Tool Transformation Contest 2018
29 June 2018, Toulouse, France

YAMTL Solutions

Artur Boronat

UNIVERSITY OF LEICESTER

YAMTL

- Declarative M2M trafos in Xtend
  - Inspired in ATL semantics
  - Does not separate initialization and binding phases
  - Resolution strategy
  - Rules with multiple inheritance
  - Module composition
- Execution modes
  - Batch
  - **Incremental for rule applications ☺**
    - Dependency tracking
    - EMF change model for representing deltas
    - Forward propagation
  - But…**no incremental queries ☹**
  - And…**no parallelism ☹**

Abstract Solution Approach

- Map/Reduce metaphor
  - Map:
    - use transformation rules for computing scores
    - store scores per element
  - Reduce: use Xtend's collection extensions to sort elements (posts for Q1/comments for Q2)
- Optimized version
  - Map:
    - use transformation rules for computing scores
    - keep only three best options (sorting for each insert)
  - Reduce: print result
  - Usually more efficient when many rule applications (e.g. comments)

Solutions

- Solution A
  - Q1: Map/Reduce
  - Q2: Optimized Map/Reduce
- Solution B
  - Q1: Optimized Map/Reduce
  - Q2: Optimized Map/Reduce

Solution A Q1 (MR): map

```
class Q1_yamtl_batch extends YAMTLModule {
    @Accessors
    val public Map<Post,Integer> postToScore = newHashMap

    val SN = SocialNetworkPackage.eINSTANCE

    new () {
        header().in('sn', SN).out('out', SN)
        ruleStore( newArrayList(
            new Rule('CountPosts')
                .in('post', SN.post).build()
                .out('postAux', SN.post, [
                    val post = 'post'.fetch as Post
                    val commentList = EcoreUtil2.getAllContentsOfType(post, Comment)
                    var int score = 0
                    if (commentList.size > 0)
                        score = commentList.map[c | 10 + c.likedBy.size].sum
                    postToScore.put(post,score)
                ]).build()
                .build()
        ))
    }

    // HELPERS
    def private sum(List<Integer> list) {
        list.reduce[v1, v2 | v1+v2]
    }
}
```

Compute score

Solution A Q1 (MR): reduce

```
public class SolutionQ1 extends Solution {

    new() {
        xform = new Q1_yamtl_batch
        xform.stageUpperBound = 1
        xform.extentTypeModifier = ExtentTypeModifier.LIST
        xform.fromRoots = false
        xform.executionMode = ExecutionMode.INCREMENTAL
    }

    override String Initial() {
        xform.execute()
        (xform as Q1_yamtl_batch).postToScore.selectThree
    }

    override String Update(String deltaName) {
        xform.propagateDelta('sn', deltaName)
        (xform as Q1_yamtl_batch).postToScore.selectThree
    }

    def private selectThree(Map<Post,Integer> postToScore) {
        postToScore.entrySet
            .sortWith([p1,p2|
                val result = Integer::compare(p1.value, p2.value) * -1
                if ( result == 0)
                    p1.key.timestamp.compareTo(p2.key.timestamp) * -1
                else
                    result
            ]).take(3)
            .map[it.key.id].join('|')
    }

}
```

configuration

batch invocation

forward
delta propagation
invocation

“reduce”

Solution B Q1 (OMR): map

```
class Q1_yamtl_batch_v2_threeBest extends YAMTLModule {
    @Accessors
    val Map<Post,Integer> controversialPosts = newHashMap

    val SN = SocialNetworkPackage.eINSTANCE

    new () {
        header().in('sn', SN).out('out', SN)

        ruleStore( newArrayList(
            new Rule('CountPosts')
                .in('post', SN.post).build()
                .out('postAux', SN.post, [
                    val post = 'post'.fetch as Post
                    val commentList = EcoreUtil2.getAllContentsOfType(post, Comment)
                    var int score = 0
                    if (commentList.size > 0)
                        score = commentList.map[c | 10 + c.likedBy.size].sum
                    controversialPosts.put(post, score)
                    controversialPosts.trimToBestThree
                ]).build()
                .build()
        ))
    }

    // HELPERS
    def private sum(List<Integer> list) {
        list.reduce[v1, v2 | v1+v2]
    }

    def public static trimToBestThree(Map<Post,Integer> map) {
        val list = map.entrySet.sortWith([c1,c2|
            val result = - Integer::compare(c1.value, c2.value)
            if ( result == 0) {
                - c1.key.timestamp.compareTo(c2.key.timestamp)
            } else
                result
        ])
        if(list.size>3) map.remove(list.last.key)
        list
    }
}
```

Optimization

Solution B Q1 (OMR): reduce

```
public class SolutionQ1 extends Solution {

    new() {
        xform = new Q1_yamtl_batch_v2_threeBest
        xform.stageUpperBound = 1
        xform.extentTypeModifier = ExtentTypeModifier.LIST
        xform.fromRoots = false
        xform.executionMode = ExecutionMode.INCREMENTAL

    }

    override String Initial() {
        xform.execute()
        val list = Q1_yamtl_batch_v2_threeBest
            .trimToBestThree((xform as Q1_yamtl_batch_v2_threeBest).controversialPosts)
        list.map[it.key.id].join('|')
    }

    override String Update(String deltaName) {
        xform.propagateDelta('sn', deltaName)
        val list = Q1_yamtl_batch_v2_threeBest
            .trimToBestThree((xform as Q1_yamtl_batch_v2_threeBest).controversialPosts)
        list.map[it.key.id].join('|')
    }
}
```

Solution Q2 (OMR): graph components

```
public class FriendsComponents extends GraphComponents {

    @Accessors
    var private List<User> userList;

    new(List<User> list) {
        super(list.size)
        userList = list
    }

    def static public computeComponents(List<User> list) {
        val FriendsComponents fc = new FriendsComponents(list)

        for (var i=0; i<list.size; i++) {
            if (i+1<list.size) {
                for (var j=i+1; j<list.size; j++) {
                    if (fc.connected(i,j)) {
                        fc.union(i,j)
                    }
                }
            }
        }
        fc
    }

    def public getSquaredComponentSizes() {
        this.parent.groupBy[it].values.map[size * size]
    }

    override public boolean connected(int p, int q) {
        userList.get(p).friends.contains(userList.get(q))
        ||
        userList.get(q).friends.contains(userList.get(p))
    }
}
```

Solution Q2 (OMR): map

```
class Q2_yamtl_batch extends YAMTLModule {
    @Accessors
    val Map<Comment,Integer> influentialComments = newHashMap

    val SN = SocialNetworkPackage.eINSTANCE

    new () {
        header().in('sn', SN).out('out', SN)

        ruleStore( newArrayList(
            new Rule('UserComponentsByComment')
                .in('comment', SN.comment).build()
                .out('commentAux', SN.comment, [
                    val comment = 'comment'.fetch as Comment
                    var score = 0
                    if (comment.likedBy.size > 0) {
                        val fc = FriendsComponents.computeComponents(comment.likedBy)
                        score = fc.squaredComponentSizes.sum
                    }
                    influentialComments.put(comment, score)
                    influentialComments.trimToBestThree
                ]).build()
                .build()
        ))
    }

    // HELPERS
    def public static trimToBestThree(Map<Comment,Integer> map) {
        val list = map.entrySet.sortWith([c1,c2|
            val result = - Integer::compare(c1.value, c2.value)
            if ( result == 0) {
                - c1.key.timestamp.compareTo(c2.key.timestamp)
            } else
                result
        ])
        if(list.size>3) map.remove(list.last.key)
        list
    }

    def private sum(Iterable<Integer> list) {
        list.reduce[v1, v2 | v1+v2]
    }
}
```

Solution Q2 (OMR): reduce

configuration

batch

forward
delta propagation

```
public class SolutionQ2 extends Solution {
    new() {
        xform = new Q2_yamtl_batch
        xform.stageUpperBound = 1
        xform.extentTypeModifier = ExtentTypeModifier.LIST
        xform.fromRoots = false
        xform.executionMode = ExecutionMode.INCREMENTAL
    }

    override String Initial() {
        xform.execute()
        val list = Q2_yamtl_batch.trimToBestThree(
            (xform as Q2_yamtl_batch).influentialComments
        )
        list.map[it.key.id].join('|')
    }

    override String Update(String deltaName) {
        xform.propagateDelta('sn', deltaName)
        val list = Q2_yamtl_batch.trimToBestThree(
            (xform as Q2_yamtl_batch).influentialComments
        )
        list.map[it.key.id].join('|')
    }
}
```

Adaptation of Benchmark Framework

- Change representation has been converted to EMF representation
- LiveContestDriver:
  - Converted to Xtend for convenience
  - Load initial models and deltas

```
def private static Object loadFile(String path) {
    val modelPath = '''«ChangePath»/«path»'''
    println("model path: " + modelPath)
    solution.xform.loadInputModels(#{'sn' -> modelPath})
    println("loaded")
    val mRes = solution.xform.getModelResource('sn')
    return mRes.getContents().get(0);
}

def static void Load()
{
    stopwatch = System.nanoTime();
    solution.setSocialNetwork(loadFile("initial.xmi") as SocialNetworkRoot);

    for (var iteration = 1; iteration <= Sequences; iteration++)
    {
        val deltaName = '''change«iteration»'''
        val deltaPath = '''«ChangePath»/change«iteration».documented.xmi'''
        solution.xform.loadDelta('sn', deltaName, deltaPath, new Timestamp(System.nanoTime()))
    }

    stopwatch = System.nanoTime() - stopwatch;
    Report(BenchmarkPhase.Load, -1, null);
}
```

Conclusions

- Correctness
  - Q1: correct ☺
  - Q2: correct for change sets: 1, 8 ☺
    - Incorrect for the rest ☹ but only for one out of three ☺
- Usability
  - Xtend ☺
  - Java integration ☺
  - Several files (map/reduce/aux) 😐

Conclusions

- Performance
  - Run up to size 512
  - A Q1 (512, update time):
    - Initial: 309.62 ms
      - YAMTL_Solution_A;Q1;512;0;0;Initialization;Time;309622336
    - Update 20: 217 ms ☹
      - YAMTL_Solution_A;Q1;512;0;20;Update;Time;217959330
  - B Q1 (512, update time):
    - Initial: 2438.2108 ms
      - YAMTL_Solution_B;Q1;512;4;0;Initial;Time;2438210874
    - Update 20: 9.7339 ms ☺
      - YAMTL_Solution_B;Q1;512;4;20;Update;Time;9733973
  - Q2 (512, update time):
    - Initial: 7011.1708 ms
      - YAMTL_Solution_B;Q2;512;4;0;Initial;Time;7011170879
    - Update 20: 10.18 ms
      - YAMTL_Solution_B;Q2;512;4;20;Update;Time;10183543