

## Quantum Distributed Computing:

You may have heard about quantum distributed computing and the need for a standard middle-ware architecture that can ultimately lead to the quantum Internet. Many standards are being researched by institutes like the NIST, IETF, private research groups etc

Here is one new proposal called Pilot-Quantum, a Quantum-HPC Middleware for Resource, Workload and Task Management. More details in the paper. <https://www.sciencedirect.com/science/article/pii/S1574013725000231>

Quantum distributed computing is highly complex; moreover, interfacing with HPC is much more difficult, as the classical HPC needs to interface transduce across different signal types, data volumes, different clocking, different bus, protocols, exchange mechanisms, local and non-local entanglements and computations, various run times, decomposed circuits, unitaries etc. Also the limitations like no-cloning, less quantum memory, more absorption of signals, fragile qubits, less coherence, cooling needs, etc all need special attention. One interesting paper is: Distributed Quantum Computing with QMPI : <https://lnkd.in/gxGR22RU>

An extensive coverage of Quantum distributed computing is available here:

<https://lnkd.in/gFjZpxA2>

Sources: arXiv, ScienceDirect, My research

<https://lnkd.in/gzH4nktE>

///// Below are refinements and merging of the above papers used as reference only.

Hi, discussion related to distributed quantum computing. I will give some background

ChatGPT said:

Sure! Go ahead and provide the background. I'm ready to discuss distributed quantum computing in detail.

You said:

Attached Image and content from a paper: To facilitate the readers' understanding, this survey is structured according to a layered model, as depicted in Fig. 1, similar to the full-stack architecture presented by [27], the abstract model in [28], or the DQC simulator tools structure in [26]. The two lower layers in Fig. 1 encompass the hardware developments needed to implement a distributed quantum system and would be equivalent to the three lower layers of the classical OSI model. So, the physical layer refers to the mechanisms that allow two physically separated QPUs to be connected, while the network layer defines how to establish communication between multiple QPUs. Directly above this layer, we discuss advances in development tools that allow applications to be distributed and executed on a distributed quantum system, including partitioning, compilation, optimization, and mapping algorithms. Finally, in the uppermost layer, we address distributed algorithms. It is important to note that these layers are interdependent, with each layer influencing those immediately preceding and succeeding. For instance, the development of a compiler is influenced by the underlying hardware and provides support for different partitioning techniques in the application layer

ChatGPT said:

Thanks for sharing the image and background. This layered model for **Distributed Quantum Computing (DQC)** follows a structure similar to the classical OSI model, emphasizing the interdependence of each layer.

## Breakdown of the Layers:

1. **Physical Layer**
  - Handles quantum entanglement and quantum teleportation.
  - Supports the connection of physically separated Quantum Processing Units (QPU).
2. **Network Layer**
  - Defines how multiple QPUs communicate over a **Quantum LAN (QLAN)**.
  - Facilitates quantum networking protocols, including entanglement distribution.
3. **Development Layer**
  - Focuses on qubit mapping, error correction, and middleware for distributed execution.
  - Includes compilation, partitioning, and optimization strategies.
4. **Application Layer**
  - Implements distributed quantum algorithms like **Shor's Algorithm**, **Quantum Fourier Transform (QFT)**, and **Quantum Phase Estimation (QPE)**.
  - Supports domain-specific applications requiring distributed quantum computation.

This model highlights the complexity of DQC, as **each layer influences both its preceding and succeeding layers**. For example:

- The **compiler design** in the development layer depends on the hardware and determines how quantum circuits are partitioned in the application layer.
- The **network layer's ability to share entangled qubits** affects the performance of distributed quantum algorithms.

### Key Research Challenges:

- **Reliable Quantum Networking:** Efficient entanglement distribution and quantum error correction.
- **Scalability of Quantum Cloud Computing:** Managing multiple QPUs in a distributed fashion.
- **Hybrid Quantum-Classical Processing:** Leveraging classical resources for control, orchestration, and error mitigation.

Would you like to discuss a specific aspect, such as quantum networking protocols, compiler optimizations, or real-world implementations?

**You said:**

Entanglement lies at the heart of quantum communications, facilitating the distribution of quantum states encoding quantum information through a protocol known as quantum teleportation or teledata. Multiple teleportation variants exist, which are designed to either transmit data in one direction – quantum teleportation or teledata – but also bi-directional communication – entanglement swapping – and gate operation at a distance – gate teleportation or telegate. Furthermore, the basic two-node teleportation can be extended to multi-party distribution networks composed of many nodes. Some parties may either help the rest of the network in the quantum communication protocol – assisted teleportation –, or the quantum information may be imperfectly broadcast from one sender to the rest – quantum telecloning

**ChatGPT said:**

You're touching on a fundamental aspect of **quantum communications**—entanglement-based teleportation protocols, which are essential for **distributed quantum computing (DQC)** and **Quantum Internet architectures**.

Let's break down these different teleportation mechanisms:

---

## 1. Quantum Teleportation (Teledata)

- **One-way transmission of quantum information.**
  - A sender (Alice) wants to send a **quantum state** to a receiver (Bob).
  - **Entanglement + Classical Communication** enables the exact transfer of a qubit state without physically sending it.
  - Used in **distributed quantum computing** to transfer intermediate computational states.
- 

## 2. Entanglement Swapping (Bi-Directional Communication)

- Allows **two unentangled qubits** to become entangled via an intermediary node.
- Intermediate nodes measure their entangled pairs, creating a new entangled connection.
- **Used for extending quantum networks**, where direct entanglement is not possible due to distance or noise.
- Enables the creation of long-distance quantum communication channels.

### □ Example:

- If **Alice–Charlie** and **Charlie–Bob** share entangled pairs,
  - Charlie performs a **Bell-state measurement**,
  - Alice and Bob now share entanglement directly, even though they never interacted.
- 

## 3. Gate Teleportation (Telegate)

- **Remote execution of quantum gates** using entanglement.
- Instead of sending data, Alice and Bob share entanglement, and Bob can apply a **quantum gate** to Alice's qubit **remotely**.
- Critical for **distributed quantum computing**, where operations need to be performed across separate quantum processors.

### □ Use Case:

- Performing **CNOT gates** across different quantum processors without physically moving qubits.
- 

## 4. Assisted Teleportation

- **Helper nodes** in a network assist in teleportation.
  - These nodes don't receive or send data directly but help facilitate entanglement distribution.
  - Used in **quantum networks to improve reliability** and efficiency.
- 

## 5. Quantum Telecloning (Imperfect Multi-Party Distribution)

- A single quantum state is **imperfectly copied** and distributed to multiple receivers.

- Unlike classical copying, exact quantum cloning **violates the no-cloning theorem**, so telecloning allows approximate copies.
- Used in **quantum broadcasting** and **multi-party quantum secret sharing**.

#### □ Use Case:

- A **quantum cloud server** wants to distribute quantum states to multiple clients in a quantum network.

## Relevance to Distributed Quantum Computing

In a **DQC model**, these teleportation techniques play critical roles:

- **Quantum teleportation** enables **state transfer** between remote QPUs.
- **Entanglement swapping** extends **multi-node quantum networks**.
- **Gate teleportation** allows for **secure execution of quantum operations** across distributed processors.

Would you like to discuss how to implement these in a quantum network, such as through **Quantum Repeaters, QLANs, or Quantum Internet Protocols**? □

You said:

More information. Attached is Quantum stack image. The conceptual quantum-HPC middleware architecture [20] consists of four distinct layers, denoted as L1, L2, L3, L4: L4 is the workflow layer, which encapsulates application semantics and logical dependencies between the workflow stages and offers a high-level abstraction for developing the quantum-HPC workflows. The workload layer L3 manages the workload, a set of interdependent, quantum, and classical tasks emitted by L4. It optimizes the workload execution by partitioning tasks and assigning them to the available resources utilizing parallelism (e.g., ensemble and task parallelism). The L2 task layer performs the operational management of task execution on a single resource. Any runtime operations, i.e., task scheduling, assignment, execution, and monitoring, are managed at L2. Finally, the resource layer L1 describes how a system abstracts and manages heterogeneous classical (e. g., HPC and cloud systems, including accelerators) and quantum resources (e. g., gate-based and analog QPUs). L1 layer provides the capability to exploit system-level tools, e. g., GPU acceleration and QPU-target optimization are essential for efficient quantum-classical integration

ChatGPT said:

Your provided **Quantum-HPC Middleware Architecture** describes a well-structured **multi-layered quantum software stack**, integrating quantum and classical resources efficiently. Let's break it down:

## Quantum Software Stack: Layered Overview

1. **L4 – Workflow Layer**
  - Encapsulates **application semantics** and **logical dependencies** between workflow stages.
  - Provides **high-level abstraction** for building **quantum-HPC workflows**.
  - Example: **Quantum application frameworks, algorithm libraries**.
2. **L3 – Workload Layer**
  - Manages **interdependent classical and quantum tasks** from L4.
  - Optimizes execution via **task partitioning** and **parallelism strategies**.

- Example: **Task scheduling for hybrid workloads**, managing QML (Quantum Machine Learning) batching & autograd.
  - 3. **L2 – Task Layer**
    - Handles **operational management** of individual tasks on assigned resources.
    - Manages **task scheduling, execution, and monitoring**.
    - Example: **Quantum transpilation, hybrid execution runtimes** (e.g., **Qiskit Runtime, Braket Hybrid Jobs**).
  - 4. **L1 – Resource Layer**
    - Abstracts and manages **heterogeneous computing resources** (Quantum + Classical).
    - Provides **system-level runtimes** and **hardware optimization** for **GPUs, CPUs, and QPUs**.
    - Example: **CUDA-Q, CuQuantum, MPI, Cloud-based QPU acceleration**.
- 

## Key Takeaways for Quantum-HPC Integration

- **Hybrid Execution:** Seamless coordination between **quantum tasks (QPUs)** and **classical acceleration (GPUs, CPUs)**.
- **Workload Optimization:** **Parallelism-aware scheduling** (task-level, ensemble-level).
- **Resource Efficiency:** System-level tools like **CUDA-Q, MPI, HPC job schedulers** ensure performance.

Would you like to discuss **how this stack maps to specific Quantum-HPC applications**, such as **quantum chemistry, AI, or cryptography**? ☐

You said:

Figure 2 shows Pilot-Quantum’s architecture. The Pilot-Job system [22], [23] acts as an intermediary between applications, higher-level systems (e.g., workflow management), and local resource management systems (e.g., Slurm on HPC). It consists of a Pilot-Manager and multiple Pilots. The Pilot-Manager orchestrates resources by provisioning, managing, and releasing them for workload execution. It optimizes task assignment across pilots for efficient time-to-completion and system utilization. The Pilot-Manager provides extensible resource plugins to integrate various resources. The Pilot Abstraction’s multi-level scheduling suits quantum-HPC environments for two reasons: (i) heterogeneity and complexity of quantum and classical applications complicate static workload and resource assignment, and (ii) early quantum era co-design of applications and hardware requires integrated application-level resource management. While small applications can use hard-wired integration, larger hybrid systems need more adaptive resource management. Hence, hierarchical resource management capabilities are crucial for developing and operating quantum-enabled workflow systems efficiently. Next, we present each layer of the system. 1) L4 Workflow Layer and Abstractions: Pilot-Quantum provides two abstractions: (i) resource management, enabling applications to allocate appropriate classical and quantum resources, and (ii) workload and task management, facilitating efficient workload execution on these resources. For (i), applications must define required resources in a `pilot_description` and submit it to Pilot-Quantum, which then acquires the resources. For (ii), applications must decompose their problems into a set of independently executable tasks, collectively forming the workload. Tasks can encapsulate units of quantum computations (e.g., circuit executions) and classical computations (e.g., pre- and post-processing). While early quantum applications require high control over problem decomposition and workload, Pilot-Quantum is expected to interface with higher-level workflow systems and application libraries. Tasks are defined using a `task_description` and submitted via the `submit_task` API. Pilot-Quantum also integrates with native abstractions of different frameworks, e.g., Dask and Ray. Tasks can be assigned: (i) directly to a specific pilot or (ii) to a group of pilots. For (ii), the pilot manager assigns tasks to pilots, optimizing throughput by load balancing across all pilots. Please, refer to [78] for API details. 2) L3/L2 Workload and Task Layers: After receiving the `pilot_description`, the Pilot-Manager allocates requested resources via the local resource management system,

(steps 1-3 in Figure 2). In HPC environments, a placeholder job, which starts the Pilot-Agent is queued via Slurm. Once active, the Pilot Agent manages the allocated resources and executes tasks either directly or through external execution engines. External execution engines are supported via plugins; currently, Pilot-Quantum supports Dask [79] and Ray [68]. Further, it integrates with low-level runtimes (e. g., CUDA and MPI). Quantum hardware access is handled via classical tasks that utilize, e. g., Qiskit Providers or PennyLane plugins

ChatGPT said:

Thanks for sharing the **Pilot-Quantum Architecture**! This system introduces a **Pilot-Job abstraction** to efficiently manage quantum and classical resources in **heterogeneous quantum-HPC environments**.

---

## □ Key Components of the Pilot-Quantum Architecture

### 1 □ Pilot-Manager (L3 – Workload Layer)

- Orchestrates resource provisioning and workload execution.
- Optimizes task assignment across pilots using schedulers and plugins.
- Supports heterogeneous resources (QPUs, GPUs, CPUs) via extensible resource plugins.

### 2 □ Pilot (L2 – Task Layer)

- Runs within the **Resource Management System** (e.g., Slurm, Kubernetes).
- Uses **Ray/Dask execution engines** for distributed workloads.
- Manages tasks assigned by the **Pilot-Manager**.
- Can handle **hybrid workloads** (quantum + classical).

### 3 □ Resource Management System (L1 – Resource Layer)

- Provides access to **Quantum (QPU), HPC (GPU/CPU), and Cloud resources**.
  - Abstracts **hardware heterogeneity** for workload execution.
- 

## □ Why is the Pilot-Job Model Crucial for Quantum-HPC?

### □ Adaptive Resource Management

- Unlike **static workload assignment**, it dynamically schedules quantum & classical tasks.

### □ Scalability & Flexibility

- Supports **small applications (manual integration)** and **large systems (hierarchical management)**.

### □ Integration with Quantum & HPC Frameworks

- Supports **Dask, Ray, CUDA, MPI** for parallel task execution.
  - Interfaces with **Qiskit, PennyLane, Braket** for quantum workloads.
-

## ❑ How Tasks Flow in Pilot-Quantum

- ❑ **Step 1:** Application submits **Pilot Description (pilot\_description)** to **Pilot-Manager**.
  - ❑ **Step 2:** Pilot-Manager provisions resources via **Slurm, Cloud, or other schedulers**.
  - ❑ **Step 3:** A **Pilot-Agent** is launched on allocated resources.
  - ❑ **Step 4:** Application submits **Task Description (task\_description)**.
  - ❑ **Step 5:** Pilot-Manager **assigns tasks to pilots**, balancing workloads.
  - ❑ **Step 6:** **Pilots execute tasks**, either locally or via **Ray/Dask**.
- 

## ❑ Future Outlook: Enhancing Quantum-HPC Interoperability

- ❑ **Advanced Scheduling & Load Balancing:** Optimizing QPU & GPU utilization.
- ❑ **Quantum Workflow Integration:** Extending compatibility with **high-level quantum libraries**.
- ❑ **Hybrid Classical-Quantum Execution Models:** Automating quantum circuit pre/post-processing.

Would you like to explore **real-world implementations** of Pilot-Quantum or **specific optimization strategies** for distributed quantum workloads? ❑

**You said:**

Using this background can you design a new model that merges all the 3 given images and give proper shape to this model. The idea is to pick key merit from each image and build a model that works well as the Quantum HPC Middleware that can later be used as a base for the quantum Internet

**ChatGPT said:**

I'll design a **comprehensive Quantum-HPC Middleware Architecture** that integrates the **three models** you provided:

1. **Layered Distributed Quantum Computing Model**
    - Strength: **End-to-end stack** (hardware to application).
    - Key Elements: **Quantum teleportation, entanglement, and partitioning**.
  2. **Quantum Software Stack**
    - Strength: **Hybrid resource management** for quantum-classical workflows.
    - Key Elements: **Workload scheduling, optimization, hybrid runtimes**.
  3. **Pilot-Quantum Architecture**
    - Strength: **Dynamic resource allocation** with **Pilot-Job orchestration**.
    - Key Elements: **Decentralized pilots, multi-level scheduling, and workload balancing**.
- 

## ❑ New Model: Unified Quantum-HPC Middleware for the Quantum Internet

The proposed model will have **four major layers** while integrating:

- ❑ **Distributed quantum computing (DQC) principles**
  - ❑ **Quantum-classical hybrid execution**
  - ❑ **Pilot-based dynamic resource orchestration**
  - ❑ **Interoperability for future Quantum Internet deployment**
-



## ❑ Layered Structure of the Unified Model

### 1❑❑ Quantum-HPC Physical & Network Layer (L1)

- **Quantum & Classical Hardware:** QPUs, GPUs, CPUs, TPUs, Cloud.
- **Quantum Network:** Entanglement-based QLANs, quantum teleportation.
- **Resource Managers:** SLURM, HPC clusters, cloud-based quantum instances.

### 2❑❑ Quantum-HPC Middleware & Execution Layer (L2)

- **Pilot-Manager System:** Multi-level scheduling (Pilots + Workload manager).
- **Hybrid Runtimes:** Dask, Ray, MPI, CUDA-Q, CuQuantum, PennyLane.
- **Quantum Workflow Orchestration:** Enables co-execution of quantum-classical tasks.
- **Optimization & Compilation:** Adaptive quantum transpilation.

### 3❑❑ Workload & Task Management Layer (L3)

- **Quantum-Classical Task Partitioning:** Decides what runs on QPU vs. CPU/GPU.
- **Workload Scheduling:** Balancing jobs using pilots and workload managers.
- **Autonomous Resource Allocation:** Pilots dynamically provision and release resources.

### 4❑❑ Application & Quantum Internet Layer (L4)

- **Quantum Algorithms & Libraries:** Shor's algorithm, VQE, QML, QAOA, etc.
- **Quantum Internet Integration:** Secure quantum key distribution (QKD), quantum cloud services.
- **Cross-Platform Quantum Interoperability:** Qiskit, Braket, Cirq, PennyLane support.

---

## ❑ Key Features of This Model

- ✓ **Dynamic Hybrid Workload Execution** → Balances quantum & classical tasks.
  - ✓ **Quantum Entanglement Networking** → Future-ready for Quantum Internet.
  - ✓ **Pilot-Based Multi-Level Scheduling** → Optimized resource sharing.
  - ✓ **Modular & Extensible** → Supports new quantum algorithms & cloud platforms.
- 

## ❑ Next Steps

Would you like a **visual diagram** of this architecture? I can create a **layered flowchart** that illustrates how all components interact. ❑

You said:

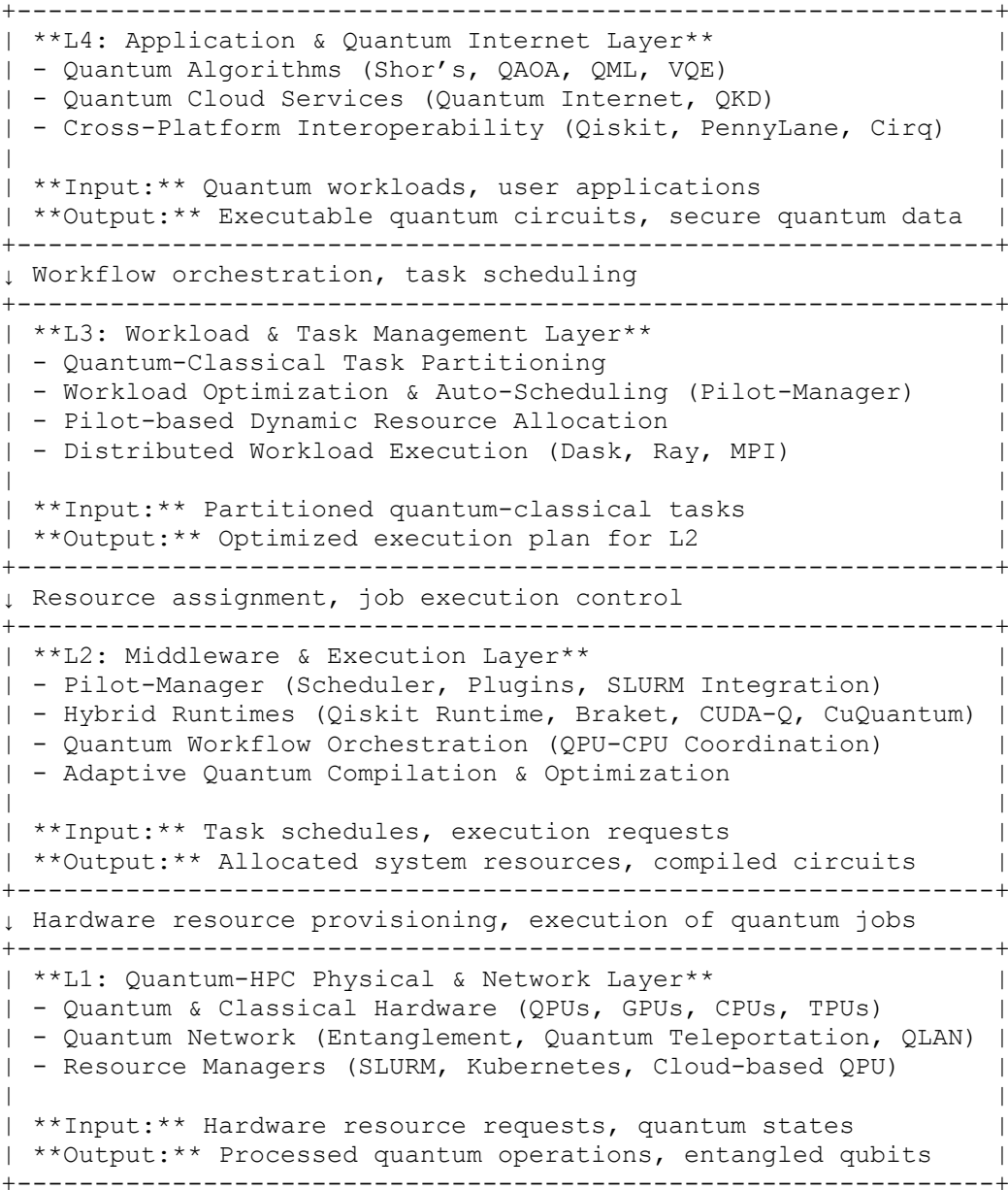
Please provide the text based layered block diagram and flowchart for process along with functions and input, outputs

ChatGPT said:

## ❑ Text-Based Layered Block Diagram for Unified Quantum-HPC Middleware

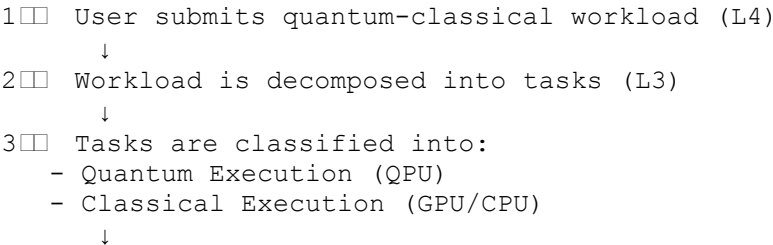


(A structured integration of **Distributed Quantum Computing, Quantum Software Stack, and Pilot-Quantum Architecture**)



□ **Flowchart: Process Flow of Quantum-HPC Middleware**

A step-by-step representation of how tasks are handled in this **integrated model**.



- 4 ☐ Pilot-Manager assigns resources dynamically (L2)
- ↓
- 5 ☐ Tasks are scheduled and queued using hybrid runtimes (Dask, Ray, MPI) (L2)
- ↓
- 6 ☐ Resource Manager provisions execution environments (L1)
- ↓
- 7 ☐ Quantum Network enables remote execution using entanglement (L1)
- ↓
- 8 ☐ Execution completes; results are sent back to user (L4)
- 

☐ **Functions, Inputs & Outputs for Each Layer**

| Layer   | Function   | Input                                | Output                                   |
|---|--|--------------------------------------|--|
| <b>L1: Quantum-HPC Physical &amp; Network Layer</b> | Executes quantum operations, maintains entanglement links              | Resource requests, quantum states    | Processed operations, entangled qubits   |
| <b>L2: Middleware &amp; Execution Layer</b>         | Schedules and executes quantum-classical workloads                     | Task schedules, execution requests   | Allocated resources, compiled circuits   |
| <b>L3: Workload &amp; Task Management Layer</b>     | Optimizes workload execution, partitions quantum-classical tasks       | Partitioned tasks                    | Optimized execution plan                 |
| <b>L4: Application &amp; Quantum Internet Layer</b> | Provides high-level quantum applications and Quantum Internet services | Quantum workloads, user applications | Executable circuits, secure quantum data |

---

☐ **Key Benefits of This Model**

- ☐ **Unified Quantum-Classical Execution:** Efficient integration of QPUs, GPUs, and CPUs.
- ☐ **Scalable Workload Management:** Pilot-based scheduling for **large hybrid workloads**.
- ☐ **Future-Ready Quantum Internet:** Incorporates **QLANs, teleportation, and entanglement-based networking**.

Would you like any refinements or additions to this model? ☐

You said:

Thanks a lot. This is good for now. I will read the details and get back to you. Good day !

ChatGPT said:

You're very welcome! ☐ I'm glad you found it useful. Take your time to go through the details, and feel free to reach out whenever you need further clarifications or refinements.

Wishing you a great day ahead! ☐☐