Consistência Transacional para Computação Serverless

HydroCache

Desenvolvimento de Aplicações Distribuídas

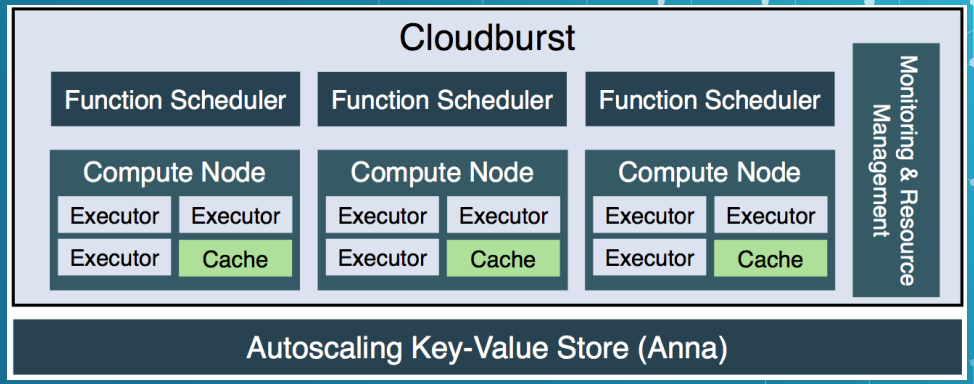
Grupo 5:
Sara Machado, 86923
Rafael Figueiredo, 90770
Ricardo Grade, 90774

Sistemas FaaS (Function-as-a-Service)

- Composição de múltiplas funções independentes;
- Separação entre a camada de execução e de armazenamento;
- Automaticamente escalável;
- Problema:
  - Latência de operações I/O;
- Solução:
  - Caches.

Arquitetura de um CloudBurst

- Camada de armazenamento:
  - Anna.
- Camada de execução -> Cloudburst:
  - Planeador de Funções;
  - Nó:
    - Executor de Funções (Thread);
    - Cache.
- Encadeamento de Funções:
  - DAG de funções.



Consistência Causal (CC)

- Relação de Lamport: “Happens-Before”: $a_i \rightarrow b_j$;
- Se $a_i \nrightarrow a_j$ e $a_j \nrightarrow a_i$, então são concorrentes: $a_i \sim a_j$.
- Chave tem 4 componentes: $[k, VCk, deps, payload]$
  - *k*: Identificador da chave;
  - VC_k: Conjunto de relógios lógicos, com pares $< id, relógio >$;
  - *deps*: Conjunto de dependências, com pares $< dep\_key, VC >$;
  - *payload*: Valor associado a *k*.

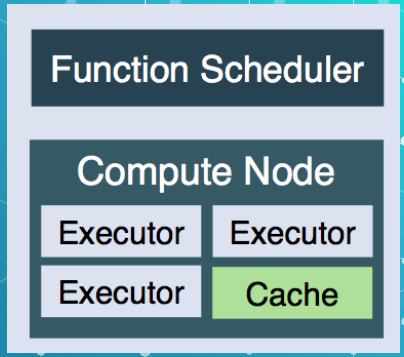
Consintência Causal+ (CC+)

- Extensão da CC, em que réplicas com a mesma chave vão eventualmente convergir para o mesmo valor;
- Política de Resolução de conflitos: Merge de Versões Concorrentes;
  - Sendo $a_i \sim aj$ e $a_k = a_i \, U \, a_j$ então:
    - Payloads são guardados num conjunto:
      - $a_k.payload = < a_i.payload, a_j.payload >$;

Consistência Causal Transacional (TCC)

- Extensão de CC+;
- Garante consistência em leituras e escritas para um conjunto de chaves;
- Snapshot Causal:
  - *R* é um snapshot causal sse $\forall(a_i, b_j) \in R, \nexists\, a_k \mid a_k \rightarrow b_j \wedge a_i \rightarrow a_k$;
  - *R*: Conjunto de chaves de leitura (readset).

HydroCache: Funções individuais



- Corte Causal:
  - C é um corte causal sse $\forall\, d_j.deps, \forall\, k_i \in C, \exists\, d_k \in C \mid d_k\ sobreexcede\ d_j$.
- Conceito de Sobreexceder:
  - Recebendo 2 versões da mesma chave, k_i e k_j , k_i sobreexcede k_j quando $k_i == k_j \vee k_j \rightarrow k_i$.
- Diferenças entre corte e snapshot causal:
  - Num corte se uma chave está no corte então as suas dependências também estão;
  - Concorrência não é permitida no corte.

HydroCache: Funções individuais num único nó

- Como fazer atualização de um corte local:
  - Função pede uma chave, b, não presente no corte;
  - Cache vai buscar uma versão da chave, b_j ;
  - Cache verifica se todas as dependências de b_j sobreexcedem as chaves que já estão no corte;
  - Caso não esteja, a cache vai buscar versões da dependência até ser sobreexcedida;
  - Recursivamente feito até todas as dependências serem sobreexcedidas;
  - Cache atualiza C .

HydroCache:
Funções individuais num único nó

- Garante snapshot causais;
- Garante visibilidade atómica:
  - Quando uma função escreve duas chaves, estas ficam mutualmente dependentes;
  - Todos as escritas são feitas no fim do DAG.
- Possui um garbage collector;
- Garante tolerância a faltas.

HydroCache
Multiplas funções em diferentes nós

- Corte Distribuído;
- Consistência causal transacional multisite (MTCC):
  - Snapshots são construídos para cada DAG;
  - Snapshots apenas precisam de possuir as chaves usadas pelo DAG.

Centralizado (CT)

- Todas as funções num DAG são feitas no mesmo nó;
- Criação de um snapshot do corte local antes da execução;
- Problema:
  - Escalabilidade.

Rumo aos Snapshots Distribuídos

Definições de alguns conceitos e suas relevantes propriedades.

- Keysets;
- Versionsets;
- Keyset-Overlapping Cut: $Corte \cap Keyset$:
  - Um Keyset-Overlapping Cut é um Snapshot;
  - União de Keyset-Overlapping Cuts é um Keyset-Overlapping Cut.

Otimista (OPT)

- Protocolo *"ansioso"*.
- Depende da sincronia existente entre os Cortes dos nós.
- Snapshot Distribuído construido enquanto o DAG é executado.
- Atrativo para sistemas com escritas infrequentes.

Validações (OPT)

Ri: $Versionset$ associado ao $Keyset$ de Fi.

- Fluxo Linear
  - Dadas as funções Fu e Fd | Fu chama Fd
    - Ru ∪ Rd Tem de formar um Snapshot.
- Fluxo Paralelo
  - Dadas as funções Fu_1, Fu_2, … , Fu_n | O seu resultado é acumulado
    - $\bigcup_{i=1}^{n} Ru_i$ Tem de formar um Snapshot.

Conservativo (CON)

- Protocolo *"cauteloso"*.
- Começa por acordar um Snapshot Distribuído entre os nós envolvidos.
- Prevenindo violações ao Snapshot Distribuído.
- Garantindo o sucesso da execução do DAG.

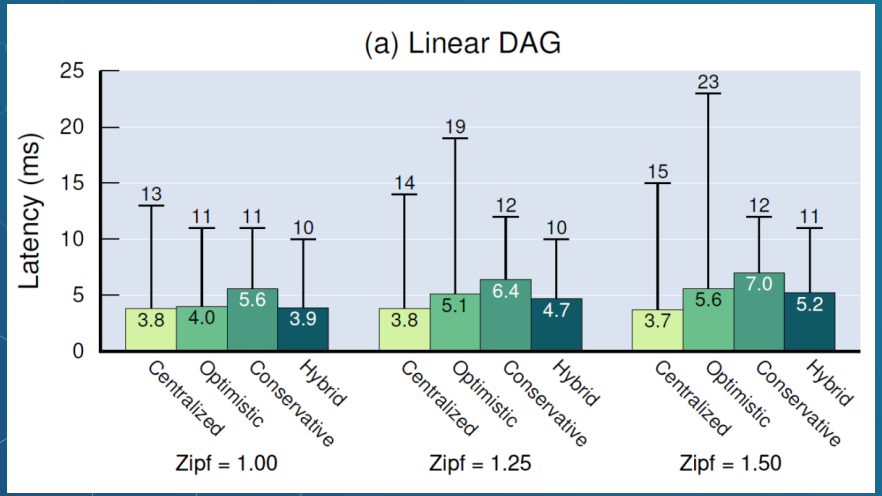
Hibrido (HB)

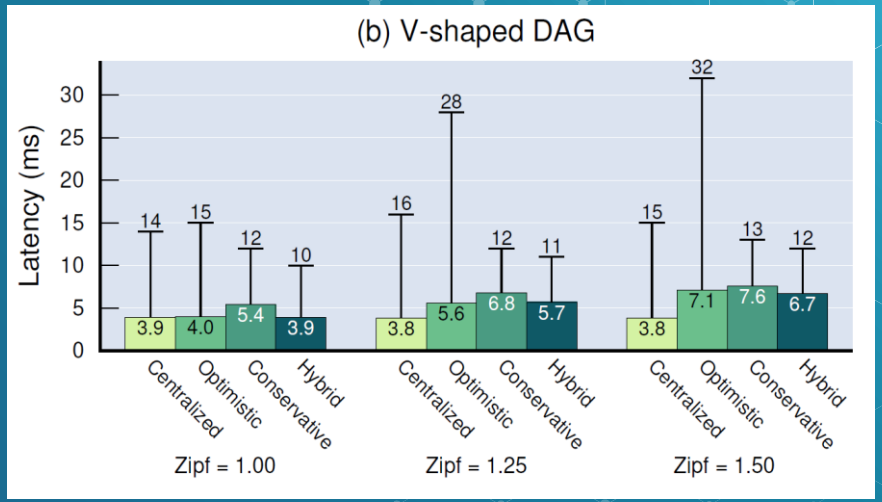
- Combina benefícios do Otimista com o Conservativo;
- Contém otimizações usando os dois protocolos permitindo que seja mais rápido:
  - Pré-Busca;
  - Abortar mais cedo;
  - Cache de resultado das funções.

Discussão Teórica dos Protocolos

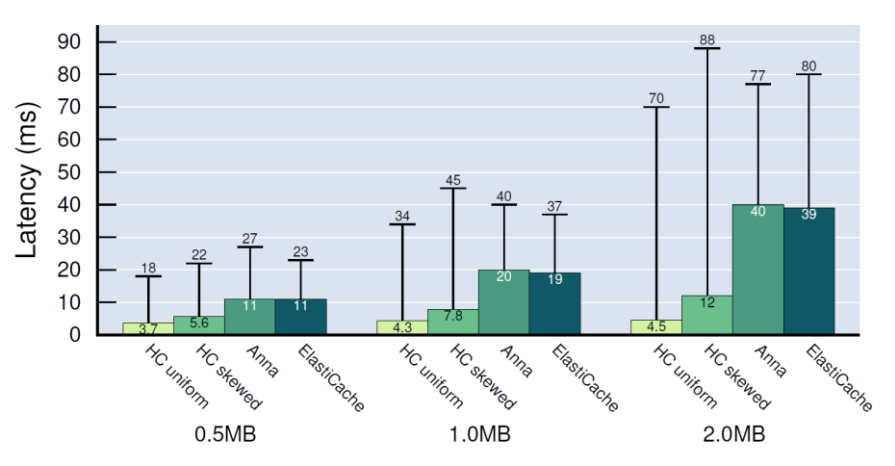
- Existem 4 aspetos a ter em conta:
  - Auto escalabilidade;
  - Leituras Repetidas;
  - Versões;
  - Readset desconhecido.

Avaliação Prática dos Protocolos





Avaliação Prática dos Protocolos



	Write Skew					
	Uniform	0.5	0.75	1.0	1.25	1.5
HydroCache	0%	0%	0%	0%	0%	0%
Anna	0.02%	0.4%	1.9%	6.6%	15%	21%
ElastiCache	0.03%	0.4%	2.0%	6.6%	14%	20%

20 Conclusão