Runtime resource management for lifetime extension in multi-core systems

Cristiana Bolchini
Dip. Elettronica, Informazione e Bioingegneria - Politecnico di Milano, Italy
Email: cristiana.bolchini@polimi.it

The availability of numerous, possibly heterogeneous, processing resources in multi-core systems allows one to exploit them to optimize performance and/or power/energy consumption. In particular, strategies have been defined to map and schedule tasks on the system resources, with the aim of optimizing the adopted figure of merit, at *design time*, if the working context is known in advance and relatively stable, at *run time* when facing changing/unpredictable working conditions [1]. However, it is important to be aware that such strategies may have an impact on the overall lifetime of the system because of aging and wear-out mechanisms. Therefore such management strategies, generally adopted for handling performance and power consumption aspects, should be enhanced in order to consider such issues. Furthermore, specific Dynamic Reliability Management (DRM) policies have been devised to deal with lifetime issues in multi-core systems, acting mainly on the workload distribution (and eventually on architectural knobs, such as voltage/frequency scaling) to mitigate the stress caused by the running applications.

Here we will focus on DRM strategies, whose goal is pursuing the improvement of lifetime reliability by means of load distribution policies that identify the resource where to map a new application entering the system, or where to periodically migrate tasks to balance stress. More precisely, a selection of state-of-the-art solutions will be presented and analysed, with respect to the achieved expected lifetime, evaluated when considering the first failure as well as the sequence of failures leading to the system being unable to fulfill the user's performance of service requirements.

In several approaches, as the work in [2], [3], stress is maintained homogeneous among the processing cores, in order to postpone the moment one core among the working ones will fail. The analysis shows how this pursuit effectively postpones the time the first core is expected to fail. In fact, as discussed in [3] to maximize the Mean Time To Failure (MTTF) of the first core, perfect spatial and temporal load balancing must be achieved, so that the maximum core wear state among all the cores must be minimized. Approaches in this category estimate the wear-out level of each core, for instance based on the previous executed workload, and decide to map tasks on the least wear-out core. Indeed load/wear-out balancing solutions succeed in prolonging the moment the first core fails. Such load/wear-out balancing is reasonably easy to perform when considering single-thread application, incurring a reduced number of constraints in the decision process. On the other hand, these DRM policies are more complex and difficult to be integrated in runtime resource managers when considering multi-threaded applications to be mapped, since more tasks need to be allocated on several cores thus increasing the complexity of the decision space [4]. Moreover, these policies take into account the system to be failed when the first core fails. However, when adopting the more realistic (although more complex to be computed) system MTTF where the system survives until the k-th core failure, wear-out balancing is not necessarily the most effective choice, as shown in [5].

In this perspective, another class of approaches focuses on maintaining spare cores, if the performance requirements allow for it, to be used only when a core fails. Different strategies can be adopted when deciding which cores (spatially and temporally) should be considered spares, having different impact on the final system lifetime, since wear-out does not only depend on the actual workload executed by a core, but also by the activity (and consequent temperature) of the adjacent cores [2], [6]. Moreover, some strategies exploit the use of spare to periodically migrate workload from one core to a spare one, in order to mitigate wear-out effects and possibly increase the overall system MTTF.

A comparison of the selected approaches will be presented and analyzed from an empirical point of view, by adopting a state-of-the-art reliability modeling and estimation framework.

References

[1] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on Multi/Many-core Systems: Survey of Current and Emerging Trends," in *Proc. Design Automation Conf.*, 2013, pp. 1–10.

[2] L. Huang and Q. Xu, "Characterizing the lifetime reliability of manycore processors with core-level redundancy," in *Proc. Int. Conf. on Computer-Aided Design*, 2010, pp. 680–685.

[3] T. Chantem, Y. Xiang, X. S. Hu, and R. P. Dick, "Enhancing Multicore Reliability Through Wear Compensation in Online Assignment and Scheduling," in *Proc. Conf. on Design, Automation and Test in Europe*, 2013, pp. 1373–1378.

[4] M.-H. Haghbayan, A. Miele, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Lifetime-aware load distribution policies in multi-core systems: An in-depth analysis," in *Proc. Conf. on Design, Automation and Test in Europe*, 2016, pp. 854–857.

[5] C. Bolchini, L. Cassano, and A. Miele, "A lifetime-aware runtime mapping approach for many-core systems in the dark silicon era," in *Proc. Conf. on Design, Automation and Test in Europe*, 2016, pp. 804–809.

[6] A. Simevski, R. Kraemer, and M. Krstic, "Increasing multiprocessor lifetime by Youngest-First Round-Robin core gating patterns," in *Proc. NASA/ESA Conf. Adaptive Hardware and Systems*, 2014, pp. 233–239.