IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITB8694

Chapter 5

Scheduling and Latency – Addressing the Bottleneck

Michael J. Oudshoorn
University of Adelaide, Australia

ABSTRACT

As e-business applications become more commonplace and more sophisticated, there is a growing need to distribute the server side of the application in order to meet business objectives and to provide maximum service levels to customers. However, it is well known that the effective distribution of an application across available resources is difficult, especially for novices. Careful attention must be paid to the fact that performance is critical – business is likely to be lost to a competitor if potential customers do not receive the level of service they expect in terms of both time and functionality. Modern globalised businesses may have their operational units scattered across several countries, yet they must still present a single consolidated front to a potential customer. Similarly, customers are becoming more sophisticated in their demands on e-business systems and this necessitates greater computational support on the server side of the transaction. This chapter focuses on two performance bottlenecks: scheduling and communication latency. The chapter discusses an adaptive scheduling system to automatically distribute the application across the available resources such that the distribution evolves to a near-optimal allocation tailored to each user, and the concept of Ambassadors to minimize communication latency in wide-area distributed applications.

INTRODUCTION

The effective distribution of an e-business application across available resources has the potential to provide significant performance benefits. However, it is well known that effective distribution is difficult, and there are many traps for novices. Despite these difficulties, the average programmer is interested in the benefits of distribution, provided that his/her program continues to execute correctly and with well-defined failure semantics. Hence we say that the programmer is "all care." Nevertheless, the reality is that the average programmer does not want to be hampered with managing the distribution process. He/she is not interested in dealing with issues such as the allocation of tasks to processors,

optimisation, latency, or process migration. Hence we say that the programmer is "no responsibility." This gives rise to the "all care and no responsibility" principle of distribution whereby the benefits of distributed systems are made available to the average programmer without burdening him or her with the mechanics behind the distributed system.

The customer, or end user, of an e-business application has similar demands to the E-business applications developer, namely, the need for performance. As end users become more sophisticated and place more complex and computationally intensive demands on the e-business application, the need for distribution across multiple processors become necessary in order to obtain increased throughput so as to meet these demands.

As businesses themselves become more globalised and distributed, no one business unit provides all of the information/resources required to satisfy a complex request. Consider a business that has interests in steel, glass and rubber products. It is likely that all of its products are manufactured in the same place, but all of its products may be related to motor vehicles (sheet steel, windscreens, rubber hoses and floor mats). A vehicle producer may want to place an order for components for 1,000 vehicles. The vehicle producer will act as the client and attempt to order the necessary components from the manufacturer in a single E-business transaction. The e-business application may, however, need to contact several business units within the organisation to ensure that the order is met. The problem of latency across a wide area network now becomes apparent.

The ongoing Alchemy Project aims to provide automated support for the "all care and no responsibility" principle. The Alchemy Project aims to take user applications and perform appropriate analysis on the source code prior to automatically distributing the application across the available resources. The aim is to provide a near-optimal distribution of the application that is tailored to each individual user of the application, without burdening the applications developer with the details of, and issues related to, the physical distribution of the application. This permits the developer to focus on the issues underlying the application in hand without clouding the matter with extraneous complications. The project also examines issues surrounding fault tolerance, load balancing (Fuad & Oudshoorn, 2002), and distributed simulation (Cramp & Oudshoorn, 2002)

The major aim of the Alchemy Project is to perform the distribution automatically. This chapter focuses on two aspects of the project – namely, the scheduling of tasks across the available distributed processors in a near-optimal manner, and the minimisation of communication latency within distributed systems. These two features alone provide substantial benefits to distributed application developers. Existing applications can be easily modified readily to utilise the existing benefits provided, and new applications can be developed with minimal pain. This provides significant benefits to developers of e-business systems who are looking to develop distributed applications to better harness the available resources within their organisations or on the internet without having to come to terms with the intricacies of scheduling and communication within hand-built distributed systems. This frees developers from the need to be concerned with approaches such as Java RMI (Sun Microsystems, 1997) typically used to support distribution in e-business applications, and allows developers to concentrate more on the application itself.

The chapter focuses on scheduling through the discussion of an adaptive system to allocate tasks to available processors. Given that different users of the same application may have vastly different usage patterns, it is difficult to determine a universally efficient distribution of the software tasks across the processors. An adaptive system called ATME is introduced that automatically allocates tasks to processors based on the past usage statistics of each individual user. The system evolves to a stable and efficient allocation