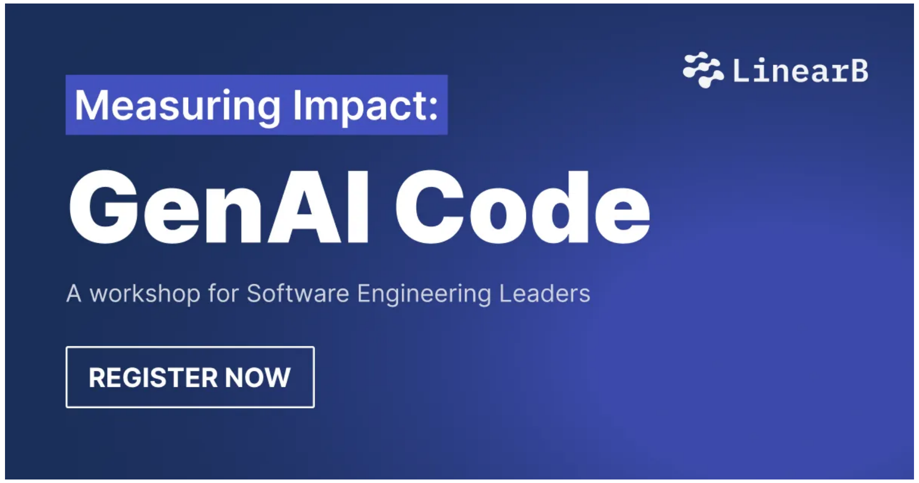
How Discord Serves 15-Million Users on One Server

BYTEBYTEGO
JAN 9, 2024

496 7 12 Share

Measuring GenAI Code's Impact: Free Workshop (Sponsored)



How is GenAI impacting software development?

Join LinearB and ThoughtWorks' Global Lead for AI Software Delivery to explore the metrics showing AI's impact, unpack best practices for leveraging AI in software development, and measure the ROI of your own GenAI initiative.

This workshop includes:

📊 **Data insights** from LinearB's new GenAI Impact Report

🗣️ **Case studies** into how others are already doing it

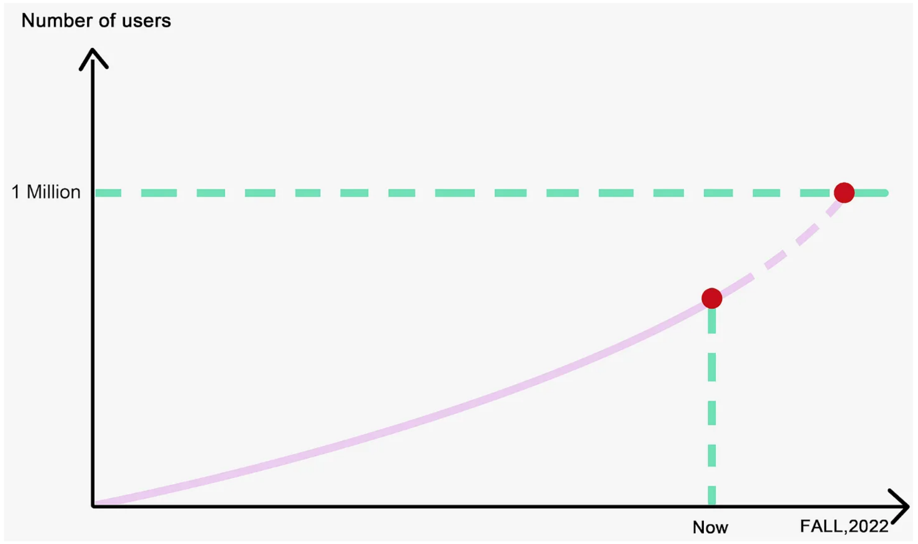
🔎 **Impact Measures:** adoption, benefits & risk metrics

✅ **Live demo:** How you can measure the impact of your GenAI initiative today

Join the conversation on January 25th or 30th.

In early summer 2022, the Discord operations team noticed unusually high activity on their dashboards. They thought it was a bot attack, but it was legitimate traffic from MidJourney - a new, fast-growing community for generating AI images from text prompts.

To use MidJourney, you need a Discord account. Most MidJourney users join one main Discord server. This server grew so quickly that it soon hit Discord's old limit of around 1 million users per server.



Discord risked losing this important new community if they didn't act fast.

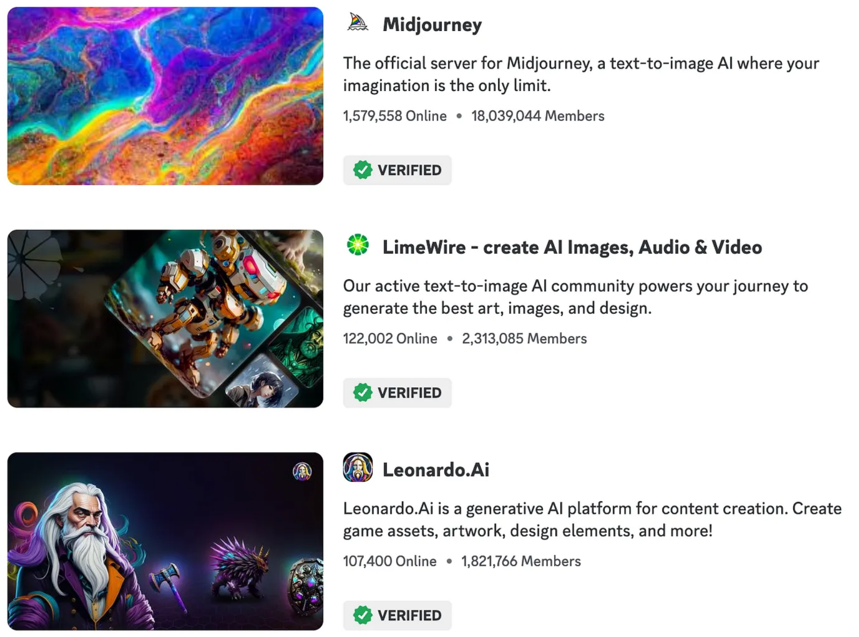
This is the story of how the Discord team creatively solved this challenge. They found ways to dramatically expand what their infrastructure could handle - keeping the thriving MidJourney community active on Discord.

What is Discord?

Discord is a popular chat app used by hundreds of millions to connect. Originally for gamers, now all types of communities use it - from hiking clubs to study groups to large gaming communities.

In Discord, a "server" hosts a community. It has chat channels to discuss topics chosen by the server owner.

Internally, Discord calls these servers "guilds" - so we'll use that term going forward.



Midjourney

The official server for Midjourney, a text-to-image AI where your imagination is the only limit.

1,579,558 Online • 18,039,044 Members

VERIFIED

LimeWire - create AI Images, Audio & Video

Our active text-to-image AI community powers your journey to generate the best art, images, and design.

122,002 Online • 2,313,085 Members

VERIFIED

Leonardo.Ai

Leonardo.Ai is a generative AI platform for content creation. Create game assets, artwork, design elements, and more!

107,400 Online • 1,821,766 Members

VERIFIED

Largest Discord Guilds (image source: Discord)

Before MidJourney, the biggest guilds had around 1 million members - huge gaming communities like Roblox and Fortnite.

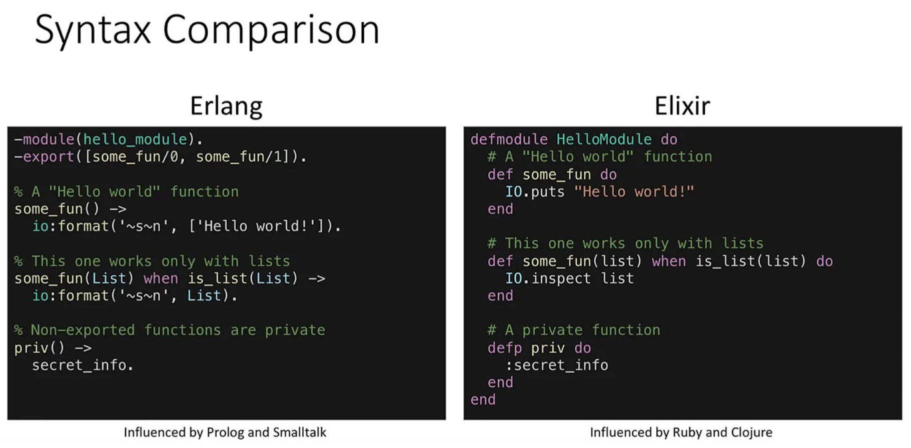
The Discord engineering team thought 1 million members was very close to the maximum a guild could handle. Let's explore why - but first, some quick background on the technologies powering Discord.

Introduction to BEAM and Elixir

Discord's real-time messaging backend is built with Elixir. Elixir runs on the BEAM virtual machine. BEAM was created for Erlang - a language optimized for large real-time systems requiring rock-solid reliability and uptime.

A key capability BEAM provides is extremely lightweight parallel processes. This enables a single server to efficiently run tens or hundreds of thousands of processes concurrently.

Elixir brings friendlier, Ruby-inspired syntax to the battle-tested foundation of BEAM. Combined they make it much easier to program massively scalable, fault-tolerant systems.



Syntax Comparison

Erlang

```erlang
-module(hello_module).
-export([some_fun/0, some_fun/1]).

% A "Hello world" function
some_fun() ->
  io:format('~s~n', ['Hello world!']).

% This one works only with lists
some_fun(List) when is_list(List) ->
  io:format('~s~n', List).

% Non-exported functions are private
priv() ->
  secret_info.
```

Influenced by Prolog and Smalltalk

Elixir

```elixir
defmodule HelloModule do
  # A "Hello world" function
  def some_fun do
    IO.puts "Hello world!"
  end

  # This one works only with lists
  def some_fun(list) when is_list(list) do
    IO.inspect list
  end

  # A private function
  defp priv do
    :secret_info
  end
end
```

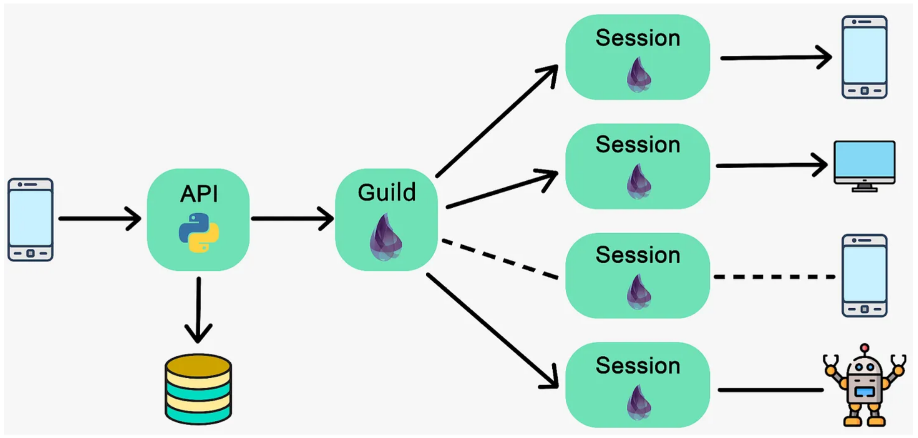
Influenced by Ruby and Clojure

Code snippets comparing Erlang and Elixir syntax (image source: elixirforum)

So by leveraging BEAM's lightweight processes, the Elixir code powering Discord can "fan out" messages to hundreds of thousands of users around the world concurrently. However, limits emerge as communities grow larger.

Discord's Real-time Infrastructure

As mentioned, Discord handles all real-time communication using Elixir processes on the highly concurrent BEAM virtual machine.



The path of a message through Discord's real-time infra to other users and bots in a guild (Source: Discord eng blog)

Internally, each Discord community is called a "guild". A dedicated Elixir "guild process" handles coordination and routing for each guild. This tracks all connected users to the guild.

Every online user has a separate Elixir "session process". When the guild process gets a new message, event, or update, it fans out this information to the relevant session processes. These session processes then push the update over WebSocket to the Discord clients.

This architecture provides a cost-effective way to handle millions of active guilds across a large pool of Linux servers in Discord's cloud infrastructure.

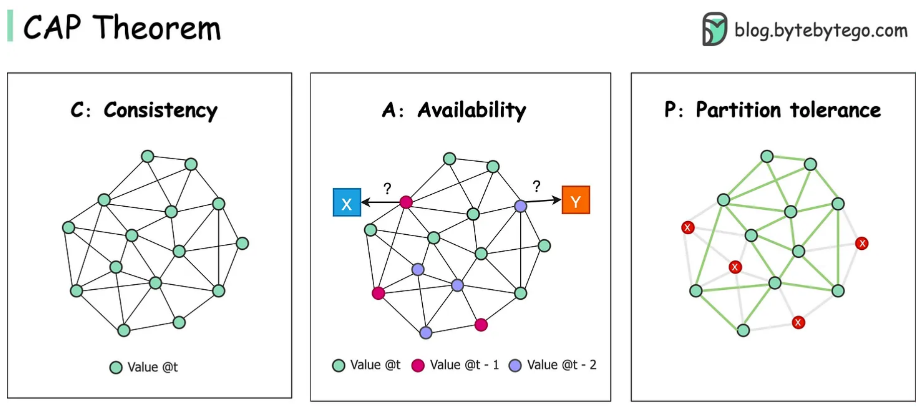
However, scaling limits emerge as guilds grow larger. Distributing messages and events to more users creates exponentially more work. Larger guilds also have more activity to distribute.

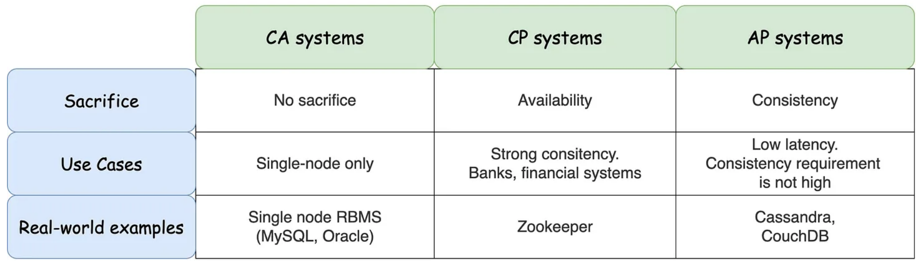
So the guild process load grows much faster as its number of users increases. BEAM helps tremendously, but there's only so much one BEAM process can handle.

This is why Discord thought breaking 1 million concurrent users per guild would be very difficult.

Latest articles

If you're not a paid subscriber, here's what you missed this month.





	CA systems	CP systems	AP systems
Sacrifice	No sacrifice	Availability	Consistency
Use Cases	Single-node only	Strong consitency. Banks, financial systems	Low latency. Consistency requirement is not high
Real-world examples	Single node RBMS (MySQL, Oracle)	Zookeeper	Cassandra, CouchDB

1. Netflix: What Happens When You Press Play?
2. 6 More Microservices Interview Questions
3. 7 Microservices Interview Questions
4. Why the Internet Is Both Robust and Fragile
5. Unlock Highly Relevant Search with AI

To receive all the full articles and support ByteByteGo, consider subscribing:

MaxJourney

With that background established, let's return to the main story. Facing a scaling crisis from Midjourney's runaway growth, Discord formed a small team of senior engineers to dig into the problems. This team was called MaxJourney.

Here's what they accomplished.

Detailed Performance Profiling

Understanding where systems spend time and memory is critical before improving them. The team used various profiling techniques to analyze guild process performance.

The simplest was sampling stack traces to reveal expensive operations. This quickly highlights issues without much effort. However, richer data was needed.

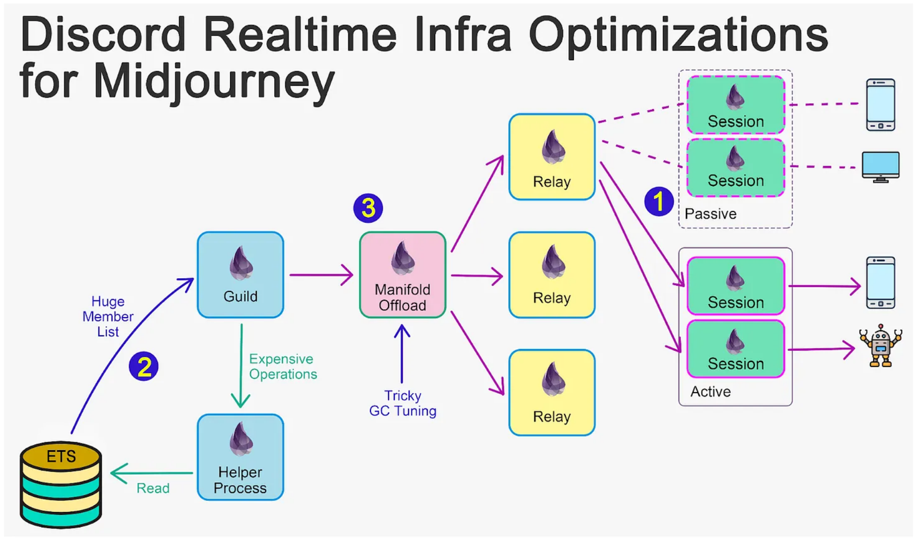
So they instrumented the event loop to record metrics on each message type. This included frequency, min/max/average processing times. This analysis revealed the costliest operations to optimize. Cheap ones could be ignored.

Memory usage was also examined, since it impacts hardware needs and garbage collection throughput.

To estimate sizes of large data structures reasonably quickly, a helper library was built to sample maps and lists. It avoids fully traversing all elements.

This sampling revealed memory-intensive fields to refactor.

Armed with visibility into these time and memory hotspots, the team could now systematically target optimizations to rewrite inefficient code.



Passive Sessions - Avoiding Unnecessary Work

The team's first optimization was reducing unnecessary work. They realized the client app did not always need every update for guilds that users were not actively viewing in the app's foreground.

So they implemented "passive" connections for those guilds. Passive connections skip processing and data transmission until the user opens the guild.

Over 90% of the user-guild connections became passive for large servers. This cut required work by 90%, greatly reducing load.

However, MidJourney kept growing. So this alone was not enough.

Optimizing Relays - Distributing Fanout Across Machines

Relays already existed to split fanout work across BEAM processes for scaling. Relays are only enabled for large guilds, where they maintain session connections on behalf of the guild.

Each relay handles fanout and permissions for up to 15,000 users. This allowed leverage more BEAM processes to serve large guilds.

Originally, relays duplicated full member lists. It was simple to implement, but for massive guilds with millions of members, dozens of copied lists wasted huge amount of RAM.

Also, creating relays stalled massive guilds for seconds while serializing and transmitting member data.

So the team optimized relays to track just the tiny subset of members needed per relay.

Keeping Servers Responsive

In addition to overall throughput, ensuring low latency was critical. So the team analyzed operations with high per-call duration, beyond just total time.

Worker Processes and ETS

Key culprits were member iterations taking seconds, blocking guilds. The solution was worker processes to offload these. Workers leverage ETS, an in-memory database for fast inter-BEAM-process data sharing.

Members were stored in ETS, with recent changes in the guild's heap. This hybrid model kept the guild's memory small.

For slow tasks, workers are spawned to run them asynchronously using the shared ETS data, freeing the guild to continue handling messages.

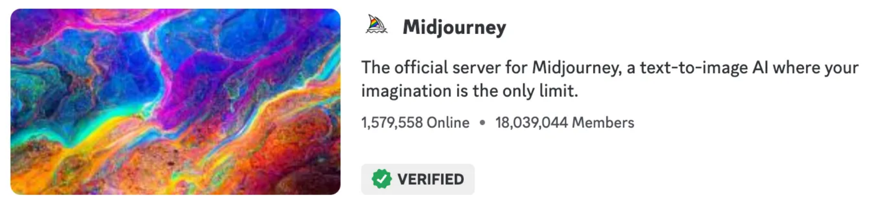
An example slow task is handling guild migration between machines. Copying state from the old guild process to the new process normally stalls the old one for minutes. But offloading this to a worker avoids blocking the old guild process from handling incoming messages.

Manifold Offload

Another idea was offloading fanout from guilds to separate "sender" processes, further reducing guild workload and insulating the guild processes from network backpressure.

However, this unexpectedly tanked performance due to pathological garbage collection. Analysis showed it was triggered by freeing small memory outside the heap.

Tuning the virtual binary heap size fixed this. Now offload could be enabled, significantly improving throughput.



Midjourney

The official server for Midjourney, a text-to-image AI where your imagination is the only limit.

1,579,558 Online • 18,039,044 Members

VERIFIED

Latest MidJourney Member Count - 18 Million! (Image source: Discord servers - home)

Through systematic optimization, the MaxJourney team achieved the seemingly impossible - expanding guild capacity 15x to keep MidJourney thriving on Discord.

References

[1] Maxjourney: Pushing Discord's Limits with a Million+ Online Users in a Single Server

Using Rust to Scale Elixir for 11 Million Concurrent Users

[2] How Discord Scaled Elixir to 5,000,000 Concurrent Users

[3] Discord Developer Portal — Documentation — Guild

[4] GitHub - discord/manifold: Fast batch message passing between nodes for Erlang/Elixir.

[5] BEAM (Erlang virtual machine) - Wikipedia

[6] Erlang's virtual machine, the BEAM

[7] Introduction — Elixir v1.16.0

496 Likes · 12 Restacks