Parte 2

Rust na vida real

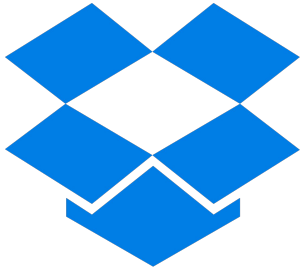
Onde?

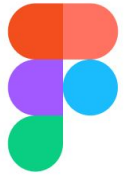
MANGA sucks











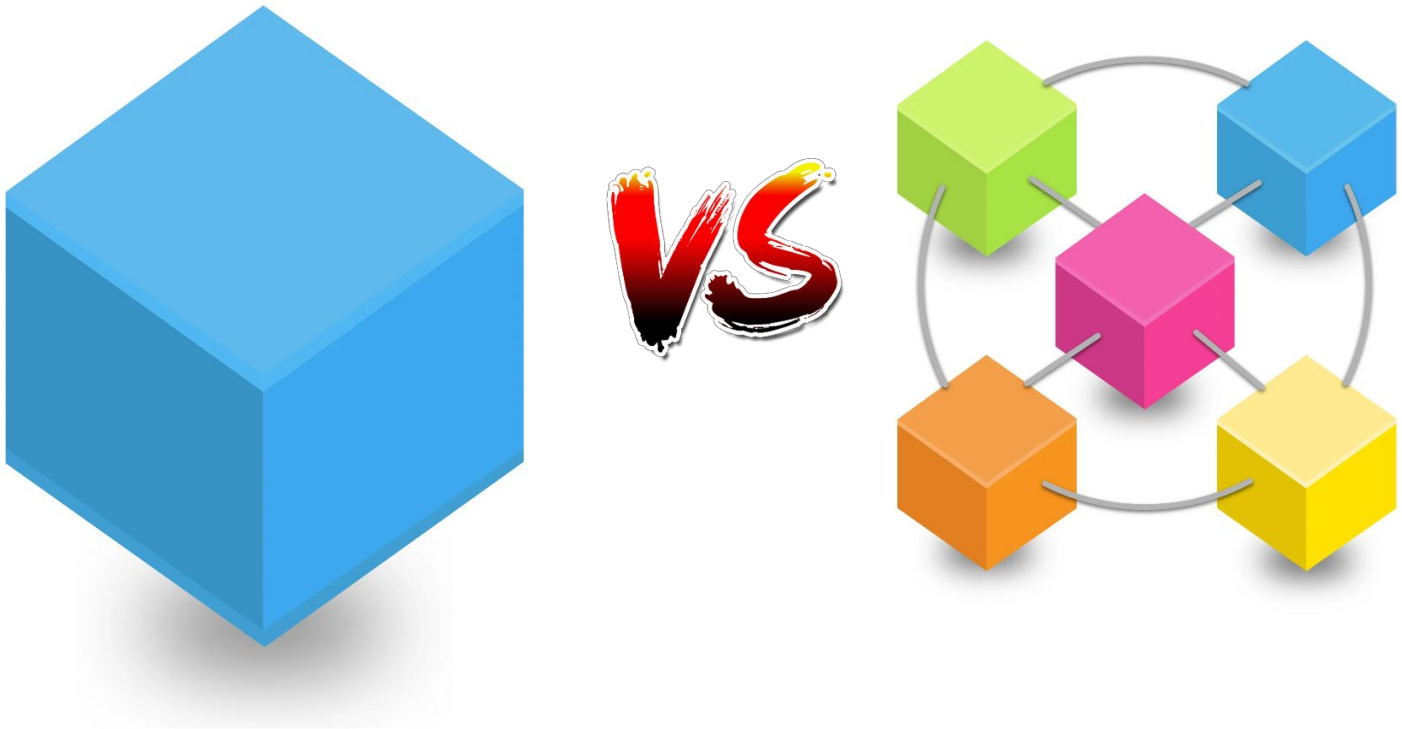

Onde no Brasil? 

Braza sucks

Referência: https://github.com/rust-br/eu-uso-rust

Onde?

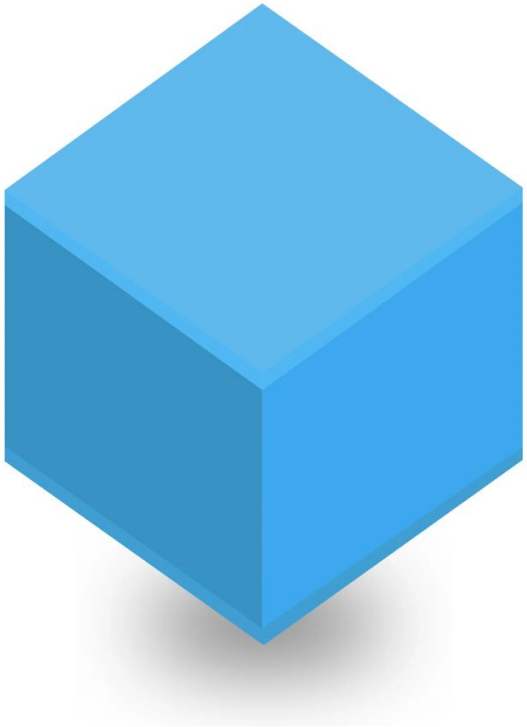
MANGA sucks



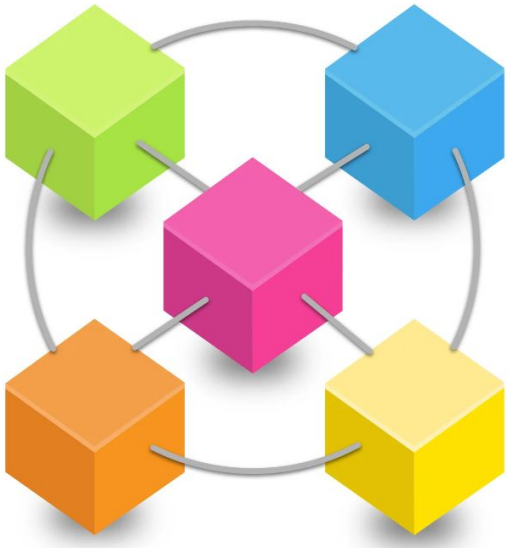
Onde?

MANGA sucks

Monolito



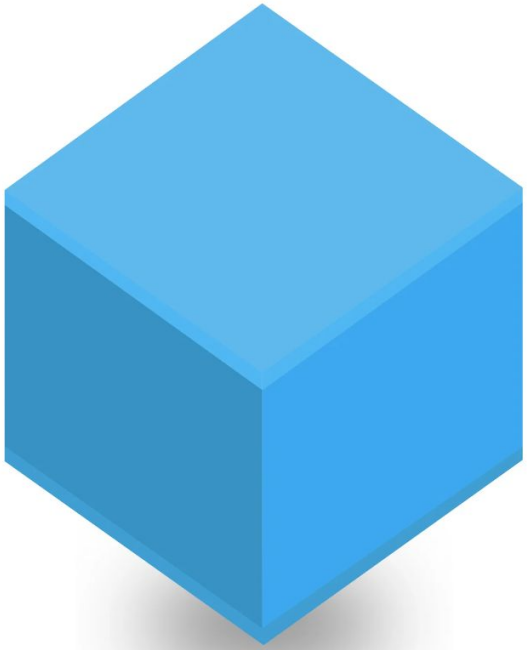
Microsserviços



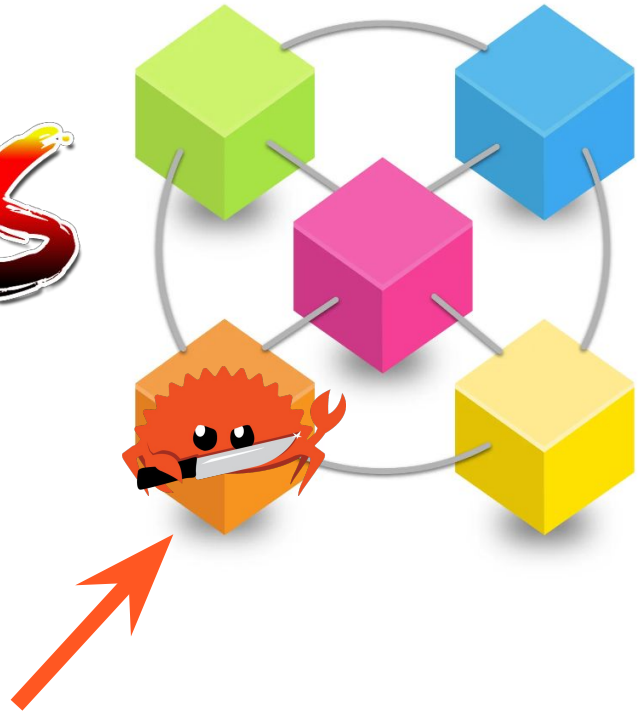
Onde?

MANGA sucks

Monolito



Microsserviços



Onde?

MANGA sucks

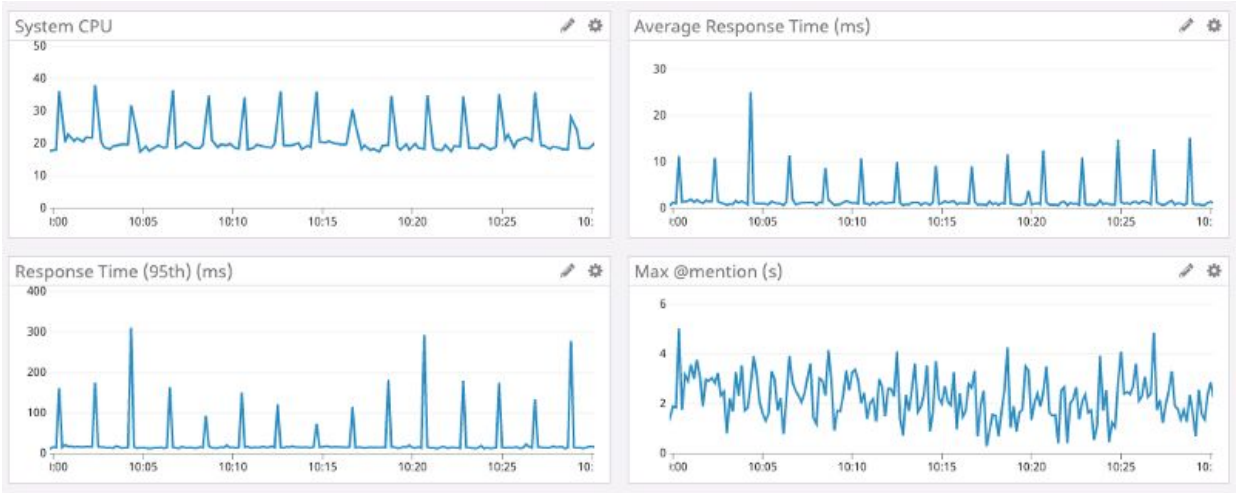
Onde?

MANGA sucks

- Dentro da arquitetura de microsserviços, Rust é usado nos serviços onde o tempo de resposta e uso de memória devem ser baixos, e a segurança de memória é necessária. Ou, ainda, quando não se deseja utilizar garbage collection.
- Ou seja, fazer uma API em Rust não se justifica muito. Fazer um serviço de processamento de dados que a API chama, sim.

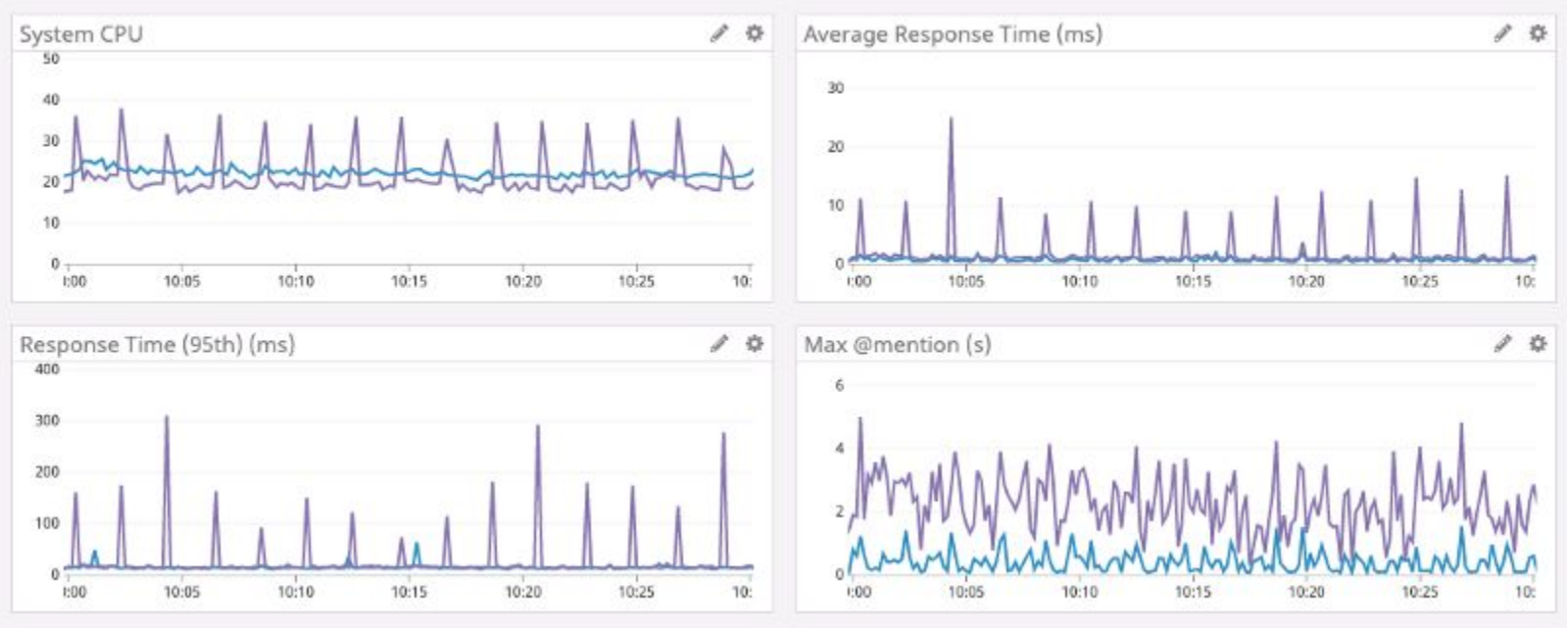
Onde? Exemplo ótimo: Discord

- O Discord utiliza Rust no seu serviço de “Read States”, responsável por rastrear quais mensagens foram lidas por cada usuário da plataforma.
- **Mais de 140 milhões de usuários mensais ativos. Mais de 300 milhões de usuários registrados.**
- Ou seja, precisa ser rápido, e não pode falhar. Lentidão e falhas implicam em perdas significativas.
- Mas isso não significa que eles precisavam usar Rust. Na verdade, eles usavam Go! Contudo...



Onde? Exemplo ótimo: Discord

- A mudança de Go para Rust, que não tem garbage collector, implicou em ganhos muito significativos para o Discord.



Mas...

VOCÊ
NÃO
É
O
DISCORD

Quando?

Você não é a Google!

- Em um contexto de atendimento de poucos usuários, com uma infraestrutura simples, é difícil justificar sequer uma arquitetura de microsserviços. Se microsserviços fosse solução para tudo, todo mundo usava MINIX.
- Mesmo que você atenda muita gente...
  - Não precisa de MUITA eficiência? JavaScript, Java, Python, Ruby...
  - Precisa de eficiência? C/C++
  - Precisa de eficiência e segurança de memória? Go
- ...precisa de **eficiência, segurança de memória e eficiência de memória?** Aí sim, Rust.

Quando?

Você não é a Google!

- **De modo geral, use Rust quando houver o combo:**
  1. Minha aplicação precisa ser rápida
  2. Minha aplicação precisa ter segurança de memória
  3. Minha aplicação precisa usar eficientemente a memória

     ou

     minha aplicação não pode contar com um garbage collector

- Caso contrário, o custo de desenvolvimento e treinamento dificilmente justificará o uso da linguagem Rust.

Onde não... ainda?

- Onde ocorre o combo, mas ainda não se justifica: onde há sistemas legado muito bem estabelecidos — ex.: bancos, sistemas médicos, nos quais uma falha seria catastrófica.
- Lembre-se que a linguagem, e tudo decorrente dela, teve sua primeira versão funcional divulgada há menos de 10 anos.

Onde não... ainda?

- Areweyet?

Name	Url	Description	Owner
Are we async yet?	https://areweasyncyet.rs/	Asynchronous I/O in Rust	Web Services WG
Are we audio yet?	https://areweaudioyet.com/	Audio related development in Rust	RustAudio
Are we Chrome yet?	http://arewechromeyet.com/	Parity-Chrome bugs	??
Are we compressed yet?	https://arewecompressedyet.com/	Video codec quality	Xiph.Org Foundation
Are we distributed yet?	https://arewedistributedyet.com/	Peer-to-peer web experience	https://github.com/arewedistributedyet/arewedistributedyet
Are we everyone yet?	http://areweeveryoneyet.org/	Total, Employees, and Volunteer contributors per Firefox release.	Mike Hoye
Are we fast yet?	http://arewefastyet.com/	Tracking performance of popular JavaScript engines	JavaScript team
Are we fun yet?	http://arewefunyet.com/	Web platform game development	Martin Best
Are we game yet?	http://arewegameyet.com/	Rust game development	doppioslash
Are we GUI yet?	http://areweguiyet.com/	Rust GUI development	https://github.com/areweguiyet/areweguiyet

Referências (empresas)

Meta

1. A brief history of Rust at Facebook
2. Programming languages endorsed for server-side use at Meta

Amazon

1. Why AWS loves Rust, and how we'd like to help

Microsoft

1. https://github.com/krustlet/krustlet
2. In Rust We Trust: Microsoft Azure CTO shuns C and C++

(entre outros projetos internos, em especial de laboratórios – alguns artigos dão a entender que eles também usam internamente no Azure)

Restante: Rust. Usuários em produção.

Onde mais?

- Não somente em arquitetura de microsserviços, mas também em:
  - Serviços de sistemas operacionais (daemons)
  - Software de servidor
  - Processamento interno de dados
- Lembre-se que, mesmo nesses casos, deve-se considerar os custos e se o uso de Rust realmente de justifica

Mas....

E se eu **QUISER** usar Rust?
*It's **blazingly** fast...*
and memory-efficient!

- Use.
- Seja feliz.

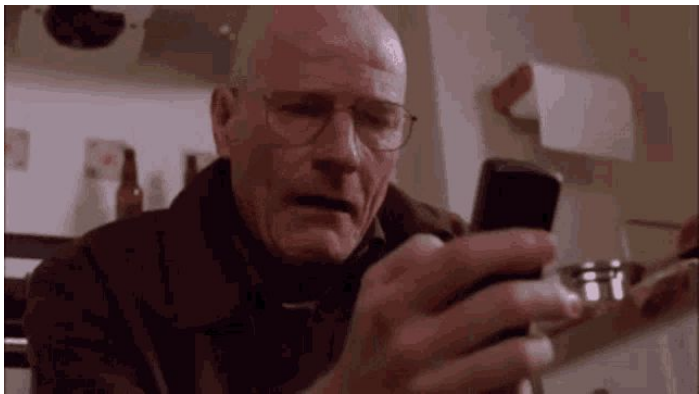
Primeira Milestone.

Exemplo 1: o pior DAW do mundo

- Ferramenta de composição musical
- Sistemas muito grandes costumam ser quebrados em programas menores, ao invés de um monolito gigante
- Justifica-se o uso de Rust pela:
  - Eficiência (baixo tempo de resposta)
  - Segurança (o serviço precisa estar disponível sempre que o cliente o puder requisitar)
  - Baixo uso de memória (assim pode-se concentrar o uso de RAM em componentes mais robustos, como sintetizadores virtuais)
  - Bônus: rodio é muito legal

Exemplo 2: serviço caçador de comunistas

- Oh não! A NSA integrou nos principais aplicativos de comunicação um serviço espião para caçar ameaças a democracia!
- O serviço detecta alguns padrões de mensagens que indicam possíveis comunistas, incluindo imagens.
- O uso de Rust justifica-se pela:
- Necessidade de eficiência: são muitas mensagens processadas, a todo momento
- Segurança e baixo uso de memória, ausência de garbage collector: é necessário o maior uptime possível - nenhum comunista pode passar! É necessário consumir o mínimo de recursos possíveis das máquinas para poder rodar a maior quantidade possível de instâncias do programa. Temos que pegar todos os comunas!



Exemplo 3: ferramenta CLI codificação

- Uma simples ferramenta CLI para uso por desenvolvedores
- Justifica-se Rust:
  - Pelo usu contínuo: falta de eficiência significaria queda em produtividade
  - Muito uso de memória potencialmente atrapalha demais atividades
  - Crashes dentre outras falhas podem significar perda de trabalho realizado, ou perda de produtividade tendo que reinicializar o ambiente

Hora do exercício!

- Vocês tem uma tarefa muito importante: caçar as pessoas que identificam-se como “furries”. São gente da pior espécie.
- Para isso, devem construir uma ferramenta CLI que, ao receber uma mensagem escrita pelo usuário, diz se ela é “felpuda” ou não.
- Mensagens felpudas caracterizam-se pela presença das palavras:
  - “Furry”, “felpudo”, “paw”, “Sonic”, “Shadow”, “ouriço”, “e621”, “scalie”, “fursuit”, “uwu”, “catgirl”, “fluffy”
- Ao detectar uma mensagem felpuda, o programa deve exclamar “FELPUDO DETECTADO”, mas continuar executando normalmente.
- Vocês podem fazer a atividade tanto no seu computador, como no Rust Playground (acho...?)
- Mãos à obra!

That's all folks!

Obrigado.