Topology and Traffic-Aware Two-Level Scheduler in a Heterogeneous Cluster

Leila Eskandari, Jason Mair,
Zhiyi Huang and David Eyers

University of Otago, New Zealand

Scheduling

- **Scheduling**: The allocation of available resources to a set of tasks
- **Data locality** in store-data-then-process
  - Goal: to put the tasks near the data to avoid moving data across the nodes
- **Communication awareness** in stream processing
  - Goal: to put the communicating tasks near each other in order to prevent moving data

Our Goal

- Find groups of communicating tasks and minimise the communication between the nodes
- Consider a potentially heterogeneous collection of resources within a cluster and reduce the inter-node traffic by using larger capacity nodes

T3-Scheduler

- We propose T3-Scheduler, a **T**opology and **T**raffic-aware **T**wo-level scheduler
- Why topology and traffic-aware?
  - Find the traffic pattern between the communicating tasks
- Why two-level?
  - First level: Which tasks should go into the same node
  - Second level: Which tasks should go into the same worker process

T3-Scheduler (Cont.)

- Monitoring
- Constructing a simplified graph
- Node selection
- First level of scheduling
- Second level of scheduling

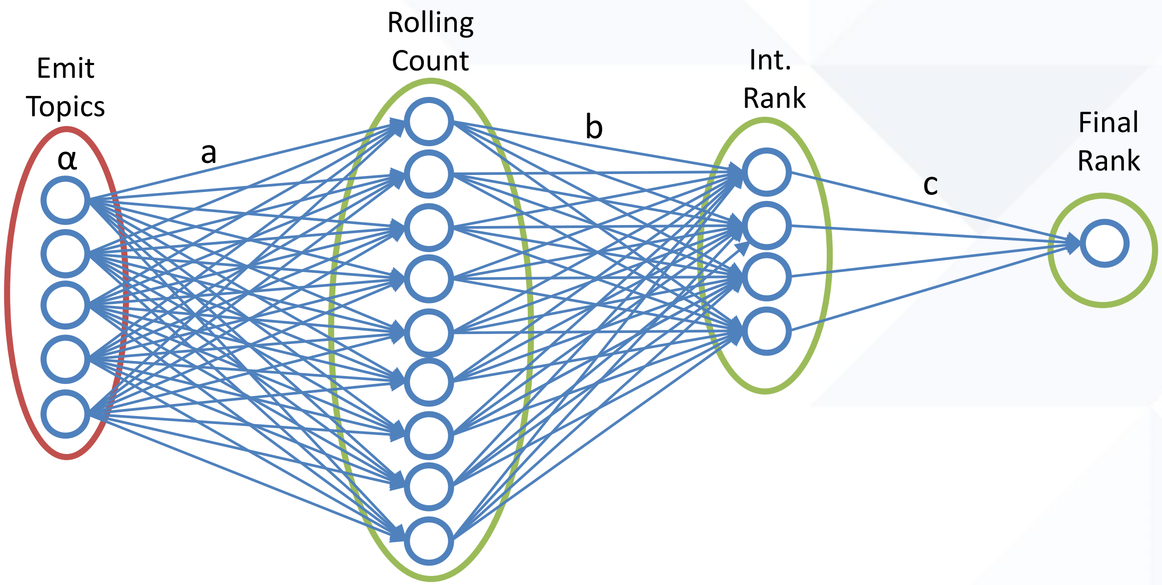
Monitoring

- Monitors the execution of the stream application
- Measures data transfer rate and task loads
- Regularly stores the collected values in a monitoring log
- Periodically reads when rescheduling

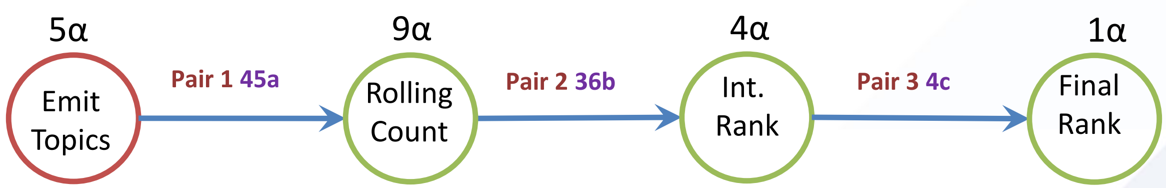
Constructing a Simplified Graph

- Constructs a simplified graph based on the online profile
- Vertex weight: sum of all the tasks load within each processing element
- Edge weight: sum of data transfer rate of communicating tasks

An Example



Emit
Topics
α
a
Rolling
Count
b
Int.
Rank
c
Final
Rank



5α
Emit
Topics
Pair 1 45a
9α
Rolling
Count
Pair 2 36b
4α
Int.
Rank
Pair 3 4c
1α
Final
Rank

Node Selection

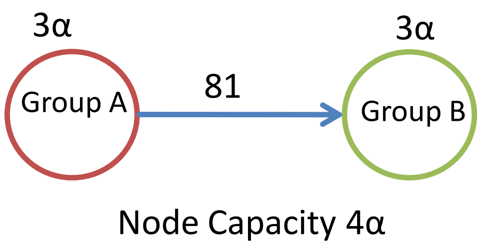
- Selects the highest capacity node
- Results in minimising the inter-node communication
- Fills a node with as many communicating tasks as possible, up to its capacity, and then moves to the next highest capacity node

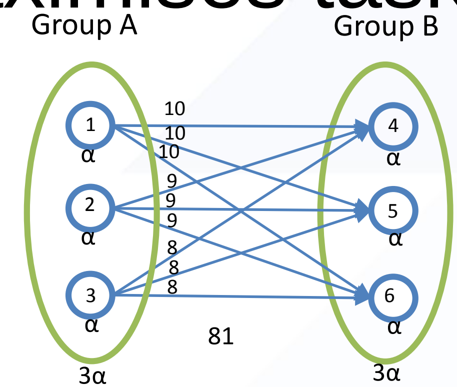
First Level of Scheduling

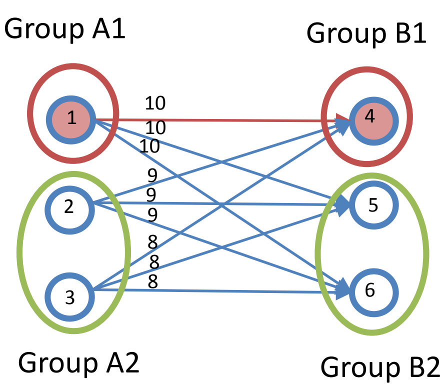
- Uses a greedy approach to find the group of tasks that communicate most
- Finds a starting point
- Expands the subgraph by finding the most highly connected neighbors
- Results in dividing the simplified graph into multiple parts
- Assigns each part to a compute node

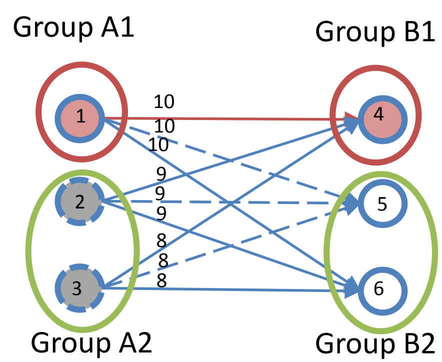
First Level of Scheduling (Cont.)

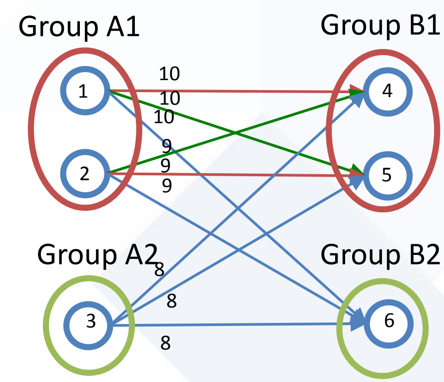
- Fine grained group pair partitioning
- Minimises edge cut, maximises task pairs





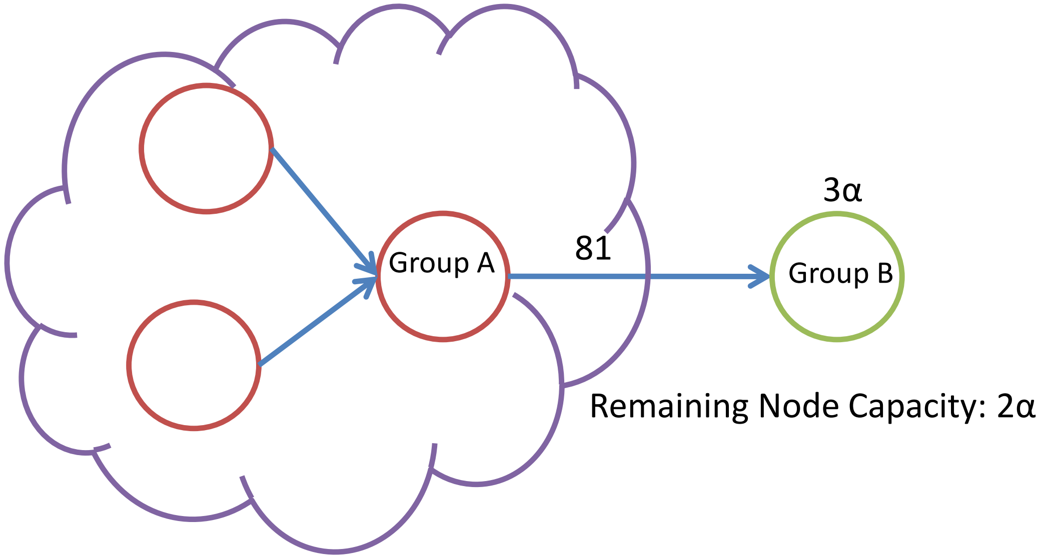


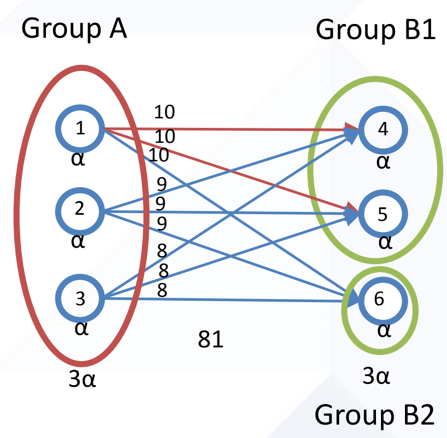




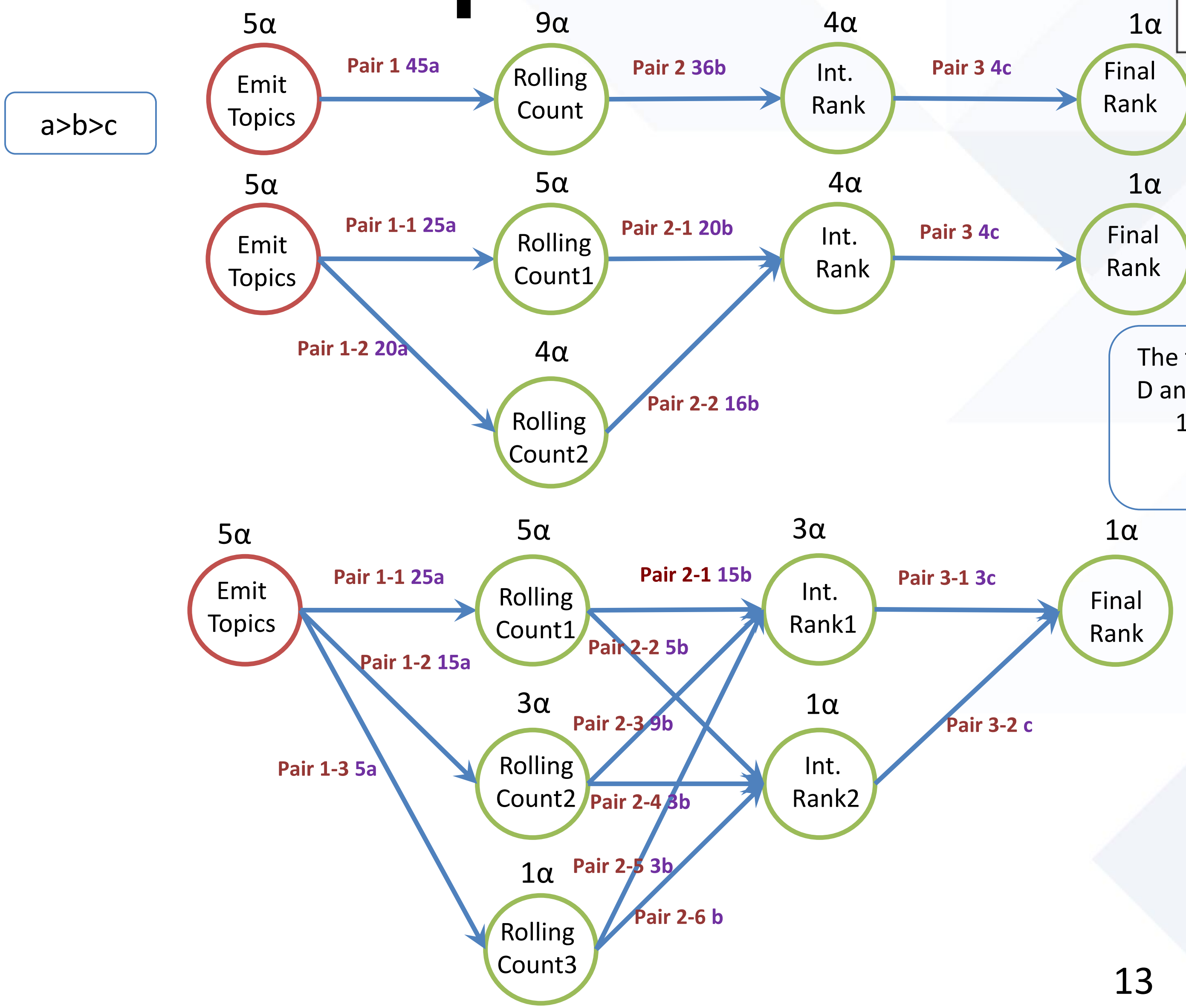
First Level of Scheduling (Cont.)

- Fine grained single group partitioning





An Example



The five worker nodes A, B, C, D and E have the capacities of 10α, 6α, 6α, 5α and 4α respectively.

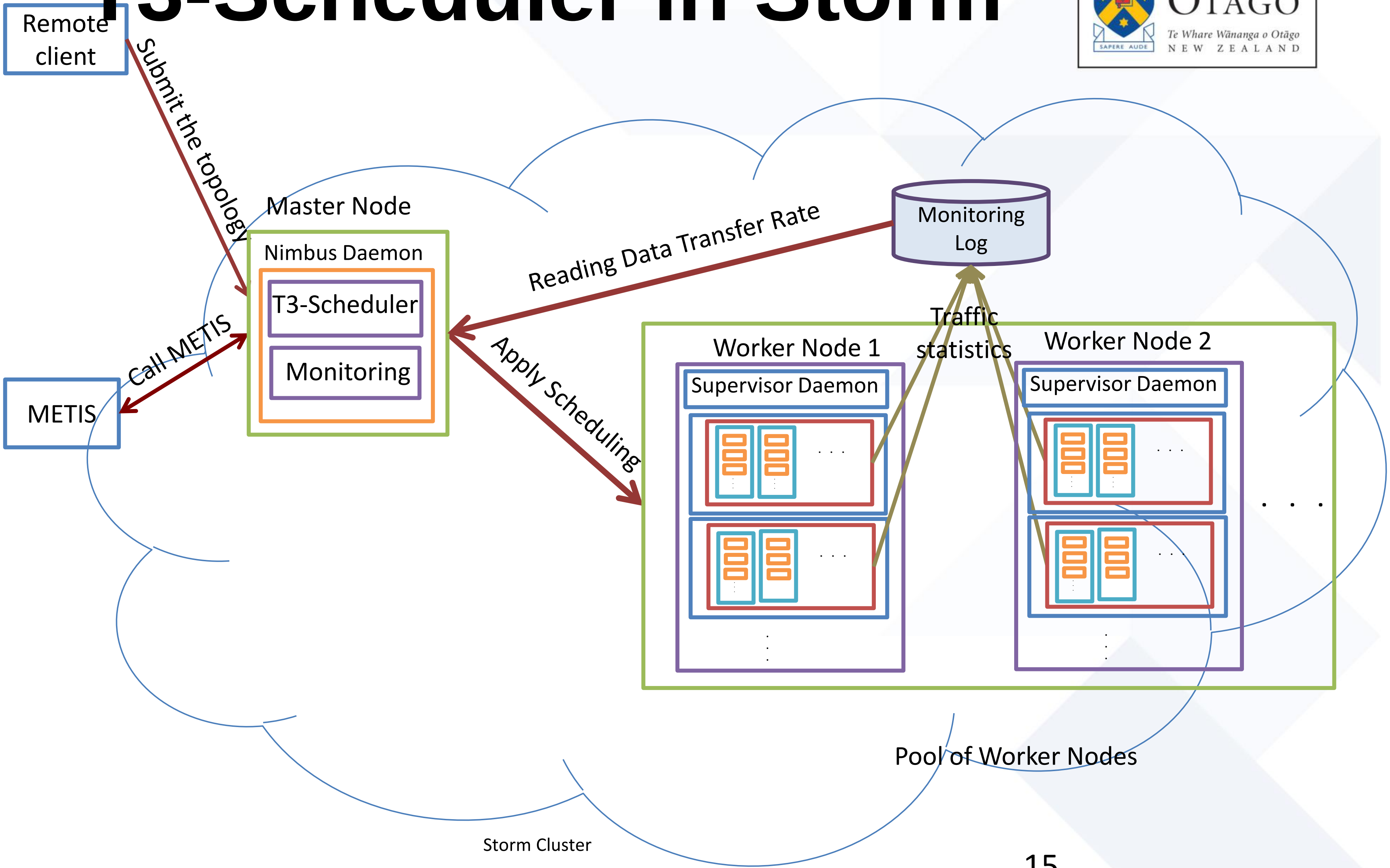
Second Level of Scheduling

- Use k-way partitioning to divide each subgraph of size t into a number of parts of size T
- T: The number of tasks per worker process

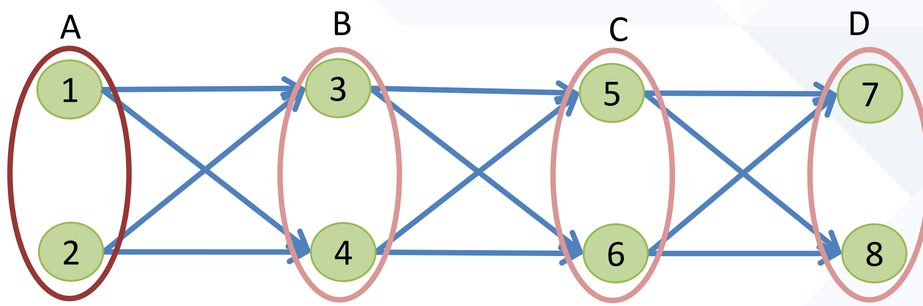
$$w = \left\lceil \frac{t}{T} \right\rceil$$

- Each part is assigned to a worker process

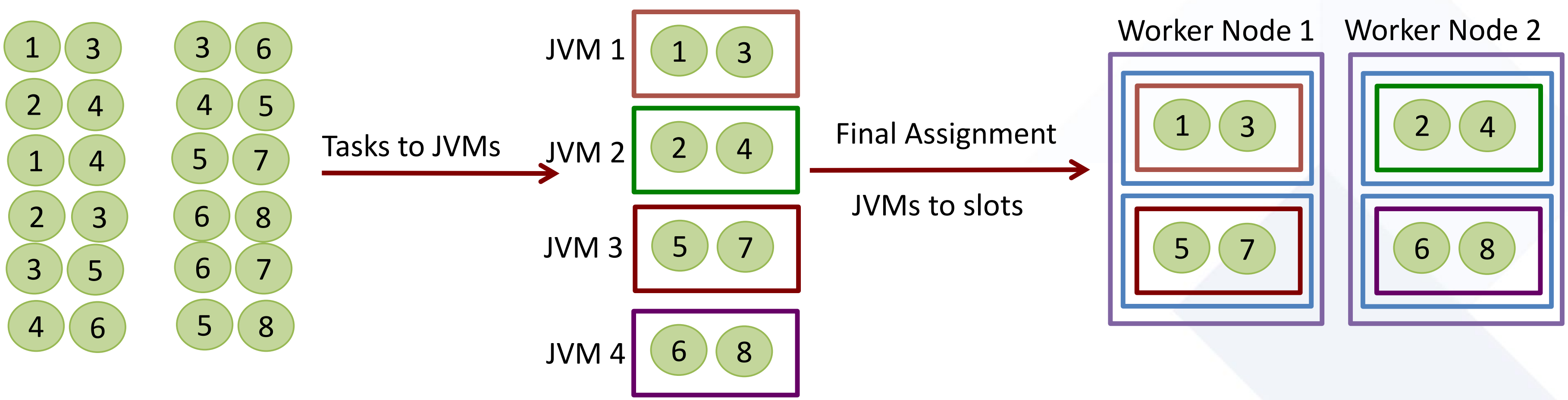
T3-Scheduler in Storm



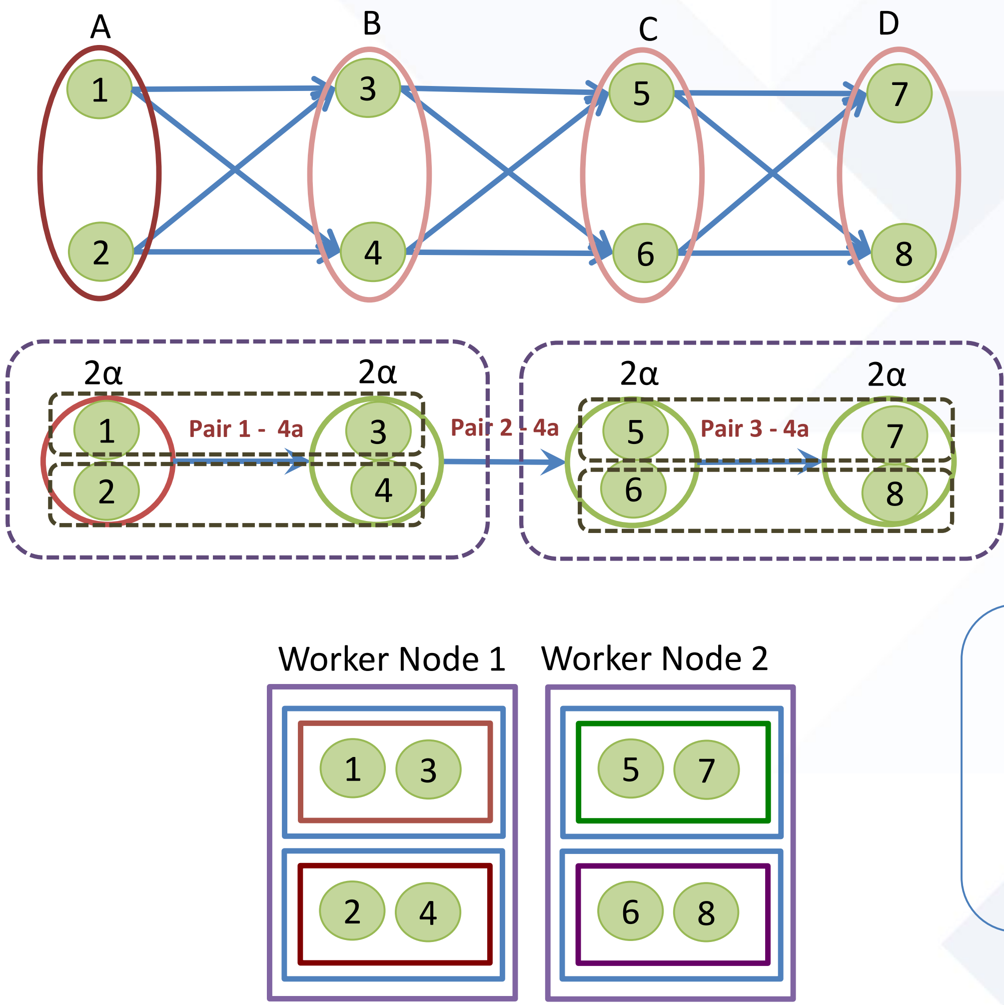
Online Scheduler



Each task's load is α.
The worker nodes 1 and 2 have the capacity of 4α



T3-Scheduler

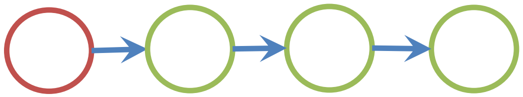


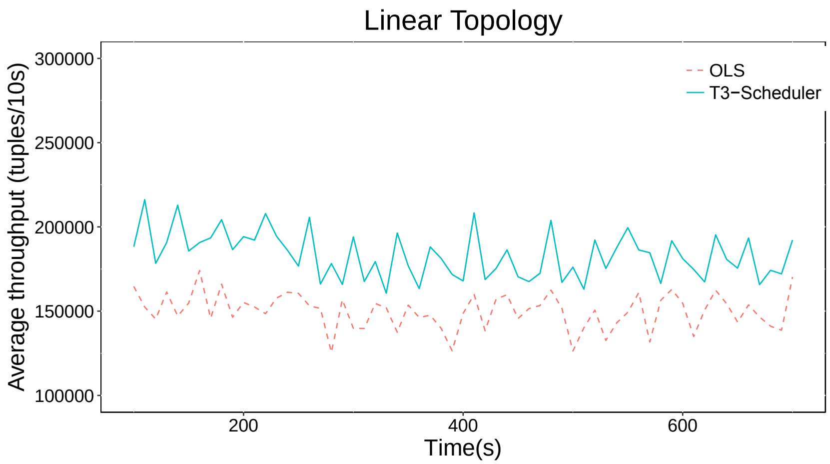
Each task's load is α. The worker nodes 1 and 2 have the capacity of 4α

Experimental Setup

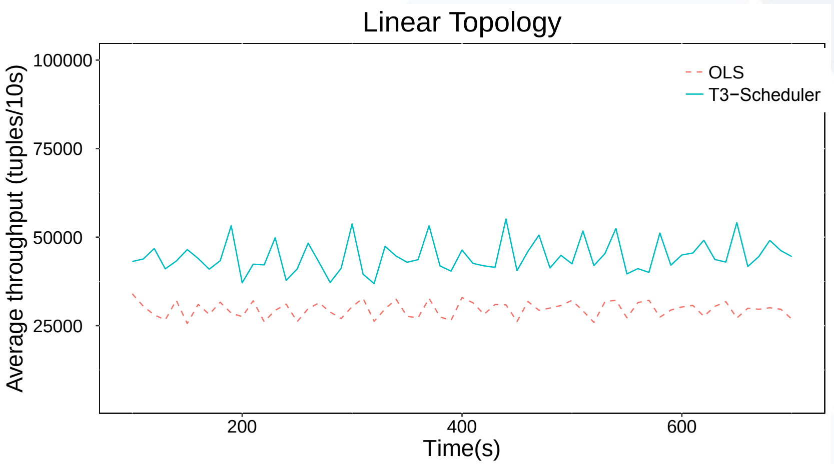
- A Storm cluster with 8 worker nodes, one master node and one ZooKeeper node
- Each node has a 2.7 GHz Intel CPU
- Four nodes with 4 cores and 4 GiB of RAM and four slots
- Four nodes with 2 cores and 2 GiB of RAM and two slots
- Connected by 1 Gbps network
- We use the average number of tuples executed in each bolt as performance metric

Micro-benchmark Topologies



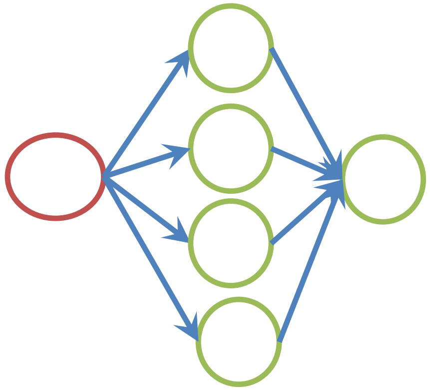


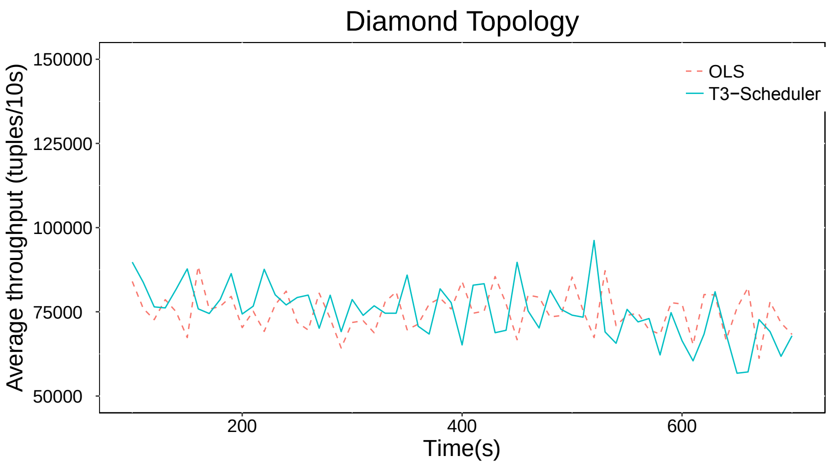
CPU-intensive



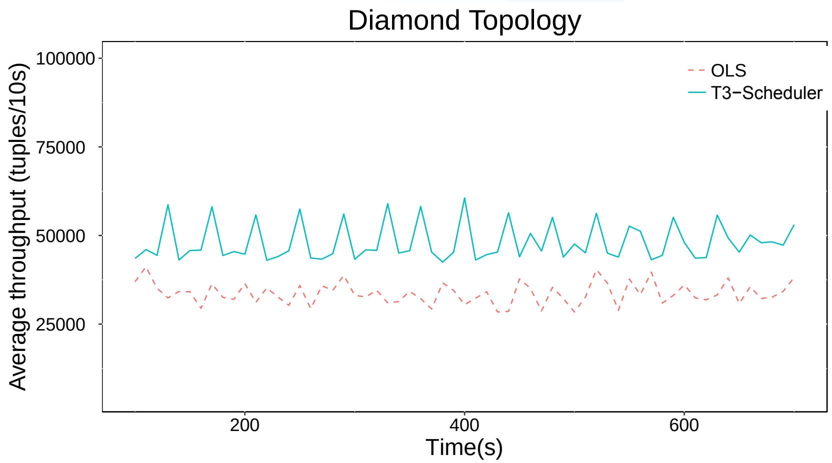
Network-intensive

Micro-benchmark Topologies (Cont.)



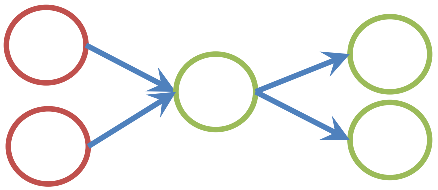


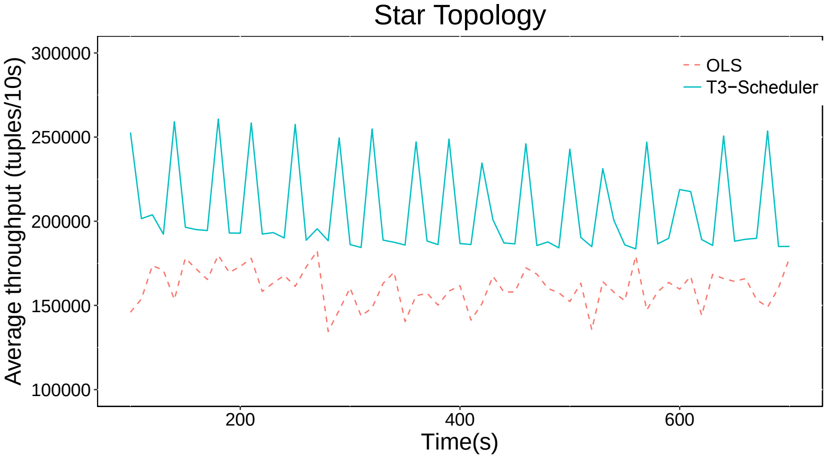
CPU-intensive



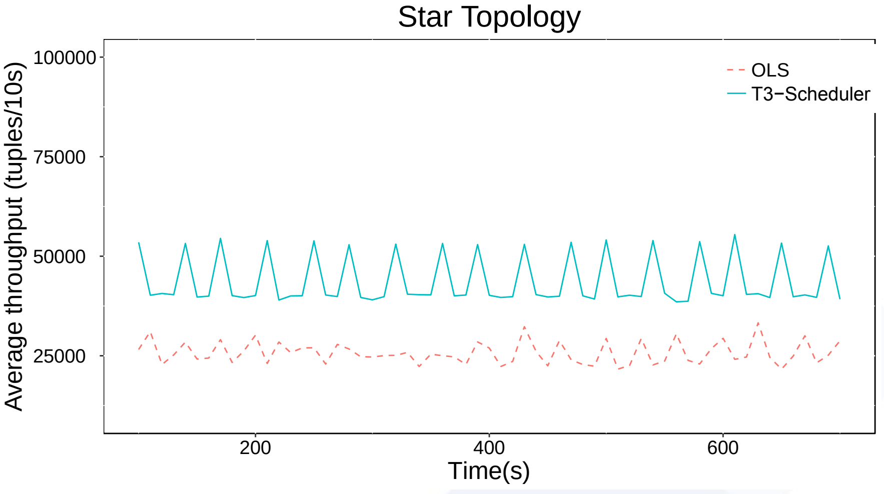
Network-intensive

Micro-benchmark Topologies (Cont.)





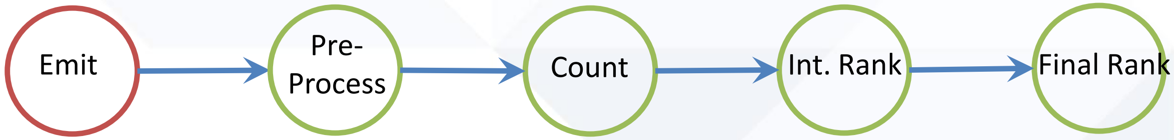
CPU-intensive

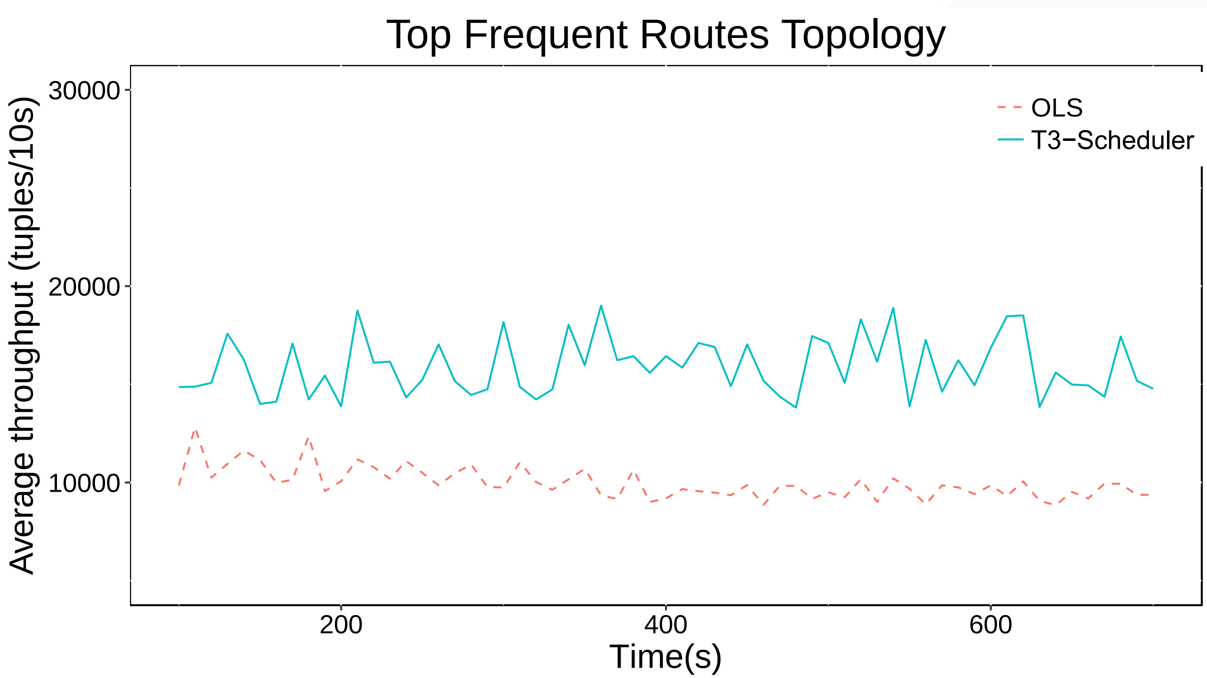


Network-intensive

Top Trending Routes

- NYC frequent routes





Conclusion and Future Work

- We reduced inter-node communication by:
  - Considering communication pattern
  - Prioritising nodes based on capacity and utilising each node
- We will compare T3-Scheduler with optimal placement for common layouts
- We will collect inter-node communication for our real-world application

Thank you!

Any questions?