ՀՀ ԳԱԱ ԻՆՖՈՐՄԱՏԻԿԱՅԻ ԵՎ ԱՎՏՈՄԱՏԱՑՄԱՆ ՊՐՈԲԼԵՄՆԵՐԻ ԻՆՍՏԻՏՈՒՏ

Սահակյան Հովհաննես Կամոյի

ՏՎՅԱԼՆԵՐԻ ՓՈԽԱՆՑՄԱՆ ՕՊՏԻՄԻԶԱՑԻԱ ԲԱՇԽՎԱԾ ՀԱՄԱԿԱՐԳԵՐՈՒՄ

Ե.13.04 - «Հաշվողական մեքենաների, համալիրների, համակարգերի և ցանցերի մաթեմատիկական և ծրագրային ապահովում» մասնագիտությամբ տեխնիկական գիտությունների թեկնածուի գիտական աստիճանի հայցման ատենախոսության սեղմագիր

Երևան 2025

INSTITUTE FOR INFORMATICS AND AUTOMATION PROBLEMS OF THE NAS RA

Sahakyan Hovhannes Kamo

DATA TRANSFER OPTIMISATIONS IN WORKFLOW APPLICATION

An abstract of dissertation for obtaining a Ph.D. degree in Technical Sciences on specialty 13.04 "Provision of mathematical equations and programs for computing machi-nes, complexes, networks and systems"

Yerevan 2024

Ատենախոսության թեման հաստատվել է ՀՀ ԳԱԱ Ինֆորմատիկայի և Ավտոմատացման Պրոբլեմների Ինստիտուտում

Գիտական ղեկավար՝ տեխ. գիտ. դոկտոր Հ. Ասցատրյան

Պաշտոնական ընդդիմախոսներ՝ _____

Առաջատար կազմակերպություն՝ _____

Պաշտպանությունը կայանալու է _____ մասնագիտական խորհրդում

Ատենախոսությանը կարելի է ծանոթանալ ՀՀ ԳԱԱ ԻԱՊԻ գրադարանում

Սեղմագիրն առաքված է _____ ամսաթիվ

Մասնագիտական խորհրդի գիտական քարտուղար _____

The topic of the dissertation was approved at the Institute for Informatics and Automation Problems of NAS RA

Scientific Supervisor: H. Astsatryan, D.Tech.S.

Official Opponents: _____

Leading Organization: _____

The Defense will take place _____

The Dissertation is available at the library of IIAP NAS RA.

The abstract was delivered on _____

Scientific Secretary of the Specialized Council _____

Description of the Work

Topic Relevance and Problem Definition: Modern workflow applications, ranging from web services and content delivery networks to data-processing pipelines and IoT systems, often involve multiple distributed components that must exchange and store large volumes of data. In such systems, the transfer of data between components can become a major performance bottleneck. Every extra hop over the network or to an external storage service adds latency and consumes bandwidth. This thesis explores techniques to minimize unnecessary data movement by adopting a strategy of lazy data transfers, where data is only moved or copied when absolutely necessary. Keeping data closer to its point of use (for example, caching results within the workflow or embedding information in existing messages), reduces overhead and improves responsiveness. Recent advances in system design, such as embedding key-value stores directly into application components and using in-band telemetry with eBPF (extended Berkeley Packet Filter), suggest that it is now feasible to rethink how data is handled in workflows. The following sections provide background and context for this research, outline the motivation and problem statement, and summarize the goals, contributions, and structure of the thesis.

Efficient data handling is both a practical necessity and a scientific challenge in today's computing environments. Workflow applications are growing in scale and complexity, processing ever-larger data sets and serving ever-higher request volumes. Even modest inefficiencies in how data is transferred or stored can translate into significant slowdowns or resource wastage at scale. From a scientific standpoint, the topic of lazy data transfers addresses a clear gap in current systems design: while there have been many advancements in high-performance computing and networking, we lack integrated solutions that minimize data movement and maintain performance transparency. For example, as noted earlier, modern load balancers have not yet incorporated durable in-process storage. They either use in-memory caches that vanish on reboot or rely on external caches. This gap means current systems often spend time and network bandwidth repeatedly fetching or reconstructing data after failures or across components. Embedding a KV store within the load balancer and similar components directly addresses this gap, enabling the system to persist and reuse data locally. Scientifically,

exploring this uncharted design space yields new insights into how caching, persistence, and application logic can be co-designed for better fault tolerance and speed.

Another motivation comes from the observability gap in performance tuning. As embedded databases and in-network processing become more common, it is increasingly important to understand their internal behavior. Traditional metrics (throughput, average latency) are not sufficient to identify performance anomalies like tail-latency spikes caused by background tasks (e.g. compaction in a KV store). This thesis is relevant to the research community because it demonstrates an approach to close this gap using eBPF. By turning benchmarks into tools that reveal internal operation costs, researchers and engineers can pinpoint exactly why a given system slows down under certain conditions. This kind of visibility is crucial right now. As systems scale, simply knowing that something is slow is not enough; it's important to understand the cause in order to optimize. The timing is important because eBPF technology has matured in recent years and is now stable in mainstream Linux kernels, making it a practical tool for systems research and development. The combination of modern high-performance NVMe storage, advanced KV engines, and eBPF instrumentation creates an opportunity today to solve problems that were previously very hard to tackle. Ten years ago, embedding a high-throughput, persistent KV store in a latency-critical path might have been impractical, but now it is within reach.

Thesis Aim and Problem Statement: Despite the advancements in distributed systems and data management, current workflow applications still suffer from excessive and inefficient data transfers. The core problem addressed in this thesis is that critical components in a workflow (such as load balancers, caches, or pipeline stages) often handle data in a way that causes unnecessary movement of information and lack of persistent state, leading to avoidable latency, bandwidth usage, and complexity. For example, a load balancer without local persistent storage must fetch cached content from an external store on every cache miss or after every restart, and a monitoring system that polls servers for metrics adds extra network traffic. Additionally, the lack of internal visibility in these components makes it hard to pinpoint performance bottlenecks or optimize data handling strategies. The research question can be summarized as: How can workflow application architectures be designed and implemented to minimize unnecessary data transfers while preserving or improving performance and reliability? The problem divides into

four parts: embedding a durable KV store in latency-critical services without adding bottlenecks; extracting fine-grained performance data from these stores; steering traffic in real time with minimal overhead; and uniting layer-7 features, such as caching, with layer-4 network optimizations.

Thesis Scientific Contribution: The goal of this thesis is to improve the efficiency and performance of data-intensive workflows by reducing unnecessary data movement and enhancing system introspection. In pursuit of this goal, the thesis makes several original contributions in the areas of embedded storage design, performance benchmarking, and load balancing mechanisms:

- Embedded KV-Store in Workflow Components: A novel approach to caching and state management is designed and implemented by embedding a full-featured key-value store inside a Layer-7 load balancer. This contribution demonstrates that it is feasible to integrate engines like RocksDB or LMDB directly into a high-speed request handling path. This embedded cache achieves significantly higher throughput compared to traditional file-based or in-memory caches, while also providing durability (state persistence across restarts) and fast fail-over recovery. This result indicates that workflow components can be made more self-sufficient, caching data lazily on-site and thus avoiding constant external data transfers.

- eBPF-Based Benchmarking and Observability: A benchmarking methodology enhanced with eBPF tracing (the UCSB-eBPF framework) is developed to analyze the internal behavior of embedded key-value stores under realistic workloads. This contribution provides a tool and approach that go beyond reporting aggregate performance; it can capture where time is spent inside the system with microsecond resolution. The methodology adds minimal overhead (under 1% runtime cost), making it practical to use in evaluating real systems. Applying this tool uncovers causes of latency spikes (such as compaction or I/O stalls) that were previously invisible in black-box benchmarks. This contribution is both a methodological advance for research and a practical aid for engineers to tune systems, essentially turning performance evaluation into a fine-grained diagnostic process.

- Dynamic Load Balancing with In-Band Feedback: This work introduces a dynamic load balancing mechanism for Direct Server Return networks

that utilizes in-band metric feedback via eBPF. Specifically, the solution encodes server load indicators within regular response packets (using an IP option) and leverages eBPF at the load balancer to extract this information on the fly lazily. This eliminates the need for separate monitoring messages or polling, thereby exemplifying the lazy transfer principle at the network control level. The contribution includes a Linux-based prototype and a demonstration that this approach can improve request handling capacity by up to 47% compared to a conventional approach with explicit polling lazily. Importantly, it achieves this without requiring any modifications to client or server application code, showing that the technique can be deployed incrementally in real systems. This work contributes a new practical method for load balancing that is highly responsive and efficient, adapting to workload changes in real time while minimizing overhead.

- Layer-7 DSR Handoff with Minimal Proxying: The load balancer stays in the path only long enough to inspect the request, selects a server, then transfers the connection, so data flows directly to that server. Lightweight metadata keep sessions coherent, and the system falls back to normal proxying if the transfer fails. This preserves Layer-7 insights for policy while removing the balancer from steady-state traffic and cutting latency.

Overall, these contributions address the thesis goals by providing both new system designs (embedded caching in L7 balancers, in-band feedback for load distribution) and new analytical tools (eBPF-augmented benchmarking) to support those designs. The results include not only performance improvements in prototypes, but also insights and techniques that can be applied to a variety of workflow application scenarios. All software developed (from the modified benchmarks to the load balancer prototype) has been made available as open-source, amplifying the potential impact of this work by allowing others to build on it.

Practical Importance: From a practical perspective, solving the problem of excessive data transfers stands to benefit a wide range of real-world applications. Consider large web server farms or cloud services that serve millions of users: a load balancer that can cache content internally and intelligently distribute load can reduce both latency and real service load, leading to faster response times for users and lower infrastructure costs. In such a scenario, if the load balancer crashes or restarts, an embedded

persistent cache can allow it to pick up where it left off, instead of starting cold and overwhelming real services, improving reliability and fault recovery. Another example is edge computing and IoT gateways, where bandwidth to the cloud may be limited: by keeping data (sensor readings, intermediate analytics results) locally as long as possible and only sending summarized or necessary data to the cloud, one can operate efficiently under bandwidth constraints. The techniques in this thesis, like in-situ caching and in-band metric feedback, are directly applicable to these cases. The DSR load balancing approach with in-band feedback is immediately relevant to high-traffic websites and CDNs, where bypassing the load balancer for responses (DSR) and cutting out separate monitoring traffic can significantly increase throughput lazily. Achieving a 47% increase in requests per second in a prototype scenario indicates substantial practical gains lazily. This is especially important now as services continue to demand higher performance and lower latency. Users expect real-time responsiveness, and back-end systems are becoming more heterogeneous (mixing general-purpose servers, GPUs, etc.) which leads to variable load conditions lazily. Traditional static load balancing or naive data shuffling cannot cope optimally with such variability. Thus, it is both timely and important to develop smarter, lazy data transfer mechanisms that react to real-time conditions with minimal overhead.

While Layer-4 DSR lets real services bypass the load balancer once selected, modern load balancers are also expected to perform Layer-7 tasks, parsing HTTP headers, applying security rules, logging user context, and steering requests based on content. The challenge is to keep that rich Layer-7 metadata accessible to the LB long enough for these features, then shift the data path to the server without adding a permanent proxy hop or breaking the client connection.

In summary, the relevance of this research is highlighted by a convergence of factors: a recognized gap in how current systems handle state and performance insight, the availability of new tools and technologies to fill that gap, and a pressing practical need to improve efficiency in the face of growing data and performance demands. By addressing lazy data transfers in workflow applications, this thesis contributes to making distributed systems more scalable, efficient, and robust, which is of high importance both to the academic community and industry practitioners.

Publications List: There are already four published journal articles, and three more articles are currently under review, one of which is being considered by a Scopus-indexed journal.

Thesis Structure

The second chapter surveys the existing literature and technologies relevant to lazy data transfers, including caching strategies in load balancers, key-value store architectures, performance benchmarking tools, dynamic load balancing techniques, and the use of eBPF in systems. It provides context and distinguishes this approach from prior work.

Subsection 2.1: Explains embedded key-value stores are small database libraries linked directly into the application, turning data look-ups into local function calls instead of network trips. They're ideal for latency-sensitive services, mobile or edge devices, and single-node systems that need fast reads and writes

Subsection 2.2: A detailed description of benchmarking methods is provided, including their evolution from YCSB to modern tools. Engine-specific micro-benchmarks complement general suites but, yet their instrumentation rarely extends beyond throughput and latency counters. eBPF has emerged as a lightweight alternative: vetted probe programs can time system calls, user-space functions and scheduler events with less than 1 % overhead.

Subsection 2.3: Explanation of load balancer role in distributing traffic and improving scalability and fault tolerance is described.

The third chapter details the UCSB-eBPF benchmarking extension, including instrumentation points, data collection and analysis methods, and validation of tracer overhead. The chapter then uses this enhanced benchmark to profile multiple KV store engines in various scenarios, illustrating the kind of insights gained (for instance, identifying which internal operations are limiting throughput in each configuration).

Subsection 3.1: This section describes the benchmark design to evaluate the embedded KV store approach, a prototype L7 load balancer with caching was benchmarked under realistic web workloads. The study tests different caching configurations (file-per-key, embedded KV stores, and in-memory) with varying read-to-write ratios.

Subsection 3.2: During the benchmarks, standard and scenario-specific metrics are collected, including throughput (operations per second), memory usage, and CPU usage. The results show that while the in-memory hash map delivers the highest throughput, it lacks durability and uses significantly more RAM. LevelDB consistently showed the lowest throughput and highest CPU and memory usage across most workloads, except in single-threaded read-only and read-heavy scenarios. Where memory pressure is modest, LMDB emerges as the top performer among KV stores, matching or exceeding RocksDB's throughput while maintaining a moderate CPU profile. Thus, practitioners can opt for an in-memory hash map for speed, LMDB or RocksDB for a balance of reliability and throughput, or a file-per-key model when memory usage is critical.

Subsection 3.3: This section concludes that embedding modern KV stores in L7 load balancers is beneficial. These stores improve cache durability and eliminates cold-start latency at only modest CPU cost, yielding a favorable performance-reliability trade-off. What was once dismissed as too heavy for the data plane is now a feasible path to faster fail-over and simpler stacks, and thus merits re-evaluation in every modern Layer-7 load balancer.

The fourth chapter presents the design and implementation of the embedded key-value store within a load balancer. It describes the integration of the KV engine, the modifications to the load balancer's workflow, and mechanisms for ensuring durability and efficiency are described. Challenges, such as managing additional CPU and I/O overhead, are also discussed, along with the solutions adopted to address them.

Subsection 4.1: This section explains how the benchmark is set up by combining UCSB (Unbranded Cloud Serving Benchmark) with eBPF tracing to analyze embedded key-value stores. The benchmark is executed twice for each database workload combination: first, a timing pass with minimal eBPF overhead to measure syscalls and latency, then a memory pass to gather allocation metrics. Probes are grouped into phase markers (for workload alignment), syscall tracing (for latency breakdowns), and memory telemetry (for allocation analysis). A helper thread collects snapshots every 15 seconds.

Subsection 4.2: This section presents performance results for four key-value stores across seven workloads (e.g., write-only, read-heavy, range scans). System calls were breaked down into categories (I/O, synchronization,

memory ops). This helps make decisions without benchmarking on specific hardware. WiredTiger and RocksDB use heavy read system calls, therefore their performance is highly correlated with fast file systems, specifically fast reading hardware. LevelDB executes mainly in user space; its performance is therefore CPU-bound rather than I/O-bound.

Subsection 4.3: The chapter concludes by summarizing ucsb-ebpf, a low-overhead (<1%) eBPF extension for UCSB that transforms benchmarks into diagnostic tools. All code and analysis scripts are open-sourced. The chapter shows how eBPF can turn black-box benchmarking into a causal performance analysis for embedded databases.

The fifth chapter describes the dynamic load balancing solution using in-band load metrics. It covers the design of the eBPF programs and IP-option mechanism for embedding metrics, as well as the algorithm the load balancer uses to redistribute traffic based on the feedback. Experimental evaluations compare this approach with traditional load balancing, demonstrating performance improvements and discussing scenarios where it is most beneficial.

Subsection 5.1: This subsection describes the proposed dynamic load-balancing architecture that enhances DSR (Direct Server Return) with real-time, in-band metric feedback using eBPF. It introduces a Layer 3 extension to DSR that manages routing decisions directly at the network level, allowing for efficient traffic handling without of large-scale traffic without requiring changes to application-layer protocols or server configurations. The method embeds busyness scores directly in the return packets from real servers, avoiding the need for extra out-of-band polling or separate control messages. The load balancer encodes the client's IP and port in the packet header using custom IPv4 options or IPv6 extension headers and, based on locally maintained busyness scores, routes requests to the most suitable server. The system operates in four stages: client tuple encoding, real service selection, metric triggering, and feedback propagation. The implementation uses two packet flows, illustrated in Fig. 1, Flow 1 enables standard DSR, where the load balancer forwards requests with embedded client info, and servers reply directly. Flow 2 incorporates dynamic feedback, the LB periodically requests server load metrics, which are piggybacked on return packets routed back through the LB for processing. For clarity, the diagram shows Flow 1 on RS 1 and Flow 2 on RS 2, but in practice, both flows apply to any server.

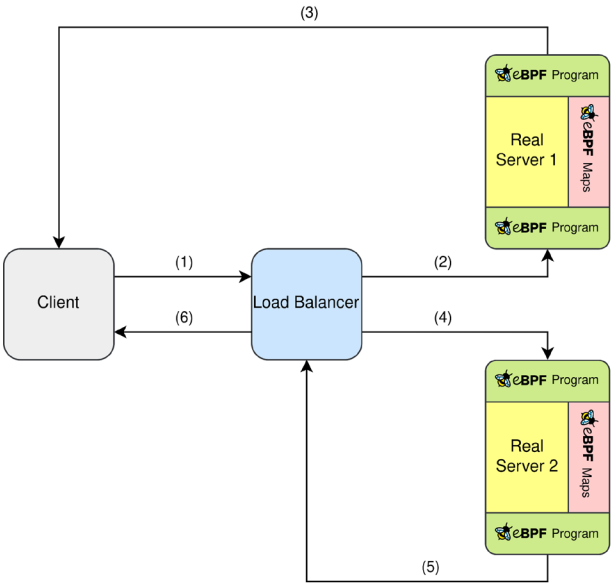

Fig. 1. Packet flow in the implementation.

By leveraging eBPF, the solution avoids modifying application logic while enabling dynamic load balancing without additional network overhead. Although it introduces overhead by adding custom IP options, it avoids additional packets. This approach is particularly suitable for dynamic, bursty workloads.

Subsection 5.2: This subsection presents benchmark results comparing four load balancing configurations: Pass Through (baseline), DSR Round-Robin, DSR with explicit busyness queries, and the proposed Dynamic DSR. All experiments were conducted using the Armenian national cloud infrastructure resource. Dynamic DSR improves average response time by 3.2 times over the baseline (Pass Through) and 20% over DSR with explicit busyness queries. It also achieves a 47% increase in RPS compared to explicit polling. While RPS significantly improves, latency percentiles (90th-99th) remain within a 5% margin between dynamic DSR and DSR with busyness requests. As shown in Figure 2, both DSR variants maintain at least 1.5× higher and

more stable RPS than the baseline. The proposed method combines these gains to deliver a 2.8× overall improvement.

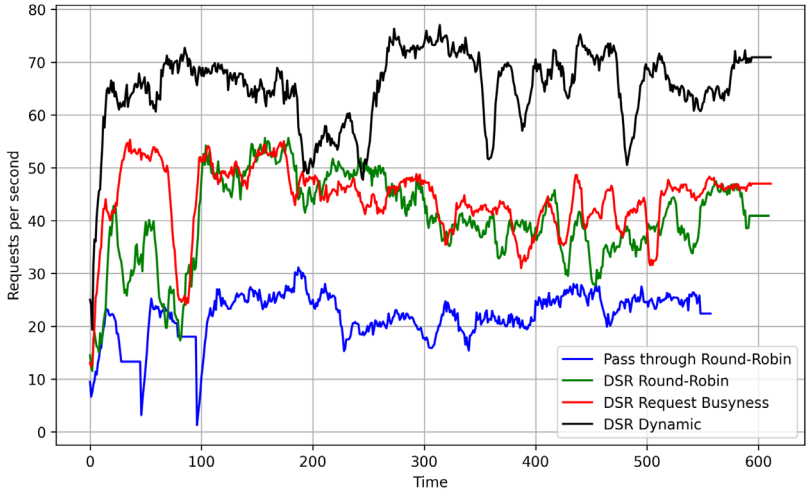

Fig. 2. Request-rate trace (10 min) for baseline and three DSR variants.

Subsection 5.3: The conclusion summarizes the benefits of the feedback-driven DSR approach, which improves scalability for unpredictable workloads. Key achievements include a 47% throughput gain and sub-100µs latency for load-aware routing. Future directions include IPv6/QUIC implementation, middlebox resilience studies, Layer-7 integration, and adaptive feedback tuning.

The sixth chapter introduces TCPE, a streamlined alternative to Multipath TCP designed for dynamic path switching and load-balancer integration. The chapter details TCPE's design and implementation, highlighting its advantages over MPTCP.

Subsection 6.1: The subsection explains why TCPE (TCP Extension) was developed as a lightweight alternative to MPTCP. While MPTCP allows a single TCP connection to use multiple paths simultaneously, it introduces complexity with cryptographic handshakes, subflow management, and middlebox compatibility issues. TCPE simplifies this by using a Connection ID

to group related flows, communicated via an experimental TCP option. This approach avoids MPTCP's kernel dependencies. The design prioritizes low overhead, minimal latency, explicit routing control, and seamless Layer-7 load balancer handoffs.

A primary design goal of TCPE is to enable on-the-fly TCP handoff from a Layer-7 load balancer (LB) to a backend server after the LB has inspected the application layer data. After completing the initial TCP handshake, the load balancer uses the Connection ID to coordinate handoff to the chosen backend. Because the Connection ID is carried in clear in a TCP option, the backend and client can recognize each other. Unlike TCPLS (which tightly integrates TLS encryption with multipath capabilities and hide such identifiers inside the encrypted channel), TCPE deliberately keeps its session metadata unencrypted to simplify coordination, the load balancer can see and use the Connection ID for routing, and network elements don't need to be aware of TLS keys or perform deep packet inspection.

TCPE is designed to gracefully fall back to standard proxying if handoff fails. The LB continues forwarding traffic at Layer 7, ensuring no connection disruption. This fail-safe approach ensures TCPE can be used in heterogeneous networks without risking connectivity. From the client's perspective, even if handoff fails, the worst-case outcome is a slight performance reduction (staying on the proxied path) rather than a broken connection.

TCPE's design rationale is to achieve some of the benefits of MPTCP in a more deployable way, specifically tailored for scenarios like layer-7 load balancing. By using a lightweight, connection identifier and deferring multipath setup until after payload inspection, it combines the intelligence of application-layer routing with the performance of direct connections. When successful, TCPE yields nearly the same efficiency as a direct client-to-server TCP connection (after handoff, the LB is out of the data path), with only a minor initial coordination overhead. And if conditions aren't right, it gracefully falls back to standard behavior, ensuring compatibility and reliability.

Subsection 6.2: This section outlines the implementation of TCPE using Linux's eBPF capabilities, which allows attaching small programs to various kernel hook points at runtime. This program can intercept TCP lifecycle events

(e.g. socket creation, connection establishment, state changes) and can also inject or parse TCP header options via special helpers.

It describes how TCPE groups TCP flows using a consistent Connection ID stored in a TCP header options. An eBPF sockops program selects a key Connection ID, or it is set from userspace. Notifications are sent via pure ACKs to avoid disrupting sequence numbers. The receiver updates its path mappings, enabling dynamic multipath management. After selecting the real server, the LB signals the server to take over. The backend can then take over the session, while the LB sends add-path for server's path, and remove-path for load balancer's path notifications to the client to switch paths. If the client migrates successfully, all subflows related to LB should be closed, and future communication uses the new path while maintaining the same Connection ID. The server processes requests and sends responses through available subflows, falling back to standard L7 load balancing if no subflows exist, while TCPE dynamically manages multiple paths through runtime additions, removals, and eBPF-monitored socket updates.

eBPF tracks connection termination (TCP_FIN/TCP_RST) and cleans up stale mappings. Additionally, the implementation uses printk logging in BPF (bpf_print) for debugging, which shows state changes. This helped verify that the logic is working, and in a production setting could be replaced by ring-buffer events to user-space if detailed tracing is needed.

While eBPF does the heavy lifting in the kernel like ID assignment and socket redirection, there is a user-space component that orchestrates the overall process. This user-space is essentially the TCPE-enabled load balancer and TCPE-aware client library. The load balancer directs traffic based on initial data and communicates with the chosen server, which runs a TCPE server component with its own eBPF setup. The server listens for client connections or instructions from the load balancer to manage traffic efficiently.

If anything in the handoff process fails, TCPE's implementation allows the LB to seamlessly continue as a proxy. A timeout or lack of response will be detected, and the LB can decide to keep handling the traffic itself. Fallback thus ensures that TCPE's enhancements never reduce reliability, they only improve performance when they succeed.

The TCPE prototype has several limitations, including the lack of automatic MTU/MSS negotiation per path, state loss and fault tolerance, no in-band security or authentication of subflows, limited multipath scheduling logic. The TCPE prototype demonstrates that Linux kernel features like eBPF can extend TCP, such that they can be used for socket migration and multipath routing without kernel modifications. By combining kernel-level packet processing with user-space control, TCPE offers a flexible, middlebox friendly alternative to new transports like QUIC, with potential for further enhancements in multipath scheduling and fault tolerance.

ՏՎՅԱԼՆԵՐԻ ՓՈԽԱՆՑՄԱՆ ՕՊՏԻՄԻԶԱՑԻԱ ԲԱՇԽՎԱԾ ՀԱՄԱԿԱՐԳԵՐՈՒՄ

Ամփոփագիր

Այս աշխատանքը ուսումնասիրվում է ժամանակակից բաշխված համակարգերում տվյալների փոխանցման արդյունավետության բարձրացման մեթոդներ: Ժամանակակից տվյալների մշակման պլատֆորմները հաճախ ներառում են մի քանի բաշխված բաղադրիչներ, որոնք մեծածավալ տվյալներ են փոխանակում և պահպանում: Տվյալները օգտագործման կետին մոտ պահելը, նվազեցնում է ծախսերը և բարելավում արձագանքողականությունը: Վերջին ժամանակաշրջանում համակարգերի նախագծման ոլորտում նորությունները, ինչպիսիք են KV պահոցների ներդրումը և eBPF-ի ներդրված չափումները օգտագործումը, հնարավորություն են տալիս վերանայելու տվյալների հետ աշխատանքը բաշխված համակարգերում:

Աշխատանքի նպատակը և դիտարկված խնդիրները

Արդյունավետ տվյալների մշակումն այսօր համարվում է և գործնական անհրաժեշտություն և մարտահրավեր: Բաշխված համակարգերը աճում են իրենց ծավալով և բարդությամբ, սկսում են մշակել ավելի մեծ տվյալների և սպասարկելով ավելի մեծ քանակի հարցումներ: Այս ատենախոսության հիմնական նպատակը տվյալների հետ աշխատանքի արտադրողականության և արդյունավետության բարձրացումն է: Դիարկվել է խնդիր, թե ինչպես նախագծել և իրագործել բաշխված համակարգեր, որպեսզի հնարավոր լինի նվազեցնել ավելորդ տվյալների փոխանցումը՝ միաժամանակ պահպանելով կամ բարձրացնելով արտադրողականությունը և հուսալիությունը: Հիմնական խնդիրներն են.

- KV պահոցների ներդրում համակարգերի բաղադրիչներում, զգալիորեն ավելի բարձր թողունակություն ապահովելու համար, միաժամանակ բարելավելով վթարային վերականգնում
- Ստեղծել միջոց մանրամասն արտադրողականության մոնիթորինգ իրականացնելու, օգտվելով նաև eBPF տեխնոլոգիայով

- Ստեղծել նոր գործնական DSR մեթոդ դինամիկ load balancing օգնությամբ, որը ապահովում է բարձր արդյունավետություն հարմարվելով աշխատանքային հոսքի փոփոխություններին:
- Ստեղծել առանց դադարի ժամանակի DSR մեխանիզմ, որը թույլ կտա միավորել DSR-ը արագությունը proxy-ների հատկյանիշների հետ, ինչպիսինն են քեշավորումը և անվտանգային զննումները:

Աշխատանքի հիմնական արդյունքները

Ատենախոսության արդյունքները ներառում են մի քանի նորարարական արդյունքներ՝

- Մշակվել և իրականացվել է ներկառուցված KV պահոցների (օրինակ՝ RocksDB կամ LMDB) համակարգերի բաղադրիչներում ինտեգրված քեշավորման նոր մոտեցում: Սա թույլ է տալիս տվյալների ավելի մոտ պահպանում: Ապահովում է զգալիորեն ավելի բարձր թողունակություն՝ համեմատած ավանդական ֆայլային կամ օպերատիվ հիշողության հիման վրա աշխատող պահոցների, միաժամանակ արագացնելով վթարային վերականգնում:
- Ստեղծվել է գործիք eBPF-ի վրա հիմնված benchmarking մեթոդաբանություն, որը թույլ է տալիս գտնել արտադրողականության կորստի պատճառները KV պահոցներում: Գործիքի միջոցով հնարավոր է ուսումնասիրել թե համակարգում որտեղ ինչքան ժամանակը է ծախսվում, ավելացնելով նվազագույն լրացուցիչ բեռ (ոչ ավելի քան 1% ժամանակ):
- Մշակվել է նոր գործնական DSR մեթոդ դինամիկ load balancing հետ համատեղելով: Նոր տեխնոլոգիան ապահովում է սերվերների բեռնվածության մասին տեղեկատվությունը փոխանցվումը պակետների IP վերնագրերում: Սա թույլ է տալիս մինչև 47% արտադրողականության աճ:
- Մշակվել է նոր բազմուղի ընդլայնում TCP պրոտոկոլի համար, որը թույլ է տալիս իրականացնել L7 DSR կապի փոխանցում առանց դադարի ժամանակի: Բալանսերը մնում է կապի մեջ այնքան ժամանակ որպեսզի ուսումնասիրի հրամանը և ընտրի սերվերը, ապա փոխանցում է կապը սերվերին: Փոքր ավելացված ինֆորմացիան օգնում է պահպանել կապի հետևողականությունը, և համակարգը վերադառնում է նորմալ proxy կապին, եթե փոխանցումը ձախողվում է: Սա պահպանում է L7 մակարդակի հատկանիշները,

միարժամանակ օպտիմիզացնելով բալանսերը պատասխանի ճանապարհից, որը նվազեցնում է փաթեթի փոխանցման վրա ծախսվող ժամանակը:

ОПТИМИЗАЦИЯ ПЕРЕДАЧИ ДАННЫХ В РАСПРЕДЕЛЁННЫХ СИСТЕМАХ

Абстракт

В данной работе исследуются методы повышения эффективности передачи данных в современных распределённых системах. Современные платформы обработки данных часто включают несколько распределённых компонентов, которые обмениваются и хранят большие объёмы данных. Хранение данных ближе к точке использования снижает затраты и улучшает отзывчивость. В последнее время нововведения в области проектирования систем, такие как внедрение KV-хранилищ (ключ-значение) и использование встроенных измерений eBPF, позволяют пересмотреть подходы к работе с данными в распределённых системах.

Цель и рассматриваемые задачи: Эффективная обработка данных сегодня является как практической необходимостью, так и вызовом. Распределённые системы растут в объёме и сложности, обрабатывая всё большие объёмы данных и обслуживая больше запросов. Основная цель данной диссертации - повышение производительности и эффективности работы с данными. Рассматривается задача проектирования и реализации распределённых систем, позволяющих сократить избыточную передачу данных, одновременно сохраняя или повышая производительность и надёжность. Основные задачи включают:

- Внедрение KV-хранилищ в компоненты систем для значительного повышения пропускной способности, а также предоставляет быстрое восстановление при сбоях
- Создание инструмента для детального мониторинга производительности с использованием технологии eBPF

- Разработка механизм динамической балансировки нагрузки для сетей с Direct Server Return (DSR), обеспечивающий высокую эффективность и адаптацию к изменениям рабочего потока.
- Разработка механизма DSR с нулевым временем простоя, позволяющего сочетать скорость DSR с особенностями прокси, такими как кэширование и проверки безопасности.

Основные результаты работы:

Результаты диссертации включают несколько инновационных решений в следующих направлениях:

- Разработан и реализован новый подход к интегрированному кэшированию с использованием встроенных KV-хранилищ (например, RocksDB или LMDB) в компонентах систем. Это позволяет хранить данные ближе к месту использования, обеспечивая значительно более высокую пропускную способность по сравнению с традиционными файловыми или оперативными хранилищами, а также ускоряя восстановление при сбоях.
- Создан инструмент для бенчмаркинга на основе eBPF, позволяющий выявлять причины потерь производительности в KV-хранилищах. Инструмент позволяет анализировать, где и сколько времени тратится в системе, добавляя минимальную дополнительную нагрузку (не более 1% времени).
- Разработан новый механизм динамической балансировки нагрузки для сетей с Direct Server Return (DSR). Эта технология передаёт информацию о загрузке серверов в IP-заголовках пакетов, что позволяет добиться повышения производительности до 47%.
- Предложено новое многопутевое расширение для протокола TCP, позволяющее реализовать передачу соединения L7 DSR без времени простоя. Балансировщик остаётся в соединении ровно настолько, чтобы, чтобы проанализировать команду и выбрать сервер, после чего передаёт соединение серверу. Небольшой объём дополнительной информации помогает сохранить согласованность соединения, а система возвращается к обычному прокси-соединению в случае сбоя передачи. Это сохраняет функциональность уровня L7, одновременно оптимизируя путь ответа балансировщика, что сокращает время передачи пакетов.