CurTail: Distributed Cotask Scheduling with Guaranteed Tail-Latency SLO

Zhijun Wang, Hao Che and Hong Jiang

The University of Texas at Arlington

Zhijun Wang

Zhijun Wang (zhijun.wang@uta.edu) received his Ph.D degree in Computer Science and Engineering from The University of Texas at Arlington in 2005. He is an research associate in UTA. His research is focused on traffic control, resource management , job/task scheduling in cloud and edge computing.

Motivation

- Maximize resource utilization while meet tail latency SLO

1. Resource is over privisionning to provide tail latency service level objective (SLO) guarantee
2. How to maximize resource utilization while provide tail latency service level objective (SLO) guarantee is critical to increase the profit

- How to jointly allocate compute and networking resource

The compute and network resources should be jointly allocated to maximize resource utilization

Challenges

- How to translate job-level tail latency SLOs into precise runtime system resource demands at the task level.

- How to joint compute and networking resource allocation.

Existing solutions

- The existing solutions are centralized by design and hence not scalable, and focused on one performance metric, such as mean job completion time
- Most existing job scheduling and resource provisioning solutions are point by design, concerned with either compute or networking aspect of resource provisioning, rather than both jointly.
- The existing tail-latency-aware job scheduling solutions are exclusively focused on storage applications and jobs with fanout degree of one only.

Major contributions

- CurTail is a top-down approach, it decouples an upper job-level design from a lower task-level design and is independent on underlying systems, and hence it can be easily implemented.

- Curtail jointly allocates compute and networking resources based on the task resource demand to meet tail latency SLO, and hence can greatly improve system resource utilization.

Curtail architecture

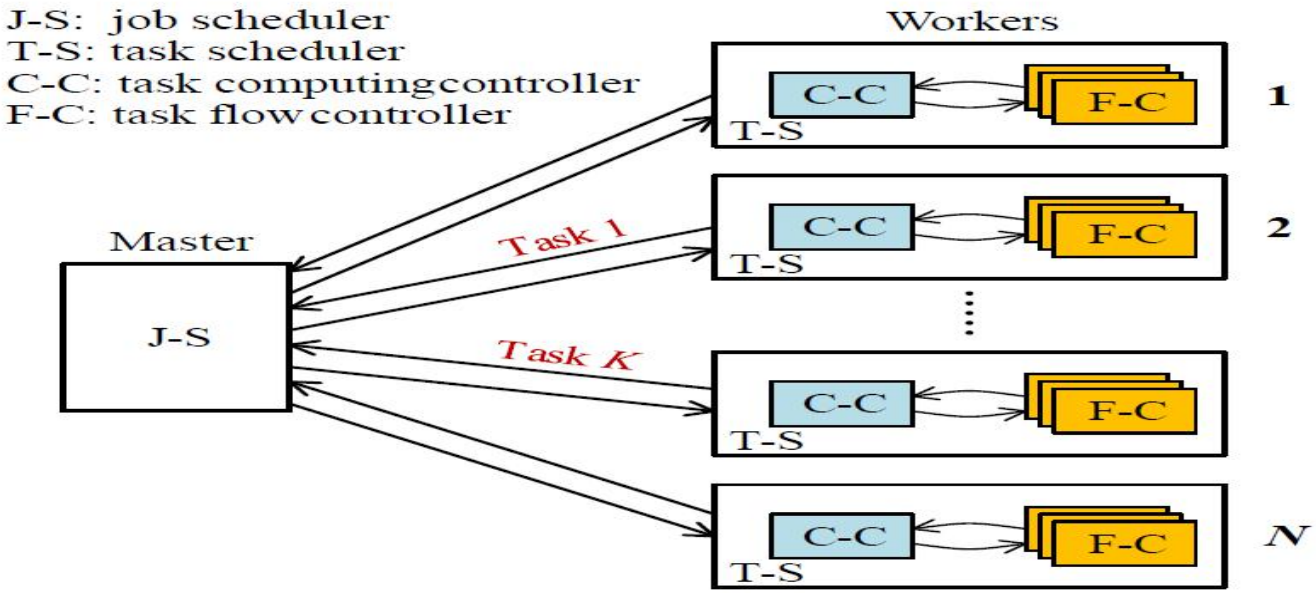


Fig. 1. A job scheduler, J-S, runs in a master node and distributed task schedulers, T-S, run in individual workers in the cluster, each of which is mainly composed of a compute controller, C-C, and a flow scheduler, F-C, per flow emitted from the worker.

Task budget translation

Give job tail latency SLO *xp* ,--the *p*th-percentile job tail-latency no more than *xp* and job fanout degree (K). The task budget can be described as (E_B, V_B) pair.

E_B: the mean task response time

V_B: variance of task response time

If the mean E and variance V of tasks in within (E<=E_B, V<=V_B), *xp* can be guaranteed.

Assume $\sqrt{V}/E = \alpha E$, then the budget can be mapped to the mean task response time *E*.

Task resource allocation

The task response time budget is divided by computing budget *Ec* and networking budget *En.*

--Set a minimum guaranteed networking rate

Λ^f, the network time is W^f / Λ^f

--The computing rate

Λ^c, the compute rate should be no less than is $(E_B - W^f / \Lambda^f) / W^C$

Task scheduling

A system with tail latency sensitive job (T-job) and back ground job (B-job)

--Allocate $p_{Kq}^{c} = \frac{T_q^c}{E_c}$ percent of total compute resource to task K.

--Batch job served as a first-in-first-out queuing policy

the compute resources except allocated to T-job are allocated to B-jobs.

Simulation Setup

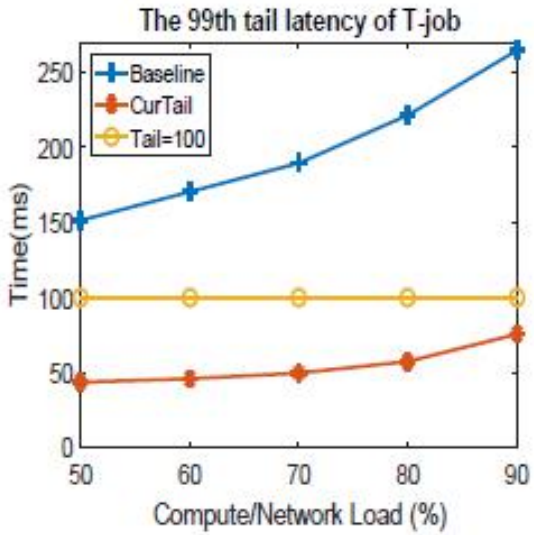
--5x5 leaf-spine network topology with 80 hosts

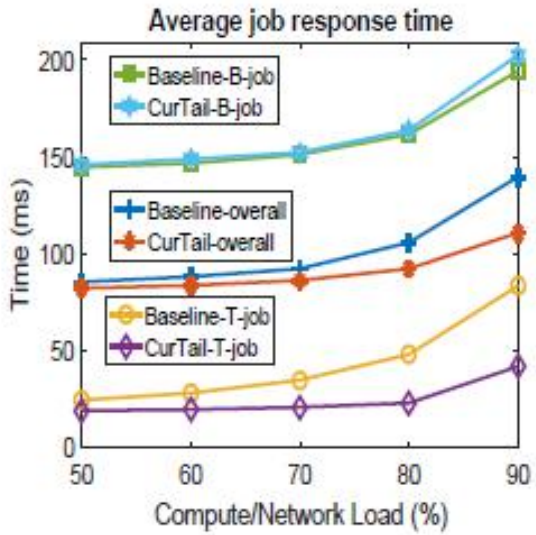
--T-job tail latency SLO is 100ms

--Mean task execution time B-job:40ms T-job: 5 ms

--Minimum rate is set as 0.75 Gbps

Performance-I



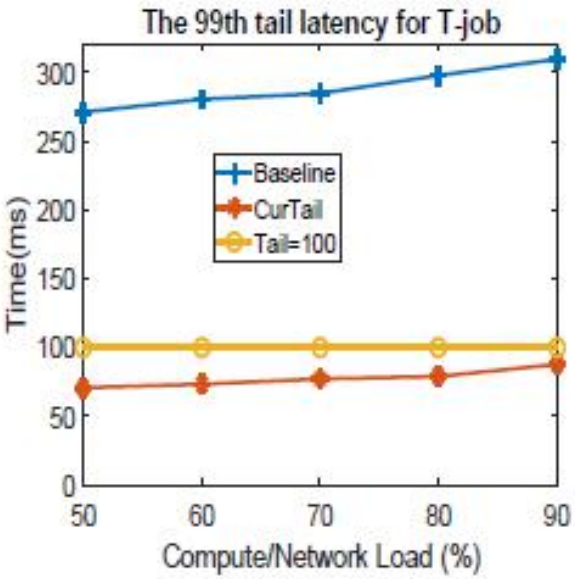


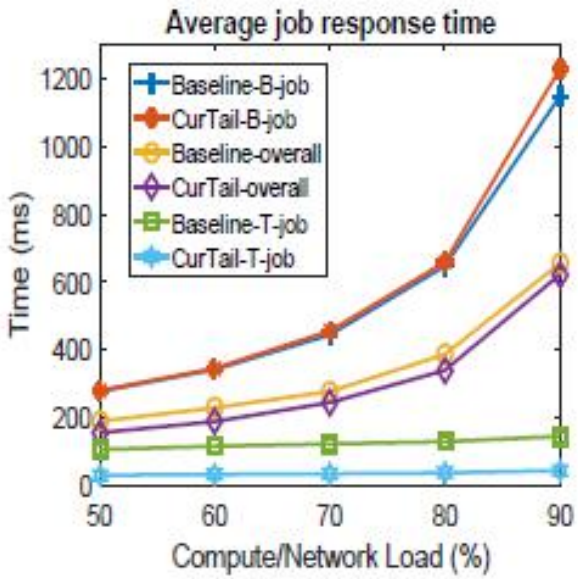
CurTail: provide tail latency guarantee for high load.

Better mean T-job and B-job response

Task dispatching with global information

Performance-II





CurTail: provide tail latency guarantee for high load.

Better mean T-job and B-job response

Random task dispatching

Conclusion:

Propose Curtail:

--decompose Job tail latency into task run budget

--jointly allocate compute and network resource

--provide tail latency SLO guarantee

--maximize resource utilization

Question??