Generating Efficient Execution Plans for Vertically Partitioned XML Databases

Research paper review by

QING Pei, Edward	11500811g
LO Wing Yi, Wing	11523479g
SHAO Shuai, Philip	11552402g

April 10, 2012

What ?

Why ?

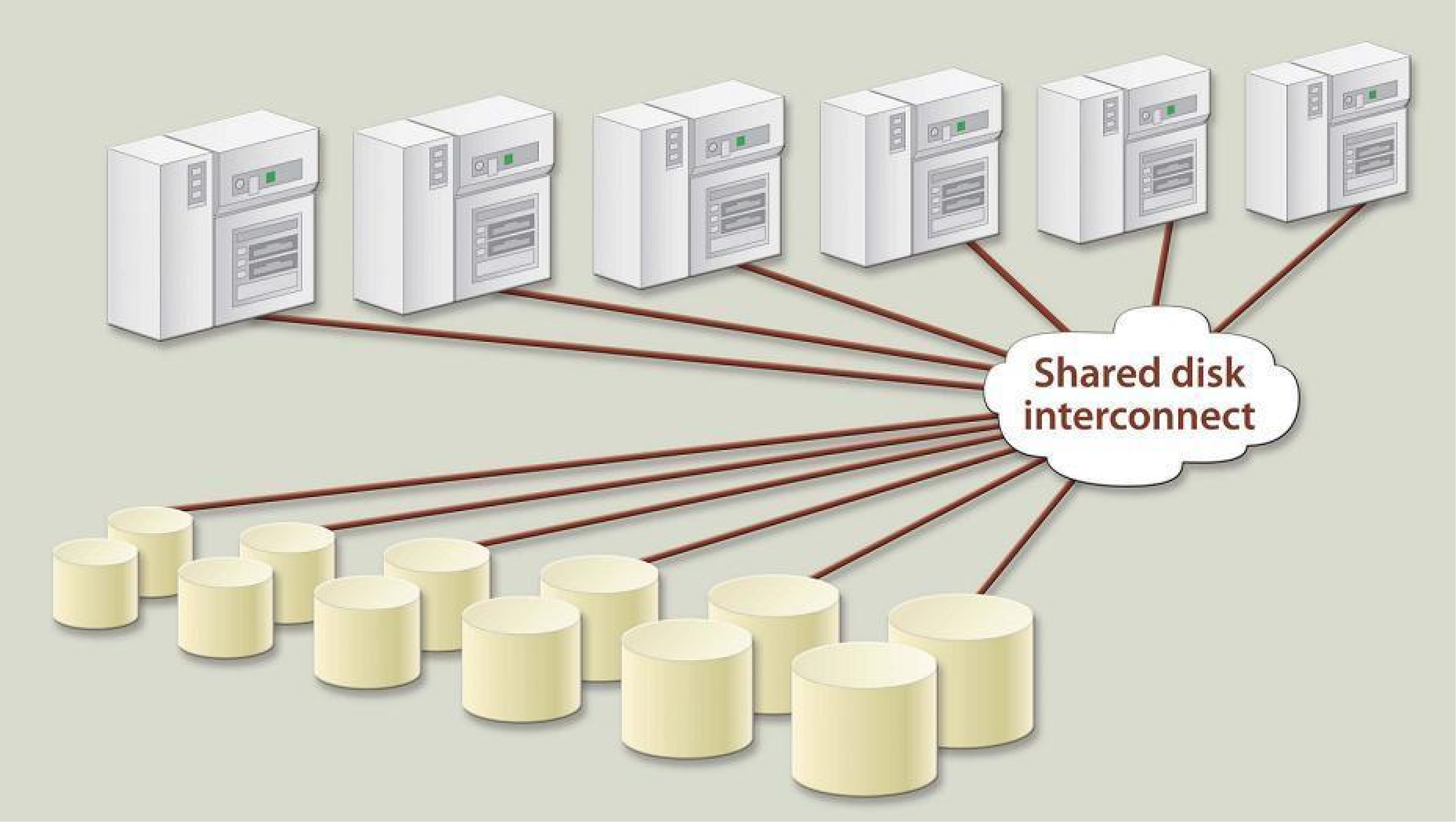
How ?

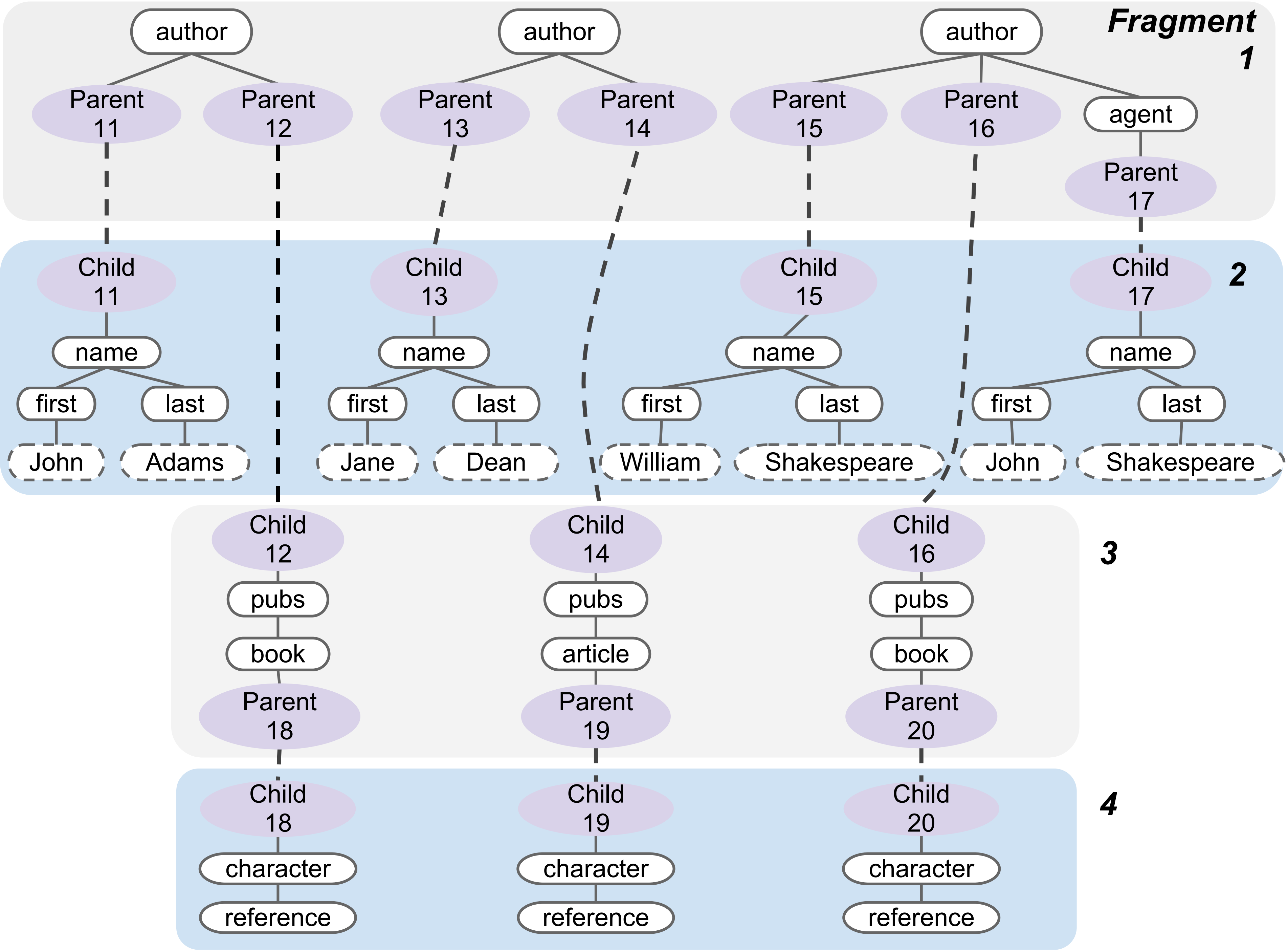
What ?

Query Processing

	Centralized	**Distributed**
RDBMS	✔	✔
XML	✔	***This paper***

XML in the Cloud




Fragment 1
author
Parent 11
Parent 12
author
Parent 13
Parent 14
author
Parent 15
Parent 16
agent
Parent 17
2
Child 11
name
first
last
John
Adams
Child 13
name
first
last
Jane
Dean
Child 15
name
first
last
William
Shakespeare
Child 17
name
first
last
John
Shakespeare
3
Child 12
pubs
book
Parent 18
Child 14
pubs
article
Parent 19
Child 16
pubs
book
Parent 20
4
Child 18
character
reference
Child 19
character
reference
Child 20
character
reference

Why ?

Distributed architecture leads to Different execution plans

For a single query, the **order** in which *joins* are performed results in various time consumed.

Response time

=

local execution time

+

joining time

local execution time

snip(i): the number of document subtrees accessed by the local plan at *fragment i*

smaller snip(i) preferred

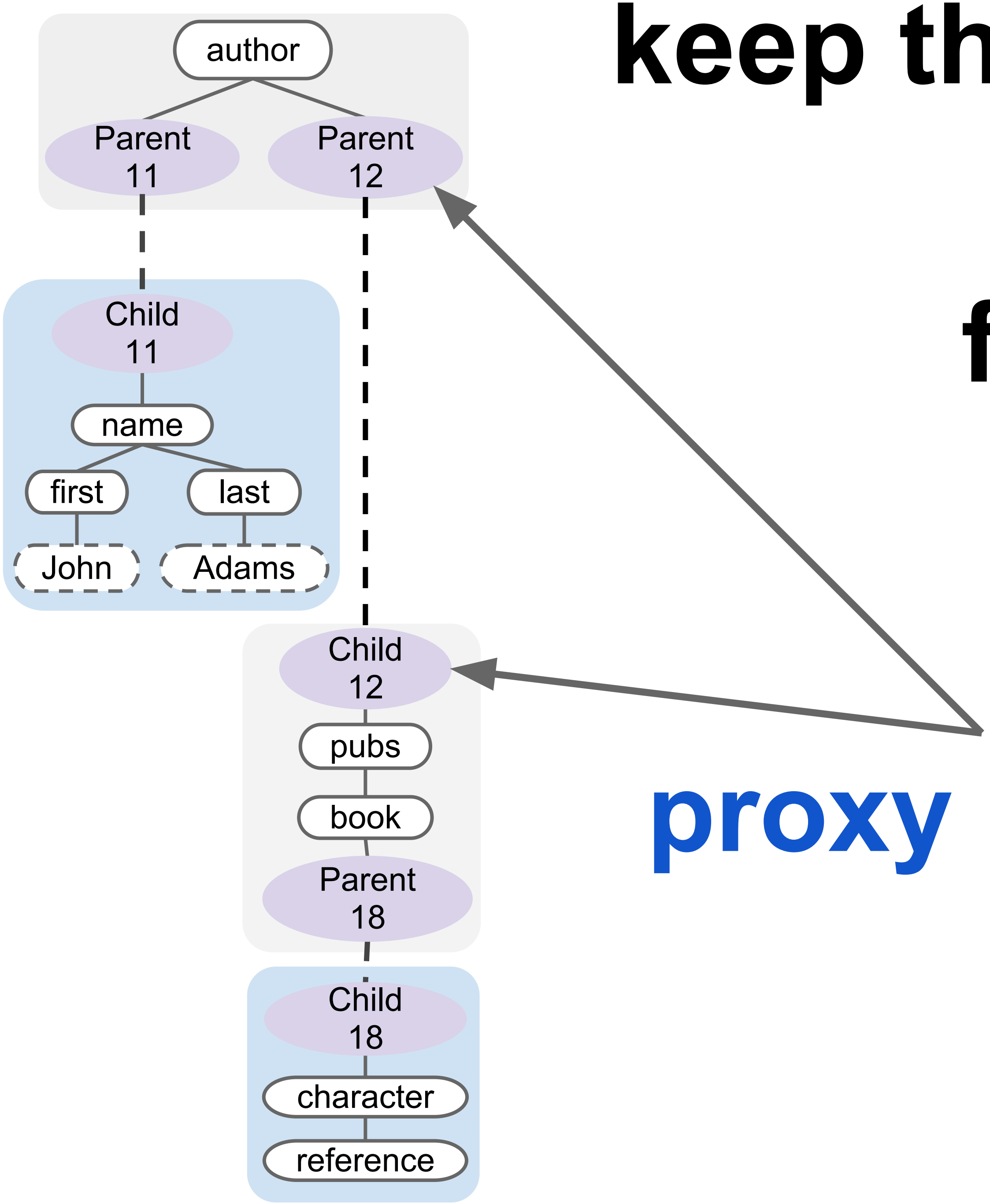
joining time

card(i): the number of tuples that are returned by the local plan when evaluated at *fragment i*

smaller card(i) preferred

Which **plan has the** ***minimum*** **response time?**

How ?



keep the relation between fragments

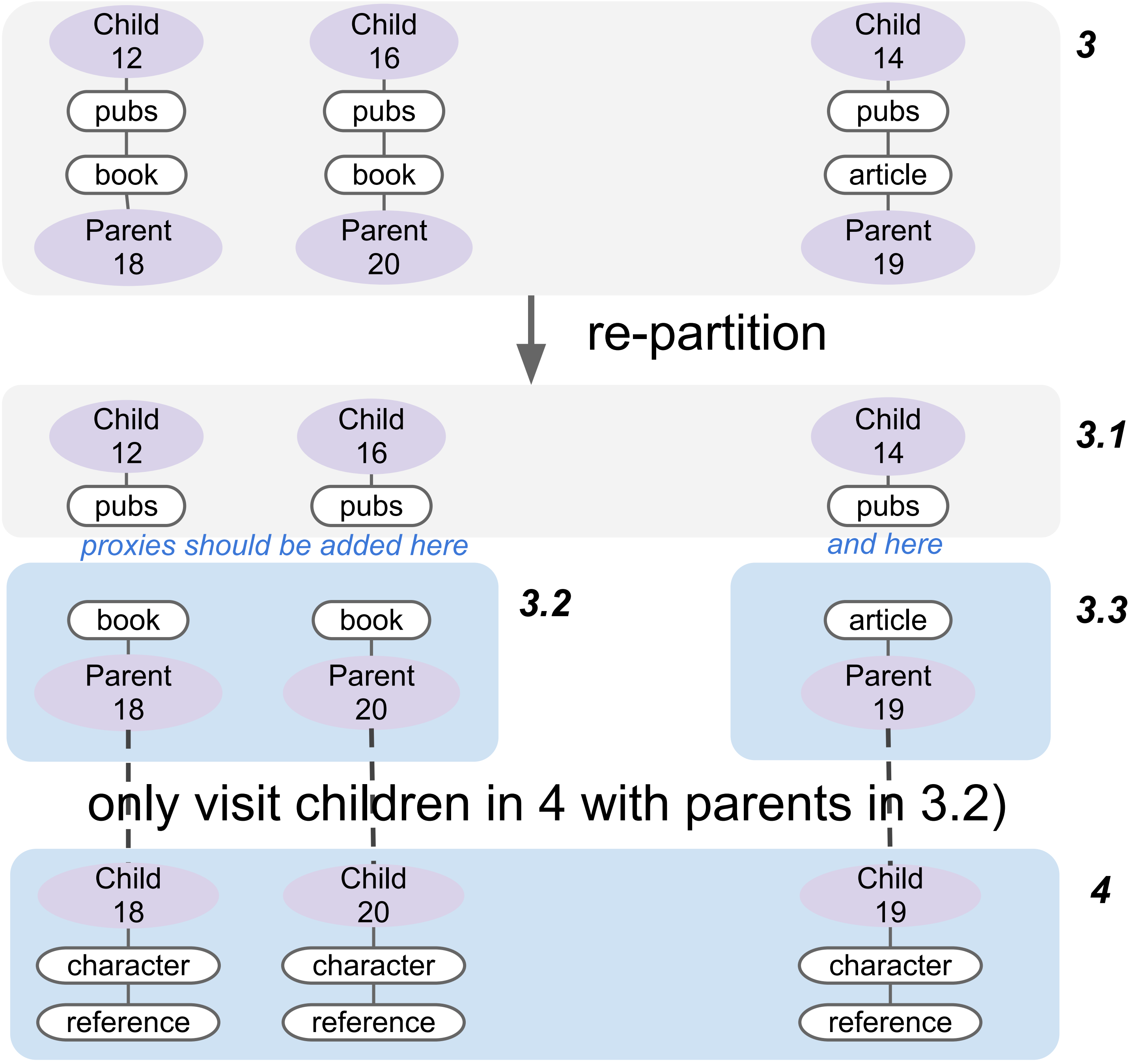
proxy nodes

Optimizing distributed plans

Optimizing distributed plans

Pushing Cross-Fragment Joins

fully works on left-deep plans



re-partition

proxies should be added here

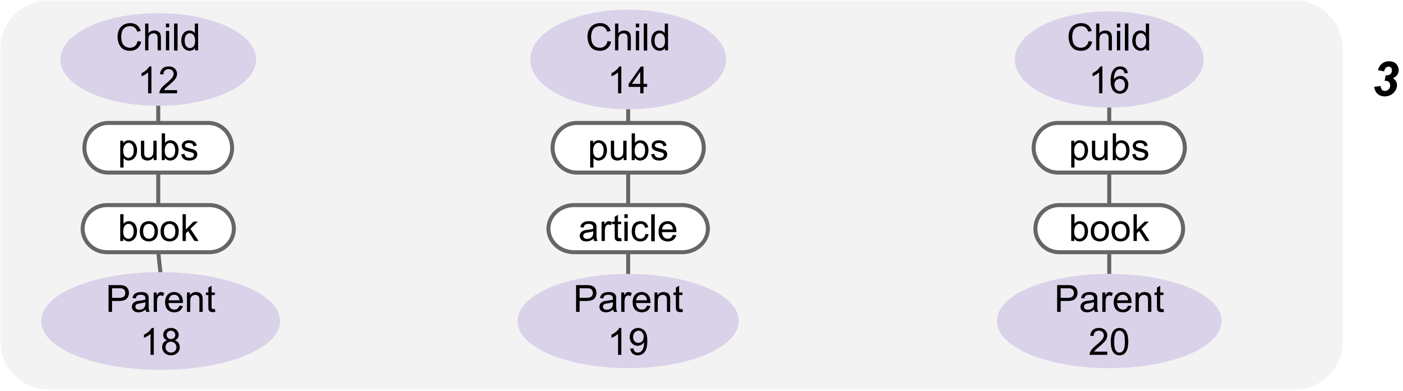
and here

only visit children in 4 with parents in 3.2)

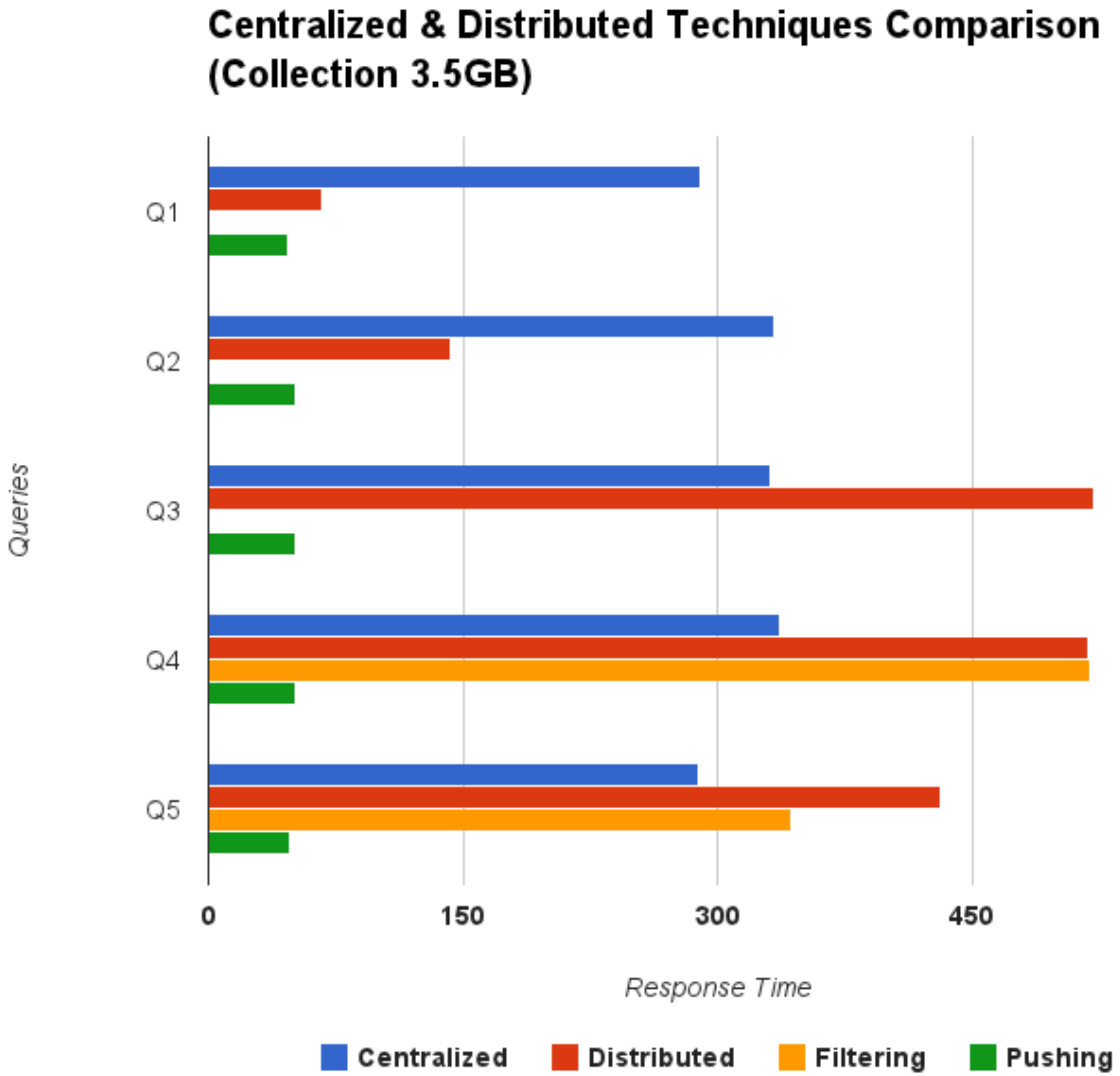
Optimizing distributed plans

Label Path Filtering

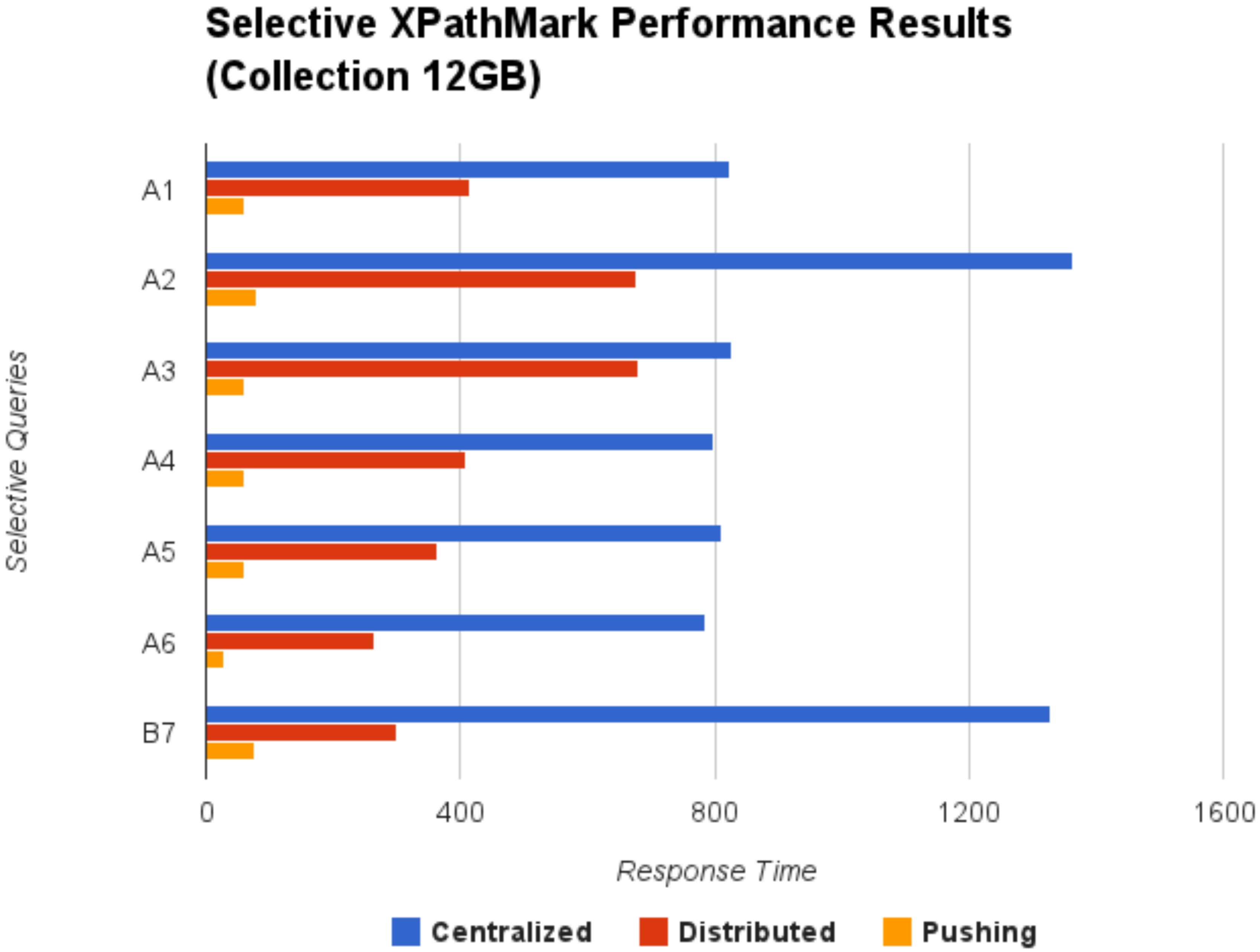
//book//reference



Evaluation



Centralized & Distributed Techniques Comparison (Collection 3.5GB)



Selective XPathMark Performance Results (Collection 12GB)

Conclusion

Greatly improves response time of querying large XML collections.

Small overhead. Choosing the fastest plan took **< 0.01** seconds.

Q & A

Merci beaucoup