Future Trends & Emerging Technologies

Chapter 6

Outline

- ❑ New Paradigms
  - Sky computing
  - Computing power network
  - Meta computing
  - Future direction: cross-domain collaboration
- ❑ Integration with AI
  - Why edge computing is needed for LLMs
  - Training and distributed edge learning
  - Inference, model splitting, and model caching
  - Generative AI, applications, and challenges
- ❑ 6G and Edge
  - Basic characteristics of 6G
  - Mutual influence of 6G and edge computing
  - Network slicing and reconfigurable intelligent surfaces
  - Resource sharing, offloading, applications, and challenges
- ❑ Bias mitigation
  - Edge computing concepts and Open research problems
  - Typical OEC architectures and Kodan
  - Collaborative computing, applications, and challenges
- ❑ Summary & Practice

New Paradigms

Why new paradigms

Limitations of today's edge

- Resources often remain isolated within one provider.
- Cloud platforms may use different architectures and APIs.
- Cross-domain trust and security are difficult.
- Computing power is distributed across cloud, edge, and device layers.

Emerging response

- Sky computing connects heterogeneous clouds.
- Computing power networks pool cloud + edge capacity.
- Meta computing targets cross-provider collaboration with security.

Long-term direction

- Sky computing connects heterogeneous clouds.
- Computing power networks pool cloud + edge capacity.
- Meta computing targets cross-provider collaboration with security.

Sky Computing

Cross-cloud collaboration

Core idea. Treat multiple cloud providers as one programmable “sky,” reducing provider-specific barriers and enabling applications to move across clouds.

Compatibility layer

- Masks differences in cloud APIs.
- Uses common interfaces for storage, networking, and runtime.
- Simplifies multi-cloud application development.

Intercloud layer

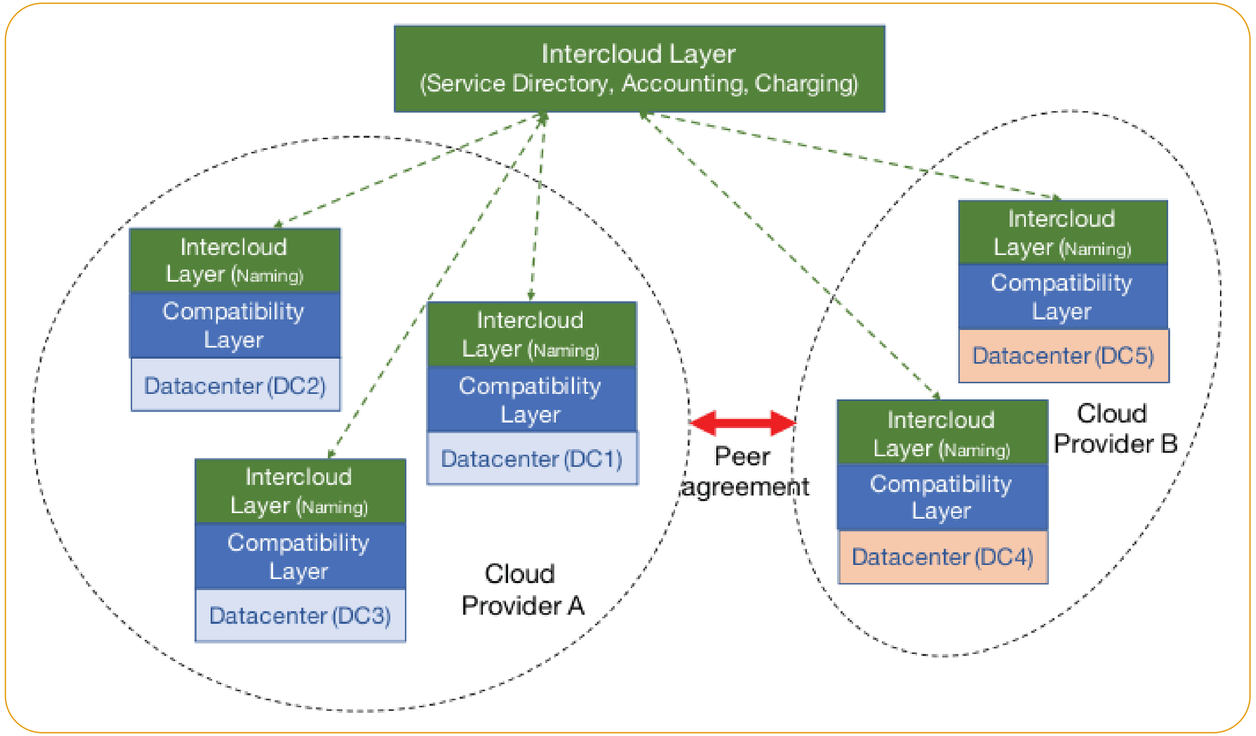
- Accepts applications and policies.
- Selects a cloud based on cost, location, capacity, or time.
- Coordinates execution across providers.

Tiemo Bang et al. “SkyPIE: A fast & accurate oracle for object placement”. In: Proceedings of the ACM on Management of Data 2. 1 (2024), pp. 55:1–55:27. DOI: 10.1145/3639310.

Ion Stoica and Scott Shenker. “From cloud computing to sky computing”. In: HotOS '21: Workshop on Hot Topics in Operating Systems, Ann Arbor, Michigan, USA, June, 1–3, 2021. Ed. by Sebastian Angel, Baris Kasikci, and Eddie Kohler. ACM, 2021, pp. 26–32. DOI: 10.1145/3458336.3465301.

Sky Computing

Cross-cloud collaboration



Sky computing architecture

➢ Key takeaways

- ➢ Multi-cloud should appear as one coherent service
- ➢ Internet interoperability motivates the architecture.
- ➢ Three layers — compatibility, intercloud, and peering.
- ➢ **Reciprocal peering** is the key proposed enabler

Ion Stoica and Scott Shenker. “From cloud computing to sky computing”. In: HotOS '21: Workshop on Hot Topics in Operating Systems, Ann Arbor, Michigan, USA, June, 1–3, 2021. Ed. by Sebastian Angel, Baris Kasikci, and Eddie Kohler. ACM, 2021, pp. 26–32. DOI: 10.1145/3458336.3465301.

Computing Power Network (CPN)

Compute as a networked utility

Goal

Package distributed computational power and deliver it where users need it—analogous to electricity flowing through a power grid.

Evolution

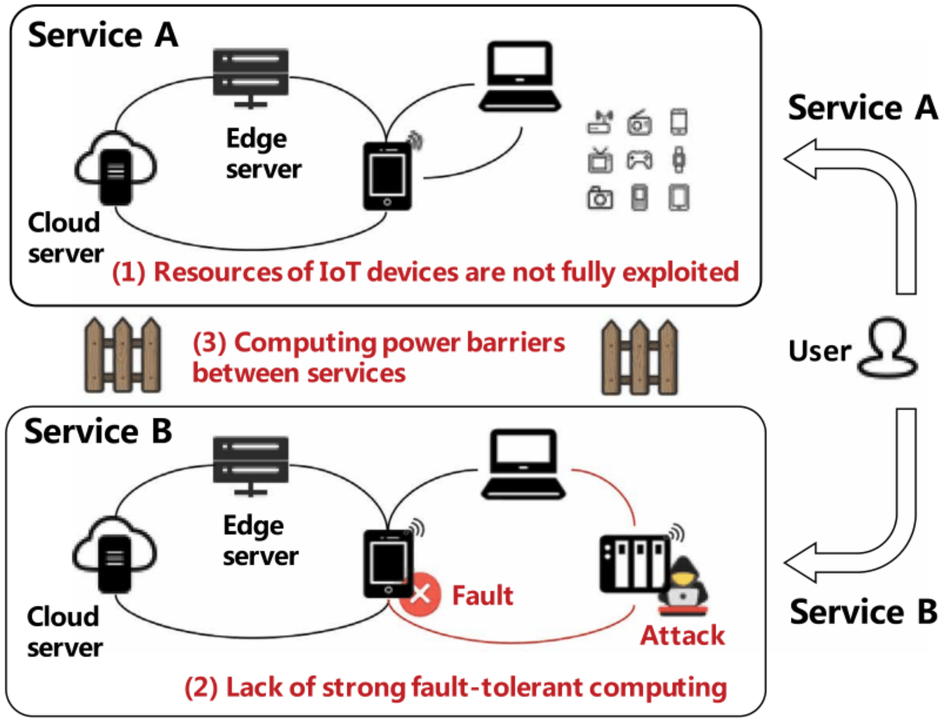
- Single data center execution
- Scheduling across homogeneous centers
- Heterogeneous data-center selection
- Coordinated execution across heterogeneous resources

Relation to edge

- Pools resources across cloud, edge, and device layers.
- Supports unified scheduling and resource access.
- Still early: full network-wide integration remains an open research problem.

Meta Computing

Cross-domain collaboration



Problems of existing computing paradigms

Provider barriers

- Nodes across service providers do not automatically interoperate.
- Different management and trust domains block collaboration.

Underused resources

- IoT and edge resources may be fragmented or idle.
- Limited-service capability when each provider acts alone.

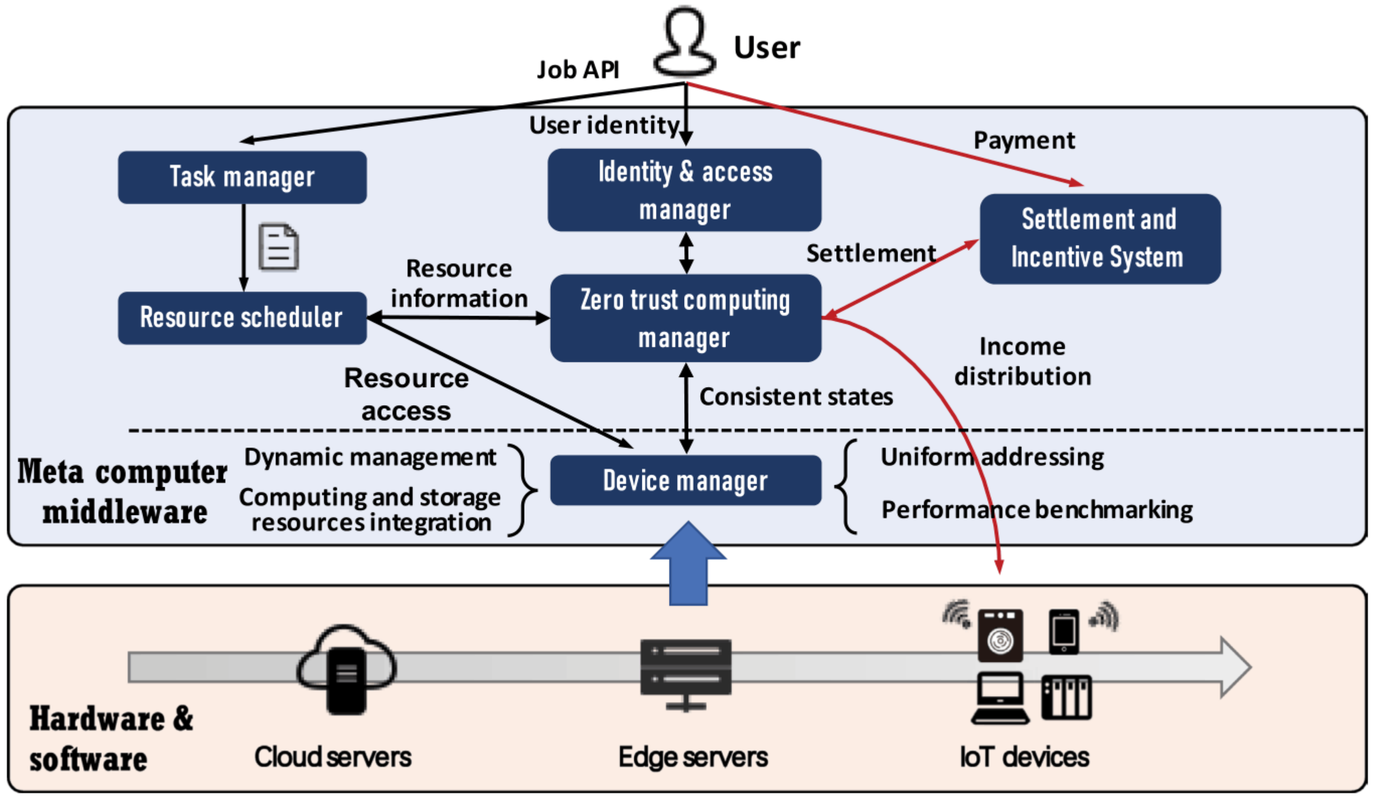
Security & resilience

- Cross-provider computation introduces identity, access, settlement, and fault-tolerance challenges.
- Meta computing aims for “network-as-a-computer.”

Xiuzhen Cheng et al. “Meta computing”. In: IEEE Network 38. 2 (2024), pp. 225–231. ISSN: 1558-156X. DOI: 10.1109/MNET003.2300092.

Meta Computing

Cross-domain collaboration



Problems of existing computing paradigms

Zero-trust computing manager

- Identity and access management for users and devices
- Resource scheduling and task management
- Distributed ledger for state consistency
- Settlement and incentive mechanisms

Device manager

- Uniformly addresses cloud servers, edge servers, and IoT devices
- Maintains resource and performance information
- Supports dynamic management across heterogeneous resources

Xiuzhen Cheng et al. “Meta computing”. In: IEEE Network 38. 2 (2024), pp. 225–231. ISSN: 1558-156X. DOI: 10.1109/MNET003.2300092.

Comparison

How the new paradigms extend edge computing

Name	Features	Relationship to edge computing
Sky computing	Cross-cloud collaboration	Cloud extensions
Computing power network	Data center and edge integration	Cloud and edge extensions
Meta computing	Cross-service provider collaboration	Cross-domain security extensions

Shared direction: heterogeneous, cross-provider collaboration across the complete device–edge–cloud computing path.

Future Direction: Cross-Domain Edge Collaboration

Research agenda

Interoperability

Common abstractions and APIs across providers, clouds, edge platforms, and devices.

Trust

Identity verification, access management, security, and settlement across service providers.

Scheduling

Coordinate heterogeneous resources according to application requirements and available computing capacity.

Unified path

Coordinate heterogeneous resources according to application requirements and available computing capacity.

Integration with Artificial Intelligence

Why Need Edge Computing

Name	R & D team	Parameter scale
GPT-1	OpenAI	0.11 B
GPT-3	OpenAI	175 B
GPT-4	OpenAI	1.8 T
Sora	OpenAI	7 B
Gemini 1.5 pro	Google	175 B
Gemma-2B (opensource)	Google	2 B
Gemma-7B (opensource)	Google	7 B
StableDiffusion	Qualcomm	65 B
Llama (opernsource)	Meta	7-65 B
ChatGLM (opensource)	Tsinghua University	6-130 B
Hunyuan	Tencent	100 B
ERNIE Bot	Baidu	260 B

Model parameter scales

Integration with Artificial Intelligence

Why Need Edge Computing

Resource limits

- Many IoT endpoints cannot host large models.
- Memory, compute, and energy constraints remain severe.
- On-device execution may require aggressive compression.

Heterogeneity

- Devices differ in accelerators, memory, network quality, and latency needs.
- A fixed deployment strategy cannot fit every endpoint.

Privacy & responsiveness

- Keeping raw data local reduces exposure.
- Nearby edge resources can provide low-latency assistance when the endpoint is insufficient.

Training and Fine-Tuning with Edge Computing

Centralized vs Distributed

Centralized edge learning

- Devices upload data to an edge server.
- Training happens on distributed edge GPUs or nearby infrastructure.
- Practical only where substantial edge compute is available.

Federated edge learning

- Participants train local copies of the model.
- Model parameters/updates are exchanged.
- Privacy improves, but communication can be heavy for LLM-scale models.

Split learning

- The model is divided into smaller submodels.
- Devices train only part of the network.
- The full model is coordinated across device, edge, and cloud.

Deepak Narayanan et al. “Efficient large-scale language model training on GPU clusters using megatron-LM”. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 1–15. ISBN: 978-1-4503-8442-1. DOI: 10.1145/ 3458817.3476209.

Guanqun Wang et al. “Cloud-device collaborative learning for multimodal large language models”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, pp. 12646–12655.

Federated Learning vs. Split Learning for LLMs

Trade-off

Federated Learning

- Each participant trains the full model or a local replica.
- Communication is dominated by model updates.
- Simple conceptual boundary: data stays local, parameters move.

Split Learning

- Different devices execute/train different model partitions.
- Different devices focus on training specific model subparts, which are ultimately integrated into a complete model.
- The chapter identifies split learning as a potentially more viable paradigm for large models on resource-constrained devices.

For LLM-scale edge learning, partitioning the model can be more realistic than asking every endpoint to host the entire model.

Zheng Lin et al. “Efficient parallel split learning over resource-constrained wireless edge networks”. In: IEEE Transactions on Mobile Computing (2024), pp. 1–16. ISSN: 1558-0660. DOI: 10.1109/TMC.2024.3359040.

Guanqun Wang et al. “Cloud-device collaborative learning for multimodal large language models”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, pp. 12646–12655.

Efficient LLM Inference at the Edge

Reduce compute + communication

Compression

- Pruning
- Quantization
- Knowledge distillation

Speculative decoding

- A smaller model first produces a result on the edge device.
- Data can then be sent to edge/cloud services to execute the complete large model.
- Tabi uses calibrated confidence scores to decide whether a query should be rerouted to the LLM.

Early exit

- Execute only part of the model instead of the entire model when early exiting is possible.
- This can reduce inference cost and latency.

Token / feature reduction

- Compress intermediate representations.
- Reduce transmission cost in split inference.

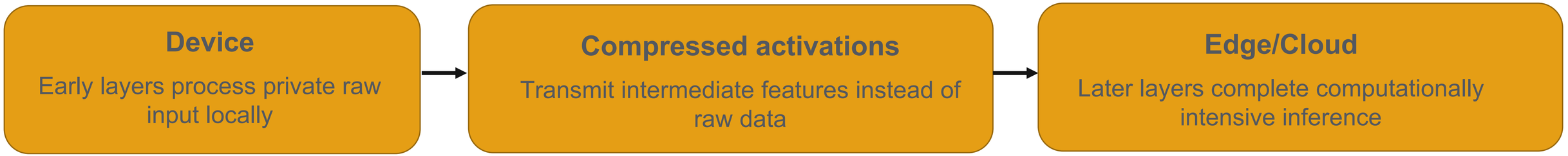
Qingqing Cao et al. BTR: Binary Token Representations for Efficient Retrieval Augmented Language Models. May 2024. DOI: 10.48550/arXiv.2310.01329. arXiv: 2310.01329 [cs].

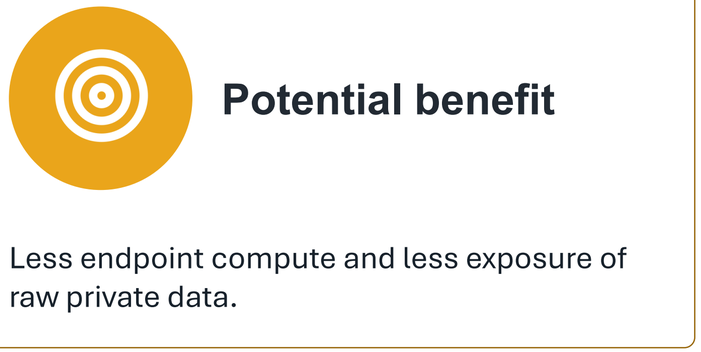
Yanxi Chen et al. EE-LLM: Large-Scale Training and Inference of Early-Exit Large Language Models with 3D Parallelism. June 2024. DOI: 10.48550/arXiv.2312.04916. arXiv: 2312.04916 [cs].

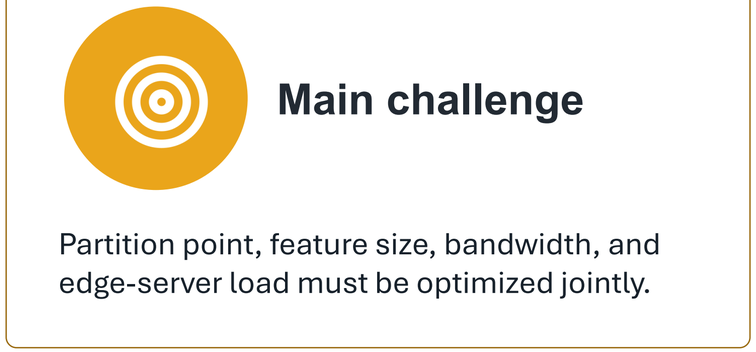
Yiding Wang et al. “Tabi: An efficient multi-level inference system for large language models”. In: Proceedings of the 18th European Conference on Computer Systems. EuroSys '23. New York, NY, USA: Association for Computing Machinery, May 2023, pp. 233–248. ISBN: 978-1-4503-9487-1

Split LLM Inference Across Device and Edge

Model partitioning







Model Caching at the Edge

Reuse shared model components

Why caching helps

Fine-tuned large models can share identical submodels. Caching those shared submodels at the edge can reduce memory overhead for learning and inference.

Where it helps

- Split learning
- Split inference
- Distributed edge learning
- Batch processing to accelerate inference

Example

TrimCaching explores parameter-sharing edge caching for AI model downloading—a useful building block for collaborative inference

Guanqiao Qu et al. TrimCaching: Parameter-sharing Edge Caching for AI Model Downloading. arXiv:2404.14204 [cs]. May 2024. DOI: 10.48550/arXiv.2404.14204.

Generative AI at the Edge

Why local generation matters

Real-time decisions

Generate or reason locally for autonomous systems and interactive applications.

Lower bandwidth

Process video, audio, and sensor data locally instead of streaming everything to the cloud.

Model optimization

Compression makes generative models more feasible on smartphones and IoT devices.

Hybrid architecture

Lightweight edge processing + offload of heavy generation to nearby servers or cloud.

Guanqiao Qu et al. TrimCaching: Parameter-sharing Edge Caching for AI Model Downloading. arXiv:2404.14204 [cs]. May 2024. DOI: 10.48550/arXiv.2404.14204.

AI Applications and Open Challenges

From capability to trust

Applications

- Robots and multi-robot planning
- Autonomous driving and in-vehicle assistants
- Industrial design and domain assistants

Explainability

How can LLMs remain explainable, especially during model splitting and submodel extraction in edge computing?

Verifiability + trimmability

How can edge execution be verified, and how can selected model modules be trimmed or replaced by smaller models to accelerate inference?

Guanqiao Qu et al. TrimCaching: Parameter-sharing Edge Caching for AI Model Downloading. arXiv:2404.14204 [cs]. May 2024. DOI: 10.48550/arXiv.2404.14204.

6G and Edge

Mutual influence

Edge enables 6G

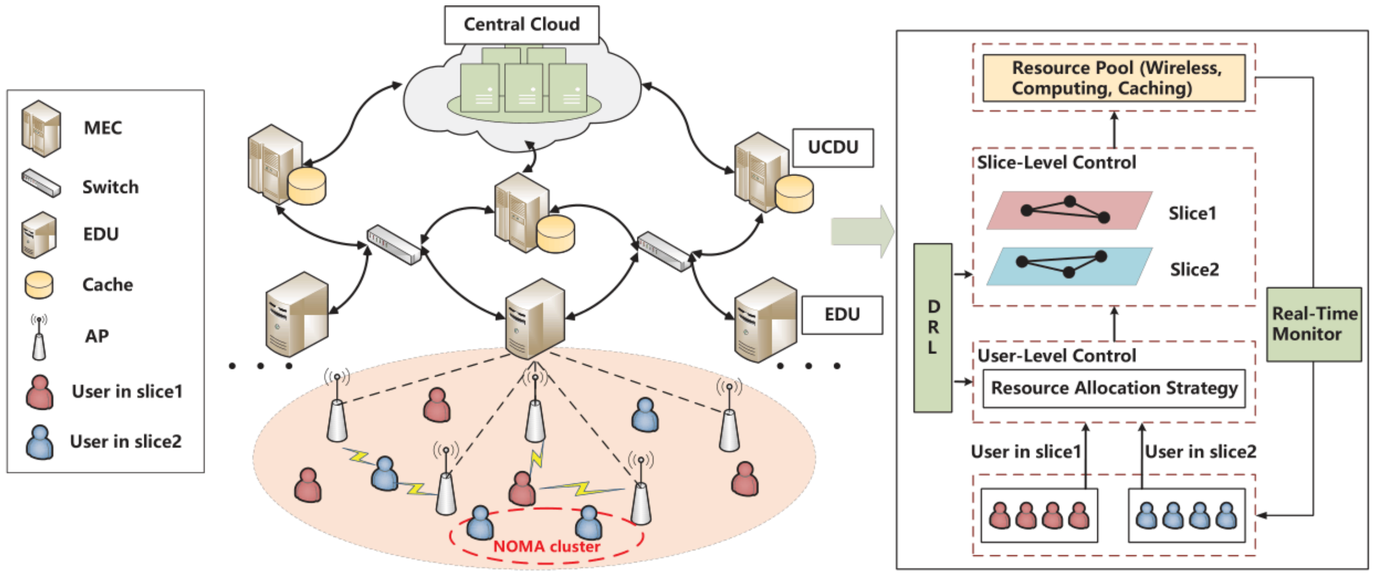
- Base-station compute for AI-driven wireless optimization
- Coordination across multiple base stations
- Personalized AI services
- Communication, computation, and caching close to users

6G enables edge

- Higher bandwidth and lower latency for offloading
- More flexible resource sharing and mobile edge execution
- Integrated sensing and communication
- Satellite access integrated into a unified space–ground network

Edge-Enabled Network Slicing for 6G

Network slicing



A hierarchical network slicing architecture

Why the edge matters

- Traffic decisions are most effective near network entry points.
- Caching at base stations accelerates data access.
- Edge compute supports decentralized control.

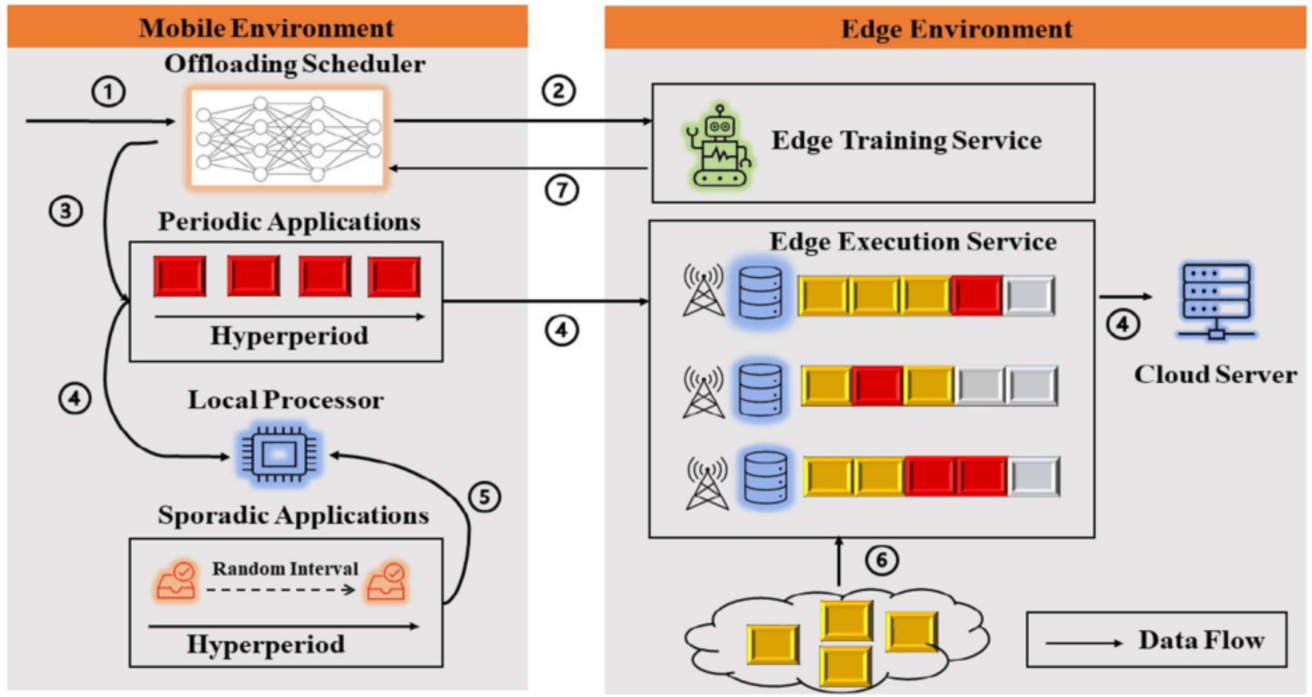
Architecture

Jointly allocate communication, computation, and caching resources, combining conventional optimization with multi-agent deep reinforcement learning.

Feng Ye et al. “Intelligent hierarchical NOMA-based network slicing in cell-free RAN for 6G systems”. In: IEEE Transactions on Wireless Communications 23. 5 (2023), pp. 4724–4737

Edge-Enabled Network Slicing for 6G

Multi-access edge learning-based offloading



Architecture of multi-access edge learning-based offloading (MELO). The data that flow in the MEC system include: (1) environment states; (2) training samples; (3) offloading policy; (4) periodic jobs; (5) sporadic jobs; (6) periodic jobs from other mobile devices; and (7) parameters of edge actor network

EdgeGo

- Stationary edge servers handle strict real-time tasks.
- Mobile edge servers process non-real-time tasks or temporarily augment capacity.
- Task offloading and execution are decoupled from one fixed location.

MELO

- Reinforcement learning solves real-time offloading/scheduling decisions.
- Training can occur on edge servers while inference runs on devices.
- Targets real-time task scheduling in 6G-enabled multi-access edge computing.

Hui Huang, Qiang Ye, and Yitong Zhou. “6G-empowered offloading for realtime applications in multi-access edge computing”. In: IEEE Transactions on Network Science and Engineering 10. 3 (2023), pp. 1311–1325. ISSN: 2327-4697. DOI: 10.1109/TNSE.2022.3188921.

6G Applications and Open Challenges

Opportunity

Cloud / edge VR

High bandwidth plus nearby rendering resources can support richer VR, holographic media, and lighter terminals.

Sensing + digital twins

Communication signals can also sense position and motion, enabling factories and other environments to be modeled in real time.

Research challenges

- Robust, flexible, and scalable 6G system architecture
- Coordination of compute across dense base stations
- Privacy for integrated sensing and communication

Edge Computing in Space Exploration

Computation in orbit

Traditional satellite

Primarily relays signals or sends collected data to Earth. Data processing on orbit is limited, so downlink capacity can become a bottleneck.

Orbital edge computing

Processes selected data on satellites and can coordinate computation across a constellation before transmitting results to Earth.

Why LEO matters

- Lower latency than higher orbits
- Global coverage via constellations
- Growing onboard compute capability

- **OEC advantages and typical architecture**

Low latency

LEO links can provide low communication latency, especially when services execute on orbit and avoid multiple terrestrial forwarding hops.

Global coverage

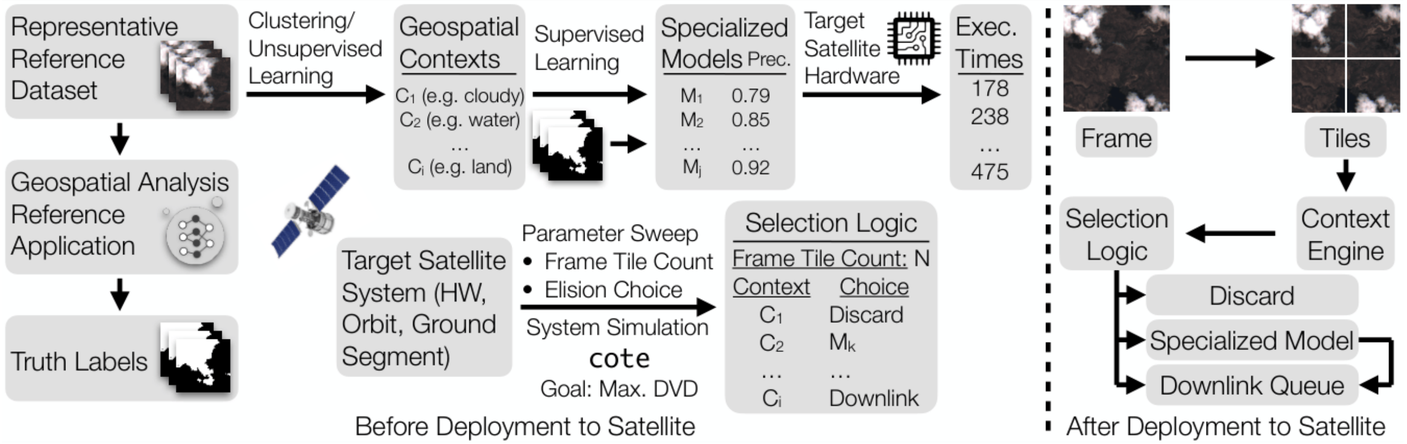
Satellite constellations can extend computation to regions that lack dense terrestrial cloud or edge infrastructure.

Two OEC modes

- End–edge collaboration: satellite processes sensed data before downlink.
- Edge–edge collaboration: satellites share or offload tasks across the constellation.

Computing on Satellites

Kodan



Kodan architecture design

Prelaunch phase

- Customize models for satellite hardware and mission conditions.
- Plan models around processing time, location, and resource limits.
- Select what “high-value” data should be retained.

Postlaunch phase

- Run onboard models on sensed data.
- Transmit only the most valuable or relevant results.
- Use scarce downlink bandwidth more effectively.

Bradley Denby et al. “Kodan: Addressing the computational bottleneck in space”. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, Mar. 2023, pp. 392–403. ISBN: 978-1-4503-9918-0.

OEC Collaboration, Applications, and Challenges

Collaborative computing

- Decompose user tasks across satellites.
- Offload tasks to peer satellites or ground resources.
- Optimize energy, communication windows, and execution placement.

Applications

- Earth observation and remote sensing
- AR/VR collaboration with global coverage
- Space exploration and distributed in-space processing

Challenges

- Limited research platforms and simulators
- Specialized system software / interfaces
- Rapidly changing topology and short service windows
- Seamless handoff and resource management

Summary

- Future edge systems will cross clouds, providers, and administrative domains instead of remaining isolated.
- Training, inference, splitting, caching, and generative AI make the edge an active intelligence layer.
- Edge compute becomes part of the communications infrastructure, while 6G improves edge mobility and service quality.
- OEC moves computation into satellite constellations to reduce downlink pressure and provide global services.

Practice

1. What is the relationship between edge computing and a new distributed computing paradigm, using one as an example?

2. How does edge computing enhance artificial intelligence, particularly large language model technology?

3. What are the main characteristics of 6G, and how do 6G and edge computing technologies influence each other?

4. Discuss the challenges and opportunities of integrating orbital computing technology with edge computing

Project

Deploy large language models, such as Edge-LLM (**https://github.com/GATECH-EIC/Edg e-LLM**), LlamaEdge (**https://github.com/LlamaEdge/LlamaEdge**), on heterogeneous hardware and explore the segmentation of specific models at different layers in an edge computing environment, observing the changes in transmission costs and computational costs

➢ What you'll do

1. **Deploy an LLM** Use Edge-LLM or LlamaEdge as the project model.
2. **Use heterogeneous hardware** Run the model in an edge computing environment across different hardware.
3. **Segment at different layers** Explore multiple layer-level segmentation points in the selected model.
4. **Compare configurations** Observe how changing the segmentation point changes the project costs.

➢ What to measure

- **Transmission cost** Record the transmission cost for each segmentation configuration.
- **Computational cost** Record the computational cost for each segmentation configuration.
- **Layer-split effect** Compare how both costs change when the model is segmented at different layers.
- **Hardware comparison** Compare the observed costs across the heterogeneous hardware used in the deployment.

Example: Deploy LlamaEdge on heterogeneous hardware, test several layer split points, and report the transmission and computational cost for each configuration.

References

- Tiemo Bang et al. “SkyPIE: A fast & accurate oracle for object placement”. In: Proceedings of the ACM on Management of Data 2. 1 (2024), pp. 55:1–55:27. DOI: 10.1145/3639310.
- Ion Stoica and Scott Shenker. “From cloud computing to sky computing”. In: HotOS '21: Workshop on Hot Topics in Operating Systems, Ann Arbor, Michigan, USA, June, 1–3, 2021. Ed. by Sebastian Angel, Baris Kasikci, and Eddie Kohler. ACM, 2021, pp. 26–32. DOI: 10.1145/3458336.3465301.
- Deepak Narayanan et al. “Efficient large-scale language model training on GPU clusters using megatron-LM”. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 1–15. ISBN: 978-1-4503-8442-1. DOI: 10.1145/3458817.3476209.
- Guanqun Wang et al. “Cloud-device collaborative learning for multimodal large language models”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024, pp. 12646–12655.
- Qingqing Cao et al. BTR: Binary Token Representations for Efficient Retrieval Augmented Language Models. May 2024. DOI: 10.48550/arXiv.2310.01329. arXiv: 2310.01329 [cs].
- Yanxi Chen et al. EE-LLM: Large-Scale Training and Inference of Early-Exit Large Language Models with 3D Parallelism. June 2024. DOI: 10.48550/arXiv.2312.04916. arXiv: 2312.04916 [cs].
- Yiding Wang et al. “Tabi: An efficient multi-level inference system for large language models”. In: Proceedings of the 18th European Conference on Computer Systems. EuroSys '23. New York, NY, USA: Association for Computing Machinery, May 2023, pp. 233–248. ISBN: 978-1-4503-9487-1.
- Guanqiao Qu et al. TrimCaching: Parameter-sharing Edge Caching for AI Model Downloading. arXiv:2404.14204 [cs]. May 2024. DOI: 10.48550/arXiv.2404.14204.
- Feng Ye et al. “Intelligent hierarchical NOMA-based network slicing in cell-free RAN for 6G systems”. In: IEEE Transactions on Wireless Communications 23. 5 (2023), pp. 4724–4737
- Hui Huang, Qiang Ye, and Yitong Zhou. “6G-empowered offloading for realtime applications in multi-access edge computing”. In: IEEE Transactions on Network Science and Engineering 10. 3 (2023), pp. 1311–1325. ISSN: 2327-4697. DOI: 10.1109/TNSE.2022.3188921.
- Bradley Denby et al. “Kodan: Addressing the computational bottleneck in space”. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3. ASPLOS 2023. New York, NY, USA: Association for Computing Machinery, Mar. 2023, pp. 392–403. ISBN: 978-1-4503-9918-0.