Using Data-Flow Analysis in MAS for Power-Aware HPC Runs

Sébastien Varrette*, Grégoire Danoy*, Mateusz Guzek † and Pascal Bouvry*
* Computer Science and Communications (CSC) Research Unit
† Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg, 16, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
Emails: {Firstname.Name@uni.lu}

EXTENDED ABSTRACT

***Keywords*—Multi-Agent System (MAS); DataFlow Graph (DFG); Energy Efficiency**

With a growing concern on the considerable energy consumed by HPC platforms and data centers, research efforts are targeting toward green approaches with higher energy efficiency. Hence, the use of low power processors coming from the mobile market (such as ARM or Intel Atom) gains more and more interest [6]. In parallel, novel software approaches are mandatory to effectively use such cutting-edge technologies and dynamically adapt to the execution being performed. This remains one of the biggest challenges to make HPC applications able to take advantage of Exascale platforms once they will be available. In this context, the authors propose a dynamic and flexible scheme based on a Multi-Agent System (MAS) to handle parallel or distributed execution in an HPC environment. The proposed approach uses a portable representation for the distributed execution E of a parallel program on a fixed input: a bipartite Direct Acyclic Graph (DAG) $G = (\mathcal{V}, \mathcal{E})$ known as a *macro DataFlow Graph (DFG)*. The first class of vertices is associated to the tasks (in the sequential scheduling sense) whereas the second one represents the parameters of the tasks (either inputs or outputs according to the direction of the edge). The total number of tasks T_j in G is denoted by $|G| = n$. A DFG example is proposed in Figure 1. In the following, we will adopt the notation and assumptions of [7]. In particular, $G^{>}(T)$ denotes the sub-graph induced by all successors of a task $T \in G$ and $G^{\geq}(T) = G^{>}(T) \cup \{T\}$. Tasks in G and therefore E are computed on a distributed computing platform, typically an HPC system. Modelling an execution by a data-flow graph is part of many parallel programming languages and some efficient execution engines such as KAAPI [3] or Cilk [2] use the graph G to schedule and execute programs on distributed architectures.

In this work we propose to use a MAS organizational model, named ParaMoise [4], to model the HPC environment structure and functioning. ParaMoise, has been

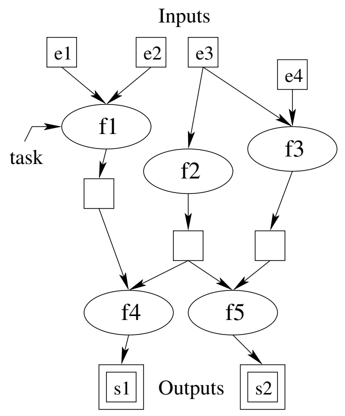

Figure 1. Instance of a data-flow graph associated to the execution of five tasks $\{f_1, ..., f_5\}$. The input parameters of the program are $\{e_1, ..., e_4\}$ whereas the outputs (i.e the results of the computation) are $\{s_1, s_2\}$.

specifically designed for such large-scale distributed systems and relies on three organizational specifications: structural, functional and deontic. ParaMoise Functional Specification (FS), responsible for describing how agents achieve their goals, is based on the concept of Workflow Specification (WFS), instantiated as a workflow (WF) as presented in Figure 2. A WF is defined as $\langle \mathcal{G}, \mathcal{E}, \mathcal{M}, mo, nm, alt, fh \rangle$, where $\mathcal{G}$ is the set of global goals and $\mathcal{E}$ the set of precedence relations, which create the structure of the DAG. Since ParaMoise allows to divide the set of global goals into other subsets by introducing the concepts of *primitive* and *composed* goals, in the rest of the paper we relate WF to G. The WF also provides additional useful conceptualization for modelling the KAAPI execution, such as tracking the status of goals (KAAPI tasks) or defining an explicit representation of roles played by the agents (KAAPI threads). As proposed in the Cloud Computing scenario of ParaMoise usage [4], each functional element in a data center can be controlled by and agent. Thus, we assume that each computing resource runs an agent which is responsible of carrying

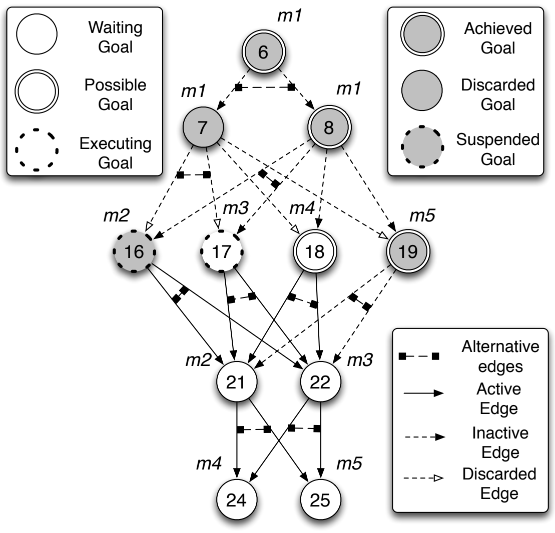

Figure 2. An example of WF during execution [4].

the local computations on the associated resource in a fully distributed manner. In particular, upon reception of a given task T to execute, the agent builds dynamically the corresponding DFG. Assuming it would have to perform locally the full execution of this task, it mean that $G^{\geq}(T)$ need to be scheduled and run locally. Yet in general, the tasks in $G^{\geq}(T)$ are scheduled using a work-stealing algorithm [1]. Generally, a work-stealing operation is purely random to ensure good performances and a reasonable good load balancing. Here, we extend this approach as a definition of a specific ParaMoise execution design and implementation. Moreover, the algorithm of dynamic building of DFG presents a way of building WF in a MAS. In this work, a work stealing operation *i.e.* the decision taken by an agent to delegate the execution of part or all of the subgraph $G^{\geq}(T)$ is driven by the objective to dynamically react to the load and optimize the global energy efficiency of the run. More precisely, the DFG representation allows each agent to evaluate and anticipate the incoming tasks to be executed. Coupled with a holistic modelling of the HPC platform introduced in a previous article [5] and adapted to the latest hardware advances, work-stealing operations and local decisions as regards the power state of the computing resources can be taken by the agents in order to dynamically optimize the power-efficiency of the execution, if possible without degrading the computing performance.

Holistic Model for the Power Measure of HPC Platforms

The holistic modelling of a virtualized High Performance Computing (HPC) data centre from [5] refers to a model which includes to the biggest extent all of the important factors that impact the performance and energy consumption of such a HPC facility. In this previous work, a lightweight modelling based on multiple linear regression concept was presented. We here extend the model with cutting edge equipment modelling (ARM etc.) and better measures of the platform performance.

This holistic model is based on the three main layers [5]: (1) the *Machine* layer describes the physical characteristics of the host. Modelling that layer assumes the derivation of the energy model of a machine which is based on the measured utilization of its components or environmental factors and taking into account machines heterogeneity. (2) The middle *Configuration* layer corresponds to the overhead induced by the software that is used to process tasks. The basic element of this layer is the *Container* that represents the used OS, including a potential virtualization technology. In case of virtualized systems there can exist multiple, possibly heterogeneous, containers on a single machine. (3) The top *Task* layer represents the computation or work performed by applications. A *Task* represents a workload processed by an application and its corresponding data. Multiple tasks can be executed simultaneously in a single Container, with the performance depending on the availability of resources.
An interesting contribution of the model is the extension of the classical definition of resources. Instead of representing resources as discrete entities, the holistic model represents each resource by a resource vector, further decomposed into resource supplies. Resource supplies are representation of hardware components installed in the machine. Each resource supply is described by its capacity and architecture. The capacity corresponds to the measurable performance or volume of the component. The architecture has the impact on the power of the node and it can be also used as a criterion for Task allocation. The resource allocation in a holistic model is represented by resource provisions and resource demands. A *Resource provision* is the representation of the resource offered by a lower layer to the higher layer. Resource providers are the resources of machines and the resources offered by containers to tasks. The *Resource demand* is the representation of the resources consumed by higher layer entities and it corresponds to the resources reserved by a container on a machine, or the resources requested by a task from a container.

Experimental Results

Our scheme has been implemented using Kaapi [3], a C++ middleware library to build with a low overhead the DFG (unfold at runtime) and schedule the tasks it is composed of. The proposed power-aware scheme has been validated on two typical applications (Fibonacci and N-Queens). Version 2.4 of the Kaapi library has been used. Experiments were conducted on the HPC platforms of the **(!** ((**!**)UL) and demonstrated promising results,

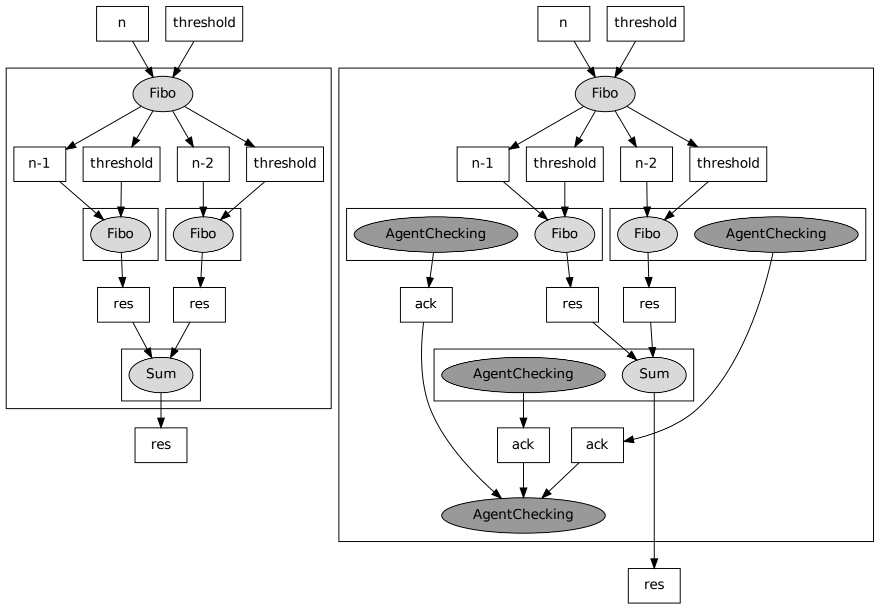

Figure 3. DFG without (left) and with (right) the `AgentCheking` function responsible for coordinating the agents during the run.

competitive with the ones pre-computed offline by classical scheduling heuristics. As an illustration, the first set of experiments of the folk recursive Fibonacci number computation has been executed to lead to the formal graph exhibited in the Figure 3. This benchmark program demonstrates a configuration with massive task creation, which is the worst configuration for our scheme as every new task created will have to be concretely executed The granularity of the program is fully controlled by the `threshold` parameter: a small value increases drastically the number of forked tasks, letting the sequential "leaf" functions of the data-flow graph (*i.e.* `Fibosec` tasks) with little work to operate. On the contrary, bigger values for the threshold limits the number of spawned tasks and makes the sequential functions longer, *i.e.* able to cover the task creation process or, in our case, the agent checking operation.

Conclusion

In this paper we presented an implementation of ParaMoise organizational model that successfully achieves energy saving in an HPC system. The organizational modelling enables to explicitly reason about the impact of the used algorithm using a DFG representation of the execution unfold dynamically at runtime. This works proves the usefulness of division of a model from its exact implementation.

The future work directions include: adding more intelligent behaviours to the work-stealing agents (e.g. learning, dynamic optimization by more advanced reorganization) with the possibility to extend to other objectives than performance and energy, experimentation with algorithms that enable distributed computation other than work-stealing, using other frameworks than Kaapi, and finally designing novel frameworks tailored for the ParaMoise model for other applications and usage areas.

References

[1] M. A. Bender and M. O. Rabin. Online Scheduling of Parallel Programs on Heterogeneous Systems with Applications to Cilk. *Theory Comput. Syst.*, 35(3):289–304, 2002.

[2] M. Frigo, C. E. Leiserson, and K. H. Randall. The implementation of the Cilk-5 multithreaded language. In *ACM SIGPLAN conference on Programming language design and implementation PLDI '98*, pages 212–223. ACM, 1998.

[3] T. Gautier, X. Besseron, and L. Pigeon. KAAPI: a Thread Scheduling Runtime System for Data Flow Computations on Cluster of Multi-Processors. In *Workshop on Parallel Symbolic Computation'07 (PASCO'07)*, London, Ontario, Canada, 2007. ACM.

[4] M. Guzek, G. Danoy, and P. Bouvry. ParaMoise: Increasing Capabilities of Parallel Execution and Reorganization in an Organizational Model. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS'13*, pages 1029–1036. IFAAMAS, May 2013.

[5] M. Guzek, S. Varrette, V. Plugaru, J. E. Sanchez, and P. Bouvry. A Holistic Model of the Performance and the Energy-Efficiency of Hypervisors in an HPC Environment. In *Proc. of the Intl. Conf. on Energy Efficiency in Large Scale Distributed Systems (EE-LSDS'13)*, LNCS. Springer Verlag, Apr 2013.

[6] M. Jarus, S. Varrette, A. Oleksiak, and P. Bouvry. Performance Evaluation and Energy Efficiency of High-Density HPC Platforms Based on Intel, AMD and ARM Processors. In *Proc. of the Intl. Conf. on Energy Efficiency in Large Scale Distributed Systems (EE-LSDS'13)*, LNCS. Springer Verlag, Apr 2013.

[7] J.-L. Roch and S. Varrette. Probabilistic Certification of Divide & Conquer Algorithms on Global Computing Platforms. Application to Fault-Tolerant Exact Matrix-Vector Product. In *Proc. of the ACM Intl. Workshop on Parallel Symbolic Computation'07 (PASCO'07)*, pages 88–92, London, Ontario, Canada, July 27–28 2007. ACM.