ELF

Efficient Lightweight Fast Stream Processing at Scale*

Liting Hu  Karsten Schwarshrishikesh Amur
Xin Chen

Georgia Institute of Technology

*Funded in part with support from the Intel Cloud ISTC
Outline

Motivation

ELF Design

Evaluation
Applications

- Micro-promotions
- Product-bundling
- Sales-prediction
  ... ... ...
- Coupon
- Recommendations
- Predictions
Concurrent, diverse jobs with varied delays

- Filter @item
- Sort #clicks
- Filter @buys
- Batch model
- Updates every half hour
- Stream model
- Updates per min
- Updates every x milliseconds
- Query Angry Birds5?
- Yes/No?
- User query
System Needs

Flexible
- Diverse jobs
- Cross-job coordination
- Runtime job change

Scalable
- Hundreds of concurrent jobs with shared inputs

Performance
- High throughput
- Low latency
Existing Solutions

Log collection systems
- e.g., Flume, Kafka

Web server

Storage

N jobs
Outline

Motivation

ELF Design

Evaluation
ELF runs directly on the web server tier.
hash('app3') = 2080c3
hash('app1') = d502fb
hash('app2') = d462ba

Each organized into a peer-to-peer overlay

Master

Many Masters/Many Workers
CBT-based agent `pre-reduces’ local key-value pairs

SRT globally `shuffles’ and `reduces’ key-value pairs

ELF APIs for programmers to implement diverse jobs
Example log event
{"created_at":"23:48:22 +0000 2013", "id":299665941824950273, "product":"Angry Birds Season", "clicks_count":2, "buys_count":0, "user":{"id":343284040, "name":"@Curry", "location":"Ohio", ...} ...

ELF QL ->
SELECT product,SUM(clicks_count) FROM *
WHERE store == `video_games`
GROUP BY product
SORT BY SUM(clicks_count) DESC
LIMIT 10
WINDOWING 30 SECONDS;
Compressed Buffer Tree (CBT)

in-memory data structure

<(a,1)(b,1)(a,1)(b,1)(b,1)>

(1) ‘insert’ to fill
(2) ‘flush’ to fetch the pre-reduced result
(3) ‘empty’ to empty the states

<(a,2)(b,3)>
CBT-based agent `pre-reduces’ local key-value pairs
Stateful, asynchronous, and synchronous execution

SRT globally `shuffles’ and `reduces’ key-value pairs

ELF APIs for programmers to implement diverse jobs
CBT-based agent `pre-reduces’ local key-value pairs
Stateful, asynchronous, and synchronous execution
SRT globally `shuffles’ and `reduces’ key-value pairs

ELF APIs for programmers to implement diverse jobs
Per-job dataflow

hash(`micro-promotion’)  = d462ba

hash(`product-bundling’) = 2080c3
hash(\`micro-promotion\`) = d462ba

hash(\`product-bundling\`) = 2080c3

Synchronizing CBT to be emptied
Last iteration’s result
New job function or parameter
Cross-job Coordination

hash(`micro-promotion`)  
= d462ba

hash(`product-bundling`)  
= 2080c3

hash(`Sale-predication`)  
= 43bc3c

Limited to $O(\log(N))$ hops
CBT-based agent `pre-reduces’ local key-value pairs
Stateful, asynchronous, and synchronous execution
SRT globally `shuffles’ and `reduces’ key-value pairs
Per-job dataflow, and cross-job coordination
ELF APIs for programmers to implement diverse jobs
CBT-based agent `pre-reduces’ local key-value pairs
Stateful, asynchronous, and synchronous execution

SRT globally `shuffles’ and `reduces’ key-value pairs
Per-job dataflow, and cross-job coordination

ELF APIs for programmers to implement diverse jobs
ArrayList<String> topk;
void OnTimer () {
    if (this.isRoot()) {
        this.Multicast(hash("micro-promotion"), new topk(topk));
        this.Multicast(hash("micro-promotion"), new update());
    }
}
void OnMulticast(Id appid, Message message) {
    if (message instanceof topk) {
        for(String product: message.topk) {
            if(this.hasProduct(product))
                //if it is an topk message, appear discount ... }
    }

    //if it is an update message, start a new batch
    else if (message instanceof update) {
        //if leaves, flush CBT and update to the parent vertex
        if (!this.containsChild(appid)) {
            PA0 paos = cbt.get(appid).flush();
            this.SendTo (this.getParent(appid), paos); 
            cbt.get(appid).empty();
        }
    }
}
Outline

Motivation

ELF design

Evaluation
Latency

Replay Twitter’s stream
1280 agents
50 events/second

60 × 12-core 2.66 GHz AMD Opteron
48 GB RAM per server
Gigabit Ethernet

✓ ELF outperforms for large windows due to the efficient CBT flush
New Functionality

- 10s millisecond for multicast messages
- 100s millisecond to update functions
Scalability

ELF scales well up to thousands of concurrent jobs

- Deploying 500 SRTs
- Deploying 1000 SRTs
- Deploying 2000 SRTs

60 × 12-core 2.66 GHz AMD Opteron
48 GB RAM per server
Gigabit Ethernet
1000 agents
Conclusions and Future Work

ELF introduces a `many masters/many workers’ framework:

• To achieve **good performance**: low latency and high throughput
• **scalability**: for diverse, concurrent jobs, and
• **new functionality**: job flexibility feedback and coordination.

Future work: understand the limitations of Elf; Explore more complex web-tier processing; look at task isolation.
Thank you!

Liting Hu

foxting@gatech.edu