Object Level Locality in Real-Time Physics Applications

Paul Salzman

Advisor: Professor Glenn Reinman
CS 194 Winter 2007 – Spring 2007

Abstract

As the interactive entertainment industry grows, so does the quest for software realism. Real-time physics simulation has become closely tied with interactive entertainment as it helps to convey a true to life dynamic environment to the user. These simulations require many complex calculations and can utilize computing resources for long periods of time in order to complete. Unfortunately, there is a very strict upper limit on the length of time in which all calculations must complete to allow frames to be rendered at an acceptable rate. In this paper we explore the properties of objects in a physics simulator to see if their correlation to physical objects in motion lends to object level locality and how that can be leveraged for increased performance.

Introduction

Accurate real-time physics simulations are very calculation intensive. In order for such simulations to be practical for use in commercial applications and in interactive entertainment, the physics simulation iterations must complete in well under 0.033 seconds. This restriction must be enforced to allow other portions of the application to complete as well and still have processing time for frames to be rendered and displayed 30 times a second. Some of the more demanding aspects of physics simulation include collision detection, cloth deformation, and fluids.

If the values pertaining to the software objects representing physical object properties can be accurately predicted, instruction level parallelism (ILP) can be increased. ILP allows multiple independent instructions to be issued simultaneously which increases overall performance. Key deterrents to ILP are data dependencies, which prevent a long string of instructions in the instruction queue from issuing until the dependent data is obtained. Longer latency instructions like loads cause longer delays in instruction issuing. By predicting the value of a load the processor can continue on its execution path while the actually data is loaded. If the load is correctly predicted, the processors work is useful and it can continue without the penalty of the load latency. If the load is incorrectly predicted, all issued dependent instructions must be squashed and the processor can start the string of instructions as if they were new. If the increased ILP is harnessed, the gain in performance can be used to increase the complexity and accuracy of physics simulations. This would in turn help make the use of real-time physics more feasible for interactive entertainment applications.

Previous research shows that value prediction is a useful method to increase process performance and that there are various implementations for these predictors [2, 4]. It has also been shown that value prediction performs even better when appropriate instructions can be filtered out for confident predictions [1]. Finding methods for improving performance of real-time physics simulations

has proved an interesting topic for recent research. Correlation between high level application data locality and critical portions of physical simulation have been found, as well as means of using this locality for improved control prediction [5]. In this paper we plan to explore if object level locality exists in real-time physics simulations and if so, how it can be utilized with value prediction to increase performance.

Methodology

The process of examining object level locality will consist of tracing object behavior through selected portions of the physics simulator. This object trace data must then be analyzed for any appearance of locality. These results will lead to the synthesis of various value predictors that can be tested to take advantage of any locality found at the object level.

To observe any object level locality, I will examine object performance in the Open Dynamics Engine (ODE) [3]. ODE is an open source real-time physics simulator that is commonly used in interactive entertainment applications. To fully utilize ODE in an examination of object locality, I will use a performance heavy benchmark created by Thomas Yeh [5] that exercises many facets of physics simulations through multiple concurrent instances of physical object interaction.

The GNU gprof utility will be used to focus in on the functions occupying the large portion of physics simulation time. Upon locating these functions, trace code will be injected around object loading code in these functions in order to collect data. This code will associate an object with its various states in a chronological order as well as the overall result of the function and dump it into a text file for analysis.

Once the data collection is complete, the traces must be analyzed. The trace data will be indexed based on two different conventions, the instruction's PC value and the exclusive OR of the instruction's PC value with the corresponding object's identifying value. For the purposes of this paper an object's identifying value is defined as its location in memory as this value persists through the benchmark. The load values associated with each index will be chronologically examined for adjacent values, stride values, and trivial values of 1, 0, or -1. This data will be used to determine whether not any object level locality exists.

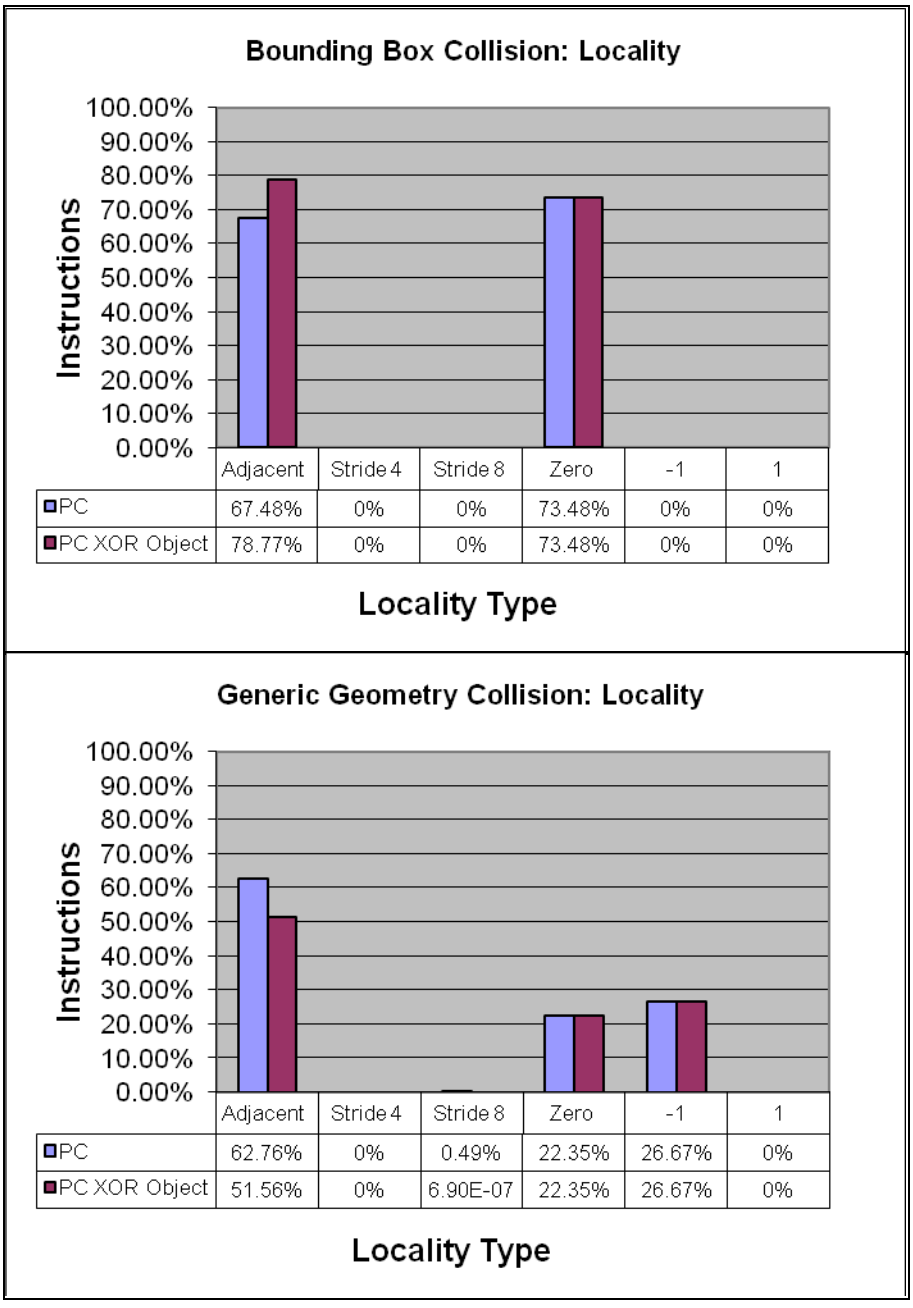
Using the object level locality data, value predictors are constructed. These predictors will also be tested by indexing by only an instruction's PC value and by the exclusive OR of the instruction's PC value with the corresponding object's identifying value. There will be no limit placed on the predictors' size as to avoid any constructive or destructive aliasing.

Results

The function that uses the most processing time in our benchmark is a bounding box collision function, dBoxBox, which occupies 43% of the overall simulation time. The second longest function is a generic geometry collision detection function, collisionAABBs, which occupies 18% of the simulations processing time. As these functions consist of over 60% of the overall simulation time, they will be the

examined functions for the use of this paper. It is also important to note that the instructions we will examine spread across these functions scale up to approximately 40% of the load instructions in the benchmark.

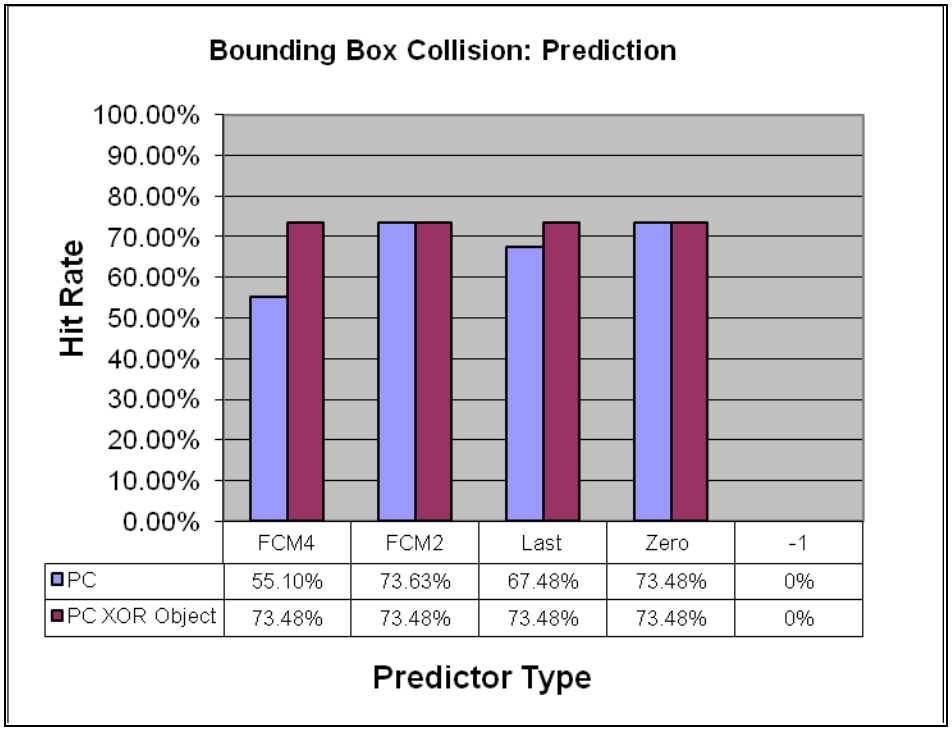
Collision detection is one of the more common demanding components in real-time physics simulation. Each object in a scene must be compared to all other objects to determine if a collision is occurring. Colliding objects are then grouped together in order to determine which parts of each object's physical representation are colliding and how that affects each objects properties. If object level locality exists, it may be harnessed to help speed up or even predict the result of a collision calculation and may greatly impact the performance of one of the largest portions of the simulator.



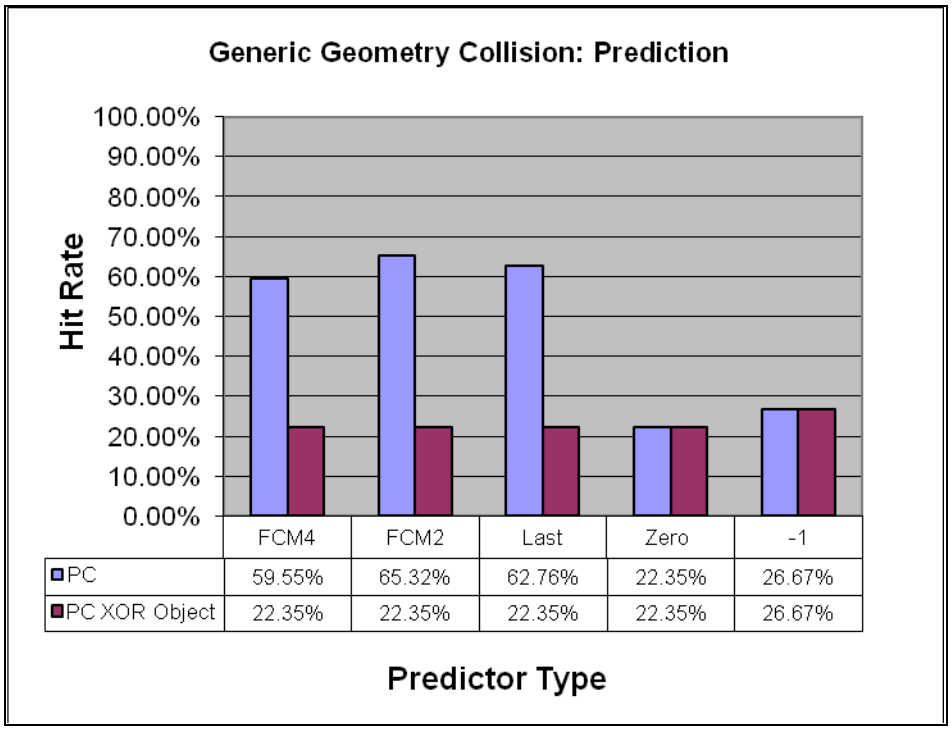
Six different forms of examination are applied to the collected values. Adjacency locality is the chronological appearance of a repeated value. Stride 4 and Stride 8 locality is the chronological appearance of values separated by the 4 or 8, respectively. The 0, 1, and -1 examinations are trivial checks to see if the values associated with an object can be easily guessed as 0, 1, or -1.

The data displays a strikingly large appearance of adjacent value locality at the object level. The bounding box function's adjacency locality is 67.5% and 78.8% for PC indexing and PC XOR object id indexing, respectively. Generic geometry collision adjacency locality is lower at 62.8% for pc indexing and 51.6% for pc XOR object id indexing. Stride locality does not appear to be present in object values, but the intuitive properties of a physical object do not coincide with the properties of stride locality. The trivial value appearance of objects has a striking presence in the bounding box function. 73.5% of the values seen in this function are zero, and could lend to a very powerful and simple to implement zero value predictor. In the generic geometry function a substantial but smaller 22.4% of values are zero, and 26.7% of values are -1.

Predictors are formulated from the large appearance of adjacency locality and trivial values in the collected data. To attempt to leverage adjacency locality in a predictor, a last value predictor as well as two implementations of a Finite Context Method (FCM) predictor [4] are tested. FCM predictors predict values based on an implementation dependent number of previous values, which is also called its order. Counters are used to keep track of the occurrence of stored values and this in turn is used to decide what values should be used as predictions. Our tests will use an order 2 FCM predictor and an order 4 FCM predictor. To attempt to leverage the appearance of trivial values, very simple trivial value predictors are constructed.



	FCM4	FCM2	Last	Zero	-1
PC	55.10%	73.63%	67.48%	73.48%	0%
PC XOR Object	73.48%	73.48%	73.48%	73.48%	0%



A noticeable and perhaps strange occurrence is that all XOR indexed predictors have similar statistics as zero value predictor. This is attributed to the fact that we have chosen an unlimited predictor size to avoid any effects of aliasing and that all predictors initially default to a prediction of zero when not enough values have been seen. The large number of objects in the benchmark combined with this form of indexing cause the spread of predictors to collect too little data and basically act as zero value predictors. It is also noteworthy that the trivial value predictors have a hit rate equal to the corresponding appearance of trivial values in the locality test.

The PC indexed FCM order 2 predictor showed the greatest hit rate across both functions. The PC indexed FCM order 4 did not perform as well, perhaps from its larger histories slower pace at replacing old predicting values. Last value prediction performed very well through both functions, and when considered with the simplicity of its implementation might make the best trade off between accuracy and physical implementation.

Generic geometries vary and have more complex input than bounding box collision detection. Lower appearance of locality is understandable as the inputs vary much more outside of simple bounding boxes. This also provides a reason why the values for generic collisions are noticeably less predictable.

Summary

This paper shows that there is a large appearance of adjacency locality and trivial values in real-time physics simulations. Object level locality can be harnessed for approximately 40% of loads in an

application to be predicted with the over 60% accuracy rate seen in our tests to greatly increase ILP. This in turn can increase performance and make more realistic and complex real-time physics simulation feasible.

For this locality to be harnessed, further work must be done to bring a software level object's identifying value to the physical architecture level in order to index into the value predictors. Highly specialized real-time physics simulation architectures currently exist, and it is feasible that further iterations of these architectures can utilize this information to implement object level

References

[1] Brad Calder, Glenn Reinman, and Dean Tullsen. Selective Value Prediction. In *26th International Symposium on Computer Architecture*, May 1999

[2] Mikko Lipasti, Christopher Wilkerson, and John Shen. Value locality and load value prediction. In Seventh International Conference on Architectural Support for Programming Languages and Operating Systems, 1996.

[3] Open Dynamics Engine. http://ode.org/.

[4] Yiannak Sazeides and James E. Smith. The predictability of data values. In 30th International Symposium on Microarchitecture, pages 248-258, December1997.

[5] Thomas Y. Yeh, Petros Faloutsos, and Glenn Reinman. Accelerating Real-Time Physics Simulation by Leveraging High-Level Information. In UCLA CSD-TR 060023, 2006.