مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

Performance evaluation of task scheduling in cloud computing

Dr. Kassem Kablan*
Hazem Deeb **

(Received 8 / 8 / 2017. Accepted 2 / 11 / 2017)

□ ABSTRACT □

Cloud computing infrastructure is suitable for meeting computational needs of large task sizes. Optimal scheduling of tasks in cloud computing environment has been proved to be an NP-complete problem, hence the need for the application of heuristic methods.

Several algorithms have been developed and used in addressing this problem, but choosing the appropriate algorithm for solving task assignment problem of a particular nature is difficult since the methods are developed under different assumptions.

Therefore, Four rule based algorithms are implemented and used to schedule autonomous tasks in homogeneous environment with the aim of comparing their performance in terms of cost, degree of imbalance, makespan, throughput , resource utilization and Quality of Service.

Optimized First Come First Served algorithm (OFCFS), Minimum Completion Time algorithm (MCT), Sufferage algorithm and Inter Cloud Scheduling with Priority using PTC Algorithm (ICSPPTC) are the heuristic algorithms considered for the performance evaluation and analysis of task scheduling in cloud computing.

Keywords : Cloud Computing , Task Scheduling , homogeneous environment , performance evaluation .

* **Assistant Professor, Department of System and Computing Network Engineering , Faculty of Informatics, Engineering, Tishreen University, Lattakia, Syria.**
****Postgraduate Student, Department of System and Computing Network Engineering , Faculty of Informatics, Engineering, Tishreen University, Lattakia, Syria.**

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (6) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (6) 2017

تقييم أداء خوارزميات جدولة المهام في الحوسبة السحابية

د. قاسم قبلان*
حازم ديب**

(تاريخ الإيداع 8 / 8 / 2017. قُبِل للنشر في 2/ 1 / 2017)

□ ملخّص □

تعتبر البنية التحتية للحوسبة السحابية مناسبة لتلبية الاحتياجات من الموارد الحاسوبية و التي تطلبها المهام الكبيرة . ولقد ثبت أن عملية جدولة المهام في بيئة الحوسبة السحابية بشكل أمثلي هي مشكلة من تعقيد NP–Complete . ومن هنا برزت الحاجة إلى تطبيق أساليب الاستدلال .

تم اقتراح عدد من خوارزميات الجدولة واستخدامها في معالجة هذه المشكلة، ولكن اختيار الخوارزمية المناسبة لحل مشكلة جدولة مهمات مختلفة لكل منها طبيعة معينة يعتبر أمراً صعباً و ذلك لأن تلك الخوارزميات يتم تطويرها في إطار ظروف و بيئات مختلفة.

لذلك تم تطبيق واستخدام أربع خوارزميات لجدولة المهام المستقلة في بيئة متجانسة بهدف مقارنة أدائها من حيث الكلفة Cost، ودرجة عدم التوازن Degree of Imbalance ، و زمن التنفيذ الكلي لمجموعة من المهام Makespan ، و الإنتاجية Throughput، واستخدام الموارد Resource Utilization وجودة الخدمة Quality of Service .

خوارزمية القادم أولاً يُخدَّم أولاً المحسنة Optimized FCFS (OFCFS) و خوارزمية زمن الانتهاء الأدنى Minimum Completion Time (MCT) و خوارزمية المعاناة (Sufferage) و خوارزمية الجدولة مع الأولوية بالاعتماد على قوة الاختيار الثنائي Inter Cloud Scheduling with Priority using PTC Algorithm (ICSPPTC) هي الخوارزميات التي تمت مناقشتها من أجل تقييم أداء و تحليل عملية جدولة المهام في الحوسبة السحابية .

الكلمات المفتاحية : الحوسبة السحابية ، جدولة المهام، البيئة المتجانسة ، تقييم الأداء .

*مدرس – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.
** طالب دراسات عليا(ماجستير) –قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين –اللاذقية – سورية.

مقدمة :

أصبحت الحوسبة السحابية من أهم المجالات البحثية العلمية التي يتطرق إليها الباحثون في حقلي تكنولوجيا المعلومات و الاتصالات.

باستخدام تقنيات الحوسبة السحابية يستطيع المستخدمون الوصول إلى أي نوع من الموارد الحاسوبية التي يحتاجونها و في أي وقت و ذلك دون الحاجة إلى شراء تلك الموارد . كما يمكنهم الوصول إلى أي نوع من التطبيقات التي يوفرها مزودو خدمة الحوسبة السحابية و ذلك حسب نظام الدفع حسب الاستخدام و الذي يتيح للمستخدمين بأن يدفعوا فقط لقاء الموارد أو التطبيقات التي يقومون باستخدامها .

هناك العديد من الفوائد التي تقدمها بيئة الحوسبة السحابية مثل التكلفة المنخفضة و الثبات و قابلية التوسع و سهولة الصيانة [3-1] .

خوارزميات جدولة المهام لها تأثير مباشر على سرعة تنفيذ المهام التي يطلبها المستخدمون و أيضاً على كفاءة الاستخدام الفعال للموارد ، حيث ثبت أن جدولة المهام بشكل أمثلي في بيئة الحوسبة السحابية من تعقيد NP-Complete [5-4] . خوارزميات جدولة المهام المستخدمة حالياً في بيئة الحوسبة السحابية من الصعب مقارنتها و تحديد أي منها هي الخوارزمية الأفضل و ذلك بسبب اختلاف أداء كل منها باختلاف البيئات و السيناريوهات المفروضة ، بالإضافة إلى تفاوت أداء هذه الخوارزميات و ذلك تبعاً لطبيعة العامل التي تتم دراسته.

تم في هذا البحث اختيار أربع خوارزميات أثبتت كفاءتها في نظام الحوسبة السحابية و هي خوارزمية القادم أولاً يُخدَّم أولاً المحسنة (OFCFS) و خوارزمية زمن الانتهاء الأدنى (MCT) و خوارزمية المعاناة (Sufferage) و خوارزمية الجدولة مع الأولوية بالاعتماد على قوة الاختيار الثنائي (ICSPPTC) . هذه الخوارزميات تم تطبيقها في بيئة متجانسة و هي البيئة التي تكون فيها خصائص جميع الآلات الافتراضية التي يتم إنشاؤها ثابتة و لا تتغير بتغير نوع المهمة التي يتم إسنادها إلى هذه الآلة و تم تحقيق هذه العملية باستخدام محاكي Cloudsim [6] .

أهمية البحث و أهدافه :

الحوسبة السحابية تتطور يوما بعد يوم و تواجه عدداً من التحديات لعل أهمها هي كيفية جدولة الطلبات الواردة. و يقصد بالجدولة مجموعة السياسات التي تحدد أولوية تنفيذ الطلبات الواردة (أي الطلبات سيتم تخديمها أولاً). و إلى يومنا هذا لم يتم تطوير تقنية الحوسبة السحابية بشكل كامل بعد، فما زالت هناك بعض النقاط التي تحتاج للتركيز عليها ومن أهمها عملية جدولة المهام حيث تعد جدولة المهام هي المشكلة الرئيسية في الحوسبة السحابية .

لذلك كان الهدف الرئيسي من هذا البحث هو المقارنة بين أداء خوارزميات جدولة المهام المستخدمة و ذلك بهدف استنتاج خوارزمية الجدولة الأفضل في بيئة الحوسبة السحابية.

طرائق البحث و مواده :

تم في هذا البحث التعرف على عملية جدولة المهام والتي تعني تحديد الترتيب المناسب لتنفيذ الطلبات الواردة و تعرفنا أيضاً على نموذج الجدولة في مراكز بيانات الحوسبة السحابية و مراحل عملية الجدولة بالإضافة إلى تصنيف خوارزميات الجدولة بين خوارزميات ساكنة و ديناميكية و شفعية و لاشفعية. كما تم التطرق إلى أهمية عملية المحاكاة

في الحصول على نتائج قريبة من النتائج الحقيقية و بتكاليف أقل بكثير و تطرقنا إلى المحاكي Cloudsim المستخدم في تحقيق عملية المحاكاة و البنية التحتية له و السيناريو المقترح في تقييم أداء خوارزميات الجدولة المدروسة والتجارب و النتائج التي حصلنا عليها.

الأعمال السابقة :

لقد تطرقت العديد من الأبحاث إلى خوارزميات جدولة المهام المختلفة و تم اقتراح العديد من الخوارزميات ليتم استخدامها في بيئة الحوسبة السحابية . حيث تعرفنا في [7] على خوارزمية القادم أولاً يخدم أولاً First Come First Served (FCFS) والتي تقوم فكرتها على عملية تجميع الطلبات الواردة في رتل و ذلك حتى تصبح الموارد متاحة و عندما تصبح هذه الموارد متاحة يتم تخصيص هذا المورد للمهام و ذلك تبعاً لزمن وصولها أي يتم تخديم أو تنفيذ المهمة الأقدم في الرتل .

عرف [8] خوارزمية زمن التنفيذ الأدنى Minimum Execution Time(MET) والتي تعتمد على إسناد المهمة إلى الآلة الافتراضية أو المورد الذي يقوم بتنفيذ المهمة ضمن أقل زمن . فهذه الخوارزمية تختار الآلة الافتراضية (Virtual Machine) الأفضل للتنفيذ و لكنها لا تأخذ بعين الاعتبار حالة المورد و توافره في اللحظة التي تتم فيها عملية الجدولة .

تم في [9] اقتراح خوارزمية زمن الانتهاء الأدنى Minimum Completion Time (MCT) و التي تعتمد على إسناد المهمة إلى الآلة الافتراضية أو المورد المتاح في تلك اللحظة والذي يقوم بتنفيذ المهمة ضمن أقل زمن حيث أن هذه الخوارزمية تأخذ بعين الاعتبار معايير توزيع الحمولة على جميع الآلات الافتراضية في لحظة الجدولة وذلك لأنها تقوم بحساب أقل زمن للتنفيذ من ضمن الموارد المتوافرة فقط مما ساهم في تحقيق مبدأ موازنة الحمولة .

تم التطرق في [10] إلى خوارزمية Min–Min التي تعتمد على اختيار المهمة الأقصر أي التي تملك أقصر زمن انتهاء من بين جميع المهام المتوافرة و تخصيصها إلى الآلة التي تقوم بتنفيذ هذه المهمة بأقل زمن و لكن هذه الخوارزمية تقوم بجدولة المهام بغض النظر عن أحمال الآلات الافتراضية قبل اتخاذ قرار الجدولة حيث تقوم بإسناد المهام الصغيرة إلى الآلات الأسرع مما قد يسبب حالة حرمان للمهام الكبيرة .

تم التعرف على خوارزمية Max–Min في [11] والتي تشبه إلى حد كبير خوارزمية Min–Min و لكنها تختار المهمة الأكبر أو الأطول أي التي تملك أكبر زمن انتهاء من بين جميع المهام المتوافرة و تقوم بتخصيص هذه المهمة إلى الآلة التي تقوم بتنفيذها بأقل زمن . ولكن كما خوارزمية Min–Min فإنها تقوم بجدولة المهام بغض النظر عن أحمال الآلات الافتراضية قبل اتخاذ قرارات الجدولة و لكن هنا قد تعاني المهام الصغيرة من الحرمان و ذلك لأن الاولوية في هذه الخوارزمية هي لتنفيذ المهام الأكبر .

تم اقتراح خوارزمية القادم أولاً يخدم أولاً المحسنة Optimized FCFS Algorithm في [12] و التي تعمل على تلافي السلبيات الموجودة في الخوارزمية التقليدية حيث تنص هذه الخوارزمية على أن الطلبات الواردة يتم ترتيبها في رتل و فقاً لزمن وصولها و يتم تخديمها على هذا الأساس و لكن في حال الوصول إلى طلب لا يمكن تخديمه مؤقتاً لعدم توفر الموارد المتاحة يتم تخصيص جزء من الموارد لهذا الطلب و الانتقال إلى الطلبات التالية في الرتل في محاولة لتنفيذها.

تم التطرق إلى خوارزمية المعاناة Sufferage Algorithm في [13] ، حيث تبدأ هذه الخوارزمية عملها بحساب زمن الانتهاء الكلي MCT لكل الأعمال الواردة إليها على مختلف الآلات المتوافرة. و بعد الانتهاء تأخذ أقل

زمن انتهاء و ثاني أقل زمن انتهاء لكل عمل من الأعمال الواردة و الفرق بين هاتين القيمتين أي ناتج طرح ثاني أفضل زمن انتهاء من أفضل زمن انتهاء ينتج قيمة تدعى قيمة المعاناة لهذا العمل . بعدها يتم حساب قيمة المعاناة لكل الأعمال الموجودة لدينا . و في الخطوة التالية يتم اختيار العمل ذو قيمة المعاناة الأكبر و تخصيصه إلى المورد الذي ينفذه بأقل زمن انتهاء .

و أخيراً اقترح [14] خوارزمية الجدولة مع الأولوية بالاعتماد على قوة الاختيار الثنائي Inter Cloud Scheduling with Priority using PTC Algorithm (ICSPPTC) و التي تهدف إلى إعطاء زمن استجابة أفضل للطلبات ذات الأولوية بالمقارنة إلى الطلبات بدون أولوية.

تقوم فكرة هذه الخوارزمية على استبدال الرتل المركزي الوحيد الموجود ضمن خوارزمية الانضمام إلى الرتل الأقصر برتلين أحدهما للطلبات ذات الأولوية و الآخر للطلبات بدون أولوية . و لذا فإن الطلبات ذات الأولوية الواردة إلى مزود الخدمة يتم وضعها ضمن رتل الطلبات ذات الأولوية و الطلبات بدون أولوية و التي ترد إلى مزود الخدمة يتم وضعها ضمن رتل الطلبات بغير أولوية . و يتم اختيار الطلبات المراد تخديمها من هذين الرتلين بحيث تضمن هذه الخوارزمية زمن استجابة أفضل للطلبات ذات الأولوية و بدون أن يتأثر زمن استجابة الطلبات بدون أولوية بشكل ملحوظ . حيث تبدأ هذه الخوارزمية عملها باختيار طلبين من رتل الطلبات ذات الأولوية لتنفيذهما و بعدهما يتم اختيار عمل من رتل الطلبات التي لا تملك أولوية و تستمر الخوارزمية بعملها بشكل مشابه مما يعطي نتائج أفضل للأعمال التي تملك أولوية دون ان تتأثر الأعمال التي لا تملك أولوية بشكل كبير .

جدولة المهام [15]:

بازدياد أعداد مستخدمي الحوسبة السحابية فإن المهام التي تحتاج إلى جدولة تزداد بشكل مطرد لذلك يوجد حاجة ملحة إلى وجود خوارزميات جدولة أفضل تعمل على نظام الحوسبة السحابية .

عملية جدولة المهام هي العملية التي يتم فيها إسناد المهمة الواردة إلى المورد المناسب . حتى اللحظة لا يوجد خوارزمية مثالية من كافة الجوانب . و على نطاق آخر فإن الخوارزمية الأفضل في الجدولة هي الخوارزمية التي تقدم حلولاً مناسبة لكلا الطرفين سواء كان طرف المهمة أو طرف المورد من حيث الزمن اللازم لتنفيذ المهمة و الذي يعتمد على آلية عمل هذه الخوارزمية .

1 نموذج الجدولة في مراكز بيانات الحوسبة السحابية :

عملية الجدولة في الحوسبة السحابية تملك عدة مكونات و أهمها :

1) مجدول المهام Job Scheduler : هو المكون الأهم في عملية جدولة المهام في الحوسبة السحابية فهو يحدد ترتيب تنفيذ المهام الواردة المنتظرة في رتل المهام .

2) رتل الانتظار Job Waiting Queue : هو الرتل الذي تنتظر فيه المهام الواردة تخصيصها إلى آلة افتراضية أو مورد معين و ذلك من أجل تنفيذها .

3) المهمة الواصلة Job Arriving Process : هي العملية أو الطلب الواصل لنظام الحوسبة السحابية من قبل الزبون.

2 مراحل عملية الجدولة :

يمكن تقسيم عملية الجدولة إلى ثلاث مراحل أساسية :

1) **اكتشاف و فلترة الموارد Resource Discovering and Filtering** : في هذه المرحلة يقوم المجدول باستكشاف الموارد الموجودة في الشبكة و التعرف عليها و يقوم بجمع المعلومات عن حالتها و عن خصائص كل مورد من تلك الموارد .

2) **اختيار الموارد Resource Selection** :إن عملية استهداف مورد معين و إسناده لمهمة ما يتعلق بالعديد من المتحولات الخاصة بالمورد نفسه و بالمهمة المطلوب إنجازها.

3) **إسناد المهمات Task Submission** : و في هذه المرحلة يتم اتخاذ القرارات أي يتم تحديد أن المهمة X سوف يتم إسنادها إلى المورد Y .

5-3 تصنيف خوارزميات الجدولة :

1) **جدولة شفعية Preemptive** : من الممكن أن تتم مقاطعة تنفيذ مهمة معينة في أي وقت و ذلك حتى بعد إسنادها إلى الآلة الافتراضية من أجل تحرير المورد و إسناده إلى مهمة أخرى .

2) **جدولة لا شفعية Non Preemptive** : عندما تبدأ أي مهمة بالتنفيذ على آلة افتراضية أو مورد فإنه سيتم متابعة تنفيذ هذه المهمة حتى النهاية و لن تتم مقاطعتها مهما حصل .

3) **جدولة ساكنة Static** : حيث يتم اتخاذ قرارات الجدولة بالاعتماد على المتحولات الثابتة و تتخذ هذه القرارات قبل إرسال أي مهمة للتنفيذ .

4) **جدولة متغيرة Dynamic** : حيث يتم اتخاذ قرارات الجدولة بالاعتماد على المتحولات المتغيرة و التي قد تتغير في أي لحظة أثناء التنفيذ و هذا سيزيد من فعالية استخدام الموارد بالشكل الأمثل .

المحاكاة و تقنياتها :

المحاكاة هي عبارة عن عملية تمثيل للبيئة الحقيقية و التي تم استخدامها من أجل توفير الوقت و تذليل الصعوبات التي تواجهه عند القيام بتجربته على بيئة حقيقية و تضمن عملية المحاكاة الحصول على نتائج قريبة كثيراً من النتائج الحقيقية و بتكاليف أقل بكثير و بشكل أسهل .

1 محاكي Cloud Simulator (Cloudsim) [16]:

يستطيع المستخدمون الوصول إلى الموارد المشتركة من خلال الاستفادة من منصة السحابة العامة المتاحة. ومع ذلك فإن الوصول إلى بيئة سحابة حقيقية أو سحابة عامة ليست دائماً في متناول اليد . و لذلك بدلاً من الوصول إلى البيئة الحقيقية فإن برنامج المحاكاة يمكن أن يسهل عملية إجراء التجارب المطلوبة في سبيل الحصول على النتائج المرجوة .

بيئة المحاكاة تسمح للعملاء أو المستخدمين بتقييم أنواع مختلفة من الميزات وفق توزيعات الحمل المختلفة. محاكي Cloudsim هو أداة محاكاة تم استخدامها على نطاق واسع لمحاكاة التطبيقات السحابية وخوارزميات الجدولة ذات الصلة بالحوسبة السحابية .

تتك عملية الجدولة في هذا المحاكي على مستويين مختلفين، المستوى الأول بين المضيفين والآلات الافتراضية حيث يقوم كل مضيف بإنشاء عدد من الآلات الافتراضية التي يحتاجها ، و المستوى الثاني بين الآلات الافتراضية و الأعمال الواردة أي التطبيقات أو العمليات التي سيتم تنفيذها عبر السحابة . عادة، يعرف الأول باسم جدولة الآلات الافتراضية والثاني هو المعروف بجدولة الأعمال الواردة .

2 البنية التحتية لمحاكي Cloudsim :

إن البنية التحتية للمحاكي Cloudsim تتألف من أربع طبقات :

1) **طبقة موارد السحابة Cloud Resources** : في الجزء السفلي تدار الموارد السحابية (المضيفين و مراكز البيانات) و أثناء تنفيذ عملية المحاكاة يتم إنشاء هذه الكيانات الأساسية وتنفيذها.

2) **طبقة خدمات السحابة Cloud Services** : توجد طبقة الخدمات التي تقدمها السحابة فوق طبقة موارد السحابة و تحوي هذه الطبقة خدمات مثل تخصيص وحدة المعالجة المركزية والذاكرة و مساحات التخزين وعرض الحزمة.

3) **طبقة خدمات الآلات الافتراضية Virtual Machines Services** : و هي الطبقة المسؤولة عن إدارة الآلات الافتراضية مثل عدد هذه الآلات و إعداداتها .

4) **طبقة هيكل واجهة المستخدم User Interface Structure** : و هي الطبقة المسؤولة عن طلبات المستخدمين من حيث الحجم و المتطلبات .

القسم العملي :

تم في مجموعة التجارب التي تمت دراستها تقييم أداء أربع خوارزميات أثبتت كفاءتها في نظام الحوسبة السحابية و هذه الخوارزميات هي خوارزمية القادم أولاً يخدم أولاً المحسنة Optimized FCFS algorithm و خوارزمية زمن الانتهاء الأدنى Minimum Completion Time وخوارزمية المعاناة Sufferage و خوارزمية الجدولة مع الأولوية بالاعتماد على قوة الاختيار الثنائي Inter Cloud Scheduling with Priority using PTC Algorithm ، من حيث عدة عوامل و هي الكلفة و درجة عدم التوازن و زمن التنفيذ الكلي لمجموعة من المهام و خرج النظام الكلي و معدل استخدام الموارد و عامل جودة الخدمة و ذلك بهدف الوصول إلى معرفة الخوارزمية الأفضل الواجب تطبيقها في نظام الحوسبة السحابية حيث تمت المقارنة ضمن نفس الظروف لجميع الخوارزميات المدروسة و لكن مع تغيير عدد الأعمال الواردة لتقييم أداء كل خوارزمية عند الأحمال الخفيفة و المتوسطة و الثقيلة.

1 السيناريو المقترح [17]:

تم دراسة هذه الخوارزميات في المحاكي Cloudsim3.0 بناءً على عدة عوامل و كانت هذه المتحولات هي الكلفة و زمن التنفيذ الكلي و الإنتاجية و درجة عدم التوازن و معيار جودة الخدمة و تمت عملية المقارنة بالاعتماد على قيم المتحولات المبينة في الجدول (1).

في البيئة المتجانسة تم تثبيت و توحيد الخصائص لكل الآلات الافتراضية التي تم إنشاؤها في النظام و ذلك عند تقييم أداء خوارزميات الجدولة المقترح دراستها و لكن مع تغيير عدد الأعمال الواردة من قبل الزبائن حيث تم دراسة كل من هذه الخوارزميات تحت حمل أو عدد متغير من الأعمال الواردة وفقاً لكل عامل من هذه العوامل .

الجدول (1) قيم المتحولات المستخدمة في دراسة الخوارزميات

التسلسل	النوع	المتحولات	القيم
1	المستخدم User	عدد المستخدمين	5
2	الأعمال الواردة Cloudlet	عدد الأعمال الواردة	1000 – 100

		طول الأعمال الواردة	2000 تعليمة
3	المضيف Host	عدد المضيفين	2
		RAM	2048 MB
		Storage	1 TB
		Bandwidth	10 GB
4	الآلات الافتراضية Virtual Machines	عدد الآلات الافتراضية	15
		RAM	512 MB
		Bandwidth	1 GB
		MIPS	1000
		عدد المعالجات	2
5	مراكز البيانات Data Centers	عدد مراكز البيانات	2

2 عوامل تقييم الأداء Performance Metrics :

تقييم الأداء هذا قام على قياس عدة عوامل توضح أداء هذه الخوارزميات و كانت هذه العوامل هي الكلفة و زمن التنفيذ الكلي و الإنتاجية و درجة عدم التوازن و معيار جودة الخدمة ، و سوف نشرح كل واحد من هذه العوامل :

الكلفة Cost :

الكلفة تعني المبلغ الكلي الذي سوف يدفعه المستخدم لمزود خدمة الحوسبة السحابية لقاء استخدامه لموارد الحوسبة السحابية و لنفترض أن الكلفة تختلف من آلة افتراضية لأخرى و ذلك حسب عامل الوقت و خصائص الآلة الافتراضية المحجوزة ، و تعطى قيمة الكلفة بالقانون التالي [18] :

$$Cost = \sum_{i=1}^{n} task^i (Ci * Ti)$$

حيث Ci تمثل كلفة استخدام الآلة الافتراضية التي يتم تنفيذ المهمة i عليها .

و Ti تمثل زمن تنفيذ المهمة i على الآلة الافتراضية المخصصة لها .

درجة عدم التوازن Degree of Imbalance (DI) :

يصف هذا المعيار كيفية توزيع الحمل بين الآلات الافتراضية المختلفة المتوافرة من حيث كفاءة التنفيذ. و كلما انخفض هذا المعيار كلما كان أداء الخوارزمية أفضل و كلما كان مفهوم موازنة الحمولة محققاً بشكل أمثلي .

و يعطى هذا المعيار بالقانون [18]:

$$DI = \frac{T_{\max} - T_{\min}}{T_{avg}}$$

حيث Tmax يمثل الزمن الأكبر اللازم لتنفيذ مهمة معينة على آلتها الافتراضية .

و Tmin يمثل الزمن الأصغر اللازم لتنفيذ مهمة معينة على آلتها الافتراضية .

و Tavg يمثل متوسط الزمن اللازم لتنفيذ مهمة معينة على آلتها الافتراضية و يتم حسابه عن طريق تقسيم المجموع الكلي لزمن تنفيذ كل مهمة على عدد المهام الكلي .

زمن التنفيذ الكلي لمجموعة من المهام Makespan :

يتم معرفة هذا الزمن الذي يعبر عن زمن التنفيذ الكلي لمجموعة من المهام و ذلك عن طريق حساب زمن الانتهاء الكلي لآخر مهمة يتم تنفيذها و ذلك بعد جدولة جميع المهام الواردة .

و يعطى زمن التنفيذ الكلي بالقانون [18]:

$$Makespan = max_{task^i}(Fnh_{Time})$$

حيث أن FnhTime يمثل زمن الانتهاء للمهمة i على آلتها الافتراضية .

الإنتاجية Throughput :

يعتمد هذا المعيار بشكل أساسي على عدد المهام التي تم تنفيذها بنجاح في نظام الحوسبة السحابية و ذلك حسب الخوارزمية المقترحة .

تعطى الإنتاجية بالقانون [18]:

$$Throughput = \sum_{task^i}(Exe_{Time})$$

حيث ExeTime يمثل زمن تنفيذ المهمة i على آلتها الافتراضية .

معدل استخدام الموارد Resource Utilization :

معدل استخدام الموارد يمثل المعدل بين الانشغال الكلي للآلة الافتراضية بالنسبة لزمن التنفيذ الكلي لجميع المهام Makespan .

يعطى من خلال القانون التالي :

$$\text{معدل استخدام كل آلة افتراضية} = \frac{\text{الزمن النهائي لانتهاء انشغال العمل في ال VM}}{\text{زمن التنفيذ الكلي}} * 100$$

$$\text{معدل الاستخدام الكلي} = \frac{\text{مجموع معدل استخدام كل آلة افتراضية}}{\text{عدد الآلات الافتراضية}}$$

جودة الخدمة Quality Of Service (QOS) :

عامل جودة الخدمة هو العامل الرئيسي الذي يساعد في تقييم مدى نجاح الخوارزمية في تنفيذ المهام المطلوبة منها و حساب حالات الفشل في تنفيذ المهام والذي وفقها سوف نقيم أداء كل خوارزمية من الخوارزميات المدروسة .

يعطى معيار جودة الخدمة بالقانون التالي :

$$QoS = \frac{No.of\ Tasks\ succeeded}{No.of\ all\ Tasks} * 100$$

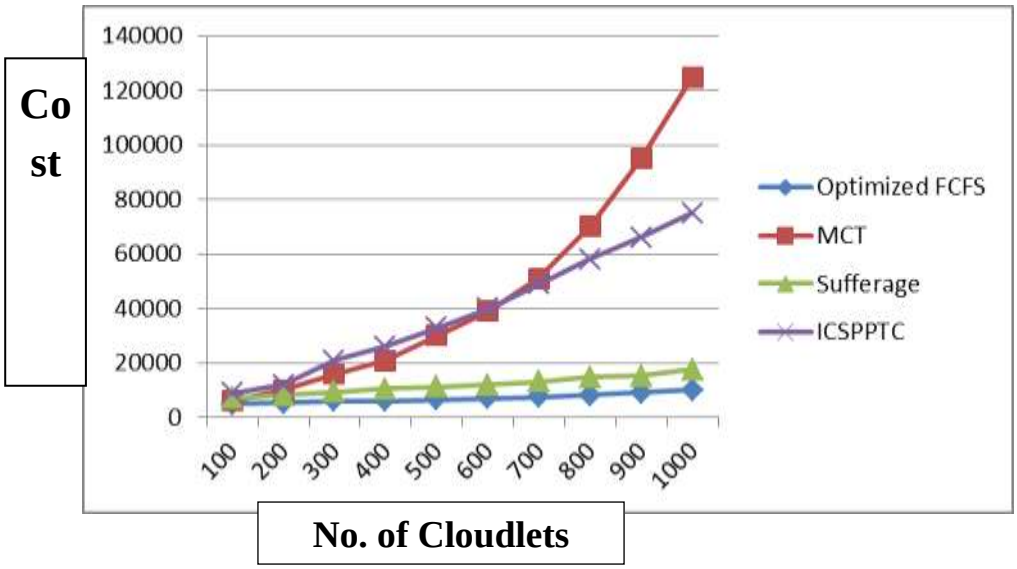
حيث أن No. of Tasks succeeded يمثل عدد المهام التي تم تنفيذها بنجاح في النظام .

و No. of all Tasks يمثل العدد الكلي للمهام التي تم إسنادها إلى جميع الموارد الافتراضية في نظام الحوسبة السحابية .

النتائج و المناقشة :

التجربة الأولى : من حيث الكلفة Cost :

نلاحظ من الشكل (1) أن المحور الأفقي يوضح عدد الأعمال الواردة إلى نظام الحوسبة السحابية و المحور العمودي يمثل الكلفة لجميع الخوارزميات المدروسة على اعتبار أن كلفة استخدام كل آلة افتراضية هو 0.90$ في الساعة و قد تم استخدام الدولار الأمريكي و ذلك بسبب الحاجة إلى التعامل بعملة موحدة بين جميع مزودات خدمة الحوسبة السحابية .



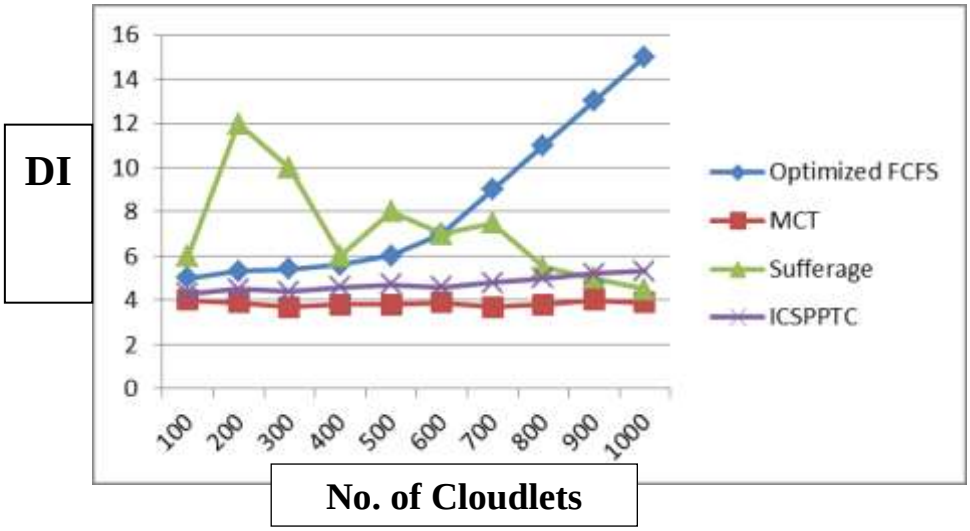
الشكل (1) نتائج مقارنة كل من الخوارزميات الأربعة من حيث عامل التكلفة Cost

و نستنتج من الشكل (1) أن كلاً من خوارزميتي MCT و ICSPPTC تعطيان أكبر كلفة في كل الحالات بالمقارنة مع خوارزميتي Optimized FCFS و Sufferage و ذلك لأن خوارزميتي MCT و ICSPPTC تقومان بإنجاز أكبر عدد ممكن من الأعمال الواردة لذا فإن الكلفة ستكون أكبر بالمقارنة مع خوارزميتي Optimized FCFS و Sufferage .

و في المرتبة الثالثة تأتي خوارزمية Sufferage التي تعطي كلفة أقل من خوارزميتي MCT و ICSPPTC في جميع الحالات و ذلك لأن عدد الأعمال المنجزة فيها أقل من الخوارزميتين السابقتين فبالتالي تكلفتها ستكون أقل .

و في المرتبة الأخيرة تأتي خوارزمية Optimized FCFS ، حيث تعتبر الأقل تكلفة في جميع الحالات من بين جميع الخوارزميات المدروسة و ذلك بسبب أنها تملك أقل عدد من المهام التي يتم تنفيذها بنجاح بالمقارنة مع باقي الخوارزميات .

التجربة الثانية : من حيث درجة عدم التوازن Degree of Imbalance (DI) :

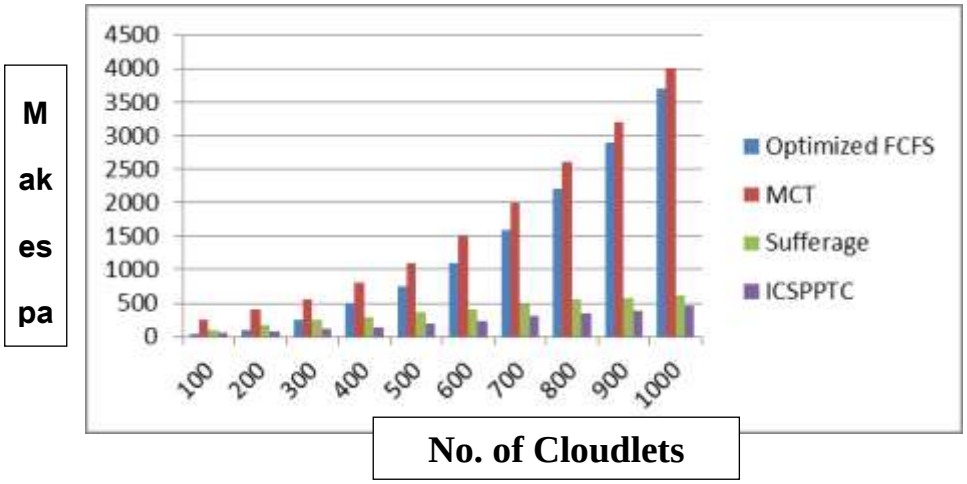


الشكل (2) نتائج مقارنة كل من الخوارزميات الأربعة من حيث درجة عدم التوازن Degree of Imbalance

نلاحظ من الشكل (2) أن خوارزمية MCT هي الأفضل في جميع الحالات من بين كل الخوارزميات المدروسة حيث نلاحظ أنها مستقرة بشكل كبير و لا تتأثر بازدياد عدد الأعمال الواردة

وتأتي في المرتبة الثانية خوارزمية ICSPPTC حيث تتراوح درجة عدم التوازن فيها بين القيمتين 4 و 5 و تعتبر هذه الخوارزمية مستقرة أكثر من خوارزميتي Sufferage و Optimized FCFS و التي يمكن اعتبارها خوارزميات غير مستقرة لأن الخوارزميتين MCT و ICSPPTC تراعي الأحمال السابقة للآلات الافتراضية على عكس خوارزميتي Sufferage و Optimized FCFS.

التجربة الثالثة : زمن التنفيذ الكلي لمجموعة من المهام Makespan :

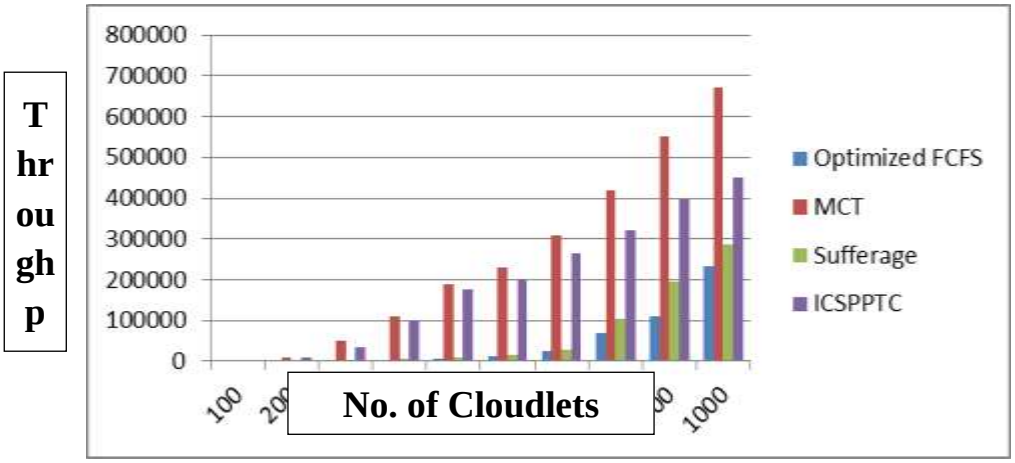


الشكل (3) نتائج مقارنة كل من الخوارزميات الأربعة من حيث زمن التنفيذ الكلي لمجموعة من المهام Makespan

نلاحظ من الشكل (3) أن خوارزمية ICSPPTC هي الأفضل من حيث زمن التنفيذ الكلي لمجموعة من المهام حيث تقوم باختيار العقدة الأقل حملاً من بين عقدتين عشوائيتين و في المرتبة الثانية تأتي خوارزمية Sufferage و

ذلك لأنها تقوم بإعطاء أولوية للأعمال ذات الفروق الكبيرة في زمن التنفيذ أما في المرتبة الثالثة تأتي خوارزمية Optimized FCFS و ذلك بسبب بساطتها حيث يتم اتخاذ قرارات الجدولة حسب زمن الوصول و في المرتبة الأخيرة تأتي خوارزمية MCT التي تعتبر الأسوأ من بين جميع الخوارزميات المدروسة من حيث زمن التنفيذ الكلي و ذلك بسبب الوقت الضائع الذي تستغرقه في حساب زمن الانتهاء الكلي للمهمة المطلوبة على جميع الآلات الافتراضية المتوافرة .

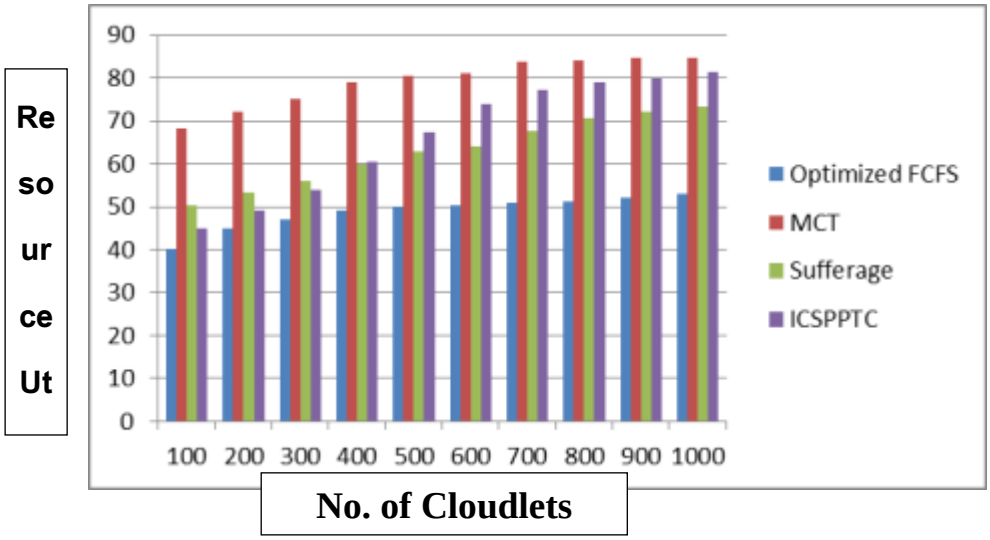
التجربة الرابعة : من حيث الإنتاجية Throughput :



الشكل (4) نتائج مقارنة كل من الخوارزميات الأربعة من حيث الإنتاجية Throughput

نلاحظ من الشكل (4) أن خوارزمية MCT هي الأفضل من حيث الإنتاجية و ذلك لأن عدد الأعمال المنفذة بنجاح في هذه الخوارزمية هو الأعلى من بين جميع الخوارزميات المدروسة و بعدها تأتي خوارزمية ICSPPTC لأن نسبة نجاح الأعمال المنفذة عند استخدام هذه الخوارزمية مرتفع أيضاً ومن ثم تأتي خوارزمية Sufferage و التي تعاني من فشل تنفيذ بعض الأعمال الواردة و ذلك بسبب أن هذه الخوارزمية لا تراعي الأحمال السابقة للآلات الافتراضية عند اتخاذ قرارات الجدولة مما يسبب في فشل تنفيذ بعض الأعمال وأخيراً تأتي خوارزمية Optimized FCFS التي تعتبر الأسوأ حسب عامل الإنتاجية و ذلك لأن هذه الخوارزمية لا تأخذ بعين الاعتبار الا معيار زمن الوصول و ذلك عند اتخاذ قرارات الجدولة و بالتالي هي لا تراعي أحمال الآلات و حالتها مما يسبب في حدوث حالات فشل كبيرة في تخديم الطلبات الواردة إلى مزود خدمة الحوسبة السحابية.

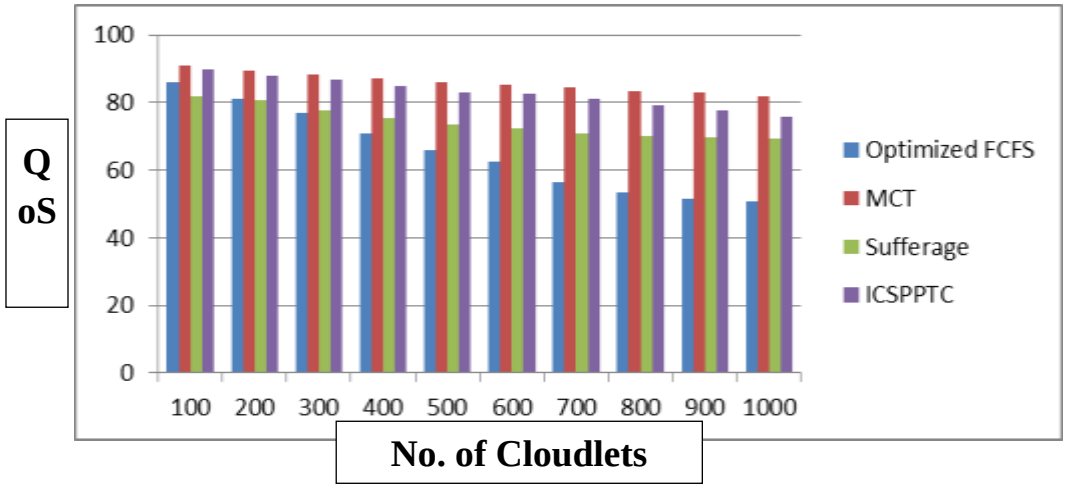
التجربة الخامسة : من حيث معدل استخدام الموارد Resource Utilization :



الشكل (5) نتائج مقارنة كل من الخوارزميات الأربعة من حيث معدل استخدام الموارد Resource Utilization

نستنتج من الشكل (5) أن خوارزمية MCT هي الأفضل من حيث معدل استخدام الموارد في كل الحالات بين جميع الخوارزميات السابقة و ذلك لأن فكرتها الأساسية تقوم على موازنة الحمولة بين جميع الآلات الافتراضية المتوافرة، بينما تأتي في المرتبة الثانية خوارزمية ICSPPTC لأن أغلب العقد تكون ذات أحمال متقاربة وفي المرتبة الثالثة تأتي خوارزمية Sufferage و تليها خوارزمية Optimized FCFS و يعتبر أداؤهما سيئاً من ناحية الاستخدام الفعال للموارد لأنها لا تراعي أبداً عملية موازنة الحمولة على الآلات الافتراضية المتوافرة.

التجربة السادسة : من حيث عامل جودة الخدمة Quality Of Service (QoS) :



الشكل (6) نتائج مقارنة كل من الخوارزميات الأربعة من حيث عامل جودة الخدمة Quality Of Service

نلاحظ من الشكل (6) أن المحور الأفقي يوضح عدد الأعمال الواردة إلى نظام الحوسبة السحابية والمحور العمودي يمثل عامل جودة الخدمة لجميع الخوارزميات المدروسة على اعتبار أن هذا العامل يقيس نسبة نجاح الأعمال في النظام المستخدم.

نستنتج من الشكل (6) أن خوارزمية MCT هي الأفضل من حيث عامل جودة الخدمة في كل الحالات بين جميع الخوارزميات السابقة، فعلى سبيل المثال عندما يكون عدد الأعمال الواردة 100 يكون عامل جودة الخدمة91% و عندما يزداد عدد الأعمال الواردة إلى 1000 يصبح عامل جودة الخدمة 81.98% و ذلك لأن احتمال الفشل في هذه الخوارزمية ضئيل حيث تقوم بتخصيص العمل الوارد إلى الآلة الافتراضية التي تقوم بإنهاء هذا العمل بأقل زمن ممكن.

تأتي في المرتبة الثانية خوارزمية ICSPPTC ، حيث عندما يكون عدد الأعمال الواردة 100 يكون عامل جودة الخدمة89.9% لأن هذه الخوارزمية تقوم باختيار العقدة الأقل حملاً من بين عقدتين عشوائيتين و عندما يزداد عدد الأعمال الواردة إلى 1000 يصبح عامل جودة الخدمة 75.6% و ذلك بسبب زيادة احتمال أن كلاً من العقدتين المختارتين بشكل عشوائي ذات أحمال ثقيلة و لكن لا تتجاوز نسبة الفشل في هذه الخوارزمية الربع تقريباً و هو ما يعتبر إلى يومنا هذا نسبة مقبولة من الخسارة .

تأتي في المرتبة الثالثة خوارزمية Sufferage و التي تعاني من فشل تنفيذ بعض الأعمال حيث يتم جدولة الأعمال الحرجة أولاً على الآلات الأكثر مناسبةً لها بينما تخصص الأعمال التي تجدول في النهاية إلى آلات افتراضية غير مناسبة لها و بالتالي يزيد احتمال فشل تنفيذها على تلك الآلات .

تأتي في المرتبة الأخيرة خوارزمية Optimized FCFS ، حيث تعتبر الأسوأ من بين جميع الخوارزميات المدروسة وذلك لأن هذه الخوارزمية لا تراعي أبداً أية معايير عند إسناد الأعمال الواردة إلى الآلات الافتراضية سوى معيار زمن الوصول و لا تأخذ بعين الاعتبار الأحمال السابقة على الآلات الافتراضية و لا حالة الآلة الافتراضية و بالتالي حوالي نصف الأعمال الواردة يفشل تنفيذها وهي نسبة غير مقبولة على الاطلاق بالنسبة لمزودي خدمات الحوسبة السحابية و حتى بالنسبة للزبائن ، لأن احتمال فشل تنفيذ المهمة التي يطلبونها من النظام تقارب 50% أي حوالي النصف .

نلاحظ من التجارب التي أجريناها أن خوارزمية MCT تعطي أفضل النتائج إلا عند دراسة عامل زمن التنفيذ الكلي بسبب الوقت المهدور من قبلها في قياس زمن الانتهاء على جميع الآلات الافتراضية المتاحة.

الاستنتاجات و التوصيات :

-لا يوجد خوارزمية يمكن اعتبارها الخوارزمية الأفضل بشكل مطلق ، فقد وجد أن جميع الخوارزميات تملك نقاط ضعف وفق بعض العوامل .

- إن خوارزمية زمن الانتهاء الأدنى هي الأفضل في معظم الحالات و لكنها الخوارزمية الأسوأ من حيث زمن التنفيذ الكلي .

-إن خوارزمية الجدولة مع الأولوية بالاعتماد على قوة الاختيار الثنائي تعطي نتائج تعتبر جيدة في جميع الحالات و حققت مع خوارزمية زمن الانتهاء الأدنى أداءً أفضل بكثير من أداء كل من خوارزميتي المعاناة و القادم أولاً يخدم أولاً المحسنة .

-إن خوارزمية المعاناة أعطت نتائج أفضل من خوارزمية القادم أولاً يخدم أولاً المحسنة و كان لها نتائج مقبولة بالنسبة لبعض العوامل .

-إن خوارزمية القادم أولاً يخدم أولاً المحسنة كانت الأسوأ من بين جميع الخوارزميات المدروسة و لكن لاحظنا أنها تعطي أداءً جيداً في حالة الأحمال الخفيفة .

الأعمال المستقبلية:

-اقتراح خوارزمية جدولة تعتمد على الاستفادة من نقاط القوة الموجودة في كل خوارزمية من الخوارزميات المقترحة و تلافي السلبيات الموجودة في كل من تلك الخوارزميات و تجميعها بهدف الوصول إلى خوارزمية تعطي نتائج أفضل من الخوارزميات الموجودة حتى الآن .

-اقتراح دراسة أداء كل من الخوارزميات الموجودة أو الخوارزمية الجديدة المقترحة في بيئة غير متجانسة و هي البيئة التي تتغير فيها خصائص الآلات الافتراضية التي يقوم نظام الحوسبة السحابية بإنشائها حيث يمكن أن تختلف قدرة المعالجة وعرض الحزمة والذاكرة بين آلة افتراضية و أخرى و ذلك تبعاً لطبيعة العمل الوارد و أولويته و حجمه .

المراجع:

[1] FOSTER I, ZHAO Y, RAICU I, LU S. *Cloud computing and grid computing 360-degree compared*. Grid Computing Environments Workshop, 2011 GCE'08; IEEE:2008.

[2] MELL P, GRANCE T. *The NIST definition of cloud computing.* National Institute of Standards and Technology.2012.

[3] RIMAL BP, CHOI E, LUMB I. *A taxonomy, survey, and issues of cloud computing ecosystems*. Cloud Computing: Springer; 2010.

[4] GORBENKO A, POPOV V. *Task-resource scheduling problem*. International Journal of Automation and Computing. 2014.

[5] ZHANG Q, CHENG L, BOUTABA R. *Cloud computing: state-of-the-art and research challenges.* Journal ofinternet services and applications. 2012.

[6] CALHERIOS RN, RANJAN R, BELOGLAZOV A, DE ROSE CA, BUYYA R. *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.* Software: Practice and Experience. 2011.

[7] LAKRA AV, YADAV DK. *Multi-objective tasks scheduling algorithm for cloud computing throughput optimization*. Procedia Computer Science. 2015.

[8] DU KIM H, KIM JS. *An online scheduling algorithm for grid computing systems. International Conferenceon Grid and Cooperative Computing*; Springer: 2011.

[9] BRAUN TD, SIEGEL HJ, BECK N . *A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems.* Journal of Parallel and Distributed computing. 2011.

[10] WANG G, YU HC. *Task Scheduling Algorithm Based on Improved Min-Min Algorithm in Cloud Computing Environment.* Applied Mechanics and Materials. 2013.

[11] Li X, MAO Y, XIAO X, ZHUANG Y. *An improved max-min task-scheduling algorithm for elastic cloud.* Computer,Consumer and Control (IS3C), 2014 International Symposium on; IEEE: 2014.

[12] ZUO L, SHU L, DONG S, ZHU C, HARA T. *Optimization of FCFS Based Resource Provisioning Algorithm for Cloud Computing*. Access, IEEE: 2015 .

[13] HAN H, DEYUI Q, ZHENG W, BIN F. *A Qos Guided task Scheduling Model in cloud computing environment.* Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on; IEEE: 2013.

[14] ZHU Y, LIANG H. *Inter-Cloud Scheduling Technique Using Power Of Two Choices*. Information Management, Innovation Management and Industrial Engineering (ICIII), 2016 6th International Conference on; IEEE: 2016.

[15] Z. QIAN, G. YUFEI, L. HONG and S. JIN, *"A Load Balancing Task Scheduling Algorithm based on Feedback Mechanism for Cloud Computing"*, International Journal of Grid and Distributed Computing, vol. 9, no. 4, pp. 41-52, 2016.

[16] TCHERMYKH A, LOZANO L, SCHWIEGELSHOHN U, BOURVY P, PECERO JE, NESMACHHNOW S, et al. *Online biobjective scheduling for IaaS clouds ensuring quality of service.* Journal of Grid Computing. 2016 .

[17] GANDOMI AH, YANG X-S, TALATHARTI S, ALAVI AH. *Cloud Computing CPU Allocation and Scheduling Algorithms Using CloudSim Simulator* . Newnes; 2016.

[18] BARQUET AL, TCHERNYKH A, YAHYAPOUR R. *Performance Evaluation of Infrastructure as Service Clouds with SLA Constraints.* ComputacioÂn y Sistemas. 2017.