TANGO Framework: The Embedded Programming Target

Bruno Wéry, Sébastien Magdelyns - DELTATEC

TANGO is a European Project funded under the H2020-ICT-04-2015 programme. TANGO aims at producing an energy aware development framework for parallel programming on heterogeneous hardware.

The objective of this white paper is to demonstrate the pertinence of the framework provided by TANGO when applied to the embedded programming world and the conditions under which the components and the tools of this framework are relevant in this context.

Introduction: A Parallel World

Two decades ago, parallel programming was a type of computation restricted to large scale specialised applications that were running in computing centres. Today's picture is very different with the generalisation of multi-core processing devices, not only inside desktop computers, but inside embedded and mobile devices as well.

Advances in vision technologies is a major fact of the beginning of this century, leading to the creation of small devices that recognise their environment. These image processing technologies are naturally parallel. This is an example of the way parallel computing moves from specialised software design teams towards all developers.

The Internet of Things (IoT) and cloud computing are other examples of domains in which concurrent cooperative programming develops. That-is-to-say parallel programming.

While the multi-threading concept appeared "advanced" twenty years ago, it is part now of the day-to-day life for all programmers. The same move is now ongoing for parallel programming techniques.

Massively parallel computing has entered the general purpose computing market through customer oriented GPUs in desktop computers. It is now a standard technology for applications like gaming or imaging. This technology spreads to the embedded or mobile devices through the introduction of System-On-the Chip (SOC) devices including not only multiple CPU cores, but also GPU accelerators.

A recent move is the introduction of FPGA accelerators not only in the high performance computing market, but also into the embedded market through the inclusion of such programmable logic into SOC devices.

Therefore, tooling for parallel programming is now of highest importance for all developers. While techniques in which parallelism is explicitly expressed through the design of inter-process communication schemes were acceptable for specialised teams designing a few large scale applications, today programmers need efficient tooling that hides parallel computation management complexity. Programming parallel processes therefore must become as natural as sequential programming. This is true for all markets from embedded to high-performance computing, including office, leisure, industry or communications.

As heterogeneous environments become more and more widespread, these tools have to handle them, providing in the most transparent way the possibility to use the best device for each job.

Moreover, today computing efficiency is not the single focus. As High Performance Computing (HPC) centres grow and multiply, their energy consumption becomes a major concern. As mobile devices become more and more powerful, their autonomy or small size become key competitive assets. As a result, “best device” means today not only the one that provides the best computing performance but also the best energy efficiency.

An objective of TANGO project is to provide a toolbox that will help unifying the way parallelism is considered in the different programming market segments, optimizing both computing and energy efficiency, helping to make appropriate trade-offs through adaptation and Quality-of-Service (QoS) management.

What is an Embedded Application?

TANGO framework addresses all the range of parallel computing applications. However, this white paper focuses on the embedded market needs. Our goal is to highlight the way TANGO can fulfil these needs, while maintaining a maximum coherence with the other markets.

Let’s first define what we mean by “embedded device”. It does not mean “small” or “mobile”, but “specialised” and “autonomous”.

A smartphone or a tablet is not automatically an embedded device considering this meaning: as long as it executes various applications, and that these applications can be installed, started or managed on user requests, it is a general purpose computer.

A smartphone, a tablet, an embarked processor or a desktop computer becomes an embedded device once it is dedicated to the execution of a specialised task or application and runs autonomously. For instance, a system that controls traffic lane alignment in a car is an embedded system. A dedicated GPS is also an embedded system, while a GPS software that is executed on a smartphone is not, even if very similar.

The underlying hardware infrastructure is not the element that distinguishes embedded computing from general purpose computing or high performance computing. If embedded computing can use small processors, SOCs or very specialized platforms (which may be heterogeneous), embedded computers may be based on desktop or server computer architectures similar to those used for general purpose or for high performance computing. Operating systems may be the same, even if their range is wider.

The following properties and constraints are attached to embedded applications. They do not relate directly to the computing power of the platform but to the nature of the application.

- The application architecture is “communication”, “service request” or “event” driven. The computing part of the application is embedded into a processing context that manages communications and external events. This means that the computing part of the application is not its single task. Other threads or processes performing management, interface, data creation or communications in parallel are also included inside the application. In addition, the computing tasks are often not running in the main application thread or process and but started asynchronously into secondary threads as a function of the incoming events or requests.

- The application is either real-time or responsive: Most embedded application must process data within a strongly constrained time-frame. For some of them, a real-time behaviour is expected. For others, the responsiveness has a significant impact on user satisfaction.
- Computing tasks are smaller in the embedded world than in the high performance computing world, as they relate often to “instantaneous” processing. This means that parallelization must rely on grains that are significantly smaller than in the usual HPC applications.
- Resources in embedded systems are limited. The system will include usually a small number of computing elements. Often, there is only a single CPU. This means that the management software must be hosted on the devices that are used for computation and supported by the same operating system. This means that the management software footprint and computing load must remain low compared to those of the application.
- Systems are usually deployed statically and run in a continuous way: the application is loaded and started when the system is powered-on and run until the system is powered-off. The management of multiple users or requests is performed by the software itself, not by an external manager nor a launcher nor the end-user himself.

Another key element that distinguishes embedded computing from HPC, which is shared with general purpose computing, is the multiplicity of applications and the need for fast time-to-market development cycles.

Requirements for an Embedded Computing Framework

In the embedded world, the following key-features are expected from a parallel programming framework:

- Small footprint: As resources are limited, they may not be committed significantly to the management of the framework. The framework runtime must be operated using resources that are shared with the computing part of the application.
- Small overhead: Because resources are limited and the system must be very responsive, the task granularity is small and the framework overhead has a direct impact on performance.
- Programming simplicity: Programming efficiency is of highest importance for time-to-market.
- Versatility: the framework should allow integration in various system architectures with different system level complexity. It must be adapted to real-time and communicating application architectures.
- Scalability and possibility to extent over networks: This is needed to cope with the multiplication of sensors in applications and with the needs for the communicating devices in the IoT.

This is of course to be added to the need for management of heterogeneous platforms, including CPUs, GPUs, FPGAs or even dedicated accelerators.

TANGO project also focuses on energy management and power savings. This objective is perfectly relevant for the embedded market. While ecological impact of energy saving in the embedded market is expected for no more than a few applications, energy efficiency anyway is a key point for designers.

Reducing energy consumption in embedded applications enables the simplification of power supply schemes, the increase of autonomy, the reduction of battery weight or the increase of their life duration and the reduction of power dissipation. As a consequence, this may lead to case size reduction, mechanical simplification (i.e. fanless design) and reliability improvement.

The TANGO framework for the embedded world

The TANGO framework is a toolbox that supports parallel programming on heterogeneous platforms, with a strong articulation around a programming model and an underlying run-time abstraction layer.

To be a complete framework, TANGO includes a large set of components, all not being relevant for embedded applications. For instance, a set of components are built around a workload manager dedicated to the deployment of applications on a large set of computing nodes. While such a component is mandatory in the HPC world, it is not relevant when there are only a few computing devices executing the same code all the time.

The implementation of some components may also vary as a function of the targeted market.

In the following paragraphs, we present the foreseen TANGO tools that will bring value to the embedded market.

More details about the TANGO tools can be found in project deliverable document D2.2: “TANGO Requirements and Architecture Specification, Beta Version”

TANGO Programming Model and Run-Time Abstraction Layer

The TANGO programming model helps creating programming code for parallel execution on a heterogeneous hardware architecture. Its strength is that it is directive driven: the code remains similar to a sequential code and parallel aspects are described through directives that drive a code generator that creates the actual parallel code. As the code itself keeps its sequential semantics, the code can still be compiled for a sequential execution. This fact, of course, eases the verification of code correctness and debugging.

Even if some code for specific devices must be written using dedicated language extensions, for instance CUDA or OpenCL, this code is finally integrated through a simple function call, the programming model compilation chain managing the generation of the “glue” code.

Moreover, the programming model supports distribution of code on multiple independent platforms. It is a major feature for embedded computing, where software to be distributed on multiple dedicated computing devices may be managed within a single source code.

Another great feature of the programming model is the asynchronous execution managed by the underlying run-time abstraction layer. In this programming model, the execution model is task based. Directives inserted in the code split the executions into tasks, identifying the input and output relationship between these tasks. The run-time is able to schedule these tasks accordingly to the availability of data produced by the other tasks. This provides the programmer a very simple way to manage efficiently the workload on its system, optimising easily available resource usage.

The TANGO programming model is therefore a very promising component, whatever is the targeted market. It brings its highest added value for an embedded use case as it simplifies parallel programming and therefore is a key element for time-to-market reduction.

Requirement & Design Modelling Tool

A key decision in programming an embedded heterogeneous system that includes multiple computing devices is the distribution of tasks onto these different devices. To take a decision, one must consider not only the computing capabilities of each device but memory locality and communication link availability and bandwidth as well.

Because code optimisation may be deeply linked to the nature of the target device, making the best choice before implementation is important.

TANGO toolbox will provide the Requirement and Design Modelling tool that will enable simulating the execution of the global process considering these different aspects using execution time evaluations and hardware specifications.

Code Profiler

The profiler tool is dedicated to the determination of the energy foot print of the code. This tool should enable the developers to identify energy hot spots in different sections of their code and therefore help them to determine the parts that need optimisation to reduce the energy consumption.

Application Life-Cycle Deployment Engine

This component devoted to the deployment in large high performance computing centres may look having no relevance into an embedded context, where the number of devices is limited, while loading is static and may be configured once for the life of the system.

Nevertheless package building functions that are proposed in the ALDE component will be of interest for the embedded use-case, particularly if they are coupled with the R&D modelling tool, to ease the creation of alternate configurations during the development cycle.

Self-Adaptation Manager

Another great component for the development of embedded applications is the Self-Adaptation Manager that the TANGO framework will propose.

Self-adaptation management relates to the implementation of the same computing tasks on different devices, with dynamic selection of the actual execution target during operation, as a function of different parameters including QoS and possible energy savings.

This self-adaptation can provide an intelligent way for managing the execution of multiple processing requests from clients or users, for instance when the number of accelerating devices like GPUs is limited, or when the faster device is not the most energy efficient one.

This component is one that may have a different implementation as a function of the target market. For the high performance computing market, this self-adaptation can be performed on a node and process granularity. Therefore, in HPC, it can be deeply associated with the workload management software. In Embedded world, self-adaptation should be performed at the level of run-time tasks, that-is-to-say at the level of programming methods. Therefore, the self-adaptation mechanism requires integration with the programming model to select the best execution support for these tasks or methods.

Conclusion

The TANGO framework will propose the parallel programming community a set of valuable tools, and particularly an advanced programming model that will provide support for hardware heterogeneity, multi-node processing and asynchronous execution, with a strong focus on energy management.

As product time-to-market and project development life-cycles are increasingly critical in many competitive companies, the unification of the tools exposing abstracted and easy-to-use layers that

accelerate the overall development process compared to the actual complexity of heterogeneous systems is a key feature of the framework provided by TANGO.

This framework does not target only the HPC market, but also the general purpose computing market as well as the embedded market. Providing a single workflow for all of them is the very ambitious challenge of the TANGO project.

In this white paper, we tried to demonstrate the relevance of this approach for the embedded market while identifying the requirements that the TANGO framework has to address to be successfully adopted by the embedded world.