Scaled Real–Time Parallelization for DEVS Simulation of Hybrid Systems

Federico Bergero and Ernesto Kofman and François Cellier

CIFASIS, CONICET - FCEIA, UNR - Argentina
ETH Zürich - Switzerland

July 24th, 2012

Summary

- We introduce a novel parallelization technique for discrete event simulation of hybrid systems.
- The models are first split into several sub–models that are concurrently simulated on different processors.
- In order to avoid the cost of the synchronization between processes, the simulation time of each sub–model is locally synchronized in a real–time fashion with a scaled version of physical time, which implicitly synchronizes all sub–models.
- An extended version of the present work, including a more detailed description of the implementation together with an analysis of theoretical error introduced by these techniques can be found in [BKC12].

Introduction

Previous Concepts

- DEVS stands for Discrete EVent System specification [ZPK00]. A DEVS model processes an input event trajectory and –according to that trajectory and its own initial conditions– provokes an output event trajectory. Discrete time systems can be easily expressed in DEVS.
- Quantized State Systems (QSS) [CK06] is a recently developed family of numerical integration methods for continuous and hybrid system simulation. QSS methods are based on the idea of state discretization instead of time discretization and their application to a ODE system results in a DEVS model.
- *PowerDEVS* is a general purpose open source DEVS simulator that implements the complete family of QSS methods on which this work is based and can be run under RTAI (an open source RTOS).

Parallel Discrete Event Simulation

- Parallel Discrete Event Simulation (PDES) is a technique that can be used for the simulation of large DEVS models in multi–core or cluster environments. In PDES, the model is first split into several subsystems called physical processes. Then, the processes are simulated concurrently on different logical processors (LP).
- Compared to simulating a large model in a sequential manner, PDES reduces the simulation time but it introduces a new problem related to the need of synchronization between processes. If the LPs are not correctly synchronized, the simulation may receive events out of order (with time–stamps lower than the actual simulation time), which can lead to an incorrect result. This is called a *causality constraint*.

Synchronization Algorithms

There are many approaches to overcome this problem.

- One of the earliest was the CMB algorithm proposed by Chandy, Misra, and Bryant. CMB implements a *conservative* synchronization where none of the LPs advances its simulation time until it is safe.
- A different approach is that proposed by Jefferson, where an *optimistic* synchronization mechanism is introduced relaxing the causality constraint. The LPs are allowed to advance their simulation time as fast as they can, and proper actions are taken when inconsistencies are detected.
- Finally, a technique called NOTIME explores the effects of not synchronizing the processes at all.

Scaled Real Time Synchronization

Basic Idea SRTS

DEVS models resulting of the application of QSS methods to large–scale continuous systems have particular features:

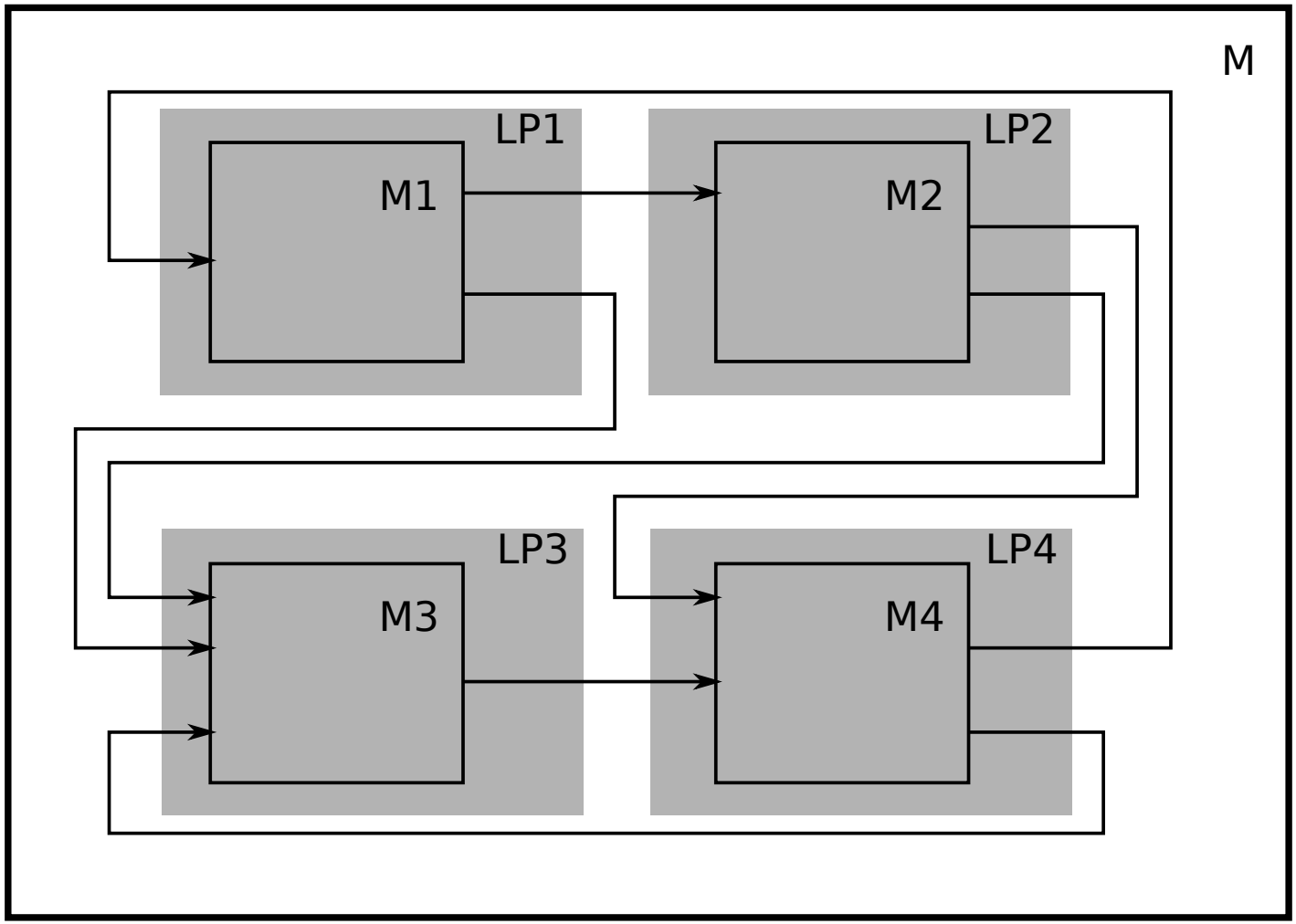
1. Each event originates at a quantized integrator and is instantaneously transmitted to other quantized integrators through the static functions depending on the corresponding state variable.
2. The events received by quantized integrators change their future evolution, but they do not provoke instantaneous output events.
3. Different quantized integrators provoke output events at different times.
4. Each event represents a section of a piecewise polynomial function.

On the one hand, the second feature implies that synchronization is necessary and that techniques like NOTIME cannot be used. Using a *conservative* algorithm almost nothing is gained by parallelization. An *optimistic* algorithm will allow the simulation time of other LPs to advance, but as soon as the effect of an event is propagated to them, a rollback mechanism will need to take place.

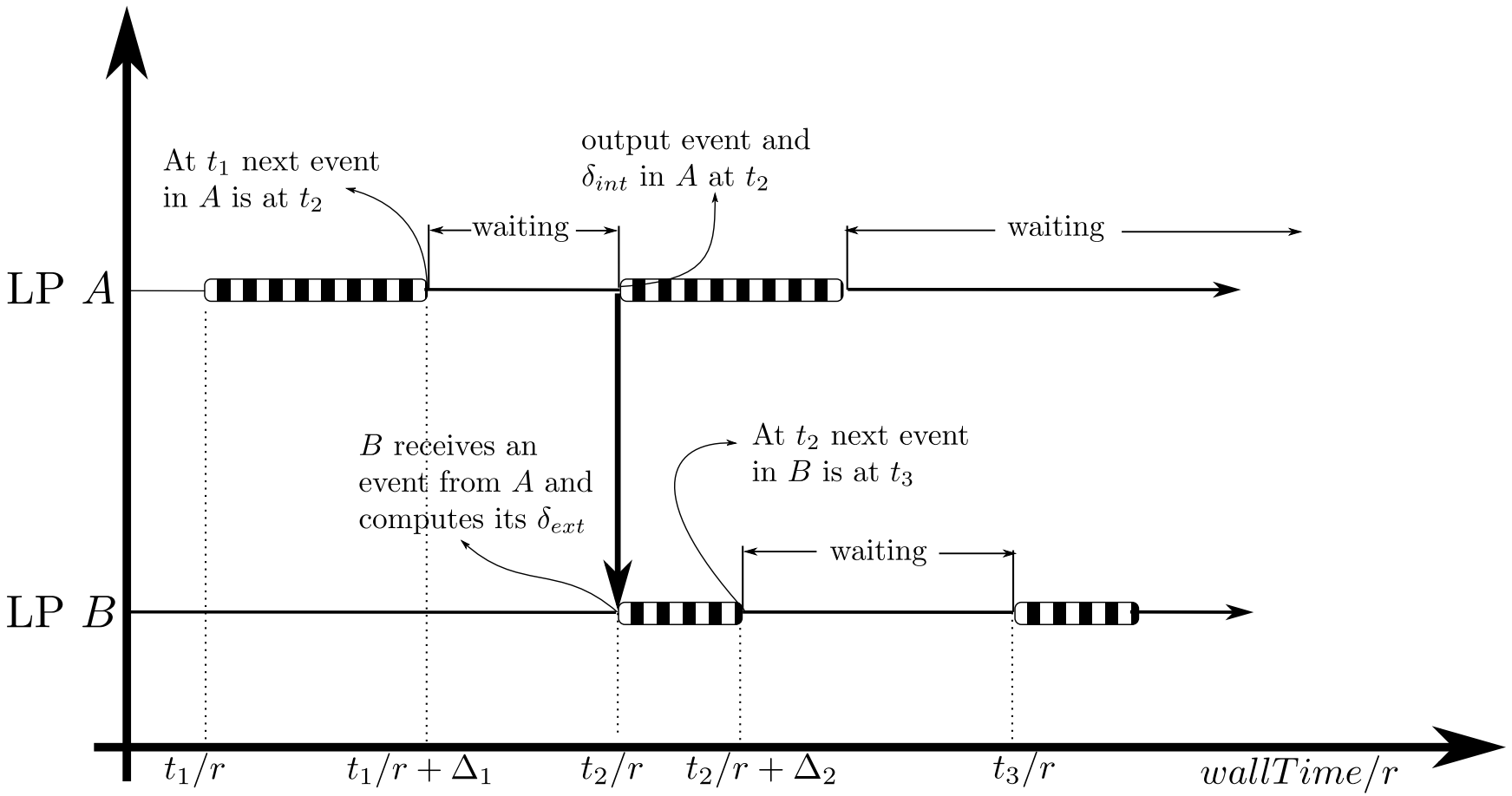
Basic Idea SRTS –cont.

- To overcome these difficulties, we propose a technique, in which we perform a non–strict synchronization. Recalling that events represent sections of polynomials, errors in the synchronization imply a bounded numerical error in the trajectories transmitted from one process to another.
- As with other PDES techniques, SRTS divides the overall model into sub–models representing physical processes, and each of these processes is simulated on a different LP.
- In order to avoid the inter–process synchronization overhead, instead of synchronizing the simulation time between all LPs, we synchronize in SRTS the simulation time of each LP with a scaled version of the physical (wall–clock) time. We scale the physical clock by a magnitude of r.
- The only communication between different LPs takes place when events are transmitted from one LP to another.

Structure of Coupling SRTS



Time diagram of SRTS



Adaptive Scaled Real Time Synchronization

Basic Idea - ASRTS

- The main drawback of SRTS is that the real–time scaling factor r must be chosen by the user.
- Unless the user performs some previous trial and error experiments, he would be hard–pressed to know a priori, which value to assign to the real–time scaling factor.
- Another problem is that r remains constant during the entire simulation. In many cases, the simulation workload changes with time, and it makes sense to adapt r according to the actual workload.

Basic Idea - ASRTS - cont.

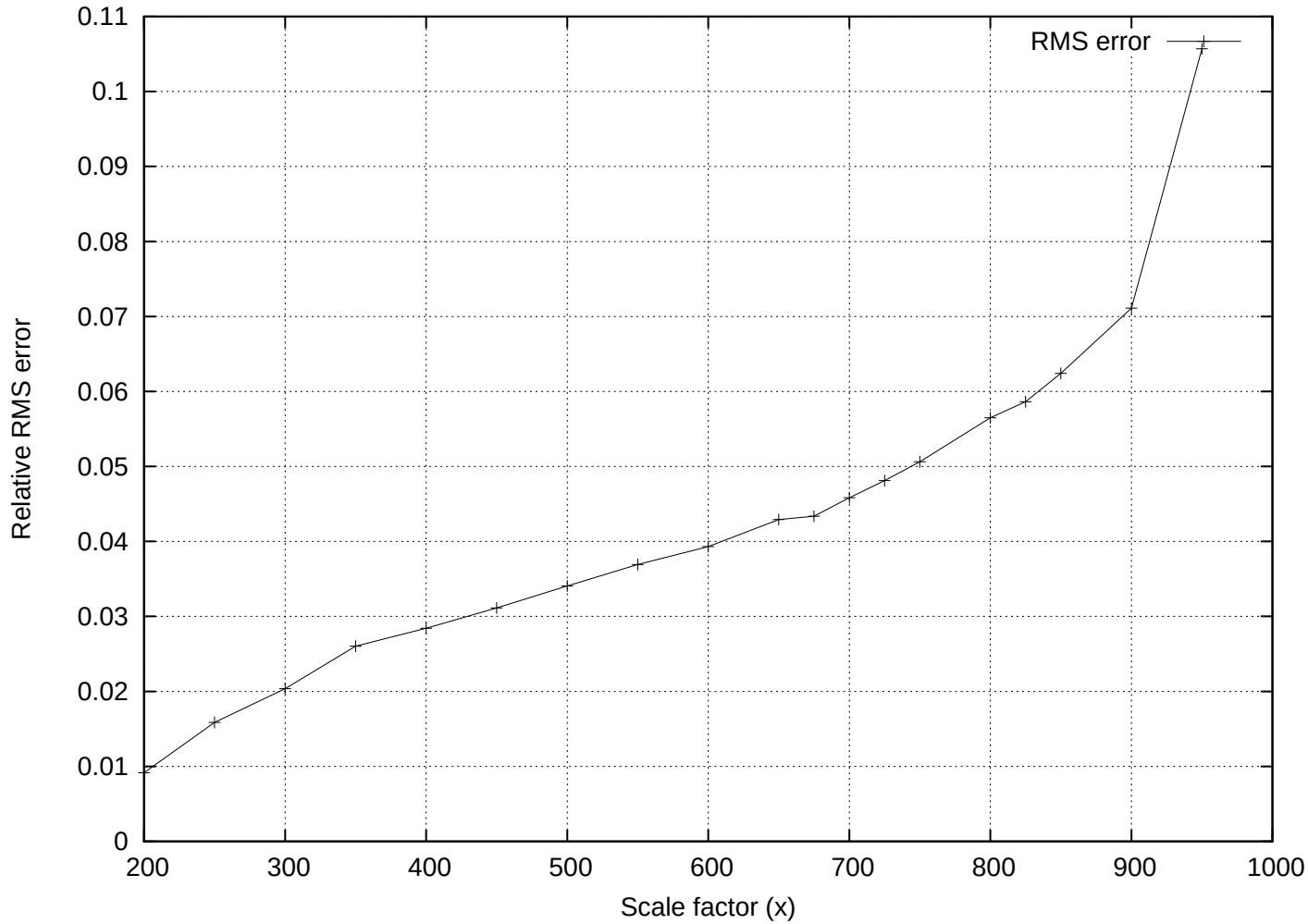
- Adaptive SRTS attempts to automatically change the real–time scaling factor to minimize the time that processors spend waiting but without provoking overrun situations.
- ASRTS is identical to SRTS, except that it periodically changes the real–time scaling factor. ASRTS cuts the simulation time into sampling periods of equal duration. During each such period, each LP measures the total time that it spends waiting. At the end of the period, one of the threads collects the accumulated waiting times of all LPs and determines their minimum. Using that information, it computes the ratio between the smallest accumulated waiting time and the length of the sampling period.
- If the minimum waiting ratio w is greater than a prescribed *desired* waiting ratio w_0 (we use values around 10%), the algorithm is allowed to increase the real–time scaling factor and simulate faster. Otherwise, the algorithm decreases r. In this way, ASRTS tries to keep the minimum waiting ratio around the *desired* waiting ratio w_0.

Applications and Examples

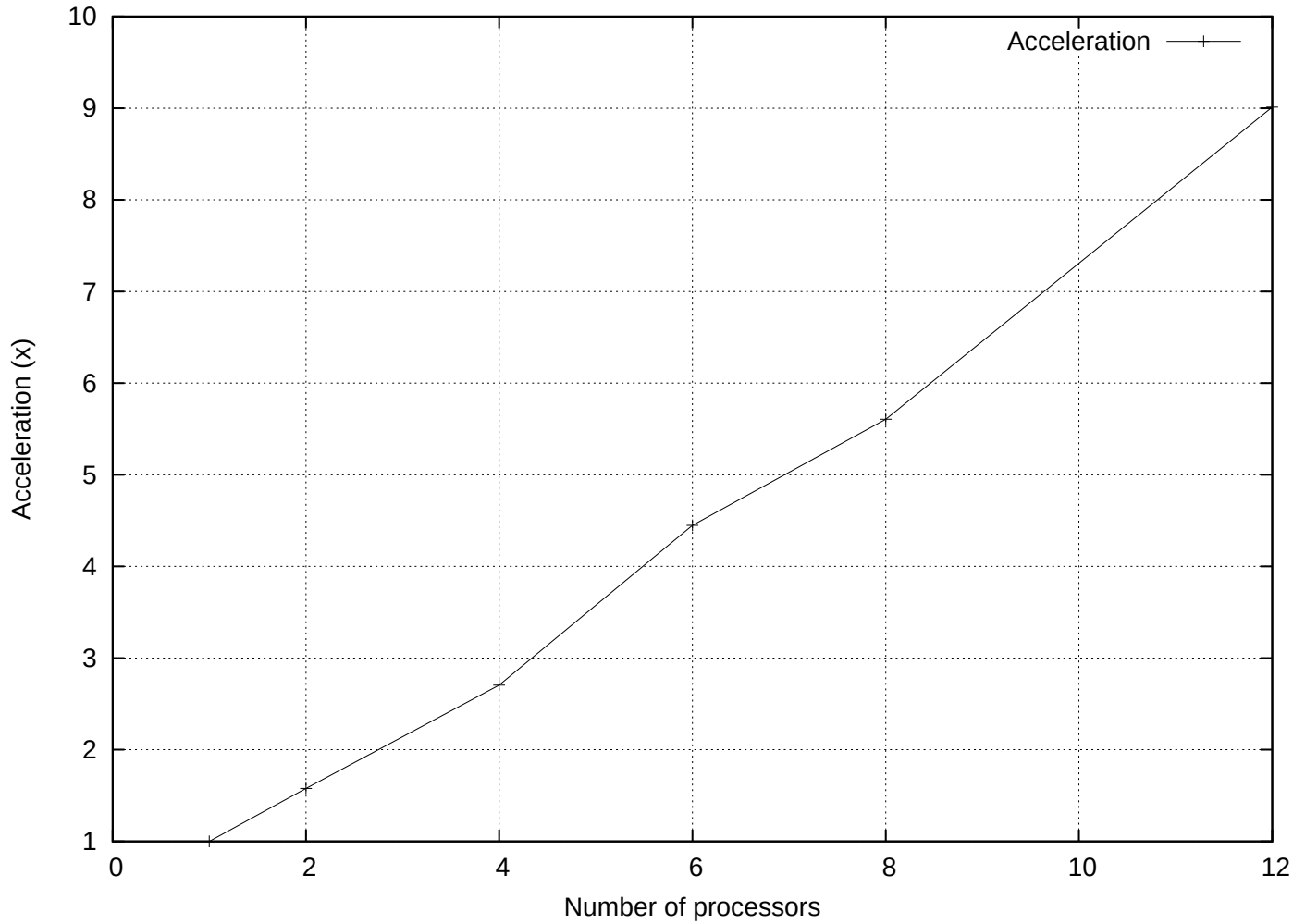
Results

We simulated two separate large–scale systems using QSS methods, first in a sequential way and then using SRTS and ASRTS techniques.
The first example is taken from Perfumo et al. [PKBW12], is a model to study the dynamics and control of the power consumption of a population of air conditioners (2400 ACs).
The second example is a chain of 500 logical inverters according to the model given in Savcenco and Mattheij [SM07],
All examples were run using an Intel i7 970@3.2GHz–6 cores machine with hyper–threading.

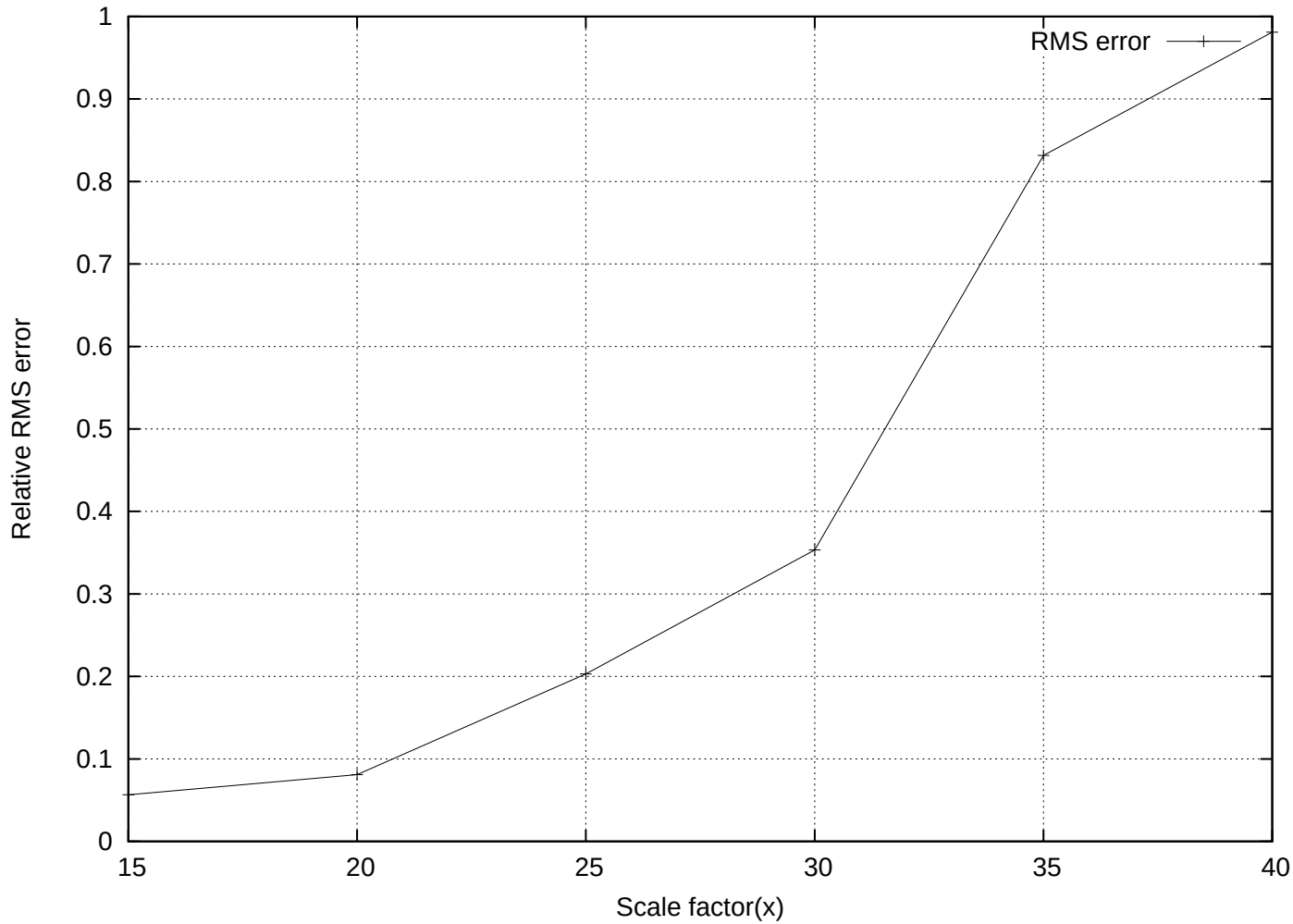
Results – AC Example – RMS Error vs Scale Factor



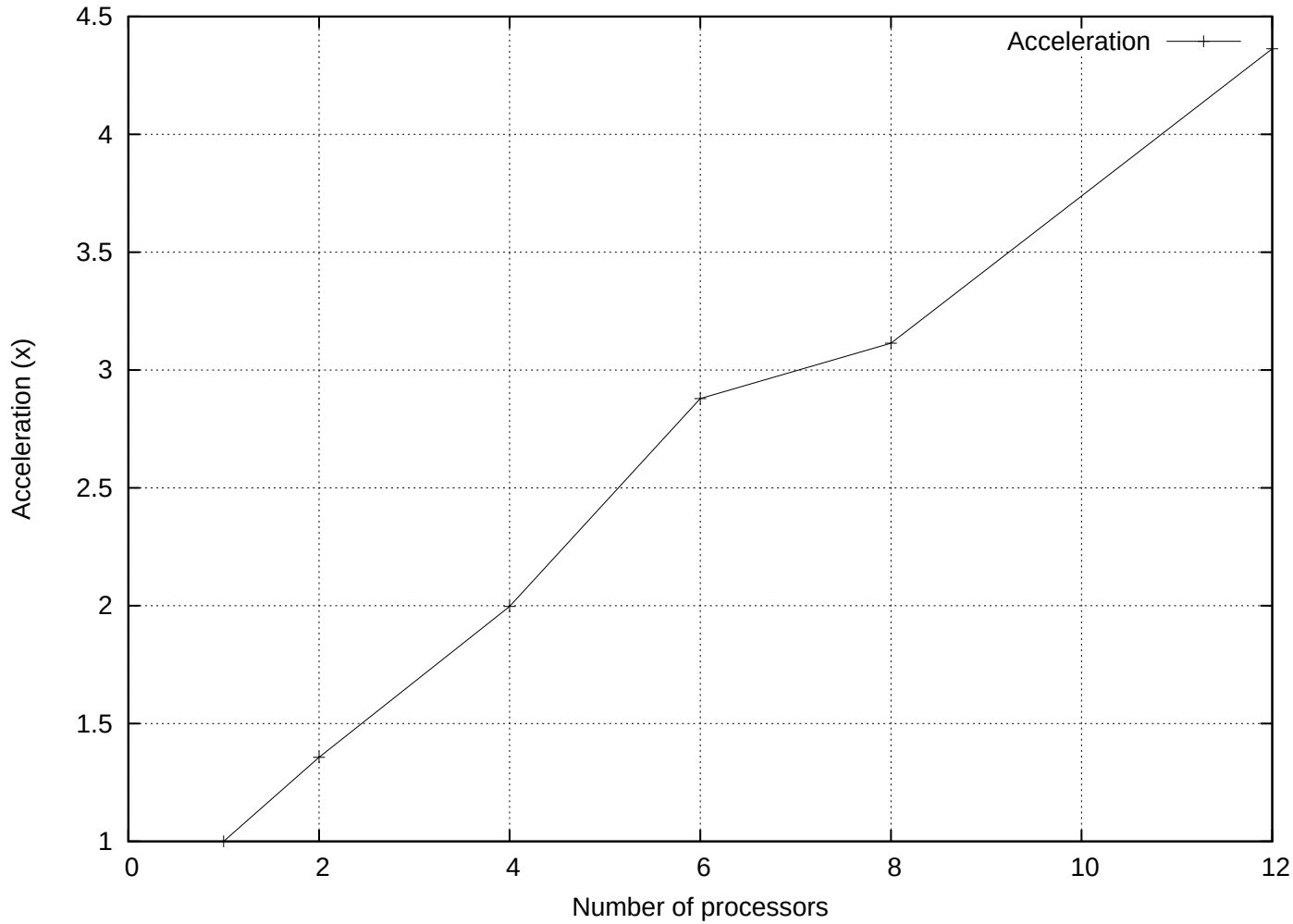
Results – AC Example – Speedup vs Number of Processors



Results – Inverter Chain – RMS Error vs Scale Factor



Results – Inverter Chain – Speedup vs Number of Processors



Conclusions

Conclusions

- We introduced SRTS and ASRTS, two new techniques for Parallel Discrete Event Simulation of continuous and hybrid systems.
- We implemented these new techniques in *PowerDEVS* using the real–time operating system RTAI and tested them with two application examples and studied their performance finding a major improvement on the simulation speed (ranging from 4 to 9 times faster).
- In this work, we manually split the model, it is interesting to investigate an automatic way to split the model and also to study techniques to dynamically re–balance the workload among the available processors. We are currently working on an extension to the DEVS formalism, called Vectorial DEVS, that will simplify this task.
- The current implementation was done in a multicore architecture. A interesting line of work is how to port the algorithm to a cluster implementation.

References

Federico Bergero, Ernesto Kofman, and François Cellier.

A Novel Parallelization Technique for DEVS Simulation of Continuous and Hybrid Systems.
Simulation, 2012.
Submitted.

François Cellier and Ernesto Kofman.

Continuous System Simulation.
Springer, New York, 2006.

Cristian Perfumo, Ernesto Kofman, Julio Braslavsky, and John K. Ward.

Load management: Model-based control of aggregate power for populations of thermostatically controlled loads.
Energy Conversion and Management, 55:36–48, 2012.

V. Savcenco and R.M.M. Mattheij.

A Multirate Time Stepping Strategy for Stiff Ordinary Differential Equations.
BIT Numerical Mathematics, 47:137–155, 2007.

Bernard Zeigler, Herbert Praehofer, and Tag Gon Kim.

Theory of Modeling and Simulation - Second Edition.
Academic Press, 2000.

Muchas Gracias!

Scaled Real–Time Parallelization for DEVS Simulation of Hybrid Systems

Federico Bergero and Ernesto Kofman and François Cellier

CIFASIS, CONICET - FCEIA, UNR - Argentina
ETH Zürich - Switzerland

July 24th, 2012