On October 19, 2024, Ray Connect 2024 was held in Shanghai, gathering numerous internal and external experts who discussed Ray’s promising future as a next-generation cluster computing tool. Eight distinguished speakers represented major tech companies, including Alibaba, ByteDance, Tencent, Xiaohongshu, and eBay. The in-person event attracted over 100 experts from more than 30 enterprises and universities, while the online attendance exceeded 30,000 views.

Key Ray observations and highlights:

* Huawei use Ascend clusters and Ray’s framework to optimize data processing within autonomous driving, achieving enhanced efficiency in **real-time data handling and significant improvements in NPU utilizatio**n.
* Tencent has achieved **latency reduction and throughput growth** with PyIceberg and Ray , marking significant progress in distributed data handling.
* Xiaohongshu’s implementation of the **Ray Klein engine** has demonstrated remarkable success in handling resource-demand fluctuations, supporting **40+ business scenarios** and advancing towards an integrated streaming-batch engine.
* Alibaba group introduced Llumnix which is a revolutionary approach to large model scheduling by dynamically adapting to varying request and memory demands, **significantly reducing latency and improving system robustness** through Ray-based global scheduling and real-time migration.
* ByteDance has integrated Ray into its video generation model processing, overcoming scale and processing limitations through Actor-based data handling, achieving **enhanced fault tolerance and data lineage management**.
* Huawei Cloud’s Ray-powered infrastructure provides scalable, reliable services across various business scenarios, **supporting cutting-edge model deployment through serverless and resource-scaling features**.
* Alibaba has utilized Ray to streamline distributed video data processing, leveraging Actor-based task management and optimized resource usage, achieving efficiency gains and enabling **large-scale video model training**.
* eBay’s integration of Ray into its AI platform has bridged computational silos **between Python and Java** environments, enhancing resource utilization and reducing deployment complexity in generative AI model handling.

**Huawei Car BU:**

In the autonomous driving sector, the collection, mining, and processing of massive amounts of data demand high computational resources and efficient data handling mechanisms. Huawei faced significant challenges in achieving rapid data processing and efficient resource allocation to handle this data effectively. The main goal was to develop a solution that integrates Ascend’s hardware capabilities with Ray’s distributed computing to optimize large-scale data processing for autonomous driving. This integration aimed to enable rapid, scalable deployment and high-throughput processing, enhancing overall performance in a high-demand, real-time environment.

**Why Ray:**

Key challenges that led to the adoption of Ray alongside Ascend included:

**High Data Volume and Processing Demand:** Autonomous driving generates a vast amount of data, which requires intensive processing power and efficient scheduling to meet real-time demands. Traditional processing solutions couldn’t keep up with the required deployment speed and elasticity needed for rapid data ingestion and handling.

**Need for Optimized Resource Utilization:** Maximizing the use of NPU (Neural Processing Unit) resources was essential to improve processing throughput. Without a tailored mechanism, NPU utilization remained suboptimal, limiting the system’s potential throughput and efficiency.

**Complex Scheduling Requirements:** The varied nature of tasks in autonomous driving data workflows requires function-level scheduling for precise resource allocation, along with caching mechanisms to minimize latency in data access and processing.

**What they achieve with Ray:**

**Increased NPU Utilization:** With function-level scheduling and optimized resource allocation through Ray Data’s back-pressure mechanism, Huawei achieved a 4x increase in NPU utilization, ensuring higher throughput for data-intensive tasks. **Throughput Boost:** The system’s throughput was enhanced by 2.5x, enabling Huawei to handle greater volumes of data with improved processing speed, meeting the stringent demands of real-time autonomous driving data workflows.

**Efficient Resource Management and Scalability:** The combined use of Ascend and Ray’s elastic clustering allowed Huawei to deploy at a large scale with minute-level setup and second-level elasticity, optimizing cluster resources dynamically to meet changing workload requirements.

**Future Plans:** Huawei plans to further refine its data processing pipeline by enhancing Ray’s function-level scheduling capabilities, improving caching mechanisms, and exploring advanced profiling for even more precise resource allocation.

**Tencent**

Tencent is addressing the shift from traditional offline lake-warehouse architectures (often T+1 latency) to real-time data warehouses with minute-level responsiveness. With the convergence of Data and AI, the demand for rapid, scalable, and heterogeneous compute handling has reached new heights. The primary goal of Tencent’s Data+AI initiative is to achieve seamless integration of real-time data processing with AI functionalities. This includes overcoming inherent limitations in traditional architectures where real-time and offline compute integration can significantly strain systems, increase latency, and limit flexibility in AI model applications.

**Why Ray**:

1. **Python and Java Integration Challenges:** Traditional big data architectures at Tencent rely heavily on Java-based databases, limiting Python's interactivity and flexibility. Additionally, there’s a lack of support for fine-grained, heterogeneous compute scheduling, which hinders resource efficiency and data handling.
2. **Single-node Constraint in PyIceberg:** Tencent had adopted PyIceberg for its purely Python-based interface, eliminating dependency on JVM. However, PyIceberg initially lacked distributed support, limiting it to single-machine usage and impacting its scalability for large-scale data requirements.
3. **GPU Utilization and Data Processing Bottlenecks:** With no parallel processing and GPU support within the PyIceberg framework, Tencent faced high system latency and memory peaks, significantly limiting throughput and the effectiveness of GPU resource usage.

**What they achieve with Ray:**

Tencent’s integration of PyIceberg with Ray has led to substantial improvements in both data processing and AI application efficiency:

* **Throughput Enhancement:** By implementing Ray for parallel GPU processing, Tencent has achieved a 200x increase in throughput, enabling faster data processing and analysis.
* **Latency and Memory Optimization:** The adaptive data partitioning reduced system load times by 26%, while memory consumption was fine-tuned to optimize GPU utilization, now reaching over 80% efficiency, compared to previous 30% usage. New Use Cases: Interactive Data Analysis on Ray: Implementing Ray Data for Iceberg task handling resulted in a performance boost of approximately 3x over traditional Spark processing.
* **Causal Inference Applications:** Distributed causal inference models on Ray have been developed, expanding analytical capabilities and speed for AI-driven insights.
* Future Plans: Tencent plans to further develop its multi-modal data support, enabling multi-tenant clusters and advanced caching to enhance both performance and data access efficiency. This includes deepening integration with Ray for comprehensive, scalable, and reliable big data analytics solutions in AI-centric applications.

**Xiaohongshu**

Xiaohongshu, a platform with significant variability in resource demand, faces unique challenges in handling streaming and batch data workloads. Existing solutions, like PyFlink, while effective in certain scenarios, fail to provide the flexibility and resource efficiency required for tidal workloads and complex inference tasks. The goal of this initiative was to design a unified engine, Klein, capable of handling both streaming and batch inference workflows efficiently, with optimal resource utilization in tidal environments. This approach would address the existing shortcomings of resource constraints and non-unified execution across workflows, creating a highly adaptable and scalable solution.

**Why Ray:**

**Key challenges and reasons for adopting Ray include:**

**Diffuculity to Handle Resource Tides with PyFlink**: PyFlink, though effective for batch processing, lacked the elasticity to efficiently managevariations in resource demand, resulting in resource underutilization during off-peak hours and overloading during peaks.

**Lack of GPU Support and Performance Limitations:** Traditional batch processing platforms, primarily JVM-based, often fall short in supporting GPU-accelerated workflows. PyFlink’s performance in Python-dependent workflows also showed significant limitations, especially when handling high-throughput, low-latency demands in GPU-intensive tasks.

**Need for Streamlined Execution Across Tasks:** Managing disparate workflows with separate systems resulted in increased operational complexity and the need for extensive resource and process management. Klein was designed to consolidate streaming and batch processing within a single framework, leveraging Ray’s architecture to unify task execution.

**What they achieve with Ray:**

Xiaohongshu’s transition from PyFlink to the Ray-powered Klein engine has yielded substantial improvements in resource management, performance, and operational efficiency:

**Implementation Across 40+ Scenarios:** Klein has been successfully deployed across more than 40 business use cases, demonstrating its versatility and adaptability in various real-world applications on Xiaohongshu’s platform.

**Improved Resource Utilization and Flexibility:** With Klein’s dynamic resource allocation, Xiaohongshu achieved optimal utilization, ensuring high availability of compute power during peak times while reducing costs during off-peak periods. **Seamless Task Integration and Enhanced Processing Speed:** The unified engine improved task handling efficiency, allowing for more streamlined processing and faster task execution across streaming and batch workloads.

**Future Plans:** Xiaohongshu aims to further enhance the real-time capabilities of Klein and is exploring opportunities for its open-source release. This includes continued refinements in streaming-batch integration and optimizations for GPU-accelerated workloads to meet evolving demand.

**Alibaba**

Alibaba Cloud developed Llumnix, a dynamic scheduling system, to address these issues and optimize the efficiency of LLM inference. Traditional inference scheduling struggles to manage LLM-specific challenges, such as varying request lengths, output unpredictability, and fluctuating memory (KV cache) needs. The goal of Llumnix is to provide a scalable, responsive scheduling solution capable of adapting to LLM workloads by balancing system load, managing memory fragmentation, and improving overall service performance.

**Why Ray:**

**Unpredictable Request Patterns in LLMs:** Due to varying input lengths and response complexity, traditional schedulers face difficulties in managing dynamic resource requirements, often leading to increased tail latency and inefficient memory use. **Memory Fragmentation:** The frequent and varied memory demands of LLMs can result in significant memory fragmentation. This fragmentation often exacerbates delays for longer requests, leading to inconsistent service quality.

**Difficulty in Managing Multi-Priority Requests:** Serving multiple applications requires the system to distinguish between high- and low-priority requests. Traditional schedulers lack the flexibility needed to allocate resources in line with priority levels, impacting the service level agreements (SLAs) for time-sensitive requests.

**What they achieve with Ray:**

Alibaba Cloud’s deployment of Llumnix using Ray has achieved impressive results, particularly in balancing complex, large-model serving workloads:

**Reduced Tail Latency with Real-Time Scheduling:** Llumnix’s load balancing has successfully lowered tail latency by dynamically adjusting resources based on request length and complexity. This has enhanced the responsiveness of LLM applications, meeting high-performance SLAs.

**Improved Memory Utilization:** By addressing memory fragmentation issues, Llumnix provides dedicated memory space for larger requests, achieving optimal memory allocation even under heavy loads. This has significantly improved the handling of KV cache requirements, ensuring smooth and uninterrupted service.

**Prioritized Request Management:** Llumnix’s ability to distinguish between priority levels allows the system to meet different SLAs across various applications, thus catering to high-priority tasks effectively without disrupting lower-priority ones. **Future Plans:** Alibaba Cloud plans to extend Llumnix’s capabilities to include enhanced multi-tenant support and further optimizations in real-time memory allocation. Efforts are underway to refine the global-local scheduling interactions, further minimizing latency and boosting service efficiency across distributed models.

**ByteDance**

ByteDance aims to develop sophisticated AI models capable of generating realistic, imaginative outputs, such as video generation from text prompts. Multimodal models require vast amounts of video and other data types, and processing this data at scale presents significant challenges. The goal of ByteDance’s initiative was to create a unified, scalable data processing pipeline capable of handling the unique demands of multimodal model training, including efficient GPU/CPU resource allocation, robust data lineage tracking, and minimized latency during large-scale data transformations.

**Why Ray:**

**Key challenges leading to the adoption of Ray included:** High Volume of Diverse Data and Processing Complexity: ByteDance’s data pipeline requires processing a vast amount of video data, often in real-time, which strains existing systems with high data transmission and task management demands.

**Lack of Fault Tolerance and Scalability in Traditional Pipelines:** Initially, ByteDance’s pipeline relied on independent actors for each task (download, process, upload), which increased serialization load, caused disk overflow, and lacked built-in fault tolerance. These limitations hindered scalability and made recovery from failures challenging.

**Need for Unified Data Flow and Lineage Tracking:** Managing complex video data workflows across multiple stages required tracking data lineage to ensure data integrity and enhance fault tolerance, which was difficult to achieve with existing single-stage or non-distributed systems.

**What they achieve with Ray:**

ByteDance has developed a robust multimodal data pipeline with significant efficiency gains and enhanced fault tolerance:

**Reduced Processing Complexity and Latency:** Through task consolidation, ByteDance minimized serialization overhead and eliminated disk overflow issues, leading to faster and more streamlined data processing workflows.

**Enhanced Fault Tolerance with Data Lineage Tracking:** Data lineage tracking at the operator level has enabled ByteDance to achieve better fault recovery and ensure data consistency across various stages of the pipeline.

**Optimized Resource Utilization with Video Packing:** The pipeline now utilizes video packing, where multiple segments are grouped to avoid scheduling bottlenecks caused by large volumes of small files. This innovation has improved scheduling efficiency and reduced resource fragmentation.

**Future Plans:** ByteDance aims to enhance multimodal support by extending Ray-based optimizations to include caching mechanisms and further reduce latency. Additionally, plans are underway to make the pipeline more accessible through YAML-based task configurations, facilitating wider adoption across new AI-driven applications.

**Huawei Clould**

Huawei Cloud has been developing a high-reliability, intelligent cloud service framework to handle these demands. This initiative centers around providing flexible, scalable, and highly reliable services that integrate tightly with large-scale data and AI applications. The primary goal of Huawei Cloud’s Ray-powered services is to ensure robust performance and adaptability across multiple business scenarios, ranging from general cloud hosting to specific large language model (LLM) inference services. This framework leverages Ray to facilitate seamless scalability, fault tolerance, and high-performance task execution.

**Why Ray:**

Huawei Cloud’s adoption of Ray was driven by several specific challenges:

**Need for Reliable, Scalable Infrastructure for LLM and AI Workloads:** With the demand for LLM inferencing and real-time data processing on the rise, Huawei required a system capable of rapid scaling and automatic resource allocation to meet varying workload demands without sacrificing reliability.

**Challenges with Fault Tolerance and Resilience:** Traditional cloud services often face challenges in maintaining high availability and resilience, particularly under heavy loads and during fault recovery scenarios. Huawei aimed to design a system with built-in fault tolerance at multiple levels, from individual processes to full clusters. **Requirement for Unified Multi-Tenant Support:** To serve multiple clients efficiently, Huawei Cloud needed a solution that could support isolated workloads within a single infrastructure while providing real-time performance and reliability guarantees.

**What they achieve with Ray:**

By implementing Ray, Huawei Cloud has successfully launched a scalable, reliable cloud service infrastructure optimized for AI and data intelligence:

**High Availability for Diverse AI Models:** Huawei Cloud’s Ray-powered platform supports various models, including LLaMA, Qwen, GLM, and Huawei’s own Pangu, providing a versatile environment for LLM inference with flexible billing and automatic scaling.

**Enhanced Reliability with Multi-Layer Fault Tolerance:** The Ray-based infrastructure includes advanced fault tolerance mechanisms at the Pod, container, and process levels. This setup allows Huawei Cloud to provide consistent, uninterrupted service even under peak load conditions.

**Optimized Resource Management with Serverless Flexibility:** Huawei’s infrastructure enables node-level scaling and gray deployment upgrades, giving enterprises a responsive, adaptable system that scales resources according to real-time requirements.

**Future Plans:** Huawei Cloud intends to expand its Ray-powered cloud services with enhanced virtual cluster support, further integrating intelligent scheduling and monitoring. Additionally, Huawei is exploring more fine-grained security controls and resource allocation techniques to better serve diverse, multi-tenant environments.

Alibaba Tongyi Lab

At Alibaba’s Tongyi Lab, the need for an efficient, scalable video processing pipeline is critical, particularly for tasks requiring large volumes of video data to be processed in distributed environments. Traditional single-node processing setups could not meet the computational demands, prompting the need for a more robust, distributed solution. The primary objective was to build a scalable infrastructure capable of processing extensive video datasets quickly and effectively, supporting Alibaba’s video-centric AI initiatives. Ray’s architecture was selected to enable distributed video processing, optimizing CPU/GPU resources and minimizing latency across a multi-node setup.

**Why Ray:**

**High Computational Demand for Large-Scale Video Data:** Video processing tasks, especially for AI training, require high CPU and GPU resources that are challenging to meet with a single-node setup. Alibaba needed a solution that could distribute tasks effectively across multiple nodes.

**Complexity in Managing Real-Time, High-Volume Video Tasks:** Given the continuous nature of video streaming and processing, the solution needed to handle dynamic task distribution, real-time resource allocation, and maintain high throughput without increasing latency.

**Need for Simplified Task Management:** Managing video data processing through conventional frameworks required complex orchestration and high management overhead, which impacted efficiency and slowed down the overall workflow.

**What they achieve with Ray:**

Through Ray, Alibaba’s Tongyi Lab has achieved significant advancements in distributed video data processing, with tangible benefits in scalability, efficiency, and ease of use:

**Enhanced Resource Utilization and System Flexibility:** By leveraging Ray’s distributed task handling, Alibaba maximized node utilization, achieving close to 100% resource efficiency and significantly reducing error rates during processing. **Streamlined Task Submission and Execution:** Implementing a web-based console for Ray allowed seamless task submissions, minimizing the learning curve for users and streamlining task initiation across the distributed system.

**Efficiency Gains with Batch Data Processing:** Using message queues for task handling, OSS (Object Storage Service) for data storage, and batched data transmission improved throughput by 20x, accelerating video data handling and reducing processing time.

**Future Plans:** Alibaba’s Tongyi Lab plans to continue optimizing the pipeline for distributed video processing, focusing on enhancing data transfer efficiency and refining message queue management. They aim to explore deeper integration with Ray’s library ecosystem for additional performance improvements.

**eBay**

As eBay’s AI capabilities grow to support more advanced models, particularly in generative AI, the need for a unified, scalable platform to handle complex data workflows has become critical. eBay's AI environment traditionally involves both Python (for model processing) and Java (for database and request handling), which has led to complex integrations, resource inefficiencies, and high deployment overhead. The goal of eBay’s next-generation AI platform is to streamline model deployment, optimize resource allocation, and improve fault tolerance, while unifying Python and Java workflows within a single, scalable framework. Ray’s distributed architecture was chosen to enable these capabilities and to drive higher performance for large AI models and their associated data handling requirements.

Why Ray:

**Complex Integration of Python and Java Workflows**: eBay’s models require Python for AI processing, but backend systems are heavily Java-based. This mismatch created significant friction in model deployment and data handling, resulting in resource underutilization and elevated maintenance complexity.

**Resource Inefficiencies in Model Deployment**: Balancing CPU and GPU workloads proved challenging with existing infrastructure, often leading to resource bottlenecks and limited scalability for AI tasks that require high compute resources. **High Fault Tolerance and Reliability Demands: G**iven the critical nature of AI-driven tasks in eBay’s marketplace, maintaining reliable, fault-tolerant systems is essential. Previous setups struggled with load balancing and recovery, especially under peak demand, which affected model performance and response times.

**What they achieve with Ray:**

**Optimized Resource Utilization:** With Ray, eBay has successfully separated CPU- and GPU-heavy tasks, leading to more efficient resource usage and streamlined deployment. This has significantly reduced resource bottlenecks and improved system responsiveness.

**Improved Fault Tolerance and Reliability:** Ray’s robust fault-tolerance capabilities have enhanced the platform’s reliability, allowing eBay to maintain consistent performance across distributed nodes and reducing downtime in the event of node failures.

**Unified Deployment Pipeline:** By utilizing Ray Serve in conjunction with pre-processing modules, eBay has created a streamlined deployment pipeline that integrates seamlessly with backend databases, improving model deployment efficiency and reducing operational overhead.

**Future Plans:** eBay plans to extend the platform’s capabilities by enhancing Ray’s integration with real-time inference, developing historical logging services, and incorporating additional security features. Expansions are also planned for Ray’s role in eBay’s LLM services and multi-modal AI applications.