論文内容の要旨 (Abstract)

論文題目 (Title) Exploring Lightweight User-level Threading Frameworks for Massive Fine-Grained Parallelism
(細粒度並列プログラムを効率的に実行する軽量なスレッド処理系)

氏名 (Name) 岩崎 慎太郎 (Shintaro Iwasaki)

As the growth of CPU clock speed has been no longer sustained, modern processors adopt a multicore architecture to provide more computing power to users. Since multithreading becomes essential to unleashing the power of modern multicore machines, more and more applications, libraries, and runtime systems are parallelized by threads. The key to achieving high performance on massively parallel processors is keeping all cores busy. Fine-grained multithreading that decomposes a problem into many small pieces of work is considered to be a promising approach to exploiting parallelism in irregular and complex parallel programs. A threading overhead becomes more and more important since it dictates the finest grain size of threads.

Multithreading programming with a traditional OS-level thread, however, imposes significant threading overheads, leading to poor scalability of fine-grained and irregular multithreaded programs on highly parallel systems. To overcome the heavyweight nature of OS-level threads, another thread implementation has gained popularity. A stackless thread is a lightweight threading method that trims down overheads by removing context switches on fork and join operations. A stackless thread has a few hundred times lower fork-join overheads than an OS-level thread does. However, omitting context switches lacks several scheduling capabilities including suspension and intermediate termination, limiting its programmability.

A user-level thread (ULT) is an alternative threading technique that implements a context switch in user space. A ULT is positioned between the two opposite threading techniques; thanks to a lightweight user-level context switch, it is more efficient than an OS-level thread while more capable than a stackless thread. Nonetheless, because of its limited functionality than that of an OS-level thread and its higher threading overhead than that of a stackless thread, parallel unit abstractions in major parallel systems are based on either OS-level threads out of fear of missing capabilities or lightweight stackless

threads for performance. This prevailing practice imposes high threading overheads or excessively limits the functionality, making high-performance multithreading unnecessarily challenging.

This dissertation presents that a ULT can be a solid replacement of an OS-level thread and a stackless thread as a means of multithreading. To investigate a highly optimized implementation of ULTs, we design a user-level threading library that accommodate several user-level threading techniques with different trade-offs of performance and functionality. We also develop a ULT-based runtime implementation for OpenMP, which is one of the most popular high-level multithreading programming models. Our OpenMP library demonstrates that mapping lightweight ULTs to OpenMP threads and tasks remarkably enhances the performance of real-world applications without violating the OpenMP standard.

We first show the design of a highly customizable user-level threading library that offers optimization opportunities to knowledgeable developers. Our proposed framework, Argobots, is a highly optimized user-level threading library that balances generality and specialization by providing a low-level interface to define and control the runtime behavior. The default policies and parameters in Argobots are well optimized for generic use, while Argobots provides flexibility to manage memory resources and customize scheduling algorithms via several functions. Argobots exposes three different parallel execution units associated with an OS-level thread, a stackless thread, and a ULT, each of which provides different capabilities so that users can control the concurrency with necessary features.

Minimizing threading overheads is indispensable to exploiting fine-grained parallelism. We perform an in-depth analysis of performance vs. functionality trade-offs of user-level threading techniques. We identify a point during the execution of a thread that triggers a context switch as one of the highest sources of overheads. This point, which we refer to as a deviation point, disrupts an otherwise low-overhead run-to-completion execution. We conduct a comprehensive investigation of a wide spectrum of user-level threading methods with respect to how they handle deviations while covering both parent- and child-first scheduling policies.

We implement Argobots with these user-level threading techniques and evaluate their performance on various CPU architectures including Intel Skylake, Intel Xeon Phi, IBM POWER8, and 64-bit ARM. Our evaluation involves an instruction- and cache-level analysis of all methods. Our experiments present that threading methods that assume the absence of deviation and dynamically provide context-switching support at deviation points offer the best trade-off between performance and functionality when the likelihood of deviation is low.

To showcase that a ULT can be substituted for a thread abstraction in a high-level multithreading programming model, we developed BOLT, a highly optimized ULT-based OpenMP library derived from LLVM OpenMP. BOLT significantly improves performance over existing OpenMP implementations using OS-level threads thanks to lightweight threading operations. We find that such performance improvement is hardly obtained by a simple adaption of lightweight ULTs. Specifically, it is accomplished on three fronts: (1) advanced data reuse and thread synchronization strategies; (2) a novel thread coordination algorithm that adapts to the level of oversubscription; and (3) an implementation of the modern OpenMP thread-to-CPU binding interface tailored to ULT-based runtimes.

Our evaluation focuses on OpenMP nested parallel regions, which are often unintentionally introduced in real-world applications as a result of independent OpenMP parallelization in multiple software layers. Nested parallel regions have been known to cause the destructive performance with leading OpenMP runtimes because of their reliance on heavyweight OS-level threads. Our BOLT runtime system can successfully exploit such nested parallelism; our experimental results show that BOLT outperforms all existing runtimes under nested parallelism while transparently achieving similar performance compared with leading state-of-the-art OpenMP runtimes under flat parallelism.

These lightweight threading frameworks with our extensions and optimizations are publicly available. Thanks to numerous collaborators and contributors, these libraries are maintained high quality and used by several research and industrial projects. Our developed lightweight threading libraries, Argobots and BOLT, prove that a ULT is a scalable and practical tool for multithreading, which elevates the performance of massive and fine-grained parallel programs in the era of multicore and many-core processors.