Challenges for Operating Systems arising from Manycore Architectures and their Solutions in MyThOS

Vladimir Nikolov   Lutz Schubert
Stefan Bonfert

Inst. of Inform. Resource Management [*]
Ulm University
D-89069 Ulm, Germany

December 2016

Keywords: Operating System, HPC, MyThOS, Manycore

Abstract

While heading towards exascale computing by resource replication the coordination of consumers, i.e. throughput oriented processes and applications, becomes even more challenging for the involved operating and runtime systems. Highly elastic and parallel HPC applications require clever strategical decisions for the decomposition and placement of computations and particular functions, an almost non-disruptive task and process coordination, fast allocation and clean-up of ressources, and dynamic reconfiguration and adaptation mechanisms of the management and computing infrastructure. It turns out that modern operating systems and runtime environments only partly fulfill these requirements, which leads to suboptimal resource utilization and throughput and hinders the overall performance gain by effective parallelization.

This talk will summarize experiences and solutions which were gathered and developed during the course of the MyThOS project [1]. The project was funded by the German Federal Ministry of Education and Research (BMBF) since 2013. It aimed at the development of a new minimal and scalable operating system for throughput oriented and highly parallel HPC applications, such as molecular and fluid dynamics simulations. The focus was set on manycore architectures such as the Intel Xeon Phi™ Co-Processor which in its Knights Corner version consist of 60 cache-coherent cores

[*]This research has been supported by BMBF under Grant No.01/H13003 (MyThOS) and EU FP7-ICT-2013.3.4 Project ref:610686 (POLCA)

with 4 hardware threads per each core. Severeal issues were identified with respect to the runtime coordination of processes and tasks on such architectures, as it is realized by modern operating systems. Those can be generally categorized as follows:

1. **Heavy Process & Thread Environments:** with a lot of kernel module dependencies, leading to long instantiation and activation times.

2. **Runtime Overhead of the OS:** with respect to in-kernel synchronization, task coordination, inter-process communication and resource allocation and clean-up.

3. **Time Multiplexing:** leading to suboptimal task preemptions, cache pollution and costly TLB invalidations.

In this talk we will descend into those particular categories and the respective issues, giving a brief overview of their solutions in already existing OSs and runtime environments. Some of these solutions are more or less convenient for the HPC application domain, while others are obstructive. However, the different approaches generally motivate different kernel designs (e.g. microkernel, multikernel, hybrid-kernel, etc.) and processing models, which will be compared from the perspective of highly parallel and elastic HPC applications. On each level, we will explicitly refer to the particular solutions for which we decided and that were integrated into MyThOS.

In summary, MyThOS implements a highly configurable hybrid of a micro- and a multikernel, which can be dynamically adapted to differend load and resource utilization scenarios. First of all, a shared base kernel with basic functionality for memory management, IPC, interrupt handling and task scheduling is mapped into the address space of each process and thread. Furthermore, applications can dynamically extend their view of the kernel by placing own performance critical objects into a preconfigured "untyped" kernel space. This offers a new palette of online decisions and dynamic reconfigurations, for example of functional placement, replication, offloading, invocation delegation and dynamic load balancing. On the other side, fine-grained synchronization mechanisms on kernel object level diminish the probability of in-kernel task serialization. Thereby, object dependencies and invocation capabilities are managed via a shared resource inheritance tree, which allows for a fine-grained dependency tracking and a fast resource clean-up. Furthermore, MyThOS implements an own processing and task model, in which system functions decompose into a set of short-running asynchronous tasks. Tasks can be transmitted to particular objects and processing units and thus provide a very lightweight IPC-mechanism based on functors and continuations. It is useful for work delegation to explicit specialized locations, which improves the overall cache locality and system performance. In this talk, besides depicting concrete concepts and their realization, we will also provide first benchmarks for thread creation and activations costs, as well as for different communication patterns between kernel objects.

Having full control about the functional distribution and kernel configuration, object placement and task delegation have a crucial impact on the performance of the system. As a future work we will extend our system with a special task-based application model, which will allow the OS to better understand task-interdependencies and to predict their communication and interaction patterns. This information will be exploited by a special compiler that will automatically establish an optimal initial application and system configuration, i.e. the separation and location of objects, tasks and system functions, with respect to pecularities of the particular underlying hardware architecture. Thereby, related and frequently communicating tasks can be grouped and settled on nearby processing units and cores, e.g. in order to exploit shared caches and shorter interconnection links between processing units for lower interaction latencies. However, since actual workload depends on the processed data, load variations are possible even in HPC applications and may lead to jitter, higher computation latencies and lower throughput. For that reason, as previously explained, MyThOS has been designed to be dynamically reconfigurable and highly customizeable to different load scenarios, i.e. it provides mechanisms for object replication, function offloading and invocation forwarding allowing to dynamically balance and smooth load fluctuations.

Besides this, MyThOS provides a basis platform for further improvements and research. For example, we plan an extension of its processing and task model with real-time capabilities and scheduling mechanisms, in order to support interactive HPC scenarios. These are scenarios, where the user can iteratively intervene during the computations in order to adjust and fine-tune their results. Consequently, user interactions are bound to a timelined processing and response by the system. Potential use cases are interactive simulations and rapid prototyping scenarios that can be found in the areas of automotive solutions, aerospace and robotics. For this, we will extend the Real-Time Divisible Load Theory [3, 2] which provides a basic computational model, axioms and formal analysis methods for parallel applications in distributed clustes. We will extend these models with hierarchical real-time scheduling and vertical as well as horizontal scaling mechanisms, in order to support interleaved job execution, to minimize the scheduling overhead for tasks and to maximize the cluster utilization, while still ensuring timeliness of results. These topics will be covered in the talk more formally as a work in progress.

References

[1] MyThOS - Operating System for Massively Threaded Applications. https://manythreads.github.io/mythos/, December 2016.

[2] S. Chuprat and S. Baruah. Real-time divisible load theory: Incorporating computation costs. In *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, volume 1, pages 33–37, 2011.

[3] X. Lin, A. Mamat, Y. Lu, J. Deogun, and S. Goddard. Real-time scheduling of divisible loads in cluster computing environments. *Journal of Parallel and Distributed Computing*, 70(3), 2010.