As Strong As Possible Mobility

Tim Walsh, Paddy Nixon, Simon Dobson

Distributed Systems Group, Department of Computer Science,
Trinity College Dublin
Email: {Tim.Walsh, Paddy.Nixon, Simon.Dobson}@cs.tcd.ie

An executing thread, in an object oriented programming language, is spawned, directly or indirectly, by a main process. This in turn gets its instructions from a primary class. In Java there is no close coupling of a thread and the objects from which they were created. The use of a container abstraction allows us to group threads and their respective objects into a single structure. A container that holds threads whose variables are all housed within the container is a perfect candidate for strong migration. To achieve this we propose a combination of three techniques to allow the containers to migrate in a manner that approaches strong mobility yet does not resort to retaining bindings to resources across distant and unreliable networks.

Stack Collapse
Java serialisation does not allow the thread stack to be captured. Therefore, it may prove prudent to try to collapse the stack naturally. If no methods are being executed when migration is initiated then the stack will be flat (no local variables to be saved), thus allowing the object to be moved and restarted on a new node. Essentially the object is being placed in a hibernation state. The main method will be the sole remaining frame. The remains of the object’s state will be encapsulated in the instance variables. Java’s serialisation can capture these variables automatically.

However, just like threads, not all variables are serialisable. Some objects are not capable of being migrated. Instead, providing alternatives to these resources on the destination host can provide a mechanism that once again aim to reduce the dependency on retaining references across the Internet.

Data Space Management
Prior to migration an object may hold references to many other different services and resources. Objects have generic requirements that are required wherever they may execute. Each host system will provide generic resources that can be automatically rebound to. Examples include display screens for reporting progress, printers, proxy servers, classloaders, replicated databases and object request brokers. A migrating object may use these generic resources instead of attempting to use the same resources on its previous host. This allows an object to migrate and continue executing using identical services. The effect of this is to reduce the dependency the object has on potentially unreliable remote resources. The objective is to only retain bindings to remote resources when they are either essential or desirable.

Strong Mobility in Java
In Java, threads and objects are distinctly separate entities. A Java thread may be created from object methods but it then becomes an abstract entity onto itself. The state of the object is retained separately on the heap. Executing threads hold all local variables generated in working memory.

The standard Java API does not provide mechanisms to achieve strong migration. There is no way to access these thread objects directly. We investigate whether this is feasible using JavaSoft’s Just In Time (JIT) Compiler API. This API gives hooks into the virtual machine that can be used, amongst other things, to extract the execution environment for particular threads.

A single Java stack frame is comprised of 3 parts. The operand stack, the execution environment and the local variable array. The operand stack is a 32-bit LILO data structure where values are pushed onto and popped off during expression evaluations. The Frame Data Structure (or execution environment) stores the Virtual Machine’s registers and program counters.

Since the values on the stack are essentially just a collection of numbers it is necessary to build a parallel type stack which allows us to make sense of the values and their meaning. As the structure of the stack is defined at compile time it is possible to build a type stack and identify the value stack’s types.

The JIT hooks allow the stack frame of specific threads to be examined and saved. They also give access to the Virtual Machine functions for memory allocation and deallocation. This forms the main basis for this part of the migration process.

A thread whose stack scan reveals no references outside is container can be suspended. The entire frame is then saved, transmitted and re-seeded in a new Virtual machine where it can continue execution from its original checkpoint.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ICSE 2000, Limerick, Ireland
© ACM 23000 1-58113-206-9/00/06 ...$5.00