مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (2) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (2) 2017

تحسين توافرية وأداء النظام الموزع بإعادة انتشار مكونات النظام

الدكتور محمد حجازية*
علي سليمان**

(تاريخ الإيداع 26 / 2 / 2017. قُبِل للنشر في 30/ 3 / 2017)

□ ملخّص □

استخدمت النظم الموزعة كثيراً في الآونة الأخيرة كبديل عن النظم المركزية نظراً لأدائها الأفضل عندما يزداد حجم النظام أو تزيد الحاجة إلى موارد المعالجة. رغم ذلك لازالت تعاني تلك النظم من مشاكل ضعف التوافرية عند حدوث تغيير في البنية التحتية التي تعمل فوقها مما أدى إلى ظهور مفهوم إعادة انتشار مكونات النظام الموزع. في هذا البحث تم تقديم طريقة لأخذ القرار بإعادة الانتشار الأفضل لبعض مكونات النظام الموزع عن طريق المراقبة الذاتية لحالة عقد النظام الموزع بالاستفادة من نظرية الأرتال، من ثم تقديم تقارير عن تلك الحالة إلى المكون المركزي، الذي يقوم حينئذٍ بأخذ قرار إعادة الانتشار إلى العقد المناسبة.

الكلمات المفتاحية: نظام موزع – إعادة الانتشار – نظرية الأرتال – نموذج المراقب – التوافرية.

* أستاذ مساعد– قسم هندسة الحاسبات والتحكم الآلي–كلية الهندسة الميكانيكية والكهربائية –جامعة تشرين –اللاذقية– سورية.
** طالب دراسات عليا(ماجستير)– قسم هندسة الحاسبات والتحكم الآلي –كلية الهندسة الميكانيكية والكهربائية–جامعة تشرين–اللاذقية– سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (2) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (2) 2017

Enhancing Availability and Performance of Distributed System via Redeployment of System Components

Dr. Mohammed Hijazieh*
Ali Sulaiman**

(Received 26 / 2 / 2017. Accepted 30 / 3 / 2017)

□ ABSTRACT □

Distributed Systems have been use too often recently as an alternative for central systems, due to their better performance when the size of the system or the need to the system resources becomes bigger. On the other hand, distributed systems are still fussing about the availability lack when a change of the infrastructure of the network take a place, which presents the approach of component redeployment in the distributed systems. In this research, we view a method for making decision about best component redeployment in a distributed system via self-monitoring for node status using the Queuing Theory, then handle reports to the central component, which make the choice about redeployment to the suitable node.

Key words: Distributed System, Redeployment, Queuing Theory, Observer Pattern, Availability.

***Assistant Professor, Department of Computer & Automatic Control, Faculty of Mechanical &Electrical Engineering, Tishreen University, Lattakia, Syria. University, Lattakia, Syria.**
****Postgraduate Student, department of Computer & Automatic Control, Faculty of Mechanical &Electrical Engineering, Tishreen University, Lattakia, Syria.**

مقدمة:

النظم الموزعة هي النظم التي تعمل على عدة أجهزة مرتبطة مع بعضها بشبكة حاسوبية وتظهر للمستخدم وكأنه يعمل على نظام مركزي واحد. من أهم مزايا النظم الموزعة هي التوافرية (Availability) أي عدم التعطل، فبدلاً من زيادة توافرية كل جهاز على حدى بتكلفة كبيرة يمكن استخدام نظام موزع مكون من عدة أجهزة (تُسمى عقد node) وعند توقف أحد العقد عن العمل تعمل عقدة أخرى، مما يسمح بالاستفادة من تجميع القدرة الحاسوبية من خلال توزيع العمل على عدة عقد أيضاً. كذلك تتيح النظم الموزعة قابلية التوسع (Scalability) حيث يمكننا في أي لحظة إضافة عقد جديدة إلى النظام الموزع يمكنها أن تساعد في تحسين الأداء الكلي.عند تطوير النظام الموزع يتم مراعاة إحدى النموذجين المعماريين: الند للند (Peer–to–Peer) أو زبون–مخدم (Client–Server)، وفي جميع المعماريات يجب مراعاة خاصية الشفافية (Transparency) التي تشعر المستخدم النهائي أنه يعمل على نظام مركزي واحد وليس على عدة مكونات تتواصل مع بعضها لتؤمن وظيفة ما.بعد الانتهاء من تطوير النظام الموزع نقوم بنشر مكوناته على البنية التحتية له. نشر مكونات النظم الموزعة (Deployment) هو توزيع مكونات النظام الموزع على مجموعة أجهزة عتادية (عقد) متصلة فيما بينها، بدون عملية نشر المكونات لا فائدة من توزيع النظام بل يكون من الأفضل جعله نظام مركزي واحد يعمل على نفس الجهاز بما يوفر من عمليات الاتصال بين المكونات.
يُعتبر نشر النظم الموزعة عملية معقدة حيث أن النشر الجيد يجب أن يراعي استمرار توافرية النظام الموزع، ولضمان ذلك غالباً يتم نشر المكونات بشكل يدوي أثناء مرحلة التصميم. لكن في التطبيقات العملية نحتاج غالباً إلى إضافة / حذف/ خدمات / عقد إلى / من/ النظام الموزع، كذلك يجب مراعاة أن الاتصال بين العقد التي يعمل عليها النظام الموزع ليس مستقر دوماً وبالتالي توافرية النظام ستتأثر بذلك. تم عرض مجموعة حلول لمشكلة نشر النظم الموزعة أثناء التصميم (Design time) تتلخص في تجميع المكونات التي تتفاعل مع بعضها كثيراً على نفس الجهاز [12] ولكن تلك الحلول تبقى محدودة لأن معرفة مدى تواصل المكونات مع بعضها يحصل في زمن التنفيذ (Run Time) وليس في زمن التصميم. ومن هنا ظهر مفهوم إعادة نشر مكونات النظام الموزع (Redeployment) لتحسين التوافرية، لذلك من الطبيعي أن تتم عملية إعادة نشر المكونات في زمن التنفيذ وبشكل تلقائي.

أهمية البحث وأهدافه:

تعد مشكلة جودة الخدمة من أعقد وأهم المشاكل التي تواجه النظم البرمجية الموزعة في وقتنا الحالي والتي تتمثل بنمو النظم الموزعة المستمر مما يفرض تغيرات في بارامترات جودة خدمة النظام كالتوافرية والأداء، بالإضافة إلى أن الوسطاء (Parameters) التي تؤثر على جودة بنية انتشار النظام غير معروفة قبل الانتشار ويمكن أن تتغير بأي وقت. لذلك تم في هذا البحث الاستفادة من نظرية الأرتال التي تساعد بالحصول على قيم بارامترات مهمة تساعد بنشر وإعادة نشر مكونات النظام الموزع على العقد بما يضمن تقديم جودة خدمة أعلى من قبل ذلك النظام.

طرائق البحث ومواده:

- دراسة الأبحاث المتعلقة بنشر مكونات النظم الموزعة.
- دراسة الأبحاث المتعلقة بإعادة نشر مكونات النظم الموزعة بما يحسّن في جودة الخدمة.
- تقديم الطريقة المقترحة لإعادة نشر المكونات بما يحقق توافرية وأداء أفضل في النظام الموزع.
- القيام بثلاث تجارب بواسطة البنية البرمجية Java RMI وتقييم النتائج.

نشر وإعادة نشر مكونات النظام الموزع:

برز مفهوم إعادة نشر مكونات النظام الموزع كحل ديناميكي لمشاكل ضعف توافرية تلك النظم، ومنذ ذلك الحين بدأت الأبحاث في إطار ذلك المفهوم واستمرت حتى يومنا هذا. ومن تلك الأبحاث البحث المقدم في [9] حيث قدم الباحثون اثني عشر نموذج تصميمي تساعد في مراقبة النظام برمجياً وعتادياً واتخاذ القرار المناسب تصميمياً. وفي [12] تم إجراء دراسة استقصائية (Survey) عن طرق نقل التطبيقات في الحوسبة الموزعة بين العقد، وقام الباحثون بعرض برمجيات وسيطة (Middle ware) تقوم بمهمة النقل تلك والمقارنة بينها، تميزت تلك البرمجيات بأنها تقوم بنقل حالة التطبيق التنفيذية من عقدة إلى أخرى بحيث يكون مسار النقل محدد سابقاً وبحيث يتم نقل كامل التطبيق. بعض الباحثون اتجهوا نحو نمذجة مكونات التطبيق الموزع كبيان (Graph) وحل مسألة نشر المكونات بحيث تتحقق جميع شروط النظام الموزع مع مراعاة القيود التي تفرضها الشبكة كما في [8] حيث قام الباحثون بصنع النموذج ثم اقتراح خوارزمية ADE (Automatic Distributed Environment) تقوم بمراقبة أداء وموارد النظام الموزع الكلي بدون طلب معلومات عن حالة العقد أو النفاذ إلى تلك العقد التي تعمل عليها مكونات النظام الموزع، وإنما يتم اتخاذ قرار إعادة الانتشار بشكل مركزي، علماً أن البيان الذي تم بناؤه لا يتغير بعد أول مرة وبالتالي تغيرات الشبكة لا تؤخذ بالحسبان. في الأبحاث المعروضة سابقاً تم تقديم حلول لنقل مكونات النظام الموزع من جهاز إلى آخر بما يضمن اكتمال تنفيذ المكونات لمهامها وبحيث يبقى النظام الموزع في حالة توافرية معظم الأوقات، لكن لم يتم اقتراح أية حلول تتعلق بإعادة الانتشار الأمثل. إن إعادة نشر مكونات النظام الموزع في حال أردنا اختبار كافة الاحتمالات من أجل n مكون للنظام و k عقدة نكون أمام مسألة تعقيدها أسي من رتبة k^n. في الدراسات التالية تم تقديم حلول إعادة نشر المكونات مع مراعاة أمثلية الحل المقترح. في [4] طور الباحث نموذج برمجة صحيحة ثنائية (Binary Integer Programming) يأخذ بالحسبان نماذج الاتصال بين التطبيق والشبكة بحيث يكون الهدف هو تصغير عدد الاتصالات الكلية للتطبيق وذلك بفرض أن الشبكة ستبقى في حالة استقرار من حيث عدم إضافة أو حذف عقد، وأن سبب إعادة الانتشار هو فقط نتيجة لضعف الموارد في العقدة التي ينفذ عليها المكون. في [10] قدّم الباحث طبقة تقوم بتوزيع طبقات النظام الموزع بشكل تلقائي بين الجهاز المحمول (Mobile) والتطبيق المستضاف على السحابة (Cloud application) بما يحقق أمثلية من حيث الكلفة والتأخير، وذلك بافتراض أن التطبيق المستضاف على السحابة ذو أداء مستقر وأن المشاكل قد تنشأ فقط بسبب محدودية مصادر الجهاز المحمول أو أثناء الاتصال. استفاد الباحثون في [5] من مفهوم الذكاء الصنعي حيث وضعوا نموذج لوصف النظام الموزع من وجهة نظر الشبكة وخصائص النظام والقيود باسم (Component Placement Problem, CPP)، ثم تشغيل خوارزمية ذكاء صنعي (Sekitei) لحل النموذج تبعاً لطبولوجية الشبكة والقيود الموضوعة. المشكلة التي قد تؤثر على اعتمادية الحل الناتج من الخوارزمية هي أن ذلك الحل مرتبط بصحة النموذج وتكاملية القيود. بعض الباحثين حاولوا إنقاص تعقيد تجريب

كل احتمالات إعادة الانتشار كما في [6] حيث قام الباحثون بتطوير خوارزمية (Aval) التي تقوم بإنقاص تعقيد عملية إعادة الانتشار من k^n إلى k^{n-m} وذلك بتثبيت m مكون وتجريب الاحتمالات الأخرى (n−m)، وهذا تطلب وجود مكون مركزي على معرفة بحالة النظام الكلية وذو إمكانية للنفاذ إلى كل عقدة في النظام الموزع الكلي، مما يجعله غير مناسب في كثير من الأنظمة الموزعة (مثلاً شبكات المحمول عند الطلب، ad hoc mobile network). وفي [7] تم اقتراح خوارزمية (DesAP) التي تقرر إعادة الانتشار بدون وجود مكون مركزي وإنما عن طريق مزاد علني (Auction). حيث تحتوي كل عقدة في الخوارزمية عميل (agent) لتحديد المزاد، ويشترط على كل عقدة تدخل مزاد علني ألا تدخل في مزاد آخر حتى تحسم النتيجة التي تكون بانتقال مكون من عقدة مضيفة إلى أخرى ضمن نفس المزاد. يُلاحظ هنا عدم شمولية الخوارزمية لكل العقد وإنما فقط للعقد التي دخلت في المزاد، كذلك تفرض الخوارزمية أن المضيفين في حالة استقرار ريثما يتم انتهاء المزاد.

الطريقة المقترحة:

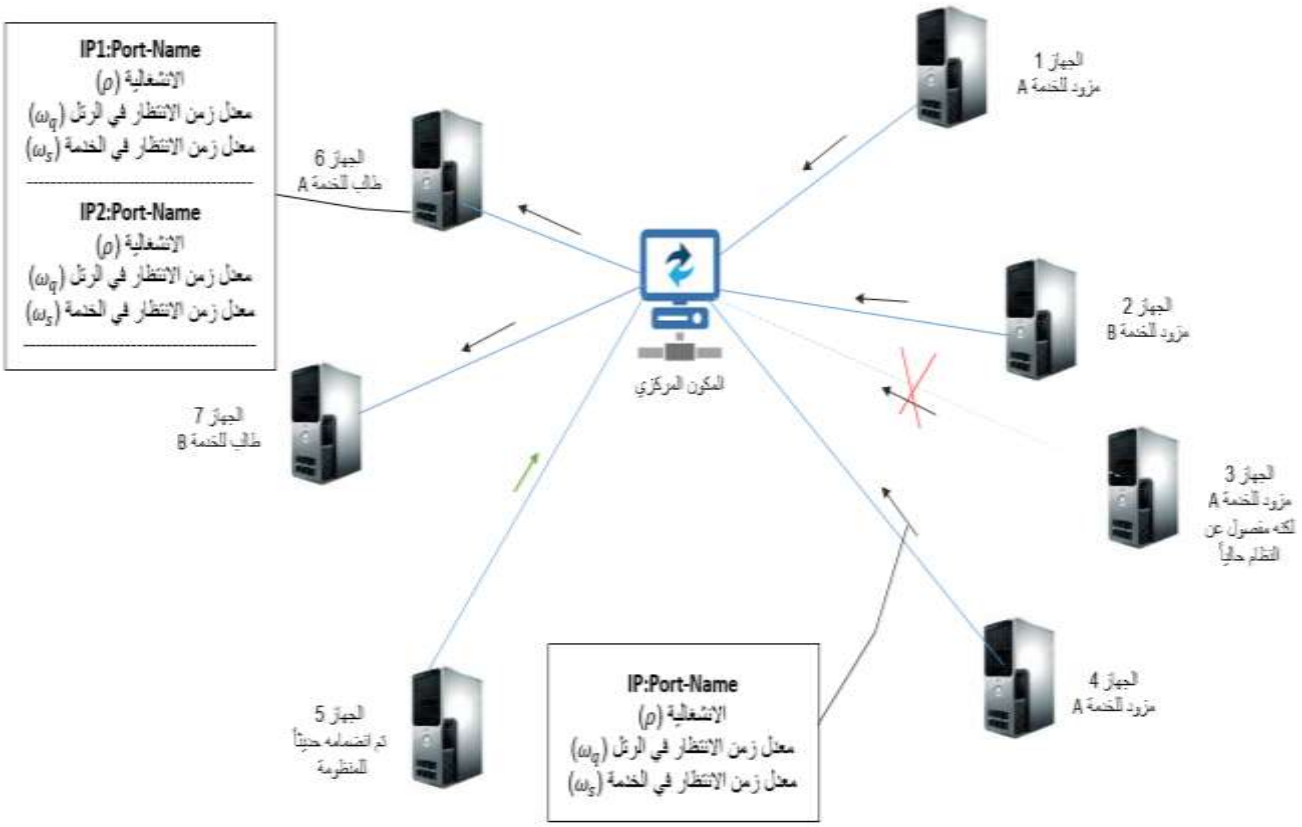
ترتكز الطريقة التي تم اقتراحها من أجل إعادة انتشار النظام الموزع على نقطتين أساسيتين:

1)إضافة مكون مركزي يقوم باستقبال تقارير من مكونات النظام الموزع تبين حالة الاستجابة لتلك المكونات، ويقوم بتوزيع تلك التقارير على باقي مكونات النظام ويقوم بأخذ قرار بتفعيل أحد المكونات على جهاز آخر.

2) إضافة طبقة برمجية إلى كل مكون لمعالجة التقارير الواردة من المكون المركزي وتقرر ممن ستُطلب الخدمة عند الحاجة أو عند وجود أكثر من مكون يقدم نفس الخدمة. يبين الشكل (1) النموذج المقترح.

نناقش فيما يلي ثلاث نقاط هامة بخصوص صحة وتكاملية الحل المقترح:

1) عند وجود مكون مركزي في النظم الموزعة يُخشى دوماً من توقف هذا المكون عن العمل حتى لو تم رفع مستوى توافرية العقدة المستضيفة لهذا المكون عتادياً على حساب التكلفة (تبقى تكلفة رفع توافرية عقدة واحدة أقل بكثير من رفع توافرية كافة عقد النظام الموزع عتادياً). ولذلك رأينا أن يتم تفعيل خدمة المكون المركزي عند العقدة الأقل مشغولية في حالة توقف المكون المركزي عن العمل وبحيث يتم إعادة انتشار المكونات التي تعمل على العقدة والتي تم تفعيل خدمة المكون المركزي عليها إلى عقد أخرى وإكمال خدمة الطلبات الحالية التي قامت تلك العقدة سابقاً باستقبالها، أي بما معناه جعل المكون المركزي مختص باستقبال وتوزيع التقارير بين المكونات فقط. وفي حال دخلت العقدة التي كانت تقوم بمهمة المكون المركزي مرة أخرى إلى النظام ستدخل كعقدة بمشغولية معدومة ونستطيع إعادة نشر المكونات من جديد عند الحاجة.



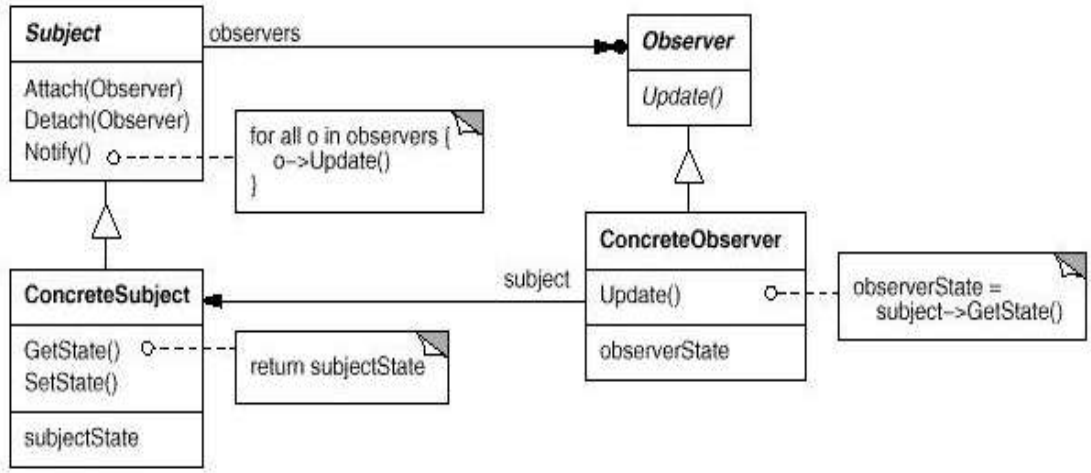
الشكل (1): نموذج الطريقة المقترحة

2) في الحل المقترح يجب توفير آلية قياس لحالة كل عقدة لكي يتم إصدار تقرير ببيانات صحيحة وحقيقية ولذلك رأينا أن يتم تخزين طلبات الخدمة الواردة إلى كل عقدة في رتل (Queue) يُخدم بأسلوب القادم أولاً يُخدم أولاً (FIFO،First In First Out)، ولكي نقوم بالتحليل الرياضي لعمليات ومقاييس الرتل مثل معدل زمن الانتظار في الرتل، معدل زمن الانتظار في الخدمة، ومدى مشغولية العقدة المزودة للخدمة (Utilization)، لجأنا إلى محور من بحوث العمليات (Operation Researches) هو نظرية الأرتال (Queuing Theory) و التي تُساعد على إعطاء بيانات حقيقية من كل مزود خدمة على شكل تقرير يُقدم إلى المكون المركزي. وتبعاً لتدوينKendall لوصف الرتل [1] سنحتاج إلى رتل من النموذج M/M/1. حيث الوسيط الأول M من اليسار يعبر عن التوزيع الاحتمالي لعدد الطلبات الواردة إلى الرتل خلال فترة زمنية، وهو توزيع احتمالي منفصل بمعدل λ يمكن قياسه لأن الطلبات قابلة للعد كل ثانية وبالتالي فهو خاضع لتوزيع بواسون (Poisson Distribution) [3] ذو خاصية فقدان الذاكرة (Memory less) ولذلك وضعت M في التدوين اختصاراً. الوسيط الثاني M يعبر عن التوزيع الاحتمالي لأزمنة الخدمة للطلبات وبالتالي فهو توزيع احتمالي مستمر بمعدل μ يمكن قياسه خلال وحدة الزمن وبالتالي يخضع للتوزيع الأسي (Exponential Distribution) [2] الذي يتمتع بخاصية فقدان الذاكرة أيضاً لذلك نجد M في المركز الثاني في التدوين. أما الوسيط الثالث 1 يدل على عدد العناصر التي ستقدم الخدمة عند الوصول إلى المكون المطلوب من النظام الموزع وهو طريقة (Method) واحدة فقط في ذلك المكون لذلك وُضع 1 في التدوين. مما سبق نجد أنه يمكننا الاستفادة من قوانين الرتل M/M/1 الواردة في الجدول (1) أثناء إصدار التقارير من قبل كل مكون مقدم للخدمة.

الجدول (1): مقاييس الرتل من النمط M/M/1

المقياس	القانون
المشغولية	$\rho = \frac{\lambda}{\mu}$
معدل زمن الانتظار في الخدمة	$\omega_s = \frac{1}{\mu}$
معدل زمن الانتظار في الرتل	$\omega_q = \frac{\rho\omega_s}{1-\rho}$

3) تقنية توزيع التقارير الصادرة من قبل مزودات الخدمة يمكن أن تؤثر على أداء النظام، حيث ستضاف مهمة للمكونات هي تقييم حالتها كل فترة وتبادل رسائل مع المكون المركزي فقط عند حدوث تغيير في الحالة وليس كل فترة زمنية حيث تتضمن الرسالة فقط الوسيط(Parameter) الذي طرأ تعديل عليه مما يُخفف من حجم تبادل المعطيات عبر الشبكة. وبحيث يقوم المكون المركزي بإرسال تلك التعديلات إلى العقد طالبة الخدمة المعروفة من قبله، وأي عقدة طالبة للخدمة جديدة يجب أن تسجل في هذا المكون المركزي لكي تحصل على التقارير والتعديلات على وسطاء تلك التقارير. وبسبب صعوبة ذلك التنسيق برمجياً، كان لابد من اللجوء إلى نماذج التصميم (Design Patterns) المعروضة من قبل GOF(Gang Of Four) في [13] وتحديداً نموذج المراقب (Observer) الذي يقوم بتعريف اعتمادية من النمط واحد لكثير (one–to–many)، وعندما يقوم المكون "واحد" بتغيير حالته يتم إعلام المكونات الأخرى وتحديثها أوتوماتيكياً، وهذا مناسب لحالة المكون المركزي لدينا حيث أن حالته ستتغير باستمرار مع كل تعديل من العقد المزودة للخدمة ويجب نقل هذا التغير في حالة المكون المركزي لتتحدث وسطاء الملفات الموجودة عند طالب الخدمة. يبين الشكل (2) مخطط الصفوف (Class Diagram) للنموذج التصميمي "مراقب". حيث يمثل المكون المركزي دور الـSubject، أما المكونات طالبة الخدمة تلعب دور المراقبين الـObservers اللذين يراقبون تغير حالة المكون المركزي.



الشكل (2): مخطط الصفوف للنموذج التصميمي "مراقب"

في هذا النموذج يعرّف المكون المركزي طريقة Attach تمكن كل طالب خدمة من أن يسجل نفسه لدى المكون المركزي كمهتم بالتعديلات المقدمة من طالبي الخدمة. كذلك يزود المكون المركزي بطريقة Notify تقوم بإعلام جميع المكونات طالبة الخدمة بالتعديل عند حدوثه عن طريق استدعاء الطريقة Update الخاصة بكل مكون طالب خدمة.

النتائج والمناقشة:

تم بتطوير نظام موزع A يعمل بتقنية (Java RMI, Remote Methods Invocation). ثم تم إضافة المكون المركزي مع مراعاة النموذج التصميمي "مراقب" ، ونظام الرتل M/M/1 إلى النظام الموزع A و بالنتيجة تم الحصول على النظام الموزع B الذي يمثل الطريقة المقترحة في هذا البحث. كلا النظامين A,B سيقوم بحساب cosx باستخدام سلسلة تايلور (Taylor Series) المعطاة بالقانون (1):

$$\cos x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} \qquad (1)$$

حيث نحتاج هنا إلى مكون α لحساب a!، ونحتاج إلى مكون β لحساب a^b. وبالتالي لدينا طلب خدمة إلى هذين المكونين الذين سيكونان متاحين على عقد مختلفة عن العقدة التي يتم التنفيذ عليها وإدخال قيم x منها إلى البرنامج. يستمر التنفيذ من أجل n=0 وحتى الحصول على حدين متتاليين في السلسلة يكون الفرق بينهما أصغر هو من رتبة $\varepsilon = 0.00001$. تم في البحث إجراء ثلاث تجارب على النظامين A,B كالتالي:

1) في التجربة الأولى: تم ادخال قيمتين لـ x باستخدام العلاقة (1) المبينة أعلاه وتركنا كلا النظامين ينفذان بشكل طبيعي دون تغيير في العقد التي تنفذ عليها المكونات فحصلنا على الجدول (2) الذي يبين أن النتائج متطابقة بالنسبة للنظامين رغم أن زمن التنفيذ في النظام A أقل بقليل من زمن تنفيذ النظام B بسبب الزمن اللازم لإصدار تقارير من قبل المكونين β,α، وزمن الاتصال.

الجدول (2): نتائج التجربة الأولى

قيمة x بالراديان	نتيجة النظام A	نتيجة النظام B
$\frac{\pi}{4}$	0.707212	0.707212
$\frac{3\pi}{2}$	0	0

2) في التجربة الثانية: تم تكرار التجربة السابقة مع استخدام العلاقة (1) ووضعنا المكون α على العقدة N_1 والمكون β على العقدة N_2، ثم وفور إدخال قيمة x إلى طالب الخدمة تم إطفاء العقدة N_1 مما أدى إلى توقف النظام A عن الاستجابة، أما مع تكرار الخطوات نفسها مع النظام B استمر النظام بالعمل وأعطى النتائج بقيمها كما في التجربة السابقة/1/. ويبين الجدول (3) نتائج التجربة الثانية.

الجدول (3): نتائج التجربة الثانية

قيمة x بالراديان	نتيجة النظام A	نتيجة النظام B
$\frac{\pi}{4}$	--------	0.707212

3) في التجربة الثالثة : تم تكرار التجربة الثانية لكن وضعنا البرنامج p الموضح في الشكل (3) على العقدة N_1 ، وبعد ذلك تم تنفيذ البرنامج p على العقدة N_1 مما أدى إلى توقف العقدة N_1 عن الاستجابة وفقدان التواصل مع المكون α في النظام A. حيث ان البرنامج p يقوم بتكرار طلب إدخال قيمة لـ x وعند إدخال القيمة 3 يدخل في حلقة لا نهائية مما يعني انشغال موارد النظام على العقدة N_1 المضيفة للمكون α وبالتالي سيتوقف النظام A عن الاستجابة أما النظام B يستمر بالتنفيذ.

```
#include<iostream>
 using namespace std;
void main() {
                    int x = 0;
while (true)
 if (x != 3)
cin >> x;
 else
 cout<<"Infinite Loop!!\n";
 }
```

الشكل (3): البرنامج p على العقدة N_1

يبين الجدول (4): نتائج التجربة الثالثة

قيمة x بالراديان	نتيجة النظام A	نتيجة النظام B
$\frac{\pi}{4}$	0.707212	0.707212
$\frac{3\pi}{2}$	------	0

الاستنتاجات والتوصيات:

تم في هذا البحث عرض طرق إعادة انتشار مكونات النظام الموزع باختصار، وتم دراسة طرق إعادة الانتشار الأمثل، تم بيان آلية الطريقة المقترحة التي ترتكز على مكون مركزي يتم تقديم تقارير له عن حالة المكونات المزودة للخدمة عند تغير حالتها الداخلية المُقاسة بواسطة مقاييس رتل من النمط M/M/1.

تبين من النتائج أن الطريقة المقترحة تأخذ زمناً أطول بقليل في حالة الظروف المثالية للنظام الموزع مقارنة بالطريقة التقليدية لتطوير النظم الموزعة،

تضمن الطريقة المقترحة توافرية أعلى بكثير في حالة انقطاع الاتصال بالعقدة المستضيفة للمكون مزود الخدمة، أو في حالة انشغال تلك العقدة بتنفيذ برامج أخرى.

تجدر الاشارة الى أنه يمكن تطوير النظام المقترح بحيث يصبح إطار عمل (Framework) يزود بواجهة برمجية لبناء التطبيقات (Application, API Programming Interface) الموزعة، بما يساعد المطورين، ويمكنهم من تصميم نظم موزعة ذات درجات توافرية وأداء عالية لتلك النظم .

المراجع

1- MEYN, SEAN P., RICHARD L. TWEEDIE. *Markov chains and stochastic stability*. *Springer Science & Business Media*, 2012Esary.

2- BICKEL, P. J., E. L. LEHMANN. "*Descriptive statistics for nonparametric models I. Introduction."* Selected Works of EL Lehmann. Springer US, 2012. 465-471

3- SHAKED, MOSHE, GEORGE SHANTHIKUMAR. *Stochastic orders*. Springer Science & Business Media, 2007

4- MEDVIDOVIC, NENAD, AND SAM MALEK. *"Software deployment architecture and quality-of-service in pervasive environments."* International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting. ACM, 2007.

5- MIKIC-RAKIC, MARIJA, SAM MALEK, NENADMEDVIDOVIC. *"Improving availability in large, distributed component-based systems via redeployment."* International Working Conference on Component Deployment. Springer Berlin Heidelberg, 2005.

6- MALEK, SAM, MARIJAMIKIC-RAKIC, NENADMEDVIDOVIC. *"A decentralized redeployment algorithm for improving the availability of distributed systems."* International Working Conference on Component Deployment. Springer Berlin Heidelberg, 2005.

7- DEB, DEBZANI, M. MUZTABAFUAD, MICHAEL J. OUDSHOORN. *"ADE: Utility Driven Self-management in a Networked Environment."* JCP 2.9 (2007): 7-15.

8- RAMIREZ, ANDRES J., AND BETTY HC CHENG. *"Design patterns for developing dynamically adaptive systems."* Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. ACM, 2010.

9- CHUN, BYUNG-GON, et al. *"Clonecloud: elastic execution between mobile device and cloud."* Proceedings of the sixth conference on Computer systems. ACM, 2011.

10- MANGLIK, GAURAV, AND TIANYING FU. "*Apparatus, systems and methods for dynamic adaptive metrics based application deployment on distributed infrastructures.*" U.S. Patent Application No. 13/489,223.

11- YU, PING, et al. *"Application mobility in pervasive computing: A survey*." Pervasive and Mobile Computing 9.1 (2013): 2-17.

12- HUNT, JOHN. "*Gang of four design patterns*." Scala Design Patterns. Springer International Publishing, 2013. 135-136.