1

Introduction

Cloud-edge-device computing systems are at the forefront of modern technology, serving as a critical enabler for diverse applications ranging from real-time data processing to autonomous vehicles. This chapter provides an overview of these systems, outlines the key tasks they perform, discusses scheduling challenges, and sets the stage for the subsequent chapters of this book.

1.1 Cloud-edge-device Computing Systems

The evolution of computing paradigms has led to the integration of cloud computing, edge computing, and device-level processing into a cohesive ecosystem. Each layer in this triad has distinct capabilities and plays a complementary role in modern applications:

- **Cloud Computing:** It offers virtually unlimited computational resources and storage, making it ideal for data-intensive analytics, centralized control, and long-term archiving. For example:
  - In smart grid management, cloud platforms analyze historical electricity usage data to forecast demand and optimize distribution.
  - In precision medicine, cloud-based systems store and analyze genomic data from thousands of patients to identify patterns and recommend personalized treatments.
- **Edge Computing:** It operates closer to data sources, reducing the latency associated with transmitting data to the cloud and responding in real time. It is well-suited for latency-sensitive applications, such as:
  - In manufacturing, edge devices process data from industrial sensors to detect anomalies like overheating or mechanical wear in real time.
  - In video surveillance, edge nodes perform real-time face or license plate recognition to trigger immediate alerts without involving cloud servers.

Intelligent Scheduling of Tasks for Cloud-Edge-Device Computing Systems, First Edition.
Biao Hu et al. © 2025 The Institute of Electrical and Electronics Engineers, Inc.
Published 2025 by John Wiley & Sons, Inc.

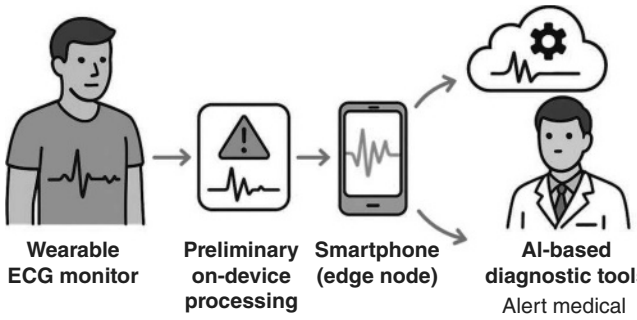

Figure 1.1 Wearable electrocardiogram (ECG) assisted with edge computing.

- **Device-level Processing:** It involves computation directly on endpoint devices, such as embedded systems, smartphones, or Internet of Things (IoT) sensors. These devices are crucial for collecting and preprocessing data:
  - In home automation, a smart thermostat uses on-device sensors to detect occupancy and adjust temperature settings autonomously.
  - In environmental monitoring, a water quality sensor embedded in a river can analyze basic chemical parameters locally and only send alerts when thresholds are breached.

This hierarchical architecture enhances system responsiveness and reduces unnecessary data transmission. For example, in a smart transportation system, vehicles continuously collect data on location, speed, and traffic conditions. Edge units installed on road infrastructure aggregate this information to manage traffic lights and avoid congestion. Meanwhile, the cloud aggregates and analyzes long-term traffic patterns to inform urban planning and infrastructure development.

Another emerging application is in remote healthcare. As shown in Figure 1.1, a wearable electrocardiogram (ECG) monitor detects irregular heart rhythms and processes preliminary results on-device. If anomalies are detected, the data is passed to a nearby smartphone (acting as an edge node) for detailed processing. If a critical pattern is confirmed, the data is transmitted to the cloud for analysis by artificial intelligence (AI)-based diagnostic tools and alerting medical personnel.

1.2 Tasks

Tasks in cloud-edge-device systems are the atomic units of computation. They can be simple or complex, and their characteristics often determine their ideal placement in the architecture. Based on execution dependencies and timing requirements, tasks fall into the following categories:

- **Real-time Dependent Tasks:** These require strict timing and often involve data dependencies between tasks. Examples include:
  - In an autonomous drone delivery system, real-time video analysis for obstacle detection must be followed by trajectory adjustment within milliseconds.
  - In augmented reality (AR) applications, rendering and positioning of virtual objects must occur in sequence and with minimal delay to maintain a smooth user experience.
- **Independent Tasks:** These have no interdependencies and can be executed in parallel or out of order. They are ideal for distributed scheduling. Examples include:
  - In financial services, fraud detection models may process user transaction logs independently to detect suspicious patterns.
  - In scientific simulations, multiple Monte Carlo simulations with different seed values can run in parallel across cloud servers.
- **Mixed-criticality Tasks:** These combine tasks of varying urgency and importance, requiring differentiated handling:
  - In aerospace systems, navigation control (high-criticality) must be prioritized over cabin temperature regulation (low-criticality).
  - In e-health applications, a fall detection alert from a wearable must take precedence over routine vitals logging.

Effectively categorizing and managing these tasks enables systems to make informed decisions about task offloading, redundancy, and scheduling. For instance, in a connected vehicle network, engine diagnostics (low-criticality, batch tasks) can be deferred to the cloud, while lane-change detection (high-criticality, real time) is executed at the edge or device level.

1.3 Task Scheduling

Scheduling governs how tasks are allocated across the cloud-edge-device hierarchy. Efficient scheduling is essential for optimizing energy, meeting deadlines, and ensuring a seamless user experience. Several key challenges must be addressed:

- **Energy Efficiency:** Energy conservation is vital, especially for battery-powered devices and sensors. Scheduling strategies may:
  - Batch low-priority data uploads to the cloud during low-energy-cost periods.
  - Utilize low-power modes on edge devices when not actively processing.

  For example, a wildlife tracking system on GPS collars may only activate data transmission when the animal moves beyond a geofence, thereby extending battery life.

- **Resource Utilization:** Balancing load prevents both overload and underuse. Effective strategies include:
  - Offloading heavy analytics tasks to the cloud during peak edge load.
  - Prefetching and caching frequently accessed data at edge nodes.

  In smart retail, when a surge of users interacts with a digital kiosk, computation for recommendations can be load-balanced to nearby edge nodes while preserving UI responsiveness.
- **Real-time Constraints:** Meeting deadlines is essential in safety-critical applications. For instance:
  - In telemedicine, patient monitoring systems must deliver abnormal health event notifications within seconds.
  - In cooperative adaptive cruise control, vehicle-to-vehicle communication delays must be minimal to avoid collisions.
- **Workload Variability:** Real-world systems must dynamically adapt to changing workloads. Scheduling algorithms can:
  - Use machine learning to predict upcoming workloads and pre-allocate resources accordingly.
  - Apply load-shedding strategies when resources are temporarily unavailable.

  In cloud gaming, servers may preload player profiles and game state during anticipated high-traffic periods (e.g., evenings or weekends) to maintain consistent performance.

Advanced scheduling frameworks are increasingly leveraging AI-driven optimization and feedback loops. For example, reinforcement learning-based schedulers can continuously learn optimal task placement strategies under varying network, compute, and energy constraints.

1.4 Outline of the Book

This book explores the design, implementation, and optimization of scheduling techniques in cloud-edge-device computing systems. It systematically addresses the challenges of real-time task execution across heterogeneous computing layers through theoretical models, algorithmic innovations, and practical applications. The content is organized as follows:

- Chapter 2 focuses on scheduling mixed real-time tasks in automotive systems with vehicular networks. It introduces a hybrid scheduler and provides offline and online task assignment strategies, along with complexity analysis and schedulability testing.

- Chapter 3 explores workload-aware scheduling of real-time independent tasks in cloud environments. It discusses virtual CPU and power models, formulates the scheduling problem, and presents solutions for both single and multiple server settings with energy-efficient strategies.
- Chapter 4 examines energy-minimized scheduling for real-time-dependent tasks in the cloud. It provides task placement strategies that balance real-time constraints with power consumption goals, utilizing both forward and backward scheduling techniques.
- Chapter 5 transitions to vehicular edge computing and discusses scheduling strategies for real-time-dependent tasks. It introduces decentralized auction-based methods and group scheduling to support multiple applications with power-aware task execution.
- Chapter 6 delves into the complexity of scheduling mixed-criticality dependent tasks in vehicular edge systems. This chapter presents task decomposition, frequency prediction, and dynamic programming techniques to ensure real-time guarantees in multimode operational environments.
- Chapter 7 bridges theory and practice by illustrating real-world applications of scheduling technologies in domains such as smart cities, industrial automation, healthcare monitoring, and autonomous systems.
- Chapter 8 summarizes the key contributions and insights from the preceding chapters and outlines potential directions for future research, including adaptive scheduling, AI-assisted task management, and cross-layer orchestration.

Each chapter concludes with a summary to reinforce key concepts and facilitate understanding, while extensive performance evaluations and case studies ground the discussion in practical relevance.

1.5 Summary

The cloud-edge-device computing paradigm has transformed the digital landscape by enabling distributed intelligence and responsive services. As modern applications grow in complexity and require low latency, high availability, and energy efficiency, the importance of effective task scheduling becomes paramount.

This introductory chapter has laid the foundation for understanding the structure and objectives of cloud-edge-device systems, categorized task types, and highlighted critical scheduling challenges such as energy efficiency, resource utilization, real-time constraints, and workload variability. These insights establish the context for the detailed exploration that follows.

The subsequent chapters of this book will delve deeper into task scheduling strategies tailored to different scenarios and system architectures. Through rigorous analysis, algorithm design, and practical application, this book aims to equip readers – researchers, practitioners, and advanced students alike – with the tools and knowledge to design more efficient and intelligent cloud-edge-device computing systems.