



# Efficient Distribution of Full-Fledged XQuery

Y. Zhang N. Tang P. Boncz  
CWI Amsterdam, the Netherlands

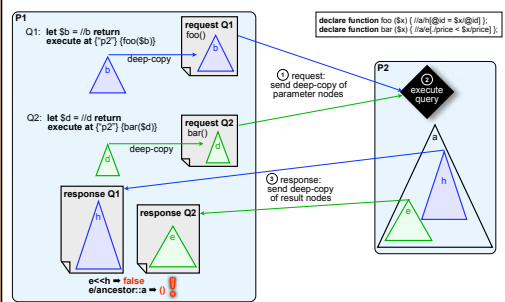


## Goals

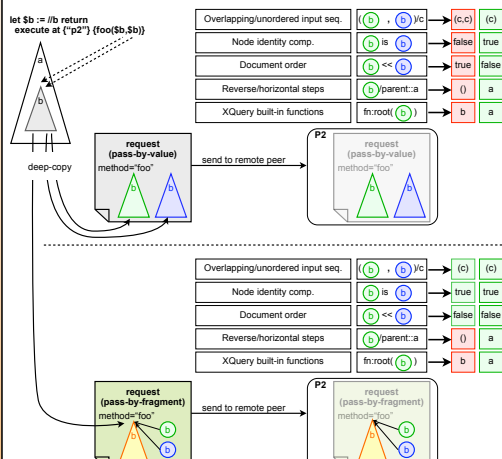
We investigate techniques to automatically decompose any XQuery query into sub-queries, that can be executed near their data sources, i.e., function-shipping. In this scenario, the sub-queries being executed remotely may have XML node-valued parameters or results, that must be shipped in some way. The main challenge addressed here is to ensure that the decomposed queries properly respect XML *node identity* and preserve *structural properties*, when (parts of) XML nodes are sent over the network, effectively copying them.

## XQuery Remote Execution (pass-by-value)

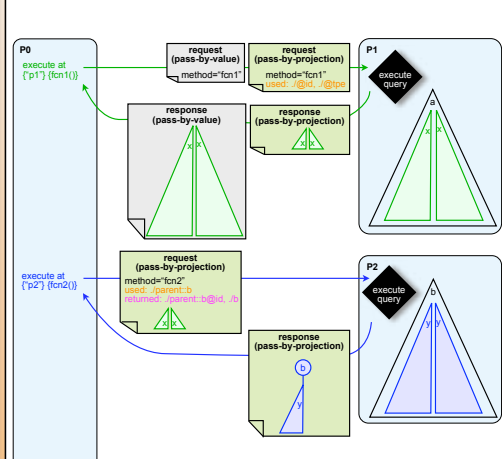
- Node identities: lost when deep copies of XML nodes are exchanged among peers
- Structure properties: modified when XML nodes are cut off from their original ancestors and siblings



## Semantic differences (pass-by-value vs. pass-by-fragment)



## Remote Execution of Q<sup>c</sup> (pass-by-projection vs. pass-by-value/pass-by-fragment)



## Q: a basic XQuery query

```
let $y2 := for $x in doc('xpc:/p1/a.xml')/a/x,
           $y in doc('xpc:/p2/b.xml')/b/y
           where ($x/@id = $y/@id and $x/@tpe = $y/@tpe)
           return $y
return ($y2/b/@id, $y2/b)
```

### Q<sup>c</sup>: XQuery Core variant of Q

```
let $y2 := for $x in doc('xpc:/p1/a.xml')/child::a:child::x return
           for $y in doc('xpc:/p2/b.xml')/child::b:child::y return
           if ($x/@attribute:id = $y/@attribute:id and $x/@attribute:tpe = $y/@attribute:tpe)
           then $y else ()
return ($y2/parent::b/@attribute:id, $y2:child::b)
```

### Q<sup>c</sup>: decomposed Q<sup>c</sup> under pass-by-value (and pass-by-fragment)

```
declare function fcn1() as node()* {
  doc('xpc:/p1/a.xml')/child::a:child::x;
}
declare function fcn2() as item* {
  let $y2 := for $x in execute at ["p1"] (fcn1()) return
             for $y in doc('xpc:/p2/b.xml')/child::b:child::y return
             if ($x/@attribute:id = $y/@attribute:id and $x/@attribute:tpe = $y/@attribute:tpe)
             then $y else ();
  return ($y2/parent::b/@attribute:id, $y2:child::b);
}
fcn2()
```

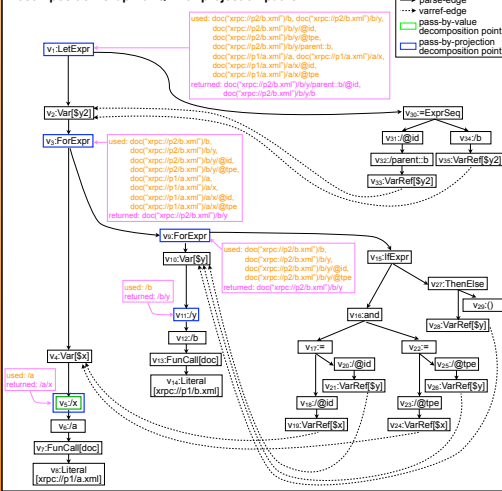
### Q<sup>c</sup>: decomposed Q<sup>c</sup> under pass-by-projection

```
declare function fcn2($x as node()) as node()* {
  for $y in doc('xpc:/p2/b.xml')/child::b:child::y return
  if ($x/@attribute:id = $y/@attribute:id and $x/@attribute:tpe = $y/@attribute:tpe)
  then $y else ();
}
declare function fcn1() as node()* {
  doc('xpc:/p1/a.xml')/child::a:child::x;
}
let $y2 := for $x in execute at ["p1"] (fcn1()) return execute at ["p2"] (fcn2($x))
return ($y2/parent::b/@attribute:id, $y2:child::b);
fcn2()
```

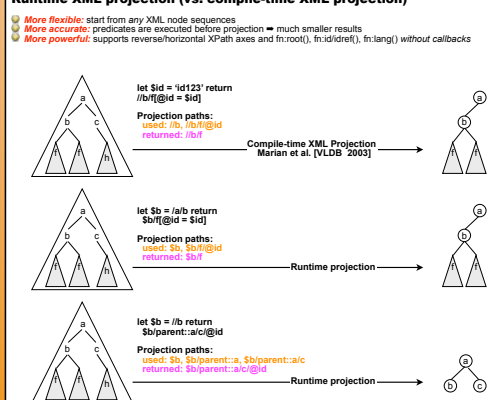
### Applying Distributed Code Motion in Q<sup>c</sup>

```
declare function fcn2($param1 as xs:string, $param2 as xs:string) as node()* {
  for $y in doc('xpc:/p2/b.xml')/child::b:child::y return
  if ($param1 = $y/@attribute:id and $param2 = $y/@attribute:tpe) then $y else ();
}
declare function fcn1() as node()* {
  doc('xpc:/p1/a.xml')/child::a:child::x;
}
let $y2 := for $x in execute at ["p1"] (fcn1()) return execute at ["p2"] (fcn2($x/@attribute:id, $x/@attribute:tpe))
return ($y2/parent::b/@attribute:id, $y2:child::b);
fcn2()
```

## Decomposition Graph of Q<sup>c</sup> with projection paths



## Runtime XML projection (vs. compile-time XML projection)



## Future work

- Placement of decomposed subqueries
- N sub-queries, M peers => N^M possibilities!
- Decomposition of updating queries