論 文 の 内 容 の 要 旨

論文題目　Discovery of Memory Performance Problems Using Hardware Instruction Sampling
（命令サンプリングによるメモリ性能問題の検出手法）

氏　　名　ヘルム クリスティアン ルドルフ

In high-performance computing, many performance problems are caused by the memory system. Because such performance bugs are hard to identify, analysis tools play an important role in performance optimization. Today's processors offer feature-rich performance monitoring units (PMU). Information, which is not available through software-based techniques can be obtained, and low overhead profiling is possible. One of the features offered by the PMU is instruction sampling. It allows better attribution to code and data, and it provides more detailed information about memory accesses compared to previous hardware-based profiling methods.

The instruction sampling information is already used by some performance analysis tools. They present the data to the user for manual analysis. Some of the previous tools provide automatic discovery of performance problems. They are specialized for one specific performance problem and cannot detect other performance problems. In contrast, we combine the automatic detection of two different performance problems with manual analysis features. We show that it is a viable, low overhead approach to collect data from the whole application first and then find different potential performance problems from the recorded data.

One of the problems that we can automatically discover is DRAM contention.
We introduce a new approach based on latency measurement. This approach can benefit from the precision of instruction sampling to identify specific code locations and objects that are responsible for the DRAM contention. It can also differentiate harmless high bandwidth consumption from contention, consider the effectiveness of prefetching and measure the severity of contention.

The practical implementation of such a diagnosis system on CPUs is difficult. In modern CPUs, there is an abundance of performance counters and only superficial documentation. Different types of counters for bandwidth or latency, that seemingly measure the same thing, produce different results. There is no in-depth understanding of those performance counters, and naive usage may lead to incorrect measurements.

We compare various hardware latency and bandwidth measurement methods on CPUs by using micro-benchmarks. We show results of Intel Haswell, Broadwell, and Skylake systems. With our experiments, we show how and why performance counters for bandwidth and latency differ. Only the counters inside of the memory controller correctly measure bandwidth. Latency measured by instruction sampling is suitable to find DRAM contention, even though it consists of DRAM delays and in-core delays. Based on these experimental results, we establish our new detection method for bandwidth contention.

Another common performance problem is false sharing. False sharing is hard to detect manually because its occurrence depends on the data layout and cache line size. Despite numerous previous efforts, detecting false sharing is still difficult, and previous tools could not identify some cases of false sharing as we show in this work. Our approach can differentiate false and true sharing, and identify objects and source code lines where the accesses to falsely shared objects are happening Our approach uses information from the hardware coherency protocol to find shared data. In a second step, unintentionally shared cache lines are identified by analysis of access patterns of threads. A challenge is the exact specification of conditions, that samples must meet, for false sharing to occur. The specification must be tight enough to not cause false positives, but loose enough to require only a few samples for detection.

We implemented these detection methods in an open-source tool called PerfMemPlus. The tool design is simple, provides support for many existing and upcoming processors, and the recorded data can be easily used in future research. PerfMemPlus also has manual performance data exploration features.

We show that PerfMemPlus can automatically report performance problems across a wide range of systems and benchmarks. First, we use artificial benchmarks that generate a configurable load on the memory system and benchmarks that deliberately cause false sharing and true sharing. Second, we compare known and detected performance problems in the PARSEC and Phoenix benchmarks. Additionally, we present case studies that show how PerfMemPlus can pinpoint memory performance problems in the PARSEC benchmarks and machine learning applications. The average profiling overhead of our tool is around 5\%.