Session I: Compiler and Runtime Optimizations for WCET Determination

Chair: Raimund Kirner (TU Vienna, Austria)

Presentations

In the first presentation *"Predictable Timing Behavior by using Compiler Controlled Operations"* Vesa Hirvisalo gave a position to use compiler techniques to improve temporal predictability of real-time systems. The motivation is to replace unpredictable hardware and operating system features by more predictable features. As a first approach, caches are replaced by scratchpad memories to improve predictability of memory references.

The second talk given by Hemendra Singh Negi was about *"Simplifying WCET Analysis by Code Transformations"* as a preprocessing step before doing the path analysis to gather information about feasible paths. These code transformations make infeasible paths in programs more obvious and therefore simplify the path analysis. Several examples are given to give an idea how this transformation should look like.

In the third presentation *"Optimizing JVM Object Management Operations to Improve WCET Predictability"* Corrado Santoro presents technical solutions for a Java Virtual Machine to make its timing behavior more predictable. The garbage collection becomes predictable as it is only called directly before allocating memory and frees only as much memory as is needed for the new allocation. Sources of unknown behavior like dynamic classes or dynamic arrays are handled by restricting the programming language respectively by using code annotations.

Discussion

The following discussion centered around the properties of compilers for real-time systems and their possible support for timing analysis.

Christian Ferdinand: I come from the industrial side, working for the company AbsInt. We are providing commercial timing analysis tools. Regarding the point of emitting information by the compiler for timing analysis, all of our customers are using existing compilers. These compilers are typically quite old because for people developing safety-critical applications there is typically no chance to change the compiler. For example in the automotive industry, people are not only using compilers but there is also a high amount of hand-written assembly code in it. So I don't see a chance for the safety-critical area; one cannot change the tool chain and to somehow getting the information from the compiler what optimizations have been performed. For the generation of more predictable code it is the same problem. Depending on the domain it requires many years to define how a safety-critical application is build. For example, there is a design to use ADA or C then it is frozen at the beginning of the project and only ten years later they indeed use C. Ten years is a typical time period for the avionic area when it will come into production. It is not easy to propose anything else then C or ADA. Therefore, I don't see that the industry is going to switch from the current programming paradigms. It will be C for the next 20 years. And still

declining but active there is ADA and that's all. I do not see that the industry is willing to pick up any other coding paradigm for the next years.

Peter Puschner: Coming back to Christian's statement, of course, I see the current situation in industry. And of course, things will change very slowly, I agree. And we shouldn't have any illusions on how things will develop and I think it would be quite difficult to get kinds of *revolutions*. We will rather have to expect *evolutions*, but I think we should not be too pessimistic even if it takes some time. It is the purpose and also the responsibility of research to come up with new solutions and to work out new solutions. It is our responsibility to push the state of the art into a direction. Someday, people cannot deny that technology is there, that certain techniques are state of the art and have to be used. This sometimes happens. If you build a system and the system puts people into problems and later on you find out that the tools that have been used do not reflect the state of the art. After some time of course, everybody will accept it. It will take some time to reflect research results in industrial technology but at some time I think our research results have to be picked up. And so, even it takes time, I would not say that we will not ever see them as strictly as you [Christian Ferdinand] did in your first statement. I would say that would make any research ridiculous and useless. But that is not the case.

Christian Ferdinand: Maybe, that was a little bit too pessimistic. It is not to discourage you in that area. Probably it is a kind of dilemma. As for the most safety-critical applications like fly-by-wire, airbags in cars and all such things, I see people that do have a budget and might be interested in using WCET tools and say that they will not change other tools, so that is the dilemma. But there are huge other areas which are less critical, like telecommunication. They use also C or in some cases Real-Time Java. These areas are much less critical but they are typical in projects where timing is not so critical, i.e. it is more soft real-time. They are not really interested in the worst case but in some worst average case or something like that; things that can you usually do with measurements. But they are not going to pick up the tools, they are not going to invest into them. But this community provides tools that are working for them, they are working for Real-Time Java and maybe some other languages, there is a chance. But currently, these tools are very specially and very expensive, and they are only going to be used for the most critical system designs.

Iain Bate: Listening to the conversation of you, you are both right. The problems we are always have to face in the critical systems domain is legacy. Therefore, we often have to keep the antiquated approaches like frames systems development. And any such approach has to deal with legacy. What we always have to remember is that these are large legacy components which are hard to deal with, using languages like ADA or C; they are conservative coding style approaches. And the other thing to remember from the industrial practice is that timing analysis is only a minor part of what they have to do. And anything that we propose like code transformation has to be weighted off against the bigger problems like functional verification, unit testing, etc.

Guillem Bernat: One extra comment to this special area is that now there is already the awareness that there is a problem of finding the worst-case timing behavior. A few years ago there were these simple chips where worst-case analysis tools where easy to build and the hardware was easy to analyze. Now we hear somewhere in the avionics they were starting using caches, pipelines and very complex processors. There is a lot research to do. People are already committing to using these advanced processors and then they say: *"oh, we have a problem and we do not know how to do in that case"*. As you say, it is changing very slowly, but engineers are starting to use these processors even in very critical systems. The other point on the compilers is that it is true that people buy the compiler and stay with that compiler. But we had some experiences recently where compiler vendors where approaching to us: if our compiler knew how to do the WCET

analysis we can sell it to more customers. This is the first time we got this message of compiler vendors now trying to find an edge on their problems, having more real-time features, support for real-time operating systems, and a lot of support for instrumentation and WCET. To grab the argument from Peter [Puschner] and you [Iain Bate]: things are changing, slowly but they are changing. The good thing is that we must be there when people start asking for this.

Raimund Kirner: I can just express my opinion of how it works in industry, they just pick every solution that is available ready to use, that's all. In a company in that area there is no significant time for research or development budget. But there is also a good chance for applying research on WCET compiler support to industrial practice. For certain new chip products a compiler has to be written from the scratch, for example, for some special signal processors and so on. In such a situation it could be useful to think from the beginning which data structures and mechanism are suitable to support aspects like predictability or timing analysis. And as Guillem [Bernat] said, it helps if compiler vendors are looking for niche markets by supporting timing analysis. Then if other people see that it works quite well it may also convince them to adapt such technologies. It may be still a question of time, but as long as we do not have a solution, industry will not adapt it, for example, for their compilers. It may still take 20 years, but at least we have to do the basic research, because timing analysis is still a very serious problem. And of course, if everyone just says that due to all the legacy systems it is too complicated to do anything, there will be no change. But of course, this will not bring any advantage for the software development process used by them. Luckily, development departments of many companies already think of using better and more structured software development processes. And I think, for new projects, they are also willing to change their whole software development method to achieve more predictability. Therefore, I think there are some realistic possibilities to introduce these technologies.

Peter Puschner: If I am allowed I would like to ask a question, actually more than making a statement or answer to one of these statements we had before. Since we had been talking about transformation, every compiler does some kind of code transformation. If we speak about optimization it does a little bit more than just code transformation, but even translating from C to machine language is some kind of transformation. Since we have some people here from industry and also other people are dealing with industry or having contacts to industry, I would be interested to know what are the limits of transformations that do you think would be allowed in order to generate code for a safety-critical system. I don't know if it is possible to define such a limit, but I think that there have to be some transformations and I know it is easy to certify compilers which do very simple transformations from source code into machine code, but where does this simplicity end when you want to certify a system? Is there any idea or does anybody have an idea how this can be expressed, because I think the second talk[1] pointed out to - at least from a research point of view – a very interesting solution to the problem of WCET analysis. But I heard a comment that it would not be feasible in an industrial setting, so therefore my question.

Iain Bate: I think the answer is easy. What you are allowed to do is driven by cost. If it is cheaper to transform the code and just skip the transformation and do the analysis without the transformation you will be allowed to do it. But you have then go to the customer just to define the transformation. It is a cost tradeoff. There is nothing to stop you doing optimizations and transformations as long as you can show that you do not introduce any additional hazards into the software. And that is the tricky bit, showing it for all conceivable cases to that it might be applied. Normally, the reality is that they just say that you cannot just do it. But I see a lot of work on tools like code generation, you know, tools like SCADE, that successfully brings models into the

[1] H. S. Negi, A. Roychoudhuri and T. Mitra: *Simplifying WCET analysis by code transformations.*

code and compile them down. That is allowed because people did a lot of efforts just to define those transformations which was very expensive the first couple of times they did it. But what they consider is that it was worth doing the transformation. Any compiler just does transformations.

Raimund Kirner: I am not really the expert in this area on certification. So, I'm interested about what are compiler optimizations – let's say code transformations to improve performance – are currently used for certified systems in the avionics. As far as I know they already use code transformations to improve performance. Maybe Iain [Bates] knows some details on this.

Iain Bate: No, they do not use optimizations. Still, they turn off all optimizations in the compiler.

Raimund Kirner: I was asking because I heard from some unofficial sources that people are also thinking about using compiler optimizations in the avionics.

Iain Bate: It depends on the system. There are a lot of systems in the avionics, on an aircraft like the flight control system which is one the most critical system on an aircraft and any form of optimization will not be considered. Then, the various integrity levels down from that, low integrity level, less real-time, less critical systems, then certainly code optimizations would be considered. Though, on those systems they probably do not do WCET analysis, they would use more test-oriented methods. But certainly, on the parts of the system where one is doing static analysis one would not turn the optimizations on. Would you agree?

Christian Ferdinand: If you have a verifier, a tool that can verify that the optimized code is also correct, i.e. corresponds to the source, you can do it. But as long as such tool is not available they will probably switch off all optimizations.

Stefan M. Petters: I heard once the argument, probably not in the highest critical systems area, the most used optimization would be actually "-O1" in terms of – and there comes the argument now - this is the most often used compiler optimization stage and as such is most trusted compared to the no-optimization setting. I cannot confirm that this is really the case but I have been told that.

Guillem Bernat: One thing that we should not forget is that the scope for which WCET applies is quite broad. On one extreme there are the absolute safety-critical systems as the avionics. What they actually say is about you do not trust the compiler. On another scope we have the people in the automotive industry which are very keen using WCET approaches because they start having to share the same chip with other applications and they have budgets. They also have to guarantee timing properties as this is safety-critical but not at the level of the avionics. And now we have the telecom industry where something does not work and this is always the problem. And possibly the timing problems, how do we go to analyze this? My suggestion is that we try to keep the conversation open in the sense that if we talk about WCET, some people always do the absolutely safety-critical, some do the soft real-time systems. Actually, we should try to get a broader view. Some aspects cover all these approaches. For example the real-time application of Java which I do not have seen flying in a plane but doing a lot of other efforts.

Iain Bate: If you look at the avionics standards they define critical systems of possibly causing harm of lives. If you look at some standards like IEC 61508 which is a reference standard for programs and electronic controllers, etc. They define critical systems if it could loss life or economic damage in case of a failure. And therefore, to a certain extent I agree with Guillem [Bernat] in a sense that there are a lot of people at telecommunication, mobile phones manufacturers etc. who are beginning to get more willing to consider execution time analysis

because if the system fails and they have shipped a millions units that it is very costly, and that is a different type of critical system, where economics is driving it. They spend much more efforts into verification, certification, power proofed, etc.

Christian Ferdinand: In avionics everything is regulated, so there is a need for certification. In the car manufacturers OEMs and suppliers there is no such thing, there is no regulation and they typically use all optimizations the compiler provides, even for the safety-critical applications.

Stefan M. Petters: Not yet! They use not yet a standard, but…

Christian Ferdinand: Yes, but even if it involves them the people *insist* that they *can use all optimizations* that are there. And if you look at the USA and the critical systems there, there is a danger for the OEMs there that they will be filed with a huge amount of money if they fail to use state of the art. So, if there are tools that show the absence of errors, so to say in the timing domain the WCET tools, they are going to use them to avoid these damages. So, *we have to have WCET tools for the car manufacturers that also work with the highest level of optimization.*

Raimund Kirner: I want to make an open comment on this because when you think about introducing compiler support for timing analysis, of course, it requires a lot of changes. For really safety-critical systems there is just a very small market compared to other systems. Therefore, it might be also useful to consider the application of timing analysis for domains that do not require absolutely safe bounds. This may also introduce WCET analysis techniques and also compiler support into mass production. If this happens it will also convince people from hard real-time systems domain that compiler support for WCET and predictability would help to improve the overall design flow. This would be a chance we can think about.