A Detailed Venn Responsibility

Venn delegates responsibilities such as device selection, device fault tolerance, and privacy protection to individual CL jobs. Device failures are both inevitable and difficult to predict in CL. Rather than imposing a one-size-fits-all solution, Venn empowers CL jobs to take the reins on fault tolerance based on their specific workloads and objectives. Therefore, Venn offloads handling device fault tolerance to CL jobs, who can better detect and react to device failures (e.g., deciding the amount of overcommit (Bonawitz et al., 2019)). Similarly, Venn offers CL jobs the freedom to design their own device selectors (Lai et al., 2021b), where they can incorporate customized resource criteria into their requests. Venn also does not interfere with job-specific privacy solutions such as secure aggregation (Bonawitz et al., 2016; Huba et al., 2022) or differential privacy (Geyer et al., 2017; Xu et al., 2023).

B ILP Formulation of IRS

We now formulate the IRS that allocates resources to jobs under the constraints with the objective of minimizing the average scheduling delay. Assume we have devices $\mathbb{S} = \{s_1, s_2, ..., s_q\}$ continuously arriving at times $\{t_i, t_2, ..., t_q\}$. There are m jobs $\mathbb{J} = \{J_1, J_2, ..., J_m\}$ with their resource demands $\mathbb{D} = \{D_1, D_2, ..., D_m\}$. Let e_{ij} be a binary variable in the eligibility matrix, which is set to 1 if device i is eligible to job j, and 0 otherwise. Let x_{ij} be a binary variable of resource allocation, which is 1 if device i is assigned to job j, and 0 otherwise.

We have to follow these constraints during the resource allocation:

$$\sum_{j=1}^{m} x_{ij} \leq 1, \forall i \in [1, q]$$

$$\sum_{j=1}^{m} x_{ij} \times e_{ij} \leq 1, \forall i \in [1, q]$$

$$\sum_{i=0}^{q} x_{ij} = D_j, \forall j \in [1, m]$$

Therefore, the scheduling delay of each job is determined by the time it acquires the last needed device, i.e., $T_j = \max_i(x_{ij} \times t_i)$ under these constraints. The overall objective can then be expressed as:

$$\min \frac{\sum_{j=1}^{m} T_j}{m}$$

C Theoretical Insight to the Heuristic of IRS

Lemma 1. *Given a diverse set of CL jobs with one round request, if jobs are scheduled optimally in terms of the average JCT, first within each job group and then across job groups, the resulting average JCT is optimal.*

Proof. Let us assume there is an optimal scheduling algorithm that optimizes the average JCT within each group, and there is an optimal scheduling algorithm which decides how to merge the job order across job groups to minimize the average JCT. Since the second step is assumed to provide optimal average JCT based on the previous within group job order, we only need to prove the global optimal schedule follows the order generated by the within job group step.

Venn employs smallest remaining job demand first algorithm within each job group. Since prioritizing jobs with smaller remaining resource demands has been shown to be effective in similar scheduling problems (Garey et al., 1976), we skip the proof that the scheduling algorithm within job group gives the local optimal average JCT for each group.

We prove the rest by contradiction. Assume that there exists an optimal schedule S that does not follow the order given by each job group. In this assumed optimal schedule S, let us say there are two jobs J_A and J_B in the same group such that J_A comes after J_B, but J_A has fewer resource requirements than J_B. Let us swap J_A and J_B to create a new schedule S'. Since J_A has fewer resource demand, the average JCT of S' will be less than that in S. This contradicts our original assumption that S is an optimal schedule, as we've found a schedule S' with a lower average JCT. Therefore, the assumption is false, and the order given by each job group (sorted by resource demands) must be part of the optimal schedule. If we have an optimal scheduling across job groups, the overall average JCT will be optimal.

□

D Effectiveness of Venn Scheduling Heuristic

To illustrate the effectiveness of Venn's approach, we start with proving Lemma 2, which considers a simplified case involving only two job groups with arbitrary resource contention patterns. Through mathematical proof, we can demonstrate that our algorithm achieves the optimal solution under this setting.

Lemma 2. *Given two job groups with arbitrary resource contention patterns, the scheduling plan generated by Venn as in Algorithm 1 is capable of minimizing the average scheduling delay, if a future resource allocation plan is set.*

To better prove the Lemma, we introduce a new representation of the scheduling problem in a more scalable way. Firstly, as depicted in Figure 15a, we represent the two job groups by two distinct sets of squares, where the area of each square corresponds to the size of the request demand

for that job.

Secondly, to visualize the temporal dynamics of resource allocation, we refer to Figure 15c. For the sake of this example, let's assume a constant inflow of 100 devices per time unit. Within this set, 'x' devices possess memory $\geq$ 2GB, while all 100 devices have memory $\geq$ to 1GB. The y-axis is partitioned into two segments: the 0 to 'x' range signifies devices with memory exceeding 2GB, and the 'x' to 100 range represents devices with memory ranging between 1GB and 2GB.

Resource allocation over time is illustrated using rectangles, each indicating the job request to which devices are assigned. For instance, in the right subfigure of Figure 15c, devices in the 0 to 'x' memory range are allocated to job group B at time 0, while those in the 'x' to 100 range are allocated to job group A. This representation allows us to dynamically track resource allocation across different jobs over time.

Proof. As shown in Figure 15a, there are m_A requests that ask for devices with 1GB memory and m_B jobs that request devices with 2GB memory, resulting in two job groups A and B. The devices constantly check-in and execute one CL task, where 100% devices with memory size $\geq$ 1GB and x% of the devices have memory size $\geq$ 2GB. Note that, the proof is not limited to the contention pattern draw in Figure 15a, it can be generalized to job group with intersected resource contention and give the same conclusion.

Based on Algorithm 1, the first step is to sort these jobs within each job group by job size in ascending order (Figure 15b). In the second step, we generate an initial resource allocation for each job group by focusing on the job group with the scarcest resources. This results in an initial allocation plan that avoids resource sharing across job groups, setting the stage for subsequent cross-group allocations.

Based on the group-level initial allocation plan (left subfigure in Figure 15c), we need to determine the job order across groups, that is, to decide whether to prioritize jobs from Group A over Group B (right subfigure in Figure 15c) at current time in order to achieve a smaller average scheduling delay. In this case, we focus on determining the order of the first job with size l in Group A and calculate the queuing delay difference (Δt) if we prioritize the first job from Group A over Group B.

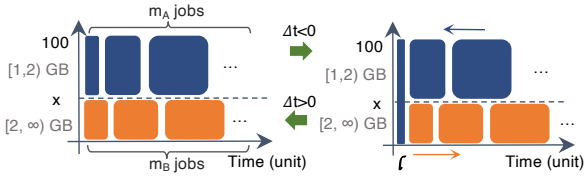
$$\Delta t = l * m'_B - (\frac{l}{1-x} - l) * m'_A$$

where m'_A, m'_B represents the number of remaining jobs whose queuing delay may be affected by this prioritization. Since the future resource allocation is set by the previous initial allocation or assumed to be given, m'_A, m'_B are feasible to get. We prioritize the first job from Group A only if $\Delta t < 0$, which gives $\frac{m'_A}{1-x} > \frac{m'_B}{x}$, otherwise we stick with the original plan. $\Delta t < 0$ is actually the prototype of the scheduling decision as in Algorithm 1 line 15.

□

(a) Resource supply and demand. There are 100% devices with memory size $\geq$ 1GB and x% of the devices have memory size $\geq$ 2GB. There are two job groups where Group A with m_A jobs requests for devices with memory size $\geq$ 1 GB and Group B with m_B jobs requests for devices with memory size $\geq$ 2 GB.

(b) Step 1: Sort within Job Group.



(c) Step 2: Schedule across Job Groups.

Figure 15. Venn scheduling algorithm.

By leveraging the conclusion of Lemma 2, Venn can further generalize to the scenario with more than two job groups with arbitrary resource contention patterns. Specifically, Venn greedily compares each pair of job groups (G_j, G_k) following the order. For each pair, Venn applies the logic proven in Lemma 2 to minimize the average scheduling delay between G_j and G_k.