Aeminium: Freeing Programmers from the Shackles of Sequentiality

Paulo Marques, CISUC, University of Coimbra, Portugal
Nestor Catano, CCM, University of Madeira, Portugal
Jonathan Aldrich, ISR, Carnegie Mellon University, USA
João Pedro Silva, Novabase ACD, Portugal

Abstract. Current programming systems shackle developers to a sequential coding paradigm. This paradigm hampers developers from taking advantage of emerging large-scale multicore hardware. We propose a platform which builds in concurrency by default: instead of sequencing code, programmers express dependency information, which is used by a compile-time checker to verify correctness conditions, and by the libraries and runtime system to enable concurrent execution. As a result, developers can write parallel code in a natural style and have confidence in its correctness and performance.

Executive Summary

We are now at the verge of a fundamental change on how software is designed and implemented. Since 2004 processors are no longer doubling their single core performance every 18 months. Instead, all major chip manufactures are developing multi-core processors. Most desktops sold today are dual- or quad-core, and considering Intel is already building 80-core experimental processors, we expect that desktops will be massively parallel within the next 15 years. This creates major software development problems, because current programming languages and software development models shackle developers to a sequential coding paradigm, and are woefully inadequate for designing massively concurrent applications. These problems include: a) lock-based schemes are too error-prone to implement large-scale concurrent applications; b) current industrial programming models are not adequate to extract massive amounts of parallelism from ordinary applications, developed by mainstream programmers; c) existing concurrent programming mechanisms do not scale to take advantage of hundreds of processing units; and d) the data structures and algorithms provided by mainstream platforms (e.g. Java and .NET) primarily support sequential execution, regardless of the parallel resources available. These issues are serious because large multi-core systems are the computing platform of the future. Unfortunately, programmers will not benefit from them if platforms continue to be sequential by default, leaving to the programmer the hard task of trying to achieve effective concurrency.

The Aeminium Project – *Freeing Programmers from the Shackles of Sequentiality* – aims at addressing this fundamental problem: how to provide mainstream programmers with a practical framework for developing massively concurrent applications. In particular, we propose to create a platform which builds in concurrency by default: instead of sequencing code, programmers express dependency information, which are used by a compile-time checker to verify correctness conditions, and by the libraries and runtime system to enable concurrent execution. As a result, developers can write parallel code in a natural style and have confidence in its correctness and performance. We explicitly target modern mainstream object oriented programming environments, like Java and .NET, used by common programmers in the majority of common application domains.

The research will be focused on the following ideas:

- A language where sequencing between commands is not expressed directly by the programmer (i.e., sequentiality by default), but instead is specified indirectly through dependencies between commands. These could be ordinary dataflow dependencies, or they could be dependencies on a side effect of some other command. We track these dependencies through the type system at compile time, and through libraries and the runtime system at run time.

- A runtime platform which has concurrent libraries and data-structures built-in and turned-on by default. Even if programmers write commands with sequential dependencies, they can *by default*, obtain performance improvements from parallel library code when hardware resources are available (e.g. extra cores). The runtime system leverages the dependency information to optimize concurrent execution, making tradeoffs between task granularity and parallelism among tasks.
- A lightweight static verification system checks at compile time that no dependencies have been missed by the programmer, ensuring a lack of race conditions and other common concurrency errors. The verification tool will leverage the PIs' ongoing work on verifying the interface protocols of libraries and frameworks.

To our knowledge, Aeminium is the first proposed concurrency-by-default platform compatible with mainstream object-oriented programming. Our system is natively concurrent; the only way to achieve sequential execution is indirectly through expressed dependencies. Yet for all that, our design supports a natural programming style with functions, variable binding, and mutable state. We reconcile a natural programming style with native concurrency through top-to-bottom concurrency support, including a compiler, lightweight verifier, concurrent libraries, and a concurrency-aware runtime system.

In terms of the team, the project is a consortium of three highly qualified principal investigators and one industrial partner. *Paulo Marques* is an Assistant Professor at the University of Coimbra. His main research interests are concurrent systems, software reliability, and modern programming languages. He has recently led the RAIL and RAIL2 projects on code instrumentation for virtual machines, sponsored by Microsoft Research, and the LeonVM and QERL projects on dynamic code translation for the LEON2 processor, under contract with the European Space Agency. He has published extensively at international research conferences and journals and has authored two books on software development in C#. *Nestor Catano* is an Assistant Professor in the Department of Mathematics and Engineering at The University of Madeira, Portugal. He earned a M.Sc. and Ph.D. in Computer Science from the University of Paris 7, France. His Ph.D. was funded by the European project Verificard (Verification of Java Card programs). He was a research associate at the University of York, U.K, from 2004 to 2006. Nestor's main research interests are the development of JML-based tools and techniques for the verification of programs, and the mechanization of machine-checked correctness proofs of mathematical algorithms. He has published in international research conferences on formal methods. *Jonathan Aldrich* is an Assistant Professor of Computer Science at Carnegie Mellon University and Director of the Software Engineering Minor Program in the School of Computer Science. Aldrich's research contributions include techniques for verifying object and component interaction protocols, modular reasoning techniques for aspects and stateful programs, and new object-oriented language models. For his work on verifying software architecture, Aldrich received a 2006 NSF CAREER award and the 2007 Dahl-Nygaard Junior Prize, the top international award for junior faculty making significant contributions to object-oriented programming. Finally, our industrial partner *Novabase* is one of the largest software companies in Portugal, generating more than 140M€/year in software solutions and consulting. Novabase's ACD (*Advanced Custom Development*) business unit brings a wealth of experience in developing advanced custom software solutions for its customers. Effectively leveraging the emerging class of highly parallel and multicore systems has been a significant challenge for Novabase, and the company will be an active project participant through providing motivating examples and testbeds for ensuring that Aeminium solves critical real-world problems.

We believe that, if successful, this project will make a strong contribution to shaping how concurrent software development will be performed in the future and to strengthening Europe and America's capabilities in this important area.