Garbage collection for managed runtime on multicore processors

Mots clés :
- Directeur de thèse : Marc Shapiro
- Co-encadrant(s) :
- Unité de recherche : Laboratoire d'informatique de Paris 6
- École doctorale : École Doctorale Informatique, Télécommunications, Électronique de Paris
- Domaine scientifique principal : Divers

Résumé du projet de recherche (Langue 1)
Managed Runtime Environments (MREs), such as a JVM or a CLI, are increasingly successful for application development and execution, mainly because they provide good properties of manageability, isolation, and safety in general. A key component is the automated memory manager or garbage collector (GC). Garbage collection automatically frees unused memory, relieving application programmers from complex protocols and guaranteeing the absence of memory leaks or access violations. However, this comes at a price. Experimentally, GC may add up to 20% or 30% to mutator (i.e., application) execution time. This performance penalty is particularly an issue with the recent emergence of multicore architectures. Indeed, current GC algorithms used in production MREs environments execute sequentially and block all mutator threads. Amdahl's law predicts that, due to this sequential bottleneck, even hundreds or thousands of cores will not speed up a mutator any better than a factor of five. The performance of mutators cannot scale with the number of cores. Furthermore, the performance impact of GC on multicore architectures is exacerbated by the memory architecture. Indeed, current GC algorithms destroy cache locality, leading to numerous cache misses, and further decreasing performance.

Résumé du projet de recherche (Langue 2)
The goal of this PhD is to study GC algorithms that scale well with the number of cores. To achieve this, algorithms should not halt execution of the mutator during collection and should respect cache locality. One possible approach is segregating GC to a dedicated core, leaving mutator cores undisturbed; another is per-core collection of the local cache. Concurrency between mutator threads and the GC thread is a challenge, since mutator threads continue to modify the memory during garbage collection, a classical problem of read/write access concurrency. Several algorithms were have been proposed in this area (from Dijkstra or Steele in the 1970's, to Petrak recently). These state-of-the-art algorithms form the baseline of our proposal. We will adapt them to take into account cache locality and to minimise their sequential phases. The result of this work is expected to be substantial increase of the performances of GCs for multicore architectures, and more generally a better understanding of the concurrency problems raised by these architectures.

Informations complémentaires (Langue 1)
Stage à l'étranger obligatoire pendant la thèse