発表概要

仮想化環境における文字列オブジェクトの重複回避

堀江 倫大[1,a)]　緒方 一則[1]　河内谷 清久仁[1]　小野寺 民也[1]

2013年11月12日発表

1つの物理マシン上で，OS や Java VM といったゲスト環境が複数動作する構成においては，プログラムのメモリ使用量を削減し物理メモリを効率的に使用することが，サーバの利用効率向上のために重要である．文字列データは，多くのソフトウェアで最もメモリを消費するものの1つであり，プログラム実行の際に重複する文字列データが大量に生成されることがよくある．さらに，ほぼ同一のソフトウェア群が動作する環境では，各ゲスト VM 内に同じ文字列データが生成されるため，ゲスト VM 間で重複する文字列データがメモリを無駄に消費してしまう．そこで本論文では，物理マシンのメモリ使用効率を上げるために，Java ヒープ・メモリ中の文字列オブジェクトの重複を削減し，さらに，各ゲスト VM 間をまたぐ文字列オブジェクトの重複も削減する手法 String Deduplication を提案する．ヒープ・メモリ内における重複文字列を削減するために，文字列オブジェクト生成に至る calling context を利用して，生成される文字列のほとんどが重複文字列であるようなパスを抽出し，それを用いて効率よく重複文字列の生成を回避する．また，ゲスト VM 間の重複を削減するために，事前に文字列データをひとつの DLL にまとめて各ゲスト VM に配布し，各 Java VM がこの DLL をロードし文字列を共有することで重複する文字列を生成しないようにする．本手法を IBM J9 Java VM に実装し，IBM WebSphere Application Server と Apache DayTrader ベンチマークを使って効果を測定したところ，性能を劣化させることなくメモリ消費量を削減することができた．

String Deduplication in Cloud Servers

Michihiro Horie[1,a)]　Kazunori Ogata[1]　Kiyokuni Kawachiya[1]　Tamiya Onodera[1]

Presented: November 12, 2013

To increase the memory efficiency in physical servers is a significant concern in the virtual environment. When similar web application service runs in each guest VM, many string data with the same values are created in every guest VMs. These duplicated string data are redundant from the viewpoint of memory efficiency in the host OS. To solve this problem, we devised String Deduplication to reduce the duplication of string values found in one JVM and across JVMs in guest VMs. For deduplicating the strings inside JVM, our StringProfiler tool collects the calling contexts for the allocations of string objects during a trial program run. It determines which calling contexts created duplicated string values, and it suggests which calling contexts should be optimized. StringProfiler also generates the optimized code samples. By applying these optimized codes to the base code by developers, the actual deduplication can be enabled. For deduplicating the strings that are found across JVMs in the guest VMs, StringProfiler also generates a DLL including common strings that are created among the JVMs in guest VMs. Since the string DLL includes String and char [] objects as the same memory format in the Java heap, these objects can be used immediately after JVM maps the DLL into the memory. To avoid the performance degradation for looking up a string from the DLL, our StringProfiler generates the code that is specialized for an efficient lookup mechanism. This paper also evaluates our approach by using actual Java applications. Our prototype implementation achieved 7% to 12% reduction in the total size of the objects allocated over the lifetime of the Apache DayTrader and the DaCapo benchmark suite. In addition, we observed a performance improvement when using DayTrader benchmark running on three guest VMs in a KVM host machine.

[1] 日本 IBM 東京基礎研究所
IBM Japan, IBM Research-Tokyo, Koto, Tokyo 135–8511, Japan
[a)] horie@jp.ibm.com