Introduction

In today’s economy one of the biggest driving trend is represented by Big Data applications. Big Data applications extract value out of a huge amount of data by searching correlation that can be used to predict new business trend, find the best medical treatment for diseases, perform financial risk analysis, determine the best location to drill for oil or gas, and so on. Usually, these applications exhibit poor temporal and spatial locality and perform simple operations on billions of data elements. As a consequence, data stream in and out the processor core at high rate and the cache hit rate is really low. This behavior puts extremely high demands on the today’s High Performance Computing (HPC) systems. This is even amplified by current technological trends that prevent the transfer of large amount of data at acceptable power dissipation costs. To cope with these problems nowadays’s HPC systems are moving toward a more data-centric approach where computing is performed near to where the data reside making use of specialized or general-purpose accelerators. However, near-memory computing it is still in its infancy and a lot of challenges need to be addressed before it can be established as an essential component for HPC systems. Below some of the main challenges for near-memory computing are reported. Feel free to contact one of the PhDs if you have an idea related to this field or simply for more information (emails are at the end of this doc).

Application and run-time challenges:

- Application Characterization

Application characterization is critical to understand the intrinsic behaviour of Big-Data workloads in order to understand which properties a near-memory accelerator should have. The current work focuses on developing an application characterization tool based on LLVM Intermediate Representation (IR). IR would allow us to implement an hardware agnostic tool; In other words: an analysis without architecture artifacts.
• **Big-Data Applications**
  An other research goal is to identify a set of real-world applications suitable for near-memory acceleration.

• **Run-Time Management**
  Run-time mechanisms are fundamental when it is needed to reach maximum performance. Hence, researches should be carried out on run-time adaptive algorithms.

**Architectural challenges:**

• **Design Space Exploration** At the moment we are making use of analytical models to evaluate the huge design space for near-memory computing. [More info later on..]

**Compiler Infrastructure challenges:**

• **Data Mapping**
  Data mapping is an important research direction for near-data processing (NDP) systems. Consider for instance the architecture shown in Figure 1. As you can see it consists of multiple memory stacks interconnected with a main unit. Each memory stack houses smaller NDP accelerators. It is clear that if you want to maximize the benefit of NDP acceleration you need to ensure code and data co-location. For example, consider again Figure 1. If you decide to offload the addition $c = a + b$ to the top-right memory stack you need to ensure that $c$, $a$ and $b$ are indeed mapped to the top-right memory stack; If this is not the case it is likely that data movements will dwarf the benefits of executing closer to the memory. Researches need to be carried out on different data mapping strategies (i.e. static vs dynamic mapping) as well as light-weight mechanisms for data migration. A good starting point would be the following paper [1].

• **Code Offloading**
  Code offloading is another interesting direction for near-memory computing. The concept is sketched in Figure 2. Several key research questions that should be investigated for code offloading include: Which “portions” of your C/C++ application should be executed on the main core and which should be executed on the NDP accelerator? Which code metrics (i.e. high MPKI) trigger the execution near the memory? What is the best granularity for NDP execution? Should we offload an entire application, a single kernel or a single instruction? What is the cost in terms of communications between the CPU and the accelerator for each of the three aforementioned choices? How do we synchronize the code executing on the CPU and the one running on the accelerator?
• Programming model
To enable the wide spreading of NDP system it is critical to provide a well defined interface to programmers. Hence, researches are needed to study the best way on how to integrate NDP instructions in existing compiler-based mechanism or library.

Information:
For details about the project visit http://nemeco.ele.tue.nl/ or contact us:

- Stefano Corda Applications and Run-time optimizations (s.corda@tue.nl)
- Lorenzo Chelini Compiler Infrastructure (l.chelini@tue.nl)
- Gagandeep Singh Architecture (g.singh@tue.nl)