مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (38) العدد (5) 2016
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (38) No. (5) 2016

دمـج قاعدة بيـانات مـع Jgroup بالاعتماد على أدوات مقابلة الغرض العلائقية (Hibernate, EclipseLink)

د. رضـوان دنـده*
د. قاسـم قبـلان**
علـي اسـماعيل***

(تاريخ الإيداع 2 / 6 / 2016. قُبِل للنشر في 20/ 9 / 2016)

□ ملخّص □

تقدّم Jgroup نظام اتصالات مجموعة يدمج نموذج الغرض الموزّع Java RMI مع تقنية مجموعة الغرض (Object Group)، وتتميّز بتزويدها مزايا هامّة تجعلها مناسبة لتطوير التطبيقات الشبكية الحديثة.

يضيف هذا البحث ميزة جديدة إلى Jgroup؛ وهي التعامل مع قواعد البيانات بالاعتماد على أدوات مقابلة الغرض العلائقية، حيث تتطلّب العديد من تطبيقات الانترنيت الحالية تخزين البيانات ضمن قواعد البيانات بالإضافة إلى إمكانية استرجاعها في وقت لاحق من خلال عمليات الاستعلام.

تعتبر Hibernate و EclipseLink من أدوات مقابلة الغرض الشائعة والمفتوحة المصدر، فيقدّم هذا البحث طريقة دمج قاعدة بيانات مع Jgroup بالاعتماد على هاتين الأداتين، كما يقارن بين أداء Jgroup المدمجة مع Hibernate وأداء Jgroup المدمجة مع EclipseLink من أجل أنماط استعلام متعددة. تظهر النتائج تفوق أداء Jgroup/EclipseLink على أداء Jgroup/Hibernate، حيث يمكن أن ينخفض زمن التأخير اللازم لتنفيذ الاستعلام مع EclipseLink إلى النصف تقريباً مقارنة مع Hibernate.

تقترح هذه المقالة تصميماً جديداً لإضافة خدمة دوام البيانات (Persistence) إلى Jgroup؛ وذلك من خلال إدراج هذه الخدمة كطبقة ضمن طبقات مدير المجموعة المرتبط مع كلّ مخدّم عضو ضمن مجموعة غرض مخدم Jgroup.

الكلمات المفتاحية: نموذج الغرض الموزع؛ Jgroup؛ دوام البيانات؛ أدوات مقابلة الغرض العلائقية؛ Hibernate ؛ EclipseLink.

* أستاذ– قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية – جامعة تشرين– اللاذقية – سورية.
** مدرّس– قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية – جامعة تشرين– اللاذقية – سورية.
*** طالب دراسات عليا (دكتوراه) – قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية– جامعة تشرين– اللاذقية– سورية.

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (38) العدد (5) 2016
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (38) No. (5) 2016

Database Integration with Jgroup based on Object Relational Mapping tools (Hibernate, EclipseLink)

Dr. Radwan Dandah*
Dr. Kasem Kabalan**
Ali Esmaeel***

(Received 2 / 6 / 2016. Accepted 20 / 9 / 2016)

□ ABSTRACT □

Jgroup presents a Group Communication System that integrates the Java RMI distributed object model with Object Group, and provides several important features that make it suitable for developing modern networked applications.

This research adds new feature into Jgroup, which is the ability to deal with database based on Object Relational Mapping (ORM) tools, where many current Internet applications require storing data in database and retrieving them later through lookup operations.

Hibernate and EclipseLink are popular and open source ORM tools, so we present a method integrating database with Jgroup based on these tools, as we compare between the performance of Jgroup integrated with Hibernate and the performance of Jgroup integrated with EclipseLink for different types of queries. The results show that Jgroup/EclipseLink outperforms Jgroup/Hibernate, and the delay time required to perform query with EclipseLink may decrease to the half compared with Hibernate.

This paper proposes a new design to add Data Persistence service to Jgroup; by inserting this service as a layer into Group Manager which is connected to each replica in Jgroup object group.

Keywords: Object Group Model; Jgroup ;Data Persistence; Object Relational Mapping tools; Hibernate; EclipseLink.

* **Professor, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**
** **Assistant Professor, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**
*** **Postgraduate Student, Department of Systems and Computer Networks, Faculty of Information Engineering, Tishreen University, Lattakia, Syria.**

مقدمـة:

تزايد الاعتماد على الأنظمة الشبكية في النشاطات اليومية؛ وعلى اعتبار أن الخدمة (Service) المزوّدة هي الأساس في هذه الأنظمة، توجّب أن تكون هذه الخدمة متوفّرة وموثوقة. يتم تحقيق توافرية الخدمة من خلال زيادة عدد المخدّمات باستخدام تقنية النسخ (Replication)، في حين تتحقّق الموثوقية من خلال آليات التنسيق بين هذه المخدمات لتقدّم الاستجابة الصحيحة إلى زبائن النظام.

تتطلّب عملية النسخ دعماً من النظام لبدئيات (primitives) البثّ المتعدّد (multicast) التي تسمح باستدعاء الطريقة نفسها على عدّة نسخ خدمة؛ الأمر الذي تفتقر إليه إطارات عمل (Frameworks) الوسيط البرمجي (Middleware) مثل RMI(Remote Method Invocation) وCORBA [15].

تساهم أنظمة اتصالات المجموعة الموجّهة بالمنظار (View-Oriented Group Communication System)[16] في تحقيق شفافية التضاعف، حيث تزوّد هذه الأنظمة خدمتين أساسيتين؛ هما خدمة اتصالات موثوقة وخدمة عضوية المجموعة. من الأمثلة على هذه الأنظمة JavaGroups [17]، Jgroup [1].

تعتبر Jgroup نظام اتصالات مجموعة للبيئات القابلة للتجزئة، تدمج أسلوب مجموعة الغرض (Object Group)[18] مع مفهوم الأغراض الموزّعة المعتمدة على Java RMI؛ سامحةً بذلك للزبائن بالتفاعل مع مجموعة الغرض من خلال استدعاء الطرائق عليها. قدّمت Jgroup خدمة عضوية المجموعة القابلة للتجزئة التي تتبّع مسار التغيّرات الإرادية في عضوية مجموعة المخدم (مغادرة أحد الأعضاء أو انضمام أعضاء جدد) وغير الإرادية الناتجة عن حالات التعطّل لتزوّد كلّ مخدّم بمنظار(view) يحوي قائمة بالأعضاء الحاليين في المجموعة. تسمح هذه الخدمة باستمرارية توفّر الخدمة في كل جزء من أجزاء الشبكة؛ وذلك بعد حدوث تجزئة في شبكة الاتصال تؤدي إلى انفصال الأجهزة المستضيفة عن بعضها. في حين تدعم أنظمة اتصالات المجموعة التي تعتمد أسلوب التجزئة الأساسي (primary partition) مثل ISIS استمرارية الخدمة في جزء واحد فقط يسمى الجزء الأساسي، وتتوقّف النسخ المنفصلة عن هذا الجزء عن تزويد الخدمة. تعالج Jgroup أيضاً عملية العودة من التجزئة من خلال خدمة دمج الحالة. تعيد هذه الخدمة جميع نسخ الخدمة إلى حالة عامة متناسقة لتحقيق التوافرية؛ معالجةً بذلك التحديثات المتناقضة الممكن حدوثها على الحالة المشتركة لمجموعة المخدم خلال انفصال أعضائها في أجزاء متعددة. استبدلت Jgroup تقنية RMI بتقنية تمرير الرسائل المتبعة في أنظمة اتصالات المجموعة الأخرى مثل JavaGroups وSpread، فاعتمدت بذلك تقنية واحدة في جميع تفاعلاتها؛ سواء الداخلية لتحقيق التنسيق بين نسخ الخدمة؛ أو الخارجية اللازمة لاتصال الزبون مع مجموعة المخدم، مما يسهّل من عملية تطويرها. تعتمد Jgroup نموذج المجموعة المفتوح (Open Group Communication System) الذي لا يتوجّب فيه على الزبون الانضمام إلى مجموعة المخدّم حتى يتمكّن من الحصول على الخدمة.

تمّ تطوير Jgroup إلى منصّة العمل Jgroup/ARM (Autonomous Replication Management). تسمح ARM [11] بتوزيع نسخ المخدّم على الأجهزة ذاتياً دون تدخل بشري، وتحافظ على عدد محدد وثابت من النسخ في بيئة الهدف من خلال مراقبتها عبر مكوّن مدير النسخ (Replication Manager). عند تعطّل إحدى الأجهزة؛ يكتشف مدير النسخ ذلك ويقوم بإنشاء نسخة بديلة على جهاز آخر. طوّر ميلينغ ARM إلى منصة جديدة Jgroup/DARM (Distributed Autonomous Replication Management) [12]، يتم فيها توزيع النسخ بطريقة موزّعة دون الحاجة إلى المكوّن المركزي (مدير النسخ RM) المستخدم في ARM. تعالج كلّ مجموعة غرض

مخدم في DARM حالات التعطّل بشكل ذاتي (self-healing) من خلال إحدى نسخ المخدم وهي النسخة القائدة (Leader).

تم طرح مجموعة من الأعمال المستقبلية عبر موقع مشروع Jgroup [13] بهدف إضافة وتحقيق مزايا جديدة. تحقّق هذه المقالة الميزة الأولى المطروحة؛ وهي دمج قاعدة بيانات مع Jgroup بالاعتماد على أدوات مقابلة الغرض العلائقية. تتطلّب العديد من تطبيقات الانترنيت الحالية تخزين بيانات ضمن قاعدة البيانات بالإضافة إلى إمكانية تفاعل الزبائن مع قواعد البيانات هذه عبر إجراء عمليات استعلام للحصول على البيانات المطلوبة.

تستخدم أدوات مقابلة الغرض العلائقية ORM (Object Relational Mapping) لتحقيق مفهوم دوام البيانات (Persistence)، حيث تقوم بتحويل البيانات بين تمثيلها ضمن قواعد البيانات العلائقية وتمثيلها في لغات البرمجة غرضية التوجّه مثل Java. من أدوات ORM الشائعة والمفتوحة المصدر أداة Hibernate [2] والأداة EclipseLink [3]. تمثّل هذه الأدوات تحقيقاً لـ (JPA Java Persistence API) التي تشكّل جسراً بين نموذج الغرض (تمثيل الكيان ضمن برنامج جافا) والنموذج الممثّل للكيان ضمن قاعدة البيانات، وتحتوي مجموعة من الطرائق والصفوف لتخزين كميات كبيرة من البيانات ضمن قاعدة البيانات وبشكل دائم.

تنظيم المقالة: بعد الحديث عن أهمية البحث وأهدافه في الفقرة2، توضّح الفقرة3 من هذه المقالة منهجية البحث المتبعة، بينما تتحدث الفقرة 4 عن Jgroup وآلية عملها. يتم الحديث عن أدوات مقابلة الغرض (Hibernate وEclipseLink) في الفقرة5. تنتقل الفقرة 6 لتوضّح خطوات عملية دمج قاعدة بيانات مع Jgroup مع مثال تطبيقي. تقترح الفقرة7 من هذه المقالة إضافة طبقة دوام بيانات إلى Jgroup. تنهي الفقرة8 هذه المقالة من خلال توضيح نتائج المقارنة بين أداء Jgroup المدمجة مع Hibernate مع أداء Jgroup المدمجة مع EclipseLink بالإضافة إلى مناقشة النتائج التي تم التوصل إليها لمعرفة الأداة الأفضل للاستخدام مع Jgroup.

أهمية البحث وأهدافه:

في الوقت الذي نسعى فيه إلى وجود عدد قليل نسبياً من المخدّمات لتحقيق المستويات المرغوبة من التوافرية والموثوقية؛ مع احتمالية وجود عدد كبير من زبائن الخدمة، تدعم Jgroup قابلية التوسّع (Scalability) لعدم حاجة زبائنها إلى الانضمام إلى مجموعة غرض المخدم قبل الحصول على الخدمة المرغوبة.

إن دعم Jgroup مع ميّزة التعامل مع قواعد البيانات بالاعتماد على أدوات مقابلة الغرض العلائقية يسمح بتحقيق قاعدة بيانات مكرّرة، مما يزيد من توافرية الخدمة المزوّدة.

يتميّز تكرار قواعد البيانات على Jgroup بدعم هذه المنصة لقابلية التجزئة. تعالج Jgroup جميع المشاكل الناشئة عن عملية دمج أجزاء الشبكة بعد العودة من التجزئة من خلال خدمة دمج الحالة (state merging service SMS) التي تسهّل عملية عودة النظام إلى حالة عامة متناسقة بين جميع المخدمات (الوصول إلى نسخة قاعدة بيانات موحّدة ومتناسقة).

يهدف البحث إلى تقديم منهجية لدمج قاعدة بيانات مع Jgroup بالاعتماد على أشهر أداتين من أدوات مقابلة الغرض العلائقية وهما Hibernate و EclipseLink، كما يقارن بين أداء Jgroup/Hibernate وأداء Jgroup/EclipseLink وذلك من أجل عدة أنواع من الاستعلامات (إضافة – تحديث- بحث ...) لمعرفة الأداة الأفضل للاستخدام مع Jgroup.

منهجية البحث:

تمّ تحميل الأدوات الموضّحة في الجدول1 واللازمة لتحقيق عملية الدمج وإجراء عملية المقارنة.

جدول 1: أدوات البحث.

Jgroup 3.0.1 distribution [1]
Java J2SE version 1.8.0-20 [10]
Apache Ant version 1.8.4 [8]
Apache Log4j version 1.3-alpha-8 [9]
Hibernate version 4.3.10 Final [7]
HSQLDB version 2.3.3 [6]
EclipseLink2.6 [4]

تم تظليل الأدوات اللازمة لتشغيل تطبيقات Jgroup. طوّرت هذه المنصة باستخدام JAVA J2SE version1.5، ويمكن تحميل هذه النسخة أو أية نسخة أحدث منها (هنا تم اختيار النسخة 1.8). تزوّد الحزمة Apache Ant بنسختها 1.8.4 وسيلة سهلة وبسيطة لترجمة واختبار وتشغيل تطبيقات Jgroup. بينما تستخدم الأداة log4j بهدف تصحيح الأخطاء التي يمكن مواجهتها خلال تشغيل التطبيقات. تم استخدام قاعدة بيانات HSQLDB لإنشاء قاعدة بيانات وإجراء عمليات الاستعلام عليها. بالإضافة إلى الحزم البرمجية الخاصة بأداة Hibernate. بنسختها 4.3.10 وأداة EclipseLink بنسختها 2.6 وهي أحدث النسخ المتوفرة من هذه الأدوات. بعد تحميل هذه الأدوات؛ تم اعتماد الإعدادات الافتراضية الخاصة بها دون إجراء أية تعديل عليها.

بعد توضيح كيفية استخدام Hibernate و EclipseLink لدعم Jgroup في تعاملها مع قواعد البيانات، تتم مقارنة أداء Jgroup المدمجة مع Hibernate مع أداء Jgroup المدمجة مع EclipseLink عن طريق قياس زمن التأخير اللازم لإجراء أنواع مختلفة من الاستعلامات وذلك مع تزايد عدد النسخ المستخدمة ضمن مجموعة غرض المخدم. يمثّل زمن التأخير الفاصل الزمني ما بين لحظة طلب تنفيذ الاستعلام وحتى لحظة إتمام تنفيذه (الحصول على النتائج من طرف الزبون). يتم تنفيذ كل استعلام عشر مرات، من ثم يتم أخذ المتوسط الحسابي لأزمنة التأخير مقدرة بالميلي ثانية. تم تنفيذ التجارب على جهاز محمّل عليه نظام التشغيل windows7، ويمتلك ذاكرة RAM بحجم 4GB، ومعالج Intel(R) Core™ 2Duo CPU T5750 @2.00GHz 2.00GHz.

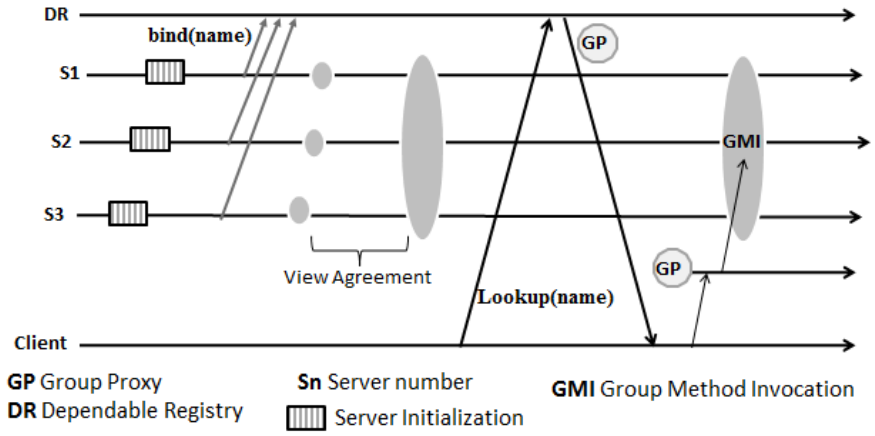
1 منصّة عمل مجموعة الغرض الموزّع Jgroup:

استبدلت Jgroup تقنية RMI بتقنية تمرير الرسائل المتبعة في التفاعلات ضمن Object Group، وأصبح النظام يستخدم تقنية واحدة (RMI) في جميع تفاعلاته، سواء الداخلية بين المخدمات الأعضاء، أو الخارجية مع زبائنه. توضّح الفقرات التالية آلية العمل في Jgroup وخدمة استدعاء طريقة المجموعة من النمط الخارجي لإظهار كيفية تحقيقها لتكامل البيانات.

4 1 - آلية العمل في Jgroup:

يوضح الشكل(1) مراحل العمل في Jgroup. تقوم نسخ المخدمات (s1,s2,s3) بتسجيل نفسها تحت اسم المجموعة ضمن المسجل DR (Dependable Registry) في Jgroup من خلال الطريقة ()bind. تكتشف خدمة عضوية المجموعة على كلّ نسخة مخدم قدرة جميع النسخ الثلاث على الاتصال فيما بينها، فتقوم بتحميل منظار واحد مؤلف من النسخ الثلاث، وذلك على كلّ نسخة مخدم منها. حتى يتمكّن الزبون من الاتصال مع مجموعة المخدم؛

ينبغي عليه الاستعلام (Lookup) من DR عن اسم المجموعة المطلوبة، ليحصل على وكيل المجموعة GP (Group Proxy). يحوي وكيل المجموعة معلومات عن النسخ المكوّنة للمجموعة، ممّا يمكن الزبائن من الاتصال مع المجموعة ككيان مفرد بإنجاز الاستدعاء عبر هذا الوكيل. تنسّق المخدمات أحداثها ضمن المنظار نفسه للحفاظ على حالتها المشتركة وتزويد خدمة موثوقة من خلال خدمة استدعاء الطريقة الداخلي (Group Method Invocation GMI).



الشكل 1: مراحل العمل في Jgroup.

4 2 – خدمة استدعاء طريقة المجموعة من النمط الخارجي في Jgroup:

يعيد النمط الخارجي من خدمة استدعاء طريقة المجموعة إجابة واحدة فقط بدلاً من مصفوفة نتائج، فيحقّق بذلك الشفافية في عملية الاستدعاء (كما لو أنه استدعاء طريقة على مخدم مفرد). يمكن التمييز بين نوعين في هذا النمط من الاستدعاء:

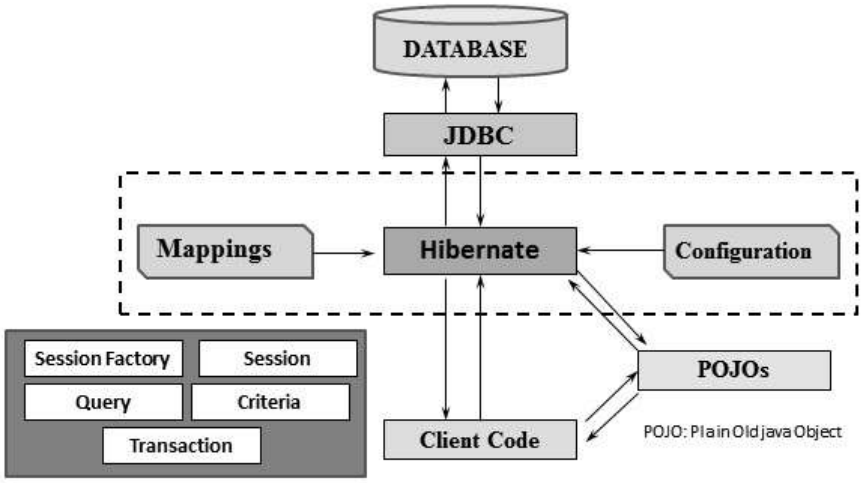
- Anycast EGMI: يتم إنجاز الاستدعاء من خلال تنفيذ الطريقة على مخدم واحد فقط على الأقلّ. ويعتبر بذلك ملائماً للطرائق التي لا يشمل تنفيذها كامل أعضاء المجموعة، كطرائق القراءة من قواعد البيانات المكرّرة.
- Multicast EGMI: يتم إنجاز الاستدعاء من خلال تنفيذ الطريقة نفسها على كل مخدم موجود ضمن جزء الشبكة الذي ينتمي إليه الزبون المستدعي. وهي ملائمة للطرائق التي يؤثر تنفيذها على كلّ مخدم في المجموعة، كطرائق تعديل الحالة في قواعد البيانات المكرّرة.

يضمن تحقيق EGMI في Jgroup خاصية تزامن المنظار (view synchrony)، فعندما يقوم مخدمان (s1, s2) بتحميل زوج من المناظير المتتالية (v1,v2) من خلال خدمة عضوية المجموعة، فإن كل من المخدمين يكون قد نفّذ المجموعة نفسها من استدعاءات EGMI وذلك ضمن المنظار الأول v1. بمعنى آخر، قبل أن يتم تحميل المنظار الجديد v2، يتوجّب على جميع المخدمات المنتمية إلى v1 أن تتوافق على جميع استدعاءات EGMI المنجزة خلال تواجدها معاً في المنظار v1. تضمن هذه الخاصية تكامل البيانات، ففي حالة قواعد البيانات المكررة نضمن أن التعديل المنجز على أحد النسخ من خلال استدعاء طريقة مجموعة خارجي قد أنجز على جميع المخدمات وبالتالي انتقل التعديل إلى جميع قواعد البيانات المكررة فتحافظ بذلك على حالة مشتركة فيما بينها.

2 - أدوات مقابلة الغرض العلائقية (Object Relational Mapping):

5-1- Hibernate:

تتوضع Hibernate بين كود الزبون (Client Code) وقاعدة البيانات (DATABASE) كما هو موضّح في الشكل(2). حتى تتمكّن Hibernate من تحقيق الربط بين أغراض جافا والكيانات المقابلة لها في قاعدة البيانات؛ فإنها تتطلّب ملفات إعدادت لتحقيق أداء دقيق وفعّال للنظام. تشمل هذه الإعدادات ملفي الإعدادات (Configuration) والمطابقة (Mappings). يوضّح الشكل(2) المكونات الأساسية في Hibernate [14]، حيث يعتبر غرض الإعداد (Configuration Object) الغرض الأول الذي يتم إنشاؤه في أي تطبيق Hibernate وغالباً يتم إنشاؤه لمرة واحدة فقط أثناء تهيئة التطبيق.



الشكل 2: هيكلية Hibernate وتوضّعها بين كود الزبون وقاعدة البيانات.

يحدّد هذا الغرض خصائص الملفات المطلوبة كمسار قاعدة البيانات المراد الاتصال معها، وضبط المقابلات (Mappings) بين خصائص صف جافا وأعمدة جدول قاعدة البيانات. تحوي Hibernate مجموعة من الأغراض التي يتم استخدامها في التفاعل مع قاعدة البيانات نذكر منها:

a) غرض مصنع جلسة (Session Factory Object): يتم الحصول عليه من غرض الإعداد ويسمح بتهيئة غرض الجلسة (Session Object) المستخدم لتحقيق الاتصال مع قاعدة البيانات. ينشأ هذا الغرض عند إقلاع التطبيق ويتم الاحتفاظ به خلال تشغيل التطبيق لاستخدامات لاحقة. يحتاج التطبيق إلى غرض Session Factory واحد لكل قاعدة بيانات باستخدام ملف إعدادات منفصل. وعند التفاعل مع عدة قواعد بيانات، يجب إنشاء أغراض Session Factory متعدّدة.

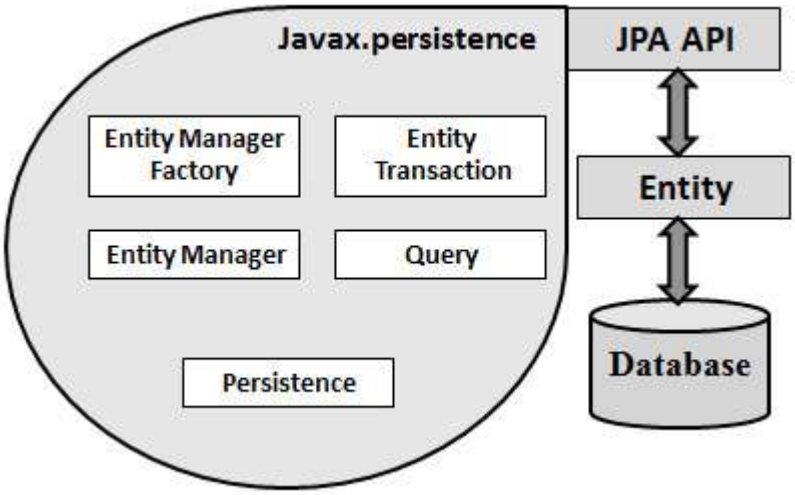
b) غرض جلسة (Session Object): تستخدم الجلسة للحصول على اتصال فيزيائي مع قاعدة البيانات. تتم تهيئة هذا الغرض في كل وقت يتطلّب تفاعلاً مع قاعدة البيانات. لا ينبغي أن تبقى أغراض الجلسة مفتوحة لوقت طويل لأنها غير آمنة من التهديد، وبالتالي ينبغي إنشاؤها وحذفها عند الحاجة بينما يبقى مصنع الجلسة مهيأً طيلة حياة التطبيق.

c) غرض الاستعلام (Query Object): تستخدم أغراض الاستعلام لغة SQL أو لغة استعلام Hibernate (HQL) لاستعادة البيانات من قاعدة البيانات وإنشاء الأغراض. يستخدم غرض الاستعلام لربط بارامترات

الاستعلام المطلوب، وتنفيذ الاستعلام في النهاية. توجد أيضاً أغراض اختيارية أخرى كغرض المناقلة (Transaction Object)، وغرض مقايسة (Criteria).

5-2- EclipseLink:

يوضّح الشكل(3) الأغراض الأساسية المكونة للأداة EclipseLink (Entity Manager Factory، Entity Manager، Query، Entity Transaction). تستخدم EclipseLink ملف xml يملك الاسم (Persistence.xml) لتحديد مسار قاعدة البيانات المراد الاتصال معها ومسار الصف المراد تخزينه، بالإضافة إلى إعدادات أخرى.



الشكل 3: هيكلية EclipseLink.

تمتلك EclipseLink هيكلية مشابهة لـ Hibernate (الموضّحة في الشكل2)، حيث يمكن توضيح التقابل بين أغراض Hibernate وأغراض EclipseLink من خلال الجدول2:

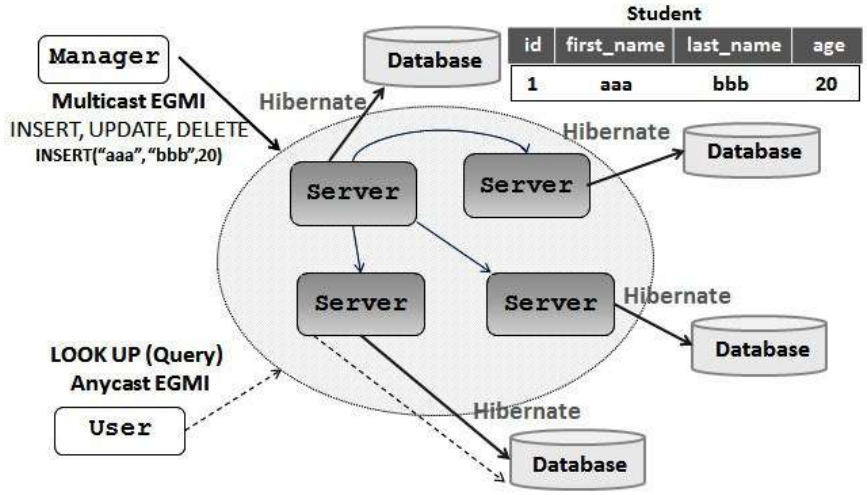
جدول 2: الأغراض المستخدمة في Hibernate وما يقابلها في EclipseLink [5]

javax.persistence	org.hibernate	التوصيف
Persistence	cfg.configuration	يزود صفاً تمهيدياً لإعداد مصنع جلسة(في Hibernate) أو مصنع مدير كيان (في EclipseLink) ويستخدم لإنشاء الجلسة
EntityManagerFactory	SessionFactory	يستخدم لفتح الجلسة أو مدير الكيان لمعالجة طلب المستخدم.
EntityManager	Session	يزوّد صفوفاً وطرائق لتخزين وتحميل كيانات من و إلى قاعدة البيانات. يزوّد أيضاً API للحصول على مناقلة وإنشاء استعلام.
EntityTransaction	Transaction	تزوّد APIs لإدارة المناقلات.
Query	Query	تزوّد APIs لتنفيذ استعلامات

توضّح الفقرة التالية طريقة دمج قاعدة بيانات مع Jgroup بالاعتماد على Hibernate، حيث تتم عملية الدمج مع EclipseLink بطريقة مشابهة.

3 - طريقة دمج قاعدة بيانات مع Jgroup:

يوضّح الشكل(4) طريقة لدمج قاعدة بيانات مع Jgroup، حيث تم اختيار قاعدة بيانات بسيطة مكوّنة من جدول واحد فقط للتبسيط.



الشكل 4: الشكل العام لدمج قاعدة بيانات مع Jgroup بالاعتماد على Hibernate.

يرتبط كلّ مخدّم (Server) مع قاعدة بيانات. تتكوّن قاعدة البيانات الموضّحة في الشكل 4 من الجدول Student، يحتوي هذا الجدول أربعة أعمدة وهي على الترتيب (id, first_name, last_name, age). يمكن لكلّ مخدّم أن يتفاعل مع قاعدة البيانات بالاعتماد على Hibernate أو EclipseLink، حيث يحدّد الطلب الوارد من زبائن النظام وهي إما المدير (Manager) أو المستخدم (User) الإجراء اللازم تنفيذه على قاعدة البيانات (تعديل سجلات، حذف، إضافة، استعلام ....). يتفاعل الزبائن مع مجموعة غرض المخدم في Jgroup من خلال خدمة استدعاء طريقة المجموعة من النمط الخارجي الموضحة في الفقرة 2-4. نميّز هنا بين نوعين من الزبائن:

- زبون الإدارة – Manager– يجري عمليات تعديل على قاعدة البيانات (كتابة)، وبما أن هذا التعديل يجب إجراؤه على جميع المخدمات فإن كل طريقة يقوم المدير باستدعائها عبارة عن طريقة مجموعة من نوع البثّ المتعدّد (multicast EGMI). تشمل عمليات التعديل هنا إنشاء جداول جديدة، حذف جداول من قاعدة البيانات، إضافة سجلّات أو تعديلها، حذف سجلّات. كما يمكنه إجراء عمليات استعلام عن بيانات مخزّنة مسبقاً.
- الزبون العادي –User– يجري عمليات الاستعلام (قراءة)، وبما أن جميع المخدّمات تمتلك الحالة المشتركة نفسها؛ أي تخزّن قاعدة البيانات نفسها، فإنّ أي مخدّم ضمن مجموعة الغرض يمكنه الإجابة على طلب الاستعلام المطلوب، وبالتالي فإنّ كل استعلام من قبل User يمثّل طريقة مجموعة خارجية من نمط البثّ الوحيد (Anycast EGMI).

بعد توضيح الشكل العام لطريقة دمج قاعدة بيانات مع Jgroup، نعرض فيما يلي تطبيقاً يوضّح كيفية الاستفادة من هذه الطريقة.

مثال تطبيقي: تحتاج العديد من التطبيقات الشبكية إلى التعامل مع قواعد البيانات، ففي تطبيق قاعدة بيانات لتسجيل الطلاب (Student registration) مثلاً، يقوم مدراء التطبيق (موظفو شؤون الطلاب) بإنشاء مجموعة من المخدّمات وتخزين نسخة كاملة من قاعدة البيانات على كلّ مخدّم. تحتوي قاعدة البيانات في هذه الحالة معلومات عن جميع الطلاب المستجدين أو الذين تم تسجيلهم سابقاً. يحتاج مدراء التطبيق في مرحلة ما إلى تعديل سجّلات أحد الطلاب المخزّنة (عند انتقاله من سنة إلى أخرى مثلاً) أو إضافة سجّل جديد إلى قاعدة البيانات وهو ما يحدث عند تسجيل طالب للمرّة الأولى، الأمر الذي يتطلّب تفاعلاً مع جميع المخدّمات لنقل التعديل إلى جميع نسخ قواعد البيانات. يقوم الموظفون أيضاً في مرحلة من مراحل هذا التطبيق بالاستعلام من قاعدة البيانات للحصول على معلومات مخزّنة عن أحد الطلاب. يمكن تخديم طلب الاستعلام هذا من قبل مخدّم واحد فقط على اعتبار أن جميع نسخ قواعد البيانات متماثلة على جميع المخدّمات. يمثّل الموظّف هنا زبون الإدارة (Manager) الموضّح في الفقرة السابقة. يمكن لأحد الطلاب أو مستخدم آخر أن يقوم بالاستعلام عن معلومات أحد الطلاب من قاعدة البيانات، حيث يمثّل المستخدم في هذه الحالة الزبون العادي (User) الموضّح في الفقرة السابقة.

يمكن عرض طريقة دمج قاعدة بيانات مع Jgroup بالاعتماد على Hibernate في هذا التطبيق من خلال توضيح الخطوات اللازمة لتحقيق التفاعل مع الجدول Student وهي على النحو التالي:

1 - تعريف صف POJO (Plain Old Java Object) الذي يملك تمثيلاً في قاعدة البيانات. يمكن لـ POJO أن يكون أي غرض جافا لا يرث أي غرض آخر ولا يحقّق واجهة. في هذا التطبيق تم إنشاء الصف Student.java.

2 - إنشاء جدول ضمن قاعدة البيانات باستخدام HyperSQL [6] بالاسم STUDENT من خلال كتابة الأمر التالي:

```
CREATE TABLE STUDENT (
        id INTEGER IDENTITY,
        first_name VARCHAR(25),
        last_name VARCHAR(25),
        Age INTEGER);
```

3 - إنشاء ملف المطابقة والذي يملك الاسم Student.hbm.xml، يتم فيه مقابلة خصائص الصف Student المنشأ في الخطوة1 مع الأعمدة المقابلة لها في الجدول STUDENT المنشأ في الخطوة 2.

4 - إنشاء ملف إعدادات Hibernate والذي يملك الاسم hibernate.cfg.xml. يتم من خلال هذا الملف تحديد مسار قاعدة البيانات المراد الاتصال معها ومسار ملف المطابقة المنشأ في الخطوة 3، ومجموعة من الإعدادات الأخرى.

5 - تتمثّل الخطوة الأولى في إنجاز وتصميم تطبيق Jgroup في إنشاء واجهة خارجية (External Interface) تحوي الطرائق التي يمكن استدعاؤها من قبل الزبائن. تدعى الواجهة الخارجية هنا HExternal. من الطرائق الممكن تعريفها الطريقة ()listStudents والتي تستدعى من قبل User أو Manager لاستعراض كامل محتويات السجلّات ضمن الجدول Student، والطريقة ()addStudent والتي يمكن استدعاؤها من قبل المدير(الموظف) فقط لإضافة سجّل طالب جديد إلى الجدول Student.

يمكن توضيح الواجهة الخارجية مع التصريح عن إحدى الطرائق المعرفة ضمنها من خلال الشكل(5).

```
public interface HExternal extends ExternalGMIListener {
public String listStudents( ) throws RemoteException;
```

الشكل 5: الواجهة الخارجية HExternal والمصرح عن إحدى الطرائق ضمنها.

بعد تعريف الواجهة الخارجية، يتم كتابة كود المخدّم HibernateServer، والذي يتم فيه التركيز على الأجزاء التي يتم فيها استخدام Hibernate في مخدم Jgroup للتفاعل مع قاعدة البيانات.

```
1-  private static SessionFactory factory;
2-  try{
3-  factory = new Configuration().configure().buildSessionFactory();
```

الشكل 6: جزء من كود مخدم Jgroup يبين إنشاؤه لغرض مصنع الجلسة.

يقوم المخدم بعد حصوله على مدير المجموعة وتهيئة الطبقات المطلوبة ببناء غرض SessionFactory من غرض الإعداد Configuration كما هو موضّح في السطر 3 من الشكل(6). عند الحاجة للتفاعل مع قاعدة البيانات وذلك بعد استدعاء إحدى طرائق الواجهة الخارجية من قبل أحد الزبائن؛ يتم فتح جلسة من خلال استدعاء الطريقة ()openSession التابعة لغرض مصنع الجلسة المنشأ مسبقاً، كما في السطر 3 من الشكل (7). يلاحظ بأننا صرحنا عن هذه الطريقة على أنها من النمط Anycast.

```
1-  @Anycast public String listStudents( )
    throws RemoteException {
2-  StringBuilder str = new StringBuilder("Results: ");
3-  Session session = factory.openSession();
```

الشكل7 : كود تحقيق الطريقة ()listStudents ضمن المخدّم، والتي يمكن استدعاءها من قبل Manager وUser.

تتطلّب الطريقة ()addStudent تعديلاً على جميع المخدمات الموجودة ضمن مجموعة غرض المخدم؛ وبالتالي ينبغي أن تكون من نمط البثّ المتعدّد Multicast. يوضّح الشكل(8) جزءاً من كود تحقيق الطريقة ()addStudent والتي يسمح للمدير Manager فقط باستدعائها لإضافة سجلات طلاب جدد إلى قاعدة البيانات. ويتم تحقيق طرائق الحذف والتعديل بنفس الطريقة من حيث التصريح على أنها من النمط Multicast.

```
@Multicast public Integer addStudent(String fname, String lname, int age)
throws RemoteException {
Session session = factory.openSession();
```

الشكل8: جزء من كود الطريقة ()addStudent.

بعد تحقيق كود المخدّم مع جميع الطرائق المصرح عنها ضمن الواجهة الخارجية، يتم الانتقال إلى تحقيق كود المدير (الصف Manager)، حيث يوضّح الشكل9 أجزاء منه تبيّن كيفية استدعائه الطريقة ()addStudent لإضافة سجل طالب إلى جدول قاعدة البيانات كما في السطر 4 من الشكل9.

```
1-    DependableRegistry registry = RegistryFactory.getRegistry();
2-    String[] bindings = registry.list();
.....
3-    HExternal server =(HExternal)registry.lookup("Jgroup/HibernateServer");
......
4-    server.addStudent("Fname","Lname", 19);
```

الشكل 9: أجزاء من كود Manager تبين كيفية استدعائه الطريقة addStudent() لإضافة طالب جديد.

وفي الخطوة الأخيرة يتم كتابة كود المستخدم (الصف User) والذي يكون مشابهاً للصف Manager من حيث اتصاله مع مجموعة غرض المخدم واستدعائه طرائق على الواجهة الخارجية. يوضّح الشكل (10) أجزاء من كود الصف User تبين كيفية استدعائه للطريقة listStudents().

```
1-    DependableRegistry registry = RegistryFactory.getRegistry();
2-    String[] bindings = registry.list();
.....
3-    HExternal server=(HExternal)registry.lookup
          ("Jgroup/HibernateServer");
......
4-    String S = server.listStudents();
```

الشكل 10: أجزاء من كود الصف User توضّح كيفية استدعائه للطريقة listStudents للحصول على جميع سجلّات الطلّاب المخزّنة في جدول قاعدة البيانات.

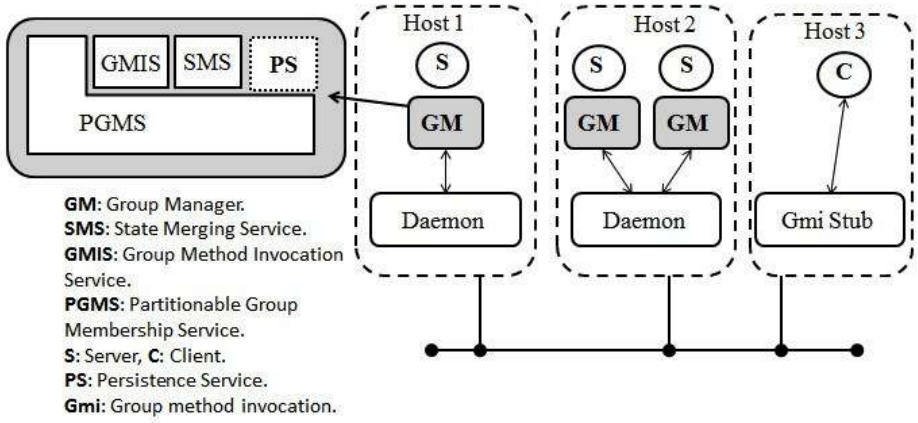
يتم دمج قاعدة بيانات مع Jgroup بالاعتماد على EclipseLink بطريقة مشابهة لـ Hibernate من حيث تحقيق كود المخدم وكود الزبون والتصريح عن الواجهة الخارجية، إلّا أنها تتطلب إنشاء الملف Persistencr.xml بدلاً من الملف hibernate.cfg.xml.

4 - مقترح لإضافة خدمة دوام البيانات إلى Jgroup:

يتكوّن نموذج النظام الأساسي في Jgroup من مجموعة من آلات جافا الافتراضية المتّصلة مع بعضها البعض من خلال شبكة اتصال كما هو موضّح في الشكل 11 . تستضيف كلّ آلة جافا افتراضية مجموعة من الأغراض (نسخ الخدمة S). تنسّق جميع أغراض المخدم أحداثها لتزوّد خدمة معيّنة لأغراض الزبون C. يحصل الزبائن على الخدمات التي تزوّدها مجموعات الغرض من خلال استدعاءات طريقة المجموعة.

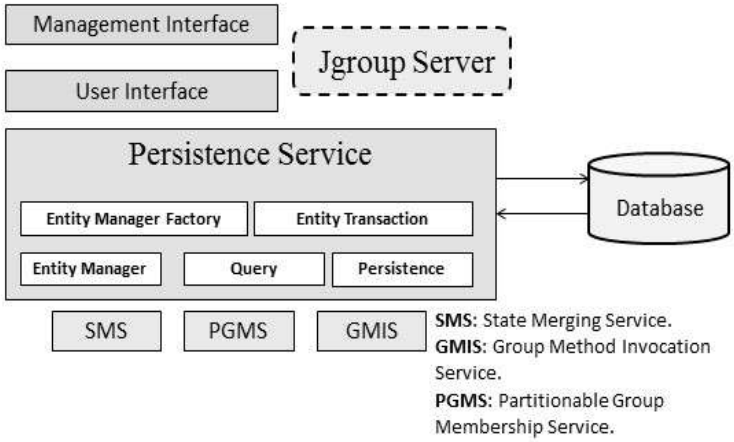
ترتبط كل نسخة مخدّم مع مكوّن مدير المجموعة (GM) المؤلف من عدّة طبقات، تمثّل كلّ طبقة خدمة يتم تزويدها من قبل Jgroup. يهدف التوصيف الموضّح في الشكل12 إلى إضافة خدمة دوام بيانات تعتمد JPA (java persistence API) إلى Jgroup من خلال إدراجها في طبقة جديدة مع طبقات مدير المجموعة. تستخدم هذه الطبقة (دوام البيانات) كلّ من خدمة عضوية المجموعة القابلة للتجزئة لمعرفة جميع المخدّمات القادرة على الاتصال مع بعضها والتي سوف تتشارك العمليات المطلوبة على قاعدة البيانات، وخدمة استدعاء طريقة المجموعة الخارجي

للتفاعل مع زبائن النظام، بالإضافة إلى خدمة دمج الحالة لتسوية الحالة بين نسخ قواعد البيانات بعد حدوث تجزئة في شبكة الاتصال.



الشكل 11: إضافة خدمة دوام البيانات إلى طبقات مدير المجموعة.

يوضّح الشكل12 كيفية توضّع الطبقة الجديدة –طبقة دوام البيانات– فوق طبقات مدير المجموعة. تتم تهيئة طبقة دوام البيانات عند الحاجة للتفاعل مع قاعدة البيانات مع جميع الطبقات التي تستخدمها عند تشغيل المخدم، وذلك بناء على الطلب الوارد من زبائن الخدمة (المدير أو المستخدم).



الشكل 12: بنية مدير المجموعة بعد إضافة طبقة دوام البيانات.

في حال التمكّن من إضافة الطبقة الجديدة؛ تصبح آلية عمل Jgroup على النحو التالي:

-عند تهيئة النسخة (تشغيل المخدّم) يتم تهيئة الطبقات المطلوبة بما فيها طبقة دوام البيانات وصولاً إلى Entity Manager Factory. عند ورود طلب من الزبائن (استدعاء لإحدى الطرائق ضمن الواجهة الخارجية) يتم إنشاء Entity Manager يعتبر مسؤولاً عن تحقيق التفاعل المطلوب مع قاعدة البيانات.

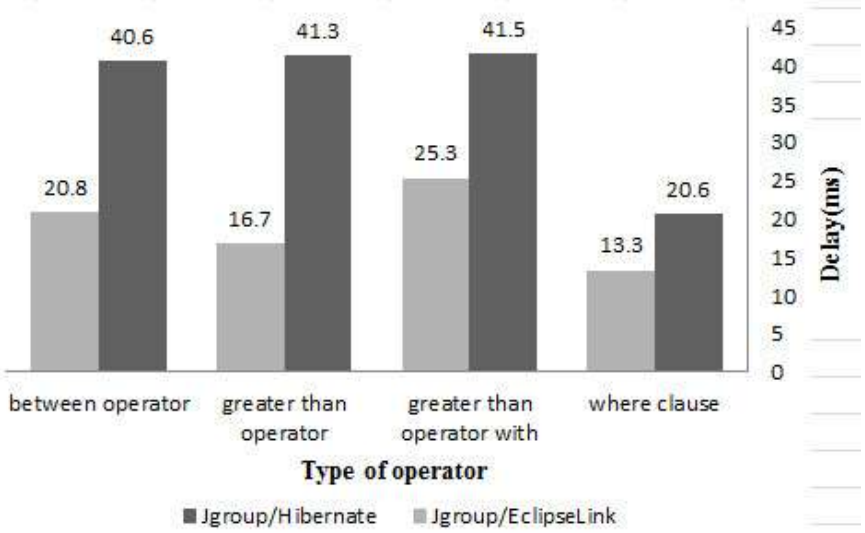
-يتم التفاعل مع النظام وفقاً للتصميم المقترح من خلال واجهتين:

- واجهة الإدارة Management Interface: تمكّن هذه الواجهة مدير النظام من إجراء عمليات التعديل على قاعدة البيانات وذلك من خلال استخدام دلالات البثّ المتعدّد (Multicast) لخدمة استدعاء طريقة المجموعة من النمط الخارجي (EGMI) في Jgroup.

- واجهة المستخدم User Interface: تسمح هذه الواجهة للمستخدمين بالاستعلام عن سجّلات تم تخزينها مسبقاً في قاعدة البيانات.

النتائج والمناقشة:

يمكن توضيح نتائج المقارنة من خلال الشكل 13. حيث يتم توضيح زمن التأخير(بالميلي ثانية) اللازم لإجراء الاستعلام (SELECT) من أجل معاملات مختلفة؛ مع وجود نسخة مخدم Jgroup واحدة.

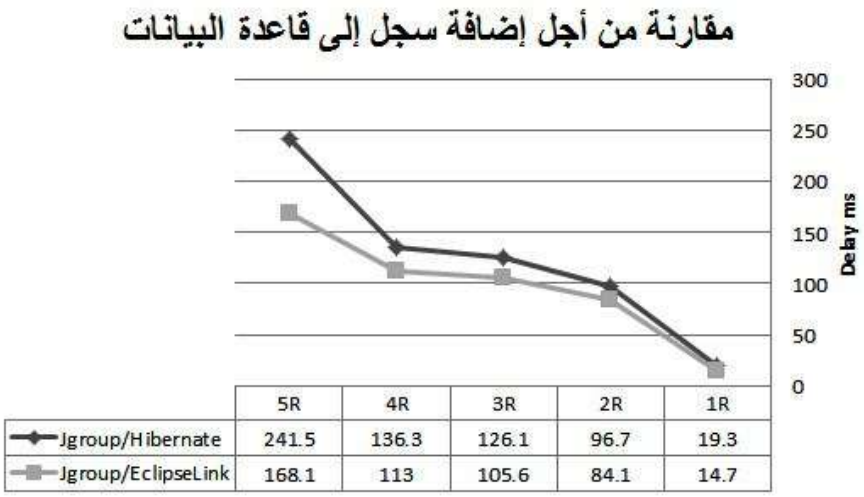


الشكل13: مقارنة أداء Jgroup/EclipseLink مع أداء Jgroup/Hibernate من أجل الاستعلام select مع معاملات مختلفة.

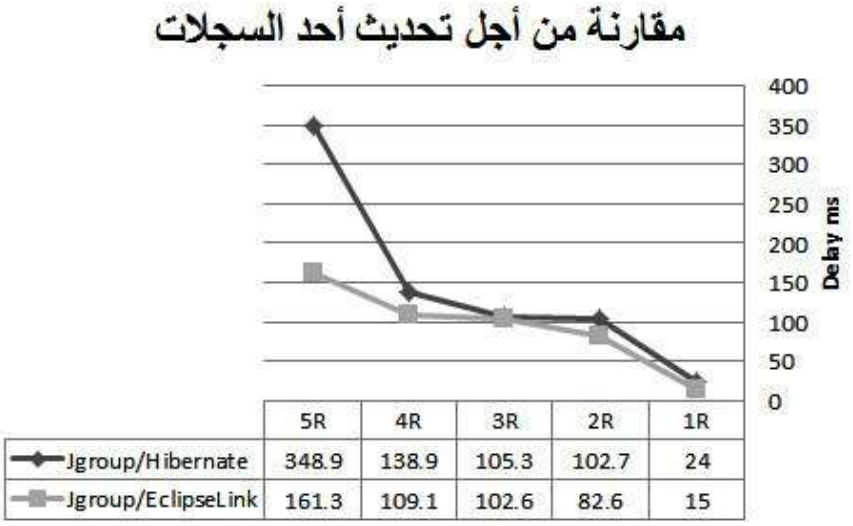
يلاحظ من الشكل(13) تزايد زمن التأخير اللازم لتنفيذ الاستعلام مع وجود معاملات إضافية (أكبر من، المعامل between ...) ويزداد بوضوح عند استخدام Hibernate، حيث ينتقل من 20.6ms إلى 40.6ms مع وجود المعامل between. في حين لم تتأثر Jgroup/EclipseLink بوجود هذه المعاملات (حافظت على قيمة 20ms بشكل تقريبي).

يتفوّق أداء Jgroup المدمجة مع EclipseLink على أداء Jgroup المدمجة مع Hibernate وذلك من أجل مختلف أنواع استعلامات select المنجزة، ويظهر ذلك جلياً مع وجود معاملات في الاستعلام؛ فقد بينت النتائج التي تم التوصل إليها أن زمن التأخير اللازم لإجراء استعلام select مع المعامل greater than عند استخدام EclipseLink مع Jgroup أقل من نصف زمن التأخير اللازم لإجراء الاستعلام نفسه عند استخدام Hibernate مع Jgroup.

يظهر الشكل 14 زمن التأخير (بالميلي ثانية) اللازم لتنفيذ إضافة سجل إلى قاعدة البيانات (اختبار أداء الطريقة addStudent)، في حين يظهر الشكل 15 زمن التأخير اللازم لتعديل سجل؛ وذلك مع تزايد عدد نسخ المخدم، حيث نفّذ كل استعلام عشر مرات وتم أخذ المتوسط مقدراً بالميلي ثانية.



الشكل 14: مقارنة من أجل عملية إضافة سجل إلى جدول قاعدة البيانات.



الشكل 15: مقارنة من أجل عملية تحديث سجل.

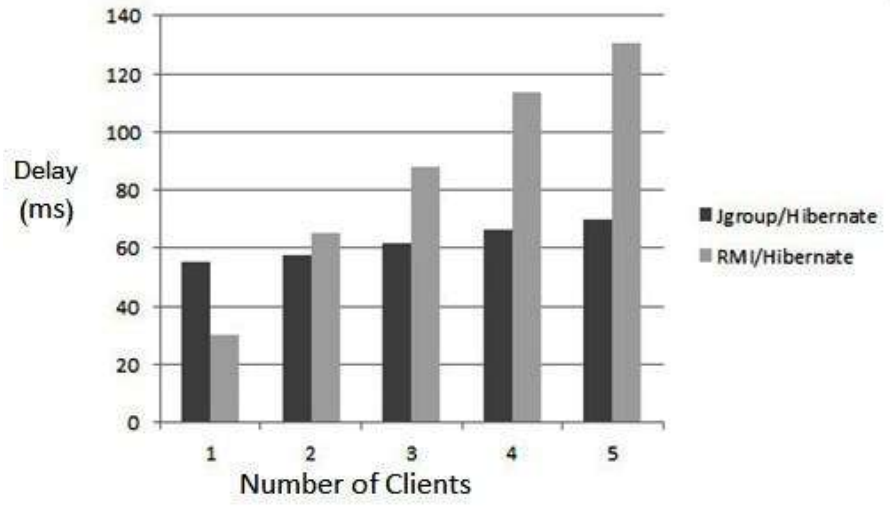
يتبين من خلال الشكلين تزايد زمن التأخير اللازم لإضافة وتعديل سجل مع تزايد عدد النسخ، ويعود ذلك لتنفيذ بروتوكولات الموثوقية في النمط multicast والتي تضمن الحفاظ على حالة قواعد البيانات المشتركة بين جميع نسخ المخدم المتواجدة معاً ضمن المنظار نفسه.

تتطلب عملية إضافة وتحديث سجل باستخدام Jgroup/EclipseLink زمن تأخير أقل مقارنة مع Jgroup/Hibernate،يلاحظ من الشكل 15 انخفاض زمن التأخير اللازم لتحديث سجل من 348.9 ms في Jgroup/Hibernate إلى 161.3 ms في Jgroup/EclipseLink.

من خلال النتائج السابقة يتبين أن أداء Jgroup مع EclipseLink يتفوّق على أداء Jgroup مع Hibernate وذلك في جميع أنواع الاستعلامات المنفذّة، إضافة وتعديل، استعلامات البحث مع معاملات مختلفة (select with operators). لذلك تعتبر EclipseLink هي الأداة المفضّل استخدامها مع Jgroup في تفاعلها مع قواعد البيانات.

تعود فعاليّة EclipseLink لاستخدامها ميّزة الذاكرة الوسيطة (Caching). عندما تتم قراءة غرض من قاعدة البيانات للمرة الأولى؛ فإن قراءته مرة ثانية (من خلال العملية find) لا تحتاج نفاذاً إلى قاعدة البيانات، كما أن عملية الاستعلام Query لا تتطلب إعادة جلب (re-fetched) مما يحسّن من الأداء عبر تقليل عدد مرات الولوج إلى قاعدة البيانات والحصول على البيانات المطلوبة من Cache. في الوقت الذي تدعم فيه EclipseLink هذه الميزة بشكل ضمني، يمكن لـ Hibernate أن تفوّض هذه العملية إلى أداة مخصصة مثل EhCashe of Hazelcast .

يظهر البحث ***فائدة تكرار قواعد البيانات على Jgroup***، من خلال المقارنة بين أداء RMI (Remote Method Invocation) عند استخدامها لمخدم واحد فقط مرتبط مع قاعدة بيانات بالاعتماد على Hibernate، مع أداء Jgroup بوجود 5 مخدّمات؛ كلّ منها مرتبط مع نسخة قاعدة بيانات بالاعتماد على Hibernate أيضاً. تمت المقارنة في حالة تنفيذ الاستعلام listStudents لاستعراض كامل السجلات المخزنة، وذلك مع تزايد عدد الزبائن المنفذين لهذا الاستعلام.



الشكل16: مقارنة أداء Jgroup المدمجة مع Hibernate مع أداء RMI المدمجة مع Hibernate مع تزايد عدد الزبائن.

✓ يلاحظ من الشكل(16) تزايد زمن التأخير بشكل خطي مع تزايد عدد الزبائن؛ وذلك في كل من Jgroup/Hibernate و RMI/Hibernate.

✓ عند تنفيذ الاستعلام من قبل زبون واحد فقط، تتفوّق RMI/Hibernate على Jgroup/Hibernate. (30ms في RMI بينما 55ms في Jgroup). يعود ذلك لوجود طبقات إضافية في Jgroup لمعالجة استدعاءات الطرائق الواردة (الاستعلامات).

✓مع تزايد عدد الزبائن يتفوّق أداء Jgroup/Hibernate على RMI/Hibernate. نظراً للآلية التي تتبعها Jgroup في استدعاء طريقة المجموعة الخارجي من النوع Anycast، حيث لا تقوم خدمة استدعاء طريقة المجموعة الخارجية في Jgroup باختيار المخدم نفسه للإجابة على كلّ استعلام مطلوب؛ فهي تقوم بتوجيه كلّ زبون إلى مخدّم مختلف ضمن مجموعة الغرض وذلك مع كلّ عملية استعلام مطلوبة ممّا يساعد على موازنة الحمل بين مخدمات مجموعة الغرض ويزيد من سرعة تنفيذ الاستعلامات (يبرز هنا فائدة تكرار قواعد البيانات باستخدام Jgroup).

الاستنتاجات والتوصيات :

a. تبدي Jgroup مرونة عالية في دمجها مع أدوات برمجية أخرى، مما يساعد على تطوير هذه المنصّة عبر تزويدها بمزايا جديدة.

b. يساهم إنشاء نسخ متعدّدة من قاعدة البيانات على Jgroup في زيادة توافرية الخدمة، كما يحسّن من أداء النظام في استجابته للاستعلامات المطلوبة وذلك مع وجود عدد كبير من الزبائن.

c. يتزايد الزمن اللازم لإجراء عمليات الإضافة والتعديل على قواعد البيانات المكررة باستخدام Jgroup مع إحدى أدوات ORM مع تزايد عدد نسخ المخدم، يعود ذلك إلى تنفيذ بروتوكولات الموثوقية المطلوبة لتحقيق تزامن المنظار وتحقيق تكامل البيانات بين النسخ المكررة.

d. يعتبر استخدام EclipseLink مع Jgroup لدمجها مع قاعدة بيانات أفضل من استخدام Hibernate، لأن النظام يبدي استجابة أسرع للاستعلامات المطلوبة عند استخدامه EclipseLink.

e. تم اختيار أشهر أداتين من أدوات مقابلة الغرض في هذا البحث؛ من الممكن دمج قاعدة بيانات مع Jgroup باستخدام أدوات أخرى مثل OpenJPA و JDO (Java Data Object) وغيرها، واختبار فيما إذا كان استخدام إحدى هذه الأدوات مع Jgroup أكثر فعالية من EclipseLink.

f. تعتبر إضافة طبقة دوام البيانات إلى طبقات مدير المجموعة المرتبط مع كل نسخة مخدم من الأعمال المستقبلية المقترحة. وفي حال التمكن من إضافة هذه الطبقة؛ ستتم المقارنة بين أداء المنصة مع الخدمة الجديدة وأداء المنصة عند استخدامها للأدوات المذكورة في هذا البحث.

g. يمكن الاستفادة من عملية دمج قاعدة بيانات مع Jgroup في كثير من التطبيقات الشبكية التي تقدّم خدمات موثوقة إلى زبائنها وتتطلّب تفاعلاً مع قواعد البيانات؛ كمكتبة بيع الكتب عبر الانترنيت (Bookshops). في مرحلة من مراحل هذا التطبيق يمكن للزبائن الاستعلام من قاعدة البيانات هذه عن جميع الكتب المتوفرة، والحصول على معلومات تتعلّق بإحدى هذه الكتب. كما يحتاج مدراء التطبيق في مرحلة ما إلى إضافة كتب جديدة وعرضها للبيع، بالإضافة إلى الحاجة لتعديل سجلات أحد الكتب، مثل تعديل عدد النسخ المتوفرة منه وذلك بعد بيع قسم منها.

المراجـع:

[1] UNIVERSITY OF BOLOGNA , 2008, June.2015 . <http://jgroup.sourceforge.net/index.html>.

[2] TUTORIAL POINT, 5May.2015. <http://www.tuturialspoint.com/>.

[3] ECLIPSE, 2016, 10January.2016. <http://www.eclipse.org/eclipselink/documentation/2.6/concepts/toc.htm>.

[4] ECLIPSE, 2016, 12January.2016. <http://www.eclipse.org/eclipselink/downloads/>.

[5] ECLIPSE, 2016, 14January.2016. <http://www.eclipse.org/eclipselink/documentation/2.6/ solutions/migrhib002.htm>.

[6] SOURCEFORGE ,2015, 5July.2015. <http://sourceforge.net/projects/hsqldb/files/>.

[7] SOURCEFORGE, 2015, 9July.2015. <http://sourceforge.net/projects/hibernate/files/hibernate3/3.6.4.Final/>.

[8] APATCHE ANT ,2014, 3February.2015 . <http://ant.apache.org/>.

[9] APACHE ,2012, 10July.2015. <http://logging.apache.org/log4j/1.2/>.

[10] SOFTONIC , 2009, 7June.2015.
<http://java-development-kit-jdk.en.softonic.com/>.

[11] MELING, H.; MONTRESOR, A.; HELVIK, B. E. and BABAOGLU, O. *'Jgroup/ARM: a distributed object group platform with autonomous replication management'*, *Softw. Pract. Exper.*, 38: 885–923. DOI: 10.1002/spe.853, 2008.

[12] MELING, H. *'An Architecture for Self-healing Autonomous Object Groups'.* University of Stavenger, Department of Electrical Engineering and Computer Science, N-4036 Stavenger, Norway, 2008.

[13] The Jgroup/ARM Project, 2008, 5May.2015 . <http://jgroup.sourceforge.net/index.html> .

[14] MINTER, D.; LINWOOD, J. *'Beginning Hibernate: From Novice to Professional'.* Apress,Inc, United States of America, 2006, pp. 11-25.

[15] DEITEL, H.M.; DEITEL, P.J. and SANTY, S.E. *Advanced Java 2 Platform: How To PROGRAM*. New Tersey: Prentice-Hall, Inc, 2002, pp. 1499-1571.

[16] VITENBERG, R.; KEIDAR, I.; CHOCKLER, G. and DOLEV, D. *'Group Communication Specifications: A Comprehensive Study'*. Technical Report CS99-31, Institute of Computer Science, The Hebrew Univ. of Jerusalem, 1999, pp. 4-7.

[17] BAN, B. *'JavaGroups:Group communication patterns in Java'*. Technical Report, Department of Computer Science, Cornell University, July 1998.

[18] MAFFEIS, S. *' The Object Group Pattern'.* In Proceedings of the 2nd USENIX Conference on Object-Oriented Technologies and Systems (COOTS), Toronto, Canada, 1996.