INSTITUTO DE INFORMÁTICA
Universidade Federal de Goiás

High Availability Strategies in Distributed Systems: A Practical Guide

Vol. 13, Issue: 01, January: 2025
(IJRSML) ISSN (P): 2321 - 2853

Siddharth Choudhary Rajesh | Dr. Ravinder Kumar

Publicação: https://ijrsml.org - Impact Factor: 6.112 (2023)

INF
INSTITUTO DE INFORMÁTICA

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

Palavras chaves

- High availability
- Distributed systems
- Fault tolerance
- Redundancy
- Load balancing
- Replication
- Recovery mechanisms
- Active-active configuration
- Active-passive configuration
- CAP theorem
- System resilience
- Downtime minimization
- Fault-tolerant design

Qual é o problema o artigo tenta abordar?

Aborda o problema de **como garantir alta disponibilidade em sistemas distribuídos**, considerando os desafios práticos enfrentados por arquitetos e engenheiros de software.

Especificamente, o texto investiga **estratégias práticas, ferramentas e padrões arquiteturais** que podem ser adotados para **minimizar o tempo de indisponibilidade** e **aumentar a resiliência** para sistemas distribuídos modernos.

Quais são as contribuições reivindicadas no artigo?

As principais contribuições reivindicadas pelos autores são:

- A sistematização e **categorização das principais estratégias de alta disponibilidade** (HA) adotadas na prática.
- Uma análise comparativa de técnicas como replicação de serviços, failover automatizado, quorum-based consensus e deploy multi região.
- A proposição de uma estrutura prática para avaliação e escolha de estratégias de HA com base em critérios como **custo**, **complexidade operacional** e **criticidade do serviço**.
- Um estudo de caso que ilustra a aplicação dessas estratégias em um sistema real.

Como os autores comprovam suas alegações?

Os autores utilizam os seguintes métodos para sustentar suas afirmações:

- Referências a literatura consolidada em sistemas distribuídos e computação em nuvem.
- Exemplos práticos e padrões arquiteturais bem conhecidos, como o uso de load balancers, serviços stateless, replicação ativa/passiva, entre outros.
- Um estudo de caso detalhado, que demonstra a aplicação de diferentes estratégias de HA em uma arquitetura baseada em micro serviços em nuvem, avaliando os impactos em termos de disponibilidade, latência e custo.

Quais são as conclusões?

O artigo conclui que:

- Não existe uma solução única para alta disponibilidade; a escolha da estratégia ideal depende de múltiplos fatores contextuais.
- Estratégias bem-sucedidas equilibram resiliência, simplicidade e custo operacional.
- A aplicação consciente de padrões arquiteturais e técnicas de tolerância a falhas é fundamental para alcançar altos níveis de disponibilidade.
- A adoção de uma abordagem orientada por dados e testes de resiliência contínuos (como o chaos engineering) é essencial para validar a eficácia das estratégias adotadas em ambientes reais.

Técnicas de alta disponibilidade

Tolerância a falhas

Capacidade do sistema de continuar operando mesmo quando componentes individuais falham.

Exemplos:

- Servidores e aplicações redundantes
- Circuit breaker
- Self healing
- Fallback automático

Redundância

Duplicação de componentes críticos (hardware/software) para evitar ponto único de falha (SPOF).

Tipos:

- Redundância de servidores e aplicações
- Redundância de dados

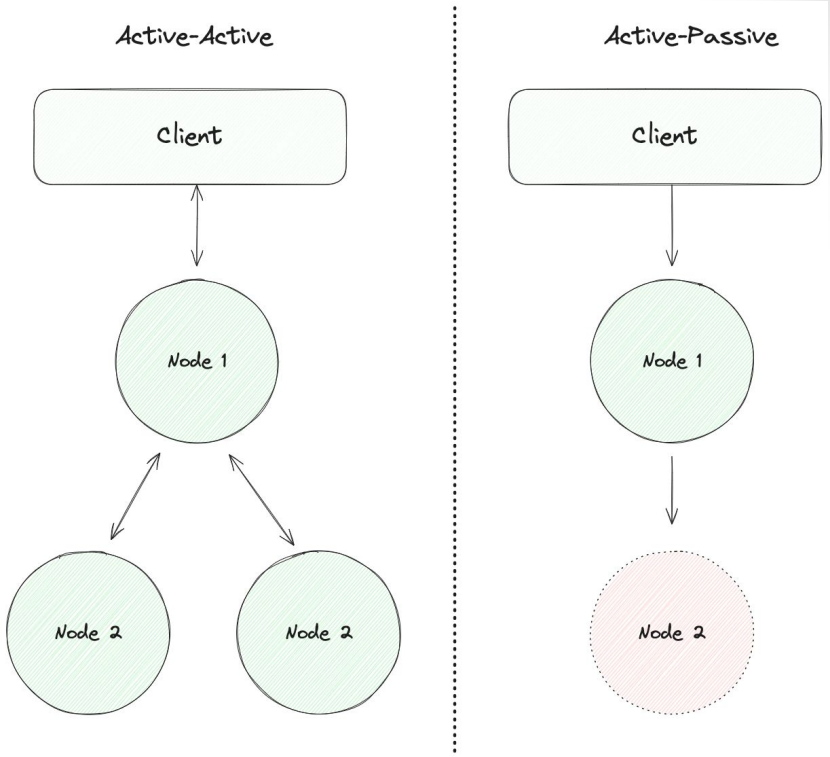
Mecanismos de failover

Active-Active

- Todos os nós estão ativos simultaneamente
- Alta performance, alta complexidade.

Active-Passive

- Um nó em espera assume em caso de falha
- Menor custo, maior tempo de recuperação.



Balanceamento de carga

Distribuição do tráfego entre várias instâncias para evitar sobrecarga.

Ferramentas

- HAProxy
- NGINX
- Load Balancers

Replicação de dados

Síncrona

Garantia de consistência, mas maior latência.

Assíncrona

Alta disponibilidade, risco de inconsistência em falhas.

Monitoramento e manutenção proativa

- Deteção de anomalias
- Alertas
- Automatizar correções com scripts ou orquestradores

Recuperação e resiliência

Capacidade do sistema de se recuperar rapidamente após falhas.

Práticas:

- Backup e restore automatizados
- Testes de recuperação de desastres

Testes de resiliência

Injetar falhas controladas para validar a robustez das estratégias de HA.

Chaos Engineering

Escala horizontal automática

Ajuste dinâmico do número de instâncias com base na carga.

Ferramentas:

- Kubernetes HPA
- AWS Auto Scaling.

Protocolos de consenso

Raft

Garantir que os nós concordem sobre o estado do sistema. No artigo, a utilização do protocolo é abstraída com a utilização do Kubernetes

Como funciona:

Baseia-se em logs replicados e possui três papéis por nó:

- **Leader**: aceita comandos e os propaga
- **Follower**: responde ao líder, mantém cópias do log.
- **Candidate**: tenta se eleger líder.

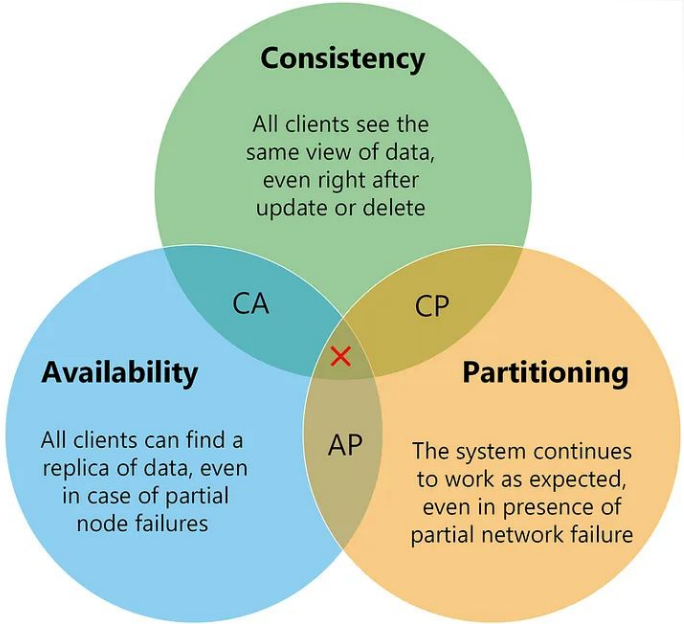
No K8s o etcd usa Raft internamente

- Raft é usado para manter consistência entre os nós do etcd.
- Garante que todos os nós etcd estejam sincronizados mesmo em presença de falhas.
- Um nó atua como líder e os demais como seguidores

CAP

O artigo reconhece o CAP theorem como um princípio central ao projetar sistemas distribuídos altamente disponíveis

- C (Consistency): Todos os nós veem os mesmos dados ao mesmo tempo.
- A (Availability): Cada solicitação recebe uma resposta — mesmo que parte do sistema falhe.
- P (Partition tolerance): O sistema continua operando mesmo com falhas de comunicação entre nós.



Enfatiza que não é possível garantir simultaneamente os três elementos, então arquitetos precisam fazer escolhas dependendo do caso de uso.

Estudo de caso do artigo

Ambiente de simulado

Os autores montaram uma infraestrutura virtualizada com ferramentas amplamente usadas na indústria:

- Docker e Kubernetes: Orquestração de microsserviços.
- HAProxy: Load balancer para simular balanceamento de carga.
- MySQL: Replicação síncrona e assíncrona para testes de consistência.
- Prometheus e Grafana: Monitoramento e coleta de métricas.
- Chaos Monkey: Simular falhas reais como desligamento de serviços, queda de nós e picos de tráfego.

Cenários simulados

Foram avaliados quatro tipos principais de falhas:

Falha de nó (Node Failure)	Término aleatório de um ou mais nós para avaliar o impacto no desempenho e recuperação do sistema
Partição de rede (Network Partition)	Simulando um cenário onde a comunicação entre subconjuntos de nós é interrompida.
Indisponibilidade do banco de dados	Testando a resposta do sistema quando o banco de dados primário fica indisponível.
Pico súbito de tráfego	Simulando um aumento inesperado nas solicitações recebidas para testar o balanceador de carga e a alocação de recursos.

Métricas avaliadas

- Uptime (%)
- Tempo médio de recuperação (MTTR)
- Latência média por requisição
- Consistência dos dados
- Throughput (requisições por segundo)

Análises estatísticas - Uptime

Objetivo:

Medir a porcentagem de tempo em que o sistema permaneceu operacional durante falhas em diferentes estratégias.

Scenario	Active-Active (%)	Active-Passive (%)	Synchronous Replication (%)	Asynchronous Replication (%)
Node Failure	99.99	99.85	99.95	99.90
Network Partition	99.80	99.75	99.70	99.85
Database Failure	99.95	99.80	99.90	99.88
Sudden Traffic Surge	99.98	99.85	99.92	99.87

Análises estatísticas - Latency

Objetivo:

Avaliar a latência média experimentada pelos usuários em diferentes cenários

Scenario	Active-Active (ms)	Active-Passive (ms)	Synchronous Replication (ms)	Asynchronous Replication (ms)
Node Failure	50	75	60	55
Network Partition	70	85	100	65
Database Failure	60	80	90	70
Sudden Traffic Surge	55	70	65	60

Análises estatísticas - System Throughput

Objetivo:

Avaliar o número de solicitações bem-sucedidas processadas por segundo durante e após falhas.

Scenario	Active-Active (req/s)	Active-Passive (req/s)	Synchronous Replication (req/s)	Asynchronous Replication (req/s)
Node Failure	2000	1800	1900	1950
Network Partition	1800	1600	1700	1850
Database Failure	1900	1700	1800	1925
Sudden Traffic Surge	2100	1900	2000	2050

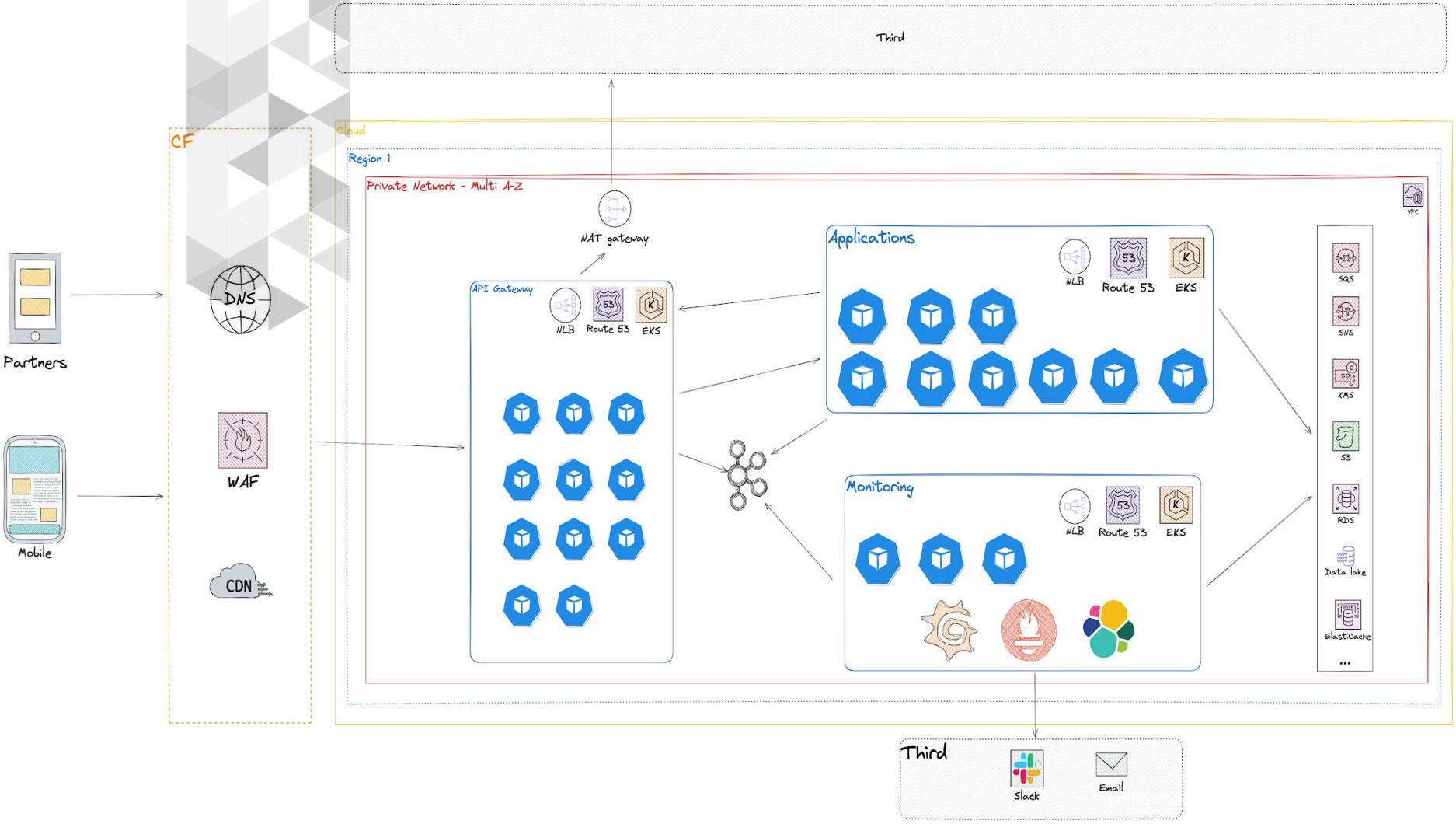
Principais insights do estudo

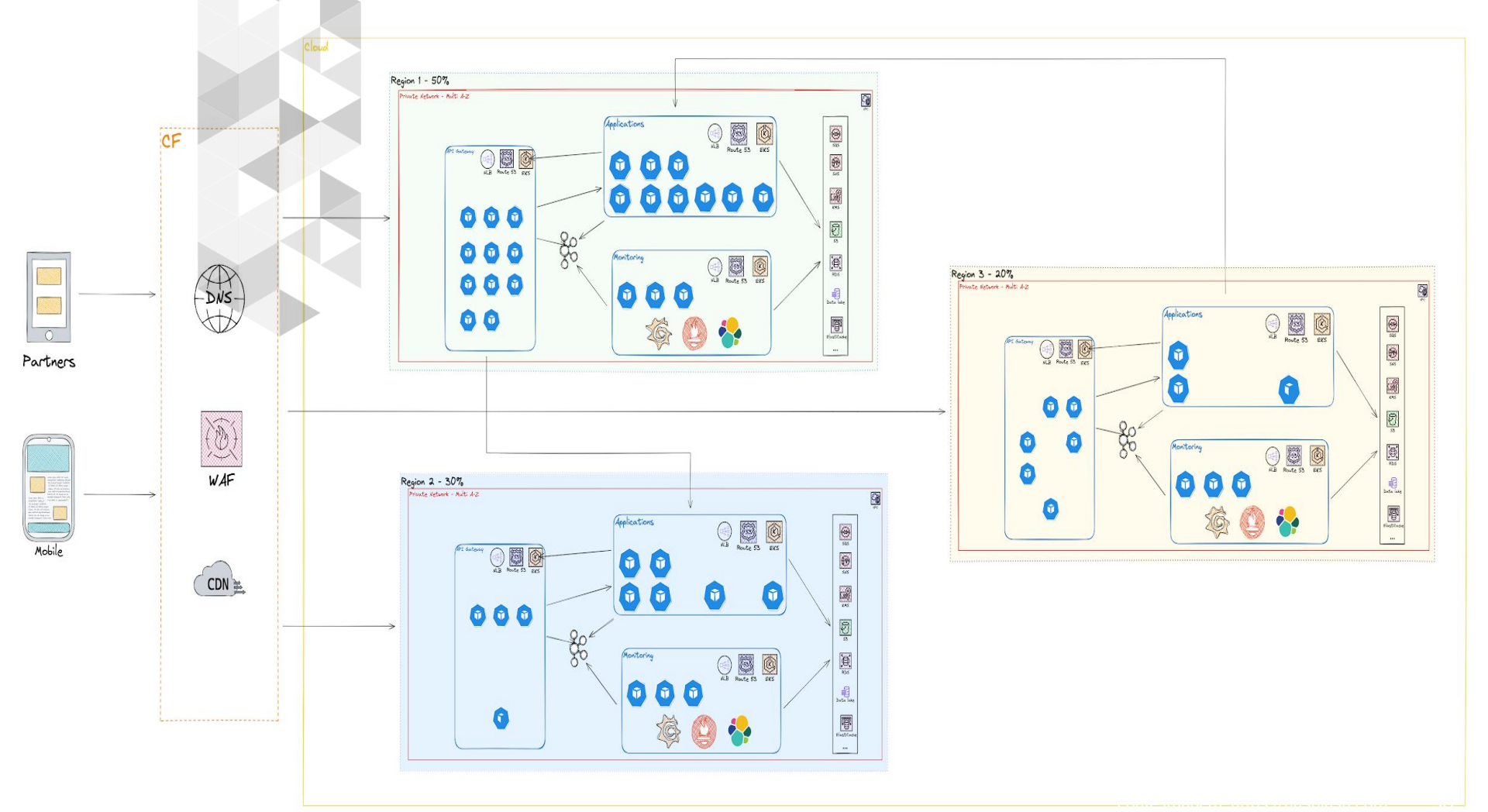
- Active-Active com replicação assíncrona teve o melhor desempenho em disponibilidade e throughput, mas com pequenas perdas de consistência.
- Active-Passive foi eficiente em termos de custo e complexidade, mas mais lento na recuperação.
- Replicação síncrona garantiu consistência total, mas introduziu mais latência e menor disponibilidade durante partições de rede.
- Monitoramento e auto escalabilidade foram cruciais para evitar degradação durante picos de carga.
- A implementação de estratégias de alta disponibilidade envolve custos significativos, incluindo hardware redundante, software especializado e pessoal qualificado. Estudos sugerem uma abordagem equilibrada para otimizar custos sem comprometer os requisitos essenciais de disponibilidade.

Na prática

Números

- Em operação desde 2020 alcançamos de forma escalável +3M Clientes globais ativos
- + R$4 bilhões de ativos sob custódia
- Em pico respondemos em média a mais de ~1.7K Req/s abaixo de 1s
- + de 300 aplicações em execução que se comunicam de forma síncrona e assíncrona
- SLA uptime 99.999...
- Estratégia CAP: AP - (Availability) (Partition tolerance)


Third
CF
Cloud
Region 1
Private Network - Multi A-Z
VPC
NAT gateway
Partners
Mobile
DNS
WAF
CDN
API Gateway
NLB
Route 53
EKS
Applications
NLB
Route 53
EKS
Monitoring
NLB
Route 53
EKS
SQS
SNS
KMS
S3
RDS
Data lake
ElastiCache
...
Third
Slack
Email


cloud
CF
Partners
Mobile
DNS
WAF
CDN
Region 1 - 50%
Private Network - Multi AZ
Applications
Monitoring
Region 2 - 30%
Private Network - Multi AZ
Applications
Monitoring
Region 3 - 20%
Private Network - Multi AZ
Applications
Monitoring

Dúvidas

nogsantos@discente.ufg.br

Ou sugestões?

INF
INSTITUTO DE INFORMÁTICA

UFG
UNIVERSIDADE FEDERAL DE GOIÁS