# Load Balancing for On-Demand Business Event Processing

STEPHEN L. OLIVIER
JAN F. PRINS
University of North Carolina at Chapel Hill

Simple, flexible, and scalable Business Event Processing (BEP) is a key approach to define and support rapidly changing data-driven business models. Events of significance to the business may arise asynchronously and from many disparate sources. They may vary greatly in quantity, content, and required processing. Event processing might need to scale to millions of events per second. In this paper, we consider how to design a BEP system that employs scheduling and load balancing techniques from run time systems for emerging parallel languages.

Categories and Subject Descriptors: D.2.11 [**Software Engineering**]: Software Architectures

General Terms: Design, Performance

Additional Key Words and Phrases: Event processing, work stealing

## 1. INTRODUCTION

Business Event Processing (BEP) responds to both internal and external events of significance to a business, launching computational tasks that operate on related enterprise data. For example, when a customer clicks on a support service link on a web site, the service history may be examined for issues in products that customer has recently purchased. Events often vary widely in size and scope and may arrive sporadically [IBM Software Group 2008]. In this paper, we investigate a scalable approach to meet performance goals for BEP.

## 2. A WORK SCHEDULING MODEL FOR BEP

We posit a basic two-level work scheduling model for BEP. A set of applications participate in BEP. At the base level, applications are scheduled to processors, each with affinity to co-located data resources. Executing within each application are one or more tasks triggered by internal or external business events or generated by preceding tasks. An application may be phased. For example, an online auction may experience a high-demand phase when bidding accelerates near closing time. All tasks in a particular phase must complete before the succeeding phase begins.

In practice, scalable BEP operates on cluster platforms comprised of multicore nodes. Among the available processors, each application is assigned multiple nodes

Author's address: S. Olivier, Dept. of Computer Science, UNC, Chapel Hill, NC 27599-3175 USA.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20YY ACM 0000-0000/20YY/0000-0001 $5.00

that comprise its processor set for execution. Data is distributed and replicated across nodes in the processor set to accommodate both data size and application demand. While the processor set and data distribution for an application are fixed within phases, they may change at phase boundaries. If application resource demands increase, more processors may be dynamically added to the application and data further replicated. If resource demands decrease, processors can be reallocated to other applications or may even be suspended to save power.

## 3. TASK SCHEDULING STRATEGIES

Task scheduling assigns tasks to particular processors in an application's processor set. Two general rules allow for scheduling based on locality of reference. First, a task triggered by incoming events should execute on a processor on a node containing the data to be accessed by the task. Second, a task generated by some preceding task should execute on the same processor as the parent task.

Since task arrival rates, execution times, and data access patterns vary, continuous load balancing is necessary to fully utilize the available processors during each phase of an application. Thus, we allow the system to deviate from the two general rules stated above. If a processor is idle, a task assigned to another processor should be reassigned to the idle processor for execution. Under the scheduling scheme known as work stealing [Blumofe and Leiserson 1994; Blumofe et al. 1995], an idle processor ("thief") is responsible for finding tasks from another processor ("victim") to reassign to itself. To exploit locality when possible for BEP, the thief should first attempt to steal a task from one of the other processors on its node before attempting to steal from processors in other nodes in the processor set for that application. If the thief must resort to stealing from a processor on a remote node, it should whenever possible steal a task which accesses data local to its node.

Overheads–time not spent executing tasks–should be minimized for scalable BEP. An efficient protocol for work stealing in large-scale search problems is detailed in [Olivier and Prins 2008]. Efficient termination detection is also provided. In the context of BEP, termination detection is required to ensure that all tasks, whose completion times are not known a priori, are complete before beginning the next phase of the application. In BEP, tasks may be fine-grained. If so, multiple tasks may be stolen at once. Stealing many tasks at once lowers the overhead costs of stealing, but stealing too many tasks at once leads to load imbalance.

This is a sketch of an software architecture and implementation techniques for highly scalable BEP. We are currently exploring ways to integrate these techniques with IBM WebSphere solutions for BEP [IBM Software Group 2008].

## REFERENCES

Blumofe, R., Joerg, C., Kuszmaul, B., Leiserson, C., Randall, K., and Zhou, Y. 1995. Cilk: An efficient multithreaded runtime system. In *PPoPP '95: Proc. of 5th ACM SIGPLAN Symposium on Principles and Practices of Parallel Programming*. ACM, 207–216.

Blumofe, R. and Leiserson, C. 1994. Scheduling multithreaded computations by work stealing. In *Proc. of 35th Annual Symposium on Foundations of Computer Science*. IEEE, 356–368.

IBM Software Group. 2008. Empowering the business to sense and respond: Delivering business event processing with IBM WebSphere business events.

Olivier, S. and Prins, J. 2008. Scalable dynamic load balancing using upc. In *ICPP '08: Proc. of 37th International Conference on Parallel Processing*. IEEE, 123–131.