Computing-Aware Traffic Steering
Internet-Draft
Intended status: Informational
Expires: 2 September 2026

Q. Li
Z. Luan
Pengcheng Laboratory
Y. Jiang
Tsinghua Shenzhen International Graduate School & Pengcheng Laboratory

1 March 2026

A Framework for Compute-Aware Task Placement and Traffic Steering in Heterogeneous Computing Networks
draft-luan-cats-catpts-00

Abstract

The increasing deployment of geographically distributed computing infrastructures equipped with heterogeneous compute resources (e.g., CPU, GPU, memory) has motivated new architectural approaches for jointly optimizing task placement and traffic steering. In heterogeneous computing networks, tasks must be assigned to compute-capable nodes while respecting multi-dimensional resource constraints and network bandwidth limitations.

This document presents a conceptual framework for Compute-Aware Task Placement and Traffic Steering (CATPTS). The framework models a computing network as a directed graph containing compute-capable nodes and forwarding-only nodes. Task execution location selection and two-stage traffic steering are jointly optimized under link bandwidth and multi-dimensional compute capacity constraints. The objective is to achieve global load balancing across compute and network resources.

This document defines the architectural principles, conceptual model, terminology, and optimization formulation underlying such systems. It does not specify protocol mechanisms.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-luan-cats-catpts/.

Discussion of this document takes place on the Computing-Aware Traffic Steering Working Group mailing list (mailto:cats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/cats/. Subscribe at https://www.ietf.org/mailman/listinfo/cats/.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Modern geographically distributed computing infrastructures integrate network transport and heterogeneous compute resources. In many scenarios, tasks are generated at source nodes, processed at intermediate compute-capable nodes, and deliver results to destination nodes. Examples include AI inference pipelines, edge-cloud collaboration, and distributed data processing.

Traditional traffic engineering focuses on routing source-destination flows. Traditional compute scheduling assumes tasks are assigned within geographically centralized clusters. However, in geographically distributed heterogeneous networks, task placement decisions directly affect network traffic patterns, and traffic steering decisions affect network-wide compute utilization.

Therefore, task deployment and traffic steering cannot be optimized independently.

This document introduces a unified framework in which:

* Tasks select execution nodes from candidate compute-capable nodes.

* Task data flows consist of two stages:

  - Input path: source node to execution node

  - Output path: execution node to destination node

* Traffic may be split across multiple candidate paths.

* Both link bandwidth and multi-dimensional compute capacities are constrained.

* The objective is to minimize maximum compute resource utilization (and optionally network congestion).

2. Background and Architectural Motivation

2.1. Heterogeneous Compute Networks

Emerging computing networks contain:

* Forwarding-only nodes
* Compute-capable nodes
* Heterogeneous resource types (CPU, GPU, memory, etc.)

These resources support distributed task execution rather than simple packet forwarding.

2.2. Coupling Between Task Placement and Traffic Steering

Selecting a compute node for a task determines:

* Input traffic injected into the network
* Output traffic delivered to the destination
* Compute resource load at the execution node

Task placement therefore changes the traffic matrix, while steering decisions determine feasibility under bandwidth constraints.

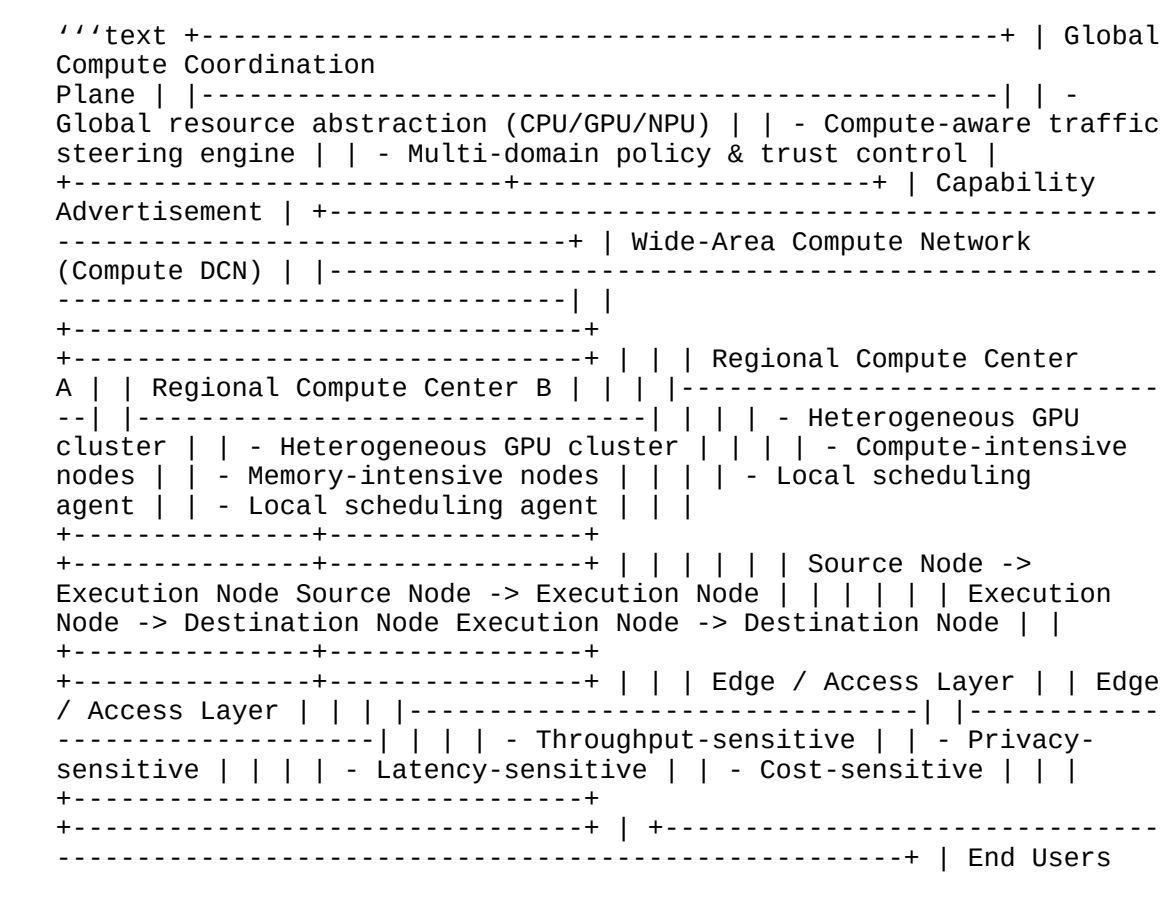
This coupling motivates joint optimization.

3. Compute-Aware Task Routing Framework

3.1. Design Principles

* Joint Optimization: Placement and steering are solved together.
* Two-Stage Flow Structure: Each task induces input and output flows.
* Multi-Dimensional Resource Awareness: Compute nodes have vector capacities.
* Load Balancing Objective: Minimize worst-case resource utilization.

4. High-Level Architecture



5. Network Model

The network is represented as:

G = (V, E)

Where:

* V: set of nodes

* E: set of directed links

* M: compute-capable nodes, subset of V

* R: forwarding-only nodes, subset of V
* Each link e has bandwidth capacity B[e]
* Each compute node m has d-dimensional capacity vector C[m]

6. Task Model

Each task i is defined by:

* Source node s_i
* Destination node d_i
* Compute demand vector w_i
* Input data size b_i[in]
* Output data size b_i[out].
* Candidate execution node set M_i

Each task selects exactly one execution node from M_i.

7. Resource Constraints

7.1. Link Capacity Constraint

For each link e:

LinkLoad[e] is equal or less than B[e] * U[link][max]

Where U[link][max] is a configurable utilization threshold.

7.2. Compute Capacity Constraint

For each compute node m and resource dimension k:

Sum of assigned task demand in dimension k is equal or less than C[m][k] * U[node][max]

Where U[node][max] is a configurable utilization threshold.

8. Optimization Objective

Minimize the maximum compute utilization across all compute nodes.

This corresponds to min寰杜ax load balancing.

The framework may be extended to incorporate weighted trade-offs between compute utilization and link congestion.

9. Applicability

This framework applies to:

* AI inference distribution
* Edge-cloud collaboration
* Distributed accelerator networks
* Wide-area compute fabrics
* Compute-aware traffic engineering

It is particularly relevant when:

* Compute resources are heterogeneous
* Tasks have multiple candidate execution locations
* Network bandwidth is constrained
* Load balancing is required

10. Security Considerations

Execution nodes process application data. Placement decisions may depend on:

* Data locality requirements
* Administrative domain boundaries
* Trust relationships

This document does not define security mechanisms.

11. Terminology

COMPUTE-CAPABLE NODE:
A node that can execute tasks and provides multi-dimensional compute resources.

FORWARDING-ONLY NODE:
A node that forwards traffic but does not execute tasks.

TASK PLACEMENT:
Selection of an execution node for a task.

INPUT PATH:
Route from task source to execution node.

OUTPUT PATH:
Route from execution node to destination.

COMPUTE LOAD:
Aggregate resource utilization at a compute node.

LINK LOAD:
Aggregate traffic load on a link.

MIN-MAX LOAD BALANCING:
Optimization objective minimizing worst-case resource utilization.

12. Informative References

[Elf] "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading", September 2021, <https://dl.acm.org/doi/abs/10.1145/3447993.3448628>.

[RFC9556] "Internet of Things (IoT) Edge Challenges and Functions", RFC 9556, n.d., <https://www.rfc-editor.org/rfc/rfc9556>.

Appendix A. IANA Considerations

This document has no IANA actions.

Appendix B. References

B.1. Normative References

{::include normative}

B.2. Informative References

{::include informative}

Authors' Addresses

Qing Li
Pengcheng Laboratory
Email: liq@pcl.ac.cn

Zeyu Luan
Pengcheng Laboratory
Email: luanzy@pcl.ac.cn

Yong Jiang
Tsinghua Shenzhen International Graduate School & Pengcheng Laboratory
Email: jiangy@sz.tsinghua.edu.cn