب المحاكاة أن الخوارزمية المقترحة تقلل بشكل فعال نسبة الفشل لكن مع تكلفة وقت أطول قليلاً بالمقارنة مع خوارزمية Binary Search Algorithm .

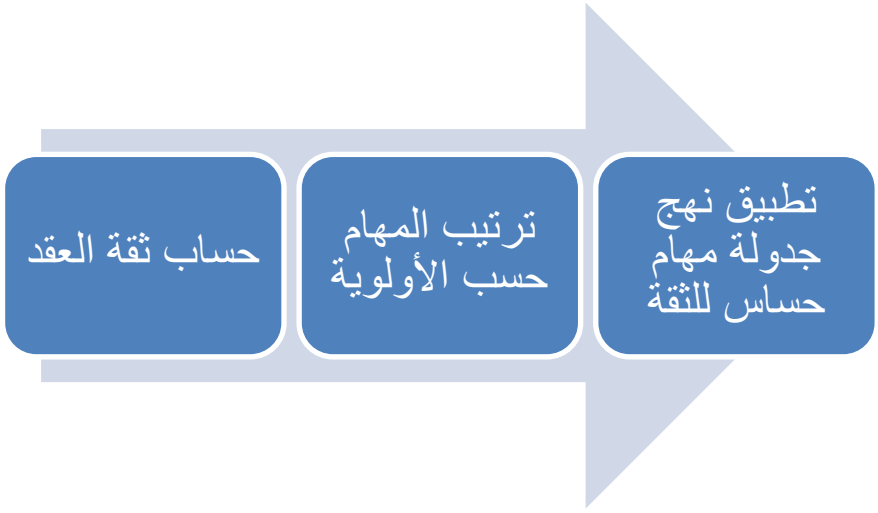
• **الدراسة الثالثة:** في ٢٠١٩ قام الباحثون [7] بتقديم خوارزمية جدولة مهام هجينة تسمى Fuzzy Membership Particle Swarm Optimization (FMPSO) لتحسين توزيع الحمل والإنتاجية في البيانات الضخمة، من خلال استخدام المنطق الضبابي بالإضافة لخوارزمية أسراب الطيور المعدلة، حيث قاموا بتحسين خوارزمية الطيور من خلال إدخال تقنية Shortest Job to Fastest Processor (SJFP) التي تقوم باختيار المهام ذات زمن التنفيذ الأقصر أولاً مما يؤدي إلى تقارب أسرع للخوارزمية، كما استخدموا تقنية Self-adaptive learning technique بحيث تبحث العقدة عن المورد الأفضل لها بعيداً عن بقية السرب، وفي النهاية يأتي دور المنطق الضبابي لاختيار الحل الأمثل. أظهر نتائجهم أن الخوارزمية المقترحة أسرع و متقاربة بنسبة كبيرة مقارنة مع الخوارزمية الجينية GA, كما توفر تحسينات واضحة في استهلاك الموارد[8] .

• **الدراسة الرابعة:** في ٢٠١٦ قام الباحثون [9] باستخدام Feedback لتقييم العلاقة بين المستخدمين و عقد المعالجة، بعدها تم استخدام خوارزمية Direct Cycling Graph(DAG) لتحليل الاعتمادات بين المهام لتحسين توزيع المهام و تخطيطها بشكل فعال مما يقلل احتمالية حدوث تعارض عند تنفيذ المهام و يزيد إنتاجية العمل. وأظهرت النتائج التي حصلوا عليها أن الحل المقترح يقلل الزمن الإجمالي للمهام وكذلك الكلفة المادية بالمقارنة مع خوارزميتي PSO و ACO .

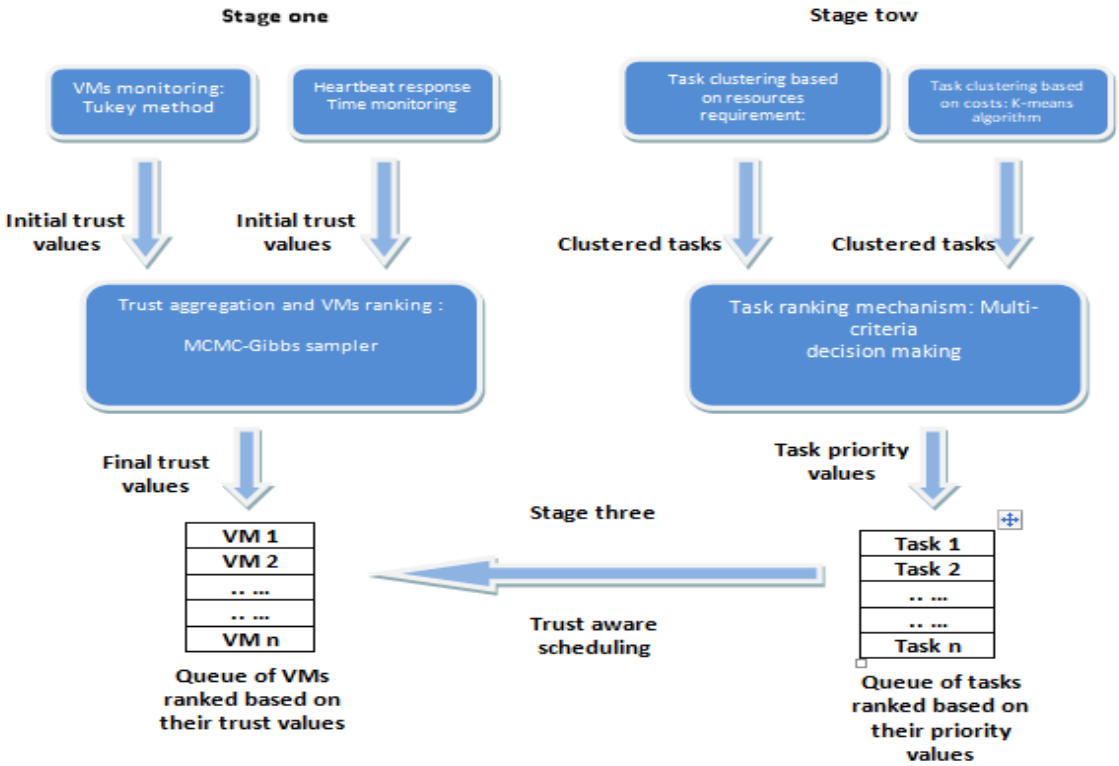
• **الدراسة الخامسة:** في ٢٠١٥ قام Sumathi [10] بتحسين خوارزمية Heterogeneous Earliest Finish Time (HEFT) التي تقوم بإعطاء أولوية مرتفعة للمهام التي شارفت على الانتهاء في الأنظمة الموزعة، فقام بأخذ تقييم العقد بعين الاعتبار من خلال سجلات الأداء السابقة[11] وأظهرت النتائج التي حصل عليها الباحث أن التحسين المطبق قلل الزمن الإجمالي للمهام بنسبة ٢.٤٥٩% .

٧.منهجية العمل المقترحة :

تتكون منهجية العمل من ثلاث مراحل كالتالي (الشكل ١) :



الشكل(١): المراحل الثلاث الأساسية في المنهجية المقترحة



الشكل (٢) : حساب ثقة العقد و ترتيب المهام حسب أولويتها ثم الربط بين المهام والعقد المناسبة لها باستخدام نهج حساس للثقة

١.٧. المرحلة الأولى: حساب الثقة للعقد:

تنقسم هذه المرحلة إلى عدة خطوات جزئية وهي :

١.١.٧. الخطوة الأولى: حساب الثقة المبنية على نبض الاستجابة:

يتم صنع قرارات جدولة المهام من قبل عقدة رئيسية تسمى متعقب المهام jobtracker، بينما تتحمل العقد العاملة، التي تسمى tasktrackers، مسؤولية تنفيذ المهمة. يحتفظ متعقب المهام الرئيسي jobtracker بقائمة انتظار للمهام التي تعمل حاليًا، وحالة كل tasktrackers، وقائمة المهام المخصصة لكل tasktracker. يقوم كل tasktracker بدوره بتقرير حالته (نشط، غير نشط) بانتظام إلى متعقب المهام الرئيسي jobtracker على شكل رسالة نبضة.

يجمع متعقب المهام الرئيسي jobtracker النبضات التي تم استلامها من مجموعة tasktrackers. إذا لم يتم استلام نبضة في فترة زمنية معينة، فهذا يعني اعتبار الآلة الافتراضية ميتة أو معطلة. لذلك، لحساب نسبة تردد استجابة النبض Q_j ، نفرض γ_j متوسط تردد النبض المعتاد (المسجل سابقاً) للآلة v_j، و ليكن φ_j تردد النبض الحالي لآلة الافتراضية v_j. فيكون لدينا [12] :

$$Q_j = \frac{\varphi_j}{\gamma_j} \times 100\% \qquad (١)$$

و يكون متوسط زمن استجابة النبضات Φ_j كالتالي [12]:

$$\Phi_j = \frac{\sum_{x=1}^{N_j} \eta_j^x}{N_j}$$

(٢)

حيث يُمثّل η_j^x وقت استجابة النبضات x للعقدة $v_j \epsilon V$ ، و N_j هو إجمالي عدد النبضات المستلمة لـv_j. بعد ذلك، يمكن حساب النسبة المئوية متوسط وقت الاستجابة كما يلي[12]:

$$W_j = \frac{\Phi_j}{\psi_j} \times 100\% \qquad (٣)$$

حيث يُمثّل ψ_j وقت استجابة النبضات المعتاد للعقدة v_j من ملف سجلها الزمني.

أخيرًا، يتم حساب قيمة الثقة الأولية للآلة الافتراضية v_j FirstInitialTrust عن طريق قسمة مجموع نسبة تردد استجابة النبض ونسبة زمن الاستجابة على اثنان . والسبب وراء هذه الصيغة هو أننا نفترض أن كل نسبة تردد استجابة النبض ونسبة زمن الاستجابة لهما أهمية متساوية.

$$FirstInitialTrust_j = \frac{Q_j + W_j}{2} \qquad (٤)$$

٢.١.٧. الخطوة الثانية: حساب الثقة المبنية على استهلاك الموارد:

في هذه المرحلة، يقوم كل tasktracker بمراقبة استخدام وحدة المعالجة المركزية (CPU) وذاكرة الوصول العشوائي (RAM) وعرض النطاق الترددي للشبكة network bandwidth واستهلاك مساحة التخزين على الأقراص disk storage للآلات الافتراضية (VMs) لإنشاء مجموعة بيانات استهلاك لكل VM (Virtual Machine). ثم نستخدم تقنية الإحصاء IQR (Interquartile Range) للكشف عن أي استهلاك غير طبيعي للموارد. حيث يعتبر IQR مقياسًا للتباين يساعد في الكشف عن القيم الشاذة في مجموعة بيانات، وبالتالي معرفة الموارد التي يتم استهلاكها بشكل أعلى أو أقل من الحد المسموح به.

تعتمد فكرة IQR على تقسيم البيانات إلى أرباع (Q_1, Q_2, Q_3) الربع الأول Q_1 وهو القيمة التي تكون ٢٥% من البيانات أقل منها، و Q_2 هي القيمة المتوسطة لمجموعة البيانات، Q_3 هو القيمة التي تكون ٢٥% من البيانات أكبر منها. أما قيمة IQR يتم حسابه من خلال حساب الفرق بين Q_3 و Q_1 .

بعدها تتم إضافة IQR إلى IQ_2 (نهاية الربع الثاني) ليتم حساب الحد الأعلى، كما يتم طرح IQR من IQ_1 (بداية الربع الأول) لحساب الحد الأدنى.

ثم يقوم jobtracker بمراقبة أي استهلاك مستقبلي للآلة الافتراضية VM في زمن $t+\delta$ لتحديد ما إذا كان هناك أي استهلاك يتجاوز الحد العلوي أو الأدنى المحسوب، فإذا تم اكتشاف أي استهلاك غير عادي يتم تخزين القيم الشاذة ضمن جدول ليتم حساب متوسط الاستهلاك الغير عادي لكل مقياس، عن طريق قسمة مجموع المتوسطات غير العادية لاستهلاك الموارد على عدد الموارد التي استخدمتها العقدة بشكل زائد أو التي استخدمتها بشكل غير كاف، على النحو التالي [13]:

$$UpperInitialTrust_j = \frac{\sum_{z \in M} PropOverUse_j^z}{|OverusedMetrics_j|} \qquad (5)$$

$$LowerInitialTrust_j = \frac{\sum_{z \in M} PropUnderUse_j^z}{|UnderusedMetrics_j|} \qquad (6)$$

إذا لم يتم استخدام أي مورد بشكل زائد أو غير كاف، فإن الثقة الأولية للعقدة ستكون ١.

ثم، عن طريق حساب مجموع $UpperInitialTrust_j$ و $LowerInitialTrust_j$ وقسمته على اثنين نحصل على الثقة الشاملة لكل عقدة SecondInitialTrust .

$$SecondInitialTrust_j = \frac{UpperInitialTrust_j + LowerInitialTrust_j}{2} \times 100\% \quad (7)$$

قمنا باختيار مقياس IQR لسببين :

- متانته أمام البيانات غير المنظمة وغير المرتبة.
- طبيعته البسيطة والخفيفة التي لا تتطلب جهدًا حاسوبيًا كبيرًا.
- قدرته على كشف القيم الشاذة.

٣.١.٧. الخطوة الثالثة: ترتيب العقد حسب قيم الثقة باستخدام MCMC gibbs sampling :

إن قيم الثقة الخاصة بالعقد هي قيم متغيرة في كل لحظة، لذلك لابد من استخدام توزيع احتمالي لإيجاد القيم الحالية للثقة بناءً على القيم السابقة.

تعتبر MCMC Gibbs sampling أحد الأساليب الشائعة للتعامل مع الموديلات الاحتمالية المرتبطة بمتغيرات مختلفة، وتستخدم في العديد من مجالات البحث والتحليل مثل الإحصاءات البيئية وتحليل البيانات والذكاء الاصطناعي وغيرها.

في هذه المرحلة ، نستخدم عينات Gibbs sampling لترتيب العقد بناءً على قيم الثقة الخاصة بها. يتم استخدام القيم الأولية للثقة التي تم الحصول عليها من خوارزميات المراقبة وقيم الثقة التي تم الحصول عليها من مرحلة النبضات كمدخلات لهذه المرحلة.

عملية توليد عينات MCMC Gibbs تعتمد على تحليل سلسلة ماركوف التي يتم فيها تحديث كل متغير في الوقت الحالي مع الاحتفاظ بقيم المتغيرات الأخرى ثابتة. يتم تكرار هذه العملية عدة مرات للوصول إلى توزيع مستقر.

نفرض $X_{V,t}$ مصفوفة مرتبة من قيم الثقة لجميع VMs خلال اللحظات الزمنية $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ ،ولتكن e_{jh} قيمة الثقة الحالية لـ v_j في اللحظة τ_h ،نوضح في الجدول التالي قيم الثقة لـ VMs في كل لحظة زمنية:

VMs \ Time Moments	τ_1	τ_2	τ_3	.....	τ_n
v_1	e_{11}	e_{12}	e_{13}	.....	e_{1n}
v_2	e_{21}	e_{22}	e_{23}	.....	e_{2n}
v_3	e_{31}	e_{32}	e_{33}	.....	e_{3n}
⋮	⋮	⋮	⋮		⋮
v_m	e_{m1}	e_{m2}	e_{m3}	.....	e_{mn}

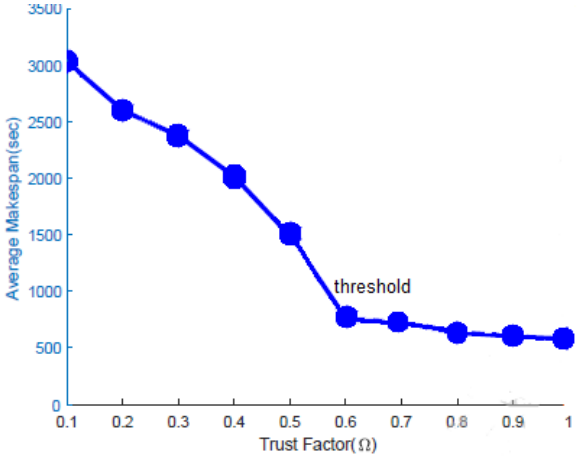
جدول ١ قيم الثقة لكل VM في كل لحظة زمنية

لتكن المجموعة $\mu = (\mu_1, \mu_2, \ldots, \mu_m)$ تعبر عن متوسط قيم الثقة لـ VM عبر اللحظات الزمنية . فتكون قيمة الثقة لـ VM تعطى بالعلاقة [14]:

$$e_{jh} = \mu_j + \varepsilon_{jh}, \qquad \varepsilon_{jh} \sim N(0, \sigma^2), \qquad (1 \le h \le n\,; 1 \le j \le m) \quad (8)$$

حيث ε_{jh} قيمة صغيرة جداً، وقيمتها لكل عقدة v_j في اللحظة h مستقلة تماماً.

بعد حساب قيم الثقة ، يمكننا استبعاد العقد التي لديها قيم ثقة منخفضة ، والتي تعني أداءً منخفضًا ، على سبيل المثال العقد الخاملة والبطيئة في الاستجابة. ويتم ذلك عن طريق مقارنة قيمة الثقة لكل عقدة مع عتبة الثقة W التي نقوم بتحديدها من خلال تغيير قيمة الثقة بشكل ديناميكي لقياس التأثير على الوقت الإجمالي. وتظهر النتائج بأن بعد قيمة ٦٠% للثقة ينخفض الوقت الإجمالي للمهام بشكل طفيف كما في الشكل (٣) ، و بالتالي يمكننا استنتاج أن قيمة ٦٠% هي قيمة مناسبة كعتبة للثقة W.



الشكل (٣) انخفاض الوقت الإجمالي للمهام مع زيادة عامل الثقة

٢.٧ المرحلة الثانية: تجميع المهام حسب أولويتها (Task Clustering)

وتنقسم هذه المرحلة إلى عدة خطوات جزئية وهي:

١.٢.٧. الخطوة الأولى: تطبيق طريقة النسبة المئوية Percentile method لتجميع المهام بناءً على متطلبات الموارد :

بعد حساب درجات الثقة للعقد و ترتيبها بناءً على قيم ثقتها، يتم تجميع المهام على أساس متطلبات الموارد الخاصة بها. للقيام بذلك، نستفيد من طريقة النسبة المئوية [15]، التي تستخدم لتقسيم البيانات في مجموعة معينة إلى نسب مئوية .

- المبدأ العام لهذه الطريقة يقوم على فكرة ترتيب متطلبات الموارد للمهام لكل مقياس (أي CPU و RAM والتخزين وعرض النطاق الترددي).
- ثم نحسب المرتبة الأدنى و المرتبة الأعلى ، حيث المرتبة الأدنى تمثل المؤشر الذي تقع تحته ٢٥٪ من القيم، و المرتبة العليا المؤشر الذي يقع فوقه ٢٥٪ من القيم.
- في النهاية يتم تعيين المهام التي تتطلب موارد تقع تحت النسبة المئوية الأدنى في مجموعة المتطلبات المنخفضة k_{low} والمهام التي تتطلب موارد تقع فوق النسبة المئوية العالية في مجموعة المتطلبات العالية k_{high}

٢.٢.٧. الخطوة الثانية: تجميع المهام حسب التكلفة المادية باستخدام خوارزمية K-Means :

تعتبر خوارزمية K-means واحدة من أكثر تقنيات التعلم غير المشرف عليها انتشارًا، حيث تهدف إلى تجميع مجموعة من البيانات بناءً على درجة تشابهها. في هذا العمل، قمنا باستخدام هذه الخوارزمية لتجميع المهام إلى خمسة مجموعات مختلفة (منخفض جدًا، منخفض، متوسط، مرتفع، ومرتفع جدًا) بناءً على تكاليفها. تقوم الخوارزمية بالتصنيف باتباع الخطوات التالية :

١. تحديد عدد المجموعات المطلوبة (k) التي نرغب في تجزئة البيانات إليها، و في حالتنا لدينا ٥ مجموعات.

٢. تحديد نقاط البداية الأولية للمراكز المتوقعة للمجموعات الأولية . يتم تحديد هذه النقاط عادة بشكل عشوائي أو بناءً على معلومات مسبقة إذا كانت متاحة.

٣. حساب المسافة بين كل نقطة في مجموعة البيانات وبين المراكز الحالية للمجموعات باستخدام مقياس المسافة المناسب (مثل مسافة الأقليدية أو مسافة مانهاتن)

٤. تعيين كل نقطة إلى المجموعة التي تحتوي المركز الأقرب إليها من حيث المسافة.

٥. حساب المراكز الجديدة للمجموعات بناءً على النقاط المتواجدة في كل مجموعة.

٦. تكرار الخطوتين ٣ و ٤ حتى يتم تعيين جميع النقاط إلى أقرب مركز. ثم يتم تعيين المراكز الجديدة المحسوبة في الخطوة ٥ كمراكز جديدة للمجموعات، وتتكرر عملية حساب المسافة وإعادة التعيين للنقاط.

السبب وراء استخدام خوارزمية K-means أنها فعالة ومتينة في البيئات الديناميكية التي تتغير فيها خصائص المهام من تكرار إلى آخر.

٣.٢.٧. الخطوة الثالثة: تحديد مستوى أولوية المهمة متعددة المعايير

بعد تجميع المهام استنادًا إلى متطلبات الموارد الخاصة بها (القسم ١.٢.٦) وتكلفتها المرتبطة (القسم ٢.٢.٦) ، نقوم بتحديد مستوى الأولوية لكل مهمة. نعتمد تقنية صنع القرارات متعددة المعايير [6] عن طريق اتباع الخطوات التالية:

• نقوم بتحديد المعايير التي سيتم بناءً عليها إعطاء وزن للمهمة و هذه المعايير هي تكلفة المهام المشتقة في القسم ٢.٢.٦ ، و متطلبات المهام المشتقة في القسم ١.٢.٦ .

• وضع المهام في مصفوفة قرار بناءً على المجموعات التي ينتمون إليها.

• حساب وزن كل مهمة، لكل مقياس (أي وحدة المعالجة المركزية، الذاكرة العشوائية، النطاق الترددي، وتخزين القرص)، بناءً على موضعها في المصفوفة.

• حساب الأولوية النهائية لكل مهمة، بالنسبة لجميع المقاييس.

بفرض المجموعة A تعبر عن متطلبات المهام A = {a1, a2, a3} حيث يمثل a1 مجموعة متطلبات الموارد المنخفضة ، في حين يمثل a2 مجموعة متطلبات الموارد المتوسطة ، ويمثل a3 مجموعة متطلبات الموارد العالية.

و المجموعة C تعبر عن تكلفة المهام C = {c1, c2, c3, c4, c5} حيث يمثل c1 مجموعة تكلفة المهمة المنخفضة جدًا ، في حين يمثل c2 مجموعة تكلفة المهمة المنخفضة ، ويمثل c3 مجموعة تكلفة المهمة المتوسطة ، ويمثل c4 مجموعة تكلفة المهمة العالية ، ويمثل c5 مجموعة تكلفة المهمة العالية جدًا.

فتكون لدينا مصفوفة القرار كالتالي :

$$D^z(T) = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccccc} c_1 & c_2 & c_3 & c_4 & c_5 \end{array} \\ \begin{pmatrix} G_{11} & G_{12} & G_{13} & G_{14} & G_{15} \\ G_{21} & G_{22} & G_{23} & G_{24} & G_{25} \\ G_{31} & G_{32} & G_{33} & G_{34} & G_{35} \end{pmatrix} \end{array} \quad (٨)$$

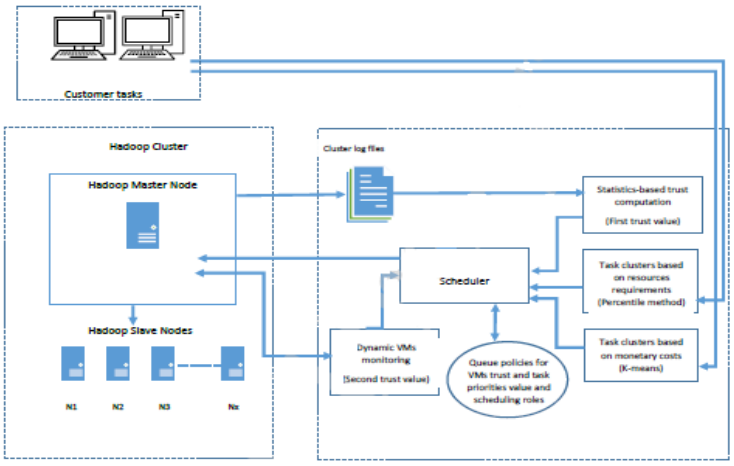
في المصفوفة يمثل كل G_{ru} موضع المهمة التي تنتمي إلى مجموعة a_r ومجموعة c_u . باستخدام هذه المصفوفة ، يتم تقديم صيغة درجة الأولوية للمهمة t_i في المعادلة (٩) [11] :

$$\rho_i = \frac{\sum_{z \in M} G_{ru}^z}{4 \times G_{35}} \times 100\%, \quad (٩)$$

٣.٧. المرحلة الثالثة :

تطبيق نهج جدولة مهام حساس للثقة:

هي المرحلة الأخيرة التي يتم تطبيقها في النهج المقترح، توضح كيف يتم إسناد كل مهمة task في بيئة Hadoop إلى الآلة الافتراضية VM المناسبة لها ، تكون مدخلات هذه المرحلة هي قائمة بالآلات الافتراضية المرتبة حسب قيم الثقة الخاصة بها، وقائمة بالمهام المرتبة حسب أولويتها. يوضح الشكل (٤) آلية عمل النهج المقترح ضمن بيئة Hadoop.



الشكل (٤) مخطط يوضح آلية جدولة المهام

في هذه المرحلة يتم إسناد المهام بعد ترتيبها وفق الأولوية إلى العقد بعد حساب قيم الثقة الخاصة بها ، وفيما يلي تلخيص لخطوات العمل، الشكل (٤) :

١. حساب قيمة الثقة لكل VM بالاعتماد على مرحلة تردد نبض الاستجابة heartbeats frequency، ومرحلة قياس استهلاك الموارد.

٢. مقارنة قيمة الثقة لكل VM مع عتبة الثقة W. إذا كانت قيمة الثقة لـ VM أقل من W، تعتبر الـ VM غير موثوق بها ويتم إنهاؤها لتجنب تعيين البيانات والمهام إلى VMs غير موثوق بها، ولتقليل مساحة المدخلات لعملية الجدولة. وإلا، سيتم إضافة الـ VM إلى قائمة الانتظار Q_V، التي تخزن الـ VMs بترتيب تنازلي حسب قيم الثقة الخاصة بها.

٣. تحديد مستوى الأولوية لكل مهمة باستخدام طريقة النسبة المئوية و خوارزمية K-means.

٤. إسناد المهام المرتبة حسب أولويتها إلى العقد المرتبة حسب قيم الثقة.

٩. النتائج والمناقشة :

سنقوم بمناقشة نتائجنا مع خوارزميات الجدولة الأخرى المستخدمة في **الدراسة المرجعية الثالثة** [7] وهي كالتالي :

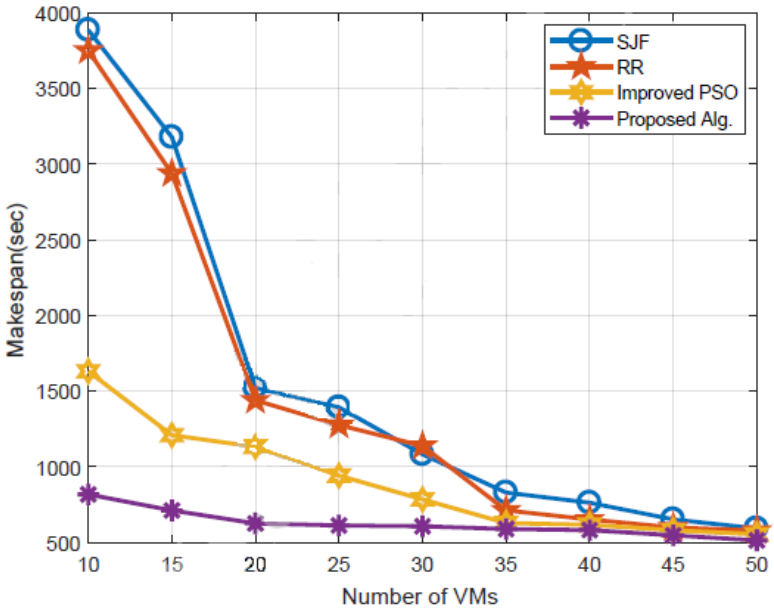
١. **خوارزمية Shortest Job First (SJF)** : وهي خوارزمية جدولة تستخدم في نظم التشغيل. تعتمد هذه الخوارزمية على فكرة تخصيص المعالج للمهام ذات مدة تنفيذ قصيرة أولاً. أي أن هذه الخوارزمية تأخذ بالاعتبار حجم المهام لتحديد ترتيب تنفيذها. تكمن الميزة الرئيسية لـ SJF في بساطتها و في حقيقة أنها تقلل من المتوسط الزمني الذي تضطر فيه كل عملية إلى الانتظار حتى يتم الانتهاء من تنفيذها. بينما تكمن القيود الرئيسية لها في أنه :

- يمكن أن يترتب عليها وقت انتظار طويل للمهام الطويلة إذا تمت إضافة مهام قصيرة باستمرار ،
- بالإضافة إلى أن هذه الوارزمية تتجاهل قيم الثقة للآلات الافتراضية في عملية الجدولة.

٢. **خوارزمية Round Robin (RR)** : تعتمد هذه الخوارزمية على تقسيم وحدة المعالجة المركزية بين المهام المختلفة بشكل متساوٍ ومستمر. عند استخدام خوارزمية RR، يتم تخصيص وحدة المعالجة المركزية لكل مهمة لفترة زمنية محددة تسمى (time quantum) أو "الكم الزمني". عند انتهاء الكم الزمني، يتم تعليق تنفيذ المهمة الحالية ونقلها إلى نهاية قائمة الانتظار ويتم إعطاء الفرصة للمهام الأخرى في الانتظار لتنفيذها. يتم تكرار هذه العملية حتى استكمال تنفيذ جميع المهام. تكمن مزايا RR في أنه سهل التنفيذ و يتجنب حالة الحرمان للمهام starvation ، على عكس SJF.بينما يتمثل العيب الرئيسي لـ RR في أن :

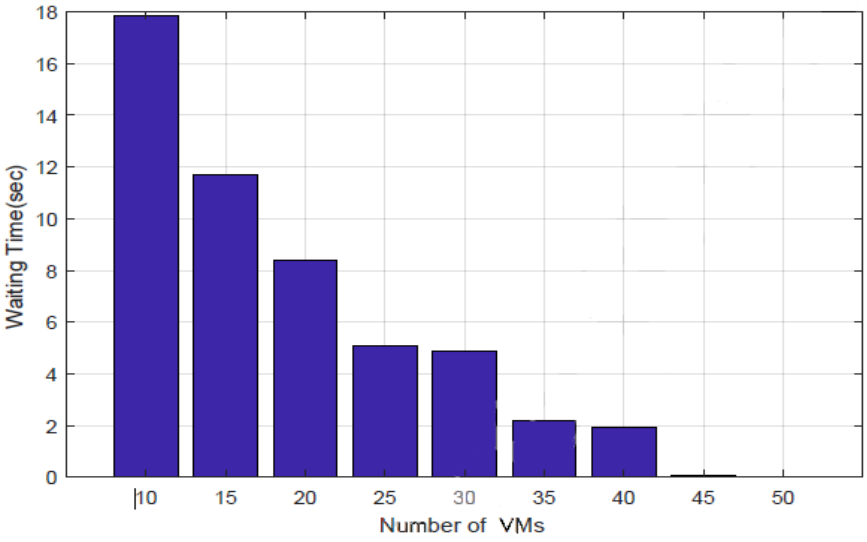
- الإنتاجية تعتمد إلى حد كبير على طول الكم الزمني. حيث أنه إذا تم اختيار الكم الزمني ليكون كبيرًا جدًا ، فسيتصرف RR كطريقة القادم أولاً يخدم أولاً (first come first serve). و بالمقابل إذا تم اختيار كم زمني صغير فإن الخسارة في زمن تبديل المهام ستكون كبيرة ، مما يعني إنتاجية أقل.
- القيد الثاني لـ RR هو أنه يتجاهل قيم الثقة للآلات الافتراضية عند جدولة المهام.

٣. **خوارزمية Improved PSO** : تتألف الخوارزمية من مجموعة من الجسيمات الافتراضية. يتم تمثيل كل جسيم بموقع في الفضاء البحثي، ويحتفظ بذاكرة لأفضل حل وصل إليه حتى الآن ، بالإضافة إلى سرعته واتجاهه الحالي. الفكرة الرئيسية للنهج المحسن لـ PSO هي تغيير أوزان الجسيمات مع زيادة عدد التكرارات و إضافة بعض الأوزان العشوائية في المراحل النهائية لتجنب توليد حلول محلية مثلى بدلاً من إيجاد الحل الأمثل في كامل فضاء البحث . ومع ذلك، مثل SJF و RR، تتجاهل improved PSO أيضًا ثقة الآلات الافتراضية في عملية الجدولة، مما قد يؤدي إلى جودة خدمة أقل وتكاليف تنفيذ أعلى.



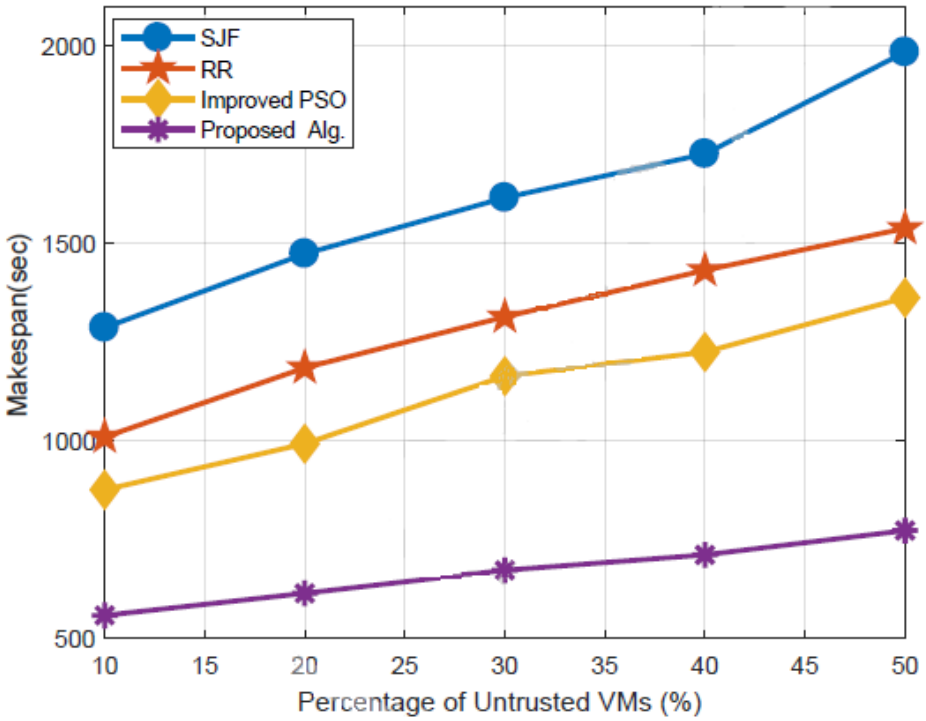
الشكل (٥) انخفاض الوقت الإجمالي للمهام مع زيادة عدد العقد

نلاحظ في الشكل (٥) أن الحل المقترح يقلل من الوقت الإجمالي المستغرق للمهمات Makespan مقارنة بالحلول الأخرى، خاصةً في بيئة وحدات المعالجة الافتراضية ذات الكثافة المنخفضة (أي عندما يكون عدد وحدات المعالجة المنتشرة صغيرًا نسبيًا). السبب في ذلك هو أننا في الحل المقترح نوفر خوارزمية تقوم بربط متطلبات الموارد للمهام مع قدرات وحدات المعالجة الافتراضية بشكل ذكي. بعبارة أخرى، نقلل من احتمالية تعيين المهام التي تتطلب كميات كبيرة من الموارد إلى وحدات المعالجة الافتراضية ذات الأداء المنخفض التي لن تكون قادرة على خدمة تلك المهام بكفاءة. الملاحظة الثانية من هذا الشكل هي أنه بزيادة عدد وحدات المعالجة الافتراضية المنتشرة، يتجه الفارق في الأداء بين الحلول المختلفة إلى أن يكون أصغر. يمكن تبرير ذلك بحقيقة أن زيادة عدد وحدات المعالجة (مع افتراض أن جميع هذه الوحدات موثوقة) يؤدي إلى زيادة الموارد المتاحة لخدمة المهام. ومع ذلك، بتحسين الأداء في حالة وحدات المعالجة الافتراضية ذات الكثافة المنخفضة، سيساعد هذا الحل مقدمي خدمات السحابة في تقليل تكاليفهم الإجمالية من خلال تقليل حاجتهم إلى نشر المزيد من وحدات المعالجة.



الشكل (٦) انخفاض زمن الانتظار مع زيادة العقد

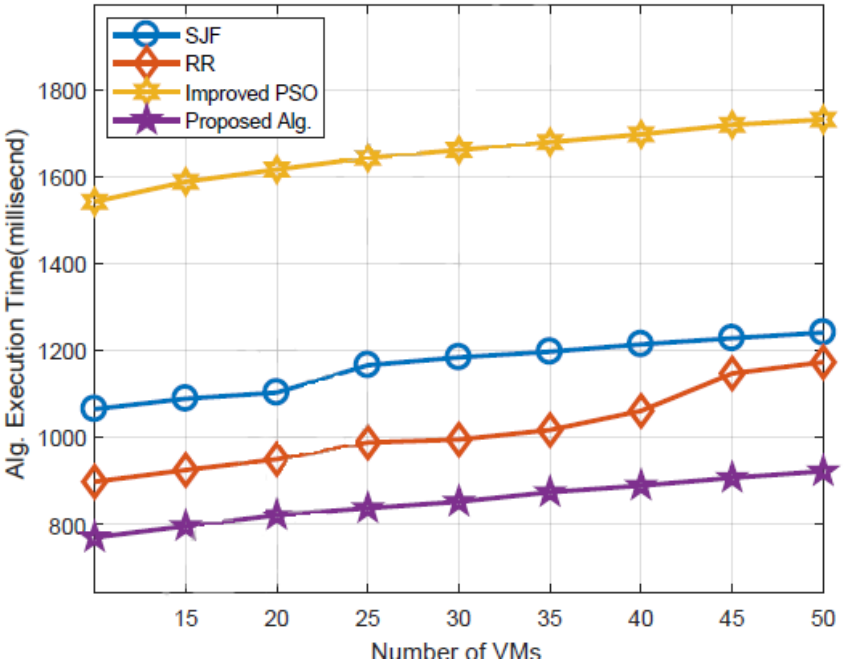
كما نلاحظ في الشكل (٦) أن زيادة عدد وحدات المعالجة المنتشرة يؤدي إلى انخفاض كبير في أوقات الانتظار. على الرغم من أن هذه النتيجة متوقعة، إلا أن هدف هذا الشكل هو أن يوضح أنه حتى لعدد صغير من وحدات المعالجة المنتشرة وعدد كبير جدًا من المهام (مثل ٧,٨٨٤)، فإن الوقت المستغرق في الانتظار الناتج عن نموذجنا لا يزال مقبولًا. على سبيل المثال، الوقت الإجمالي للانتظار لتعيين ٧,٨٨٤ إلى ١٠ وحدات معالجة افتراضية هو ١٨ ثانية.



الشكل (٧) انخفاض الوقت الإجمالي للمهام مع زيادة نسبة العقد غير الموثوقة

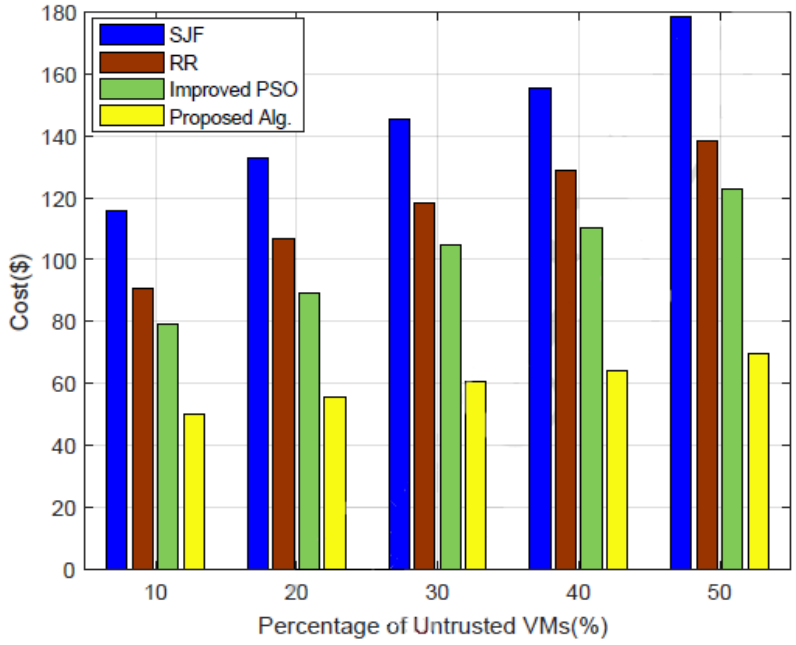
نثبت عدد وحدات المعالجة عند ٥٠ وحدة، وعدد المهام عند ٧,٨٨٤. نلاحظ في الشكل (٧) أن الحل المقترح يمكنه تقليل الوقت المستغرق بشكل كبير عندما يبدأ نسبة وحدات المعالجة غير الموثوقة من ١٠٪.

السبب وراء هذا التحسن هو أننا في الحل المقترح نستخدم آلية تأسيس الثقة ذات المرحلتين وتقنية تجميع تستند إلى MCMC Gibbs Samplerbased لاستنتاج قيمة الثقة لكل وحدة معالجة وتصنيفها بناءً على درجة موثوقيتها ، و هذا أمر مهم لتجنب تعيين المهام إلى وحدات معالجة غير موثوقة و التي قد يؤثر أداؤها الضعيف على الفترة الزمنية الكلية وفرص نجاح عملية تنفيذ المهمة.



الشكل (٨) الحل المقترح يحقق زمن تنفيذ أقل من باقي الحلول

في الشكل (٨) نلاحظ أن الحل المقترح يتمتع بوقت تنفيذ أقل من الحلول الأخرى، السبب في ذلك أنه على الرغم من احتوائه عدة مراحل، لكن اعتماد نهج جدولة مبني على الثقة يقلل احتمال فشل بعض المهام على بعض وحدات المعالجة غير الموثوقة وإعادة جدولتها. علاوة على ذلك، يساعد تجميع وحدات المعالجة بناءً على الثقة وتحديد أولوية المهام في الحل المقترح على تحسين التوافق بين متطلبات الموارد للمهام وتوافر هذه الموارد في وحدات المعالجة الهدف.



الشكل (٩) الحل المقترح يقلل تكلفة تنفيذ المهام على مزودي الخدمة عند زيادة نسبة العقد غير الموثوقة

يبين الشكل (٩) أن الحل المقترح يساعد مزودي الخدمات على تقليل تكاليفهم المالية مقارنة بالأساليب الأخرى في وجود آلات ظاهرة غير موثوقة. و يمكن تبرير ذلك بالنتائج التي وجدناها سابقاً بأن الحل المقترح يقلل من إجمالي وقت جدولة وتنفيذ المهام في وجود عقد غير موثوقة. وهذا، بدوره، يؤدي إلى تقليل كمية الموارد المستهلكة لخدمة المهام، مما يؤدي إلى تقليل التكاليف المالية لمزودي الخدمة.

١٠. الاستنتاجات و التوصيات :

في هذه البحث، قدمنا أسلوب لجدولة المهام المعتمد على الثقة، الذي يعتبر مفيدًا بشكل خاص لمهام البيانات الكبيرة. الفكرة الرئيسية هي استنتاج قيمة الثقة لكل عقدة معالجة بناءً على أدائها الأساسي، ثم تحديد أولويات المهام بناءً على متطلبات الموارد و تكاليفها. يقوم الحل المقترح بمنح المهام بطريقة ذكية للعقد المناسبة لتقليل وقت خدمة المهام وتكلفة معالجتها. وبالمقارنة مع خوارزميات الجدولة المستخدمة في الدراسة المرجعية الثالثة [7] فإن الحل المقترح يقلل وقت الخدمة بنسبة تصل إلى ٥٩% مقارنة بـ SJF ، و ٤٨٪ مقارنة بـ RR ، و ٤٠٪ مقارنة بالطرق المحسنة لل PSO وتحت نسب مختلفة من العقد غير الموثوقة. وأيضاً، يقلل بشكل كبير وقت التنفيذ مقارنةً بالحلول المدروسة.

يمكن تطبيق النتائج المقدمة في هذه الورقة على مشاكل أخرى. ويمكن ذلك من خلال ضبط المقاييس المناسبة في صياغة المشكلة والحل، وكذلك في بيئة التجربة. في الواقع، يمكن توسيع نموذج الثقة المقترح إلى بيئات أخرى بما في ذلك الحوسبة السحابية والإنترنت من الأشياء والحوسبة المتوازية وغيرها.

١١. قائمة المراجع :

[١] S. Basu, M. Karuppiah, K. Selvakumar, K.-C. Li, S. H. Islam, M. M. Hassan, M. Z. A. Bhuiyan, An intelligent/cognitive model of task scheduling for iot applications in cloud computing environment, Future Generation Computer Systems–٢٥٤ (٢٠١٨) ٨٨ ٨٢٠ .٢٦١

[2] G. Zhong-wen, Z. Kai, The research on cloud computing resource scheduling method based on time cost-trust model, in: Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on, IEEE, 2012, pp. 939–942.

[3] Statistical Characterization of Business-Critical Workloads Hosted in Cloud Datacenters, Siqi Shen, Vincent van Beek, Alexandru Iosup, ٢٠١٦

[4] Big Data Analytics in Support of the Decision Making Process, Nada Elgendy & Ahmed Elragal, 2016

[5] P. GaneshKumar A. S. Sofia **.**Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using nsga-ii .Journal of Network and Systems Management.٢٠١٨ ،٢٦ .

[6] G. Zeng, D. Tang, J. Yao W. Wang **.**Cloud-dls: Dynamic trusted scheduling for cloud computing .Expert Systems with Applications.٢٠١٨ ،٣٩ .

[7] B. M. H. Zade, M. M. Javidi N. Mansouri **.**Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory .Computers & Industrial Engineering.٢٠١٩ ،١٣٠ .

[8] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, M. U. Chowdhury, An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments, Neural Computing and Applications (2019) 1–11.

[9] X. Peng Y. Yang **.**Trust-based scheduling strategy for workflow applications in cloud environment .Eighth International Conference on, IEEE.٢٠١٦ .

[10] P. Poongodi D. Sumathi .An Improved Scheduling Strategy in Cloud Using Trust Based Mechanism .World Academy of Science, Engineering and Technology International Journal of Computer and Systems Engineering.٢٠١٥ .

[11] Developing an XML Schema for the Standard System Data Dictionary, Dr. Eng. Maher Ibrahim, Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (7) 2022

[12] A Data-Aware Remote Procedure Call Method for Big Data Systems, Jin Wang, Yaqiong Yang, Jingyu Zhang, Xiaofeng Yu, Osama Alfarraj and Amr Tolba, Comput Syst Sci & Eng (2020)

[13] O. A.Wahab, J. Bentahar, H. Otrok, A. Mourad, Optimal load distribution for the detection of vm-based ddos attacks in the cloud, IEEE Transactions on Services Computing DOI: 10.1109/TSC.2017.2694426 (2017).

[14] D. Yi, X. Li, J. S. Liu, Bayesian aggregation of rank data with covariates and heterogeneous rankers, arXiv preprint arXiv:1607.06051.(2018)

[15] R. J. Hyndman, Y. Fan, Sample quantiles in statistical packages, The American Statistician 50 (4) (1996) 361–365.

[16] R. R. Yager .Categorization in multi-criteria decision making .*Information Sci* . .٢٠١٨