1. **Requisitos**

* **Modificações Concorrentes de Arquivos**

Quaisquer modificações feita nos arquivos são visíveis a todos os servidores imediatamente. Cada arquivo do sistema pode ser montado em milhares de servidores ao mesmo tempo com alta-performance com leituras e escritas concorrentes e informação compartilhada.

* **Replicação de Arquivos**

Cada arquivo do sistema pode ser montado em milhares de servidores ao mesmo tempo.

* **Heterogeneidade**

O SAD JuiceFS pode ser usado como um sistema de arquivos locais, pois faz uma interface perfeita com os arquivos existentes. Além de ser totalmente compatível com HDFS API (que melhora a performance de metadata), provê *S3-gateway* para implementar uma *S3-compatible* *access interface* e é possível utiliza-lo no Kubernetes através do CSI Driver.

* **Tolerância a Falhas**

Os arquivos do sistema podem estra em diversos servidores ao mesmo tempo, tendo informação compartilhada e por padrão, o JuiceFS, monitora os arquivos em tempo de execução por porta local TCP através do pprof. Além disso, há um sistema de backup baseado na utilização de arquivos json que pode importar informações de um dataset para outro dataset vazio. Junto há isso há um sistema de Log para ajudar a identificar falhas que ocorrerem no SAD.

* **Consistência**

Cada arquivo do sistema pode ser montado em milhares de servidores ao mesmo tempo com alta-performance com leituras e escritas concorrentes e informação compartilhada.

* **Proteção**

Possui suporte a BSD lock (flock) e POSIX lock (fcntl) e realiza a criptografia de informação “*in Transit*” e “*at Rest*”.

* **Eficiência**

A latência pode chegar a poucos milissegundos e a taxa de transferência pode ser quase ilimitada dependendo da escala de armazenamento do objeto. A performance do sistema não é afetada pela criptografia.

* **Transparência**
  + **De acesso**

A plataforma de cloud irá providenciar linhas de comunicação tanto internas como pública. Para alcançar as condições de execução na mesma plataforma como em cloud, o JuiceFS irá automaticamente resolver através de linhas de comando internas o acesso ao objeto de armazenamento, dessa forma não terá apenas baixa latência como a comunicação interna gerada pelo tráfico vai estar livre.

* + **De localização**

Os arquivos serão separados em *Chunks*, *Slices* e *Blocks* e armazenados no objeto de armazenamento. Devido a isso, os arquivos fontes não serão encontrados no explorador de arquivos (file browser) do objeto de armazenamento na plataforma. Há *chunks* de diretórios e diretórios numerados digitalmente (*digitally numbered directories*) e arquivos em *buckets*.

* + **De mobilidade**

JuiceFS supports POSIX compatible ACL to manage permissions in the granularity of directory or file. The behavior is the same as a local file system.

To provide users with an intuitive and consistent permission management experience (e.g. the files accessible by user A in host X should be accessible in host Y with the same user), the same user who want to access JuiceFS should have the same UID and GID on all hosts.

O Juice

* + **De desempenho**
  + **De escalabilidade**
  + **Contra falhas**
  + **De replicação**