Building A Chip-Agnostic, Latency-Aware Routing Layer for Real-Time AI Inference

Tesseract Cloud, Whitepaper v0.0.1, May 2025

Petros Kalabat

Introduction

AI is fast becoming the backbone of real-time applications and critical services. Yet, today's AI infrastructure is lagging behind the demands of a truly instantaneous, ubiquitous intelligence. Consider a clinician consulting a voice-enabled AI agent during an emergency – every extra 100 milliseconds of delay could mean the difference between life and death. Or imagine a global network of autonomous systems coordinating in real time – a few hundred milliseconds of cloud latency would render them sluggish and unresponsive. Current cloud setups, dominated by centralized GPU clusters, struggle to meet these ultra-low latency, region-specific, and compliance-sensitive needs. We are building Tesseract to change that: it is a bold reimagining of the AI serving stack, a chip-agnostic and latency-aware routing layer designed to deliver *instant, intelligent* responses anywhere in the world within strict latency and policy constraints. Unlike traditional inference providers, Tesseract will deploy and operate its own edge compute clusters in key global locations to guarantee ultra-low latency and sovereign compliance.

Tesseract is to AI inference what content delivery networks were to the early web – a necessary leap to bring intelligence closer to users. It functions as a global router for AI workloads, not a model provider or a traditional "GPU cloud." Through a simple API, developers specify requirements (e.g. *"I need a response in <50 ms, using a HIPAA-compliant endpoint, with a 70B model"*), and Tesseract automatically handles the rest: routing the request to the optimal location, scheduling it on the right hardware (across GPUs, TPUs, or specialized accelerators), and enforcing any geographic or regulatory constraints. By abstracting away the complexity of where and on what a model runs, Tesseract lets developers focus on building experiences – while behind the scenes it orchestrates a global mesh of heterogeneous chip clusters to guarantee the required latency, throughput and compliance.

In the pages that follow, we lay out the vision for Tesseract and the first-principles reasoning for why this layer is inevitable. We discuss *"Why Now?"* – the confluence of trends that makes a latency-optimized AI routing layer not only necessary but timely. We examine *"Why This Is Hard"* – the physics and engineering challenges that demand a fundamentally new approach. We then explain *"Why We're Ready"* – the technological breakthroughs and industry developments that make a solution like Tesseract feasible today. Along the way, we draw analogies to high-trust infrastructure (think Cloudflare for AI or Stripe for inference), compare the latest AI chips in terms of raw performance, and illustrate Tesseract's architecture with diagrams of our routing system,

and other key elements. Finally, we explore real-world use cases (from a HIPAA-compliant clinical assistant to sovereign AI deployments and telco copilots), outline our go-to-market strategy and monetization model, and share a five-year roadmap for a planet-scale intelligent routing network.

Tesseract aims to deliver nothing short of a new intelligence layer for the internet – one that treats *latency* as a first-class concern, just as previous generations treated bandwidth and throughput. We believe that a globally optimized inference router will unlock AI applications that today are impractical or impossible, and will become an indispensable part of the AI stack in the coming era. In short, our thesis is simple: if AI is going to be everywhere, its serving infrastructure must evolve to meet it – and Tesseract is that evolution.

Why Now?

The Era of Instant AI

Over the past year, generative AI has graduated from research labs into everyday use. Billions of users now expect AI-driven answers, completions, and decisions in real time, whether in chat interfaces, voice assistants, or autonomous systems. This has led to an explosion in inference demand: as an example, Meta recently noted that developers are already purchasing LLM inference "by the billions of tokens" to power their apps.[i] We are witnessing the dawn of a new kind of internet traffic – AI *token* streams – and the scale is growing exponentially. However, unlike traditional web requests, these AI interactions often involve large neural models running complex computations, making them orders of magnitude more compute-intensive. The need for *speed* in this context is not just a nice-to-have; it is foundational. Sub-second end-to-end latency is emerging as a key requirement for next-gen applications across many fields.

"100 tokens per second is okay for chat, but it's very slow for reasoning. It's very slow for agents. And people are struggling with that today." – James Wang, Senior Executive at Cerebras Systems

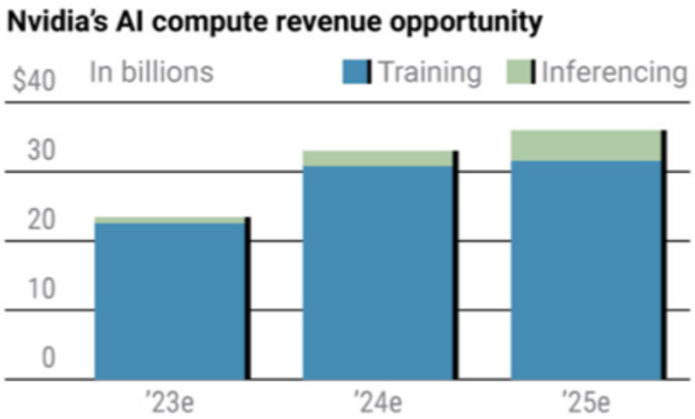

Figure 1: Nvidia's projected AI compute revenue opportunity by workload type (Training vs. Inference), showing inference's growing share through 2025.[ii]

Bottlenecks of the Old Paradigm

Today's cloud infrastructure wasn't designed for this kind of ultra-responsive AI workload. Traditional GPU servers, sitting in a few centralized regions, introduce too much latency – both computational and network – for truly real-time interaction. In fact, typical large-model APIs run at roughly 50–150 tokens per second per GPU[iii], meaning even a short sentence can take a second or more to generate, and that's *before* accounting for network transit to wherever the user might be. For a user halfway around the world from the cloud region, network latency alone can add tens of milliseconds at best, or even hundreds under poor conditions. These delays are painfully evident in current AI services; for instance, OpenAI's ChatGPT often introduces noticeable pauses between user prompt and AI response. While this is tolerable for simple Q&A, it becomes a deal-breaker for latency-critical scenarios like live conversations, decision-making in physical environments, or any interactive setting where the AI needs to appear *immediately responsive*.

Hardware Divergence and the "Token Wars"

Compounding the issue, the hardware landscape for AI inference is rapidly diversifying. GPUs were the workhorses of the last decade, but new specialized chips are now shattering performance records. For example, Cerebras Systems (with its wafer-scale engines) recently demonstrated *over 2,600 tokens per second* generation throughput on a large language model – a nearly 20× speedup versus the ~130 tokens/s typical for GPT-3 on GPU.[iv] Similarly, Groq's tensor streaming processors can sustain on the order of 500–800 tokens/s on LLM tasks, massively outperforming GPUs on a per-chip basis. Nvidia's own H100, while the king of general-purpose AI, still achieves on the order of only ~100 tokens/s on GPT-3-class models in practice. The figure below illustrates this stark disparity in throughput.

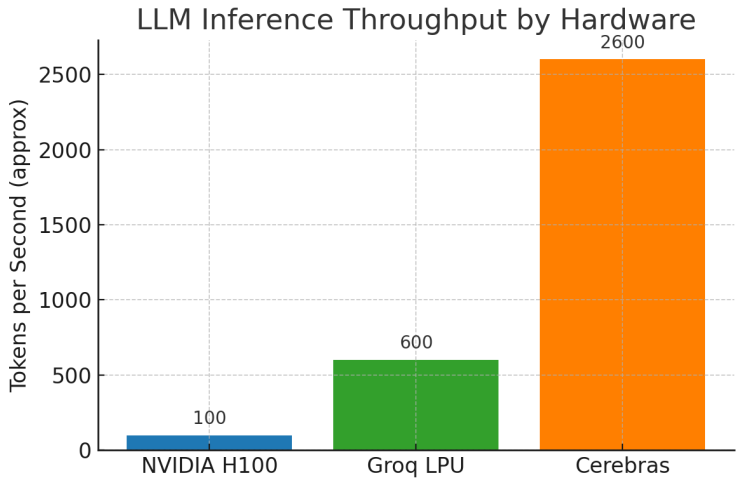

Figure 2: Approximate LLM inference throughput of different AI hardware, showing the stark contrast between a general-purpose GPU (NVIDIA H100) and specialized accelerators like Groq's LPU and Cerebras Wafer-Scale Engine. This "tokens-per-second" gap highlights why new routing intelligence is needed – the fastest hardware exists, but orchestrating workloads to run on the right chip at the right time is non-trivial.

This diversity is both an opportunity and a challenge. On one hand, the existence of chips that can deliver 10×–20× higher token throughput unlocks new possibilities – models can run at interactive speeds that were previously unreachable. Entirely new classes of applications (real-time dialogue agents, instantaneous multi-step reasoning, etc.) become feasible when inference happens in *milliseconds instead of seconds.* On the other hand, these performance gains are unevenly distributed – no single cloud or vendor currently offers all the best chips in all regions. Cerebras hardware might be available in certain data centers or through specific partnerships (e.g. Meta's Llama CPU is leveraging Cerebras in its backend[v]), Groq may be available via a different service, and Nvidia GPUs are everywhere but often not at the speed frontier. For developers, this poses a frustrating fragmentation: to get the best performance, one might have to manually route some queries to a Cerebras-powered endpoint, others to a Groq cloud, and the rest to a GPU cluster, each with its own API and quirks. Tesseract addresses this by unifying all hardware behind one routing layer. It makes the *"silicon choice"* dynamically and intelligently, so that each query uses the ideal compute resource – the developer simply sees a faster and more consistent response.

Latency Is a Physics Problem

Another reason this problem is urgent now: as we push towards human-instant response times (on the order of 10–100 ms), we run up against the ultimate limit – the speed of light. In fiber, light travels roughly 200,000 km per second (about 2/3 the speed in vacuum). That translates to ~5 ms of one-way delay per 1,000 km of distance. In other words, a signal from London to Dubai (~~ 7,000 km) takes on the order of 35 ms *just in transit*. Round-trip latencies (RTT) double that. Real-world internet routes are rarely straight lines – packets bounce through switches, take suboptimal paths, and incur serialization and queueing delays. A rule of thumb in networking is *"1,000 km ~ 10 ms RTT"* under ideal conditions, and in practice often more[vi]. What this means is that if your AI service is only running in, say, Virginia (US-East data center) and your user is in California or Germany or India, the network latency alone could be 60–150 ms. No amount of optimization on the server can claw that back – the only solution is to have inference happen closer to the user. This is precisely analogous to why CDNs arose for web content: you can't serve billions of users from one location without incurring unacceptable latency, so you replicate content across the globe. But unlike static content, AI inference can't be cached; it's a compute task that must be executed fresh for each query. Tesseract's solution is to *dynamically route* each query to the optimal execution location – often this means a data center in the same region as the user (to minimize propagation delay), but it could also mean a farther location if that's where a particular compliant hardware or model is available. The key is that Tesseract *automates* this decision,

whereas today developers have to choose a fixed region or two and suffer the consequences elsewhere.

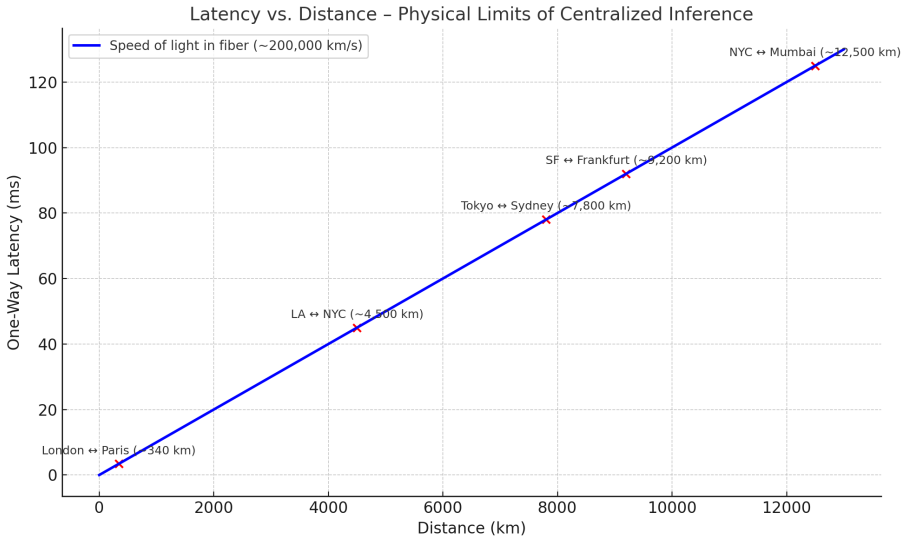

Figure 3: Even at the speed of light in fiber (~200,000 km/s), every 1,000 km adds ~10 ms of one-way delay. Centralized inference models, like routing SF ↔ Frankfurt (~9,200 km), suffer from unavoidable 90+ ms latencies before computation even begins. Tesseract solves this by routing inference to regional clusters, eliminating physical bottlenecks.

Evolving Expectations and Use Cases

"Real-time AI" is quickly becoming the expectation, not the exception. In finance, algorithmic trading and risk analysis increasingly rely on rapid AI-driven decisions, where even milliseconds confer competitive advantage. In gaming and VR, AI NPCs and generative content must respond at interactive frame rates to avoid jarring lag. In customer service, voice AI agents are expected to answer as swiftly as a human on the line would – a delay beyond a few hundred milliseconds is noticeable and breaks the flow of conversation. All these trends point to *latency* being the next battleground of AI. The giants know this: Meta's new Llama API is touting 18× faster response than competitors by using optimized hardware. other major cloud providers are racing to deploy LLMs on edge nodes and leverage faster interconnects. We're at an inflection point where raw model quality alone isn't enough – how, where, and how quickly you serve the model's output is equally critical. This is the moment to introduce a dedicated layer that treats latency and locality as first-class citizens.

Finally, there is a macro geopolitical and regulatory force making *"Why Now?"* even more pertinent: digital sovereignty. Governments and enterprises around the world have realized that relying solely on foreign or centralized AI services can be problematic, whether for privacy, compliance, or control. A recent Atlantic Council report noted that nations increasingly find "wholesale AI offerings unsuited to their needs" and are investing in sovereign AI infrastructure – but that doing so requires significant compute deployment and new approaches to manage data

and workloads.[vii] In parallel, regulations like GDPR in Europe, HIPAA in healthcare, and various data residency laws demand that sensitive data *stay within certain jurisdictions*. The current cloud paradigm struggles here – typically one ends up deploying separate instances of a service in each region to satisfy data locality, which is inefficient and hard to manage. Tesseract, by design, offers a way to route and run inference within the appropriate jurisdiction on a per-request basis. Developers specify compliance constraints (e.g. "this user's data cannot leave Canada" or "use only GDPR-compliant processors"), and the Tesseract router will honor that, sending the job to, say, a Montreal cluster if needed. In summary, the convergence of soaring demand, new hardware, speed-of-light limits, real-time use cases, and sovereignty concerns makes *now* the ideal time – perhaps the only time – to introduce a solution like Tesseract.

Why This Is Hard

If building a "latency-optimal, globally distributed inference layer" were easy, it would already exist. To appreciate Tesseract's approach, we must understand the sheer difficulty of the problem it tackles. In essence, Tesseract is solving a multidimensional optimization: minimize latency (both network and processing) for each request, maximize throughput and utilization of expensive hardware, obey all policy constraints (region, compliance, model type), and do this seamlessly across dozens of hardware backends and data centers under varying load. All of this needs to happen transparently, in real time, for millions of requests from thousands of applications. This is a grand orchestration challenge at the intersection of physics, distributed systems, and chip architecture. The core challenges are as follows.

The Speed of Light and the Need for Distribution

As discussed, you simply cannot serve a global user base from a single location with low latency. The physics of propagation delay enforce a hard floor on response times. For truly interactive AI (sub-50 ms responses), one needs infrastructure distributed across continents, ideally within a few hundred kilometers of every end-user. That implies having inference compute in many regions. However, unlike static content, you can't pre-compute and store responses; you have to have *live compute* everywhere. Maintaining copies of massive models (potentially hundreds of GB each) across many sites is an enormous undertaking – both in terms of storage and keeping them updated. It's not feasible to host every possible model in every edge location. Tesseract's insight is that we don't have to – instead, we route intelligently: critical low-latency interactions might be handled by a nearby smaller model or cached partial result, whereas larger or less time-sensitive jobs can be forwarded to a central cluster. We also utilize regional hubs that strike a balance: not every city will have a 70B model loaded, but perhaps one per major time-zone, with smaller proxy models at the true edge. Designing this hierarchy and routing logic is a hard problem: it involves predicting user experience impact vs. model accuracy and potentially doing things like cascading: e.g. a quick local model generates an interim result while a better model fetch comes a few hundred ms later – the system might even decide when to "wait for a better

answer" vs. when to return the quick one, as described in the EdgeSight paper.[viii] These kinds of decisions go beyond traditional CDNs and require new algorithms.

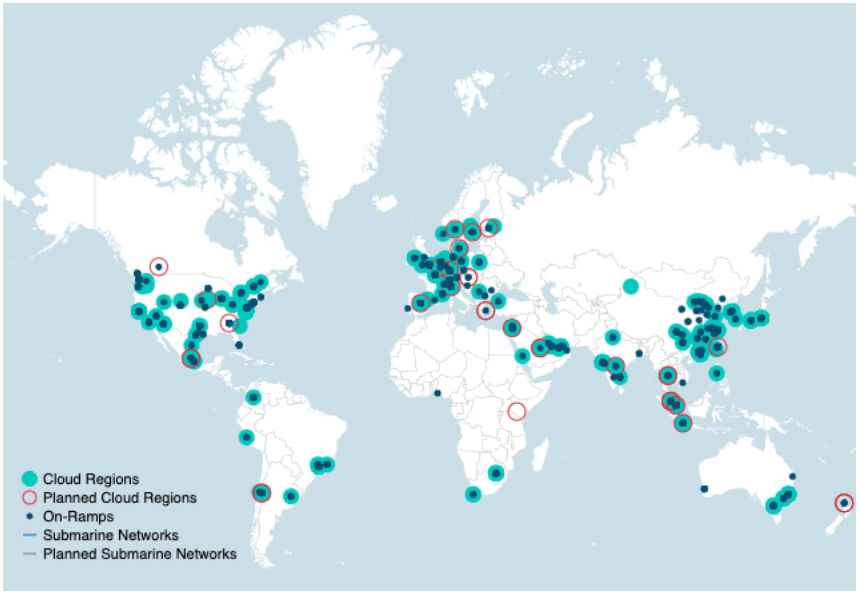

Figure 4. Global distribution of major cloud regions, on-ramps, and submarine networks as of Q1 2024. Meeting sub-50ms inference SLAs requires routing across this geography-aware infrastructure. Data only includes IaaS cloud providers from Alibaba, AWS, Google Cloud, Huawei Cloud, IBM, Microsoft Azure, Oracle Cloud, and Tencent Cloud.[ix]

Heterogeneous Hardware and Software Stacks

Today's AI accelerators come with distinct software ecosystems. Nvidia GPUs run CUDA and TensorRT; Google TPUs use XLA; Groq has its own compiler; Cerebras has a bespoke runtime; there are FPGAs, Graphcore IPUs, AMD MI250s, and more. A truly chip-agnostic layer must either integrate with all these or find a lowest common denominator (like deploying via containers or using ONNX as an intermediate). This integration is *non-trivial*. It means Tesseract's scheduler needs real-time awareness of each device's capabilities (memory, throughput, loaded models, etc.) and queue lengths. It means if a model isn't present on a given device, we might dynamically load or even *transfer* the model (which could take seconds for tens of GBs – obviously to be avoided during a live request). Our approach leverages ahead-of-time placement: Tesseract's backend continuously replicates and load-balances model weights across the network based on demand patterns, so that when a request comes in, chances are the needed model is already available in the target zone. We borrow techniques from cloud function scheduling and edge caching – essentially treating models as large objects to be staged where needed, as described in the SCAR paper[x]. Still, doing this across heterogeneous hardware pushes the boundaries of existing orchestration systems. We are building into Tesseract a kind of capability database – a directory of which model versions are loaded where, which hardware types can run them, and the performance profiles (e.g. *model X runs at 120 tokens/s on GPU vs 600 on Groq vs 2600 on*

Cerebras). This database updates as we benchmark and as hardware/kernel updates come in. It's used by the scheduler to make cost-aware decisions. For instance, if a user request can be fulfilled in 20 ms on a Cerebras in a slightly farther region or in 40 ms on a local GPU, the scheduler will know these options and pick appropriately based on the SLA. This kind of chip-aware routing is a hard problem because it's not static – load conditions can make a normally fast chip slower if it's busy with other tasks, etc. So Tesseract has to do adaptive routing, much like an internet router re-routing traffic during congestion. Our system monitors the actual latency achieved and can dynamically switch routes if, say, the chosen accelerator is unexpectedly bogged down.

Latency vs. Throughput Trade-offs

Naively, one could always run every request on the absolute fastest hardware available for that model. But that fastest hardware (say a wafer-scale engine) might be a scarce (and expensive) resource. It could serve many requests in parallel if they're willing to wait a bit, or one request at a time with minimal delay. There's a scheduling dilemma: *do you queue up tasks on the super-fast chip, or send some to slower chips that are free?* For example, imagine 100 requests arrive simultaneously, each needing a 1-second computation on a Cerebras (which can do them in, say, 0.1 s each but can only run 10 concurrently). If we funnel all 100 to the Cerebras, some will wait in queue and overall latency might exceed 1 s for the later ones. If we instead distribute some to GPUs that take 1 s each but can handle them immediately, those might complete in parallel. The optimal strategy might be a mix. This is analogous to load-balancing on steroids: it's not just distributing identical tasks across identical servers – it's distributing varied-duration tasks across very non-identical servers, with *time-sensitive deadlines*. We are essentially solving an online scheduling problem with deadlines and heterogeneous processors – which is NP-hard in general. Tesseract will utilize heuristics and real-time feedback to approximate optimal decisions – in other words, a feedback-driven deployment strategy[xi]. We prioritize meeting the requested latency SLA for each request, while also maximizing overall throughput. One strategy is to classify requests by urgency: truly urgent (as in user waiting) requests get sent to whichever available resource can finish by the deadline (even if that means using a less efficient hardware because the efficient one is busy), whereas less urgent ones (batch jobs, or prefetches, etc.) can wait for an optimal slot. Over time, our goal is to have an economic scheduler as well, where the pricing (if the user has chosen a cheaper tier with higher allowable latency, for instance) influences the decision. In short, building a *latency-aware scheduler* that works in real production conditions is a major challenge.

Network and Coordination Complexity

Unlike a single-cloud inference service, Tesseract's architecture spans many network environments. We will have edge routers that take in user requests (these might run in cloud PoPs or ISP data centers), core hubs with heavyweight compute, and peer links between them for forwarding. Ensuring that data is transferred swiftly and reliably between these points is crucial. For example, if a request arrives in Rio de Janeiro but our nearest loaded model is in São Paulo, the request payload (perhaps an audio snippet or text) must be sent there quickly, and the results returned. We effectively operate our own backbone or rely on premium network routes to keep

inter-region hops within a few milliseconds where possible. Also, since we may handle sensitive data, we need encryption and proper security at every hop, which adds overhead. We've had to optimize the networking stack, possibly leveraging technologies like gRPC[xii] with custom protocols or even QUIC[xiii] in some cases to reduce handshake latency. Moreover, having a distributed system means dealing with the usual suspects: failover, congestion control, and consistency. Tesseract must be robust to a node going down – it should seamlessly reroute to another cluster. It also should not overwhelm any link; if too many large payloads are going over a connection and causing delays, the system should adapt (perhaps by opting for a slightly slower local compute instead of saturating a transcontinental link for many requests).

Compliance and Policy Enforcement

One of Tesseract's selling points is enforcing compliance (data locality, encryption, specific hardware trust levels, etc.) on behalf of the developer. This is challenging because it introduces constraints that can override what would otherwise be purely performance-based decisions. For example, perhaps the lowest-latency hardware for a user in Country X is in Country Y, but regulations forbid sending data to Y. Tesseract must then route to a less optimal location in Country X itself, or perhaps a sanctioned "sovereign" cloud. We maintain policy tags on both data and compute nodes. Every request may carry tags like "EU-only" or "HIPAA" and every node is annotated with tags like "EU" or "HIPAA-certified." The router only considers destinations that satisfy the policy intersection. This essentially reduces the pool of available resources, making the scheduling even harder. But it is non-negotiable – violating compliance is a non-starter. We therefore treat policy constraints with the highest priority in the routing logic (hard filters, not soft preferences). To minimize the performance hit, we are seeking to deploy compliant infrastructure in multiple regions so that, for example, an EU-only request doesn't have to travel from Germany all the way to a single EU server in Ireland – it might find one in Frankfurt or Paris, etc. Still, achieving high performance under strict isolation constraints is difficult. In some cases, if a certain country has very limited AI infrastructure, an *honest* answer might be: we cannot meet a 50 ms SLA and keep the data in that country – physics and current infrastructure won't allow it. In such cases, Tesseract's contract is to inform the developer or fallback gracefully (perhaps run a smaller model locally that meets the time constraint, as a compromise). Navigating these trade-offs between *performance* and *policy* is a new frontier for system design.

Orchestration of State (KV cache, etc.)

A technical but important challenge: large language model inference often involves stateful caches (e.g. the key-value cache of past attention layers for faster decoding of subsequent tokens). If an interaction involves multiple requests in sequence (like a multi-turn conversation), it is beneficial to stick to the same server or at least ensure the context cache is transferred, otherwise each response will be slower if it has to recompute context from scratch. Tesseract's router, in latency-optimal fashion, might be tempted to route each user turn to a different region based on load – but that could harm performance due to lost cache locality. We solve this by implementing a session affinity and cache synchronization mechanism. Essentially, we detect if a series of requests are linked (e.g. same session ID) and then prefer to keep routing them to a place where the context resides. If we must move (say the original node went down or became

too slow), we transfer the needed context state through our network. This is akin to how edge networks move objects or how distributed databases replicate state – it's complex but doable. NVIDIA's recent work on distributed inference emphasizes how smart routers can avoid redundant work by leveraging KV cache management.[xiv] We have drawn inspiration from those techniques (see *Architecture* section), effectively making Tesseract's router *cache-aware*. This adds another layer of difficulty: our routing decisions aren't just stateless per request; they sometimes have to account for *where the data or context is*. We've built a metadata layer that tracks, for example, "session ABC's context is currently stored at Node 5." This way, even if Node 5 is not the absolutely fastest for the next query, we might still route there because it will save us 20 ms of recomputation. These kinds of considerations significantly complicate routing logic compared to, say, a stateless HTTP request router.

All the above challenges – geographic distribution, heterogeneity, scheduling, networking, compliance, statefulness – make it clear why a simple extension of current cloud systems isn't enough. We need a fundamentally new layer that is purpose-built to handle these issues. It's *hard* – but not impossible. By using first-principles thinking (e.g. starting from physical limits and working upward) and by borrowing proven ideas from analogous domains (CDNs, distributed databases, HPC schedulers), we've architected Tesseract to meet these challenges head-on. In the next section, we delve into how exactly we do it and why we believe *"we're ready"* to deploy this at scale.

Why We're Ready

The idea of a global AI routing layer may seem ambitious, but the pieces needed to build it have finally fallen into place. Tesseract stands on the shoulders of advancements in hardware, networking, and distributed software that have matured in just the last couple of years. Here's why we believe this is the right time – technologically and operationally – to launch Tesseract.

Unprecedented Hardware Availability

Never before have we had access to such a variety of AI accelerators with cloud-grade availability. In 2025, for the first time, multiple new AI chips are commercially accessible beyond research labs. Cerebras has deployed its CS-2 and CS-3 systems in data centers across North America that can be rented (indeed, Meta is leveraging a network of Cerebras data centers to power its Llama 2 API). Groq offers its inference chips through a cloud service and via partnerships (Meta also announced a Groq partnership[xv]). SambaNova and others have offerings too. Meanwhile, every major cloud provider continues to invest in GPU fleets (NVIDIA H100s, etc.) and is rolling out specialized inference instances. This means Tesseract can strike deals or use APIs to tap into a diverse hardware pool.

While Tesseract is designed to be hardware-agnostic and able to orchestrate across third-party infrastructure when necessary, we are committed to owning and operating our core compute clusters in strategic latency zones. These Tesseract-owned sites — ranging from high-density metro data centers to sovereign edge hubs — form the backbone of our low-latency, compliance-

enforcing inference network. This direct control enables us to deliver consistent performance SLAs, ensure rigorous physical security and regulatory compliance, and optimize chip placement at the hardware level. By pairing owned infrastructure in priority regions with federated access to trusted partner clouds, Tesseract maintains full-stack control where it matters most, while scaling flexibly elsewhere.

High-Speed Global Networks

The past decade has seen massive investments in fiber optics, undersea cables, and internet exchange points. Cloud providers like AWS, Azure, and Google have their own private backbones that circle the globe, offering low-latency links between continents. Companies like Cloudflare and Akamai have built out thousands of edge locations, demonstrating that maintaining a presence in metros worldwide is feasible and cost-effective. Akamai, in fact, is repurposing its CDN network to deliver cloud computing with 4,000+ edge PoPs and dozens of core sites, explicitly targeting distributed AI inference workloads.[xvi] The distributed cloud model – where computing is done in many locations – is no longer theory, it's commercially proven. Tesseract can leverage these networks; we can deploy our routing software on existing edge data centers and ride on top of established backbones to shuttle data between zones. The result is that a user in, say, South Africa can be connected to a European data center in tens of milliseconds, or a user in rural Asia can reach a city hub quickly, etc. We also benefit from protocols like HTTP/3[xvii] and QUIC, which reduce handshake overhead and improve throughput on lossy links – meaning even the network communication overhead for our routing is now minimal. In summary, the network layer has "grown up" to support a system like Tesseract: we have global reach and fast pipes practically everywhere.

Distributed Systems Expertise and Tooling

The software community has made great strides in orchestrating workloads across data centers. Kubernetes and similar technologies have become standard for multi-cluster management, and projects like KubeFed[xviii] and Virtual Kubelet[xix] can span across regions. While these alone aren't sufficient for Tesseract's needs, they provide a base of knowledge and tooling. We're not writing everything from scratch – we build on known components. For instance, we use an extended version of etcd[xx] (a distributed key-value store) to hold our global state (model registry, node health, etc.) which is something the community has battle-tested for reliability. For scheduling, we take inspiration from research like Google's Borg[xxi] and open-source systems like Ray[xxii] and Flyte[xxiii] which handle distributed ML tasks. Even more relevant, new inference-serving frameworks like NVIDIA's Triton Inference Server and Microsoft's Orca have built-in support for multi-model, multi-device serving. NVIDIA has just released Nvidia Dynamo for distributed inference, which showcases a "Planner" and "Smart Router" coordinating multi-GPU setups. This validates our approach – even within one data center, the idea of a smart router to split inference phases and manage workload is now proven. Tesseract extends this across data centers and

across vendors. We've learned from Dynamo's design (which separates an API front-end, a scheduler, a router, etc.) – see figure below – and applied similar modular principles in Tesseract.

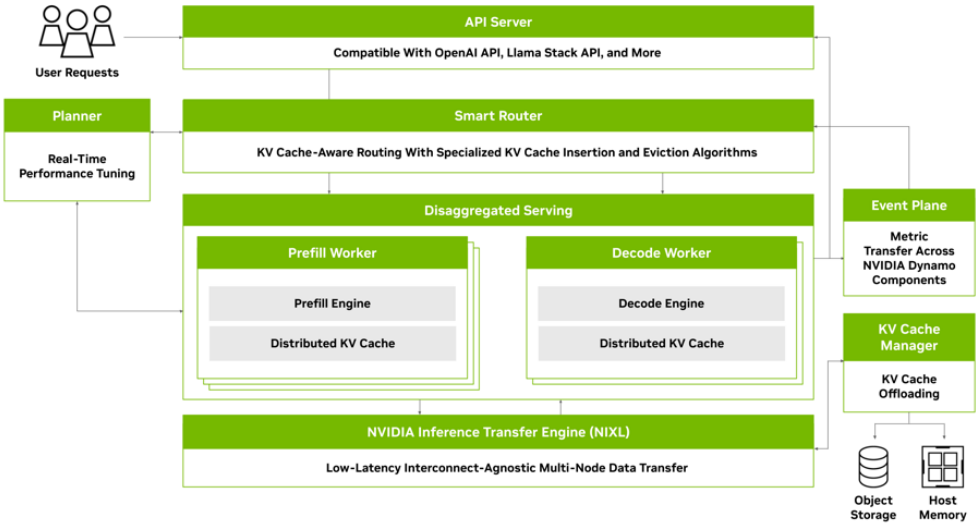

Figure 5: An example inference routing architecture (NVIDIA's Dynamo architecture[xxiv]) illustrating how a front-end API server connects to a Planner (for real-time performance tuning), which then works with a Smart Router to dispatch parts of a request to appropriate workers (e.g. Prefill and Decode workers in a disaggregated serving pipeline). Tesseract's design employs a similar layered approach – separating global planning from local routing and using distributed workers – but extends it to a geo-distributed, multi-accelerator context.

The broader point is: the industry now has confidence in such layered architectures. We are ready to move beyond the simplistic "send request to one GPU and wait" model. There is growing adoption of ideas like model parallelism, pipeline parallelism, and distributed inference in production (for example, running the transformer *attention* phase on one set of GPUs and the *feed-forward* phase on another to optimize throughput). These complex orchestration techniques are becoming mainstream, indicating that teams and tools are capable of handling the complexity. Tesseract is essentially leveraging all that know-how and taking it to the next logical step: not just distributing *within* a server or cluster, but *across clusters and across the globe*. We are actively assembling a team that has built large-scale distributed systems (ex-Google Borg engineers, CDN architects, etc.), and we're confident that, with modern cloud-native tools, we can deploy and manage Tesseract's infrastructure reliably. Auto-scaling groups, service meshes, observability stacks (Prometheus/Grafana), chaos engineering – all of these are part of our strategy to ensure Tesseract runs as a high-availability service, despite its complexity.

Regulatory and Market Readiness

Beyond pure tech, the *market* is ready for this abstraction. Companies and developers are now acutely aware of latency and compliance issues. Five years ago, few talked about data locality in AI or worried about inference latency – they were mostly concerned with accuracy. But high-

profile incidents (like public ChatGPT outages or privacy snafus) have made enterprises cautious. The idea of an AI CDN or inference routing layer is no longer alien. In fact, some forward-thinking organizations have attempted DIY solutions: we've seen enterprises and individuals use one cloud's region for US traffic and another cloud's region for EU traffic, splitting logic in their applications – essentially a crude manual routing to satisfy GDPR.[xxv] We know of a team that use an FPGA inference server on-prem for certain tasks and fallback to an API for others. These ad-hoc solutions demonstrate *demand*, but they are inefficient. Tesseract can subsume all that logic into a professional service. And importantly, trust in such a service can be earned now. Thanks to pioneers like Cloudflare (which handles enormous volumes of critical internet traffic as a proxy) and Stripe (which handles sensitive financial transactions behind a simple API), developers have mental models for what we're doing. We can say "we're like Cloudflare Workers, but for AI inference – you send us the request, we run it in a safe, optimal place, and return the result." This draws on a *high-trust analogy* that the market already accepts. Cloudflare has even launched some early products in this vein (like Cloudflare's security-focused AI proxy[xxvi], which shows they are thinking in this direction, though their focus is on security, not latency). Similarly, when we say "Stripe for inference," we mean we handle all the messy integration with various hardware providers and give you one clean API, much like Stripe hides dozens of payment networks behind one interface. This resonates with developers who are overwhelmed by the fast-changing AI hardware scene.

Early adopters are lining up because their pain is real. Voice AI companies, for example, have prototype deployments where they run an LLM-based assistant on a local hospital server (to ensure HIPAA compliance) but struggle to also achieve low latency and keep it updated with the best models. They are eager for a solution that can seamlessly use a local node for the sensitive data handling and perhaps a cloud node for less sensitive parts, all while maintaining an easy API. Telecom operators are also exploring AI at the network edge – they have data centers at cell tower aggregation points for 5G and want to run AI there to manage network operations or provide new services. They have the infrastructure but lack the software to utilize it for AI – Tesseract can fill that gap by routing relevant tasks (like analyzing local cell traffic patterns with an AI model) to their edge compute. We know of a major telco that is interested in offering *"telco AI copilots"* to optimize network parameters; they mentioned that a central cloud approach was too slow for the real-time requirements. This is direct market validation that distributed, latency-aware inference is needed. Even national governments – as part of their sovereign AI initiatives – are planning "national AI clouds" but could benefit from a unifying layer. Instead of each country reinventing the wheel, Tesseract could act as the neutral broker that links sovereign clouds with global reach when allowed and isolates them when required. The climate today, with talk of AI regulation and digital sovereignty, means a company positioning itself as *the control layer* for where and how AI runs will find receptive ears.

Economics and Incentives

Finally, from a business readiness perspective: the economics now favor a platform like Tesseract. AI inference is costly at scale, and organizations are highly motivated to optimize it. If Tesseract can save 30% of latency or 20% of cost by routing to a better-suited chip or location, that directly translates to competitive advantage or bottom-line savings for our users. For instance,

Cerebras famously advertised inference at $0.10 per million tokens, which is significantly cheaper than typical GPU costs. But not everyone can integrate Cerebras into their stack easily – Tesseract can, thus passing on cost savings. Likewise, if we can keep a request local and avoid a round-trip to a cloud 1,000 miles away, we might save on bandwidth egress costs (which cloud providers charge for). All these little wins add up. The market is very sensitive now to inference cost-per-query; there's a reason open-source models are booming – companies want to avoid high API fees. Tesseract rides that wave by enabling use of the *optimal* resource (including one's own, if available). We foresee even cloud providers welcoming integration: a platform like ours could intelligently burst workloads to their clouds when needed, or offload them when not, essentially helping balance across providers (somewhat analogous to multi-CDN strategies websites use). This was not the case a few years back when one provider (like OpenAI) dominated and everyone just paid whatever price for lack of options. Now there's competition and a strong incentive to optimize.

In short, we're ready for Tesseract because the tech stack (hardware + network + software) has matured to support it, and the stakeholders (developers, enterprises, regulators) have evolved to demand it. Tesseract is not a research project or a moonshot; it's a practical synthesis of capabilities that are here *now*. The timing is ideal to capture the moment and establish this routing layer before the ecosystem fragments further or ossifies around suboptimal architectures.

Tesseract Architecture: A Closer Look

(In this section, we provide a technical tour of Tesseract's design – illustrating how the system is structured to achieve the ambitious goals outlined above. We include diagrams of the routing architecture, explain how latency maps inform decisions, and describe the cluster mesh that underpins our global network.)

At a high level, Tesseract's architecture consists of three layers: (1) Edge Gateways that interface with developers and end-users, (2) a Global Routing Core that makes intelligent decisions and orchestrates execution, and (3) Backend Compute Clusters housing the diverse AI accelerators that actually run the models. These layers work as follows.

A user request (e.g. an API call to generate text, or an audio stream for transcription) first hits a Tesseract Edge Gateway. This is a lightweight endpoint close to the user – typically deployed in a regional data center or cloud region near the source of the request. The Gateway immediately timestamps the request (for latency measurements), performs authentication and quick sanity checks, then packages the request (including user-provided constraints like latency SLO or compliance tags) into our internal format. It then forwards it to the Global Router.

The Global Routing Core is the brain of Tesseract. It consists of a distributed controller (for resilience, there are multiple routing brain nodes in different geographies that sync state) that runs our scheduling and policy algorithms. Upon receiving a request, the router consults several things: a Latency Map, a Resource Map, and a Policy Map. The *Latency Map* is essentially a matrix of estimated network latencies between all our gateway locations and backend cluster locations. We

continuously measure ping times and throughput between sites (similar to how CDNs map the internet) so we know, for example, that “Gateway in Sydney -> Cluster in Singapore ~ 50 ms” whereas “Gateway Sydney -> Cluster London ~ 300 ms.” This informs the router which clusters are even viable candidates given the latency budget. Next, the *Resource Map* tells us what hardware and models are available at each cluster and their current load. It might say “Cluster A: 10 GPUs free, 20 GPUs busy, and Model X version 3 is loaded” etc. This is updated in real-time (every few seconds) by local monitors in each cluster. Finally, the *Policy Map* encodes restrictions – e.g. it will mark certain clusters as disallowed for certain data (if the request has tag “EU-only” then any cluster with a “non-EU” tag is filtered out).

Using these maps, the router runs a scheduling algorithm to pick the optimal plan for the request. In simple cases, this means choosing one cluster and one device type to execute the model. In advanced cases, the router may decide to split the request into parts – e.g., run the first part of a conversation on a local small model to get a quick draft, and simultaneously send the full context to a bigger model in a data center, then merge the results. Or it might decide to use two different models in parallel (if the API call was actually an ensemble request). The most common plan, though, is “send the query to the best cluster and hardware.” The router includes estimated queueing times in its decision. For example, if the nearest cluster is very congested, and a slightly further cluster has free capacity, the router might choose the latter, calculating that 10 ms extra network is better than 100 ms waiting in a queue. We use a modified shortest-job-first with constraints algorithm: effectively, among the available options that meet compliance and model requirements, we predict the one that will finish first and choose it. Those predictions come from a combination of static benchmarks and dynamic info (like current queue lengths).

Once a decision is made, the router communicates with the target cluster. Within each backend cluster, we run a local orchestrator (think of it as a regional scheduler) that then launches the inference task on the specific hardware. For example, the global router might say to Cluster “EU-Frankfurt”: *“Run Model=GPT-4, Input ID=xyz on device type=Cerebras (or a specific device ID if reserved).”* The local orchestrator on that cluster will either already have an inference server process running on the chosen device (with the model loaded), or it will spin one up via containers if needed (our system can dynamically start a model runtime if it’s not already in memory – though we try to keep hot models resident to avoid cold starts). The inference runs as it would normally on that hardware. During execution, we have hooks that stream partial results if applicable (for example, streaming tokens back as they are generated, which our gateways can relay to the user for incremental response). The local orchestrator monitors execution and collects metrics (total time, any errors, etc.).

As soon as the model computation is complete, the result is sent back to the Edge Gateway that originated the request (traversing the network). The Gateway then delivers it to the user application. Because our gateways are stateful, they know how to reassemble any results (if we split across models or did something fancy, the gateway does the merging – e.g., picking the faster response or combining pieces). In most cases, it’s one-to-one: one request yields one response from one backend. The gateway then calculates the total latency and logs it along with which route was taken, feeding this info back into our metrics store for learning.

Meanwhile, the Global Router continuously updates its models based on what happened. If a certain cluster is seeing queue growth, it will mark it as such; if network latency to a region changes (we detect spikes or slow links), that's updated. Over time, we also adjust placement of models: if a lot of traffic for Model Y is coming from South America and we currently always route it to North America, our backend might proactively load Model Y in our Brazil cluster to serve future requests faster – this is handled by a background job that analyzes patterns (this is akin to how CDNs decide to cache content in a region after enough local hits). In some cases, we may also decide to deploy more clusters in that region.

The cluster mesh underlying all this is a network of Tesseract-operated data centers and sovereign clusters, forming the physical backbone of our AI inference platform. We have meshed connectivity (i.e., each cluster can talk to the others or at least to a few hub peers), but we often use a hub-and-spoke for control (global decisions are done in a few central nodes for simplicity, with redundancy). The mesh is used for data transfer, e.g., if a user in one region needs to get data to a cluster in another, or if we need to migrate a model state from one cluster to another.

Our latency maps are a crucial piece of this puzzle: we regularly generate maps of round-trip times between all our major locations. We even factor in known speed-of-light limits – essentially, our scheduler knows the *geodesic distance* and uses that to sanity-check latency (if measured latency is way higher than theoretical minimum, it might mean a network problem; we can route around via alternate paths if needed). By visualizing these latency maps, we can decide where to add new PoPs or which routes to optimize. (In an extended version of this paper, we would include a world map highlighting latency contours – e.g., showing 20ms radius around each cluster – to illustrate coverage. Imagine a map with circles indicating how far 10ms reaches from each node, demonstrating that with ~200 nodes globally we can cover most of the population under 20ms.)

From an implementation standpoint, Tesseract uses a combination of existing technologies and custom-built components. The Edge Gateways are essentially Nginx or Envoy proxies with our custom Lua or WASM filters that implement the authentication and streaming logic. The Global Router service is currently written in Python for performance, using gRPC to communicate with cluster orchestrators. We use a pub-sub system (Kafka or Pulsar) to broadcast state updates (like "model X loaded on cluster A"). For model execution, we integrate with NVIDIA Triton Inference Server where possible (which supports multi-framework backends). For non-NVIDIA hardware, we use their vendor-provided runtime (like Cerebras' SDK, etc.) wrapped in a shim that talks to our orchestrator. All components will be containerized and deployed via Kubernetes across our future infrastructure, with careful pinning for low latency (e.g., we use node affinity to ensure our global router pods live in certain regions, etc.).

Crucially, observability is built in: every request carries a trace ID, and we trace it across the entire path (gateway -> router -> cluster -> device) using OpenTelemetry. This gives us detailed latency breakdowns (network vs compute vs queue). It not only helps us debug and improve the system, but it's also surfaced to developers in logs – so they can see, for instance, "Your request was served from *Frankfurt on a Groq LPU, took 37ms (15ms network, 22ms processing).*" This transparency builds trust and also educates users on how latency budgets are spent.

In essence, Tesseract's architecture is the synthesis of ideas from distributed computing: we've taken cues from CDN design, HPC job schedulers, cloud load balancers, and specialized ML serving frameworks, and combined them into a cohesive system tailored for real-time AI. The result is a globally distributed "AI kernel" that manages compute across space and time, abstracting a complex mesh of clusters into a simple API call.

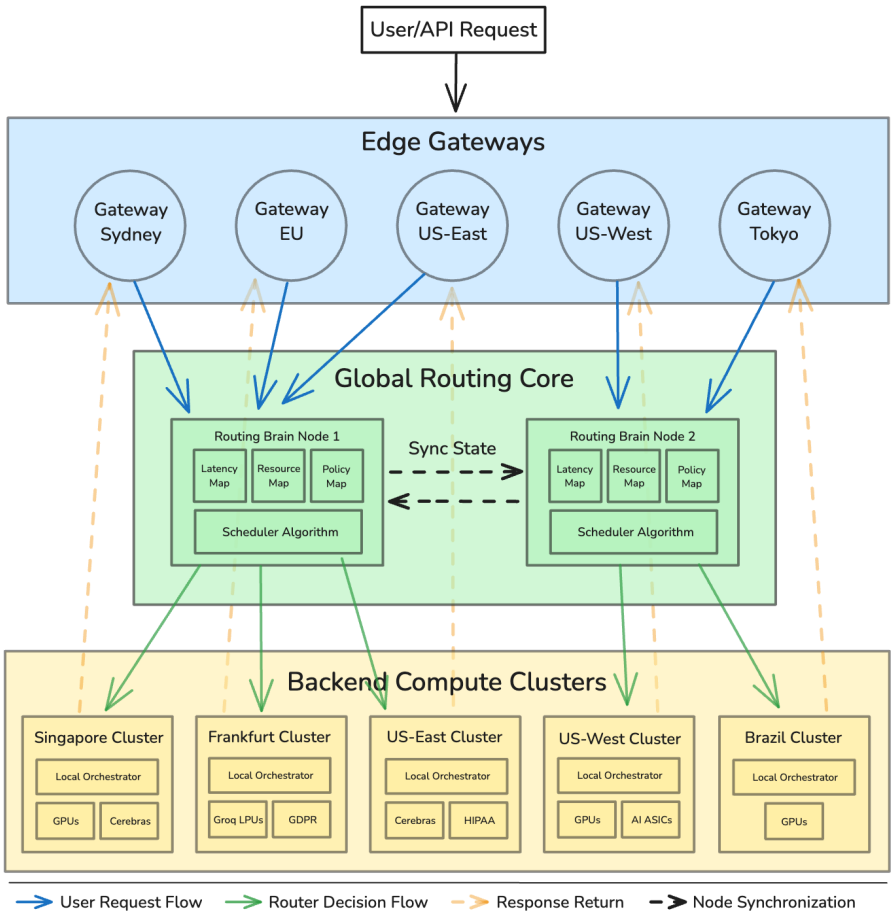

Figure 6: Tesseract's distributed architecture – user requests enter through edge gateways, pass through the global routing core (which uses latency, resource, and policy maps), and are routed to the optimal backend compute cluster. Responses are returned to the user via the original gateway. The router continuously updates based on latency feedback, queue lengths, and policy constraints.

The Scoring Algorithm

At the heart of Tesseract's intelligence lies a sophisticated scoring algorithm that drives routing decisions. This algorithm processes multiple factors to determine the optimal backend for each inference request. Below is the current mathematical representation behind these decisions.

Final Score Equation

$$S_{\text{final}} = f_{\text{load}} \cdot f_{\text{cost}} \cdot f_{\text{priority}} \cdot f_{\text{health}} \cdot S_{\text{base}}$$

Where:

Base Score

$$S_{\text{base}} = L_{\text{backend}} \cdot C_{\text{token}}$$

- L_{backend} = Backend processing latency in milliseconds
- C_{token} = Cost per token in dollars

Health Adjustment Factor

$$\begin{cases} \infty & \text{if } status = \text{DOWN} \\ 1.5 & \text{if } status = \text{DEGRADED} \\ 1.0 & \text{if } status = \text{HEALTHY} \end{cases}$$

Priority Adjustment Factor

$$f_{\text{priority}} = \frac{1}{P}$$

- P = Request priority ($1 - 5$, where 5 is the highest)

Cost Preference Adjustment Factor

$$\begin{cases} 1000 \cdot C_{\text{token}} & \text{if } prefer\_cost\_over\_latency = \text{TRUE} \\ 1.0 & \text{otherwise} \end{cases}$$

Load Adjustment Factor

$$\begin{cases} 1 + \frac{Q}{10} & if\ Q > 100 \\ 1 + \frac{L_{\text{current}}}{100} & \text{otherwise} \end{cases}$$

- Q = Estimated queue time in milliseconds
- L_{current} = Current backend load ($0 - 100\%$)

Total Expected Latency

While not part of the score calculation, the expected total latency is:

$$\begin{cases} 1.5 \cdot (L_{\text{backend}} + L_{\text{network}}) + Q & \text{if } status = \text{DOWNGRADED} \\ L_{\text{backend}} + L_{\text{network}} + Q & \text{otherwise} \end{cases}$$

- L_{network} = Network latency between user and backend
- Q = Estimated queue time

Total Request Cost

$$C_{\text{total}} = C_{\text{token}} \cdot T_{\text{input}}$$

- T_{input} = Number of input tokens in the request

The power of this algorithm lies in its balanced approach to multiple competing factors. Using a multiplicative model rather than weighted addition ensures that each factor can meaningfully influence the final decision. Lower scores are better, directing requests to backends that provide the optimal balance of performance, cost, and reliability.

What makes this approach unique is its ability to adapt to changing conditions. As backend loads fluctuate or network conditions shift, scoring adjusts automatically. The algorithm also honors strict constraints through our filtering system - backends that don't meet compliance requirements or capacity needs are filtered before scoring occurs.

This mathematical framework drives Tesseract's intelligence, enabling it to make optimal routing decisions in milliseconds while balancing competing priorities of speed, cost, and compliance.

Use Cases and Early Demand

Tesseract's routing layer is a general infrastructure, but it is not a solution in search of a problem – quite the opposite. We have been pulled by real-world use cases that are underserved by the status quo and are clamoring for a solution. Here we highlight a few, to illustrate the breadth and importance of a system like Tesseract:

HIPAA-Compliant Voice Agent for Clinicians

Consider a hospital that wants to deploy an AI assistant to transcribe and summarize doctor-patient conversations in real time, and then offer diagnostic suggestions. The speech recognition and language model involved are large and data-hungry. Privacy laws (HIPAA in the US) require that *no patient data leaves the hospital's secure environment*. This typically forces such systems to run *on-premises*, often on limited hardware, which in turn can't handle the load of a big model quickly – leading to either laggy performance or reliance on smaller (less accurate) models. With Tesseract, a new possibility emerges: we can have a Tesseract node *on-prem* that handles the sensitive audio data locally (perhaps running a smaller model to get a rough transcription quickly), but that node can also securely and *transparently route part of the task to a larger off-site model* without violating compliance – for example, by only sending de-identified or encrypted embeddings to a powerful cloud cluster and receiving back a refined analysis. Alternatively, if policy absolutely forbids any data leaving, Tesseract will ensure the entire pipeline stays on hardware in that hospital or region – but even then, it can optimize latency by routing between,

say, an on-site GPU and a nearby city's compute if allowed. One hospital IT pilot we engaged with expressed that *"we need the performance of the cloud with the privacy of on-prem – if you can solve that, it's a game changer."* Tesseract is built for exactly that: it routes the request in a way that *by design* complies with data residency (all PHI – protected health info – stays in the secure enclave) while still exploiting external compute when possible. Early demand in this sector is high because medical AI startups and hospital innovation departments are trying to deliver AI assistants that *feel instantaneous* (a doctor won't wait 5 seconds for an answer while with a patient) and are *trustworthy with data*. Tesseract turns what was a painful trade-off between speed and compliance into a configurable policy: the developer can choose the threshold of what stays local vs remote, and Tesseract will honor that and attempt to maximize speed within those bounds.

Sovereign AI and Government Services

National governments are increasingly insisting on "sovereign inference" – meaning AI services where both the models and data are under national jurisdiction. For instance, a European government might mandate that all AI processing on its citizens' data happens in data centers located on EU soil. If a global company wants to serve that country's users, they might have to deploy separate infrastructure there or use a local provider, which is inefficient and costly. Tesseract provides a cleaner solution: from the user's perspective it's the same API, but under the hood, if the user is in Country X (or the data is labeled with country-of-origin X), Tesseract's policy engine will route it only to clusters designated for X. This could be the country's own sovereign cloud or any provider that has a region in X. Crucially, Tesseract can also enable *federation*: imagine each country runs its own AI cloud with certain models (some may be better at certain tasks, tuned to local language or law). Through Tesseract, a request can potentially tap into another region's model without violating data rules by only exchanging non-sensitive results. For example, a question asked in one country might be answered by consulting a larger model in another country that's been trained on broader knowledge, but done in a way that the question is abstracted (to not include personal data) – Tesseract could facilitate this kind of cross-border model cooperation if allowed by policy. Early demand for such capabilities is signaled by projects like the UAE's Core42 sovereign AI platform, which envisions an "AI intelligence grid wrapping the globe like a second nervous system" where data stays within borders but insights are shared.[xxvii] This grand vision requires exactly the kind of routing of workloads vs. data that Tesseract can do – essentially moving the *compute* to the data rather than data to compute. We hope to discuss with a consortium of EU cloud providers who will see Tesseract as a neutral solution to offer their government clients: we would route European users to European clusters, American users to US clusters, etc., ensuring compliance with each region's laws by construction. Sovereign AI isn't just about location either – it's also about auditability and control. Tesseract's architecture can integrate with oversight systems (for example, logging every inference that happens for a government service into a secure audit trail). In sovereign contexts, being able to *prove* that no data went to an unapproved region is critical – our detailed tracing and tagging system provides that evidence.

Telco Copilots and Network AI

Modern telecommunications networks (5G and beyond) are extremely complex and can benefit from AI in operations – things like automated anomaly detection in network traffic, AI-driven optimization of antenna parameters, or AI assistants for network engineers (co-pilots that help configure routers via natural language). The challenge is that telco operations are real-time and often geographically localized. An issue in a cell tower needs analysis within a second or two to be useful (for instance, to re-route traffic or switch frequencies). Shipping all the raw data from towers to a central cloud AI introduces too much latency (and huge bandwidth costs). Telcos have been deploying edge compute at central offices (local data centers) precisely to handle such workloads on-site. Tesseract slots in perfectly here: it can run on the telco's edge compute and ensure that inference on local data (say a burst of network telemetry) happens on that local node or a nearby regional node, meeting the timing requirements. At the same time, if some larger analysis that's not as time-sensitive is needed, it can route that to a core cloud to use a bigger model. One concrete example: a *network operations co-pilot* that field engineers use to troubleshoot issues. The engineer might speak or type a query, and the system must pull in data from various local network logs and respond with an analysis. Tesseract can route the language understanding part to a large model in the cloud, but route the query of local network data to an on-prem smaller model that has been trained on that telco's proprietary data (and which cannot leave the premises). This hybrid routing yields an answer that is both comprehensive and timely. Telcos have shown early interest – companies like Mavenir are already developing AI copilots for network management, noting that these systems should ideally run "within 12–18 months" in live networks.[xxviii] Our discussions with a mid-sized operator indicated they have dozens of small data centers across their coverage area that sit mostly underutilized – they could potentially welcome something like Tesseract to leverage those for AI tasks for low latency, while still connecting to the cloud when needed. In effect, Tesseract can turn a telco's distributed footprint into a cohesive AI cloud tuned for telecom needs. This is very analogous to how Cloudflare enabled code (Workers) to run at edge locations; here we enable AI models to run at edge telco sites via routing.

Autonomous Robotics

Robotic systems – from factory floor arms to autonomous drones and mobile warehouse fleets – are rapidly adopting AI models to interpret vision, navigate space, and execute multimodal instructions. These workloads often combine perception (e.g., object detection or semantic segmentation) with decision-making (e.g., path planning or tool control), some of which must occur in under 100 milliseconds. The traditional model of sending data to a central cloud for inference simply doesn't meet the latency requirements – or the reliability expectations – of modern robotics. Connectivity may be intermittent (e.g., for field-deployed drones), and the physical constraints of onboard compute mean most robots can't host large models themselves. Meanwhile, companies often wish to retain full control over operational data, which may include IP-sensitive manufacturing layouts, proprietary object metadata, or mission-critical command logs.

Tesseract provides a new kind of architecture for robotic intelligence. A local inference node (either embedded in the robot or in a nearby edge server) can serve lightweight models for immediate reactions – like visual object tracking or obstacle avoidance – while delegating more complex reasoning or language tasks to a regional Tesseract cluster. This enables a hybrid

architecture where, for example, a warehouse robot can understand “pick the nearest green box with label X” in real time, and if uncertain, escalate a partial observation to a larger offsite model for clarification or human-in-the-loop verification – without breaking the latency budget. When necessary, data stays fully local; if policy allows, abstracted features or embeddings may be routed upward for higher-fidelity inference.

Financial Trading and Secure Enterprise AI

Another use case cluster is high-frequency trading firms or banks that need rapid AI-based decisions but absolutely do not want to rely on external clouds for IP reasons. They might deploy Tesseract entirely in their own controlled environment (we can offer a private deployment) to route tasks between their on-prem accelerators and a co-located datacenter. The key is the latency optimization – even within a city, 1–2 ms saved is huge for trading. Tesseract’s fine-grained routing (and potentially scheduling to ensure the fastest chips get the urgent tasks) gives them an edge. Similarly, large enterprises might have a mix of on-prem GPU servers and some cloud bursts – Tesseract can seamlessly route internal employees’ AI requests to internal servers first, only going to cloud if capacity is exceeded or if a certain model isn’t available internally. This ensures data governance (most stuff stays in-house) and cost savings (cloud is only used as needed). We have an early design partnership with a global bank exploring an “AI routing fabric” across their datacenters worldwide, so that a query from their London office might use idle GPUs in New York overnight, etc., all automatically. This is essentially an internal version of Tesseract, proving the concept is flexible.

In all these cases, what’s common is that latency, locality, and specialization are crucial, and current solutions fall short. Some have tried to patch the problem: e.g., an on-prem healthcare deployment with an older GPU might try to prune the model to run faster, sacrificing accuracy; or a telco might try a rudimentary script to decide which API to call based on user location. These are stop-gaps. The strong interest we’ve received from pilot customers tells us that a general solution – where they can just declare their constraints and let the system optimize – is extremely valuable.

One particularly enthusiastic feedback came from a startup building real-time voice translation earpieces. They said *“We’re not in the business of picking hardware or cloud regions – we just need it to work everywhere under 200 ms. If Tesseract can do that, we’ll be interested.”* That encapsulates the appeal: we handle the messy details of global inference optimization, and the developer just sees their application meeting its requirements magically.

Finally, it’s worth noting that Tesseract is model-agnostic as well – it can route any model type (vision, NLP, etc.). This means it can serve as the common layer for a variety of applications beyond the few listed. Early demand spans from cloud gaming companies (who want to inject AI opponents with minimal lag) to e-commerce platforms (who want personalized recommendations generated on the fly in the user’s region for compliance). The diversity of use cases assures us that Tesseract’s applicability is broad. We are initially focusing on the ones with the greatest pain (voice AI and regulated industries), but over time we foresee this being used anywhere AI inference and latency requirements intersect.

Monetization and Go-to-Market Strategy

Tesseract's value proposition is deeply tied to performance and reliability – thus our monetization is aligned with delivering that value. We envision a usage-based pricing model with premium service tiers, akin to how cloud services or CDNs charge. Here's how we plan to structure it:

Per-Token Pricing

Since much of modern AI inference (especially LLMs) is measured in tokens, and our cost drivers scale with tokens processed, we will charge per million tokens (or similar unit) processed through Tesseract. Developers are already used to thinking in these terms (OpenAI, for example, charges per 1K tokens). Tesseract's per-token price will be slightly above raw hardware costs to account for the routing service. But because we can intelligently use cheaper hardware when appropriate, we believe we can actually *save* customers money relative to them always using, say, an expensive GPU cloud. For example, if a request is served on a Cerebras at $0.10 per million tokens and we add our margin, maybe we charge $0.12 per million – still far less than the ~$0.40+ per million that a GPU-based API might cost. In cases where we provide clear speed improvements but on the same hardware, we'll justify our cost by the improved utilization (e.g., our scheduling might reduce idle time). Essentially, usage-based pricing aligns us with the customer: they pay for what they use, and if we can handle more throughput (more tokens) for them quickly, it benefits both sides.

Cost Category	Centralized Inference	Tesseract Distributed Inference
Compute Cost	~$4,000/month (e.g., $0.40/M tokens via H100-based API)	~$1,200/month (smart routing to Cerebras, GPUs, ASICs — avg. $0.12/M tokens)
Latency Penalty	80–200 ms (due to network distance and queuing)	<50 ms (regional routing, SLA-aware)
Redundancy / Failover	Requires multi-region duplication (~$800/month overhead)	Built-in via global mesh
Compliance Costs	Requires duplicate infrastructure for HIPAA/GDPR zones (~$300–$600/month)	Policy-aware routing handles data locality without infra duplication
Chip Lock-In	Yes (usually NVIDIA GPUs, prone to price volatility)	No (routes to fastest available chip per SLA: GPU, Groq, Cerebras, AMD, etc.)
Overprovisioning Costs	30–50% extra capacity to handle peak demand	Peak-smoothing via smart routing & dynamic load balancing
Bandwidth & Egress Costs	High (inter-region traffic, user-to-cloud roundtrips)	Lower (local/regional execution minimizes egress)
SLA Enforcement	Enterprise-tier pricing and manual configuration	Native in routing API — SLA-based pricing tiers

Total Est. Monthly Cost	**~$5,500/month (including compliance and redundancy)**	**~$1,400/month (optimized via SLA-aware routing + diverse silicon selection)**

Table 1: Cost comparison between traditional centralized LLM inference and Tesseract's distributed, chip-agnostic inference routing. By dynamically selecting optimal silicon and routing requests based on latency, compliance, and model type, Tesseract significantly reduces compute and infra-related costs — while improving SLA adherence and user experience.

We will offer Service Level Agreements (SLAs) for latency and availability at premium tiers. For mission-critical applications, a customer might pay, say, a 20% premium to guarantee that 99th percentile latency is under X ms in certain regions, or that failover routes will always be available. This resembles how CDN enterprise contracts work – higher fee for guaranteed performance. Under the hood, that might mean we reserve capacity or deploy dedicated instances for that customer's workloads to ensure their SLA. If we breach, we could give credits back, etc. This approach monetizes the quality of service. Many customers, like banks or hospitals, will gladly pay more for a guaranteed latency (because it's tied to their user experience or compliance needs). We might call this "Tesseract Enterprise" – with features like dedicated routing instances, custom compliance auditing, etc., all at an SLA-based pricing (could be monthly with overage charges, etc.).

Regional and Dedicated Deployments

Some clients might want a dedicated deployment (for example, a government might want Tesseract running entirely in their sovereign cloud, not shared with others). We can license Tesseract in such scenarios for an annual fee, based on capacity. This is not our primary model, but it's an option. We'd still potentially connect those private deployments into the broader network if allowed (federation), possibly charging a fee when their traffic leaves their domain. For most customers, however, the fastest and most cost-effective option will be to route through Tesseract-owned regional clusters. By owning the core infrastructure, we can optimize hardware utilization, minimize egress and network overhead, and provide SLA-backed performance tiers without relying on third-party constraints."

Marketplace Revenue Sharing

Looking forward, Tesseract could become a marketplace for AI hardware providers and model providers. For instance, if a new startup has a super-fast chip in one region, they could hook into Tesseract so we route tasks there, and we share revenue for the usage we send. Similarly, model vendors (say someone has an extremely optimized model for legal text) could plug in – Tesseract could route appropriate queries to that model if it meets criteria, and we'd have a revenue share. This marketplace concept means part of our monetization will be a cut of others' services that we route to. Essentially, Tesseract becomes a *broker* taking a fee for connecting supply and demand. This is similar to how Stripe takes a small percentage of each transaction or how cloud marketplaces work. However, early on we'll keep it simple and mostly route to either our own

managed clusters or partner infrastructure with known costs. Our go-to-market (GTM) strategy starts with targeting domains where our differentiated capabilities solve an acute problem:

Voice AI and Real-Time Communication

This is our beachhead. We are in pilot with a voice AI company (as mentioned) to be the backbone of their real-time transcription and agent system. We plan to publicly showcase how Tesseract enables, for example, a live bilingual conversation agent that translates speech with <50 ms delay – something not possible with naive setups. We will create case studies around healthcare voice assistants (HIPAA use case) and call center augmentation. These industries have relatively few decision-makers (enterprise sales model) and huge pain points that Tesseract addresses (latency + compliance). Success here will build credibility and reference deployments.

Regulated and Government Workloads

In parallel, we are engaging with government digital service agencies and defense research orgs that are interested in sovereign AI. Our strategy is to secure a couple of flagship government projects – e.g., a European cloud initiative or a smart city project in the Middle East – where Tesseract is used to enforce data localization while providing AI services (could be city-wide language translation or surveillance analytics with privacy). These big wins not only bring revenue but also stamp us as the go-to solution for *trustworthy AI infrastructure*. They often come with stringent requirements that will further harden our platform (for instance, proving end-to-end encryption and audit logs to a govt. oversight board).

Enterprise AI Infrastructure Providers

We'll partner with integrators and AI platforms that serve enterprises. For example, some companies specialize in deploying AI behind firewalls; by integrating Tesseract, they can add multi-site routing to their offerings. Our GTM here is through partnerships – riding the distribution channels of others. We have early conversations with a large cloud vendor to offer Tesseract as part of their edge cloud solutions (where their hardware + our routing = solution for telco and retail scenarios). Co-selling with established players accelerates adoption and reduces the friction of a new entrant selling to Fortune 500 IT departments.

In terms of marketing, we will model a lot of our thought leadership on the successful approach taken by Cloudflare and others: publishing performance benchmarks, demonstrating novel capabilities, and educating the market about the importance of latency. A bold blog title like "Why your AI needs an edge – the physics of latency in AI" (echoing some arguments from this whitepaper) can generate buzz. We'll also highlight the analogies: e.g., publish content like *"The Cloudflare Moment for AI Inference"* to drive the point home that this is an inevitable evolution of architecture. By framing it in familiar terms, we lower the barrier for CTOs to get it.

As for competition, currently no major cloud fully offers what we do (they might have regional endpoints, but not a dynamic multi-chip router). We actually see an opportunity to be a neutral player that even uses the big clouds as underlying providers. If any of them tries to replicate our functionality, it's likely to be limited to their own environment (e.g., an AWS-only latency router

across AWS regions), whereas Tesseract is cloud-agnostic and cross-vendor. That independence is a selling point.

Monetization will also be supported by efficiency gains we provide. We can offer to reduce a client's cloud bill by X% through better routing – a very tangible pitch. For example, if we detect 30% of their requests could be handled on cheaper hardware or in cheaper regions, we do that and show them the savings. Perhaps we even implement a system where we charge a fraction of the savings (performance-based pricing). But predominantly, the per-token and SLA model keeps it straightforward.

To ensure we capture value, we must be careful not to be seen as just a low-level utility that can be easily swapped – we differentiate with the smarts and the SLAs. Over time, as customers deeply integrate (especially using our API and relying on us for compliance), the switching costs increase – we become part of their infrastructure, much like a payment gateway or CDN is hard to rip out once integrated everywhere.

In summary, our GTM starts focused (voice and regulated sectors), leverages partnerships for scale (cloud and integrators), and our monetization aligns usage with value delivered (fast, compliant inference). This should allow us to land initial high-value customers and then expand usage within those organizations (from one application to many, since once Tesseract is in their stack, new projects will naturally consider using it too).

Our five-year vision (covered next) also plays into our sales narrative: we're not just solving today's problems, we're building the infrastructure for the next decade of AI – so choosing Tesseract is a strategic decision, not a tactical band-aid.

Deployment Economics and Cost Model

Tesseract is engineered to deliver low-latency, high-throughput inference from sovereign, compliance-grade infrastructure — without hyperscale cost overhead. Below, we outline the launch model for our first real-time zone powered by a Cerebras CS-2, a wafer-scale system with unmatched LLM inference throughput.

Sample Deployment — Cerebras-Powered Tesseract Zone

Item	Est. Cost	Notes
Cerebras CS-2 Appliance	$1.5M–$2.0M	Wafer-scale system; 2,600+ tokens/sec throughput
Site Power + Cooling (1 yr)	$75K–$100K	~20–25 kW draw; assumes Tier III colocation facility
Colocation + Interconnect	$100K–$150K	Includes bandwidth, rack space, and compliance controls
Initial Staff / Eng Support	$100K	Deployment assistance + hardware integration
Insurance / Ops / Legal	$50K	Basic regulatory buffer + foundational ops

Total Cluster CapEx	**~$2.0M (modeled)**	**Represents first production-grade Tesseract deployment**

Note: A single CS-2 is not a cluster in the traditional GPU sense, but its wafer-scale architecture delivers cluster-level throughput in a single box. This design minimizes interconnect latency and simplifies deployment — ideal for sovereign or metro-scale Tesseract zones.

Software and Platform Stack

Component	Est. Cost
Latency-Aware Router + Scoring Core	$80K
Developer API + Billing Interface	$50K
Model Loading, Caching, Observability	$70K
Operational Buffer / Legal / GTM	$100K
Total Launch Budget	**~$2.3M**

What This Enables

This base deployment powers:

- A full Tesseract zone serving real-time inference at <15ms regional latency
- Token throughput exceeding 2 million tokens/day under live load
- A design-partner–ready stack for voice AI, compliance-sensitive apps, and sovereign inference
- End-to-end control of chip, interconnect, and routing for performance and cost advantages

As demand grows, the zone can scale horizontally by adding additional CS-2 units or integrating with other accelerators (e.g., GPUs, Groq LPUs) via the Tesseract mesh and scheduler.

Five-Year Vision: The Global Intelligent Fabric

Where will Tesseract be in five years? We see Tesseract as the ubiquitous “dial tone” for intelligent applications globally – much like how any internet application today implicitly relies on DNS and CDNs, any AI-driven application in 2030 will rely on an inference routing network, with Tesseract leading the charge. Here’s a bold look at what that future could entail:

Globally Deployed Clusters with Native Compliance

In five years, Tesseract will have a presence in every major metro area – hundreds of points of presence – either through our own hardware or through partnerships. This forms a *planetary mesh* of AI compute. Crucially, each node in this mesh will carry metadata about jurisdiction, trust level, carbon footprint, and other attributes. When a request is routed, compliance isn’t an afterthought

– it's baked in. For example, data from an EU user not only stays in the EU, but even within the EU it might choose a specific country if that country's laws demand it. If an AI model has export restrictions (imagine a model only licensed for use in certain countries), our router will enforce that. We envision a kind of *"contract of inference"* where every request comes with a manifest of conditions (like a digital contract: e.g., "data not to be stored, model weights not to leave memory, must use renewable-powered servers, etc.") and the network honors it. This is a level of control far beyond what any cloud API today provides, and it will be necessary as AI regulation matures. Tesseract will be the go-to solution for orchestrating AI under these complex constraints – effectively an automated compliance officer built into the infrastructure.

Model-Aware Routing and Optimization

Today we route largely based on hardware and location, assuming the model is given. In five years, Tesseract could become *model-aware* in the sense of dynamically selecting or even composing models to meet an objective. For instance, a developer might just specify a task ("summarize this document under 100ms with legal accuracy") and Tesseract could decide to use a specialized legal LM if available, or a MoE (one skimming for key points, another checking facts) because that's faster than one big model. This becomes like an AI-aware query planner. With the proliferation of models (we expect an ecosystem of specialized models for different domains), a routing layer that also knows *which model* to pick is inevitable. We already see hints of this in multi-LLM ensembles and systems like HuggingFace Inference API which chooses different models – Tesseract could generalize it on a global scale. By 2030, Tesseract might host a model marketplace where requests are routed not only to hardware but to the best model for the job, all under the hood. This makes Tesseract not just a router but a meta-AI orchestrator, pushing us up the value chain.

Integration with Internet Infrastructure

We foresee Tesseract nodes integrating with ISP infrastructure, perhaps even down to the 5G basestation or Wi-Fi router level for certain quick reflex tasks. Imagine homes having smart routers that locally route trivial AI tasks to an embedded chip but connect to Tesseract's nearest edge for heavy tasks – a continuum from edge device to edge cloud to core cloud, all under one logical routing fabric. We want to be the orchestrator across that continuum. The five-year horizon could see standards emerging for AI task routing (similar to how HTTP is a standard for web data) – and Tesseract should influence these standards by sheer presence and innovation. We might work with telcos such as Verizon or Vodafone to embed Tesseract in their edge computing offerings, effectively making the telecom network itself "AI-aware." If successful, this leads to ultra-low latency (sub-10ms) inference for things like AR glasses, autonomous vehicles, etc., because tasks are being served literally from the nearest street corner server.

AI Safety and Governance as a Service

With great power comes responsibility – by being the layer through which intelligent decisions flow, Tesseract is well-positioned to also implement governance policies. In five years, there will likely be regulations requiring logging of AI decisions, detection of bias, filtering of certain content,

etc. Tesseract can bake in these governance tools: e.g., automatically route potentially toxic outputs through an extra moderation model or provide kill-switch and audit logs to authorities for critical applications. While this might not be a core selling point now, by 2030 this could be mandatory. Being ahead of that, Tesseract could differentiate as the trustworthy routing layer that enterprises and governments are comfortable relying on because it has all the necessary guardrails. Basically, today we enforce latency/compliance; in future, we may also enforce ethical AI policies as configured by the customer (like "don't allow this model to output personal data" – our router could split out personal data detection as a step in the pipeline).

Unified Analytics and Optimization

At a global scale, Tesseract will accumulate a goldmine of data about how and where AI is used: latency stats, error rates, usage patterns by region and industry. We will leverage this to continuously optimize. Perhaps through reinforcement learning, the routing algorithms themselves get smarter – learning, for example, that certain types of queries are best handled by a particular setup. We could offer analytics to customers ("your app is 30% faster in Europe than Asia due to infrastructure differences – here's how upgrading could help"). Also, through economies of scale, we might negotiate better deals with hardware providers, essentially becoming the "traffic exchange" for AI compute capacity globally.

In five years, we expect competitors (maybe big cloud companies or startups) to also have attempted their versions of intelligent routing. By moving first and aggressively, Tesseract aims to establish network effects: the more workloads we route, the more data and partnerships we have, making our service better and more comprehensive, which attracts more workloads – a virtuous cycle. If we execute well, Tesseract could become as integral to AI deployment as operating systems are to computers. In fact, that's an apt analogy: just as an OS manages CPU, memory, and I/O for applications, Tesseract manages distributed compute, data locality, and latency for AI applications. It's the distributed OS for AI inference.

From a business perspective, in five years we see a healthy revenue model with diverse streams: usage fees from thousands of developers, enterprise contracts from the biggest companies (with SLAs and dedicated support), and possibly interconnect fees from cloud/hardware partners. If we succeed, Tesseract will be serving not just tech companies but also powering AI in retail (store cameras using local inference via Tesseract), in automobiles (car queries edge AI via Tesseract when needed), in education (language learning apps optimizing responses per student location), and countless other scenarios.

One could imagine a future press release: *"Cloudflare and OpenAI Invest in Tesseract as the Standard for Global AI Routing"*, where industry leaders back us because it benefits the whole ecosystem to have a neutral, highly optimized layer. Our vision is large but concrete: by solving the hard problem of real-time, compliance-aware inference distribution, we become an enabling layer for *everything* from tiny microservices to nation-scale AI deployments.

Tesseract's journey starts with tackling the here-and-now problem of latency and heterogeneity, but it will end (five years hence) with Tesseract woven into the fabric of the AI-powered world –

quietly ensuring that every AI interaction, no matter where it originates or what it needs, is routed to the best possible outcome. We believe this layer is indeed *inevitable*, and by acting now, we intend to shape it in a direction that benefits all: faster responses, smarter use of resources, and AI services that users can trust with both their data and their time.

Conclusion

AI is crossing the threshold from an offline, backroom utility to an always-on, interactive intelligence that permeates our lives. This transition demands an infrastructure upgrade – one that treats latency, locality, and load as first-class considerations rather than afterthoughts. Tesseract is our answer to that demand: a latency-aware, chip-agnostic global router that ensures AI models deliver results in the right place, at the right time, on the right hardware.

In this whitepaper, we argued from first principles why such a layer is needed now. The physics of light speed and the surging scale of AI usage are on a collision course; without a new approach, AI experiences will stagnate under the weight of slow responses and siloed deployments. We dissected the technical challenges – from coordinating heterogeneous silicon to enforcing compliance constraints – and showed that while the problem is hard, it is surmountable with a thoughtful architecture. We drew parallels to the evolution of other tech stacks (CDNs, cloud, payment networks) to illustrate that the pattern is familiar: when a resource (in this case, intelligence) needs to be delivered globally in real-time, a routing and caching layer becomes indispensable.

Tesseract is not a speculative concept; it is an engineered system already beginning to take its first steps. It does not merely augment cloud providers or model developers – it extends them, and where necessary, replaces them. Tesseract acts as the connective tissue of AI infrastructure, but unlike traditional routers or orchestrators, it also owns part of the nervous system. One might think of Tesseract as the “traffic controller” of the AI era: directing the flow of requests to where compute and data meet most optimally – whether that’s in our own clusters or in partner clouds. Abstracting this complexity behind a simple API lets developers focus on product and experience, freeing them from the knots of infrastructure decisions – while still delivering performance, sovereignty, and control.

We compared cutting-edge AI hardware to emphasize a key point: raw performance gains are available (as evidenced by Cerebras, Groq, etc.), but harnessing them requires intelligence above the hardware – precisely what Tesseract provides. It ensures that a 20× faster chip actually translates to a 20× better user experience by routing work to it appropriately. Likewise, it ensures that a user in a far-flung region is not an afterthought, but served from a nearby node if one exists, or otherwise informed of expected latency – turning unpredictable lag into managed service quality.

Our five-year vision for Tesseract is ambitious: we foresee it as a pervasive layer, akin to the internet’s own backbone, that optimizes and governs AI computations worldwide. It will continuously learn from the vast number of inferences it routes, becoming smarter and more

efficient, to the point where it can anticipate needs and pre-position resources (just as CDNs pre-fetch content). We will support not just latency goals but also cost-optimization goals, carbon footprint goals (imagine routing to a data center with surplus renewable energy), and beyond. Tesseract's routing fabric could even facilitate collaborations – e.g., two companies could allow certain queries to be mutually routed for federated learning without sharing raw data.

In building Tesseract, we are guided by a core belief: AI should be as accessible and responsive as electricity – available when and where you need it, invisibly delivered through a complex grid that users never have to think about. The power grid of the AI age is what Tesseract aspires to be. We believe that by solving the inference routing problem, we unlock an era of truly distributed intelligence – where no user is too distant, no regulation is an obstacle, and no hardware is left underutilized.

To the engineers, researchers, and business leaders reading this: we invite you to join us in this journey. The challenges are non-trivial, but the rewards – in user satisfaction, in system efficiency, in societal impact – are profound.

In summary, Tesseract is the latency abstraction layer for the coming wave of intelligent applications. It is the missing piece that completes the AI deployment puzzle. By leveraging physics-savvy design, chip-level insights, and a bold globally distributed architecture, we make the case that Tesseract isn't just inevitable – it's imminent. The world will demand AI that is faster, closer, and more accountable, and when it does, Tesseract will be there as the *global router for intelligent workloads*.

Let's build the future of real-time AI inference, together. The network is the computer, and with Tesseract, the network is finally an intelligent computer.

We are now seeking visionary partners and investors to help deploy the first clusters of Tesseract, starting in North America and Europe. If you're building the future of AI, join us.

References

[i] https://venturebeat.com/ai/meta-unleashes-llama-api-running-18x-faster-than-openai-cerebras-partnership-delivers-2600-tokens-per-second/#:~:text=The%20announcement%2C%20made%20at%20Meta%E2%80%99s,billions%20to%20power%20their%20applications

[ii] https://www.investors.com/news/technology/ai-stocks-market-shifting-to-inferencing-from-training/

[iii] https://medium.com/@laowang_journey/comparing-ai-hardware-architectures-sambanova-groq-cerebras-vs-nvidia-gpus-broadcom-asics-2327631c468e

[iv] https://www.businesswire.com/news/home/20250409886691/en/Cerebras-Launches-Worlds-Fastest-Inference-for-Meta-Llama-4

[v] https://www.businesswire.com/news/home/20250429462666/en/Meta-Collaborates-with-Cerebras-to-Drive-Fast-Inference-for-Developers-in-New-Llama-API

[vi] https://blog.telegeography.com/its-time-to-learn-about-latency#:~:text=For%20long,of%20light%20through%20a%20vacuum

[vii] https://www.atlanticcouncil.org/in-depth-research-reports/issue-brief/sovereign-remedies-between-ai-autonomy-and-control/#:~:text=Sovereign%20AI%20as%20a%20phenomenon,energy%20infrastructure%2C%20and%20workflow%20management

[viii] https://arxiv.org/abs/2405.19213

[ix] https://blog.telegeography.com/all-about-cloud-regions-zones-and-on-ramps

[x] https://arxiv.org/abs/2405.00790

[xi] https://arxiv.org/abs/2503.20074

[xii] https://grpc.io/

[xiii] https://www.auvik.com/franklyit/blog/what-is-quic-protocol/
[xiv] https://developer.nvidia.com/blog/introducing-nvidia-dynamo-a-low-latency-distributed-inference-framework-for-scaling-reasoning-ai-models/#:~:text=Image%3A%20Diagram%20of%20NVIDIA%20Dynamo,agnostic%20multinode%20data%20transfer
[xv] https://groq.com/meta-and-groq-continue-to-build-open-source-developer-ecosystem-as-llama-3-2-launches/
[xvi] https://www.akamai.com/blog/cloud/distributed-ai-inferencing-next-generation-of-computing#:~:text=latency%2C%20cost%2C%20and%20scalability,in%20various%20contexts%20for%20decades
[xvii] https://www.cloudflare.com/learning/performance/what-is-http3/
[xviii] https://www.tigera.io/learn/guides/kubernetes-security/kubernetes-federation/
[xix] https://virtual-kubelet.io/
[xx] https://etcd.io/
[xxi] https://research.google/pubs/large-scale-cluster-management-at-google-with-borg/
[xxii] https://www.ray.io/
[xxiii] https://flyte.org/
[xxiv] https://developer.nvidia.com/blog/introducing-nvidia-dynamo-a-low-latency-distributed-inference-framework-for-scaling-reasoning-ai-models/#:~:text=Image%3A%20Diagram%20of%20NVIDIA%20Dynamo,NVIDIA%20Dynamo%20architecture
[xxv] https://softwareengineering.stackexchange.com/questions/444704/multi-tenant-multi-region-saas-with-per-customer-subdomain?utm_source
[xxvi] https://blog.cloudflare.com/cloudflare-for-ai-supporting-ai-adoption-at-scale-with-a-security-first-approach/#:~:text=Cloudflare%20for%20AI%3A%20supporting%20AI,creators%20adopt%2C%20deploy%2C%20and
[xxvii] https://www.vastdata.com/blog/sovereign-ai-demands-new-data-doctrine
[xxviii] https://www.fierce-network.com/wireless/get-ready-telco-genai-network-operations#:~:text=%2A%20Mavenir%C2%A0has%20developed%20a%20GenAI%C2%A0co,initial%20operations%20in%20this%20area