An execution model for the next end-user-oriented massively parallel system

Yoshiki Ohshima

This material is based upon work supported in part by the National Science Foundation under Grant No. 0639876. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

VPRI Memo M-2008-002

Viewpoints Research Institute, 1209 Grand Central Avenue, Glendale, CA 91201 t: (818) 332-3001 f: (818) 244-9761

An execution model for the next end-user-oriented massively parallel system

Yoshiki Ohshima
yoshiki@vpri.org
December 2nd, 2008

1 Introduction

Drawing upon the experiences from other end-user programming systems and the authors own (the "Kedama" system), this memo discusses the design trade offs of a possible new system with massively parallel execution models.

One of the big goals of such an end-user system is to provide an easy and gradual transition for the user between writing the program for an exemplar and the program for the group of objects. Namely, the user can start exploring the problem by creating one concrete object and express its behavior and then scale the number of objects (let us call them the siblings of the exemplar). Of course, the transition from massive number to one should be possible when the user wants to work on in that way.

2 Design Trade offs

2.1 Granularity

Because of the scaling requirement, the unit in the parallel execution needs to be, or at least needs to look like, a real object. And the same script should be able to applied to it and its siblings. This leads to the idea that the granularity of execution is formulated around the individual object, and a "command" operates on it or the group of them. The invocation of a command for a group of objects should give the user the "illusion" that the objects are acting at the same time.

2.2 Homogeneous vs. Heterogeneous

The Kedama system only allows the uniform objects in a group (a breed) of objects. This simplified the implementation but of course the user would want to have different grouping of objects other than one single kind of breed.

With heterogeneous collections, there will be errors at runtime, even though the compiler can detect some kind of errors at compile time. For the target audience, this is a feature and not a bug.

2.3 Synchronization

A user-defined script may be applied to a group of objects. Upon such invocation, the objects in the group should give the illusion of all of these objects running at the same time. If the script contains a statement with side-effects, the Kedama compiler used to automatically serialize the execution. However, For the sake of the pseudo-time based execution that will be discussed below and the expressiveness that some explicit parallelism

specification (such as “do-together” and “do-in-order”), the side-effects from one in the group should not be visible to others by default.

To give the illusion of the same time actions, a simple command such as “forward by” and complex one that consists of simple commands, the parallel execution for a group of objects should take about the same time; in a sense they should be SIMD-like, and should handle the polymorphic or heterogeneous case.

2.4 Pseudo-time based execution

For doing a better simulation, the concept of time is important but the speed of computer and the time to execute one step varies so the system cannot rely on the wall clock time (which is what Kedama did).

The tradition is to use the pseudo-time, and each action is associated with the pseudo-time (as in discrete event simulation). Every command even simple one such as “forward by” should consume some (pseudo-)time to execute. The side effects from such a command should be visible to the next command in the same script. In other words, each command should be seen as a kind of SIMD-instruction.

2.5 Control Structure

Loop structure need to be accommodated. When a script with a loop is applied to a group of objects, the script is executed on the objects in parallel.

The number of iteration of the loop may be different for different objects. The model here is still to stick to the SIMD-like model; the objects that finish early wait for the one that takes longest.

Other kinds of control structures should be added and they should behave in the same principle.

3 Conclusion

The overall model would be somewhat similar to the Kedama, with more elaborated control structure, homogeneous groups, and pseudo-time based execution. There should be different kinds of surface languages, one perhaps in tiles, and perhaps one in a textually-coded language.