Paddler  

Self-hosted Large Language Models at a scale

github.com/distantmagic/paddler

Where are we now when it comes to LLMOps?

- Infancy stage 👶
- Still figuring out what works and what doesn’t
- Businesses are yearning for stability 🏢
- Too early to standardize anything

What is Paddler? 🏓🦙

- Load balancer custom-tailored for llama.cpp
- Provides scalability
- Aims to provide more stability in the ecosystem

How do you end up with a custom load balancer?

Follow those few simple steps ╰┈➤

I want to host an open-source Large Language Model. How?

- Let’s start with a single host
- VLLM, llama.cpp, Ollama, something else? 🤔
- Why llama.cpp?

Ok, I have my model hosted. Something does not add up. 😳

- 7B-12B models require 4-10 GB of VRAM
- We can get up to ~200 tokens/sec on T100
- T100 costs about ~$300/month
- Does that mean we can serve poor experience to ~8 users for ~$300/month? 😱

Back to the 90s

- We are again in a situation where a single server can handle a few dozen users at most
- We need scalability from the start
- Feels like we are operating on servers from the 90s again

Looks like I need load balancing.

- Ok, but what balancing algorithm should I use?

Option 1: Round Robin

- Not reliable in an environment where responses can take variable amounts of time

Option 2: Least Resources Used

- GPU usage stays almost the same no matter how many users use the system
- The reason lies in the specifics of continuous batching
- Not reliable

Option 3: Least Connections

- Better than the other options
- Resources are limited
- We need to use application-aware features to handle the load on top of it

Option 4: Application-aware + least connections + scaling = Paddler

- Knows exactly when a server has no available resources; does not bother it further
- Can keep requests on hold without dropping them
- Resilient, can handle llama.cpp instances going down
- Supports StatsD metrics for autoscaling

Paddler Dashboard

127.0.0.1:8095/dashboard

Paddler 🏓

Registered Agents

Name	Issues	Llama.cpp address	Last update	Idle slots	Processing slots	
wohoo	None	127.0.0.1:8081	11/20/2024, 9:38:59 PM	4	0	
George	None	127.0.0.1:8082	11/20/2024, 9:38:56 PM	4	0	

current tick: 19

Paddler Load Balancer

llama.cpp health statuses are aggregated

llama.cpp 1

```
{
  "status": "ok",
  "slots_idle": 7,
  "slots_processing": 1
}
```

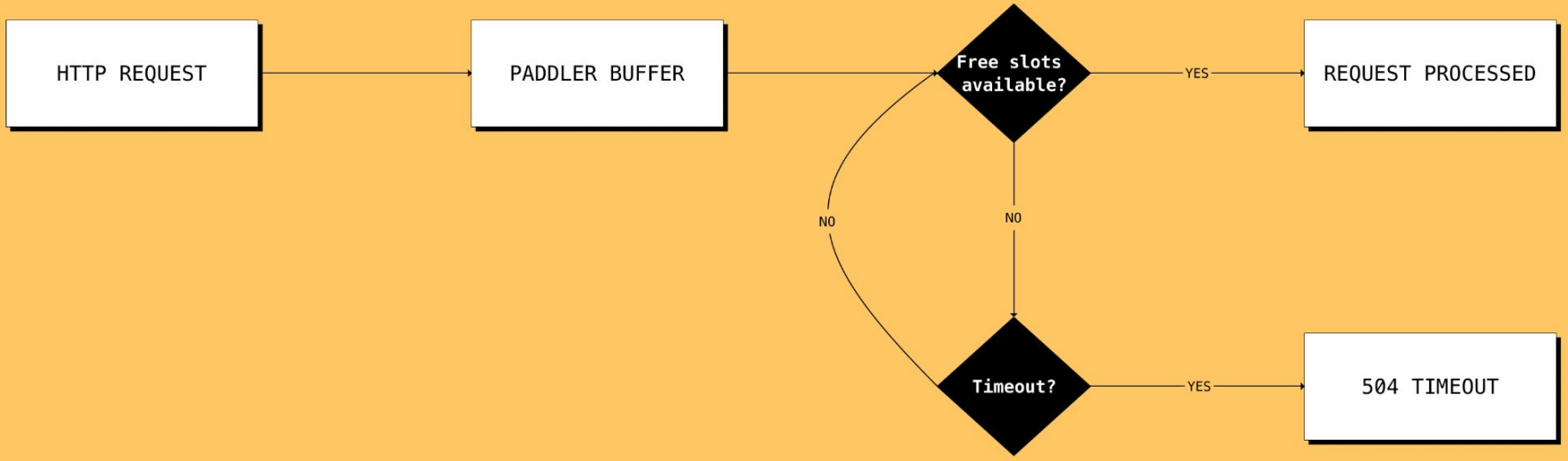
llama.cpp 2

```
{
  "status": "ok",
  "slots_idle": 8,
  "slots_processing": 0
}
```

Paddler balancer

```
{
  "status": "ok",
  "slots_idle": 15,
  "slots_processing": 1
}
```

Paddler Buffered Requests



Paddler is lightweight 

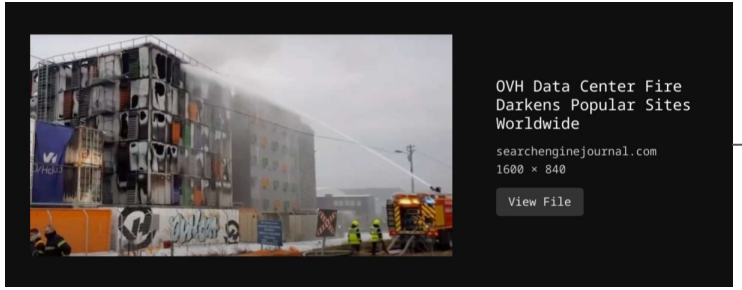
- Based on Pingora framework
- Adds llama.cpp instances health checks

Scalability

- Tree-like cascading structure
- Similar to how you handle other types of load balancers
- Useful for staging environments

High Availability

- Put haproxy in front of Paddler, add a standby host in a backup region



Redirect traffic to

Paddler vs… 

- Paddler is not really made to compete directly with anyone
- But how does it compare to:

Paddler vs llama.cpp RPC

- It makes sense to use both at the same time
- Paddler adds dynamic setup and resiliency
- Reverse proxy vs forward proxy

Future plans

- Bundling llama.cpp
- Support Ollama and other runners
- Inference API with semantic versioning for stability

Shout out to the contributors!

- Luiz Miguel, https://github.com/Propfend
  ^ contributed console dashboard, Paddler supervisor
- ScottMcNaught, https://github.com/ScottMcNaught
  ^ patiently helped me to debug lots of stuff
- zamazan4ik, https://github.com/zamazan4ik
  ^ great optimization tips

Thank you!

Join our community:

- GitHub: https://github.com/distantmagic/paddler
- Discord: https://discord.gg/kysUzFqSCK

Reach out to me (Mateusz Charytoniuk):

- GitHub: https://github.com/mcharytoniuk
- LinkedIn: https://www.linkedin.com/in/mateusz-charytoniuk/