PhD position in Computer Science at INSA Lyon

Title: Dynamic Memory Management For Embedded Non-Volatile Memory
Location: CITI laboratory, INSA-Lyon, France
Partners: eVaderis (Grenoble Fr) and Verimag laboratory (Grenoble Fr) [1]
Funding: 3-years PhD grant «Allocation Doctorale de Recherche» from Région Rhône-Alpes
Start date: October 2016
Contact: Guillaume Salagnac +33 4 72 43 64 13 `guillaume.salagnac@insa-lyon.fr`

Short summary

The general context of this project is smart objects, aka the Internet of Things. Technologically speaking, we are interested in next generation System-on-Chip platforms, based on non-volatile memory technologies. Such technologies make it possible to design very energy-efficient embedded systems, but they also require significant changes in terms of software programming. We aim at designing and studying novel memory management mechanisms, both hardware and software, to improve the performance of such systems. More precisely, we will address the problem of dynamic memory management, i.e. allocating, placing, and moving around the various data structures used by the application program.

Recruitment process

Your profile: Master's degree or equivalent in the area of computer science / computer engineering
Required skills:
software: low-level and/or embedded programming in C and/or assembly
hardware: system-on-chip architecture: processor, memory hierarchy
languages: fluent English, both written and spoken ; French is a plus but is not required
Bonus skills:
software: discrete event simulation
work environment: Linux with command-line, scripting language(s)
personal: autonomy, sense of organization, curiosity, initiative
relational: good communication skills, intellectual rigour
How to apply: send an e-mail with one PDF attachment, containing:

- detailed curriculum vitae
- application letter and/or your last internship report
- academic transcripts (including ranking if available) for the last two years of study
- 2 to 3 letters of recommendation, or a list of reference persons and their e-mail addresses

Scientific context

The vision of ubiquitous computing is gradually turning into a reality. Recent progress in semiconductor technology make it possible to design and produce computing chips with very small size and very low power consumption. One key breakthrough in this domain is the emergence of non-volatile embedded memories (NVRAM) which promise to greatly reduce the energy needs of embedded systems. But they also pose significant new challenges in terms of software programming [BCGL11].

In classical embedded systems, the memory hierarchy is based upon two types of memory with opposing features: main memory (RAM) on the one hand, and mass storage on the other hand. Main memory is typically implemented with SRAM or eDRAM. Such technologies provide fast access but are volatile i.e. data is lost as soon as power is switched off. Conversely non-volatile technologies like Flash memory have a high access latency which limits their usefulness to long-term storage. In recent years however, various memory families are emerging which combine both non-volatility and fast, fine grained access. Examples include magnetoresistive

[1] You will mostly be located in Lyon but you'll have to travel to Grenoble several times during the project (expenses covered).

RAM or phase change memory (PCM), among many others. These technologies are collectively described as non-volatile RAM. With this new type of memory, it becomes possible to design innovative embedded systems with a simpler memory hierarchy, and thus to greatly reduce the power consumption of the system [KKSM13]. This is the field of expertise of the eVaderis company, which designs ultra-low power embedded systems [LJB+15; LBJ+16] based on magnetoresistive memory (STTRAM).
For ambient intelligence applications, the advent of NVRAM allows for designing so-called "normally-off" systems, where power supply is kept completely switched off most of the time, thus saving energy. Such non-volatile systems can turn on almost instantly, saving and restoring the software context [AAMS14] when needed so that power cycles are transparent for the user program. The idea is to adapt the duty cycle of the system to applicative requirements, which reduces overall power consumption with little to no impact on performance.

However, naively replacing the whole memory hierarchy with NVRAM is not a good idea [RL14]. Each technology comes with a set of inherent flaws, like write endurance limitation, or high access latency, etc. To mitigate this problem, the architecture of the eVaderis system-on-chip combines several regions of memory with different characteristics. For instance, one region will have low access times, while another region will offer high storage density. As a result, software programs have to be adapted in order to take full advantage of such hardware [MV15]. In particular, the program will have to decide at runtime in which region to allocate each and every piece of data. But the application developer doesn't want to pay attention to such low-level issues. Moreover, we would like to be able to reuse off-the-shelf software components and not have to rewrite everything from scratch.

The goal of this thesis is to address the problem of dynamic memory management in the context of such NVRAM-based systems. Our approach is to co-design both and hardware components and software (OS-level) mechanisms that will work together to monitor the usage pattern of each data structure and dynamically adjust memory allocation accordingly.

Objectives

This project aims at providing a general solution to the problem of dynamic memory allocation for embedded systems with non-volatile main memory (NVRAM). The technical challenge comes from the fact that the application program(s) is (are) written at a higher level of abstraction than what the underlying hardware platform works with. In particular, each piece of dynamically allocated data must be explicitly mapped to one or another region of physical memory with suitable characteristics. Some applications may even require data to be dynamically moved from one region to another during execution. The software developer is not in a good position to manage these low-level details. Instead, our approach is to propose an automated solution, where data allocation and movement are handled as transparently as possible by a dedicated component in the operating system. The idea is to design a dual mechanism in the form of both a (software) memory allocator interfaced with the application, and a (hardware) monitoring block continuously counting memory accesses at runtime. These two components will work together during execution to match the requirements of the application to the capabilities of the hardware.

We have identified several milestones for this project. In a first phase, the PhD student will establish a suite of benchmark programs typical of the relevant application domain: mobile multimedia processing, or data-intensive embedded applications in general. For that, we will not only survey the scientific literature but also work in close cooperation with eVaderis. Some of these programs might be adapted from existing benchmarks, and others might have to be developed for the occasion. This phase will help us identify the right programming interface to provide the developer with. Then, we will set up a lightweight simulation platform (e.g. instruction set simulator) so that we can execute the benchmarks in a controlled environment and study their behaviour. Tracing the memory access patterns will allow us to study how data structures are created and used in practice: where are the allocations, what does the size distribution look like, how are (read/write) accessed distributed, and so on. Some data structures will be "hot" (frequently accessed or modified) while some others will be "cold". We will also look at the effective lifetime of various kinds of data structures. This study about the shape and demographics of data structures will give us a better understanding of the memory profile of embedded applications.

In a second phase, we will use this knowledge to design memory allocation mechanisms. To begin with, we will propose "optimal" allocation policies: assuming that everything is known in advance, how can we place and/or move data structures in memory so as to optimally improve execution performance. For instance, if one allocation site only produces short-lived objects then it is best to allocated them in RAM. Conversely, “cold” data structures could be placed in denser STTRAM, etc. Of course, in real life the execution is not known in advance, so we have to resort to using heuristics. We will derive more practical allocation policies, which we will implement in a dynamic memory manager. The key challenge here is to identify what information is needed by our placement heuristics, and how we can collect this information at runtime without compromising execution performance. Typically, we will propose a solution in the form of “Performance Monitoring Counter” (PMC) modules to be added at certain locations within the architecture. We will then assess the performance benefits by comparing replaying the memory access traces on our model. If the gain is significant enough, the proposed changes will be considered for integration into the eVaderis reference architecture.

The third phase of the project aims at developing a more detailed system simulator, modeling not only the processor but also non-volatile memory features like access latency and of course, the proposed PMCs. This will serve for a global empirical validation of the contributions. The idea is to execute the same benchmark applications running on top of our OS (memory manager) and of the modified platform. In this thesis, we intend to work in simulation only for several reasons. First, because the target hardware platform is not available yet in silicon. Second and more important, we want to propose modifications to the architecture, codesigning the software and hardware components together. Simulation is thus the best approach to quickly explore the design space, assess the performance of the system, and validate our contributions.

References

[AAMS14] Fayçal Ait Aoudia, Kevin Marquet, and Guillaume Salagnac. “Incremental checkpointing of program state to NVRAM for transiently-powered systems”. In *ReCoSoC 2014: 7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip*. 2014.

[BCGL11] Katelin Bailey, Luis Ceze, Steven D. Gribble, and Henry M. Levy. “Operating system implications of fast, cheap, non-volatile memory”. In *HotOS 2011: 13th USENIX conference on Hot topics in Operating Systems*. 2011.

[KKSM13] Emre Kultursay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu. “Evaluating STT-RAM as an energy-efficient main memory alternative”. In *ISPASS’13: IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE. 2013.

[LBJ+16] Christophe Layer, Laurent Becker, Kotb Jabeur, Sylvain Claireux, Bernard Dieny, Guillaume Prenat, Gregory Di Pendina, Stephane Gros, Pierre Paoli, Virgile Javerliac, Fabrice Bernard-Granger, and Loic Decloedt. “Reducing System Power Consumption Using Check-Pointing on Nonvolatile Embedded Magnetic Random Access Memories”. In *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 12.4 (2016).

[LJB+15] Christophe Layer, Kotb Jabeur, Laurent Becker, Bernard Dieny, Stephane Gros, Pierre Paoli, Virgile Javerliac, and Fabrice Bernard-Granger. “Hybrid STT/CMOS Design of an Interrupt based Instant On/Off Mechanism for Low-Power SoC”. In *ISVLSI 2015: IEEE Computer Society Annual Symposium on VLSI*. 2015.

[MV15] Sparsh Mittal and Jeffrey Vetter. “A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems”. In *IEEE Transactions On Parallel And Distributing Systems* 27.5 (2015).

[RL14] Benjamin Ransford and Brandon Lucia. “Nonvolatile Memory is a Broken Time Machine”. In *MSPC 2014: ACM SIGPLAN Workshop on Memory Systems Performance and Correctness*. 2014.