Berliner Schriften zu modernen Integrationsarchitekturen

herausgegeben von
Prof. Dr.-Ing. habil. Andreas Schmietendorf
Hochschule für Wirtschaft und Recht Berlin, FB II

Band 4

Makram Hanin,
Andreas Schmietendorf,
Wolfram Greis (Eds.)

Application Performance Management in complex integration architectures: JEE, SOA and Web 2.0

ceCMG APM Workgroup

Shaker Verlag
Aachen 2010

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Printed in Germany.

ISBN 978-3-8322-8692-7
ISSN 1867-7088

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

Towards the Maturity Era of IT Performance

In the beginning, we had low-level system programming. Programs consisted of instructions directly accessing computing resources. Then, performance became a major concern to the program and system designers, especially because of the limited computing capacity available.

Later, when the first business-relevant systems appeared, IT performance concerns continued to be strongly related to the resource usage in terms of CPU, memory and I/O. However, performance and tuning specialists became involved in design decisions as well as in development, with more structured and procedural programming languages.

The rapid evolution of computing hardware gave IT professionals huge opportunities in terms of resources and performance, while the business relevance of IT systems increased. A kind of an informal "convention" took place: developers were to concentrate on functional requirements, while system engineers were to tune the system and allocate resources later in production. However, universities kept teaching complexity theory, data structures and algorithm optimization methods, which were used by only a minority of IT professionals and in rare situations. The first Software Engineering models, such as waterfall or even the iterative models, concentrated mainly on functional requirements, to optimize the productivity of developers.

The age of the client-server architecture added new challenges to the scope of IT performance. Understanding, evaluating and managing concurrency on the server and the network became tedious tasks for network and system engineers. Network sniffers and probes, server and database agents, log and trace files emerged as performance management techniques. However, the performance of production environments remained a concern: software test engineers (not architects or developers) started using load testing tools to evaluate the stability and resource usage of applications with a large number of users (especially with the emergence of the web). System and network engineers were also involved, since they owned the access to agents, probes and log files.

At this stage, the increasing rate of project failures at the end of the development phase or in the deployment phase, because of performance issues (mostly related to design decisions), led to the first attempts to take performance into account in both of these phases. C. Smith coined the term "Software Performance Engineering", where architecture and design decisions could be evaluated using models combining use case steps and computing resources requirements. Due to the complexity and large effort required to build and maintain such models, model-based performance engineering was not widely adopted in the industry: this meant that, system engineers and some testing engineers continued to work on performance issues like fire-fighters; they were sometimes lucky to be able to fix (or hide) per-

formance issues. The “e-enthusiasm” experienced in the early days of the Internet induced a management of performance that was both reactive and expensive: excessive and unplanned resources had to face major performance issues in production. “Lucky” projects survived, while “unlucky” ones died!

The severe failure of some prominent e-projects, combined with the economic restrictions that followed the Internet bubble burst, were behind the emergency of new performance engineering concepts, where a systematic approach replaced the early prediction models. Such an adoption of a measurement-based approach, used to evaluate performance requirements modeled using scenarios (a combination of critical use cases and usage patterns and profiles) was enabled by the wide use of the Java Platform. Indeed, the generalization of virtual machines between the OS and the application, with garbage collection and later component containers, made it necessary to use measurement techniques such as profiling and bytecode injection. Today, these measurement- and lifecycle-based performance engineering approaches are recognized by experienced and mature IT organizations. However, a systematic integration into existing organization, processes and technical environments requires changes at several levels, as well as governance approaches, to improve costs and sustainability within the organization.

This booklet on the topic of performance is intended to be a first concrete participation of our Application Performance Management Workgroup within the CeCMG, towards more maturity and standards of IT Performance within the community.

The first paper introduces a proven and widely-used methodology for Performance Engineering, based on measurement. The second one compiles several real-world experiences of measurement-based performance engineering in the financial industry. The tedious task of understanding and handling memory issues in Java and how the capabilities of Performance Engineers can be extended, are dealt with in the third paper.

The second part of the booklet deals with some technical aspects related to Java Performance: the trade-off between security and performance when using web services, and the impact of the choice of a remoting technique such as EJB or Spring on performance.

The third part of the booklet tackles innovative approaches to increase the value of measurement-based performance engineering: a Java-based load testing approach is presented to help in the evaluation of the performance of JEE designs or critical JEE components. Another paper shows how to further leverage a continuous integration platform by adding performance checks and dashboards to the artifacts produced by nightly builds. In addition to the industrialization approaches mentioned in the latter paper, “more intelligence” is needed in the field of IT Performance, when dealing with huge amounts of performance data collected every

day in production. To reach that goal, a self-learning performance management approach is presented in the last paper.

Working towards more maturity in the IT Performance area requires more experience and real-world reports from professional users of performance engineering, more studies and benchmark reports when it comes to new frameworks and technologies, and finally more innovation to handle technical and organizational barriers. We consider this as our mission at the ceCMG APM Workgroup, and we are looking forward to more feedback and cooperation.

Basel, March 2010

Makram Hanin - adhoc International
Email: makram.hanin@adhoc-international.com

Table of contents