MANGO: exploring Manycore Architectures for Next-GeneratiOn HPC systems

José Flich*, Giovanni Agosta†, Philipp Ampletzer‡, David Atienza Alonso§, Carlo Brandolese†, Etienne Cappe‡‡, Alessandro Cilardo¶, Leon Dragić‖, Alexandre Dray**, Alen Duspara‖, William Fornaciari†, Gerald Guillaume**, Ynse Hoornenborg††, Arman Iranfar§, Mario Kovač‖, Simone Libutti†, Bruno Maitre‡‡, José Maria Martínez*, Giuseppe Massari†, Hrvoje Mlinarić‖, Ermis Papastefanakis‡‡, Tomás Picornell*, Igor Piljić‖, Anna Pupykina†, Federico Reghenzani†, Isabelle Staub**, Rafael Tornero*, Marina Zapater§, Davide Zoni†
*Universitat Politècnica de València, †DEIB – Politecnico di Milano, ‡Pro Design GmbH,
§ESL – École Politechnique Fédérale de Lausanne (EPFL), ¶Centro Regionale Information Communication Technology SCRL,
‖University of Zagreb, **Eaton Industries SAS, ††Philips Medical Systems, ‡‡Thales Communications & Security
Email: *nsurname@disca.upv.es, †name.surname@polimi.it, ‡philipp.ampletzer@prodesign-europe.com,
§name.surname@epfl.ch, ¶acilardo@unina.it, ‖name.surname@fer.hr, **NameSurname@eaton.com,
††ynse.hoornenborg@philips.com, ‡‡name.surname@thalesgroup.com

***Abstract*—The Horizon 2020 MANGO project aims at exploring deeply heterogeneous accelerators for use in High-Performance Computing systems running multiple applications with different Quality of Service (QoS) levels. The main goal of the project is to exploit customization to adapt computing resources to reach the desired QoS. For this purpose, it explores different but interrelated mechanisms across the architecture and system software. In particular, in this paper we focus on the runtime resource management, the thermal management, and support provided for parallel programming, as well as introducing three applications on which the project foreground will be validated.**

***Keywords*-High Performance Computing, Customization, Energy Efficiency.**

I. Introduction

The push towards Exascale is going to radically change High-Performance Computing (HPC). First, the sheer amount of computational resources available are pushing the energy envelope available through the power grid to the point where the size of an HPC centre may be constrained by the availability of power supply. Second, the increase in scale of HPC resources across the world is enabling new use case scenarios, where players previously unable to access HPC resources may now do so through innovation in delivery modes, e.g. through cloud HPC [1]. Thus, the evolution of HPC hardware and software architectures needs to embrace technologies that allow high performance at low power consumption. The current trend is to leverage application-based customization to this end. Deeply heterogeneous architectures can provide such performance/watt improvements, but are clearly much more difficult to program and manage. Furthermore, new application classes are entering the HPC domain that are QoS sensitive. In particular, time-predictability is a key need for applications such as video transcoding or medical imaging. Since time-predictability and QoS are often not taken into account in HPC, there is the need to explore these challenges, extending the traditional optimization space from power/performance to *power*, *performance*, and *predictability* – the PPP space. In fact, predictability, power, and performance appear to be three inherently diverging perspectives on HPC.

The key goal of MANGO [2] is to address the PPP space by achieving extreme resource efficiency in future QoS-sensitive HPC. The research investigates the architectural implications of such emerging requirements of HPC applications, to define a new generation of high-performance, power-efficient, deeply heterogeneous architectures with native mechanisms for isolation and QoS.

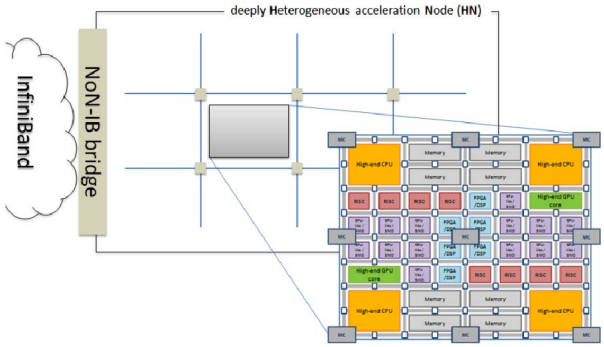

Fig. 1. MANGO Hardware Architecture

A. The MANGO Approach

The performance/power efficiency wall poses the major challenge faced nowadays by HPC. Looking straight at the heart of the problem, the hurdle to the full exploitation of today computing technologies ultimately lies in the gap between the applications' demand and the underlying computing architecture: the closer the computing system matches the structure of the application, the most efficiently the available computing power is exploited. Consequently, enabling a deeper customization of architectures to applications is the main pathway towards computation power efficiency. Theoretically, customization can enable improvements in power efficiency as high as two orders of magnitude, since it enables the computing platform to approximate the ideal intrinsic computational efficiency (ICE), defined as the energy consumption per operation achieved by purely computation circuits, e.g. FP adders.

The current uncertainty regarding on-chip HPC solutions and the essentially open nature of current architecture-level research will be regarded by MANGO as an opportunity, rather than a limitation. The fundamental intuition behind the project is that effective techniques for both performance/power efficiency and predictability ultimately share a common underlying mechanism, i.e., some form of fine-grained adaptation, or customization, used to tailor and/or reserve computing resources only driven by the application requirements. Along this path, the project will involve many different, and deeply interrelated, mechanisms at various architectural levels, from the heterogeneous computing cores, up to the memory architecture, the

Publisher's version: DOI 10.1109/DSD.2017.51

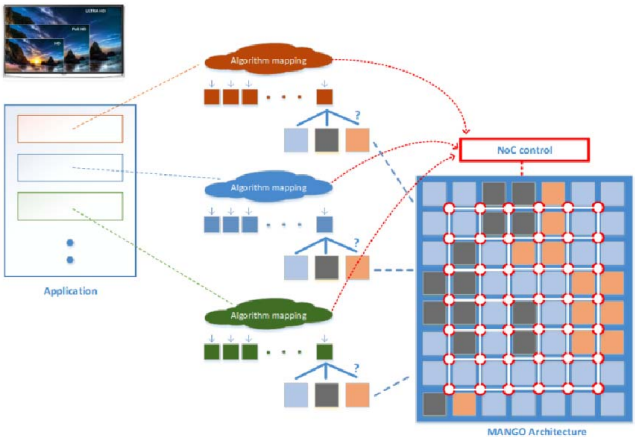

Fig. 2. Mapping Applications on the MANGO Platform

interconnect, the runtime resource management, power monitoring and cooling, also evaluating the implications on programming models and compilation techniques. In particular, to explore a new positioning across the PPP space, MANGO will investigate system-wide, holistic proactive thermal and power management aimed at extreme-scale energy efficiency by creating a hitherto inexistent link between hardware and software effects, which will involve all layers of an HPC system, from server to rack, to datacenter. The combined interplay of the multi-level innovative solutions brought by MANGO will result in a new positioning in the PPP space, ensuring sustainable performance as high as 100 PFLOPS for the realistic levels of power consumption ($<$ 15MWatt) delivered to QoS-sensitive applications in large-scale capacity computing scenarios. MANGO will provide essential building blocks at the architectural level enabling the full realization of the long-term objectives foreseen by the ETP4HPC strategic research agenda [3].

B. Organization of the paper

The rest of the paper is organized as follows. In Section II, we introduce the three application scenarios that drive the project. In Section III we describe the targeted MANGO architecture. Then, in Section IV we describe the programming models and runtime management resources to be used in MANGO, and in Section V the thermal and cooling innovations proposed in the project. Section VI shows the prototyping roadmap. Finally, we draw our conclusions in Section VII.

II. MANGO Application Space

The MANGO project aims at showcasing the need for a dynamic nature of the hardware/software architecture and its capabilities to dynamically use heterogeneous processing elements in a QoS sensitive computing scenario. Therefore, the project draws its requirements and performs its validation on a set of three applications that involve significant QoS aspects.

A. MANGO architecture online video transcoding application platform

Multimedia playback on different presentation devices has experienced significant growth. Some market forecasts [4] show that annual global IP traffic will pass the zettabyte (1000 exabytes) threshold by the end of 2016, and will reach 2 zettabytes per year by 2019. In the same time, globally, IP video traffic will be in the range of 80 to 90 percent of all IP traffic (both business and consumer) by 2019.

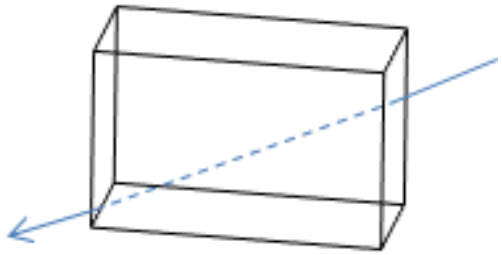

Fig. 3. Ray casting through a volume [8]

Because of the great variety of devices which are accessing the multimedia content under adverse network conditions, current video streaming systems in most cases do not provide optimal video quality and thus waste valuable resources or unnecessarily lower the Quality of Experience (QoE).

Efficient processing of video transcoding, which is extremely compute-intensive and has stringent timing requirements, provides an ideal case-study for the QoS-aware HPC solutions explored by MANGO. The heterogeneous core MANGO HPC transcoding application platform can therefore truly demonstrate how the novel MANGO HPC architecture may become dominant architecture for applications that generate more than 80% of global internet traffic. The application of same algorithms to medical domains where interoperability requirements are defined (see [5]) is also under consideration.

The real time video transcoding will use novel video coding algorithms such High Efficiency Video Coding HEVC/H.265. To enable efficient transcoding, significant work will be required in modeling, mapping and optimizing parts of the algorithms to different underlying MANGO architecture elements (tiles), as shown in Figure 2. Research will not only be focused to high optimization of SW implementation but also design of application specific tiles, implemented in HW (such as [6]), that will allow more efficient processing from performance, power and QoS points of view.

B. Volume rendering for medical imaging

Philips will deploy the Volume rendering technique on the MANGO prototype. Philips ships a variety of imaging equipment, which includes MRI (Magnetic Resonance Imaging), CT (Computed Tomography) and PET (Positron Emission Tomography). In such equipment, medical images are stored as sets of parallel planar images. Such a set can be stacked together, which forms a 3D rectangular space around the stacked images. We refer to such a set as a volume. Volume rendering is a visualization technique that uses the ray casting technique to visualize a volume on a 2D plane. Ray casting visualizes this volume by calculating the attenuation of rays of light [7].

In Ray Casting, the color of each resulting image pixel is determined by following a single ray of light through the volume, see Figure 3. Density values in the images of the volume are sampled along the ray. Transfer functions for the color and transparency, determine the color and transparency of these sample points along the ray. The attenuation of these colors and transparencies along a single ray will determine the color of a resulting pixel. This process is repeated for every pixel in the single visualization result image. The calculations are highly memory intensive (data sizes range from 250MB to 1GB). Fast rendering and a low latency transfer from central processing to the users workstation are crucial. This is to

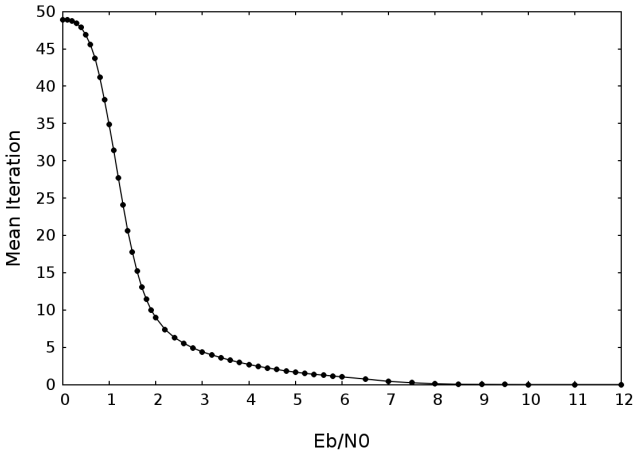

Fig. 4. LDPC convergence (mean iterations) in relation to SNR (energy per bit to noise power spectral density ratio, E_b/N_0)

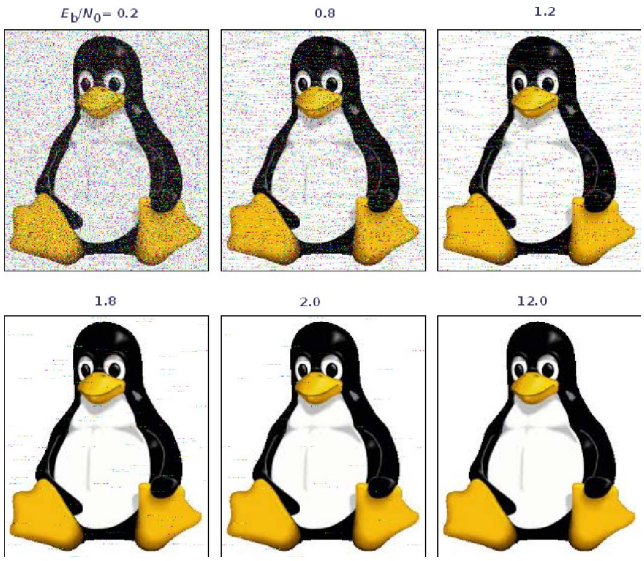

Fig. 5. LDPC convergence in relation to SNR

ensure a better hospital workflow and better diagnosis outcome. The solution must scale among many healthcare users, all operating on different patient data. Users interact with the system in real time, requesting new renderings with frequencies of up to 25 times per second. In more severe cases, imaging equipment is increasingly used during minimal invasive intervention. As the surgeon cannot see what he/she is doing, an image stream with low latency and low jitter has to be presented. The MANGO solution will allow Philips to improve the offering for diagnosis equipment to the hospitals. The final product will much better serve the interventional market.

C. Error correcting codes in communications

Thales will explore the offloading and parallelization capabilities of Low Density Parity Check (LDPC) using the MANGO architecture. LDPC is a linear error correcting code, a method of transmitting a message over a noisy transmission channel [9]. It is among the most efficient error-correction techniques and although it has been invented in the 1950s, it was practically used from the 1990s [10] due to the fact it requires high-end floating point computations. Nowadays, LDPC is used in a growing number of standards in wireless and satellite communications such as WiFi, Wimax, DVB-S2 [11]. In communications, data integrity can be severely impacted from the transmission channel's interference, noise and fading. Error correction algorithms are used in order to maintain transmission capacity by tolerating a loss in transmission rates and latency. The compromise in transmission rates is due to the additional information relating to data redundancy tables and information reconstruction which increases the payload size and reduces the useful bandwidth. Similarly, end-to-end latency is added because of the processing time needed to encode the useful payload at the sender and for decoding at the receiving end. Since communication capacities and storage capacities are growing, it in necessary to provide architectures where the scalability of LDPC can be maintained.

Figure 4 shows the calculations needed to reconstruct the information depending on the Signal-to-Noise Ratio (SNR), in the form of number of iterations (limit fixed to 50). As the noise in the channel decreases (increasing SNR), less iterations are needed to converge. In Figure 5 we can observe the quality of reconstruction of an image in respect to the different levels of SNR in a simulated noisy channel.

Through the advancements in the MANGO project, Thales will benefit from the performance increase, providing QoS guarantees to time sensitive transmission (ex. voice, video) and from energy efficiency. In more detail, parallelism will allow to obtain a significant performance gain in terms of the number of transmission flows that can be active simultaneously. In addition, dynamic resource allocation and heterogeneity will enhance time predictability and energy efficiency. This will be achieved by adapting the aggressiveness of parallelism and the nature of the processing nodes to the channel's SNR in order for all communications to maintain their latency bounds. Finally optimally placing tasks will improve resource usage and provide energy efficiency.

III. Hardware Architecture Concept

At the architecture level, the MANGO project foresees a scenario where General-purpose compute Nodes (GNs), hosting commercial-off-the-shelf solutions (e.g. Intel Xeon Phi processors or high-end NVIDIA GPU accelerators), coexist with *Heterogeneous Nodes* (HNs), forming a common HPC infrastructure. HNs, as depicted in Figure 1, will essentially be on-node clusters of next-generation manycore chips coupled with deeply customized heterogeneous computing resources. The manycore architecture will be open, it will not rely on COTS solutions available today, but rather it will enable broad-spectrum, ground-breaking research in the area of on-/off-chip architecture. Building on recent trends in HPC research, in fact, HNs will allow borrowing solutions from the embedded/System-on-Chip domain, which is now recognized as a promising pathway to extreme-scale low-power HPC. HNs will contain a multi-chip mesh of power-efficient RISC cores augmented with custom vector resources (SIMD and lightweight GPU-like cores) as well as a dedicated memory architecture and a custom Network-on-Chip providing advanced support for partitionability and time-predictability. The cores in the multi-chip manycore architecture will be connected through a *Network-on-Node* (NoN), forming a continuum at the off-chip (on-node) level from the on-chip interconnect.

Since the first stages of the project, the architecture exploration will be extensively supported by a purposely developed emulation platform. HNs will not be prototyped in a final ASIC form, but a mixed approach will be taken. In fact, RISC processors will be instantiated as ASIC cores tightly coupled with a large-scale

reconfigurable hardware fabric used to emulate in near real-time the customized acceleration units, the advanced memory management architecture and the NoN, as well as the NoN bridge to the external interconnect. The platform will support fast design space exploration and validation of the solutions at both the software- and thermal/power-level. These techniques will inherently involve multiple aspects within the system, from programming down to the architecture definition, deeply intertwined with chip- and system-wide control mechanisms of physical parameters, primarily power consumption and temperature. To gain a holistic understanding of their impact on performance/power/predictability (PPP) and quantitative information about their effectiveness, MANGO will also develop a comprehensive toolset for PPP and thermal models, which will operate in close relation with the PPP run-time information collected from the platform.

The MANGO experimental platform will include 16 GN nodes with standard high-end processors, i.e. Intel Xeon E5, as well as NVIDIA Kepler GPUs, along with 64 HN nodes. GNs and HNs will be connected through InfiniBand. HNs will contain ASIC ARM cores and a high-capacity cluster of FPGAs used to emulate the rest of the HN system. The final HN infrastructure will contain dozens of manycore chips, and thus thousands of cores. The prototypical board will enable components to be easily plugged and removed and will allow different resource mixes, e.g. nodes highly populated with ARM cores and few high-end FPGAs (e.g. 192 + 64) or vice versa (e.g. 64 + 192), plus memory modules.

IV. Programming Model and Runtime Management

To reach exascale parallelism, the programming model needs to be hierarchical, much like the runtime management system. Traditionally, the programming model for homogeneous HPC systems is based on a combination of MPI and OpenMP. When heterogeneity comes into the game, the programming model needs to be extended to allow the exploitation of hardware resources. OpenCL is an open standard for the development of parallel applications on a variety of heterogeneous multi-core architectures [12]. It provides explicit management of heterogeneity, but at a significant cost in terms of tuning performance, which must be performed by the programmer, and building boilerplate code [13], [14]. In MANGO, we aim at integrating the expression of new architectural features as well as QoS concerns and parameters within the existing stack of languages and libraries for extreme-scale HPC systems, by augmenting the runtime library APIs with new functions, as well as by introducing new pragmas or keywords to the language.

a) Resource Management: The main challenge for heterogeneous resource management is to optimize resource allocation while taking into account that: 1) each application may be composed by multiple tasks, each of them possibly having data and timing dependencies with the other ones; 2) executing a task on different computing units of an heterogeneous architecture would lead to different throughput, QoS, and power/energy consumption; 3) especially in case of data dependencies, the performance of an application depends not only on where its tasks are executed, but also on where the data of its task is located in the system; 4) requirements coming from each application (usually throughput and QoS) must be complied with, while also addressing the system-wide (power/thermal/energy) requirements. To address this problem, we decouple the description of a task graph, which encodes the work to be done and its QoS requirements, from the decisions that must be taken to optimally allocate tasks and data. The former is addressed by the application developer through an appropriate programming model, while the latter is handled by a resource manager which has a system-wide view of the available resources and workload, as shown in Figure 6.

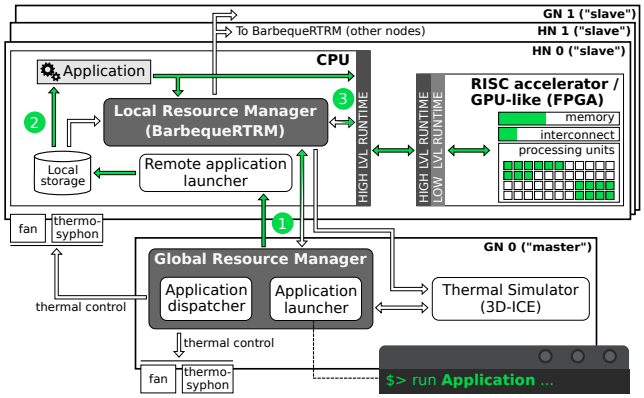

Fig. 6. The MANGO Runtime Resource Management Approach

b) Memory Management: The MANGO architecture is based on a shared memory among all the heterogeneous units in a node. To efficiently manage the available memory resources, we design a memory manager that serves memory requests in a resource allocation-aware fashion, employing knowledge about the evolution of the workload to maximize the utilization of resources while optimizing the ability of the node to serve high priority applications. We focus on the need to balance the needs of the application currently requesting resources with those of future requests, in sight of the presence of high priority applications. To this end, we build predictive models of the requests, and adapt the memory allocation in order to leave enough memory resources for the execution of new tasks on units that are currently free. Since a typical use of MANGO hardware is to allow the consolidation of applications from a small set of application domains, currently deployed on local servers to a remote HPC cluster, this prediction approach can prove quite effective.

In Figure 7, we show the outcome of an initial experiment, run in simulation, where we perform the allocation of 60,000 kernels including a mix of periodic and aperiodic requests. Experiment 1 and 2 differ in memory size – in experiment 1, memory is twice as much as in experiment 2. Green bars represent the percentage of successful requests on high-priority requests, whereas blue bars consider the overall set of requests. Three algorithms are considered: a baseline allocator with no prediction (solid bar), and two predictors employing a moving average method (thin stripes) and an exponential weighted average method (thick stripes).

In can be seen that the moving average method provides a good advantage, especially in experiment 1, where the workload is lighter with respect to the available resources.

c) Programming Model Support: MANGO aims at supporting parallel programming models across a wide range of different accelerators. We adopt an intermediate runtime layer that exposes basic features which easily map on the hardware features common to all the accelerators (i.e., those provided by the communication architecture). The intermediate runtime support exposes basic tools for communication, synchronization and task spawning, as shown in Figure 8. For brevity, only the `Context`, `Event`, `Kernel` and `Buffer` classes are shown. `Context` provides facilities to access the resource manager, registering instances of the other three classes. `Event`, `Kernel` and `Buffer` are the objects which need to be allocated resources (either memory or execution units). The final main

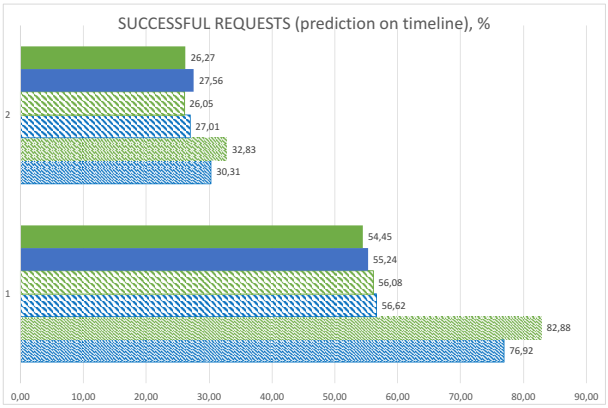

Fig. 7. Initial experiments on predictive algorithm for global memory management.

class, the `TaskGraph`, represents a subset of the objects registered with the `Context`, which is used in a specific run.

Higher level models then build over the intermediate model. The programming model exposes the managed device in a transparent way, allowing the application programmer to define one or more version of each kernel, and delegating to the runtime manager the selection of the actual execution unit employed to run it.

The following listing exemplifies the use of the MANGO API to run a simple kernel on an accelerator.

```
using namespace mango;

class KernelRunner {
 private:
  BBQContext *mango_rt;
  KernelArguments *argsKSCALE;
  KernelArguments *argsKSMOOTH;
  TaskGraph *tg;
  enum { HOST=0, KSCALE=1, KSMOOTH };
  enum { B1=1, B2, B3 };

 public:
  KernelRunner(int SX, int SY){
   // Initialization
   mango_rt = new BBQContext();
   auto kf_scale = new KernelFunction();
   kf_scale->load("./scale_kernel",
    UnitType::GN,
    FileType::BINARY);
   auto kf_smooth = new KernelFunction();
   kf_smooth->load("./smooth_kernel",
    UnitType::GN,
    FileType::BINARY);

   // Registration of task graph
   auto kscale = mango_rt->register_kernel(
     KSCALE, *kf_scale, {B1}, {B2});
   auto ksmooth = mango_rt->register_kernel(
     KSMOOTH,*kf_smooth, {B2}, {B3});

   auto b1 = mango_rt->register_buffer<Buffer>(
     B1, SX*SY*3*sizeof(Byte), {HOST}, {KSCALE});
   auto b3 = mango_rt->register_buffer<Buffer>(
     B3, SX*2*SY*2*3*sizeof(Byte), {KSCALE}, {HOST});

   tg = new TaskGraph({ kscale, ksmooth },
        { b1, b2, b3 });
```

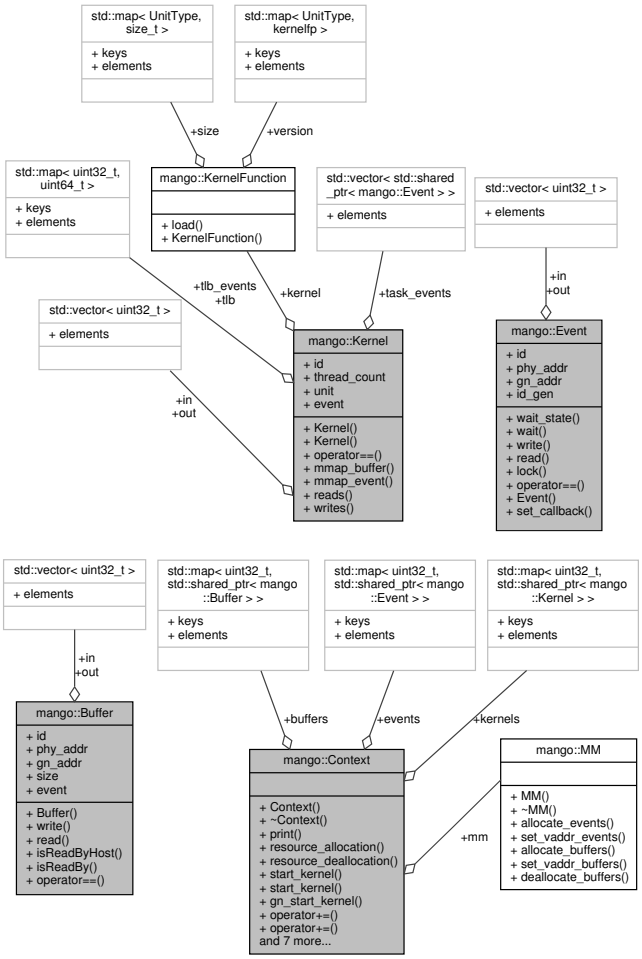

Fig. 8. Collaboration diagram for the main classes of the MANGO API.

```
   // Resource Allocation
   mango_rt->resource_allocation(*tg);

   // Execution setup
   auto argB1 = BufferArg( b1 );
   auto argB2 = BufferArg( b2 );
   auto argB3 = BufferArg( b3 );
   auto argSX = ScalarArg<int>( SX );
   auto argSY = ScalarArg<int>( SY );
   auto argSX2 = ScalarArg<int>( SX*2 );
   auto argSY2 = ScalarArg<int>( SY*2 );
   auto argE = EventArg( b3->event );

   argsKSCALE = new KernelArguments(
    { &argB2, &argB1, &argSX, &argSY },
    kscale);
   argsKSMOOTH = new KernelArguments(
    { &argB3, &argB2, &argSX2, &argSY2,
      &argE },
    ksmooth);
  }

  ~KernelRunner() {
   // Deallocation and teardown
   mango_rt->resource_deallocation(*tg);
  }

  void run_kernel(Byte *out, Byte *in) {
```

```
    auto b1 = mango_rt->buffers[B1];
    auto b3 = mango_rt->buffers[B3];
    auto kscale = mango_rt->kernels[KSCALE];
    auto ksmooth = mango_rt->kernels[KSMOOTH];

    // Data transfer and kernel execution
    b1->write(in);
    auto e1=mango_rt->start_kernel(kscale,
      *argsKSCALE);
    e1->wait();
    auto e3=mango_rt->start_kernel(ksmooth,
      *argsKSMOOTH);
    e3->wait();
    b3->event->wait();
    b3->read(out);
  }
};
```

d) Low Level Runtime Access Support: The set of accelerators in MANGO will be interconnected through a QoS-aware interconnect and spread over an infrastructure of FPGAs physically (pin-to-pin) interconnected. The system will be connected to high-end servers through PCIe and Gigabit connections. All the communication variety must be uniformly accessed by the resource manager. Indeed, the different communication interfaces must be transparent. A low level runtime library has been developed providing transparent access to the heterogeneous components. The runtime library provides efficient means for key functional processes required by the resource manager, such as: 1) booting the system and the accelerators; 2) quering about current utilization of resources and other structural information required such as power consumption or temperature; 3) enabling configuration of the system, mostly for the proper configuration of QoS parameters of the interconnect; 4) reading and writing memory distributed over the heterogeneous system; 5) spawning tasks into the accelerators; 6) providing means of synchronization between tasks and main applications running on the high-end servers.

V. Thermal and Cooling Innovations in MANGO

MANGO will extend the experience acquired in the latest research on advanced compact modelling for liquid-cooling monitoring [15] to explore the time constants of thermal and energy control knobs to develop next-generation cooling technologies for HPC systems. In particular, we will explore the use of a novel passive thermosyphon (gravity-driven) cooling technology that will attempt to include multiple *parallel heat sources at multiple elevations* to eliminate energy consumption[16]. Thus, in MANGO we will carry out for the first time in an heterogeneous HPC system a preliminary evaluation of the benefits and drawbacks of a gravity-driven two-phase liquid cooling prototype, developed and measured in the facilities of EPFL. The objective will be to proof the possibility of achieving radically low Power Usage Effectiveness (PUE) values for heterogeneous HPC systems, contributing at improving the efficiency of next-generation HPC workloads by working on the PPP axis of MANGO: power, performance and predictability.

Moreover, apart from those three metrics, thermal management is a major challenge that needs to be tackled jointly with cooling control to ensure reliability and maximize energy efficiency. Therefore, as a complimentary measure to the design of efficient cooling systems, one of the goals of the project is the development of thermal-, power- and performance-aware allocation strategies, both at the global and the local level, able to exploit the new architectures and the heterogeneity of the MANGO platform for the particular target applications. In this sense, the MANGO project will characterisation the applications, to understand their constraints, and propose novel thermal-, power-, and performance-aware run-time resource management strategies. As a first case-study, within MANGO the research undertaken has focused first on the HEVC *video transcoding application*.

1) Development of a framework for thermal, power and performance characterisation: We envision two options for the thermal, power and performance characterisation that needs to be undertaken within the MANGO project. Even though the final demonstrator infrastructure will allow automated and real-time monitoring, as will be explained in the following subsections, there is a need for off-line profiling and characterisation of the applications within the heterogeneous MANGO resources. In this sense, we have followed two approaches:

1) **The direct measurement of applications running on the hardware**. This implies running the applications on the target platform while collecting performance counters, power, and temperature values. For x86 architectures, such as the ones of GNs, we can use Intel Running Average Power Limit (RAPL) [17] to estimate the power consumption of the CPU (cores and package). Temperature sensors are usually available as an average for the whole CPU. Thus, to be able to obtain a finer granularity (i.e., to obtain the temperature gradients of the chip), we leverage the usage of the 3D-ICE simulator [18].
2) **Using a full simulation framework**. In this sense, we propose running the MANGO applications first on the Gem5 architectural simulator [19], to obtain performance metrics. Those metrics can then be plugged into McPAT [20] to obtain power traces, and finally into 3D-ICE to compute temperature floorplans.

It must be taken into account that, to profile separately the various kernels of the applications, a small effort on code instrumentation needs to be performed. This effort is needed to separate the kernels. However, this does not add an overhead to the application programmer, as the separation and characterisation is performed per-kernel.

The proposed setup is currently being used for the profiling and characterisation of the MANGO applications. In particular, characterisation of the x86 GNs is being performed via direct measurements (performance counters, RAPL and 3D-ICE), whereas currently the ARM cores are being profiled via simulation (Gem5, McPAT and 3D-ICE). We envision also the incorporation of Gem5 models of other accelerators of the MANGO platform.

2) Challenges and constraints of the video transcoding application: The undeniable complexity of the HEVC encoders, together with the increase of video streaming users, poses an important challenge for power- and thermal-aware resource allocation and management of these applications when running on MPSoCs. Because of the lack of a HEVC encoder that is able to transcode on real-time (i.e., to achieve an encoding frame rate of 30fps), current solutions in the area are mostly focused on the optimization of one or several blocks of the HEVC encoding algorithm to reduce processing time per frame [21], [22]. However, to address the challenge of power and thermal management in HEVC transcoding applications, application-level configuration and system-level knobs need to be jointly integrated on top of algorithmic optimizations. Few works jointly consider temperature constraints as well as encoding efficiency of next generation video encoders [23]. Nonetheless, none of these works consider power consumption as a different parameter from temperature. Moreover, power and thermal management of HEVC has not been addressed when multiple streams are running at the same time on a multicore platform.

Each block in HEVC encoder contains several parameters to configure the encoder (i.e., configuration knobs). A few of these

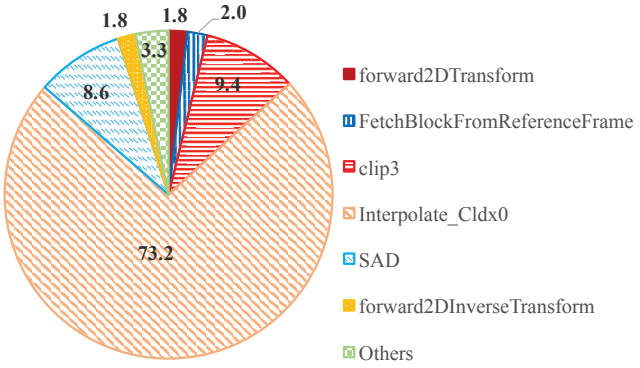

Fig. 9. Percentage of time used by the different phases of the transcoding application when encoding a frame in a GN

configuration knobs have large impacts on the encoding efficiency, power consumption, temperature and processing time, including search area, prediction mode, size of Group of Picture (GOP), Quantization Parameter (QP), and Coding Unit (CU) size. All these knobs can be dynamically tuned frame-by-frame, except for the GOP size that can only be changed every several frames.

Finally, apart from inherent and exclusive features of each video type, such as frame rate, frame resolution, bit depth, etc., the contents of a video also play a major role in the obtained performance (encoding time per frame), quality (peak signal-to-noise ratio, PSNR, measured in dB), compression (bitrate, measured in bits per second, bps), power consumption, and peak temperature, resulting from a specific encoding configuration.

The frame-by-frame power and thermal management is well motivated by such variations. As a consequence, the encoding configuration and the CPU frequency must be dynamically adjusted to provide the best possible outcomes. The great number of different combinations of configuration knobs, in addition to sudden content variations within a video and substantial differences between different videos require a more generic solution than that proposed by previous works.

Despite the sophistication of managing power and temperature in MPSoCs for HEVC, ML-based methods, and among them, reinforcement learning algorithms, are promising solutions, as they cope with environment-dependant problems using dynamic optimization programming.

3) Leveraging the MANGO HNs for the transcoding application: The first step towards exploiting the HNs to increase the efficiency of the MANGO applications is understanding which kernels would benefit the most from a hardware implementation of from acceleration. For this purpose, we have performed a profiling of the video transcoding application, obtaining its task call graph to understand which function and kernels require higher computational effort. This exercise has been performed for both the x86 and the ARM cores of MANGO, obtaining similar relative results.

Figure 9 summarises the percentage of time spent in each functions for a particular encoding configuration and video input of the transcoding application. As can be observed, the interpolation and DCT phases are the ones that would benefit the most from a hardware implementations and thus, represent good candidates for parallelization on the MANGO HNs.

4) Extending the approach to other MANGO applications: As opposed to transcoding, medical imaging has strict deadlines for the processing of each frame. When a deadline cannot be met, it is better to drop the frame. Because of the nature of this problem, we envision tackling the thermal and power-aware resource allocation problem by using Staged Multi-Armed Bandits (MABs) [24]. MABs are generally used in solving decentralized sequential decision making problems involving multiple learners. We believe that the problem of scheduling multiple streams for the MANGO bio-medical application can be seen as a staged decision problem in which the performance obtained for various resource allocations is unknown a priori but learned over time. Unlike other online learning methods such as standard multi-armed bandits and reinforcement learning, in our tentative formulation the outcome of each scheduling action depends on a sequence of previous scheduling decisions and feedbacks that are taken at a certain stage of time.

VI. The MANGO Platform Roadmap

The MANGO strategy for building an effective largescale emulation platform will be articulated in three phases.

a) Phase 1 – Stand-alone single-board emulator: The research activities involving architecture exploration initially relies on current available hardware made of a standalone emulation platform based on FPGA devices and a general purpose node. The standalone emulator is based on a modular and scalable approach, with several FPGAs being assembled on dedicated daughter modules plugged on a common motherboard. The motherboard gives complete access to all available I/Os of the FPGA, leaving maximum freedom regarding the FPGA interconnection structure, which will allow to define the HN interconnect. The proFPGA quad V7 system, provided by ProDesign as a standalone emulation platform, is used. The board is equipped with three Xilinx Virtex 7 XCV2000T FPGA modules and one Zynq module, containing a dual core ARM processor as well as reconfigurable hardware fabric to prototype external subsystems, handling up to 48 M ASIC gates alone in one board. Several proFPGA systems are interconnected enabling the full HN infrastructure to be implemented. Due to the fact that multiple proFPGA quad or duo systems can be stacked or connected together, scalability is ensured. The highspeed boards together with the specific high speed connectors allow a maximum point to point speed of up to 1.8 Gbps over the standard FPGA I/O and up to 12.5 Gbps over the MGT of the FPGA.

b) Phase 2 – From FPGA stand-alone board to a dedicated chassis: A new board for HPC will be implemented complying with the physical constraints of HPC and datacenter racks, considering as well requirements for cooling and power supply researched within the project. The board will be extended to deliver further number of daughter boards and will provide proper connectivity through optical links to other boards. Pin-to-pin connectivity between FPGAs (either at the same board or at different boards) will allow expandability and scalability. This enables MANGO to explore future chip configurations in a predictable and accurate manner. Daughter boards will be extensible and open to new developments, particularly to new 64-bit ARM cores or even more advanced solutions like the hybrid Xeon E5+FPGA chip recently announced by Intel. In this phase, the HN interconnect will be applied to the set of HN nodes (the board) developed. It will embrace connectivity at the board level, between ARM and FPGA modules, inside the FPGA modules (within the accelerators and RISC processors implemented), and between the boards. This means a single and unified interconnect will be designed for the overall HN infrastructure (made of 64 nodes).

c) Phase 3 – Rack assembly: As a final phase, the complete rack will be implemented and populated of GNs and HNs. The system

will enable a large-scale platform used to reproduce in near real-time the behavior of the MANGO manycore architecture. The full platform will consist of a rack collecting up to 16 blades equipped with high-end CPUs, e.g. Intel Xeon chips, and GPUs, mounted on the motherboard, as well as 64 HN nodes. A custom backplane will provide connectivity across the blades, both through standard bridges and using pin-to-pin connections across the FPGA chips, effectively providing a single large-scale reconfigurable hardware fabric used to emulate the fine-grained accelerator tiles envisioned in the MANGO architecture. The inter-FPGA pin-to-pin backplane interconnection will be reconfigurable on-field, providing a large degree of flexibility for the emulation of the on-chip network interconnect.

VII. Conclusions

The MANGO project, started in October 2015, will last for three years, with the goal of addressing power, performance and predictability in HPC systems. It leverages customization and deep heterogeneity to adapt the available computing resource to reach these goals. In this paper, we have presented the main approach and architectural solution, the application scenarios considered, and a more in-depth view of the software stack, including initial results on memory management and a characterisation of the HEVC transcoding application from the point of view of thermal management, as well as the roadmap towards a large scale emulation platform.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 671668.

References

[1] W. Ziegler, R. D'ippolito, M. D'Auria, J. Berends, M. Nelissen, and R. Diaz, "Implementing a "one-stop-shop" providing smes with integrated hpc simulation resources using fortissimo resources," in *eChallenges e-2014, 2014 Conference*. IEEE, 2014, pp. 1–11.

[2] J. Flich, G. Agosta, P. Ampletzer, D. A. Alonso, C. Brandolese, A. Cilardo, W. Fornaciari, Y. Hoornenborg, M. Kovać, B. Maitre, G. Massari, H. Mlinarić, E. Papastefanakis, F. Roudet, R. Tornero, and D. Zoni, "Enabling hpc for qos-sensitive applications: The mango approach," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 702–707.

[3] European Technology Platform For HPC, "ETP4HPC Strategic Research Agenda: Achieving HPC leadership in Europe," http://www.etp4hpc.eu/strategy/strategic-research-agenda/, 2013.

[4] CISCO, "The Zettabyte Era – Trends and Analysis," http://www.cisco.com, May 2015.

[5] M. Kovač, "E-Health Demystified: An E-Government Showcase," *Computer*, vol. 47, no. 10, pp. 34–42, Oct 2014.

[6] M. Kovač and N. Ranganathan, "Vlsi circuit structure for implementing jpeg image compression standard," Aug. 1997, US Patent 5,659,362. [Online]. Available: http://www.google.com/patents/US5659362

[7] J. Pawasauskas, "Volume Visualization With Ray Casting," http://web.cs.wpi.edu/ matt/courses/cs563/talks/powwie/p1/ray-cast.htm, Feb 1997.

[8] B. Preim and C. P. Botha, *Visual Computing for Medicine: Theory, Algorithms, and Applications*. Newnes, 2013.

[9] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[10] D. J. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics letters*, vol. 32, no. 18, p. 1645, 1996.

[11] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel ldpc decoder for dvb-s2 ip," in *Design, Automation & Test in Europe Conference & Exhibition, 2009. DATE'09.* IEEE, 2009, pp. 1308–1313.

[12] Khronos Group, "The Open Standard for Parallel Programming of Heterogeneous Systems," https://www.khronos.org/opencl/, (retr. Jul 2015).

[13] G. Agosta, A. Barenghi, A. Di Federico, and G. Pelosi, "OpenCL Performance Portability for General-purpose Computation on Graphics Processor Units: an Exploration on Cryptographic Primitives," *Concurrency and Computation: Practice and Experience*, 2014.

[14] G. Agosta, A. Barenghi, G. Pelosi, and M. Scandale, "Towards Transparently Tackling Functionality and Performance Issues across Different OpenCL Platforms," in *2nd Int'l Symp. on Computing and Networking (CANDAR)*, Dec 2014, pp. 130–136.

[15] A. Sridhar, A. Vincenzi, M. Ruggiero, and D. Atienza, "Neural network-based thermal simulation of integrated circuits on gpus," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 1, pp. 23–36, 2012.

[16] N. Lamaison, C. L. Ong, J. B. Marcinichen, and J. R. Thome, "Two-phase mini-thermosyphon electronics cooling: Dynamic modeling, experimental validation and application to 2u servers," *Applied Thermal Engineering*, vol. 110, pp. 481 – 494, 2017.

[17] Intel Corp., "Intel 64 and IA-32 Architectures Software Developer Manual," 2012.

[18] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, "3d-ice: Fast compact transient thermal modeling for 3d ics with inter-tier liquid cooling," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2010, pp. 463–470.

[19] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[20] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM Int. Symp. on*. IEEE, 2009, pp. 469–480.

[21] G. Correa, P. Assuncao, L. Agostini, and L. A. S. Cruz, "Complexity scalability for real-time hevc encoders," *Journal of Real-Time Image Processing*, vol. 12, no. 1, pp. 107–122, 2016.

[22] M. Shafique, M. U. K. Khan, and J. Henkel, "Power efficient and workload balanced tiling for parallelized high efficiency video coding," in *Image Processing (ICIP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1253–1257.

[23] D. Palomino, M. Shafique, A. Susin, and J. Henkel, "Tone: Adaptive temperature optimization for the next generation video encoders," in *Proceedings of the 2014 international symposium on Low power electronics and design*. ACM, 2014, pp. 33–38.

[24] K. Kanoun, C. Tekin, D. Atienza, and M. Van Der Schaar, "Big-data streaming applications scheduling based on staged multi-armed bandits," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3591–3605, 2016.