# LOAD BALANCING: AN AUTOMATED LEARNING APPROACH

BY

PANKAJ MEHRA

Recom Technologies/NASA Ames Research Center
Moffett Field, CA 94035, USA

BENJAMIN W. WAH

University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

World Scientific Publishing Co. Pte. Ltd., 1995

# PREFACE

With the advent of cluster computing, there is growing need for tools and techniques that will transparently and intelligently harness distributed resources. Commercial software packages such as LSF and a plethora of public-domain software from Condor to NQS have made remote execution convenient, but the policies governing remote execution are buried deep inside the software manuals and involve parameters too numerous to set manually. In this world of heterogeneous systems and rapidly evolving configurations, factory setting of parameters is not possible. Therefore, the dream of intelligent resource management can only be realized through the development of self-tuning software.

This book examines automated learning of dynamic load-balancing strategies, addressing issues ranging from measurement and generation of computer workloads to automated refinement of load indices and load-balancing policies via machine learning. It describes SMALL (Systematic Method for Automated Learning of Load-balancing strategies), an integrated learning system that can be used for efficiently and automatically tuning commercial-grade load-balancing software.

*Dynamic workload generation*. We present DWG (Dynamic Workload Generator), a program for generating realistic and reproducible background workloads with high accuracy and high resolution. DWG's most innovative feature is its ability to mimic the behavior of a background workload in the presence of a foreground job. The critical functions for measurement and generation of resource-utilization levels are implemented in each site's operating-system kernel, ensuring low overhead. Actual resource-usage patterns of a distributed system can be captured and replayed with high fidelity, test jobs introduced at precise instants, and their completion time measured accurately. Such experiments can be repeated, running the job at a different site each time but under the same background workload. Since a policy's performance is completely determined by where it schedules an incoming job, alternative policies can be compared *under identical loading conditions*. Thus, DWG allows us to perform reproducible load-balancing experiments, a facility hitherto unavailable to experimenters in this area.

*Learning consistent and comparable performance indices*. DWG tracks the utilization levels of each site's key resources: computational, primary memory, secondary storage, and communication. It provides a precise account of the loading conditions prevalent just before a job begins execution. DWG also measures the completion time of that job when it finishes execution. Such ‘‘before’’ and ‘‘after’’ data suffice for learning to compare alternative destinations for incoming tasks. The problem is one of *learning to compare functions of multivariate time series*.

We have adopted the neural-network architecture used for learning evaluation functions in Tesauro's renowned backgammon-playing program [160, 161], and added various smoothing and extrapolation capabilities to his method of learning to compare multivariate functions. We also present an innovative learning algorithm that obviates the ‘linking of weights’ [93] required by Tesauro's original architecture. This modification allows us to use off-the-shelf neural-network-simulation packages [52]. Extensive statistical tests on the load-index functions learned using our *comparator neural network* reveal high positive correlations (at 99% level of significance) between the true ranking of sites and the one induced by the new index functions. Thus, if load indices were computed right before each decision point, and if they could be communicated instantly across sites, then we could (with high confidence) select the destination having the the least completion time for each incoming job.

The comparator neural network transforms the multi-dimensional and highly dynamic measurements provided by DWG into smooth one-dimensional load indices that can be efficiently communicated over the network, and compared across sites in a meaningful fashion, unlike the traditional load average.

*Automated tuning of policy parameters* Wah, *et al.* have developed a domain-independent population-based learning system, called TEACHER (TEchniques for Automated Creation of HEuRistics), which also accommodates point-based learning [165]. TEACHER rationally schedules limited learning time between *generating* new parameter sets and *testing* the promising ones in the current population. We have integrated into TEACHER a point-based learning procedure for adjusting certain parameters of load-balancing policies.

We find that the optimal setting of parameters is sensitive to the overheads involved in migrating tasks and communicating load indices from site to site. Further, we report experiments with different intervals of computation for the load-index function; these affect the average age of load-index values. Since the quality of a load index degrades with age, the setting of policy parameters is also sensitive to the interval of load-index computation. In all these cases, the integrated (population- and point-based) learning system described above is able to quickly determine the appropriate parameter set with high confidence, given data about the completion times of test jobs and information about various overheads and delays.

Thus, what used to be an unsystematic, manual, and tedious process of discovering good parameter settings by trial and error has been replaced by a systematic, automated, and efficient process of population-based learning. Likewise, what used to be the ad hoc process of adjusting the thresholds of policies based on human experience has now been replaced by an automated performance-driven process of point-based learning.

By carrying out an in-depth investigation into the use of artificial intelligence for developing self-tuning load-balancing software, we have conclusively demonstrated not only that we can develop good load-balancing policies but also that we can do so systematically, automatically and efficiently. We believe that it is only a matter of time before machine learning techniques will be used for improving other similar computer-system functions, such as branch prediction in RISC microprocessors, data prefetching in NUMA multiprocessors and file placement in distributed information servers.

**Acknowledgements**

We are grateful for the support of National Aeronautics and Space Administration Grants NCC 2-481 and NAG 1-613, National Science Foundation grant MIP 92-18715, and a gift from Sumitomo Electric Industries in the preparation of this book.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES