Newcastle University

COMPUTING SCIENCE

Scalable and Responsive Event Processing in the Cloud

Visalakshmi Suresh, Paul Ezhilchelvan and Paul Watson

TECHNICAL REPORT SERIES

No. CS-TR-1251 June 2011

TECHNICAL REPORT SERIES

No. CS-TR-1251 June, 2011

Scalable and Responsive Event Processing in the Cloud

V. Suresh, P. Ezhilchelvan and P. Watson

Abstract

Event processing involves the continuous evaluation of queries over streams of events. Response-time optimization is traditionally done over a fixed set of nodes using a variety of techniques. The emergence of cloud computing makes it easy to acquire and release computing nodes as required. Leveraging this facility, we propose an architecture that meets a specified response-time target against fluctuating event arrival rates by dynamically and adaptively drawing an adequate amount of computing resources from a cloud platform. In this paper we explore how to achieve dynamism and adaptation, by regarding the entire processing engine of a distinct query as the atomic unit for optimization and reconfiguration.

© 2011 Newcastle University.
Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

SURESH, V., EZHILCHELVAN, P.,WATSON. P

Scalable and Responsive Event Processing in the Cloud
[By] V. Suresh, P. Ezhilchelvan, P. Watson
Newcastle upon Tyne: Newcastle University: Computing Science, 2011.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1251)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1251

Abstract

Event processing involves the continuous evaluation of queries over streams of events. Response-time optimization is traditionally done over a fixed set of nodes using a variety of techniques. The emergence of cloud computing makes it easy to acquire and release computing nodes as required. Leveraging this facility, we propose an architecture that meets a specified response-time target against fluctuating event arrival rates by dynamically and adaptively drawing an adequate amount of computing resources from a cloud platform. In this paper we explore how to achieve dynamism and adaptation, by regarding the entire processing engine of a distinct query as the atomic unit for optimization and reconfiguration.

About the authors

Visalakshmi is a research associate in school of computing science.Her research interests are in the areas of the pervasing sensing information management and real time event processing systems. Currently she is working in EPSRC funded SiDE project and TSB funded Swich-EV project to build a scalable and distributed event processing infrastructure using cloud computing. She created real time and historic decision support systems for traffic and environment pollution monitoring in the EPSRC funded MESSAGE project.

Paul Devadoss Ezhilchelvan received Ph.D. degree in computer science in 1989 from the University of Newcastle upon Tyne, United Kingdom. He received the Bachelor of Engineering degree in 1981from the University of Madras, India, and the Master of Engineering degree in 1983 from the Indian Institute of Science, Bangalore. He joined the School of Computing Science of the University of Newcastle upon Tyne in 1983 where he is currently a Reader in Distributed Computing.

Paul Watson is Professor of Computer Science and Director of the North East Regional e-Science Centre. He graduated in 1983 with a BSc (I) in Computer Engineering from Manchester University, followed by a PhD in 1986. In the 80s, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University, where he has been an investigator on research projects worth over £13M. His research interests are in scalable information management. This includes parallel database servers, data-intensive e-science and grid computing. In total, he has over thirty refereed publications, and three patents. Professor Watson is a Chartered Engineer, a Fellow of the British Computer Society, and a member of the UK Computing Research Committee.

Suggested keywords

SCALABLE

Scalable and Responsive Event Processing in the Cloud

Visalakshmi Suresh, Paul Ezhilchelvan, Paul Watson

Abstract:
Event processing involves the continuous evaluation of queries over streams of events. Response-time optimization is traditionally done over a *fixed* set of nodes using a variety of techniques. The emergence of cloud computing makes it easy to acquire and release computing nodes as required. Leveraging this facility, we propose an architecture that meets a specified response-time target against fluctuating event arrival rates by dynamically and adaptively drawing an adequate amount of computing resources from a cloud platform. In this paper we explore how to achieve dynamism and adaptation, by regarding the entire processing engine of a distinct query as the atomic unit for optimization and reconfiguration.

Introduction:

Event processing is characterized by the continuous processing of streamed data tuples or *events* in order to evaluate, in a timely manner, the queries deployed by decision support systems. Event sources can, for example, be pervasive sensors; while their number of sources is normally fixed in an application, the rates at which they generate events can vary widely and often unpredictably, driven purely by the external processes they monitor. Similarly, the number of queries that need to be evaluated over the streams can also vary over time. Thus, an event processing system with real-time performance requirements must meet targeted response times despite being subjected to these two types of varying loads.

A query evaluation can be modeled as a directed acyclic graph wherein nodes are *operators* and the links are event streams that are either raw or partially processed by the preceding operators. Early commercial systems, such as Aurora [5], used single server solutions and proposed a variety of techniques, such as multi-query optimization, for response-time optimization. Later, distributed solutions [1] handled the optimization problem as a load-balancing issue over a *fixed* set of nodes: moving query operators to nodes where their resource requirements are best met and thereby achieving the best overall response time. Such solutions however have two drawbacks: they require the placement of low-level probes to measure operator execution rates, queue lengths, extent of disk-writes, etc., making their implementation hard and certainly not portable across heterogeneous machines; at times, the load has to be 'shed' to meet response time targets [4]. In this paper, we propose a responsive, event-processing architecture that avoids both these drawbacks; it leverages the advantages offered by, and is best suited for implementation in, cloud computing platforms.

System Description:

The system processes several event streams, each emanating from a distinct source. These streams are denoted as $s_1, s_2, s_3, .., s_\infty$, and $\Sigma = \{s_1, s_2, .., s_\infty\}$. The system evaluates *q* queries, $Q_1, Q_2, .., Q_q$. The state machine that implements the directed acyclic graph (DAG) for Q_i is called the *event processing network*, EPN_i. Evaluating Q_i involves processing one or more event streams and the set of all streams input to EPN_i is denoted as S_i. Note that there is no implication that two EPNs have distinct S, e.g., S_i and S_j may overlap. Also, an input stream to EPN_i can be an output stream from another EPN_j; if so, $S_i \not\subset \Sigma$. If all inputs to EPN_i are output streams from other EPNs, then $S_i \cap \Sigma = \{\}$. An EPN is also associated with a performance target T. It is said to be *distinct* if any one of its three attributes is unique: DAG, S or T. We consider all EPNs to be distinct.

The system itself is made up of *n* hosts or *nodes* drawn from a cloud computing platform. A high bandwidth network interconnects these nodes. The number of hosts used, *n*, is increased (or decreased) when the load increases (or decreases) to an extent that the current *configuration* over these *n* nodes is deemed inadequate (or more than strictly necessary) to meet the performance targets. A configuration is a mapping from the set of EPNs onto the set of hosts. Figure 1 shows a configuration where, EPN1, EPN2 and EPN3 are mapped to (e.g., hosted by) node 1, and the rest of the EPNs are mapped to a distinct node. The system has a configuration scheduler, CS for short, which decides the configuration appropriate to the load conditions and

performance targets associated with the EPNs. For brevity, we assume that CS is centralized, hosted on a single node. The workings of CS are discussed later.
The Architecture:

The front end of our system has a messaging oriented middleware (MOM), to which all the computing nodes in the cloud that might be included into the system are asynchronously connected. Event sources publish their data streams to MOM. When CS announces the EPN to host mapping, each host node subscribes to relevant input streams and transmits its relevant output streams, if any, to nodes of EPNs which use them as inputs. In Figure 1, nodes 4 and 5 will supply their relevant outputs to node 2; all other output streams will be transmitted to an archival warehouse.

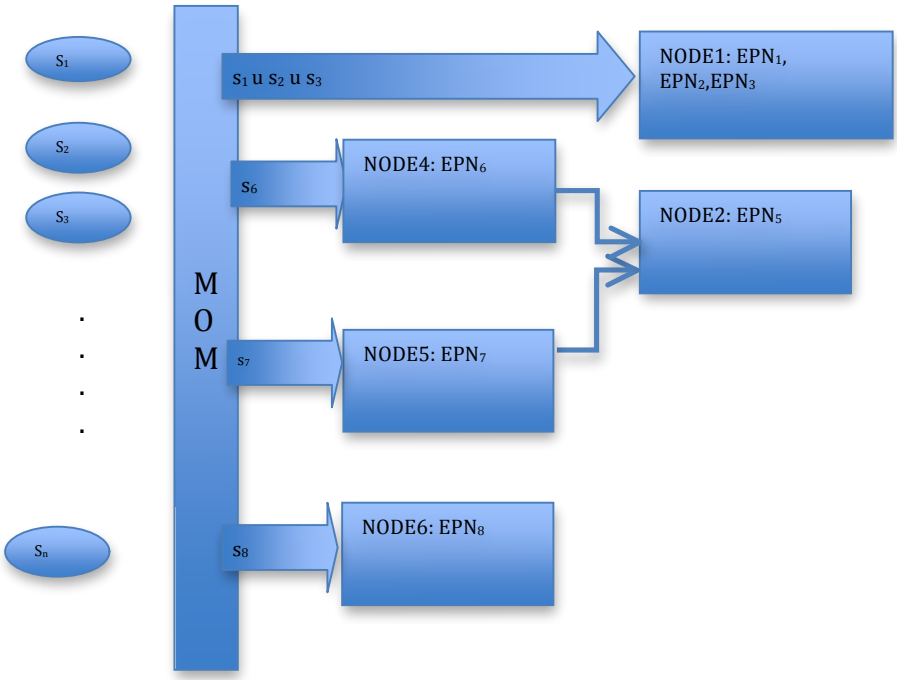

Figure 1: The Event Processing Architecture

Central to our architecture is the configuration scheduler CS, and an outline of its design is sketched below, with complete details left to the full paper. In a nutshell, each EPN takes macro-level measurements of its own performance and reports periodically to CS which constructs a global view and attempts to re-map EPNs to host nodes, if response times of some EPNs are either above or far below their target levels; in the former case, new nodes may have to be brought in and in the latter some of the existing nodes may be released. Note that re-mapping EPNs requires support mechanisms and extracts a cost, both of which are not considered here.

Configuration Scheduler and Response-time Optimization:

CS design requires an off-line standardization of EPN response times, the principles of which are described below. For brevity, let us assume that (i) the performance of two EPNs hosted on identical platforms is determined by their DAGs and stream input arrival rates and, (ii) the response time of a given EPN increases *linearly* with its input arrival rates when all other conditions remain the same. We choose one of the EPNs to be used as the 'standard' EPN, and we fix a meaningful arrival rate as the standard rate SR.

Denote the response time of the 'standard' EPN when all its input streams are arriving at SR as R. Let R_i be the response time of the EPN_i when it is running on the same platform in identical conditions including all its input streams arriving at SR. The computational *load factor* λ_i for EPN_i is defined as R_i/R. A large λ_i, for example, indicate that EPN_i operates on a relatively more number of streams and/or its operators take a relatively longer time to execute. Note that λ_i is solely dependent on the DAG and S_i of EPN_i and hence it fixed over the lifetime of EPN_i.

Each EPN_i monitors its response time RT_i and the arrival rates of each of its input streams; the maximum RT_i and the maximum arrival rate (AR_i) of its input streams observed over the reporting interval are sent to CS; the latter metric is expressed as the arrival *rate factor* $\rho_i = AR_i/SR$. CS marks EPN_i for eviction from its current node if $RT_i > T_i$ (where T_i is the response-time target for EPN_i) and computes the standard free-time due to eviction of EPN_i as (RT_i) $\lambda_i\rho_i$; otherwise, it marks EPN_i to be retained in its current node and computes the standard free-time due to EPN_i: (RT_i - T_i) $\lambda_i\rho_i$. A node can *additionally* host EPN_j if T_j > (total standard free-time of that node)/ $\lambda_j\rho_j$; otherwise, it cannot.

Using the above-mentioned (albeit oversimplified) principles, EPNs are re-mapped, possibly taking in new nodes or releasing existing nodes at the end of the re-mapping process. Of course, we can have a situation wherein EPN_i is not meeting its target T_i even though its host node is hosting no other EPN. It can occur, for example, if ρ_j is very large. On these occasions, we resort to Intra-EPN parallelism as depicted in Figure 2: EPN_i is hosted on multiple nodes (two nodes in Figure 2) and *each* input stream in S_i is temporally split and distinct (and temporally disjoint) splits are input to distinct hosts. For example, an input stream s_i can be split as: (t to t+100) tuples as s_i^1, (t+101 to t+200) tuples as s_i^2, (t+201 to t+300) tuples as s_i^3, and so on. The splits s_i^1, s_i^3, s_i^5, … are sent to EPN_i^1 (in that order) and the rest to EPN_i^2, halving the arrival at each destination. The results from EPN_i^1 and EPN_i^2 are to be 'reduced' to the final version.

Our approach of Inter-EPN parallelism corresponds to the well-known MapReduce paradigm. In the context of VLDBs, it is known as intra-operator (or partitioned) parallelism[2]. The existing solutions for intra-operator parallelism use multi-query optimization by scheduling the incoming workload in a fixed number of nodes. Operators are shared by several queries based on a dynamic data scheme to maximize resource utilization.

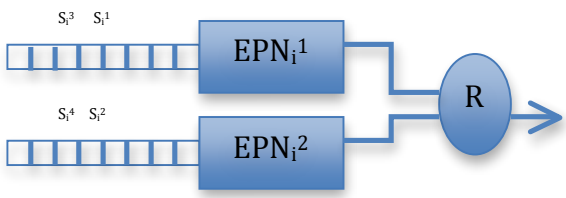

Figure 2: Intra-EPN parallelism

Conclusions:
We have outlined an architecture and the design principles for deploying an event processing systems on Cloud platforms in a scalable and responsive manner. Currently, we are developing the full system which is designed to process event streams generated by more than 600 devices in the ambient kitchen [3] originating from 12 different kitchens in multiple locations for the purpose of activity recognition.

References:

1. M. Balazinska, H. Balakrishnan, and M. Stonebraker. Load management and high availability in the medusa distributed stream processing system. In Proceedings of 2004 ACM SIGMOD international conference on Management of data, SIGMOD '04
2. Managing Intra-operator Parallelism in Parallel Database Systems, Manish Mehta, David J DeWitt, Proceedings of 21st VLDB conference, Zurich, Switzerland,1995.
3. C. Pham, T. Plo¨tz, and P. Olivier. A dynamic time warping approach to real-time activity recognition for food preparation. In Proceedings of the First international joint conference on Ambient Intelligence, AmI'10, pages 21–30, 2010.
4. Zhou, Y. Ooi, B. C. Tan, K.-L. Wu, J.,Efficient Dynamic Operator Placement in a locally Distributed Continuous Query Systems, published in springer-verlag, lecture notes in computer science, 2006, numb 4275, pages 54-71.
5. Daniel J. Abadi, D Carney, U Cetintemel, M Cherniack, C Convey, S Lee, M Stonebraker, N Tatbul,S Zdonik, Aurora: a new model and architecture for data stream processing, published in The VLDB Journal (2003) / Digital Object Identifier (DOI) 10.1007/s00778-003-0095-z