**Tiresias: Optimizing NUMA Performance with CXL Memory and Locality-Aware Process Scheduling**

Wenda Tang\(^1\),\(^2\), Tianxiang Ai\(^1\) and Jie Wu\(^2\)
\(^1\)China Telecom eSurfing Cloud
\(^2\)China Telecom Cloud Computing Research Institute
\{tangwd1,aitianxiang,wujie\}@chinatelecom.cn

**ABSTRACT**
The growing demand for memory systems with larger capacities and faster data transfer speeds has driven progress in the widespread adoption of multi-socket machines and memory expansion through Compute eXpress Link (CXL). However, processes running on such multi-socket machines suffer non-uniform bandwidth and latency when accessing physical memory. Despite prior efforts to propose data allocation and placement strategies in NUMA environments over the years, they still fall short due to the semantic gap between the process scheduling and memory access pattern – the process scheduler has limited knowledge of its running processes’ memory access latency. Actually, the latency of memory access is influenced not only by the distance between NUMA nodes but also by the memory bandwidth pressure, especially in scenarios involving co-located workloads. We propose Tiresias, a feedback-based controller that migrates NUMA effects on data access latency by transparently employing memory locality-aware process scheduling and provisioning differentiated memory bandwidth allocations with assistance from CXL memory. Tiresias exploits multiple resource optimization techniques, including (1) workload-aware and software-based memory bandwidth management, (2) a memory page migration strategy to alleviate memory bandwidth contention by leveraging CXL memory, and (3) page-table self-replication (PTSR) based locality-aware process scheduling. To evaluate the impact of Tiresias on performance, we conduct an analysis that focuses on the temporal and spatial correlation of memory access patterns.

**KEYWORDS**
CXL, NUMA, TLB, page-table replication, memory tiering

---

**1 INTRODUCTION**

Multi-socket architectures, connected via cache-coherent interconnects, provide scalable memory bandwidth at high capacities and are commonly used in modern data centers and cloud deployments. Meanwhile, emerging architectures using chiplets and multi-chip modules are driving the NUMA (Non-Uniform Memory Access) paradigm: accessing memory connected to the local socket typically offers greater bandwidth and lower latency compared to accessing memory linked to a remote socket. While NUMA can offer advantages, it may also present challenges such as load imbalance, resource fragmentation, and sub-optimal resource scheduling. Furthermore, in order to mitigate the rising expenses associated with the construction and operation of cloud data centers, cloud service providers are actively exploring diverse strategies to enhance resource efficiency. For instance, using dynamic resource management methods to co-locate a greater number of application workloads into standard physical servers in order to optimize server resource utilization. However, increased workload density poses a significant challenge in multi-tenant cloud environments, leading to performance degradation issues. For example, the consequences of Quality of Service (QoS) violation can be severe in high-density cloud scenarios with “noisy neighbors”: some applications consume a disproportionately high amount of shared memory bandwidth, leading to saturation of the memory bandwidth. As a result, many applications suffer significantly higher memory access latency. Fig. 1 shows how the memory access latency (measured by Intel Memory Latency Checker) increases monotonically as the memory bandwidth pressure increases. The memory access latency first increases linearly and then increases exponentially when the memory bandwidth reaches a knee-point around 60% [23]. To mitigate the performance interference in memory subsystem, Intel introduces Resource Director Technology (RDT)
While the memory access latency of CXL may not matchwidth while provides higher memory access latency [22]. These motivate us to provide differentiated services to these(e.g., offline analytics) with high memory bandwidth resource(ArrayList the remaining memory bandwidth resources but have strict service levelobjectives (SLOs) concerning tail latency. On the other hand,BE workloads are generally throughput-focused applications(e.g., offline analytics) with high memory bandwidth resourcerequirements and less stringent latency constraints [23]. These motivate us to provide differentiated services to these two types of workloads by leveraging RDT and NUMA-awarestrategies, where we aim to meet the SLOs of LC workloadsby ensuring memory access performance and to maximize thethroughput of BE workloads by efficiently allocating theremaining memory bandwidth resources. Additionally,Compute eXpress Link (CXL) is gaining recognition as agroundbreaking technology that enhances memory bandwidth while provides higher memory access latency [22]. While the memory access latency of CXL may not matchthat of local NUMA, its supplementary memory bandwidthserves as an valuable resource for BE workloads.

In this paper, we propose Tiresias\textsuperscript{1}, a feedback-basedcontroller that optimize NUMA performance by transparentlyemploying memory locality-aware process scheduling andprovisioning differentiated memory bandwidth allocations with assistance from CXL memory. In Tiresias, workloads that are user-oriented and have strict Service-Level-Agreement (SLA) targets are given more reliable guarantee of high performance memory resource allocation. On the contrary, for tasks that are not latency-sensitive (i.e., BE workloads), CXL memory resources serve as supplementary provisions, ensuring the availability of burstable resources during instances of resource scarcity.

In summary, the contributions of this work include:

- **Mitigating memory interference via CXL memory expansion.** By integrating CXL memory into the traditional NUMA platform, we enhance the memory bandwidth management capabilities, effectively mitigating the negative impacts of memory contention among co-located workloads, thereby leading to improved system performance and higher efficiency.

- **Optimizing NUMA performance via resource coscheduling.** We exploit multiple resource optimization techniques in Tiresias, including (1) workload-aware and software-based memory bandwidth management (§3.1), (2) a memory page migration strategy to alleviate memory bandwidth contention by leveraging CXL memory (§3.2), and (3) page-table self-replication (PTSR) based locality-aware process scheduling (§3.3).

- **Performance analysis.** We analyze the expected performance of Tiresias by discussing the temporal and spatial locality of memory access patterns of workloads. Besides, by utilizing both local memory and CXL memory resources simultaneously, Tiresias demonstrates noteworthy resource efficiency.

The remainder of the paper is organized as follows. §2 provides an overview of background and discusses the research motivation behind our paper. §3 describes the design details. In §4, we carry out performance analysis to showcase the effectiveness of our design. §5 presents the conclusion and future work.

## 2 BACKGROUND AND MOTIVATION

In this section, we review the NUMA performance optimization literature related to our proposal, e.g., CXL-based memory pooling/tiering and data placement strategies in NUMA systems.

### 2.1 Memory Pooling/Tiering and CXL

Memory pooling and tiering have become essential techniques for enhancing resource utilization and cost efficiency. Memory pooling allows for the aggregation of memory resources from multiple servers, enabling them to be shared

\textsuperscript{1}A revered figure in Greek mythology, known for his profound wisdom and unique ability to foresee the future.
and dynamically allocated among cloud workloads. This approach helps to mitigate memory stranding, where allocated memory remains underutilized due to the fixed configuration of physical servers [13]. Additionally, memory tiering enhances pooling by classifying memory into distinct tiers according to performance attributes like latency, bandwidth, and expenses. The upper tier typically consists of faster, more expensive memory (e.g., DRAM), while the lower tier includes slower, more cost-effective alternatives (e.g., NVM).

By intelligently placing frequently accessed data (i.e., hot data) in the faster tier and less frequently accessed data (i.e., cold data) in the slower tier, tiered memory systems can achieve a balance between performance and cost [8, 18–20]. The Compute eXpress Link (CXL) [6, 26] is a protocol that connects devices using the PCI Express (PCIe) interface as the physical layer. It enables the connection of remote byte-addressable CXL-memory to the physical address space of the host machine, presenting itself as a CPU-less NUMA node to applications [12, 13, 15]. The CXL Consortium has recently introduced CXL version 3.0 [21], which includes memory sharing features.

In contrast to memory pooling in CXL 2.0, the memory sharing functionality in version 3.0 allows the CXL switch to assign the same remote memory area to several physical addresses of host machines simultaneously. This facilitates concurrent updates within the same coherency domain [26]. Fig. 2 provides a conceptual representation of the CXL memory pool architecture.

Several research efforts have explored the design and implementation of CXL-based memory pooling [13] and CXL-based tiering solutions [15]. For instance, Li et al. [13] focuses on memory pooling using CXL, aiming to reduce DRAM costs while meeting stringent cloud performance goals. Their design builds on the insight that pooling across a manageable number of sockets is sufficient to capture most of the benefits, thus enabling small-pool designs with low access latency. Maruf et al. [15] presents TPP (Transparent Page Placement) that leverages the CXL to enable efficient page placement across different memory tiers. TPP is designed to be application-transparent and can significantly improve the performance of memory-intensive applications in production environments.

### 2.2 NUMA-Aware Data Placement

Applications tend to operate optimally when the tasks access memory located on the local NUMA node. The Automatic NUMA Balancing (ANB) approach consistently seeks to re-locate application data to the memory node that is nearest to the tasks [7]. In the process of ANB, there’s a kernel task that periodically examines a fraction of a process’s memory. By default, it inspects 256MB of pages on each memory node. If a CPU interacts with a sampled page, it results in a minor page-fault, referred to as a NUMA hint fault. Pages that are accessed by a remote CPU are migrated to the local memory node of that particular CPU, a process known as page promotion [15].

As modern servers continue to expand their memory size, surpassing the capacity of the Translation Lookaside Buffer (TLB), they experience more TLB misses. This typically happens when executing large-memory workloads or during the migration of a process or thread between NUMA nodes. A TLB miss initiates a page-table walk, a process that incurs significant overhead. This overhead is magnified if the page-table resides in remote memory, resulting in what is known as the NUMA effect induced by the page-table [2, 17]. To alleviate the impact of the page-table-induced NUMA effect, contemporary research suggests the implementation of page-table self-replication (PTSR). Fig. 3 provides a conceptual representation of PTSR and process of a TLB miss for "Data" in CXL memory. The principal concept is to create duplicates of an application’s page-tables, ensuring each NUMA node possesses an identical replica. Consequently, every page table access is consistently conducted in local memory, thereby diminishing the NUMA effect instigated by the page-table.
3 TIRESIAS DESIGN

3.1 Differentiated Memory QoS Guarantee

Due to significant differences in the impact of memory latency on performance between LC and BE workloads, it is important to prioritize meeting the memory demands of LC workloads within reasonable limits when facing memory bandwidth resource contention.

To this end, Tiresias firstly classifies black-box workloads as either LC or BE by leveraging resource utilization pattern. Then, Tiresias proceeds to offer a differentiated memory QoS guarantee for both types of workloads. At beginning, all black-box workloads are marked as LC by default since it is fine to classify a delay-insensitive workload as interactive, but not vice-versa [24]. We periodically issue non-temporal store/load instructions to sample real-time memory access latency in local NUMA. When the mean value of sampled memory access latencies exceeds a pre-determined threshold (set at 180ns in our configuration), it indicates the occurrence of memory bandwidth contention of local NUMA. To address this issue, we throttle the memory bandwidth of BE workloads, prioritizing LC workloads. However, not all processors have RDT support. Therefore, we design a software memory bandwidth control method to address the limitation. Fig. 4 shows the details. We employ a page-table based scheme to constrain the memory bandwidth utilization of BE workloads in the absence of RDT, thereby ensuring the performance of LC workloads is not compromised. As such, Tiresias can relieve the performance pressure on LC workloads, thereby optimizing the overall system performance without compromising the quality of BE workloads.

3.2 Bandwidth Expansion via CXL Memory

It is important to highlight that controlling memory bandwidth solely can significantly impact the performance of BE workloads. While recent work [23] suggests that using long-term resource isolation and short-term resource sharing for LC and BE workloads can offer differentiated QoS guarantees, it fails to fully exploit the extra memory bandwidth resource provided by CXL memory. Therefore, we want to fill this gap by dynamically managing and allocating the extra bandwidth resource, thereby enhancing the performance of both LC and BE workloads without compromising on their respective QoS requirements.

CXL memory has recently been the focus of significant research efforts as an emerging technology that provides high-speed, byte-addressable data access [3, 9, 22, 25]. We exploit CXL memory to complement memory bandwidth in Tiresias. Fig. 5 provides the workflow of our designed mechanism. Firstly, a profiling phase is performed to sample the memory access addresses of BE workloads using Intel PEBS (Precise Event Based Sampling) technology. We then invalidate corresponding page table entries for BE workloads (1). To unthrottle memory bandwidth by leveraging CXL memory, we migrate less frequently used memory pages (2) from sampled records to CXL memory during the page fault handling procedure and filter out these memory access records in subsequent rounds of PEBS sampling. Empirically, we sort the sampled records to identify the hot data by a predetermined hotness threshold. We also employ feedback control to dynamically modulate the hotness threshold through periodical real-time memory access latency measurements.

Upon identifying that the memory access latency of local NUMA falls below a pre-determined threshold (for instance, 100ns in our settings), we disable memory bandwidth throttling and page migration for BE workloads. Additionally, to enhance performance, we employ memory page promotion from CXL memory to local NUMA, which in turn minimizes memory access latency. This mechanism ensures an adaptive response to varying workload characteristics, thereby optimizing the memory resource utilization while minimizing the likelihood of potential memory access bottlenecks.

3.3 Locality-Aware Process Scheduling

While conventional PTSR solutions may decrease remote memory accesses for page tables, they still necessitate remote memory access if the data is situated in a remote NUMA location. Therefore, we propose a locality-aware process scheduling in Tiresias. When remote memory access exhibits temporal and spatial locality, we enhance efficiency of PTSR.
by substituting the replicated page tables with partial page-tables distributed across NUMA nodes. Specifically, dedicated partial page-tables are configured for each running workload across NUMA nodes. Each partial page table is constructed properly to ensure it only succeeds in translating the corresponding virtual memory addresses whose physical memory addresses are situated within its NUMA region.

The first-touch page placement policy is adopted in *Tiresias*. This policy places every page at the processor that first reads from/writes to this page after page allocation. Considering the fact that many scientific loop-parallel programs reportedly contain various data access patterns that are mutually incompatible, these programs often experience a significant number of expensive remote memory accesses [14]. Therefore, in *Tiresias*, if a thread issues a memory access request that has to retrieve data from a different NUMA node’s memory, a “cross-NUMA page fault” will take place. Subsequently, the OS will handle this page fault by rescheduling the thread to the CPU located on the specific target NUMA node. This method ensures that the thread is executed on the appropriate CPU for efficient data retrieval and processing within the designated NUMA domain.

4 PERFORMANCE ANALYSIS

In this section, we analyze the performance obtained by using *Tiresias* under CXL memory expansion. Given a workload with $M$ accesses to local NUMA and $N$ accesses to remote NUMA totally, $l_{\text{access}}$ (approximately 100ns) and $s_{\text{access}}$ (approximately 1us) represent the access latency for local and remote NUMA, respectively. The expected average access latency $E$ can be calculated by employing the linear superposition of expectations:

$$ E = \frac{l_{\text{access}} + 2\sum_{i=2}^{M+N} [(p_1 + p_2)l_{\text{access}} + 2p_3s_{\text{access}}]}{M + N} \quad (1) $$

where $p_1$, $p_2$ and $p_3$ represent probabilities of occurrence for two consecutive local access, two consecutive remote access, and other cases. Specifically, $p_1 = \binom{M+N-2}{N}/\binom{M+N}{N} = \frac{M(M-1)}{(M+N)(M+N-1)}$, $p_2 = \binom{M+N-2}{M}/\binom{M+N}{M} = \frac{M}{(M+N)(M+N-1)}$, and $p_3 = \binom{M+N-2}{N-1}/\binom{M+N}{N} = \frac{N}{(M+N)(M+N-1)}$. Then, $E = \frac{M^2 + N^2}{M + N} l_{\text{access}} + 2\frac{MN}{M + N} s_{\text{access}} / (M + N). \quad (2)$

It is worth noting that $s_{\text{access}}$ encompasses both the scheduling overhead and the actual local NUMA access latency. Recent findings [26] suggest that the memory latency for CXL memory, denoted by $r_{\text{access}}$, typically hovers around 390ns, which is significantly higher than the access latency values for local NUMA. It is important to acknowledge that the expected average memory access latency ($i.e., E$) is theoretical in nature. The actual performance is frequently impacted by the concept of temporal and spatial locality of memory access of application workloads. This principle indicates that memory accesses following a particular access are more likely to be directed towards data that has been recently accessed or data located close to previous accesses. Fig. 6 represents a kernel density estimation plot showing the pattern of memory address accesses over time in a Memcached system captures by DynamoRIO [1]. As can be seen, Memcached demonstrates notable temporal and spatial locality in memory accesses. This kind of locality plays a significant role in decreasing effective latency, potentially resulting in situations where $E \ll r_{\text{access}}$.

However, it should be noted that there is a noticeable presence of remote memory access contributing to a significant portion of the total memory bandwidth for some benchmarks (e.g., NPB suite [10]), ranging from 11% to 48% [14]. As a result, only data locality-aware CPU scheduling may lead to substantial overhead due to the challenges posed by the low temporal and spatial locality of memory accesses. Therefore, *Tiresias* periodically detects such fluctuations and switches back to conventional PTSR solution.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose *Tiresias*, a feedback-based controller that migrates NUMA effects on data access latency by transparently employing memory locality-aware process scheduling and provisioning differentiated memory bandwidth allocations with assistance from CXL memory. By employing multiple resource optimization techniques, *Tiresias* not only provides differentiated performance QoS guarantees for both LC and BE workloads but also significantly enhances NUMA system performance. Based on our performance analysis, we have also identified substantial potential performance enhancements with the observations of temporal and spatial locality of memory access. In the future, we plan to conduct experiments on real CXL hardware to evaluate the effectiveness of *Tiresias*. 

![Figure 6: Kernel density estimation plot of memory address accesses over time in a Memcached.](image)
REFERENCES


