PQL: A Purely-Declarative Java Extension for Parallel Programming

Peter Pilgerstorfer

Reichenbach, C., Smaragdakis, Y., & Immerman, N. (2012). PQL: a purely-declarative java extension for parallel programming. In *ECOOP 2012–Object-Oriented Programming* (pp. 53-78). Springer Berlin Heidelberg.

PQL

- Declarative query language for parallel programs
  - First order logic
- Java Extension
- Design goals
  - Easy to write parallel code
  - Focus on performance
- Not Turing complete

Overview

- Queries
- Implementation
- Evaluation

Quantifying Queries

- Boolean expression
- Syntax
  $\langle QUANT \rangle \langle ID \rangle : \langle QUERY \rangle$
- Examples

```
forall int x : x == x
exists int x : myArray[x] == x
```

Container Queries

- Create a Map, Set or Array
- Syntax
  query $(\langle MATCH \rangle) : \langle QUERY \rangle$
- Examples

```
query(Set.contains(x)) : range(1,10).contains(x)
query(Map.get(x) == y) :
   range(1,10).contains(x) && y == 2*x
```

Reduction Queries

- Compute reduction
- Syntax
  reduce(⟨OP⟩) ⟨ID⟩ [**over** ⟨IDs⟩] : ⟨$QUERY$⟩
- Examples

```
reduce(sumInt) int x : mySet.contains(x)
reduce(sumInt) int x over y : myMap.get(y) == x
```

- (restricted) user defined operations possible

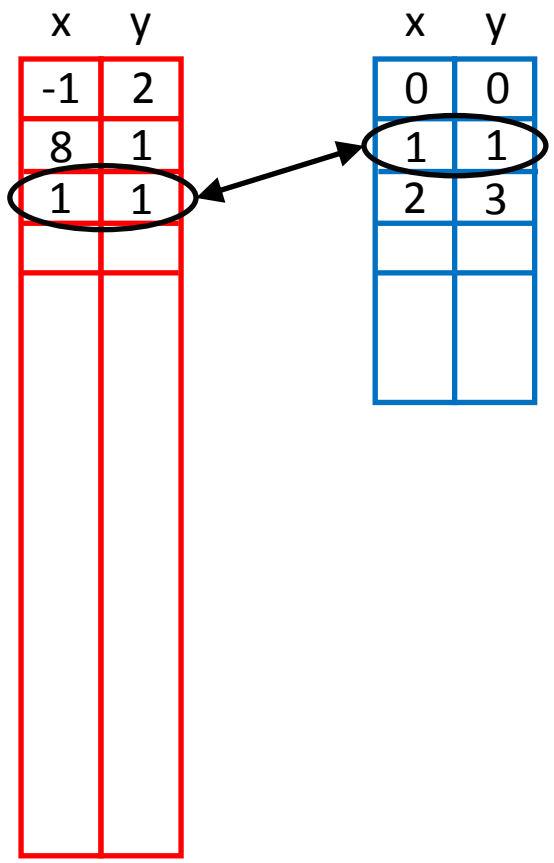
Implementation

- Translate PQL into intermediate language PQIL
  - Only reduce operations and joined expressions
  - Java expressions are seen as constants
- Translate PQIL to nested loops before compilation

Implementation - Access Path Selection

```
query(Set.contains(y)):
    func(x) == y && arr[x] == y
```

- Relation size
- Parallelizability
  - In practice outer loop parallel
  - Loop has to be large enough



Optimization

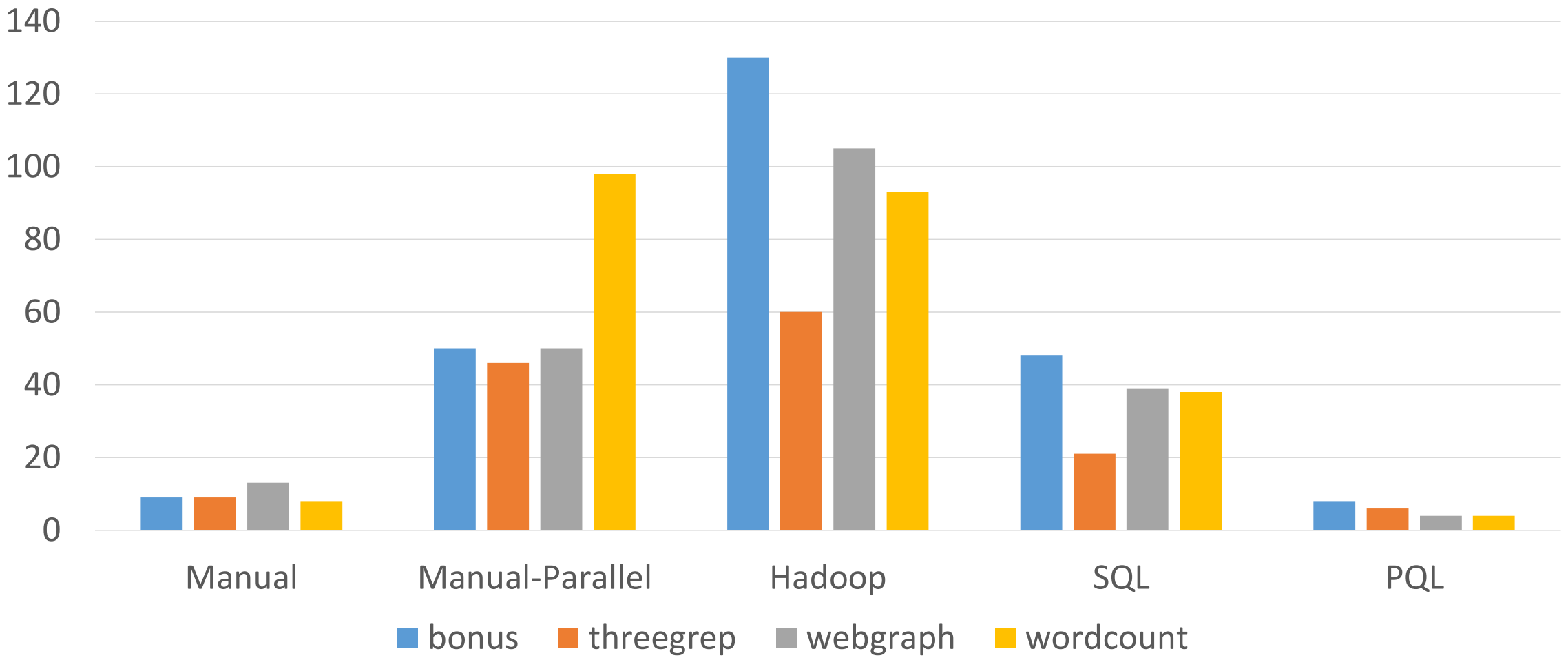
- Elimiate redundant joins
- Merge nested queries
  - Original:

```
query(Map.get(key) == newset):
        newset == query(Set.contains(value)): array[value] == key;
```

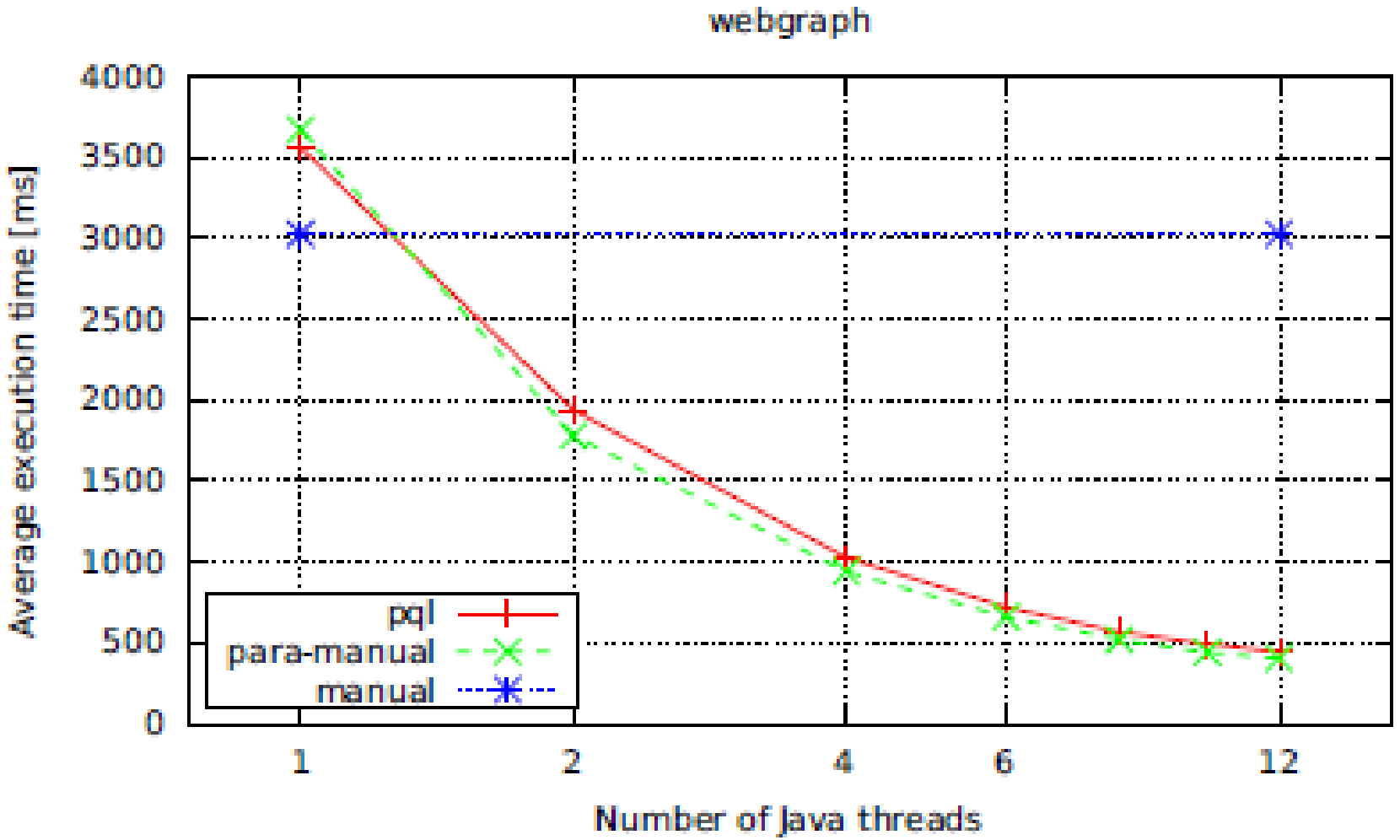
  - Merged:

```
query(Map.get(key) == Set.contains(value)):
        array[value] == key;
```

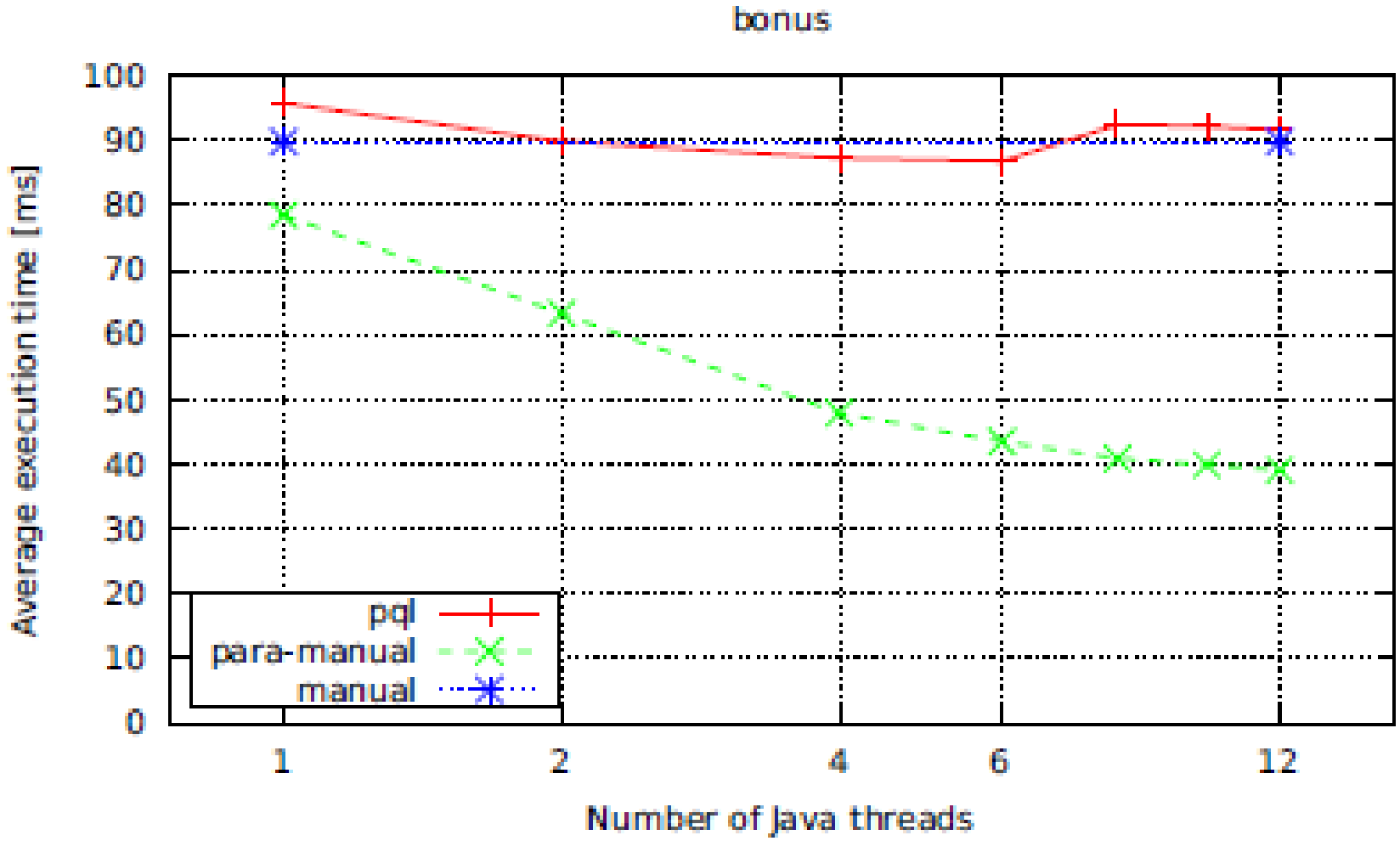
Evaluation – Lines of Java code



Evaluation - Speed



Evaluation - Speed



Discussion

- Declarative language
- Works with Java data structures
- Little computational overhead compared to manual parallelization