Oolong: Programming Asynchronous Distributed Applications with Triggers

Christopher Mitchell ◆ Russell Power ◆ Jinyang Li
{cmitchell,power,jinyang}@cs.nyu.edu

Introduction

◆**MapReduce, Dryad, Piccolo:** Good for…
- •Batch computation
- •Synchronous iteration with global barriers
- •Examples: PageRank, K-Means, Matrix Mult

◆**MapReduce, Dryad, Piccolo:** Bad for…
- •Sparse execution: not all data needs equal analysis, previous results determine data for future processing
- •Asynchronous tasks: no global barrier needed
- •Convergence: inherent detection of termination without separate check job

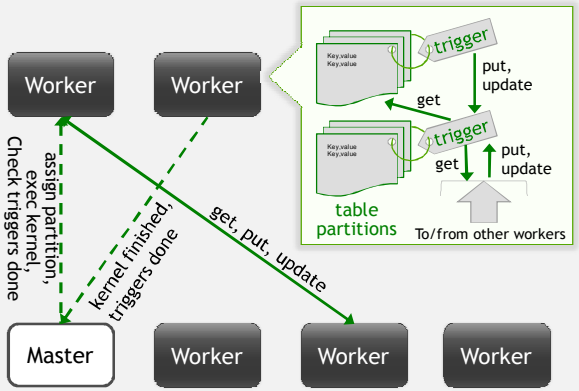
◆**Oolong** targets:
- •Asynchronous execution without global barriers
- •Incremental recomputation
- •Examples: Crawling, incremental PageRank, SSSP

Key Innovations

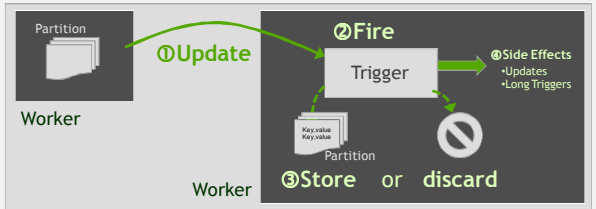
•Key Insight: Synchronous execution wastes time for many problems. Triggers save time with fast asynchronous execution

1. Flexible asynchronous distributed execution
2. Database-style triggers as first-class citizens
3. Triggers in a lightweight shared key-value store
4. Fine-grained checkpointing and failure recovery

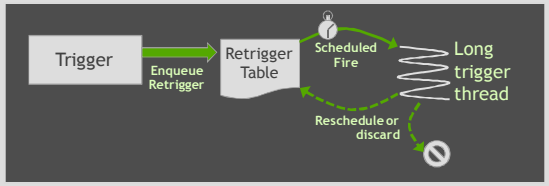
Oolong Architecture



What is a Trigger?



Normal/Short Triggers



Long Trigger

Normal or **short triggers:**
- •Accept, reject, or modify updates
- •Fast, small sections of code
- •Minimize delay before updates visible in tables

Long triggers:
- •Perform longer, complex tasks
- •Periodic tasks
- •Tasks waiting for prerequisites

Sample Trigger

```
def SSSP_Kernel():
    dists.update(0,0) //initialize source node
def SSSP_Trigger(node_ID, old_dist, new_dist):
    if new_dist < old_dist:
        for target in nodes(node_ID).targets:
            dists.update(target,1+new_dist)
        accept update (new_dist->old_dist)
    else reject update
def SSSP(Config conf):
    dists = Table(int, double)
    nodes = Table(int, Node)
    initialize all dists <- infinity
    enable SSSP_Trigger on dists table
    RunOne(SSSP_Kernel) #initializes source
```

Challenges

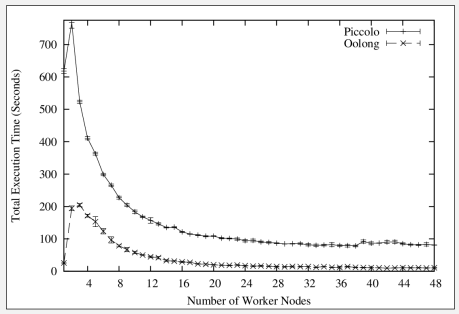
◆Balancing trigger responsiveness with flexibility and power
- •Solution: long and short triggers
- •Short triggers provide responsiveness
- •Long triggers handle powerful or blocking code

◆Designing a trigger-friendly fault tolerance mechanism
- •Global checkpointing is too slow
- •Economical replication-based checkpointing

◆High-performance distributed execution and storage
- •Build on successes of Piccolo
- •Simplicity of a key-value table

◆Data-execution locality with triggers
- •Solved with Piccolo's shared partitioned tables

Performance



◄Scalability:
- •SSSP workload
- •Scales well from 2 to 11 workers
- •Outperforms Piccolo by 2-4x

Asynchronous Performance ►
- •PageRank workload
- •10x-500x faster to re-quiesce augmented web graph