Adaptive Heap Management on Many-Core Platforms

Ioannis Koutras, Alexandros Bartzas and Dimitrios Soudris

Institute of Communication and Computer Systems (ICCS) – National Technical University of Athens, Greece
{joko, alexis, dsoudris}@microlab.ntua.gr

Abstract

The current design trend in System-on-Chips (SoCs) utilizes heavily multiple processors and is therefore shifted towards the Multi-Processor SoC (MPSoC) design paradigm. As technology allows the integration of an aggressively increasing number of transistors, the concept of many-core computing steadily emerges, suggesting tens of processor cores integrated in one chip as a computing fabric. Without any question, memory management on such architectures could contribute notably in improving performance and mitigating the scalability bottleneck, as the processors struggle constantly to access data from shared memory locations. This bottleneck could have an even greater impact on many-core architectures designed for the embedded system context due to the small size of memories used in such hardware.

Applications developed for such systems are slowly adapting to this model while trying to exploit every possible resource by using data- and task-level parallelism. This leads to applications with highly dynamic behavior and parallel execution of their tasks. This increased dynamism leads to unexpected memory footprint and fragmentation variations, which are difficult to be identified adequately at the design time. Developing dynamic multi-threaded applications using worst-case estimates for managing memory in a static manner would impose severe overheads in memory footprint and power consumption. In order to avoid such type of costly over-estimations, developers are motivated to efficiently utilize dynamic memory.

Dynamic memory managers (DMM) are responsible for organizing the dynamically allocated data in memory and also servicing the application's memory requests (allocation/de-allocation) that happen during the application's execution [3]. In typical C programming, dynamic memory allocation is performed through malloc()/ realloc()/free() function calls.

Memory management is one of the key challenges in the design of computing systems where the memory is often the main bottleneck [2]. The problem scales disproportionally as new systems are based on many-core architectures where the cores have to struggle accessing a limited amount of resources. Moreover, the excessive variations of modern systems, both in hardware and in software, makes necessary the usage of dynamic memory management (DMM) mechanisms. Extensive research has been conducted for general-purpose DMM, which targets single processor or multi-processor domain. However, the inherent generality of existing DMM eliminates the potential for customization optimizations.

Here we present **dmmlib**, a highly portable DMM library written in C (Figure 1). It allows developers to generate custom heap managers by choosing the desired features and policies. Previous analysis indicated that there is a need for extending the current status in multi-threaded DMM towards adaptive implementations that can be fine-tuned during the runtime of the system [1][4]. Software-controlled runtime tuning will be the main future approach to system designs due to the excessive variations in both hardware and software. In **dmmlib** we have completely decoupled the DMM's data structures from their manipulation services, enabling the runtime switching of multiple DMM management policies and mechanisms that are applied to the same data structures in a mutual exclusive manner. In addition, we properly extended the DMM structures to incorporate runtime software monitoring mechanisms regarding each of the allocated heaps' status. These runtime monitors form the actual data that guide the runtime decision-making process regarding the DMM's reconfiguration.

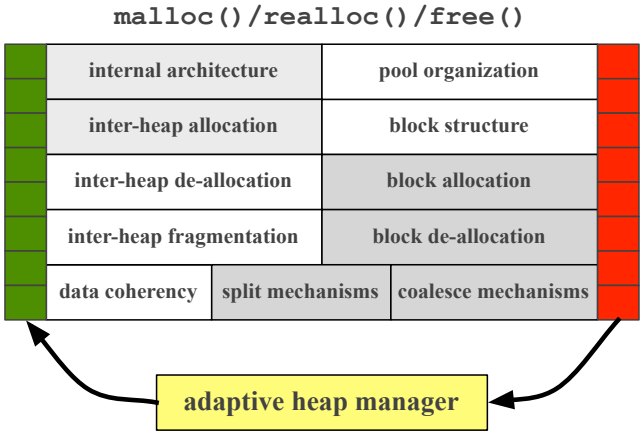

Figure 1: Adaptive heap manager (grey boxes support adaptivity).

References

[1] S. Xydis et al., "Runtime Tuning of Dynamic Memory Management For Mitigating Footprint-Fragmentation Variations," in Proc. of PARMA. VDE Verlag, 2011.

[2] E. Berger et al., "Hoard: A scalable memory allocator for multithreaded applications," in ACM SIGPLAN Notices, 35(11): 117–128, 2000.

[3] P.R. Wilson et al., "Dynamic storage allocation: A survey and critical review," in IWMM, 1995.

[4] I. Koutras et al., "Efficient Memory Allocations on a Many-Core Accelerator," to appear in Proc. of PARMA, 2012.

Partially supported by the E.C. funded 2PARMA FP7-ICT-2009-4-248716 Project – http://www.2parma.eu