TOSS: A Topology-based Scheduler for Storm Clusters

Yi Zhou, Yangyang Liu, Chaowei Zhang, Xiaopu Peng, Xiao Qin

Agenda

- MOTIVATION
- SYSTEM DESIGN & IMPLEMENTATION
- EVALUATION AND EXPERIMENTAL RESULTS
- CONTACT INFORMATION

MOTIVATION

- Inter-node traffic is likely to become a performance bottleneck.
- Scheduling executors represented in form of topologies play a vital role in optimizing performance of storm clusters.
- Reducing rescheduling overhead in the run time.

SYSTEM DESIGN

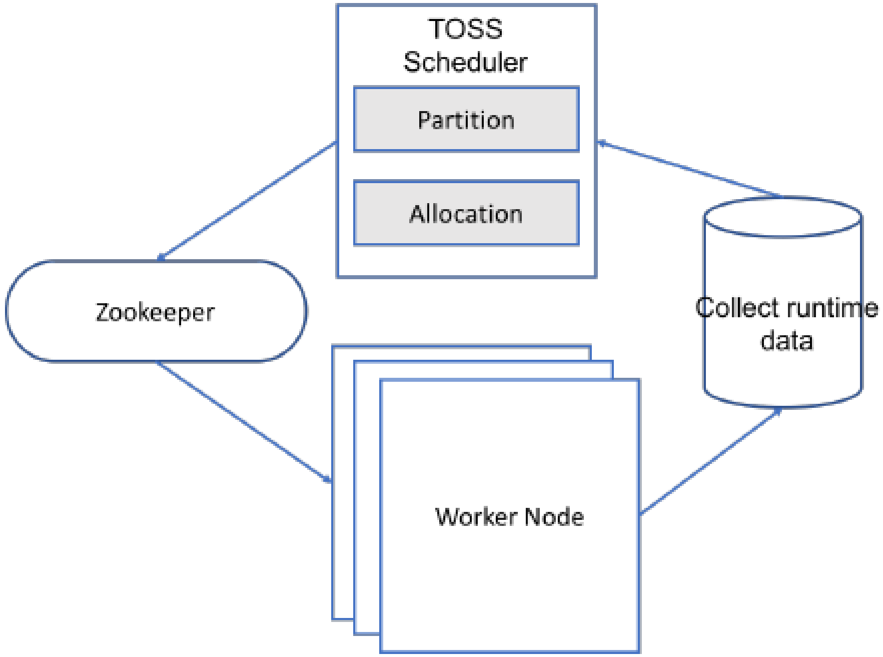


Fig. 6: The System Design of the TOSS scheduler.

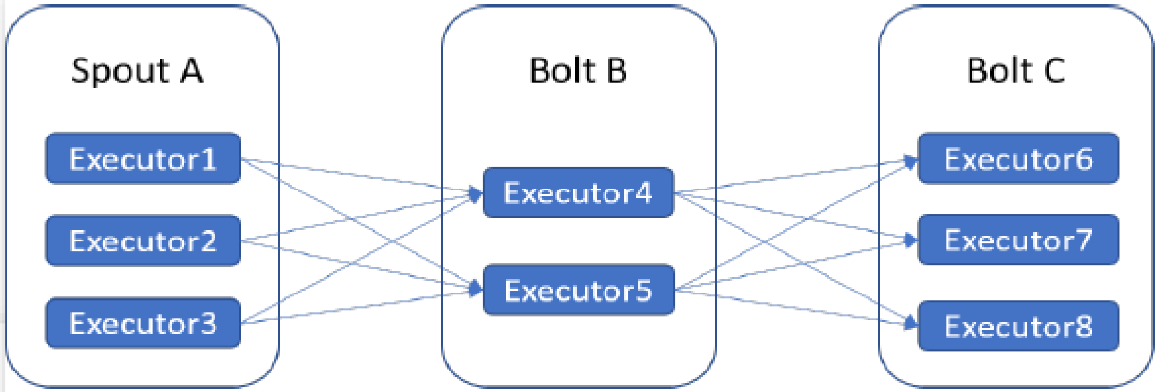


Fig. 4: Communications among multiple executors before executor allocation.

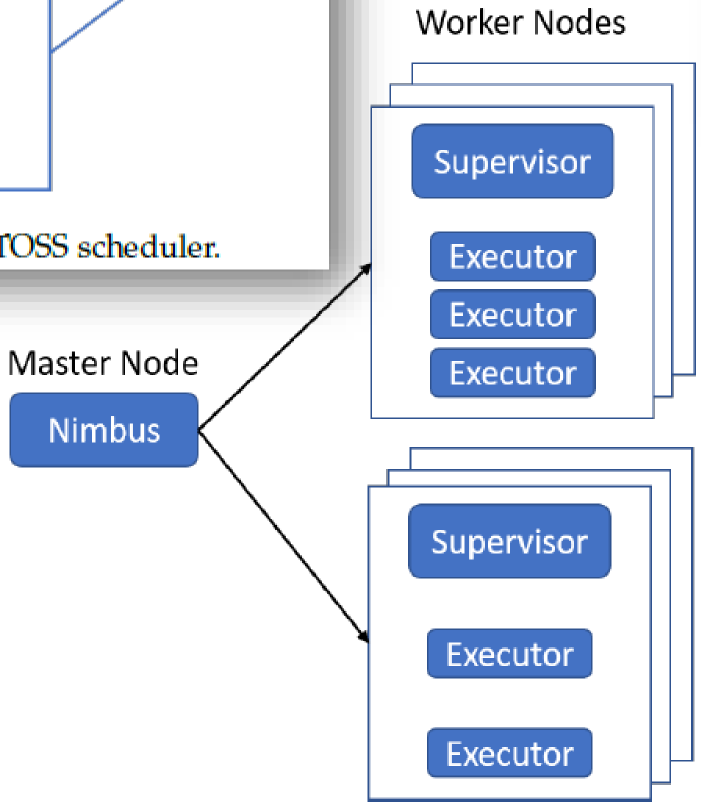


Fig. 2: A master node and worker nodes in Storm.

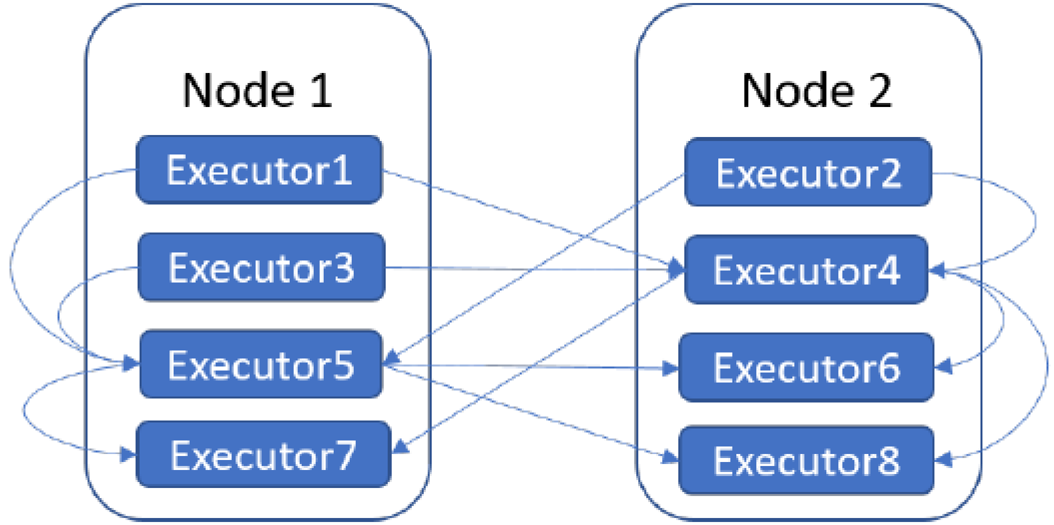
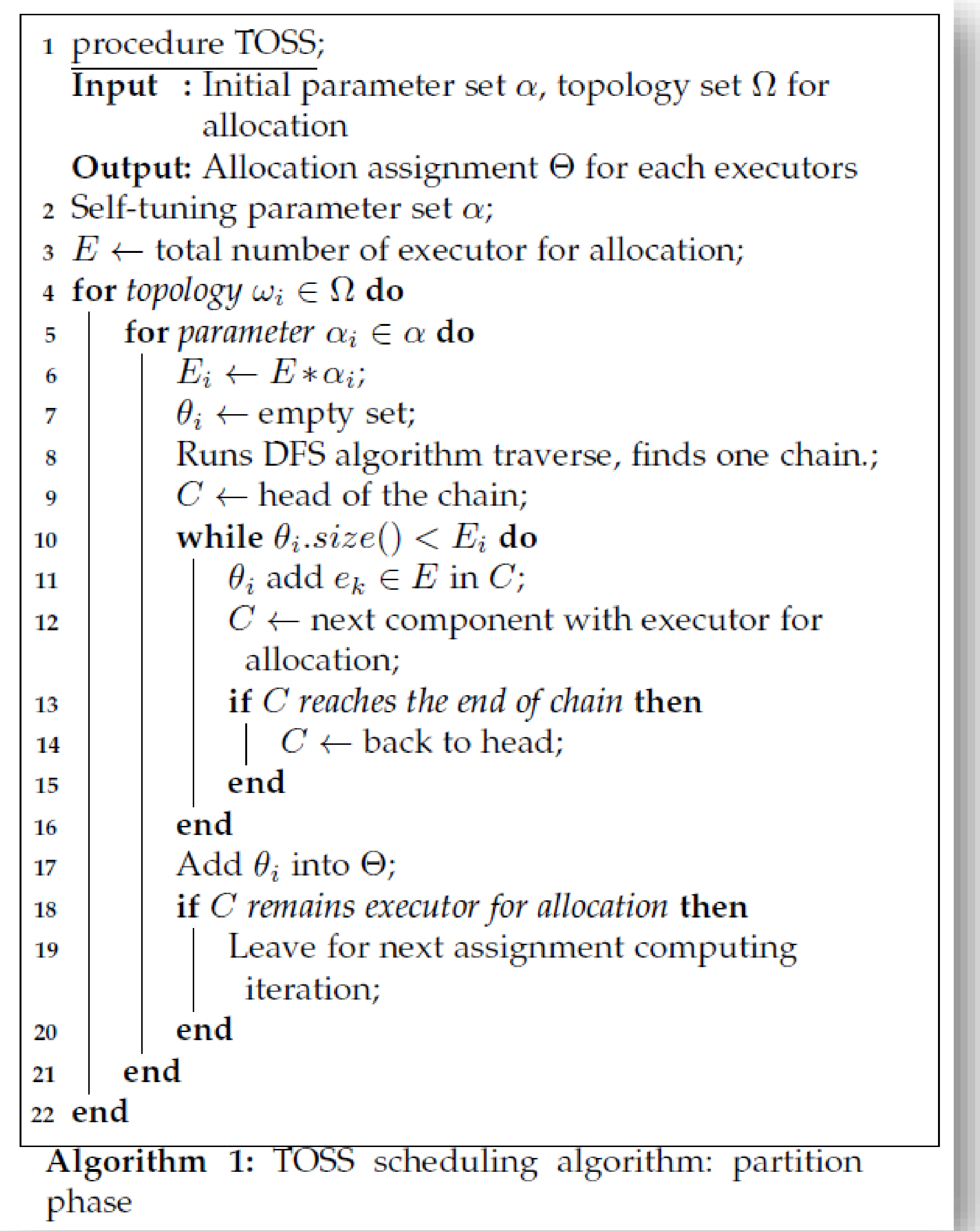


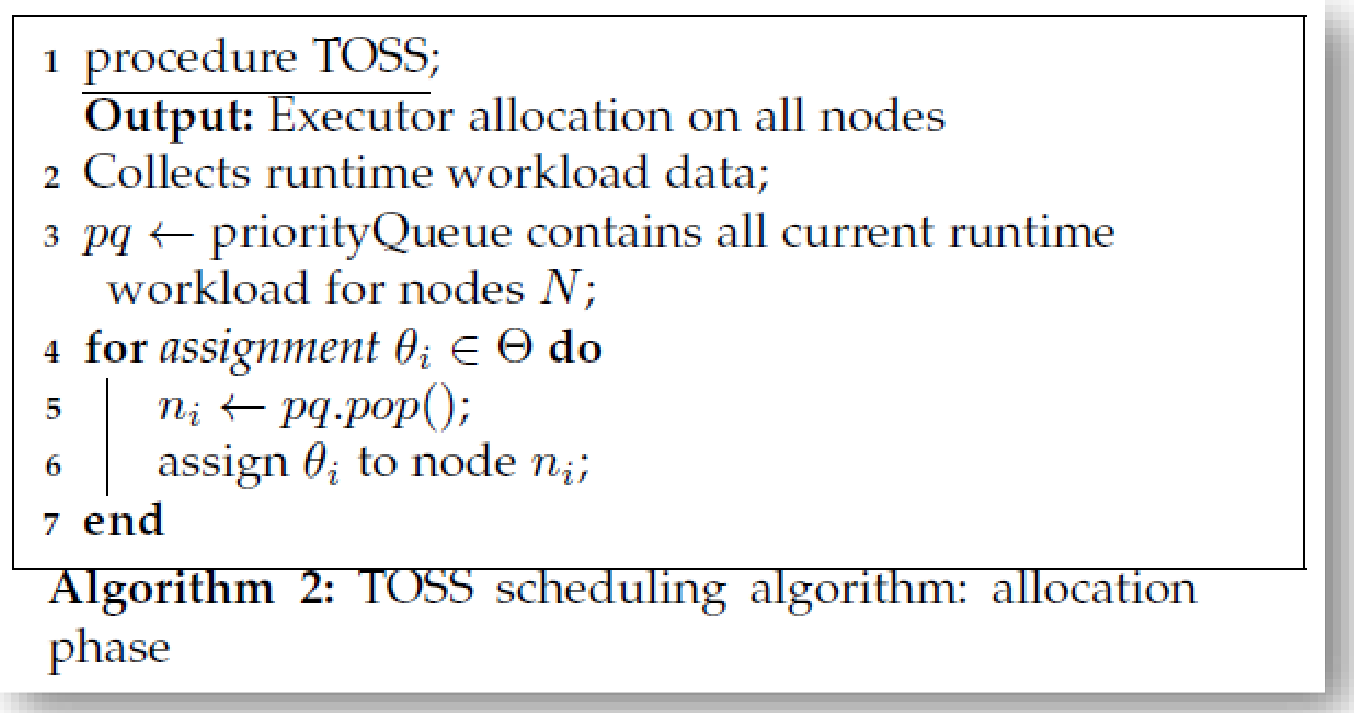
Fig. 5: Communications among multiple executors after executor allocation.

SYSTEM DESIGN Cont. – Algorithms Design



```
procedure TOSS;
Input  : Initial parameter set $\alpha$, topology set $\Omega$ for
         allocation
Output: Allocation assignment $\Theta$ for each executors
Self-tuning parameter set $\alpha$;
$E \leftarrow$ total number of executor for allocation;
for topology $\omega_i \in \Omega$ do
    for parameter $\alpha_i \in \alpha$ do
        $E_i \leftarrow E * \alpha_i$;
        $\theta_i \leftarrow$ empty set;
        Runs DFS algorithm traverse, finds one chain.;
        $C \leftarrow$ head of the chain;
        while $\theta_i.size() < E_i$ do
            $\theta_i$ add $e_k \in E$ in $C$;
            $C \leftarrow$ next component with executor for
              allocation;
            if $C$ reaches the end of chain then
                $C \leftarrow$ back to head;
            end
        end
        Add $\theta_i$ into $\Theta$;
        if $C$ remains executor for allocation then
            Leave for next assignment computing
              iteration;
        end
    end
end
```

Algorithm 1: TOSS scheduling algorithm: partition phase

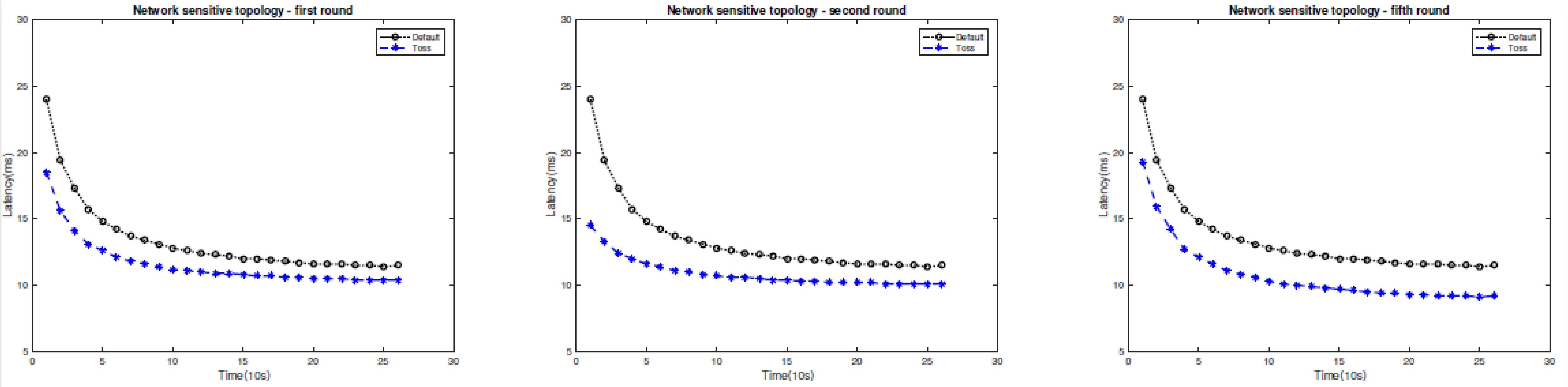


```
procedure TOSS;
Output: Executor allocation on all nodes
Collects runtime workload data;
$pq \leftarrow$ priorityQueue contains all current runtime
  workload for nodes $N$;
for assignment $\theta_i \in \Theta$ do
    $n_i \leftarrow pq.pop()$;
    assign $\theta_i$ to node $n_i$;
end
```

Algorithm 2: TOSS scheduling algorithm: allocation phase

EVALUATION AND EXPERIMENTAL RESULTS

- The latency comparison between multiple submission of TOSS and default scheduler



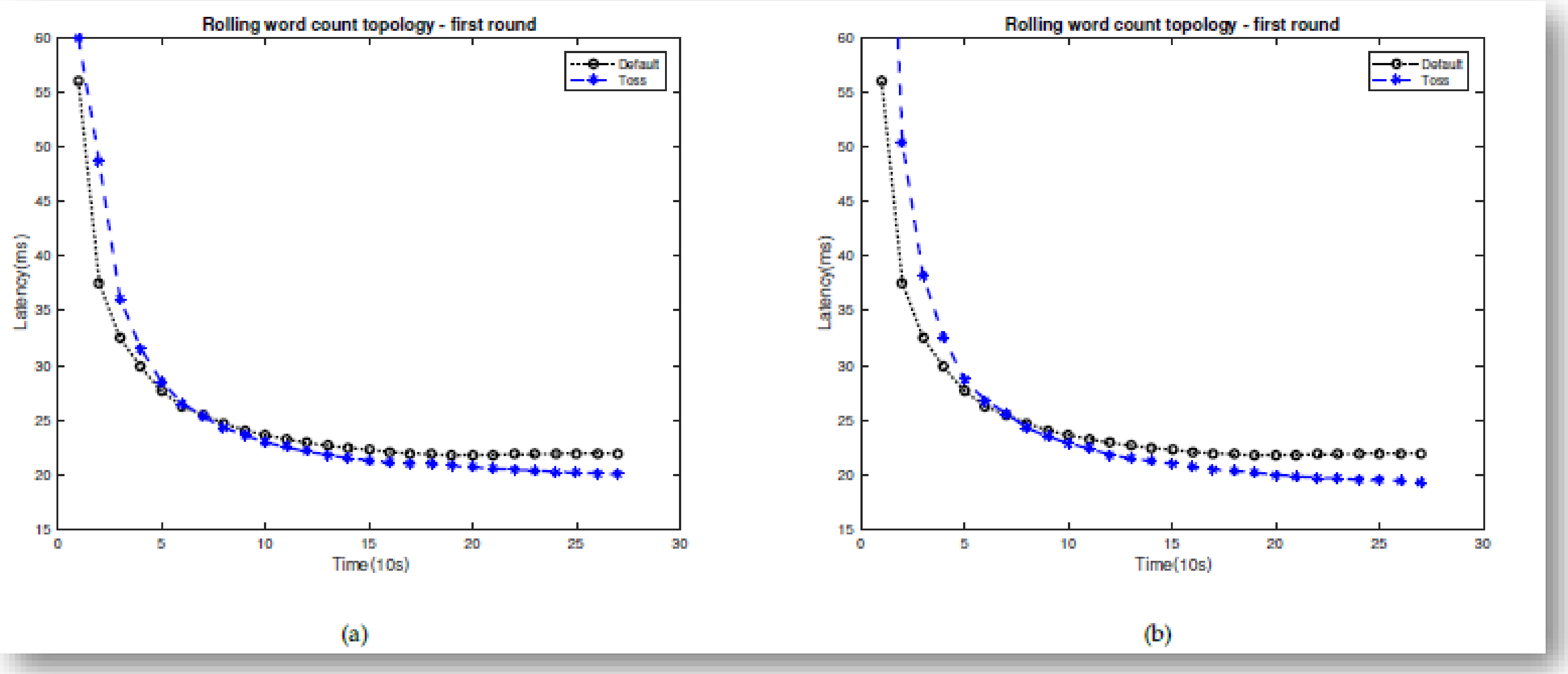
(a)

(b)

(c)

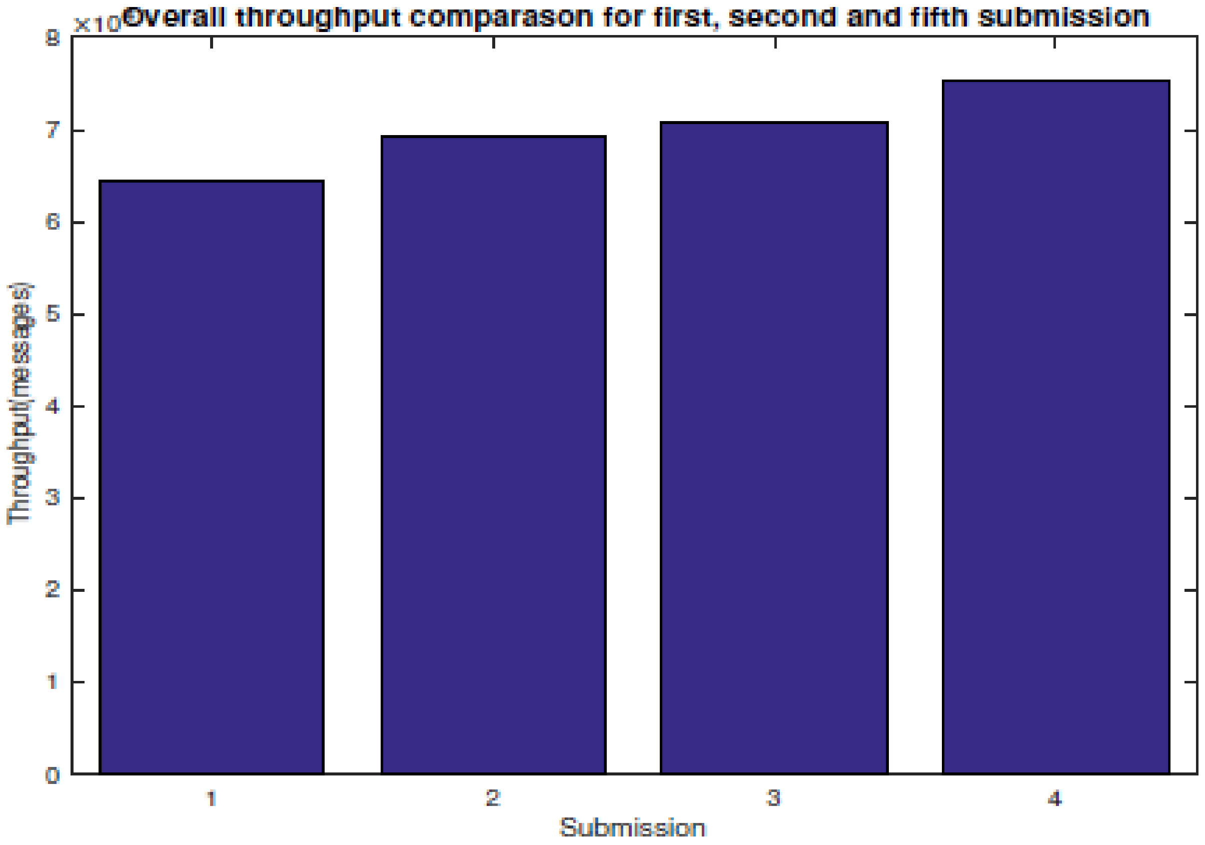
EVALUATION AND EXPERIMENTAL RESULTS

- The latency comparison using the Wordcount benchmark.

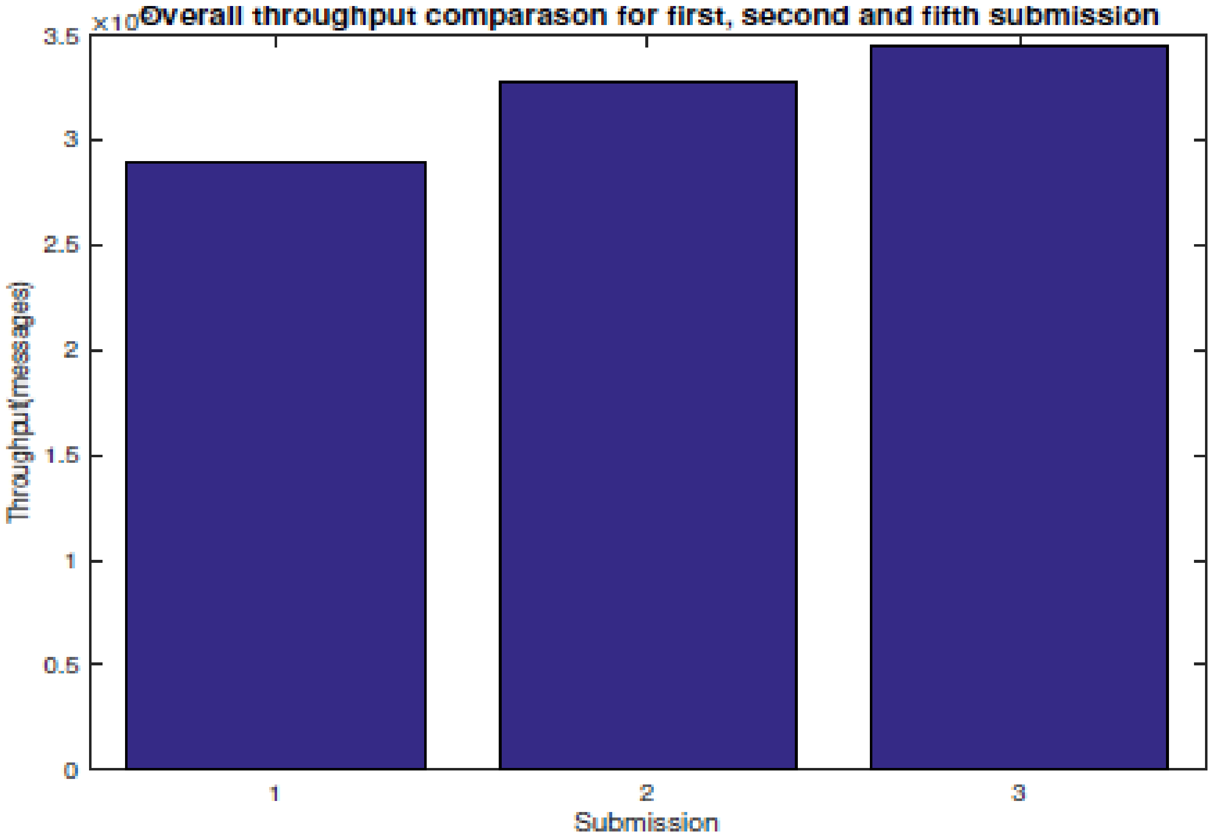


EVALUATION AND EXPERIMENTAL RESULTS

- The throughput comparison between default scheduler and multiple runs TOSS.



- The throughput comparison between TOSS and the exisiting scheduler under multiple runs.



CONTACT INFORMATION

- Y. Zhou is with TSYS School of Computer Science, Columbus State University, 4225 University Avenue, GA 31907-5645.

  E-mail: zhou_yi@columbusstate.edu

- Y.-Y. Liu, C.-W. Zhang, X.-P. Peng and X. Qin are with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, 36849-5347 USA.

  Email:

  yzl0098@tigermail.auburn.edu,czz0032@tigermail.auburn.edu,

  xzp0007@tigermail.auburn.edu ,

  xqin@auburn.edu