Applying the VVM Kernel to Flexible Web Caches

Ian Piumarta, Frederic Ogel, Carine Baillarguet, Bertil Folliot

email: {ian.piumarta, frederic.ogel, carine.baillarguet}@inria.fr, bertil.folliot@lip6.fr

1 Introduction

The VVM (virtual virtual machine)[1] is a systematic approach to adaptability and reconfigurability for portable, object-oriented applications based on bytecoded languages such as Java and Smalltalk [FP+00].

The main objectives of the VVM are (i) to allow adaptation of language *and* system according to a particular application domain; (ii) to provide extensibility by allowing a "live" execution environment to evolve according to new protocols or language standards; and (iii) to provide a common substrate on which to achieve true interoperability between different languages [FPR98,Fol00].

On the way to implement a VVM we already implemented VVM1 (and it's application to active networks [KF00]) and VVM2 (and it's application to flexible web cache and distributed observation). The VVM2 is a highly-flexible language kernel which consists of a minimal, complete programming language in which the most important goal is to maximise the amount of reflective access and intercession that are possible—at the lowest possible "software level".

2 Our architecture

The VVM2 contains a *dynamic compiler* front-end/back-end, which converts input into optimized native code and an object-oriented environment (with automatic, transparent memory management) used internally by the VVM2 (this work is under consideration for publication).

3 Example application: flexible web caches

Flexibility in web caches come from the ability to configure a large number of parameters[2] that influence the behaviour of the cache (protocols, cache size, and so on). What's more, some of these parameters cannot be determined before deploying the cache, like: user behaviour, change of protocol or the "hot-spots-of-the-week" [Sel96]. However, reconfiguring current web caches involves halting the cache to install the new policy and then restarting it, therefore providing only "cold" flexibility. Our flexible cache architecture is built directly over the VVM2 and so provides "warm" replacement of policies, without compromising the ease of writing new protocols found in existing web caches. Other advantages include the ability to tune the web cache on-line, to add arbitrary new functionality (observation protocols, performance evaluation, protocol tracing, debugging, and so on) at any time, and to remove them when they are no longer needed.

Our approach supports both initial configuration, based on simulation, *and* dynamic adaptation of the configuration in response to observed changes in real traffic as they happen.

This approach is highly reflexive because the dynamic management of the cache is expressed in the *same* language that is used to *implement* the cache. The resulting cache, called C/NN[3], can be modified at any time: new functionality and policies can be introduced and activated during execution. It is therefore possible to *dynamically* define reconfiguration policies to process adaptations, while preserving the cache contents and delaying request for a few μs.

In order to evaluate the flexibility of our cache we made both quantitative and qualitative measurements[4]. Timing the principal operations in C/NN was trivial because of the use of the highly-reflexive VVM2 at the lowest level. We were able to "wrap" timers around the functions without even stopping the cache. Results are very promising : handling a hit (the main bottleneck for the cache itself) takes less than 200μs, switching from one policy to another takes less than 50μs, at least defining a new policy and re-evaluating 5,000 documents takes a couple of tens of ms. It seems clear that dynamic flexibility does not penalise the performance of the cache. It is also important to consider the ease of use of reconfiguration in our cache : typical replacement and reconfiguratoin functions are short and quickly written (a few minutes for a system administrator).

4 Conclusions

This paper presented and evaluated shortly, due to lack of space, the benefits of using a highly-flexible language kernel, the VVM2, to solve a specific computer science problem : flexible web caching. The resulting web cache, C/NN, demonstrates that reconfigurability can be simple, dynamic *and* have good performance.

We finished to incorporate Pandora[PM00a] into VVM2. Pandora is a system for dynamic evaluation of the performance of web cache configurations: this opens the way for "self-adapting" web caches, were the policies are constantly re-evaluated and modified as *and* when needed.

References

[Fol00] B. Folliot, *The Virtual Virtual Machine Project*, Invited talk at the SBAC'2000, Brasil, October 2000.

[FPR98] B. Folliot, I. Piumarta and F. Ricardi, *A Dynamically Configurable, Multi-Language Execution Platform* SIGOPS European Workshop 1998.

[FP+00] B. Folliot, I. Piumarta, L. Seinturier, C. Baillarguet and C. Khoury, *Highly Configurable Operating Systems: The VVM Approach*, In ECOOP'2000 Workshop on Object Orientation and Operating Systems, Cannes, France, June 2000.

[KF00] C. Khoury and B. Folliot, *Environnement de programmation actif pour la mobilit*, Proceedings of Jeunes Chercheurs en Systemes, GDR ARP et ASF, Besanon, France, June 2000.

[PM00a] S. Patarin and M. Makpangou, *Pandora: a Flexible Network Monitoring Platform* Proceedings of the USENIX 2000 Annual Technical Conference, San Diego, June 2000.

[Sel96] Margo Seltzer, *The World Wide Web: Issues and Challenges* , Presented at IBM Almaden, July 1996.

[ZMF+98] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd and V. Jacobson, *Adaptive Web Caching: towards a new global caching architecture*, Computer Networks and ISDN Systems, 30(22-23):2169-2177, November 1998

---

[1]VVM is both a concept, an implementation and the project's name.

[2]See the configuration file for Squid...

[3]The *Cache with No Name*.

[4]On a G3 233MHz, running LinuxPPC 2000