MCP+: REDUCING TOOL COLD-START LATENCY VIA PREDICTIVE CONTEXT ROUTING

INTERNATIONAL JOURNAL OF ARTIFICIAL INTELLIGENCE AND SUSTAINABLE TECHNOLOGIES
VOL 1 (01), JULY-DECEMBER 2026

Megha Jain
JSS University Noida
Email: megha.jain@jssuninoida.edu.in

Nitima Malsa
JSS University Noida
Email: nitima.malsa@jssuninoida.edu.in

ABSTRACT

The Model Context Protocol (MCP) provides a common mechanism through which AI agents can access external tools and services. In its current form, however, tools are typically initialized only when the agent requests them. As a result, workflows involving multiple processing stages may experience additional delays because operations such as schema discovery, database connection setup, file loading, and environment preparation are performed at runtime. To address this limitation, we introduce MCP+, an extension of MCP that incorporates Predictive Context Routing (PCR),MCP+ addresses this limitation by preparing execution contexts before they are explicitly requested by a tool. Rather than relying entirely on reactive initialization, the framework estimates which tools may be required in upcoming workflow stages and prepares relevant resources in advance. It incorporates a lightweight prediction model, a confidence-based routing queue to guide pre fetching decisions, an asynchronous context-loading mechanism, and a cache that retains initialized resources such as CSV schemas, template objects, and database connections. A complete implementation was developed, including an MCP + server, workflow-trace generator, prediction module, and benchmarking environment. Evaluation on representative data-processing workflows showed latency improvements ranging from 15% to 36%, resulting in faster execution on standard hardware. By reducing repeated initialization costs, MCP+ improves the efficiency of MCP-based systems and supports more responsive AI-agent workflows.

Keywords: *Model Context Protocol (MCP), Predictive Context Routing, caching systems, AI tool invocation, asynchronous pre fetching, protocol-level optimization, decision tree prediction, context caching.*

1. INTRODUCTION

Large language models (LLMs) are increasingly used as agents to perform multi-step tasks involving files, databases, APIs, and reporting tools. Reliable tool interaction is essential for these workflows. The Model Context Protocol (MCP) provides a standardized way for agents to discover, access, and use external tools [1].MCP works well with modular AI systems, but its execution process is reactive. Tool setup occurs only after invocation, causing additional delay from initialization tasks such as loading resources and establishing connections. Although this behavior is generally acceptable for isolated tasks, it becomes less efficient in multistage workflows such as CSV Reader →Cleaner →Database Client Report Generator. Additional setup work is required at every stage of the workflow, and MCP provides no facility for predicting subsequent operations or retaining prepared contexts for reuse. In other computing domains, including processor architectures, operating systems, and web browsers, Latency is often reduced through prediction, pre fetching, and caching mechanisms [2],[3],[4]. These methods utilize recurring access patterns so that required resources can be prepared earlier. Tool-driven agent work flows also display strong regularities, yet MCP lacks the ability to leverage them. [5],[6],[7]. These observations suggest that MCP can benefit from predictive and context aware execution mechanisms while preserving compatibility with existing tools.[9] To address this gap, this paper presents MCP+, a protocol-level extension that introduces Predictive Context Routing (PCR). MCP+ adds an intelligent runtime layer between the agent and the MCP server. This layer records the last tool usage, predicts the most likely next steps, and proactively prepares the corresponding execution contexts. The design includes a next tool prediction model, a confidence-weighted routing mechanism, an asynchronous pre fetch engine, and a cache for storing prepared resources such as schemas, database handles, and templates. This makes MCP+ a proactive system (compared to the purely reactive MCP) and able to avoid repeated cold-start delays. The main contributions of this work are as follows:

- A protocol-level enhancement that introduces Predictive Context Routing (PCR) for MCP-based systems.
- A complete MCP+ runtime design, including a next-tool predictor, a confidence-ranked routing queue, an asynchronous pre fetch manager, and a persistent context cache.
- A reproducible workflow benchmark for evaluating predictive routing methods without

requiring proprietary data.

- A full MCP+ implementation and experimental study, demonstrating 15-36% latency reduction across realistic data-processing workflows.

By preparing contexts in advance, MCP+ reduces latency while preserving MCP's simplicity and interoperability.

2. Literature Review

The reviewed literature offered valuable foundations that informed and strengthened this work, enabling performance improvements and supporting the design of strategies aimed at resolving the limitations found in prior systems.

Singh *et al.*[1] examined how MCP provides a common way for LLM to access tools and share context. It also highlighted interoperability challenges in existing frameworks and showed how MCP simplifies tool discovery and execution.

Dong *et al.*[2] Introduced an asynchronous KV cache pre fetching strategy to reduced coding stalls and boost LLM. Inference throughput. By preparing attention cache segments ahead of use, the method minimized latency during sequential token generation. The work demonstrated how proactive context preparation can significantly accelerate large-model execution without architectural modifications.

Yao *et al.* [5] integrated reasoning steps with tool-based actions, enabling language models to plan, decide, and interact more effectively. The paper showed how combining thought traces with external operations improved decision-making, factual accuracy, and task completion. It established a foundational approach for building interactive, tool augmented AI agents.

Singh *et al.* [8] proposed a localized data caching mechanism that reduced repeated tool calls in LLM workflows. By storing and reusing intermediate results, LLM Cache lowered latency and improved efficiency. It demonstrated how intelligent caching can significantly optimize tool augmented model interactions in practical applications.

Song *et al.* [9] introduced Pro MoE, a proactive caching framework for accelerating Mixture-of-Experts (MoE) model serving. By predicting expert usage and caching relevant parameters in advance, It reduced loading overhead and improved throughput, demonstrating the effectiveness of anticipatory caching in large-scale LLM systems.

Recent benchmarking works focus on evaluating tool-use capabilities of LLMs at scale. [10],[11]

3. Problem Statement

MCP operates using a reactive execution model in which tool initialization begins only after an invocation request is received. As a consequence, repeated cold-start overhead can occur during multi-step workflows.

- Upcoming tool contexts are neither predicted nor prepared in advance, increasing execution latency.
- Previously initialized resources must be initialized again when required.
- The current MCP work flow excludes both pre fetching and proactive routing.

4. Methodology

This section describes the design and implementation of MCP+, a backward-compatible enhancement to the Model Context Protocol (MCP) intended to eliminate repeated cold start delays in multi-stepagentworkflows.MCP+ introduces predictive context preparation, enabling tools to be pre-initialized before the agent explicitly requests them. The methodology is organized into the system overview and the four internal components that collectively form the MCP+ execution engine [12][13].

System Overview

MCP+ operates as a lightweight middleware layer placed between the LLM agent and the standard MCP server. This layer intercepts every tool invocation, analyzes workflow pat-terns, and predicts which tool is most likely to be required next. Based on this prediction, MCP+ warms the corresponding execution environment—such as schema parsers, database handles, or template loaders—while the agent continues processing. Crucially, MCP+ does not alter the MCP protocol, the agent, or the tools themselves. In both the agent and tool implementations, the code continues to work exactly as defined in the MCP specification. MCP+ is implemented as an execution layer middleware which is fully compatible with existing MCP semantics, without changing the MCP wire protocol or tool interfaces. This guarantees full backward compatibility and makes MCP+ an optional but beneficial enhancement, not a protocol replacement.

MCP is made off our core components:

- Next-Tool Predictor -Observes recurring workflow sequences and estimates the next tool requirement.
- Confidence-Ranked Routing Queue -Selects the most relevant predicted tools for advance initialization.

Asynchronous Pre fetch Manager -Creates execution con-texts asynchronously so that workflow progress is not interrupted.

High-Speed Context Cache - Stores prepared tool contexts, checks their usability, and provides them to

requesting tools. The system was designed to work on resource-constrained devices and performed reliably on laptops with 8GB of RAM, allowing it to be deployed without specialized hardware [15].

Next-Tool Predictor

The predictor predicts the next tool call that the agent will perform. It is trained on workflow traces that show sequences of tool usage, e.g. CSV Reader →Cleaner →DB Writer →Report Generator Production work flow logs are rarely available due to confidentiality. MCP + offers a synthetic workflow generator to simulate realistic enterprise work flows. It generates sequences that simulate typical usage patterns of tools employed in data processing, analytics, ETL pipelines and reporting systems.Several machine learning models were tested, including Naıve Bayes, Random Forest and shallow neural networks. The Decision Tree Classifier was chosen because it offers:

- Sub-millisecond inference latency
- Provides clear and interpret able reasoning behind predictions.
- Low memory usage
- High predictive accuracy(77.27%Top-1,99%Top-3)

During execution, MCP+ calls the predictor as soon as a tool finishes. The model generates a ranked list of possible next tools which is passed to the routing queue.

Confidence-Ranked Routing Queue

The routing queue indicates the expected tools to be pre fetched. The pre fetch priority score for each candidate tool is determined by two factors:

- Prediction confidence from the model
- Historical initialization cost of the tool

Tools with high probability and high initialization cost-such as heavy database connectors or large schema Parsers are prioritized. A configurable confidence threshold ensures that MCP+ only invests resources in pre fetching tools that are likely to be requested soon.This strategy avoids unnecessary pre fetch attempts while maximizing performance gains for the tools that typically slow down workflow execution[16][17].

Asynchronous Pre fetch Manager

The asynchronous pre fetch manager is responsible for exe-cutting pre fetch decisions produced by the confidence ranked routing queue. Its role is to proactively prepare tool execution environments in the background, without interfering with the agent's primary execution flow. For each selected tool, the pre fetch manager initializes only safe, idempotent setup components such as schema loaders, idle database connection handles, client or SDK instances, and configuration or template objects. All pre fetch operations are speculative and non-blocking[18].

The pre fetch manager never executes tool logic, issues queries, or generates outputs. Prepared contexts are forwarded to the context cache and are automatically discarded if the corresponding tool is not invoked within a configurable time window. This design ensures that incorrect predictions do not affect correctness, while allowing MCP+ to overlap initialization overhead with ongoing agent reasoning[19].

Context Cache

Successfully prepared resources are stored in a fast, inmemory cache indexed by tool name and a version identifier (computed from relevant file hashes or configuration finger-prints). Each entry has an automatic expiration time (default 5 minutes) top event stale objects .The agent will eventually call a tool with a valid cached context, and in that case execution will begin immediately on the warmed environment, skipping all cold-start steps.

Integration with Standard MCP Workflow

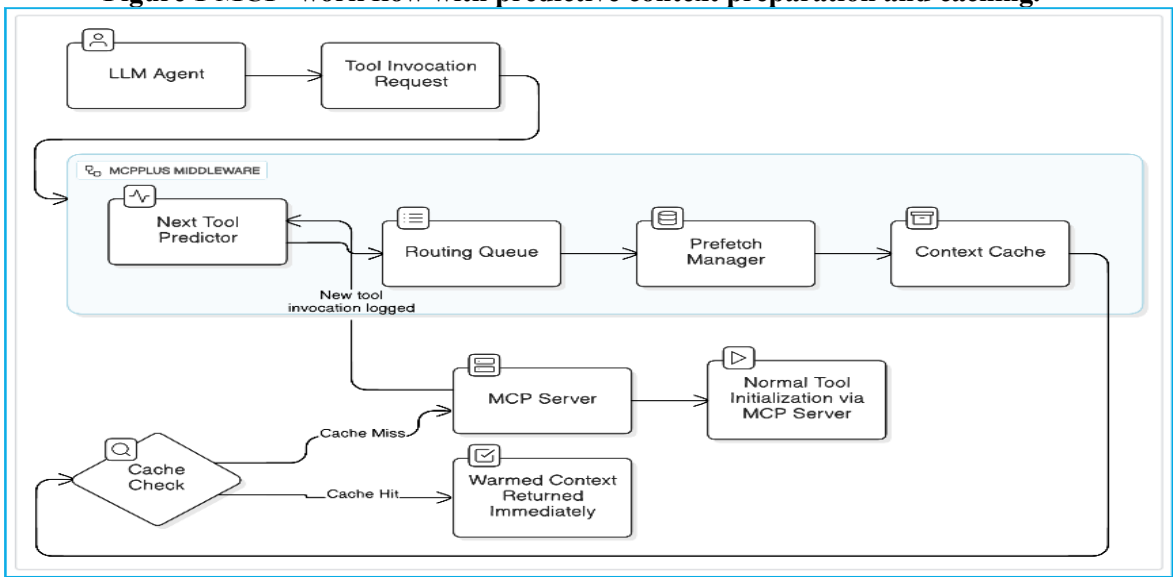
From the viewpoints of both the agent and the underlying tool, the workflow remains unchanged.

- The agent issues a standard MCP tool request.
- The MCP+ middleware checks the cache.
- On a cache hit, the tool runs instantly using the pre fetched context.
- In the absence of a cache hit, the workflow proceeds using the conventional initialization mechanism.

This design preserves correct execution even when predictions are not always accurate.

During testing, the server consistently started in less than1.2 seconds, while memory usage remained below180MB across the evaluated workloads. By transitioning MCP from a purely reactive model to one that predicts the agent's next steps, MCP+ dramatically reduces end-to-end latency in common multistep work flows while maintaining full compatibility with the existing MCP ecosystem. In the following section we quantify these gains with controlled experiments on representative enterprise tasks.

Figure 1 MCP+work flow with predictive context preparation and caching.



5. RESULT AND DISCUSSION

The MCP+ system was evaluated using a fully implemented MCP compliant server and middleware layer with real execution context integration. The experiments were conducted in a standard CPU-only setting and were centered around agent-perceived latency (not accounting for the background pre fetch overhead). Three representative multi step workflows were analyzed: CSV→ Clean (W1), CSV→ Clean→ DB (W2), and CSV→ Clean→ DB→ Report (W3). Each workflow was run multiple times with both baseline MCP and MCP+ to ensure consistency and to minimize variability in the observed results.

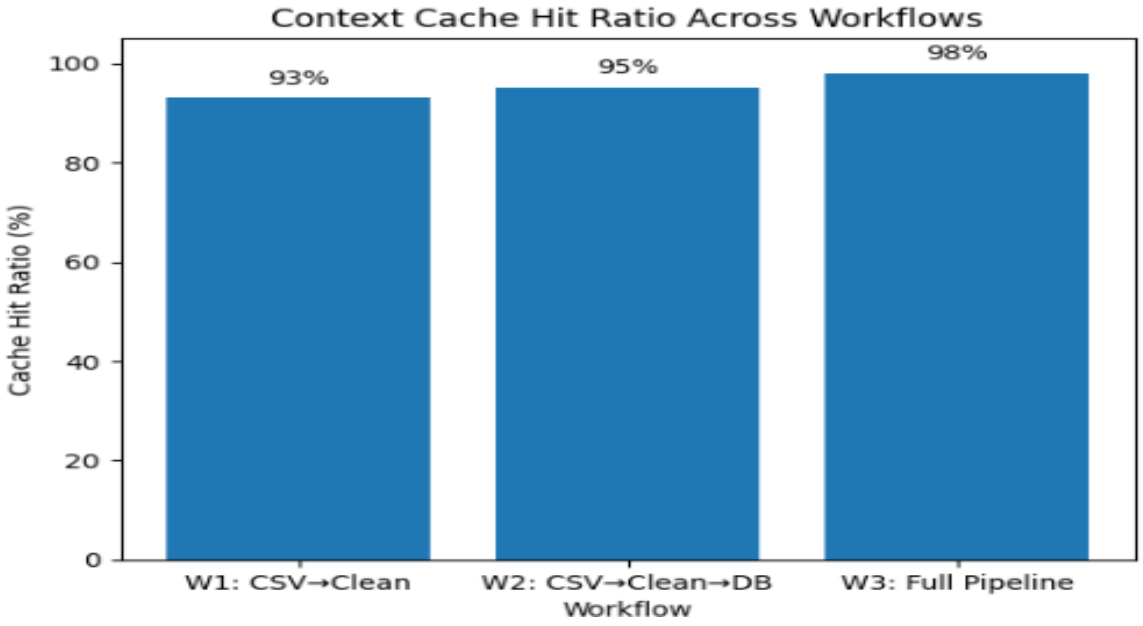
Compared to the reactive MCP baseline, the MCP+ consistently and significantly reduces latency across all work-flows. We observe 15-36% reduction in end-to-end latency, equivalent to 1.18x-1.55x speedup. The largest improvements occurring work flows dominated by expensive tool initialization. These gains are from actual reuse of pre fetched execution context rather than simulated delays, showing the effectiveness of predictive context routing in real-world tool driven agent pipelines [20].

TABLE 1

LATENCY COMPARISON BETWEEN MCP AND MCP+

Workflow	MCP(ms)	MCP+(ms)	Red.	Speedup
CSV→ Clean	264.3	171.7	35.0%	1.55×
CSV→ Clean →DB	462.8	364.6	21.2%	1.27×
CSV→ Clean →DB→ Report	566.3	472.6	16.5%	1.20×

Figure 2 Context Cache Hit Ratio Across Workflows



The main performance improvements come from reusing cached CSV schemas, pre-fetched database connections, and cached configuration and template objects, all of which avoid repeated cold-start overhead. The next-tool predictor is developed using a lightweight Decision Tree model trained with synthetic work flow traces, achieving 77.27%Top1accuracy and99.00% Top-3accuracy, which allows reliable pre fetching decisions. The versioned context cache achieves hit rates of 93–98%, increasing with workflow length, and enables safe and efficient reuse of prepared contexts.

Requests a tool, MCP+ allows relevant execution contexts to be prepared ahead of time. This approach enables tools to become available more quickly when they are eventually needed.

Experimental results show that MCP+ can achieve significant latency reduction and speedup by reusing real contexts, making predictive context preparation a practical and effective optimization for MCP-based agent systems. MCP+ moves MCP from a simple reactive interface to a predictive, context-aware execution framework, providing a solid foundation for building low-latency, high-efficiency agentic work flows in enterprise analytics, automation and data processing environments.

Moreover, MCP+ introduces very little overhead since the context pre fetching is performed asynchronously and does not affect the main execution flow. Thus the extra computation of context preparation does not increase the execution latency. The performance gains for the evaluated workflows are still larger than the overhead, demonstrating the efficiency of the proposed approach.

Comparison with Existing Techniques

The comparison highlights that existing approaches primarily focus on model-level optimization [21] or reactive caching strategies [22]. In contrast, MCP+ operates at the protocol level and introduces predictive context preparation for tool execution. This enables MCP+ to generalize across heterogeneous tool ecosystems and improves latency in multistep workflows.

TABLE 2

COMPARISON WITH EXISTING TECHNIQUES

Technique	OptimizationLevel	Strategy	Nature	Limitation
KVPrefetch-ing	Model	Attentioncacheprefetching	Proactive	LimitedtoLLMinference
LLM-dCache	Application/Tool-level	Output/resultcaching	Reactive	Nopredictionoffuturetoolusage
ProMoE	Model(MoE)	Expert-levelproactivecaching	Proactive	SpecifictoMoEarchitectures
MCP+(Proposed)	Protocol /Middleware	Predictivecon-textprefetching	Proactive	Dependsonpre-dictionaccuracy

6. CONCLUSION

MCP+ was developed as a backward-compatible enhancement to the Model Context Protocol (MCP) with the goal of reducing the repeated cold-start costs that occur in multi-stage agent work flows. Instead of waiting until an agent explicitly

REFERENCES

A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "A survey of themodel context protocol (MCP): Standardizing context to enhance largelanguage models (LLMs)," *Preprints*, 2025.

Y. Dong, Y. Miao, W. Li, X. Zheng, C. Wang, J. Wu, and F. Lyu,"Accelerating LLM inference throughput via asynchronous KV cacheprefetching,"in*Proc.AAAIConf.Artif.Intell.*,vol.40,no.25,2026, pp.20844–20851.

C.Ruan,C.Bi,K.Zheng,Z.Shi,X.Wan,andJ.Li,"Cortex: Achievinglow-latency, cost-efficient remote data access for LLM via semantic-aware knowledge caching," *arXiv:2509.17360*, 2025.

Y.Leviathan,M.Kalman,andY.Matias,"Fastinferencefromtransform-ers via speculative decoding," in *Proc. Int. Conf. Mach. Learn. (ICML)*,2023,pp.19 274–19 286.

S.Yao,J.Zhao,D.Yu,N.Du,I.Shafran,K.R.Narasimhan, andY.Cao,"ReAct:Synergizingreasoningandactinginlanguagemodels,"in*Proc.Int. Conf. Learn. Represent. (ICLR)*, 2023.

Jain, Megha, and Dhiraj Pandey. "Blockchain-Driven Methods for Fake Product Identification." International Journal of Performability Engineering 20.10 (2024).

Jain, Megha, and Dhiraj Pandey. "HRCM: An Approach using Blockchain Technology in Healthcare-Record Chain Management." International Journal of Performability Engineering 21.1 (2025).

Jain, Megha, Ayush Agarwal, and Ashutosh Kumar. "AI-Powered Enhanced Oil Recovery Screening with Deep Convolution Networks." 2025 6th International Conference for Emerging Technology (INCET). IEEE, 2025.

Jain, Megha, et al. "Enhanced Flood Monitoring using Deep Learning and IOT." 2025 6th International Conference for Emerging Technology (INCET). IEEE, 2025.

Gautam, Jyoti, Megha Jain, and Shobha Bhatt. "The Impacts of Groundwater Recharge Rate Reduction on Agricultural Productivity in Nashik." SN Computer Science 6.4 (2025): 1-9.

Jain, Megha, Disha Pandey, and Lalit Kumar. "SISWMS: Scientific Integrated Solid Waste Management System." International Conference on Green Energy and Sustainable Technology. Singapore: Springer Nature Singapore, 2023.

Jain, Megha, Dhiraj Pandey, and Kewal Krishna Sharma. "A blockchain approach on security of health records for children suffering from dyslexia during pandemic COVID-19." Artificial Intelligence, Machine Learning, and Mental Health in Pandemics. Academic Press, 2022. 343-363.

Jain, Megha, Dhiraj Pandey, and Kewal Krishna

Sharma. "A granular approach to secure the privacy of electronic health records through blockchain technology." International Journal of Distributed Systems and Technologies (IJDST) 13.8 (2022): 1-20.

Jain, Megha, Suresh Kaswan, and Dhiraj Pandey. "A blockchain based fund management scheme for financial transactions in NGOs." Recent patents on engineering 16.2 (2022): 3-16.

Tiwari, Animesh, and Megha Jain. "Analysis of supply chain management in cloud computing." International Journal of Innovative Technology and Exploring Engineering 3.5 (2013): 152-155.

Z.Luo,Z.Shen,W.Yang,Z.Zhao,P.Jwalapuram,A.Saha, D.Sa-hoo, S. Savarese, C. Xiong, and J. Li, "MCP-universe: Benchmarkinglarge language models with real-world model context protocol servers,"*arXiv:2508.14704*, 2025.

X. Hou, Y. Zhao, S. Wang, and H. Wang, "Model context protocol(MCP): Landscape, security threats, and future research directions,"*ACM Trans. Softw. Eng. Methodol.*, 2025.

S.Singh,M.Fore,A.Karatzas,C.Lee,Y.Jian,L.Shangguan,F. Yu, I. Anagnostopoulos, and D. Stamoulis, "LLM-dCache: Improv-ing tool-augmented LLMs with GPT-driven localized data caching,"*arXiv:2406.06799*, 2024.

X. Song, Z. Zhong, R. Chen, and H. Chen, "ProMoE: Fast MoE-basedLLM serving using proactive caching," *arXiv:2410.22134*, 2024.

S. Fan, X. Ding, L. Zhang, and L. Mo, "MCPToolBench++: A large-scale AI agent model context protocol (MCP) tool use benchmark,"*arXiv:2508.07575*, 2025.

P.Wang,Y.Wu,Z.Wang,J.Liu,X.Song,Z.Peng,K.Deng,C. Zhang,J. Wang, J. Peng *et al.*, "MTU-Bench: A multi-granularity tool-usebenchmark for large language models," *arXiv:2410.11710*, 2024.

O. Romero and A. Abello´, "A survey of multidimensional modelingmethodologies," *Int. J. Data Warehous. Mining*, vol. 5, no. 2, pp. 1–23,2009.