P2P-Based Query Processing over Linked Data

Marcel Karnstedt Kai-Uwe Sattler Manfred Hauswirth

22 de Novembro de 2016

Giovanni Venâncio de Souza
gvs11ufpr@gmail.com

Linked Data

- Gerenciamento de dados públicos
- Perspectiva "*the world as a database*": Dados acessados e combinados de maneira uniforme
- Incentivo ao compartilhamento de dados
- RDF: Heterogeneidade, generalidade e flexibilidade

Desafios: Escalabilidade

- Volume de dados
- Quantidade de nodos
- Quantidade de usuários e de consultas

- Solução: Sistema distribuído
  - Adicionar nodos
  - Particionar os dados
  - Distribuir o processamento das consultas entre os nodos
- **Todos** os conjuntos de dados compartilhados e indexados em um único repositório de dados distribuído global
  - É necessário uma distribuição justa dos dados entre os nodos para garantir um bom desempenho

Desafios: Robustez e Disponibilidade

- Resiliência contra falhas na rede (nodos ou canais de comunicação)
- Solução: Canais de comunicação redundantes e replicação de dados
- Estratégia comum em sistemas P2P

Motivação

Como processar, de maneira eficiente, consultas complexas em *Linked Data* em um sistema amplamente distribuído (Web)?

Redes Peer-to-Peer

Redes P2P (*Peer-to-Peer*)

Uma rede é considerada *Peer-to-Peer* se os nodos compartilham seus recursos para fornecer o serviço que o sistema foi projetado. Os elementos da rede fornecem serviço para outros elementos e também fazem requisições do serviço para outros elementos [Camarillo et al., 2009].

Distributed Hash Table (DHT)

- Um tipo de sistema distribuído e descentralizado
- *Overlay Network* (Uma rede sobre outra rede – DHT sobre P2P)
- Armazena chave-valor (chave é um *hash* do valor)
- Busca valores através da chave (*lookup*)
- Armazenamento e *lookup* são distribuídos entre os nodos da rede
- *Lookup* tem custo $\mathcal{O}(\log n)$
- Tolerante a falhas e possui bom desempenho
- Possui diversas implementações!
  - Chord, CAN, Pastry, P-Grid ...

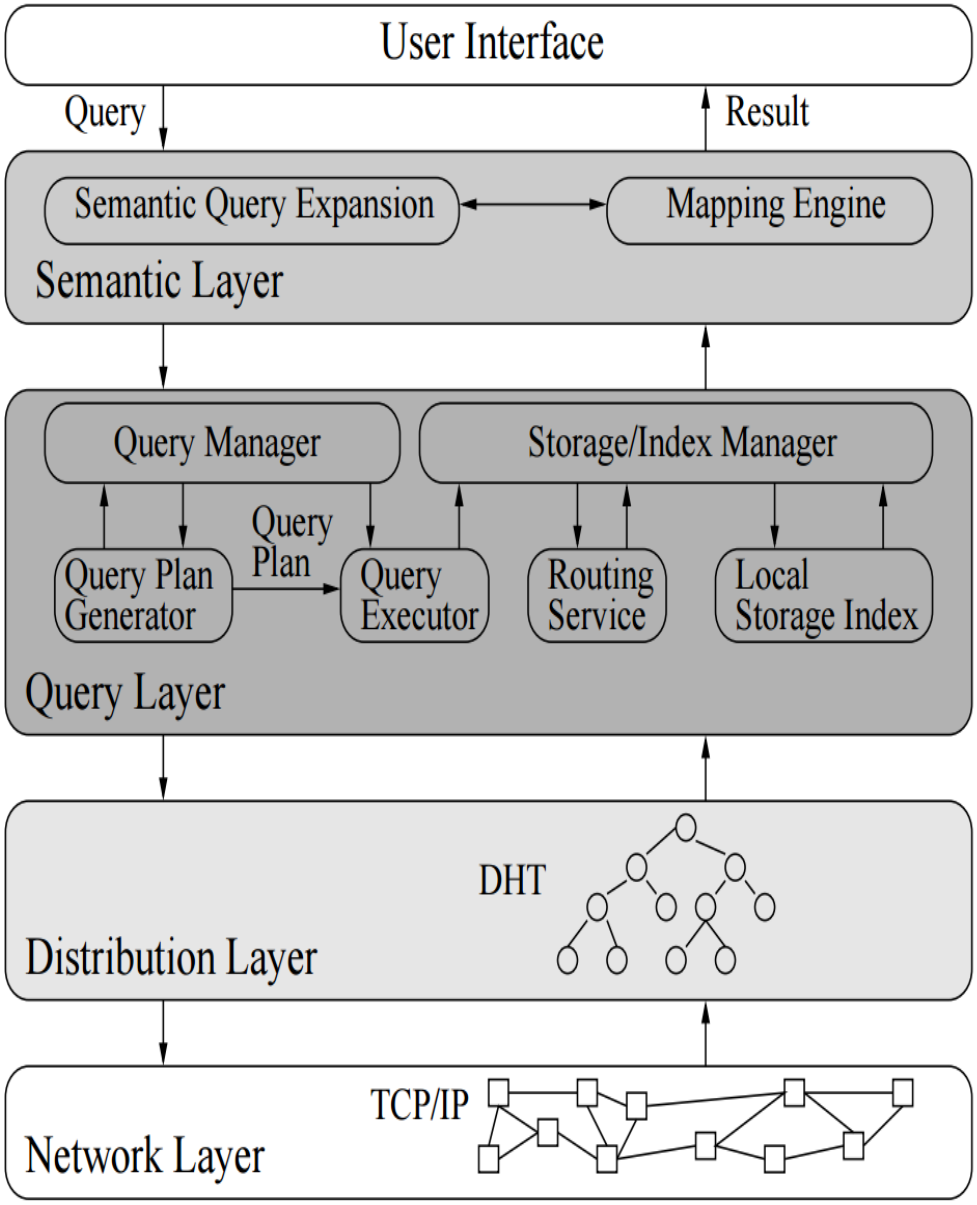
Vantagens de uma DHT

- Escalabilidade com relação a número de participantes, volume de dados e processamento de consulta
- Desempenho eficiente (mensagens, *hop complexity*)
- Mecanismos para organização automatizado
- Balanceamento dos dados

UniStore

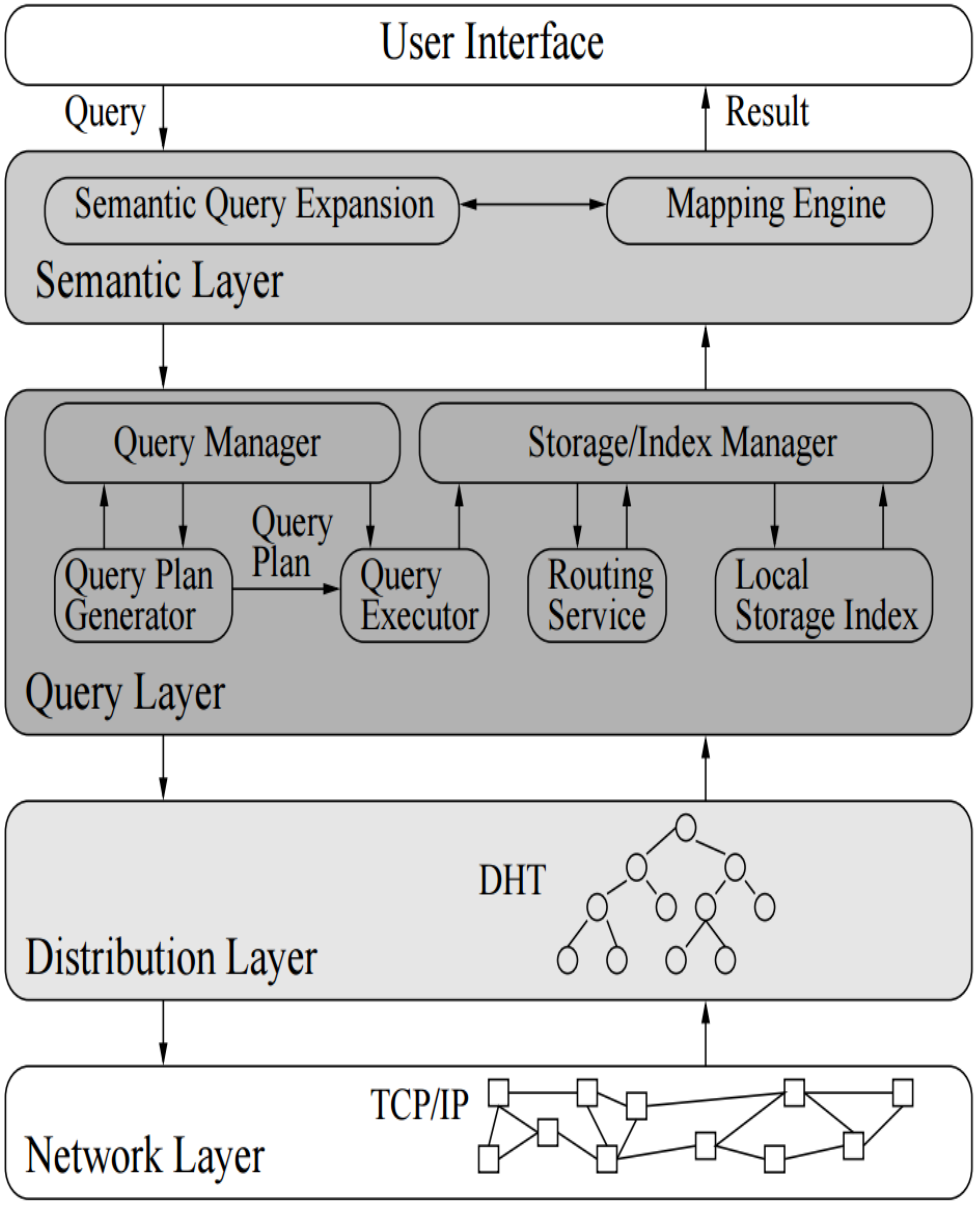
- Sistema para processamento distribuído de consultas (*SPARQL*) sobre RDF
- Armazenamento distribuído de triplas
- Funções adicionais
  - Ranking
  - Similaridade
- Processamento de consultas *ad-hoc* e *in-situ*
  - Resultado é completo e correto se contém todos os dados de todos os participantes disponíveis na situação atual
- DHT é utilizada *também* para rotear e processar consultas de uma maneira distribuída

UniStore: Arquitetura



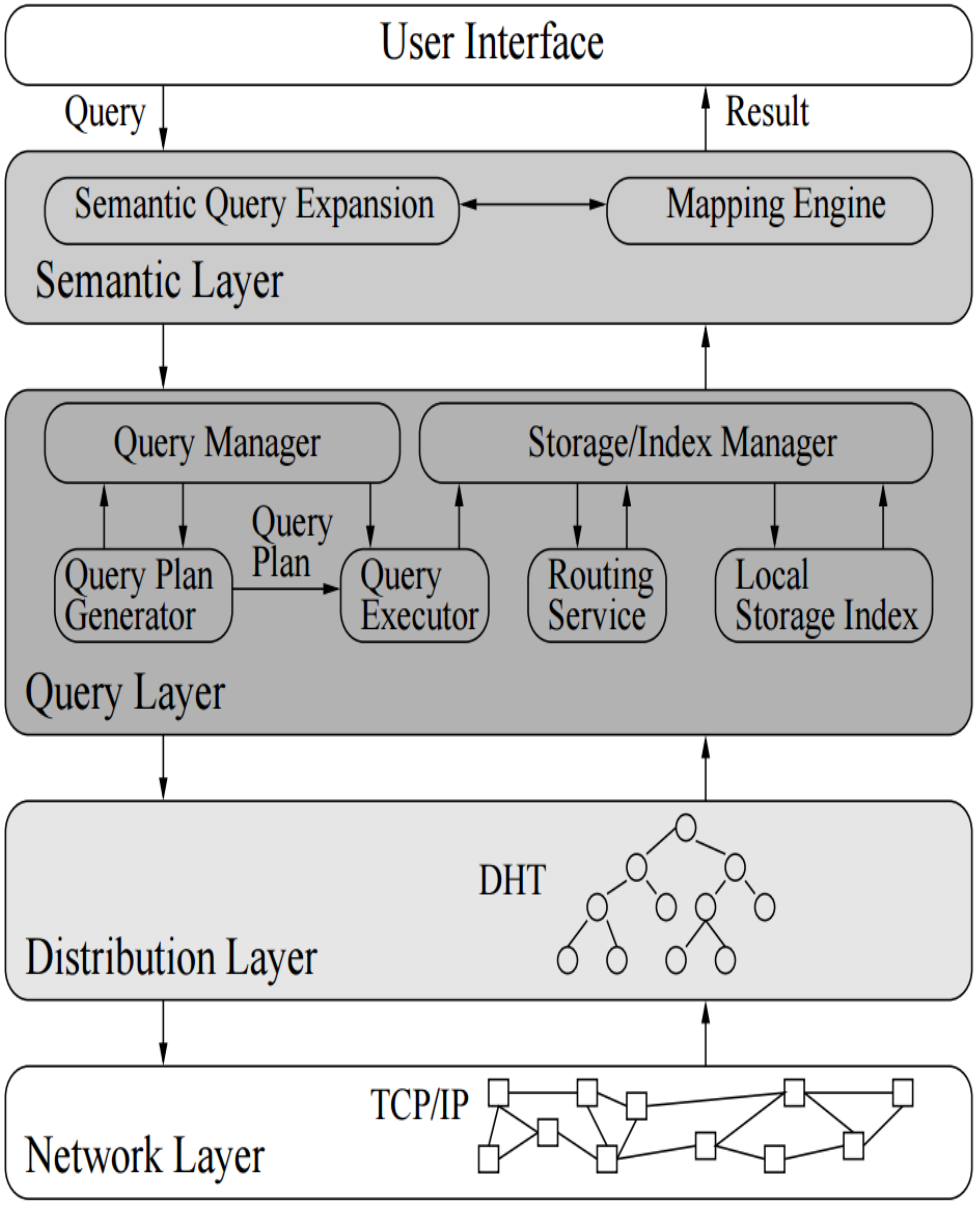
Camada de transporte para troca de mensagens

Arquitetura do UniStore



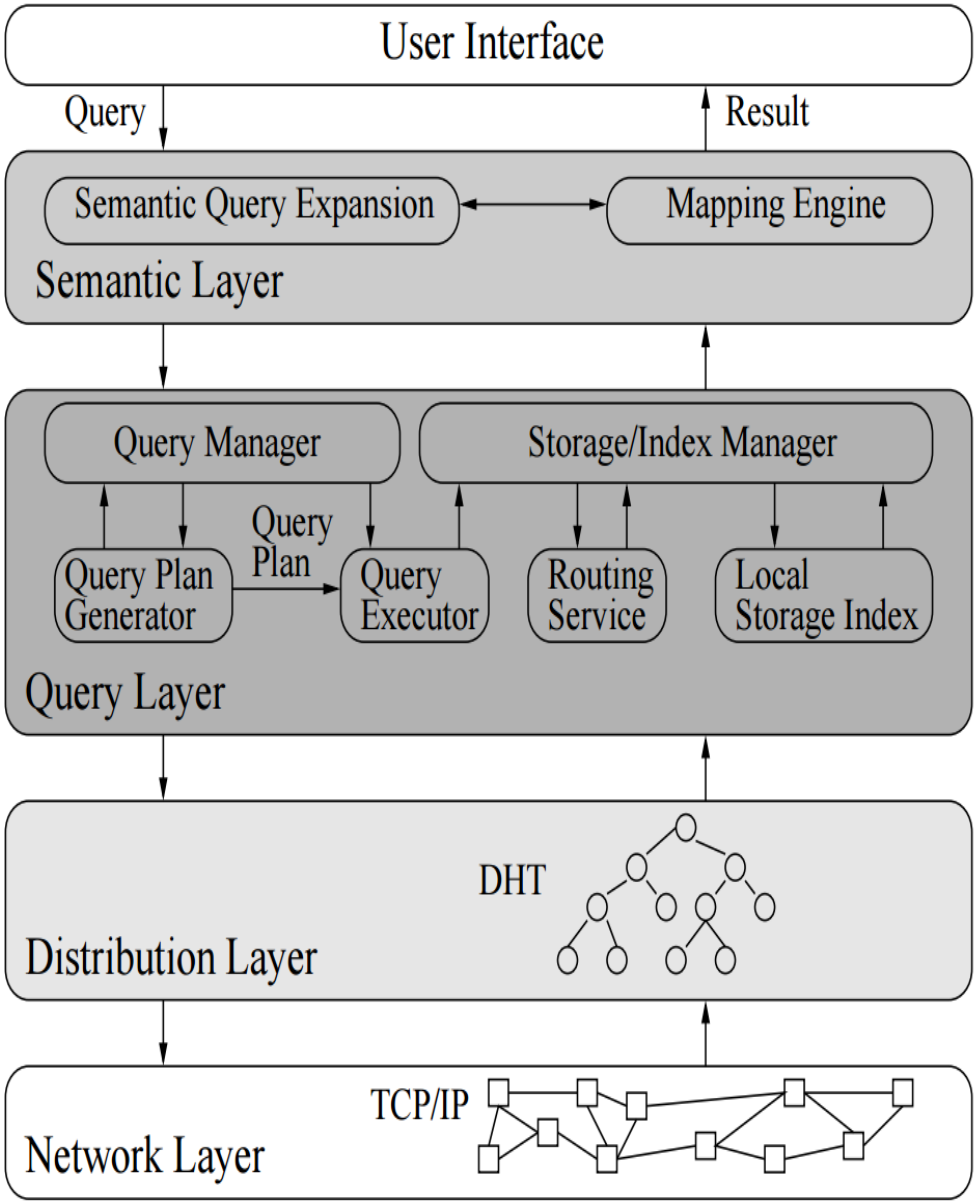
Distribuição transparente de dados, extração dos dados e estruturas de índices através de um sistema P2P

Arquitetura do UniStore



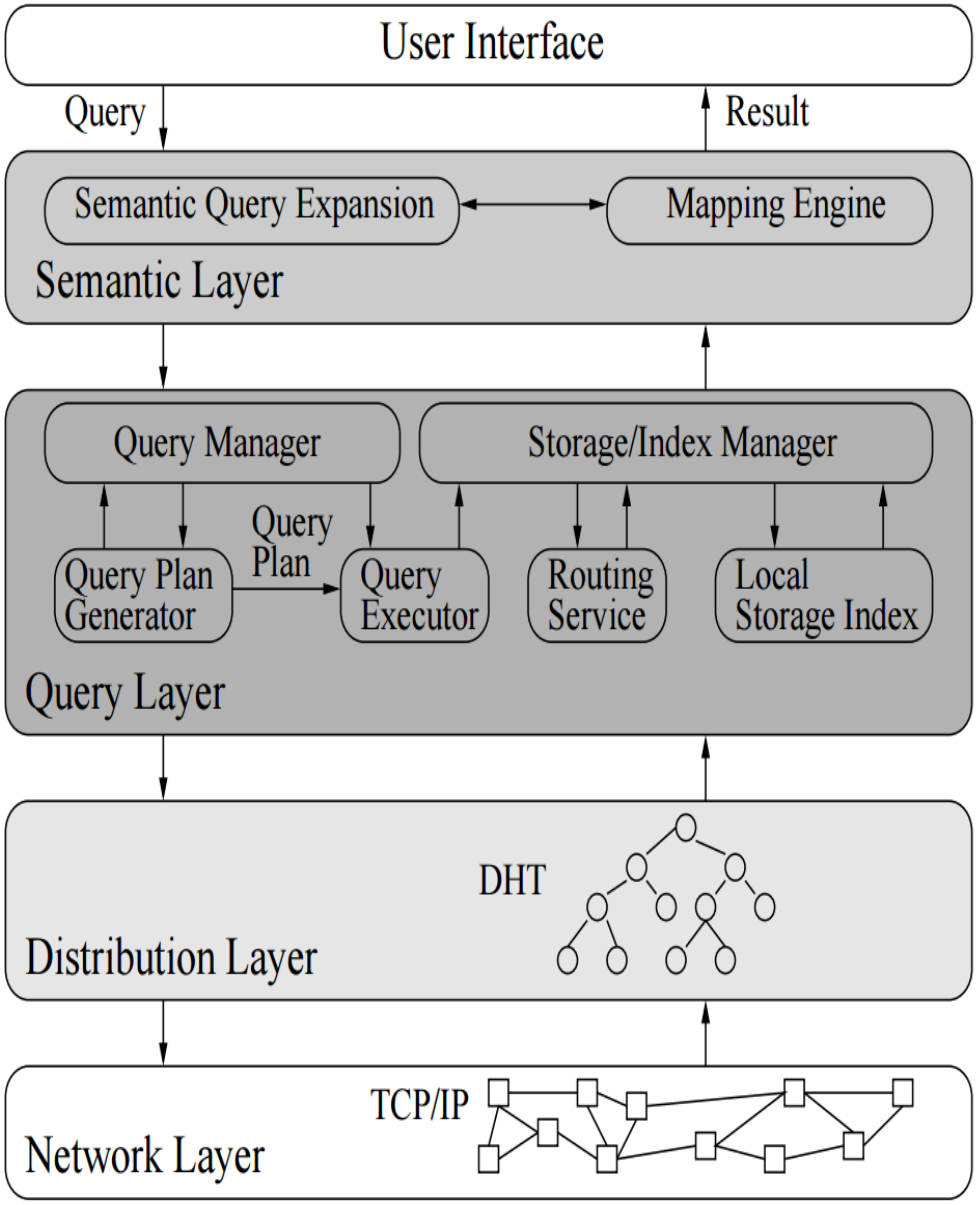
Indexação e mecanismos de consultas local e distribuído

Arquitetura do UniStore



Agrupamento virtual dos dados, relações semânticas e integração de dados

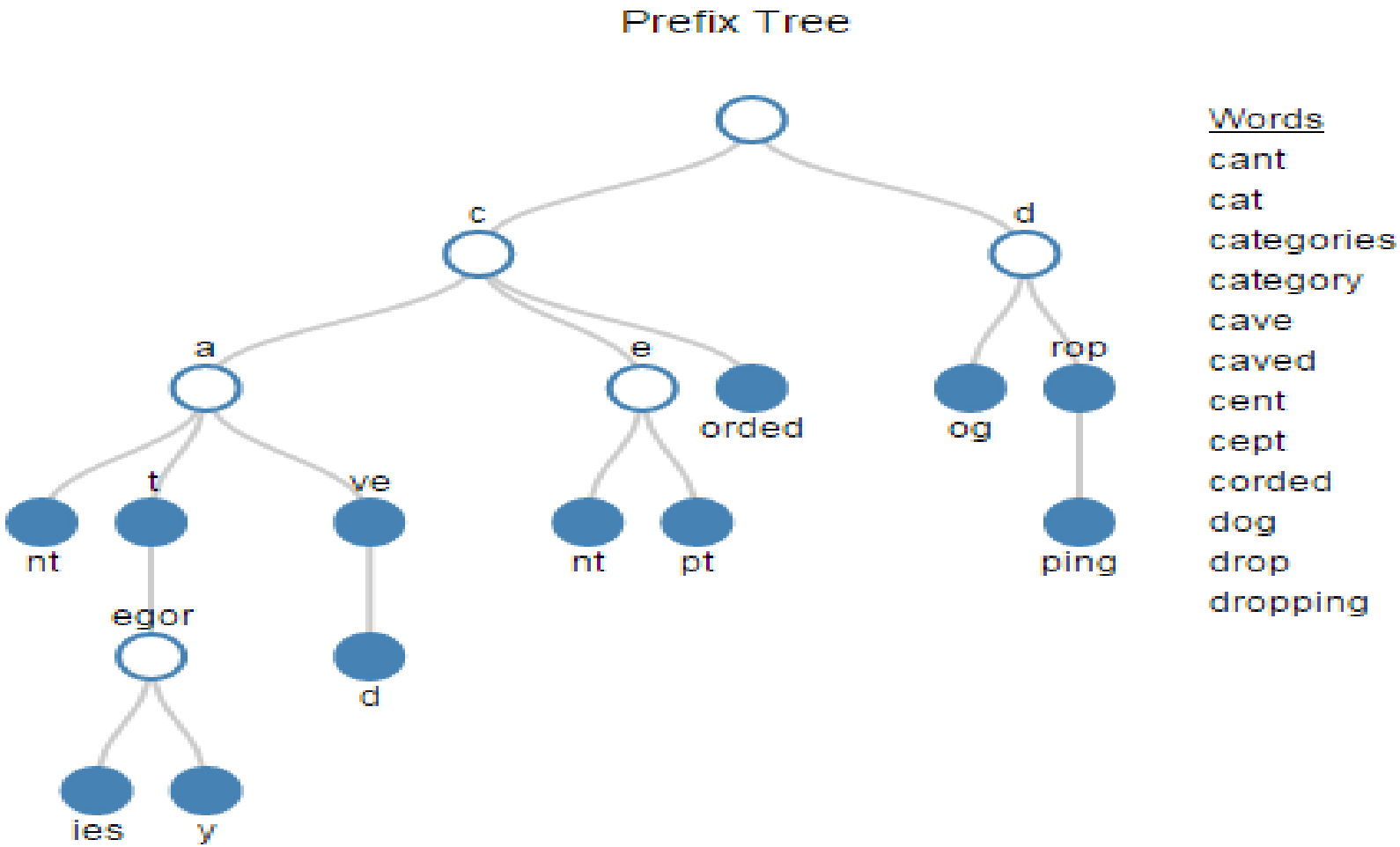
Arquitetura do UniStore



Acesso transparente para o usuário

Índice Distribuído

- *UniStore* utiliza a DHT *P-Grid* para armazenar índices de maneira distribuída
- *P-Grid* é baseado em uma árvore *Trie* (*Prefix Tree*) – dados ordenados!



Índice Distribuído

- *Hash* não é ordenado!
- $a < b \Rightarrow h(a) < h(b)$
- Buscas podem começar em qualquer nodo

Indexando dados RDF

- Tripla inserida na DHT através de uma chave k
- k é um *hash* de partes da tripla
- **Quais** partes da tripla devem ser indexadas?
- Triplas indexadas com a mesma chave estão no mesmo nodo
- Triplas indexadas com chaves vizinhas estão em nodos vizinhos (considerando a topologia da DHT, e não da rede!)
- Quantidade de índices depende da relação *desempenho* $\times$ *armazenamento*

Índices Utilizados

1. *h(s)*: Consultas sobre o sujeito
2. *h(p||o)*: Consultas que contém filtros ($<, =, >=$, ...) do tipo $p\theta c$
3. *h(o)*: Consultas sobre o objeto

Índices de Similaridade

- Similaridades entre *strings*
- Soluciona problemas de erros de digitação
- Baseia-se em funções de distância
- *Q-gram*: Pedaços de uma *string*
- Índices em *q-grams*
- Utilizado apenas para roteamento – *q-grams* não são armazenados nas triplas

Consulta

- Consulta que contém operações de similaridade
- Identificar potenciais URIs de fontes diferentes e que representam o mesmo objeto

```
1 select ?v1 ?s1 ?s2 ?c
2 where { ?s1 ?A ?v1 .
3         ?s2 ?B ?v2 ; <created> ?c .
4         filter ( edist (?v1 , 'Marcel') < 2 ) .
5         filter ( edist (?A, ?B) < 3 ) .
6         filter ( ?v1 = ?v2 ) }
```

Consulta

```
1 select ?v1 ?s1 ?s2 ?c
2 where { ?s1 ?A ?v1 .
3         ?s2 ?B ?v2 ; <created> ?c .
4         filter ( edist (?v1 , 'Marcel') < 2 ) .
5         filter ( edist (?A, ?B) < 3 ) .
6         filter ( ?v1 = ?v2 ) }
```

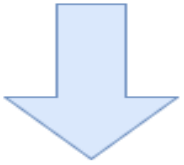
Consulta

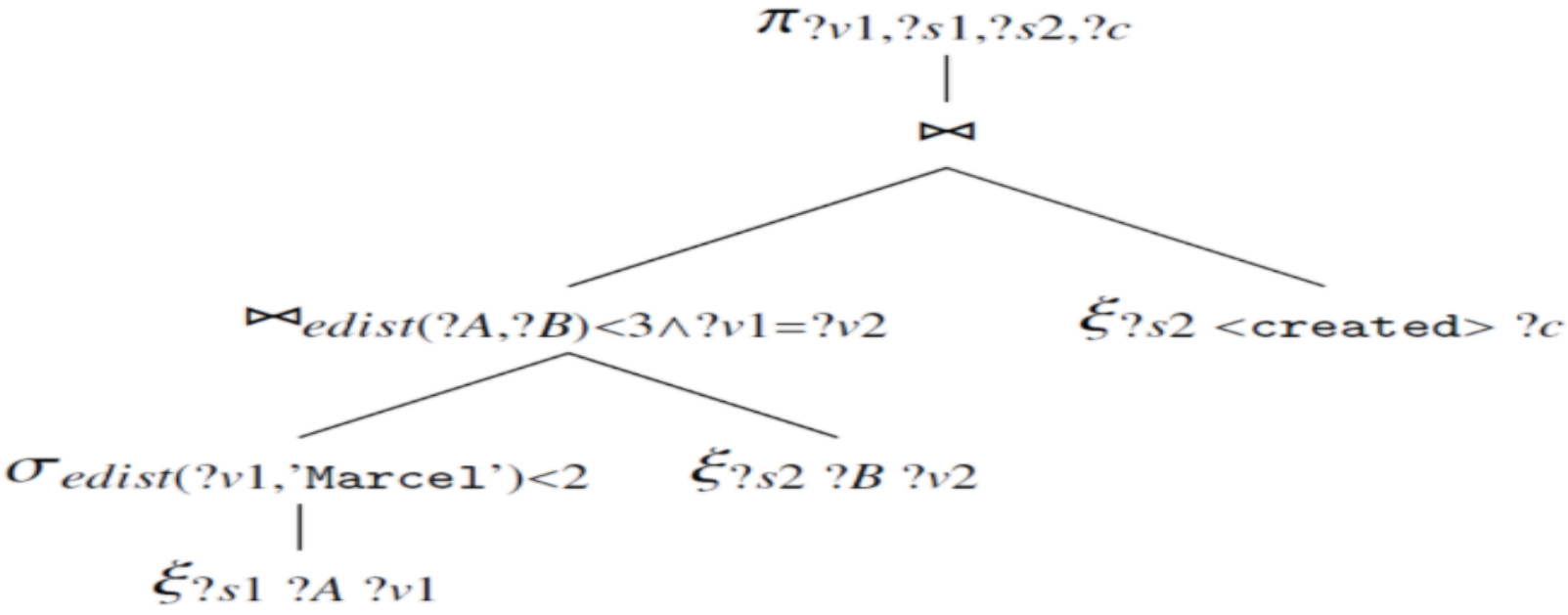
```
1 select ?v1 ?s1 ?s2 ?c
2 where { ?s1 ?A ?v1 .
3         ?s2 ?B ?v2 ; <created> ?c .
4         filter ( edist (?v1 , 'Marcel') < 2 ) .
5         filter ( edist (?A, ?B) < 3 ) .
6         filter ( ?v1 = ?v2 ) }
```

Consulta

```
1 select ?v1 ?s1 ?s2 ?c
2 where { ?s1 ?A ?v1 .
3         ?s2 ?B ?v2 ; <created> ?c .
4         filter ( edist (?v1 , 'Marcel') < 2 ) .
5         filter ( edist (?A, ?B) < 3 ) .
6         filter ( ?v1 = ?v2 ) }
```

```
select ?v1 ?s1 ?s2 ?c
where { ?s1 ?A ?v1 .
        ?s2 ?B ?v2 ; <created> ?c .
        filter (edist(?v1,'Marcel')<2) .
        filter (edist(?A,?B)<3) .
        filter (?v1=?v2)}
```





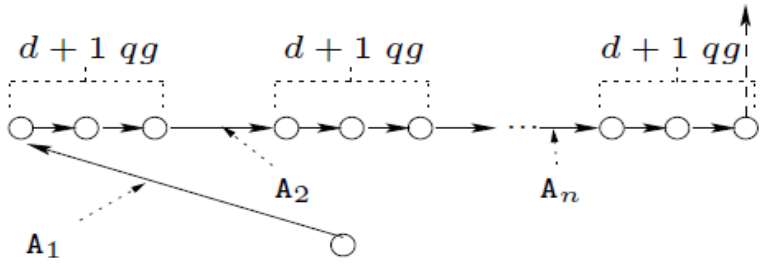
Álgebra Lógica e Operadores

- Consultas são formadas através de uma versão estendida do SPARQL
- Consultas são transformadas em um plano de consulta lógico
- Operador lógico especial ξ

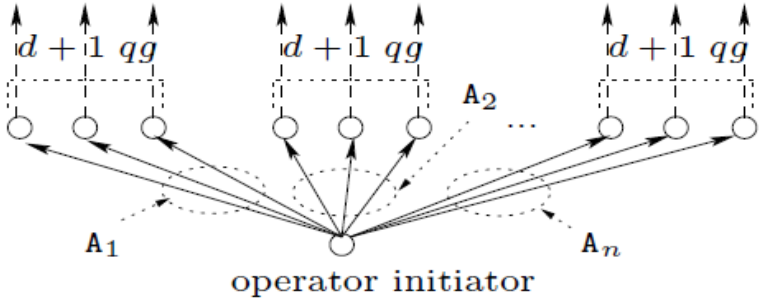
Álgebra Lógica e Operadores

- Operadores físicos dependem das funcionalidades fornecidas pela DHT
- Operadores físicos são classificados em:
  - op^{LOC}: dados disponíveis no nodo local
  - op^{SEQ}: comunica-se com os outros nodos em sequência
  - op^{PAR}: comunica-se com os outros nodos em paralelo
- Qual operador é utilizado depende da disponibilidade dos índices

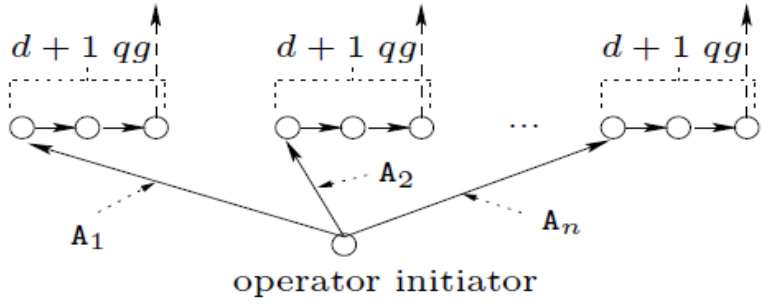
Álgebra Lógica e Operadores



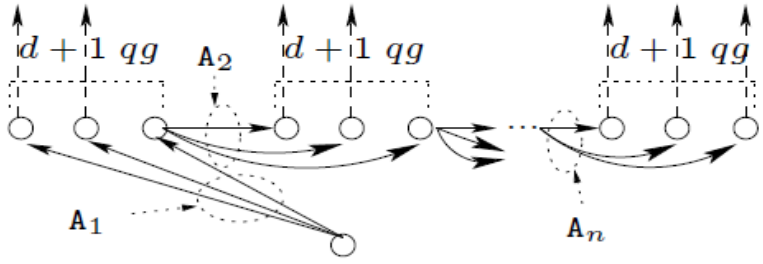
(a) $\bowtie^{HASH,SEQ}$ $(\xi^{QG,SEQ})$



(b) $\bowtie^{HASH,PAR}$ $(\xi^{QG,PAR})$



(c) $\bowtie^{HASH,PAR}$ $(\xi^{QG,SEQ})$



(d) $\bowtie^{HASH,SEQ}$ $(\xi^{QG,PAR})$

Operadores de Ranking

- Essencial para um conjunto grande de dados
- Permite analisar os resultados mais relevantes
- Duas classes de operadores de *ranking*: Top-*N* e *skylines*
- Top-*N*: Retorna apenas os *N* resultados mais importantes
  - Utiliza uma função de *ranking* baseada em uma ou mais dimensões
- *Skylines*: Baseia-se em dominância entre dimensões
- Utiliza apenas operações fornecidas pela DHT

Plano de Consulta

- Plano de consulta lógico $\rightarrow$ Plano de consulta físico
- Otimização de operadores lógicos e físicos
  - *N* operadores físicos para *M* operadores lógicos
- Modelo de custo para escolher a melhor combinação de operadores físicos
- Medidas: Número de *hops* e número de mensagens
- Considerar garantias da DHT (e.g., Complexidade logarítmica)
- Suporte a processamento de consulta adaptativo

Processamento de Consulta Adaptativo

- Problemas do planejamento estático: Mudanças de estado na rede, mudanças na DHT (entrada ou saída de nodos)
- Substituição dos operadores apenas quando ele for ser processado (operadores folha)

Execução de Consulta

- Processamento *bottom-up*
- Estratégia básica: Pós-ordem
- *Mutating Mutant Query Plans* (M^2QP)
- *Pipelining*

Mutating Mutant Query Plans (M^2QP)

- Várias cópias de um plano de consulta são enviadas pela rede
- Cada plano contém operadores que faltam ser processados e os dados de operadores que já foram processados
- Os planos de consulta são enviados de acordo com a pós-ordem
- Nodos podem duplicar e mudar a estrutura e/ou dados do plano
- Consequência: Maior número de mensagens

Pipelining

- Técnica comum em SGBD para melhorar desempenho de processamento de consulta
- Encaminha o resultado de um operador para o próximo assim que ele esteja disponível
- *UniStore* suporta *triple-set pipelining* e *peer pipelining*
- *triple-set pipelining*
  - Tripla processada pelos operadores
  - semelhante ao SGBD
- *peer pipelining*
  - Extensão do *triple-set pipelining*
  - Suporte a operadores que necessitam de mais de um *nodo* (e.g., $\bowtie$)
  - Envio de resultados intermediários

Conclusão

- Sistema de processamento de consulta em conjuntos distribuídos de dados RDF
- Sistema distribuído e escalável
- Utilização de DHT para armazenamento e roteamento de consultas
- Extensão do SPARQL: Similaridade e ranking

Perguntas

1. O sistema UniStore utiliza três índices: (i) no sujeito; (ii) no predicado concatenado com o objeto; (iii) no objeto. Na sua opinião, por qual motivo o predicado não é indexado?
2. Com relação a um sistema distribuído para processamento de consultas, cite quais são os desafios existentes e como uma DHT soluciona estes problemas.


Camarillo, G. et al. (2009).
Peer-to-peer (p2p) architecture: definition, taxonomies, examples, and applicability.
Technical report.