Load Awareness in FliX

Mohammad Alrifai

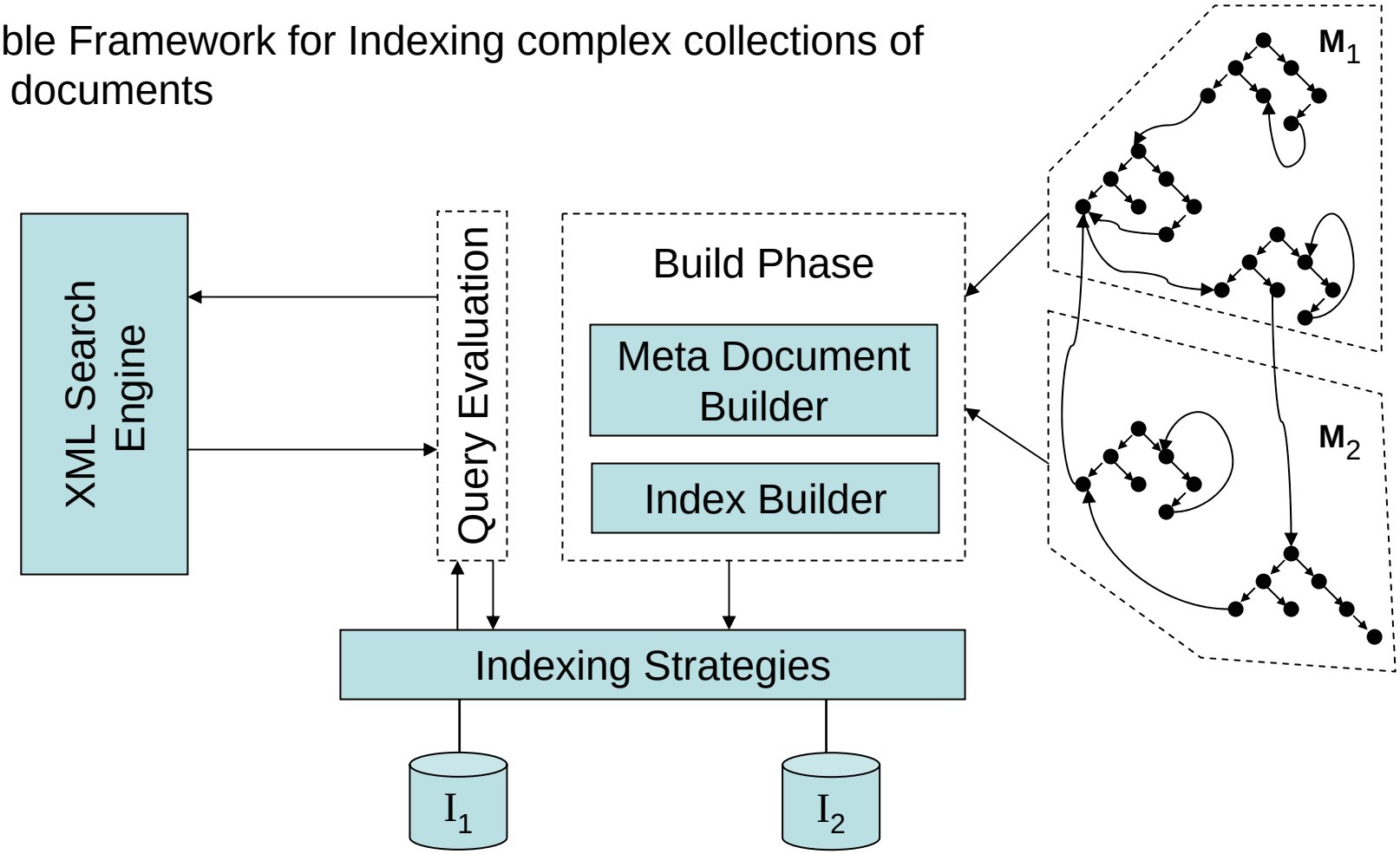
Supervisor: Dr. Ralf Schenkel

Overview

- What is FliX?
- Extended version of FliX
- Online Caching
- Load-aware Caching in FliX

What is FliX?

Flexible Framework for Indexing complex collections of XML documents

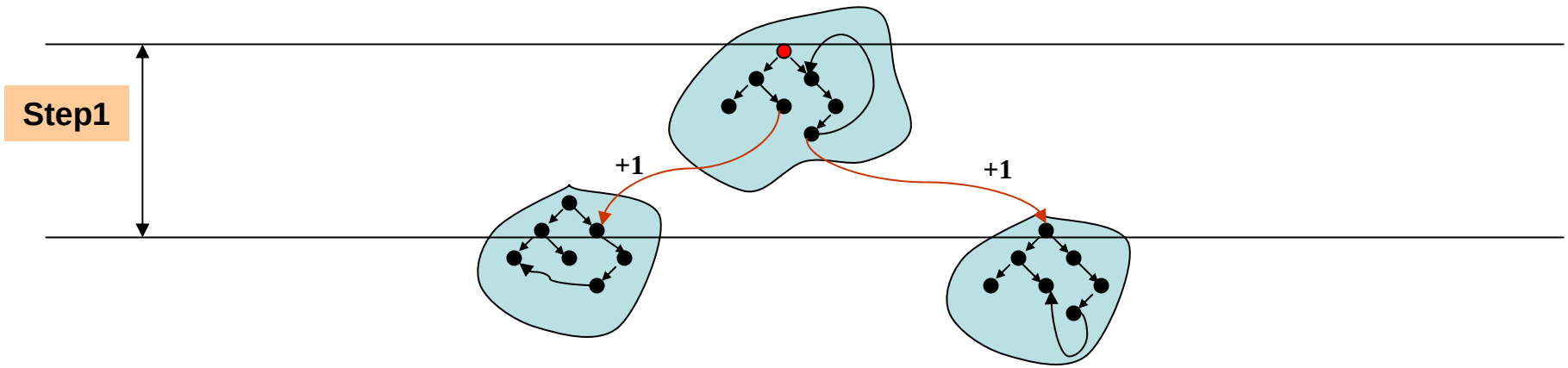


Types of queries

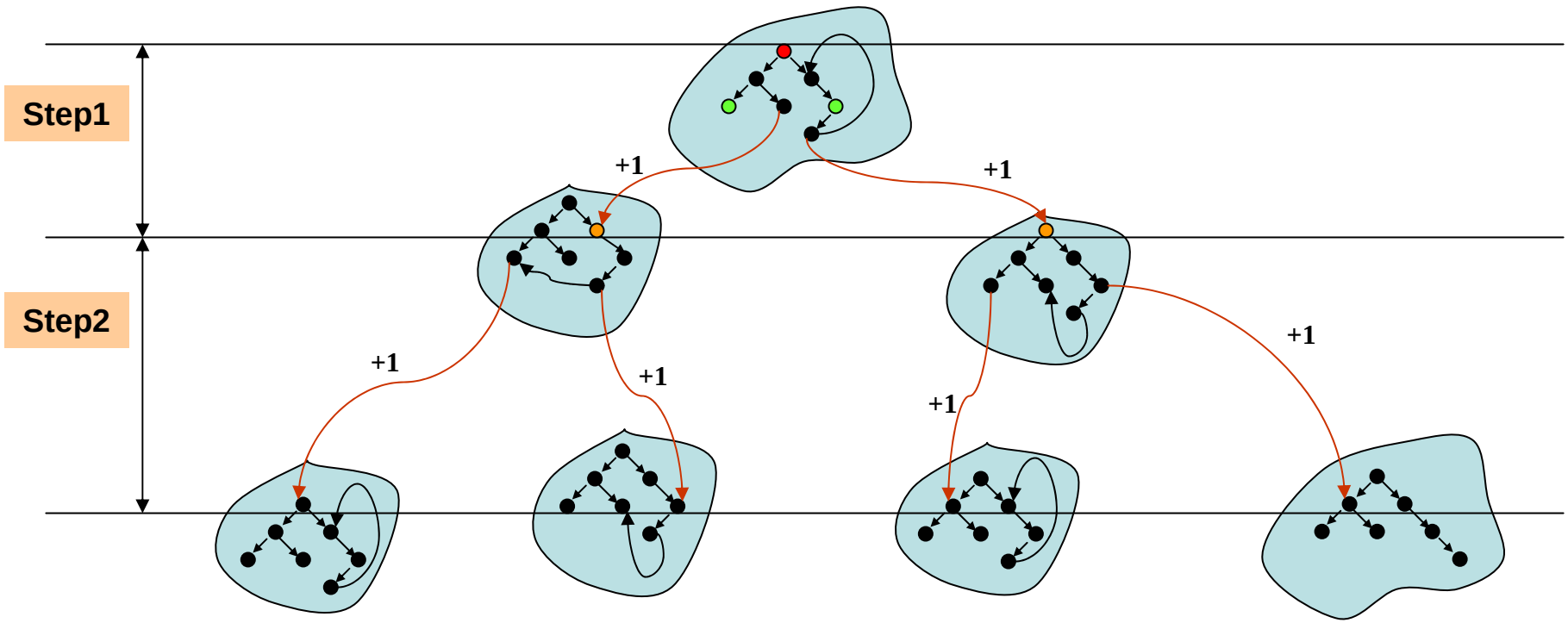
- Single-source query: ***s // T***
  - ***s*** is a single element
  - T is a tag name
  - Example: *ElementByID(53) // book*
  - Result of query is the set $R(q) = (s,b,d)$
  - b is a descendant element of ***s*** with tag name ***T*** and minimal distance d

- All-sources query: ***S // T***
  - ***S*** is a tag name
  - T is a tag name
  - Example: *author // book*
  - Result of query is the set $R(q) = (a,b,d)$
  - a is an element with tag name ***S***
  - b is a descendant element with tag name ***T*** and minimal distance d

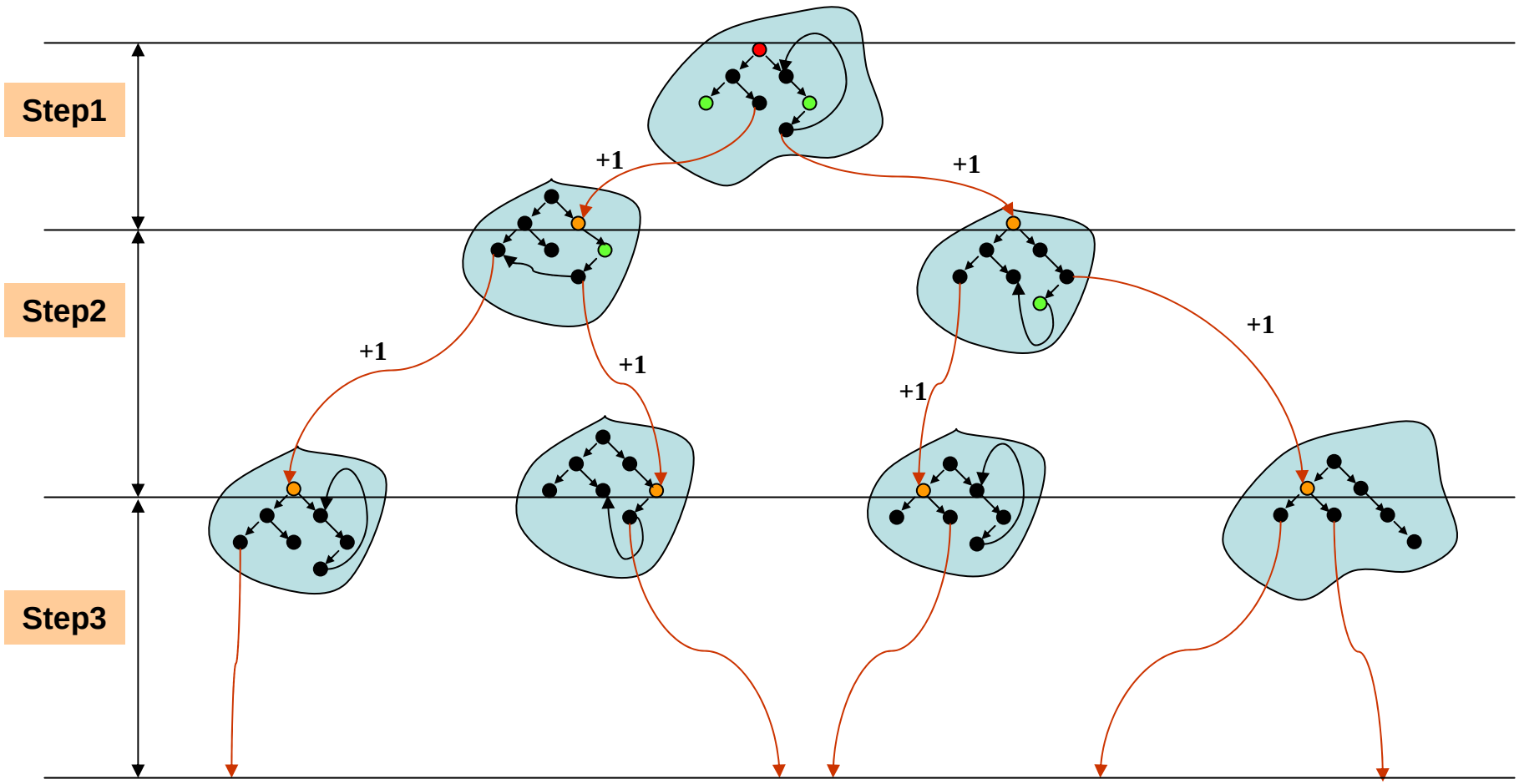
Query Evaluation in FliX



Query Evaluation in FliX



Query Evaluation in FliX



Query Load

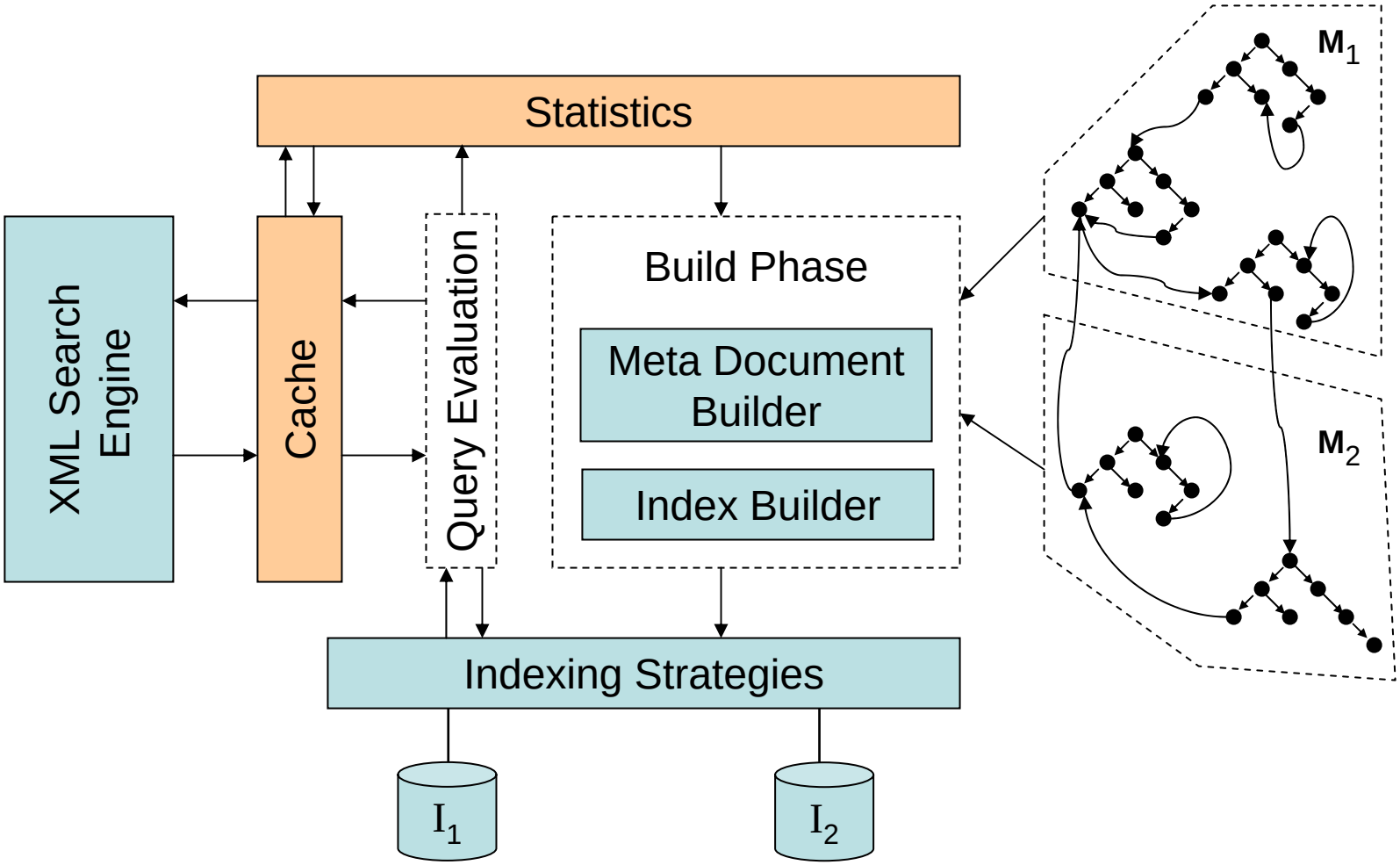
- Query Load $QL = \{q_1,\ldots,q_N\}$, N = query load size
  - q_i is either of type $(s \,//\, T)$ or $(S \,//\, T)$
  - $R_i \subseteq R(q_i)$ is the subset of query results that the client actually read
  - We consider a fixed window of the query load of size W
  - The *absolute frequency* of a query $f(q_i) = |\{k : QL(k) = q_i\}|$
  - The *relative frequency* of a query $rf(q_i) = \frac{f(q_i)}{W}$
  - Total cost of the query load QL: $c(QL) = \sum_{i=1}^{N} c(q_i)$

$$= \sum_{q \in QL} f(q_i) * c(q)$$

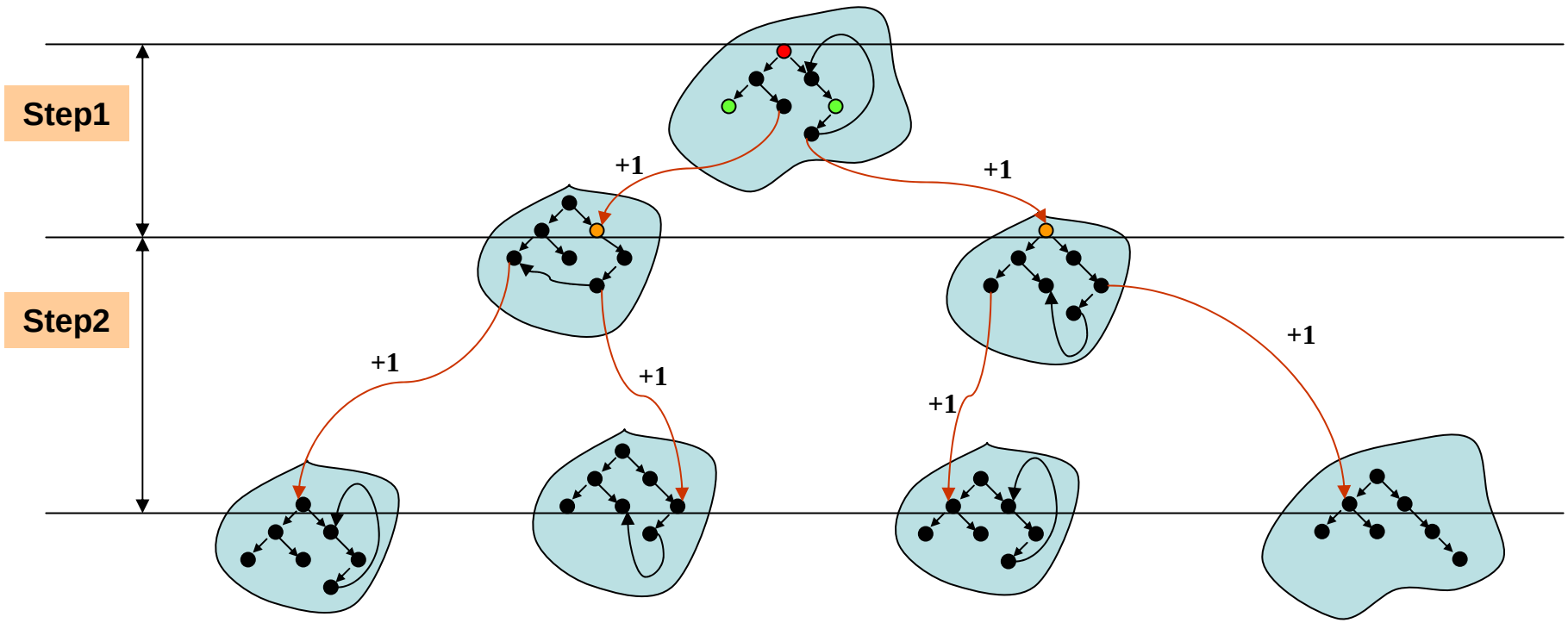
Goal → minimize the total cost of query load

Proposed solution → cache results of frequent queries

Extended version of FliX



Query Evaluation steps in FliX



- Cache query results (green colored nodes)
- cache source nodes of next step (orange colored nodes)
- Keep meta information about each step (cost, # of results, max distance, etc)

Online Caching (background)

- Online vs. offline caching algorithms
  - Offline: future requests are known
  - Online: future requests can not be predicted
- Caching models:
  - Bit Model: cost (object) = size (object)
  - Fault Model: cost (object) =1, size (Object) =arbitrary
  - Cost Model: cost (object) = arbitrary, size (object)=1
  - General Model: both cost (object) , size (object) = arbitrary

Caching in FliX : online , general model

Online Caching (background)

- Caching problem
  - Given a cache with a specified size *k*
  - Given a sequence of requests to objects

Optimal replacement policy is needed!

s

s

- the total retrieval cost of all requests is minimized.
- the total size of objects in the cache is at most *k*.

Caching algorithm

- Some well-known algorithms (replacement policies) :
  - FIFO (first cached is first replaced)
  - LRU (least recently used is replaced)
  - LandLord (frequency +cost + size)

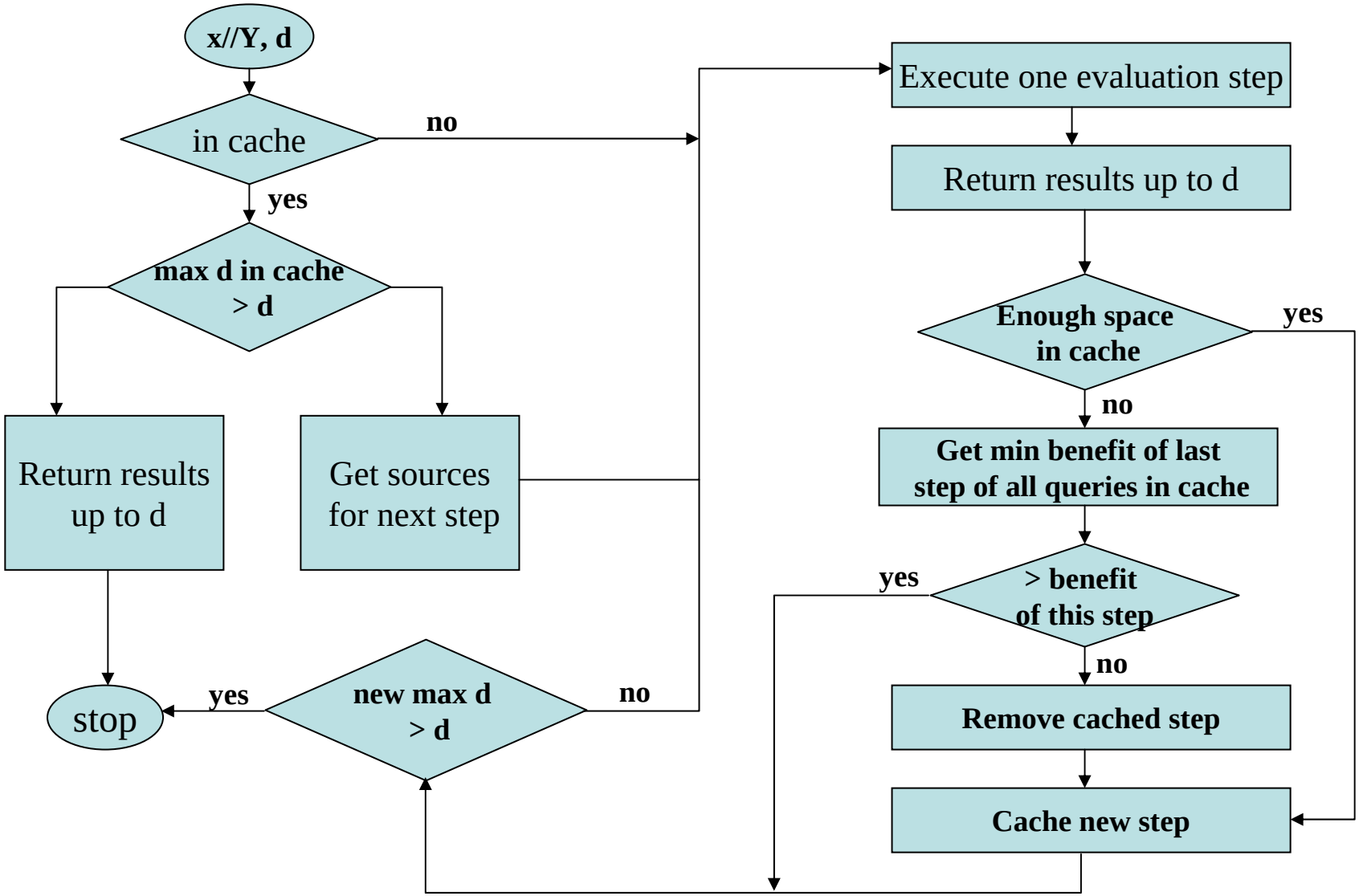
Caching in FliX : Hybrid (LRU + frequency + cost + size)

replace query with minimum *benefit* , where:

$$benefit(q) = \frac{rf(q).c(q)}{a(q).|R(q)|}$$

- $rf(q)$: relative frequency of q
- $c(q)$: cost (i.e. evaluation time) of q
- $|R(q)|$: # of result nodes of q
- $a(q)$: age (i.e. how far is the last occurrence) of q

Load–aware caching algorithm in FliX



Cache structure

Query load

query	Distance
Q1	3
Q2	2
Q3	6
Q4	5
Q2	3
Q4	4
Q1	6
Q2	5