Multicore versus FPGA in the Acceleration of Discrete Molecular Dynamics*

Tony Dean Josh Model† Martin C. Herbordt
Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215

1 Introduction

For several years microprocessor performance stagnated making alternative architectures, such as FPGA, GPU, and Cell, attractive alternatives for compute intensive applications. Now with multicore becoming *the* mainstream, the comparisons have become more complex. While the *potential* performance of processor chips is once again increasing rapidly, changing the accelerator comparison dynamic, they are harder to use at high efficiency than their single core predecessors, and have inherent limitations for some applications.

In this abstract, we describe a case study of an application that has only been parallelized with great difficulty, has not yet been shown to be scalable, but which is amenable to massive speed-ups with FPGAs. The key in the latter design is taking advantage of the ultra fine-grained parallelism supported by FPGAs. Here we show that the same basic design can also be applied to multicore processors, and has the potential to be scalable at least while processing remains on-chip.

2 Discrete Molecular Dynamics

Molecular dynamics (MD) simulation is a fundamental tool for gaining the understanding of chemical and biological systems. Reductionist approaches to MD alone, however, are insufficient for exploring a vast array of problems of interest. With traditional time-step driven MD, the computational gap between the largest published simulations and cell-level processes remains at least 12 orders of magnitude.

In contrast, intuitive modeling is hypothesis-driven and based on tailoring simplified models to the physical systems of interest. Using intuitive models, simulation length and time scales can exceed those of time-step driven MD by eight or more orders of magnitude [1]. The most important aspect of these physical simplifications, with respect to their effects on the mode of computation, is discretization: atoms are modeled as hard spheres, covalent bonds as infinite barriers, and the van der Waals force as square wells. This enables simulations to be advanced by event, rather than time step: events occur when two particles reach a discontinuity in interparticle potential.

*This work was supported in part by the NIH through award #R01-RR023168-01. Web: http://www.bu.edu/caadlab. Email: {tdean95|jtmodel|herbordt}@bu.edu

†Now at the MIT Licoln Laboratory

Even so, as with most simulations, discrete event simulation (DES) of molecular dynamics (DMD) is compute bound. A major problem with DMD is that, even more so than with DES in general [2], causality concerns make DMD challenging to scale to a significant number of processors [4].

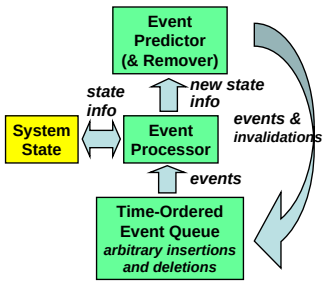

Figure 1: DES sketch.

Figure 1 shows the basic components of a generic DES system: simulation proceeds as a series of discrete element-wise interactions. As each event is processed, its consequences must be predicted. These include the generation of new events, and the cancellation of previously predicted events. Performance of well-tuned systems [6] for basic force models can reach 200K events/s [5].

Parallelization of DES has generally taken one of two approaches: (i) conservative, which guarantees causal order, or (ii) optimistic, which allows some speculative violation of causality and corrects violations with rollback. Neither approach has worked well for DMD. The conservative approach, which relies on there being a "safe" window, falters because in DMD there is none. Processed events invalidate predicted events anywhere in the event queue with equal probability, and potentially anywhere in the simulated space. The optimistic approach has frequent rollbacks, resulting in only $O(\sqrt{P})$ scaling, even in theory [4].

3 DMD on FPGA

In our FPGA implementation we take a different approach that incorporates both conservative and optimistic aspects and achieve a performance of 50M events/s. We process the entire simulation as one long computation pipeline (see Figure 2). While dozens of events are processed simultaneously, at most one event is *committed* per cycle. To achieve maximum throughput, the following must be done within a single cycle: (i) update the system state, (ii) process all causal event cancellations and (iii) new event insertions, and (iv) advance the event priority queue. Each of these operations requires a complex structure (as described in [5]).

As in a CPU, dependences combined with overlapped executions cause hazards. These are compounded by speculation, and also by another factor generally not an issue with microprocessors: part of the processing pipeline (the event queue) is necessarily in off-chip memory. These issues lead to four complications, which are now enumerated; sketches of their solutions can be found in [3, 5].

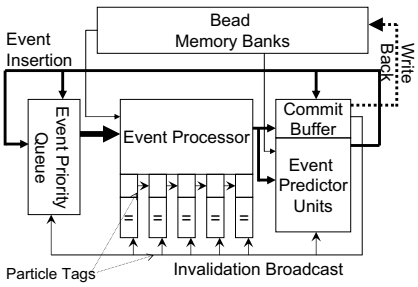

Figure 2: DMD on FPGA.

1. Causality Hazards. New events can be inserted anywhere in the pipeline, including processing stages.

2. Coherence Hazards. Events entering the predictor unit may be cancelled by an event that has not yet committed.

3. Combined Causality/Coherence Hazards. Events that need to be inserted into the predictor unit stages face both types of hazards.

4. Off-chip event queue performance. Enabling $O(1)$ execution of these functions depends heavily on associative operators. For simulations which enqueue more than a few thousand of events, off-chip memory must be used. The issue is to prevent software emulation from degrading performance to $\Omega(\log(N))$.

4 DMD on Multicore

While parallel DMD (PDMD) has been studied extensively, we know of no existing production implementation. We postulate that a key issue is that previous studies (where any speed-up was achieved) all used spatial decomposition plus some form of rollback. This approach is problematic because, as processing technology has advanced, the relative time to rollback has increased with respect to time to process events.

We propose an alternate method based on our FPGA implementation: task-based decomposition with overlapped event processing, but with serial commitment. While this serialization obviously limits potential speed-up, we see that the FPGA still gains 200x over a highly tuned code. With multicore, the baseline is that each event requires at least 5us, while commitment only involves updating a small number of structures: commitment should take only a small fraction of that time. With complex force models, the potential should be even greater.

There is some overhead involved to make this work. We need to check or ensure the following:

– Before speculative event processing, is another event being processed that could cause this event to be cancelled? Cancelling this event is not difficult, the problem is mispredicting new events.

– During speculative event processing, were we invalidated?

– During event prediction, was another event inserted ahead of us which would cause those predictions to become invalid?

– Atomic, in order, commitment.

The implementation adds certain structures and functions: a list of events being executed speculatively, neighborhood checks to ensure coherent predictions, and locks to ensure correct execution.

5 Results and Discussion

Table 1: Multicore performance. BL is baseline code with time/event in us. Other numbers are speed-ups. Model 1 is uniform ideal gas with 131,000 particles. Models 2 and 3 add overhead per event.

Threads	model 1	model 2	model 3
BL	10.0us	57.6us	523us
1	0.81x	1.00x	1.00x
2	0.79x	1.64x	1.92x
3	0.47x	2.20x	2.80x
4	0.23x	2.39x	3.65x

Table 1 shows performance of our just-completed initial (unoptimized) multicore implementation. Simulations were run on an Intel 2.5GHz Xeon E5420 Quad Core processor. For the basic ideal gas simulation, multithreading resulted in a substantial slowdown. Model 2 represents more closely simple models of biological interest: it shows good performance with two threads. Model 3 represents substantially more complex models and shows reasonable performance through four threads.

We have just begun analyzing this implementation. We anticipate being able to determine soon the partition of overhead between the intrinsic, such as synchronization, and application specific, such as pauses due to coherence hazards. Further work will certainly be necessary in optimizing the locks. Future work is to extend this to multiple sockets, and then to combine spatial and task-based decomposition.

References

[1] Dokholyan, N. Studies of folding and misfolding using simplified models. *Current Opinion in Structural Biology 16* (2006), 79–85.

[2] Fujimoto, R. Parallel discrete event simulation. *Communications of the ACM 33*, 10 (1990), 30–53.

[3] Herbordt, M., Kosie, F., and Model, J. An efficient $O(1)$ priority queue for large FPGA-based discrete event simulations of molecular dynamics. In *Proc. FCCM* (2008).

[4] Miller, S., and Luding, S. Event-driven molecular dynamics in parallel. *J. Comp. Physics 193*, 1 (2004), 306–316.

[5] Model, J., and Herbordt, M. Discrete event simulation of molecular dynamics with configurable logic. In *Proc. FPL* (2007), pp. 151–158.

[6] Rapaport, D. *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2004.