Universidade:
presente!
UFRGS
PROPESQ
XXXI SIC
CONHECIMENTO FORMAÇÃO INOVAÇÃO
Salão UFRGS 2019
21. 25. OUTUBRO • CAMPUS DO VALE

Evento	Salão UFRGS 2019: SIC - XXXI SALÃO DE INICIAÇÃO CIENTÍFICA DA UFRGS
Ano	2019
Local	Campus do Vale - UFRGS
Título	Computational Experiments on Task-Based Parallel Applications
Autor	HENRIQUE CORRÊA PEREIRA DA SILVA
Orientador	LUCAS MELLO SCHNORR

Computational Experiments on Task-Based Parallel Applications

Salão de Iniciação Científica UFRGS 2019

Henrique Corrêa Pereira da Silva and Lucas Mello Schnorr (advisor)

Informatics Institute
Universidade Federal do Rio Grande do Sul

{hcpsilva, schnorr}@inf.ufrgs.br

June 20, 2019

The incessant pursuit of ever-more performance in all kinds of computing systems led software developers inevitably into the development of parallel applications. These applications make use of, traditionally, multiple *homogeneous* hardware, be it multiple *CPU* cores or *GPU* boards. However, these implementations often lack the ability to conform to less homogeneous realities, such as systems comprising of different GPUs and CPUs joined in a *computer cluster*. Given this demand, there has been considerable effort into the development of *middleware* capable of effectively distributing the workload into these hybrid architectures, taking advantage of the whole system and providing better performance without the need of huge investments into massive quantities of homogeneous hardware.

There are several popular parallel programming models used currently, be it on the *HPC* (High-Performance Computing) environment or not. Some of the more popular ones are **message passing**, **shared memory** and, even more common, any combination of other models, which then is called a **hybrid model**. Furthermore, the implementation of any combination of the previously cited models has been, characteristically, a very manual process, and, consequently, very error-prone, complex, and time-consuming. In order to mitigate that, most popular implementations of those models, such as *OpenMPI*, *OpenMP* and the combination of the previous, sought out to provide a friendly programming interface to the developer through better *APIs* (Application Programming Interfaces) or compiler directives. Still, a problem with those APIs is that the domain decomposition is fixed to the number of resources. So, despite the fact that these APIs can be adapted to more heterogeneous resources, by having those unequal pieces of work it becomes harder to obtain desirable performance due to dynamic load imbalances.

One of the efforts into making a middleware capable of such *heterogeneous computation* is called *StarPU*, which approaches the problem with the task-based model in mind, defining tasks into a directed acyclic graph. While the instantiation of this graph is made in a common sequential way, StarPU takes care of the execution and distribution of tasks in the hardware, freeing the developer from most low-level issues. Besides its programmer-friendly way of defining the tasks that will carry out the intended computation, a graph is also a powerful way to declare data dependencies between said tasks, as the data declared to the API data can also be thought as a node in said graph.

Following the intent behind the efforts of StarPU, we aim to investigate the task-based model of designing and implementing parallel applications using StarPU and how StarPU can be employed to code typical applications such as linear algebra. Once these applications are coded, we intend to run them in HPC platforms to evaluate the performance through common HPC metrics, such as *makespan* (execution time), load imbalance, occupation, and others.

Thus far, we have implemented a block vector reduction utilizing the task-based model. This implementation uses StarPU's C API to achieve a multi-level reduction determined by a block size parameter, which defines the size of each block that will be reduced to a single element. That is, the program splits the initial vectors in sub-vectors of size determined by parameter and sums the elements in these sub-vectors into single elements, which are then the new vector in which this split will be reapplied until we have a single element representing the sum of all elements. Thus, through the implementation of this application, it was shown that the StarPU runtime is a viable way to apply the task-based parallel computing method.