Projeto 5 - Análise de Desempenho de Armazenamento de Kubernetes

Apresentação Final

Laboratório em Engenharia Informática

PG42577 - Daniel Assunção Regado
PG42824 - Diogo Araújo Ferreira
PG42828 - Filipe José da Silva Freitas
PG42852 - Vasco António Lopes Ramos




Orientador: Prof. João Tiago Paulo

Contexto

- **Kubernetes:** ferramenta para orquestração de sistemas e serviços muito utilizada.
- Uma parte importante dos sistemas são os dados, i.e., storage.
- É importante perceber como se comporta o Kubernetes com dados e diferentes mecanismos de storage.
- Entender como os diferentes tipos de aplicações se comportam com os diferentes tipos de armazenamento e o seu impacto na performance das mesmas.

Objetivos

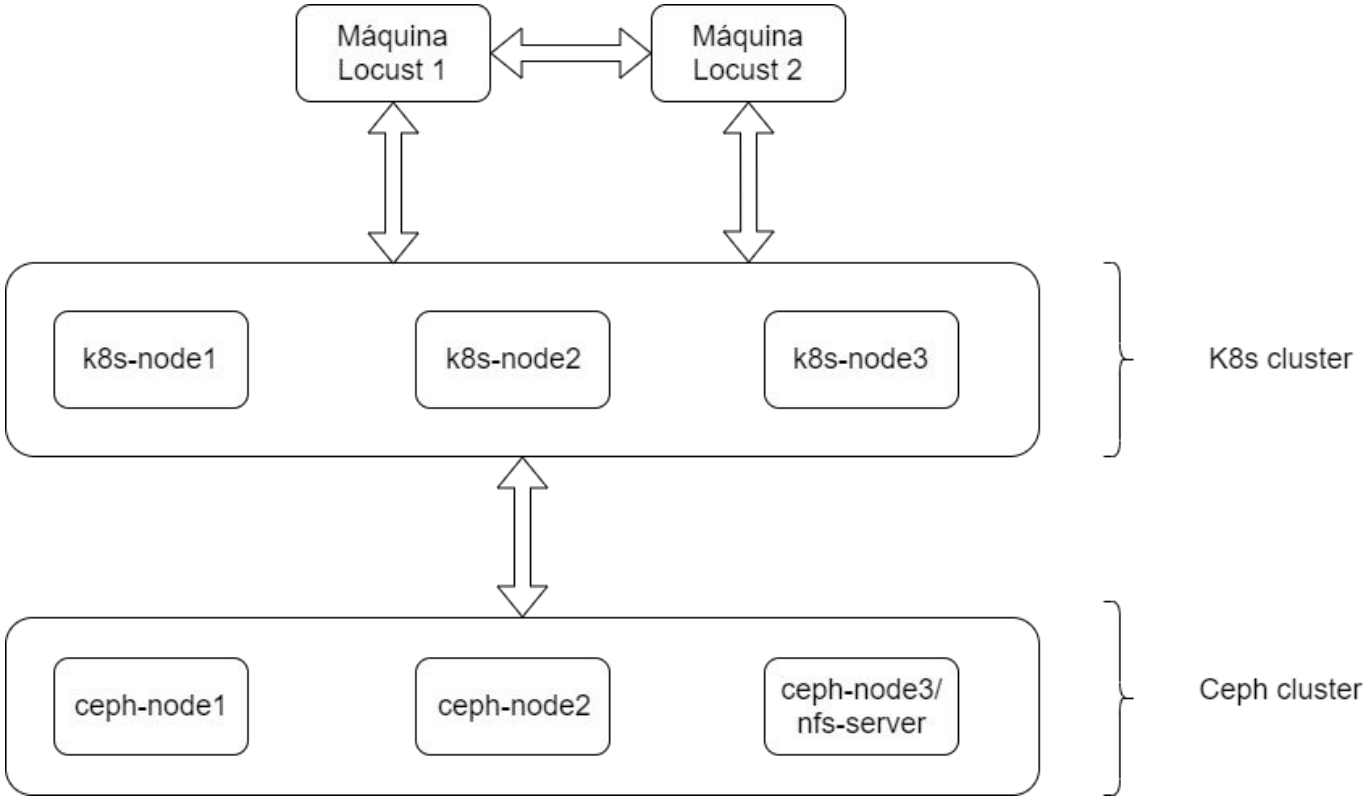
- Escolher aplicações distribuídas como suporte às avaliações
- Instalar as aplicações com recurso a Kubernetes
- Analisar o desempenho das mesmas com os diferentes mecanismos de storage.
- Otimização das instalações, se necessário.

Aplicações e backends de storage escolhidos

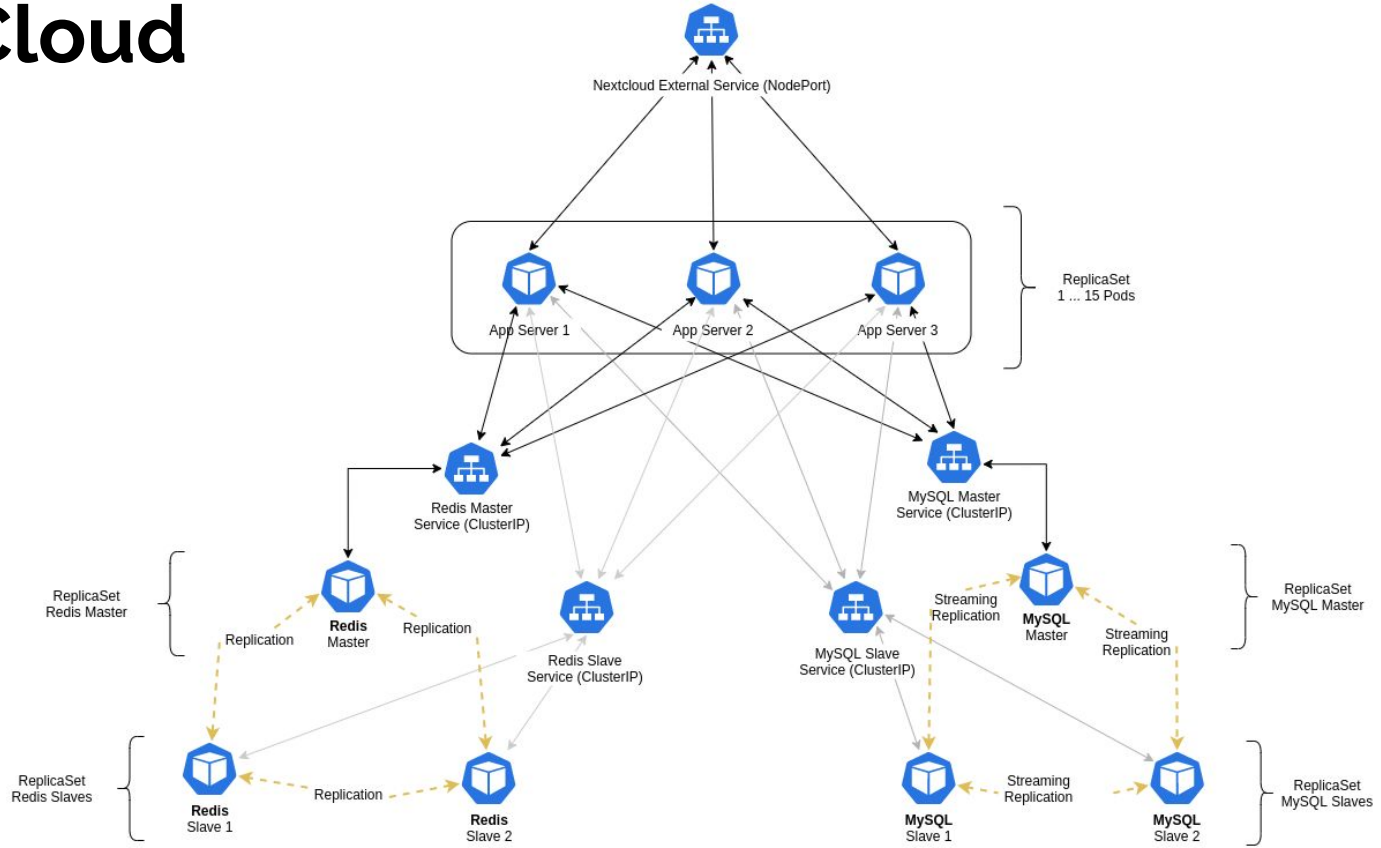
- Aplicações:
  - Wiki.JS
  - NextCloud
  - PeerTube

- Backends de Storage:
  - NFS
  - Ceph

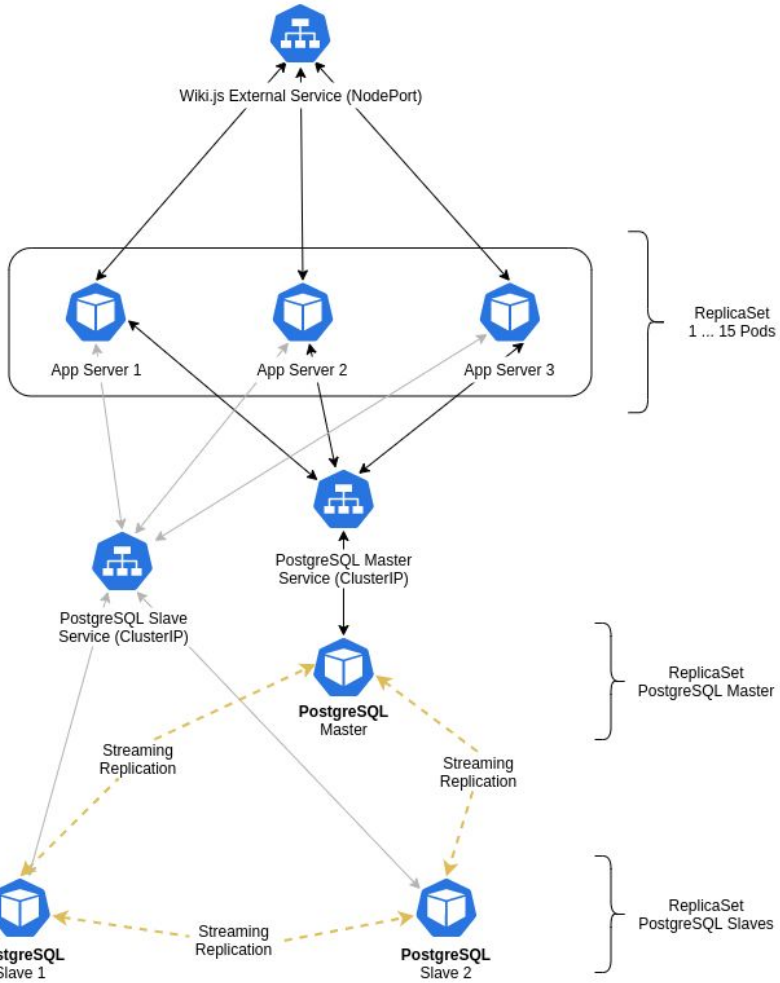
Infraestrutura



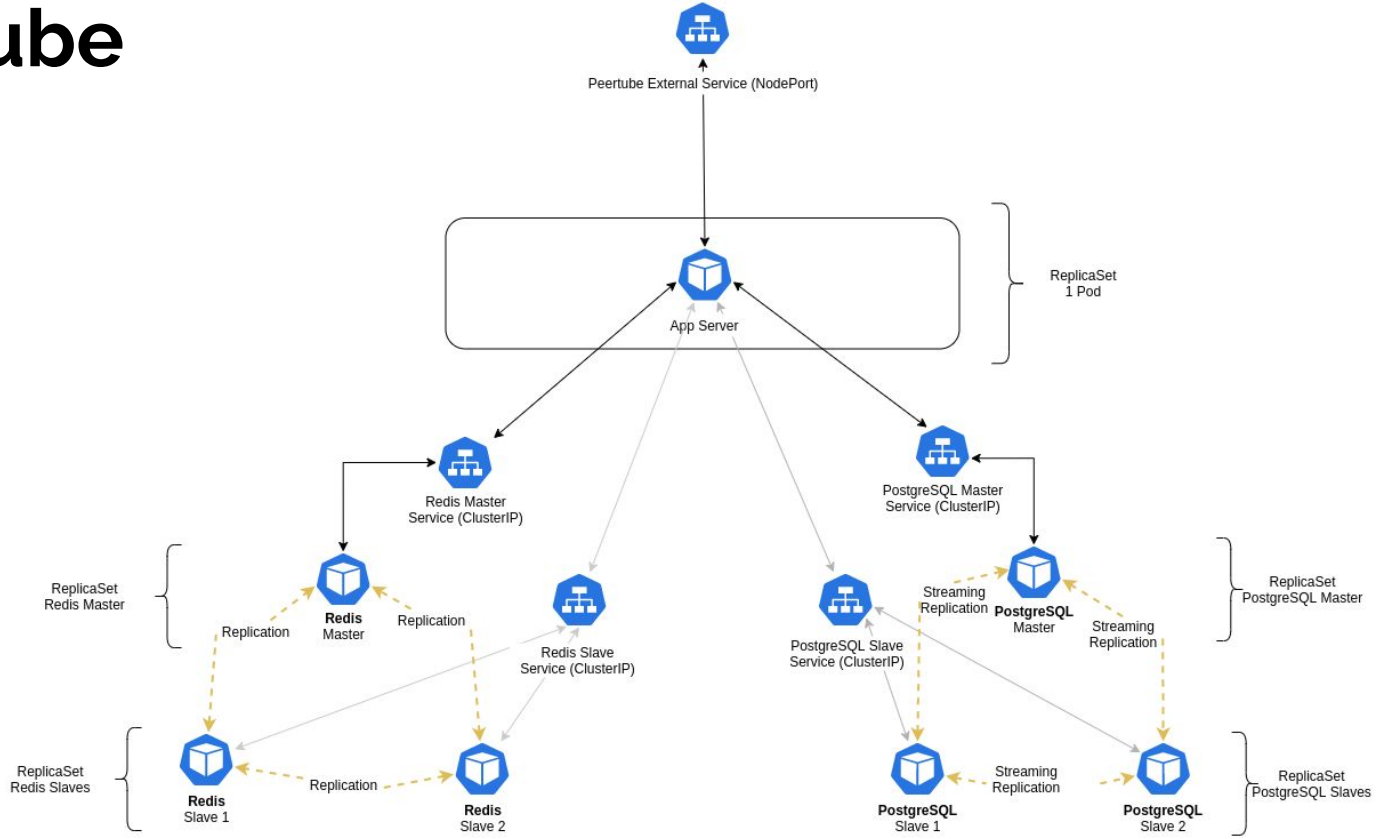
NextCloud


Nextcloud External Service (NodePort)
App Server 1
App Server 2
App Server 3
ReplicaSet
1 ... 15 Pods
Redis Master
Service (ClusterIP)
MySQL Master
Service (ClusterIP)
ReplicaSet
Redis Master
Redis
Master
Replication
Replication
Redis Slave
Service (ClusterIP)
MySQL Slave
Service (ClusterIP)
Streaming
Replication
MySQL
Master
Streaming
Replication
ReplicaSet
MySQL Master
ReplicaSet
Redis Slaves
Redis
Slave 1
Replication
Redis
Slave 2
MySQL
Slave 1
Streaming
Replication
MySQL
Slave 2
ReplicaSet
MySQL Slaves

Wiki.JS


Wiki.js External Service (NodePort)
App Server 1
App Server 2
App Server 3
ReplicaSet
1 ... 15 Pods
PostgreSQL Master
Service (ClusterIP)
PostgreSQL Slave
Service (ClusterIP)
PostgreSQL
Master
ReplicaSet
PostgreSQL Master
Streaming
Replication
Streaming
Replication
PostgreSQL
Slave 1
Streaming
Replication
PostgreSQL
Slave 2
ReplicaSet
PostgreSQL Slaves

PeerTube


Peertube External Service (NodePort)
App Server
ReplicaSet
1 Pod
Redis Master
Service (ClusterIP)
PostgreSQL Master
Service (ClusterIP)
ReplicaSet
Redis Master
Redis
Master
Replication
Replication
Redis Slave
Service (ClusterIP)
PostgreSQL Slave
Service (ClusterIP)
Streaming
Replication
PostgreSQL
Master
Streaming
Replication
ReplicaSet
PostgreSQL Master
ReplicaSet
Redis Slaves
Redis
Slave 1
Replication
Redis
Slave 2
PostgreSQL
Slave 1
Streaming
Replication
PostgreSQL
Slave 2
ReplicaSet
PostgreSQL Slaves

Ferramentas utilizadas no *Benchmarking*

- **Locust**
  - Baseado em Python, executado em ambiente distribuído, com 4 workers (2 por VM).
  - Métricas recolhidas:
    - *Throughput* (pedidos/s);
    - Tempos de resposta;
    - *Failure rate*.
- **Collectl**
  - Responsável por recolher métricas (de monitoriazação), relativas à infraestrutura.
  - Métricas recolhidas:
    - Utilização de CPU;
    - Utilização de memória;
    - Débito dos clientes de *storage*;
    - Tráfego de rede.

Metodologia de Benchmarking

# Utilizadores	Spawn Rate (utilizadores/s)	Tempo de execução (min)
25	0.5	15
50	1	15
75	1.5	15
100	2	15

- Testes executados com apenas uma aplicação instalada no cluster de cada vez.
- Spawn Rate calculada de modo a que todos os utilizadores estejam ativos após o primeiro minuto

Cenários de teste

Concebemos 3 cenários de teste para cada uma das aplicações:

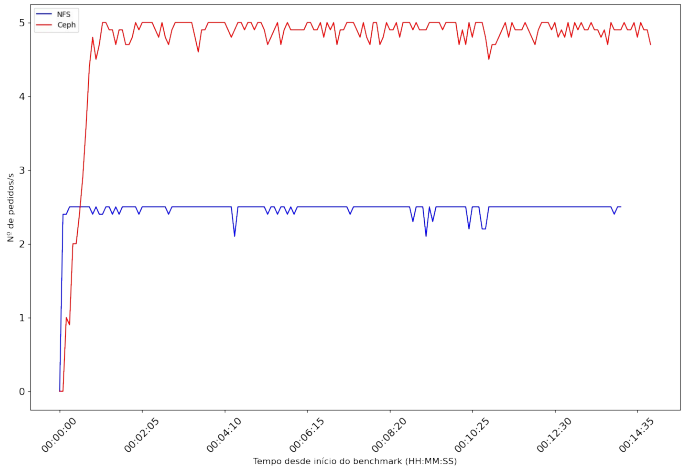
- **Leituras:** um cenário que consiste apenas em consumir informação da aplicação (páginas web e downloads).
- **Escritas:** este cenário consiste apenas em inserir informação na aplicação (uploads).
- **Misto:** este último cenário tenta simular um utilizador normal de cada uma das aplicações combinando os cenários anteriores.

Resultados

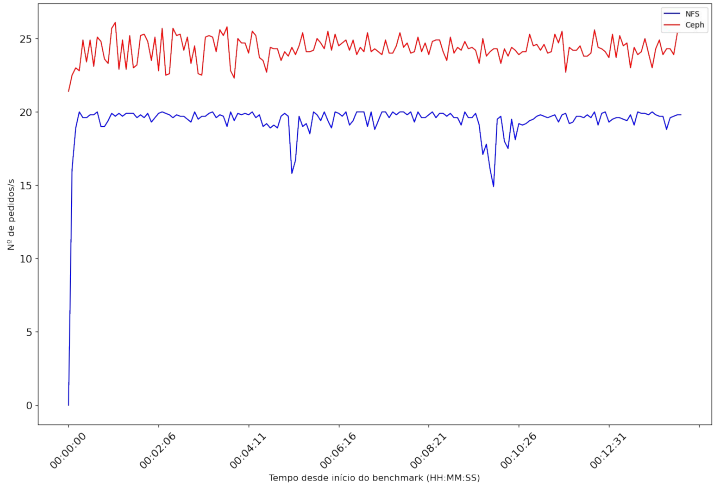
Nextcloud - Cenário de Leituras

	Tempo médio de resposta (ms)	
# Utilizadores	NFS	*Ceph*
25	119	90 -24.37%
50	96	88 -8.33%
75	123	85 -30.89%
100	112	88 -21.43%

Tabela 5.5: *NextCloud - Ceph* vs NFS - Tempos médios de Resposta

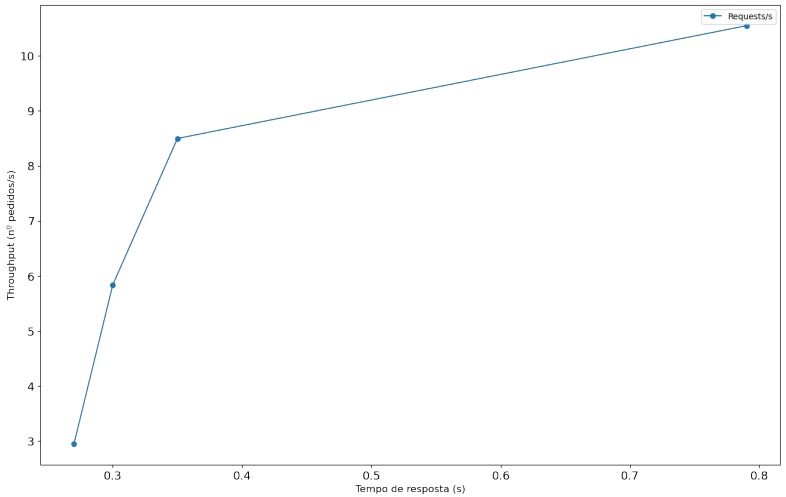


Throughput (pedidos/s), 25 utilizadores

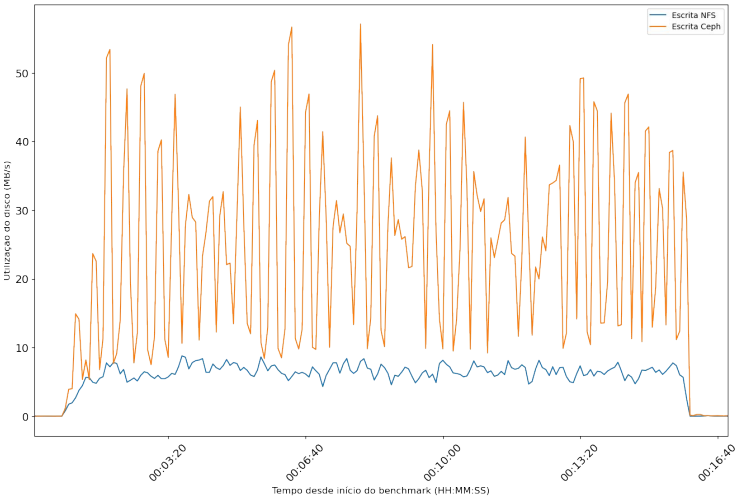


Throughput (pedidos/s), 100 utilizadores

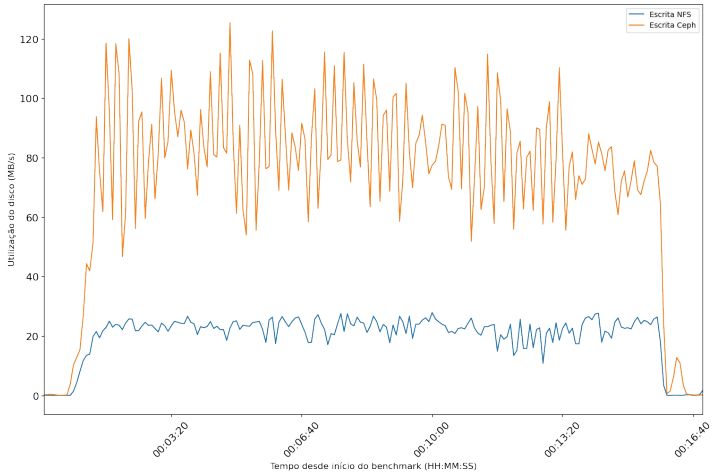
Nextcloud - Cenário de Escritas



Tempo de Resposta (s) vs. Throughput (pedidos/s), para os vários níveis de utilizadores



Débito de escritas (Mb/s), 25 utilizadores

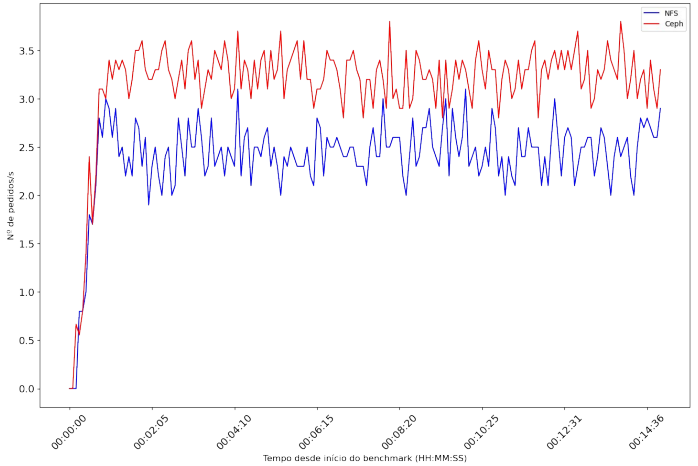


Débito de escritas (Mb/s), 100 utilizadores

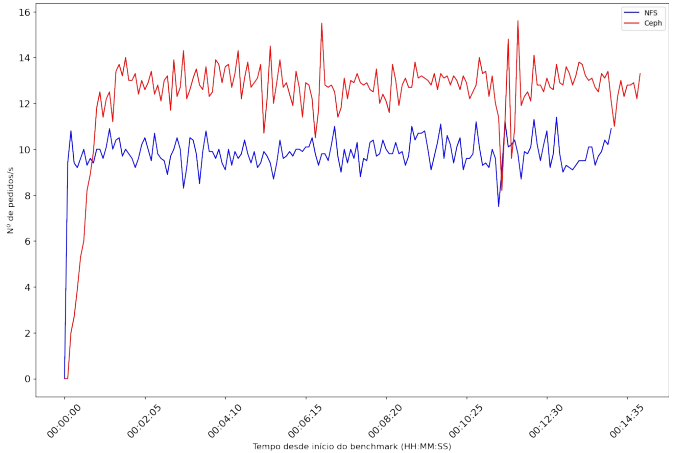
Nextcloud - Cenário Misto

	Tempo mediano de resposta (ms)	
# Utilizadores	NFS	*Ceph*
25	180	96 -46.67%
50	265	110 -58.49%
75	170	120 -29.41%
100	160	120 -25.00%

Tabela 5.8: *NextCloud - Ceph* vs NFS - Tempos medianos de Resposta

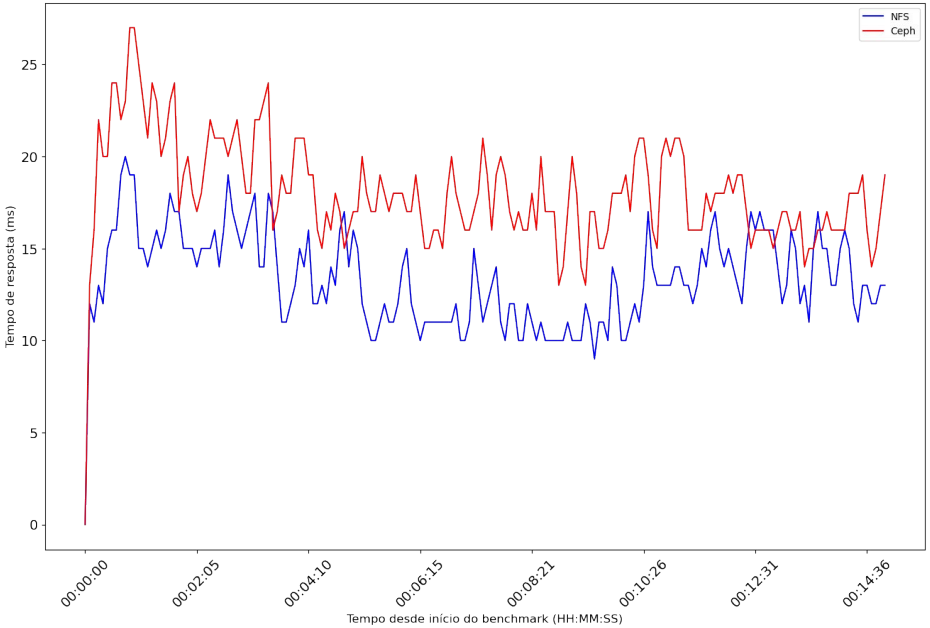


Throughput (pedidos/s), 25 utilizadores

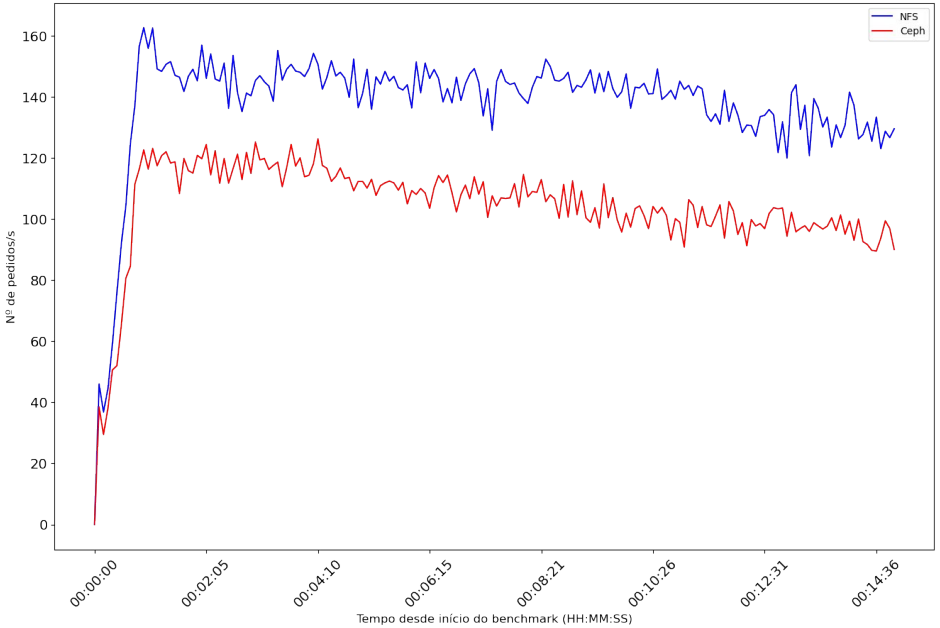


Throughput (pedidos/s), 100 utilizadores

Wiki.JS - Cenário de Leituras

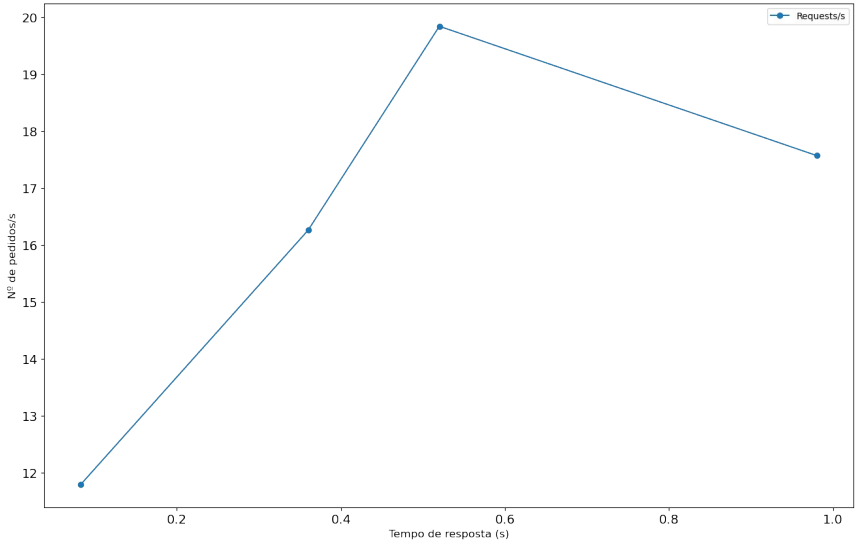


Tempo mediano de resposta, 75 utilizadores

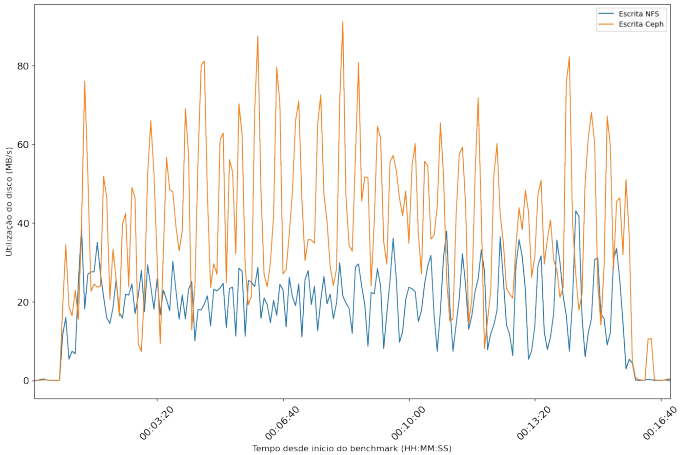


Throughput (pedidos/s), 75 utilizadores

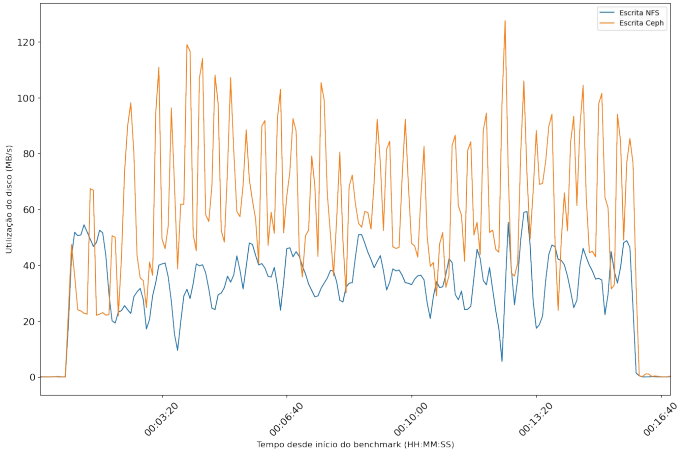
Wiki.JS - Cenário de Escritas



Tempo de Resposta (s) vs. Throughput (pedidos/s), para os vários níveis de utilizadores

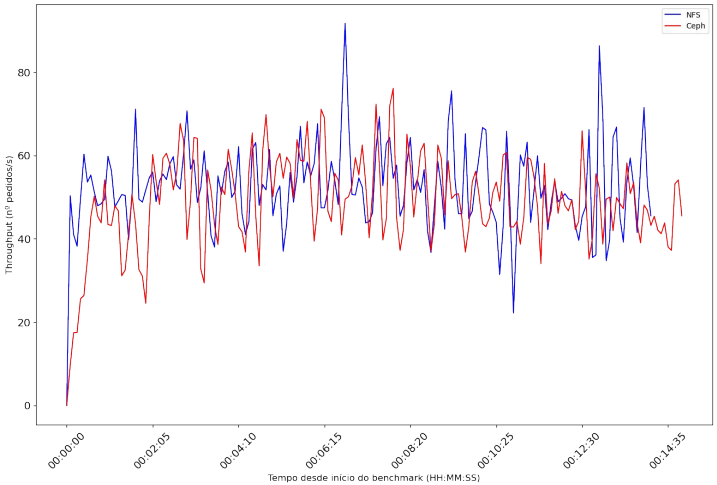


Débito de escritas (Mb/s), 25 utilizadores

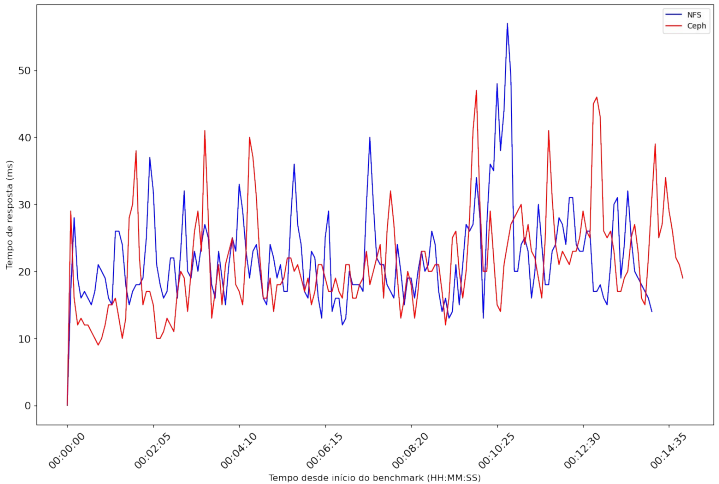


Débito de escritas (Mb/s), 100 utilizadores

Wiki.JS - Cenário Misto



Throughput (pedidos/s), 50 utilizadores



Tempo mediano de resposta, 50 utilizadores

# Utilizadores	*Throughput* (pedidos/s)	Tempo mediano de resposta (ms)	Erro (%)
25	28.44	13	0
50 +100%	52.15 +83.37%	21 +61.54%	0.070
75 +200%	70.82 +149.02%	25 +92.31%	0.049
100 +300%	70.27 +147.08%	42 +223.08%	0.039

Tabela 5.11: *Wiki.JS* NFS - Resumo dos resultados para simulação de utilizador

# Utilizadores	*Throughput* (pedidos/s)	Tempo mediano de resposta (ms)	Erro (%)
25	28.07	14	0.007
50 +100%	49.70 +77.06%	20 +42.86%	0.006
75 +200%	60.66 +116.10%	25 +78.57%	1.044
100 +300%	61.14 +117.81%	47 +235.71%	0.069

Tabela 5.14: *Wiki.JS Ceph* - Resumo dos resultados para simulação de utilizador

Discussão (extra resultados)

- PeerTube produziu resultados inconclusivos.
- Facilidade de Instalação.
- Disponibilização de storage: utilização de *Storage Classes* (*dynamic provisioning*).
- Relativamente ao *Ceph*, a *kernel driver* tem performance superior à *FUSE driver*.

Conclusão

- Em geral, *Ceph* > *NFS*:
  - Diferença de *performance* negligível
  - Ganhos em resiliência compensam as possíveis pequenas perdas de *performance*
- O *design* interno das aplicações tem uma maior influência na sua *performance* do que o *backend* de *storage* utilizado.
- Trabalho futuro:
  - Cenários de teste relativos ao *Ceph* com diferentes níveis de replicação
  - Utilização do *Rook.io* para uma mais fácil instalação dos *backends* de *storage*
  - Cenários de teste onde são executadas duas aplicações em simultâneo

Obrigado!