Parallel Twig Joins - Multicore XML Query processing
Lila Shnaiderman (presenting), Oded Shmueli

**1. XML Document representation**

**Stream representation:**
- Document D: <Movie> "Harry Potter" <Name> Daniel Radcliffe <First> Rupert Grint <Last> Daniel Radcliffe <Actor>
- Query Q: <Actor>

**Query Twig Pattern:**
- Given query Q and document D, compute all matches to Q in D.

**2. TwigStack (base alg.)**

**Stacks Role**
- Matches: N_1M_1P_1, N_2M_2P_2, N_3M_3P_3, N_4M_4P_4
- Data: Query N, M, P

**Example for query Q and document D:**
- Next node: 16:25

**Main idea:**
- Separate the work on the query between many threads, using data partitioning approach

**First step:**
- Choose partition from the PA.
  - "large enough" subtrees.
  - number of subtrees much larger than number of threads.

**Second step:**
- Activate thread.
  - Each thread:
    - runs until there are tasks in the pool.
    - Takes numOfTasks tasks from the pool.
    - Finds the limits of sub-streams for each stream in query streams set.
      - (by using Find-Sub-Stream method)
    - Performs almost original TwigStack alg.
      - (changes in eof, getNext, etc)

**3. Parallel TwigStack**

**Find-Sub-Stream Method:**
- Logical union
- ft-root
- ft-root's sub-streams

**Find start index of ss within s:**
- Binary search for node n with smallest leftPos.
- n.rightPos > rt1.leftPos AND n.leftPos < rtx.leftPos

**Example for query Q and document D:**
- Step 1: Choose partition from PA: (2,5,2), (6,15,2), (16,25,2).
  - Processing subtrees:
    - First thread: (2,5,2) subtree.
      - Nothing to do.
    - Second thread: (6,15,2) subtree.
      - Nothing to do.
    - Second thread: (16,25,2) subtree.

**4. Experimental results**

- Used XMark to create different XML docs.
- Used different path and twig patterns.
- Main metric of performance: run time.
- Experiment input files:
  - 1. XML document.
  - 2. Text file with queries.
- Experiment steps:
  - Load doc D into XML Streams storage.
  - Build PA.
  - Run the queries on D with different number of threads.

**Goal:**
- Speed XML documents querying by speeding a single query.

**Algorithm outline:**
- First step:
  - Choose partition from PA.
    - "large enough" subtrees.
    - number of subtrees much larger than number of threads.
  - Second step:
    - Activate thread.
    - Each thread:
      - runs until there are tasks in the pool.
      - Takes numOfTasks tasks from the pool.
      - Finds the limits of sub-streams for each stream in query streams set.
        - (by using Find-Sub-Stream method)
      - Performs almost original TwigStack alg.
        - (changes in eof, getNext, etc)