
Une approche de *benchmarking* des pratiques de gestion basées sur un middleware JMX pour les services et les applications

Abdelkader Lahmadi — Laurent Andrey — Olivier Festor

LORIA - INRIA Lorraine - Université de Nancy 2
615 rue du Jardin Botanique
F-54602 Villers-lès-Nancy, France
{Abdelkader.Lahmadi,Laurent.Andrey,Olivier.Festor}@loria.fr

RÉSUMÉ. Les communautés systèmes, systèmes distribués et applications réparties ont développé des suites de benchmarking pour un nombre considérable de leurs applications. Ces suites sont centrées sur les calculs intensifs, serveurs web et bases de données. Cependant, dans le monde de la gestion d'applications et de services, des plates-formes de benchmarking sont rares et même absentes pour des technologies de gestion comme JMX, le standard pour la gestion des applications et des services du monde Java. Cette absence de plates-formes de benchmarking pour les systèmes de gestion est acceptable pour des systèmes gérés avec des ressources considérables, mais l'est moins pour des systèmes gérés ayant des ressources contraintes. Nous proposons dans cet article une approche de benchmarking du middleware JMX basée sur son évaluation selon différentes pratiques de gestion. Dans ce but, nous avons identifié et isolé les paramètres relatifs à chaque pratique sous test ainsi que la définition de métriques d'évaluation de leurs performances.

ABSTRACT. The systems, distributed systems and applications communities have developed benchmarks for a number of domains, including CPU, web servers and databases. But no widely accepted benchmark exists for emerging management technologies such as JMX, the defacto standard for managing Java specific systems and services. While the lack of management benchmarks may be acceptable for a managed-application that runs on a single powerful server with abundant resources, such an omission is less acceptable for large communities of small devices within dynamic and constrained environments. In this work, we propose a benchmarking approach for the management practices based on the JMX middleware. Our approach identifies and isolates the parameters related to a specific management practice under test and the definition of their performance evaluation metrics.

MOTS-CLÉS : Benchmarking de JMX, Performance de JMX, Pratiques de gestion basées sur JMX.
KEYWORDS: JMX benchmarking, JMX performance, JMX-based Management practices.

1. Introduction

L'évaluation de performance par mesure (*benchmarking*) est une approche naturelle pour estimer la performance d'un système. La communauté des systèmes distribués a depuis longtemps mis en place des plates-formes pour le *benchmarking* de leurs domaines d'applications, notons à titre d'exemple : RUBiS, ECperf, SPEC JBB, Stock-Online et TPC-W. Ces applications sont souvent basées sur des technologies venant du monde Java (J2EE, EJB, servlets, etc). Or dans le monde Java, JMX (Java Management eXtension) [SUN 02, SUN 03] est devenu le standard *de facto* pour la gestion d'applications et de services et ne dispose à ce jour d'aucune infrastructure de mesure. L'architecture JMX est généralement couplée au plan fonctionnel de l'application à superviser et l'agent de supervision est implanté sous la forme d'un composant dans l'application. L'impact de cette intégration des deux plans, gestion et fonctionnel, sur les performances n'est pas négligeable même pour les gros serveurs d'applications disposant des ressources considérables [LAH 05a]. Dans la communauté de gestion de réseaux et de services l'évaluation des architectures de gestion est devenue une préoccupation croissante [PRA 04, PAV 04, LUD 97]. Le *benchmarking* a un rôle certain pour concevoir, améliorer et comparer ces systèmes de gestion ainsi que pour définir les modèles de performance associés aux pratiques de gestion les plus courantes. Dans un travail précédent [LAH 05b] nous avons présenté nos premiers résultats de calcul du coût de JMX au niveau de l'agent, essentiellement en terme de ressources consommées (CPU, mémoire et réseaux), sa résistance à la charge et la robustesse de l'implantation JMX. L'objectif de ce papier est de présenter notre approche de *benchmarking* et son instanciation sur une architecture JMX, ainsi que les difficultés rencontrées et leurs solutions respectives. Dans la section 2, nous présentons les objectifs et l'intérêt notre approche de *benchmarking* de l'architecture JMX. Puis nous détaillons les difficultés liées à sa mise en œuvre dans la section 3 ainsi que les métriques utilisées dans la section 4. Une conclusion et les travaux futurs sont présentés dans la section 5.

2. Approche de *benchmarking* d'une architecture JMX

Notre suite de *benchmarking* de JMX est orientée vers l'évaluation de performance des pratiques de gestion basées sur l'ensemble de paramètres du modèle JMX tels qu'ils peuvent être abstraits de la définition de ce *framework* [SUN 02, SUN 03, KRE 03]. Ces paramètres sont liés aux objets gérés de JMX, les *MBeans* (type, nombre, attributs), au type d'interactions (scrutation, notification), au type de connecteurs et les entités (voir figure 1). Les objectifs de notre approche de *benchmarking* sont les suivants : (i) évaluer le coût d'une pratique de gestion dans le modèle JMX en utilisant des métriques spécifiques, (ii) élaborer des campagnes de mesures qui identifient systématiquement des problèmes de performance dans une pratique de gestion, (iii) comparer les performances entre différentes pratiques de gestion, (iv) développer une suite de tests automatisée, flexible et modulaire, supportant les points précédents.

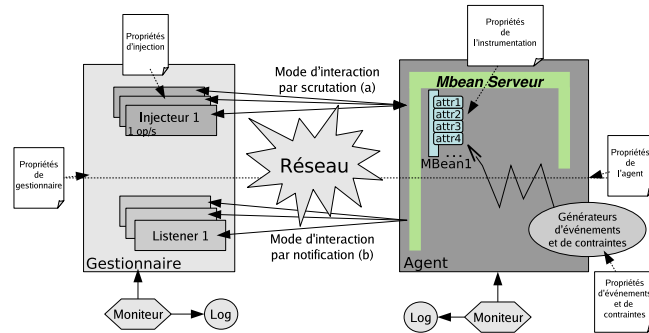


Figure 1. Approche de benchmarking d'une architecture JMX avec le modèle gestionnaire-agent sous les deux modes d'interactions : scrutation et notification.

3. Benchmarking de JMX : difficultés et solutions

Notre approche de *benchmarking* se base sur la terminologie utilisée dans [JAI 91, page 61] où notre système sous test (SUT) est une pratique de gestion avec l'ensemble de ses paramètres associés. Les paragraphes suivants détaillent les principales difficultés rencontrées ainsi que leur résolution.

3.1. Abstraction et déploiement des paramètres d'une pratique de gestion

Une étape importante lors du *benchmarking* d'une pratique de gestion est de définir clairement les *facteurs de test* : des paramètres que l'on juge significatifs et que l'on fait varier dans des séries de mesures. Nous avons définis ces paramètres d'une façon générique pour qu'on puisse comparer les performances d'une pratique avec d'autres technologies de gestion (SNMP, NetConf, CIM, WBEM). Dans ce but, ces paramètres sont relatifs aux trois couches de l'architecture JMX (instrumentation, agent et gestionnaire), valables aussi pour les autres technologies de gestion. Au cours de la réalisation d'un test de *benchmarking* sur la plate-forme physique de test, ces paramètres sont déployés sur les entités de JMX (les gestionnaires et les agents) sous la forme de fichiers de propriétés. Nous distinguons ainsi, 4 types de fichiers de propriétés relatifs à une pratique de gestion :

- Les propriétés de la couche instrumentation incluant le nombre de MBeans (un ou plusieurs), le nombre d'attributs dans chaque MBean, le type de ces MBeans et le type de leurs attributs ;
- Les propriétés de la couche agent incluant le type de connecteur (RMI, SOAP), le lien d'invocation du service JMX sur le MBean serveur et son modèle d'intégration dans le service géré ;

– Les propriétés de la couche gestionnaire incluant le nombre d’agents, les propriétés du connecteur pour chaque agent (type et lien de service), le mode d’interaction (scrutation ou notification) et le modèle de distribution de la fonction de gestion (centralisé, faiblement décentralisé ou fortement décentralisé) ;

– Les propriété de la charge d’évaluation incluant la fréquence d’injection de la charge en terme d’attributs par seconde, le type d’opérations de gestion (Get, Set ou invocation) avec leur taux de représentativité, leurs opérations facultatives et leurs granularité (nombre d’attributs par opération).

Les propriétés de la charge sont déployées sur les entités de gestion responsables de son injection.

3.2. Contraintes d’une pratique de gestion

Une pratique de gestion intervient dans plusieurs points du cycle de vie d’un service ou d’une application gérée [BRO 04]. Cette intervention, selon la pratique de gestion et son algorithme associé, se manifeste sous plusieurs formes. Les pratiques de gestion interviennent pour la configuration de certaines variables de l’application gérée ou pour la collecte périodique des variables d’états de l’application gérée au cours du son de cycle de vie ; etc. Chacune de ces interventions possède un objectif et des contraintes. La collecte périodique de variables d’états d’une application gérée possède comme objectif, la détection d’anomalies et de problèmes. Cependant, elle a comme contraintes : la performance propre de la pratique de gestion (un nombre considérable de réponses de supervision peut engendrer l’écroulement du gestionnaire, une latence importante provoque le dépassement du délai limite de résolution du problème), et le comportement propre de l’application gérée (sa performance et sa disponibilité) pour résoudre le problème après sa détection. (voir figure 2). Une façon de modéliser la contrainte relative à l’application gérée, est de définir son comportement comme étant une trajectoire dans un espace multi-dimension où chaque dimension représente une variable d’état du système géré. Lors du déplacement du système géré sur cette trajectoire, sa qualité de service acceptable (QoS) est définie dans une région spécifique de cet espace. Si le système géré sort de cette région, les pratiques de gestion interviennent à nouveau pour le ramener dans cette région spécifique. Une façon de quantifier cette trajectoire est de la définir comme étant la productivité de l’application gérée [LAH 05a]. Dans notre approche du *benchmarking* d’une pratique de gestion, la contrainte relative à l’application gérée est présente ou absente selon l’objectif de l’évaluation. Dans des tests synthétiques [JAI 91, page 50] d’une pratique de gestion, cette contrainte est absente. Seule la limitation propre à la pratique de gestion est présente. Ce genre de tests permet d’identifier les problèmes de performance qui peuvent émerger dans une pratique de gestion sans prise en considération de l’état de services gérés. Cependant, dans des tests réalistes la présence de cette contrainte permet d’identifier l’impact de l’état du service géré sur la performance de la pratique de gestion. Si le service géré est chargé (charge due aux utilisateurs du service), la performance d’une pratique de gestion ne sera pas la même que dans le cas d’un service

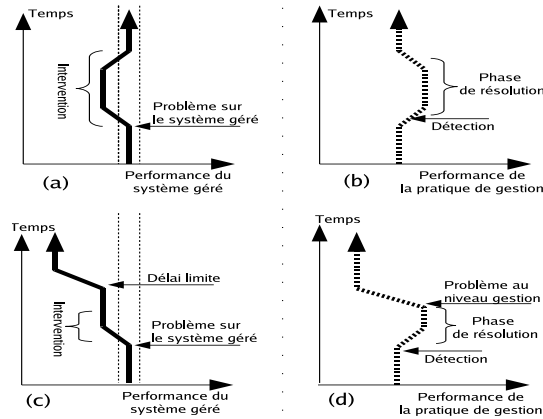


Figure 2. Les contraintes d'une pratique de gestion : (a,b) performance et disponibilité du système géré et (c,d) la performance propre de la pratique de gestion.

géré non chargé. Dans le paragraphe suivant nous décrivons la façon d'implanter dans notre plate-forme de *benchmarking* ces contraintes liées à une pratique de gestion.

3.3. Les injecteurs de la charge et des contraintes

Tout travail de *benchmarking* nécessite la définition du système d'injection de la charge (*workload*). Nous avons utilisé une approche similaire à celle de [CEC 02] pour développer nos systèmes d'injection de la charge relative à une pratique de gestion. L'approche est basée sur l'utilisation de *threads* d'injections où chaque *thread* représente une opération unique de gestion. Une opération peut être soit une opération de lecture (*Get*), d'écriture (*Set*), d'invocation (*invoke*) ou de génération d'une notification (*sendNotification*). Dans le cas d'un mode d'interaction par scrutation les *threads* d'injection sont implantés au niveau du gestionnaire. Chaque *thread* est responsable de l'envoi d'une opération par unité de temps. Ainsi, le nombre total des *threads* représente le taux d'injection défini dans le fichier de propriétés relatifs à la charge. Dans le cas d'un mode d'interaction par notification, les *threads* d'injection sont présents sur l'agent. Chaque *thread* est responsable de l'envoi d'une notification par unité de temps. Ainsi, le taux d'injection est une fonction (produit) du nombre total de *threads* multiplié et du nombre de *listeners* enregistrés par les gestionnaires. Selon le modèle de distribution de la fonction de gestion entre les agents et les gestionnaires, les *threads* d'injection sont répartis sur des gestionnaires intermédiaires ou sur certains agents pour accomplir certaines fonctions de gestion portant sur les valeurs des attributs des MBeans (somme, moyenne, maximum ou minimum).

Nous avons utilisé un autre ensemble de *threads* pour modéliser le comportement du service géré et l'impact de la gestion sur ce comportement. Ils sont implantés soit dans le même processus que l'agent dans le cas où le modèle d'intégration est de type composant ou noyau selon la terminologie de [KRE 03] ou dans un processus indépendant dans le cas où le modèle d'intégration est de type démon.

4. Benchmarking de JMX : métriques de performance

Une pratique de gestion est évaluée selon trois axes : opérationnel, coût et qualité. Pour le *benchmarking* de JMX, nous avons scindé ces trois axes d'évaluation sur les trois couches de l'architecture JMX. Les métriques liées à la couche instrumentation mesurent le coût de JMX du point de vue des ressources gérées. Elles incluent la consommation de ressources (CPU et Mémoire) par les MBeans. Ces métriques font partie de l'axe d'évaluation de coût. Les métriques liées à la couche agent mesurent sa robustesse à la charge injectée par les fonctions de gestion sous différents modes d'interactions (scrutation ou notification), son passage à l'échelle en terme de nombre de MBeans et son coût du point de vue gestion. Ces métriques font partie des axes : opérationnel et coût. Elles incluent le trafic réseau généré par l'agent et sa consommation CPU et mémoire propre à la gestion. Les métriques liées à la couche gestionnaire mesurent son passage à l'échelle en terme de nombre d'agents, la latence par type d'opération et la précision de données de gestion associée à une pratique vis à vis des exigences des algorithmes de gestion qui la dirigent.

Le *benchmarking* d'une pratique de gestion, nécessite la sélection des axes d'évaluation qui permettent de capturer sa performance et son coût. Cette sélection dépend essentiellement de l'objectif de l'évaluation (comparaison, surcoût ou pertinence) et des exigences en terme de performance de son algorithme de gestion associé. Par exemple, une pratique de supervision utilisant la scrutation nécessite l'ensemble des métriques relatives aux deux axes coût et opérationnel. Cependant, les métriques de l'axe qualité prennent de l'importance dans le cas d'un algorithme de supervision réactif où la précision de données collectées ainsi que la qualité de la latence deviennent critiques pour déclencher les opérations de gestion nécessaires à la résolution d'un problème.

5. Conclusions et travaux futurs

Dans cet article nous avons présenté notre approche de *benchmarking* d'un middleware de gestion basé sur la technologie JMX. Dans cette approche, nous proposons de mesurer la performance et le surcoût d'un système de gestion selon la pratique de gestion utilisée pour gérer un système. Une pratique de gestion est définie sous sa forme la plus simple comme étant l'ensemble des paramètres placées dans les différentes entités de gestion pour réaliser des tâches de gestion sur le système géré. Dans JMX, cet ensemble de paramètres est relatif à son modèle en trois couches (gestionnaire, agent, instrumentation). Notre approche permet l'isolation et la définition de ces paramètres afin qu'on puisse analyser la performance et le surcoût d'une pratique de gestion spécifique sur la performance du système géré. Nous avons aussi identifié

les contraintes d'une pratique de gestion. Ces contraintes sont liées au comportement du service géré, essentiellement sa performance et sa disponibilité, et au modèle de distribution des fonctions de la gestion entre les gestionnaires et les agents.

6. Bibliographie

- [BRO 04] BROWN A. B., HELLERSTEIN J. L., « An Approach to Benchmarking Configuration Complexity », *Proceeding of the 11th ACM SIGOPS European Workshop, Leuven, Belgium*, september 2004.
- [CEC 02] CECCHET E., MARGUERITE J., ZWAENEPOEL W., « Performance and scalability of EJB applications », *OOPSLA '02 : Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, New York, NY, USA, 2002, ACM Press, p. 246–261.
- [JAI 91] JAIN R., *The art of Computer Systems Performance Analysis*, John Wiley & Sons, Inc, 1991, ISBN : 0-471-50336-3.
- [KRE 03] KREGER H., HAROLD W., WILLAMSON L., *Java and JMX : Building Manageable Systems*, Addison-Wesley, January 2003, ISBN : 0672324083.
- [LAH 05a] LAHMADI A., ANDREY L., FESTOR O., « On the Impact of Management on the Performance of a Managed System : a JMX-Based Management Case Study », SCHÖNWALDER J., SERRAT J., Eds., *16th IFIP/IEEE Distributed Systems : Operations and Management, DSOM 2008, Barcelona, Spain*, vol. LNCS 3775, Springer-Verlag's Lecture Notes in Computer Science (LNCS), 24-26 October 2005, p. 24-37.
- [LAH 05b] LAHMADI A., ANDREY L., FESTOR O., « Performances et résistance au facteur d'échelle d'un agent de supervision basé sur JMX : Méthodologie et premiers résultats », *Colloque GRES 2005 : Gestion de REseaux et de Services, Luchon, France*, vol. 6, Mar 2005, p. 269-282, ISBN : 2-9520326-5-3.
- [LUD 97] LUDERER G. W. R., KU H., SUBBIAH B., NARAYANAN A., « Network Management Agents Supported by a Java Environment », *Integrated Network Management*, vol. 86, Chapman & Hall, 1997, p. 790-790.
- [PAV 04] PAVLOU G., FLEKAS P., GOVERIS S., LIOTTA A., « On management technologies and the potential of Web services », *Communications Magazine, IEEE*, vol. 42, 2004, p. 58-66, ISSN : 0163-6804.
- [PRA 04] PRAS A., DREVER T., DE MEENT R. V., QUARTEL D., « Comparing the Performance of SNMP and Web Services-Based Management », *eTransactions on Network and Service Management (eTNSM)*, vol. 1, n° 2, 2004.
- [SUN 02] SUN, « JavaTM Management Extensions, Instrumentation and Agent Specification, v1.2 », october 2002, Maintenance Release 2.
- [SUN 03] SUN, « JavaTM Management Extensions(JMXTM) Remote API 1.0 Specification », october 2003, Final Release.