Infinidat Cures RAG Inferencing Response Time Troubles

October 2024

Written by: Marc Staimer, President DSC, Sr. Contributor to theCUBE research

Introduction

The way the world interacts with unstructured data radically changed in November 2022. That's when OpenAI released its first Generative AI (GenAI) Large Language Model (LLM) called ChatGPT. What are LLMs?

LLMs are advanced artificial intelligence systems designed to understand and generate human language. They're trained on vast amounts – billions – of textual data. This enables the LLM to recognize patterns, context, and nuances in one or more languages. To put that in an easily identifiable application that most people have used, Google Translate was an early LLM.

Training is the heart of the LLM. It enables it to perform an extensive variety of tasks. Tasks such as answering questions, summarizing information, translating languages, and just as importantly engaging in conversational dialogues with users. Examples include models like GPT-3, GPT-4, and GPT-4o which utilize deep learning techniques, specifically transformer architectures, to produce coherent and contextually relevant content.

LLMs are far from perfect. They can cause some serious pains. It starts with how LLMs work. Their two modes are training and production. Effective training depends on the amount of training parameters. More is better, delivering increased accuracy. That's why the more popular LLMs are trained on hundreds of billions of parameters. The downside is that as the number of training parameters increase so does the amount time it takes to train. It takes longer and uses a lot of costly GPU hardware, power, and cooling. Yet, larger training parameters correlate with greater accuracy.

When it moves into production mode delivering content, an LLM is no longer training or learning. And this is where a significant LLM problem arises. LLMs can hallucinate – provide untrue results – if it does not know the answer to a question. By hallucinating, it means the LLM is making stuff up that seems real. This is not a trivial problem. LLM hallucinations can result is significant negative consequences. Consequences such as: the spread of misinformation leading to critical domains harm; reduced trust eroding confidence in AI systems; legal and ethical concerns increasing the potential liability for defamatory content, discriminatory content, or copyright infringement.

LLMs hallucinate or generate misleading information for several reasons including:

- **Flawed training data** – LLMs are trained on massive datasets that may contain errors or misinformation. When a model learns from flawed data, it can and often reproduces those inaccuracies in its content.
- **Incorrect information or missing context** – An LLMs purpose is to identify and replicate patterns in the data or language. This means LLMs can generate plausible content even when that content is based on incorrect information or missing context.
- **No inherent understanding of the data** – Just because LLMs mimic human languages does not mean they actually understand the content. LLMs don't natively grasp facts, nor do they have the ability to reason like a human. That can cause it to generate incorrect content or answers.
- **Tendency to generalize** – When LLMs are asked about less common topics, topics they have minimal information about, or highly specific topic details, models are likely to generalize based on what they've seen in their training data, leading to inaccuracies.
- **Question/Prompt Vagueness** – When a question is vaguely phrased it can lead an LLM to misinterpret what is being asked resulting in incorrect yet plausible results.
- **Obsolete Data** – Obsolete or stale LLM training data. The data can become stale during or after training, resulting in once again wrong but believeable results.

Eliminating or at least mitigating LLM hallucinations has become paramount to LLM usefulness. The three methods of increasing LLM accuracy are fine-tuning, prompt engineering, and retrieval augmented generation (RAG) inferencing.

Fine-tuning is the process of additional training to the LLM based on a specific dataset or for a particular task. The goal is to adapt the general capabilities of the model to perform better on specialized tasks. Fine-tuning is essentially, more training. Users often find fine-tuning time-consuming and disruptive. It also may be training on confidential data in a publicly shared model such as ChatGPT meaning others will benefit from their business' proprietary data. That's unlikely to be acceptable to most businesses. This inputting of proprietary data security issue becomes moot when it's a private LLM for their exclusive use.

Prompt engineering is the practice of creating effective inputs, or "prompts," for large language models (LLMs) to guide them in producing desired outputs. Another time-consuming process and it's more difficult to train the human users to craft better prompts than it is to train the LLM.

Retrieval-Augmented Generation (RAG) inferencing has become the method of choice for most LLM implementations. It is a hybrid model architecture combining the best capabilities of retrieval-based systems – primarily databases and data lakes – with LLMs to improve information generation, reasoning, and content creation.

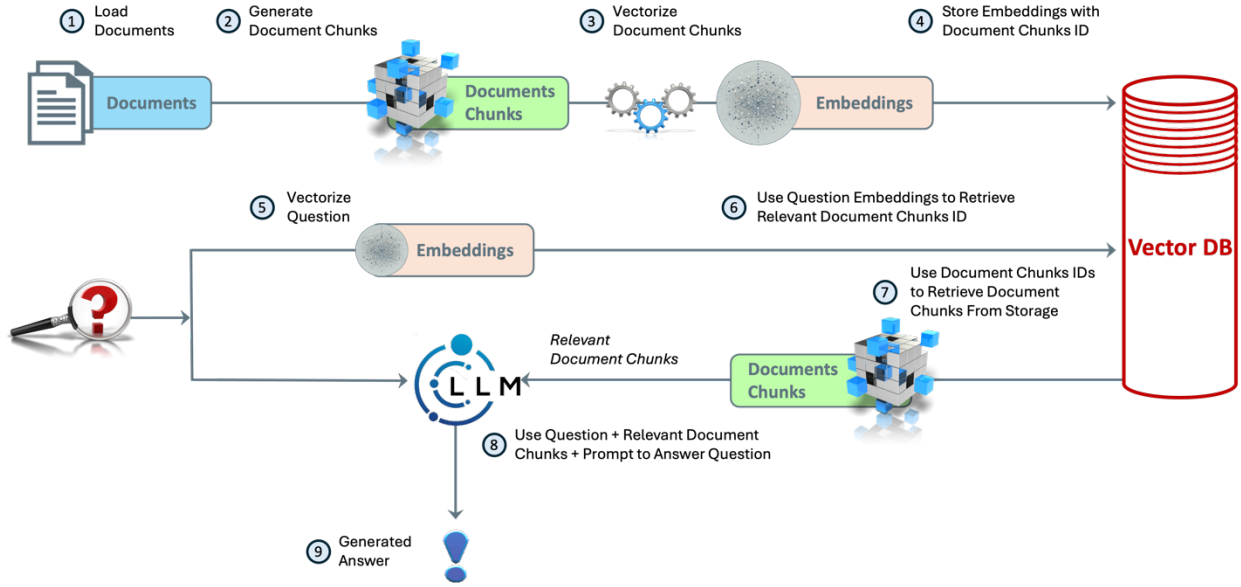


Illustration 1: RAG Inferencing Flow Chart

RAG inferencing systems are designed to retrieve the latest up-to-date relevant external information from documents, databases, or web content, and then use that information to generate more accurate and contextually germane responses.

Retrieving relevant data requires encoding the unstructured data such as documents, video, audio, photographs, illustrations, drawings, spreadsheets, their associated metadata, and more using an embedding model that abstracts features of unstructured data converting them into a set of numeric values that are coordinates known as vectors. Vectors represent the data content and not the underlying words or pixels. These vector embeddings are then indexed and mapped relative to each other, creating a vector database.

Keep in mind that regardless of where the data resides – databases, data warehouses, file stores, data lakes, etc. – it must be chunked, vectorized, and embedded into a vector database. Queries, questions, or prompts also must be vectorized and embedded. It is key that vectorization needs to be done as effortlessly and automatically as possible.

One of the things driving RAG inferencing acceptance is that when the LLM retrieves the information from RAG it does not add it back into the LLM. This means organizations can use public LLMs with their proprietary data without unintentionally sharing or providing insights from that data with anyone else.

Overall, RAG significantly boosts the performance and reliability of LLMs in real-world applications. These are the primary reasons why RAG inferencing has become the partner of choice for LLMs. However, as with all new technologies, there are issues that need to be resolved.

Premise

The fundamental key issue with RAG inferencing is latency. Also known as the amount of time the LLM must wait for the appropriate RAG response. Why is latency such a big issue? It comes down to LLM response times.

Several studies on the value of application response time over the years have connected response times to a profound impact on the organization's business. The most complete study comes from IBM's "The Economic Value of Rapid Response Time". Their extensive research revealed a deep correlation between application response time and user productivity, quality of work, morale, turnover, personnel cost, time-to-actionable-insights, time-to-action, time-to-market, and ultimately time-to-unique-revenues/profits. The numbers are stunning, as seen in the transaction rate versus application response time chart below.

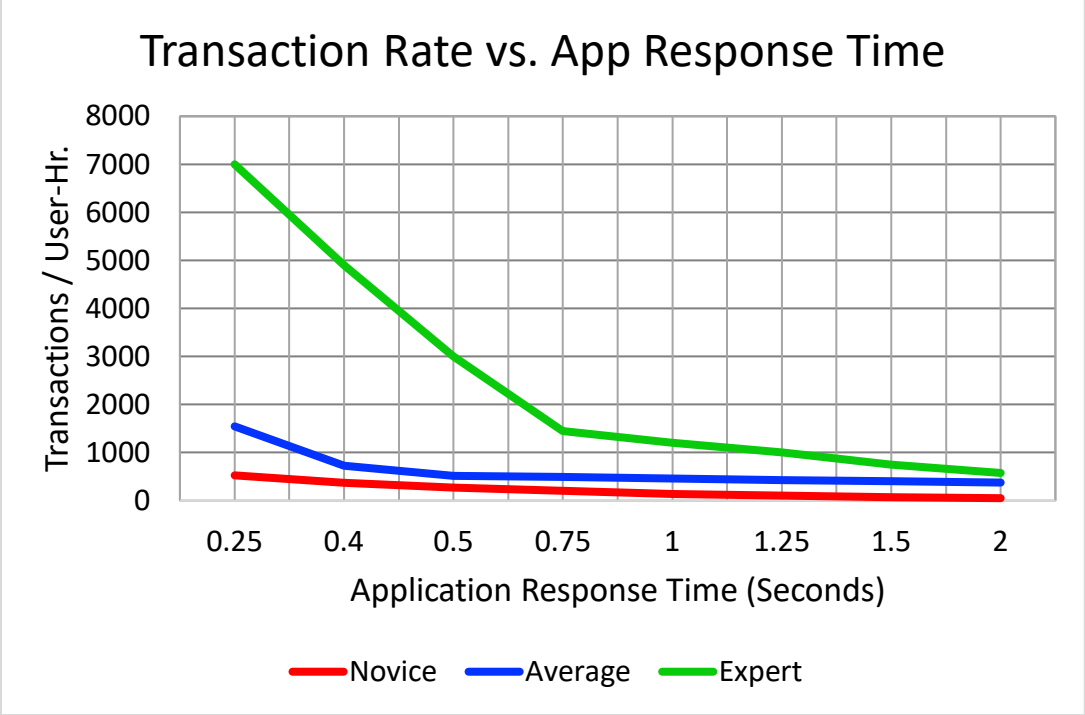


Chart 1: Application Response Time's Effect on User Transaction Rates per IBM's "Economic Value or Rapid Response Times"

The increased productivity from subsecond response times shown on this chart is mind-blowing. That's not hyperbole.

Consider that adequate user productivity requires a minimum application response time of 2 seconds or less[1]. Above 2-second response times, productivity plunges. As application response time decreases, productivity increases. Productivity takes off when application response time becomes subsecond. It then increases exponentially for expert users at ¾ of a second and for the average and novice users at 400 ms – .4 seconds[2]. The

[1] The human brain loses focus as response times exceed 2 seconds. Frustrations increase exponentially as response times slow.

[2] The 400 ms application response time is also known as the "Doherty Threshold." At the Doherty Threshold or less, users no longer perceive they're waiting on the application.

response time feels as if it is instantaneous. Any response time above 400 ms enables the user's attention to waver. When application response times come in at 400 ms or lower, user productivity skyrockets, morale significantly improves, as does quality of work, employee turnover decreases, customer loyalty increases, time-to-actionable insights accelerates, time-to-action quickens, time-to-market shortens, which in turn delivers unique revenues and profits. The savings alone from the increase in productivity can eclipse the total cost of the entire RAG inferencing infrastructure – see more in Appendix A.

Many IT LLM pros frequently focus on optimizing LLM and vector database latency. That's definitely important. But, the biggest and most common RAG latency bottleneck comes from **data storage**. Addressing that issue is urgent.

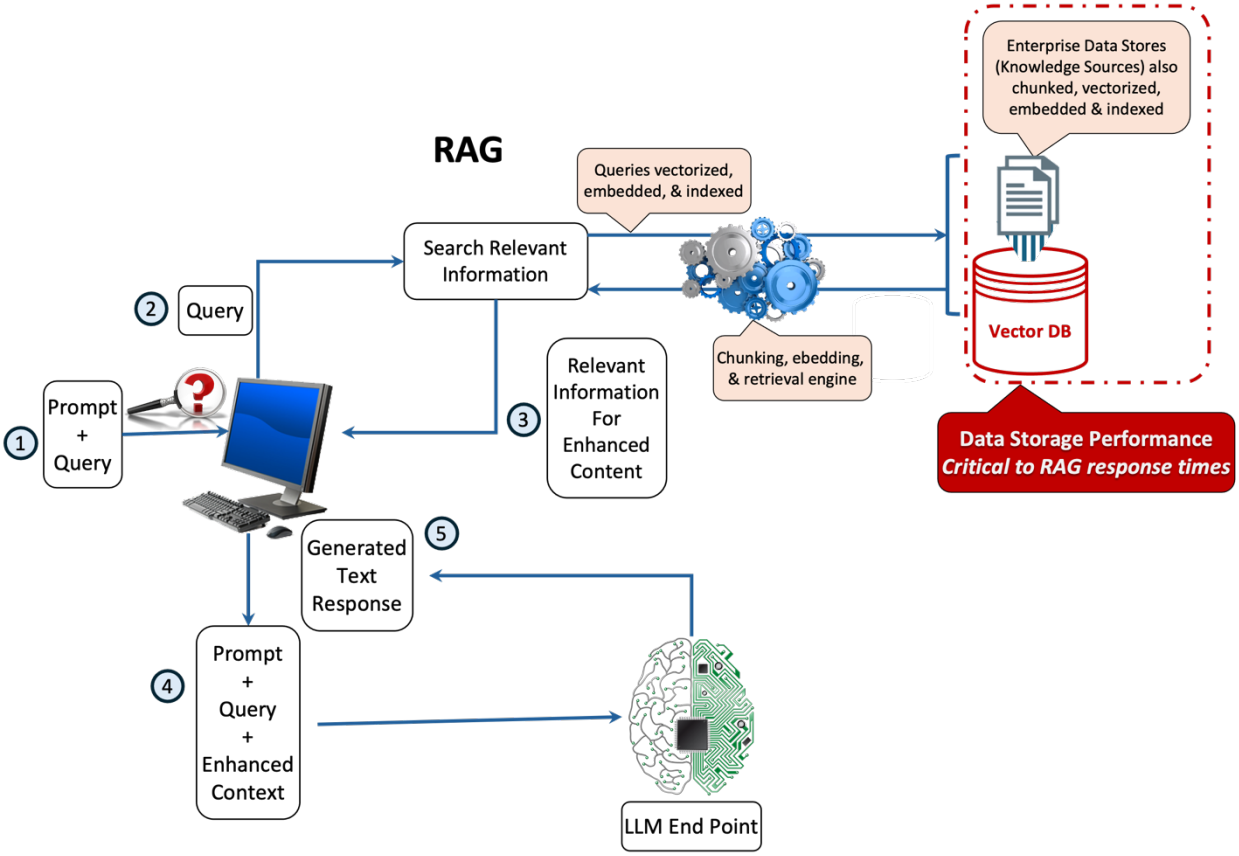


Illustration 2: RAG Inferencing Data Storage Latency Bottleneck

RAG inferencing queries often require multiple roundtrips to the storage. Each roundtrip adds to the latency. RAG storage latencies need to be as low as possible, consistent, and persistent to deliver the lowest possible LLM response times. This is not to trivialize the latency from the network and endpoints. That is an important consideration. However, data storage is commonly the biggest latency bump.

How Infinidat Solves the RAG Storage Latency Issue

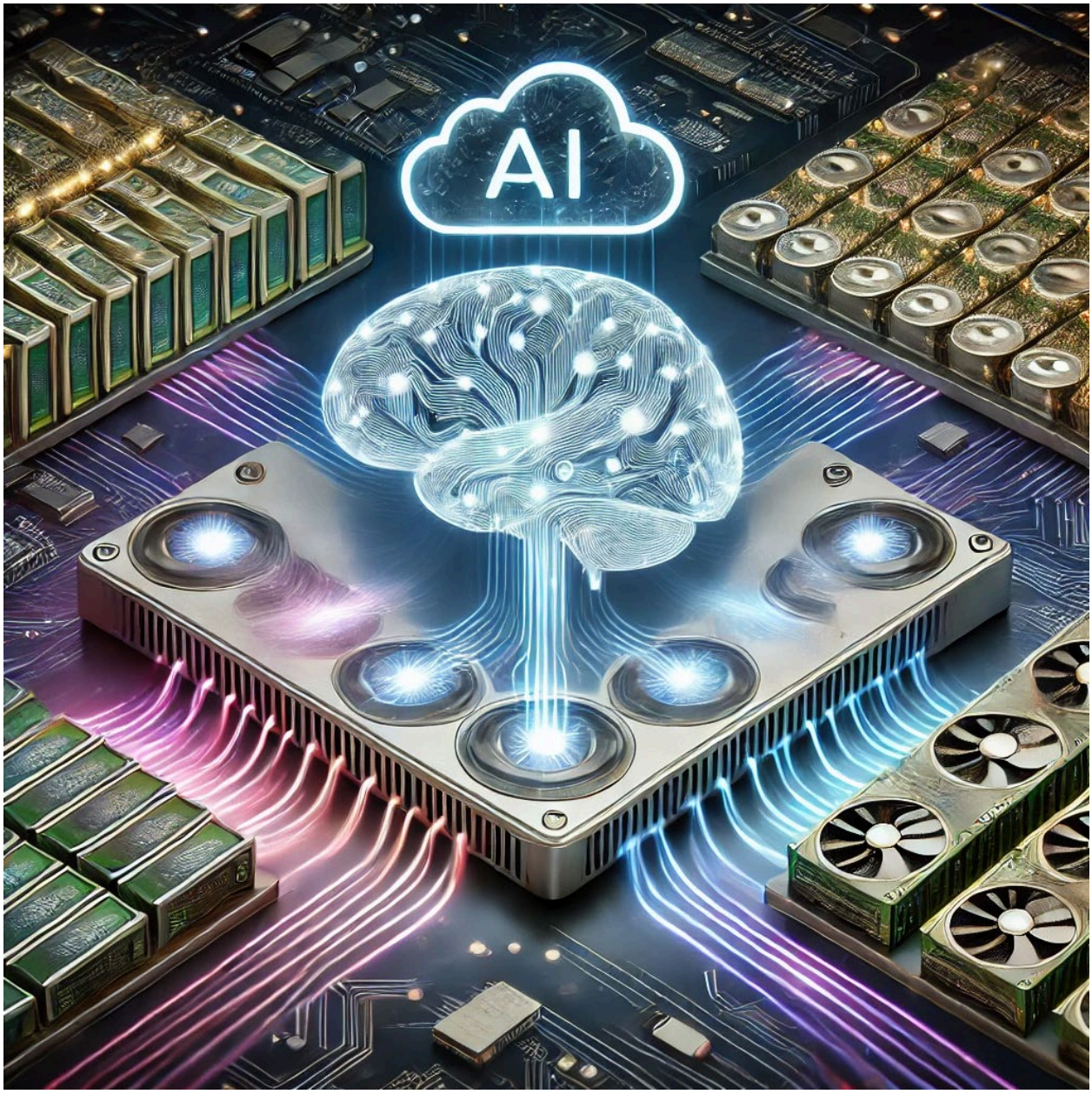
Infinidat provides two storage platforms based on an identical software platform, InfuzeOS – InfiniBox (hybrid SSD and HDD) and InfiniBox SSA (all flash SSDs). Both are ideal at solving RAG inferencing data storage latency bottlenecks. The way Infinidat solves it is unique and quite clever. It's called Neural Cache.

Neural Cache

Neural Cache applies advanced machine learning, predictive algorithms, and access trends analysis, with extensive system-level DRAM caching to heuristically optimize data placement. In other words, Neural Cache makes sure the right data is available at the lowest latency. Neural Cache does this through the use of advanced AI machine learning algorithms to predict access patterns. Over time, it learns the workload I/O patterns, or in this case LLM

RAG inferencing patterns. Those patterns enable Neural Cache to anticipate what data will likely be requested next. This results in the most relevant data being available in the high-performance caches – mainly DRAM – delivering best-in-class latency and very fast RAG inferencing response times.

The question becomes how low is the latency from Neural Cache? Since the vast majority of hits are going to be in Neural Cache's DRAM, it is very low. Infinidat claims in its documentation, 35 µs latency. To put that in perspective, consider that most all NVMe flash arrays average much higher read latencies that generally come in between 100-500 µs and as high as 1 ms, ~ 2.9 x to 29 x higher latencies. Neural Cache latencies are usually lower than the NVMe drives themselves and at worst equal to them.



Some technically astute readers will be thinking that's great, but what about the hit ratios? Hit ratios are the percentage of queries or prompts where it finds the data it's seeking in the Neural Cache. This can be a concern considering InfiniBox storage platforms have a limit of 3.456 TB. That's not a lot of cache for a storage platform that supports more than 6.6 PB of fast SSDs.

However, Neural Cache has been shown to achieve an astoundingly high cache hit ratio generally exceeding 90% based on Infinidat statistics from production customers. Keep in mind that Infinidat storage platforms are a common mainstay for Fortune 2,000 companies as well as governments, and service providers. Neural Cache and its AI machine learning algorithms are a huge factor in achieving this exceptionally high hit ratio. It unfailingly monitors the Neural Cache, learns what data is frequently accessed and the I/O patterns. From this, it anticipates what data is most likely going to be accessed next, then makes sure that data stays in the Neural Cache.

This high cache hit ratio means data is regularly served from the fastest, lowest latency DRAM tier without needing to access higher latency NVMe SSDs[3]. And even when the RAG inferencing query/prompt needs to get the data from the NVMe SSD level, it's still very low latency and lower than most other storage system platforms. The reason is that their innovative storage platform architecture, powered by InfuzeOS, onsists of:

- Active-active-active controllers providing 100% availability, guaranteed
- Market performance-leading AMD Gen4 EPYC processors, with performance guarantees
- High-performance, very low latency InfiniBand interconnect between the controllers
- Full access from any controller to any SSD/disk
- InfiniSafe technology and the extensive cyber security it provides

This architecture delivers substantial performance gains, particularly in the most demanding RAG inferencing environments. That is true even in the unlikely chance of a query/prompt that has to go to the NVMe SSDs.

Infinidat's Neural Cache ability to balance data resilience, efficiency, performance, and cost between DRAM, SSDs, and even HDDs in hybrid environments reduces the cost without reducing the performance. LLM users get the lowest latency and fastest RAG inferencing response times they need at a very affordable cost. Keep in mind Neural Cache is available in all Infinidat Storage Platforms, including InfuzeOS Cloud Edition[4].

One More Thing About Infinidat RAG Inferencing Acceleration

The Infinidat Neural Cache is not the only functionality that radically reduces RAG inferencing latencies. Infinidat recently upgraded the InfiniBox and InfiniBox SSA storage controllers to the AMD Gen4 EPYC processors. This by itself increased performance by 2.5X.

It is this combination of the Neural Cache and much faster processors that enables RAG inferencing latency to be ideally low. Thus resulting in optimal LLM response times.

Conclusion

GenAI LLMs have taken the world by storm. Unfortunately, they have a couple of significant flaws. The first is LLMs hallucinate and make things up. The second is that few organizations want to use their proprietary data to train public LLMs because it could be used by competitors to help them. The best way today to rigorously mitigate LLM hallucinations and use proprietary data with public LLMs is RAG inferencing.

The problem with RAG inferencing is too often high latencies with the biggest latency too often being the storage for the vectorized data. That high latency causes slower LLM response times. Higher LLM response times decrease user productivity, morale, turnover, and training costs while extending the time to market. Not good.

Infinidat's InfiniBox family of storage platforms change the RAG inferencing game. Neural Cache opens up the RAG inferencing storage latency bottleneck, radically reducing LLM response times. Win-win-win. Win for the LLM, win for RAG inferencing, and win for the LLM users.

And don't worry about support and cyber resilience, Infinidat has some of the best support and cyber resilience capabilities in the storage market.

[3] Even when it's not in the Neural Cache, it resides on high-performance, low-latency NVMe SSDs – just not as low latency as DRAM.

[4] InfuzeOS cloud edition is 100% functionally equivalent to Infinidat on-premises storage. This enables users to take advantage of cloud-based AI resources.

For more information

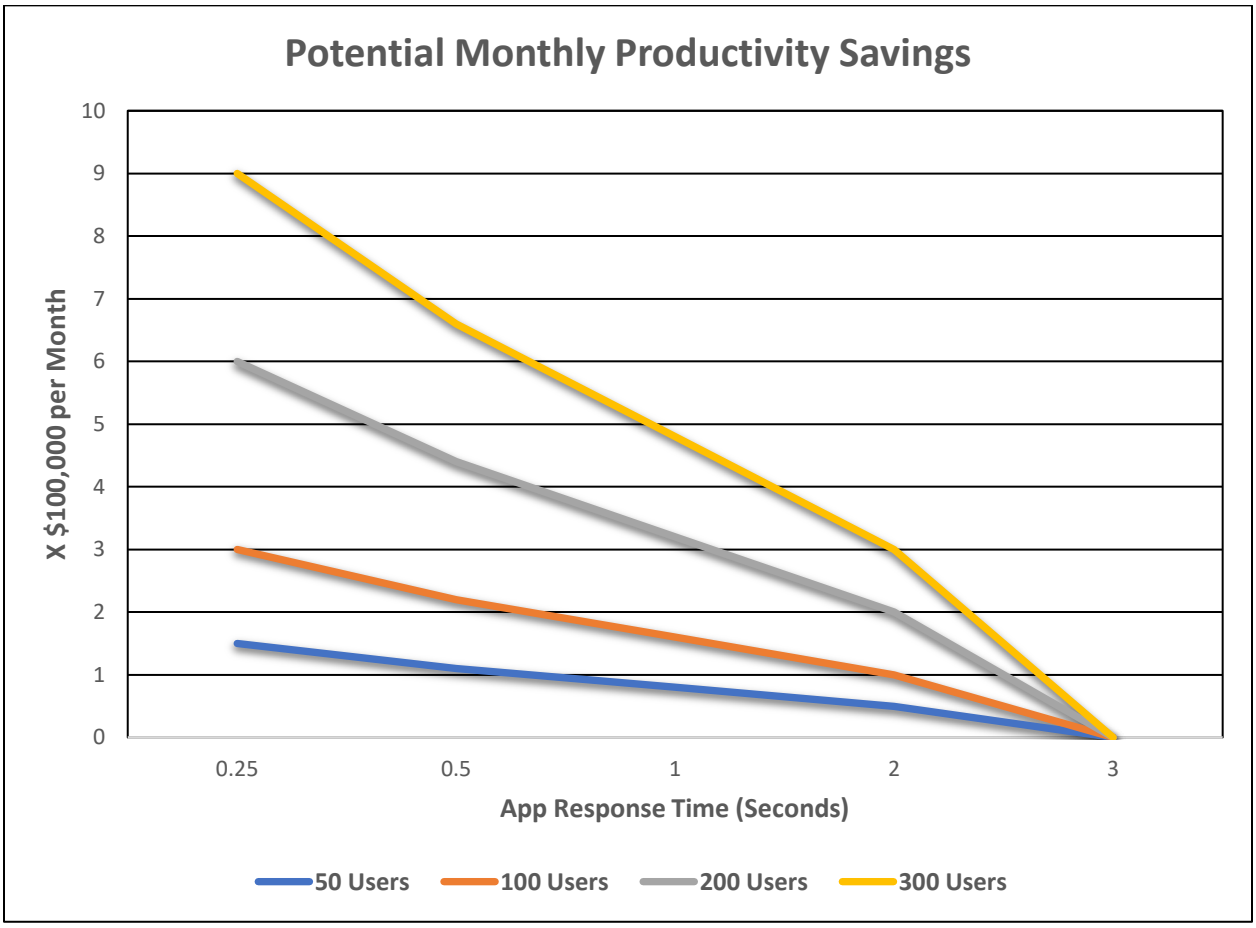
Go to: http://www.infinidat.com/

Appendix A: Measuring Productivity Cost Savings Based on Faster Application Response Times

Based on IBM's "Economic Value or Rapid Response Times" research, IT organizations have a lot to gain by providing subsecond response times. Especially response times that equal or exceed the Doherty threshold.

IBM's measured calculation productivity gains assume a generally conservative cost savings of approximately $3,000 USD per month per user at ¼ of a second. That will vary by customer. And the savings will go up the more skilled the users are.

However, as the following IBM chart shows, even a small decrease in application response times saves substantial productivity costs.



Just to put this in perspective. In the chart above based on ¼ second application response times:

- 50 users can save $1.8 million per year:
- 100 users can save as much as $3.6 million per year:
- 200 users can save as much as $7.2 million per year:
- 300 users can save as much as $10.8 million per year:

Obviously, that is a lot of potential cost savings. Just as obviously, it will vary by customer, skill sets of the users, and the application. This can have a huge impact on RAG inferencing and LLMs.