2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Load Balance Strategies for DEVS Approximated Parallel and Distributed Discrete-Event Simulations

Alonso Inostrosa-Psijas*, Veronica Gil-Costa‡, Roberto Solar† and Mauricio Marín*†
*Universidad de Santiago, Chile
†Yahoo! Research Latin America, Chile
‡Universidad Nacional de San Luis, Argentina, Email: gvcosta@unsl.edu.ar

***Abstract*—DEVS is a formalism for modeling and analysis of discrete event systems. PDEVS is an extension of DEVS for supporting Parallel and Discrete Event Simulation (PDES). PCD++ is a simulation platform that supports parallel simulations of DEVS models, where the model component allocation in processors is not an automatic process. This can be a time consuming task requiring knowledge of communication patterns among model components. In this paper, we propose and evaluate different allocation strategies devised to improve load balance of parallel DEVS simulations. The experimentation is made on a Web search engine application whose workload is featured by dynamic and unpredictable user query bursts, and high message traffic among processors.**

***Keywords*-Parallel Discrete Event Simulation; PDES; DEVS;**

I. Introduction

Parallel discrete event simulation (PDES) has been widely used in different application areas as an efficient tool to study performance of large scale systems. A PDES program consists of a collection of logical processes (LPs), each simulating a different portion of the model. LPs communicate to each other by exchanging timestamped event messages.

A major difficulty of PDES is to efficiently process all events in parallel in global time-stamp order. In the literature, dealing with causality related events issues are two major strategies: *conservative* and *optimistic* [1]. In conservative algorithms, the process is blocked until all execution conditions are satisfied. In optimistic algorithms, the process will continue even if some execution condition is not fulfilled, but including mechanisms to recover from causality issues. The optimistic protocol is relaxed in [2] by removing the mechanisms to recover from causality errors leading to approximate simulations.

DEVS is a simulation formalism for modeling and simulating discrete event systems [3], [4]. We work with a PDES platform called Parallel CD++ (PCD++) [5] that supports parallel simulation of DEVS models.

In this paper we propose and evaluate different load balancing strategies devised to automatically allocate the elements of a DEVS model to the distributed set of processors.

The remaining of this paper is organized as follows. Section II presents the background. In section III we present the WSE application modeled with PCD++. In Section IV we describe the allocation strategies evaluated in this work. In section V we present our experimental results and conclusions are discussed in Section VI.

II. Background

A. Parallel discrete event simulation

Parallel discrete event simulation (PDES) consists of executing a single simulation program on a parallel computer. The simulation program is decomposed into a set of concurrent processes called *logical process* (LP) that may be independently executed on different processors. Each LP is composed by a separate set of state variables of the simulation program. The communication between LPs is performed by exchanging timestamped messages or events.

In PDES, synchronization protocols are used to avoid that the parallel simulation violates the local causality constraint (events must be executed in timestamp order). Synchronization protocols can be classified as *conservative* or *optimistic* [1]. A new approach for *approximate parallel simulations* is presented in [2]. It relaxes causality constraints of *conservative protocols* permitting out of time execution of straggler events, leading to faster execution of simulations.

B. Service Based Web Search Engines

Modern WSE are composed of services deployed in computer clusters. Typical services are: Front Service (FS) that handles submitted user queries by routing them to the appropriate service, it also manages the delivery of final results to the user. The Cache Service (CS) implements a partitioned distributed cache storing previously computed results for the most popular queries. And the Index Service (IS), holds an index of the document collection and is responsible of delivering partial results to the Front Service. These services are deployed on clusters of computers and its processing nodes are allocated in racks connected via network switches. To reduce query response times and increase throughput of queries, most services are implemented as arrays of $P \times R$ processing nodes, where P is the level of partitioning and R is the level of replication of each service partition, as seen in Figure 1.

1066-6192/15 $31.00 © 2015 IEEE
DOI 10.1109/PDP.2015.13

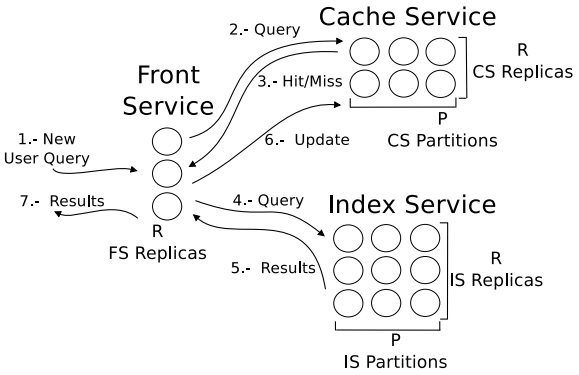


Figure 1. Query processing

Processing nodes from FS can communicate with every processing node of the other services. Thus, the communication pattern tends to be uniform which makes this a difficult application to be partitioned.

C. DEVS

The Discrete-Event System Specification (DEVS) [3] formalism provides the means for describing discrete-event systems. DEVS provides two different types of elements to model a real system: Atomic and Coupled. Atomic models allow to represent the behavior of a system whereas coupled models permit to represent its structure. Coupled models are composed of two or more atomic or coupled models [3]. DEVS allows model reuse of existing models.

There are several implementations of DEVS, one of them is Parallel CD++ (PCD++), with versions that use conservative and optimistic protocols [5], [7], [8].

III. Modeling WSE with PCD++

Coupled models were not considered in the design of our simulation model since its atomic elements are highly interconnected and its use would lead to an increase in communication levels (specially if atomic models belonging to composed models are assigned to different LPs)

Our DEVS model of the WSE is shown in Figure 2. Shaded areas are only intended to help the reader to identify the different services of the model. The Query Generator generates and delivers user queries through its output ports (out_1,...,out_n) to the FS replicas selecting them in a round-robin fashion. Query inter-arrival time is simulated using an exponential distribution. Once a FS replica receives a new query (on its "In" input port), it immediately sends it to a single replica of the CS through one of its "outCS_i_j" output ports. The FS replica chooses the $i-th$ partition by computing a hash function on the query terms, and the $j-th$ replica in the selected partition is chosen in a round-robin form. Then, the CS replica responds the query to the FS replica that sent it in the first place with a "hit/miss". In case there was a "hit" the query now contains the results for that search, the FS responds with the query results to the user and the processing for that query is finished. However, if the answer was a "miss", the FS replica needs to get the results from the IS. The query is sent to the P IS partitions (the replicas among the partitions are also selected in a round-robin mode) through its "outIS_i_j" output ports. Once the FS replica has received the partial results from all the selected IS replicas it performs a merge operation producing the the top$-k$ final document results.

IV. Partitioning Strategies

The main objective behind the partitioning strategies is to achieve good workload balance, since its consequences are a reduction in execution time, an increase in speedup and a reduction in network communication (which translates as lower probabilities of straggler events). These goals are achieved by properly allocating at the same LP the simulated entities that communicate the most to each other and by having similar amounts of simulated entities at each LP. Each LP is exclusively hosted into a physical processor. We use the following allocation strategies of entities to LPs:

- **User Defined**: This strategy requires knowledge about the application and the communication patterns to properly allocate entities evenly at the different LPs.
- **RB**: A communication graph is built based on the amount of queries sent by each pair of simulated entities. This graph is partitioned with the METIS software[1] by means of a multilevel recursive biscetion approach. As a result, each partition groups the simulated entities that communicate the most.
- **Hash-Based**: The Fowler-Noll-Vo non-cryptographic hash function was used. FNV hashes are designed to be fast while maintaining a low collision rate.
- **RR**: Entities are assigned in a round-robin fashion.
- **RND**: Entities are assigned in a random fashion.
- **SP**: We apply a spectral clustering algorithm [9] in order to obtain k partition groups.

In PCD++ the allocation of atomic models to the different LPs is defined by the user in an offline fashion by explicitly specifying the processor location for each model component. Entity migration among LPs is not allowed. Also, PCD++ lacks of dynamic load balance capabilities. Under these constraints is that the study of different strategies for partitioning the simulation model becomes relevant.

V. Evaluation

A. Experimental Setting

Partitioning strategies were evaluated by simulating a WSE DEVS model (as in Figure 2) to obtain metrics related to model simulation accuracy and performance of the parallel simulator itself.

[1] http://glaros.dtc.umn.edu/gkhome/views/metis

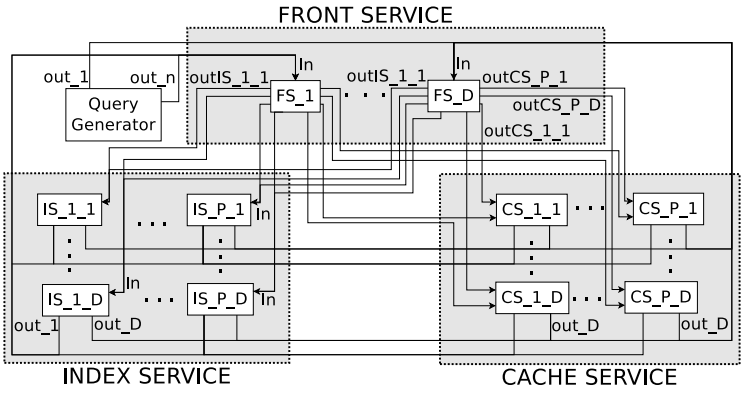


Figure 2. DEVS model of a WSE

A services configuration is a user defined simulation parameter indicating the levels of partitioning and replication of each service of the WSE model (i.e., <3,4,5,6,7> specifies a WSE with 3 FS, 4 CS partitions and 5 replicas per partition, and 6 IS partitions with 7 replicas each). The model used in our experiments was specified with a <32,32,8,16,15> configuration simulating 528 physical processors. Our simulator model supports different query arrival rates because in practice user queries do not arrive at evenly-spaced time intervals. Different query arrival rates were used in order to avoid the saturation of the simulated services, that is, with workload over 80%. It is important to simulate unsaturated services, because the metrics studied in this work cannot be well estimated when services run under overloaded conditions because results are unpredictable and do not follow a stable behavior.

To correctly simulate the behavior of a WSE and its relevant costs, the simulator uses actual query logs, document posting lists and ranking times. In particular, we use a log of $36,389,567$ queries submitted to the AOL Search service. It was pre-processed according to the rules described in [10].The costs of the most significant operations of a WSE were measured on production hardware [11] and then used by the DEVS model to properly simulate the query processing process.

Experiments were executed on the Deepthought cluster at the ARS Laboratory at Carleton University using up to sixteen HP PROLIANT DL Servers with Dual 3.2Ghz Intel Xeon processors and 2GB of RAM memory. In the following experiments we use the PCD++ parallel DEVS simulator implementing an approximated optimistic protocol [2] using the MPI message passing library.

B. Simulation Accuracy Evaluation

Throughput (processed queries per second) results show that the strategies behave well with models of different sizes.

According to results in table I strategy **RB** provides the lowest error levels as the amount of processors involved in the parallel simulation increases. However, the percentage of throughput error of the rest of the strategies only raises to a maximum of almost 0.21%.

Table I
THROUGHPUT PERCENTAGE ERROR

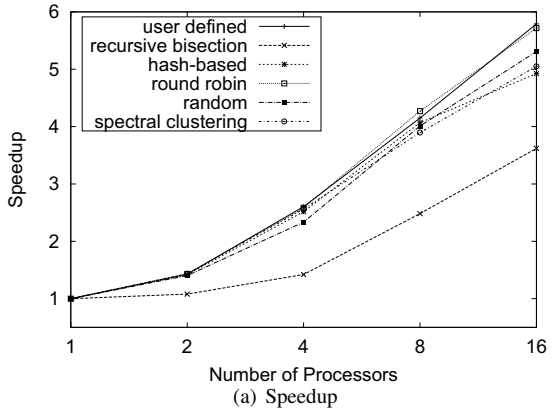
Strategy	2 Nodes	4 Nodes	8 Nodes	16 Nodes
User Defined	**0.102%**	0.170%	0.186%	0.186%
RB	0.105%	**0.134%**	0.193%	**0.182%**
FNV1A32	0.107%	0.175%	0.193%	0.195%
RR	0.118%	0.166%	**0.177%**	0.211%
RND	0.106%	0.163%	0.208%	0.189%
SP	0.107%	0.172%	0.192%	0.212%

As more processors are involved in the parallel simulation there should be more probabilities of straggler events. Nevertheless, this is not always the case with strategies **RB** and **RND** (as shown in Table I) as they tend to group simulated services with higher communication interaction to the same partition, reducing overall network communication.

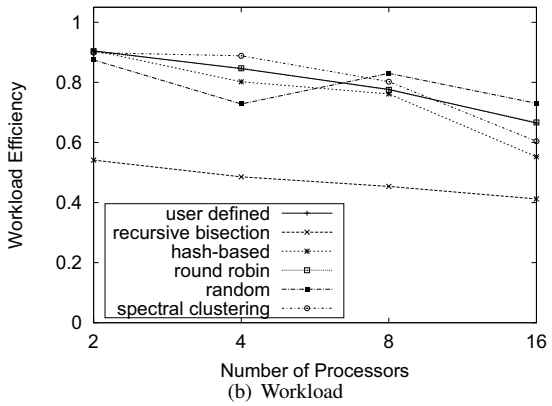
C. Simulator Performance Evaluation

Figure 3 shows speedup and workload efficiency of our simulating model. Strategies show similar results but **RB**, mainly because it has partitions of different sizes.

As expected, it can be observed in figure 3(a) that **User Defined** and **RR** strategies obtain the highest speedup levels since these strategies uniformly distribute entities to LPs. **Hash-based** and **SP** strategies also present good performance results. Workload efficiency is calculated as $\text{W\_Efficiency} = \frac{1}{P}\sum_{i=1}^{P}\frac{w_i}{max(w_i)}$. w_i is the amount of events processed by the LP running on processor i, and P is the number of processors involved in the parallel simulation. A workload efficiency close to 1, means that all LPs tends to do the same amount of work. It can be observed in figure 3(b) that the workload efficiency decreases as more processors are involved in the parallel simulation. **RB** presents the lowest levels of workload efficiency since



(a) Speedup



(b) Workload

Figure 3. Speedup vs Workload balance.

this strategy does not guarantee a uniform assignment of entities to LP. Therefore, some LPs process more events than others, producing a high variance among the workload of the physical processor. **User defined** and **RR** practically overlap each other, showing to be strategies with the best workload efficiency, since they distribute the entities among the parallel processors more uniformly than the rest.

As more processors are involved in the simulation, it can be observed in figure 3 that even when **RB** has the poorest load balance and speedup levels, it presents the lowest straggler events ratio among the studied strategies (as shown in figure 4) since it effectively allocate (no matter how unbalanced) the simulated processing services that communicate the most at the same LP, effectively reducing overall communication costs.

VI. Conclusions

In this work we proposed and evaluated different static partitioning strategies devised to automatically allocate DEVS model components to different processors for the PCD++ parallel simulation framework. In particular, the **RR** and **User Defined** strategies present the best choices for implementing an automatic allocation strategy in PCD++

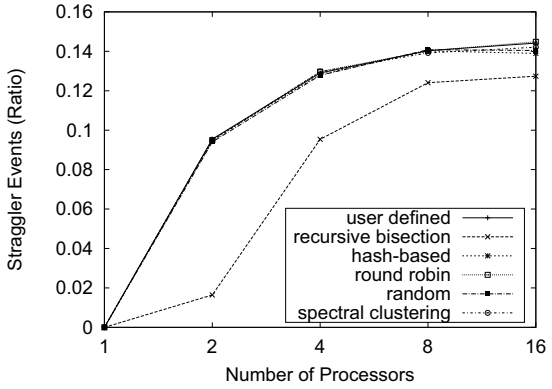


Figure 4. Straggler Events

for DEVS. We performed our experimentation using the approximate optimistic protocol, nevertheless, the allocation strategies are also directly applicable to the other conservative and optimistic protocols in PCD++.

References

[1] R. M. Fujimoto, *Parallel and Distribution Simulation Systems*, 1st ed. NY, USA: J. Wiley & Sons, Inc., 1999.

[2] M. Marin, V. Gil-Costa, C. Bonacic, and R. Solar, "Approximate parallel simulation of web search engines," in *SIGSIMPADS*, 2013, pp. 189–200.

[3] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation*, 2nd ed. Orlando, USA: Academic Press, Inc., 2000.

[4] G. Wainer, *Discrete-Event Modeling and Simulation: a practitioner approach*. CRC Press. Taylor and Francis, 2009.

[5] Q. Liu and G. Wainer, "Parallel environment for devs and cell-devs models," *SIMULATION*, vol. 6, no. 83, pp. 449–471, 2007.

[6] K. Schloegel, G. Karypis, and V. Kumar, "Graph partitioning for high-performance scientific simulations," in *Sourcebook of Parallel Computing*, 2003, pp. 491–541.

[7] Q. Liu, "Distributed optimistic simulation of devs and cell-devs models with pcd++," Master's thesis, Carleton University, 2006.

[8] S. J., "New algorithms for the parallel cd++ simulation environment," Master's thesis, Carleton University, 2007.

[9] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances In Neural Information Processing Systems*. MIT Press, 2001, pp. 849–856.

[10] Q. Gan and T. Suel, "Improved techniques for result caching in web search engines," in *WWW*, 2009, pp. 431–440.

[11] V. G. Costa, M. Marín, A. Inostrosa-Psijas, J. Lobos, and C. Bonacic, "Modelling search engines performance using coloured petri nets," *Fundam. Inform.*, vol. 131, no. 1, pp. 139–166, 2014.