A Device Aware Thread Placement Policy for NUMA-based Linux Servers

Junhee Lee  Seungjae Jung  Bumjin Kim  Yongseob Lee  Sungyoung Park
Department of Computer Science and Engineering
Sogang University, Seoul, Republic of Korea
E-mail { leejunhee23, jung881005, bum9191, tel.05056622231 } @gmail.com, parksy@sogang.ac.kr

Abstract

It is generally reported that the latency for remote memory access in NUMA systems is several times larger than that of accessing local memory. In this environment, an efficient thread placement policy not only improves the performance of applications but reduces the contention on the bus. Unlike previous research efforts where only the memory locality is considered to decide the location of a thread, this paper proposes a device aware thread placement policy that utilizes the topology of processors, the locations of SSD devices (I/O locality), and the memory locality at the same time. This idea is based on an observation that the performance improvement resulting from running a thread on a node near the I/O device is sometimes greater than that caused by locating a thread based only on the locality of the memory access. We implemented the proposed idea on a NUMA server running Linux and showed that the proposed scheme improved the read/write performance by 5% up to 40% compared to NUMAsched and AutoNUMA.

Keywords: NUMA, Linux, Many-core, Thread placement policy, SSD

1. Introduction

A node in a NUMA system consists of a local memory and several cores, and each node is connected to the other nodes through a high-speed interconnect such as QPI (Intel QuickPath Interconnect). While all memory available in a node has the same access characteristics, access to a remote memory located in other NUMA nodes induce an additional latency. As a result, a running thread experiences a difference in performance according to the location of the thread [1]. In order to avoid unanticipated NUMA memory latencies, Linux uses NUMAsched [2] and AutoNUMA [3] to palce threads so that they can maximize the affinity between the threads and their data. From the experiments we had over the NUMA-based Linux servers, we noticed that the location of an I/O device such as SSD also affects the overall performance of I/O-bound threads, and it is beneficial for scheduler to locate them near SSD devices as much as possible. It should be noted that the performance improvement resulting from running a thread on a node near I/O device is sometimes greater than that caused by locating a thread based only on the locality of memory access. Another observation is that the SSD devices from different vendors perform differently. Some SSDs show better read performance against write performance, while other SSDs exhibit better write performance. Therefore, the I/O workload (read/write ratio) based on real read/write performance can be another factor to determine the location of a thread to maximize the performance of I/O-bound threads. This paper proposes a device aware thread placement policy that considers the locations of SSD devices (I/O locality) and real read/write performance while maximizing the affinity of threads and their data. The proposed policy is implemented in a NUMA-based Linux server and is compared to NUMAsched and AutoNUMA. The performance results based on AIM7 benchmark [4] showed that the proposed scheme outperformed two previous approaches by 5% up to 40% using two SSD devices from Intel and Samsung.

2. Device Aware Thread Placement Policy

As described in previous section, the location of a running thread is determined based on the combination of three performance factors: the affinity of a thread and its data, the proximity between an I/O-bound thread and...
SSD devices, and the performance difference of read/write operations from different vendors. For this purpose, it is necessary to know the locations of SSD devices, the read/write performance related to the SSDs used, the memory and I/O usages of running threads. For simplicity, we assumed that the locations of SSD devices and their performance parameters are provided by the system configuration and each SSD vendor. However, we are also investigating the mechanisms of automatic detection of SSD location and its performance characteristics. While the memory usage and the affinity of a thread and its data are measured by modifying the NUMA fault statistics used in Linux AutoNUMA (as shown in (1)), the I/O usage is measured by using the number of I/O (read/write) counts and the I/O locality (as shown in (2)). The I/O locality is a performance figure based on the physical distance from the I/O (SSD) device. In order to determine the locations of running threads, we define a parameter called Performance per Thread ($P_j(T_j)$), which represents the overall performance of a target thread $j$ on a node $i$ that exponentially averages the performance gain ($P(T_j^{MEM})$) of a thread $j$ by increasing memory locality and the performance gain ($P(T_j^{IO})$) by increasing I/O locality. When a thread needs to be relocated, the problem is to find a node $i$ which has maximum performance of all the threads running on the node (summation of $P_j(T_j)$).

\[ P_j^{MEM}(T_j) = \frac{\text{NUMA.Faults}_j^T}{\sum_i^N \text{NUMA.Faults}_j^T} \]  
\[ P_j^{IO}(T_j) = \frac{\text{IO.Count}_j^T}{\sum_i^N \text{IO.Count}_j^T} \times \text{IO.Locality}_j \]  

3. Evaluation

In this experiment, the Intel(R) NUMA server running Linux kernel 3.17.0 with 4 nodes (each node includes 10 cores), and two SSDs from Intel (Intel 530) and Samsung (Samsung 840 EVO 2) were used. We compared our approach with those of NUMAsched and AutoNUMA using AIM7 File Server Benchmark by increasing the number of I/O-bound threads. Figure 1 shows the results where we use Intel and Samsung SSDs, respectively. As shown in Figure 1, the performance of the entire system improved. The performance using the Intel SSD was improved from 5 to 12% and the performance using the Samsung SSD was also improved from 14 to 40%.

![Figure 1. Performance comparison using AIM 7 file sever benchmark (Intel and Samsung)](image)

4. Conclusions

We have proposed a thread placement policy based both on memory locality and I/O locality at the same time. We place threads near the I/O (SSD) device based on the performance difference caused by the physical distance between the I/O (SSD) device and the node. Through this, we improved the performance over the traditional NUMA thread placement policy.

Acknowledgments

This work was supported by the ICT R&D program of MSIP/IITP.[B0101-15-0644, Research on High Performance and Scalable Manycore Operating System]

References